

University of Southampton Research Repository ePrints Soton

Copyright © and Moral Rights for this thesis are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given e.g.

AUTHOR (year of submission) "Full thesis title", University of Southampton, name of the University School or Department, PhD Thesis, pagination

University of Southampton

**NUMERICAL SIMULATION AND CONTROL
OF A FLUID STRUCTURE INTERACTION
FOR A PLATE IN A TRANSVERSE FLOW**

Volume 1 of 1

Etienne Sourdille

Doctor of Philosophy

~

**School of Engineering Sciences
Aerodynamic and Flight Mechanics Group**

November 2006

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

SCHOOL OF ENGINEERING

AERODYNAMIC AND FLIGHT MECHANICS GROUP

Doctor of Philosophy

NUMERICAL SIMULATION AND CONTROL
OF A FLUID STRUCTURE INTERACTION
FOR A PLATE IN A TRANSVERSE FLOW

by Etienne Sourdille

The control of a moving structure in an unbounded flow has numerous applications in engineering, such as the aileron on an airplane. Here an approach is proposed where a CFD method is coupled with a controller to provide a qualitative flow model, and a tool for the development and the validation of the control scheme. A rotating rigid flat plate in transverse flow is considered.

For the CFD, a discrete vortex method is used due to its relevance for separated flows, which implies approximating the flat plate by a thin ellipse. The simulation for a fixed plate has been completed with a plate approximated by a 20:1 ellipse and placed in an inviscid flow. A comparison with an image method is also undertaken. The results show encouraging features for modeling the vortex street, but also problems in the transient behavior of the flow.

The control method is based on fuzzy logic, and has shown a remarkable ability to adapt to the nonlinear nature of the force generated by the flow/structure system. Comparison is made with more classical schemes such as a controller based on optimal control theory using an intermediary flow/structure model, similar to a gain scheduling model, instead of the full simulation.

Table of Contents

Abstract	1
Table of Contents	4
List of figures	14
Acknowledgments	15
Nomenclature	23
1 Introduction	24
1.1 General overview	24
1.1.1 Objectives	25
1.1.2 Systems	26
1.1.3 Methodology	27
1.1.4 Integration	28
1.2 Literature review	28
1.2.1 Overview	28
1.2.2 Guiding papers	29
1.2.3 Discrete vortex schemes	30
1.2.4 Vortex shedding phenomenology	31
1.2.5 Motion vibration control schemes	34
2 The discrete vortex method	36
2.1 Approach	36
2.1.1 Discrete introduction of vorticity	38
2.1.2 Boundary vortices convected in the flow	41
2.1.3 Geometry for the simulation	42
2.2 General theory	43
2.2.1 Governing equations	43
2.2.2 Discretization of the vorticity field	46
2.2.3 Convection of the vortices	51
2.3 Implementation of the method	51
2.3.1 Boundary conditions	52
2.3.2 Vortex core size	64
2.3.3 Boundary vortices position	66
2.3.4 Discrete introduction of vorticity at separation points	67

2.3.5	Body representation	72
2.3.6	Force and moment evaluation	73
2.3.7	Time step choice	78
2.3.8	Spalart Code implementation	81
2.4	Changes for the moving ellipse	82
2.4.1	Boundary condition	83
2.4.2	Flow geometry	83
2.4.3	Force and moment computation	83
2.5	Algorithm overview	84
3	Vortex/plate simulations	86
3.1	Outline	86
3.2	Validation of the codes	87
3.2.1	Blob vortex solution on the boundary	87
3.2.2	Influence of the global number of vortices	99
3.2.3	Plate at fixed angles	105
3.2.4	Moving plate simulations	129
3.3	Angular oscillation of a plate	143
3.3.1	Oscillations of a spring damped plate with no freeflow	147
3.3.2	Spring damped plate with angular oscillations	162
3.4	Summary	195
4	Plate and flow system modeling	197
4.1	Outline	197
4.2	Introduction	198
4.3	Flow model identification	203
4.3.1	Strategy for the identification	203
4.3.2	Fuzzy logic theory	209
4.3.3	Model frequency change	217
4.3.4	Basic linear model identification	221
4.3.5	Steady-angle flow model	227
4.3.6	Plate model and equations of motion	235
4.4	Coupled plate and flow characteristics	236
4.5	Summary	244
5	Coupled plate/flow system control	246
5.1	Outline	246
5.2	Control Development	247
5.2.1	General approach for the control design	247
5.2.2	Controller specifications	251
5.2.3	Classical control theory	253
5.2.4	Optimal control	264
5.2.5	Fuzzy logic controller	276
5.3	Integrated Controller/Flow simulation	291
5.3.1	Using the Matlab interface	291
5.3.2	Direct implementation in the Spalart simulation	292
5.4	Flow/plate and controller system	292
5.4.1	Plate stabilization	294

5.4.2	Prescribed plate motion	309
5.5	Summary	349
6	Conclusion and Discussion	350
A	Streamline function of the flow elements	361
B	Derivatives for vortex induced velocity	362
C	Force and moment calculus using complex images	364
C.1	Force calculus	366
C.1.1	Integral I1	366
C.1.2	Integral I2	370
C.1.3	Integral I3	373
C.1.4	Integral I4	373
C.2	Moment calculus	374
C.2.1	Integral M1	374
C.2.2	Integral M2	375
C.2.3	Integral M3	375
D	Force and moment calculus using Wu method	378
D.1	Force calculus	379
D.2	Moment calculus	380
	Bibliography	382

List of Figures

1.1	General scheme	25
1.2	Implementation	27
2.1	General geometry	43
2.2	Comparison between different vortex cores	50
2.3	Example of disposition for the panel method (courtesy of Takeda)	54
2.4	Wall approximation for straight panels	58
2.5	Wall approximation for curved panels	59
2.6	Illustration of the reflection used (courtesy of N.C.Clarke)	73
2.7	Comparison of C_m for the Spalart simulation at $\alpha = 45$, for different Δt^*	79
2.8	Comparison of C_m for the Spalart simulation at $\alpha = 45$, for different Δt^* for a total number of vortices $N = 6800$	80
2.9	General algorithm	85
3.1	Streamline of a flow past a 20:1 ellipse for the blob vortex method	89
3.2	Streamline of a flow past a 20:1 ellipse for the potential flow solution	90
3.3	$\text{Log}_{10}(\varepsilon)$ distribution in the grid for the 20:1 ellipse and uniform translation	92
3.4	Logarithm of the normal velocity at the boundary for two incidence an- gles α	93
3.5	Velocity and streamline between the boundary vortices and the wall for 90° incidence	94
3.6	Comparison of the velocity induced by the blob vortices (red vector) and the velocity from the exact potential solution (blue line). The wall is marked by a black line, and the boundary vortices are marked by green crosses.	94
3.7	Streamline of a flow past a 20:1 ellipse for the blob vortex method	95
3.8	Streamline of a flow past a 20:1 ellipse for the potential flow solution	96

3.9	$\text{Log}_{10}(\varepsilon)$ distribution in the grid for the 20 : 1 ellipse and uniform rotation	97
3.10	Normal velocity at the boundary	98
3.11	Blob vortices plot for $t^* = 51.5$ and $\alpha = 50$ with various global number of vortices N using the Spalart code. See figure 3.12 for the scale.	102
3.12	Scale for figure 3.11.	103
3.13	C_n for various angle of incidence and global number of vortices N	103
3.14	C_t for various angle of incidence and global number of vortices N	104
3.15	S for various angle of incidence and global number of vortices N	104
3.16	Measures characterizing the flow	105
3.17	Vortex plot at a given time step from the Spalart code. Vortex are blue crosses, the body is black, the direction of the ellipse motion is indicated by the arrow	107
3.18	Nascent vortices strength shed from the ellipse for $\alpha = 80^\circ$. The filter is defined by equation 3.7.	109
3.19	Streamline of the flow for the Spalart code at $t^* = 51.5$ with $\alpha = [45, 50, 60]$	112
3.20	Streamline of the flow for the Spalart code at $t^* = 51.5$ with $\alpha = [70, 80, 90]$	113
3.21	Streamline of the flow for the C code at $t^* = 51.2$ with $\alpha = [45, 50, 60]$	114
3.22	Streamline of the flow for the C code at $t^* = 51.2$ for $\alpha = 70^\circ$, $t^* = 40$ for $\alpha = 80^\circ$ and $t^* = 72$ for $\alpha = 90^\circ$	115
3.23	Enlarged streamline plot at $t^* = 51.5$ for the Spalart code for $\alpha = 50$ (the black line represents the end of the streamline plots in fig 3.19 and 3.20)	116
3.24	σ and $(\sigma)\sqrt{N}$ influence on the Strouhal number for $\alpha = 50^\circ$ using the Spalart code. Both variables units are in meter.	118
3.25	Normal and tangential force coefficients time plot for the Spalart code with $\alpha = [45, 50]^\circ$	123
3.26	Normal and tangential force coefficients time plot for the Spalart code with $\alpha = [60, 70]^\circ$	124
3.27	Normal and tangential force coefficients time plot for the Spalart code with $\alpha = [80, 90]^\circ$	125
3.28	Normal and tangential force coefficients time plot for the C code with $\alpha = [45, 50]^\circ$	126
3.29	Normal and tangential force coefficients time plot for the C code with $\alpha = [60, 70]^\circ$	127

3.30	Normal and tangential force coefficients time plot for the C code with $\alpha = [80, 90]^\circ$	128
3.31	Streaklines and streamlines sequence for an impulsively started flat plate at $Re = 200$, $\alpha = 45^\circ$. The streamfunction values in (b) range from $[-1.13, 1.13] \times (2a U_\infty)$ with a concentration around 0 at $t^* = 6.78$	135
3.32	Force and moment coefficients for a stationary ellipse in a parallel freestream flow at 45 degrees with $Re = 200$	136
3.33	Force and moment coefficients for a rotating ellipse in a parallel freestream flow (for subfigure 3.33(b) the dotted line represents the forces and moment coefficients from Lugt and Ohring).	137
3.34	Plot scale for the streamline plots in figure 3.35, 3.36 and 3.37.	138
3.35	See figure 3.37 for the caption and figure 3.34(a) for the scale.	139
3.36	See figure 3.37 for the caption and figure 3.34(a) for the scale.	140
3.37	Sequence of streamlines around a rotating ellipse in a parallel flow for $Re = 200$, $Ro = 2$, and an initial incidence of $\pi/2$. In contrast to every other sections, the freeflow is flowing from right to left. See figure 3.34 for the scale.	141
3.38	Illustration of the system ellipse and fluid	143
3.39	Scale for the streamline plot in figures 3.45, 3.46, and 3.47.	154
3.40	Time evolution of the angle of the plate θ for a spring damped plate with $f_n^* = 1/2$, $\xi^* = 0$ and starting with an initial angle $\theta_0 = -1^\circ$	155
3.41	Time evolution of the angle of the plate θ for a spring damped plate with $f_n^* = 1/2$, $\xi^* = 0$ and starting with an initial angle $\theta_0 = 1^\circ$	156
3.42	Time evolution of the angle of the plate θ for a spring damped plate with $f_n^* = 1/8$, $\xi^* = 0$ and starting with an initial angle $\theta_0 = -1^\circ$	157
3.43	Time evolution of the angle of the plate θ for a spring damped plate with $f_n^* = 1/8$, $\xi^* = 0$ and starting with an initial angle $\theta_0 = 1^\circ$	157
3.44	Time evolution of the angle of the plate θ for a spring damped plate with $\xi^* = 0$ and starting with an initial angle $\theta_0 = -2.5^\circ$	158
3.45	Streamlines comparison of the flow for the Spalart code with $\theta_0 = -1^\circ$ and $f_n^* = 1/[2, 8]$. Streamfunction values range is $[-0.066, 0.066] \times (2a U_\infty)$. It corresponds to $[-0.0423, 0.0423] \times (2a U_{max})$ for $f_n^* = 1/2$, and $[-0.169, 0.169] \times (2a U_{max})$ for $f_n^* = 1/8$ (U_{max} varies with f_n^* see equation 3.27). In every cases, values are concentrated around 0. See figure 3.39 for the scale.	159

3.46 Streamlines comparison of the flow for the Spalart code with $\theta_0 = 1^\circ$ and $f_n^* = 1/[2, 8]$. Streamfunction values range is $[-0.066, 0.066] \times (2aU_\infty)$. It corresponds to $[-0.0423, 0.0423] \times (2aU_{max})$ for $f_n^* = 1/2$, and $[-0.169, 0.169] \times (2aU_{max})$ for $f_n^* = 1/8$ (U_{max} varies with f_n^* see equation 3.27). In both cases, values are concentrated around 0. See figure 3.39 for the scale. 160

3.47 Streamline of the flow for the Spalart code with $\theta_0 = -1^\circ$ and $f_n^* = 1/2$. Streamfunction values range is $[-0.066, 0.066] \times (2aU_\infty)$. It corresponds to $[-0.0423, 0.0423] \times (2aU_{max})$. Values are concentrated around 0. See figure 3.39 for the scale. 161

3.48 Time plot of the angle evolution for $f_n^* = 1/4$ with various ξ^* 165

3.49 Time plot of the angle evolution for $f_n^* = 1/8$ with various ξ^* 166

3.50 Time plot of the angle evolution for $f_n^* = 1/11$ with various ξ^* 166

3.51 Angle evolution $\theta(t)$ and PSD plot for θ^* and C_m (for $t^* = [12.8, 204.8]$) in the case $\xi^* = 0.02$ and $1/f_n^* = 4$. The crosses in the PSD are the main harmonics represented in $\theta^*(t)$ 168

3.52 Angle evolution $\theta(t)$ and PSD plot for θ^* and C_m (for $t^* = [12.8, 204.8]$) in the case $\xi^* = 0.02$ and $1/f_n^* = 7$. The crosses in the PSD are the main harmonics represented in $\theta^*(t)$ 169

3.53 Angle evolution $\theta(t)$ and PSD plot for θ^* and C_m (for $t^* = [12.8, 204.8]$) in the case $\xi^* = 0.02$ and $1/f_n^* = 9$. The crosses in the PSD are the main harmonics represented in $\theta^*(t)$ 170

3.54 Angle evolution $\theta(t)$ and PSD plot for θ^* and C_m (for $t^* = [12.8, 204.8]$) in the case $\xi^* = 0.02$ and $1/f_n^* = 11$. The crosses in the PSD are the main harmonics represented in $\theta^*(t)$ 171

3.55 Scale plot for figures 3.57, 3.58, 3.56, 3.59, 3.61, 3.62, 3.64, 3.65 173

3.56 History of θ and C_m in phase C for $1/f_n^* = 4$ 174

3.57 Streamlines for the flow in phase C for $1/f_n^* = 4$ with $t^* = [174.8, 178.3]$ (see figure 3.56). The scale is in figure 3.55. 175

3.58 Streamlines for the flow in phase C for $1/f_n^* = 4$ with $t^* = [178.8, 182.3]$ (see figure 3.56). The scale is in figure 3.55. 176

3.59 Streamlines for the flow in phase C for $1/f_n^* = 4$ with $t^* = [182.8, 184.3]$ (see figure 3.56). The scale is in figure 3.55. 177

3.60 History of θ and C_m in phase C for $1/f_n^* = 7$ 179

3.61	Streamline for the flow in phase C for $1/f_n^* = 7$ with $t^* = [135\ 138.4]$ (see figure 3.60). The scale is in figure 3.55.	180
3.62	Streamline for the flow in phase C for $1/f_n^* = 7$ with $t^* = [139\ 142.4]$ (see figure 3.60). The scale is in figure 3.55.	181
3.63	History of θ and C_m in phase C for $1/f_n^* = 11$	184
3.64	Streamline for the flow in phase C for $1/f_n^* = 11$ with $t^* = [154.8\ 158.3]$ (see figure 3.63). The scale is in figure 3.55.	185
3.65	Streamline for the flow in phase C for $1/f_n^* = 11$ with $t^* = [158.8\ 162.3]$ (see figure 3.63). The scale is in figure 3.55.	186
3.66	Summary of the simulations detailing the variations of the shedding frequency f_s^* and the maximum amplitude of angle oscillation regarding the nondimensional natural frequency f_n^*	189
3.67	Mean values and root mean square of C_n depending on $1/f_n^*$	192
3.68	Mean values and root mean square of C_t depending on $1/f_n^*$	193
3.69	Mean values and root mean square of C_m depending on $1/f_n^*$	194
4.1	Plan for the system modeling	203
4.2	Starting transfer functions for the magnitude and frequency interpolation in figure 4.3.	205
4.3	Magnitude interpolation only compared to magnitude and frequency interpolation for a constant input x_r . Functions f_1 and f_2 are presented in figure 4.2.	206
4.4	Example of simulation time range retained	208
4.5	General diagram for identification method (courtesy of G.Zwingelstein)	209
4.6	General example for fuzzy logic	210
4.7	Fuzzy set for u_1	211
4.8	Fuzzy set for u_2 and y	212
4.9	Example of inference table	213
4.10	Example of inference for one fuzzy rule	214
4.11	Results from all the fuzzy rules, and aggregation	215
4.12	Defuzzification using the centroid method	216
4.13	General linear model for the flow at a steady angle	223
4.14	Validation results for a plate with steady incidence angles $\alpha = 45, 50$ degrees	224

4.15	Validation results for a plate with steady incidence angles $\alpha = 60, 70$ degrees	225
4.16	Validation results for a plate with steady incidence angles $\alpha = 80, 90$ degrees	226
4.17	Global model for the plate at steady angles	229
4.18	Frequency and mean interpolation function	230
4.19	Membership functions of α	232
4.20	Comparison between simulation and steady plate model for $\alpha = 55$ degrees, output history plot.	234
4.21	Comparison between simulation and steady plate model using a periodogram. P_{XX} is the power spectral density (PSD) of the signal $x(t)$ or –for this figure– C_m	234
4.22	Plate dynamics model	235
4.23	Coupled plate and flow model Bode diagram at $\alpha = 50^\circ$ using polystyrene	238
4.24	Coupled plate and flow model Bode diagram at $\alpha = 50^\circ$ using aluminium	239
4.25	Coupled plate and flow model Bode diagram at $\alpha = 60^\circ$ using polystyrene	240
4.26	Coupled plate and flow model Bode diagram at $\alpha = 60^\circ$ using aluminium	241
4.27	Coupled plate and flow model Bode diagram at $\alpha = 70^\circ$ using polystyrene	242
4.28	Coupled plate and flow model Bode diagram at $\alpha = 70^\circ$ using aluminium	243
5.1	Coupled plate and flow model	249
5.2	Controller, Coupled plate and flow model	250
5.3	Classical based, linear controller block diagram	256
5.4	Control and system closed loop block diagram for the classical controller.	256
5.5	Output of the closed loop system for a step input for the classical controller.	259
5.6	Error of the closed loop system for a step input for the optimal controller.	260
5.7	Control signal evolution in time for the classical controller (the vertical line is actually a control input peak due to the step input). Cf C_e definition in eq. 5.19.	261
5.8	Control signal evolution in time for the classical controller without taking into account the torque due to the spring. Cf C_e definition in eq. 5.19 for the non-dimensionalization.	262
5.9	Power signal evolution in time for the classical controller. The power is divided by $(1/2)\rho U_\infty^3 L$	263
5.10	Optimal, linear controller block diagram	270

5.11	Control and system closed loop block diagram for the optimal controller.	271
5.12	Output of the closed loop system for a step input for the optimal controller.	271
5.13	Error of the closed loop system for a step input for the optimal controller.	272
5.14	Control signal evolution in time for the optimal controller (the vertical line is actually a control input peak due to the step input). Cf C_e definition in eq. 5.19.	273
5.15	Control signal evolution in time for the optimal controller without taking into account the torque due to the spring. Cf C_e definition in eq. 5.19 for the non-dimensionalization.	274
5.16	Power signal evolution in time for the optimal controller. The power is divided by $(1/2)\rho U_\infty^3 L$	275
5.17	Membership functions of the input	277
5.18	Inference table for the fuzzy control	278
5.19	Control and system closed loop block diagram for the fuzzy logic controller.	284
5.20	Output of the closed loop system for a step input for the fuzzy logic controller.	285
5.21	Error from the optimal controller minus the error from the fuzzy controller with relative calculus precision of 10^{-6}	285
5.22	Error of the closed loop system for a step input for the fuzzy logic controller.	286
5.23	Control signal evolution in time for the fuzzy logic controller (the vertical line is actually a control input peak due to the step input). Cf C_e definition in eq. 5.19.	287
5.24	Control signal evolution in time for the fuzzy logic controller without taking into account the torque due to the spring. Cf C_e definition in eq. 5.19 for the non-dimensionalization.	288
5.25	Power signal evolution in time for the fuzzy controller. The power is divided by $(1/2)\rho U_\infty^3 L$	289
5.26	Control signal and angular error evolution in time for the fuzzy logic controller with relative calculus precision of 10^{-6} . Cf C_e definition in eq. 5.19 for the non-dimensionalization.	290
5.27	Power signal evolution in time for the fuzzy controller with relative calculus precision of 10^{-6} . The power is divided by $(1/2)\rho U_\infty^3 L$	291
5.28	Control and system closed loop block diagram for the optimal controller with an average delay at the flow simulation output.	294

5.29 Time plot of the angle evolution and controller torque input for the flow/plate system controlled by the fuzzy logic controller with $f_n^* = S_s$ and $\Delta t^* = 0.05$ 299

5.30 Time plot of the angle evolution and torque input for the flow/plate system controlled by the fuzzy logic controller with $f_n^* = 1.5 S_s$ and $\Delta t^* = 0.05$ 300

5.31 Time plot of the angle evolution and torque input for the flow/plate system controlled by the optimal controller with $f_n^* = S_s$ and $\Delta t^* = 0.04$ 301

5.32 Time plot of the angle evolution and torque input for the flow/plate system controlled by the optimal controller with $f_n^* = 1.5 S_s$ and $\Delta t^* = 0.04$ 302

5.33 Time plot of the angle evolution and torque input for the flow/plate system controlled by the optimal controller with $f_n^* = S_s$ and $\Delta t^* = 0.05$ 303

5.34 Time plot of the angle evolution and torque input for the flow/plate system controlled by the optimal controller with $f_n^* = 1.5 S_s$ and $\Delta t^* = 0.05$ 304

5.35 Time plot of the angle evolution and torque input for the flow/plate system controlled by the optimal controller with $f_n^* = 1.5 S_s$ and $\Delta t^* = 0.05$ 305

5.36 Time plot of the angle evolution for the flow/plate system with $f_n^* = S_s$, $\Delta t^* = 0.05$ and the fuzzy-logic controller enabled at $t^* = 51.2$ 306

5.37 Time plot of the torque input for the flow/plate system with $f_n^* = S_s$, $\Delta t^* = 0.05$ and the fuzzy-logic controller enabled at $t^* = 51.2$ 307

5.38 Time plot of the angle evolution and torque input for the flow/plate system with $f_n^* = S_s$, $\Delta t^* = 0.05$ and the fuzzy-logic controller enabled at $t^* = 51.2$ during transition 308

5.39 Time plot of the angle evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = S_s$ 316

5.40 Time plot of the angle evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = 1.5 S_s$ 316

5.41 Time plot of the error evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = S_s$ 317

5.42 Time plot of the controller torque input for the flow/plate system motion controlled by the fuzzy logic controller with $f_n^* = S_s$ 318

5.43 Time plot of the power signal evolution in time for the fuzzy logic controller with $f_n^* = S_s$. The power is divided by $(1/2)\rho U_\infty^3 L$ 319

5.44 Time plot of the error evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = 1.5 S_s$ 320

5.45 Time plot of the controller torque input for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = 1.5 S_s$ 321

5.46 Time plot of the power signal evolution in time for the fuzzy logic controller with $f_n^* = 1.5 S_s$. The power is divided by $(1/2)\rho U_\infty^3 L$ 322

5.47 Time plot of the angle evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = S_s$ 323

5.48 Time plot of the angle evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = 1.5 S_s$ 323

5.49 Time plot of the error evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = S_s$ 324

5.50 Time plot of the torque input for the flow/plate system motion controlled by the optimal controller with $f_n^* = S_s$ 325

5.51 Time plot of the power signal evolution in time for the optimal controller with $f_n^* = S_s$. The power is divided by $(1/2)\rho U_\infty^3 L$ 326

5.52 Time plot of the error evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = 1.5 S_s$ 327

5.53 Time plot of the torque input for the flow/plate system motion controlled by the optimal controller with $f_n^* = 1.5 S_s$ 328

5.54 Time plot of the power signal evolution in time for the optimal controller with $f_n^* = 1.5 S_s$. The power is divided by $(1/2)\rho U_\infty^3 L$ 329

5.55 Time plot of the angle evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = S_s$ and oscillating input . . 335

5.56 Time plot of the angle evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = 1.5 S_s$ and oscillating input 335

5.57 Time plot of the error evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = S_s$ and oscillating input . . 336

5.58 Time plot of the controller torque input for the flow/plate system motion controlled by the fuzzy logic controller with $f_n^* = S_s$ and oscillating input 337

5.59 Time plot of the power signal evolution in time for the fuzzy logic controller with $f_n^* = S_s$ and oscillating input. The power is divided by $(1/2)\rho U_\infty^3 L$ 338

5.60 Time plot of the error evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = 1.5 S_s$ and oscillating input 339

5.61	Time plot of the controller torque input for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = 1.5 S_s$ and oscillating input	340
5.62	Time plot of the power signal evolution in time for the fuzzy logic controller with $f_n^* = 1.5 S_s$ and oscillating input. The power is divided by $(1/2)\rho U_\infty^3 L$	341
5.63	Time plot of the angle evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = S_s$ and oscillating input . . .	342
5.64	Time plot of the angle evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = 1.5 S_s$ and oscillating input . .	342
5.65	Time plot of the error evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = S_s$ and oscillating input . . .	343
5.66	Time plot of the torque input for the flow/plate system motion controlled by the optimal controller with $f_n^* = S_s$ and oscillating input	344
5.67	Time plot of the power signal evolution in time for the optimal controller with $f_n^* = S_s$ and oscillating input. The power is divided by $(1/2)\rho U_\infty^3 L$.	345
5.68	Time plot of the error evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = 1.5 S_s$ and oscillating input . .	346
5.69	Time plot of the torque input for the flow/plate system motion controlled by the optimal controller with $f_n^* = 1.5 S_s$ and oscillating input	347
5.70	Time plot of the power signal evolution in time for the optimal controller with $f_n^* = 1.5 S_s$ and oscillating input. The power is divided by $(1/2)\rho U_\infty^3 L$.	348
C.1	Conformal transformation illustration	365
C.2	Complex decomposition of contour	367
C.3	Contour C_k description	377

Acknowledgments

I would like to thank first and foremost Dr Gary N Coleman for supervising this work, and especially for his sound advice and his comprehension over the years. Also many thanks to Dr Owen R Tutty for his technical help as an advisor and his frankness. I am grateful too to the Aeronautics and Astronautics department staff as well as the University of Southampton for funding my work.

There have also been many other collaborators to which I'm indebted. Professor Sergei I Chernyshenko for having enlightened me on Complex problems and certain flow concepts; Doctor Kenji Takeda for his help on the previous vortex method research in the department; Professor Philippe Spalart for having given some insight, and sparking some interests on the vortex method as well as for helping me with the numerical flow method. I also want to thank the many people which I've known in my years in the University and the colleagues who passed over the years who have always supported me (in an english and french sense) among those David Simonin, Ajay N Modha, Sandrine Morin, Cary Turagan, A. Bouferrouk, Joanna Hyde, Anne Roques and many other people.

Finally, special thanks to my wife who has born part of the burden during this thesis writing.

Nomenclature

(\vec{X}, \vec{Y})	Stationary frame during the motion centered in $O(0,0)$
α	Ellipse incidence angle or angle of attack
α_0	Incidence angle of the ellipse at $t = 0$
\bar{x}	Mean value of x
$\Delta\Gamma$	Rate of vorticity
$\Delta\theta$	Angular error
δN	Distance between a boundary vortex and the corresponding boundary control point
Δt	Timestep (Note : sometimes referred to as dt or dt in graphics)
δ	Two-dimensional Dirac delta function
δ_0	Average distance between the control points on the boundary
δ_{bl}	Boundary layer thickness
ϵ	Distance of the nascent vortex to the body when using the discrete introduction of vorticity method
Γ	Strength of a discrete vortex or circulation
γ	Vorticity distribution or vortex core function
Γ_b	Circulation on the boundary
Γ_f	Wake circulation
μ	Torsional damping of the spring damped ellipse sometimes referred to as b in graphics

μ_s	Effective damping of the spring-damped ellipse system using alternative nondimensionalization (see section 3.3.1 for mathematical definition)
∇	Gradient operator
ν	Kinematic viscosity
Ω	Body angular velocity
ω	Magnitude of $\vec{\omega}$
$\vec{\omega}$	Vorticity of the fluid
\vec{F}_s	Smoothed aerodynamic force
\vec{F}	Aerodynamic force
\vec{f}	Body force per unit mass
\vec{K}	unit vector orthogonal to the (\vec{x}, \vec{y}) plane
\vec{n}	Unit vector normal to the wall
\vec{P}	Point position in a flow domain
\vec{s}_b	Position vector defining the location of the surface boundary
\vec{s}	Unit vector tangential to the body surface
\vec{U}_∞	Freestream flow velocity vector
\vec{u}_n	Normal velocity component on the boundary
\vec{u}_o	Initial velocity field
\vec{u}_s	Tangential velocity component on the boundary at $\vec{x} = \vec{s}_b$
\vec{u}	Fluid velocity
\vec{U}_e	External flow velocity
\vec{v}_s	Body velocity
$\vec{V} = (u, v)$	Velocity vector with u and v respectively the \vec{x} and \vec{y} axis component
\vec{x}_o	Center of rotation of the ellipse

\vec{x}_v	Discrete vortex position in a flow domain
\vec{x}	major axis of the ellipse (can also denotes a position)
\vec{y}	minor axis of the ellipse
Φ	Velocity potential in a stationary frame
ψ	Streamline function
ψ_∞	Stream function from the uniform freestream velocity
ψ_b	Stream function associated to the boundary vortices
ψ_f	stream function coming from the flow vortices
ψ_r	Streamfunction due to the rotation
ρ	Fluid mass per volume unit
ρ_b	Body mass per volume unit
σ	Cut-off radius, or vortex core radius
$\theta(t)$	Angular position of the plate <i>from the rest position of the torque spring</i> , $\alpha = \alpha_0 + \theta(t)$ and $\Omega = \dot{\theta}$
θ_0	Initial ellipse spring angle
θ_i	Angle determining the position of the i^{th} boundary vortices in chapter 2
θ_{max}	Maximum θ oscillations amplitude
\underline{E}	External disturbance vector
$\underline{R_c}$	Reference command signal
$\underline{R_v}$	Covariance of \underline{V}
$\underline{R_w}$	Covariance of \underline{W}
\underline{U}	Input vector of system
\underline{V}	Sensor noise
\underline{W}	Process noise

\underline{X}_c	Controller state vector
\underline{X}	State vector of system
\underline{Y}	Output vector of system
$\varepsilon(x, y)$	Error at a point of position (x, y) (noted ϵ in figures)
ε_p	Position error
ε_v	Velocity error
ϖ_n	Natural frequency of the spring damped ellipse in rad/s
ξ	Reduced damping of the spring damped ellipse
ξ_s^*	Nondimensional effective ξ of the $\theta(t)$ produced by the coupled mechanical/flow system
ζ	Complex position in a circle plane (before a Joukowski transformation) for example
ζ'	Complex value of the location of an image vortex in the ζ plane
ζ^*	Conjugate of a complex position in a circle plane
a	Ellipse half chord on its primary axis
b	Half chord of the ellipse on the minor axis
c	Half the chord value or radius of the circle in the complex image plane
C_a	Torque output of the controller, scalar dependent on time
C_d	Aerodynamic drag coefficient
C_e	Non-dimensionalized torque output of the controller (see equation 5.19), scalar dependent on time.
C_e	Nondimensional external torque provided by Simulink to the C flow simulation
$C_G = 1/k$	Steady-state gain of the transfer function. (see section 3.3 for mathematical definition)

C_J	Additional mass to the spring-damped ellipse system (see section 3.3.1 for mathematical definition)
C_l	Aerodynamic lift coefficient
C_{ma}	Approximated C_m
C_m	Aerodynamic moment coefficient
C_n	Aerodynamic normal force coefficient (associated to the \vec{y} axis)
C_p	incident flow power impinging on the plate of chord L . It is defined by $C_p = (1/2)\rho U_\infty^3 L$ for a $2D$ plate.
C_t	Aerodynamic tangential force coefficient (associated to the \vec{x} ellipse axis)
D	Entire flow domain
D_0	Ad hoc Spalart simulation parameter: length parameter that allows one to obtain better resolution near the wall
$D\%$	Peak response in percent
e	Error signal (input of the fuzzy logic controller), scalar dependent on time
$E_{c,p}(T_1, T_2)$	Energy provided by P_c^* between T_1 and T_2 (see equation 5.9)
$E_{c,p}(T_1, T_2)$	Signal energy provided by P_c^* between T_1 and T_2 (see equation 5.9)
$E_c(T_1, T_2)$	Energy used by the controller between T_1 and T_2 (see equation 5.3)
$E_s(T_1, T_2)$	Signal energy of the controller between T_1 and T_2 (see equation 5.5)
$F(s)$	Laplace transform of a function $f(t)$
$f(t)$	Generic function dependent on time when applied to a control system
f_θ	θ oscillation frequency for the coupled spring damped ellipse/flow system
f_m	Flow linear model main harmonics frequency
$f_{n,s}^*$	Nondimensional effective f_n of the $\theta(t)$ produced by the coupled mechanical/flow system
f_n	Spring damped system natural frequency

f_{samp}	Sampling frequency
f_s	Flow vortex shedding frequency
h	Length of a square cell in a uniform grid defining the approximate flow domain simulated
J	Angular inertia of the spring damped ellipse
J_s^*	Effective nondimensional inertia of the spring-damped ellipse system (see section 3.3.1 for mathematical definition)
k	Torsional spring coefficient applied of the spring damped ellipse, exceptionally generic scalar in section 4.3.3
L	Ellipse Chord
M	Aerodynamic moment
M_e	External torque provided by Simulink to the C flow simulation
mfx	Generic name for membership function for fuzzy logic controller
N	Total number of discrete vortices
N_b	Number of boundary vortices
O or $O(0, 0)$	Origin
p	pressure
P_b	Set of parameters defining the position and orientation of a boundary element
P_c	Instantaneous power used by the controller (see equation 5.2)
P_{XX}	Signal power per frequency in the spectrum considered for a time signal $x(t)$. This is related to the signal power not the physical power. The units of P_{XX} is that of x^2 . The power spectral density (PSD) designates the P_{XX} considered over the whole spectrum.
Q	State penalty matrix
R	Cost penalty matrix
R_∞	Limitless region jointly occupied by the solid body and the fluid

R_s	Region occupied by the solid body
Ro	Rosby number, $Ro = U_\infty / (a\Omega)$
S	Strouhal number or Strouhal number extracted from experiment
S_{body}	Cross section surface of a body
S_s	Actual Strouhal number extracted from previous simulation
T	A duration value
t	Real timescale or time value
t'	Alternative timescale
$t_{5\%}$	Minimal settling time
t_r	Minimal rise time
u	Input signal of a system, or output signal of a controller (scalar form)
U_∞	Magnitude of the uniform freestream velocity or uniform freestream complex velocity
U_{max}	Maximum angular velocity of the spring damped ellipse system in vacuum (see section 3.3.1 for mathematical definition)
U_{sh}	Shear layer velocity magnitude
u_X	\vec{X} velocity component of \vec{u}
u_Y	\vec{Y} velocity component of \vec{u}
V_0	Ad hoc Spalart simulation parameter: tolerance parameter for blob vortex merging adjustment
V_1	Velocity magnitude at the outer edge of the shear layer
V_2	Velocity magnitude at the inner edge of the shear layer
w	complex flow potential
w_f	Freeflow complex potential
w_t	Entire flow complex potential

$x_j(t)$ Position of the j^{th} discrete vortex

y Output signal when applied to a control system (scalar form)

(\vec{x}, \vec{y}) Body fixed frame of reference

* Convolution operator

Re Reynolds number

Subscript $_b$ Associated to the boundary vortices

Subscript $_e$ Associated to the estimator (when applied to control system)

Subscript $_p$ Associated to the vortices with positive circulation in the method with discrete introduction of vorticity

Subscript $_q$ Associated to the vortices with negative circulation in the method with discrete introduction of vorticity

Subscript $_w$ Associated to the vortices released in the flow

Superscripts * Nondimensionalized variable (except for complex variables)

VN,N,Z,P,VP Fuzzy set when related to a fuzzy logic controller

Chapter 1

Introduction

1.1 General overview

Many engineering devices involve the interaction of a flexible structure and a moving fluid. In some cases this interaction is dynamic, with the deflected structure altering the original flow pattern - and the altered flow pattern then re-deflecting the structure and so on. Accurate analysis of these cases requires a careful multi-disciplinary approach. They also provide a challenging test for active control strategies. With this project, a study is made about a rotating rigid plate in a transverse crossflow. By considering an idealized fluid-structure interaction, an integrated fluid, structure and active control analysis is developed, which is expected to be relevant to the more complicated situations found in practice.

In order to investigate the flow around the plate, whether stationary or rotating about an axis normal to the flow, a CFD method is implemented. The main aim is to design a method simple enough to avoid using too much computer power, but accurate enough to model realistically the flow pattern and the flow forces. Therefore, a $2D$ (2 dimensional) blob vortex method has been chosen for modeling the unsteady wake. The position and the strength of the nascent vortices are able to vary with time through the use of the Kutta condition.

The flow simulation will enable investigations of the effect of the moving plate on the downstream flow. The control will then be used to stabilize the plate by either maintaining a stationary state or by following a user specified motion.

The goals of the project are :

- Study the flow past a rotating rigid flat plate in different configurations (steady inclination, oscillating between two inclinations, or rotating)
- Validate the blob vortex method with a rotating body
- Investigate the strategies and limits of a control scheme for this test case

1.1.1 Objectives

The main objective of this project is the control of a flat plate placed in a transverse unbounded flow, with angle of attack ranging from 45 to 90 degrees. The fluid is assumed to be incompressible. The plate is assumed to be perfectly rigid, and bending and vibration of the structure are ignored.

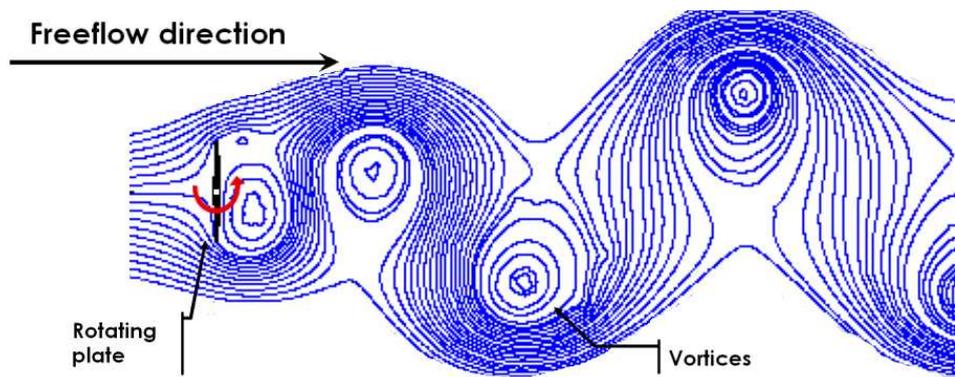


Figure 1.1: General scheme

The plate will be held fixed in some cases and in others allowed to rotate about a central axis, with a torsional spring and damper to resist the motion. This will illustrate the feasibility and the limitations of this approach applied to similar flows.

Although this project remains fairly idealized, it is intended to act as a preliminary for studies of more realistic real-world flows, such as that around a moving spoiler on an airfoil.

1.1.2 Systems

Several systems are to be implemented to achieve the full simulation. These include the $2D$ flow simulation and modeling, the computation of the dynamics of the rigid plate, and the control. The control part will be implemented in Matlab, whereas the flow and kinematics parts will be implemented in C/C++.

1.1.2.1 $2D$ Dynamic flow simulation

First of all, a simple flow simulation is considered, that is to say in a way that could allow not only the coupling with the control, but also good computational speed. It is also necessary here that the simulation algorithm can be readily modified. However, the simulation must still be also be able to produce realistic flow characteristics, for example the Karman Street in the case of the fixed plate. Therefore, a vortex method was chosen as the most appropriate method, since it is ideal for simulating separated bluff-body flows.

For the programming language, C/C++ was used for reasons of computational speed and availability. A first implementation has been done in Matlab to settle the algorithm.

1.1.2.2 Plate motion

Using classical mechanics equations and the input forces from the $2D$ flow simulation, one can then deduce the movement of the plate. A homogeneous rigid plate is considered with mass density corresponding to Plexiglass.

This part will also be coded in C/C++ as a module for the $2D$ flow simulation.

1.1.2.3 Control

Considering now the control part, it must ensure numerous roles, among which are the control of the angle of attack as a function of time. In addition, the plate stabilization has to be achieved. Finally, for this project, the flow simulation and the control must communicate at each iteration. Matlab was chosen as there are a number of tools dedicated to control design.

1.1.3 Methodology

The methodology consists of three steps. At each step, an assessment is done to validate the completed part. The first part will be dedicated to the implementation of the vortex method with a fixed plate in a crossflow. In the second part, the plate will rotate freely around a given axis under the flow action. Finally, the third step will consist of the implementation of the control as well as the development of a system model of the plate and the flow.

Although this type of implementation does not provide optimal computational speed, it enables to simplify the debugging as well as giving flexibility if the configuration of the system needs to be changed.

The first two steps provide the flow simulation, whereas the third gives the control. It should be noted that between the control part and the simulation part, the input and output were to be as simple as possible in order to keep these two parts independent.

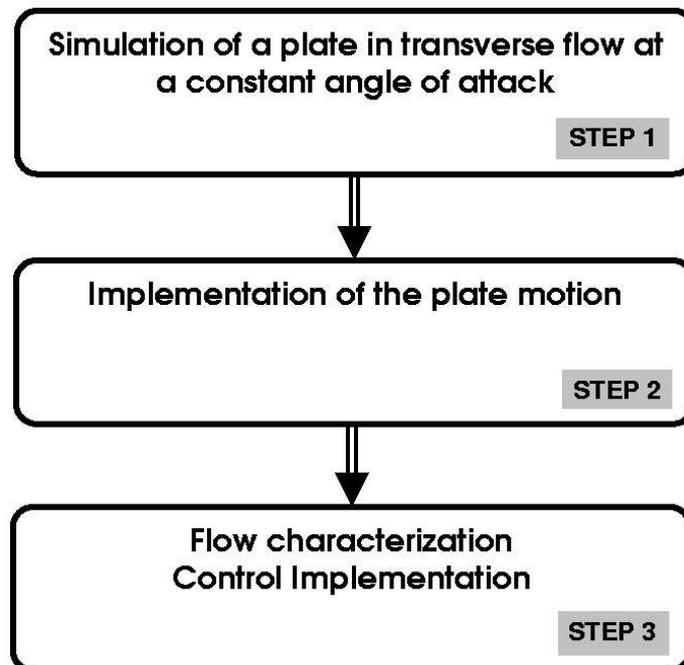


Figure 1.2: Implementation

1.1.4 Integration

The simulation and the control need to be integrated. As previously stated, the flow simulation will be implemented in C/C++, and the control part in Matlab. Therefore, several approaches are possible, namely :

- To use a file as an interface between C/C++ and Matlab. At each time step, the flow simulation generates a file containing all the useful data of the flow for use by the control. In Matlab, this file is then read, and create a file as an input for the flow simulation after having determined the resulting control.
- To implement a program in either C/C++ or Matlab to handle the data as well as dynamically determine the interaction between the flow simulation and the control.

The first approach is the easiest, but is intrinsically slower than the second option. Furthermore, it requires careful synchronization between C and Matlab. On the other hand, the second approach requires a driver to couple both parts (C/C++ and Matlab). The main implementation difficulty consists in using an intermediary function to enable the simulation and Matlab to share data through a common data structure. Thus, this driver implementation needs little modification of both parts. Therefore, the second approach was preferred to integrate the Matlab controller and the C simulation.

1.2 Literature review

1.2.1 Overview

As standard methods will be used for the control (linear systems, system modeling, etc), the literature review will be oriented toward the vortex method. Nevertheless, some papers have appeared on the subject of vibrating plates (no aerodynamics involved), or of flap of a wing. A brief review of these studies is provided in the third section.

The vortex method is widely used in research, mainly because of its ability to create physically relevant dynamics at high Reynolds number. Consequently, there is an extensive literature on the subject; one of the most exhaustive papers is by Sarpkaya (1994) [53], which includes more than 500 references. The reader is directed to that

work for a comprehensive review, while here are the papers which are more significant for the current project purposes.

This review is divided into three sections. The first considers papers on the complete vortex method, i.e. providing the theoretical and computational tools to develop and implement the method. The second section introduces different schemes available as part of the vortex method. The third section focuses on some of the schemes available for controlling motion in structural and mechanical problems.

1.2.2 Guiding papers

The most used papers, as well as the most extensive in their review, are those of Spalart (1988) [57], Sarpkaya (1994), Leonard (1980) [33], and Chorin (1993) [10]. One should also take into account the books of Katz and Plotkin (1991) [30] and Koutsoumakos (2000) [13].

The Chorin paper [10] exposes some of the theoretical aspects of the discrete vortex method in two and three dimensions. Although very useful for convergence and error study, it does not provide the material necessary for constructing the method. The Koutsoumakos book [13] has the same scope albeit more comprehensively. It studies increasingly complex cases including hybrid schemes (finite difference on the boundary, vortex difference for the far field). Koutsoumakos also provide some scope over the parameters for convergence of the method.

Leonard [33] presents a complete review on the method which is more oriented toward application. He exposes some of the problems such as the accuracy of convection schemes, the determination of an appropriate blob shape, and the viscosity problem, as well as some test cases. He also considers the $3D$ (3 dimensional) case. It has been found very useful since he discusses how to set the ad-hoc parameters (vortex core radius, time step, minimum number of vortices to use) and presents an explanation about convergence. However, not enough details are provided for a full implementation of the method.

Spalart (1988) [57] is a very positive paper exposing impressive results for flow over

complicated body shapes. The paper is “recipe book” giving details of the implementation of the discrete vortex method. There is even a part dedicated to the efficient programming of the method.

Sarpkaya (1994) [53] is a thorough review paper about the different schemes in use. He first develops the mathematics related to the method, and discusses the different schemes in chronological order. He then gives a great deal of detail about each of them. Compared to other papers, this review is oriented toward an engineering point of view.

Katz and Plotkin (1991) [30] in their book provide a useful background for the elements method in general (discrete source, discrete vortices, $2D$ and $3D$). Although it is lacking some details about the schemes, like the Spalart paper it is a “recipe book” with an engineering perspective. Therefore, it is a good starting point for the understanding and implementation of discrete methods. It does not however discuss blob vortex methods.

1.2.3 Discrete vortex schemes

Numerous vortex methods have been devised for steady as well as unsteady flows. Here is not an exhaustive review of the various methods but rather a concise presentation of the most representative ones. The flow configuration considered in this project (transverse flow over a flat plate) is incompressible, inviscid and with separation.

Most flows treated are incompressible. The compressible case are not reviewed here; it is discussed in references [1], [2], [6], [8], [20] and [57] on.

Sarpkaya (1974) [50] used an image method to model the flow around a transverse flat plate. Unlike the preceding papers, the shedding of vorticity is done discretely at the separation points. He obtained some fairly good results, qualitatively speaking. He conducted some further studies with the scheme being applied around a cylinder [49], and plates with a parabolic leading edge [52].

Koutsoumakos and Shiels [32] uses a Vortex In Cell (VIC) method and a Particle Strength Exchange (PSE) scheme to model the flow around an impulsively started and uniformly accelerated transverse flat plate. They present a great deal of theoretical

details. Their results agree well with experiment and yield insight into some of the effects observed.

Blevins (1991) [7] used a code developed by Spalart and Leonard (discrete vortex with a surface distribution for the vortices, random walk method to account for viscosity) to simulate flow around moving structures in an unbounded domain. Results show fairly good agreement with experiment for the mean lift and drag. However, their work is mostly qualitative as the phase and amplitude of the vortex shedding were overpredicted.

Clarke (1992) [11] developed a discrete vortex method for simulating the flow around a cylinder, using for the viscosity model a combination of diffusion velocity near the wall, and random walk away from the wall. He also managed to run the simulation with a rotating cylinder, and conducted some long-term simulation for Reynolds number ranging from 5000 to 31700, with good experimental agreement.

Takeda (1998) [59] did similar work but using viscosity distribution method to account for viscosity. The method is similar to a PSE method but differs in that there is no regular regriding. He employed a highly parallel code. Like Clarke, he obtained high-quality results.

Most recently, Shiels (1998) [56] proposed a viscous vortex method based on blob vortex core expansion. He applied this method to a transversely moving cylinder and specified various means of computing the fluid forces. Although his simulation captured the behaviour accurately enough to be confident that the dominant flow structures are properly resolved, the accuracy at small scales is in doubt.

1.2.4 Vortex shedding phenomenology

The focus of this thesis requires an understanding of the interaction of the plate/flow system, and such the behavior of the basic flow. Thus, a review was made based on studies of vortex-street and periodic flow phenomena. The scope of these studies is a bit less wide than for the thesis as they usually attempt to understand the intrinsic flow mechanisms and contribute to accurate predictions of their effect.

Marris (1964 [37]) provides a survey of research about periodic wakes behind a cylinder in a uniform steady stream and is a synopsis of knowledge of the vortex formation and of the associated hydrodynamic forces for a wide range of Reynolds numbers. He also discusses characteristics of the induced vibration in a uniform stream for various spring-mounted bluff bodies configurations, and then focuses on the cylinder case. After analyzing the vortex wake behind a cylinder at low Reynolds number of $Re < 300$, he concludes that the shedding process is associated with a deep low-pressure valley immediately aft of the cylinder. This implies that this low pressure area can be removed by means of a splitter vane, which in turn stops the vortex shedding. He begins with analysis on flow-induced vibration on the cylinder. He noted that self excitation can occur when bodies are rounded such that the boundary-layer separation points can move on the body surface, emphasizing the role of the position of the separation point. When self excitation does occur, the crossflow velocity is no longer negligible in relation to the streamwise velocity.

Berger and Wile (1972 [15]) also provide an extensive review of periodic flows. They note that, following Bearman, in an attempt to predict vortex-street parameters for various body form, Kronauer proposed that for any vortex velocity the vortex street set itself so as to reach the minimum drag due to the vortex formation, thus departing from Von Kalman's idea based on stability theory. Among others, Gerrard (1966 [14]), in a study of vortex formation behind bluff bodies, proposed the shedding frequency scale is determined by two characteristics lengths: the length of the formation region, and a length termed diffusion length. The first length simply stresses the importance of the entrainment of fluid from the interior of the formation region of vortices behind the bluff body and its replenishment by reversed flow. The diffusion length represents the thickness of the shear layer at the end of the formation region where the layer is drawn across the wake. He also observed that the Strouhal number for vortex shedding with different freestream turbulence levels is almost independent of the Reynolds number.

The vibrating cylinder problem, although usually discussed in terms of 3D flow features, offers several reflexions. First, the system consisting of an obstacle and a periodic wake behaves like a nonlinear self-excited oscillator under forced vibration. Thus, aspects of a nonlinear system formed by an elastic solid body and fluid wake include "synchronization" (lock in), hysteresis, frequency demultiplication, and in certain cases suppression of vortex shedding. Otherwise little is known of the interaction between an

oscillating cylinder and its periodic wake.

Other studies of interaction between a vortex street and bluff bodies and the oscillations induced on the bluff body have been done by Bearman (1984 [61]) and more practically by Sarpkaya (1979 [55]). They highlight the difficulties in establishing a mathematical model for the coupled flow/body system and stress the need for a complete simulation model. Sarpkaya [55] focuses mainly on a numerical study, providing results for a cylinder oscillating transversely in a uniform stream using discrete introduction of vorticity. Bearman [61], on the other hand, is more general and also analyzes the fixed body case.

Lugt was involved in two studies investigating the flow around an ellipse using a streamfunction-vorticity scheme for the Navier-Stokes equations, with both a steady and rotating ellipse. The first, with Haussling in 1974 [25], focused on the steady ellipse at 45 degrees angle of attack with thickness ratios from 1 to 0.1 and Reynolds numbers ranging from 1 to 200. They found that similarly to the cylinder case, when $Re > 45$ the steady state becomes unstable and the flow changes to a Karman vortex street. Also, at $Re = 200$, the inertial effects of the flow were more pronounced than at $Re = 15$ and 30. Those inertial effects manifests themselves in a distortion of the vorticity generation and spreading close to the body, compared to the steady state solution. At this Reynolds number, they noted the zero streamline is parallel to the chord of the ellipse, which is a manifestation of the Kutta condition for viscous flow. The period of the drag and lift coefficient history coincides with the vortex shedding. Effects of pressure dominate over those of friction. The maximum aerodynamic force coefficients occur when the recirculatory region is the largest, which marks the end of the roll-up of vorticity and the beginning of the vortex shedding. This state is characterized by a vorticity extremum. There is also a small maximum in the drag coefficient emerging from the pressure influence and resulting from the proximity of the recently shed vortices.

A moving ellipse study was carried out by Lugt and Ohring [26]. It involved an ellipse of length a rotating at constant angular velocity at a Reynolds number of 200. To characterize the angular velocity $\vec{\Omega}$ compared to the freestream flow \vec{U}_∞ , they used the Rossby number ($|\vec{U}_\infty|/(|\vec{\Omega}|.a)$). The flow was studied from the transient phase until the flow became steady or periodic. For a thin ellipse and $Re = 2000$, the flow reaches

periodic behavior where two vortices are shed at each cycle. At this stage, the minimum for the moment coefficient in a cycle occurs when four recirculatory regions exist, i.e. two suction regions under and above the surface of the ellipse which cancel each other, with the others at the tip of the ellipse acting in an antisymmetric manner.

Ohmi et al [45] did an experimental and numerical investigation of an elliptic cylinder with high angle of attack (in the static stall regime, 15 to 45 degrees). They considered the effect of different parameters of the flow on the oscillation to determine the important components. The Reynolds number ranged from 1500 to 10000. These simulations would be difficult to reproduce, because the flow model in chapter 4 and the control in chapter 5 strategy assume high angles of attack (see section 1.1.1). They noted a weak influence of the Reynolds number and angle of attack on the flow dynamic and pattern with the main influences coming from reduced frequency up to a threshold and then to the product of the reduced frequency by the reduced amplitude.

1.2.5 Motion vibration control schemes

Although stabilization of structural vibrations is not directly a concern par se, these studies can illustrate problems that also appear in other systems involving control of structural motion. For example, both cases are concerned with characteristic values that oscillate at a given period (at least in the case of the fixed plate).

Most of the literature on the subject is dedicated to the control of structural deformation due to the aerodynamic forces or to the control of the flow through various devices like a dynamically deforming airfoil (deforming leading edge to reduce drag for instance), or boundary layer transpiration to delay separation.

Meirovitch and Silverberg (1984) [39] present a method based on the modal control (based on the mode of excitation of the free structure), and a displacement and velocity feedback. They then provide a linear model for the relation between the aerodynamic forces and the structural displacement. Although the formulation is very close to the problem tackled here, its input is limited to a flap angle on a wing and they have no interest in the flow characteristics. Nevertheless, it emphasizes the possibility of real-time control on this type of structure and the fact that the aerodynamic forces in the case of flutter (coupling between the aerodynamic forces and the vibration mode of the

plate) can be modeled through a linear model.

Ballas (1978) [4], proposes an active control method for a distributed parameter system through modal control. However, he also provides an elegant solution to reduce the effect of the spillover (instability due to uncontrolled elastic modes of a structure) and an example with a flexible beam. To apply this approach to the rigid body motion case would require a rewriting of the body displacement equation. Nevertheless, it could prove to be useful if the emphasis was put on the damping ability of the control.

Fromme and Golberg [19], discuss the solution of the equations of motion with structural damping, aerodynamic stiffness and nonlinear generalized forces provided. Then a complete method of resolution is provided but without aerodynamic forces. Although it has a different focus considering the project need, this paper nonetheless supplies an interesting point of view on how to implement and solve the equations of motion.

Ribeiro (1998) [47] models a vibrating structure through a hierarchical method, involving a finite element method with a high degree polynomial as an approximation for the element. This is beyond the scope of the project.

Monaco and Normand-Cyrot (1997) [42] provide a common framework for the study of nonlinear discrete-time and sampled dynamics. Although the simulation in the work described in section 5.3 could be considered as giving some discrete time input, this study is of limited use for the project purposes. Newman (1994) [44] used a distributed controller for the control of structural vibration. Although giving an interesting point of view, it too is not directly relevant to this project.

Chapter 2

The discrete vortex method

2.1 Approach

The vortex element method enables one to recreate the physically relevant dynamics of two-dimensional incompressible flows through the use of the Lagrangian or the Lagrangian-Eulerian description of the evolution of discretized vorticity fields. Helmholtz was the first to show that, in what is now regarded as one of the most important contributions to fluid mechanics, that in an inviscid fluid vortex lines remain continually composed of the same fluid elements and flows with vorticity can be modeled with vortices of appropriate circulation.

As a Lagrangian technique, its advantages lie in the grid-free nature of the simulation, the exact treatment of the far-field boundary condition (if the Vortex In Cell scheme is excluded), and the concentration of the computational power where it is necessary (i.e. vorticity at specific points). On the other hand, the use of vortex methods introduces some error in the convection, and special treatments are needed to take into account viscous diffusion (random walk scheme for example). The method also requires explicit treatment of the turbulent diffusion, otherwise it is limited to preturbulent, almost laminar flow.

There are additional problems associated with use of the vortex method and the assumption of a purely two-dimensional ($2D$) flow. As stated by Sarpkaya [53], the $2D$ method is unable to capture three-dimensional effects such as vortex filament tilting or stretching, whatever the specific scheme considered. This means that without the use of an ad-hoc circulation reduction scheme, it is difficult to predict accurately the

dynamics of 3D flows, in particular the lift and drag forces and the pressure distribution. Nevertheless, the Strouhal number is generally correctly predicted. Note, though, that circulation reduction is strictly an ad-hoc scheme for 2D methods and does not enable to properly emulate the physical 3D effects in the flow thus it is a purely artificial mathematical model. As such, it requires to be adjusted depending on the type of the body and flow. This kind of model is thus somewhat awkward due to the use of purely ad-hoc parameters with no physical means.

Finally, other problems arise due to the fact that for stability reasons the vortices must have a finite core. Unfortunately however, these finite-core or 'blob' vortices violate Helmholtz's law that vorticity is a material quantity. This introduces a formal inconsistency in the dynamics. Furthermore, the nonlinearity in the Navier-Stokes equations does not allow the superposition of finite-core vortices. However, there are some advantages to be gained through the use of blob vortices, in particular a smoothing of the velocity and vorticity distributions, provided that one increases the number of blobs, by forcing them to overlap, and by judiciously choosing the core radius and the shape of the velocity cutoff function. Besides, increasing the number of vortices also mitigates the effect of the Navier-Stokes nonlinearities.

For this study, two different models are considered for separated flow, both of which are modifiable to take into account the effect of a moving plate. The first is based on the discrete introduction of vorticity at a fixed pre-specified separation point using blob vortices to represent both the flow and the plate. This scheme was coded in the C language, with the aim of coupling it with the Matlab utility Simulink. The second was adapted from the algorithm (and Fortran code) of Spalart [57], which represents the boundary layer by blob vortices that are eventually shed into the flow.

Both methods have been used previously to model a moving body in a flow using the vortex method. For the first method, a similar scheme was proposed by Ham [23] to model a flat plate during dynamic stall, as well as by Sarpkaya [51] to model a transversely oscillating cylinder. As for the second method, Spalart [57] used his code to model an airfoil during dynamic stall, Blevins [7] characterized a transversely oscillating cylinder, and Shiels [56] used a similar method with blob vortices with growing cores to account for the viscous diffusion to model a free-falling flat plate as well as the transversely oscillating cylinder.

Both schemes are used for two different roles; the first method is used to further develop and validate the control, while the second method serves as a benchmark for validating the code based on the first method, particularly for moving-body flows. Indeed, the Fortran code could not be directly coupled to a Matlab utility without first translating it into C. However, the second method is much more accurate than the discrete introduction of vorticity at predetermined locations, due to the higher count of vortices and less arbitrary parameters. On the other hand, it is also more difficult to use correctly as there can be significant discrepancies from one simulation to another simply by changing one parameter, for example the timestep, within the same flow condition. In that respect, the first method is more consistent as it does not have such discrepancies; that is why trial were made to couple it with the control.

The main complication with both methods when introducing a rotation is the extraction of the force and moment, as some terms are added, due not only to the rotation itself but also to the frame of reference used when modeling the flow around the plate.

2.1.1 Discrete introduction of vorticity

In programming the C simulation, a scheme was adapted based on the model of Sarpkaya [50]. He used a potential-flow model of $2D$ vortex shedding, using image vortices and a Joukowski transformation to model the unsteady flow past a stationary plate. The vortices present in the flow are convected using a complex potential. At each timestep two nascent point vortices are generated to account for the effect of separation. For a flat plate in a transverse flow, the separation is located at the two edges of the flat plate, which will be referred to as the separation points. The strengths of the nascent vortices shed at the separation points are computed through an approximation of the shear layer velocity at these points. The position of the nascent vortices is then determined by enforcing the Kutta equation at the separation points on the body surface. At this point, it is important to distinguish the separation points which are defined as the points *on the body surface* where the flow separation is supposed to take place, and the discrete vortices creation points which are the points *in the flow domain without the body (including the body wall)* where the nascent vortices are positioned in order for their induced velocity field to enable the instantaneous satisfaction of the Kutta condition at the separation points.

Sarpkaya [53] provides a very good discussion about this kind of method. It appears well suited to modeling a flow around a body with sharp edges where the determination of the separation location is not a problem. Additionally, there is a feedback from the vortex street to the nascent vortex, dynamically affecting its strength and initial position. This is a crucial point, as the initial position of the nascent vortices is of major importance on how the vortex sheet rolls up, as stressed by Gerrard [21]. It should be added that the normal force determination is also very sensitive to the distance from the plate of the nascent vortex.

However, some of the inconsistencies of the scheme are that it does not relate the time step Δt to both the distance of the nascent vortex to the body ϵ and the rate of vorticity $\Delta\Gamma$ (the latter, indirectly through the determination of an approximated shear layer velocity). Furthermore, there is no explicit treatment of the breakaway angle between the separation streamline and the body surface tangent, as well as for the velocity distribution in the vicinity of the separation point after the introduction of the nascent vortex.

Also, it is hard to establish proof of the convergence for this kind of method for a given set of parameters, and such proof have not been found in the literature cited in the current work. Nevertheless, observe that even for more general vortex methods such as the Spalart code, it is not thought possible to prove convergence for a given set of parameters (see Spalart citation in section 2.2.2) despite the existence of demonstration of the convergence of the method (see Koutsoumakos [13] and section 2.2.2). Note that Koutsoumakos [13] clearly stated that even for undersolved system the method is expected to give good qualitative results. This should be considered regarding the sections related to the numerical parameters (notably section 2.3) and the flow results presentation (chapter 3 and section 5.4).

Still, this model produces satisfactory results for a cylinder [49] and for a flat plate [50], despite overestimating the resulting aerodynamic forces. However, for long-time the simulation tends to become unstable due to the singularity of point vortices. Furthermore, the force and moment evaluation are dependent on the velocity close to the wall surface. Blob vortices were thus used both for flow and boundary modeling to

help maintain stability by smoothing the velocity field. Indeed, as pointed out by Sarpkaya [53], this helps to achieve convergence provided that the blobs sufficiently overlap. However, one must keep in mind that this scheme does not ensure a smooth roll-up, and constitutes a mathematical artifice to smooth the velocity field. As stated earlier, the nonlinear nature of the Navier Stokes equations does not formally permit such superposition of vorticity fields.

Another modification of the Sarpkaya scheme is that instead of using a complex potential for modeling the body boundaries, a boundary element method was used in which the strength of blob vortex elements is chosen to satisfy a no-penetration boundary condition. The main drawback of this scheme is that the use of blob vortices allows a net flow leakage into the body, although in practice it is small enough so that it does not exert a great influence on the flow. On the other hand, it has some real advantages compared to the use of the complex potential. First of all, it allows the use of other types of vortices than the point vortex. Another benefit is that it avoids the use of image vortices required by the complex potential, which can introduce some artificial flow phenomena [57] when a near-wall vortex is too close to its own image. Finally, despite being of lesser importance for the thin ellipse case, it is more flexible concerning the geometry of the body.

As a last remark, Sarpkaya in his review [53] noted that the Strouhal number does not seem to be overly sensitive to the position of the nascent vortex when introduced into the flow. This is important for the current study, since the emphasis here is on the frequency characteristics of the fluid/structure system, even if it does not guarantee a proper modeling of all of the dynamic characteristics. Again, Sarpkaya [53] provides a simple example whereby the normal-force coefficient on a steady cylinder shows unexpected (but reasonably small) oscillations compared to experiment. This is also why a comparison is made of this model against the well-established model from Spalart, in order to assess the behavior of the Sarpkaya based method. Nevertheless, one must keep in mind that this is another vortex method, and as such will presumably have some of the limitations inherent in all vortex-method predictions of bluff-body flow dynamics.

Note that due to the choices made, the simulation developed specifically for the current project is often referred to as “the Sarpkaya-like simulation” by contrast to the “Spalart simulation”.

2.1.2 Boundary vortices convected in the flow

Spalart, instead of having discrete introduction of vorticity from a nascent vortex, used vortices shed from the boundary layer. The boundary is modeled by a boundary element method with a no-flow-through condition, and the elements used are blob vortices with a Gaussian core function. Two alternative programs are available: KPD1 and KPD2. They have both been created by P.Spalart and A.Leonard, and are available freely from COSMIC (a part of NASA). Their details are explained in reference [57]. KPD1 is the basic bluff-body code with no boundary-layer treatment, whereas KPD2 has an integral-method treatment of the boundary layers able to control separation and is therefore able to handle viscous flows, such as a smooth surface separation. However, the KPD2 does not seem to take into account rotating bodies, and furthermore requires the use of ad-hoc parameters. Therefore, the KPD1 code was used for the rest of this study.

Concerning program KPD1, note that Spalart [57] points out that even in the absence of any viscous model, the vortex method is able to recreate “pseudo-viscous” behaviour. Two major reasons are involved. Each vortex is created near the wall so that whenever there is a deceleration of the boundary layer, a vortex acquires a small velocity component directed away from the wall, which initiates the separation. The second reason is that time-integration errors move the vortices away from the wall, especially with large velocities and strong streamline curvature (as for a transverse flat plate) as it causes a rapid acceleration and deceleration of the boundary-layer vorticity, which causes separation. He further added that this behavior is consistent near a sharp edge or high curvature. This shows the feasibility for the vortex method to reproduce viscous separation, even in the absence of explicit viscous diffusion. Nevertheless, one should note that the shedding of vorticity can be either transient (as for a starting airfoil) or permanent (as in a flow past a square). Indeed, the separation from a smooth surface is much more delicate to handle, which motivates attempts to control or delay separation.

Turning to KPD2, the integral-method treatment of the boundary layers has been devised in order to predict the separation point on a body surface, able to take into account the effect of Reynolds number and thereby affect separation position. The method

is based on the computation at each time step of the pressure distribution on the body, which is used to define the behaviour of the boundary layers upstream of separation. The next step is to check if the vortices are located upstream of separation, in which case they are deleted and their circulations recorded to be injected back on the wall surface through the boundary condition. This is completed by use of equation (2.31) below. Conversely, beyond the separation point the vortices are free and subjected to the straining associated with the adverse pressure gradient, and they leave the vicinity of the wall. Further details and references are given in [57].

This method shows many of the characteristics typical of a vortex method in $2D$, e.g. over-prediction of the force coefficients, in some cases an inaccurate prediction of either the dynamics of vortex shedding (especially if $3D$ effects are important), or simply of the flow structure if separation is present (for example in the case of a fixed flat plate at low incidences). However, Bearman [61] and Spalart [57] for bodies oscillating transversely to a flow and for an airfoil in dynamic stall, respectively, indicates that the structure of the flow becomes more two dimensional with oscillation. The scheme remains a good basic tool to predict separated flows behind bluff bodies.

2.1.3 Geometry for the simulation

For this study, the geometry considered is sufficiently thin to approximate a flat plate, but thick enough so as to have non-negligible inertia. An analytical shape is required, to simplify for example the calculation of the surface force distribution. An ellipse was chosen with a ratio of length $2a$ to thickness $2b$ of 20 to 1.

Based on transverse flat-plate results, the ellipse is expected to generate unsteady vortex shedding for Reynolds number (based on $2a$) between 200 and 11000 (Lugt and Haussling 1975[34], Sarpkaya 1975 [50], Blevins [7]). In this range, the flow is not yet fully turbulent and is thus still within the limit of vortex methods.

However, here no viscous model will be used so as to keep the simulation as simple as possible. The ellipse and model geometry are shown in figure 2.1, where \vec{x} and \vec{y} form the body fixed frame of reference. Note that in the case of the ellipse, \vec{x} is also called the major axis of the ellipse, and \vec{y} the minor axis.

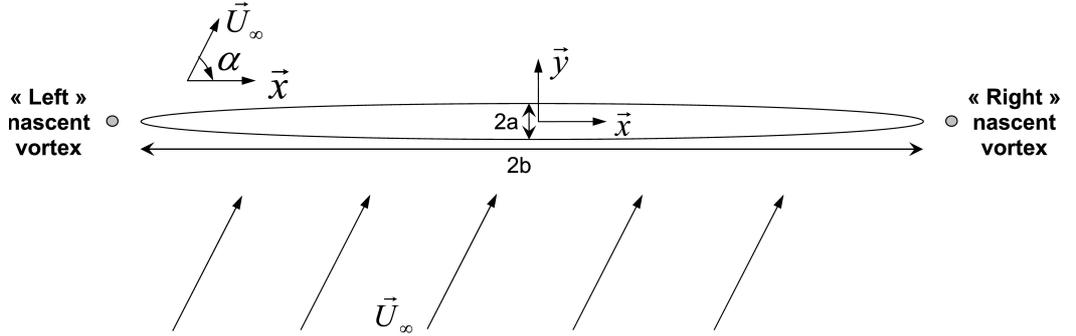


Figure 2.1: General geometry

2.2 General theory

2.2.1 Governing equations

For an incompressible fluid the mass conservation equation becomes

$$\nabla \cdot \vec{u} = 0, \quad (2.1)$$

where \vec{u} is the fluid velocity and ∇ is the gradient operator. The incompressible Navier-Stokes equations for conservation of linear momentum are usually written in pressure-velocity form,

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = \frac{1}{\rho} \nabla p + \nu \nabla^2 \vec{u} + \vec{f}, \quad (2.2)$$

where ν is the kinematic viscosity, p is the pressure, ρ the fluid density and \vec{f} the body force per unit mass. However, an alternative is to look at the vorticity $\vec{\omega}$, defined as the curl of the velocity field

$$\vec{\omega} \equiv \nabla \times \vec{u}. \quad (2.3)$$

Taking the curl of 2.2 and neglecting \vec{f} , the vorticity form of the Navier-Stokes equations is obtained

$$\frac{\partial \omega}{\partial t} + (\vec{u} \cdot \nabla) \omega = (\omega \cdot \nabla) \vec{u} + \nu \nabla^2 \omega, \quad (2.4)$$

where ω is the magnitude of $\vec{\omega}$, which in 2D is $\vec{\omega} = \omega \vec{k}$ where $\vec{k} = \vec{i} \times \vec{j}$ is the unit vector normal to the plane. The vorticity $\vec{\omega}$ is subject to initial conditions

$$\vec{\omega}_o \equiv \nabla \times \vec{u}_o, \quad (2.5)$$

where \vec{u}_o is the initial velocity field. For flows in contact with solid, impermeable boundaries the no-slip boundary condition must also be satisfied, with

$$\vec{u} = \vec{u}_s(\vec{s}_b, t) = \vec{0}, \quad (2.6)$$

where \vec{s}_b is the position vector that defines the location of the surface boundary, and \vec{u}_s is the tangential velocity component on the boundary at $\vec{x} = \vec{s}_b$. The no-penetration condition must also be satisfied

$$\vec{u} = \vec{u}_n(\vec{s}_b, t) = \vec{0}, \quad (2.7)$$

where \vec{u}_n is the normal velocity component on the boundary.

Free-stream boundary conditions at infinity are defined as

$$\vec{u} \rightarrow \vec{U}_\infty(t) \text{ as } |\vec{x}| \rightarrow \infty. \quad (2.8)$$

In two dimensions the $(\omega \cdot \nabla) \vec{u}$ term, representing the stretching of vortex lines, is zero as the vorticity is normal to the velocity gradient. Therefore, from (2.4) one obtain

$$\frac{\partial \omega}{\partial t} + \vec{u} \cdot \nabla \omega = \nu \nabla^2 \omega. \quad (2.9)$$

Now, if one rewrite this equation in term of the Lagrangian derivative, one obtain

$$\frac{D\omega}{Dt} = \frac{\partial \omega}{\partial t} + \vec{u} \cdot \nabla \omega = \nu \nabla^2 \omega, \quad (2.10)$$

illustrating that the vorticity is only changed by diffusion. Hence, the vorticity magnitude of a particle moving in an inviscid flow remains constant in time, as

$$\frac{D\omega}{Dt} = 0. \quad (2.11)$$

Using equations (2.1), (2.3) and (2.8), it is necessary to now look for a way to relate velocity to vorticity. A classical way is to use an integral representation. Turning back to the mass conservation equation (2.1), one see that it can be satisfied by a scalar stream function, $\psi(\vec{x}, t)$, such that

$$\vec{u} = -\nabla \times (\psi \vec{k}). \quad (2.12)$$

Hence, one can obtain a Poisson equation relating the stream function to the vorticity

$$\nabla^2 \psi = -\omega. \quad (2.13)$$

Using a Green's function approach, one can solve for the stream function ψ for flows in free space :

$$\psi(\vec{x}, t) = \int_D (G(\vec{x} - \vec{x}')) \omega(\vec{x}', t) d\vec{x}', \quad (2.14)$$

where R_∞ is the entire domain, G is the Poisson kernel satisfying

$$\nabla^2 G(\vec{x}) = \delta(\vec{x} - \vec{x}'), \quad (2.15)$$

δ is the two-dimensional Dirac delta function and ∇^2 is the Laplacian operator. For two-dimensional cases the Poisson kernel is of the form

$$G(\vec{x} - \vec{x}') = -\frac{1}{2\pi} \ln |\vec{x} - \vec{x}'|. \quad (2.16)$$

Letting $\vec{K} = \nabla \times (G \vec{k})$, one can express the velocity in terms of the vorticity by means of a Biot-Savart integral such that $\vec{u} = \vec{K} * \omega$, where $*$ denotes a convolution

$$\vec{u}(\vec{x}, t) = \frac{1}{2\pi} \int_{R_\infty} (\vec{x} - \vec{x}') \omega(\vec{x}', t) d\vec{x}'. \quad (2.17)$$

That is, if one take into account the boundary condition (2.8),

$$\vec{u}(\vec{x}, t) = \int_{R_\infty} \vec{K}(\vec{x} - \vec{x}') \omega(\vec{x}', t) d\vec{x}' + \vec{U}_\infty. \quad (2.18)$$

Using the 2D version of (2.16), remembering that $\vec{K} = \nabla \times (G \vec{k})$, then one obtain

$$\vec{u}(\vec{x}, t) = \frac{1}{2\pi} \vec{k} \times \int_{R_\infty} \frac{\vec{x} - \vec{x}'}{|\vec{x} - \vec{x}'|^2} \omega(\vec{x}', t) d\vec{x}' + \vec{U}_\infty. \quad (2.19)$$

Kelvin's theorem, which asserts the conservation of circulation in an unbounded flow, provides the other important result needed for vortex-method calculations:

$$\frac{d}{dt} \int_{R_\infty} \omega d\vec{x} = 0. \quad (2.20)$$

This theorem, defining the net global circulation, provides the constraint needed when new vortex elements are introduced into the calculation.

The strategy in vortex methods is then to partition the computational domain into cells in which the initial circulation is concentrated in a single point or particle. In a Lagrangian framework, one can now discretize the flow and calculate the evolution of the computational vortex elements based on the local fluid velocity (equation (2.17)). The strength of the particles stays constant through the simulation (equation 2.11), and is determined through the use of boundary conditions (equations (2.6) and (2.7))

and initial conditions (equation (2.5)). The global introduction of circulation is controlled through equation (2.20).

2.2.2 Discretization of the vorticity field

In two-dimensions, vortex methods rely on spatially discretizing the vorticity field into small areas of rotating fluid known as discrete vortices. These mathematical artifacts represent regions of fluid whose area is (ideally) smaller than the finest length scales in the flow, to be able to model the system accurately. The Lagrangian grid is comprised of these discrete vortices, with their number and spacing being parameters analogous to the grid-resolution of Eulerian CFD methods. The overall vorticity field can therefore be mathematically approximated by

$$\omega(\vec{x}, t) = \sum_{j=1}^N \Gamma_j \gamma(\vec{x} - \vec{x}_j(t)), \quad (2.21)$$

where N is the total number of discrete vortices, γ is the vorticity distribution or core function of the j^{th} discrete vortex at $x_j(t)$, and Γ_j is its strength. Substituting the approximate vorticity (2.21) into (2.17) gives the velocity field

$$\vec{u}(\vec{x}, t) = \sum_{j=1}^N \Gamma_j \vec{K}_\delta(\vec{x} - \vec{x}_j(t)), \quad (2.22)$$

where $\vec{K}_\delta = \vec{K} * \gamma$. Note that the smoothness of \vec{K}_δ is dependent upon the vorticity distribution γ . Implementing the Dirac delta function as γ results in the classical point vortex method. Its interest lies in that the dynamics of point vortices provides an exact albeit singular solution of the incompressible Euler equation in $2D$. However, the point vortex method is numerically unstable after a certain amount of time for $N > 4$ due to the singular nature of the vorticity distribution. However, if one chooses a smooth bounded distribution instead of the Dirac function the vortex can still be useful. This is widely known as the vortex blob method, and in general one chooses

$$\gamma(\vec{x}) = \frac{1}{\sigma^m} \phi\left(\frac{\vec{x}}{\sigma}\right), \quad (2.23)$$

for some function ϕ that integrates to unity. Leonard [33] suggested that ϕ be radially symmetric to simplify evaluation of the velocity field. The parameter σ determines the so called support of γ and is often referred to as the cut-off radius, or vortex core radius. The order of the kernel is determined by m . Assuming the exact solution of the velocity field is sufficiently smooth, then the properties of ϕ determine the accuracy of

the vortex method. Thus, it is very important that the vortex core is chosen carefully, to balance accuracy with computational efficiency. A well-tested core is the Lamb, or Gaussian, vortex which has a vorticity distribution

$$\gamma(\vec{x}) = \frac{1}{\pi\sigma^2} e^{\left(-\frac{|\vec{x}|^2}{\sigma^2}\right)}. \quad (2.24)$$

This core is therefore second-order accurate in σ and has been used successfully by many authors to construct discrete vortex methods to approximate solutions of the Euler and Navier-Stokes equations in two-dimensions.

Stability and convergence of this method to the Euler and Navier-Stokes equations for unbounded flow has been demonstrated by Hald [22], Beale and Madja [5], Leonard [33], Chorin [10] and Koutsoumakos [13], among others. Leonard focused on the consistency of methods whereas Hald, Beale and Madja obtained complete convergence proofs including stability. Chorin's proof, albeit restricted to $2D$ unbounded flows, also provides some insight into the evolution of the error in time and highlights the fact that the error in a vortex method is primarily due to errors in evaluating (2.18). More recently, Koutsoumakos has given a thorough discussion and demonstration of the convergence related to the discretization of the vorticity field.

As stated previously, one source of error of the blob vortex method is the nonlinear nature of the Navier-Stokes equations. Superposition of non-point element solutions will by definition introduce inaccuracies that are related to the size of the vortex cores. This also means that in a vorticity field modeled by the sum of fluid elements, the vorticity distribution of each fluid element should deform in the flow. However in $2D$ incompressible inviscid flow, the whole blob moves at the same velocity, whereas in reality the fluid region that coincides with it is strained.

In addition to the small-core accuracy constraint is the need for the vortices to overlap more and more as they get closer and closer, and the requirement of a constant core radius for the vortices if one chooses such classical convection scheme as equation (2.27). The first condition is especially important to achieve smooth velocity and vorticity distributions in order to mitigate the effect of nonlinearity. Otherwise, if the spacing between vortices is too large, the method effectively behaves like a point vortex method.

In the case of discrete introduction of vorticity, the core radius is chosen to be small because it is required for the boundary vortices to properly model the boundary. Nevertheless, it is expected that there will not be enough vortices to have a proper overlap in the whole flow. The benefit of using blob vortices will be for the wall model, as well as the smooth velocity distribution inherent to blob vortices which will bring further stability. Nonetheless, it also means that as Sarpkaya [53] remarked, the simulation may be limited in time and although it may be longer than when using point vortices, its time range of validity will have to be assessed. Apart of these properties, the C algorithm is expected to behave similarly to the Sarpkaya algorithm.

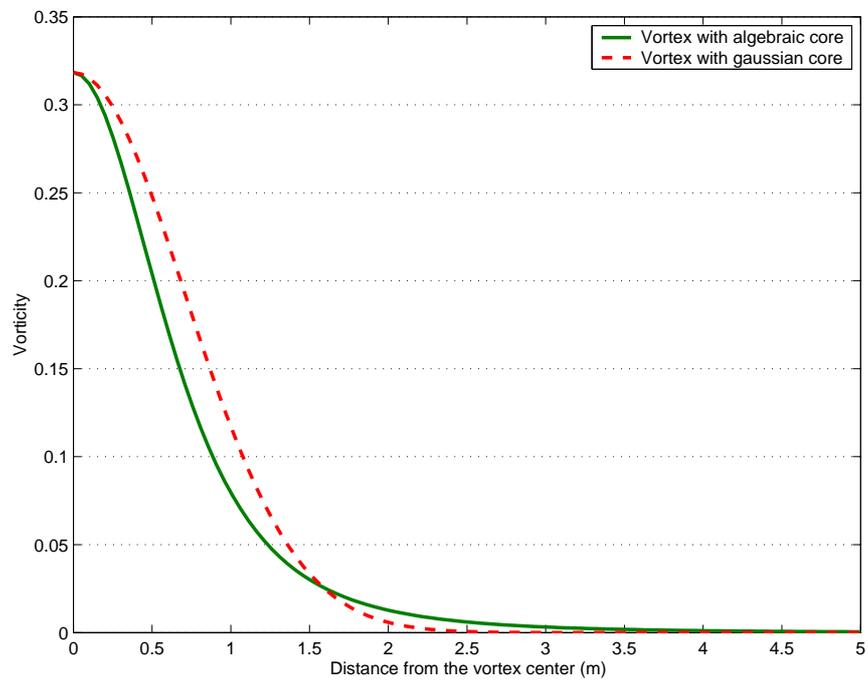
Now, as for the Spalart code, which is effectively the full vortex method, Spalart [57] remarked that the method rarely “blew up” and therefore does not provide a strong warning to the user. The vortices simply look disordered. Therefore, establishing the appropriate range of parameters is difficult. To quote Spalart [57] : “...we were unable to improve them [the results] or to convincingly demonstrate the convergence of the method, even via large variations of the numerical parameters ($N, \Delta t, \delta_0$, etc.). ”. Assessing the effect of nonlinearities and overlapping is thus difficult, and the determination of parameters such as the core radius is often done empirically (Spalart [57], Clarke [11], Takeda [59]); it is interesting to note that Leonard [33] provides some tools to first determine the number of vortices at the boundary.

In order to keep the program as simple as possible, the first choice was to use a Gaussian core for the simulation and increase the accuracy by using as many vortices as possible, and putting the emphasis on overlapping the vortices. A higher-order core would increase the accuracy but at the expense of more computational effort. One could also consider using a different core radius, however that is ill advised because, as Leonard [33] remarked, it prevents one from conserving even the most basic flow invariant such as the linear momentum.

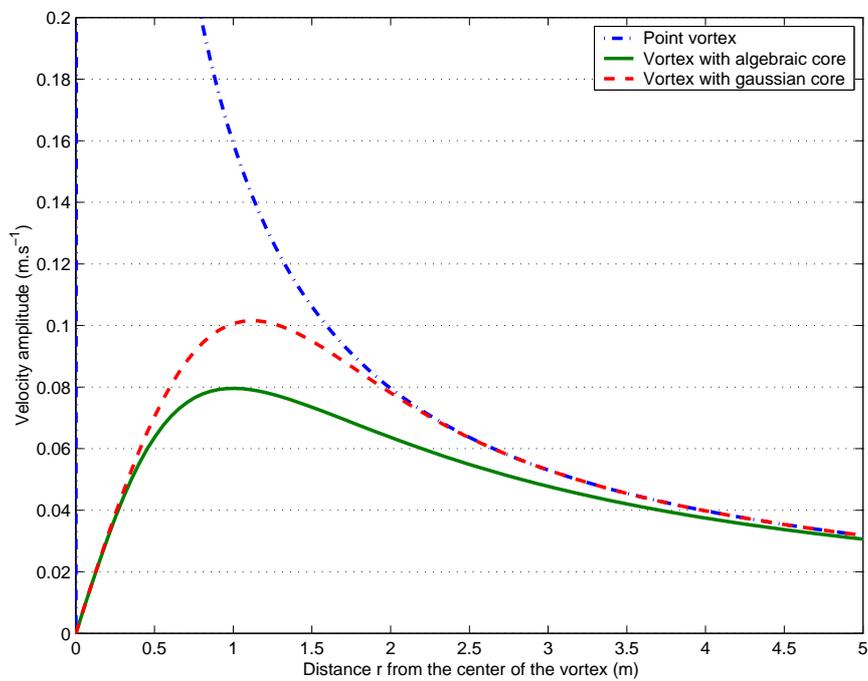
Calculation of the exponential function for the Gaussian core has a particularly high overhead on modern RISC processors, so Takeda [59] proposed a computationally efficient alternative algebraic core with similar properties :

$$\gamma(\vec{x}) = \frac{1}{\pi\sigma^2 \left(1 + \left|\frac{x^2}{\sigma^2}\right|\right)^2}. \quad (2.25)$$

This is significant as the code spends a large proportion of time evaluating the core function for each vortex. The difference in the form of these vorticity distributions of these two cores is minimal; see figure 2.2. This simulation as well as the Spalart code all used the algebraic core (2.25).



(a) Vorticity distribution



(b) Velocity distribution

Figure 2.2: Comparison between different vortex cores

2.2.3 Convection of the vortices

For flows without boundaries, the vortex method for inviscid flow reduces to evolving the positions of the discrete vortices with time, that is to say solving $2N$ non-linear ordinary differential equations, based on equation (2.19)

$$\frac{d\vec{x}_i}{dt} = \vec{U}_\infty - \frac{1}{2\pi} \sum_{j=1, j \neq i}^N \frac{(\vec{x}_i - \vec{x}_j) \times \vec{k} \Gamma_j \vec{K}_\delta(\vec{x}_i - \vec{x}_j)}{|\vec{x}_i - \vec{x}_j|^2}. \quad (2.26)$$

One should note that Leonard [33] devised a different scheme using the vorticity-weighted average of the velocity over the blob, which allows more flexibility with the core radius, and satisfies the flow energy invariant. However, this requires much more computational effort. Thus, for the sake of computational speed, it was chosen to stick to equation (2.26).

In vortex methods, one uses in general an explicit method for time integration. This is because most of the computational cost is due to the evaluation of the time derivative, and the algorithm requires little memory. Moreover, an implicit scheme would require the inversion of equation (2.26), or at least its linearization, which would be very complex and expensive. Hence, a majority of previous studies use a multi-step scheme such as second or fourth-order Runge Kutta, second-order Adams-Bashforth or first-order Euler. An Adams-Bashforth scheme was used for it represents a good compromise between accuracy, simplicity and cost. The position of each vortex is thus updated according to

$$\vec{x}_j(t + \Delta t) \approx \vec{x}_j(t) + \Delta t \left(\frac{3}{2} \vec{u}(\vec{x}_j, t) - \frac{1}{2} \vec{u}(\vec{x}_j, t - \Delta t) \right), \quad (2.27)$$

where Δt is the time step value.

2.3 Implementation of the method

In this section, the model choices are explained which have been made for the simulation. For detail of the Spalart approach, which is also used, the reader is invited to consult his paper [57]. Major differences between the two schemes will be mentioned below.

Clearly, parameters value for both schemes were searched empirically. Indeed, for each parameter, one should keep in mind the remark in section 2.1.1 concerning the lack

of convergence proof for a given set of parameters despite a proof for the general convergence of vortex methods. This remains relevant for either the discrete introduction of vorticity (Sarpkaya-like code) or the Spalart-code. This points out the difficulty of parameters determination and is fundamental for the results interpretation.

2.3.1 Boundary conditions

The focus of this project is a bluff body moving in a flow. Blevins [7] showed how the vortex method could model at least qualitatively such a flow with a moving or a vibrating body. This approach allows to solve the Navier-Stokes equations to determine the flow near the body, the flow at infinity and the motion of the body.

As the focus is on an impulsively started flow, the initial conditions on velocity and vorticity are zero everywhere at $t < 0$. This produces a steady inviscid non-lifting solution just after the beginning of the simulation.

2.3.1.1 Far field

Here, one must recall that the case considered is that of an unbounded flows past a bluff body. In this case, one of the main advantages of the vortex method is that the boundary condition far from the body is solved exactly. The flow induced by the vortex elements decays to zero in the far field, so the boundary condition at infinity is automatically set to be a uniform flow.

Also, all the flow elements are mathematically valid at any distance from the body, because no grid is used. Therefore, there is no extra computational cost associated with accurately specifying the flow far from the body.

2.3.1.2 Near wall

In an unbounded flow, the velocity is zero at the body surface due to the inviscid no-penetration condition (equation (2.7)) and the viscous no-slip condition (equation (2.6)). The no-penetration condition can be satisfied either approximately – using an element method – or exactly using a conformal transformation to a circle and an image system inside the circle. The no-slip condition cannot be exactly enforced since an inviscid flow is considered. Nevertheless, it is possible, for a fixed body, for both conditions to be satisfied in an inviscid $2D$ flow, as will be explained below.

The modeling of the flow in the vicinity of the wall is of particular importance, since if the flow field is too “noisy” the model will fail to capture correctly such features as the separation points or will add difficulties in the enforcement of the Kutta condition ($D\vec{V} = 0$ at the edge of the plate, see section 2.3.4).

Considering the conformal transformation, the solution is exact (no leakage into the body), but it requires an image system within the body, which increases the computing power required. Furthermore, Spalart [57] maintains that an image system can introduce a large error in the flux of vorticity along the boundary layer, because a vortex and its image can spuriously approach and convect each other too strongly.

On the other hand, the panel method requires one linear-matrix decomposition at the start of the calculation, but can be applied at a fixed cost per element thereafter. Whereas the blob vortex method is a discrete distribution of vorticity along the boundary, the panels represent a continuous distribution. Therefore, similarly to blob vortex methods, the use of vortex panels reconstruct the physical processes of a viscous boundary layer, causing vortices to separate correctly from sharp corners.

By analogy with the vortex panel, blob vortices can also be positioned slightly away from the body boundary. The main drawback in this case is that a larger number of elements has to be created to satisfy the overlap criterion (see section 2.3.2). However, as stated in the previous sections, the use of nonsingular elements to model the boundary helps to smooth the flow field near the body.

Therefore, in order to preserve a low computational cost and relative simplicity in the code to allow arbitrarily shaped bodies, it was decided to use the approach of the element methods rather than the conformal transformation. A choice had to be made between different sorts of elements:

- A straight source panel, with constant strength along the length
- A straight vortex panel, with constant strength along the length
- A curved vortex panel, with linear variation along the length as developed by Clarke [11]
- Blob vortex distributed regularly along the wall

A comparison and a discussion of the different boundary elements is given in section 2.3.1.3. Eventually, the blob vortex method was used, because it could deal more easily with the no-through-flow condition. Furthermore, the creation of discrete vortices at the edges of the ellipse near the wall boundary introduces some perturbations in the velocity field. It was found that applied to thin geometries (thin ellipse as the body, unbounded inviscid 2D flow), the blob vortex method is less sensitive to this perturbation, probably due to the ensuing vorticity distribution helping to smooth the velocity field.

As a general description of an element method, suppose that one has a vortex element of unit strength with velocity distribution $\vec{v}(\vec{r}, P_b)$ (P_b being a set of parameters defining the position and orientation of the element, \vec{r} the position where the velocity is calculated). Choose a set of N_b positions $P_{b,j}$ for elements just outside the boundary. As a mathematical tool, the element could equally well be just inside the boundary, but later the panels are used as a source of vorticity in the flow; placing the elements outside gives the correct sign for the generated vorticity (see figure 2.3). Next choose N_b control points \vec{x}_i on the body contour, with corresponding unit tangent \vec{s}_i . The external flow is denoted $\vec{U}_e(\vec{r})$.

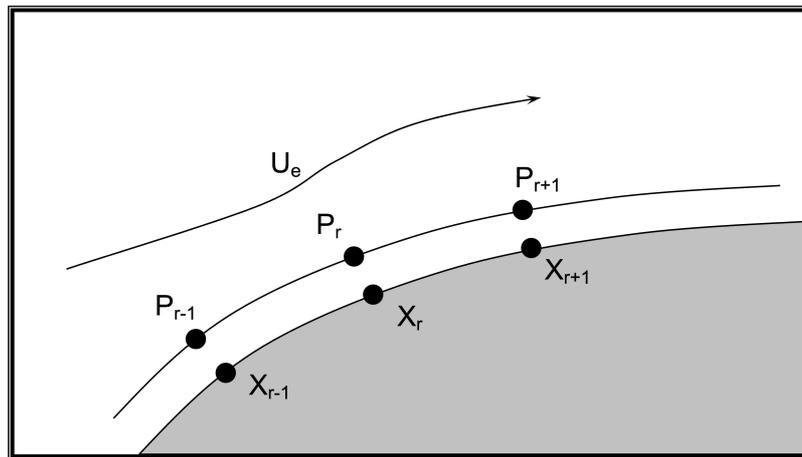


Figure 2.3: Example of disposition for the panel method (courtesy of Takeda)

For the no-penetration condition, several formulations are available. One could cancel the normal velocity at or near the control points (equation (2.7)), or use the streamline formulation :

$$\psi = 0 \text{ , along the boundary.} \quad (2.28)$$

That is, between two control points one has :

$$\psi(\vec{x}_i) = \psi(\vec{x}_{i+1}). \quad (2.29)$$

This second method is much more robust, as one is now working in the integrated flow domain; it also provides a well-structured system matrix, it was thus chosen. This is the method used by Spalart [57] to model the boundary, using blob vortices as flow elements.

Usually in an unbounded flow, there are three components to the flow streamline function, first the stream function from the uniform freestream velocity ψ_∞ , second the stream function associated from the boundary vortices ψ_b , and finally the stream function from the flow vortices ψ_f . Note that a slight generalization is possible in 2D flows: if there is a component with constant vorticity in the freestream flow, then the vortex method can still be applied to the deviation from the freestream vorticity. Now that the terms of the freestream function are known, one can form a set of linear equations to determine the strength of these vortices, which will exactly satisfy the boundary condition at the control points. Equation (2.29) becomes:

$$\psi_b(\vec{x}_{i+1}) - \psi_b(\vec{x}_i) = \psi_f(\vec{x}_i) - \psi_f(\vec{x}_{i+1}) + \psi_\infty(\vec{x}_i) - \psi_\infty(\vec{x}_{i+1}). \quad (2.30)$$

So far one can only have $N_b - 1$ equations, which arise from implementing the non-penetration condition. Since the total circulation is unknown, one can find the N_b^{th} equation by applying Kelvin's Circulation theorem (equation (2.20)) for a nonrotating body in 2D. Then, assuming that the Kelvin condition was met at the previous time step, at the i^{th} time step one has :

$$\sum_{k=1}^{N_b} \Gamma_{b_k} = - \sum_{k=0}^N \Gamma_{f_k}, \quad (2.31)$$

where Γ_b is the circulation on the boundary and Γ_f is the wake circulation, N_b is the number of vortices at the boundary and N – only in this subsection – the number of vortices in the flow (including the nascent vortices). Additionally, once a vortex is shed into the flow, there is no circulation-reduction scheme applied to its strength. Note that condition (2.31) is not unique and that other conditions can be applied, such as symmetry. Spalart [57] provides a discussion on this topic.

One now has N_b equations for N_b unknowns, and generally the system obtained from these equations is well conditioned. Therefore, to solve this system a QR decomposition

was used for the project for simplicity. Since the system does not change with time, another advantage is that the QR decomposition needs to be done only once at the beginning of the computation.

Due to the nature of the Laplace equation, which is being solved here (considering the problem in terms of stream function), satisfying the no-penetration condition has the side effect of satisfying the no-slip condition, provided that the velocity inside the body is zero (see Katz and Plotkin [30], and Spalart [57]).

For the other element methods considered, such as the vortex panel or source panel methods, the boundary conditions were used formulated in terms of velocity. This enables to have N_b equations to which equation (2.31) was added (except of course in the case of source panel methods). In these cases, a Singular Value Decomposition (SVD) method was used to solve the $N_b + 1$ equations. Although the SVD requires a large computing time, it needs to be done only once. Its main advantage is that if a system is badly conditioned, this method helps to remove the equation causing the singularities enabling a smooth solution.

As a special remark in the case of the straight vortex panel, note that enforcing the no-penetration leads to a badly conditioned system, which was alleviated by implementing the no-slip condition. As stated above, in $2D$ inviscid flow the two boundary conditions are equivalent inside the body.

2.3.1.3 Boundary representation

Determining the best way to model the boundary of the ellipse and specifying the proper elements was no trivial task. Many problems arose, mainly due to the high curvature at the edges of the ellipse, which induces in return a high curvature in the velocity field. Furthermore, as discrete vortex elements are shed from the ellipse at the point of separation, the nascent vortices were created close to the wall, which can adversely affect the smoothness of the velocity gradient and impedes the convergence of the method.

To choose between the different methods, several criteria were used:

- Global convergence of the flow, i.e. physical relevance of the long term flow (cf table 2.1).

- Strength of the nascent vortices at the end of a time step. This effectively represents the vorticity shed into the flow. Because it is computed from the existing velocity field at their creation points, e.g. near the plate, it helps to assess the flow near the plate (smoothness, structure, etc)
- Behavior concerning the introduction of nascent vortices at the edges of the ellipse, a typical symptom being that the nascent vortices are created too far away from the edges. Indeed, their position is computed from an initial position near the edges of the ellipse. Thus, the calculation is very sensitive to the smoothness of the velocity field near the edges.
- The form of the velocity field for the steady flow (without any vortices), in particular the amount of leakage through the body wall.
- Ease with which motion of the ellipse can be introduced.

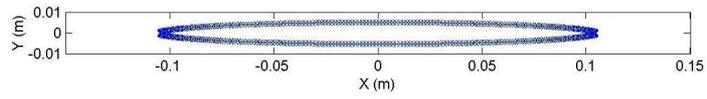
Now concerning the discretization of the ellipse geometry, the ellipse formula was chosen to determine the position of the control points:

$$\vec{x}_i = \overrightarrow{(a * \cos(\theta_i); b * \sin(\theta_i))}. \quad (2.32)$$

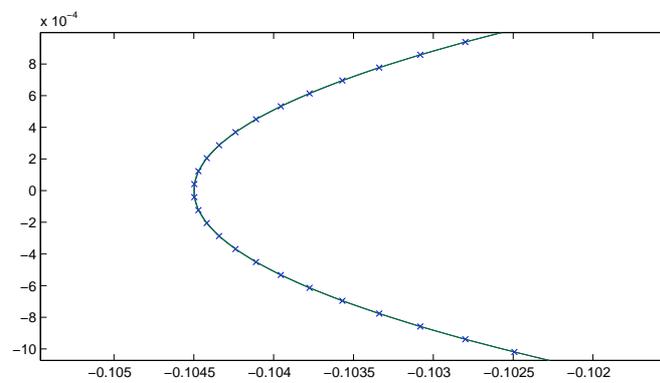
Hence, one can use θ_i to decide the location of the control points. It was decided to apply more panels on the edge of the ellipse where the flow needs greater details. Nevertheless, as a rule of thumb in the case of element methods, it is better to keep the distance variation between two adjacent panels under a factor of two.

Regarding the straight source and vortex panel, the panel coordinates are applied directly on the coordinates of the ellipse (figure 2.4), with the control points being on the center of the panels. Additionally, to obtain better results, the panel at the edge of the ellipse must be “flat” as in figure 2.4(c), as opposed to figure 2.5(c). This was found to provide a better system matrix after applying the boundary conditions. One of the side effects of the slight offset from the control point to the actual ellipse surface is the artificial removal of the discontinuity at the point where the Kutta condition is solved (cf section 2.3.4.2), i.e. the tip of the ellipse.

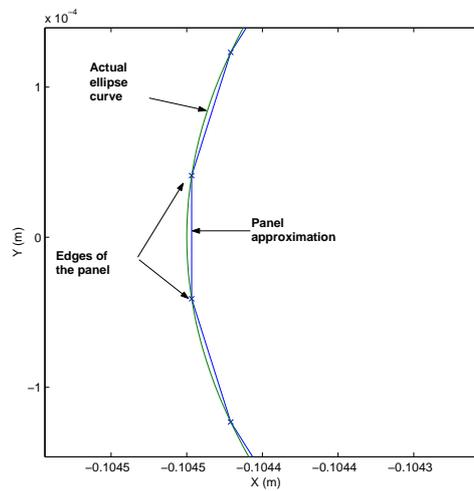
A different technique was used for the curved vortex panel and the blob vortex method. Here, the procedure was to choose first the distribution of the control points and then to place the panel or the vortex at a certain distance along the normal (figure 2.5).



(a) General view

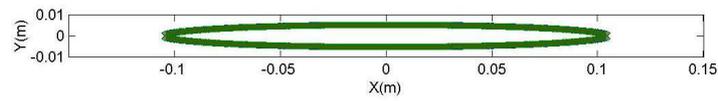


(b) Close up

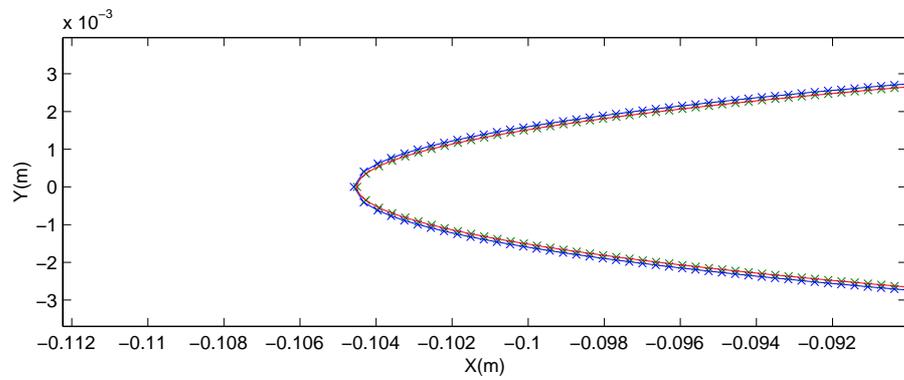


(c) Tip of the ellipse

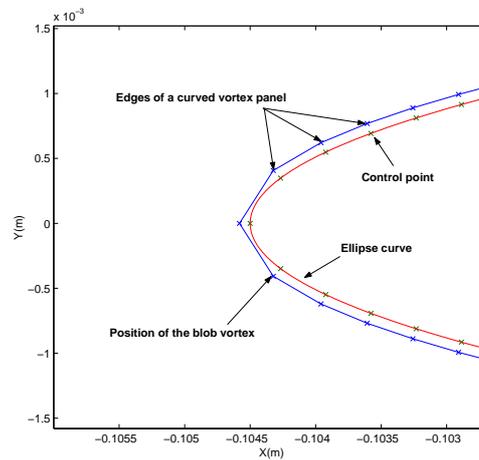
Figure 2.4: Wall approximation for straight panels



(a) General view



(b) Close up



(c) Tip of the ellipse

Figure 2.5: Wall approximation for curved panels

Unlike the preceding method, the control points distribution is uniform along the wall (equal distance between the points). Indeed, the curved vortex panel proved to be much more sensitive to the variation of the size of the panel along the wall, or, in the case of blob vortices, of the distance between vortices. There seems to be an increased leakage across the wall as the panels are stretched and concentrated on the edges of the ellipse. Furthermore, unlike with the straight panels, if one compare the flow field obtained with these methods and the one obtained with a conformal transformation (exact solution for a steady incompressible inviscid 2D flow), the best results are obtained if one places one edge of the panel – or one vortex – at the tip of the ellipse (figure 2.5).

The parameters used to test the various panel methods are listed in table 2.1.

<i>Parameters used</i>	<i>Straight source panel</i>	<i>Straight vortex panel</i>	<i>Curved vortex sheet</i>	<i>Blob vortices</i>
U_∞ (m/s)	0.72	0.72	0.72	0.72
Ellipse configuration (Notation refers to figure 2.1)	a = 0.1045 m b = a/20	a = 0.1045 m b = a/20	a = 0.1045 m b = a/2 or a = 0.1045 m b = a/20	a = 0.1045 m b = a/20
Time step Δt (s)	$\Delta t=0.04a/U_\infty$ $\Delta t=0.004a/U_\infty$ $\Delta t=0.002a/U_\infty$	$\Delta t=0.04a/U_\infty$ $\Delta t=0.004a/U_\infty$ $\Delta t=0.002a/U_\infty$	$\Delta t=0.04a/U_\infty$ $\Delta t=0.004a/U_\infty$ $\Delta t=0.002a/U_\infty$	$\Delta t=0.04a/U_\infty$ $\Delta t=0.004a/U_\infty$
Vortex core radius σ (m)	$\sigma=5 * 1.4\Delta t$	$\sigma=0.5\delta_0$ $\sigma=5 * 1.4\Delta t$ $\sigma=5 * 5 * 1.4\Delta t$	$\sigma=0.5\delta_0$ $\sigma=5 * 1.4\Delta t$ $\sigma=5 * 5 * 1.4\Delta t$	$\sigma=0.2075\delta_0$
Angle of attack α	90°	90°	90°	90°
Tested for a number of iteration N	N = 5 N = 15 N = 80 N = 300	N = 5 N = 15 N = 80 N = 300	N = 5 N = 15 N = 80 N = 300	Until N = 1200

Table 2.1: Parameters table

As a benchmark, a method was implemented based on work by Sarpkaya [50], and used in many numerical experiments by Sarpkaya (cf references [49], [52]). It uses conformal transformation, and is here slightly modified for the case of an ellipse. When the ellipse

is fixed, this algorithm is very close to the currently developed simulation and differs only in the use of complex potential instead of the blob vortices, and in the use of a flat plate. This allowed to check the basic structure of the code, to gain expertise with vortex methods, and to investigate the effect of parameters such as time step and the implementation of the Kutta condition. Nevertheless, the method becomes unstable after a given amount of time, probably due to the use of point vortices in the flow leading to a Kevin Helmholtz instability.

The blob vortex method has proven to be the best by providing a smooth velocity field, as well as a stable simulation. The source panels are inappropriate because they are unable use Kelvin's theorem. The vortex panel due to its singular nature was unable to provide a smooth vorticity shedding, and the curved panel method did not provide a velocity field appropriate for the method. As stated previously, blob vortices were used by Spalart [57] to model the boundary in the same manner as presented here.

Straight source panel For the source panel, the formulation of the straight source panel is used as described in [30]. This involves imposing the no-penetration condition on the velocity.

In this form, the method required the use of a point vortex as nascent vortices, to ensure that the velocity gradient at the edge of the plate was sufficient to ensure that the Kutta condition (cf 2.3.4) is satisfied at each time step. Then when they were freed into the flow, the point vortices were transformed into blob vortices. This was done in order to avoid the instability problems arising with the use of point vortices observed with the conformal method. Remembering that the point vortices and algebraic core vortices have a velocity distribution whose difference (in term of its norm) is less than $O(10^{-2})$ at radiuses ten times the core radius, one can note that the influence of the change on the flow field is limited to the region very close to the nascent vortices.

The results obtained were satisfying with regards to the stability, the convergence and the nascent vortices strength and positioning was smooth enough, e.g. the flow stabilizes to a time-periodic regime. The vorticity is shed continuously at the nascent vortices which indicates that the flow is smooth near the plate. However, as just mentioned, it was found necessary for the convergence of the method to use two different types of

vortices (point vortices for the nascent vortices, and blob vortices in the flow) to get a solution. If point vortices are not used as nascent vortices, the method is unable to cope with the first vortex shedding. This is apparently due to an interaction between the resolution of the boundary condition and the nascent vortices. Indeed, when the vortices are located close to the source panel, the solution of equation (2.7) leads to near singular strength for panels near the edges. Another inconvenience with this method is that circulation can only be applied on the body through the nascent vortices. This is especially crucial when there is a vortex shedding, because Kelvin's theorem (equation (2.20)) states that the global vorticity in the flow should remain constant whereas here there is no mechanism to compensate for the added vorticity. Because of this drawback, this method was not developed for moving bodies.

Straight vortex panel Here again use of a panel was made as described in [30], but with point sources replaced by point vortices and the boundary condition changed from non-penetration to the no-slip condition. As with the previous method, point vortices had to be used only as nascent vortices, with those in the wake being blob vortices.

Unlike the straight-source panel, although this method behaves properly in the first stage of the flow (development of two symmetrical vortices behind the plate), it was unable to cope with the vortex shedding and thus became unstable after a while. It seems that two mechanism are implicated here. First, the no-slip condition combined with the straight vortex panels tends to exaggerate the importance of vortices if they come close to control points. Second, inherent to the mechanism of creation of the nascent vortices (section 2.3.4.2): these are created very close to the wall, and thus bring a very high velocity contribution to the tips of the ellipse. This impedes the proper resolution of the no-slip boundary condition, and in return disturbs the solution of the Kutta condition (section 2.3.4) to place the nascent vortices as the velocity at the edges, and thus the right hand side in equations (2.39) and (2.40) tends to be singular. Furthermore, this high velocity on the edges prevents correct convection of the vortices.

It is possible to fix the position of the nascent vortices, for example by neglecting the effect of the downstream wake, but it further amplifies this phenomenon. Additionally, the no-slip condition is used here as it is not possible to enforce the non penetration

condition for the velocity with vortex panels. For use with a moving body, a more robust method is required such as the streamline formulation (equation (2.28)). However, due to the singular nature of the vortex panel, it is unclear as to how the method would have behaved. Therefore, this method did not appear suitable for the project purpose.

Curved vortex panel Clarke’s [11] implementation of the curved panel vortex was used here. One difficulty arose because of the large curvature on the edge of the ellipse, which implies that the panels on the tip are very close together. Therefore, after some trials, it was decided to implement it with a “fat” ellipse $b/a = 1/2$ instead of $b/a = 1/20$ (see figure 2.1). In this case, it was possible to use only blob vortices to avoid adversely influencing the solution.

Although the method is stable, the vortex shedding is symmetrical and reattaches to the ellipse after a short distance at the wall. The reason is that, when created and shed into the flow, the vortices have acquired a too strong vorticity, which prevents them from separating, as the velocity induced by the body is not sufficient to move them away from the wall. As they collide with the wall, the free vortices (as opposed to the boundary vortices, which are fixed) are canceled too quickly to form an eddy. This may be due to the geometry of the ellipse, as the velocity on the tip is less strong than that of an ellipse resembling a flat plate. In addition, one must not forget that shedding is made at only at two points; the location of the separation points for the non-thin ellipse will presumably vary significantly. For instance, with a cylinder the flow separates at several locations along the surface, depending on Reynolds number.

In conclusion, as it was not possible to accurately model the boundary with this method, or obtain proper behaviour with the wake, it was not used. However, this conclusion must be moderated, as it was found later that there is a sign error in the panel velocity formula provided in Clarke thesis [11]. Therefore formula 31 in section 3.6 should be read : $u^* = (-i/(2\pi\zeta)) ((s_0 - 1)\log((s_0 - 1)/s_0) - (s_1 - 1)\log((s_1 - 1)/s_1))$. This mistake could not be corrected in time and it was not possible to make trials with the modified formula, but surely this error on the basic panel velocity formula also explains why it was not possible to properly exploit this method for the ellipse geometry.

Blob vortex In the case of the thin ellipse, some difficulties arise from the fact that due to the high curvature on the edges the boundary vortices can get too close. That is why here was used a uniform repartition of the control points along the boundary. However, other grids with reasonable differences in panel size (not more than twice the size of the adjacent panels) also work properly. The non-penetration condition was implemented as the boundary condition using the streamfunction as in equation (2.30). Lastly, note that applying an offset from the wall also has the side effect of easing the resolution requirement when imposing the boundary condition.

According to Leonard (1980) [33], with the 1 : 20 ellipse one should use about 5000 boundary vortices to accurately model the boundary layer. However, it was chosen for this project to concentrate on the flow near the plate edges, and it was not given too much importance to the boundary layer representation per se, as there is no viscous model. Thus, 1200 vortices were used to model the wall, because the computational cost of using 5000 vortices was too high. It affects for example the resolution of the Kutta equation which is the part of the code that has the highest computational cost because of the nonlinear equations. Furthermore, the code is not sensitive to the boundary-layer model, thus it does not justify this high count of vortices for the entire wall.

After intensive testing, this method has proven to be the most reliable and the boundary condition (equation (2.28)) makes it the most appropriate to adapt to a moving body algorithm. In appendix A, a list of the streamline functions is provided for the algebraic core vortex, and the freestream flow. Given its suitability for the project objectives, this is the method employed to obtain the moving-plate results presented below.

2.3.2 Vortex core size

The radius of a vortex core (σ) is set to satisfy a number of conditions, as the blob vortices are used not only as a flow model but also to solve the no-penetration boundary condition. Indeed, all papers related to the convergence of the vortex method such as Hald [22], Beale and Madja [5], Leonard [33], and Chorin [10] show that as the number of vortices increases, the vortices must become closely spaced but also overlap more and more for the method to converge. In practice this means that as the number of vortices per time step increase, the average distance between them (δ_0), as well their core size,

should tend to zero, but the core radius σ not as fast as δ_0 does.

One must also consider the residual mass flux across the wall which should also be as small as possible. Spalart [57] discovered that in the case of an inviscid boundary, the core radius should be about 0.153 times the average distance between the control points for the algebraic core. In practice, this value can rarely be exactly respected because the spacing of the points is generally uneven.

Furthermore, σ should also be large enough compared to δ_0 for the core to overlap along the boundary, so that the velocity field is as smooth as possible.

Within these limits, the choice of the core size is arbitrary. It was decided, in the case where all the blob vortices had the same core radius, to find a compromise between the boundary leakage and the smoothness of the evolution of the position of the nascent vortices, which is related to the velocity field near the ellipse edges and to the resolution of the Kutta condition as explained in section 2.3.4. It was found that a value of 0.2075 times the average distance between vortices δ_0 gave good results.

One could use a time-varying core size, which has many advantages including simulating the viscous diffusion; an example is given by Rossi [48]. However, as discussed by Spalart [57] and Leonard [33], the convergence is affected and there is of course the extra computational cost. Furthermore, the effect of viscosity was not intended to be simulated here. Therefore, a constant radius core was used. However, with such a small core radius σ , smoothness can only be ensured on and near the boundary and where the nascent vortices are. Thus in most of the wake, the vortices tend to act like point vortices, despite the nonsingular core. Nevertheless, this method has shown better stability than the implementation of the Sarpkaya method [50] (use of complex potential flow solution) with the ellipse.

Trials were made using different core radius for the boundary vortices and for the vortices in the wake. The core size defined above was used for the boundary vortices, and a core size of order $U_\infty \cdot \Delta t$ (where Δt is the timestep value) for the flow vortices in order to have a sufficient number of superposed vortices. This was compared with the uniform core radius flow where the core radius is fixed at $0.2075\delta_0$ everywhere, which can be justified in that in the boundary conditions, the term from the boundary elements is

dissociated from the wake vortices term; moreover as the boundary vortices stay still, there is presumably not too much influence on the flow dynamics especially as a blob vortex velocity field is nonsingular. The boundary vortices are simply a nonsingular flow model of the wall, keeping this in mind one can dissociates the flow and boundary vortices core size while still satisfying equation (2.9).

Comparison between the uniform and non-uniform core radius vortices with the freestream flow at 45 degrees of attack, revealed that the uniform vortex core radius produced better results in terms of smoothness of the solution, in the roundness of the physical vortices and in the symmetry of the vorticity shed (i.e. at both edges of the ellipse the mean vorticity shed must be the same; see Fage and Johanssen (1927) [16]). Apparently, this is because in the non-uniform case, as for the vortex panel method, the nascent vortices come too close to the edge, but it is also related to the free vortices core size. Indeed, the radius in the case of the flow vortices was about 100 times larger than the boundary vortices, or about $a/2$, thus creating a non-negligible vorticity inside the ellipse which degrades the resolution of the boundary condition.

In conclusion, it was decided to keep a uniform radius throughout the flow in order to maintain normal convergence. This means however that with a high number of boundary vortices – as for the current simulation case, where 1200 vortices are used – the core radius is much smaller than would be required for the flow vortices. In other words, the algebraic vortex velocity field is similar here to that of discrete vortices, even though using the nonsingular core function seems to provide extra stability in the flow.

In the Spalart code, the vortex core radius is uniform everywhere because the boundary vortices are shed into the flow. He also relates the core radius σ to the average distance between vortices δ_0 using a factor of 0.25, i.e. $\sigma = \delta_0/4$.

2.3.3 Boundary vortices position

In the usual stationary plate creation scheme the vortices are created slightly off the boundary at a normal distance δN from a boundary point (section 2.3.1.2). The choice of δN is then determined by several, often conflicting, criteria.

First of all, it should be large enough compared to σ for the residual vorticity inside the body to be small and thus to limit the leakage inside the body, especially with unbounded core functions such as the algebraic vortex (cf figure 2.2(a)). But δN should also be small enough compared to δ_0 to maintain a strong coupling between the boundary vortices and the control points, to have a well-conditioned system matrix.

The effect of moving a boundary vortex away from the wall is to decrease its influence. From figure 2.2, the core radius is approximately the point of highest induced velocity. Therefore, if one do not want to underestimate the vortices influence, δN should be of order σ .

Keeping these factors in mind, the most appropriate distance seems to be that of the maximum velocity on the boundary for the algebraic core (equation 2.25), e.g. $\delta N \approx 1.12\sigma$. In the Spalart code, the distance is slightly larger at $\delta N = \delta_0/2 = 2\sigma$.

2.3.4 Discrete introduction of vorticity at separation points

This model is based on a method developed by Sarpkaya (1974) [50], and uses an approximation of the continuous vortex sheet emanating from the free shear layers at the side of the ellipse by an array of blob vortices. Again, Sarpkaya (1994) [53] provides a discussion of different schemes for discrete shedding of vortices. This is the main departure from the Spalart code. Instead of introducing only two vortices at each time step determined from the separation points, one could have instead used the boundary vortices and shed part or all of them as done by Spalart [57]. However, it was decided to limit the number of vortices to a minimum; furthermore this method enables to directly monitor the vorticity shed into the flow at each time step; and finally it was chosen to dissociate formally the boundary elements from the wake vortices in order to be able to be free to test other type of boundary elements as explained in section 2.3.1.2.

Sarpkaya [53] makes an important point that although there is a need for interaction between the newly created vortices, the boundary elements at the wall and the vortices in the wake in order to estimate correctly the forces (in particular the lift), the Strouhal number does not seem to be overly sensitive to the type of nascent vortices introduced.

2.3.4.1 Calculation of the shed vorticity

In order to determine the strength of the nascent vortices at each time step the term $\partial\Gamma/\partial t$ is used, which is the rate at which vorticity is shed from the boundary layer into the wake. It is given by:

$$\int_0^{\delta_{bl}} \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) u dy, \quad (2.33)$$

where δ_{bl} is the boundary-layer thickness, and u and v respectively the x and y axis component of the velocity vector $\vec{V} = (u; v)$.

There is more than one way to calculate the rate of shed vorticity; the options are:

- The use of a shear layer approximated by discrete vortices which enables calculating the vorticity rate using an average of the velocity over the first vortices introduced; see Sarpkaya [50]. This is the method used here.
- The use of the velocity at a suitable fixed point near the separation point, as done for instance by Clements [12]. Since the point is fixed, there is no interaction between the shed vortices and the amplitude of oscillations at the point of introduction of the vortices.
- The isolation of the shear layer from the separation point (Gerrard [21]), the selection of a control surface downstream of the separation and the computation of the vorticity flux across this control surface. In this case, the effect of the vortex sheet upstream of the control surface is neglected, and separate calculations are required to determine the arbitrary parameters such as the control surface width.
- The use of the maximum velocity in the boundary layer near the separation point (Sarpkaya [49]), however for the thin ellipse case, due to the use of boundary vortices it is far too noisy to be useful.

Equation 2.33 may be closely approximated by

$$\frac{\partial\Gamma}{\partial t} = \frac{1}{2} (V_1^2 - V_2^2) \simeq \frac{1}{2} V_1^2, \quad (2.34)$$

where V_1^2 and V_2^2 respectively represent the square of the velocity magnitude at the outer and inner edges of the shear layer. The fact that this simple expression gives a close estimate of the total vorticity flux through each sheet per unit time even for flows with rapidly curving streamlines has been demonstrated by Fage and Johansen ([16], [17]) through numerous experiments with inclined plates, cylinders, wedges and ogival

models.

Equation (2.34) can then be employed in a discrete vortex method provided that certain basic experimental facts are not contradicted, and that the numerical procedure used to implement the method is stable. Finally, the resulting vorticity rate should not be critically dependent upon the magnitude of the arbitrary parameters introduced such as the number of vortices considered. These parameters will be defined later on in this section.

The use of (2.34) involves several assumptions. Fage and Johansen ([16], [17]), through experiments with various bluff bodies, have shown that vorticity is shed from the two sides of an asymmetric body at the same rate and that the motion in a vortex sheet is steady near the body, except possibly near the inner edge of the shear layers. Also for a flat plate, a fluid flows through both edges of the vortex sheet, but at a much greater speed through the outer edge. In addition, at each section of the sheet the velocity rises to a well-marked maximum value (approximately $V_1/U_\infty = 1.45$) and then very slowly decreases to about 1.35 within a downstream distance of $y \simeq 2c$ (c represents half the chord of the plate)¹ where the breadth of the sheet² reaches a value of $\Delta \simeq c$. Finally, at the edges, they showed that the velocity V_1 at the outer edge of the sheet is much larger than the velocity V_2 at the inner edge and V_2^2 may be ignored in equation 2.34 in calculating the vorticity flux.

As stated previously, the scheme relies on an approximation of the shear layer velocity U_{sh} . To determine it, using the results from Fage and Johansen, one can reduce (2.34) to:

$$\frac{\partial \Gamma}{\partial t} = \frac{1}{2} U_{sh}^2. \quad (2.35)$$

The idea is then to approximate U_{sh} by taking the average of the transport of a number of vortices in each shear layer. Not only does this smooth out numerical errors, but also gives a better representation of the velocity and vorticity flux in the shear layers. In the case of the flat plate, Sarpkaya [50] found that taking the first four vortices was sufficient. Indeed, the number of vortices must not be too large to avoid having the last vortex taken too close to a vortex cluster, and losing the interaction between the wake

¹In the ellipse case, c is equal to a in figure 2.1. y is also described in figure 2.1.

²Past the ellipse edges, the vortex sheet is rolling up forming a nascent eddy. In the conditions described in the paragraph, the breadth of the sheet can also be seen as the width of the vortex. Here, it was chosen to refer it as vortex sheet as it better illustrates the numerical analogy used to model the flow.

and the free shear layer. Considering that the thin ellipse is fairly close in geometry to the plate the same number was chosen.

Therefore, in a shear layer initiated at one of the separation points one has

$$U_{sh} = \frac{1}{4} \sum_{k=1}^4 (u_k^2 + v_k^2)^{\frac{1}{2}}, \quad (2.36)$$

where u_k and v_k are respectively the x and y component of the velocity \vec{V}_k at the position of the k^{th} free vortices. In the code implementation, the nascent vortices created at the previous time step are the first in the array, then the higher the number the older is the vortex.

In summary, it is a very important aspect of the discrete-vortex model that the procedure above recognizes the possibility of oscillation in the rate of the shedding of the circulation, even without an oscillating wake, and permits interaction between the shear layers and the wake, and finally reduces the number of arbitrary parameters in the calculation of the vorticity to one: the number of vortices near the origin of the shear layer.

2.3.4.2 Position of the nascent vortices

To position the nascent vortices once their strength has been determined via equations 2.35 and 2.36, the Kutta condition is used, which in the case of a flat plate can be interpreted as the requirement that the flow must leave tangentially and smoothly at the edges. For the thin ellipse, the analogy is kept with the flat plate and then make the assumption that the separation points on the ellipse are placed on the extremities of the ellipse.

This is equivalent to verifying the no-penetration condition at the edges of the ellipse, that is:

$$\vec{V} \cdot \vec{n} = 0, \text{ at } (-a;0) \text{ and } (a;0), \quad (2.37)$$

with \vec{n} the normal to the wall. The unknowns in this equation are the two vortices positions. Two nonlinear equations must then be solved simultaneously, one in $(-a; 0)$ and the other at $(a; 0)$. At the moment, there are four unknowns (the two coordinates of the first nascent vortex and the two coordinates of the second nascent vortex) for two equations, and there is a need to reduce the number of unknowns.

It is then necessary to make complementary assumptions. One may assume that the positions resulting from solving (2.37) are placed along the radius from the center of the ellipse to both edges of the ellipse. Equation (2.37) is then automatically satisfied due to the velocity field of the vortex, provided that one change the value of the boundary vortices according to the no-penetration boundary condition (equation (2.28)). The number of unknown has then been reduced to two, and the Kutta condition is automatically satisfied. In return, this requires a change of equations. In inviscid flows, the no-slip condition and the no-penetration conditions are equivalent. Thus, it is possible to use a no-slip condition at the separation points, as was used by Sarpkaya and Shoaff (1979) [55]. One can then state that the separation points are also stagnation points on the boundary, which is appropriate for asymptotically steady flows.

One now has two unknowns (positions of the nascent vortices along the major axis of the ellipse, which in reference to figure 2.1 is called the x axis) in the two equations related to the no-slip conditions as stated previously:

$$\vec{V} \cdot \vec{s} = 0, \text{ at } (-a,0) \text{ and } (a,0), \quad (2.38)$$

with \vec{s} the tangent to the wall. This is equivalent to writing:

$$(\vec{V}_{0p}(\overrightarrow{(-a;0)}) + \vec{V}_{0q}(\overrightarrow{(-a;0)})) \cdot \vec{s} = (-\vec{V}_w(\overrightarrow{(-a;0)}) - \vec{V}_b(\overrightarrow{(-a;0)})) \cdot \vec{s}, \text{ at } (-a;0), \quad (2.39)$$

and

$$(\vec{V}_{0p}(\overrightarrow{(a;0)}) + \vec{V}_{0q}(\overrightarrow{(a;0)})) \cdot \vec{s} = (-\vec{V}_w(\overrightarrow{(a;0)}) - \vec{V}_b(\overrightarrow{(a;0)})) \cdot \vec{s}, \text{ at } (a;0), \quad (2.40)$$

where $\vec{V}_{0p}(\vec{r})$ and $\vec{V}_{0q}(\vec{r})$ are the velocity at \vec{r} from respectively the vortices with positive circulation and the vortices with negatives circulation, $\vec{V}_w(\vec{r})$ is the velocity at \vec{r} induced by the wake vortices, and $\vec{V}_b(\vec{r})$ is the velocity induced by the boundary elements. Therefore, all there remain to do is to solve two nonlinear equations with two unknowns once their strength has been estimated. For this, the Broyden method was used according to [64]. Additionally, when evaluating $\vec{V}_w(\vec{x})$, the strength of the boundary elements is updated at each function estimation. Albeit with high computing cost, this gives better precision.

This method can be sensitive to the initial conditions, unless a second-order vortex-strength time stepping integration such as the Adams-Bashforth scheme is used.

2.3.5 Body representation

Since the boundary is only virtual, it is possible that vortices cross the wall into the interior of the body. This is due to the movement across the boundary of vortices near the body surface (essentially caused by time-stepping errors) or boundary conditions not being satisfied exactly (fluid leaking inside the boundary).

According to Clarke [11] there is some debate over whether these vortices should be reflected back into the exterior flow or removed from the model completely. Indeed, most of the discussion focused on the vorticity destruction mechanism. Morton [43] denied that the vorticity decayed because of wall diffusion. However, Fage and Johansen [16] noted that in the case of the flat plate, the vortices – physical ones, not to be confused with blob vortices – which approach the rear of the plate will ordinarily be deleted by the action of viscosity. Therefore, this latter approach was used by removing the vortices whenever they come within the plate. Another problem rises in the application of this procedure, that is whether or not to remove the vortices during the intermediate substep which are required to have an average over the strength of the nascent vortices. In the current scheme it is of lesser importance than in methods where vortices are shed from all the wall since the number of created vortices is limited to two per full time-step. Thus, even for long-term simulations (e.g. 1000 iterations), it has been found that a maximum of two vortices (e.g. the two nascent vortices) were deleted at each time step. However, there is no limit to the maximum number of vortices to be deleted at each time step, and the deletion scheme is usually used for method with vortices shed continuously along the wall.

Indeed, appealing to the similarity of the present flow with that of Clarke [11] (both about a bluff body), his scheme was implemented, which involves deletion of vortices which are inside the body at the end of each full time step and use a special reflection of any vortices entering the body in a predictor sub-step (illustrated in figure 2.6). It is worth noting that, due to the enforcement of (2.31), any net circulation removed by deletion of vortices will be replaced on the next time step through boundary conditions. Also, the coordinates used are those of the control points as the wall boundary.

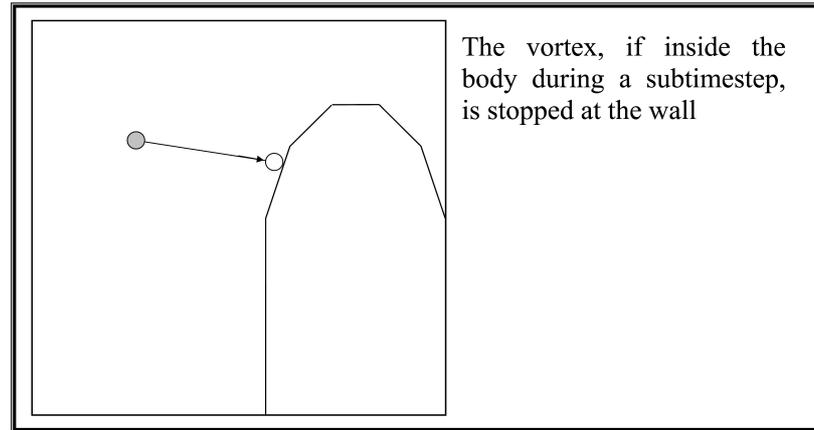


Figure 2.6: Illustration of the reflection used (courtesy of N.C.Clarke)

2.3.6 Force and moment evaluation

It is especially important to have a correct force evaluation, as this information is crucial in the flow/structure model. In the case of the vortex method, several choices are available. One can choose the Bernoulli theorem in unsteady flow (Katz (1991) [30]), a direct computation of the pressure by integration (suggested by Koutsoumakos and Cottet (2000) [13] or Shiels (1998) [56]), an alternative formula from Spalart (1988) [57] (presented on a more generalized form by Wu [65]) based on the vorticity flux across the surface, or the generalized Blasius formula from Milne and Thompson [40] considering all the vortices in the wake as point vortices in the complex plane. There is also the formula developed by Quartapelle and Napolitano (1983) [46], but it is difficult to develop for a non cylindrical body as it requires solving a Poisson equation (see Van der Vegt [62]), and even then it is difficult to exploit as it involves an approximate function.

The pressure integration method of Koutsoumakos and Cottet [13] is based on:

$$\frac{1}{\rho}\Delta P = -\nabla \cdot \left(\frac{1}{2}|\vec{u}|^2 - \vec{u} \times \omega \right). \quad (2.41)$$

It led to poor results, due partly to the use of the elliptical geometry. Another problem was that the velocity field at the wall is very noisy when the vortices are very close to the wall. Another drawback was the extra computation time, which was considerable as one is obliged to solve a Poisson equation at each time step.

Spalart 1988 [57] derived a formula applicable to arbitrary wall shape from (2.2) and (2.4). To obtain the formula first rewrite equation (2.4):

$$\frac{\partial \omega}{\partial t} = -\nabla \cdot (\vec{u} \omega - \nu \nabla \omega). \quad (2.42)$$

At the wall, one has $\vec{u} = \vec{0}$, and it reduces to the flux of vorticity normal to the wall (or the rate at which the wall is emitting or creating vorticity):

$$-\nu \frac{\partial \omega}{\partial n} \equiv \frac{\partial^2 \Gamma}{\partial s \partial t}. \quad (2.43)$$

Moreover in $2D$, it is possible to rewrite (2.2), on the body boundary:

$$\frac{\partial p}{\partial s} = -\nu \frac{\partial \omega}{\partial n}. \quad (2.44)$$

The pressure is thus dependent upon the rate of creation of circulation per unit length and unit time. For the case where new vortices are created at each time step, it is then a simple matter to determine the wall pressure in $2D$:

$$p_{j+1} - p_j = \frac{\Gamma_{j+1} - \Gamma_j}{2\Delta t}. \quad (2.45)$$

For the Sarpkaya type simulation, it is not applicable since, in contrast to Spalart simulation, the boundary vortices are not diffused at each time step. Therefore, this formula was used only for the Spalart code.

The Bernoulli formula in unsteady inviscid flow in two dimensions is

$$\frac{p_\infty - p}{\rho} = \frac{1}{2} \left[\left(\frac{\partial \Phi}{\partial x} \right)^2 + \left(\frac{\partial \Phi}{\partial y} \right)^2 \right] - \vec{U}_\infty \cdot \vec{\nabla} \Phi + \frac{\partial \Phi}{\partial t}, \quad (2.46)$$

with Φ the velocity potential in a stationary frame (\vec{X}, \vec{Y}) . In case of translation, one still has $\partial \Phi / \partial x = \partial \Phi / \partial X$ and $\partial \Phi / \partial y = \partial \Phi / \partial Y$ where (\vec{x}, \vec{y}) represents a body fixed frame. Intrinsic to the wall model (distributed blob vortices with a slight offset to the wall), the velocity field fluctuates between the wall surface and the blob vortices, rendering the direct integration of (2.46) difficult. This is illustrated in figure 3.5 in section 3.2.1.1.

From the generalized Blasius formula (cf Milne and Thompson [40]), Sarpkaya [50], [54] derived a general expression for the flat plate:

$$F_n = Y_1 + Y_2, \quad (2.47)$$

with

$$X_1 - iY_1 = -\rho \sum_{k=0}^m (\Gamma_k v_k) - i\rho \sum_{k=0}^m (\Gamma_k u_k), \quad (2.48)$$

$$X_2 + iY_2 = 4\pi\rho c^2 (\sin \alpha) \frac{\partial |U_\infty|}{\partial t} + \rho \frac{\partial}{\partial t} \sum_{k=0}^m \Gamma_k \left(\frac{c^2}{\zeta_k} + \frac{c^2}{\zeta_k^*} \right), \quad (2.49)$$

where ζ_k^* is the conjugate of the position of the k^{th} vortex in a circle plane, α is the angle of attack, U_∞ is the complex freestream velocity, Γ_k is the strength of the k^{th} vortex, ρ is the flow mass density, and u_k and v_k are respectively the \vec{x} and \vec{y} components of the velocity of the k^{th} vortex in the physical plane, and c is the radius of the circle in the image plane.

This formula has proven to be very efficient. To change from the flat plate to the ellipse requires only a simple change of conformal transformation. This has given good results for the thin ellipse compared to the flat plate. However, deriving a similar expression for the moment is more tricky, and it was thus decided to redevelop entirely the formula for the moment with the same approach (as in Lagally's theorem in Milne and Thompson 1968 [40]). For the ellipse, one then use the conformal transformation:

$$z = C \left(\zeta + \frac{\lambda}{\zeta} \right), \quad (2.50)$$

with $C = (a + b)/2$ and $\lambda = (a^2 + b^2)/(4C^2)$, which maps a unit circle into an ellipse. If one consider zero circulation, and an invariant angle of attack, the forces applied on the ellipse are then:

$$X_1 + iY_1 = -\rho \sum_{k=0}^m (\Gamma_k v_k) + i\rho \sum_{k=0}^m (\Gamma_k u_k), \quad (2.51)$$

$$X_2 + iY_2 = 2\pi\rho C^2 (\lambda e^{-i\alpha} - e^{i\alpha}) \frac{\partial |U_\infty|}{\partial t} + iC\rho \frac{\partial}{\partial t} \sum_{k=0}^m \Gamma_k \left(\frac{\lambda}{\zeta_k} + \frac{1}{\zeta_k^*} \right). \quad (2.52)$$

Concerning the moment M , one evoke the generalized Blasius formula in the complex plane (cf Milne and Thompson [40]) for an elliptical body with zero global circulation, and with an incidence α . Considering the complex potential w in the ζ plane:

$$w = -CU \frac{e^{i\alpha}}{\zeta} + \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \ln(\zeta - \zeta_k) - \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \ln \left(\zeta - \frac{1}{\zeta_k^*} \right). \quad (2.53)$$

In the real plane, it is equivalent to the potential in the stationary frame (\vec{X}, \vec{Y}) , thus removing the freestream velocity. It is actually an elegant way to simplify the

calculation, by removing some complications arising from the freestream flow potential $-zCUe^{-i\alpha}$. From this formula, one can see that there is a circulation introduced in the equation through the image vortices because they are inside the circle. The moment is then the real part of:

$$-\frac{1}{2}\rho \oint z \left(\frac{dw}{dz} \right)^2 dz - \rho U_\infty^* \oint z dw + \rho \frac{\partial}{\partial t} \oint z w dz^*. \quad (2.54)$$

All vortices in the formula are then assimilated to point vortices. The form is similar to that of the force, except for the second term of this integral.

The formula is then, for a steady freestream flow and the centroid of the ellipse at $(0, 0)$:

$$M = M_1 + M_2 + M_3, \quad (2.55)$$

$$M_1 + iN_1 = -\rho \sum_{k=0}^m \Gamma_k z_k (u_k - i.v_k), \quad (2.56)$$

$$M_2 + iN_2 = -\rho CU_\infty^* \left\{ 2i\pi CU_\infty + \sum_{k=0}^m \Gamma_k \left[\zeta'_k + \frac{\lambda}{\zeta_k} \right] \right\}, \quad (2.57)$$

$$M_3 + iN_3 = \rho \frac{\partial}{\partial t} \left\{ C^2 \sum_{k=0}^m \left[-(\lambda^2 - 1)\Gamma_k \ln(-\zeta_k) - \Gamma_k \left(\frac{\lambda \zeta_k'^2}{2} - \frac{\lambda}{2\zeta_k'^2} \right) + (\lambda^2 - 1)\Gamma_k (\ln(\zeta'_k) + \ln(a_k + \zeta'_k) - \ln(\zeta'_k)) \right] \right\}, \quad (2.58)$$

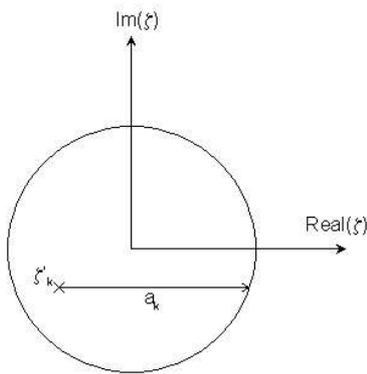


figure 2.7 : Definition of a_k

where ζ'_k represents in the ζ plane the complex value of the location of the image vortex (which is for a unit circle $\zeta'_k = 1/\zeta_k^*$), a_k represents a distance, and is defined by its representation in figure 2.7. For the full derivation, refer to appendix C. This formula is used to generate the aerodynamic moment for steady flow in chapter 3.

Following the same formulation, the forces are for a steady freestream flow and no rotation:

$$F = F_1 + F_2 + F_3 + F_4 + F_5, \quad (2.59)$$

$$F_1 = i\rho \sum_{k=0}^m \Gamma_k z_k (u_k + i.v_k), \quad (2.60)$$

$$F_2 = -i\rho C \frac{\partial}{\partial t} \left\{ 2i\pi C U_\infty + \sum_{k=0}^m \Gamma_k \left[\zeta'_k + \frac{\lambda}{\zeta_k} \right] \right\}, \quad (2.61)$$

$$F_3 = 0, \quad (2.62)$$

$$F_4 = \rho S_{body} \frac{\partial U_\infty}{\partial t}, \quad (2.63)$$

$$F_5 = -i\rho C U_\infty \sum_{k=0}^m \Gamma_k, \quad (2.64)$$

where S_{body} is the cross section surface of the body, for an ellipse with infinite span; this is $S_{body} = \pi ab$, from figure 2.1. Note that when there is rotation, F_3 is actually nonzero; details are presented in appendix C. Also note that this is equivalent to formula 2.51, except that in the potential the terms related to the freestream flow are separated. Nevertheless, it has proven difficult to use because it can lead to a very noisy force when the vortices comes close to the vicinity of the boundary vortices. This is especially noticeable for the nascent vortices.

Similarly to the Spalart method for force evaluation, Wu [65] used a similar formula but over the entire domain, thus making it simpler to use with the Sarpkaya-like simulation, as it also involves boundary vortices. The force \vec{F} is calculated as a function of the rate of change of vorticity for a finite domain :

$$\vec{F} = -\rho \frac{d}{dt} \int_{R_\infty} \vec{P} \times \vec{\omega} dr + \rho \frac{d}{dt} \int_{R_s} \vec{v}_s dr, \quad (2.65)$$

$$M = \vec{l} \times \vec{F} = \frac{1}{2} \rho \frac{d}{dt} \int_{R_\infty} |\vec{P}|^2 \vec{\omega} dr + \rho \frac{d}{dt} \int_{R_s} \vec{P} \times \vec{v}_s dr, \quad (2.66)$$

where \vec{P} is the position, $\vec{\omega} = \omega \vec{K}$ is the local vorticity with \vec{K} a unit vector orthogonal to the (\vec{x}, \vec{y}) plane, \vec{l} is a position vector describing the line of action of the aerodynamic forces \vec{F} , R_∞ is a limitless region jointly occupied by the solid body and the fluid, R_s is the region occupied by the solid body, and \vec{v}_s is the velocity of the body. A similar formula is used for the moment, which is presented in appendix D.

2.3.7 Time step choice

Another numerical parameter is the time step Δt . First, one can normalize the time by using $t^* = |\overline{U_\infty}|t/(2a)$, i.e. the normalized time step is Δt^* . After several trials, using $\Delta t^* = 0.08, 0.04$, and 0.02 , the best compromise between computation time and precision was found to be 0.04 . Indeed, there is minimal difference between the results obtained with $\Delta t^* = 0.02$ and $\Delta t^* = 0.04$. Trials with lower time step do not lead to significantly better results.

The method used by Spalart (1988) [57] was found to be very sensitive to the time step value especially with low N values. This is especially visible in the resulting aerodynamic forces. For example, a comparison was made using $\Delta t^* = 0.04, \Delta t^* = 0.02, \Delta t^* = 0.01$ at an incidence $\alpha = 90^\circ$ for the simulation from Spalart, with 1200 boundary vortices and limited the total number of vortices to 4000 to speed up the simulation (recall that in the Spalart code the number of free vortices is conceptually identical to the total number of vortices since at each time step the boundary vortices are freed into the flow). The history of the normal force coefficient C_n (i.e. the coefficient related to the force normal to the ellipse major axis) is presented for the different timesteps in table 2.2.

α (degree)	C_n (Fage and Johansen)	$\Delta t^* = 0.01$	$\Delta t^* = 0.02$	$\Delta t^* = 0.04$
90	1.84	3.60	3.66	3.14

Table 2.2: Force coefficient comparison considering the timestep for the Spalart code

In table 2.2, note that the factor 2 between the C_n obtained with the Spalart code and the one measured in the Fage and Johansen [16] experiments is purely coincidental. Spalart [57] using a flat plate model found a value of 2.8 for a similar case. The different simulation parameters used were not detailed. However, the difference of the Spalart code with the simulations done for this table can mainly be explained through the total number of vortices in Sarpkaya like simulation which is low compared to the number of boundary vortices used. The core radius used also seem to be an important parameter on the C_n . The reader is invited to refer to sections 3.2.2 and 3.2.3 for further details on the influence of the number of vortices and core radius.

To illustrate the differences of the evolution of the output in frequency and in amplitude as Δt^* varies, in figure 2.7 is plotted the moment coefficient C_m for $\alpha = 45$ using 1200 boundary vortices and a total number of 4000 vortices with three different timesteps.

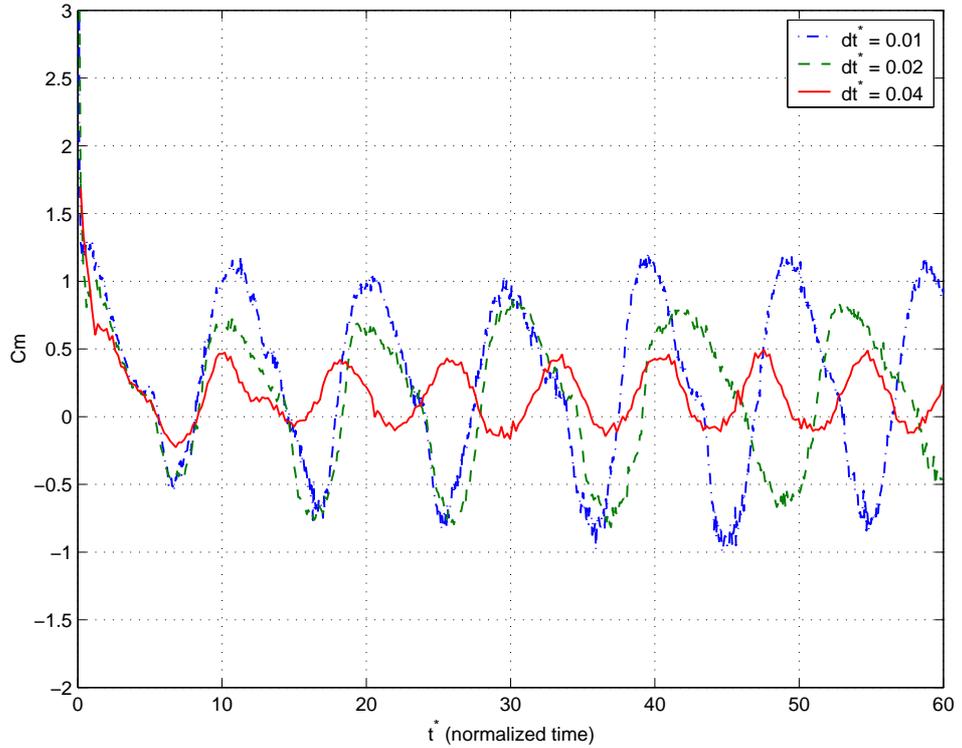


Figure 2.7: Comparison of C_m for the Spalart simulation at $\alpha = 45$, for different Δt^*

These inconsistencies in figure 2.7 are likely due to the limited number of vortices. Indeed, with the Spalart code as the total number of vortices increases, merging occurs closer to the body. This effect depends on time step because during one timestep the vortices will convect less far with a smaller time step than with a larger one; furthermore since at each time step every boundary vortex is convected, the limit number of vortices is reached more quickly. Therefore, one should increase the limit number of vortices as the time step is reduced.

In figure 2.8, the same timestep is used but with an increased number of vortices in the flow, the total number of vortices has then become $N = 6800$. One can then see that the effect of the timestep are reduced considering the C_m magnitude. Now concerning the flow periodicity, looking at the C_m peaks shows that the vortex shedding is tending to slow down as Δt^* decreases. Nevertheless the differences when Δt^* increases

are less pronounced than for $N = 4000$, although the C_m plots seems to have much more numerical noise when Δt^* decreases. Remarkably, the solution for $\Delta t^* = 0.04$ is now close to the solution in figure 2.7 for $\Delta t^* = 0.01$. Concerning the vortex shedding slowdown with the timestep decrease, it can be caused by the integration scheme used for the blob vortices convection (cf section 2.2.3) which would then introduce some numerical damping, or it could be due to the core radius σ which if too big compared to the timestep also tends some damping (see also section 3.2.3, and in particular the comments on figure 3.24). Further investigations would be required to point out the exact mechanism as it seems it is due to a conjunction of parameters.

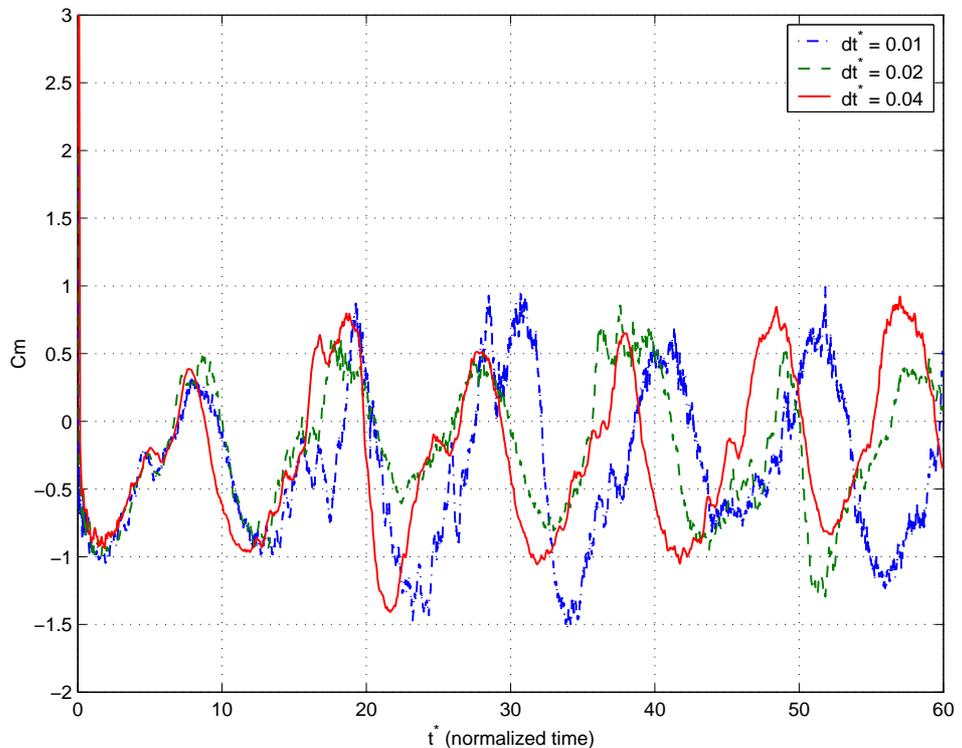


Figure 2.8: Comparison of C_m for the Spalart simulation at $\alpha = 45$, for different Δt^* for a total number of vortices $N = 6800$

This difficulty with the parameters is unsurprising as it was already noted in section 2.2.2 the difficulty of establishing the numerical parameters, and as Spalart [57] remarked similar difficulties. To solve this problem, it was required for the Spalart code to calibrate the simulation considering both the time step and the maximum number of vortices in order to have the best fit concerning the mean C_m and Strouhal number compared to the data from Fage and Johansen [16].

2.3.8 Spalart Code implementation

As stated previously, the main difference between the inviscid Spalart and the Sarpkaya-like codes was that Spalart convects all of the boundary vortices at each time step, conversely to the discrete introduction of vorticity at the edges of the ellipse. The near-wall boundary conditions in the Spalart algorithm are also enforced using streamfunctions to meet the permeability condition with the same location for blob vortices at the boundary as the initial location before the blob vortices convection into the flow (which corresponds to their creation position and is different from the control points as is stated in section 2.3.3).

All the vortices in the flow are blob vortices with an algebraic core function $\gamma(\vec{x})$. However, he uses a different value compared to the Sarpkaya like simulation for the core radius σ , because the distance to the wall δ_N is slightly larger in Spalart case.

Collision of vortices with the body is dealt by method presented in section 2.3.5, i.e. the vortices are deleted when they collide with the wall and their circulation is registered to be put into equation (2.31).

Another major difference is that his code uses vortex merging to keep the number of vortices at a user defined level N , N being the global number of vortices. The global number of vortices includes the number of boundary blob vortices, which is denoted N_b , and the number of free blob vortices which is uncontrolled.

The merging of two vortices p and q is done by ensuring minimal far-field velocity perturbation. The resulting vortex has a circulation $\Gamma' = \Gamma_p + \Gamma_q$, and has a position in the complex plane $z' = (\Gamma_p z_p + \Gamma_q z_q) / \Gamma'$. The merging is:

$$\frac{|\Gamma_p \Gamma_q|}{|\Gamma_p + \Gamma_q|} \frac{|z_p - z_q|^2}{(D_0 - d_p)^{3/2} (D_0 - d_q)^{3/2}} < V_0, \quad (2.67)$$

where d_p and d_q are the distance of respectively the vortices p and q to the closest wall. D_0 is a length parameter that allows one to obtain better resolution near the wall (small D_0) or in the wake (larger D_0). Spalart noted that the calculation showed little sensitivity to this parameter, and used a typical value of 10% of the chord. The other parameter V_0 represents the tolerance. This criterion is adjusted so that the number of vortices stays around the prescribed number N . One should note that although it keeps the

number of vortices low, the vortex is a very noisy procedure for the velocity field locally.

At this point, remark that N is a simulation parameter independent of time, determining the *maximum* global number of vortices. The actual global number of vortices is actually varying during the simulation. At $t = 0$, there are only the N_b boundary vortices required to enforce the no-penetration boundary conditions. The actual number of blob vortices increases afterwards with the release of N_b boundary vortices into the flow at each time step and then stabilizes later around N using the merging algorithm. Notably, the V_0 parameter, which is dependent on the actual number of vortices, enables the simulation to adjust the merging.

2.4 Changes for the moving ellipse

In order to reach a solution to the rotating ellipse problem, a fixed ellipse was considered with a rotating free flow. Thus, the boundary conditions have to be modified (section 2.4.1), as well as the flow geometry (section 2.4.2), and the force evaluation (section 2.4.3). Most of the complications arise from the force evaluation and the boundary condition.

Both the Sarpkaya-like simulation and the Spalart code use the body-fixed frame of reference (\vec{x}, \vec{y}) with origin O moving in the stationary or inertial frame (\vec{X}, \vec{Y}) . However, the boundary vortices are moving with the angular velocity $\dot{\theta}$. The origin O of (\vec{x}, \vec{y}) is still moving away from the origin (\vec{X}, \vec{Y}) with a translation velocity \vec{U}_∞ .

Concerning the Spalart method, only slight modification is required in the case of a rotating plate; one only has to modify the boundary condition by adding an additional term detailed in the following sections. As noted by Spalart [57] (p25) in $2D$: “if the freestream has uniform vorticity the vortex method can still be applied to the deviation from the freestream vorticity”.

Now concerning the Sarpkaya-like simulation, as the same boundary conditions are used as for the simulation from Spalart the same kind of modification is brought to the boundary conditions and the flow geometry. The main differences are the evaluation of the resulting aerodynamic force and moment as well as for the evaluation of the nascent vortices.

2.4.1 Boundary condition

As noted previously, (2.28) still holds for both simulations, i.e. the no-penetration condition is still satisfied. The only addition in the case of the rotating ellipse is just to add a term concerning the rotation. Equation 2.30 now becomes:

$$\psi_b(\vec{x}_{i+1}) - \psi_b(\vec{x}_i) = \psi_f(\vec{x}_i) - \psi_f(\vec{x}_{i+1}) + \psi_\infty(\vec{x}_i) - \psi_\infty(\vec{x}_{i+1}) + \psi_r(\vec{x}_i) - \psi_r(\vec{x}_{i+1}), \quad (2.68)$$

with $\psi_r(\vec{x})$ the streamfunction due to the rotation at the point \vec{x} . The form of ψ_r is easy to manipulate, and is :

$$\psi_r(\vec{x}) = \frac{\dot{\theta}}{2} |\vec{x} - \vec{x}_o|^2, \quad (2.69)$$

with $\dot{\theta}$ the angular velocity, and \vec{x}_o the center of rotation of the ellipse.

As suggested by Katz and Plotkin [30] for unsteady flow, and following Kelvin's theorem for the moving body, equation (2.31) was employed for both the Spalart simulation and the Sarpkaya-like simulation.

2.4.2 Flow geometry

The main change in the flow geometry concerns the motion of the boundary vortices with an angular velocity $\dot{\theta}(t)$. Their position is updated at each time step. Again, the same modifications are applied to both simulations. At the end of the timestep, each boundary vortex is moved by an angle of $\dot{\theta}(t) \Delta t$. No further modification is required either for the velocity computation or for equation (2.68).

2.4.3 Force and moment computation

As the boundary vortices are moving and as there is no modification applied to the flow velocity field, one only has to add one additional term concerning the moment to Wu's formula (Wu [65]). The added term and the complete expression for the force and moment is detailed in appendix D. Appendix C contains the complex expression derived from the Milne and Thompson [40] formula.

2.5 Algorithm overview

This section details the general algorithm used to implement the Sarpkaya-like method. In order to keep the algorithm as simple as possible, no fast summation method was used nor a vortex elimination scheme in the far field.

To start the method, the suggestion by Sarpkaya [50] to determine the strength of the vortices was followed. It is based on the observation that asymptotically the shear layer velocity tends to $1.4U_\infty$. Thus for the first four timesteps, one fix :

$$\Delta\Gamma = \frac{1}{2}U_{sh}^2\Delta t \simeq U_\infty^2\Delta t. \quad (2.70)$$

In addition, to ensure that the velocity is updated frequently enough, one solve the boundary condition three times per substep. That is to say, first before the resolution of the Kutta equation with an initial guess on the nascent vortices positions, and then iterating to ensure that the resolution of the Kutta condition has not been adversely affected by the wall solution. The boundary conditions are imposed after the Kutta condition is set, and then again after convection of the vortices.

The algorithm is summarized in figure 2.9. Such details as the vortex deletion from the diagram were excluded for the sake of clarity.

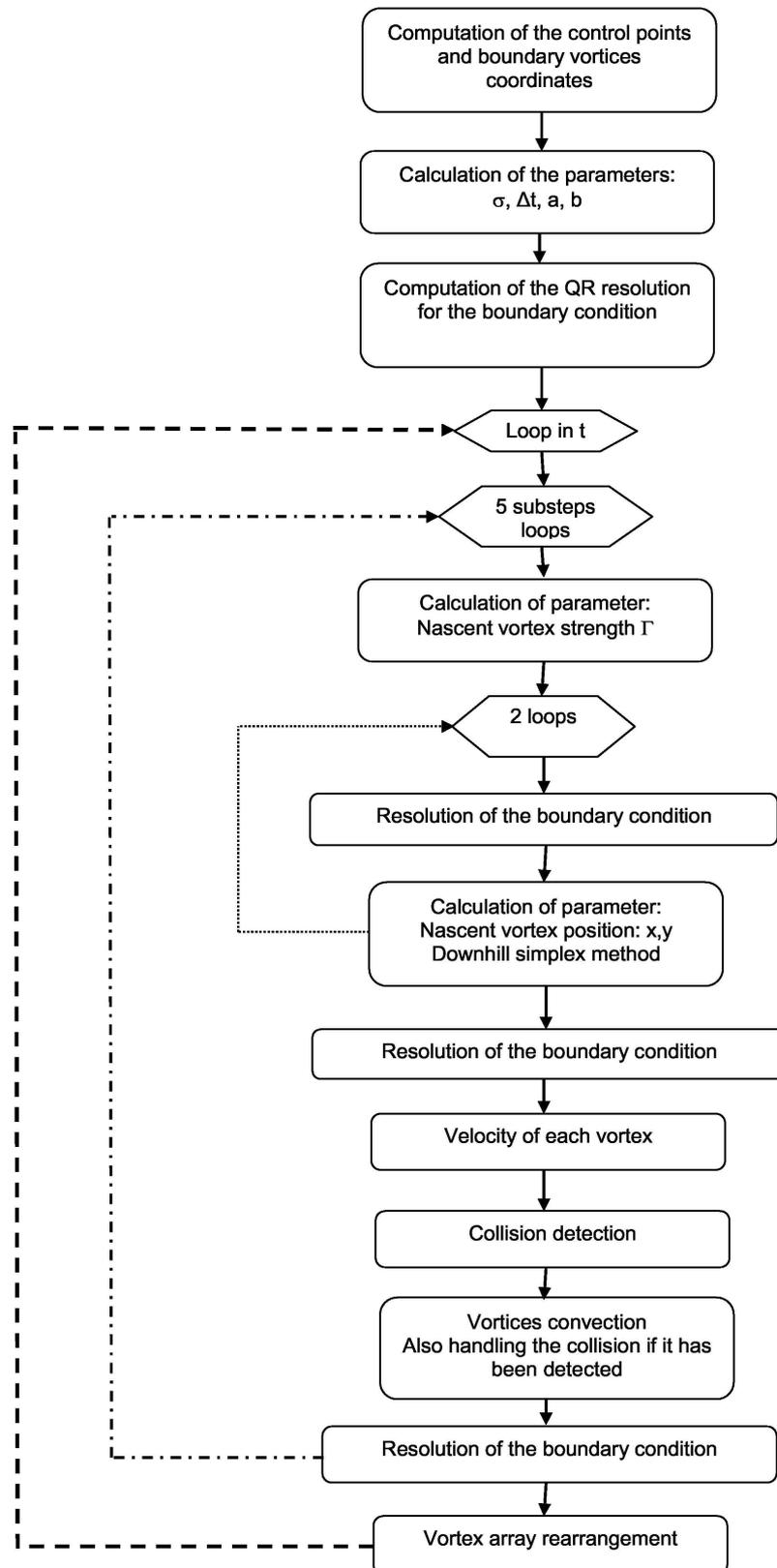


Figure 2.9: General algorithm

Chapter 3

Vortex/plate simulations

3.1 Outline

So far, the blob vortex method has been described in chapter 2 –more specifically in section 2.2– using an implementation based on the discrete introduction of vorticity (section 2.3.4) and a pure blob vortex method developed by Spalart (section 2.3.8). Although the implementation has been mainly oriented toward the fixed body problem in section 2.3, development for a moving body has also been shown in section 2.4.

Now in this chapter, the blob vortex simulation will be studied by first trying to validate the simulation (section 3.2) by comparison with a potential solution for the steady flow case in section 3.2.1 for a translating plate case and a rotating plate case. Then in section 3.2.3, the case of a fixed plate in a uniformly translating crossflow will be studied by comparing the simulations results (Sarpkaya-like and Spalart implementation) with experiments from Fage and Johanssen [16] and Honji [27] in section 3.2.4.1. Finally, in section 3.2.4, comparison will be made with other simulations and experimental results for a vertical oscillation of a cylinder (section 3.2.4.2) and a rotating ellipse (section 3.2.4.1).

Observe that for this kind of method, a proof of the convergence is difficult to establish for a given set of parameters, and such proof have not been found in the literature cited in the current work. Nevertheless, note that even for more general vortex methods such as the Spalart code, it is not thought possible to prove convergence for a given set of parameters (see Spalart citation in section 2.2.2) despite the existence of demonstration of the convergence of the method (see Koutsoumakos [13] and section 2.2.2). Still note that Koutsoumakos [13] clearly stated that even for undersolved system the method

is expected to give good qualitative results. This should be considered regarding the sections related to the numerical parameters (notably section 2.3) and the flow results presentation (chapter 3 and section 5.4).

In the second part of this chapter in section 3.3, the case of the spring damped plate in a uniformly translating crossflow will be studied through different cases after defining some study parameters. The case of a plate allowed to oscillate in rotation with no freeflow will be exposed in section 3.3.1 in order to assess the simulation and flow influence. Finally, the plate allowed to oscillate in rotation in a uniform translating crossflow will be studied by first considering the influence of the reduced damping (section 3.3.2.1), and then the reduced resonance frequency (section 3.3.2.2).

3.2 Validation of the codes

3.2.1 Blob vortex solution on the boundary

As mentioned in section 2.3.1.3, the modeling of the boundary by blob vortices is only an approximation of the boundary conditions. Therefore one must assess the accuracy of the boundary blob vortices used by the discrete vortex method of both the code from Spalart and the Sarpkaya-like code (or C code as it is sometimes written at least until section 3.2.4). As the body moves in translation and in rotation, two inviscid steady cases will be investigated: uniform translation (section 3.2.1.1) and uniform rotation (section 3.2.1.2). Comparison will be made with the exact potential solution. The leakage on the body surface will also be assessed as well as the smoothness of the solution.

The test uses 1200 boundary vortices with a uniform distribution (i.e. each panel is the same size). Indeed, a distribution where the vortices are concentrated at the edge of the ellipse produces worse results than the uniform distribution considering the velocity field smoothness near the boundary. In the former case, the vortices strength is much too strong at the edge of the ellipse and produces a solution equivalent to a vortex dipole as a model of the ellipse. The error (in velocity direction and magnitude) and leakage occur mainly at the edges of the ellipse. It originates mainly from the straining of the streamlines at the edges of ellipse, as well as the slight distance of the vortices from the wall (as illustrated in figure 3.5) which is related to some of the method features.

3.2.1.1 Stationary ellipse in steady crossflow

Two different cases were chosen: a symmetric flow with an incidence $\alpha = 90^\circ$, and an asymmetrical flow with $\alpha = 45^\circ$. For each incidence, different plots were used: the streamlines (figure 3.1 and figure 3.2 for comparison with the potential solution), the error compared to the exact potential solution (figure 3.3), and the leakage on the boundary (figures 3.4).

The potential solution for uniform flow past an ellipse is given by Milne-Thompson [40] through the complex potential:

$$w = -C \left(\zeta U_\infty + \frac{U_\infty^*}{\zeta} \right), \quad (3.1)$$

where U_∞ is a complex number corresponding in the complex plane to \vec{U}_∞ and ζ is the image of z through the Joukowski transformation

$$z = C \left(\zeta + \frac{\lambda}{\zeta} \right), \quad (3.2)$$

with $C = (a + b)/2$, $\lambda = (a^2 - b^2)/(4C^2)$ and $z = x + iy$.

To compare the solution generated by the boundary vortices to the potential solution, the velocity field was calculated at points on a uniform 200×200 cartesian grid. The velocity generated from boundary vortices is named $\vec{V}_b = (u_b, v_b)$ and the potential flow solution velocity $\vec{V}_p = (u_p, v_p)$. The error was then quantified at a point (x, y) as $\varepsilon(x, y) = |\vec{V}_b - \vec{V}_p| / |\vec{U}_\infty|$.

The streamline plots (figure 3.1) show that on a large scale one obtains a solution broadly similar to the inviscid steady solution (figure 3.2). However a closer examination of the streamlines in figure 3.5 shows the effects of the slight offset of the boundary vortices (section 2.3.3); indeed close to the body the streamlines cross the boundary between the control points with a wavy pattern.

From the velocity error plot in figure 3.3, one can see that the error is largest near the tips of the ellipse where there are high curvature regions and a decreasing error away from the ellipse. One of the sources of error lies in the spacing of the vortices, which is large regarding to the curvature. It is especially pronounced at the tips of the ellipse where much of the vorticity is concentrated. For instance for the steady flow at

$\alpha = 90^\circ$, 100 boundary vortices at one tip represent about 27% of the total strength of all vortices. Therefore, in the vicinity of these vortices, the error in the vortices strength solution is magnified. Nevertheless, the overlap ensures that the velocity remains smooth and prevents this effect from dominating the solution.

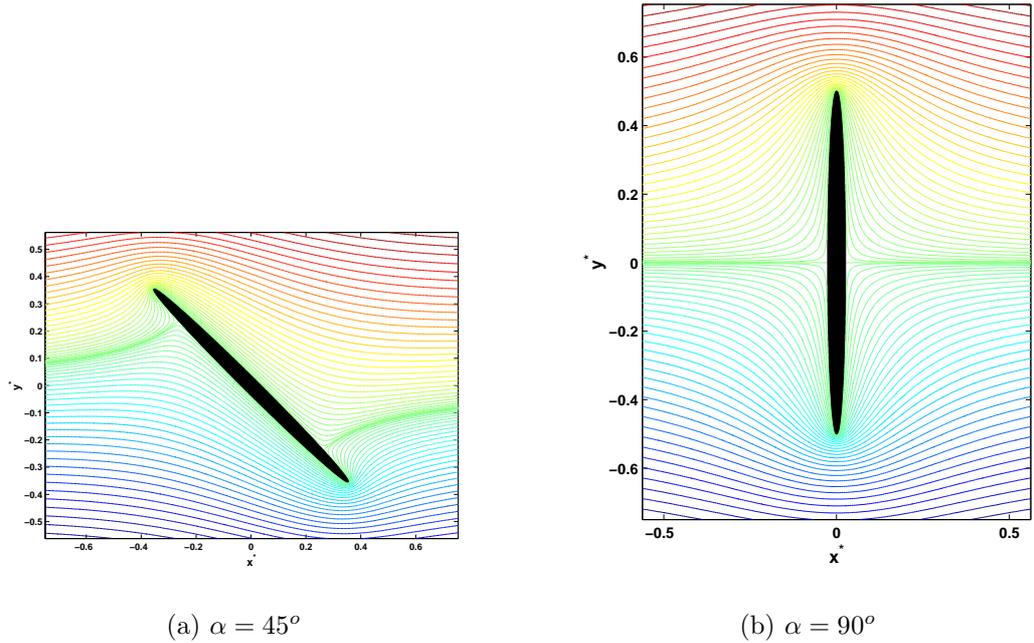


Figure 3.1: Streamline of a flow past a 20:1 ellipse for the blob vortex method

Additionally, there are some effects inherent to the boundary resolution method. As stated earlier, the boundary vortices are slightly off the wall, and though the streamlines are continuous, they are wavy. Then, as here the points considered are between the wall and the boundary vortices and to a lesser extent up to a distance of about $0.12 \delta_0$ (δ_0 is the distance between the control points) outside the boundary defined by the boundary vortices, large discrepancies in direction and magnitude appear between the two solutions if one look closely at the velocity field near an edge, as in figure 3.6.

Looking at figure 3.3, the error is concentrated at both tips of the ellipse. The error field (if assimilated as an added velocity to the exact solution) looks similar to two point vortices placed at the edges of the ellipse. The vorticity of one of those point vortices at one edge would be of sign opposite to the vorticity of the point vortex at the other edge. It could thus be possible to roughly model the error field as two virtual

point vortices of opposite vorticity at both edges of the ellipse.

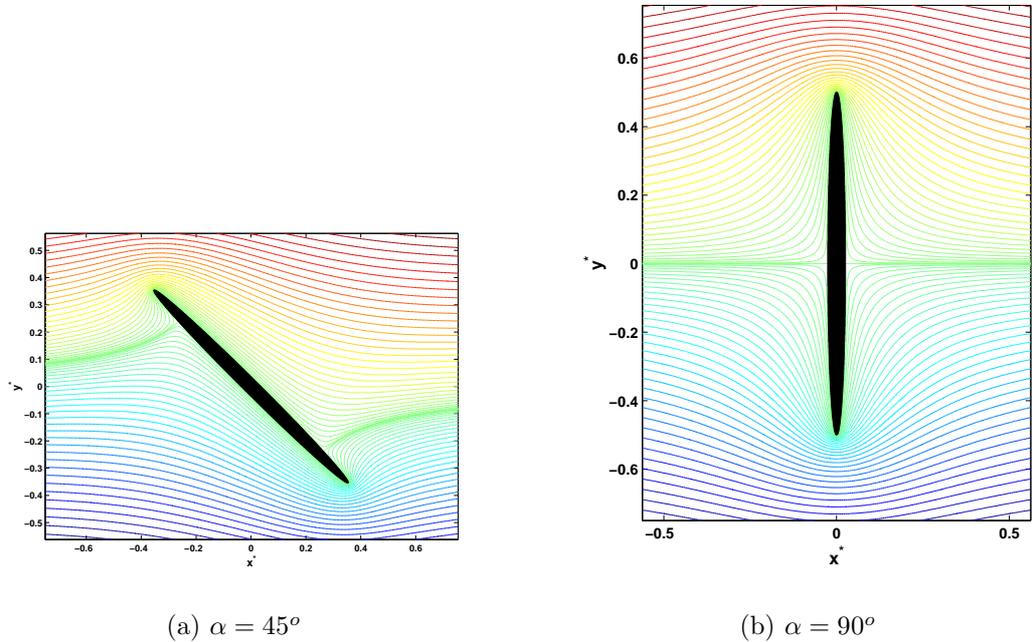


Figure 3.2: Streamline of a flow past a 20:1 ellipse for the potential flow solution

Observing the velocity field off the boundary defined by the boundary vortices in figure 3.6, one can see that the velocity is in the same direction but with a larger magnitude than the corresponding potential solution velocity. It can be deduced that the vorticity of these virtual vortices behaves like an added vorticity to the potential solution. This indicates an overestimation of the strength of the blob vortices placed at the ellipse tips.

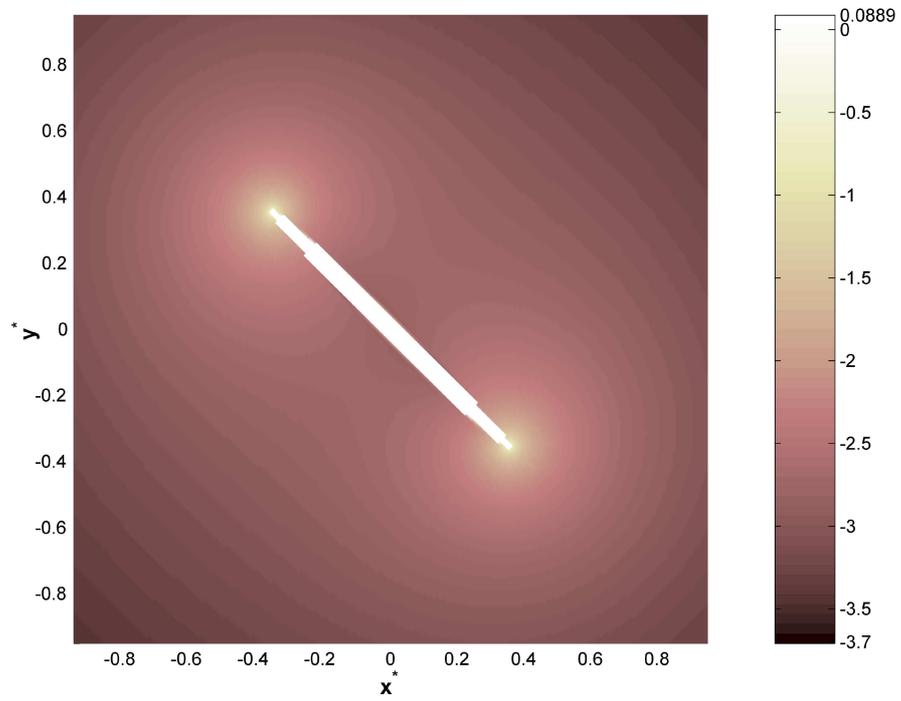
The normal velocity in figure 3.4 is characterized as $(\vec{V}_b \cdot \vec{n}) / |\vec{U}_\infty|$ with \vec{n} the normal unit vector. Again due to the distance of the vortices from the wall, the velocity is not taken directly on the wall surface, but at the boundary defined by the vortices. With the current parameters (20:1 ellipse, 1200 boundary blob vortices) the distance between the control points and the wall is very small, approximately $\delta N = 8.37 \cdot 10^{-4}$ or $\delta N \simeq b/30 \simeq a/600$, where b is half the maximum thickness of the ellipse and a half the length of the ellipse. Note also that figure 3.5 shows that the velocity field remains continuous in this transition region. It then indicates that the two surfaces (physical and the one defined by the boundary vortices) may be considered equivalent.

If one examines the “leakage” (nonzero local normal velocity) in figure 3.4, one can see some rapid fluctuation along the boundary. These fluctuations in normal velocity magnitude observed in figure 3.4 originate from the wavy pattern of the streamlines near the wall (figure 3.5). Additionally, considering the velocity around $\theta/\pi = 1$ in figure 3.4(b), one can see a smooth evolution of the local leakage. The leakage is in fact not fluctuating but smooth along the boundary. The fluctuations are caused by the wavy pattern, as well as the scale chosen for the abscissa. Indeed, with the boundary model used, more boundary vortices are used at the edges of the ellipse. Not unexpectedly and following the previous observations on the velocity error plot, this leakage is more important in the region of high curvature, i.e. at the tips of the ellipse (figures 3.3 and 3.4). The maximum leakage is then equal to about $O(10^{-3.5}) \times \left| \overrightarrow{U_\infty} \right| b$.

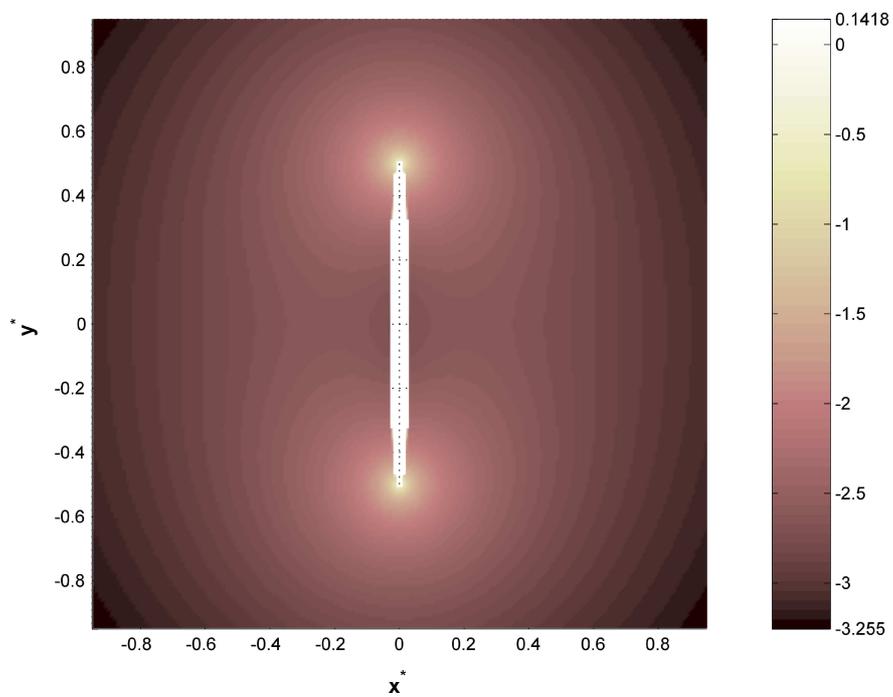
The error can be reduced by putting more vortices in the region of interest. However as a rule of thumb the ratio between the smallest and largest distance separating vortices should not be greater than two. Otherwise, the matrices resulting from equation 2.30 become ill-conditioned and the solution is unstable. Thus the main parameter one can adjust to improve the velocity accuracy near the wall is preferably the number of boundary vortices.

Interestingly in the case of an asymmetric body, such as with the plate with 45° incidence, the global leakage is a bit more important than at 90° . If one integrates the normal velocity around the surface, the total leakage at 45° appears to be three times that of 90° . However, the value of this integral remains very small, of order $O(10^{-10})$ or $O(10^{-8}) \times \left| \overrightarrow{U_\infty} \right| b$ which is close to the machine precision. Therefore, the zero total leakage boundary condition remains correctly verified in both cases with the blob vortex boundary model.

The blob vortex method has shown to provide a reasonable inviscid non-lifting solution for the range of positions of which the ellipse will be allowed to rotate. Nevertheless, there is still a residual error in velocity near the wall, which for the current geometry is largest near the tips of the ellipse. Despite this error due in part to a wavy local velocity pattern, the local leakage remains at an acceptable level. This cannot be changed unless one uses another kind of boundary elements such as a constant vortex sheet.

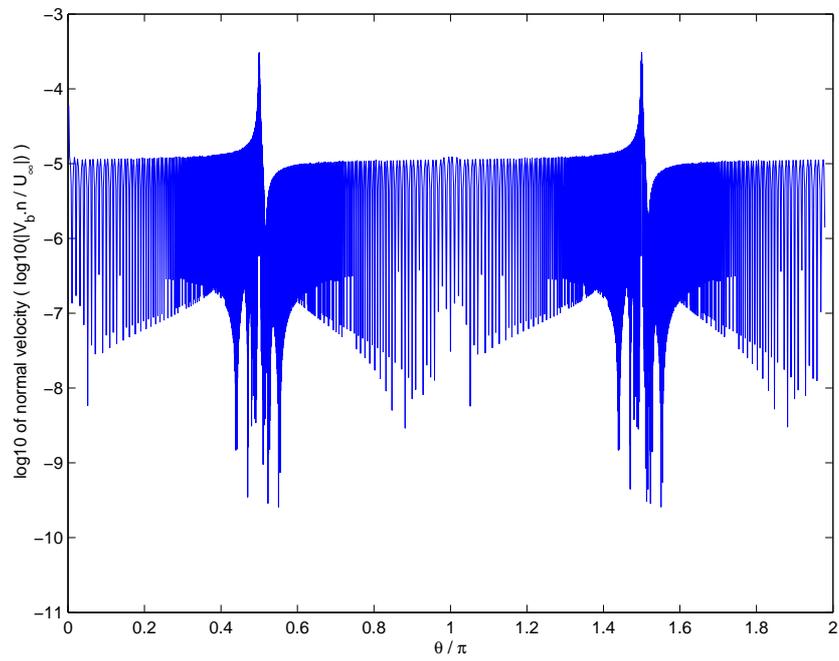


(a) $\alpha = 45^\circ$

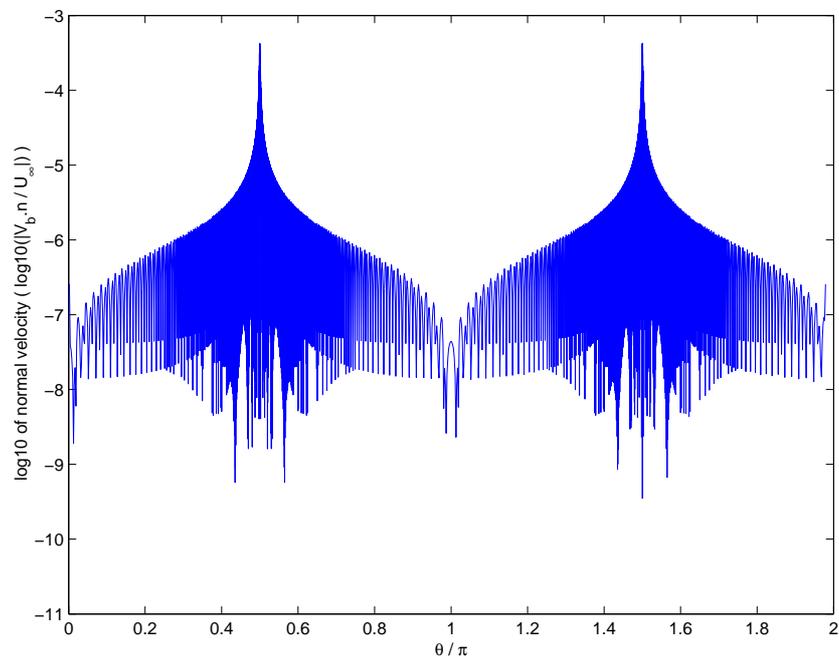


(b) $\alpha = 90^\circ$

Figure 3.3: $\text{Log}_{10}(\varepsilon)$ distribution in the grid for the 20:1 ellipse and uniform translation



(a) $\alpha = 45^\circ$



(b) $\alpha = 90^\circ$

Figure 3.4: Logarithm of the normal velocity at the boundary for two incidence angles α

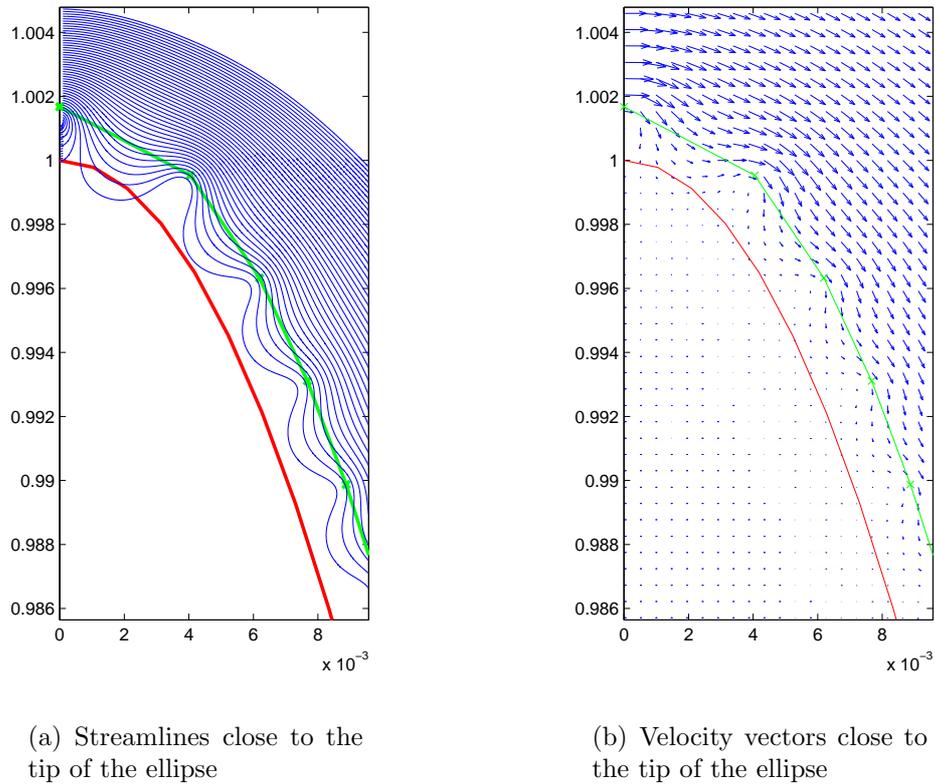


Figure 3.5: Velocity and streamline between the boundary vortices and the wall for 90° incidence

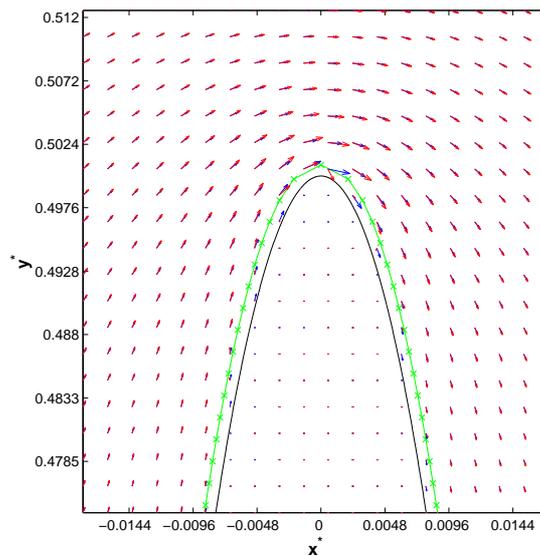


Figure 3.6: Comparison of the velocity induced by the blob vortices (red vector) and the velocity from the exact potential solution (blue line). The wall is marked by a black line, and the boundary vortices are marked by green crosses.

3.2.1.2 Stationary ellipse steady flow field rotation

In this subsection, the case study will be the steady plate rotation with no crossflow. As for the steady crossflow case, a comparison will be made between the vortex blob and potential solution (figures 3.7, 3.8 and 3.9), and also a normal velocity plot on the surface so as to assess the leakage (figure 3.10). The frame used for the plot is fixed to the plate, that is to say the flow is rotating around the plate at a constant angular velocity of Ω .

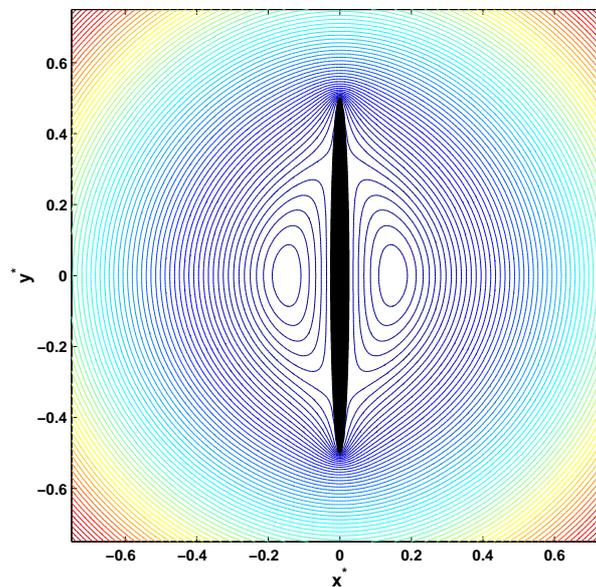


Figure 3.7: Streamline of a flow past a 20:1 ellipse for the blob vortex method

Again the potential solution is provided by Milne and Thompson ([40]). Using the convention from section 3.2.1.1, and the transformation defined by equation 3.2, one obtain for the complex velocity:

$$\frac{dw}{dz} = \frac{2B}{C\zeta(\lambda - \zeta^2)}, \quad (3.3)$$

with B defined as $B = (i/4)\Omega(a^2 - b^2)$, while C and λ are defined by the transformation in equation 3.2.

From the streamline plots in figure 3.7, the solution is again broadly similar to the inviscid steady solution (see figure 3.8). There still remains some leakage through the boundary as is shown in the plot of normal velocity in figure 3.10. Again, it is more important in the region of high curvature. Note however that over the whole boundary the leakage is ten times less pronounced than in the steady inviscid crossflow case. Some irregularities are visible on the streamlines near the body wall but this a problem related to the streamline extraction and does not involve the simulation characteristics.

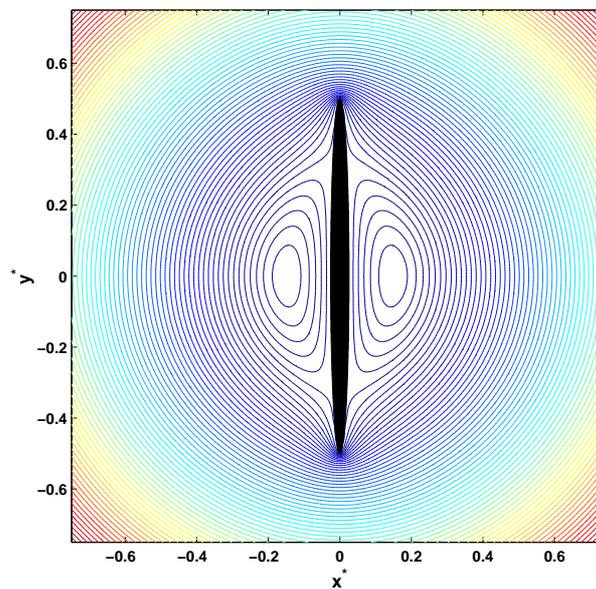


Figure 3.8: Streamline of a flow past a 20:1 ellipse for the potential flow solution

Like in section 3.2.1.1, to compare solution, a uniform 200×200 cartesian grid was used. The velocity magnitude error $\varepsilon(x, y)$ is defined in the case of the steady rotation as $\varepsilon(x, y) = |\vec{V}_b - \vec{V}_p|/(\Omega a)$. The normal velocity in figure 3.10 is computed in the same manner as in section 3.2.1.1.

As for the non rotating case, errors appear concentrated in the high curvature region of the ellipse if one compare figure 3.9 and the steady crossflow case in figure 3.3. The similar error pattern simply illustrates that the same source of error applies, e.g. the spacing of the vortices which is large regarding the curvature and the wavy nature of the streamlines near the wall.

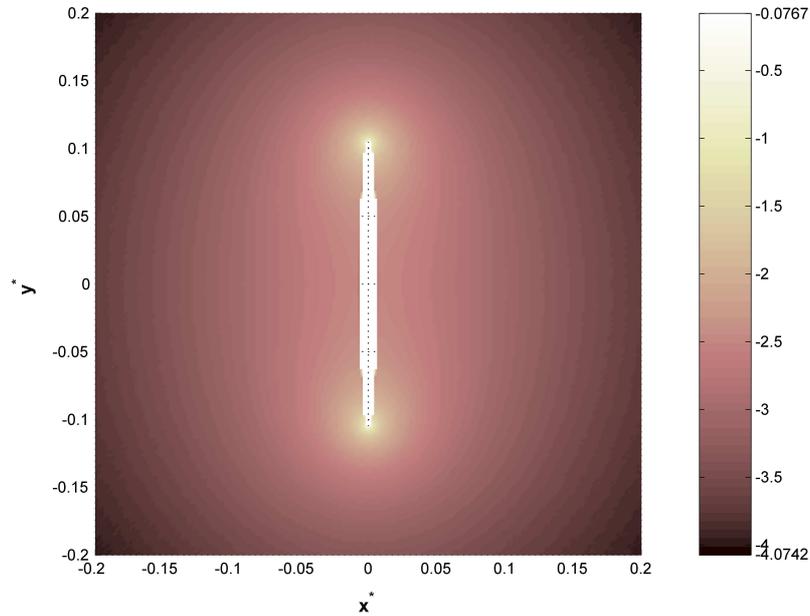
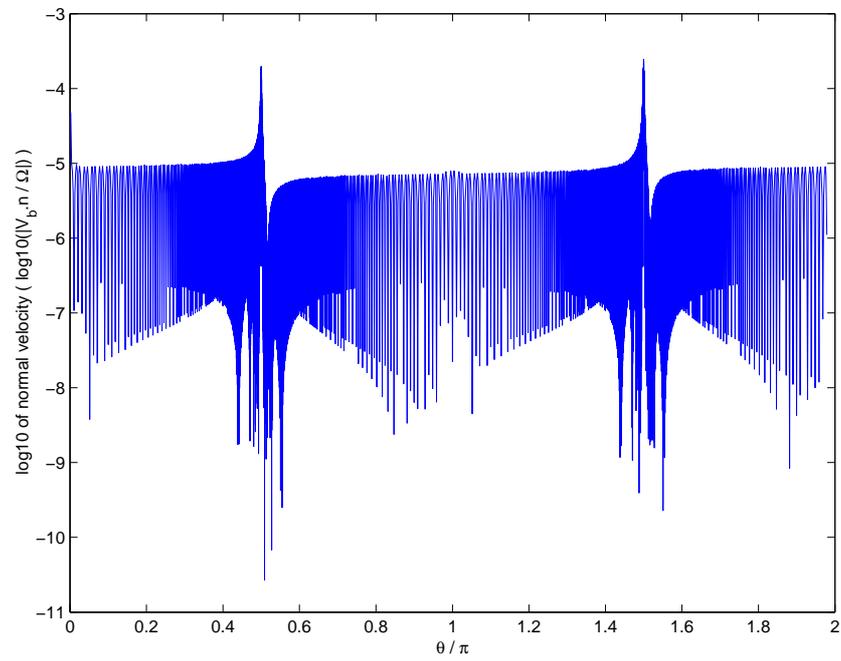
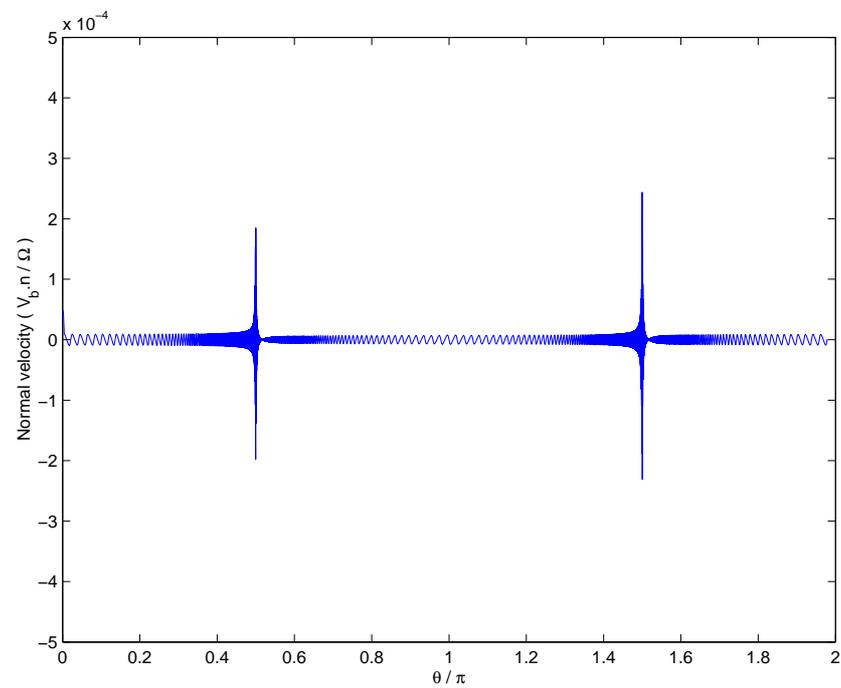


Figure 3.9: $\text{Log}_{10}(\varepsilon)$ distribution in the grid for the 20 : 1 ellipse and uniform rotation

The solution for the steadily rotating flow has proven to give a good broad approximation for the velocity field, similarly to the steady translating crossflow. Thus, one may be confident that the rotating ellipse can effectively be smoothly implemented using blob vortices as boundary elements. Thanks to the additivity property from the linear system (equation 2.30), the combination of the two kinds of flow does not pose any problem. The existing error at the tips of the ellipse is a result of the boundary vortex distribution off the wall surface and can be reduced for example by using a higher number of boundary vortices, although one must be careful not to have a too high computer cost.



(a) Logarithm of the normal velocity magnitude



(b) Normal velocity magnitude

Figure 3.10: Normal velocity at the boundary

3.2.2 Influence of the global number of vortices

The global number of vortices N is of primary importance in the Spalart code as it determines the spatial resolution of the method. The global number of vortices includes the boundary blob vortices (the number of boundary vortices is denoted N_b) and the free blob vortices. As stated in section 2.3.8, N is a simulation parameter independent of time, determining the *maximum* global number of vortices. The actual global number of vortices is actually varying during the simulation. At $t = 0$, there are only the N_b boundary vortices required to enforce the no-penetration boundary conditions. The actual number of blob vortices increases afterwards with the release of N_b boundary vortices into the flow at each time step and then stabilizes later around N using the merging algorithm. Remark that in figure 3.11 (whose scale is given in figure 3.12), N is given as individual figure title, while the actual number of blob vortices is written above the window. The “number of vortices in the window” corresponds to the actual number of vortices within the plot limits.

Thus specific to the Spalart code, the influence of N in the flow was studied for a fixed plate in a parallel crossflow. Parameters were then fixed: the number of boundary vortices, the time step, and D_0 , whereas N and α values were changed to be sure that the behaviour remains the same as the incidence angle evolves. A standard set of variables was chosen for this simulation, which is also used in section 3.2.3, $\Delta t^* = 0.05$, $N_b = 1200$, $D_0 = 4a$ and 2048 time steps for $\alpha = [45, 50, 60, 70, 80, 90]$ degrees, and $N = [1200, 1800, 2400, 4800, 6000, 8400]$.

As indicators for the comparison, the time averaged normal force coefficient and tangential force coefficient were used as defined in equation 3.4, because they represent an integration over the most basic flow variables. They are however not sufficient, as the integration leave space for “fortuitous” results deriving from a cancellation of errors. A spectral analysis of the force coefficient history was thus also used, and the Strouhal number $S = f U_\infty / (2a)$, with f the frequency of the vortex shedding. In the case of a bluff body with a vortex method, the Strouhal is generally taken as the peak in the frequency spectra of the aerodynamic forces coefficient variation in time. Performing a Fast Fourier Transform (FFT) of the force history then yields not only spectral frequency characteristics but also the Strouhal number.

Regarding the non-dimensionalization, the ellipse chord $2a$ was used for both codes. Henceforth, for this section and unless specified otherwise, this convention will be used throughout this chapter. The force and moment coefficients are defined as

$$C_n = \frac{\text{Force normal to the ellipse major axis}}{\frac{1}{2} \rho 2a U_\infty^2}, \quad (3.4)$$

$$C_t = \frac{\text{Force tangential to the ellipse major axis}}{\frac{1}{2} \rho 2a U_\infty^2}, \quad (3.5)$$

$$C_m = \frac{\text{Moment}}{\frac{1}{2} \rho a^2 U_\infty^2}, \quad (3.6)$$

where C_n is the normal force coefficient, C_t the tangential force coefficient and C_m the moment coefficient.

In figures 3.13 and 3.14 are plotted C_n and C_t respectively, both averaged over time, while in figure 3.15 the Strouhal number S was plotted. In figure 3.15, note that the Strouhal number cannot be plotted for $N = [1200, 1800]$ as the flow becomes erratic (figure 3.11) and vortex shedding no longer takes place. To compare the blob vortices distribution, the blob vortices were plotted at $\alpha = 50^\circ$ and $t^* = 51.5$ for different N in figure 3.11. Although it does not give as much insight as the streamline plots, it does provide some underlying information such as the vortex positions and thus can help to assess the degree of clustering of the vortices. To better differentiate the different regions of vorticity, the vortices are plotted according to their sign. For example, in figure 3.11(f), a regular vortex shedding and the roll-up of sheets of vorticity can be visualized.

From the vortices plot in figure 3.11, it is clear that the simulation quickly converges to a vortex shedding regime as N is raised. Remarkably, when $N \geq 2 N_b$ or $N \geq 2400$ here, the flow structure is on the whole unchanged, which is remarkable considering the lack of vortex core overlap. Indeed as one lower N , the vortex core radius remains the same – it depends only on N_b . Therefore, with low N value, one also lower the spatial resolution as the vortices keep a low core radius, which affects greatly the local velocity field.

A look at the force coefficients and Strouhal number plots in figures 3.13, 3.14, and 3.15 confirms that the overall flow dynamic is relatively insensitive to the global number of vortices for $N \geq 2 N_b$. Furthermore, the various angles used and the similarities in the results also show that the incidence angle α is not as important as choosing a proper N .

A vortex shedding solution has been shown to exist and the basic flow dynamics properly captured for N past $N = 2 N_b$ for this set of parameters σ , N_b and Δt^* . Below this level, the main cause in the loss of accuracy is presumed to be the blob vortices merging which is designed to least affect, in term of velocity, asymptotically distant points from the merging vortices. It is an otherwise noisy process for the local velocity and vorticity fields, but it remains indispensable if one wishes to keep the computing cost under control. It was chosen to not expend much effort investigating the problem as it was not central to the work, and as the broad flow dynamics is of more interest for the project.

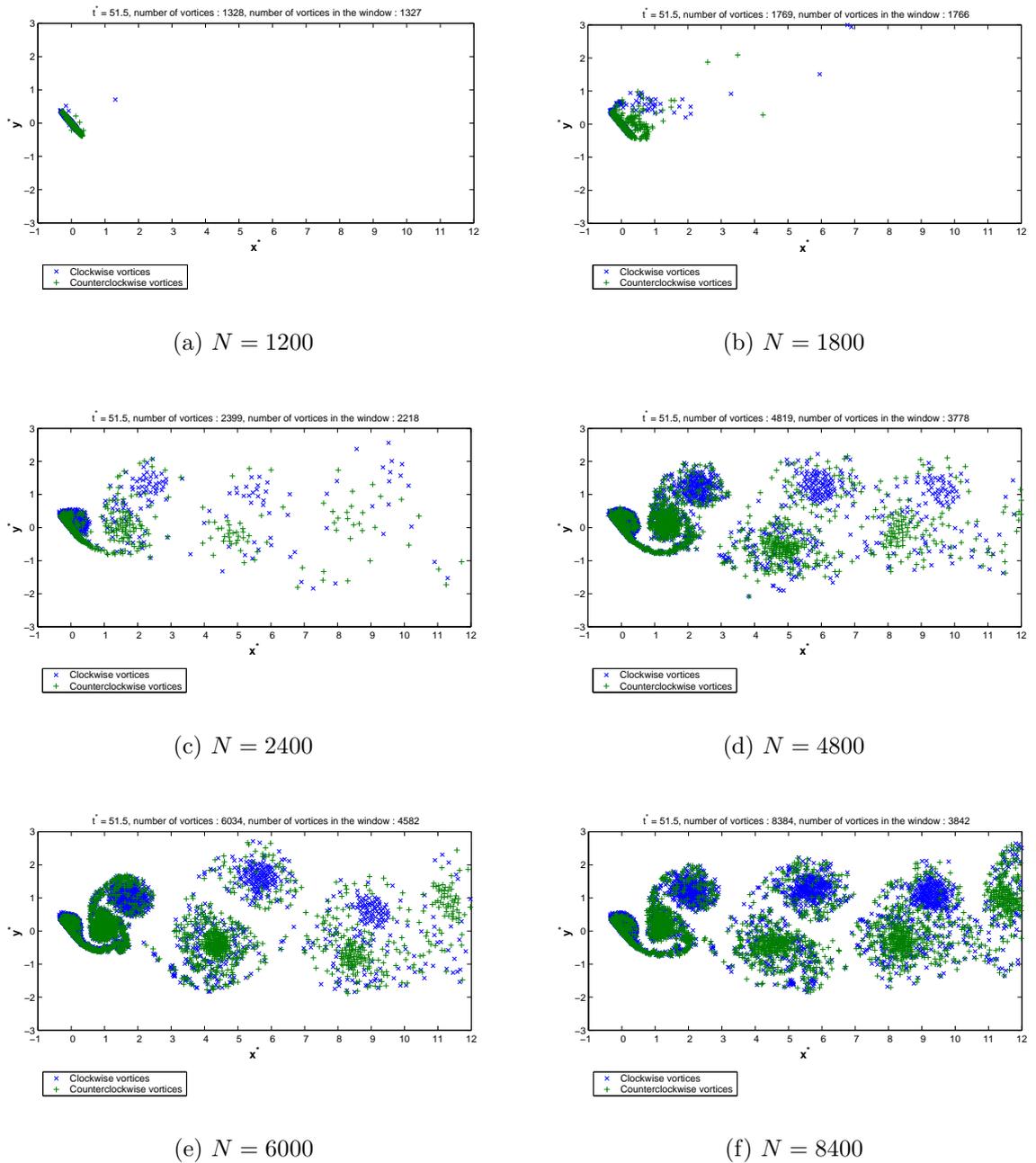


Figure 3.11: Blob vortices plot for $t^* = 51.5$ and $\alpha = 50$ with various global number of vortices N using the Spalart code. See figure 3.12 for the scale.

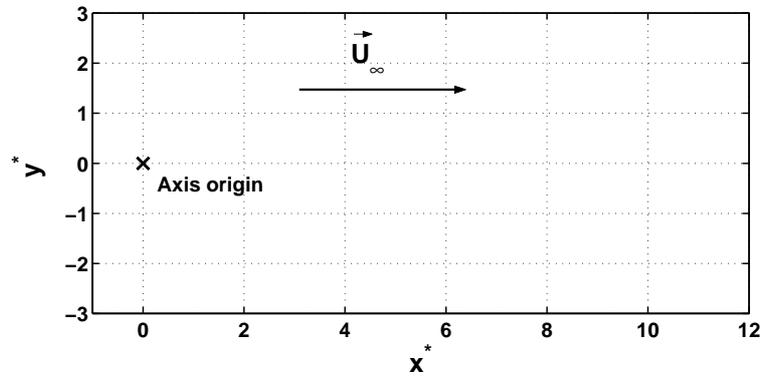


Figure 3.12: Scale for figure 3.11.

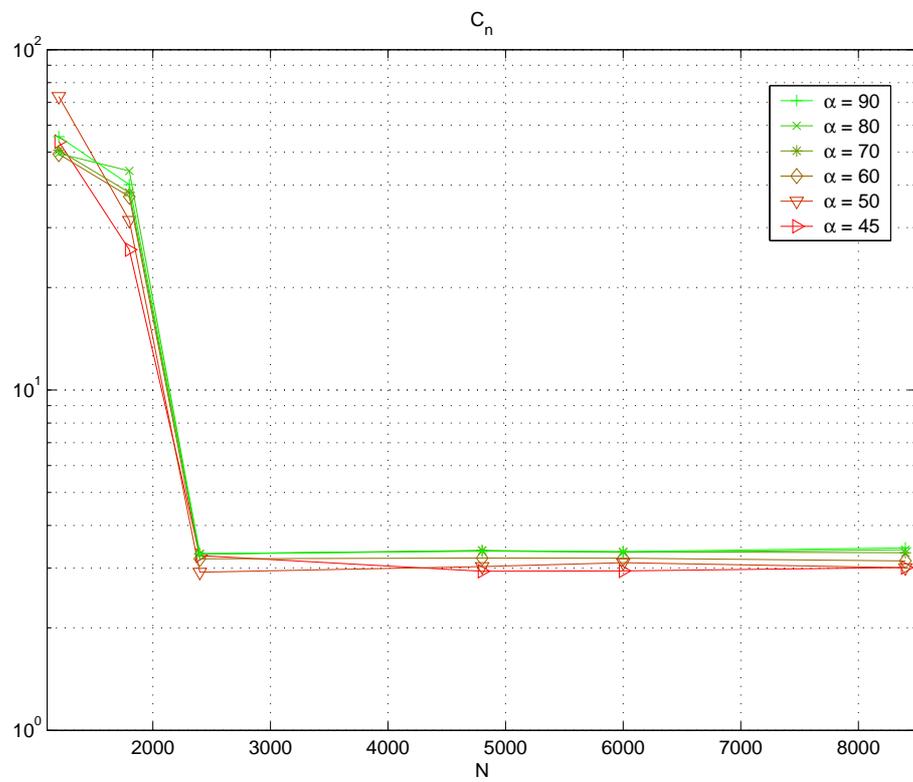


Figure 3.13: C_n for various angle of incidence and global number of vortices N

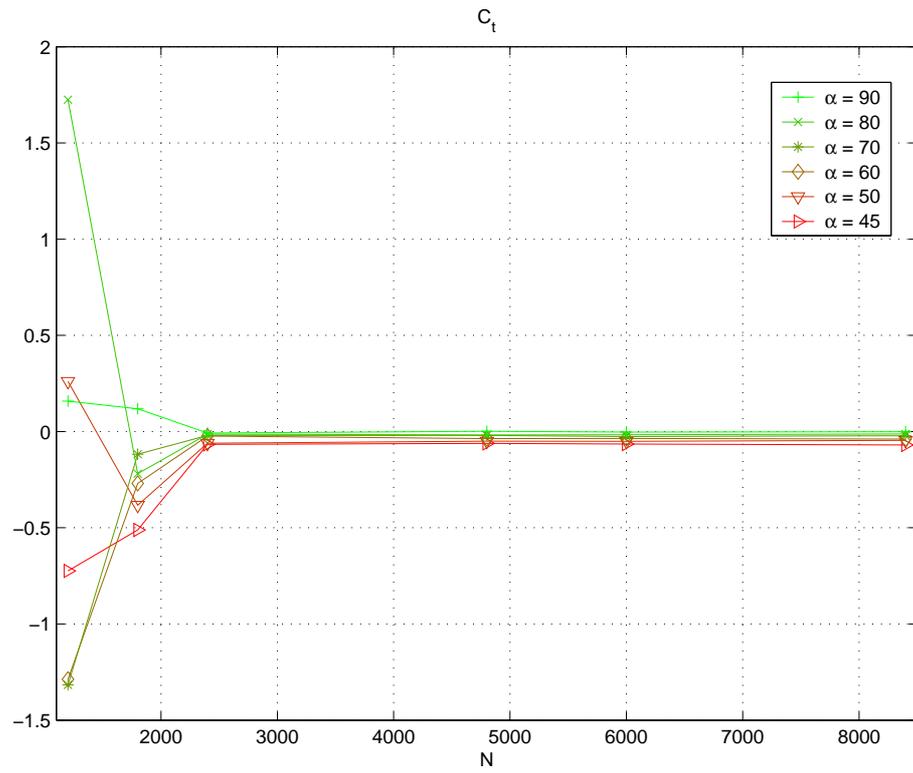


Figure 3.14: C_t for various angle of incidence and global number of vortices N

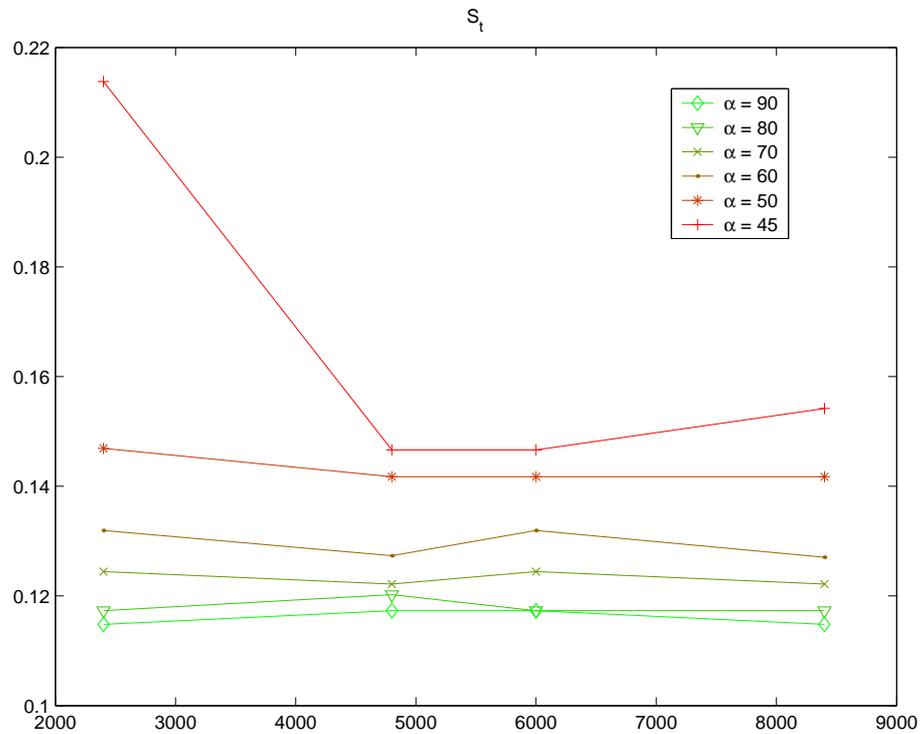


Figure 3.15: S for various angle of incidence and global number of vortices N

3.2.3 Plate at fixed angles

As stated previously, in order to test the solution produced by both codes (Spalart based code and the Sarpkaya-like code) the case study chosen is the well studied problem of an impulsively started flow over a flat plate. In this section, the focus will be on the non-rotating plate problem over a range of incidence angles and on a long term basis. The objective is to compare both codes with the few experimental results available.

Therefore, an analysis will be made of the force coefficients and the Strouhal number. Then the flow geometry will be studied through the use of streamline visualization. As no experiment provides the aerodynamic moment, they are not investigated quantitatively here. Some of the frequency characteristics of the forces will also be provided for the fixed plate, as the objective of the control will be to stabilize the plate. The general setup for the simulation is presented in figure 3.16. Generally in this section, the terms vortex and eddy do not refer to the individual blob vortices but to the actual vortex structures shed into the flow.

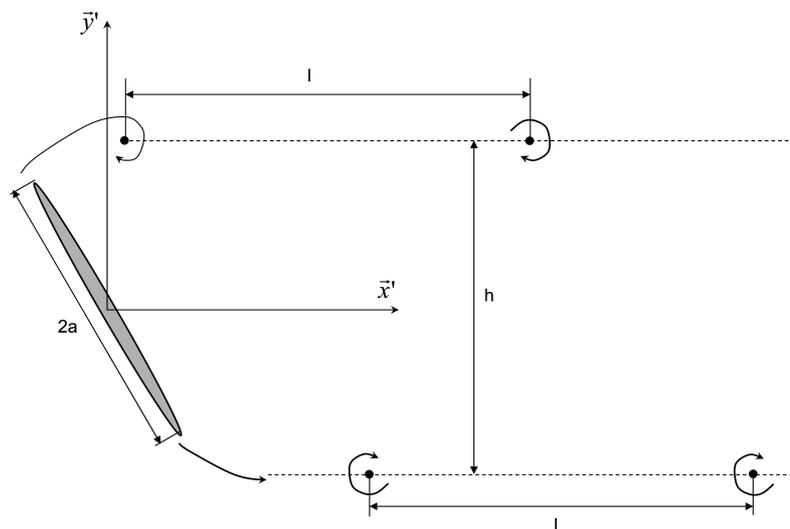


Figure 3.16: Measures characterizing the flow

Concerning the aerodynamic forces, long term results will be plotted using equation 2.59 for the Sarpkaya-like simulation, and equation 2.45 for the Spalart code. While both formula are akin to an integration, note that formula 2.59 used for the Sarpkaya-like code utilizes data from each free vortices in the formula, whereas equation 2.45 uses data only from boundary vortices. Thus equation 2.45 effectively introduces a grid for the force calculation, which in return induces a noisier result. It is however not possible to use formula 2.59 with the Spalart code because of the merging which means that the time-dependent coefficient cannot be evaluated properly. Therefore to smooth the results, for the time plots and formula 2.45 a simple three-point filter was used of the form:

$$\vec{F}_{s,i} = \frac{\vec{F}_{i-1} + 2\vec{F}_i + \vec{F}_{i+1}}{4}, \quad (3.7)$$

where \vec{F}_s is the smoothed force, \vec{F} is the original force, and the subscript i represents the measure taken at the i^{th} time step.

As stated in an earlier section, because those two formula are integrations over a given domain (whether at the surface boundary for formula 2.45 or outside the boundary for formula 2.59), the results provide only broad insight of the flow, but local errors can occur and cancel each other. That is why a region-by-region analysis is necessary, however it is difficult here due to the lack of experimental data. In addition to the history plot of the force, the time averaged force coefficient will also be used in order to compare it with the data from Fage and Johanssen [16]. The forces are given in the fixed body frame (\vec{x}, \vec{y}) , which has the \vec{x} axis aligned with the ellipse major axis as in figure 2.1.

The normal force coefficient C_n and the tangential force coefficient C_t (related to the \vec{y} and the \vec{x} axis respectively) were used according to the definitions in section 3.2.2, except that the ellipse chord $2a$ was used instead of a for both codes. This non-dimensionalization convention applies for every parameter. Therefore the nondimensional coordinates corresponds to $x^* = x/(2a)$ and $y^* = y/(2a)$. Note that in this chapter, the $*$ denotes nondimensional variables.

Simulations using the Spalart code were done with an inviscid flow, $\Delta t^* = 0.05$, $N_b = 1200$, $D_0 = 4a$ and a total of 8400 vortices in the flow (including the boundary vortices) after 2048 time steps in order to have a long term solution available. The

time averaged force coefficients are taken from the 1024th time step so that the simulation can be almost certainly in a stabilized semi-periodic regime. Similarly, the power spectrum was taken from the 1024th time step to the end of the simulation using a FFT. Indeed, the peak frequency of the C_t FFT over different time windows is steadily decreasing from the simulation beginning until it reaches a stable frequency at a variable time depending on the incidence angle. This particular point will be discussed later in the chapter.

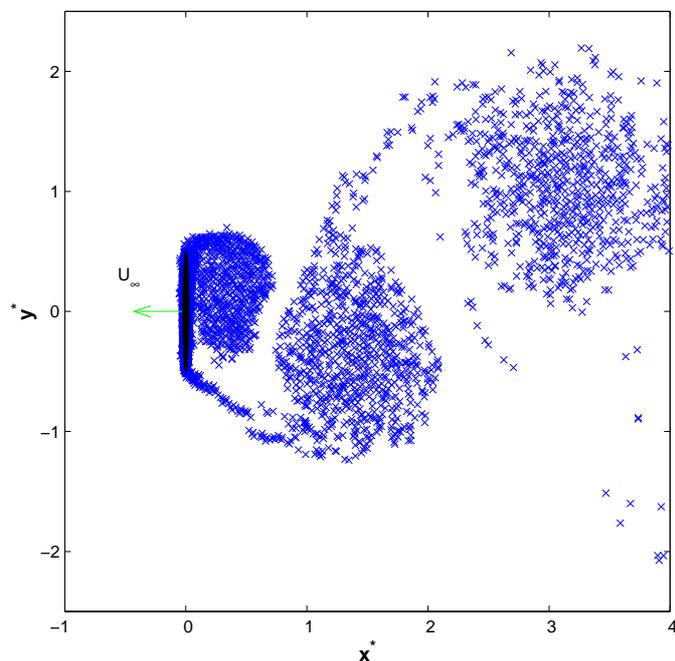


Figure 3.17: Vortex plot at a given time step from the Spalart code. Vortex are blue crosses, the body is black, the direction of the ellipse motion is indicated by the arrow

Sarpkaya [50] noted that the shedding frequency is about equal to the frequency of the force fluctuations. The Strouhal has thus been taken as the peak in the tangential force coefficient power spectrum, that is to say the force main period. In order to verify the data, the Strouhal number has also been extracted by using plots of vortices such as the one shown in figure 3.17. Although the plot here is only a crude representation of the vortex shedding, the blob vortices pattern clearly indicates a vortex shedding from the lower tip of the plate. The vortex shedding frequency can then be quickly extracted by noting the time where the individual vortex shedding takes place in these plots. Comparison will be made with experimental results by Fage and Johanssen.

Now concerning the Sarpkaya-like simulation, one problem was that as the incidence decreases from 90 to 45 degrees the simulation becomes more and more unstable until eventually the vortex shedding breaks down. The instabilities are manifesting by a breakdown in vortex structure as the vortices are fed with vorticity near the plate leading to an irregular shedding. This is due to equation 2.37 whose resolution is done using a Broyden algorithm [64].

After the first vortex shedding and as the simulation approaches the next shedding, blob vortices shed from one edge comes closer to the other edge as the vortex grows. Two problems then arise, first because there are also blob vortices of opposing circulation it creates some singularities locally where the nascent vortices are introduced. Secondly one has seen in figure 3.5 that the velocity fluctuation is smooth but irregular near the ellipse wall. The Broyden algorithm has proven to be sensitive to these two problems. Furthermore, the different parameters required (initial points position, tolerance for position, function error tolerance) for equation 2.37 have an important influence on the simulation long term stability. Regarding these parameters, the only robust way to improve the simulation was found to reside in decreasing the time step. This solution is also limited as, due to computing power required, it is difficult to use more than 2000 time steps, hence limiting the usable time range.

Consequently, a good compromise has been to use a time step of $\Delta t^* = 0.04$ for the Sarpkaya-like simulation. However, using $\Delta t^* = 0.01$, the simulation remains stable for a longer time.

Again because there are fluctuations in the nascent vortices strength, a more restricted time range was used for the power spectrum. The initial time step is taken at the beginning of the vortex shedding (typically at $t^* = 8$), and the latest time step is the 1000th time step ($t^* = 40$ with $\Delta t^* = 0.04$), except in the case $\alpha = 90^\circ$, because separation occurs after a longer time. For that case, as for the Spalart code, the power spectrum was taken from the 1024th time step to the end of the simulation. The time-averaged force coefficient used the same ranges.

As for the Spalart code, the Strouhal number was extracted as the frequency of the peak in the tangential force coefficient spectrum. As a complement and similarly to Sarpkaya [50], the actual strength of the nascent blob vortices released into the flow

was also plotted. A typical plot of this kind is presented in figure 3.18. As described in section 2.3.4, it represents the actual vorticity released into the flow at each timestep. Note that with the Sarpkaya-like code, in figure 3.18 the zero shed vorticity means that the blob vortex has been erased because of a too strong vorticity after its creation. The comments show how one can then relate the nascent vortex strength to the vortex shedding.

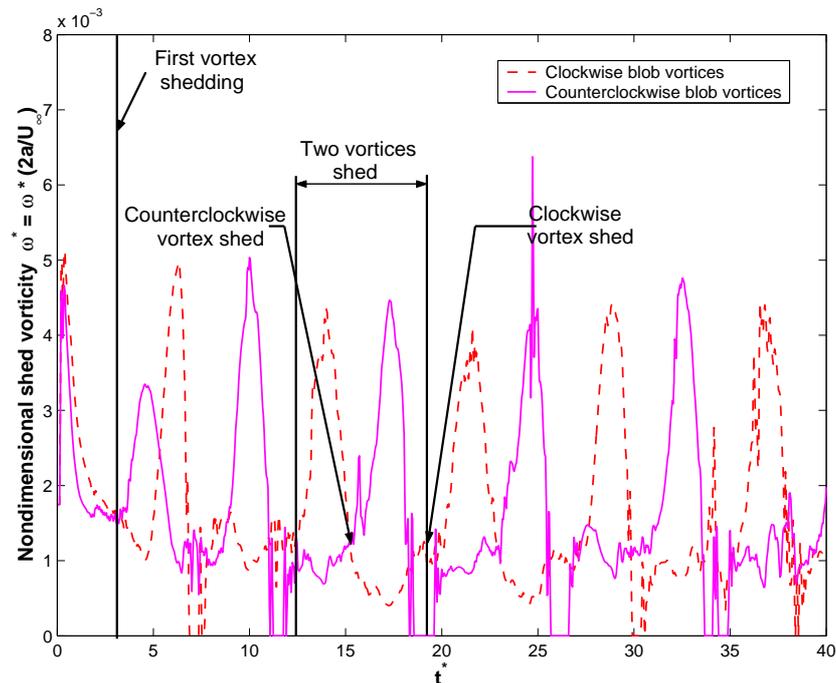


Figure 3.18: Nascent vortices strength shed from the ellipse for $\alpha = 80^\circ$. The filter is defined by equation 3.7.

Now concerning the streamline plots, despite the different time step used for the Sarpkaya-like simulation, whenever possible the same timestep was used for the streamlines plot. In general, t^* was taken as 51.5 except for $\alpha = [80, 90]$. This is because for $\alpha = 80^\circ$ the vortex shedding breaks down past $t^* = 40$, and thus it seemed preferable to show the streamline when a stable vortex shedding was present. As for $\alpha = 90^\circ$, the vortex street only takes place after a longer time, and again it seemed preferable to show how the code handles the vortex street once the vortex shedding has taken place.

For the streamlines plots in figures 3.19, 3.20, 3.21 and 3.22 the streamline values range was defined as $[-2.66, 2.66] \times (2aU_\infty)$ with a concentration around 0. In figure

3.23 the streamline values range is $[-9.63, 9.63] \times (2aU_\infty)$ with a concentration around 0. Note that with these values, not all streamlines are visible.

Concerning the Spalart code simulations, one can see from the streamlines plot 3.19 and 3.20 that the vortex street is seemingly becoming irregular at the end of the plot, e.g. at $x^* \simeq 12$. However, larger streamline plots, such as those in figure 3.23, reveal past $x^* \simeq 12$ a slight upward trajectory of the vortex pairs as they flow downstream. This is a symptom of flow instability, as several shedding modes can occur. Such phenomena has also been observed by Clarke [11] with a vortex method for a flow over a cylinder with Reynolds number of 5000 and 31700.

He proposed that this flow asymmetry is a consequence of the nonlinearity of the Navier-Stokes equations. This is consistent with the Sarpkaya-like simulation conditions as asymptotically an inviscid flow can be considered as a flow with a very high Reynolds number. Furthermore, a blob vortices merging process was used which is numerically very noisy.

According to Clarke : “a simple consequence of vortex dynamics is that two concentrated regions of opposite-sign vorticity may convect each other, causing a flow pattern similar to a dipole. We note that placing side walls on an experiment will tend to suppress side motions. Finite difference boundary conditions will probably have a similar effect.”. Remark that the fact that both codes show asymmetry implies that this phenomenon is indeed physical.

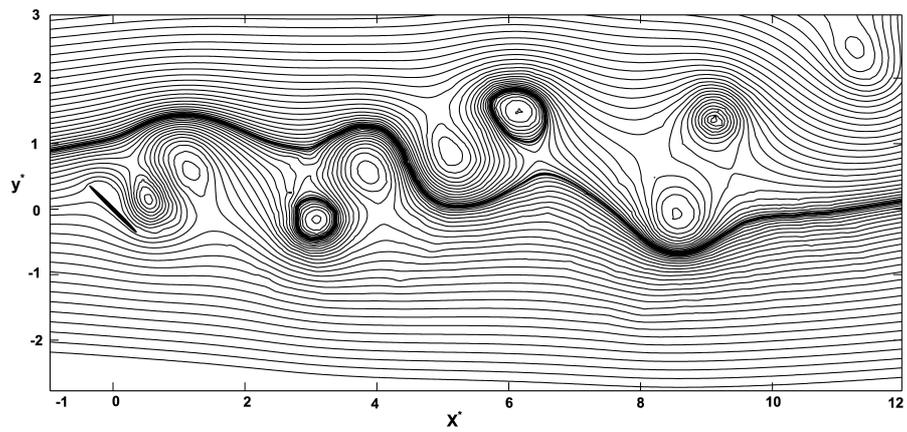
At this incidence, the flow has proven to be relatively insensitive to numerical parameters except the incidence angle α . However, as the incidence angle varies from 50 to 90 degrees the flow becomes more and more symmetric. At 90 degrees, the vortex street is fully symmetric and shows no transverse vortex motion. Thus, the asymmetry seems to be linked to the body asymmetry compared to the flow. As for the blob vortices merging process, it can only be conjectured that it introduces numerical noise farther downstream from the body owing to the implementation which tends to emphasize the near-body wake. As this is not the main subject here, this particular point is left to future studies.

To summarize, there is a clear tendency for transverse motion of the vortices, that

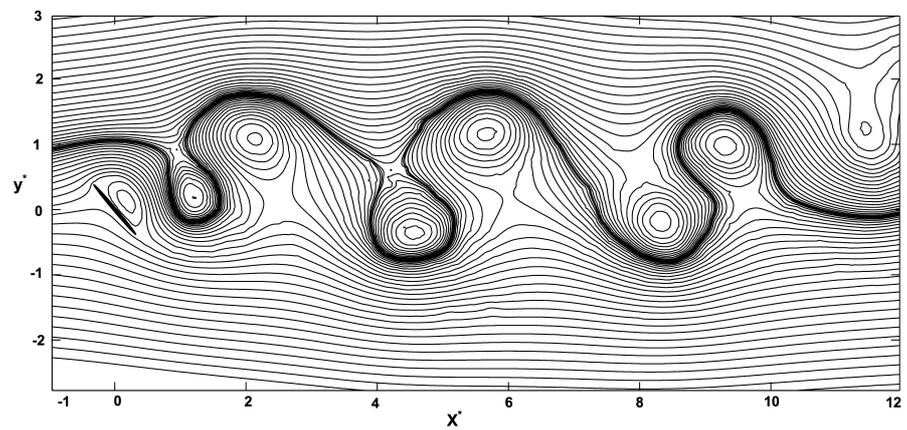
is, there is an angle between the general vortex direction and the freestream flow direction. This angle tends to be more pronounced as the angle evolves from 90 to 45 degrees (with the vortex street symmetric at $\alpha = 90$ degrees). It marks the influence of the body asymmetry. Indeed, the freeflow incidence causes a nonzero average circulation around the body which in turn influences the vortex shedding. Remarkably, the added circulation to the body is independent of the numerical parameters and is found to be equal to $\Gamma \simeq 0.24 U_\infty 2 a \cos(\alpha)$.

The streamline plots from the C code are presented in figures 3.21 and 3.22. Overall, one can see the streamlines pattern are similar to those from the Spalart code. Indeed, for $\alpha = [45, 50, 60, 70]^\circ$ (see also figure 3.23), the vortices also exhibit the transverse asymmetry mentioned above. However in this case, as there is no merging process, the numerical noise presumably comes primarily from the nascent vortices strength determination. Again, the flow becomes more symmetric as the incidence angles approaches 90° , and the asymmetry is essentially gone at $\alpha = 90^\circ$.

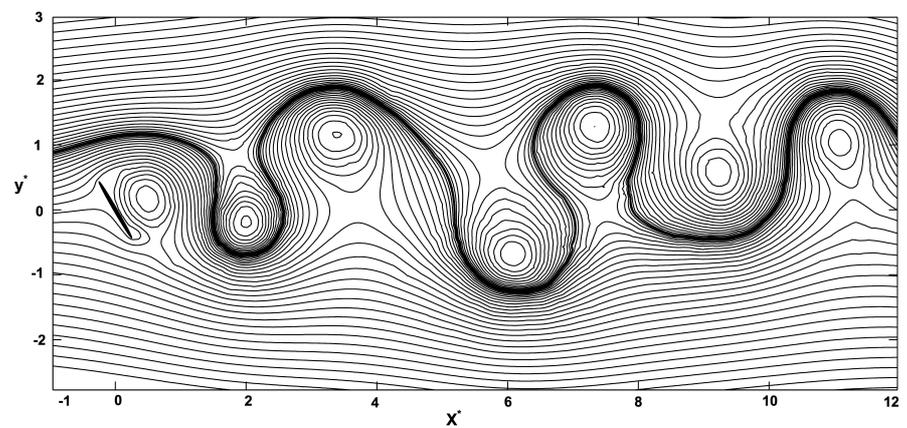
Therefore, it can be concluded that the C code and Spalart code produces similar qualitative results as far as the overall velocity field is concerned.



(a) $\alpha = 45^\circ$

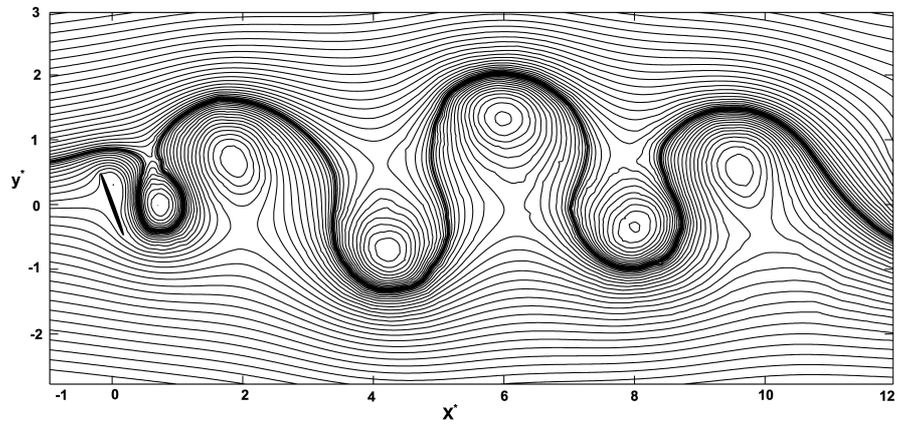


(b) $\alpha = 50^\circ$

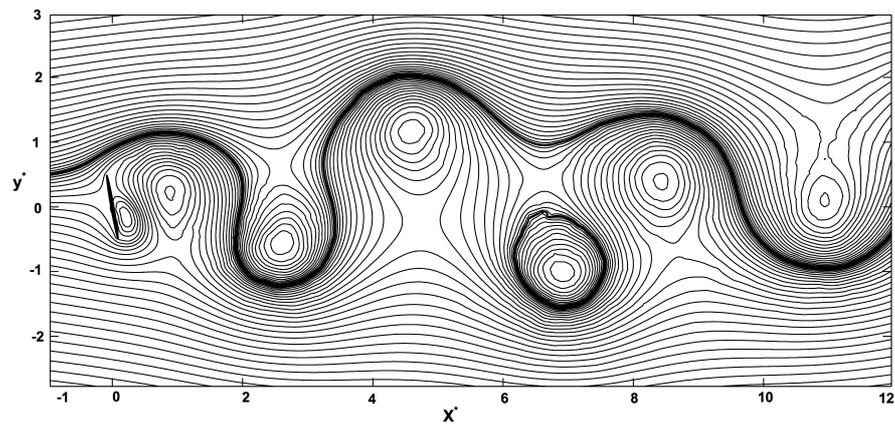


(c) $\alpha = 60^\circ$

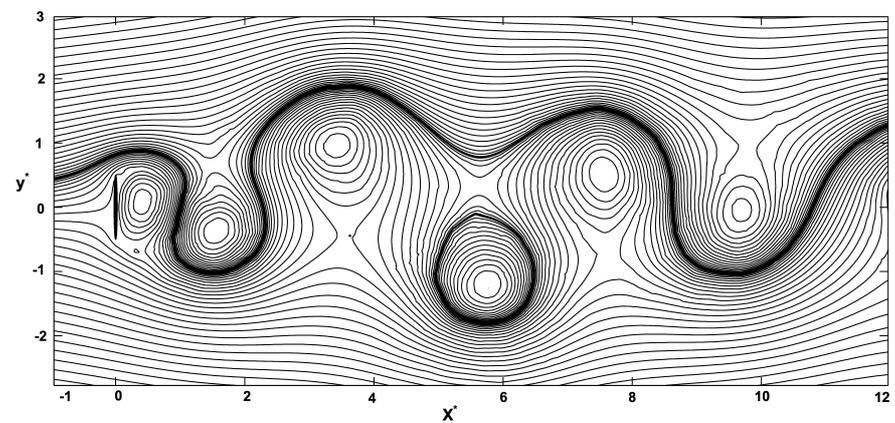
Figure 3.19: Streamline of the flow for the Spalart code at $t^* = 51.5$ with $\alpha = [45, 50, 60]$



(a) $\alpha = 70^\circ$

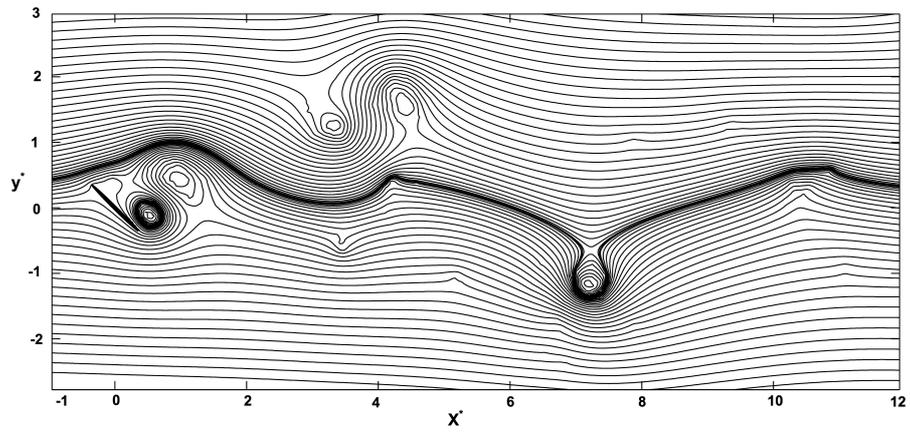


(b) $\alpha = 80^\circ$

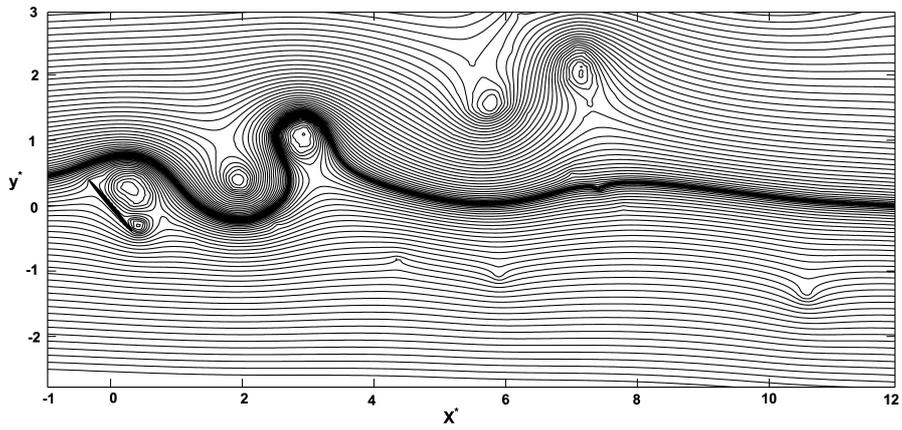


(c) $\alpha = 90^\circ$

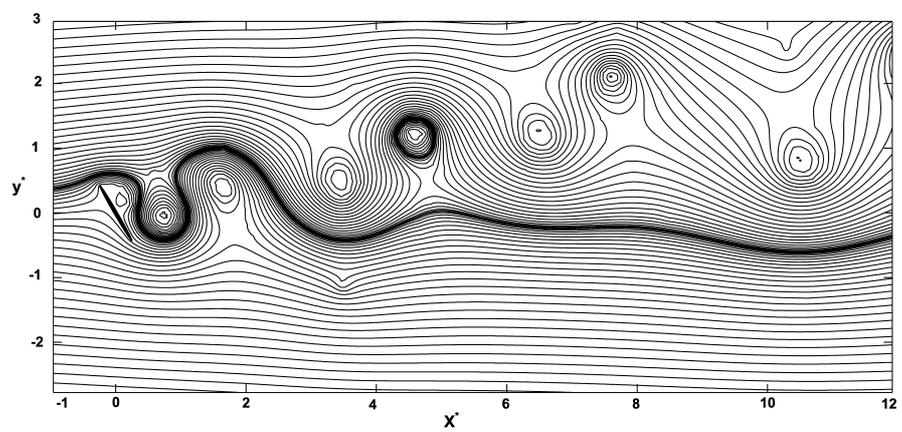
Figure 3.20: Streamline of the flow for the Spalart code at $t^* = 51.5$ with $\alpha = [70, 80, 90]$



(a) $\alpha = 45^\circ$

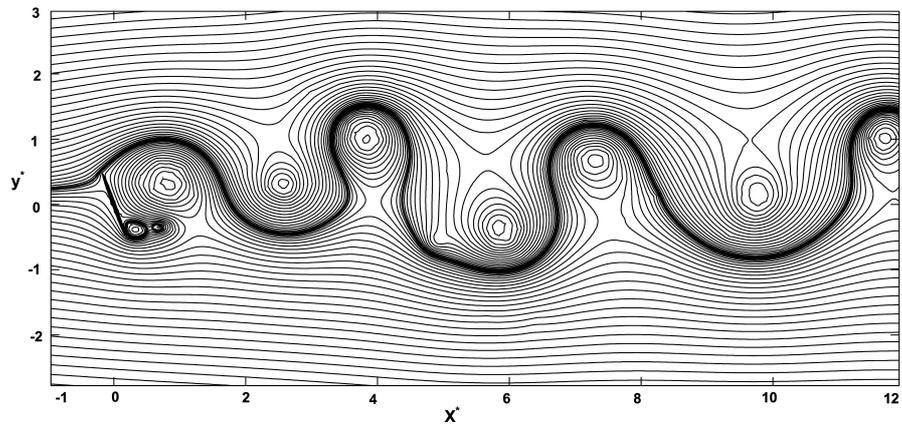


(b) $\alpha = 50^\circ$

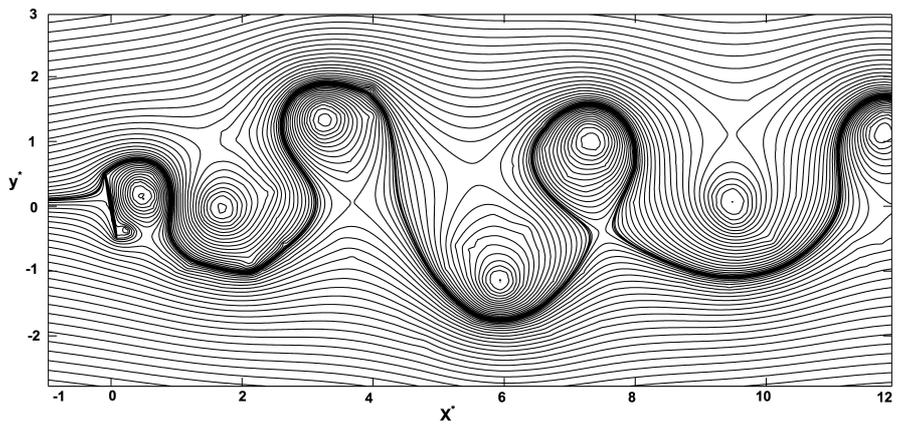


(c) $\alpha = 60^\circ$

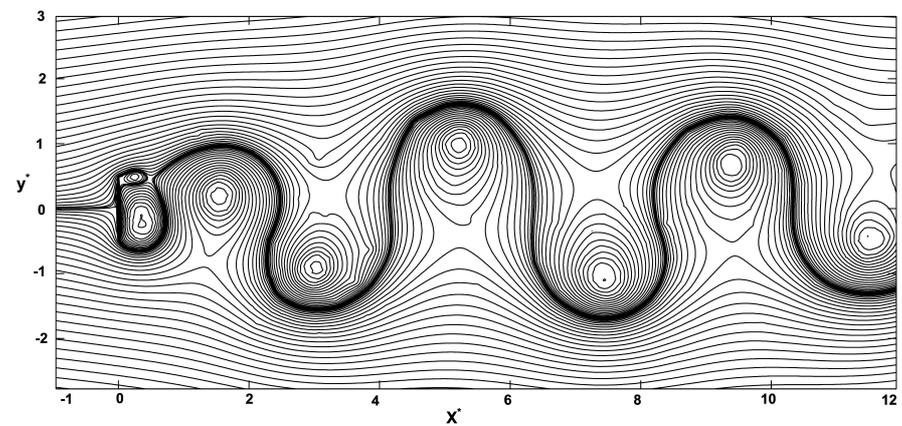
Figure 3.21: Streamline of the flow for the C code at $t^* = 51.2$ with $\alpha = [45, 50, 60]$



(a) $\alpha = 70^\circ$

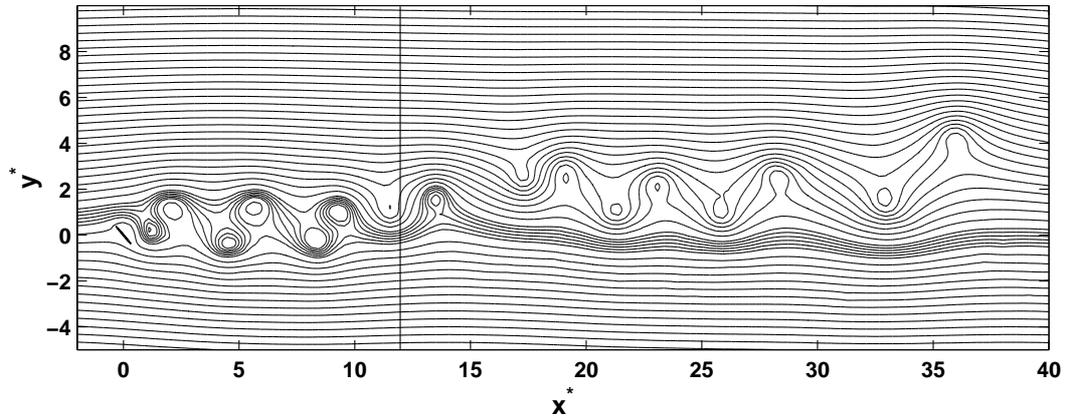


(b) $\alpha = 80^\circ$

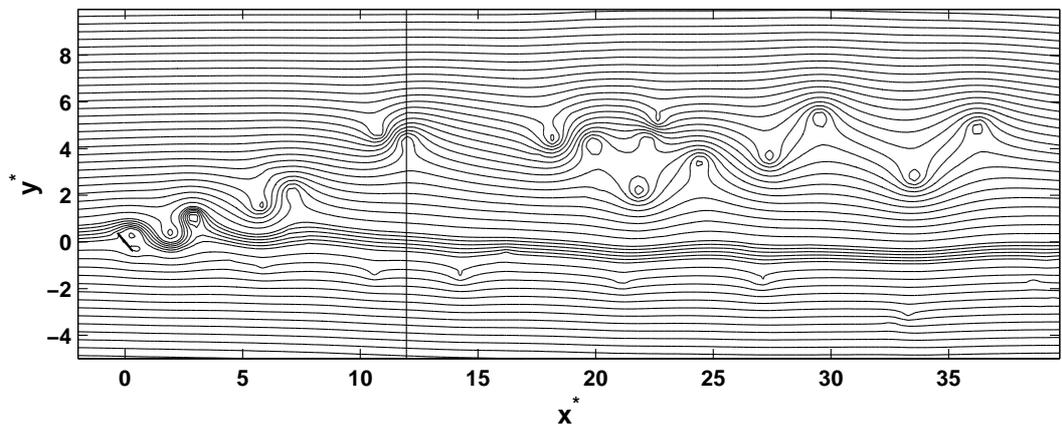


(c) $\alpha = 90^\circ$

Figure 3.22: Streamline of the flow for the C code at $t^* = 51.2$ for $\alpha = 70^\circ$, $t^* = 40$ for $\alpha = 80^\circ$ and $t^* = 72$ for $\alpha = 90^\circ$



(a) Spalart code



(b) C code

Figure 3.23: Enlarged streamline plot at $t^* = 51.5$ for the Spalart code for $\alpha = 50$ (the black line represents the end of the streamline plots in fig 3.19 and 3.20)

To turn to the aerodynamic forces coefficient, from table 3.1, the Spalart method overestimates C_n by an average of 75%. This is consistent with his previous simulations (see reference [57]) for a flat plate at 90 degrees incidence, where C_n was found to be too large by 60%. This can be explained by the fact that the vortices shed are much stronger than those found in experiment. Indeed vortex shedding along a flat plate is a mainly three dimensional ($3D$) process, and furthermore the actual vortices also have a slight oscillation motion spanwise which helps diffuse the vorticity, whereas the vortex simulations here are purely two dimensional ($2D$) and there is no mechanism to account for the spanwise diffusion of vorticity or viscous diffusion.

α	Fage and Johansen data [16]			Spalart code			C code		
	C_n	C_t	Strouhal S	C_n	C_t	Strouhal S	C_n	C_t	Strouhal S
45	1.53	N/A	0.205	2.83	-0.06	0.155	2.42	0.0461	0.198
50	1.63	N/A	0.196	2.84	-0.044	0.140	2.67	0.0516	0.184
60	1.76	N/A	0.173	2.97	-0.05	0.127	2.9	0.011	0.207
70	1.84	N/A	0.156	3.23	-0.016	0.123	2.75	-0.014	0.171
80	1.86	N/A	0.152	3.29	-0.018	0.118	3.3	0.033	0.154
90	1.85	N/A	0.146	3.29	-0.003	0.113	2.93	-0.01	0.141

Table 3.1: Normal force and Strouhal number for the Spalart and C code

More interestingly, the Strouhal number is equal to $S \sin(\alpha) = 0.1119 \pm 0.006$ and is less than the $S \sin(\alpha) = 0.148 \pm 0.003$ found by Fage and Johansen. Several elements may explain this discrepancy. First, it could be a consequence of the error observed in section 3.2.1.1. In the steady state case, the velocity error was more pronounced on the ellipse edge following a pattern similar to that of two point vortices placed at the edges of the ellipse. Therefore although the added circulation resulting from this error is zero, there is actually an increase in the vorticity shed from both tips. Then one could expect an effect similar to that found by Sarpkaya and Shoaff [55] for the circular cylinder, for which the larger the strength of the vortices in the near wake, the smaller the Strouhal number and vice versa.

From trials with varied core radius values, boundary blob vortices distance from the wall and time-step, it appears that the blob vortex core radius σ is a key parameter in ensuring a correct Strouhal number, though it has little effects on the force coefficients. Additionally, the core radius value is also important for the characterization of the other parameters. Indeed, σ is one of the main parameters representative of the degree of blob vortex core overlapping.

The σ influence is illustrated in figure 3.24, where the product $\sigma\sqrt{N}$ enables to compare different core-radius-to-grid ratio σ/h . h represents the length of a square cell in a uniform grid defining the approximate flow domain simulated. It was chosen to

approximate the flow domain by a $27a \times 12a$ window, which is a window similar to that of those in figure 3.19. This window encompasses 80% of the actual number of vortices once the actual number of vortices is stabilized around N . As the area of an individual square cell of length h is h^2 , it enables to compute the approximate flow domain area as $0.8 N h^2 = 27a \times 12a$. This leads to establish the relationship $h \propto (1/\sqrt{N})$. These simulations were all done for $\alpha = 50^\circ$, and the Strouhal error is given in % relative to the experimental Strouhal at this angle. Otherwise, the Strouhal number data in figure 3.24 were obtained with the same procedure as for the Spalart code described in above paragraphs. Note that despite some cases where the Strouhal number error is less than 5%, it does not mean that S will be correct with the same parameters for other incidences. As a reference, in a test case from Blevins [7] for a circular cylinder which produced correct Strouhal number, he used $\sigma\sqrt{N} = 0.51$.

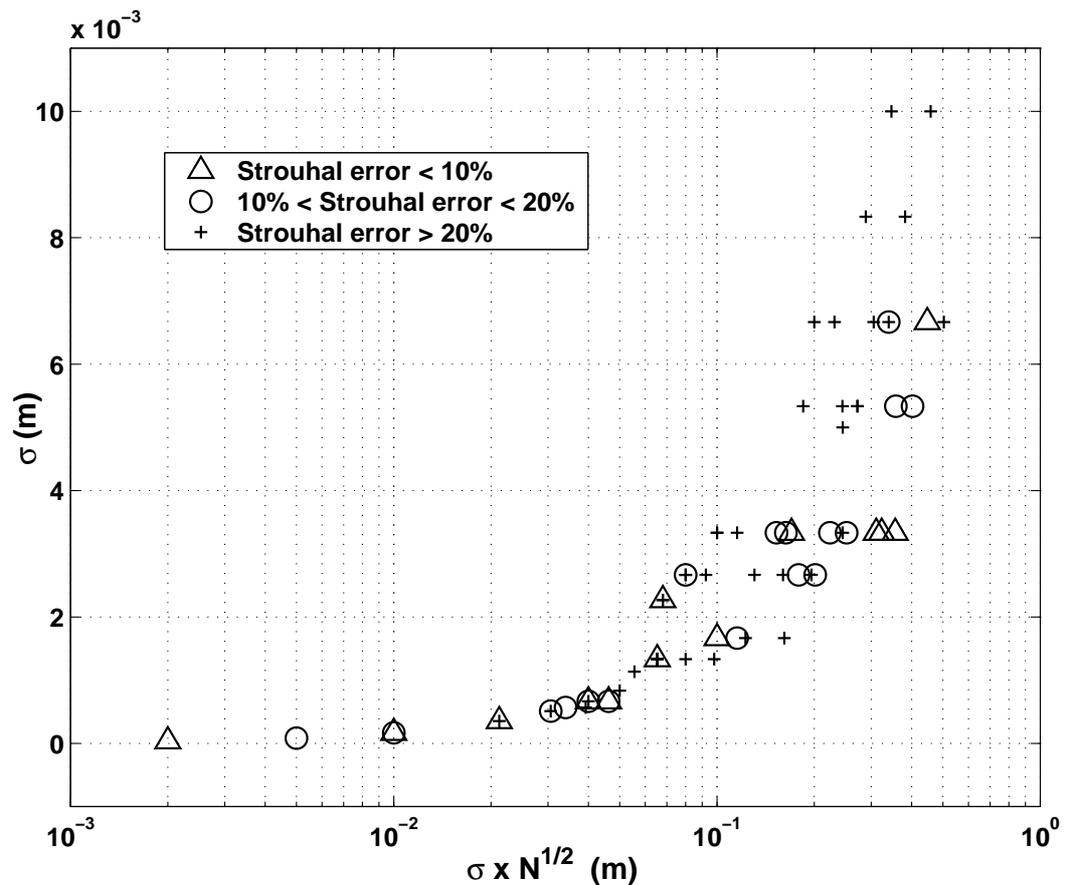


Figure 3.24: σ and $(\sigma)\sqrt{N}$ influence on the Strouhal number for $\alpha = 50^\circ$ using the Spalart code. Both variables units are in meter.

When the core radius is too small and the total number of vortices N is too small, the simulation tends to act similarly to a smoothed point vortex method. As the core radius further decreases there is then more and more leakage between the wall control points and larger velocity variation near the boundary (figure 3.5), raising the velocity field error near the boundary. Consequently, the leakage and velocity field error tend to create an artificial vorticity diffusion near the boundary (similarly to a random-walk method). This is visible when $\sigma\sqrt{N} < 0.1$ in figure 3.24, as in this range the simulation tends to converge toward the correct Strouhal number.

Conversely, a too large σ value, relatively to N , has two effects. First, the numerical boundary layer – defined as the space between the wall control point and the boundary blob vortices – is too thick, thus artificially lowering the Reynolds near the wall. More generally, a large σ value generates additional vorticity diffusion once the blob vortices are shed. These cases can be seen in the upper right corner of figure 3.24.

Both effects imply a lower Strouhal number S than for experimental results, but the amplitude of S in the thin ellipse case has been unexpected, especially since it is recognized that in a vortex method S is typically insensitive to the numerical parameters. Note that the overlap condition is that the convergence is attained when $N \rightarrow \infty$, $h \rightarrow 0$, $\sigma \rightarrow 0$ but σ not as fast as h (Spalart [57]). When values of σ are bounded – as is the case here due to the boundary model – and recalling that h is related to $\sigma\sqrt{N}$, one must raise N in order to lower h so that, for example, when one uses a lower core radius $C\sigma$ for a given N (with $C < 1$ is a real constant), one uses a total number of vortices $(1/C)^n N$ with $n > 2$. Thus, one has $C^{n/2} < C$.

Despite many trials, it has seemed impossible to find a set of parameters Δt , σ , δN and N able to produce a satisfying Strouhal for a range of flow incidence $\alpha = [50, 60, 80, 90]$ degrees. However, this only influences the flow shedding frequency, as the vortex street is present at all times. Note that it has been possible to find a good set of parameters in the case of the cylinder, suggesting that in that case a proper set of parameters does exist with a dramatic increase in the number of vortices.

In figures 3.25, 3.26, 3.27 the forces coefficient history are presented for the Spalart code. Each time the C_n and C_t follow a similar evolution. After a starting phase at early timestep, both force coefficient then set into a stabilized phase. This is especially

visible at $\alpha = 45^\circ$ in figure 3.25(a), where the starting phase is visible up to $t^* \simeq 35$. In both phases, the force coefficient evolution is periodic which is not surprising as it reflects the fact that the flow is always in vortex shedding mode. Note that the duration of this starting phase is reduced as the incidence angle is increased to $\alpha = 90^\circ$.

From the C_t plot, one can see that generally during the starting phase, the frequency is actually higher than during the periodic regime. Again, this is best visible at $\alpha = 45^\circ$ in figure 3.25(a). During this phase the Strouhal number has been found to be actually closer to the experimental Strouhal number than during the stabilized periodic phase. It is not clear why there are such discrepancies between the starting phase and the stabilized periodic phase, or even why there is a starting phase reduction when incidence angle increases. In the absence of experimental results (Fage and Johansen study [16] was about the periodic behavior), one cannot clearly part the physical-flow behavior from the numerical induced effects. Nevertheless, from the C_t frequency evolution in figure 3.25(a) and 3.25(b) (Strouhal number closer to the experimental number in the starting phase than in the stabilized periodic phase), it can again be conjectured that this is another consequence from the lack of overlap.

As for the C code, from table 3.1, one can see that it also overestimates the C_n by an average of 62%, although this is better than for the Spalart code. As explained above, this is mainly attributable to the $2D$ nature of the simulation which is unable to take into account for the three-dimensionality of the flow. On the other hand, the Strouhal number is in much better agreement on average with the Fage and Johansen value $S \sin(\alpha) = 0.14 \pm 0.04$, note however that the deviations around the mean S are larger than for the Spalart code.

From the forces coefficient history plots in figures 3.28, 3.29, and 3.30 one can see that the C_t plots are significantly different from the ones coming from the Spalart code although they both have a near zero mean value. Namely, in the C code the C_t tends to have a transition period where after a peak value, the C_t tends to stagnate near a zero value for some time before reaching another peak. Conversely, for the Spalart code there is no such period and the C_t crosses the zero with a constant slope.

This is again a problem related to the nascent blob vortices strength and position determination. For instance, at $\alpha = 80^\circ$, the plot in figure 3.18 shows that the simulation has difficulties in determining a nascent blob vortices strength when a vortex is shed. During this phase, the nascent blob vortices are introduced close to the ellipse tips almost symmetrically, and their strengths are also about equal. From equation 2.51 one can see that the closer the vortex is to the ellipse tip, the greater its influence on the force coefficients along the ellipse main axis. Therefore, it is easy to see that C_t will be close to zero when the nascent blob vortices are in the situation described above. More generally, the difficulties in determining the nascent blob vortices result in noisier force coefficient time plot than those coming from the Spalart code. FFT spectra histories, however, are able to clearly distinguish a single large-amplitude low frequency mode with which to calculate the Strouhal number.

Otherwise, in almost every C_n history plot except for $\alpha = 90^\circ$ one can see some vertical lines. This is actually some noise due to a very high value of C_n . This is related to the same problem as for the C_t determination, that is the nascent blob vortices strength and position determination. For example, at $\alpha = 80^\circ$ (figure 3.30(a)), at around $t^* = 25$, there is single vertical line. Now figure 3.18 shows that around this time there is a vortex shedding with nascent vortex strength fluctuation (notice the high vortex strength around $t^* = 25$), moreover it is probable that a nascent vortex has been introduced too closely to the ellipse edge which provokes this C_n singularity.

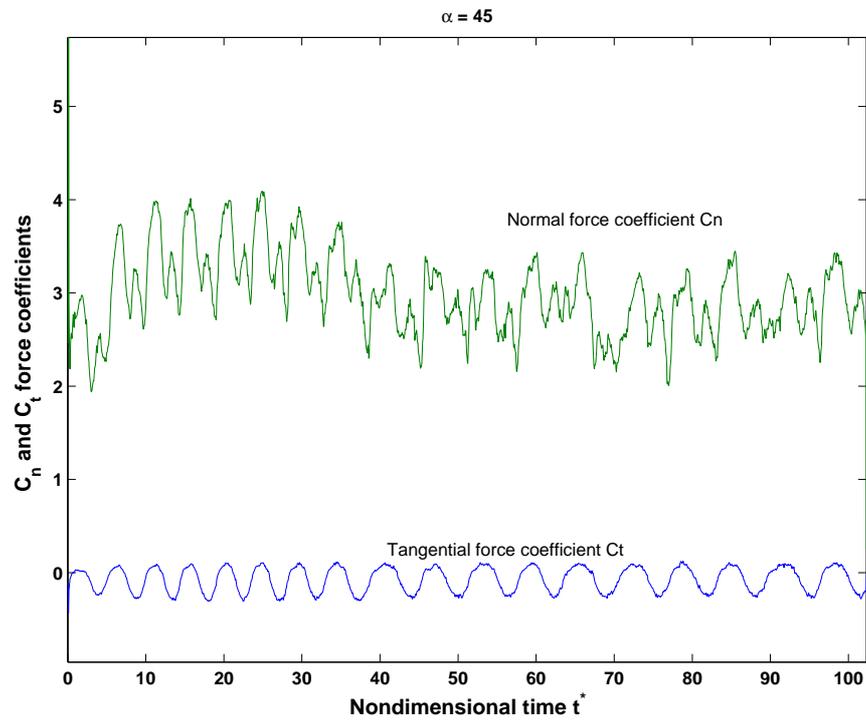
Nevertheless, the C_n and C_t are similar to those originating from the Spalart code (figures 3.25, 3.26, 3.27) with a starting phase and a stabilized periodic phase except that the starting phase is almost nonexistent. Yet remark at $\alpha = 90^\circ$, the C_t and C_n evolution is slightly different from the other forces coefficient history for the C code. Indeed, there seems to be a long starting phase up to $t^* \simeq 23$ before the flow reaches a stabilized periodic phase. This simply because up to $t^* \simeq 23$, there are two symmetric eddies originating from both edges (similarly to the start of a flow over an impulsively started cylinder).

Afterwards, the flow becomes asymmetric and stabilizes into a vortex street. As stated previously, this flow sequence is similar to that of a flow regime over an impulsively started cylinder and is predictable. It does not occur with the Spalart simulation, because the separation is triggered from the simulation beginning through a negligible

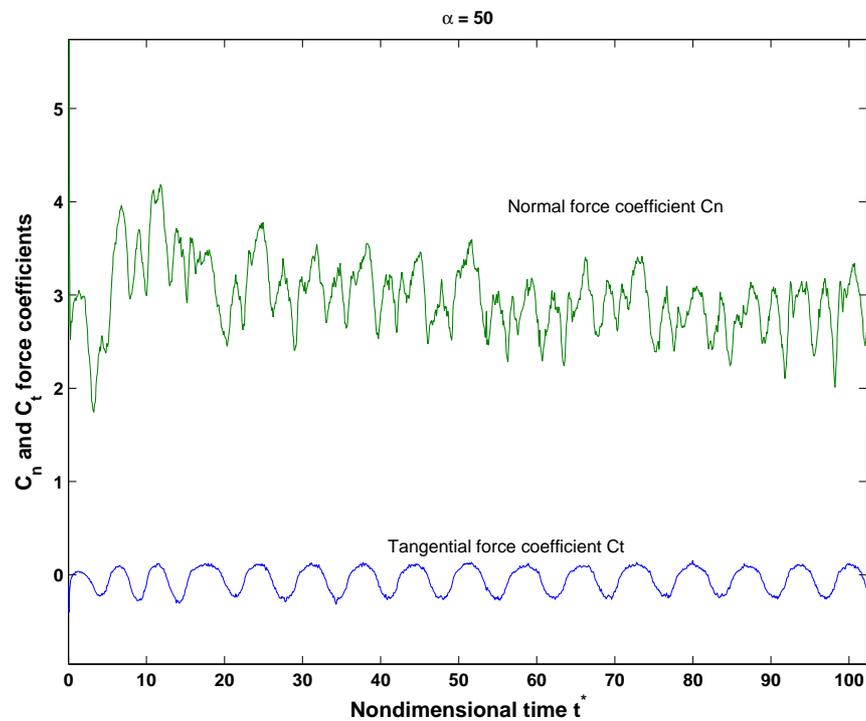
circulation injection on the body wall. It was not used with the C code, because it favors the kind of noise in the C_n and C_t seen for the C_n as described in an earlier paragraph.

From this comparison, it is clear that although the discrete blob vortices provide better quantitative results than the Spalart code for the current test case, its introduction of numerical noise – mainly due to the nascent blob vortex strength determination algorithm – is more important than for the Spalart code, as can be seen from the Strouhal number results. On the other hand, the Spalart code, despite poor quantitative performance, still manages to enable a consistent vortex street for all incidence angles considered. Note that the Spalart results could be improved by injecting many more vortices and using different core radius, but as no formal procedure exists to find a set of parameters where the simulation converges toward the correct Strouhal number, it can only be found empirically. Maybe optimization methods such as Monte-Carlo algorithm could help optimize the set of parameters, but these come with a high computing cost.

Additionally, to help control the blob vortices vorticity (and thus have a better control over the introduction of vorticity), it would be more efficient to use a viscosity mechanism such as the ones used by Clarke [11] and Takeda [59]. As a last remark, note that it is difficult to make a meaningful comparison concerning the streamlines owing to the lack of experimental results. However, both codes are sufficiently accurate to suggest that it is reasonable to use them in the control strategies discussed below.

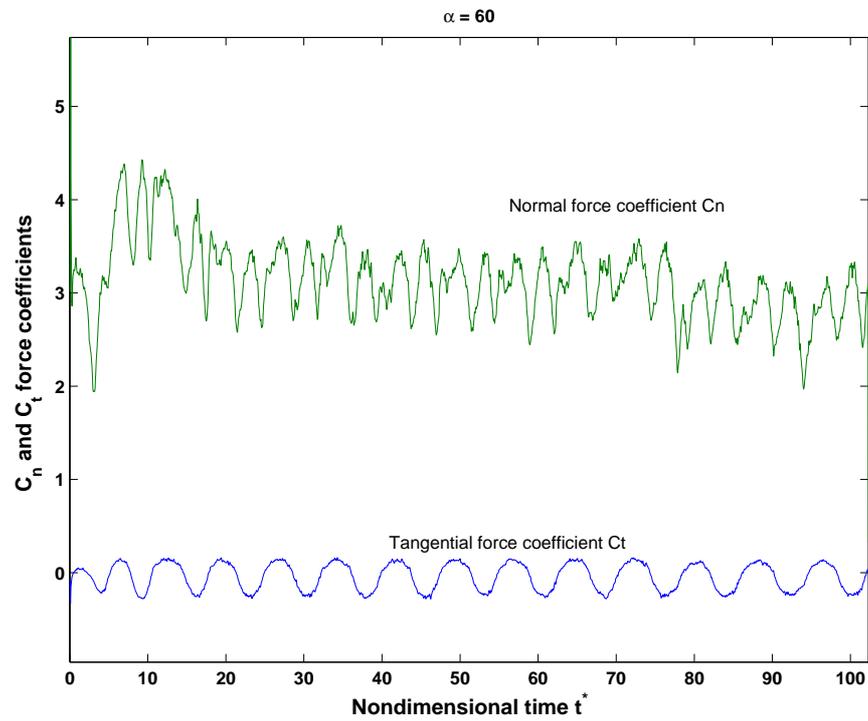


(a) $\alpha = 45^\circ$

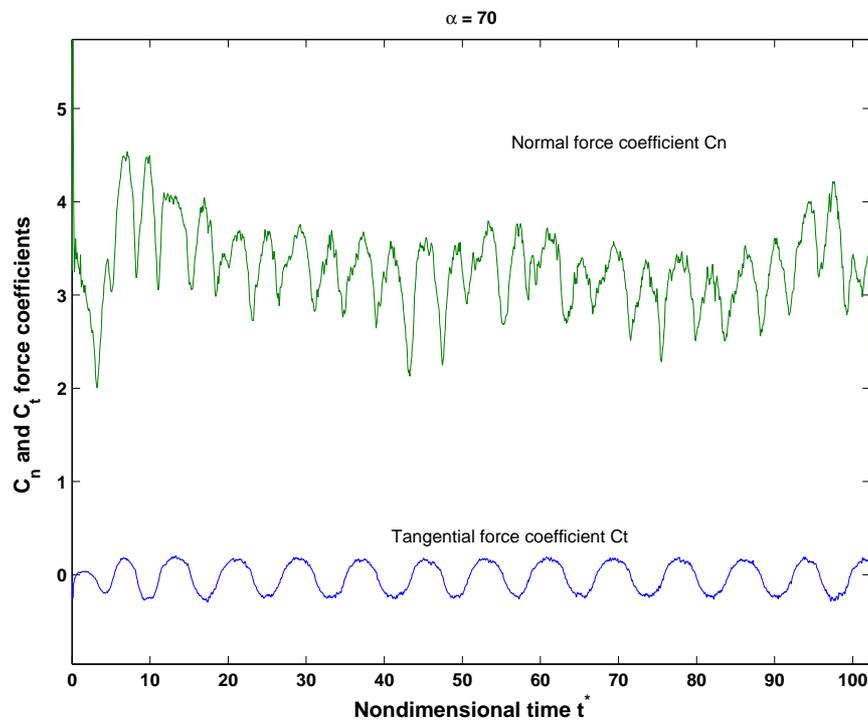


(b) $\alpha = 50^\circ$

Figure 3.25: Normal and tangential force coefficients time plot for the Spalart code with $\alpha = [45, 50]^\circ$

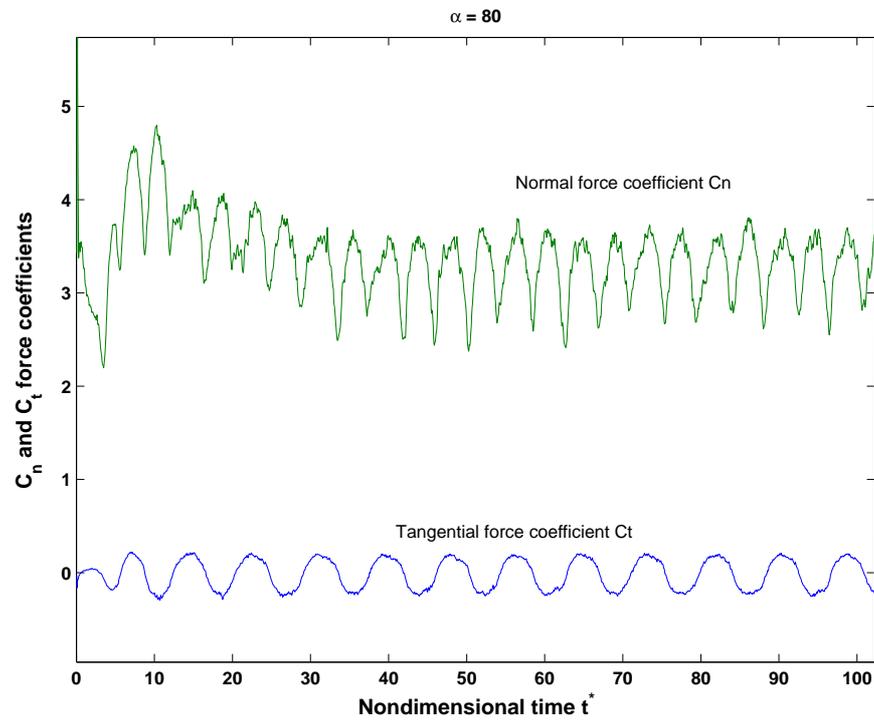


(a) $\alpha = 60^\circ$

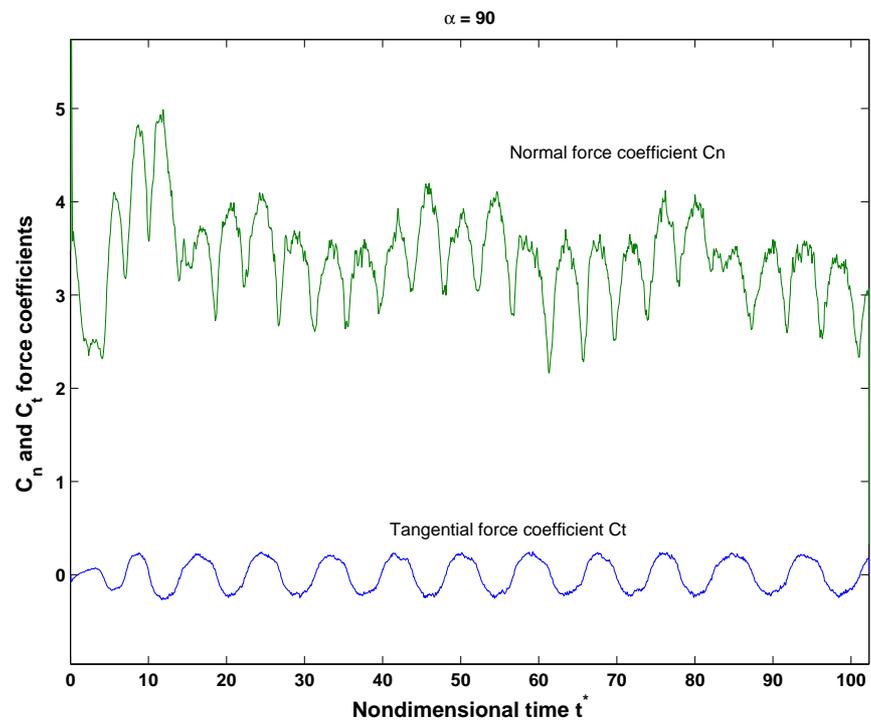


(b) $\alpha = 70^\circ$

Figure 3.26: Normal and tangential force coefficients time plot for the Spalart code with $\alpha = [60, 70]^\circ$

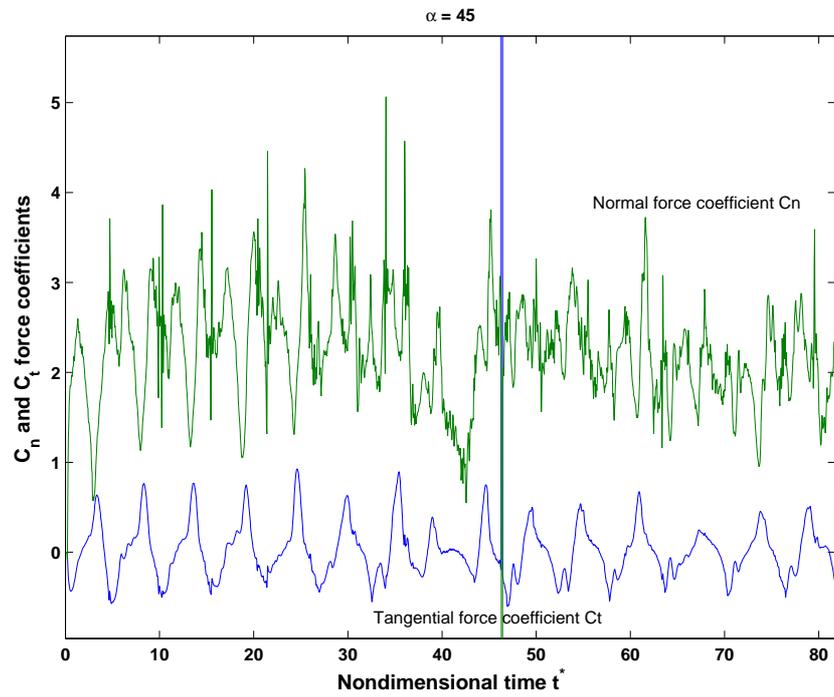


(a) $\alpha = 80^\circ$

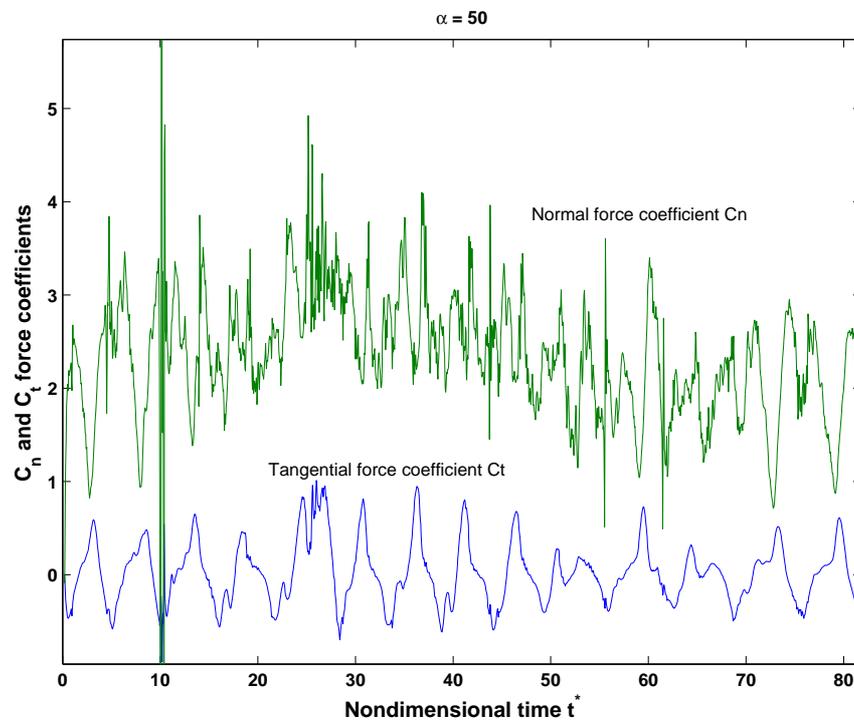


(b) $\alpha = 90^\circ$

Figure 3.27: Normal and tangential force coefficients time plot for the Spalart code with $\alpha = [80, 90]^\circ$

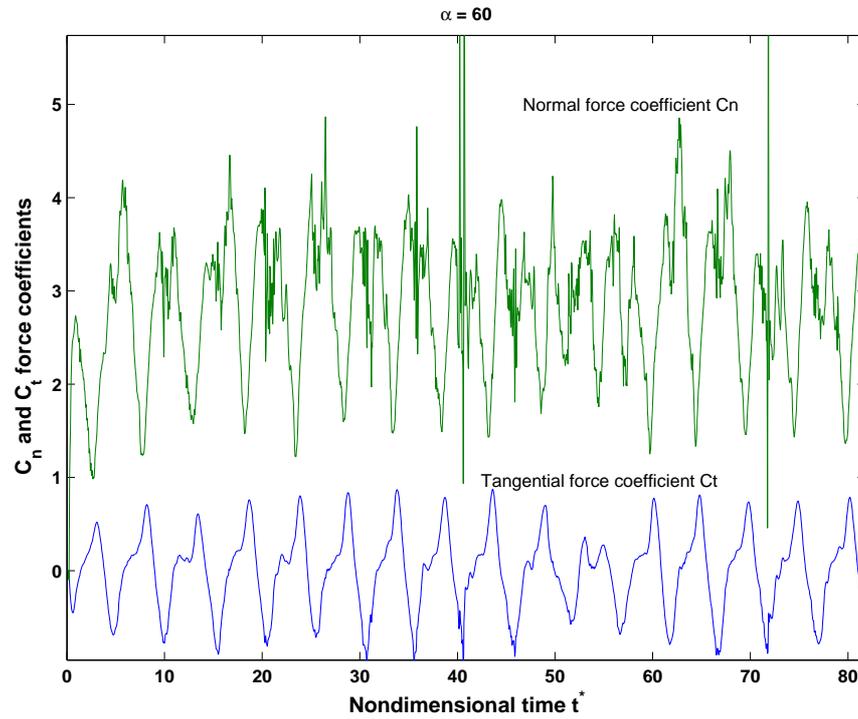


(a) $\alpha = 45^\circ$

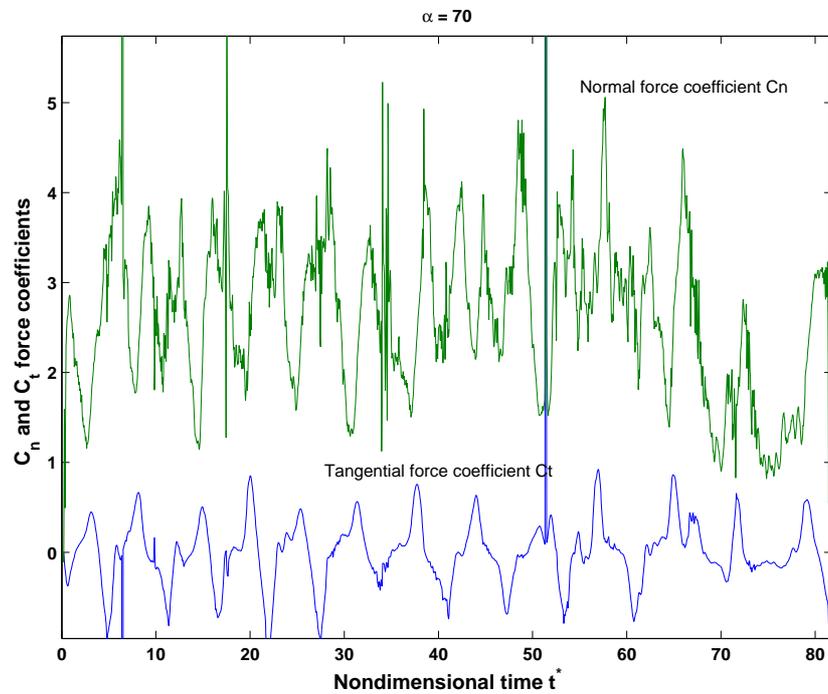


(b) $\alpha = 50^\circ$

Figure 3.28: Normal and tangential force coefficients time plot for the C code with $\alpha = [45, 50]^\circ$

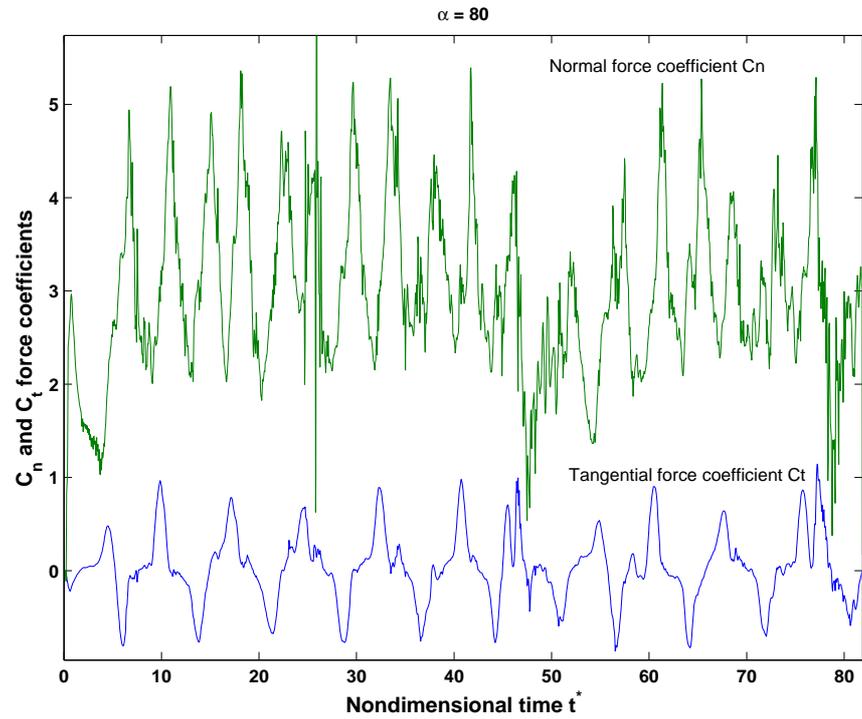


(a) $\alpha = 60^\circ$

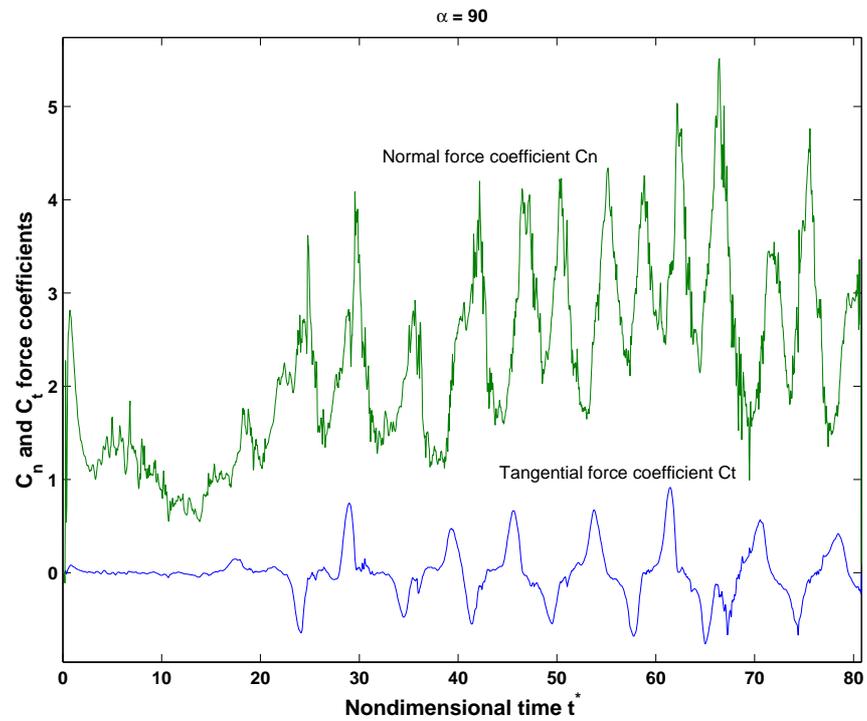


(b) $\alpha = 70^\circ$

Figure 3.29: Normal and tangential force coefficients time plot for the C code with $\alpha = [60, 70]^\circ$



(a) $\alpha = 80^\circ$



(b) $\alpha = 90^\circ$

Figure 3.30: Normal and tangential force coefficients time plot for the C code with $\alpha = [80, 90]^\circ$

3.2.4 Moving plate simulations

In order to assess the accuracy and feasibility of the rotating rigid plate case for the two codes, different cases were tested such as a sudden change of angle from 0 to 30 degrees incidence, reproducing simulations from Ham [23], in section 3.2.4.1 a rotating ellipse in a parallel flow using simulations from Lugt and Ohring [35], and in section 3.2.4.2 the vertical oscillations of a cylinder in a parallel flow as in Blevins [7].

In the first case, the Spalart code appeared to be unable to provide a correct solution with a 30 degrees incidence flow. This is because vortices near the wall generate noise in the velocity field which tend to scatter them. With higher incidence the separation is triggered by the high velocity induced at the edges of the ellipse, but at this incidence angle the noise is predominant and tends to scatter the blob vortices away from the wall along the plate, whereas separation should occur farther along the wall. That is to say the pure vortex method only predicts separation induced by an adverse pressure gradient. To avoid this effect, Spalart [57] used a separate method to predict the point of separation, and then intervenes into the dynamic of the vortices to delay their separation. An alternative method suggested by Professor Sergei Chernyshenko from the University of Southampton has been to simply use a Tikhonov regularization to solve the boundary condition equation 2.30. Results comparable to those of Spalart [57] have been obtained by A.Bouferrouk for a flat plate at low incidence. However, since both methods involve the use of ad hoc parameters, they were not implemented, as the current project is not related to the plate at low incidences.

In the first two cases, the C code was unable to provide a solution without additional modifications, the discrete shedding from both tips being inappropriate in both cases.

Now for the last case, Blevins [7] has implemented a slightly modified version of the Spalart code to account for the vertical oscillations of a body; similar modifications were applied to the Spalart code. Concerning the C code, because it has been designed with a flat plate as a model it cannot be applied here directly and requires extensive modifications. Nonetheless, Sarpkaya has successfully developed a method based on potential flow and boundary-layer interaction, discretization of the shear layer, and blob vortices circulation dissipation to emulate the characteristics of a flow with cylinder oscillations transverse to the freeflow (see reference [51]).

Overall, these tests have shown that it is difficult to apply the discrete vorticity methods –or Sarpkaya-like simulation– to these moving body cases, and almost impossible for a rotating plate without extensive modification of the vorticity introduction.

3.2.4.1 Rotating ellipse in a parallel freestream flow

The Spalart and Sarpkaya-like simulations will be compared with numerical simulations from Lugt and Ohring [26], who used a finite difference scheme to solve the 2D Navier-Stokes equations using a $\psi - \omega$ formulation. Their numerical analysis was carried out using DuFort-Frankel and Hockney techniques. The grid was 96×97 , with an average distance from the plate wall to the far grid boundary of about $26a$ with a half the chord value. The simulations were applied to a constantly rotating rigid plate placed in a uniform freestream flow. They used an ellipse with a length to thickness ratio of 10. To characterize the flow they used the Rossby number $Ro = U_\infty / (a\Omega)$ and the Reynolds number $Re = 2aU_\infty / \nu$ with U_∞ the norm of the freestream velocity. They chose $Ro = 2$, and $Re = 200$. Due to the lack of experimental results, the results were compared with theirs using streamlines, force and moment plots in order to have a qualitative reference. In a previous article [34], Lugt and Haussling tested their simulation for a stationary ellipse with a length to thickness ratio of 10 at a 45 degrees incidence in a flow with a Reynolds number of 200, and compared it to results from Honji [27]. It showed good agreement regarding the flow geometry near the body surface, and appropriate decay of the central vorticity of the initial vortex shed.

Only the Spalart code was tested, because the Sarpkaya-like code is not valid for $\theta < 45^\circ$ and for large values $\dot{\theta}$. More specifically, the mechanism of introduction of discrete vorticity is not proper for flows with $\alpha < 20$ degrees of incidence, as the interpretation of the Kutta condition used for the discrete vorticity introduction is no longer valid. The vorticity thus needs to be shed through other mechanisms than those described in section 2.3.4.1. Other difficulties arise as the Sarpkaya-like code is unable to cope with large angular velocities, and becomes unstable before $\alpha = 45^\circ$, mainly due to the resolution requirement from equation 2.37. Indeed, the difficulty comes from the close vicinity of the previously shed blob vortices that create local disturbances which the solver has difficulties coping with. The problem is less acute for an oscillating rigid plate provided one limit the rotation angle and velocity. That is, as the oscillation amplitude have a zero mean, the incidence angle stays on average over time at a value appropriate

for the Sarpkaya-like code if one use the incidence angle range used in section 3.2.3. In that case, one can put $\theta(t)$ on the form $\theta = \alpha + \theta_A \sin(\varpi_\theta t)$ with θ_A the oscillation amplitude, and ϖ_θ (rad/s) the oscillations frequency.

To implement the viscous effect for the Spalart code, a simple random walk scheme was used. The method implementation is inherently linked to the viscous splitting algorithm. It is based on splitting the momentum equation 2.9 into two parts:

$$\frac{\partial \omega}{\partial t} = -\vec{u} \cdot \nabla \omega, \quad (3.8)$$

$$\frac{\partial \omega}{\partial t} = \nu \nabla^2 \omega. \quad (3.9)$$

Note that equation 3.8 is actually the Euler equation and equation 3.9 is the viscous diffusion equation. The two equations are then solved consecutively rather than simultaneously like in equation 2.9, with the vorticity established after the convection step (eq. 3.8) used as initial condition for the diffusion equation (eq. 3.9). The solution to equation 3.8 is obtained with the inviscid Spalart code. In turn equation 3.9 can be solved by adding a random walk drawn from a Gaussian distribution with zero mean and variance $2\Delta t/Re$ (where Δt is the time-step of the simulation) to the position of each vortex at every time step (cf Wax (1954) [63]). Statistically, the random walk is equivalent to the viscous diffusion of equation 3.9. It was introduced by Chorin [9] for the simulation of slightly viscous flows by vortex methods. Koutsoumakos [13] provides a proof of the convergence of the method. There is also the usual convergence requirement that the mean spacing between vortices is much smaller than the variance $2\Delta t/Re$. Therefore, the main disadvantage is that it requires a large number of vortices to converge to the analytic diffusion solution. Obviously, a smaller number of vortices introduces noise in the velocity field. Some equivalence can be drawn to the method as the Spalart code with simulation parameters identical to those in section 3.2.3 would be equivalent to a random-walk with $Re \simeq 3000$.

For the non-dimensionalization, Lugt and Ohring used the ellipse half length a . Therefore for convenience, the force and moment coefficients will be defined specifically in

this section by:

$$C_d = \frac{\text{Drag}}{\frac{1}{2}\rho a U_\infty^2}, \quad (3.10)$$

$$C_l = \frac{\text{Lift}}{\frac{1}{2}\rho a U_\infty^2}, \quad (3.11)$$

$$C_m = \frac{\text{Moment}}{\frac{1}{2}\rho a^2 U_\infty^2}, \quad (3.12)$$

with C_d the drag coefficient, C_l the lift coefficient and C_m the moment coefficient. The non-dimensionalization convention for time was also changed by using $t^* = t(U/a)$. Results were obtained for the plate at fixed angle in conditions similar (in terms of Re), using a small flow comparison with experimental streakline from Honji [27] in figure 3.31, as well as Lugt and Ohring [34] results for plate at a fixed angle in figure 3.32. This part will be developed in a later paragraph. For the rotating plate case, results comparison is also made with Lugt and Ohring [35] in figure 3.33.

Note that the streamlines in figures 3.35, 3.36 and 3.37 were plotted in a frame fixed to the body regarding the translation but the body rotates relative to it. This kind of frame is useful as the streamlines are then equivalent to fictitious instantaneous streamlines observed in a wind tunnel for a rotating body. In all these plots, the freeflow is going from right to left. The relevant streamlines plots have been reproduced with permission from Lugt and Ohring in reference [35]. The scale of the streamline plot is provided in figure 3.34.

The Spalart simulations were done using a time step $\Delta t^* = 0.05$, $D_0 = a$, $N_b = 1200$, $Re = 200$, $Ro = 2$ for 340 time steps. The total number of blob vortices N used was of 6000, because as shown in section 3.2.2, N has little influence on the flow dynamics once it is greater than about twice the number of boundary points N_b . Furthermore, the simulations are run over a shorter time span, and there is less influence from the merging procedure to the nearby wake as the initial blob vortices move off the body. Note that the force and moment coefficients are filtered in the same manner as in section 3.2.3.

First, a comparison is given of the two codes for the viscous flow over a *stationary* plate (i.e. $Ro \rightarrow \infty$), the simulation was then ran for the case of an ellipse with a length to thickness ratio of 10 with $Re = 200$, using the same set of parameters mentioned above. Lugt and Haussling [34] made numerical simulations with the same Re , ellipse geometry, α and time-span using the same scheme as for the rotating ellipse.

A streamline plot is produced for qualitative comparison with the experimental streakline result from Honji [27] at a given time step in figure 3.31. On the other hand, the force and moment coefficients were compared with those of Lugt and Haussling [34] in figure 3.32.

From figure 3.31, one can see that the general structure of the flow is well rendered with one eddy already shed at the end of the plot, and another eddy being shed near the plate at about the same distance as in the Honji experiment. The nearby wake is more difficult to assess because of the experimental picture quality near the body. One can also see some non-smooth behavior in the Spalart streamline plot due to the small core radius of the blob vortices. Otherwise, the streamlines curvature seem generally higher than for Honji experimental streaklines. Although direct comparisons between streamlines and streaklines are difficult in unsteady flow cases, it would be not be surprising as it indicates a stronger vorticity for the eddies, which would be in phase with the results found in section 3.2.3.

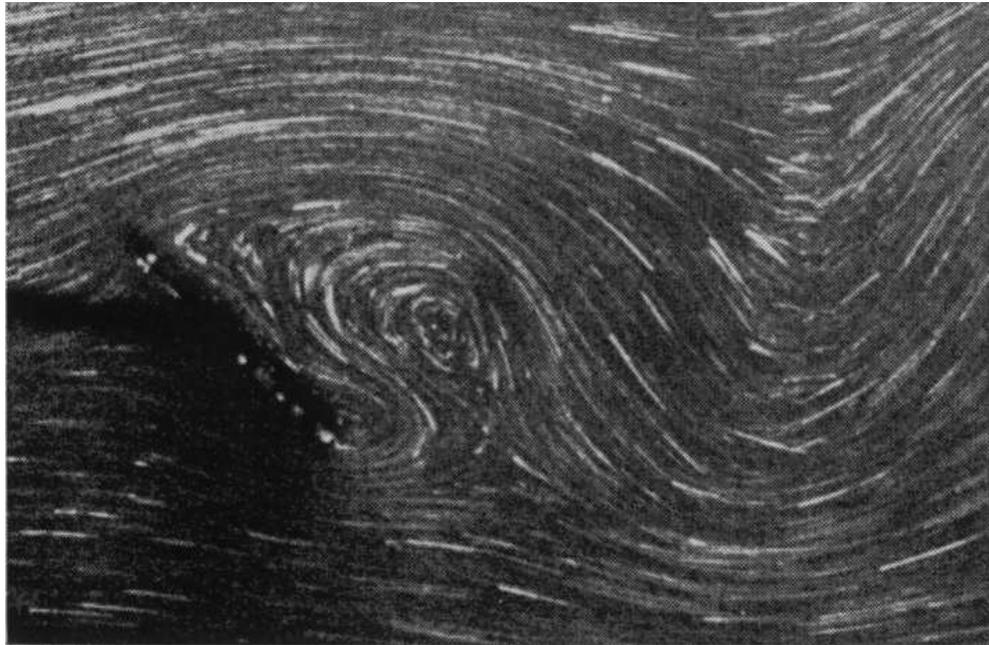
The forces and moment histories in figure 3.32 show that the code tends to overestimate these quantities compared to the Lugt simulation. As for the inviscid case (section 3.2.3), this is thought to originate from an overestimation of the vorticity in the center of the eddies, owing mainly to the absence of $3D$ vorticity dissipation, due to the intrinsic $2D$ nature of the simulation. This also confirms the previous streamlines and experimental streaklines comparison in the above paragraph. Another finding from the C_m plot is that the first vortex shedding occurs earlier with the Spalart code, and that both coefficient evolution periods are about equal. The previous experience with the constant incidence case leads to think that the simulation has a Strouhal close to the experimental value at early timesteps and then stabilizes on another incorrect frequency at later times.

Overall, the simulation shows that it is able to reproduce successfully the broad features of the flow and the viscous vortex street dynamics. However in light of the aerodynamic forces and moment coefficients, one see that the agreement is only qualitative because of too strong shed vorticity in the absence of the $3D$ vorticity dissipation mechanism. Finally, there remains the question of the convergence of the long-term simulation, which is difficult to address here, owing to the short timespan used.

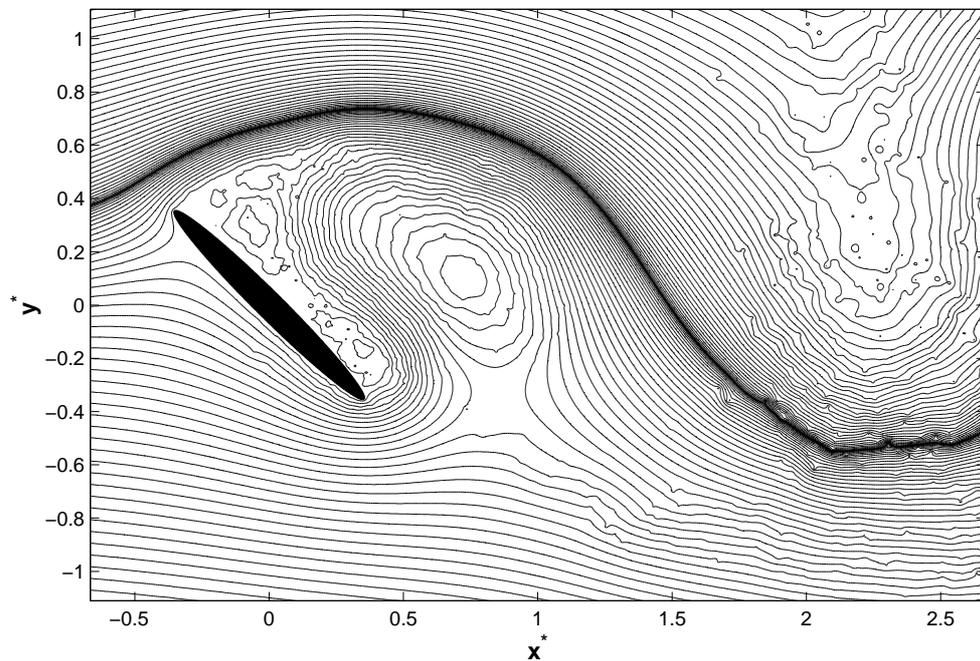
Turning toward the rotating case, figure 3.33 shows that the forces and moment coefficients are in good agreement with Lugt and Ohring results ([35]). In order to compute the time-averaged values, data were taken from $\theta_0 = \pi$. Similarly to the stationary ellipse case, the code overestimates the time-averaged forces coefficients, compared to the Lugt simulation, albeit to a lesser extent than in the stationary ellipse case. The differences are of 14% for the aerodynamic forces coefficient, and 11% for the time-averaged moment. Otherwise, the force and moment coefficients from both cases exhibit a similar period and are found to be in better agreement than in the stationary ellipse case. However since Lugt and Ohring also did their study in $2D$ their results are also expected to overestimate the shed vorticity, neglecting the $3D$ effects as already mentioned in section 3.2.3. It can be concluded that the Spalart simulation behaves consistently with other $2D$ methods.

The streamlines plots (whose scale is provided in figure 3.34) in figures 3.35, 3.36, and 3.37 reveal that although the Spalart code captures the overall flow features, it again yields the non-smooth behaviour mentioned above when discussing the stationary case. This is also thought to originate from the relatively small blob vortex core radius, as well as the vortex merging process and the random walk method. The latter has also been found in the random-walk method used by Clarke [11] for the stationary cylinder, indicating an insufficient number of vortices. However, because the random-walk method has a slow convergence, the computing cost of the increased number of vortices needed to produce a smoothed flow structure is expected to be prohibitive.

It can thus be deduced from the rotating and non rotating cases that the Spalart code, in the absence of boundary layer separation control, is well suited to these two flows, and more generally for massively separated flow. It is not as well suited to the cleaner problem of a plate at $\alpha = 90^\circ$ where the flow can be modeled by discrete introduction of vorticity. However given the lack of relevant experimental results, one can only conclude that the Spalart simulation behaves consistently in $2D$ when there is a rotation. More particularly, the basic flow features and dynamics are preserved throughout the computation, but the force and moment coefficients are overestimated, presumably because of the absence of vorticity diffusion inherent to the $3D$ nature of the real flow.

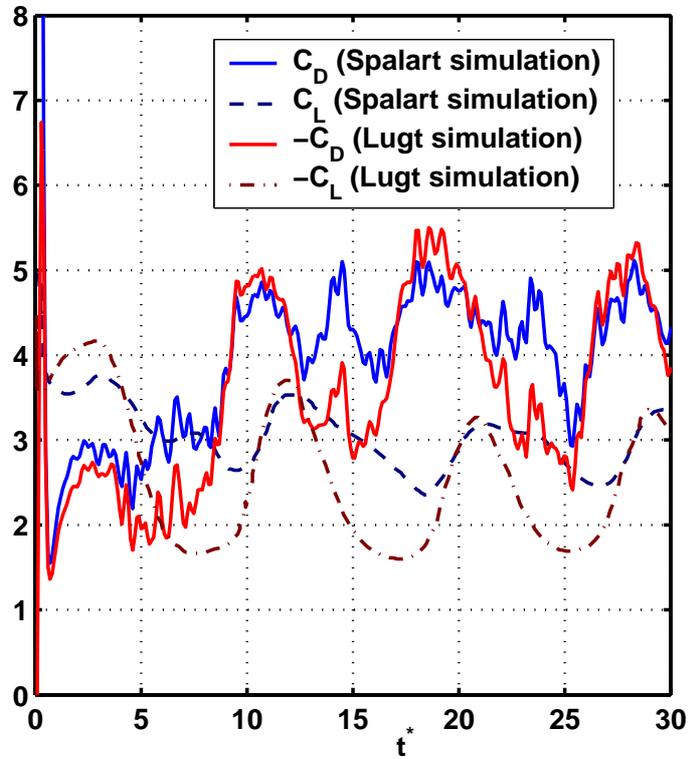


(a) Honji [27] experiments

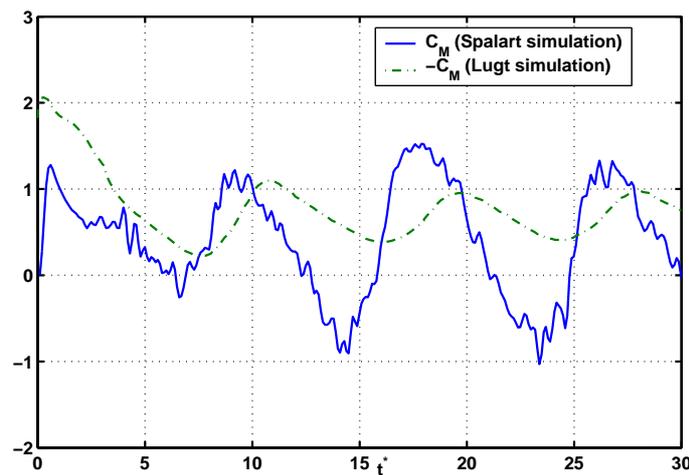


(b) Spalart code simulation

Figure 3.31: Streaklines and streamlines sequence for an impulsively started flat plate at $Re = 200$, $\alpha = 45^\circ$. The streamfunction values in (b) range from $[-1.13, 1.13] \times (2aU_\infty)$ with a concentration around 0 at $t^* = 6.78$

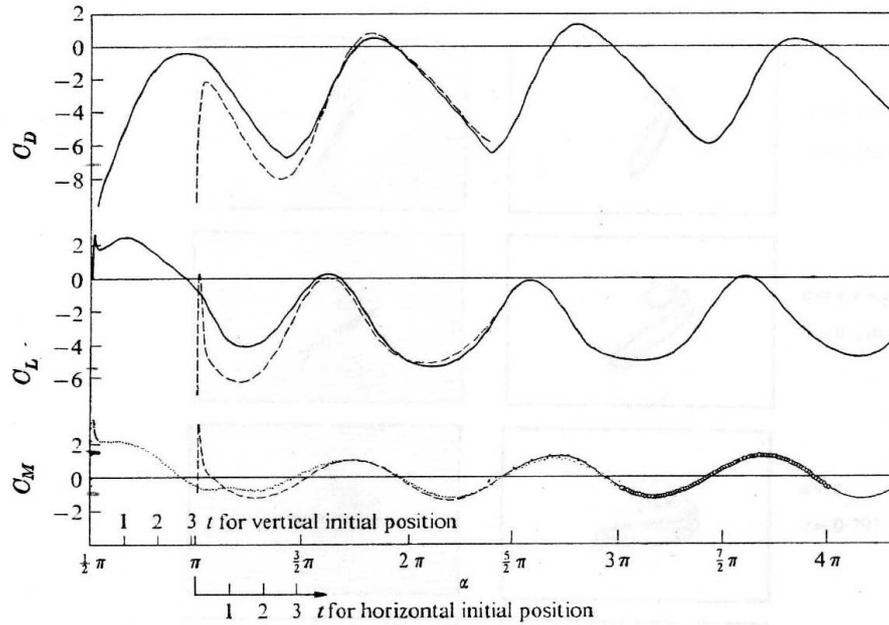


(a) Drag and lift coefficients

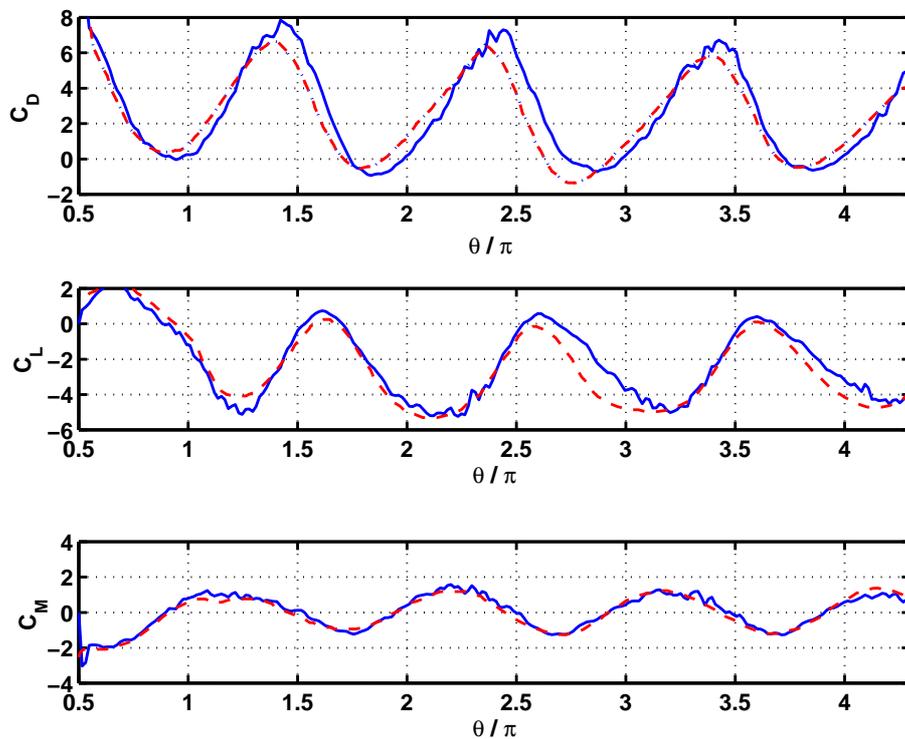


(b) Moment coefficient

Figure 3.32: Force and moment coefficients for a stationary ellipse in a parallel freestream flow at 45 degrees with $Re = 200$

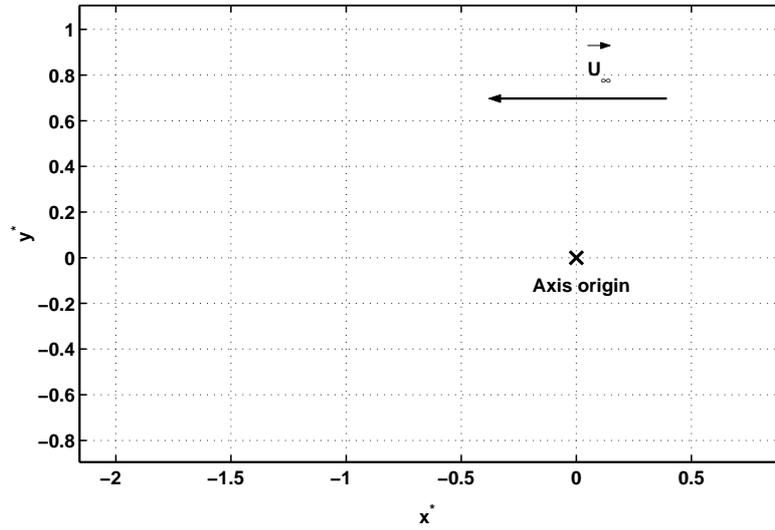


(a) Coefficients from Lugt and Ohring [35]

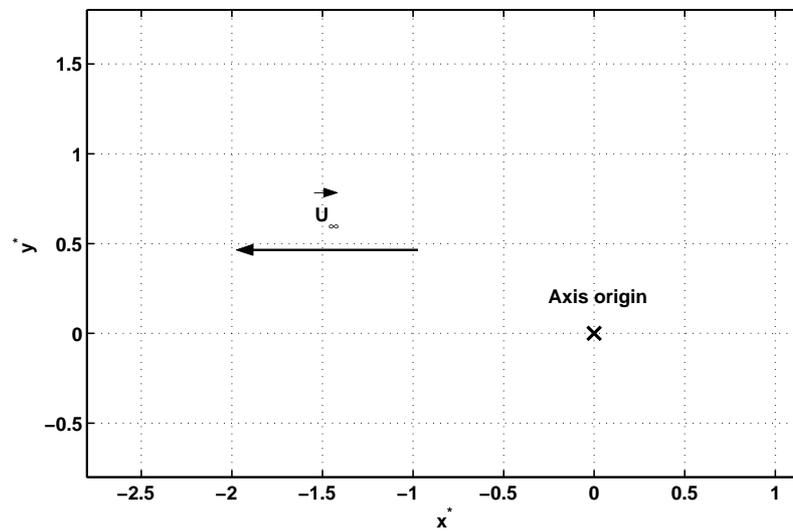


(b) Coefficients from Spalart code

Figure 3.33: Force and moment coefficients for a rotating ellipse in a parallel freestream flow (for subfigure 3.33(b) the dotted line represents the forces and moment coefficients from Lugt and Ohring).

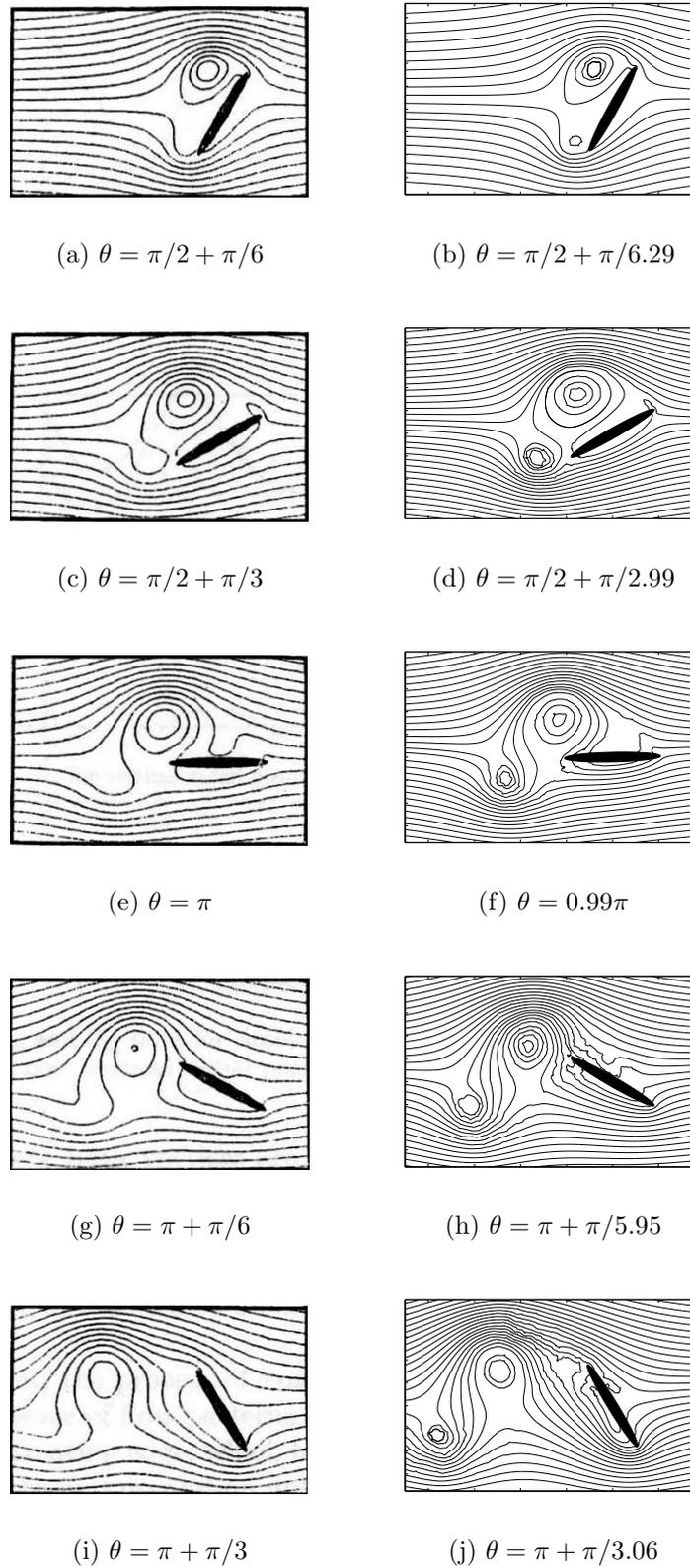


(a) θ between $\pi/2 + \pi/6$ and $\pi + \pi/2$



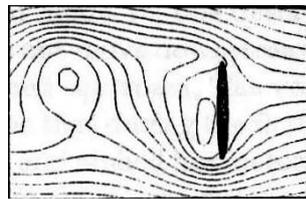
(b) θ between $\pi + \pi(2/3)$ and $\pi + \pi/3$

Figure 3.34: Plot scale for the streamline plots in figure 3.35, 3.36 and 3.37.

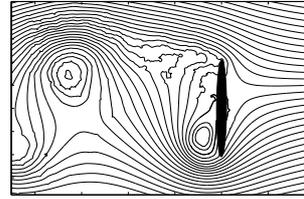


Lugt and Ohring computations [35] Spalart code

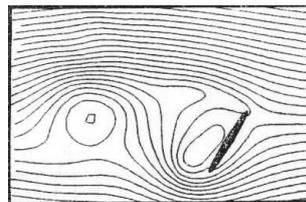
Figure 3.35: See figure 3.37 for the caption and figure 3.34(a) for the scale.



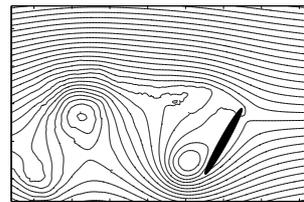
(a) $\theta = \pi + \pi/2$



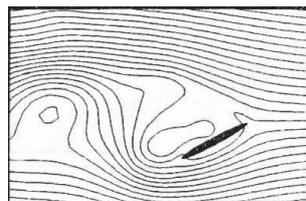
(b) $\theta = \pi + \pi/1.99$



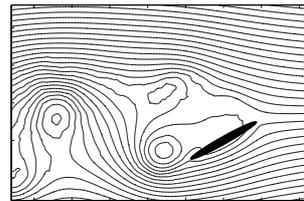
(c) $\theta = \pi + \pi(2/3)$



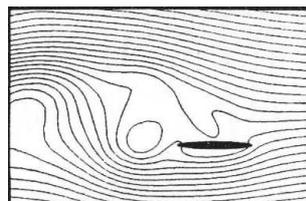
(d) $\theta = \pi + \pi(2/3.03)$



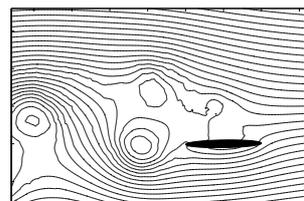
(e) $\theta = \pi + \pi(5/6)$



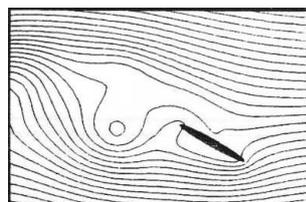
(f) $\theta = \pi + \pi(5/5.98)$



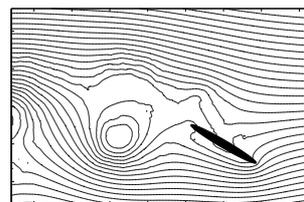
(g) $\theta = 2\pi$



(h) $\theta = 1.99\pi$



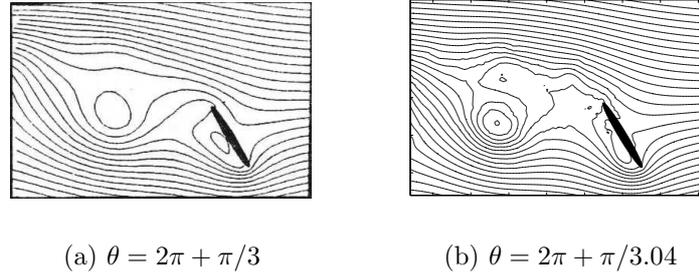
(i) $\theta = 2\pi + \pi/6$



(j) $\theta = 2\pi + \pi/5.89$

Lugt and Ohring computations [35] Spalart code

Figure 3.36: See figure 3.37 for the caption and figure 3.34(a) for the scale.



Lugt and Ohring computations [35] Spalart code

Figure 3.37: Sequence of streamlines around a rotating ellipse in a parallel flow for $Re = 200$, $Ro = 2$, and an initial incidence of $\pi/2$. In contrast to every other sections, the freeflow is flowing from right to left. See figure 3.34 for the scale.

3.2.4.2 Transversely oscillating cylinder

Blevins [7] applied the Spalart code to the flow over a rigid circular cylinder allowed to move transversely to a parallel freestream. He compared results to experiments by Feng [18]. He found that the method was able to predict a lock-in, albeit in a narrower band than for experimental results, and that the vertical oscillation peak clearly exhibits an increase near resonance. The too-narrow frequency-band problem is apparently related to the phase between the structural motion and fluid forces being not sufficiently accurate.

The governing equation of motion for the cylinder section is:

$$m \frac{d^2 Y}{dt^2} + 2m \xi \varpi_n \frac{dY}{dt} + kY = F_Y, \quad (3.13)$$

where $Y(t)$ is the vertical displacement in the stationary frame (\vec{X}, \vec{Y}) , m is the structural mass of the section, ξ is the structural damping with k the spring constant, and F_Y is the aerodynamic force in the \vec{Y} direction. He used as parameters a circular cylinder of radius R , a damping ξ of 0.02, a reduced damping of $2m(2\pi\xi)/(\rho D^2)$ (ρ being the fluid density and D the cylinder diameter). The Reynolds number, based on cylinder diameter, was 20,000, the time step $\Delta t = 0.05$, $N_b = 50$ boundary vortices, global number of vortices $N = 350$ and σ one half the average distance between control points (cf section 2.3.2), i.e. $\sigma \simeq 0.06 R$. The cylinder is kept stationary for 1000 time steps, then allowed to move vertically for 8192 time steps. He then compared it to

experimental data from Feng [18].

To illustrate the ability and the properties of the vortex method for such a case, the exact same conditions were used except for $N_b = 150$, $N = 1050$ and $\sigma = \delta_0/4 = 0.01 R$. In the result table 3.2, f_s refers to the peak of F_Y spectrum, and f_n refers to the spring damped cylinder natural frequency of vibration.

$U/(Df_n)$	2	3	4	4.5	5	5.5	6	7	8
Y_{max}/D Feng	N/A	N/A	0.0051	0.0062	0.021	0.124	0.172	0.012	N/A
Y_{max}/D Blevins	0.0097	0.041	0.2	0.21	0.19	0.23	0.3	0.22	0.27
Y_{max}/D present	0.011	0.046	0.1325	0.2508	0.3656	0.409	0.3574	0.3454	0.4725
f_s/f_n Feng	N/A	N/A	0.8	0.92	0.99	1.0	1.0	1.38	N/A
f_s/f_n Blevins	0.4	0.67	1.0	1.0	1.0	0.82	1.0	1.4	1.6
f_s/f_n present	0.44	0.69	1.07	1.08	1.123	1.1548	1.4	1.68	1.83

Table 3.2: Vortex induced vertical oscillation of a cylinder

The results are very similar to those obtained by Blevins. The vortex method successfully predicts a lock-in frequency, however it occurs at a lower frequency than for Feng experiments. Again, the lock-in band is much narrower, as can be seen from the maximum amplitude Y_{max} . The differences in amplitude and in lock-in frequency compared with Blevins simulations can be explained by the larger number of boundary blob vortices, and the smaller core radius used here. Again, as already noted in case of the plate at fixed incidence in section 3.2.3, it may be a consequence of the small core radius as it implies an insufficient blob vortices overlap causing a too-low Strouhal. Additionally, it can be conjectured that similarly to section 3.2.3 it also causes stronger vorticity for the shed eddies. As the flow dynamics remains broadly the same this means that the forces are again overestimated, which in turn explains the higher cylinder amplitude. It is however possible to obtain similar results to Blevins by raising the value of σ and N . Since a correct set of parameters can only be found empirically, a long time must be dedicated to finding such a set. Finally it seems the same problems as for Blevins

(described earlier paragraph in this section) arise, which is not surprising as the code based on Spalart differs mainly by the simulation parameters used. The results do not significantly differ otherwise.

Again, the vortex method has shown that it was able to simulate the qualitative behavior of an interacting fluid-structure system. Good quantitative results are also possible, but only after a careful and time-consuming search of the numerical parameters, and possibly very expensive calculation. Therefore, in the case of angular, rather than transverse, oscillations, one may assume that as the flow is likely to be massively separated (as in section 3.2.4.1), and since the method reproduces the coupled mechanical/flow system frequency characteristics for the cylinders, the vortex scheme will also be able to simulate the relevant dynamics of the target flow.

3.3 Angular oscillation of a plate

In this part, the behavior is studied for a spring damped rigid plate hinged about its centerpoint, and placed in a uniform transverse flow. The setup of the plate is shown in figure 3.38.

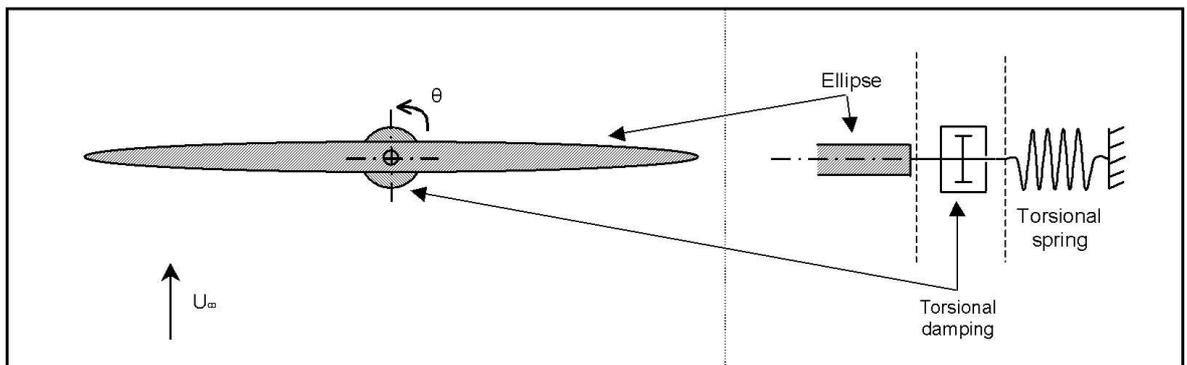


Figure 3.38: Illustration of the system ellipse and fluid

The added features of the code are characterized by simulating the oscillations of a plate with no freeflow, in section 3.3.1. Then, the influence of the different mechanical parameters will be investigated for a spring-damped plate placed the freeflow, in section 3.3.2. In chapter 4, an identification of a simplified flow model will be implemented in order to use a model-based design. Afterwards, in chapter 5, an assessment will be made

about the efficiency of the control of the system composed of this oscillating plate/flow system using results in this section.

For the plate model, the equation of motion is

$$J\ddot{\theta} = -\mu\dot{\theta} - k\theta + M, \quad (3.14)$$

with θ the angular position of the plate *from the rest position of the torque spring* (that is $\theta \neq \alpha$), J the angular inertia, μ the damping at the center of rotation, k the constant of the torsional spring, M the external torque, i.e. the resulting aerodynamic moment (which has units of torque here), finally the overdot indicates a time derivative.

In the most used form, it is written

$$J\ddot{\theta} + \mu\dot{\theta} + k\theta = M, \quad (3.15)$$

One should recall the properties of this kind of system are determined by the natural frequency $\varpi_n = \sqrt{k/J}$ and the reduced damping $\xi = (\mu/2k) \varpi_n = (\mu/2k) \sqrt{k/J}$ which are used to assess the properties of the system transfer function:

$$L(s) = \frac{C_G}{1 + 2\xi\left(\frac{s}{\varpi_n}\right) + \left(\frac{s}{\varpi_n}\right)^2}. \quad (3.16)$$

where $C_G = 1/k$ is the steady-state gain of the function.

One then has as options:

1. $\xi = 0$: no damping; the roots are purely imaginary, the system behaves as a pure oscillator of frequency ϖ_n .
2. $0 < \xi < \frac{\sqrt{2}}{2}$: a resonance frequency exists; there are two complex roots.
3. $\frac{\sqrt{2}}{2} \leq \xi < 1$: two complex roots but no resonance frequency; the system is then a low pass filter of order two.
4. $\xi > 1$: two real roots, the system can then be reduced to two low-pass filter of order one in series.

For the current work, the cases considered are without external damping, or with little natural damping and weak spring. Therefore, during this study, only options 1. and 2.

are of concern.

In order to define a parameter space for the study of the angular oscillations of the ellipse, the equation of motion has to be put into non dimensional form. Traditionally, for the transverse oscillations of a cylinder, authors have used the natural frequency of the mechanical system in the belief that the mechanical frequency governs the response as well as the cylinder diameter. However, this is not appropriate for certain limiting cases, for instance if $k = 0$. Thus following recommendations by Shiels [56], it was chosen to use instead the magnitude of the flow freestream velocity $\overrightarrow{U_\infty}$, and the ellipse chord $2a$ to form the nondimensional variables, which are indicated by the * superscript:

$$t^* = t \frac{|\overrightarrow{U_\infty}|}{L}, \quad (3.17)$$

$$\theta^* = \frac{\theta}{\pi}, \quad (3.18)$$

$$C_m = M^* = \frac{M}{\frac{1}{2} \rho L^2 U_\infty^2}, \quad (3.19)$$

where M is the dimensional aerodynamic moment, L the chord of the ellipse, and ρ the fluid density. Note that with the definition of θ , the incidence angle α is equal to $\alpha = \alpha_0 + \theta$ with α_0 the ellipse incidence angle at $t^* = 0$. The equation of motion 3.14, thus becomes in the inertial frame:

$$J^* \frac{d^2 \theta^*}{dt^{*2}} + \mu^* \frac{d\theta^*}{dt^*} + k^* \theta^* = C_m, \quad (3.20)$$

$$J^* = J \frac{2\pi}{\rho L^4}, \mu^* = \mu \frac{2\pi}{\rho L^3 U_\infty}, k^* = k \frac{2\pi}{\rho L^2 U_\infty^2}. \quad (3.21)$$

For an ellipse, the angular inertia (or moment of inertia) J is equal to:

$$J = \rho_b \frac{\pi}{4} a b (a^2 + b^2), \quad (3.22)$$

where ρ_b is the body density, and a and b are respectively the major and the minor axes of the ellipse.

It is sometimes more practical when $J^* \neq 0$ and $k^* \neq 0$ to rewrite equation 3.20 in a form that highlights the mechanical properties of the spring damped plate:

$$J^* \frac{d^2 \theta^*}{dt^{*2}} + 2J^* \xi^* \varpi_n^* \frac{d\theta^*}{dt^*} + J^* \varpi_n^{*2} \theta^* = C_m, \quad (3.23)$$

$$\varpi_n^* = \sqrt{\frac{k^*}{J^*}}, \xi^* = \frac{\mu^*}{2\sqrt{J^* k^*}}, \quad (3.24)$$

$$\varpi_n^* = \varpi_n \frac{L}{U_\infty}, \xi^* = \xi. \quad (3.25)$$

The aerodynamic moment coefficient $C_m(t)$ is computed in the flow simulation and takes into account the effects of the plate motion. The plate position is updated using equation 3.23 using the C_m between two flow simulation timesteps (see also section 2.4).

Note that from equation 3.23, it appears that an important parameter to assess the fluid influence on the oscillation is J^* . It helps to relate the body moment of inertia to the fluid moment of inertia due to the fluid added mass. For bodies of significant volume such as circular cylinders the added mass is taken as proportional to the displaced fluid mass but for thin bodies such as plates it is proportional to (actually approximately equal to) the mass of a circular cylinder of fluid having the same diameter as the plate. Thus for the thin ellipse case, the parameter would be:

$$\frac{J}{\frac{1}{32}\rho\pi L^4} = 16\pi^2 J^* = \frac{\rho_b b(a^2 + b^2)}{2\rho a^3}, \quad (3.26)$$

considering $L = 2a$ in our case.

As a step toward implementing the angular plate oscillations in the flow with the C code, experiments were done by first considering a plate in a uniform flow with prescribed forced oscillations. It has proven very difficult to obtain a stable simulation for this case. The problem seems to come from the equations related to the determination of the nascent blob vortices strength and position. When the plate pivots back and forth, the edges approach closely the blob vortices generated during the previous timesteps, thus creating local singularities in the velocity field near the plate tips, and undermining the nascent vortices position determination. In turn, this affects the nascent vortices strength determination, which then affects the nascent vortices position, and so forth. A filter was then applied on the nascent vortex strength in order to limit the fluctuations in strength, and in other trials a filter on the nascent vortices position. However, it turns out that it introduces far too much numerical damping, lowering the Strouhal number for a plate at steady angle by about 20%. Indeed, the damping slows down the nascent blob vortices evolution, which means the shedding eddies are fed for longer times with vorticity from the shear layer (see section 2.3.4).

On the other hand, despite a lower Strouhal number error compared to the C code with filtered nascent vortex strength, the Spalart code has proven to behave well in terms of the qualitative flow dynamics in the case of moving bodies. Furthermore, the

aerodynamic force coefficients obtained with the C code with filtered strength are also noisy for the plate case at fixed angle (presented below in section 3.2.3). This has prompted to use, for the remainder of the chapter, instead of the C code (or Sarpkaya-like code) with discrete vorticity introduction a translation of the Spalart code in C. The results are equivalent between the C and the Fortran simulations, except that the C simulation runs more slowly, probably due to the compiler (visual C/C++).

Finally in the streamline plotting procedure, note that this is the same fixed body reference frame as in section 3.2.4.1, it is thus normal to see streamlines crossing the body as they denote the body motion.

3.3.1 Oscillations of a spring damped plate with no freeflow

This test is done in inviscid flow at $\alpha = 90^\circ$ with a reduced damping $\xi^* = 0$, a time step $\Delta t^* = 0.05$, a body density $\rho_b = 1200 \text{ kg.m}^{-3}$, a fluid density $\rho = 1.2 \text{ kg.m}^{-3}$, and a fluid at rest ($U_\infty = |\vec{U}_\infty| = 0$). Otherwise, the same parameters were used as for the simulation in section 3.2.3.

The natural frequencies chosen were $f_n^* = [0.5, 0.25, 0.125]$ with $f_n^* = \varpi_n^*/(2\pi)$ using for the non-dimensionalization of the variables a virtual $U_\infty = 0.72$. This U_∞ value was chosen so as to better relate results in this section to results in section 3.2.3 and 3.3.2. Therefore, the f_n^* correspond to $f_n^* = [3.4, 1.7, 0.86] * S$ the experimental Strouhal number for the plate at $\alpha = 90^\circ$, as in section 3.2.3. S is defined here by $S = f(2a)/U_\infty$ using the virtual U_∞ .

The equation of motion 3.23 of the ellipse would lead without taking into account the flow influence ($C_m = 0$) to the solution:

$$\theta = \theta_0 \cos(\varpi_n t), \quad (3.27)$$

$$\dot{\theta} = -\theta_0 \varpi_n \sin(\varpi_n t), \quad (3.28)$$

$$\ddot{\theta} = -\theta_0 \varpi_n^2 \cos(\varpi_n t). \quad (3.29)$$

In these trials, the initial ellipse incidence angle α_0 was set at 90° , with an offset to account for an initial spring angle θ_0 . Therefore, the initial incidence angle of the plate is $\alpha = \alpha_0 + \theta_0$. At $t^* = 0$, the spring damped ellipse is released in the quiescent fluid.

Angle evolution for $\theta_0 = -1^\circ$ and $\theta_0 = 1^\circ$ are presented for $f_n^* = 1/2$ in figures 3.40 and 3.41 respectively. The angle evolution results is also presented for $\theta_0 = -1^\circ$ and $\theta_0 = 1^\circ$ for $f_n^* = 1/2$ and $f_n^* = 1/8$ in figures 3.42 and 3.43 respectively. Cases are also presented with $\theta_0 = -2.5^\circ$ and $f_n^* = 1/[2, 0.8]$ in figure 3.44.

Summarily, the angle evolution does not change significantly when f_n^* or θ_0 varies, except of course for the θ oscillations frequency and amplitude. Thus every plot follows a similar pattern with the oscillations decreasing with time, due to the damping associated with the “added mass” inertia of the fluid. Note that with the frequency values used, lowering the timestep value up to $\Delta t^* = 0.04$ does not change significantly the results.

However, for $1/f_n^* = 8$, the simulation becomes unstable after the 1024th timestep (figures 3.42 and 3.43). The simulation also shows some remaining non negligible oscillations (about 20% of θ_0 after 400 timestep) for $\theta_0 = -2.5^\circ$ and $f_n^* = 2$. These oscillations are more or less randomly distributed as can be seen from figure 3.44(a). This instability is probably due to the absence of a freeflow in conjunction with the merging process.

The simulation being inviscid, there is no explicit viscous mechanism through which the vorticity can be dissipated, instead, there are other mechanisms intrinsic to the simulation. First, there is the blob vortex wall model (see section 2.3.5), which delete the blob vortices colliding with the wall and reinject their vorticity in the simulation through equation 2.31. In this case, the sum in equation 2.31 of vortex strength is no longer equal to zero but to the strength sum of the blob vortices erased. There is a second dissipation mechanism which is due to the numerical error due to the integration of equation 2.27 (see section 2.2.3). Finally, there another source of numerical errors coming from the merging process in the Spalart code (see section 2.3.8). Remark that the first two mechanisms are linked to the blob vortices motion. Therefore in the absence of freeflow, or when the body motion does not predominate over the numerical noise, the vorticity field is and remains very noisy. This is what happens when the plate does not rotate quickly enough.

When the plate is moving it is shedding blob vortices, and if the plate is oscillating quickly enough the blob vortices are either absorbed and their vorticity redistributed

to the boundary blob vortices (through the mechanism described in the above paragraph) or they move naturally with the plate motion. This is visible for $1/f_n^* = 2$ in the streamline plots in figures 3.45(a), 3.46(a), 3.45(c) and 3.46(c). At $t^* = 25$ (figures 3.45(a) and 3.46(a)) the vortices pattern along the body wall is clearly visible. At $t^* = 60$, this pattern remains although there are some strong vortices visible for example behind the ellipse, which implies a non-zero mean C_m . This explains why there is body motion at this time in figure 3.40 and 3.41.

On the other hand, if the motion is not quick enough, the blob vortices tends to stagnate and then, due to the merging process, concentrate into strong vortices far enough from the plate to not be influenced by the plate motion but close enough to affect the aerodynamic force evaluation, making the simulation blow up. This is visible for $1/f_n^* = 8$ in the streamlines plots in figure 3.45(c), 3.46(c), and 3.45(d). While at $t^* = 25$ (figures 3.45(b) and 3.46(b)), there have been few oscillations, and vortices have been shed in a symmetrical manner at both tips of the ellipse. At $t^* = 60$ (figure 3.45(d)) one can see a large eddy which has been formed at the lower ellipse edge.

Despite the numerical noise introduced in the vorticity field, the merging process remains necessary. This is mainly because the merging criterion D_0 does not allow a very fine control of the merging distance. Therefore, raising D_0 nullifies the merging process, so that it is no longer effective. Other simulations with $D_0 = 2a$ (instead of a) have shown that the blob vortices number is no longer kept under control and is increasing steadily with time, while not having much influence on the effective damping ξ_s^* introduced by the flow on the plate motion. With $D_0 = 1.5a$, the number of vortices stabilizes. In this case, with $f_n^* = 0.5$, $\xi^* = 0$. and $\theta_0 = -1^\circ$, the effective damping ξ_s^* and $f_{n,s}^*$ are equivalent (up to the third digit) to those found for $D_0 = a$ (see table 3.3).

However the main problem is not so much the merging distance but rather the physics of the flow. The trial with $D_0 = 2a$ has also shown that in the first angular oscillations period, the vortices tend to stay close to the plate wall while the motion does not differ significantly from the motion with the merging process (difference of 0.6% of the maximum of the plate angular velocity when $D_0 = 1a$). This shows that it is difficult to apply the blob vortices merging mechanism as it implies that one introduces some noise near the boundary. The merging may perhaps be enhanced by putting the emphasis on the blob vortex strength, but due to the lack of time it was not possible

to implement such a mechanism, remark though that the application in this paragraph is quite specific and not directly to the project.

An explicit vorticity diffusion mechanism such as a PSE mechanism would help to solve this problem by diffusing and controlling the vorticity in a more natural way. As a complement to the blob vortices merging mechanism, it could thus help to reduce the numerical noise introduced.

Remark that the solution at $t^* = 60$ is different for $\theta_0 = 1^\circ$ and $1/f_n^* = 8$ as shows figure 3.46(d). For this last case, there are two large eddies on both ellipse edges, plus an additional one at the upper edge on the plate right side. Proof of numerical noise is then also coming from this asymmetry in the flow simulations visible when comparing figures 3.45 and 3.46. In particular in the case where the simulation produce a stable solution for $1/f_n^* = 2$, if one compares figures 3.45(c) and 3.46(c), similar stagnating vortices are visible behind the plate but at a similar location, whereas there should have been a central symmetry with respect to the ellipse inertia center.

From other trials with varied f_n and θ_0 , it seems that the simulation stability is dependent upon f_n / θ_0 . That is to say, if a simulation is stable for a given f_n / θ_0 , then for different f_n and θ_0 the simulations will be stable if f_n / θ_0 remains constant. The f_n / θ_0 value can be regarded as a “sweeping frequency”, and simply illustrates how quickly the ellipse is sweeping the surface defined by the ellipse position between $-\theta_0$ and θ_0 .

Due to a lack of time, the varying parameters did not include the timestep. Nonetheless, it would be interesting to assess the Δt^* influence in the study. Even in the absence of data, it can be conjectured that Δt^* influence could be introduced by using the modified criterion $f_n \Delta t^* / \theta_0$ which corresponds to a “swept distance” parameter. In other words, the criterion corresponds to the ratio between the surface swept during one timestep compared to the surface defined by the ellipse position between $-\theta_0$ and θ_0 . It would be interesting to confirm this stability criterion influence as well as to see if it would be possible to extrapolate it to other vortex method simulations.

After looking at these criteria, also remark that the frequency at which the ellipse is oscillating is more important than the maximum angular velocity, as from equation 3.27 the maximum angular velocity is dependent on the product $\theta_0 f_n$.

Results in table 3.3 are for timesteps ranging from 1 to 1024. The results concern the evolution of the effective f_n^* and ξ^* of the $\theta(t)$ results produced by the coupled mechanical/flow system, they are noted respectively $f_{n,s}^*$ and ξ_s^* .

f_n^*	0.5	0.25	0.166	0.125
$f_{n,s}^*$	0.435	0.208	0.147	0.11
ξ_s^*	0.019	0.017	0.014	0.011
C_J	0.321	0.445	0.275	0.291
μ_s	1.35	1.26	0.978	0.773
$\varpi_{s,n}^a$	1.74	1.66	1.77	1.76

Table 3.3: Effective f_n^* and ξ^* for a spring damped plate with various f_n^* for an initial angle $\theta_0 = -1^\circ$ and $\xi^* = 0$

where C_J represents the ‘‘added mass’’ from the fluid inertia and is defined by:

$$C_J = \frac{J_s^*}{J^*} - 1 = \frac{f_n^*}{f_{n,s}^*} - 1, \quad (3.30)$$

where $J_s^* = k^*/\varpi_{n,s}^*$ is the effective nondimensional inertia.

μ_s is the effective damping of the spring-damped ellipse system using an alternative non-dimensionalization. That is to say the maximum angular velocity in the vacuum U_{max} was used instead of the arbitrary $U_\infty = 0.72$, or to relate U_{max} to the physical system: $U_{max} = a \theta_0 \varpi_n$. This nondimensional space is denoted by the superscript a . Thus μ_s is equal to:

$$\mu_s = 2 \xi_s^a J_s^a \varpi_{s,n}^a, \quad (3.31)$$

with $\varpi_{s,n}^a = 2 \pi f_n L / (a \theta_0 \varpi_n)$. From this table, it is apparent that ξ_s^* introduced by the simulation decreases with f_n^* in a quadratic manner, while $f_{n,s}^*$ is systematically lower than the one prescribed in the simulation. Under these conditions, f_n^* is proportional to the prescribed f_n^* , and can be approximated by $f_{n,s}^* = 0.87 f_n^*$, this is also reflected in the $\varpi_{s,n}^a$ and C_J evolution, where one can see that $\varpi_{s,n}^a$ and C_J are almost constant.

The ξ_s^* evolution is confirmed by the μ_s evolution which shows that the damping decreases faster than $f_{n,s}^*$. The presence of a ξ_s^* in the ellipse flow simulation is not unexpected and can be attributed to the effective viscosity of the simulation as well as the fluid inertia induced damping. As the initial angular velocity varies in a similar fashion to the prescribed f_n^* (see equation 3.27), one can also deduce that ξ_s^* is very

sensitive the angular velocity. Indeed, it seems that ξ_s^* increases very quickly with f_n^* to stabilize afterwards around 0.02 for $f_n^* \geq 0.25$. The evolution of $f_{n,s}^*$ is difficult to interpret and would require to further assess the motion influence. As a preliminary analysis, it can nevertheless be stated that, despite a discrepancy for $f_n^* = 0.25$, the fluid inertia influence seem relatively constant as can be deduced from the C_J values.

Further trials with other initial angles show the ξ_s^* increases and the $f_{n,s}^*$ decreases (albeit to a lesser degree than ξ_s^*) when θ_0 increases. It follows the analogy with the added mass for the $f_{n,s}^*$, and the additional viscosity for the ξ_s^* . However, it is difficult to assess those parameters past $\theta_0 = -2.5^\circ$, because as there is no freeflow, the blob vortices tend to stagnate around the plate and provoke a similar instability observed at $f_n^* = 8$ for $\theta_0 = -1^\circ$. The origin of this instability is already explained in an above paragraph and has been seen to be linked to an f_n / θ_0 criterion. This criterion implies that to have a stable simulation, if one raises θ_0 , one must raise f_n in a similar manner. However, one can only raise f_n up to a limit related to the timestep chosen. This limit is due to the differential equation integration and can only be changed by lowering the timestep. This would imply that another parameter is introduced which is beyond the scope of the present study.

Generally, these variations of ξ^* and f_n^* were not unexpected since even for a translation motion in a quiescent fluid the additional damping is nonlinear (see Blevins [7]). Concerning the numerical effects, it can be conjectured that because there is inadequate blob vortices overlap, the simulation behaves like a point vortex method except that the blob vortices core radius is big enough to introduce artificially a viscosity, as explained in section 3.2.3.

Results in table 3.4 concern the evolution of the effective ξ^* and f_n^* (titled respectively $f_{n,s}^*$ and ξ_s^*) produced by the ellipse/flow system for a varying ξ^* with $\theta_0 = -1^\circ$ and $f_n^* = 0.5$.

ξ^*	0.	0.02	0.04	0.08	0.2
$f_{n,s}^*$	0.435	0.432	0.435	0.479	0.431
ξ_s^*	0.019	0.0398	0.0562	0.0955	0.1993

Table 3.4: Effective f_n^* and ξ^* for a spring damped plate with various ξ^* for an initial angle $\theta_0 = -1^\circ$ and $f_n^* = 0.5$

One can see that for low values of prescribed ξ^* , there is an additional damping of about 0.02, similarly to what has been noted above for $f_n^* = 0.5$ and $\xi^* = 0$. On the other hand for much higher value of ξ^* , this additional damping disappears. Remark that the ξ^* variations have little influence on $f_{n,s}^*$. It also confirms the influence of the maximum angular velocity, as U_{max} stays constant here.

Now looking to at early times of the simulation –just after the ellipse is set in motion– streamline are plotted using simulation with $\theta_0 = -1^\circ$ and $1/f_n^* = 2$. It is then easy to see the flow evolution as the ellipse is moving. At $t^* = 2.5$, figure 3.47(a), there has been one pseudo-period (the ellipse has been rotated once counterclockwise and once clockwise) and is now rocking counterclockwise. Here, the ellipse has a near zero angular velocity. Predictably, there are two eddies present at both ellipse tips. The upper eddy has a counterclockwise circulation, while the lower one has a clockwise circulation. Both eddies are therefore now opposing the ellipse motion. Slightly above these eddies, one can see the eddies shed from the initial ellipse counterclockwise motion. The counterclockwise ellipse motion is also noticeable due to the streamline drawn toward the ellipse on the lower edge. A slight streamline asymmetry also appears which denotes a slight asymmetry in the vortices position and vorticity.

At $t^* = 8.$, figure 3.47(b), the ellipse has had four pseudo-periods and is in the middle of the fifth. After a counterclockwise motion, its angular velocity is now close to 0. Nevertheless, the streamlines crossing the body denote a slight clockwise body motion. At both tips, there are eddies shed from the previous timesteps. Here the flow asymmetry has developed, and is continuing as the ellipse is rotated clockwise at $t^* = 8.6$ (figure 3.47(c)), which illustrates that the angular velocity is near a peak. As the ellipse edges reach the previous shed vortices, one can see how the eddies closer to the edges are diffused while the surrounding eddies are passing round this eddy. At this point it seems that these last eddies force one eddy to stay near the edge. Furthermore, during one

pseudo period, at each edge two eddies of opposing strength are generated. From those two effects, it follows that while the motion is still shedding vortices at each oscillation, the newly created vortices have less and less circulation.

Therefore, here is a flow mechanism as the body is set in motion which, independently from the simulation shortfall, explains part of the damping introduction. As the shed vortices oppose the ellipse motion, it may also partly explain why there is a diminution of θ oscillation frequency, because the shed eddies strength is directly linked to the ellipse angular velocity during the oscillation, their strength being increased with the angular velocity and vice versa.

In conclusion, the flow dynamics introduce some additional damping, but the numerical simulation has been seen to also add some further modifications. A flow added mass diminishes f_n^* , as well as a nonlinear ξ^* which is dependent on the angular velocity of the plate. Concerning this “effective ξ^* ”, the evolution is due not only to the flow added mass but also the simulation effective viscosity. Other trials have shown that even when using $\xi^* = 0.2$, there is still an effect due to damping introduced by the simulation, and that non zero U_∞ introduces further modifications to the “effective ξ^* ”. However, further tests are necessary to better assess to which extent the simulation influences the effective ξ^* and f_n^* , especially concerning the influence of viscosity on the system.

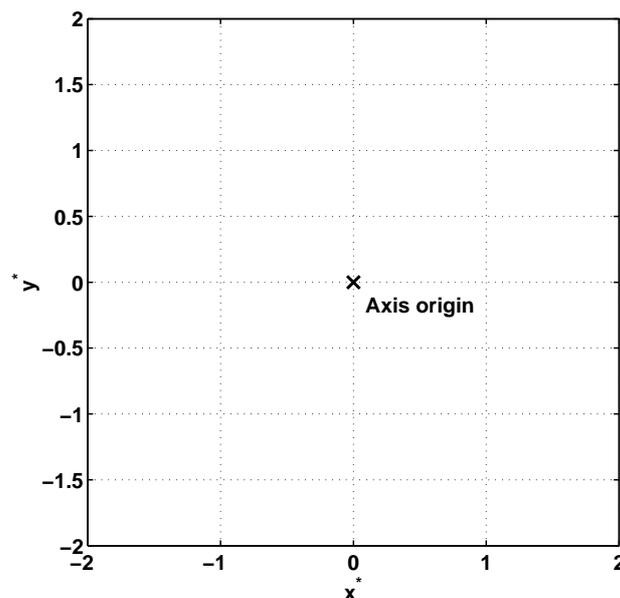
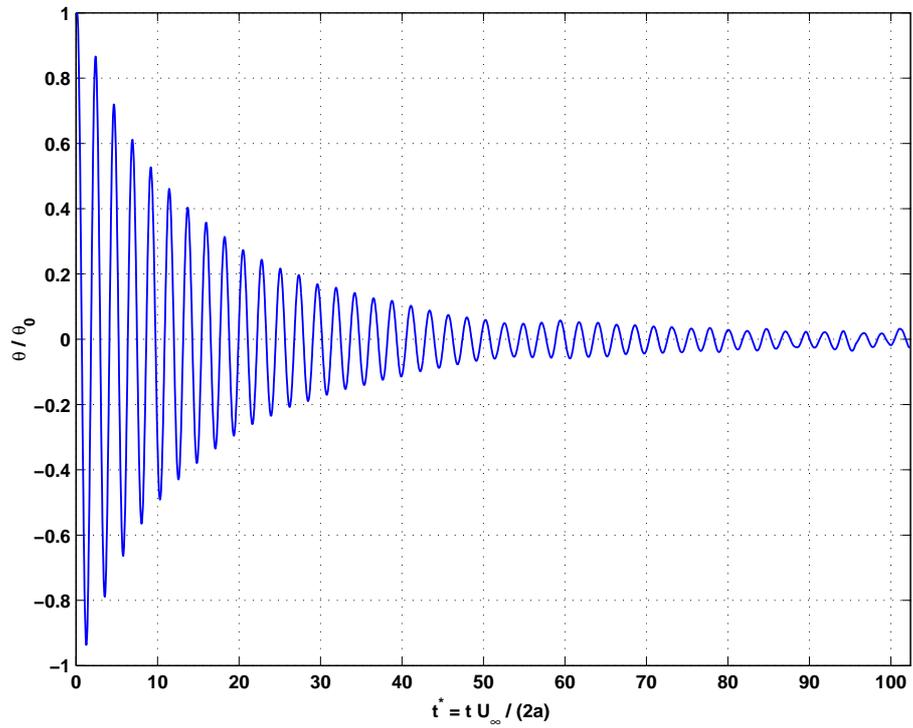
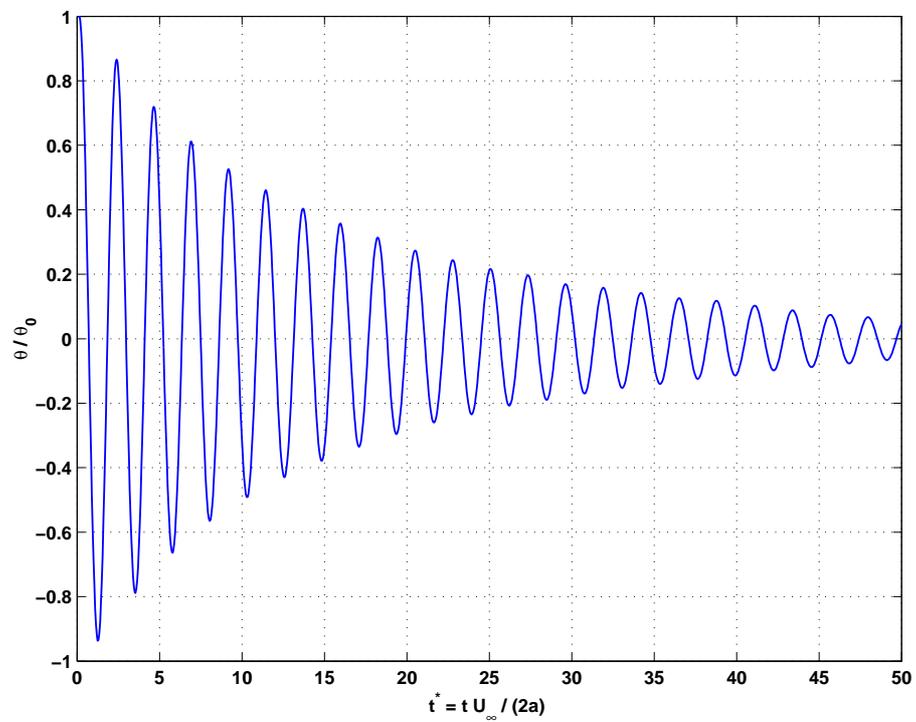


Figure 3.39: Scale for the streamline plot in figures 3.45, 3.46, and 3.47.

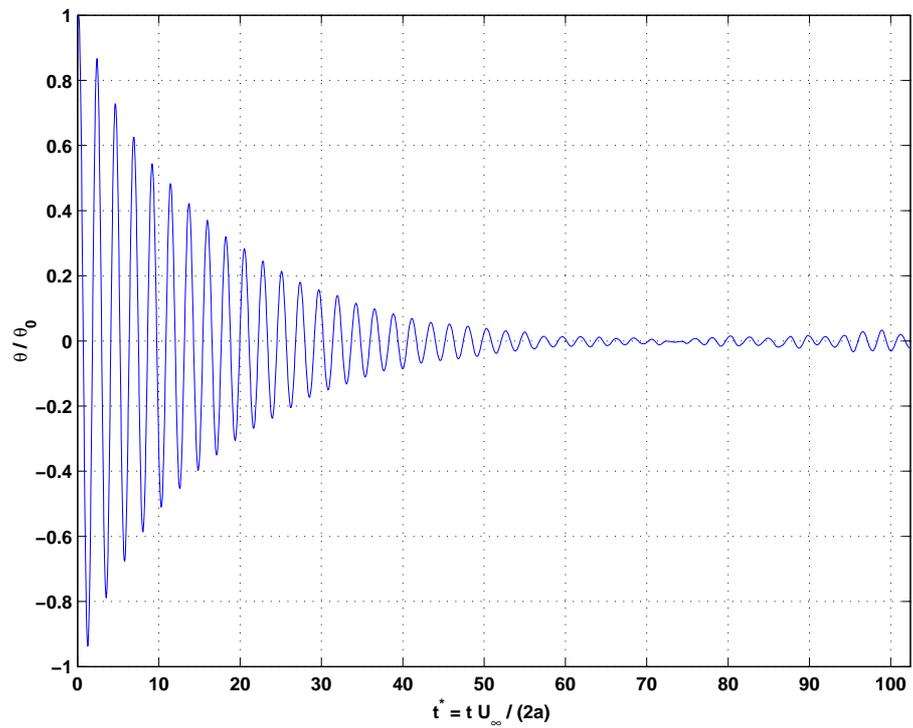


(a) Time evolution

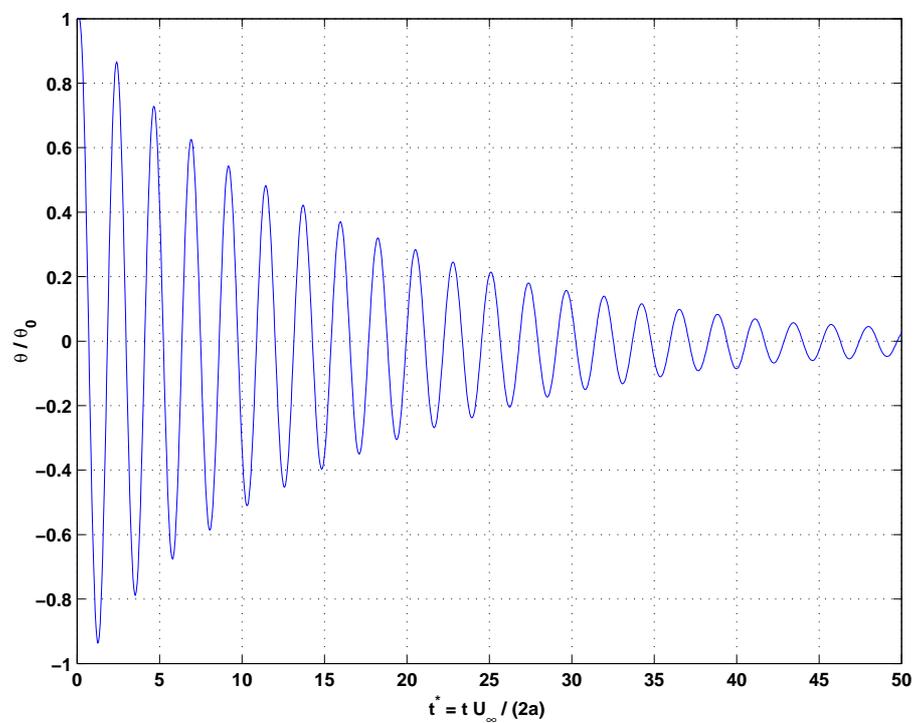


(b) Close-up view of the time evolution

Figure 3.40: Time evolution of the angle of the plate θ for a spring damped plate with $f_n^* = 1/2$, $\xi^* = 0$ and starting with an initial angle $\theta_0 = -1^\circ$.



(a) Time evolution



(b) Close-up view of the time evolution

Figure 3.41: Time evolution of the angle of the plate θ for a spring damped plate with $f_n^* = 1/2$, $\xi^* = 0$ and starting with an initial angle $\theta_0 = 1^\circ$.

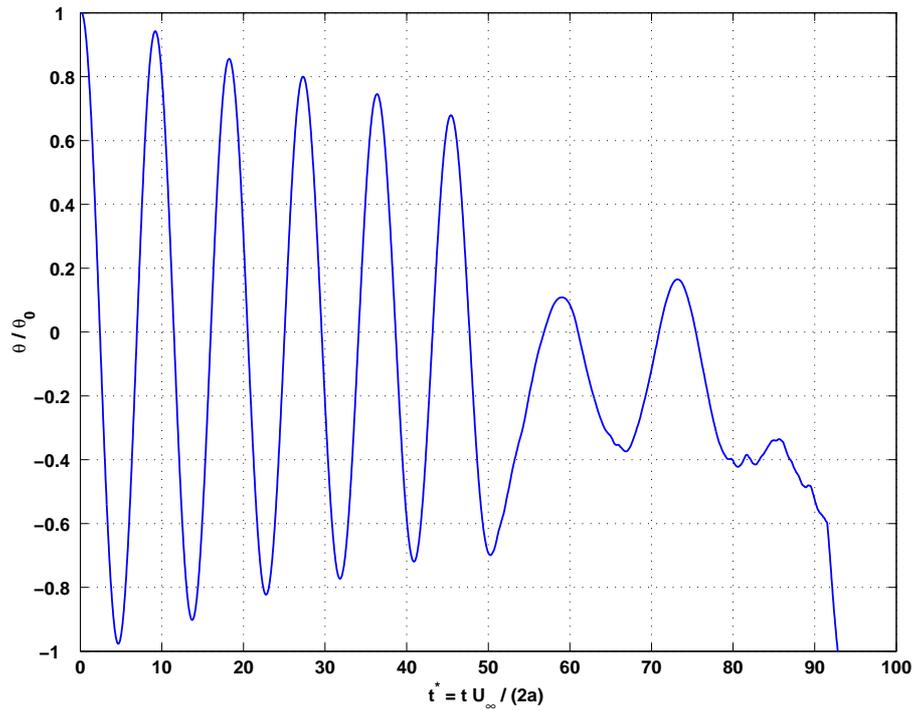


Figure 3.42: Time evolution of the angle of the plate θ for a spring damped plate with $f_n^* = 1/8$, $\xi^* = 0$ and starting with an initial angle $\theta_0 = -1^\circ$.

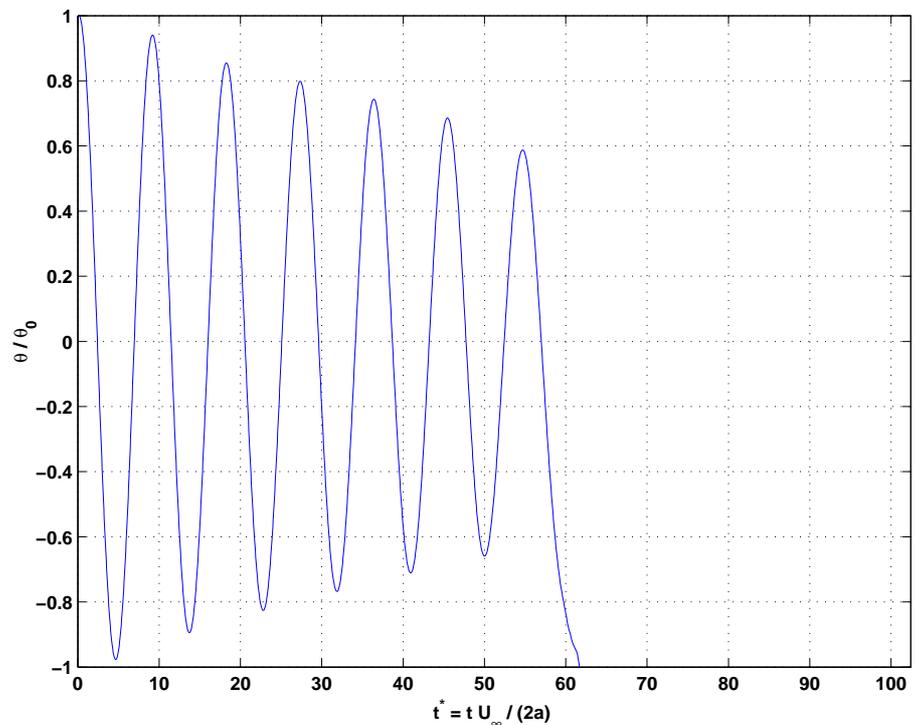
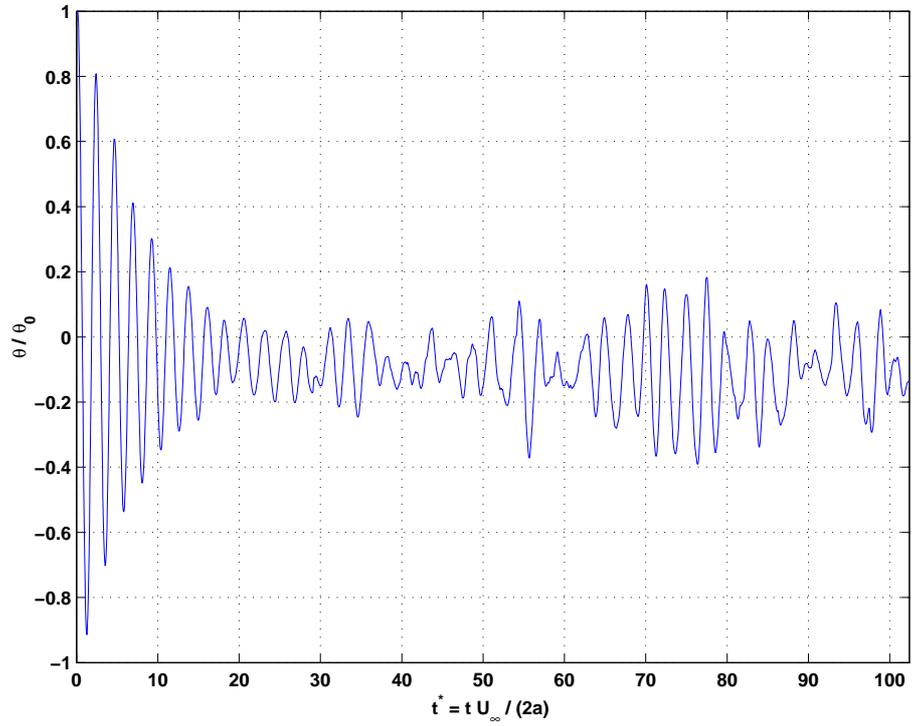
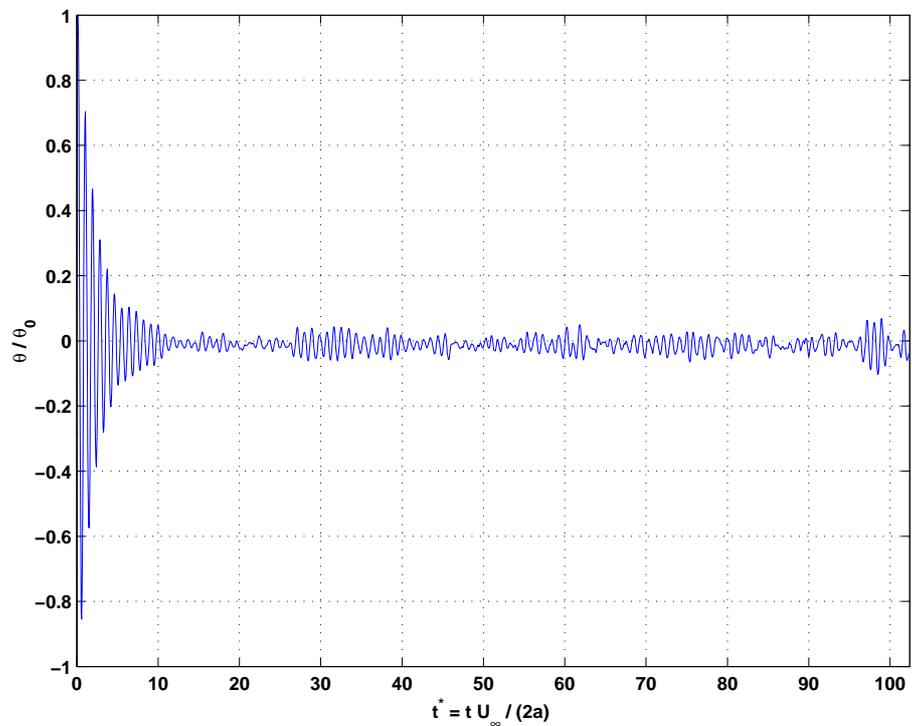


Figure 3.43: Time evolution of the angle of the plate θ for a spring damped plate with $f_n^* = 1/8$, $\xi^* = 0$ and starting with an initial angle $\theta_0 = 1^\circ$.

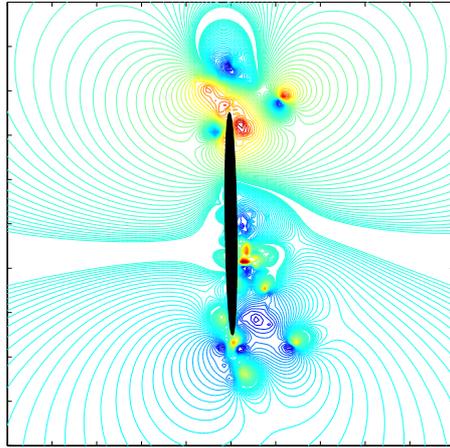


(a) Time evolution for $f_n^* = 1/2$

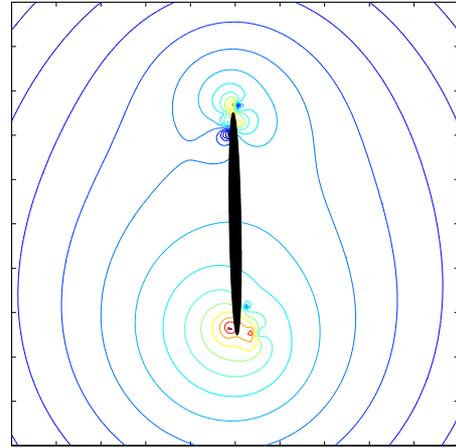


(b) Time evolution for $f_n^* = 1/0.8$

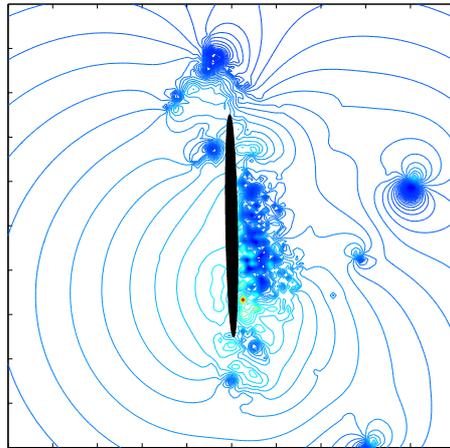
Figure 3.44: Time evolution of the angle of the plate θ for a spring damped plate with $\xi^* = 0$ and starting with an initial angle $\theta_0 = -2.5^\circ$.



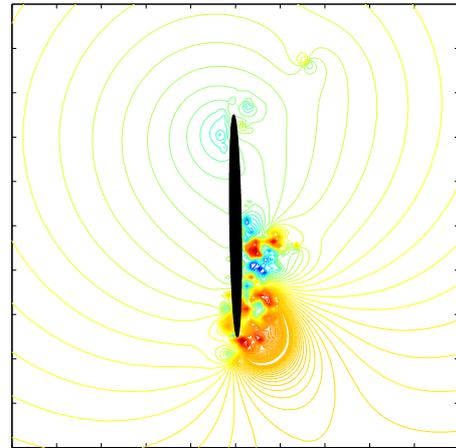
(a) $f_n^* = 1/2, t^* = 25$



(b) $f_n^* = 1/8, t^* = 25$

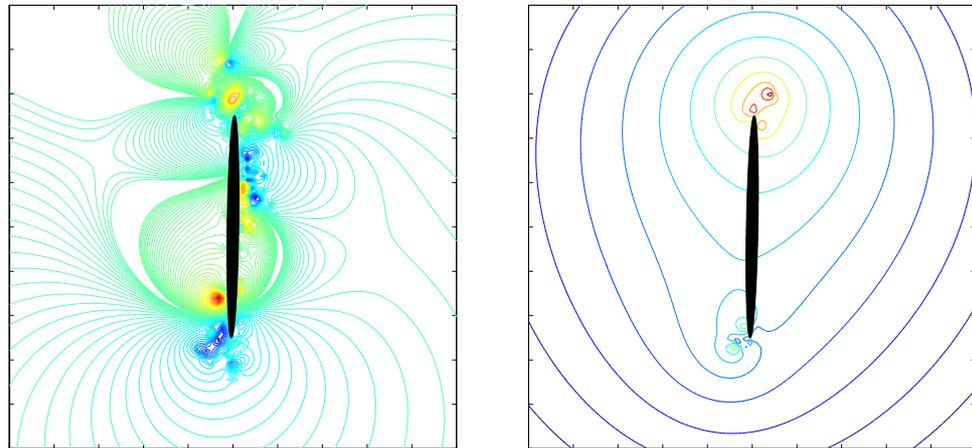


(c) $f_n^* = 1/2, t^* = 60$



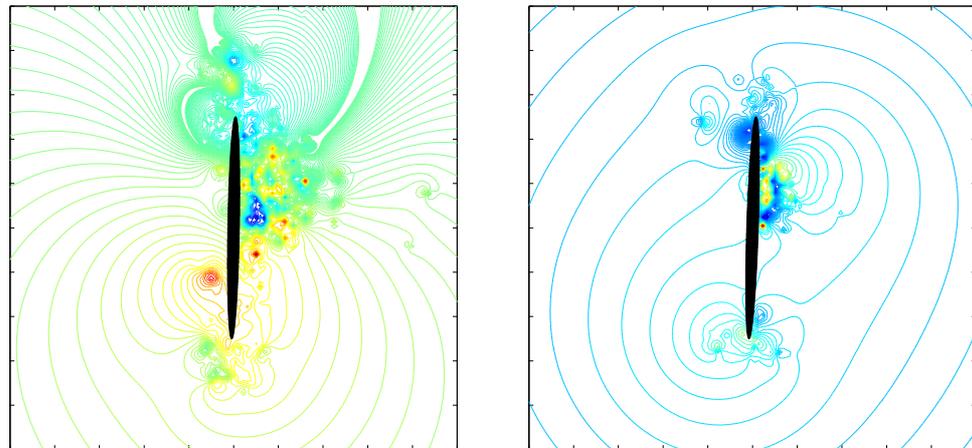
(d) $f_n^* = 1/8, t^* = 60$

Figure 3.45: Streamlines comparison of the flow for the Spalart code with $\theta_0 = -1^\circ$ and $f_n^* = 1/[2, 8]$. Streamfunction values range is $[-0.066, 0.066] \times (2aU_\infty)$. It corresponds to $[-0.0423, 0.0423] \times (2aU_{max})$ for $f_n^* = 1/2$, and $[-0.169, 0.169] \times (2aU_{max})$ for $f_n^* = 1/8$ (U_{max} varies with f_n^* see equation 3.27). In every cases, values are concentrated around 0. See figure 3.39 for the scale.



(a) $f_n^* = 1/2, t^* = 25$

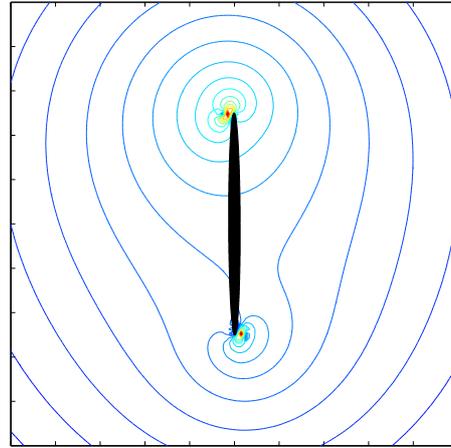
(b) $f_n^* = 1/8, t^* = 25$



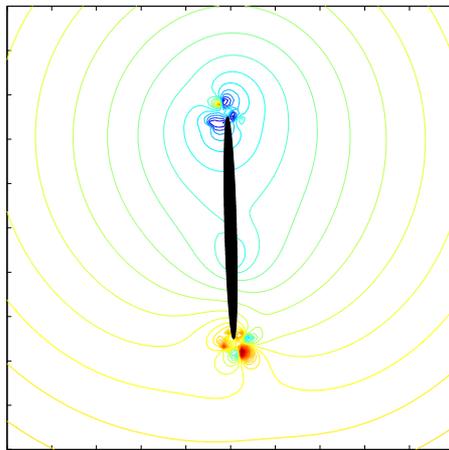
(c) $f_n^* = 1/2, t^* = 60$

(d) $f_n^* = 1/8, t^* = 60$

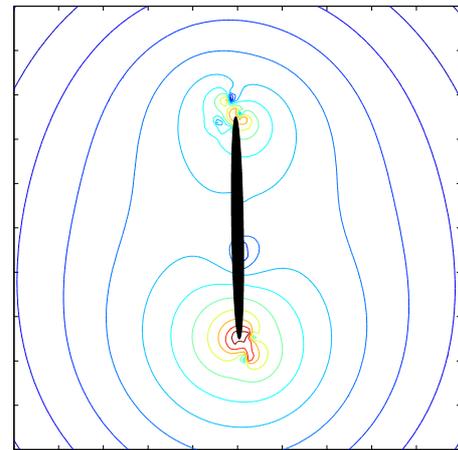
Figure 3.46: Streamlines comparison of the flow for the Spalart code with $\theta_0 = 1^\circ$ and $f_n^* = 1/[2, 8]$. Streamfunction values range is $[-0.066, 0.066] \times (2aU_\infty)$. It corresponds to $[-0.0423, 0.0423] \times (2aU_{max})$ for $f_n^* = 1/2$, and $[-0.169, 0.169] \times (2aU_{max})$ for $f_n^* = 1/8$ (U_{max} varies with f_n^* see equation 3.27). In both cases, values are concentrated around 0. See figure 3.39 for the scale.



(a) $t^* = 2.5$



(b) $t^* = 8$



(c) $t^* = 8.6$

Figure 3.47: Streamline of the flow for the Spalart code with $\theta_0 = -1^\circ$ and $f_n^* = 1/2$. Streamfunction values range is $[-0.066, 0.066] \times (2a U_\infty)$. It corresponds to $[-0.0423, 0.0423] \times (2a U_{max})$. Values are concentrated around 0. See figure 3.39 for the scale.

3.3.2 Spring damped plate with angular oscillations

Several important simulation parameters are to be considered for the study of the spring damped plate in a uniformly translating freeflow: the Reynolds number, the body to fluid density ratio, the reduced damping and the natural frequency of the torsional spring damped plate. As inviscid flows are studied here, the Reynolds number is not to be considered. However, the density ratio needs to be carefully chosen. Indeed, it conditions which part of the coupled system is dominant as well as the angular amplitude. This last criterion is important because in contrast to the transversely moving cylinder, the Spalart simulation is only able to simulate incidence angles α ranging from 45 to 135 degrees without modification¹, thus limits the maximum amplitude (sections 3.2.4 and 3.2.4.1). From equation 3.23 in the nondimensional space, one can see that the maximum angular amplitude for the spring-damped ellipse motion is indirectly related to this ratio through the body angular inertia and the C_m . Also, in order to restrict the number of parameters, it was chosen to focus on the natural frequency and the reduced damping. The body density will be of $\rho_b = 1000\rho$, with ρ the fluid density (keeping in mind the remark related to equation 3.26).

Even with such a body density, the maximum angle amplitude is close to the simulation limit, but this kind of flow with a rotating body is better suited to a vortex method, as shown in section 3.2.4.

The simulations were done using the Spalart code. The rest position of the spring ($\theta = 0^\circ$) is chosen at an incidence angle of $\alpha = 80^\circ$ in order to have an asymmetric loading concerning the aerodynamic moment to start the oscillations. The timestep is $\Delta t^* = 0.05$, and unless otherwise specified 2048 iterations were used per simulation. The body motion is initiated at $t^* = 12.8$ (256th time-step) in order to have a stabilized flow when starting the simulation.

3.3.2.1 Influence of the reduced damping

To study the influence of the reduced damping three values were used, $\xi^* = [0.01, 0.02, 0.04]$, and three reduced frequencies $f_n^* = [1/4, 1/8, 1/11]$. That is to say, if one consider the Strouhal number found in the experiment S for the flow at

¹That is considering that the plate may be in the limit case of very small oscillations, meaning one has to keep in mind the validity of the simulation for the plate at fixed angle.

$\alpha = 80^\circ$, it would correspond to $f_n^* = [1.64, 0.82, 0.59] S$, or with the actual simulation Strouhal number S_s found in section 3.2.3, $f_n^* = [2.12, 1.05, 0.77] S_s$. This allows to have one f_n^* greater than the value of Strouhal number of the flow, one f_n^* near S , and one f_n^* smaller than S . Otherwise, in order to obtain the nondimensional flow shedding frequency f_s^* , the Strouhal number extraction was made in the same manner as in section 3.2.3.

The main focus here was to choose a reduced damping low enough to allow a sharp resonance of the spring-damped ellipse, but larger than the damping due to the numerical simulation. That is why, this range of ξ^* was chosen.

$1/f_n^*$	4			8			11		
ξ^*	0.01	0.02	0.04	0.01	0.02	0.04	0.01	0.02	0.04
f_s^*/f_n^*	0.4842	0.471	0.4689	1.0937	1.1069	1.1171	1.3564	1.3753	1.4269
θ_{max} (radian)	0.1199	0.097	0.0785	0.5765	0.5453	0.5223	0.4564	0.3973	0.3725

Table 3.5: Summary of the simulations detailing the variations of the shedding frequency f_s^* and the maximum amplitude of angle oscillation regarding the nondimensional natural frequency f_n^* .

From the various time plots (figures 3.48, 3.49, 3.50), it is obvious that in this range the reduced damping has little influence on the dynamic behaviour of the plate/flow system. This is confirmed by tables 3.5. Indeed, the evolution of both shedding frequency of the flow and maximum amplitude remains roughly the same as the reduced damping is changed. Three cases can then be defined:

- Mode 1: Little influence from the plate fluctuation on the vortex shedding process, for example at $1/f_n^* = 4$
- Mode 2: Resonance with a noticeable increase in θ_{max} and $f_s^*/f_n^* \simeq 1$
- Mode 3: Past this lock in range, the θ_{max} remains high compared to mode 1, and f_s^*/f_n^* is larger than 1. It seems the flow imposes the frequency.

From table 3.5, one can observe that the reduced damping ξ^* has specific effects on the broad behaviour which seems defined by f_n^* . For a given f_n^* , when one lower ξ^* the ratio f_s^*/f_n^* gets closer to 1, that is to say there tends to be better synchronization between

the flow frequency and the spring damped plate system as one lower the reduced damping. Conversely, the maximum amplitude tends to diminish for higher ξ^* . Overall, this seems to indicate that there is less coupling between the flow dynamics and the spring damped ellipse as one increase ξ^* .

Otherwise, the variations from $\xi^* = 0.02$ to $\xi^* = 0.01$ and 0.04 are about $\pm 3\%$ for the f_s^*/f_n^* ratio, and for the maximum amplitude θ_{max} at most $\pm 20\%$, with minimal variation of θ_{max} at $f_n^* = 8$. Thus, in the parameter range considered, ξ^* has little influence concerning the flow shedding frequency.

As a conclusion, the reduced damping ξ^* does not seem to be a major parameter for the dynamic behaviour of the combined flow/plate system. Indeed, it is expected to broaden the lock-in range as ξ^* is lowered but not to fundamentally change the different modes occurring as f_n^* evolves, nor the mean f_n^* where the lock in takes place. Nevertheless, further tests would be required to fully know the effects of the reduced damping on the system with a bigger range of parameters, in particular with lower J^* .

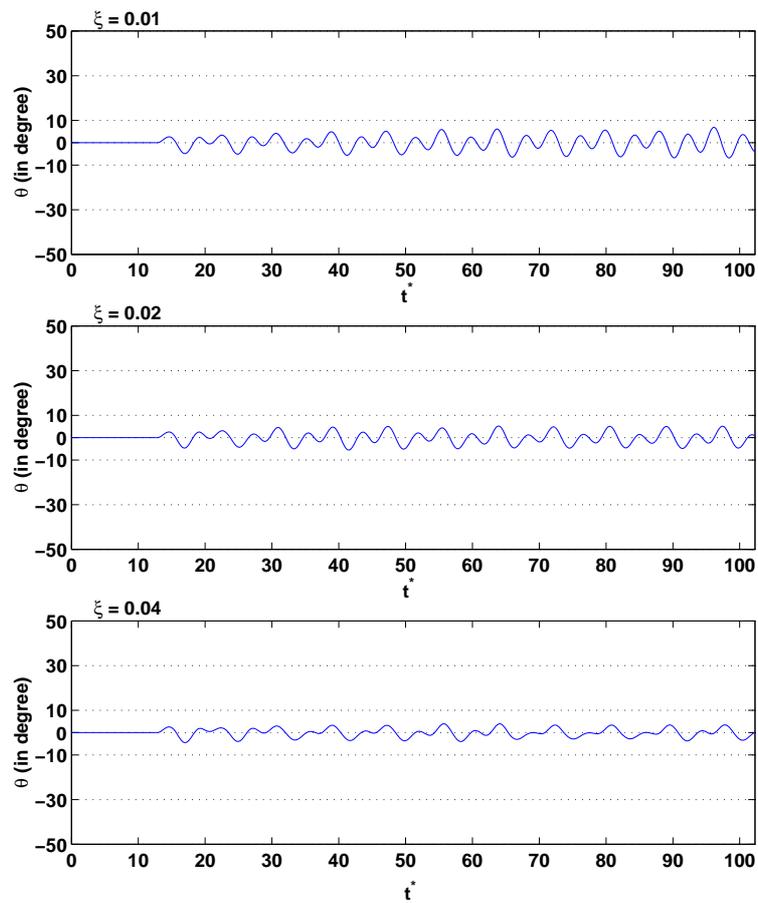


Figure 3.48: Time plot of the angle evolution for $f_n^* = 1/4$ with various ξ^*

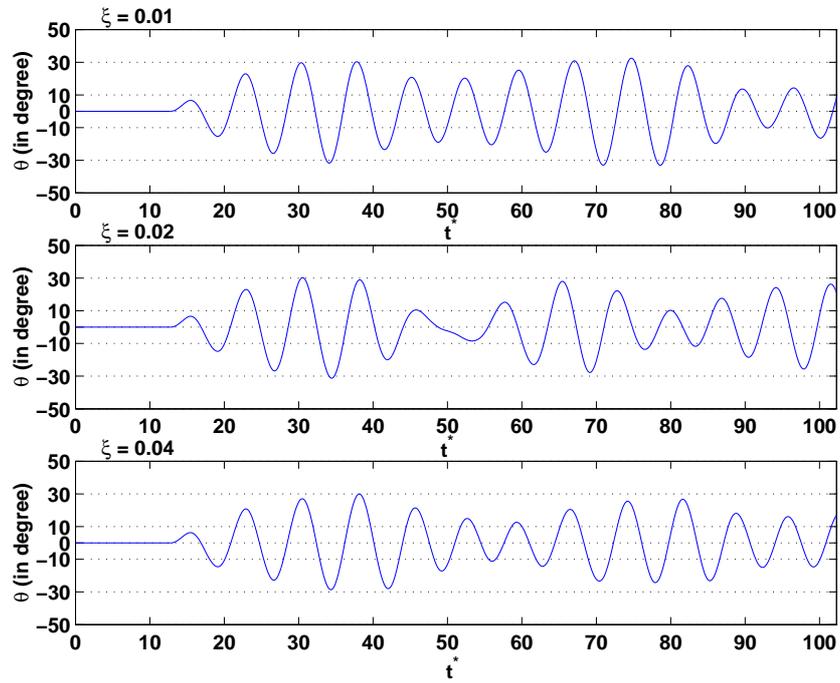


Figure 3.49: Time plot of the angle evolution for $f_n^* = 1/8$ with various ξ^*

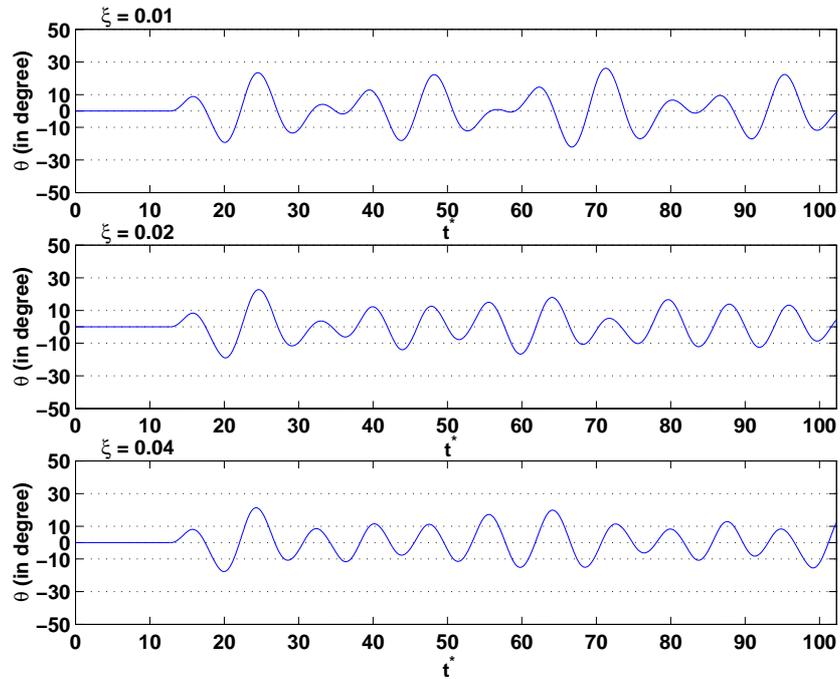


Figure 3.50: Time plot of the angle evolution for $f_n^* = 1/11$ with various ξ^*

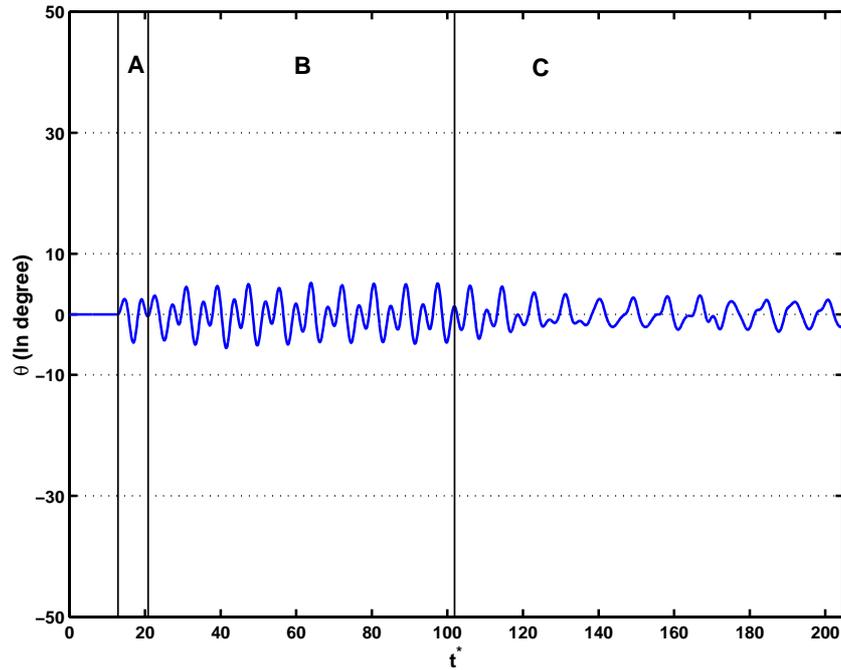
3.3.2.2 Influence of the natural frequency

To investigate the influence of the natural frequency, a given reduced damping $\xi^* = 0.02$ was used with values of reduced frequencies $f_n^* = [1/4, 1/7, 1/9, 1/11]$, which correspond respectively to $[1.64, 0.94, 0.73, 0.6] S$ or $[2.12, 1.21, 0.94, 0.77] S_s$. Again, this enables to investigate the dynamic behaviour of the system around a frequency equal to the Strouhal number of the flow with a thin ellipse at a fixed 80° incidence. Otherwise, in order to obtain the nondimensional flow shedding frequency f_s^* , the Strouhal number extraction is the same as in section 3.2.3, except that the C_m power spectral density (PSD or P_{XX}^2) was preferred as it is less sensitive to the body motion than C_t , and is thus better suited to extract the flow shedding frequency. The simulation each used 4096 iterations.

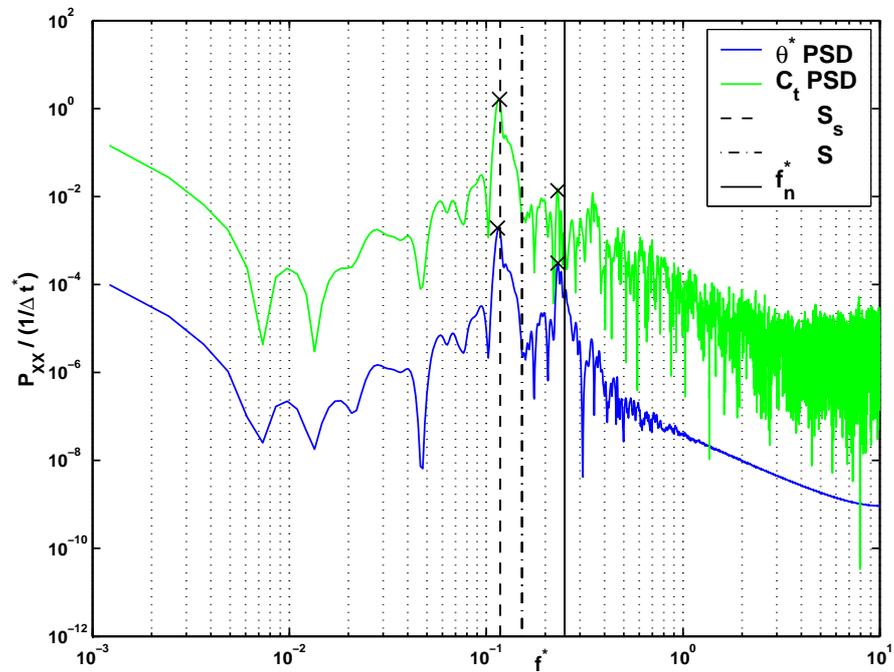
Incidentally, note that the range for the sequence used for the PSD is taken from the time where the spring damped plate is freed until the 4096th iterations (or $t^* = [12.8, 204.8]$), in contrast to the range used for extracting the flow frequency f_s^* , which starts at $t^* = 51.2$ or iteration 1024. This enables to look at the frequency characteristics from the start of the motion.

Using angle $\theta(t)$ evolution plots (figures 3.51(a), 3.52(a), 3.53(a), 3.54(a)), one see that the mean value of the angle θ is very close to zero; the minimum is reached when $1/f_n^* = 4$ with $\bar{\theta} = -0.002^\circ$ ($\max(|\theta|) \simeq 5^\circ$) and the maximum at $1/f_n^* = 11$ with $\bar{\theta} = 0.57^\circ$ ($\max(|\theta|) \simeq 28^\circ$). Overall, the mean value is increasing with $1/f_n$, but is nevertheless remaining low compared to the maximum amplitude. One can thus conclude that the spring damped ellipse/flow system tends to stabilize around the zero spring deflection rest position.

²The P_{XX} represents the signal power per frequency in the spectrum hence the term power spectrum density.

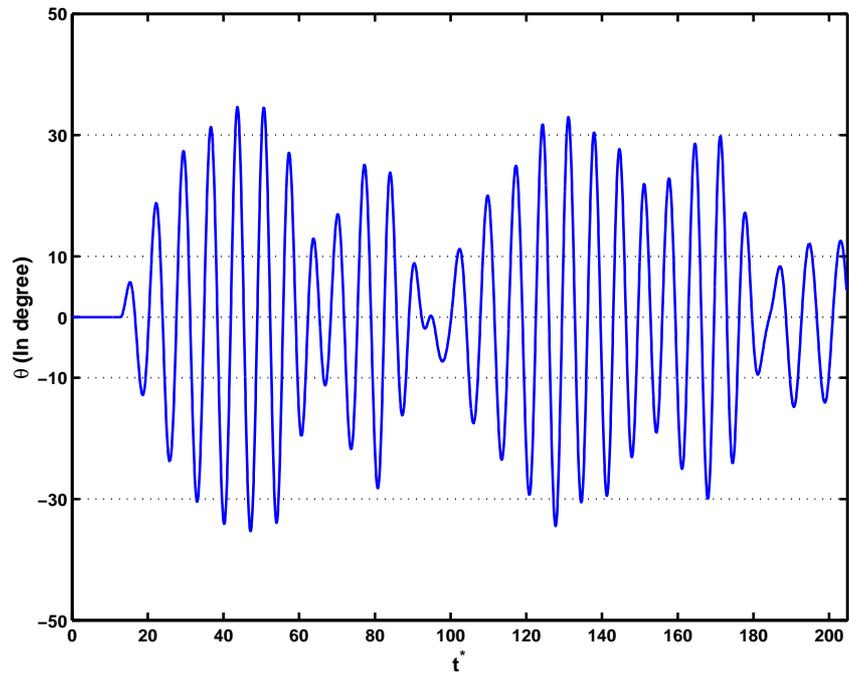


(a) Time plot of the angle $\theta(t)$ evolution. Vertical lines indicate boundaries of phase A , B and C described in text

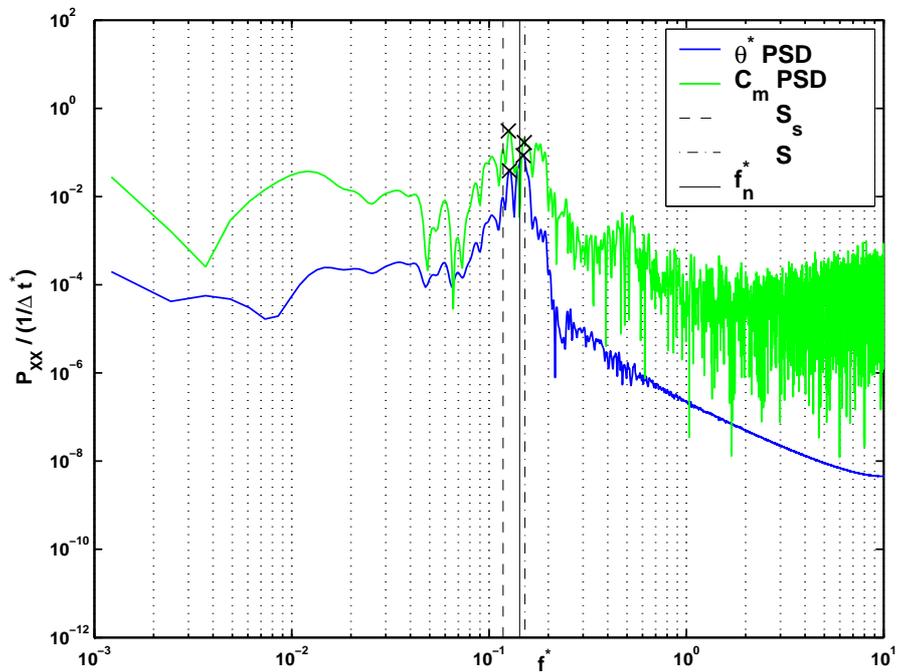


(b) PSD plot for θ^* and C_t

Figure 3.51: Angle evolution $\theta(t)$ and PSD plot for θ^* and C_m (for $t^* = [12.8, 204.8]$) in the case $\xi^* = 0.02$ and $1/f_n^* = 4$. The crosses in the PSD are the main harmonics represented in $\theta^*(t)$

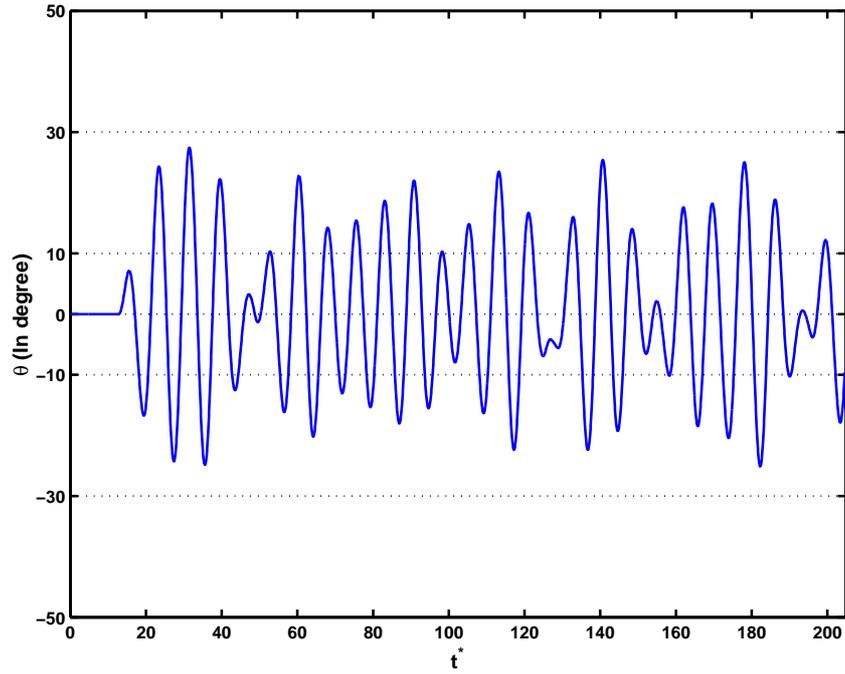


(a) Time plot of the angle $\theta(t)$ evolution

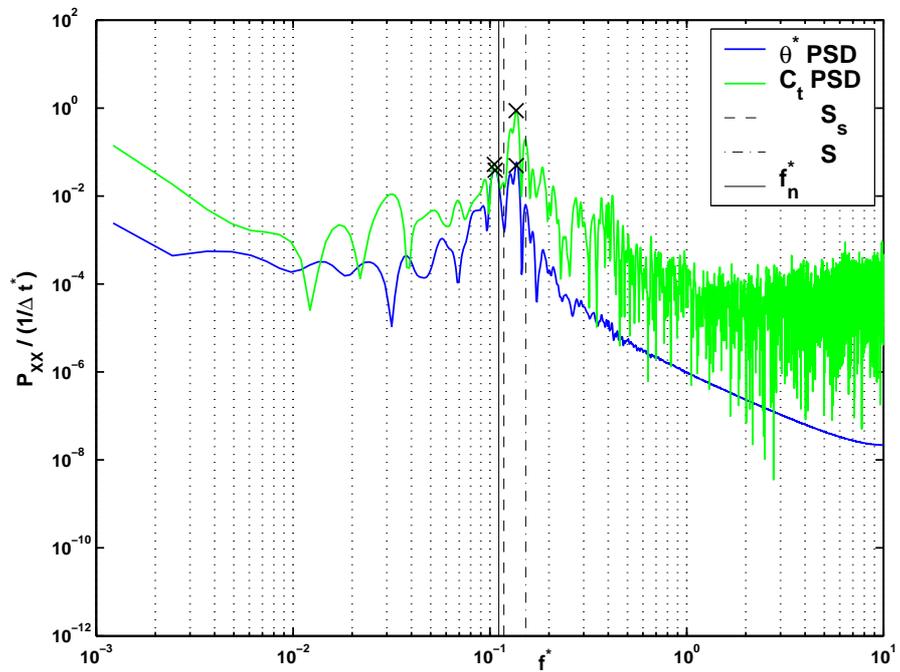


(b) PSD plot for θ^* and C_t

Figure 3.52: Angle evolution $\theta(t)$ and PSD plot for θ^* and C_m (for $t^* = [12.8, 204.8]$) in the case $\xi^* = 0.02$ and $1/f_n^* = 7$. The crosses in the PSD are the main harmonics represented in $\theta^*(t)$

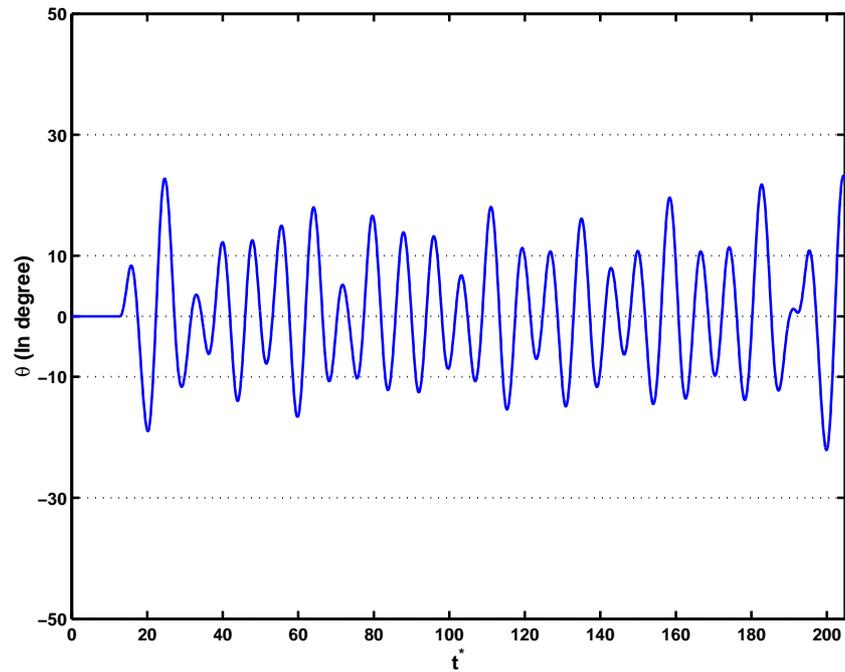


(a) Time plot of the angle $\theta(t)$ evolution

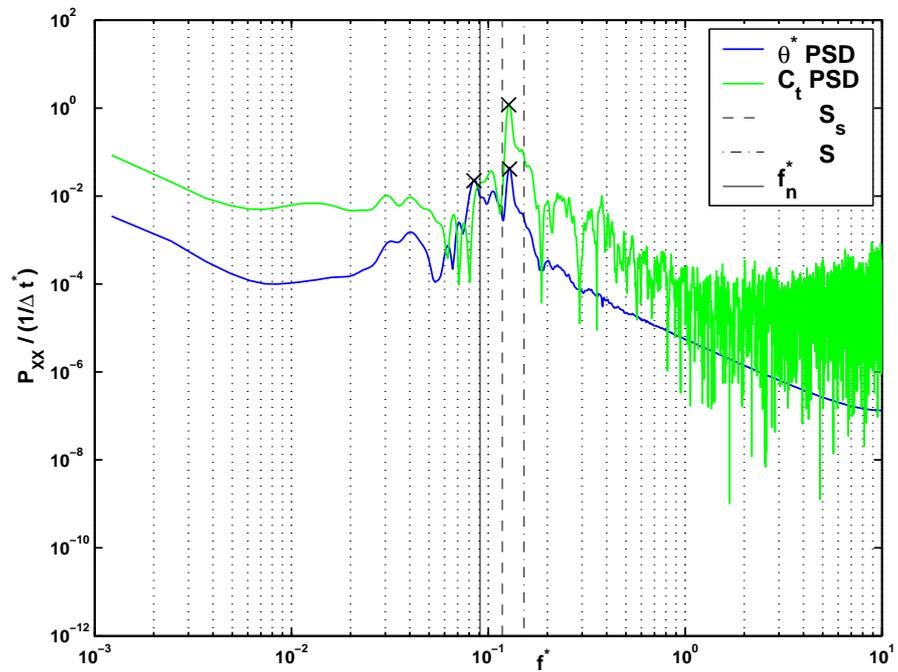


(b) PSD plot for θ^* and C_t

Figure 3.53: Angle evolution $\theta(t)$ and PSD plot for θ^* and C_m (for $t^* = [12.8, 204.8]$) in the case $\xi^* = 0.02$ and $1/f_n^* = 9$. The crosses in the PSD are the main harmonics represented in $\theta^*(t)$



(a) Time plot of the angle $\theta(t)$ evolution



(b) PSD plot for θ^* and C_t

Figure 3.54: Angle evolution $\theta(t)$ and PSD plot for θ^* and C_m (for $t^* = [12.8, 204.8]$) in the case $\xi^* = 0.02$ and $1/f_n^* = 11$. The crosses in the PSD are the main harmonics represented in $\theta^*(t)$

The PSD plots³ (figures 3.51(b), 3.52(b), 3.53(b), 3.54(b)) show that $\theta^*(t)$ contains two harmonics. It is visible for example for $1/f_n^* = 4$ (figure 3.51), one can see one harmonics originating from the spring damped ellipse/flow system. As in section 3.3.1, it is generally lower than the prescribed f_n^* . There is also another harmonic, this one associated with the vortex shedding. Note that in the $\theta(t)$ PSD, the harmonic associated with f_n^* is weaker than the harmonics associated with the flow shedding frequency. The flow shedding frequency, which can be directly evaluated from the $C_m(t)$ PSD, shows that there is mainly one shedding frequency. The behaviour differs slightly at $1/f_n^* = 7$, with a flow frequency f_s^* which is continuously varying around 0.155 in the range $f_s^* = [0.13, 0.18]$. In every case, there is always a simple vortex shedding, with the ellipse motion simply superimposed upon the shedding frequency.

$T^* = 1/f_n^*$		4	6.5	7	8	9	11
f_n^*		0.25	0.1538	0.1429	0.125	0.1111	0.0909
Time range used for PSD (start t^* end t^*)	12.8 12.8 + 3 T^*	0.2239	0.1531	0.1403	0.1256	0.1262	0.1318
	51.25 102.4	0.2377	0.1552	0.1574	0.1387	0.133	0.1247
	102.4 204.8	0.1159	0.1279	0.1549	0.1392	0.1367	0.1306

Table 3.6: Frequency f_θ^* of the main harmonics of $\theta(t)$ in the time range considered

In table 3.6, PSD was used to extract the main harmonic frequency f_θ^* of the signal $\theta(t)$ for different time ranges. This illustrates three phases characterizing the evolution of the flow once the plate is set in motion, and which are visible for $1/f_n^* = 4$ in figure 3.51. Prior to $t^* = 12.8$, when the ellipse is kept fixed, the main harmonic of $C_m(t)$ is at a nondimensional frequency of $1/f_s^* = 6.45$. The Strouhal number for a fixed ellipse at angle of 80° is $S_s = f_s^* = 0.118$. Therefore, during the starting regime (Phase A in figure 3.51), the coupled spring damped ellipse/flow θ frequency is dominated by the spring damped plate reduced frequency f_n^* . Then, during a transition regime (Phase B), two main harmonics are acting in the flow: one associated with f_n^* , and the other

³Using a numerical extraction of the power spectral density in Matlab, the resulting P_{XX} is always divided by the sampling frequency, that is the sample time.

one representing the resulting frequency of the coupled system and mainly dependent on S and S_s . Finally, in a near-periodic regime (Phase C) there remains one dominant harmonic in the coupled system, which mainly depends on S and S_s . The latter conforms to the mechanical response of a spring-damped ellipse forced with a periodic torque. These different phases are most obvious for the lower values of f_n^* .

When $1/f_n^*$ is around 7, the phases are slightly different and as there is another cycle present, which is due to the interaction of the spring-damped ellipse system and the flow. During a first cycle, at first there are actually the three phases albeit barely distinguishable because the two frequency (f_n^* and f_s^*) are very close. Then, it tends to provoke a C_m reduction through phase opposition as the frequency of f_s^* evolves. The motion amplitude then decreases, and as C_m increases as the motion is slowed and the flow recovers, another cycle begins. This process is described in the paragraphs below presenting some example of flow evolution.

Here are some examples of the stabilized flow evolution for the spring damped ellipse/flow system using different $1/f_n^*$ by plotting streamlines in phase C for one period of vortex shedding. The scale of those streamline is given in figure 3.55. $1/f_n^* = 4$ was used in figures 3.56, 3.57, 3.58, 3.59 with $t^* = [174.83, 184.32]$, $1/f_n^* = 7$ in figures 3.60, 3.61, 3.62 with $t^* = [134.87, 142.36]$, and $1/f_n^* = 11$ in figures 3.63, 3.64, and 3.65 with $t^* = [154.85, 162.34]$. The streamfunction value range had values of $[-1.19, 1.19] (2aU_\infty)$ with a concentration of values around 0.

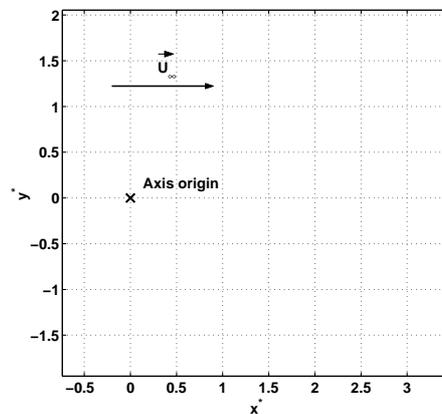
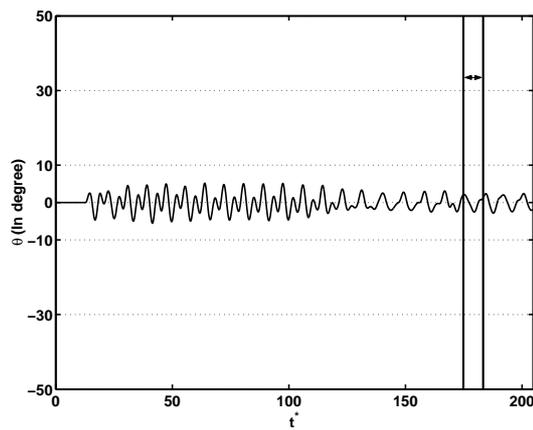
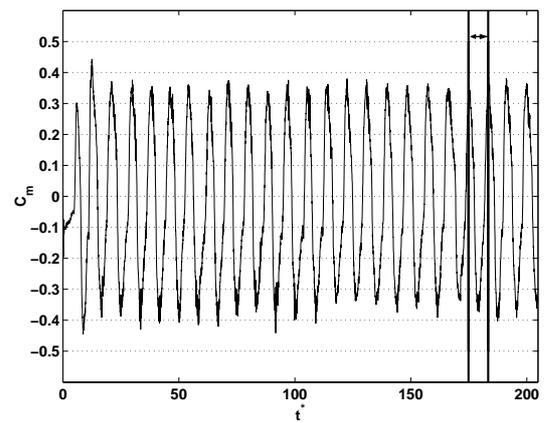


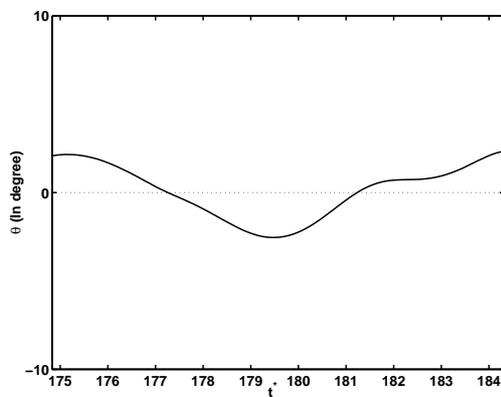
Figure 3.55: Scale plot for figures 3.57, 3.58, 3.56, 3.59, 3.61, 3.62, 3.64, 3.65



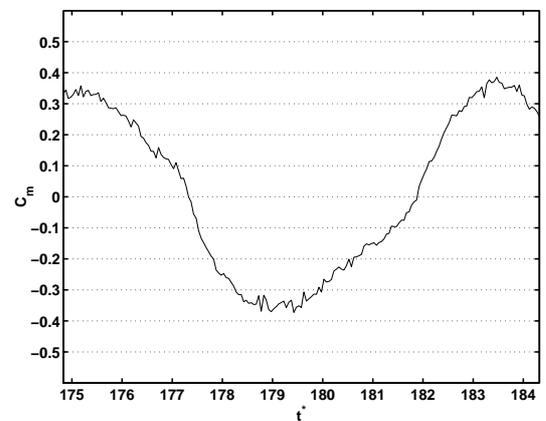
(a) θ range



(b) C_m range

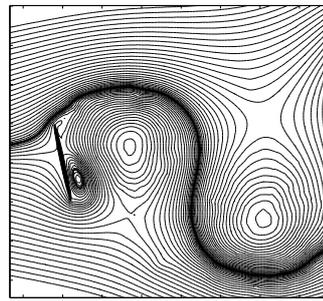


(c) θ range detail

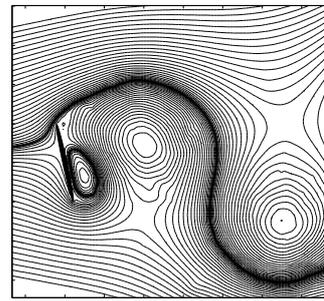


(d) C_m range detail

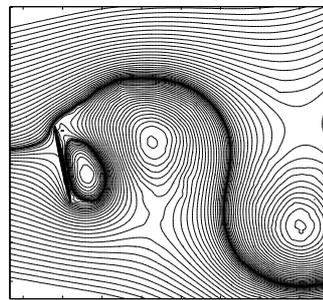
Figure 3.56: History of θ and C_m in phase C for $1/f_n^* = 4$



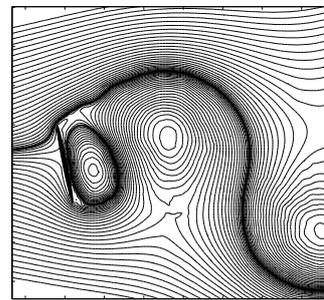
(a) $t^*=174.8, \theta=2.08^\circ$



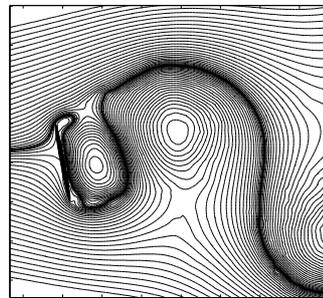
(b) $t^*=175.3, \theta=2.13^\circ$



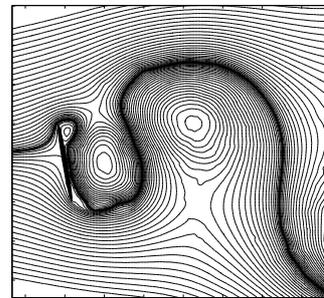
(c) $t^*=175.8, \theta=1.85^\circ$



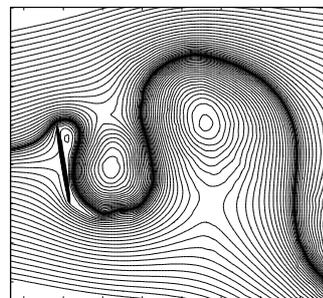
(d) $t^*=176.3, \theta=1.29^\circ$



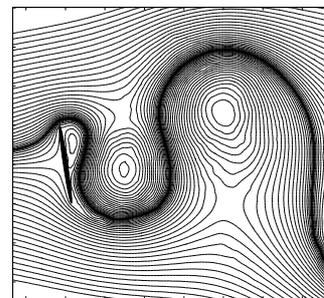
(e) $t^*=176.8, \theta=0.59^\circ$



(f) $t^*=177.3, \theta=-0.08^\circ$

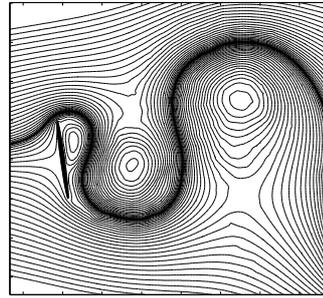


(g) $t^*=177.8, \theta=-0.68^\circ$

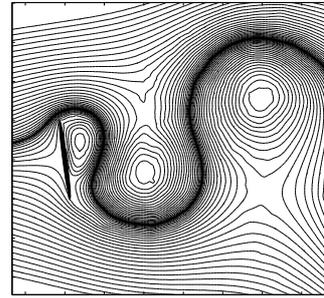


(h) $t^*=178.3, \theta=-1.40^\circ$

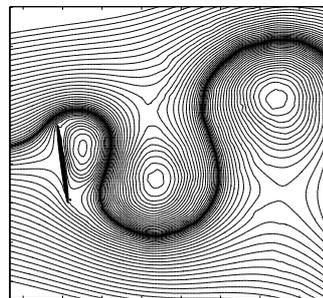
Figure 3.57: Streamlines for the flow in phase *C* for $1/f_n^* = 4$ with $t^* = [174.8, 178.3]$ (see figure 3.56). The scale is in figure 3.55.



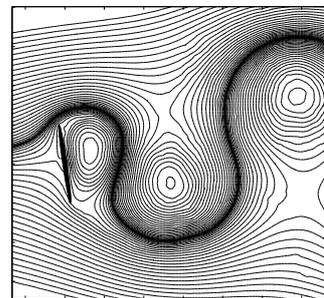
(a) $t^*=178.8, \theta=-2.11^\circ$



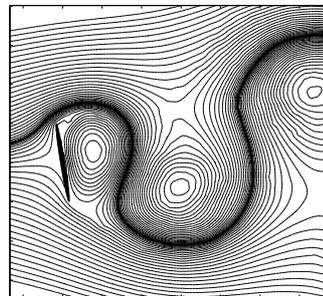
(b) $t^*=179.3, \theta=-2.52^\circ$



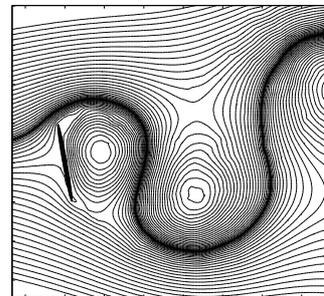
(c) $t^*=179.8, \theta=-2.40^\circ$



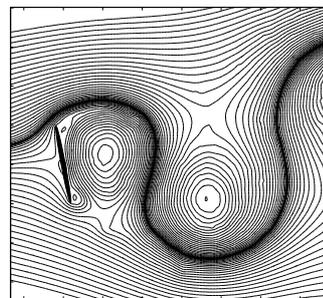
(d) $t^*=180.3, \theta=-1.76^\circ$



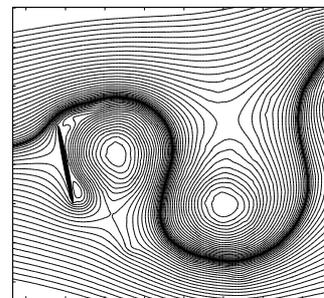
(e) $t^*=180.8, \theta=-0.78^\circ$



(f) $t^*=181.3, \theta=0.14^\circ$



(g) $t^*=181.8, \theta=0.64^\circ$



(h) $t^*=182.3, \theta=0.73^\circ$

Figure 3.58: Streamlines for the flow in phase *C* for $1/f_n^* = 4$ with $t^* = [178.8, 182.3]$ (see figure 3.56). The scale is in figure 3.55.

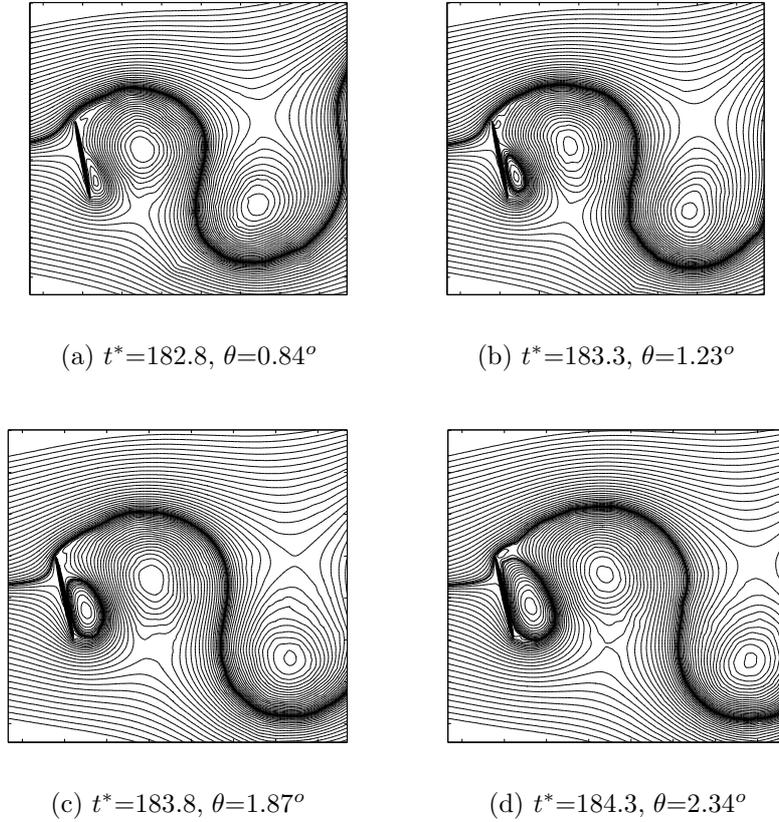


Figure 3.59: Streamlines for the flow in phase *C* for $1/f_n^* = 4$ with $t^* = [182.8, 184.3]$ (see figure 3.56). The scale is in figure 3.55.

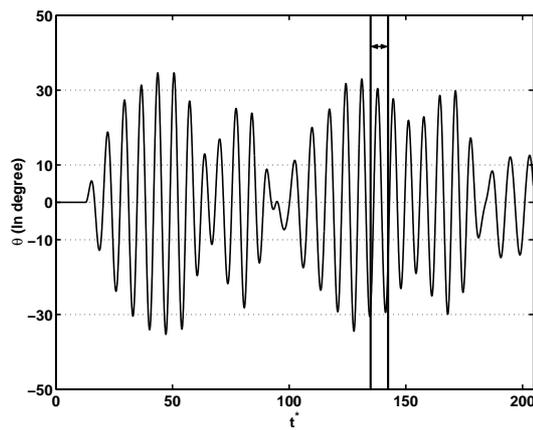
From the streamline plot figures 3.57, 3.58, and 3.59, phase *C* of the flow for $1/f_n^* = 4$ is characterized by a vortex street with little influence from the ellipse angular motion. Recall that for the fixed plate case with $\alpha_0 = 80^\circ$, there is a slightly positive circulation around the thin ellipse due to the flow asymmetry (see figure 3.20(b)). Here this circulation is now varying with the ellipse angular velocity. Indeed, a clockwise angular velocity introduces negative circulation around the plate and vice versa.

The angular velocity influence can first be seen around $t^* = 176.8$ (figure 3.57(e) and 3.57(f)), when the ellipse is moving clockwise at near maximum velocity. It ensues a slight deformation of the lower part of the nascent eddy⁴, due to the ellipse rotation. Note also that the zero value streamfunction is slightly off the ellipse body due to the added circulation around the ellipse. While also occurring when the ellipse is fixed, it

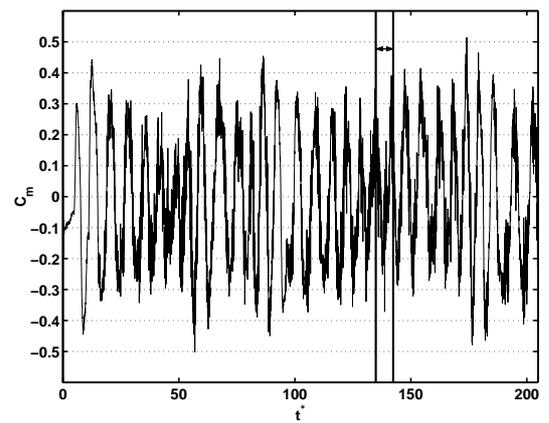
⁴The nascent eddy lower part is convected by the body motion. The reader must remember that the nascent eddy is composed of blob vortices. The nascent eddy “deformation” –or nascent eddy lower part convection– is thus the consequence of the blob vortices convection due to the body rotation.

happens faster here. The eddy then continues to be fed until $t^* = 179.3$ (figure 3.58(b)) where separation occurs. At $t^* = 181.8$ (figure 3.58(g)), it is near maximum angular velocity, this time counterclockwise. A small secondary eddy appear at the upper edge of the ellipse. Again, while this is also visible for a fixed ellipse, the eddy is slightly stronger here. Meanwhile on the lower tip a nascent eddy is still fed by the upper shear layer. Afterwards, the vortex shedding process continues until the separation at $t^* = 184.3$ (figure 3.59).

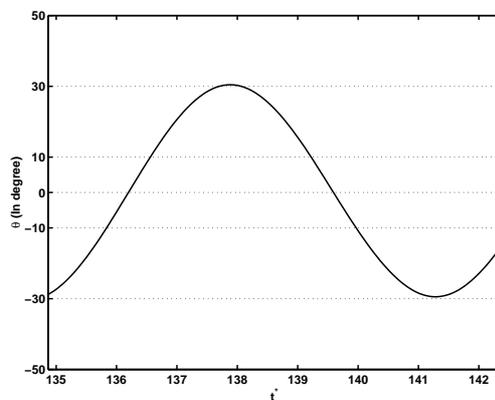
In figure 3.56, one can see that the evolution of θ is to be similar to that for C_m as they both reach their maximum and minimum at around the same time. There is also a delay between the two curves, which can be explained by the fact that the θ curve is mainly composed of two harmonics (one originating from the flow and the other from the spring damped ellipse), whereas there is only one harmonics in C_m (two eddies are shed per period). Overall, one can conclude that when $1/f_n^* = 4$, the spring damped ellipse and the flow simulation are more or less independent of each other.



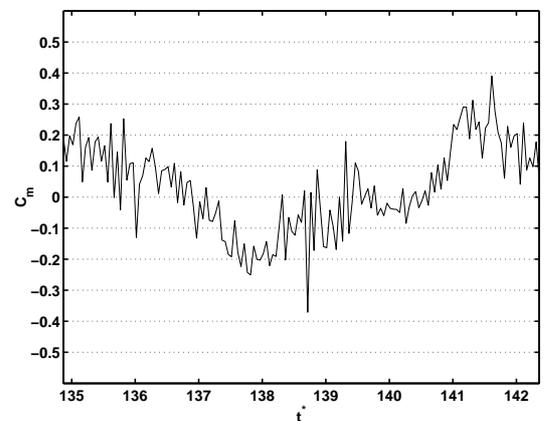
(a) θ range



(b) C_m range

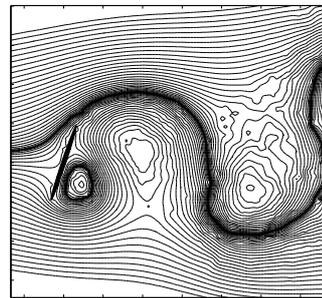


(c) θ range detail

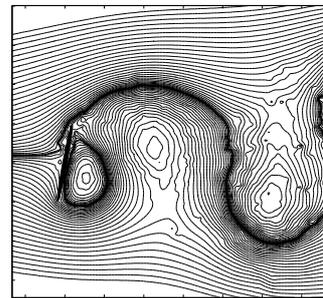


(d) C_m range detail

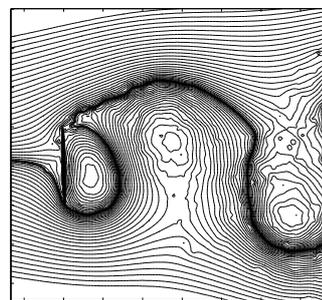
Figure 3.60: History of θ and C_m in phase C for $1/f_n^* = 7$



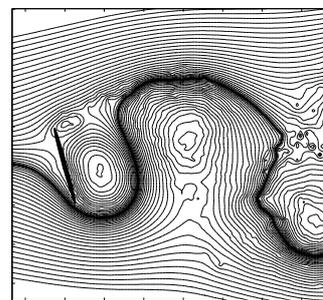
(a) $t^*=135, \theta=-28.8^\circ$



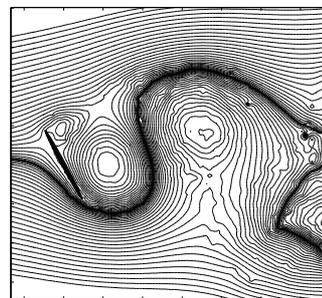
(b) $t^*=135.3, \theta=-21.2^\circ$



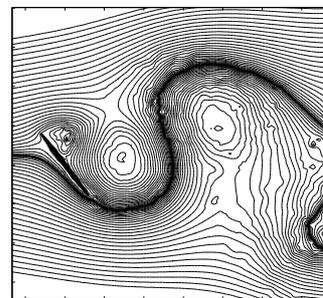
(c) $t^*=136, \theta=-9.1^\circ$



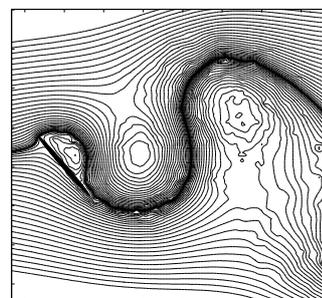
(d) $t^*=136.4, \theta=4.8^\circ$



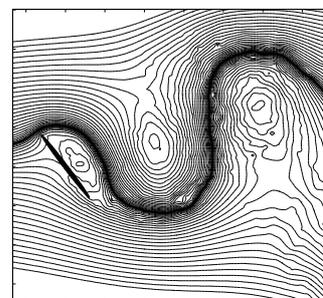
(e) $t^*=137, \theta=17.7^\circ$



(f) $t^*=137.4, \theta=26.9^\circ$

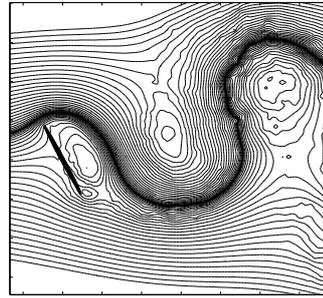


(g) $t^*=138, \theta=30.4^\circ$

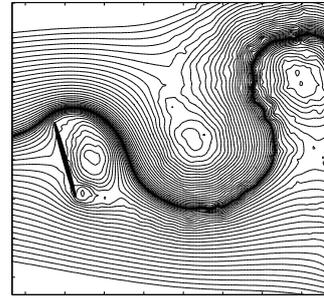


(h) $t^*=138.4, \theta=27.4^\circ$

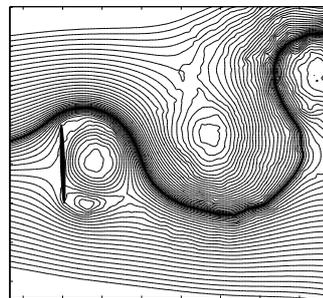
Figure 3.61: Streamline for the flow in phase *C* for $1/f_n^* = 7$ with $t^* = [135 \ 138.4]$ (see figure 3.60). The scale is in figure 3.55.



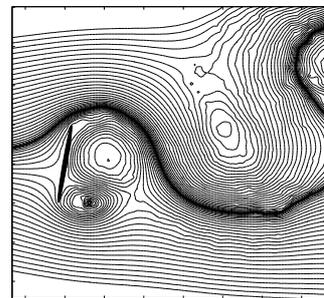
(a) $t^*=139, \theta=18.7^\circ$



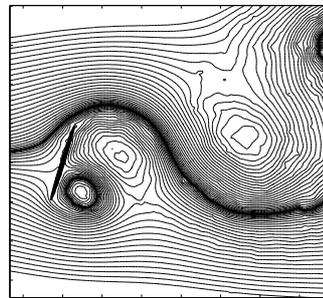
(b) $t^*=139.4, \theta=6.2^\circ$



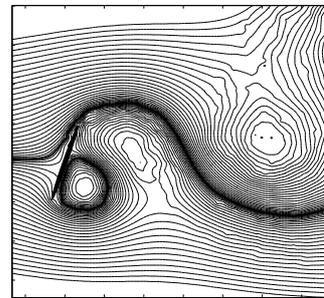
(c) $t^*=140, \theta=-7.3^\circ$



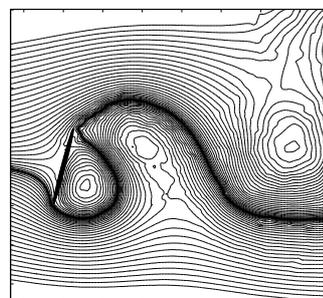
(d) $t^*=140.4, \theta=-19.2^\circ$



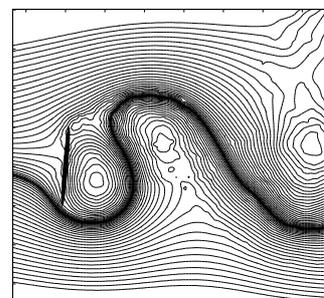
(e) $t^*=141, \theta=-27.2^\circ$



(f) $t^*=141.4, \theta=-29.4^\circ$



(g) $t^*=142, \theta=-25.1^\circ$



(h) $t^*=142.4, \theta=-15.5^\circ$

Figure 3.62: Streamline for the flow in phase *C* for $1/f_n^* = 7$ with $t^* = [139\ 142.4]$ (see figure 3.60). The scale is in figure 3.55.

From figures 3.60, 3.61, and 3.62, during phase C of the flow for $1/f_n^* = 7$, the flow and ellipse motion enter a time range where there is phase opposition between θ and C_m evolutions. This mode is actually specific to this time range, as it is temporary for this f_n^* value. Indeed, this mode is preceded by an increase of the amplitude value of $\theta(t)$, and is followed by a decrease of the amplitude of $\theta(t)$ due to a decrease of C_m values. Thus, the two values do not stay in phase opposition as the shedding frequency evolves with $\theta(t)$ oscillations amplitude. Another consequence is that it introduces a third harmonic (the other two being linked to f_n^* and the flow shedding frequency f_s^*) in $\theta(t)$ as the flow recovers after the θ amplitude decrease.

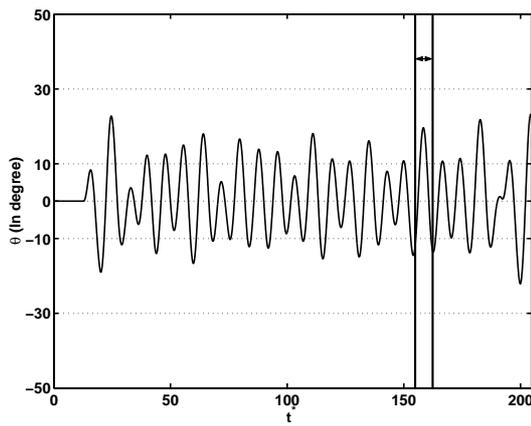
At $t^* = 135$ (figure 3.61(a)), an eddy has just been shed from the upper ellipse edge. On the lower edge, there is a growing eddy and the ellipse is beginning to move counterclockwise. It is then at $t^* = 136$ (figure 3.61(c)), that one can first see the effects of the high angular velocity. Here, the angular velocity is just prior to a maximum. Due to the angular motion, the shed eddy is slightly drawn toward the upper edge of the ellipse. The added circulation around the ellipse has the side effect of injecting more vorticity into the fed eddy. Afterwards, at $t^* = 136$ (figure 3.61(d)), another nascent eddy appears on the upper edge of the ellipse, while on the lower edge the eddy continues to be fed. The two eddies then grows while the plate is moving until $t^* = 138.4$ (figure 3.61(h)) where the lower eddy begins to be shed into the flow, at which point the angular velocity is at a minimum and the upper eddy continues to grow.

As the plate starts to move again, but this time clockwise, reaching a near maximum velocity at $t^* = 139.36$ (figure 3.62(b)), similarly to $t^* = 135.87$ a second nascent vortex is generated at the lower edge of the ellipse. Again the two vortices continues to grow simultaneously and while the upper eddy is mainly fed by the upper shear layer, the lower eddy is fed by the ellipse angular motion vorticity. As the plate slows down its motion, both vortices begin to be fed mainly by the shear layers. Again, when the angular velocity reaches a minimum value at $t^*=142.4$ (figure 3.62(h)), the upper vortex begins to shed.

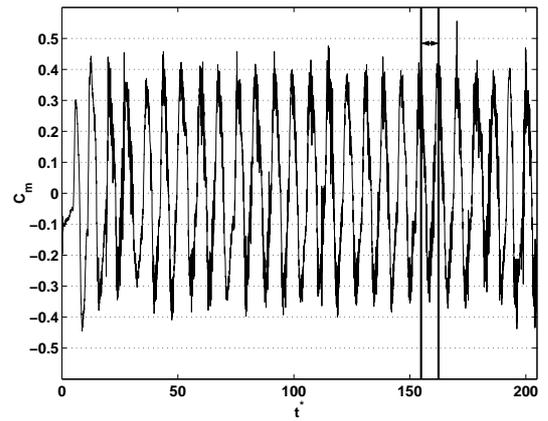
Note that the blob vortices introduce some local flow perturbations, as can be seen from the jaggedness in the contours, for example in figures 3.61(c), and 3.61(d). As explained in section 3.2.4.1, this can be attributed to the lack of overlap of the blob vortices.

One can thus conclude that for this value of $1/f_n^*$, there is a lock-in mechanism whereby the mechanical system and flow become coupled and the flow frequency adjusts to f_n^* through modifications of the vortex feeding and generation process. Typically, at a given timestep a primary nascent vortex is formed at one ellipse tip, and the plate is then rotated toward this nascent vortex. As the plate gains angular velocity, it creates another secondary nascent vortex at the opposite ellipse edge fed by the resulting ellipse rotation. The ellipse angular velocity also results in the weakening of the primary nascent vortex. Afterwards, the primary vortex is shed into the flow, and as the ellipse rotation slows down the secondary vortex now becomes the primary vortex and is fed by the shear layer corresponding to the ellipse edge from which it is shed.

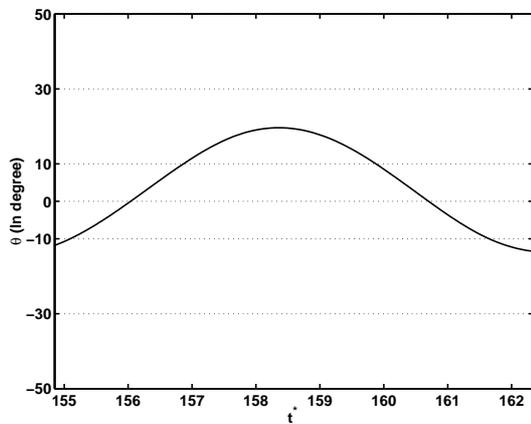
This phenomenon has two effects: first the vortex shedding is now faster due to the creation of this secondary nascent vortex, and second the aerodynamic moment coefficient is lowered by the secondary nascent vortex on the opposite edge of the primary shed vortex, and by the diminution of the vorticity feeding the primary nascent vortex. The ellipse angular oscillations are thus able to raise the flow shedding frequency through this secondary nascent vortex creation; there is also a mechanism limiting the flow shedding frequency, as if the shedding frequency is too high it lowers the aerodynamic moment coefficient considerably. This explains part of the flow vortex shedding synchronization mechanism. It means that for lower $1/f_n^*$ (compared to the obvious resonance point), the flow shedding frequency can be raised, one example is at $1/f_n^* = 11$. On the other hand, for higher mechanical frequency the flow shedding frequency can be lowered; this happens for $1/f_n^* < S$ in figure 3.66. Note that this mechanism is very dependent on the angular velocity attained in the early timestep.



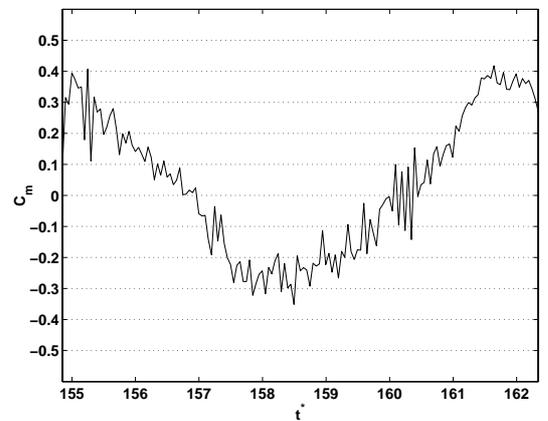
(a) θ range



(b) C_m range

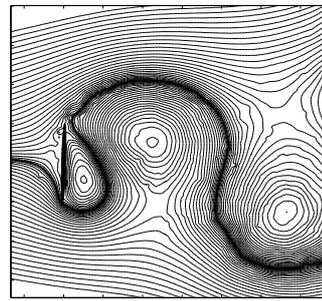


(c) θ range detail

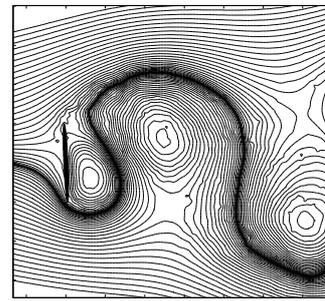


(d) C_m range detail

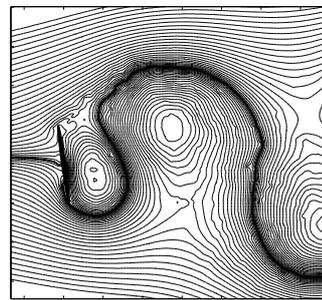
Figure 3.63: History of θ and C_m in phase C for $1/f_n^* = 11$



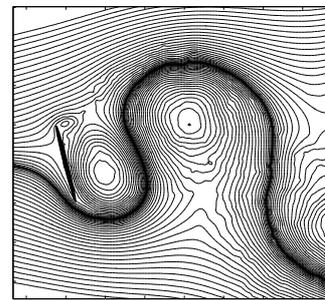
(a) $t^*=154.8, \theta=-11.8^\circ$



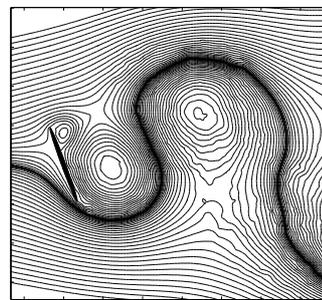
(b) $t^*=155.3, \theta=-7.7^\circ$



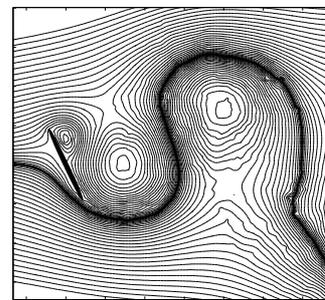
(c) $t^*=155.8, \theta=-2.3^\circ$



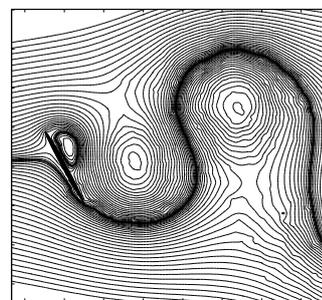
(d) $t^*=156.3, \theta=3.8^\circ$



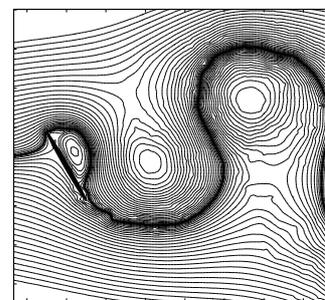
(e) $t^*=156.8, \theta=9.8^\circ$



(f) $t^*=157.3, \theta=15^\circ$

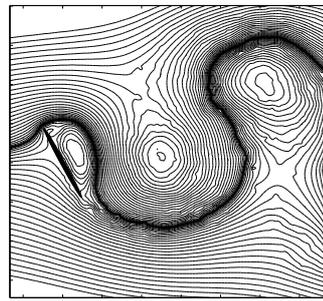


(g) $t^*=157.8, \theta=18.4^\circ$

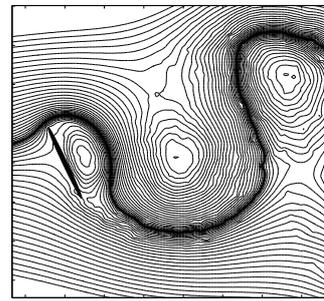


(h) $t^*=158.3, \theta=19.6^\circ$

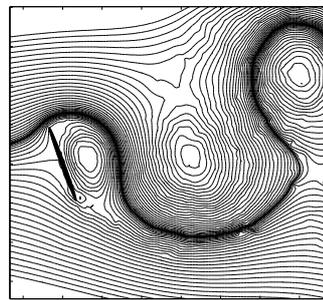
Figure 3.64: Streamline for the flow in phase *C* for $1/f_n^* = 11$ with $t^* = [154.8 \ 158.3]$ (see figure 3.63). The scale is in figure 3.55.



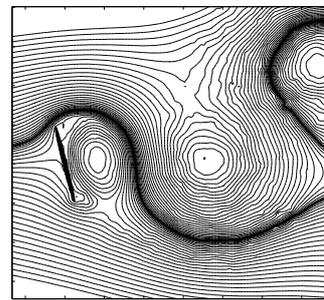
(a) $t^*=158.8, \theta=18.5^\circ$



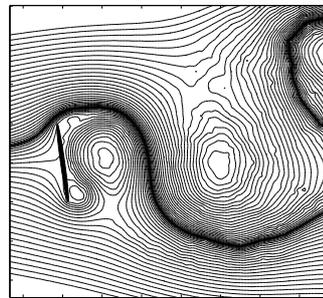
(b) $t^*=159.3, \theta=15.3^\circ$



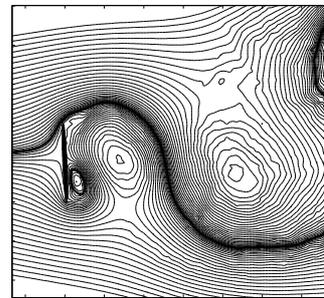
(c) $t^*=159.8, \theta=10.3^\circ$



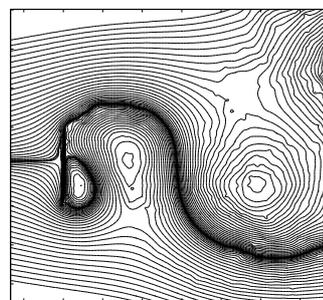
(d) $t^*=160.3, \theta=4.4^\circ$



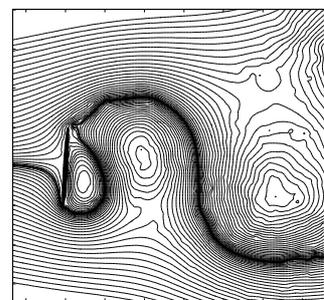
(e) $t^*=160.8, \theta=-1.7^\circ$



(f) $t^*=161.3, \theta=-7.3^\circ$



(g) $t^*=161.8, \theta=-11.3^\circ$



(h) $t^*=162.3, \theta=-13.4^\circ$

Figure 3.65: Streamline for the flow in phase *C* for $1/f_n^* = 11$ with $t^* = [158.8 \ 162.3]$ (see figure 3.63). The scale is in figure 3.55.

From figures 3.63, 3.64, and 3.65, during the phase C of the flow for $1/f_n^* = 11$, C_m and θ enter a time range where there is phase opposition. The process is similar to that of $1/f_n^* = 7$ presented previously. However, in contrast to the $1/f_n^* = 7$ case, the phase opposition is here a stable mode and the coupled spring-damped ellipse/flow system stays in this mode afterwards. It is thought that because the flow frequency is unable to adjust to f_n^* , the angular velocity does not reach a value sufficient to interfere too much with the shedding process as for $1/f_n^* = 7$. Still as the process is similar to that of $1/f_n^* = 7$, there is an adjustment in frequency as the vortex shedding process is slightly modified.

The sequence is started at $t^* = 154.8$ (figure 3.64(a)), when the angular velocity is very low and the eddy originating from the upper edge of the ellipse is about to be shed. Similarly to $1/f_n^* = 7$, at $t^* = 155.3$ (figure 3.64(b)) the eddy from the upper edge of the ellipse is shedding and one can see a small eddy on the upper edge of the ellipse as it rotates counterclockwise. Afterwards, as the lower nascent eddy develops and is fed by the lower shear layer, another nascent eddy is created on the upper edge of the ellipse at $t^* = 156.3$ (figure 3.64(d)). At $t^* = 159.8$ (figure 3.65(c)), this eddy continues to grow and another nascent vortex is created at the lower ellipse tip as the ellipse rotates clockwise and accelerates. The same process as for the counterclockwise rotation repeats until the lower nascent is shed at $t^* = 162.3$ (figure 3.65).

Therefore, from the $1/f_n^* = 11$ case, it can be deduced that the ellipse system is able to reach a stable state when a certain angular velocity (or more precisely a certain angular oscillation frequency) is reached. In this state, $\theta(t)$ and $C_m(t)$ are in phase opposition, and the ellipse motion then enables to generate nascent eddies at earlier times than in the fixed ellipse case. Two nascent vortices coexist at the same time, one primary nascent vortex feeding from the flow shear layer on one ellipse edge, and on the other edge a secondary nascent vortex initially fed in vorticity by the ellipse rotation. The limit seems to stem from the fact that as the ellipse is rotating, it is decreasing the amount of vorticity feeding the primary nascent eddy. The flow is then able to change its shedding frequency. However, as is signaled in section 3.2.3, there is an insufficient blob vortices overlap with the current parameters which renders difficult any assessment of the exact limit of this kind of mechanism and its influence on the flow shedding frequency.

Finally note that with this value of $1/f_n^*$, the spring-damped ellipse is more sensitive to C_m than for $1/f_n^* = 4$, which explains why the system is able to reach higher angular velocity. That is to say the mechanical system transfer function $H(s)$ has a lower value $H(0)$ in the case $1/f_n^* = 4$. Indeed, $H(0)$ (and thus the maximum amplitude of the spring damped ellipse) is proportional to $1/(J^*\varpi_n^{*2})$ as can be deduced from equation 3.16 in section 3.3. This dependence on the maximum reachable amplitude—and therefore the maximum angular velocity—also illustrates how sensitive the flow shedding frequency is to the angular velocity.

Now with figures 3.66, 3.67, 3.68, 3.69, for the various $1/f_n^*$ are plotted the flow shedding frequency (f_s^*), the maximum amplitude angle, and the mean and standard deviation, which is noted for convenience *rms*. Each item concerns the different forces and moment coefficients in order to present a summary of the spring damped ellipse/flow system. As for the standard deviation, the definition is:

$$rms(x) = \sqrt{\sum_1^N \frac{1}{N-1} (x_i - \bar{x})^2}, \quad (3.32)$$

where x is a member of the set of N values, and \bar{x} the mean value of x . Note that considering the amount of values in the set, the standard deviation definition is not very important.

Figure 3.66 provides a summary of the system response. One can see a transition occurring around $1/f_n^* \simeq 6.8$ (equivalent to the experimental Strouhal number), where the flow frequency f_s^*/f_n^* and the maximum amplitude angle θ_{max} abruptly increase. Afterwards, θ_{max} starts to decrease, while f_s^*/f_n^* increases more steadily. In the range $1/f_n^* = [6.8, 7.5]$, there seems to be a lock-in between the flow frequency and the spring damped plate natural frequency, even if f_s^* is slightly higher than f_n^* . It is more notable with θ_{max} , as it stays at near a maximum value in this range and then decreases. Recalling that the flow shedding frequency is $1/f_s^* = 6.45$, it is not surprising to see that the lock-in range is close to the spring-damped ellipse resonance frequency. As in section 3.2.3, the discrepancies in f_s^*/f_n^* can be attributable to the simulation errors when integrating equation 2.26 with too few blob vortices, which makes it difficult to predict the influence of the simulation on $f_{n,s}^*$.

Generally, figures related to the force coefficients and flow shedding frequency (3.66,

3.67, 3.68 and 3.69) generally show a sharp increase around $1/f_n^* = 6.8$, suggesting a resonance. Blevins [7] found also large amplitudes for transverse oscillations of a cylinder in a uniform freestream flow and interestingly a narrower lock-in band than in experiments (see also section 3.2.4.2). He attributed it to the phase between the structural motion and the flow simulation which does not model accurately the synchronization effect. It can be conjectured that there is a similar effect here, as well as errors due to the lack of overlap of blob vortices which impede the lock-in band.

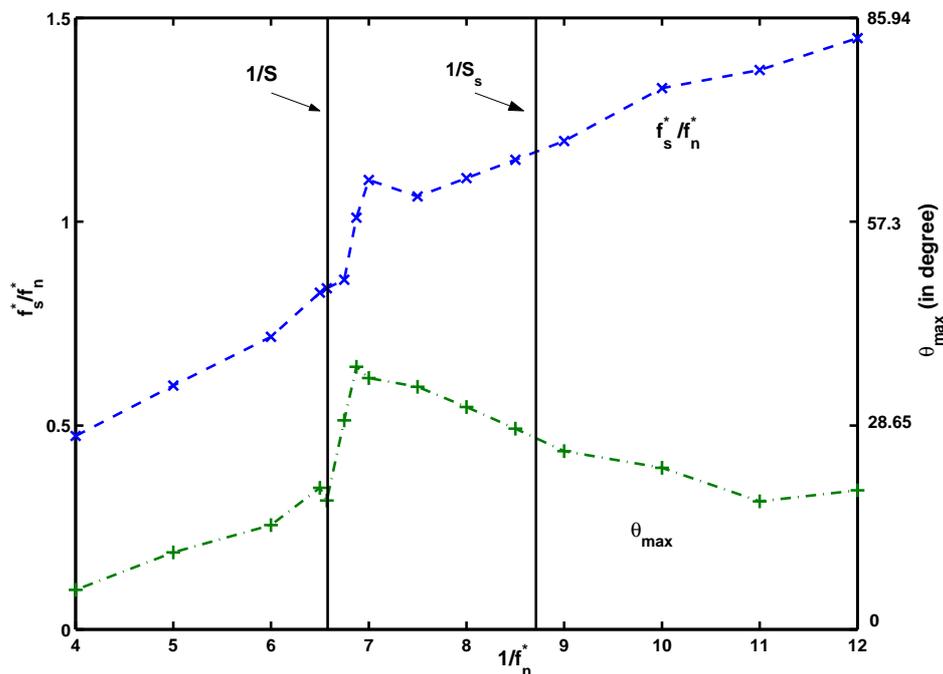


Figure 3.66: Summary of the simulations detailing the variations of the shedding frequency f_s^* and the maximum amplitude of angle oscillation regarding the nondimensional natural frequency f_n^*

As is also apparent from table 3.6, at $1/f_n^* = 4$ the end f_θ^* (f_θ^* for the last PSD range) is very close to the Spalart simulation Strouhal number S_s for a fixed ellipse at $\alpha = 80^\circ$. Conversely, for $1/f_n^* = 7$, $f_\theta^* = 0.155$ which is the same frequency as found at an earlier timestep, and equally very close to the experimental Strouhal number for a steady plate at $\alpha = 80^\circ$ ($S = 0.154$). Note that the end f_θ^* generally coincides with the flow frequency f_s^* as seen in the PSD plots (figures 3.51, 3.52, 3.53, 3.54). Changing the motion starting time of the ellipse at $t^* = 102$, the flow frequency prior to the motion is now $f_s^* = 0.118$ and the coupled system (spring-damped ellipse and flow simulation) shows the same pattern with a lock-in around $1/f_n^* = 7$ but a lower end f_s^* at around

0.139 with $\theta_{max} = 0.63 \text{ rad} = 36^\circ$.

Aside from the lock-in problem, figure 3.66 also implies the end flow frequency f_s^* is linked to θ_{max} with the lower the θ_{max} , the closer f_s^* is to S_s . On the other hand, the higher the θ_{max} , the closer f_s^* is to S . Also recall that it has been observed in sections 3.3.1 and 3.2.4.1 that the ellipse angular velocity influences the numerical flow behavior and the coupled spring-damped ellipse/flow system vortex shedding frequency.

Consequently a transition point around $1/f_n^* = 6.8$ suggests that the maximum ellipse angular velocity during the ellipse angular oscillations influences the lock-in region as well as the end value of f_s^* , whereas the flow shedding frequency when the plate motion is initiated influences the end f_s^* . However, given the simulation sensitivity to the angular velocity, it is difficult to generalize this analysis to real flows without further experimental results.

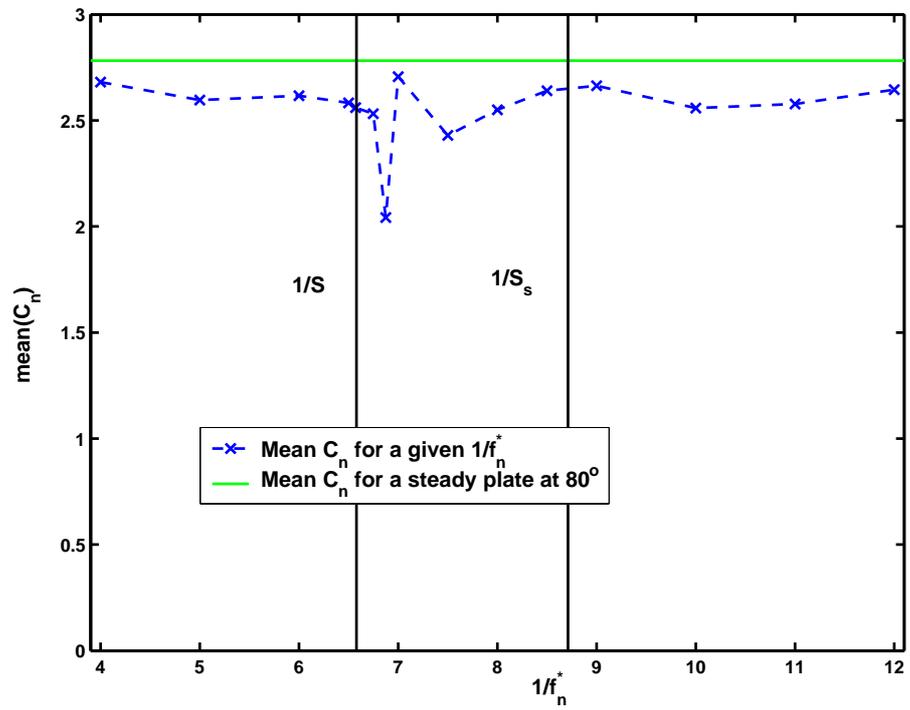
Concerning the force coefficients $C_t(t)$ and $C_n(t)$, from figures 3.67(a) and 3.68(a) one can see that their mean values are lower than that of a plate with steady incidence angle, while the aerodynamic moment $C_m(t)$ in figure 3.69(a) is higher in most cases except around $1/f_n^* = 7$ (at about the same value it is changing sign). Conversely the *rms* of $C_t(t)$ and $C_n(t)$ (figures 3.67(b) and 3.68(b)) are higher than for the steady plate case, while the moment *rms* is lower (figures 3.69(b)). For the C_t and C_m , their mean coefficient values are always much lower than their *rms*, which suggests that they are less representative than the standard deviation and moreover that the flow dynamics remains in vortex shedding mode.

In the absence of vorticity plots, it is difficult to assess what has been the exact influence of the flow. However, from the general evolution of the aerodynamic forces and moment coefficients, one can deduce that at all times there is a vortex shedding, and that the forces are less dependent on $1/f_n^*$ than the moment. If confronted with the θ_{max} values (figure 3.66), these results mean that generally higher fluctuations imply a lowered moment coefficient compared to the fixed ellipse case. This effect is due notably to the phase opposition between $C_m(t)$ and $\theta(t)$ which takes place with the rotating ellipse oscillations as exposed previously. Note that the fluctuations past $1/f_n^* = 6.8$ in the C_t *rms* plot 3.68(b) can be attributable to the vorticity change at both tips of the ellipse as rotations occur. Indeed, as already presented in section 4.2, as the plate rotates

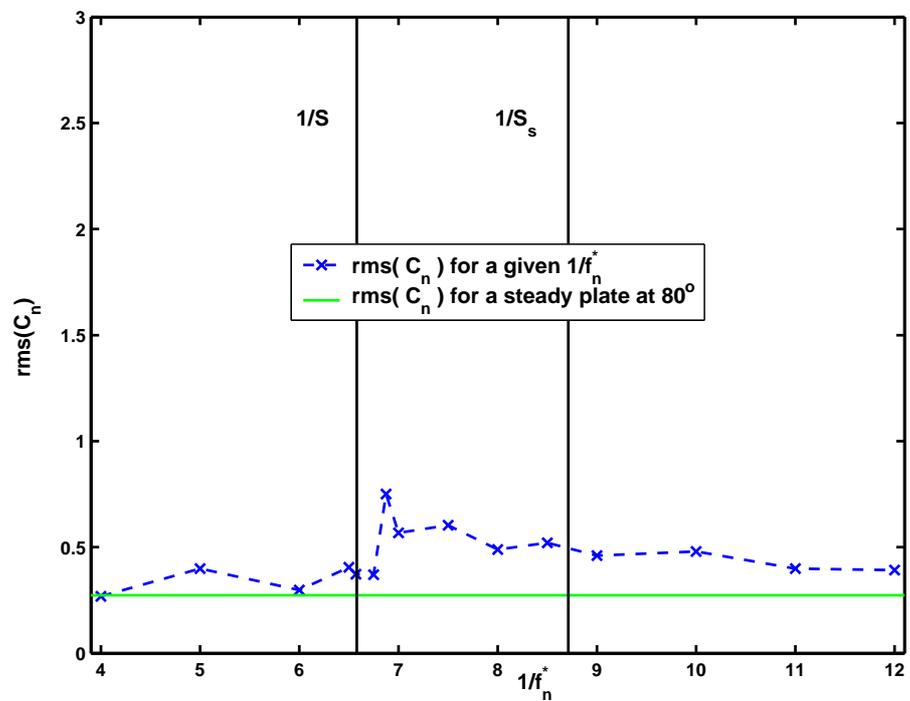
the boundary blob vortex strength increases at one tip and decreases at the other tip. Using Spalart formula for forces calculation (2.45), one can see that this increase in C_t *rms* is a consequence of the increase in the oscillation amplitude.

Generally, due to the spring-damped ellipse/flow interactions the aerodynamic moment coefficient magnitude range is lowered, while it raises the range of the aerodynamic drag and lift coefficient. The mechanism of diminution of the moment has already been explained. It seems that the drag (C_t) and lift (C_n) coefficients are raised through the presence of a secondary nascent vortex as well as the circulation around the body induced by the angular rotations.

In conclusion, variations in $1/f_n^*$ mainly influence the frequency characteristics of the flow with a flow having one or two harmonics while retaining a vortex-street flow regime. Therefore, not much difficulty is expected in stabilizing the spring-damped ellipse angle using a fuzzy logic controller (as it has shown to be very robust). However, it could prove more problematic for the optimal controller, as the controller implementation is reliant on the system characterization, which here is based on a stable system. These topics are taken in next chapter.

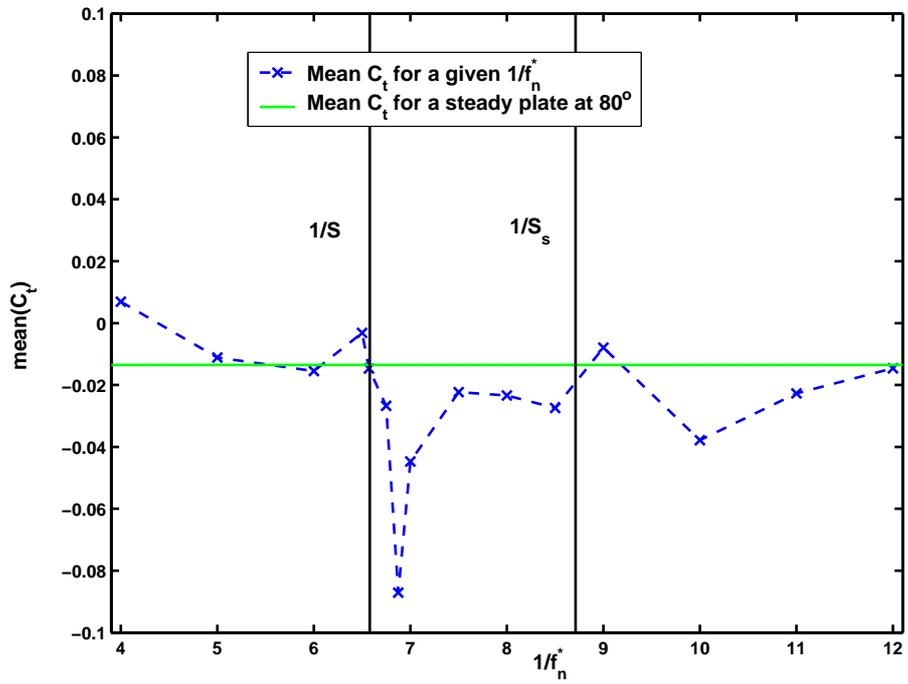


(a) Mean value for C_n

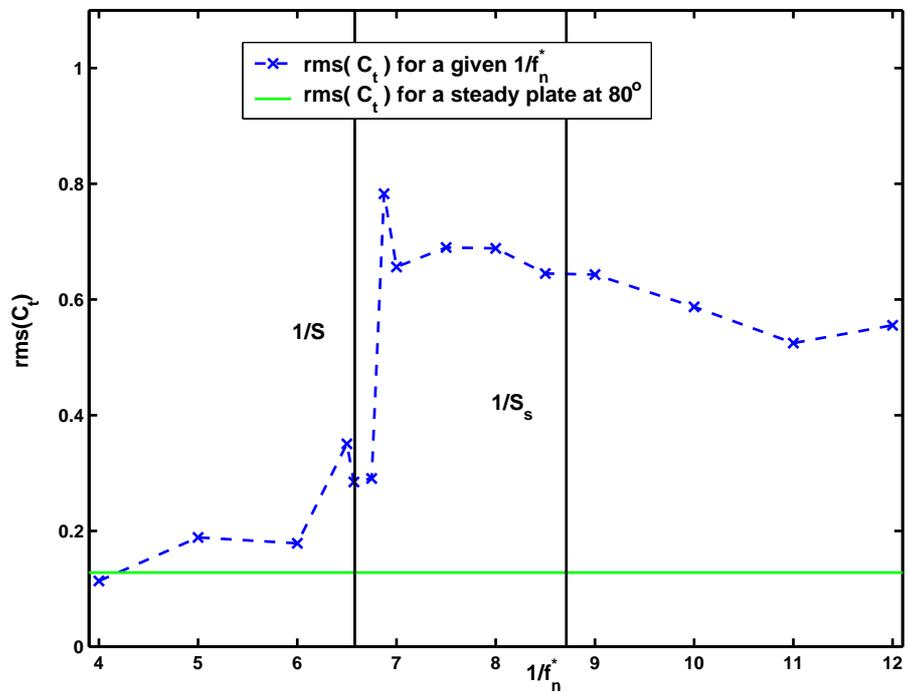


(b) Root mean square for C_n

Figure 3.67: Mean values and root mean square of C_n depending on $1/f_n^*$

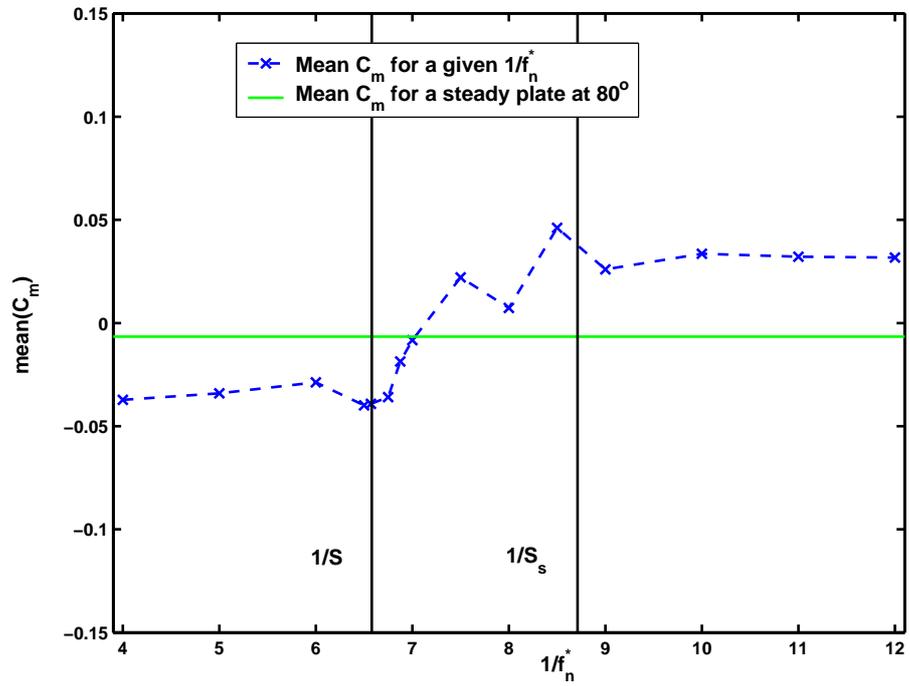


(a) Mean value for C_t

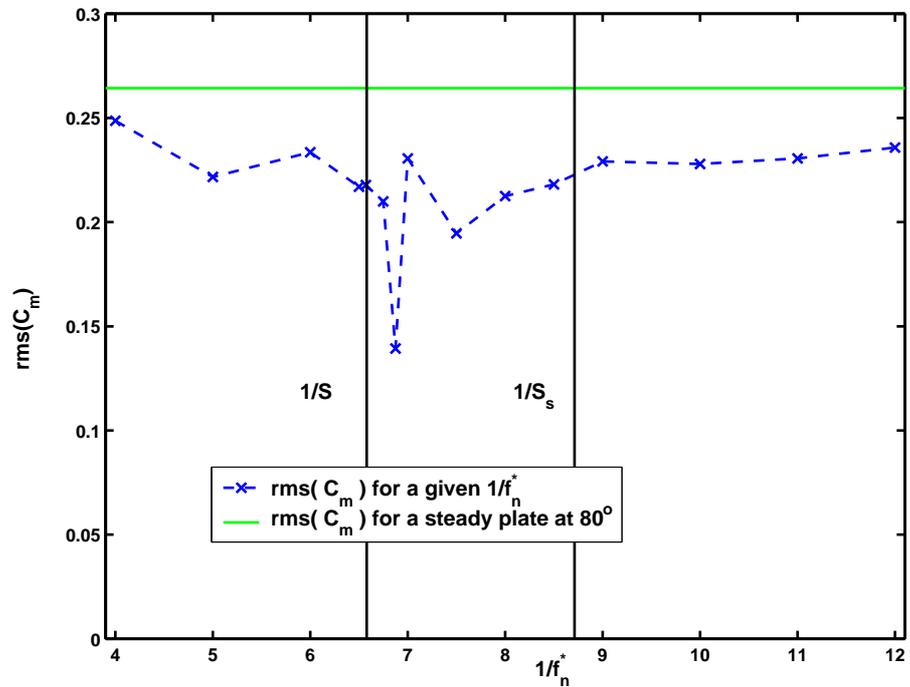


(b) Root mean square for C_t

Figure 3.68: Mean values and root mean square of C_t depending on $1/f_n^*$



(a) Mean value for C_m



(b) Root mean square for C_m

Figure 3.69: Mean values and root mean square of C_m depending on $1/f_n^*$

3.4 Summary

Through this chapter, the blob vortex simulations have been studied against various test cases for a given set of parameters in section 3.2 and shown to converge broadly for the plate in a translating crossflow (section 3.2.3 and 3.2.4.1)). Indeed, one conserves the broad flow features at least qualitatively to the flow parameters considered, although there is an overestimation of the forces coefficient and an underestimation of the Strouhal number. Nevertheless, section 3.2.3 has also presented some of the difficulties in adjusting properly the simulation parameters in order to obtain converged results and the necessity for a high number of vortices.

The introduction of section 3.2.4 has explained the inadequacy of the Sarpkaya-like code. Thus, any simulation made with a moving plate uses the code derived from Spalart. Afterwards, using the same set of simulation parameters, the cases of the vertically oscillating cylinder has shown that the Spalart simulation is able to reproduce a lock-in, albeit in a narrower band than for experimental results (section 3.2.4.2). Finally, section 3.2.4.1 has mainly shown that the modified Spalart code behaves consistently to other numerical $2D$ simulations. Again, it cannot be stressed enough that for this kind of method, a proof of the convergence is difficult to establish for a given set of parameters.

In the second part of this chapter in section 3.3, the case of the oscillating plate in a uniformly translating crossflow has been studied through different cases after defining some study parameters. The case of a plate allowed to oscillate in rotation with no freeflow has been simulated in section 3.3.1 in order to assess the influence of the simulation but also to investigate the flow reaction. It shows that though the flow dynamics introduce some additional damping, the numerical simulation has been seen to also add some further modifications. Namely there is introduction of a flow added mass which diminishes f_n^* , as well as a nonlinear ξ^* which is dependent on the angular velocity of the plate. Concerning the influence on ξ^* , the evolution is due not only to the flow added mass but also the simulation effective viscosity. Other trials have shown that non zero U_∞ introduces further modifications to the flow/plate system.

Finally, the plate allowed to oscillate in rotation in a uniform translating crossflow has been studied first considering the influence of the reduced damping (section 3.3.2.1).

This section has shown that the spring damped plate reduced damping was not one of the main parameters for the flow reaction within the range chosen and has rather shown the influence of f_n^* . Then the reduced resonance frequency (section 3.3.2.2) has helped assessed the behavior of the coupled flow/plate system. It has shown that variations in $1/f_n^*$ mainly influence the frequency characteristics of the flow with a flow having one or two harmonics while retaining a vortex-street flow regime. Therefore, not much difficulty is expected in stabilizing the spring-damped ellipse angle using a fuzzy logic controller. Although, it could be more problematic for the optimal controller, as the controller implementation is reliant on the system characterization, which is based on the plate in a uniformly translating crossflow in chapter 4.

Chapter 4

Plate and flow system modeling

4.1 Outline

Chapter 3 has shown in section 3.2 that the flow simulation converges broadly toward the right parameters using different tests cases (fixed plate in a uniformly translating crossflow, vertically oscillating cylinder in a crossflow, rotating plate in a translating crossflow).

In particular for the fixed plate in a uniformly translating crossflow (section 3.2.3), it has also enabled to remind of the behaviour of the forces coefficient as the angle of attack rises. As the angle increases, the forces coefficient level and main harmonics are influenced but their function in time remain quite similar from one angle to the other.

In sections 3.3.2.1 and 3.3.2.2, it has been shown that it was mainly f_n^* which was influencing the simulation behaviour and that the flow forces coefficients presented a rather consistent behaviour when f_n^* with mainly two harmonics in their spectrum. This brings here to the flow/structure interaction.

The main aim in this chapter is to proceed to a flow identification in order to enable the control design especially for the classical and optimal controller in section 5.2.3 and 5.2.4. In section 4.2, an overview of the flow/structure interaction and modeling is first provided to complement the data. Afterward, in section 4.3.1 the strategy is presented while the fuzzy logic theory is given in section 4.3.2. The general flow model for a fixed plate at a given angle of attack is then described in section 4.3.5. Finally, while section 4.3.6 provides the model for the spring damped plate, section 4.4 gives a short study of the couple flow/plate properties.

4.2 Introduction

Interactions between a flow and a moving bluff body are of interest because of the need to understand and predict the behavior of such complex coupled systems. There are many examples where such a need rises for structures with a circular or otherwise bluff cross section such as a cable. The prediction of the behavior of such a structure is hindered by the nonlinear nature of the flow around the bluff body, and the way it is affected by the reaction of the structure.

Many studies involving interaction between an oscillating body and vortex shedding have been achieved for the case of transverse oscillations of a $2D$ cylinder. Considering a uniform flow, the cylinder is allowed only to move in the crosstream direction. The motion of the body is modeled as a spring and viscous damper system. On the other hand, little has been carried out on torsion reaction to the freestream flow, although in some cases the flow induced moment on the body cross-section becomes non negligible. Therefore this study involves the control of flow/structure vibrations along an axis of rotation, with a body considered to be of infinite stiffness fixed on an axis with a torsional spring and damping, as illustrated in figure 3.38. The plate was chosen (modeled here as a thin ellipse) because in the case of a cylinder rotation, the moment mainly derives from the skin friction (i.e. dependent on viscosity). Conversely, in the case of the flat plate, the moment derives primarily from the pressure forces. As the pressure is strongly related to the vortices strength, the interaction between shed vorticity and body rotation is stronger than with the cylinder. This distinction between circular and non circular bodies does not exist when considering the translational motion of a body. Furthermore, the plate problem is much cleaner and less dependent on the Reynolds number than the cylinder.

From section 3.3, the equation for the rotation of a two dimensional ($2D$) body is:

$$J\ddot{\theta} + \mu\dot{\theta} + k\theta = M, \quad (4.1)$$

with J the angular inertia of the body, μ the torsional damping, k the torsional spring coefficient, and M the flow induced moment on the body. One advantage of the ellipse is that when one uses $b = a$ (cf figure 2.1) the ellipse becomes a cylinder, and conversely when $b \ll a$ the ellipse is similar to a flat plate. Thus, a large range of geometry is available.

In addition to the flow analysis in section 3.3, on the basis of equations (2.68) and (2.31), it is possible to produce a qualitative analysis of the aerodynamic forces and moments induced by a rotating body. With this algorithm, this means that in a steady flow with constant incidence angle, if the thin ellipse rotates, at the next time step one can expect a decrease in absolute value of the vorticity shed at one tip and an increase at the other tip of about the same magnitude due to the rate of introduction of vorticity mechanics described in sections 2.3.4.1 and observed in section 3.3.2.

Then using the different aerodynamic forces and moment formula, and, as a first step, Blasius extended formula (equations (C.8) and (C.55)) and their subsequent development (equations (C.28), (C.45), (C.49), (C.54) for the forces, and equations (C.59), (C.60), (C.67) for the moment), one can deduce that the closer a shed blob vortex, the more important it is to the force magnitude, and the same applies to the moment magnitude; in other words the influence of a blob vortex decreases as the inverse of its distance to the body surface. Also looking at Wu's formula and especially to the forces equation (2.65) (see also Appendix D), one can see that the rotation has no direct effect on the forces, as the second term in the equation is null for a pure body rotation. This result still holds for a translation combined with a rotation. In fact, only the aerodynamic moment is directly influenced by a rotation, as can be deduced from the second term in formula (2.66). These still hold for Spalart's Formula as both are derivable from one another.

Using these various formula and from the previous remarks, one can conclude that at times just past the motion the forces do not change significantly, and the moment shows much stronger variations than for a nonrotating plate because the rotation term does not influence the aerodynamic forces formula as opposed to the moment. Consequently, forces are influenced only by the change of flow pattern as the body moves in the flow. However in the C code, this mechanism is limited by the fact that the vorticity of the ellipse at low angular velocity is shed only from its tips, as opposed to the Spalart code where the vorticity is shed uniformly from the body surface. A similar analysis for a vertical motion of an ellipse at low angle of attack would reveal a similar action on the vertical component of the force and thus an analogy can be drawn with the moment introduction provided that one consider the wake as not too affected.

The flow reactions to a bluff body motion can be predicted only with difficulty. As little work exists for such an oscillation, extrapolation can be made from the vertical oscillations of a cylinder in order to assess the approach used here. As emphasized by Shiels (1998) [56], most studies attempt to obtain an analytical model of the behavior of the resulting aerodynamic forces from the body motion. Therefore, attention has focused on extracting models from experiments and simulations. Unfortunately, the sheer number of spatial and dynamic parameters and the number of possible combination hinders the definition of a model usable for the prediction of flow characteristics. In the case of the translational oscillation of a cylinder, some have tried to develop a mathematical model for the wake prediction, as described in the review by Sarpkaya (1979) [51]. However these are limited to a precise motion and for a precise geometry, and cannot be used elsewhere.

One can distinguish two main approaches for the study of the coupled system. The first one is characterized by using forced body motion to attempt to predict the behavior of similar free-body motion. Forced motion can be a satisfactory way of evaluating fluid-dynamic damping/excitation when the response of a system is periodic, however with more complex response (modulated or transients occur) it is not satisfactory. Indeed in that case, such an attempt is limited because it neglects the link between the body motion and the flow dynamics.

The second approach, which is followed in the current work, is to directly study the coupled fluid-structure system through experiments or simulations (through section 3.2.4 notably). Again, this limits the validity of the predictions to a specific configuration. Indeed, the force and moment responses are particular to a motion type. This is visible for example, if one looks at the extended Blasius' theorem (Appendix C), where one can see the effect on the force is different whether the motion is a translation or a rotation. Of course, this is also dependent upon the body geometry and freeflow.

Using the latter methodology, Feng (1968) [18] described in a set of experiments the behavior of the transverse oscillations of a cylinder for a given set of parameters at low Reynolds number ($Re \approx 200$). He found an hysteresis in the flow frequency depending on the cylinder's natural frequency as well as a lock-in phenomenon whereby the flow frequency matches the natural frequency of the cylinder. Some explanations have been proposed, for example by Sarpkaya 1979 [51], who suggested a mechanism where the

vorticity at the tip exposed to the “leading edge” of the cylinder moving upwards becomes more important through velocity orientation and the cylinder proximity. In the meantime, as the cylinder moves upward, the influence of the motion on the velocity direction moves the upper separation point further downstream, and conversely the lower separation points moves further upstream compared to a stationary cylinder. Finally, he observed that the rates of vorticity at the lower and upper separation points do not reach respectively their maximum and minimum simultaneously as for a stationary cylinder.

Thus on the edge toward which the cylinder is moving, the vorticity shed in the flow becomes higher more quickly than on the other edge. Furthermore, the instantaneous net circulation around the cylinder increases as a result of the motion. The net effect is that the lift tends to synchronize with the cylinder motion. The absence of synchronization is then said to come from intrinsic flow characteristics, for example a high fluid inertia may damp the increase in the vorticity rate. However, Shiels (1998) [56] pointed out that these results on the cylinder motion were only part of the parameter space, as illustrated by the much wider range of cylinder response observed by Gharib et al (1997).

Early studies such as Hartlen and Currie (1970) [24] offer a mathematical model, using for example a van der Pole-type oscillator. However, these are limited to the case of the cylinder oscillation, and even then the model lacks generality for other types of flow. Thus they fail to obtain the functional relationship between basic parameters which could provide a better understanding of the phenomenon.

A slightly different approach was chosen in modeling the effect of a moving body, namely a ‘black-box’ model based on linearization of the aerodynamic moment, with additional effects due to rotation and oscillation introduced through interpolation between different linear models. These latter models being defined through identification methods.

This introduces complications in the model because it is difficult to assess the effect of real time interpolation from one function to another as they evolve in time, all the more since each function interacts differently with the body motion. This problem can be overcome by choosing smooth functions of similar shape. Nevertheless, it is expected that the use of different interpolation functions will help to generalize the method to other kinds of flow. The idea is then simply to use the superposition of functions to

predict the response from the flow and thus to predict oscillating behavior. This has the advantage of breaking down the complicated physics into simpler more manageable functions. This is somewhat similar to a Takagi-Sugeno or a gain-scheduling approach.

The main aim of this project is then to be not only able to assess the aerodynamic effect of such oscillation on the crossflow, but also assess how a simplified flow model (rather than the simulation) could help in designing a controller in order to control the motion of the cylinder/ellipse. Therefore, using a simple linear flow model based for example on the angle of attack, one can now use classical technics such as pole positioning or optimal controller theory in order to stabilize the plate. Such methods can be compared to adaptive methods such as the model reference scheme.

As a last point, the use of a coupling between a controller implementation in Matlab with the vortex simulation in C shall facilitate the validation and development of the method. This enables to check directly the viability of the controller. Thanks to the computing time due to the vortex simulation (with the Spalart code used in chapter 3), the cost is minimal on modern computers. It avoids doing experiments, although the precision is limited during the controller design phase by the number of interpolation functions used by the simplified flow model.

It was thus planned to proceed as in figure 4.1 to develop the control, that is first identify a model of the transfer function of the flow dynamic with the moment as an output and the plate position as an input. Then, the controller development will be continued. And, finally the link between Matlab and the flow simulation will be implemented in order to assess the control validity.

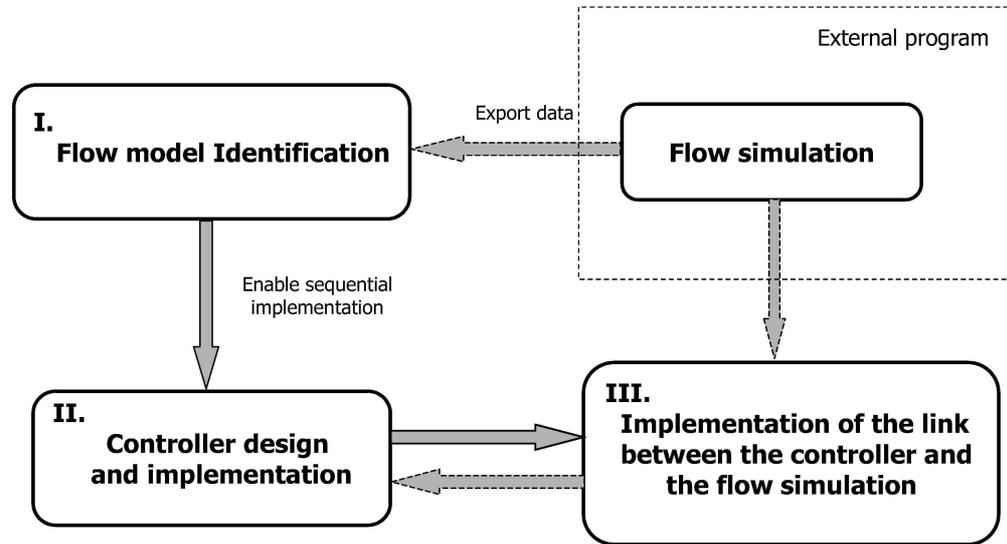


Figure 4.1: Plan for the system modeling

4.3 Flow model identification

4.3.1 Strategy for the identification

For this implementation, the emphasis is put chiefly on devising a model appropriate for the flow at steady angle of attack, and then on focusing on a flow about a rotating plate. Nevertheless, the two objectives are not incompatible since for small oscillation amplitude, around a fixed angle of attack one can consider the evolution due to the oscillation to be caused by an additional unsteady term. Finally, the model is to stay valid for a limited time period as it will be applied for the controller validation.

As the main aim of the controller is the stabilization of the plate, during a first step the angle of attack α was chosen as the input for the signal, and the moment coefficient C_m as an output. Of course, other input vectors are possible and could be desirable considering the nature of the fuzzy logic box which for better efficiency demands for a given input value, a unique output value (similarly to a bijective function). It means that one must not attempt to model an hysteresis with a unique input and a unique output. For instance, one could use the angle of attack as well as its derivative as inputs if there is an hysteresis in C_m , depending upon whether the plate is pitching up or down. Furthermore, the angle of attack is a very convenient variable because it is

easy to extract and more representative of the state of the system compared to other variables such as static pressure at given points, for example.

The signal is considered to be deterministic, i.e. without previous characterization of the noise introduced by the simulation onto the response. Indeed, this noise depends not only on the time step chosen for the simulation, but also on the angle of attack, as the system is fundamentally nonlinear, thus impairing the characterization of the signal. Given the model input and output, identification had to be done on the transfer function characterizing the relationship between the C_m response to an angle input α . Then the data from the simulation are used and the angle of attack function as input. Obviously, the sampling time is taken as the timestep of the simulation. These data are then used to obtain a frequency response through a fast Fourier transform (FFT) of the signal. However, the frequency resolution is hindered by the sampling frequency.

As stated in the previous section, the intent is to use several base “blackbox” functions, and then fuzzy logic boxes to interpolate them according to the system kinetic state, keeping the angle of attack α as an input. In extracting the model one make the assumption that there is a low angle oscillation, and thus it is acceptable to take as base functions for the interpolation the transfer function from the input α to the aerodynamic moment C_m at a given angle, so that $C_m = f(\alpha, t)$. Note that model nonlinearities can also be introduced through introduction of nonlinearities in the input, which enables the flow model to be fully linear; however this is not sufficient in this case where the model frequency response evolves depending on the angle of attack; e.g. considering the output to be a sine function for a step input, then depending on the end value of the step the frequency of the sine wave will change.

From the simulations results (section 3.2.3) and from Fage and Johansen’s (1927) [16] experiments, it has been shown that for a flat plate in the range of angle of attack 45 to 90 degrees, the aerodynamic forces and moment magnitude and frequency are dependent solely upon the angle of attack. The drag increases with the angle of attack, and the lift decreases. The frequency increases with the angle of attack; however as pointed out by Fage and Johansen (1927) it is mainly dependent on the projected area “seen” by the freeflow, which is proportional to $\sin(\alpha)$. Therefore, the system is nonlinear because the time and space scale of reference changes with the angle of attack. Hence the idea of implementing an interpolation based on both C_m and the shedding

frequency providing an interpolation not only in magnitude but also in frequency in order to avoid simply having a superposition of the base functions mentioned above.

Figures 4.2 and 4.3 provides an illustration of such an interpolation for a simplified case as well as a comparison with the magnitude interpolation alone. As an example, y is denoted as the output of a process, x the input of a process and f the transfer function of a process.

In figure 4.2, two transfer functions $f_1(x)$ and $f_2(x)$ are used which are actually process output y models for constant inputs at two different levels x_1 and x_2 respectively. At x_1 and x_2 , the transfer function f_1 and f_2 produce the output Y_1 and Y_2 respectively.

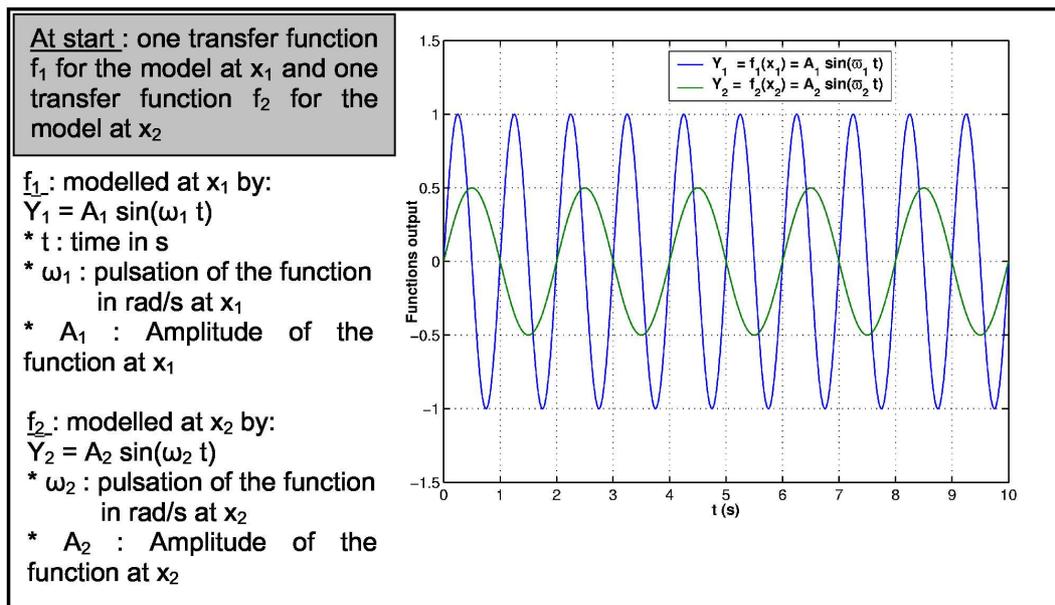


Figure 4.2: Starting transfer functions for the magnitude and frequency interpolation in figure 4.3.

Now in figure 4.3, a resulting output from these two transfer functions is given at x_r . Note that for simplicity sake, a constant input is used.

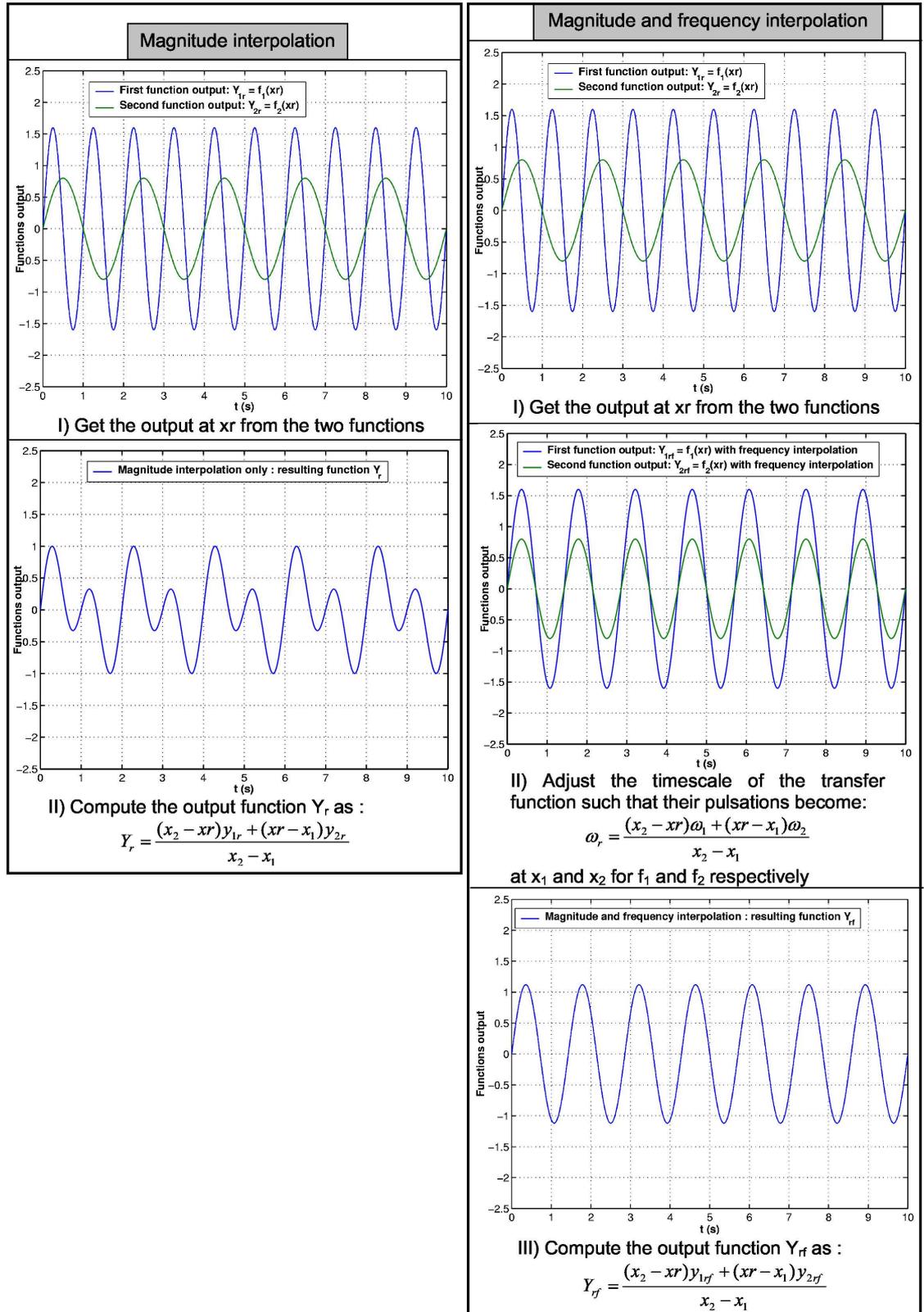


Figure 4.3: Magnitude interpolation only compared to magnitude and frequency interpolation for a constant input x_r . Functions f_1 and f_2 are presented in figure 4.2.

Through this example has been applied for sine functions, it is possible to do the same for a C_m model identification. Using such frequency interpolation, one can build a simple $C_m(\alpha, t)$ model for a plate with a constant angle of attack at a low computing cost, and using only simulation data with a limited number of angle of attack (see sections 4.3.4 and 4.3.5). That is to say, if one wants to obtain a model of the C_m with steady angle of attack, the simplest would be to proceed using an interpolation in magnitude of various $C_m(\alpha, t)$ model identified for a given angle of attack. In that case, the number of angles of attack would be large in order to limit the influence of the magnitude interpolation when α is at a value between two identified models. The frequency interpolation then enables partially to raise this kind of issue.

In order to implement the frequency interpolation, one way would be for a given α , such that $\alpha_1 < \alpha < \alpha_2$ with α_i the angle of attack of a given $C_{m,i}$ model, to compute the corresponding shedding frequency (which for a plate at steady angle is equivalent to C_m main frequency), and then implement for the $C_{m,1}$ and $C_{m,2}$ models a change of frequency such that they both have this desired frequency. The calculus used to implement this change in the C_m models frequency is shown in section 4.3.3.

Remark that the frequency change mainly consists in a individual change of timescale for the modeling transfer functions f_1 and f_2 . It is possible here because each transfer function can be seen as narrow band-pass filter, therefore the model dynamics can be defined using a single frequency per transfer function. However, with more complex motion, such model would no longer be appropriate. Because the main aim is to model the resulting C_m for a stable plate, i.e. with small variation of the input, this modeling is still applicable.

This mechanism of frequency adjustment can also be used to model a frequency lock-in, as one only has to change the interpolation functions for the model frequency. As a downside, it implies in return that one make extensive study across the parameter space to be sure to accurately capture this phenomenon. It has then not been used here.

Here is not taken into account the transition part at the beginning of the simulation, because in the case of the vortex method it is dependent upon the time step chosen. C_m is not monitored over the entire simulation time, but only when the simulation stabilizes, i.e. when $T^* > 5.8$ (using here $T^* = T \times a/U$). For the Spalart code, this is the

time required for the simulation to reach the vortex shedding, that is a quasi periodic stationary state. Figure 4.4 illustrates the time range used.

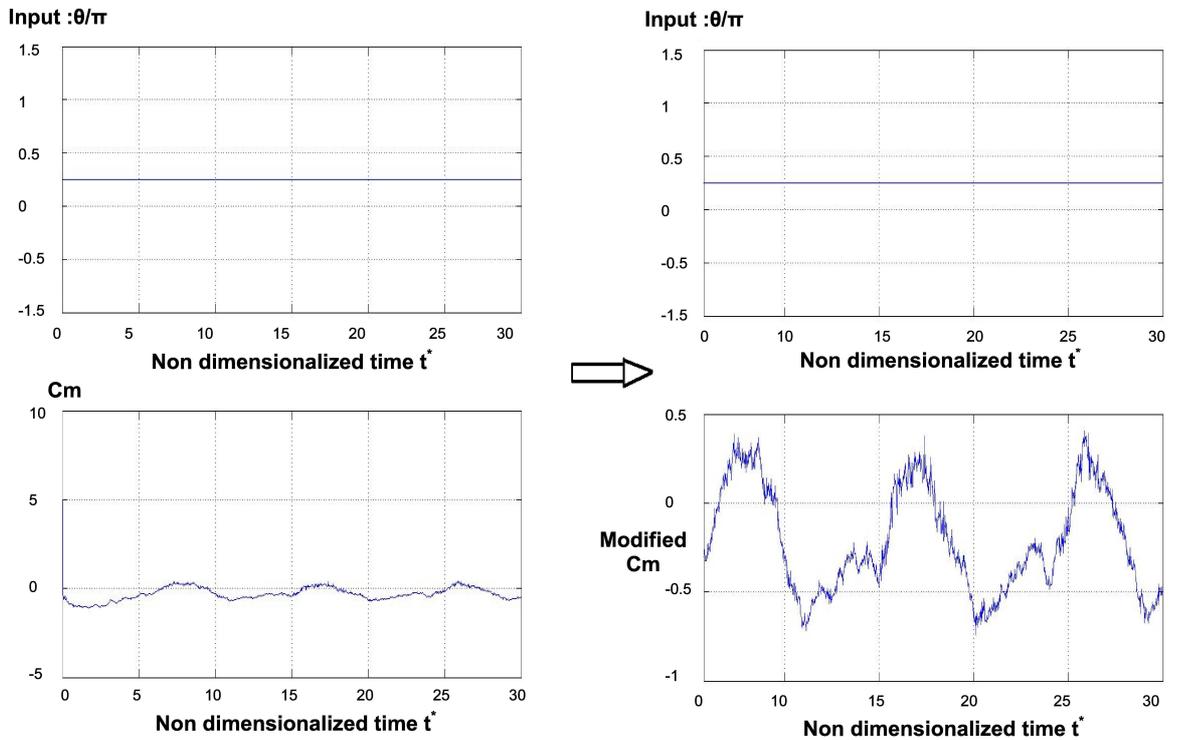


Figure 4.4: Example of simulation time range retained

Additionally, the signal has had the mean value of C_m removed in order to ease the identification of a basic transfer function.

Generally in this case (transfer function identification, deterministic signal, discrete linear identification, parameterized model), identification is done by means of the reference model method. It consists basically of the minimization of an error criterion formed from the output signals from the process and the reference model. The general scheme of the method is shown in figure 4.5.

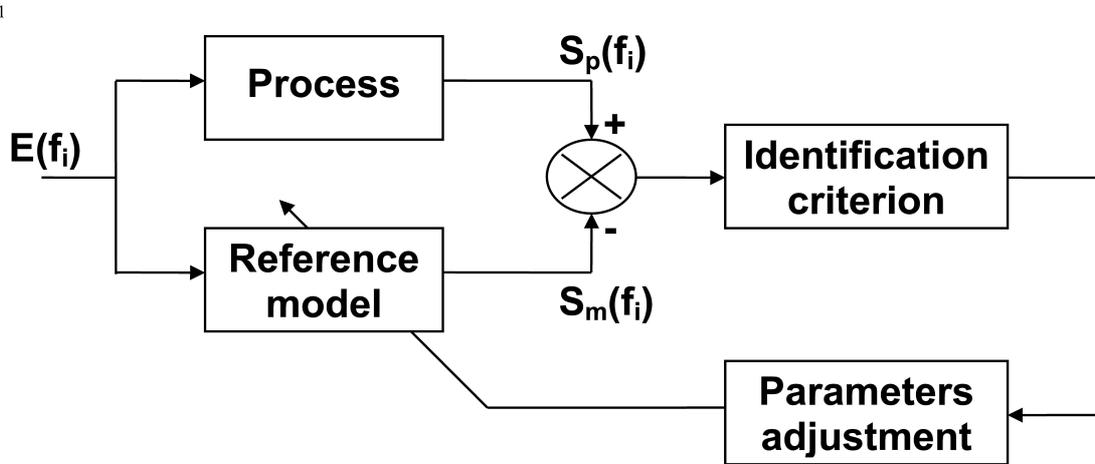


Figure 4.5: General diagram for identification method (courtesy of G.Zwengelstein)

4.3.2 Fuzzy logic theory

It is not intended here to redevelop the fuzzy logic theory but simply to offer a quick review of the method, and of its main governing principles. The main resources include reference [41], Kosko [31], and Jenkins and Pasino [29] provides a good introduction on the topic.

It is similar to an expert system, because it specifies a general behavior for the output given an input and it is not a function. It is therefore much simpler to handle than a nonlinear system, and is open to a broader variety of system characteristics. However, it is ill adapted to be used in an algebraic loop as it cannot reproduce periodic behavior from a constant input. It is thus not suited for modeling a system like the linear model implemented in sections 4.3.4 and 4.3.5 for $C_m(\alpha, t)$, but it is ideal for prediction (for example for some industrial process).

For prediction use in a control, the identification is achieved similarly to that for a linear model except that the focus is on data-set identification: which input is associated with which output through which syntactical rules, rather than model parameters.

For example, consider a dataset $A_1 = [0 \ 1]$ associated with input number 1: u_1 , a dataset $A_2 = [0 \ 1]$ associated with input number 2: u_2 , and a particular output value

y_1 of the output y . The value y_1 is then associated with A_1 and A_2 such as that $(u_1 \in A_1) \text{ AND } (u_2 \in A_2) \Rightarrow (y = y_i)$. Identification thus consists of defining for each input different datasets which cover their range of definition, the same procedure is applied for the output. Simultaneously, the rules are defined to fit the output from the input as effectively as possible. Note that there can be further complexities with different weight for the rules, but usually it is not used for identification. The degree of membership of an input (or output) to an associated dataset is defined by a membership function. Before identification (or control design), the shape of these membership function is generally fixed.

Using these notions, one can distinguish three major steps in a fuzzy system design:

- Fuzzification of the problem. This consists in the definition of a fuzzy dataset and membership functions for the inputs as well as for the outputs.
- Inference using fuzzy rules. One establish these according to the degree of membership of the input to draw conclusions about the state of the output and its degree of membership to a fuzzy dataset.
- Defuzzification. According to the degree of membership of the output to each rule, one allocate one value to the control output.

4.3.2.1 Example of fuzzy logic implementation

As an example, consider the system with two inputs u_1 , u_2 and one output y defined in figure 4.6.

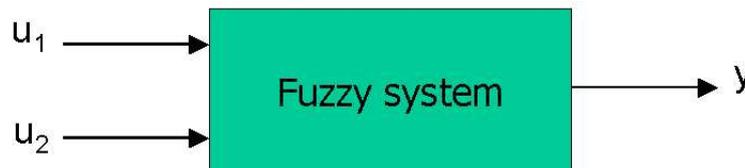


Figure 4.6: General example for fuzzy logic

Fuzzification: The first step is defining a domain for each variable. For instance, one can consider u_1 is bounded by the domain $[-10, 10]$, and u_2 by $[-10, 10]$. In the case

of an error signal this could mean that above a certain error level, the system can no longer be controlled. As for the output y , assume that the output remains within the domain $[-20, 20]$.

Then one must define membership functions, and a subset in each domain. For u_1 , one can choose as membership functions triangle-type function, and three fuzzy sets:

- N : u_1 is considered as negative, $u_1 \in [-10, 0]$
- Z : u_1 is considered as zero, $u_1 \in [-10, 10]$
- P : u_1 is considered as positive, $u_1 \in [0, 10]$

The first input u_1 fuzzy sets are illustrated in figure 4.7.

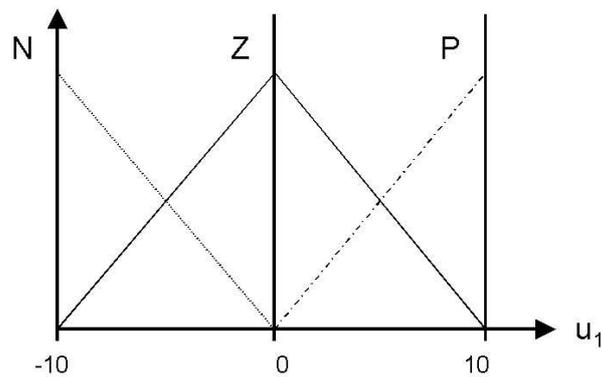
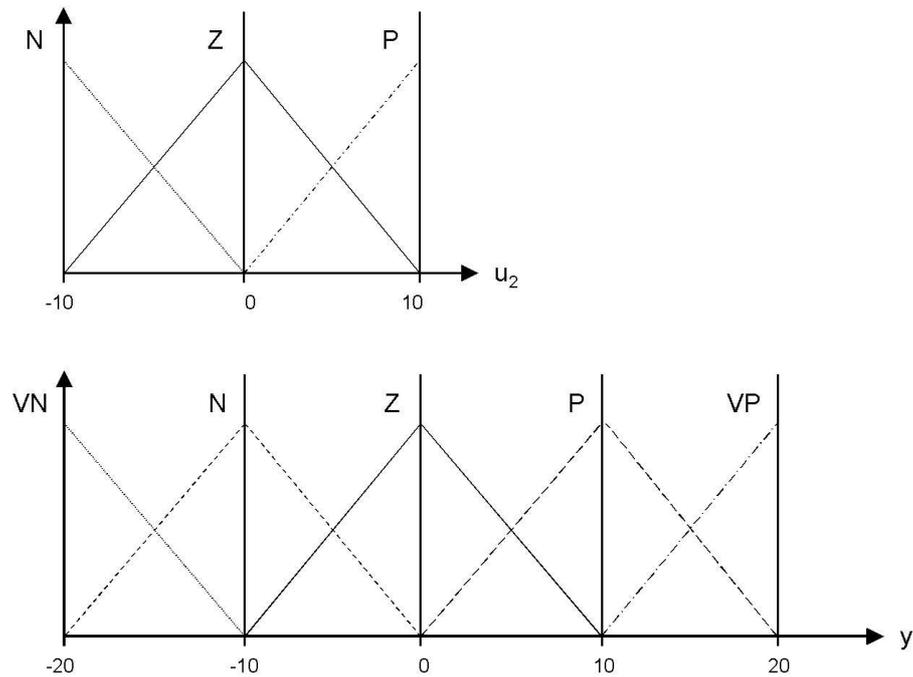


Figure 4.7: Fuzzy set for u_1

As the membership functions represent the degree of membership corresponding to a u_1 value, their value (along the ordinate axis) is always within the range $[0, 1]$. This is true for every membership function.

One can then proceed similarly for u_2 and y , except for y where further sets VN and VP are introduced for “very negative” and “very positive” cases (figure 4.8).


 Figure 4.8: Fuzzy set for u_2 and y

Inference : Now one must establish the fuzzy rules mapping the inputs to the outputs through their datasets. One can consider the signal u_1 and u_2 time derivative of u_1 , and specify that y must follow the evolution of u_1 . In this simple case, one can establish rules build on premises such as “IF (input 1 is in state A) AND (input 2 is in state B) THEN (output state is C)” using intuitive reasoning such as “if u_1 is positive and increasing (u_2 is positive) then y must be very positive”. One can then state:

1. if (u_1 is N) AND (u_2 is N) THEN (y is VN)
2. if ((u_1 is N) AND (u_2 is Z)) OR ((u_1 is Z) AND (u_2 is N)) THEN (y is N)
3. if ((u_1 is Z) AND (u_2 is Z) OR ((u_1 is P) AND (u_2 is N)) OR ((u_1 is N) AND (u_2 is P)) THEN (y is Z)
4. if ((u_1 is P) AND (u_2 is Z)) OR ((u_1 is Z) AND (u_2 is P)) THEN (y is P)
5. if (u_1 is P) AND (u_2 is P) THEN (y is VP)

Or one can draw the inference table for all these rules, which is clearer and serves as a rules summary. As there are three states for u_1 and three states for u_2 , then there are nine cases as presented in figure 4.9.

$u_2 \backslash u_1$	N	Z	P
N	VN	N	Z
Z	N	Z	P
P	Z	P	VP

Figure 4.9: Example of inference table

There is not one unique way to apply the AND or OR operator, so one can use for example the intuitive assumption that AND is the *min* operator, and OR is the *max* operator. The THEN operator applies the result of the fuzzy rule to the consequent of the rule through an *implication* operation; it can take several forms, the simplest of which is the *min* operator. The consequent is the output dataset after the THEN operation is applied. For example, for the second rule, the consequent would be the dataset N after the THEN operator as illustrated in figure 4.10. There can be several rules activated at once, to decide how to make the final “decision”. The *aggregation* of the rules is achieved through the *max* operator.

To illustrate the use of these rules and how the operators are used, the inputs are fixed at $u_1 = -6$ and $u_2 = -2$. Using the current example with $u_2 = -2$, one see from figure 4.8, that there are two states concerned : the set (u_2 is N) and the set (u_2 is Z). If one use the value -2 to determine the degree of membership to one set with the membership function defined in figure 4.8, one can see that the degree of membership is thus:

- (u_2 is N) : 0.2¹
- (u_2 is Z) : 0.8
- (u_2 is P) : 0

¹To simplify one could say that when $u_2 = -2$, u_2 belongs at 20% to the set N, or that the statement “ u_2 is N” is 20% true according to the current definition.

Similarly proceed in the same manner for $u_1 = -6$ using figure 4.7, the degree of membership to each set is then:

- $(u_1 \text{ is } N) : 0.6$
- $(u_1 \text{ is } Z) : 0.4$
- $(u_1 \text{ is } P) : 0$

Using AND as the *min* operator, and OR as the *max* operator, the result of the second rule is shown in figure 4.10. The degree of membership to the different statements are:

- $((u_1 \text{ is } N) \text{ AND } (u_2 \text{ is } Z)) : \min(0.6, 0.8) = 0.6$
- $((u_1 \text{ is } Z) \text{ AND } (u_2 \text{ is } N)) : \min(0.4, 0.2) = 0.2$
- $((u_1 \text{ is } N) \text{ AND } (u_2 \text{ is } Z)) \text{ OR } ((u_1 \text{ is } Z) \text{ AND } (u_2 \text{ is } N)) : \max(0.6, 0.2) = 0.6^2$

Thus the ordinate of the set $(y \text{ is } N)$ is leveled at a value of 0.6 as shown in figure 4.10. The same inputs are applied to the other rules, and one can proceed with the aggregation of the rules in figure 4.11.

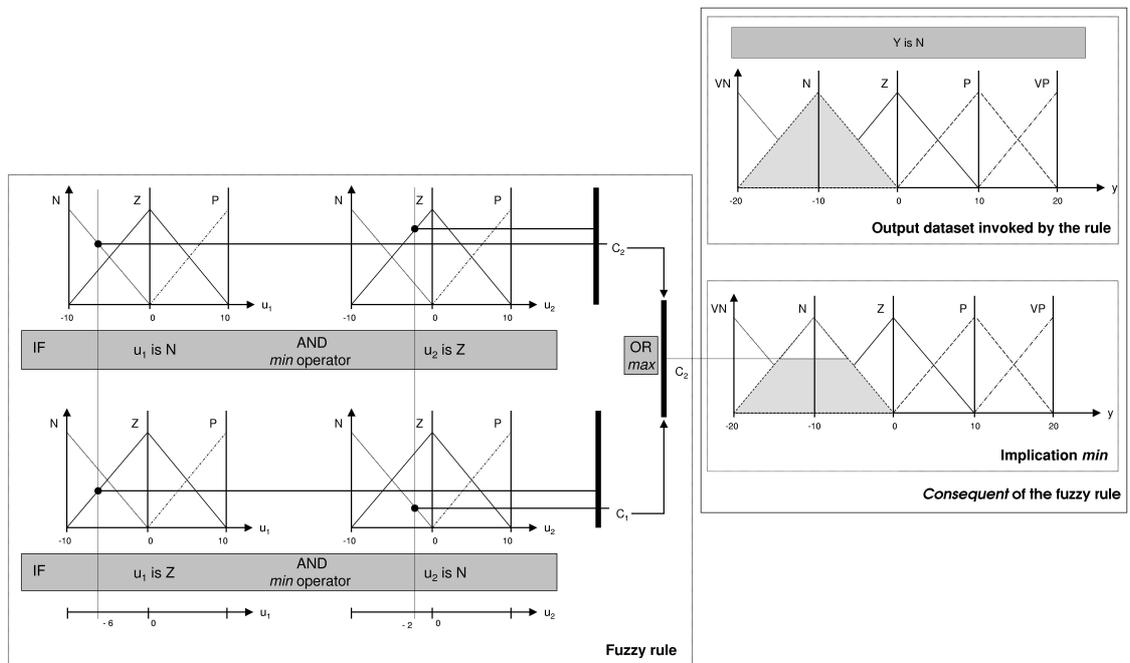


Figure 4.10: Example of inference for one fuzzy rule

²Recall that means the second rule is 60% true according to the current definition.

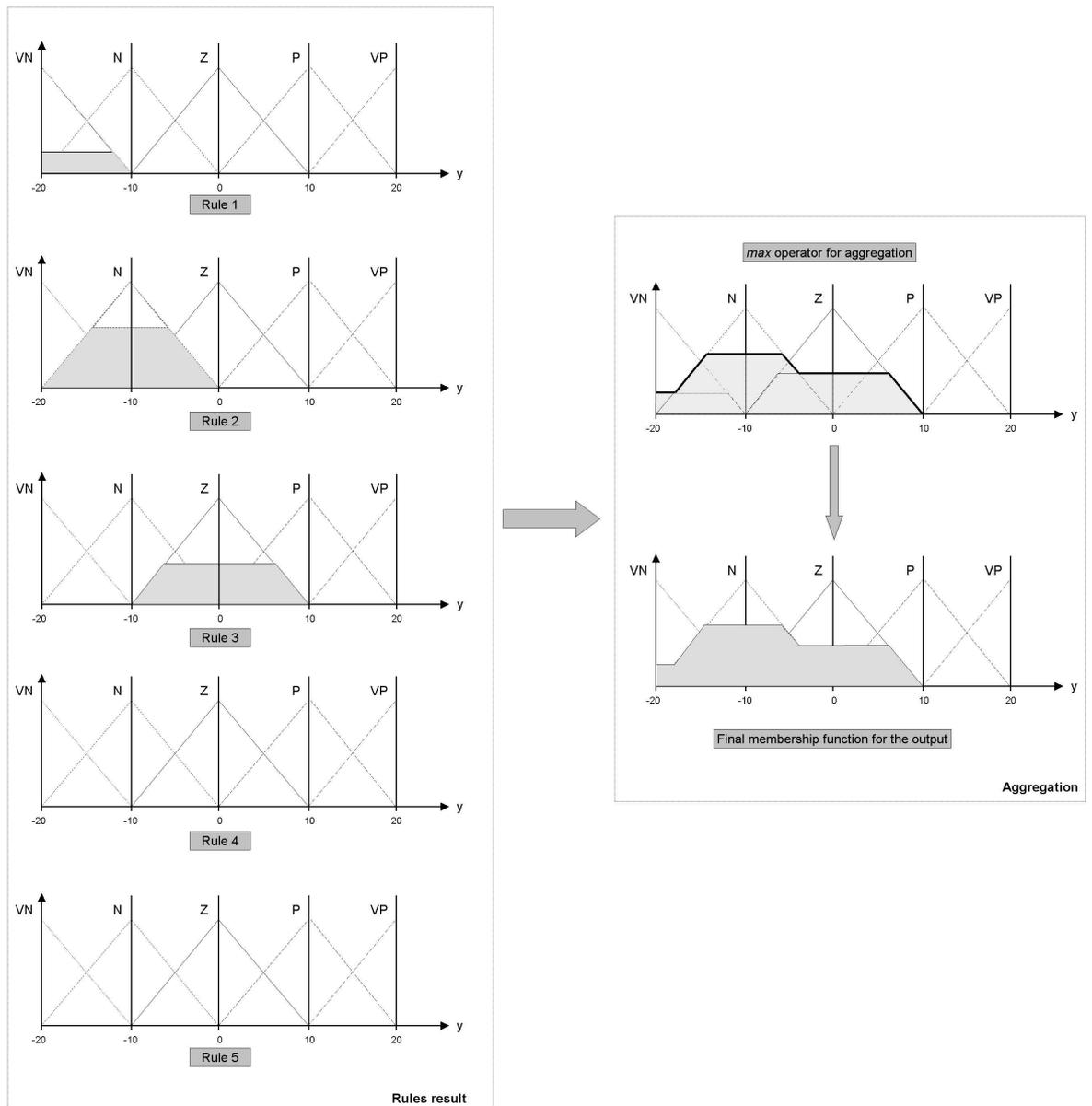


Figure 4.11: Results from all the fuzzy rules, and aggregation

Note that applying the fuzzy set notion to the output is related to the Mamdani method [36], but there are other methods like the Sugeno method [58], where the output is considered as a singleton or as a function of the input; see section 4.3.2.2 afterwards for a comparison between these two methods.

Defuzzification: Following the inference, the defuzzification consists of attributing a real value to the surface attributed to the previous output. There are several methods of defuzzification, but one of the most commonly used is that of the center of gravity. This method implies simply the calculation of the center of gravity of the surface obtained, and then taking the abscissa as the output as is illustrated in figure 4.12.

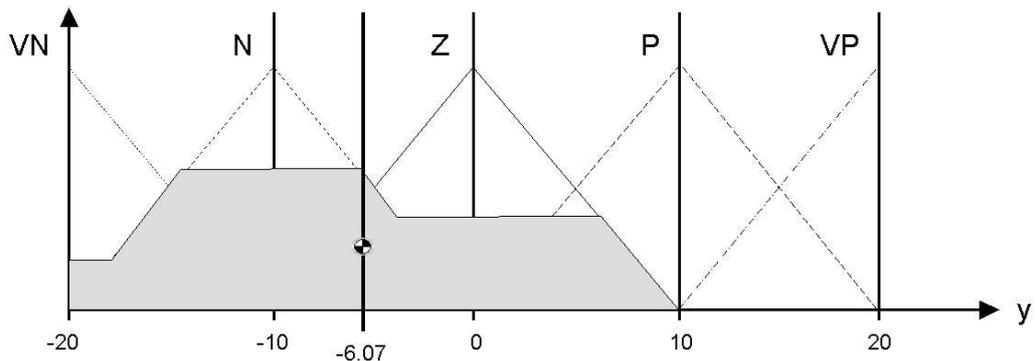


Figure 4.12: Defuzzification using the centroid method

Other methods exist, which are less computationally intensive. Among others, there are:

- Bisector of the surface
- Maxima average
- Smallest maxima in absolute value
- Largest maxima in absolute value

The latest two methods have the advantage of having the lowest computational cost.

4.3.2.2 Sugeno and Mamdani method

During the preceding sections, the Mamdani method [36] was implicitly used to illustrate the fuzzy logic method. The main differences with the Sugeno method [58] lies in the definition of the output variable and in the defuzzification method.

The fuzzification of the input variable remains the same as the Mamdani method. One proceed in the same manner for the fuzzification of the outputs, i.e. definition

of datasets for each variables. On the other hand, for the Sugeno method, instead of defining membership functions for the outputs datasets, one use either a constant value (singleton) independent of the input, or a linear combination of the input.

In the Sugeno method during the defuzzification, this output will be weight averaged by its degree of membership to the inputs, each weight being determined by the fuzzy rules. The weight determination therefore amounts to the *implication* operation. The *agregation* being done through the weight-averaged output.

The advantages of the Sugeno Method are that it is:

- computationally efficient
- works well with linear techniques (e.g., PID control)
- works well with optimization and adaptive techniques
- guaranteed continuity of the output surface
- well-suited to mathematical analysis

The advantages of the Mamdani Method include:

- intuitive
- widespread acceptance
- well-suited to human input

4.3.3 Model frequency change

As already stated in section 4.3.1, from the simulation results and from Fage and Johansen's (1927) [16] experiments for a flat plate in the range of angle of attack 45 to 90 degrees, one know that for a flat plate with a steady angle the shedding frequency is mainly dependent upon the angle of attack, more precisely on the projected area "seen" by the freeflow, which is proportional to $\sin(\alpha)$.

Now remind that the general flow model is based on several black boxes which are models of C_m taken at a given α . The shedding frequency being directly linked to the C_m main frequency, the idea here is to implement an interpolation using not only the

C_m model magnitude for a given α but also the shedding frequency in order to avoid simply having a superposition of the base functions magnitude mentioned above such as illustrated in the example shown in figure 4.3.

Through this example has been applied for sine functions, it is possible to do the same for a C_m model identification. Using such frequency interpolation, one can build a simple $C_m(\alpha, t)$ model for a plate with a constant angle of attack at a low computing cost, and using only simulation data with a limited number of angle of attack (see sections 4.3.4 and 4.3.5). That is to say one would use various $C_{m,i}$ taken at a finite number of angle of attack α_i .

In order to implement the frequency interpolation, one way would be for a given α , with $\alpha_1 < \alpha < \alpha_2$, to first compute the corresponding shedding frequency (which for a plate at steady angle is equivalent to C_m main frequency), and then implement for the $C_{m,1}$ and $C_{m,2}$ models a change of frequency such that they both have this desired $C_m(\alpha, t)$ frequency. In truth, it is not actually a model frequency change, but a change of timescale for the identification model used, which produces a model close to a gain-scheduling approach.

As the frequency has to be changed at will without impairing time continuity one has several options :

1. A time varying function to adjust the new time scale to the previous one, i.e. use $t' = f(t) t$.
2. Consider the change of timescale to be sufficiently small so as to adjust directly via a scalar k , such that $t' = k t$.

The first implementation has the advantage that the time continuity of the simulation is guaranteed; however as a side effect this implies that there are some further dynamics introduced into the model. The second implementation has the advantage of being immediate but there can be some discontinuity if one raise k too quickly. As a further remark, as discrete systems are used here (section 4.3.4), if the delay of the first implementation is smaller than the timestep, then it amounts to the same as the second implementation. Conversely, if the timestep is large enough, one can consider the second implementation as a linear adjustment of the timestep. Therefore, another problem parallel to the choice of implementation is how small the delay and timestep

should be.

For the delay, one can compare the transition duration in these models, in order to set a value which is less than the transition period. Hence, first one has to determine the minimum transition time, and from this find a reasonable delay for the timescale change.

Eventually, provided that the delay is properly adjusted, the first solution seems to be the most appropriate. Taking the Laplace transform (for causal system) of the response yields

$$H(s) = \int_0^{\infty} h(t)e^{-st} dt, \quad (4.2)$$

where $H(s)$ is the Laplace transform of $h(t)$, s a complex variable, and t the time. If one introduces now the new timescale $t' = f(t)t$ in this expression, one obtains

$$H'(p) = \int_0^{\infty} h(t')e^{-pt'} dt', \quad (4.3)$$

or in terms of t :

$$H'(p) = \int_0^{\infty} h(f(t)t)e^{-pf(t)t} \frac{dt'}{dt} dt, \quad (4.4)$$

which is difficult to develop from a single model $H(s)$, apart from the special case where $f(t) = k$, i.e. a constant.

An alternative is to use the fact that in state space form, the system can be written as

$$\dot{\underline{X}} = A.\underline{X} + B.\underline{U} \quad (4.5)$$

$$\underline{Y} = C.\underline{X}, \quad (4.6)$$

with \underline{X} the state vector, \underline{U} the input vector, \underline{Y} the output vector, A , B , C the system matrices, and $\dot{} = d/dt$. Then using the relationship $t' = f(t)t$ and the chain rules for derivatives:

$$\frac{d}{dt'} = \frac{1}{\dot{f}t + f} \frac{d}{dt}. \quad (4.7)$$

Equation (4.5) becomes when one changes the timescale from t to t' :

$$\frac{1}{\dot{f}t + f} \dot{\underline{X}} = A.\underline{X} + B.\underline{U}, \quad (4.8)$$

$$\underline{Y} = C.\underline{X}. \quad (4.9)$$

To smoothly modify the timescale while the model is running, a simple timescale change function f was first chosen such that

$$f = k_1(t - t_0) + k_0, \quad (4.10)$$

where t_0 is the time where the time change is begun, k_0 is the timescale at time t_0 and k_1 is the slope of the timescale change. The implementation is done using a Matlab function, and then inserted in Simulink.

However, this implementation poses some difficulty as one has to choose at which speed one want to modify the frequency, which essentially amounts to add another layer to the model. At this point, remind that the identification is considering the C_m variation according to α , and that this is an attempt to model the case of the continuous motion of the plate in a flow. Therefore, α is smoothly varying, and so is the desired frequency of the C_m model. Considering this remark, it was found to be sufficient to use as a function f (with $t' = f(t) t$):

$$f = k, \quad (4.11)$$

which means in fact $C_m(\alpha, t')$ is used in Simulink, after having replaced t by t' in the identified model. To compute the new value of the timescale, consider that at the end of the change of the timescale one obtains eventually $t' = k.t$. To calculate k , the frequency is taken as f_o at the previous timestep of the periodic transfer function. Now to change it to a new frequency f_1 one then has to use $k = f_o/f_1$.

This brings further complications as the form chosen for the model was the ARX model, which is a discrete state-space model. In Matlab/Simulink the ARX models are the simplest form of a numerical model. They can be expressed for a SISO (Single Input Single Output) model as:

$$y(t) + a_1 y(t - 1.t_s) + \dots + a_{na} y(t - na.t_s) = b_1 u(t - 1.t_s) + \dots + b_{nb} u(t - (nk + nb - 1).t_s), \quad (4.12)$$

with $y(t)$ the output, $u(t)$ the input, t_s the sampling time (as a numerical model is described here). This equation relates the current output $y(t)$ to a finite number of past outputs $y(t - k)$ and inputs $u(t - k)$. na then represents the number of poles, $nb - 1$ the number of zeros, and nk is a pure time delay in the system (similarly to a dead time). Note that for identification purpose, the form of the ARX model is slightly different as is shown in equation 4.14.

It is then simple to proceed by converting the ARX model into continuous state-space form, as it is easier and more intuitive to apply the timescale change in that case.

4.3.4 Basic linear model identification

As stated previously, each individual model is based on the flow simulation at a steady angle, such that the input is the fixed angle at a given time step. For this, the Spalart code was used, as it is quicker than the Sarpkaya-like method, with the following parameters:

- Number of timesteps: 600.
- Freestream velocity: $|\vec{U}_\infty| = 0.72m.s^{-1}$.
- Ellipse half width: $a = 0.1045m$.
- Ellipse half thickness: $b = a/20$ (approximating a flat plate).
- Timestep: $\Delta t^* = \Delta t |\vec{U}_\infty| / (2a) = 0.1$.
- Inviscid flow.

These conditions worked well for simulating the steady rotating plate and considering the resulting flow-induced moment. As for the choice of angle, as a first step a relatively simple set was chosen between 45 and 90 degree: [45, 50, 60, 70, 80, 90]. Other values could be chosen to improve the accuracy of the interpolation of the mean variation of C_m , which is not linear. Only six increments were chosen because of a limitation imposed by the fuzzy logic toolbox under Matlab. Note that fuzzy logic identification was tested, and in such case the absolute limitation is seven inputs. The fuzzy model would then encompass the six incidence angles plus as the seventh input the incidence angle itself. Eventually, as explained in section 4.3.2, it was not possible to use this method here.

Concerning the choice of the model structure, using the Matlab Identification toolbox, a relatively simple one was chosen: the ARX model whose determination of parameters is achieved through a least-square method. As a brief summary of the ARX model, the general equation for a discrete linear system with input \underline{U} and output \underline{Y} with an exterior disturbance \underline{E} (taken as a white noise) is

$$A(z)\underline{Y}(t) = \sum_{i=1}^{n_u} [B_i(z)/F_i(z)] u_i(t - nk_i) + [C(z)/D(z)] \underline{E}(t), \quad (4.13)$$

where n_u is the number of input, u_i denotes the i^{th} input, nk_i is the delay associated to the i^{th} input, and A, B_i, C, D and F_i are polynomial in the shift operator z . The general structure is defined by giving the time delays nk and the order of these polynomials (A, B_i, F_i), as well as those related to the disturbance model (C, D). With the ARX model, the equation is simplified as (similarly to the form of equation 4.12):

$$A(z)\underline{Y}(t) = B(z)\underline{U}(t - nk) + \underline{E}(t). \quad (4.14)$$

Another comparable alternative is the continuous-state-space model in innovations form (or single source of error (SSOE) state space model):

$$\underline{X}(t + 1) = A\underline{X}(t) + B\underline{U}(t) + K\underline{E}(t), \quad (4.15)$$

$$\underline{Y}(t) = C\underline{X}(t) + D\underline{U}(t) + \underline{E}(t). \quad (4.16)$$

After trials with the model structure at different angles the ARX model was eventually chosen over others such as ARMAX (structure 4.13 with $C \neq 1, D = 1$), as there seemed to be no need for a more sophisticated model for the disturbance. Furthermore, in this case, the disturbance is linked to the modeling of the velocity field close to the wall, so that effectively for the current simulation the external disturbance is linked to the geometry, the vortex core radius, and the timestep. The main alternative would then have been the state-space model, for it is more suitable to model multiple input, and it is structurally easy to manipulate. Eventually, the simplest model was chosen, i.e. the ARX with a least-square method option as first step, and the instrumental variables option (*iv* under Matlab) method thereafter to refine the model.

Certain modifications were applied to the data in order to make the model more efficient. To ease the ARX identification, the time-averaged value from the output signal was used (this was not required for the input as it is constant). An anti-aliasing filter was also used so as to limit spectral overlap due to the sampling. Finally, the time range retained for the simulation was selected so as to avoid the artificial start-up transition period of the simulation; this is especially important when the angle of attack is near 90 degrees. As the model is required only for a constant angle, there is less need in this case for splitting the data in two sets (one for the evaluation of the system and the other for the validation) as the input is always the same, and as the intermediary Matlab flow model is aimed at being precise for constant angle.

For the validation the emphasis was put on the output by the model, compared to

that from the simulation, regarding the frequency response and the time evolution. The aim here is to obtain as close a match as possible with the simulation output, both for the moment history and the power spectrum. This is because due to the noise, it is difficult to have reliable phase information. This is also due to the limitation in sampling time which must be equal to the simulation timestep. Nevertheless, having a meaningful power spectrum is necessary for future integration with the plate dynamics. Additionally, the ARX models identified are in general stable and have a faster decrease in amplitude than for the actual flow simulation, thus limiting their validity in time.

For a given angle, the box-diagram of a model is presented in figure 4.13.

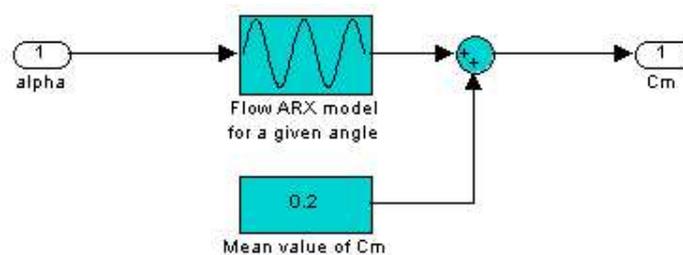


Figure 4.13: General linear model for the flow at a steady angle

For the validation, a *FIT* number was used which is provided by Matlab to see how well the model represented the simulation on the time interval. It is defined by:

$$FIT = 100 \left[1 - \sqrt{\frac{\sum_{i=1}^N (C_{m,i} - C_{ma,i})^2}{\sum_{i=1}^N (C_{m,i} - \text{mean}(C_{m,i}))^2}} \right], \quad (4.17)$$

where $C_{m,i}$ denotes the C_m value from the simulation at the i^{th} timestep, and $C_{ma,i}$ the approximated C_m at the i^{th} timestep. It is effectively the percentage of the output variations faithfully reproduced by the model.

The filter used to limit the effect of aliasing (or spectral overlap due to the sampling) is a pass-band Butterworth filter of order five provided directly in its interface. The first frequency is rejected in the low frequency range, here of order 10^{-2} Hz, in order to have the filter act like a highpass filter; the second frequency is then a cutoff frequency.

In figures 4.14, 4.15, 4.16, the results are presented for the plate at fixed stationary angles.

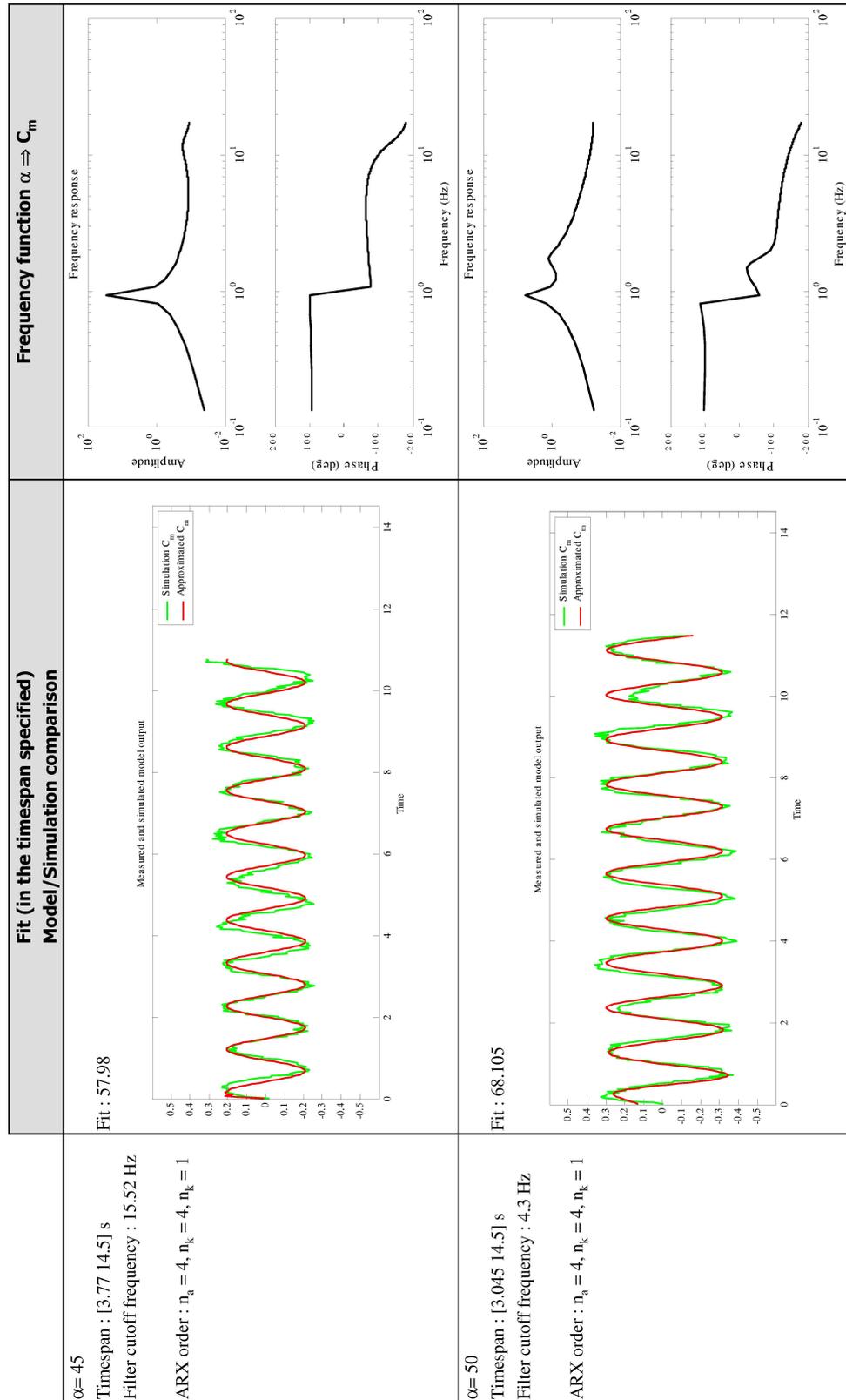
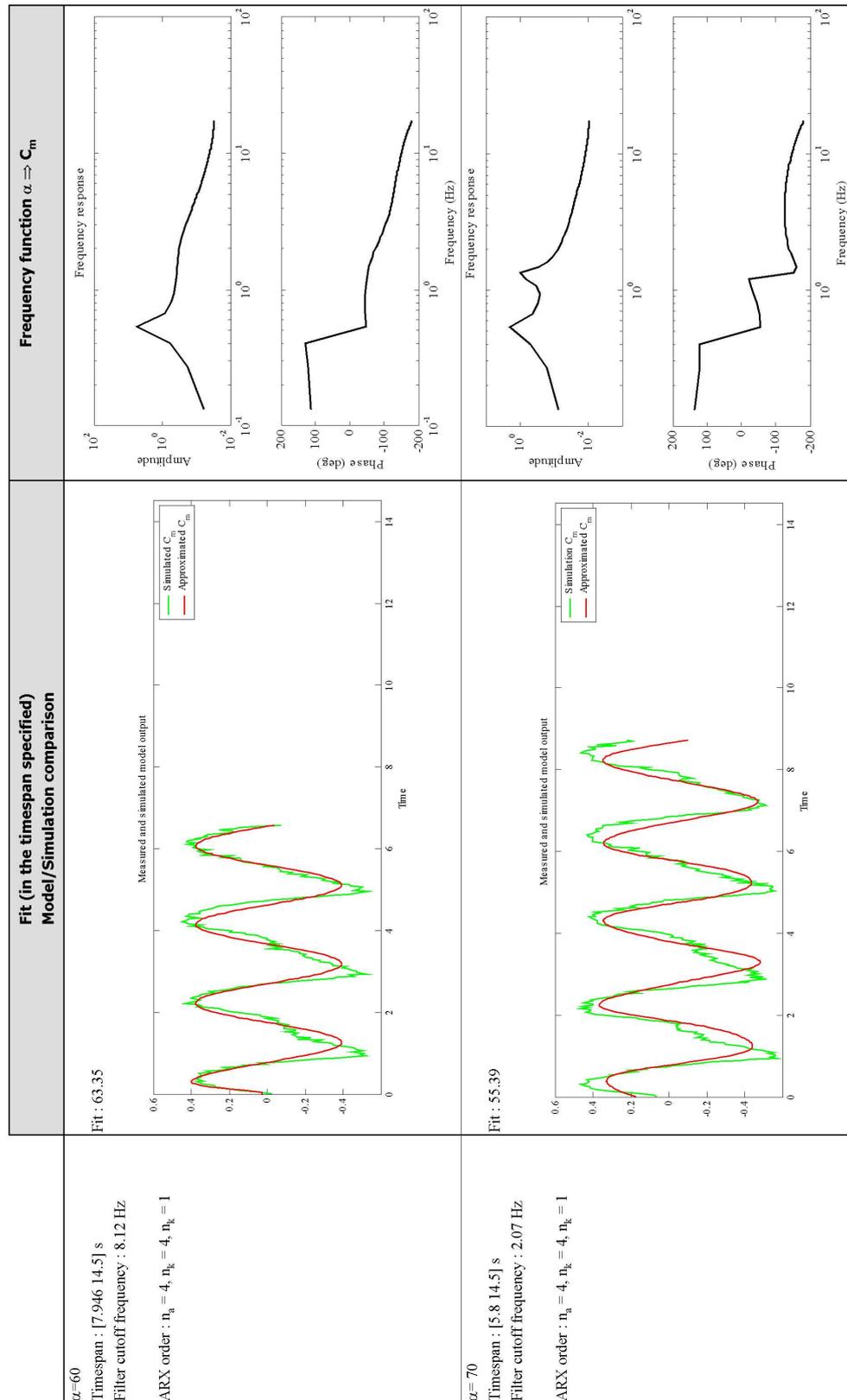
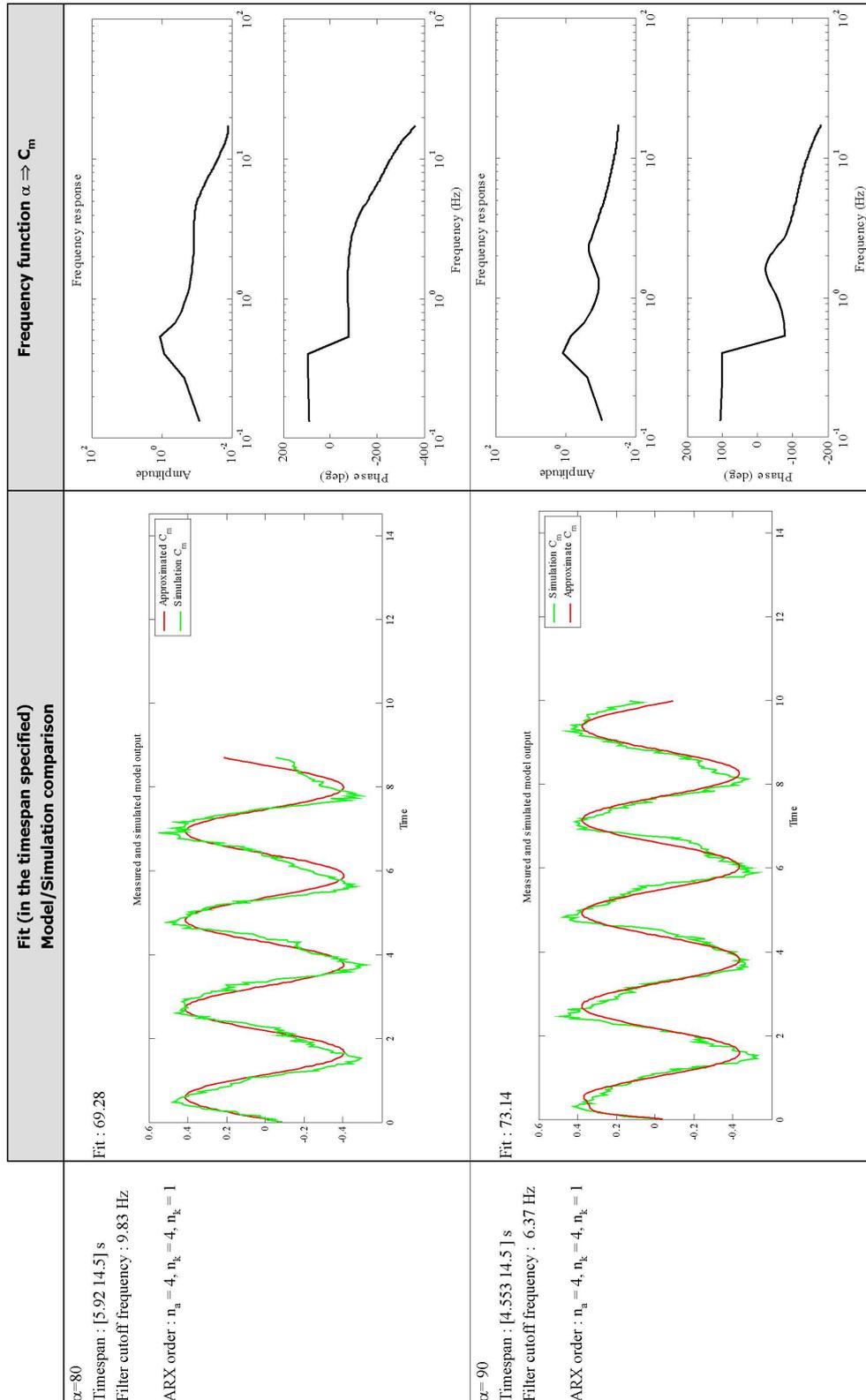


Figure 4.14: Validation results for a plate with steady incidence angles $\alpha = 45, 50$ degrees


 Figure 4.15: Validation results for a plate with steady incidence angles $\alpha = 60, 70$ degrees


 Figure 4.16: Validation results for a plate with steady incidence angles $\alpha = 80, 90$ degrees

The models can be improved by raising the order of the ARX model, however the high frequency part of the model then becomes erratic. Furthermore, this is an asymptotic improvement as it is limited in accuracy. For example, in the case of $\alpha = 50$ degrees the *FIT* (equation (4.17)) with a sixth-order model was 67, whereas with order 36 the *FIT* becomes 74. Other models with more sophisticated treatment of the noise did not fare better.

The other remark on the models is sometimes there is a “hump” on the model frequency response amplitude diagram, which is partly due to the filter if the cutoff frequency is too high and the sampling frequency too low. The sampling frequency can be expressed as $f_{samp} = 1/\Delta t$ and thus related to the timestep of the simulation, which is slightly too large. Decreasing the timestep of the simulation changes the simulation output. As for the cutoff frequency, it was kept high because at certain angles, it was impossible to obtain a stable model otherwise. The main consequence of this hump is that the transient regime has some additional perturbations at frequency of order 10 Hz, however they decrease rapidly after about 0.5 s. Furthermore, this does not have much effect on the approximated output *FIT* value if one change the filter cutoff frequency.

As for the model frequency response phase, on average the model is behaving like a classical non-minimal delay model, i.e the phase is following the slope of the amplitude in the Bode plots (negative phase with a decreasing amplitude and conversely positive phase with increasing amplitude).

To conclude, the general trend for every model is to act similarly to band-pass filter, albeit with an almost singular peak at a given frequency. This is not surprising considering the quasi-sinusoidal aspect of the simulation output under a constant input.

4.3.5 Steady-angle flow model

So far, the model considered is purely linear and based on the oscillation amplitude. For the global model, this simple interpolation considering the angle of two models is not satisfactory because in this form it is tantamount to the addition of two sinusoids of two different periods. Other nonlinearities have to be taken into account when the plate is moving.

The third implementation therefore consisted of using a global semi-linear model and adjusting the frequency for the input (as presented in section 4.3.3). Using fuzzy logic for interpolation in amplitude and to obtain base functions for the linear models, it is possible easily to develop a model for the plate at fixed stationary angles. To extract the frequency and mean function, look-up boxes were used which are basically interpolants for the given fixed data. The global model for flow at steady angles for $\alpha = [40; 90]$ degrees thus uses box model representation and is presented in figure 4.17. The functions used in the box “frequency determination corresponding to the angle” and “mean variation corresponding to the angle” are presented figure 4.18.

To avoid a phase problem when utilizing the different functions, a delay was added to the models so that the beginning of the period of each model is the same. To interpolate the model frequency f_m , data were taken from the Spalart simulation. This was done even though these data slightly differ from Fage and Johansen’s [16]; similar behaviour was expected from Spalart’s simulation and the Sarpkaya simulation, since both are vortex methods with the same boundary conditions and vortex elements (see also section 3.2.3). As for the mean value $mean(C_m)$, data were also taken from the Spalart simulation. Both functions $f_m(\alpha)$ and $mean(C_m)(\alpha)$ have been developed for α between 45 and 90 degrees. In case $\alpha > 90^\circ$, the functions $f_m(\alpha)$ (which is used as the box “frequency determination corresponding to the angle” in figure 4.17) and $mean(C_m)(\alpha)$ (which is used as the box “mean variation corresponding to the angle” in figure 4.17) then extrapolates from the functions values given for $\alpha \leq 90^\circ$. These functions are shown in figure 4.18.

The fuzzy logic box is of Sugeno type and the input membership functions are defined only for the angle of attack. An interpolation of the signals is implemented during the simulation, hence in order to determine fuzzy rules only the angle of attack is needed. Triangular functions are used to represent the membership functions of the angle, except for the $\alpha = 45^\circ$ and 90° functions for which z-shaped functions were used. This is because in this case, these functions are the end functions for the angle domain. The triangular function is a well-rounded function as it provides a smooth weighed interpolation while being simpler to use than an exponential function.

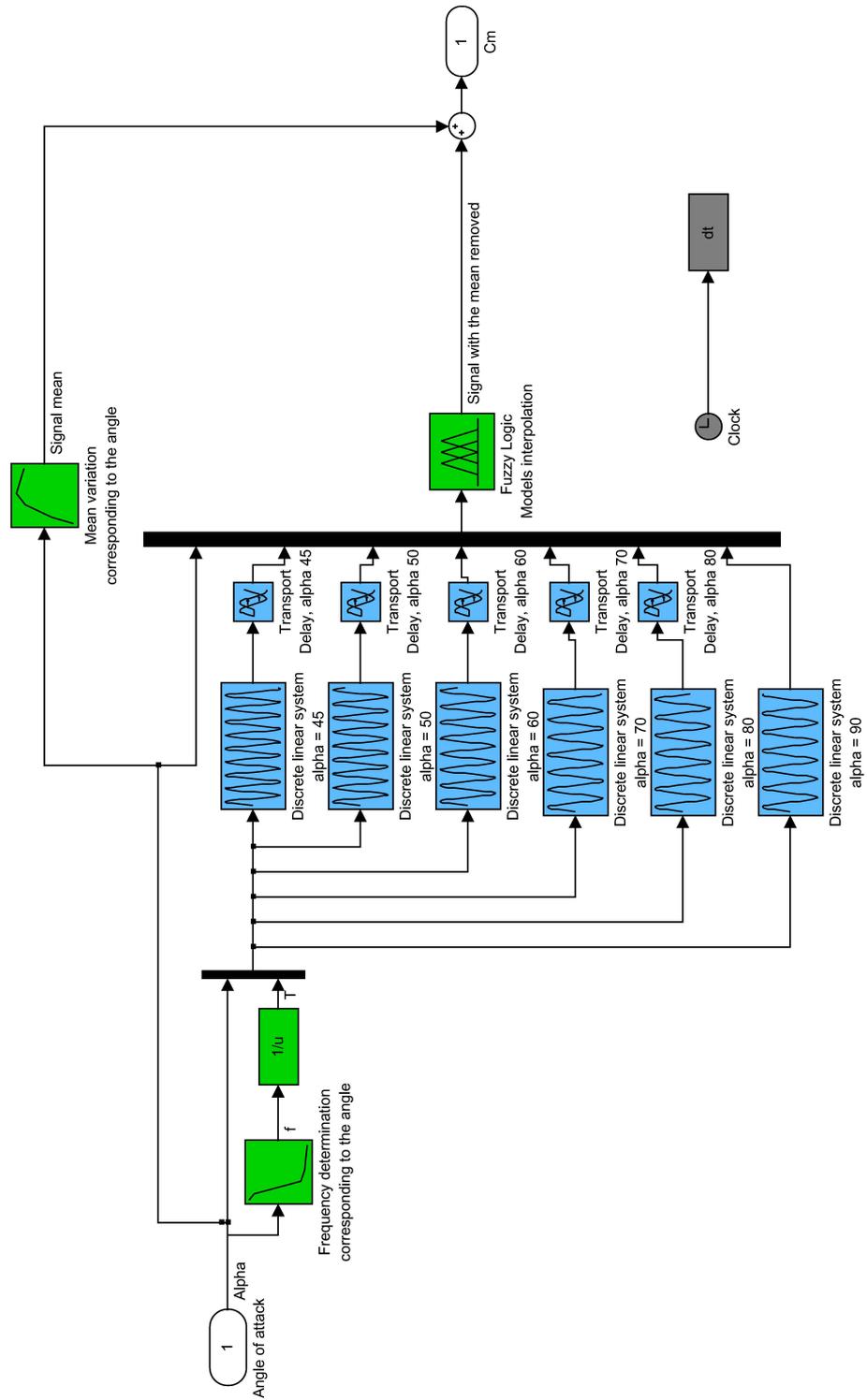
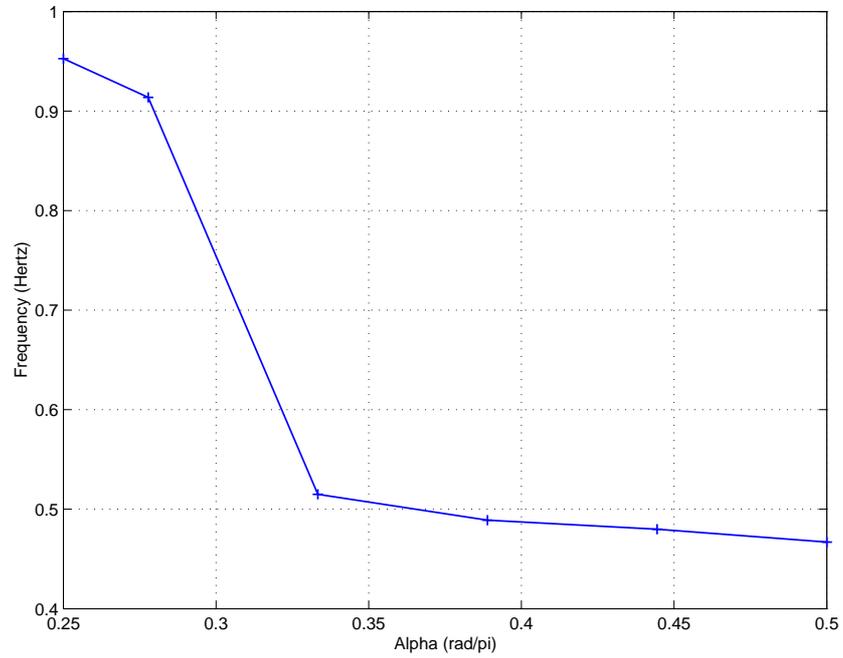
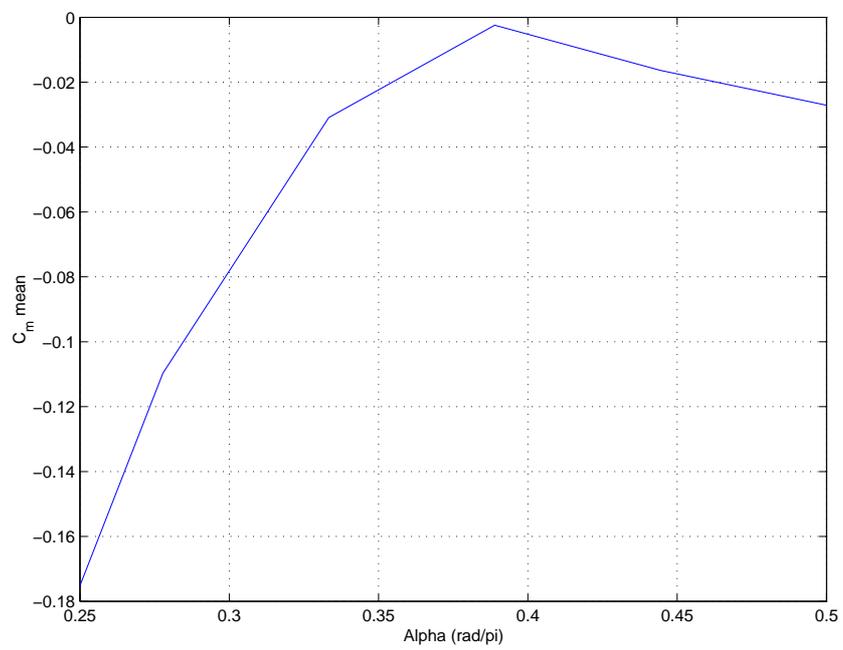


Figure 4.17: Global model for the plate at steady angles



(a) Frequency interpolation



(b) Mean interpolation

Figure 4.18: Frequency and mean interpolation function

The membership functions for the angle are defined in figure 4.19. In this figure, $mf45$ denotes the membership function of the input incidence angle α for the statement “ α is at 45 degrees”. u_i designates the i^{th} input with u_1 the angle input α , u_2 the output from the linear model based on $\alpha = 45$ degrees, u_3 the output from the linear model based on $\alpha = 50$ degrees, etc. The variable y is the output of the fuzzy model.

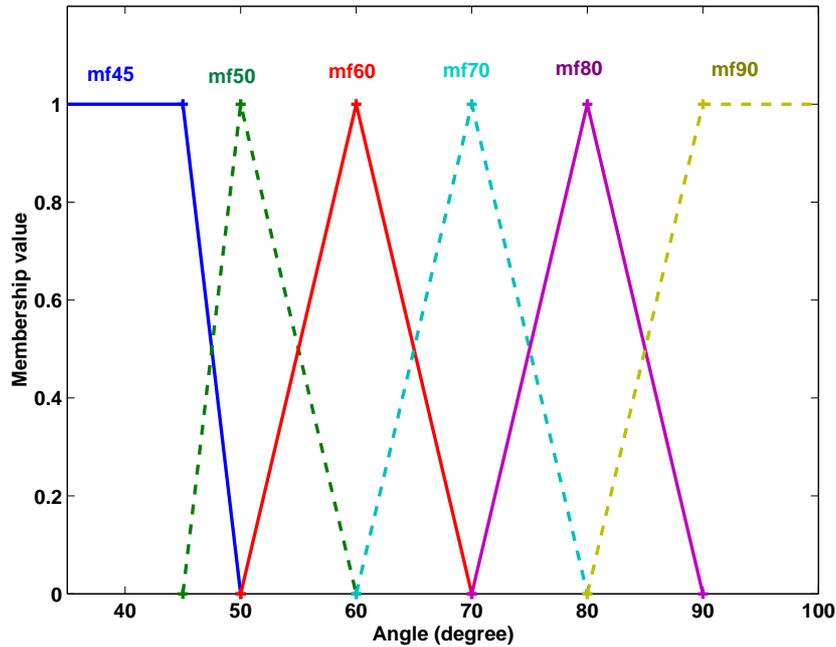
The fuzzy rules are defined as:

1. IF (u_1 is $mf45$)[i.e. the angle is 45 degrees] THEN ($y = u_2$)[the output is that of the linear model at 45 degrees].
2. IF (u_1 is $mf50$) THEN ($y = u_3$).
3. IF (u_1 is $mf60$) THEN ($y = u_4$).
4. IF (u_1 is $mf70$) THEN ($y = u_5$).
5. IF (u_1 is $mf80$) THEN ($y = u_6$).
6. IF (u_1 is $mf90$) THEN ($y = u_7$).

The result is a model similar to the one obtained for steady angle discussed in section 4.3.4. However, at angles intermediate between the fixed values, one obtains an interpolation of the different linear models.

In order to compare the model and the flow simulation, the intermediary angle of 55 degrees was chosen, with the same conditions as those used for the identification of the models (i.e. 600 timesteps, $|\vec{U}_\infty| = 0.72m.s^{-1}$, $\Delta t^* = \Delta t |\vec{U}_\infty| / (2a) = 0.1$, inviscid flow). The simulation C_m and approximation C_{ma} are compared in figure 4.20. Note that the starting period of the flow simulation is not plotted in the figure.

A delay was also added to the approximated model so that the permanent regime of both moment coefficients begins approximately at the same time. This adjustment is necessary as the model has a different transition history, and is tuned to approximate the permanent part of the simulation. As for the frequency response, the PSD was plotted using a periodogram for the simulation and the model approximation in figure 4.21.


 Figure 4.19: Membership functions of α

The *FIT* value (cf equation 4.17) is 18, and the results are shown in table 4.1.

	Simulation	Model approximation
Mean value	-0.0185	-0.0696
$\sigma = \frac{1}{N} \sum_1^N (f_i - \text{mean}(f_i))^2$	0.0757	0.0565
Maximum amplitude	0.454	0.3556
Minimum amplitude	-0.5387	-0.436

Table 4.1: Output values comparison between simulation and steady plate model for $\alpha = 65$ degrees. Here N is the number of samples used.

The discrepancies between the simulation and approximated data are due to the flow simulation taking some time to stabilize for this angle. This was not expected as the simulation was tested for other angles and did not appear to have this kind of behavior. Indeed (see section 3.3.2.1), the simulation C_m signal goes through roughly three different regimes in amplitude and frequency. It takes place around the angle of attack where the biggest change in Strouhal number occurs for the flow with the stationary plate ($\alpha = 55$ degrees corresponds to $0.31 \text{ rad}/\pi$ in figure 4.18(a)). Thus better results are attained at $\alpha = 65$ degrees for example where the *FIT* value is 67.

It was chosen to keep this model as this problem arises mainly near a given angle. A solution would be to increase the number of models, especially near this critical α ; the cost would be an increase of the complexity of the controller. Furthermore, as can be seen from figure 4.21, the periodogram³ around the peak frequency is on average very similar, which is what interests for the control design. The control has to be designed so as to be robust enough to advert this problem.

³As in section 3.3.2.2, using numerical extraction of the power spectral density in Matlab, the resulting P_{XX} is always divided by the sampling frequency, that is the inverse of the sample time

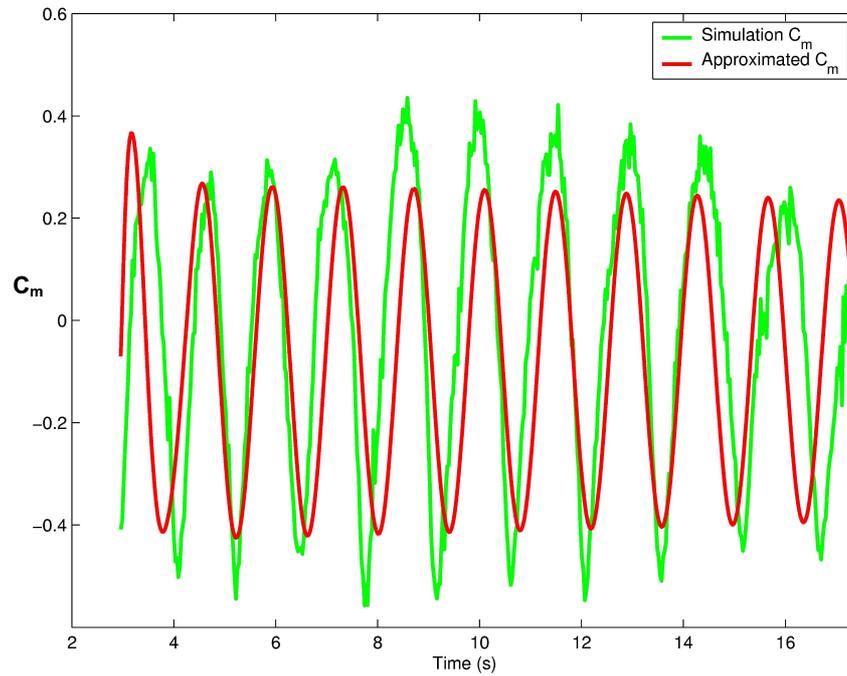


Figure 4.20: Comparison between simulation and steady plate model for $\alpha = 55$ degrees, output history plot.

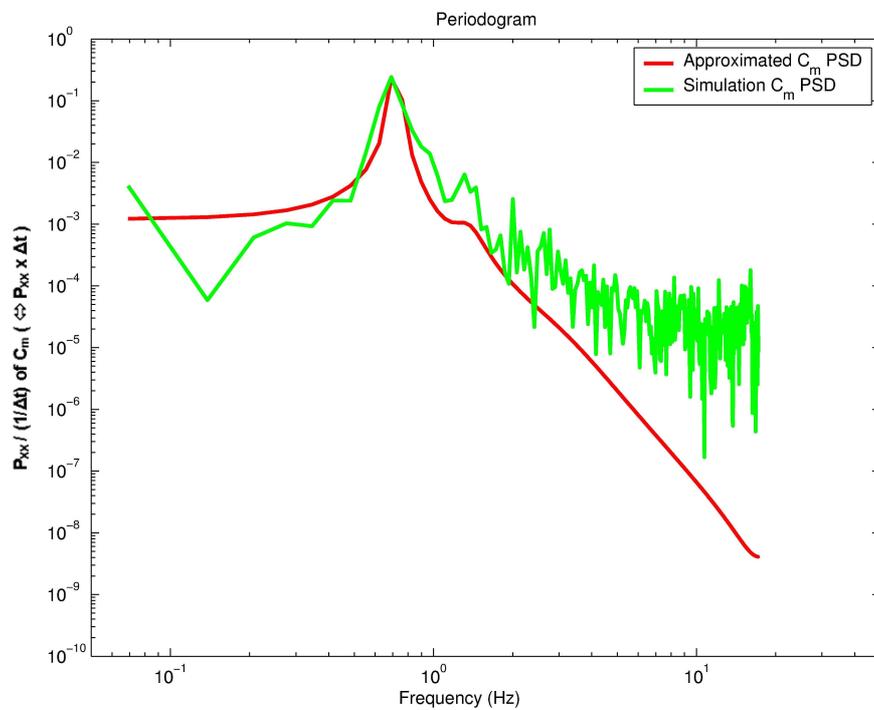


Figure 4.21: Comparison between simulation and steady plate model using a periodogram. P_{XX} is the power spectral density (PSD) of the signal $x(t)$ or –for this figure– C_m .

4.3.6 Plate model and equations of motion

The nondimensional equation of motion for a spring-damped pivoting rigid ellipse, equation 3.23, is implemented using Simulink. This involves using the non-dimensionalized timescale, $t^* = t \left| \overline{U_\infty} \right| / L$. Note that in order to specify initial conditions, Simulink uses the state-space formulation of the system. Due to the simplicity of the rotating plate system, this causes no problem.

The system description was chosen to be continuous, despite the vortex flow simulation being considered as discrete by Simulink. This allows a different timestep for the system simulation and the flow simulation; furthermore it is easier to define and describe the plate spring/damping system this way.

This yields as state space equations for the spring damped plate system :

$$\dot{\vec{X}} = A\vec{X} + BU \quad (4.18)$$

$$Y = C\vec{X}, \quad (4.19)$$

$$\text{with } \vec{X} = \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix}, \quad \vec{Y} = \theta,$$

$$\text{and } A = \begin{pmatrix} 0 & 1 \\ -\frac{k^*}{J^*} & -\frac{\mu^*}{J^*} \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ \frac{1}{J^*} \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 0 \end{pmatrix}.$$

The block-diagram Simulink representation of the system dynamics is presented in figure 4.22.

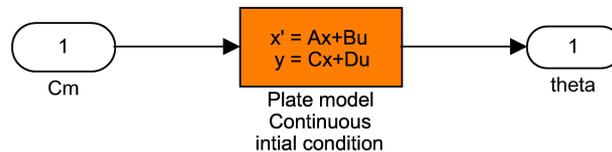


Figure 4.22: Plate dynamics model

For the plate density, the assumption is that the plate is either made of aluminium or polystyrene in order to consider two extremes for the angular inertia (see section 4.4 for the density values). Note that (as stated with equation 3.26) the ratio “plate to fluid

density” per se is not the main parameters for the oscillations, but rather J^* to assess the influence of the fluid added. The emphasis is put here on ρ_b/ρ because the same kind of ellipse is kept throughout the study (except for section 3.2.4.1), however its use would be abusive for a broader study. The parameters μ and k will be set to very low values, so that one always has $0 \leq \xi \ll \sqrt{2}/2$ but keep a wide range for ϖ_n .

4.4 Coupled plate and flow characteristics

In order to provide an overview of the plate model coupled to the stationary plate flow model, six cases are considered. As a basis, the natural frequency of the spring damped plate ϖ_n is taken as equal to the predominant frequency in the C_m spectrum for the flow simulation for a stationary plate at $\alpha = 60$, so that there is a resonance when $\alpha = 60$ in the model; ξ is also fixed as equal to $\xi = 0.01\sqrt{2}/2$ to have very low damping. The rest position of the spring ($\theta = 0$) is then fixed at $\alpha = 50$, $\alpha = 60$ and $\alpha = 70$, i.e. the model considered is the plate placed at angles of attack of respectively 50, 60 and 70 degrees. To set the angular inertia of the plate, the density used are those of aluminium and extruded polystyrene, that are respectively of $\rho_b = 2700 \text{ kg/m}^3$ and $\rho_b = 35 \text{ kg/m}^3$. The ellipse is assumed to be 20 to 1 (major axis twenty times the length of the minor axis). This covers most of the situations to be coped with by the controller, and enables to assess not only the influence of the inertia of the plate, but also the behavior of the plate on the coupled system. Implicitly, the flow characteristics are inherited of the flow simulation in chapter 3.

Note that the resonance obtained is artificial and results from the linear model of the flow at steady angle. Therefore, it does not necessarily correspond to the behavior of the true system, nor guarantee the control efficiency. Other nonlinearities are not taken into account like possible variations in flow characteristics with high amplitude θ , or with varying ϖ_n (see section 3.3.2.2); furthermore the flow damping is not represented here. Again, the controller is to stabilize the plate so that the flow coupled with the spring-damped plate behaves like a flow with a stationary plate.

$P(s)$ is the Laplace transfer function of the plate with spring and damper, and $F(s)$ the transfer function of the flow system, and α is assumed to be constant, then the coupled system transfer function $G(s)$, which can be extracted from figure 5.1 will be:

$$G(s) = \frac{P}{1 - FP}. \quad (4.20)$$

P_1 and P_2 denote the plate transfer function corresponding to respectively polystyrene and aluminium. Concerning the flow model, F_i denote the flow transfer function from the linear model at $\alpha = i$. Finally $G_{i,j}$ designates the coupled model of the P_i plate transfer function and the F_j flow transfer function.

The different transfer function are shown in figure 4.23, 4.24, 4.25, 4.26, 4.27 and 4.28 using Bode diagrams.

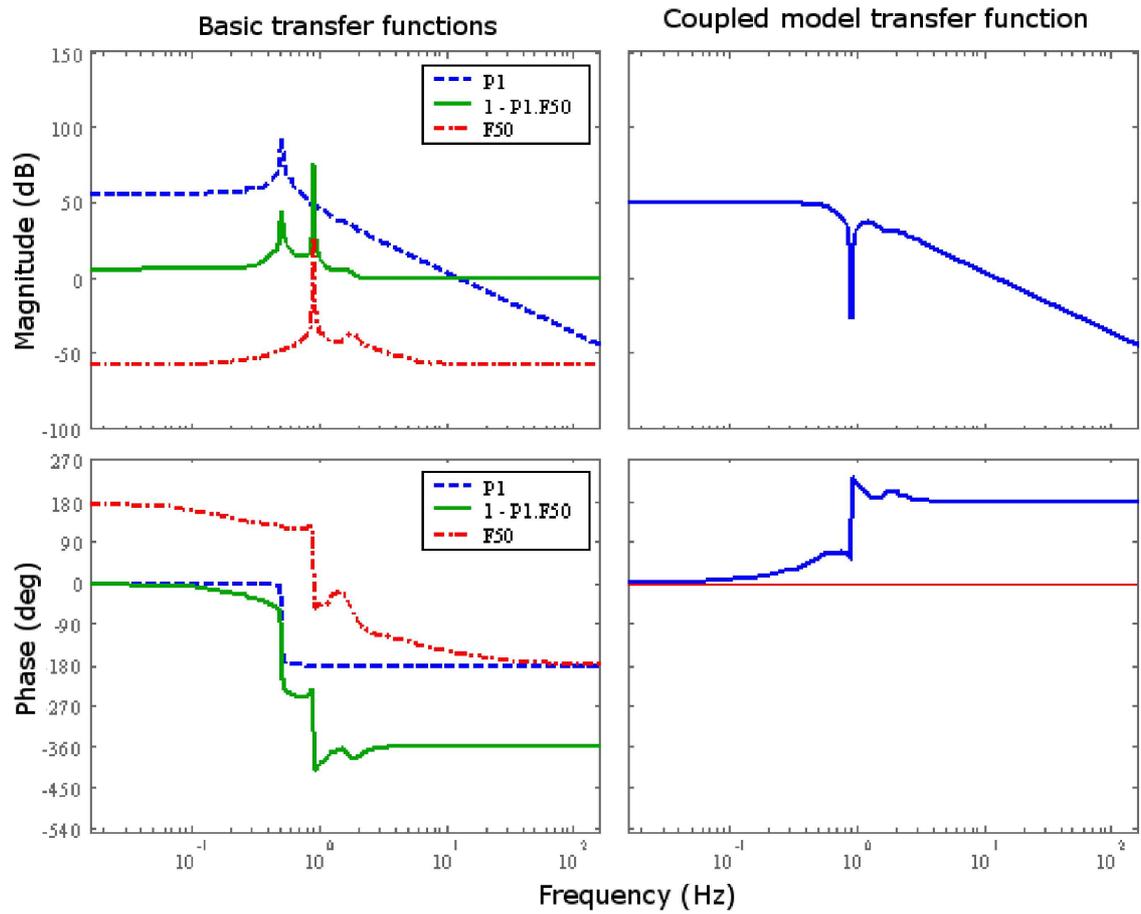


Figure 4.23: Coupled plate and flow model Bode diagram at $\alpha = 50^\circ$ using polystyrene

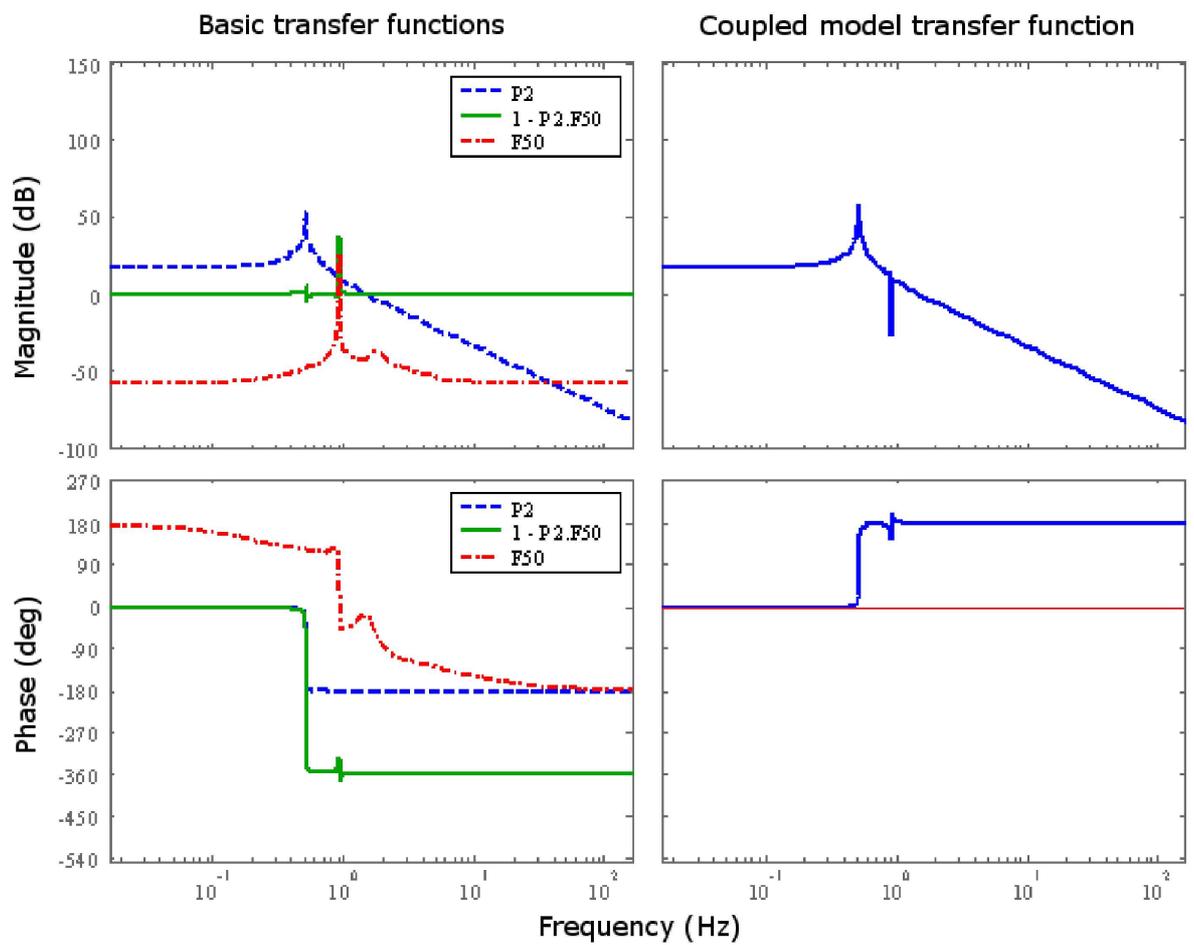


Figure 4.24: Coupled plate and flow model Bode diagram at $\alpha = 50^\circ$ using aluminium

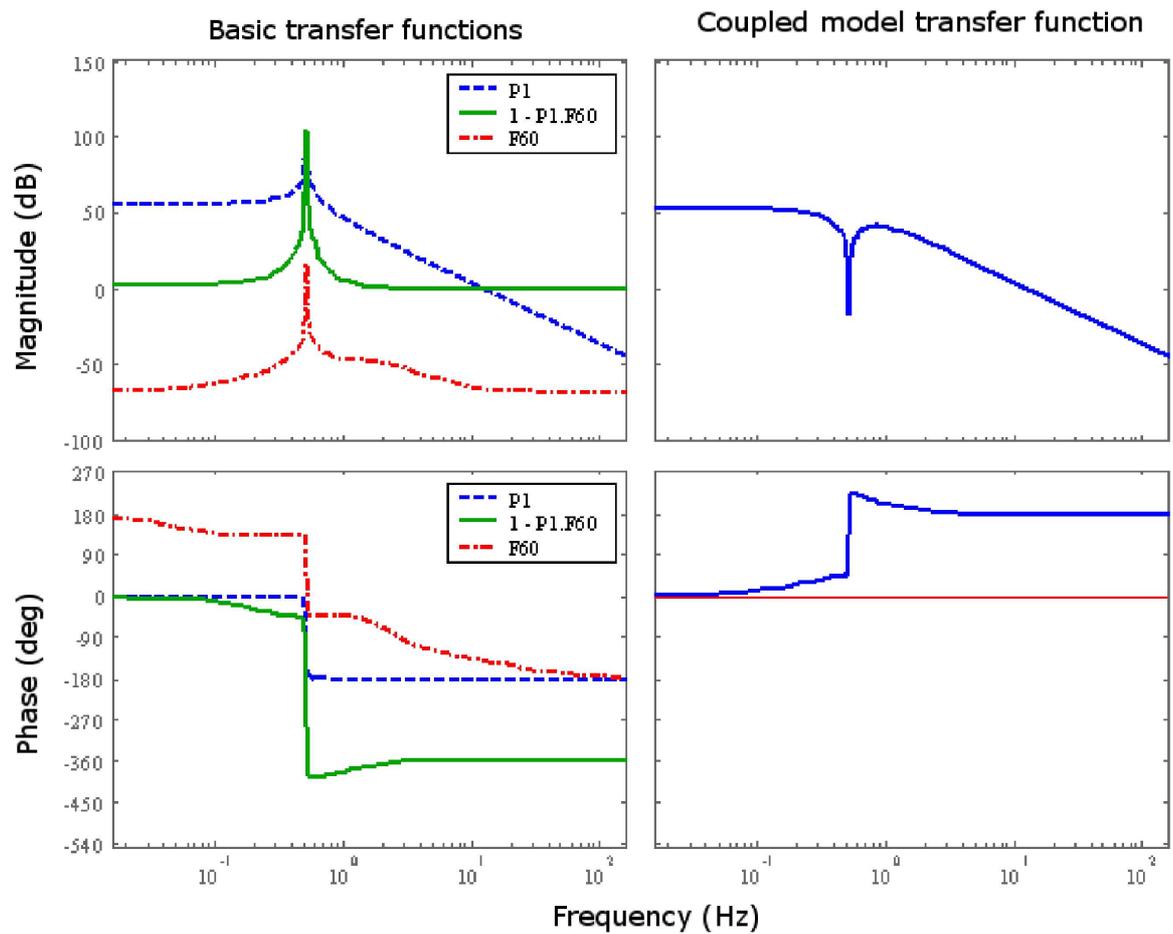


Figure 4.25: Coupled plate and flow model Bode diagram at $\alpha = 60^\circ$ using polystyrene

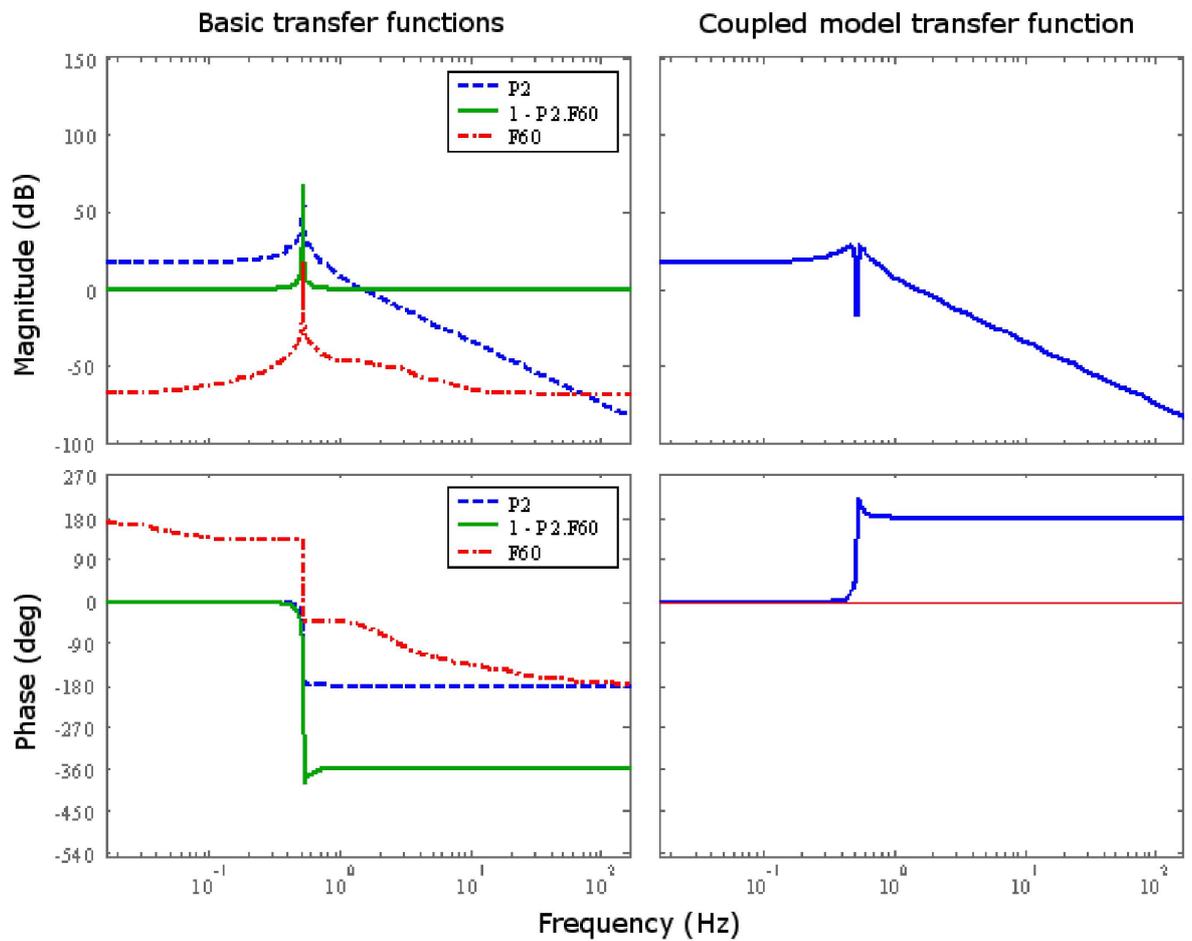


Figure 4.26: Coupled plate and flow model Bode diagram at $\alpha = 60^\circ$ using aluminium

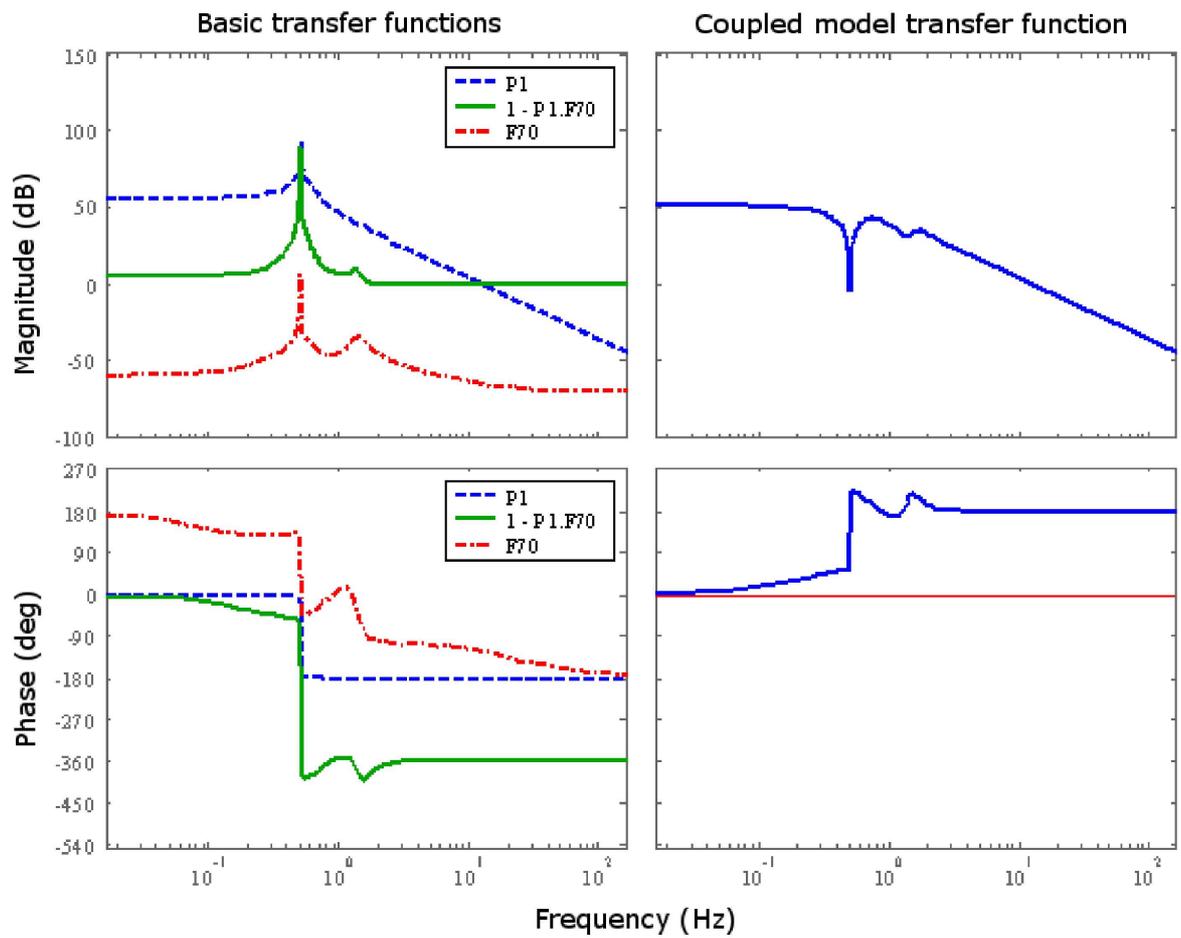
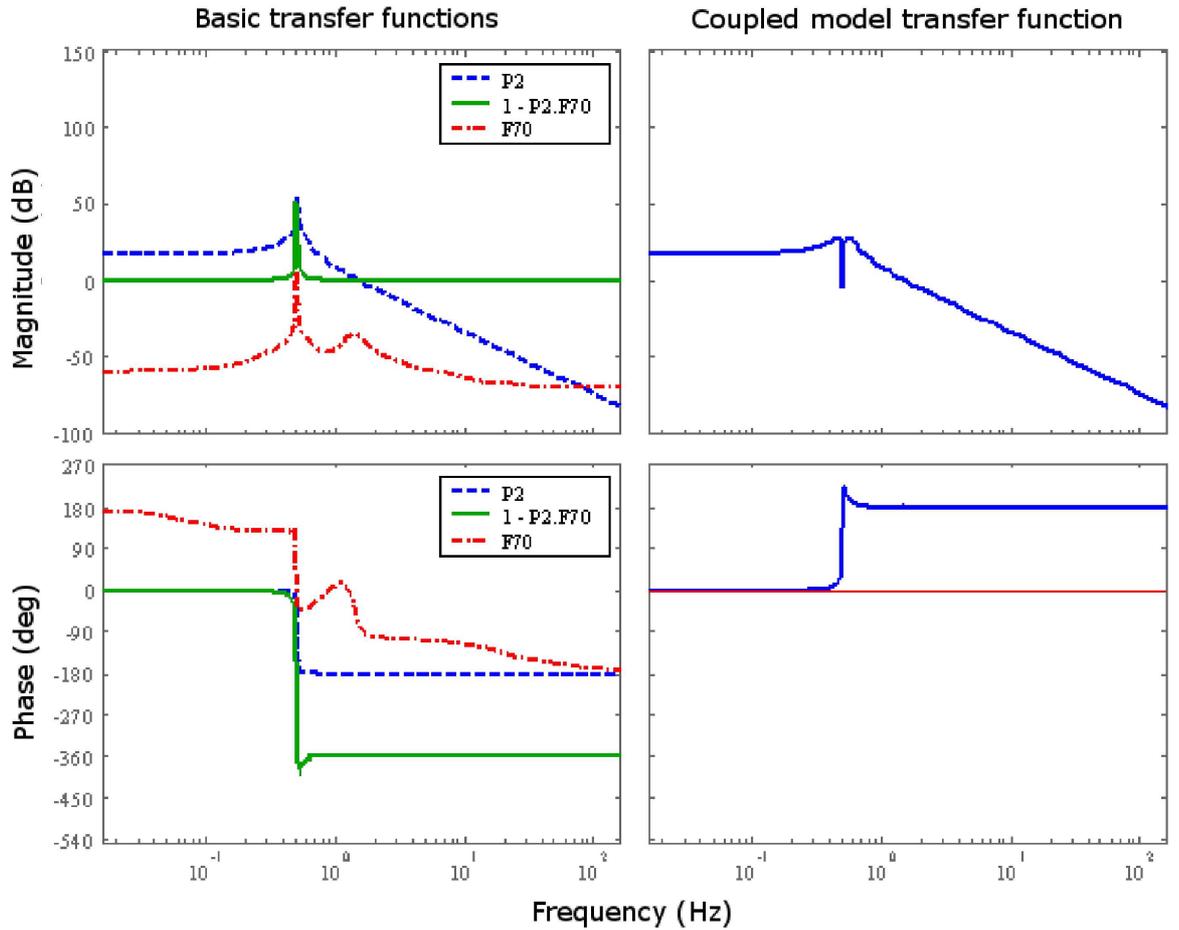


Figure 4.27: Coupled plate and flow model Bode diagram at $\alpha = 70^\circ$ using polystyrene


 Figure 4.28: Coupled plate and flow model Bode diagram at $\alpha = 70^\circ$ using aluminium

At infinite frequency the resulting coupled system is similar to that of the conjugate of the plate frequency characteristics. This means that now the system has non minimal delay, and also that the frequency at infinity is well attenuated. However, one can see that the transformation from discrete – the linear flow model was identified assuming discrete systems – to continuous has not had an adverse effect on the model.

Although system stability is difficult to deduce using for example the transfer functions in figures 4.27 and 4.28. Nevertheless, after computing the gain and phase margin and plotting the Nyquist plot of FP in the conditions used in figures 4.27 and 4.28, the system appears to be stable albeit with a very low phase margin of $O(10^{-1})$ degrees. This means that the system reaches a steady state after about two minutes. In every other case the system is unstable. It appears that if the flow frequency is lower than the plate frequency, the system is stable. This is mainly because the plate transfer

function is of second order with a very low damping, and the flow frequency is above ϖ_n . Thus the effect of coupling both systems is that there is a low value peak on the global transfer function after ϖ_n , implying that the gain margin is positive, with a positive phase margin. It is then obvious that a simple proportional controller is sufficient to have a stable closed loop system, even though the phase margin is still very low.

There is also a predictable system tendency of being more influenced by the flow as the inertia of the plate diminishes. But this effect is less important than that concerning the angle. It is also clear from the figures that ξ has minimal influence on the final system from a control point of view. Indeed, it would not change a system from stable to unstable like a change of ϖ_n , and the problem remains within the same class. Therefore it can be deduced that the influence of ϖ_n is dominant over that of the angular inertia.

Because the linear flow model at a given angle is close to a pure oscillator, once this model is inserted into the loop, it renders the system more difficult to control – although some results can be obtained using root locus techniques.

One must also remember that using this system for analysis is only partially correct as it evolves with changing θ , and thus is only true as long as the control sufficiently stabilizes the system. Without this assumption, the system would be much more difficult to characterize, making it impossible to use classical techniques. Finally, the system is of rank zero, that is to say there is always a static error.

4.5 Summary

With this chapter, the difficulties in flow/structure modeling have first been outlined in section 4.2. Then, section 4.3.1 has outlined the strategy for the design of the identified model namely a model based on the fixed plate in a crossflow at different angle of attack. One linear model (whose identification is described in section 4.3.4) is used per angle of attack, and the transition between the model is ensured through a fuzzy logic controller (section 4.3.2) enabling magnitude and frequency interpolation (section 4.3.3) with an approach similar to gain-scheduling method.

The final model is presented in section 4.3.5 and shows that for a given angle of attack

it manages to broadly reproduce the spectrum characteristics of the numerical simulation which is essential for the controller design. Part of the problem could be overcome by a better angle of attack distribution for the final model. This problem could also be attenuated by verifying the controller robustness after its design.

Finally, section 4.4 has shown the properties of such plate coupled with the simplified flow model. With such flow model, the reduced damping for the spring/damped plate is of less importance, especially compared to the influence of the ϖ_n which is a property similar to that observed in section 3.3. Otherwise, with a plate density of $\rho_b = 1200 \text{ kg/m}^3$ the system would be stable on the long term, although the flow linear model being close to that of a pure oscillator means that the coupled flow/plate system would be difficult to control regarding the system dynamic. Nevertheless, one should keep in mind the limit of this exercise as the flow model here is linear as opposed to the physical system which evolves with changing θ , and thus is only broadly true as long as the control sufficiently stabilizes the system.

Chapter 5

Coupled plate/flow system control

5.1 Outline

In chapter 4, the development of a flow model (section 4.3.5) based on the magnitude and frequency interpolation (section 4.3.3) of several linear model. Each of this linear model is based on the identification on the flow at a given angle of attack (section 4.3.4).

In section 4.4, a short study of this flow model has shown the properties of this flow model coupled with the plate, the most important being the importance of ϖ_n , and the fact that for $\rho_b = 1200 \text{ kg/m}^3$ the system is stable. The dynamic of the system is then the most difficult part to control.

Analysis from the Spalart flow simulation for the oscillating plate in section 3.3 also confirm this analysis. It has also shown that despite the inherent nonlinearity, the parameters did not affect wholly the forces coefficient as they retain a similar spectrum within the parameter space considered.

In this chapter, first the control approach will be described in section 5.2.1 and then the controller design objectives in section 5.2.2. Afterwards, after having proceeded to the design of the controller, a comparison is given between a classical controller 5.2.3, an optimal controller 5.2.4 –both similar to a Takagi-Sugeno controller– and a fuzzy logic controller in section 5.2.5. Every controller were compared using a step input as the test case and the steady angle flow model from section 4.3.5.

Finally, in the second par of the chapter (section 5.3), results are presented for the different controller coupled with the Spalart flow simulation for three test cases. The

first test case in section 5.4.1 aims at assessing the plate stabilizing for a given angle of attack. The second in section 5.4.2.1 shows the response for a step-input type command. The last test case in section 5.4.2.2 submit the flow and controller to a sine input with a sine frequency corresponding to the f_n^* of the spring/damped plate.

5.2 Control Development

The controller here is developed with a similar approach to the “model based design” used in the industry. That is to say, here the controller design is based on a linear model of the physical process. The identification for the linear flow model has been realized in section 4.3.5, and the model for the plate rotational motion has been presented in section 4.3.6. In this section is thus presented the approach for the controller design based on this flow and plate dynamic model. Section 5.2.1 presents the general approach for the controller design, section 4.4 presents a small analysis of the coupled flow and plate system based on the linear models. Sections 5.2.3, 5.2.4 and 5.2.5 present the results obtained using respectively classical design, an optimal controller, and a fuzzy controller using these linear models for the plate and flow dynamics.

5.2.1 General approach for the control design

The aim of the controller is to control the angle of the system, for example in a typical sequence: bring the plate into the desired position and keep it there. The system is the plate coupled with the aerodynamic forces, and is thus based on equation (3.14) with modification to account for the input from the controller:

$$J\ddot{\theta} + \mu\dot{\theta} + k\theta = M(t) + C_a(t), \quad (5.1)$$

where C_a is the torque coming from the actuator. This is a SISO system. The block diagram of the plate/flow system is shown in figure 5.1.

Here θ is again the angular position of the plate from the rest position of the torsion spring. This is because it is more sensible to use the plate position compared to the spring rest position as opposed to the plate angle of attack α . Nonlinearities due to high spring deflection value are not taken into account, and the plate is assumed to be perfectly rigid with infinite stiffness. In figure 5.2, the full system is represented under these conditions.

The first approach was to implement a strategy similar to that used in aircraft control, that is to stabilize the plate such that it would be insensitive to perturbations. However, the external flow perturbations considered in flight conditions are of the white-noise type, or a delta function (for a gust). Furthermore, the aircraft is assumed to be in steady flight condition whereas the plate is in stall. Therefore, the aircraft-control analogy is ill adapted to this case, which involves periodic excitation bringing constantly energy from outside the structure.

The closest analogy in aircraft control would be a flutter controller. But there, the vibrations involved originate from structural flexibility as well as aerodynamic forces, in contrast to the rigid plate assumed here. The same applies for flexural vibration suppression control, some of which involves modal control (Ballas 1978 [4]).

Robust control applied to a nonlinear system can also be found, but Theolis 1994 [60] points out its computational complexity. Aside from the complexity, a control where the robustness is explicitly taken into account does not seem necessary as the measures taken either directly from the simulation, or from the Simulink model from section 4.3.5, are not very noisy.

As noted in the previous section, the resulting aerodynamic moment itself is nonlinear, so that the plate coupled with the flow model is inherently nonlinear. Thus, comparison is made here between three implementations. Firstly, because they sometimes provide good first insight into the nature of the system, classical control techniques are considered.

The second is an optimal based controller with α (or θ) as the control parameter. Similar to the strategy outlined in section 4.3.5, a given set of fixed angles and design is employed for each angle an optimal controller that is linearly interpolated for intermediary angles using a fuzzy box; this is an approach similar to the Takagi-Sugeno models. Indeed, one can assume that if the plate is able to stay stable around a given angle – with the controller maintaining the position – the system is close to a linear one at that angle. Athans [3] emphasized that the Takagi-Sugeno model produces results that were less satisfying than a gain-scheduling control for MIMO design, the main weakness being that all mismatched control/systems (control specific to an angle applied to a model based on another angle) are not guaranteed to be stable. However, he did not

precise the exact kind of gain-scheduling design he used, nor does it appear clearly in the proposed literature review in its debate [3].

The third implementation was a fuzzy-logic controller, which if prescribed for stabilization and command, is much more appropriate for a nonlinear system. Its advantage over adaptive or neural control lies in its implementation simplicity, as it does not require an extensive study of the system. However, it does not explicitly take into account stability and robustness, and thus is mainly heuristic in nature. Remark that certain method are using robust control parameters and methodology (the circle criterion for example) in order to find optimized set for the membership functions (see Jenkins and Pasino [29] for example).

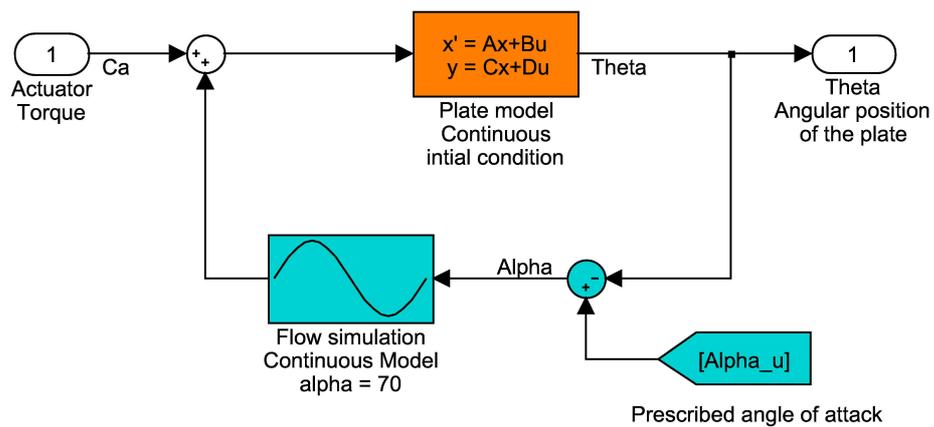


Figure 5.1: Coupled plate and flow model

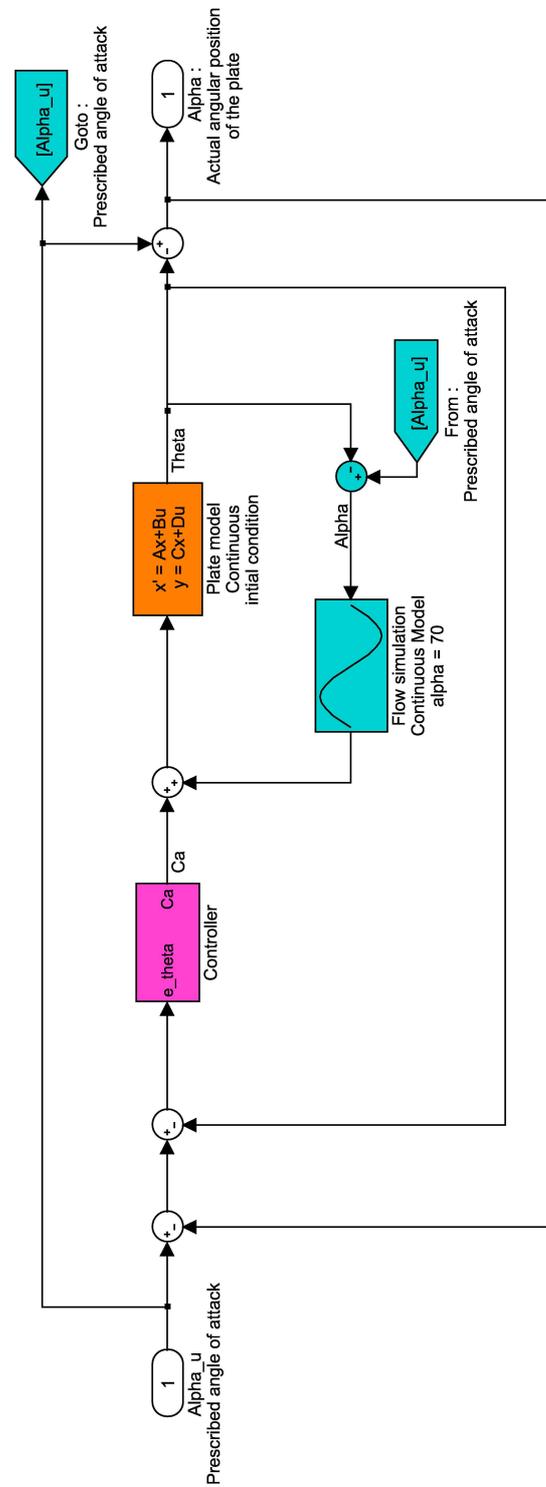


Figure 5.2: Controller, Coupled plate and flow model

5.2.2 Controller specifications

When specifying the controller properties, the goal was mainly to use specifications compatible with the properties of the flow simulation when the controller is coupled with the Spalart flow simulation as in section 5.3. This has stressed to take into account the limitation due to the simulation timestep. Indeed, as presented in section 5.2.5 and in section 4.3.4, the flow simulation can be taken as a discrete model with a sampling frequency, whose internal variables are updated at the given simulation timestep.

It has been found that a large angular velocity can cause problems in equation (2.68), and more generally it creates some singularities in the vorticity field as the body rotates in the flow. Therefore a limitation must be placed on the magnitude of the angular velocity. Due to a lack of time, it was not possible to quantify the limit angular velocity compared to the timestep with flow simulation experiment. Instead, based on the simulation in section 3.2.4.1 the Rossby number Ro was used to define a limiting velocity. First, Ro is taken here as $Ro = |\vec{U}_\infty| / (a * \Omega) = 2$ with $\Omega = \dot{\theta}$ the angular velocity as the simulation has shown a coherent behavior with this parameter. Implicitly, it does also limit the angular acceleration by imposing a limit velocity, but again it was not possible to specify explicitly the angular acceleration due a lack of time. Another limitation comes from the linear flow model, because the controller design uses the model defined in section 4.3.5 which is limited to plate angle of 45 to 90 degrees. These two limitations have mainly be used for the rise time specification.

To come back to the controller specifications, the performance criteria was chosen with the objective of designing a motion control for the plate. Still, as both optimal control techniques and fuzzy-logic control are to be used, the exact specifications for each criteria are not always explicitly taken into account for the controller design. Thus, the criteria stated here are objectives of the design rather than explicit specifications.

The criteria are:

- Minimal rise time t_r , which is the time for the output value to go from 10% to 90% of the input with a step input. It should be chosen so that it is consistent with the flow variables, i.e. mainly the simulation timestep. As seen in section 2.3.7, and section 3.3.2, the used timestep range is between $\Delta t^* = 0.05$ and $\Delta t^* = 0.04$ (see section 5.4). The maximum angle command intended to be used

is $\Delta\alpha = 10$ degrees. Using the previous *Ro* definition and limit, it can be found that $t_r = 0.17 a/U_\infty$.

- Peak response in percent $D_\%$, which is the value of the first maximum with a step input. The maximum peak must be of 1 degree, which since the largest value of α in the range of the control is 90 degrees, corresponds to $D_\% \approx 1\%$.
- Minimal settling time $t_{5\%}$, which is defined as the time after which the output is within 5 percent of the input value, assuming the whole system as stable with a step input. Considering the specified peak response this is equivalent to twice the rise time.
- Zero positioning error $\varepsilon_p = 0$.
- Zero velocity error $\varepsilon_v = 0$.

Other criteria arise in the case of classical control design, but they will not be detailed here, since the classical control plays only a secondary, illustrative role in this study.

In order to quantify the power provided by the controller, the following formula was used :

$$P_c(t) = C_a(t) \times \dot{\theta}(t), \quad (5.2)$$

with P_c the power (in *W*) provided by the system, and C_a the torque provided by the controller, $\dot{\theta}$ is the angular velocity. It is an expression of the instantaneous power and with the controls used here, the power can be negative. Physically, it is similar to the power used by an electrical engine using AC voltage, that is to say when the power is negative it simply means that the engine is giving back stored energy to the environment. To calculate an energy used by the control, the definition is used:

$$E_c(T_1, T_2) = \int_{T_1}^{T_2} [P_c(t)]^2 dt, \quad (5.3)$$

where E_c can be considered as the effective energy used by the controller between the instant T_1 and T_2 . Remark that this equivalent to writing:

$$E_c(T_1, T_2) = \frac{|\overrightarrow{U_\infty}|}{L} \times \int_{T_1^*}^{T_2^*} [P_c(t^*)]^2 dt^*, \quad (5.4)$$

That is to say $E_c(T_1^*, T_2^*) = (L/|\overrightarrow{U_\infty}|) * E_c(T_1, T_2)$. The average power P_a is then defined as $P_a(T_1, T_2) = E_c(T_1, T_2)/(T_2 - T_1)$. Another parameters which is useful is the signal

analysis signal energy $E_s(T_1, T_2)$:

$$E_s(T_1, T_2) = \int_{T_1}^{T_2} [P_c(t)]^2 dt = \frac{|\vec{U}_\infty|}{L} \times \int_{T_1^*}^{T_2^*} [P_c(t^*)]^2 dt^*, \quad (5.5)$$

However, this is only partly related to the physical energy and hence removes part of the physical meaning related to the power and energy, however it does provides some insight regarding the power peak and the smoothness of the power curve.

To provide a wider scope for comparison it is useful to compare this value with the incident flow power impinging on the plate of chord L which can be expressed by $C_p = (1/2)\rho U_\infty^3 L$ for a 2D plate. It is interesting to note the similarity with another factor that could be used for dimensionalization by using $(1/2)\rho(2a)^2 U_\infty^2$ (see equation 3.4) and U_∞/a (in reference to the Rossby number). The number used is then similar to the flow power provided by a moment applied to on ellipse tip, the factor can then be written as $(1/2)\rho(2a)^2 U_\infty^2 \times U_\infty/a = 2\rho a U_\infty^3 = 2C_p$. This lead us to the definition to get a nondimensional power P_c^* , signal energy E_s^* and energy E_c^* :

$$P_c^* = \frac{P_c}{C_p}, \quad (5.6)$$

$$E_c^*(T_1^*, T_2^*) = \int_{T_1^*}^{T_2^*} [P_c^*(t^*)]^2 dt^*, \quad (5.7)$$

$$E_s^*(T_1^*, T_2^*) = \int_{T_1^*}^{T_2^*} [P_c^*(t^*)]^2 dt^* \quad (5.8)$$

For commodity reason, the intermediary definition for the energy is also used:

$$E_{c,p}(T_1, T_2) = \int_{T_1}^{T_2} [P_c^*(t)]^2 dt, E_{s,p}(T_1, T_2) = \int_{T_1}^{T_2} [P_c^*(t)]^2 dt \quad (5.9)$$

$E_{c,p}$ could be defined as the energy provided by the nondimensional power in real time, the same is true for $E_{s,p}$.

5.2.3 Classical control theory

Using the plate and flow system used in sections 4.3.5, 4.3.6 and presented in section 4.4, trials were done with several methods: root locus, simple pole placement, PI (Proportional Integral), PID (Proportional Integral Derivative), PI coupled with a derivative feedback, and proportional controller. For those cases, Simulink was used with the SISO design tool provided with Matlab.

Other methods of pole placements, such as the model following, for example, are not attempted here because they are equivalent to more modern optimal control techniques, and Matlab functions are not available for these older techniques.

The robustness of the controller obtained is evaluated using the gain margin and the phase margin. One could also use other methods involving sensitivity function (magnitude of $1/(1 + C(s).G(s))$ with $C(s)$ the control transfer function) magnitude plots to assess the robustness of the method.

Considering the cases in section 4.4, the PI, and PID techniques were not sufficient to ensure stability for each individual case. A typical classical root locus technique was then used with one real pole, three real zeros and two integrators so as to ensure zero position error, and zero velocity error. The different poles and zeros were not affected by pole compensation, except for the resonance case. Because the final coupled system was with non minimal delays, it was sometimes difficult to assess whether it was stable or not considering only the Bode diagram, so that the root-locus diagram had also to be used. Additional difficulties appeared when the C_m frequency corresponded to ϖ_n . However, this turned out to be a side effect due to the C_m linear model. In most cases, the sensitivity function plot was satisfactory. There remains however for future studies the question of whether the control is physically reasonable, since the poles tend to be rejected far away in frequency.

For one simplified closed loop system, that is a control and the coupled plate/flow model described in section 4.4, one can then obtain:

- $\varepsilon_p = 0$ and $\varepsilon_v = 0$ due to the two integrators
- $t_r = O(10^{-2})$ s
- $D_{\%} < 1\%$
- $t_{5\%} = O(10^{-2})$ s

However, the relevance of these characteristics is difficult to assess as the coupled plate/flow system is not linear, and since there remains the simulation of the full gain scheduling option with the flow model used in section 4.3.5. Also refer to table 5.1 and its comment at the end of the section for further result characterization.

This flow model used the polystyrene plate parameter. Then as stated in section 5.2.1, when using classical control techniques, the control used was similar to a Takagi-Sugeno global controller; i.e. there are several linear controllers designed for each of the angles of attack considered (50, 60 and 70 degrees here) which are afterwards linearly interpolated (no model frequency change described in section 4.3.3) using a fuzzy logic box. The full control setup is shown in figure 5.3.

More precisely, each controller has thus been designed based on the flow model identification at a given angle used for the steady angle flow model in section 4.3.5. Every controller have transfer functions with the same structure (rational function with order 3 polynomials). The controller interpolation was done using a similar technique to that used for the steady flow model without the model frequency change. Like in section 4.3.5, a fuzzy logic box was used as an interpolation routine for the classical controller. The different controllers are interpolated as a function of the angle of attack α . Note that the “fuzzy box” in figure 5.3 is only the fuzzy logic box used for the interpolation. It is totally different from the fuzzy controller used in section 5.2.5.

Afterwards, in order to test this controller, a control was designed and used for an angle of 55 degrees and $\rho_b = 35 \text{ kg/m}^3$ ¹ Afterwards, a step input of one degree was applied as a reference signal. The full systems (control and coupled plate/flow system) setup is given in figure 5.4. The results for the θ (theta in Simulink diagram) values are presented in figure 5.5 for a step input of 10° at $t = 5s$, the error signal is plotted in figure 5.6 and the resulting control signal u is illustrated in figure 5.7.

For the sake of precision, the control torque minus the torque required by the spring is also plotted 5.8, in that case there is little difference compared to the other controller case (in section 5.2.4 and 5.2.5). This is simply due to the calculus of k which requires a value of ρ_b to obtain the correct ϖ_n , remember that the value used in the other case for the simulation is $\rho_b = 1200$ to compare with $\rho_b = 35$ in this section. In order to compare with result of section, the definition and parameters of section 5.4 are used as well as the torque definition for C_e in equation 5.19.

¹At the time of the thesis writing, it has not been possible to modify the control so as to take into account the case $\rho = 1200$.

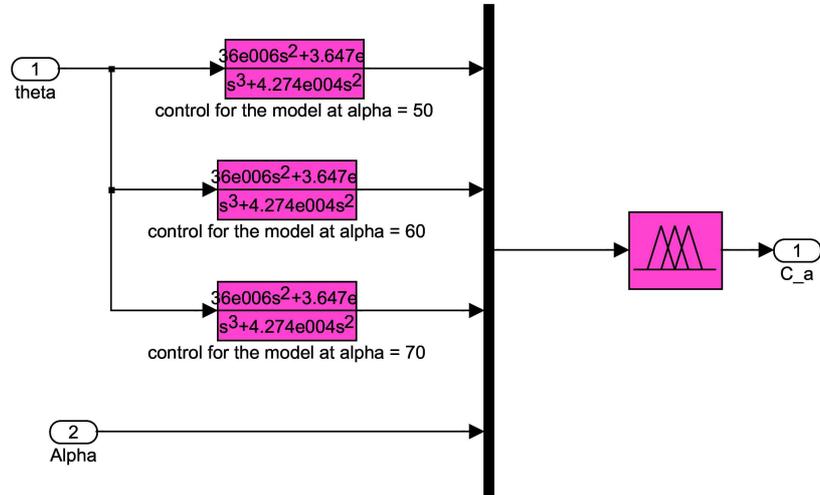


Figure 5.3: Classical based, linear controller block diagram

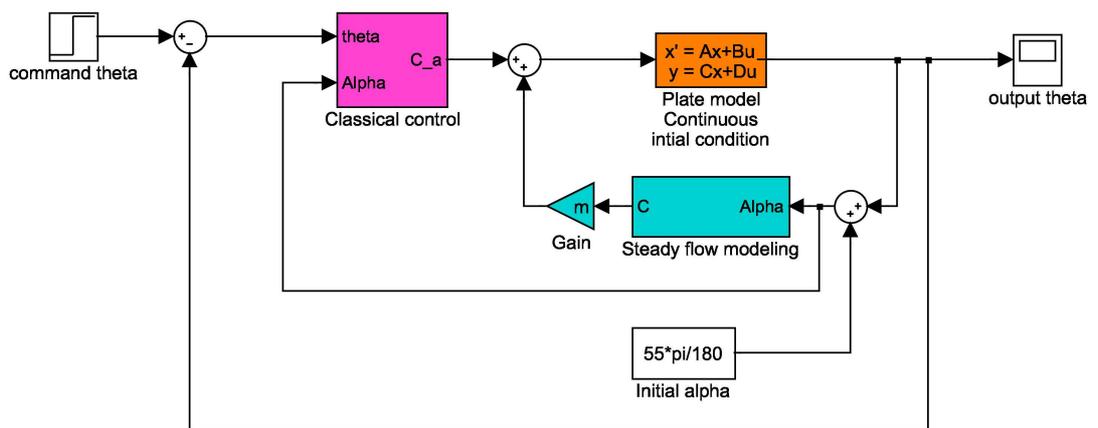


Figure 5.4: Control and system closed loop block diagram for the classical controller.

Finally, the power divided by C_p (using the definition in section 5.2.2) is shown in figure 5.9 so as to provide an estimate of the power used by the control in that case. A result summary is provided in table 5.1

From figures 5.5 and 5.6, one can see that the control is successfully able to stabilize the plate with a precision of roughly $\pm 0.0036^\circ$ once the step has been passed which is confirmed to the *RMS* value in table 5.1 when $t > 5.2s$. Astonishingly –considering the *RMS*– the control precision seems better during the motion than when the plate is stabilized, however this must be tempered because it is then close to the simulation precision.

To complete the power plot (in figure 5.9), one should use table 5.1 in order to assess the amount of power used, the energy and how the power is used.

	time t (s)			
	$t = [0, 30]s$	$t = [0, 5]s$	$t = [5, 5.2]s$	$t = [5.2, 30]s$
<i>ErrorRMS</i> (in degree)	0.013383	0.002897	0.15785	0.0021883
$E_{c,p}$	-619.54	-3.4228×10^{-7} $5.52 \times 10^{-8}\%$	-619.54 > 99%	3.0833×10^{-5} $-4.98 \times 10^{-6}\%$
P_a^*	-20.651	-6.8457×10^{-8}	-2961.7	2.4877×10^{-11}
$E_{s,p}$	6.4785×10^{11}	6.7539×10^{-10} $1.04 \times 10^{-19}\%$	6.4785×10^{11} > 99%	6.1671×10^{-10} $9.52 \times 10^{-20}\%$

Table 5.1: Summary of the control results for the classical controller. The power and energy are calculated from the power divided by C_p and t (see equations 5.6 and 5.9). The % are compared to the total energy. The *rms* is the common *RootMeanSquare* formula.

Observe that the negative $E_{c,p}$ and E_c^* originate from the energy “given back” by the coupled spring damped plate/flow to the system. Also remark the high $E_{s,p}$ and E_s^* values, this is not that surprising considering that the motion asked is precisely asking for infinite power (the derivative would be a Dirac). Although the controller manages to give good results regarding the precision, one should also consider the energy $E_{s,p}$, E_s^* , $E_{c,p}$ and E_c^* and power during the motion ($t = [5, 5; 2]s$) as it shows that there is large power peak during the motion and that much of the power is also coming from

the spring as shown by the negative average power. As spring nonlinearities are not implemented, it is difficult to assess the exact physical reaction of the system. It mainly means that there would be here a physical limitation due to the physical environment rather than the controller and that it may induce a lot of stress on the spring/damped plate. Finally, remark that for stabilization ($t < 5s$ and $t > 5.2s$), the power remain remarkably low which seems normal as there is very little plate motion.

Incidentally, to provide a comparison regarding section 5.3, those values are also given using t^* in table 5.2.

	Time t^* using $T^* = T \times a/U$ and section 5.4.2 parameters			
	$t^* = [0, 103.4]$	$t^* = [0, 17.2]$	$t^* = [17.2, 17.9]$	$t^* = [17.9, 103.4]$
E_c^*	-2136.3	-1.1803×10^{-6} $5.52 \times 10^{-8}\%$	-2136.3 > 99%	0.00010632 $-4.98 \times 10^{-6}\%$
P_a^*	-20.651	-6.8457×10^{-8}	-2961.7	2.4877×10^{-11}
E_s^*	2.234×10^{12}	2.3289×10^{-9} $1.04 \times 10^{-19}\%$	2.234×10^{12} > 99%	2.1266×10^{-9} $9.52 \times 10^{-20}\%$

Table 5.2: Summary of the control results. The power and energy are calculated from the power divided by C_p and t^* (see equations 5.6). The % are compared to the total energy. The *rms* is the common *RootMeanSquare* formula.

It thus appears that at this angle, the system is stable and does comply with the specifications, with $t_r = 1.25e^{-4}$ s, $D_{\%} = 0.18\%$, and no static error. In this particular case, a single linear control is sufficient. However, it is not known if this controller would be physically feasible, as it would be certainly requires to integrate explicitly an actuator model.

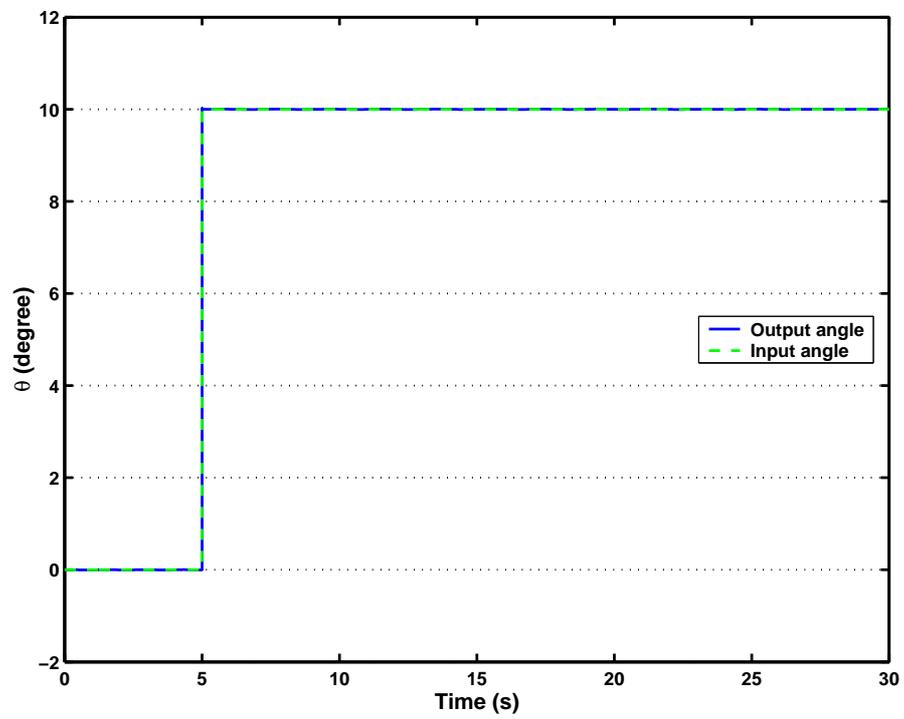
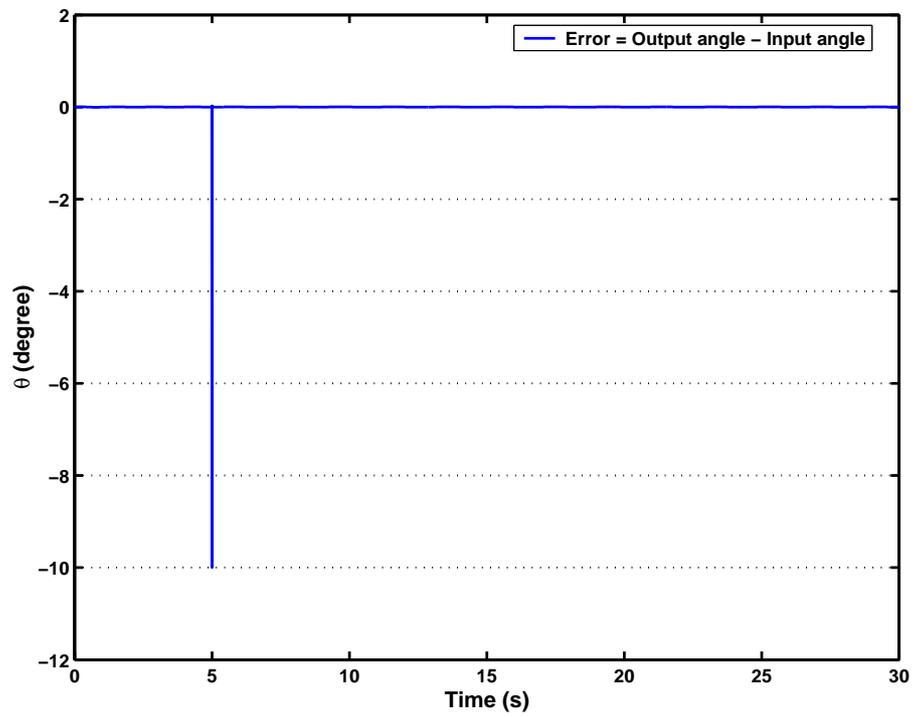
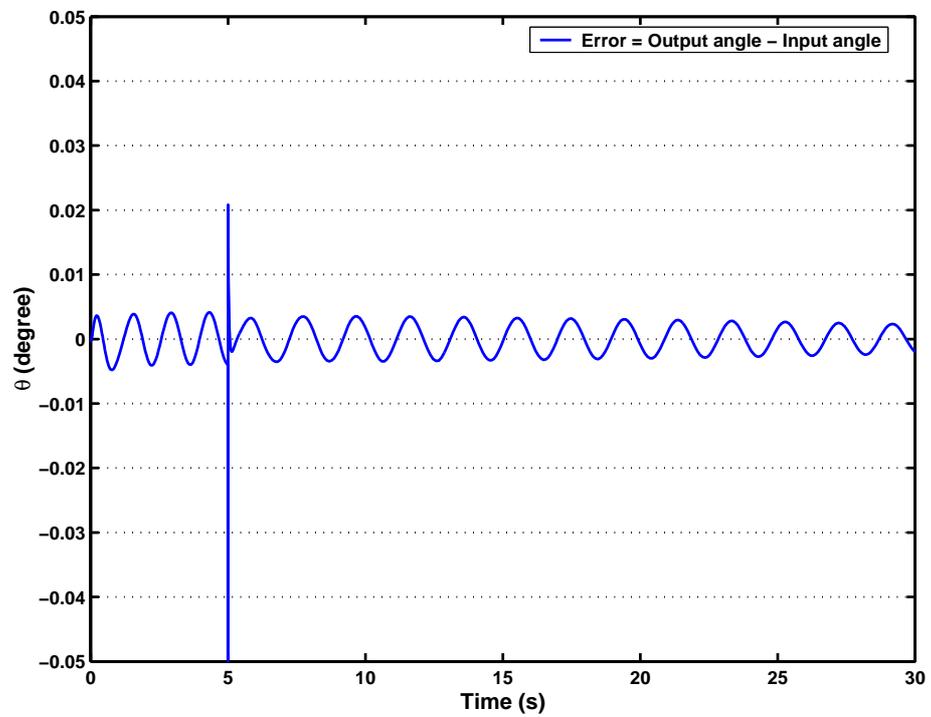


Figure 5.5: Output of the closed loop system for a step input for the classical controller.

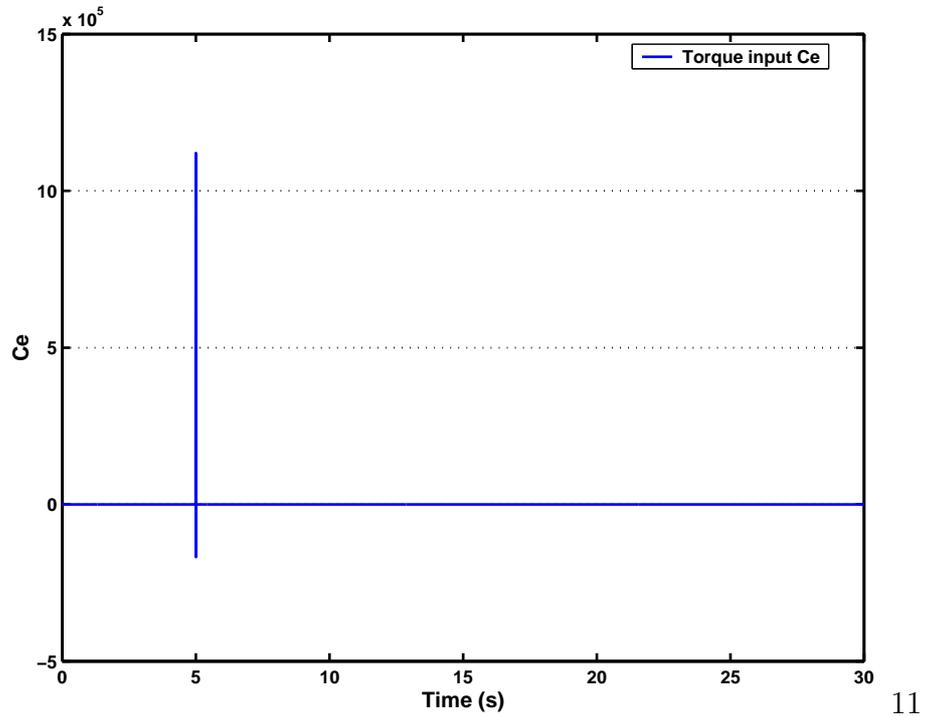


(a)

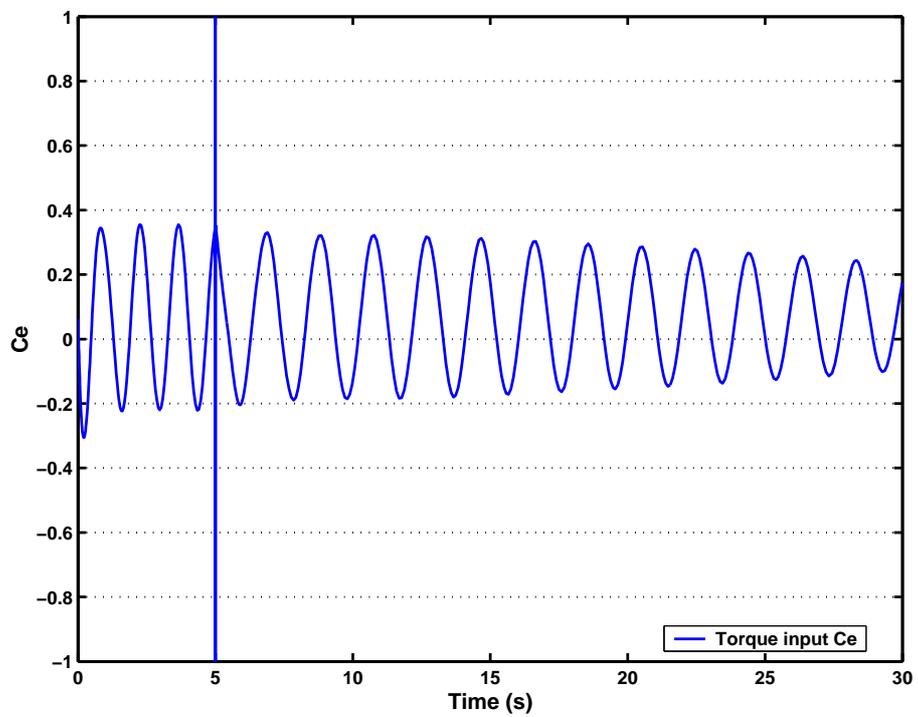


(b) Close-up view of the error

Figure 5.6: Error of the closed loop system for a step input for the optimal controller.

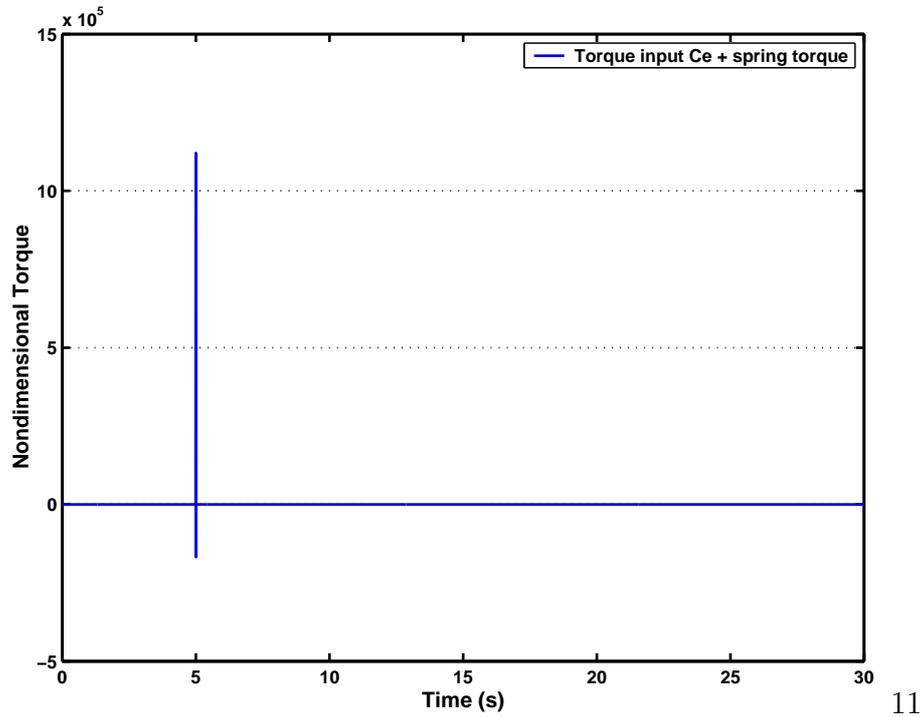


(a)

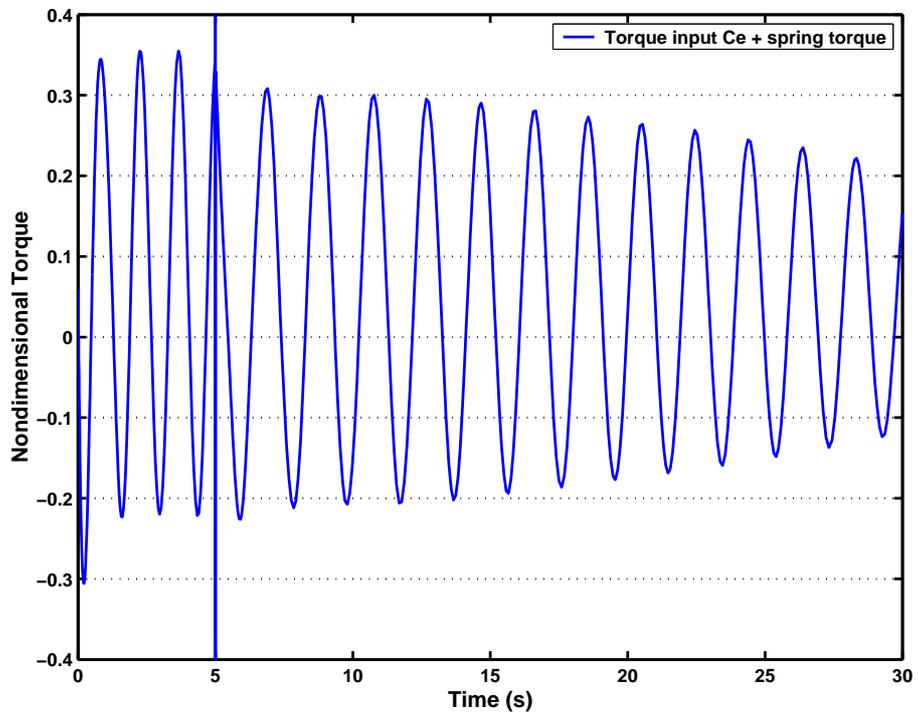


(b) Close up of the control signal

Figure 5.7: Control signal evolution in time for the classical controller (the vertical line is actually a control input peak due to the step input). Cf C_e definition in eq. 5.19.

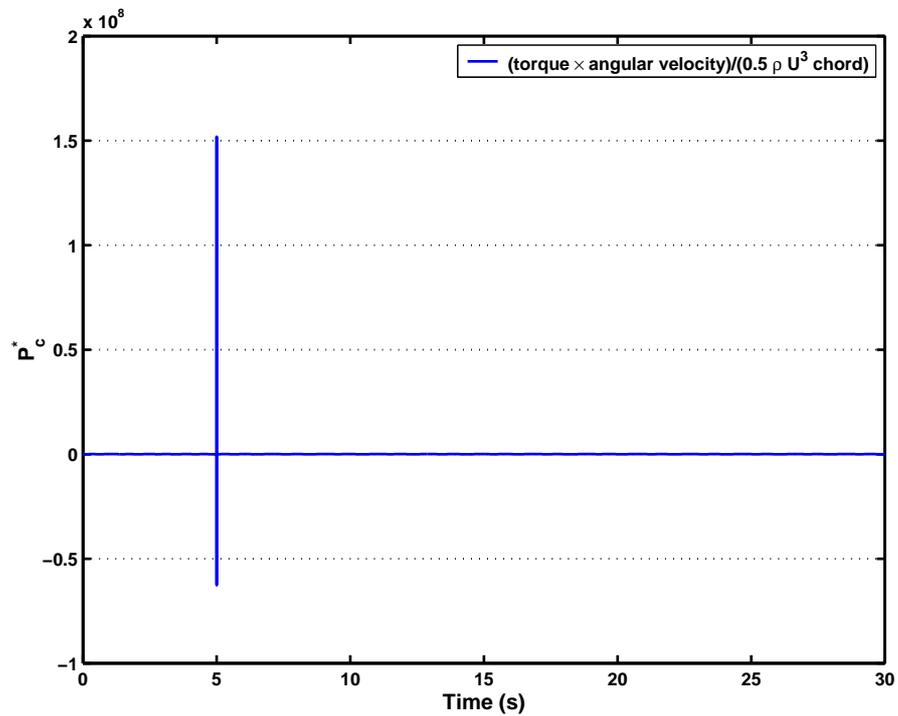


(a)



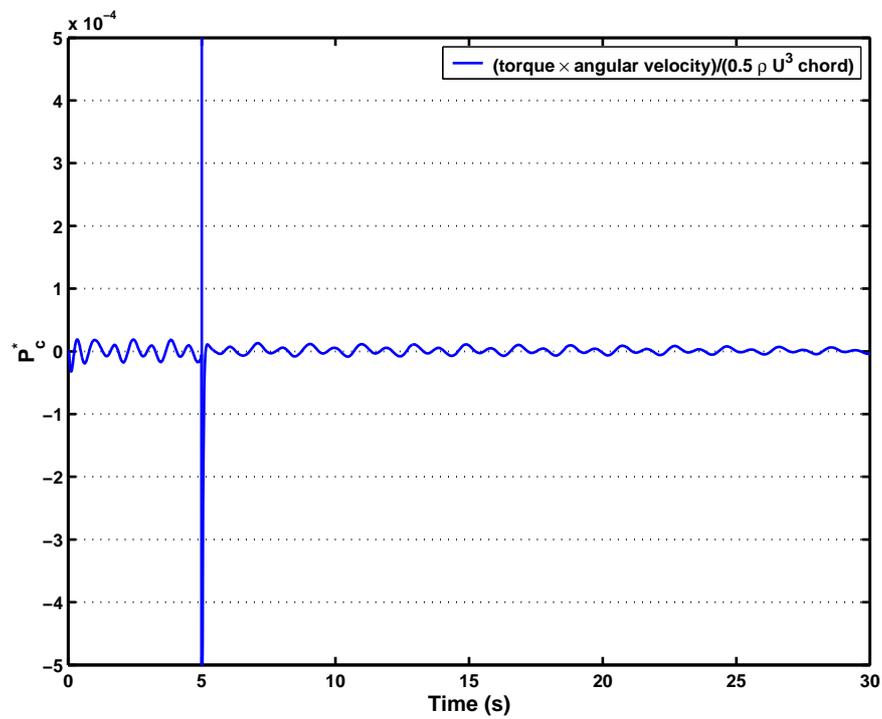
(b) Close up of the control signal minus the torque due to the spring

Figure 5.8: Control signal evolution in time for the classical controller without taking into account the torque due to the spring. Cf C_e definition in eq. 5.19 for the non-dimensionalization.



11

(a)



(b) Close up of the power signal

Figure 5.9: Power signal evolution in time for the classical controller. The power is divided by $(1/2)\rho U_\infty^3 L$

5.2.4 Optimal control

As for the previous method, this is a Takagi-Sugeno control, but this time for each angle of reference optimal control theory is used to design the control according to the plate and flow models used in sections 4.3.5, 4.3.6 and presented in section 4.4. More specifically, a LQR (Linear Quadratic Regulator) technique was used with an optimal regulator and an estimator with full state feedback.

Using the separation principles, one first design the regulator:

$$\underline{U} = \underline{R}_c - K \underline{X}, \quad (5.10)$$

where \underline{U} is the command signal, \underline{X} the state signal, and both are vectors; K is the gain matrix, and \underline{R}_c the reference command signal, a column vector. In a SISO system, u and r are a scalar and K a line vector. The LQR design simply consists of finding K such that the criterion:

$$J_{LQR} = \int_0^{\infty} \underline{X}'(t)Q\underline{X}(t) + \underline{U}'(t)R\underline{U}(t)dt, \quad (5.11)$$

is minimized, where (using for simplicity SISO system notation) $x'(t)Qx(t)$ is the state cost, and $u'(t)Ru(t)$ the control cost. One call Q and R respectively the state and cost penalty matrices. It was chosen to put Q in the form $Q = \lambda C' C$ with C defined in the system under state-space form:

$$\begin{aligned} \dot{\underline{X}} &= A \underline{X} + B \underline{U} \\ \underline{Y} &= C \underline{X}, \end{aligned} \quad (5.12)$$

where \underline{Y} is the output signal. For SISO system, y is a scalar and C a line vector. Therefore, $Q = \lambda C' C$ enables to write $\underline{X}'(t)Q\underline{X}(t) = \lambda \underline{Y}'(t)\underline{Y}(t)$. The choice was made of $norm(Q) > norm(R)$ because as the system is close to auto-oscillation and also the plate/flow system can be unstable, the effort in terms of control of the output is justified in order for the system to meet the specification.

The following estimator was used:

$$\begin{aligned} \dot{\underline{X}}_e &= A \underline{X}_e + B \underline{U} + L (\underline{Y} - \underline{Y}_e) \\ \underline{Y}_e &= C \underline{X}_e, \end{aligned} \quad (5.13)$$

where L is the selectable gain matrix. L was tuned such that the poles of the polynomial $\Phi_0(s) = det(Is - A + LC)$ are mapped to that of a Bessel polynomial scaled so that

the dynamics of the estimator is faster than the system dynamics, i.e. the estimation error is cancelled faster than the system dynamics. Note that when this method fails to produce a useful estimator – it leads to a stiff compensator function –, a Kalman filter was used to account for the fact that some noise is introduced in the true system. This is represented by the formula:

$$\begin{aligned}\dot{\underline{X}} &= A \underline{X} + B \underline{U} + B \underline{W}, \\ \underline{Y} &= C \underline{X} + \underline{V},\end{aligned}\tag{5.14}$$

where \underline{W} is the process noise, while \underline{V} is the sensor noise; they are column vectors with the same number of rows as respectively \underline{X} and \underline{Y} . Consequently, in a SISO system both are scalars. \underline{R}_w is noted as the covariance of \underline{W} , and \underline{R}_v as the covariance of \underline{V} . Then usually to produce a correct system, one needs $\underline{R}_w > \underline{R}_v$. This is because it produces a high bandwidth estimator, thus the estimator dynamics are actually quicker than the system dynamics. A side effect is that it assumes there is relatively little noise from the output sensor, therefore the actual implementation may require some adjustments. Nevertheless, the Kalman filter approach enables to easily design the estimator. There are other methods for the design of an optimal estimator (see MacLean 1990 [38], MIT course [28]), but it was not deemed necessary to consider them because the Kalman filter approach lead to an acceptable design.

For the full compensator, use was made of an approach from the MIT [28]. The compensator is then implemented using the formula:

$$\begin{aligned}\dot{\underline{X}}_c &= A_c \underline{X}_c + L \underline{Y} + B N_c \underline{R}_c, \\ \underline{U} &= -K \underline{X}_c + N_c \underline{R}_c.\end{aligned}\tag{5.15}$$

with \underline{X}_c the state variable of the controller which is a vector equivalent to $\underline{X}_c \equiv \underline{X}_e$, A_c is a matrix equal to $A_c = A - B K - L C$, and N_c is designed so that

$$\frac{\underline{Y}}{\underline{R}} \triangleq [C \ 0] \left(- \begin{bmatrix} A & -B.K \\ L.C & A_c \end{bmatrix}^{-1} \right) \begin{bmatrix} B \\ B \end{bmatrix} N_c = 1.\tag{5.16}$$

Such a design enables for a given model a zero steady-state error, and better transient characteristics (thanks to the elimination of the estimator dynamics). To have an equivalence with the steady-state C_m model, the angles considered were: 45, 50, 60, 70, 80, and 90 degrees. Using the case with $\rho_b = 35$ from section 4.4 for each angle, it was impossible to attain the $D\%$ criterion unless $norm(Q) \gg norm(R)$. The $\varepsilon_p = 0$

specification could be met, but $\varepsilon_v = 0$ cannot be matched exactly unless an integral feedback is added. However, the other specifications were met.

Like the classical controller implementation in section 5.2.3, the control is similar to a Takagi-Sugeno global controller to cover several angles of attack. There are several linear controllers designed for each of the angles of attack considered (50, 60 and 70 degrees here) which are afterwards linearly interpolated (no model frequency change described in section 4.3.3) using a fuzzy logic box. The fuzzy logic box (in figure 5.10) is only used as an interpolation routine for the different optimal controllers and is different from the fuzzy controller used in section 5.2.5. The controllers are interpolated as a function of the angle of attack α . Again, every controller have transfer functions with the same structure. The control setup is illustrated in figure 5.10 and the full setup (control and flow/plate model) is presented in figure 5.11.

To test the controller, a slightly different case was used compared to the previous section in order to better compare these results with results in section 3.3.2 and 5.4. Thus here the angle is of 55 degrees with a step input of 10 degrees and an intermediary body density of $\rho_b = 1200 \text{ kg/m}^3$. f_n^* was chosen as $f_n^* = S_s$, which enables to have the flow near the spring-damped plate resonance frequency even when the plate is kept stable.

A short results summary is provided in tables 5.3 and 5.4. In order to provide a frame for comparison with results of section 5.3, the table results using t^* is also given in table 5.4. The results concerning θ values are shown in figure 5.12 for a step input of 10° at $t = 5s$. Note the input angle (dash line in figure 5.12) representing the prescribed angle, it is simply here for the sake of clarity. The error between the prescribed angle output and the angle input is shown in figure 5.13.

The corresponding control signal u is presented in figure 5.14. To remove the constant torque required by the spring, the controller torque minus the torque spring is also plotted in 5.15. In order to compare with result of section, the definition and parameters from section 5.4 are used as well as the torque definition for C_e in equation 5.19.

Finally, the power divided by C_p is plotted in figure 5.16. Other trials with $\rho_b = 35 \text{ kg/m}^3$ have shown very similar results, in particular the residual oscillation visible in

5.13 seems to be closely linked to the flow system rather than to the spring damped plate.

Note the control signal peak in figure 5.14 and 5.15, this is actually due to the step input, as there is no angular velocity limit here. In other words, the angular velocity is at the time $t = 5s$ of the step input: $\dot{\theta}(t = 5s) = \infty$. As the plate/flow system is controlled with a torque input, theoretically an infinite amount of power would be necessary for the angular motion. The reason why the peak is not strictly vertical and does not go to infinity is because of the frequency bandwidth limitation of the controller (similarly to a band-pass filter), and the plot resolution. This peak was thus expected. Consequently, the -10° angular error in figure 5.13 peak simply represents the fact that in contrast to the prescribed signal, the response is not instantaneous.

Once the influence from the spring is removed, the differences are most noticeable in figures 5.14(b) and 5.15(b) where one can see the permanent torque due to the spring. However it does not influence much the torque during the motion as it is only linked to the plate position and not to the plate angular velocity. Hence its influence is somewhat reduced in the current study.

One can remark a slight steady state error past the step input. Otherwise, the system seems to have well followed the step. The rise time is $t_r = 0.05$ s and $D_{\%} \approx 4\%$. Further tests show that the system remains stable over 1600 seconds, although there remains a small oscillation of amplitude of about ± 0.02 degrees as well as a position error (or steady state error) of -0.002 degrees. The remaining oscillations simply reflect the linear model used for the flow model which is close to a lightly damped harmonic oscillator. Indeed, the compensator reduces the harmonic influence by using pole compensation, but because the pole causing the harmonic is very lightly damped it is difficult to fully compensate it hereby causing small oscillations but not affecting the system stability. The position error is mainly a numerical effect as it is computed as an average over the time range shown in figure 5.12. It is thus simply a consequence of the decreasing oscillations.

When regarding tables 5.3 and 5.4, first considering the power P_a^* , it is considerably reduced during the motion compared to the classical controller as well as the fuzzy controller. Again, P_a^* is negative which denotes that the controller uses part of the

spring. Also regarding E_s ($E_{s,p}$ and E_s^*), it shows that there is less fluctuations in the power which is somewhat more realistic than for the classical controller. The negative power indicate is to be put in parallel to the torque plot and the peak torque observed. It denotes that past the peak torque, the controller is mainly controlling the spring provided by the spring rather than from the controller.

	time t (s)			
	$t = [0, 30]_s$	$t = [0, 5]_s$	$t = [5, 5.2]_s$	$t = [5.2, 30]_s$
<i>ErrorRMS</i> (in degree)	0.29153	0.012034	3.5713	0.01047
$E_{c,p}$	-0.20586	-2.7365×10^{-5} 0.0133%	-0.20576 > 99%	-6.5998×10^{-5} 0.0321%
P_a^*	-0.0068619	-5.4729×10^{-6}	-1.0307	-2.6612×10^{-6}
$E_{s,p}$	26.521	1.2338×10^{-9} $4.65 \times 10^{-9}\%$	26.521 > 99%	5.6513×10^{-7} $2.13 \times 10^{-6}\%$

Table 5.3: Summary of the control results for the optimal controller. The power and energy are calculated from the power divided by C_p and t (see equation 5.6 and 5.9). The % are compared to the total energy. The *rms* is the common *RootMeanSquare* formula.

Now regarding the small remaining oscillations there are about 10 times larger than for the classical controller. It is also visible if one consider the error *RMS* which is 0.01047° compared to 0.0021883° for the classical controller (tables 5.3 and 5.2). It thus seems at first that there is a tradeoff to be expected between the power used –instantaneous but also the average power– and the precision.

As for the energy used during the motion (again using the power divided by C_p), one obtain a total energy $E_{c,p}$ of -0.206 and $E_{s,p} = 26.521$ with an energy used between $t = 5$ and $t = 5.2$ of more than 99% of the total energy. Past $t = 5.2$, $E_{c,p}$ and $E_{s,p}$ becomes both about $O(10^{-5})$. In that respect, the classical controller was a bit more efficient for stabilization as the signal energy E_s ($E_{s,p}$ and E_s^*) was much more concentrated around $t = [5, 5.2]_s$. It is nevertheless difficult to judge due to the huge power peak in this range which impairs artificially the calculus (the ratio comes close to the machine precision). As for the classical controller, the average power P_a used for stabilization remains very small compared to C_p .

	Time t^* using $T^* = T \times a/U$ and section 5.4.2 parameters			
	$t^* = [0, 103.4]$	$t^* = [0, 17.2]$	$t^* = [17.2, 17.9]$	$t^* = [17.9, 103.4]$
E_c^*	-0.70985	-9.4361×10^{-5} 0.0133%	-0.70953 > 99%	-0.00022758 0.0321%
P_a^*	-0.0068619	-5.4729×10^{-6}	-1.0307	-2.6612×10^{-6}
E_s^*	91.45	4.2546×10^{-9} $4.65 \times 10^{-9\%}$	91.45 > 99%	1.9487×10^{-6} $2.13 \times 10^{-6\%}$

Table 5.4: Summary of the control results. The power and energy are calculated from the power divided by C_p and t^* (see equation 5.6). The % are compared to the total energy. The *rms* is the common *RootMeanSquare* formula.

Like the fuzzy logic controller, it is difficult here to assess fully the stability and robustness of the controller shown in figure 5.10 as the control law is effectively the weighed mean value of two controller outputs because of the interpolation. This means that the full controller does not take into account the variation in frequency of the plate/flow model or the effect of transition from one compensator model ² to another within the controller (figure 5.11). Nonetheless, as for the flow model (figure 4.17), the transition from one compensator to another is done smoothly. Furthermore, all the flow models are based on a lightly damped harmonic oscillator, thus the compensators are all based on the same kind of regulator, that is a combination of a “lead” compensator and pole compensation at the harmonic frequency. It is then reasonable to assume that the nonlinearities will have reduced influence on the system stability, and that it will more likely affect the system performance such as the remaining oscillation amplitude or the rise time. Comparison tests over 1600 seconds with step input of 10 and 30 degrees for different initial angles and angular velocity have shown that this idealized system remains stables over different conditions.

This approach is similar to the Takagi-Sugeno models – use of multiple linear models and weighed means to determine the output – and leads to results less satisfying than when using gain scheduling (see section 5.2.1 and Athans [3]). Usually for a gain-scheduling controller, one usually adjust the different parameters of the controller, that

²The compensator model is actually the name of the individual controller in figure 5.10.

is to say for example for a controller of the form :

$$G(s) = K_c \left(1 + \frac{1}{T_i s} + T_d s \right), \quad (5.17)$$

where K_c , T_i , and T_d are the parameters of a classical *PID* controller, $K_c(x)$, $T_i(x)$, and $T_d(x)$ would be adjusted according to a variable x adapted to the controlled system. In the current case, such a variable would be α for example. Now, for a Takagi-Sugeno controller, one do not adjust the controller output but interpolate the output magnitude considering a variable x adapted to the controlled system. Thus, such an implementation neglects the variation in frequency or the variation of the natural mode of the controlled system. A remedy would be to modify the timescale, i.e apply the modification described in section 4.3.3 to the compensator system matrices described in equation 5.15 in order to try to follow the frequency evolution. Remark that this would then be equivalent to gain scheduling method. As the focus is on the fuzzy control, the frequency change feature was not implemented for the full controller (figure 5.10).

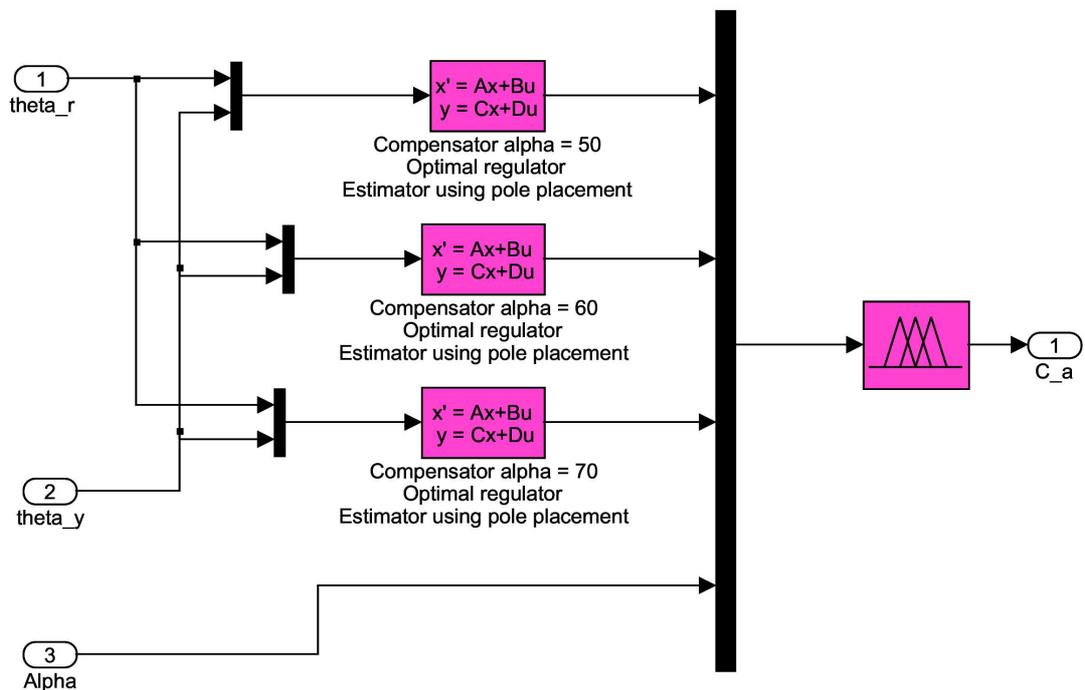


Figure 5.10: Optimal, linear controller block diagram

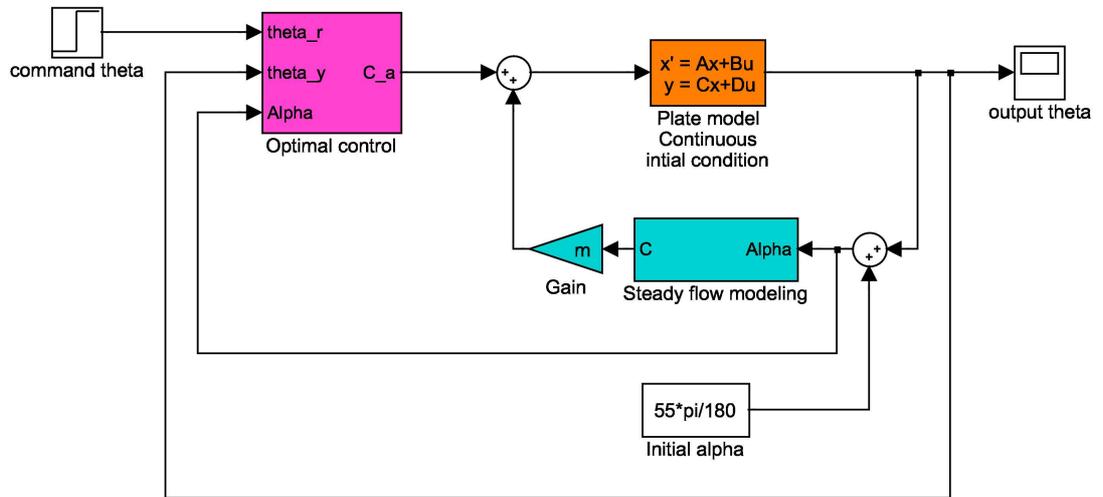


Figure 5.11: Control and system closed loop block diagram for the optimal controller.

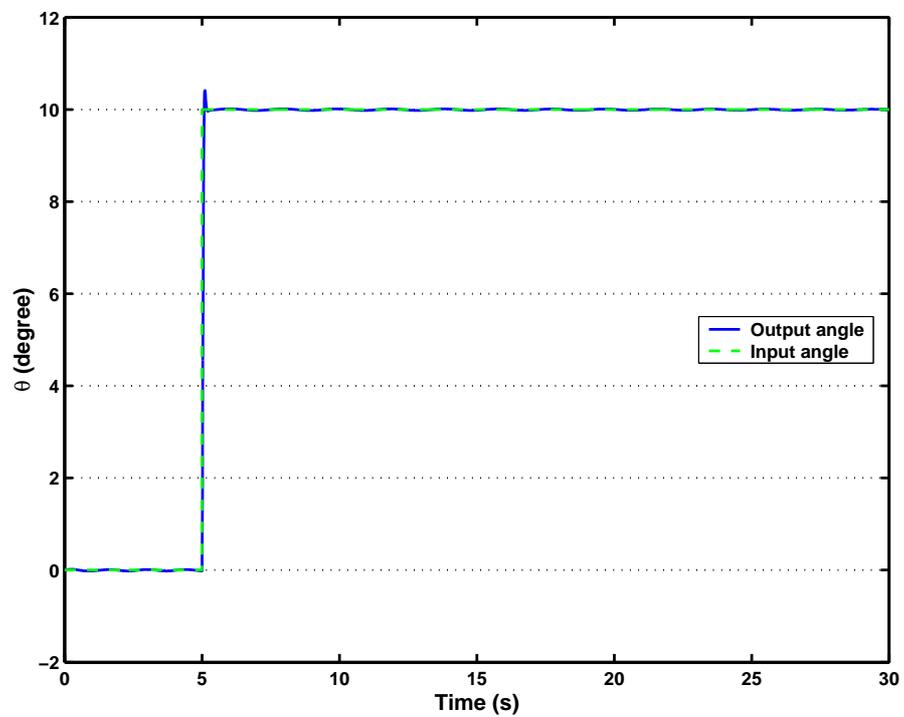
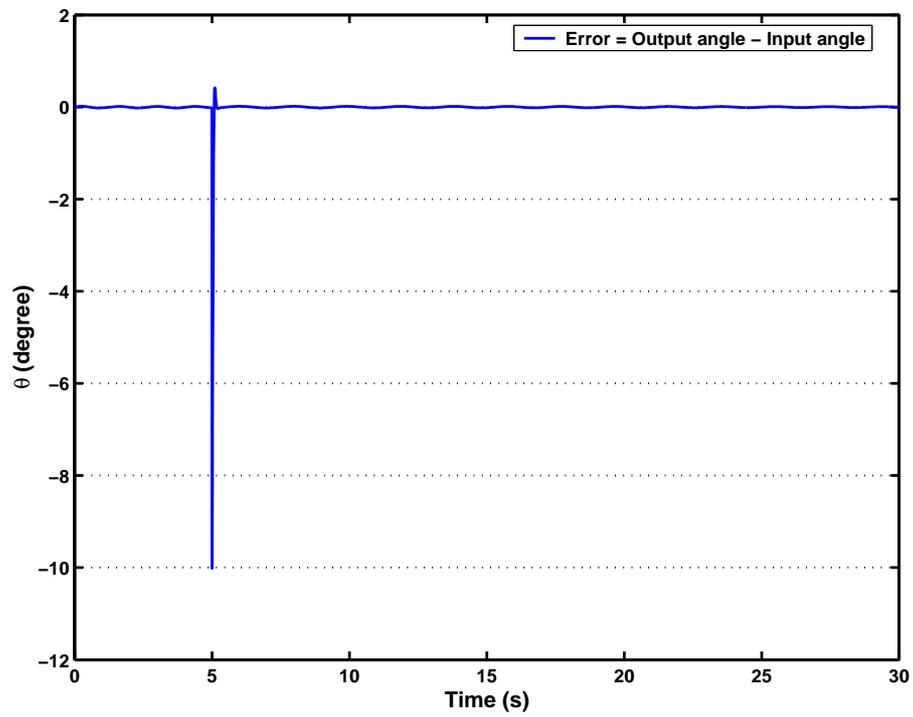
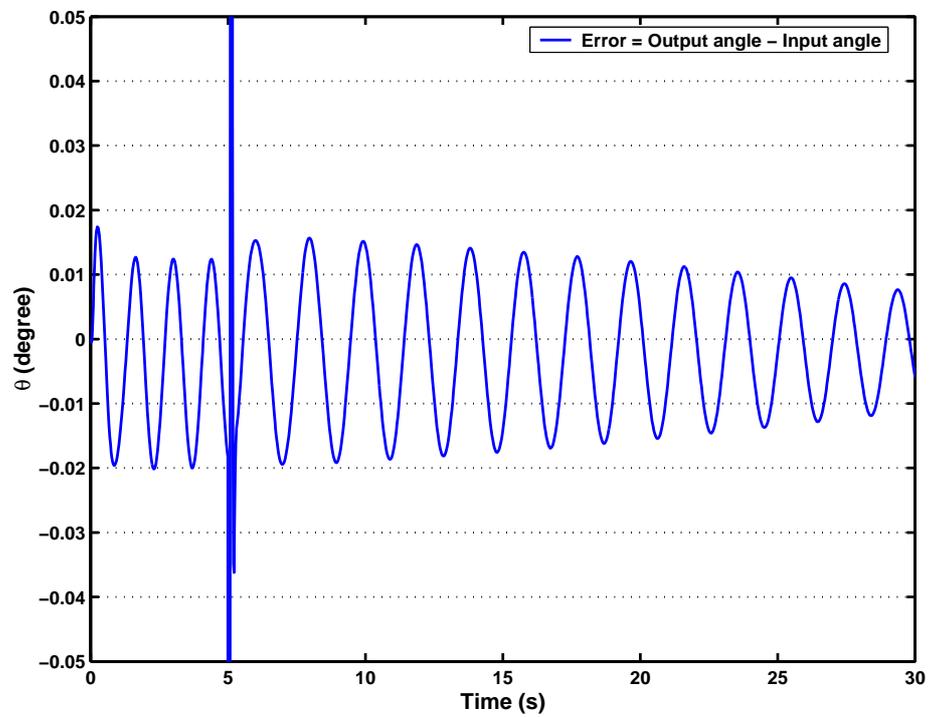


Figure 5.12: Output of the closed loop system for a step input for the optimal controller.

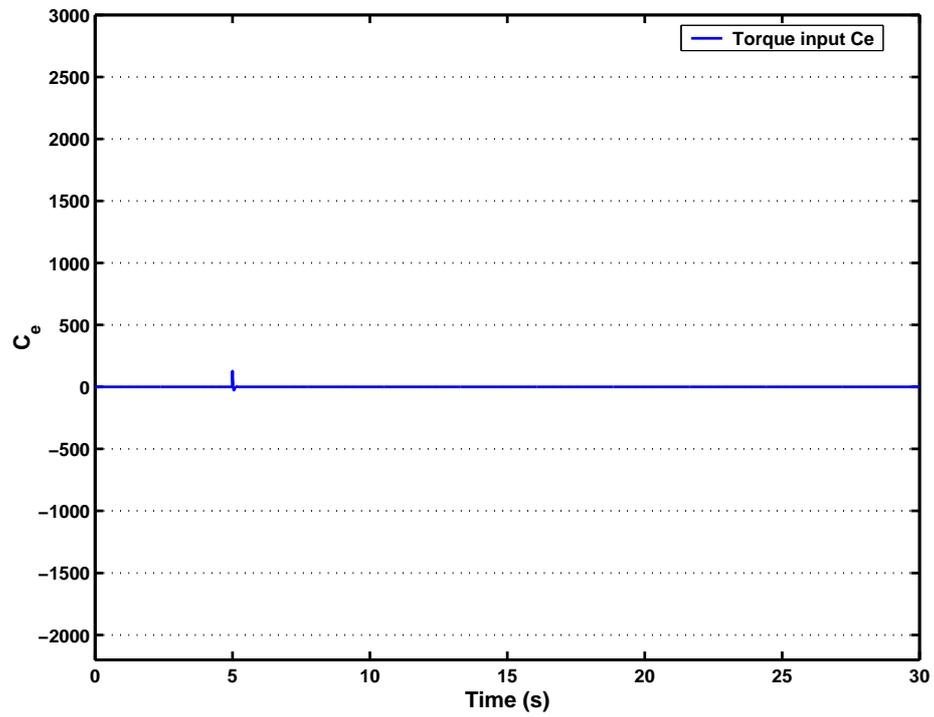


(a)

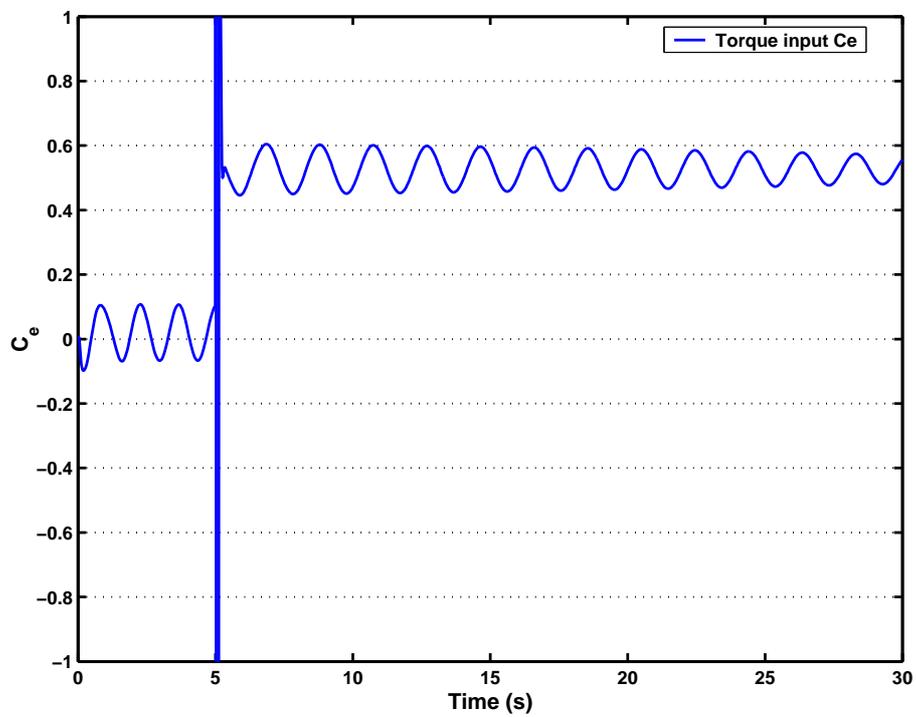


(b) Close-up view of the error

Figure 5.13: Error of the closed loop system for a step input for the optimal controller.

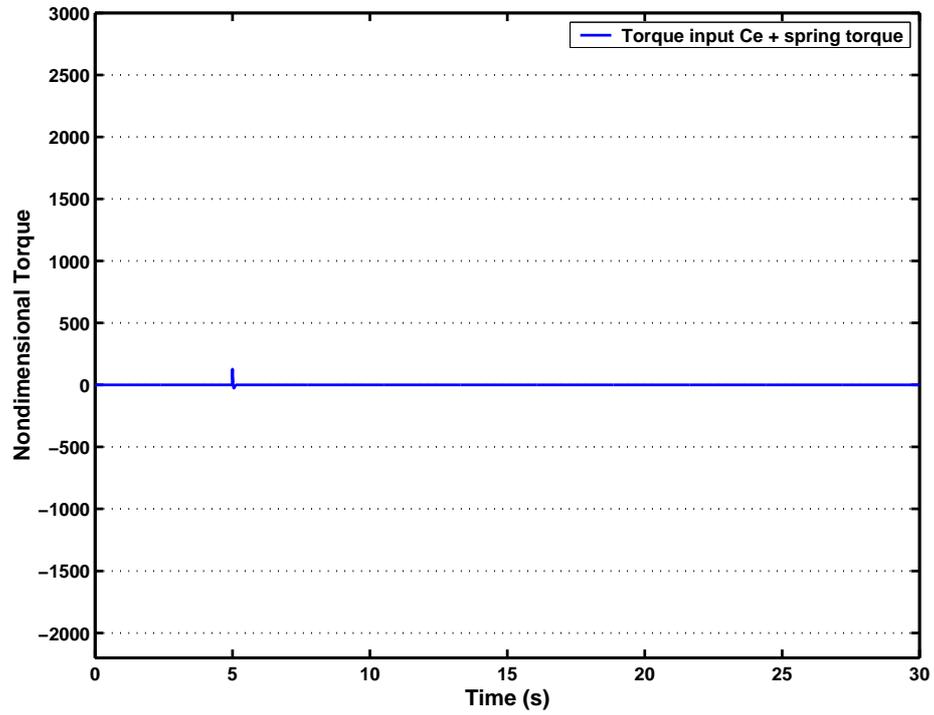


(a)

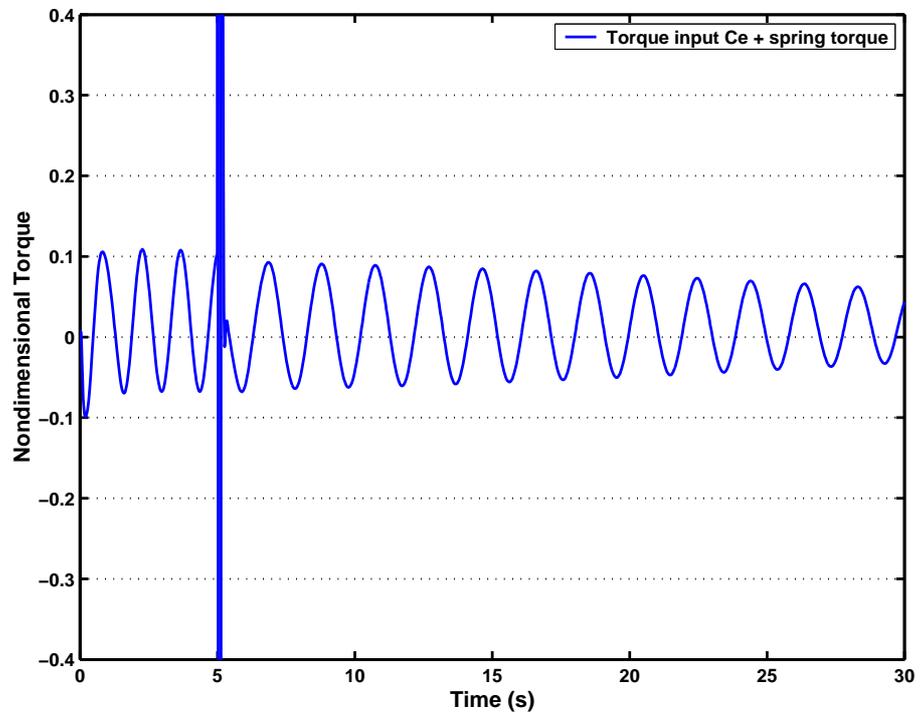


(b) Close up of the control signal

Figure 5.14: Control signal evolution in time for the optimal controller (the vertical line is actually a control input peak due to the step input). Cf C_e definition in eq. 5.19.

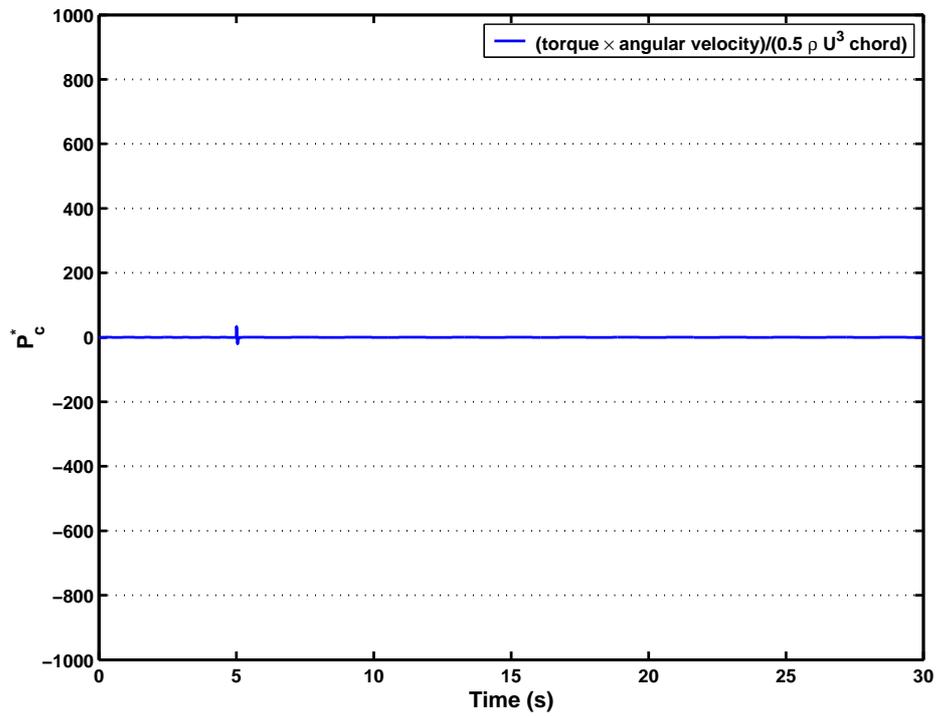


(a)



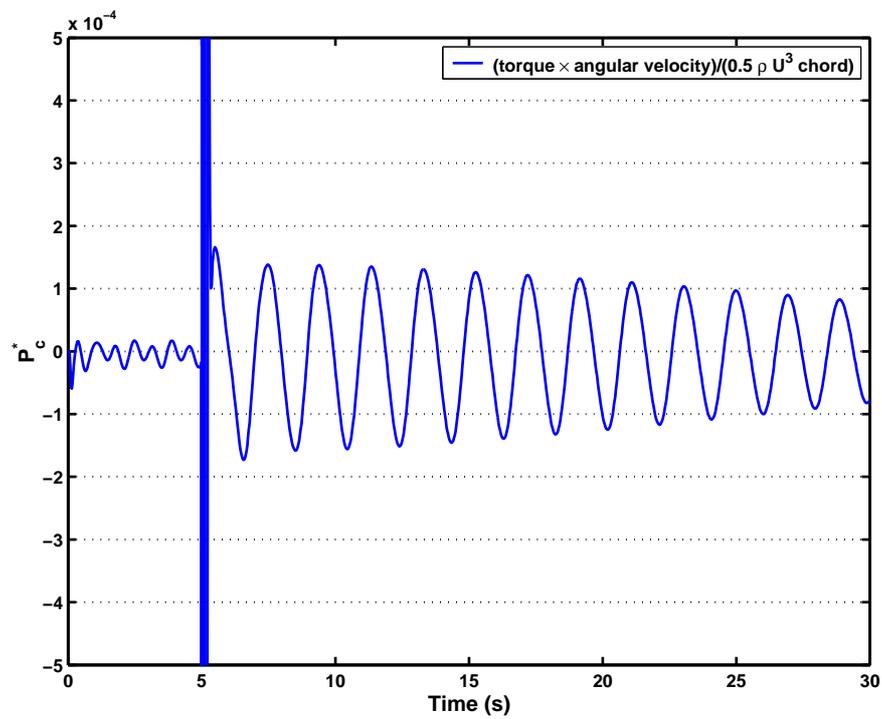
(b) Close up of the control signal minus the torque due to the spring

Figure 5.15: Control signal evolution in time for the optimal controller without taking into account the torque due to the spring. Cf C_e definition in eq. 5.19 for the non-dimensionalization.



11

(a)



(b) Close up of the power signal

Figure 5.16: Power signal evolution in time for the optimal controller. The power is divided by $(1/2)\rho U_\infty^3 L$

5.2.5 Fuzzy logic controller

For the general principle linked to the fuzzy logic controller implementation, readers are referred to the overview given in section 4.3.2. As in section 5.2.4, the fuzzy logic controller has been developed according to the plate and flow models used in sections 4.3.5, 4.3.6 and shown in section 4.4. The general structure used here is presented in the example section 4.3.2.1, as well as a typical set of rules [41], although some adjustments were made on the output value. The number of the fuzzy set was increased for better precision. In fact, it is similar to the fuzzy controller presented in section 4.3.2.1, and the main difference is that the Sugeno method is used here instead of the Mamdani method. The two methods use different defuzzification methods, and section 4.3.2.2 provides a general comparison. Later in the section, an example is provided of how the fuzzy controller implements the output to be compared with the Mamdani example in section 4.3.2.1.

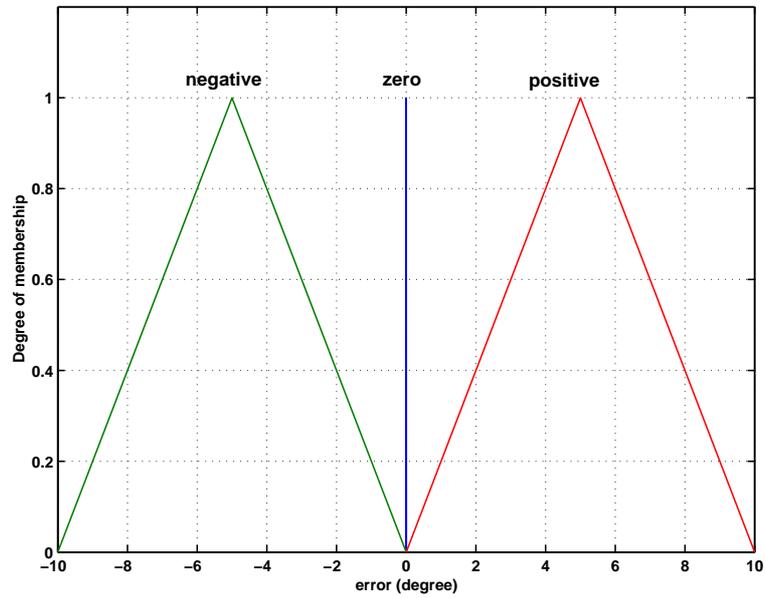
The fuzzy controller involves two inputs, the error signal e , and the error signal derivative \dot{e} , and one output C_a . The membership functions for e (in Matlab diagram, it is named “error”) are composed of three membership functions:

- *negative*: triangular function, parameters : [-10;-5;0] ([beginning;middle;end]).
- *zero*: triangular function, [-0.01;0;0.01].
- *positive*: triangular function, parameters : [0;5;10].

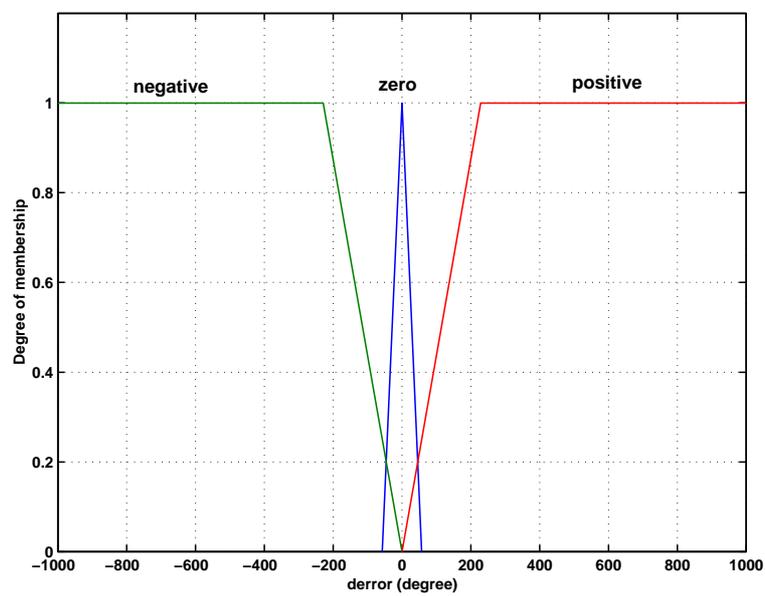
The membership functions for \dot{e} (in Matlab diagram, it is named “derror”) are composed of three membership functions:

- *negative*: trapezoidal function, parameters : [-17189;-11459;-229;0] ([beginning;beginning of the flat part;end of the flat part;end]).
- *zero*: triangular function, [-57;0;57].
- *positive*: trapezoidal function, parameters : [0;229;11459;17189].

Note that the membership function parameters of e and \dot{e} are respectively in units of degrees and degrees/s. Their representation is shown in figure 5.17.



(a) Membership function of e



(b) Membership function of \dot{e}

Figure 5.17: Membership functions of the input

The membership functions for C_a (the Matlab name is Ca) are composed of five constants:

- VN (Very Negative): -300.
- N (Negative): -150.
- A (Average): 0.
- P (Positive): 150.
- VP (Very Positive): 300.

The inference table is represented in figure 5.18.

error derror	negative	zero	positive
negative	VN	N	A
zero	N	A	P
positive	A	P	VP

Figure 5.18: Inference table for the fuzzy control

Note that every output in the inference table is the result of an “AND” rule that is to say there are nine rules. To illustrate the table one these rules can be specified as:

1. if (e is N) AND (\dot{e} is N) THEN (C_a is VN)

The inference is done using the *min* operator for the “AND” and the *max* operator for the “OR”. It is otherwise done in the same manner as in section 4.3.2.1. That is to say, using the simple example where $e = -2.5$ and $\dot{e} = -229$ the degree of membership is thus for e using figure 5.17(a):

- (e is N) : 0.5
- (e is Z) : 0

- (e is P) : 0

One can proceed in the same manner for \dot{e} using figure 5.17(b), the degree of membership to each set is :

- (\dot{e} is N) : 1
- (\dot{e} is Z) : 0
- (\dot{e} is P) : 0

Therefore, every rule produce a zero result using the *min* operator except the rule “if (e is N) AND (\dot{e} is N) THEN (C_a is VN)” which gives $\min(0.5, 1) = 0.5$. This is the degree of membership to the result “ C_a is VN”. Remind that the result VN is associated to the constant value -300 , thus the output of this rule is $0.5 * VN = -150$, and the output for every other rule is 0. The defuzzification is much simpler than for the Mamdani method, and consists simply in adding every rule output. With this example, it leads for the global output C_a of this fuzzy controller to the result : $C_a = -150 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = -150$.

This design is ideally suited for stabilization and can be adapted to a wide variety of plate and flow conditions. Nevertheless, these current membership function albeit leading to a stable has shown small remaining auto-oscillation as well as a very high computing time which is actually due to those remaining oscillations. This problem has prompted to use relaxed fuzzy sets which will use for the trial with the true flow simulation, these sets are:

- *negative* : triangular function, parameters : [-91;-45;0] ([beginning;middle;end]).
- *zero* : triangular function, parameters : [-1;0;1].
- *positive* : triangular function, parameters : [0;45;91].

As for \dot{e} , there are now three triangular membership functions:

- *negative* : triangular function, parameters : [-916;-458;0]
- *zero* : triangular function, parameters : [-114;0;114].
- *positive* : triangular function, parameters : [0;458;916]

This helps to reduce the computing time with a minimal loss regarding the performance.

Otherwise, The fuzzy control was used in the same conditions as in section 5.2.4 and tested it at $\alpha = 55$ degrees and with $f_n^* = S_s$. The control setup is given in figure 5.19. The results concerning θ are presented in figure 5.20 for a step input at $t = 5s$. A comparison between the error from the optimal controller versus the error from the fuzzy controller is shown in figure 5.21 while the error signal (difference between the prescribed angle output and the angle input) is presented in figure 5.22. The control signal u is shown in figure 5.23. Again, the controller torque minus the torque spring is plotted in 5.16. In order to compare with result of section, the definition and parameters from section 5.4 are used as well as the torque definition for C_e in equation 5.19.

Lastly, figure 5.25 shows the power divided by C_p . A short result summary was also provided in table 5.5 (to compare with table 5.3). Again, to provide a frame for comparison with results of section 5.3, the table results using t^* is also given in table 5.6. Note that in tables 5.5 and 5.6, the results used were those resulting from simulations with a relative calculus precision of 10^{-6} which are the results in figure 5.27. Further precision on this subject in later paragraph. Similarly to the optimal controller, trials with $\rho_b = 35 \text{ kg/m}^3$ have been conducted leading to results similar to the case $\rho_b = 1200 \text{ kg/m}^3$.

Further tests have shown that the system remains stable even for large simulation time. The performance is given by $t_r = 0.0653$, $D\% = 0.0114\%$. Regarding the $D\%$, one should temper this better value compared to both the classical controller and the optimal controller, using the root mean square of the error in the range $t = [5, 5.2]$ (tables 5.1, 5.3 and 5.5) one obtain (in term of % of 10°) :

- fuzzy controller : 37.8%
- fuzzy controller with higher calculus precision : 38.3%
- optimal controller : 35.7%
- classical controller : 1.5%

Overall, the motion precision is a little better with the optimal controller. A comparison between the two controller error is given in figure 5.21, it shows that the maximum difference is of -2.5° and that both control are equivalent past $t \simeq 5.6$. Note also that

if the optimal controller is slightly faster it tends to overshoot as well.

Like the optimal controller in section 5.2.4, there are remaining oscillations of amplitude of about $O(10^{-3})$ degrees, and a mean position error (which is also called permanent error) is of $O(10^{-4})$ degrees. Other plate parameters have led to similar results, so that for this problem the fuzzy control has proven to be the most versatile of the controls considered and the easiest to design. On the downside is the fact that with Matlab the computational cost is higher than for a linear system. Furthermore, there is no way other than trial and error to measure the robustness and stability of the control, if one does not want to use model based parameters. Nonetheless, some method do exists (see Jenkins and Pasino [29]) based for example on robust control design methodology.

One can see in the torque plots (figure 5.23 and in particular figure 5.24), that there are still some small oscillations remaining after the motion was set. This is interesting as it shows again that the effect of too tight input fuzzy sets. Indeed, this effect combined with a relaxed calculus precision (the parameter for relative precision was 10^{-3}) lead to a system which tends to create some very small auto-oscillations. Although not visible due to the scale, these auto-oscillation are also present for the angular velocity. This can be remedied under Simulink by choosing a tighter relative precision.

Figures 5.26 and 5.27 shows the result with a relative calculus of 10^{-6} . The order of magnitude for both variables has not been regarding the maximum limits. The performance is now given by $t_r = 0.072$, and $D\% = 0.011\%$. From the torque plot in figure 5.26(b) one can see there is no longer any oscillations in the Torque signal. Due to the tighter precision, the calculus is slightly longer for on timestep but the smoother evolution makes up for this effect. This shows the sensitivity of the fuzzy controller to numerical effect as no effect due to the calculus is visible with the optimal control.

Similarly to the optimal control case, note the peaks in the control input in figures 5.23, 5.24, 5.26(b), the error signal 5.22 and the power signal 5.25. Again, it is due to the step input, which means that at some point the angular derivative is infinite. Because of the fuzzy-logic control intrinsic characteristics, the system is “out of control” for a limited time, about 0.01 second. This time is dependent upon the simulation timestep used so that it is difficult to assess the period for which an actual system would go outside the boundary of the membership functions.

Turning now to the power analysis in figure 5.25 and 5.27, the instantaneous power is higher during the motion, afterwards it becomes much lower. Once the plate position is stabilized, it is in the worst case in figure 5.25 about half those of the optimal controller, and it is better with improved calculus precision 5.25. There however higher than for the classical controller. It is hard to explain why there is a discrepancy past $t = 5.2s$, and why one find 2.21% regarding the total energy. But as the total motion energy E_c ($E_{c,p}$ and E_c^*) is about ten times lower (in absolute value) than the total energy E_c of the optimal controller, it can be conjectured that it is simply a numerical effect.

Turning to tables 5.5 and 5.6, one can see now that the average power during each phase of the simulation (before, during and after the motion) is now positive although the precision during plate stabilization and during the motion remains comparable to the optimal controller (about ten times better for the motion). Before and after the plate motion, if the power P_a^* and energy E_c ($E_{c,p}$ and E_c^*) remain positive, they are however at very low values so that the distinction is difficult with the calculus precision. Looking now at E_s ($E_{s,p}$ and E_s^*), it shows that the power has higher peak than for the optimal controller, and that almost every peak is concentrated during the motion.

	time t (s)			
	$t = [0, 30]s$	$t = [0, 5]s$	$t = [5, 5.2]s$	$t = [5.2, 30]s$
<i>ErrorRMS</i> (in degree)	0.31335	0.0010297	3.8335	0.0055909
$E_{c,p}$	0.013561	-3.4598×10^{-6} -0.0255%	0.013265 97.8%	0.00029917 2.21%
P_a^*	0.00045203	-6.9195×10^{-7}	0.066196	1.2064×10^{-5}
$E_{s,p}$	151.12	9.1971×10^{-12} $6.09 \times 10^{-12}\%$	151.12 > 99%	4.6885×10^{-7} $3.1 \times 10^{-7}\%$

Table 5.5: Summary of the control results for the fuzzy logic controller. The power and energy are calculated from the power divided by C_p and t (see equation 5.6 and 5.9). The % are compared to the total energy. The *rms* is the common *RootMeanSquare* formula.

	Time t^* using $T^* = T \times a/U$ and section 5.4.2 parameters			
	$t^* = [0, 103.4]$	$t^* = [0, 17.2]$	$t^* = [17.2, 17.9]$	$t^* = [17.9, 103.4]$
E_c^*	0.046761	-1.193×10^{-5} -0.0255%	0.045742 97.8%	0.0010316 2.21%
P_a^*	0.00045203	-6.9195×10^{-7}	0.066196	1.2064×10^{-5}
E_s^*	521.12	3.1714×10^{-11} $6.09 \times 10^{-12}\%$	521.12 > 99%	1.6167×10^{-6} $3.1 \times 10^{-7}\%$

Table 5.6: Summary of the control results for the fuzzy logic controller. The power and energy are calculated from the power divided by C_p and t^* (see equation 5.6). The % are compared to the total energy. The *rms* is the common *RootMeanSquare* formula.

Despite the higher energy used during the motion by the fuzzy logic controller, the optimal controller has a higher t_r compared to the fuzzy controller, nonetheless the precision during the motion remains comparable between the two controllers even if the optimal controller is slightly faster. The higher energy E_c ($E_{c,p}$ and E_c^*) of the fuzzy logic controller can be justified by the better $D\%$, even though the response time t_r and error *RMS* are comparable between the optimal controller. Overall, it tends to indicate that the optimal controller follows more quickly the motion but with less precision. Past the motion, the fuzzy is then comparable as it ensures a similar precision –even a bit better with half the optimal controller value– with a similar power.

This analysis shows that the fuzzy logic controller is more power consuming than the optimal controller. It must be pondered though by several facts, first of all the range $t = [5, 5.2]s$ has been chosen regarding the optimal controller fluctuation past the motion. A more appropriate range for the fuzzy logic controller would have been to decompose this set into several sets. Secondly, this case is not favorable to the fuzzy controller as it is relatively “out of control” at the start of motion (as seen in a previous paragraph). The resulting peak visible in the torque plot (figure 5.23) is thus a by product of a failure mode than truly a control torque. It would thus be interesting to push the parameter regarding the *derror* fuzzy sets *negative* and *positive*. The simulation cases in section 5.4.2 would also be a better reflection of physical system. Finally, the fuzzy logic controller is more versatile as it does not need to be changed for a range of f_n^* whereas the optimal controller has been designed with particular spring/damped

plate parameters.

Thus for the fuzzy controller, there seems to be a tradeoff in power compared to the optimal controller where the t_r is degraded as well as the motion precision, albeit to a lesser extent, in favor of the precision and efficiency of the stabilized state of the plate. Maybe a better optimization of the fuzzy sets would help overcome this problem and further experiments would be required as well as a wider exploration of the parameter space.

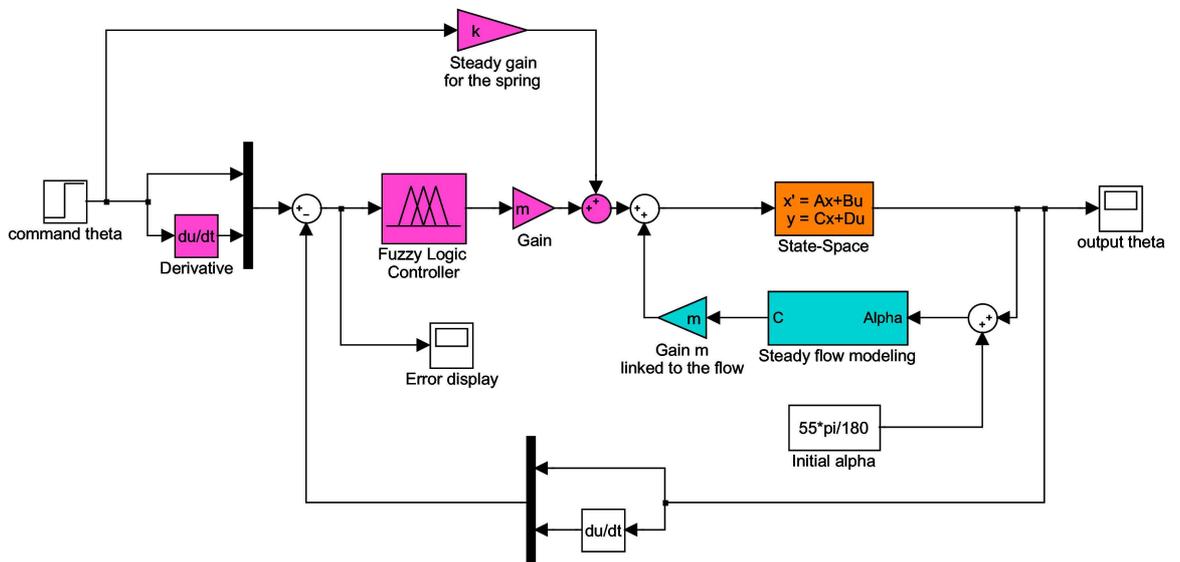


Figure 5.19: Control and system closed loop block diagram for the fuzzy logic controller.

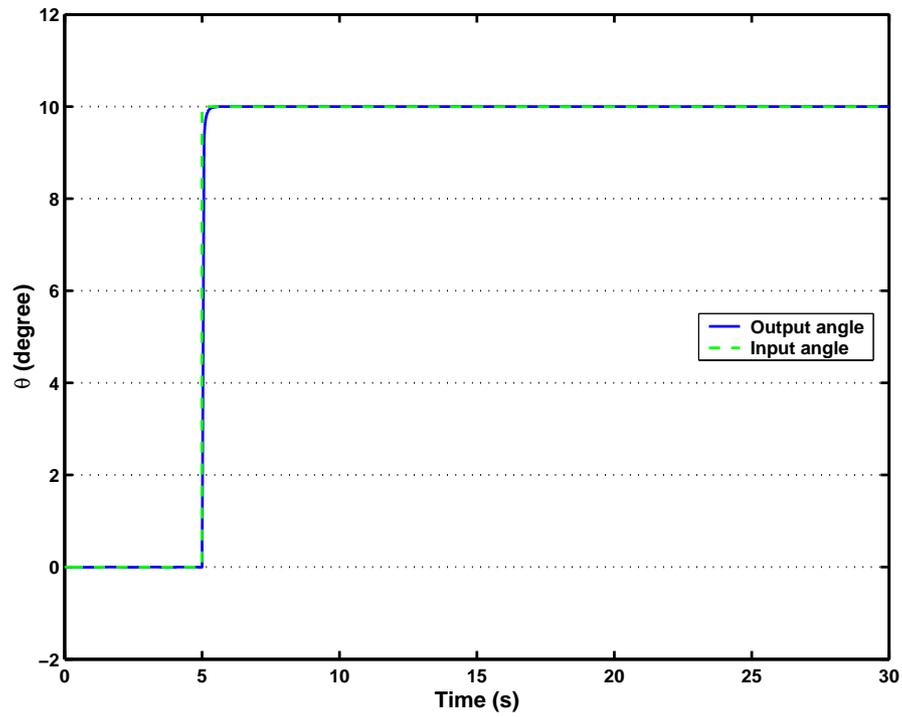


Figure 5.20: Output of the closed loop system for a step input for the fuzzy logic controller.

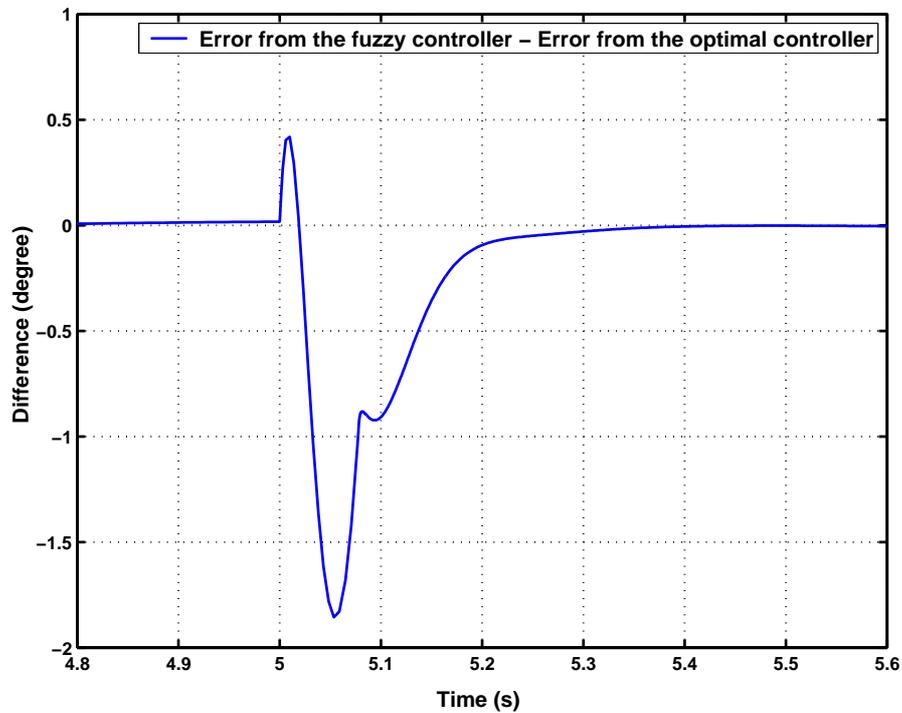
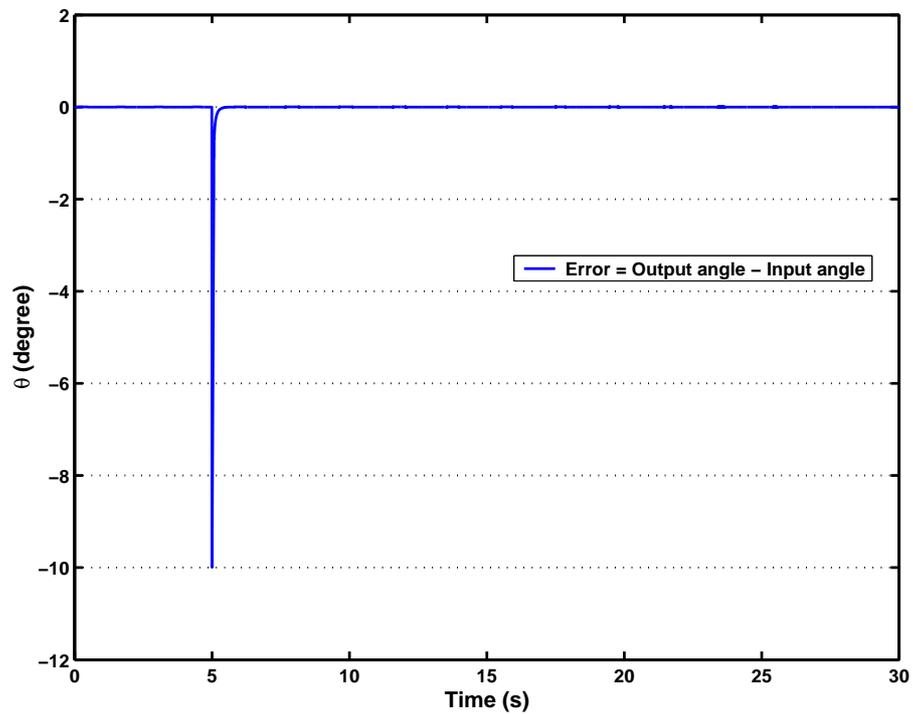
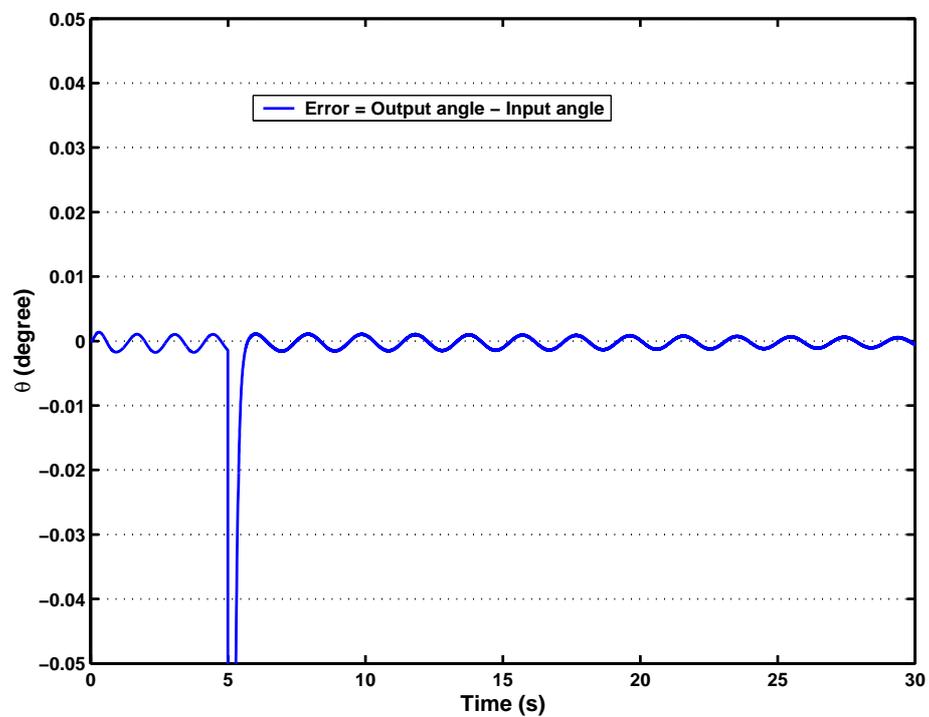


Figure 5.21: Error from the optimal controller minus the error from the fuzzy controller with relative calculus precision of 10^{-6} .

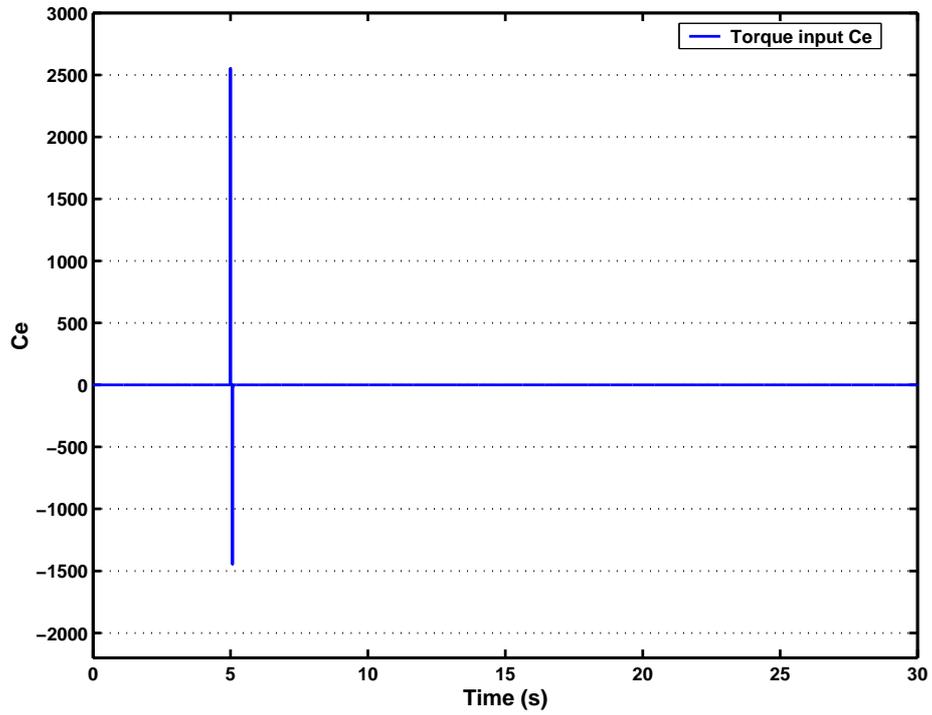


(a)

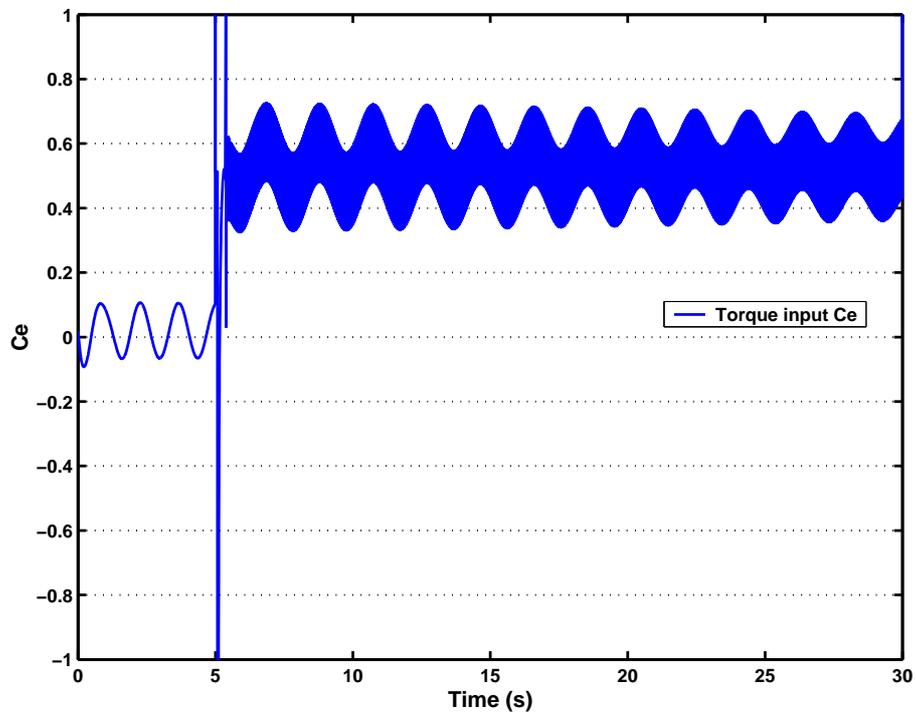


(b) Close-up view of the error

Figure 5.22: Error of the closed loop system for a step input for the fuzzy logic controller.

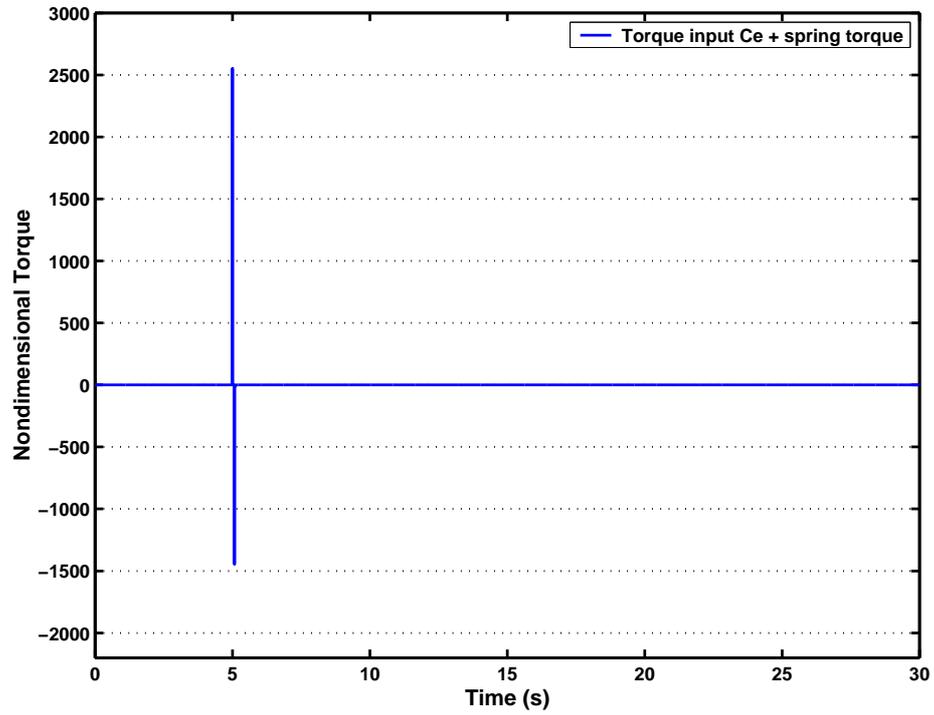


(a)

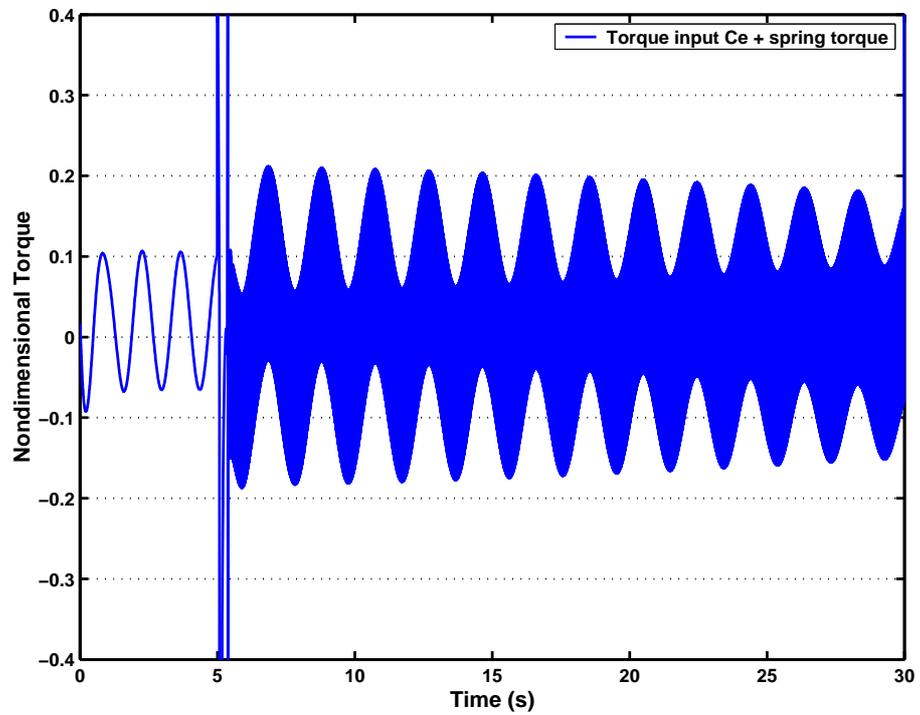


(b) Close up of the control signal

Figure 5.23: Control signal evolution in time for the fuzzy logic controller (the vertical line is actually a control input peak due to the step input). Cf C_e definition in eq. 5.19.

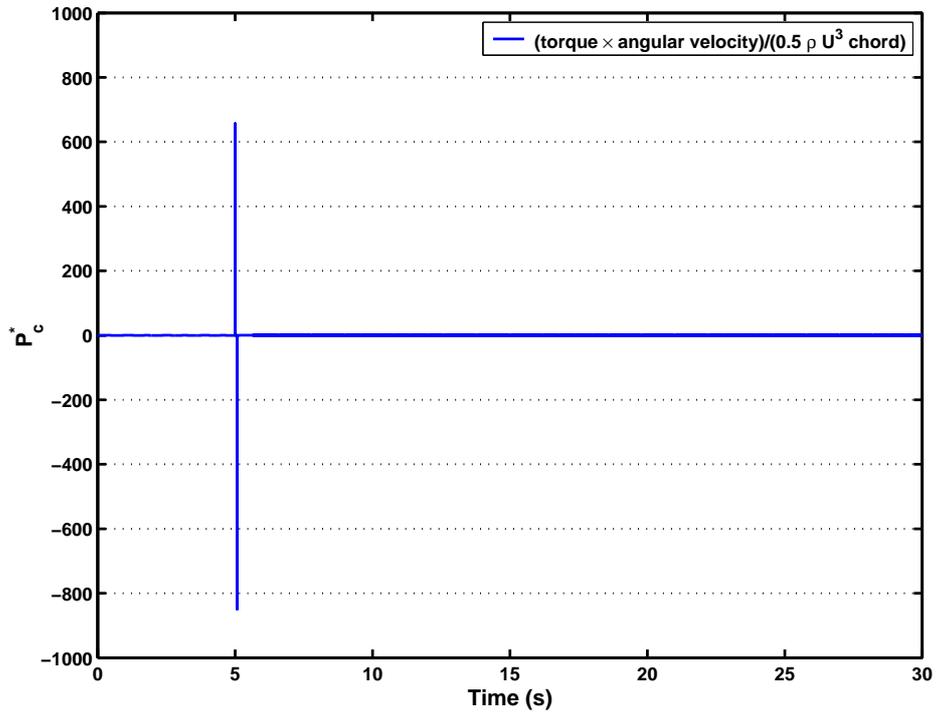


(a)

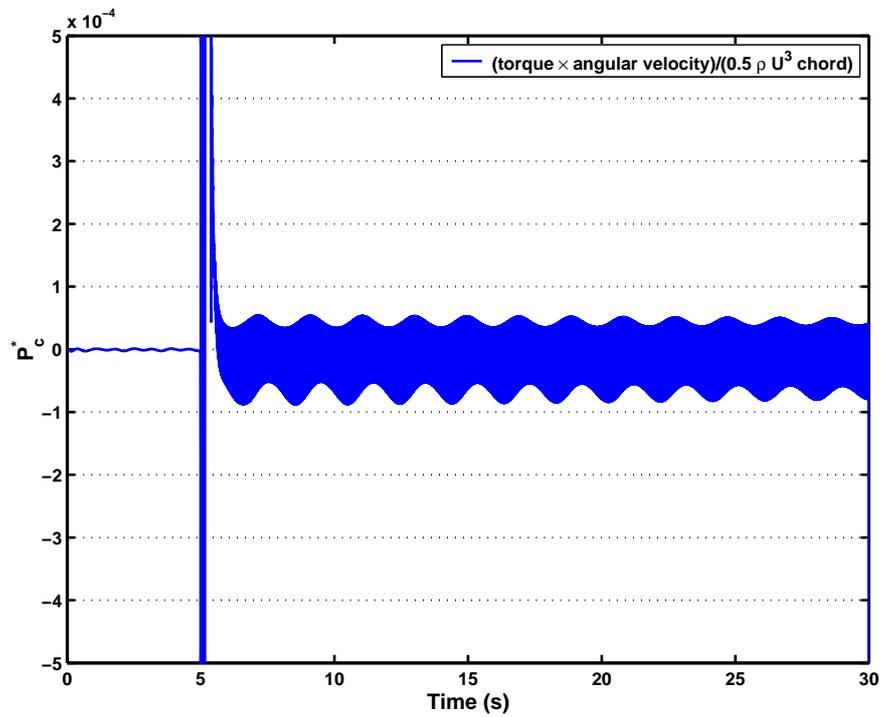


(b) Close up of the control signal minus the torque due to the spring

Figure 5.24: Control signal evolution in time for the fuzzy logic controller without taking into account the torque due to the spring. Cf C_e definition in eq. 5.19 for the non-dimensionalization.

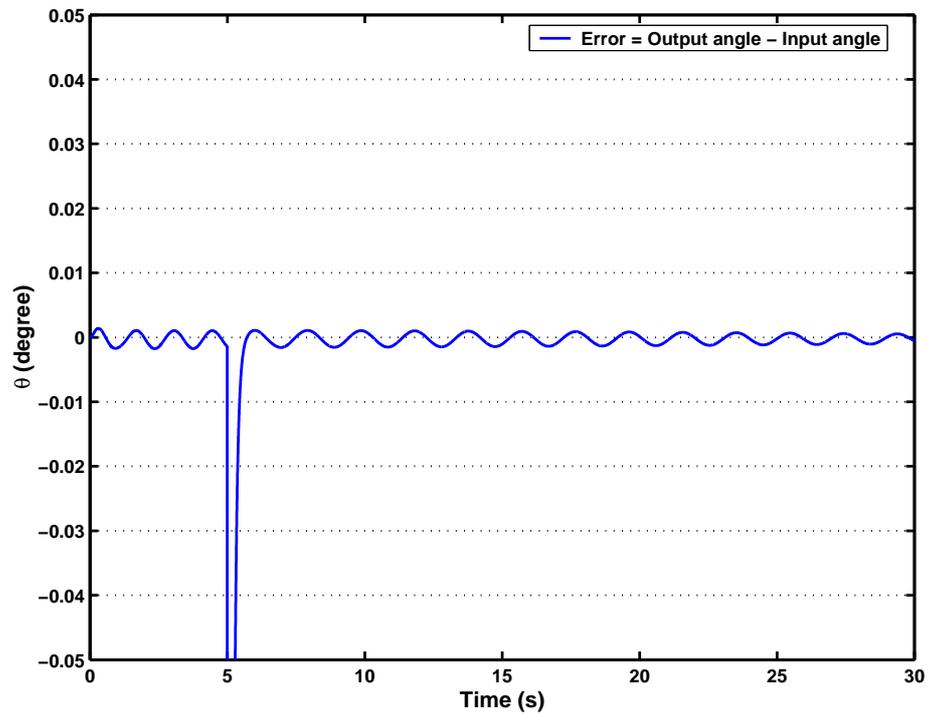


(a)

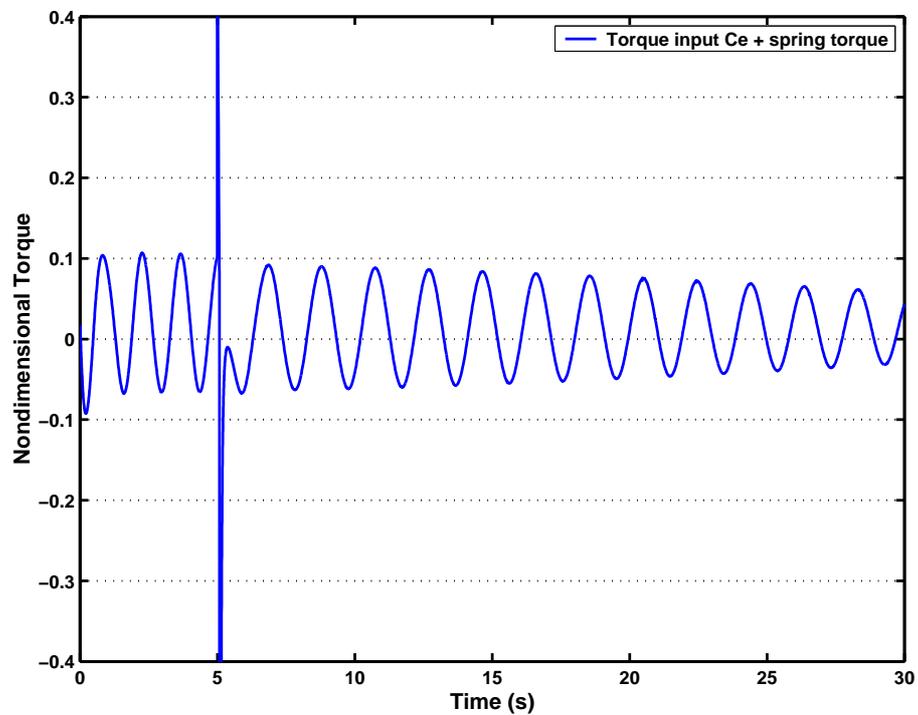


(b) Close up of the power signal

Figure 5.25: Power signal evolution in time for the fuzzy controller. The power is divided by $(1/2)\rho U_\infty^3 L$



(a) Close-up view of the error



(b) Close up of the control signal minus the torque due to the spring

Figure 5.26: Control signal and angular error evolution in time for the fuzzy logic controller with relative calculus precision of 10^{-6} . Cf C_e definition in eq. 5.19 for the non-dimensionalization.

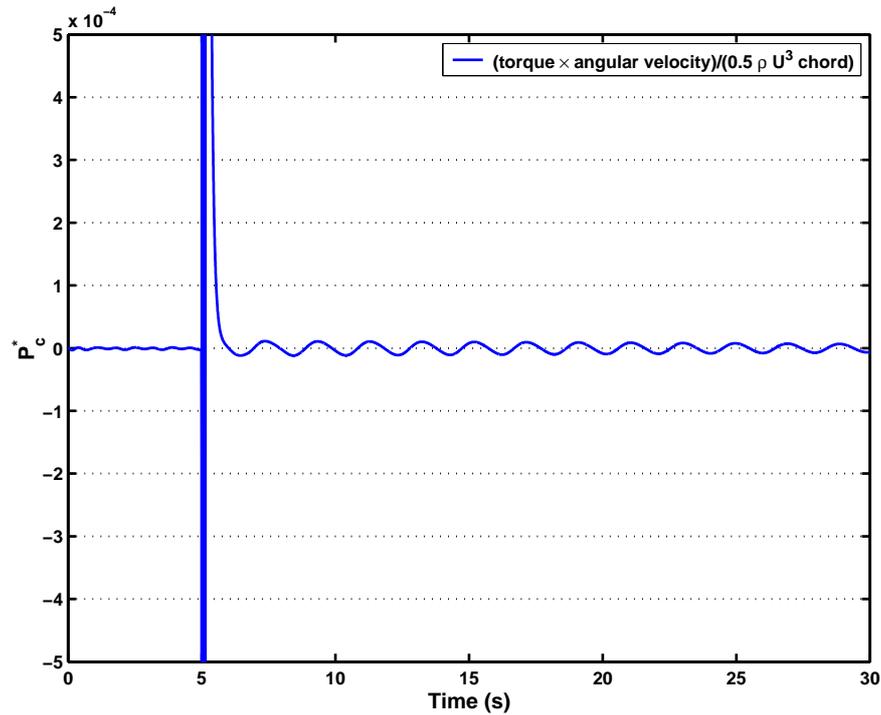


Figure 5.27: Power signal evolution in time for the fuzzy controller with relative calculus precision of 10^{-6} . The power is divided by $(1/2)\rho U_\infty^3 L$.

5.3 Integrated Controller/Flow simulation

5.3.1 Using the Matlab interface

The Matlab/C interface for the vortex simulation has been successfully implemented. It has been implemented in two ways, one function being fully compiled under Matlab, while the other goes through an intermediary Matlab function (it is easier to change the parameter entries for the latter).

Due to the fact that the flow simulation only gives data at given time steps independent of the Matlab simulation, it has been decided to interpret the flow simulation data as discrete, although the time step of the flow simulation is usually much smaller than that of Matlab, which leads to an almost continuous behavior of the system. However, the intrinsic implementation of system simulations implies that it is difficult to couple fully the two simulations therefore this compromise was chosen.

One should note that the slowdown due to the interface is not noticeable during the global simulation. Nevertheless, due to the time taken by the flow simulation, it was

preferred to use the approximations made under Matlab (linear flow model at given incidence angle where their timescale is changed depending on the incidence angle, with fuzzy logic boxes for the transition between the models) as in sections 4.3.5 to design the controller; the interfaced Matlab/C flow simulation is then used to validate the controller.

5.3.2 Direct implementation in the Spalart simulation

Considering the structure of the fuzzy logic controller and optimal controller (with the state-space form), it is possible to implement them directly into the Spalart Vortex method to integrate the plate position from one time-step to another. Nevertheless, due to a lack of time it has not been possible.

5.4 Flow/plate and controller system

The flow simulation is done using the translated Spalart code coupled with the Simulink controller. Unless otherwise stated, the Simulink torque input will be started at the flow simulation beginning. The plate is freed to rotate at $t^* = 12.8$. Finally, note that θ is again the angle from the rest position of the spring, and α is the incidence angle, such that $\alpha = \alpha_0 + \theta$, with α_0 the initial angle at $t^* = 0$. Here, α_0 was taken as $\alpha_0 = 55^\circ$. In order to use the results from section 3.3.2, the plate density was kept at $\rho_b = 1200 \text{ kg/m}^3$.

For the Simulink part of the simulation, a discrete simulation was used for the fuzzy logic controller (Simulink parameters: constant integration time-step, Discrete integration/Flow simulation parameters: constant time-step, Adams-Bashforth of order 2). On the other hand, because of Simulink limitation concerning the optimal controller implementation, a continuous integration was used for the optimal controller (Simulink parameters: variable integration time-step, Dormand-Price algorithm vs Flow simulation parameters: constant time-step, Adams-Bashforth of order 2). In the latter case, this continuous implementation, as opposed to the discrete output provided by the flow simulation, creates some additional problems by creating artificially some delays between the torque input and the torque taken into account by the flow simulation, thus generating nonlinearities.

The effect is difficult to reproduce exactly using Simulink alone, most notably because they are not actual delays but rather a dephasing between the input C_a applied to the flow simulation and the flow simulation output in the feed back loop (controller inputs variable $Alpha$ and $theta_y$ in figure 5.11). Indeed, as previously stated the continuous implementation under Simulink means that the Simulink models use a variable timestep while the flow simulation is using a constant timestep. Therefore certain Simulink timestep tends to be slightly different from the simulation timestep, meaning that the flow simulation output is slightly dephased from the Simulink input as Simulink is computing the C_a before the flow simulation has provided the $Alpha$ and $theta_y$ corresponding to the previous input. The reciprocal effect is also true. At this point, this means that the simulations dephasing can be simplified as a delay in the feedback loop. Nevertheless, such delay would actually be varying with the simulation, it is thus very difficult to assess.

Trials were made to estimate an “average delay” which would cause a system instability with an experiment using the steady flow model from section 4.3.5, a and a pure delay in feedback loop (such as the feedback loop presented in figure 5.11). The setup is presented in figure 5.28. It is thus possible to show at least to show that this delay is slightly inferior to the timestep used for the flow simulation considering the results in figures 5.33 and 5.34. Due to a lack of time, it was not possible to deepen the study on this effect, all the more since it was found from trials using different parameters with the coupled flow simulation/Simulink model and the setup in figure 5.28 that this problem could easily be overcome by simply decreasing the flow simulation timestep from $\Delta t^* = 0.05$ to $\Delta t^* = 0.04$. With this Δt^* modification, the continuous optimal controller was able to control successfully the plate as if the flow simulation was continuous. The fuzzy-logic controller was less affected with such problems, even when using the same continuous Simulink implementation as the optimal controller, showing the controller robustness.

Note that in the following sections, the input torque provided by Simulink will be denoted M_e , and the nondimensional input torque C_e is

$$C_e = \frac{M_e}{\frac{1}{2} \rho L^2 |\vec{U}_\infty|^2}. \quad (5.18)$$

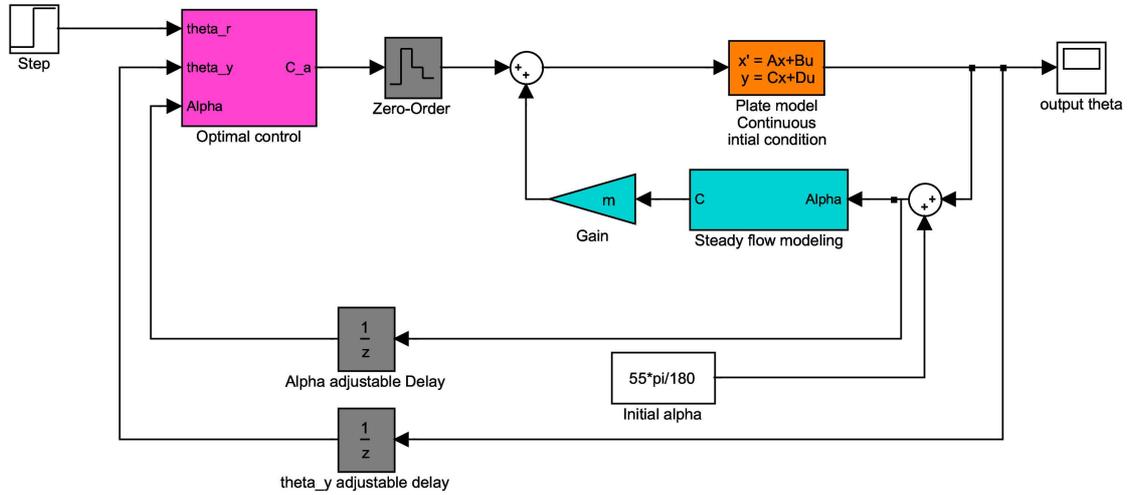


Figure 5.28: Control and system closed loop block diagram for the optimal controller with an average delay at the flow simulation output.

5.4.1 Plate stabilization

In this section, the plate must be stabilized at $\alpha = 55^\circ$ by using the fuzzy controller and optimal controller designed above. Slight modifications were applied to the fuzzy logic controller to take into account a larger range of incidence angles α . Keeping the nomenclature used in section 5.2.5, the fuzzy sets for the input (named e and \dot{e}) are also less tight in order to speed up the computation.

The new membership functions for e are:

- *negative* : triangular function, parameters : $[-91;-45;0]$ ([beginning;middle;end]).
- *zero* : triangular function, parameters : $[-1;0;1]$.
- *positive* : triangular function, parameters : $[0;45;91]$.

As for \dot{e} , the membership functions have also been modified to compensate for the changes of the e membership functions. The former \dot{e} membership functions slowed down enormously the controller output computation and introduced some oscillations in the Torque output (section 5.2.5). There are now three membership functions:

- *negative* : triangular function, parameters : [-916;-458;0]
- *zero* : triangular function, parameters : [-114;0;114].
- *positive* : triangular function, parameters : [0;458;916]

As in section 5.2.5, the membership function parameters of e and \dot{e} are respectively in unit of degree and degree/s. The output fuzzy sets for C_a are the same as those defined in section 5.2.5.

The flow simulation will be carried out using the same parameters as in the sections 3.3.2.1 and 3.3.2.2 except for Δt^* which was chosen as equal to $\Delta t^* = 0.04$ for the optimal controller and $\Delta t^* = 0.05$ for the fuzzy logic controller. More specifically, the reduced damping was chosen as $\xi^* = 0.02$, and the plate motion is started at the 256th iteration. Nevertheless, in both cases, controllers are activated from the simulation beginning. Note, that as the controllers are dependent on plate motion variables θ and $\dot{\theta}$, and as up to the 256th iteration $\theta = 0$, $\dot{\theta} = 0$ and the motion is commanded at zero the output C_a is equal to zero up to the 256th iteration so that it does not interfere with the plate motion. Two different reduced frequencies were chosen: $f_n^* = S_s$ and $1.5 * S_s$ corresponding to $0.77 * S$ and $1.16 * S$, with S_s and S respectively the Strouhal number resulting from the numerical simulation and the experimental Strouhal number for a fixed ellipse at $\alpha = 55^\circ$. Thus, there is one case where the reduced frequency f_n^* is close to the fixed-plate simulation Strouhal number, and another case where f_n^* is close to the experimental Strouhal number.

From section 3.3.2.2, it has been shown that the likely lock-in f_n^* would be around S if the ellipse was free to oscillate without a controller. Therefore, this choice allows to examine potentially troublesome cases where if the plate is not stabilized by the controller, the vortex shedding dynamics is slightly changed (as in the case $1/f_n^* = 11$ in section 3.3.2.2) and thus so are the coupled spring damped ellipse/flow characteristics. It will then enable to assess the controller robustness.

As noted earlier, the fuzzy-logic controller is able to handle successfully the nonlinearities caused by the discrete nature of the signal provided by the flow simulation³,

³One of the problem when dealing with the coupling between a numerical simulation inserted into a Simulink flow model is how to synchronize two parallel numerical resolution. The simplest solution is to define the flow simulation output signal (θ and $\dot{\theta}$) as discrete to Matlab. See introduction of section 5.4

$\Delta t^* = 0.05$ was then used to test the fuzzy logic controller. Results are shown in figures 5.29 and 5.30 respectively for $f_n^* = S_s$ and $f_n^* = 1.5 S_s$, which can be compared with the optimal controller using $\Delta t^* = 0.04$ in figure 5.31 and 5.32 respectively for $f_n^* = S_s$ and $f_n^* = 1.5 S_s$. For the sake of comparison, results are also provided for the optimal controller using $\Delta t^* = 0.05$ in figures 5.33, 5.34 and 5.35. Power plots were not used here because it has been shown that it is not truly representative of the controller behavior. Indeed, it is lacking the dynamic part which is linked to a plate motion.

Note that “input torque” in the torque plot denotes the C_e fed to the flow simulation provided by the control (Torque output is essentially the same and simply designates the torque output signal from the controller). C_e is then defined by:

$$C_e = \frac{C_a}{\frac{1}{2} \rho L^2 U_\infty^2}, \quad (5.19)$$

with C_a defined in equation 5.1 is the input in $N.m$ of the controller.

The fuzzy-logic controller shows less good performance than the optimal controller for the stabilized position, although the system remains stable in every case. Indeed, from figures 5.29(a) and 5.30(a), one can see that θ is stabilized at 0.04 ± 0.16 degrees independently of f_n^* compared to 0 ± 0.075 degrees for the optimal controller in figures 5.31(a) and 5.32(a). The fuzzy-logic controller performance is apparently due to the fuzzy set definition. This is because by choosing a larger range for the definition of the *zero* fuzzy set for the input e of the fuzzy logic controller, the angle precision constraint was relaxed. Therefore, the performance can be improved by choosing a tighter *zero* fuzzy sets of e , but it requires adjusting in return \dot{e} and these tighter sets require much more computing power. It is possible to find optimal sets for the fuzzy-logic controller, however they can only be found empirically due to the lack of intrinsic convergence proof in the method, otherwise some methodology exists but they explicitly use a plate/flow model which was not the original purpose in this thesis (see section 4.3.2).

These trials show nevertheless the robustness of the fuzzy logic controller, as it remains unchanged throughout the different f_n^* . Conversely, the optimal controller must be designed for a specific f_n^* .

As for the torque plots related to the fuzzy logic controller in figures 5.29(b) and 5.30(b), when comparing it to the optimal controller plot in figures 5.31(b) and 5.32(b), once the

spring-damped ellipse/flow and control system is stabilized to a periodic regime, one can see that both have similar values, with $C_e \simeq 0.0276 \pm 0.225$ ($mean(C_e) \pm std(C_e)$, $std()$ being the standard deviation also noted *rms*) for the fuzzy-logic controller as compared to $C_e \simeq 0.029 \pm 0.26$ for the optimal controller. It is logical to have less torque to stabilize the ellipse in the fuzzy-logic case, since there is less prescribed precision. Note the initial constant additional torque in the fuzzy-logic torque plots (5.29(b) and 5.30(b)), which is due to the default torque output of the fuzzy logic controller. However, it has no effect on the ellipse motion as the simulation does not use the torque input until the 256th iteration, after which the input becomes normal once the ellipse motion is allowed.

In the example provided for $\Delta t^* = 0.05$, the optimal-based controller is unable to stabilize the ellipse in figures 5.33(a) and 5.34(a). The torque input provides a smooth input for the coupled ellipse/flow system, but nevertheless its maximum value is about two thirds of the maximum value provided by the fuzzy logic controller for the same conditions. Furthermore, it seems that there is an added delay in the response due to the timestep. Comparing figure 5.33(b) and 5.29(b), the optimal controller is seen to produce an initial torque output opposite of what is required according to the fuzzy controller output. It then induces a high initial angular velocity response, further amplified by the initial torque output from the optimal controller. Due to the delay, the optimal controller becomes unable to control the motion. Additionally, the high-amplitude oscillations also imply a change in the flow dynamics, and more especially the flow frequency as described in section 3.3.2.2. In return, this means that the system characterization of the spring-damped ellipse/flow system is no longer valid.

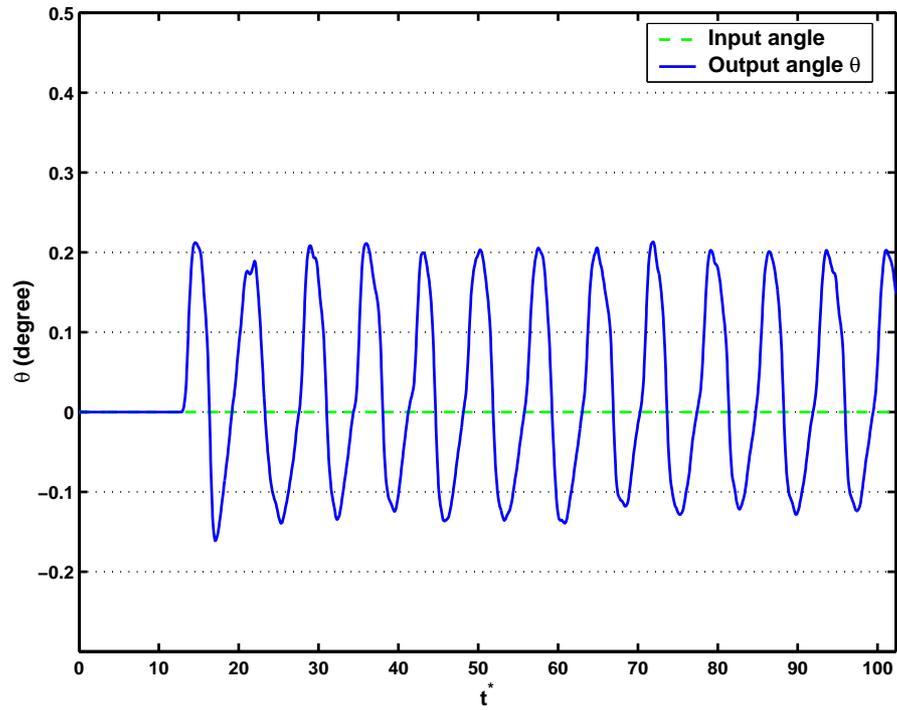
It is possible to improve the performance of the optimal controller with $\Delta t^* = 0.05$ by using a Kalman estimator, using as in section 5.2.4 a state penalty matrix such that $Q \simeq 10000 R$, whereas here $Q \simeq 100 R$ in the case of the combined optimal controller. A high state covariance noise was also used, which in this case is about 10 times that of the output covariance noise. This means that the actual process is imprecise and not well characterized, and these modified parameters thus improve the robustness of the controller. The results were much better, as can be seen in figure 5.35 (where $f_n^* = 1.5 S_s$), albeit still not ideal with $\theta = 0.12 \pm 7.4$ degrees, compared to $\theta = 1.47 \pm 23.35$ degrees for the previous optimal controller design. Better results could be presumably be obtained by raising the state covariance noise when designing the Kalman estimator.

To further illustrate the stabilization ability of the fuzzy logic controller, after allowing plate oscillation in the flow with $f_n^* = S_s$ activation of the controller was delayed until the 1023th iterations ($\Delta t^* = 0.05$). As can be seen from figure 5.36, the controller has successfully stabilized the ellipse with the same performance as in the case of the fuzzy logic controller active from the simulation beginning (figure 5.30). The torque plot (figure 5.37) confirms the similarities once the ellipse is stabilized around a rest position (figure 5.30(b)).

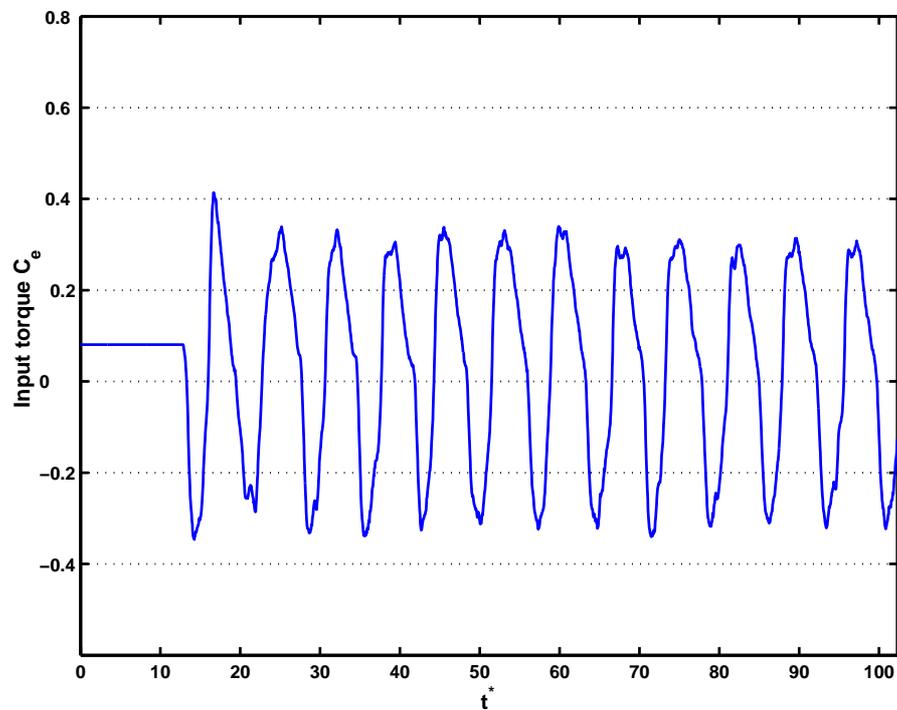
The transition is shown in figure 5.38, which lasts 0.85 nondimensional time units. Again the time limitation in the recovery from such a position is due to the fuzzy logic input \dot{e} fuzzy sets definition, which limits the maximum reachable velocity. At $t^* = 51.15$, θ is at a maximum; the controller is enabled at this time it then commands a first torque input in order to initiate a motion back to $\theta = 0^\circ$ (the first positive spike in figure 5.38(b)). As the ellipse reaches $\theta = 0^\circ$, the controller counters the inertia in order to stay around this position (negative spike and second positive spike in figure 5.38(b)). As stated in the previous paragraph, once the ellipse is stabilized, the torque output reaches an evolution similar to the torque dictated by the fuzzy-logic controller that is enabled from the beginning of the motion (figure 5.30).

Recalling that $\alpha = \alpha_0 + \theta$ with $\alpha_0 = 55$, this example is not feasible for the current implementation of the optimal controller, as the initial θ oscillations are too big and thus out of range for the α interpolation (section 5.2.4).

From these results, one may conclude that the fuzzy-logic controller is very flexible and has proven more versatile for the plate stabilization despite the nonlinearities. Nevertheless, its performance is dependent on the fuzzy sets definitions. As the sets are not independent one from another, optimal fuzzy sets can only be found empirically using the fuzzy logic controller and the physical system (section 4.3.2). Otherwise, various methods for adjusting and optimizing the fuzzy logic parameters exists based for example on method used usually on method used for robust control. Jenkins and Pasino [29] presents an introduction on such method, and Zhiqiang Gao et al [66] provide an example of more sophisticated method. However, such methods using a model of the system to control remove some of the advantages of the fuzzy logic controllers. Conversely, the optimal controller is more difficult of use but for conditions for which it is valid it performs better than the non optimized fuzzy-logic controller.

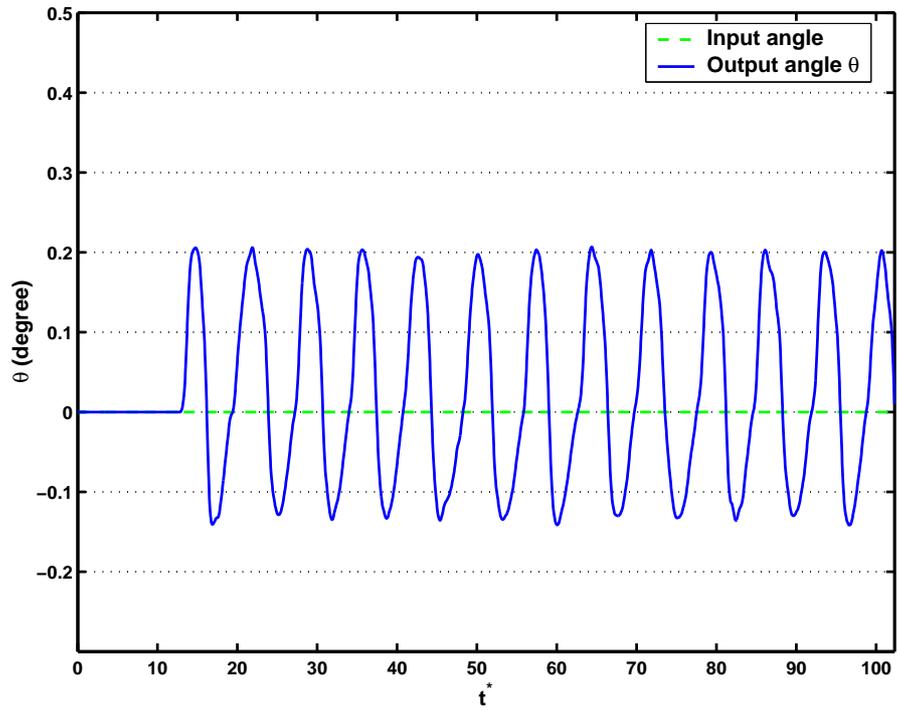


(a) $\theta(t)$

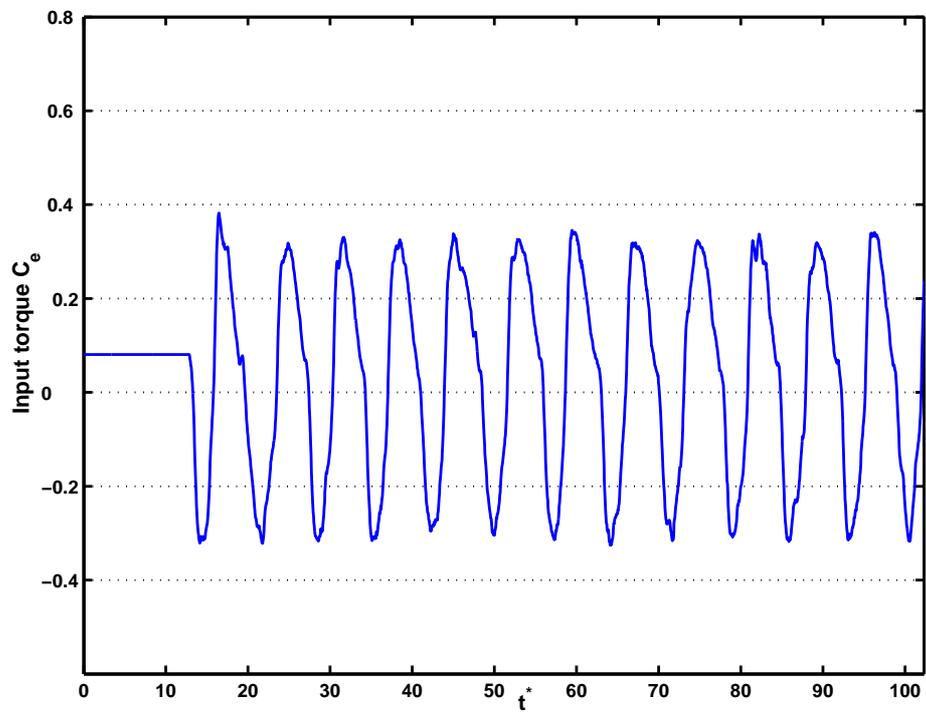


(b) Controller Torque

Figure 5.29: Time plot of the angle evolution and controller torque input for the flow/plate system controlled by the fuzzy logic controller with $f_n^* = S_s$ and $\Delta t^* = 0.05$

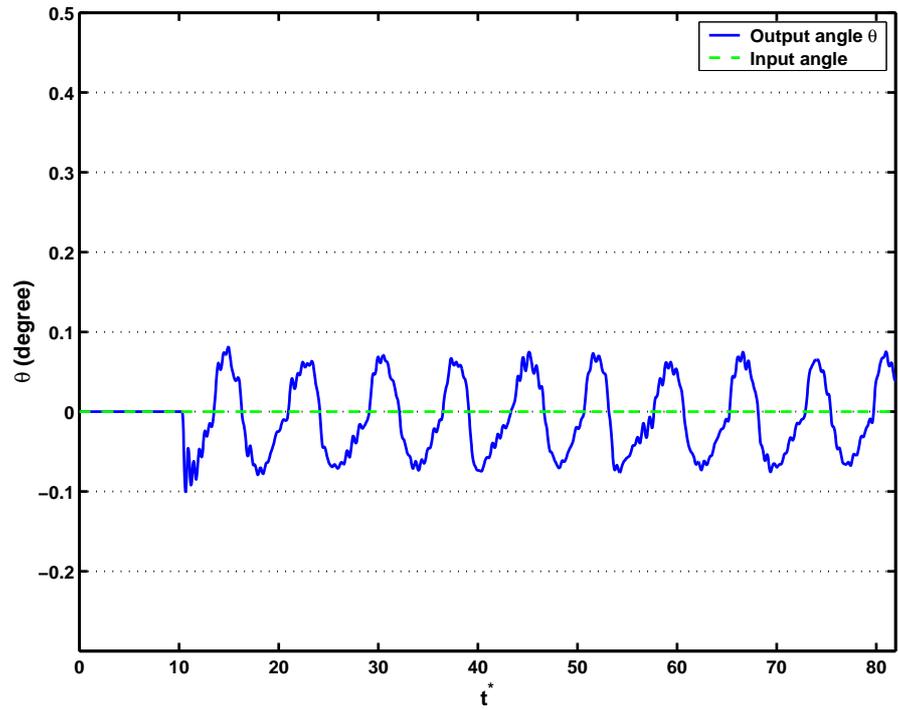


(a) $\theta(t)$

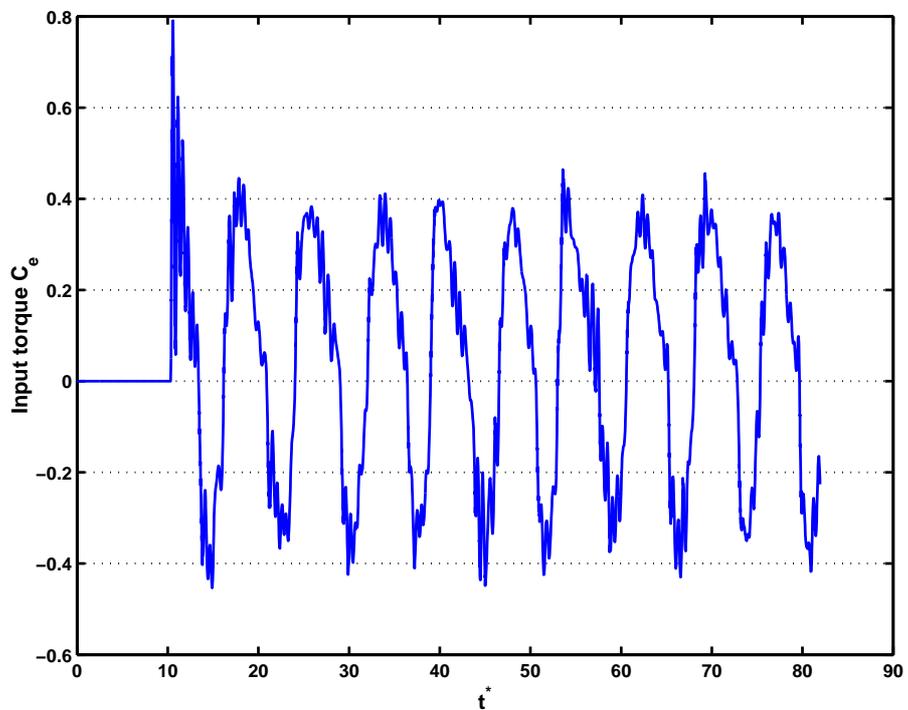


(b) Controller Torque

Figure 5.30: Time plot of the angle evolution and torque input for the flow/plate system controlled by the fuzzy logic controller with $f_n^* = 1.5 S_s$ and $\Delta t^* = 0.05$

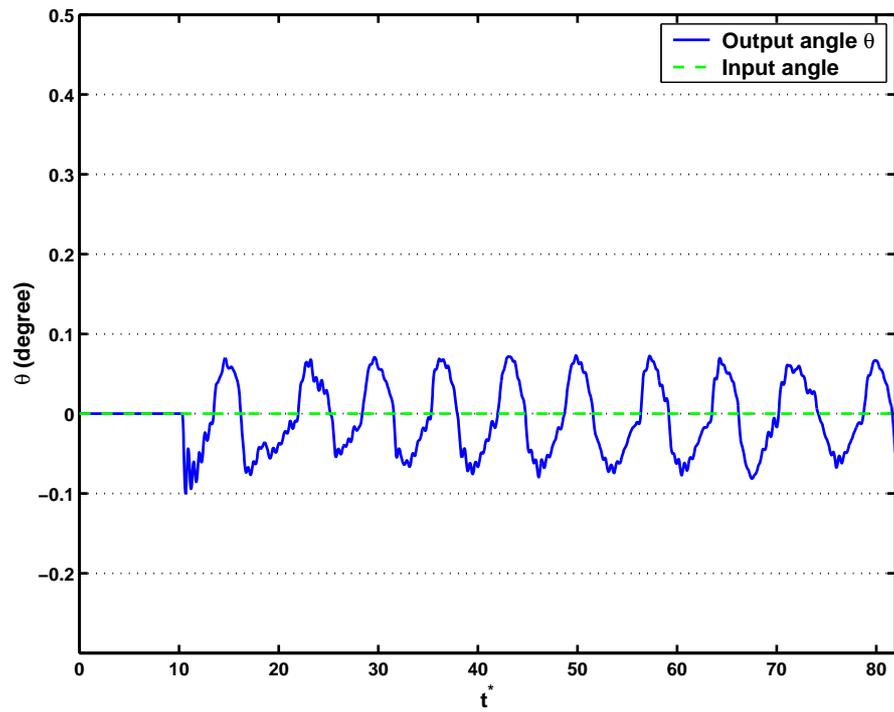


(a) $\theta(t)$

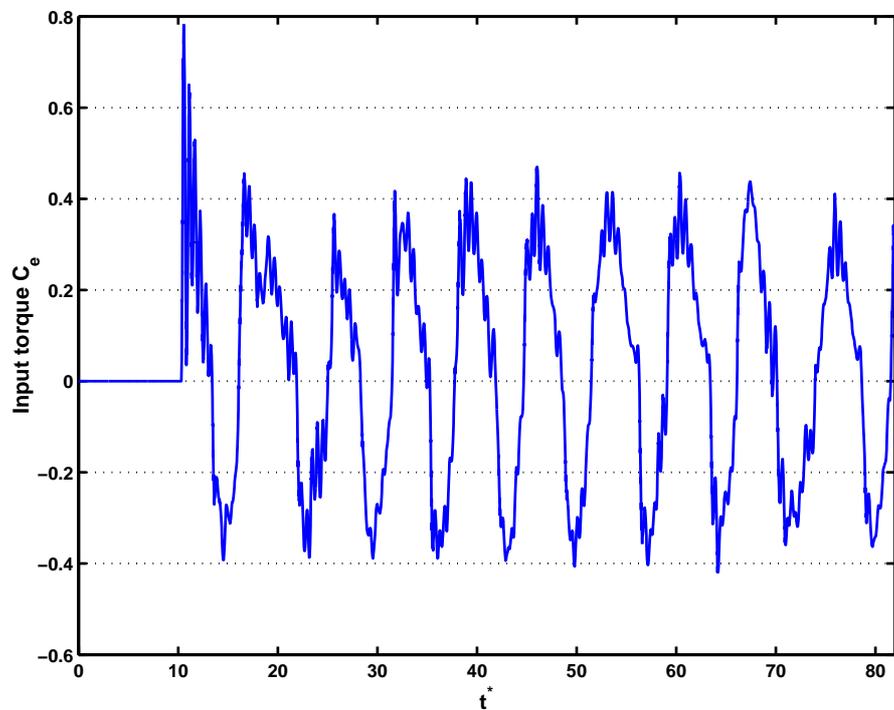


(b) Controller Torque

Figure 5.31: Time plot of the angle evolution and torque input for the flow/plate system controlled by the optimal controller with $f_n^* = S_s$ and $\Delta t^* = 0.04$

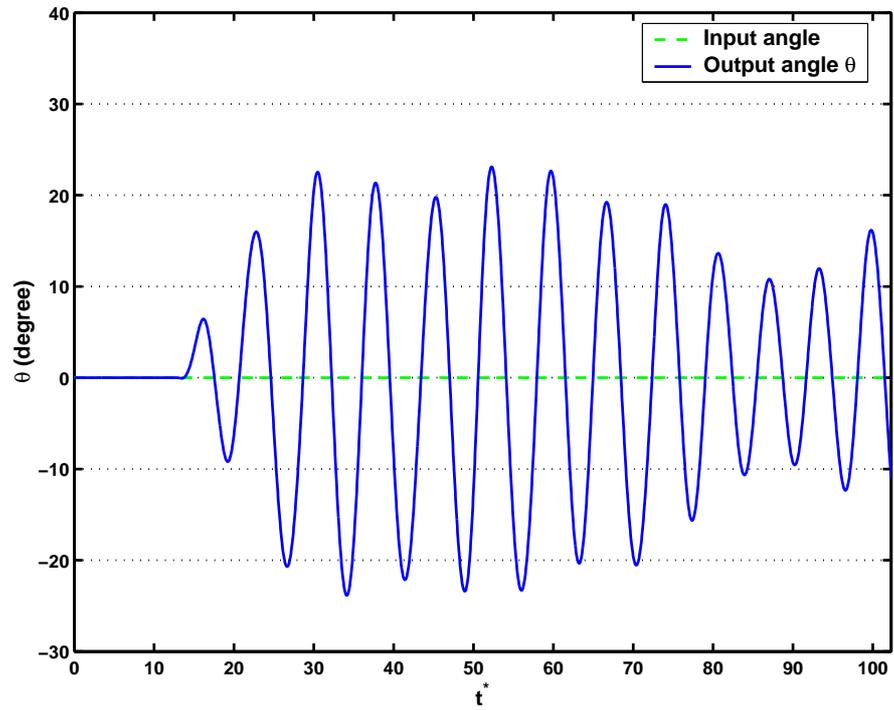


(a) $\theta(t)$

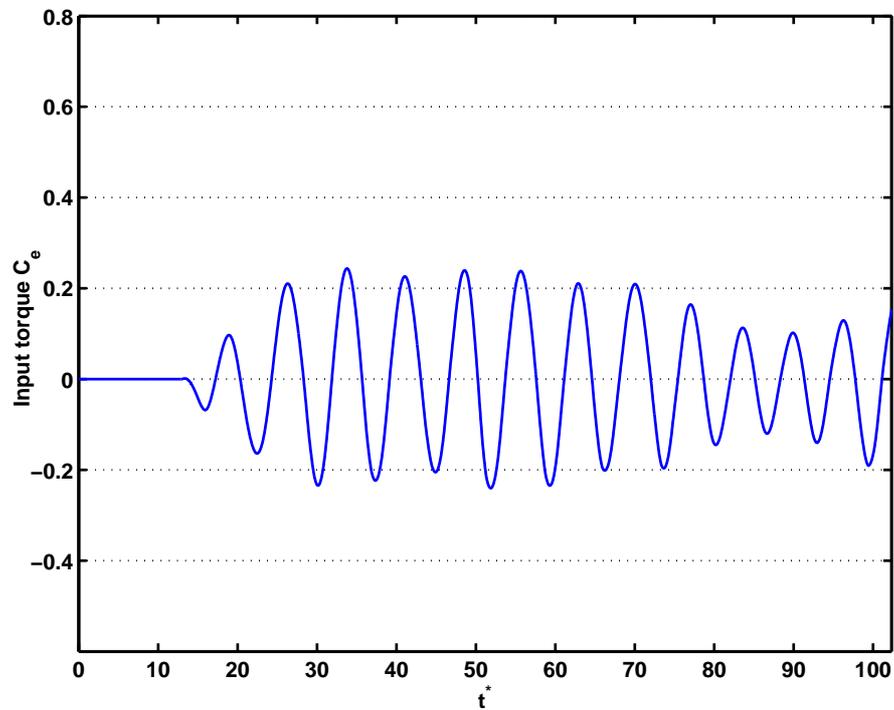


(b) Controller Torque

Figure 5.32: Time plot of the angle evolution and torque input for the flow/plate system controlled by the optimal controller with $f_n^* = 1.5 S_s$ and $\Delta t^* = 0.04$

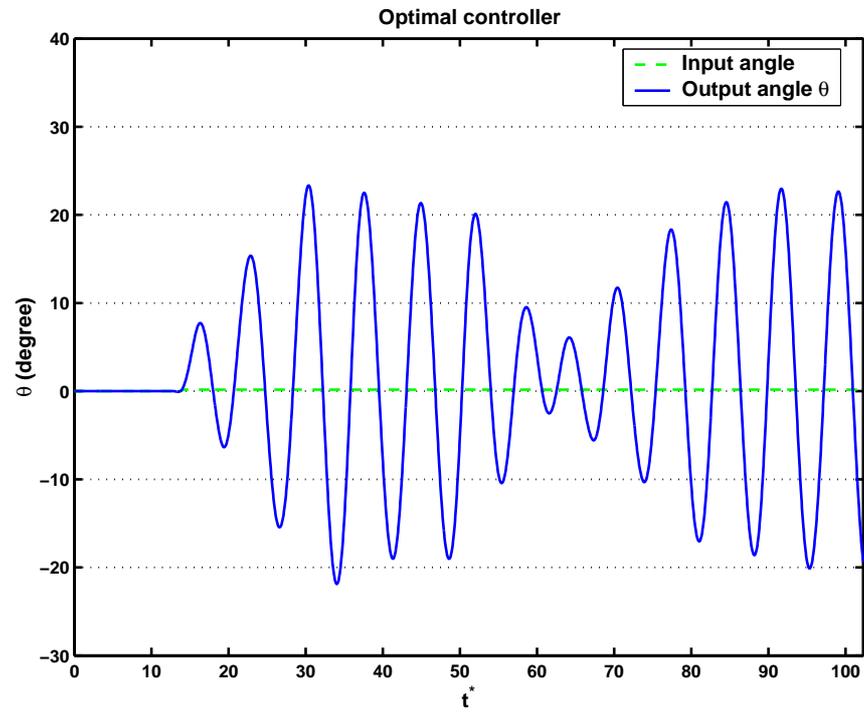


(a) $\theta(t)$

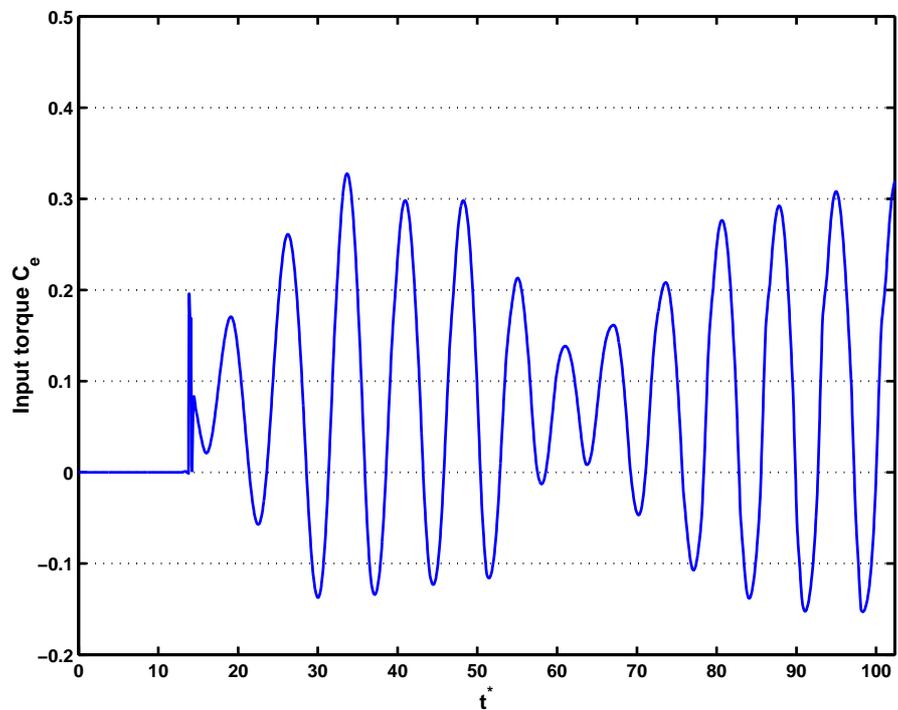


(b) Controller Torque

Figure 5.33: Time plot of the angle evolution and torque input for the flow/plate system controlled by the optimal controller with $f_n^* = S_s$ and $\Delta t^* = 0.05$

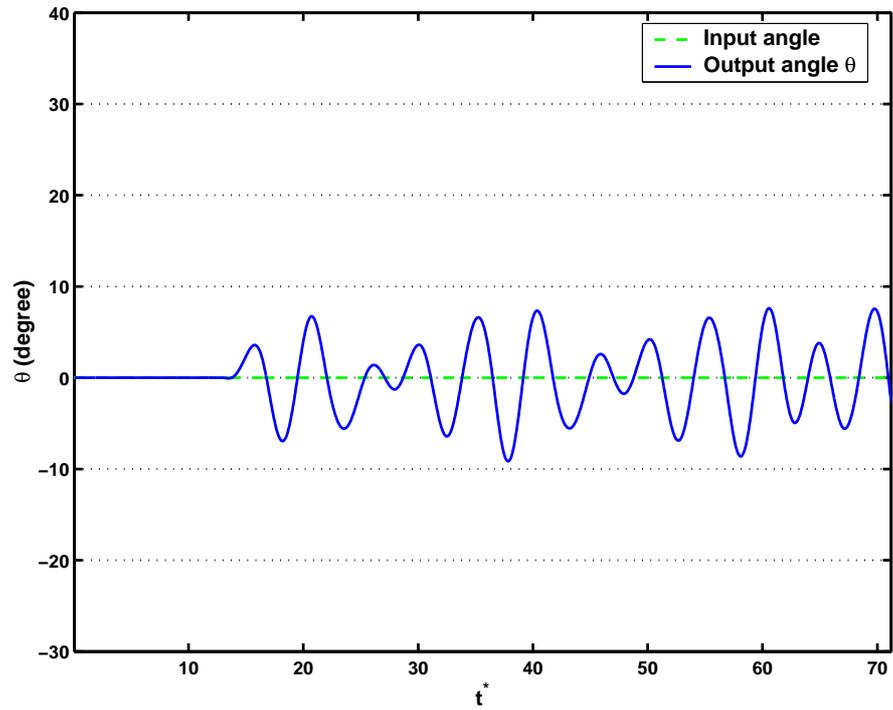


(a) $\theta(t)$

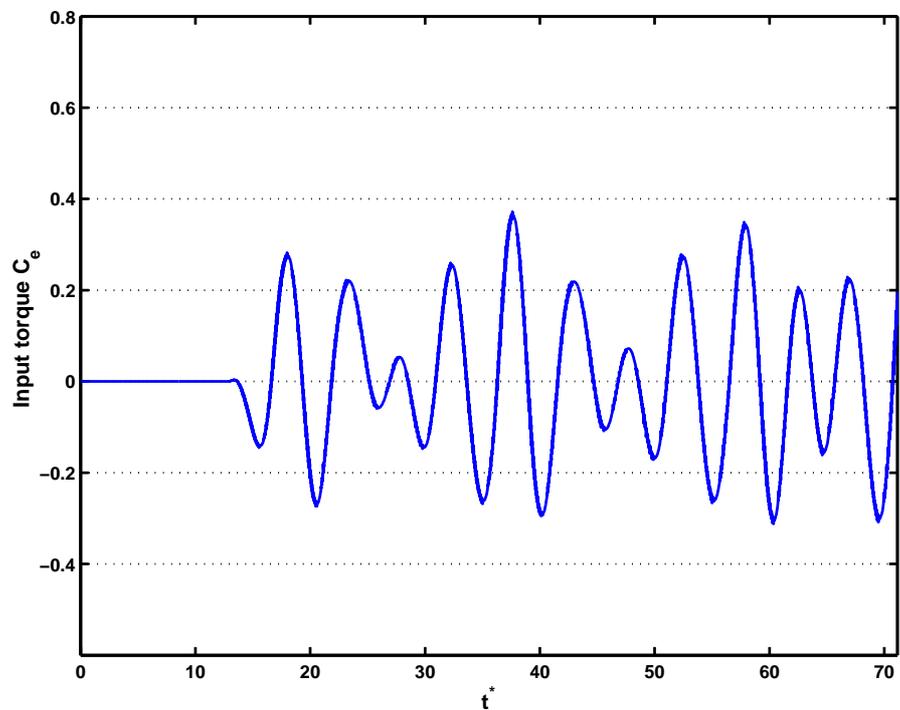


(b) Controller Torque

Figure 5.34: Time plot of the angle evolution and torque input for the flow/plate system controlled by the optimal controller with $f_n^* = 1.5 S_s$ and $\Delta t^* = 0.05$

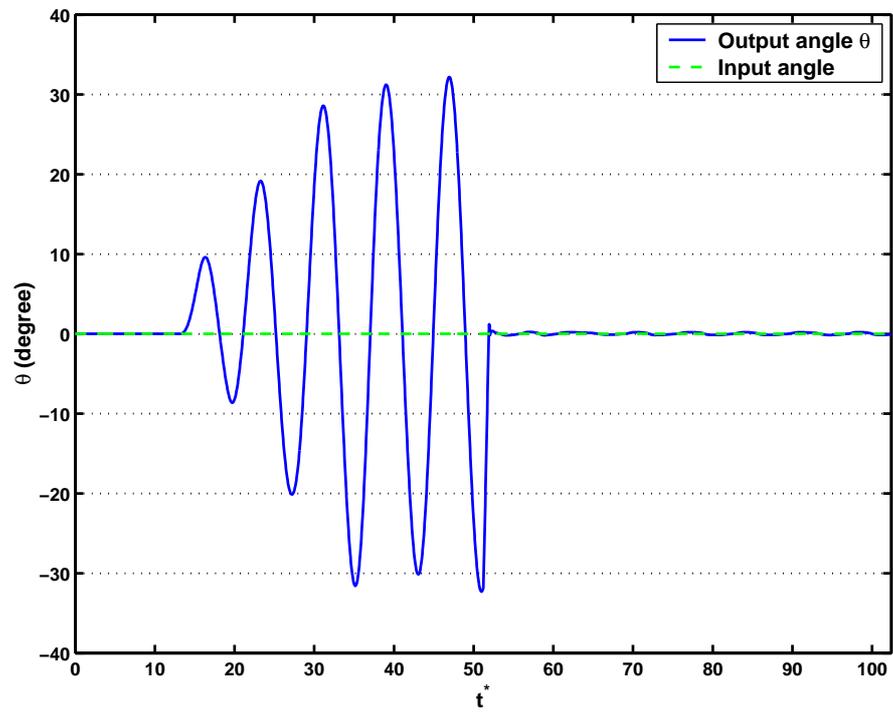


(a) $\theta(t)$

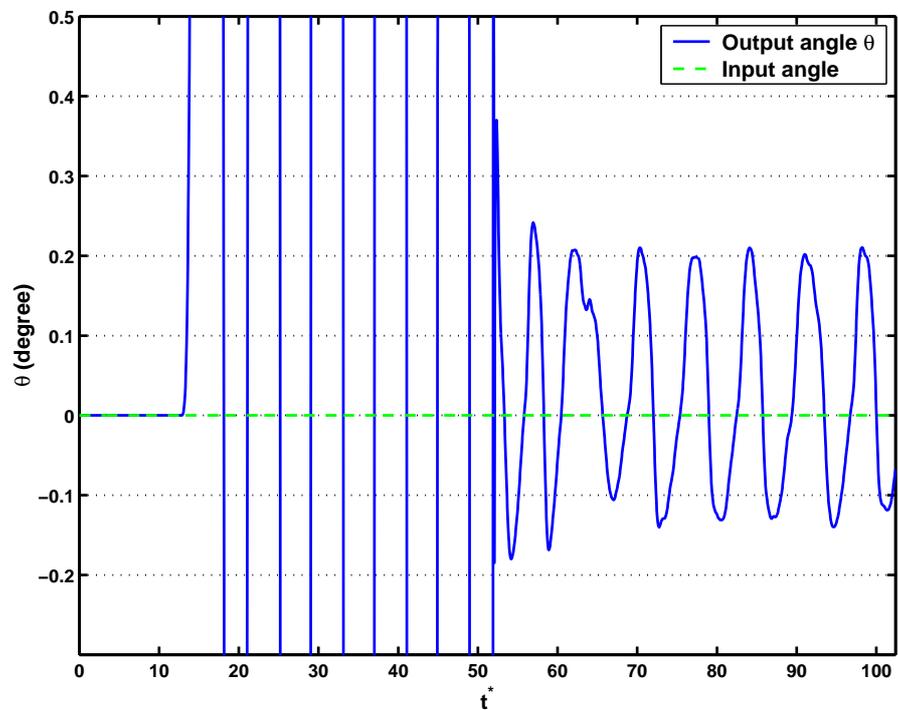


(b) Controller Torque

Figure 5.35: Time plot of the angle evolution and torque input for the flow/plate system controlled by the optimal controller with $f_n^* = 1.5 S_s$ and $\Delta t^* = 0.05$

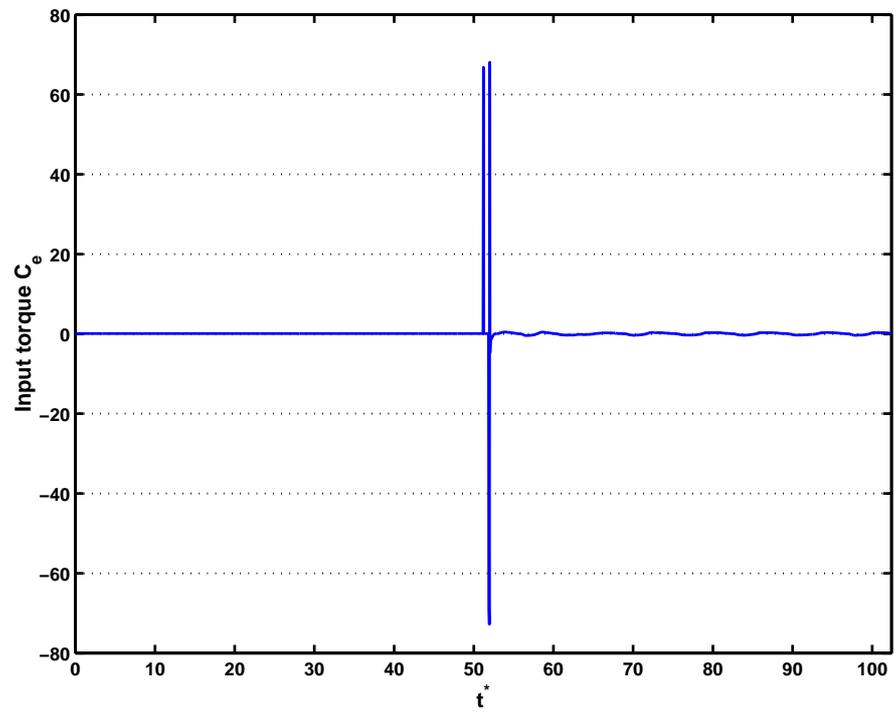


(a) $\theta(t)$

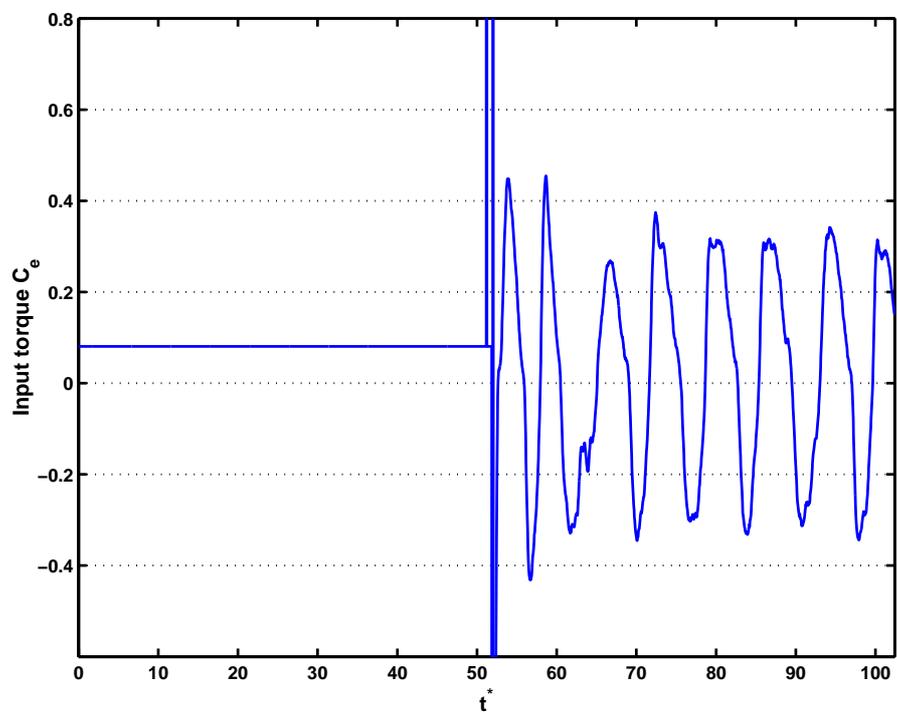


(b) $\theta(t)$ Detail

Figure 5.36: Time plot of the angle evolution for the flow/plate system with $f_n^* = S_s$, $\Delta t^* = 0.05$ and the fuzzy-logic controller enabled at $t^* = 51.2$

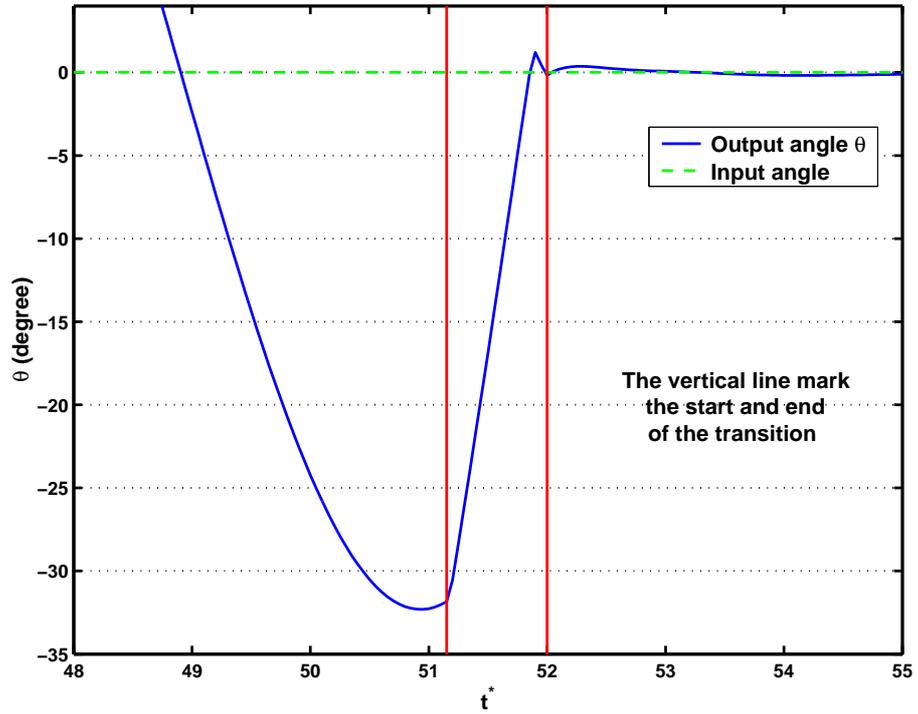


(a) Controller Torque

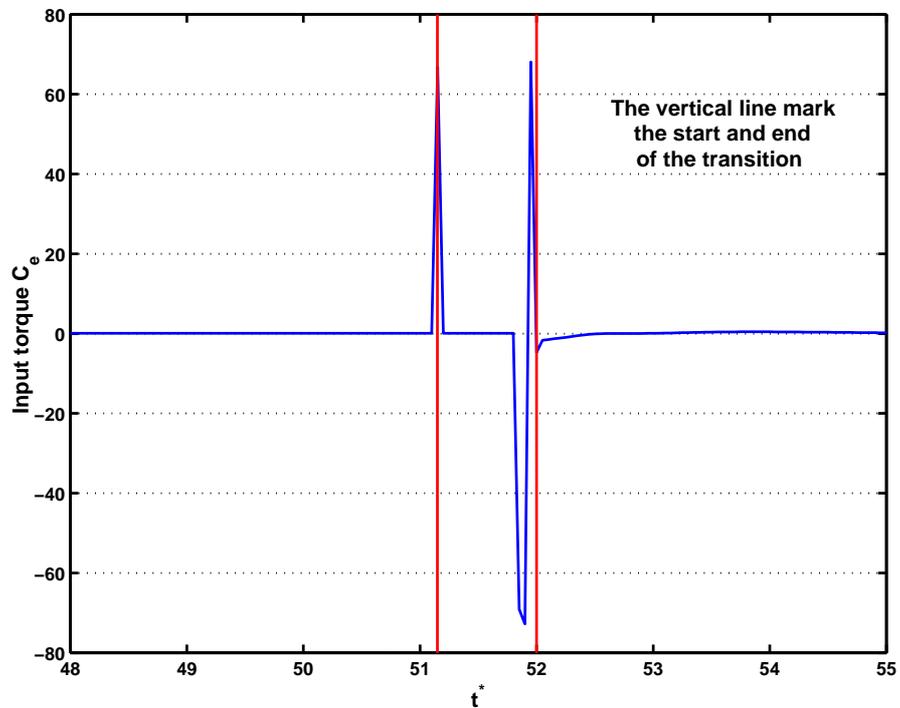


(b) Expanded view of Controller Torque history

Figure 5.37: Time plot of the torque input for the flow/plate system with $f_n^* = S_s$, $\Delta t^* = 0.05$ and the fuzzy-logic controller enabled at $t^* = 51.2$



(a) θ during Transition



(b) Controller Torque during Transition

Figure 5.38: Time plot of the angle evolution and torque input for the flow/plate system with $f_n^* = S_s$, $\Delta t^* = 0.05$ and the fuzzy-logic controller enabled at $t^* = 51.2$ during transition

5.4.2 Prescribed plate motion

In this section, the fuzzy controller and then the optimal controller have the same design as tested in sections 5.2.2 and 5.4.1.

5.4.2.1 Step input

The controllers will be tested with a sudden step input. Because the fuzzy-logic controller needs a finite angular velocity and in order to reproduce a step input of 10° , the plate is first stabilized at $\alpha = 55^\circ$, and then will be commanded with an angular velocity of $\dot{\theta} = 100 \text{ degrees/s}$ until $\alpha = 65^\circ$. The optimal controller uses the same design as the controller used in section 5.2.4, and thus integrates the Kalman filtering.

The flow simulation uses the same parameters as in section 5.4.1. Thus the reduced damping was chosen as $\xi^* = 0.02$. The motion is started at the 256th simulation iteration, and both controllers are activated from the simulation beginning. In that case, the prescribed motion is commanded at zero up to the 256th iteration, so that as the plate is blocked up to the 256th iteration ($\theta = 0$ and $\dot{\theta} = 0$) the output C_a is equal to zero up to the 256th iteration and the controller does not interfere with the plate motion. Again, two different reduced frequencies were taken at $f_n^* = S_s$ and $1.5 S_s$ or $0.77 S$ and $1.16 S$ with S_s and S respectively the Strouhal number from the numerical simulation and the experimental Strouhal number for a fixed ellipse with $\alpha = 55^\circ$.

First, tables 5.7 and 5.8 also provide a quick summary for result comparison between the two controllers. In these tables, one must note that the Error *rms* past 20.69 must be tempered by the fact that there is a permanent error remaining with the fuzzy controller as will be seen in later paragraphs and figures 5.39 and 5.41. If one removes the influence of this permanent error, one obtains for the fuzzy logic controller a Error *rms* of 0.049 and 0.086 for $f_n = 1.5 S_s$ and $f_n = S_s$ respectively which are more similar to the optimal controller values even if a little better for the optimal controller.

Then, results are presented for the fuzzy logic controller by plotting for $f_n^* = S_s$ the angular response, the angular error and the controller torque input in figures 5.39, 5.41 and 5.42 respectively. In figure 5.43 is plotted the power divided by C_p (see section 5.2.2) for this particular case. The results are then shown using $f_n^* = 1.5 S_s$ with the same controller with again the angular response, the angular error, the controller torque input as well as the power divided by C_p in figures 5.40, 5.44, 5.45 and 5.46 respectively.

To compare these, results for the optimal controller are presented with the same kind of plot, that is the angle response, the angular error, the controller torque input and the power divided by C_p respectively for $f_n^* = S_s$ in figures 5.47, 5.49, 5.50 and 5.51. The same results are also given for $f_n^* = 1.5 S_s$ in figures 5.48, 5.52, 5.53 and 5.54. Note that in the angle response plot in figure 5.39, 5.40, 5.47 and 5.48 θ was plotted which is linked to the ellipse incidence angle α by $\alpha = \alpha_0 + \theta$ with $\alpha_0 = 55^\circ$ here.

	Fuzzy logic controller		Optimal controller	
	$f_n^* = S_s$	$f_n^* = 1.5 S_s$	$f_n^* = S_s$	$f_n^* = 1.5 S_s$
<i>Error rms in degree for $t^* = [13.79, 20.69]$</i>	0.4	0.5	0.89	0.88
<i>Error rms in degree for $t^* > 20.69$</i>	0.23	0.4	0.049	0.043
$D\%$	14.69%	14.72%	42.3%	42.7%

Table 5.7: Error *rms* and $D\%$ for the optimal and fuzzy logic controller. The *rms* is the common *RootMeanSquare* formula. The % are relatively to the end value of the input θ .

From tables 5.8, 5.7 and figures 5.39, 5.40 one can see that the fuzzy logic controller successfully manages to stabilize the ellipse throughout the step input also there remains a position error after the step input⁴). The origin of this position error will be detailed in a later paragraph. Figures 5.47 and 5.48 show that this also holds true the optimal controller also stabilize the ellipse for the same conditions, but without the constant position error. Nevertheless, remark that the fuzzy logic controller is faster at following the prescribed θ than the optimal controller through the step input. This is also apparent when comparing the angular error plots in figures 5.41(a), 5.44(a) for the fuzzy logic controller and figures 5.49(a) and 5.52(a) for the optimal controller. One can see that around the time where the step input takes place, there is a large angular error $\Delta\theta \simeq 4.5^\circ$ in figure 5.49(a) where $f_n^* = S_s$, whereas the corresponding θ plot (figure 5.47) shows that the maximum θ overshoot (compared to the prescribed 10°) is about 2° . The same result is obtained for $f_n^* = 1.5 S_s$ (see figures 5.48 and 5.52(a)). When

⁴The constant position error can be seen through two ways: the angle evolution plot with the difference in value between the input θ and the output θ ; and the non-zero mean value of the error plot. Indeed, the input θ being the prescribed position for the ellipse, thus the error is equal to: input θ - output θ

comparing the corresponding plots for the fuzzy logic controller (figures 5.39 and 5.41(a) for $f_n^* = S_s$, figures 5.40 and 5.44(a) for $f_n^* = 1.5 S_s$), one sees that this difference is of less consequence.

However, as with the stabilized fixed angle case (section 5.4.1) there remains a constant position error of about 0.5° for $f_n^* = 1.5 S_s$ and 0.3° for $f_n^* = S_s$ for the fuzzy logic controller (see difference between the input θ and output θ). This also appears with the error *RMS* in table 5.7. Indeed in table 5.7, although the *RMS* past the motion seems to be much larger than for the optimal controller, this is in fact due to the permanent error for the fuzzy logic controller. When one removes the mean value of the error past the motion, the error *RMS* becomes for the fuzzy logic controller : 0.049° and 0.08° for $f_n^* = 1.5 S_s$ and $f_n^* = S_s$ respectively. These values are comparable to the optimal controller.

It was not possible to find the exact cause for this error as section 5.2.5 has shown a quasi-zero position error. One can postulate that it is due to the e and \dot{e} *zero* fuzzy sets definition. Indeed, if the ranges defined are a bit too large and in conjunction with the Δt^* induced delay, this might imply that once θ is within the range of the e *zero* fuzzy set and $\dot{\theta}$ within the range of the \dot{e} *zero* fuzzy set, the controller is stabilized around the first value close enough to the prescribed value. Reducing Δt^* to $\Delta t^* = 0.04$ has produced results equivalent to those found in figure 5.39, 5.41 and 5.42. Further tests with lower Δt^* would be interesting to assess the timestep influence on the fuzzy-logic controller.

Otherwise, the average range of angular error oscillation is the same for the fuzzy logic controller and the optimal controller, with an amplitude of 0.1° (also observable in table 5.7 when one removes the mean error value for the fuzzy logic controller). While for the optimal controller this value is similar to the one found in the stabilized ellipse case (figures 5.31 and 5.32), this is an improvement for the fuzzy logic controller (figures 5.29 and 5.30). Again, recall that there is no dependence on f_n^* to design the fuzzy logic controller, in contrast to the optimal controller.

Finally in table 5.7, the error *RMS* and *D%* during $t^* = [13.79, 20.69]$ shows that during the motion the fuzzy logic controller enables a slightly better control of the motion with an error *RMS* lower than for the corresponding optimal controller. The

change in $D\%$ enables to draw a similar conclusion.

When comparing the torque plots related to the fuzzy logic controller in figures 5.42(a) and 5.45(a) and the torque plots related to the optimal controller in figures 5.50(a) and 5.53(a), one can see that the torque produced during the step input by the fuzzy logic controller is larger than for the optimal controller. It is difficult to have a precise mathematical analysis of the fuzzy logic controller with the fuzzy sets used here, however it can be postulated that this is due to the narrow fuzzy set defined at zero. This and the fact that the other input fuzzy sets were tightened implies that the transition from one rule to another is less smooth. This enables a higher torque input to be fed to the system from the beginning of the step input. On the other side, it also implies that the slight positive and negative angle overshoot (in figure 5.41 for example) could be overcome by a broader zero fuzzy set for \dot{e} . Nonetheless, the high torque also helps to have a faster rise time for the fuzzy logic controller during the angle change. Again, the behavior of the fuzzy logic controller could be sharpened by tuning the fuzzy sets, but there is a trade-off to be expected between speed and precision with the input and fuzzy sets chosen here.

Note though that when the plate is slightly oscillating after the step input (quasi periodic motion), when comparing figures 5.42(b) and 5.45(b) (for the fuzzy logic controller) and figures 5.50(b) and 5.53(b) (for the optimal controller), the torque oscillations range is the same for both controllers. This means that both have reached a similar state, which is already visible through the angle plots.

Another remark comes as one can see the remaining oscillations in figures 5.41(b) and 5.49(b) are larger than when using the identified flow model with $f_n^* = S_s$ (figures 5.13(b) and 5.22(b)). This can be an effect of the time discretization applied to the system (the optimal controller instability when using $dt^* = 0.05$ is also a symptom of such problem). This does not seem to be due to an underestimation of the flow-induced moment applied to the plate, as in both cases the flow model and simulation C_m remains comparable in magnitude and frequency. It is difficult to assess and would require further inquiry by using different simulation timestep and by a better understanding of the computation link between the simulation and the Simulink model. An easy way to decrease the influence of the discretization would be to lower the Δt^* , but this would

require an adjustment in the simulation parameters (see section 3.2.3). Beside the discretization problem, the frequency of the remaining oscillations indicates that they are mainly related to the flow-induced moment on the plate, the spring damped plate f_n having little effect on the frequency of these oscillations (see figures 5.41(b) and 5.44(b) for example).

Now regarding the power used by both controller (figure 5.43 and 5.46 for the fuzzy logic controller and 5.51 and 5.54 for the optimal controller) and using table 5.8. One can see that independently of the f_n , the fuzzy logic controller has higher power peak during the motion even if they last during a slightly shorter time. Past the motion (for example past $t^* = 21$), the two power amplitude are very similar. The signal energy E_s ($E_{s,p}$ and E_s^*) in table 5.8 during and past the motion lead to a similar conclusion, although it seems that the optimal controller has slightly less power fluctuations past the motion.

	Fuzzy logic controller		Optimal controller	
	$f_n^* = S_s$	$f_n^* = 1.5 S_s$	$f_n^* = S_s$	$f_n^* = 1.5 S_s$
<i>Total Energy E_c^*</i>	-871.9	-862.72	-176.57	-164.9
<i>Energy E_c^* for $t^* = [13.79, 20.69]$</i>	-871.42 99.94%	-862.71 99.99%	-176.36 99.87%	-164.7 99.89%
<i>Energy E_c^* for $t^* > 20.69$</i>	-0.5 0.057%	-0.0089 0.0001%	-0.158 0.089%	-0.134 0.084%
<i>Total average power P_a^*</i>	-8.5	-8.42	-2.15	-2.01
<i>Average power P_a^* for $t^* = [13.79, 20.69]$</i>	-126.3	-125.03	-25.6	-23.92
<i>Average power P_a^* for $t^* > 20.69$</i>	-0.006	-0.00011	-0.0026	-0.0023
<i>Energy E_s^* for $t^* = [13.79, 20.69]$</i>	4.8009×10^6	4.6849×10^6	2.7279×10^5	2.6211×10^5
<i>Energy E_s^* for $t^* > 20.69$</i>	0.374	0.63	0.218	0.553

Table 5.8: Energy for the optimal and fuzzy logic controller. The energy is calculated from the power divided by C_p (see equation 5.6). The % are compared to the total energy.

Overall remark that f_n has little influence on the controller performance. Now using

E_c ($E_{c,p}$ and E_c^*), what emerges from the table 5.8 is that the fuzzy logic controller now produces a negative energy absorbed which means there is a better use of the global system –spring damped plate and flow– power. One also find a negative energy E_c ($E_{c,p}$ and E_c^*) for the optimal, albeit at a lower level (4 times less considering absolute value) than for the fuzzy logic controller. For the optimal controller, this results combined with the θ plot (figures 5.49 and 5.52) tends to indicate that the flow inertia is largely underestimated. After the motion, both controller show similar power level.

The average power P_a^* in table 5.8 provides further insight as it shows that although the spring is predominant for the fuzzy logic controller the error and power plots show that the motion is better “kept under control” than for the optimal controller as show the smaller angle oscillations and power fluctuations past the motion. This is also observable in table 5.7, where the Error *RMS* for $t^* = [13.79, 20.69]$ as well as the $D\%$ show that the error for the fuzzy is lower during the motion than for the optimal controller. Afterwards the level of precision is about the same notwithstanding the permanent error of the fuzzy logic controller and so is the level of energy E_c ($E_{c,p}$ and E_c^*).

Finally, considering the power, the results are not quite consistent for both controller with what has been found in section 5.2.2 with the fuzzy logic controller using far less power absorbing than the optimal controller and with a better precision along the commanded motion. Regarding the optimal controller, this shows the limit of the strict magnitude interpolation design and the need for a similar frequency interpolation. The design of the optimal controller being based on the flow model, this also show the need for an improved flow and plate model maybe by introducing an additional flow damping and flow inertia dependent on the plate angle in parallel to the steady flow model in order to take into account the flow properties. The flow model used in section 5.2.2 would also explain that now there is better use of power of the fuzzy logic controller.

For example, using results of sections 3.3.1 and 3.3.2, one could set an additional damping ξ_f and inertia J_f to a complementary flow model using:

$$M_f = -J_f \ddot{\theta} - \mu_f \dot{\theta}, \quad (5.20)$$

Although it can be observed from sections 3.3.1 and 3.3.2 that the evolution of both coefficient would be nonlinear and dependent upon θ . This would have the advantage of producing little interference with the steady angle flow model when the plate is not

moving and keeping the analogy with the spring damped plate. Furthermore, in the case of the spring damped plate it also enables to keep a physical meaning to the flow model. However due to a lack of time, this approach has not been implemented or tested and there remains that it would be related to introducing further ad-hoc parameters. It turn, this would mean a full related methodology just to determine those parameters for more generalized cases.

To conclude, despite a slight constant position error, the fuzzy-logic controller has proven that with no modification for f_n^* , it was able to adjust to the command signal faster than with a gain-scheduling type control, while keeping the spring damped ellipse/flow system stable. Moreover trials with $\rho_b/\rho < 1$ have led to results where the fuzzy-logic controller was able to handle the system, in contrast to the optimal controller. This may be attributable to a high-sensitivity of the spring damped ellipse system once placed in the closed loop, but one must not neglect the numerical effect due to the Simulink/flow simulation coupling. Finally, remark that the performance of the optimal controller are actually slightly degraded here due to the discretization of the signal in Matlab, and that some improvement should be applied to the Matlab/C link in order to have a more effective assessment.

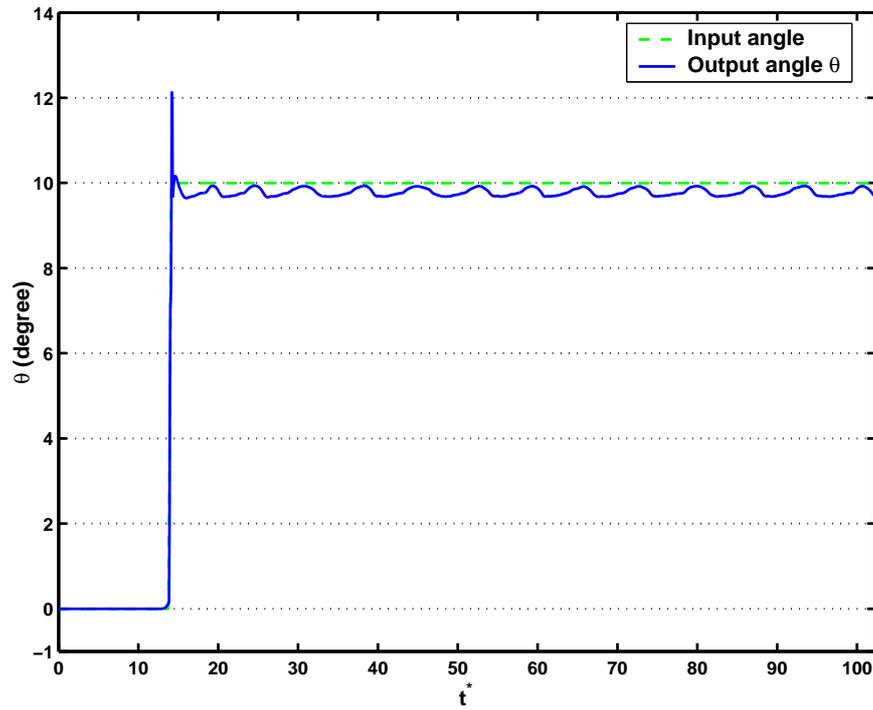


Figure 5.39: Time plot of the angle evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = S_s$

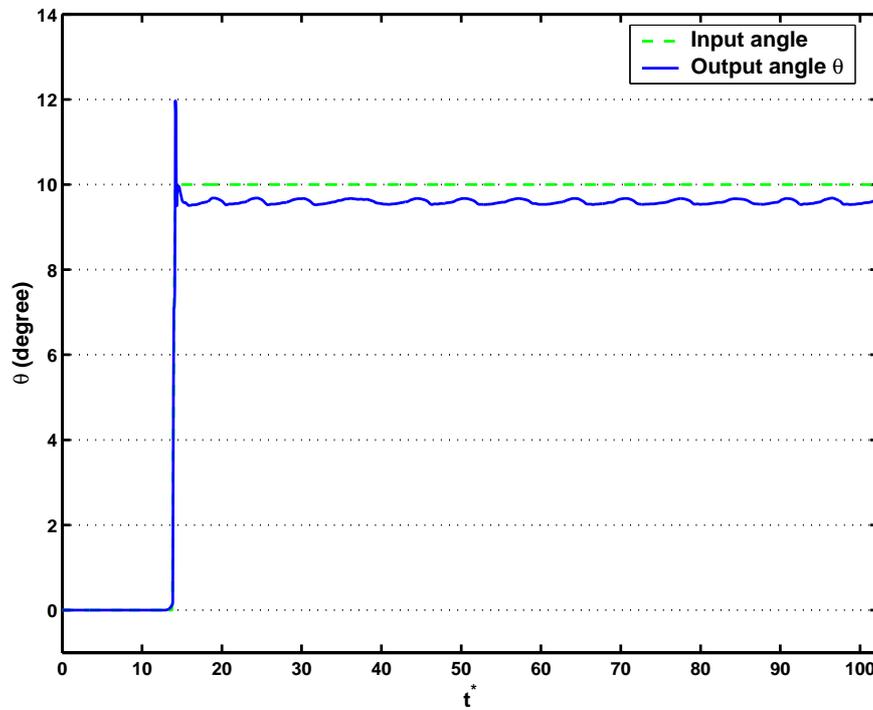
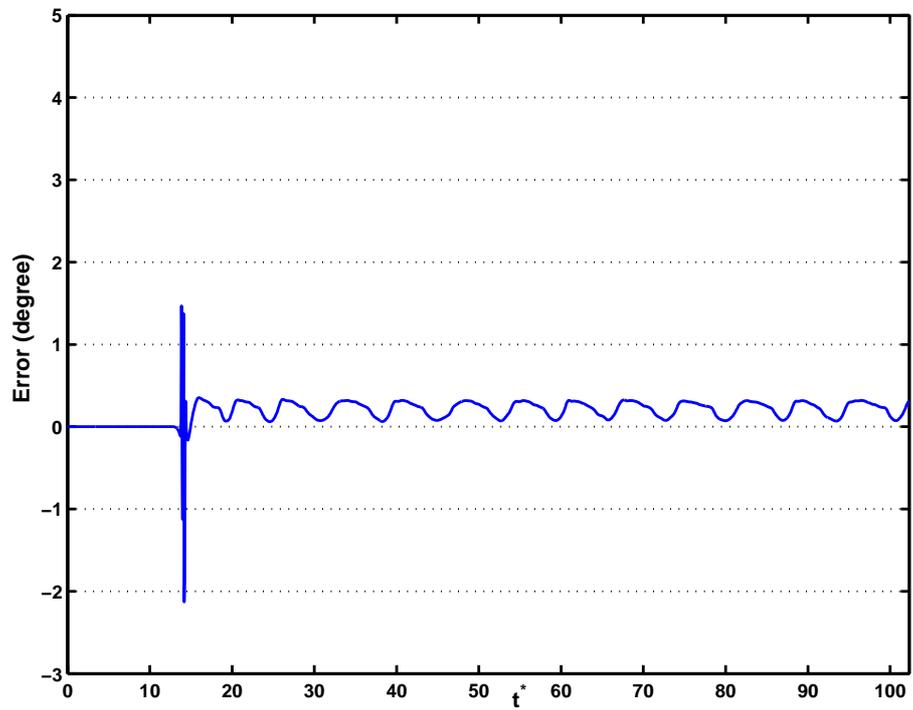
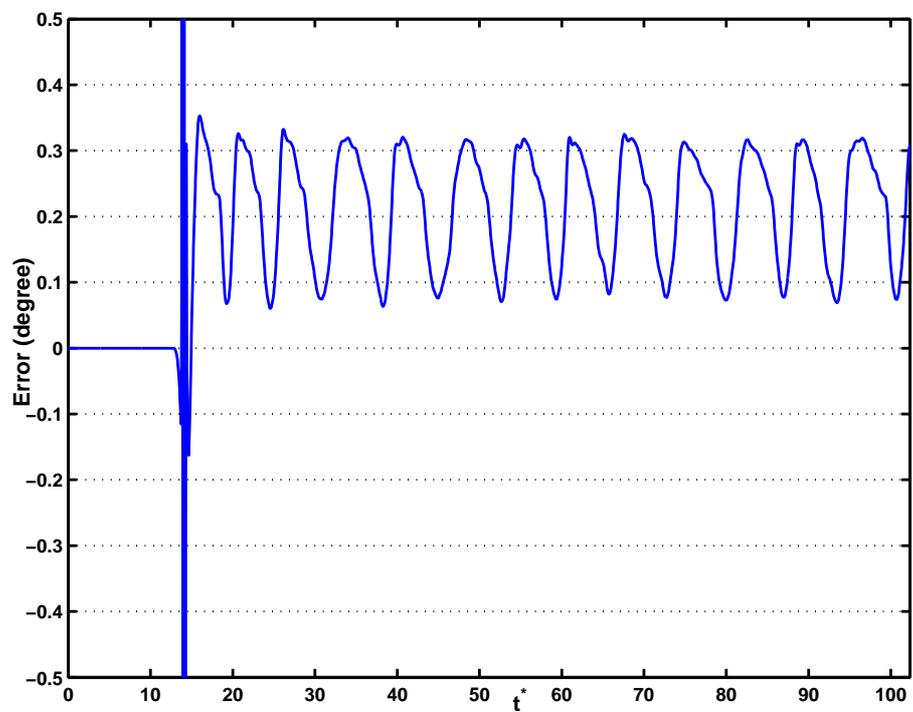


Figure 5.40: Time plot of the angle evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = 1.5 S_s$

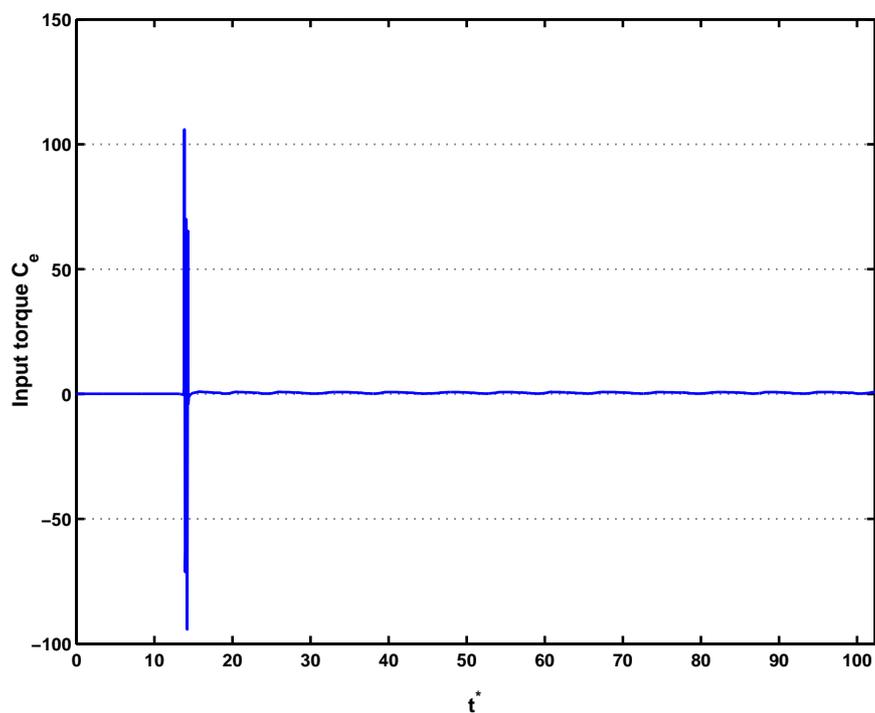


(a) Angular error

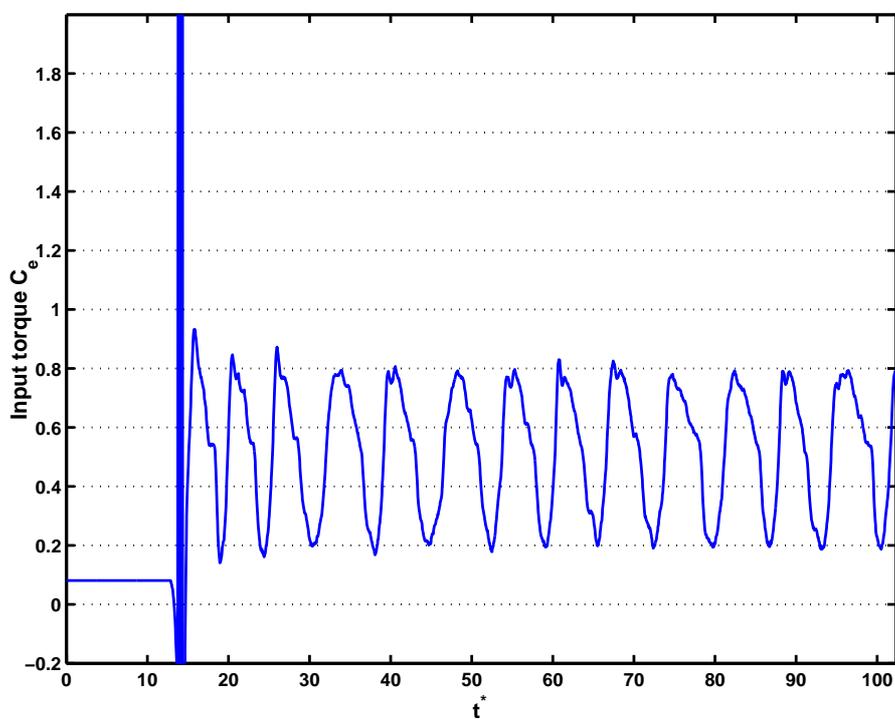


(b) Expanded view of the angular error

Figure 5.41: Time plot of the error evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = S_s$

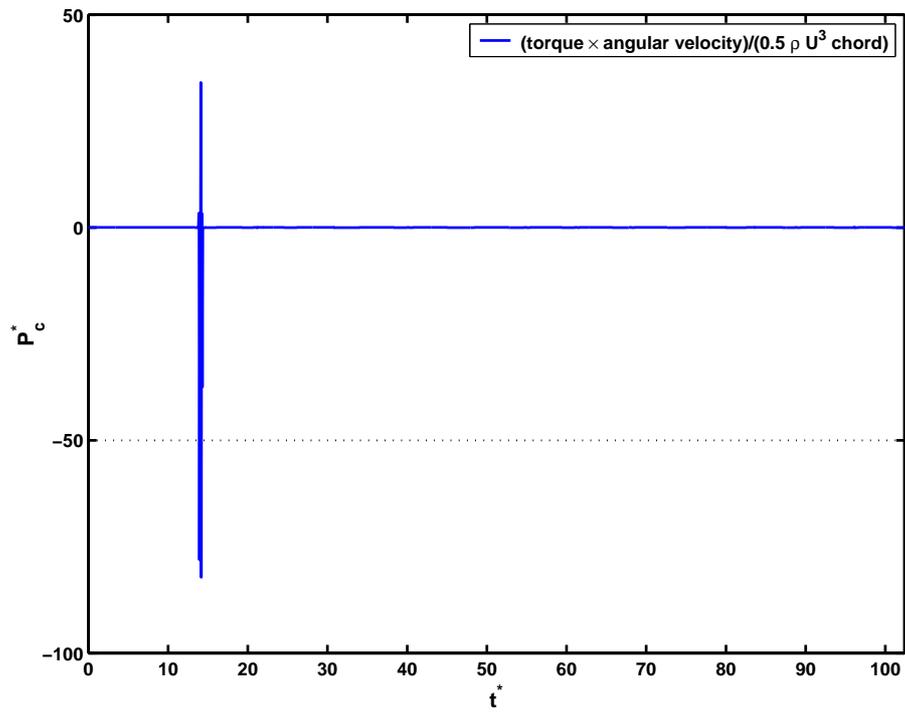


(a) Controller torque

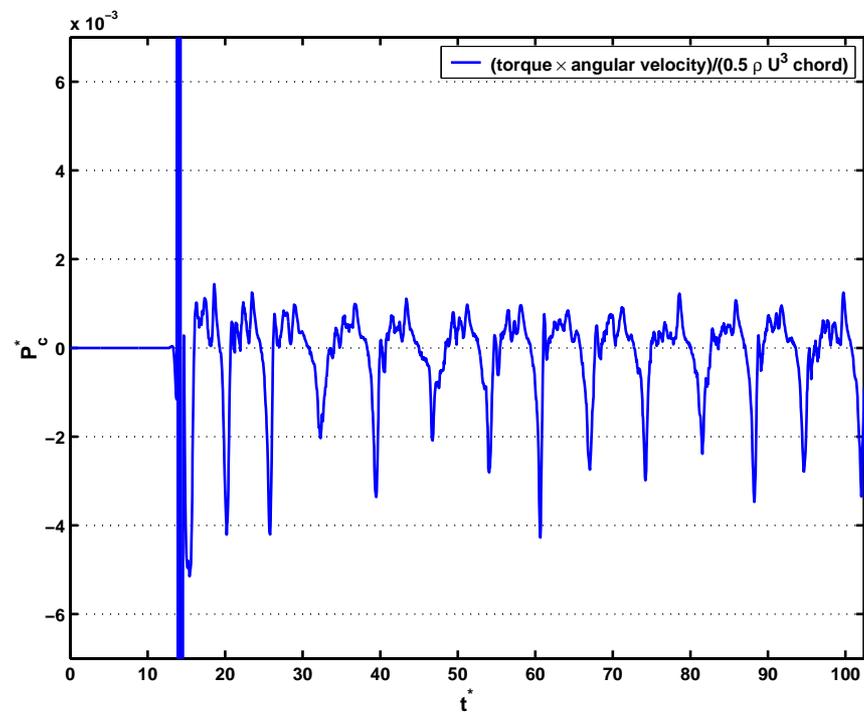


(b) Expanded view of controller torque history

Figure 5.42: Time plot of the controller torque input for the flow/plate system motion controlled by the fuzzy logic controller with $f_n^* = S_s$

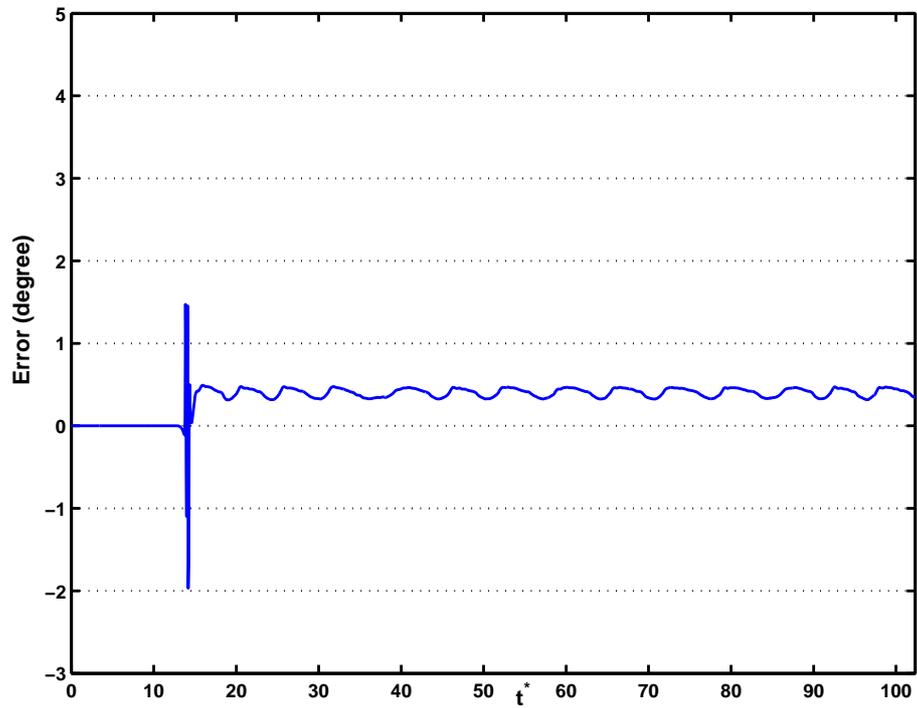


(a) Controller power

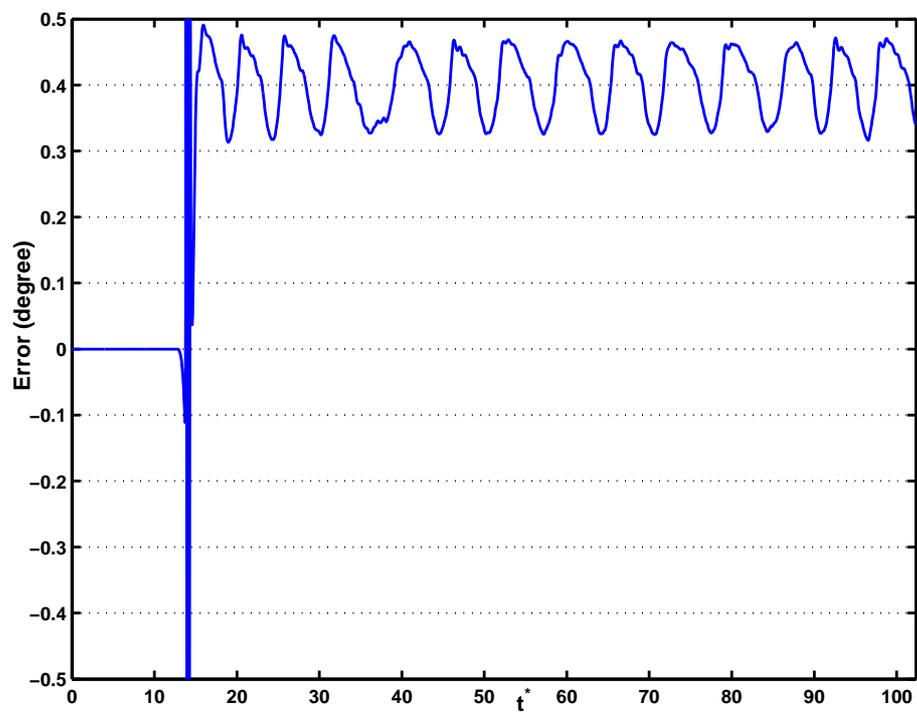


(b) Expanded view of controller power history

Figure 5.43: Time plot of the power signal evolution in time for the fuzzy logic controller with $f_n^* = S_s$. The power is divided by $(1/2)\rho U_\infty^3 L$.

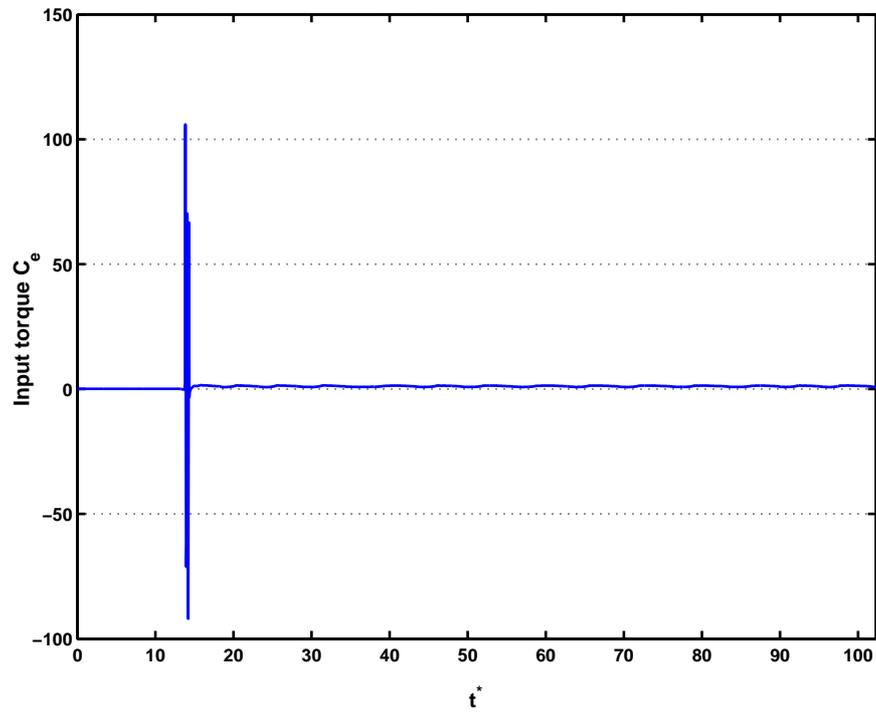


(a) Angular error

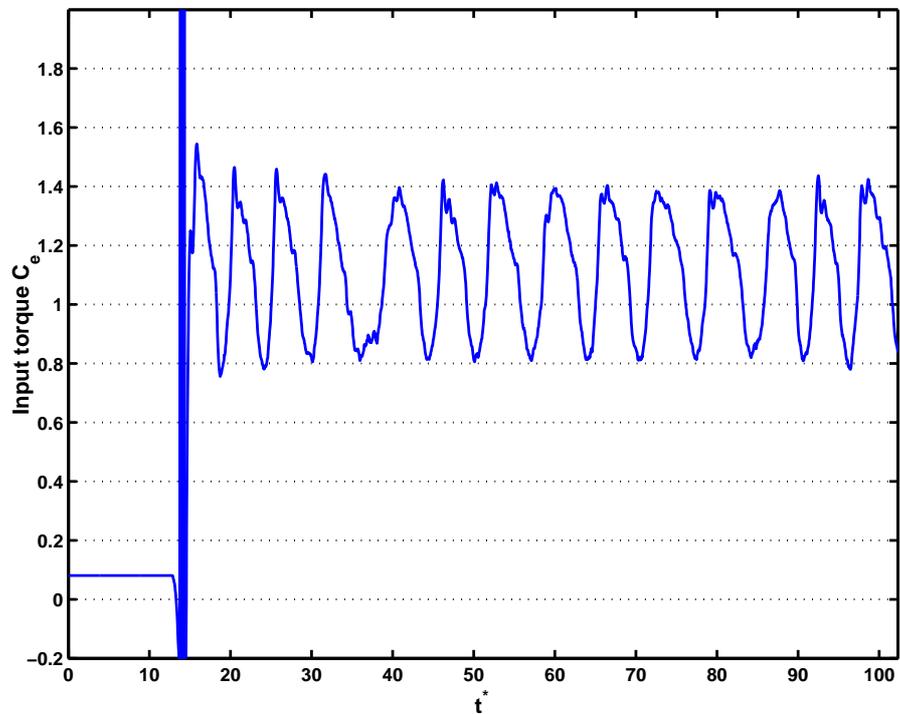


(b) Expanded view of the angular error

Figure 5.44: Time plot of the error evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = 1.5 S_s$

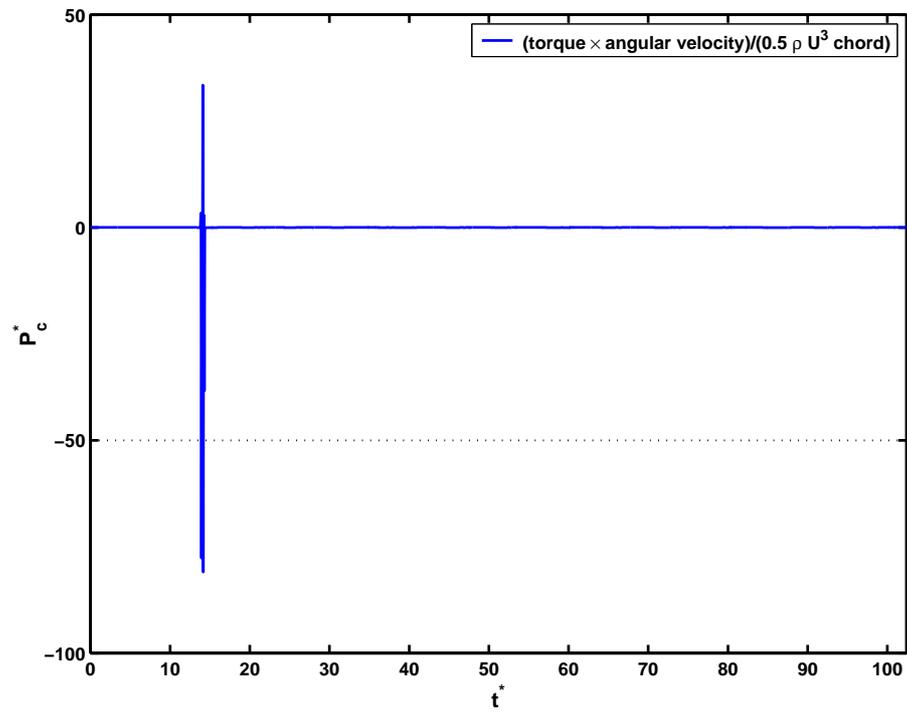


(a) Controller torque

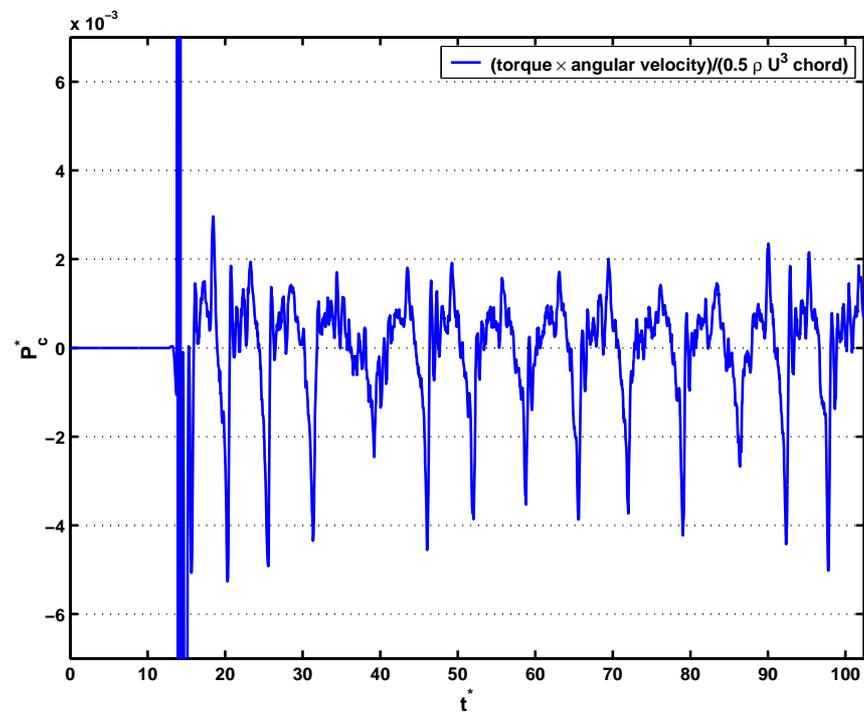


(b) Expanded view of controller torque history

Figure 5.45: Time plot of the controller torque input for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = 1.5 S_s$



(a) Controller power



(b) Expanded view of controller power history

Figure 5.46: Time plot of the power signal evolution in time for the fuzzy logic controller with $f_n^* = 1.5 S_s$. The power is divided by $(1/2)\rho U_\infty^3 L$.

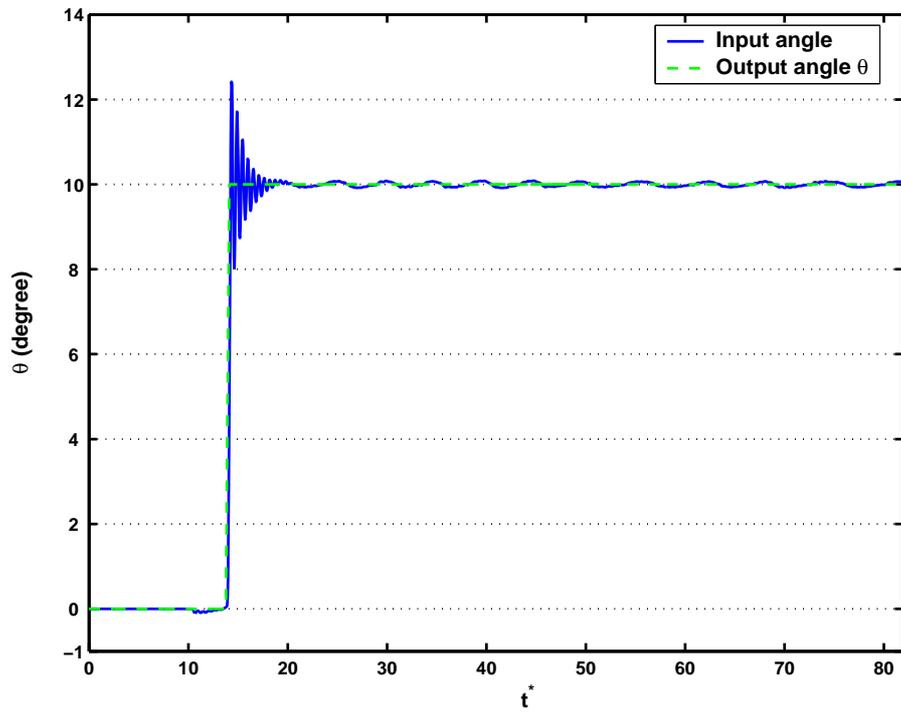


Figure 5.47: Time plot of the angle evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = S_s$

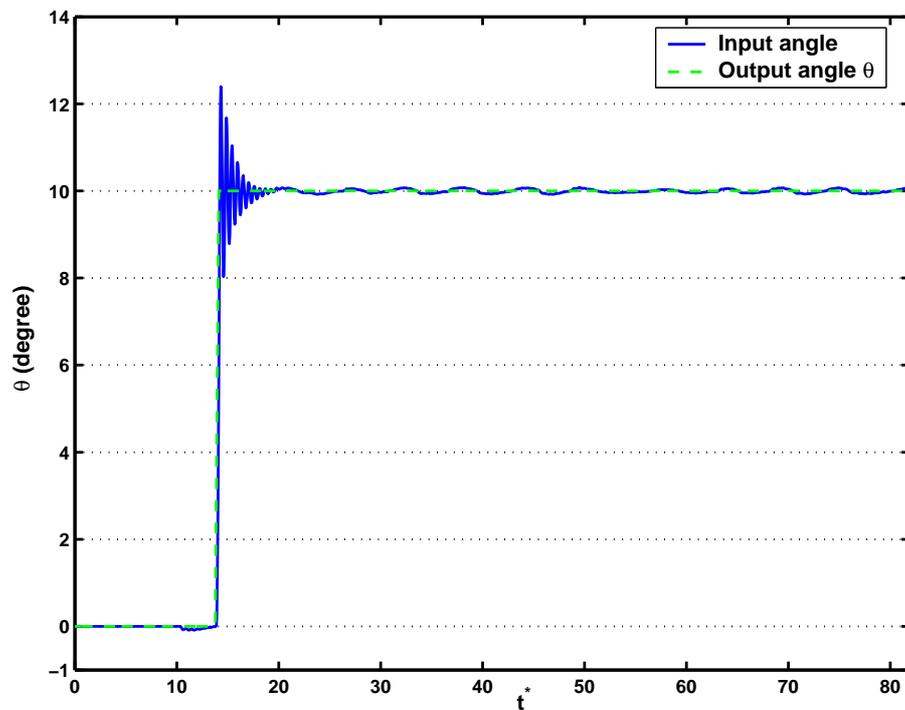
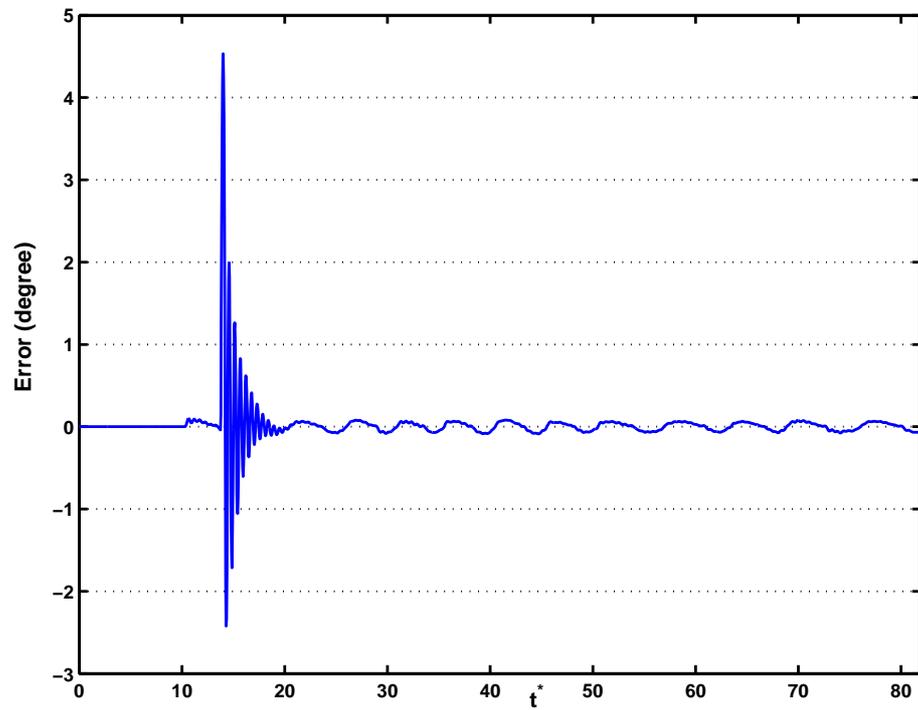
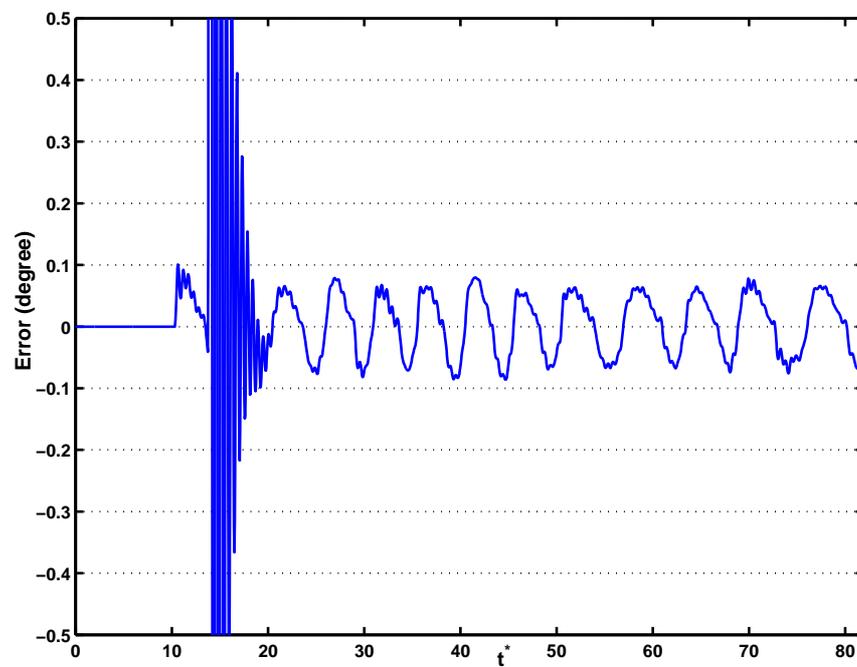


Figure 5.48: Time plot of the angle evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = 1.5 S_s$

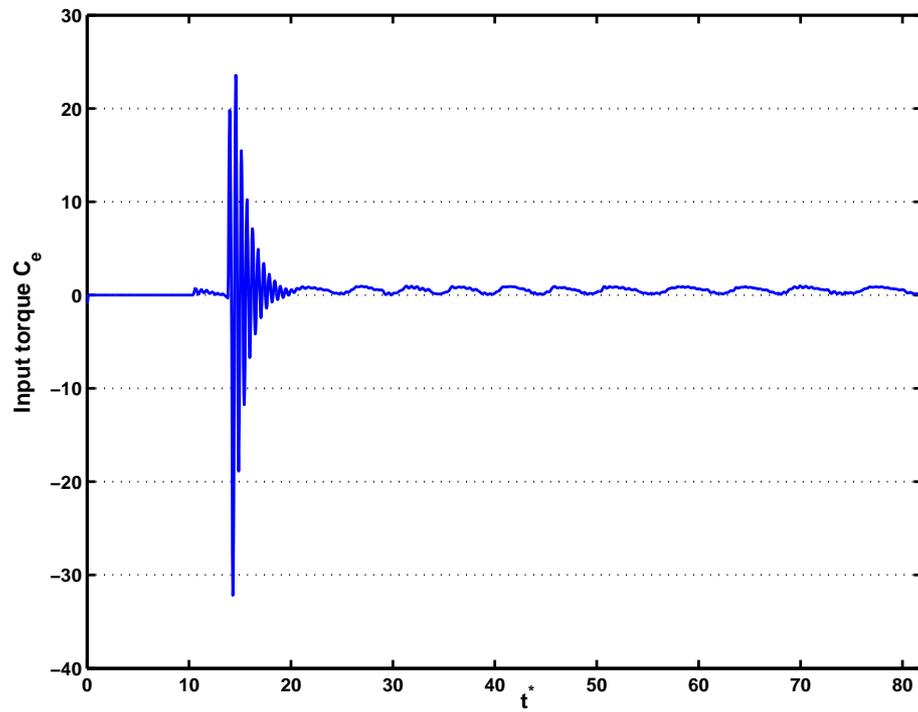


(a) Angular error

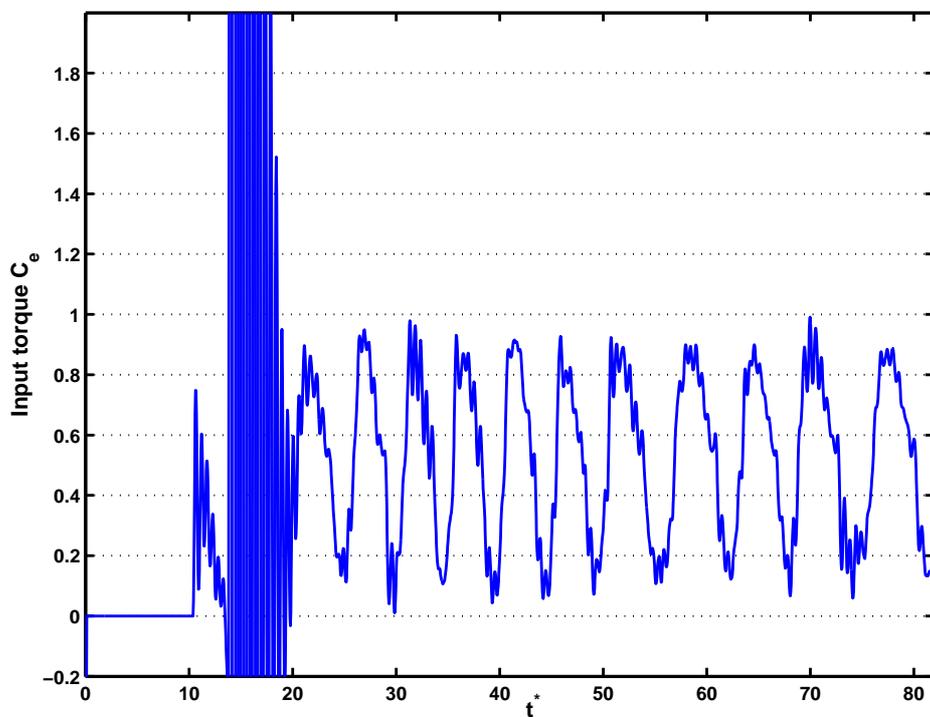


(b) Expanded view of the angular error

Figure 5.49: Time plot of the error evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = S_s$

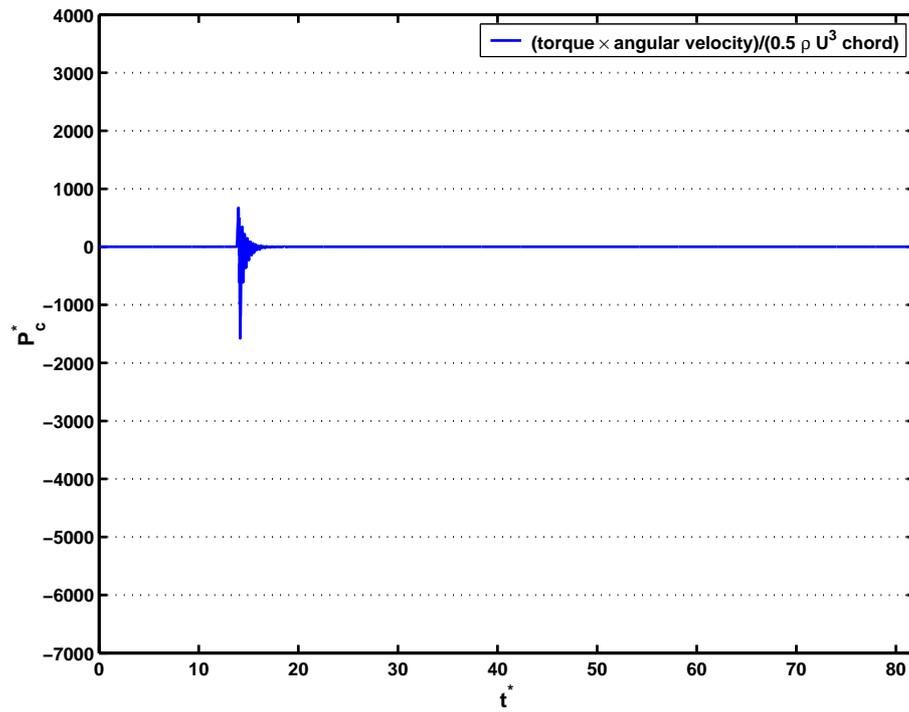


(a) Controller torque

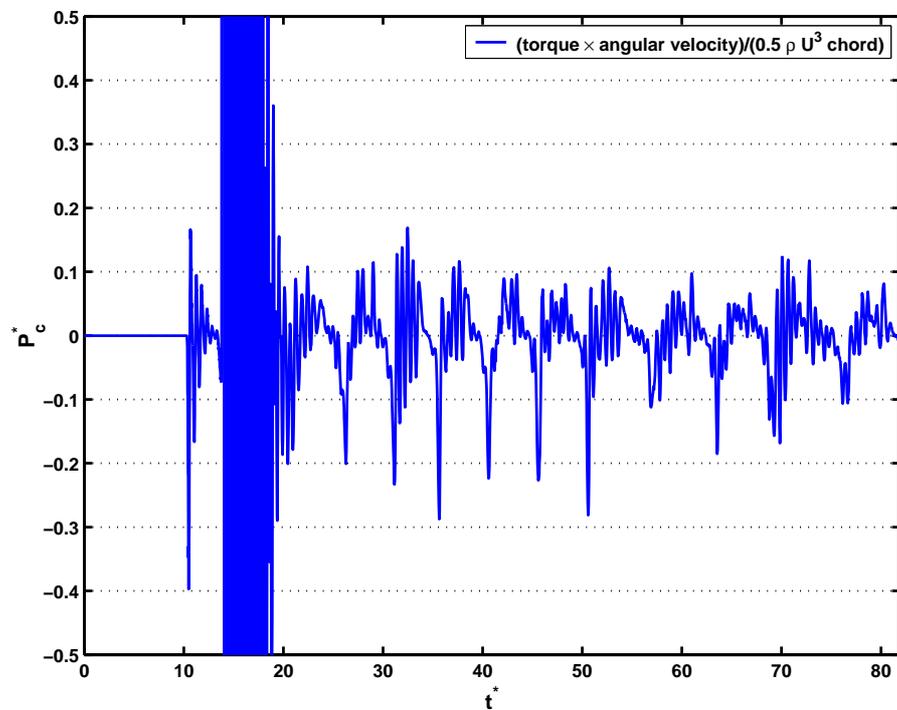


(b) Expanded view of controller torque history

Figure 5.50: Time plot of the torque input for the flow/plate system motion controlled by the optimal controller with $f_n^* = S_s$

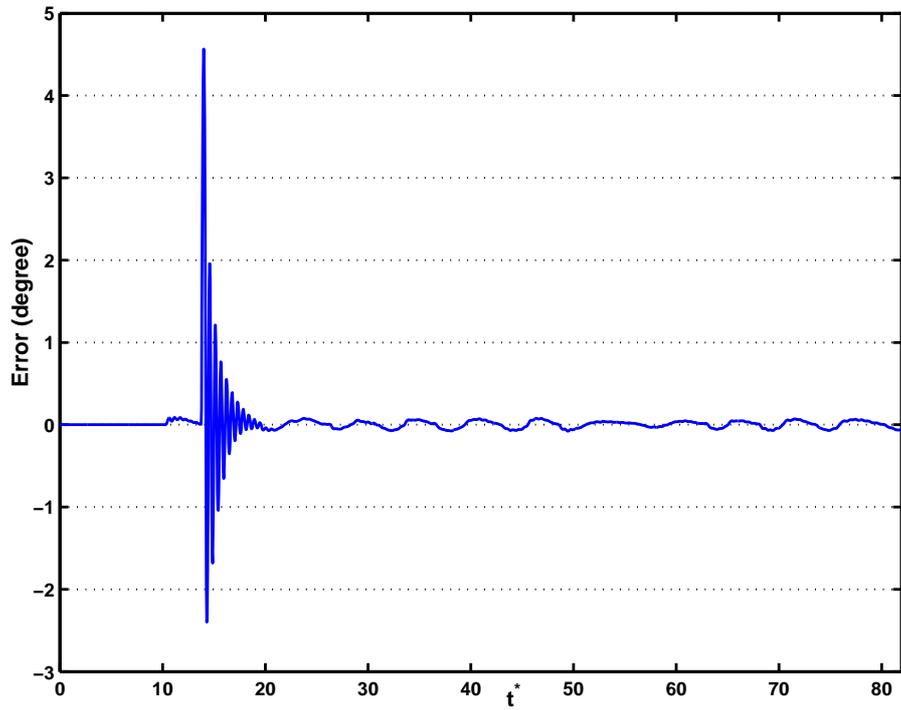


(a) Controller power

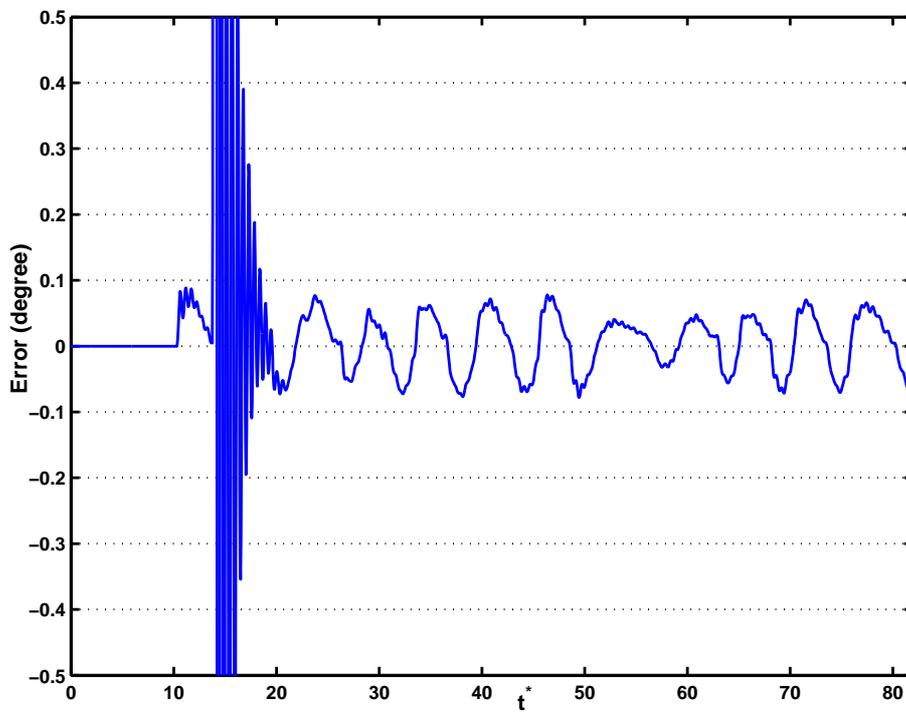


(b) Expanded view of controller power history

Figure 5.51: Time plot of the power signal evolution in time for the optimal controller with $f_n^* = S_s$. The power is divided by $(1/2)\rho U_\infty^3 L$.

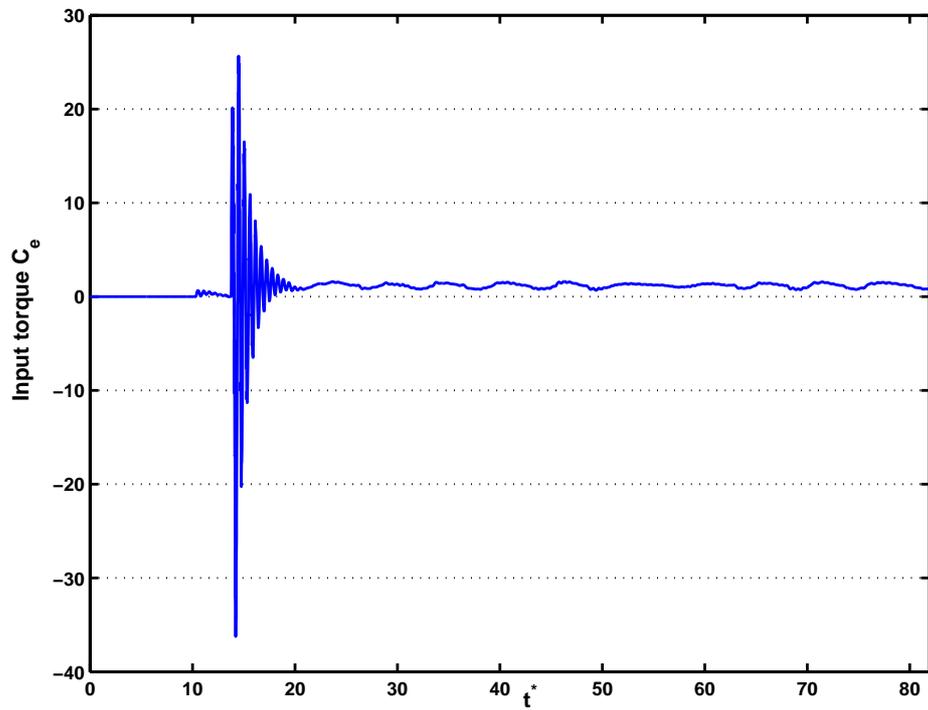


(a) Angular error

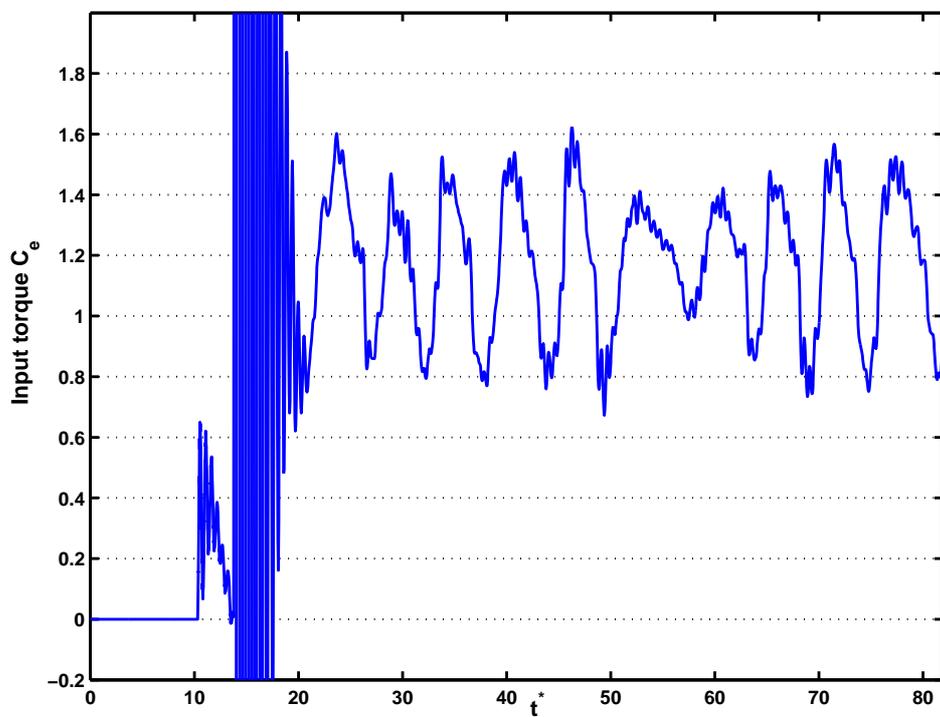


(b) Expanded view of the angular error

Figure 5.52: Time plot of the error evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = 1.5 S_s$

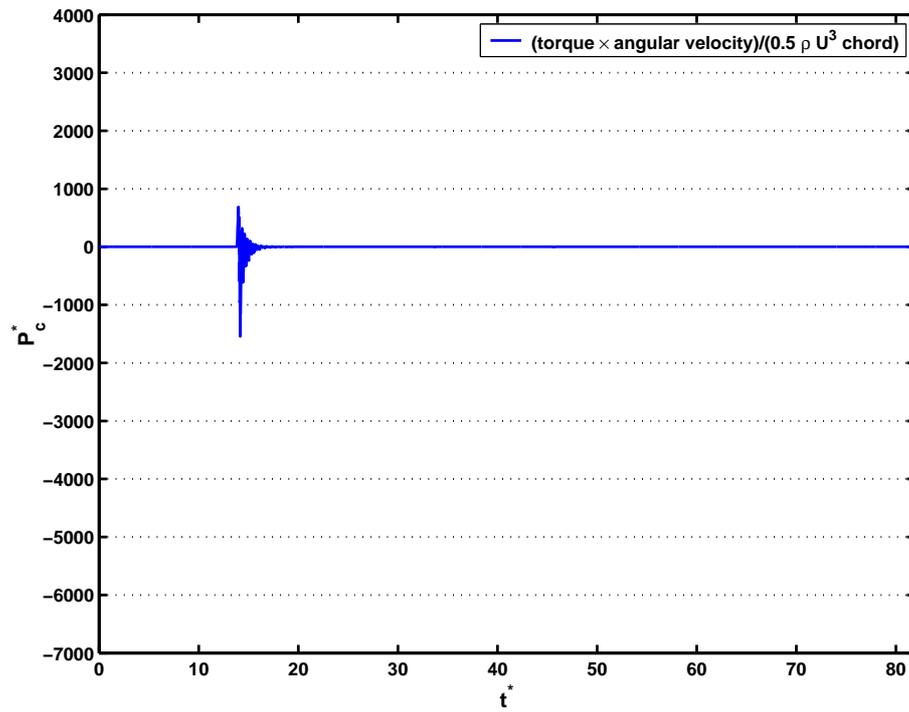


(a) Controller torque

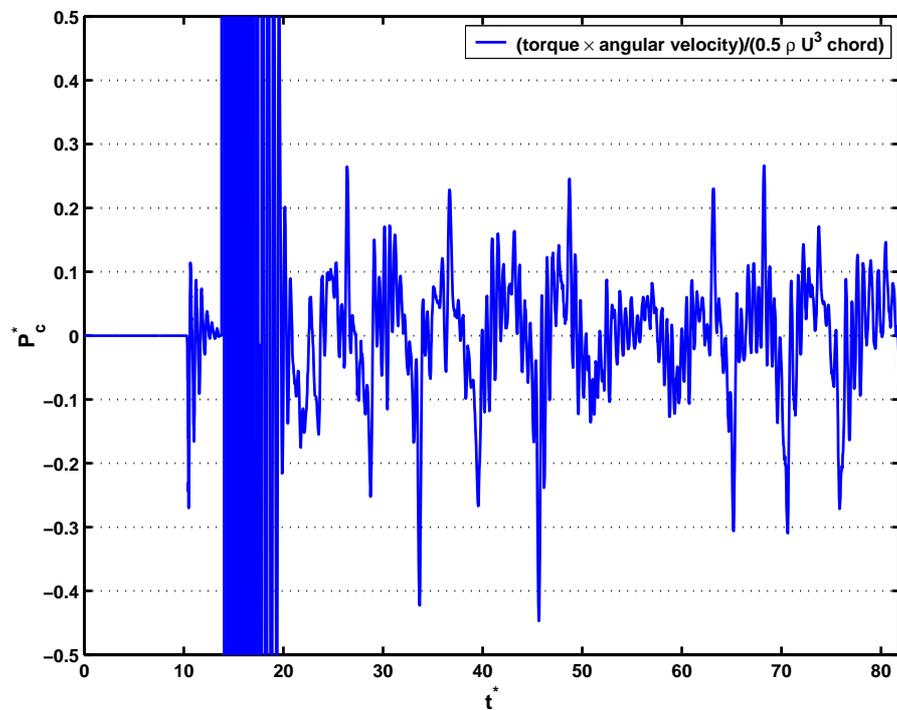


(b) Expanded view of controller torque history

Figure 5.53: Time plot of the torque input for the flow/plate system motion controlled by the optimal controller with $f_n^* = 1.5 S_s$



(a) Controller power



(b) Expanded view of controller power history

Figure 5.54: Time plot of the power signal evolution in time for the optimal controller with $f_n^* = 1.5 S_s$. The power is divided by $(1/2)\rho U_\infty^3 L$.

5.4.2.2 Oscillating input

In a second set of tests, both controller were tried for $f_n^* = S_s$ and $f_n^* = 1.5 S_s$ but instead of a step input, applied an oscillation input from $t^* = 13.78$ with the ellipse motion allowed from $t^* = 12.8$. That is, a zero input was applied up to $t^* = 13.78$, and then past this time a command signal whose time function is $f(t) = 20 \sin(2\pi f_n^* t^*)$ (in degree). Note that due to a lack of time, the case $f_n^* = 1.5 S_s$ with the fuzzy logic controller was done with $dt^* = 0.05$ instead of $dt^* = 0.04$ for every other cases.

Tables 5.9 and 5.10 provide a quick summary for result comparison between the two controllers.

Then, results are presented for the fuzzy logic controller by plotting for $f_n^* = S_s$ the angular response, the angular error and the controller torque input in figures 5.55, 5.57 and 5.58 respectively. In figure 5.59 is plotted the power divided by C_p (see section 5.2.2) for this particular case. The results are then shown using $f_n^* = 1.5 S_s$ with the same controller with again the angular response, the angular error, the controller torque input as well as the power divided by C_p in figures 5.56, 5.60, 5.61 and 5.62 respectively.

To compare these, results for the optimal controller are presented with the same kind of plot, that is the angle response, the angular error, the controller torque input and the power divided by C_p respectively for $f_n^* = S_s$ in figures 5.63, 5.65, 5.66 and 5.67. The same results are also given for $f_n^* = 1.5 S_s$ in figures 5.64, 5.68, 5.69 and 5.70. Note that, as in the previous section in the angle response plot in figure 5.55, 5.56, 5.63 and 5.64 θ was plotted which is linked to the ellipse incidence angle α by $\alpha = \alpha_0 + \theta$ with $\alpha_0 = 55^\circ$ here.

From figures 5.55, 5.56, 5.63 and 5.64 one can see that both controllers manage to roughly follow the oscillations after the transition. However, again the optimal controller has a longer transition period than the fuzzy logic controller as well as a larger θ error. This is especially visible in the error plots in figures 5.57 and 5.60 for the fuzzy-logic controller and figures 5.65 and 5.68 for the optimal controller. As shown in table 5.9, the maximum errors are larger for the optimal controller than for the fuzzy-logic controller. Remark though that the performance of the optimal controller are actually slightly degraded here due to the discretization of the signal in Matlab. Every maxima

occur when the oscillation input begin at $t^* = 13.78$. The detailed error plots (figures 5.57(b), 5.60(b) for the fuzzy-logic controller and figure 5.65(b), 5.68(b) for the optimal controller) also show that this time there is no mean position error for the fuzzy-logic controller. Also observe that during and past the transition period, the error is about half that of the optimal controller (also see the error *RMS* in table 5.9). After the transition part, the error falls within $[-0.5, 0.5]$ degrees for the fuzzy logic controller, and $[-2, 2]$ degrees for the optimal controller.

Observe that figure 5.64 for the optimal controller at $f_n^* = 1.5 S_s$ is normal and simply illustrate the fact that the controller has less difficulties than when controlling the plate/flow system at $f_n^* = S_s$. It is also visible through power analysis in later paragraph. It can be added that past the motion transition the angular error are similar as shown by the *RMS* in table 5.9 and figures 5.63, 5.64.

	Fuzzy logic controller		Optimal controller	
	$f_n^* = S_s$	$f_n^* = 1.5 S_s$	$f_n^* = S_s$	$f_n^* = 1.5 S_s$
<i>Error rms in degree for $t^* = [13.79, 25]$</i>	2.78	0.5	5.12	1.639
<i>Error rms in degree for $t^* > 25$</i>	0.12	0.15	0.969	1.41
<i>D%</i>	5.838%	9.56%	246.41%	28.99%

Table 5.9: Error *rms* and *D%* for the optimal and fuzzy logic controller and oscillating input. The *rms* is the common *RootMeanSquare* formula. The % are relatively to the end value of the input θ .

From the torque history plots (figures 5.58(b), 5.61(b) for the fuzzy logic controller and 5.66(b), 5.69(b) for the optimal controller), it is immediately visible that the torque fed to the spring-damped ellipse is much higher for the optimal controller past the transition period, with the torque ranging between about -3 and 3 . There seems to be also for the optimal controller much oscillations with a shorter period than the motion itself. Note also the constant torque input for the fuzzy logic controller before the motion has been set, it has been found that this was due to a slight error in the torque C_a membership functions for A .

During the transition period, the torque plots (figures 5.58(a), 5.61(a) for the fuzzy

logic controller and figures 5.66(a), 5.69(a) for the optimal controller) reveal that the optimal controller has more difficulties at controlling the ellipse motion than is the fuzzy-logic controller during the transition phase ($-13.78 < t^* < 25$) as can be visible from the many torque oscillations during this period as well as the signal energy during transition in table 5.9. Similarly to the remaining oscillations past the transition period, it seems that they have a shorter time period than the motion. Thus, it can be deduced that these oscillations are actually symptomatic of the flow model used for the optimal controller design in section 5.2.3 as the fuzzy controller shows no such limitation.

From table 5.10 and the different power plot (figures 5.59, 5.62 for the fuzzy logic controller and figures 5.67, 5.70 for the optimal controller), one can first observe that in most cases the energy used E_c ($E_{c,p}$ and E_c^*) is negative after the transition period ($t^* > 25$). The only exception is the case $f_n^* = S_s$ for the fuzzy logic controller. This especially noticeable after the motion has been set.

	Fuzzy logic controller		Optimal controller	
	$f_n^* = S_s$	$f_n^* = 1.5 S_s$	$f_n^* = S_s$	$f_n^* = 1.5 S_s$
Total Energy E_c^*	2.086	-294.76	-3247	-1666.7
Energy E_c^* for $t^* = [13.79, 25]$	-78.59	-252.1	-2811.8	-49.5
Energy E_c^* for $t^* > 25$	80.71	-41.66	-435.34	-1171.3
	$3.87 \times 10^3\%$	14.14%	13.4%	70.3%
Total average power P_a^*	0.0254	-3.6	-39.6	-20.3
Average power P_a^* for $t^* = [13.79, 25]$	-7.02	-22.51	-250.9	-44.2
Average power P_a^* for $t^* > 25$	1.41	-0.731	-7.64	-20.57
Energy E_s^* for $t^* = [13.79, 25]$	3.8540×10^5	1.1856×10^6	7.116×10^7	1.09×10^6
Energy E_s^* for $t^* > 25$	555.72	915.64	1.195×10^4	6.985×10^4

Table 5.10: Energy for the optimal and fuzzy logic controller and oscillating input. The energy is calculated from the power divided by C_p (see equation 5.6). The % are compared to the total energy.

Also observe that the values are much higher value (in term of absolute value) than

in the previous section. This is simply a reflection of the fact that this time the plate is continually set in motion past the transition and that the controller use the energy from the coupled spring damped plate/flow system. Indeed, one can see the positive oscillation power amplitude past $t^* > 25$ from the power plots. Unsurprisingly, the signal energy E_s ($E_{s,p}$ and E_s^*) confirms that the oscillations are much higher and more numerous for the optimal controller than for the fuzzy logic controller.

In general, there is a clear difference from $f_n^* = S_s$ to $f_n^* = 1.5 S_s$ for both types of controller. For the fuzzy logic controller, the energy E_c^* has been raised from $f_n^* = 1.5 S_s$ to $f_n^* = S_s$ and that correspondingly the error RMS has been raised as well in every part of simulation. Furthermore, the power P_a^* has become positive past the transition for $f_n^* = 1.5 S_s$. A similar behavior is encountered with the optimal controller, although P_a has increased in absolute value but remains negative, whether during or after the motion has been set. Again, the error RMS has increased at least during the motion while remaining at a similar level after the transition. This behavior could come from the influence of the flow because it has been seen in section 3.3.2.2 and in particular with figure 3.66 that around $f_n^* = [S, S_s]$ there tended to be an increase in the oscillation amplitude which pointed out to a lock-in type phenomenon. Furthermore, from figure 3.66, it is visible that as f_n^* increases –equivalent to $1/f_n^*$ decreases– the oscillations (and thus the flow forces coefficient) decrease as can be seen from figure 3.69. This in turn shows the limitation of the design method whereby the flow nonlinearities are not taken into account.

Otherwise during the transition ($-13.78 < t^* < 25$), the average power P_a^* is much lower (in term of absolute value) for the fuzzy logic controller than for the optimal controller while remaining negative in every case. This tend to show that if the controllers use energy from the coupled spring damped plate/flow system, the smaller error RMS in the fuzzy logic controller cases imply that for this controller there is better use of this energy and that the behavior of the coupled spring damped plate/flow system is less predominant on the full system. It also shows that the analysis of power alone would not be sufficient for such a system and is not in itself a sufficient indicator.

Now examining E_s ($E_{s,p}$ and E_s^*) for this time period, it shows that the level of amplitude are *overall* slightly equivalent for $f_n^* = 1.5 S_s$, and much higher for the optimal controller than for the fuzzy controller for $f_n^* = S_s$, which is not surprising regarding

the corresponding torque evolution.

This is only an example of ellipse oscillation, and it would be however interesting to test this configuration with other types of flow (translating freeflow with oscillating U_∞ for example) or motion (with other frequencies for the sine commanded motion for instance), in order to assess how the flow phenomenon described in section 3.3.2.2 interferes with the controller performances. Also, from this test case, what also emerges is that f_n^* has shown to have more influence on the controller reaction than previously. This shows the interest of using such case for assessing controller design methods. In that respect, it would be interesting to make test with controller designed for one f_n^* and to simply test them with a wider range of input oscillation frequency. This task is left for future studies.

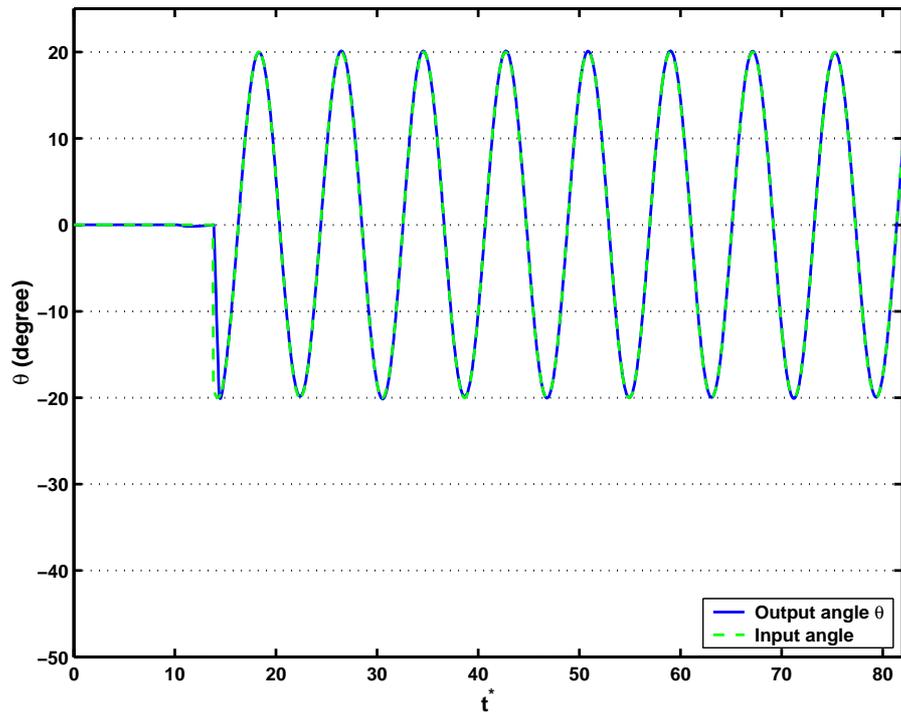


Figure 5.55: Time plot of the angle evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = S_s$ and oscillating input

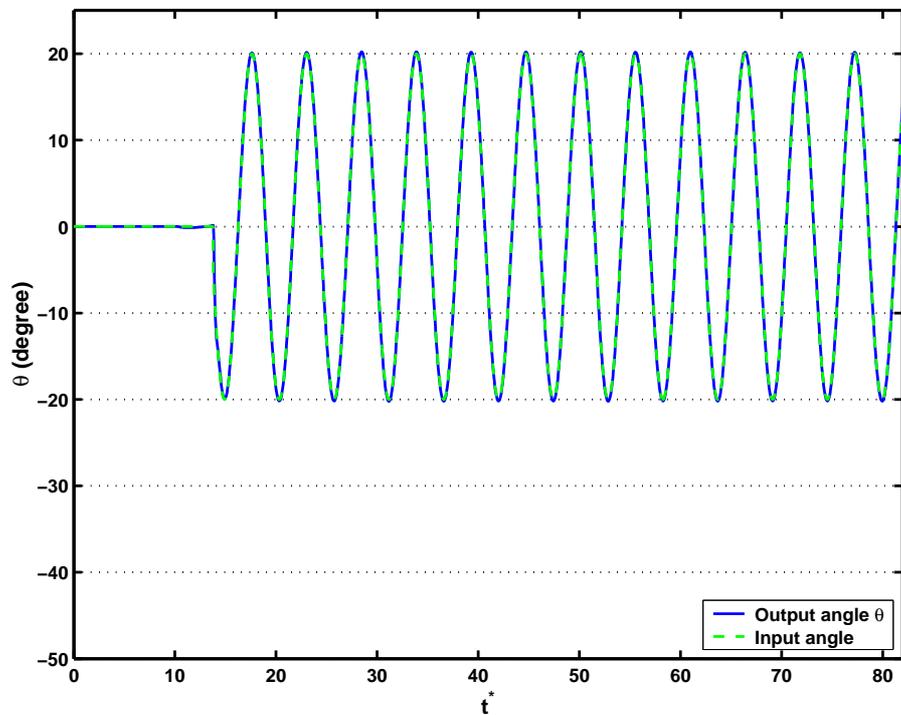
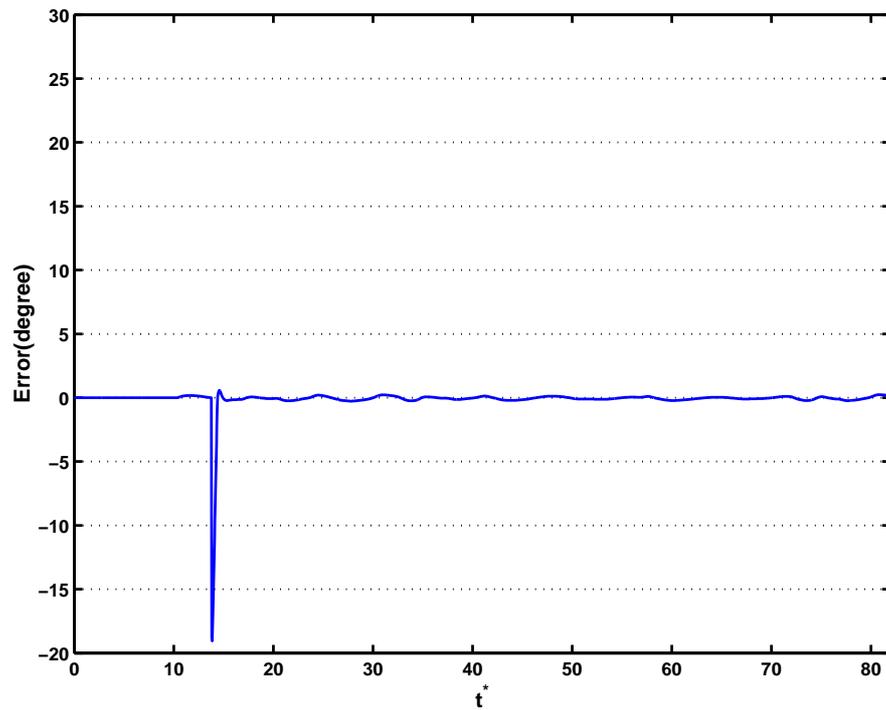
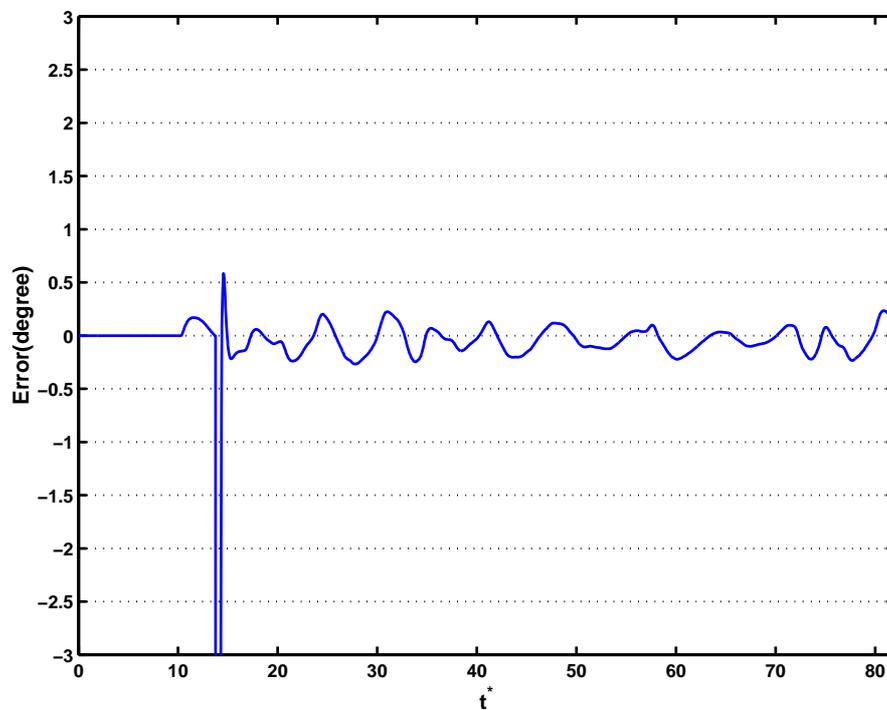


Figure 5.56: Time plot of the angle evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = 1.5 S_s$ and oscillating input

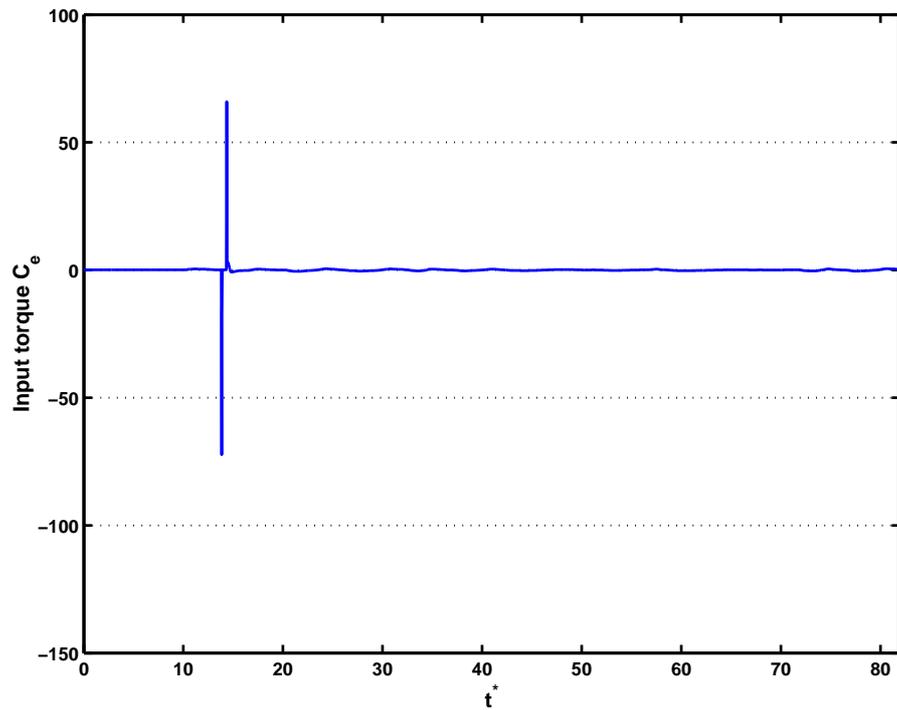


(a) Angular error

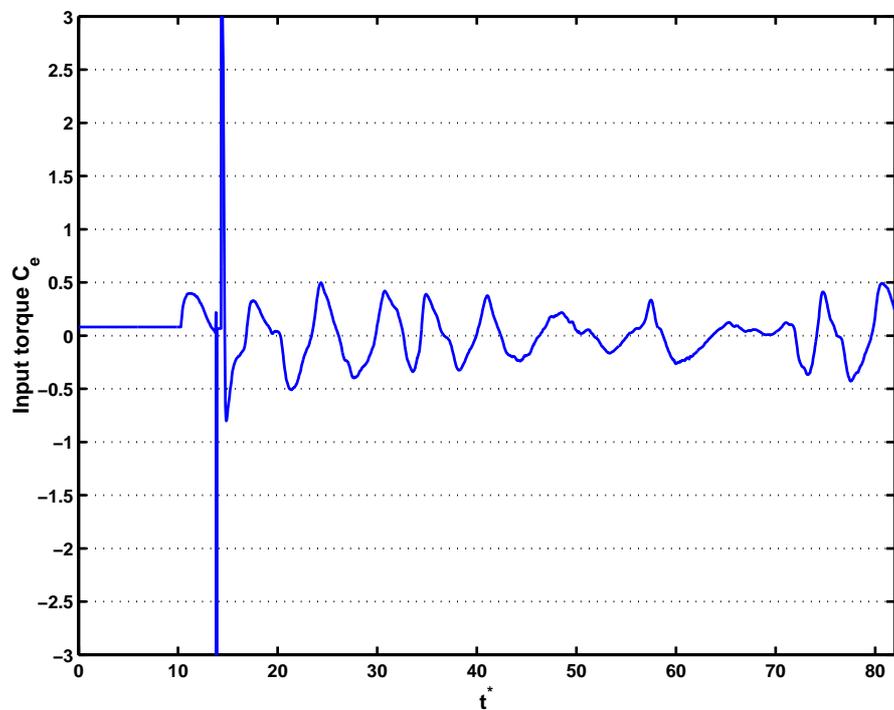


(b) Expanded view of the angular error

Figure 5.57: Time plot of the error evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = S_s$ and oscillating input

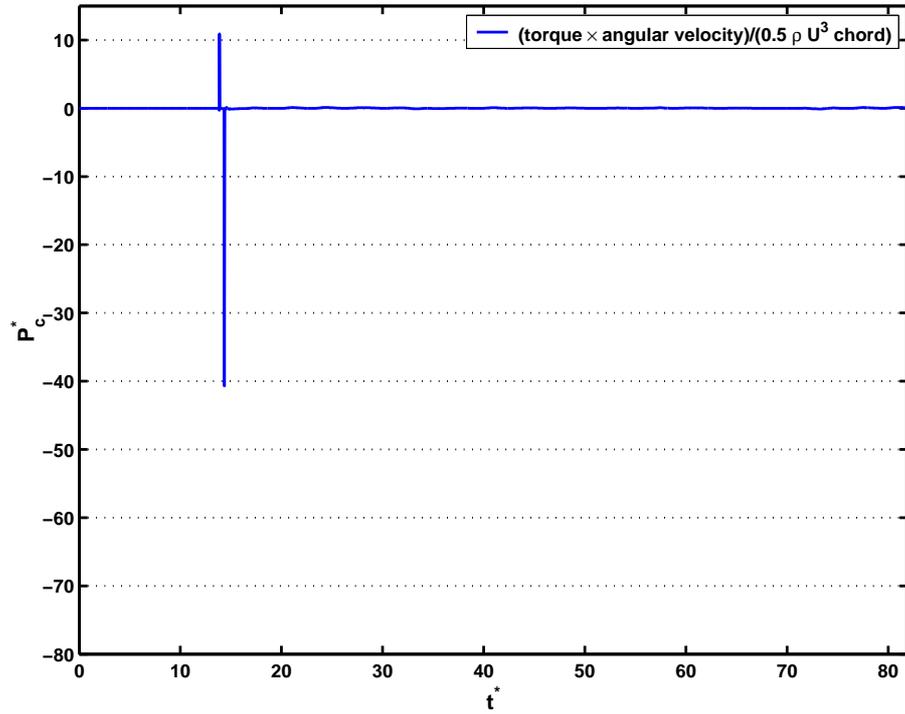


(a) Controller torque

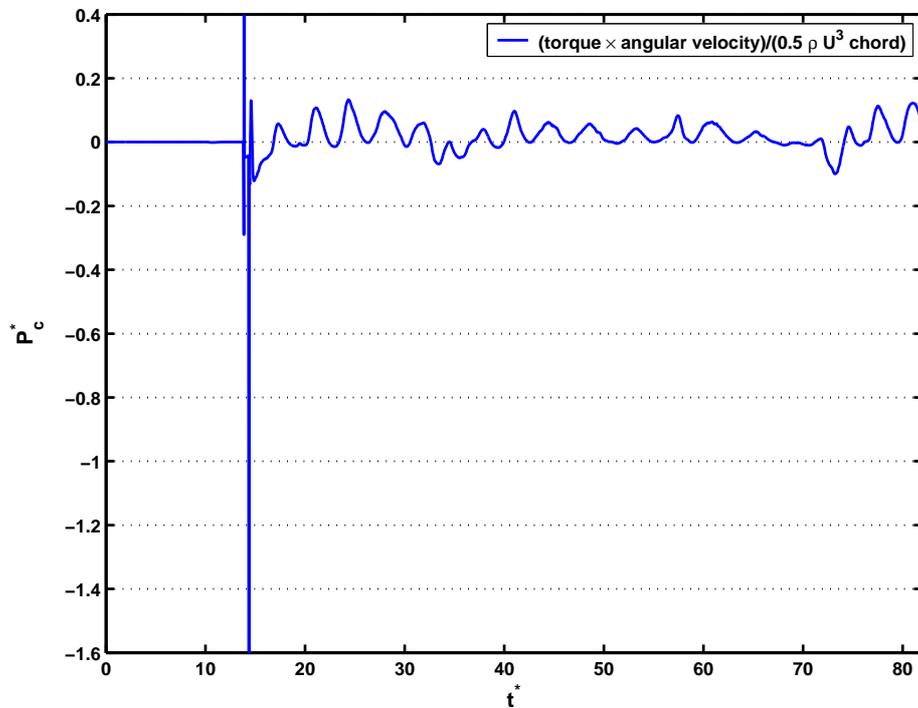


(b) Expanded view of controller torque history

Figure 5.58: Time plot of the controller torque input for the flow/plate system motion controlled by the fuzzy logic controller with $f_n^* = S_s$ and oscillating input

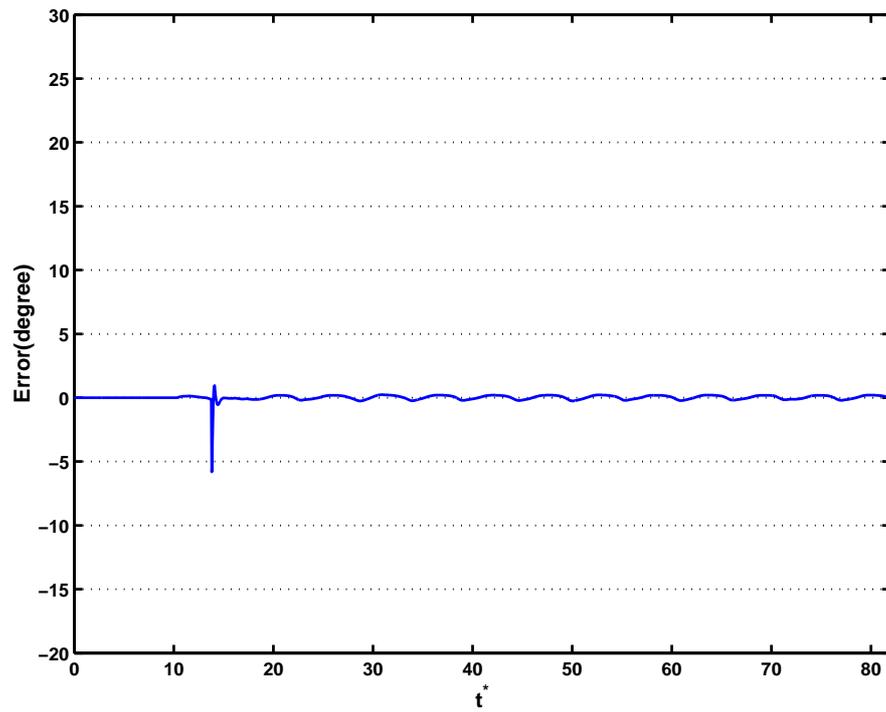


(a) Controller power

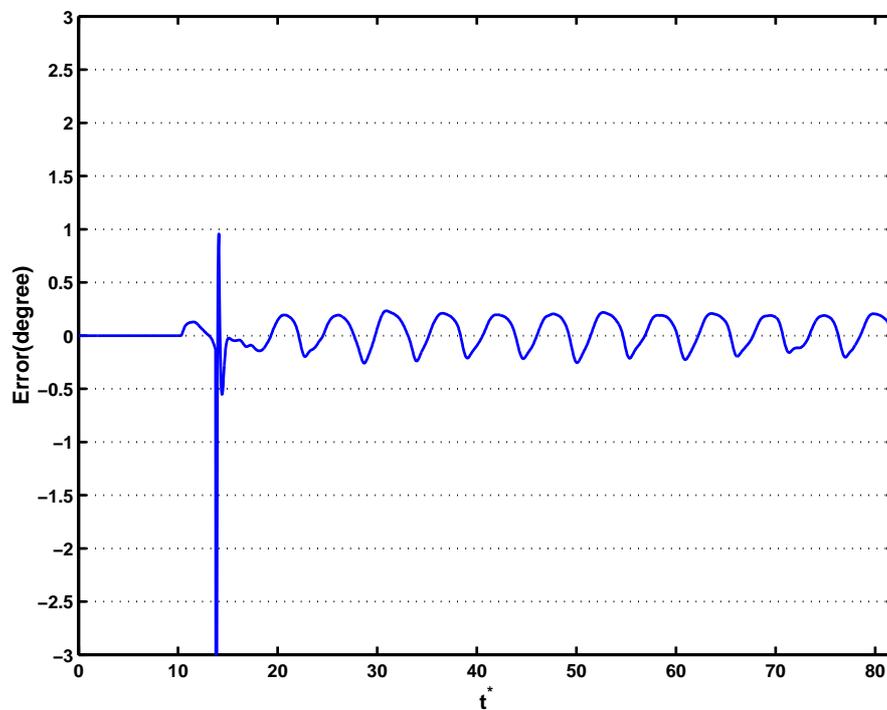


(b) Expanded view of controller power history

Figure 5.59: Time plot of the power signal evolution in time for the fuzzy logic controller with $f_n^* = S_s$ and oscillating input. The power is divided by $(1/2)\rho U_\infty^3 L$.

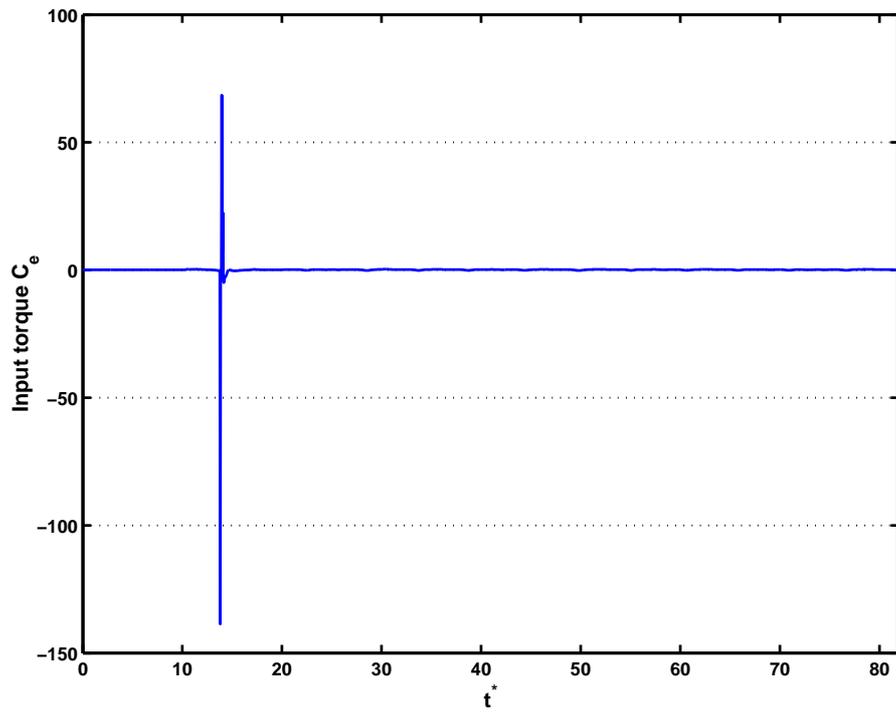


(a) Angular error

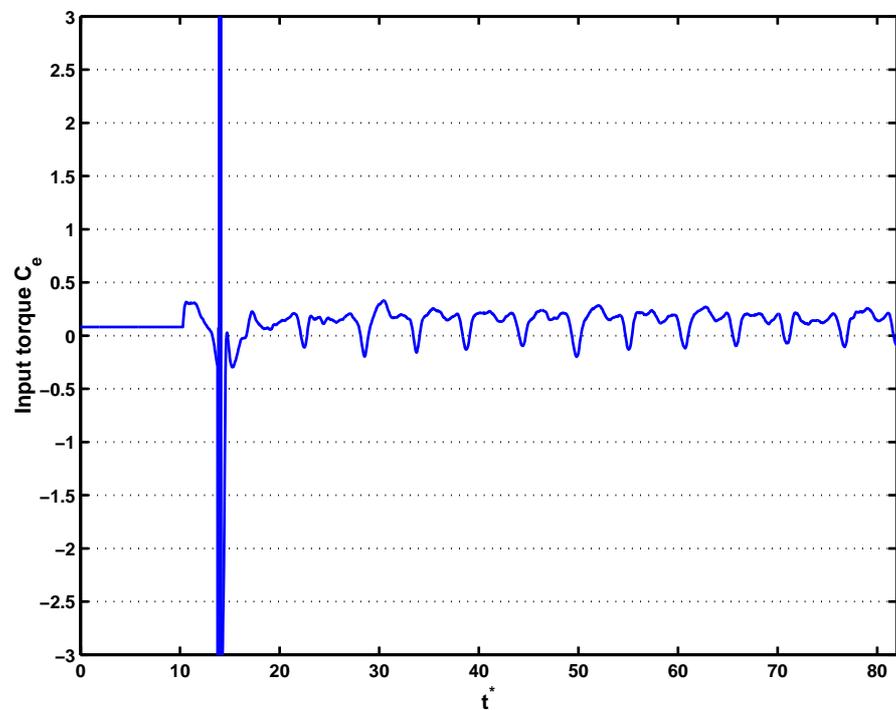


(b) Expanded view of the angular error

Figure 5.60: Time plot of the error evolution for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = 1.5 S_s$ and oscillating input

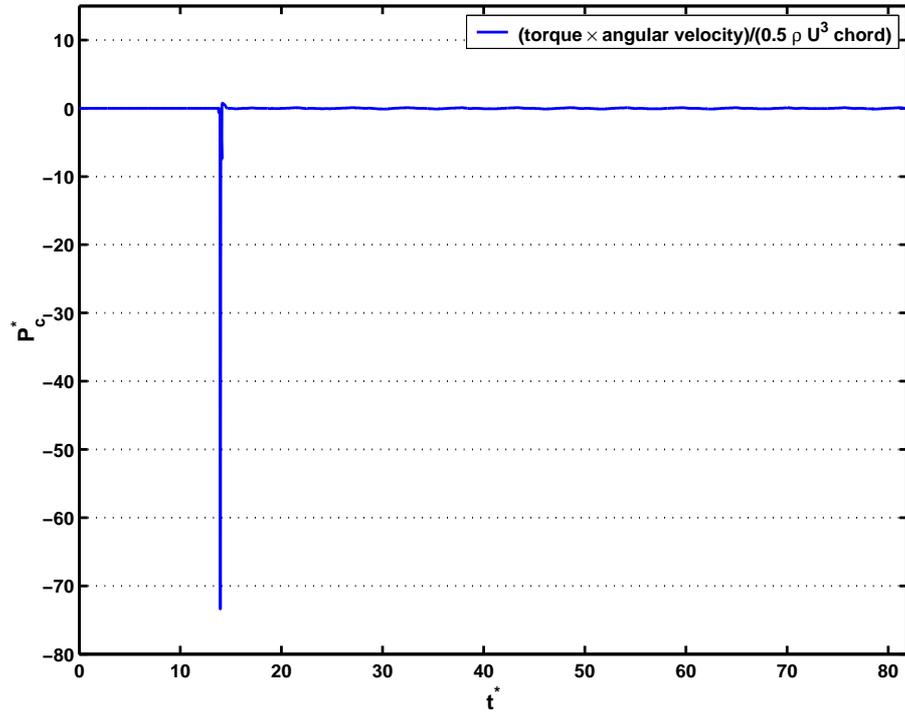


(a) Controller torque

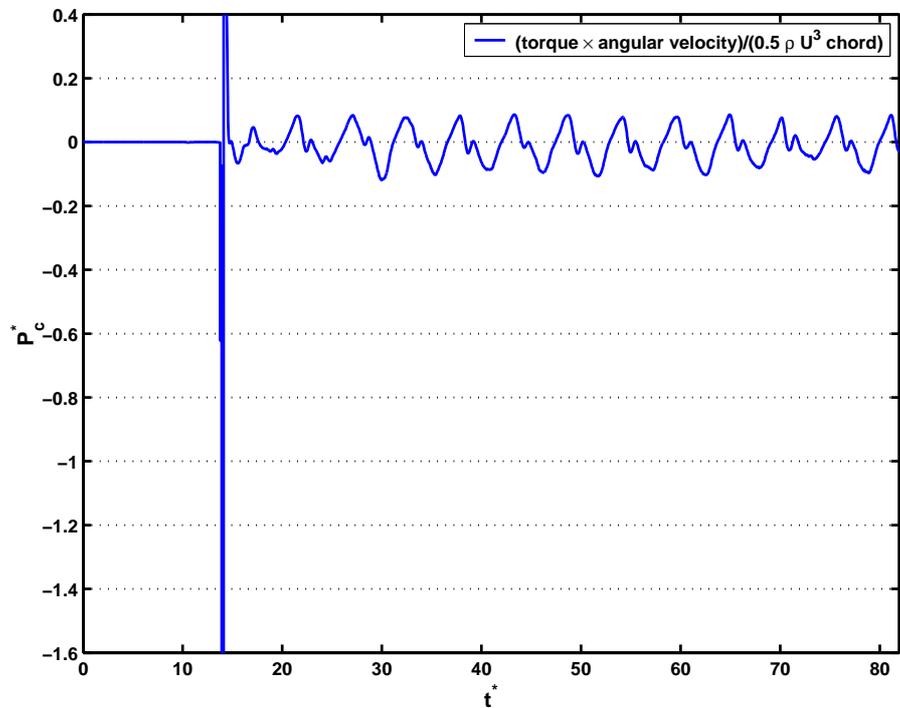


(b) Expanded view of controller torque history

Figure 5.61: Time plot of the controller torque input for the flow/plate system motion controlled by the fuzzy-logic controller with $f_n^* = 1.5 S_s$ and oscillating input



(a) Controller power



(b) Expanded view of controller power history

Figure 5.62: Time plot of the power signal evolution in time for the fuzzy logic controller with $f_n^* = 1.5 S_s$ and oscillating input. The power is divided by $(1/2)\rho U_\infty^3 L$.

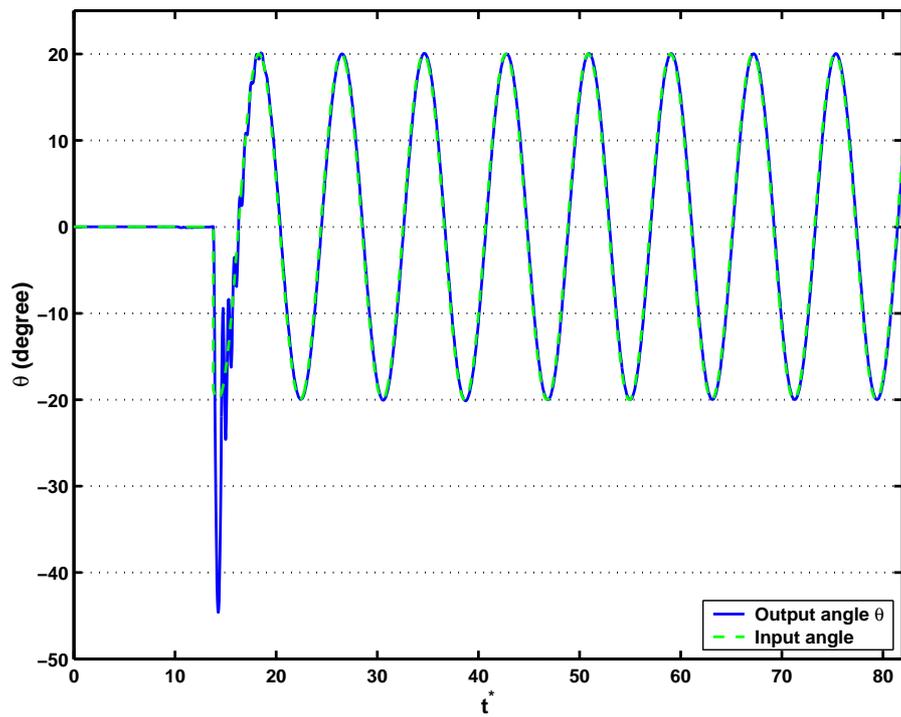


Figure 5.63: Time plot of the angle evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = S_s$ and oscillating input

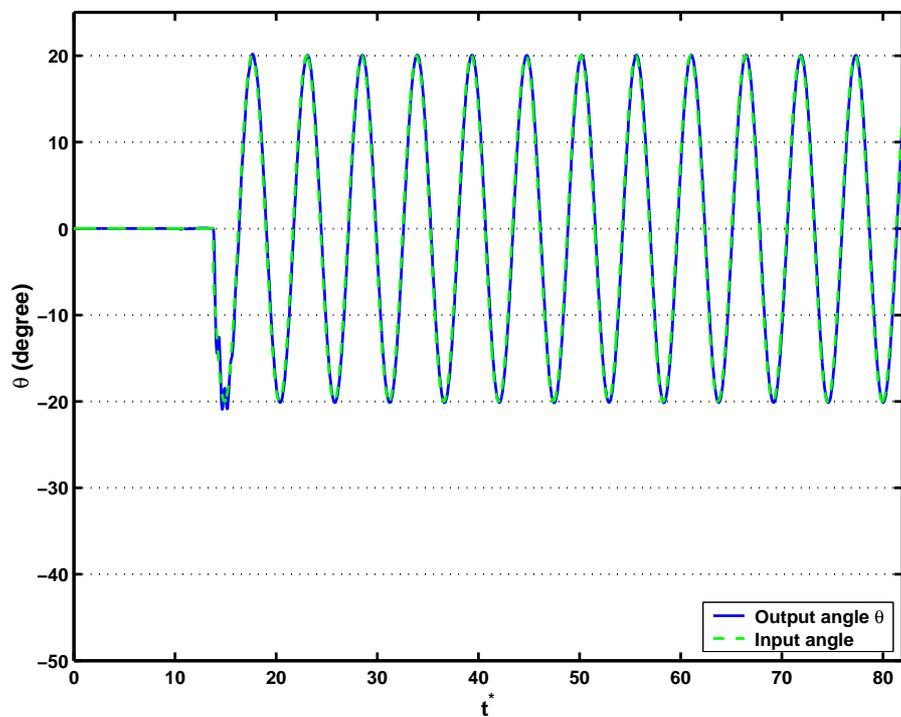
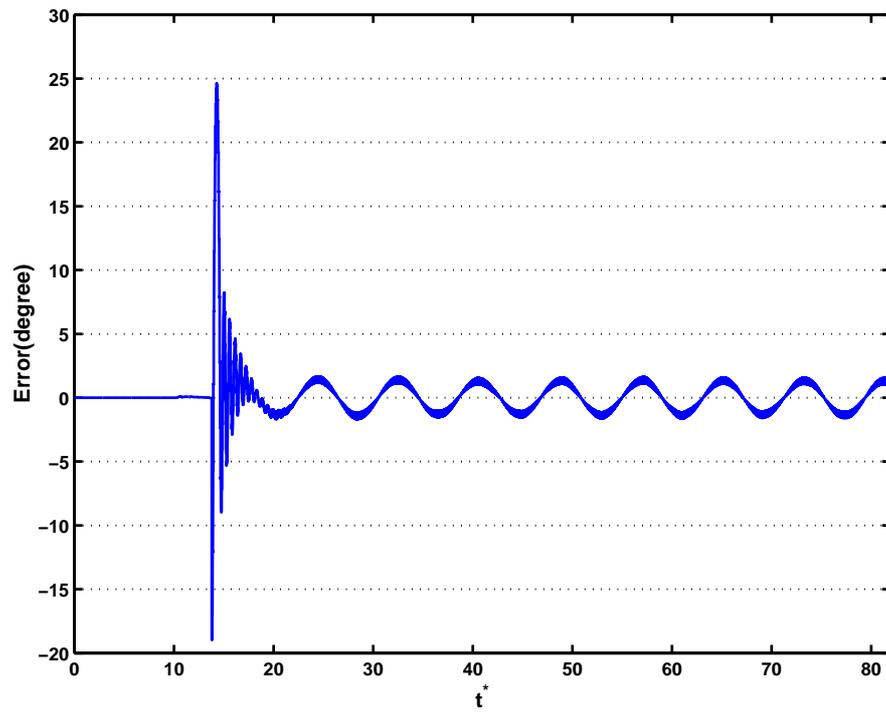
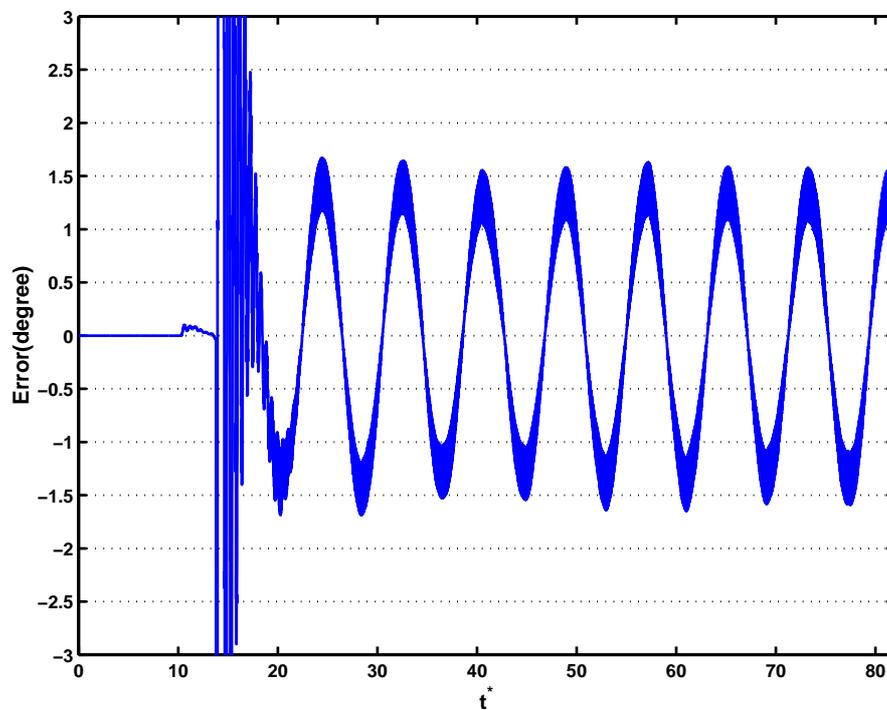


Figure 5.64: Time plot of the angle evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = 1.5 S_s$ and oscillating input

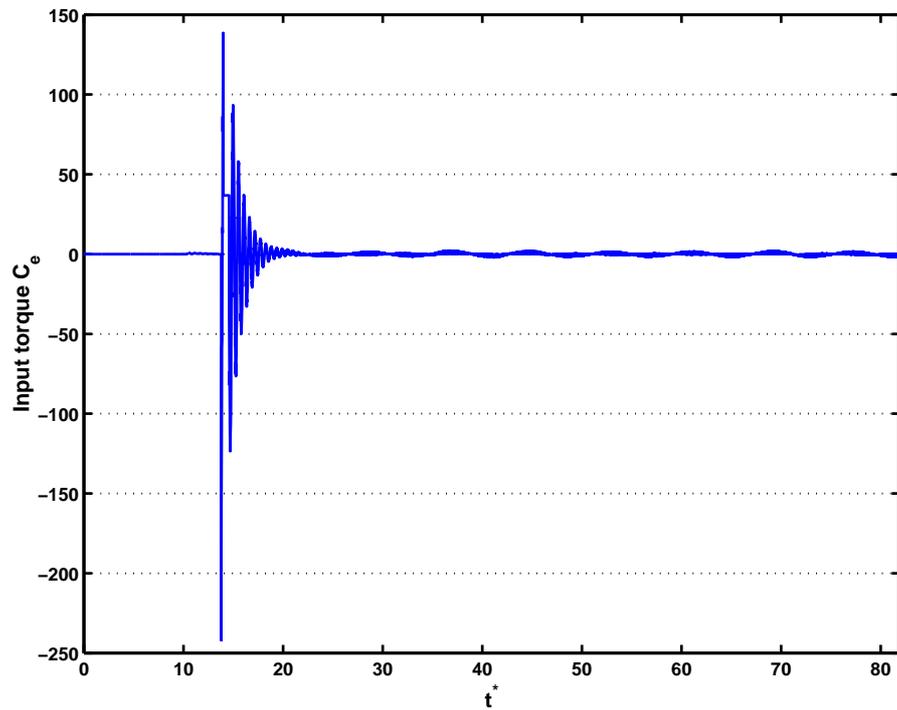


(a) Angular error

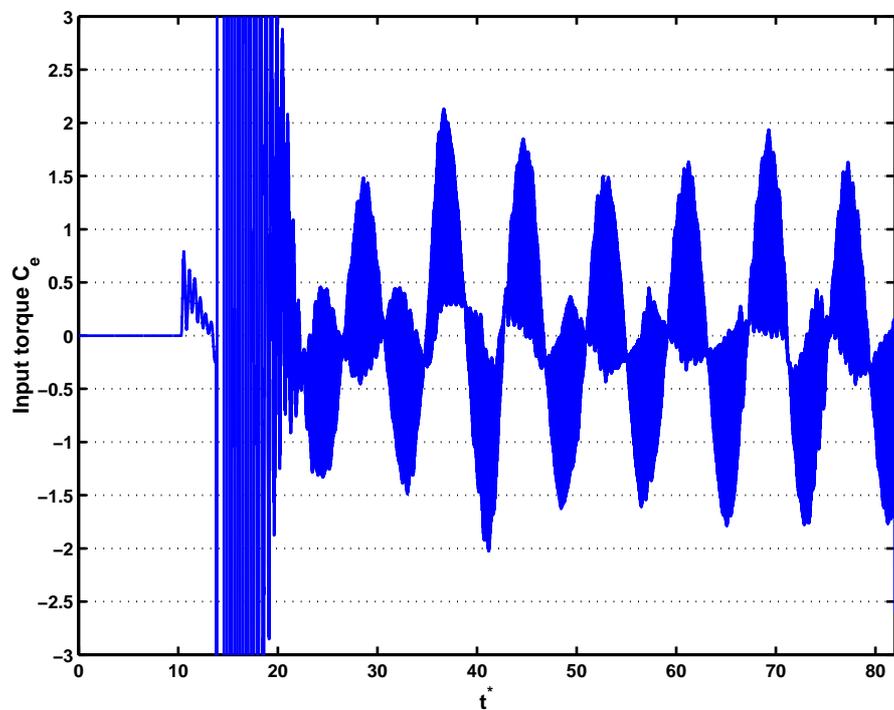


(b) Expanded view of the angular error

Figure 5.65: Time plot of the error evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = S_s$ and oscillating input

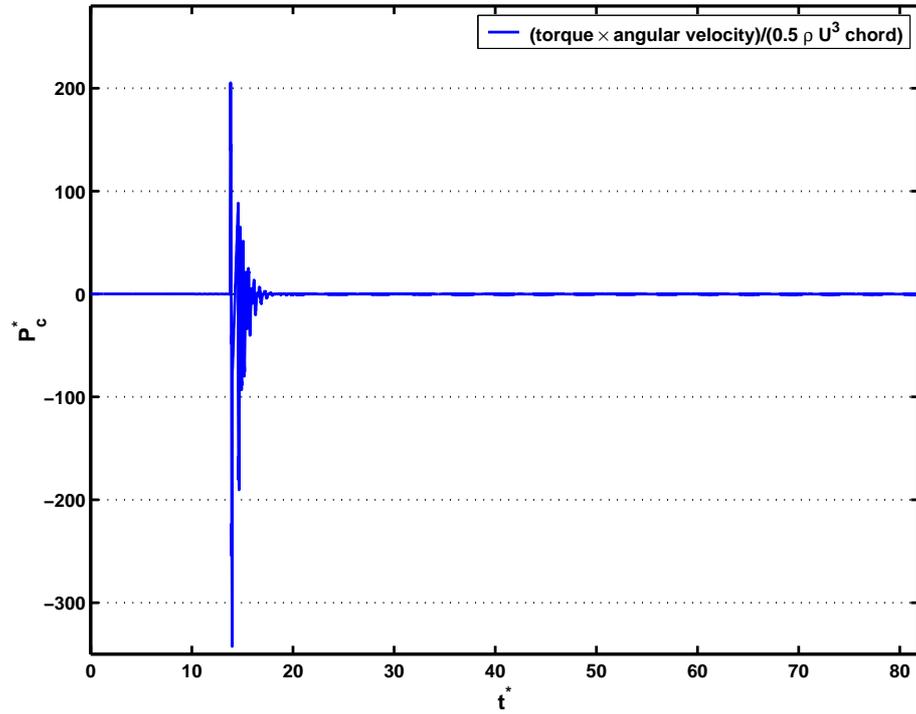


(a) Controller torque

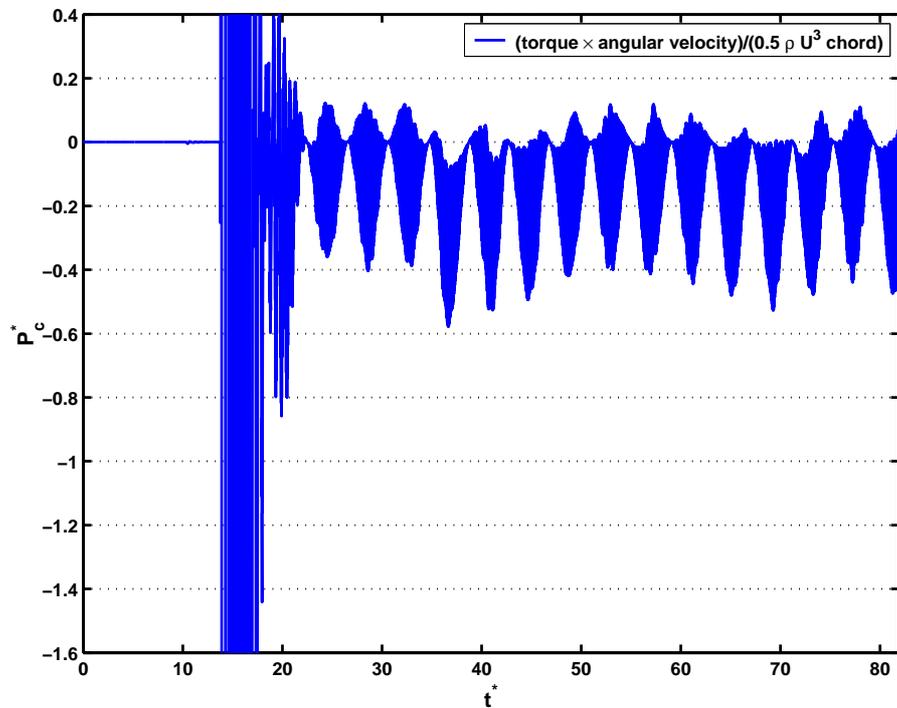


(b) Expanded view of controller torque history

Figure 5.66: Time plot of the torque input for the flow/plate system motion controlled by the optimal controller with $f_n^* = S_s$ and oscillating input

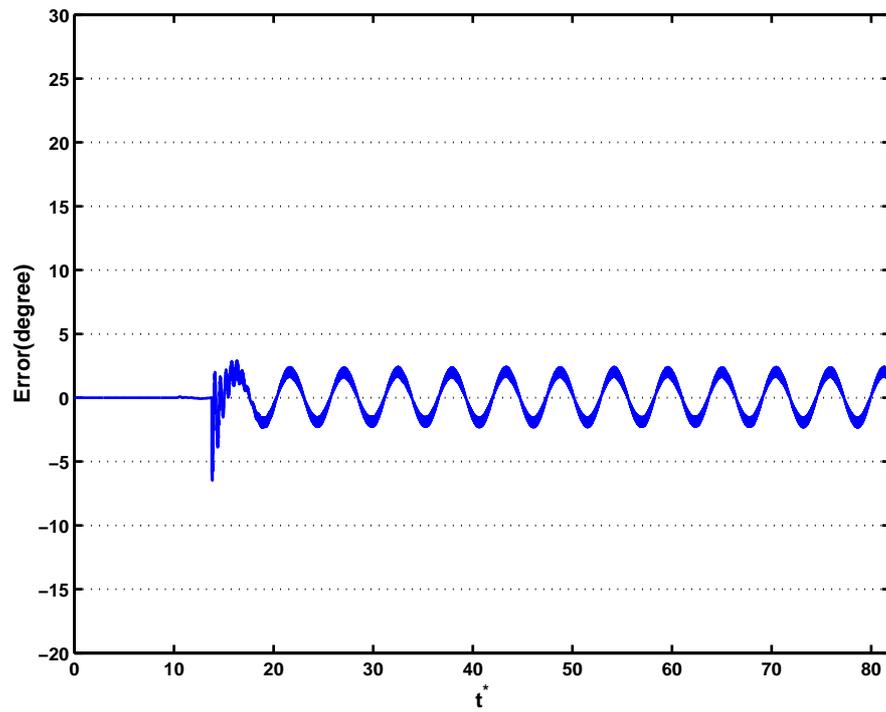


(a) Controller power

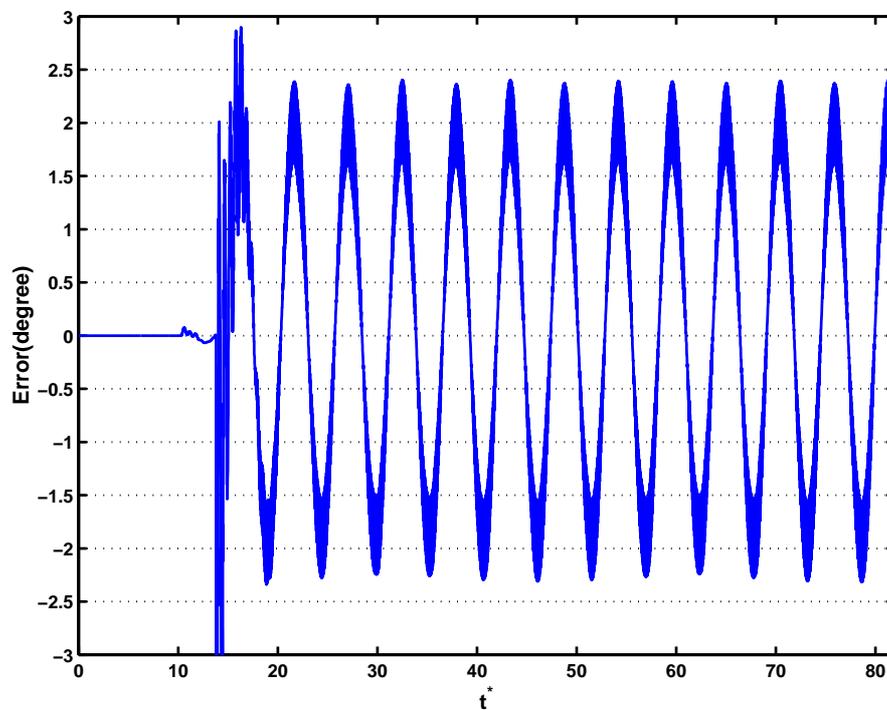


(b) Expanded view of controller power history

Figure 5.67: Time plot of the power signal evolution in time for the optimal controller with $f_n^* = S_s$ and oscillating input. The power is divided by $(1/2)\rho U_\infty^3 L$.

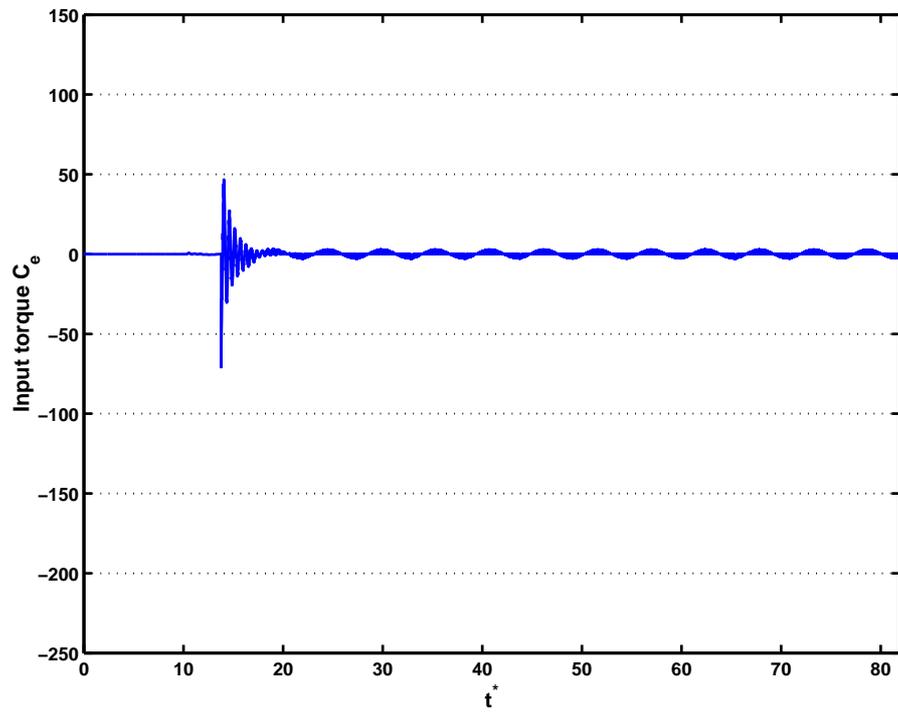


(a) Angular error

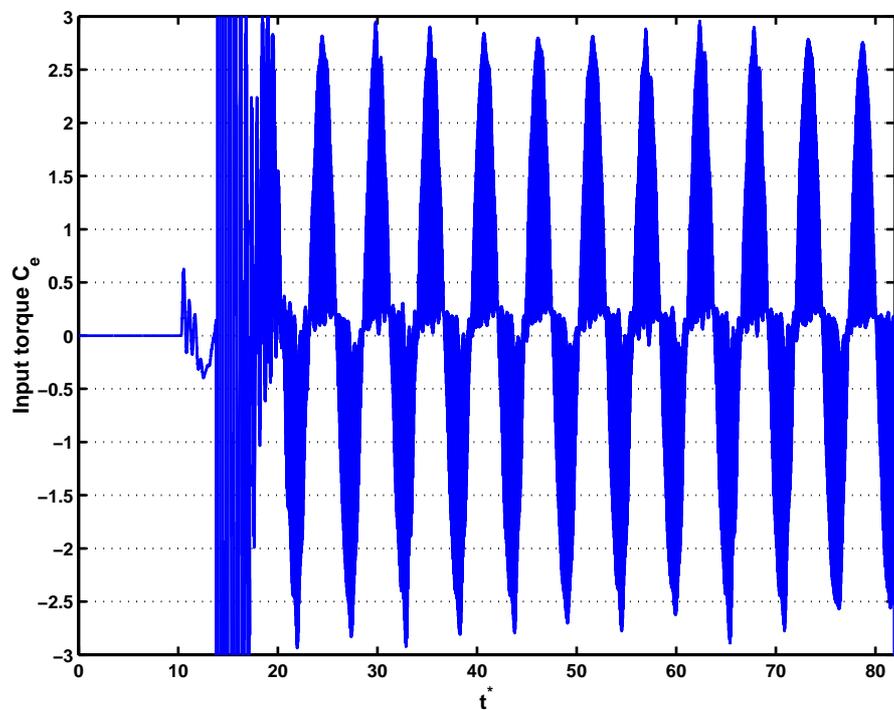


(b) Expanded view of the angular error

Figure 5.68: Time plot of the error evolution for the flow/plate system motion controlled by the optimal controller with $f_n^* = 1.5 S_s$ and oscillating input

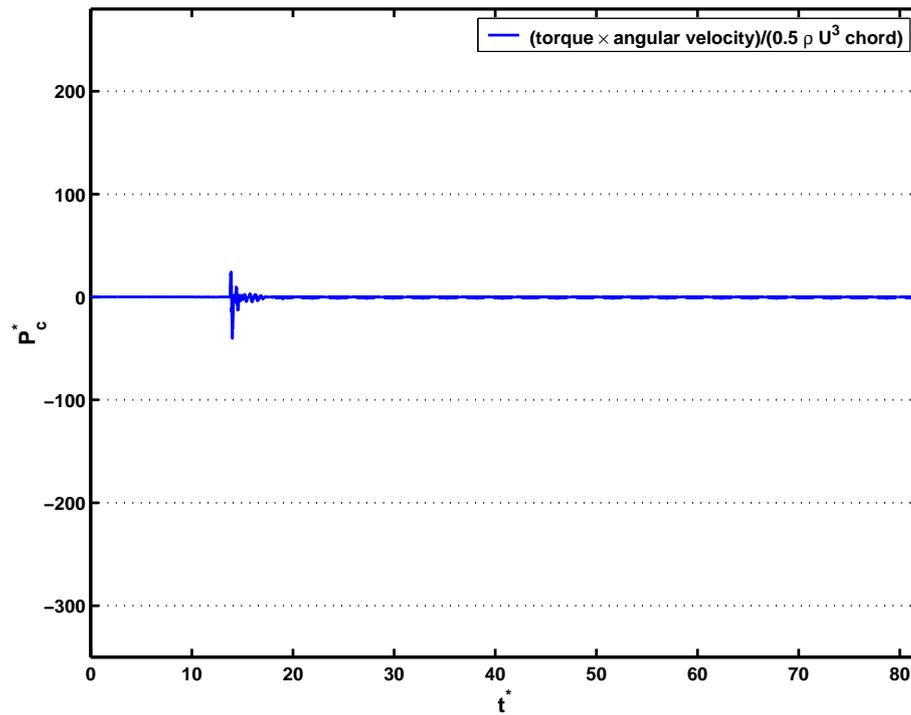


(a) Controller torque

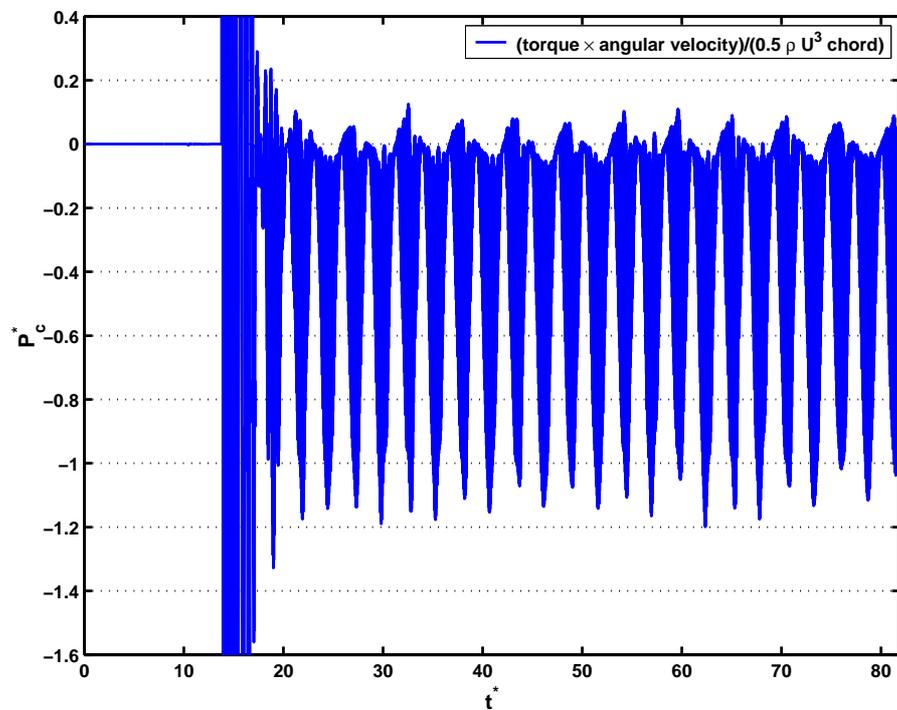


(b) Expanded view of controller torque history

Figure 5.69: Time plot of the torque input for the flow/plate system motion controlled by the optimal controller with $f_n^* = 1.5 S_s$ and oscillating input



(a) Controller power



(b) Expanded view of controller power history

Figure 5.70: Time plot of the power signal evolution in time for the optimal controller with $f_n^* = 1.5 S_s$ and oscillating input. The power is divided by $(1/2)\rho U_\infty^3 L$.

5.5 Summary

First in sections, 5.2.4, and 5.2.5, it was shown that the fuzzy logic was more precise during the motion although with a slightly higher response time by testing the optimal controller (actually closer to a Takagi-Sugeno controller) and a fuzzy controller against the steady angle flow model defined in section 4.3.5. It was also shown that past the motion, both controller were comparable albeit with a slightly more precise optimal controller. The fuzzy logic controller compares less favorably to the classical controller (section 5.2.3) but it does not require a control design per angle of attack and ρ_b . Overall it has shown a good robustness in this first part.

These results have been confirmed when coupled with Spalart code flow simulation in section 5.3. First the plate stabilization test in section 5.4.1 has shown that the fuzzy logic and optimal controllers have similar stabilization properties although the fuzzy has slightly bigger error range as well as a permanent error. This error is conjectured to be coming from the global system nonlinearities due to numerical parameter, in particular the flow simulation timestep. Observe that the timestep also affects the optimal controller effectiveness.

Finally, in sections 5.4.2.1 and 5.4.2.2, the fuzzy logic controller has shown that for a controlled motion it was more robust and overall more precise throughout the motion, even though past the motion there remain a constant error similar to the one seen for the stabilized plate case. Note however that after removing this error, the level of error is similar between the two controllers.

Chapter 6

Conclusion and Discussion

Here a study is made about the ability of a controller to command the position of a spring-damped thin rigid ellipse on a central pivot (a mechanical system) in a $2D$ inviscid transverse flow (a flow system), with the two systems able to interact through a software interface. A blob-vortex method was used in order to model the flow. A thin $20 : 1$ ellipse has been chosen as an analogy to a flat plate, allowing a clean well-defined flow separation. This coupled simulation has permitted an investigation of the spring damped plate/flow dynamics over a restricted range of parameters. A fuzzy-logic controller was used and has demonstrated its robustness compared to a controller similar to a gain-scheduling controller. This control successfully maintains the angular position of the plate from which vortices are shed in the $2D$ flow simulation. Cases where the control is applied to the plate both suddenly released from rest as well as impulsively moved to another angle of position were considered.

The numerical method was initially a scheme based on the discrete introduction of vorticity through discrete vortices at the points of separation of the flow at both edges of the ellipse. This was based on a vortex method scheme using a complex potential solution. The strength of the nascent vortices is found through an approximation of the shear layer near the separation, and the position of the nascent discrete vortices is determined afterwards using Kutta's condition.

Improvement for this scheme was then sought through the use of blob vortices as flow elements in order to smooth the velocity field, and thereby address one of the main shortfall of the discrete vortex method. The technique is thus based on a model of unbounded flow with inviscid fluid, and the use of potential theory to model the boundaries. Blob vortices are used to model the boundaries in a manner similar to a panel

element, and obtained similar results to the exact potential solution for the steady flow around a fixed ellipse. The aim was to obtain a method that was fast but nevertheless accurate enough to model the flow dynamics.

This scheme produced results in qualitative agreement with experiments in the case of the fixed ellipse at a high angle of incidence in an impulsively started flow, the first test problem. Because the incidence angle is high, the flow is in a dynamic stall state. However, although the code is able to model the dynamics of the flow in the case of a fixed body, it produces noisy aerodynamic forces and moment coefficients. Moreover, it has proved ill-suited to use with a moving thin ellipse. These two limitations are mainly due to the resolution of the equations linked to the nascent vortex strength determination, and singularities in the velocity field around the edges as the ellipse rotates back and forth in the case of oscillations or as blob vortices approach the ellipse edges. The strength determination disturbances also impede the nascent-vortex position determination. Filtering the strength does not lead to satisfying results as it damps the vorticity fluctuation, and leads to shed vortices with too strong vorticity.

As an alternative to this scheme, a blob vortex method developed by Spalart has been used and which was modified to take into account for a moving body. As with the previous method, it is based on a model of unbounded inviscid flow, and the use of potential theory to model the boundaries. With this code the nascent vortices are now the boundary vortices shed after each timestep into the flow. The higher number of vortices, the constant vortices distribution along the wall and the absence of position determination algorithm enable to remove the Sarpkaya-like code limitations. A vortex merging algorithm then permits to limit the computing power required. The main modifications to the Spalart code for the moving body code concern the boundary condition, the effect of the body motion on the velocity field, and the aerodynamic force and moment computation. This method is in fact quite fast, but requires good blob vortices overlap to converge correctly. It is also sensitive to timestep variations. Its main downside is that, although the simulation never blows up, there is no criteria that enables the user to choose a set of ad hoc parameters which ensure the correct solution.

In the first test problem, although the Spalart code simulation is able to reproduce the vortex street dynamics, it is only in qualitative agreement with experiment after the first vortices shedding. The lack of viscous vorticity diffusion and the intrinsic $3D$

nature of this kind of flow means the shed vorticity is overestimated. Moreover, with this set of parameters, the overlap is insufficient to preclude any quantitative errors as the simulation demands a larger number of vortices to converge than the computing resources allowed.

The second test problem is an ellipse with a constant angular velocity in a uniform transverse flow, for which the agreement is much better quantitatively and the flow dynamics is still respected. Thus the simulation is better suited to massively separated flow. However, the lack of experimental results in this case impedes the full determination of the simulation validity.

Generally if the flow geometry and dynamics are preserved, the aerodynamic forces and moment coefficient are overestimated, while the Strouhal number is underestimated. These flaws are mainly due to the absence, in both test problems, of vorticity diffusion inherent to the $3D$ real flow, the lack of an explicit viscous diffusion mechanism, as well as an insufficient blob vortices overlap. Finally, the merging of the blob vortices introduces some velocity field perturbation in the far field. These effects are more visible for long term simulation where the blob vortices have been convected by the uniform freeflow for longer times.

In a third test problem, an oscillating cylinder was implemented in a uniform freeflow. The cylinder is allowed to oscillate transversely to the flow. The simulation predicts correctly the oscillation-shedding lock-in frequency, although its range is too narrow and there is less sharp increase in the lock-in range than in previous literature. In this case, the problem lies in the phase between the cylinder motion and the flow dynamics, which is incorrectly predicted.

Then, it then proceeded with the study of characteristics of a spring damped ellipse oscillating about a central pivot in a fluid. Compared to the fluid (air), a large body density was chosen, in order to keep the ellipse angular oscillations amplitude within a given range, to aid the controller design.

The first case study examined was that of an ellipse released in an initially quiescent fluid (i.e. no free flow) and no damping of the torsional spring. The ellipse zero deflection position of the spring is set at a slight initial angle, so that it oscillates after

it is released. The flow dynamics has been shown to introduce some additional damping through the creation at the ellipse edges of vortices of circulation adverse to the ellipse motion as the body oscillates clockwise and counterclockwise. This also affects the oscillation frequency as it affects the ellipse angular velocity. This effect is encountered even when a high spring damping is used.

Therefore, the flow simulation effectively introduces an added mass which diminishes the angular oscillation frequency similar to a real added mass due to flow unsteadiness. There is also an additional damping as a result of the effective viscosity of the simulation as well as the fluid inertia induced damping. However, the simulation added some numerical noise affecting the added mass and damping evaluation through the lack of blob vortices overlap, the blob vortices merging mechanism (used to reduce computation time), as well as the simulation lack of explicit viscous diffusion mechanism. With adverse parameters, this can lead to the creation of one eddy with strong circulation due to blob vortices aggregation. Experiments are again necessary to better assess to which extent the simulation influences the resulting angular oscillation damping and frequency especially concerning the influence of viscosity on the system.

In another test case, an analysis was made about the influence of the reduced damping of the spring-damped ellipse on the behavior of the coupled spring damped ellipse/flow system. The ellipse was set in a transverse uniform freeflow without a initial spring angle. The ellipse is kept fixed at a high incidence angle previously to this time in order to initiate the flow separation before letting the ellipse pivot.

It has emerged that the reduced damping of the spring damped ellipse system is not a critical parameter for the dynamic behaviour of the combined flow/plate system. Indeed, it is expected to broaden the lock-in range as it is lowered but not to fundamentally change the different modes occurring, nor the mean reduced frequency of the spring-damped system where the oscillation-shedding lock-in takes place.

In a final test case, the coupled ellipse/flow system characteristics was examined by using a similar setup as in the previous case but with a fixed reduced damping. The damping was set at a value high enough so as to expect less interference from the numerical simulation, but low enough not to impede too much the coupling between the mechanical and the flow system. The mechanical system frequency was then varied in

order to study the coupled flow/structure dynamics.

Generally the rotating-ellipse/flow system exhibits a narrow lock-in band near the flow shedding frequency present at the motion start. Before this reduced mechanical frequency lock-in range, the mechanical system is driven by the flow dynamics, that is the two systems behave as if they were independent of each other, and the angular oscillations have little effects on the flow.

As can be expected, in the lock-in band, the ellipse and the flow are coupled near the mechanical system resonance frequency so that the ellipse oscillations are greatly increased by the additional torque provided by the flow. However, if there still remains a vortex shedding, the flow dynamics is little changed by the ellipse oscillations. Typically, at a given timestep a primary nascent vortex is formed at one ellipse tip, and the plate is then rotated toward this nascent vortex. As the plate gains angular velocity, it creates another secondary nascent vortex at the opposite ellipse tip which is fed by the resulting ellipse rotation. The ellipse angular velocity also results in the weakening of the primary nascent vortex. Afterwards, the primary vortex is shed into the flow, and as the ellipse rotation slows down the secondary vortex now becomes the primary vortex and is fed by the shear layer corresponding to the ellipse tip from which it is shed, and another shedding cycle begins.

This phenomenon has two effects. First the vortex shedding is now faster due to the creation of this secondary nascent vortex, and second the aerodynamic moment coefficient is lowered by the secondary nascent vortex on the opposite edge of the primary shed vortex, and by the diminution of the vorticity feeding the primary nascent vortex. The angular oscillations are thus able to raise the flow shedding frequency through this secondary nascent vortex creation. There is also a mechanism limiting the shedding frequency as if it is too high it lowers the aerodynamic moment coefficient considerably through the previously described interaction mechanism. This explains part of the vortex shedding synchronization mechanism; it means that for lower prescribed mechanical frequency (compared to the obvious resonance point), the shedding frequency can be raised, and for higher mechanical frequency the flow shedding frequency can be lowered. Note that this mechanism is very dependent on the angular velocity attained in the early part of the simulation.

Generally, due to the rotating-ellipse/flow interaction the aerodynamic moment coefficient range is lowered, while it increases the aerodynamic drag and lift coefficient value range. The mechanism of diminution of the moment has already been explained. It seems that the drag and lift coefficients are raised through the presence of the secondary nascent vortex as well as the circulation around the body induced by the angular rotation.

Concerning the controller implementation, first the flow was characterized by implementing a rotating-ellipse/flow model based on the moment coefficient from the Spalart simulation of the fixed ellipse for different incidence angles in a uniform freeflow. For the flow characteristics model, the incidence angle was taken as the input, and the aerodynamic moment as the output. For each incidence angle, a coupled spring damped ellipse/flow model was then constructed based on the linear moment coefficient output model, as well as the linear model associated with the angular pivoting-ellipse. The two models are coupled as a closed loop system. Then the models associated with the different incidence angles are interpolated according to the incidence angle value. The different models are interpolated not only in output amplitude but also in frequency. This enable to have the equivalent of a flow model with an aerodynamic moment coefficient mainly comprised of one harmonic, which conforms to the experimental moment coefficient evolution for a plate at fixed incidence angle. The input of the coupled spring damped ellipse/flow model is then an external torque while the output is the ellipse angle position.

This system characterization is intended as a simple model for a pivoting-ellipse whose angular position is stabilized in a transverse uniform freeflow. It is thus not an accurate model for a moving ellipse in a flow. This model characterization has permitted the design of two different kinds of controller. The first one is based on fuzzy logic, while the second is more classical and is similar to Takagi-Sugeno models (use of multiple linear models and weighed means to determine the output). Both controllers have been tested with the coupled spring-damped ellipse/flow model and have shown that they are able to stabilize the ellipse position around a prescribed position, or to apply a step input from one angle position to another.

The fuzzy-logic controller design has been very easy and has shown good robustness compared to the gain-scheduling controller. It is also independent of the spring-damped

ellipse mechanical frequency. Its main downside is a high computing cost and the fact that there is no way other than trial and error to measure the robustness and stability of this kind of control without explicitly taking into account the coupled plate/flow model.

The second controller, similar to Takagi-Sugeno models, is based on optimal control theory. Like the coupled spring-damped plate/flow model, each optimal controller output has been interpolated regarding the ellipse incidence angle. In contrast to the previous controller, the stability is guaranteed when applied to a linear system. Moreover, its robustness has been increased through the use of a Kalman filter. However, the interpolation makes it difficult to predict the ideal stabilization. Note that the Takagi-Sugeno design is slightly different of a gain scheduling method in that it does not involve changing parameters from a compensator with a fixed transfer function, but rather the interpolation between different controller output. It has also been noted in the literature that this kind of design has lead to results less satisfying than when using gain scheduling. This is because such an implementation neglects the variations in frequency or the variations of the natural mode of the controlled system.

Concerning the coupling, a discrete simulation was used for the fuzzy-logic controller, but due to a Simulink constraint a continuous integration was chosen for the optimal controller. In the latter case, this continuous implementation (as opposed to the discrete output provided by the flow simulation) has created some additional problems by creating artificially delays between the torque input and the torque taken into account by the flow simulation, thus creating nonlinearities. The delays are linked to Simulink and flow simulation dephasing due to a difference in the differential equation integration scheme (variable timestep for Simulink, and discrete timestep for the flow simulation). On average, the delay is estimated to be inferior to the flow simulation timestep but is difficult to asses exactly. This problem could easily be overcome by simply decreasing the flow simulation timestep. Under these conditions, the optimal controller was able to successfully control the plate as if the flow simulation was continuous. The fuzzy-logic controller was less affected due with such problems, even when using the same continuous Simulink implementation as the optimal controller, showing the controller robustness.

As the controllers are added to the coupled ellipse/flow simulation, both have shown their ability to stabilize the ellipse position. And although the optimal controller has

shown better precision than the fuzzy logic controller when stabilizing the plate, the fuzzy logic controller has proven more versatile notably for the stabilization of an ellipse left oscillating with a spring damped ellipse frequency fixed near resonance. Furthermore, its implementation is independent of the mechanical system frequency. However, its performance is dependent on the fuzzy set definitions. As the sets are not independent one from another and, as stated earlier, an optimal definition of the fuzzy sets can only be found empirically. Conversely, the Takagi-Sugeno controller is more difficult of use as it must be redesigned for a given mechanical system frequency. It has performed better here as the fuzzy logic controller is not optimized.

When tested with an input similar to a smoothed step input, the fuzzy logic has proven that with no modification regarding f_n^* and despite a slight constant position error, it was able to adjust to the command signal faster than with a Takagi-Sugeno type control, while keeping the ellipse/flow system stable. Surprisingly, the fuzzy-logic controller has shown better performance for a varying signal. Note that in both cases, the controllers were able to command the plate position. Note also that step input smoothing was required because the fuzzy controller needs a finite angular velocity as an input due to the fuzzy set definition.

Overall, the fuzzy controller as well as the Takagi-Sugeno controller have shown good ability to stabilize this nonlinear system. However, in light of its implementation almost independent of the mechanical system characteristics and its robustness to a variety of situation, the fuzzy-logic controller might be preferred. Its main shortfall is the lack of theory to prove the robustness with given fuzzy sets. Note, though, that if one considers the angular accuracy as the main performance criterion, then the optimal controller would be preferred. Nevertheless, here the fuzzy logic control parameters have not been optimized. Various methods for adjusting and optimizing the fuzzy logic parameters exists based for example on method used usually on method used for robust control. Jenkins and Pasino [29] presents an introduction on such method, and Zhiqiang Gao et al [66] provide an example of more sophisticated method. However, such methods using a model of the system to control remove some of the advantages of the fuzzy logic controllers by using explicitly a model for the controller design similarly to robust control design.

For both the optimal controller and the fuzzy logic controller, remaining oscillations

can be noticed in the angle plot. The frequency of the remaining oscillations indicates that they are mainly related to the flow-induced moment on the plate, the plate f_n having little effect on the frequency of these oscillations. The remaining oscillations are larger when using the flow simulation than when using the identified flow model. They may be a side effect of the time discretization applied to the system (the optimal controller instability when using $dt^* = 0.05$ is also a symptom of such problems). Further analysis show that this does not seem to be due to an underestimation of the flow-induced moment applied to the plate, as in both cases the flow model and simulation C_m remain comparable in magnitude and frequency.

One could improve the fuzzy logic controller, for example by adding an input using the time-integration of the error. With this complementary input, the fuzzy logic controller is then similar to a PID controller regarding the inputs used for the calculation of the output. Under this configuration though, the fuzzy sets are much harder to parameter, and the full controller tends to lead to an overestimation of the output torque. Furthermore, it does not help to dampen the remaining oscillations.

As precised earlier, the remaining oscillations are mainly due to the flow. One can then remark that the identified model has only been identified over a given period of time which limits the representativity of the system in time. It also means that the model can introduce artificially some damping in the coupled spring damped plate/flow system. This is visible for example for the optimal controller where the decrease in oscillations amplitude is in fact related to a damping introduced by the identified flow model. Moreover, long term oscillation of period $T^* \simeq 300$ are also modulating the signal magnitude, which looks similar to a damping with a limited time window. Thus this can have an effect on the controller design especially in the case of the optimal controller as it uses flow and spring damped plate models.

Another additional remark comes from the study of the plate at steady angles of attack. When the plate is at a given angle of attack, one can see that the flow-induced moment is not behaving, when the angle is stable, similarly to a spring-damped oscillator with very light damping (positive or negative as the system is identified), but rather like a system with a narrow bandwidth and no damping. Indeed, the constant transverse flow is constantly bringing some energy to the system notably through the vortex street, and when the body moves the motions also brings some energy to the vortex street. The

consequence is that with the spring damped plate/flow system, when the plate is not moving, there are still some energy brought in, thus it is not only the plate motion that must be controlled but also the introduction of vorticity to the flow.

This shows the limitation of this approach, and emphasize the need to better take into account the flow. Indeed, implicitly the flow model is based on $C_m = f(\alpha, t)$ for the identification, but a more appropriate approach would be to use $C_m = f(\alpha, \dot{\alpha}, t)$. Then, one could use a C_m estimator to be taken into account explicitly by the fuzzy logic controller. Remark that the optimal controller design already use an estimator but based on $C_m = f(\alpha, t)$. This tantamounts to a rather phenomenological approach of the vorticity introduction due to the plate motion. Maybe more appropriate would be an approach based on flow control using a controller designed for minimizing a synthetic criterion using the vorticity introduced at both edges by the plate position and motion (to use explicitly $\alpha, \dot{\alpha}, t$) and using as well the position error. This would be especially useful for the fuzzy logic controller as in its current implementation it tends to minimize mainly the velocity error.

Finally, the flow simulation has shown to model qualitatively the flow dynamics for massively separated flow for a fixed and moving body, and the coupling between this flow simulation and the controller through the software interface has enabled to make a better assessment of the controllers performance in a situation qualitatively close to the real-life environment. However, there are some limitation due to the integration of both model in Simulink.

It would thus be necessary to make further experiments to assess the exact extent of the numerical flow simulation influence, and to improve the Simulink/flow simulation integration. An easy way would be to simply lower the Δt^* but this would require an adjustment in the simulation parameters. The current implementation is a compromise which enable to still take the simulation partly as a continuous function (deliver continuous output but at given timestep). Nonetheless, it would be possible to improve the Simulink/simulation link by a better adaptation of the simulation to Simulink standard function in order to enable a continuous implementation. It was not possible due to a lack of time as it asks for partly rethinking the simulation timestep iteration.

It would also be interesting to examine a broader range of mechanical parameters,

for example using body density over fluid density which are of the order of or less than unity in order to generalize results. It has been found from the literature that with such parameters, the behavior of a transversely oscillating cylinder in a uniform flow can be vastly changed. Additionally, it would be interesting to find an indicator to better characterize the coupled body/flow system behavior, as it has been shown that the effective oscillation damping and frequency can be changed depending on the flow interaction.

This study could be expanded by assessing the fuzzy logic controller performances for more challenging situation, such as helicopter blade position control, or even turbine blade position control with influence from multiple bodies separation. Another extension would be to implement one of the turbulence treatment which exists for vortex methods, or better still a use of fully 3D vortex method code. This last improvement would require much computing power, and may prove impractical. Nonetheless this could lead eventually to improvements in the body/system characterization in a qualitative and quantitative sense.

Appendix A

Streamline function of the flow elements

With the blob vortex method, there are two streamline functions, There are the streamline function related to the blob vortices, and the streamline function related to the freeflow. In all subsequent formula, the streamline function are computed at the point $\vec{P} = (x, y)$.

The streamline function for one blob vortex with a algebraic core is:

$$\psi(\vec{P}) = \frac{\Gamma}{4\pi} \ln(\|\vec{P} - \vec{x}_v\|^2 + \sigma^2), \quad (\text{A.1})$$

with a core radius σ , \vec{x}_v the vortex position.

The streamline function for a uniform stream at incidence α is:

$$\psi_\infty(\vec{P}) = U_\infty(x \cos(\alpha) + y \sin(\alpha)). \quad (\text{A.2})$$

The streamline function for a pure rotation at an angular velocity of Ω is:

$$\psi_\Omega(\vec{P}) = -\frac{\Omega}{2}(x^2 + y^2). \quad (\text{A.3})$$

Note that although there is an associated streamline function to the rotation, there is no potential.

Appendix B

Derivatives for vortex induced velocity

In order to find the solution of the Kutta condition ($dV = 0$), one must get the derivatives of the components of the velocity induced by a blob vortex in the stationary frame coordinates.

The velocity induced by a single blob vortex with an algebraic core function and unit strength is:

$$u_X = \frac{1}{\pi \sigma^2} \frac{r_Y}{2 \left(1 + \frac{r^2}{\sigma^2}\right)} = \frac{1}{2\pi} \frac{r_Y}{\sigma^2 + r_X^2 + r_Y^2}, \quad (\text{B.1})$$

$$u_Y = -\frac{1}{\pi \sigma^2} \frac{r_X}{2 \left(1 + \frac{r^2}{\sigma^2}\right)} = -\frac{1}{2\pi} \frac{r_X}{\sigma^2 + r_X^2 + r_Y^2}, \quad (\text{B.2})$$

where $\vec{r} = \vec{P} - \vec{x}_v = \overrightarrow{(r_X, r_Y)}$ with \vec{x}_v the vortex position and \vec{P} the point position where we are calculating the velocity, u_X and u_Y respectively the \vec{X} and \vec{Y} velocity component of \vec{u} , the induced velocity at the point \vec{P} , σ the vortex core radius size, and $r = |\vec{r}|$ the vector norm.

The different derivatives are then for u_X :

$$\frac{\partial u_X}{\partial X} = -\frac{1}{\pi} \frac{r_X r_Y}{(\sigma^2 + r_X^2 + r_Y^2)^2}, \quad (\text{B.3})$$

$$\frac{\partial u_X}{\partial Y} = \frac{1}{2\pi} \frac{\sigma^2 + r_X^2 - r_Y^2}{(\sigma^2 + r_X^2 + r_Y^2)^2}. \quad (\text{B.4})$$

$$(\text{B.5})$$

For u_Y , the derivatives are:

$$\frac{\partial u_Y}{\partial X} = -\frac{1}{2\pi} \frac{\sigma^2 - r_X^2 + r_Y^2}{(\sigma^2 + r_X^2 + r_Y^2)^2}, \quad (\text{B.6})$$

$$\frac{\partial u_Y}{\partial Y} = \frac{1}{\pi} \frac{r_X r_Y}{(\sigma^2 + r_X^2 + r_Y^2)^2}. \quad (\text{B.7})$$

$$(\text{B.8})$$

Appendix C

Force and moment calculus using complex images

In order to calculate the force and moment applied to the ellipse, one can use the extended Blasius formula as developed in Milne and Thompson (1968) [40]. This implies with the simulation that all the vortex elements are considered as point vortices, which because of their small core radius σ compared to the scale of the flow (in this case σ is of order $a \times 10^{-4}$) is still consistent. Like in section 2.3.6 with equation 2.50, one can use a conformal transformation to map the ellipse into a unit circle through the use of:

$$z = f(\zeta) = C \left(\zeta + \frac{\lambda}{\zeta} \right), \quad (\text{C.1})$$

with $C = (a + b)/2$, $\lambda = (a^2 + b^2)/(4C^2)$, z is the complex physical coordinate, and ζ the image coordinate in the unit circle. Note that this transformation is only conformal outside the ellipse. Afterwards, one can denote C_{el} the contour of the cylinder in the physical plane (i.e. the ellipse), and C the contour in the image plane (the cylinder). The geometry of the problem is presented in figure C.1.

In the calculation, one must distinguish between the complex freeflow velocity and the flow potential. Using this convention, the complex velocity potential in the ζ space for a uniform flow and a doublet at the origin to simulate the cylinder is:

$$w = \frac{B}{\zeta^2} - CU_\infty \frac{1}{\zeta} e^{i\alpha} + \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \ln(\zeta - \zeta_k) - \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \ln \left(\zeta - \frac{1}{\zeta_k^*} \right), \quad (\text{C.2})$$

with U_∞ the freeflow magnitude, α the angle of attack, Γ_k and ζ_k respectively the strength and position in the ζ space of the k^{th} vortex; * indicates a complex conjugate, n is the number of vortices and B/ζ^2 denotes the potential due to the rotation [40].

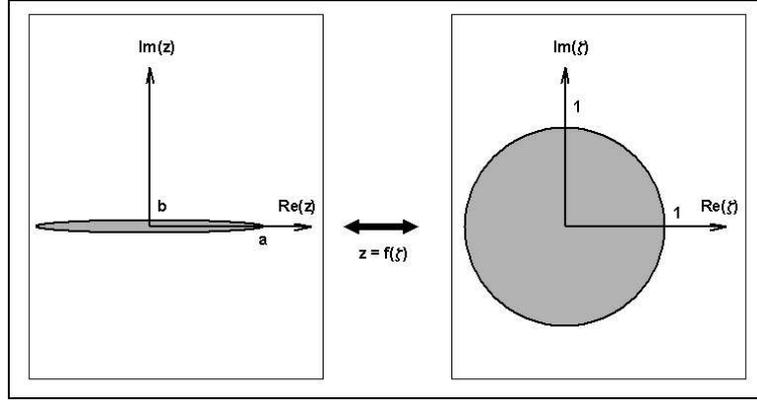


Figure C.1: Conformal transformation illustration

Here $B = (i/4)\Omega(a^2 + b^2)$ with $\Omega = \dot{\theta}$ the angular velocity (refer to figure 2.1 for definition of a and b).

On the other hand, the velocity due to freeflow velocity is:

$$\frac{dw_f}{dz} = -U_\infty e^{-i\alpha} - i * \Omega z. \quad (\text{C.3})$$

The velocity in the physical plane z due to w is:

$$\frac{dw}{dz} = \frac{dw}{d\zeta} \frac{d\zeta}{dz} = \frac{dw}{d\zeta} \left(\frac{dz}{d\zeta} \right)^{-1}. \quad (\text{C.4})$$

Using equation C.1, one has then:

$$\left(\frac{dz}{d\zeta} \right)^{-1} = \frac{\zeta^2}{C(\zeta^2 - \lambda)}. \quad (\text{C.5})$$

One should the note that as $\lambda < 1$, there is a singularity inside the cylinder defined by a circle of radius $\sqrt{\lambda}$ when computing the physical velocity. Although not important to compute the velocity of the vortices, problems arises when one wishes to integrate inside C_{el} .

Outside C_{el} , i.e. outside C , w is finite everywhere and analytical thus the velocity due to the potential is:

$$\frac{dw}{d\zeta} = -\frac{2B}{\zeta^3} + CU_\infty \frac{e^{i\alpha}}{\zeta^2} + \sum_{k=0}^n \frac{\Gamma_k}{2i\pi} \frac{1}{\zeta - \zeta_k} - \sum_{k=0}^n \frac{\Gamma_k}{2i\pi} \frac{1}{\zeta - \frac{1}{\zeta_k^*}}. \quad (\text{C.6})$$

The total velocity in the physical plane is then

$$\frac{dw_t}{dz} = \frac{dw_f}{dz} + \frac{dw}{d\zeta} \left(\frac{dz}{d\zeta} \right)^{-1}. \quad (\text{C.7})$$

Note that due to the sign convention used in the velocity formulation, the sign applied to vortices is inverted for the forces formula when one uses algebraic vortices.

C.1 Force calculus

For a cylinder, the extended theorem of Blasius for the forces states that the forces are equal to, if the ellipse is centered in $z = 0$:

$$\begin{aligned} X - iY = \frac{1}{2}i\rho \oint_{C_{el}} \left(\frac{dw}{dz} \right)^2 dz + \Omega\rho \oint_{C_{el}} z^* dw^* - i\rho \frac{\partial}{\partial t} \oint_{C_{el}} w^* dz^* \\ - 2\pi\kappa\rho iW^* - i\rho A \left\{ \Omega W^* + i \frac{dW^*}{dt} \right\}, \end{aligned} \quad (\text{C.8})$$

where $W = U_\infty e^{i\alpha}$, κ is the circulation around the cylinder, A is the surface of the contour C_{el} , and $\Omega = \dot{\theta}$ is the angular velocity. One can then distinguish the integrals:

$$X - iY = I_1 + I_2 + I_3 + I_4 \quad (\text{C.9})$$

$$I_1 = X_1 - iY_1 = \frac{1}{2}i\rho \oint_{C_{el}} \left(\frac{dw}{dz} \right)^2 dz \quad (\text{C.10})$$

$$I_2 = X_2 - iY_2 = -i\rho \frac{\partial}{\partial t} \oint_{C_{el}} w^* dz^* \quad (\text{C.11})$$

$$I_3 = X_3 - iY_3 = \Omega\rho \oint_{C_{el}} z^* dw^* \quad (\text{C.12})$$

$$I_4 = X_4 - iY_4 = -2\pi\kappa\rho iW^* - i\rho A \left\{ \Omega W^* + i \frac{dW^*}{dt} \right\}. \quad (\text{C.13})$$

C.1.1 Integral I1

For the first integral, one can use a development provided by Milne and Thompson (1968) [40], but adapted to the ellipse; more details on the complex formulation can be found in their textbook. One consider the basic integral:

$$X_1 - iY_1 = \frac{1}{2}i\rho \oint_{C_{el}} \left(\frac{dw}{dz} \right)^2 dz. \quad (\text{C.14})$$

One can also write it:

$$X_1 - iY_1 = \frac{1}{2}i\rho \oint_{C_{el}} \left(\frac{dw}{d\zeta} \right)^2 \frac{d\zeta}{dz} dz. \quad (\text{C.15})$$

There are two options. One is to integrate directly around C_{el} , but as noted earlier there is a singularity inside which is the circle of center 0, and of radius $\sqrt{\lambda}$, thus calculating the integral inside C_{el} could prove complicated. Thus, an alternative way was chosen. Using Cauchy's theorem for a function which is holomorphic almost everywhere except at singularity points, one can enlarge the contour C_{el} to S a circle with a very large radius, as in fig C.2. γ_k is a small contour drawn around the k^{th} vortex.

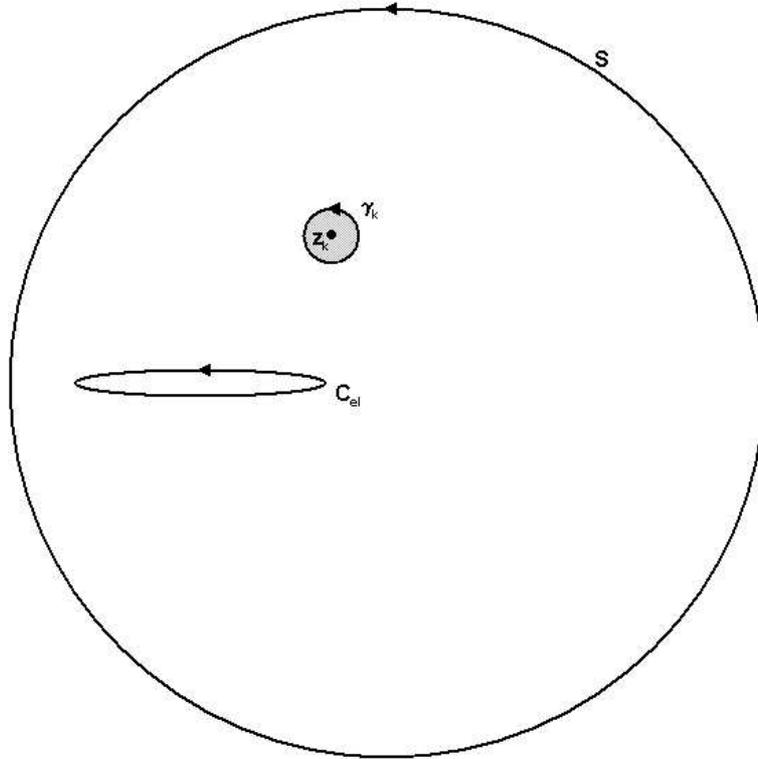


Figure C.2: Complex decomposition of contour

The equation then becomes:

$$X_1 - iY_1 = \frac{1}{2}i\rho \oint_{C_{el}} \left(\frac{dw}{dz} \right)^2 dz = \frac{1}{2}i\rho \oint_S \left(\frac{dw}{dz} \right)^2 dz - \sum_{k=0}^n \frac{1}{2}i\rho \oint_{\gamma_k} \left(\frac{dw}{dz} \right)^2 dz. \quad (\text{C.16})$$

In order to ease the resolution use for both integrals, one could choose the form of equation C.15 for the integral around S , and the form of equation C.14 for the integrals around γ_k . For the integral around S , one can further simplify the complex velocity (eq.

C.6), since $|z| \rightarrow \infty \Leftrightarrow |\zeta| \rightarrow \infty$, and one can then expand $1/(\zeta - \zeta_k)$ and $1/(\zeta - \frac{1}{\zeta_k^*})$ in powers of $1/\zeta$:

$$\begin{aligned} \frac{dw}{d\zeta} &= -\frac{2B}{\zeta^3} + CU \frac{e^{i\alpha}}{\zeta^2} \\ &+ \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta} \left(1 + \frac{\zeta_k}{\zeta} + \frac{\zeta_k^2}{\zeta^2} + \dots \right) \\ &- \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta} \left(1 + \frac{\zeta'_k}{\zeta} + \frac{\zeta'^2_k}{\zeta^2} + \dots \right), \end{aligned} \quad (\text{C.17})$$

with $\zeta'_k = 1/\zeta_k^*$ the image vortex of the k^{th} vortex inside C . Furthermore, one has:

$$\left(\frac{dz}{d\zeta} \right)^{-1} = \frac{\zeta^2}{C(\zeta^2 - \lambda)} \simeq \frac{1}{C} \quad \text{as } |\zeta| \rightarrow \infty. \quad (\text{C.18})$$

Therefore one can write on S :

$$\begin{aligned} \left(\frac{dw}{d\zeta} \right)^2 &= \frac{4B^2}{\zeta^6} + \frac{(CUe^{i\alpha})^2}{\zeta^4} - 4BCU_\infty \frac{e^{i\alpha}}{\zeta^5} \\ &- 4B \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta^4} \left(1 + \frac{\zeta_k}{\zeta} + \frac{\zeta_k^2}{\zeta^2} + \dots \right) \\ &+ 4B \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta^4} \left(1 + \frac{\zeta'_k}{\zeta} + \frac{\zeta'^2_k}{\zeta^2} + \dots \right) \\ &+ 2CU_\infty e^{i\alpha} \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta^3} \left(1 + \frac{\zeta_k}{\zeta} + \frac{\zeta_k^2}{\zeta^2} + \dots \right) \\ &- 2CU_\infty e^{i\alpha} \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta^3} \left(1 + \frac{\zeta'_k}{\zeta} + \frac{\zeta'^2_k}{\zeta^2} + \dots \right) \\ &+ 2 \sum_{j=0}^n \sum_{k=0}^n \frac{\Gamma_j \Gamma_k}{4\pi^2} \frac{1}{\zeta^2} \left(1 + \frac{\zeta_k}{\zeta} + \frac{\zeta_k^2}{\zeta^2} + \dots \right) \left(1 + \frac{\zeta'_k}{\zeta} + \frac{\zeta'^2_k}{\zeta^2} + \dots \right) \\ &- \sum_{k=0}^n \frac{\Gamma_k^2}{4\pi^2} \frac{1}{\zeta^2} \left(1 + \frac{\zeta_k}{\zeta} + \frac{\zeta_k^2}{\zeta^2} + \dots \right)^2 \\ &- \sum_{k=0}^n \frac{\Gamma_k^2}{4\pi^2} \frac{1}{\zeta^2} \left(1 + \frac{\zeta'_k}{\zeta} + \frac{\zeta'^2_k}{\zeta^2} + \dots \right)^2. \end{aligned} \quad (\text{C.19})$$

One can further reduce this to:

$$\begin{aligned}
 \left(\frac{dw}{d\zeta}\right)^2 &= \frac{4B^2}{\zeta^6} + \frac{(CU_\infty e^{i\alpha})^2}{\zeta^4} - 4BCU_\infty \frac{e^{i\alpha}}{\zeta^5} \\
 &- 4B \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta^4} \left[\left(\frac{\zeta_k}{\zeta} + \frac{\zeta_k^2}{\zeta^2} + \dots \right) - \left(\frac{\zeta'_k}{\zeta} + \frac{\zeta_k'^2}{\zeta^2} + \dots \right) \right] \\
 &+ 2CU_\infty e^{i\alpha} \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta^3} \left[\left(\frac{\zeta_k}{\zeta} + \frac{\zeta_k^2}{\zeta^2} + \dots \right) - \left(\frac{\zeta'_k}{\zeta} + \frac{\zeta_k'^{*2}}{\zeta^2} + \dots \right) \right] \\
 &+ 2 \sum_{j=0}^n \sum_{k=0}^n \frac{\Gamma_j \Gamma_k}{4\pi^2} \frac{1}{\zeta^2} \left(\frac{\zeta_k}{\zeta} + \frac{\zeta_k^2}{\zeta^2} + \dots \right) \left(\frac{\zeta'_j}{\zeta} + \frac{\zeta_j'^2}{\zeta^2} + \dots \right) \\
 &- \sum_{k=0}^n \frac{\Gamma_k^2}{4\pi^2} \frac{1}{\zeta^2} \left[\left(\frac{\zeta_k}{\zeta} + \frac{\zeta_k^{*2}}{\zeta^2} + \dots \right)^2 + \left(\frac{\zeta'_k}{\zeta} + \frac{\zeta_k'^{*2}}{\zeta^2} + \dots \right)^2 \right].
 \end{aligned}$$

That is, this equation has the form:

$$\left(\frac{dw}{d\zeta}\right)^2 = \frac{a_3}{\zeta^3} + \frac{a_4}{\zeta^4} + \dots$$

Now if one multiply this equation by equation C.18, one has:

$$\left(\frac{dw}{d\zeta}\right)^2 \frac{d\zeta}{dz} = \frac{1}{C} \left[\frac{a_3}{\zeta^3} + \frac{a_4}{\zeta^4} + \dots \right].$$

And thus by applying the residue theorem inside S one can see that the integral C.15 becomes:

$$\frac{1}{2} i\rho \oint_S \left(\frac{dw}{d\zeta}\right)^2 \frac{d\zeta}{dz} d\zeta = 0. \quad (\text{C.20})$$

If one wants to add a circulation term, one has to add a term $i(\kappa/2\pi) \ln(\zeta)$ to w in the ζ plane for a circulation κ around the ellipse. The complex velocity becomes:

$$\begin{aligned}
 \frac{dw}{d\zeta} &= -\frac{2B}{\zeta^3} + CU_\infty \frac{e^{i\alpha}}{\zeta^2} + i\frac{\kappa}{2\pi} \frac{1}{\zeta} \\
 &+ \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta} \left[\left(\frac{\zeta_k}{\zeta} + \frac{\zeta_k^2}{\zeta^2} + \dots \right) - \left(\frac{\zeta'_k}{\zeta} + \frac{\zeta_k'^2}{\zeta^2} + \dots \right) \right].
 \end{aligned}$$

Using then the same development, it is easy to see that in this case the result is unchanged and thus:

$$\frac{1}{2} i\rho \oint_S \left(\frac{dw}{d\zeta}\right)^2 \frac{d\zeta}{dz} d\zeta = 0. \quad (\text{C.21})$$

The other integrals are a bit simpler to develop. One can concentrate on the integrals $I_{1,k}$ which are the components of I1. At a given k :

$$I_{1,k} = -\frac{1}{2} i\rho \oint_{\gamma_k} \left(\frac{dw}{dz}\right)^2 dz. \quad (\text{C.22})$$

One can rewrite the complex velocity as:

$$\frac{dw}{dz} = f(z) + \frac{i\Gamma_k}{2\pi} \frac{1}{z - z_k}, \quad (\text{C.23})$$

$$\text{where } f(z) = \frac{d}{dz} \left[w - \frac{i\Gamma_k}{2\pi} \ln(z - z_k) \right]. \quad (\text{C.24})$$

Hence, the function $f(z)$ represents the complex velocity obtained by omitting the k^{th} vortex from the original potential, and f is holomorphic within the contour γ_k . Then, the complex velocity can be written:

$$\left(\frac{dw}{dz} \right)^2 = (f(z))^2 + 2f(z) \frac{i\Gamma_k}{2\pi} \frac{1}{z - z_k} - \frac{\Gamma_k^2}{4\pi^2} \frac{1}{(z - z_k)^2}. \quad (\text{C.25})$$

Now by Taylor's theorem, near z_k :

$$f(z) = f(z_k) + (z - z_k) \frac{df}{dz}(z_k) + \dots \quad (\text{C.26})$$

Then again, using the residue theorem for γ_k , one obtains:

$$I_{1,k} = -\frac{1}{2} i\rho \oint_{\gamma_k} \left(\frac{dw}{d\zeta} \right)^2 \frac{d\zeta}{dz} d\zeta = i\rho \Gamma_k (u_k - iv_k). \quad (\text{C.27})$$

Thus, integral I1 is equal to:

$$I_1 = \sum_{k=0}^n i\rho \Gamma_k (u_k - iv_k). \quad (\text{C.28})$$

Note that crucially $u_k - iv_k$ is equal to the complex velocity in the fixed frame (or inertial frame), that is the freeflow velocity is excluded, and thus:

$$u_k - iv_k = \frac{dw}{d\zeta} \left(\frac{dz}{d\zeta} \right)^{-1}. \quad (\text{C.29})$$

C.1.2 Integral I2

This integral is slightly easier to calculate:

$$I_2 = X_2 - iY_2 = -i\rho \frac{\partial}{\partial t} \oint_{C_{el}} w^* dz^*. \quad (\text{C.30})$$

It is more convenient to consider:

$$I_2^* = X_2 + iY_2 = i\rho \frac{\partial}{\partial t} \oint_{C_{el}} w dz. \quad (\text{C.31})$$

If one chose to decompose the integral by parts:

$$I_2^* = i\rho \frac{\partial}{\partial t} \left\{ [wz]_{pc} - \oint_{C_{el}} z \frac{dw}{dz} dz \right\}, \quad (C.32)$$

where $[wz]_{pc}$ is the difference between the value of wz at the beginning and end of the plate contour. Since there is no jump in the stream function along the contour, and since the nascent vortices are not connected by the vortex sheet to the edge of the ellipse, this term is identically equal to 0. Then, the remaining integral can be put on the form

$$\oint_{C_{el}} z \frac{dw}{dz} dz = \oint_C z \frac{dw}{d\zeta} \frac{d\zeta}{dz} d\zeta = \oint_C z \frac{dw}{d\zeta} d\zeta = \oint_C C \left(\zeta + \frac{\lambda}{\zeta} \right) \frac{dw}{d\zeta} d\zeta, \quad (C.33)$$

with the conformal transformation of z described by equation C.1. Then using the complex velocity as defined in equation C.6, one has:

$$\oint_{C_{el}} z \frac{dw}{dz} dz = \oint_C C \left(\frac{\zeta^2 + \lambda}{\zeta} \right) \left(-\frac{2B}{\zeta^3} C U_\infty \frac{e^{i\alpha}}{\zeta^2} + \sum_{k=0}^n \frac{\Gamma_k}{2i\pi} \frac{1}{\zeta - \zeta_k} - \sum_{k=0}^n \frac{\Gamma_k}{2i\pi} \frac{1}{\zeta - \zeta'_k} \right) d\zeta, \quad (C.34)$$

which can be rewritten:

$$\begin{aligned} \oint_{C_{el}} z \frac{dw}{dz} dz = & \oint_C \left\{ C^2 U_\infty \frac{e^{i\alpha}}{\zeta^2} \frac{\zeta^2 + \lambda}{\zeta} \right\} d\zeta \\ & - \oint_C \left\{ C \frac{2B}{\zeta^3} \frac{\zeta^2 + \lambda}{\zeta} \right\} d\zeta \\ & + \oint_C \left\{ \sum_{k=0}^n C \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta - \zeta_k} \frac{\zeta^2 + \lambda}{\zeta} \right\} d\zeta \\ & - \oint_C \left\{ \sum_{k=0}^n C \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta - \zeta'_k} \frac{\zeta^2 + \lambda}{\zeta} \right\} d\zeta. \end{aligned} \quad (C.35)$$

Then inside C_{el} , the first integral has one pole in 0 of order three, the second integral has one pole in 0 of order four, the third integral has one pole in 0 of order one, and the last integral has two poles of order one in 0 and ζ'_k . Therefore, using the residue

theorem one obtain for the different integrals :

$$\oint_C \left\{ C^2 U_\infty \frac{e^{i\alpha}}{\zeta^2} \frac{\zeta^2 + \lambda}{\zeta} \right\} d\zeta = C^2 U_\infty e^{i\alpha} \oint_C \left(\frac{1}{\zeta} + \frac{\lambda}{\zeta^3} \right) d\zeta, \quad (\text{C.36})$$

$$- \oint_C \left\{ C \frac{2B}{\zeta^3} \frac{\zeta^2 + \lambda}{\zeta} \right\} d\zeta = -2BC \oint_C \left(\frac{1}{\zeta^2} + \frac{\lambda}{\zeta^4} \right) d\zeta, \quad (\text{C.37})$$

$$\oint_C \left\{ \sum_{k=0}^n C \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta - \zeta_k} \frac{\zeta^2 + \lambda}{\zeta} \right\} d\zeta = 2iC\pi \text{res}(f_2, 0) = 2iC\pi \lim_{\zeta \rightarrow 0} \zeta f_2 \quad (\text{C.38})$$

$$\text{with} \quad f_2 = \sum_{k=0}^n C \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta - \zeta_k} \frac{\zeta^2 + \lambda}{\zeta}$$

$$\oint_C \left\{ \sum_{k=0}^n C \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta - \zeta'_k} \frac{\zeta^2 + \lambda}{\zeta} \right\} d\zeta = 2iC\pi \{ \text{res}(f_3, 0) + \text{res}(f_3, \zeta'_k) \} = 2iC\pi \left\{ \lim_{\zeta \rightarrow 0} \zeta f_3 + \lim_{\zeta \rightarrow \zeta'_k} (\zeta - \zeta'_k) f_3 \right\} \quad (\text{C.39})$$

$$\text{with} \quad f_3 = \sum_{k=0}^n C \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta - \zeta'_k} \frac{\zeta^2 + \lambda}{\zeta}.$$

The results are:

$$\oint_C \left\{ C^2 U_\infty \frac{e^{i\alpha}}{\zeta^2} \frac{\zeta^2 + \lambda}{\zeta} \right\} d\zeta = 2i\pi C^2 U_\infty e^{i\alpha} \quad (\text{C.40})$$

$$- \oint_C \left\{ C \frac{2B}{\zeta^3} \frac{\zeta^2 + \lambda}{\zeta} \right\} d\zeta = 0 \quad (\text{C.41})$$

$$\oint_C \left\{ \sum_{k=0}^n C \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta - \zeta_k} \frac{\zeta^2 + \lambda}{\zeta} \right\} d\zeta = C \sum_{k=0}^n \Gamma_k \frac{\lambda}{\zeta_k} \quad (\text{C.42})$$

$$\oint_C \left\{ \sum_{k=0}^n C \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta - \zeta'_k} \frac{\zeta^2 + \lambda}{\zeta} \right\} d\zeta = -C \sum_{k=0}^n \Gamma_k \zeta'_k. \quad (\text{C.43})$$

Thus equation C.35 is the equal to:

$$\oint_C z \frac{dw}{dz} dz = 2i\pi C^2 U_\infty e^{i\alpha} + C \sum_{k=0}^n \Gamma_k \left(\frac{\lambda}{\zeta_k} + \zeta'_k \right), \quad (\text{C.44})$$

and I_2^* is equal to

$$I_2^* = -2\pi\rho C^2 \frac{\partial}{\partial t} [U_\infty e^{i\alpha}] + i\rho C \frac{\partial}{\partial t} \left[\sum_{k=0}^n \Gamma_k \left(\frac{\lambda}{\zeta_k} + \zeta'_k \right) \right]. \quad (\text{C.45})$$

Note that if one add a circulation κ to the potential w as in section C.1.1, it is easy to see that the integral remains unchanged.

C.1.3 Integral I3

This term is:

$$I_3 = X_3 - iY_3 = \Omega\rho \oint_{C_{el}} z^* dw^*. \quad (C.46)$$

Again, to suppress the conjugate terms, one can write:

$$I_3^* = X_3 + iY_3 = \Omega\rho \oint_{C_{el}} z dw. \quad (C.47)$$

This in turn, similar to the development for I2 for equation C.33, is equal to:

$$\begin{aligned} I_3^* = X_3 + iY_3 &= \Omega\rho \oint_{C_{el}} z \frac{dw}{dz} dz = \Omega\rho \oint_C z \frac{dw}{d\zeta} \frac{d\zeta}{dz} \frac{dz}{d\zeta} d\zeta \\ &= \Omega\rho \oint_C z \frac{dw}{d\zeta} d\zeta = \Omega\rho \oint_C C \left(\zeta + \frac{\lambda}{\zeta} \right) \frac{dw}{d\zeta} d\zeta, \end{aligned} \quad (C.48)$$

with the conformal transformation described by equation C.1. Then, I_3^* is equal to:

$$I_3^* = \Omega\rho \left[2i\pi C^2 U_\infty e^{i\alpha} + C \sum_{k=0}^n \Gamma_k \left(\frac{\lambda}{\zeta_k} + \zeta'_k \right) \right]. \quad (C.49)$$

And as for I_2 , the integral is unchanged with an additional circulation.

C.1.4 Integral I4

The last term is equal to

$$I_4 = X_4 - iY_4 = -2\pi\kappa_t \rho i W^* - i\rho A \left\{ \Omega W^* + i \frac{dW^*}{dt} \right\}, \quad (C.50)$$

where κ_t represents the total circulation around the cylinder, A the surface of the cross-section of the cylinder, Ω the angular velocity of the flow, and W^* the complex freeflow velocity at infinity. Then, for an ellipse :

$$A = \pi ab. \quad (C.51)$$

$$W^* = U_\infty e^{-i\alpha}. \quad (C.52)$$

Hence

$$I_4 = -2\pi\kappa_t \rho i U_\infty e^{-i\alpha} - i\rho A \left\{ \Omega U_\infty e^{-i\alpha} + i \frac{dU_\infty e^{-i\alpha}}{dt} \right\}. \quad (C.53)$$

Note that with the w formulation in equation C.2, there is a circulation $-\sum_{k=0}^n \Gamma_k/(2\pi)$ in the ellipse due to the image vortices. This is consistent with Kelvin's theorem as exposed in equation 2.20, and thus is actually the same. Equation C.53 becomes:

$$I_4 = i\rho U_\infty e^{-i\alpha} \sum_{k=0}^n \Gamma_k - C\kappa\rho i U_\infty e^{-i\alpha} - i\rho A \left\{ \Omega U_\infty e^{-i\alpha} + i \frac{dU_\infty e^{-i\alpha}}{dt} \right\}. \quad (\text{C.54})$$

C.2 Moment calculus

For a cylinder, the extended theorem of Blasius for the moment states that the moment is equal to the real part of:

$$M + iN = -\frac{1}{2}\rho \oint_{C_{el}} z \left(\frac{dw}{dz} \right)^2 dz - \rho W^* \oint_{C_{el}} z dw + \rho \frac{\partial}{\partial t} \oint_{C_{el}} z w dz^*,$$

where the ellipse being centered in $z = 0$. One can then distinguish the integrals:

$$M + iN = M_1 + M_2 + M_3, \quad (\text{C.55})$$

$$M_1 = -\frac{1}{2}\rho \oint_{C_{el}} z \left(\frac{dw}{dz} \right)^2 dz, \quad (\text{C.56})$$

$$M_2 = -\rho W^* \oint_{C_{el}} z dw, \quad (\text{C.57})$$

$$M_3 = \rho \frac{\partial}{\partial t} \oint_{C_{el}} z w dz^*. \quad (\text{C.58})$$

While M_1 and M_2 are fairly straightforward to calculate, M_3 has proven to be much more difficult because of the presence of the conjugate inside the integral.

C.2.1 Integral M1

The integral M_1 is calculated in the exact same manner as integral I_1 in section C.1.1. That is one decomposes the integral into several circular contours S where $z \rightarrow \infty$, and several small contours γ_k around the vortices. The end result is consequently with a circulation κ and n vortices in the flow:

$$M_1 = \frac{i}{2}\rho\kappa^2 - \rho \sum_{k=0}^n z_k \Gamma_k (u_k - iv_k). \quad (\text{C.59})$$

One can use as a potential for a circulation κ : $i[\kappa/(2\pi)] \ln(\zeta)$.

C.2.2 Integral M2

Integral M_2 can be deduced directly from I_2^* and one thus obtains:

$$M_2 = -\rho W^* \left[2i\pi C^2 U_\infty e^{i\alpha} + C \sum_{k=0}^n \Gamma_k \left(\frac{\lambda}{\zeta_k} + \zeta'_k \right) \right]. \quad (\text{C.60})$$

C.2.3 Integral M3

Because of the conjugate, this is the most complicated integral and it obliges to decompose M_3 into several parts. The basic integral is:

$$M_3 = \rho \frac{\partial}{\partial t} \oint_{C_{el}} z w dz^*. \quad (\text{C.61})$$

One can rewrite it as:

$$M_3 = \rho \frac{\partial}{\partial t} \oint_C z \frac{dz^*}{d\zeta} w d\zeta. \quad (\text{C.62})$$

On a unit cylinder, one has the relation $\zeta^* = 1/\zeta$. Hence $dz^*/d\zeta$ becomes:

$$z^* = C \left(\zeta^* + \frac{\lambda}{\zeta^*} \right) = C \left(\frac{1}{\zeta} + \lambda\zeta \right), \quad (\text{C.63})$$

$$\frac{dz^*}{d\zeta} = C \left(\lambda - \frac{1}{\zeta^2} \right). \quad (\text{C.64})$$

And thus equation C.62 becomes:

$$M_3 = \rho \frac{\partial}{\partial t} \oint_C C^2 \left(\zeta + \frac{\lambda}{\zeta} \right) \left(\lambda - \frac{1}{\zeta^2} \right) w d\zeta. \quad (\text{C.65})$$

Expanding the integrand leads to:

$$M_3 = \rho \frac{\partial}{\partial t} \oint_C C^2 \left(\lambda\zeta + \frac{\lambda^2 - 1}{\zeta} - \frac{\lambda}{\zeta^3} \right) w d\zeta. \quad (\text{C.66})$$

Using w from equation C.2, one has:

$$M_3 = \rho \frac{\partial}{\partial t} \oint_C C^2 \left(\lambda\zeta + \frac{\lambda^2 - 1}{\zeta} - \frac{\lambda}{\zeta^3} \right) \times \left[\frac{B}{\zeta^2} - C U_\infty \frac{e^{i\alpha}}{\zeta} + \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \ln(\zeta - \zeta_k) - \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \ln(\zeta - \zeta'_k) \right] d\zeta. \quad (\text{C.67})$$

One can then decompose this integral into three parts $M_{3,1}$, $M_{3,2}$, and $M_{3,3}$ such that:

$$M_{3,1} = \rho C^2 \frac{\partial}{\partial t} \oint_C \left(\lambda\zeta + \frac{\lambda^2 - 1}{\zeta} - \frac{\lambda}{\zeta^3} \right) \left[\frac{B}{\zeta^2} - C U_\infty \frac{e^{i\alpha}}{\zeta} \right] d\zeta, \quad (\text{C.68})$$

$$M_{3,2} = \rho C^2 \frac{\partial}{\partial t} \oint_C \left(\lambda\zeta - \frac{\lambda}{\zeta^3} \right) \left[\sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \ln(\zeta - \zeta_k) - \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \ln(\zeta - \zeta'_k) \right] d\zeta, \quad (\text{C.69})$$

$$M_{3,3} = \rho C^2 \frac{\partial}{\partial t} \oint_C \left(\frac{\lambda^2 - 1}{\zeta} \right) \left[\sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \ln(\zeta - \zeta_k) - \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \ln(\zeta - \zeta'_k) \right] d\zeta. \quad (\text{C.70})$$

Integral $M_{3,1}$ is easy to develop using the definition of the residus and produces:

$$M_{3,1} = \rho C^2 \frac{\partial}{\partial t} \oint_C \left\{ \frac{\lambda B}{\zeta} - CU_\infty \frac{e^{i\alpha}}{\zeta^3} + (\lambda^2 - 1) \left(\frac{B}{\zeta^3} - CU_\infty \frac{e^{i\alpha}}{\zeta^2} \right) - \frac{\lambda B}{\zeta^5} + CU_\infty \frac{e^{i\alpha}}{\zeta^4} \right\} d\zeta,$$

that is $M_{3,1} = \rho C^2 2i\pi \frac{\partial}{\partial t} (\lambda B)$. (C.71)

Integral $M_{3,2}$ is more tricky and requires an intermediary development using an integration by parts:

$$M_{3,2} = \rho C^2 \frac{\partial}{\partial t} \left[\left(\lambda \frac{\zeta^2}{2} - \frac{\lambda}{2\zeta^2} \right) \left\{ \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \ln(\zeta - \zeta_k) - \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \ln(\zeta - \zeta'_k) \right\} \right]_C$$

$$- \rho C^2 \frac{\partial}{\partial t} \oint_C \left(\lambda \frac{\zeta^2}{2} - \frac{\lambda}{2\zeta^2} \right) \left\{ \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta - \zeta_k} - \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \frac{1}{\zeta - \zeta'_k} \right\} d\zeta. \quad (C.72)$$

The first part of this formula is obviously zero as there are no potential discontinuities over the boundary and using the residues theorem with the remaining of the formula, one obtains:

$$M_{3,2} = -\rho C^2 \frac{\lambda}{2} \sum_{k=0}^n \Gamma_k \left\{ \zeta_k'^2 - \frac{1}{\zeta_k^2} \right\}. \quad (C.73)$$

Finally, in integral $M_{3,3}$ (equation C.68) although it is easy to see that as $\ln(\zeta - \zeta_k)$ is continuous on (C) the first term of the integral can be obtained with the residues theorem and therefore it is equal to:

$$\rho C^2 \frac{\partial}{\partial t} \oint_C \left(\frac{\lambda^2 - 1}{\zeta} \right) \left[\sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \ln(\zeta - \zeta_k) \right] d\zeta = -\rho C^2 (\lambda^2 - 1) \frac{\partial}{\partial t} \sum_{k=0}^n \Gamma_k \ln(-\zeta_k). \quad (C.74)$$

Conversely the second term requires a change of variable because ζ'_k is inside C ; at the k^{th} vortex one can then use $Z_k = \zeta - \zeta'_k$, and thus the second term becomes:

$$\rho C^2 \frac{\partial}{\partial t} \oint_C \frac{\lambda^2 - 1}{\zeta} \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \ln(\zeta - \zeta'_k) d\zeta = \rho C^2 (\lambda^2 - 1) \frac{\partial}{\partial t} \sum_{k=0}^n \frac{i\Gamma_k}{2\pi} \oint_{C_k} \left(\frac{1}{Z_k + \zeta'_k} \right) \ln(Z_k) d\zeta. \quad (C.75)$$

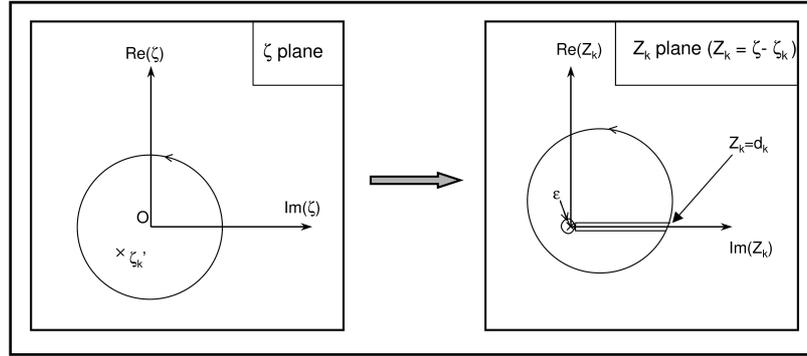
For the k^{th} vortex, the contour C_k is defined by figure C.3.

Then using Cauchy theorem, one has on the different parts of the contour in the Z_k plane, for a given k :

$$\oint_{C_k} \left(\frac{1}{Z_k + \zeta'_k} \right) \ln(Z_k) d\zeta + \int_{d_k}^\varepsilon \frac{\ln(re^{2i\pi})}{r + \zeta_k} dr$$

$$+ \int_{\varphi=2\pi}^0 \frac{\ln(\varepsilon e^{i\varphi})}{\varepsilon e^{i\varphi} + \zeta_k} d(\varepsilon e^{i\varphi}) + \int_\varepsilon^{d_k} \frac{\ln(r)}{r + \zeta_k} dr$$

$$= 2i\pi \text{res} \left(\frac{1}{Z_k + \zeta'_k} \ln(Z_k), -\zeta'_k \right), \quad (C.76)$$


 Figure C.3: Contour C_k description

with $\text{res}(f, x)$ denoting the residue of f in x . As $\varepsilon \rightarrow 0$, the third term of expression C.76 disappears, and one get:

$$\begin{aligned} & \frac{i\Gamma_k}{2\pi} \oint_{C_k} \left(\frac{1}{Z_k + \zeta'_k} \right) \ln(Z_k) d\zeta \\ &= \frac{i\Gamma_k}{2\pi} \left\{ 2i\pi \int_0^{d_k} \frac{1}{r + \zeta_k} dr + 2i\pi \text{res} \left(\frac{1}{Z_k + \zeta'_k} \ln(Z_k), -\zeta'_k \right) \right\} \\ & \quad = -\Gamma_k [\ln(d_k + \zeta'_k) - \ln(\zeta'_k) + \ln(-\zeta'_k)]. \end{aligned} \quad (\text{C.77})$$

Therefore, $M_{3,3}$ is equal to:

$$\begin{aligned} M_{3,3} &= -\rho C^2 (\lambda^2 - 1) \frac{\partial}{\partial t} \left\{ \sum_{k=0}^n \Gamma_k \ln(-\zeta_k) \right\} \\ & \quad - \rho C^2 (\lambda^2 - 1) \frac{\partial}{\partial t} \left\{ \sum_{k=0}^n \Gamma_k [\ln(d_k + \zeta'_k) - \ln(\zeta'_k) + \ln(-\zeta'_k)] \right\}, \end{aligned} \quad (\text{C.78})$$

or more simply

$$\begin{aligned} M_{3,3} &= \\ & -\rho C^2 (\lambda^2 - 1) \frac{\partial}{\partial t} \left\{ \sum_{k=0}^n \Gamma_k [\ln(-\zeta_k) + \ln(d_k + \zeta'_k) - \ln(\zeta'_k) + \ln(-\zeta'_k)] \right\}. \end{aligned} \quad (\text{C.79})$$

Appendix D

Force and moment calculus using Wu method

Because the simulations are done in two dimensions ($2D$), all calculus presented hereafter are carried out in $2D$. They can be generalized to $3D$ cases but it is beyond the scope of this thesis.

Well adapted for unbounded flows, Wu [65] presented a formula to compute the force \vec{F} as a function of the rate of change of vorticity for a finite domain:

$$\vec{F} = -\rho \frac{d}{dt} \int_{R_\infty} \vec{P} \times \vec{\omega} dr + \rho \frac{d}{dt} \int_{R_s} \vec{v}_s dr, \quad (\text{D.1})$$

where \vec{P} is a position, $\vec{\omega} = \omega \vec{K}$ is the local vorticity, with \vec{K} a unit vector orthogonal to the (\vec{x}, \vec{y}) plane, R_∞ is a limitless region jointly occupied by the solid body and the fluid, R_s is the region occupied by the solid body, and \vec{v}_s is the velocity of the body.

A similar formula is used for the aerodynamic moment:

$$\vec{M} = \frac{\rho}{2} \frac{d}{dt} \int_{R_\infty} P^2 \vec{\omega} dr + \rho \frac{d}{dt} \int_{R_s} \vec{P} \times \vec{v}_s dr, \quad (\text{D.2})$$

where $P = |\vec{P}|$.

D.1 Force calculus

It is convenient to separate first the formula in equation D.1 into two terms:

(D.3)

$$\vec{F} = \vec{F}_1 + \vec{F}_2 \quad (D.4)$$

$$\vec{F}_1 = -\rho \frac{d}{dt} \int_{R_\infty} \vec{P} \times \vec{\omega} dr, \quad (D.5)$$

$$\vec{F}_2 = \rho \frac{d}{dt} \int_{R_s} \vec{v}_s dr. \quad (D.6)$$

With a blob vortex method, the integral in \vec{F}_1 can be rewritten as:

$$\int_{R_\infty} \vec{P} \times \vec{\omega} dr = \int_{R_\infty} \vec{P} \times \left[\sum \Gamma_j \vec{K} \gamma(\vec{P} - \vec{x}_{v,j}) \right] dr, \quad (D.7)$$

with $\vec{x}_{v,j}$ and Γ_j the position and strength of the j^{th} vortex, respectively. For an algebraic core vortex, as σ tends to zero and as the domain is limitless, one can approximate the core function as a two dimensional Dirac function:

$$\Gamma_j \vec{K} \gamma(\vec{P} - \vec{x}_{v,j}) = \Gamma_j \vec{K} \delta^2(\vec{P} - \vec{x}_{v,j}) = \Gamma_j \vec{K} \delta(x - x_j) \delta(y - y_j). \quad (D.8)$$

Thus equation D.7 becomes:

$$\int_{R_\infty} \vec{P} \times \vec{\omega} dr = \int_{R_\infty} \vec{P} \times \left[\sum \Gamma_j \vec{K} \delta(x - x_j) \delta(y - y_j) \right] dr. \quad (D.9)$$

Using the properties of the dirac function, the integral becomes simply:

$$\int_{R_\infty} \vec{P} \times \vec{\omega} dr = \sum \vec{x}_{v,j} \times \left(\Gamma_j \vec{K} \right). \quad (D.10)$$

Now, one simply obtains for \vec{F}_1 :

$$\vec{F}_1 = -\rho \frac{d}{dt} \left[\sum \vec{x}_{v,j} \times \left(\Gamma_j \vec{K} \right) \right]. \quad (D.11)$$

As for the integral \vec{F}_2 , one can use a body velocity

$$\vec{v}_s = \vec{U}_\infty + \vec{\Omega} \times \vec{P}, \quad (D.12)$$

with Ω an angular velocity and $\vec{\Omega} = \Omega \vec{K}$, and \vec{U}_∞ a linear velocity. In equation D.12, both terms are time dependent but uniform in R_∞ . For an ellipse, there are two parameters: a half the length of the ellipse on its major axis, and b half the length of

the ellipse on its minor axis (see figure 2.1 in chapter 2). Then in cylindrical coordinates one has $x = a s \cos(\theta)$ and $y = b s \sin(\theta)$ and \vec{F}_2 can be rewritten as:

$$\vec{F}_2 = \rho \frac{d}{dt} \left[\vec{U}_\infty \int_{R_s} dr + \int_0^{2\pi} \int_0^1 \vec{\Omega} \times \vec{P} s ds d\theta \right], \quad (\text{D.13})$$

which leads to:

$$\vec{F}_2 = \rho \pi a b \frac{d\vec{U}_\infty}{dt}. \quad (\text{D.14})$$

Finally, one obtains:

$$\vec{F} = -\rho \frac{d}{dt} \left[\sum \vec{x}_{v,j} \times \left(\Gamma_j \vec{K} \right) \right] + \rho \pi a b \frac{d\vec{U}_\infty}{dt}. \quad (\text{D.15})$$

This formula is only valid if the calculation is done in an inertial frame (static in time). If one is working in the body fixed frame, the added term depends on the transformation; if one only applies a translation, there is no additional term. Conversely, there are some modifications if one applies a rotation to the velocity field.

D.2 Moment calculus

As for the force, it is convenient to separate first the formula in equation D.2 into two terms:

$$\vec{M} = \vec{M}_1 + \vec{M}_2, \quad (\text{D.16})$$

$$\vec{M}_1 = \frac{\rho}{2} \frac{d}{dt} \int_{R_\infty} P^2 \vec{\omega} dr, \quad (\text{D.17})$$

$$\vec{M}_2 = \rho \frac{d}{dt} \int_{R_s} \vec{P} \times \vec{v}_s dr. \quad (\text{D.18})$$

For \vec{M}_1 , developing the calculus in the same manner as for \vec{F}_1 , one can find that around the centroid of the ellipse which is also the origin:

$$\vec{M}_1 = \vec{K} \frac{\rho}{2} \frac{d}{dt} \left[\sum x_{v,j}^2 \Gamma_j \right]. \quad (\text{D.19})$$

Again, for \vec{M}_2 , developing the calculus in the same manner as the second part of the integral in equation D.13 for \vec{F}_2 , one finds:

$$\vec{M}_2 = \vec{K} \frac{\pi \rho}{4} (a^2 - b^2) \frac{d\Omega}{dt}. \quad (\text{D.20})$$

Therefore, \vec{M} is equal to:

$$\vec{M} = \vec{K} \frac{\rho}{2} \frac{d}{dt} \left[\sum x_{v,j}^2 \Gamma_j \right] + \vec{K} \frac{\pi \rho}{4} a b (a^2 + b^2) \frac{d\Omega}{dt}. \quad (\text{D.22})$$

As for the force, this formula is also only valid if the calculation is done in an inertial frame. Thus, if one is working in the body fixed frame, one has to add a term to take into account the additional effect. Therefore, the formula becomes:

$$\vec{M} = \vec{K} \frac{\rho}{2} \frac{d}{dt} \left[\sum x_{v,j}^2 \Gamma_j \right] - \vec{K} \rho \left[\sum \Gamma_l \vec{P}_l \cdot \vec{U}_\infty \right] + \vec{K} \frac{\pi \rho}{4} a b (a^2 + b^2) \frac{d\Omega}{dt}, \quad (\text{D.23})$$

with l an indices going through every vortex except the boundary vortices.

Bibliography

- [1] C Anderson and C Greengard. Vortex methods. *Lecture Notes in Mathematics Series, Springer-Verlag, Berlin*, 1360, 1988.
- [2] C Anderson and C Greengard. Vortex dynamics and vortex methods. *Lecture in Applied Mathematics Series, American Mathematical Society, New York*, 28, 1991.
- [3] M Athans. Crisp control is always better than fuzzy feedback control. Eufit 99 debate with prof L.Zadeh, 1999. IP : <http://fuzzy.iau.dtu.dk/download/athans99/> and the debate is at the IP : <http://fuzzy.iau.dtu.dk/debate.nsf/>.
- [4] M.J Ballas. Active control of flexible system. *Journal of Optimization Theory and Applications*, 25:415–436, 1978.
- [5] Beale and Madja. Vortex methods, i and ii. *Mathematical Computation*, 39:1, 1982.
- [6] J.T Beale, G.H Cottet, and S Huberson. Vortex flows and related numerical methods. *NATO ASI Series, Kluwer Academic, Dordrecht, The Netherlands*, August 1993.
- [7] R.D Blevins. Application of the discrete vortex method to fluid structure interaction. *Journal of Pressure Vessel Technology*, 113:437–445, August 1991.
- [8] R Caffish. Mathematical aspects of vortex dynamics. *Society for Industrial and Applied Mathematics, Philadelphia, PA*, August 1989.
- [9] A.J Chorin. Numerical study of slightly viscous flow. *Journal of Fluid Mechanics*, 57, 1973.
- [10] A.J Chorin. Vortex methods. *University of California, Berkeley, Course 2*, 1993.
- [11] N.R. Clarke. Two dimensional flow simulation using discrete vortex methods on mimd arrays. *Southampton University PhD*, 1992.

- [12] R.R. Clements. An inviscid model of two-dimensional vortex shedding. *Journal of Fluid Mechanics*, 57:321, 1973.
- [13] G.H Cottet and P Koutsoumakos. Vortex methods, theory and practice. *Journal of Basic Engineering, ASME*, 2000.
- [14] R Wille E Berger. The mechanics of the formation of vortices behind bluff bodies. *Journal of Fluid Mechanics*, 25:401–413, 1966.
- [15] R Wille E Berger. Periodic flow phenomena. *Annual Review of Fluid Mechanics*, 4:313–340, Jan 1972.
- [16] A. Fage and F.C. Johansen. On the flow of air behind an inclined flat plate of infinite span. *Proceeding of Royal Society*, 79:170–197, 1927.
- [17] A. Fage and F.C. Johansen. The structure of the vortex sheet. *Philosophy Magazine*, 7:417, 1928.
- [18] C.C Feng. The measurement of vortex-induced effects in flow past stationary and oscillating cylinder and d section cylinder. *MASc Thesis, University of British Columbia*, 1968.
- [19] J.A Fromme and M.A Golberg. Transient response analysis of nonlinear modally coupled structural dynamics equations of motion. *AIAA Journal*, 36:1486–1493, 1998.
- [20] Y Gagnon, G.H Cottet, D Dritschel, A Ghoniem, and E Meisburg. Vortex flows and related numerical methods ii. *ESAIM Proc*, 1996.
- [21] J.H Gerrard. Numerical computation of the magnitude and frequency of the lift on a circular cylinder. *Phil. Trans. A*, 261:137, 1967.
- [22] O. Hald. The convergence of vortex methods ii. *SIAM, Journal of Numerical Analysis*, 16:726, 1979.
- [23] N.D Ham. Aerodynamic loading on a two-dimensional airfoil during dynamic stall. *AIAA Journal*, 6:1927–1934, 1968.
- [24] R.T Hartlen and I.G Currie. Lift-oscillator model of vortex induced vibration. *Journal of the Engineering Mechanics Division of ASCE*, 96 EM5:577–591, 1970.

- [25] H.J.Lugt and H.J.Haussling. Laminar flow past an abruptly accelerated elliptic cylinder at 45 incidence. *Journal of Fluid Mechanics*, 65:711–734, 1974.
- [26] H.J.Lugt and S.Ohring. Rotating elliptic cylinders in a viscous fluid at rest or in a parallel stream. *Journal of Fluid Mechanics*, 79:127–156, 1977.
- [27] H. Honji. Starting flows past spheres, cylinder and elliptic cylinder through viscous liquid. *Reports of Research Institute for Applied Mechanics, Kyushu University*, 19:271–281, 1972.
- [28] J. How. Feedback control systems. Internet MIT OpenCourseWare, fall 2001. IP : <http://ocw.mit.edu/OcwWeb/Aeronautics-and-Astronautics/16-31Feedback-Control-SystemsFall2001/LectureNotes/>.
- [29] D.F. Jenkins and K.M. Pasino. An introduction to nonlinear analysis of fuzzy control systems. *J. Intelligent and Fuzzy Systems*, 7:75–103, 1999.
- [30] Katz and Plotkin. Low speed aerodynamics, from wing theory to panel methods. *McGraw Hill*, 1991.
- [31] B. Kosko. *Fuzzy Thinking: The New Science of Fuzzy Logic*. Hyperion Press, 1993.
- [32] P Koutsoumakos and D Shiels. Simulations of the viscous flow normal to an impulsively started and uniformly accelerated flat plate. *Journal of Fluid Mechanics*, 328:177–226, 1996.
- [33] A Leonard. Vortex methods for flow simulation. *Journal of Computational Physics*, 37:289–335, 1980.
- [34] H.J Lugt and H.J Haussling. Laminar flow past an abruptly accelerated elliptic cylinder at 45 degrees incidence. *Journal of Fluid Mechanics*, 65:711–734, 1974.
- [35] H.J Lugt and S. Ohring. Rotating elliptic cylinder in a viscous fluid at rest or in a parallel stream. *Journal of Fluid Mechanics*, 79:127–156, 1977.
- [36] E.H. Mamdani and S.Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7:1–13, 1975.
- [37] Marris. a review on vortex streets, periodic wakes, and induced vibration phenomena. *Journal of Basic Engineering*, page 185, 1964.
- [38] D. McLean. *Automatic Flight Control System*. Prentice Hall, 1990.

- [39] L Meirovitch and L.M Silverberg. Active vibration suppression of a cantilever wing. *Journal of Sound and Vibration*, 97:489–498, 1984.
- [40] Milne and Thompson (fifth edition). Theoretical hydrodynamics. *Dover Publications Inc*, 1968.
- [41] M. Mokhtari and M. Marie. *Applications de Matlab 5 et Simulink 2*. Springer, 1998.
- [42] S Monaco and D Normand-Cyrot. A unified representation for nonlinear discrete-time and sampled dynamics. *Journal of Mathematical Systems, Estimation and Control*, 7:477–503, 1997.
- [43] B.R. Morton. The generation and decay of vorticity. *Geophys. Astrophys. Fluid Dynamics*, 28:277–308, 1984.
- [44] M.J Newman. Active vibration control using a distributed controller. *Southampton University, PhD*, 1994.
- [45] K.Ohmi M.Coutanceau Ta phuoc loc and Dulieu A. Vortex formation around an oscillating and translating airfoil at large incidences. *Journal of Fluid Mechanics*, 211:37–60, 1990.
- [46] L Quartapelle and M Napolitano. Force and moment in incompressible flows. *AIAA Journal*, 21:911–913, 1983.
- [47] P.M.L Ribeiro. Geometrical nonlinear vibration of beams and plates by the hierarchical finite elements method. *Southampton University, PhD*, 1998.
- [48] L.P. Rossi. A spreading blob vortex method for viscous unbounded flow. *University of Arizona PhD*, 1993.
- [49] T. Sarpkaya. An analytical study of separated flow about circular cylinders. *Journal of Basic Engineering, ASME*, 90:511–520, 1968.
- [50] T. Sarpkaya. An inviscid model of two-dimensional vortex shedding for transient and asymptotically steady separated flow over an inclined plate. *Journal of Fluid Mechanics*, 68:109–128, 1975.
- [51] T. Sarpkaya. Vortex-induced oscillations, a selective review. *Journal of Applied Mechanics*, 46:241–259, 1979.

- [52] T. Sarpkaya. Unsteady flow about porous cambered shells. *Journal of Aircraft*, 28:502–508, 1991.
- [53] T. Sarpkaya. Vortex element methods for flow simulation. *Advances in Applied Mechanics*, 31:113–247, 1994.
- [54] T. Sarpkaya. Lift, drag, and added-mass coefficients for a circular cylinder immersed in a time-dependent flow. *Journal of Applied Mechanics*, 85:13–15, March 1963.
- [55] T. Sarpkaya and R.L Shoaff. An inviscid model of two-dimensional vortex shedding for transient and asymptotically steady separated flow over a cylinder. *AIAA journal*, 17:1193–1200, 1979.
- [56] D. Shiels. Simulation of controlled bluff body flow with a viscous vortex method. *California Institute of Technology, Pasadena, PhD*, 1998.
- [57] P.R. Spalart. Vortex methods for separated flows. *NASA Technical Memorandum 100068*, 1988.
- [58] M. Sugeno. *Industrial applications of fuzzy control*. Elsevier Science Pub. Co., 1985.
- [59] K. Takeda. Parallel discrete vortex methods for viscous flow simulation. *Southampton University PhD*, 1998.
- [60] C.A Theolis. Robust h-infinity output feedback control for nonlinear systems. *Institute for System Research, PhD*, 1994.
- [61] T.Sarpkaya. Vortex shedding from oscillating bluff bodies. *Annual Review of Fluid Mechanics*, 16:195–222, 1984.
- [62] J.J.M Van der Vegt. Calculation of force and moment in vortex method. *Journal of Engineering Mathematics*, 22:225–238, 1988.
- [63] N. Wax. *Selected papers on noise and stochastic processes*. Dover, 1954.
- [64] W.Vetterling W.Press, S.Teukolsky and B.Flannery. *Numerical Recipes in C, The Art of Scientific Computing*. Cambridge University Pres, 1992.
- [65] J.C Wu and Sankar N.L. Aerodynamic force and moment in steady and time-dependent viscous flows. *AIAA Journal*, 19:432–441, 1981.

- [66] Thomas.A. Trautzsch Zhiqiang Gao and J. Dawson. A stable self-tuning fuzzy logic control system for industrial temperature control problems. Accepted for publication in IEEE IAS Transactions, October 2000.