

University of Southampton Research Repository

Copyright © and Moral Rights for this thesis and, where applicable, any accompanying data 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 and the accompanying data cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content of the thesis and accompanying research data (where applicable) must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holder/s.

When referring to this thesis and any accompanying data, full bibliographic details must be given, e.g.

Thesis: David J. Lusher (2020) "Shock-wave/boundary-layer interactions with sidewall effects in the OpenSBLI code-generation framework", University of Southampton, Aerodynamics and Flight Mechanics group, PhD Thesis, 216 pages.

Data: David J. Lusher (2020) Dataset: Shock-wave/boundary-layer interactions with sidewall effects in the OpenSBLI code-generation framework.

DOI: <https://doi.org/10.5258/SOTON/D1302>

UNIVERSITY OF SOUTHAMPTON

**Shock-wave/boundary-layer interactions
with sidewall effects in the OpenSBLI
code-generation framework**

by

David J. Lusher

A thesis submitted for the degree of
Doctor of Philosophy

in the
Faculty of Engineering and Physical Sciences
Aerodynamics and Flight Mechanics Group

March 2020

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

AERODYNAMICS AND FLIGHT MECHANICS GROUP

Doctor of Philosophy

**SHOCK-WAVE/BOUNDARY-LAYER INTERACTIONS WITH SIDEWALL
EFFECTS IN THE OPENSBLI CODE-GENERATION FRAMEWORK**

by David J. Lusher

Numerical work on shockwave/boundary-layer interactions (SBLIs) to date has largely focused on span-periodic quasi-2D configurations that neglect the influence lateral confinement has on the core flow. The present thesis is concerned with the effect of flow confinement on Mach 2 laminar and transitional SBLI in rectangular ducts. An oblique shock generated by a deflection plate forms a conical swept SBLI with sidewall boundary layers before reflecting from the bottom wall of the domain. In the laminar cases, multiple large regions of flow-reversal are observed on the sidewalls, bottom wall and at the corner intersection of the duct. The main interaction is found to be strongly three-dimensional and highly dependent on the geometry of the duct. Comparison to quasi-2D span-periodic simulations showed that sidewalls strengthen the main interaction, with a 31% increase in the central separation bubble length for the baseline aspect ratio one configuration. Parametric studies of shock strength and duct aspect ratio were performed to find limiting behaviours. Topological features of the three-dimensional separation are identified and shown to be consistent with ‘owl-like’ separations of the first kind.

Time-dependent forcing strips are added to the laminar duct to generate disturbances for the transitional SBLI cases. The transition is observed to develop first in the low-momentum corners of the duct and spread out in a wedge shape. The central separation bubble is seen to react dynamically to oncoming turbulent spots, shifting laterally across the span. While instantaneous corner separations do occur, the time-averaged corner flow remains attached. An assessment of low-dissipative shock-capturing schemes is also performed for transitional and turbulent shock interactions. Targeted Essentially Non-Oscillatory (TENOs) schemes are found to provide improved resolution for compressible turbulence compared to conventional Weighted Essentially Non-Oscillatory (WENO) schemes.

A significant portion of the PhD project has involved the co-development of a new open-source CFD code named OpenSBLI. OpenSBLI is a Python-based code-generation framework that generates C code in the Oxford Parallel Structured (OPS) embedded Domain Specific Language (eDSL). OpenSBLI is an explicit CFD solver for the 3D compressible Navier-Stokes equations, utilizing high-order finite-difference schemes on structured meshes. The simulation code generated by OpenSBLI targets massively-parallel High-Performance Computing (HPC) architectures including CPUs, GPUs and many-core accelerator cards. The thesis documents the structure and implementation of the code. A suite of validation cases are performed to demonstrate the accuracy and correctness of the code implementation.

Contents

List of Figures	ix
List of Tables	xv
List of Example codes	xvii
Nomenclature	xix
Acronyms	xxi
Declaration of Authorship	xxiii
Acknowledgements	xxv
1 Introduction	1
1.1 Motivation for the project	1
1.2 Shock-wave/boundary-layer interactions	2
1.2.1 Background and motivation	2
1.2.2 Physical aspects of shock-induced flow separation	4
1.2.3 Laminar shock-wave/boundary-layer interactions	5
1.2.4 Recent progress and applications of SBLI	7
1.2.5 SBLI with three-dimensional sidewall effects	8
1.3 Novel programming approaches for modern computational fluid dynamics	12
1.4 Thesis objectives and outline	14
1.5 Statement of external contributors	15
1.6 Publications and conference presentations	15
1.6.1 Paper publications	15
1.6.2 Conference presentations	15
2 Numerical methods	17
2.1 Introduction	17
2.1.1 A review of high-order ENO-based shock-capturing methods	18
2.1.2 Alternative WENO formulations	19
2.2 Governing equations	19
2.3 Weighted Essentially Non-Oscillatory (WENO) schemes	21
2.3.1 Scalar reconstruction	21
2.3.2 Jiang-Shu (JS) smoothness indicators	22
2.3.3 WENO-Z smoothness indicators	23
2.3.4 Reconstruction procedure	24
2.3.5 Application to systems of equations	25
2.3.6 Characteristic decomposition	25
2.3.7 Summary of the reconstruction algorithm	26
2.4 Targeted Essentially Non-Oscillatory (TENEO) schemes	27

2.4.1	Adaptive TENO-A schemes	29
2.5	Hybrid central-WENO scheme	30
2.6	Viscous & metric derivatives	31
2.7	Time advancement	34
3	OpenSBLI: automatic code-generation for computational fluid dynamics	37
3.1	Introduction and motivation	37
3.2	Structure of an OpenSBLI problem script	39
3.2.1	List of the required components	39
3.2.2	Defining the equations	41
3.2.3	Parsing and expansion of the equations	41
3.2.4	Creating a simulation block	43
3.2.5	Creating equation classes and coordinate transformations	43
3.2.6	Numerical scheme selection	44
3.2.7	Setting the boundary conditions	45
3.2.8	Initialisation of the domain	46
3.2.9	Creating the simulation input/output	47
3.2.10	Generating a simulation code and writing it to a C file	47
3.2.11	Setting values for the simulation parameters	48
3.3	Components of the OpenSBLI implementation	49
3.3.1	Characteristic decomposition components	49
3.3.2	WENO components	50
3.3.3	TENO components	51
3.3.4	OpenSBLI objects	51
3.3.5	OpenSBLI kernels	53
3.3.6	Boundary conditions	53
3.3.7	Similarity solution for laminar boundary layers	55
3.3.8	Algorithm and code generation	56
3.4	Oxford Parallel Structured (OPS) software library	59
3.4.1	Introduction	59
3.4.2	Declaring OPS blocks, constants, and datasets	60
3.4.3	OPS stencils for relative data access	62
3.4.4	Parallel loops in OPS	62
3.4.5	Computational kernels	63
3.4.6	Writing the simulation output to file	65
3.4.7	Performance and scaling tests	65
4	OpenSBLI validation cases	69
4.1	Introduction	69
4.2	1D Shu-Osher shock/density-wave interaction	70
4.3	2D Euler wave propagation	71
4.4	2D Laminar isothermal-adiabatic channel flow	73
4.5	2D Viscous normal-shock tube	73
4.6	2D Laminar oblique shock-wave/boundary-layer interaction	76
4.7	3D Incompressible turbulent channel flow	79
4.8	3D Compressible supersonic turbulent channel flow	81
4.8.1	Sensitivity to grid refinement	81
4.8.2	Numerical scheme comparison	83
5	Assessment of low-dissipative shock-capturing schemes for transitional and turbulent shock interactions	85
5.1	Introduction	85
5.2	Supersonic Taylor-Green vortex	86

5.2.1	Problem specification	86
5.2.2	Grid refinement study	88
5.2.3	Physical effects of increasing Mach number	88
5.2.4	Shock-capturing scheme comparison	96
5.2.5	Computational cost comparison	97
5.3	Transitional shockwave/boundary-layer interaction	98
5.3.1	The effect of upstream disturbances	99
5.3.2	Scheme comparison on coarse grids	101
5.3.3	Coarser grid implicit-LES with the TENO-A scheme	103
5.4	Weight selection for TENO6 schemes	105
5.4.1	Inviscid shock reflection test case	105
5.4.2	Viscous test cases	106
5.5	Discussion	107
6	The effect of sidewall flow confinement on laminar shockwave/boundary-layer interactions	109
6.1	Introduction	109
6.2	Problem specification and duct selection	110
6.2.1	Domain specification and physical parameters	110
6.2.2	Numerical method and sensitivity to grid refinement	113
6.2.3	The role of the shock-generator and trailing expansion fan	113
6.3	Three-dimensional rectangular laminar duct SBLI with sidewall effects	117
6.3.1	Baseline duct configuration	117
6.3.2	Topology of the interaction	119
6.4	Parametric sensitivity	124
6.4.1	The effect of duct aspect ratio	124
6.4.2	Variation of incident shock-strength	128
6.4.3	Further investigation of the the trailing edge expansion fan effect	131
6.5	Discussion	133
7	The effect of sidewall flow confinement on transitional shockwave/boundary-layer interactions	135
7.1	Introduction	135
7.2	Computational method	136
7.2.1	Problem specification	136
7.2.2	Time-dependent upstream disturbances	137
7.2.3	Test case specification	138
7.3	Results	139
7.3.1	Baseline half-aspect ratio duct	139
7.3.2	Aspect-ratio one duct	142
7.4	Discussion	145
8	Conclusions	147
8.1	Summary of contributions	147
8.2	Conclusions	149
8.3	Outlook	153
8.3.1	Limitations of the current study	153
8.3.2	Scope for future work	154
A	Installation guide	157
A.1	Installation of libraries on an Ubuntu local machine	158
A.2	OPS installation	158
A.3	OpenSBLI installation and code generation	159

A.4	Compiling and running the code	160
A.5	Things to note for HPC clusters:	160
B	Description of the source code files	163
B.1	Source code directory structure	163
B.2	Contents of the source code files	165
C	SymPy functionality for OpenSBLI development	169
C.1	Pretty printing of equations: pprint	169
C.2	Creating symbolic objects: symbols	169
C.3	Off-setting an indexed object	170
C.4	Substitution of symbolic quantities	171
C.5	Left and right hand sides of equations	171
C.6	De-nesting multiple iterables	171
C.7	Accessing attributes in Python objects: __dict__	172
C.8	Using built-in SymPy functions (sin, cos, exp, ...)	172
C.9	Conditional expressions (if-else branching)	173
C.10	String representation: srepr	173
C.11	Breaking down an expression into its components: atoms/args	174
C.12	Isolating terms of a certain type: isinstance	175
C.13	Using GridVariables as temporary evaluations	175
C.14	Simplifying long expressions and reducing operation counts	175
D	Procedure for restarting simulations	177
	References	179

List of Figures

1.1	Examples of four scenarios in which shock-wave/boundary-layer interactions can occur (Babinsky and Harvey, 2011).	3
1.2	Schematic of a shock-induced separation bubble (Touber and Sandham, 2009). .	4
1.3	An example of the wall skin-friction (top) and wall pressure (bottom) distributions found in a laminar separation bubble (Yao et al., 2007).	5
1.4	Early examples of three-dimensional SBLI from Green (1970), showing (left) a photograph of oil-flow patterns in a wind tunnel and (right) a schematic of the limiting streamlines.	8
2.1	Stencil for 5th order WENO, upwind reconstruction. Nodes are cell centres. . . .	22
2.2	Staggered stencil arrangement for TENO schemes of increasing order, based on Fu et al. (2016). The flux is evaluated at the half-node $x_{i+1/2}$ location.	27
3.1	Schematic of the major components in the OpenSBLI automatic source code generation framework.	40
3.2	Characteristic decomposition classes for the shock-capturing routines.	49
3.3	Schematic of the boundary condition for a no-slip wall. The wall is located at $i = 0$ where $u, v, w = 0$ to enforce a no-slip condition. The values in the halo points (grey) are extrapolated from the interior solution points.	53
3.4	Schematic of the (left) inlet and (right) outlet boundary conditions. The shaded grey areas represents halo points outside of the physical domain.	54
3.5	An initial laminar boundary-layer profile is imposed at the start of the computational domain.	55
3.6	HDF5 meta-data for a dataset contained in an OpenSBLI output file.	62
3.7	(left) Weak and (right) strong scaling for OpenSBLI on Intel Phi Knight's landing accelerator cards, up to 64 KNLs.	66
3.8	(left) Weak and (right) strong scaling for OpenSBLI on NVIDIA Pascal P100 GPU cards, up to 64 GPUs.	67
4.1	Shu-Osher density distribution at a non-dimensional simulation time of $t = 1.8$, for WENO orders: 3Z (blue), 5Z (green) and 7Z (red). Reference fine mesh solution (continuous, black). Every third data point is displayed, with consecutive data shown in the 2x zoom inset.	70
4.2	Convergence of the L^1 norm for three orders of WENO-Z scheme.	72
4.3	Schematic of the 2D laminar mixed-wall isothermal-adiabatic channel flow. The configuration has a constant isothermal wall temperature at $y = -1$ and a zero heat-flux wall at $y = 1$. The expected velocity and temperature profiles over the channel are shown.	73
4.4	Comparison of the 2D mixed-wall condition laminar channel flow to the analytical solution in equation (4.3). Showing profiles for the (top) streamwise velocity and (bottom) temperature.	74

4.5	Schematic of the viscous shock-tube. The configuration has three adiabatic walls and a symmetry condition. At $t = 0$ the domain is initialised with a diaphragm at $x = 0.5$. The discontinuous density and pressure values are shown on each side. A normal shock propagates in the direction shown and reflects from the opposing wall.	75
4.6	Density contours for the 2D viscous shock-tube at Reynolds number of $Re = 200$. WENO-7Z scheme with $[x_0, x_1] = [600, 300]$ points.	75
4.7	Schematic of the Mach 2 laminar oblique SBLI validation case. The rectangle represents the computational domain. An initial oblique shock-wave is imposed on the upper boundary of the domain at $x = 40$	76
4.8	Time evolution of the two-dimensional SBLI separation bubble length to measure convergence.	77
4.9	Instantaneous pressure (top) and streamwise velocity (bottom) contours for the converged 2D laminar SBLI.	78
4.10	Comparison of wall pressure (left) and wall normal skin-friction (right) distributions compared to the reference solution published in Sansica et al. (2013)	79
4.11	Computational domain for the 3D turbulent channel flow cases. Periodic boundary conditions are applied in both the streamwise and spanwise directions. The domain is bounded by two isothermal no-slip walls located at $y = -1$ and $y = 1$	79
4.12	Turbulent statistics for the low-Mach channel flow compared to the spectral result of Vreman and Kuerten (2014)	81
4.13	Validation of the OpenSBLI code for the supersonic Mach 1.5 turbulent channel, based on the flow conditions of Coleman et al. (1995)	82
4.14	Numerical scheme comparison for the supersonic turbulent channel flow of Coleman et al. (1995)	84
5.1	Initial condition for the 3D Taylor-Green vortex test case. The domain is a periodic box of size $2\pi^3$. A Q-criterion of 0.25 is displayed, coloured by streamwise velocity.	87
5.2	Grid sensitivity of the $M_\infty = 1$ compressible Taylor-Green vortex problem for (a) kinetic energy and (b) total dissipation with the TENO6 scheme.	88
5.3	The effect of increasing Mach number for the compressible $Re = 1600$ Taylor-Green vortex problem on 512^3 grids, displaying (a) kinetic energy and (b) total dissipation.	89
5.4	The effect of increasing Mach number for the compressible $Re = 1600$ Taylor-Green vortex problem on 512^3 grids, displaying (a) dilatational dissipation and (b) solenoidal dissipation.	89
5.5	Formation of shock-waves during the early stages of the compressible $M_\infty = 1.25$ Taylor-Green vortex case. Showing (left) dilatation rate and (right) local Mach number at a simulation time of $t = 2$	90
5.6	Formation of shock-waves in the compressible Mach 1.25 Taylor-Green vortex case. Showing (a) discontinuous Mach profile along the $y, z = \pi$ line and (b) time evolution of the maximum Mach number in the domain.	90
5.7	Vortical structures in the $M_\infty = 1.25$ Taylor-Green vortex case at $Q = 2.5$ coloured by the x component of velocity. Showing simulation times of $t = [2, 7, 15, 20]$ in the clockwise direction starting from the upper left corner.	92
5.8	Top view of the vortical structures in the $M_\infty = 1.25$ Taylor-Green vortex case at $Q = 12.5$ coloured by velocity. Showing an $(x-y)$ view of the simulation at a time of $t = 7$	93
5.9	Magnitude of density gradients $ \nabla\rho $ for the compressible Taylor-Green vortex case at $M_\infty = 1.25$ on a 512^3 grid. Showing the flow structures at simulation times of $t = [2, 7, 15, 20]$ in (a) to (d), on the centreline $(x-y)$ plane located at $z = \pi$	94
5.10	Magnitude of density gradients $ \nabla\rho $ for the compressible Taylor-Green vortex case at $M_\infty = 1.25$ on a 512^3 grid. Showing the flow structures at simulation times of $t = [2.5, 5, 7.5, 10, 12.5]$ in (a) to (e), on the centreline $(x-z)$ plane located at $y = \pi$	95

5.11	Taylor-Green vortex $M_\infty = 0.1$ scheme comparison on coarse 256^3 grids for (a) kinetic energy and (b) enstrophy compared to a reference fine 512^3 spectral solution (blue).	96
5.12	Taylor-Green vortex $M_\infty = 1$ scheme comparison on coarse 256^3 grids for (a) kinetic energy and (b) total dissipation compared to a fine 1024^3 mesh TENO6 solution (dashed line).	96
5.13	Activation of the modified Ducros sensor after mapping to the TENO parameter C_T . (a) An instantaneous slice of streamwise velocity u for the transitional SBLI (b) The regions in which the shock sensor is active.	100
5.14	Instantaneous comparison of the oblique SBLI with and without upstream forced disturbances. (a) span-averaged wall pressure normalized by the inlet pressure, and (b) span-averaged skin friction on the wall.	100
5.15	Instantaneous density contours for the transitional SBLI. (a) x - y slice evaluated at $z = 12$ and (b) x - z slice evaluated at $y = 1$. The transition is symmetric about the centreline until $x = 215$	101
5.16	Grid sensitivity averaged over 10 periods of the forcing for the TENO6-A solution, the baseline mesh has size $(N_x, N_y, N_z) = (750, 550, 250)$. (a) Span-averaged wall pressure normalized by the inlet pressure and (b) Span-averaged skin friction on the wall. The grid is refined by 50% in each direction independently.	102
5.17	Transitional SBLI for the different shock capturing schemes. Displaying (a) instantaneous transverse velocity component w in the near-wall region at $y = 1$, $z = 0.25L_z$ showing the initial disturbance and (b) time averaged skin friction along the wall for 90 periods of the forcing.	103
5.18	Instantaneous streamwise velocity u slice above the bottom wall for (a) TENO6-A (b) TENO5 (c) TENO6 and (d) WENO-5Z. Contour levels are fixed for (a)-(b) and (c)-(d) to reflect the difference in separation length between the schemes. . .	104
5.19	Transitional SBLI with the TENO6-A scheme. The fine mesh is compared to successive coarsening in the wall normal y direction. Displaying (a) wall pressure normalised by the inlet pressure and (b) time averaged skin friction along the wall for 90 periods of the forcing.	104
5.20	Comparison of the shock resolving ability for an inviscid shock reflection at Mach 2. Pressure jump over the initial and reflected shock is shown at a height of $y = 57.5$. Showing (a) comparison of the TENO schemes to WENO-JS and WENO-Z and (b) The effect of using the optimized weightings from Fu et al. (2018) for TENO-A.	105
5.21	The effect of the optimized weights from Fu et al. (2018) on the compressible Taylor-Green vortex at Mach 1 on 256^3 grids for the (a) kinetic energy and (b) total dissipation rate, compared to the fine mesh solution.	106
5.22	The effect of the optimized weights from Fu et al. (2018) on the transitional SBLI ILES for (a) span averaged wall pressure and (b) span averaged skin friction distribution on the wall.	106
6.1	Schematic of the computational domain. An oblique shockwave is generated by deflecting the oncoming flow with a ramp angled at θ_{sg} to the freestream. No-slip isothermal wall conditions are enforced on the bottom wall, both sidewalls and on the upper surface between L_{sg} and L_{out}	110
6.2	(a) Streamwise velocity contours of the inlet laminar boundary layer profile at the intersecting corner between two no-slip walls. (b) Convergence of the centreline separation bubble length in time. One flow-through time of the freestream is equal to $t = 550$ time units.	112
6.3	Sensitivity of the centreline (a) wall pressure and (b) skin friction to grid refinement for $AR = 1$. In each direction 50% additional grid points are added independently.	114
6.4	Sensitivity of the 2D simulation (a) wall pressure and (b) skin friction to shock generator length.	115

6.5	Sensitivity of the 3D simulation with sidewalls at $AR = 1$ for the (a) wall pressure and (b) skin friction to shock generator length.	115
6.6	Baseline duct SBLI density contours ($AR = 1$). Displaying a centreline density slice ($z = 87.5$) with regions of reverse flow ($u \leq 0$) on the bottom and sidewall highlighted in dark blue.	117
6.7	(a) Centreline skin friction on the bottom wall ($y = 0$) for a duct with and without SBLI at $AR = 1$, compared to a case without sidewalls. (b) Skin friction relative to the sidewall ($z = 0$) for the duct SBLI at various y heights, showing the early streamwise onset of the corner separation.	118
6.8	Streamline patterns coloured by the shock-induced pressure jump of the main interaction for $AR = 1$. Displaying (a) $u-w$ streamlines above the bottom wall at $y = 0.14$ and (b) $u-v$ streamlines above the sidewall at $z = 0.14$	118
6.9	2D slices of dilatation ($\nabla \cdot \bar{u}$) plotted at $y = 10$ and $z = 10$ for the $AR = 1$ baseline case. The influence of the sidewalls leads to a curving of the incident shock. Negative dilatation shown in blue corresponds to the strongest regions of the incident shock. A pair of transverse shockwaves are seen to emanate from the interaction region and reflect off the sidewalls.	119
6.10	Streamlines evaluated in the $x-z$ plane at $y = 1$ for the baseline $\theta_{sg} = 2.0^\circ$ case. Streamlines are coloured by the transverse velocity component w with a constant colour background. The flow diverges at saddle points (S) at the front and back of the main recirculation region. The SBLI generates strong transverse velocity gradients that cause an ejection of the corner flow towards the centreline. Streamlines within the separation bubble are directed into two foci (F) that are symmetric relative to the centreline. Two additional foci (CF) required for topological consistency are labelled in each corner region.	120
6.11	Schematic of the ‘owl-like’ separation of the first kind adapted from Colliss et al. (2016) , based on the work of Perry and Hornung (1984) . The front saddle point (S) acts as a separating line at the start of the recirculation bubble. A focus (F) either side of the centreline signifies a tornado-like vortex that lifts fluid away from the surface. The description is consistent with the results presented in figure 6.10 for the baseline case.	121
6.12	Numerical schlieren of density gradient $\log_{10}((\nabla \rho)^2)$ showing the complex shock structure downstream of the three-dimensional SBLI at $AR = 1$. Three intersecting slices are shown at $x = 550$, $y = 15$ and $z = 15$. Notable features include: (A) Compression waves from the initial sidewall boundary layer development. (B) Main incident shock. (C) Compression waves from the start of the central separation. (D) Two conical shocks from the corner of the shock generator. (E) Expansion fan formed from the reflection of the incident shock. (F) Trailing edge expansion fan. (G) First crossing point of the reflected conical shocks. (H) Secondary reflection of the central compression waves.	122
6.13	Numerical schlieren density gradient $\log_{10}((\nabla \rho)^2)$ showing the streamwise development of the conical swept SBLI. $y-z$ slices are displayed at streamwise locations (a) $x = 225$ (b) $x = 275$ (c) $x = 325$ and (d) $x = 375$. Two conical shocks generated by the initial swept SBLI in the upper left and right corner cross through each other in (a) and (b), before reflecting off the bottom wall and opposite sidewall in (c) and (d). The dark horizontal line in (a) and (b) is the main incident shock, while in (c) and (d) it is the expansion after the interaction. The start of the trailing edge expansion fan can also be seen in the upper region of (d).	123
6.14	The effect of varying aspect ratio on the centreline bottom wall skin friction in the case of (a) narrowing and (b) widening aspect ratios. In each case the skin friction is compared to the one-to-one aspect ratio baseline duct (solid line).	125
6.15	(a) Centreline skin friction comparison of the $AR = 4$ wide duct with sidewalls to an infinite span. (b) Velocity streamline pattern above the bottom wall for the $AR = 4$ duct, coloured by pressure. Half of the span is shown $z = [0, 350]$ due to the centreline symmetry.	126

6.16	(a) The effect of duct aspect ratio on centreline streamwise separation length L_{sep} and the distance between the foci and sidewalls L_f . The dashed horizontal line denotes L_{sep} for the span-periodic case. (b) Normalized pressure on the sidewall ($z = 0$) at increasing y heights above the interaction. The ‘CS’ marker is the start of corner separation near the bottom wall.	127
6.17	Slices of x - z pressure for the $AR = 2$ case at (a) $y = 7$ above the separation bubble and at (b) $y = 70$, corresponding to 40% of the duct height above the bottom wall. In both cases the black line is the zero crossing of skin friction on the bottom wall ($y = 0$). The crossing shocks are observed to be generated by sidewall compressions from the initial swept SBLI, independent of the onset of corner separation near the bottom wall.	128
6.18	Sensitivity of the centreline (a) wall pressure and (b) skin friction to incident shock strength for the $AR = 1$ duct. The solid line represents the baseline configuration.	128
6.19	(a) u - w velocity contours evaluated at $y = 1$ above the bottom wall. Coloured by transverse velocity component w . Saddle (S) and foci (F) are highlighted as in figure 6.10. (b) Streamwise velocity at $y = 1$ above the bottom wall. The solid black line shows the $u = 0$ line to highlight regions of recirculation. Four high-speed streaks are observed downstream of the interaction in red. A darkened imprint of the conical swept shock is also visible.	130
6.20	Long domain duct ($L_x = 850$, $\theta_{sg} = 2^\circ$, $AR = 1$) demonstrating the ability of the trailing expansion fan to control the interaction. The figures correspond to shock generators with $L_{sg} = 600$ in (a,c,e) and $L_{sg} = 200$ in (b,d,f). Showing centreline density ($z = 87.5$) (a,b), skin friction on the bottom wall ($y = 0$) (c,d) and skin friction on the sidewall ($z = 0$) (e,f). The solid white line highlights the zero crossing of skin friction, enclosing regions of flow recirculation.	132
7.1	Non-dimensional computational domain for the rectangular duct. The width of the span L_z is 87.5 and 175 for the half and one aspect ratio cases respectively.	136
7.2	Time averaged skin friction distributions on the bottom wall. For the cases of (top) forced sidewalls and bottom wall (A1) and (bottom) forced sidewalls only (A2). The dashed white line encloses regions of flow recirculation.	139
7.3	Time averaged centreline normalised pressure (left) and skin friction (right) on the bottom wall of the duct. For the cases of forced sidewalls and bottom wall (red)(A1) and forced sidewalls only (black)(A2).	140
7.4	Instantaneous logarithm of density gradients $\log_{10}(\nabla \rho^2)$ between $x = [450, 800]$ for case (A1) at a height of $y = 25$. Red regions correspond to areas of high density gradients showing the shock structure downstream of the SBLI.	140
7.5	Instantaneous streamwise velocity snapshots above the bottom wall for the half aspect ratio case A2. The solid white line represents the $u = 0$ contour enclosing regions of recirculation. The central recirculation bubble shifts laterally due to oncoming intermittent turbulent spots.	141
7.6	Time averaged skin friction distributions on the bottom wall. For the cases of (top) forced sidewalls and bottom wall (B1) and (bottom) forced sidewalls only (B2). The dashed white line encloses regions of flow recirculation.	142
7.7	Time averaged centreline normalised pressure (left) and skin friction (right) on the bottom wall of the duct. For the $AR = 1$ cases of forced sidewalls and bottom wall (red)(B1) and forced sidewalls only (black)(B2).	143
7.8	An x - z plane of instantaneous pressure evaluated at a height of $y = 105$ for $AR = 1$. The main incident shock is visible at $x = 450$ and is seen to be strengthened by compressions from both of the sidewalls. The sidewall compressions coalesce into shock-waves that cross at $x = 475$	144

-
- 7.9 (a) A $z - y$ plane of instantaneous pressure evaluated at a streamwise location of $x = 450$ for $AR = 1$. Two high pressure regions are observed that originate from the upper corners of the duct at the start of the shock generator. (b) Dilatation rate ($\nabla \cdot \vec{u}$) at a height of $y = 105$ for $AR = 1$. Negative and positive regions of dilatation correspond to shock-waves and expansions respectively. (c) Turbulent kinetic energy at a height of $y = 105$ for $AR = 1$. The sonic line ($M = 1$) is shown in dashed black. 145
- 7.10 Streamwise pressure profiles at a height of $y = 105$ for (left) $z = 20$ and (right) $z = 87.5$ at $AR = 1$, normalised by the inlet pressure. Near the sidewall ($z = 20$) there is an expansion directly after the incident shock. At $x = 475$ on the centreline ($z = 87.5$) the crossing point in figure 7.8 is observed as a pressure peak downstream of the incident shock. 146

List of Tables

3.1	Runtime comparison for 500 iterations of a 3D SBLI test case. The Ivy-Bridge 24 CPU cores (MPI) platform is taken as the baseline time.	66
4.1	Error in the L^∞ norm for 3rd, 5th and 7th WENO-Z schemes, and convergence rates for smooth flows.	72
4.2	Simulation parameters for the two-dimensional laminar SBLI.	76
4.3	Incident shock-jump conditions for the Mach 2 two-dimensional laminar SBLI.	77
5.1	Runtime for 1000 iterations of the TGV problem at Mach 1 on a 256^3 grid. CPU: Intel Skylake node (40 cores @ 2 GHz, 40 MPI, Intel 17.0 -O3 -fp-model fast). GPU: 1x NVIDIA Volta 16GB V100 (CUDA 9.0, nvcc -O3). The relative computational cost is compared to the WENO5-JS scheme.	97
6.1	Domain specification and grid distributions. Aspect ratio (AR) is defined as the ratio of duct width to height (L_z/L_y). A one-to-one aspect ratio is taken as the baseline configuration.	111
6.2	Simulation parameters for the three-dimensional laminar SBLI cases in rectangular ducts.	111
6.3	Sensitivity of the centreline separation to increasing L_{sg} for two-dimensional SBLI without sidewalls. Increasing L_{sg} causes the trailing edge expansion fan to impinge further downstream on the bottom wall. Percentage increase is relative to the shortest $L_{sg} = 200$ case.	114
6.4	Sensitivity of the centreline separation to increasing L_{sg} for three-dimensional SBLI at $AR = 1$ with sidewall effects. Increasing L_{sg} causes the trailing edge expansion fan to impinge further downstream on the bottom wall and also modifies the pressure distribution downstream of the interaction. Percentage increase is given relative to the shortest $L_{sg} = 200$ case.	115
6.5	The effect of aspect ratio on the baseline $\theta_{sg} = 2^\circ$ shock generator case. Comparison is also made to a span-periodic simulation without sidewalls demonstrating the strengthened three-dimensional interaction. Separation length is shown as a percentage of the one-to-one aspect ratio sidewall case. L_{sep} is the separation length along the centreline and L_f is the distance between the foci and the sidewalls.	124
6.6	Reduction in streamwise separation length L_{sep} of the main interaction with decreasing incident shock strength. Comparison is made to the centreline skin friction for the baseline $\theta_{sg} = 2.0^\circ$ case with $AR = 1$	129
7.1	Overview of the four computational test cases with different forcing configurations. The prefixes (A) and (B) refer to cases with half and one aspect ratio ducts respectively.	138

List of Example codes

3.1	Defining the governing equations in OpenSBLI.	41
3.2	Defining the evaluation of constituent relations in OpenSBLI.	42
3.3	Parsing and expansion of equations in OpenSBLI.	42
3.4	Creating a simulation block in OpenSBLI.	43
3.5	Using the SimulationEquations and ConstituentRelations classes in OpenSBLI. .	43
3.6	Generating metric transformations in OpenSBLI.	44
3.7	Spatial and temporal scheme selection in OpenSBLI.	44
3.8	Selecting boundary condition classes in OpenSBLI.	45
3.9	Creating initial conditions in OpenSBLI.	46
3.10	HDF5 input/output file writing in OpenSBLI.	47
3.11	Creating an OPS C code in OpenSBLI.	47
3.12	Setting numerical values of the simulation parameters in OpenSBLI.	48
3.13	Creating and off-setting the location of indexed objects in OpenSBLI.	52
3.14	Creating a computational kernel in OpenSBLI.	53
3.15	Generating sequential ‘for loops’ in the output C code.	56
3.16	An example of OPS C code generation in OpenSBLI.	57
3.17	The corresponding stencil declaration in the OPS C code.	57
3.18	Defining custom C code printer functions in OpenSBLI.	58
3.19	Pre-computation and printing of rational constants in OpenSBLI.	58
3.20	Declaration of simulation constants in OPS.	60
3.21	Initialising OPS and passing the constant declarations.	60
3.22	Declaring a simulation block in OPS.	60
3.23	Declaring a dataset in OPS.	61
3.24	Declaring a dataset in OPS from a restart data file.	61
3.25	Stencil declaration and OPS partitioning of the domain.	62
3.26	An example of calling a parallel loop region in OPS.	63
3.27	A user kernel to calculate a central derivative in OPS.	63
3.28	An example of local and global storage in an OPS user kernel.	64
3.29	Creating a simulation output HDF5 file.	65

Nomenclature

a	Non-dimensional speed of sound
A_0	Forcing amplitude
A_i, B_i	Low-storage Runge-Kutta coefficients
AR	Duct aspect ratio
C_f	Skin-friction coefficient
C_T	Threshold value for the TENO scheme stencil selection
$\hat{f}_{i+1/2}$	Half-node flux reconstruction
$(F, G, H), (F_v, G_v, H_v)$	Inviscid and viscous fluxes in vector form
L_f	Length between the separation bubble foci
L_{sep}	Streamwise separation bubble length
(L_x, L_y, L_z)	Domain lengths per direction
M	Local Mach number
M_∞	Freestream Mach number
(N_x, N_y, N_z)	Number of grid points per direction
Pr	Prandtl number
p_w, T_w	Wall pressure and temperature
q_k	Heat flux
Q	Q-criterion
Re	Reynolds number
$Re_{\delta_0^*}$	Reynolds number based on the inlet displacement thickness
Re_τ	Reynolds number based on friction velocity
s	Grid stretching factor
t	Non-dimensional time unit
T_s	Sutherland's law temperature
T_∞	Reference freestream temperature
T_{aw}	Adiabatic wall temperature
(u, v, w)	Velocity components
(u_0, u_1, u_2)	Velocity components in index notation
(x, y, z)	Spatial coordinates
(x_0, x_1, x_2)	Spatial coordinates in index notation
x_{sg}	Location of the shock-generator plate
α_1, α_2	Upper and lower bounds of the TENO-A shock threshold
γ	Ratio of specific heat capacities
δ_0^*	Inlet boundary displacement thickness
Δ_s	Similarity solution scale factor

Δt	Non-dimensional time-step
$(\Delta x, \Delta y, \Delta z)$	Uniform grid spacing per direction
ϵ	Small parameter to avoid division by zero in the shock-capturing schemes
$\epsilon^T, \epsilon^S, \epsilon^D$	Total, solenoidal and dilatational dissipation
θ_{sg}	Shock-generator deflection angle
κ	Von-Karman constant
(ξ, η, ζ)	Curvilinear spatial coordinates
$ \nabla \rho $	Magnitude of density gradients
$\hat{\rho}, \hat{u}, \hat{a}$	Roe-averaged quantities
$(\rho, \rho u, \rho v, \rho w, \rho E)$	Non-dimensional conservative flow variables
(ρ, u, v, w, p, T)	Non-dimensional primitive flow variables
$\tau_{i,j}$	Stress tensor
τ_K	TENO global smoothness indicator
τ_w	Wall shear stress
ϕ_l, ϕ_m	Random phases in forcing terms
Φ	Ducros shock sensor
χ_r, γ_r	TENO smoothness measures and non-linear weights
ω_m	Forcing frequencies
ω_r, d_r, σ_r	WENO non-linear weights, ideal weights and smoothness indicators

Acronyms

API	Application Programming Interface
CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
DNS	Direct Numerical Simulation
DSL	Domain Specific Language
EDSL	Embedded Domain Specific Language
GPU	Graphical Processing Unit
HDF5	Hierarchical Data Format 5
HPC	High Performance Computing
ILES	Implicit Large Eddy Simulation
LES	Large Eddy Simulation
MPI	Message Passing Interface
OPS	Oxford Parallel Structured
RK	Runge-Kutta scheme
SBLI	Shock-wave/boundary-layer interaction
SSP	Strong Stability Preserving
TENO	Targeted Essentially non-Oscillatory
TGV	Taylor-Green Vortex
TVD	Total Variation Diminishing
WENO	Weighted Essentially non-Oscillatory

Declaration of Authorship

I, David J. Lusher, declare that this thesis entitled Shock-wave/boundary-layer interactions with sidewall effects in the OpenSBLI code-generation framework and the work presented in it are my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Either none of this work has been published before submission, or parts of this work have been published as: David J. Lusher (2018-2020).

Signed:

Date:

Acknowledgements

I would like to express my gratitude to Prof. Neil D. Sandham for supporting my development as a researcher. The encouragement and guidance I have received have been fundamental to the progress made during the project. Thank you to Dr. Satya P. Jammy for the time we spent coding together, and his role as a mentor during the early stages of the project. I would like to thank Dr. Gary N. Coleman for giving me the opportunity to visit NASA Langley as a visiting researcher. The discussions and feedback provided on the code have been very beneficial. Thank you to Dr. Gihan R. Mudalige and Dr. István Z. Reguly for the development and continued support of the OPS library.

To my friends James Gill, Mischa Wiedouw, Elisabeth Mitchell and Li Shao: thank you for your support, and for the good times we have shared over the years. I hope you get better soon James.

The PhD project was funded by an EPSRC Centre for Doctoral Training grant (EP/L015382/1). The work has been performed using resources provided by the ‘Cambridge Service for Data Driven Discovery’ (CSD3, <http://csd3.cam.ac.uk>) system operated by the University of Cambridge Research Computing Service (<http://www.hpc.cam.ac.uk>) funded by EPSRC Tier-2 capital grant EP/P020259/1. I would also like to acknowledge the use of the IRIDIS 5 High Performance Computing Facility, and associated support services at the University of Southampton. Additional computing time was provided by the UK Turbulence Consortium (EPSRC grant EP/L000261/1). A research visit was funded by the National Aeronautics and Space Administration, Langley Research Center under Cooperative Agreement No.80LARC17C0004, awarded to the National Institute of Aerospace.

Chapter 1

Introduction

1.1 Motivation for the project

Shockwave/boundary-layer interactions (SBLI) play an important role in the study of high-speed compressible gas dynamics. The ubiquity of SBLIs in aeronautical flows of practical interest is well established ([Dolling, 2001](#); [Gaitonde, 2015](#)), posing considerable challenges for high-speed aircraft design. SBLIs can occur in both internal and external flow configurations, comprised of a complex coupling between inviscid and viscous effects. An incident shock in an internal flow can interact with multiple surfaces and can result in a complex and dynamic shock system. Flow separation and unsteadiness are major concerns in applications such as supersonic engine intakes, where non-uniform flow entering the compressor can lead to variable heat transfer rates and pressure losses. Detrimental effects include reduced engine efficiency and an increase in the structural fatigue of components. In severe cases, SBLIs lead to a full un-start of the engine. The adverse pressure gradient applied by an impinging shock causes a thickening of the target boundary layer, and for sufficiently strong shocks, a separation of the flow will occur. For a given strength of incident shock, the susceptibility of the boundary layer to separate is largely dependent on the upstream state of the boundary layer ([Babinsky and Harvey, 2011](#)). Turbulent boundary layers are most capable of resisting flow separation: the higher mixing rates effectively energise the boundary layer and stave off stagnation by transferring high-momentum fluid towards the wall. Laminar boundary layers separate far more easily than their turbulent counterparts, with flow separation observed for shocks weaker than required for incipient separation of a turbulent boundary layer.

Current concerns over the environmental impact of aircraft has contributed to a renewed interest in laminar aerodynamics, taking advantage of the lower skin-friction drag of laminar boundary layers. The present work focuses on numerical simulation of laminar and transitional SBLI for internally confined rectangular duct flows, which are simplified forms of supersonic engine intakes. An initial oblique shockwave interacts with boundary layers on both the sidewalls and bottom wall of the duct, resulting in multiple regions of three-dimensional reverse flow. While many real-world applications will be fully turbulent, laminar and transitional solutions provide useful comparisons to wind tunnel experiments where small-scale models are investigated at lower Reynolds numbers. Furthermore, shock and expansion wave patterns are easier to distinguish in

the absence of turbulence and the mechanism of transition can be investigated in laminar SBLI. Laminar flows can be used as a basis for stability analysis to gain insight into the mechanism of transition to turbulence. As noted by the laminar-transitional SBLI work of [Giepman et al. \(2016\)](#), experimental techniques such as particle image velocimetry (PIV) can suffer from seeding issues with laminar boundary layers. Numerical simulations are well placed to complement the existing experimental literature, offering additional insight into the complex flow features of laterally confined SBLI.

The underlying aim of the project is to investigate laminar and transitional shock-wave/boundary-layer interactions in the presence of sidewalls. The majority of the research in this area to date has been limited to either two-dimensional or span-periodic three-dimensional flow configurations. While span-periodicity provides a relatively straightforward numerical problem and allows the SBLI mechanism to be investigated, it neglects the complex three-dimensional phenomena found in real world applications that are bounded by physical surfaces on multiple sides. Computation of supersonic fluid flows poses notable challenges for numerical modelling, the most pertinent of these is the need for numerical schemes that can handle discontinuities in the form of shock-waves. As such, robust shock-capturing methods from the family of Weighted Essentially non-Oscillatory (WENO) ([Shu, 1997](#)) and Targetted Essentially Non-Oscillatory (TENOWENO) ([Fu et al. \(2016, 2017\)](#)) schemes have been implemented in the OpenSBLI code developed during this project.

The OpenSBLI framework ([Jacobs et al., 2017](#); [Lusher et al., 2018b](#)) is a recent approach to computational fluid dynamics (CFD), comprised of a Python-based code-generation system that generates a simulation code in the C programming language. The resulting C code is compliant with the Oxford Parallel Structured (OPS) software domain-specific language [Mudalige et al. \(2014, 2019\)](#), for massively-parallel execution of the code on a range of CPU and GPU-based computational architectures.

1.2 Shock-wave/boundary-layer interactions

1.2.1 Background and motivation

The study of shock-wave/boundary-layer interactions is important in the context of aeronautical design, with common applications including but not limited to: transonic aerofoils, supersonic engine intakes, and over-expanded nozzles ([Clemens and Narayanaswamy, 2014](#)). SBLIs have been an active area of research for the past 70 years ([Dolling, 2001](#)). The review by [Dolling \(2001\)](#) highlighted the need for an improved understanding of unsteady shock oscillations and complex three-dimensional (3D) effects. In the two decades since that review, significant improvements have been made in available computational power, which is now sufficient to allow unsteady 3D SBLI effects to be investigated numerically. A literature review of the progress in numerical modelling of SBLIs is given in the following sections of this chapter.

As illustrated in Figure 1.1, shock-wave/boundary-layer interactions are typically characterised into four main scenarios ([Babinsky and Harvey, 2011](#)): (a) an incident oblique shock reflecting from a boundary-layer over a solid surface, (b) a shock with induced separation formed by a

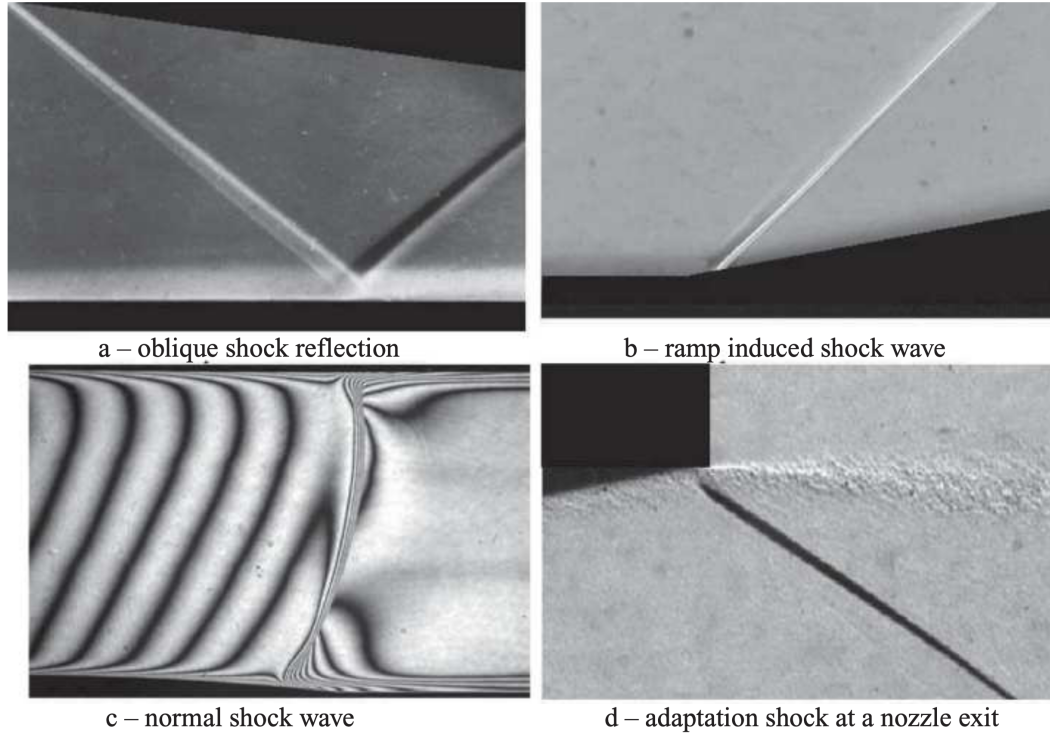


FIGURE 1.1: Examples of four scenarios in which shock-wave/boundary-layer interactions can occur (Babinsky and Harvey, 2011).

compression ramp, (c) a normal shock from a supersonic flow being reduced to subsonic conditions by a back pressure, and (d) flow deflection adapting to an abrupt pressure jump. Transonic SBLIs can be found on the wings of transonic aircraft, inside compressor blade cascades in turbines, and in supersonic intakes. They are usually characterised by the interaction of a normal shock with a boundary-layer, followed by a region of subsonic flow. One of the foremost issues resulting from transonic SBLIs is the unsteady oscillation of the shock leading to buffeting and undesirable flight mechanics. Significant efficiency losses also occur on transonic aerofoils, with the total pressure drop from the interaction causing increased wave drag (Babinsky and Harvey, 2011). For the majority of supersonic engine intakes, a primary design goal is efficient compression and retardation of the incoming air to suitable subsonic engine operating conditions. While this can be achieved via a normal shock, the subsequent drop in stagnation pressure is too large. Instead, a series of successive oblique shocks leads to reduced losses and improved engine performance. One further problem is the non-uniformity that is introduced to the flow from shock induced separation of the boundary-layer. Flow control is often applied (Xiang and Babinsky, 2018; Grébert et al., 2018) to suppress boundary-layer separation and avoid performance losses and load variability on the engine.

This thesis focuses on the reflection of an oblique shock on a boundary-layer as in Figure 1.1 (a), commonly found in aeronautical applications where supersonic flow is turned by external fins or changes in geometry. Additionally, the inclusion of sidewall effects creates complex three-dimensional phenomena, as the incident shock is swept down the sidewall boundary-layer. Central to this study is the relationship between the main separation at the foot of the reflected shock, and the corner separation caused by the swept shock along the sidewall.

1.2.2 Physical aspects of shock-induced flow separation

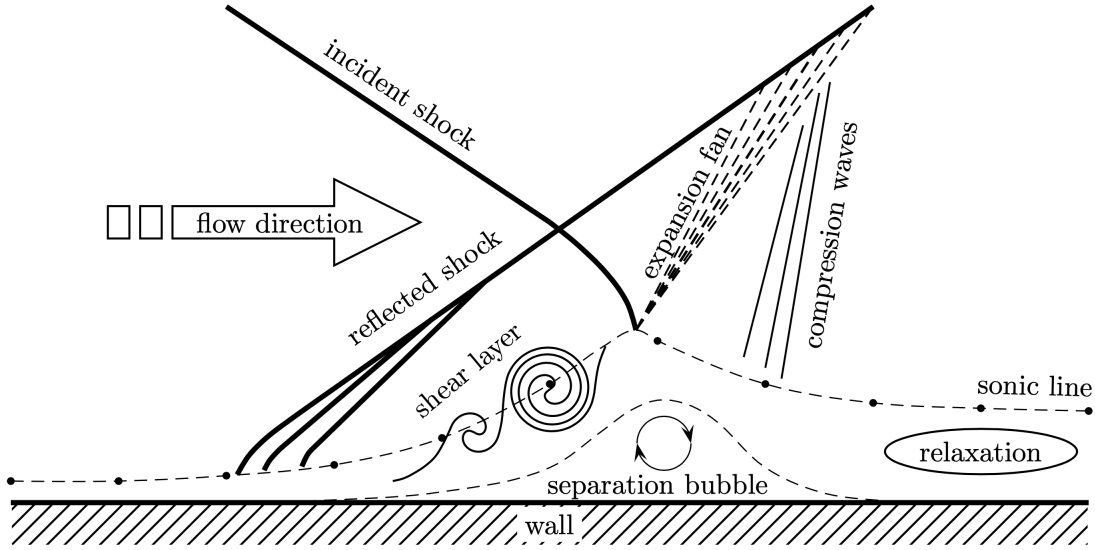


FIGURE 1.2: Schematic of a shock-induced separation bubble (Touber and Sandham, 2009).

The effect of an oblique shock-wave impinging on a boundary-layer is represented in the schematic in Figure 1.2. The adverse pressure gradient around the impingement point leads to a thickening of the boundary layer (Dolling, 2001; Gaitonde, 2015). Static pressure is known to be uniform across an undisturbed boundary layer, but the rise in pressure due to an incident shock-wave causes a corresponding retardation and potential reversal of the flow velocity (Babinsky and Harvey, 2011). If the shock is sufficiently strong, separation occurs and a region of recirculating detached flow is observed beneath the impingement point. At the start of the separation bubble upstream of the incident shock, compression waves are emitted due to the sudden increase of curvature of the thickened boundary-layer, which coalesce into a reflected shock-wave. Directly at the impingement point an expansion fan is observed as the flow turns over the apex of the separation bubble. The expansion fan has the effect of thinning the boundary-layer to give the separated region an approximate triangular shape. Downstream of the central interaction a further compression occurs as the boundary-layer reattaches to the solid wall.

Figure 1.3 (a) highlights the structure of a laminar separation bubble in terms of the streamwise variation of skin-friction coefficient C_f along the wall. The figure is of a Mach 2 laminar SBLI taken from the work of Yao et al. (2007), based on previous numerical (Katzner, 1989; Wasistho, 1997) and experimental studies (Hakkinen et al., 1959) at the same flow conditions. The effect of the shock reflection can be seen from the dip in skin-friction coefficient when compared to an undisturbed laminar boundary-layer (thick black line). Two troughs are observed either side of the impingement point, inside the region of separated flow within the bubble. A larger magnitude minima is seen in the trough downstream of the central point, where the flow recirculation is at a maximum. Ultimately the factors determining the shape and extent of the separation bubble are the incident shock-strength, boundary-layer thickness, and the laminar/transitional/turbulent state of the upstream boundary-layer. As a general rule, turbulent boundary-layers are far less prone to separate, with laminar SBLIs exhibiting separation even for relatively weak shock strengths.

Similarly, Figure 1.3 (b) shows the streamwise variation of wall pressure normalised by the inlet pressure. At the start of the separation bubble at $x = 100$ a significant rise in pressure is seen due to the compression waves resulting from the thickened boundary-layer. The middle part of the pressure distribution features a plateau, before a secondary pressure rise occurs at the reattachment point. The gradual rise in pressure near the separation and reattachment points is characteristic of a shock impinging on a boundary-layer. The thick black line in the same figure represents the ideal inviscid pressure jump over a reflected shock. In the case of a separation bubble undergoing transition, if the transition point is inside the separation bubble, a much steeper pressure jump is observed in the reattachment region.

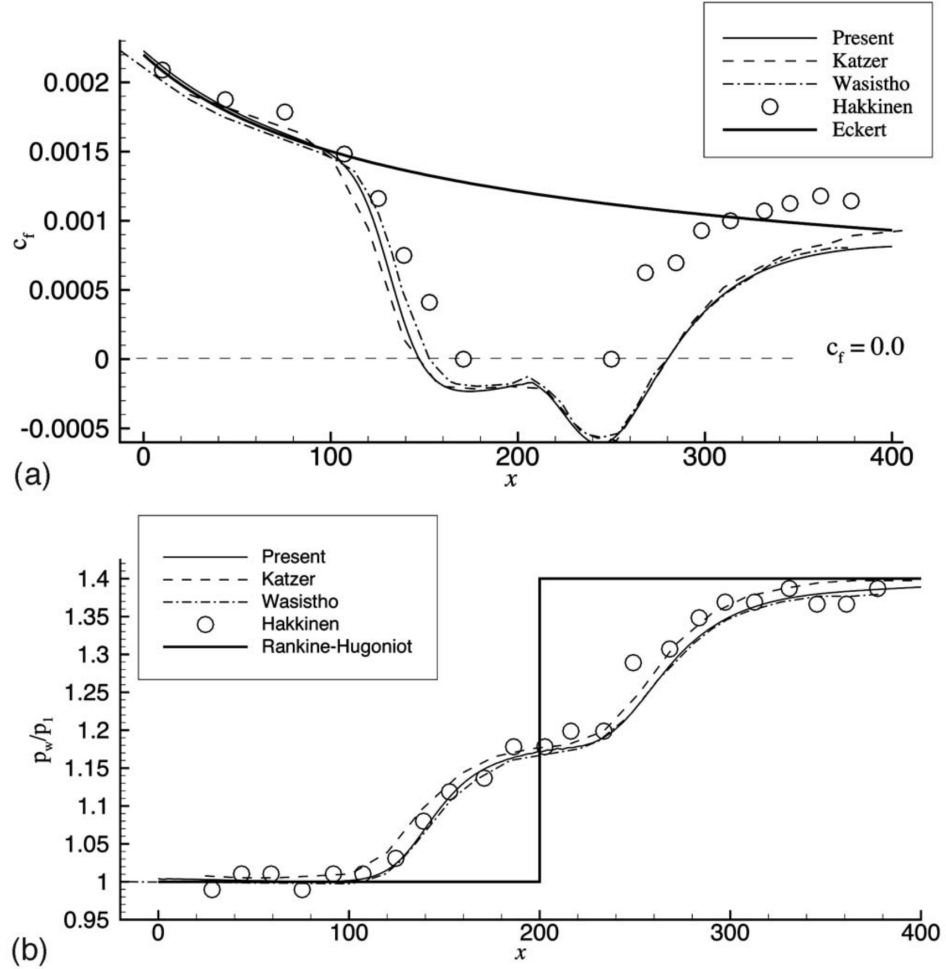


FIGURE 1.3: An example of the wall skin-friction (top) and wall pressure (bottom) distributions found in a laminar separation bubble (Yao et al., 2007).

1.2.3 Laminar shock-wave/boundary-layer interactions

Two-dimensional laminar SBLI is a largely well-understood phenomenon and has historically been treated by a range of both theoretical and numerical approaches (Adamson and Messiter, 1980). An important numerical study on laminar oblique-SBLI was carried out by Katzer (1989), based on the earlier experiments of Hakkinen et al. (1959). Laminar SBLI were simulated over a flat plate for a range of Mach numbers from 1.4 to 3.4, with the results agreeing well with

predictions from free interaction theory. The length of the separation bubble was found to be linearly dependent on the incident shock strength. A recent review of this study by [Sansica et al. \(2013\)](#) examined some of the issues with the original work via two-dimensional direct numerical simulation (DNS). Among the findings, it was noted that large integration times are required to obtain fully converged solutions for laminar separation bubbles. Previous studies were shown to have terminated the simulations too early, before the separation bubble had been allowed to fully develop. As an extension to previous work, the response of the 2D laminar separation bubble to an applied forcing was also investigated, with low-frequency unsteadiness observed. Comparison to the solution published by [Sansica et al. \(2013\)](#) is given in this thesis in section 4.6, serving as a comparison of the SBLI and OpenSBLI codes developed at the University of Southampton.

A combined numerical/experimental study on a nominal two-dimensional laminar SBLI was performed by [Degrez et al. \(1987\)](#) at Mach 2.15. It was reported that experimental configurations with an aspect ratio greater than 2.5 were required to achieve two-dimensional behaviour of the SBLI. More recent work has focused on instability and the transition mechanisms, often motivated by the widely reported low-frequency unsteadiness also present in turbulent SBLI ([Clemens and Narayanaswamy, 2014](#)). A numerical study using the same conditions as [Katzer \(1989\)](#) at Mach 2 was carried out by [Sivasubramanian and Fasel \(2015\)](#) with and without upstream disturbances. The disturbances were found to be strongly amplified by the laminar separation bubble and at higher shock strengths the flow transitioned to turbulence downstream of the bubble. Both high and low frequency unsteadiness was observed. The high frequency component was attributed to vortical shedding at the reattachment point during the breakdown.

Efforts to characterise the SBLI process and its instabilities analytically often involves the use of global linear-stability analysis. In the work of [Robinet \(2007\)](#), the unsteady characteristics of a laminar SBLI were investigated through numerical simulation and stability analysis. A span-periodic 3D DNS of laminar oblique SBLI was performed for a 3.75° deflection angle at Mach 2.15. At this starting shock strength the laminar SBLI was 2D and steady in nature, but quickly became 3D and unsteady at higher shock strengths. The shock strength required to obtain unsteady 3D behaviour within the separation bubble was investigated, with unsteady self-sustained low-frequency dynamics observed within the interaction region. Linear global stability analysis identified a stationary three-dimensional mode in the linearly globally unstable flow. A follow-up study by [Guiho et al. \(2016\)](#) focused purely on a 2D laminar oblique SBLI flow configuration. A selection of shock angles were simulated for Mach numbers between 2.1 and 3.0, with global stability analysis used to identify and catalogue global modes.

In the hypersonic regime [Dwivedi et al. \(2017\)](#) performed DNS of a Mach 5.92 laminar SBLI. Above a critical shock angle the flow became three-dimensional and unsteady, with the downstream region being found to support significant growth of perturbations starting at the reattachment point. Further work with the same flow conditions ([Hildebrand et al., 2018a,b](#)) studied transient growth of disturbances and the instability mechanism within the laminar recirculation bubble itself. Above a critical shock angle a self-sustaining process was identified using global stability analysis. The instability was attributed to streamwise vortices created within the recirculation bubble that redistribute momentum normal to the wall and develop into elongated streaks downstream of reattachment. A similar study by [Lee and Gross \(2019\)](#) confirmed the existence of a global mode and investigated the effect of sweep on the configuration. Travelling

oblique waves were observed due to a crossflow instability. Higher sweep angles were shown to have a stabilising effect on the disturbances, resulting in a fully steady flow.

1.2.4 Recent progress and applications of SBLI

Relatively early examples of numerical simulations being applied to more complex SBLI include [Adams \(2000\)](#), for a turbulent boundary-layer over an 18° compression ramp at Mach 3, and an oblique shock reflection for a turbulent boundary layer at Mach 2.25 by [Pirozzoli and Grasso \(2006\)](#). In the second study, shock-capturing was performed with a 7th order WENO scheme, and compact 4th order differencing for viscous terms. Shedding of large scale structures was observed near the impingement point, generating acoustic disturbances that caused flapping of the reflected shock. The low-frequency unsteady motion within the separation region was attributed to this acoustic resonance. One issue intrinsic to shock-capturing schemes is the excessive numerical dissipation they induce in smooth regions of the flow where it is not desired, damping small-scale turbulent structures. This creates a requirement for very fine grids to achieve acceptable resolution of small-scale flow structures, exacerbating the already high computational cost of DNS. The issues associated with shock-capturing schemes are reviewed in section 2.1.1, with low-dissipative shock-capturing schemes assessed in chapter 5 of this thesis. An example of ramp-induced SBLI in the hypersonic regime was given by [Ritos et al. \(2018\)](#), for an LES of a 33° compression ramp at Mach 7.2. A 9th order WENO scheme was used for convection, with standard 2nd order central differencing used for viscous terms. A high level of asymmetry was observed in the turbulence at the edge of the boundary-layer and inside the separation bubble. The work of [Grisham et al. \(2017\)](#) investigated incipient separation for ramp-induced SBLIs, via numerical solution of both the triple-deck and governing equations. Good qualitative agreement was found between triple-deck theory and the numerical solution, providing data on the physical ramp angle required for incipient separation as a function of Mach and Reynolds numbers.

A central theme for research into SBLIs is the focus on the source of low-frequency unsteadiness within the separated region, and the mechanisms that sustain it. Low-frequency unsteady motion is found to be two orders of magnitude lower than typical frequencies of a turbulent boundary-layer, and can have detrimental effects on the safety and performance of aircraft. Recent examples of numerical studies investigating the unsteadiness in SBLI includes [Wu and Martin \(2008\)](#), in which DNS was performed for a turbulent boundary-layer over a compression ramp at Mach 2.9. It was observed that the motions of the shock and the separation point were both correlated with the motion of the point of reattachment, suggesting there is a significant downstream influence on the interaction. The experimental work by [Ganapathisubramani et al. \(2007\)](#) of a Mach 2 compression ramp attributed the unsteadiness to elongated structures in the upstream boundary-layer. In contrast, the LES of [Touber and Sandham \(2009\)](#) showed the existence of low-frequency unsteadiness even in the absence of large-scale upstream structures, suggesting the motion is intrinsic to the interaction itself. A follow-up study ([Touber and Sandham, 2011](#)) expanded on this assertion, with the development of a low order stochastic model of the unsteadiness. In the work of [Sansica et al. \(2016\)](#), it was shown that a low-frequency unsteadiness is present even in the case of a laminar SBLI containing no upstream fluctuations. [Bonne et al. \(2019\)](#) applied resolvent analysis to a steady Reynolds-Averaged Navier-Stokes (RANS) solution, for the case of a two-dimensional transitional SBLI at Mach 1.6. A feedback

mechanism was proposed for the observed low-frequency dynamics. It was argued that a resonance occurs due to amplification of the separated shear-layer of the separation bubble, forced by a backward travelling density wave. In the fully turbulent SBLI regime, [Nichols et al. \(2017\)](#) applied Dynamic Mode Decomposition (DMD) coupled with global stability analysis to a Large-Eddy Simulation (LES) database at Mach 2.28. The findings supported previous claims that proposed that low-frequency unsteadiness is inherent to the interaction zone. It was concluded however that some form of upstream forcing is required to excite the system.

Recent experimental studies on laminar-transitional SBLI include [Giepman et al. \(2015, 2018\)](#), and [Diop et al. \(2019\)](#). In the experiments of [Giepman et al. \(2015, 2018\)](#), a range of shock impingement locations were investigated for Mach numbers in the range $M = [1.6, 2.3]$. All experiments were performed with high-resolution PIV in a wind tunnel with a partial-span shock generator. For the laminar impingement locations long, triangular recirculation bubbles were observed, with a linear dependence of shock strength on the distance between the separation point and the top of the bubble. The largest separation bubbles were recorded for the purely laminar interactions, while a significant shortening of the separation length was observed when the boundary layer was in a transitional state. The dependence on the upstream boundary layer state has led to studies on optimal tripping methods to obtain a transitional state close to the SBLI. The experiments of [Giepman et al. \(2016\)](#) and the complimentary numerical study by [Quadros and Bernardini \(2018\)](#) investigated tripped transition of laminar SBLI at $M = 1.7$. Both cases confirmed that for a given shock strength the size of the recirculation bubble was highly dependent on the incoming boundary layer state. The experimental work showed that the recirculation region could be removed entirely by placing a trip close to the interaction. Although this grants control of the separation, the trade-off is a substantially thicker boundary layer and increased skin-friction drag downstream of the interaction.

1.2.5 SBLI with three-dimensional sidewall effects

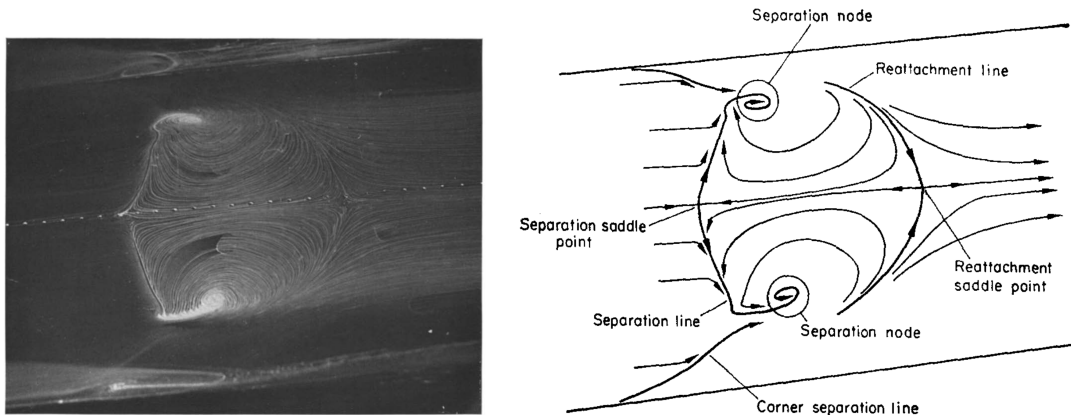


FIGURE 1.4: Early examples of three-dimensional SBLI from [Green \(1970\)](#), showing (left) a photograph of oil-flow patterns in a wind tunnel and (right) a schematic of the limiting streamlines.

Much of the early work on three-dimensional effects for SBLI was reviewed by [Green \(1970\)](#), in which it was stated that almost all SBLI cases of practical importance contain substantial three-dimensionality. This statement was motivated by the surface oil-flow pattern of a

three-dimensional separation bubble shown in figure 1.4 (left). The observed complex three-dimensional flow patterns cannot be explained by two-dimensional analysis. The corresponding schematic in figure 1.4 (right) highlights some of the main features of the interaction, including separation and reattachment lines, plus a pair of foci referred to as separation nodes. Further examples of early experimental studies on this topic include the works of [Reda and Murphy \(1973a,b\)](#). A significant departure from two-dimensional behaviour was observed in [Reda and Murphy \(1973a\)](#), for configurations that had been specifically designed to produce two-dimensional flow fields. The demonstrated sidewall and corner effects brought into question the two-dimensionality of previous studies. The follow up study ([Reda and Murphy, 1973b](#)) expanded on these findings, by showing that isolating the interaction with side-plates substantially modified the interaction region. Separated flow patterns of the type shown in figure 1.4 (right) were denoted ‘owl-like’ separations by [Perry and Hornung \(1984\)](#), and further formalised using critical-point theory in [Chong et al. \(1990\)](#) and [Perry and Chong \(1987\)](#).

Despite the progress made in understanding certain features of SBLI ([Dolling, 2001](#); [Gaitonde, 2015](#)), the infinite-span (quasi-2D) assumption persists in much of the numerical literature as a way of reducing computational complexity. For internally bounded flows this may not be a valid assumption, as lateral confinement leads to multiple boundary layers for the shock to interact with. The modified interaction may be highly three-dimensional and strongly influenced by the geometry of the duct. The recent review of [Clemens and Narayanaswamy \(2014\)](#) reaffirmed the need for a better understanding of three-dimensional effects for SBLIs. Central to this aim is the addition of sidewall effects that are present in most realistic geometries. A second concern is the need for more extensive parametric studies to investigate the universality of the findings. Internal flows contain substantial complexity, as multiple shocks interact with each other and reflect between the walls of the enclosed environment. At the corner between solid walls, conical shocks form and interact with the initial incident shock. The presence of a sidewall is known to cause a curving of the incident shock compared to quasi-2D cases ([Wang et al., 2015](#)), causing a strengthening of the interaction. As previously mentioned for supersonic engine intakes, the efficient retardation of the upstream flow down to subsonic conditions is of great importance for optimal operating conditions. After a series of reflected oblique shocks, the entrance to the internal workings of the engine contains a weak normal shock ([Babinsky and Harvey, 2011](#)). Behind the normal shock is a region of subsonic flow, providing a further source of unsteady forcing of the system.

Numerical studies of confined turbulent SBLI include [Garnier \(2009\)](#), [Bermejo-Moreno et al. \(2014\)](#) and [Wang et al. \(2015\)](#). In each case the presence of sidewalls resulted in strong three-dimensionality and a significant strengthening of the central interaction. The work of [Garnier \(2009\)](#) performed a detached eddy simulation (DES) of a full wind tunnel configuration with turbulent boundary-layers. It was observed that the presence of sidewalls reduced the effective section of the wind tunnel geometry, and led to a strengthening of the main interaction. Strong fluctuations were found in the corner regions, and a low frequency motion of the reflected shock was observed. The author concluded that quasi-2D simulations disregarding sidewall effects are ‘irrelevant’ ([Garnier, 2009](#)) for strongly separated interactions. The wall-modelled large-eddy simulations (LES) of [Bermejo-Moreno et al. \(2014\)](#) studied turbulent SBLI with comparison to experimental PIV data for rectangular ducts with a 20° flow deflection. It was observed that

the structure and location of the internal shock system was heavily modified compared to span-periodic simulations. Furthermore, Mach stems were observed at the primary interaction for the case strengthened by sidewalls, a feature not present in the span-periodic simulations. Wang et al. (2015) performed wall-resolved LES at Mach 2.7 with a flow deflection of 9° . An upstream shift of the separation and reattachment points was observed as the aspect ratio was decreased from four to one. The same reduction in aspect ratio led to a 30% increase in centreline separation length compared to quasi-2D predictions. Three-dimensional flow features near the main interaction included corner compression waves, secondary sidewall shocks and strong attached transverse flow between the central and corner separations. The main factors responsible for the modified interaction were the swept sidewall SBLI and the aspect ratio.

Considerable attention has been given to three-dimensional corner effects experimentally in recent years owing to their prevalence in supersonic intake applications. Duct SBLI for normal-shock interactions have been investigated by Bruce et al. (2011), Burton and Babinsky (2012) and Vaisakh et al. (2019) among others. Oblique duct SBLI studies include Eagle et al. (2011), Eagle and Driscoll (2014) and Morajkar and Gamba (2016). An open question is to determine the importance of corner separations in relation to the main interaction and how modifications to the corner flow results in divergence from quasi-2D predictions. Much of the work has focused on identifying compression waves generated by the flow deflection in the corner. Eagle et al. (2011) studied corner flows in rectangular supersonic inlets for Mach numbers of 2.0 and 2.75. These exploratory experiments used wedges of 6° and 10° to generate the incident shock, and noted the difficulty in matching simulation and experiment in this area. Recirculation in the corner region was observed to move upstream and form secondary shocks that interacted with the main shock, and in several configurations unstart was observed. They noted that cases with more moderate separation were better suited to CFD, as disturbances would not propagate upstream to the leading edge. Furthermore, reattachment would also occur more promptly to avoid separation induced issues with outflow boundary conditions.

Oil-streak patterns and pressure-sensitive paint are two of the methods which have been used experimentally to infer the impact of corner compressions and their ability to modify other parts of the flow. Xiang and Babinsky (2019) is a recent example of work in this area at Mach 2.5, adding corner blockages to shrink the duct cross section and obtain exaggerated corner separations. It was observed that the central separation was sensitive to variations in the onset and magnitude of the corner separation. A mechanism was proposed to predict the central separation based on the crossing point of the inferred corner compression waves near the bottom wall. For increased corner separations the topology of the central interaction was seen to transition between the ‘owl-like’ first and second states introduced by Perry and Hornung (1984). The transition to the secondary owl-like topology is indicative of increased three-dimensionality of the separated region. It was argued that corner compression waves crossing on the centreline before the interaction region led to reduced separation, while a crossing point within the interaction resulted in larger separations.

Differences also exist between experimental configurations for SBLI, one notable feature being the effect of sidewall gaps for partial-span shock generators. Grossman and Bruce (2017) investigated the effect of duct geometry and the sidewall gap on a Mach 2 SBLI with a 12° flow deflection. The central separation bubble length was sensitive to the size of the sidewall gap, with reduced three-dimensionality and smaller separations seen for larger gaps. Furthermore, the

impingement location of the trailing edge expansion fan was observed to be a critical parameter when determining the size of the central separation. Shifting the expansion fan downstream led to an increase in both the strength and streamwise extent of the separation. A follow up study (Grossman and Bruce, 2018) expanded on these themes in the context of regular-irregular transition of SBLI, where for a fixed initial flow deflection Mach reflections were observed only for certain duct aspect ratios. The streamwise separation length was found to be linearly dependent on the distance between the main SBLI and the impingement point of the trailing expansion fan. The increase in separation length was shown to be linked primarily to an upstream shift of the separation line.

In the work of Poggie and Porter (2019), sidewall confinement was investigated for a 24° turbulent compression ramp configuration at Mach 2.25. The large-scale DNS was performed on $N = 9.4 \times 10^9$ grid cells, demonstrating the substantial computational requirements for investigating these challenging three-dimensional SBLI. Their findings showed that the centreline separation length was strongly dependent on the level of applied flow confinement. It was also noted that regions of subsonic corner flow offer a path for upstream propagation of disturbances, that could influence the unsteadiness of the SBLI. The role of the corner flow for unsteadiness in SBLI was also investigated by Rabey et al. (2019), who compared experimental data to the LES data of Wang et al. (2015). The most significant low-frequency energy content was observed off-centre, and was attributed to the separation shock being strengthened due to shocks generated from the swept sidewall interaction. It was also noted that a significant correlation was not found to extend across the attached flow region between the central and corner separations.

Efforts have also been made to apply flow control to duct SBLI. The work of Koike and Babinsky (2018) used wind tunnel experiments to assess the impact of rectangular vortex generators on regions of corner separation. For a Mach 1.4 inlet, the experiments studied the interaction of a turbulent boundary-layer with a normal shock-wave. Vortex generators were shown to be effective at reducing the observed separation in the corner region, but large regions of separation remained in the centre of the duct. A separate numerical study (Grébert et al., 2018) looked at the effect of upstream micro Vortex-Generators (mVGs) on a turbulent SBLI at Mach 2.7. Hairpin vortices shed from the rear of the mVGs led to overall reductions in pressure load fluctuations and separation size of 9% and 20% respectively. There has recently been an interest in the unsteady dynamics of shock-trains that can form within confined flow configurations. The DNS of Fiévet et al. (2017) attempted to quantify the role of inflow confinement ratio on shock-trains in turbulent duct flows. It was observed that the shock train moved upstream for higher inflow confinement ratios. Temporal variation of the inflow boundary-layer thickness led to a dynamic response of the shock-train, dependent on the excitation frequency. Shock-train unsteadiness was also the subject of Hunt and Gamba (2019) for a duct flow at Mach 2. The complex shock-system was shown to be influenced by upstream-propagating acoustic waves from regions of flow separation, and downstream-travelling vortices shed from the separation bubble. Finally, the wind tunnel tests of Cheng et al. (2017) investigated the response of an oblique shock-train to downstream pressure perturbations at Mach 2.7. The oblique shock-train was shown to be sensitive to downstream pressure perturbations within the separation bubble and subsonic shear layer. Translational motion of the shock-train was observed at the same frequency as the downstream forcing.

1.3 Novel programming approaches for modern computational fluid dynamics

All of the numerical studies discussed in the previous sections share a common reliance on the efficient execution of code on massively-parallel computer systems. Since the turn of the century, diminishing returns from increases in single-core CPU clock-speeds has led to modern CPUs being comprised of multiple processor cores. By spreading the computational workload over multiple processor cores, significant speed-up of application runtime is possible while avoiding the high thermal heat loads associated with higher clock-speeds. A central challenge for developers of scientific software is to be able to achieve efficient utilisation of their software on multiple processor cores. In the domain of CFD, this requires efficient decomposition, communication, and manipulation of the flow data over potentially hundreds of thousands of processor cores. In addition to conventional CPU-based architectures, the past 5-10 years has seen a rapid growth in the use of graphical programming units (GPUs) ([Nvidia GPU applications, 2019](#)) for computational science. The creation of modern CFD codes that can effectively utilise these emerging architectures is one of the main challenges facing state of the art numerical studies today.

A recent school of thought within the domains of computational science and software engineering has called for what is known as a ‘separation of concerns’. This principle is designed to tackle the growing complexity found in major engineering projects ([Ober and Ober, 2017](#)). The idea is born out of the observation that many software projects share similar parallel programming components common to each of them. Furthermore, extracting the maximum performance from emerging computational architectures requires in-depth hardware-specific knowledge and low-level programming techniques ([Reguly et al., 2014](#)), that are not widely taught in the physical sciences. By creating standardised parallel implementation libraries, human resources can be freed-up and better spent on other tasks within that domain. In the context of CFD, the researcher is able to dedicate more time to the numerical algorithms and physical modelling of the problem, rather than optimisation of the parallel code. In addition to x86-CPU and GPU-based architectures, ARM-based CPUs ([McIntosh-Smith et al., 2019](#)), Field Programmable Gate Arrays (FPGAs), and many-core accelerator cards such as the Intel Xeon Phi have all been cited as possible architectures of the future ([Bergman et al., 2019](#)). As the recent discontinuation of the Intel Xeon Phi product line ([AnandTech, 2019](#)) has highlighted however, it is not clear at this stage which, if any, of these architectures will ultimately prevail from this crowded marketplace in the coming decades. As such, codes using cross-platform standardised libraries agnostic to the computational architecture, may be better placed to respond to changes in the computational landscape.

One example of such an approach is the Oxford Parallel Structured (OPS) Embedded Domain Specific Language (EDSL) ([Reguly et al., 2014](#); [Mudalige et al., 2014, 2019](#)) used in this thesis. The OPS library is coupled to the current version of the OpenSBLI code ([Jacobs et al., 2017](#); [Lusher et al., 2018b](#)) developed during this PhD project. The OPS library is an Application Programming Interface (API) embedded within the well known C programming language. All of the standard functionality of the host C language is preserved, with the addition of OPS-specific API calls to enable parallel execution of calculations within the simulation code. There is also an unstructured version of the OPS library called OP2 ([Mudalige et al., 2012](#)), which has been widely applied to simulations for turbo-machinery applications. Other examples of DSLs

include Pluto ([Bondhugula et al., 2008](#)), Mint ([Unat et al., 2011](#)), and Devito ([Lange et al., 2016](#)), however unlike OPS these are often restricted to a single computational platform. OPS takes the base C code generated by the OpenSBLI code-generation framework and performs a source-to-source translation to a variety of parallel programming languages. At present, OPS is capable of generating MPI, OpenMP, and MPI+OpenMP code to target CPU/accelerators, and CUDA, OpenCL and OpenACC for GPUs, with additional MPI support for multi-GPU clusters.

The performance of the OPS library has been investigated externally in [Mudalige et al. \(2014, 2015, 2019\)](#), and [McIntosh-Smith et al. \(2019\)](#) among others. In [Mudalige et al. \(2015\)](#), the performance of the code was assessed on multiple architectures and comparisons were made to hand-written code for each of the computational back-ends. In each case the performance was within 10% of the hand-written code. In [Mudalige et al. \(2019\)](#), a comparison was made to the Fortran-based ‘SBLI’ code developed at the University of Southampton. There was no performance degradation compared to the hand-written SBLI code, with OpenSBLI proving to be 30% faster on Intel CPU nodes. Furthermore, it was shown that low memory-intensity algorithms ([Jammy et al., 2016](#)) were capable of providing a further $2\text{--}3\times$ speed-up compared to the baseline version. The practical aspects of the OpenSBLI and OPS systems are discussed in much greater detail in chapter 3 of this thesis.

Another prominent fluid dynamics project aimed at massively-parallel execution on GPUs is the high-order flux-reconstruction code PyFR ([Witherden et al., 2014](#)). PyFR is a Python-based unstructured mesh framework to solve advection-diffusion problems on streaming architectures. The code utilises a domain specific language that uses Mako templates for platform portability, allowing PyFR to be compatible with both the CUDA and OpenCL programming languages, and with OpenMP in C. In general, unstructured mesh computations often face difficulties on GPUs due to poor data locality. The high-order flux-reconstruction method, however, benefits from a good degree of element locality to achieve good performance on streaming architectures. In the work of [Vermeire et al. \(2017\)](#), PyFR was compared to the popular low-order industrial solver Star-CCM+. The benefits of high-order schemes were demonstrated for a range of test cases, including aerofoils and turbulent flow over a cylinder. The PyFR approach was shown to offer significant accuracy-to-cost improvements compared to the industrial code. The scalability of PyFR on large peta-scale GPU clusters was shown in [Vincent et al. \(2016\)](#). Excellent weak scaling was demonstrated on up to 15,000 NVIDIA K20X GPUs.

Devito ([Lange et al., 2016](#); [Louboutin et al., 2019](#)) is another Python-based project comparable to the aforementioned OpenSBLI and PyFR. Devito is similar to OpenSBLI in the sense that it is also a symbolic stencil-based platform to generate finite-difference models from a high-level abstraction. Mainly targeting geophysical applications, Devito also uses the SymPy library to generate stencil-based finite-difference approximations. The symbolic library is used to manipulate and optimise expressions to improve computational efficiency ([Luporini et al., 2018](#)). Devito was developed to run on Intel Xeon CPU nodes and the now discontinued Intel Xeon Phi accelerator cards. Recently the Devito project has investigated a switch to the OPS library already in active use by OpenSBLI, to enable them to also target NVIDIA GPU platforms ([Luporini et al., 2018](#); [Pandolfo, 2019](#)).

1.4 Thesis objectives and outline

The present work is organised as follows. Chapter 2 begins with an overview of the progress made in high-order shock capturing methods, describing the numerical modelling techniques used to solve the compressible Navier-Stokes equations for supersonic flows. The theory behind WENO schemes is given, with details of the flux-splitting method and characteristic decomposition used. Within this chapter, section 2.4 discusses the implementation of Targeted Essentially non-Oscillatory (TENO) methods (Fu et al., 2016), which are low-dissipative schemes aimed at addressing some of the shortcomings of traditional WENO. Chapter 3 describes the structure of the OpenSBLI code developed during this project, with the various components of the system and how they tie into the OPS parallel library. Of particular importance are the steps needed to go from a symbolic representation of a continuous partial differential equation to its discretized form, and finally to a compilable code. At the end of the chapter is a demonstration of code performance with the OPS library, with single and multi-node GPU performance and scaling. Chapter 4 presents a suite of validation cases to demonstrate specific capabilities of the code implementation. The test cases include a two-dimensional Mach 2 laminar SBLI and turbulent channel flows.

Chapter 5 assesses the impact of numerical dissipation intrinsic to shock-capturing schemes for a supersonic Taylor-Green vortex case and a Mach 1.5 transitional SBLI. These cases were selected to test the numerical methods on cases that are more representative of research problems. The main research component of the physical SBLI problem is contained in Chapter 6 and Chapter 7. Chapter 6 extends the two-dimensional laminar SBLI from the validation chapter to three dimensions with sidewall effects. Complex shock structures within the rectangular duct are shown and the topology of the main interaction are investigated. Forcing strips are added to a laminar duct in Chapter 7, to trigger a transition to turbulence upstream of the main SBLI. The transition is observed to develop first in the corners of the rectangular duct and spread out in a wedge shape across the span. The response of laminar separation bubbles to the intermittent upstream turbulent spots is shown.

The main aims and objectives of the project can be summarised as follows:

- To develop and validate the open-source code-generation framework OpenSBLI.
- To implement high-order WENO/TENO shock-capturing schemes into OpenSBLI.
- To demonstrate that code-generation techniques are feasible for DNS of challenging research problems in fluid mechanics.
- To assess low-dissipative shock-capturing schemes for compressible turbulence.
- To investigate sidewall confinement effects for three-dimensional laminar and transitional shock-wave/boundary-layer interactions (SBLI).
- To conduct parametric studies of aspect ratio and shock strength for rectangular duct SBLI and to elucidate the complex shock-structures and flow topology for three-dimensional SBLI.

1.5 Statement of external contributors

The development of the OpenSBLI code generation system detailed in chapter 3 was carried out by the author in collaboration with Dr Satya P. Jammy (University of Southampton) on a fifty-fifty basis during the first half of the PhD project. I have acted as the sole developer of the code for the second half of the project and have provided support for the various users of the code. The OpenSBLI validation cases in chapter 4 were performed solely by the author with the exception of the turbulent channel flow cases in sections 4.7, 4.8. These two validation cases were a collaborative work with Dr Arash Hamzehloo (Imperial College London), resulting in a paper submitted to the International Journal for Numerical Methods in Fluids. The physical results in chapters 5, 6, and 7 are the individual work of the author under the guidance of Prof Neil D. Sandham.

1.6 Publications and conference presentations

The publication and conference presentation output generated by the author during the PhD project is outlined below. The publications listed in this section form the bulk of the material contained within the thesis.

1.6.1 Paper publications

- D.J. Lusher, S.P. Jammy, N.D. Sandham. Shock-wave/boundary-layer interactions in the automatic source-code generation framework OpenSBLI. *Computers & Fluids*, Volume 173, 2018, Pages 17-21,
- D.J. Lusher, N.D. Sandham. Assessment of low-dissipative shock-capturing schemes for transitional and turbulent shock interactions. *AIAA Aviation Forum*, AIAA 2019-3208.
- D.J. Lusher, N.D. Sandham. The effect of flow confinement on laminar shockwave/boundary-layer interactions. *Journal of Fluid Mechanics* 2019 (Under review). [arXiv:1909.01287 \[physics.flu-dyn\]](https://arxiv.org/abs/1909.01287).
- D.J. Lusher, N.D. Sandham. Transitional shock-wave/boundary-layer interactions in rectangular ducts. *Flow Turbulence and Combustion* 2019 (Under review).

1.6.2 Conference presentations

- Presentations on OpenSBLI at the UK Turbulence Consortium, Imperial College London. September 2017, 2018, 2019.
- Shock-wave/boundary-layer interactions in the automatic code generation framework OpenSBLI. Parallel CFD conference. Glasgow, May 2017.
- Transitional shockwave/boundary-layer interactions with side-wall effects. 7th International Conference on Fluid Dynamics (ECFD 7). Glasgow, June 2018.

- Transitional shockwave/boundary-layer interactions in the automatic source-code generation framework OpenSBLI. International Conference on Computational Fluid Dynamics 10 (ICCFD10), Barcelona, July 2018.
- Shockwave/boundary-layer interactions in transitional rectangular duct flows. ERCOF-TAC Workshop on Direct and Large Eddy Simulation (DLES12). Madrid, June 2019.
- Assessment of low-dissipative shock-capturing schemes for transitional and turbulent shock interactions. AIAA Aviation Forum. Dallas, June 2019.

Chapter 2

Numerical methods

2.1 Introduction

This chapter describes the numerical methods implemented into the compressible flow solver OpenSBLI during the project. All of the methods are performed within a stencil-based finite-difference framework on structured meshes. Structured body-fitted meshes allow for the use of high-order accurate numerical methods for high-fidelity fluid simulations. For low Mach number flows standard central differencing can be applied without issue. Skew-symmetric formulations of the governing equations can be used with central derivatives to improve numerical stability. Central difference methods fail in the presence of discontinuous shock-waves found in transonic, supersonic and hypersonic flows. Resolving shock-waves on a discrete grid creates Gibbs oscillations for finite-difference methods, as they attempt to average over a discontinuous jump occurring over a very thin region of the domain.

Interactions between shock-waves and turbulence are ubiquitous in high-speed flows of practical aeronautical interest. Recent advances in computational power have made implicit large eddy simulation (ILES) and direct numerical simulation (DNS) feasible tools for investigating the underlying physical mechanisms involved. A variety of different shock capturing schemes exist to treat this issue, typically by introducing some artificial numerical dissipation to smear the shock over several grid points. However, the success of tackling these problems depends on contrasting requirements of the applied numerical schemes. Shock-capturing schemes stabilise the solution by adding numerical dissipation in the vicinity of flow discontinuities, but have the detrimental effect of damping small-scale turbulence. Recent reviews of shock-capturing methods have highlighted the need to reduce the numerical dissipation of the schemes while retaining the ability to robustly capture shock-waves ([Pirozzoli, 2011](#)). The next section reviews recent progress in shock-capturing schemes and introduces the Weighted Essentially Non-Oscillatory (WENO) and Targeted Essentially Non-Oscillatory (TENO) schemes implemented within OpenSBLI. The current work focuses on the application of these schemes to the full-3D unsteady compressible Navier-Stokes equations.

2.1.1 A review of high-order ENO-based shock-capturing methods

Traditional approaches to shock-capturing include total variation diminishing (TVD), monotonic upstream-centred scheme for conservation laws (MUSCL), and the family of essentially non-oscillatory (ENO) schemes. The past two decades has seen the emergence of more sophisticated methods with superior shock-capturing properties, addressing some of the issues with traditional schemes. The work by [Tenaud et al. \(2000\)](#) gave a comparison of TVD, MUSCL, and ENO, to an early version of a WENO scheme. It was observed that the TVD scheme using a Van-Leer Harmonic limiter added large amounts of numerical diffusion, and the scheme was sensitive to grid refinement. The MUSCL scheme produced anomalous results, including low frequency oscillations induced downstream of the shock. Better agreement with reference results was obtained via the ENO scheme, but the best shock-resolving performance was obtained by the 5th order WENO formulation. A more recent review was performed by [Johnsen et al. \(2010\)](#), in which shock-fitting, WENO, hybrid-WENO, and artificial diffusivity (AD) schemes were compared for a suite of test problems. Hybrid methods work by switching between a dissipative shock-capturing scheme near discontinuities and non-dissipative methods elsewhere. Standard WENO and AD schemes were found to be unsuitable for compressible turbulence, as the excessive numerical dissipation overwhelmed the physical dissipation and led to under-predictions of the upper 3/4 of resolvable wavenumbers. Hybrid-WENO methods fared better but there was a clear difficulty in creating a shock sensor applicable to all flow types.

A similar study was performed by [Brehm et al. \(2015\)](#) at NASA Ames, motivated by the need to resolve highly shocked unsteady rocket launch flows. The aim of the review was to replace the existing 2nd order MUSCL scheme within the Launch Ascent and Vehicle Aerodynamics (LAVA) finite difference code, with a modern alternative. Comparison was made between central finite-differencing with artificial dissipation (Central-AD), localized artificial diffusivity (LAD), and WENO schemes. The 6th order Central-AD was the computationally cheapest approach, comparable to the cost of the existing MUSCL scheme while offering better spectral resolution. Despite being an attractive option due to the cheap cost, Central-AD performed poorly compared to the other schemes, and large oscillations were still present across flow discontinuities. LAD schemes work by introducing local grid-dependent artificial diffusion through transport coefficients, smoothing out discontinuities in the flow. These methods obtain good spectral resolution for turbulent flows, and were found to be a credible option for flows containing weak shocks. Their performance over strong shocks however was found to be poor, with significant numerical oscillations observed; both Central-AD and LAD were inferior to the more robust WENO schemes.

In the original ENO procedure ([Harten et al., 1987](#)), a selection of smaller candidate stencils were considered over which the smoothness of the solution was evaluated. The candidate stencil over the smoothest region of flow was then selected to construct the numerical flux. The WENO method ([Jiang and Shu, 1996](#); [Shu, 1997](#)) improved upon this by taking a weighted linear combination of all the stencils with a weighting proportional to the local smoothness of the flow. Despite their robust shock-capturing ability, WENO schemes are still too dissipative for smooth regions of the flow and have a significantly higher computational cost. In the review of [Brehm et al. \(2015\)](#), the computational cost of WENO schemes was of the order of 2.5 – 3 times higher than the MUSCL and Central-AD approaches. WENO can be applied to either primitive or conservative variables of the Euler equations directly, or after a transformation into characteristic

space. Performing WENO in characteristic space is more expensive, but offers the sharpest shock resolution, with numerical oscillations observed to be more significant if performed in physical space (Brehm et al., 2015). The work of Gross and Fasel (2016) is an example of WENO schemes being applied to a wide range of laminar-transitional SBLI, relevant to the topics discussed in this thesis.

A recent extension to the family of ENO schemes was proposed by Fu et al. (2016), Fu et al. (2017), named Targeted Essentially non-Oscillatory (TENO) schemes. TENO shares similarities from both ENO and WENO approaches, and boasts significantly lower numerical dissipation. While WENO performed a weighted linear combination of all candidate stencils to build a flux reconstruction, TENO completely discards stencils deemed to contain discontinuities while retaining as many smooth stencils as available. In this sense TENO can be seen as a compromise between ENO and WENO, and also benefits from improved weighting formulations and a staggered stencil arrangement to reduce the number of stencils crossed by a shock. The performance of TENO schemes for transitional SBLI within OpenSBLI is reported in chapter 5 and the corresponding paper (Lusher and Sandham, 2019a).

2.1.2 Alternative WENO formulations

There is considerable breadth within the family of WENO schemes, with improved weighting and stencil formulations addressing some of the shortcomings of the original WENO-JS (Jiang-Shu) method (Jiang and Shu, 1996; Shu, 1997). One example of this was the WENO-M formulation by Henrick et al. (2005), who observed a reduction in spatial order of the scheme at critical points, and formulated an improved weighting via the use of a mapping function. Borges et al. (2008) achieved the same improvement but without the need for a mapping function in the 5th order WENO-5Z scheme. WENO-5Z has a computational cost saving of 25% over WENO-M for the same shock-capturing properties, giving improved rates of convergence and lower induced numerical dissipation than WENO-JS for the same computational cost.

Further improvements include the 6th order central-upwind WENO-6 scheme by Hu et al. (2010), which introduces an additional candidate stencil and results in reduced numerical dissipation. Other alternatives are the compact CWENO developed by Ghosh and Baeder (2012), and WENO-5B (Brehm et al., 2015) which blends between 5th and 6th order formulations via an upwinding parameter. Performance of WENO-5Z, WENO-6, CWENO and WENO-5B was assessed for a variety of standard shock test cases in Brehm et al. (2015), relative to the original WENO-JS formulation. There was considerable difference among the WENO formulations, with significant improvements observed over WENO-JS for all of the new weightings. WENO-6 and CWENO had spectral resolution close to central schemes, but had higher numerical oscillations compared to WENO-5Z and WENO-5B. CWENO had the highest spectral resolution, but is the most computationally expensive as it requires the solution of an implicit tri-diagonal system.

2.2 Governing equations

The application of OpenSBLI to date has been for aeronautical fluid flow problems, solving the compressible 3D Navier-Stokes equations with finite-difference methods. Despite aeronautics

being the main research interest of the authors, OpenSBLI is also capable of generating finite-difference codes for other applications; the core design of OpenSBLI aims to be a general code-generation platform for computational science. The physical results within this thesis focus on the application of OpenSBLI to the compressible Navier-Stokes equations where robust shock-capturing schemes are required. OpenSBLI uses numerical indices to distinguish variables, for example the Cartesian coordinate base (x, y, z) and standard velocity components (u, v, w) are taken to be (x_0, x_1, x_2) , and (u_0, u_1, u_2) respectively in the code. This allows the code generation process to be more dynamic: all of the indexed quantities and loops are set globally based on the number of dimensions specified at the top of the user's problem script.

The governing equations for all of the simulations in this work are the dimensionless compressible Navier-Stokes equations for a Newtonian fluid. Applying conservation of mass, momentum and energy, in the three spatial directions x_i ($i = 0, 1, 2$) results in a system of five partial differential equations. These equations are defined for a density ρ , pressure p , temperature T , total energy E , and velocity components u_k as

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_k} (\rho u_k) = 0, \quad (2.1)$$

$$\frac{\partial}{\partial t} (\rho u_i) + \frac{\partial}{\partial x_k} (\rho u_i u_k + p \delta_{ik} - \tau_{ik}) = 0, \quad (2.2)$$

$$\frac{\partial}{\partial t} (\rho E) + \frac{\partial}{\partial x_k} \left(\rho u_k \left(E + \frac{p}{\rho} \right) + q_k - u_i \tau_{ik} \right) = 0, \quad (2.3)$$

with heat flux q_k and stress tensor τ_{ij} defined as

$$q_k = \frac{-\mu}{(\gamma - 1) M_\infty^2 Pr Re} \frac{\partial T}{\partial x_k}, \quad (2.4)$$

$$\tau_{ik} = \frac{\mu}{Re} \left(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i} - \frac{2}{3} \frac{\partial u_j}{\partial x_j} \delta_{ik} \right), \quad (2.5)$$

where Pr , Re and γ are the Prandtl number, Reynolds number and ratio of heat capacities respectively. The equations are non-dimensionalized by freestream velocity, density and temperature $(U_\infty^*, \rho_\infty^*, T_\infty^*)$. For the shock-wave/boundary-layer interaction cases a characteristic length based on the displacement thickness δ^* of the boundary layer imposed at the inlet is used. Pressure is normalised by $\rho_\infty^* U_\infty^{*2}$, to give $(U_\infty, \rho_\infty, T_\infty) = 1$ in the freestream. Unless specifically stated, temperature dependent dynamic viscosity $\mu(T)$ is given by Sutherland's law

$$\mu(T) = T^{\frac{3}{2}} \left(\frac{1 + \frac{T_s^*}{T_\infty^*}}{\frac{T}{T_\infty} + \frac{T_s^*}{T_\infty^*}} \right), \quad (2.6)$$

with freestream and Sutherland temperatures taken to be $T_\infty^* = 288.0\text{K}$ and $T_s^* = 110.4\text{K}$. For a freestream Mach number M_∞ , pressure and local speed of sound are defined as

$$p = (\gamma - 1) \left(\rho E - \frac{1}{2} \rho u_i u_i \right) = \frac{1}{\gamma M_\infty^2} \rho T \quad \text{and} \quad a = \sqrt{\frac{\gamma p}{\rho}}. \quad (2.7)$$

Throughout this work the wall-normal skin friction C_f is calculated as

$$C_f = \frac{\tau_w}{\frac{1}{2} \rho_\infty U_\infty^2}, \quad (2.8)$$

for a wall shear stress

$$\tau_w = \mu \frac{\partial u}{\partial y} \Big|_{y=0} \quad \text{or} \quad \tau_w = \mu \frac{\partial u}{\partial z} \Big|_{z=0, L_z} \quad (2.9)$$

depending on whether the quantity is being evaluated on the bottom wall ($y = 0$) or sidewalls ($z = 0, L_z$) of the domain.

2.3 Weighted Essentially Non-Oscillatory (WENO) schemes

This section introduces the WENO scheme option for shock-capturing in OpenSBLI, beginning with the fundamental reconstruction procedure for the scalar case. The implementation of WENO schemes within OpenSBLI was published in [Lusher et al. \(2018b\)](#) with validation of the two-dimensional laminar SBLI case shown in section 4.6. Modified WENO-Z non-linear weights are introduced, discussion of flux splitting methods, and application of WENO to systems of equations. Finally, the algorithm used to perform the characteristic decomposition is outlined: WENO is applied to each component of the Euler equations transformed into characteristic space.

2.3.1 Scalar reconstruction

This section details the reconstruction procedure used by essentially ENO and WENO schemes. WENO was an evolution of the earlier ENO formulation and uses the low order numerical ENO stencils to build up a higher order weighted approximation. As such, it is necessary to cover the basics of the ENO method before proceeding to WENO.

The basis for an ENO reconstruction is to approximate a function $f(x_i)$, at a half node position $f(x_{i+\frac{1}{2}})$, by creating a set of numerical stencils and comparing the smoothness of the function over each. In the original ENO scheme smoothness indicators were calculated for each of the stencils, with a single stencil from the set chosen based on the local smoothness of the solution; stencils containing discontinuities are discarded. WENO improves upon this method by taking a convex combination of each of the candidate stencils, weighted by their local smoothness. As a result, better coverage about the reconstruction point is achieved, and the effect of stencils containing discontinuities is removed as their assigned weighting approaches zero.

For a WENO scheme of order $2k - 1$, k ENO stencils are required. Therefore, in a 5th order WENO scheme we have $k = 3$ ENO stencils, covering a total of $2k - 1 = 5$ points around $f(x_i)$, with the points in each stencil given by

$$S_r = \{x_{i-r}, \dots, x_{i-r+k-1}\}, \quad r = [0, k - 1], \quad (2.10)$$

which for 5th order gives

$$S_0 = \{f(x_i), f(x_{i+1}), f(x_{i+2})\}, \quad (2.11)$$

$$S_1 = \{f(x_{i-1}), f(x_i), f(x_{i+1})\}, \quad (2.12)$$

$$S_2 = \{f(x_{i-2}), f(x_{i-1}), f(x_i)\}, \quad (2.13)$$

as shown in Figure 2.1. These stencils are then used to create a reconstruction with the linear expansion

$$\hat{f}_{i+\frac{1}{2}}^{(r)} = \sum_{j=0}^{k-1} c_{rj} f_{i-r+j}, \quad r = [0, k-1], \quad (2.14)$$

where c_{rj} are the ENO coefficients given by [Shu \(1997\)](#).

For standard ENO, k stencils are proposed over $2k-1$ cells, but as only one stencil is chosen the scheme is of k -th order accuracy. In the WENO case all $2k-1$ cells can be used in smooth regions, resulting in a scheme of $(2k-1)$ -th order accuracy.

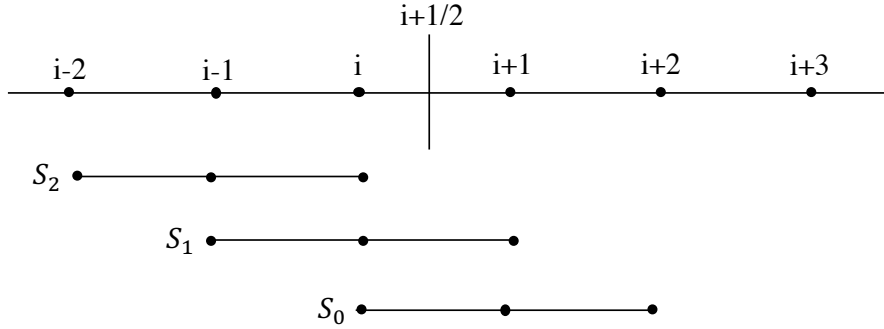


FIGURE 2.1: Stencil for 5th order WENO, upwind reconstruction. Nodes are cell centres.

2.3.2 Jiang-Shu (JS) smoothness indicators

The most commonly used smoothness indicator was introduced by [Jiang and Shu \(1996\)](#), defined as

$$\sigma_k = \sum_{l=1}^{r-1} \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \Delta x^{2l-1} \left(q_k^{(l)} \right)^2 dx, \quad (2.15)$$

where $q_k^{(l)}$ is the l -th derivative of the $(r-1)$ -th order interpolating polynomial $q_k(x)$ over the stencil S_k . This smoothness indicator forms the sum of L^2 norms of the interpolating polynomial derivatives over a cell width. To construct the $(2k-1)$ -th order WENO approximation, the non-linear WENO weights are calculated for $r = [0, k-1]$ as

$$\omega_r = \frac{\alpha_r}{\sum_{n=0}^{k-1} \alpha_n}, \quad (2.16)$$

with the smoothness indicator (2.15) forming part of the alpha terms as

$$\alpha_r = \frac{d_r}{(\epsilon + \sigma_r)^p}, \quad (2.17)$$

for optimal weights d_r and constants, p , ϵ . The standard value of $p = 2$ is used, and ϵ set to a small non-zero number (10^{-6}) to avoid division by zero, as the smoothness indicator is essentially zero for smooth regions. Smaller values of ϵ lead to slower rates of convergence and higher absolute error for the WENO scheme, a comparison of ϵ values and their influence on results can be found in [Henrick et al. \(2005\)](#). The optimal weights d_r are the constants that result if the solution is smooth in all candidate stencils, and depend on the k values of r as [Shu \(1997\)](#)

$$d_0 = \frac{2}{3}, \quad d_1 = \frac{1}{3}, \quad k = 2, \quad (2.18)$$

$$d_0 = \frac{3}{10}, \quad d_1 = \frac{3}{5}, \quad d_2 = \frac{1}{10}, \quad k = 3, \quad (2.19)$$

with the condition that they must sum to 1 in each case. For 3rd order WENO ($k = 2$) the smoothness indicators can be calculated explicitly for the upwind reconstruction as in [Shu \(1997\)](#) such that

$$\sigma_0 = (f_{i+1} - f_i)^2, \quad (2.20)$$

$$\sigma_1 = (f_i - f_{i-1})^2, \quad (2.21)$$

and for fifth order ($k = 3$) as

$$\sigma_0 = \frac{13}{12} (f_i - 2f_{i+1} + f_{i+2})^2 + \frac{1}{4} (3f_i - 4f_{i+1} + f_{i+2})^2, \quad (2.22)$$

$$\sigma_1 = \frac{13}{12} (f_{i-1} - 2f_i + f_{i+1})^2 + \frac{1}{4} (f_{i-1} - f_{i+1})^2, \quad (2.23)$$

$$\sigma_2 = \frac{13}{12} (f_{i-2} - 2f_{i-1} + f_i)^2 + \frac{1}{4} (f_{i-2} - 4f_{i-1} + 3f_i)^2. \quad (2.24)$$

The WENO schemes have been implemented in OpenSBLI for arbitrary odd orders. The stencil locations (2.11), non-linear weights (2.16), and Jiang-Shu smoothness indicators (2.15) are all generated symbolically during the code generation process. In practice WENO schemes of 3rd to 7th order are most commonly used, as the increasingly wide numerical stencils become computationally inefficient at very high order.

2.3.3 WENO-Z smoothness indicators

The second shock capturing scheme implemented into OpenSBLI was the improved WENO-Z formulation of [Borges et al. \(2008\)](#), [Castro et al. \(2011\)](#). Non-linear weights ω_r from the improved

formulation are given as

$$\omega_r^z = \frac{\alpha_r^z}{\sum_{n=0}^2 \alpha_n^z}, \quad \alpha_r^z = d_r \left(1 + \left(\frac{\tau}{\sigma_r + \epsilon} \right)^2 \right), \quad (2.25)$$

with ($\epsilon \sim 10^{-40}$), and smoothness indicators σ_r and optimal weights d_r as before. The parameter τ is a global smoothness measure, calculated as the absolute difference between smoothness indicators using the formula in [Castro et al. \(2011\)](#). As an example, at 5th order ($k = 3$) the global smoothness measure is $\tau = |\sigma_0 - \sigma_2|$.

2.3.4 Reconstruction procedure

Having constructed the WENO weights ω_r , the final flux contribution at $\hat{f}_{i+\frac{1}{2}}$ is given by the linear combination of ω_r and the ENO expansion (2.14) as

$$\hat{f}_{i+\frac{1}{2}} = \sum_{r=0}^{k-1} \omega_r \hat{f}_{i+\frac{1}{2}}^{(r)}, \quad (2.26)$$

satisfying the conditions

$$\omega_r \geq 0, \quad \text{and} \quad \sum_{r=0}^{k-1} \omega_r = 1. \quad (2.27)$$

Applying the WENO reconstruction to equations of the form

$$U_t + f(U)_x = 0, \quad (2.28)$$

the flux term $f(U)_x$ can be approximated with (2.26), so (2.28) becomes

$$\frac{\partial U_j}{\partial t} + \frac{1}{\Delta x} \left(\hat{f}_{i+\frac{1}{2}} - \hat{f}_{i-\frac{1}{2}} \right) = 0, \quad (2.29)$$

where $\hat{f}_{i+\frac{1}{2}}$ and $\hat{f}_{i-\frac{1}{2}}$ are the WENO approximations of $f(U_i(t))$ on the cell walls $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$ of node x_i , and Δx is the grid spacing. A simple flux-splitting approach can be taken such that

$$f(U) = f^+(U) + f^-(U), \quad (2.30)$$

where we have

$$\frac{\partial f^+(U)}{\partial U} > 0 \quad \text{and} \quad \frac{\partial f^-(U)}{\partial U} < 0, \quad (2.31)$$

and take the Lax-Friedrich splitting method

$$f^\pm(U) = \frac{1}{2} (f(U) \pm \alpha U), \quad \alpha = \max_U |f'(U)|. \quad (2.32)$$

The evaluation of α for a characteristic system of equations is taken over the local WENO stencil, this is known as local Lax-Friedrichs flux splitting. An alternative approach is to compute the maximum over the entire grid range, giving a global Lax-Friedrichs method which is more dissipative.

The flux at the interface ($\hat{f}_{i+1/2}$) is evaluated using (2.30). Care must be taken with the direction of the flux term, requiring different stencil biasing based on the direction of the wind speed. The upwind $f^+(U)$ reconstruction uses a stencil biased with an additional point on the left and the downwind $f^-(U)$ an additional point to the right. For a fifth order WENO scheme this results in the stencils $[f_{i-2}, f_{i-1}, f_i, f_{i+1}, f_{i+2}]$ for upwind, and $[f_{i-1}, f_i, f_{i+1}, f_{i+2}, f_{i+3}]$ downwind. Multi-dimensional problems are solved by applying the WENO procedure in each spatial dimension subsequently, which is the main computational cost-saving advantages of the finite-difference formulation over finite-volume (Shu, 1997).

2.3.5 Application to systems of equations

The WENO procedure can also be applied to systems of hyperbolic equations in the same manner, for an $m \times m$ system with solution vector U the Jacobian $f'(U)$ has m real eigenvalues

$$\lambda_1(U) \leq \dots \leq \lambda_m(U), \quad (2.33)$$

with a set of independent eigenvectors $r(U)$ that form the columns of the right eigenvector matrix

$$R(U) = (r_1(U), \dots, r_m(U)), \quad (2.34)$$

to diagonalise the Jacobian such that

$$R^{-1}(U)f'(U)R(U) = \Lambda(U), \quad (2.35)$$

where the diagonal entries of $\Lambda(U)$ are the eigenvalues (2.33). A simple method for systems of equations is to apply WENO component by component, in terms of either the primitive or conservative variables. These methods however can be inadequate for problems involving strong shocks Brehm et al. (2015), for which the more robust characteristic decomposition outlined in section 2.3.6 is used.

2.3.6 Characteristic decomposition

The characteristic decomposition algorithm is provided by Shu (1997), outlined here for the case of linear $f(U) = AU$ with a constant matrix A . A change of variable is performed using the left eigenvector matrix R^{-1} such that

$$V = R^{-1}U, \quad (2.36)$$

to transform the hyperbolic PDEs to diagonal form

$$V_t + \Lambda V_x = 0, \quad (2.37)$$

where the m equations above are decoupled into

$$W_t + \lambda_j W_x = 0, \quad (2.38)$$

for each eigenvalue λ_j . The reconstruction process is then applied to each of the scalar equations (2.38), returning to the physical space afterwards by calculating

$$U = RV. \quad (2.39)$$

In cases where $f'(U)$ is not constant, $R(U)$, $R^{-1}(U)$ and $\Lambda(U)$ all depend on the value of U , and must be frozen locally to apply the decomposition procedure. This requires calculation of an average approximation of the Jacobian at $U_{i+\frac{1}{2}}$, with the simplest being the arithmetic mean

$$U_{i+\frac{1}{2}} = \frac{1}{2} (U_i + U_{i+1}). \quad (2.40)$$

Matrices $R(U_{i+\frac{1}{2}})$, $R^{-1}(U_{i+\frac{1}{2}})$ and $\Lambda(U_{i+\frac{1}{2}})$ can then be used to calculate the numerical flux at the interface $x_{i+\frac{1}{2}}$. An alternative to the simple average given in (2.40) is to form the system using Roe averaging. For the interface location $U_{i+\frac{1}{2}}$, the Roe averaged density is defined as

$$\hat{\rho} = \sqrt{\rho_i \rho_{i+1}}, \quad (2.41)$$

and averaged velocity components $\hat{u}^j, j = (0, 1, 2)$ as

$$\hat{u}^j = \frac{u_i^j \sqrt{\rho_i} + u_{i+1}^j \sqrt{\rho_{i+1}}}{\sqrt{\rho_i} + \sqrt{\rho_{i+1}}}. \quad (2.42)$$

The Roe averaged speed of sound is given by

$$\hat{a} = \sqrt{(\gamma - 1) \left(\left(\frac{1}{\sqrt{\rho_i} + \sqrt{\rho_{i+1}}} \right) \left(\frac{p_i + \rho E_i}{\sqrt{\rho_i}} + \frac{p_{i+1} + \rho E_{i+1}}{\sqrt{\rho_{i+1}}} \right) - \frac{\sum_{j=0}^2 \hat{u}^j}{2} \right)}. \quad (2.43)$$

Both of these averaging options are available in OpenSBLI by invoking either the `SimpleAverage` or `RoeAverage` classes prior to selecting a shock-capturing scheme.

2.3.7 Summary of the reconstruction algorithm

For 1D characteristic-wise finite-differencing, the following steps must be applied:

1. Construct the flux $f(U)$ and solution U over all values of i .
2. Take each $x_{i+\frac{1}{2}}$ and carry out the following:
 - (a) Compute the average state $U_{i+\frac{1}{2}}$ as in (2.40) or (2.41)-(2.43).
 - (b) Obtain the eigensystem $R(U_{i+\frac{1}{2}})$, $R^{-1}(U_{i+\frac{1}{2}})$ and $\Lambda(U_{i+\frac{1}{2}})$ of the Jacobian $f'(U)$.
 - (c) Perform the change of variable transformation to the flux $f(U)$ and solution U into local characteristic fields $V_j = R^{-1}U_j$, and $g_j = R^{-1}f(U_j)$.
 - (d) Apply the flux splitting WENO procedure outlined in section 2.3.1 for each characteristic component to get the flux $\hat{g}_{i+\frac{1}{2}}^\pm$. For the case of local Lax-Friedrichs splitting $f^\pm(U) = \frac{1}{2} (f(U) \pm \alpha U)$, we calculate α for the l^{th} characteristic component as

$$\alpha = \max |\lambda_l(U_j)|, \quad (2.44)$$

where j are the points contained in the local WENO stencil.

(e) Transform the flux back into physical space $\hat{f}_{i+\frac{1}{2}}^\pm = R\hat{g}_{i+\frac{1}{2}}^\pm$.

3. Build the difference approximation as in (2.29)

$$\frac{1}{\Delta x} \left(\hat{f}_{i+\frac{1}{2}} - \hat{f}_{i-\frac{1}{2}} \right), \quad (2.45)$$

using

$$\hat{f}_{i+\frac{1}{2}} = \hat{f}_{i+\frac{1}{2}}^+ + \hat{f}_{i+\frac{1}{2}}^-. \quad (2.46)$$

where \hat{f} is the sum of the split flux $\hat{f} = \hat{f}^+ + \hat{f}^-$. The simplest flux splitting approach is the Lax-Friedrich method, where the positive and negative flux contributions are defined for a component j of the system of equations as

$$f^\pm(U_j) = \frac{1}{2} (f(U_j) \pm \alpha_j U_j), \quad \alpha_j = \max |\Lambda_l(U_j)|. \quad (2.47)$$

The α_j term is the characteristic wave-speed for the j^{th} component of the system, evaluated as the maximum eigenvalue (2.33) over the l WENO stencil points.

2.4 Targeted Essentially Non-Oscillatory (TENO) schemes

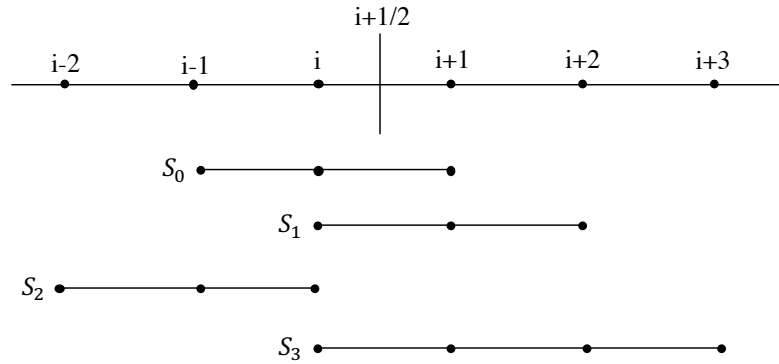


FIGURE 2.2: Staggered stencil arrangement for TENO schemes of increasing order, based on [Fu et al. \(2016\)](#). The flux is evaluated at the half-node $x_{i+1/2}$ location.

Shock-capturing methods introduce excessive levels of numerical dissipation to a flow, making them a poor choice for resolving small scale structures in transitional and turbulent flows. To obtain similar resolution to a non-dissipative scheme, excessively fine grids are required. These large grids are impractical for DNS and severely limit the scale of the problem that can be tackled. There have been numerous attempts to reduce the numerical dissipation of the underlying WENO schemes, such as the improved WENO-Z scheme introduced in section 2.3.3, the WENO-5B of [Brehm et al. \(2015\)](#) and the 6th order WENO6 of [Hu et al. \(2010\)](#).

A more significant improvement was introduced recently by the TENO schemes proposed in [Fu et al. \(2016\)](#), [Fu et al. \(2017\)](#). TENO schemes fit into the same flux reconstruction framework as WENO, with the same flux splitting and characteristic decompositions used. TENO schemes

differ from WENO however in three fundamental ways: a staggered ordering of candidate stencils as in Figure 2.2, the complete removal of candidate stencils deemed to be non-smooth, and modified non-linear weights optimized for low numerical dissipation.

The significance of the stencil staggering is said to reduce the number of candidate stencils that are crossed by a discontinuity (Fu et al., 2016), helping to enforce the underlying principles of the basic ENO approach. In the standard WENO formulation each candidate stencil is given a non-linear weighting (2.16), with candidate stencils crossing discontinuities given a low albeit non-zero weight. TENO schemes abandon this approach in favour of a discrete cut off function that discards candidate stencils completely from the flux reconstruction if they are deemed to be sufficiently non-smooth. Candidate stencils that are considered smooth are included in the reconstruction with their ideal non-linear weight to further reduce numerical dissipation. The non-linear weights from (2.16) are reformulated with ideal weights d_r for a scheme of order K to be

$$\omega_r = \frac{d_r \delta_r}{\sum_{r=0}^{K-3} d_r \delta_r}, \quad (2.48)$$

where δ_r is a discrete cut-off function of the form

$$\delta_r = \begin{cases} 0 & \text{if } \chi_r < C_T \\ 1 & \text{otherwise} \end{cases}$$

for a tunable cut-off parameter C_T . The smoothness measures χ_r here are the same as the weight normalization process in WENO

$$\chi_r = \frac{\gamma_r}{\sum_{r=0}^{K-3} \gamma_r}, \quad (2.49)$$

comprised of the WENO-Z inspired form of non-linear TENO weights (Fu et al., 2016) given by

$$\gamma_r = \left(C + \frac{\tau_K}{\beta_r + \epsilon} \right)^q, \quad r = 0, \dots, K-3. \quad (2.50)$$

Smoothness indicators β_r are unchanged from the standard Jiang-Shu formulation (Jiang and Shu, 1996), and $\epsilon \sim 10^{-40}$ remains a small parameter to avoid division by zero. The global smoothness indicator τ_K measures smoothness over the entire stencil, and is given for 5th and 6th order TENO schemes as

$$\tau_5 = |\beta_0 - \beta_2|, \quad (2.51)$$

$$\tau_6 = |\beta_3 - \frac{1}{6}(\beta_0 + \beta_2 + 4\beta_1)|. \quad (2.52)$$

One major improvement of the TENO schemes is the optimization of the parameters C and q within the non-linear weights (2.50), and the tunable cut-off parameter C_T found in the discrete weight selection function (2.4). The parameters C and q can be varied to adjust the level of dissipation invoked by the non-linear weight and for physically-motivated scale-separation mechanisms (Fu et al., 2016). TENO schemes take the values of $q = 6$ and $C = 1$, which was shown to significantly enhance the discontinuity detection capability of the scheme (Fu et al., 2016).

C_T meanwhile is the parameter that determines whether a given candidate stencil is rejected or contributes to the flux reconstruction. Lower values of C_T are suitable for compressible turbulence simulations where minimal numerical dissipation is required, but this comes at the cost of increased spurious oscillations around shock-waves. In the work of [Fu et al. \(2016\)](#) an optimization process was performed, with the recommended optimal values of C_T for 5th and 6th order TENO schemes said to be 1×10^{-5} and 1×10^{-7} respectively. Lower values of 1×10^{-10} may be used for shock-free compressible turbulence cases, such as the compressible turbulent channel flow in section 4.8.

The computational cost of the TENO schemes is approximately 15-20% greater than a WENO scheme of equivalent order (table 5.1), but offers significantly lower dissipation while retaining sharp shock capturing. An assessment of the various shock-capturing options in OpenSBLI is given in chapter 5 and in [Lusher and Sandham \(2019a\)](#). The increase in computational cost can be attributed to the addition of if-else statements required to evaluate the cut-off function in (2.4).

2.4.1 Adaptive TENO-A schemes

A recent development of the TENO schemes was proposed in [Fu et al. \(2018\)](#), which will be referred to as TENO-A throughout. The TENO-A (adaptive TENO) scheme applies adaptive control of the scheme's numerical dissipation in different regions of the flow. This is achieved by adding a shock sensor to modify the threshold value C_T defined in section 2.4. In smooth regions of the flow fewer stencils will be discarded from the reconstruction when compared to the base TENO schemes. Near shocks the value of C_T is increased to robustly capture shocks and minimize spurious oscillations. The linear weights in equation (2.50) are also tuned in [Fu et al. \(2018\)](#) to further improve the spectral resolution of the scheme. The impact of the optimized weights on shock capturing and numerical dissipation is demonstrated in chapter 5.

In this work a modified version of the Ducros sensor ([Ducros et al., 1999](#))

$$\Phi = \frac{(\nabla \cdot \mathbf{u})^2}{(\nabla \cdot \mathbf{u})^2 + (\omega)^2 + \epsilon}, \quad (2.53)$$

is used for the adaptive dissipation control. The Ducros sensor was designed to distinguish regions of high dilatation $(\nabla \cdot \mathbf{u})$ near shocks from turbulent regions of high vorticity ω . The modified sensor takes the form

$$\Phi = \text{Min} \left(1, \frac{1}{2} (1 - \tanh(2.5(1 + a(\nabla \cdot \mathbf{u})))) \cdot \left(\frac{|\nabla \cdot \mathbf{u}|}{|\nabla \cdot \mathbf{u}| + b\omega^2 + \epsilon} \right) \right), \quad b > 1, \quad (2.54)$$

where the minimum function ensures the sensor is bounded between $\Phi = [0, 1]$. The hyperbolic tangent factor is similar to the form used in [Bhagatwala and Lele \(2009\)](#) for artificial viscosity methods. It is motivated by the inability of the original Ducros sensor to distinguish between shocks and regions of expansion, leading to the sensor being active and excessively dissipative where it is not needed. Positive dilatation is filtered out very effectively with this method. The parameter a can be used to strengthen activation near dilatation and is taken to be $a = 100$ in this study. Other approaches to improve the Ducros sensor includes that of [Hendrickson et al.](#)

(2018), in which a filter in a similar form as the sensor is applied as a pre-processing step to help localize the sensor to shocks.

For laminar-turbulent transitional SBLI problems the strength of the incident oblique shock is often relatively low compared to fully turbulent interactions. Flow deflections in the region of $\theta = 2^\circ$ are common in transitional studies Giepmans et al. (2018), while often in excess of $\theta = 10^\circ$ for turbulent SBLI. As a result, the strength of the dilatation rate at shocks can be similar to the dilatation rate present in turbulence. To ensure the sensor is switching off effectively in turbulence a constant $b > 1$ is multiplied into the vorticity term of (2.54) to sharpen its response. For cases involving wall bounded flows, we require a sensor that turns off within boundary layers. For the SBLI cases in chapter 5, the parameter is chosen to be $b = \left(\frac{M_\infty}{M}\right)^2$ for a streamwise local Mach number M .

The original sensor is also very sensitive to the value of ϵ despite its inclusion merely to avoid division by zero. A small value of $\epsilon = 1 \times 10^{-30}$ is used in this work to fulfil its original purpose. Mapping of the shock sensor Φ to the value of C_T for the TENO-A scheme is taken to be the same form as proposed by Fu et al. (2018)

$$\begin{cases} g(\Phi) = (1 - \Phi)^4(1 + 4\Phi) \\ \bar{\beta} = \alpha_1 - \alpha_2(1 - g(\Phi)) \\ C_T = 10^{-\lfloor \bar{\beta} \rfloor} \end{cases} \quad (2.55)$$

for a Gauss bracket $\lfloor \bar{\beta} \rfloor$. The parameters α_1, α_2 are user defined depending on the specific problem considered and are taken as $\alpha_1 = 10.5$ and $\alpha_2 = 5.5$ in this work. The resulting stencil cut-off parameter varies from $C_T = 10^{-10}$ in smooth regions to $C_T = 10^{-5}$ around shocks. An example of the sensor activation regions is shown in section 5.3 for a Mach 1.5 transitional shock-wave/boundary-layer interaction.

2.5 Hybrid central-WENO scheme

In addition to the WENO, TENO and TENO-A schemes added to OpenSBLI, a hybrid central-WENO scheme was also developed for the code. Hybrid methods pair a non-dissipative central differencing scheme to a dissipative shock-capturing method via a shock-sensor. The spatial fluxes in (2.1) are reconstructed direction by direction with either the central or shock capturing scheme depending on activation of the sensor. The default shock sensor is the modified Ducros sensor in equation (2.54), bounded on the interval $\theta = [0, 1]$. The threshold value of the sensor depends on the specific problem but is usually taken to be 0.65. For smooth regions of the domain below this threshold value, the flux reconstruction is performed with a non-dissipative scheme instead of the shock-capturing.

Initial attempts with standard central differences for the non-dissipative scheme proved to be unstable, requiring the use of the Ducros split skew-symmetric formulation (Ducros et al., 2000) to improve the numerical stability. The non-dissipative part was evaluated at the half-node locations $x_{i+1/2}$ to be consistent with the half-node formulation of the shock-capturing schemes. A standard 4th order conservative derivative treatment (Ducros et al., 2000) for two functions U, V , to be evaluated at $x_{i+1/2}$ is

$$f_{i+1/2}^{div} = \frac{1}{12} (-U_{i+2}V_{i+2} + 7U_{i+1}V_{i+1} + 7U_iV_i - U_{i-1}V_{i-1}). \quad (2.56)$$

This conservative formulation can lead to aliasing errors that will affect the stability of the solution. To combat this, the derivative can be written in skew-symmetric form such that

$$f_{i+1/2}^{skew} = \frac{1}{3} (U_i + U_{i+1})(V_i + V_{i+1}) - \frac{1}{24} (U_{i-1}V_{i-1} + U_{i-1}V_{i+1} + U_iV_i + U_iV_{i+2} + U_{i+1}V_{i+1} + U_{i+1}V_{i-1} + U_{i+2}V_i + U_{i+2}V_{i+2}). \quad (2.57)$$

Terms in the governing Navier-Stokes equations (2.1) containing one variable are evaluated using the divergent form (2.56) with $V = 1$. Multiple product terms are evaluated using the skew-symmetric formulation (2.57) to improve the stability of the scheme. The general form of the Ducros-type skew-symmetric flux splitting is given by

$$f_{i+1/2} = \sum_{k=1}^p \frac{1}{2} \alpha_k^{(p)} \sum_{m=0}^{k-1} (a_{i-m} + a_{i+k-m})(b_{i-m} + b_{i+k-m}), \quad (2.58)$$

for a scheme of $(2p)$ th order (Yee and Sjögren, 2018). The quantities a and b are the indexed flow variables to which the skew-symmetric form is being applied. The coefficients $\alpha_k^{(p)}$ are the finite-difference coefficients for a $(2p)$ th order scheme evaluated at the $x_{i+1/2}$ grid location. By default the skew-symmetric formulation generated in OpenSBLI will be of one order less than the user-selected odd-order of the shock-capturing scheme.

2.6 Viscous & metric derivatives

Heat flux and viscous terms of the compressible Navier-Stokes equations in (2.4) and (2.5) respectively, do not require the treatment from the WENO/TENO schemes and can simply be computed with standard central differencing. To improve numerical stability the Laplacian terms are expanded, and one-sided derivatives are used at all domain boundaries, aside from periodic. OpenSBLI can generate central finite-difference schemes of arbitrary even order. For the simulations in this work a 4th order formulation of the form

$$\frac{\partial f_i}{\partial x} \approx \frac{f_{i-2} - 8f_{i-2} + 8f_{i-2} - f_{i+2}}{12\Delta x}, \quad (2.59)$$

is used for the interior points of the domain. To be able to simulate problems with wall boundaries, one-sided derivative formulations were implemented into OpenSBLI. The main one-sided derivative formulation is that of Carpenter et al. (1998), with the derivative evaluation at the first four points near domain boundaries replaced by alternative weightings. The central differencing inside the domain is kept at fourth order to match the fourth order of the one-sided boundary treatment, avoiding numerical errors from matching schemes of different orders at different places in the domain. The Carpenter et al. (1998) first-derivative operator \mathbf{D} is defined for a grid spacing Δx_i as

$$\mathbf{D}\mathbf{f} = \frac{1}{\Delta x_i} \mathbf{P}^{-1} \mathbf{Q}\mathbf{f}, \quad (2.60)$$

where \mathbf{f} is a vector of the boundary point and five interior values of a flow variable normal to the boundary. The matrices \mathbf{P}, \mathbf{Q} , are defined as

$$P = \begin{bmatrix} \frac{-(216b+2160a-2125)}{12960} & \frac{(81b+675a+415)}{540} & \frac{-(72b+720a+445)}{1440} & \frac{-(108b+756a+421)}{1296} \\ \frac{(81b+675a+415)}{540} & \frac{-(4104b+32400a+11225)}{4320} & \frac{(1836b+14580a+7295)}{2160} & \frac{-(216b+2160a+655)}{4320} \\ \frac{-(72b+720a+445)}{1440} & \frac{(1836b+14580a+7295)}{2160} & \frac{-(4104b+32400a+12785)}{4320} & \frac{(81b+675a+335)}{540} \\ \frac{-(108b+756a+421)}{1296} & \frac{-(216b+2160a+655)}{4320} & \frac{(81b+675a+335)}{540} & \frac{-(216b+2160a-12085)}{12960} \end{bmatrix} \quad (2.61)$$

$$Q = \begin{bmatrix} \frac{-1}{2} & \frac{-(864b+6480a+305)}{4320} & \frac{(216b+1620a+725)}{540} & \frac{-(864b+6480a+3335)}{4320} & 0 & 0 \\ \frac{(864b+6480a+305)}{4320} & 0 & \frac{-(864b+6480a+2315)}{1440} & \frac{(108b+810a+415)}{270} & 0 & 0 \\ \frac{-(216b+1620a+725)}{540} & \frac{(864b+6480a+2315)}{1440} & 0 & \frac{-(864b+6480a+785)}{4320} & \frac{-1}{12} & 0 \\ \frac{(864b+6480a+3335)}{4320} & \frac{-(108b+810a+415)}{270} & \frac{(864b+6480a+785)}{4320} & 0 & \frac{8}{12} & \frac{-1}{12} \end{bmatrix} \quad (2.62)$$

where a and b are calculated by

$$\begin{aligned} a &= \frac{-(2177\sqrt{295369}-1166427)}{25488} \\ b &= \frac{(66195\sqrt{53}\sqrt{5573}-35909375)}{101952} \end{aligned} \quad (2.63)$$

A 4th order alternative to the [Carpenter et al. \(1998\)](#) boundary closure has also been implemented into OpenSBLI for first derivatives. The scheme of [Ekaterinaris \(2005\)](#) is also 4th order accurate at the boundaries but is $\sim 10\%$ faster computationally due to reduced data access. For first order derivatives, the scheme modifies two points at either the $i = 0$ or $i = N - 1$ boundary such that

$$f'_0 = \frac{1}{12\Delta x} (-25f_0 + 48f_1 - 36f_2 + 16f_3 - 3f_4), \quad (2.64)$$

$$f'_1 = \frac{1}{12\Delta x} (-3f_0 - 10f_1 + 18f_2 - 6f_3 + f_4), \quad (2.65)$$

$$f'_{N-2} = \frac{1}{12\Delta x} (-f_{N-5} + 6f_{N-4} - 18f_{N-3} + 10f_{N-2} + 3f_{N-1}), \quad (2.66)$$

$$f'_{N-1} = \frac{1}{12\Delta x} (3f_{N-5} - 16f_{N-4} + 36f_{N-3} - 48f_{N-2} + 25f_{N-1}). \quad (2.67)$$

For both boundary schemes the second derivatives are computed for the final two interior points from [Carpenter et al. \(1998\)](#) using the formula

$$f''_0 = \frac{35f_0 - 104f_1 + 114f_2 - 56f_3 + 11f_4}{12\Delta^2}, \quad (2.68)$$

$$f''_1 = \frac{11f_0 - 20f_1 + 6f_2 + 4f_3 - f_4}{12\Delta^2}. \quad (2.69)$$

To be able to simulate problems on curvilinear meshes, a system of generalised coordinate transformations were implemented into OpenSBLI following the guidelines of [Rumsey and Biedron](#)

(2013). The transformation maps the coordinates (x, y, z) to a new system of (ξ, η, ζ) such that

$$\begin{aligned}\xi &= \xi(x, y, z) \\ \eta &= \eta(x, y, z) \\ \zeta &= \zeta(x, y, z)\end{aligned}\tag{2.70}$$

which allows derivatives to be expressed using the chain-rule as

$$\begin{aligned}\frac{\partial}{\partial x} &= \xi_x \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta} + \zeta_x \frac{\partial}{\partial \zeta} \\ \frac{\partial}{\partial y} &= \xi_y \frac{\partial}{\partial \xi} + \eta_y \frac{\partial}{\partial \eta} + \zeta_y \frac{\partial}{\partial \zeta} \\ \frac{\partial}{\partial z} &= \xi_z \frac{\partial}{\partial \xi} + \eta_z \frac{\partial}{\partial \eta} + \zeta_z \frac{\partial}{\partial \zeta}.\end{aligned}\tag{2.71}$$

The components $(\xi_x, \eta_x, \zeta_x, \xi_y, \eta_y, \zeta_y, \xi_z, \eta_z, \zeta_z)$ are the metric terms that quantify how the grid is changing. The Jacobian of the transformation is defined as

$$J = \frac{\partial(\xi, \eta, \zeta)}{\partial(x, y, z)} = \begin{vmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{vmatrix},\tag{2.72}$$

with its inverse

$$\frac{1}{J} = \frac{\partial(x, y, z)}{\partial(\xi, \eta, \zeta)} = \begin{vmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{vmatrix}.\tag{2.73}$$

The metric terms are related to the derivatives of the (x, y, z) coordinates by

$$\begin{aligned}\xi_x &= J(y_\eta z_\zeta - y_\zeta z_\eta) \\ \xi_y &= J(x_\zeta z_\eta - x_\eta z_\zeta) \\ \xi_z &= J(x_\eta y_\zeta - x_\zeta y_\eta) \\ \eta_x &= J(y_\zeta z_\xi - y_\xi z_\zeta) \\ \eta_y &= J(x_\xi z_\zeta - x_\zeta z_\xi) \\ \eta_z &= J(x_\zeta y_\xi - x_\xi y_\zeta) \\ \zeta_x &= J(y_\xi z_\eta - y_\eta z_\xi) \\ \zeta_y &= J(x_\eta z_\xi - x_\xi z_\eta) \\ \zeta_z &= J(x_\xi y_\eta - x_\eta y_\xi),\end{aligned}\tag{2.74}$$

and are evaluated using the 4th order central scheme and boundary closure as described in (2.59). In the case of central schemes in OpenSBLI the transformation is applied symbolically to each of the derivative terms. The derivatives are calculated on a uniform spacing as in (2.59) and the metric terms are then multiplied into the derivative storage arrays in the right-hand-side of the governing equations during the time update. The same procedure is applied when using the shock-capturing schemes, except that the convective terms of (2.1) are written explicitly with the metric terms included. This gives the inviscid flux terms of (2.1) in vector form (F, G, H) for the (x, y, z) directions as

$$F = \frac{1}{J} \begin{bmatrix} \rho U \\ \rho uU + \xi_x p \\ \rho vU + \xi_y p \\ \rho wU + \xi_z p \\ (\rho E + p)U \end{bmatrix}, G = \frac{1}{J} \begin{bmatrix} \rho V \\ \rho uV + \eta_x p \\ \rho vV + \eta_y p \\ \rho wV + \eta_z p \\ (\rho E + p)V \end{bmatrix}, H = \frac{1}{J} \begin{bmatrix} \rho W \\ \rho uW + \zeta_x p \\ \rho vW + \zeta_y p \\ \rho wW + \zeta_z p \\ (\rho E + p)W \end{bmatrix} \quad (2.75)$$

where U, V, W are the contravariant velocities defined as

$$U = \xi_x u + \xi_y v + \xi_z w, \quad (2.76)$$

$$V = \eta_x u + \eta_y v + \eta_z w, \quad (2.77)$$

$$W = \zeta_x u + \zeta_y v + \zeta_z w. \quad (2.78)$$

Viscous fluxes are given by

$$F_v = \frac{1}{J} \begin{bmatrix} 0 \\ \xi_x \tau_{xx} + \xi_y \tau_{xy} + \xi_z \tau_{xz} \\ \xi_x \tau_{xy} + \xi_y \tau_{yy} + \xi_z \tau_{yz} \\ \xi_x \tau_{xz} + \xi_y \tau_{yz} + \xi_z \tau_{zz} \\ \xi_x b_x + \xi_y b_y + \xi_z b_z \end{bmatrix}, G_v = \frac{1}{J} \begin{bmatrix} 0 \\ \eta_x \tau_{xx} + \eta_y \tau_{xy} + \eta_z \tau_{xz} \\ \eta_x \tau_{xy} + \eta_y \tau_{yy} + \eta_z \tau_{yz} \\ \eta_x \tau_{xz} + \eta_y \tau_{yz} + \eta_z \tau_{zz} \\ \eta_x b_x + \eta_y b_y + \eta_z b_z \end{bmatrix}, \quad (2.79)$$

$$H_v = \frac{1}{J} \begin{bmatrix} 0 \\ \zeta_x \tau_{xx} + \zeta_y \tau_{xy} + \zeta_z \tau_{xz} \\ \zeta_x \tau_{xy} + \zeta_y \tau_{yy} + \zeta_z \tau_{yz} \\ \zeta_x \tau_{xz} + \zeta_y \tau_{yz} + \zeta_z \tau_{zz} \\ \zeta_x b_x + \zeta_y b_y + \zeta_z b_z \end{bmatrix}, \quad (2.80)$$

where for the heat-flux q_k (2.4) and stress-tensor τ_{ij} (2.5), we have

$$\begin{aligned} b_x &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + q_x, \\ b_y &= u\tau_{xy} + v\tau_{yy} + w\tau_{yz} + q_y, \\ b_z &= u\tau_{xz} + v\tau_{yz} + w\tau_{zz} + q_z. \end{aligned} \quad (2.81)$$

2.7 Time advancement

Large-scale direct numerical simulation (DNS) of the compressible Navier-Stokes equations has considerable memory requirements, making low-storage time-advancement schemes an attractive option to tackle challenging flows (Kennedy et al., 2000). Furthermore, explicit methods avoid having to compute the expensive inversion of systems required by implicit schemes. Explicit methods with structured meshes are well suited to modern computational hardware options such as GPUs, as they avoid performance issues related to poor data locality. OpenSBLI currently has two low-storage time-stepping schemes available, a standard 3rd order Runge-Kutta scheme in the form proposed by Wray (1990), and an alternative Williamson (1980) formulation.

The alternative formulation has been implemented for 3rd and 4th order, plus a 3rd order strong-stability-preserving (SSP) version to improve stability for flows containing discontinuities. Due to the reduced global data access of the [Williamson \(1980\)](#) formulation within OPS, this method is also computationally more efficient than the algorithm described in [Wray \(1990\)](#). The [Williamson \(1980\)](#) formulation requires only two storage registers per equation and is implemented as follows. For an m -stage scheme, time advancement of the solution vector u from time level u^n to u^{n+1} is performed at stage $i = 1, \dots, m$ such that

$$du^{(i)} = A_i du^{(i-1)} + \Delta t R(u^{(i-1)}), \quad (2.82)$$

$$u^{(i)} = u^{(i-1)} + B_i du^{(i)}, \quad (2.83)$$

$$u^{n+1} = u^{(m)}, \quad (2.84)$$

for a constant time-step Δt , initial conditions $u^{(0)} = u^n$ and $du^{(0)} = 0$, and residual $R(u^{(i-1)})$. The 3rd and 4th order schemes have three and five sub-stages per iteration respectively. The coefficients A_i and B_i are taken from [Carpenter and Kennedy \(1994\)](#) for 3rd order as

$$(A_1, A_2, A_3) = \left(0, -\frac{5}{9}, -\frac{153}{128}\right) \quad \text{and} \quad (B_1, B_2, B_3) = \left(\frac{1}{3}, \frac{15}{16}, \frac{8}{15}\right), \quad (2.85)$$

and for 4th order as

$$(A_1, B_1) = (0, 0.1496590219993), \quad (2.86)$$

$$(A_2, B_2) = (-0.4178904745, 0.3792103129999), \quad (2.87)$$

$$(A_3, B_3) = (-1.192151694643, 0.8229550293869), \quad (2.88)$$

$$(A_4, B_4) = (-1.697784692471, 0.6994504559488), \quad (2.89)$$

$$(A_5, B_5) = (-1.514183444257, 0.1530572479681). \quad (2.90)$$

Note that for this formulation the A_1 coefficient is always zero, meaning that the previous $du^{(i-1)}$ term does not need to be zeroed at the start of each iteration as in other formulations. As outlined in [Carpenter and Kennedy \(1994\)](#), the constants A_i, B_i are part of a family of solutions that take the form

$$z_1 = \sqrt{36c^4 + 36c^3 - 135c^2 + 84c - 12} \quad (2.91)$$

$$z_2 = 2c^2 + c - 2 \quad (2.92)$$

$$z_3 = 12c^4 - 18c^3 + 18c^2 - 11c + 2 \quad (2.93)$$

$$z_4 = 36c^4 - 36c^3 + 13c^2 - 8c + 4 \quad (2.94)$$

$$z_5 = 69c^3 - 62c^2 + 28c - 8 \quad (2.95)$$

$$z_6 = 34c^4 - 46c^3 + 34c^2 - 13c + 2. \quad (2.96)$$

The value of c represents an intermediate time-level coefficient from the standard Runge-Kutta Butcher diagrams (Butcher, 1987), and can be optimised based on certain constraints (Carpenter and Kennedy, 1994). One such example is for the family of strong-stability-preserving (SSP) RK schemes of Gottlieb and Shu (1998). The optimal value of c to enforce the SSP property is taken from Gottlieb and Shu (1998) to be $c = 0.924574$. The resulting Runge-Kutta SSP coefficients are given by

$$A_1 = 0 \quad (2.97)$$

$$A_2 = \frac{-z_1(6c^2 - 4c + 1) + 3z_3}{(2c + 1)z_1 - 3(c + 2)(2c - 1)^2} \quad (2.98)$$

$$A_3 = \frac{-z_1z_4 + 108(2c - 1)c^5 - 3(2c - 1)z_5}{24z_1c(c - 1)^4 + 72cz_6 + 27c^6(2c - 13)} \quad (2.99)$$

$$B_1 = c \quad (2.100)$$

$$B_2 = \frac{12c(c - 1)(3z_2 - z_1) - (3z_2 - z_1)^2}{144c(3c - 2)(c - 1)^2} \quad (2.101)$$

$$B_3 = \frac{-24(3c - 2)(c - 1)^2}{(3z_2 - z_1)^2 - 12c(c - 1)(3z_2 - z_1)}, \quad (2.102)$$

This chapter has outlined the governing compressible Navier-Stokes equations and explained the numerical methods used in this thesis to solve them. Modern high-order Weighted and Targeted Essentially Non-Oscillatory (WENO/TENO) schemes have been selected to perform the task of shock-capturing. The next chapter discusses their implementation within the OpenSBLI code and how these schemes can be selected within a user-defined problem script. Chapter 4 presents a suite of validation cases, that demonstrates the accuracy and correctness of the numerical methods discussed in this chapter.

Chapter 3

OpenSBLI: automatic code-generation for computational fluid dynamics

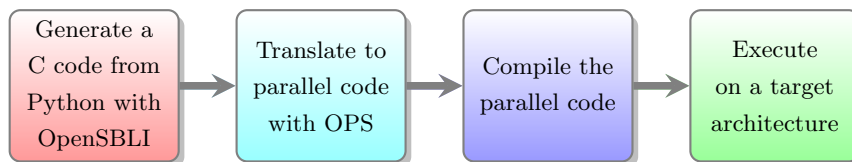
3.1 Introduction and motivation

Academic codes are challenging to maintain and often lack the resources associated with commercial software engineering projects. Furthermore, they often contain large amounts of legacy code which can be inflexible to changes in computational hardware, such as the recent growth in usage of graphical programming units (GPUs) in scientific High-Performance-Computing (HPC) ([Nvidia GPU applications, 2019](#)). One of the codes developed in the Aerodynamics and Flight Mechanics group at the University of Southampton is the Fortran-based SBLI code. SBLI has been in active development and use over the past two decades and has been applied to a variety of challenging flow problems, including in [Sansica \(2015\)](#) for laminar and transitional shock-wave/boundary-layer interactions. SBLI is a 4th order explicit finite-difference code on structured meshes that applies a total-variation-diminishing (TVD) scheme to capture shock-waves. The first half of the present PhD project focused on further development of an open-source successor to the SBLI code called OpenSBLI ([Lusher et al., 2018b](#)).

OpenSBLI aims to be a sustainable and modern compressible computational fluid dynamics code that is able to target a wide range of computational hardware. OpenSBLI is an automatic code-generation framework written in the high-level language Python. The code-generation framework uses the symbolic algebra library SymPy ([Meurer et al., 2017](#)) to perform manipulations on user-specified equations in a high-level script. The equations are discretized symbolically and converted into low-level C code for compilation. An initial proof-of-concept version of OpenSBLI was developed in [Jacobs et al. \(2017\)](#). The first version was limited to periodic subsonic cases with central differencing, without the ability to capture shock-waves. It demonstrated however that code-generation techniques could be a viable approach for computational science. At the start of this project a version 2 of the OpenSBLI code was written from scratch alongside Dr Satya P. Jammy. Using lessons learnt from the first version of the code, the current version of

OpenSBLI began from a new code base with this project. This chapter discusses the structure of OpenSBLI version 2, including the implementation of shock-capturing routines as published in Lusher et al. (2018b). Some of the features described in this thesis were implemented into the code independently by the author in the second half of the project. These include the standard and adaptive Targeted Essentially Non-Oscillatory (TENØ) schemes, hybrid methods, time-stepping schemes, boundary-layer initialisations, and wall boundary conditions and closures.

Code-generation techniques enable what is known as a *separation of concerns*: the separation of the physical problem and numerical schemes from their parallel implementation. This allows users to focus on their particular area of expertise without distraction. The size of the code base is also greatly reduced compared to static codes, and is therefore easier to maintain. As all of the equations are manipulated symbolically prior to writing the simulation code, code-generation also enables performance optimizations that would not be feasible in a hand-written code (Jammy et al., 2016). Another concern inherent to all HPC codes is the efficiency of the parallel programming implementation. To perform large-scale Direct Numerical Simulations (DNS) on HPC clusters, OpenSBLI utilizes the Oxford Parallel Structured (OPS) embedded domain-specific-language (DSL) (Reguly et al., 2014; Mudalige et al., 2019). Previous work (Mudalige et al., 2015) has shown that this system is able to match or exceed the performance of hand-written code. The OPS library is a programming abstraction for multi-block structured mesh applications. C code is written by OpenSBLI with calls to certain functions defined in the OPS library to perform computations on massively-parallel systems. Each call to a parallel region in the source-code is translated automatically by OPS to a variety of parallel programming paradigms, including MPI, OpenMP, CUDA, OpenCL and OpenACC. The four main steps for generating and running a simulation code in OpenSBLI can be summarised as



Although the OpenSBLI code is still relatively new in its development cycle, the project has attracted users both internally and in other institutes worldwide. Part of the PhD project has involved the author supporting these efforts and providing training for the code. A user manual has been written which covers many of the themes discussed in this thesis. A notable component of the user manual is the development guide given in appendix C, which discusses the SymPy functionality required for development of OpenSBLI.

OpenSBLI is one of the five open-source codes selected for support by the UK Turbulence Consortium (UKTC) (<https://www.ukturbulence.co.uk/flow-solvers.html>), along with PyFR, Incompact3D, Nektar++, and Code_Saturne. OpenSBLI has been used for benchmarking of the UK national CPU-based HPC facilities ARCHER/ARCHER2 (<https://github.com/hpc-uk/archer-benchmarks>), and for emerging HPC architectures such as the ARM-based clusters in McIntosh-Smith et al. (2019). OpenSBLI has also been deployed as a teaching asset in the SESA6061 Turbulence: Physics and modelling module at the University of Southampton. Students are provided with a GNU/Linux virtual machine with OpenSBLI installed, which they run using the VirtualBox (<https://www.virtualbox.org/>) software on university workstations.

Students gain valuable experience in using a research DNS code, performing coursework tasks to extract and post-process turbulent data for the Taylor-Green vortex (TGV) case (Brachet et al., 1983) described in chapter 5.

3.2 Structure of an OpenSBLI problem script

The structure of an OpenSBLI problem script is presented in this section, with code snippets to demonstrate the steps a user needs to perform when generating a code. OpenSBLI classes and data structures are given in italics throughout the text, and inheritance from a parent class is displayed in parentheses. OpenSBLI makes use of fundamental components of the symbolic algebra library SymPy (Meurer et al., 2017), to construct and discretize symbolic expressions during the code-generation process. To cater to the specific requirements of a CFD code, OpenSBLI introduces a number of symbolic objects that are a superset of the base SymPy building blocks.

The OpenSBLI objects inherit mathematical properties such as associativity and commutativity from SymPy, but contain the additional functionality required to build a CFD code. Indexed representations of symbolic quantities allow for discretized versions of the user-specified equations to be built. To set up a simulation case in OpenSBLI, the user must understand the fundamental components of the problem specification script. This high-level Python script defines all of the equations, numerical schemes, boundary conditions, and simulation parameters required to create a complete CFD solver. For this chapter the example code is given for a two-dimensional SBLI problem using WENO schemes. The applications folder in the source-code and the user manual in the Appendix should be consulted for further guidance on setting up other types of problems.

3.2.1 List of the required components

The steps below summarise the main components of the problem script, which will be explained in greater detail in the subsequent sections of this chapter.

1. Set the number of dimensions for the problem.
2. Write the governing equations as strings in index notation.
3. Parse the strings into symbolic equations and expand over the free indices.
4. Provide equations for any undefined quantities appearing in the governing equations (pressure, temperature, viscosity, ...)
5. Create a *SimulationBlock* for the equations to be solved on.
6. For non-uniform grids: create a metric transformation class and use it to transform the governing equations.
7. Select the spatial and temporal numerical schemes to be used.
8. Specify the boundary conditions required to model the physical problem.
9. Create equations for the initial condition and grid distributions.

10. Specify the input/output (IO) options for reading/writing of simulation data files.
11. Set the equations, schemes, and IO class on the simulation block.
12. Call the discretisation routine from the simulation block.
13. Create an algorithm and call the code-writer class to create the OPS C code.
14. Substitute numerical values for the simulation parameters ($Re, Pr, M_\infty, \Delta x_i, \Delta t, \dots$).

The discretisation of the governing equations only occurs at step 12, prior to this step the user is only selecting the options to be used. While each of these steps must be completed to generate a code in OpenSBLI, for some problem cases the ordering of the steps may change slightly. It is important to note that the generation of the OPS C code is only performed at step 13. This means that all of the equation handling and numerical methods specific to the problem are independent of their execution up until the point of code writing. To modify the parallel execution of the code to a different domain-specific-language, the only part of OpenSBLI to change would be the OPS C code writer class. A schematic of the main components of OpenSBLI and how they relate to the OPS library is given in figure 3.1.

OpenSBLI Framework

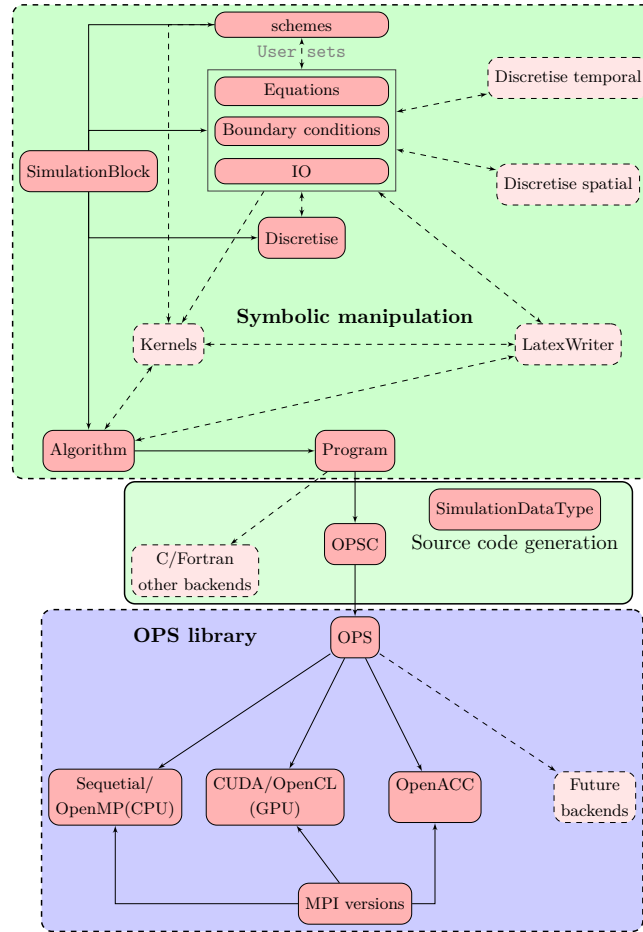


FIGURE 3.1: Schematic of the major components in the OpenSBLI automatic source code generation framework.

3.2.2 Defining the equations

The standard method for specifying the equations to be solved in OpenSBLI is via Python strings. The contents of the strings are parsed and converted into symbolic expressions which can then be manipulated as required. Once parsed, the equations are expanded over their free indices for the number of dimensions selected by the user. The following sections illustrate how this is done in practice. All the base equations are provided by the user as strings in the SymPy equality class *Eq*, which takes the convention

$$\text{Eq}(\text{Temporal derivatives, Spatial derivatives}) \implies \text{Temporal derivatives} = \text{Spatial derivatives}. \quad (3.1)$$

The temporal derivatives on the left-hand side of the equation are set equal to the spatial derivatives on the right-hand side of the equation. For example the continuity equation in (2.1)

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u_j)}{\partial x_j} = 0, \quad j = 0, \dots, 2, \quad (3.2)$$

would be defined in OpenSBLI as

```
mass = "Eq(Der(rho,t), - Conservative(rhou_j,x_j))"
```

The negative sign refers to the fact that the spatial derivative has been moved to the right-hand side of the equation. Non-zero source terms would be added to the right-hand side of the equation with a positive sign. A selection of derivative operators is available in OpenSBLI depending on the application. *Der* is a standard derivative, *Conservative* performs a conservative derivative after multiplying the product terms, and *Skew* is available for skew-symmetric splitting formulations. Second derivatives are evaluated with explicit second derivative formulae, rather than two applications of a first derivative.

3.2.3 Parsing and expansion of the equations

The code below defines the mass, momentum, and energy equations defined in the compressible Navier-Stokes equations (2.1). For each of the derivative operators a third argument can be passed to specify the type of scheme that should be used for discretisation. In this example the user is specifying that these terms should be *TenoDerivatives* for the TENO schemes in section 2.4. Alternatively the argument here could be of type *Central* or *Weno*. If no scheme is passed as in the case of the heat flux, then OpenSBLI will default to a central derivative operator.

```
ndim = 2
sc1 = "**{'scheme': 'Teno'}"
# Define the compressible Navier-Stokes equations in Einstein notation.
mass = "Eq(Der(rho,t), - Conservative(rhou_j,x_j,%s))" % sc1
momentum = "Eq(Der(rhou_i,t) , -Conservative(rhou_i*u_j + KD(_i,_j)*p,x_j , %s) +
               Der(tau_i_j,x_j) )" % sc1
energy = "Eq(Der(rhoE,t), - Conservative((p+rhoE)*u_j,x_j, %s) - Der(q_j,x_j) +
               Der(u_i*tau_i_j ,x_j) )" % sc1
```

```

stress_tensor = "Eq(tau_i_j, (mu/Re)*(Der(u_i,x_j)+ Der(u_j,x_i) - (2/3)* KD(_i,_j)*
    Der(u_k,x_k)))"
heat_flux = "Eq(q_j, (-mu/((gama-1)*Minf*Minf*Pr*Re))*Der(T,x_j))"
# Substitutions
substitutions = [stress_tensor, heat_flux]
constants = ["Re", "Pr", "gama", "Minf", "SuthT", "RefT"]

```

EXAMPLE CODE 3.1: Defining the governing equations in OpenSBLI.

Constants that appear in the equations such as Reynolds number and Prandtl number are defined in a list at this stage. Additional mathematical operators such as *KD*, *Dot* and *LC* are defined in OpenSBLI for the Kronecker delta, vector dot product and Levi-Civita symbol respectively. To avoid errors when specifying equations, OpenSBLI supports substitutions to be defined for the governing equations. In this example the heat flux and stress tensor terms are written as separate equations and added to a substitutions list to be added once the equations are parsed and expanded.

```

# Formulas for the variables used in the equations
velocity = "Eq(u_i, rho_u_i/rho)"
pressure = "Eq(p, (gama-1)*(rhoE - rho*(1/2)*(KD(_i,_j)*u_i*u_j)))"
speed_of_sound = "Eq(a, (gama*p/rho)**0.5)"
temperature = "Eq(T, p*gama*Minf*Minf/(rho))"
viscosity = "Eq(mu, (T**(1.5)*(1.0+SuthT/RefT)/(T+SuthT/RefT)))"

```

EXAMPLE CODE 3.2: Defining the evaluation of constituent relations in OpenSBLI.

The evaluation of undefined quantities appearing in the governing equations are known in OpenSBLI as *ConstituentRelations*. In the example above the user is providing equations for the primitive variables, speed of sound, and the dynamic viscosity via Sutherland's law (2.6). The ordering of the equations does not matter: once the code-generation process starts, the constituent relations are sorted based on their dependencies. The output OPS C code would evaluate for example the velocity components before the pressure equation. A similar sorting process is applied for simulation constants that may have relative dependencies on each other.

The expansion process is separate from the definition of the equations, giving the user the flexibility of providing either the full equations term by term, or compact equations in index notation to be expanded. The *EinsteinEquation* class performs expansion, and requires the equation to expand, the number of dimensions, a coordinate symbol to expand over, and any substitutions and constants appearing in the equations. In the code example below a standard Python loop is used to iterate over each of the governing equations and constituent relations and store the expanded versions in lists.

```

# Define coordinate direction symbol (x) this will be x_i, x_j, x_k
coordinate_symbol = "x"
# Instantiate equation classes
eq = EinsteinEquation()
base_eqns = [mass, momentum, energy]
constituent_eqns = [velocity, pressure, speed_of_sound, temperature, viscosity]

```

```

# Expand the base equations
for i, base in enumerate(base_eqns):
    base_eqns[i] = eq.expand(base, ndim, coordinate_symbol, substitutions, constants)
# Expand the constituent relations
for i, CR in enumerate(constituent_eqns):
    constituent_eqns[i] = eq.expand(CR, ndim, coordinate_symbol, substitutions,
    constants)

```

EXAMPLE CODE 3.3: Parsing and expansion of equations in OpenSBLI.

3.2.4 Creating a simulation block

```

# Create a simulation block
block = SimulationBlock(ndim, block_number = 0)

```

EXAMPLE CODE 3.4: Creating a simulation block in OpenSBLI.

A block is created with the *SimulationBlock* class, which generates place-holder grid constants ($N_{x_i}, \Delta x_i$) for the selected number of dimensions. Future releases of OpenSBLI will include the capability for multi-block applications. For those cases multiple *SimulationBlocks* would be instantiated in the same way but with an incremented block number. The purpose of the block is to link all of the components of the simulation. Equations, boundary conditions, and numerical schemes must all be set on the block they are to be used on, prior to calling the discretisation routine in section 3.2.10. As will become clear in section 3.3.4, the block plays an important role when determining the type of symbolic objects within an expression.

By default when equations are created in the problem script, flow variables to be stored globally are of type *DataObject*. *DataObjects* are block-independent quantities that could be used (set) on multiple blocks. Once an equation is set on a specific block, all of the *DataObjects* in an expression are converted into *DataSets* defined on that block. As an example, a *DataObject* for pressure p would be represented as a *DataSet* on block 0 as $p_B0[0,0,0]$ in three dimensions. The *DataSet* has gained a ‘_B0’ suffix and is now an indexed object that can be incremented in each spatial direction to build discrete representations of operators.

3.2.5 Creating equation classes and coordinate transformations

```

# Create Simulation Equations and Constituent relations classes and add the previously
# expanded equations
simulation_eq = SimulationEquations()
constituent = ConstituentRelations()
for eqn in base_eqns:
    simulation_eq.add_equations(eqn)
for eqn in constituent_eqns:
    constituent.add_equations(eqn)

```

EXAMPLE CODE 3.5: Using the *SimulationEquations* and *ConstituentRelations* classes in OpenSBLI.

Having created and expanded the governing time-advance equations and constituent relations, the next step is to add them to their respective classes in OpenSBLI. Once the classes have been instantiated, the equations are iterated over and added to the class with the *add_equations* routine. Both of these equation classes will later be set on the simulation block to be discretized for the chosen numerical schemes.

```
# Create a metrics class and generate the transformations
metric_eq = MetricsEquation()
metric_eq.generate_transformations(ndim, coordinate_symbol, [(False, False), (True,
    False)], 2)
simulation_eq.apply_metrics(metric_eq)
```

EXAMPLE CODE 3.6: Generating metric transformations in OpenSBLI.

For problems involving non-uniform meshes, the user must invoke the metric transformation class. The *generate_transformations* routine creates the metric transformations defined in section 2.6 depending on the user input. Input arguments include the number of dimensions and the coordinate symbol used for the governing equations. The number two signifies that first and second derivatives are to be computed.

Each Python tuple in the third input argument controls whether the transformation is stretched or curvilinear in a given direction. While grid stretching can be applied to any direction independently, at least two directions must be specified to enable curvilinear coordinates. The example above is uniform (False, False) in the x_0 direction and stretched but orthogonal (True, False) in x_1 . A two-dimensional curvilinear problem would have [(True, True), (True, True)]. This allows OpenSBLI to generate the minimum number of transformations required for a given problem. For a three-dimensional case that is curvilinear in two dimensions and uniform in the third, the input argument would be [(True, True), (True, True), (False, False)]. Once the metrics have been created, they can be used to transform the governing equations with the *apply_metrics* routine. Alternatively, individual equations may be transformed by applying the *apply_transformation* routine from the metric class.

3.2.6 Numerical scheme selection

```
schemes = {}
teno_order = 5
Avg = RoeAverage([0, 1])
LLF = LLFTeno(teno_order, averaging=Avg)
schemes[LLF.name] = LLF
cent = Central(4)
schemes[cent.name] = cent
rk = RungeKuttaLS(3, formulation='SSP')
schemes[rk.name] = rk
block.set_discretisation_schemes(schemes)
```

EXAMPLE CODE 3.7: Spatial and temporal scheme selection in OpenSBLI.

At this stage the numerical scheme options should be selected by the user. An empty Python dictionary is created to store the scheme choices with the name of the schemes taken as the dictionary keys. A 5th order TENO scheme with local Lax-Friedrich flux splitting is selected by instantiating the *LLFTeno* class. It is essential that the scheme chosen here is consistent with the type of derivatives selected in section 3.2.2. The shock-capturing requires an averaging procedure for the characteristic decomposition. Either the *SimpleAverage* or *RoeAverage* class must be selected as one of the input arguments. The *LLFTeno* object can then be added to the scheme dictionary.

The same procedure is applied for any other numerical schemes to be used. In this viscous case a *Central* scheme of order 4 is selected to calculate the heat flux, metrics, and viscous terms of (2.1). Changing the order of the scheme here and repeating the code-generation procedure will generate a code tailored to that scheme. A 3rd order explicit Runge-Kutta scheme is selected for time-advancement, with a strong-stability-preserving formulation. Once all of the schemes have been selected, the schemes are set on the simulation block with the *set_discretisation_schemes* routine.

3.2.7 Setting the boundary conditions

For a 2D problem we have to specify four boundary conditions. The convention we use for each boundary condition is to select the coordinate direction and then either 0 or 1 for the two possible sides in that direction. For a 2D shock-wave/boundary-layer interaction the possible combinations are [0,0] and [0,1] for the inlet and outlet in the x_0 direction, plus [1,0] and [1,1] for the no-slip wall and upper shock jump conditions in x_1 .

```

boundaries = [[0, 0] for t in range(ndim)]
# Left pressure extrapolation at x= 0, inlet conditions
direction = 0
side = 0
boundaries[direction][side] = InletPressureExtrapolateBC(direction, side)
# Right extrapolation at outlet
direction = 0
side = 1
boundaries[direction][side] = ExtrapolationBC(direction, side, order=0)
local_dict = {"block": block, "GridVariable": GridVariable, "DataObject": DataObject}
# Bottom no-slip isothermal wall
direction = 1
side = 0
wall_const = ["Minf", "Twall"]
for con in wall_const:
    local_dict[con] = ConstantObject(con)
# Isothermal wall condition
rhoE_wall = parse_expr("Eq(DataObject(rhoE),
    DataObject(rho)*Twall/(gama*(gama-1.0)*Minf**2.0))", local_dict=local_dict)
wall_eqns = [rhoE_wall]
boundaries[direction][side] = IsothermalWallBC(direction, 0, wall_eqns)
# Top dirichlet shock generator condition

```

```

direction = 1
side = 1
x_loc = parse_expr("Eq(GridVariable(x0), block.deltas[0]*block.grid_indexes[0])",
    local_dict=local_dict)
rho = parse_expr("Eq(DataObject(rho), Piecewise((1.129734572, (x0)>40.0),
    (1.00000596004, True)))", local_dict=local_dict)
rho0 = parse_expr("Eq(DataObject(rhou0), Piecewise((1.0921171, (x0)>40.0),
    (1.00000268202, True)))", local_dict=local_dict)
rho1 = parse_expr("Eq(DataObject(rhou1), Piecewise((-0.058866065, (x0)>40.0),
    (0.00565001630205, True)))", local_dict=local_dict)
rhoE = parse_expr("Eq(DataObject(rhoE), Piecewise((1.0590824, (x0)>40.0),
    (0.94644428042, True)))", local_dict=local_dict)
upper_eqns = [x_loc, rho, rho0, rho1, rhoE]
boundaries[direction][side] = DirichletBC(direction, side, upper_eqns)
block.set_block_boundaries(boundaries)

```

EXAMPLE CODE 3.8: Selecting boundary condition classes in OpenSBLI.

The first step creates a list of lists to hold the four boundary conditions. The inlet $[0,0]$ is then selected as a pressure extrapolation boundary condition that only has the direction and side as input arguments. A simple zero order spatial extrapolation condition is selected for the outlet boundary. At the bottom of the domain an isothermal no-slip wall is selected. A local dictionary with common OpenSBLI objects is created as the isothermal wall requires an equation for how the user wants to evaluate energy for this boundary condition. Two constants relating to the freestream Mach number and a constant wall temperature are created using the *ConstantObject* class. In this example the parser is used directly to parse the energy equation that has been defined by the user. The last step creates the isothermal wall boundary condition and passes the energy equation as the final function argument.

The fourth boundary condition is a Dirichlet condition to impose flow conditions before and after the oblique shock-wave. As this condition has a dependence on the x_0 coordinate we create a local equation to evaluate the current x_0 position during the simulation. The next four equations give the pre and post shock values for a $\theta = 3.08^\circ$ oblique shock at Mach 2. The upstream conditions are fixed as a boundary layer similarity solution as in section 3.3.7. The SymPy class *Piecewise* is used to create an if-else condition dependent on the local x_0 coordinate value. The parsed equations are added to a list and then passed as an argument to the *DirichletBC* class. Having selected all four boundary conditions for the problem, they are set on the block in the last line.

3.2.8 Initialisation of the domain

At this stage we need to select how the domain should be initialised and provide equations to generate a grid. In OpenSBLI a grid can either be read in from a pre-computed HDF5 file or evaluated at runtime in the initialisation kernel. For this example the grid is specified as equations and added to the initialisation.

```

# Reynolds number, Mach number and free-stream temperature for the initial profile
Re, xMach, Tinf = 950.0, 2.0, 288.0

```

```

polynomial_directions = [(False, DataObject('x0')), (True, DataObject('x1'))]
n_poly_coefficients = 50
grid_const = ["Lx1", "by"]
for con in grid_const:
    local_dict[con] = ConstantObject(con)
gridx0 = parse_expr("Eq(DataObject(x0), block.deltas[0]*block.grid_indexes[0])",
    local_dict=local_dict)
gridx1 = parse_expr("Eq(DataObject(x1),
    Lx1*sinh(by*block.deltas[1]*block.grid_indexes[1]/Lx1)/sinh(by))",
    local_dict=local_dict)
coordinate_evaluation = [gridx0, gridx1]
initial = Initialise_Katzer(polynomial_directions, n_poly_coefficients, Re, xMach,
    Tinf, coordinate_evaluation)

```

EXAMPLE CODE 3.9: Creating initial conditions in OpenSBLI.

For this SBLI case we initialise the domain with a similarity solution to impose a laminar boundary layer profile. The implementation of the similarity solution in *Initialise_Katzer* is described in section 3.3.7. The (x_0, x_1) coordinates are generated by equations that will be computed at the start of the simulation. For other simulation cases not using the boundary-layer initialisation, the *GridBasedInitialisation* class would be used here instead, with a set of user defined equations for the initial flow conditions at $t = 0$.

3.2.9 Creating the simulation input/output

```

kwargs = {'iotype': "Write"}
h5 = iohdf5(**kwargs)
h5.add_arrays(simulation_eq.time_advance_arrays)
h5.add_arrays([DataObject('x0'), DataObject('x1')])
block.setio(copy.deepcopy(h5))

```

EXAMPLE CODE 3.10: HDF5 input/output file writing in OpenSBLI.

Input/output file access in the simulation code is controlled by the *iohdf5* class. In the example above the *iohdf5* class is called with the *write* keyword to control the output of the simulation. The *add_arrays* routine is used to add the conservative time-advance quantities and the grid coordinates. Depending on the requirements of the user, additional input/output quantities can be added here with the same method. The *read* keyword would be used here to read in pre-computed grid files. The final step is to use the *setio* routine to store the file access options on the block.

3.2.10 Generating a simulation code and writing it to a C file

```

# Set equations on the block
block.set_equations([constituent, simulation_eq, initial, metriceq])
# Begin the symbolic discretisation process

```

```

block.discretise()
# Create an algorithm to order the computations in the output code
alg = TraditionalAlgorithmRK(block)
SimulationDataType.set_datatype(Double)
# Generate the OPS C code and write it to file
OPSC(alg)

```

EXAMPLE CODE 3.11: Creating an OPS C code in OpenSBLI.

Up until this point we have defined many of the components of the simulation but none of the symbolic differentiation process has been performed. The simulation equations, constituent relations, initial condition and metric transformations are now all set on the block to form a complete CFD solver. To initiate the main discretisation process we call *block.discretise*. Once this is complete, all of the governing equations will have been discretised with the chosen numerical schemes. Each of the major computations required by the simulation will have been converted into OpenSBLI kernels to be executed in parallel with OPS. Computational kernels are discussed in further detail in sections 3.3.5 and 3.4.

An algorithm class is created with the information contained on the simulation block that links all of the components together. The purpose of the algorithm class is to place the various components of the simulation in the correct part of the output C code. The simulation is then set to be double precision. Finally, the *OPSC* code-writer is called with the algorithm class. The OPSC class converts all of the symbolic quantities to OPS compliant C code. The current file directory now contains several C code files to be compiled with OPS. Further details of the OPS translation and compilation of the code are given in section 3.4.

3.2.11 Setting values for the simulation parameters

```

# Substitute simulation parameter values
constants = ['gama', 'Minf', 'Pr', 'Re', 'Twall', 'dt', 'niter', 'block0np0',
            'block0np1',
            'Delta0block0', 'Delta1block0', 'SuthT', 'RefT', 'eps', 'TEN0_CT',
            'Lx1', 'by', 'epsilon']
values = ['1.4', '2.0', '0.72', '950.0', '1.67619431', '0.04', '250000', '500', '250',
          '400.0/(block0np0-1)', '115.0/(block0np1-1)', '110.4', '288.0', '1e-15',
          '1e-5', '115.0', '5.0', '1.0e-30']
substitute_simulation_parameters(constants, values)
print_iteration_ops(NaN_check='rho_B0')

```

EXAMPLE CODE 3.12: Setting numerical values of the simulation parameters in OpenSBLI.

By default the generated code has no numerical values for simulation constants. These values are populated by using the *substitute_simulation_parameters* function. Two aligned lists must be provided that contain the name of the constant and its numerical value. Parameters such as the number of grid points, number of iterations, and the time-step are set here. The C code is now ready to compile with OPS. Once an OPS C code has been generated in OpenSBLI, manual changes can be made to the C code without needing to regenerate. A common example of this

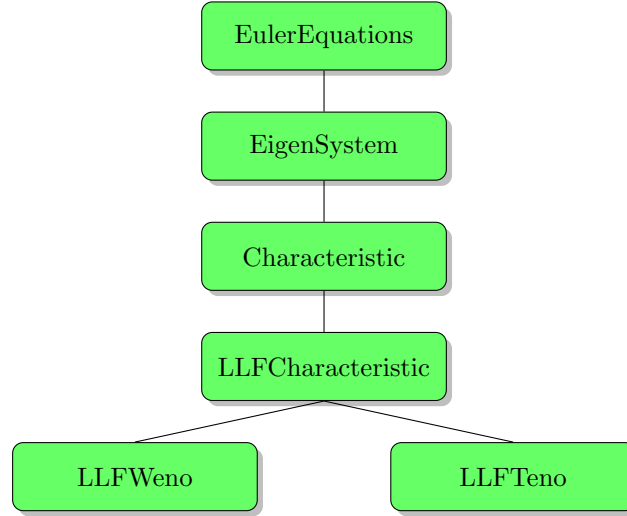


FIGURE 3.2: Characteristic decomposition classes for the shock-capturing routines.

is the modification of the simulation parameters between runs for a given C code. The last line in the example code adds an optional iteration number printer and in-simulation NaN check.

3.3 Components of the OpenSBLI implementation

One of the main challenges for this version of OpenSBLI was the addition of shock-capturing schemes within a code-generation framework. To simulate wall-bounded flows, new boundary conditions and schemes were also required. Having described the critical parts of the OpenSBLI user interface in the previous section, this section discusses more fundamental features of the code base in the following layout.

The classes that perform the characteristic decomposition and shock-capturing options in the OpenSBLI source-code are described in sections 3.3.1, 3.3.2, and 3.3.3. To be able to use OpenSBLI, it is important to understand the various types of symbolic objects used to build equations. The various types of symbolic objects are explained in section 3.3.4. Section 3.3.5 gives an overview of OpenSBLI kernels, which control the loop range and data access for each of the computations in the OPS C code. Section 3.3.6 provides a detailed explanation of the boundary conditions used for the latter chapters of this thesis. Cases that involve shock-wave/boundary-layer interactions require an initial boundary-layer profile to be imposed on the domain. Boundary-layer profiles are calculated by the numerical procedure described in section 3.3.7. Finally, functions related to the conversion of symbolic expressions to C code are covered in section 3.3.8. Further details of the implementation are also available in the user manual in the appendix.

3.3.1 Characteristic decomposition components

The classes implemented in the OpenSBLI source-code to perform the characteristic decomposition are shown in figure 3.2. Either the *LLFWeno* or *LLFTeno* class is selected by the user in the problem script. The characteristic decomposition classes are modularised to enable additional flux-splitting schemes to be added in the future. The *LLFWeno* and *LLFTeno* classes inherit

functionality from the characteristic classes and initiate either the *Weno* or *Teno* schemes for a given input order.

- *SimpleAverage(object)*: The characteristic decomposition requires the matrix components to be calculated on an averaged state. This class performs the simple arithmetic average over two successive grid points. The alternative *RoeAverage(object)* class is also available.
- *EulerEquations(object)*: The location where symbolic matrices are defined that diagonalize the Euler equations.
- *EigenSystem(object)*: A class containing routines required for manipulation of the matrices used in the characteristic decomposition of the Euler equations.
- *Characteristic(Eigensystem)*: This class contains all of the helper functions to generate symbolic/grid based variables for evaluation of the matrix terms required by the characteristic decomposition.
- *LLFCharacteristic(Characteristic)*: The main class for the LLF flux-splitting in characteristic space. The pre-process step performs the transformation of conservative variables and flux vectors into characteristic space, using the provided eigensystems. Flux splitting is then applied on the characteristic fields, in preparation for the WENO/TENO reconstructions. The post-processing section contains the routines required to transform the reconstructed flux approximation back to physical space after the WENO/TENO procedure has been applied. It also generates expressions for the evaluations of the wave-speeds for the local Lax-Friedrichs flux splitting. The characteristic wave speed is evaluated over the local stencil rather than globally over the domain.

3.3.2 WENO components

The OpenSBLI source-code contains the following classes for the WENO implementation.

- *ReconstructionVariable(object)*: An object to store the quantities produced by either the WENOJS/WENOZ classes. Allows the separation of upwind/downwind biased reconstructions.
- *ShockCapturing(object)*: The base shock-capturing class that contains functionality shared between all shock-capturing options. It contains utility functions to create the reconstructions and residual evaluation kernels. This class also finds the constituent relation formulas required by WENO/TENO (pressure, speed of sound, etc)
- *Weno(Scheme, ShockCapturing)*: The main WENO class, performs the interpolations shared between all WENO formulations.
- *ConfigureWeno(object)*: A class to set up the optimal weights and stencil points for a given order. Called from inside the main WENO class.
- *JS_smoothness(object)*: Calculates the standard Jiang-Shu smoothness indicators depending on the selected order of the scheme. Called from inside the main WENO class.

- *WenoJS(object)*: Generates the non-linear WENO weights for the original (Shu, 1997) formulation, of arbitrary odd order.
- *WenoZ(object)*: Generates the non-linear WENO weights for the improved WENO-Z schemes (Borges et al., 2008).
- *LLFWeno(LLFCharacteristic, Weno)*: The class that is instantiated in the problem script when using the WENO schemes. Contains the main discretisation calling function that is called through the simulation block in the main problem script.

3.3.3 TENO components

The TENO schemes of Fu et al. (2016, 2017) are implemented in the following classes.

- *ConfigureTeno(object)*: A class to generate all of the stencil locations, ideal weightings, and smoothness indicators from Fu et al. (2016).
- *Teno(Scheme, ShockCapturing)*: The main TENO class, performs all of the interpolations shared between the TENO schemes. Calls either the *TENO5* or *TENO6* classes depending on the input order.
- *LLFTeno(LLFCharacteristic, Teno)*: The class that is instantiated in the problem script when using the TENO schemes. Contains the main discretisation calling function that is called through the simulation block in the main problem script.

3.3.4 OpenSBLI objects

All of the expressions contained within OpenSBLI are built from the following symbolic objects.

- *DataObject* - A block-independent quantity that will be stored globally in memory in the simulation code.
- *DataSet* - A block-dependent quantity that was converted from a *DataObject* onto a specific block.
- *GridVariable* - A local temporary evaluation for a single grid point. Used to evaluate intermediate quantities that will be used to set global *DataSets*.
- *ConstantObject* - A parameter that is given a constant value during the initialisation of the simulation.
- *ConstantIndexed* - An array of indexed constant values. Used for example to store the coefficients for the Runge-Kutta schemes.
- *CoordinateObject* - An object to represent coordinates to take derivatives with respect to.

A common use case where knowledge of these object types is required would be for the implementation of a boundary condition. Boundary conditions in OpenSBLI are applied on the conservative time-advanced quantities ($\rho, \rho u, \rho v, \rho w, \rho E$). Boundary conditions often require specification of primitive variables such as pressure or temperature. A user would create equations to enforce primitive quantities as *GridVariables* locally on a certain grid point, and use them to set the conservative variables as *DataSets* which are stored globally in the simulation code.

There are three main ways of generating symbolic objects in OpenSBLI. The first is to simply call one of the OpenSBLI object classes with a string as a name:

```
p = DataObject('p')
```

This creates a *DataObject* which will later be converted into a *DataSet* defined on a block. Note that the variable p on the left hand side is simply a Python variable used to refer to this object during code generation, it has no effect on the output code. The name passed to the *DataObject* class as a string will be written out in the simulation code, so it is important to ensure that this variable has been defined elsewhere in the script. If the user has already created a *SimulationBlock*, they can create a *DataSet* directly on that block with:

```
p = block.location_dataset('p')
pprint(p)
p_B0[0,0,0]
```

Note that the *DataSet* has indices as it is an indexed object with i indices for the i number of dimensions in the problem. The *DataSet* has also been given a ‘_B0’ suffix to denote that it is defined on a block numbered 0. The third method is a quick way to create several symbols of the same type using the *symbols* class of SymPy.

```
p, u, v, w = symbols('p u v w', **{'cls': DataObject})
```

The type of object created here could be *DataObject*, *GridVariable* or *ConstantObject* depending on the application. Off-setting the location of an indexed quantity is performed by the *increment_dataset* function. In the example below a *DataSet* is created and shifted by three grid points in its second index.

```
from opensbli.utilities.helperfunctions import increment_dataset
p = block.location_dataset('p')
pprint(p)
p_B0[0,0,0] # The base [0,0,0] location for a generic stencil
direction, increment = 1, 3 # index direction 1 by 3 grid points
p_shifted = increment_dataset(p, direction, increment)
pprint(p_shifted)
p_B0[0,3,0]
```

EXAMPLE CODE 3.13: Creating and off-setting the location of indexed objects in OpenSBLI.

3.3.5 OpenSBLI kernels

All computations to be written out as OPS C code are stored inside *Kernels* within OpenSBLI. At the time of code-generation, the complete list of kernels is iterated over by the *OPSC* class in section 3.3.8. Kernels are executed in parallel through calls to the *ops-par-loop* routine described in section 3.4.5, with the provided range of evaluation and data input/outputs.

```
kernel = Kernel(block, computation_name="Example computation name")
kernel = self.set_kernel_range(kernel, block)
kernel.add_equations([Equation1, Equation2,...,])
```

EXAMPLE CODE 3.14: Creating a computational kernel in OpenSBLI.

A computational *Kernel* in OpenSBLI is created by instantiating the *Kernel* class with a descriptive name to help identify it in the simulation code. The range of the kernel is set to the grid range in the second line. Depending on the application the *halo_ranges* attribute of the kernel may also need to be set using the scheme-dependent halo information stored on the simulation block. The final line adds a list of equations to evaluate in this kernel. The ordering of the equations added to the kernel is important as this determines the order they will be evaluated in the simulation code.

3.3.6 Boundary conditions

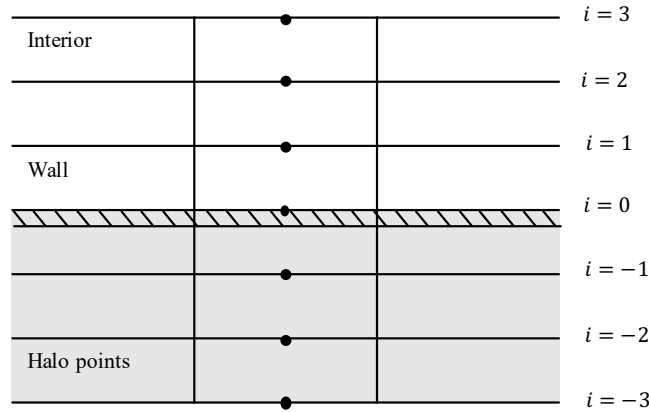


FIGURE 3.3: Schematic of the boundary condition for a no-slip wall. The wall is located at $i = 0$ where $u, v, w = 0$ to enforce a no-slip condition. The values in the halo points (grey) are extrapolated from the interior solution points.

A selection of the boundary conditions used in the proceeding chapters are listed here with their implementation. A full list of boundary conditions available in OpenSBLI is given in the appendix. Boundary conditions within OpenSBLI make use of halo points (Blazek, 2001) when shock-capturing schemes are being used. By default all boundary conditions in OpenSBLI are imposed on the conservative variables $(\rho, \rho u, \rho v, \rho w, \rho E)$ at the start of each sub-stage. Conditions can also be set for primitive variables (ρ, u, v, w, p, T) by evaluating them as local *GridVariables* which are then used to set the conservative arrays.

Figure 3.3 shows an example schematic for a no-slip viscous wall boundary condition in OpenSBLI. The wall boundary is located at $i = 0$ for side 0 or $i = N - 1$ for side 1. The no-slip condition is enforced by setting $u, v, w = 0$ on the boundary. Total energy ρE is set depending on the thermal properties of the wall in question. For a constant temperature T_w isothermal wall the energy is set for a wall density ρ_w as

$$\rho E = \frac{\rho_w T_w}{M_\infty^2 \gamma (\gamma - 1)}, \quad (3.3)$$

where the wall density is obtained from solving the continuity equation (2.1). The ghost flow in the halo points ($i < 0$) is enforced by extrapolating temperature from the interior and wall as

$$T_{-i} = (i + 1)T_w - iT_1. \quad (3.4)$$

Density in the halos is evaluated using the extrapolated temperature values and the wall pressure p_w as

$$\rho_{-1} = \frac{M_\infty^2 \gamma p_w}{T_{-i}}. \quad (3.5)$$

Momentum components in the halos are set by reflecting the velocity components from the interior flow with a reversed sign such that

$$\rho u_{-i} = -\rho_{-i} u_i, \quad (3.6)$$

$$\rho v_{-i} = -\rho_{-i} v_i, \quad (3.7)$$

$$\rho w_{-i} = -\rho_{-i} w_i, \quad (3.8)$$

where ρ_{-i} is the halo density calculated in equation (3.5). In the case of an adiabatic wall, the formula for a 4th order approximation of $\frac{\partial T}{\partial y} = 0$ is rearranged using the interior points to calculate the unknown wall temperature T_w and enforce the zero heat-flux condition. This wall temperature is then used to set the wall energy as in equation (3.3). In both cases the value of the wall density is left to float by letting the scheme solve the continuity equation to avoid over-specifying the boundary condition.

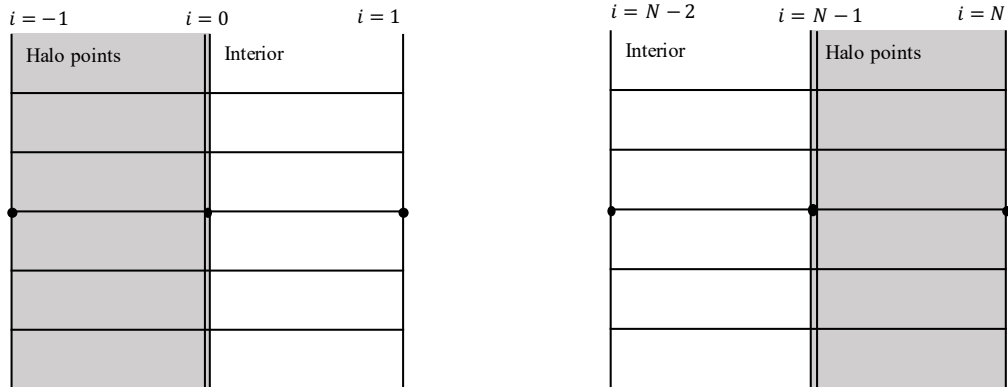


FIGURE 3.4: Schematic of the (left) inlet and (right) outlet boundary conditions. The shaded grey areas represents halo points outside of the physical domain.

Figure 3.4 shows the schematics for the inlet pressure extrapolation and outlet extrapolation boundary conditions. The outlet extrapolation in figure 3.4 (right) is a zero order spatial extrapolation of all of the flow variables from the interior point $i = N - 2$ to the boundary plane at $i = N - 1$, and all of the halo elements $i = N, N + 1, \dots$, depending on the order of the numerical scheme. The code also has an option to apply a first order linear extrapolation for the outflow, but the zero order approximation proved more stable for the supersonic outlets in this work.

The pressure extrapolation inlet method in figure 3.4 (left) follows a similar approach but places a condition on the local sound speed. For the SBLI cases in the next four chapters of this thesis, a laminar boundary-layer profile is computed for the inflow as in section 3.3.7. At the start of the simulation this initial profile is imposed in the inlet halo points $i = -1, -2, \dots$, depending on the order of the selected numerical scheme. The inlet boundary condition calculates the local speed of sound a at all points on the $i = 0$ boundary plane and compares it to the streamwise velocity component u . In supersonic regions where ($u \geq a$), the boundary condition copies the imposed profile from $i = -1$ to the inlet plane $i = 0$ for all of the conservative flow variables. In the subsonic region ($u < a$), the total energy in the halos is set based on the pressure in the interior of the domain to allow for upstream travelling waves to exit the domain. Boundary conditions are also applied on the metric terms defined in section 2.6, to populate the values of the halo points once at the start of the simulation. At a domain boundary as in figure 3.4 (left) the metric values and determinant of the Jacobian are copied symmetrically over the $i = 0$ boundary such that the value at $i = -1$ is equal to that at $i = 1$ for each of the i halo points.

3.3.7 Similarity solution for laminar boundary layers

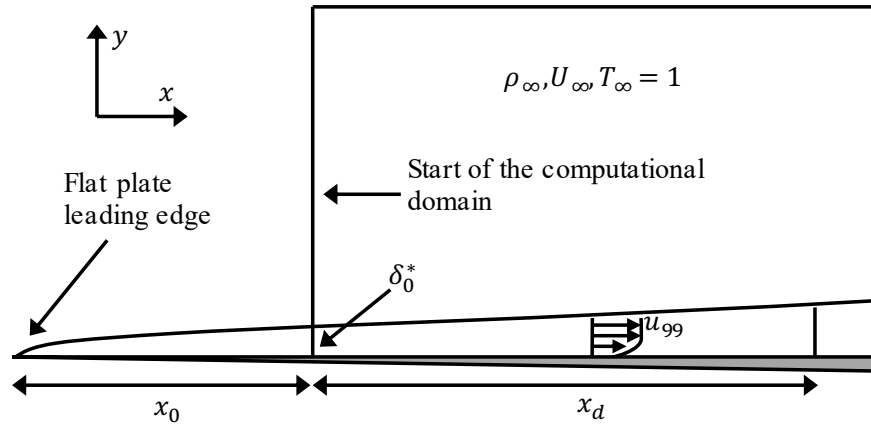


FIGURE 3.5: An initial laminar boundary-layer profile is imposed at the start of the computational domain.

Simulations of shock-wave/boundary-layer interactions require the presence of a target boundary-layer. One method for obtaining the target boundary-layer is to simply place a flat plate within a uniform inflow, and let the no-slip wall retard the oncoming fluid at the wall. This approach can lead to numerical instabilities at the discontinuous leading-edge of the plate, and may require smaller time-step values for stability in the thin initial boundary-layer. Furthermore, the sudden turning of a supersonic flow introduces a leading-edge shock into the simulation. An alternative approach is to calculate an initial profile prior to the simulation and impose it at the inlet of the

computational domain. This method is also preferred as it greatly reduces the computational domain length required to acquire a certain target boundary-layer thickness. This section describes the numerical procedures added to OpenSBLI to generate laminar boundary-layer profiles.

The source-code for the boundary-layer initialisation is contained within the *Initialise_Katzer* class, named after the validation case described in section 4.6. The initial profile is obtained via a similarity solution to the compressible boundary-layer equations, using the Illingworth transformation from White (2006), described in the appendix of Sansica (2015). Referring to figure 3.5, the numerical integration requires user input of the freestream Mach number, reference temperature, and a Reynolds number based on the inlet displacement thickness $Re_{\delta_0^*}$. The numerical integration results in a boundary-layer profile $(u(y), T(y))$ at the start of the computational domain that has developed over a distance of x_0 from the leading-edge of a flat plate. A target boundary-layer at a distance of x_d from the start of the computational domain can then be calculated as

$$Re_x = \frac{1}{2} \left(\frac{Re_{\delta_0^*}}{\Delta_s} \right)^2 + x_d Re_{\delta_0^*} \quad (3.9)$$

where Δ_s is a scaling factor obtained from the similarity solution. Density is evaluated as $\rho = 1/T$ and the wall-normal velocity profile v is calculated by integrating the continuity equation (2.1). For isothermal wall boundary conditions in a simulation, the wall temperature is set equal to the value from the similarity solution.

To be able to convert the numerical profiles $(\rho(y), u(y), v(y), T(y))$ into a form suitable for the OPS C code, a polynomial fitting procedure was applied. The polynomial fit generates symbolic high-order polynomial expressions for each of the boundary-layer profiles. The polynomials are then written into an OpenSBLI kernel to be evaluated at the start of the simulation. The same inlet profile is initialised at each x location within the domain and is left to develop as the simulation is integrated in time. For cases involving multiple adjacent boundary-layers as in chapters 6 and 7, the blending procedure described in section 6.2 is applied in the corners.

3.3.8 Algorithm and code generation

The aim of this section is to provide some insight into how OpenSBLI translates symbolic expressions in Python to compilable C code. The procedure for generating the output C code is controlled by the *Algorithm* and *OPSC* classes in the source-code. The examples in this section are selected to show how equations and OPS library calls can be generated through a high-level Python abstraction.

The *Algorithm* class in OpenSBLI controls how components are ordered in the final simulation code. It also contains classes that generate string representations of loops and other structures required to build a complete CFD code. The example below is an extract from the *DoLoop* class that defines how a ‘for loop’ should be written in C.

```
@property
def opsc_start(self):
    """ A function to generate a template for a standard C for loop."""
```

```

return "for(int %s=%s; %s<=%s; %s++)\n{" % (self.loop, str(self.loop.lower),
self.loop, str(self.loop.upper), self.loop)

```

EXAMPLE CODE 3.15: Generating sequential ‘for loops’ in the output C code.

This function is used to generate the iteration and sub-stage loops for the RK scheme. An example output would be: `for(int iter=0; iter<=niter - 1; iter++){`. Note that these are not the parallel *ops_par_loops* (section 3.4.4) used to perform computations over the grid.

The *Algorithm* class also contains a number of components used to define where computations should appear in the simulation code. Examples include *BeforeSimulationStarts*, *InTheSimulation*, and *AfterSimulationEnds*. Computations or declarations can be added as components to these location classes. The ordering within each of the location classes is also important when controlling the ordering of the output.

Aside from the translation of symbolic equations to syntactically correct C code, a major part of the *OPSC* code-writer class consists of calls to functions in the OPS library. OpenSBLI has to write strings for these OPS function calls with the appropriate input arguments. The example below is the function that declares *ops_stencils* in the C code. Stencils are the lists of integer grid locations used for data access of arrays. Stencil access within OPS is discussed in greater detail in section 3.4.3.

```

def ops_stencils_declare(self, s):
    out = []
    dtype = s.dtype.opsc()
    name = s.name + 'temp'
    sorted_stencil = s.sort_stencil_indices()
    out = [self.declare_inline_array(dtype, name, [st for st in flatten(sorted_stencil)
if not isinstance(st, Idx)])]
    out += [WriteString('ops_stencil %s = ops_decl_stencil(%d,%d,%s,\"%s\");' %
(s.name, s.ndim, len(s.stencil), name, name))]
    return out

```

EXAMPLE CODE 3.16: An example of OPS C code generation in OpenSBLI.

For a given input stencil object a sorting procedure is applied to the grid locations. The *declare_inline_array* function is used to create a simple in-line C array to hold the grid locations. The next line writes a string to declare an *ops_stencil* object in the C code and passes in the arguments required by the OPS library. The resulting output code is shown below. This example is the stencil for a 4th order central difference in the *x* direction that requires the grid locations `[-2, -1, 1, 2]`.

```

int stencil_0_01temp[] = {-2, 0, 0, -1, 0, 0, 1, 0, 0, 2, 0, 0};
ops_stencil stencil_0_01 = ops_decl_stencil(3,4,stencil_0_01temp,"stencil_0_01temp");

```

EXAMPLE CODE 3.17: The corresponding stencil declaration in the OPS C code.

The C code writing functionality in SymPy’s *C99CodePrinter* class (Meurer et al., 2017) is flexible in the sense that it allows custom definitions to be built on top of it. This gives OpenSBLI

developers the ability to control how the symbolic expressions are written out to the final simulation code. Examples of this can be found in the *opsc.py* file. For a given function or object in an OpenSBLI expression, a function in the code-writer class with the *_print_* prefix controls the form of its output.

```
def _print_Max(self, expr):
    """Print function to calculate the max of the input arguments:
    Max(a,b,c,d) is written as max(a, max(max(b,c),d)) in the output code."""
    nargs = len(expr.args)
    args_code = [self._print(a) for a in expr.args]
    for i in range(nargs-1):
        # Max of the last 2 arguments in the array
        string_max = 'fmax(%s, %s)' % (args_code[-2], args_code[-1])
        # Remove the last 2 entries and append the max of the last 2
        del args_code[-2:]
        args_code.append(string_max)
    return str(args_code[0])
```

EXAMPLE CODE 3.18: Defining custom C code printer functions in OpenSBLI.

In the example above, custom printing of the *Max* function was added to OpenSBLI. This custom print class has two main advantages: maximums of multiple input arguments can be handled in C such that *Max(a,b,c,d)* is written as *Max(a, Max(Max(b,c),d))*, and a computationally more efficient *fmax* function is called. At any point during the code writing process where a call to *Max* is found, the function call will be converted into a string in the above format.

```
def _print_Rational(self, expr):
    """Pre-computes rational constants at the start of the simulation. Expressions will
    instead contain the place-holder constant names."""
    if self.settings_opsc.get('rational', True):
        p, q = int(expr.p), int(expr.q)
        return '%d.0/%d.0' % (p, q)
    else:
        if expr in rc.existing:
            return self._print(rc.existing[expr])
        else:
            return self._print(rc.get_next_rational_constant(expr))
```

EXAMPLE CODE 3.19: Pre-computation and printing of rational constants in OpenSBLI.

A second example of where custom code printing is used in OpenSBLI is shown above. For integers p, q , SymPy represents floating point rational quantities as fractions p/q in the *Rational(p,q)* class. To avoid unnecessarily computing expensive divisions multiple times throughout the simulation code, OpenSBLI computes these factors once at the start of the simulation. Every time a rational constant is encountered in the symbolic expressions, OpenSBLI performs a substitution to one of these place-holder constants named *rcXX*.

3.4 Oxford Parallel Structured (OPS) software library

In this section the Oxford Parallel Structured Software (OPS) library (Mudalige et al., 2014, 2019) is introduced in the context of OpenSBLI. The main components of an OPS C code are discussed, with example code snippets throughout. All of the code shown in this section was generated by the current version of OpenSBLI. Further information about the OPS project and a user manual can be found at: <https://op-dsl.github.io/>. Formal definitions of all the OPS functions discussed in this section are given in the OPS user manual.

3.4.1 Introduction

OPS is an open-source library to generate low-level parallel code for multi-block structured grid applications (Mudalige et al., 2014). OPS is known as an Embedded Domain-Specific Language (EDSL), in the sense that it is an Application Programming Interface (API) embedded in a conventional programming language (C/C++, Fortran) (Mudalige et al., 2019). All of the standard features of the conventional programming language are still present, with additional functionality added specific to the domain of interest. The purpose of OPS is to automatically generate parallel versions of an input code written in the required format. It does this by parsing the input C code in Python and applying source-to-source translation to multiple parallel versions (MPI, OpenMP, CUDA, OpenCL, OpenACC).

Unlike many black-box type solvers, OPS generates parallel source-code that the user can view and then edit, enabling them to make changes at different levels within the system. The benefit of OPS within computational science is to be able to separate the parallel implementation of scientific codes from the numerical algorithms. Academic codes often target a single computational architecture and struggle to adapt to a changing computational landscape. Porting a CPU-based code to run on GPUs is extremely time consuming and error prone, taking up valuable research time and resources. By employing this separation of concerns the necessary parallel communication and work distribution is handled in a standardised way, by software developers who have much greater expertise in this area.

The currently available computational back-ends include MPI, OpenMP, MPI+OpenMP and OpenCL for CPU based architectures, and CUDA, OpenACC, and OpenCL for GPU/accelerator cards. Parallel I/O is provided by the HDF5 library, and code translators are available for C and Fortran. Performance of the OPS system has been shown to be comparable to hand-written code, with run-times typically within $\pm 10\%$ of conventional methods (Mudalige et al., 2015). The work of Mudalige et al. (2019) investigated the performance of the OpenSBLI code using OPS, against the the hand-written SBLI Fortran code. The OPS approach showed no signs of performance degradation compared to the hand-written SBLI code, performing up to 30% faster in like-for-like comparisons on a Taylor-Green-Vortex (TGV) test case. Minimal-storage algorithms (Jammy et al., 2016) enabled by the code-generation approach were also shown to provide a $2 \sim 3\times$ speed-up over the baseline cases.

3.4.2 Declaring OPS blocks, constants, and datasets

Any OPS application can be split in the simplest terms into two distinct parts: initialisation of quantities required by the simulation, and their parallel execution. The most fundamental requirements of any CFD code are the ability to allocate system memory for storage of data, and to provide an interface to access it. This data must be retrievable at later instances of the simulation to perform algebraic manipulations on.

The simplest type of memory allocation is for the declaration of constants within a simulation. In the following two code examples the code has been shortened to highlight the important components of the code preamble. Constants including integers for the size of the grid are declared at the start of the code, before being given numerical values inside the main program. Header files required by OPS are also included at this stage.

```
int block0np0;
int block0np1;
int block0np2;
#define OPS_3D
#include "ops_seq.h"
#include "opensblliblock00_kernels.h"
int main(int argc, char **argv)
{
    block0np0 = 64;
    block0np1 = 64;
    block0np2 = 64;
}
```

EXAMPLE CODE 3.20: Declaration of simulation constants in OPS.

OPS is formally initialised with a call to *ops_init*. The integer argument of 1 refers to the level of diagnostics to perform. Increasing this value from 1 to 5 will generate more performance information output through the OPS diagnostics API. Constants declared within the C file must then be passed to OPS using the *ops_decl_const* routine. The values of these constants can now be accessed globally at any point within the simulation.

```
# Initialising OPS
ops_init(argc,argv,1);
ops_decl_const("block0np0" , 1, "int", &block0np0);
ops_decl_const("block0np1" , 1, "int", &block0np1);
ops_decl_const("block0np2" , 1, "int", &block0np2);
```

EXAMPLE CODE 3.21: Initialising OPS and passing the constant declarations.

Simulation data arrays are stored in OPS by declaring datasets on a given simulation block. An OPS block is declared by the command

```
# Declare an OPS simulation block
ops_block opensblliblock00 = ops_decl_block(3, "opensblliblock00");
```

EXAMPLE CODE 3.22: Declaring a simulation block in OPS.

which would be found near the start of any OPS code. In this example an *ops_block* is created for a three-dimensional problem and is given the name *opensblliblock00*. A dataset in OPS can then be declared on this block and will contain HDF5 meta-data specific to this block. A dataset in OPS is referred to as an *ops_dat* and is declared as follows.

```
ops_dat rho_B0;
{
# Set the number of halo points and the size of the array
int halo_p[] = {5, 5, 5};
int halo_m[] = {-5, -5, -5};
int size[] = {block0np0, block0np1, block0np2};
int base[] = {0, 0, 0};
double* value = NULL;
# Declare the dataset
rho_B0 = ops_decl_dat(opensblliblock00, 1, size, base, halo_m, halo_p, value, "double",
    "rho_B0");
}
```

EXAMPLE CODE 3.23: Declaring a dataset in OPS.

A three-dimensional global dataset for density is declared with the name *rho_B0*. The size of the array is set to be the full grid range and five layers of halo points are added to pad the data on either side. The number of halo points is set within OpenSBLI and can be modified depending on the numerical schemes being used. The values of the array are set to *NULL* at the start of the simulation and will be populated during the initialisation procedure. The dataset is declared by calling the *ops_decl_dat* function for an OPS block named *opensblliblock00*.

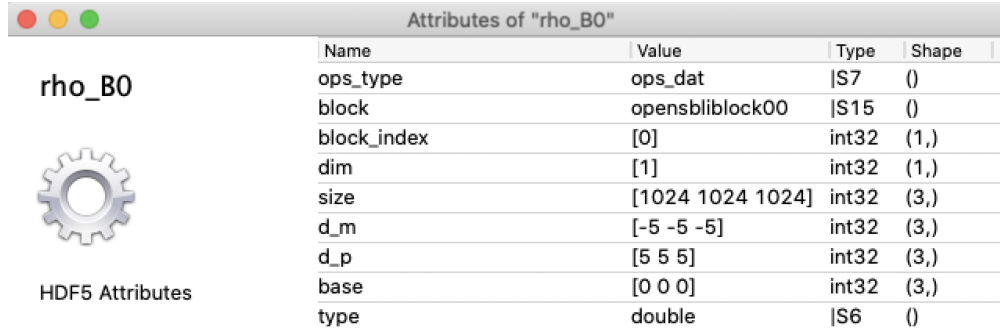
```
ops_dat rho_B0;
{
rho_B0 = ops_decl_dat_hdf5(opensblliblock00, 1, "double", "rho_B0", "data.h5");
}
```

EXAMPLE CODE 3.24: Declaring a dataset in OPS from a restart data file.

Alternatively, a dataset in OPS can be declared and initialised with the values from a previous simulation output. All input/output of data to file in OPS must be performed using the Hierarchical Data Format 5 (HDF5) library which has the ‘.h5’ file extension. In the code above the dataset is declared by the *ops_decl_dat_hdf5* routine from a file named *data.h5*. Data within HDF5 files is stored as named datasets within one of the groups in the file. To be able to initialise the data in the example above, the string argument ‘*rho_B0*’ must match one of the named datasets within the file.

It is important to note here that when restarting from a file there is no halo or array size information being explicitly passed to OPS. This is because that information is stored in the dataset itself from the previous simulation as HDF5 meta-data. Figure 3.6 shows an example of this for a three-dimensional dataset. For simulations where a grid is computed externally and read into the simulation, the HDF5 file must have all of the meta-data to be accepted by OPS.

OpenSBLI contains helper functions in the utilities directory of the source-code to apply this meta-data to pre-computed files.



Name	Value	Type	Shape
ops_type	ops_dat	S7	()
block	opensbliblock00	S15	()
block_index	[0]	int32	(1,)
dim	[1]	int32	(1,)
size	[1024 1024 1024]	int32	(3,)
d_m	[-5 -5 -5]	int32	(3,)
d_p	[5 5 5]	int32	(3,)
base	[0 0 0]	int32	(3,)
type	double	S6	()

FIGURE 3.6: HDF5 meta-data for a dataset contained in an OpenSBLI output file.

Software such as *HDF5Compass* (<https://support.hdfgroup.org/projects/compass/>) can be used to view the simulation data and HDF5 meta-data of any OpenSBLI output file.

3.4.3 OPS stencils for relative data access

```
# Declaring an OPS stencil
int stencil_0_01temp[] = {0, -1, 0, 0, 0, 0, 0, 1, 0};
ops_stencil stencil_0_01 = ops_decl_stencil(3,3,stencil_0_01temp,"stencil_0_01temp");
# Partition the domain and end the initialisation portion of the OPS code
ops_partition("");
```

EXAMPLE CODE 3.25: Stencil declaration and OPS partitioning of the domain.

OPS is a stencil-based framework that uses relative indexing to define data access patterns for each of the parallel computations to be performed. It does this by defining a set of *ops_stencils* that were introduced in section 3.3.8. The convention for the stencil is $\{x_0, y_0, z_0, \dots, x_n, y_n, z_n\}$ for accessing $i = 0, \dots, n-1$ points in the x, y, z directions. In the example above the y direction is being accessed at $[-1, 0, 1]$ grid locations relative to $[0, 0, 0]$. If manual changes are made to the equations in a code that require a change of data access, the stencil must also be updated accordingly. OpenSBLI generates all of the stencil access patterns automatically during the code generation process. The last line of the code calls the *ops_partition* routine which performs the domain decomposition based on the number of MPI processes being used. The call of the partition function denotes the end of the initialisation portion of the OPS code.

3.4.4 Parallel loops in OPS

The core component of the OPS library is its interface for parallel execution of user kernels. This interface is called an *ops_par_loop* as in the example below. Calls to a parallel region require the following information:

1. A computational kernel that defines the equations to be executed in parallel.

2. The grid range on which to perform the computation.
3. Input and output dataset arrays with their *OPS_READ/OPS_WRITE/OPS_RW* status.
4. Stencil objects defining the relative data access required on each dataset.
5. Halo information which is implicitly passed through definitions stored in the datasets.
6. An (optional) integer object *ops_arg_idx()*, if the computation explicitly requires the current (i, j, k) grid index.

```
# Grid range to execute the computation over
int iteration_range_43_block0[] = {0, block0np0, 0, block0np1, 0, block0np2};
# Call the parallel region with the appropriate input arguments
ops_par_loop(opensblliblock00Kernel043, "Convective CD p_B0 x1 ", opensblliblock00, 3,
             iteration_range_43_block0,
ops_arg_dat(p_B0, 1, stencil_0_02, "double", OPS_READ),
ops_arg_dat(wk20_B0, 1, stencil_0_00, "double", OPS_WRITE));
```

EXAMPLE CODE 3.26: An example of calling a parallel loop region in OPS

With this basic set of information OPS takes ownership of the data and is responsible for all of its decomposition, execution and movement in parallel. The way in which this is done will differ depending on the computational back-end being used (MPI, CUDA, ...). Both the data layout and execution will be re-organised by OPS in a manner most optimal for that architecture. Depending on the levels of halo points, OPS also takes control of the necessary data exchanges between MPI processes. In the example code the pressure array *p_B0* is read in over the grid range to *Kernel043*. The numbering of these kernels is generated automatically by OpenSBLI and will vary depending on the problem being solved. In this instance a central derivative is computed and the result is written to a work array *wk20_B0*. The next section shows the corresponding kernel code that is being called within this parallel loop.

3.4.5 Computational kernels

```
void opensblliblock00Kernel043(const double *p_B0, double *wk20_B0)
{
    wk20_B0[OPS_ACC1(0,0,0)] = inv_1*(-rc7*p_B0[OPS_ACC0(0,2,0)] -
    rc8*p_B0[OPS_ACC0(0,-1,0)] +
    (rc8)*p_B0[OPS_ACC0(0,1,0)] + (rc7)*p_B0[OPS_ACC0(0,-2,0)]);
}
```

EXAMPLE CODE 3.27: A user kernel to calculate a central derivative in OPS

By default all of the user kernels in OpenSBLI are written to a file named *opensblliblock00_kernels.h*. This header file is included at the start of the main *opensbli.cpp* code to import all of the user kernels. Full instructions for generating and compiling codes in OpenSBLI is given in A.3 in the appendix. In the user kernel above the pressure array *p_B0* is being indexed at four offset grid

locations to form the central derivative approximation. The result is written to the work array *wk20_B0*, to be used later in the simulation. For each of the arrays there is an OPS_ACCX[(0,0,0)] decorator macro, where X is an integer referring to the position of that array in the input function arguments. The numbering of the array access codes must be consistent with the ordering of the kernel arguments. The quantity in parenthesis is the relative grid indexing, with (0,0,0) referring to the current grid point the kernel is being evaluated on.

```

void opensbliblock00Kernel100(double *rhoE_B0, double *rho_B0, double *rhoU0_B0,
    double *rho_B0, double *rhoU2_B0,
const int *idx)
{
    # Create local doubles to store intermediate values
    # These are the output of the GridVariables
    double u0 = 0.0;
    double r = 0.0;
    double x0 = 0.0;
    double u2 = 0.0;
    double p = 0.0;
    double x2 = 0.0;
    double x1 = 0.0;
    double u1 = 0.0;
    # Calculate the local coordinate values at this specific (i,j,k) grid location
    x0 = Delta0block0*idx[0];
    x1 = Delta1block0*idx[1];
    x2 = Delta2block0*idx[2];
    # Set the primitive variables based on the coordinate values at this single grid
    point
    u0 = sin(x0)*cos(x1)*cos(x2);
    u1 = -sin(x1)*cos(x0)*cos(x2);
    u2 = 0.0;
    p = (0.0625*cos(2.0*x0) + 0.0625*cos(2.0*x1))*(cos(2.0*x2) + 2.0) + 1.0/(pow(Minf,
        2)*gama);
    r = pow(Minf, 2)*gama*p;
    # Use the primitive variables to evaluate the conservative variables
    # These are the DataSets from the code-generation
    rho_B0[OPS_ACC3(0,0,0)] = r;
    rhoU0_B0[OPS_ACC2(0,0,0)] = r*u0;
    rhoU1_B0[OPS_ACC0(0,0,0)] = r*u1;
    rhoU2_B0[OPS_ACC4(0,0,0)] = r*u2;
    rhoE_B0[OPS_ACC1(0,0,0)] = p/(gama - 1) + 0.5*r*(pow(u0, 2) + pow(u1, 2) + pow(u2,
        2));
}

```

EXAMPLE CODE 3.28: An example of local and global storage in an OPS user kernel.

A second example of an OPS user kernel is shown above. This kernel calculates the initial condition for the Taylor-Green Vortex (TGV) case described in section 5.2.1. While this kernel has no relative grid indexing, it demonstrates the differences between OpenSBLI *GridVariables* and *DataSets* introduced in section 3.3.4. Each of the *GridVariables* that were present in the

code-generation for this kernel have been converted by OpenSBLI into eight local doubles in C. The grid-index locator *ops_arg_idx()* from section 3.4.4 is the final input to this kernel. The current (i, j, k) index being evaluated is used to calculate the local coordinate by accessing *idx[0,1,2]*. These coordinate entries are then used to calculate local values of the primitive variables. The final five lines set the conservative arrays globally in the simulation based on the local primitive values.

3.4.6 Writing the simulation output to file

```
char name[80];
sprintf(name, "opensbli_output.h5");
ops_fetch_block_hdf5_file(opensbliblock00, name);
ops_fetch_dat_hdf5_file(rho_B0, name);
ops_fetch_dat_hdf5_file(rhou0_B0, name);
ops_fetch_dat_hdf5_file(rhou1_B0, name);
ops_fetch_dat_hdf5_file(rhou2_B0, name);
ops_fetch_dat_hdf5_file(rhoE_B0, name);
ops_fetch_dat_hdf5_file(x0_B0, name);
ops_fetch_dat_hdf5_file(x1_B0, name);
ops_fetch_dat_hdf5_file(x2_B0, name);
ops_exit();
```

EXAMPLE CODE 3.29: Creating a simulation output HDF5 file.

The bulk of the simulation code consists of calls to the parallel regions within the iteration and sub-stage loops of the time-stepping scheme. The final component of the code is to call the HDF5 library to generate output of the simulation data to file. This process has two steps: creating an HDF5 file for a specific simulation block and writing each of the named datasets to that file. The first step is to call the *ops_fetch_block_hdf5_file* function with the simulation block and an output file name. Once this file has been created each of the datasets are written to it using the *ops_fetch_dat_hdf5_file* command. The final part of the simulation code is a call to *ops_exit()* to safely terminate the simulation.

3.4.7 Performance and scaling tests

Detailed performance analysis of the OPS library is beyond the scope of the current thesis and has been reported externally in [Mudalige et al. \(2014, 2015, 2019\)](#), and [McIntosh-Smith et al. \(2019\)](#) among others. In [Mudalige et al. \(2014, 2015, 2019\)](#), the performance of OPS has been assessed on multiple architectures and comparisons have been made to hand-written code for each of the computational back-ends. The present work focuses instead on the generation of OPS code for the solution of physical problems, and the application of a range of numerical schemes. This is a core component of the ‘separation of concerns’ philosophy described in section 3.1, where the components of the simulation are separated from their parallel implementation. Nevertheless, it is informative to provide some examples of cross-architecture performance comparisons for the simulation cases discussed in this thesis.

Target architecture (processes/threads)	Time (s)	Speed-up
Intel Xeon E5-2697 24 core node (CPU, 24 MPI)	413.1	1.00
Intel Xeon Phi 7210 (64 MPI x 4 OMP, AVX512)	224.7	1.84
NVIDIA Tesla K20X (GPU, CUDA)	233.9	1.77
NVIDIA Tesla K40 (GPU, CUDA)	204.6	2.02
NVIDIA Tesla P100 (GPU, CUDA)	44.0	9.39

TABLE 3.1: Runtime comparison for 500 iterations of a 3D SBLI test case. The Ivy-Bridge 24 CPU cores (MPI) platform is taken as the baseline time.

A runtime comparison was performed for a 3D span-periodic version of the SBLI case described in Lusher et al. (2018b) and section 4.6. The simulations were performed on a grid consisting of $(300 \times 400 \times 25)$ points for 500 iterations of the simulation. Timers are added by OpenSBLI to the generated OPS code, which measure only the portion of time spent in the main iteration loop to avoid variability from file-systems used for input/output of data. All of the reported timings were averaged over multiple runs but varied only on the order of $10^{-1}s$. In addition to the performance numbers reported in this section, section 5.2.5 shows the relative CPU and GPU performance of the shock-capturing options for the compressible TGV case on 256^3 grids.

Table 3.1 shows the runtime on various architectures, with a speed-up factor relative to one node (24 CPU cores, MPI) on the UK’s national ARCHER HPC facility. MPI codes were compiled with the Intel compiler optimised with function in-lining and the -O3 flag, CUDA code was compiled using GCC/NVCC and the -O3 optimisation. The Intel Xeon Phi with 256 threads enabled achieved a speed-up of 1.84 relative to the CPU node, utilizing a hybrid MPI/OpenMP OPS generated code. Similar performance was found for the NVIDIA K20X and K40 GPUs, both proving to be fast alternatives to a conventional CPU node. Significant performance improvements were seen for the Pascal-based NVIDIA P100 architecture, achieving a 4.65x speed-up relative to a previous generation NVIDIA K40 GPU.

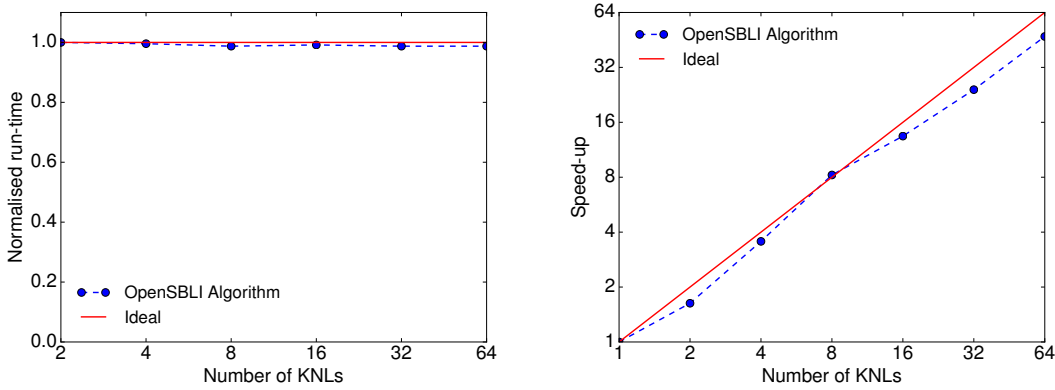


FIGURE 3.7: (left) Weak and (right) strong scaling for OpenSBLI on Intel Phi Knight’s landing accelerator cards, up to 64 KNLS.

Multi-node performance of the generated OPS code was assessed during an initial test of the University of Cambridge Service for Data Driven Discovery (CSD3) HPC facility (<http://csd3.cam.ac.uk>). Runtime was measured for 500 iterations of the 3D Taylor-Green vortex problem described in chapter 5. Figure 3.7 displays the weak and strong scaling results for the Intel Xeon

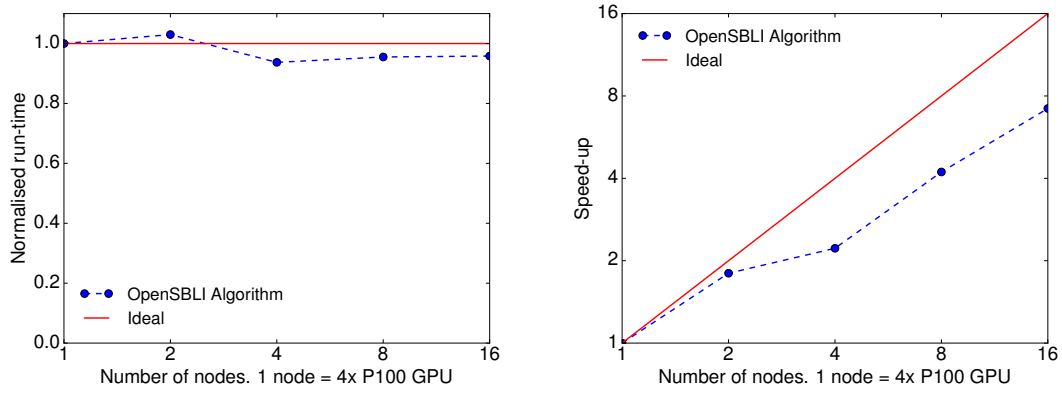


FIGURE 3.8: (left) Weak and (right) strong scaling for OpenSBLI on NVIDIA Pascal P100 GPU cards, up to 64 GPUs.

Phi Knight's landing (KNL) accelerator cards. In both cases the MPI back-end was selected from OPS, with the Intel 17.04 compiler and Intel MPI used. Excellent weak scaling was achieved up to the total of 64 KNLs that were accessible, for meshes ranging from $1.6 \times 10^6 - 1.1 \times 10^9$ grid points. Similarly, a good linear speed-up was observed for strong scaling on up to 64 KNLs.

GPU scaling was assessed using the CUDA+MPI OPS back-end, with the Intel 17.04 compiler, CUDA 8.0, and Intel MPI for communication between GPUs. Each GPU node of the CSD3 facility consists of 4 NVIDIA Pascal P100 GPU cards running on a 24 core Intel Broadwell CPU. Good weak scaling was obtained up to 64 P100 GPU cards, as seen in Figure 3.8 (left). Strong scaling is considerably more difficult to achieve on multi-GPU configurations, with the MPI communication overhead and limited transfer rates of the PCI-E bus. Nevertheless, reasonable strong scaling of around 50% was seen up to 64 GPUs (Figure 3.8 (right)), restricted by the lower memory capacities of the GPUs compared to KNL (16GB compared to 192GB). Older GPUs with reduced compute capability per card would fare better in strong scaling tests. The scaling data resulted in the award of compute time for two twelve month allocations on the CSD3 GPU machine, which formed the bulk of the resources used for the simulations in this thesis. Additional compute resources were also provided by the IRIDIS5 high performance computing facility at the University of Southampton.

This chapter has described the major code development performed during the PhD project. Detailed guidelines of how to use both the OpenSBLI and OPS systems have been presented. An example user-defined problem script has been broken down into each of its constituent components. Particular detail has been given to the specifics of the shock-capturing routines and the code-writer class. Examples have been given of how the code-writer class generates calls to functions in the OPS library. The resulting OPS C code was broken down into small snippets to highlight the important functionality contained in this library. Differences between the various data structures in OpenSBLI have been discussed, with examples provided of their representation in the OPS C code. Having described the code in detail, the next chapter demonstrates the correctness of the implementation. A selection of varied test cases are performed to verify and validate the features described in this chapter.

Chapter 4

OpenSBLI validation cases

4.1 Introduction

In this chapter a selection of validation cases are presented to demonstrate the correct implementation and accuracy of the OpenSBLI code. While it is easy to set up a wide range of test cases in OpenSBLI, each of the validation cases in this chapter are selected to demonstrate specific aspects of the code implementation. The validation cases are arranged as follows.

In section 4.2 the well known Shu-Osher ([Shu and Osher, 1988](#)) shock/density-wave interaction is performed with various shock-capturing options. This test case highlights the improved resolution of higher order schemes, and demonstrates the ability of the code to capture both a 1D shock and the high frequency density waves. Section 4.3 verifies the implementation of the shock-capturing schemes and characteristic decomposition for the inviscid Euler equations in two dimensions. Simulations are compared against an analytic solution for a travelling density wave. The order of convergence of the WENO schemes is confirmed for this smooth test case by varying the grid resolution and monitoring the L_1 error norm. Verification of both adiabatic and isothermal no-slip boundary conditions is shown in section 4.4 for a 2D laminar channel flow. The mixed-wall condition subsonic channel is verified against an analytical solution.

A 2D viscous shock-tube case is presented in section 4.5, with flow conditions taken from [Daru and Tenaud \(2009\)](#). This problem requires the code to simulate the reflection of a normal shock from a solid adiabatic wall, and reproduce the resulting flow separation and complex unsteady shock structures. Section 4.6 demonstrates a Mach 2 2D laminar oblique shock-wave/boundary-layer interaction using the flow conditions from [Katzner \(1989\)](#), based on the earlier experiments of [Hakkinen et al. \(1959\)](#). This test case provides validation of the shock-capturing schemes and boundary-layer initialisation for a challenging viscous flow involving flow separation. The results are validated against the published solution of [Sansica et al. \(2013\)](#) and [Sansica \(2015\)](#). The results for this SBLI case and the OpenSBLI WENO implementation were published during the PhD project in [Lusher et al. \(2018b\)](#). The physical results to be presented in chapters 6 and 7 are a three-dimensional extension of this test case with the inclusion of sidewall effects.

The ability of OpenSBLI to simulate turbulent flows is highlighted in section 4.7, for a low-speed 3D turbulent channel flow based on the original work of [Kim et al. \(1987\)](#). The simulations are

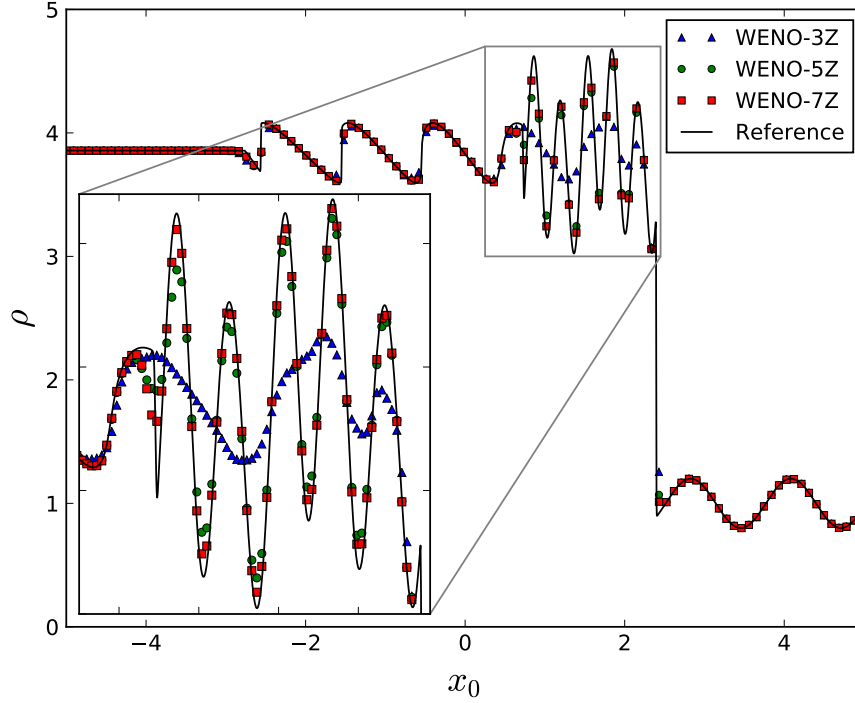


FIGURE 4.1: Shu-Osher density distribution at a non-dimensional simulation time of $t = 1.8$, for WENO orders: 3Z (blue), 5Z (green) and 7Z (red). Reference fine mesh solution (continuous, black). Every third data point is displayed, with consecutive data shown in the 2x zoom inset.

validated against the more recent work of [Vreman and Kuerten \(2014\)](#). Finally, a supersonic turbulent channel flow is performed at Mach 1.5 using the flow conditions of [Coleman et al. \(1995\)](#). The compressible problem is simulated with both the non-dissipative central scheme and various shock-capturing options. The two turbulent channel flow cases formed part of a collaborative work with Dr Arash Hamzehloo of Imperial College London, resulting in a joint paper. All of the simulations in this chapter were performed using the low-storage 3rd order explicit Runge-Kutta time-stepping scheme.

A further example of OpenSBLI validation was shown by [Lefieux et al. \(2019\)](#), who simulated roughness-induced laminar-turbulent transition at Mach 6. Disturbances were added to the laminar portion of the flow by means of a time-dependent acoustic source above the inlet boundary-layer. These disturbances were subsequently amplified as they convected downstream and led to a transition to turbulence after passing over a three-dimensional bump. The simulations were validated against both the existing SBLI code from the University of Southampton and the elsA code from the French government research institute ONERA.

4.2 1D Shu-Osher shock/density-wave interaction

The Shu-Osher problem is a one-dimensional inviscid test case involving the interaction of a Mach 3 shock with a smooth density wave. It acts as a useful validation for both the characteristic decomposition and shock-capturing schemes for the Euler equations in one dimension. Three

different orders of the WENO-Z scheme are used in this section to highlight the benefits of higher-order shock-capturing schemes. The simulation is initialised with the discontinuous conditions given in [Shu and Osher \(1988\)](#); [Taylor et al. \(2007\)](#), such that

$$(\rho, u_0, p) = \begin{cases} (3.857143, 2.629369, 10.33333) & \text{if } x_0 < -4 \\ (1 + 0.2 \sin(5x_0), 1.0, 0.0) & \text{if } x_0 \geq -4 \end{cases}$$

with Dirichlet conditions enforced at the domain boundaries $x_0 = [-5, 5]$. The simulation is advanced to a non-dimensional time of $t = 1.8$ with a time-step of $\Delta t = 2 \times 10^{-4}$. A reference solution was computed with a WENO-7Z scheme and a fine mesh of $N = 3200$ grid points. Figure 4.1 shows a comparison of results for the WENO-3Z, WENO-5Z, and WENO-7Z schemes on an $N = 320$ grid.

The normal-shock propagates in the positive x direction from its start location at $x = -4$, interacting with the smooth imposed initial density perturbation. A series of shocklets are formed from the interaction in the region of $-2 < x < 1$, with a series of high-frequency waves present behind the normal-shock at $0 < x < 2$. The qualitative features of the solution are consistent with previous work such as figure 4 of [Taylor et al. \(2007\)](#). While all three of the WENO-Z schemes match the reference solution for the shocklets and smooth regions of the flow, the lowest order WENO-3Z scheme struggles to resolve the high-frequency waves behind the main shock. This can be seen in the zoomed inset of figure 4.1, highlighting the benefits of higher-order schemes on coarse meshes.

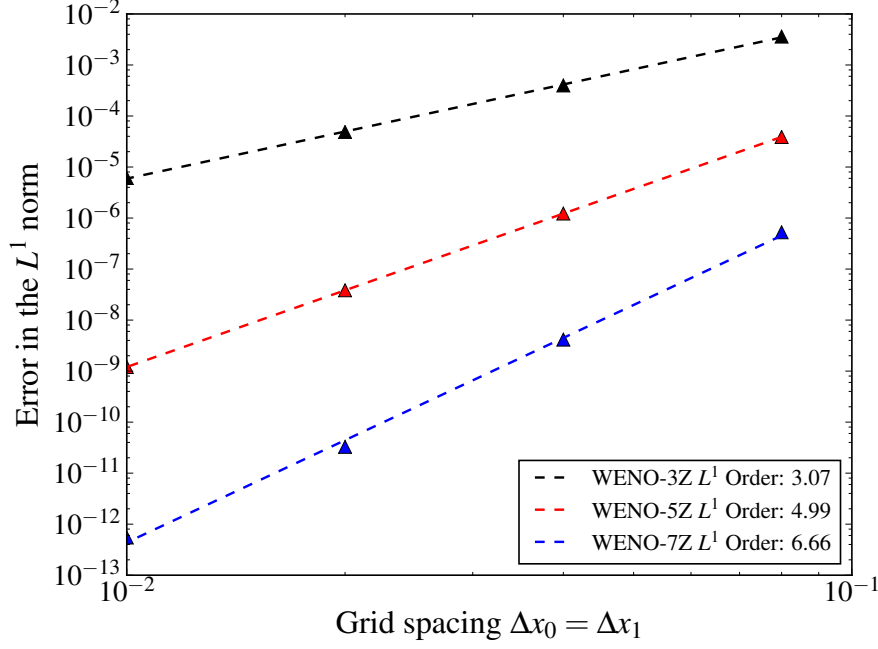
4.3 2D Euler wave propagation

WENO and TENO schemes are adaptive in the sense that the number of candidate stencils (figure 2.1) used for the flux reconstruction will vary depending on the local flow conditions. The schemes will reduce to lower order approximations around shocks and maintain the full numerical stencil in smooth regions of the flow. To be able to test the formal order of convergence of the schemes, it is therefore necessary to apply them to a smooth flow. The selected case is a smooth travelling 2D density perturbation for the Euler equations. The case acts as validation of the characteristic decomposition in two dimensions and demonstrates the order of convergence for the underlying shock-capturing schemes.

The initial perturbation takes the form

$$\rho(x_0, x_1, t) = 1 + 0.2 \sin(\pi(x_0 + x_1 - t(u_0 + u_1))), \quad (4.1)$$

with constant velocity components $u_0 = 1.0$, $u_1 = -0.5$ and pressure $p = 1.0$. Periodic boundaries are applied on all sides of the domain, with a time-step of $dt = 1 \times 10^{-4}$. To verify the implementation of the 2D Euler equations, a comparison is made to the analytical solution. Table 4.1 shows the L_1 error, relative to the analytic solution (4.1) after a non-dimensional time of $t = 2.5$, given by

FIGURE 4.2: Convergence of the L^1 norm for three orders of WENO-Z scheme.

$$L^1 = \frac{\sum_{i,j} |\rho_{\text{exact}} - \rho_{i,j}|}{N_{x_0} N_{x_1}}. \quad (4.2)$$

Figure 4.2 shows the rate of convergence for WENO-Z schemes of order 3, 5, and 7 on four different grid resolutions. The WENO-3Z and WENO-5Z have consistent orders of accuracy of 3 and 5 as expected, with 7th order accuracy obtained for WENO-7Z over the three coarsest grid resolutions. At the finest grid the WENO-7Z scheme is limited by the scale of the problem and the order of the time-stepping scheme. At the finest mesh resolution there are diminishing returns with this time-step and the absolute error is approaching machine precision. Table 4.1 shows the relative convergence order with each grid refinement, demonstrating that the scheme is performing at 7th order but is limited by the scale of this small problem in absolute error at the finest resolution.

	WENO-3Z		WENO-5Z		WENO-7Z	
Grid	L^1_{Error}	Order	L^1_{Error}	Order	L^1_{Error}	Order
25×25	2.05e-3	-	2.47e-5	-	3.33e-7	-
50×50	2.50e-4	3.04	7.81e-7	4.98	2.64e-9	6.98
100×100	3.10e-5	3.01	2.45e-8	5.00	2.08e-11	6.99
200×200	3.88e-6	3.00	7.65e-10	5.00	2.39e-13	6.44

TABLE 4.1: Error in the L^∞ norm for 3rd, 5th and 7th WENO-Z schemes, and convergence rates for smooth flows.

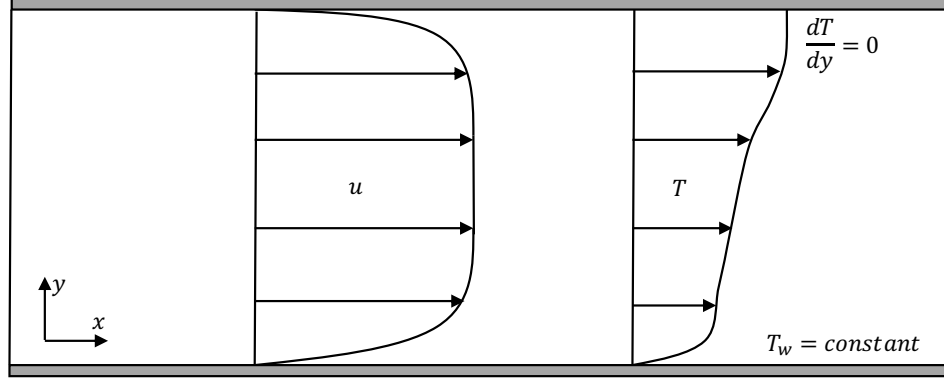


FIGURE 4.3: Schematic of the 2D laminar mixed-wall isothermal-adiabatic channel flow. The configuration has a constant isothermal wall temperature at $y = -1$ and a zero heat-flux wall at $y = 1$. The expected velocity and temperature profiles over the channel are shown.

4.4 2D Laminar isothermal-adiabatic channel flow

To verify the implementation of the isothermal and adiabatic no-slip wall boundary conditions from section 3.3.6, a 2D laminar channel flow simulation is compared to an analytic solution. As shown in figure 4.3, the channel is comprised of a constant temperature wall at $y = -1$ and a zero heat flux wall at $y = 1$. For constant viscosity, the streamwise velocity and temperature profiles for the analytic solution are given by

$$u = \frac{Re(1 - y^2)}{2}, \quad (4.3)$$

$$T = 1 - \frac{Re^2 M_\infty^2 Pr(\gamma - 1)(y^4 - 4y - 5)}{12}. \quad (4.4)$$

Simulation parameters are taken to be $Re = 90$, $Pr = 0.72$, $M_\infty = 0.01$, and $(N_x, N_y) = (32, 64)$. With these physical values, the analytic solution gives a predicted adiabatic wall temperature at $y = 1$ of $T_{aw} = 1.155520$ (7 s.f.). The simulation is advanced until a non-dimensional time of $T = 1000$ with a time-step of $\Delta t = 1 \times 10^{-4}$. The domain has dimensions of size $L_x = 2\pi$ and $L_y = 2$, with a uniform grid distribution of $(N_x, N_y) = (32, 64)$.

Figure 4.4 shows the results for both a 4th order central scheme and a 6th order TENO shock-capturing scheme. Excellent agreement is observed to the analytic result for both the streamwise velocity and temperature profiles. The correct near-wall behaviour is produced for both of the schemes. At this grid resolution, the central and TENO adiabatic wall temperatures are 1.155510 and 1.155508 respectively. This corresponds to relative percentage errors for T_{aw} of $0.8 \times 10^{-3} \%$ and $1.0 \times 10^{-3} \%$ for the central and TENO schemes respectively.

4.5 2D Viscous normal-shock tube

The viscous shock-tube is a demanding case for shock capturing schemes and tests the ability of the code to simulate a 2D normal shock. The problem combines complex shock structures,

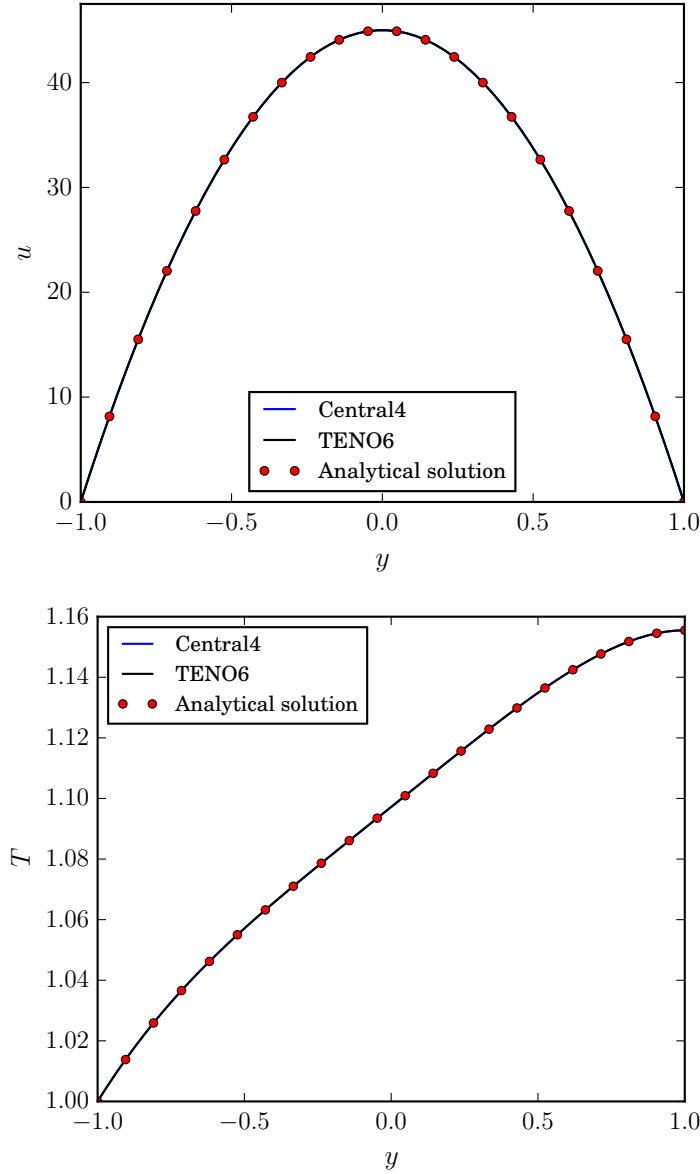


FIGURE 4.4: Comparison of the 2D mixed-wall condition laminar channel flow to the analytical solution in equation (4.3). Showing profiles for the (top) streamwise velocity and (bottom) temperature.

shock reflection from a solid wall, and regions of unsteady boundary-layer separation. An in-depth study of the inviscid and viscous 2D shock tubes was given by [Daru and Tenaud \(2009\)](#), from which the flow conditions in this section are taken. The computational domain $(x_0, x_1) = ([0, 1], [0, 0.5])$ is partitioned by an initial diaphragm located at $x_0 = 0.5$ as shown in figure 4.6. A discontinuous initial profile is imposed which generates a normal shock that propagates in the positive x direction. The initial states to the left and right of this discontinuity are given by

$$(\rho, u_0, u_1, p) = \begin{cases} (120, 0, 0, 120/\gamma) & \text{if } x_0 < 0.5 \\ (1.2, 0, 0, 1.2/\gamma) & \text{if } x_0 \geq 0.5, \end{cases}$$

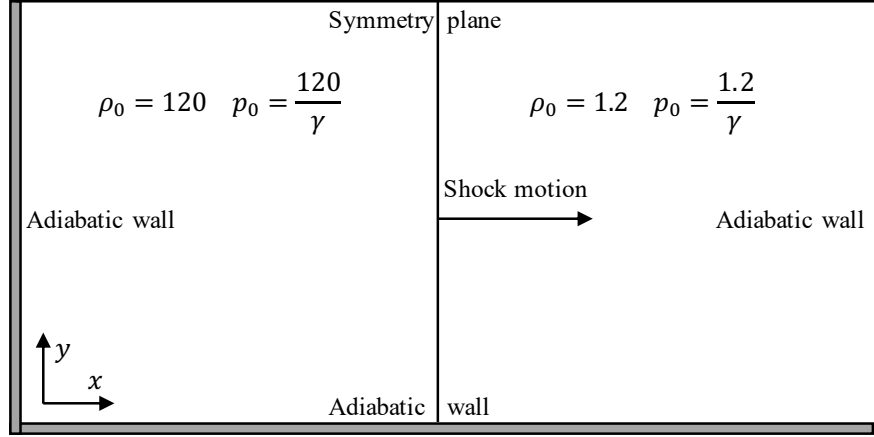


FIGURE 4.5: Schematic of the viscous shock-tube. The configuration has three adiabatic walls and a symmetry condition. At $t = 0$ the domain is initialised with a diaphragm at $x = 0.5$. The discontinuous density and pressure values are shown on each side. A normal shock propagates in the direction shown and reflects from the opposing wall.

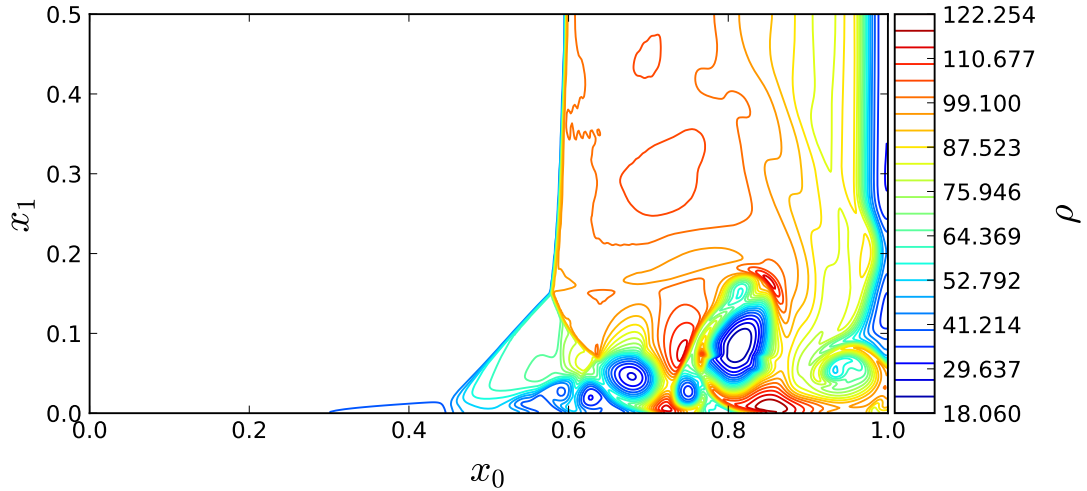


FIGURE 4.6: Density contours for the 2D viscous shock-tube at Reynolds number of $Re = 200$. WENO-7Z scheme with $[x_0, x_1] = [600, 300]$ points.

with a reference Mach number of $M = 1$, Reynolds number $Re = 200$, and Prandtl number $Pr = 0.73$. A symmetry condition is enforced on the upper boundary of the domain, with adiabatic no-slip wall conditions set on all other boundaries and constant viscosity assumed. The simulation is advanced with time-step of $dt = 1 \times 10^{-5}$ until a non-dimensional time of $t = 1$. The grid consists of a uniform distribution of $[x_0, x_1] = [600, 300]$ points.

Figure 4.6 shows the instantaneous density contours at $t = 1$. The shock has propagated to the $x = 1$ adiabatic wall and reflected back to a position of $x = 0.6$. The relative motion of the shock generates a thin boundary-layer which separates after the shock reflects from the side wall. Above the separation bubble a lambda shock structure can be seen to form. Good qualitative agreement is found compared to the results of Daru and Tenaud (2009). The location of the triple point shock pattern at $(x_0, x_1) \approx (0.58, 0.13)$ is in good agreement with Daru and Tenaud (2009).

4.6 2D Laminar oblique shock-wave/boundary-layer interaction

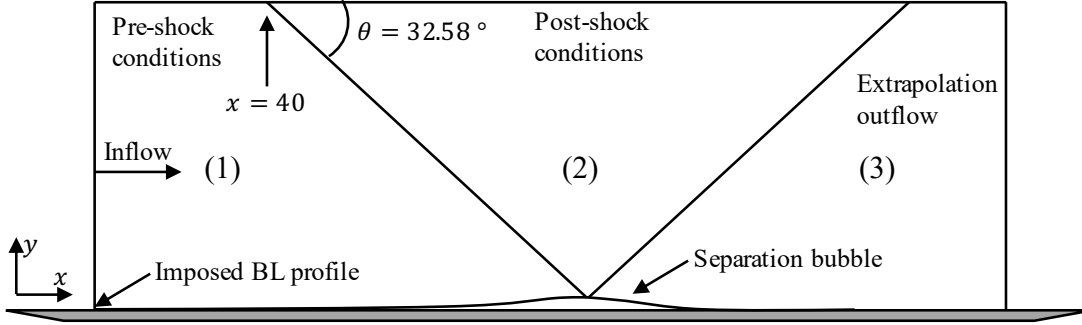


FIGURE 4.7: Schematic of the Mach 2 laminar oblique SBLI validation case. The rectangle represents the computational domain. An initial oblique shock-wave is imposed on the upper boundary of the domain at $x = 40$.

Quantity	Symbol	Value
Free-stream Mach number	M_∞	2.0
Reynolds number based on shock impingement	Re_x	3.0×10^5
Reynolds number based on inlet displacement thickness	$Re_{\delta_0^*}$	950
Prandtl number	Pr	0.71
Sutherland temperature	T_s	110.4 K
Reference freestream temperature	T_∞	288.0 K
Non-dimensional wall temperature	T_w	1.676

TABLE 4.2: Simulation parameters for the two-dimensional laminar SBLI.

The validation case in this section is the one most relevant to the physical results in the later chapters of this thesis. It demonstrates that the code can simulate an oblique shock-reflection with an appropriate inflow boundary-layer profile. Both the SBLI and the initial boundary-layer development needs to be correct in order to match the reference data. The results contained in this section were published in [Lusher et al. \(2018b\)](#), along with details of the WENO-5Z implementation used to perform the simulations. Flow conditions are taken from the laminar SBLI of [Katzner \(1989\)](#), with comparison to the more recent results of [Sansica et al. \(2013\)](#). Katzner performed the study of laminar SBLI with free-stream Mach numbers ranging from 1.4 to 3.4 over a range of impingement Reynolds numbers. The numerical study of Katzner was based on the earlier laminar SBLI experiments of [Hakkinen et al. \(1959\)](#). For this validation case a Mach 2 freestream is selected with a Reynolds number of $Re_x = 3.0 \times 10^5$ at the shock impingement location.

The computational domain is shown in figure 4.7. The domain is of size $[x, y] = [400, 115]$, to ensure that reflections on the upper boundary from the initial reflected shock fall outside of the computational domain. The similarity solution from section 3.3.7 is used to impose a boundary-layer profile with Reynolds number based on inlet displacement thickness of $Re_{\delta_0^*} = 950$. The freestream at the inlet is set to Mach 2, with a Prandtl number of $Pr = 0.71$ as in table 4.2. A no-slip isothermal wall is imposed at the bottom of the domain. The isothermal wall is held

at a constant non-dimensional temperature of $T_w = 1.676$ (4 s.f.), equal to the adiabatic wall temperature obtained from the similarity solution described in section 3.3.7.

A piecewise Dirichlet condition is applied on the upper boundary to impose the Rankine-Hugoniot shock-jump conditions at $x = 40$. A shock corresponding to a shock angle of 32.58° is applied. The pre and post shock-jump conditions applied on the upper boundary are given to 5 d.p. in table 4.3. Referring to the zones in figure 4.7, an outlet pressure ratio of $p_3/p_1 = 1.4$ is expected for these flow conditions. Pressure extrapolation and zero-th order extrapolation are applied at the inlet and outlet respectively. Dynamic viscosity is computed by Sutherland's law (2.6) with Sutherland and reference temperatures of $110.4K$ and $288.0K$ respectively.

Conservative flow variable	Pre-shock (5 d.p.)	Post-shock (5 d.p.)
ρ	1.00000	1.12973
ρu	1.00000	1.09211
ρv	0.00565	-0.05886
ρE	0.94644	1.05908

TABLE 4.3: Incident shock-jump conditions for the Mach 2 two-dimensional laminar SBLI.

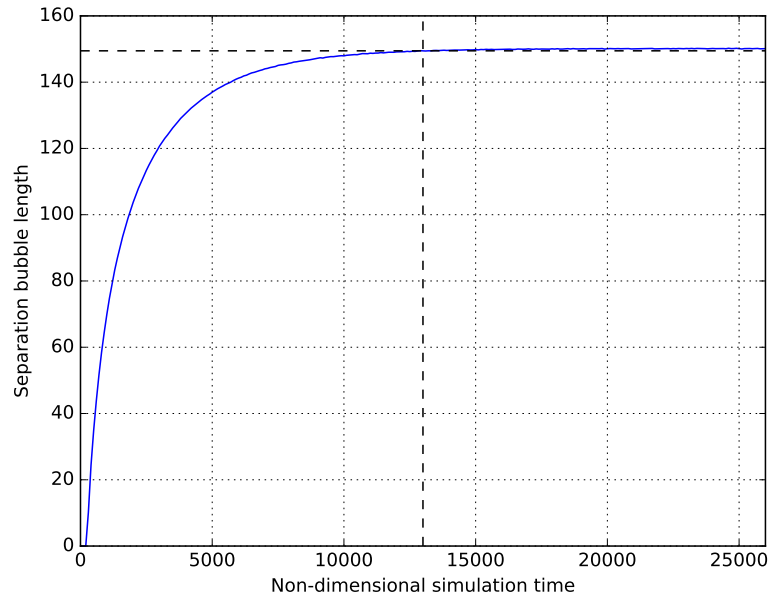


FIGURE 4.8: Time evolution of the two-dimensional SBLI separation bubble length to measure convergence.

Based on preliminary studies, a grid resolution of $(N_x, N_y) = (609, 255)$ was selected. Grid stretching is applied in the wall normal y direction with points clustered hyperbolically towards the wall as

$$y = L_y \frac{\sinh(s_1 \xi)}{\sinh(s_1)}, \quad (4.5)$$

for uniformly distributed points $\xi = [0, 1]$ and a stretch factor of $s_1 = 5$. Laminar SBLI are known to take a long time to fully converge (Sansica et al., 2013), with earlier studies often

reporting shorter separation bubble lengths where convergence had not been met. For this study convergence of the SBLI is assessed by monitoring the streamwise length of the separation bubble as it evolves in time.

Figure 4.8 shows this development, the separation bubble initially grows rapidly before converging to a constant length. For this validation case the non-dimensional time for convergence is taken to be $t = 13000$, represented by the vertical dashed line in figure 4.8. This simulation time corresponds to 32.5 freestream flow-through times of the domain. The simulation was then restarted for a further 32.5 flow-through times up until $t = 26000$. After doubling the length of the time integration, the relative error in the separation bubble length was 0.4% compared to the $t = 13000$ solution.

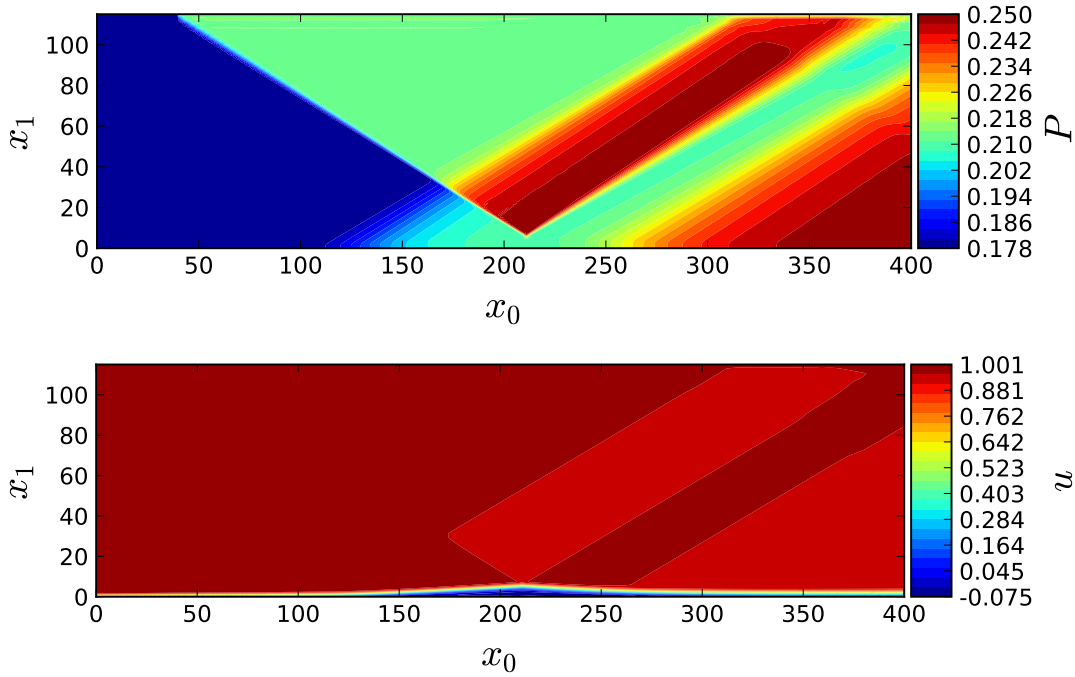


FIGURE 4.9: Instantaneous pressure (top) and streamwise velocity (bottom) contours for the converged 2D laminar SBLI.

Figure 4.9 (top) shows pressure contours for the converged SBLI. Comparing to the SBLI schematic in figure 1.2, a well-resolved oblique shock-wave impinges on the boundary-layer, causing a thickening of the profile and the development of a large separation bubble observed in Figure 4.9 (bottom). A series of compression waves can be seen originating from the front of the separation bubble at $x = 110$, which coalesce and form the reflected shock. At $x = 205$ there is an expansion fan as the flow turns over the apex of the separation bubble. At $x = 260$ there is a reattachment shock at the back of the separation bubble. The maximum value of streamwise flow-reversal is $u = -7.5 \times 10^{-2}$.

Figure 4.10 (right) shows the skin-friction distribution in x along the wall, with the negative region corresponding to the separated flow. The expected asymmetric shape of the laminar separation bubble is observed (Katzner, 1989; Degrez et al., 1987), with excellent agreement to the reference solution of Sansica et al. (2013). Figure 4.10 (left) provides a similar picture, highlighting the two pressure rises from the shock interaction and the correct outlet pressure ratio

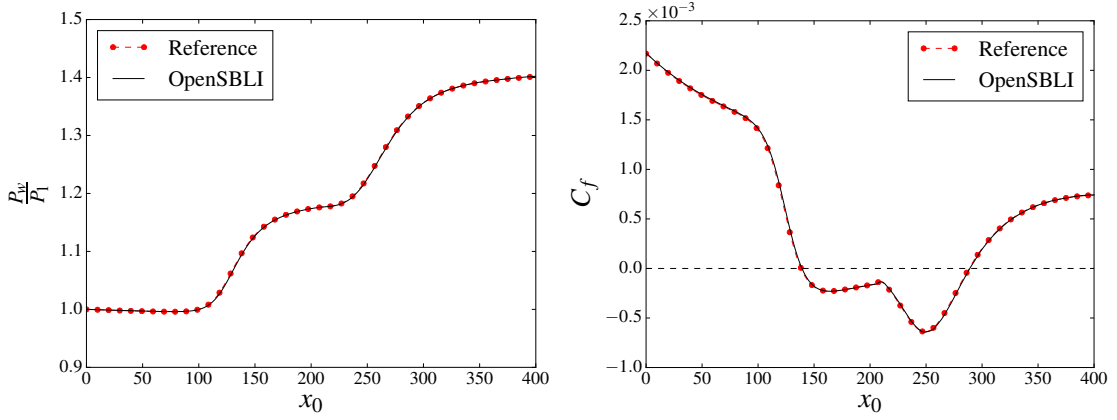


FIGURE 4.10: Comparison of wall pressure (left) and wall normal skin-friction (right) distributions compared to the reference solution published in [Sansica et al. \(2013\)](#)

of 1.4 obtained downstream of the interaction. The two-dimensional laminar SBLI is extended to three-dimensions with sidewall effects in the enclosed duct simulations in Chapter 6.

4.7 3D Incompressible turbulent channel flow

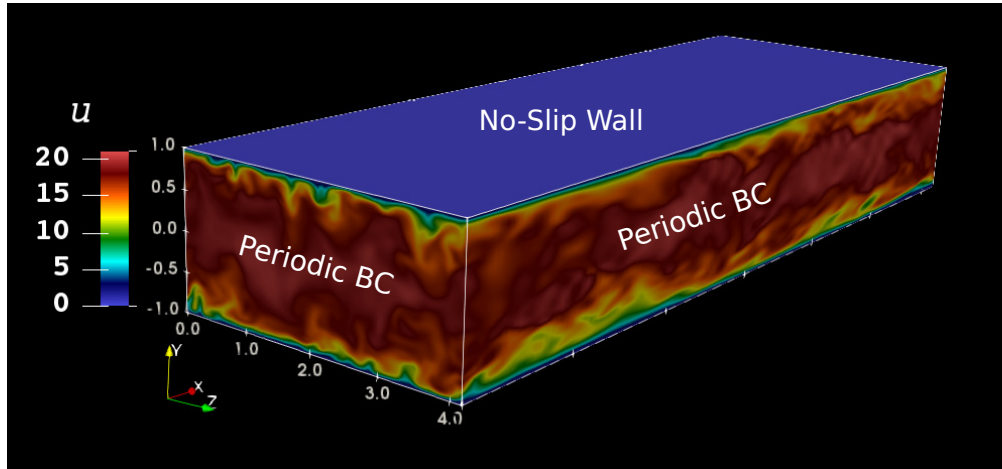


FIGURE 4.11: Computational domain for the 3D turbulent channel flow cases. Periodic boundary conditions are applied in both the streamwise and spanwise directions. The domain is bounded by two isothermal no-slip walls located at $y = -1$ and $y = 1$.

In this section a DNS of a turbulent channel flow is conducted at Mach 0.01, with a Reynolds number based on friction velocity of $Re_\tau = 180$. This was performed as a joint work with Dr. Arash Hamzehloo of Imperial College London. The case is based on the original work of [Kim et al. \(1987\)](#), with comparison to the more recent work of [Vreman and Kuerten \(2014\)](#). As this is a low-Mach number test case, the results are obtained via the 4th order central differencing scheme. The purpose of the test case is to demonstrate good agreement of statistical quantities and acts as a further validation of the 4th order boundary closure detailed in section 2.6.

As shown in figure 4.11, the domain consists of a rectangular channel of size $(4\pi H \times 2H \times \frac{4}{3}\pi H)$, where $H = 1$ is the half-channel height. A hyperbolic tangent stretching function is applied symmetrically in the wall normal direction with length $L_y = 2$ such that

$$y = 0.5L_y \left(1.0 - \left[\frac{\tanh \left(s \left(1 - 2 \left(\frac{j}{N_y - 1} \right) \right) \right)}{\tanh(s)} \right] \right) - 1, \quad (4.6)$$

resulting in a y coordinate in the range $y = [-1, 1]$. The quantities $s = 1.7$, j and N_y refer to the stretch factor, current grid index and number of points in that direction. The channel is periodic in the streamwise (x) and spanwise (z) directions, with two isothermal no-slip walls located at $y = -1, 1$. A skew-symmetric form of the governing equations is used to improve numerical stability, similar to that described in [Jacobs et al. \(2017\)](#). A constant pressure-gradient term is added to the momentum and energy equations in 2.1 to drive the channel flow in the streamwise direction. The terms added to the momentum ($i = 0, 1, 2$) and energy equations are $\delta_{i,j}c_j$ and $u_j c_j$ respectively, for the Kronecker delta $\delta_{i,j}$ and constant $c_j = (c_0, c_1, c_2) = (-1, 0, 0)$.

The domain is initialised by combining a streamwise laminar velocity profile with a set of sinusoidal perturbations. The initial streamwise profile is given by

$$\bar{u} = \begin{cases} (1 - |y|)Re_\tau, & \text{if } (1 - |y|)Re_\tau < 10 \\ \kappa \ln((1 - |y|)Re_\tau) + b, & \text{otherwise} \end{cases} \quad (4.7)$$

where $b = 5.5$ and the von-Karman constant is $\kappa = 2.5$. The three velocity components are initialised as

$$u = \bar{u} + A c_x s_y s_z \left(\frac{L_x}{2} \right), \quad (4.8)$$

$$v = -A s_x c_y s_z, \quad (4.9)$$

$$w = -A s_x s_y c_z \left(\frac{L_z}{2} \right), \quad (4.10)$$

where the perturbations are defined as

$$(s_x, s_y, s_z) = \left(\sin \left(\frac{4\pi x}{L_x} \right), \sin(\pi y), \sin \left(\frac{2\pi z}{L_z} \right) \right), \quad (4.11)$$

$$(c_x, c_y, c_z) = \left(\cos \left(\frac{4\pi x}{L_x} \right), 1 + \cos(\pi y), \cos \left(\frac{2\pi z}{L_z} \right) \right), \quad (4.12)$$

and the amplitude A is given by

$$A = 0.1 \times (\kappa \ln(Re_\tau) + b). \quad (4.13)$$

The initial density and temperature are set to $\rho, T = 1$, with pressure obtained from the ideal gas law as $p = (\gamma M_\infty^2)^{-1}$. A non-dimensional time-step of $\Delta t = 5 \times 10^{-5}$ is used with a spatial grid resolution of $(N_x, N_y, N_z) = (129, 129, 129)$. The simulations were run for an initial time of $T = 50$ to clear the transient, before being advanced in time for another $T = 1000$ to collect turbulent statistics.

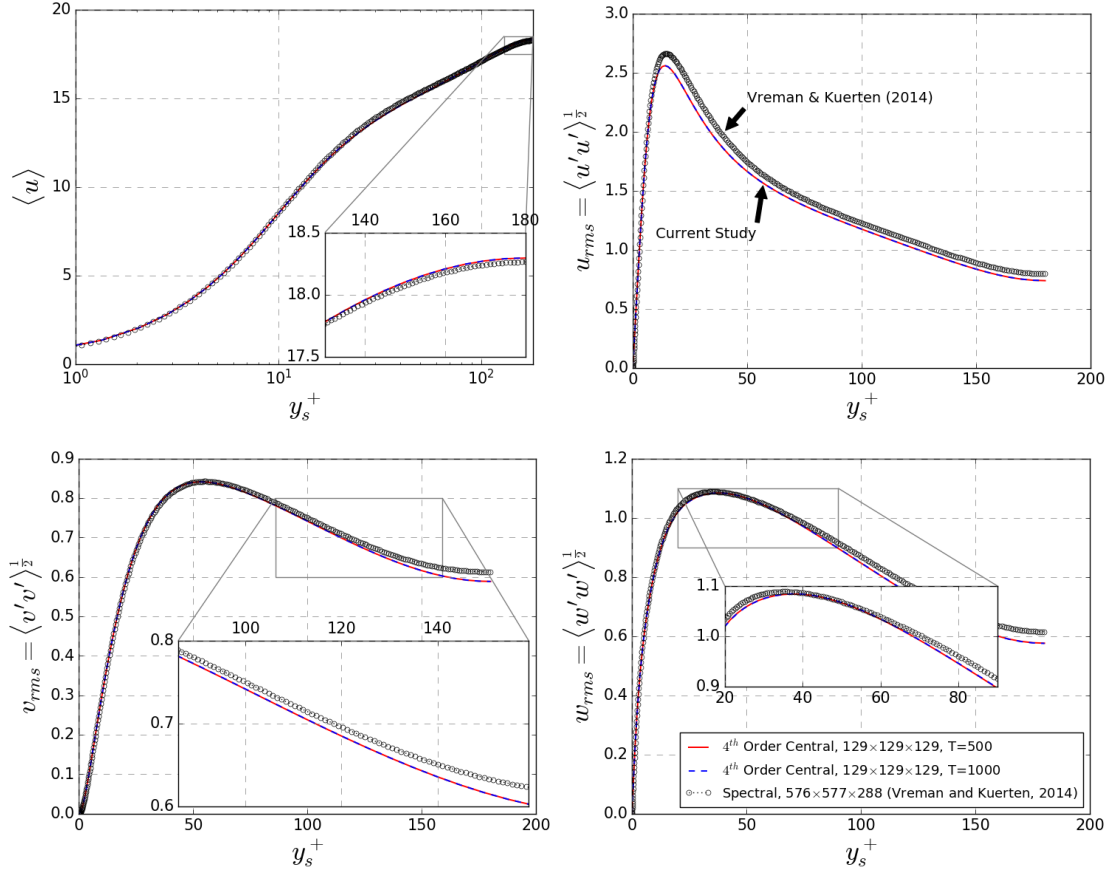


FIGURE 4.12: Turbulent statistics for the low-Mach channel flow compared to the spectral result of Vreman and Kuerten (2014).

Figure 4.12 shows the results of the simulation compared to the spectral code of Vreman and Kuerten (2014). The plots show the mean streamwise velocity profile $\langle \bar{u} \rangle$, and the root-mean-square (RMS) of the velocity fluctuations in wall units y^+ . The angled brackets here denote an average taken over both time and the homogeneous spatial directions x and z . Good agreement is found to be reference solution, the mean centreline velocity $\langle \bar{u}_c \rangle = 18.295$ is within 0.18% of the reference data. The peak values for the RMS of the velocity fluctuations are $u_{rms} = 2.558$, $v_{rms} = 0.841$ and $w_{rms} = 1.084$. These correspond to relative errors of 3.94%, 0.11% and 0.36% compared to the reference spectral data. The largest discrepancy to the reference is found in the streamwise velocity fluctuations. A grid study for the compressible case in the next section demonstrates that the turbulent channel flow is most sensitive to grid refinement in the streamwise direction.

4.8 3D Compressible supersonic turbulent channel flow

4.8.1 Sensitivity to grid refinement

The second turbulent channel flow case is selected to test the ability of the code to simulate high-speed compressible turbulence. This was performed as a joint work with Dr. Arash Hamzehloo of

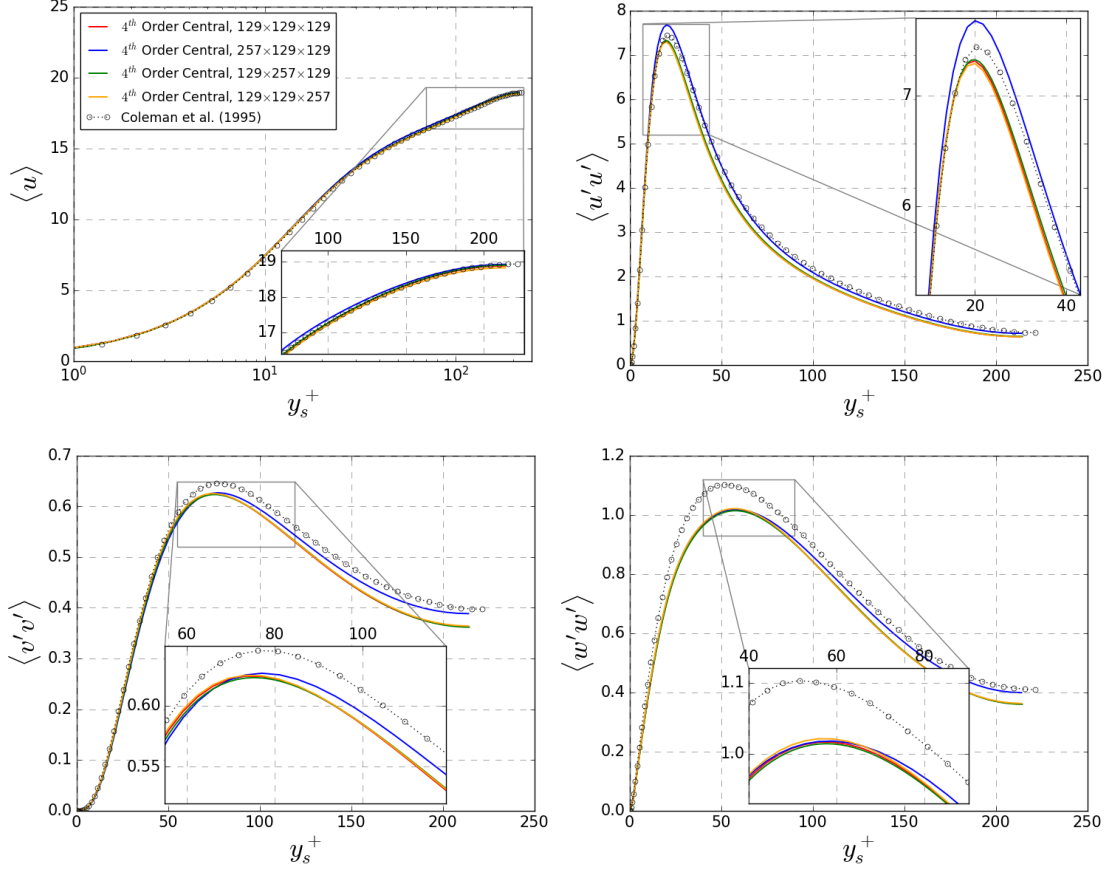


FIGURE 4.13: Validation of the OpenSBLI code for the supersonic Mach 1.5 turbulent channel, based on the flow conditions of Coleman et al. (1995).

Imperial College London. The flow conditions are taken from the work of Coleman et al. (1995), who investigated isothermal-wall channel flows for centreline Mach numbers of 1.5 and 3. The Mach 1.5 case is selected for the validations performed in this section. While this case does not contain shock-waves, it tests the ability of the shock-capturing schemes to resolve compressible turbulence against published reference data. The domain size, initialisation, and forcing terms are the same as in the low-speed case from the previous section. The selected case has a Reynolds number based on friction velocity of $Re_\tau = 222$, with a friction Mach number of $M_\tau = 0.082$.

Since the bulk-averaged density is used to non-dimensionalise the equations, a factor of $\frac{\rho_{wall}}{\rho_{bulk}} = 1.355$ is used in the simulation to adjust the values of Re_τ, M_τ to be consistent with the reference results. The values specified in the simulation are $\widehat{Re}_\tau = 190.71$ and $\widehat{M}_\tau = 0.0955$. The dynamic viscosity is calculated using the power law

$$\mu(T) = T^{0.7}. \quad (4.14)$$

For this compressible case there are less stringent demands placed on the size of the time-step required for stability. The time-step for the baseline case of $(N_x, N_y, N_z) = (129, 129, 129)$ is taken to be $\Delta t = 2 \times 10^{-4}$, a factor of four larger than in the previous section. Figure 4.13 shows the mean streamwise velocity profile and RMS velocity fluctuations for the 4th order central-difference scheme compared to the solution of Coleman et al. (1995). To demonstrate sensitivity

to grid resolution, three additional simulations were performed with twice the number of grid points in each direction independently.

Figure 4.13 shows the mean streamwise velocity profile and RMS velocity fluctuations for the 4th order central scheme on different grid resolutions. Good agreement is found for the mean streamwise velocity profile compared to [Coleman et al. \(1995\)](#). The zoomed inset shows that the turbulent channel flow is most sensitive to grid refinement in the streamwise direction. A similar picture is observed for all of the components of the RMS velocity fluctuations values, with the refined streamwise result most closely matching the reference data. While there is a potential 3.17% overshoot in the near-wall peak value of $\langle u'u' \rangle$, the profile in the core of the channel is significantly closer to the reference than it is for the other grid resolutions. The results have demonstrated that OpenSBLI can reproduce turbulent DNS data for both low and high-speed channel flows. It has been shown that care must be taken when selecting the streamwise grid resolution.

4.8.2 Numerical scheme comparison

The final validation case investigates the performance of a selection of schemes in OpenSBLI for compressible turbulence. The simulations are performed with 4th and 6th order central schemes, 5th order WENO-JS and WENO-Z, 5th and 6th order TENO, and a 6th order adaptive TENO-A scheme. Referring to the parameters defined in section 2.4, the TENO C_T parameter is taken to be 1×10^{-5} and 1×10^{-7} for 5th and 6th orders respectively, with $\alpha_1 = 10.5$ and $\alpha_2 = 4.5$ for the TENO6-A scheme.

Figure 4.14 shows the RMS values of the velocity fluctuations on the baseline $(N_x, N_y, N_z) = (129, 129, 129)$ grid. The dissipative nature of the shock-capturing schemes is clearly evident, with significant overshoots in the peak value of $\langle u'u' \rangle$. While the two central schemes match the reference data of [Coleman et al. \(1995\)](#) well, the worst performing WENO5-JS scheme results in a 33.5% overshoot for the peak value on these relatively coarse grids. The WENO5-Z scheme performs noticeably better for a comparable computational cost, but is still too dissipative to match the reference data. The results for the WENO schemes are consistent with the behaviour seen for turbulent channel flows in [Matsuyama \(2014\)](#), and the transitional and turbulent cases detailed in chapter 5 of this thesis.

The TENO schemes in contrast perform more admirably, with the adaptive TENO6-A scheme exhibiting an overshoot of only 3.3% in $\langle u'u' \rangle$ compared to the reference data. This corresponds to an order of magnitude improvement in the relative error compared to the more dissipative WENO5-JS scheme. The profiles for $\langle v'v' \rangle$ and $\langle w'w' \rangle$ show similar trends, WENO5-JS is the worst performing of the shock-capturing options for compressible turbulence. Both the standard and adaptive 6th order TENO schemes perform well at this resolution and are close to the results from the central scheme. Finally, the shape of the Reynolds stress $\langle u'v' \rangle$ is seen to correctly converge to one at the centre of the channel with small discrepancies observed in the near-wall region. There is an under prediction of $\langle u'v' \rangle$ in the region of $y = -0.8$ for the WENO schemes on the order of 6.31%. The TENO and central schemes meanwhile produce consistent results throughout the channel.

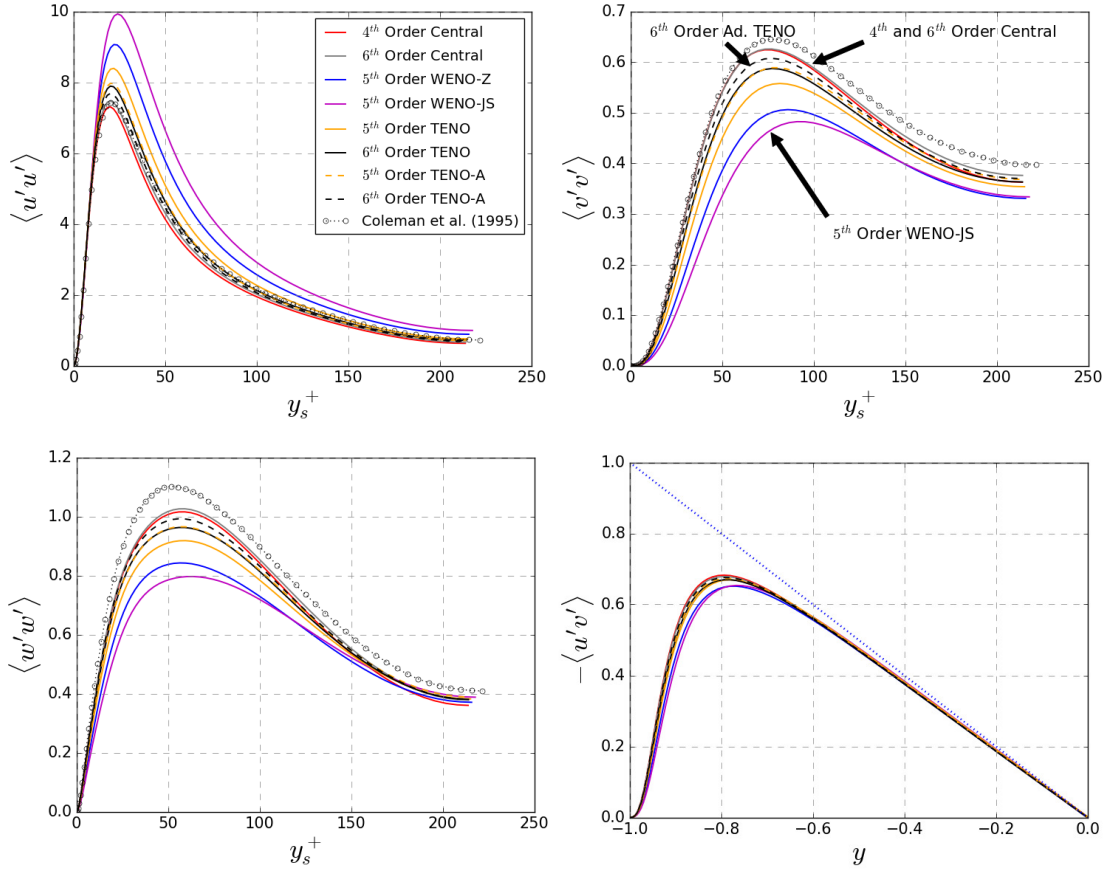


FIGURE 4.14: Numerical scheme comparison for the supersonic turbulent channel flow of Coleman et al. (1995).

This chapter has demonstrated the validity of the OpenSBLI code and numerical method implementations for a suite of challenging test cases. The ability of the code to capture both normal and oblique shock-waves has been demonstrated, including a laminar SBLI involving flow separation. The laminar SBLI also acts as a validation of the boundary-layer initialisation from section 3.3.7, used throughout the remainder of this thesis. The implementation of wall boundary conditions described in section 3.3.6 were validated for both laminar and turbulent channel flow simulations. Having performed these validations, the subsequent chapters of this thesis apply the same code and methods to a series of complex research applications.

Chapter 5

Assessment of low-dissipative shock-capturing schemes for transitional and turbulent shock interactions

5.1 Introduction

Simulation of the compressible Navier-Stokes equations at high Mach numbers requires numerical schemes that are capable of resolving discontinuities in the form of shock-waves. Standard central differencing generates spurious oscillations at discontinuities and becomes unstable, making it unsuitable for shocked flows. To ensure stability a common approach is to add numerical dissipation to smear shocks over several grid points while trying to maintain the accuracy of the solution. However, such shock capturing methods also introduce high levels of numerical dissipation to the whole flow-field, making them a poor choice for resolving small scale structures in transitional and fully turbulent flows. To obtain similar resolution to a non-dissipative scheme, excessively fine grids are required. These large grids are impractical for DNS and severely limit the scale of the problem that can be tackled. Hybrid methods pair a non-dissipative scheme with shock-capturing applied only in areas of high gradients, reducing the numerical dissipation that is introduced into the flow field. While these methods are widely used, care has to be taken with the choice of shock-sensor and tuning may be required between different physical problems. In the past few decades considerable efforts have been made to develop numerical schemes that can sharply resolve shock-waves while minimising detrimental effects elsewhere in the flow.

In this chapter the efficacy of a selection of low-dissipative and hybrid WENO/TENO shock-capturing methods is assessed in the OpenSBLI finite-difference framework, for transitional and turbulent flows containing shock-waves. The classic subsonic Taylor-Green vortex problem ([Brachet et al., 1983](#)) is extended to a range of Mach numbers up to Mach 1.25, where compressibility

and dilatational dissipation become important. Secondly, the shock-induced transition of a flat-plate laminar boundary layer with upstream disturbances is investigated for an oblique shock reflection with initial deflection $\theta = 2.5^\circ$ at Mach 1.5. The results in this chapter were presented at the American Institute of Aeronautics and Astronautics (AIAA) Aviation Forum in Dallas, Texas (June 2019) (Lusher and Sandham, 2019a).

The effect of increasing Mach number is demonstrated for the supersonic Taylor-Green vortex case with a Reynolds number of $Re = 1600$, selected to ensure a wide range of scales are present to test the numerical schemes. For the second test case forced upstream disturbances are added to a Mach 1.5 laminar shockwave/boundary-layer interaction (SBLI). Amplification of the disturbances and a transition to turbulence is observed downstream of the separation bubble. In both cases the effectiveness of low-dissipative shock capturing methods is assessed, in terms of their resolving ability and computational cost. The Implicit Large Eddy Simulation (ILES) approach uses the numerical dissipation intrinsic to the scheme to act as a sub-grid model found in conventional LES approaches. An additional aim of this chapter is to assess whether implicit LES methods can correctly compute the complex supersonic laminar-turbulent transition for the TGV case. In particular, the transitional SBLI assesses whether implicit LES is a feasible approach for a shock-induced breakdown to turbulence with flow separation and amplification of upstream disturbances. The discontinuity-capturing ability of low-dissipative methods is also examined for an inviscid shock reflection.

5.2 Supersonic Taylor-Green vortex

5.2.1 Problem specification

The Taylor-Green vortex (TGV) is a well-known test case consisting of the laminar-turbulent transition of a decaying vortex. A seminal investigation of this problem was presented in the work of Brachet et al. (1983), who investigated both the inviscid and viscous subsonic TGV cases. Part of this study highlighted the intrinsic symmetries of the initial vortex condition. Despite the several decades that have passed since this work, the Taylor-Green vortex remains a physically interesting problem for the study of vortex dynamics and breakdown to turbulence (Peng and Yang, 2018; Sharma and Sengupta, 2019). The TGV case is also widely used for assessing the efficiency of numerical schemes and often acts as a validation of CFD codes (Bull and Jameson, 2015; Jacobs et al., 2017; Houba et al., 2019). The standard subsonic ($M_\infty = 0.1$, $Re = 1600$) case (DeBonis, 2013) is extended in this chapter up to $M_\infty = 1.25$, where the presence of shocklets and compressibility effects become an important factor. The aim of this section is both to elucidate the intricate flow structures for a supersonic TGV, and to use it as a challenging test case for assessing shock-capturing schemes. A comparable physical study of high Mach number TGV was presented by Peng and Yang (2018) up to Mach 2, albeit at a lower Reynolds number of $Re = 400$. At this Reynolds number the range of turbulent scales is not large enough to show significant differences between the numerical schemes.

The initial condition is taken from DeBonis (2013), solving the compressible Navier-Stokes equations on a domain of size $0 \leq x \leq 2\pi L$, $0 \leq y \leq 2\pi L$, and $0 \leq z \leq 2\pi L$ as in figure 5.1. Periodic boundary conditions were applied in all directions. The domain was initialised as

$$u(x, y, z, t = 0) = \sin\left(\frac{x}{L}\right) \cos\left(\frac{y}{L}\right) \cos\left(\frac{z}{L}\right), \quad (5.1)$$

$$v(x, y, z, t = 0) = -\cos\left(\frac{x}{L}\right) \sin\left(\frac{y}{L}\right) \cos\left(\frac{z}{L}\right), \quad (5.2)$$

$$w(x, y, z, t = 0) = 0, \quad (5.3)$$

$$p(x, y, z, t = 0) = \frac{1}{\gamma M_\infty^2} + \frac{1}{16} \left(\cos\left(\frac{2x}{L}\right) + \cos\left(\frac{2y}{L}\right) \right) \left(2 + \cos\left(\frac{2z}{L}\right) \right). \quad (5.4)$$

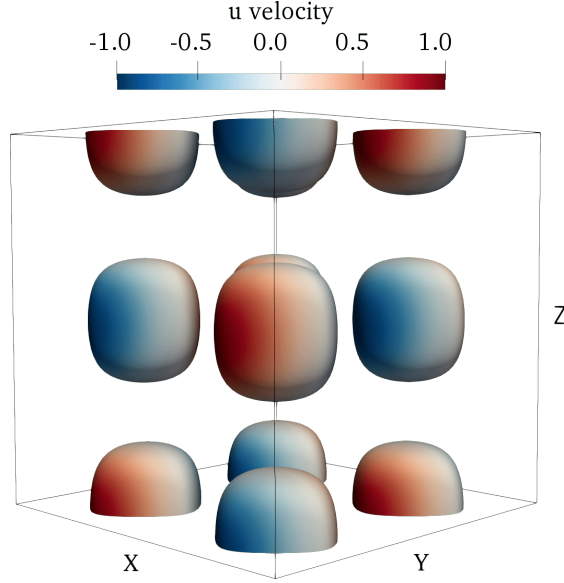


FIGURE 5.1: Initial condition for the 3D Taylor-Green vortex test case. The domain is a periodic box of size $2\pi^3$. A Q-criterion of 0.25 is displayed, coloured by streamwise velocity.

Reynolds, and Prandtl numbers were set to $Re = 1600$ and $Pr = 0.71$ respectively, the ratio of heat capacities was taken to be $\gamma = 1.4$. The length L is taken to be 1, with the initial density evaluated from the equation of state (2.7) as

$$\rho(x, y, z, t = 0) = \gamma M_\infty^2 p. \quad (5.5)$$

Temperature dependent dynamic viscosity was computed by Sutherland's law from Peng and Yang (2018) as

$$\mu(T) = \frac{1.4042T^{1.5}}{T + 0.40417}. \quad (5.6)$$

A fixed non-dimensional time-step of $\Delta t = 5 \times 10^{-4}$ is set throughout, based on the stability requirements for the finest $N = 1024^3$ mesh simulation. Larger time-steps could be used to improve computational efficiency for the coarser meshes. The TGV case is advanced until a non-dimensional time of $t = 20$. To assess the numerical schemes, comparison is made for the kinetic energy integrated over the domain

$$E_k = \frac{1}{\rho_{\text{ref}} \Omega} \int_{\Omega} \frac{1}{2} \rho u_j u_j d\Omega, \quad (5.7)$$

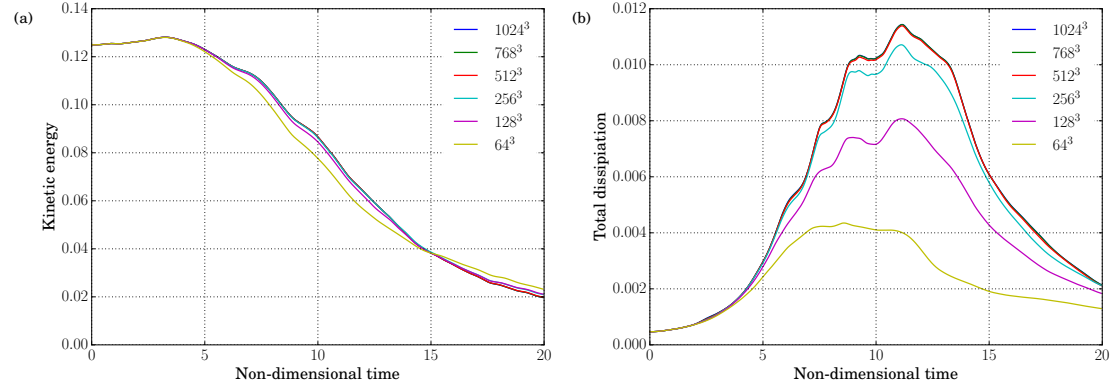


FIGURE 5.2: Grid sensitivity of the $M_\infty = 1$ compressible Taylor-Green vortex problem for (a) kinetic energy and (b) total dissipation with the TENO6 scheme.

and total viscous dissipation rate

$$\varepsilon^T = \varepsilon^S + \varepsilon^D = \frac{1}{\rho_{\text{ref}} Re} \int_{\Omega} \mu \left(\epsilon_{ijk} \frac{\partial u_k}{\partial x_j} \right)^2 d\Omega + \frac{4}{3\rho_{\text{ref}} Re} \int_{\Omega} \mu \left(\frac{\partial u_j}{\partial x_j} \right)^2 d\Omega, \quad (5.8)$$

corresponding to the solenoidal (enstrophy) and dilatational contributions to the viscous dissipation rate for compressible turbulence (Sarkar et al., 1991; Zeman, 1990). Q-criterion is calculated in this chapter from Chong et al. (1990) as

$$Q = \frac{1}{2} \left[\left(\frac{\partial u_i}{\partial x_i} \right)^2 - \frac{\partial u_i}{\partial x_j} \frac{\partial u_j}{\partial x_i} \right]. \quad (5.9)$$

5.2.2 Grid refinement study

Figure 5.2 shows the compressible Taylor-Green vortex problem at Mach 1 for meshes ranging from 0.25 million to 1 billion grid points, computed with the 6th order TENO scheme. The kinetic energy in figure 5.2 (a) is largely grid independent, with only the coarsest 64^3 case showing a clear deviation from the converged result. At 64^3 the kinetic energy is underestimated between $5 < t < 15$ and over-estimated at $t > 15$. The total dissipation (5.8) is displayed in figure 5.2 (b) and shows a much greater dependency on mesh refinement. Meshes between 64^3 and 256^3 substantially underestimate the peak dissipation rate. Above 512^3 a converged solution is obtained. Comparison of Mach number effects in the next section use the 512^3 grid, while the scheme comparisons are performed on coarser 256^3 meshes to assess how the schemes perform as ILES when limited by grid resolution. The kinetic energy and viscous dissipation rate were calculated in the domain every 100 time-steps and integrated over the volume.

5.2.3 Physical effects of increasing Mach number

Figure 5.3 shows the effect of Mach number scaling for the TGV problem over the range $M_\infty = [0.1, 1.25]$. In figure 5.3 (a) the first significant deviation from the subsonic $M_\infty = 0.1$ case is observed at $M_\infty = 0.5$ in the kinetic energy distribution. Unlike the low-subsonic case, the

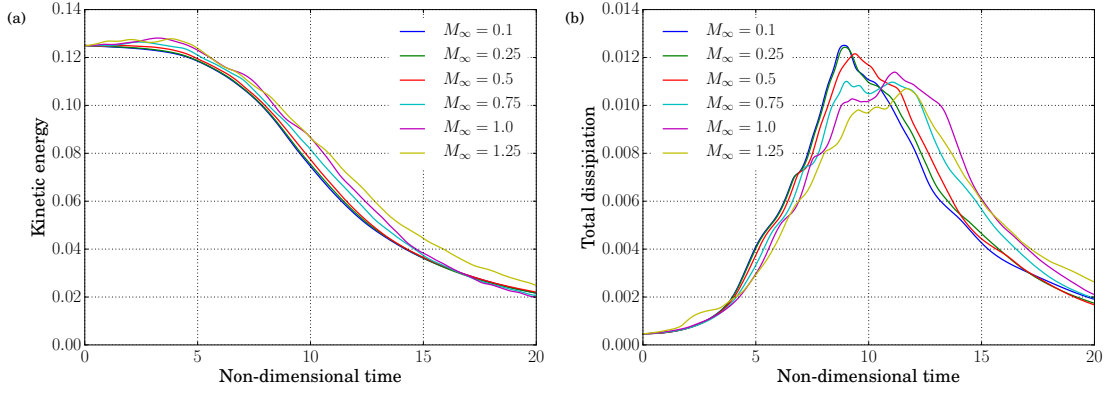


FIGURE 5.3: The effect of increasing Mach number for the compressible $Re = 1600$ Taylor-Green vortex problem on 512^3 grids, displaying (a) kinetic energy and (b) total dissipation.

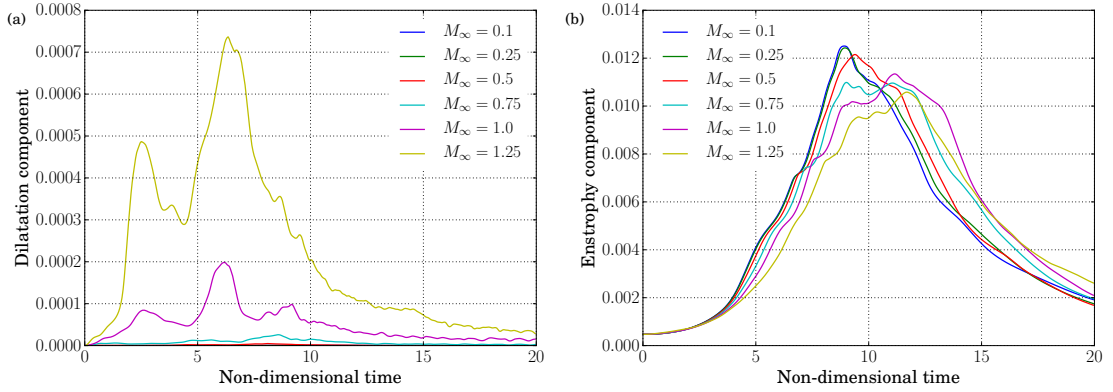


FIGURE 5.4: The effect of increasing Mach number for the compressible $Re = 1600$ Taylor-Green vortex problem on 512^3 grids, displaying (a) dilatational dissipation and (b) solenoidal dissipation.

compressible problem has regions of increasing kinetic energy rather than a constant decline. This is consistent with the compressible TGV simulations of Peng and Yang (2018), where increases in the kinetic energy were reported during the time interval $2 < t < 4$. In compressible flows the average pressure-work can exceed the viscous dissipation rate, as internal energy is converted into kinetic energy during the early stages of the problem. The increase in kinetic energy is most obvious at $M_\infty = 1.25$, but is clearly evident once the freestream Mach number reaches 0.75. There is good agreement between the $M_\infty = 0.1$ and $M_\infty = 0.25$ results, suggesting that incompressible approximations remain largely valid at $M_\infty = 0.25$.

Figure 5.3 (b) shows the total viscous dissipation for the same range of Mach numbers. A large spread of dissipation curves is observed, the main trends being a delay and a flattening of the peak dissipation rate for increasing Mach numbers. Similar to the kinetic energy, peak dissipation rate for the $M_\infty = 0.25$ curve matches closely to the $M_\infty = 0.1$ result. For the higher Mach numbers the dissipation rate is significantly higher in the $12 < t < 20$ region than in the incompressible limit. At Mach 1.25 an additional peak is seen for the early $2 < t < 4$ time period, similar to the one observed in Peng and Yang (2018). The total viscous dissipation can be decomposed into the dilatational and solenoidal components shown in figure 5.4 (a) and (b) respectively. There are two distinct peaks in the dilatational contribution for the Mach 1 and Mach 1.25 curves, the latter corresponding to the peak seen in figure 5.3 (b) for $2 < t < 4$. Shocklets are

generated during the early stages of the high Mach number simulations, a compressibility effect that is not present at lower Mach numbers. At Mach 1.25 the magnitude of the dilatation is still small compared to the solenoidal contribution, but at even higher Mach numbers is expected to match the solenoidal dissipation (Peng and Yang, 2018). The combination of vortex dynamics, shocklets, and small-scale turbulence makes this a challenging problem for numerical schemes.

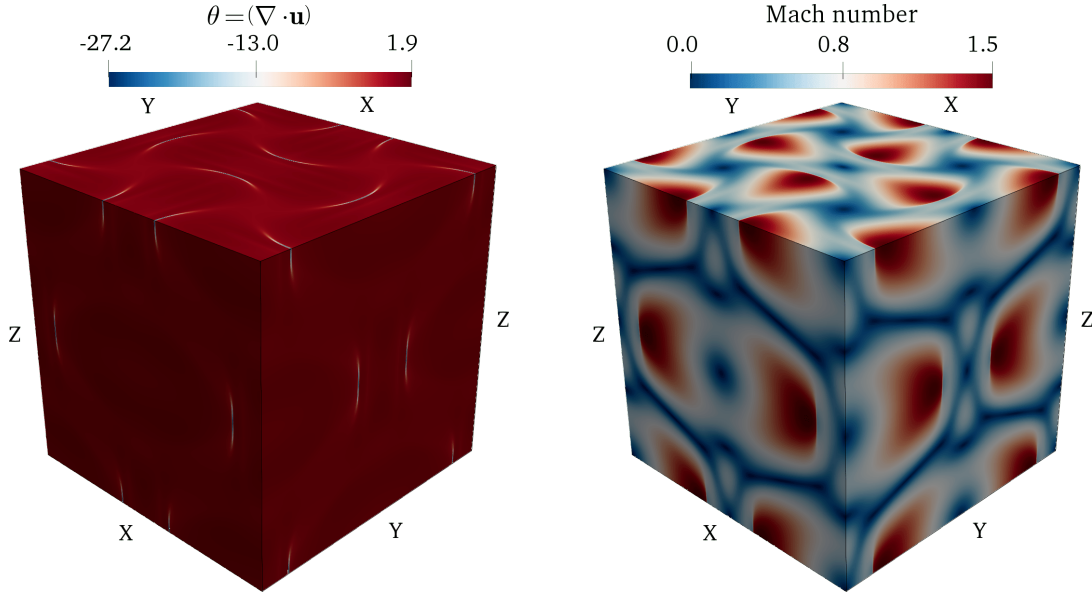


FIGURE 5.5: Formation of shock-waves during the early stages of the compressible $M_\infty = 1.25$ Taylor-Green vortex case. Showing (left) dilatation rate and (right) local Mach number at a simulation time of $t = 2$.

To further demonstrate the physical complexity of this test case, figure 5.5 shows a three-dimensional surface during the initial stages of the Mach 1.25 simulation on the $N = 512^3$ mesh. The simulation time is taken to be $t = 2$, corresponding to the first dilatational peak shown in figure 5.4. The left and right images show the instantaneous dilatation rate ($\nabla \cdot \mathbf{u}$), and the three-dimensional local Mach number

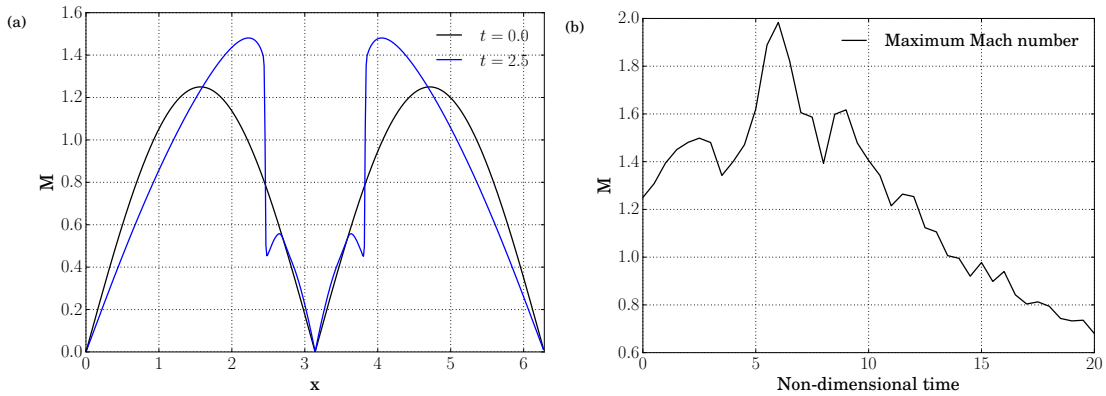


FIGURE 5.6: Formation of shock-waves in the compressible Mach 1.25 Taylor-Green vortex case. Showing (a) discontinuous Mach profile along the $y, z = \pi$ line and (b) time evolution of the maximum Mach number in the domain.

$$M = \frac{|U|}{a} = \frac{\sqrt{u^2 + v^2 + w^2}}{a}. \quad (5.10)$$

Recalling the observation that negative dilatation rates are associated with shock-waves (Johnsen et al., 2010), we see multiple thin well-resolved shock-waves throughout the domain in white and blue. Depending on the orientation, these shock-waves propagate both inwards and outwards relative to the periodic boundaries of the domain. The corresponding Mach number surface in figure 5.5 (right) shows the high-speed regions adjacent to the shock-waves. At this simulation time of $t = 2$, regions of the flow have locally accelerated from $M_\infty = 1.25$ up to Mach 1.5. On the upper surface of the Mach number plot five distinct low-speed regions can be seen. As we will see, the four of these outside of the centre correspond to counter-rotating flow driven by the relative propagation of the initial shock-waves.

Figure 5.6 (a) shows two line profiles in x on the centreline plane of $y, z = \pi$. The smooth black line is the initial condition (5.1) evaluated at $t = 0$. The blue line corresponds to the simulation at $t = 2.5$, to demonstrate the discontinuous shocked regions of the flow. Either side of the centreline, a sharp well-resolved shock-wave is observed as a sudden drop in the Mach number from $M = 1.5$ to $M = 0.45$. The smooth initial distribution rapidly develops into a discontinuous shock-wave due to the non-linearity of the compressible Navier-Stokes equations. For subsequent time snapshots, the rounded top of the Mach profile sharpens and the shock-waves propagate away from each other. In other parts of the domain shock-waves cross the periodic boundaries and interfere with each other. In figure 5.6 (b), the time evolution of the maximum Mach number within the volume is shown. From the initial freestream Mach number of 1.25, a peak Mach number of 2 is observed at $t = 6$. Locally supersonic flow is observed even in the latter stages of the simulation, up until a time of $t = 14$.

Turning the attention to the small-scale structures for this transitional case, figure 5.7 shows the generation of vortical structures at a Q -criterion of $Q = 2.5$, coloured by the streamwise velocity. For the plots clockwise from the upper left corner, the flow is shown at simulation times of $t = [2, 7, 15, 20]$. At $t = 2$ the propagation of the shock-waves observed in figure 5.5 (left) can be seen to be generating non-zero values of Q . The opposite colouring of the shock-surfaces demonstrates the relative motion of the shock-waves in the early stages of the simulation. The second Q plot at $t = 7$ corresponds to the second peak of dilatational dissipation from figure 5.4. Two distinct regions of vorticity generation are observed for this Q value at $z = \frac{\pi}{2}$ and $z = \frac{3\pi}{2}$. The relative motion of the fluid has led to the creation of thick vortical structures that merge, twist, and reconnect with one another.

Figure 5.8 shows a top-down view of the same structures at $t = 7$, albeit with a larger value of $Q = 12.5$ to highlight the largest structures. In each of the two structures from figure 5.7, the relative motion of the initial shock-waves is seen to generate elongated vortex structures along the domain boundaries and the principal axes $x, y = \pi$. The shock-waves propagating along these axes pass through the periodic boundaries and reflect from one another. This can be seen from the opposing regions of dark red and blue in close proximity to one another. Four distinct quadrants are formed due to the symmetries of the initial condition 5.1. On the perimeters of each of these quadrants, small vortex structures are generated from the relative motion of the fluid along the principal axes. We refer once more to the upper Mach number surface of figure 5.5, where the tails of the rotating low-Mach regions match the location of these small-scale vortices.

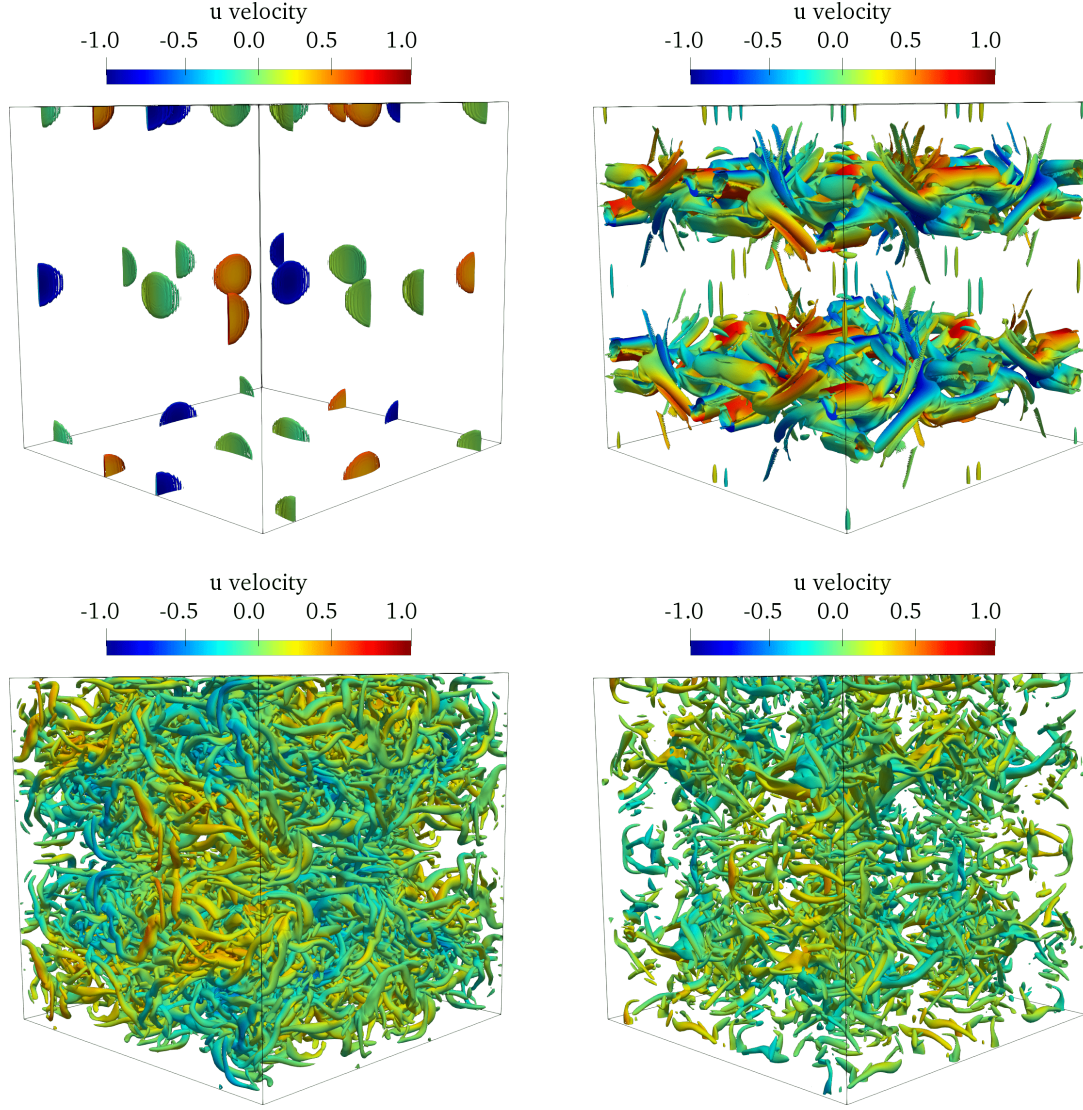


FIGURE 5.7: Vortical structures in the $M_\infty = 1.25$ Taylor-Green vortex case at $Q = 2.5$ coloured by the x component of velocity. Showing simulation times of $t = [2, 7, 15, 20]$ in the clockwise direction starting from the upper left corner.

As this simulation time corresponds to the largest peak in dilatational dissipation from figure 5.4, we conclude that the reinforcement and reflections of the propagating shock-waves play a crucial role in determining the viscous dissipation. As such, robust shock-capturing schemes that can sharply resolve the shock-waves are essential for the correct prediction of viscous dissipation rates.

The final two Q -criterion plots in figure 5.7 show the breakdown to turbulence in the latter stages of the simulation. Large vortices are observed to populate the entire computational box as the turbulent mixing rates increase. At the final $t = 20$ stage of the simulation, the breakdown and cascade to smaller turbulent scales is evident. The four-quadrant structure in the $(x-y)$ plane is shown at $z = \pi$ for the same four simulation times of $t = [2, 7, 15, 20]$ in figure 5.9. The quantity plotted is the instantaneous magnitude of density gradients

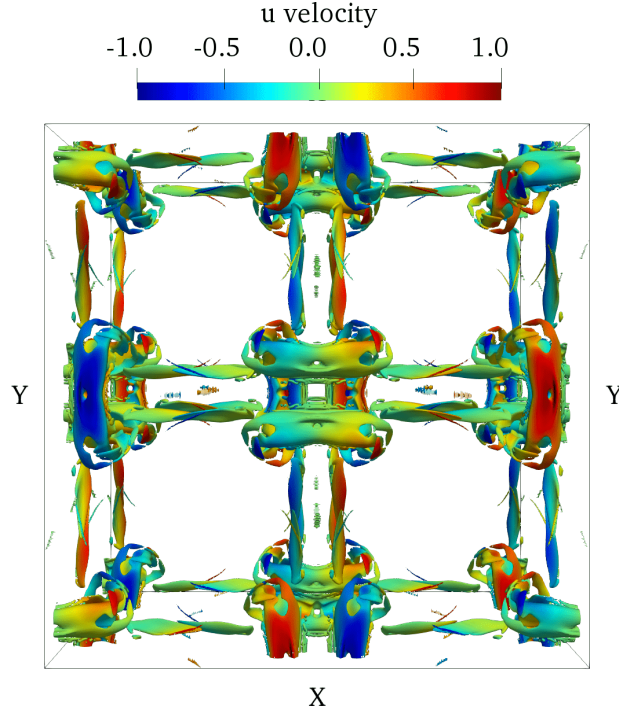


FIGURE 5.8: Top view of the vortical structures in the $M_\infty = 1.25$ Taylor-Green vortex case at $Q = 12.5$ coloured by velocity. Showing an $(x-y)$ view of the simulation at a time of $t = 7$.

$$|\nabla\rho| = \sqrt{\left(\frac{\partial\rho}{\partial x}\right)^2 + \left(\frac{\partial\rho}{\partial y}\right)^2 + \left(\frac{\partial\rho}{\partial z}\right)^2}. \quad (5.11)$$

Unlike the subsonic case which has only minor density variations, the supersonic TGV shows numerous regions of large density gradients. Due to the trigonometric symmetries of the problem (5.1), eight distinct shock-waves are observed in figure 5.9 (a) at $t = 2$ on this centreline plane. The shocks propagate away from the central $(x, y = \pi)$ axis point. By $t = 7$ the shock-waves have reflected and formed elongated structures along the two axes, which were observed previously in figure 5.8. The four quadrants are qualitatively similar to the compressible TGV simulations shown in figure 4 of Peng and Yang (2018). At $t = 15$ where the flow has become subsonic (figure 5.6), small regions of high-density gradients are still observed to propagate along the axes and domain boundaries. The relative motion of the fluid along the axes generates counter-rotating flow in each of the four visible quadrants. Figure 5.9 (d) shows the structures present at the final simulation time of $t = 20$. Despite the breakdown to small-scale structures observed in figure 5.7, at the larger scales the four quadrant symmetry is still present during this late stage of the simulation.

Figures 5.10 (a)-(e) show the magnitude of density gradients in the $(x-z)$ plane on the $y = \pi$ centreline, during the early-to-mid stages of the simulation ($t \in [0, 12.5]$). In this plane at $t = 2.5$ well-defined normal shocks are observed, straighter than those seen in the $(x-y)$ plane of figure 5.9 (a). The middle two shock-waves travel in opposite directions to the ones at the domain boundaries. By $t = 5$ in figure 5.9 (b), the initial shock-waves have interacted with one another, with regions of both constructive and destructive interference. At this time multiple shocks can be seen moving obliquely relative to the coordinate axes. The large-scale shock-structures

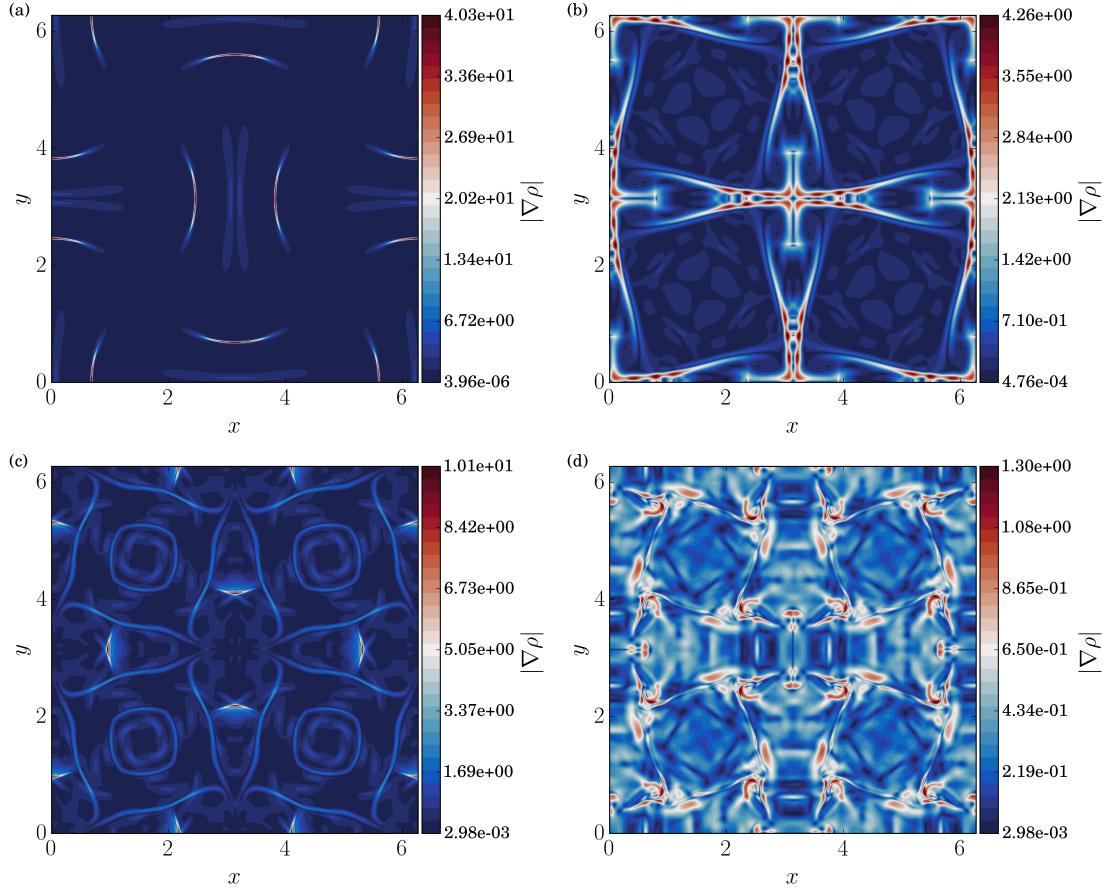


FIGURE 5.9: Magnitude of density gradients $|\nabla\rho|$ for the compressible Taylor-Green vortex case at $M_\infty = 1.25$ on a 512^3 grid. Showing the flow structures at simulation times of $t = [2, 7, 15, 20]$ in (a) to (d), on the centreline (x - y) plane located at $z = \pi$.

persist at $t = 7.5$, shortly after the peak local Mach number shown in figure 5.6 (b). Finally at $t = 7.5$ and $t = 10$, two layers of vortical structures are observed: one in the region of $0 < z < \pi$, and the other within $\pi < z < 2\pi$. These large-scale vortex structures correspond to the two layers of Q surfaces seen in figure 5.7 and figure 5.8. Finally at $t = 12.5$ in figure 5.10 (e), eight compact high density-gradient regions are observed in the cross-section of the vortex structures. The travelling shock-waves are interconnected and interfere with these vortex tubes. As a result, correctly capturing the shock-waves is essential to reproduce the vortex dynamics and subsequent breakdown to turbulence in the latter stages of the problem.

In this section we have seen the additional complexity that is present in a supersonic version of the Taylor-Green vortex test case. In contrast to the standard subsonic $M_\infty = 0.1$ case, the kinetic energy distribution is no longer found to be monotonically decreasing (figure 5.3 (a)). In the early stages of the simulation, internal energy is converted into kinetic energy which can exceed the viscous dissipation rate. While the incompressible approximation is still largely valid at $M_\infty = 0.25$ (figure 5.3 (b)), compressibility effects quickly become important at higher Mach numbers and lead to a delay in the peak dissipation rate. It is possible that this is due to the stabilising effect of increasing Mach number on the growth rate of instabilities, seen previously in compressible mixing-layers and turbulent spots (Vreman et al., 1996; Sandham, 2016). Temporally decaying shock-waves were observed throughout the domain (figure 5.5),

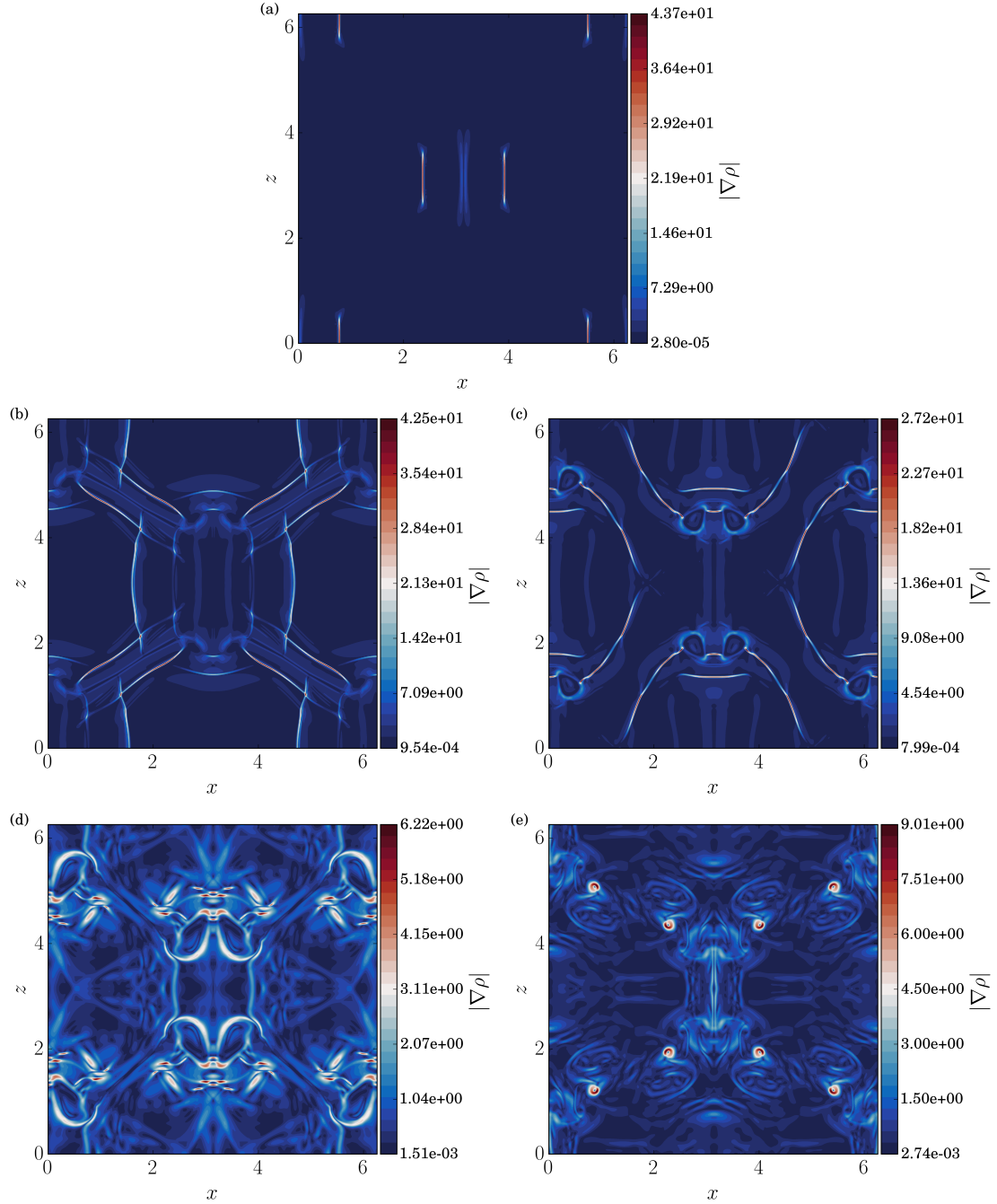


FIGURE 5.10: Magnitude of density gradients $|\nabla\rho|$ for the compressible Taylor-Green vortex case at $M_\infty = 1.25$ on a 512^3 grid. Showing the flow structures at simulation times of $t = [2.5, 5, 7.5, 10, 12.5]$ in (a) to (e), on the centreline (x - z) plane located at $y = \pi$.

demonstrating the need for shock-capturing capabilities for this test case. Despite the similarities to the well-studied subsonic case, the demonstrated compressibility effects make the supersonic Taylor-Green vortex an interesting physical problem in its own right. A supplementary animation was produced to observe the dynamics of this test case. For the purposes of this chapter, the supersonic case poses a suitably challenging problem to test the performance of shock-capturing schemes in the presence of compressible turbulence.

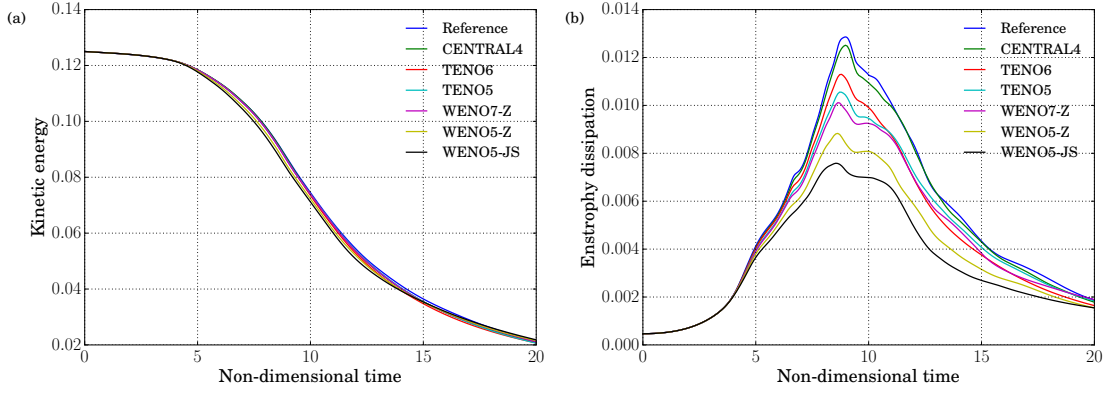


FIGURE 5.11: Taylor-Green vortex $M_\infty = 0.1$ scheme comparison on coarse 256^3 grids for (a) kinetic energy and (b) enstrophy compared to a reference fine 512^3 spectral solution (blue).

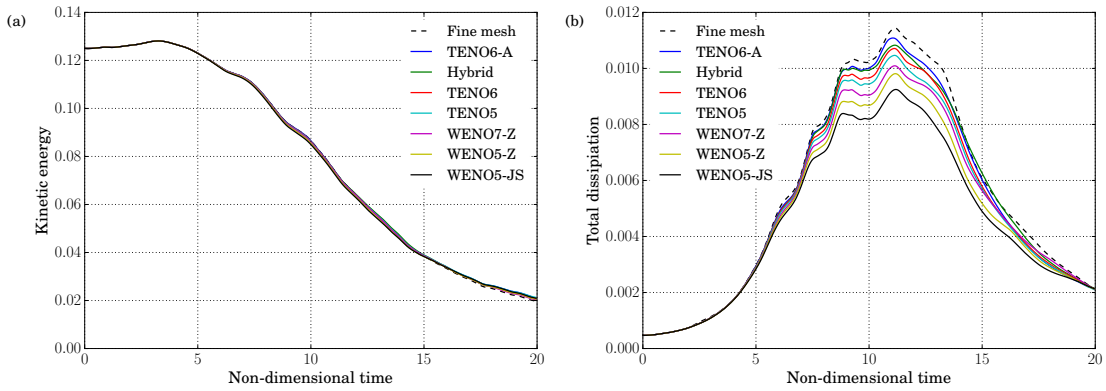


FIGURE 5.12: Taylor-Green vortex $M_\infty = 1$ scheme comparison on coarse 256^3 grids for (a) kinetic energy and (b) total dissipation compared to a fine 1024^3 mesh TENO6 solution (dashed line).

5.2.4 Shock-capturing scheme comparison

In this section, results are shown from simulations at $M_\infty = 0.1$ and $M_\infty = 1.0$, using the 5th order WENO-JS, 5th and 7th order WENO-Z, 5th and 6th order TENO schemes, 6th order TENO6-A and a hybrid central-WENO-Z scheme. In the $M_\infty = 0.1$ subsonic case the schemes are compared for 256^3 grids compared to a 512^3 spectral result from Wang et al. (2013). In the incompressible limit shock capturing schemes are not required for this problem, but the existence of well validated reference solutions allows a comparison of the small-scale resolving ability of the schemes.

Figure 5.11 (a) shows that for $M_\infty = 0.1$, the kinetic energy is largely independent of the scheme at 256^3 ; only the WENO schemes slightly underestimate the reference result. The excessive numerical dissipation of shock capturing schemes is clear to see, however, in figure 5.11 (b), where all of the schemes severely underestimate the dissipation rate compared to a non-dissipative 4th order central scheme. The hybrid scheme (not shown) is identical to the central result for the subsonic $M_\infty = 0.1$ case as the shock sensor is never activated for this problem. The standard WENO5-JS is the worst performing scheme, underestimating the dissipation rate by up to 40%. The WENO-Z results are better, especially when considering the minor modifications the scheme requires compared to the original WENO-JS formulation. The addition of the global smoothness

TABLE 5.1: Runtime for 1000 iterations of the TGV problem at Mach 1 on a 256^3 grid. CPU: Intel Skylake node (40 cores @ 2 GHz, 40 MPI, Intel 17.0 -O3 -fp-model fast). GPU: 1x NVIDIA Volta 16GB V100 (CUDA 9.0, nvcc -O3). The relative computational cost is compared to the WENO5-JS scheme.

Scheme	CPU Runtime (s)	Cost	GPU Runtime (s)	Cost
WENO5-JS	1202.3	-	190.8	-
WENO5-Z	1258.6	1.05	195.9	1.03
WENO7-Z	1651.8	1.37	236.7	1.24
TENO5	1377.1	1.15	201.8	1.06
TENO6	1683.8	1.40	216.5	1.13
TENO6-Adaptive	1772.3	1.47	223.7	1.17
Hybrid Central4-WENO5-Z	1016.8	0.84	179.0	0.94

measure to the non-linear weights (2.25) substantially lowers the numerical dissipation of the WENO scheme, making it difficult to recommend the standard WENO-JS over WENO-Z when numerical dissipation is an important consideration. While still below the non-dissipative central scheme, both the 5th and 6th order TENO schemes perform well; low numerical dissipation is achieved while the schemes retain a good shock capturing ability.

Having demonstrated the numerical dissipation inherent to each of the schemes for an incompressible problem, the natural extension is to see how they fare at higher Mach numbers where shock capturing is also required. Figures 5.12 (a) and (b) show the kinetic energy and total dissipation at Mach 1 compared to the 1024^3 fine mesh result from figure 5.2. All of the schemes agree well for the kinetic energy evolution, but a large variance is again seen for the dissipation rate. The WENO5-JS scheme is the worst performer again, but the spread between the schemes is not as large as in the subsonic case. The TENO schemes prove to be less dissipative than WENO also when shocks are present, but fall short of the hybrid central-WENO-Z scheme. The hybrid and adaptive TENO-6A schemes with the dilatation/vorticity-based shock sensor (2.54) perform well and give the closest agreement to the fine mesh solution, but may not be suitable for every flow type (Johnsen et al., 2010). While the standard TENO schemes are slightly more dissipative than the hybrid and adaptive TENO-6A methods, they rely only on one tunable parameter and avoid potential tuning of shock sensors for different flow configurations.

5.2.5 Computational cost comparison

In addition to the dissipation and shock-resolving ability of the schemes, an important consideration for DNS is the computational cost they incur. Table 5.1 shows the runtime for 1000 iterations of the TGV problem on the 256^3 grids used for the scheme comparison. In each case multiple runs were performed to determine if there was any variability in the runtime, but repeated runs of the code were always consistent to one decimal place. Each scheme was tested for both a CPU and GPU-based platform, the domain specific language and code-generation described in chapter 3 allows for the same single source-code to be translated to a wide range of architectures. For both platforms the relative runtime was compared to a standard WENO5-JS scheme, to see the increase in computational cost required to achieve low-dissipative shock capturing.

WENO5-Z is an attractive option if very low dissipation is not the primary concern, it performs well for only a 3–5% increase in computational cost over WENO5-JS. It is difficult to recommend the use of WENO-JS for this type of problem considering the improvements made by WENO-Z for a negligible increase in cost. The 7th order WENO-Z scheme has both an additional stencil to compute and a 2-point wider numerical stencil than the 5th order scheme, making it up to 37% more expensive in these tests. A 5th order TENO scheme proved marginally more expensive than WENO5-JS, owing to the need to evaluate the discrete cut-off functions in equation 2.4. The 5th and 6th order TENO schemes were 15% and 40% more expensive on CPU and only 6% and 13% more expensive on GPU. Our hypothesis for this difference between CPU and GPU-based computation, is that GPUs are often limited by memory bandwidth/latency despite having large compute capacities. The TENO6 scheme has the same number of candidate stencils and weights to compute as WENO-7Z, but the smaller full numerical stencil of TENO6 leads to better performance, as less data access is required. The adaptive TENO-6A scheme has a further small increase in cost owing to the need to evaluate the shock sensor. Despite concerns about load balancing for hybrid methods that switch between two numerical schemes of differing cost, the hybrid method performs well and is cheaper than the WENO5-JS scheme.

5.3 Transitional shockwave/boundary-layer interaction

Studies on numerical methods are often limited to simple one or two-dimensional test cases, or problems in which idealised fully periodic domains are used. While these are useful validation cases, they may not be representative of more complex research problems. The aim of this section is to assess low-dissipative shock capturing for a more challenging flow case that includes inflow/outflow and wall boundaries. The problem involves shock-induced flow separation and forced transition of a boundary layer, where initial disturbances are convected downstream before being amplified within the recirculation bubble. The presence of an oblique shock reflection, a recirculation bubble, and a transitioning boundary layer makes this a challenging case for numerical schemes. The flow configuration is taken from Sansica (2015), based on the specifications of the EU FP7 TFAST project.

A Mach 1.5 freestream is initialised with the similarity solution for a laminar boundary-layer (White, 2006) described in section 3.3.7, with Reynolds number based on inlet boundary-layer thickness of $Re_\delta = 750$ and a reference temperature of $T_\infty = 202.17K$. The similarity solution provides profiles for temperature, streamwise velocity and wall normal velocity, which are then used to set the conservative variables throughout. Inviscid shock-jump relations are enforced on the upper Dirichlet boundary at $x = 20$, corresponding to a flow deflection of 2.5° and pressure ratio of 1.132. Extrapolation methods are used at the inlet (for pressure) and outlet, the span is periodic, and isothermal no-slip conditions are set on the bottom wall using the non-dimensional wall temperature of $T_w = 1.381$ (4 s.f.) from the similarity solution. The non-dimensional domain size is $(L_x, L_y, L_z) = (310.0, 140.0, 27.32)$, selected to ensure spurious reflections from the top boundary do not cause a secondary separation bubble on the bottom surface. A mesh of $(N_x, N_y, N_z) = (750, 550, 250)$ is selected for the baseline case, with a refinement study given in section 5.3.1.

To introduce upstream disturbances to the otherwise laminar SBLI, modal time-dependent forcing is applied as a piecewise condition on the no-slip wall such that the wall-normal velocity is modified as

$$v = \begin{cases} A \cos(\beta z) \sin(\omega t) \exp\left(-\left(\frac{(x-20)^2}{B}\right)\right) & \text{if } 10 < x < 30 \\ 0 & \text{otherwise,} \end{cases}$$

modelling a blowing/suction effect through the wall. For the standard forced case the constants have numerical values of $A = 2.5 \times 10^{-2}$, $\beta = \frac{2\pi}{L_z} = 0.23$, $\omega = 0.1011$ and $B = 4$, approximating the most unstable mode obtained by linear stability theory of a laminar separation bubble (Sansica, 2015). For the refined and ILES meshes of section 5.3.2 there are 19 and 9 streamwise grid points over the peak of the Gaussian respectively. The simulations were advanced up to a non-dimensional time of $t = 5000$ with $dt = 0.025$, corresponding to 16 flow-through times of the domain. Grid stretching was applied in the wall normal direction with points clustered hyperbolically towards the wall as

$$y = L_y \frac{\sinh(s_1 \xi)}{\sinh(s_1)}, \quad (5.12)$$

for uniformly distributed points $\xi = [0, 1]$ and a stretch factor of $s_1 = 4$. Figure 5.13 shows the activation of the modified Ducros shock sensor in equation (2.54) for the TENO6-A scheme. The sensor activates only at shock-waves where required and is fully turned off in both the laminar and transitional boundary layers. The hyperbolic tangent modification (2.54) is effective at removing all regions of expansion where numerical dissipation is not required. The parameter a in equation (2.54) can be increased if required to increase the activation near shocks.

5.3.1 The effect of upstream disturbances

In this section the shock capturing schemes are tested for a challenging SBLI case that includes transition to turbulence on a fine mesh of $(N_x, N_y, N_z) = (750, 550, 250)$. The sensitivity of the SBLI to upstream disturbances is shown in figure 5.14 for span-averaged (a) wall pressure and (b) wall-normal skin friction. For each case the streamwise length of the separation bubble is defined as the distance between the zero crossings of the span-averaged skin-friction in figure 5.14 (b). As the data is stored on a discrete grid, a univariate interpolation procedure is applied to the skin-friction curve to find the $C_f = 0$ crossing locations.

When no forcing is applied ($A = 0$) a steady laminar SBLI is obtained, with a large separation bubble of $L_{\text{sep}} = 132.5$. Two asymmetrical peaks are seen within the separation bubble, with the largest magnitude of flow-reversal occurring on the reattachment side. After the flow reattaches a laminar profile is recovered with a steady-state solution. Time-dependent disturbances added to the flow in the forced case are convected downstream and become amplified within the separation bubble. The reattached flow downstream of the separation bubble undergoes the early stages of transition to turbulence. There is a sharp increase in skin-friction compared to the laminar solution and a region of unsteady flow near the outlet. The skin-friction downstream of the SBLI is a factor of six larger than the laminar solution. As the flow near the wall begins to transition there are increased rates of mixing, which move higher momentum fluid into the near-wall region, with the effect that the length of the separation bubble is reduced to 22% of the laminar solution.

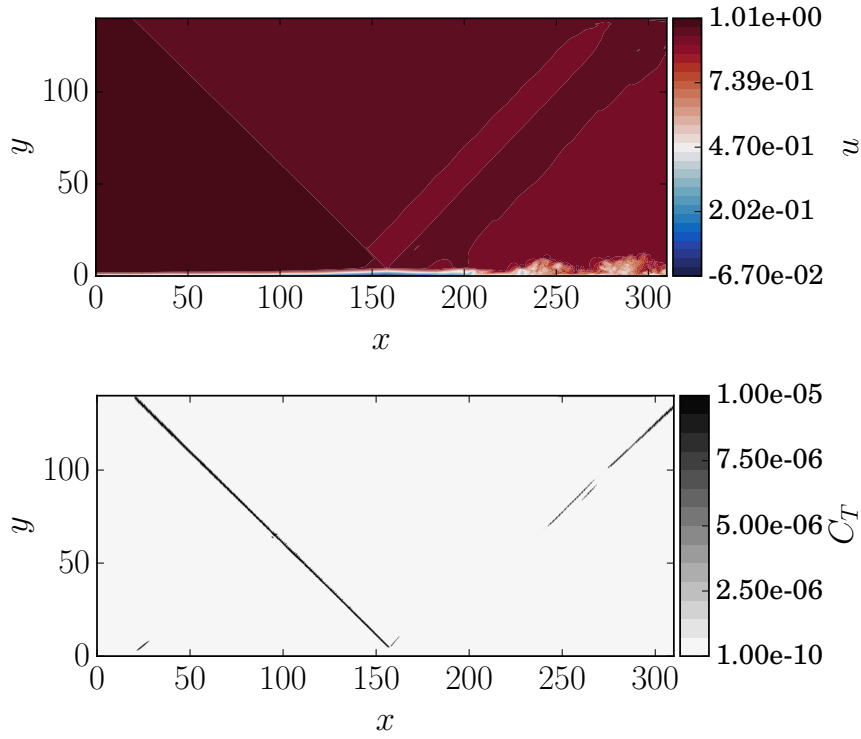


FIGURE 5.13: Activation of the modified Ducros sensor after mapping to the TENO parameter C_T . (a) An instantaneous slice of streamwise velocity u for the transitional SBLI (b) The regions in which the shock sensor is active.

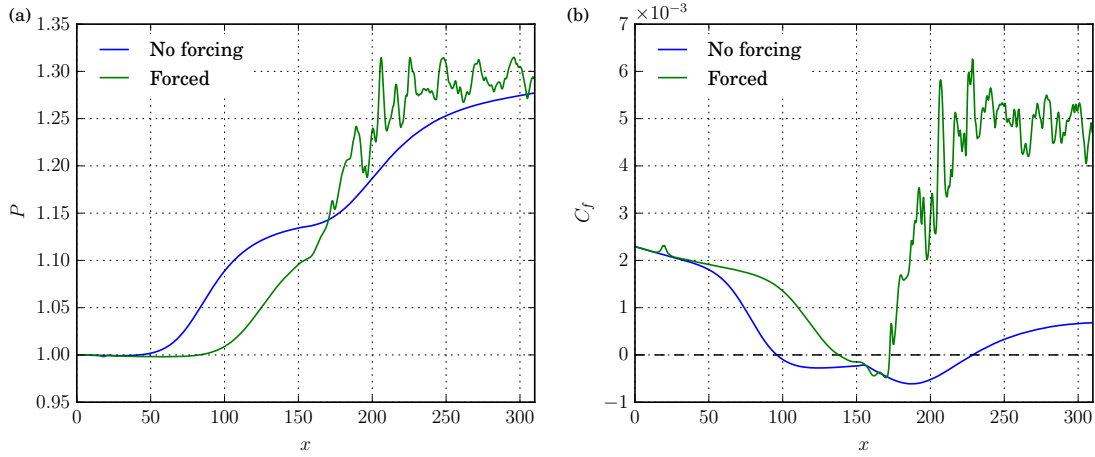


FIGURE 5.14: Instantaneous comparison of the oblique SBLI with and without upstream forced disturbances. (a) span-averaged wall pressure normalized by the inlet pressure, and (b) span-averaged skin friction on the wall.

It can be seen that the zero crossing of the skin-friction shifts towards the centre of the bubble at both the separation and reattachment locations. The transition process is not self-sustained for this relatively weak SBLI, since when the disturbances were turned off the forced case returned to the laminar solution.

Figure 5.15 shows instantaneous 2D slices of density for the forced-transition case. In figure 5.15 (a) the main features of an oblique SBLI are observed, the incident shock causes a thickening of

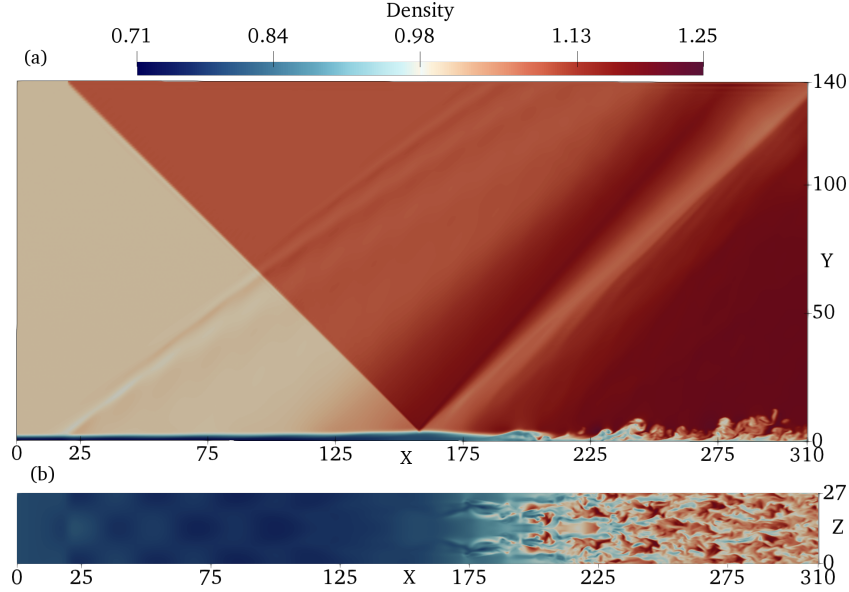


FIGURE 5.15: Instantaneous density contours for the transitional SBLI. (a) x - y slice evaluated at $z = 12$ and (b) x - z slice evaluated at $y = 1$. The transition is symmetric about the centreline until $x = 215$.

the laminar boundary layer and reflects as an expansion fan from the apex of the recirculation bubble. Compression waves can be seen emanating from the separation and reattachment regions of the boundary layer, with a clear transition to turbulence downstream of the reattachment point. At $x = 20$ a weak shock-wave is present due to the flow deflection generated by the blowing/suction effect. The domain is selected to ensure this wave does not reflect back onto the downstream boundary layer. Figure 5.15 (b) shows the flow close to the wall at $y = 1$, including the initial breakdown of the laminar boundary layer. The transition is seen to be symmetric about the centreline up until approximately $x = 215$, which is expected as the span-wise width corresponds to one wavelength of the unstable mode.

To determine the grid sensitivity of the transitional solution, a refinement was performed by adding 50% grid points in each direction independently to the baseline mesh of $(N_x, N_y, N_z) = (750, 550, 250)$. Figure 5.16 shows (a) span-averaged wall pressure and (b) span-averaged skin friction, averaged over 10 periods of the forcing. The laminar portion of the flow and the average length of the separation bubble are observed to be insensitive to further grid refinement. In the transition region a similar insensitivity is observed up until the symmetry of the problem breaks at $x = 215$. Downstream of the broken symmetry there is an initial rise in the skin friction for all of the meshes peaking at $x = 240$, which decreases towards the outlet. Variations between the solutions after symmetry breaks can be attributed to the short averaging period. The averaging length of 10 forcing periods corresponds to two flow-through times of the freestream. Grid independence of the recirculation bubble and initial transition process has been demonstrated. For the scheme comparison in the next section a longer time average is taken.

5.3.2 Scheme comparison on coarse grids

To test the ability of low-dissipative shock capturing schemes to act as an implicit LES, the transitional SBLI is repeated on a coarsened grid of $(N_x, N_y, N_z) = (350, 300, 80)$. A relatively

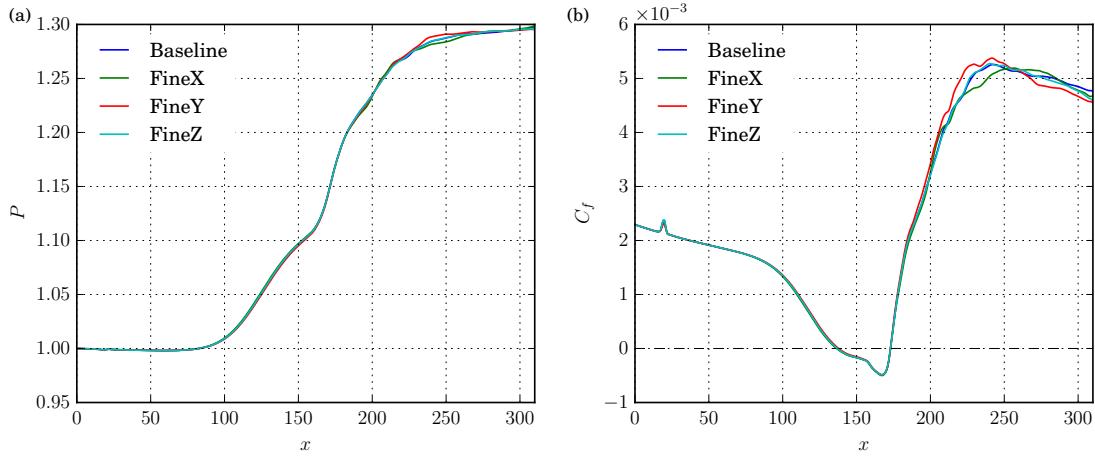


FIGURE 5.16: Grid sensitivity averaged over 10 periods of the forcing for the TENO6-A solution, the baseline mesh has size $(N_x, N_y, N_z) = (750, 550, 250)$. (a) Span-averaged wall pressure normalized by the inlet pressure and (b) Span-averaged skin friction on the wall. The grid is refined by 50% in each direction independently.

conservative coarsening has been applied in the wall normal direction, to maintain a fixed impingement point of the incident shock. Further savings in y would be possible by increasing the strength of the grid stretching and reducing the number of points, however excessive coarsening at the top of the domain leads to a smearing of the incident shock and a shift of the impingement point. The coarse grid for this section corresponds to an overall factor of 12 reduction in grid points compared to the baseline mesh. Previous DNS studies of this problem (Sansica, 2015) used three times finer resolution in the streamwise direction, so the saving over a DNS is expected to be larger still. All of the results in this section were simulated up until a non-dimensional time of $t = 5000$, before being averaged for a further 90 periods of the forcing unless otherwise stated.

Figure 5.17 shows the results for each of the schemes compared to the fine mesh solution of section 5.3.1. In figure 5.17 (b) we see a large disparity between the schemes on coarsened grids, both in the transition region and the size of the recirculation bubble. While the TENO6-A and TENO5 schemes agree well with the separation length of the fine mesh, the hybrid, TENO6-A and WENO5-Z schemes all exhibit a much larger recirculation region. Although there are still small differences between each of the schemes, the results for different schemes cluster into one of two possible transition states. In an attempt to understand the discrepancy between schemes, the transverse velocity w of the initial disturbances inside the boundary layer is shown in figure 5.17 (a). The laminar solution initially has a value of $w = 0$ as there are no gradients across the span until the forcing strip. The time dependent forcing from section 5.3 is seen to create a single frequency oscillating disturbance in w that is convected downstream within the boundary layer. The TENO-A and TENO5 schemes match the disturbances from the fine mesh well, consistent with their skin friction distributions in figure 5.17 (b). The other schemes underestimate the initial impulse and produce disturbances roughly half the amplitude of the fine mesh solution.

The variability between the various schemes in resolving the initial disturbances means that the schemes giving smaller amplitude disturbances have a much larger recirculation bubble. Within each of the two solution classes there are also differences in the reattached transitional boundary layer. For the shorter bubble solutions the TENO6-A is seen to be less dissipative than TENO5,

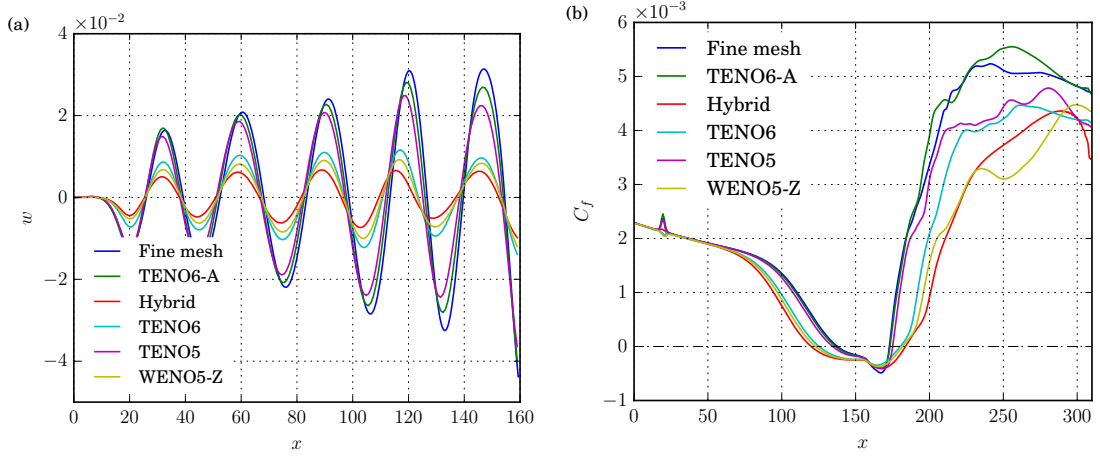


FIGURE 5.17: Transitional SBLI for the different shock capturing schemes. Displaying (a) instantaneous transverse velocity component w in the near-wall region at $y = 1$, $z = 0.25L_z$ showing the initial disturbance and (b) time averaged skin friction along the wall for 90 periods of the forcing.

with the TENO6-A overshooting the fine mesh solution. For the longer bubble cases the TENO6 scheme is less dissipative than the WENO5-Z and hybrid WENO methods. Overall it can be seen that there is an inverse relationship between the magnitude of the disturbances and the length of the recirculation bubble. This suggests that for a transitional SBLI on coarse meshes, sufficient resolution of both the transition region and the initial disturbances is important. The performance of the hybrid scheme is disappointing when considering the compressible TGV result in figure 5.12 (b). Finally, instantaneous slices of streamwise velocity above the bottom wall are shown in figure 5.18 for four of the schemes. Similarities in flow structures are seen between TENO6-A and TENO5 in (a) and (b), and also between TENO6 and WENO5-Z in (c) and (d). The transition location is further upstream in the case of the shorter separation bubbles for (a) and (b).

A blowing/suction strip within the wall is a computationally cheap way of generating disturbances to study transition, but has highlighted the difficulty some schemes can face on coarse grids. Alternatives to forcing strips include the addition of disturbances to the inflow profile or the inclusion of an acoustic source term above the boundary layer. Both of these alternative forcing methods would still need to be spatially resolved however.

5.3.3 Coarser grid implicit-LES with the TENO-A scheme

The performance of implicit LES with the TENO6-A scheme on successively coarser grids is shown in figure 5.19 compared to the fine mesh solution. The coarse $(N_x, N_y, N_z) = (350, 300, 80)$ ILES grid from the previous section is further reduced in the wall normal direction to use $N_y = 100, 200$ with the same stretching factor. The wall pressure distribution in figure 5.19 (a) only shows significant deviation from the fine mesh solution for the coarsest $N_y = 100$ case. At $N_y = 100$ the initial pressure rise occurs earlier in the streamwise direction as the coarsest grid overestimates the length of the separation bubble. A similar picture is seen in the skin friction distribution of figure 5.19 (b). The $N_y = 200$ ILES performs surprisingly well and closely matches the results from much finer grids during the early stages of transition.

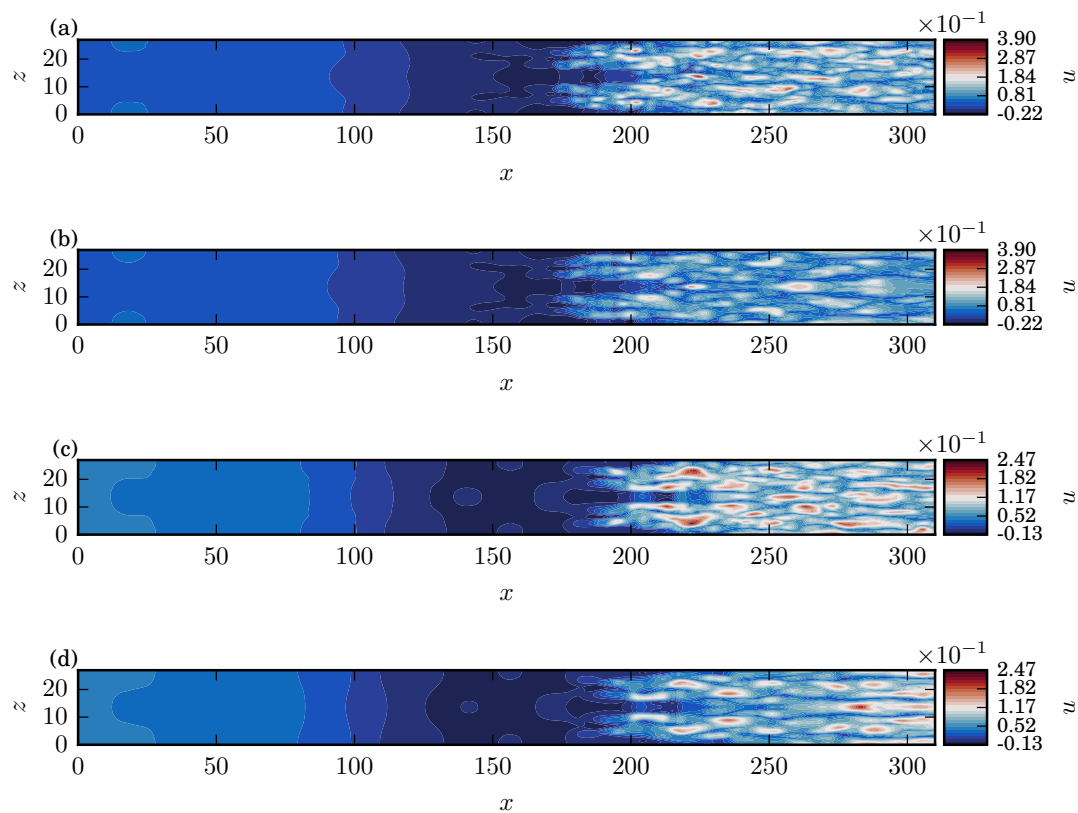


FIGURE 5.18: Instantaneous streamwise velocity u slice above the bottom wall for (a) TENO6-A (b) TENO5 (c) TENO6 and (d) WENO-5Z. Contour levels are fixed for (a)-(b) and (c)-(d) to reflect the difference in separation length between the schemes.

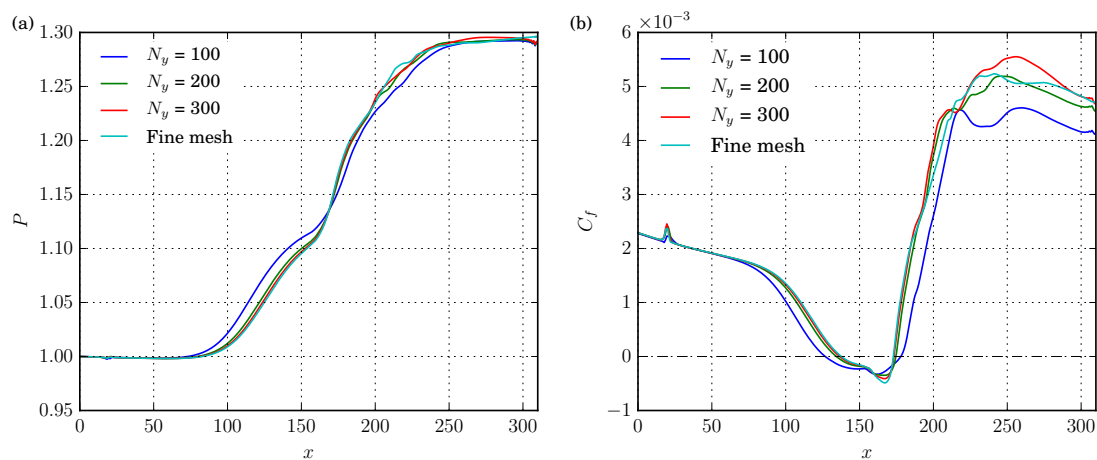


FIGURE 5.19: Transitional SBLI with the TENO6-A scheme. The fine mesh is compared to successive coarsening in the wall normal y direction. Displaying (a) wall pressure normalised by the inlet pressure and (b) time averaged skin friction along the wall for 90 periods of the forcing.

5.4 Weight selection for TENO6 schemes

5.4.1 Inviscid shock reflection test case

In this section a simple two-dimensional inviscid shock reflection is performed at Mach 2 to assess the relative shock-capturing ability between the schemes in the absence of viscosity. The case considered is an inviscid version of the laminar SBLI validation case presented for OpenSBLI in section 4.6. A domain of size $(x, y) = (350, 115)$ is initialised with a uniform Mach 2 inlet on a $(N_x, N_y) = (600, 300)$ grid. An initial oblique shock of $p_2/p_1 = 1.1865$ is imposed on the upper boundary that reflects from an inviscid wall condition at $y = 0$. Each of the schemes should be able to reproduce an inlet/outlet pressure ratio of $p_3/p_1 = 1.4$ in the region after the shock reflection. Figure 5.20 (a) shows the pressure jump at a height of $y = 57.5$ for a selection of the schemes compared to a refined $(N_x, N_y) = (2400, 1200)$ WENO-JS solution. While all of the schemes give the correct $p_3/p_1 = 1.4$ at the outlet, significant defects are seen in the zoomed inset near discontinuities for the TENO schemes. For the standard TENO5 and TENO6 of Fu et al. (2016) there is a noticeable undershoot before the first pressure jump which does not occur with WENO schemes. The defect near discontinuities is similar to that reported by the authors of Motheau and Wakefield (2019) for TENO schemes within a finite volume formulation. There are also increased post-shock oscillations compared to conventional WENO options.

For the TENO6-A scheme we observe a large oscillation at the first pressure jump and a small undershoot before the second. The TENO6-A scheme also shows visible overshoots near discontinuities in Fu et al. (2018) using optimized linear weightings. Figure 5.20 (b) shows the TENO6-A with the optimized linear weights from Fu et al. (2018) compared to the weights given in Fu et al. (2016). The less optimized weights significantly reduce the size of the oscillation before the pressure jump but suffer from larger post-shock oscillations. While it is possible there are differences in the detailed implementation when compared to previous publications, we note the increased pre-shock oscillations occur within the same implementation only when using the optimized linear weights of Fu et al. (2018). The code is written in a modular way

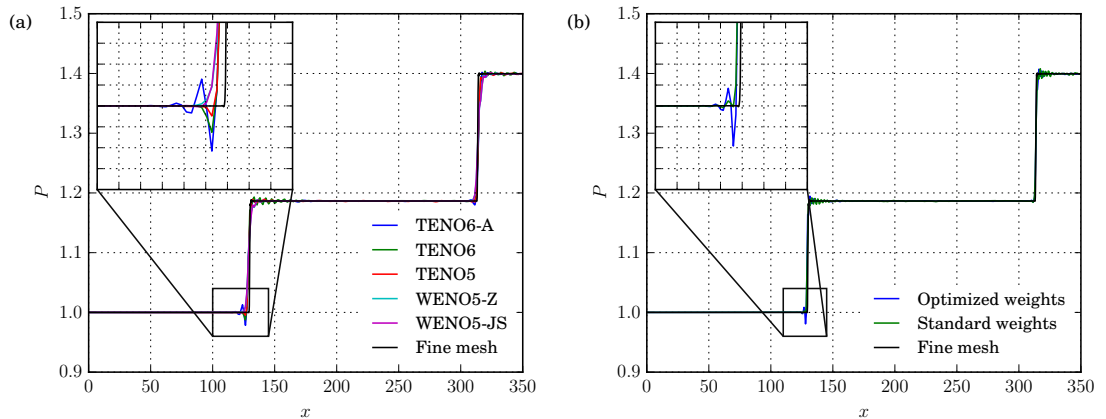


FIGURE 5.20: Comparison of the shock resolving ability for an inviscid shock reflection at Mach 2. Pressure jump over the initial and reflected shock is shown at a height of $y = 57.5$. Showing (a) comparison of the TENO schemes to WENO-JS and WENO-Z and (b) The effect of using the optimized weightings from Fu et al. (2018) for TENO-A.

so the WENO/TENO schemes all share the same flux-splitting and characteristic decomposition; the only differences are in the reconstructions. Further work may be needed to improve the discontinuity capturing for TENO schemes while retaining their low-dissipative qualities for resolving turbulence. This is especially true for transitional shock interactions where numerical oscillations near shocks could unphysically contribute to the sensitive transition process.

5.4.2 Viscous test cases

Finally, the effect of reverting back to the standard weights of [Fu et al. \(2016\)](#) for a TENO6-A ILES is shown in figure 5.21 and figure 5.22, for the compressible Taylor-Green vortex and transitional SBLI respectively. The TGV kinetic energy shows very little sensitivity to the weight change except a small discrepancy near $t = 20$. A more noticeable difference can be seen in the total dissipation rate, with the standard weights proving more dissipative during the peak at $11 < t < 13$. The standard weights show a slight improvement in resolution, however, between

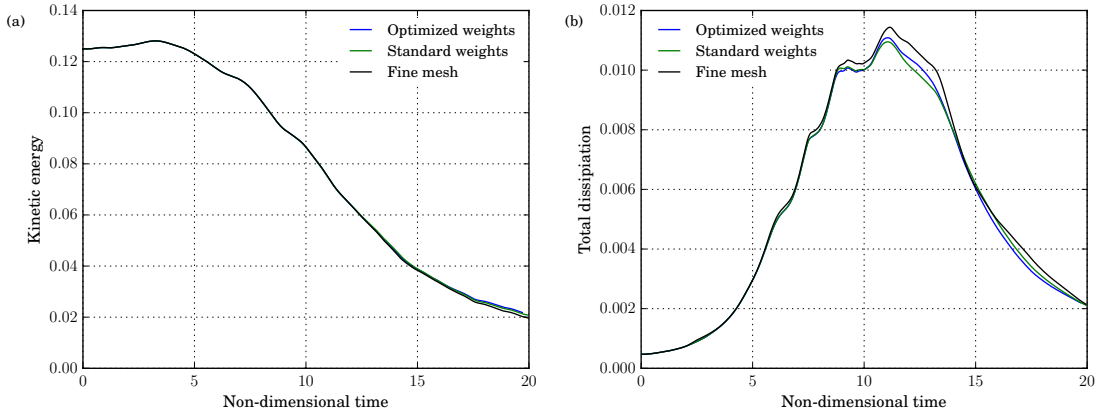


FIGURE 5.21: The effect of the optimized weights from [Fu et al. \(2018\)](#) on the compressible Taylor-Green vortex at Mach 1 on 256^3 grids for the (a) kinetic energy and (b) total dissipation rate, compared to the fine mesh solution.

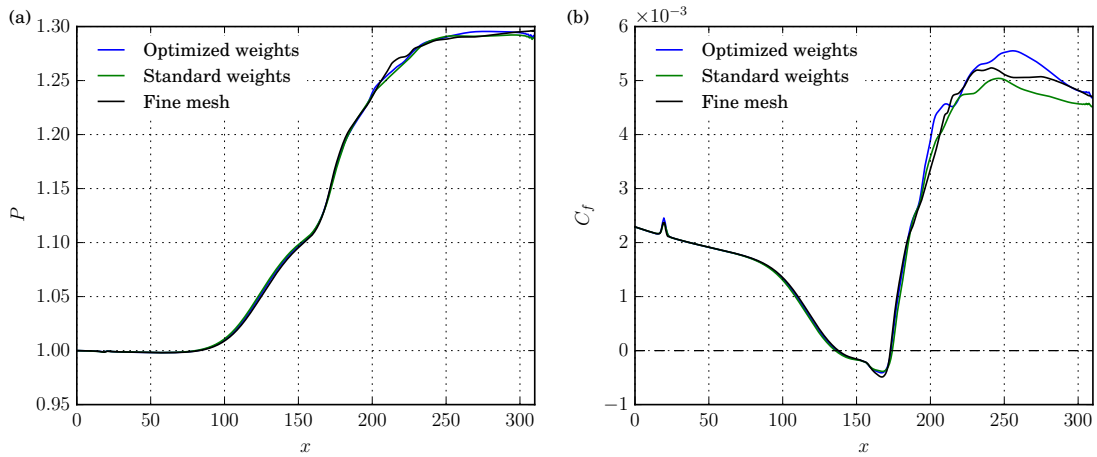


FIGURE 5.22: The effect of the optimized weights from [Fu et al. \(2018\)](#) on the transitional SBLI ILES for (a) span averaged wall pressure and (b) span averaged skin friction distribution on the wall.

$15 < t < 19$ as the flow breaks down into smaller scales. For the transitional SBLI ILES in figure 5.22, both sets of weights agree well for the span-averaged wall pressure when compared to the fine mesh solution. Skin friction through the laminar boundary layer and separation bubble are also similar, both sets of weights slightly underestimate the magnitude of the minima. The overshoot from the optimized weights on coarse grids at $x = 200$ is reduced for the standard weightings.

5.5 Discussion

A selection of low-dissipative shock capturing schemes from the ENO family have been used to simulate a compressible Taylor-Green vortex to supersonic Mach numbers and a transitional SBLI with forced upstream disturbances. The compressible Taylor-Green vortex problem places substantial demands on numerical schemes, with conventional WENO schemes proving excessively dissipative for compressible turbulence unless paired with a hybrid central scheme. Low-dissipative TENO schemes are a significant improvement over WENO for comparable computational cost, making them better suited to DNS or ILES of compressible turbulence. In particular the adaptive TENO6-A scheme of [Fu et al. \(2018\)](#) with built-in shock sensor performed well when resolving compressible turbulence on coarse grids. The schemes struggled more with a transitional flat plate SBLI on coarse grids, with some schemes giving significantly larger separation bubbles than the refined solution. It was demonstrated that schemes giving larger separation bubbles under predicted the initial disturbance which led to a delayed transition. For the schemes that replicated the initial amplitude well on the coarse mesh, good agreement was found to the refined solution. The TENO6-A ILES was able to closely match the refined-grid skin friction and wall pressure profiles despite a factor of 12 reduction in total grid points. A further coarsening in the wall normal direction showed that additional savings in computational cost are possible.

For the ILES of the compressible TGV, all of the schemes were able to produce similar shaped distributions for the kinetic energy and total dissipation rate. There was a clear and consistent ordering of the schemes. In general the low-dissipative and higher order schemes performed better, with the TENO schemes consistently less dissipative than their WENO counterparts. The aim of the transitional SBLI was to assess how the schemes performed for a more representative research problem that included non-periodic boundaries, flow separation, and a shock-induced transition to turbulence. The transition location and size of the separation bubble were found to be highly sensitive to the amplitude of the initial disturbances, with certain schemes producing much larger separation bubbles than others on the coarse mesh. Increased oscillations and undershoots were also observed at discontinuities for the TENO schemes when compared to conventional WENO. A comparison of different weight choices for the TENO6-A scheme showed that these defects were greatly reduced when not using optimized weights. Although the standard weights were slightly more dissipative for the compressible TGV and transitional SBLI, the difference was not significant enough to recommend usage of the optimized weights. While low-dissipative methods perform well for compressible turbulence with shocks on coarse grids, care must be taken with optimization of the schemes to avoid excessive oscillations near discontinuities.

Chapter 6

The effect of sidewall flow confinement on laminar shockwave/boundary-layer interactions

6.1 Introduction

In this chapter an extension of the two-dimensional laminar shock-wave/boundary-layer interaction (SBLI) from section 4.6 is presented. The work can be viewed as a three-dimensional confined extension of the [Katzner \(1989\)](#) case, based on the earlier experiments of [Hakkinen et al. \(1959\)](#). The problem is extended into the third dimension and features the addition of sidewall effects. The underlying aims of the chapter are to investigate the effect of flow confinement on laminar SBLIs, and to elucidate the resulting complex three-dimensional flow structures and shock system. As previously mentioned, the majority of numerical studies of SBLI make the approximation of an infinitely-wide span. For internally bounded flows this is not a valid assumption as lateral confinement leads to multiple boundary layers for the shock to interact with. The modified interaction may be highly three-dimensional and strongly influenced by the geometry of the duct.

The base flow configuration remains a Mach 2 inlet, with a similarity solution laminar boundary-layer initialised in the domain. The profile is blended in the corner regions at the intersection between two solid walls. While 2D and span-periodic 3D cases are able to apply exact shock-jump conditions on the upper boundary to trigger a shock-wave without issue, the sidewall case requires the initial shock to be generated by a flow deflection ramp. This requirement is to avoid the non-physical matching of a no-slip sidewall and its subsonic boundary-layer, with the free-stream supersonic Dirichlet conditions required to impose an incident shock-wave. Instead, the freestream flow is deflected by a full-span shock generator ramp that is connected to both of the sidewalls as in figure 6.1.

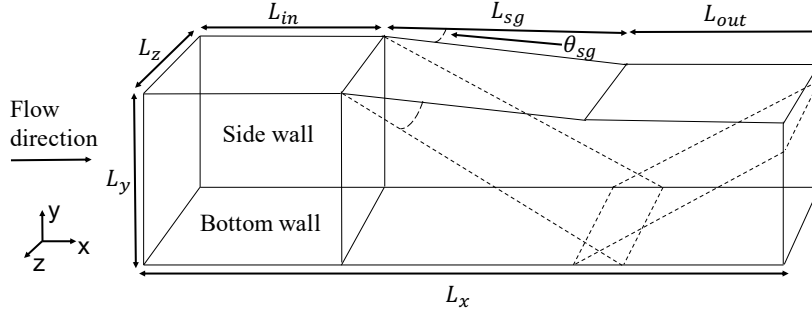


FIGURE 6.1: Schematic of the computational domain. An oblique shockwave is generated by deflecting the oncoming flow with a ramp angled at θ_{sg} to the freestream. No-slip isothermal wall conditions are enforced on the bottom wall, both sidewalls and on the upper surface between L_{sg} and L_{out} .

The chapter is organised as follows. Section 6.2 specifies the physical problem and computational domain. In section 6.2.2 a grid refinement study is performed to demonstrate grid independence of the results. Section 6.2.3 examines the effect that the shock generator length has on the central separation size for both two and three-dimensional flows. Section 6.3.1 discusses the baseline configuration, highlighting the main flow features and making comparisons to quasi-2D predictions. The topology of the laminar SBLI is shown in section 6.3.2, analysing both the global shock structures and critical points found in near-wall streamlines. Qualitative comparisons are made to previous turbulent studies to assess whether similarities can be drawn to flow structures in the laminar case. Parametric effects of duct aspect ratio and incident shock strength are given in sections 6.4.1 and 6.4.2 respectively. Section 6.4.3 uses the trailing expansion fan effect of section 6.2.3 to demonstrate its use as a flow control method.

6.2 Problem specification and duct selection

6.2.1 Domain specification and physical parameters

For span-periodic simulations of SBLI the standard method of generating an incident shock is to apply the inviscid Rankine–Hugoniot jump conditions on the upper or inlet boundary of the domain. For confined duct flows this is not valid as it creates a non-physical interface between the sidewall boundary layers and the shock jump conditions on the upper surface. In this chapter the oblique shock is generated instead by deflecting the flow with a no-slip ramp as shown in figure 6.1. Duct dimensions, aspect ratio and the length of the shock generating ramp are the primary considerations when selecting a computational domain. The domain must also be long enough in the streamwise direction to allow the central recirculation to fully develop.

The baseline case is selected to have a one-to-one aspect ratio with non-dimensional dimensions of $(L_x \times L_y \times L_z) = (550, 175, 175)$ as in table 6.1. As these are laminar simulations, a modest flow deflection of $\theta_{sg} = 2.0^\circ$ is selected for all simulations to follow [Katzer \(1989\)](#) unless otherwise stated. On the upper surface in figure 6.1, L_{in} , L_{sg} and L_{out} refer to the distance between the inlet and the shock generator, the length of the shock generator and the remaining distance to the outlet. For the baseline case the shock generator starts at $x = 45$, with $L_{sg} = 300$ and $L_{out} = 205$. For this L_{sg} the trailing edge expansion fan generated at $x = 345$ leaves through

Simulation case	Domain dimensions ($L_x \times L_y \times L_z$)	Grid distribution ($N_x \times N_y \times N_z$)
0.25AR	$550 \times 175 \times 43.75$	$750 \times 455 \times 87$
0.5AR	$550 \times 175 \times 87.5$	$750 \times 455 \times 177$
Baseline 1.0AR	$550 \times 175 \times 175$	$750 \times 455 \times 355$
2.0AR	$550 \times 175 \times 350$	$750 \times 455 \times 715$
4.0AR	$550 \times 175 \times 700$	$750 \times 455 \times 1435$

TABLE 6.1: Domain specification and grid distributions. Aspect ratio (AR) is defined as the ratio of duct width to height (L_z/L_y). A one-to-one aspect ratio is taken as the baseline configuration.

the outlet of the domain without impinging on the bottom wall. The effect of L_{sg} on the central recirculation bubble is given in section 6.2.3. The other cases in table 6.1 correspond to the aspect ratio study in section 6.4.1 for aspect ratios between one-quarter and four.

All simulations are performed at Mach 2 with a laminar boundary layer profile imposed at the inlet of the domain. The profile is obtained via the similarity solution of the compressible boundary layer equations (White, 2006) described in section 3.3.7. The simulation parameters for the laminar duct SBLI are given in table 6.2. The Reynolds number based on the displacement thickness at the start of the computational domain is $Re_{\delta^*} = 750$. For the baseline configuration a flow deflection of $\theta_{sg} = 2^\circ$ is applied by the shock generator located at $x_{sg} = 45$, giving an inviscid impingement point of $x = 328$ for the incident shock. Reynolds number based on the distance from the leading edge of the plate to the impingement point is $Re_x = 3 \times 10^5$ as in one of the cases from Katzer (1989). For the variation of incident shock strength in section 6.4.2, the location of the shock generator is shifted to maintain the same Re_x at impingement.

Quantity	Symbol	Value
Free-stream Mach number	M_∞	2.0
Reynolds number based on shock impingement	Re_x	3.0×10^5
Reynolds number based on inlet displacement thickness	$Re_{\delta_0^*}$	750
Prandtl number	Pr	0.71
Sutherland temperature	T_s	110.4 K
Reference freestream temperature	T_∞	288.0 K
Non-dimensional wall temperature	T_w	1.676

TABLE 6.2: Simulation parameters for the three-dimensional laminar SBLI cases in rectangular ducts.

On the bottom and both sidewalls of the domain a no-slip isothermal condition is applied with a constant non-dimensional temperature of $T_w = 1.676$ (4 s.f.), corresponding to the adiabatic wall temperature from the similarity solution. A zero gradient condition is applied on the upper boundary over L_{in} in figure 6.1 to maintain the freestream and sidewall boundary layers upstream of the shock generator. At x_{sg} the upper surface becomes a no-slip wall with the same isothermal condition as on the bottom and sidewalls of the domain. The no-slip condition on the upper surface is maintained until the outlet. At the inlet and outlet a pressure extrapolation and low order extrapolation method are applied, respectively, to improve stability. No boundary layer is initialised on the shock generator; it is left to develop naturally during the initial stages

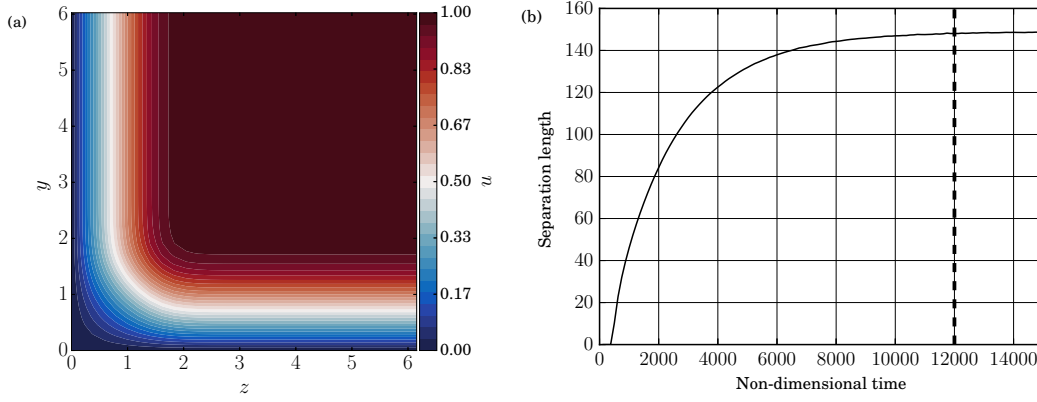


FIGURE 6.2: (a) Streamwise velocity contours of the inlet laminar boundary layer profile at the intersecting corner between two no-slip walls. (b) Convergence of the centreline separation bubble length in time. One flow-through time of the freestream is equal to $t = 550$ time units.

of the simulation. An open condition upstream of the shock generator was selected to mimic experimental configurations where the freestream is incident on a shock generator plate.

In the corner region, boundary layer profiles of equal thickness from two adjacent walls are blended together as follows. The streamwise velocity profile for each wall is multiplied by the wall normal velocity component of the adjacent wall to create a combined profile that smoothly tends to zero in the corner. For temperature, the profiles $T_1(y), T_2(z)$ are scaled for a constant wall temperature T_w such that

$$T_1 = \frac{T - T_w}{T_\infty - T_w} \quad \text{and} \quad T_2 = \frac{T - T_w}{T_\infty - T_w}, \quad (6.1)$$

to give $T_1, T_2 \in [0, 1]$. The scaled profiles for two intersecting walls are then blended together by

$$T(y, z) = T_w + T_1(y)T_2(z)(1 - T_w), \quad (6.2)$$

giving a smooth profile that varies from $T = 1$ in the freestream to $T = T_w$ at the wall. The wall normal velocity component w from each of the sidewalls is of equal magnitude but opposite direction, requiring it to be damped with the z coordinate in both directions to create a zero w component of velocity on the centreline. Figure 6.2 (a) shows the resulting profile that is imposed on the inlet; the normalised laminar flow is seen to vary smoothly from zero at the walls to one in the freestream.

As highlighted by [Sansica et al. \(2013\)](#), laminar separation bubbles require long time integration to fully develop and previous studies have often reported shorter lengths from non-converged simulations. To verify the simulations were sufficiently converged for this work the evolution of centreline separation length is presented in figure 6.2 (b). After impinging on the bottom wall boundary layer, the incident shock rapidly creates a region of flow-reversal during the early stages of the simulation. With increasing time the separation length converges and a stable separation bubble length is observed. For all simulations in this work the convergence time is taken to be $t = 12000$ (≈ 22 flow-through times of the domain), denoted by the vertical dashed line in figure 6.2(b). Integrating the simulation for a further ≈ 5.5 flow-through times up to

$t = 15000$ only resulted in a 0.3% change in separation length. Having defined the domain and physical parameters for the simulations, the next section demonstrates the grid independence of the solution.

6.2.2 Numerical method and sensitivity to grid refinement

All of the simulations of laminar duct SBLI in this chapter were performed using the 5th order WENO-Z scheme described in section 2.3. As these are all laminar simulations, the low-dissipative TENO schemes are not required. The WENO schemes give reduced oscillations near shocks, generating smooth laminar base-flows for future stability analysis. Based on initial exploratory simulations, a starting grid resolution of $(N_x, N_y, N_z) = (700, 295, 295)$ was selected to perform the grid refinement study and investigate the effect of shock generator length. Insensitivity to grid refinement was assessed by increasing the number of grid points by 50% in each spatial direction independently. Grid stretching is performed symmetrically in the y and z directions to cluster points in the boundary layers of each wall, with a uniform distribution in x . Grid points in y and z are distributed with a stretch factor $s = 1.3$ as

$$y = \frac{1}{2}L_y \frac{1 - \tanh(s(1 - 2\xi))}{\tanh(s)}, \quad z = \frac{1}{2}L_z \frac{1 - \tanh(s(1 - 2\xi))}{\tanh(s)}, \quad (6.3)$$

for uniformly distributed points $\xi = [0, 1]$. Figure 6.3 (a) and (b) show the effect of increased grid resolution for the centreline wall pressure and skin friction respectively. For the baseline $\theta_{sg} = 2^\circ$ case with one-to-one aspect ratio the shock induced pressure rise normalised by the inlet is $p_3/p_1 = 1.31$. There is minimal discrepancy between each of the simulations and the centreline pressure is insensitive to further grid refinement. A similar picture is seen for the skin friction in figure 6.3 (b), all grids produce the expected asymmetric twin trough shape of a laminar separation bubble. A small deviation is seen downstream of the reattachment point in the case of streamwise grid refinement.

The separation bubble length is the streamwise extent of flow-reversal, defined as the distance between the two zero crossings of the skin friction curve in figure 6.3 (b). The crossing locations are found by an interpolation procedure on the discrete grid. The separation length is insensitive to grid refinement; the largest variation occurred for the ‘FineZ’ case which was 1% larger than the coarse grid. There is also a slight discrepancy at the outlet in the ‘FineX’ case. Based on these results and to improve resolution on the shock generator a refined grid of $(N_x, N_y, N_z) = (750, 455, 355)$ was selected for the default one-to-one aspect ratio cases in this paper. Parametric studies of aspect ratio in section 6.4.1 use the grids outlined in table 6.1.

6.2.3 The role of the shock-generator and trailing expansion fan

During the selection of the computational domain it became apparent that, in addition to the aspect ratio and dimensions of the domain, the length of the shock generator is an important factor. This influence is most significant when considering laminar SBLI as the separation regions are considerably larger than in the presence of turbulence and are therefore more likely to be crossed by expansion fans emitted from the trailing edge of the shock generator. To quantify

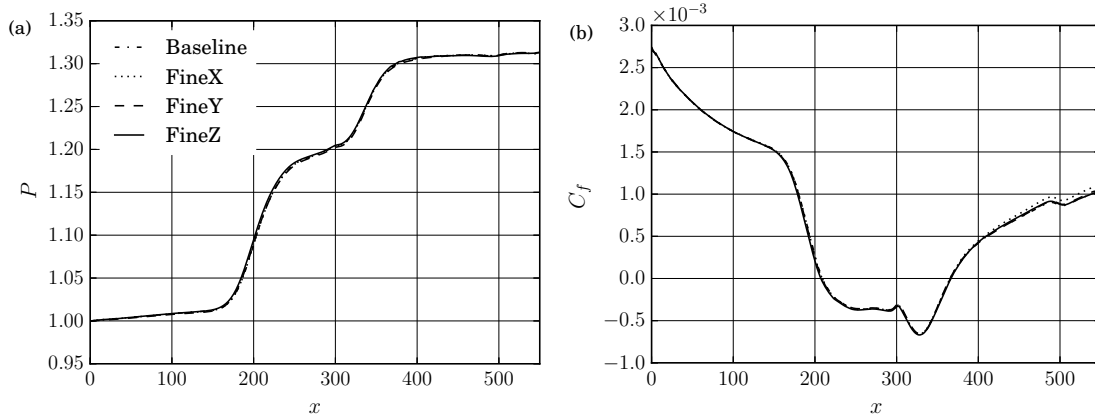


FIGURE 6.3: Sensitivity of the centreline (a) wall pressure and (b) skin friction to grid refinement for $AR = 1$. In each direction 50% additional grid points are added independently.

Shock generator length (L_{sg})	Interaction region (x_{start}, x_{end})	L_{sep}	Increase in L_{sep} (%)
200	(251.4, 369.5)	118.09	—
250	(251.1, 370.4)	119.32	1%
300	(251.0, 370.5)	119.52	1%
350	(251.0, 370.5)	119.55	1%

TABLE 6.3: Sensitivity of the centreline separation to increasing L_{sg} for two-dimensional SBLI without sidewalls. Increasing L_{sg} causes the trailing edge expansion fan to impinge further downstream on the bottom wall. Percentage increase is relative to the shortest $L_{sg} = 200$ case.

this effect, a selection of shock generator lengths are reported in this section for 2D and 3D simulations, using the baseline grid from the previous section. The comparison of 2D to 3D is useful because it illustrates the role of the sidewalls when considering the shock generator length.

Four shock generator lengths in the range $L_{sg} \in [200, 350]$ are considered, which correspond to (36 – 64)% of the streamwise domain length. The lengths were chosen to ensure that the trailing expansion fan did not impinge directly on the separation bubble, but were close enough to ascertain the downstream influence on the main interaction region. As these are all laminar interactions, 2D simulations are equivalent to a 3D simulation with span-periodic boundary conditions. The 3D simulations include the effect of sidewalls and so a deviation from the quasi-2D results in this section can only be attributed to 3D effects resulting from physical flow confinement in the duct.

Figures 6.4 (a) and (b) show the centreline wall pressure and skin friction for the 2D simulations as the shock generator length is varied. For the shortest two shock generator lengths an expansion fan impinges on the bottom wall of the domain downstream of the reattachment point. Although there is a significant decrease in pressure and increase in skin-friction near the outlet, the separation bubble is largely unchanged by this downstream influence. Table 6.3 quantifies the effect the shock generator length has on separation for quasi-2D interactions. The shortest two shock generators agree to within 1% of each other and further increases in shock generator length have no significant influence on the separation bubble.

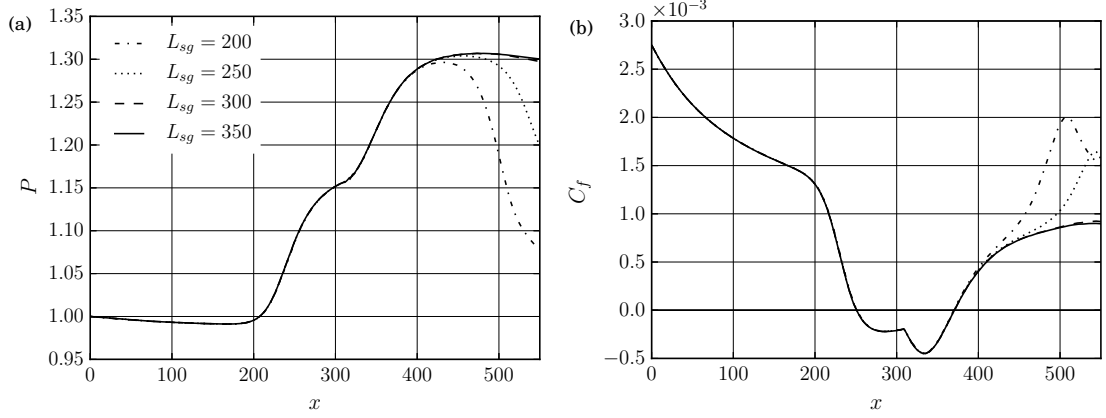


FIGURE 6.4: Sensitivity of the 2D simulation (a) wall pressure and (b) skin friction to shock generator length.

Shock generator length (L_{sg})	Interaction region (x_{start}, x_{end})	L_{sep}	Increase in L_{sep} (%)
200	(220.0, 351.9)	131.89	—
250	(213.5, 359.8)	146.35	11%
300	(209.0, 365.8)	156.77	19%
350	(205.2, 371.2)	166.05	26%

TABLE 6.4: Sensitivity of the centreline separation to increasing L_{sg} for three-dimensional SBLI at $AR = 1$ with sidewall effects. Increasing L_{sg} causes the trailing edge expansion fan to impinge further downstream on the bottom wall and also modifies the pressure distribution downstream of the interaction. Percentage increase is given relative to the shortest $L_{sg} = 200$ case.

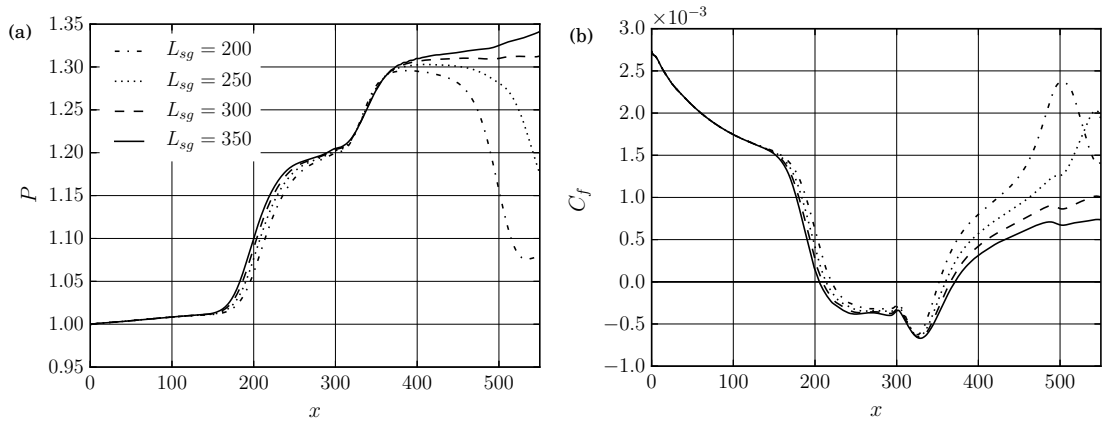


FIGURE 6.5: Sensitivity of the 3D simulation with sidewalls at $AR = 1$ for the (a) wall pressure and (b) skin friction to shock generator length.

3D results with sidewall effects are shown in figures 6.5 (a) and (b) for the same range of shock generator lengths as in the 2D cases but with $AR = 1$. In contrast to figure 6.4 (b) the skin friction distribution of figure 6.5 (b) shows a clear influence of the trailing expansion fan on the main interaction. In addition to the previously seen skin friction rise at the outlet, the central separation bubble has been shortened significantly in the 3D case for the shorter shock generator lengths. When the sidewall influence is included, the separation and reattachment locations of the separation bubble are both modified. A similar pattern is seen in figure 6.5 (a), where the initial pressure rise at the point of separation is delayed downstream for shorter shock generators.

Table 6.4 gives the size of the interaction region and increases in separation length for the 3D cases. As the length of the shock generator is increased from $L_{sg} = 200$ the separation and reattachment locations shift upstream and downstream respectively. This leads to (11 – 26)% increases in overall separation length compared to the shortest shock generator. Importantly we see there is an increase in separation length even between $L_{sg} = 300$ and $L_{sg} = 350$, where in both cases the trailing expansion fan is leaving the computational domain before impinging on the bottom wall. As the largest two shock generators disagree with each other despite the expansion fans not directly hitting the bottom wall, the discrepancy can only be attributed to 3D effects of the trailing expansion fan on the sidewall flow and its subsequent influence on the central separation. Experimentally this effect has been observed for a turbulent case by [Grossman and Bruce \(2017\)](#), in which the physical length of the shock generator was varied to move the location of the expansion fan. The authors noted that as the expansion fan is moved downstream, there is an increase in the strength and streamwise length of the central separation accommodated by an upstream shift in the separation point. Despite the differences in incident shock strength and boundary layer state to the present work, their findings are consistent with those of figure 6.5 (b). The main difference to this work is that in the current laminar case a downstream shift of the reattachment location is observed while remaining largely independent of the expansion fan location in [Grossman and Bruce \(2017\)](#).

Having quantified the role of the shock generator length for the 3D simulations, we select a domain with $L_{sg} = 300$ as the default configuration for all of the following simulations unless otherwise stated. It must be emphasised that this is a design choice of the duct and differences in the separation length would occur for different configurations. Including three-dimensional flow confinement into the problem increases the complexity of the flow field and naturally adds a dependence of the duct aspect ratio, domain dimensions, and shock generator length to any reported results. This is in contrast to quasi-2D simulations where the SBLI depends only on the incident shock strength and incoming boundary layer state. The effect of the trailing-edge expansion fan on the central interaction is investigated further in section 6.4.3 for a longer domain with a considerably longer shock generator ramp.

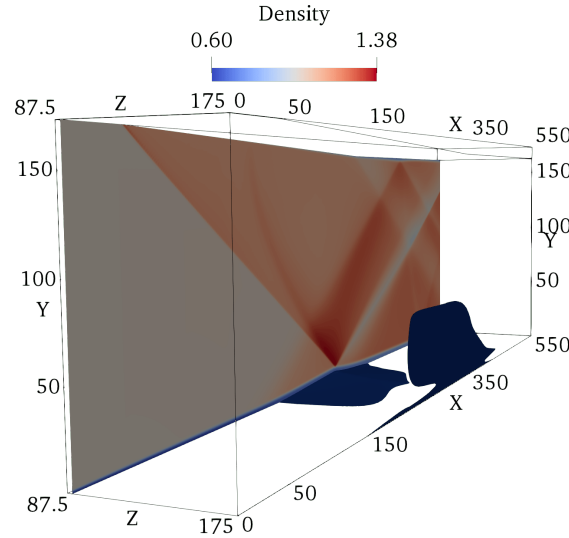


FIGURE 6.6: Baseline duct SBLI density contours ($AR = 1$). Displaying a centreline density slice ($z = 87.5$) with regions of reverse flow ($u \leq 0$) on the bottom and sidewall highlighted in dark blue.

6.3 Three-dimensional rectangular laminar duct SBLI with sidewall effects

6.3.1 Baseline duct configuration

Figure 6.6 shows density contours for the laminar base flow obtained for the default configuration of aspect ratio one and $\theta_{sg} = 2^\circ$. The regions of flow-reversal are highlighted in dark blue on the sidewall and in the centre. Despite the relatively weak initial shock, large regions of reverse flow develop in the corners and on the bottom and side walls of the domain. This is in contrast to turbulent SBLI such as Wang et al. (2015), where the greatly enhanced mixing rates in the boundary layer help prevent flow separation on the sidewalls. A slice of density along the centreline shows that the separation bubble extends far upstream of the impingement point, with a series of compression waves emitted from the start of the bubble due to a thickening of the boundary layer. For the laminar base flow the features are symmetric about the centreline ($z = 87.5$), with each sidewall containing a large region of reverse flow. A long thin corner separation is seen that extends further upstream than both the central and sidewall separations. Between the sidewall and central separations is a distinct region of attached flow where the initial shock has been weakened by the sidewall influence. At the trailing edge of the shock generator an expansion fan can be seen crossing the reflected shock and leaving the computational domain. The reflected shock creates a secondary separation bubble on the upper wall of the domain before passing through the outlet.

Figure 6.7 (a) compares the bottom-wall centreline skin friction at $y = 0$ for the duct with and without a shock generator. It can be seen that the reattached flow downstream of the SBLI recovers to match the laminar boundary layer near the outlet. A further comparison is made to a span-periodic case to demonstrate the effect that sidewall confinement has on the central flow. The strengthening of the incident shock from the sidewalls leads to an increase in central separation length of 31.5%. It is again emphasised that as in Table 6.4, this percentage increase is

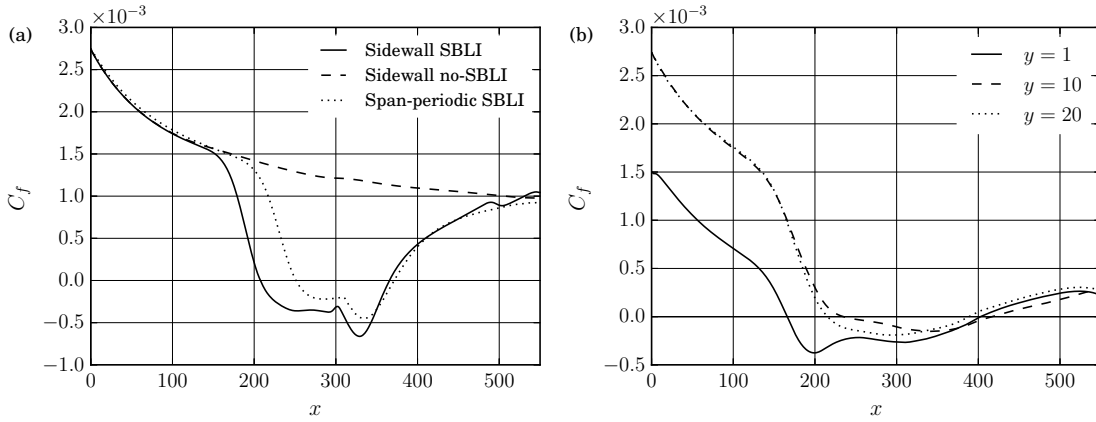


FIGURE 6.7: (a) Centreline skin friction on the bottom wall ($y=0$) for a duct with and without SBLI at $AR=1$, compared to a case without sidewalls. (b) Skin friction relative to the sidewall ($z=0$) for the duct SBLI at various y heights, showing the early streamwise onset of the corner separation.

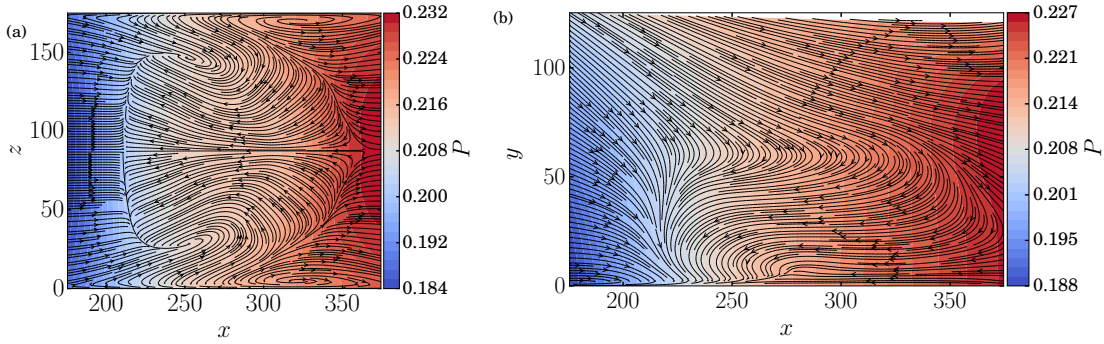


FIGURE 6.8: Streamline patterns coloured by the shock-induced pressure jump of the main interaction for $AR=1$. Displaying (a) $u-w$ streamlines above the bottom wall at $y=0.14$ and (b) $u-v$ streamlines above the sidewall at $z=0.14$.

highly dependent on the shock generator length and subsequent position of the trailing expansion fan.

Separation and reattachment locations (x_{start}, x_{end}) are found at $x = (251.2, 371.4)$ and $x = (207.7, 365.6)$ for the span-periodic and duct SBLI respectively. Although the reattachment locations are similar, the separation point has moved upstream substantially due to the sidewall influence. The early onset of the corner separation relative to the sidewall separation can be seen in figure 6.7 (b). Skin-friction relative to the sidewall (2.9) is shown at three different y locations on the $z=0$ side of the domain. Within the corner boundary layer at $y=1$ the flow first detaches at $x=166.1$, at which point the centreline and sidewall boundary layers are still attached. The skin friction distributions at $y=10$ and $y=20$ agree well up until the point of separation, occurring at $x=215.3$ and $x=236.5$ respectively. The strongest flow-reversal on the sidewall occurs early within the corner region, as seen in the trough around $x=200$. From this we conclude that the corner regions of the duct are most susceptible to shock-induced separation, as there is a build up of low-momentum fluid being simultaneously retarded by no-slip walls in two directions.

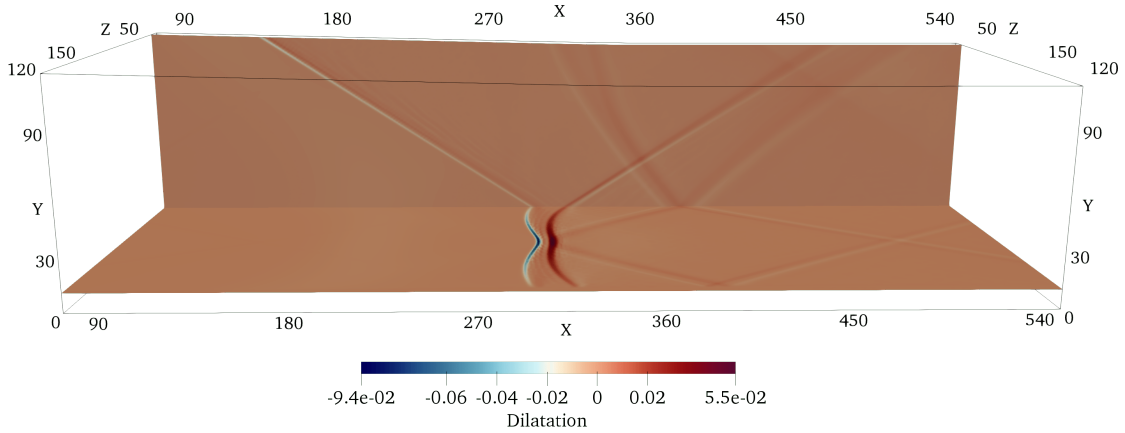


FIGURE 6.9: 2D slices of dilatation ($\nabla \cdot \bar{u}$) plotted at $y = 10$ and $z = 10$ for the $AR = 1$ baseline case. The influence of the sidewalls leads to a curving of the incident shock. Negative dilatation shown in blue corresponds to the strongest regions of the incident shock. A pair of transverse shockwaves are seen to emanate from the interaction region and reflect off the sidewalls.

To further elucidate the regions of flow-reversal in figure 6.6, velocity streamline patterns are shown in the near-wall region in figure 6.8 for (a) the bottom wall and (b) the sidewall of the domain. A more formal analysis of flow topology is given in the next section after the main recirculation zones are highlighted here. The structure of the central recirculation is clear to see by noting the direction of the streamlines; flow is ejected from each corner and towards the centreline where it travels upstream. Streamlines diverge at the separation (blue) and reattachment (red) regions of the interaction and recirculation is also visible in each of the corners. In between the central and corner separations the attached flow region in figure 6.6 is seen as the region where velocity streamlines diverge away from the attachment line and continue downstream. Figure 6.8 (b) shows the down-wash of fluid near the sidewall as a result of the swept SBLI. Streamlines from all directions are directed into a nodal point at $x = 220$ with an accompanying focal point similar to the type 1 separation of [Tobak and Peake \(1982\)](#). Flow-reversal dominates a large portion of the sidewall and extends to almost 50% of the duct height.

Finally we draw attention to the curved shape of the incident and reflected shocks shown by the dilatation plot ($\nabla \cdot \bar{u}$) in figure 6.9. Two orthogonal intersecting slices are plotted at $y = 10$ and $z = 10$, with negative (blue) and positive (red) dilatation representing the incident shock and expansion fan above the bubble respectively. There is a curving of the shock, that is consistent with figure 10 (a) of [Wang et al. \(2015\)](#), despite the differences in flow conditions and boundary layer state. The shock deforms downstream at the centreline, which is also seen to be the strongest part of the incident shock. Away from the centre the incident shock decreases in strength, consistent with the conditions needed to produce regions of attached flow bordering the central recirculation.

6.3.2 Topology of the interaction

To understand three-dimensional SBLI, it is important to look at both global shock structures and the topological features visible in near-wall streamline traces. Experimental streamline patterns are typically obtained via an injection of an oil mixture upstream of the interaction,

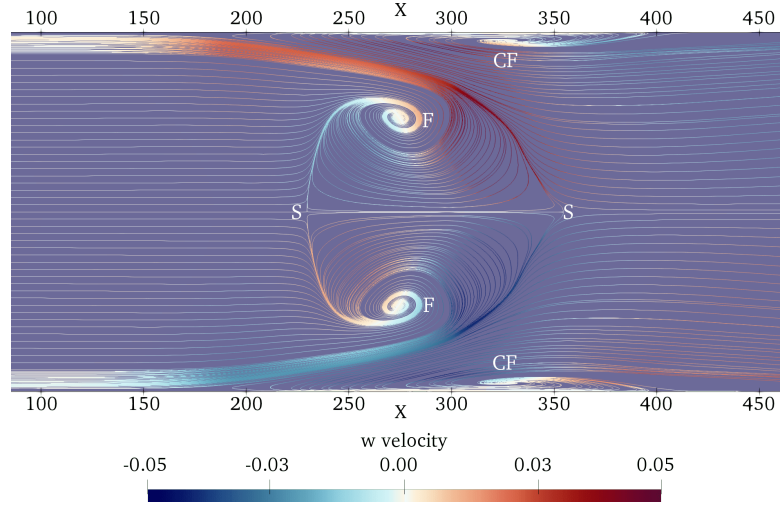


FIGURE 6.10: Streamlines evaluated in the x - z plane at $y = 1$ for the baseline $\theta_{sg} = 2.0^\circ$ case. Streamlines are coloured by the transverse velocity component w with a constant colour background. The flow diverges at saddle points (S) at the front and back of the main recirculation region. The SBLI generates strong transverse velocity gradients that cause an ejection of the corner flow towards the centreline. Streamlines within the separation bubble are directed into two foci (F) that are symmetric relative to the centreline. Two additional foci (CF) required for topological consistency are labelled in each corner region.

which gives an imprint of the mean flow on the walls of the test chamber. Examples of oil injection include figure 11 of [Grossman and Bruce \(2018\)](#), where the oil injection points are clearly visible upstream of the central SBLI. In addition to the potential for oil injection to cause undesirable modification of the flow, care must also be taken to avoid imprints of the transient behaviour during wind tunnel start-up/shutdown. Modern experimental techniques such as stereo-PIV can capture velocity data in three dimensions, allowing for the construction of limiting streamline patterns ([Eagle and Driscoll, 2014](#)). Among the benefits of numerical work is the access to full three-dimensional time-dependent flow data which can supplement observations made experimentally.

Critical point analysis is a useful tool for identifying three-dimensional separations from streamline patterns. Critical points occur where skin friction lines terminate on a surface or, equally, where the magnitude of two-dimensional skin friction vectors becomes zero. Points are classified into either nodes or saddle points, with further subdivisions of nodes into nodal points and foci of either attachment or separation depending on the direction of the streamlines ([Tobak and Peake, 1982](#)). While usually described in the context of skin friction, the same analysis can be performed on streamlines obtained from velocity fields ([Babinsky and Harvey, 2011](#)). Attachment nodes (N) are classified as the source of streamlines emerging from an object and separation nodes are found where they terminate. A focus (F) is a point about which streamlines spiral around and ultimately terminate. Saddle points (S) are defined as singular locations at which only two streamlines enter, one inwards and one outwards. All other streamlines diverge away from a saddle point hyperbolically, separating the streamlines that emerge from adjacent nodes. A two-dimensional separation bubble is characterised by a streamline that lifts off a surface at a separation point and reattaches at a point downstream of the bubble. Within the bubble, closed streamlines circulate around a single common point and do not escape to the outer flow. This description is incompatible with three-dimensional separations where streamlines instead have a

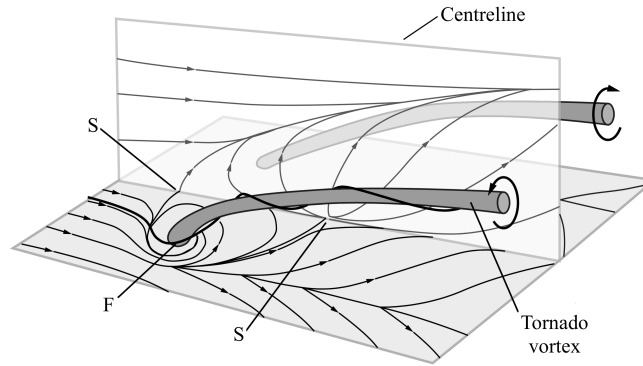


FIGURE 6.11: Schematic of the ‘owl-like’ separation of the first kind adapted from Colliss et al. (2016), based on the work of Perry and Hornung (1984). The front saddle point (S) acts as a separating line at the start of the recirculation bubble. A focus (F) either side of the centreline signifies a tornado-like vortex that lifts fluid away from the surface. The description is consistent with the results presented in figure 6.10 for the baseline case.

decaying orbit around a focus point that terminates them. In three dimensions the criteria for identifying flow separation can be defined as streamline patterns that contain at least one saddle point (Détery, 2001). At a focus fluid escapes laterally and signals the presence of a tornado-like vortex (Babinsky and Harvey, 2011). A vortex above a surface acts to lift fluid entering the focus upwards and transfer it downstream to the outer flow. In this sense three-dimensional separations are denoted ‘open’ separations as flow attaching downstream of the interaction is distinct from that which separated previously (Eagle and Driscoll, 2014).

Figure 6.10 shows (u, w) velocity streamlines evaluated near the bottom wall at $y = 1$. Streamlines are coloured by the transverse velocity component w over a constant colour background. Additional streamlines are added in the corner regions to demonstrate the ejection of the corner flow into the central separation. Streamlines are deflected at $x = 150$ as the corner profile thickens, with strong transverse velocity directing the flow towards the centreline on either side. At $x = 300$ the streamlines diverge between the saddle point (S) on the reattachment line and the focus (F) within the separation. Flow ejected from the corner spirals into the tornado vortex at each focus, is lifted up from the surface and transported downstream. At the front of the bubble a well defined saddle point (S) is observed; a single streamline is seen entering the saddle point laterally along the separation line from both sides indicating the presence of a surface lifting off the wall. Streamlines adjacent to the separating line are deflected hyperbolically into one of the foci. The pattern agrees well with ‘owl-like’ separations of the first kind introduced by Perry and Hornung (1984) as shown in figure 6.11. There is a noticeable bulge in the reattachment line as the saddle point is shifted downstream at the centre of the span. The shift of the saddle point was less pronounced for the weaker interactions simulated in section 6.4.2, which were observed to have a reattachment line approximately perpendicular to the downstream flow. We also note the presence of two additional foci located in the near-wall corner region denoted as CF in figure 6.10. These satisfy the topological rule that, for a given surface, the number of nodes (nodal points or foci) must exceed the number of saddle points by two (Tobak and Peake, 1982). Downstream of the central circulation the attached flow follows a smooth laminar profile with streamlines remaining mostly parallel to each other.

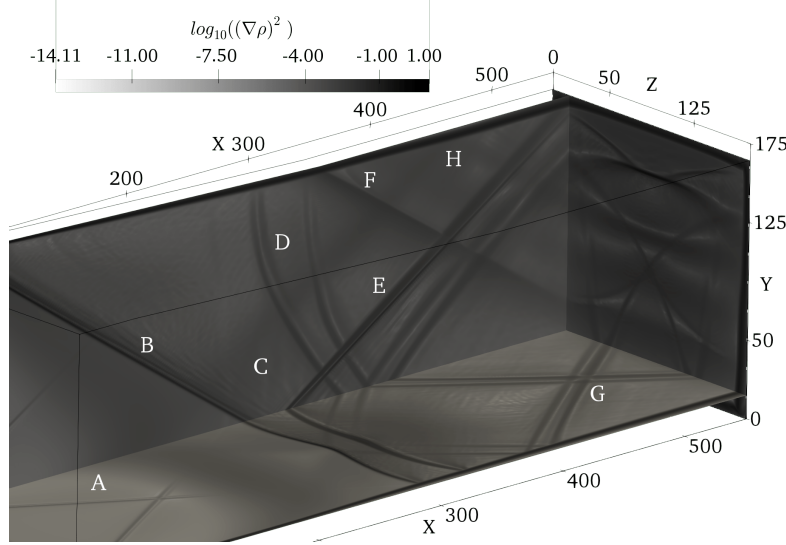


FIGURE 6.12: Numerical schlieren of density gradient $\log_{10}((\nabla\rho)^2)$ showing the complex shock structure downstream of the three-dimensional SBLI at $AR = 1$. Three intersecting slices are shown at $x = 550$, $y = 15$ and $z = 15$. Notable features include: (A) Compression waves from the initial sidewall boundary layer development. (B) Main incident shock. (C) Compression waves from the start of the central separation. (D) Two conical shocks from the corner of the shock generator. (E) Expansion fan formed from the reflection of the incident shock. (F) Trailing edge expansion fan. (G) First crossing point of the reflected conical shocks. (H) Secondary reflection of the central compression waves.

Computing the logarithm of density gradient magnitudes $\log_{10}((\nabla\rho)^2)$ is an effective way of numerically detecting shock structures, providing a more sensitive version of the schlieren photography techniques found in experiments. Figure 6.12 shows a numerical schlieren of three intersecting orthogonal slices evaluated at $x = 550$, $y = 15$ and $z = 15$. The main shock structures are identified as follows: (A) Weak compression waves from the initial development of the imposed boundary layer profile, coalescing into weak intersecting shocks. (B) The trace of the incident swept shock through the $z = 15$ plane with curvature visible at $y = 15$. (C) Reflected compression wave caused by the thickened boundary layer at the base of the SBLI. (D) Reflections of the conical shocks generated at the corner of the sidewalls and shock generator ramp, discussed in more detail later in this section. (E) Expansion fan developing as the flow turns over the apex of the recirculation bubble. (F) The trailing expansion fan generated at $x = x_{sg} + L_{sg}$. (G) Crossing of the conical shocks after reflection from the sidewalls. The crossing point is the visible kink at $x = 500$ in the C_f plot of figure 6.7 (a) (solid line). (H) A secondary reflection of the compression wave (C) as it reaches the upper boundary layer.

It is clear that the numerical schlieren is a more sensitive metric for detecting shock structures than the dilatation of figure 6.9. Although there is a subtle shading upstream of the incident shock in the y plane, we do not identify significant corner shocks as suggested by [Xiang and Babinsky \(2019\)](#) for a stronger turbulent interaction. This may be due to the turbulent boundary layer in the experiments, but we note that such corner shocks were also not clearly visible in [Wang et al. \(2015\)](#) either. We highlight that even the compression waves resulting from the streamline curvature caused by boundary layer development at (A) are more prominent than the corner compression. By far the strongest structures seen downstream of the SBLI are the reflecting conical shocks of (D) and (G), consistent with the results for the turbulent case of [Wang et al. \(2015\)](#).

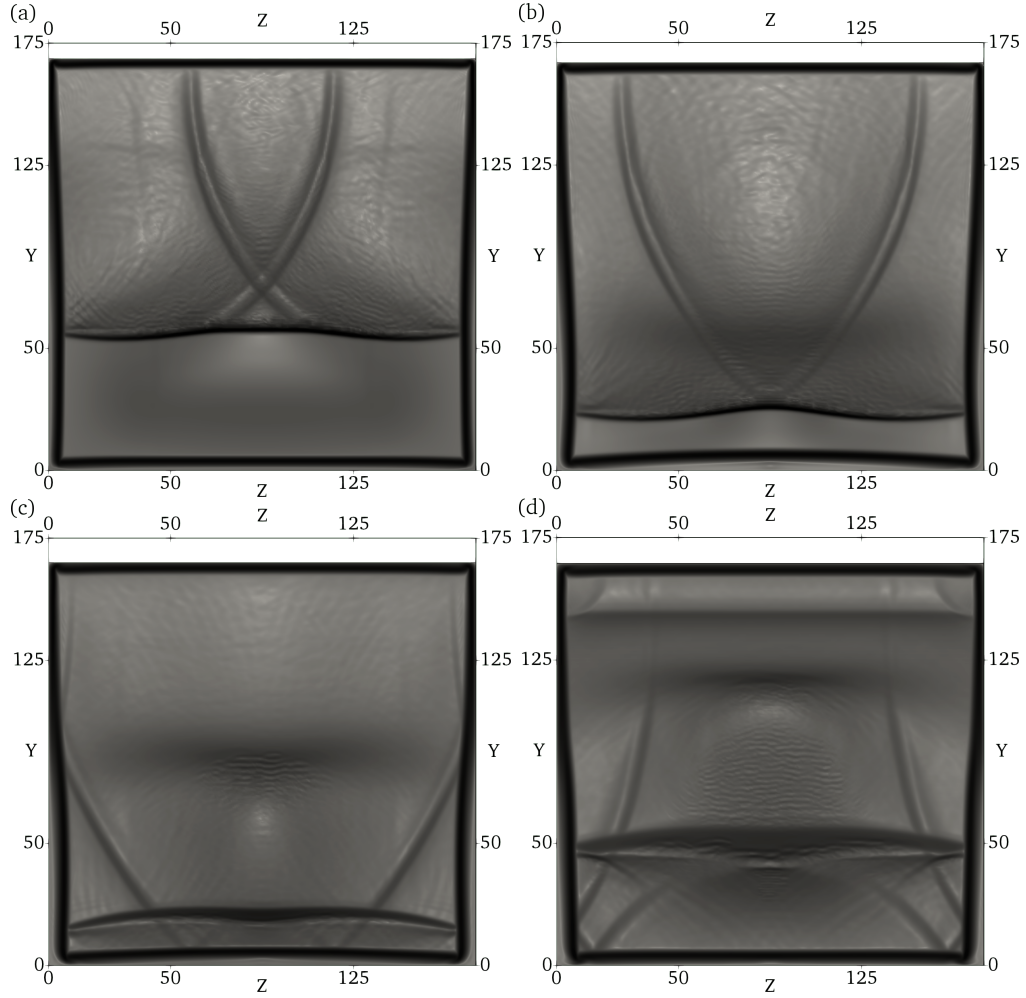


FIGURE 6.13: Numerical schlieren density gradient $\log_{10}((\nabla\rho)^2)$ showing the streamwise development of the conical swept SBLI. $y-z$ slices are displayed at streamwise locations (a) $x = 225$ (b) $x = 275$ (c) $x = 325$ and (d) $x = 375$. Two conical shocks generated by the initial swept SBLI in the upper left and right corner cross through each other in (a) and (b), before reflecting off the bottom wall and opposite sidewall in (c) and (d). The dark horizontal line in (a) and (b) is the main incident shock, while in (c) and (d) it is the expansion after the interaction. The start of the trailing edge expansion fan can also be seen in the upper region of (d).

The role of the reflected conical shocks (D) is clearer to see when looking at the $y-z$ slices at different x locations in figure 6.13. The slices show the same numerical schlieren as before, this time located at $x = (225, 275, 325, 375)$ in (a)-(d). The conical shocks are generated in the upper left and right corners of the plane at the intersections between the shock generator ramp and the two sidewalls. Stepping forward in x along the duct, the conical shocks expand outwards from their starting location. In figure 6.13 (a) and (b) the conical shocks are seen to intersect each other and continue towards the opposite sidewall. The thin dark line is the $y-z$ projection of the incident shock which is attached to the conical shock fronts. At the edges of (a) and (b) we see that the boundary layer has thickened as a result of the swept SBLI. Figures 6.13 (c) and (d) show the impact of the conical shocks on the opposite wall to that which they originated as well as a dark horizontal line marking the expansion above the apex of the main separation bubble. The upper and lower portions of the shock front reflect first in (c) and can be seen to propagate back towards their starting sidewall in (d). Comparing these slices again to the

Aspect ratio (W/H)	p_3/p_1	Interaction ($x_{sep}, x_{reattach}$)	L_{sep}	% of baseline L_{sep}	L_f
1/4	1.439	(167.5, 199.9)	32.43	21%	-
1/2	1.325	(185.0, 301.8)	116.80	74%	28.0
1	1.313	(207.7, 365.6)	157.93	-	42.1
2	1.306	(231.8, 388.2)	156.42	99%	45.3
4	1.321	(243.6, 375.8)	132.15	84%	49.0
Span-periodic	1.300	(251.2, 371.4)	120.12	76%	-

TABLE 6.5: The effect of aspect ratio on the baseline $\theta_{sg} = 2^\circ$ shock generator case. Comparison is also made to a span-periodic simulation without sidewalls demonstrating the strengthened three-dimensional interaction. Separation length is shown as a percentage of the one-to-one aspect ratio sidewall case. L_{sep} is the separation length along the centreline and L_f is the distance between the foci and the sidewalls.

$x = 550$ end slice of figure 6.12 we can see that the conical shocks trace a path across the span and reach their original sidewall near the outlet. The slice in figure 6.13 (d) also shows the start of the trailing edge expansion fan generated on the upper surface at $x_{sg} + L_{sg} = 345$.

An extensive search was performed using various forms of density gradients, pressure, and dilatation, but no obvious corner compressions were observed near the bottom of the duct. It is possible that there are fundamental differences between simulation and experiment causing the discrepancy, or that the laminar interaction is simply too weak to generate strong corner compressions. Many experimental configurations also feature gaps between the sidewall and the shock generator, which would lead to a weakened swept SBLI effect compared to enclosed ducts. For the present work the dominant structures crossing the centreline originate from the conical shocks generated between the upper ramp and sidewalls.

6.4 Parametric sensitivity

6.4.1 The effect of duct aspect ratio

The aim of this section is to determine how laminar SBLI are affected by a varying degree of flow confinement for the selection of narrow and wide duct configurations given in table 6.1. For each case the upstream flow conditions and shock strength are held constant to the $\theta_{sg} = 2^\circ$ one-to-one aspect ratio baseline case in the previous section. Aspect ratios ranging from one-quarter to four are considered, with the largest aspect ratio expected to show strong two-dimensional behaviour on the centreline. By comparing the span-periodic result to the larger aspect ratios, an estimation can be made of how wide a duct must be before span-periodicity is a valid assumption. The narrow aspect ratio cases will assess whether the observed strengthening of the SBLI due to sidewalls continues in the presence of even stronger flow confinement.

Figure 6.14 shows the centreline skin friction along the bottom wall for (a) narrow and (b) wide aspect ratios. In each of the two plots the solid line represents the one-to-one aspect ratio case from figure 6.7 (a). For the narrow ducts in figure 6.14 (a) a severe reduction in central separation length is seen and the separation point has noticeably been shifted upstream. For an aspect ratio of $AR = 0.25$ the flow is almost entirely attached; the separation bubble from the

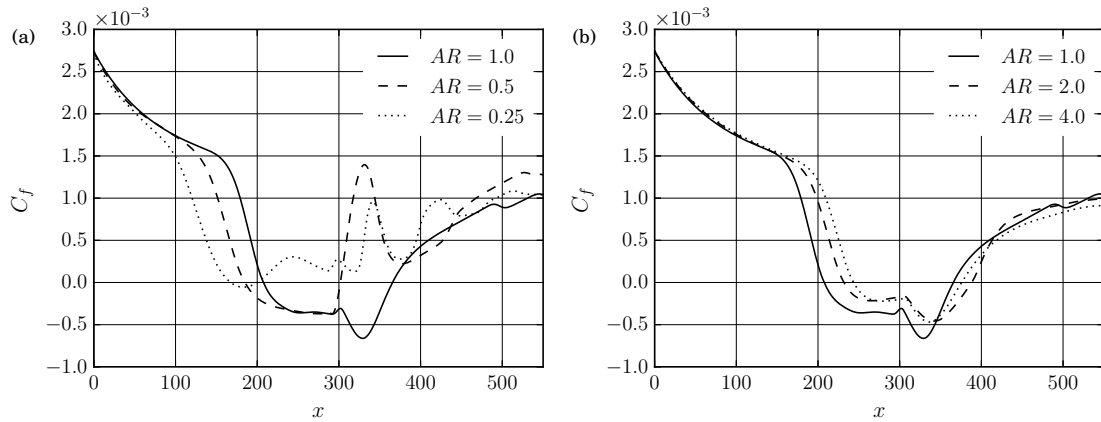


FIGURE 6.14: The effect of varying aspect ratio on the centreline bottom wall skin friction in the case of (a) narrowing and (b) widening aspect ratios. In each case the skin friction is compared to the one-to-one aspect ratio baseline duct (solid line).

initial SBLI covers only 5% of the streamwise duct length. At $AR = 0.5$ the expected asymmetric double-trough profile of a laminar separation bubble (Katzner, 1989) is also not seen since the flow abruptly reattaches at the back of the separation bubble. The most notable features of the narrower aspect ratios are the multiple peaks in skin friction downstream of the initial interaction. These correspond to the successive crossings of the incident swept-interaction that was highlighted in figure 6.12. As the width of the duct has been reduced, the waves reflecting between the sidewalls have less distance to travel and cross the centreline multiple times before reaching the outlet. Rather than a further strengthening of the interaction, aspect ratios below unity are seen to suppress the interaction and exhibit shock-trains that traverse the span of the domain.

More regular behaviour is found for the larger aspect ratios in figure 6.14 (b), in which the three-dimensionality of the sidewall flow has less of an impact on the centreline dynamics. Consistent with the narrower cases, an increase in aspect ratio causes a downstream shift of the separation point. The same is not true for the reattachment location however, with the widest $AR = 4$ case reattaching before $AR = 2$. The kink in C_f at $x = 500$ for $AR = 1$, due to crossing of the sidewall reflections is not seen for larger aspect ratios, the wider span results in the crossing occurring downstream of the computational domain. Table 6.5 shows the strong dependence of aspect ratio on the centreline SBLI. The third column gives the separation and reattachment locations for each of the aspect ratios.

As seen in figure 6.14 the downstream shift of the separation location is a consistent trend each time the aspect ratio is widened, with the largest $AR = 4$ case still farther downstream than for an infinite span. For the $AR = 1$ and $AR = 2$ cases the centreline L_{sep} differs by only 1%, despite the downstream shift of the interaction at the higher aspect ratio. At the two smallest aspect ratios the separation length was reduced to 21% and 74% of the $AR = 1$ result. Above $AR = 2$ the bubble decreases in size as three-dimensional effects become less important to the central flow. Figure 6.15 (a) shows a comparison of the widest duct with sidewalls to an infinite span. The shape of the skin friction distribution agrees well at an aspect ratio of four, although L_{sep} is still 9% longer than for the idealised infinite span. The extent to which the corner

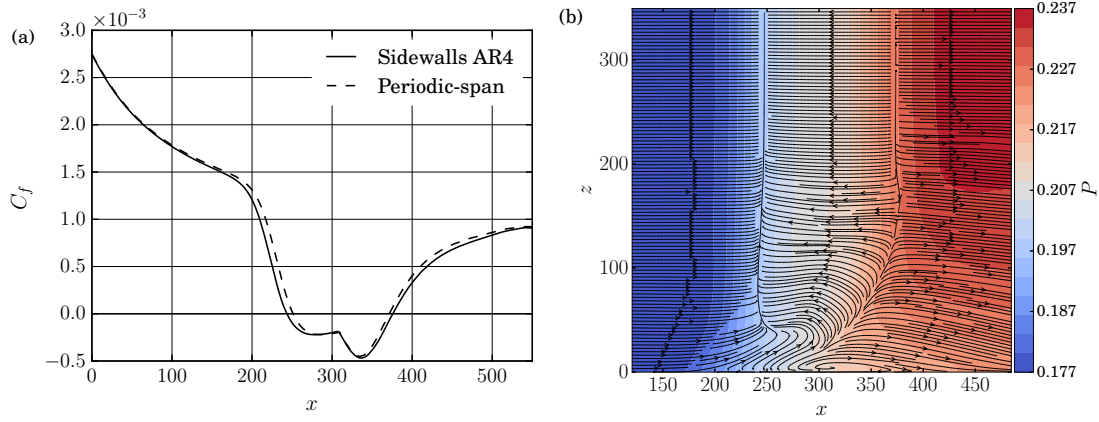


FIGURE 6.15: (a) Centreline skin friction comparison of the $AR = 4$ wide duct with sidewalls to an infinite span. (b) Velocity streamline pattern above the bottom wall for the $AR = 4$ duct, coloured by pressure. Half of the span is shown $z = [0, 350]$ due to the centreline symmetry.

interaction affects the centreline flow is of interest when assessing the viability of the infinite span assumption for modelling SBLI.

Figure 6.15 (b) shows velocity streamlines coloured by pressure above the bottom wall. Half of the span is displayed, as the flow is symmetric about the centreline. The near wall structures are similar to those seen for the narrower aspect ratio in figure 6.8 (a); an attached region of flow is turned inwards by the swept shock and feeds into the central separation bubble. Foci are seen in the corner and at the edge of the central separation, but they are not as pronounced as in the $AR = 1$ case. The bubble is longest in the streamwise direction at roughly 20% of the span away from the sidewall. Separation length then reduces to a constant value at $z \approx 200$, which is maintained as the flow becomes two-dimensional towards the centreline. Beyond this point the streamlines are seen to be anti-parallel to the oncoming flow. The sidewall influence causes visible deflection of the streamlines over almost 30% of the span, which would explain the strong dependence of aspect ratio on L_{sep} for $AR = 2$ and narrower. For example at $AR = 2$ the centreline is located at $z = 175$, within the range of influence seen here. These results are consistent with the experimental laminar SBLI of Degrez et al. (1987), in which an aspect ratio of at least 2.5 was required to see two-dimensional behaviour of the interaction.

Figure 6.16 (a) compares the effect of aspect ratio on the streamwise separation length L_{sep} and the lateral distance in z between the foci and the sidewall L_f . Similar trends are found to those in the experimental literature (Babinsky et al. (2013), Xiang and Babinsky (2019)); smaller aspect ratios lead to suppression of the central separation compared to the quasi-2D result denoted by the dashed line. At medium aspect ratios a peak occurs that is in good agreement with figure 11 of Babinsky et al. (2013) and asymptotes towards the quasi-2D result at the largest aspect ratio. The distance of the foci from the sidewall L_f increases with aspect ratio, noting that at the smallest aspect ratio of $AR = 0.25$ a clear focus could not be identified. The distance the foci shift is small relative to the width of the duct. Increasing the aspect ratio from one to four led to an increase of only 16.5% in L_f , suggesting that the ejection of the corner flow remains mostly localised to the sides of the duct at large aspect ratios.

A method for predicting the central separation size in duct SBLI was proposed by the experimental turbulent studies of Babinsky et al. (2013), Xiang and Babinsky (2019). The key criteria

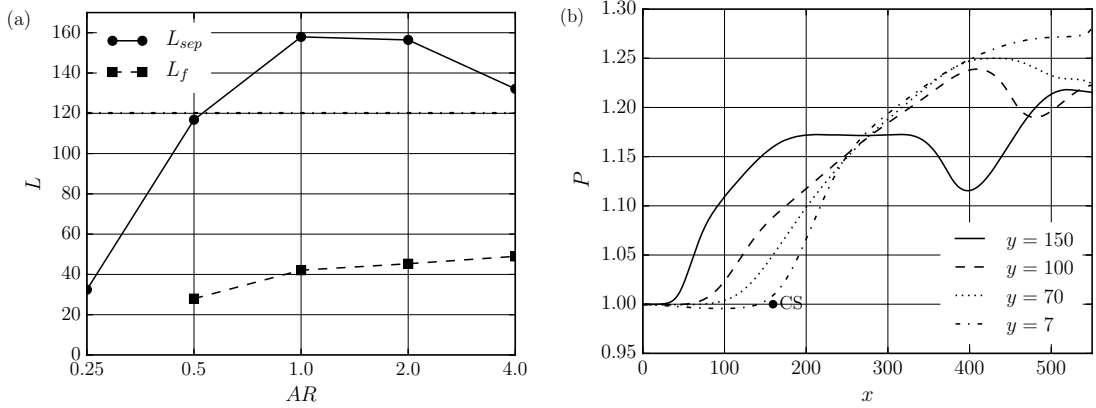


FIGURE 6.16: (a) The effect of duct aspect ratio on centreline streamwise separation length L_{sep} and the distance between the foci and sidewalls L_f . The dashed horizontal line denotes L_{sep} for the span-periodic case. (b) Normalized pressure on the sidewall ($z = 0$) at increasing y heights above the interaction. The ‘CS’ marker is the start of corner separation near the bottom wall.

was proposed to be the streamwise crossing location of shockwaves generated at the onset of corner separation. Corner shocks crossing upstream of the central interaction led to a weakened SBLI compared to quasi-2D predictions. For shocks crossing within the interaction a strengthened SBLI and larger separation bubble were observed. At larger aspect ratios the flow was two-dimensional on the centreline, as the shocks crossed far downstream of the interaction. For comparison to the present work, figure 6.17 (a) shows pressure contours for the $AR = 2$ case in the present study at a height of $y = 7$. This height corresponds to the apex of the central separation bubble, with the imprint of the incident shock seen at $x = 300$ in this plane. The solid black line represents the $C_f = 0$ crossing on the bottom wall of the domain ($y = 0$), to serve as a reference point for the separation bubble. Two oblique structures are seen to traverse the span and cross at the back of the central interaction. As the $AR = 2$ case has the largest recorded L_{sep} , this crossing is consistent with the trends noted in the experimental work. The same experimental trends were observed for the other aspect ratios in this work (not shown), with the smaller aspect ratio cases having structures that crossed upstream of the interaction.

Experimental work has attributed these crossing shockwaves to a bottom wall corner compression effect. However, in the present simulations, observing pressure contours higher up in the duct at $y = 70$ as in figure 6.17 (b), they are seen to originate from the swept SBLI of the incident shock with the sidewalls. It is important to note here that the sidewall compressions leading to the crossing shockwaves occur at an earlier upstream location than the onset of bottom wall corner separation. At $x = 205$ the two conical shocks interact with the incident shock and cause it to strengthen, consistent in shape with figure 9 of Wang et al. (2015). The streamwise development of pressure on the sidewall is given at four y heights in figure 6.16 (b), relative to the start of the corner separation (CS). While there is a pressure rise close to the onset of corner separation at $y = 7$, this effect is also present all the way up the height of the duct due to the swept SBLI and occurs upstream of the corner separation.

This section has reaffirmed that aspect ratio is a crucial parameter for confined SBLI, highlighting that span-periodicity is not a suitable assumption for internally confined flows even for relatively

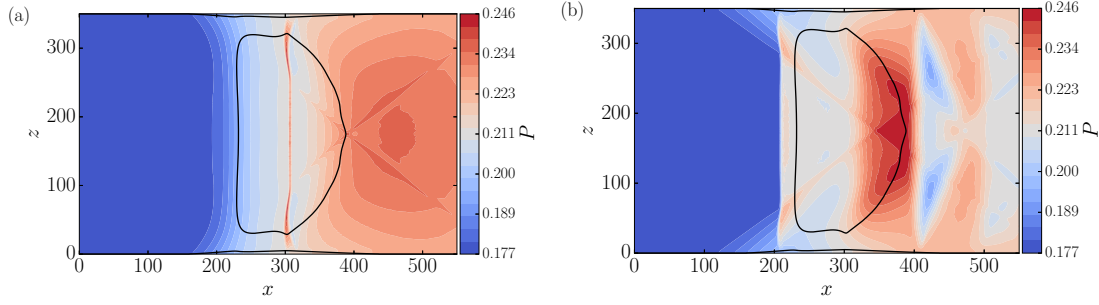


FIGURE 6.17: Slices of x - z pressure for the $AR = 2$ case at (a) $y = 7$ above the separation bubble and at (b) $y = 70$, corresponding to 40% of the duct height above the bottom wall. In both cases the black line is the zero crossing of skin friction on the bottom wall ($y = 0$). The crossing shocks are observed to be generated by sidewall compressions from the initial swept SBLI, independent of the onset of corner separation near the bottom wall.

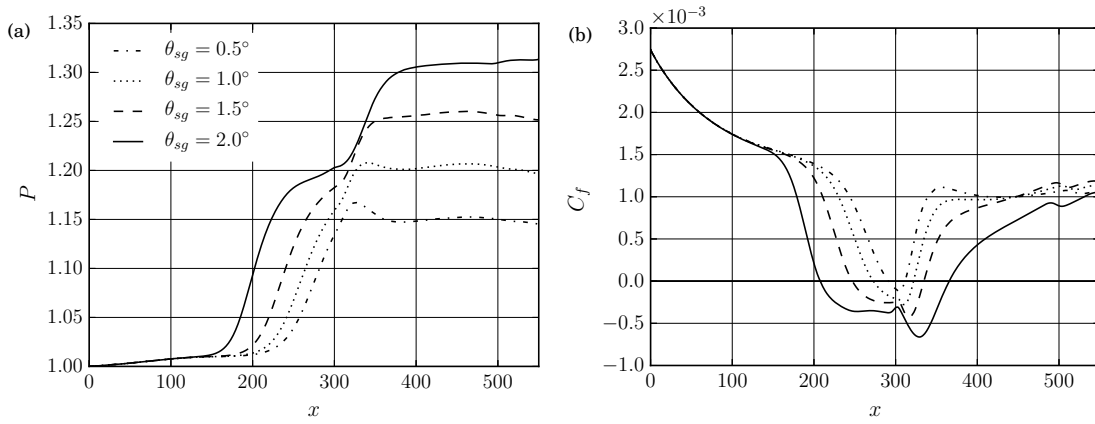


FIGURE 6.18: Sensitivity of the centreline (a) wall pressure and (b) skin friction to incident shock strength for the $AR = 1$ duct. The solid line represents the baseline configuration.

wide ($AR = 2$) ducts. The flow at $AR = 4$ still showed small differences compared to span-periodic predictions. The general impact of aspect ratio on central separation length agreed well with experimental findings. We were, however, unable to attribute this effect to the crossing of shocks from bottom wall corner compressions. It was shown that for the present study the origin of the crossing shockwaves was the initial swept conical SBLI as previously identified in figure 6.13. This effect was shown to occur at a height above the influence of bottom wall corner compressions, and substantially further upstream than any corner separations. As a result, the crossing location is dependent on the height y . There are several differences in this study to the experimental work, which could be causing the difficulty in identifying strong corner shocks. The present study is laminar and far weaker ($\theta_{sg} = 2^\circ$ vs $\theta_{sg} = 8^\circ$), plus there is no gap between the shock generator and the sidewalls which would emphasise the swept conical SBLI relative to weak corner compressions. Nevertheless, the present results are consistent with previous LES (Wang et al., 2015) of turbulent interactions with sidewalls.

6.4.2 Variation of incident shock-strength

In this section the initial flow deflection θ_{sg} is modified to determine the sensitivity of the SBLI to variations in incident shock strength. For each case the starting location of the shock generator

Flow deflection (θ_{sg}°)	p_3/p_1	Interaction region ($x_{sep}, x_{reattach}$)	L_{sep}	% of baseline L_{sep}
0.5	1.145	(291.2, 311.5)	20.28	13%
1.0	1.197	(272.7, 321.9)	49.20	31%
1.5	1.252	(247.1, 334.6)	87.53	55%
2.0	1.313	(207.7, 365.6)	157.93	–

TABLE 6.6: Reduction in streamwise separation length L_{sep} of the main interaction with decreasing incident shock strength. Comparison is made to the centreline skin friction for the baseline $\theta_{sg} = 2.0^\circ$ case with $AR = 1$.

x_{sg} is modified to maintain the same value of Re_x at the inviscid impingement location. One stronger and three weaker interactions are considered for this section, corresponding to flow deflections in the range of $\theta_{sg} \in [0.5, 1, 1.5, 2.5]$. Figure 6.18 (a) shows the normalised centreline pressure along the bottom wall for the baseline and three weaker interactions. As the interaction is weakened, the pressure rise at the start of separation shifts downstream. The outlet pressure ratio p_3/p_1 for the three weaker interactions is given in table 6.6. For each shock strength there is a lack of a pressure plateau in the middle interaction, often seen in quasi-2D laminar SBLI (Katzner (1989), Sansica et al. (2013)). Instead, the flow reattaches quickly and for the two weakest interactions there is actually a pressure reduction after the initial compression.

The effect of the weakened shock on the central separation is shown in figure 6.18 (b). Both the separation and reattachment locations shift down and upstream respectively for the weaker interactions, with the $\theta_{sg} = 0.5^\circ$ case being close to incipient separation on the centreline. It was observed during the simulations that the corner region was the first to separate, owing to the large regions of low-momentum fluid in the corners. The crossing at $x = 500$ of the reflected conical shocks identified in section 6.3.2 is less obvious for the weaker interactions. Aside from the baseline case only the $\theta_{sg} = 1.5^\circ$ case shows the asymmetric double trough C_f distribution; the weaker cases reattach abruptly in a manner similar to the narrow aspect ratio ducts in figure 6.14 (a). Table 6.6 shows the separation and reattachment points for each of the shock strengths and gives the size of L_{sep} relative to the baseline. Although the separation and reattachment are both sensitive to incident shock strength, the effect on the separation point is more severe. A reduction of 1.5° in the initial flow deflection leads to an overall 87% reduction in centreline separation. Each of the weaker interactions followed owl-like topologies of the first kind, with two distinct foci and saddle points. As the interaction was strengthened there was an elongation of the separated region in the streamwise direction.

A stronger interaction at $\theta_{sg} = 2.5^\circ$ was also performed and was found to be very close to the limits of stability downstream of the main interaction. The transition process for duct SBLI is studied in more detail in the next chapter. The results for the stronger interaction are included here to give insight into the limiting behaviour of large separations. Figure 6.19 (a) shows $u - w$ velocity streamlines at $y = 1$ over the same range as in figure 6.10. The same combination of saddle points (S) and foci (F) are observed for the stronger interaction but the size and magnitude of the recirculation is increased. The separation line has shifted upstream and each of the foci have been elongated in the streamwise direction relative to the weaker interaction in figure 6.10. The deformation of the saddle point at the reattachment line is increased, there is no longer a well defined set of two streamlines entering the saddle point. Downstream of the interaction the streamlines no longer remain parallel to each other and a steady laminar flow

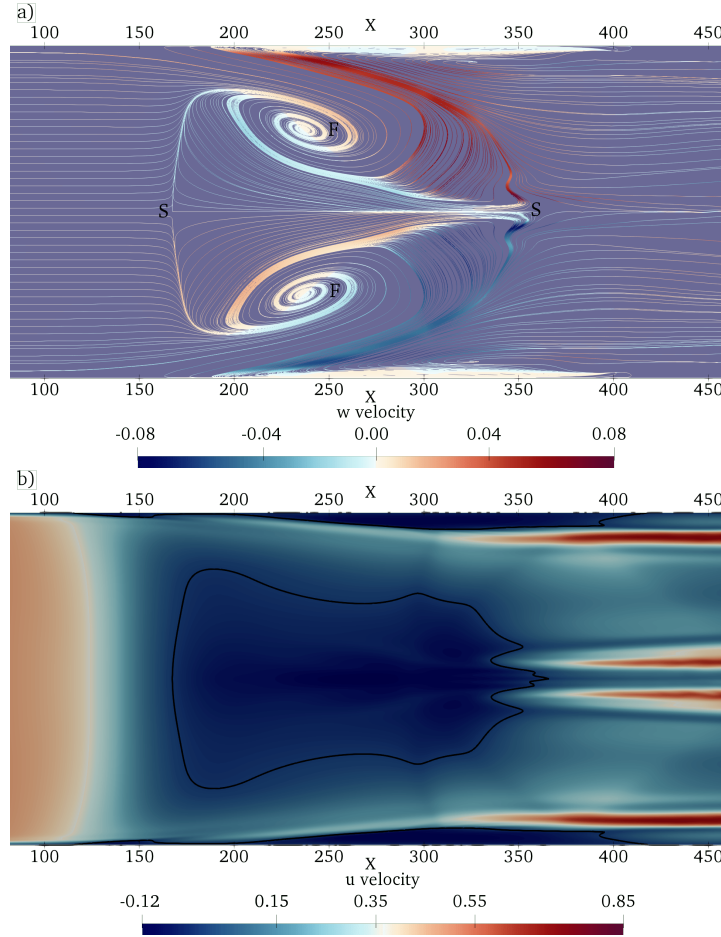


FIGURE 6.19: (a) u - w velocity contours evaluated at $y = 1$ above the bottom wall. Coloured by transverse velocity component w . Saddle (S) and foci (F) are highlighted as in figure 6.10. (b) Streamwise velocity at $y = 1$ above the bottom wall. The solid black line shows the $u = 0$ line to highlight regions of recirculation. Four high-speed streaks are observed downstream of the interaction in red. A darkened imprint of the conical swept shock is also visible.

was not obtained. A grid refinement study was performed with 100% additional grid points in each direction independently. A laminar solution was not obtained for any of the refined grids, with intermittent regions of transition observed near the outlet.

The streamwise velocity contours of figure 6.19 (b) show four high-speed streaks downstream of the main interaction. The flow was found to accelerate rapidly after the apex of each of the corner separations and on either side of the centreline saddle point. While the interaction is still roughly ‘owl-like’ of the first kind, the deformation of the attachment line saddle point could indicate an intermediate state approaching the second owl-like state shown schematically in figure 4 (a) of [Eagle and Driscoll \(2014\)](#). For owl-like patterns of the second kind, the rear saddle point transitions into a node with multiple streamlines directed into it. Owl-like patterns of the second kind for turbulent SBLI have been shown to correspond to stronger interactions in experiments ([Xiang and Babinsky, 2019](#)). Based on the topological trends seen in this case, it is feasible that the transition mechanism for stronger three-dimensional laminar SBLI involves the bifurcation from topologies of the first kind to the second.

6.4.3 Further investigation of the the trailing edge expansion fan effect

It was demonstrated in section 6.2.3 during selection of the domain that for laminar duct SBLI the flow on the centreline is sensitive to the length of shock generator used. For longer shock generators the trailing expansion fan recovery occurs further downstream and leads to an increase in the central separation bubble length (figure 6.5(b)). A comparison to figure 6.4(b) showed this is a purely three-dimensional effect that is not present in quasi-2D laminar SBLI. The only difference between the quasi-2D and 3D simulations is the lateral confinement imposed by the sidewalls. Table 6.4 reported a 6% increase in L_{sep} between $L_{sg} = 300$ and $L_{sg} = 350$, a limiting case of maximum L_{sep} was not found for the range of L_{sg} tested. This section aims to address this by investigating the limiting behaviour of the interaction for a longer duct with both short ($L_{sg} = 200$) and very long ($L_{sg} = 600$) shock generators. In both cases the trailing expansion fan is located far downstream of the central interaction and does not impinge directly on the bottom wall separation bubble. The baseline domain from table 6.1 is extended by $\sim 55\%$ in the streamwise direction to a length of $L_x = 850$. A modified grid distribution of $(1150 \times 455 \times 455)$ is applied to maintain the same streamwise resolution as before.

Figure 6.20 shows the instantaneous flow features of the two long domain cases at $t = 12000$ for shock generators of length $L_{sg} = 600$ (a,c,e), and $L_{sg} = 200$ (b,d,f). Instantaneous skin friction relative to the bottom ($\frac{\partial u}{\partial y}$) and sidewalls ($\frac{\partial u}{\partial z}$) are shown in (c,d) and (e,f) respectively, with centreline ($z = 87.5$) density contours given in (a,b). For the skin friction plots the solid white line highlights the $C_f = 0$ crossing of the component relative to that wall. In both of the density plots of figure 6.20 (a,b) a shock train is clearly visible; compression waves at the start of the central separation coalesce into a shock that impinges on the upper surface and causes a small region of secondary flow-reversal. A further reflection occurs and this wave impinges on the bottom wall. The reflected incident shock also reflects between the upper and lower walls of the domain, highlighting the complex secondary reflections present for internally confined flows. A trailing edge expansion fan can be seen originating from the upper surface at $x = 245$ for the short shock generator, first hitting the bottom wall boundary layer at around $x = 500$. In the long case an expansion fan is generated at $x = 645$ which exits through the outlet before hitting the bottom wall. For the long shock generator the flow transitions to turbulence while remaining laminar for the shorter $L_{sg} = 200$. This suggests that if geometry permits, the strength and size of the central separation can be modified by the use of shorter shock generators.

The difference in the reattached boundary layer state is clearer to see in figures 6.20 (c,d). For the long shock generator in figure 6.20 (c) the flow transitions at $x = 600$ on the centreline and in both corners of the duct. The corner separation is truncated as the boundary layer transitions and remains attached in the corner towards the outlet. The central separation is substantially larger for the longer shock generator and is characterized by a flat separation line across the span and a distortion of the reattachment line peaking on the centreline. The separated region covers more of the span and is also noticeably thicker in the corner regions. The shape of the interaction is very similar to the stronger $\theta_{sg} = 2.5^\circ$ interaction shown in figure 6.19 (b). This demonstrates that for two different methods of achieving a stronger interaction, the shape of the central recirculation has a topology common to both. Comparing to the short shock generator of figure 6.20 (d) we note the downstream shift of the expansion fan has led to an upstream and downstream shift of the separation and reattachment points respectively. The downstream shift of the expansion fan in the longer case also leads to smaller secondary recirculation zones

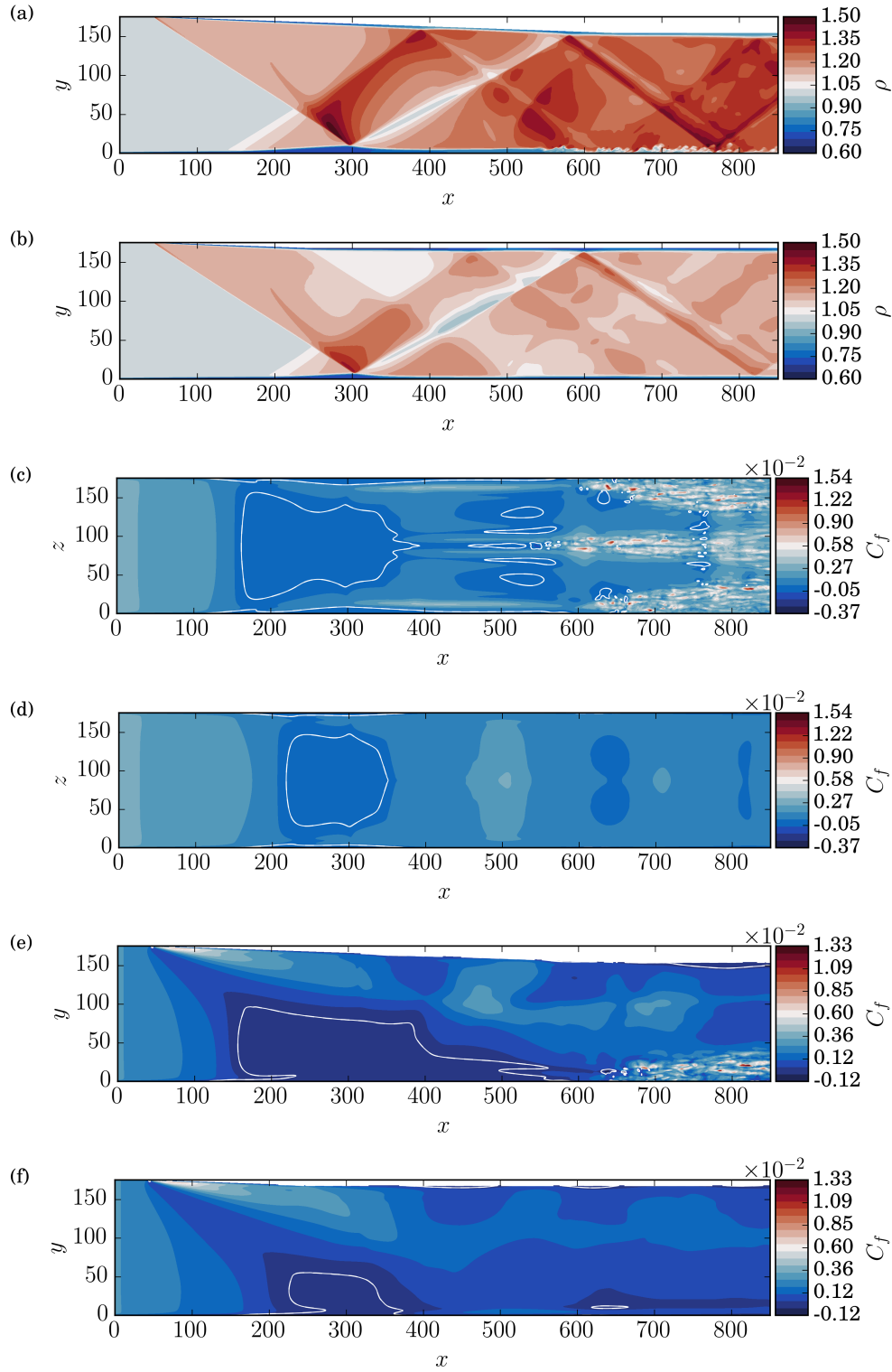


FIGURE 6.20: Long domain duct ($L_x = 850$, $\theta_{sg} = 2^\circ$, $AR = 1$) demonstrating the ability of the trailing expansion fan to control the interaction. The figures correspond to shock generators with $L_{sg} = 600$ in (a,c,e) and $L_{sg} = 200$ in (b,d,f). Showing centreline density ($z = 87.5$) (a,b), skin friction on the bottom wall ($y = 0$) (c,d) and skin friction on the sidewall ($z = 0$) (e,f). The solid white line highlights the zero crossing of skin friction, enclosing regions of flow recirculation.

at $x = 500$. Comparing their location to the shock pattern in figure 6.20 (a) it is clear the secondary separations cannot be attributed to the first vertical secondary reflection that impinges at $x = 600$, and instead are likely caused by the lateral reflections shown downstream of the interaction in figure 6.12.

Finally the impact of shock generator length on the sidewall flow is illustrated in figures 6.20 (e,f). While the shape of the sidewall separation is similar in both cases, the downstream shift of the trailing expansion fan leads to a significant enlargement of the sidewall recirculation. For the short case it can be seen that the expansion fan causes a recovery of the sidewall flow that wards off separation after $x = 400$. Smaller separation bubbles are seen further downstream due to the secondary reflections present in figure 6.20 (b). For the long shock generator case in figure 6.20 (e) the sidewall recirculation covers a much larger portion of the sidewall and is only truncated as the flow begins to transition around $x = 600$. As the expansion fan originates from the upper surface at $x = 645$ and is directed towards the outlet, we conclude that the sidewall separation is limited by the onset of transition in the sidewall boundary layer. This is in contrast to the short shock generator case where the sidewall separation is prematurely terminated by the upstream shift of the trailing expansion fan. The importance of transition in limiting sidewall separation suggests that attempts such as [Giepman et al. \(2016\)](#) to suppress the size of laminar SBLI with transition could also benefit from placing trips on the sides of the duct. We have seen that the central separation and state of the reattached boundary layer can be modified simply by shortening the shock generator. Shorter shock generators modify the sidewall flow considerably and limit the growth of both the corner and sidewall separations. The modified interaction is observed to be weaker and leads to reduced three-dimensionality and a suppressed central separation bubble.

6.5 Discussion

Three-dimensional laminar SBLI at Mach 2 have been investigated numerically for enclosed rectangular ducts. Similar to previous turbulent cases ([Bermejo-Moreno et al. \(2014\)](#), [Wang et al. \(2015\)](#)), a strong dependence of duct aspect ratio was observed in the laminar case, albeit with much larger recirculation regions. Lateral confinement of laminar SBLI from sidewalls leads to a strengthened and highly three-dimensional interaction. Span-periodic analysis is unable to predict centreline skin friction distributions except in the limit of very wide aspect ratios ($AR \geq 4$). The streamwise extent of the central separation was almost identical for ducts with an aspect ratio of one or two. Aspect ratios less than unity had a substantial decrease in separation length and showed multiple reflections of laterally travelling shockwaves. At an aspect ratio of four the three-dimensionality of the interaction was limited to a region 30% of the width of the span away from the sidewall. The centreline separation length was observed to still be 9% different from quasi-2D predictions. Due to the dependency of the interaction on duct geometry and shock generator length, it is not possible to predict centreline separations without prior knowledge of these parameters. The baseline one-to-one aspect ratio configuration had a 30% stronger interaction compared to quasi-2D predictions, but this was shown to be dependent on the length of shock generator used.

In addition to aspect ratio, the interaction was found to be strongly influenced by the expansion fan generated from the trailing edge of the shock generator. For expansion fan impingement points further upstream, a decrease in the size and magnitude of central recirculation was observed. The expansion fan effect was purely a three-dimensional effect; for quasi-2D simulations there was no dependence on shock generator length over the same range. Critical point analysis showed that the confined SBLI had ‘owl-like’ topologies of the first kind as introduced by [Perry and Hornung \(1984\)](#). For a stronger interaction the central recirculation was elongated in the streamwise direction and a distortion of the attachment line was observed. Preliminary results highlighted four high-speed streaks downstream of the interaction near the centreline and in the corner. Therefore, streak instability is one potential transition mechanism of confined SBLI.

Shock structures identified downstream of the interaction were shown to result from reflections of the conical swept SBLI generated between the shock generator and sidewalls. Significant corner compressions from the bottom of the domain could not be identified; instead the primary mechanism behind the strengthened interaction was the swept conical SBLI. The swept interaction was shown to begin substantially further upstream than the onset of bottom wall corner separation. A 55% longer domain case was simulated to demonstrate the ability to control the central separation with shorter shock generators. The recovery from the trailing expansion fan suppresses sidewall recirculation and modifies the strength of the interaction as a whole. The role of transition is considered in the next chapter. Differences in the origin of crossing shocks identified in numerical work and experiment, which have been attributed to both the swept SBLI and bottom wall corner compressions respectively should also be investigated.

Chapter 7

The effect of sidewall flow confinement on transitional shockwave/boundary-layer interactions

7.1 Introduction

In this chapter transitional oblique shock-wave/boundary-layer interactions (SBLI) are simulated for a rectangular duct with a $\theta_{sg} = 5^\circ$ shock generator ramp at Mach 2. The simulations can be viewed as an extension of the previous chapter, to investigate the influence of intermittent upstream transition on the sidewall-bounded SBLI. The baseline configuration is a duct with an aspect ratio of 0.5. The higher initial flow deflection of $\theta_{sg} = 5^\circ$ was selected with the aim of achieving incipient flow separation despite the now transitional state of the boundary layer. Time-dependent disturbances are added to the base laminar flow via wall localised blowing/suction strips to obtain intermittent transition upstream of the SBLI. Two forcing configurations are evaluated to assess the response of the SBLI to different tripping locations. Both the length of the duct and the Reynolds number are increased to allow for transition to develop upstream of the SBLI. While lower amplitude disturbances could have been selected to investigate the transition procedure downstream of the SBLI, this chapter focuses on scenarios in which significant transition occurs upstream of the SBLI. This is an intentional design choice, as it allows for the large-scale unsteady dynamics of a transitioning separation region to be investigated. Furthermore, the forcing is selected to be asymmetrical across the duct, to mimic potential real-world scenarios where transition is influenced by both surface imperfections and unsteady environmental effects.

The computations were performed as an Implicit Large Eddy Simulation (ILES) using the 6th order Targeted Essentially Non-Oscillatory (TENØ) scheme described in chapter 2, using a value of $C_T = 1 \times 10^{-7}$. As previously mentioned, many studies of SBLI make the assumption of a

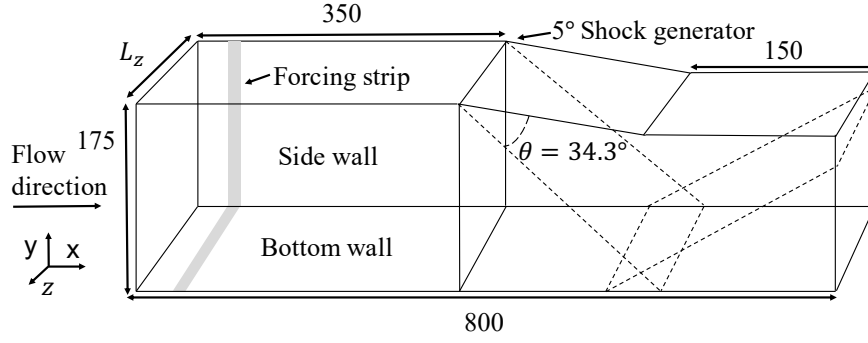


FIGURE 7.1: Non-dimensional computational domain for the rectangular duct. The width of the span L_z is 87.5 and 175 for the half and one aspect ratio cases respectively.

simplified periodic (quasi-2D) span to reduce computational cost. As seen in the previous chapter for confined laminar SBLI, this approach is unable to reproduce the complex flow features that arise for internally confined flows. Examples of span-periodic numerical studies of transitional oblique SBLI include [Sansica et al. \(2016\)](#), [Larchevêque \(2016\)](#) and [Quadros and Bernardini \(2018\)](#). In each case the transitional SBLI was highly unsteady with a range of low and high frequency phenomena arising from the complex coupling between inviscid and viscous effects. Many of the existing works focus on configurations where the flow transitions downstream of the SBLI. In the present contribution the SBLI is exposed to intermittent transition that develops as turbulent spots upstream of the primary shock reflection. Examples of the OpenSBLI code being applied to high-speed transitional flows includes [Lusher and Sandham \(2019b\)](#) as in chapter 5, and the hypersonic roughness-induced transition of [Lefieux et al. \(2019\)](#).

The chapter is organised as follows. The physical problem is specified in section 7.2.1, including details of the computational domain and the initialisation of laminar boundary layers within the duct. Section 7.2.2 introduces the time-dependent blowing/suction strips that generate disturbances upstream of the SBLI. In section 7.3.1 the baseline half aspect-ratio results are presented for two different forcing configurations. Instantaneous snapshots of the near-wall region show the response of the separation bubble to intermittent turbulent spots. Finally in section 7.3.2 the same two forcing configurations are simulated for a wider 1:1 aspect-ratio duct to demonstrate the effects of sidewall confinement.

7.2 Computational method

7.2.1 Problem specification

The computational domain consists of a rectangular duct with a finite-length $\theta_{sg} = 5^\circ$ internal shock generator as shown in Figure 7.1. The non-dimensional streamwise length and height of the duct are set as $L_x = 800$ and $L_y = 175$ respectively. Ducts with aspect ratios ($AR = L_z/L_y$) of $AR = 0.5$ and $AR = 1$ are simulated in this chapter. The shock generator ramp starts at $x_{sg} = 350$ and has a length of $L_{sg} = 300$; a trailing edge expansion fan is generated at the end of the ramp and exits through the outlet plane of the computational domain. Following the canonical works of [Hakkinen et al. \(1959\)](#) and [Katzer \(1989\)](#), a Mach 2 freestream is initialised

in the domain with laminar boundary layer profiles imposed on the bottom and sidewalls of the duct. Freestream conditions are maintained on the top portion of the domain upstream of the shock generator. The laminar profiles are obtained via the same similarity solution (White, 2006) used in the previous chapter, described in section 3.3.7. The Reynolds number based on the inlet displacement thickness is $Re_{\delta^*} = 1500$.

No-slip isothermal wall conditions are enforced on the bottom wall, sidewalls, and on the shock generator for a non-dimensional wall temperature equal to the adiabatic wall condition from the similarity solution of $T_w = 1.676$ (4 s.f.). A pressure extrapolation is applied at the inlet, with zero gradient conditions enforced on the outlet and upstream of the shock generator. In the corners of the duct boundary layer profiles of equal thickness from two adjacent walls are blended together to form a smooth corner profile as described in section 6.2.1.

7.2.2 Time-dependent upstream disturbances

The aim of the chapter is to observe the response of oblique duct SBLI for situations where transition occurs upstream of the interaction but the boundary layers are not yet fully turbulent. To induce transition upstream of the shock reflection, disturbances are added to the laminar boundary-layer via strips of wall normal blowing/suction as shown in Figure 7.1. Random phases are added to the forcing to observe how asymmetry in the initial disturbances can influence the resulting SBLI flow structures. Further asymmetry can be added to the problem by forcing specific walls within the duct while keeping others laminar. The two configurations considered here are one in which the sidewalls and bottom wall are forced, and one where the bottom wall is kept initially laminar.

For the disturbance strip located on the bottom wall of the domain ($y = 0$): the v velocity component is forced using a modified version of the forcing presented in Pirozzoli et al. (2004) such that

$$v(x, z, t) = \begin{cases} A_0 f(x) g(z) h(t) & \text{if } x_a < x < x_b \\ 0 & \text{otherwise,} \end{cases} \quad (7.1)$$

for streamwise $f(x)$, spanwise $g(z)$ and time $h(t)$ dependence

$$f(x) = \exp \left(- \left(\frac{(x - (x_b - x_a))^2}{16} \right) \right), \quad (7.2)$$

$$g(z) = \sum_{l=1}^{l_{\max}} Z_l \sin(2\pi l(z/L_z + \phi_l)), \quad (7.3)$$

$$h(t) = \sum_{m=1}^{m_{\max}} T_m \sin(\omega_m t + 2\pi \phi_m), \quad (7.4)$$

where the constants Z_l , T_m are defined by the relations

TABLE 7.1: Overview of the four computational test cases with different forcing configurations. The prefixes (A) and (B) refer to cases with half and one aspect ratio ducts respectively.

Simulation cases	Aspect ratio (L_z/L_y)	Grid size (N_x, N_y, N_z)	Wall forcing locations
A1	0.5	(1050, 375, 325)	Bottom & sidewalls
A2	0.5	(1050, 375, 325)	Sidewalls only
B1	1	(1050, 375, 645)	Bottom & sidewalls
B2	1	(1050, 375, 645)	Sidewalls only

$$\sum_{l=1}^{l_{\max}} Z_l = 1, \quad Z_l = 1.25Z_{l+1}, \quad \sum_{m=1}^{m_{\max}} T_m = 1, \quad T_m = 1.25T_{m+1}. \quad (7.5)$$

Random phases ϕ_l, ϕ_m distributed between $[0, 1]$ are added for each of the $l, m = 20$ terms in the spatial dependence of the forcing. The same phases are fixed between different simulations but differ on each of the three forcing strips to create the asymmetry in the initial breakdown. In the corners the forcing is linearly damped to zero within a distance of one boundary layer thickness from the walls. This damping ensures that the forcing from adjacent walls does not introduce discontinuities to the flow. Forcing is applied for a range of evenly spaced frequencies ω_m in the interval $[0.04, 0.12]$, taken from linear stability analysis of compressible boundary-layers [Sansica \(2015\)](#). The forcing strip is located between $x_a = 10.0$, $x_b = 30.0$ with an amplitude of $A_0 = 0.2$. A high amplitude is selected to trigger transition upstream of the shock reflection. On both sidewalls of the domain the same forcing is applied for the wall-normal velocity w , where the spatial variation $g(z)$ is instead taken over $g(y)$.

7.2.3 Test case specification

Two forcing configurations are considered in this section as shown in Table 7.1: one with disturbances added to both the bottom and sidewalls of the domain (cases A1, B1), and one where the bottom wall is kept laminar, to observe how the transition develops over the span (cases A2, B2). The cases (A) and (B) refer to half and one aspect ratio ducts respectively to assess the effect aspect ratio has on the duct SBLI. Grid stretching is performed symmetrically in the y and z directions to cluster points in the boundary layers of each wall, with a uniform distribution in x . Grid points in y and z are distributed with stretching factors $s_1, s_2 = 1.6, 1.4$ as

$$y = \frac{1}{2}L_y \frac{1 - \tanh(s_1(1 - 2\xi))}{\tanh(s_1)}, \quad z = \frac{1}{2}L_z \frac{1 - \tanh(s_2(1 - 2\xi))}{\tanh(s_2)}, \quad (7.6)$$

for uniformly distributed points $\xi = [0, 1]$. Polynomial smoothing is applied to each of the grid lines in the streamwise x direction to ensure continuity at the start and end locations of the shock generator section of the duct. The baseline half-aspect ratio grid configuration from table 7.1 has wall units of $\Delta x^+ = 18$, $2.5 < \Delta y^+ < 18$ and $2.2 < \Delta z^+ < 10$, based on C_f evaluated in the turbulent region at the outlet. The streamwise grid resolution was selected using the LES recommendations of [Georgiadis et al. \(2010\)](#). We note that the y and z resolution in the first cell is lower than the recommended values for DNS by a factor of around 2 in each direction, which is not unreasonable for an ILES approach. While a stronger stretching factor could have been

used to improve near-wall resolution, this would have decreased the resolution in the middle of the sidewalls in y . As such, it was decided to use a relatively low stretching to avoid severe under-resolution in other areas of the duct.

Grid sensitivity was shown for the laminar duct cases of a similar set-up in section 6.2.2, and the performance of the schemes for span-periodic transitional SBLI was reported in [Lusher and Sandham \(2019a\)](#). The number of grid points is doubled in the span-wise z direction for the wider $AR = 1$ cases (B1, B2). A non-dimensional time-step of $\Delta t = 5 \times 10^{-3}$ is used throughout. Three flow-through times of the domain were simulated to clear the initial transient. The initial run was monitored qualitatively to observe the development of the transition and to let the shock reflection form. Time averaging was applied for a further four flow-through times with statistics collected every time-step.

7.3 Results

7.3.1 Baseline half-aspect ratio duct

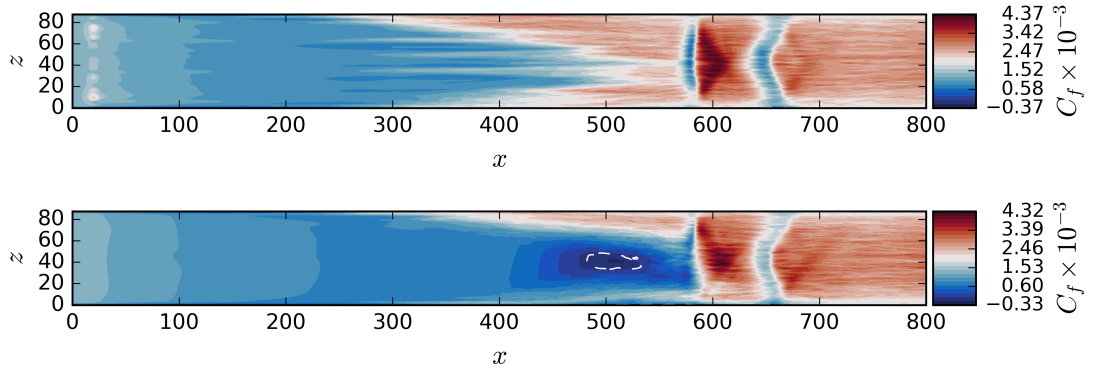


FIGURE 7.2: Time averaged skin friction distributions on the bottom wall. For the cases of (top) forced sidewalls and bottom wall (A1) and (bottom) forced sidewalls only (A2). The dashed white line encloses regions of flow recirculation.

In this section the baseline ($AR = 0.5$) duct is simulated for the two forcing configurations A1 and A2. Time averaged skin friction on the bottom wall is shown in Figure 7.2 for the fully forced (top) and laminar bottom wall cases (bottom). The incident shock is set up to impinge on the bottom wall boundary layer at $x = 580$, which is curved as a result of the swept sidewall SBLI. In the presence of sidewalls the incident shock is strengthened near the centreline and weakened towards the sidewalls, consistent with [Wang et al. \(2015\)](#), [Lusher and Sandham \(2019b\)](#), and the previous chapter. For the fully forced case (A1) there is no mean flow separation ($C_f < 0$) at the centreline, the transition develops first in the corners and spreads out in a wedge across the span. In contrast, the laminar bottom wall case shows an oval region of mean flow separation bordered by the turbulence generated in the corners. The latter case also exhibits an asymmetry in the development of the transition, likely caused by the asymmetry in the random phases of the forcing and a relatively short averaging time. The mean flow does not exhibit flow separation in the corner regions at this moderate shock strength of $\theta_{sg} = 5.0^\circ$. This is in contrast to the

laminar cases from the previous chapter, where $AR = 0.5$ ducts were shown to separate for a weaker interaction of $\theta_{sg} = 2.0^\circ$.

At $x = 600$ in figure 7.2 (top) there is a chevron shaped region of high skin friction shown in red followed by a secondary shock in blue at $x = 650$. Comparing this to the centreline pressure plot of figure 7.3 (left) we see that, after the initial shock-wave, there is a decrease in pressure corresponding to an expansion before the secondary pressure rise at $x = 650$. Both forcing configurations lead to the same outlet pressure value; however the case with a transitional bottom wall (A2) shows a later initial pressure rise. This behaviour can be attributed to compression waves emerging from the front of the mean separation bubble which is only present in case A2.

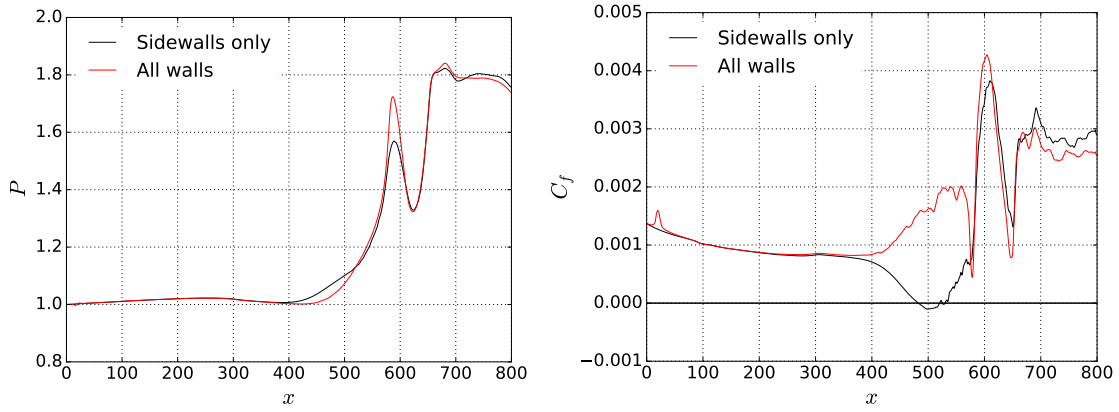


FIGURE 7.3: Time averaged centreline normalised pressure (left) and skin friction (right) on the bottom wall of the duct. For the cases of forced sidewalls and bottom wall (red)(A1) and forced sidewalls only (black)(A2).

The flow features immediately downstream of the SBLI are surprisingly similar to the shock-train patterns found in rectangular turbulent ducts of small cross-sectional area. For example the time-averaged contours in figure 8 of Fiévet et al. (2017) show a similar type of alternating expansion-shock pattern. A triangular region of expansion is followed by a curved shock bounded by rapidly growing sidewall boundary layers. It is plausible that the shock-induced transition and subsequent thickening of the sidewall boundary layers in the present study leads to a reduction in cross-sectional area and the formation of a weak shock train. Furthermore, we note that the blue ‘shock-associated’ region in the time averaged contours of figure 7.2 is relatively thick in the streamwise direction. The width of this region suggests that this secondary shock structure is unsteady in time.

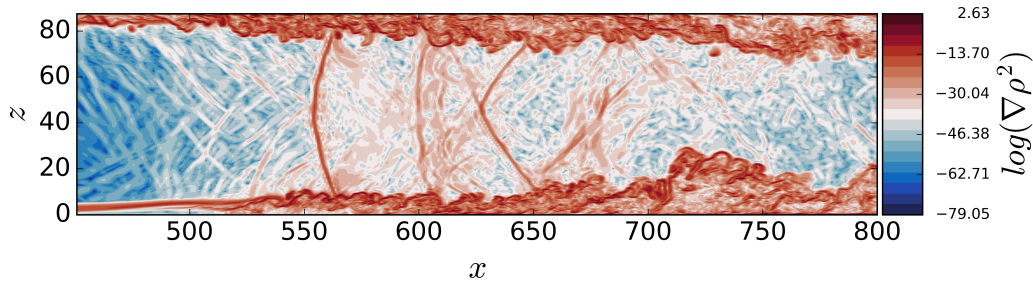


FIGURE 7.4: Instantaneous logarithm of density gradients $\log_{10}(\nabla \rho^2)$ between $x = [450, 800]$ for case (A1) at a height of $y = 25$. Red regions correspond to areas of high density gradients showing the shock structure downstream of the SBLI.

To reinforce this conclusion, figure 7.4 shows the instantaneous logarithm of density gradients $\log_{10}(\nabla \rho^2)$ for case A1 at a height of $y = 25$ above the bottom wall ($\approx 0.15L_y$). At this height the secondary shock is observed as a thin region of high density gradients between $x = 625$ and $x = 650$. The secondary shock was visible in y planes up until a height of ($\approx 0.5L_y$) but had no obvious origin. It is believed that the secondary shock develops as a result of the sudden reduction in cross-sectional area as the flow transitions to turbulence after the SBLI in the lower half of the duct.

A weaker feature in figure 7.2 is the wave pattern that crosses the centreline at $x = 700$. These crossing structures are a result of the narrow duct as seen in chapter 6, and would not be present in infinite span simulations. They are associated with shocks that are generated at the top of the domain where the sidewall flow is deflected by the shock generator ramp. The conical shock-waves propagate inwards towards the centreline and cause multiple lateral reflections across the span (Lush and Sandham, 2019b). In the skin friction curve of figure 7.3 (right) the effect of a more developed transition is clear for the fully forced case (A1). Near both sidewalls there is a rise in skin friction signifying that the transition has reached the centreline and is warding off mean flow separation. When only the sidewalls are tripped (A2), incipient centreline separation is reached. The peak in skin friction is about 10% higher in this case, indicating a more energetic breakdown process.

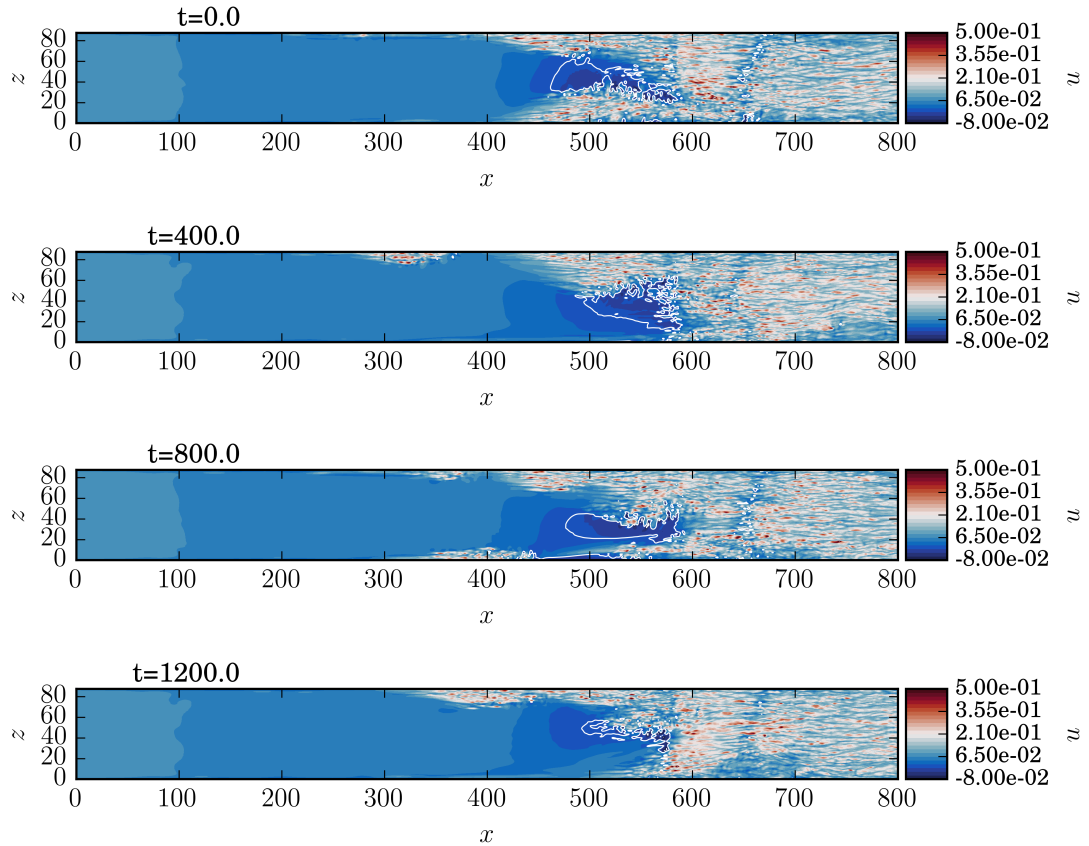


FIGURE 7.5: Instantaneous streamwise velocity snapshots above the bottom wall for the half aspect ratio case A2. The solid white line represents the $u = 0$ contour enclosing regions of recirculation. The central recirculation bubble shifts laterally due to oncoming intermittent turbulent spots.

To further elucidate the response of the central separation bubble to intermittent turbulence, a selection of instantaneous streamwise velocity snapshots are shown in Figure 7.5. The snapshots cover one and a half flow-through times after the initial SBLI has formed. The case shown is the one with the unforced bottom wall (A2). The four figures are spaced at intervals of $\Delta t = 400$, which each correspond to half a flow-through of the freestream. In the top image the transition is relatively symmetric about the centreline and a small region of recirculation is present. At $t = 400$ the transition is stronger on the upper $z = 87.5$ sidewall, pushing the recirculation bubble down towards the $x = 0$ sidewall. By $t = 800$ a corner separation has formed on the lower sidewall at $x = 500$, which is amplifying initial disturbances and causing a breakdown near the corner reattachment point of $x = 525$. At $x = 400$ a turbulent spot has just reached the front of the corner separation and starts interacting with the bubble. Finally by $t = 1200$ the corner separation has been fully removed. The increased mixing rates inside the turbulent spot transfers low-momentum fluid away from the wall. The central separation has contracted and shifts back towards the upper sidewall.

7.3.2 Aspect-ratio one duct

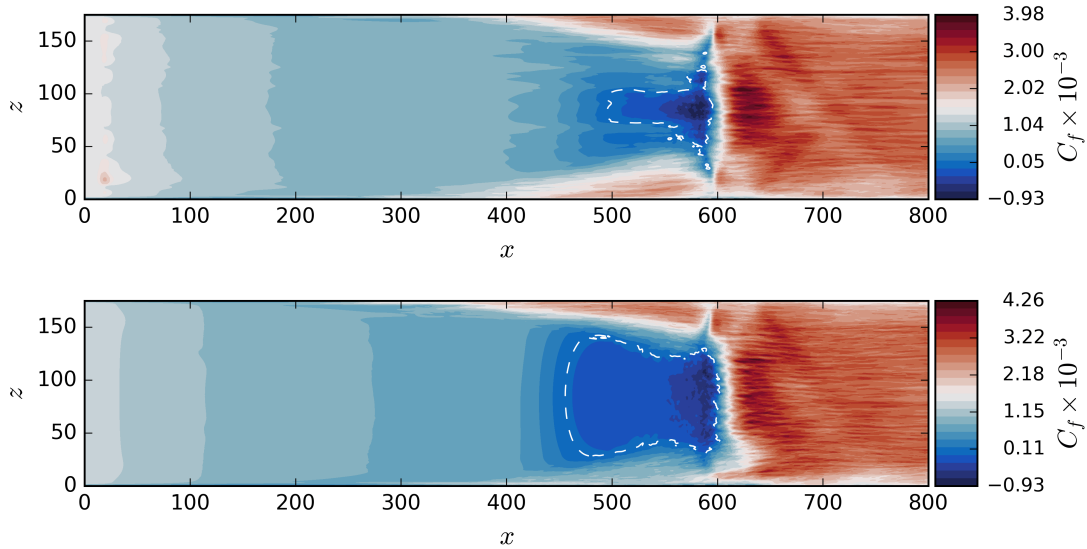


FIGURE 7.6: Time averaged skin friction distributions on the bottom wall. For the cases of (top) forced sidewalls and bottom wall (B1) and (bottom) forced sidewalls only (B2). The dashed white line encloses regions of flow recirculation.

In this section the duct is widened to an aspect ratio of $AR = 1$, to observe how the lateral confinement influences the oblique SBLI. The grid resolution across the span is increased as in Table 7.1 to compensate for the increase in duct width. Figure 7.6 shows time-averaged skin friction contours on the bottom wall for cases B1 and B2. In contrast to the $AR = 0.5$ cases both of the $AR = 1$ ducts exhibit mean flow separation on the centreline. This strong dependence on the SBLI of aspect ratio is consistent with previous numerical (Wang et al., 2015) and experimental (Xiang and Babinsky (2019) at $AR = 1.33$) studies of confined oblique SBLI. As it has been observed previously (e.g. figure 20 of Xiang and Babinsky (2019) and chapter 6 of this thesis), there is a weakened interaction relative to infinite span predictions for narrower ducts.

As the duct aspect ratio is increased, a stronger interaction is observed, with larger regions of flow separation on the centreline. For larger ducts ($AR \gg 1$) the separation sizes reduce again and tend towards the infinite-span predictions. The same asymmetry is observed in the development of the transition front for case B2 as was seen in A2. In case B2 the initially laminar bottom wall leads to a substantially larger separation bubble compared to the forced case B1. For both of the aspect ratios it has been shown that tripping only the sidewalls is not sufficient when attempting to suppress the central flow separation. While case B1 still exhibits mean flow separation, the addition of upstream disturbances to the bottom wall has significantly reduced the extent of flow separation.

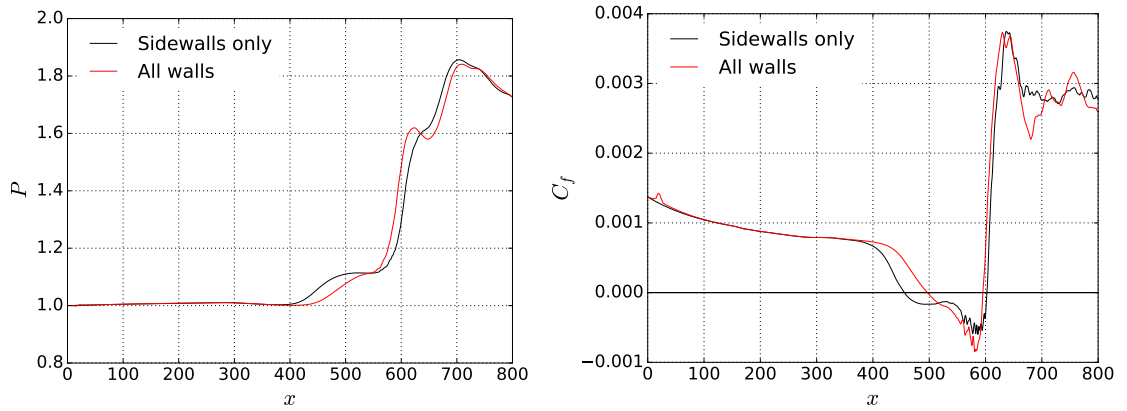


FIGURE 7.7: Time averaged centreline normalised pressure (left) and skin friction (right) on the bottom wall of the duct. For the $AR = 1$ cases of forced sidewalls and bottom wall (red)(B1) and forced sidewalls only (black)(B2).

The secondary shock that was present in the $AR = 0.5$ case is not seen at $AR = 1$. Despite the same flow conditions and initial disturbances, reducing the lateral flow confinement has suppressed the formation of the structures seen in figure 7.2. Figure 7.7 shows the time-averaged centreline wall pressure and skin friction distributions at $AR = 1$. In the pressure plot we see the initial pressure rise from the SBLI starts further upstream for case (B2) where the bottom wall remains laminar. The same can be seen in the skin friction curve: the separation point moves much further upstream compared to case B1 despite both having a similar reattachment location. There is a more pronounced pressure plateau at $x = 525$ for case B2, which is characteristic of a laminar SBLI (Katzner, 1989; Degrez et al., 1987). Similar to the $AR = 0.5$ case there is no mean flow separation in the corner regions, due to the onset of energetic mixing near the sidewalls. This is in contrast to laminar duct SBLI from chapter 6 and stronger-shock turbulent SBLI of Wang et al. (2015), in which long thin corner separations develop.

Figure 7.8 shows an $x - z$ plane of pressure evaluated at $y = 105$ for the $AR = 1$ case. The incident oblique shock is visible at $x = 450$ and is curved symmetrically about the centreline. A series of compression waves can be seen upstream of the main shock. These waves pass through and reinforce the incident shock. The compression waves coalesce into weak shock-waves and are observed to cross in this plane at $x = 475$. These features start at the intersection of the sidewall and shock generator ramp at the top of the domain and develop following a conical structure. The result is that there are two regions of the incident shock either side of the centreline that have been strengthened by the sidewall compressions. The crossing location is dependent on which y plane is being observed.

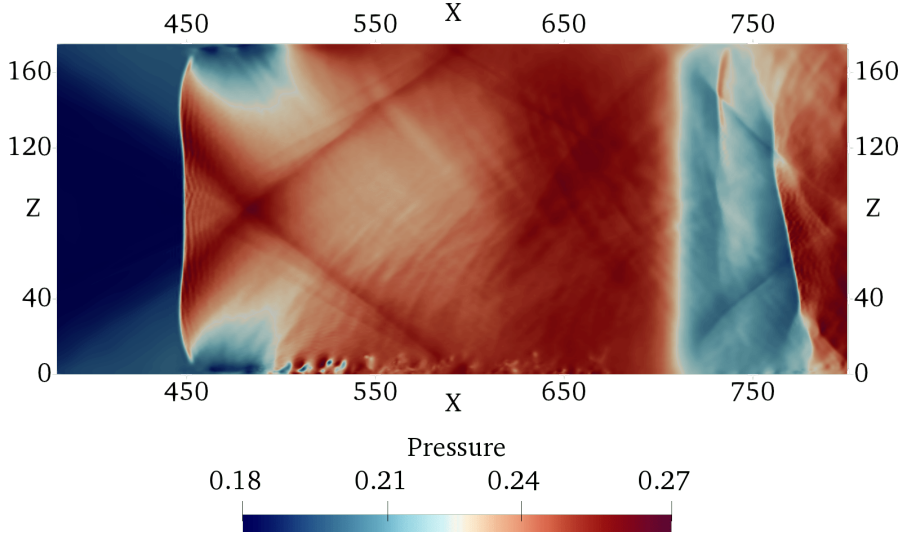


FIGURE 7.8: An $x - z$ plane of instantaneous pressure evaluated at a height of $y = 105$ for $AR = 1$. The main incident shock is visible at $x = 450$ and is seen to be strengthened by compressions from both of the sidewalls. The sidewall compressions coalesce into shock-waves that cross at $x = 475$.

At $x = 710$ the expansion fan generated at the trailing edge of the shock generator is visible. Further downstream there is a shock that is skewed relative to the incident shock. The rotation of the shock is a consequence of the highly unsteady behaviour of the transitional SBLI. As was shown in figure 7.5, the central separation bubble reacts dynamically to oncoming turbulent spots. As the central separation bubble becomes distorted the reflected shock will change in magnitude and orientation. In figure 7.9 (a) instantaneous pressure is shown for an $z - y$ slice at $x = 450$ where the incident shock is at the same height as in figure 7.8. The two crossing shocks discussed in connection with figure 7.8 correspond to the high pressure regions above the incident shock-wave, with a clear conical form. These conical shocks are the ones that originate in both of the upper corners of the duct as the flow is deflected by the shock generator plate.

At both sidewalls there are low pressure regions similar to those observed in figure 7.8. Figure 7.9 (b) shows dilatation rates ($\nabla \cdot \vec{u}$) for the incident shock near the sidewall at a height of $y = 105$. Expansion fans and shock-waves are characterised by regions of positive (red) and negative (blue) dilatation respectively (Johnsen et al., 2010). The incident shock is curved upstream near the sidewall and is weakened relative to the shock closer to the centreline. Above the sidewall boundary layer there is a small region of expansion as the boundary layer thickens due to the swept sidewall SBLI. Figure 7.9 (c) shows that the boundary layer rapidly transitions in this region due to upstream disturbances being amplified by the shock-wave.

Streamwise pressure profiles at a height of $y = 105$ are shown in figure 7.10 at span-wise locations (left) $z = 20$ and (right) $z = 87.5$. Between $x = 350$ and $x = 450$ in figure 7.10 (left) there is a steady pressure rise resulting from the sidewall compression waves seen previously in figure 7.8. There is a well-defined discontinuity at the incident shock-wave, followed by a pressure reduction from the sidewall expansion. At the centreline, figure 7.10 (right) shows a stronger incident shock compared to the one at $z = 20$ and no initial pressure rise from sidewall compressions. Furthermore, there is a pressure peak at $x = 475$ corresponding to the crossing of conical structures highlighted in figure 7.8. Differences are also observed between the two forcing

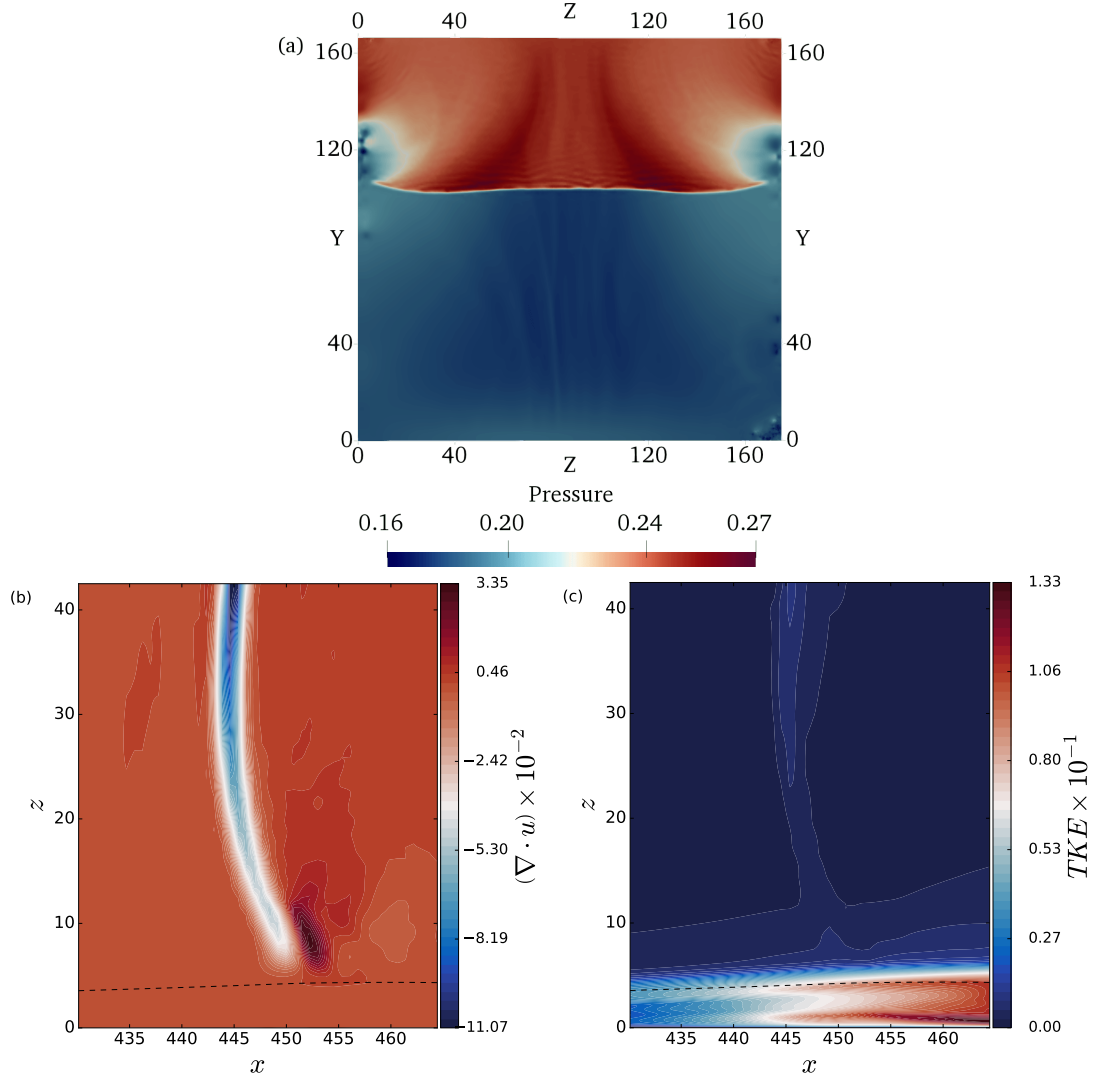


FIGURE 7.9: (a) A $z - y$ plane of instantaneous pressure evaluated at a streamwise location of $x = 450$ for $AR = 1$. Two high pressure regions are observed that originate from the upper corners of the duct at the start of the shock generator. (b) Dilatation rate $(\nabla \cdot \bar{u})$ at a height of $y = 105$ for $AR = 1$. Negative and positive regions of dilatation correspond to shock-waves and expansions respectively. (c) Turbulent kinetic energy at a height of $y = 105$ for $AR = 1$. The sonic line ($M = 1$) is shown in dashed black.

configurations downstream of the main interaction due to the unsteady nature of the transitional SBLI, with a delayed pressure rise near $x = 600$ for the case where all of the walls are tripped.

7.4 Discussion

Oblique shock-wave/boundary-layer interactions (SBLI) have been simulated for rectangular ducts at Mach 2. Disturbances that were added to a laminar flow via blowing/suction caused intermittent turbulent spots that originated in the low-momentum corners of the duct. The transition spread out in a wedge shape across the width of the span. A pair of conical shock-structures form at the swept sidewall SBLIs and reflect laterally within the duct. Their origin and the height dependence of the crossing point was demonstrated. For a half-aspect ratio case, with

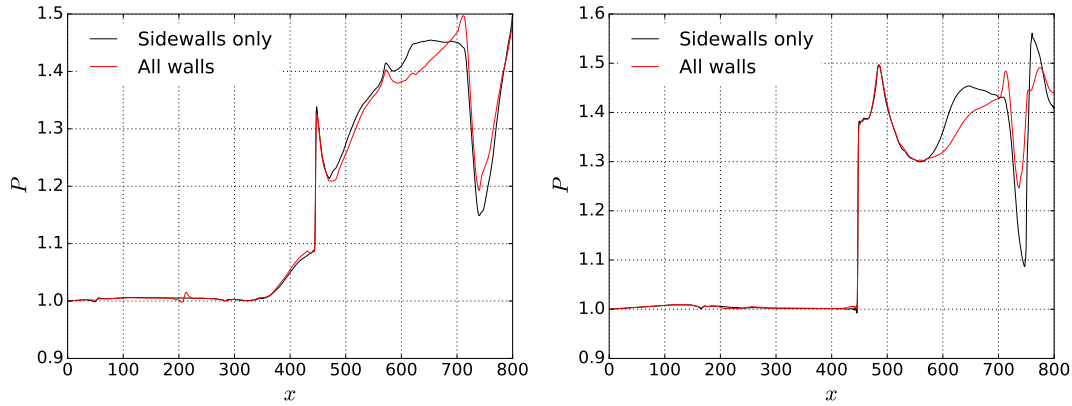


FIGURE 7.10: Streamwise pressure profiles at a height of $y = 105$ for (left) $z = 20$ and (right) $z = 87.5$ at $AR = 1$, normalised by the inlet pressure. Near the sidewall ($z = 20$) there is an expansion directly after the incident shock. At $x = 475$ on the centreline ($z = 87.5$) the crossing point in figure 7.8 is observed as a pressure peak downstream of the incident shock.

disturbances added to both the side and bottom walls, incipient mean separation was not reached for a $\theta_{sg} = 5.0^\circ$ shock generator. When the bottom wall was kept initially laminar a region of shock-induced mean flow recirculation was observed on the centreline. Instantaneous snapshots of the near-wall streamwise velocity showed that the recirculation bubble reacts dynamically to oncoming turbulent spots. Temporary corner separations were seen to develop that caused an earlier streamwise destabilisation of the flow.

A wider aspect-ratio one case was simulated to assess the effect of flow confinement. Both forcing configurations exhibited central mean flow separation. The separation bubble was largest for a case where the bottom wall was kept initially laminar. The strengthened interaction at $AR = 1$ was consistent with previous studies and the laminar duct SBLI of chapter 6. The orientation of the reflected shock was shown to be skewed due to the unsteady response of the separation bubble to intermittent turbulent spots. For the narrower $AR = 0.5$ cases an additional secondary shock was observed downstream of the interaction. This phenomenon was attributed to the start of a weak shock train caused by the reducing duct cross-sectional area. In addition, a low pressure region was observed on the sidewalls and was shown to be related to the transitional behaviour near the sidewall shock. At $AR = 1$ the secondary shock was not observed. The shock structures downstream of the initial SBLI were shown to be highly unsteady and influenced by the transitional state of the boundary layers on the sidewall.

Chapter 8

Conclusions

8.1 Summary of contributions

This section summarises the main contributions of the author during the PhD project, including paper publications, oral conference presentations, and extensive code-development efforts. The main aim in the early stages of the PhD was to develop the open-source code-generation framework ‘OpenSBLI’, which is a modern successor to the existing Fortran-based ‘SBLI’ code from the University of Southampton. OpenSBLI is a high-level Python abstraction that uses the symbolic algebra library SymPy ([Meurer et al., 2017](#)) to create a complete computational fluid dynamics (CFD) solver from a simple Python interface. OpenSBLI applies a ‘separation of concerns’ philosophy ([Ober and Ober, 2017](#)), that decouples the physical problem and numerical algorithms from the low-level parallel code implementation. OpenSBLI is coupled to the Oxford Parallel Structured (OPS) Embedded Domain-Specific Language (EDSL) ([Mudalige et al., 2014, 2015, 2019](#)), which enables users to run the code on multiple massively-parallel High-Performance Computing (HPC) architectures including Graphical Programming Units (GPUs).

The starting point for the software was the initial proof of concept for the methodology ([Jacobs et al., 2017](#)), limited to only subsonic problems with periodic boundaries on uniform meshes. During the PhD project this was rewritten and extended ([Lusher et al., 2018b](#)) to allow the capabilities of shock-capturing, complex geometry, and an extensive set of boundary conditions. As well as the overall design of the software with Dr. Satya P. Jammy, the author was responsible for independently implementing the standard and adaptive Targeted Essentially Non-Oscillatory (TENO) schemes, hybrid methods, time-stepping schemes, boundary-layer initialisations, and wall boundary conditions and closures. Wall-Adapting Local Eddy-Viscosity (WALE) Large Eddy Simulation (LES) models were also added to the OpenSBLI code as part of a collaborative work with Dr Arash Hamzehloo of Imperial College London.

Aside from the code developments, the project has included investigations into the physical aspects of confined shock-wave/boundary-layer interactions (SBLI) in the presence of sidewalls. The simulations in chapters 6 and 7 investigated sidewall effects for both laminar and transitional SBLI in rectangular ducts. The main flow features and complex shock structures were

elucidated, with parametric effects of duct aspect ratio and shock strength among the other contributions. The numerical methods work in chapter 5 investigated the performance of high-order shock capturing schemes for two challenging test cases. The supersonic Taylor-Green vortex and transitional SBLI were selected to assess the performance of schemes on cases that were more representative of common research problems. The final conclusions for all of the physical results are given in section 8.2 of this chapter.

In the remainder of this section, a chronological timeline of the important project milestones is given. An initial version of the Weighted Essentially Non-Oscillatory (WENO) schemes was implemented by the author into the first version of the code (Jacobs et al., 2017) during a summer project with Dr Satya P. Jammy and Prof. Neil D. Sandham in July 2016. A new version of OpenSBLI was started in October 2016 at the start of the research portion of the project. The results for the two-dimensional laminar SBLI from section 4.6 were presented at the ‘*29th International Conference on Parallel Computational Fluid Dynamics (PARCFD)*’, at the University of Strathclyde in Glasgow (May 2017). The results and discussion of the code implementation were subsequently published in Lusher et al. (2018b). A collaborative work investigating the energy-efficiency of different algorithms in OpenSBLI (Jammy et al., 2018), was published in the proceedings of the ‘*7th European Conference on Computational Fluid Dynamics (ECFD 7)*’. A separate conference presentation on the preliminary work on duct SBLI was also presented at this conference. The boundary closures implemented by the author in section 2.6 were used in the work of Jacobs et al. (2018) on spectral error indicators. Work on transitional duct SBLI was presented in Lusher et al. (2018a), in the ‘*Proceedings of the International Conference on Computational Fluid Dynamics (ICCFD10)*’ in Barcelona (June 2018). Additional conference presentations on OpenSBLI include the UK Turbulence Consortium (UKTC) meetings (<https://www.ukturbulence.co.uk/>) at Imperial College London in 2017, 2018, and 2019.

The simulations of transitional rectangular duct SBLI in chapter 7 were first presented at the ‘*ERCOTAC workshop on Direct and Large-Eddy Simulation*’ (DLES12) in Madrid (June 2019). The paper was selected for submission to the special edition of the Flow Combustion and Turbulence journal, and is currently under review. The work on shock-capturing schemes in chapter 5 was presented in Dallas at the ‘*American Institute of Aeronautics and Astronautics (AIAA) Aviation Forum*’ (June 2019), with the accompanying paper Lusher and Sandham (2019a). The physical results for the laminar duct SBLI in chapter 6 are under review in the Journal of Fluid Mechanics (2019). A pre-print of the publication is available in Lusher and Sandham (2019b). Two successful twelve-month computational grants were awarded during this project, for GPU time on the CSD3 supercomputer (<http://csd3.cam.ac.uk>) operated by the University of Cambridge. Finally, the author was invited to be a visiting researcher at NASA Langley and the National Institute of Aeronautics (NIA) in Hampton, Virginia, during February 2020. The research visit included presentation of a seminar on the OpenSBLI framework and duct SBLI. Additional code features were developed to support the collaborative research efforts with Dr. Gary N. Coleman.

8.2 Conclusions

In light of the project aims outlined in section 1.4, the principal conclusions of this thesis can be summarised as follows:

Sidewall confinement effects for shock-wave/boundary-layer interactions

Sidewall confinement of oblique shock-wave/boundary-layer interactions (SBLIs) leads to a significant departure from nominally two-dimensional predictions. Three-dimensional simulations with an infinite-span assumption (quasi-2D), are not valid outside of scenarios in which the aspect ratio of the test section is large. For the laminar cases in chapter 6, an aspect ratio of four was required before nominally two-dimensional centreline behaviour was recovered. Even at this large aspect ratio, a discrepancy of 9% was observed in the length of the central separation bubble compared to infinite span cases. Compared to turbulent duct SBLI such as Wang et al. (2015), laminar duct SBLI were shown to induce much larger regions of flow-reversal in the centre, corners, and sidewalls of the duct. These separation regions were observed for far more modest flow deflections ($0.5^\circ \leq \theta_{sg} \leq 2.5^\circ$) than typically used in turbulent studies ($\theta_{sg} \geq 8^\circ$). This is due to the absence in laminar cases of efficient turbulent mixing that transfers high-momentum fluid towards the wall.

- The corner separations were observed to be very thin regions of flow-reversal elongated in the streamwise direction (figure 6.6). Importantly, significant corner separations as in the experimental turbulent works of Xiang and Babinsky (2018, 2019), could not be identified in the present study. This is, however, consistent with the numerical studies of turbulent duct SBLI by Wang et al. (2015). The corner region was observed to separate first (see figure 6.7 (b)).
- For the baseline duct of aspect ratio one, sidewalls led to a strengthening of the main SBLI and a subsequent 31% increase in the central separation bubble length compared to infinite-span cases. It was emphasised that this is strongly dependent on the duct aspect ratio and length of the shock generator ramp. Weaker initial flow-deflections were simulated, with incipient separation present even for the smallest $\theta_{sg} = 0.5^\circ$ laminar case.
- For the largest aspect ratio of four, the sidewall influence was shown to extend to a distance of 30% of the span width away from either of the sidewalls (figure 6.15 (b)). For aspect ratios of $AR \leq 2$, the sidewall influence permeated throughout the entire spanwise extent of the domain.
- The trends shown in a plot of the central separation bubble length at different duct aspect ratios (figure 6.16 (a)), were in good agreement with the experimental findings of Babinsky et al. (2013); Xiang and Babinsky (2019). For duct aspect ratios below unity, a suppression of the central separation bubble was observed compared to the infinite-span result. The separation length was almost identical for aspect ratios of one and two, both being approximately 31% longer than cases without sidewalls. At larger aspect ratios the separation bubble length tends towards the nominally two-dimensional result as it should.

- In addition to the aspect ratio of the duct, it was demonstrated that the length of the shock-generator ramp plays a crucial role in determining the extent of flow separation for confined oblique SBLI. This effect was due to variation in the impingement point of the trailing-edge expansion fan from the end of the shock-generator ramp.
- The expansion fan effect was demonstrated by simulating a range of shock-generator ramps ($L_{sg} \in [200, 350]$) comparing two-dimensional results to three-dimensional simulations with sidewalls. In each of the cases the expansion fan impinged far downstream of the central separation bubble. No dependence on the shock-generator length was observed for the 2D simulations (figure 6.4). In contrast, simulations with sidewalls showed variation of the central separation bubble length for all shock-generators tested (figure 6.5).
- To further investigate this dependence on shock-generator length, a longer laminar duct was simulated in figure 6.20 (a)-(f), for both a short ($L_{sg} = 200$) and long ($L_{sg} = 600$) shock-generator ramp. Significant suppression of the interaction and all of the separation regions was observed for the short ramp. The longer ramp duct had far larger regions of flow separation, and the flow was observed to transition to turbulence downstream of the SBLI. This is in contrast to the shorter ramp duct, in which the flow remained laminar throughout.
- The flow topology of the laminar duct SBLI was discussed in detail using a combination of critical point theory and identification of the complex three-dimensional shock structures. The laminar solutions were found to follow the ‘owl-like’ topologies of the first kind formalised by [Perry and Hornung \(1984\)](#) (figure 6.10). All of the main shock structures were identified (figure 6.12) through numerical schlieren. Notably, conical shock-waves originating from the intersection of the shock-generator and sidewall were identified to cross on the centreline and reflect laterally within the duct. Figure 6.13 clarified this by showing the development of the conical shocks at a range of streamwise locations.
- Substantial efforts were made to identify the strong compression waves proposed by [Babinsky et al. \(2013\)](#); [Xiang and Babinsky \(2019\)](#), in their mechanism of the sidewall strengthened SBLI. No significant corner compressions could be observed for the present laminar cases. While shocks were observed to cross the centreline with trends consistent with their findings, it was clearly shown that these crossing structures originated from far above the bottom wall corners of the duct (figure 6.17). The location of the crossing was shown to be dependent on the height of the y plane at which they were observed. The strengthening of the interaction was instead attributed to reinforcement of the incident shock by sidewall compression waves.
- A stronger laminar interaction of $\theta_{sg} = 2.5^\circ$ was simulated for the baseline $AR = 1$ duct to observe the limiting behaviour of large shock-induced separation bubbles. The flow began to transition downstream of the SBLI in the vicinity of four high-speed streaks. Two of these streaks were observed near the corner regions and one either side of the centreline (figure 6.19). Despite refinement of the grid by 100%, a laminar solution was not obtained at this higher shock strength. Similarities were drawn to the transition in the longer duct cases in figure 6.20. Although beyond the scope of the current work, it was proposed that streak instability may play a role in the transition process of duct SBLI.

- The laminar duct was extended to contain time-dependent forcing strips to generate a transition to turbulence upstream of the $\theta_{sg} = 5^\circ$ SBLI at $AR = 0.5$. Transition was observed to develop as turbulent spots in the corner regions and spread out across the span in a wedge shape (figure 7.2).
- Two different forcing configurations were compared to assess the effect of disturbance locality on the transitional SBLI. For a case where the bottom wall was left initially laminar, a small oval region of flow separation was bounded by the developing turbulence from the corners (figure 7.2). The case where all walls were forced led to an absence of mean flow-separation in the duct at this interaction strength.
- Snapshots of the instantaneous flow (figure 7.5) showed that the central separation bubble responds dynamically to the oncoming turbulence. The separation bubble shifts laterally depending on the extent of the transition in that time instance. Although no mean flow separation was observed in the corners at this interaction strength, significant corner separations were shown to develop momentarily before being removed by the oncoming intermittent turbulence. Regions of flow separation were also observed to accelerate the transition process and cause a more energetic breakdown.
- A comparison was made for the transitional cases to a wider duct of $AR = 1$. Consistent with the laminar results of chapter 6, the wider duct showed a stronger interaction at $AR = 1$, with large regions of flow-reversal observed for both of the forcing configurations.

Low-dissipative high-order shock-capturing schemes

A selection of high-order Weighted and Targeted Essentially Non-Oscillatory (WENO/TENO) shock-capturing schemes were implemented into the OpenSBLI code to perform the simulations in this thesis. Two challenging supersonic test cases have been proposed to test the efficacy of shock-capturing schemes. Each of the cases include the interaction of both compressible turbulence and shock-waves, which are often tested separately in numerical methods literature. As shock-capturing and hybrid methods often require problem-dependent tuning of parameters, it is important to design test cases that test both of these contrasting requirements simultaneously.

- In the first case, a compressible Taylor-Green vortex (TGV) case was demonstrated to supersonic Mach numbers and was shown to be more appropriate for assessing the effectiveness of low-dissipative shock-capturing schemes than the classic subsonic problem. Compared to previous work on compressible TGV at $Re = 400$ (Peng and Yang, 2018), the present contribution was performed at $Re = 1600$ to ensure that a wide enough range of turbulent scales were present to suitably test the numerical schemes.
- After demonstrating grid independence for a Mach 1 flow, the multiple complex shock structures were shown for the Mach 1.25 case (figure 5.5). At supersonic Mach numbers it was shown that dilatational contributions become important for the viscous dissipation rate, as shock-waves develop in the locally accelerating regions of the flow. Two distinct peaks of dilatation were observed at $t = 2$ and $t = 7$ during the first half of the simulation (figure 5.4).

- Unlike the subsonic ($M = 0.1$) problem, the supersonic TGV no longer had a monotonically decreasing kinetic energy distribution. In the early stages of the simulation, the conversion of internal energy to kinetic energy was observed to exceed the viscous dissipation rate. A delayed peak dissipation rate was also observed as the Mach number was increased. This was attributed to compressibility effects that decrease the growth rate of instabilities as the Mach number is increased. This was suggested due to similar observations reported previously for compressible mixing-layers and turbulent spots (Vreman et al., 1996; Sandham, 2016).
- A clear hierarchy of schemes was established (figure 5.12), with low-dissipative TENO schemes (Fu et al., 2016, 2017) shown to perform much better on coarse grids compared to conventional WENO approaches. In particular, the adaptive TENO-6A schemes of Fu et al. (2018) that utilise a shock-sensor within the scheme, were shown to give comparable results to the hybrid central-WENO methods.
- The second test case performed the same scheme comparison for a forced Mach 1.5 transitional shock-wave/boundary-layer interaction. While a similar hierarchy of schemes was observed, an undesirable scheme dependence on the original disturbance amplitude was also recorded (figure 5.17). For this sensitive transitional case, the differences in initial disturbance amplitude led to two different separation bubble lengths. This effect was attributed to an improper interaction of the schemes with the wall-localised forcing.
- While the adaptive TENO-6A scheme was very effective in resolving compressible turbulence, it was demonstrated that care must be taken when optimising the values of weights within the scheme. For overly optimized weights (figure 5.20), degradation of the scheme's shock-capturing ability was shown.

Code-generation techniques for computational fluid dynamics

An open-source compressible CFD solver ‘OpenSBLI’ has been developed during the PhD project alongside Dr Satya P. Jammy. The code is a symbolic high-level Python abstraction that generates stencil-based finite-difference approximations on structured meshes. Chapter 3 discussed the implementation of the code in great detail. The OpenSBLI framework generates a complete C code solver written in the OPS (Mudalige et al., 2014, 2015, 2019) domain-specific language. This allows the simulation code to be executed on various massively-parallel architectures including GPUs.

A suite of validation and verification cases was presented for OpenSBLI in chapter 4. Each of the test cases were selected to demonstrate specific features of the code implementation. The ‘separation of concerns’ approach (Ober and Ober, 2017) splits the physical modelling of a problem and its numerical algorithms from their parallel implementation. Performance of the code has been shown externally to be equivalent or better than hand-written versions of the same application (Mudalige et al., 2015, 2019). Additional performance comparisons have been given in this thesis in the context of the simulations described throughout.

- The simulations contained in the final three physical chapters have demonstrated that code-generation techniques are feasible for performing ILES/DNS of challenging flow problems.

Rather than demonstrating the code for simple ‘toy’ problems, the code developed in this project has already been applied to a number of challenging research questions (Lusher and Sandham, 2019a,b; Lefieux et al., 2019). Furthermore, collaborations are ongoing with researchers in other institutes including Imperial College London, NASA Langley, JAXA, and ONERA, which will continue after completion of the project.

- The OpenSBLI code has also provided benchmark data for national HPC systems within the UK, and emerging architectures as in McIntosh-Smith et al. (2019). Efforts have been made to support other users of the code in the future, culminating in the OpenSBLI user manual that is partially included in the appendix of this thesis. The deployment of the code as a teaching asset for a turbulence module at the University of Southampton, has demonstrated the wider outreach impact of the code beyond the personal research interests of this project.

8.3 Outlook

8.3.1 Limitations of the current study

Some of the limitations of the present study are summarised in this section.

- The laminar duct SBLI in chapter 6 made the choice of a fixed Reynolds number based on the earlier two-dimensional study of Katzer (1989). While parametric studies of duct aspect ratio, shock-generator length, and incident shock strength were performed, the effect of Reynolds number was not investigated.
- A secondary limitation of the laminar duct SBLI was the fixed ratio of inlet boundary-layer thickness to duct height δ/L_y . This was an intentional design choice to narrow the parameter space. An assumption was also made that the inlet boundary-layer thickness was the same on each of the duct walls.
- While the three-dimensional duct geometry with four no-slip walls was a significant improvement over simulations assuming an infinite-span, the leading edge of the duct was not modelled in this study. In real-world supersonic engine intake applications, a leading edge shock-wave would be present from the initial flow deflection. In practical applications, geometries are often designed to ensure this leading edge shock does not enter the duct. Nevertheless, in future studies it would be more appropriate to model the entire intake.
- Although the laminar duct SBLI has given valuable insight into the shock-structures and flow features within a sidewall-confined SBLI, almost all practical applications would be at higher Reynolds numbers and turbulent. As the available computational power increases in the future, DNS of turbulent ducts and full engine intake configurations will become possible. As previously noted however, it is more difficult to identify the shock-structures when turbulence is present.
- A stronger $\theta_{sg} = 2.5^\circ$ interaction was simulated, and was shown to deviate from a laminar solution. The mechanism for transition downstream of the duct SBLI was beyond the scope of the present study.

- SBLI with transition upstream of the interaction in chapter 7 was performed as an implicit LES. Finer grids would be required for DNS. Furthermore, the time averaging periods of this transitional study were relatively short, owing to the expensive nature of the calculation. Longer averaging periods would generate the data required to extract the characteristic frequencies of the interaction.
- The transitional SBLI numerical test case proposed in chapter 5 highlighted an undesirable scheme dependence on the generation of the initial forcing. Different magnitude disturbances were generated from the blowing/suction effect between schemes. As any transitional study is naturally sensitive to the magnitude of initial disturbances, this made numerical scheme comparison difficult. A modified forcing method based on a freestream acoustic source is expected to avoid the documented issue.
- While Targeted Essentially Non-Oscillatory (TENO) schemes were shown to offer significant improvements over more established schemes, more work is needed to improve their performance for compressible turbulence. It was also demonstrated that efforts to optimise these schemes can lead to a deterioration of their ability to robustly capture shock-waves.

8.3.2 Scope for future work

While the thesis has been successful in extending the current literature on confined SBLI, shock-capturing schemes, and code-generation techniques, there are several avenues for possible future work.

Shock-wave/boundary-layer interactions

- Laminar duct SBLI could be performed over a wider parameter space. These studies should focus on Reynolds number scaling and the ratio of inlet boundary layer thickness to duct height δ/L_y . The effects of non-isothermal and cold wall conditions could also be investigated.
- Further work is required to resolve the apparent disparity between numerical and experimental work on the sidewall influence on SBLIs. Central to this aim is the identification of the dominant structures that modify the main interaction. While experimental studies have attributed this to the crossing location of corner compression waves relative to the central separation (Babinsky et al., 2013; Xiang and Babinsky, 2019), this study and the numerical work of Wang et al. (2015) were unable to detect significant corner waves. It is possible that this is due to the experimental limitation of having to infer shock-structures through the bottom-wall imprint of pressure-sensitive paint maps. Direct numerical simulation in contrast has access to full three-dimensional flow data in time, to detect flow features throughout the entire duct.
- There are several other differences to experiment which could be investigated in future studies. One notable example is the role of sidewall gaps between the shock-generator plate and the sidewall. Many experimental configurations feature a partial-span shock-generator plate that does not extend all of the way to the sidewalls. As such, it is expected that the swept SBLI structures in the present study would become less important. Simulations

that included a partial-span shock-generator and sidewall gaps would help to address the aforementioned discrepancy between experiment and simulation in this field.

- Having obtained a number of laminar duct base flows, the natural extension to this project would be to perform a global stability analysis. Identifying the least-damped global mode would help to give insight into the transition mechanism for laminar duct SBLLI.
- Other modal decomposition techniques such as Dynamic Mode Decomposition (DMD) or Proper Orthogonal Decomposition (POD) could be applied to the transitional cases to extract coherent structures.

Low-dissipative shock-capturing schemes

- Despite the progress in numerical methods in recent years, more work is required for low-dissipative shock-capturing schemes. Optimisation of TENO schemes to balance the scale-resolving ability of the scheme to its shock-capturing capabilities would be a valuable avenue for future study. The two numerical test cases proposed in this thesis would be suitable problems to test both of these requirements simultaneously.
- The present thesis has focused on WENO/TENO methods within a Local Lax-Friedrichs (LLF) flux-splitting in characteristic space. Alternative flux-splitting methods such as the AUSM or HLLC approaches would be valuable contributions to the code base. Furthermore, it would be interesting to test the shock-capturing schemes applied directly on conservative variables as a computationally cheaper option compared to the characteristic formulation.
- It would be interesting to compare the existing capabilities to the non-linear filtering approaches outlined in [Yee and Sjögren \(2018\)](#) among others. Rather than computing the flow field each sub-stage of the Runge-Kutta scheme, the dissipative WENO contribution is subtracted from the flux at the end of each time-step as a filtering method. These methods may prove to be computationally more efficient as they only require computation of the WENO terms at the end of each time-step.
- Within a finite-difference framework, much of the numerical literature to date has focused on the development of schemes for uniform meshes. Adding metric terms to the derivatives can lead to a performance deterioration of the scheme and impact stability. More work is needed to quantify the effect of non-uniform meshes, especially in the context of high-order shock-capturing schemes.

OpenSBLLI code-generation framework

- All of the simulations performed in this report were on a single simulation block. Future work is needed to develop the multi-block capabilities of the OpenSBLLI code. This will enable the study of more complex flow geometries such as aerofoils. An initial proof-of-concept multi-block version has been developed previously, but was limited to subsonic flow with central differencing. The main challenge for multi-block is demonstrating that it works for supersonic cases with shock-capturing schemes.

- The major research simulations in this thesis were typically run on 8-16 Nvidia P100 GPUs. The largest simulations for the compressible Taylor-Green vortex and scaling tests used up to 64 of these GPUs. Going forward, the scaling and performance of the OPS system should be tested and optimised on much larger GPU-configurations. One of the major bottlenecks is the MPI communication between multiple GPUs [Mudalige et al. \(2019\)](#). It is expected that as larger exascale machines become available, more work is needed to efficiently utilise these resources.
- Throughout the project it became evident that memory access is one of the main bottlenecks for performance in computational fluid dynamics codes. Algorithms that vary the amount of computational and memory intensity have previously been tested in OpenSBLI ([Jammy et al., 2016](#)). Future work should expand on these themes for the current version of the code to improve runtime performance. Additionally, further development of numerical schemes with smaller stencils as shown in section 2.6, are well suited to efficient computation within OPS. Roof-line models ([Williams et al., 2009](#)) can be applied to assess which kernels within the code are memory or compute bound.
- One feature missing from the code is non-reflecting characteristic boundary conditions. These boundary conditions should be added to the code to minimise the reflection of disturbances within the domain. Alternatives to non-reflecting characteristic boundary conditions include the use of sponge zones which have previously been utilised in OpenSBLI. A comparison of non-reflecting characteristics to sponge zones would be useful, especially for disturbance-sensitive applications in the field of computational aero-acoustics.
- At present, the OpenSBLI code is explicit in both space and time. While explicit methods achieve excellent parallel performance, the strict restrictions on time-step values can require excessively long computation times on the finest meshes. Furthermore, compact spatial-difference schemes are well-known to offer improved resolution compared to their explicit counterparts. Future work could involve the implementation of implicit methods in OpenSBLI, which would require the use of a tri-diagonal solver within OPS.
- The present study has focused on solution of the compressible Navier-Stokes equations. The high-level Python interface is ideal for extension to more complicated flow scenarios. Examples of possible extensions would include multi-phase flow and chemical reactions for high-enthalpy hypersonic flows, the equations for magnetohydrodynamics (MHD) to model plasmas, and fluid-structure interactions.
- A finite-volume version of the OpenSBLI code would complement the existing solvers by applying a lower-order finite volume formulation to target more complex flow geometries.

Appendix A

Installation guide

OpenSBLI requires a Python 2.7 install for code generation and the Oxford Parallel Structured software (OPS) library to build executables. The guide is for Linux platforms only, it hasn't been tested on OSX/Windows. Section 1 outlines the libraries that must be installed to use OpenSBLI + OPS, section 2 explains the build process of OPS, section 3 covers how to generate a code within OpenSBLI and section 4 gives the steps to compile and run the code. Section 5 has some additional information that may be applicable if running on an HPC cluster.

The installation of libraries will depend on whether you are using a local machine or an HPC cluster. If you are intending to use OpenSBLI on an HPC cluster you will need to load suitable system modules and point OPS to them via the environment variables outlined in section A.2. MPI and HDF5 libraries must have been compiled with the same compiler or you will face errors during compilation of OPS executables. If you are not sure where the modules are located, please contact your system administrator. **Important note: SymPy version 1.1 is required, newer versions are currently not supported.** The requirements are:

OpenSBLI requirements (Python 2)

SymPy version 1.1 : Symbolic Python library used by OpenSBLI
Matplotlib : Python plotting library
h5py : Python library to read in HDF5 output files for plotting
NumPy : Numerical Python library used in some plot scripts
SciPy: Scientific Python library used in some OpenSBLI classes

OPS requirements

git : Version control software used to clone the repository
A C compiler : (gcc, icc, Cray, ..)
An MPI distribution : (OpenMPI, MPICH, Intel MPI, ..)
HDF5 libraries : Used for I/O within OPS

CUDA/OpenCL libraries : Required only if you intend to run on GPUs

Important note: Codes can be generated with OpenSBLI on a local machine (laptop) and copied onto another machine to be translated, compiled and run with OPS. While you can install both OpenSBLI and OPS on an HPC cluster, it is not necessary to have both on the same machine. OPS however must be installed (and built) on the machine you wish to run simulations on.

A.1 Installation of libraries on an Ubuntu local machine

****Ignore this section if installing OpenSBLI on an HPC cluster****

This section was tested on a clean Ubuntu 18.04 LTS installation, similar packages can be found for other Linux distributions. Python packages are installed with pip, if pip is not already present on your system it can be installed with e.g. `sudo apt-get install python-pip`, on Ubuntu/Debian based systems. Alternatively, software distributions like Anaconda can be used to install the Python libraries. The guide is for installation on a local machine with the GNU C compiler (gcc) and OpenMPI.

1. Install OpenMPI and HDF5 libraries, git and pip/tk

```
sudo apt-get install python-pip python-tk
sudo apt-get install libhdf5-openmpi-dev
sudo apt-get install git
```

2. Install the required Python packages for this user. Note that SymPy 1.1 is required, more recent versions will cause issues:

```
pip install --user scipy numpy h5py matplotlib
pip install --user sympy==1.1
```

A.2 OPS installation

In this section the installation of the OPS library is explained. **The paths that you put in step 2 will change depending on whether you are on a local machine or an HPC cluster.**

1. Clone the OPS library:

```
git clone https://github.com/OP-DSL/OPS.git
```

2. Export OPS environment variables:

Edit your `~/.bashrc` to include the lines:

Local Ubuntu machine:

```
export OPS_INSTALL_PATH=/home/<username>/OPS/ops/
export OPS_COMPILER=gnu
export OPS_TRANSLATOR=/home/<username>/OPS/ops_translator/c/
export MPI_INSTALL_PATH=/usr/
export HDF5_INSTALL_PATH=/usr/lib/x86_64-linux-gnu/hdf5/openmpi/
export CUDA_INSTALL_PATH=/path/to/cuda-install/
export USE_HDF5=1
```

HPC cluster: (fill these in)

```
export OPS_INSTALL_PATH=/home/<username>/OPS/ops/
export OPS_COMPILER=gnu/cray/intel (pick one)
export OPS_TRANSLATOR=/home/<username>/OPS/ops_translator/c/
export MPI_INSTALL_PATH=/path/to/mpi-install/
export HDF5_INSTALL_PATH=/path/to/hdf5-install/
export CUDA_INSTALL_PATH=/path/to/cuda-install/
export USE_HDF5=1
```

To make these changes active in the current shell type: `source ~/.bashrc`

3. Build the OPS libraries:

```
Change directory to ~/OPS/ops/c/
make core seq mpi hdf5_seq hdf5_mpi
```

If you are using CUDA/OpenCL for GPUs you will also have to build:

```
make cuda mpi_cuda opencl mpi_opencl
```

You may see some warnings that can be ignored, the `./lib/` folder should now contain some OPS libraries. If you encounter errors at this point it will be due to improper configuration of the modules/environment variables.

A.3 OpenSBLI installation and code generation

1. Clone the latest OpenSBLI with your username and switch branch:

```
git clone https://<your-username>@bitbucket.org/spjammy/opensbliweno.git
cd opensbliweno
git fetch && git checkout internal_dev
```

2. Export the OpenSBLI folder to your PYTHONPATH in `~/.bashrc`:

```
export PYTHONPATH=$PYTHONPATH:/home/<username>/opensbliweno/
source ~/.bashrc
```

3. Generate a code in OpenSBLI (e.g. Taylor-Green vortex):

```
cd ~/opensbliweno/apps/taylor_green_vortex/
python taylor_green_vortex.py
```

You should now have four new files:

`opensbli.cpp`, `opensbli_block00_kernels.h`, `defdec_data_set.h`, `bc_exchanges.h` and some LaTeX files.

At this point the files may be copied onto another machine (if desired) where you have OPS installed. The OPS translation and compilation steps are described in the next section.

A.4 Compiling and running the code

1. Translate the code using OPS to generate parallel versions:

```
python $OPS_TRANSLATOR/ops.py opensbli.cpp
```

You should now have folders for CUDA, MPI and so on, plus an `opensbli_ops.cpp` file. You need to apply the translation step each time you make changes to the `opensbli.cpp` code generated by OpenSBLI. Ignore the clang-format warning, this is just an optional code indentation formatter and has no effect on the simulation.

2. Copy a Makefile into the directory and build the executable for the desired architecture (MPI, CUDA, OpenMP etc):

```
cp ../Makefile ./
make opensbli_mpi
```

3. Run the executable for number of processes 'np':

```
mpirun -np 4 ./opensbli_mpi
```

The simulation will now run, when finished the output data is written to an `opensbli_output.h5` HDF5 file for post-processing. If you wish to run the simulation again remove the HDF5 file from the directory before repeating the above steps as existing files won't be overwritten.

A.5 Things to note for HPC clusters:

1. For different compilers the `OPS_COMPILER` variable set in section A.2 must be changed and you may need to modify some of the Makefiles. The easiest way is usually to export `cc` and `CC` to where your compiler is located, but you may need to explicitly edit the Makefiles to use `icc/icpc` for example. In the current OPS (02/2019) the Makefiles are all located in `/OPS/makefiles/`
2. If there is no suitable HDF5 available on the system you will have to build it yourself:

Obtain the source: <https://support.hdfgroup.org/HDF5/release/obtainsrc.html>

Extract the file, change to the directory and configure the build.

For building parallel HDF5:

```
./configure --enable-parallel cc=/path/to/mpicc CC=/path/to/mpicc
```

```
make
```

```
make install
```

3. Depending on the machine you may need to add your HDF5/CUDA library locations to your LD_LIBRARY_PATH environment variable.
4. Job submission scripts will often need module load commands in them as the compute nodes are separate to login nodes. In addition LD_LIBRARY_PATH may need to be exported in the submission script with the HDF5/CUDA library locations.
5. If using gcc you may need to add:

```
-fpermissive to CCFLAGS/CXXFLAGS in OPS/makefiles/Makefile.gnu
```

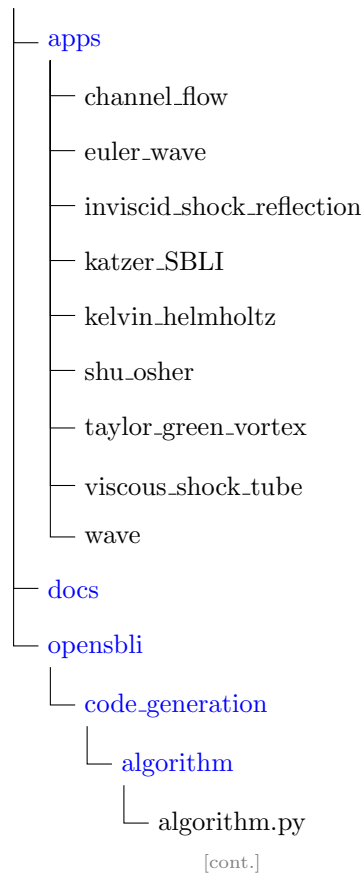

Appendix B

Description of the source code files

B.1 Source code directory structure

The directory tree below shows the structure of the OpenSBLI source code files. Folders are highlighted in blue and files in black.

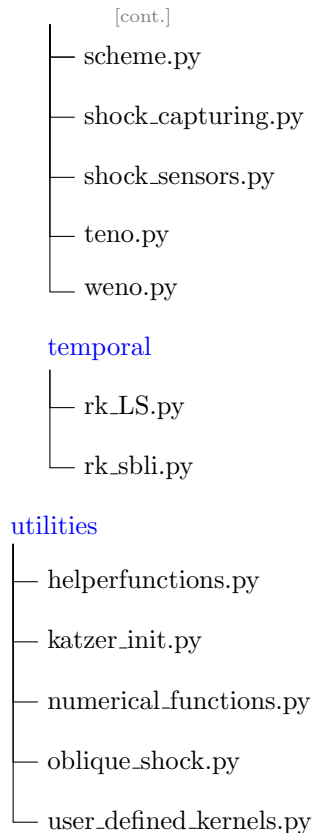
OpenSBLI base directory



continued

- [cont.]
- └─ latex.py
- opsc.py
- core
 - └─ bcs.py
 - └─ block.py
 - └─ datatypes.py
 - └─ grid.py
 - └─ io.py
 - └─ kernel.py
 - └─ opensblifunctions.py
 - └─ opensbliobjects.py
 - └─ parsing.py
- equation_types
 - └─ metric.py
 - └─ opensbliequations.py
- initialisation
 - └─ gridbasedinit.py
- optimisations
 - └─ optimisations.py
- physical_models
 - └─ euler_eigensystem.py
 - └─ ns_physics.py
- post_process
 - └─ post_process_eq.py
 - └─ removehalos.py
- schemes
 - └─ spatial
 - └─ averaging.py
 - └─ hybrid.py
- [cont.]

continued



B.2 Contents of the source code files

This is a brief description of each of the source code files in alphabetical order.

1. **algorithm.py:** Placement and ordering of the different solution components in the output code. Defines the iteration and sub-stage loops and their respective components.
2. **averaging.py:** Simple (mean) and Roe averaging routines for the characteristic decomposition. Used for the shock-capturing schemes.
3. **bcs.py:** All of the boundary conditions defined in OpenSBLI. Also contains the one-sided boundary derivative schemes.
4. **block.py:** The simulation block contains grid information and links all of the main OpenSBLI components together.
5. **datatypes.py:** Floating point precision definitions.
6. **euler_eigensystem.py:** Definitions of the transformation matrices used in the characteristic decomposition of the Euler equations. Used for improving the robustness of the shock-capturing schemes.
7. **grid.py:** Defines the parameters related to the grid (sizes, ranges) and the declaration of global work arrays.

8. **gridbasedinit.py**: The class used to initialise the solution at $t = 0$. Equations given to the initialisation class are calculated once at the start of the simulation.
9. **helperfunctions.py**: Miscellaneous functions frequently used throughout the internal structure of OpenSBLI. It also contains routines for adding the problem-specific HDF5 metadata required by OPS for reading/writing files.
10. **hybrid.py**: Routines for hybrid central-WENO schemes via a shock sensor.
11. **io.py**: The main input/output class controlling reading and writing of simulation data within OpenSBLI.
12. **katzer_init.py**: Numerical similarity solution of the compressible boundary-layer equations. Creates laminar boundary-layer profiles for wall-bounded domain initialisation.
13. **kernel.py**: The main class for OpenSBLI kernels which perform a computation for a given kernel range in parallel.
14. **latex.py**: Functionality to convert symbolic equations in OpenSBLI to TeX format. The output can be compiled with pdflatex to produce a pdf file.
15. **metric.py**: Classes to perform the metric transformation to generalised curvilinear coordinates.
16. **numerical_functions.py**: Numerical spline interpolation routines used for the similarity solution. These functions are for creating initial profiles in Python and are not part of the output simulation.
17. **ns_physics.py**: A physics object containing frequently used definitions related to the compressible Navier-Stokes equations.
18. **oblique_shock.py**: Oblique shock jump relation calculator.
19. **opensbliequations.py**: The main file for defining equations in OpenSBLI. Contains the routines for the SimulationEquations and ConstituentRelation classes.
20. **opensblifunctions.py**: Definitions of common derivative objects (Central, WENO, TENO) and the EinsteinStructure for indexing symbolic quantities in OpenSBLI.
21. **opensbliobjects.py**: Definitions of various symbolic objects within OpenSBLI that can be used to build up symbolic equations.
22. **opsc.py**: This controls the conversion of symbolic objects to C code compliant with the OPS domain-specific language.
23. **optimisations.py**: Optimisations for reduced work array usage.
24. **parsing.py**: Routines to parse equations written as Python strings into equations comprised of symbolic objects.
25. **post_process_eq.py**: A class to create a post processing OPS code.
26. **removehalos.py**: Functionality to strip halo points from OpenSBLI output HDF5 files.

27. **rk_LS.py**: Low-storage explicit Runge-Kutta schemes in the two-register Williamson formulation. Contains schemes for 3rd, 4th order and 3rd order with strong-stability-preservation (SSP).
28. **rk_sbli.py**: Low-storage explicit 3rd order Runge-Kutta scheme in the form used in the old SBli Fortran code.
29. **scheme.py**: Contains the central differencing scheme and coefficient generator for the finite difference stencils.
30. **shock_capturing.py**: All of the common functionality for the characteristic decomposition shared between the various WENO/TENO shock-capturing schemes.
31. **shock_sensors.py**: The modified Ducros shock sensor.
32. **teno.py**: Routines for the Targeted Essentially Non-Oscillatory (TENO) shock-capturing schemes.
33. **user_defined_kernels.py**: Functionality to generate one-off OpenSBli kernels. Useful for non-standard processing or filtering operations that the user may want to add to their simulation.
34. **weno.py**: Routines for the Weighted Essentially Non-Oscillatory (WENO) shock-capturing schemes. Including the WENO-JS and WENO-Z formulations.

Appendix C

SymPy functionality for OpenSBLI development

To be able to develop new features for OpenSBLI, a good knowledge of Python and the SymPy library is required. The best resource is the SymPy online documentation: <https://docs.sympy.org/1.1/index.html>. Some commonly used functionality in OpenSBLI is listed here with examples.

C.1 Pretty printing of equations: pprint

At any point within the code generation process the user is able to take a list of symbolic equations and print them to screen to be checked. The best way to do this is with SymPy's `pprint` function. Many of the classes in OpenSBLI will store equations as an `equations` attribute to an object, which can be iterated over and printed to screen:

```
from sympy import pprint
for equation in kernel.equations:
    pprint(equation)
```

The pretty print function can be used throughout the entire source-code and in the problem set up files.

C.2 Creating symbolic objects: symbols

There are three main ways of generating symbolic objects in OpenSBLI. The first is to simply call one of the OpenSBLI object classes with a string as a name:

```
p = DataObject('p')
```

This creates a `DataObject` which will later be converted into a `DataSet` defined on a block. Note that the variable `p` on the left hand side is simply a Python variable used to refer to this object during code generation, it has no effect on the output code. The name passed to the `DataObject` class as a string will be written out in the simulation code, so ensure that this variable has been defined elsewhere in the script. If you have already created a `SimulationBlock` you can create a `DataSet` directly on that block with:

```
p = block.location_dataset('p')
pprint(p)
p_B0[0,0,0]
```

Note that the `DataSet` has indices as it is an indexed object with i indices for the i number of dimensions in the problem. The `DataSet` has also been given a `'_B0'` suffix to denote that it is defined on a block numbered 0. The third method is a quick way to create several symbols of the same type using the `symbols` class of SymPy.

```
p, u, v, w = symbols('p u v w', **{'cls': DataObject})
```

The type of object created here could be `DataObject`, `GridVariable` or `ConstantObject` depending on the application.

C.3 Off-setting an indexed object

OPS uses a stencil-based approach for data access. As the `ops_par_loop` iterates over the discrete grid points, the current grid point being evaluated is always denoted as `[0,0,0]` in three dimensions. To build finite-difference approximations we need the ability to access data that is offset from the `[0,0,0]` location. OpenSBLI contains a function to let users index `DataSets` for a given number of grid points away from the current location in a certain direction. For example to index the above `DataSet` in the $i = 1$ dimension by three grid points:

```
from opensbli.utilities.helperfunctions import increment_dataset
p = block.location_dataset('p')
pprint(p)
p_B0[0,0,0] # The base [0,0,0] location for a generic stencil
direction, increment = 1, 3 # index direction 1 by 3 grid points
p_shifted = increment_dataset(p, direction, increment)
pprint(p_shifted)
p_B0[0,3,0]
```

This functionality is frequently used when developing numerical schemes or boundary conditions, where the expression requires data offset from the current `[0,0,0]` grid point. For example to create a 4th order central difference approximation for a given direction, the user would create a list `[-2, -1, 1, 2]` and build the finite-difference approximation by looping over the offsets with four applications of the `increment_dataset` function.

C.4 Substitution of symbolic quantities

SymPy has various methods for substituting terms into existing symbolic expressions. Most of these rely on setting up either a Python dictionary or tuples to create pairs of the existing terms and the new term to be substituted in to the expression. An example with the `replace` function would be:

```
for old, new in zip(to_be_replaced, new_terms):
    term = term.replace(old, new)
```

Alternatively the SymPy function `subs` can be used with a substitution dictionary containing `{key: value}` pairs such that:

```
expression.subs(substitution_dictionary)
```

Always `pprint` the expression before and after the substitution to check that you are getting the correct behaviour. Sometimes for more complex expressions multiple applications of the substitution functions will be required.

C.5 Left and right hand sides of equations

When manipulating equations in OpenSBLI, the user will often need to access the left and right hand sides of an equation (of type `Eq`) individually. This is true for the substitutions in the previous section, where it is often the right hand side of an equation that needs to be modified. The left and right hand sides of an equation can be accessed simply with:

```
print(example_equation)
Eq(p, (gama - 1)*(-rho*u0**2/2 - rho*u1**2/2 - rho*u2**2/2 + rhoE))
print(example_equation.lhs)
p
print(example_equation.rhs)
(gama - 1)*(-rho*u0**2/2 - rho*u1**2/2 - rho*u2**2/2 + rhoE)
```

C.6 De-nesting multiple iterables

Often in OpenSBLI you will encounter iterables that contain iterables. A common example of this would be a list of lists in Python. To be able to iterate over all of the individual items directly in a single loop it is often convenient to use the SymPy function `flatten`. This function collapses the list of list structure into a single list containing all of the items to iterate over.

```
>>> from sympy import flatten
>>> x = [[1, 2, 3], [4, 5, 6]]
```

```
>>> print(x)
[[1, 2, 3], [4, 5, 6]]
>>> print(flatten(x))
[1, 2, 3, 4, 5, 6]
```

C.7 Accessing attributes in Python objects: `__dict__`

All objects in Python have a `__dict__` routine to show the attributes contained in that object. This is frequently used when developing features for OpenSBLI. The `__dict__` routine returns a Python dictionary `{key: value}` pair. For example, in an OpenSBLI `Kernel` class the equations defined in that kernel will be stored and can be accessed in the attribute `kernel.equations`. In the code snippet below the `__dict__` function is showing the attributes defined in the Runge-Kutta time-stepping class after it has been instantiated.

```
>>> rk = RungeKutta(3)
>>> print(rk.__dict__)
{'solution_coeffs': rkold[stage], 'stage_coeffs': rknew[stage], 'name': 'RungeKutta',
 'iteration_number': iter, 'temporal_iteration': iter, 'solution': {}, 'schemetype':
 'Temporal', 'constant_time_step': True, 'time_step': dt, 'order': 3, 'nloops': 2,
 'stage': stage}
```

C.8 Using built-in SymPy functions (sin, cos, exp, ...)

It's important to understand the differences between the common mathematics functions that are available in Python. In the majority of cases for OpenSBLI, common math functions should be imported from the symbolic SymPy library. A common mistake is to use either the Python `math` library or NumPy. Both of these will throw errors when trying to build symbolic expressions in OpenSBLI.

```
>>> import math, numpy, sympy
>>> print([math.cos(0.5), numpy.cos(0.5), sympy.cos(0.5)])
[0.8775825618903728, 0.8775825618903728, 0.877582561890373]
>>> x = sympy.symbols('x')
>>> sympy.cos(x)
cos(x)
>>> math.cos(x)
Traceback (most recent call last):
  raise TypeError("can't convert expression to float")
TypeError: "can't convert expression to float"
```

Sometimes it can be useful to ask SymPy not to evaluate numerical expressions, or explicitly use symbolic expressions in a function:

```
>>> sympy.cos(0.5)
0.877582561890373
>>> sympy.cos(0.5, evaluate=False)
cos(0.5)
>>> sympy.cos(sympy.Rational(1,2))
cos(1/2)
```

Other common mathematical functions in SymPy can be used in OpenSBLI, which will be converted into C code at the code generation stage. Examples include `Abs`, `Min`, `Max`, `ceil`.

C.9 Conditional expressions (if-else branching)

SymPy also has a boolean logic system that allows us to symbolically create branching if-else statements in the C code. A simple conditional function such as

$$\rho = \begin{cases} 10, & \text{if } x < 3 \\ 5, & \text{if } x \geq 5 \\ 0, & \text{otherwise} \end{cases}$$

would be defined in OpenSBLI using SymPy's `Piecewise` function as:

```
x = DataObject('x0')
eqn = Eq(DataObject('rho'), Piecewise((10.0, x<3), (5.0, x>=5), (0.0,True)))
```

Each `(value, condition)` tuple pair will be written out as branching if and else-if statements. The `(value, True)` tuple is evaluated if all of the other conditions are not met in the simulation code. The `Piecewise` function will throw an error if an else statement is not provided in this format. More sophisticated conditional expressions can be built up using the SymPy `And`, `Or` and `Equality` classes. Each of these logic functions will be translated to the C equivalent in the final code.

OpenSBLI also has a `GroupedPiecewise` class derived from the base SymPy version. `GroupedPiecewise` is useful when multiple equations are to be evaluated inside one of the conditional branches. Usage can be found for example in the `teno.py` file. Tuple pairs of values and conditions can be built up using SymPy's `ExprCondPair` class to be used to form a set of branching statements. Note that SymPy will simplify conditional logic if possible so you may not see every condition entered into the function.

C.10 String representation: `srepr`

Often it is useful to find out how SymPy is storing the symbolic quantities through its string representation functionality. The function `srepr` breaks an expression down into its individual

components and can be useful for debugging purposes. For example the pressure evaluation

$$p = (\gamma - 1) \left(\frac{\rho u_0^2}{2} + \frac{\rho u_1^2}{2} + \frac{\rho u_2^2}{2} + \rho E \right) \quad (\text{C.1})$$

Has the string representation of:

```
OpenSBLIEquation(DataObject('p'), Mul(Add(ConstantObject('gama'), Integer(-1)),
    Add(Mul(Integer(-1), Rational(1, 2), DataObject('rho'), Pow(DataObject('u0'),
    Integer(2))), Mul(Integer(-1), Rational(1, 2), DataObject('rho'),
    Pow(DataObject('u1'), Integer(2))), Mul(Integer(-1), Rational(1, 2),
    DataObject('rho'), Pow(DataObject('u2'), Integer(2))), DataObject('rhoE'))))
```

This is a good way of finding out which quantities are DataObjects, DataSets, ConstantObjects and GridVariables within an OpenSBLI expression. Finding the type of a single object can be done via the standard `type()` function in Python.

C.11 Breaking down an expression into its components: atoms/args

Being able to manipulate symbolic expressions in OpenSBLI relies on knowing how to isolate certain components. Two useful functions within SymPy are `atoms` and `args`.

```
>>> print(expression)
(gama - 1)*(-rho*u0**2/2 - rho*u1**2/2 - rho*u2**2/2 + rhoE)
>>> print(expression.atoms())
set([-1, -1/2, 2, gama, rho, rhoE, u0, u1, u2])
>>> print(expression.atoms(DataObject))
set([rho, rhoE, u0, u1, u2])
>>> print(expression.atoms(ConstantObject))
set([gama])
```

A call to the `atoms()` routine on any expression will return a set of symbolic quantities in it. This includes integers and rational constants. To extract a certain type of object from the expression to iterate over, an object type can be passed to `atoms` such as `atoms(DataObject)`. Note that the output is a Python `set`, which is an unordered collection of unique objects. These differ from a Python `list`, which are an ordered collection that may contain duplicate entries. Useful operations can be performed on multiple sets such as `intersection`, `union` and `difference`. These operations are often used in OpenSBLI to find all of the unique objects that have been defined throughout the code.

```
# Take the right hand side of the continuity equation
>>> continuity_equation = simulation_eq.equations[0].rhs
>>> print(continuity_equation)
-CD rhou0_B0 x0 - CD rhou1_B0 x1 - CD rhou2_B0 x2
>>> print(continuity_equation.args)
```

```
(-CD rhou0_B0 x0, -CD rhou1_B0 x1, -CD rhou2_B0 x2)
# Take the first term d(rhou0)/dx
term = continuity_equation.args[0]
# Iterate over the components of the derivative operator
>>> print([x for x in term.args])
[-1, CD rhou0_B0 x0]
# Iterate over the derivative term
>>> derivative = term.args[1]
>>> print([x for x in derivative.args])
[rhou0_B0, x0]
```

The example above shows how to break down an expression into its individual terms and then isolate the individual components that make up those terms.

C.12 Isolating terms of a certain type: `isinstance`

The Python function `isinstance` is frequently used to isolate objects of a certain type for modification. A use case for this would be if the user wanted to perform a modification only on a certain type of object in an expression while leaving the rest untouched. For example to modify only `DataObjects` in a given expression one could use:

```
for term in equation.rhs.atoms():
    if isinstance(term, DataObject):
        # Perform some operation on the term
```

C.13 Using `GridVariables` as temporary evaluations

The purpose of the `GridVariable` object is to evaluate a quantity locally for the current grid point, use it in an expression to set a `DataSet` and then discard it. The `GridVariable` will take different values at different grid points, but unlike a `DataSet` it is not stored globally for later use. Examples of this are seen throughout OpenSBLI, for example in the definition of boundary conditions.

C.14 Simplifying long expressions and reducing operation counts

The final form of more complex expressions is not always the most efficient from a computational performance point of view. SymPy has a number of functions that can be used to try and simplify an expression in OpenSBLI. It is not always obvious which one is suitable (if any), without printing the expression before and after the modification. Among the possibilities for a given expression are:

1. `simplify(expression)`: Attempts to simplify an expression.
2. `factor(expression)`: Attempts to factor polynomials.
3. `horner(expression)`: Reduces a polynomial into a form requiring the minimum number of additions and multiplications by the Horner algorithm.

One method of assessing the impact of the attempted simplification is the `count_ops()` routine. Using `expression.count_ops()` gives the number of operations in the expression. This can be applied before and after one of the simplification methods to assess the output. Further methods for simplifying expressions can be found here: <https://docs.sympy.org/1.1/tutorial/simplification.html>.

Appendix D

Procedure for restarting simulations

This section outlines the procedure for restarting simulations from a previous OpenSBLI output file. For restarting simulations there is a `generate_restart.py` script available in the OpenSBLI repository. This file modifies how arrays in the simulation are declared. By default storage arrays are declared as zeros using the `ops_decl_dat` function in OPS. When restarting a simulation we instead want to initialise the values of the time advance arrays ($\rho, \rho u, \rho v, \rho w, \rho E$) from a previous simulation checkpoint file. The `generate_restart.py` script performs a simple text replace to use `ops_decl_dat_hdf5` instead of `ops_decl_dat` with the appropriate input parameters.

1. Perform a simulation in OpenSBLI to obtain an `opensbli_output.h5` file.
2. Modify the `generate_restart.py` script available in the repository for the arrays you wish to restart.
3. Move the `opensbli_output.h5` HDF5 file to `restart.h5`.
4. Run `python generate_restart.py` in the directory where your `defdec_data_set.h` file is located.
5. Comment out the grid based initialization kernel in your main `opensbli.cpp` file to prevent the initial condition being applied again.
6. Change the number of iterations to perform in `opensbli.cpp` if necessary.
7. Translate and compile the modified code within OPS to generate a new executable.

Important note: If step 5 is not performed then the simulation will overwrite the restarted data arrays in memory with the initial condition. If the grid coordinates (x_0, x_1, x_2) were originally generated in the initialisation kernel, they should also be written to the output file and added to the `generate_restart.py` script to be reinitialised.

References

- N. Adams. Direct simulation of the turbulent boundary layer along a compression ramp at $M=3$ and $Re = 1685$. *Journal of Fluid Mechanics*, 420:47–83, 2000.
- T. C. Adamson and A. F. Messiter. Analysis of Two-Dimensional Interactions Between Shock Waves and Boundary Layers. *Annual Review of Fluid Mechanics*, 12(1):103–138, Jan 1980. ISSN 0066-4189.
- AnandTech. The Larrabee Chapter Closes: Intel’s Final Xeon Phi Processors Now in EOL, 2019. <https://www.anandtech.com/show/14305/intel-xeon-phi-knights-mill-now-eol/>, Last accessed on 2019-12-16.
- H. Babinsky and J. Harvey. *Shock Wave-Boundary-Layer Interactions*. Cambridge University Press, Cambridge, 2011. ISBN 9780521848527.
- H. Babinsky, J. Oorebeek, and T. Cottingham. Corner effects in reflecting oblique shock-wave/boundary-layer interactions. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, Aerospace Sciences Meetings. American Institute of Aeronautics and Astronautics, Jan 2013.
- K. Bergman, T. Conte, A. Gara, M. Gokhale, M. Heroux, P. Kogge, B. Lucas, S. Matsuoka, V. Sarkar, and O. Temam. Future High Performance Computing Capabilities: Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee. Technical report, United States, 2019. URL <https://www.osti.gov/servlets/purl/1570693>.
- I. Bermejo-Moreno, L. Campo, J. Larsson, J. Bodart, D. Helmer, and J. K. Eaton. Confinement effects in shock wave/turbulent boundary layer interactions through wall-modelled large-eddy simulations. *Journal of Fluid Mechanics*, 758:5–62, 2014. ISSN 14697645.
- A. Bhagatwala and S. K. Lele. A modified artificial viscosity approach for compressible turbulence simulations. *Journal of Computational Physics*, 228(14):4965–4969, 2009. ISSN 0021-9991. doi: <https://doi.org/10.1016/j.jcp.2009.04.009>.
- J. Blazek. *Computational Fluid Dynamics: Principles and Applications*. Elsevier Science, 2001. ISBN 9780080430096.
- U. Bondhugula, A. Hartono, J. Ramanujam, and P. Sadayappan. A practical automatic polyhedral program optimization system. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, jun 2008.

- N. Bonne, V. Brion, E. Garnier, R. Bur, P. Molton, D. Sipp, and L. Jacquin. Analysis of the two-dimensional dynamics of a Mach 1.6 shock wave/transitional boundary layer interaction using a RANS based resolvent approach. *Journal of Fluid Mechanics*, 862:1166–1202, 2019. ISSN 0022-1120.
- R. Borges, M. Carmona, B. Costa, and W. S. Don. An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws. *Journal of Computational Physics*, 227(6):3191–3211, 2008.
- M. E. Brachet, D. I. Meiron, S. A. Orszag, B. G. Nickel, R. H. Morf, and U. Frisch. Small-scale structure of the Taylor–Green vortex. *Journal of Fluid Mechanics*, 130:411–452, 1983. ISSN 0022-1120.
- C. Brehm, M. F. Barad, J. A. Housman, and C. C. Kiris. A comparison of higher-order finite-difference shock capturing schemes. *Computers and Fluids*, 122:184–208, 2015.
- P. J. Bruce, D. M. Burton, N. A. Titchener, and H. Babinsky. Corner effect and separation in transonic channel flows. *Journal of Fluid Mechanics*, 679:247–262, 2011. ISSN 00221120.
- J. R. Bull and A. Jameson. Simulation of the Taylor–Green Vortex Using High-Order Flux Reconstruction Schemes. *AIAA Journal*, 53(9):2750–2761, Jul 2015. ISSN 0001-1452.
- D. M. Burton and H. Babinsky. Corner separation effects for normal shock wave/turbulent boundary layer interactions in rectangular channels. *Journal of Fluid Mechanics*, 707:287–306, 2012. ISSN 00221120.
- J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*. Wiley-Interscience, USA, 1987. ISBN 0471910465.
- M. H. Carpenter and C. A. Kennedy. Fourth-order 2N-storage Runge-Kutta schemes. *NASA Langley Research Center*, Jun 1994.
- M. H. Carpenter, J. Nordström, and D. Gottlieb. A Stable and Conservative Interface Treatment of Arbitrary Spatial Accuracy. *Journal of Computational Physics*, 365(98):341–365, 1998. ISSN 00219991.
- M. Castro, B. Costa, and W. S. Don. High order weighted essentially non-oscillatory WENO-Z schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 230(5):1766–1792, 2011. ISSN 00219991.
- C. Cheng, C. Wang, and K. Cheng. Response of an oblique shock train to downstream periodic pressure perturbations. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 233(1):57–70, Sep 2017. ISSN 0954-4100.
- M. S. Chong, A. E. Perry, and B. J. Cantwell. A general classification of three-dimensional flow fields. *Physics of Fluids A: Fluid Dynamics*, 2(5):765–777, May 1990. ISSN 0899-8213.
- N. T. Clemens and V. Narayanaswamy. Low-Frequency Unsteadiness of Shock Wave/Turbulent Boundary Layer Interactions. *Annual Review of Fluid Mechanics*, 46(1):469–492, Jan 2014. ISSN 0066-4189.
- G. N. Coleman, J. Kim, and R. D. Moser. A numerical study of turbulent supersonic isothermal-wall channel flow. *Journal of Fluid Mechanics*, 305:159–183, 1995. ISSN 0022-1120.

- S. P. Colliss, H. Babinsky, K. Nübler, and T. Lutz. Vortical Structures on Three-Dimensional Shock Control Bumps. *AIAA Journal*, 54(8):2338–2350, May 2016. ISSN 0001-1452. doi: 10.2514/1.J054669.
- V. Daru and C. Tenaud. Numerical simulation of the viscous shock tube problem by using a high resolution monotonicity-preserving scheme. *Computers and Fluids*, 38(3):664–676, 2009. ISSN 00457930.
- J. DeBonis. Solutions of the Taylor-Green Vortex Problem Using High-Resolution Explicit Finite Difference Methods. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, Aerospace Sciences Meetings. American Institute of Aeronautics and Astronautics, Jan 2013.
- G. Degrez, C. H. Boccadoro, and J. F. Wendt. The interaction of an oblique shock wave with a laminar boundary layer revisited. An experimental and numerical study. *Journal of Fluid Mechanics*, 177:247–263, 1987. ISSN 0022-1120. doi: DOI:10.1017/S0022112087000946.
- J. M. Détery. Robert Legendre and Henri Werlé: Toward the Elucidation of Three-Dimensional Separation. *Annual Review of Fluid Mechanics*, 33(1):129–154, Jan 2001. ISSN 0066-4189.
- M. Diop, S. Piponnier, and P. Dupont. High resolution LDA measurements in transitional oblique shock wave boundary layer interaction. *Experiments in Fluids*, 60(4):57, 2019. ISSN 1432-1114.
- D. S. Dolling. Fifty Years of Shock-Wave/Boundary-Layer Interaction Research: What Next? *AIAA Journal*, 39(8):1517–1531, Aug 2001.
- F. Ducros, V. Ferrand, F. Nicoud, C. Weber, D. Darracq, C. Gacherieu, and T. Poinso. Large-eddy simulation of the shock/turbulence interaction. *Journal of Computational Physics*, 152(2):517 – 549, 1999.
- F. Ducros, F. Laporte, T. Soulères, V. Guinot, P. Moinat, and B. Caruelle. High-order fluxes for conservative skew-symmetric-like schemes in structured meshes: Application to compressible flows. *Journal of Computational Physics*, 161(1):114 – 139, 2000.
- A. Dwivedi, J. W. Nichols, M. R. Jovanovic, and G. V. Candler. Optimal spatial growth of streaks in oblique shock/boundary layer interaction. In *8th AIAA Theoretical Fluid Mechanics Conference*, AIAA AVIATION Forum. American Institute of Aeronautics and Astronautics, Jun 2017.
- W. E. Eagle and J. F. Driscoll. Shock wave–boundary layer interactions in rectangular inlets: three-dimensional separation topology and critical points. *Journal of Fluid Mechanics*, 756: 328–353, 2014. ISSN 0022-1120.
- W. E. Eagle, J. F. Driscoll, and J. a. Benek. Experimental Investigation of Corner Flows in Rectangular Supersonic Inlets with 3D Shock-Boundary Layer Effects. *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, (January): 1–11, 2011. doi: 10.2514/6.2011-857.
- J. A. Ekaterinaris. High-order accurate, low numerical diffusion methods for aerodynamics. *Progress in Aerospace Sciences*, 41(3):192–300, 2005. ISSN 0376-0421.

- R. Fiévet, H. Koo, V. Raman, and A. H. Auslender. Numerical Investigation of Shock-Train Response to Inflow Boundary-Layer Variations. *AIAA Journal*, 55(9):2888–2901, May 2017. ISSN 0001-1452.
- L. Fu, X. Y. Hu, and N. A. Adams. A family of high-order targeted ENO schemes for compressible-fluid simulations. *Journal of Computational Physics*, 305:333–359, 2016. ISSN 10902716.
- L. Fu, X. Y. Hu, and N. A. Adams. Targeted ENO schemes with tailored resolution property for hyperbolic conservation laws. *Journal of Computational Physics*, 349:97 – 121, 2017. ISSN 0021-9991.
- L. Fu, X. Y. Hu, and N. A. Adams. Improved Five- and Six-Point Targeted Essentially Nonoscillatory Schemes with Adaptive Dissipation. *AIAA Journal*, 57(3):1143–1158, Dec 2018. ISSN 0001-1452.
- D. V. Gaitonde. Progress in shock wave/boundary layer interactions. *Progress in Aerospace Sciences*, 72:80–99, 2015. ISSN 0376-0421.
- B. Ganapathisubramani, N. T. CLEMENS, and D. S. DOLLING. Effects of upstream boundary layer on the unsteadiness of shock-induced separation. *Journal of Fluid Mechanics*, 585:369–394, 2007.
- E. Garnier. Stimulated Detached Eddy Simulation of three-dimensional shock/boundary layer interaction. *Shock Waves*, 19(6):479–486, 2009. ISSN 09381287.
- N. J. Georgiadis, D. P. Rizzetta, and C. Fureby. Large-Eddy Simulation: Current Capabilities, Recommended Practices, and Future Research. *AIAA Journal*, 48(8):1772–1784, Aug 2010. ISSN 0001-1452.
- D. Ghosh and J. D. Baeder. Compact reconstruction schemes with weighted ENO limiting for hyperbolic conservation laws. *SIAM Journal on Scientific Computing*, 34(3):A1678–A1706, Jan 2012.
- R. H. M. Giepman, F. F. J. Schrijer, and B. W. van Oudheusden. High-resolution PIV measurements of a transitional shock wave–boundary layer interaction. *Experiments in Fluids*, 56(6):113, 2015. ISSN 1432-1114. doi: 10.1007/s00348-015-1977-8.
- R. H. M. Giepman, R. Louman, F. F. J. Schrijer, and B. W. van Oudheusden. Experimental Study into the Effects of Forced Transition on a Shock-Wave/Boundary-Layer Interaction. *AIAA Journal*, 54(4):1313–1325, Feb 2016. ISSN 0001-1452. doi: 10.2514/1.J054501.
- R. H. M. Giepman, F. F. J. Schrijer, and B. W. van Oudheusden. A parametric study of laminar and transitional oblique shock wave reflections. *Journal of Fluid Mechanics*, 844:187–215, 2018. ISSN 0022-1120.
- S. Gottlieb and C.-W. Shu. Total variation diminishing Runge-Kutta schemes. *Mathematics of Computation of the American Mathematical Society*, 67(221):73–85, 1998. ISSN 0025-5718.
- A. Grébert, J. Bodart, S. Jamme, and L. Joly. Simulations of shock wave/turbulent boundary layer interaction with upstream micro vortex generators. *International Journal of Heat and Fluid Flow*, 72:73–85, 2018. ISSN 0142-727X.

- J. E. Green. Interactions between shock waves and turbulent boundary layers. *Progress in Aerospace Sciences*, 11:235–340, 1970. ISSN 0376-0421.
- J. R. Grisham, B. H. Dennis, and F. K. Lu. Incipient Separation in Laminar Ramp-Induced Shock-Wave/Boundary-Layer Interactions. *AIAA Journal*, 56(2):524–531, Oct 2017.
- A. Gross and H. F. Fasel. Numerical Investigation of Shock Boundary-Layer Interactions. In *54th AIAA Aerospace Sciences Meeting*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, Jan 2016. doi: doi:10.2514/6.2016-0347.
- I. J. Grossman and P. J. Bruce. Effect of Test Article Geometry on Shock Wave-Boundary Layer Interactions in Rectangular Intakes. In *55th AIAA Aerospace Sciences Meeting*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, Jan 2017.
- I. J. Grossman and P. J. K. Bruce. Confinement effects on regular–irregular transition in shock-wave–boundary-layer interactions. *Journal of Fluid Mechanics*, 853:171–204, 2018. ISSN 0022-1120. doi: DOI:10.1017/jfm.2018.537.
- F. Guiho, F. Alizard, and J.-C. Robinet. Instabilities in oblique shock wave/laminar boundary-layer interactions. *Journal of Fluid Mechanics*, 789:1–35, 2016.
- R. J. Hakkinen, I. Greber, L. Trilling, and S. Abarbanel. The interaction of an oblique shock wave with a laminar boundary layer. *NASA Memorandum 2-18-59W*, 1959.
- A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes, III. *Journal of Computational Physics*, 71(2):231–303, 1987. ISSN 0021-9991.
- T. R. Hendrickson, A. Kartha, and G. V. Candler. An Improved Ducros Sensor for the Simulation of Compressible Flows with Shocks. In *2018 Fluid Dynamics Conference*, AIAA AVIATION Forum. American Institute of Aeronautics and Astronautics, Jun 2018. doi: doi:10.2514/6.2018-3710.
- A. K. Henrick, T. D. Aslam, and J. M. Powers. Mapped weighted essentially non-oscillatory schemes: Achieving optimal order near critical points. *Journal of Computational Physics*, 207(2):542–567, 2005. ISSN 00219991.
- N. Hildebrand, A. Dwivedi, J. W. Nichols, M. R. Jovanović, and G. V. Candler. Simulation and stability analysis of oblique shock-wave/boundary-layer interactions at Mach 5.92. *Physical Review Fluids*, 3(1):1–23, 2018a. ISSN 2469990X. doi: 10.1103/PhysRevFluids.3.013906.
- N. J. Hildebrand, J. W. Nichols, G. V. Candler, and M. Jovanovic. Transient growth in oblique shock wave/laminar boundary layer interactions at Mach 5.92. In *2018 Fluid Dynamics Conference*, AIAA AVIATION Forum. American Institute of Aeronautics and Astronautics, Jun 2018b.
- T. Houba, A. Dasgupta, S. Gopalakrishnan, R. Gosse, and S. Roy. Supersonic turbulent flow simulation using a scalable parallel modal discontinuous Galerkin numerical method. *Scientific Reports*, 9(1):14442, 2019. ISSN 2045-2322.
- X. Y. Hu, Q. Wang, and N. A. Adams. An adaptive central-upwind weighted essentially non-oscillatory scheme. *Journal of Computational Physics*, 229(23):8952–8965, 2010. ISSN 00219991.

- R. L. Hunt and M. Gamba. On the origin and propagation of perturbations that cause shock train inherent unsteadiness. *Journal of Fluid Mechanics*, 861:815–859, 2019. ISSN 0022-1120.
- C. T. Jacobs, S. P. Jammy, and N. D. Sandham. OpenSBLI: A framework for the automated derivation and parallel execution of finite difference solvers on a range of computer architectures. *Journal of Computational Science*, 18:12 – 23, 2017.
- C. T. Jacobs, M. Zauner, N. De Tullio, S. P. Jammy, D. J. Lusher, and N. D. Sandham. An error indicator for finite difference methods using spectral techniques with application to aerofoil simulation. *Computers & Fluids*, 168:67–72, 2018. ISSN 0045-7930.
- S. Jammy, C. Jacobs, D. Lusher, and N. Sandham. Energy consumption of algorithms for solving the compressible Navier-Stokes equations on CPU’s, GPU’s and KNL’s. In *Proceedings of the 6th European Conference on Computational Mechanics (ECCM 6) and 7th European Conference on Computational Fluid Dynamics (ECFD 7)*, Glasgow, UK, June 2018.
- S. P. Jammy, C. T. Jacobs, and N. D. Sandham. Performance evaluation of explicit finite difference algorithms with varying amounts of computational and memory intensity. *Journal of Computational Science*, 2016. ISSN 1877-7503.
- G.-S. Jiang and C.-W. Shu. Efficient Implementation of Weighted ENO Schemes. *Journal of Computational Physics*, 126(1):202–228, 1996. ISSN 0021-9991.
- E. Johnsen, J. Larsson, A. V. Bhagatwala, W. H. Cabot, P. Moin, B. J. Olson, P. S. Rawat, S. K. Shankar, B. Sjögren, H. Yee, X. Zhong, and S. K. Lele. Assessment of high-resolution methods for numerical simulations of compressible turbulence with shock waves. *Journal of Computational Physics*, 229(4):1213 – 1237, 2010.
- E. Katzer. On the lengthscales of laminar shock/boundary-layer interaction. *Journal of Fluid Mechanics*, 206(-1):477, 1989. ISSN 0022-1120.
- C. A. Kennedy, M. H. Carpenter, and R. Lewis. Low-storage, explicit Runge–Kutta schemes for the compressible Navier–Stokes equations. *Applied Numerical Mathematics*, 35(3):177–219, 2000. ISSN 0168-9274.
- J. Kim, P. Moin, and R. Moser. Turbulence statistics in fully developed channel flow at low Reynolds number. *Journal of Fluid Mechanics*, 177:133–166, 1987. ISSN 0022-1120.
- S. Koike and H. Babinsky. Effect of Vortex Generators on Corner Flow Separation Caused by Shock Wave-Boundary-Layer Interaction. In *2018 AIAA Aerospace Sciences Meeting*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, Jan 2018. ISBN 978-1-62410-524-1.
- M. Lange, N. Kukreja, M. Louboutin, F. Luporini, F. Vieira, V. Pandolfo, P. Velesko, P. Kazakas, and G. Gorman. Devito: Towards a Generic Finite Difference DSL Using Symbolic Python. PyHPC ’16, pages 67–75. IEEE Press, 2016.
- L. Larchevêque. Low- and Medium-Frequency Unsteadinesses in a Transitional Shock–Boundary Reflection with Separation. In *54th AIAA Aerospace Sciences Meeting*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, Jan 2016. doi: doi:10.2514/6.2016-1833.

- S. Lee and A. Gross. Numerical Investigation of Super- and Hypersonic Laminar Shockwave Boundary Layer Interactions. In *AIAA Aviation 2019 Forum*, AIAA AVIATION Forum. American Institute of Aeronautics and Astronautics, Jun 2019. doi: doi:10.2514/6.2019-3441.
- J. Lefieux, E. Garnier, and N. Sandham. DNS Study of Roughness-Induced Transition at Mach 6. In *AIAA Aviation 2019 Forum*, AIAA AVIATION Forum. American Institute of Aeronautics and Astronautics, Jun 2019.
- M. Louboutin, M. Lange, F. Luporini, N. Kukreja, P. A. Witte, F. J. Herrmann, P. Velesko, and G. J. Gorman. Devito (v3.1.0): an embedded domain-specific language for finite differences and geophysical exploration. *Geoscientific Model Development*, 12(3):1165–1187, 2019.
- F. Luporini, M. Lange, M. Louboutin, N. Kukreja, J. Hükelheim, C. Yount, P. Witte, P. H. J. Kelly, F. J. Herrmann, and G. J. Gorman. Architecture and performance of devito, a system for automated stencil computation. *CoRR*, abs/1807.03032, jul 2018. URL <http://arxiv.org/abs/1807.03032>.
- D. Lusher, S. Jammy, and N. Sandham. Transitional shockwave/boundary-layer interactions in the automatic source-code generation framework OpenSBLI. In *Proceedings of the International Conference on Computational Fluid Dynamics (ICCFD10)*, Barcelona, Spain, July 2018a.
- D. J. Lusher and N. Sandham. Assessment of low-dissipative shock-capturing schemes for transitional and turbulent shock interactions. In *AIAA Aviation 2019 Forum*. American Institute of Aeronautics and Astronautics, June 2019a. doi: doi:10.2514/6.2019-3208.
- D. J. Lusher and N. D. Sandham. The effect of flow confinement on laminar shockwave/boundary-layer interactions. *arXiv:1909.01287*, 2019b.
- D. J. Lusher, S. P. Jammy, and N. D. Sandham. Shock-wave/boundary-layer interactions in the automatic source-code generation framework OpenSBLI. *Computers & Fluids*, 173:17 – 21, 2018b. ISSN 0045-7930.
- S. Matsuyama. Performance of all-speed AUSM-family schemes for DNS of low Mach number turbulent channel flow. *Computers & Fluids*, 91:130–143, 2014. ISSN 0045-7930.
- S. McIntosh-Smith, J. Price, T. Deakin, and A. Poenaru. A performance analysis of the first generation of HPC-optimized Arm processors. *Concurrency and Computation: Practice and Experience*, 31(16):e5110, Aug 2019. ISSN 1532-0626.
- A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, Jan. 2017. ISSN 2376-5992.
- R. R. Morajkar and M. Gamba. Swept shock corner flow interactions. *54th AIAA Aerospace Sciences Meeting*, (January):1–13, 2016. doi: 10.2514/6.2016-1165.
- E. Motheau and J. Wakefield. Capturing shocks and turbulence spectra in compressible flows. Part 1: Comparison of low and high-order finite-volume methods. *arXiv e-prints*, art. arXiv:1902.06665, Feb 2019.

- G. R. Mudalige, M. B. Giles, I. Z. Reguly, C. Bertolli, and P. H. J. Kelly. Op2: An active library framework for solving unstructured mesh-based applications on multi-core and many-core architectures. *2012 Innovative Parallel Computing (InPar)*, pages 1–12, 2012.
- G. R. Mudalige, I. Z. Reguly, M. B. Giles, W. Gaudin, A. Mallinson, and O. Perks. High-level abstractions for performance, portability and continuity of scientific software on future computing systems, 2014.
- G. R. Mudalige, I. Z. Reguly, M. B. Giles, A. C. Mallinson, W. P. Gaudin, and J. A. Herdman. Performance analysis of a high-level abstractions-based hydrocode on future computing systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8966:85–104, 2015. ISSN 16113349.
- G. R. Mudalige, I. Z. Reguly, S. P. Jammy, C. T. Jacobs, M. B. Giles, and N. D. Sandham. Large-scale performance of a DSL-based multi-block structured-mesh application for Direct Numerical Simulation. *Journal of Parallel and Distributed Computing*, 131:130–146, 2019. ISSN 0743-7315.
- J. W. Nichols, J. Larsson, M. Bernardini, and S. Pirozzoli. Stability and modal analysis of shock/boundary layer interactions. *Theoretical and Computational Fluid Dynamics*, 31(1): 33–50, 2017. ISSN 1432-2250.
- Nvidia GPU applications. Nvidia GPU applications catalog, 2019. <https://www.nvidia.com/en-us/data-center/gpu-accelerated-applications/catalog/>, Last accessed on 2019-11-30.
- I. Ober and I. Ober. On Patterns of Multi-domain Interaction for Scientific Software Development focused on Separation of Concerns. *Procedia Computer Science*, 108:2298–2302, 2017. ISSN 1877-0509.
- V. Pandolfo. Investigating the OPS intermediate representation to target GPUs in the Devito DSL. arXiv:1906.10811, 2019.
- N. Peng and Y. Yang. Effects of the mach number on the evolution of vortex-surface fields in compressible taylor-green flows. *Phys. Rev. Fluids*, 3:013401, Jan 2018.
- A. E. Perry and M. S. Chong. A Description of Eddying Motions and Flow Patterns Using Critical-Point Concepts. *Annual Review of Fluid Mechanics*, 19(1):125–155, Jan 1987. ISSN 0066-4189.
- A. E. Perry and H. Hornung. Some aspects of three-dimensional separation. II - Vortex skeletons. *Zeitschrift für Flugwissenschaften und Weltraumforschung*, 8:155–160, June 1984.
- S. Pirozzoli. Numerical Methods for High-Speed Flows. *Annual Review of Fluid Mechanics*, 43(1):163–194, 2011. ISSN 0066-4189.
- S. Pirozzoli and F. Grasso. Direct numerical simulation of impinging shock wave/turbulent boundary layer interaction at $M=2.25$. *Physics of Fluids*, 18(6), 2006. ISSN 10706631.
- S. Pirozzoli, F. Grasso, and T. B. Gatski. Direct numerical simulation and analysis of a spatially evolving supersonic turbulent boundary layer at $M=2.25$. *Physics of Fluids*, 16(3):530–545, Jan 2004. ISSN 1070-6631.

- J. Poggie and K. M. Porter. Flow structure and unsteadiness in a highly confined shock-wave–boundary-layer interaction. *Phys. Rev. Fluids*, 4:024602, Feb 2019.
- R. Quadros and M. Bernardini. Numerical Investigation of Transitional Shock-Wave/Boundary-Layer Interaction in Supersonic Regime. *AIAA Journal*, 56(7):2712–2724, Apr 2018. ISSN 0001-1452. doi: 10.2514/1.J056650.
- P. K. Rabey, S. P. Jammy, P. J. K. Bruce, and N. D. Sandham. Two-dimensional unsteadiness map of oblique shock wave/boundary layer interaction with sidewalls. *Journal of Fluid Mechanics*, 871:R4, 2019. ISSN 0022-1120.
- D. C. Reda and J. D. Murphy. Shock Wave/Turbulent Boundary-Layer Interactions in Rectangular Channels. *AIAA Journal*, 11(2):139–140, Feb 1973a. ISSN 0001-1452.
- D. C. Reda and J. D. Murphy. Sidewall Boundary-Layer Influence on Shock Wave/Turbulent Boundary-Layer Interactions. *AIAA Journal*, 11(10):1367–1368, Oct 1973b. ISSN 0001-1452.
- I. Z. Reguly, G. R. Mudalige, M. B. Giles, D. Curran, and S. McIntosh-Smith. The OPS Domain Specific Abstraction for Multi-block Structured Grid Computations. WOLFHPC ’14, pages 58–67. IEEE Press, 2014.
- K. Ritos, I. W. Kokkinakis, and D. Drikakis. Physical insight into a Mach 7.2 compression corner flow. In *2018 AIAA Aerospace Sciences Meeting*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, Jan 2018.
- J.-C. Robinet. Bifurcations in shock-wave/laminar-boundary-layer interaction: global instability approach. *Journal of Fluid Mechanics*, 579:85–112, 2007. ISSN 0022-1120.
- C. L. Rumsey and R. T. Biedron. NASA CFL-3D User Manual. Appendix F - Generalized coordinates. https://cfl3d.larc.nasa.gov/Cfl3dv6/cfl3dv6_v5manual.html, March 2013. (Accessed on 21/11/2019).
- N. D. Sandham. Effects of Compressibility and Shock-Wave Interactions on Turbulent Shear Flows. *Flow, Turbulence and Combustion*, 97(1):1–25, 2016. ISSN 1573-1987. doi: 10.1007/s10494-016-9733-6.
- A. Sansica. *Stability and unsteadiness of transitional shock-wave/boundary-layer interactions in supersonic flows*. PhD thesis, University of Southampton, 2015.
- A. Sansica, N. Sandham, and Z. Hu. Stability and Unsteadiness in a 2D Laminar Shock-Induced Separation Bubble. In *43rd Fluid Dynamics Conference*, Fluid Dynamics and Co-located Conferences. American Institute of Aeronautics and Astronautics, Jun 2013.
- A. Sansica, N. D. Sandham, and Z. Hu. Instability and low-frequency unsteadiness in a shock-induced laminar separation bubble. *Journal of Fluid Mechanics*, 798:5–26, 2016. ISSN 14697645.
- S. Sarkar, G. Erlebacher, M. Y. Hussaini, and H. O. Kreiss. The analysis and modelling of dilatational terms in compressible turbulence. *Journal of Fluid Mechanics*, 227:473–493, 1991.
- N. Sharma and T. K. Sengupta. Vorticity dynamics of the three-dimensional Taylor-Green vortex problem. *Physics of Fluids*, 31(3):35106, Mar 2019. ISSN 1070-6631. doi: 10.1063/1.5083870.

- C.-W. Shu. Essentially Non-Oscillatory and Weighted Essentially Non-Oscillatory Schemes for Hyperbolic Conservation Laws Operated by Universities Space Research Association. *ICASE Report*, (97-65):1–78, 1997.
- C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes. *Journal of Computational Physics*, 77(2):439–471, 1988. ISSN 00219991.
- J. Sivasubramanian and H. F. Fasel. Numerical Investigation of Shock-Induced Laminar Separation Bubble in a Mach 2 Boundary Layer. *45th AIAA Fluid Dynamics Conference*, 2641: 1–36, 2015. doi: 10.2514/6.2015-2641.
- E. M. Taylor, M. Wu, and M. P. Martín. Optimization of nonlinear error for weighted essentially non-oscillatory methods in direct numerical simulations of compressible turbulence. *Journal of Computational Physics*, 223(1):384–397, 2007. ISSN 0021-9991.
- C. Tenaud, E. Garnier, and P. Sagaut. Evaluation of some high-order shock capturing schemes for direct numerical simulation of unsteady two-dimensional free flows. *International Journal for Numerical Methods in Fluids*, 33(2):249–278, 2000. ISSN 02712091.
- M. Tobak and D. J. Peake. Topology of Three-Dimensional Separated Flows. *Annual Review of Fluid Mechanics*, 14(1):61–85, Jan 1982. ISSN 0066-4189. doi: 10.1146/annurev.fl.14.010182.000425.
- E. Toubert and N. Sandham. Low-order stochastic modelling of low-frequency motions in reflected shock-wave/boundary-layer interactions. *Journal of Fluid Mechanics*, 671:417–465, 2011.
- E. Toubert and N. D. Sandham. Large-eddy simulation of low-frequency unsteadiness in a turbulent shock-induced separation bubble. *Theoretical and Computational Fluid Dynamics*, 23(2):79–107, 2009.
- D. Unat, X. Cai, and S. B. Baden. Mint: realizing CUDA performance in 3D stencil methods with annotated C. *Proceedings of the international conference on Supercomputing*, pages 214–224, 2011.
- S. Vaisakh, P. R. Namratha, and T. M. Muruganandam. Effect of duct aspect ratio on normal shock wave/boundary layer interaction. *Shock Waves*, 2019. ISSN 1432-2153.
- B. C. Vermeire, F. D. Witherden, and P. E. Vincent. On the utility of GPU accelerated high-order methods for unsteady flow simulations: A comparison with industry-standard tools. *Journal of Computational Physics*, 334:497–521, 2017. ISSN 0021-9991.
- P. Vincent, F. Witherden, B. Vermeire, J. S. Park, and A. Iyer. Towards Green Aviation with Python at Petascale. In *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11, 2016. ISBN 2167-4337 VO -. doi: 10.1109/SC.2016.1.
- A. W. Vreman and J. G. M. Kuerten. Comparison of direct numerical simulation databases of turbulent channel flow at $Re\tau = 180$. *Physics of Fluids*, 26(1):15102, Jan 2014. ISSN 1070-6631.
- A. W. Vreman, N. D. Sandham, and K. H. Luo. Compressible mixing layer growth rate and turbulence characteristics. *Journal of Fluid Mechanics*, 320:235–258, 1996. ISSN 0022-1120.

- B. Wang, N. D. Sandham, Z. Hu, and W. Liu. Numerical study of oblique shock-wave/boundary-layer interaction considering sidewall effects. *Journal of Fluid Mechanics*, 767:526–561, 2015. doi: 10.1017/jfm.2015.58.
- Z. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, and M. Visbal. High-order cfd methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, 72(8):811–845, 2013.
- B. Wasistho. *Spatial direct numerical simulation of compressible boundary layer flow*. Ph. D. thesis, University of Twente. Citeseer, 1997.
- F. White. *Viscous fluid flow*. McGraw-Hill, New York, NY, 2006. ISBN 0072402318.
- S. Williams, A. Waterman, and D. Patterson. Roofline: An insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, Apr. 2009. ISSN 0001-0782.
- J. Williamson. Low-storage runge-kutta schemes. *Journal of Computational Physics*, 35(1):48 – 56, 1980. ISSN 0021-9991.
- F. D. Witherden, A. M. Farrington, and P. E. Vincent. PyFR: An open source framework for solving advection–diffusion type problems on streaming architectures using the flux reconstruction approach. *Computer Physics Communications*, 185(11):3028–3040, 2014. ISSN 0010-4655.
- A. A. Wray. Minimal storage time advancement schemes for spectral methods. *NASA Ames Research Center, California, Report No. MS*, 202, 1990.
- M. Wu and M. P. Martin. Analysis of shock motion in shockwave and turbulent boundary layer interaction using direct numerical simulation data. *Journal of Fluid Mechanics*, 594:71–83, 2008.
- X. Xiang and H. Babinsky. An Experimental Study of Corner Flow Control Applied to an Oblique Shock-Wave/Boundary-Layer Interaction. In *2018 AIAA Aerospace Sciences Meeting*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, Jan 2018.
- X. Xiang and H. Babinsky. Corner effects for oblique shock wave/turbulent boundary layer interactions in rectangular channels. *Journal of Fluid Mechanics*, 862:1060–1083, 2019. ISSN 0022-1120.
- Y. Yao, L. Krishnan, N. D. Sandham, and G. T. Roberts. The effect of Mach number on unstable disturbances in shock/boundary-layer interactions. *Physics of Fluids*, 19(5):1–15, 2007. ISSN 10706631.
- H. C. Yee and B. Sjögren. Recent developments in accuracy and stability improvement of nonlinear filter methods for DNS and LES of compressible flows. *Computers & Fluids*, 169: 331–348, 2018. ISSN 0045-7930.
- O. Zeman. Dilatation dissipation: The concept and application in modeling compressible mixing layers. *Physics of Fluids A: Fluid Dynamics*, 2(2):178–188, 1990.