ISSN 0140 3818
SSSU 38

TWO-DIMENSIONAL WIND TUNNEL TESTS OF THE SSS SERIES 1 STRUCTURALLY EFFICIENT AEROFOILS FOR WIND TURBINES

by S.R. Turnock

Ship Science Report No. 38

June 1989

UNIVERSITY OF SOUTHAMPTON

DEPARTMENT OF SHIP SCIENCE

FACULTY OF ENGINEERING

AND APPLIED SCIENCE

ISSN 0140 3818
SSSU 38

UNIVERSITY OF SOUTHAMPTON

DEPARTMENT OF SHIP SCIENCE

FACULTY OF ENGINEERING AND APPLIED SCIENCE

TWO-DIMENSIONAL WIND TUNNEL TESTS OF THE SSS SERIES 1 STRUCTURALLY EFFICIENT AEROFOILS FOR WIND TURBINES

by S.R. Turnock

Ship Science Report No. 38

June 1989

TWO-DIMENSIONAL WIND TUNNEL TESTS OF THE SSS SERIES 1 STRUCTURALLY EFFICIENT AEROFOILS FOR WIND TURBINES

SUMMARY

The two-dimensional lift, drag, and quarter chord moment coefficients were measured for the new design Southampton Ship Science series 1 aerofoils for angles of attack between a minimum of -6° and a maximum of 21°. Five of the new series aerofoils were tested with thickness-chord ratio varying in 2% increments between 21% and 29%. An LS(1) 0421 mod aerofoil was also tested. All six aerofoils had a 610mm chord and an overall span of 995mm.

The aerofoils were tested over a three week period using a two-dimensional test rig designed for use in the Southampton University 11' * 8' working section closed return low speed wind tunnel. A detailed description of the design, operation and installation of this test rig forms the main part of this report.

The aerodynamic forces and moments were measured on the central 595mm span of the aerofoil using two three-component strain gauge balances. A wake traverse mechanism 0.5 chord downstream of the aerofoil trailing edge was used to obtain a further drag measurement.

Results for each aerofoil are presented in the form of Lift-Incidence, Drag-Lift, and Moment-Lift polars. Tests were conducted at a chord Reynolds number of 1600000 and 2000000. Measurements with a roughness strip applied at 5% chord from the leading edge were also obtained.

S.R. Turnock

June 1989

TABLE OF CONTENTS

List of Figures

List of Tables

Nomenclature

C_l	- Non-Dimensional 2D Lift Coefficient.
C_d	- Non-Dimensional 2D Drag Coefficient.
C_m	- Non-Dimensional 2D Moment about (1/4) chord Coefficient.
ρ	- Density (kg/m↑3)
U	- Velocity (m/s)
D	- Drag (N)
L	- Lift (N)
M	- Moment (Nm)
t	- Thickness (m)
c	- Chord (m)
s	- Span (m)
p	- Pressure (N/m↑2)
y	- height (m)
x	- distance (m)
h	- wind tunnel section height (m)
ϵ	- parameter

α	–	angle of incidence
λ	–	aerofoil shape factor

Subscripts

∞	–	freestream
u	–	upper
l	–	lower
l	–	left
m	–	middle
r	–	right

1. INTRODUCTION

1.1 Purpose

This report serves two purposes:

1) To provide a detailed description of the aerodynamic performance testing of a series of five new aerofoils specifically designed for wind turbine applications;

2) To give sufficient instructions and detail for the operation,installation and calibration of the two dimensional aerofoil testing rig used in assessing the aerofoil performance. The tests were carried out in the 11' * 8' closed return low-speed wind tunnel at Southampton University.

The report is similarly divided into two main sections. Chapters 2 through 5 deal with the details specific to the test rig and chapters 6 to 8 with the aerodynamic performance tests and data analysis carried out.

Chapter 1 gives background information on the need for thick sectioned aerofoils specifically designed for Horizontal wind axis turbines (HAWT) machines. The design method used for the new aerofoils is briefly described in Chapter Two along with the fabrication of the aerofoil models for the wind tunnel tests. Chapter 3 covers the design, construction, installation and operating instructions for the two-dimensional aerofoil testing rig. It is intended as a 'user guide' for future use of the 'splitter' rig. For this series of tests a wake traverse was installed to give an improved profile drag measurement. The design of the traverse mechanism is described in Chapter 4.

The data acquisition methodology is detailed in Chapter 5, covering the system design, hardware and software. Chapter 6 details the data analysis techniques and the accuracy of measurements obtained. The results for the new series aerofoils are presented individually in Chapter 7. The final chapter gives a brief discussion of the implications of the results obtained and some future improvements to the two-dimensional test rig.

1.2 Wind Turbine Aerofoils

Horizontal Axis Wind Turbine blades have specific requirements for aerofoil sections of which they are composed. These requirements are usually expressed in terms of the two-dimensional aerodynamic performance characteristics: Cl, Cd and Cm. The economics of electricity production dictate that wind turbine blades have a long life until failure, usually from fatigue. In general, a thicker sectioned aerofoil (higher thickness/chord ratio t/c) will reduce the weight of a section and increase the fatigue lifetime of a blade.

It is common practice when designing blades to use aerofoil sections developed for aircraft applications. Aerofoils such as the NASA developed LS(1) 0421 Mod profile have found widespread use. Recent advances in the field of computational fluid dynamics have allowed design tools to be developed which enable specific sections to be developed for a given requirement.

An inverse design method has been developed in the Ship Science Department at Southampton University by Dr. C.J. Satchwell. Initially a series of five aerofoils varying in thickness/chord ratio between 21% and 29% have been designed. This new series of aerofoils will be referred to as the SSS-1NN (Southampton Ship Science series 1). The numeric NN designates the percentage thickness/chord ratio of the aerofoil. To validate the design procedure all five aerofoils were tested over a three week period in the Southampton University 11' * 8' working section closed return low speed wind tunnel. A specifically designed two-dimensional testing rig was used from which the aerofoil performance characteristics could be obtained as a function of angle of attack.

2. NEW SERIES AEROFOIL DESIGN AND CONSTRUCTION

2.1 Introduction

The aim of the investigation of which the aerodynamic performance testing was a part was to design thick sectioned structurally efficient aerofoils for wind turbines using modern computational fluid dynamic techniques. An inverse design method was developed by Dr. C.J.Satchwell. This uses a viscous/inviscid approach for the flow over the aerofoil. The flow in the outer region away from the aerofoil is solved using an inviscid potential panel method coupled to an inner region viscous integral boundary layer for the flow adjacent to the aerofoil. Further details can be found in reference[1].

A flow solver written by B.Williams [2] was used to determine the aerodynamic performance characteristics, giving as output a theoretical expectation for the two-dimensional parameters Cd, Cl and Cm as a function of angle of attack.

To validate experimentally the design tool and provide information on the properties of the SSS series 1 aerofoils five aerofoils varying in equal thickness increments between 21% and 29% were constructed. The aerofoils were tested using the two-dimensional testing rig previously developed for the 11'*8' tunnel [3]. The aerofoil chord length was ,therefore, fixed at 609.6mm (2') with an overall aerofoil span of 995.5mm. A sixth aerofoil was also constructed using the LS(1) 0421 mod 21% thick section to allow a direct comparison of the experimental data obtained with the two-dimensional test rig to those of McGhee and Beasley [4] for the same section.

2.2 Aerofoil Definition

The final output of the aerofoil design tool is a file of non-dimensional coordinates for the upper and lower surface of an aerofoil. To produce the final working drawing for use by the model maker several processing stages on the data have to be completed:-

1) Inspect visually, by plotting the profile obtained using the AutoCad(R) software mounted on the Ship Science Department network of Rm Nimbus computers.

2) Fair the nose region of the aerofoil. The design code produced a profile for

one fixed point and as a result had a cusped stagnation point which needed smoothing out.

3) Slight smoothing was necessary on the upper surface at the start of the pressure recovery region.

4) To obtain a prescribed thickness to chord ratio the coordinates had to be slightly scaled this resulted in a small amount of trailing edge truncation

5) The datum line produced by the design tool had to be displaced by a known amount to provide enough material for mounting the aerofoil securely.

Once these corrections had been made to the original data file, the actual aerofoil dimensions were scaled for a 2' chord and AutoCad(R) was used to plot out an exact sized aerofoil profile on dimensionally stable drawing film.

The aerofoil coordinates (Non-dimensional) used for the five new series aerofoils constructed are detailed as Appendix A of this report.

2.3 Construction

Following experience with the truncated series of aerofoils [3] it was decided to divide the aerofoil into three spanwise segments. This isolates the problems associated with the junction vortices at the intersection of the aerofoil-splitter onto the two end segments. The junction vortices adversely affect the drag measurement. Only the centre section is connected to the two strain gauge balances used to measure aerodynamic forces and therefore will give a more representative two-dimensional force measurement without having to resort to corner vortex corrections in the data analysis.

After consultation with the Ship Science Department modelmaker it was decided to construct the aerofoils from Jelutong. This is a hard wood that is relatively light, dimensionally stable and machines/works well to give an excellent finish. It was decided to retain the method of mounting the aerofoil on two ϕ 10 spigots 150mm apart at either end of the aerofoil. One pair of spigots were free to slide in bushes fixed in the aerofoil while the other pair were threaded to lock them in position. This arrangement prevented spanwise forces being transmitted to the strain gauge balance. It was decided to manufacture all three segments for each aerofoil from the same laminated block of Jelutong. The block was assembled by laminating together chordwise 1" thick planks of varying heights corresponding to the upper surface profile. The lower surface and front of the laminated block were skimmed flat to give a datum for the horizontal borer used in the machining operations. Figure 1. shows the method chosen to separate the inner segment upon which the forces were measured and the outer segments which were fixed directly to the walls of the test section. The spanwise width of both outer segments was set at 200mm. This dimension was based on the width of the junction vortex near stall observed using flow visualisation in the previous test series[3]. This width of outer segment was checked to ensure that it gave minimal displacement of test section wall when subject to aerodynamic forces.

The tongues of wood from the inner segment which pass through the outer segments have a 5mm clearance all round to prevent forces being transmitted between the inner and outer segment and affecting the balance readings. There is clearance between the end of the tongue and the test section wall for a similar reason. The inner segment span was set at 594mm. The presence of a gap between the inner and outer gap will alter the two-dimensional flow over the aerofoil. An investigation of the effect of a gap [5] on drag measurement suggested that a total gap of 1mm would be acceptable given the dimensional stability of the test rig working section width.

The method of manufacture was as follows:

1) Lay and glue up laminated 1" planks.
2) Skim lower and front surface to provide datum for horizontal borer.
3) Cut off outer segments from either end.
4) Machine face to length all three segments.
5) Dowel segments together.
6) Bore out clearance hole for tongue in two outer segments.
7) Bore out hole in end faces of inner segment 75mm deep to accept tongue.
8) Manufacture and Glue tongues in position
9) Bore two holes in outer segment faces to accept 45mm long , ⌀ 30 brass bush.
9) Bore two holes in each inner segment face to fit 85mm long ⌀30 brass bush.
10) Dowel segments together.

11) Manufacture profile template using aerofoil drawing and fit template to locate on bush holes.
12) Final profile by hand.
13) Glue bushes in position.
14) Coat with black resin paint for smooth finish.

The total working time for each aerofoil was of the order of four man weeks of labour. The finished aerofoils were dimensionally accurate with a parallel gap between the inner and outer segment faces and with an overall smooth surface.

3. TWO DIMENSIONAL AEROFOIL TEST RIG (SPLITTERS)

3.1 Introduction

The test rig to obtain two-dimensional data was developed as a result of a U.K Department of Energy contract to determine the effect of trailing edge truncation on thick sectioned aerofoils [3]. The rig was designed to fit into the working section of the 11'*8' low speed wind tunnel at Southampton University.

Figure 2 shows the overall layout of the test rig. Two floor-to-ceiling sections split the flow into three vertical sections, hence the designation of 'splitters'. The purpose of the division of the working section flow is to provide a uniform two-dimensional flow in the centre section with a significantly smaller span (1 m) than the overall width of the tunnel. This reduces the cost and problems of constructing and mounting the aerofoil sections for test. To compensate for the blockage effect of the aerofoil deflecting flow away from the inner section to the two outer sections each of the splitter test rig has a large floor-to-ceiling trailing edge flap which can be adjusted to maintain a constant dynamic pressure(q) in all three flow sections at a given wind tunnel dynamic pressure setting.

3.2 Construction

The splitter aerofoils are constructed from a sturdy skeletal frame of aluminium angle and plywood covered with a surface skin of 1/4" ply. Each splitter is assembled from three sections: a nose section; middle or centre section containing the dynamometry and aerofoil mounts; aft section with trailing edge flaps. The overall dimensions of the splitter assembly is 2.55m high and with a full length of 3.68m and width of 1.33m.

The aerofoils are mounted between two, three component strain gauge balances specifically designed for the test rig. The balances measure Lift and Drag forces and Moment torques and are described in detail in the Truncated Aerofoil test report[3]. As shown in Figure 2. corner fillets are used to improve the flow quality in the centre section. Strip fluorescent tube lighting is mounted in both the upper and lower fillets attached to the middle section. The whole assembly is attached to a 3/4" ply baseboard which is located along the centreline of the wind tunnel working section floor. A similar board of ply is used as a top plate and fixed to the working section roof. A small upstream window in the top plate is provided to allow access for a video camera.

The three splitter sections are joined together by overlapping ply on one edge being screwed into position onto the aluminium framework on the adjacent section. Threaded tie bars were found to be necessary to pull adjacent sections into the correct vertical orientation relative to each other.

Each splitter section has an access panel at both the top and bottom to allow the 20mm long 5mm HexHd bolts that attach the section to be fitted to the floor and roof boards. These access panels are made from 1/4"ply and are mounted on the skeletal frame by 15mm long by 4BA C/SK screws once the splitter section has been fitted in the tunnel.

The detail of the aerofoil mounting area in the central splitter section is shown in Figure 3. The side facing in towards the aerofoil consists of a 600mm diameter 6.5mm thick aluminium disk. This is secured to a surrounding plate, also aluminum, from within the splitter by three equispaced toggle clamps which pull up against a lip on the surrounding plate. The outer side of the middle splitter section has a 750mm by 750mm removable 1/4"ply panel which is fixed in place by six externally mounted toggle clamps.

The balances are secured to a levelled plate which ensures that they are set on the same centreline and are horizontal. A yoke arrangement is bolted to the balance as shown in Figure 4. This is attached to the aerofoil model through two spigots 150mm apart. On one side these fasten into tapped bushes in the aerofoil whereas on the other side they are a sliding fit. This prevents transmission of spanwise forces to the balances. The yoke has a collar arrangement which allows the aerofoil to be set to the required angle of attack relative to the horizontal using an inclinometer. The spigots pass through clearance holes in the aluminum disk which is also rotated when setting the aerofoil and then clamped in position.

3.3 Installation

The following section described the individual steps necessary to install the splitters into the 11'*8' wind tunnel. The time that should be allocated will depend on the number of people available and their collective experience of installation of the splitters. Before installation all necessary parts should have been identified and checked. This includes making sure sufficient numbers of bolts and screws are available. The key steps and the order in which they should be executed have been identified. This description is as exhaustive as possible, however, some operations are a matter of adjustment and cannot be described in detail. Also included are details on the fitting of the wake traverse, the design and construction of which is described in Chapter Four. No information is given for electrical and pressure connections outside of the working section as these are covered in Chapter Five. The time taken to install the splitters is between 1 and 2 working days.

Operation	Description
1)	Lift into position onto the platform outside the tunnel working section the six main splitter sections,floor and ceiling boards.
2)	Bolt floor board into place using four lengths of studding.
3)(i)	Erect two centre section splitter sections using aluminium guide plates on floor board.
(ii)	Bolt sections to floor board.
(iii)	To hold centre sections in place fix four aluminium crosspieces between them.
4)(i)	Erect two nose splitter sections.
(ii)	Bolt to floor.
5)(i)	Manouevre aft sections adjacent to final position.
(ii)	Feed four pressure tubes from aft section through into the centre section balance area on the left hand side.
(iii)	Fit Wake Traverse toothed belt onto the pulleys on both aft section splitters.
(iv)	Move aftsections into final position and bolt to floor.
6)(i)	Use threaded tie bars to align section relative to each other.
(ii)	Bolt together nose,centre and aft sections on each side.
7) (i)	Slide roof board from the downstream end of the rig along the gap between the top of the sections and the tunnel roof.
(ii)	Use aluminium packing plates to fit roof board in exact position.
(iii)	Bolt splitter sections to roof board.
8)(i)	Bolt in place fore and aft boards on both floor and ceiling.
(ii)	Bolt in place removable panels for both sides for the Nose section (upper and lower), Centre section (lower only), and Aft section(upper only).

(iii) Attach central section fillets (lighting units).

(iv) Take perspex covers off lighting units, wire internal cabling, fit fluorescent tube and replace cover.

(v) Fit remaining fillets in position.

9) Wake Traverse

(i) Fit belt tensioning device.

(ii) Feed pressure tubes through into the instrumentation strut.

(iii) Ensure pressure probe holders are on strut.

(iv) Fit instrumentation strut between two leadscrews

(v) Tension belt when instrumentation tube is levelled.

10) Trailing Edge Flap Control Rod

(i) Open covers on upper part of flap.

(ii) Feed studding through attachment for each side making sure it passes through cover.

(iii) Attach studding through holes, using washers, in walls of working section.

(iv) Adjust studding until flaps centred relative to each other.

(v) Fit taper pin to join the studding.

(vi) Close covers.

(vii) Fit handle.

(viii) May need to be readjusted after initial tunnel calibration if flow not the same in two outer sections.

11) Balances

(i) Bolt balances in position ensuring that they are level and on the same axis.

(ii) Feed channel leads up through centre section

(iii) Fit yokes onto balances.

(iv) Fit aluminium discs using toggle clamps.

(v) Remove four aluminum cross pieces.

12)(i) Feed wiring and pressure tubing into rig as per section 5.4

(ii) Fit remaining removable panels.

(iii) Fit wake traverse beltcover in position avoid applying loads to it.

(iv) Use masking tape to seal joints within working section of the splitters.

3.4 Calibration

The dynamic pressure in the working section of the splitters will be greater than the tunnel dynamic pressure so to determine the central working section dynamic pressure a calibration has to be carried out. Four calibrated standard Pitot–Static probes were used, two of which were mounted on the inner faces of the splitter centre section and the other two on the outer faces. The location of the probe heads was on the 3/8 chord line of the aerofoil. The calibration should be carried out for the range of tunnel speeds to be used. At each tunnel speed setting the trailing edge flap control handle should be rotated until the four manometer dynamic pressure readings are identical. This dynamic pressure is the calibrated value to be used when the aerofoil is present. The flaps will be adjusted until this calibrated value is the dynamic pressure in the outer section using the same tunnel speed setting as the calibration experiment.

3.5 Operation

This section covers : aerofoil installation, setting of angle of attack and flap adjustment and roughness strips. For operation of the data acquisition system the reader is referred to Chapter Five.

1) Aerofoil Installation

The complete aerofoil is placed on special mounts on a hand operated hydraulic trolley. The trolley and aerofoil are wheeled into position from the upstream end of the splitter working section. The spigot bar on the balance yoke is loosened and set at about 45° to the horizontal. Once in position the aerofoil is raised until the spigots can be inserted through the spigot bar, clearance holes and into the bushes in the end face of the aerofoil. The special mounts on the trolley ensure the aerofoil is at the correct angle. Precise alignment requires considerable adjustment. It is usually easier to insert the sliding spigots first into the inner segment of the aerofoil and then to screw in the fixed pair. The outer segments are bolted in place using holes drilled in the aluminium disk. Spacers should be used to ensure that the inner segment is centred and there is an even gap at either end.

2) Angle of Attack

To set the angle of attack to the required incidence the allen nut on the yoke hub is loosened on both sides. The aluminium disk toggle clamps are then undone. The whole aerofoil can then be rotated to approximately the desired angle of attack. The discs are clamped up and an inclinometer placed on the spigot bar on one side. The inner segment of the aerofoil is then adjusted until it is at the correct orientation. The yoke hub is locked up and the process repeated on the other side. The outer segments are then moved until there are no step discontinuities in surface across the gap between the inner and outer segment of the aerofoil.

3) Flap adjustment

Rotation of the flap control handle clockwise will increase the dynamic pressure in the outer splitter section and reduce the dynamic pressure in the inner splitter section.

4) Roughness strips

These were produced using a 1m strip of 6.5mm wide double sided adhesive tape. One side of the tape was exposed and carborundum grit of the required grade distributed over it to give a light covering of particles (approximately 10% of the surface area). The backing on the other side is gradually removed as the length of tape is laid spanwise across the aerofoil at the desired chordwise location.

For the roughness tests conducted in this investigation strips with 80 grade grit were applied to both surfaces of the aerofoil at 5% chord downstream from the leading edge across all three segments of the aerofoil.

4. WAKE TRAVERSE INSTALLATION

4.1 Requirement

To give a better measurement of drag for unstalled two-dimensional aerofoil flow it was decided to include a wake traverse mechanism within the two-dimensional aerodynamic testing rig. The method chosen for obtaining the pressure information had to be capable of measuring pressures for a number of wake positions relative to the splitters. The vertical position of the wake being dependant on the aerofoil angle of attack. It would have to provide sufficient vertical resolution to allow an accurate determination of the total momentum deficit.

The calculation of aerofoil profile drag by measuring the total momentum deficit through the wake of an aerofoil is a standard method commonly referred to as a wake traverse. As a method of measuring drag its main advantage over a strain gauge balance reading of drag force is that there is no need to correct the drag measurement for the effect of the walls of the wind tunnel or the low aspect ratio of the aerofoil. One of the original papers detailing a wake traverse as a technique for measuring drag of a wing in flight is detailed in reference[6].

4.2 Design

A variety of methods have been used for obtaining a traverse through a two-dimensional wake. These vary from a stationary comb of total head probes which fully span the wake to that of a single Pitot probe which is mechanically moved though the wake.

The restriction of the overall method chosen having minimal impact on the flowfield in the test section and being able to fit into the existing splitters dictated the methods available for use. A recommended distance downstream from the trailing edge of the measuring plane is approximately 0.7 of the aerofoil chord [7]. Equalisation of static pressure inside and outside the wake has occurred by this location but dissipation of the wake is not too large.

It was decided there was adequate space within the aft section splitter in front of the trailing edge flap for installation of a traverse mechanism. This would minimise the disruption to the flow while also providing a sturdy base for holding the wake traverse probes in the flow. Symmetry considerations suggest that the wake should be measured in the centre of the test section. Therefore, a probe supported between the two aft section splitters would give a symmetrical influence on the flow and an effective means of support for the vertical movement of the traverse. An additional benefit of a support spanning the whole width of the inner section is that it would allow a series of probes to be mounted to observe possible spanwise variation in the flow.

The obvious mechanical method for obtaining the vertical motion of the pressure probe is that of a leadscrew mounted in the side of each aft splitter driving a nut. This gives a smooth linear motion with no restriction on possible position within the limits of the leadscrew. The drive mechanism needs to have minimal backlash or lateral movement so that a vertical position is precisely defined and repeatable. For the wake thickness to be traversed a suitable tolerance for vertical motion is less than 0.1mm. The use of a leadscrew either side will reduce possible slackness in the system. The automation of the traverse required some form of electrical motor or motors to drive the two leadscrews simultaneously. To ensure the leadscrews rotate an identical amount they must be directly coupled together. A pulley and toothed belt mechanism across the working section floor between the two splitters gives such a coupling. This necessitated a slight disturbance to the flow, however, with the use of an appropriately shaped cover this could be minimised. A 1" wide toothed timing belt was used. These belts have no backlash if suitably tensioned and provide an economic method for transferring high loads smoothly. A third pulley with two pinch rollers located in the centre of the section tensions the belt for operation in both directions.

The choice of motor to drive the mechanism was made based on experience gained in the Ship Science Department in the construction of a three-axis model ship cutting machine [8]. This used DC stepper motors for each axis to give an accurate low cost positioning system without the need for expensive feedback control circuitry. A similar type of stepper motor was therefore chosen to drive the traverse. The vertical resolution for a single step of the a motor is dependent on the pulley ratio between motor and leadscrew and the pitch of the leadscrew. Figure 5 shows the design layout chosen. The motor and pulley drive fit into a cavity next to the floor on the right hand side aft splitter. The sizing of the stepper motor is dependent on: the requirement for maximum starting torque to overcome friction; and the need to match the rotational inertia of the motor core to the rotational inertia of the rest of the drive system. The inertia matching ensures that the motor and system to respond together and that the overall system does not stall and then resonate at the chosen stepping speed. The motor purchased has a starting torque of 3Nm when wired in series. It has a resolution of 400 half–steps per revolution. The extra precision of a rolled leadscrew was considered unnecessary for the system and so a standard thread studding (M20 and 1.5mm pitch) was used for the leadscrews. A 50mm long brass-nut for each leadscrew was machined to give a close fit onto the studding. The studding pitch of 1.5mm and pulley ratio of 1:2 between the motor and leadscrew gives a vertical resolution of approximately 0.003 mm for each step of the motor. This allows the open loop control to cope with the potential problem of the stepper motor missing steps and still not impair the desired positioning accuracy of 0.1mm. A simple coupling was used between the pulley unit shaft and the leadscrew to allow for possible misalignment on assembly.

The arrangement for the instrumentation tube spanning the the working section is shown in Figure 6. A hollow circular cross–section tube gave a strong structure with aerodynamic forces independent of flow angle. The four pressure probes mounts were slid onto the tube prior to assembly and the locked in position using a pinch nut. The connections between the instrumentation tube and the leadscrew nut were designed to compensate for possible misalignment of the two leadscrew axes. One end being free to slide. The four pressure tubes for the probes are fed through the instrumentation tube onto the connection fixed to the nut on the left hand side leadscrew. To take up the slack of the pressure tubes caused by the vertical motion of the instrumentation tube a tensioning device had to be included. Figure 7 shows the arrangement chosen to fit into the internal splitter space with a minimum disruption to the internal structure of the splitter aft section. The restoring force of the system was provided by a length of 6mm elasticated cord fixed at one end to the top of the aft section.

Figure 19 shows the pressure probes used. Both the Pitot and Static probes were manufactured from 1/4" stainless tube. For the Pitot probes, following reference[13], the probe tip was flattened to give a 0.5mm wide gap. This reduces the uncertainty in determining the exact location were the pressure is measured from without increasing the response time. A single hole Static probe with a rounded end was used. The pressure probes were calibrated and were found to give an accurate determination of dynamic pressure.

4.3 Construction

The technical support for the construction and fitting of the wake traverse mechanism were provided by the joint Aeronautics/I.S.V.R./Ship Science Department workshops. A full set of working drawings and parts list were provided for use. The main assembly was divided into a series of sub-assemblies for construction and fitting. These are described below:

(1) Leadscrew & Nut - Figure 8

The leadscrew was a standard M20 studding with a 1.5 mm fine pitch. Flats were machined at one end to allow the simple coupling to be attached. A single ball-race was used to support the leadscrew at either end. The load conditions and low rotational speed did not require anything more substantial. A threaded collar was used to provide a plain surface between the leadscrew and the ball race. The two sides made from 6.5mm thick angle give a rigid structure when bolted to the bearing housings. The exposed gap onto the flow was kept as small as possible at 25mm. To prevent possible seizure the nut was machined from brass to a tight tolerance which gave a minimum amount of slack.

The leadscrew assemblies were fitted by removing the 1/4" ply surface and cutting back into the three horizontal aft section 1" ply structural member to give sufficient clearance for the leadscrew and nut. The assembly was screwed up against the three structural members so that it fitted flush with the surrounding 1/4" ply splitter skin.

(2) Pulley & Motor - Figure 9.

A simple construction of aluminium angle and plate was used for both the left and right hand side pulley sub-assemblies. The right hand side assembly has two bearing races to support the mild steel 020 drive shaft. The ballbearing races, housing and collar were of identical design to those used for the leadscrew assembly. The pulley was attached by a taper lock hub.

The left hand side assembly was more complex due to the limited space available for fitting the stepper motor within the aft section splitter. The same basic components were used. The stepper motor attachment holes were slotted to allow the drive belt to be tensioned.

Both assemblies were bolted to the structure of the aft splitter and clearance slots cut in the splitter wall for the connecting pulley.

(3) Belt tensioning Device - Figure 10.

The belt tensioner consists of three plates bolted to the splitter floor. Each of the outer plates has a cylindrical roller which pinches on the back of the toothed belt. The central plate is slotted to allow the position of the pulley mounted on it to move to tension the belt. The entire assembly is covered by a wooden fairing to minimise the disturbance to the flow.

4.4 Electrical Controller

The method of translating a command issued by a computer to move the instrumentation tube through a vertical distance into the electrical power pulses received by the stepper motor had been explored during the design of the three-axis cutting machine [8]. An identical system layout was, therefore, used for the wake traverse. Slight modifications had to be made as the power requirement for the traverse was higher and needed a power supply and motor driver sized accordingly.

The overall system layout is shown in Figure 11. The individual units: motor driver, controller, and power supply were manufactured by Digiplan and wired together in the Aeronautics Dept. Electronics Laboratory. The whole control circuitry was assembled into a single box, referred to as the 'Wake Traverse Controller'. A standard RS232c 25pin 'D' socket is used for the computer controlled input. The interface board decodes the instructions from the computer into a series of pulses which are used by the Motor driver to power the stepper motor through the desired number of steps. The instruction set used by the controller and the operation of the individual circuitry is detailed in reference[9].

An emergency stop button is provided. Two normally closed lever arm limit switches are used at the upper and lower extent of the traverse motion. They are fitted internally within the right hand side splitter and are activated by the leadscrew nut. The limit switch wires and power to the motor are combined into a multi-wire cable which plugs into a socket next to the stepper motor in the aft section splitter.

5. DATA ACQUISITION

5.1 Introduction

The two-dimensional lift,drag, and moment coefficients for the test aerofoils were obtained from: two three-component strain gauge balances; traverse of a total head pressure probe through the wake in pre-stall conditions; and measurement of tunnel conditions. This chapter describes the method by which these three sources of data are acquired and then reduced to give the required coefficients.

5.2 System Design

The use of strain gauge bridge circuits as the principle method for measuring forces requires an accurate means of recording the small bridge voltages on a time-average basis. A computer controlled data acquisition system is the most efficient and reliable method of recording the large amounts of data generated. In the truncated aerofoil series of tests the system used was based on a Commodore Pet computer, however, for these series of tests it was decided to replace this computer with a standalone RM Nimbus PC. This would allow far greater programming flexibility and allow data and programs to be transported directly on to the Ship Science Department's network of 20 RM Nimbus PC computers.

The standard strain gauge bridge circuit is shown in figure 12. The bridge output voltage varies linearly with the load applied once the offset or 'zero' load voltage has been subtracted. For the strain gauge balances used the output voltage is of the order of 1000 volts. The system chosen needed to be able to measure with an accuracy of 1μ volt.

There are two common methods for automatically recording strain gauge bridge outputs. The more conventional method is to feed the bridge output into a linear amplifier with adjustable gain to give an output voltage in a range suitable for analogue-to-digital conversion. This digital representation of the voltage is then either read directly or used as input into a computer. More advanced bridge amplifier units allow the 'zero' load voltage to be effectively removed by using an offset voltage at calibration so that for zero load the bridge reads zero volts. This approach entails the use of an amplifier unit and analogue-to-digital converter for each strain gauge bridge circuit. The process of calibrating the amplifier for a particular strain gauge bridge can be time consuming especially if the circuitry has a tendency to 'drift' with changes in room temperature. For the aerofoil

application the zero load voltage is used to remove the weight of the aerofoil from the force measured by the balance. This would imply that for each change of aerofoil the individual amplifier circuits would have to be recalibrated. The department has available a CED 16 channel A-to-D converter which would have given suitable accuracy and a high data acquisition rate, however, insufficient amplifier units were available at that time for a complete system.
One drawback would have been the probable need to redetermine the influence effects of the three channels,(lift,drag,and moment) on each other for the balance.

The actual method chosen is shown schematically in figure 13. This uses a single channel accurate digital voltmeter which performs the task of measuring the voltage and converting it into a form which can be read by a computer via an IEEE standard interface. A 16 channel automatic switching unit is used to present each channel in turn to the digital voltmeter (DVM). Although the rate of data acquisition using this system is relatively slow (20-30 readings a minute) this is sufficient for the steady state measurement of voltages from the strain gauge balances. As the strain gauge bridges circuits are not individually calibrated the power supply needs to be continuously monitored and the zero load voltage should be read prior to and on completion of any test.

The combination of accurate Digital Voltmeter and multiplexing switch is available as part of the 11'*8' wind tunnel setup. To connect the RM Nimbus PC as the control computer all that was required was a GPIB (IEEE compatible) card to be fitted in the PC to allow the output of the digital voltmeter to be accessed. Special 'driver' software is required to allow data to be read down the IEEE bus and be used by a program.

The scanivalve control unit was already part of the wind tunnel setup and wired into the multiplexing switch As described in 4.4 the Wake Traverse controller is connected directly into the RM Nimbus via a Piconet module. The specifics of the hardware units used and the software to drive them are described in the following two sections.

5.3 Hardware

The data acquisition hardware wiring interconnections are shown schematically in figure 13. The design, calibration and determination of the influence matrix for the two three-component strain gauge balances is described in ref[3]. For these tests the balances were modified by the addition of 5 pin -240° DIN plugs for each of the individual channel leads. All six leads were plugged into a single junction box which connected via a 6metre single cable into the internal wiring bus of the wind tunnel data acquisition system. The stabilised power supply was connected to all six strain gauge balance through a socket in the junction box. The monitoring of the power supply seen by the strain gauge bridge should be done as close as possible to the bridge to avoid resistance losses in the power supply lead. The most convenient location for this was at the junction box.

A four channel 32 port Scanivalve Inc. rotary pressure transducer was used to measure pressures from the four Wake Traverse pressure probes. As only four pressure measurements were required one pressure probe was connected to each of the four pressure transducers and the Scanivalve left in a fixed position for the tests. This removed the need to take a compressed air supply into the left hand splitter to rotate the scanning mechanism. The scanivalve power supply and bridge circuitry is part of the wind tunnel setup and the four channels of output can be fed directly into the mutliplexing switch.

The eleven channels (6 strain gauge bridges, 4 pressure transducers, and 1 power supply monitor) were hardwired into the front panel of the 16 channel rotary switch (Minate Solartron 7010 driven from a Schlumberger DVM 7061). The DVM was connected into the RM Nimbus PC via an IEEE standard lead connected into a Nimbus GPIB card fitted in the back of the PC. The RM Nimbus PC used had a two 3 1/2" minifloppy disk drives to allow backup of data files.

The wake traverse controller is controlled from the Nimbus PC via the Nimbus Piconet serial module which is a serial RS232c protocol connection. The wake traverse controller was connected directly to the stepper motor and limit switches through a 5 metre cable taken into the tunnel working section. The working of the Wake Traverse Controller has been described in section 4.4.

5.4 Software

The software for the wind tunnel tests was developed on the RM Nimbus PC using the Prospero implementation of Pascal. Use of this version of Pascal allowed procedures already developed for driving the Wake Traverse Controller and the GPIB interface card to be used. Prospero Pascal allows groups of procedures to be written as separately compiled segments which can be linked into different programs.

Two programs were required, one as a development and initialisation program allowing the various interfaces to be tested and the other program for actually running the Wake Traverse during an individual test run and acquiring the data. These two programs INITIAL.PAS and EXPERIMENT.PAS respectively used the following the four segments which are described below. Appendix B contains listings of the two programs and the segments :

(1) TRAVSEG.PAS

Contains procedures which:

- initialise the Wake Traverse Controller.

- allow the wake traverse to be driven up and down 'manually' by entering specific control codes understood by the wake traverse controller. The function of these control codes is detailed in reference [9].

- error checking routine used to make sure the device is initialised properly, to move the instrumentation strut away from the limit switch if they are encountered, and to recover after the emergency stop has been activated.

- allows the absolute position of the strut to be defined by driving it down until it encounters the lowermost limit switch and then moving up a fixed distance.

- moves the probe through a given distance up or down, expressed in millimetres.

(2) DISPSEG.PAS

A series of procedures and functions which allow specific data types to be entered from the keyboard within a certain range of values. Also included are procedures to output information onto the screen. This segment is described in reference[10] and so is not included in Appendix B.

(3) GRAPHSEG.PAS

Procedures to generate graphical information on the screen. Allows a specific output screen to be created. Figure 14 shows the screen used in these experiments. The screen was refreshed after each new data point had been acquired.

(4) MINSEG.PAS

This segment carries out the same basic procedures for the Minate DVM and rotary switch as the TRAVSEG.PAS segment does for the Wake Traverse. The control codes used are described in reference[11]. In addition, this segment includes a routine which reads the output from the DVM. This information is sent as a string of characters. A procedure is used to convert this string of characters into a real number equal to the voltage read. The 'driver' software for the actual interface is taken from the GPIB.PAS segment supplied with the Nimbus GPIB interface card and so GPIB.OBJ has to be included in the link statement when MINSEG is used.

Figures 15 and 16 show the flow charts for the programs INITIAL.PAS and EXPERIMENT.PAS respectively. The structure of INITIAL.PAS is straightforward, the four menu options allowing the Minate and Traverse to be tested separately and a test data initialisation file to be created.

EXPERIMENT.PAS takes the user step by step through a test run. First by asking for the individual test set up. That is the angle of incidence, aerofoil type and whether there is a roughness strip. The user then enters the atmospheric conditions and tunnel dynamic pressure for the test. The program then acquires the windoff voltages for the balance and transducers and stores them. The user is then requested to start the tunnel up and adjust the trailing edge flaps until the desired splitter dynamic pressure is obtained. Test data is then acquired until the user decides sufficient has been obtained. For each test point the individual balance and transducer voltages are acquired six times and the average value stored. The program outputs a suggested next step in millimetres for the traverse. The user can override this to ensure the velocity profile is recorded in sufficient detail. On completion with the windoff a reading of the balance and transducer voltages is again made.

Test data for each test run was stored in three different files FINALXX.DAT, PROCXX.DAT, and RAWXX.DAT. The XX designation referring to the actual test run number designation. Appendix C shows example data files for test run 107. FINAL.DAT stores the specifics of flow speed and atmospheric conditions, aerofoil details and general test information. RAW.DAT stores the absolute voltages recorded for each of the eleven channels for each data point and also the initial and final 'zero' voltages for the strain gauge balance and pressure transducers. PROC.DAT stored the reduced data created during a run and used to refresh the screen display. At the end of each test run the three data files generated were copied onto another disk to provide a back up copy.

5.5 Splitter Wiring Installation

Removable panels in the roof of the 11'*8' working section gave access for the main power cabling for the fluorescent strip lighting and for the three leads for each of the strain gauge balances. The cables pass from the access panels into the top removable panel on the outside of the middle splitter section. The junction box for the balances is placed centrally on the top side of the working section roof. Figure 17 shows a sketch view of the various cable routes outside the tunnel working section.

The wiring used into the Minate front panel is detailed in figure 18. The wake traverse cable is fed through a hole in the tunnel wall and taped along the floor to the right hand side aft splitter section lower removable panel where it is plugged into a multipin socket adjacent to the stepper motor unit. The scanivalve control lead is fed down through the left hand side access panel into the balance area in the middle panel.

The Scanivalve unit containing the four pressure transducers (a box 6" by 4" by 2") was placed in position in the lower aft most corner of the balance area (shown in figure 3).

5.6 Pressure measurement

The tunnel dynamic pressure was measured using an accurate Betz manometer with water as its working fluid. Atmospheric pressure was measured from a wall mounted mercury barometer. The pressure measurement layout used within the splitter sections changed between the calibration phase and the experimental tests.

During calibration four standard calibrated pitot–static pressure probes were mounted one on each of the faces of the splitter centre sections. The probes were located on the centreline of the tunnel at the same position as the 3/8 chord point of the aerofoils to be used. All four probes were connected to a tilted manometer bank and the outer right hand side pressure probe was also connected to a Betz manometer. The splitter working section dynamic pressure was calibrated for a given overall tunnel dynamic pressure by adjusting the trailing edge flap position until all four pressure probes gave the same dynamic pressure reading.

This dynamic pressure head was read from the Betz manometer and used when setting the flaps with an aerofoil in the working section.

For the experimental tests the two inner pitot-static probes were removed to allow the aerofoil to be mounted. The right hand side outer section probe remained connected to the Betz manometer and the static line of the left hand side probe was used as a reference pressure for the pressure transducers. Figure 19 shows how the three pitot probes and one static probe of the wake traverse were tubed together. The three pitot probes were levelled using a bubble spirit level. One was positioned on the centreline and the other two 160mm either side. The static probe was mounted 50mm to the side of the centre pitot probe. This arrangement had been validated in a 1' * 1' wind tunnel against a standard pitot-static probe.

5.7 Operation

On switching on the computer it had to boot itself from a system disk. A simple batch file was used to do this. Three programs CONFIG.EXE, PICONFIG.EXE and ICONFIG.EXE ensure that the PC was configured correctly for the Piconet, Wake Traverse Controller and GPIB board respectively. Further more detailed information on the exact configuration used can be found in references [8],[14].

To verify the Wake Traverse and Minate Voltmeter were operating correctly the program INITIAL.EXE was run and a test data file created. For each test EXPERIMENT.EXE was run the three data files created were stored on 3 1/2" minifloppy disk in drive a: and on completion of a test run these files were copied onto a backup disk in drive b:.

6. DATA ANALYSIS

6.1 Experimental Programme

There were three weeks (15 working days) available for the installation, commissioning of the test rig, and testing of the five SSS series aerofoils and the LS(1) 0421 MOD aerofoil. In total, a 130 individual test runs were carried out . Table 1 show the experimental conditions for the tests conducted on each aerofoil. The aerofoils were tested at two tunnel dynamic pressure settings corresponding to chord Reynolds Number of $2*10^{6}$ and $1.6*10^{6}$ for a range of aerofoil incidence settings. Roughness tests were conducted at a few representative points.

Both the LS(1) 0421 mod and the SSS-125 aerofoils were tested in an upright and inverted sense. This was used to determine the flow angularity correction within the centre test section with an aerofoil in position. Detailed wake traverses were carried out for all the pre-stalled incidence tests. The screen displayed wake traverse velocity profile showed whether stall had occurred. For the SSS-127 the aerodynamic performance post-stall was investigated in detail.

At the end of each working day the data files generated were transferred onto the Department's computer network prior to detailed analysis.

6.2 Analysis Software

The program ANALYSIS.PAS was used to analyse the test data. This program was written in TurboPascal version 4 mounted on the Department's RM Nimbus VX computer. The advantage of TurboPascal is its quick compilation time and the large number of graphics,menu and numerical method procedures readily available. A large number of the procedures from the ProPascal EXPERIMENT.PAS could also be used directly.

A menu-layout was used to allow various types of analysis to be conducted, ranging from looking at the actual voltages for a given test point to calculating the profile drag from the wake traverse. It also facilitated the rapid development and expansion of the program. Figure 20 is a schematic of the structure of the program. The more important procedures will be briefly described; the data reduction, aerodynamic correction and Wake Traverse Drag calculation are detailed in sections 6.3, 6.4 and 6.5.

6.3 Data Reduction

The raw data stored in the files RAWnn.DAT consisted of the voltages taken from the Minate voltmeter for each of the eleven channels. To reduce this into the form of Lift,Drag and Moment coefficients the following operations were carried out:

(i) Zero Voltage Correction - To take account of possible zero drift during a run the the zero offset voltage was taken to be the average of the initial and final windoff readings from the balance.

(ii) Calibration Voltage - As the supply voltage fluctuated from the value used when the balances were calibrated it was necessary to scale the voltages read to those at a supply voltage of 8.0125V.

(iii) Influence Correction - The individual lift,drag and moment channels have a slight influence on each other. In the initial calibration of the balances detailed in reference [3] a 3*3 influence matrix was determined. This has been used to remove the dependence of the three channels

(iv) Force/Moment - From the balance calibration ref[3] a scaling term relating voltage to actual force and moment for each channel was determined. This was used to produce force and moment output in absolute S.I. units.

The forces and moments were non-dimensionalised by calculating the air density based on atmospheric temperature and pressure, and freestream velocity found from the splitter dynamic pressure reading. The total force or moment was taken to be the sum of the of the two balance readings. To change the moment measured on the balance which is about the 3/8 chord point to that about the 1/4 chord point the following expression was used :

$$Cm_{\frac{1}{4}c} = M_l + M_r = (1/8)*Cl + \Delta\ Cd$$

where Δ is the displacement of the balance centreline from the aerofoil centreline.

6.4 Aerodynamic Correction

The non-dimensional Cl, Cd and Cm obtained for the flow in the bounded three-dimensional test section need to be corrected to obtain the value for an unbounded flow over a wing of infinite aspect ratio. The corrections used for pre-stalled flow are those described in Rae and Pope reference[7] which were used previously in the Truncated aerofoil tests[3]. They are as follows:(1) Flow angle correction

The difference in angle between the flow through the section and the horizontal results in the rotation of the aerofoils resultant force vector. The flow angle can be determined by testing an aerofoil upright and inverted the displacement of the two Cl-Alpha curves giving the necessary correction to the flow angle. However, for these tests a it was found that a better correction was obtained by comparing the wake drag measurement and balance drag at zero angle of attack for the LS(1) 0421 mod aerofoil.

(2) Solid Blockage

The aerofoil in the working section reduces the available flow area and the velocity around the aerofoil correspondingly increases. This effective velocity increase is dependent on the aerofoils thickness, thickness distribution and chord. Although the solid blockage correction for the wind tunnel used was small it was included for completeness. The expression used was that following Glauert[15].

$$\epsilon_{sb} = 0.822\ \lambda_2\ (t/h)^2$$

(3) Wake Blockage-preStall

The reduced velocity in the wake behind the aerofoil through continuity of mass results in the velocity outside the wake being greater than freestream. From a potential flow source-sink analysis the increment in velocity is as follows [7].

$$\epsilon_{wb} = \frac{(c/h)}{2} \cdot Cd$$

(4) Static Pressure Gradient

The thickening of the boundary layers along the walls of the tunnel section results in a static pressure gradient. From Glauert[15] this results in a horizontal buoyancy force acting in the drag direction. The required correction is:

$$\Delta D = \frac{\pi}{2} \lambda_2 t^2 \frac{dp}{dl} \text{ (N/m)}$$

where the pressure gradient (dp/dl) was taken to be the value measured in (3).

(5) Streamline Curvature

For the bounded flow in the tunnel the aerofoil appears to have more camber than it has. Therefore, Cl,Cm and alpha have to be correspondingly reduced. A simple vortex image model is used to produce the standard corrections [7].

$$\Delta Cl = \sigma Cl$$

$$\Delta Cm_{\frac{1}{4}c} = -\frac{\sigma}{4} \Delta Cl$$

$$\Delta\alpha = \frac{57.3\ \sigma}{2\pi} (Cl_u + 4Cm_{\frac{1}{4}c}u)$$

where $$\sigma = \left(\frac{\pi^2}{48}\right) \left(\frac{c}{h}\right)^2$$

In these tests data was obtained after stall has occurred on the aerofoil. To obtain more sensible data for post-stall conditions where the flow is three-dimensional the following correction is proposed and has been used. It is based on actuator disk theory[16] to give an improved wake blockage correction. For stalled flow the wake behind the aerofoil occupies a significant area of the flow. This results in an increase in flow speed outside the wake. The wake traverse could be used to measure this speed increase. The analysis detailed in Appendix E by Dr.C.J.Satchwell derives the following expression for the equivalent unbounded freestream velocity Ucorr.

$$Ucorr = U_\infty + D/(2 * \rho * A_{act} * U)$$

Where U is the bounded freestream velocity found from a calculation of the wake area based on the wake edge velocity. A_{act} is the area of the aerofoil actuator surface.

The necessary correction to the non-dimensional coefficients is obtained by multiplying by $(Uo/Ucorr)^2$.

6.5 Wake Traverse Drag Measurement

The calculation of the wake drag requires the evaluation of the integral of momentum deficit through the whole wake. This can be expressed following the terminology of [6] as:

$$D = \rho \int u_1 \, (U_\infty - u_1) \, dy$$

where u_1 is the velocity in the plane of movement. Taking U as the velocity sufficiently downstream that the static pressure in the wake has equalised with the freestream two non-dimensional parameters g and p can be defined as:

$$g = \frac{\frac{1}{2} \rho U^2}{\frac{1}{2} \rho U_\infty{}^2}$$

and

$$p = \frac{P}{\frac{1}{2} \rho U_\infty{}^2}$$

P is the static pressure relative to the freestream value at the measurement plane. The final expression for non-dimensional drag Cd becomes when the appropriate substitutions for g and P becomes:

$$Cd = \int 2 \sqrt{g-p} \, (1 - \sqrt{g}) \, d \left(\frac{y}{c}\right)$$

The determination of Cd requires the measurement of g and p as a function of height y through the wake.

The four pressure readings from the scanivalve transducers allow g and P to be evaluated for the three spanwise pitot probes directly. The height of the traverse instrumentation tube is used for the position of the probe head. No correction is made for the deflection of the probe towards the side with lower velocity. The resultant correction for a probe of thickness 0.5mm is insignificant compared to the thickness of the wakes being measured. For the position of the pressure probes 0.5c downstream of the aerofoil trailing edge it was found that fluctuations in P were negligible across the wake and thus were ignored in the calculation of drag.

The process of calculating the profile drag was implemented as a series of procedures within the program ANALYSIS.PAS. They carried out the following:

(1) Determine the maximum and minimum heights which define the edge of the wake. Data outside these limits are not sent to be integrated but are used to calculate the wake edge velocity Uedge.

(2) The number of points obtained in any particular traverse through a wake was ~20. To give a more precise definition of the velocity in the wake for numerical integration a curve fitting was carried out. This used as a basis the profile through a wake used in [17] for a similar process.

Figure 21 illustrates the technique. The procedure determines the position of minimum velocity and then the edges of the wake to give the thickness of the two halves of the wake. This determines all the parameters for the curve to be found.

(3) The integrand was evaluated using the fitted curve profile for 60 points between the two edges of the wake.

(4) A quadratic numerical integration was performed for the generated data to give the non-dimensional Cd.

The velocity profile used gave an excellent fit to the data points obtained. This can be seen in figure 22 for the central spanwise station for test run 32.

7. RESULTS

7.1 Introduction

The results for each aerofoil are presented in a separate section. A comparison of their respective performance is made in Chapter 8. Following Von Doenhof reference[12] the aerofoil characteristics are plotted for Cl against incidence and for Cd and Cm against Cl. The calculated wake traverse profile drag is shown as separate points on the Cd against Cl plot. This is also done for the roughness tests. The post-stall corrections discussed in the previous section is used on all stalled data. The data points plotted are included as Appendix D.

The wake drag used is an average of the values at the three spanwise positions. Away from stall there was no marked difference between the positions. The flow stalled from the outer segments inwards so for some readings close to stall the centre section flow still allowed a drag reading to be made.

It was the progressive stalling of the aerofoils inwards which is thought to explain the discrepancy between the balance drag measurement and the wake drag which became more marked for the thicker aerofoils. The balance drag measures the average two-dimensional drag for the inner segment which may include areas of stalled or separated flow. The wake drag only measures the local two-dimensional drag of the particular spanwise position. For the centreline wake drag this is the defined two-dimensional Cd for low angles of attack when there are no circulatory features in the wake.

The balance drag includes a component of drag due to the gaps between the inner and outer segments again this is not included in the wake drag measurement. For the purpose of the data plots where there was a valid wake drag measurement it was used and in other areas (close to and post-stall) the balance reading was taken as being representative of the drag. Appendix D indicates whether a drag measurement was obtained by wake traverse or balance reading.

7.2 LS(1) 0421 MOD Aerofoil

The performance of the LS(1) 0421 mod is compared to that obtained by McGhee & Beasley [4] in their NACA report. The overall comparison between the two sets of data is good, especially for low angles of attack. For both the smooth flow at the Reynolds number of 2000000 shown in figure 23 and with a roughness strip attached at 5% chord shown in figure 24 both the lift and moment comparison are close enough to validate the test rig as a means of obtaining acceptable two-dimensional data pre-stall.

The drag reading is more sensitive to fluctuations from true two-dimensional flow, the balance reading drag reading diverging from those of McGhee & Beasley[4] for high angles of attack and low Cl. However, for low angles of attack both the balance drag reading and the wake traverse give excellent agreement even for the roughness tests.

7.3 SSS-121 Aerofoil

Results are only shown for a Reynolds number of 1.6*10↑6 in figure 25. The maximum Cl of 1.25 occurred at an angle of attack of around 12°. The drag rise commenced at a Cl of ~ 0.8 corresponding to an alpha of 5-6°. The minimum drag obtained from the wake traverse was 0.009 which was higher than the 0.0074 obtained from the balance.

Two roughness tests were carried out at a Reynolds number of 2*10↑6 for angles of attack of 0° and 6°. A significant drag increase of 50% was found for the 0o case with little change in Cl. However, at 6° the flow over the aerofoil as observed from the wake traverse output was stalled. The Cl changed very little from its 0o value but the balance drag had increased fivefold.

The change in lift-incidence slope marked A which occurs at an incidence of 6° appears to indicate the appearance of three-dimensional effects in the flow causing loss of lift over edge area of the middle segment of the aerofoil. This can also be seen in the large discrepancy between the balance drag and wake drag which occurs for angles of attack greater than 6°.

7.4 SSS-123 Aerofoil

Plots for Reynolds number of both 2000000 and 1600000 are shown in figure 26. The results for both cases are close up until 12° when the lower Reynolds number exhibits a large stall and corresponding dropoff in performance. The balance drag gave good agreement with the with the wake traverse for this series of tests with a minimum drag of 0.008 occurring at 2° which, interestingly, is lower than that of the 21% aerofoil.

As for the 21% thick aerofoil at about 6° angle of attack a change in the dCl/da occurred with a corresponding balance drag rise showing the influence of three-dimensional effects on the flow.

Three roughness tests were carried out for a Reynolds number of 2*10↑6 a=7°,11°,17°. For all these cases the flow exhibited stalled characteristics, although the drop off in Cl was not as marked as in the 21% case. The high drag is probably due to a breakdown of two-dimensional flow and edge effects dominating.

7.5 SSS-125 Aerofoil

Figure 27 shows the performance of the SSS-125 aerofoil for the two Reynolds numbers tested and with roughness strips attached at a Reynolds number of 2*10↑6. The performance of the aerofoil at the lower Reynolds number again drops off earlier there also being a pronounced change in dCl/da occurring around 6°.

The addition of a roughness strip causes a significant reduction of performance the wake traverse velocity profile showing stalled or separated flow even at a=2°. The sensitivity of the aerofoil to roughness is not surprising considering the low Reynolds number, small aspect ratio and the thickness of the aerofoil. Similar effects were observed for scaled versions of the LS(1) 0421 mod shape in the truncation tests [3].

7.6 SSS-127 Aerofoil

Figure 28 for the SSS-127 aerofoil shows very close correspondence between the two Reynolds number tested pre-stall. the fluctuations in lift value for the higher Reynolds number post-stall is probably an artifice due to the three-dimensional nature of the flow. The wake drag measurements are very similar to those of the 25% and 23% aerofoils although the maximum lift of 1.2 is slightly lower.

7.7 SSS-129 Aerofoil

The performance of the SSS-129 aerofoil is shown in figure 29. For the two Reynolds number tested very similar Cl and Cd results were obtained although the overall performance is low with a maximum Cl of 0.9. The balance drag measurement was significantly larger than the wake drag indicating the three–dimensional effects were important even at low angles of attack.

8. DISCUSSION

8.1 New Series Aerofoil Performance

A comparison of the relative overall performance of the five new aerofoils for Cl and Cd for the Reynolds numbers tested of 2000000 and 1600000 are shown in figures 30 through 33. From these it is clear that, as expected, performance degrades with increasing thickness-chord ratio. However, it is noticeable for Cl that up to 25%(t/c) this degradation is small.

The two-dimensional drag coefficient for low Cl for all the aerofoils is very similar at around 0.01. The thickness-chord ratio determining the Cl at which the drag rise commences.

It should be noted that the change in dCl/da slope which occurs at a=6 for all the aerofoils is a feature of the flow conditions in the test rig deviating from a true two-dimensional flow and as a result the data obtained for higher angles of attack is lower than that which would be obtained in a freestream.

The suitability of the Southampton Ship Science Series 1 aerofoils for use in horizontal wind axis turbine blades is demonstrated by the results obtained in this experimental investigation. Significant performance degradation did not occur with increasing thickness. The loss in performance compared to thinner sectioned aerofoils is more than compensated by the reduction in wind turbine blade weight. the drag coefficient of even the 29% (t/c) aerofoil was only 25% greater than the LS(1) 0421 at low angles of attack. the new series aerofoils also have a lower Cm1/4c coefficient which reduces the blade root stresses further. The sensitivity of thick aerofoils to leading edge roughness is a problem which occurs for most low-Reynolds number thick sectioned aerofoils and further research is needed in this area.

8.2 Proposed modifications to the Two-Dimensional Aerofoil Test Rig

A few minor modifications are suggested prior to future use of the two-dimensional test rig. The installation of the aerofoil and the setting to the desired angle of incidence is an elaborate and time consuming procedure. A redesign of the yoke arrangement between the balance and the aerofoil could save a significant amount of wind tunnel as would the automation of the incidence setting. This would allow an incidence traverse to be completed without the need to start and stop the wind tunnel fan.

The cover plates for the balance on the outside of the central splitter are cumbersome to remove and fit. The use of a hinge would be a better method.

REFERENCES

[1] Satchwell,C.J.,"Research, design and development of thick aerofoils for wind turbines", Proc. European Wind energy Conference 1988.

[2] Williams,B.R.,"The calculation of flow about aerofoils at low-Reynolds numbers with application to remotely piloted vehicles", Proc. of conference on Aerodynamics at low Reynolds numbers, Royal Aeronautical Society, 1986.

[3] Reddy,M.T.,"Wind tunnel tests on thickened and truncated Aerofoils for wind turbines", Southampton University Ship Science Report, December 1987.

[4] McGhee,R.J., & Beasley,W.D.,"Wind tunnel tests for an improved 21-percent-thick low-speed Airfoil section", NASA-TM-78650, 1978.

[5] Mueller,T.J.,"Low Reynolds Number Wind Tunnel Measurements: The Importance of Being Earnest",Proc. of conference on Aerodynamics at low Reynolds numbers, Royal Aeronautical Society, 1986.

[6] Melvill Jones,B. ,"The measurement of profile drag by the Pitot traverse method", A.R.C. R&M 1688, 1936.

[7] Rae, W.H., & Pope,A. ,"Low-speed Wind Tunnel testing", 2nd Ed., John Wiley & sons ,1984.

[8] Alderson, D., Cole, D., Hewavisanti,R.,Kelsall,S.,"The design and construction of a 3-axis model cutting machine", Southampton University Four year course in Engineering Design Project, 1988.

[9] Digiplan User Information: SR12,SR22,&SR32 Rack Systems Fitted with RS232c Interface, Digiplan Ltd.

[10] Cardases, M.H., Hyde, K.M., & Makinson, I.D., "Horizontal Planar Motion Mechanism", Southampton University Four year course in Engineering Design Project, 1987.

[11] Minate Solartron 7061 Digital voltmeter Technical Manual: Part3, Schlumberger Electronics.

[12] Abbott, I.H., & Von Doenhoff, A.E., "Theory of Wing Sections", Dover Publications, 1958.

[13] Bryer, D., & Pankhurst, R.C., " Pressure -probe Methods for determining wind speed and flow direction", H.M.S.O. 1971

[14] Logan, C., "Nimbus PC: The GPIB interface", Research Machines Ltd, PN15049, 1986.

[15] Glauert, H., "The elements of aerofoil and airscrew theory", Cambridge University Press, 1968.

[16,17] Duncan,W.J., Thom, A.S., & Young,A.D., "Mechanics of Fluids", Edward Arnold Ltd., 1970.

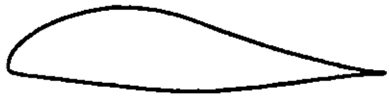
ACKNOWLEDGEMENTS

This work has been carried out as part of CEC contract EN3 W-0016-UK.

Technical support and manufacture was provided by the Universities joint ISVR/Ship Science/Aeronautics workshop staff. A special note of thanks to D. Edwards for his help with the design and construction of the splitters, D. Goldsworthy for the production of the six test aerofoils and T. Edgeley for his advice and help on the Electronics.

Appendix A

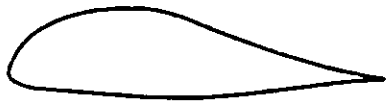
Aerofoil Coordinates



SSS-121

Number of data points : 45

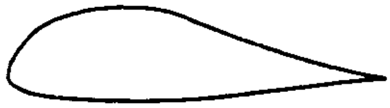
x/c	Yu/c	Yl/c
0.00000	0.00000	0.00000
0.00200	0.01685	-0.00532
0.00500	0.02666	-0.00929
0.01250	0.04057	-0.01278
0.02500	0.05649	-0.01542
0.03750	0.06942	-0.01765
0.05000	0.08002	-0.01958
0.07500	0.09727	-0.02251
0.10000	0.11094	-0.02508
0.12500	0.12300	-0.02767
0.15000	0.13269	-0.03033
0.17500	0.14058	-0.03304
0.20000	0.14699	-0.03578
0.22500	0.15208	-0.03850
0.25000	0.15595	-0.04119
0.27500	0.15869	-0.04379
0.30000	0.16046	-0.04628
0.32500	0.16104	-0.04861
0.35000	0.15982	-0.05074
0.37500	0.15758	-0.05266
0.40000	0.15432	-0.05431
0.42500	0.14956	-0.05563
0.45000	0.14344	-0.05658
0.47500	0.13571	-0.05711
0.50000	0.12685	-0.05699
0.52500	0.11772	-0.05597
0.55000	0.10879	-0.05431
0.57500	0.10027	-0.05225
0.60000	0.09199	-0.04985
0.62500	0.08395	-0.04715
0.65000	0.07615	-0.04421
0.67500	0.06858	-0.04110
0.70000	0.06126	-0.03782
0.72500	0.05419	-0.03441
0.75000	0.04738	-0.03090
0.77500	0.04085	-0.02734
0.80000	0.03461	-0.02375
0.82500	0.02868	-0.02015
0.85000	0.02308	-0.01658
0.87500	0.01782	-0.01308
0.90000	0.01292	-0.00968
0.92500	0.00840	-0.00654
0.95000	0.00425	-0.00478
0.97500	0.00108	-0.00332
1.00000	0.00000	-0.00120



SSS-123

Number of data points : 45

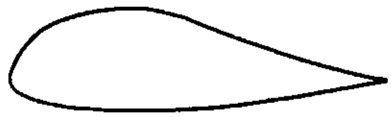
x/c	Yu/c	Yl/c
0.00000	0.00000	0.00000
0.00200	0.01459	-0.00653
0.00500	0.02702	-0.01268
0.01250	0.04226	-0.01894
0.02500	0.06282	-0.02569
0.03750	0.07835	-0.03064
0.05000	0.09187	-0.03468
0.07500	0.11271	-0.03856
0.10000	0.12759	-0.04011
0.12500	0.13937	-0.04183
0.15000	0.14883	-0.04371
0.17500	0.15647	-0.04560
0.20000	0.16258	-0.04744
0.22500	0.16736	-0.04919
0.25000	0.17090	-0.05083
0.27500	0.17330	-0.05238
0.30000	0.17469	-0.05379
0.32500	0.17494	-0.05504
0.35000	0.17377	-0.05611
0.37500	0.17153	-0.05701
0.40000	0.16820	-0.05769
0.42500	0.16328	-0.05811
0.45000	0.15652	-0.05826
0.47500	0.14780	-0.05810
0.50000	0.13805	-0.05750
0.52500	0.12841	-0.05632
0.55000	0.11908	-0.05463
0.57500	0.11011	-0.05255
0.60000	0.10138	-0.05013
0.62500	0.09289	-0.04743
0.65000	0.08462	-0.04451
0.67500	0.07658	-0.04143
0.70000	0.06878	-0.03821
0.72500	0.06123	-0.03487
0.75000	0.05394	-0.03145
0.77500	0.04692	-0.02799
0.80000	0.04019	-0.02450
0.82500	0.03377	-0.02104
0.85000	0.02766	-0.01760
0.87500	0.02190	-0.01425
0.90000	0.01651	-0.01101
0.92500	0.01149	-0.00798
0.95000	0.00695	-0.00578
0.97500	0.00315	-0.00334
1.00000	0.00012	-0.00159



SSS-125

Number of data points : 45

x/c	Yu/c	Yl/c
0.00000	0.00000	0.00000
0.00200	0.01191	-0.00789
0.00500	0.03780	-0.01889
0.01250	0.04967	-0.03078
0.02500	0.07146	-0.04068
0.03750	0.09231	-0.04866
0.05000	0.10543	-0.05062
0.07500	0.12807	-0.05546
0.10000	0.14496	-0.05591
0.12500	0.15665	-0.05664
0.15000	0.16585	-0.05767
0.17500	0.17319	-0.05864
0.20000	0.17898	-0.05948
0.22500	0.18341	-0.06015
0.25000	0.18659	-0.06067
0.27500	0.18861	-0.06104
0.30000	0.18956	-0.06126
0.32500	0.18946	-0.06133
0.35000	0.18829	-0.06124
0.37500	0.18605	-0.06100
0.40000	0.18260	-0.06060
0.42500	0.17746	-0.06003
0.45000	0.16999	-0.05929
0.47500	0.16016	-0.05836
0.50000	0.14942	-0.05719
0.52500	0.13928	-0.05577
0.55000	0.12954	-0.05401
0.57500	0.12010	-0.05185
0.60000	0.11089	-0.04936
0.62500	0.10192	-0.04661
0.65000	0.09317	-0.04367
0.67500	0.08464	-0.04058
0.70000	0.07635	-0.03736
0.72500	0.06830	-0.03406
0.75000	0.06051	-0.03070
0.77500	0.05298	-0.02730
0.80000	0.04574	-0.02390
0.82500	0.03880	-0.02053
0.85000	0.03217	-0.01720
0.87500	0.02590	-0.01397
0.90000	0.02000	-0.01087
0.92500	0.01450	-0.00795
0.95000	0.00944	-0.00523
0.97500	0.00484	-0.00281
1.00000	0.00000	0.00000

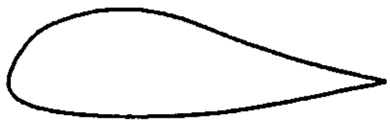


SSS-127

Number of data points : 45

x/c	Yu/c	Yl/c
0.00000	0.00000	0.00000
0.00200	0.00738	-0.00467
0.00500	0.01816	-0.01129
0.01250	0.03702	-0.01919
0.02500	0.06304	-0.02848
0.03750	0.08217	-0.03500
0.05000	0.09938	-0.04133
0.07500	0.12411	-0.05453
0.10000	0.14031	-0.06223
0.12500	0.15230	-0.06730
0.15000	0.16182	-0.07115
0.17500	0.16947	-0.07414
0.20000	0.17560	-0.07652
0.22500	0.18036	-0.07839
0.25000	0.18387	-0.07985
0.27500	0.18623	-0.08095
0.30000	0.18753	-0.08173
0.32500	0.18777	-0.08223
0.35000	0.18697	-0.08244
0.37500	0.18510	-0.08240
0.40000	0.18202	-0.08209
0.42500	0.17737	-0.08154
0.45000	0.17045	-0.08072
0.47500	0.16110	-0.07963
0.50000	0.15075	-0.07825
0.52500	0.14058	-0.07654
0.55000	0.13100	-0.07441
0.57500	0.12173	-0.07180
0.60000	0.11269	-0.06877
0.62500	0.10387	-0.06538
0.65000	0.09528	-0.06174
0.67500	0.08690	-0.05789
0.70000	0.07875	-0.05385
0.72500	0.07083	-0.04965
0.75000	0.06315	-0.04532
0.77500	0.05573	-0.04091
0.80000	0.04857	-0.03644
0.82500	0.04168	-0.03193
0.85000	0.03510	-0.02743
0.87500	0.02882	-0.02296
0.90000	0.02288	-0.01856
0.92500	0.01729	-0.01428
0.95000	0.01206	-0.01013
0.97500	0.00716	-0.00613
1.00000	0.00224	-0.00197

****



SSS-129

****

Number of data points : 45

****

x/c	Yu/c	Yl/c
0.00000	0.00000	0.00000
0.00200	0.01335	-0.00751
0.00500	0.02639	-0.01525
0.01250	0.04553	-0.02470
0.02500	0.07148	-0.03588
0.03750	0.09166	-0.04434
0.05000	0.10933	-0.05250
0.07500	0.13475	-0.06298
0.10000	0.15144	-0.06943
0.12500	0.16360	-0.07427
0.15000	0.17325	-0.07809
0.17500	0.18101	-0.08118
0.20000	0.18721	-0.08368
0.22500	0.19204	-0.08571
0.25000	0.19563	-0.08733
0.27500	0.19807	-0.08860
0.30000	0.19944	-0.08956
0.32500	0.19977	-0.09021
0.35000	0.19907	-0.09058
0.37500	0.19734	-0.09068
0.40000	0.19439	-0.09051
0.42500	0.19018	-0.09008
0.45000	0.18387	-0.08938
0.47500	0.17529	-0.08839
0.50000	0.16485	-0.08711
0.52500	0.15414	-0.08552
0.55000	0.14418	-0.08352
0.57500	0.13456	-0.08103
0.60000	0.12521	-0.07806
0.62500	0.11606	-0.07468
0.65000	0.10712	-0.07101
0.67500	0.09839	-0.06706
0.70000	0.08986	-0.06288
0.72500	0.08156	-0.05852
0.75000	0.07349	-0.05401
0.77500	0.06564	-0.04935
0.80000	0.05804	-0.04460
0.82500	0.05070	-0.03977
0.85000	0.04362	-0.03491
0.87500	0.03682	-0.03002
0.90000	0.03032	-0.02516
0.92500	0.02412	-0.02036
0.95000	0.01825	-0.01564
0.97500	0.01267	-0.01103
1.00000	0.00730	-0.00645

****

Appendix B

Program and segment listings.

```
    PROGRAM INITIAL;

{  Sets up and allows various components to be individually tested prior
to use. Allows creation of test data files

Written by : S. Turnock
Date: 1/8/88
}
CONST
         pi=3.141595428;

         maxsize = 100;
      doubmaxsize = 200;
         {   MSDOS filenames }
         rawout = 'RAW';
         refout = 'REF';
{===========================================================================}
{        type declarations}
TYPE
         TfName    = String[80];        {1401 type: used to link segment Proload}
      DWORD        = 0..maxint;                  {graphics output type}
         INT       = -32768..32767;     {Graphics output type}
         INTO      = 0..65535;                   {Graphics output type}
         BYTE      = 0..255;             {Graphics Output type}
         INT1      = -128..127;                  {1401 : 8 bit ADC/DAC data}
         INT2      = -32768..32767;     {1401 : 16 bit ADC/DAC data}
         WORD      = 0..65535;                   {1401 : PSTH family array type}
      RECT   = RECORD
                         xl,yl,xr,yr : INT;
                  END;

         BAL =  Array[1..6] of real;
         Scan = Array[1..4] of real;

{===========================================================================}
{        common declarations}
COMMON cin,cout:TEXT;        {1401 files. Must be common for Proload}

{===========================================================================}
{        variable declarations}
VAR
         { Individual Test Data}
         testno:   integer;
         time:                 string[6];
         date: string[8];
         aerofoil,description: string;
         thickness,velocity,Mach_No,Reynolds_No: real;
         incidence,airtemp,density,chord,span:   real;
         {         i/o file names}
         datainput,dataoutput,file1,file2:text;
         rawoutput:text;
      Choice : Integer;
       { wake traverse data}
         Ystep, Supply: real;
         Y : Array[1..maxsize] of REAL;
         height : real;
         NoElements,i : Integer;
       Reply, Finito, Try : boolean;
       DOSfilename : String;
         {    }
         Pitot1,Pitot2,Pitot3,Static : ARRAY[1..maxsize] OF Word;
         BV,SV : ARRAY[1..maxsize,1..6] of Word;
```

```
        SDBal,Balance : Bal;
        SDScan ,Scani : Scan;

{       external procedure / function declarations}



        {Dispseg Externals   written by K.M.Hyde}

PROCEDURE AnyKeyToCont;                                          EXTERNAL;
PROCEDURE PrintReal(Text:String;X:Real;F,D:Integer);      EXTERNAL;
FUNCTION InputInt(Text:String;Min,Max:Integer): Integer;EXTERNAL;
FUNCTION InputReal(Text:String;Min,Max:Real;f,d:integer) :Real;External;
FUNCTION Question(Text:String) : Boolean;                        EXTERNAL;

        { MinSeg Externals }

PROCEDURE BALANCEd(strain,calibrate : BAL; SuppVolt : Real;
                                             VAR balcoeff : BAL);EXTERNAL;
PROCEDURE InitMinate;                                                      EXTERNAL
;
PROCEDURE Minate( VAR  balance : BAL;scani : SCAN; suppvolt : real);EXTERNAL;
PROCEDURE Calibrated(VAR MeanBal,SDBal : BAL; MeanScan,SDScan : SCAN);EXTERNAL;
PROCEDURE DirContMin;                                                      EXTERNAL
;
PROCEDURE MinatePrint(Balance : Bal; Scani : SCAN);       EXTERNAL;

{                                            TravSeg Externals }

PROCEDURE ErrCheck( VAR height : REAL);                                    EXTERNAL
;
PROCEDURE finished(VAR height : REAL);                                     EXTERNAL
;
PROCEDURE InitTraverse;                                                    EXTERNAL
;
PROCEDURE DirCont;                                                         EXTERNAL
;
PROCEDURE ChangeParameter(VAR SS,Accrate,num,speed : Integer);EXTERNAL;
PROCEDURE MoveTraverse(ystep: real;ss,accrate,num,speed : Integer;
                                                   VAR height:real);EXTERNAL;


{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE WriteData;
{       Writes out testrun description }
BEGIN
        assign(dataoutput,'TEST.DAT');
        rewrite(dataoutput);
        writeln(dataoutput,testno);
        writeln(dataoutput,date);
        writeln(dataoutput,time);
        writeln(dataoutput,aerofoil);
        writeln(dataoutput,velocity:8:3,thickness:8:3,density:8:3);
        writeln(dataoutput,chord:8:3,span:8:3,' chord   span');
        writeln(dataoutput,incidence:8:3,airtemp:8:3,' incidence  airtemp');
        writeln(dataoutput,description,' Airfoil description');
        writeln(dataoutput);
        close(dataoutput);
END;

{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}


        PROCEDURE ReadData;
{       Reads in testrun description }
BEGIN
```

```
        assign(datainput,'TEST.DAT');
        reset(datainput);
        readln(datainput,testno);
        readln(datainput,date);
        readln(datainput,time);
        readln(datainput,aerofoil);
        readln(datainput,velocity,thickness,density);
    readln(datainput,chord,span);
        readln(datainput,incidence,airtemp);
        readln(datainput,description);
        close(datainput);
END;

{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        creates output file MSDOS name}
PROCEDURE SetUpFile(name : String);
VAR
        NoText : String;
Begin
        Str(TestNo,NoText);
        DOSfilename := concat(name,NoText,'.DATA');
        assign(dataoutput,DOSfilename);
      rewrite(dataoutput);
End;

{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        Allows experiment data to be altered if necessary}
PROCEDURE ChangeData;
VAR
        Test : Boolean;
        Enter : Integer;

BEGIN
        test := true;
        repeat;
        page;
        writeln('Test Data is as follows :-');
        writeln('0          Enter number of data ( 0 for no more changes)');
        writeln;
        writeln('1          Date : ',date);
        writeln;
        writeln('2          Time : ',time);
        writeln;
        writeln('3          Test Number : ',TestNo:4);
        writeln;
        printreal('4                    Velocity : ',velocity,5,2);
        writeln;
        printreal('5                    Density : ',density,5,2);
        writeln;
        printreal('6                    Air Temperature : ',airtemp,5,2);
        writeln;
        printreal('7                 Chord : ',chord,6,2);
        writeln;
        printreal('8                    Span : ',span,6,2);
        writeln;
        printReal('9                    Thickness/chord : ',thickness,6,3);
        writeln;
        PrintReal('10                   Angle of Attack : ',Incidence,5,2);
        writeln;
        writeln('11                     Aerofoil : ',aerofoil);
        writeln;
        writeln('12                     Experiment :    ',description);
      writeln;
        write(' Enter data item number to be changed (1 .. 10)');readln(Enter);
        CASE Enter Of
      0: test := false;
```

```
        1: begin
                              write('Revised Date (day/month/year)');
                               readln(date);
                         end;
        2: begin
                              write('Revised Time (hour:min) :-');
                               readln(time);
                         end;
        3: begin
                              TestNo :=InputInt('New Test Number',0,3000000);
                         end;
        4: begin
                              Velocity :=InputReal(' Revised freestream velocity',0.0,65.0,5,1);
                         end;
        5: begin
                              Density :=InputReal('New Air Density ',0.0,1.5,4,1);
                         end;
        6: begin
                              AirTemp :=InputReal('Air Temperature',0.0,25.0,4,1);
                         end;
        7: begin
                              Chord :=InputReal('Chord (mm)',0.0,1000.0,5,1);
                         end;
        8: begin
                              Span :=InputReal('Span (mm)',0.0,995.0,5,1);
                         end;
        9: begin
                              Thickness := InputReal('thickness/chord',0.0,1.0,5,3);
                         end;
       10: begin
                              Incidence := InputReal(' Angle of Attack',-90,90,5,2);
                          end;
        11: begin
                              write('Revised aerofoil code :-');
                              readln(aerofoil);
                         end;
        12: begin
                              write('Revised test description :-');
                              readln(description);
                          end;
        END;
        until test = false;

END;

{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        displays experiment data file}
PROCEDURE DisplayData;
BEGIN
        page;
        writeln;
        writeln;
        writeln('            AEROFOIL EXPERIMENT TEST DATA');
        writeln;
        writeln;
        writeln('            Date:  ',date);
        writeln('            Time:  ',time);
        writeln;
        writeln('           Test Number            :=  ',TestNo:8);
        writeln;
        PrintReal('            Velocity :     ',velocity,5,2);
        writeln;
        PrintReal('            Density  :     ',density,5,2);
        writeln;
        PrintReal('    Air Temperature :     ',airtemp,5,2);
        writeln;
```

```
        PrintReal('   Angle of Attack :    ',Incidence,5,2);
        writeln;
        PrintReal('   Thickness/Chord :    ',Thickness,5,3);
        writeln;
        PrintReal('              Chord  :    ',chord,6,2);
        writeln;
        PrintReal('               Span  :    ',span,6,2);
        writeln;
        writeln('    Aerofoil      :  ',aerofoil);
        writeln;
        writeln('   Experiment :    ',description);
END;


{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        Allows experiment data to be entered}
PROCEDURE InputData;
BEGIN
        page;
        writeln('Enter Test Data  :-');
        writeln;
        write('Todays  Date (day/month/year)');
         readln(date);
        write('Test Time (hour: min) :-');
         readln(time);
        TestNo :=InputInt(' Test Number',0,3000000);
        writeln;
        Velocity :=InputReal(' freestream velocity (m/s)',0.0,65.0,5,1);
        writeln;
        Density :=InputReal(' Density (Kg/m3) ',0.0,1.5,4,1);
        writeln;
        Incidence :=InputReal(' Angle of Attack (degrees)',-90.0,90.0,5,1);
        writeln;
        Thickness :=InputReal(' Thickness/Chord Ratio',0.0,1.0,5,3);
        writeln;
        AirTemp :=InputReal('Air Temperature (degrees Celsius)',0.0,25.0,4,1);
        writeln;
        Chord :=InputReal('Chord (mm)',0.0,1000.0,4,1);
        writeln;
        Span :=InputReal('Span (mm)',0.0,995.0,4,1);
        writeln;
        write('Aerofoil code :-');
        readln(aerofoil);
        write('Test description :-');
        readln(description);
END;

{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         }

{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE Data;
BEGIN
        InputData;
        DisplayData;
        AnyKeyToCont;
        ChangeData;
        WriteData;
        ReadData;
        DisplayData;
        AnyKeyToCont;
        page;
END;

{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE WakeControl;
```

```
VAR
        ent,rep,pr : boolean;
        num,accrate,ss,maxspeed,speed : Integer;
BEGIN
        repeat
                rep := Question(' Is Wake traverse powered up and ready');
        until rep = true;
        InitTraverse;
        num     :=      5;
        accrate := 4;
        ss := 4;
        maxspeed := 1;
        ChangeParameter(ss,accrate,num,speed);
        repeat
                writeln;
          writeln('  Manual Control of Wake Traverse ');
          writeln;
                writeln;
                dircont;
          pr := Question(' Direct Control of the Wake Traverse ');
          If pr = false then begin
                 Ystep := InputReal(' Enter step size in millimetres',-100,100,5,1);
                 writeln;
           MoveTraverse(Ystep,ss,accrate,num,speed,height);
                 Writeln('  Move Complete');
                 end;
                If pr = true then dircont;
                ent := Question(' Do You Wish To Finish');
     Until ent = true;
end;

{*************************************************************************
**************************************************************************

                              MAIN PROGRAM

**************************************************************************
*************************************************************************}
BEGIN
        try := false;
        repeat
        page;
        writeln;
        writeln('       INITIALISATION        MENU');
        writeln;
        writeln;
        writeln('   Enter Number for required option');
        writeln;
        writeln;
        writeln('                      Aerofoil Test Data Input    (1)');
        writeln;
        writeln('                      Wake Traverse Operation      (2)');
        writeln;
        writeln('                      Minate Operation            (3)');
        writeln;
        writeln('                      To Quit Program              (4)');
        writeln;
        Choice := InputInt('   Enter Choice ?',1,4);
        CASE Choice of
           1: begin
                      Data;
                  end;
              2: begin
                         WakeControl;
                     close(file1);
                     close(file2);
```

```
                         end;
                    3: begin
                              reply := Question(' Is Minate powered up ');
                              If reply = true then begin
                                             For i := 1 to 6 do begin
                                                       {calibrate[i] := 0.0;}
                                                       SDBal[i] := 0.0;
                                                       Balance[i] := 0.0;
                                                       if i<5 then begin
      {                                                            ScanCal[i] := 0.0;}
                                                                   Scani[i]:=0.0;
                                                                   SDScan[i] :=0.0;
      {                                                            Pressures[i] := 1.23456;
                                                                   NonDim[i] := 1.023;}
                                                                   end;
                                                       end;
                                   Supply := 0.0;
                                             InitMinate;
                                     Minate(Balance,Scani,Supply);
                                             MinatePrint(Balance,Scani);
                                             Writeln('   Supply voltage is :- ',Supply);
                                             AnyKeyToCont;
                                             Calibrated(balance,SDbal,Scani,SDScan);
                                             Writeln('    Average Value is ');
                                             MinatePrint(Balance,scani);
                                             Writeln('   Standard deviation');
                                             MinatePrinT(SDbal,SDScan);
                                             AnyKeyToCont;
                                             end;
                         end;
                    4: begin
                              try := true;
               end;
      end;
         until try = true;
END.
```

```
   PROGRAM EXPERIMENT;

{-----------------------------------------------------------------------
 AEROFOIL TWO-DIMENSIONAL DATA AQUISITION PROGRAM
 -----------------------------------------------------------------------
 NOTES:
         This program drives a Wake Traverse via an RS232C link connected
 to a Digiplan IF1 Stepper Motor Controller. Four channels of pressure
 data is acquired along with six strain gauge bridge voltages ( 12
 channels in all) using a CED1401 analogue to digital converter.
         The program is part of a suite of programs used to obtain Lift,
 Drag and Moment data for a series of two dimensional aerofoils.
       It is written in Prospero Pascal and based on the RM Nimbus
 network of the Southampton University Ship Science Department.



 Written By: S.Turnock
 Date:       5/7/88,26/7/88,29/7/88,1/8/88,2/8/88, and September '88
 -----------------------------------------------------------------------}

{        constant declarations}
CONST
         pi=3.141595428;

         maxsize = 100;
       doubmaxsize = 200;
         {  MSDOS filenames }
         rawout = 'RAW';
         refout = 'REF';
{======================================================================}
{        type declarations}
TYPE
         TfName    = String[80];        {1401 type: used to link segment Proload}
       DWORD       = 0..maxint;                  {graphics output type}
         INT       = -32768..32767;     {Graphics output type}
         INTO      = 0..65535;                   {Graphics output type}
         BYTE      = 0..255;             {Graphics Output type}
         INT1      = -128..127;                  {1401 : 8 bit ADC/DAC data}
         INT2      = -32768..32767;     {1401 : 16 bit ADC/DAC data}
         WORD      = 0..65535;                   {1401 : PSTH family array type}
       RECT    = RECORD
                             xl,yl,xr,yr : INT;
                       END;
         DATAARRAY = ARRAY[1..maxsize] of INT;
         PLOTDATA = ARRAY[1..maxsize] of REAL;
         AbsData = ARRAY[1..doubmaxsize] of INT;
         Coef = ARRAY[1..6] of STRING;

         BAL =  Array[1..6] of real;
         Scan = Array[1..4] of real;
         Spd = Array[1..3] of real;

{======================================================================}
{        common declarations}
COMMON cin,cout:TEXT;        { Must be common for SEGMENTS}

{======================================================================}
{        variable declarations}
VAR
         { Individual Test Data}
         testno:    integer;
```

```
        time:                   string;
        aerofoil,date,description:      string;
        thickness,velocity,Mach_No,Reynolds_No: real;
        statpress,pressure,incidence,airtemp,density,chord,span:     real;
    dynpress,splitpress,atmpress : real;
        {        i/o file names}
        datainput,file1,file2:text;
        raw,proc,final:text;
     { wake traverse data}
        Ystep, RoughCd: real;
        Y, Vel, Static : PlotData;
        ss,accrate,num,fast : Integer;
        NoElements,i : Integer;
     Bloop,Reply, Finito : boolean;
     DOSfilename : String;
         Calibrate,Balance,SDBAL,AeroCoeff : Bal;
        ScanCal,Scani,SDscan,Pressures,NonDim : Scan;
        speed : Spd;
        SuppVolt, height ,Stat: Real;

        { *************  graphics variables ********************}
        Marker, Mark, DataSize, Sze : INT;
        Iorig,Imax,Imin : INT;
        Jorig,Jmax,Jmin : INT;
        IJdata,Old1,Old2,Old3,P1,P2,P3 : AbsData;
        X,Xmax,Xmin,Ymax,Ymin,Xorig,Yorig : REAL;
        Xscale,Yscale : REAL;
        TickSpacing : INT;
        Response : CHAR;
        Title, Xaxis, Yaxis, NumText : STRING;
        press : text;
    CdCmCl : Coef;
        Count : Integer;
{==========================================================================}
{       external procedure / function declarations}

        {Dispseg Externals   written by K.M.Hyde}

PROCEDURE AnyKeyToCont;                                         EXTERNAL;
PROCEDURE PrintReal(Text:String;X:Real;F,D:Integer);      EXTERNAL;
FUNCTION InputInt(Text:String;Min,Max:Integer): Integer;EXTERNAL;
FUNCTION InputReal(Text:String;Min,Max:Real;f,d:integer) :Real;External;
FUNCTION Question(Text:String) : Boolean;                       EXTERNAL;

{==========================================================================}
{                       graphseg externals                                }

PROCEDURE SetCharHeight    (i : INT);                   EXTERNAL;
PROCEDURE SetCharWidth     (i : INT);                   EXTERNAL;
PROCEDURE StartGraph;                                                          EXTERNAL
;
PROCEDURE TidyUP;                                                              EXTERNAL
;
PROCEDURE DrawGraph(Imax,Imin,Iorig,Jmax,Jmin,Jorig : INT;
                                          Title,Xaxis,Yaxis : String
                                          ;Xmax,Xmin,Ymax,Ymin : REAL ); EXTERNAL;
PROCEDURE Convert(Xmin,Ymin:REAL; Imin,Jmin,Datasize :
                          INT;Xdata,Ydata :PlotData;Xscale,Yscale :
                          REAL; VAR IJdata: AbsData);        EXTERNAL;
PROCEDURE  Plot(IJdata : AbsData; sze,mark : INT); EXTERNAL;
PROCEDURE RemovePlot(IJdata : AbsData; sze,mark : INT); EXTERNAL;
PROCEDURE PlotText              (x,y : INT; s: STRING);        EXTERNAL;
PROCEDURE DrawBox(Imax,Imin,Jmax,Jmin : INT);    EXTERNAL;

{                                         MinSeg Externals   }
```

```
PROCEDURE BALANCED(strain,calibrate : BAL; SuppVolt : Real;
                                                       VAR balcoeff : BAL);EXTERNAL;
PROCEDURE InitMinate;                                                          EXTERNAL
;
PROCEDURE Minate( VAR  balance : BAL;VAR scani : SCAN; VAR suppvolt : real);EXTERNAL;
PROCEDURE AvgMin( VAR  balance : BAL;VAR scani : SCAN; VAR suppvolt : real);EXTERNAL;
PROCEDURE Calibrated(VAR MeanBal,SDBal : BAL;VAR  MeanScan,SDScan : SCAN);EXTERNAL;


{                                                     TravSeg Externals }
PROCEDURE ErrCheck( VAR height : REAL);                                        EXTERNAL
;
PROCEDURE finished(VAR height : REAL);                                         EXTERNAL
;
PROCEDURE InitTraverse;                                                        EXTERNAL
;
PROCEDURE DirCont;                                                             EXTERNAL
;
PROCEDURE ChangeParameter(VAR SS,Accrate,num,speed : Integer);EXTERNAL;
PROCEDURE MoveTraverse(ystep: real;ss,accrate,num,fast : Integer;
                                                        VAR height : real);EXTERNAL;
PROCEDURE SetToLimit(VAR Height : REAL);                                   EXTERNAL;

{==========================================================================}
{         internal procedure / function declarations}


{--------------------------------------------------------------------------}
          PROCEDURE ReadData;
{         Reads in testrun description }
BEGIN
          assign(datainput,'TEST.DAT');
          reset(datainput);
          readln(datainput,testno);
          readln(datainput,date);
          readln(datainput,time);
          readln(datainput,aerofoil);
          readln(datainput,velocity,thickness,density);
      readln(datainput,chord,span);
          readln(datainput,incidence,airtemp);
          readln(datainput,description);
          close(datainput);
END;

{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
          creates output file MSDOS name}
PROCEDURE SetUpFile(name : String; VAR DOSfilename : String);
VAR
          NoText : String;
Begin
          Str(TestNo,NoText);
          DOSfilename := concat('a:',name,NoText,'.DATA');
End;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
          Allows experiment data to be altered if necessary}
PROCEDURE ChangeData;
VAR
          Test : Boolean;
          Enter : Integer;

BEGIN
          test := true;
          repeat;
          page;
          writeln('Test Data is as follows :-');
          writeln('0            Enter number of data ( 0 for no more changes)');
```

```
   writeln('0              Enter number of data ( 0 for no more changes)');
   writeln;
   writeln('1              Date : ',date);
   writeln;
   writeln('2              Time : ',time);
   writeln;
   writeln('3              Test Number : ',TestNo:4);
   writeln;
   printreal('4                      Velocity : ',velocity,5,2);
   writeln;
   printreal('5                      Density : ',density,5,2);
   writeln;
   printreal('6                      Air Temperature : ',airtemp,5,2);
   writeln;
   printreal('7                   Chord : ',chord,6,2);
   writeln;
   printreal('8                      Span : ',span,6,2);
   writeln;
   printReal('9                      Thickness/chord : ',thickness,6,3);
   writeln;
   PrintReal('10                     Angle of Attack : ',Incidence,5,2);
   writeln;
   writeln('11                     Aerofoil : ',aerofoil);
   writeln;
   writeln('12                     Experiment :    ',description);
writeln;
   write(' Enter data item number to be changed (1 .. 10)');readln(Enter);
   CASE Enter Of
0: test :=  false;
    1: begin
                        write('Revised Date (day/month/year)');
                         readln(date);
                end;
    2: begin
                        write('Revised Time (hour:min) :-');
                         readln(time);
                end;
    3: begin
                        TestNo :=InputInt('New Test Number',0,3000000);
                end;
    4: begin
                        Velocity :=InputReal(' Revised freestream velocity',0.0,65.0,5,1);
                end;
    5: begin
                        Density :=InputReal('New Air Density ',0.0,1.5,4,1);
                end;
    6: begin
                        AirTemp :=InputReal('Air Temperature',0.0,25.0,4,1);
                end;
    7: begin
                        Chord :=InputReal('Chord (mm)',0.0,1000.0,5,1);
                end;
    8: begin
                        Span :=InputReal('Span (mm)',0.0,995.0,5,1);
                end;
    9: begin
                        Thickness := InputReal('thickness/chord',0.0,1.0,5,3);
                end;
10: begin
                        Incidence := InputReal(' Angle of Attack',-90,90,5,2);
                 end;
    11: begin
                        write('Revised aerofoil code :-');
                        readln(aerofoil);
                end;
    12: begin
                        write('Revised test description :-');
```

```
                        write('Revised test description : ');
                        readln(description);
                  end;
        END;
        until test = false;

END;

{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        displays experiment data file}
PROCEDURE DisplayData;
BEGIN
        page;
        writeln;
        writeln;
        writeln('          AEROFOIL EXPERIMENT TEST DATA');
        writeln;
        writeln;
        writeln('           Date:  ',date);
        writeln('           Time:  ',time);
        writeln;
        writeln('         Test Number              :=  ',TestNo:8);
        writeln;
        PrintReal('          Velocity :    ',velocity,5,2);
        writeln;
        PrintReal('          Density  :    ',density,5,2);
        writeln;
        PrintReal('   Air Temperature :    ',airtemp,5,2);
        writeln;
        PrintReal('   Angle of Attack :    ',Incidence,5,2);
        writeln;
        PrintReal('   Thickness/Chord :    ',Thickness,5,3);
        writeln;
        PrintReal('            Chord  :    ',chord,6,2);
        writeln;
        PrintReal('             Span  :    ',span,6,2);
        writeln;
        writeln('   Aerofoil     :  ',aerofoil);
        writeln;
        writeln('   Experiment :    ',description);
END;

{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE PlotReal( Message : String;X:Real;F,D:Integer ;
                                         Xcoord,Ycoord : Integer);

VAR
  IntText,DecText:String;
  i,L:Integer;
  y:Real;

 BEGIN
  i := 0;
  y := Abs(x);
  While (i<D) do begin
    y := 10.0*y;
    i := i + 1;
  End;
  i := Round(y);
  Str(i,NumText);
  L := Length(NumText);
  If (L>=F) then begin
    NumText := '';
    For i := 1 to F do NumText := Concat(NumText,'*');End
  Else begin
    While (L<=d) do begin
      Insert('0',NumText,1);
```

```
      L := L + 1;
    End;
    if (d=0) then NumText := Concat(NumText,'.')
       else Insert('.',NumText,(L-d+1));
    if (x<0) then Insert('-',NumText,1);
    L := Length(NumText);
    While (L<F) do begin
      Insert(' ',NumText,1);
      L := L + 1;
    End;
  End;
  NumText := Concat(Message,NumText);
  PlotText(Xcoord,Ycoord,NumText);
End;

{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE CalcYstep(VAR Ybef,Ylas,Ubef,Ulas : Real; VAR Ystep : Real);
CONST
        Ymin = 0.1  ;
        Ymax = 30.0;
        Percent = 0.05;
BEGIN
        If Ubef = Ulas then begin
                Ystep := Ymax;
                end;
        If Ubef <> Ulas then begin
                Ystep := abs(Percent*Ulas*(Ylas-Ybef)/(Ulas-Ubef));
                If Ystep < Ymin then Ystep:=Ymin;
        end;
        if Ystep > Ymax then Ystep := Ymax;
END;

{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE WriteBal(fillet : Integer; DataBal : Bal;verbage : String);
BEGIN
        Case fillet of
        1: begin
        Writeln(raw,'**************************************************');
        Writeln(raw,'                    BALANCE    ',verbage);
        Writeln(raw,'1     Lift         Drag          Moment          (RHS)');
        Writeln(raw,DataBal[1]:15:6,DataBal[2]:15:6,DataBal[3]:15:6);
        Writeln(raw,'2     Lift         Drag          Moment          (LHS)');
        Writeln(raw,DataBal[4]:15:6,DataBal[5]:15:6,DataBal[6]:15:6);
        Writeln(raw);
        end;
        2: begin
        Writeln(proc,'**************************************************');
        Writeln(proc,'                   BALANCE    ',verbage);
        Writeln(proc,'1    Lift         Drag          Moment          (RHS)');
        Writeln(proc,DataBal[1]:15:6,dataBal[2]:15:6,DataBal[3]:15:6);
        Writeln(proc,'2    Lift         Drag          Moment          (LHS)');
        Writeln(proc,DataBal[4]:15:6,DataBal[5]:15:6,DataBal[6]:15:6);
        Writeln(proc);
        end;
        end;
END;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE WriteScan( fillet : Integer; DataScan : Scan ;verbage : String);
BEGIN
        Case fillet of
        1: begin
        Writeln(raw,'**************************************************');
        Writeln(raw,'SCANIVALVE ',verbage);
        Writeln(raw,'        Pressure Transducer');
        Writeln(raw,'Channel    1             2             3             4');
        Writeln(raw,DataScan[1]:15:6,DataScan[2]:15:6,DataScan[3]:15:6
```

```
        Writeln(raw,DataScan[1]:15:6,DataScan[2]:15:6,DataScan[3]:15:6,
                                                    DataScan[4]:15:6);
        Writeln(raw);
        end;
        2: begin
      Writeln(proc,'************************************************');
        Writeln(proc,'SCANIVALVE ',verbage);
        Writeln(proc,'          Pressure Transducer');
        Writeln(proc,'Channel    1              2               3               4');
        Writeln(proc,DataScan[1]:15:6,DataScan[2]:15:6,DataScan[3]:15:6,
                                                    DataScan[4]:15:6);
        Writeln(proc);
        end;
        end;
END;
{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE OpenFiles(VAR MeanBal,SDBal : Bal; VAR MeanScan,SDScan : Scan);
VAR
        filename : String;
BEGIN
      SetUpFile('RAW',FileName);
        assign(raw,FileName);
        rewrite(raw);
        Writeln(raw,TestNo:4,'    -  Test Number   Raw Data');
        WriteBal(1,MeanBal,'  Average Zero Reading (voltage)');
        WriteBal(1,SDBal,'   Standard Deviation (voltage)');
        WriteScan(1,MeanScan,'  Average Zero Reading (voltage)');
        WriteScan(1,SDScan,'   Standard Deviation');

        SetUpFile('PROC',FileName);
        assign(proc,FileName);
        rewrite(proc);
        Writeln(proc,TestNo:4,'    -  Test Number    ProcData');

        SetUpFile('FINAL',FileName);
        assign(final,FileName);
        rewrite(final);
        Writeln(final,TestNo:4,'  -  Test Number    FinalData');

END;
{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE Integrate( VAR Xdat,Ydat : PlotData; Size : Integer;
                                                                        VAR Ans
wer : Real);
{     Quadratic Numeric Integration}
VAR
        a,b,c,Xlow,Xupp : real;
        Partial, Sum  : real;
        i : integer;
BEGIN
        {    carry out integration}
        If Size < 3 then answer:= 0.5 * (Ydat[1] + Ydat[2])*(Xdat[2]-Xdat[1])
        else begin
                sum := 0.0;
                For i:= 1 to Size-2 do begin
                        a:= (Ydat[i]-Ydat[i+2])/(Xdat[i]-Xdat[i+2]);
                        a:= ( a - (Ydat[i]-Ydat[i+1])/(Xdat[i]-Xdat[i+1]));
                        a:= -a/(Xdat[i+1]-Xdat[i+2]);
                        b:= (Ydat[i]-Ydat[i+2])/(Xdat[i]-Xdat[i+2]);
                        b:= b - a*(Xdat[i+2]+Xdat[i]);
                        c:= Ydat[i+2] - a * sqr(Xdat[i+2]) - b* Xdat[i+2];
                        If i = size-2 then Xupp := Xdat[i+2] else Xupp := Xdat[i+1];
                        Xlow := Xdat[i];
                        Partial := (a*Xupp*sqr(Xupp))/3.0;
                        Partial := Partial + (b * sqr(Xupp))/2.0;
                        Partial := Partial + c * Xupp;
                        Partial := Partial -(a*Xlow*sqr(Xlow))/3.0;
```

```
                                Partial := Partial - (a*Xlow*sqr(Xlow))/3.0;
                                Partial := Partial - (b * sqr(Xlow))/2.0;
                                Partial := Partial - c * Xlow;
                                Sum := Sum + Partial;
                        End;
                        Answer := Sum;
                End;
END;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE Traverse(VAR Scani,scancal : Scan ;VAR Pressure,Density :real;VAR speed : Spd;
                                        VAR Static : real;        VAR Pressures : scan);
CONST

        Chanpsi = 923868.0; {   N/m^2 per volt , provisional}

VAR
        i : integer;
BEGIN
        Pressures[1] := Abs((Scani[1]-Scancal[1])*Chanpsi);
        Pressures[2] := Abs((Scani[2]-Scancal[2])*Chanpsi);
        Pressures[3] := Abs((Scani[4]-Scancal[4])*Chanpsi);
        Pressures[4] := Abs((Scani[3]-Scancal[3])*Chanpsi);

         Speed[1] := SQRT(Pressures[1]/(0.5 * density));
         Speed[2] := SQRT(Pressures[2]/(0.5 * density));
         Speed[3] := SQRT(Pressures[3]/(0.5 * density));

        Static := Pressures[4] - Pressures[2];
END;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
}
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++
        Initial screen display complete with graphs}
PROCEDURE Display;
VAR
        con,crn,ibu,ibl,jbu,jbl : INT;
BEGIN
        StartGraph;
        {Draw velocity graph}
        Title := '  VELOCITY DEFECT';
        Xaxis := '   Velocity (m/s)';
        Yaxis := '          Height (mm)';
        DrawGraph(330,20,40,235,50,142,Title,Xaxis,Yaxis,60,20,100,-100);
        SetCharHeight(2);
        SetCharWidth(2);
        PlotText(355,55,'  YSTEP');
        SetCharHeight(1);
        SetCharWidth(1);
        DrawBox(350,639,175,235);
        DrawBox(350,639,125,170);
        DrawBox(350,493,105,120);    DrawBox(498,639,105,120);
        DrawBox(350,493,85,100);     DrawBox(498,639,85,100);
        DrawBox(350,639,50,80);
        PlotText(352,222,'  Balance 1         Balance 2');
        PlotText(352,157,'           Pressure (N/m^2)');

        write(chr(27),'[23;3;25;79~B');       { define text scrolling window}
        write(chr(27),'[30;43m');        {change to black on white}
        page;
END;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        Update screen with new graphical information}
PROCEDURE Refresh(VAR NoElements : Integer;AeroCoeff : Bal; Vel,Y : PlotData;
                                        Pressures,NonDim : Scan; YStep : real);
CONST
        Xscal = 0.13793;
        Yscal = 1.0811;
```

```
VAR
        realtext : string;
        IJdata : Absdata;
BEGIN
        PlotReal('  L =',AeroCoeff[1],10,5,352,207);
        PlotReal('  D =',AeroCoeff[2],10,5,352,192);
        PlotReal('  M =',AeroCoeff[3],10,5,352,177);
        PlotReal('  L =',AeroCoeff[4],10,5,498,207);
        PlotReal('  D =',AeroCoeff[5],10,5,498,192);
        PlotReal('  M =',AeroCoeff[6],10,5,498,177);

        PlotReal(' P1 =',Pressures[1],10,5,352,142);
        PlotReal(' P2 =',Pressures[2],10,5,352,127);
        PlotReal(' P3 =',Pressures[3],10,5,498,142);
        PlotReal(' P4 =',Pressures[4],10,5,498,127);

        PlotReal(' Cd =',NonDim[1],9,4,352,107);
        PlotReal(' Cl =',NonDim[2],9,4,498,107);
        PlotReal(' WT =',NonDim[3],9,4,352,87);
        PlotReal(' CM =',NonDim[4],9,4,498,87);

        PlotReal('= ',Ystep,10,4,498,60);

                {     Plot velocities }
        Convert(0.0,-100.0,40,50,NoElements,Vel,Y,Xscal,Yscal,IJdata);
        RemovePlot(IJdata,NoElements,3);
        Plot(IJdata,NoElements,3);

END;

{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        allows Y step to be altered manually}
PROCEDURE ChangeYstep(VAR Ystep : REAL);
Begin
        page;
        Ystep := InputReal(' Enter new step (+ or -) size in mm',
                                                -100.0,100.0,10,4);

End;
{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      Saves  data into output files on acquisition}
PROCEDURE DataSave(VAR Balance : Bal;Scani : Scan;SuppVolt : real;
                               AeroCoeff : Bal; Speed : Spd;
                               Pressures : Scan;high : real;NoElements: Integer
;
BEGIN
        writeln(raw,NoElements,'    Data Entry');
        writeln(raw,high,'   mm from Wake Traverse Datum');
        writeln(raw,SuppVolt,'   Supply Voltage to Balance');
        WriteBal(1,Balance,'   Uncorrected Balance Voltages ');
        WriteScan(1,Scani,'    Uncorrected Scani voltages ');

        Writeln(proc,NoElements,'     Data Entry ');
        Writeln(proc,high,'   mm from Wake Traverse Datum');
        WriteBal(2,AeroCoeff,'   Lift, Drag and Moment ');
        WriteScan(2,Pressures,'     Pressures (N/M^2)');
        WriteLn(proc,'        Velocities (m/s)');
        Writeln(proc,'Pitot  1              2              3');
        Writeln(proc,speed[1],' ',speed[2],' ',speed[3]);
        writeln(proc);

END;
{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        Write test data information and 'rough' Cd,Cl,Cm values to output file}
PROCEDURE WriteData(NoElements : Integer; RoughCd : real;Balance,SDBal : Bal;
                                          ScanCal,SDScan: Scan );
```

```
                                                        ScanCal,SDScan: Scan );
BEGIN
        Writeln(raw,'    Final calibration readings - Wind Off');
        WriteBal(1,Balance,'   Average reading (volts)');
        WriteBal(1,SDBal,'    Standard Deviation (volts)');
        WriteScan(1,ScanCal,'  Average reading (volts)');
        WriteScan(1,SDScan,'  Standard Deviation(volts)');
        Writeln(raw,'         End Of File ');
        close(raw);

        Writeln(proc,'         End Of File ');
        close(proc);

        Writeln(final,NoElements,'     Total Number of Data Readings');
        Writeln(final,RoughCd,'      Coarse Traverse Drag Cd');
        writeln(final,date);
        writeln(final,time);
        writeln(final,aerofoil);
        writeln(final,velocity:8:3,thickness:8:3,density:8:3);
        writeln(final,chord:8:3,span:8:3,' chord    span');
        writeln(final,incidence:8:3,airtemp:8:3,' incidence  airtemp');
        writeln(final,pressure:10:4,dynpress:10:4,' tunnel stat & dyn press');
        writeln(final,splitpress:10:4,' splitter pressure');
        writeln(final,atmpress:10:4,' atmospheric pressure');
        writeln(final,description,' Airfoil description');
        writeln(final,'      End Of File');
        close(final);

END;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE HardCopy;
BEGIN
END;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE Sort(Size : Integer; VAR Y,VEL,Static : PlotData);
VAR
        flagg : Boolean;
        Utest,Test, Ptest : Real;
        i : Integer;
BEGIN
        Repeat
                flagg := TRUE;
          For i:=1 to Size-1 do begin
                        Test:= Y[i+1];
                        Utest:= Vel[i+1];
                        Ptest := Vel[i+1];
                        If Test < Y[i] then begin
                                flagg := false;
                                Y[i+1] := Y[i];
                                Vel[i+1] := Vel[i];
                                Static[i+1] := Static[i] ;
                                Y[i] := Test;
                                Vel[i] := Utest;
                                Static[i] := Ptest;
                        end;
                end;
        Until Flagg = True;
END;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE DragLiftMoment(VAR NoElements : Integer;
                                    VAR          Vel,Y,Static : PlotData; AeroC
eff : Bal;
                                                                    VAR Non
im : Scan);
VAR
        L,D,M, G0,G,Gc,P, Partial : real;
        Defect : PlotData;
```

```
         Defect : PlotData;
         i : Integer;
BEGIN
         L := AeroCoeff[1] + AeroCoeff[4];
         NonDim[2] := L / (0.5 * density * sqr(velocity) * span *chord);
         D := AeroCoeff[2] + AeroCoeff[5];
         NonDim[1]:= D / (0.5 * density * sqr(velocity) * span * chord);
         M := AeroCoeff[3] + AeroCoeff[6];
         NonDim[4]:= M / (0.5 * density * sqr(velocity) * span * chord * chord);

         { sort data into order of ascending Y }
         Sort(NoElements,Y,Vel,Static);

         {  Calculate Cd using method described in ARC R&M 1688 }

         G0 := 0.5 * density * sqr(velocity);

         For i := 1 to NoElements do begin

           P := Static[i];
                  Gc := 0.5 * density * sqr(Vel[i]);
           G := Gc + P ;

           Defect[i] := Sqrt(Gc/G0) * ( 1.0 - sqrt(G/G0));

         End;

         D := 0.0;

      If NoElements > 1 then Integrate(Y,Defect,NoElements,D);

         NonDim[3] := 0.001*D * 2 / chord;

END;
{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE SwitchOnWind(VAR pressure,airtemp,dynpress,splitpress,
                                                                                   atmpres
density,velocity : real);
CONST
         Kcalib = 1.2477;
         PatmHg = 760.0;
         Patm = 101300.0;
         g = 9.81;
         RhoH2O = 1000;
         RhoMeths = 815;
         ManLevel = 1.11;  {radians from horizontal}

VAR
         Quest :Boolean;
         Velt,Prest : Real;
BEGIN
         page;
         writeln;
         writeln(' WIND ON PROCEDURE');
         writeln;
         repeat
                  Quest := Question(' Is Tunnel started and settled to desired speed');
         Until Quest = TRUE;
         writeln;
         dynpress := InputReal(' Enter tunnel dynamic pressure (mm Water)'
                                                                                   ,0.0,200
12,2);
         writeln;
         splitpress := Kcalib * dynpress;
         PrintReal('Splitter Pressure should be (mm water) :-',SplitPress,10,4);
         writeln;
         writeln(' Now calibrate using flap adjustment ');
```

```
writeln( 'Now calibrate using flap adjustment' );
        repeat
                Quest := Question(' Is flap adjustment complete');
        until Quest = TRUE;
        writeln;
        splitpress :=InputReal('Enter actual splitter dynamic pressure (mm H2O)',     0.0,200.
2,2);
        writeln;
        pressure := InputReal(' Enter Static Tunnel Pressure (inches of Meths)',     -25.0,25
12,2);
        AtmPress :=InputReal(' Enter atmospheric pressure (mmHg)',     0.0,800.
2,4);
        AirTemp :=InputReal(' Enter tunnel air temperature (deg C)'     ,0.0,30.
0,4);

        { convert everything into sensible SI units}

        dynpress := g * dynpress;
        splitpress := g * splitpress;
        prest := Patm * AtmPress / PatmHg;
        pressure := RhoMeths * g * sin(ManLevel) * pressure * 0.0254 + prest;

        { calculate things }

        density := pressure/((AirTemp+273.13)*287.0);     { 287 =
as Constant for Air}
        velocity := sqrt(splitpress/(0.5*density));
        velt := sqrt(dynpress/(0.5*density));
        writeln;
        PrintReal(' tunnel air density (kg/m^3)',density,10,4);
        writeln;
        PrintReal(' splitter freestream velocity (m/s)',velocity,10,4);
        writeln;
        PrintReal(' tunnel freestream velocity (m/s)',velt,10,4);
        writeln;
        AnyKeyToCont;

END;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        closes all files}
PROCEDURE CloseUp;
BEGIN
        Close(file1);
        Close(file2);
        TidyUp;
        write(chr(27),'[;;;~B');        { reset text scrolling window}
        write(chr(27),'[;m');           {reset to white on black}
        page;
End;

{***************************************************************************
****************************************************************************

                MAIN PROGRAM starts here or hereabouts.

****************************************************************************
****************************************************************************
}
BEGIN
        height :=-200.0; SS := 4; Num := 5; Accrate := 4; Fast := 400;
        RoughCd := 0.0; SuppVolt := 0.0; StatPress := 100000;
        SplitPress := 100000; DynPress:=100000; AtmPress := 100000;
```

```
Ystep := 0.0;
Display;
For i := 1 to 6 do begin
                    calibrate[i] := 0.0;
                    SDBal[i] := 0.0;
                    Balance[i] := 0.0;
                    if i<5 then begin
                    ScanCal[i] := 0.0;
                    Scani[i]:=0.0;
                    SDScan[i] :=0.0;
                    Pressures[i] := 1.23456;
                    NonDim[i] := 1.023;
                    end;
                    If i <4 then Speed[i] := 0.0;
end;

ReadData;

DisplayData;

Reply:= Question(' Do You Wish To Change Any Information');

If Reply  THEN ChangeData;

span := span * 0.001;   chord := chord * 0.001;

InitTraverse;

ChangeParameter(SS,Accrate,Num,fast);

height :=InputReal(' Enter height of traverse in mm',-400.0,400.0,12,4);

Bloop:=   Question(' Do you wish to set to datum');

If bloop = TRUE then SetToLimit(height);

InitMinate;

Calibrated(Calibrate,SDBal,ScanCal,SDScan);

OpenFiles(Calibrate,SDBal,ScanCal,SDScan);

SwitchOnWind(pressure,airtemp,dynpress,splitpress,
                                                            atmpress,
density,velocity);

Display;

{    Start Data acquistion Loop }

NoElements := 1;

Finito := FALSE;

REPEAT

          height := height + Ystep;

  Y[NoElements] := height;

MoveTraverse(Ystep,ss,accrate,num,fast,Y[NoElements]);

AvgMin(Balance,Scani,Suppvolt);

          Balanced(balance,calibrate,SuppVolt,Aerocoeff);
```

```
            Traverse(scani,scancal,pressure,density,speed,statpress,pressures);
        Static[NoElements] := StatPress;  {  static pressure };
        Vel[NoElements] := Speed[2];
        DragLiftMoment(NoElements,Vel,Y,Static,AeroCoeff,NonDim);
        CalcYstep(Y[NoElements-1],Y[NoElements],
                                        Vel[NoElements-1],Vel[NoElements],Ystep);
            DataSave(Balance,Scani,SuppVolt,AeroCoeff,
                                                            Speed,Pressures,h
ght,NoElements);
        If NoElements <> 1 then Display;
      Refresh(NoElements,AeroCoeff,Vel,Y,Pressures,NonDim,Ystep);
            Reply :=Question('Do you wish to change the Y increment ?');
            IF Reply THEN  ChangeYstep(Ystep);
            Finito := Question('Completed Data Acquisition');
            NoElements := NoElements + 1 ;
            Y[NoElements] := Y[NoElements-1];
        If NoElements = 101 then begin
                         writeln(' Maximum data size exceeded so saving data ');
                         finito := TRUE;
                         end;
       UNTIL finito=TRUE;
       RoughCd := NonDim[3];
       Calibrated(Balance,SDBal, ScanCal,SDScan );
     WriteData(NoElements-1,RoughCd,Balance,SDBal, ScanCal,SDScan);
       HardCopy;
       CloseUp;
End.
```

```
SEGMENT MINSEG;

{   This segment contains routines used for controlling the
Minate Solartron digital voltmeter and 16 channel switch. It is
part of the Experiment.pas suite of programs for use in the new
aerofoil evaluation testing in the Southampton University 11'
* 8' wind tunnel.

                    Written by:  S. Turnock
                              Date:  2/9/88}

CONST
    {gpib subbios definitions}
        t_inst = 4000h;
        d_gpib = 0;
        f_version    = 0;    f_s_active   = 1;    f_a_active   = 2;
        f_initialise= 3;    f_reset      = 4;    f_abort      = 5;
        f_s_control = 6;    f_a_control = 7;    f_s_address = 8;
        f_a_address = 9;    f_s_eos      = 10;   f_a_eos      = 11;
        f_s_tx_mode = 12;   f_a_tx_mode = 13;   f_transmit   = 14;
        f_s_rx_mode = 15;   f_a_rx_mode = 16;   f_receive    = 17;
        f_rx_info    = 18;   f_set_bus    = 19;   f_external   = 20;
        f_send_tag   = 21;   f_send_lag   = 22;   f_send_acg   = 23;
        f_send_ucg   = 24;   f_send_scg   = 25;   f_config     = 26;
        f_unconfig   = 27;   f_s_response= 28;   f_a_response= 29;
        f_ppoll      = 30;   f_s_request = 31;   f_a_request = 32;
        f_spoll      = 33;   f_trigger    = 34;   f_dev_clear = 35;
        f_eerom      = 36;   f_pseudo     = 37;   f_pass_cont = 38;
        f_s_remote   = 39;   f_a_remote   = 40;   f_lockout    = 41;
        f_lockstate = 42;   f_local      = 43;   f_s_timeout = 44;
        f_a_timeout = 45;   f_timeout    = 46;   f_read_reg   = 47;
        f_write_reg = 48;

    {gpib errors}
        gpib_e_none = 0;
        gpib_e_base = 4000h;
        gpib_e_max = 15;

    {other stuff}
        the_end_of_the_world = false;
        no_secondary = -1;
        end_of_list = -1;

            Nimbus = 0;
                    Solartron = 16;
TYPE
        BAL = Array[1..6] of real;
        Scan = Array[1..4] of real;
    ulong = 0..maxint;          {unsigned integer}
    short = -32768..32767;  {16 bit integer}
    ushort = 0..65535;          {unsigned 16 bit integer}
    parray = ARRAY [0..7] OF short; {parameter array for subbbios}
    gpib_device =     RECORD
                        gpib_address : 0..31;
                        method : (dma, int, std);
                        eos : boolean;
                        eos_char : char;
                        eoi : boolean;
                    END;
    boards = ARRAY [1..4] OF boolean; {array of fitted boards}
```

```
{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}

VAR
        exists : boards;

        {Dispseg Externals   written by K.M.Hyde}

PROCEDURE AnyKeyToCont;                                          EXTERNAL;
PROCEDURE PrintReal(Text:String;X:Real;F,D:Integer);        EXTERNAL;
FUNCTION InputInt(Text:String;Min,Max:Integer): Integer;EXTERNAL;
FUNCTION InputReal(Text:String;Min,Max:Real;f,d:integer) :Real;External;
FUNCTION Question(Text:String) : Boolean;                        EXTERNAL;


{=========================================================================}
{                                            GPIB external procedures }
FUNCTION sbd(f, d, t : short; p : ulong) : integer;
    EXTERNAL;

PROCEDURE hex(decnum : integer; VAR hexstr : string);
    EXTERNAL;

PROCEDURE check(error_message : string; error : integer);
    EXTERNAL;

FUNCTION ask_address : integer;
    EXTERNAL;

PROCEDURE ask_controller(VAR system, active : boolean);
    EXTERNAL;

PROCEDURE device_clear(device : ushort);
    EXTERNAL;

PROCEDURE error_init;
    EXTERNAL;

PROCEDURE find(VAR exists : boards);
    EXTERNAL;

PROCEDURE gpib_init;
    EXTERNAL;

PROCEDURE gpib_reset;
    EXTERNAL;

PROCEDURE set_address(my_address : ushort);
    EXTERNAL;

PROCEDURE set_controller;
    EXTERNAL;

FUNCTION spoll(device : ushort) : ushort;
    EXTERNAL;

FUNCTION srq : boolean;
    EXTERNAL;

PROCEDURE transfer_in(VAR rx : string; talker : gpib_device);
    EXTERNAL;

PROCEDURE transfer_out(tx : string; listener : gpib_device);
    EXTERNAL;

PROCEDURE trigger(device : ushort);
    EXTERNAL;
```

```
---- -------,

PROCEDURE version(VAR version_ident : string);
    EXTERNAL;

{                                                    Internal Procedures }
{+++==============================================================}
PROCEDURE WriteSolar(message: string);

CONST
      Nimbus = 0;
         Solartron = 16;
VAR
         SolarData : gpib_device ;
BEGIN
         SolarData.gpib_address := Solartron;
         SolarData.method := std;
         SolarData.eos := TRUE;
         SolarData.eos_char := CHR(10);
         SolarData.eoi := TRue;
         transfer_out(message,SolarData);
      If message<>'E' then trigger(solartron);
END;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE ReadSolar(VAR answer : string);
CONST
         Nimbus = 0;
         Solartron = 16;
VAR
         NimData : gpib_device;
BEGIN
         with NimData do
                  begin
                    gpib_address := Solartron;
                    method := std;
                    eos := TRUE;
                    eos_char := CHR(10);
                    eoi := True;
                  end;
                  Transfer_In(answer,NimData);
END;
{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE BALANCED(strain,calibrate : BAL; SuppVolt : Real;
                                                      VAR balcoeff : BAL);
CONST CalibVolt = 8.0125;
VAR
         Matrix : Array[1..3,1..6] of real;
         Grad :    Array[1..3,1..6] of real;
         i,ii,j : Integer;

BEGIN
         { strain gauge balances calibrated gradients }

         {Balance 1 (right hand side of splitters)}
                          {LIFT                          DRAG                      M
OMENT}
{LIFT uV/N}
                  GRAD[1,1]:=-9.814379; GRAD[1,2]:=-0.0926079; GRAD[1,3]:=-0.038823;
{DRAG uV/N}
                  GRAD[2,1]:=-0.046699; GRAD[2,2]:= 64.065850; GRAD[2,3]:= 0.066802;
{MOMENT uV/NM}
         GRAD[3,1]:= 0.954645; GRAD[3,2]:= 0.3467010; GRAD[3,3]:= 112.665916;

          {Balance 2 (left hand side of splitters)}
{LIFT uV/N}
                  GRAD[1,4]:=-10.03256; GRAD[1,5]:= 0.0;       GRAD[1,6]:=-0.035093;
{DRAG uV/N}
```

```
         GRAD[2,4]:=-0.020126; GRAD[2,5]:= 65.812466; GRAD[2,6]:= 0.197878;
{MOMENT uV/NM}
         GRAD[3,4]:=-0.541648; GRAD[3,5]:= 0.3706430; GRAD[3,6]:=113.113564;

{                            Matrices for correcting interactions      }
          {Balance 1 (right hand side of splitters)}
matrix[1,1]:= 0.99998;     matrix[1,2]:=-0.00183;     matrix[1,3]:=-0.05398;
matrix[2,1]:= 0.0;         matrix[2,2]:= 1.00002;     matrix[2,3]:=-0.00563;
matrix[3,1]:= 0.00031;     matrix[3,2]:=-0.00328;     matrix[3,3]:= 1.00000;

          {Balance 2 (left hand side of splitters)}
matrix[1,4]:= 1.00003;     matrix[1,5]:=-0.00506;     matrix[1,6]:= 0.0973;
matrix[2,4]:= 0.00144;     matrix[2,5]:= 1.00001;     matrix[2,6]:=-0.00527;
matrix[3,4]:= 0.00034;     matrix[3,5]:=-0.00308;     matrix[3,6]:= 1.00005;

{              begin procedure}

          For i := 1 to 6 do begin
                           strain[i] := strain[i]*CalibVolt/SuppVolt - calibrate[i];
                           strain[i] := strain[i] * 1000000;  {  convert to uV}
          end;

          For j := 1 to 6 do begin
                   balcoeff[j] := 0.00;
                   For i := 1 to 3 do begin
                     if j>3 then ii:= 3 + i else ii := i;
                     balcoeff[j] := balcoeff[j]  + strain[ii]*matrix[i,j]/Grad[i,ii];
        end;
      end;
END;
{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}

PROCEDURE StringToReal(Value : String; VAR Result :Real);
{         conversion routine from string to real }
VAR
          flag1, Expon, Minus, Mexp   : Integer;
          Num: String;
          len,i,No,j : Integer;
          divex,sz : REAL;
          Ep : Boolean;
PROCEDURE character(Num:string;i:Integer;VAR Ep:Boolean;
                                        VAR Minus,Mexp:Integer);
begin
If num ='.' then begin flag1:= 2; No := 50; end;
If num ='E' then Ep:= true;
If i<>1 then if num ='+' then Ep := TRUE;
If num ='e' then Ep := TRUE;
If num ='-' then if i<>1 then begin
                                                            Mexp := -1;
                                                            Ep := true;
                                          end;
IF i=1 then if num='-'then Minus := -1;
end;
PROCEDURE numeral(Num:string; Var No : Integer);
begin

                   If num = '1' then No :=1;
                   If num = '2' then No :=2;
                   If num ='3' then No :=3 ;
                   If num ='4' then No :=4;
                   If num = '5' then No :=5;
                   If num ='6' then No :=6;
                   If num ='7' then No :=7;
                   If num='8' then No :=8 ;
            If num ='9' then No :=9 ;
            If num='0' then No :=0 ;
```

```
end;
PROCEDURE Indice(flag1:Integer;Ep:boolean;No:Integer;
                                        VAR result:real; VAR expon : Integer);
begin
IF flag1 = 1 then if Ep = false then begin

                                       result := result*10.0 + No;

                                                          end;
If Ep = False then If flag1 = 2 then begin
                                         result := result + No * divex;
                                 divex := divex * 0.1;
                                                    end;
If Ep = True then  Expon := Expon * 10 + No;
end;
{------------------------------------------------------}
BEGIN
        flag1 := 1;
        Minus :=1;
      Mexp := 1;
        Ep := false;
        result := 0.0;
        divex := 0.1;
        len := length(value);
        Expon := 0;
        For i := 1 To len Do begin
                No := 100;
                Num := Copy(Value,i,1);
                Numeral(Num,No);
              If No = 100 then Character(Num,i,Ep,Minus,Mexp);


                If No < 50 then Indice(flag1,Ep,No,result,expon);

     end;
        result := result * minus;
        If Ep = True then
                                                begin
                                                        If Mexp = -1 then sz:=0.1 else sz:= 10.0
;
                                                        For i:=1 to Expon do result := result *
 sz;
                                                end;
END;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE DirContMin;
VAR
         fg : Boolean;
         otty : string;
BEGIN
        writesolar('R?');
        readsolar(otty);
        delete(otty,length(otty)-1,1);
        writeln(' boo ',otty);

        repeat
                writeln('   Enter Data for next move ');
          readln(otty);
          WriteSolar(otty);
          writeln(' finished    ');
                ReadSolar(otty);
          delete(otty,length(otty)-1,1);
        writeln(' hi there ',otty);

          fg := Question('  Another Move ');
     until fg = false;
```

```
END;

{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE InitMinate;
{                                                    Initialises the Minate }
CONST
        Initial = 'D2F1K1M0N1Q0R0T1Z0U0';
        Nimbus =0;
        Solartron =16;
VAR
        answer : String;
        i,j          : Integer;
BEGIN
        answer := 'rats';
        page;
        writeln;
        writeln('          INITIALISING MINATE     ');

        gpib_reset;
        set_controller;
        set_address(Nimbus);
        device_clear(Solartron);

        DirContMin;
        WriteSolar(Initial);
        writeln('Message sent ',initial);
        ReadSolar(answer);
        j := length(answer);
        delete(answer,j-1,1);
        writeln(answer);
        writeln;
        writeln(' Now test Solartron channels');

        WriteSolar('C01>0B');
        For  i := 1 to 11 do begin
                ReadSolar(Answer);
                writeln('Channel ',i,' input is ',answer);
        end;
        Anykeytocont;
END;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE Minate( VAR balance: Bal;  VAR Scani: Scàn;  VAR Suppvolt : real);
{                                  reads strings from Minate and converts to Real }
VAR
        Number : Real;
        i,j    : Integer;
        Answer : string;
BEGIN
        WriteSolar('C01>06');
        Answer := '1.0';
        For i:= 1 to 6 do begin
          ReadSolar(answer);
          j := length(answer);
                delete(answer,j-1,1);
                StringToReal(answer,number);
                if i=1 then j:=1;
                if i=2 then j:=4;
                if i=3 then j:=3;
                if i=4 then j:=2;
                if i=5 then j:=5;
                if i=6 then j:=6;
          Balance[j] := number;
        end;
        WriteSolar('C07>0A');
```

```
        WriteSolar('C07CR');
        For i:= 1 to 4 do begin
                ReadSolar(answer);
                j:= length(answer);
                delete(answer,j-1,1);
                StringToReal(answer,number);
                Scani[i] := number;
        END;
        WriteSolar('COB');
        ReadSolar(answer);
        j:=length(answer);
        delete(answer,j-1,1);
        StringToReal(answer,number);
        SuppVolt := number;
END;
{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE MinatePrint(Balance : Bal; Scani : SCAN);
BEGIN
        PrintReal('  Balance 1        LIFT : ',Balance[1],12,6);
        Writeln;
        PrintReal('   (volts)         DRAG : ',Balance[2],12,6);
        Writeln;
        PrintReal('                 MOMENT : ',Balance[3],12,6);
        Writeln;
        PrintReal('  Balance 2        LIFT : ',Balance[4],12,6);
        writeln;
        PrintReal('   (volts)         DRAG : ',Balance[5],12,6);
        Writeln;
        PrintReal('                 MOMENT : ',Balance[6],12,6);
        writeln;
        PrintReal('ScaniValve   Channel  1 : ',Scani[1],12,6);
        Writeln;
        PrintReal('  (volts)   Channel  2 : ',Scani[2],12,6);
        Writeln;
        PrintReal('             Channel  3 : ',Scani[3],12,6);
        Writeln;
        Printreal('             Channel  4 : ',Scani[4],12,6);
        writeln;
END;
{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE Calibrated(VAR MeanBal,SDBal : BAL; VAR MeanScan,SDScan : SCAN);
{                                             Calibrates windoff readings of Scani a
nd Balances}
CONST
        CalibVolt = 8.0125;
VAR
        NoofSamp,i,j          : Integer;
        SuppVolt                        : Real;
        DataBal               : Bal;
        DataScan                        : SCAN;
        flg                                       : Boolean;
        bloop : string;
        DevBal: Array[1..6,1..15] of real;
        DevScan: Array[1..4,1..15] of real;

BEGIN
        page;
        writeln;
        writeln('    CALIBRATION OF SCANIVALVE AND BALANCES');
        writeln;
        writeln(' Please ensure that there is :-');
        writeln;
        writeln('       (1)       No flow at all in test section( Wind Off)');
        writeln;
        writeln('       (2)       Aerofoil is correctly mounted and floats');
        writeln;
        writeln('       (3)       All devices are powered up');
```

```
        writeln;
        writeln('       (4)      Supply Voltage as close to 8.0125V as poss');
        writeln;
        writeln('       (5)      Pressure Tubes are connected');
        writeln;
        AnyKeyToCont;
        readln(bloop);
        page;
        repeat
                writeln;
          writeln('  calibration procedure commenced TAKE COVER! ');
                writeln;
                NoofSamp := InputInt(' Enter number of data samples ',2,15);
                page;
                writeln;
                writeln('Reading from Minate');
                For  i:= 1 to 6 do begin
                        MeanBal[i] := 0.0;
                        SDBal[i] := 0.0;
                        if i < 5 then begin
                                MeanScan[i] := 0.0;
                                SDScan[i] := 0.0;
                        end;
                end;
                For i := 1 to NoofSamp do begin
                        Minate(DataBal,DataScan,SuppVolt);
                        SuppVolt := SuppVolt * 1.0;
                        For j:= 1 to 6 do begin
                                DevBal[j,i] := databal[j] * CalibVolt/SuppVolt;
                                MeanBal[j] := DevBal[j,i]/NoofSamp + MeanBal[j];
                                If j < 5 then begin
                                        DevScan[j,i] := DataScan[j] * 1.0;
                                        MeanScan[j] := DevScan[j,i]/NoofSamp + MeanSca
n[j];
                                end;
                     end;
                        MinatePrint(DataBal,DataScan);
                End;

          For i:= 1 to NoofSamp do begin
                        For j:= 1 to 6 do begin

                                SDBal[j] := SDBal[j]
                                                                + SQRT(SQR(DevBal[j,i]-Mean
Bal[j])/NoofSamp);
                                if j<5 then begin
                                        SDScan[j]:= SDScan[j] +
                                                                SQRT(SQR(DevScan[j,i]-MeanS
can[j])/NoofSamp);
                                end;
                        end;
                end;
                page;
                writeln('   MEAN READINGS WHERE :- ');
                MinatePrint(MeanBal,MeanScan);
                writeln('   With a Standard Deviation of :-');
                MinatePrint(SDBal,SDScan);
                flg := Question('  Is this acceptable');
        UNTIL flg = TRUE;
END;
{*****************************************************************}
PROCEDURE AVGMIN(VAR MeanBal : BAL; VAR MeanScan : SCAN;VAR SuppVolt :real);
{                                                   Average   windon readings of Scani an
d Balances}
CONST
        CalibVolt = 8.0125;
```

```
VAR
        NoofSamp,i,j          : Integer;
        DataBal                : Bal;
        DataScan                           : SCAN;
        flg                                          : Boolean;
        bloop : string;
        DevBal: Array[1..6,1..15] of real;
        DevScan: Array[1..4,1..15] of real;
BEGIN
        SuppVolt := 1.0;
        NoofSamp := 6;
                For i := 1 To 6 do begin
                        MeanBal[i] := 0.0;
                        If i< 5 then MeanScan[i] := 0.0;
       End;
                For i := 1 to NoofSamp do begin
                        Minate(DataBal,DataScan,SuppVolt);
                        SuppVolt := SuppVolt * 1.0;
                        For j:= 1 to 6 do begin
                                DevBal[j,i] := databal[j] * CalibVolt/SuppVolt;
                                MeanBal[j] := DevBal[j,i]/NoofSamp + MeanBal[j];
                                If j < 5 then begin
                                        DevScan[j,i] := DataScan[j] * 1.0;
                                        MeanScan[j] := DevScan[j,i]/NoofSamp + MeanSca
n[j];
                                end;
                    end;
                End;
END;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
{                                        End of Procedure}
BEGIN
END.
```

```
    SEGMENT TravSeg;    {           Wake Traverse Segments     }
{   contains ProPascal procedures used to control a stepper motor
controller (DIGIPLAN IF1). It is designed to run in conjunction with
EXPERIMENT.PAS and INITIAL.PAS developed for the new wind turbine
aerofoil wind tunnel testing to be carried out in the Southampton
University 11' * 8' Wind Tunnel }

{          Written by:  S. Turnock
           Date:                6/9/88}


CONST
           pi=3.141595428;

           maxsize = 100;
        doubmaxsize = 200;
           {   MSDOS filenames }
           rawout = 'RAW';
           refout = 'REF';
{==========================================================================}
{          type declarations}
TYPE
           TfName     = String[80];        {1401 type: used to link segment Proload}
        DWORD         = 0..maxint;                 {graphics output type}
           INT        = -32768..32767;     {Graphics output type}
           INT0       = 0..65535;                  {Graphics output type}
           BYTE       = 0..255;             {Graphics Output type}
           INT1       = -128..127;                 {1401 : 8 bit ADC/DAC data}
           INT2       = -32768..32767;     {1401 : 16 bit ADC/DAC data}
           WORD       = 0..65535;                  {1401 : PSTH family array type}
        RECT    = RECORD
                            xl,yl,xr,yr : INT;
                     END;

           BAL =  Array[1..6] of real;
           Scan = Array[1..4] of real;

{==========================================================================}
{          common declarations}
COMMON cin,cout:TEXT;        {1401 files. Must be common for Proload}

{==========================================================================}
{          variable declarations}
VAR
           { Individual Test Data}
           testno:    integer;
           time:                 string[6];
           date: string[8];
           aerofoil,description: string;
           thickness,velocity,Mach_No,Reynolds_No: real;
           incidence,airtemp,density,chord,span:    real;
           {          i/o file names}
           datainput,dataoutput,file1,file2:text;
           rawoutput:text;
        Choice : Integer;
         {  wake traverse data}
           Ystep: real;
           Y : Array[1..maxsize] of REAL;
           height : real;
           NoElements : Integer;
        Reply, Finito, Try : boolean;
        DOSfilename : String;
```

```
        {    }
        Pitot1,Pitot2,Pitot3,Static : ARRAY[1..maxsize] OF Word;
        BV,SV : ARRAY[1..maxsize,1..6] of Word;

        {Dispseg Externals   written by K.M.Hyde}

PROCEDURE AnyKeyToCont;                                         EXTERNAL;
PROCEDURE PrintReal(Text:String;X:Real;F,D:Integer);       EXTERNAL;
FUNCTION InputInt(Text:String;Min,Max:Integer): Integer;EXTERNAL;
FUNCTION InputReal(Text:String;Min,Max:Real;f,d:integer) :Real;External;
FUNCTION Question(Text:String) : Boolean;                      EXTERNAL;
{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE ErrCheck( VAR height : REAL);
Var Num, Stat : Integer;
BEGIN
        writeln(file1,'K');
        rewrite(file1);
        readln(file2,Num);
        If Num > 127 then begin
                writeln(' Communications Fault ');
          Num := Num - 128;
                writeln(file1,'F');
                rewrite(file1);
                readln(file2,stat);
                writeln(stat,'  error detected ');
                Anykeytocont;
          end;
        If Num > 63 then begin
                writeln(' Limit Switch reached ');
          num := num - 64;
                writeln(file1,'I');
                rewrite(file1);
                readln(file2,stat);
          stat := stat - 64 - 128;
          If stat > 31 then begin
                        writeln(' Hit negative limit switch, only move in +ve dirn');
                        stat := stat - 32;
                        writeln(file1,'X 10000 @ 300$');
                        rewrite(file1);
                        height := -400.0;   { bottom limit}
                        end;
          If stat > 15 then begin
                        writeln(' Hit positive limit switch, only move in -ve dirn');
                        writeln(file1,'X-10000 @ 300$');
                        rewrite(file1);
                        height := 400.0;  { top limit }
                        end;
                end;
        If num > 31 then begin
                writeln(' Emergency stop activated :- Please release ');
                writeln(file1,'K');
                rewrite(file1);
                writeln(file1,'K');
                rewrite(file1);
        end;
END;
{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE finished(VAR height : REAL);
{checks that IF1 has moved traverse}
CONST
        c='C';
        e='E';
VAR
        test: char;
BEGIN
        REPEAT
```

```
                  writeln(file1,e);
                  rewrite(file1);
                  readln(file2,test);
          If test = 'F' then ErrCheck(height);    {  fault condition }
         UNTIL  test=c
END;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         Initialises stepper motor controller}
PROCEDURE InitTraverse;
{        initialises the IF1 controller  }
CONST
         Send = '(';                    {initialisation command}
VAR
         Answer: string;
BEGIN
         assign(file1,'com1');     {connect to the IF1}
         assign(file2,'com1');

         rewrite(file1);             {set up files}
         reset(file2);

         write(file1,send);             {initialise}
         rewrite(file1);                {clear buffer}

         read(file2,Answer); {read o.k. signal}

         page;
         writeln;
         writeln(' IF1   INITIALISED   O.K.');
         writeln;
         writeln('          correct reply signal  U :  ',Answer,'received');

         writeln(file1,'V41');    {set up data format};
         rewrite(file1);

         finished(height);

         AnyKeyToCont;

END;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
{   direct control of Wake Traverse}
PROCEDURE DirCont;
VAR
          fg : Boolean;
          otty : string;
BEGIN
          repeat
                   writeln('   Enter Data for next move ');
            readln(otty);
            writeln(file1,otty);
            rewrite(file1);
            finished(height);
            writeln('  finished   ');
            fg := Question('  Another Move ');
      until fg = false;
end;

{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                   Change parameters for Wake Traverse}
PROCEDURE ChangeParameter(VAR SS,Accrate,num,speed : Integer);
VAR
          zip : Boolean;
BEGIN

          writeln;
```

```
      writeln;
      zip := Question('do you wish to change pararmeters ? ');
      If zip = true then begin
              page;
        writeln(' Parameter SS is ',ss);
        ss:= InputInt('Enter new parameter SS',0,16);
        writeln(' Parameter AccRate is ',Accrate);
        accrate := InputInt('Enter new acceleration rate',0,16);
        writeln(' Parameter Num is ',Num);
        num := InputInt(' Enter new Num ',0,16);
        writeln(' Parameter speed is (steps/sec) ',speed);
        speed := InputInt(' Enter new speed',100,900);
     end;
end;
{+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      move traverse through given distance in millimetres}
PROCEDURE MoveTraverse(ystep: real;ss,accrate,num,speed : Integer
                                                   VAR height : real);
{         moves Wake Traverse through required distance in mm }
CONST
      {control characters for IF1}
       at        =          '@';
    dollar =       '$';
      colon      =          ':';
      kth        =          'K';
      e          =          'E';
      range      =          '>';

  startstop =       '<';

      acc        =          '^';

      neg        =          '-';
      c          =          'C';
      relay      =          'S';
      maxspeed = 1;
      r          =          'R';
      axis       =          'X';
      {calibration constant 1mm = No of Steps}


VAR
      file1,file2 : text;
      step : integer;
      flag,direction,test : char;
      calibration,distance,yscale : real;

BEGIN
      assign(file1,'com1');
      rewrite(file1);
      assign(file2,'com1');
      reset(file2);
      yscale  :=    0.002805;                         {provisional}
      writeln(file1,range,maxspeed,startstop,ss,acc,accrate,dollar);
      rewrite(file1);

      step := round(Ystep/Yscale);

    finished(height);

      If  step > 0 THEN writeln(file1,axis,step,at,speed,dollar)
      ELSE writeln(file1,axis,neg,step,at,speed,dollar);

      rewrite(file1);
```

```
        finished(height);

END;
{++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
PROCEDURE SetToLimit(VAR height : REAL);
{                   Procedure drives traverse to bottom limit and then backs off to
allow a height datum to be set      }

CONST
{control characters for IF1}
          at        =          '@';
       dollar =        '$';
        colon     =          ':';
        kth       =          'K';
        e         =          'E';
        range     =          '>';
     speedhigh  = 750;
     speedlow = 300;
    startstop =       '<';
        ss = 3;
        Accrate = 3;
        acc       =          '^';
      go = 'G';
        neg       =          '-';
        c         =          'C';
        relay     =          'S';
        maxspeed = 1;
        r         =          'R';
        axis      =          'X';
VAR step : integer;
        yscale : real;
BEGIN
        page;
        writeln;
        writeln('       SETTING TRAVERSE TO DATUM LEVEL');
{      assign(file1,'com1');
        rewrite(file1);
        assign(file2,'com1');
        reset(file2);}
        yscale  :=    0.002805;                    {provisional}
{       writeln(file1,range,maxspeed,startstop,ss,acc,accrate,dollar);
        rewrite(file1);}

        writeln(file1,axis,neg,at,speedhigh,go,dollar);
                                                                                  { starts
raverse moving down}
        rewrite(file1);

        finished(height);

        { move traverse 200 mm off bottom limit }

        step := round(200.0/yscale);
        writeln(file1,axis,step,at,speedlow,dollar);
        rewrite(file1);

        height := -200.0;

        writeln;
        writeln('           Traverse zeroed to datum and set to 200mm up');
        writeln;
        AnyKeyToCont;
end;
{************************************************************************}
BEGIN
END.
```

Appendix C

Sample Data Files

```
107   - Test Number   Raw Data
**************************************************
                    BALANCE       Average Zero Reading (voltage)
1    Lift           Drag          Moment        (RHS)
       0.001603        0.007586       0.003337
2    Lift           Drag          Moment        (LHS)
       0.002393       -0.004152      -0.001201

**************************************************
                    BALANCE       Standard Deviation (voltage)
1    Lift           Drag          Moment        (RHS)
       0.000011        0.000002       0.000005
2    Lift           Drag          Moment        (LHS)
       0.000005        0.000011       0.000005

**************************************************
SCANIVALVE   Average Zero Reading (voltage)
        Pressure Transducer
Channel   1               2               3               4
      -0.001030        0.001062       -0.001223       -0.000612

**************************************************
SCANIVALVE    Standard Deviation
        Pressure Transducer
Channel   1               2               3               4
       0.000009        0.000064        0.000016        0.000009

          1    Data Entry
-5.0000000E+01    mm from Wake Traverse Datum
 7.9287999E+00    Supply Voltage to Balance
**************************************************
                    BALANCE       Uncorrected Balance Voltages
1    Lift           Drag          Moment        (RHS)
       0.004345        0.005857       0.001906
2    Lift           Drag          Moment        (LHS)
       0.005153       -0.001531      -0.002836

**************************************************
SCANIVALVE     Uncorrected Scani voltages
        Pressure Transducer
Channel   1               2               3               4
      -0.000079        0.002645        0.000372        0.000733

          2    Data Entry
-2.5000000E+01    mm from Wake Traverse Datum
 7.9288101E+00    Supply Voltage to Balance
**************************************************



          4    Data Entry
-1.0000000E+01    mm from Wake Traverse Datum
 7.9289400E+00    Supply Voltage to Balance
**************************************************
                    BALANCE       Uncorrected Balance Voltages
1    Lift           Drag          Moment        (RHS)
       0.004281        0.005825       0.001868
2    Lift           Drag          Moment        (LHS)
       0.005126       -0.001546      -0.002868

**************************************************
SCANIVALVE     Uncorrected Scani voltages
        Pressure Transducer
Channel   1               2               3               4
       0.000145        0.002491        0.000234        0.000542

          5    Data Entry
-1.5000000E+01    mm from Wake Traverse Datum
 7.9289102E+00    Supply Voltage to Balance
**************************************************
                    BALANCE       Uncorrected Balance Voltages
1    Lift           Drag          Moment        (RHS)
       0.004293        0.005846       0.001867
2    Lift           Drag          Moment        (LHS)
       0.005150       -0.001544      -0.002874

**************************************************
SCANIVALVE     Uncorrected Scani voltages
        Pressure Transducer
Channel   1               2               3               4
       0.000121        0.002560        0.000294        0.000547
```

```
 107   -  Test Number    ProcData
         1     Data Entry
 -5.0000000E+01   mm from Wake Traverse Datum
**************************************************
                    BALANCE       Lift, Drag and Moment
1   Lift            Drag          Moment       (RHS)
    -284.051500     -25.459259        2.961850
2   Lift            Drag          Moment       (LHS)
    -280.480440      41.046991      -42.219489

**************************************************
SCANIVALVE      Pressures (N/M^2)
        Pressure Transducer
Channel   1              2                3             4
     878.697210    1462.378300    1242.337700    1473.184900

        Velocities (m/s)
Pitot  1               2                3
 3.8137462E+01  4.9199652E+01  4.5347359E+01

         2     Data Entry
 -2.5000000E+01   mm from Wake Traverse Datum
**************************************************
                    BALANCE       Lift, Drag and Moment
1   Lift            Drag          Moment       (RHS)
    -278.255430     -26.385506        2.281334
2   Lift            Drag          Moment       (LHS)
    -278.357320      41.433387      -42.336532

**************************************************
SCANIVALVE      Pressures (N/M^2)
        Pressure Transducer
Channel   1              2                3             4
    1086.059400    1428.565000    1072.330600    1466.209700

        Velocities (m/s)
Pitot  1               2                3
 4.2399325E+01  4.8627525E+01  4.2130488E+01


         7     Data Entry
 1.4060525E+00   mm from Wake Traverse Datum
**************************************************
                    BALANCE       Lift, Drag and Moment
1   Lift            Drag          Moment       (RHS)
    -273.968630     -25.970292        2.025119
2   Lift            Drag          Moment       (LHS)
    -282.277370      41.187149      -42.746245

**************************************************
SCANIVALVE      Pressures (N/M^2)
        Pressure Transducer
Channel   1              2                3             4
    1177.491500    1135.698800    1164.717500    1153.742000

        Velocities (m/s)
Pitot  1               2                3
 4.4148004E+01  4.3357452E+01  4.3907877E+01

         8     Data Entry
 6.4060527E+00   mm from Wake Traverse Datum
**************************************************
                    BALANCE       Lift, Drag and Moment
1   Lift            Drag          Moment       (RHS)
    -277.908590     -25.236999        2.310605
2   Lift            Drag          Moment       (LHS)
    -282.476630      40.670111      -42.835354

**************************************************
SCANIVALVE      Pressures (N/M^2)
        Pressure Transducer
Channel   1              2                3             4
    1235.572000    1090.999000    1233.206700    1132.693300

        Velocities (m/s)
Pitot  1               2                3
 4.5223709E+01  4.2495635E+01  4.5180404E+01
```

```
 107  -  Test Number    FinalData
18     Total Number of Data Readings
-3.69104680000021E-0003      Coarse Traverse Drag Cd
02/11/88
11:45
SSS-127
  47.200   0.270   1.208
 609.600 595.000 chord   span
 -12.000  19.900 incidence  airtemp
101615.4400 1079.1000 tunnel stat & dyn press
 1345.9320 splitter pressure
  764.5500 atmospheric pressure
Inverted No Rough Re 2*10^6
      End Of File
```

Appendix D

Data Points

APPENDIX D

1) LS(1) 0421 mod Aerofoil

(i) Reynolds Number 2000000

Angle of Attack	Cl	Cd	Cm	
17.97375	0.92914	0.09759	-0.06628	
14.97693	1.35311	0.08380	-0.08523	
11.97564	1.41495	0.03775	-0.08164	
7.96002	1.34113	0.01948	-0.08447	
5.94049	1.20405	0.01324	-0.09074	
3.90035	0.87482	0.01242	-0.09227	
-0.15081	0.41166	0.00891	-0.08365	* WT
-0.15232	0.39253	0.00892	-0.08203	* WT
-2.17558	0.18840	0.01065	-0.07987	
-4.20082	-0.06046	0.01895	-0.07067	
-6.22777	-0.34663	0.03581	-0.05559	

(ii) Reynolds Number 2000000
Roughness Strips attached

Angle of Attack	Cl	Cd	Cm	
14.95479	1.18184	0.12750	-0.09487	
9.95227	1.23902	0.02645	-0.07527	
5.92314	0.99345	0.0192	-0.07403	* WT
-0.15345	0.31074	0.0145	-0.06398	* WT

2) SSS-121 Aerofoil

(i) Reynolds Number 1600000

Angle of Attack	Cl	Cd	Cm	
17.98637	1.13307	0.09748	-0.02138	
14.98051	1.16771	0.08521	-0.01757	*0.06865
11.96642	1.24140	0.07295	-0.05541	
9.96536	1.14773	0.06823	-0.02646	
5.94097	0.96781	0.00999	-0.03130	* WT
3.92377	0.84765	0.00916	-0.03639	* WT
1.89451	0.60610	0.00896	-0.03711	* WT
-0.12502	0.40800	0.011	-0.02854	* WT
-4.16955	-0.00531	0.0262	-0.01872	* WT

(ii) Reynolds Number 2000000

Angle of Attack	Cl	Cd	Cm
5.88475	0.45984	0.05617	-0.02159
5.88140	0.45454	0.05393	-0.02722
-0.12927	0.39106	0.01433	-0.03327

3) SSS-123 Aerofoil

(i) Reynolds Number 2000000
Roughness strips attached

Angle of Attack	Cl	Cd	Cm
16.94625	0.96589	0.15273	-0.06406
11.92771	0.92035	0.11576	-0.06172
7.38918	0.63817	0.07240	-0.05737

(ii) Reynolds Number 2000000

Angle of Attack	Cl	Cd	Cm	
16.97756	1.33740	0.14485	-0.07750	
14.97236	1.33755	0.12500	-0.07622	
11.95929	1.22355	0.08997	-0.06668	
8.94891	1.13205	0.05325	-0.05661	
7.44203	1.04337	0.03954	-0.04829	
5.93329	0.97099	0.00810	-0.04784	* wt
2.90322	0.66940	0.00819	-0.03495	* wt
-0.13288	0.31014	0.01014	-0.02062	* wt
-3.16775	0.00636	0.01090	-0.01783	* wt

(iii) Reynolds Number 1600000

Angle of Attack	Cl	Cd	Cm	
14.97185	0.87098	0.08174	-0.04813	
11.96046	1.21547	0.08900	-0.06152	
8.94924	1.11351	0.05242	-0.05125	
5.93216	0.95177	0.00980	-0.04540	* wt
2.90274	0.67079	0.00924	-0.03629	* wt
-0.13143	0.35180	0.01170	-0.02811	* wt
-3.16427	0.07072	0.01230	-0.02664	* wt

4) SSS-125 Aerofoil

(i) Reynolds Number 2000000
Roughness strips attached

Angle of Attack	Cl	Cd	Cm
17.93439	0.77532	0.15012	-0.06266
13.92460	0.92042	0.14341	-0.07332
7.88856	0.64998	0.07918	-0.0619
3.86237	0.46430	0.05854	-0.06959
1.85121	0.36805	0.04793	-0.06870

(ii) Reynolds Number 2000000

Angle of Attack	Cl	Cd	Cm	
17.96022	1.07270	0.10505	-0.05634	
14.95751	1.21019	0.12590	-0.07948	
13.95469	1.24178	0.11508	-0.08467	
11.94798	1.19100	0.09485	-0.07871	
8.93916	1.08738	0.07317	-0.06657	
5.92541	0.92948	0.00812	-0.05467	*wt
5.92294	0.94512	0.00876	-0.06361	*wt
1.88572	0.55447	0.01000	-0.04301	*wt
-0.13025	0.40280	0.01026	-0.03843	*wt
-2.15575	0.17791	0.01190	-0.03558	*wt

(iii) Reynolds Number 1600000

Angle of Attack	Cl	Cd	Cm	
13.955	1.16881	0.11059	-0.07764	
9.92133	0.98826	0.06422	-0.0784	
7.92690	0.94426	0.05106	-0.05553	
3.89882	0.74481	0.01265	-0.06359	* WT
1.88499	0.58042	0.01160	-0.05109	* WT
-2.15603	0.17281	0.01278	-0.03490	* WT

5) SSS-127 Aerofoil

(i) Reynolds Number 2000000

Angle of Attack	Cl	Cd	Cm	
19.47579	1.25254	0.17900	-0.07143	
17.94566	1.01464	0.14202	-0.05160	
16.46476	1.06313	0.13005	-0.05805	
14.95712	1.16502	0.13803	-0.06267	
13.45442	1.00895	0.10407	-0.04377	
11.94869	1.05734	0.09926	-0.04618	
8.93557	0.96792	0.07672	-0.04448	
4.91506	0.74867	0.00919	-0.03080	* wt
0.87528	0.35986	0.00954	-0.01627	* wt
-3.16845	-0.01982	0.01110	-0.01272	* wt

(ii) Reynolds Number 1600000

Angle of Attack	Cl	Cd	Cm	
17.95289	1.10652	0.14175	-0.05873	
14.95954	1.10259	0.12312	-0.05533	
11.94928	1.07449	0.09488	-0.04899	
8.93805	0.98462	0.08018	-0.04362	
4.91414	0.74154	0.01005	-0.03094	* wt
0.87576	0.37093	0.01100	-0.01803	* wt
-3.16714	-0.00258	0.01289	-0.01430	* wt

6) SSS-129 Aerofoil

(i) Reynolds Number 2000000

Angle of Attack	Cl	Cd	Cm	
9.92711	0.87918	0.11548	-0.04099	
7.91989	0.82314	0.09417	-0.04128	
5.92292	0.69150	0.01548	-0.00182	* wt
3.89895	0.55427	0.01005	-0.01651	* wt
1.89142	0.36319	0.01001	0.01511	* wt
-0.13363	0.24368	0.01083	-0.00632	* wt
-4.17416	-0.07905	0.02325	-0.00963	* wt

(ii) Reynolds Number 2000000

Angle of Attack	Cl	Cd	Cm	
7.92994	0.78863	0.13106	-0.01286	
5.92375	0.66644	0.01232	0.00634	* wt
3.90063	0.51875	0.01090	-0.00464	* wt
1.88799	0.36277	0.01120	0.00854	* wt
-0.13247	0.23778	0.01262	-0.00225	* wt
-4.17157	-0.09102	0.01470	-0.00134	* wt

APPENDIX E

POST-STALL WAKE CORRECTION

An Examination of Wake Blockage for Stalled Aerofoils Using an Actuator Surface Approach

Outline of Problem

In 2D aerofoil testing, the presence of a viscous wake in a parallel-sided tunnel results in an increase in flow speed outside the wake with a consequent reduction in tunnel static pressure. Axial velocities in the region of the aerofoil are changed by the presence of the wake and the purpose of this note is to examine the flow with a view to calculating a representative free stream axial velocity.

Nomenclature

A	–	cross sectional area of tunnel/unit span
a	–	cross sectional area of actuator surface/unit span
a_3	–	cross sectional developed wake/unit span
U_0	–	upstream (undisturbed) tunnel velocity
U	–	velocity through actuator surface
U_3	–	velocity in developed wake
U_e	–	velocity of flow outside developed wake
p_0	–	static pressure of upstream (undisturbed) flow
p_1	–	static pressure immediately upstream of actuator surface
p_2	–	static pressure immediately downstream of actuator surface
p_e	–	static pressure in flow outside developed wake
D	–	drag of actuator surface
ρ	–	fluid density
V_o	–	equivalent 'free air' free stream velocity

Basic Concepts

The stalled aerofoil produces a viscous wake of indeterminate size. An equivalent actuator surface is sought such that the drag (D) and velocity outside the wake (U_e) are the same as those for the aerofoil with tunnel speed U_0. The fluid speed (U) at the actuator surface can then be calculated and compared with a similar calculation for an actuator surface in free air, with the same values of a, D and U but for which the undisturbed free airspeed (V_0) is sought. The value of V_0 provides a datum airspeed for the test.

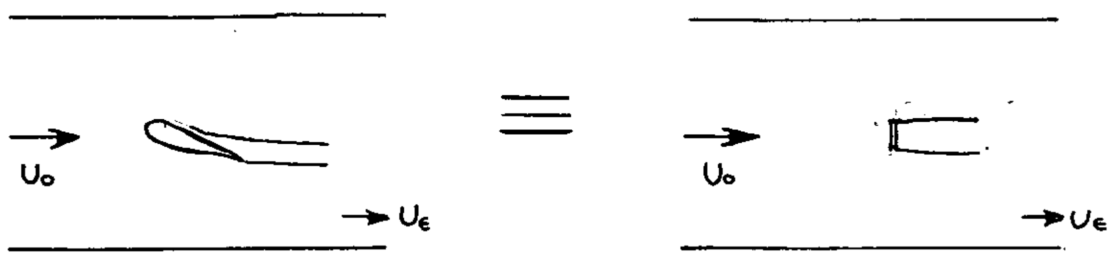


Fig. 1: Actuator Surface Equivalent to Stalled Aerofoil

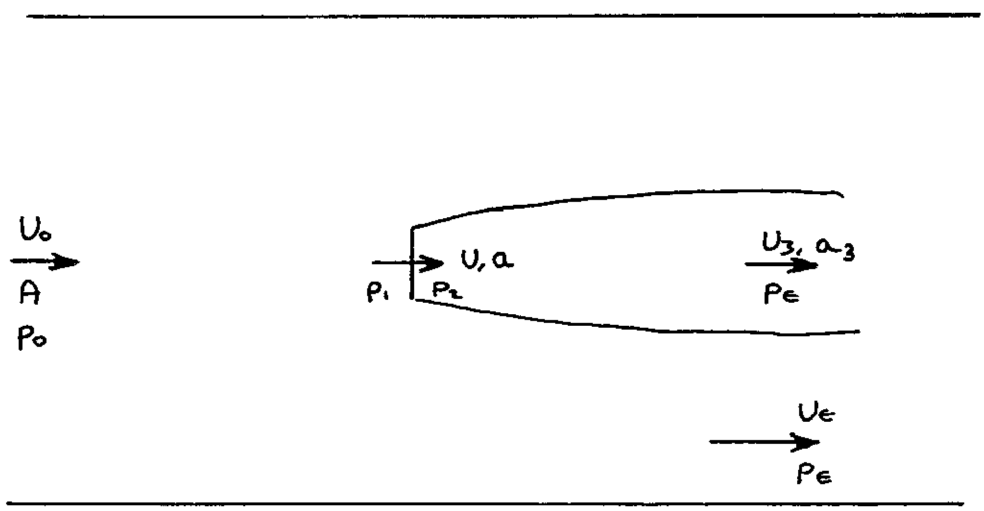


Fig. 2: Nomenclature Shown in Context of Flow

Basic Equations

Continuity through the actuator surface leads to:

$$a\ U = a_3\ U_3 \tag{1}$$

Continuity within the tunnel gives:

$$a\ U_0 = (A - a_3)\ U_e + a_3\ U_3$$

which is developed as:

$$a_3 = A\ (U_0 - U_e)/(U_3 - U_e) \tag{2}$$

Bernoulli upstream of the actuator surface gives:

$$p_0 + \tfrac{1}{2}\rho U_0^2 = p_1 + \tfrac{1}{2}\rho U^2$$

which is written:

$$p_1 = p_0 + \tfrac{1}{2}\rho U_0^2 - \tfrac{1}{2}\rho U^2 \tag{3}$$

Bernoulli applied downstream of the actuator surface gives:

$$p_2 = p_e + \tfrac{1}{2}\rho U_3^2 - \tfrac{1}{2}\rho U^2 \tag{4}$$

Subtracting (4) from (3) provides an expression for the pressure difference across the actuator surface that can be related to the drag via:

$$\frac{D}{a} = p_1 - p_2$$

i.e.

$$\frac{D}{a} = p_0 - p_e + \tfrac{1}{2}\rho\ (U_0^2 - U_3^2) \tag{5}$$

Away from the actuator disc flow, the Bernoulli equation can be applied to provide an expression for $p_0 - p_e$ in terms of ρ, U_0 and U_e that can be substituted into (5) to obtain

$$\frac{D}{a} = \tfrac{1}{2}\rho\ (U_e^2 - U_0^2) + \tfrac{1}{2}\rho\ (U_0^2 - U_3^2)$$

leading to:

$$\frac{D}{a} = \tfrac{1}{2}\rho\ (U_e^2 - U_3^2) \qquad (6)$$

Examination of momentum in the tunnel at the steamwise ordinate of the developed wake leads to:

$$D = \rho\ U_3\ a_3\ (U_0 - U_3) + (A - a_3)\ \rho\ U_e\ (U_0 - U_e) + A\ (p_0 - p_e) \qquad (7)$$

Using Bernoulli to express $p_0 - p_e$ in terms of velocity,

$$D = \rho\ U_3\ a_3\ (U_0 - U_3) + (A - a_3)\ \rho\ U_e\ (U_0 - U_e) + \tfrac{1}{2}\rho A\ (U_e^2 - U_0^2) \qquad (8)$$

Using (2) in (8):

$$D = \rho\ U_3\ (U_0 - U_3)\ .\ A\ \frac{(U_0 - U_e)}{(U_3 - U_e)} + A\ \{1 - \frac{U_0 - U_e}{U_3 - U_e}\}\ \rho\ U_e\ (U_0 - U_e) + \tfrac{1}{2}\rho A\ (U_e^2 - U_0^2) \qquad (9)$$

Dividing by ρA and multiplying by $(U_3 - U_e)$ gives:

$$\frac{D}{\rho A} (U_3 - U_e) = U_3 (U_0 - U_3) (U_0 - U_e) + U_e (U_3 - U_0) (U_0 - U_e)$$

$$+ \tfrac{1}{2} (U_e^2 - U_0^2) (U_3 - U_e) \qquad (10)$$

After reduction,

$$D = \tfrac{1}{2}\rho A (U_0 - U_e) (U_0 - U_e - 2U_3)$$

$$\therefore \qquad \frac{2D}{\rho A (U_0 - U_e)} = U_0 - U_e - 2U_3$$

$$2U_3 = U_0 - U_e - \frac{2D}{\rho A (U_0 - U_e)}$$

$$U_3 = \frac{U_0 - U_e}{2} + \frac{D}{\rho A (U_e - U_0)} \qquad (11)$$

<u>Solution for V_0</u>

A, D, ρ, U_0 and U_e are known

- Equation (11) is solved for U_3
- Equation (6) is restated as

$$a = \frac{2D}{\rho (U_e^2 - U_3^2)} \qquad (12)$$

  and solved for a

- Equation (2) is solved for a_3
- Equation (1) is re-stated as:

$$U = \frac{a_3}{a} U_3 \text{ and solved for } U \qquad (13)$$

- U, a, D and ρ are all now known and the remaining part of the problem is to examine free-air actuator disc theory to obtain a value of V_0 consistent with these inputs. Free-air actuator disc theory leads to the equation

$$U = \frac{V_0}{2} + \sqrt{\left(\frac{V_0}{2}\right)^2 - \frac{D}{2\rho A}} \qquad (14)$$

re-arranging and squaring,

$$\left(U - \frac{V_0}{2}\right)^2 = \left(\frac{V_0}{2}\right)^2 - \frac{D}{2\rho a}$$

cancelling $\left(\frac{V_0}{2}\right)^2$,

$$U^2 - U V_0 = - \frac{D}{2\rho a}$$

Hence,

$$V_0 = U + \frac{D}{2\rho a U} \qquad (15)$$

V_0 now represents the effective free-stream velocity consistent with energy and momentum changes observed.

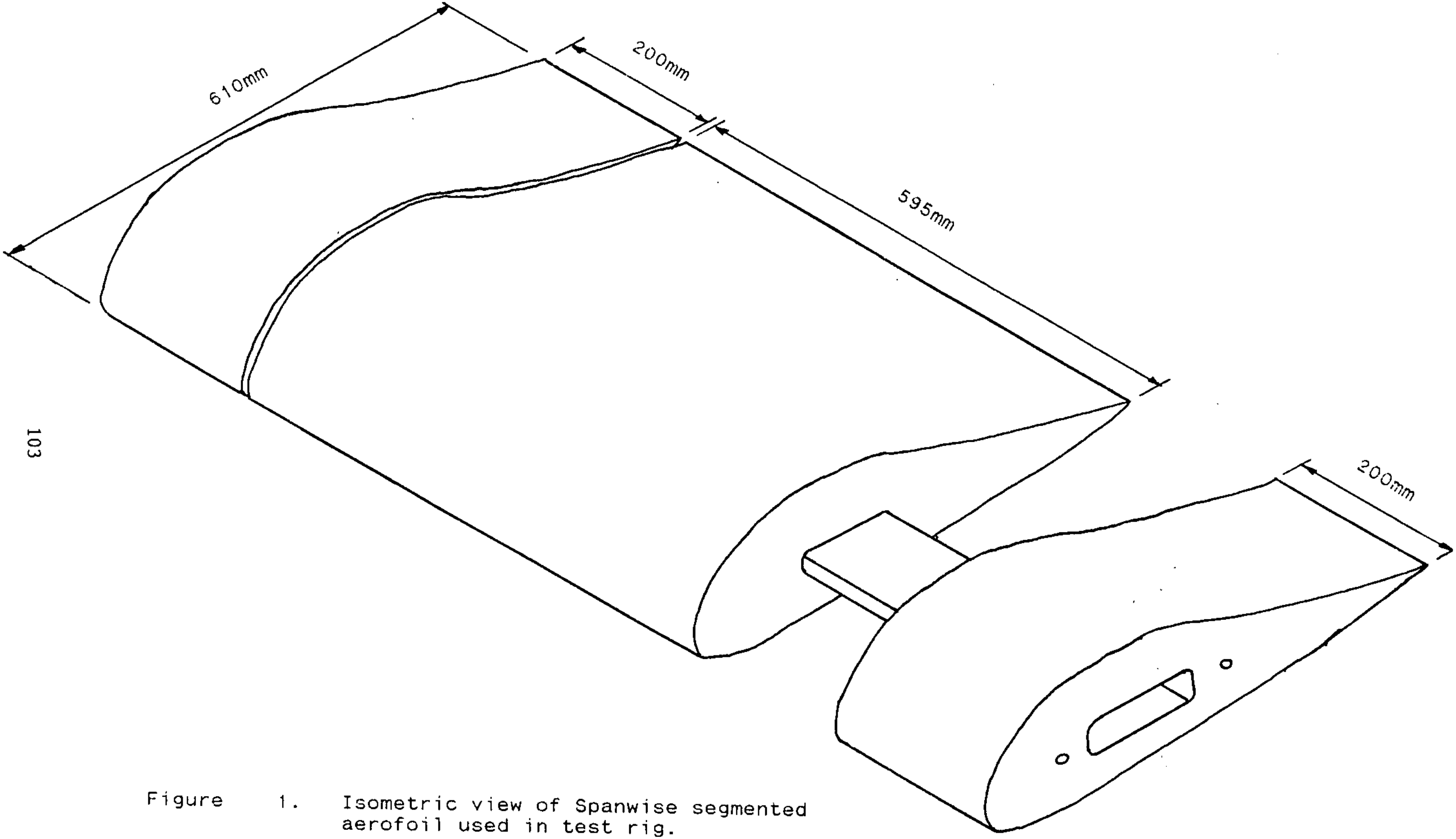


Figure 1. Isometric view of Spanwise segmented aerofoil used in test rig.

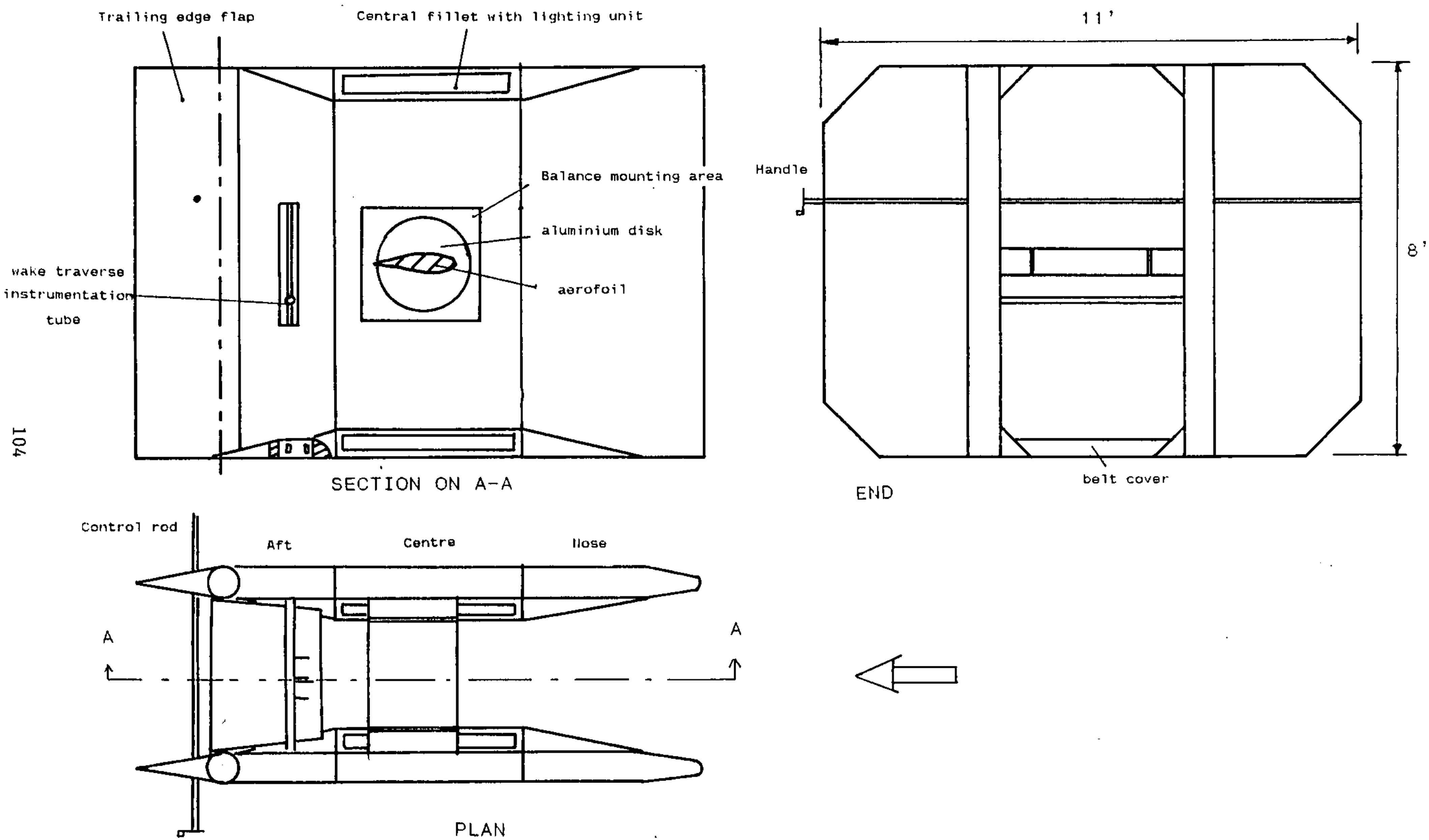


Figure 2. Two-Dimensional Test Rig Overall Layout

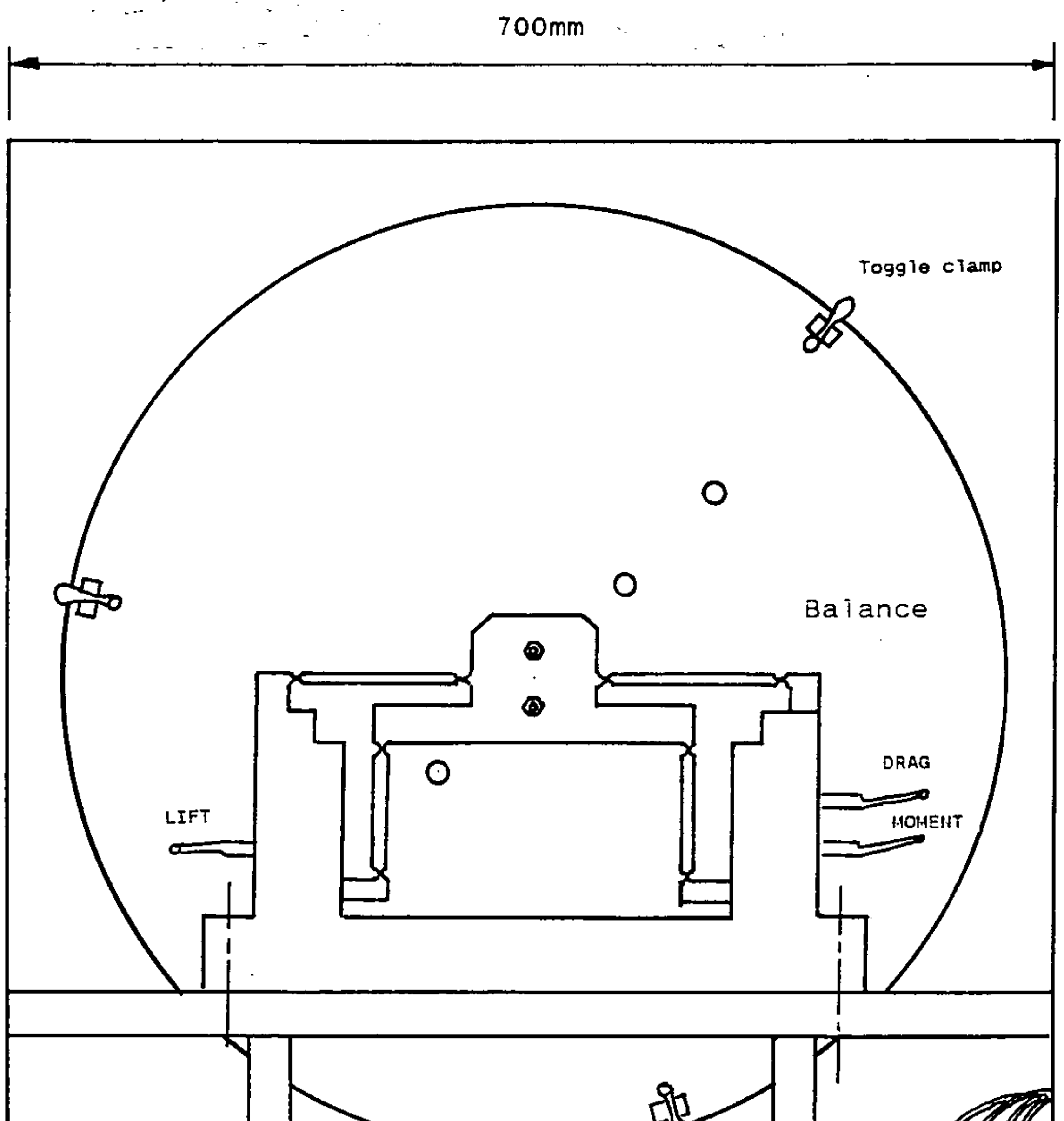


Figure 3. Strain Gauge Balance Mounting area

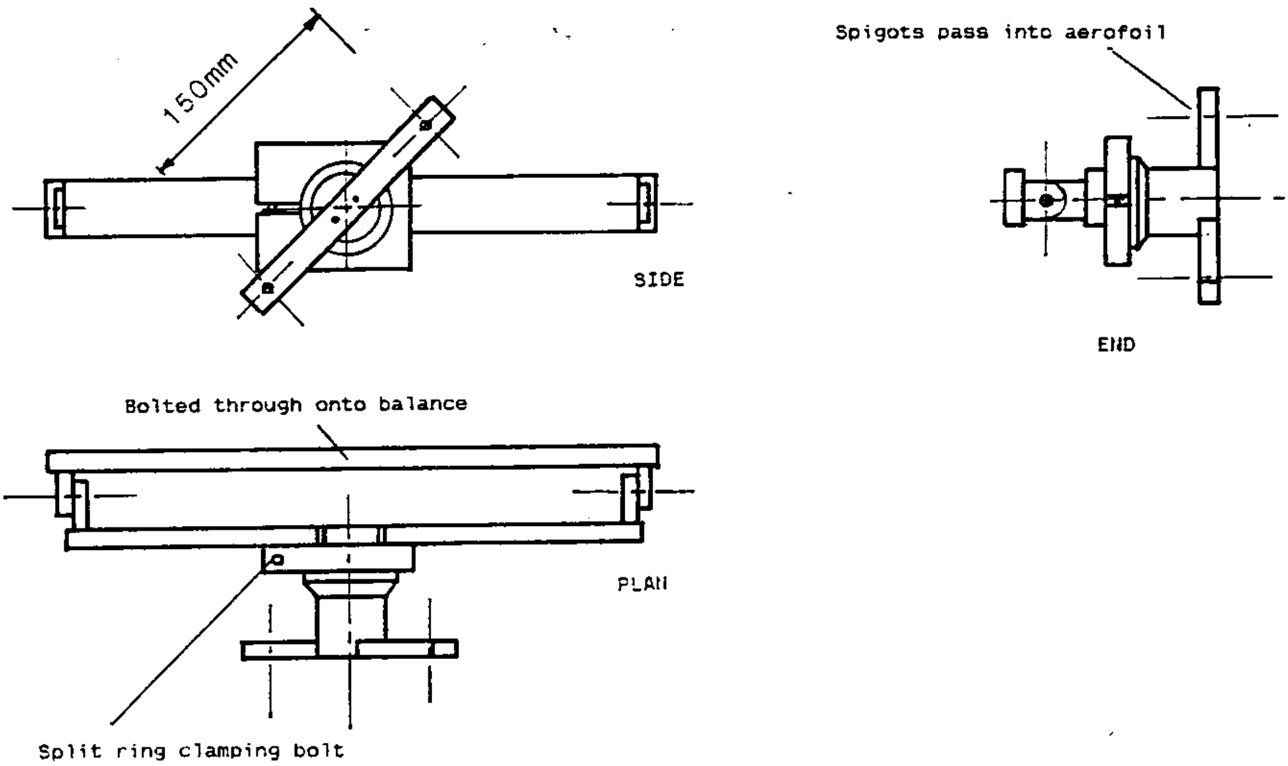


Figure 4. Yoke arrangement for transmitting forces between aerofoil and strain gauge balance.

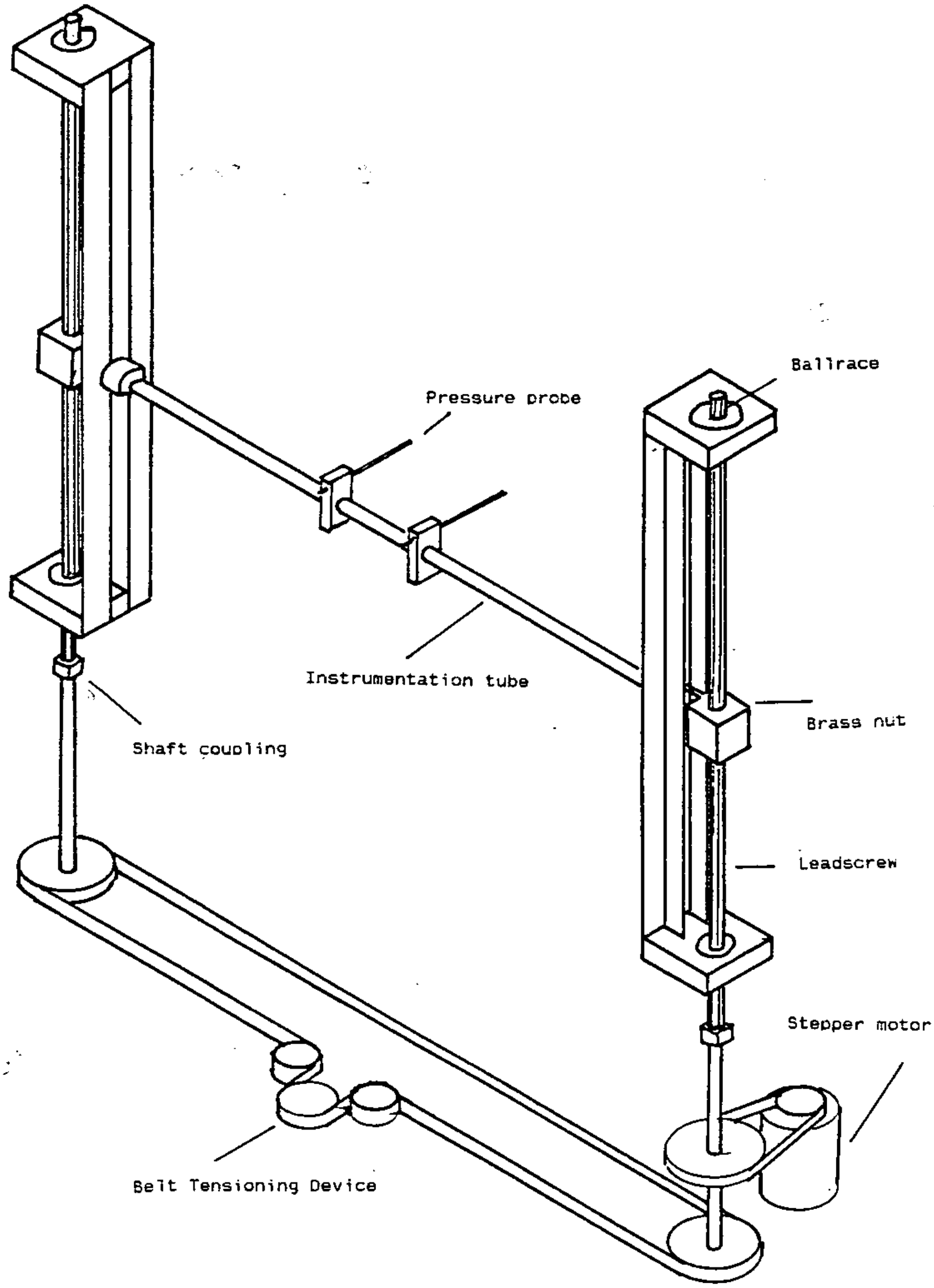


Figure 5. Schematic of Wake Traverse Mechanism.

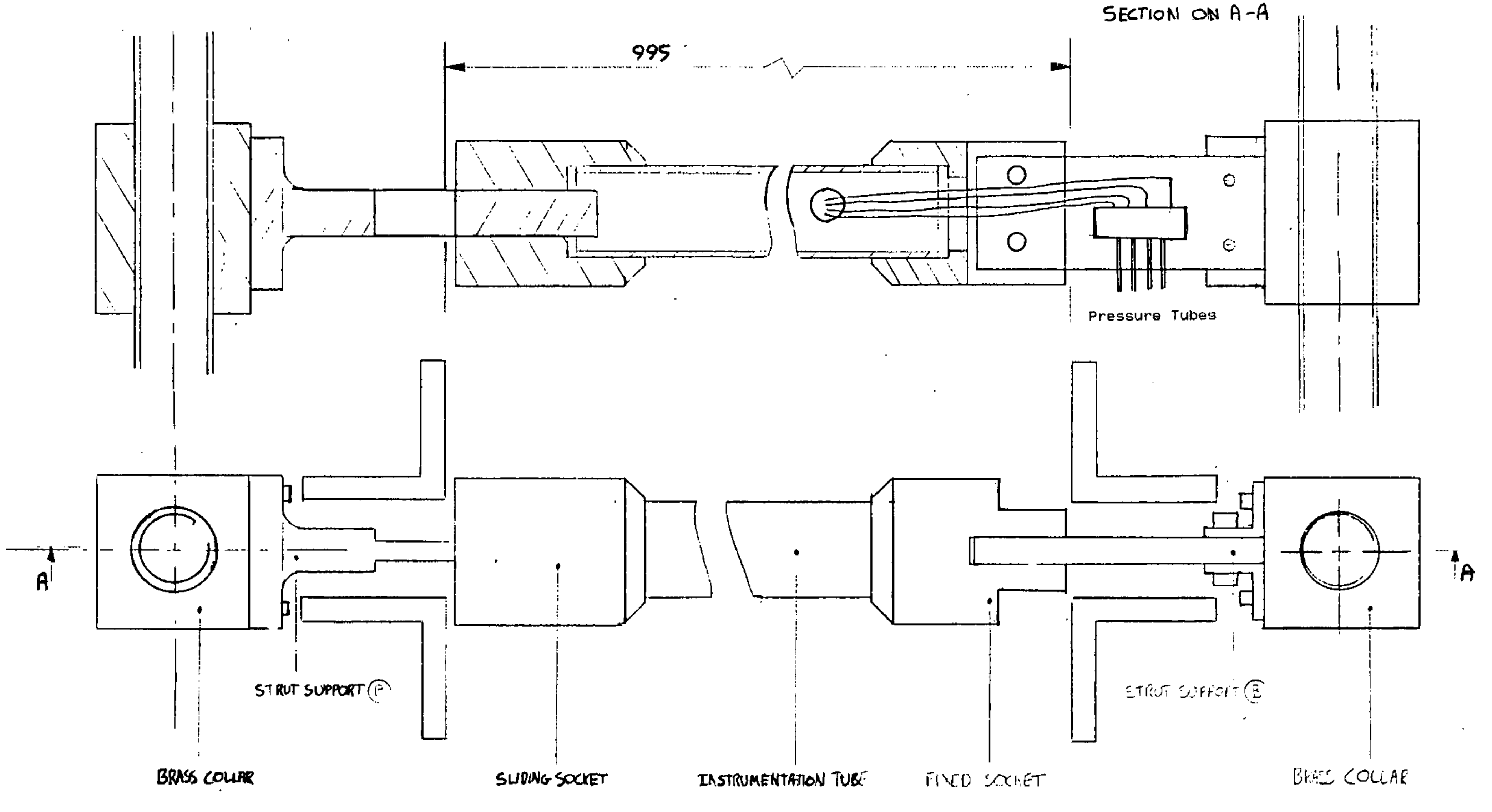


Figure 6. Instrumentation Tube for Wake Traverse.

WAKE TRAVERSE : COLLAR&STRUT SubAssembly

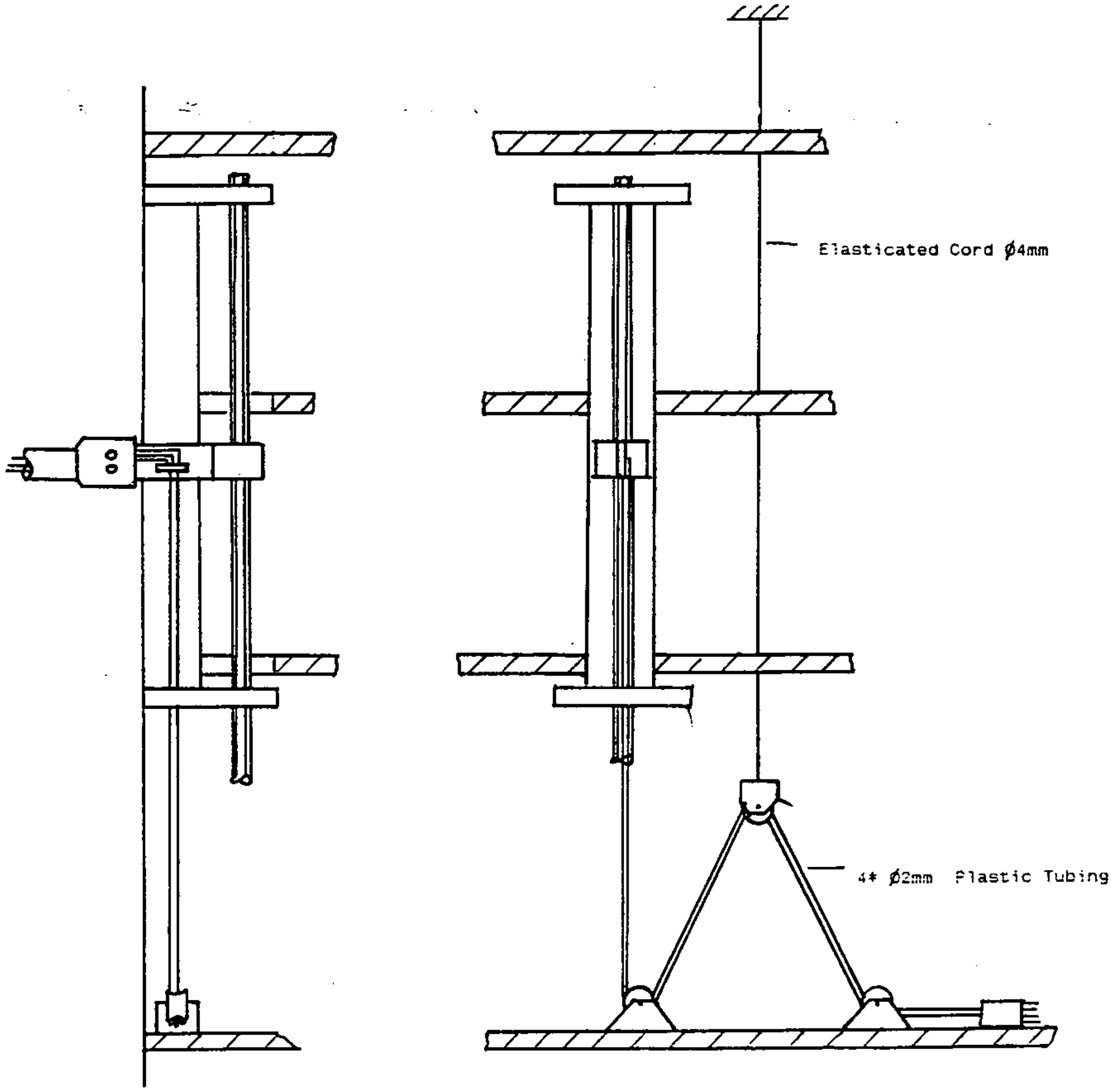


Figure 7. Wake Traverse Pressure Tube Tensioning Device

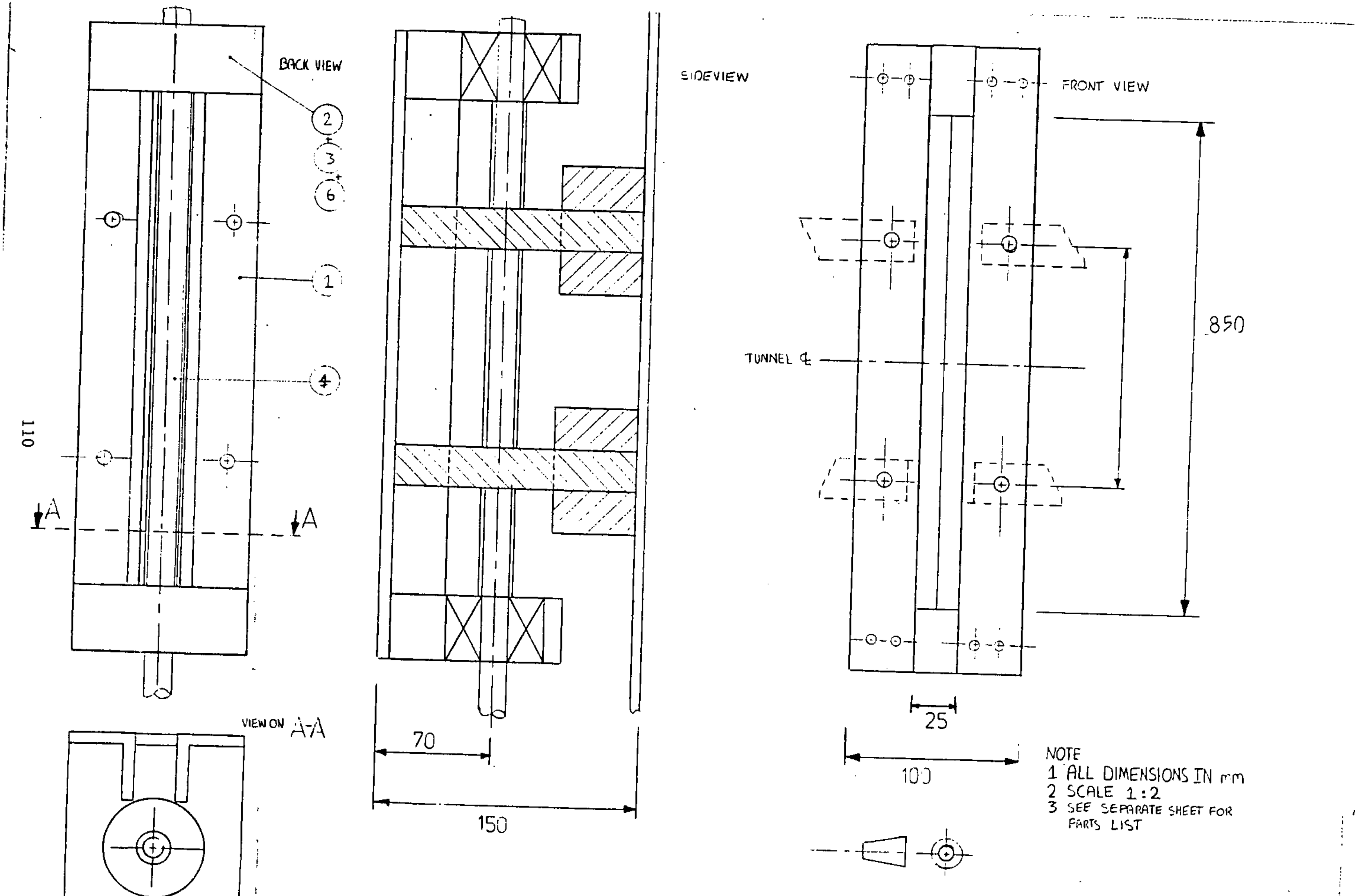
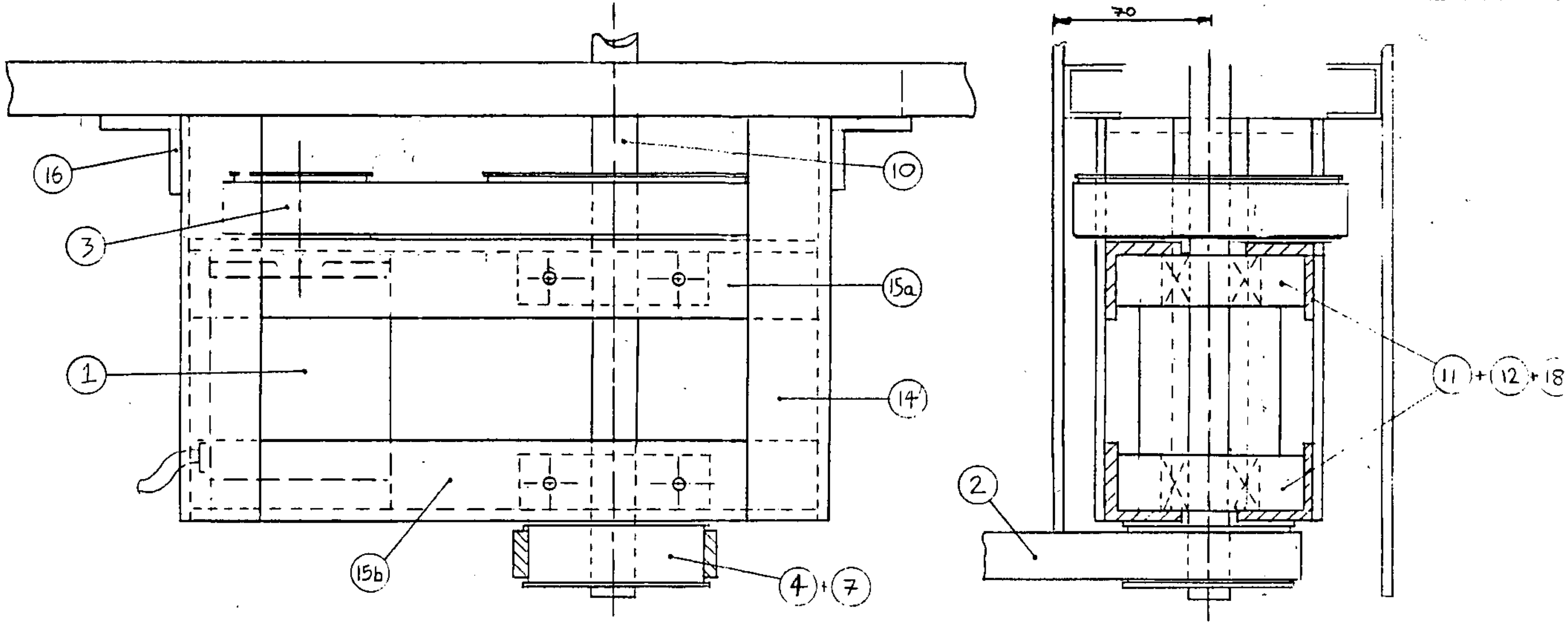
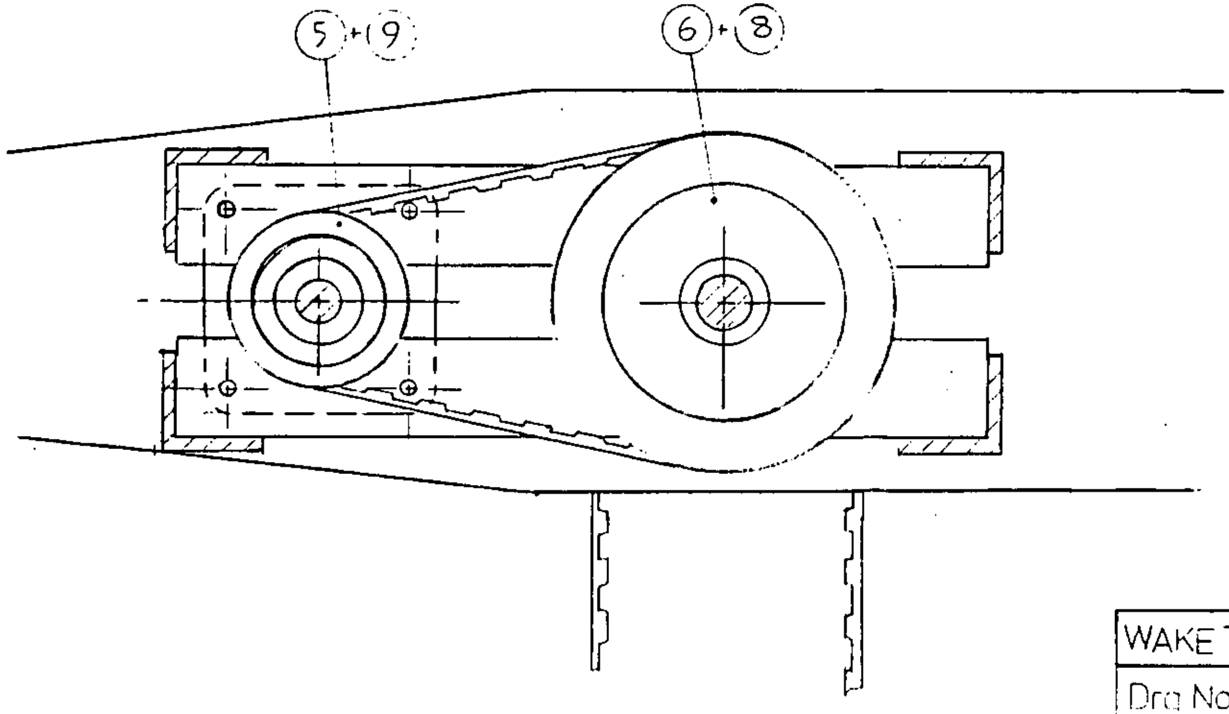


Figure 8. Wake Traverse Leadscrew and Nut sub Assembly





NOTE.
1 ALL DIMENSIONS IN MILLIMETRES
2. SCALE 1:2
3 SEE SEPARATE SHEET FOR PARTS LIST

Figure 9.

Wake Traverse Pulley and Motor Sub-assembly

WAKE TRAVERSE : MOTOR & PULLEY SubAssembly		
Drg No: 5A3	Drawn By: S Turnock	Date: 20/4/88

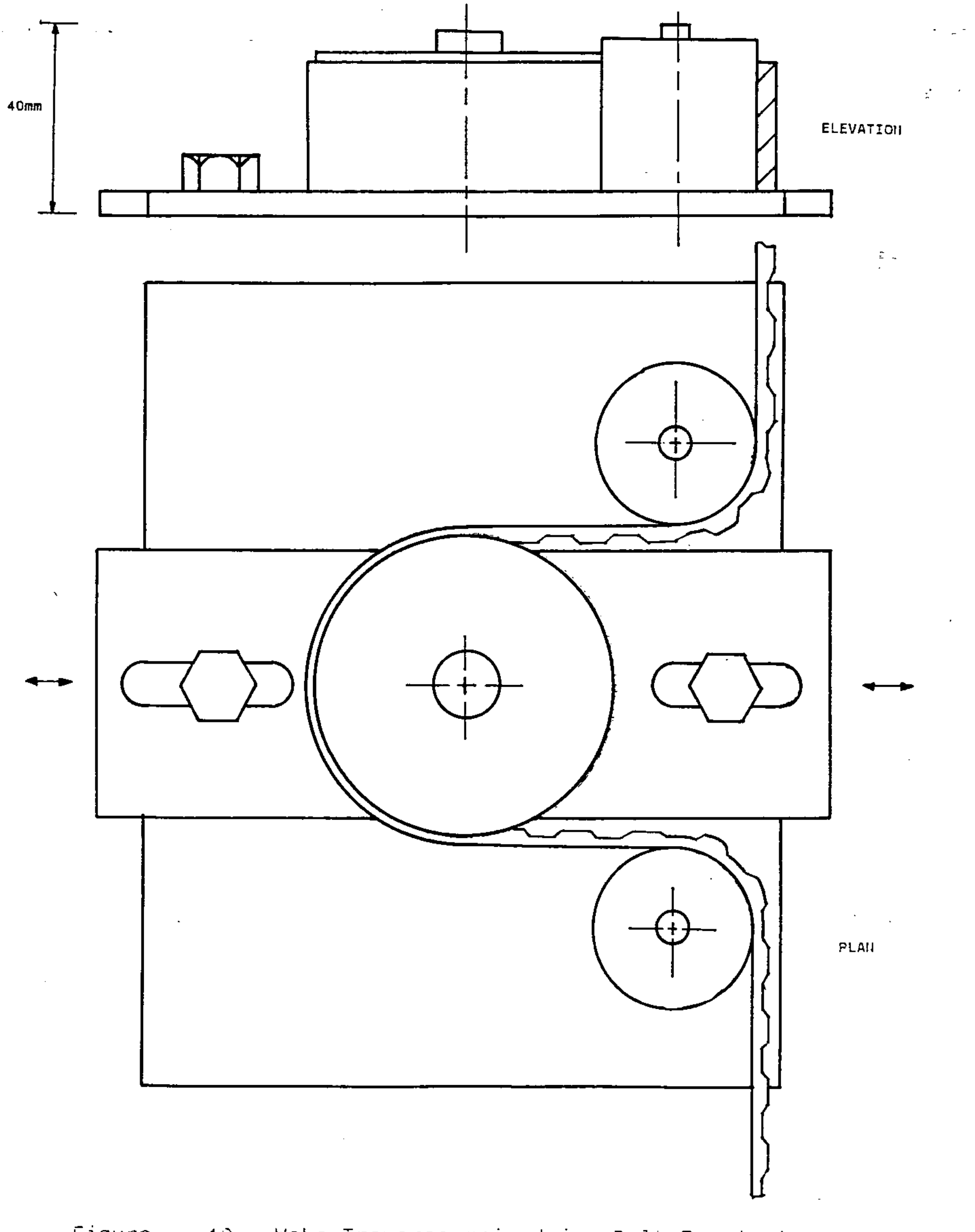


Figure 10. Wake Traverse main drive Belt Tensioning Device.

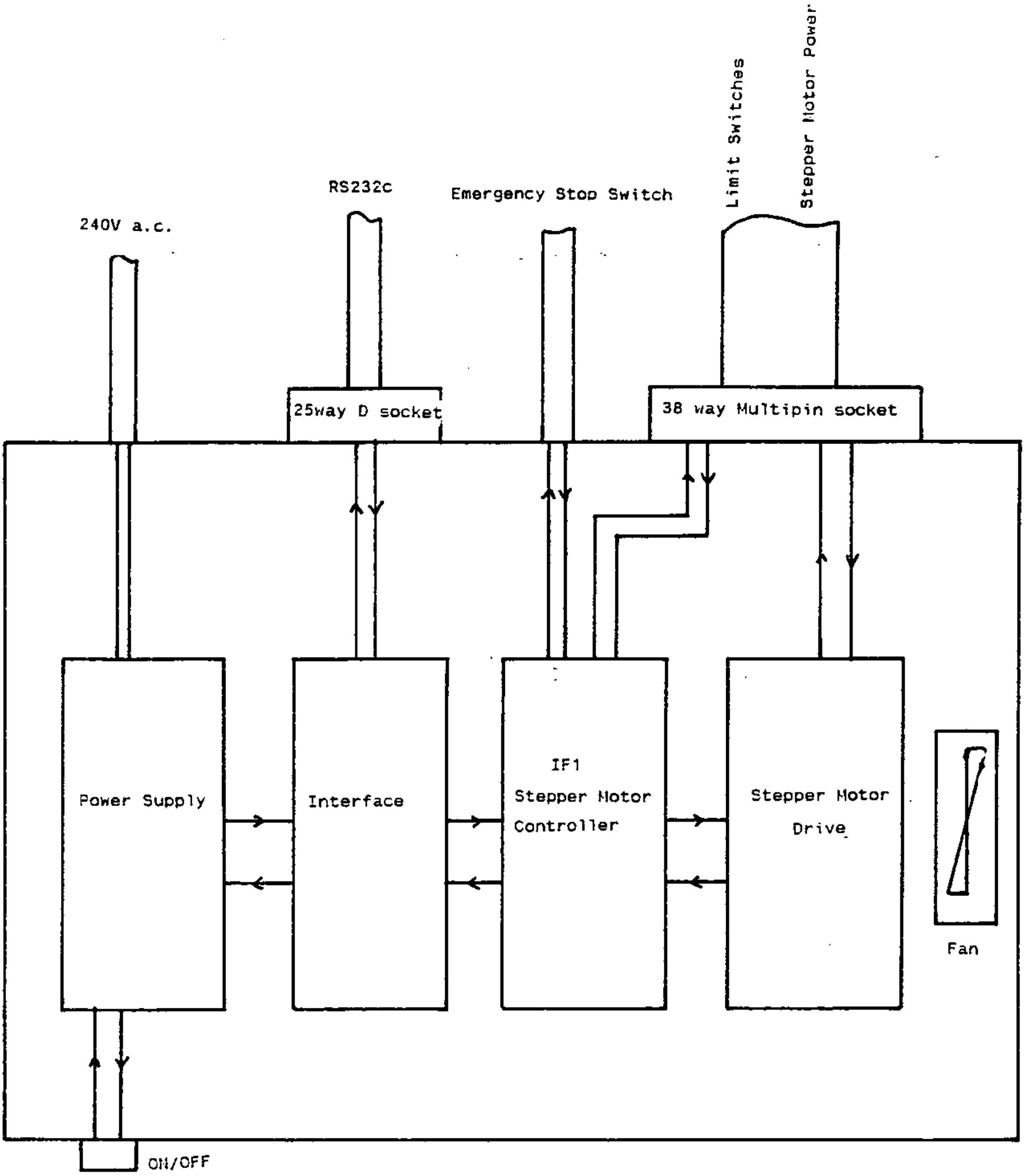


Figure 11. Schematic Layout of Wake Traverse Electrical Controller.

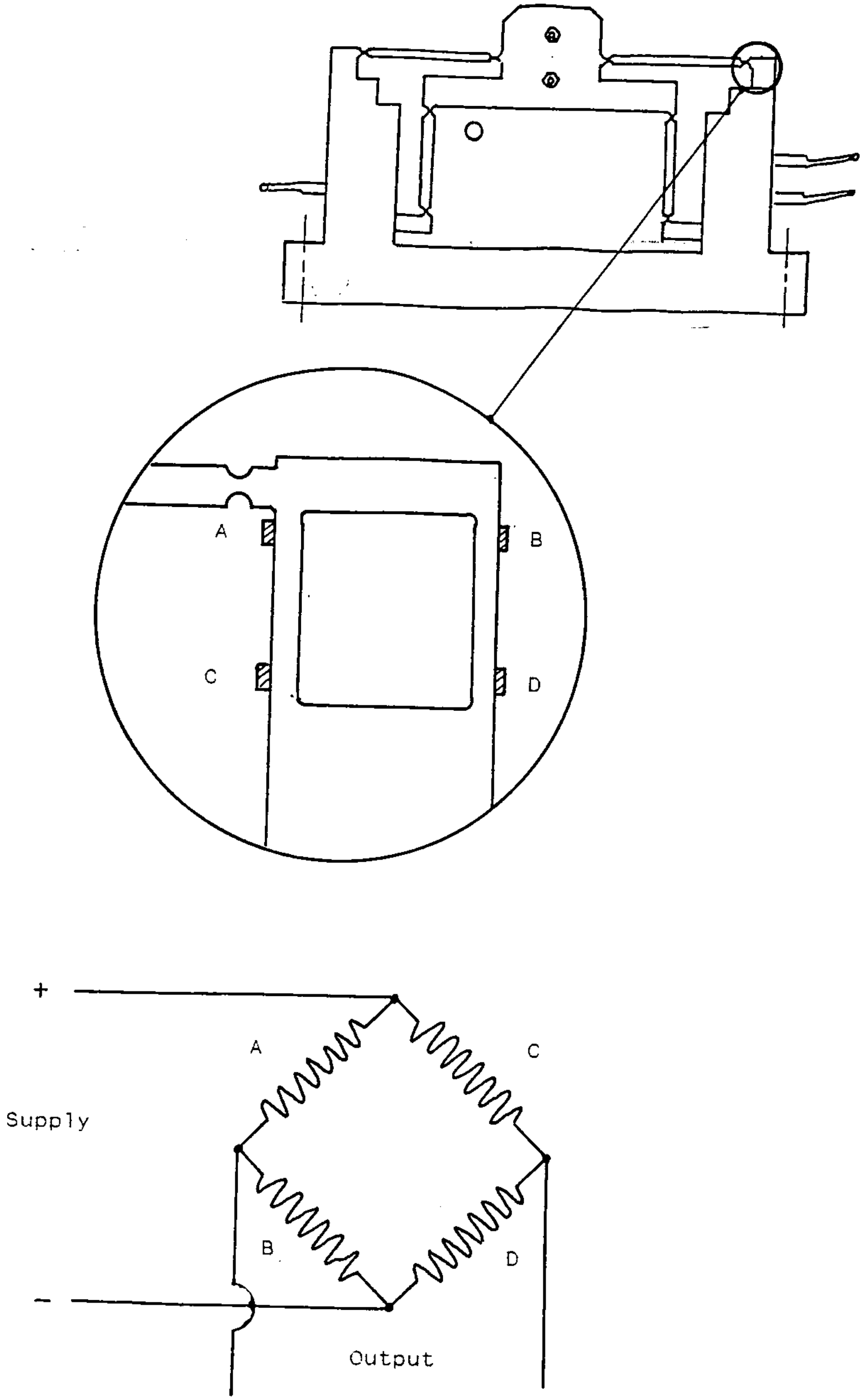


Figure 12. Strain Gauge Bridge Circuit

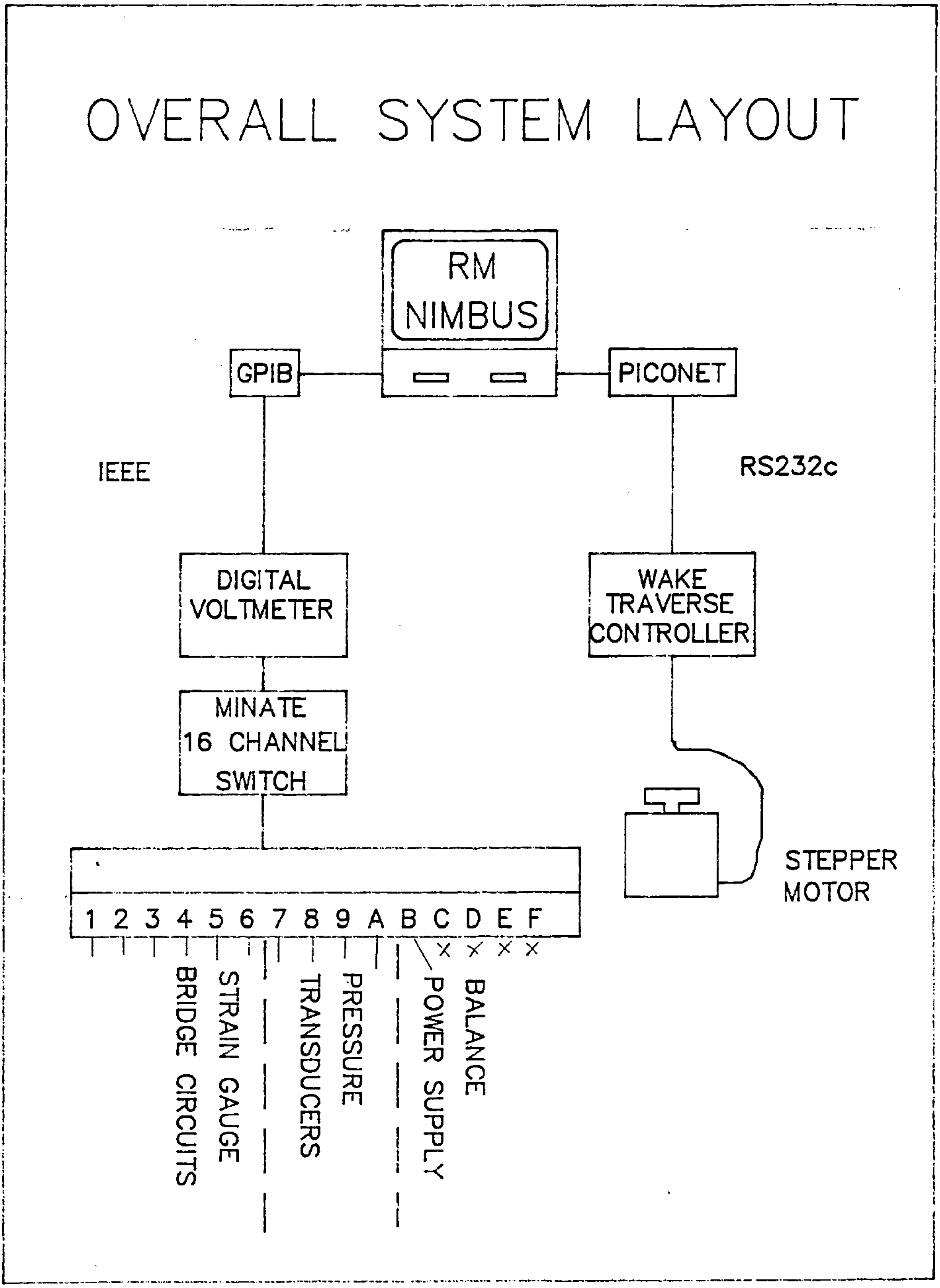


Figure 13. Two-Dimensional Test Rig Data Acquisition System

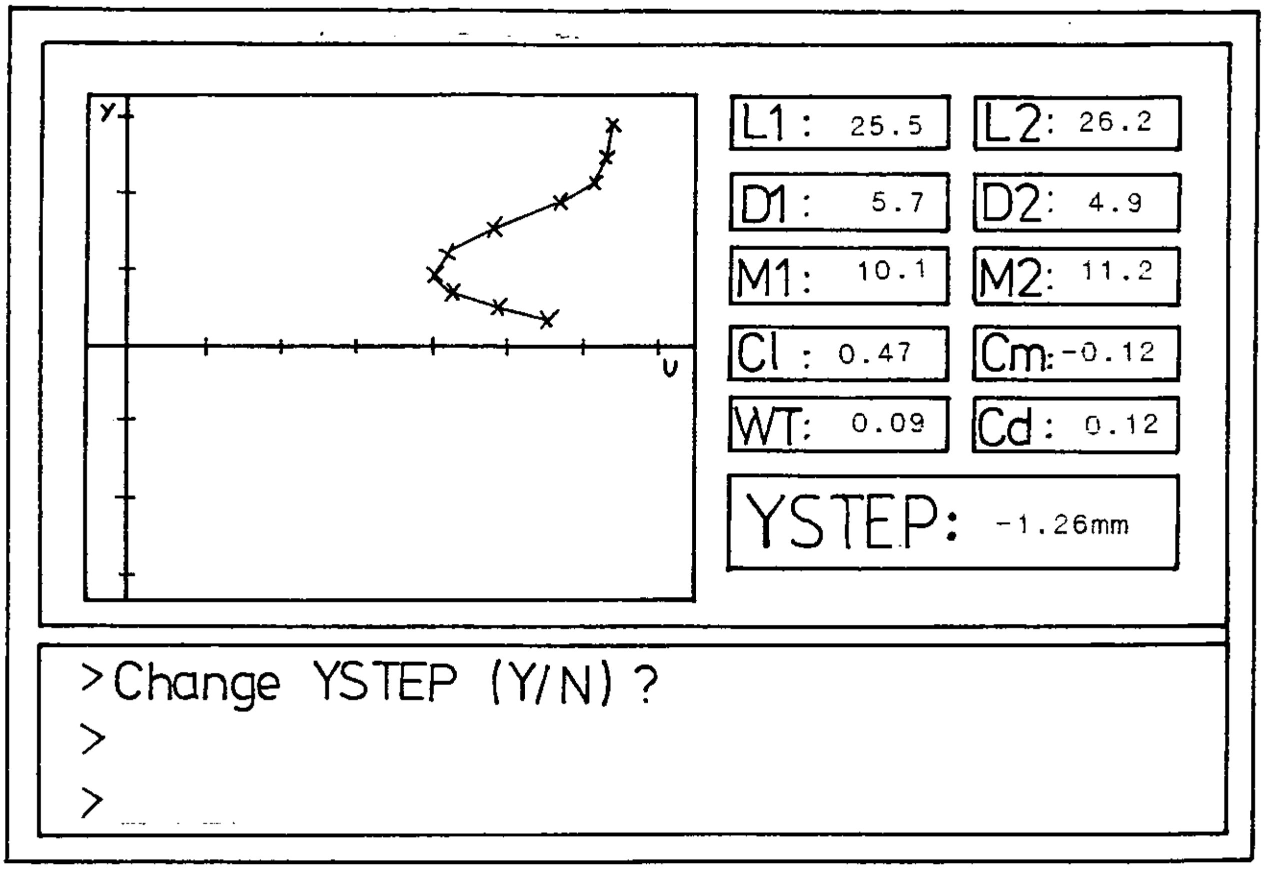


Figure 14. Screen Output of Data Acquisition Software.

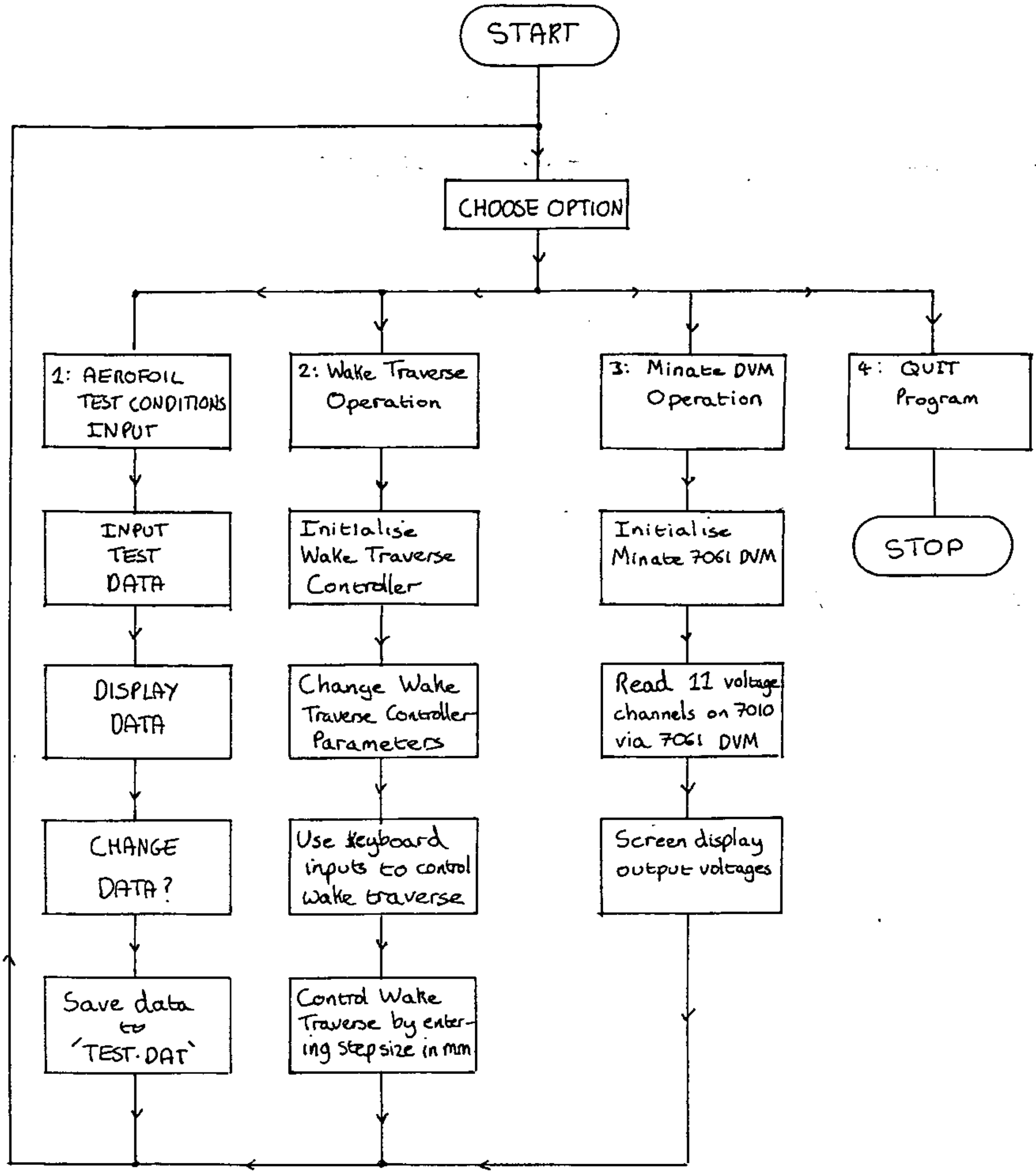


Figure 15. Flow Chart for Data Acquisition software Initial.Pas

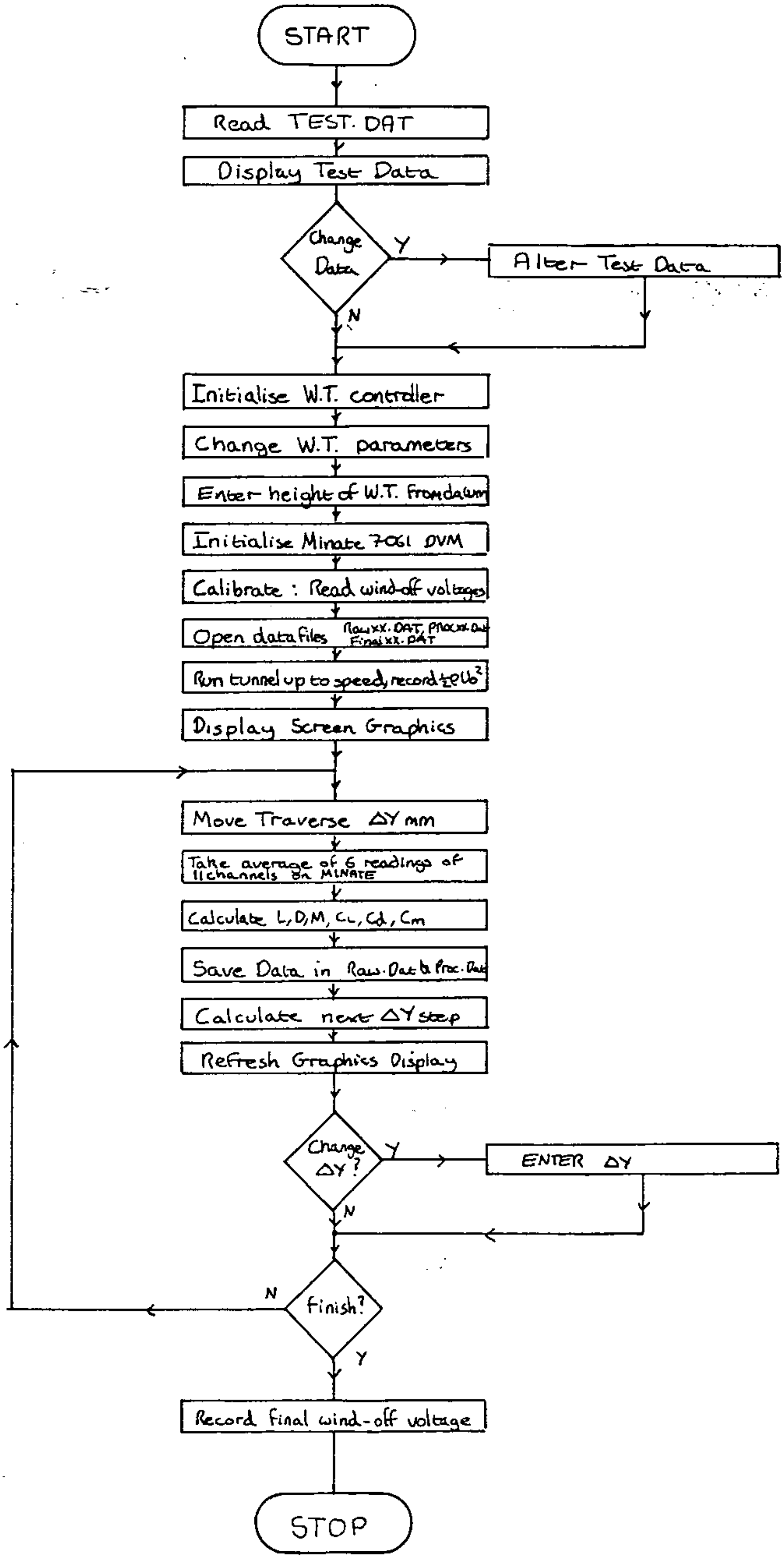


Figure 16. Flow Chart for data Acquisition Software Experiment.Pas.

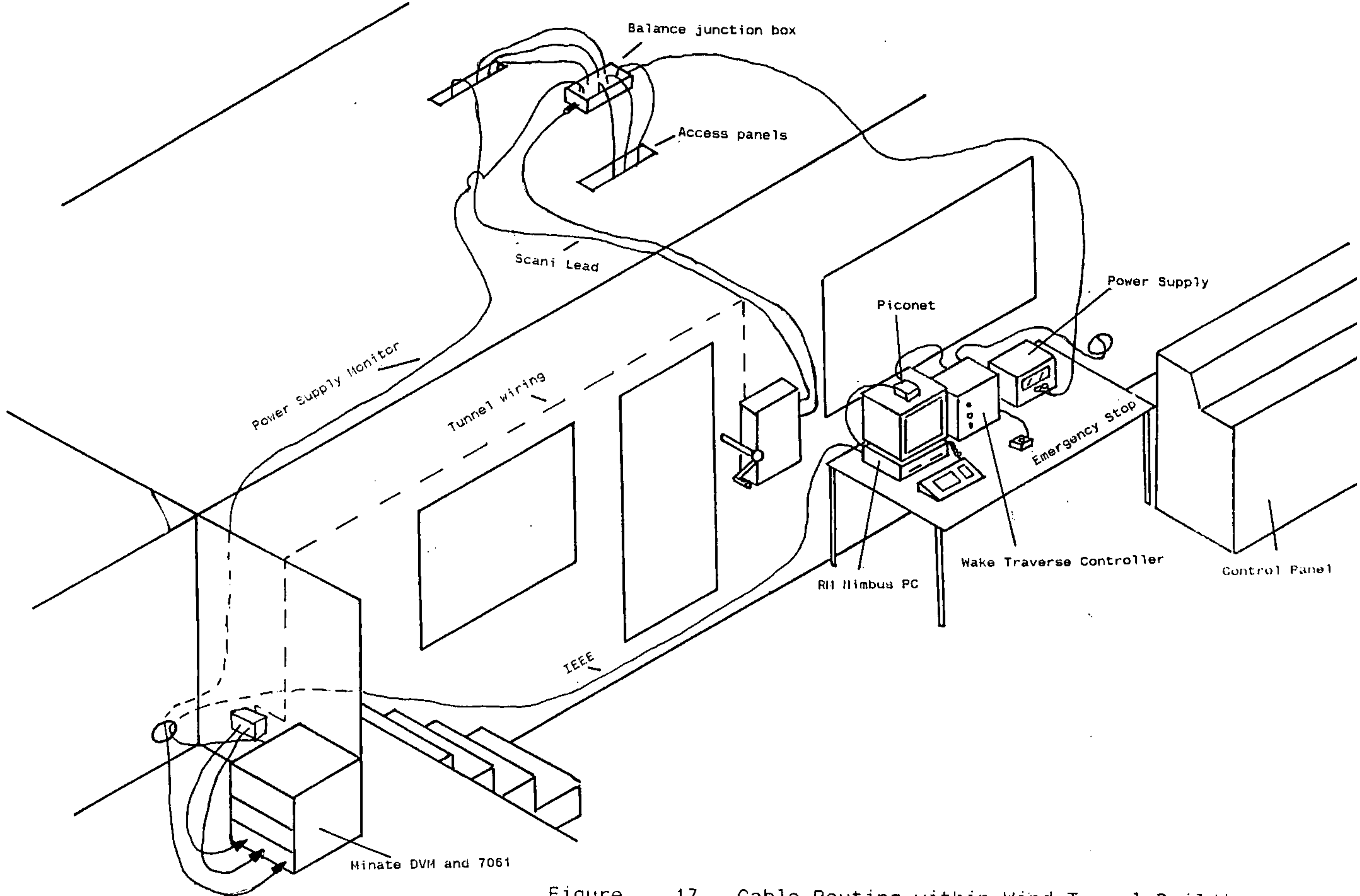
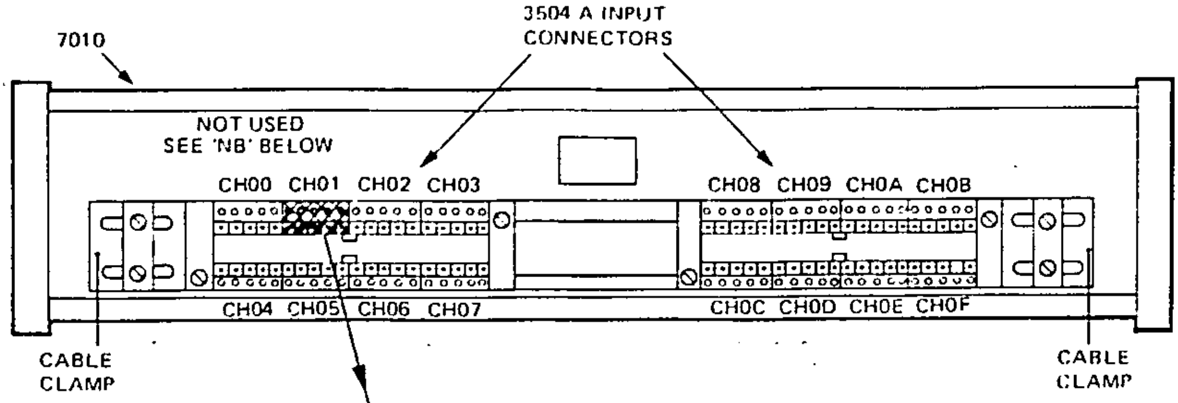
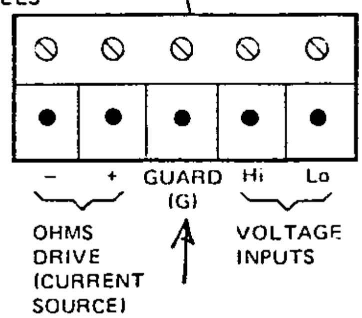


Figure 17. Cable Routing within Wind Tunnel Building.



PIN DESIGNATIONS FOR ALL CHANNELS

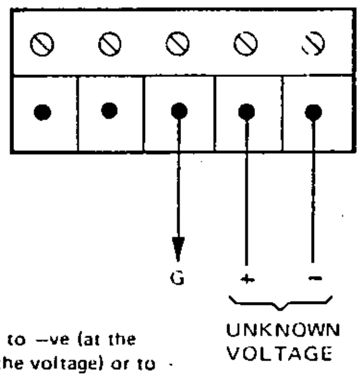
a.



Channel numbering for group 00 to 0F is shown above. Groups 10 to 1F, 20 to 2F, etc. are numbered in a similar way.
NB. Channel 00 is not used (since '00' is used as an 'OFF' command) channels 10, 20, 30, etc. are however used as normal channels.

VOLTAGE MEASUREMENT

b.



Connect G to –ve (at the source of the voltage) or to the source of common mode voltage.

Connection	Force
SG 1	L1
SG 2	D1
SG 3	M1
SG 4	L2
SG 5	D2
SG 6	M2

Channel	Connection
01	Strain Gauge 1
02	" 2
03	" 3
04	" 4
05	" 5
06	" 6
07	Transducer 1
08	" 2
09	" 3
0A	" 4
0B	Supply Voltage

Figure 18. Minate 7010 Wiring Diagram.

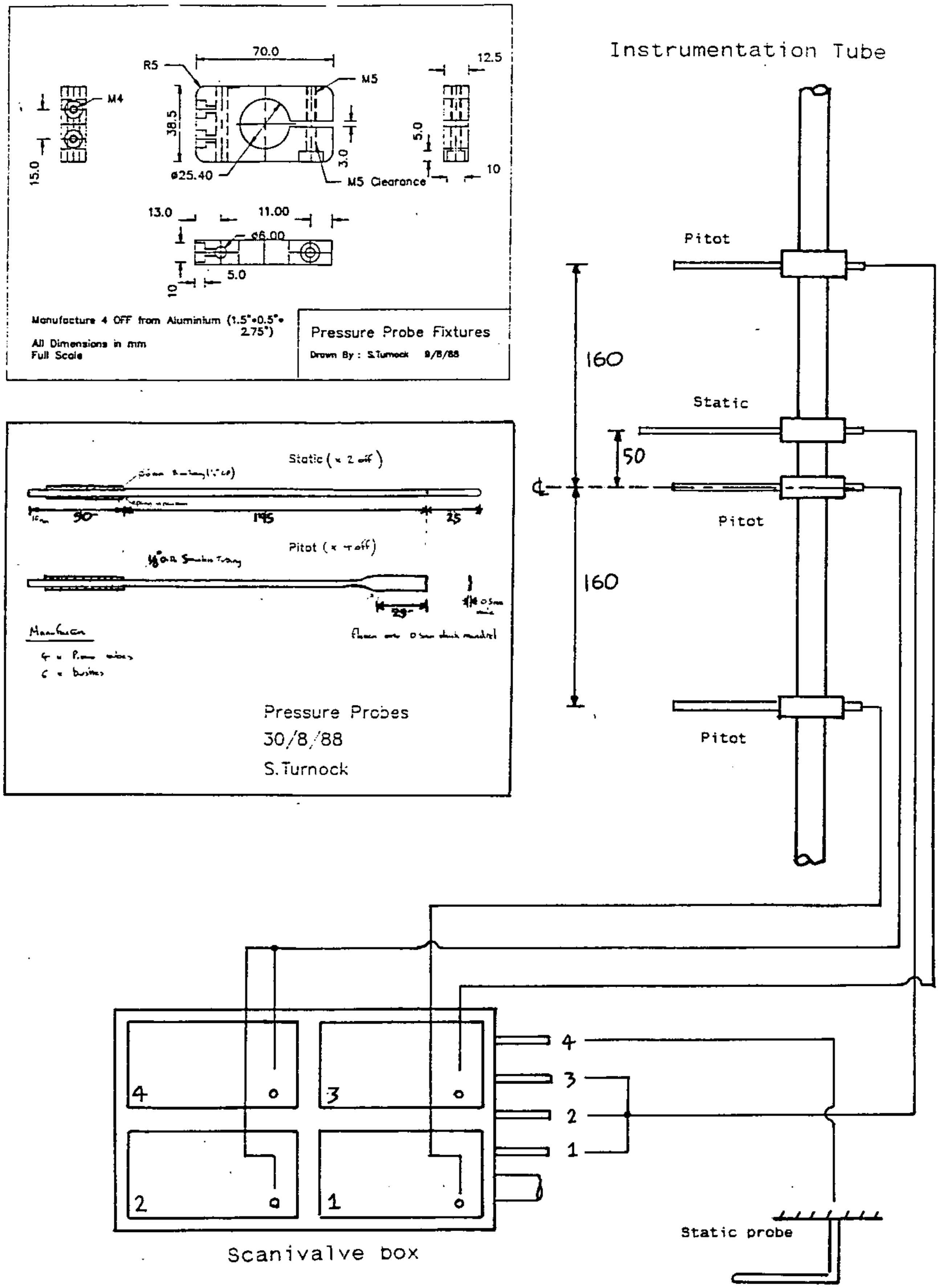


Figure 19. Pressure Probe Tubing for Pressure Transducers.

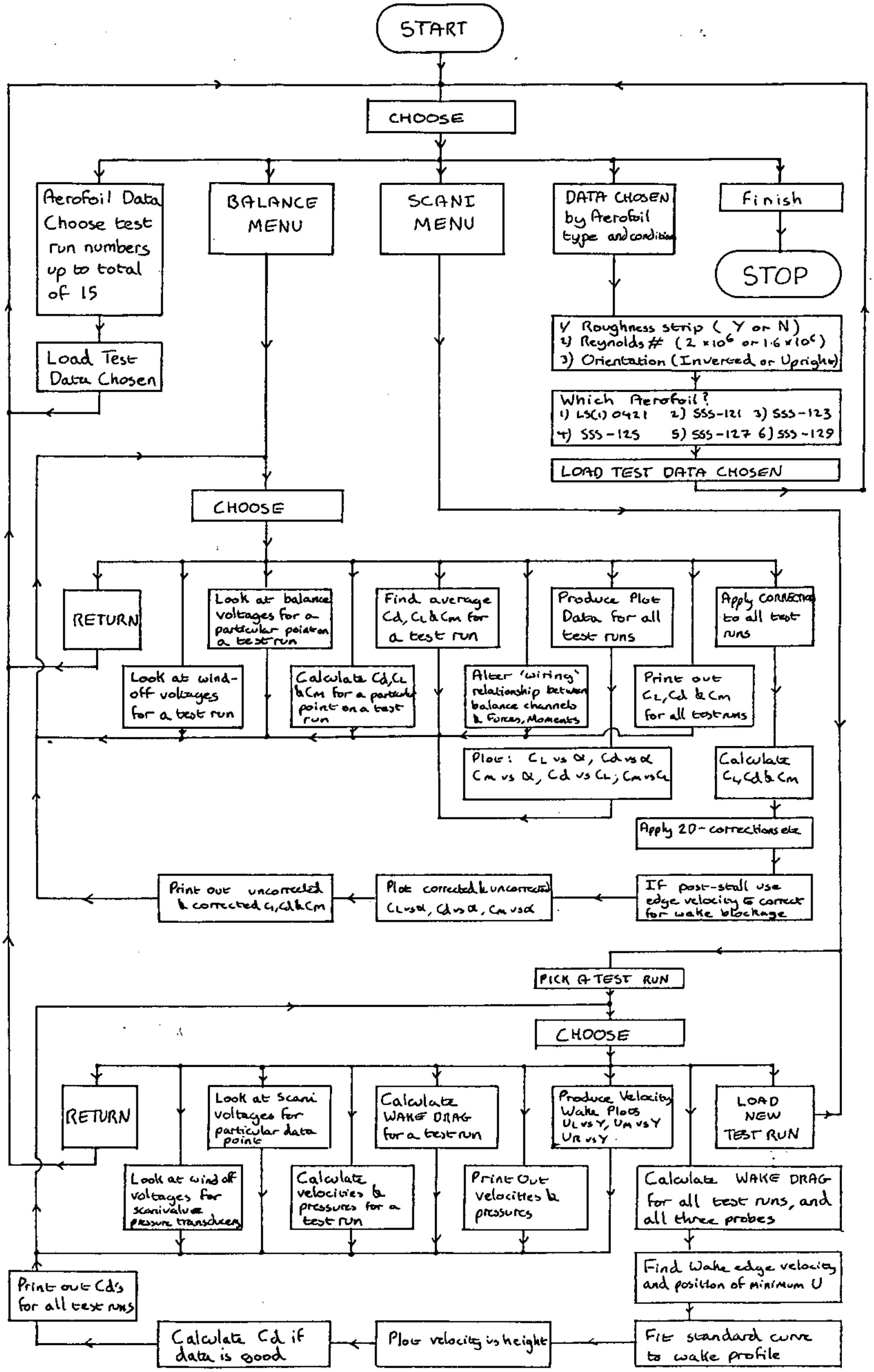


Figure 20. Flow Chart for Data Analysis Software.

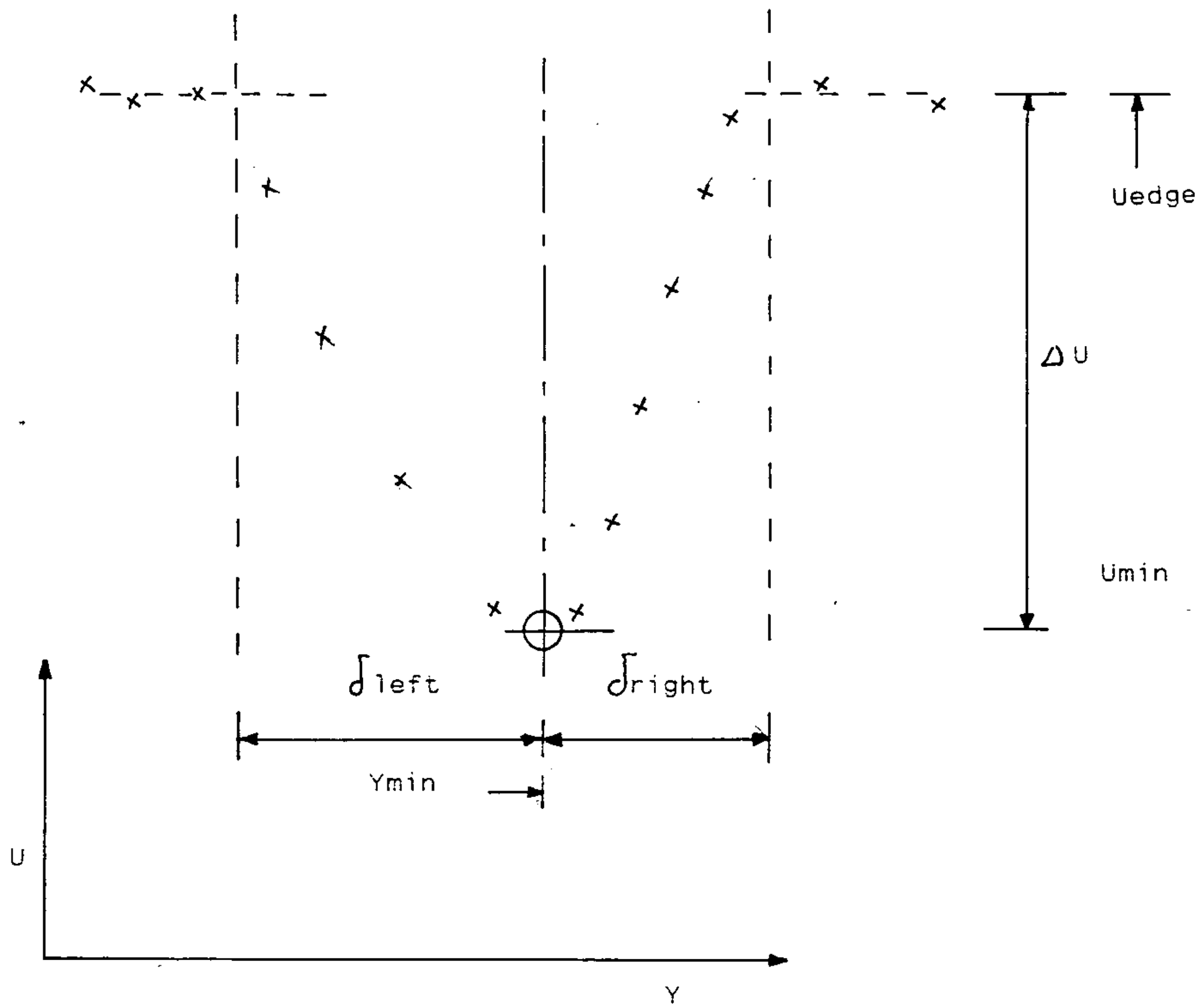


Figure 21. Wake Drag Calculation Nomenclature

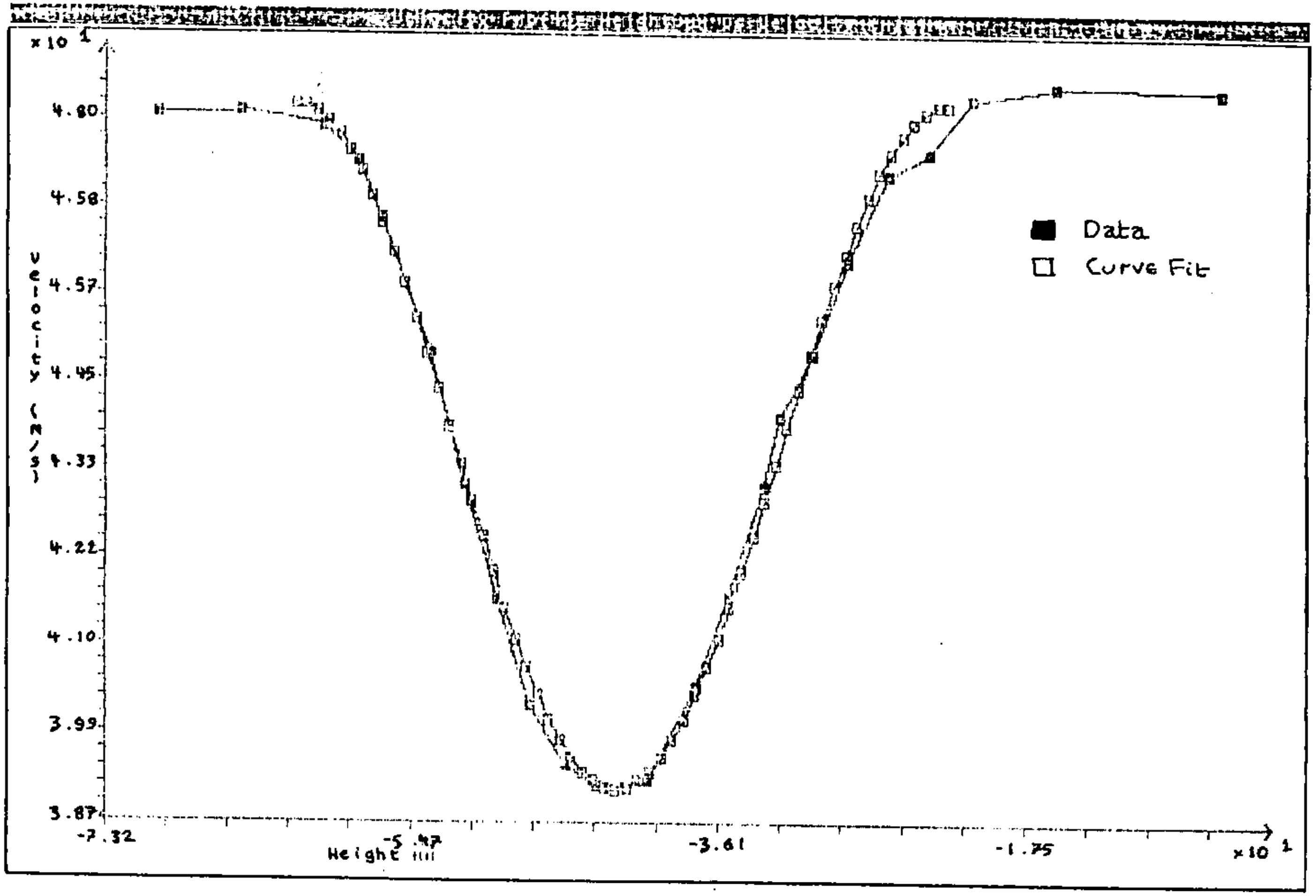


Figure 22. Sample Wake Drag Velocity Profile.

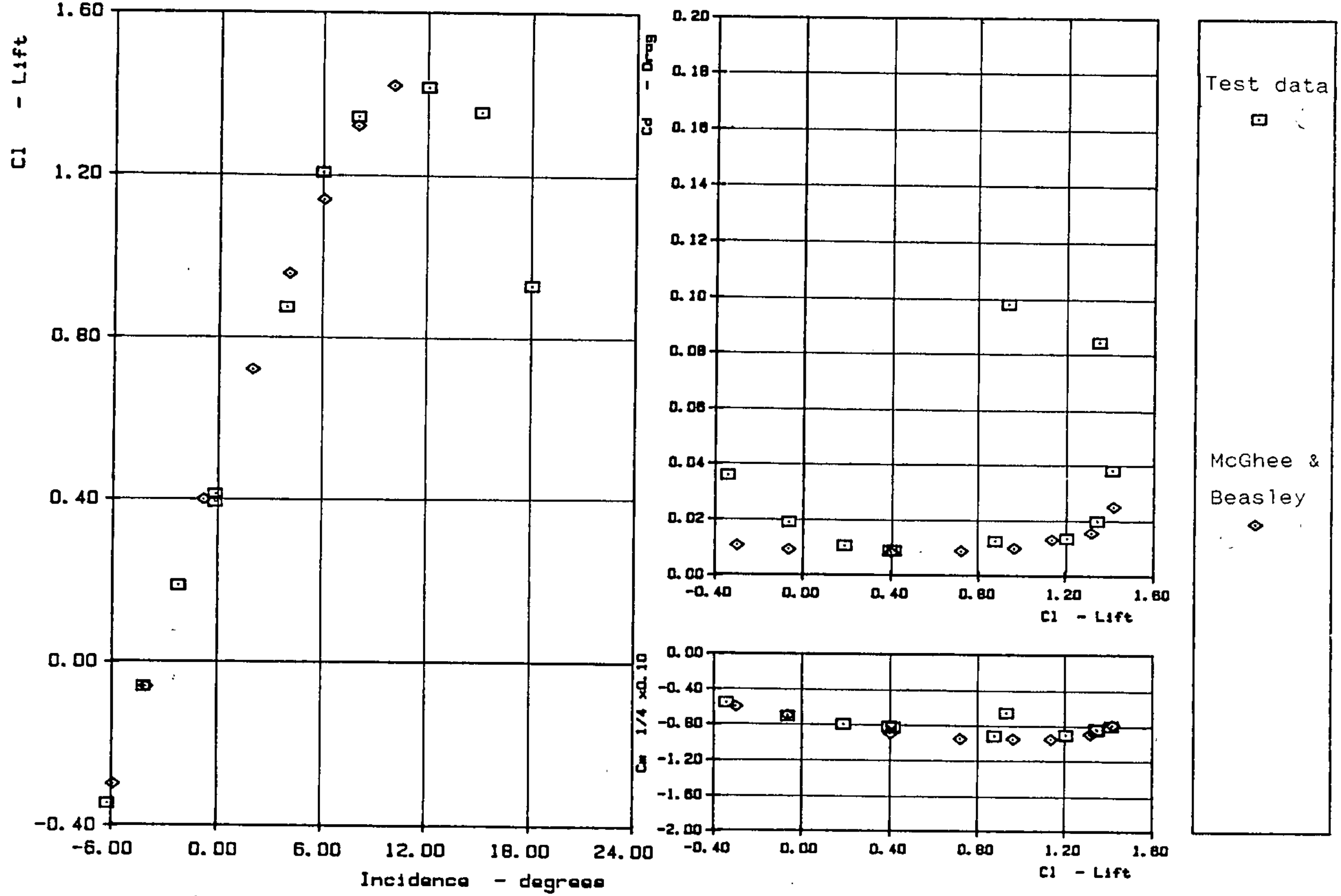


Figure 23. Comparison of measured aerodynamic performance polars and data from McGhee & Beasley[4] for the LS(1) 0421 mod aerofoil at Reynolds number of 2000000.

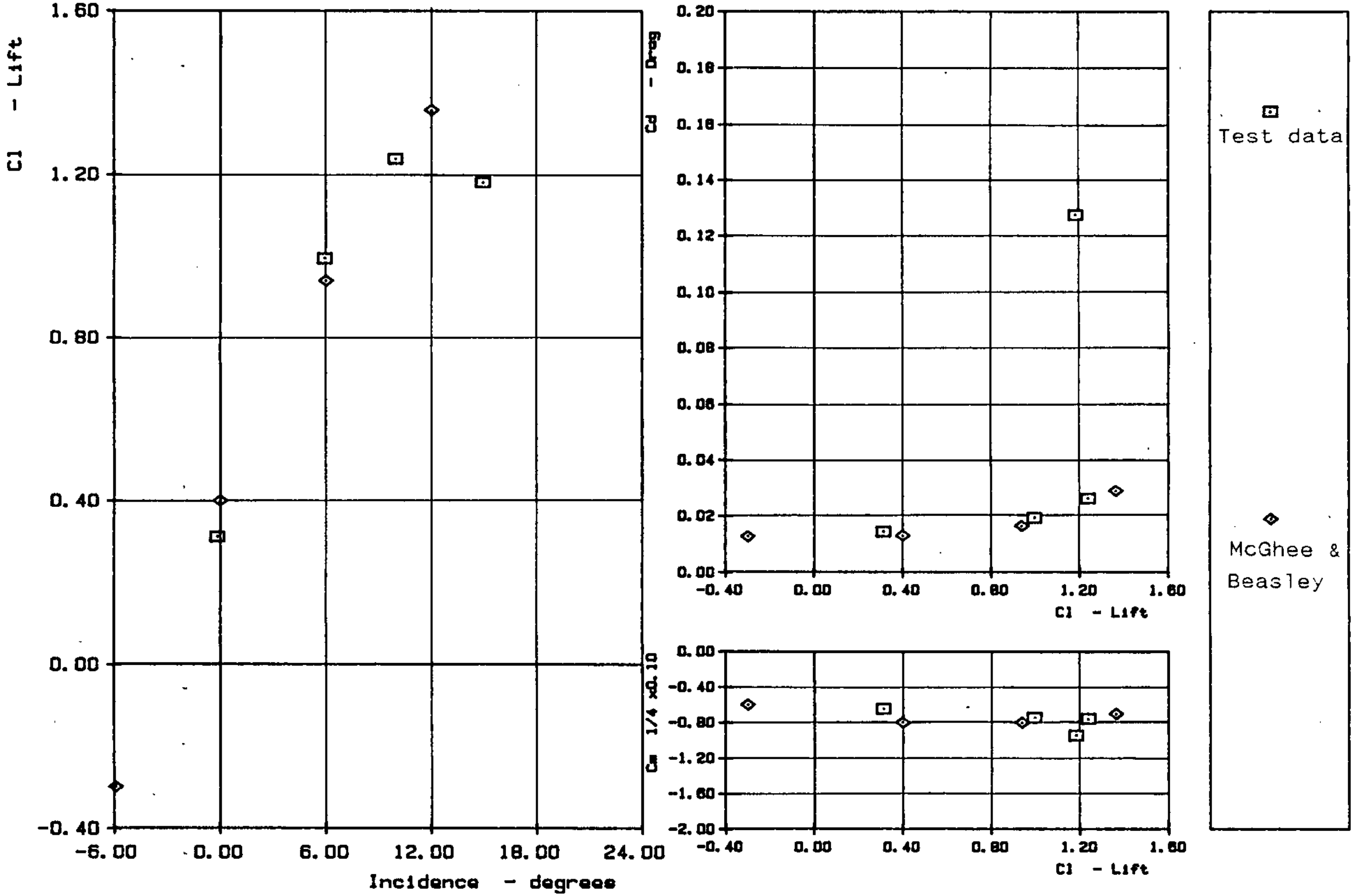


Figure 24. Comparison of measured aerodynamic performance polars and data from McGhee & Beasley[4] for the LS(1) 0421 mod aerofoil at Reynolds number of 2000000 with leading edge roughness.

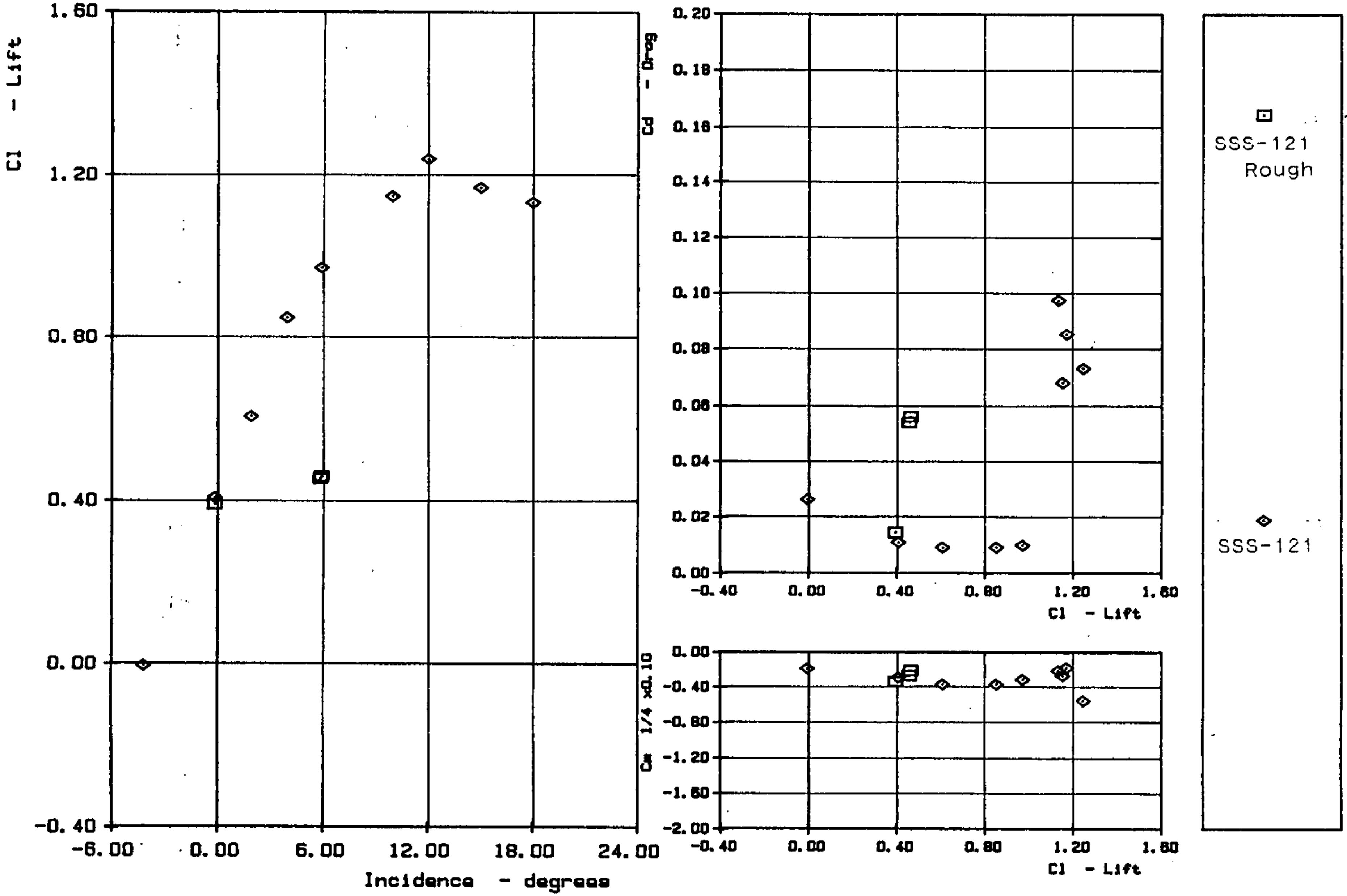


Figure 25. Aerodynamic performance polars for the SSS-121 aerofoil at a Reynolds number of 1600000.

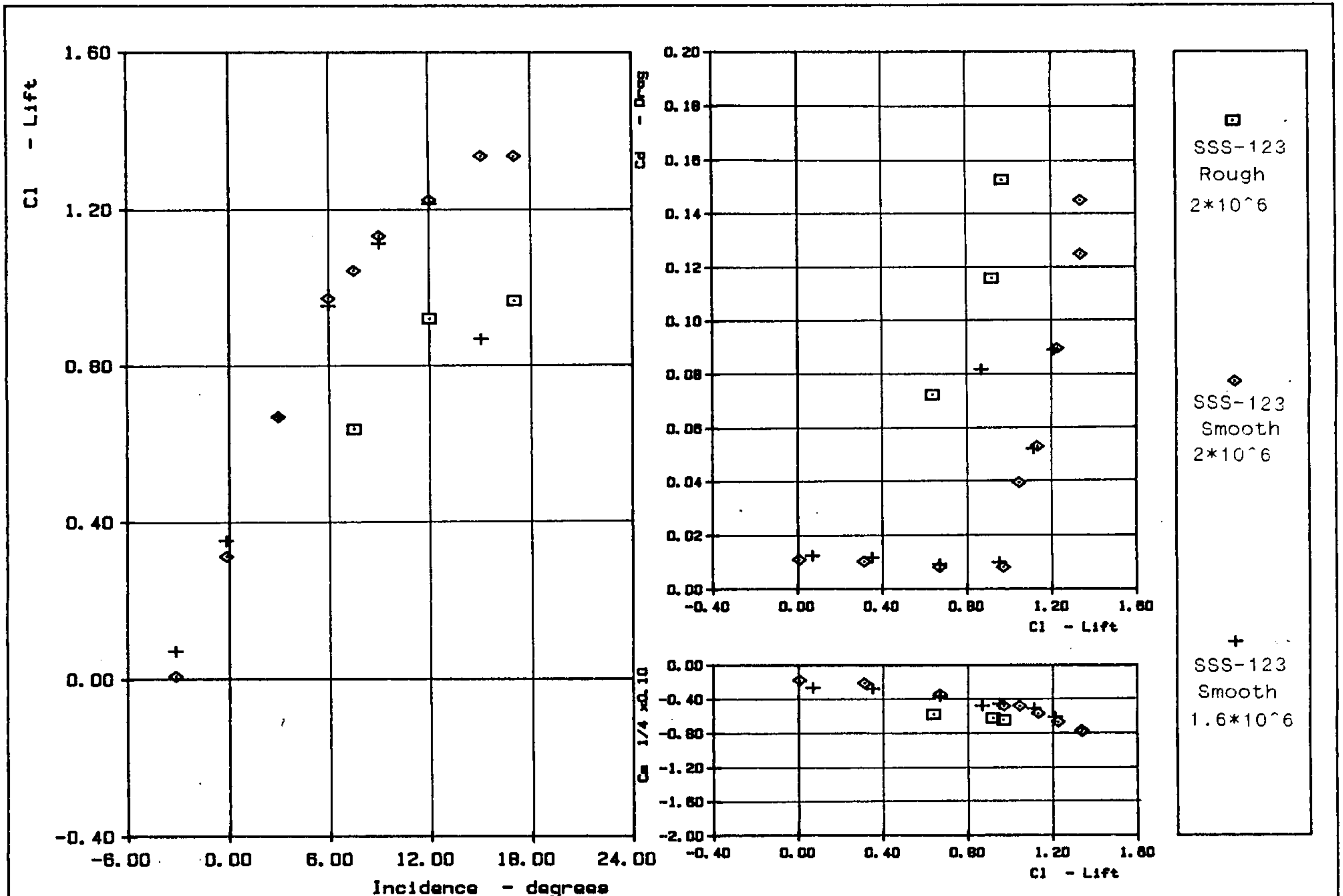


Figure 26. Aerodynamic performance polar for the SSS-123 aerofoil at a Reynolds number of 1600000 and 2000000.

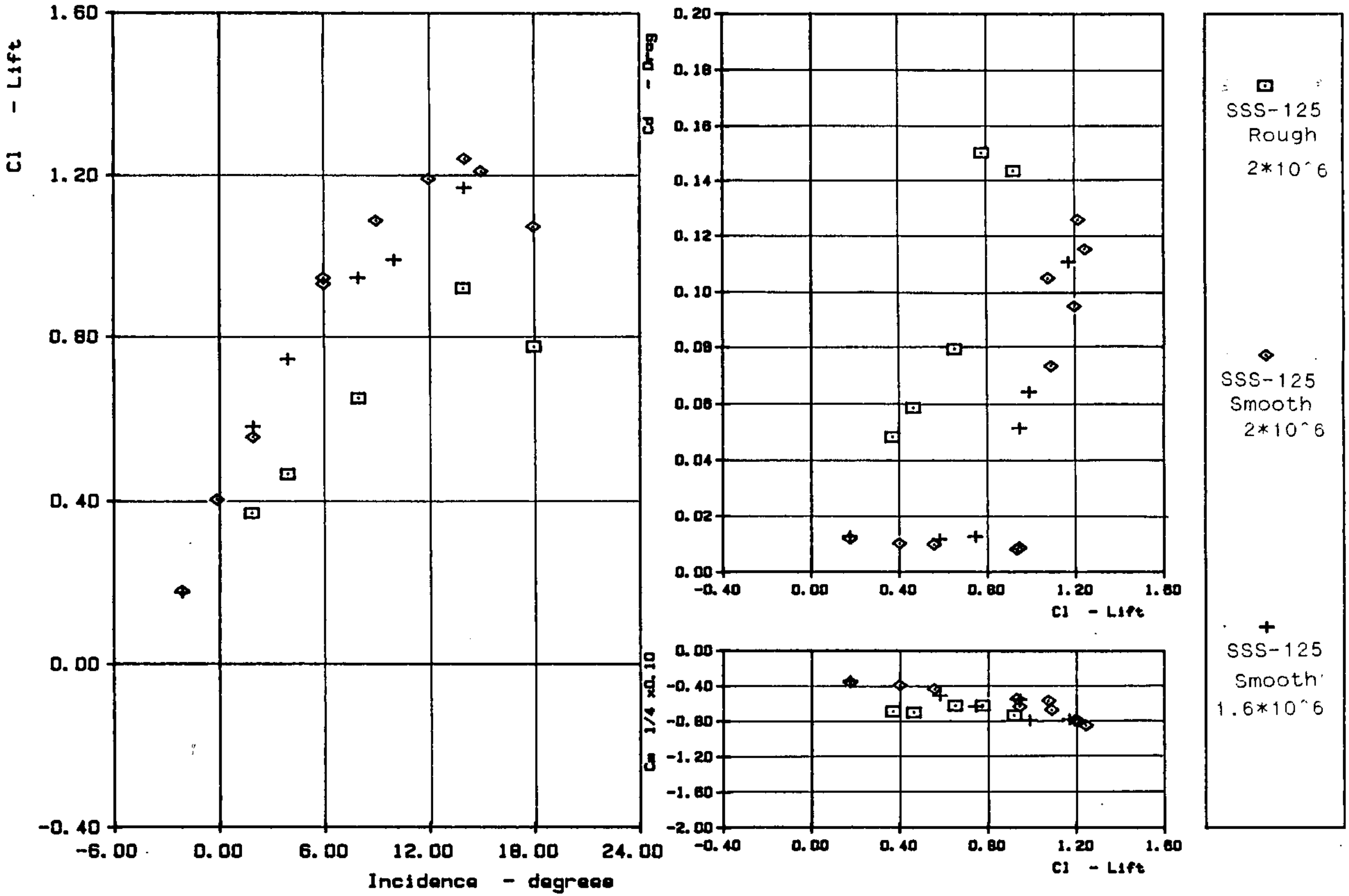


Figure 27. Aerodynamic performance polar for the SSS-125 aerofoil at a Reynolds number of 1600000 and 2000000.

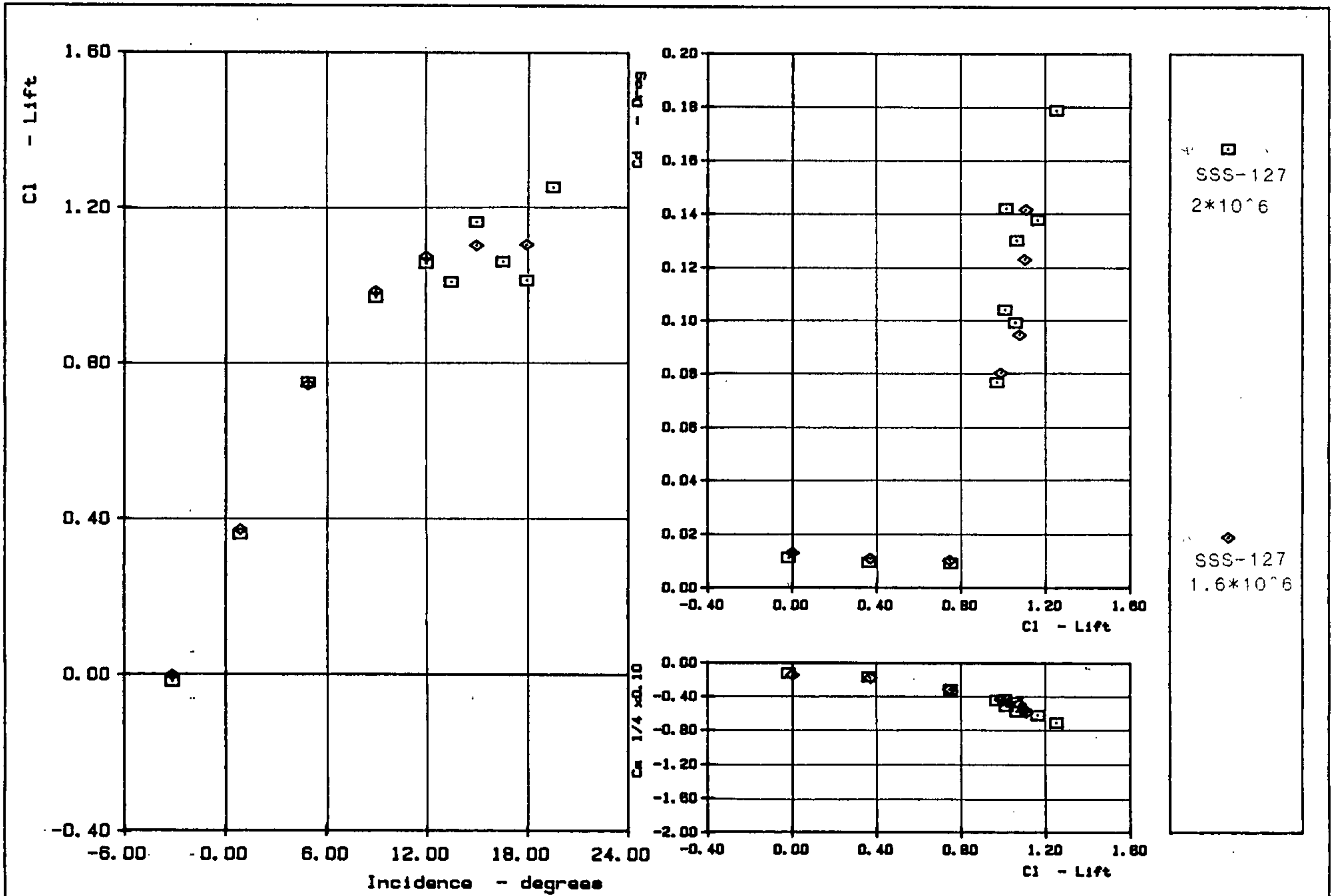


Figure 28. Aerodynamic performance polar for the SSS-127 aerofoil at a Reynolds number of 1600000 and 2000000.

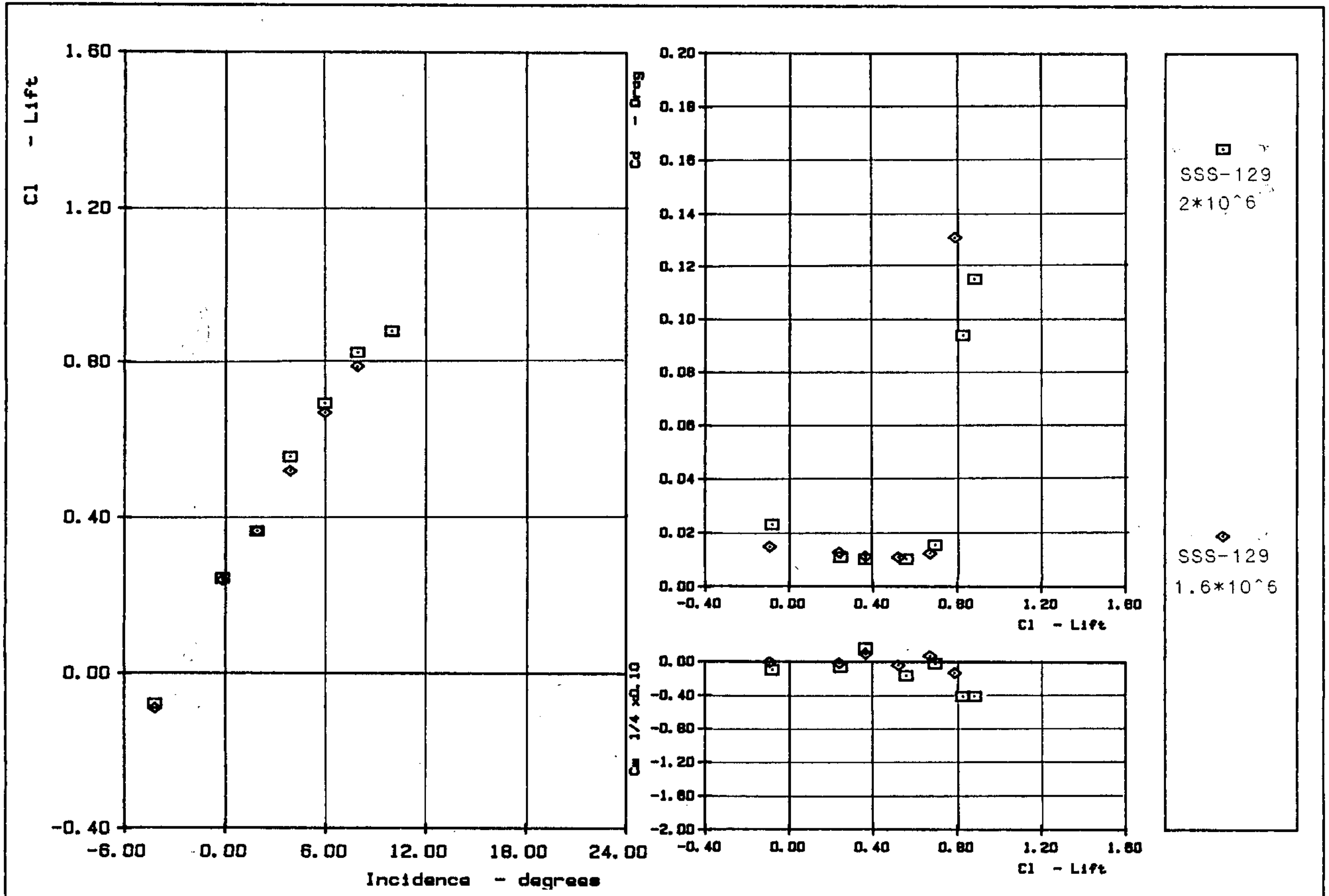


Figure 29. Aerodynamic performance polar for the SSS-129 aerofoil at a Reynolds number of 1600000 and 2000000.

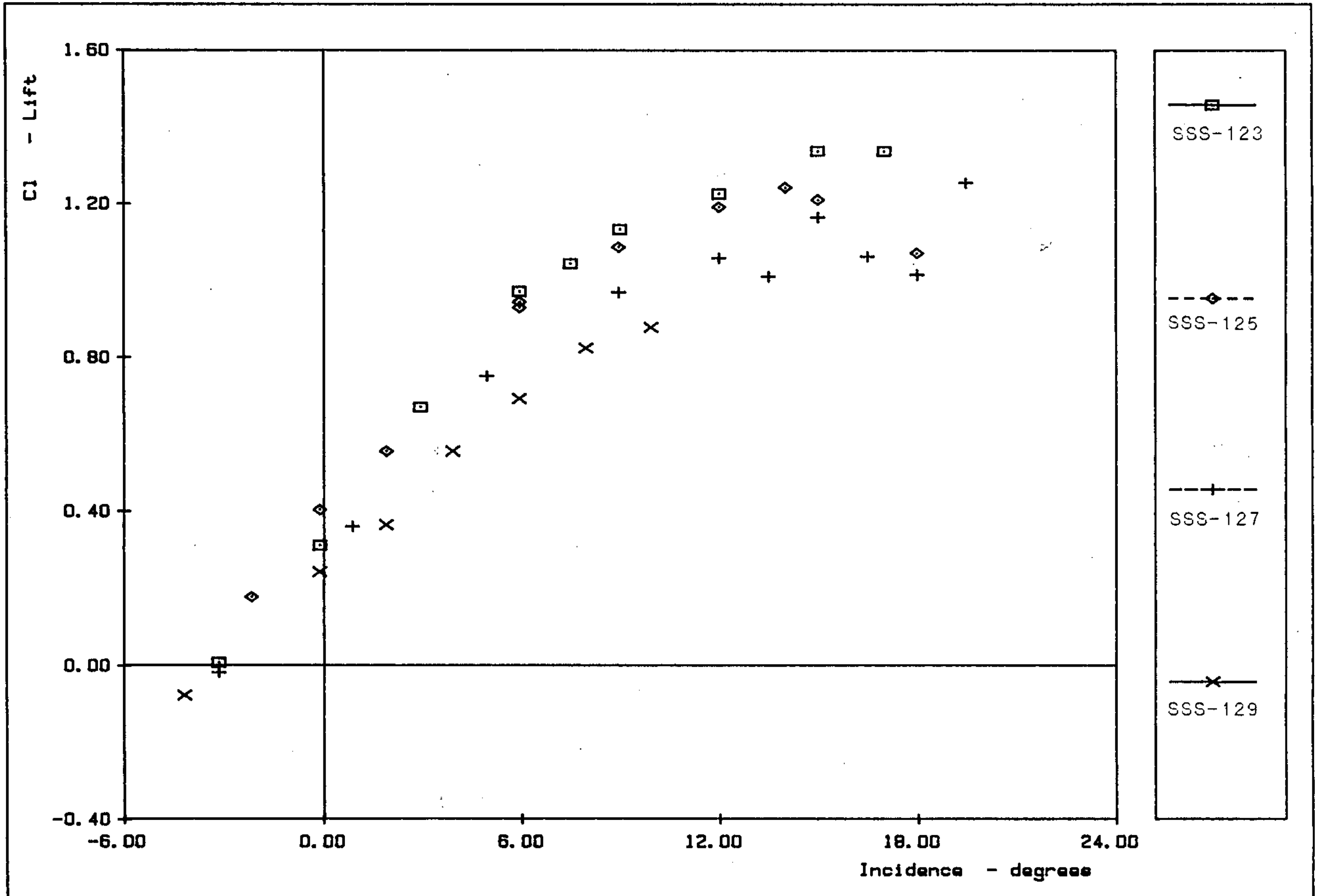


Figure 30. Comparison of Cl for 23,25,27&29% thick aerofoils at a Re Number of 2000000.

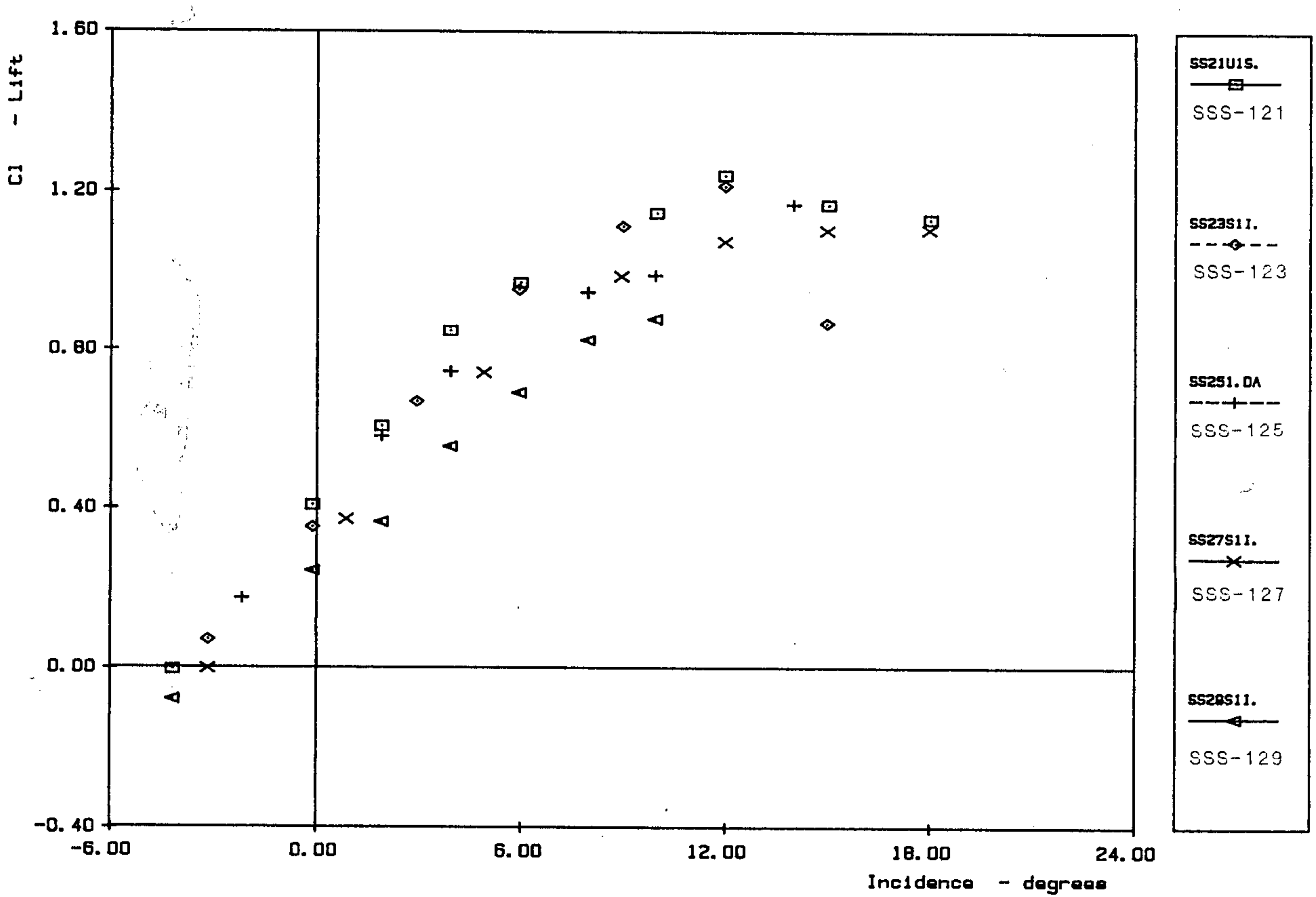


Figure 31. Comparison of Cl for 21,23,25,27&29% thick aerofoils at a Re Number of 1600000.

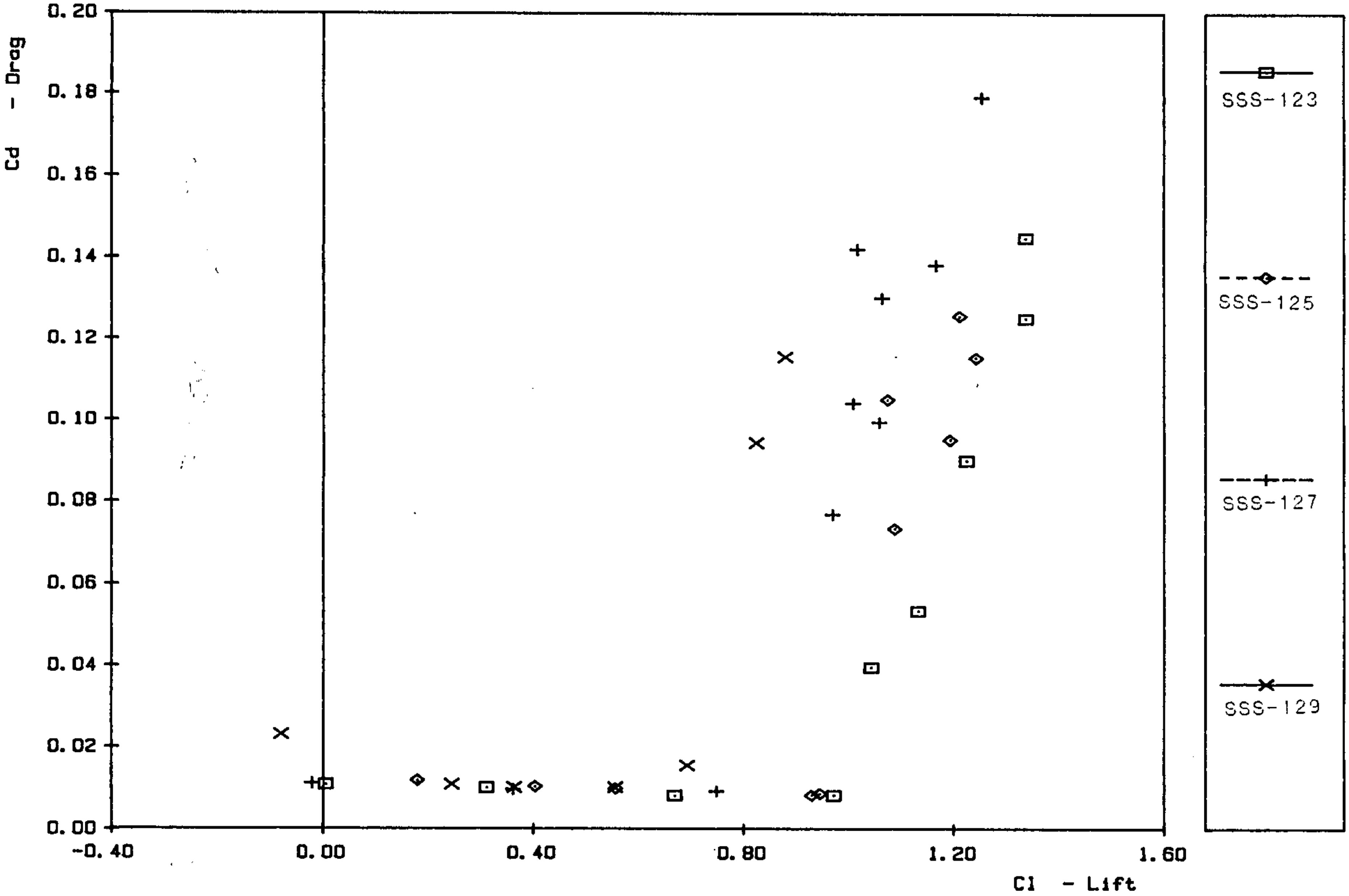


Figure 32. Comparison of Cd for 23,25,27&29% thick aerofoils at a Re Number of 2000000.

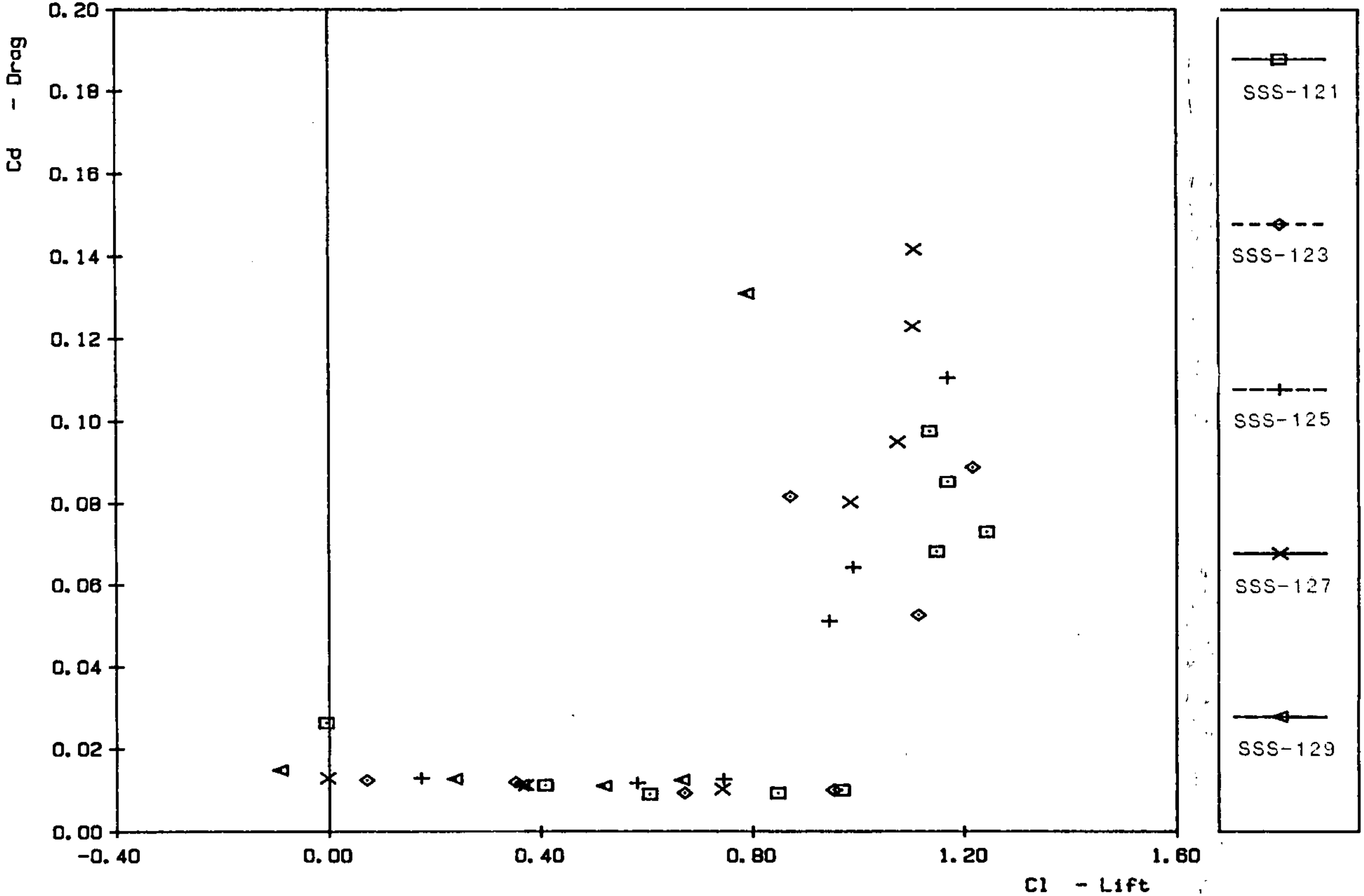


Figure 33. Comparison of Cd for 21,23,25,27&29% thick aerofoils at a Re Number of 16000000.

Table 1. Test Run Conditions

LS(I) 0421 MOD

Inverted /Upright	Re# $\times 10^6$	Rough/ Smooth	Test No
I	2	R	26, 28, 30, 128
I	2	S	15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 126, 127, 129
I	1.6	S	
I	1.6	R	27, 29, 31
U	2	R	
U	2	S	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
U	1.6	R	
U	1.6	S	

SSS-121

Inverted /Upright	Re# $\times 10^6$	Rough/ Smooth	Test No
I	2	R	
I	2	S	130, 131, 132
I	1.6	R	
I	1.6	S	
U	2	R	76, 77, 79
U	2	S	62, 64, 66, 68, 70, 72, 74, 78
U	1.6	R	
U	1.6	S	63, 65, 67, 69, 71, 73, 75

SSS-123

Inverted /Upright	Re# $\times 10^6$	Rough /Smooth	Test No
I	2	R	95, 96, 97
I	2	S	80, 82, 84, 86, 88, 90, 92, 94, 98
I	1.6	R	
I	1.6	S	81, 83, 85, 87, 89, 91, 93
U	2	R	
U	2	S	
U	1.6	R	
U	1.6	S	

SSS-125

Inverted /Upright	Re# $\times 10^6$	Rough /Smooth	Test No
I	2	R	122, 123
I	2	S	118, 119, 120, 121, 125
I	1.6	R	40
I	1.6	S	124
U	2	R	39, 41, 46
U	2	S	32, 33, 34, 37, 44, 47
U	1.6	R	
U	1.6	S	36, 38, 42, 45, 48

SSS-127

Inverted /Upright	Re# $\times 10^6$	Rough /Smooth	Test No
I	2	R	
I	2	S	99, 101, 103, 105, 107, 109, 111, 113, 114, 115, 116
I	1.6	R	
I	1.6	S	100, 102, 104, 106, 108, 110, 112
U	2	R	
U	2	S	
U	1.6	R	
U	1.6	S	

SSS-129

Inverted /Upright	Re# $\times 10^6$	Rough /Smooth	Test No
I	2	R	
I	2	S	
I	1.6	R	
I	1.6	S	
U	2	R	
U	2	S	49, 50, 52, 55, 56, 58, 60
U	1.6	R	
U	1.6	S	51, 53, 54, 57, 59, 61