```
% This is called internally by make_ops.m
%
function [D]=d_oper(n,etahat,muhat);
k=n+4;
etahat=etahat(1:k,1:k);
muhat=muhat(1:k,1:k);
D=etahat*(etahat*muhat-eye(k,k))*muhat^2;
D=D(1:n,1:n);
```

D.5.6 makeus.m

```
%
% makeus constructs the polynomial shift matrices for U and U.L
%
% [poly_U,poly_LU]=makeus(n,U,L1);
%
% The results of this function are passed as arguments to sexl.m
%
function [poly_U,poly_LU]=makeus(n,U,L1);
k=n;
[x,y]=size(U);
U(k)=0;
poly_U=polyr(k,U);
poly_LU=polyr(k,U*L1);
```

D.5.7 bc.m

```
%
% function [bc_a]=bc(n,etahat)
%
function [a]=bc(n,etahat)
a=zeros(n,n);
etahat=etahat(1:n,1:n);
v1=chebvec(1,n);
v0=chebvec(0,n);
b1=v0;                                  % phi(0)=0
b2=v1;                                  % phi(1)=0
b3=(etahat)*v0;                         % D phi(0)=0
b4=(etahat)*v1;                         % D phi(1)=0
b5=(etahat^2)*v0;
a(:,1)=b2;
a(:,2)=b4;
```

D.5.3 l1_oper.m

```
%
% L1_OPER calculates the operator L=D^2-1/r.D for the Sexl
% equation, in the transformed domain.
%
% This function is called internally by make_ops.m
%
% [L1]=l1_oper(n,etahat,muhat);
%
function [L1]=l1_oper(n,etahat,muhat);
k=n+4;
etahat=etahat(1:k,1:k);
muhat=muhat(1:k,1:k);
L1=etahat*(etahat*muhat+3*eye(k,k))...
  *muhat^2;
Li=L1(1:n,1:n);
```

D.5.4 l2_oper.m

```
%
% L2_OPER calculates the operator L^2=(D^2-1/r.D)^2 for the Sexl
% equation, in the transformed domain.
%
% This function is called internally by make_ops.m
%
% [L2]=l2_oper(n,etahat,muhat);
%
function [L2]=l2_oper(n,etahat,muhat);
k=n+4;
etahat=etahat(1:k,1:k);
muhat=muhat(1:k,1:k);

L2=etahat*( etahat^3*muhat^3 ...
  +6*etahat^2*muhat^2 ...
  +3*etahat*muhat ...
  -3*eye(k,k));
L2=L2(1:n,1:n);
```

D.5.5 d_oper.m

```
%
% [D]=d_oper(n,etahat,muhat);
%
```

```
B=B*mu(n)^2;      % shift right 2 places

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% preform eigenvalue calculation       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[AA,BB,Q,Z,V]=qz(A,B);
AA=diag(AA);
BB=diag(BB);
I=find(BB~=0);
AA=AA(I);
BB=BB(I);
V=V(I,:);
C=AA./BB;
[Y,I]=sort(imag(C));
[x,y]=size(I);
I=I(1:x-1,:);
eigval=C(I);
eigvec=V(:,I);
```

D.5.2 make_ops.m

```
%
% make_ops constructs the operators L, L^2, mu,eta,mu^3,p
%
% [L1,L2,muhat,etahat,M,D]=make_ops(n);
%
% These operators are passed as arguments to sexl.m
%
function [L1,L2,muhat,etahat,M,D]=make_ops(n);
k=n+20;
muhat=makemuhat(k);
etahat=makeetahat(k);
M=muhat^3;
L1=l1_oper(n,etahat,muhat);
L2=l2_oper(n,etahat,muhat);
D=d_oper(n,etahat,muhat);
L1=L1(1:n,1:n);
L2=L2(1:n,1:n);
D=D(1:n,1:n);
M=M(1:n,1:n);
```

```
  rvur=poly2d(V,rur,C,Cinv);
end;
N=ruux(:,1:n)*Iplusint+rvur(:,1:n);
```

D.5 The Sexl stability equation

These are the Matlab functions and scripts used to calculate the stability properties of pipe velocity profiles. The main function is `sexl.m`, which evaluates the stability of a supplied velocity profile. Before calling this function the functions `make_ops.m` and `makeus.m` must be invoked to supply the correct arguments to `sexl.m`.

Two higher levels of function which use `sexl.m` are supplied, namely `findre.m` and `getnose.m`: The former finds, for a particular value of the wavenumber α, that Re corresponding to the critical value, namely that for $c_i = 0$. The latter uses the former, and implements a simple search technique to determine the value of α and Re corresponding to the minimum critical Reynolds number (if any) for a particular velocity profile. Both of these two functions must be supplied with parameters generated by `make_ops.m` and `makeus.m`. The function `makeus.m` must in turn be supplied with a velocity profile, given as a row vector of Chebyshev coefficients.

D.5.1 sexl.m

```
%
% sexl.m calculates the eigenvalues for the Sexl problem
% for a given alpha and Re
%
% [eigval,eigvec]=sexl(n,alpha,re,omega,L1,L2,M,poly_U,...
%                      poly_LU,bc_s);
function [eigval,eigvec]=sexl(n,alp,re,c,L1,L2,M,poly_U,..
  poly_LU,bc_s)
im_A=i/(alp*re).*(L2-(2*alp^2+c).*L1+(alp^4).*M3);
re_B=(L1-(alp^2).*M3);
re_A=re_B*poly_U-poly_LU;
A=re_A+i.*im_A;
B=re_B;
clear re_A re_B im_A

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% insert boundary conditions                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

A=A*mu(n)^2+bc_s; % boundary cond's
```

```
m1=makemuhat(n+2);
em1=e1*m1;
c1=collder(m);
% c1=diag(ones(n-1,1),1)-diag(ones(n-1,1),-1);
% c1=c1/(2*(n-1));
Iint=eye(n+2,n+2)-4 .*intmat(n+2)*vec*m1;
e1=e1(1:n,1:n);
e2=e2(1:n,1:n);
m1=m1(1:n,1:n);
em1=em1(1:n,1:n);
Iint=Iint(1:n,1:n);
```

D.4.7 nonlini.m

```
%
% nonlini calculates the values of the non-linear terms
% for the integral formulation.
%
% [N]=nonlini(n,U,V,eta1,mu01,etaimu1,col1,Iplusint{,C,Cinv});
%
% n is the order, and U,V are the matrices of axial, radial
% velocities respectively. eta1, etaimu1, col1, Iplusint
% are the constant matrices generated by the function
% makematint.m
%
function [N]=nonlini(n,U,V,eta1,mu01,etaimu1,col1,Iplusint,...
 C,Cinv);

%
% calculate the term r.u.u_x...
%
ru=U*mu01;
u_x=col1*U;
if (nargin==8),
  ruux=poly2d(ru,u_x);
else,
  ruux=poly2d(ru,u_x,C,Cinv);
end;
%
% Now the term r.v.u_r...
%
rur=U*etaimu1;
if (nargin==8),
  rvur=poly2d(V,rur);
else,
```

D.4.4 kopint.m

```
%
% kopint determines the constant matrix
% K=omega.r
% for the RHS of the equation
%
% [K]=kopint(m,n,omega);
%
function [K]=kopint(m,n,omega);
mu01=makemuhat(n);
K=zeros(m,n);
K(:,1)=ones(m,1)*omega/2;
K(:,2)=ones(m,1)*omega/2;
```

D.4.5 linopint.m

```
%
% linopint calculates the linear operator:
% -omega.r-S+D+rD^2
%
% function [L]=linopint(n,omega);
%
function [L]=linopint(n,omega);
mu01=makemuhat(n);
eta1=makeetahat(n);
R_0=zeros(1,n);
R_0(1,1)=2;
S=ones(n,1)*R_0;
L= -omega .*mu01 -eta1*S*mu01 + eta1 + eta1^2*mu01;
```

D.4.6 makemati.m

```
%
% makemati generates the constant matrices eta1, eta2, eta1mu1,
% col1, and Iplusint for use by the script.
%
% [eta1,eta2,mu01,eta1mu1,col1,Iplusint]=makemati(n,m);
%
function [e1,e2,m1,em1,c1,Iint]=makemati(n,m);
vec=zeros(1,n+2);
vec(1)=1;
e1=makeetahat(n+2);
e2=e1^2;
```

```
disp(max(max(abs(diff(half+1:m,:)))));
ur=coef2pts(u,C,Cinv);
dr=coef2pts(diff,C,Cinv);
ur=ur(:,1:6:3*n/2);
dr=dr(:,1:6:3*n/2);
surf(Yplot,Xplot,dr,ur);
view([50,30]);
drawnow;
tstart=cputime;
  end;
end;
```

D.4.2 contint.m

```
%
% contint returns the radial velocity, V, when passed
% the axial velocity, U. The matrices RHS and col1 are
% generated by the function CONTMATI.M
%
% [V]=contint(U,RHS,col1);
%
function [V]=contint(U,RHS,col1);
V=-col1*U*RHS;
```

D.4.3 contmati.m

```
%
% contmati generates the constant matrices needed by contint
%
% [RHS]=contmati(n,m);
%
function [RHS]=contmati(n,m);
mu1=makomuhat(n);
eta1=makoetahat(n);
divisor=(eta1*mu1+eye(n))*mu(n)^2;
divisor(:,1)=chebvec(1,n);
divisor(:,2)=chebvec(0,n);
[l,u]=lu(divisor);
RHS=mu1*mu(n)^2*inv(u)*inv(l);
```

```
  u(:,1:n)=R*inv_N;
  if (t~=1),
u(1,:)=u_p(1,:); % the inlet BC
  end;
  v(:,1:n)=contint(u,RHS,col1);
  v(1,:)=zeros(1,n);
  if (dt(dtmark,1)<=t & dtsize>dtmark),
disp('changing discretisation');
dtmark=dtmark+1;
if n>dt(dtmark,8),
  n=dt(dtmark,8);
  output=sprintf('n is now = %g',n);
  disp(output);
  mu01=makemuhat(n+4);
  mu02=mu01^2;
  eta1=makeetahat(n);
  mu01=mu01(1:n,1:n);
  mu02=mu02(1:n,1:n);
  L=linoptint(n,omega);
end;
K=kopint(m,n,omega).*dt(dtmark,2);
N=(mu01-(dt(dtmark,3)*dt(dtmark,2))*L);
N=N*mu(n)^2;
N(:,1)=chebvec(1,n) ./n^2;
N(:,2)=eta1(1:n,1:n)*chebvec(0,n) ./n^2;
[ll,uu]=lu(N);
inv_N=mu(n)^2*inv(uu)*inv(ll);

M=(dt(dtmark,2)*dt(dtmark,4)) .*L+mu01;
disp('end of loop');
  end;

  u=inv(diag(massflow(u)))*u;
  if rem(t,Save)==0 | t==1,
disp('saving to file');
fname=sprintf(...
   'save ./acc.%02g/U%02g%g.mat u v dt t lengthC',...
omega,omega,t);
eval(fname);
fname=sprintf('save u%02g_60 u v dt t lengthC',omega);
eval(fname)
  end;

  if rem(t,Show)==0,
diff=u-u_p;
half=m/4;
disp(max(max(abs(diff(1:half,:)))));
```

```
N=(mu0i-(dt(dtmark,2)*dt(dtmark,3)).*L);
N=N*mu(n)^2;  % shift the matrix right 2 to augment BC's
N(:,1)=chebvec(1,n)./n^2;
N(:,2)=eta1*chebvec(0,n)./n^2;
[ll,uu]=lu(N);
inv_N=mu(n)^2*inv(uu)*inv(ll);

%Uinit=[1 0 0];
%[x,y]=size(Uinit);
%for i=1:m,
%  u(i,1:y)=Uinit;
%end;
%v=zeros(m,n);

Nlin_p=zeros(m,n);
Nlin_pp=zeros(m,n);
Nlin=zeros(m,n);
u_pp=zeros(m,n);
u_p=zeros(m,n);
[x,y]=size(u);
if y >= n,
  u=u(:,1:n);
else,
  u(x,n)=0;
end;
M=(dt(dtmark,2)*dt(dtmark,4)).*L+mu0i;
%
% start the loop ...
%
tstart=cputime;
for t=2:tn,
  % shift the last time matrices to previous times..
  Nlin_pp=Nlin_p;
  Nlin_p=Nlin;
  Nlin=-nonlini(n,u,v,eta1,mu0i,etaimui,coli,Iplusint,C,Cinv);
  u_pp=u_p;
  u_p=u;
  if t==1,
R=u_p(:,1:n)*M+K+ (dt(dtmark,2)*dt(dtmark,6)).*Nlin;
  else,
R=u_p(:,1:n)*M+(dt(dtmark,2)*dt(dtmark,5)).*u_pp(:,1:n)*L+K+ ...
 (dt(dtmark,2)*dt(dtmark,6)).*Nlin; + ...
 (dt(dtmark,2)*dt(dtmark,7)).*Nlin_p;
  end;
  disp(t);
  u(m,FirstN)=0;
```

D.4 The two dimensional base flow equation

D.4.1 base2d.m

```
%base2d.m
%
% this script generates the 2D entrance base flow model
% using the pressure-integral approach.
% Before implementing, specify the global matrices u, v.
%
% set parameters ...
%
%dt= [ti dt   a0  a1  a2 b0 b1 n(140) ...
clg;          % 4e-7
dt=  [100000 7e-6 1 0 0  1  0  60; ...
  100000 1e-4 1 0 0  1  0  32];
[dtsize,y]=size(dt);
n=dt(1,8);
FirstN=n;
m=60;
dtmark=1;
omega=50;
tn=10000;
lengthC=50;  % axial domain is [0,1/lengthC]
Save=20;
Filt=20;
Show=10;
%
% set global matrices ...
%

Xplot=(-lobgrid(m)+1)/(lengthC);
Yplot=(lobgrid(3*n/2)-1)/2;
Yplot=Yplot(1:6:3*n/2);
[RHS]=contmati(n,m);
[eta1,eta2,mu01,etaimu1,col1,Iplusint]=makemati(n,m);
col1=-col1' .*lengthC;
filt=rcosfilt(m);


%
% generate operator matrices
%
L=linopint(n,omega);
K=kopint(m,n,omega).*dt(dtmark,2);
[C,Cinv]=transops(3*n/2);
```

D.3.5 smat.m

```
% smat calculates the shape matrix
%
% [S]=smat(n,eta,mu);
%
% Used by l_oper.m
%
function [S]=smat(n,eta,mu);
k=n;
onemat=zeros(k,k);
onemat(:,1)=ones(k,1);
S=-2 .* (eta(1:k,1:k)*onemat*mu(1:k,1:k));
S=S(1:n,1:n);
```

D.3.6 nmat.m

```
% nmat calculates the spatial operator for the 1-D
% pressure integral formulation
%
% Used by l_oper.m
%
% [N]=nmat(n,eta,mu);
function [N]=nmat(n,eta,mu);
k=n+4;
N=eta(1:k,1:k)*(eye(k,k)+eta(1:k,1:k)*mu(1:k,1:k));
N=N(1:n,1:n);
```

D.3.7 omegamat.m

```
%
% omegamat calculates the omega premultiplier onto u
%
% Used by l_oper.m
%
% [O]=omegamat(n,omega,mu);
function [O]=omegamat(n,omega,mu);
k=n;
O=omega .* mu(1:k,1:k);
O=O(1:n,1:n);
```

```
  end;
end;
save U u dt
```

D.3.2 etamu.m

```
%
% etamu creates the matrices eta and mu
%
% A convenience function used by baseid.m
%
% [eta,mu]=etamu(n);
%
function [eta,mu]=etamu(n);
eta=makeetahat(n);
mu=makemuhat(n);
```

D.3.3 kmt.m

```
%
% kmat calculates the constant RHS matrix;
%
% Used by baseid.m
%
% [K]=kmat(m,n,omega,mu);
function [K]=kmat(m,n,omega,mu);
vec=zeros(m,n);
vec(:,1)=ones(m,1)*omega;
K=vec*mu(1:n,1:n);
```

D.3.4 l_oper.m

```
%
% l_oper calculates the implicit spatial operator
%
% Used by baseid.m
%
% [L]=l_oper(n,omega,eta,mu);
function [L]=l_oper(n,omega,eta,mu);
L=smat(n,eta,mu) + nmat(n,eta,mu) - ...
   omegamat(n,omega,mu);
```

convenience, and discussion of the flow itself in later chapters uses the more conventional radial co-ordinate system with origin on the pipe axis.

The origin of the axis system in the y-z plane (the plane perpendicular to the pipe axis) was taken as being on the inner wall of the pipe, on the horizontal major axis. This was also the most convenient location for the origin, because it corresponded to the point where the laser would be initially aligned before traversing into the flow. Definition of the pipe axis as the origin would not have been useful, because this point was not easy to find. All experimental data was aligned to the more usual centre-based co-ordinate system during processing.

E.5 Tank and pressur. control

The tank pressure had to be controlled to a set level; furthermore the limiting behaviour and position of the piston had to be monitored, both for display and safety purposes. These tasks were the responsibility of two simple objects - the tank control and pressure control objects respectively.

E.5.1 Pressure control

This object contained a reference to the analogue object, in order to obtain pressure information from the transducer mounted in the air cavity of the header tank. It also used the digital output lines of the PC14 to control the small pressure control bleed solenoid. It did not control the pressurising of the tank, merely the maintenance of a set pressure.

The pressure control was a simple 'bang-bang' control system; basically the bleed solenoid could either be switched on or off. To control pressure, the tank was allowed to become slightly over-pressurised by setting the manual regulator accordingly. This extra pressure was then bled off through the small solenoid, the line from which could be slightly throttled to reduce flowrate.

The object implemented a simple control system. Whenever it was called by the main loop it would check the pressure. If the value was above the set point (plus a small dead-band region), the valve would be open to reduce the pressure. If on the other hand the pressure reduced below the set point (minus the dead band quantity), then the valve would be closed. Figure E.2 shows this control of the pressure.

This control was a temporary measure and a more sophisticated variable bleed was planned; however the current acquisition did not use the pressure control at all. Again the merits of the OOP philosophy were evident in this design decision: although the control was crude, it was possible for the object to simply be replaced at a later stage with a better version, *with no modification of any*

E.4.3 Limit object

This encapsulated the behaviour of the limit switches. The object contained a reference to the PC14 to allow the limit data to be read. The bench object used the limit information to stop bench movement in the case of overshoot.

The bench sub-system consolidated the behaviour of the above components. Its interface was simply a set of commands relating to the movement of a particular axis through a particular distance. Internally, it ensured accurate positioning of all axes using feedback information from the position object. In the case of the stepper-motors this was used in stages - the bench was moved a calculated number of steps, a measurement was taken, and the bench was again moved until the error in position was within a certain tolerance (±0.01 mm was used). In the case of the linear actuator more dynamic control was implemented. Position was continually monitored and the unit was brought to rest gradually at the appropriate position. The limits were as mentioned used to ensure no excessive movement of the bench.

Again it can be seen that the result of the object oriented methodology is a high level representation of the LDV traversing bench. This object accepts common sense movement commands, and the complication of even what is moving the bench is hidden from the user of the object.

E.4.4 Interactive bench control

This object was implemented to allow the bench to be moved interactively with mouse or arrow keys. It instantiated a display on the screen, showing both actual and transformed bench positions. When the user pressed the arrow keys (or clicked on buttons on the screen using the mouse), the appropriate axis was moved, and a dynamic read-out of both actual and transformed positions were given. The speed could be changed, and the origin of the bench set by these means.

This functionality was developed to allow the user to position the bench carefully within the apparatus, for instance when finding the pipe wall. When adequate alignment was achieved the control dialogue box could be closed and the program reverted to the normal control screen.

E.4.5 Co-ordinate system & origin

The principal axis of the bench was taken as the long axis, parallel to the pipe. Thus the bench co-ordinate system was unusual. As mentioned, the x co-ordinate was parallel to the pipe, the y co-ordinate was taken as the other horizontal axis, while z was vertically upwards. These definitions were only for

polymorphic handling of the transformation process. Other more complicated transformations inherited from this base class. The two that were implemented were for a flat walled interface, and for a round-in-square interface (the current interface). When given a set of three co-ordinates, these objects could calculate either the forward or reverse transform, that is from bench co-ordinates to pipe co-ordinates or vice versa. Parameters, such as refractive index wall thickness and other dimensions were obviously necessary to calculate the transformation. These were provided by referencing the file settings object, which are described later. The polymorphic nature of the transformation object meant that users of this object (for example the interactive bench movement object) did not need to know what transformation was being used - they simply used the numbers returned by the current transformation.

E.4 The bench sub-system

The bench sub-system was concerned with the movement and control of the LDV bench. It was thus a container object for three other objects; the stepper, actuator and limit switch. It also contained a reference to the position object.

E.4.1 Stepper-motor object

This object used the serial port object in order to communicate to the stepper-motor drive down the serial link. It contained functions to move the stepper-motors and change parameters, which meant that the user of the functions did not have to bother with the assembly of command strings but could call high level movement functions. Thus in the spirit of object orientation, the stepper-motor object truly emulated the behaviour of a set of stepper-motors. Commands were simply of the form `Steppers.Move(X,100);` or `Steppers.Abort();`.

E.4.2 Actuator object

This was the object that controlled the linear actuator. It contained a reference to the PC30 object, in order to allow an analogue output to the actuator. Its operation was different from the stepper-motor object in that the unit had no knowledge of position[2], only of velocity, thus positioning of this unit was controlled completely from the main bench object, in conjunction with the position object.

[2]The stepper-motors 'knew' more or less where they were by virtue of the number of steps they had moved.

E.2.6 Serial I/O object

This object contained all the logic for streaming strings to and from the serial port. It differed from the other objects in that it was not concerned with physical acquisition cards. It utilised the standard DOS interrupts in its operation.

E.2.7 Position I/O object

This card was not an abstraction but represented directly the proprietary card that monitored the readings from the three co-ordinate scales. It provided functionality to read these values, and it contained conversion and offset values for each axis, and implemented a means of changing the origin in each case.

The objects listed above (except for the position and serial objects) represented components of the I/O cards. Both the PC30 and PC14 objects were made up of collections of these sub-components.

E.2.8 PC14 object

This object corresponding to the PC14 card itself, was simply a container object for its components - two digital I/O objects and a single timer object. However it too was descended from the base I/O card object (in order to inherit the port address and port control behaviours), thus it represented a composite structure. It mirrored all the aspects of the physical card.

E.2.9 PC30 object

As with the PC14 object above, this was a composite container object, mirroring all the important aspects of this card. Users of this object could simply read and write values to the ports as voltages. It also managed the interrupt service in combination with the analogue object it contained, allowing an interrupt driven acquisition service, which was used by the flowmeter.

E.3 Transformation sub-system

The path of the LDV beams through the pipe resulted in distortion of the beams, due to Snell's law. This object was created in order to be able to determine the actual measuring volume position from the LDV position. In reality the structure of this sub-system was more complicated than shown in figure E.1. A base class object was defined which was simply a unity transformation. Given three co-ordinates, it would simply return the same values. This was done to allow

E.2.2 Port control object

This was a single global object in the system, owned by the main object. It stored a list of port allocations and their uses. Thus on a crude level it prevented different objects from attempting to utilise the same port addresses. It did however have a far more important function, and that was preventing conflicts between objects utilising a single card for different purposes.

To explain this, it is best to consider the configuration of the PC14 digital card. This card was multi-purpose in that different I/O lines could be placed in different modes; for instance some in normal output and others in bi-directional handshaking input. However the mode of the card was set through a single I/O port, and thus simply writing a new value to it would corrupt settings for the other parts of the port. It was the job of the port control object to administer the allocation of the settings on the various ports to prevent such a conflict occurring.

Having explained the ancestor and port control objects, the specific types of port object will be listed. Each of these defined specific behaviour of certain types of port, for instance the serial I/O object was concerned with sending strings to the serial port whereas the digital I/O object simply transmitted integers.

E.2.3 Timer I/O object

This was primarily concerned with control of the 8253 timer chip which was resident on both the PC14 and PC30 cards. Specific functions were provided to set the counter into various modes, initialise the counter value and so on.

E.2.4 Digital I/O object

This was concerned with control of the 8255 I/O chip and its initialisation. With the use of the port control object the chip could be set into any of its modes, and handshaking and data read information could be acquired.

E.2.5 Analogue I/O object

This encapsulated specific set-up behaviour of some of the component registers of the PC30 card. It did not represent the card itself, but was a 'convenience' object to allow the collection of specific analogue acquisition behaviour.

loop does not access this object, so it should really be encapsulated within some sub-object. However, it is widely used, and because it represents a single item (the physical card itself), not more than one copy of the object should be in existence. Thus the current layout (one analogue object owned by the main object) is a means to avoid duplication of the object. This layout breaks the encapsulation in that the main object should not 'know' about the A/D card directly. However this object is never used in the main loop; rather a pointer to it is passed to each new object (i.e. Pneumatic valve) as they are created. The sole responsibility of the main loop is simply to manage the analogue card.

In general the containment structure must be a taxonomy or hierarchy, whereas the reference relationship can form a net or web.

What follows is a detailed review of the basic components shown in figure E.1. A description is given for each object, starting from the lower level and working up to the description of the high level design objects. The function of the lower objects is mainly simple control of hardware components, or the performance of administrative tasks (storing data). The higher level structure ' more abstract, and represent general functions of the system, for insta.. polling data.

E.2 The input/output sub-system

This sub-system encompassed the low level I/O objects, and included all the specific instances (PC14, PC30) of these objects. Also included were the important port control object, responsible for managing I/O address allocation amongst other concerns.

E.2.1 I/O card object

As mentioned before, the base I/O card object encapsulated the basic behaviour of all the I/O cards. This object stored the card base address and a reference to the port control object. It also defined utility functions for reading and writing bytes to the low level I/O ports of the machine[1] The behaviour of the port control object is described in detail below; it was important to avoid conflicts when utilising the I/O devices.

[1]This was generic behaviour - all I/O devices in the IBM system are controlled by reading or writing data to an I/O port.

The general restrictions, behaviour and conditions of the various interrelationships are outlined below.

E.1.2 Containment

All objects must form part of a containment hierarchy. Thus all the generic or multi-purpose objects that are not directly owned by sub-components of the system are contained by the main object. This is indicated on the diagram, using the broken line containment symbol to avoid excessive crossing of lines. At program start the main component is responsible for creating all its contained components. If these components in turn contain other sub-components, then it is the responsibility of the container in each case to instantiate its contained objects. In this way all objects in the system are created. On program termination, the reverse occurs: all containers are responsible for destroying their contained objects, and each object is further responsible for shutting down its own associated (physical or logical) system. For instance the stepper-motor object must reset the drive before shutdown. Control of the entire system by the main object is thus achieved by means of containment.

E.1.3 Inheritance

The inheritance arrows are not part of the dynamic behaviour but rather describe *static* similarities between related objects. As mentioned, they do not strictly belong in this figure, but are shown to indicate that all the I/O objects are descended from a base I/O class that describes the fundamental behaviour of all such objects. Thus the management of the I/O is through the generic functionality of the base class - common behaviour does not need to be 're-invented' for each specific I/O object.

E.1.4 Reference

The reference relationship is more an indicator of the working interaction of the objects with one another. Containment also entails reference (ownership is automatic reference), thus both open and closed circles indicate the dynamic interactions of the system. The acquisition control object is a good example of reference. When this object is running it requires control of a number of the rig components. Most of these components are used by other objects at other times, thus the acquisition object acquires or controls the object by reference. Because it knows about the referenced objects, it can call all their associated functions.

Some containment links are *not* used in the dynamic running of the system. A good example of such a link is that to the Analogue I/O object. The main

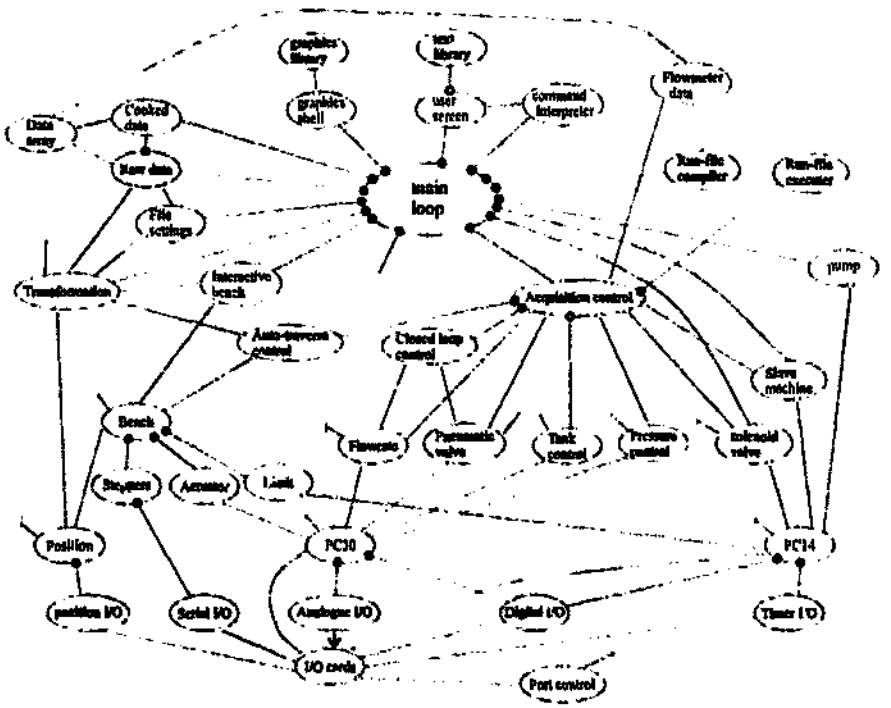

Figure E.1: The top-level object relationships in the system: →, inheritance; •, containment; o, reference.

from the object pointed to. However the one object does not own the other; it is just aware of it. A good illustration of this relationship is in the closed-loop control of the flowrate. Here the closed-loop object needs to know the flowrate, but because the flowmeter object is a general resource, other objects need to reference it too. Thus in this case it is the correct design for the closed-loop object to contain a reference to the flowmeter object. The closed circle indicates containment. Here the one object owns the other object. For instance the bench object, which encapsulates the behaviour of the entire traversing system, must have full control of the stepper-motors. No commands should be passed directly to the steppers themselves; rather all requests for bench movement must be made to the high level bench object. The bench is entirely responsible for the stepper management, including the creation and destruction of their memory representation. As far as the outside 'world' (the high level software) is concerned, only the bench exists. This is an excellent example of the strong encapsulation of the OOP method. The inheritance arrows point from the child to the parent object. The reference and containment symbols point the other way, from the referred to (or contained) class to the referring (or container) class. The line and dot can be thought of as belonging to the subordinate class and 'embedding' itself into its principal.

Appendix E

The main control software

This appendix presents a detailed description of the structure of the control and acquisition software. The design is presented in an object-oriented manner, with the functional description of the various components being in terms of the interaction between these objects. This description assumes that the reader has some basic knowledge of the concepts of object-oriented design, although the fundamental structure of the design should be evident despite this. The reader is referred to Rumbaugh (1991) for further reading on the object-oriented method.

The overall design is best illustrated by an object diagram (figure E.1), showing the functional relationships between the components.

E.1 General design principles

E.1.1 The elements of the figure

This figure follows roughly the notation of Booch (1994), concerning the representation of object relationships in Object-Oriented Programming (OOP), although in a much simplified form. The various symbols on the interconnecting lines indicate the type of relationship between the objects.

The arrows represent inheritance. Thus for instance, considering the Analogue I/O object, the arrow pointing from it to the I/O card object can be read as "The Analogue I/O object *is a type of* I/O card object". The inheritance relationship should not strictly be shown on the figure, because the parent class is merely a generalisation for its child(ren) and thus doesn't actually exist. Its use does however serve to indicate the underlying commonality of the lower I/O objects. Except for in the limited use of this relationship, all other objects exist as unique instances in the software system. The open circle indicates 'reference'; that is, an object contains a pointer to another so that it can use or obtain information

```
alb=alp+dalp;
curve=zeros(3,3);
curve(1,1)=ala;
curve(2,1)=alp;
curve(3,1)=alb;


[curve(1,3),curve(1,2)]=findre(n,curve(1,1),o,L1,L2,M,U,UL,...
   bc,rea,reb,remax,tol,k);
[curve(2,3),curve(2,2)]=findre(n,curve(2,1),o,L1,L2,M,U,UL,...
   bc,rea,reb,remax,tol,k);
[curve(3,3),curve(3,2)]=findre(n,curve(3,1),o,L1,L2,M,U,UL,...
   bc,rea,reb,remax,tol,k);

%initial guesses for alpha bracket the given value
while (abs(curve(3,3)-curve(1,3)) >alptol),
  [aa,i]=sort(curve(:,1));
  curve=curve(i,:);
  % curve is sorted in increasing alpha
  fit=polyfit(curve(:,1),curve(:,3),2);
  root=-fit(2)/(2*fit(1));
  curve(4,1)=root

  [curve(4,3),curve(4,2)]=findre(n,curve(4,1),o,L1,L2,M,U,UL,...
   bc,rea,reb,remax,tol,k);
  [aa,i]=sort(curve(:,3));
  save __temp;
  % curve is sorted in increasing Re
  curve=curve(i,:);
  curve=curve(1:3,:);
  % the smallest 3 elements of curve are retained
  A=curve(1,1);
  R=curve(1,3);
  E=curve(1,2);
end;
```

```
eigc=sexl(n,alp,rec,o,L1,L2,M,U,UL,bc);
[I]=find(real(eigc)<=k);
eigc=eigc(I);
ec=max(imag(eigc))
if sign(ec)==sign(eb),
  eb=ec;
  reb=rec;
else,
  ea=ec;
  rea=rec;
   end;
  end;
  niter=niter+1;
  if (niter>50),
error('Iteration overflow (niter>50)') ;
  end
  if (reb>remax),
error('Max Reynolds number exceeded');
  end
end
rey=rec;

[Y,I]=sort(imag(eigc));
I=flipud(I);
eigc=eigc(I);
eigval=eigc(1);
```

D.5.9 getnose.m

```
%
% This function finds the nose of the stability curve for pipe
% flow by using a polynomial iteration on findre
% function
% [alpha,rey,eig]=getnose(n,alp,rey,L1,L2,M,U,UL,
%             bc,tol,alptol,omega);
function [A,R,E]=getnose(n,alp,rey,L1,L2,M,U,UL,bc,tol,alptol,o);

remax=1e12;   %If above this Re, flow deemed stable
k=0.5;          % eigenvalue real component search cutoff
dalp=0.00002;
rea=rey;
reb=rey+200;


ala=alp-dalp;
```

D.5.8 findre.m

```
%
% this function takes n (order), L1,L2,M,D, two initial guesses
% for rey, and a toler(ence), and then finds the neutral
% stability curve crossing for that alpha by using newton
% iteration on the function genpois. A maximum Re is supplied to
% terminate the search.
%
% [rey,eig]=findre(n,alp,omega,L1,L2,M,U,UL,bc,rea,reb,remax,...
%                  tol,k)
%
function [rey,eigval]=findre(n,alp,o,L1,L2,M,U,UL,bc,rea,reb,...
 remax,toler,k)

counter=1;
eiga=sexl(n,alp,rea,o,L1,L2,M,U,UL,bc);
eigb=sexl(n,alp,reb,o,L1,L2,M,U,UL,bc);
[I]=find(real(eiga)<=k);
eiga=eiga(I);
ea=max(imag(eiga))
[I]=find(real(eigb)<=k);
eigb=eigb(I);
eb=max(imag(eigb))
niter=1;
while (abs(reb-rea)>toler) ,
 if sign(eb) == sign(ea),
disp('signs the same');
rec=reb-(reb-rea)*(eb/(eb-ea))
eigc=sexl(n,alp,rec,o,L1,L2,M,U,UL,bc);
[I]=find(real(eigc)<=k);
eigc=eigc(I);
ec=max(imag(eigc))
rea=reb;
reb=rec;
ea=eb;
eb=ec;
else,
  disp('signs differ');
%    if counter==1,
%      counter=0;
  rec=(rea*eb-reb*ea)/(eb-ea)
%    else,
%      counter=1;
%      rec=0.5*(rea+reb)
%    end;
```

Flow variation

The constants in the above table defined the required flow rate variation for the feedback control. For the various chosen flow variations, the desired flow was given by:

$$
\begin{aligned}
Q(t)_{\text{poly}} &= Q_{\max}\left[k_0 + k_1 \times t + k_2 \times t^2\right], \\
Q(t)_{\text{trig}} &= Q_{\max}\left[k_0 + k_1 \sin(k_2 \times t)\right], \\
Q(t)_{\text{exp}} &= Q_{\max}\left[k_0 + k_1 \exp(k_2 \times t)\right],
\end{aligned}
$$

E.11.1 Command-line and macro syntax

All commands issued at the command prompt, or included within a macro, are of the form

`{xx}␣COMMAND:{arg,arg,arg,...}`

All items within braces are dependent on the context, or optional. The **xx** is a numeric label, useful only in the case of a macro, where it is used in a **GOTO:** statement. The portion **COMMAND:** is any command from table E.1 below, or in the case of a macro, also from table E.2. The terms **arg** are arguments, the number depending on the specific command. Table E.1 also lists the type and number of arguments appropriate for each command.

Macro syntax

In addition to the definition of the basic line syntax defined above, there were particular issues that related to the use of the commands within macros. Firstly, there were enhanced features available as given in table E.2, and secondly, some syntactical structure was necessary for the macro. A general macro prototype is given by:

```
INFO: Some descriptive title of the macro
{nn} COMMAND:{arg,arg,...}
...
...
{nn} COMMAND:{arg,arg,...}
END:
```

E.11.2 Run file commands

The command syntax for the run-file is simpler than for the macro. Each line of the run file can either consist of a comment (indicated by starting the line with a **%**), or must to be of the form

`time␣command␣value`

where **time**, a real number, is the time in seconds, **command** is the action to be taken, and **value** is the value appropriate for the action. The list of permissible commands is contained in figure E.3.

and the rest of the program behaviour was implicit in the object interactions.

E.10.3 Data display sub-system

The function of this object was simply to display the LDV, flowmeter and other acquired data graphically, so that an evaluation of data quality, etc., could be made. Typing the command `PLOT:` invoked this component, which exhibited 'pop-up' behaviour, i.e. it had to be closed down before program execution could continue.

The graphical display could overlap data from the different channels, as well as the flowmeter data. The screen had to be explicitly 'cleared' to show the data on a fresh set of axes, and data not erased remained plotted after the graph was closed down. This allowed over-plotting of data from different tests, useful in making comparisons.

E.10.4 Run-file compiler

This is the last item to be described here. It was an utility object that allowed the generation of the control lists used in the acquisition from a text file. The input file was an ASCII format file, with each line of the file in the format

`<time>␣<command>␣<arguments>`

where `<time>` was the time of execution of the command in seconds, zero being the beginning of the test, `<command>` was a mnemonic for the particular command, for instance `SOL` for the solenoid, and `<arguments>` were the relevant parameters for the command - in the case of the solenoid, 1 meant on and 0 off.

The compiling procedure was adopted for two reasons. Firstly, by checking the passed commands during a compilation, errors were avoided during the running of the command list; secondly, the use of a binary list as opposed to an interpreted string allowed far greater speed during the control loop.

E.11 Command syntax and definitions

The above sections described the top-level object design of the system. This section summarises the commands used to operate the acquisition software.

(1) The string "MOVE:Y,200" was stripped of leading and trailing spaces, and converted to upper-case letters. It was then searched from the beginning until the colon was encountered. The characters before the colon were examined to see if they matched any recognisable command.

(2) After recognising "MOVE" as a command, the characters before and including the colon were discarded. The remaining string "Y,200" was then parsed further.

(3) The string was parsed by looking for commas, and the individual string segments were forwarded to a specific part of the parsing loop dedicated to the "MOVE" command. Here both arguments were converted into a numerical form (the "Y" became 2 - the second axis). A number corresponding to the "MOVE" command, along with extracted arguments, were then returned to the main execution loop.

(4) Using the command number, the main loop jumped to the move command. The move command was then directly executed, passing the returned parameters as arguments.

The parser also had the ability to parse macro files. Enhanced capabilities for the parser were implemented in this context, such as the ability to make branching decisions, or to loop. These extra features were however crudely implemented but were just adequate enough to allow the rig to operate automatically for the present tests.

E.10.2 Main command loop

The main command loop simply consisted of exactly that - a loop. At the beginning of the loop the string from the keyboard was fed to the parser. The resulting return value from this object caused the program to branch to the correct point, where the relevant command was carried out. This process continued until the `QUIT:` command was entered, at which point the program terminated. Every single function of the program was implemented by this mechanism. The full set of functions are tabulated and described in section E.11 that follows. The command loop was in fact embedded in the main program object. This object owned all other objects in the system, and was responsible for their construction at start-up, and their destruction at program termination. On the top-most level the main object had only three functions that were invoked: `init()`, which did the system initiation, `run()`, which started the message loop, and `done`, which destroyed all the child objects and then terminated the program. In fact the main program itself only consisted of the three lines:

```
MainLoop.Init();
MainLoop.Run();
MainLoop.Done();
```

When a command was received within the control loop for the stepper to move, a movement command was sent directly to the stepper object (bypassing the closed loop control of the bench). The stepper would then move at the set rate to approximately the correct position. Instead of controlling movement accurately with the feedback information, the accurate measurement information was rather stored in the same manner as the flowmeter data, in a time referenced form. Using this time-referenced position information, data could later be processed into time-position-velocity triplets.

While the process was novel and yielded interesting global insights into the flow development, in the accelerative context the rate of scan was too low (the entire bench had to be moved) for even reasonable qualitative conclusions to be drawn. Furthermore, data processing was extremely difficult, especially when trying to reconstruct velocity profiles from the data. It was for these reasons that the technique was not employed in the current work.

E.10 Main command loop and user interface subsystems

All the objects and sub-systems thus far presented did not interact directly with the user. A user interface sub-system was designed to facilitate communication between the user and the underlying systems.

This took the form of a textual or console interface. The user entered commands by typing them, using a well defined syntax, and the action was then carried out. Other features of the interface were: the ability to write macros (text files containing a sequence of console commands), access to a text editor for editing the macros (and run files), and a graphical interface for simple display of data. In order to achieve this functionality a number of interacting objects were designed, which are described below.

E.10.1 Command parser

The need to interpret keyboard commands required the design of a command parser or interpreter. This object accepted a string of characters as an argument, and returned an enumerated command number, and a series of numbers, dependent on the command issued. The command number was an index into a stored array of commands, and the other parameters were the arguments for that particular command.

The parsing process is best illustrated by example. For instance, the movement of the bench manually was through a command of the form `MOVE:Y,200`. This particular command moved the bench y co-ordinate by 200 mm. Roughly speaking, the parser worked as follows when receiving this command :

effects. A rubber section was inserted downstream of the flowmeter to try to diminish any pressure waves resulting from start-up, but it was decided to take further measures on rig shutdown, to avoid dangerous impulsive deceleration under pressure.

Controlled shutdown was implemented through a standard function. When the testing terminated for whatever reason, the abort function controlled the rig until it was deemed safe to simply close all valves.

On being called, the function immediately *opened* the solenoid valve and discharged the tank of air, thereafter the pneumatic valves were closed. The tank pressure and flowrate were then monitored, and the solenoid was only shut when it was deemed that the pressure pulse would be low enough not to cause any damage. The pipes would possibly have withstood the full pressure pulse - after having been weakened by passing through the rubber section - but it was decided to err on the side of caution to prevent mishap.

E.8.6 General considerations

The acquisition loop was designed to be flexible and fast. The list-operated polling mechanism made it very general, which matched the generality of the user interface, to be explained later. Finally, the acquisition and control process was safe, and the rig could be brought to a safe halt under all circumstances.

E.9 Auto traversing sub-system

The bench control described in section E.4 was not a dynamic process. Movement of the bench was only permitted during periods when the rig was not under dynamic control. However during the design of the software the idea was mooted of moving the bench *while* acquiring data. This was later successfully achieved by introducing the auto-traverse control object into the control loop, and results from the use of this technique for steady-laminar flows were presented in Abbot, Moss & Olivier (1992).

This object was limited in that it used only the stepper-motors, and utilised the fact that the steppers could be commanded to move fairly accurately to a given destination without constant intervention. This allowed the stepper control to be quick, because a command was simply sent to the remote stepper drive box for processing, with the movement not involving the central computer further. The linear actuator on the other hand required closed loop control by the computer for positioning, and was consequently too costly in terms of time to implement within the acquisition loop. Because of the exclusive use of the steppers the movement was restricted to the horizontal plane only.

The position of the piston or the occurrence of leakage were monitored by this unit with each loop. If the piston position exceeded its limits or water was detected behind the diaphragm (implying a rupture) then the test was immediately stopped.

Pressure control was also implemented on a constant basis during operation of the rig. The pressure-control object was described earlier

As a final safety consideration, the keyboard was polled for any keystroke during testing, and the detection of a key press resulted in immediate ending of the test.

E.8.4 Analogue acquisition & interrupts

The analogue acquisition did not concern itself with the acquisition loop. Rather an interrupt service was instantiated at the beginning of the test and terminated on completion. The PC30 had a hardware interrupt service that could be set to generate interrupts at a particular frequency. An interrupt handler was written to acquire the flowmeter and other analogue data from the AD card object.

The bench co-ordinates were also polled using another interrupt procedure chained into the analogue one. This was primarily for the scanning acquisition procedure given in the next section. All the other critical functions could have been tied onto the interrupt chain, but it was decided to leave them in the normal loop for a number of reasons.

- From the point of accurate polling, the interrupt was hardware driven and took precedence over all other computer functions. Thus the accuracy of the time base of both the analogue and bench acquisition was ensured.
- While some of the other functions in the main loop were critical, this loop was fast enough to ensure immediate response. This was because the polling rate of the loop was over 20 kHz.
- Complicated interaction (keyboard polling) within interrupt handlers was not possible under DOS due to innate limitations of the operating system.

The interrupt service transferred data directly to memory. At the termination of the loop the interrupt service was stopped. The function that achieved this could be called at any time; thus during premature stopping of the test the service stopped as required.

E.8.5 The abort sequence

It was feared that immediate shutdown of the rig under full pressure, and with maximum flowrate, might endanger the glass test section due to water-hammer

E.8.2 Command execution

If the return value was not zero then the appropriate command was executed. Within the loop was a case statement. Depending on the command number the appropriate command was executed. All commands had to be quick to execute, and any action requiring extensive processing had to be avoided. The available commands were:

Control the solenoid valve. The downstream solenoid valve could be switched on or off, dependent on the value of the returned parameter.

Slave machine. A signal could be sent to the slave machine, telling it to either start or stop acquiring data. Thus data acquisition could be started well into a test, which was useful when for instance only the far-time behaviour was important. After stopping a test, resuming it would clear the previous data; this action was only really intended to be carried out once.

Pneumatic valve. The valve could be moved to an arbitrary position, dependent on the returned parameter. There was no closed-loop control of the valve, but simple manual positioning. This command was useful in pre-setting the valve before implementing closed-loop control; alternatively it could simply be used to vary the flowrate in a stepped manner.

Closed loop control. The object controlling this was described earlier. Within the control loop, feedback control could be switched on or off, and the various flow variation constants could be set. Each action took the form of a separate command; thus to vary all four of the flow variation constants, 4 separate commands had to be issued in succession.

Auto-traverse control. This allowed the bench to be moved during testing. Separate commands were issued to the traversing object to either set parameters (velocity, axis) or initiate movement. This object is described in section E.9.

E.8.3 Safety & rig control

The above commands were 'optional' in a sense - it was the user's choice which (if any) should be implemented. There were however some mandatory commands and functions which ensured the proper functioning and safety of the rig during the test.

It contained references to all the controllable or measurable elements of the rig: the shave, solenoid, tank pressure, tank control, valves, closed loop control, as well as the auto-traversing object, which is described below.

The object executed a tight acquisition loop, controlled by commands from the run-file execution object. This loop read the time value from the system timer object, and then executed the command from the run-file structure when the time was correct. Dependent on the commands it received, the appropriate actions were then implemented. The general structure of this loop is shown in figure E.3

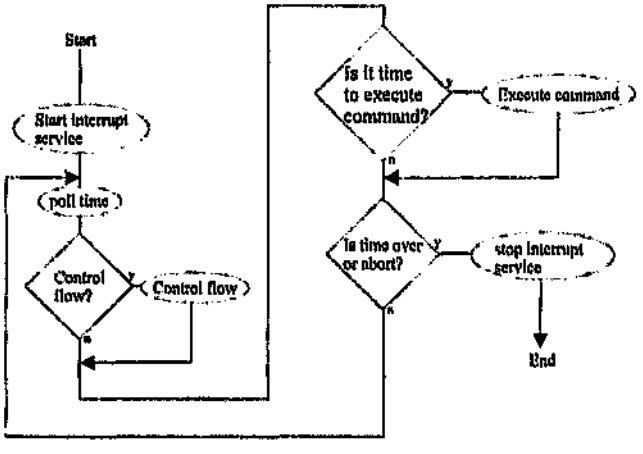

Figure E.3: The data acquisition & control loop.

The basic components of the pool were thus the interrupt service, the flow control and the command execution. Considering the execution of the command first, the structure of the run-file object must be considered.

E.8.1 Run-file execution object

This object simply contained a list of commands ordered in time. Each item on the list contained three numbers: the time of execution, the command type and the parameter value for the execution. These values were retrieved simply by calling a function `thelist.getcmd()`, which took the current time as an argument. If the time passed exceeded the time of the first value on the list, the function returned the command number and the parameter, or else it returned zero. Thus the loop could simply poll a single function on each iteration. If the command parameters were returned, an internal pointer within the run-file object was incremented to point to the next item.

needed for this study. It was used only by the more general control sub-system, described in section E.8 below.

The control of flowrate was a classic control theory problem. A measured flow (from the flowmeter) had to be controlled to follow a chosen variation in flowrate by moving a controllable valve. A general-purpose control object was built, which implemented proportional, integral, derivative and pseudo-derivative control mechanisms. The values for the various control constants were given in the settings object, and components could be deactivated simply by setting an appropriate constant to zero.

The object contained references to the flowrate and pneumatic valve object, and obtained its time base from the acquisition object. The main control function of the object was implemented as a 'fall-through' function; that is the control function was called within the data acquisition loop, and kept running totals of the various relevant control parameters (running integral, value of last point) without delaying the loop. By this means appropriate values could be sent to the valve each time the function was called, based on calculations using the stored numbers.

The desired function was supplied to the object as a series of constants. A limited implementation was developed, whereby a quadratic, sinusoidal or exponential flow variation could be specified. Four constants were passed to the object; the first determined the type of acceleration (quadratic, exponential, etc.) while the other three, dependent on the value of the first constant, defined the desired flow variation in time. These constants could be set 'on-the-fly' during control, so that piece-wise continuous or even discontinuous functions assembled from the three basic types of variation could be assembled.

Because of the nature of dynamic control, no calibration of the valve was necessary, only adjustment of the parameters. The final configuration only utilised the integral and pseudo-derivative mechanisms, on the advice of Coustantcou (1991), and achieved a suitably responsive and accurate system. However the extreme range of the flow variation coupled with the non-linearity of the valve, resulted in a control system that was not fully optimum over the entire range of operation. A suitable improvement to the control system, such as an adaptive control procedure, would correct this potential problem, although none was implemented.

E.8 Acquisition control sub-system

This object was a primary component of the system. It was fully responsible for all data acquisition and control during the running of the test. The main loop handed full control to this object during testing, thus allowing for high speed control and acquisition.

data object. This was achieved by passing a reference to this data object to the acquire function, as `Slave.acquire(^RawdataArray)`. The function then returned data in this array (or an error in the case of failure). Thus the slave object did not contain a reference to the data array, but was passed one as a parameter (on a 'need-to-know' basis). This is the weakest form of interaction between objects; a simple usage. There were as many slave objects as there were slaves. These were dynamically allocated and destroyed as the computer user varied their choice of acquisition channels.

E.6.2 Data objects and data processing

The data loaded into the raw data object was completely unprocessed. It consisted of three bytes of LDV data and four bytes of time information. This data was required in a normal velocity-time format, with time in seconds and velocity in metres per second. To achieve this conversion, a number of constant and variable data parameters were required.

Because of the two forms of data - processed and unprocessed - there were two forms of data object: raw and cooked as they were informally labelled. They both inherited from a data array, an abstract object that 'knew' how to load and store to disk, and display itself to the graphics screen. An instance of both the raw and cooked data existed for each LDV channel, thus in the present system there were two of each sort of array, one pair per channel.

The raw data contained references to the transformation object and the file settings object, from which it was able to extract enough information to convert itself to a proper velocity-time format. The transformation provided the beam angle and tilt of the measuring volume, while the file settings gave information set by the user such as the frequency of the light, LDV counter mode of operation and frequency shift[3], or measured parameters such as the timer calibration and cycles per burst.

The raw data contained a function - `convert` - which took as an argument a reference to the cooked data array. This function would then fill the cooked array with values of velocity and time values calculated using all the factors obtained from the transform and file settings objects.

E.7 Closed-loop flow control

This was obviously one of the most critical control sub-systems in the software, for the reason that it was used to generate the accelerating flow variations

[3]These settings could not be ascertained directly, thus they were a source of possible error in that it was possible for them not to match the computer values.

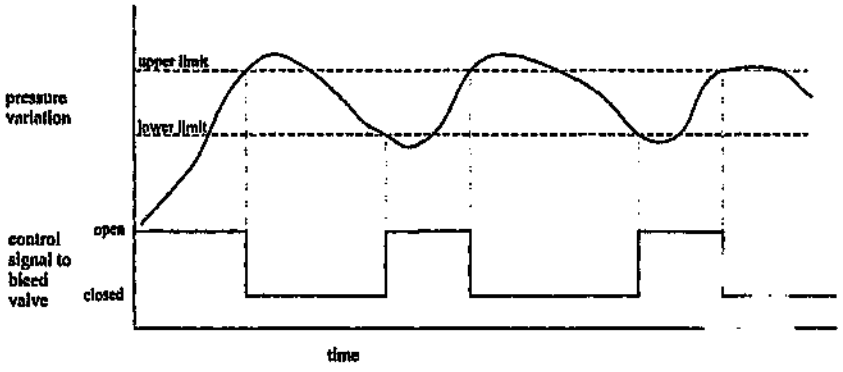

Figure E.2: The principle of controlling the tank pressure.

other source code. This is in stark contrast to conventional approaches, where many functions, all over the code, would have to be modified to implement such a change.

E.5.2 Tank control

The tank control object was merely for acquisition. It measured tank position (and any leakage status), and was thus useful to both the acquisition and main sub-systems, both described later.

E.6 The slave control, data transfer and processing sub-system

This sub-system was really composed of two different components. The slave control and data transfer was due to the slave machine object, while the processing was done by the data itself. They are grouped together here because they formed part of a contiguous process in the running of the program.

E.6.1 Slave control & data transfer

As mentioned this was achieved by the slave machine object. It contained a reference to the PC14 port and could thus communicate to the slave computer via the network described in section 5.3, issuing commands and receiving data. It encapsulated the low-level hand-shaking necessary between the main computer and the slave machine, and the interfaces were simply to send the commands and read the status. Further to simple command passing, if the `acquire` command was issued then the data from the slave was returned and loaded into the raw

Gaster M. Roberts J B. (1977) *The spectral analysis of randomly sampled records by a direct transform* Proc. Roy. Soc. A. **354** 27-58

Gersting J.M. (198 *Numerical methods for eigensystems: The Orr-Sommerfeld problem as an initial-value problem* Comp. and Maths. with Apps. **6** 167-174

Gilbert N. (1988) *Numerische simulation der transition von der laminaren in die turbulente kanalströmung* Dissertation, Univ. Karlsruhe

Gill A.E. (1965) *On the behaviour of small disturbances to Poiseuille flow in a circular pipe* J. Fluid Mech **21** 145-172

Glezer A. Katz Y. Wygnanski I. (1989) *On the breakdown of the wave packet trailing a turbulent spot in a laminar boundary layer* J. Fluid Mech. **198** 1-26

Goldstein S. (1930) *Concerning some solutions of the boundary layer equations in hydrodynamics* Proc. Cambr. Phil. Soc. **26** 1-30

Goldstein M.E. (1985) *Scattering of acoustic waves into Tollmein-Schlichting waves by small streamwise variations in surface geometry* J. Fluid Mech. **154** 509-529

Gruebel W.P. (1970) *The stability of pipe flow Part 1. asymptotic analysis for small wave-numbers* J. Fluid Mech. **43** part 2 279-290

Greenblatt D. (1990a) *Averaging and temporal gradient bias problems in unsteady turbulent LDV measurements* Fifth Int. Symp. Laser Techniques to Fluid Mech. Lisbon, Portugal.

Greenblatt D. (1990b) *Pipe flow relaminarization by temporal acceleration* Dissertation, Wits Univ.

Grosch C.E. Salwen H. (1968) *The stability of steady and time-dependent plane Poiseuille flow* J. Fluid Mech **34** 177-205

Grosch C.E. Salwen H. (1978) *The continuous spectrum of the Orr-Sommerfeld equation. Part 1. The spectrum and eigenfunctions* J. Fluid Mech. **87** 33-54

Gupta S.C. Garg V.K. (1981) *Effect of velocity distribution on the stability of developing flow in a pipe* Phys. Fluids **24** part 4 576-578

Gustavsson L.H. (1991) *Energy growth of three-dimensional disturbances in plane Poiseuille flow* J. Fluid. Mech **224** 241-260

Hegge-Zijnen B.G. van der (1924) *Measurements of the velocity distribution in the boundary layer along a plane surface* Thesis, Delft.

Erlebacher G. Hussaini M.Y. Speziale C. Zang T.A. (1987) *Towards the large-eddy simulation of compressible turbulent flow* ICASE Rep. No. 87-20 (NASA Langley Research Centre, Hampton, VA)

Fasel H. Konzelmann U. (1990) *Non-parallel stability of a flat-plate boundary layer using the complete Navier-Stokes equations* J. Fluid Mech. **221** 311-347

Fingerson L.M. Adrian R.J. Kaufman S.L. (1989) *Laser doppler velocimetry: Theory, application and techniques* TSI LDV course text.

Fox L. (1962) *Chebyshev methods for ordinary differential equations* Computer J. **4** 318-331

Fox L. Parker I.B. (1968) *Chebyshev Polynomials in numerical Analysis* Oxford University Press, London

Frazer R.A Jones W.P. Skan S.W. (1937) *Approximation to Functions and to the Solution of Differential Equations* R & M 1799 Aeronautical Research Council, London

Garg V.K. (1981) *Stability of developing flow in a pipe: non-axisymmetric disturbances* J. Fluid Mech. **110** 209-216

Garg V.K. Rouleau W.T. (1972) *Linear spatial stability of pipe Poiseuille flow* J. Fluid Mech. **54** part 1 113-127

Gary J. Helgason R. (1971) *A matrix method for ordinary differential eigenvalue problems* J. Comp. Phys. **5** 169-187

Gaster M. (1962) *A note on the relation between temporally-increasing and spatially-increasing disturbances in hydrodynamic stability* J. Fluid. Mech. **14** 222-224

Gaster M. (1965) *The role of spatially growing waves in the theory of hydrodynamic stability* Prog. Aerospace Sci. **6** 251-270

Gaster M. (1970) *The growth of three-dimensional disturbances in inviscid flows* J. Fluid. Mech. **43** 837-839

Gaster M. (1974) *On the effects of boundary layer growth on flow stability* J. Fluid Mech **66** part 3 465-480

Gaster M. (1989) *Personal communication with D.F. da Silva* Dept. of Eng. Cambridge Univ. U.K.

Gaster M. Roberts J.B. (1975) *Spectral analysis of randomly sampled signals* J. Inst. Maths. Appls. **15** 195-216

Constantcon C. (1991) *Personal communication* School of Mech. Eng. Univ. of the Witwatersrand South Africa

Corcos G.M. Lin S.J. (1984) *The mixing layer: deterministic models of a turbulent flow. part 2. The origin of the three-dimensional motion* **139** 67-95

Corcos G.M. Sellars J.R. (1959) *On the stability of fully developed flow in a pipe* J. Fluid Mech. **5** 97-112

Craik M.D.D. (1985) *Wave interactions and fluid flows* Cambridge Univ. Press

Crane C.M. Burley D.M. (1976) J. Comp. Appl. Maths. **2** 95

Cummins H.Z. Knable N. Yeh Y. (1964) *Observation of diffusion broadening of Rayleigh scattered light* Phys. Rev. Lett. **12** 150-153

da Silva D.F. (1990) *Transition to turbulence in impulsively accelerated pipe flows* (PhD Thesis, University of the Witwatersrand, Johannesburg)

da Silva D.F. Moss E.A. (1994) *The stability of pipe entrance flows subjected to axisymmetric disturbances* Trans. ASME J. Fluids Eng **116** 61-65

da Silva M.R.de.J. (1985) *A Quick Survey of Recent Developments and Applications of the τ-Method* in *Numerical Approximation of Partial Differential Equations* Int. Symp. N. Anal. ed. L.Ortiz (Elsevier Science Publ. Netherlands)

Davey A. (1973) *A simple numerical method for solving Orr-Sommerfeld problems* Quart. J. Mech. Appl. Math. **26** 401-411

Davey A. Drazin P.G. (1969) *The stability of Poiseuille flow in a pipe* J. Fluid Mech. **36** 209-218

Drain L.E (1977) *The laser doppler technique* J. Wiley & Son

Dimotakis P.E. (1979) *Single scattering particle laser doppler measurements of turbulence* Appl. of Non-Intrusive Instr. in Fluid Flow Meas. AGARD.

Drazin P.G. Reid W.M. (1981) *Hydrodynamic stability* (Cambridge University Press, Cambridge)

Durst F. Melling A. Whitelaw J.H. (1976) *Principles and practice of laser doppler anemometry* New York: Academic Press

Edward R.V. Dybbs A. (1984) *Refractive index matching for velocity measurements in complex geometries* TSI quarterly **X** issue 4

Ekman V.W. (1910) *On the change from steady to turbulent motion of liquids* Ark. f. Mat. Astron. och Fys **6** No. 12

Bouthier M (1972) *Stabilitie lineaire des ecoulements presque paralleles* J. Mechanique **11** 599-621

Bouthier M (1973) *Stabilitie lineaire des ecoulements presque paralleles. Partie II. La couche limite de Blasius* J. Mechanique **12** 75-95

Bridges T.J. Morris P.J. (1984a) *Differential eigenvalue problems in which the parameter appears nonlinearly* J. Comput. Phys. **55** 437-460

Bridges T.J. Morris P.J. (1984b) *Spectral calculations of the spatial stability of Non-parallel boundary layers* AIAA Pap. No. 84-0437

Bridges T.J. Morris P.J. (1987) *Boundary layer stability calculations* Phys. Fluids **30** Part 11 3351-3358

Buchhave P. George W.K.Jr Lumley J.L. (1979) *The measurement of turbulence with the laser doppler anemometer* Ann. Rev. Fluid Mech. **11** 443-503

Buchave P. von-Benzon H.-H Rammussen C.N. (1990) *LDA bias: comparison of measurement errors from simulated and measured data* Proc. 5th Int. Symposium on Applications of Laser Techniques to Fluid Mechanics **29.3**

Burgers J.M. (1925) *The motion of a fluid in the boundary layer along a plane smooth surface* Proc. First Intern. Congr. of Appl. Mech. Delft, ed. C.B. Biezeno J.M. Burgers

Burridge D.M. (1972) *The stability of Poiseuille pipe flow to non-axisymmetric disturbances* Geophys. Fluid Dyn. Inst. Florida State Univ. Tech. Report **43**

Chandrasekhar S. (1961) *Hydrodynamic and hydromagnetic stability* Oxford Univ. Press

Chen T.S. Joseph P.D. (1967) *Subcritical bifurcation of plane Poiseuille flow* J. Fluid Mech. **58** 337-351

Canuto C. Hussaini M.Y. Quarteroni A. Zang T.A. (1988) *Spectral Methods in Fluid Dynamics* Springer-Verlag, New York

Carnahan B. Luther H.A. Wilkes J.O. (1969) *Applied numerical methods* John Wiley & Sons, New York

Cooper E.R Jankowski D.F. Neitzel G.P. Squire T.H. (1985) *Experiments on the onset of instability in unsteady circular Couette flow* J. Fluid Mech. **161** 97-113

Conrad P.W. Criminale W.O. (1965a) *The stability of time dependent laminar flow: parallel flows* Z. Agnew. Math. Phys. **16** 233-254

Conrad P.W. Criminale W.O. (1965b) *The stability of time dependent laminar flow: flow with curved streamlines* Z. Agnew. Math. Phys. **16** 569-582

References

Abbot A.H. Moss E.A. (1994) *The existence of critical Reynolds numbers in pipe entrance flows subjected to infinitesmal axisymmetric disturbances* Phys. Fluids 6 part 10 3335-3340

Abbot A.H. Moss E.A. Olivier G.H. (1992) *A laser traversing technique for the measurment of velocity profiles in unsteady pipe flows* Proc. 6th Int. Symposium on Applications of Laser Techniques to Fluid Mechanics 2.3

Ajmani D.B.S. Roberts J.B. (1990) *Improved spectral estimation for irregularly spaced data using digital filters* Mechanical Systems & Signal Processing 4 77-94

Antar B.N. Benek J.A. (1978) *Temporal eigenvalue spectrum of the Orr-Sommerfeld equation for the Blasius boundary layer* Phys. Fluids 21 183-189

Bairstow L. (1925) *Skin friction* J. Roy. Aero. Soc. 19 3

Barnes H.T Coker E.G. (1905) *The flow of water through pipes* Proc. Roy. Soc. 74 341-356

Barry M.D.J. Ross M.A.S. (1970) *The flat plate boundary layer Part 2. The effect of increasing thickness on stability* J. Fluid Mech. 43 Part 4 813-818

Bayly B.J. Orszag S.A. Herbert T. (1988) *Instability mechanisms in shear-flow transition* Annu. Rev. Fluid Mech. 20 359-391

Bibby *Bibby technical data* J. Bibby science products limited, Stone, Staffords. ST15 OSA

Blasius H. (1908) *Grenzschichten in flüssigkeiten mit kleiner reibung* Z. Math. Phys. 56 1-37 (English translation: NACA TM 1256)

Bohr N. (1909) *Determination of the surface tension of water by the method of jet vibration* Phil. Trans. Roy. Soc. A 209 281-317

Booch G. (1994) *Object-oriented analysis and design with applications. Second edition* Benjamin/Cummings publishing California, USA

Appendix G

Raw data on CD-ROM

The CD-ROM contained in the sleeve below contains the raw data as compressed ASCII files, arranged in a subdirectory structure in accordance with the test names. The CD-ROM is self-documented: the read-me file in the root directory serves as a 'road-map', and points to more specific information files contained in the further subdirectories. The media was created using the standard CDFS file system, and as a result should be readable from both DOS/WINDOWS and UNIX systems. Data compression is via the Pkzip utility, which is freely available as shareware (i.e. at most Internet FTP sites).

Model 9169-450 focusing lens: This unit focuses the widely spaced beams emanating from the beam expander, and focuses them at a single point in the flow. Refracted light from this measuring volume is collected by this lens and transmitted back through the field-stop system, and into the photo-detectors. It had a focal length of 450 mm.

Model 9176 mounting ring: This unit serves as a support for the components. Several mounting adapters were spaced evenly down the length of the optical barrel.

Model 9180A frequency shift unit: Two of these units were incorporated into the system. Two model 9182 Brag cells were placed in the optical barrel - one horizontal and one vertical. The function of these units was to shift the frequency of one laser beam from each pair. The frequency shift system also consists of a pair of 9186 electronics model, to down-mix the signals from the photo-detectors.

Model 9175 beam steering module: A pair of these units was incorporated after the Brag cells, to allow adjustment of the beam path of the shifted beam.

Model 9144A mounting turret: This unit serves as a mounting point for the two photo-detectors. Returning light from the measuring volume is reflected up into this unit by an angled mirror (the incident beams pass this mirror on either side). A semi-silvered mirror within the turret splits the received light into two equal components, which are then focused onto each of the pair of model 9160 photo-detectors. Before passing onto the photo-detectors, the reflected light is passed through pair of model 9158 band-stop filters - green for the one photodetector and blue for the other - in order to separate the reflected components of the light.

Model 9143 field-stop system: This unit contains an aperture through which the returning light is focused. Its function is to remove unwanted flare and reflected signals not emanating directly from the measuring volume. The incident beams pass through this unit unhindered.

Model 9181 beam stop Two of these units were incorporated, in the barrel, at right-angles to each other. They attenuated the 'extra' beams generated by the Bragg cells, using two adjustable knife-edges.

Model 9113 beam spacer: A pair of these units were attached in front of the field-stop unit. Their function is to bring each pair of beams closer together, to allow them to pass properly into the beam expander.

Model 9189 beam expander: This unit separates the incident laser beams and increases their diameter. It allows them to be subsequently focused with a large increase in the signal-to-noise ratio and decrease in measuring volume dimension.

Model 9108 beam collimator: This unit collim. the laser beam; that is it ensures that the minimum beam diameter occurs at the measuring volume.

Model 9101-1 polarisation rotator: This unit ensures that the beam polarity is optimum for transmission through the later optics. It allows the laser polarisation direction to be rotated at will.

Model 9105 colour separator: This is a compound unit, whose primary task is to separate the incident laser beam into its spectral components, and then to transmit two of these components (blue at 488.0 nm and green at 514.5 nm). Its internal components are: model 9204 attenuator; model 9106 dispersion prism; one model 9106 and two model 9107 mirrors; and a model 9136 enclosure late. The entire set of components is enclosed by the model 9136 enclosure over. The two chosen colour laser beams leave the enclosure and pass into the optical barrel, the components of which are described below.

The optical barrel incorporates all the transmitting and receiving optics. The green and blue laser beams enter the unit at the rear - one on-axis and one displaced off-axis - while the resulting four beams leave the front transmitting lens to intersect in the flow being measured. The components of this unit are as follows.

Model 9102 polarisation rotator: This polarisation device alters the polarisation of an individual colour (the on-axis beam). The previous model 9101-1 unit changed the polarisation of the entire multi-spectral laser beam.

Model 9116 beam splitter: The central beam emanating from the colour separator is split into two equal components by this unit. This unit is mounted so that the resulting two beams are positioned vertically above one another.

Model 9174 beam displacer: This unit brings the off-axis beam onto the central axis.

Model 9102 polarisation rotator: The second polarisation device varies the polarisation of the other colour beam (now brought on-axis by component 9174 above).

Model 9116 beam splitter: A second beam splitter now splits the new on-axis beam in the horizontal plane. As a result two pairs of beams leave this unit - one pair in the horizontal plane and one pair vertically aligned.

Appendix F

The LDV components

This appendix contains a detailed listing of the components comprising the two channel 9100 series TSI Laser Doppler Velocimeter used in the current study. This system is a four beam-two channel device. All model numbers used below are for TSI components unless otherwise stated. A schematic of the entire unit is shown in figure F.1 below.

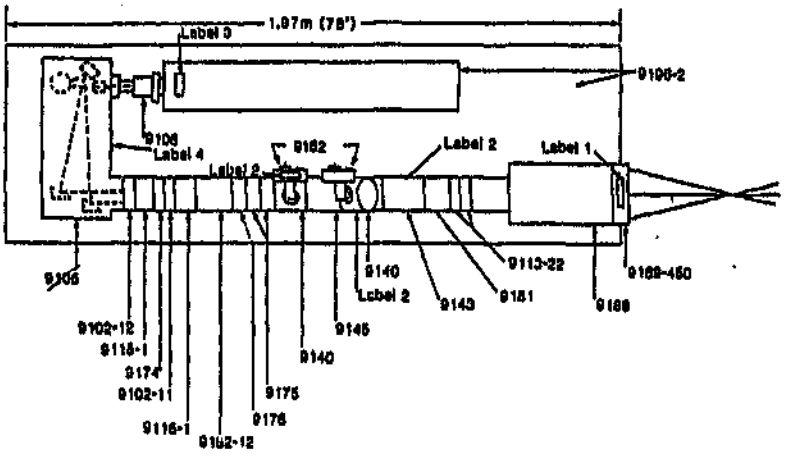

Figure F.1: Layout of the model 9100-7 LDV system (from TSI System 9100-7 LDV instruction manual).

Mounting base: The entire LDV assembly was mounted on a standard model 9127 base. This unit has a regularly spaced array of mounting holes across its face, allowing attachment of components anywhere on its surface. The rigidity of this unit allowed the entire assembly to be supported on three mounting points bearing onto its underside.

Table E.2: The enhanced program control commands for macro use.

command	parameters	meaning
`DECISION:`	int N, string S	Display the string 'S', if the user presses the 'Y' key jump to the label N.
`END:`	none	The last line of the macro. Denotes the end.
`FOR:`	char C, int I,J	Set variable C∈[I,J,K] to each value between I and J, execute commands below until `NEXT:` command is reached.
`GOTO:`	int N	Jump to the label N
`INFO:`	string S	First line of macro. A descriptive title.
`INPUT:`	string S, var V	Display S, read keyboard input into the variable V∈[VARI,VARS,VARR] (see below).
`NEXT:`	char C	See `FOR:`, return program control to the matching `FOR:` loop. C∈[I,J,K].
`VARI`	N/A	The integer system variable. Most commands expecting an integer parameter will accept `VARI` as argument.
`VARR`	N/A	As above, but for real number arguments
`VARS`	N/A	As above, but for string arguments

Table E.3: The run-file commands and their values.

command	parameter	meaning
`PID`	$N \in [0,1]$	Start/stop closed loop valve control (1-start)
`PK0`	real R	Set the k_0 flow variation constant
`PK1`	real R	Set the k_1 flow variation constant
`PK2`	real R	Set the k_2 flow variation constant
`PTY`	$N \in [0,2]$	Set the type of flow variation desired 0=polynomial, 1=trigonometrical, 2=exponential
`SLA`	$N \in [0,1]$	Start or stop LDV acquisition (1=start)
`SOL`	$N \in [0,1]$	Open/close the downstream solenoid (1-open)
`VAL`	$N \in [0,4096]$	Move the pneumatic valve to position N. The range of N corresponds to the analogue output range (0-10V)

Table E.1: The commands for the acquisition program.

command	parameters	meaning
`ABORT:`	none	Stop testing, release pressure in tank
`COMPILE:`	string S	Compile the run-file 'S.mov'. If successful the compiled file 'S.exm' is produced.
`CHANNELS:`	int N	Set the LDV acquisition channels. 0=none, 1=channel 1, 2=channel 2, 3=both.
`CYCLES:`	int M,N	Set the cycles-per-burst for channel M$\in$ [0, 1] to value N
`DIR:`	string S	Lists the contents of the directory 'S'
`EDMAC:`	string S	Edit the macro names 'S.mak'
`EDMOV:`	string S	Edit the run-file 'S.mov'
`FORWARD:`	none	Open the solenoid allowing water into the bottom tank (move piston forward)
`FSHIFT:`	int M,N	Set the frequency shift of LDV channel M to N kHz.
`GETPOS:`	char C	With C$\in [x, y, z]$, returns axis position in mm with respect to the absolute bench co-ordinates
`LISTMAC:`	string S	List all macros (*.mak) in current macro directory
`LISTMOV:`	string S	List all run-files in current macro directory
`LOAD:`	string S	Load the LDV data from the files 'S' (no extension)
`LOADDEF:`	string S	Load the parameters from the (beam angle, etc.) from the parameter file into the file settings object
`MOVEABS:`	char C, int N	Move axis C$\in$[X,Y,Z] to position N mm, relative to the absolute bench origin
`MOVEINT:`	none	Start the interactive bench movement screen
`MOVEREL:`	char C, int N	As with `MOVEABS:` above, but move relative to current position
`PLOT:`	int N	Allows plotting of the current data, plus analogue channel N.
`PRESSURE:`	none	Pressurise the header tank
`RETRACT:`	none	Pump water into the header tank until full (piston retracted)
`RUN:`	string S	Execute the test defined by the compiled run-file 'S'. Invoke the acquisition loop with list contained in the file 'S.exm'
`SAVE:`	string S	Saves the LDV and analogue data to files 'S.*' in current data directory
`SETPRESS:`	real R	Change the set-point tank pressure to R bar
`SETTLE:`	int N	Wait for N seconds before continuing
`SHELL:`	none	Open a DOS session (i.e. for deleting files, etc.). Return by typing 'EXIT'.
`STARTMAC:`	string S	Executes the macro named 'S.mak'
`TRANSFER:`	none	Retrieve the data from the slave machines
`ZERO:`	none	Set the origin of the (y, z) ordinates at the current bench position
`ZEROPRIN:`	none	Set the principal (x) ordinate to zero at the current bench position

Wygnanski J.H. Haritonidis J.H. Kaplan R.E. (1979) *On a Tollmien-Schlichting wave packet produced by a turbulent spot* J. Fluid Mech. **92** part 3 505-528

Wygnanski I.J. Sokolov M. Friedman D. (1975) *On transition in a pipe. Part 2. The equilibrium puff* J. Fluid Mech. **69** part 2 283-304

Wood G. (1992) *The growth of large amplitude disturbances in impulsively started pipe entrance flow* School of Mech. Eng. Witwatersrand Univ. internal report

Wright C. (1992) *Personal communication* Dept. of Applied Maths. Witwatersrand Univ. South Africa.

Tietgen R. (1975) *Laminar turbulent transition in pipe flow measurements made with pitot probe, hot wire sensor and LDA* Proc. LDA Symp. Copenhagen

Tietgen R. (1979) *Laminar turbulent transition in pipe flow: Development and structure of the turbulent slug* IUTAM Laminar-Turbulent Transition

Tollmein W. (1929) *Über die entstehung der turbulenz* Nachr. Ges. Wiss. Göttingen Math.-phys. Kl. 21-44

Töpfer C. (1912) *Bemerkungen zu dem aufsatz von H. Blasius: Grenzschichten in flüssigkeiten mit kleiner reibung.* Z. Math. Phys. 60 397-398

van der Sande E. Belde A.P. Hamer B.J.G. Helmstra W. (1980) *Velocity profiles in accelerating pipe flows started from rest* BHRA Fluid Eng. 3rd Internat. Conf. Press. Surges Canterbury, England.

Van Stijn Th. L. Van de Vooren A.I. (1983) Computers in fluids 10 223

Volodin A.G. (1973) Izv. Sib. Otd. Akad. Nauk. SSSR 2 p13

Von Kerczek C.H. (1982) *The instability of oscillatory plane Poiseuille flow* J. Fluid Mech. 116 91-114

Ward R.C. (1975) *The combination shift QZ algorithm* SIAM J. Numer. Anal. 12

Watson J. (1960) *On the nonlinear mechanics of wave disturbances in stable and unstable parallel flows. part 2. The development of a solution for plane Poiseuille flow and for plane Couette flow* J. Fluid Mech. 9 371-389

Watson J. (1962) *On spatially-growing finite disturbances in plane Poiseuille flow* J. Fluid Mech. 14 211-221

Wilkinson J.H. (1965) *The algebraic eigenvalue problem* Clarendon Press, Oxford

Wilkinson J.H. (1979) *Kronecker's canonical form and the QZ algorithm* Linear Alg. and Appl. 28 285-303

Wilkinson J.H. Reinsch C. (1971) *Handbook for automatic computation* Springer-Verlag, Berlin

Witoszynski C. (1924) *Über strahlerweiterung und strahlablenkung Z.-F.* Vortrage aus dem Gebiete der Hydro- und Aerodynamik, Springer, Berlin.

Wygnanski I.J. Champagne F.H. (1973) *On transition in a pipe. Part 1. The origin of puffs and slugs and the flow in a turbulent slug* J. Fluid Mech. 59 part 2 281-335

Sexl T. Spielberg K. (1948) *Zum stabilitätsproblem der Poiseuilleströmung* Acta. Phys. Austriaca **12** 9-28

Shah R.K. (1976) *A correlation for laminar hydrodynamic entry length solutions for circular and non-circular ducts* Trans. ASME J. Fluids Eng. **100** 177-179

Shen S.F. (1954) *Calculated amplified oscillations in the plane Poiseuille and Blasius flows* J. Aero. Sci. **21** 62-64

Shen S.F. (1961) *Some considerations on the linear stability of time-dependent basic flows* J. Aero. Scien. **28** 397-404, and 417

Silberman I. (1954) *Planetary waves in the atmosphere* J. Meteorol. **11** 27-34

Slater J.C. (1934) *Electronic energy bands in metal* Phys. Rev **45** 794-801

Smith F.T. (1979a) *On the non-parallel flow stability of the Blasius boundary layer* Proc. Roy. Soc. A **366** 91-109

Smith F.T. (1979b) *Non-linear stability of boundary layers for disturbances of various sizes* Proc. Roy. Soc. A **368** 573-589

Sommerfeld A. (1908) *Ein beitrag zur hydrodynamishen erklaerung der turbulenten fluessigkeitsbewegungen* Proc. 4th Int. Congress of Mathematicians Rome **vol III** pp116-124

Spalart P.R. Leonard A. (1985) *Direct numerical simulation of equilibrium turbulent boundary layers* Proc. 5th Symp. Turbulent shear flows Cornell Univ. Ithaca, NY

Sperrow E.M. Lin S.H. Lundgren T.S. (1964) *Flow development in the hydrodynamic entrance region of tubes and ducts* Phys. Fluids **7** 338-347

Squire H.B. (1933) *On the stability of three-dimensional disturbances of viscous flow between parallel walls* Proc. R. Soc. London Ser. A **142** 621-628

Stuart J.T. (1960) *On the nonlinear mechanics of hydrodynamic stability* J. Fluid Mech. **9** 353-370

Szegö G. (1939) *Orthogonal polynomials* Vol 23 (AMS Coll. Publ., New York)

Tam C.K.W. (1986) *Excitation of instability waves by sound - a physical interpretation* J. Sound **105** 169-172

Tatsumi T. (1952) *Stability of the laminar inlet flow, prior to the formation of Poiseuille regime, I* J. Phys. Soc. Japan **7** part 5 489-502

Thomas L.H. (1953) *The stability of plane Poiseuille flow* Phys. Rev. **91** part 4 780-783

Salwen H. Grosch C.E. (1972) *The stability of Poiseuille flow in a pipe of circular cross-section* J. Fluid Mech. **54** 93-112

Salwen H. Grosch C.E. (1981) *The continuous spectrum of the Orr-Sommerfeld equation. Part 2. Eigenfunction expansions* J. Fluid Mech **104** 445-465

Sandham N.D. Kleiser L. (1992) *The late stages of transition to turbulence in channel flow* J. Fluid Mech. **245** 319-348

Saric W.S Nayfeh A.H. (1975) *Nonparallel stability of boundary layer flows* Phys. Fluids **18** 945-950

Saric W.S. Thomas A.S.W. (1984) *Experiments on the subharmonic route to turbulence in boundary layers* Turb. & Chaotic Phenom. in Fluids ed. T Tatsumi 117-122 Amsterdam

Sarpkaya T. (1975) *A note on the stability of developing laminar pipe flow subjected to axisymmetric and non-axisymmetric disturbances* J. Fluid Mech **68** Part 2 345-351

Schiller L. (1922) *Untersuchungen über laminate und turbulente strömung* Forschg. Ing. -Wes. Heft **428**

Schlichting H. (1933) *Zur entstehungder turbulenz bei der plattenströmung* Nachr. Ges. Wiss. Göttingen Math.-phys. Kl. 181-208

Schlichting H. (1979) *Boundary Layer Theory* (McGraw-Hill, New York)

Schmid P.J. Henningson D.S. (1994) *Optimal energy density growth in Hagen-Poiseuille flow* J. Fluid Mech. **277** 197-225

Schmidt F.W. Zeldin W. (1969) *Laminar flows in the inlet sections of tubes and ducts* AIChE J. **15** part 4 612-614

Schubauer G.B. Skramstad H.K. (1947) *Laminar boundary layer oscillations and transition on a flat plate* J. Aeronaut. Sci. **14** 69

Sen P.K. Venkateswarlu D (1983) *On the stability of plane Poiseuille flow to finite-amplitude disturbances, considering the higher-order Landau coefficients* J. Fluid Mech. **133** 179-206

Sen P.K. Venkateswarlu D. Maji S. (1985) *On the stability of pipe-Poiseuille flow to finite amplitude axisymmetric and non-axisymmetric disturbances* J. Fluid Mech. **158** 289-316

Sexl T. (1927) *Zur stabilitätsfrage der Poiseuilleschen und Couetteschen strömung* Ann. Phys. Lpz. (4) **83** 835-848

Ortiz E.L. (1969) *The tau method* SIAM J. Numer. Anal. 6 480-492

Ortiz E.L. Samara H. (1981) *An operational Approach to the Tau Method for the Numerical Solution of Non-Linear Differential Equations* Computing 27 15-25

Ortiz E.L. Samara H. (1983) *Numerical Solution of Differential Eigenvalue Problems with an Operational Approach to the Tau Method* Computing 31 95-103

Patera A.T. Orszag S.A. (1981) *Finite-amplitude stability of axisymmetric pipe flow* J. Fluid Mech. 112 467-474

Pekeris C.L. (1948) *Stability of the laminar flow through a straight pipe of circular cross-section to infinitesmal disturbances [illegible] are symmetrical about the axis of the pipe* Proc. Nat. Acad. Sci. Wash. 34 [illegible] 95

Pekeris C.L. Shkoller B. (1967) *Stability of plane Poiseuille flowto periodic disturbances of finite amplitude in the vicinity of the neutral curve* J. Fluid Mech. 29 31-38

Popiel Cz.O. Trass O. (1991) *Visualization of free and impinging round jet* Source unknown, reprint from the author.

Rayleigh Lord (1880) *On the stability, or instability, of certain fluid motions* Proc. London Math. Soc. 11 57-70

Reynolds O. (1883) *An experimental investigat [illegible] the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels* Phil. Trans. Roy. Soc. 174

Reynolds W.C. Potter M.C. (1967) *Finite amplitude instability of parallel shear flows* J. Fluid Mech. 27 465-492

Rivlin T.J. (1990) *Chebyshev polynomials* Wiley-Interscience

Roberts J.B. Gaster M. (1980) *On the estimation of spectra from randomly sampled data: a method of reducing variability* Proc. Roy. Soc. A. 371 235-258

Ross J.A. Barnes F.H. Burns J.G. Ross M.A.S. (1970) *The flat plate boundary layer Part 3. Comparison of theory with experiment* J. Fluid Mech. 43 819-832

Rubin Y. Wygnanski I.J. Haritonidis (1979) *Further observations on transition in a pipe* IUTAM Lam. Turb. Transition.

Rumbaugh J. Blaha M. Premerlani W. Eddy F. Lorenson W. (1991) *Object-oriented modeling and design* Prentice Hall

Salwen H. Cotton F.W. Grosch C.E. (1980) *Linear stability of Poiseuille flow in a circular pipe* J. Fluid Mech. 98 part 2 273-284

Munakata K. (1979) *Linear instabili„ of pipe Poiseuille flow* J. Phys. Soc. Japan Letters **47** No. 2 685-686

Munakata K. (1983) *The instability of Poiseuille flow in a circular pipe* J. Phys. Soc. Japan **52** No. 6 2004-2015

Murdock J.W. (1986) *Three-dimensional numerical study of boundary-layer stability* AIAA Pap. No. 86-0434

Nikuradse J. *Laminare reibungsschichten an der längsangestromten platte* Monograph Zentrale f. wiss. Berichtswesen, Berlin

Ng B.S. Reid W.H. (1979) *An initial value method for eigenvalue problems using compound matrices* J. Comp. Phys. **30** 125-136

Nishioka M. Iida S. Ichikawa Y. (1975) *An experimental investigation of the stability of plane Poiseuille flow* J. Fluid Mech. **72** 731-751

Noether F. (1921) *Das Turbulenzproblem* Z. Agnew. Math. Mech. **1** 125-138, 218-219

Orr W. M'F. (1907) *The stability or instability of the steady motions of a perfect liquid and of a viscous liquid* Proc. Roy. Irish Acad. A **27** 9-138

Orszag S.A. (1970) *Transform methods for calculation of vector coupled sums: Application to the spectral form of the vorticity equation* J. Atmosph. Sci. **27** 890-895

Orszag S.A. (1971a) *Accurate solution of the Orr-Sommerfeld stability equation* J. Fluid. Mech **50** 689-703

Orszag S.A. (1971b) *Numerical simulations of incompressible flows within simple boundaries: accuracy* J. Fluid mech. **49** 75-112

Orszag S.A. (1971c) *Galerkin approximations to flows within slabs, spheres, and cylinders* Phys. Rev. Lett. **26** 1100-1103

Orszag S.A. (1971d) *Numerical simulation of incompressible flows within simple boundaries: I. Galerkin (spectral) representations* Stud. Appl. Math. **50** 293-327

Orszag S.A. (1972) *Comparison of pseudospectral and spectral approximations* Stud. Appl. Math. **51** 253-259

Orszag S.A. Kells L.C. (1980) *Transition to turbulence in plane Poiseuille flow and plane Couette flow* J. Fluid Mech. **96** 159-205

Orszag S.A. Patera A.T. (1983) *Secondary instability of wall-bounded shear flows* J. Fluid. Mech. **128** 347-385

Luke Y.L. (1969b) *The special functions and their approximations* Vol 2 Academic Press, New York, London

Mack L.M. (1976) *A numerical study of the temporal eigenvalue spectrum of the Blasius boundary layer* J. Fluid mech. **73** 497-520

Malik M.R. Hussaini M.Y. (1990) *Numerical simulation of interactions between Görtler vortices and Tollmein-Schlichting waves* J. Fluid Mech. **210** 183-199

Maruyama T. Kato Y. Mizushina T. (1977) *Transition to turbulence in starting pipe flows* J. Chem. Eng. Japan **11** No 5 340-352

Maslowe S.A. (1981) *Shear flow instabilities and transition* (from *Hydrodynamic instabilities and the transition to turbulence* ed. Swinney H.L. Gollub J.P.) Springer-Verlag

McDonnel C. Fitzpatrick J.A. (1992) *Spectral analysis of turbulent flows from LDA measurements* Proc. 6th Int. Symposium on Applications of Laser Techniques to Fluid Mechanics **8.3**

McLachlan R.I (1991) *The boundary layer on a finite flat plate* Phys. Fluids A. **3** 2 341-348

McLaughlin D.K. Tiederman W.G. (1973) *Biasing corrections for individual realizations of laser anemometer measurements in turbulent flows* Phys. Fluids **16** No 12 2082-2088

Metcalfe R.W. Orszag S.A. (1973) *Numerical calculation of the linear stability of pipe flows* Flow Research Report **25**, Kent, Wash.

Moler C.B. Stewart G.W. (1973) *An algorithm for generalized matrix eigenproblems* SIAM J. Numer. Anal. **10** 241-256

Morkovin M.V. Reshotko E. (1989) *Dialogue on progress and issues in stability and transition research* Laminar-Turbulent transition IUTAM Symp. Toulouse Springer-Verlag.

Moser R.D. Moin P. (1987) *The effects of curvature in wall-bounded turbulent flows* J. Fluid Mech. **175** 479-510

Moss E.A. (1985) *Transition to turbulence in pipe flows accelerated from rest* PhD thesis Witwatersrand Univ. South Africa

Moss E.A. (1991) *Laminar pipe flows accelerated from rest* N & O Journaal 4 7-14

Moss E.A. da Silva D.F. (1993) *The evolution of unstable regions in impulsively started pipe entrance flows* Phys. Fluids A. **5** part 11 2721-2724

Landau L.D. Lifshitz (1959) *Fluid mechanics. Vol. 6 of a course on theoretical physics* translated from russian by J.B. Sykes & W.H. Reid, Pergamon press London

Lefebvre P.J. White F.M. (1989) *Experiments on transition to turbulence in a constant-acceleration pipe flow* ASME J. Fluids Eng. 111 part 12 428-432

Lefebvre P.J. White F.M. (1991) *Further experiments on transition to turbulence in constant-acceleration pipe flow* ASME J. Fluids Eng. 113 part 6 223-227

Leite R.J. (1958) *An experimental investigation of the stability of Poiseuille flow* J. Fluid Mech. 5 81-96

Lessen M Sadler S.G. Liu T.Y. (1968) *Stability of pipe Poiseuille flow* Phys. Fluids 11 No. 7 1404-1409

Levchenko V.Y. Solovyev A.S. (1974) Izv. Sib. Otd. Akad. Nauk. SSSR 3 7

Lin C.C. (1944) *On the stability of two-dimensional parallel flows* Proc. Nat. Acad. Sci., Wash. 30 316-323

Lin C.C. (1945) *On the stability of two-dimensional parallel flows* Quart. Appl. Math. 3 117-142, 218-234, 277-301

Lin C.C. (1955) *The Theory of Hydrodynamic Stability* Cambridge Univ. Press, Cambridge

Lindgren E.R. (1969) *Propagation velocity of turbulent slugs and streaks in transition pipe flow* Phys. Fluids 12 No. 2 418-425

Ling C.H. Reynolds W.C. (1973) *Non-parallel flow corrections for the stability of shear flows* J. Fluid Mech. 59 part 3 571-591

Liu K.M. Ortiz E.L. (1982) *Approximation of eigenvalues defined by ordinary differential equations with the Tau method* in Matrix Pencils (Kägström,B. Ruhe A. eds.) LNM no. 973, (Springer-Verlag)

Liu K.M. Ortiz E.L. (1986) *Numerical solution of eigenvalue problems for partial differential equations with the tau-lines method* Comp. & Maths. with Appls. 12B 1153

Liu K.M. Ortiz E.L. (1987) *Tau method approximate solution of high-order differential eigenvalue problems defined in the complex plane, with an application to Orr-Sommerfeld stability equation* Communic. Appl. Num. Meth. 3 187

Luke Y.L. (1969a) *The special functions and their approximations* Vol 1 Academic Press, New York, London

Itoh N. (1974c) *Spatial growth of finite wave disturbances in parallel and nearly parallel flows. Part 2. The numerical results for the flat plate boundary layer* Trans. Japan Soc. Aer Space Sci. **17** 175-186

Itoh N. (1977) *Nonlinear stability of parallel flows with subcritical Reynolds numbers. Part 1. An asymptotic theory valid for small amplitude disturbances* J. Fluid Mech. **82** part 3 455-467

Itoh N. (1986) *The origin and subsequent development space of Tollmein-Schlichting waves in a boundary layer* Fluid Dyn. Res. **1** 119-130

Jacobson I. Christerson M. Jonsson P. Övergaard G. (1992) *Object-oriented software engineering. A use case driven approach* Addison Wesley

Jordinson R (1970) *The flat plate boundary layer. Part 1. Numerical integration of the Orr-Sommerfeld equation* J. Fluid Mech. **43** 801-811

Joseph D.D. Sattinger D.H. (1972) *Bifurcating time periodic solutions and their stability* Arch. Rat. Mech. Anal. **45** 79-109

Kantorovic L.V. (1934) *On a new method of approximate solution of partial differential equations* Dokl. Akad. Nauk. SSSR **4** 532-536 (Paper in Russian)

Kachanov Y.S. Levchenko V.Y. (1984) *The resonant interaction of disturbances at laminar-turbulent transition in a boundary layer* J. Fluid Mech. **138** 209-247

Kachanov Y.S. (1987) *On the resonant nature of the breakdown of a laminar boundary layer* J. Fluid Mech **184** 43-74

Katz Y. Siefert A. Wygnanski I. (1990) *On the evolution of the turbulent spot in a laminar boundary layer with a favourable pressure gradient* J. Fluid Mech. **221** 1-22

Kelvin Lord (1871) *Hydrokinetic solutions and observations* Phil. Mag. (4) **42**

Klebanov P.S. Tidstrom K.D. Sargent L.M. (1962) *The three-dimensional nature of boundary layer instability* J. Fluid Mech. **12** 1-34

Kleiser L. Zang T.A. (1991) *Numerical simulation of transition in wall-bounded shear flows* Annu. Rev. Fluid Mech. **23** 495-537

Koslov V.V. Ramazanov M.P. (1984) *Development of finite amplitude disturbances in Poiseuille flow* J. Fluid Mech. **147** 149-157

Lanczos C. (1938) *Trigonometric interpolation of empirical and analytical functions* J. Math. Phys. **17** 123-199

Laudau L.D. (1944) *On the problem of turbulence* C. R. Acad. Sci. URSS **44** 311-314

Heisenberg W. (1924) *Über stabilität und turbulenz von flussigkeitsströmen* Ann. Phys., Lpz. **74** 577-627 (Translated as: *On stability and turbulence of fluid flows* Tech. memor. Nat. Adv. Comm. Aero. Wash. (1951) **1291**)

Helmholtz H. von (1868) *Über discontinuirliche flüssigkeitsbewegungen* Monats. Königl. Preuss. Akad. Wiss. Berlin **2** 215-228.

Herbert T. (1977) *Die neutrale Fläche der ebenen Poiseuille-Strömung* Habilitation Univ. Stuttgart

Herbert T. (1980) *Nonlinear stability of parallel flows by high-order amplitude expansions* AIAA J. **18** 243-248

Herbert T. (1983a) *Stability of plane-Poiseuille flow: theory, and experiment* Fluid Dyn. Trans. **11** 77-126

Herbert T. (1983b) *Secondary instability of plane channel flow to subharmonic three-dimensional disturbances* Phys. Fluids **26** 871-874

Herbert T. (1984) *Analysis of the subharmonic route to transition in boundary layers* AIAA Pap. No. 84-0009

Herbert T. (1985) *Three-dimensional phenomena in the transitional flat-plate boundary layer* AIAA Pap. No. 85-0489

Herbert T. (1988) *Secondary instability of boundary layers* Annu. Rev. Fluid Mech. **20** 487-526

Hornbeck R.W. (1963) *Laminar flow in the entrance region of a pipe* Appl. Sci. Res. Sect. A **13** 224-232

Howarth L. (1938) *On the solution of the laminar boundary layer equations* Proc. Roy. Soc. London A **164** 547-579

Huang L.M. Chen T.S. (1973) *Stability of the developing laminar pipe flow* Phys. Fluids **17** No. 1 245-247

Huang L.M. Chen T.S. (1974) *Stability of developing pipe flow subjected to non-axisymmetric disturbances* J. Fluid Mech. **63** 183-193

Itoh N. (1974a) *A power series method for the numerical treatment of the Orr-Sommerfeld equation* Trans. Japan Soc. Aero. Space Sci. **17** 65-75

Itoh N. (1974b) *Spatial growth of finite wave disturbances in parallel and nearly parallel flows. Part 1. The theoretical analysis and the numerical results for plane Poiseuille flow* Trans. Japan Soc. Aero. Space Sci. **17** 160-174