SSSU 135
ISSN 0140 3818

University
of Southampton

School of
Engineering Sciences

Ship Science

AUTOSUB Propulsion System Investigation: Supplementary Information

CD Fallows

Ship Science Report No 135

January 2005

SSSU 135
ISSN 0140 3818

AUTOSUB Propulsion System Investigation: Supplementary Information

CD Fallows

Ship Science Report No 135

January 2005

AUTOSUB Propulsion System Investigation

Supplementary Information

University of Southampton

School of Engineering Science

Ship Science Department

C.D.Fallows

Report Number : 135 January 2005

CONTENTS

Chapter 1

AUTOSUB REQUIREMENT

SUB-SET EFFECTING PROPULSION SYSTEM DESIGN

1 Operational Performance

1.1 To deploy a variety of scientific instruments ...

Implies:

1.1.1 Specific instruments

1.1.1.1 CTD Package (Stevenson, 1995)

1.1.1.2 Acoustic backscatter package (Stevenson, 1995)

1.1.2 Payload bays:

1.1.2.1 Volume – 500 litres
1.1.2.2 Weight – 100kg
1.1.2.3 Shape - cylindrical
1.1.2.4 Position on the vehicle - forward

1.1.3 Power supplies to payload

1.1.3.1 Frequency – dc (Stevenson, 1995)
1.1.3.2 Voltage – 24V (Stevenson, 1995)
1.1.3.3 Noise
1.1.3.4 Power – 400 W
1.1.3.5 Power factor

1.1.4 Computer interface

1.1.4.1 Lonworks (Stevenson, 1995)

1.1.5 Payload environmental protection

1.1.5.1 Pressure - ambient

1.1.5.2 Temperature -10°C
1.1.5.3 Shock
1.1.5.4 Vibration

1.1.6 Hull shape

1.1.6.1 Windows

1.1.6.2 Orifices

1.1.6.3 Protuberances

1.1.7 Positioning of propulsors and control surfaces - aft

1.1.8 Environmental impact of propulsion system:

1.1.8.1 Noise

1.1.8.2 Chemical emissions

1.1.8.3 Pressure fields

1.1.9 Velocity range

1.1.9.1 Max speed – 3 m/s

1.1.9.2 Min speed – 1 m/s

1.1.10 Platform stability

1.2 Over a defined range and for a defined time ...

Implies:

1.2.1 Cruise speed - up to 2m/s

1.2.2 Endurance – 8 days (Stevenson, 1995))

1.2.3 Range – 1000 km

1.3 At a specified position or following a specified track ...

Implies:

1.3.1 Depth keeping – standard deviation of 5 cm.

1.3.2 Track keeping - 1% (improving to 0.1%) of distance travelled (within 200m of sea bed).

1.3.3 Linear and angular acceleration range

1.3.4 Max dive and surface angle – 2.9 degrees to horizontal - (Stevenson, 1995)

1.4 Throughout a defined set of geographical areas:

a) *Coastal*
b) *Shelf seas including operation in very shallow water Shelf edge*
c) *Deep ocean*
d) *Mid ocean ridges*
e) *Operation under ice...*

Implies:

1.4.1 Vehicle range – 1000km.

1.4.2 Depth range

1.4.2.1 Close to the sea surface – from 2m to 20m

1.4.2.2 Close to the sea bed - from 2 m to 200 m above the sea bed at depths up to:

currently 1,000 m. (200m (Stevenson, 1995))
to be extended to 4,000m.

1.4.3 Environmental range

Temperature - tropic (40C) to arctic under ice (-4C)
Salinity/density – fresh water to high salinity
Ability to operate under ice
Navigation using Doppler tracking off underside of ice.
Homing system.
Accurate energy reserve monitoring.

1.5 In a range of weathers

1.5.1 Sea state for launch, recovery and operation

Implies:

1.5.1.1 Weight – dry – 1500kg (Stevenson, 1995)
1.5.1.2 Shape
1.5.1.3 Handling equipment

1.6 From a variety of platforms

Implies:

1.6.1 Ability to launch in a sea way
1.6.2 Launching from shore
1.6.3 Launch from a helo.
1.6.4 Operation from small boats

1.7 World Wide

Implies:

1.7.1 Transportability

1.8 Hotel Load

Implies:

1.8.1 Additional energy storage
1.8.2 Additional power supply distribution
1.8.3 Power supply conditioning

2 Operation

2.1 To be deployable from small non specialised ships, helos and from land.

Implies:

2.1.1 Size
2.1.2 Weight
2.1.3 Shape

2.2 Reliably

Implies:

2.2.1 Mean time between failures

2.2.2 Emergency

2.2.1 Power

2.2.2 Reversing

2.3 With minimal turn round time between operations

Implies:

2.3.1 Ease of:

2.3.1.1 Reprogramming
2.3.1.2 Refuelling
2.3.1.3 Replacement of consumables
2.3.1.4 Servicing
2.3.1.5 Repair

2.4 By a small crew:

Implies:

2.4.1 Simple Man Machine interface

3 Economic

3.1 Acquisition cost

3.2 Life cycle costs

- 3.2.1 Operating costs
- 3.2.2 Maintenance
- 3.2.3 Repair
- 3.2.4 Training

3.3 Disposal costs
3.4 Environmental impact

Chapter 2

AUV PROPULSION SYSTEM PERFORMANCE PARAMETRIC MODEL

Matlab script for estimating AUV propulsion system performance

```
% system_performance

%  a) Produces estimates of AUV propulsion system performance based on
%  set of fundamental parameters that define the system.
%  b) Eables assesments of sensitivity to the values of these parameters
%  c) Plots the results.

%
______________________________________________________________________

clear
close all

% Define environment

  % Density of seawater
rhosw=1025;

%_____________________________________________________________________

% Define system using AUTOSUB parameters

  % Total volume - m^3
  vt=3.7;

  % Cruise speed range - m/s
  u=2;

  % Payload characteristics, volume -m^3, mass - kg (and hence density) and
  % power - W
  vpl=1;
  mpl=100;
  rhopl=mpl/vpl;
  Ppl=150;

  % Energy source, mass density Wh/kg, converted to Ws/kg and to volume
  % density Ws/m^3. Mass density is nominally 150 Wh/kg. This is modified by a
```

```
   % factor of 50% to allow for volume used for the hotal load and for a
   % packing factor of <100%.
   rhoe1=150;
   rhoe=rhoe1*3600;
   rhoe=rhoe*rhosw

   % Propulsion train, volume power density W/m^3, motor efficiency %,
   % propeller efficicency %
   rhop=1000;
   effm=0.7;
   effp=0.75;

   % Hull, drag coefficient
   Cd=(0.01:0.0001:0.09);

   % Hotel load, volume m^3, power W
   vh=1;
   Ph=100;

%________________________________________________________________

% Propulsion force (N) required to propel hull at speed u (m/s)

Fd=0.5*vt^(2/3)*Cd*u.^2*rhosw;

plot(Cd,Fd)
xlabel('Speed - m/s')
ylabel('Drag force - N')

% Power required for propulsion (W)

Pp=(Fd.*u)./(effm.*effp);
figure
plot(Cd,Pp)
xlabel('Speed - m/s')
ylabel('Power consumption - W')

% Total power required (W)

Pt=Pp+Ppl+Ph;
hold
plot(Cd,Pt,'r');
legend('Propulsion power','Total power',2)


% Volume required for propulsion system

vp=Pp/rhop;
figure
plot(Cd,vp)
```

```
xlabel('Speed - m/s')
ylabel('Volume useage - m^3')

% Volume available for energy supply - m^3

ve=vt-vpl-vh-vp;
hold
plot(Cd,ve,'r')
xlabel('Speed - m/s')
ylabel('Volume - m^3')
legend('Required for propulsion','Available for energy',0)

% Energy available - kWhrs

E=(rhoe*ve)/(10^3*3600);
figure
plot(Cd,E)
xlabel('Speed - m/s')
ylabel('Energy carried - kWhr')

% Mission duration - hrs

T=1000*E./Pt;
figure
plot(Cd,T)
xlabel('Speed - m/s')
ylabel('Maximum mission duration - hrs')

% Range - km

R=u.*T*3600/1000;
figure
plot(Cd,R)
xlabel('Speed - m/s')
ylabel('Maximum Range - km')
```

Chapter 3

PROPELLER PERFORMANCE ASSESSMENT

Matlab script for plotting propeller performance

```
% Propefl

% Interpolates between the data points
%       and plots the open water efficiency of the white propeller
%       tables 5,6,7 & 8 of DERA report CRP/11/131

% Speed of advance of the propeller Va m/s:

Va=[1.32 2 2.64 4];

% Propeller rotation rate in rpm:

n=[250 300 350 400 450 500 550 600];

% Torque applied to the propeller having made corrections for shafting and hub
effects:
Q=[3.05 7.7 16.17 24.95 28.13 32.95];

% Efficiency as calculated by DERA from experimental results,
%       each column  corresponding to a value of Va
%       each row to n
%       (where no value is available and where efficiency is >1, or -ve, zero is
entered).

eff=[0.471 0    0    0;
   0.439 0    0    0;
   0.437  0.462 0    0;
   0.426  0.586 0    0;
   0.405  0.505 0.371 0;
   0.382  0.557 0.549 0;
   0      0.475 0.536 0;
   0      0     0.536 0.411];

 %      Interpolate between data points.

[xi,yi]=meshgrid(1:0.1:4, 250:10:600);
zi=interp2(Va,n,eff,xi,yi,'cubic');

% plot in 3 dimensions, x=Va, y=n, z=eff.
```

```
surf(xi,yi,zi)
xlabel('Va')
ylabel('n rpm')
zlabel('Efficiency')
title('Open Water Propeller efficiency - White blades - Interpolated')
shading interp

figure
contour(xi,yi,zi,50)
[c,h]=contour(xi,yi,zi,50)
clabel(c,h,'manual')
xlabel('Va')
ylabel('n rpm')
title('Open Water Propeller efficiency - White blades - Interpolated')
axis([1.4,4,250,600])
```

Chapter 4

STATISTICAL ANALYSIS

References: [Statistics An Introductory Analysis – Taro Yamane – 3rd End – Harper International Edition, 1973
Introduction to mathematical Statistics – P G Hoel - 4th Edn – Wiley International, 1972, ISBN 0-471-40365-2]

4.1 The problem

As explained in the section on deign of the experiments, much information is sought from little data. The experiments have been conceived on the basis that the data conforms to a simple linear model:

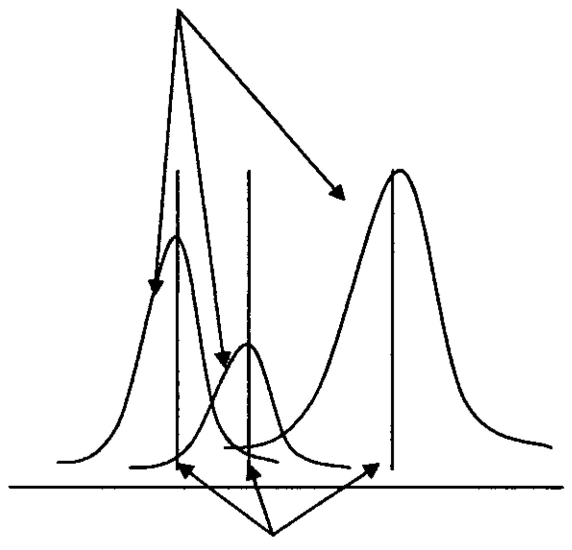
$$D = bX + \varepsilon$$

Where D is the data matrix, b a vector of coefficients and ε a vector containing the random errors resulting from uncontrolled influences. Any interactions between factors are treated as separate parameters in b and assumed to be additive.

The objective of the data analysis is to derive the vector of coefficients, b, from the data D, taking account of the errors, ε, such that it can be stated with a quantified degree of confidence.

4.2 Variance in samples and populations

The experiment is constructed such that the influence of the factors may be derived from the means of defined sub sets of the data. The noise in the signal will result in the means being corrupted. It is assumed that the noise results form random events and that, therefore, the distribution of the noise is normal. The degree of influence of each of the factors is unknown initially but it is possible that some will be small or non-existant.



The issue is, therefore, to determine the probability that the measured means are due to true influences rather than random disturbances in the signal. The variability of the population may be measured by the degree to which each of the data points differs from the mean. To remove the cancelling effect of simple addition the difference is squared, and to remove the effect of the size of the sample, divided by the number of data points in the sample n. The variance is, therefore, given by:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}$$

The root of this quantity, termed the standard deviation, produces a measure of variability in the same units as the original measurement.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}}$$

Now for a normal distribution, >90% of the data points will lie within +/- 2σ of the mean and >99% within +/- 3σ. We, therefore, now have a means of expressing whether the difference between two or more measured means is likely to be due purely to the random nature of the signal.

Now the variation of the sum or difference of two independent random variables is equal to the sum of their variances. (For the purposes of this argument independent means that the variables are not correlated. This is a fundamental assumption of orthogonal array based experiments). Thus:

$$Var(\bar{x}_1 - \bar{x}_2) = \frac{\sigma_1^{\,2}}{n_1} + \frac{\sigma_2^{\,2}}{n_2}$$

The standard deviation of the distribution of the means of the samples is referred to as the standard error of the difference:

$$Std\ error\ of\ dif = \sqrt{\frac{\sigma_1^{\,2}}{n_1} + \frac{\sigma_2^{\,2}}{n_2}}$$

For large data sets the errors inherent in this approach are insignificant, as will be evident shortly. However, in this case we are necessarily constrained to small samples. We, therefore, need to take account of the fact that, whereas the mean value of the sample, n, is an unbiased estimate of the mean of the population, the standard deviation of the sample tends to underestimate that of the population from which the sample is drawn. The underestimate becomes greater the smaller the sample. This is due to the fact that, whereas the mean is derived using all sample data, the variance is derived from the difference between sample points and, therefore, is based on one fewer samples. The expected value of the variance in a sample is, therefore, related to the population variance by Bessel's correction:

$$E(s^2) = \left(\frac{n-1}{n}\right)\sigma^2$$

Clearly if n is large:

$$E(s^2) \approx \sigma^2$$

However, the smaller the sample size, n, becomes, the larger the difference between the expected and sampled values.

It will be noted that we have now introduced three separate measures of variance:

σ^2 = population variance

s^2 = sample variance

$\hat{\sigma}^2$ = best estimate of population variance.

These are related by:

$$\hat{\sigma}^2 = \frac{n}{n-1}s^2 \tag{1}$$

Thus, if we have k samples, our best estimate of the population variance, based on pooling all of the samples is:

$$\hat{\sigma}^2 = \frac{n_1 s_1^{\ 2}}{n_1 - 1} + \frac{n_2 s_2^{\ 2}}{n_2 - 1} + \ldots + \frac{n_k s_k^{\ 2}}{n_k - 1}$$

$$\hat{\sigma}^2 = \sum_{i=1}^{k} \frac{n_i s_i^{\ 2}}{n_i - 1}$$

Where $\sum n_k - k$ is the number of degrees of freedom.

4.3 Estimate of means

Now we wish to test the hypothesis that the means of the data samples result from significant differences in the identified factors rather than due to random disturbances in the signal. Due to random variability the mean of the sample will differ from that of the population as whole, resulting in an error. Now the distribution of the error can be expressed as a ratio of the error of the mean to the standard error of the mean:

$$t = \frac{\left|\overline{X} - \overline{x}\right|}{\sqrt{\frac{\sigma^2}{n}}}$$

Now the best estimate of σ is given by equation (1). Therefore:

$$t = \frac{\left|\overline{X} - \overline{x}\right|\sqrt{n-1}}{s}$$

From this it can bee seen that as the difference between the estimate of the mean and the population mean grows larger so does *t*. The distribution of *t* has been calculated to give the maximum vale of t for a given probability that the apparent difference between the means of the sample and population is due to causes other than noise. The probability that the mean of a sample is a real reflection of the population may, therefore, be established and a judgement made as to whether this is significant in the context of the system being described. It should be noted that the decision that the factor is not significant does not mean that it effects no influence, merely that its influence is not established. Equally a decision that the effect is significant is no guarantee that it really does have an influence.

4.4 Comparison of variances

Now since the t-test is dependent upon a pooled estimate of variance it contains an inherent assumption that the population variance is equal for all of the populations considered in the experiment. It is, therefore, necessary to establish that the variances of the samples are sufficiently similar to warrant this assumption.

Now if the ratio of the variances of the populations being compared is 1, then the condition is definitely met. However, the greater the difference from 1 the less the likelihood of them having the same variance is. The way in which this ratio is distributed, termed the F distribution, as a function of the probability that the two populations have the same variance, is known. So a comparison of the F ratio (defined as the ratio of the greater estimate of variance to that of the lesser) with appropriate tables will reveal the probability that the underlying populations have the same variance. F tables provide the value of F which will be exceded for a defined probability for a given sample size., where sample size is expressed in terms of the number of degrees of freedom of the lesser and greater sample sizes.

4.5 Analysis of Variance

This builds on the F-test to establish the possible sources of variability within a set of data and their relative significance.

Now variance $$V = \sigma^2$$

And the best estimate of the variance of from the sample is

$$\hat{\sigma}^2 = \frac{\sum_{k=1}^{k} n_k s_k^{\ 2}}{\sum_{k=1}^{k} n_k - k}$$

Now even if each sample came from the same population we would not expect their means to be identical. Rather we would expect them to vary in accordance with the population from which they are drawn. Now we can determine the variance of each sample, and from the expression for pooled variance determine the estimate that we would expect from these samples for there to exist in the total population. We can, therefore, compare the 'between sample variance' with the 'within sample variance. If these should prove to be significantly different we could deduce that the samples are from populations whose means do in fact differ.

The total population sum of squares is

$$SS_t = (x - \overline{X})^2$$

The sum of squares between any sample, *k*, and the total population is is:

$$SS_k = (\overline{x}_k - \overline{X})^2$$

Since the number of sample means is k, there are k-1 degrees of freedom for the total sum of squares 'between samples'.

To calculate the sum of squares 'within samples' we now need to remove the 'between samples' average effect, i.e.

$$SS_w = (x - \overline{x})^2 - (mean(x - \overline{x}))^2$$

Where, $(mean(x - \overline{x}))^2 = 0$

the number of degrees of freedom for the 'within samples' sum of squares is the sum of the degrees of freedom of each sample separately, i.e.:

$$= \sum_{k=1}^{k} (n_k - 1) = \sum_{k=1}^{k} n_k - k$$

The F ratio of the within samples variance to the between samples variance, therefore, indicates the significance of the mean of the samples. If the derived f-ratio is > than that determined from the table for the confidence level sought then the samples are likely to be from different populations and the factor to be significant at the confidence level selected.

4.6 Data Outliers

Ref (Coleman H W and Steele W G, 1999 'Experimentation and Uncertainty Analysis for Engineers', 2nd edition, John Wiley and Sons, Inc, New York.).

Inevitably in any data set there will be outliers resulting from a rare anomaly in the measurement system such as it experiencing a power spike. In general such outliers can be expected to have little effect on the mean of a sample but to have a considerable effect on its variance. Thus, the effect of outliers, if not removed, is to reduce the confidence in the estimate of effect. Chauvenet (Ref above) has devised a test for identifying outliers as follows.

For a sample of size *n*, the data point is rejected if

$$|x_i - \bar{x}| \geq \tau S_x$$

where a curve fit equation for τ is

$$\tau = \sum_{i=1}^{5} a_i [\ln(n)]^i$$

$a_0 = 0.720185$

$a_1 = 0.674947$

$a_2 = -0.0771831$

$a_3 = 0.00733435$

$a_4 = -0.00040635$

$a_5 = 0.00000916028$.

Chapter 5

DRAG AS A FUNCTION OF VELOCITY AND ANGLE-OF-ATTACK FOR A SHALLOW SUBMERGED BODY OF SIMPLE SHAPE

5.1 Introduction

Drag is the force that has to be overcome to allow a body to pass through a fluid.

A neutrally buoyant, axi-symetric, deeply submerged body, traveling at constant speed and subject to no angular accelerations, is subject only to viscous drag. This drag results directly from the viscosity of the water. Viscosity implies that shear forces need to be applied to it to allow progress of the vehicle, and that unequal pressure distribution which result as a consequence of deformation of the boundary layer. The viscous drag force is a function of the kinematic viscosity of the liquid, the velocity of the vehicle relative to that of the liquid and the size and shape (form) of the vehicle.

The addition of a safety margin of positive buoyancy requires the continuous application of dynamic negative lift that is obtained from both purpose built lifting surfaces and from the hull traveling at an angle incident to its direction of motion (known as the angle-of-attack). The lift force is raked aft with respect to a line perpendicular to the direction of motion. It will Thus, have a horizontal component opposing the direction of motion, which will manifest itself as additional drag.

Should the submarine be moving sufficiently close to the free surface then its motion will induce a wave system at the surface, which will absorb energy from the vehicle. This will appear as further drag.

Finally, should the body travel in a narrow, shallow channel it will experience a blockage effect. This is a consequence of the channel constraining the motion of the fluid past the hull. It results in an increase in the amplitude of the wave generated at the surface, increasing the drag still further.

5.2 AUTOSUB physical modeling parameters

Water in tank:

Density	$\rho = 1000$ kg/m^3
Kinematic Viscosity (15 deg C)	$\nu = 1.19x10^{-6}$ m^2/s

Acceleration due to gravity	$g = 9.8$ m/s^2

Model dimensions:

Volume	$V = 0.184$ m^3
Length	$l = 2.5$ m

Model Speed range:	$u = 2.7$ to 4.4 m/s
Angle-of-attack range:	$\alpha = 1$ to 10 deg

5.3 Viscous drag

The ITTC '57 friction line for the viscous forces acting on an immersed body (termed friction drag) is widely used as a reliable empirical estimator of total viscous drag for surface ships:

$$C_f = \frac{0.075}{(\log(R_e) - 2)^2}$$

Where: $$C_f(u) = \frac{F_{dv}}{\frac{1}{2}\rho u^2},\ R_n(u) = \frac{ul}{\nu},\ S = V^{\frac{2}{3}}.$$

Theoretical studies undertaken by Hearn and Murphy (Hearn and Murphy, 2001/2) indicate that the ITTC '57 formula provides an accurate estimate of the drag for submerged thin plates. By extrapolation it should provide a reasonable first estimation of the viscous drag of a more complex shape such as that of an AUV.

The coefficient of friction drag as a function of R_e is given in Figure 2.3.1.1 and that of friction drag force as a function of speed in Figure 2.3.1.2.

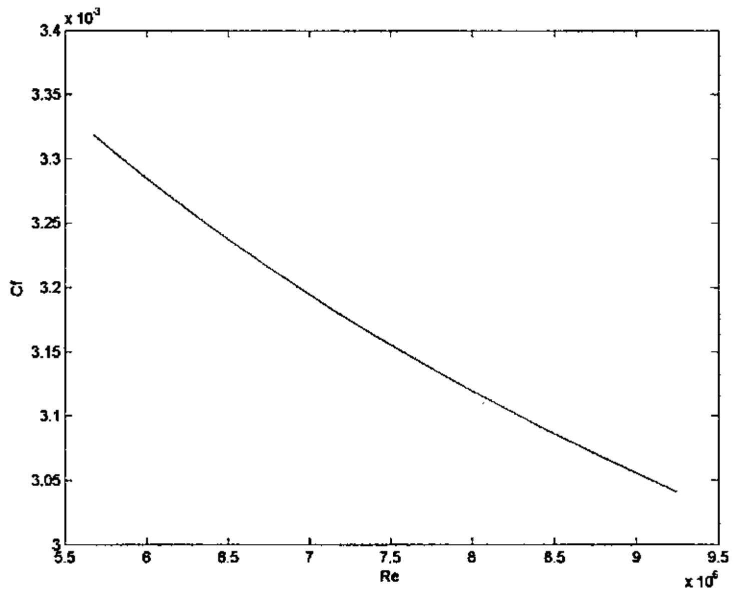


Figure 2.3.1.1 Predicted Coefficient of friction drag

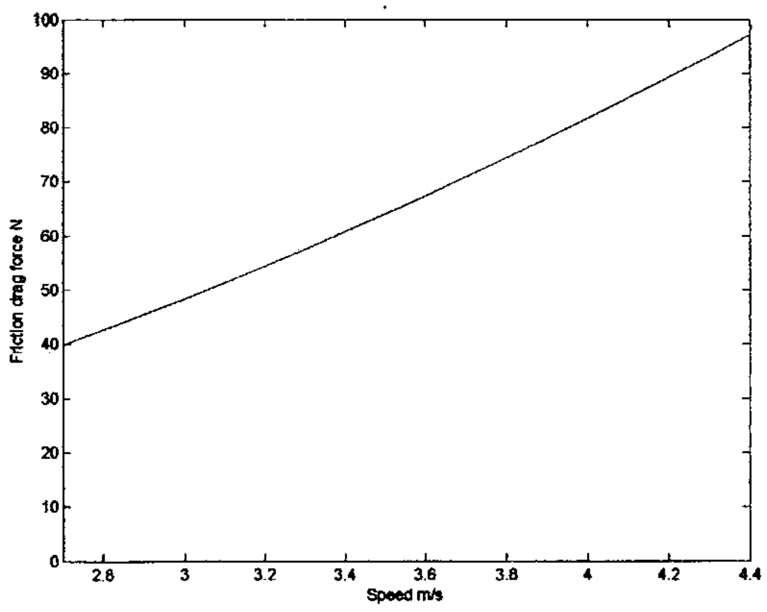


Figure 2.3.1.2 Drag force

5.4 Lift induced drag

An axi-symetric body traveling horizontally through a fluid experiences no lift. Should the body be none-axi-symetric then a lift force will be generated by thickening of the boundary layer at the aft end of one of the surfaces. AUTOSUB in its clean form is axi-symetric and will, therefore, experience no such lift. The addition of asymmetric detail may change this.

A second source of lift occurs when the body is traveling at an angle-of-attack, α. The flow over the upper and lower surfaces becomes unequal. This results in different velocity, and hence pressure, distributions along the upper and lower surfaces, which will result in a transverse lift force. For small angles of attack the lift force is expected to be directly proportional to α.

This lift force will be perpendicular to the average direction of flow and is, therefore, tilted back by a function proportional to α. It will, therefore, have a horizontal component that will appear as drag. This horizontal component motion will, therefore, be proportional to sin α.

For any given velocity, therefore, the drag force due to lift is expected to be of the form:

$$F_{dl}(\alpha) = k_1 \frac{\alpha}{2\pi} \sin\left(\frac{\alpha}{2\pi}\right).$$

To establish the shape of the curve, and assuming drag due to lift is an order of magnitude less than friction drag, let:

$$k_1 = 10.$$

The shape of the drag force response to of angle-of attack is given in Figure 2.3.1.3.

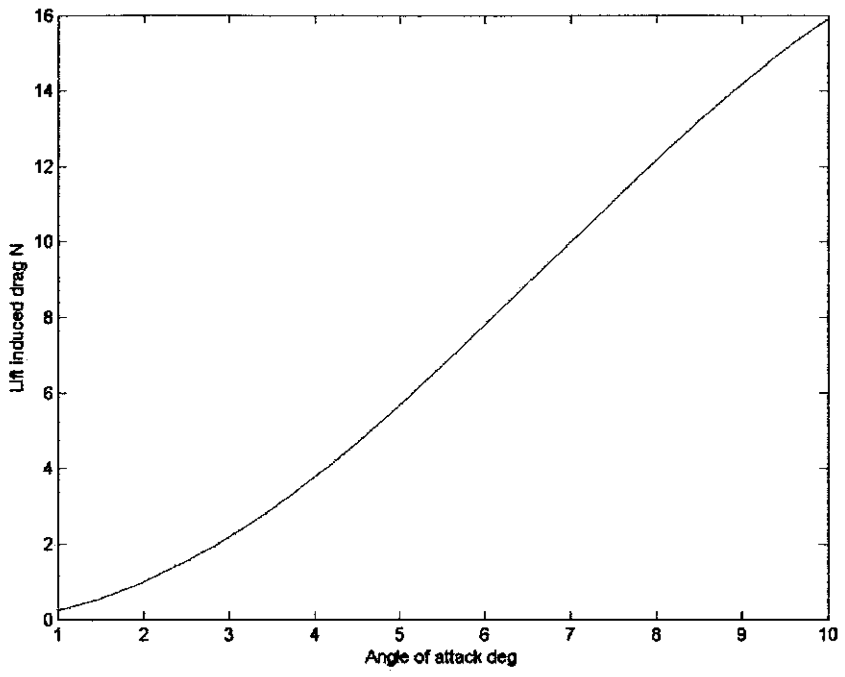


Figure 2.3.1.3 Lift induced drag

5.5 Wave Induced Drag

When traveling close to the surface, the pressure distribution along the hull (low at bow and stem and high along the mid section) induces waves in the free surface. The energy required to generate these waves will appear as additional drag.

The waves originate at the bow and stem of the vehicle and propagate fore and aft. The wavelengths and amplitudes will be proportional to the velocity of the vessel. The bow and stem waves will, therefore, constructively and destructively interfere dependent upon the speed of the vehicle. The amount of energy absorbed will Thus, be a periodic function of the velocity of the vehicle but with the last significant hump in the region of a Froude Number, $Fn = 0.45$ (Hoerner, 1965)to 0.5 (Comstock, 1980).

Where $$F_n(u) = \frac{u}{\sqrt{gl}}$$

The full-scale vehicle runs in a velocity range corresponding to Fn = 0.2 to 0.4. However, the model experiment is designed to run at constant Reynolds Number. The model will, therefore, travel at a very much higher Froude Number (in the region of 0.5 to 0.9). Such data as there is for the wave-making resistance at these high Fn's applies to surface ships, mainly with planing hulls. Clearly these will not apply to a submarine, but it seems likely that the coefficient of wave-making resistance will have a similar form, i.e. decrease with *Fn* to an asymptote at *Fn* = 1. The curve would appear to be cubic or higher, but with only a small third order term.

However, wave-making effect has a maximum on the surface and decreases rapidly with increase in depth. (Hoerner, 1965 p 8-11) indicates that in open water wave-making resistance below 5 diameters may be ignored. The model will run at a depth of approximately 2.3 hull diameters.

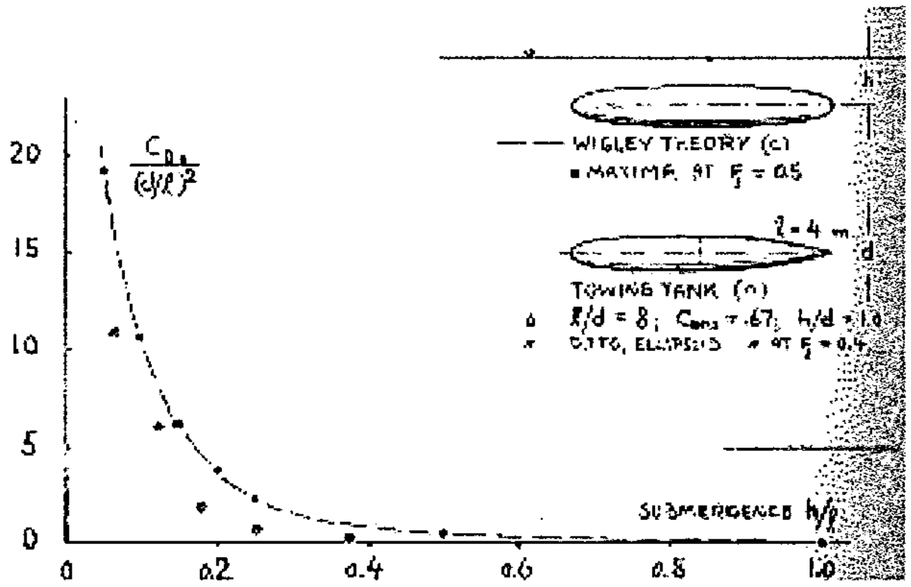


Figure 70. Wave drag coefficient of submerged streamlined bodies (Hoerner, 1965) p 11 – 18

Nevertheless, since this experiment will be run at a height to length ratio of 2.8, figure 70, indicates that the Coefficient of Drag due to wave-making will be very small (although such as there is will be compounded by drag resulting from proximity to the bottom and sides of the tank).

5.6 Net Drag

The net drag force will be the sum of the three components across the velocity, angle-of-attack plane.

Chapter 6

Orthogonal Arrays

The orthogonal arrays used in this thesis are reproduced here for completeness (Taguchi, 1988).

1. Table 2.3.2.1 L_9 array
2. Table 2.3.2.2 L_{16} array
3. Table 2.3.2.3 L_{18} array
4. Table 2.3.2.4 L_{25} array

Experiment No.	Column 1	Column 2	Column 3	Column 4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

Table 6.1 L_9 Orthogonal Array

Experiment No.	Column 1	Column 2	Column 3	Column 4	Column 5
1	1	1	1	1	1
2	1	2	2	2	2
3	1	3	3	3	3
4	1	4	4	4	4
5	2	1	2	3	4
6	2	2	1	4	3
7	2	3	4	1	2
8	2	4	3	2	1
9	3	1	3	4	2
10	3	2	4	3	1
11	3	3	1	2	4
12	3	4	2	1	3
13	4	1	4	2	3
14	4	2	3	1	4
15	4	3	2	4	1
16	4	4	1	3	2

Table 6.2 L_{16} Orthogonal Array

Column	1	2	3	4	5	6	7	8
Expt.								
1	1	1	1	1	1	1	1	1
2	1	1	2	2	2	2	2	2
3	1	1	3	3	3	3	3	3
4	1	2	1	1	2	2	3	3
5	1	2	2	2	3	3	1	1
6	1	2	3	3	1	1	2	2
7	1	3	1	2	1	3	2	3
8	1	3	2	3	2	1	3	1
9	1	3	3	1	3	2	1	2
10	2	1	1	3	3	2	2	1
11	2	1	2	1	1	3	3	2
12	2	1	3	2	2	1	1	3
13	2	2	1	2	3	1	3	2
14	2	2	2	3	1	2	1	3
15	2	2	3	1	2	3	1	3
16	2	3	1	3	2	3	2	1
17	2	3	2	1	3	1	2	3
18	2	3	3	2	1	2	3	1

Table 6.3 L_{18} Orthogonal Array

Experiment No.	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
1	1	1	1	1	1	1
2	1	2	2	2	2	2
3	1	3	3	3	3	3
4	1	4	4	4	4	4
5	1	5	5	5	5	5
6	2	1	2	3	4	5
7	2	2	3	4	5	1
8	2	3	4	5	1	2
9	2	4	5	1	2	3
10	2	5	1	2	3	4
11	3	1	3	5	2	4
12	3	2	4	1	3	5
13	3	3	5	2	4	1
14	3	4	1	3	5	2
15	3	5	2	4	1	3
16	4	1	4	2	5	3
17	4	2	5	3	1	4
18	4	3	1	4	2	5
19	4	4	2	5	3	1
20	4	5	3	1	4	2
21	5	1	5	4	3	2
22	5	2	1	5	4	3
23	5	3	2	1	5	4
24	5	4	3	2	1	5
25	5	5	4	3	2	1

Table 6.4 L_{25} Orthogonal Array

Chapter 7

SIGNAL PROCESSING

7.1 Hamming window

The Hamming window is a raised cosine of the form

$$w = a - b\cos\left(\frac{2\pi n}{N-1}\right) \quad 0 \le n < N\text{-}1$$

It is illustrated in Figure 2.5.1.1

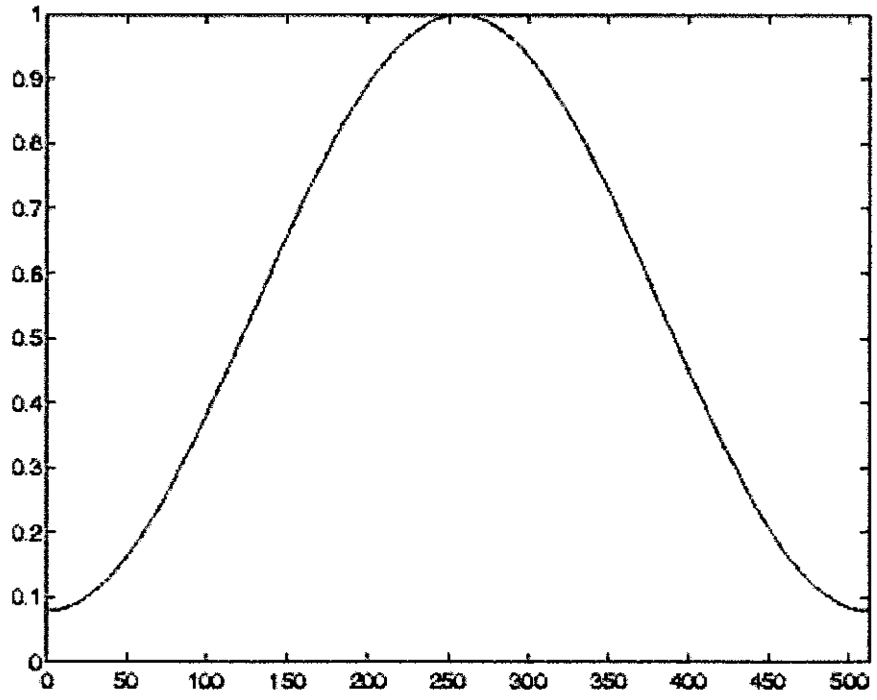


Figure 7.1 Form of the Hamming window

7.2 Periodogram

The periodogram represents an estimate of the power spectrum. For a ssignal defined by [x_1, ... , x_n], it is given by the following formula:

$$S(e^{j\omega t}) = \frac{1}{n}\left|\sum_{l=1}^{n} x_j e^{-j\omega t}\right|^2$$

If the signal sequence is weighted by a window $[w_1, \ldots, w_n]$, then the weighted or modified periodogram is defined as:

$$S(e^{j\omega t}) = \frac{\frac{1}{n}\left|\sum_{l=1}^{n} w_l x_l e^{-j\omega t}\right|^2}{\frac{1}{n}\sum_{l=1}^{n}|w_l|^2}$$

A DFT is used to compute the power spectral density as $\frac{S(e^{j\omega t})}{f_s}$, where f_s is the sampling frequency.

Chapter 8

DATA PRE-PROCESSING FUNCTIONS

8.1 Matlab function for data visualisation

```
function [frd_drag,frd_SF,aft_SF,aft_drag]=Visualise07(fname)

%'Visualise' enables the raw .WAD data file 'fname' to be seen graphically.

% Load data
[channelData,channelRate,channelZero,sampleRate]=frunData_from_wad(fname);
frd_drag=channelData(:,1);
frd_SF=channelData(:,2);
aft_SF=channelData(:,3);
aft_drag=channelData(:,4);

%Convert bits to 4 vectors representing data for each channel in physical units.

frd_drag=((frd_drag-2048)/channelRate(1,1))+channelZero(:,1);
frd_SF=((frd_SF-2048)/channelRate(1,2))+channelZero(:,2);
aft_SF=((aft_SF-2048)/channelRate(1,3))+channelZero(:,3);
aft_drag=((aft_drag-2048)/channelRate(1,4))+channelZero(:,4);

% Plot force block o/p's.

figure
plot(frd_drag);
title('Force Block Signals')
xlabel('Sample Number')
ylabel('Force - N')
hold
plot(aft_drag,'g+')
plot(frd_SF,'c.')
plot(aft_SF,'b-')
legend('Frd Drag','Aft Drag','Frd SF','Aft SF')
```

2.6.1.2 Matlab function for record truncation

```
function [frd_drag,frd_SF,aft_SF,aft_drag,mean_frd_drag,mean_frd_SF,...
     mean_aft_SF,mean_aft_drag,std_frd_drag,std_frd_SF,std_aft_SF,...
     std_aft_drag,force_means]=Truncate07(fname,n,m)

% Function 'Truncate2' uis for use in calculating mean forces under the steady state
condition.
```

```
% It only includes the data from raw .WAD data file 'fname' from the 'n'th to the
'mth' data
% points and calculates mean and variance of this data.

% Load data and create new data vectors omiting data to the nth data entry.
[channelData,channelRate,channelZero,sampleRate]=frunData_from_wad(fname);
frd_drag=channelData(n:m,1);
frd_SF=channelData(n:m,2);
aft_SF=channelData(n:m,3);
aft_drag=channelData(n:m,4);

% Test plot to show truncation successful.

plot(channelData(:,1))

figure
plot(channelData(:,2))
hold
plot(frd_drag,'r')
hold off

%Convert bits to 4 vectors representing data for each channel in physical units.

frd_drag=((frd_drag-2048)/channelRate(1,1))+channelZero(:,1);
frd_SF=((frd_SF-2048)/channelRate(1,2))+channelZero(:,2);
aft_SF=((aft_SF-2048)/channelRate(1,3))+channelZero(:,3);
aft_drag=((aft_drag-2048)/channelRate(1,4))+channelZero(:,4);

% Test plot
figure
plot(frd_drag)
hold
plot(frd_SF,'g')
plot(aft_SF,'r')
plot(aft_drag,'c')
title('Force Blocks')
legend('Frd Drag','Frd SF','Aft SF','Aft Drag')

%--------------------------------



% Calculate new mean and SD for each data channel.
mean_frd_drag=mean(frd_drag);
mean_frd_SF=mean(frd_SF);
mean_aft_SF=mean(aft_SF);
mean_aft_drag=mean(aft_drag);
std_frd_drag=std(frd_drag);
std_frd_SF=std(frd_SF);
std_aft_SF=std(aft_SF);
```

```
std_aft_drag=std(aft_drag);

% Generate vector of force means.

force_means=[mean_frd_drag,mean_frd_SF,mean_aft_SF,mean_aft_drag]';
```

Chapter 9

ANALYSIS OF SPEED DATA

9.1 Statistics

All Experiments - Speed Statistics

Speed Dial	Max	mean	Min	sigma	+3sigma	-3sigma	Regression	% error
	m/s	m/s	m/s	m/s	m/s	m/s		
0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.0
120	0.508	0.501	0.495	0.007	0.522	0.480	0.503	0.3
150	0.635	0.635	0.635	0.000	0.635	0.635	0.622	2.1
240	1.020	1.002	0.992	0.013	1.040	0.964	0.968	3.4
290	1.189	1.189	1.189	0.000	1.189	1.189	1.157	2.7
300	1.226	1.226	1.226	0.000	1.226	1.226	1.194	2.6
340	1.399	1.384	1.377	0.013	1.423	1.346	1.343	3.0
345	1.396	1.396	1.396	0.000	1.396	1.396	1.362	2.4
360	1.460	1.459	1.455	0.002	1.466	1.451	1.418	2.8
400	1.613	1.612	1.612	0.000	1.613	1.611	1.568	2.7
455	1.836	1.836	1.836	0.000	1.836	1.836	1.776	3.2
470	1.892	1.892	1.892	0.000	1.892	1.892	1.834	3.1
480	1.943	1.934	1.928	0.004	1.946	1.922	1.872	3.2
488	1.968	1.965	1.961	0.002	1.971	1.958	1.903	3.1
500	2.020	2.020	2.020	0.000	2.020	2.020	1.950	3.5
540	2.203	2.196	2.185	0.008	2.220	2.173	2.108	4.0
555	2.246	2.246	2.246	0.000	2.246	2.246	2.168	3.5
600	2.451	2.449	2.444	0.003	2.457	2.441	2.352	3.9
610	2.492	2.488	2.486	0.002	2.493	2.483	2.394	3.8
650	2.676	2.675	2.673	0.002	2.681	2.668	2.564	4.1
659	2.714	2.714	2.713	0.000	2.714	2.713	2.603	4.1
660	2.724	2.717	2.703	0.004	2.728	2.706	2.608	4.0
700	2.906	2.906	2.905	0.001	2.908	2.903	2.786	4.1
720	3.019	3.010	2.995	0.009	3.038	2.982	2.878	4.4
732	3.073	3.056	3.043	0.008	3.081	3.031	2.934	4.0
750	3.146	3.138	3.131	0.007	3.158	3.118	3.019	3.8
775	3.262	3.256	3.249	0.005	3.272	3.240	3.140	3.6
780	3.319	3.288	3.233	0.011	3.321	3.255	3.164	3.8
840	3.629	3.610	3.603	0.011	3.644	3.576	3.470	3.9
850	3.674	3.664	3.651	0.009	3.690	3.637	3.523	3.8
875	3.799	3.783	3.701	0.022	3.849	3.717	3.659	3.3
900	3.958	3.949	3.933	0.014	3.991	3.907	3.798	3.8
915	4.076	4.035	4.016	0.011	4.068	4.001	3.884	3.7
924	4.076	4.068	4.063	0.005	4.083	4.054	3.936	3.3
950	4.233	4.233	4.233	0.000	4.233	4.233	4.090	3.4
960	4.345	4.333	4.320	0.010	4.364	4.302	4.150	4.2
966	4.334	4.331	4.327	0.003	4.340	4.323	4.187	3.3
976	4.477	4.426	4.405	0.020	4.486	4.366	4.249	4.0
999	4.604	4.572	4.556	0.012	4.608	4.536	4.394	3.9

Regression % Error stats	
Mean	3.327625
Standard Error	0.147933
Median	3.479174
Mode	#N/A
Standard Deviation	0.92384
Sample Variance	0.853481
Kurtosis	5.813918
Skewness	-2.192157
Range	4.398261
Minimum	0
Maximum	4.398261
Sum	129.7774
Count	39
Confidence Level(95.0%)	0.299474

Error stats based on mean speed, all data	
Mean	0.3
Standard Error	0.0
Median	0.2
Mode	0.1
Standard Deviation	0.4
Sample Variance	0.2
Kurtosis	5.4
Skewness	2.4
Range	1.8
Minimum	0.0
Maximum	1.8
Sum	192.1
Count	646.0
Confidence Level(95.0%)	0.03

Error stats for mean speed > 1 m/s	
Mean	0.176267
Standard Error	0.007566
Median	0.124964
Mode	0.088154
Standard Deviation	0.179846
Sample Variance	0.032344
Kurtosis	13.57864
Skewness	2.809321
Range	1.675963
Minimum	0
Maximum	1.675963
Sum	99.59071
Count	565
Confidence Level(95.0%)	0.014861

9.2 Matlab speed look-up table function

```
function [u]=look_up_speed(dial)
% LOOK_UP_SPEED returns the vector of mean speeds, u, in m/s corresponding
% to the carriage speed dial settings, dial, for the SI towing tank.

% Speed look up table: Col1 = dial setting, col 2 = corresponding mean speed in m/s.

speed=[0      0.000
120    0.501
150    0.635
240    1.002
290    1.189
300    1.226
340    1.384
345    1.396
360    1.459
400    1.612
455    1.836
470    1.892
480    1.934
488    1.965
500    2.020
540    2.196
555    2.246
600    2.449
610    2.488
650    2.675
659    2.714
660    2.717
700    2.906
720    3.010
732    3.056
750    3.138
775    3.256
780    3.288
840    3.610
850    3.664
875    3.783
900    3.949
915    4.035
924    4.068
950    4.233
960    4.333
966    4.331
976    4.426
999     4.572];

% Corresponding mean speed:
u=interp1(speed(:,1),speed(:,2),dial);
```

Chapter 10

POLE DRAG ANALYSIS

10.1 Matlab script for producing a model of p[ole drag

```
% PoleDrag1

% Produces a model of pole drag by producing a 3d spline interpolation
%  between measured data points from exps 0304 & 0307 data.
%       Plots the drag of the AUTOSUB model mounting poles
%  as a function of speed and angle-of-attack as predicted by this model.

% Angle-of-attack - deg:

aoa=[0 2 7 10];

% Speed look up table: Col1 = dial setting, col 2 = corresponding mean
% speed in m/s.

% Dial setting of speeds for pole drag data:

dial=[0 240 488 610 660 780 915 976];

% Corresponding mean speed:

u=look_up_speed(dial);

% Drag force Fd - N, depth hole 5.
%       each column  corresponding to a value of aoa
%       each row to nominal u.

Fd=[0 0 0 0
18.528 18.205 20.226 20.815
77.612 76.026 96.428 101.521
122.817 123.637 145.824 170.320
147.510 137.059 155.038 177.300
182.026 172.717 186.433 195.321
224.840 226.097 246.902 252.926
261.253 261.493 284.602 295.976];

% Produce coarse mesh plot to indicate shape of data.

[xci,yci]=meshgrid(0:0.2:10, 0:0.1:4.5);
zci=interp2(aoa,u,Fd,xci,yci,'cubic');
figure
mesh(xci,yci,zci);
```

```
xlabel('aoa deg')
ylabel('u m/s')
zlabel('Fd N')
title('Pole Drag - Interpolated')
axis([0,10,0,4.5,0,350])
colorbar

% Produce detailed interpolation between data points and save as tab delimited ascii
% file

[xi,yi]=meshgrid(0:0.1:10, 0:0.01:4.5);
zi=interp2(aoa,u,Fd,xi,yi,'cubic');
save 'interpolated_pole_drag.txt' zi -ascii -tabs

% Check accuracy by returning interpolated values for other speeds for
% which measurements have been made.
%  (NOTE: need to input array address, x,y, e.g. for angle=0 is col 1 and speed
%   = 1 m/s is row 1)

% Convert aoa, a, and speed, u, into array address and return corresponding
% estimated drag to workspace

aa=7;
ua=[3.61 2.44 4.04];

x=round(ua*100)+1
y=(aa*10)+1

drag=zi(x,y)

% Plot interpolated pole drag in 3 dimensions, x=aoa, y=u, z=Fd.

figure
surf(xi,yi,zi);
xlabel('aoa deg')
ylabel('u m/s')
zlabel('Fd N')
title('Pole Drag - Interpolated')
axis([0,10,0,4.5,0,350])
shading interp;
colorbar

figure
contour(xi,yi,zi,50);
[c,h]=contour(xi,yi,zi,50);
clabel(c,h,'manual')
xlabel('aoa deg')
ylabel('u m/s')
title('Pole Drag - Interpolated')
axis([0,10,0,4.5])
```

```
colorbar

% Check accuracy of interpolation by plotting pole drag vs speed for 10 deg
% aoa for both initial and interpolated data.

figure
plot(yi,zi(:,101));
title('Drag of 2 poles at 10 deg aoa, depth 5');
xlabel('speed m/s');
ylabel('drag');
hold;
plot(u,Fd(:,4),'x');
legend('3d spline interpolated and raw data',2);

% Plot Pole drag as a function of speed for a range of aoa's.

figure
aoa=0;
x=(aoa*10)+1;
plot(yi,zi(:,x));
axis([0 4.5 0 300])
xlabel('Speed m/s')
ylabel('Drag Force N')
title('Drag of Poles as a Function of Speed for a range of Angles of Attack')
legend('Angle-of-attack from 0 to 10 deg in 1 deg steps',2)
hold
for aoa=1:10
  x=(aoa*10)+1;
  plot(yi,zi(:,x));
end

% Plot Pole drag as a function of angle-of-attack for a range of speeds.

u=1;
y=(u*100)+1;
figure
plot(xi,zi(y,:));
xlabel('Angle-of-attack, degrees')
ylabel('Drag Force, N')
title('Drag of Poles as a Function of Angles of Attack for a Range of Speeds')
legend('Speed from 1 to 4 m/s in 0.5 m/s steps',2)
hold
for u=1.5:0.5:4
  y=(u*100)+1;
  plot(xi,zi(y,:));
end
```

10.2 Matlab pole drag look-up function

```
function [Fd]=look_up_pole_drag(values)
% LOOK_UP_POLE_DRAG returns the vector of Pole Drag forces defined in
%   file 'interpolated_pole_drag.txt' corresponding to the aoa and speed dial settings
%   defined in cols 1 and 2 respectively of matrix 'values'.

load interpolated_pole_drag.txt
u=look_up_speed(values(:,2));
[xi,yi]=meshgrid(0:0.1:10, 0:0.01:5);
Fd=interp2(xi,yi,interpolated_pole_drag,values(:,1),u)
```

10.3 Matlab script for pole drag accuracy check

```
% accuracy_check1

% Checks that accuracy of the interpolated pole drag prediction against
% force measurements made in the lab, but not used in constructing the
% model.

% Load measured values from 'Pole drag.xls, Accuracy data check sheet'
% stored as 'test.txt'
%   Col 1 = angle-of-attack in deg
%   Col 2 = speed in m/s
%   Col 3 = total measured drag force

load test.txt;
aoa=test(:,1);
u=test(:,2);

% Create matrix of values to look up in pole drag model.

values=[aoa u];
Fd=look_up_pole_drag(values);

% Plot Model vs measured Values

figure
plot(Fd, test(:,3),'*')
hold
x=0:50:300;
plot(x,x)
title('Pole Drag - Predicted vs measured values')
xlabel('Measured force')
ylabel('Predicted force')
axis([0 300 0 300])

% Establish difference between model and measured drag
```

```
F_dif=Fd-test(:,3);
F_dif_pc=(F_dif./Fd)*100

% Plot error against measured force
figure
plot(Fd, F_dif,'*')
title('Pole Drag - Difference between model prediction and measured values ')
xlabel('Measured Drag - N')
ylabel('Difference - N')

% Plot %error against measured force
figure
plot(Fd, F_dif_pc,'*')
title('Pole Drag - % Difference between model prediction and measured values ')
xlabel('Measured Drag - N')
ylabel('% Difference')

output=[test(:,3) Fd F_dif F_dif_pc]
```

Chapter 11

MICHELL THIN-SHIP THEORY MODELLING

A brief outline of the Michell Thin-sip theory follows.

The steady wave pattern $z = \zeta(x, y)$ of a body moving through water at a point in the far field, is of the form of a sum of plane waves travelling at various angles of propagation to the direction of motion of the body, θ. Thus:

$$\zeta(x,y) = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} A(\theta)e^{-i\omega(\theta)}d\theta \qquad (1)$$

Where $\omega(\theta)$ is a phase function defined by:

$$\omega(\theta) = k(\theta)(x\cos\theta + y\sin\theta)$$

and $A(\theta)$ is the amplitude and $k(\theta)$ is the wave number of the wave component travelling at angle θ. Once $A(\theta)$ and $k(\theta)$ are specified then the wave pattern can be derived from Equation (1).

At any given speed, u, the amplitude $A(\theta)$ is a function solely of the shape of the hull and $k(\theta)$ is determined from the dispersion relation for plane waves. For infinite depth of water this is:

$$k(\theta) = k_0 \sec^2(\theta).$$

Where k_0 is the number of transverse waves at $\theta = 0$ and $k_0 = \frac{g}{u^2}$ for deep water (Eggers et al., 1967).

Michell derived the amplitude equation as (Newman, 1977)

$$A(\theta) = \frac{2}{\pi}\left(\frac{g}{u^2}\right)\sec^3 \iint \frac{\partial\zeta}{\partial x}\exp\left[\left(\frac{g}{u^2}\right)\sec^2\theta(y - ix\cos\theta)\right]dxdy.$$

The total energy in the wave field can then be derived from *[Ref: Newman]*:

$$R = \frac{\pi}{2}\rho u^2 \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} |A(\theta)|^2 \cos^3(\theta) d\theta .$$

Michell represents the body by a centre plane source distribution proportional to its longitudinal rate of change of thickness. The reprentation used for the AUTOSUB modelling is given in Figures 2.7.2.1, 2.7.2.2 and 2.7.2.3.

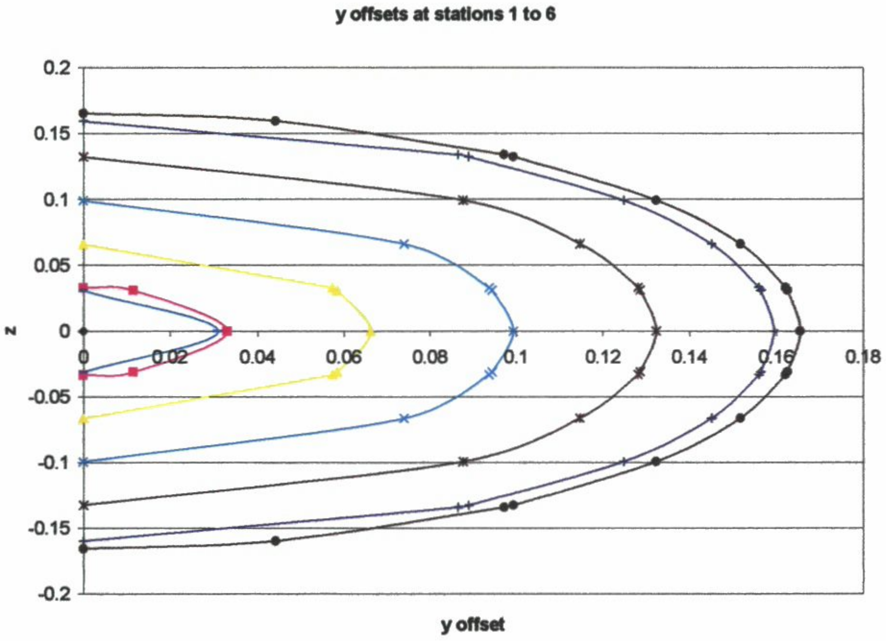


Figure 11.1 Ellipsoid bow section Y offsets

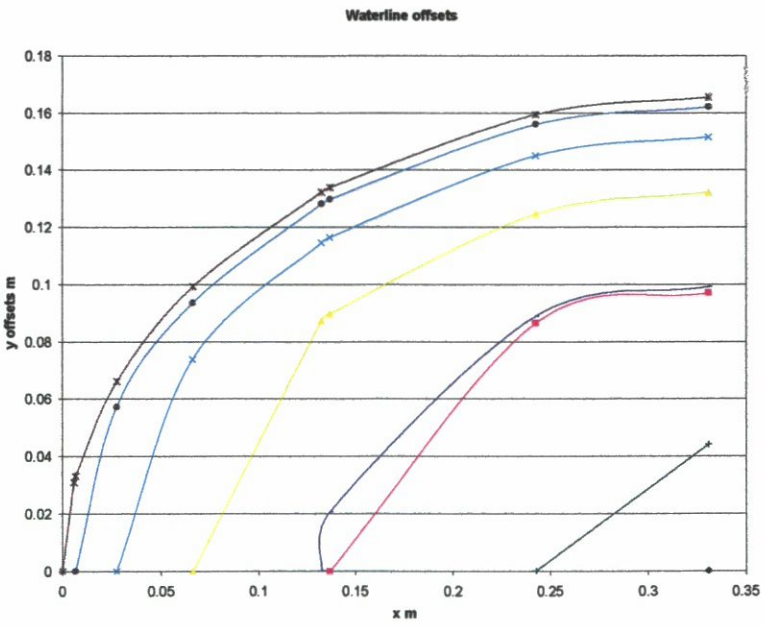


Figure 11.2 Ellipsoid bow section waterlines

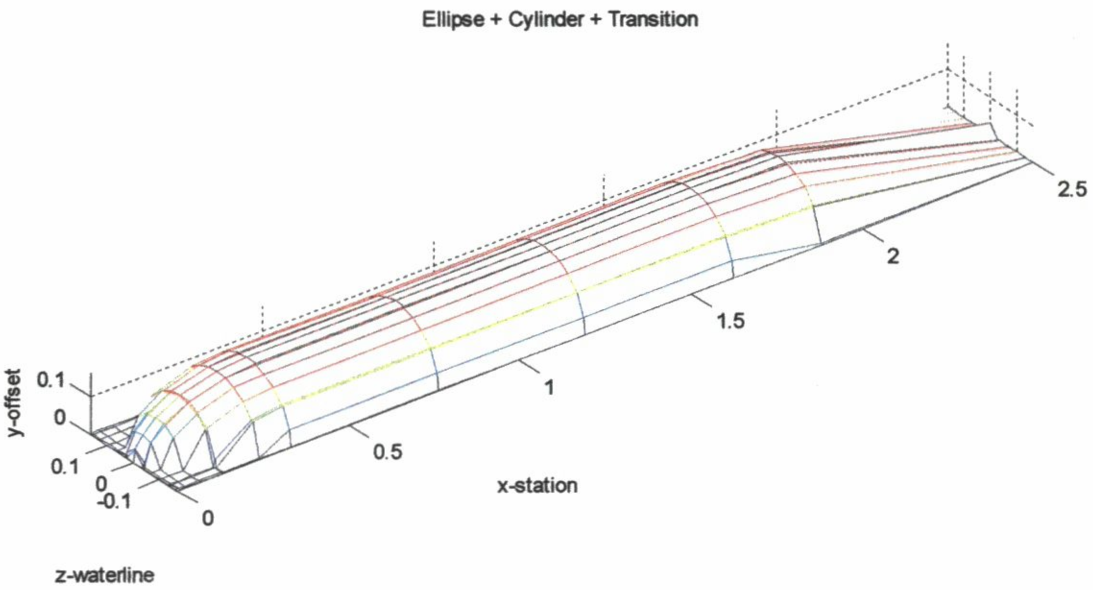


Figure 11.3 AUTOSUB space model

Chapter 12

BLOCKAGE CORRECTION

12.1 Matlab script for comparison of blockage correction methods

```
% ModelBlockageComparison

% Estimates the blockage effect of the AUTOSUB model by a number of means and
compares them.
% Data from the best method is saved in the form of an ascii text file.

%____________________________________________________________________________
%____________________________________________________________________________

% 1. Basic data.

% Acceleration due to gravity, m/s^2:

g=9.80665;

% Kinematic Viscosity (m^2/s) and density (kg/m^3) of fresh water at 15 deg C, :

nu=1.5e-6;
rho=1000;

% Length of model, m:

L=2.5;

% Cross sectional area of model, m^2:

Am=0.086;

% Wetted surface area of model, m^2:

S=2.278;

% Volume of model (excluding hydroplanes), m^3:

V=0.165;

% Depth of water in tank, m:

h=1.85;

% Width of tank, m:
```

```
w=3.7;

% Cross sectionsl Area of tank, m^2:

A=h*w;

% Blockage Ratio (ratio of model cross sectional area, Am, to that of the
% tank.

m=Am/A;

% Model net drag force as a function of aoa and speed u:

load 'interpolated_net_model_drag.txt'
Fni=interpolated_net_model_drag;

% Range of towing tank speeds, m/s:

ui=(0.01:0.01:4.5);

% Range of model angles of attack:

aoai=(0:0.1:10);

% Reynolds Number:

Rn=ui*L/nu;

% Froude Number:

Fr=ui./sqrt(g*L);

% Depth Froude Number:

Frh=ui/sqrt(g*h);

%_______________________________________________________________

%   2. Young & Squire's method.

% Difference between flow velocity experienced by the model in the tank, u,
% and the velocity it would experience in open water, uinf:

udifv=ui*V/A^(3/2);

figure
plot(ui,udifv)
xlabel('model speed - m/s')
ylabel('Speed correction - m/s')
```

```
Title('AUTOSUB model blockage velocity correction Young & Squire method')

% Plot as % speed correction.

upcv=(udifv./ui)*100;
figure
plot(ui,upcv)
xlabel('model speed - m/s')
ylabel('Speed correction - %')
Title('AUTOSUB model blockage velocity correction Young & Squire method')


%_____________________________________________________________________________

%  3. Schuster's method.

% Viscous resistance from ITTC'57 model

Cv=0.075./((log10(Rn)-2).^2);

figure
plot(Rn,Cv)
Title('AUTOSUB Model Viscous Resistance Coefficient (ITTC 57 model)')
xlabel('Rn')
ylabel('Cv')


Fv=0.5*rho*S*ui.^2.*Cv;
figure
plot(ui,Fv)
Title('AUTOSUB Model Viscous Resistance Force (ITTC 57 model)')
xlabel('Speed - m/s')
ylabel('Viscous drag - N')
figure
plot(ui.^2,Fv)
Title('AUTOSUB Model Viscous Resistance Force (ITTC 57 model)')
xlabel('Speed^2 - (m/s)^2')
ylabel('Viscous drag - N')

% Assume viscous resistance is similar for all angles of attack

for i=1:101;
  Fvi(:,i)=Fv';
  Frhi(:,i)=Frh';
  uii(:,i)=ui';
end

% Difference between flow velocity experienced by the model in the tank, ui,
% and the velocity it would experience in open water, uinf:
```

```
Fni(1,:)=[];
udifs=uii.*((m./(1-m-(Frhi).^2))+((1-(Fvi./Fni))*(2/3).*(Frhi).^10));
figure
plot(ui,udifs)
xlabel('model speed - m/s')
ylabel('Speed correction - m/s')
Title('AUTOSUB model blockage velocity correction Schuster method')

udifs1=udifs(:,1)';
upcs=(udifs1./ui)*100;
figure
plot(ui,upcs)
xlabel('model speed - m/s')
ylabel('Speed correction - %')
Title('AUTOSUB model blockage velocity correction Schuster method')

%_____________________________________________________________________

%   4. Tamura's method.

udift=ui.*(0.67*m*(L/w)^(3/4)./(1-Frh.^2));
figure
plot(ui,udift)
xlabel('model speed - m/s')
ylabel('Speed correction - m/s')
Title('AUTOSUB model blockage velocity correction Tamura method')
axis([0 4.1 0 0.4]);

% Plot as % speed correction.

upct=(udift./ui)*100;
figure
plot(ui,upct)
xlabel('model speed - m/s')
ylabel('Speed correction - %')
Title('AUTOSUB model blockage velocity correction Tamura method')
axis([0 4.1 0 20]);

%_____________________________________________________________________

%   5. Scott's method.

%   The velocity correction according to this method is calculated in the
%   folowing script.

scott_correction

%_____________________________________________________________________
% Compare methods
```

```
figure
plot(ui,udifv,'b')
xlabel('model speed - m/s')
ylabel('Speed correction - m/s')
title('AUTOSUB model blockage velocity correction -comparison of methods')
hold
plot(ui,udifs,'r')
plot(ui,udift,'g')
plot(ui3,udifsc,'k')
axis([1 4.1 0 1]);

Legend('Young & Squire','Schuster','Tamura','Scott',2)

% Presented as % increase n speed required to convert towing tank to
% equivalent open water speed:

figure
plot(ui,upcs)
xlabel('u m/s')
ylabel('speed correction %')
title('Scott Blockage Correction for AUTOSUB Model Experiments')
hold
plot(ui,upcs,'r')
plot(ui,upct,'g')
plot(ui3,upcsc,'k')
axis([1 4.1 0 20]);

Legend('Young & Squire','Schuster','Tamura','Scott',2)
```

12.2 Matlab script for determination of blockage effect using Scott's method

```
% scott_correction

% Calculates blockage correction according to Scott's method
%   for the AUTOSUB model across the range of speeds used in the twing tank
%   experiments.

% 1. Basic data.

% Acceleration due to gravity, m/s^2:

g=9.80665;

% Kinematic Viscosity (m^2/s) and density (kg/m^3) of fresh water at 15 deg C, :

nu=1.5e-6;
rho=1000;
```

```
% Length & Breadth of model, m:

L=2.5;
B=0.331;

% Cross sectional area of model, m^2:

Am=0.086;

% Wetted surface area of model, m^2:

S=2.278;

% Volume of model (excluding hydroplanes), m^3:

V=0.165;

% Depth of water in tank, m:

h=1.85;

% Width of tank, m:

w=3.7;

% Cross sectionsl Area of tank, m^2:

A=h*w;

% Blockage Ratio (ratio of model cross sectional area, Am, to that of the
% tank.

m=Am/A;

% Model net drag force as a function of aoa and speed u:

load 'interpolated_net_model_drag.txt'
Fni=interpolated_net_model_drag;

% Range of towing tank speeds, m/s:

ui=(0.01:0.01:4.5);

% Range of model angles of attack:

aoai=(0:0.1:10);

% Reynolds Number:

Rn=ui*L/nu;
```

```
% Froude Number:

Fr=ui/sqrt(g*L);

% Depth Froude Number:

Frh=ui/sqrt(g*h);

%________________________________________________________________________
_______

% Scott's formula is the sum of two terms f and w.
% f accounts for friction effects and is f(Rn).
% w accounts for wave effects and is a f(Fr).

% 5.1Plot Re and Fn for the band of speeds of interest:

figure
plot(ui,Rn)
xlabel('model speed - m/s')
ylabel('Rn')
Title('Reynolds Number for the range of model speeds of interest')

figure
plot(ui,Fr)
xlabel('model speed - m/s')
ylabel('Fr')
Title('Froude Number for the range of model speeds of interest')

%________________________________________________________________________

% 5.2 The friction term is a function of a factor k1 which in turn is a
% f(Rn). For the AUTOSUB model:
%     k1=k11=1.3 for 2 x 10^6 < Rn < 4.8 x 10^6
%     k1=k13 varies linearly from 1.3 @ Rn = 4.8 x 10^6
%                     to  0.85 @ Rn = 8.3 x 10^6

% Define speeds at which Re=2 x 10^6 and 4.8 x 10^6 and calculate udif
% (index for Rn = 4.8 x 10^6 is 288).

ui1=ui;
Rn1=Rn;
ui1(288:length(ui))=[];
Rn1(288:length(Rn))=[];
ui1(1:find(Rn==2e6))=[];
Rn1(1:find(Rn==2e6))=[];

% Calculate k1(Rn) for the interval 2 x 10^6 < Re < 4.8 x 10^6.
```

```
for i=1:length(Rn1)
k11(:,i)=1.3;
end

% Calculate k1(Rn) for the interval Re > 4.8x10^6

ui2=ui;
Rn2=Rn;
ui2(1:288)=[];
Rn2(1:288)=[];

k12g=[1.3 0.85];
Rn2g=[4.8e6 8.3e6];

[P,S]=polyfit(Rn2g,k12g,1);
k12=polyval(P,Rn2);

k13=[k11 k12];
Rn3=[Rn1 Rn2];
figure
plot(Rn3,k13)
xlabel('Rn')
ylabel('k1')
title('Scott Blockage Correction - Factor k1 for AUTOSUB Model Experiments')
axis([2e6 7e6 0.9 1.5])

% calculate friction term speed correction.

ui3=[ui1 ui2];

udiff=ui3.*(k13.*V.*(A.^(-3/2)));
figure
plot(ui3,udiff)
xlabel('u m/s')
ylabel('speed correction m/s')
title('Scott Blockage Correction - Friction Term for AUTOSUB Model Experiments')

%
```

```
% 5.3  The wave term is a function of a factor k2 which in turn is a
% f(Fr).

% First define Fr for the range of speeds of interest:

Fr1=Fr;
Fr1(1:find(Rn>2e6))=[];

% Calculate factor k2:
```

```
k2=2.4*((Fr1-0.22).^2);
figure
plot(Fr1,k2)
xlabel('Fr')
ylabel('k2')
title('Scott Blockage Correction - Factor k2 for AUTOSUB Model Experiments')

% Wave speed correction

udifw=ui3.*(B.*(L.^2).*k2.*(A^(-3/2)));
figure
plot(ui3,udifw)
xlabel('u m/s')
ylabel('speed correction m/s')
title('Scott Blockage Correction - Wave Term for AUTOSUB Model Experiments')

%________________________________________________________________

%  5.4 Total correction

udifsc=udiff+udifw;
figure
plot(ui3,udifsc)
xlabel('u m/s')
ylabel('speed correction m/s')
title('Scott Blockage Correction for AUTOSUB Model Experiments')

% Presented as % increase n speed required to convert towing tank to
% equivalent open water speed:

upcsc=(udifsc./ui3)*100;
figure
plot(ui3,upcsc)
xlabel('u m/s')
ylabel('speed correction %')
axis([1 4.1 0 15])
title('Scott Blockage Correction for AUTOSUB Model Experiments')
```

12.3 Matlab script for determination of blockage effect using Tamara's method modified for high Froude numbers

```
% ModelBlockage

% Estimates the blockage correction required for AUTOSUB model test data
%  The estimate is based on Tamara's formula for the speed range up to
%  Fr=0.7 (corresponding to u=3.4m/s).
%  This data is then extrapolated to give corrections out to 4.1 m/s.
%  Data is saved in the form of an ascii text file
%  'blockage_correction_2.txt'. Col 1 = measured speed, Col 2 = % speed
%  correction.

%_______________________________________________________________________
%_______________________________________________________________________

clear
close all

%_______________________________________________________________________

% 1. Basic data.

% Acceleration due to gravity, m/s^2:

g=9.80665;

% Kinematic Viscosity (m^2/s) and density (kg/m^3) of fresh water at 15 deg C, :

nu=1.5e-6;
rho=1000;

% Length of model, m:

L=2.5;

% Cross sectional area of model, m^2:

Am=0.086;

% Wetted surface area of model, m^2:

S=2.278;

% Volume of model (excluding hydroplanes), m^3:

V=0.165;

% Depth of water in tank, m:

h=1.85;

% Width of tank, m:

w=3.7;

% Cross sectionsl Area of tank, m^2:

A=h*w;
```

```
% Blockage Ratio (ratio of model cross sectional area, Am, to that of the
% tank.

m=Am/A;

% Model net drag force as a function of aoa and speed u:

load 'interpolated_net_model_drag.txt'
Fni=interpolated_net_model_drag;

% Range of towing tank speeds, m/s:

ui=(0.01:0.01:3.4);

% Range of model angles of attack:

aoai=(0:0.1:10);

% Reynolds Number:

Rn=ui*L/nu;

% Froude Number:

Fr=ui./sqrt(g*L);

% Depth Froude Number:

Frh=ui/sqrt(g*h);

%____________________________________________________________

%   4. Tamura's method.

udift=ui.*(0.67*m*(L/w)^(3/4)./(1-Frh.^2));
figure
plot(ui,udift)
xlabel('model speed - m/s')
ylabel('Speed correction - m/s')
Title('AUTOSUB model blockage velocity correction Tamura method')
axis([0 4.1 0 0.4]);

% Plot as % speed correction.
upct=(udift./ui)*100;
figure
plot(ui,upct)
xlabel('model speed - m/s')
ylabel('Speed correction - %')
Title('AUTOSUB model blockage velocity correction')
axis([0 4.1 0 5]);
```

```
% Fit polynomial to data, check for accuracy and extrapolate.

[P,S]=polyfit(ui,upct,2);
u=[0:0.01:4.1];
upcte=polyval(P,u);

hold
plot(u,upcte,'r')

[P,S]=polyfit(ui,upct,3);
u=[0:0.01:4.1];
upcte=polyval(P,u);

plot(u,upcte,'g')

legend('fit', 'poly1', 'poly2')

% Export blockage correction as an ascii text file

save 'blockage_correction.txt' upcte -ascii
[P,S]=polyfit(ui,upct,2);

%   Repeat for aoa=0 analysis where speeds up to 4.57 m/s need to be
%   corrected.

u2=[0:0.01:4.57];
upcte2=polyval(P,u2);

figure
plot(ui,upct)
xlabel('model speed - m/s')
ylabel('Speed correction - %')
Title('AUTOSUB model blockage velocity correction')
hold
plot(u2,upcte2,'r')

[P2,S2]=polyfit(ui,upct,3);
u=[0:0.01:4.58];
upcte=polyval(P,u2);

plot(u2,upcte2,'g')

legend('fit', 'poly1', 'poly2')
axis([0 5 0 5]);

% Export blockage correction as an ascii text file
output=[u2' upcte2'];
save 'blockage_correction_2.txt' output -ascii
```

Chapter 13

BARE HULL DRAG

13.1 Matlab script for determining Model drag

```
% BareHullDrag

%  a) Produces a 3d model of bare hull drag of model by producing a 3d spline
%  interpolation between model measured data points from exps 0304 & 0307 data,
%  subtracting the tear drag of the poles and the wave-making induced drag,
%  and correcting the effective speed to allow for blockage effects.
%       b) Plots the drag of the AUTOSUB model as a function of speed and angle of
%       attack.
%  c) Exports the corrected model bare hull data as ascii text files.

%______________________________________________________________________
%______________________________________________________________________

clear
close all

%______________________________________________________________________
% 1. Establish interpolated matrix and plot net drag (model + net pole drag
%   + blockage + wave effects corrected for acceleration and instrument drift).

% Angle-of-attack - deg:

aoa=[0 2 5 7 10];

% Speed look up table: Col1 = dial setting, col 2 = corresponding mean
% speed in m/s.
% Dial setting of speeds for pole drag data:

dial=[0 240 660 780 915];

% Corresponding mean speed:

u=look_up_speed(dial);

% Net drag force Fn in N, depth hole 5.
%       each column  corresponding to a value of aoa
%       each row to nominal u

Ft=[ 0   0   0   0   0
   25.9 26.4 27.4 30.7 32.9
   168.2 174.4 185.4 200.9 218.5
   240.3 248.4 262.8 281.2 304.6
   329.0 341.0 357.0 383.8 413.2];

% Produce coarse mesh plot to indicate shape of data.

[xci,yci]=meshgrid(0:0.2:10, 0:0.1:4.5);
zci=interp2(aoa,u,Ft,xci,yci,'cubic');
figure
mesh(xci,yci,zci);
xlabel('aoa deg')
ylabel('u m/s')
zlabel('Ft N')
title('Total Model Drag - Interpolated - hydroplanes 0 deg')
axis([0 10 0 4.5 0 450])
colorbar

% Produce detailed interpolation between data points and save as tab delimited ascii
% file

[aoai,ui]=meshgrid(0:0.1:10, 0:0.01:4.1);
Fti=interp2(aoa,u,Ft,aoai,ui,'cubic');
save 'interpolated_total_model_drag.txt' Fti -ascii -tabs

figure
surf(aoai,ui,Fti);
xlabel('aoa deg')
ylabel('u m/s')
zlabel('Fti N')
title('Total Model - Interpolated - hydroplanes 0 deg')
axis([0,10,0,4.3,0,450])
shading interp;
colorbar

figure
v=(50:50:450);
contour(aoai,ui,Fti,v,'k');
axis([1 4.2 0 10])
[c,h]=contour(aoai,ui,Fti,v,'k');
clabel(c,h)
xlabel('aoa deg')
ylabel('u m/s')
title('Total Model - Interpolated - hydroplanes 0 deg')

%_____________________________________________________________________

% 2. Import interpolated pole drag in 3 dimensions, x=aoa, y=u, z=Fd.

load interpolated_pole_drag.txt;
Fpi=interpolated_pole_drag;
```

```
% Shorten to cover only speeds up to 4.1 m/s

Fpi=Fpi(1:411,:);

% Plot net pole drag (including its wave drag).

figure
surf(aoai,ui,Fpi);
xlabel('aoa deg')
ylabel('u m/s')
zlabel('Fd N')
title('Pole Drag - Interpolated - hydroplanes 0 deg')
axis([0,10,0,4.3,0,450])
shading interp;
colorbar;

% Plot both pole and net drag on same axes.

figure
surf(aoai,ui,Fti);
xlabel('aoa deg')
ylabel('u m/s')
zlabel('Fn N')
title('Net and Pole Drag - Interpolated - hydroplanes 0 deg')

shading interp;
colorbar
axis([0,10,0,4.3,0,450])
hold
surf(aoai,ui,Fpi);
shading interp;

%________________________________________________________________

%  3. Establish net model drag by subtracting interpolated pole drag from
%  interpolated total drag and plot.

Fni=Fti-Fpi;
save 'interpolated_net_model_drag.txt' Fni -ascii -tabs

figure
surf(aoai,ui,Fni);
xlabel('aoa deg')
ylabel('u m/s')
zlabel('Fn N')
title('Model Drag (pole corrected)- hydroplanes 0 deg')
axis([0,10,0,4.3,0,200])
shading interp;
colorbar;
```

```
%______________________________________________________________________

%  3. Establish model drag by subtracting induced model generated wave drag
%  from net drag and plot.

% Import data for wave drag of AUTOSUB model at depth 0.88m
% as a function of speed deduced from thin ship theory
% in file 'wave resist.xls' and stored in text file 'thin_ship_wave_drag.txt'.
% First column is speed in m/s. Col 2 is Wave drag in N.

load 'thin_ship_wave_drag.txt';
uw=thin_ship_wave_drag(:,1);
Fw=thin_ship_wave_drag(:,2);

% Produce interpolated data;

uwi=0:0.01:5;
Fwi=interp1(uw,Fw,uwi,'cubic');
save 'interpolated_model_wave_drag.txt' Fwi -ascii -tabs

% crop uwi and Fwi tp 4.1 m/s and transpose

uwi(:,(find(uwi>4.1)))=[];
Fwi(:,((length(uwi)+1):length(Fwi)))=[];
uwi=uwi';
Fwi=Fwi';

figure
plot (uwi,Fwi)
hold
plot(uw,Fw,'x')
Title('Wave drag interpolation check')
legend('interpolated data','imported data points',2)
xlabel('Speed m/s')
ylabel('Wave drag N')

% Assume induced wave drag is independent of aoa.

for i=1:101
   uwi1(:,i)=uwi;
   Fwi1(:,i)=Fwi;
end

figure
surf(aoai,uwi1,Fwi1);
xlabel('aoa deg')
ylabel('u m/s')
zlabel('Fw N')
title('Model Wave Drag - hydroplanes 0 deg')
shading interp;
```

```
colorbar;

Fmi=Fni-Fwi1;

figure
surf(aoai,ui,Fmi);
xlabel('aoa deg')
ylabel('u m/s')
zlabel('Fm N')
title('Model Drag (pole & wave corrected)- hydroplanes 0 deg')
axis([0,10,0,4.3,0,200])
shading interp;
colorbar;

%________________________________________________________________

%  4. Correct for blockage using 3 order polynomial extrapolation.


%  Load blockage % speed correction data derived in script 'ModelBlockage'
%  and stored in file 'blockage_correction.txt'.

block_cor=load('blockage_correction.txt');
% Set correction for all aoa.

block_cor=block_cor';

for i=1:101
   block_cor_a(:,i)=block_cor;
end

figure
plot(ui, block_cor)
title('AUTOSUB Model Blockage Correction')
xlabel('Speed m/s')
ylabel('Speed correction %')

uicor=ui.*(1+(block_cor_a/100));
figure
plot(ui, uicor)
title('AUTOSUB Model Blockage Correction')
xlabel('Measured speed m/s')
ylabel('Corrected speed m/s')

% Plot corrected drag force as function of u and aoa.

figure
surf(aoai,uicor,Fmi);
xlabel('aoa deg')
ylabel('u m/s')
```

```
zlabel('Fm N')
title('Model Drag (pole,wave and bockage corrected)- hydroplanes 0 deg - speed
corrected')
axis([0,10,0,4.3,0,200])
shading interp;
colorbar;

figure
v=(10:10:150);
contour(aoai,uicor,Fmi,v,'k');
axis([1 4.2 0 10])
[c,h]=contour(aoai,uicor,Fmi,v,'k');
clabel(c,h)
xlabel('aoa deg')
ylabel('u m/s')
title('Model Drag (pole,wave and bockage corrected)- hydroplanes 0 deg - speed
corrected')

figure
plot(uicor,Fmi(:,1),'b')
hold
load 'model_drag_aoa0.txt';
plot(model_drag_aoa0(:,1),model_drag_aoa0(:,2),'r')
title('Model drag force, aoa = 0')
ylabel('Model net drag - N')
xlabel('Speed corrected for blockage - m/s')
legend('2d interpolation','3d interpolation',2)

figure
plot(uicor.^2,Fmi(:,1),'b')
hold
load 'model_drag_aoa0.txt';
plot(model_drag_aoa0(:,1).^2,model_drag_aoa0(:,2),'r')
title('Model drag force vs u^2, aoa = 0')
ylabel('Model net drag - N')
xlabel('Speed^2- (m/s)^2')
legend('2d interpolation','3d interpolation',2)

%____________________________________________________________

% 5. Save key corrected data as and save as tab delimited ascii 'txt'
% files.

save 'bare_model_drag.txt' Fmi -ascii
save 'bare_model_aoa.txt' aoai -ascii
save 'bare_model_speed.txt' uicor –ascii
```

13.2 Matlab script for determining Full-scale drag

```
% AUTOSUB_bare_hull

% Converts model data to full scale and presents graphically.

%_______________________________________________________________
%_______________________________________________________________
close all
clear

% 1. Define basic data

% Model Surface area (based on v^2/3 m^2).
sm=0.3008;

% Model wetted surface area.

smwet=2.278;

% Model length

Lm=2.5;

% Full scale AUTOSUB surface area (based on V^2/3 m^2) and length

sA=2.278;
LA=6.794;

% Density & Kinematic viscosity of fresh water kg/m^3 at 15 deg C.
rho=1000;
nu=1.14e-6;

% Density & Kinematic viscosity of salt water kg/m^3 at 10 deg C.
rho1=1027;
nu1=1.46e-6;

% 1. Load and plot processed model data produced from script BareHullDrag

load 'bare_model_drag.txt';
Fm=bare_model_drag;
load 'bare_model_speed.txt';
u=bare_model_speed;
load 'bare_model_aoa.txt';
aoa=bare_model_aoa;

figure
surf(aoa,u,Fm);
xlabel('aoa deg')
ylabel('u m/s')
zlabel('Fm N')
title('Model Drag - hydroplanes 0 deg')
```

```
axis([0,10,0,4.3,0,200])
shading interp;
colorbar;

%_______________________________________________________________________

% 2. Plot families of curves of Model drag force, Fm, vs u for a range of aoa
%   and Fm vs aoa for a range of u.

% Reduce u and aoa to col vectors.

u1=u(:,1);
aoa1=aoa(1,:);
figure
i=0;
plot(u1,Fm(:,find(aoa1==i)))
hold
for i=1:1:10;
plot(u1,Fm(:,find(aoa1==i)),'r')
end
title('Model bare hull drag for aoa = 0 to 10 deg')
xlabel('Model speed m/s')
ylabel('Drag force N')
axis([0 4.1 0 180])

figure
i=1;
plot(aoa1,Fm(find(u1>(i-0.01) & u1<i),:))

hold
for i=1.5:0.5:4.0;
plot(aoa1,Fm(find(u1>(i-0.01) & u1<i),:),'r')
end
title('Model bare hull drag for u=1 to 4 m/s')
xlabel('Angle-of-attack degrees')
ylabel('Drag force N')

%_______________________________________________________________________

%  Calculate and plot Drag Coefficient Cd (based on V^2/3) vs Reynolds number

Cd=2.*Fm./(sm*rho*(u.^2));
Cd2=Cd;
aoa2=aoa;
u2=u;
%  Convert speed to Reynolds number.

Rn2=u2.*(Lm/nu);

Cd2(1:100,:)=[];
```

```
aoa2(1:100,:)=[];
Rn2(1:100,:)=[];
figure
surf(aoa2,Rn2,Cd2);
xlabel('aoa deg')
ylabel('Rn')
zlabel('Cd')
title('Bare Hull Drag Coefficient - hydroplanes 0 deg - speed corrected')
%axis([0,10,0,4.1,0,200])
shading interp;
colorbar;

%________________________________________________________________________

% 3. Plot families of curves of Cd vs Rn for a range of aoa
%    and Fm vs aoa for a range of Rn.

% Reduce u and aoa to col vectors.

Rn3=Rn2(:,1);
aoa3=aoa2(1,:);

figure
i=0;
plot(Rn3,Cd2(:,find(aoa1==i)))
hold
for i=1:1:10;
plot(Rn3,Cd2(:,find(aoa1==i)),'r')
end
title('Bare hull drag coefficient for aoa = 0 to 10 deg')
xlabel('Rn')
ylabel('Cd')
axis([4e6 9e6 0 0.07])

figure
i=4e6;
plot(aoa3,Cd2(find(Rn3>(i-0.03e6) & Rn3<i),:))

hold
for i=5e6:1e6:9e6;
plot(aoa3,Cd2(find(Rn3>(i-0.03e6) & Rn3<i),:),'r')
end
title('Bare hull drag coefficient for Re=4e6 to 9e6')
xlabel('Angle-of-attack degrees')
ylabel('Cd')

%________________________________________________________________________

%  Plot Cdwet (based on wetted surface area) vs Rn for aoa=o
% and compare with Hoerner data for 1:7 aspect ratio
% body of revolution (fig22 p 6-16).

Cdwet=2.*Fm./(smwet*rho*(u.^2));
Cd2wet=Cdwet;
Cd2wet(1:100,:)=[];

figure
plot(Rn3,Cd2wet(:,1))
title('Bare hull drag coefficient for aoa = 0 cf Hoerner data')
xlabel('Rn')
ylabel('Cd(wet)')
hold

% Hoerner data

% R101 Airship l/d 5.5
Reh=[1.1e6 1.4e6 1.7e6 2e6 2.5e6 3.1e6 6.2e6 6.8e6];
Cdwetr=[.0021 .002 .0019 .0018 .0018 .002 .0025 .0027];
Rehi=(1.1e6:0.01e6:6.8e6);
Cdwetri=spline(Reh,Cdwetr,Rehi);
plot(Rehi,Cdwetri,'r')
plot(Reh,Cdwetr,'r+')
% Short Airship and towing tank
Rehs=[2e6 2.2e6 3e6 4e6 5e6 6e6 7e6];

Cdwets=[.0027 .0027 .0029 .0031 .003 .003 .003];
Rehsi=(2e6:0.01e6:7e6);
Cdwetsi=spline(Rehs,Cdwets,Rehsi);
plot(Rehsi,Cdwetsi,'r')
plot(Rehs,Cdwets,'rx')
axis([1e6 10e6 1e-3 5.2e-3])
legend('AUTOSUB - l/d 6.7', 'R101 Airship - l/d 5','','Short Airship - l/d 6.7','',2)

% Export the data for Cd vs Re for AUTOSUB, R101 & Short Airship as Ascii
% tab delimited file.

output1=[Rn3,Cd2wet(:,1)];
output2=[Rehi',Cdwetri'];
output3=[Rehsi',Cdwetsi'];
save 'comparison_data_AUTOSUB.txt' output1 -ascii -tabs;
save 'comparison_data_R101.txt' output2 -ascii -tabs;
save 'comparison_data_Short.txt' output3 -ascii -tabs;

%______________________________________________________________________

% 3. Plot families of curves of Cd and drag force, FA, vs speed,ua, for a range of aoa
%   and vs aoa for a range of speeds for the full scale AUTOSUB hull.

uA=Rn3*nu/LA;
```

```
figure
i=0;
plot(uA,Cd2(:,find(aoa1==i)))
hold
for i=1:1:10;
plot(uA,Cd2(:,find(aoa1==i)),'r')
end
title('Bare hull drag coefficient for aoa = 0 to 10 deg')
xlabel('Full scale speed m/s')
ylabel('Cd')
%axis([1 1.6 .01 .09])

figure
i=0.5;
plot(aoa3,Cd2(find(uA>(i-0.004) & uA<i),:))
hold
for i=0.6:0.1:1.5;
plot(aoa3,Cd2(find(uA>(i-0.004) & uA<i),:),'r')
end
title('Bare hull drag coefficient for full scale speeds of 0.5 to 1.5 m/s')
xlabel('Angle-of-attack degrees')
ylabel('Cd')

% Plot Cdwet vs full-scale speed for aoa=0.

figure
plot(uA,Cd2wet(:,1))
title('Bare hull drag coefficient for aoa = 0')
xlabel('Full-scale speed (m/s)')
ylabel('Cd(wet)')
hold

%____________________________________________________________

% Calculate drag force on full scale AUTOSUB.

for i=1:101
   uA1(:,i)=uA;
end
FA=0.5.*Cd2.*rho1.*sA.*uA1.^2;

figure
i=0;
plot(uA,FA(:,find(aoa1==i)))
hold
for i=1:1:10;
plot(uA,FA(:,find(aoa1==i)),'r')
end
title('AUTOSUB bare hull drag force for aoa = 0 to 10 deg')
xlabel('Full scale speed m/s')
```

```
ylabel('Drag Force N')
%axis([1 1.6 .01 .09])

% Plot for aoa = 0 only and compare with decelleration trials results

% Import results

load 'trials_results.txt'
utrial=trials_results(:,1);
Ftrial=trials_results(:,2);
load 'trials_results2.txt'
utrial2=trials_results2(:,1);
Ftrial2=trials_results2(:,2);
load 'trials_results1A.txt'
utrial1A=trials_results1A(:,1);
Ftrial1A=trials_results1A(:,2);
load 'trials_results2A.txt'
utrial2A=trials_results2A(:,1);
Ftrial2A=trials_results2A(:,2);
load 'trials_results1B.txt'
utrial1B=trials_results1B(:,1);
Ftrial1B=trials_results1B(:,2);
load 'trials_results2B.txt'
utrial2B=trials_results2B(:,1);
Ftrial2B=trials_results2B(:,2);


figure
i=0;
plot(uA,FA(:,find(aoa1==i)))
title('AUTOSUB bare hull drag force for aoa = 0')
xlabel('Full scale speed m/s')
ylabel('Drag Force N')
hold
plot(utrial,Ftrial,'x')
plot(utrial2,Ftrial2,'o')
legend('towing tank','deceleration trial1','deceleration trial2',2)

figure
i=0;
plot(uA,FA(:,find(aoa1==i)))
title('AUTOSUB bare hull drag force for aoa = 0')
xlabel('Full scale speed m/s')
ylabel('Drag Force N')
hold
plot(utrial1A,Ftrial1A,'x')
plot(utrial2A,Ftrial2A,'o')
legend('towing tank','deceleration trial1','deceleration trial2',2)

figure
```

```
i=0;
plot(uA,FA(:,find(aoa1==i)))
title('AUTOSUB bare hull drag force for aoa = 0')
xlabel('Full scale speed m/s')
ylabel('Drag Force N')
hold
plot(utrial1B,Ftrial1B,'x')
plot(utrial2B,Ftrial2B,'o')
legend('towing tank','deceleration trial1','deceleration trial2',2)


%

figure
i=0.5;
plot(aoa3,FA(find(uA>(i-0.004) & uA<i),:))
hold
for i=0.6:0.1:1.5;
plot(aoa3,FA(find(uA>(i-0.004) & uA<i),:),'r')
end
title('AUTOSUB bare hull drag force for full scale speeds of 0.5 to 1.5 m/s')
xlabel('Angle-of-attack degrees')
ylabel('Drag Force N')

figure
v=(10:10:160);
contour(aoa3,uA,FA,v,'k');
axis([1 4.2 0 10])
[c,h]=contour(aoa3,uA,FA,v,'k');
clabel(c,h)
xlabel('aoa deg')
ylabel('u m/s')
title('AUTOSUB Drag- hydroplanes 0 deg')
```

Chapter 14

MATLAB SCRIPT FOR DETERMINING ADDED MASS

```
% AnalyseAccellerationA is a script to explore the relationship between
%       drag force and acceleration

close all
clear

% Read in data
[nsamples,runtime,date,time,s,sf]=readwad('040408A5.WAD');

%       Establish a vector to define the time at each acceleration data point.
x=(1:length(Accel));
t=x/sf;

% Convert acceleration data from g to m/s^2

Accel=Accel*9.8;

plot(t,Accel)
title('Raw Acceleration Data - Exp 040408')
xlabel('Time - sec')
ylabel('Acceleration - m/s^2')

%       Correct acceleration data. ( At t=5.57 measured accel = 5.57, at t= 13.5
ma=0.2637,
%               at t=15 ma=-0.72, at t=15.8 ma=-0.72, whereas true accel=0 for all).

%       Determine Force as f(t) from data in force blocks 1 and 3

Force=DragAft+DragFrd;

figure(2)
plot(t,Force)
title('Raw Force Data')
xlabel('Time - sec')
ylabel('Force - N')

figure(3)
plot(Accel,Force,'+')
title('Force as a function of Acceleration - Exp 040408')
xlabel('Accel - m/s^2')
ylabel('Force - N')
```

```
% from figure 1 acceleration is constant from k(1) to k(2) sec

k=[3.5 5.9];
l=round(k*sf);

figure(4)
plot(Accel(l(1):l(2)),Force(l(1):l(2)),'+')
title('Force during period of constant Accel - Exp 040408')
xlabel('Accelleration - m/s^2')
ylabel('Force - N')
axis([-2.5 2 0 350])

% plot Force for period of constant accelleration
%       together with best fit second order polynomial.

figure(5)
plot(t(l(1):l(2)),Force(l(1):l(2)))
title('Force during period of constant Accelleration - Exp 040408')
xlabel('Time from start - sec')
ylabel('Force - m/s^2')

[p,s]=polyfit(t(l(1):l(2))',Force(l(1):l(2)),2)
hold
plot(t(l(1):l(2)),polyval(p,t(l(1):l(2))),'r-')

[p,s]=polyfit(t(l(1):l(2))',Force(l(1):l(2)),1)
plot(t(l(1):l(2)),polyval(p,t(l(1):l(2))),'g-')
legend('Data','Quadratic','Linear')
axis([2.5 6 0 350])
% mean acceleration over period of constant acceleration

mean_accel=mean(Accel(l(1):l(2)))

% plot velocity as the integral of acceleration using Simpson's Rule
%       from time start to K(2).

v(1)=0;
a=Accel;

for n=2:length(t)
v(n)=v(n-1)+(((a(n)+a(n-1))/2).*(t(n)-t(n-1)));
end;

figure(6)

plot(t,v)
title('Speed')
xlabel('Time - sec')
ylabel('Speed - m/s')
```

```
figure(7)

plot(t(l(1):l(2)),v(l(1):l(2)))
title('Speed during period of consrtant acceleration - Exp 040408')
xlabel('Time - sec')
ylabel('Speed - m/s')
axis([2.5 6 0 4])

% Drag force as f(v) at depth 6 is given by

figure(8)

plot(t(l(1):l(2)),Force(l(1):l(2)))
[p,s]=polyfit(t(l(1):l(2))',Force(l(1):l(2)),2)
plot(t(l(1):l(2)),polyval(p,t(l(1):l(2))))

Fd=8.4651*(v.*v)+6.6562*v;

hold
plot(t(l(1):l(2)),Fd(l(1):l(2)),'r-')
title('Force Components - Exp 040408')

% Total inertial force is total less drag force.

Fi=Force-Fd';
%plot(t(l(1):l(2)),Fi(l(1):l(2)),'g.')

[p,s]=polyfit(t(l(1):l(2))',Fi(l(1):l(2)),2)
plot(t(l(1):l(2)),polyval(p,t(l(1):l(2))),'g')

legend('Total','Drag','Inertial')
axis([2.5 6 0 350])

% Net apparant mass statistics

mn=Fi./a;
mean_mn=mean(mn(l(1):l(2)))
std_mn=std(mn(l(1):l(2)))
max_mn=max(mn(l(1):l(2)))
min_mn=min(mn(l(1):l(2)))

figure(9)

plot(t(l(1):l(2)),mn(l(1):l(2)))
hold
[p,s]=polyfit(t(l(1):(l(2)-10))',mn(l(1):(l(2)-10)),2)
plot(t(l(1):l(2)),polyval(p,t(l(1):l(2))))
title('Net apparent mass')
xlabel('time - sec')
```

```
ylabel('mass - kg')

figure(10)
plot(t(l(1):l(2)),mn(l(1):l(2)))
title('Apparant Mass Components - Exp 040408')
ylabel('Mass - kg')
xlabel('Time')
%axis([3 4 0 45])

% Mass of model plus support poles plus dynamometer below drag blocks (kg):

mi=116;

% Added mass

ma=mn-mi;
added_mass=mean(ma(l(1):l(2)))
hold
plot(t(l(1):l(2)),mi,'g')

% Added mass = apparent mass less weighed mass (19.9kg)

plot(t(l(1):l(2)),ma(l(1):l(2)),'r--')

legend('net mass','inertial mass','added mass')
```

Chapter 15

MULTIVARIATE LINEAR ANALYSIS

15.1 Results of multivariate linear analysis

A multivariate linear analysis was performed on the ADCP drag data, using the Matlab multivariate linear regression application, as a check on the effectiveness of the ANOVA based analysis reported in Chapter 2.10. The Matlab script is given at 2.10.1.2 below.

First a normal probability plot for the overall results was made as shown in Figure 2.10.1.1.

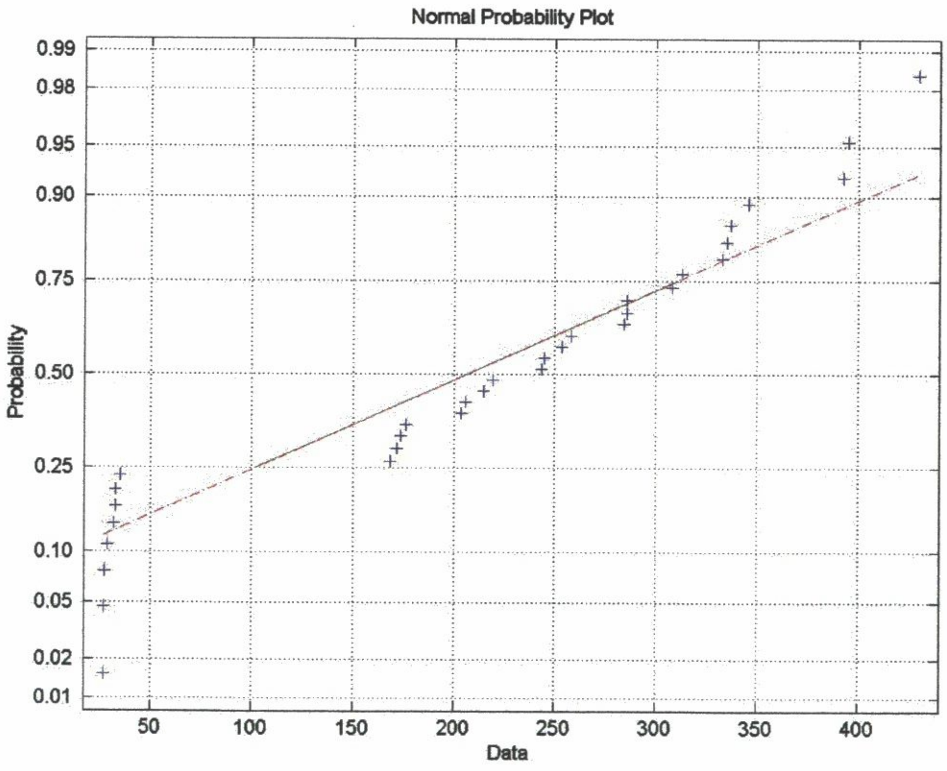


Figure 15.1.1 Normal probability plot of data

The x-axis represents the value of each data point in N. Each data point is ranked and plotted against the y axis, which gives the cumulative probability of a measured value being at least that large. For example, since there are 31 points in the data set considered here, there is a 24% probability that any value will be less than or equal to the value of the 8th largest data point, which, in this data set happens to have a value of 30N. The y-axis is drawn to a scale representative of the quantiles of a normal distribution. The quantiles are close together near the median and further apart at the extremes of the distribution, as illustrated in Figure 2.10.1.2. The solid line represents a linear fit to the data between the 1st and 3rd quartiles. The broken line extends the linear fit to cover the whole data set. If the variability in the data is due to purely random events, the data points would be expected to lie on a straight line. The fact that there are significant deviations from the straight line is a clear indication that the underlying data is not from a normal distribution and that specific effects are influencing the data.

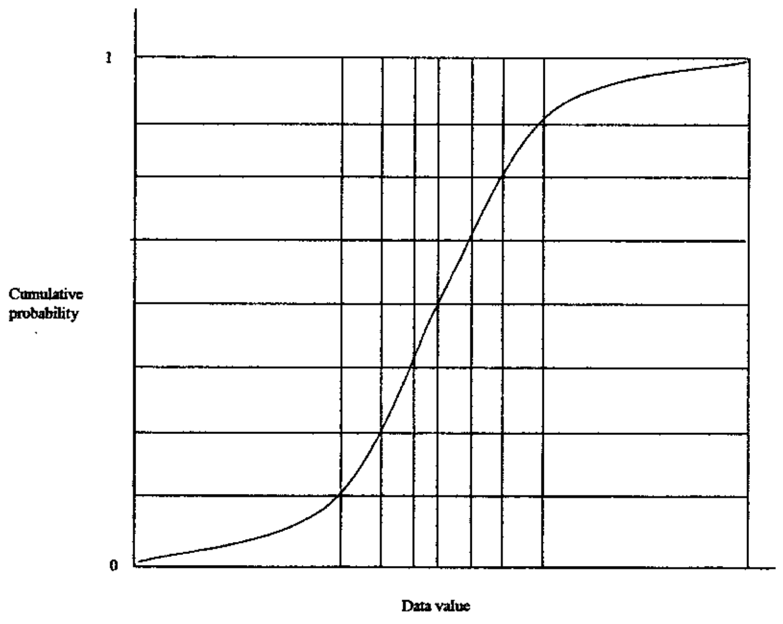


Figure 15.1.2 Cumulative probability, normal distribution

The Matlab function 'regress' performs a multiple linear regression using least squares and returns the vector of regression coefficients, b, in the linear model

$$y = bX,$$

where X is an *nxp* matrix, and y is the *nx1* vector of observations. Confidence intervals for $100(1 - \alpha)$ for any α may be returned, together with a vector of residuals

$(r = y - bX)$.

The coefficients derived from the regression applied to the ADCP data are given in Table2.10.1.1.

Factor number	Factor	Symbol	Value
1	Fwd ADCP	f	2.88
2	Fwd/aft interaction	fa	-1.60
3	Fwd ADCP/speed interaction	fu	-2.96
4	Aft ADCP	a	0.46
5	Speed	u	108.15
6	Fwd ADCP/angle of attack interaction	falpha	-2.22
7	Fwd ADCP/hydroplane angle interaction	fdelta	-4.08
8	Angle of attack	alpha	4.18
9	Hydroplane angle	delta	-0.11
10	Constant	k	-84.8

Table 15.1.1 Multiple linear regression coefficients

Thus, the regression equation is:

$$F_d = 2.9f - 1.6f_a - 3f_u + 0.5\alpha + 108u - 2.2f_\alpha - 4.2\alpha - 0.1\delta - 85$$

Where, as before *f, f_a, f_u, and f_α* may only take integer values in the range 1 to 4.

The 10% confidence limits about these coefficients are shown in Figure 2.10.1.3. The confidence limits are based on the t-statistic. The confidence bands are, therefore, accurate for large samples, but less so for small samples. In this case the total number of samples is 32. The assumption is that the influence of the factors is orthogonal. There is Thus, the equivalent of 32 samples per factor. It is, therefore, considered that there is a sufficiently large sample for the t-statistic to be appropriate and that this standard Matlab function will, therefore, provide a reasonable check on the anova based results.

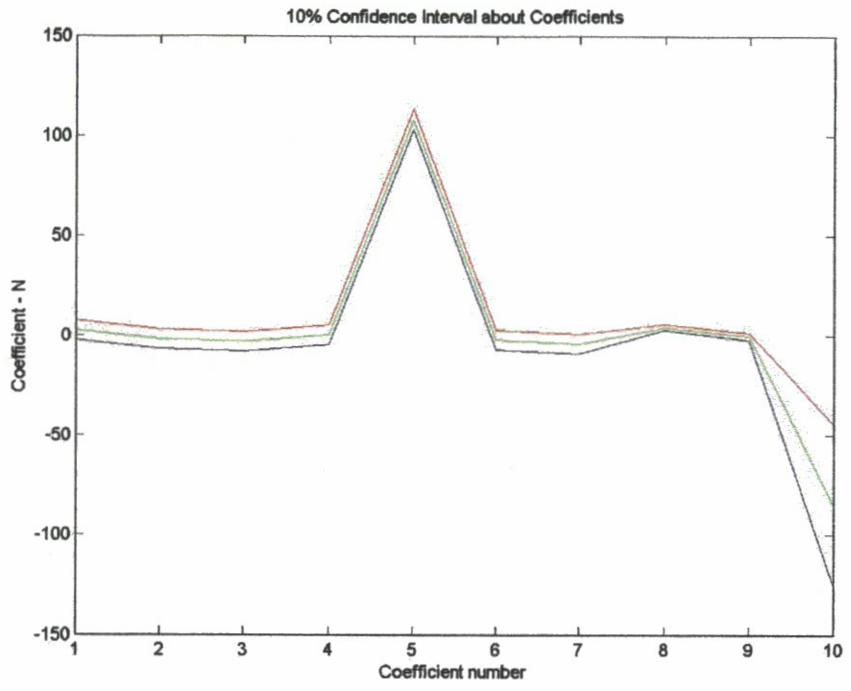


Figure 15.1.3 Coefficients

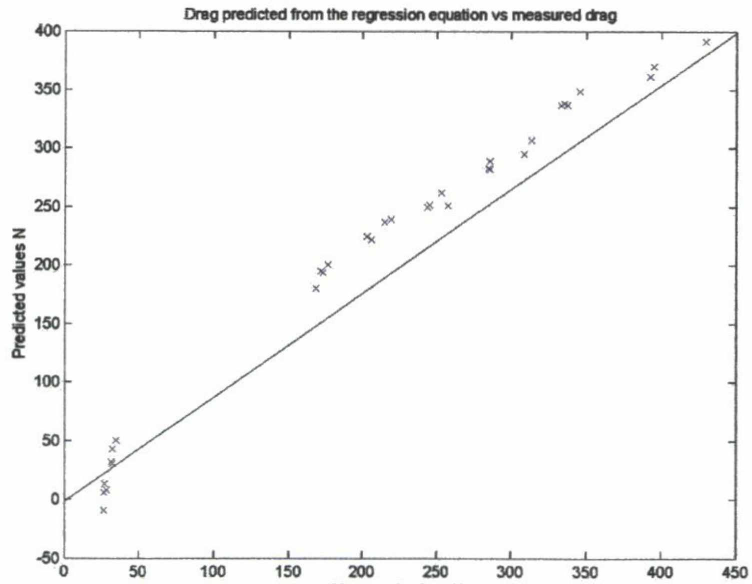


Figure 15.1.4 Regression equation prediction accuracy

The accuracy of the predictions obtained by applying the regression equation to the values used for each of the experiments is illustrated in Figure 15.1.4.

The equation consistently over-predicts due the assumed linear effect of all the factors. The residuals together with their 10% confidence limits are shown in Figure 2.10.1.5.

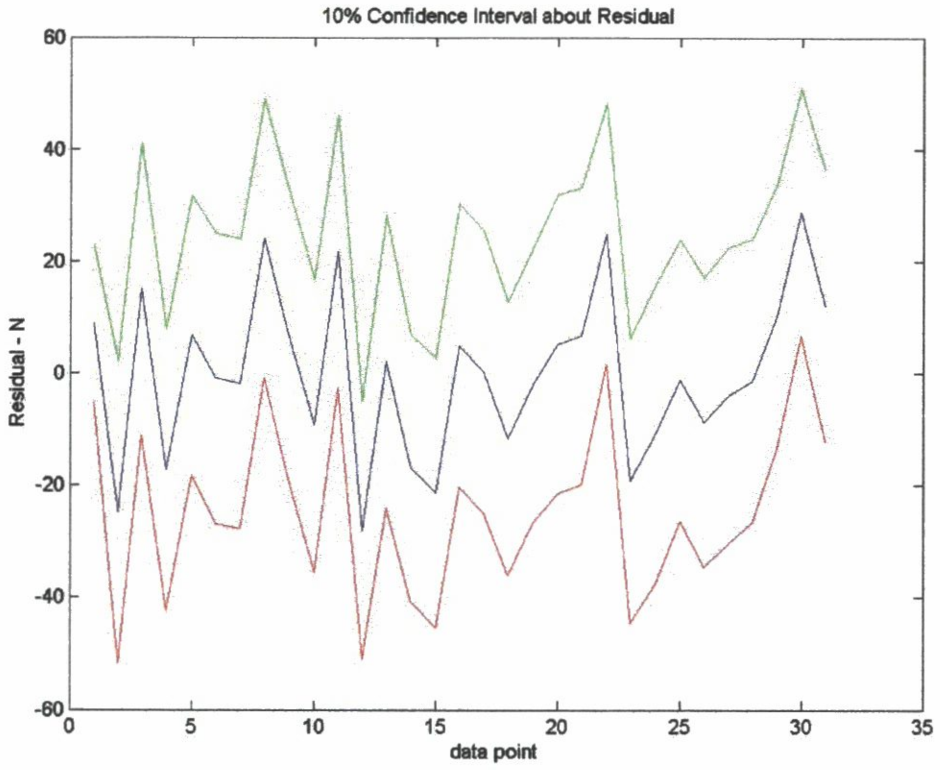


Figure 15.1.5 Residuals

The normal distribution plot of residuals (Figure 15.1.6) shows that the residuals retain some small underlying effect. This is likely to be as a result of the interaction between speed and angle-of-attack, which this experiment is not designed to quantify.

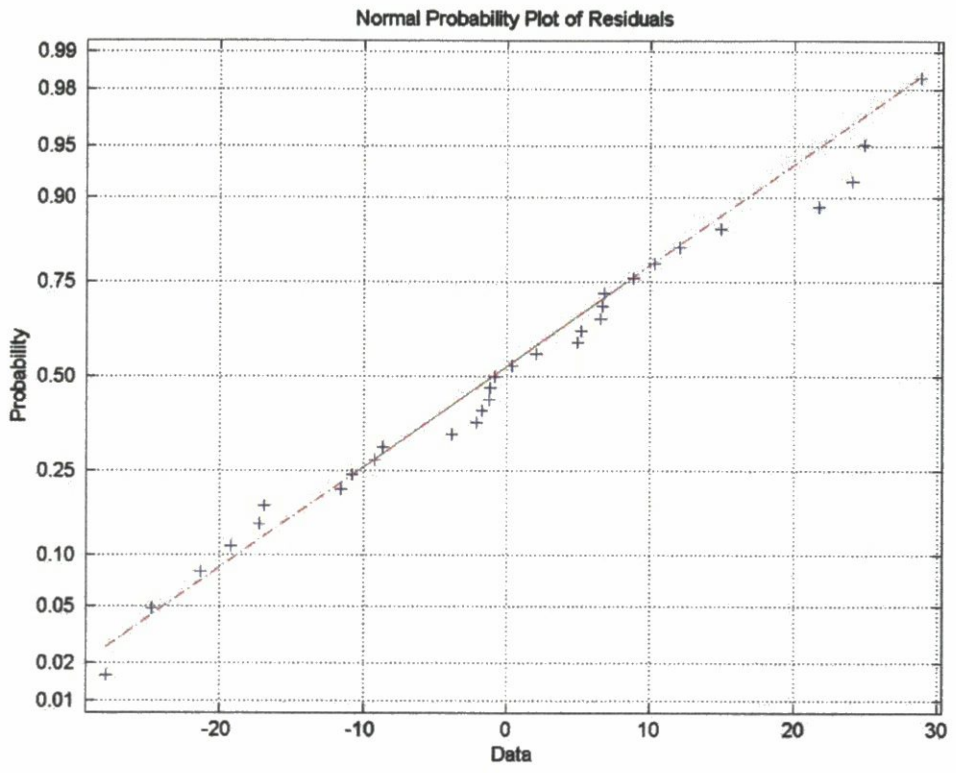


Figure 15.1.6 Normal plot of residuals

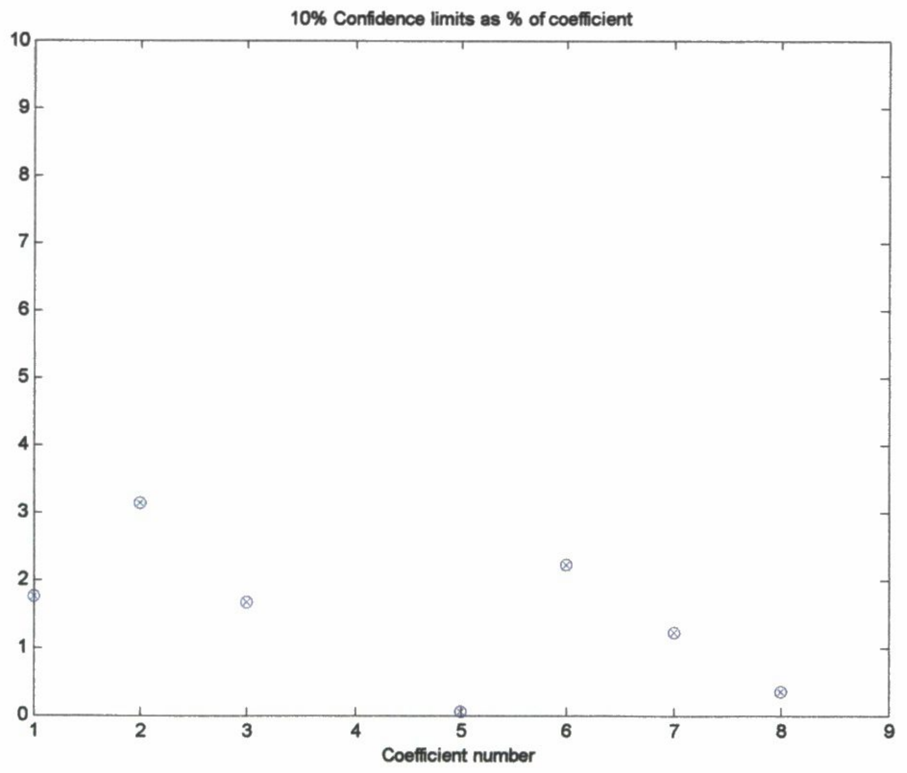


Figure 15.1.7 10% confidence interval as a percentage of the value of the coefficient

The plot of 10% confidence limits as a percentage of the values of the coefficients (Figure 2.10.1.7) confirms that parameters 4 (aft ADCP) and 9 (hydroplane angle) are ineffective at the 90% confidence level. The plot at 1% confidence limits (Figure 2.10.1.8) shows that the other factors and interactions are effective at this higher confidence level. These results broadly agree with those obtained by the previous method, although the analysis of variance indicated that only the ADCP/speed interaction has a significant effect.

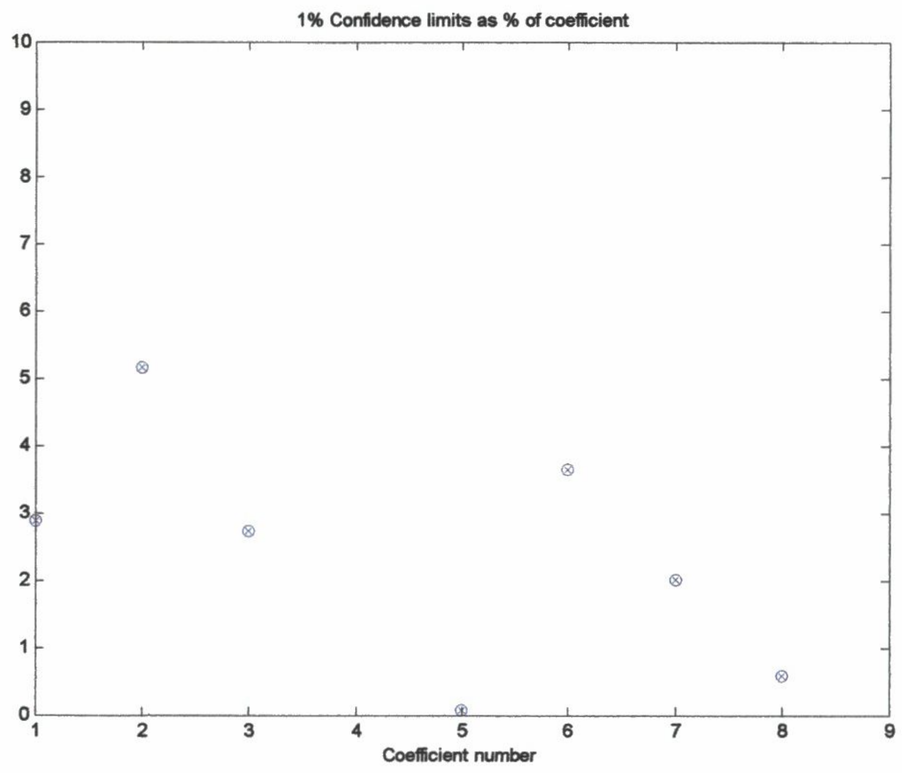


Figure 15.1.8 1% confidence interval as a percentage of the value of the coefficient

If we now remove the drag of the hull appropriate to the speed angle-of-attack and hydroplane angle appropriate to each result and analyse the residual drag the coefficients in Table 15.1.2 are obtained:

Factor number	Factor	Symbol	Value
1	Fwd ADCP	f	0.98
2	Fwd/aft interaction	fa	0.79
3	Fwd ADCP/speed interaction	fu	-0.51
4	Aft ADCP	a	1.29
5	Speed	u	2.07
6	Fwd ADCP/angle of attack interaction	falpha	0.66
7	Fwd ADCP/hydroplane angle interaction	fdelta	0.06
8	Angle of attack	alpha	0.21
9	Hydroplane angle	delta	-0.06
10	Constant	k	-10.59

Table 15.1.2 Coefficients for residual drag

The % confidence indicates that all parameters bar hydroplane angle produce a noticeable effect and that the frd and aft ADCP produce additional drag of the order of 3N over the range considered.

15.1.2 Matlab script to perform a multivariate linear regression analysis on ADCP data

```
% ADCPeffects

% Applies the regression function to the ADCP experiment data,stored in tab
%  delimited text file 'adcp_data',to provide a linear regression, where
%  cols 1 tp 9 are the levels for each of the factors and interactions
%  and col 10 is the force measurements.
%  Returns 4 vectors:
%     b=regression coefficients in the linear model y=bX
%     r=residuals, i.e y-bX
%     bint=the coefficient confidence limits at 100(1-alpha)
%     rint=the residual confidence limits at 100(1-alpha)
%  and the mean, std and max of the residuals.

close all
clear all

load 'adcp_data_2.txt';
X=adcp_data_2(:,1:9);
y=adcp_data_2(:,10);
alpha=0.1

% Add a col of 1,s to the X matrix to allow for a constant term.
```

```
for i=1:length(y);
ones(i)=1;
end

X=[X ones'];

   % Perform regression analysis

[b,bint,r,rint,stats] = regress(y,X,alpha);

%  Plot residual
plot(r)
figure
normplot(r)

%  Plot residual confidence limits
figure
plot(r)
hold
plot(rint(:,1),'r')
plot(rint(:,2),'g')
title('10% Confidence Interval about Residual')
xlabel('data point')
ylabel('Residual - N')

% Calculate % residual

r_percent=(r./y)*100;
rint_percent(:,1)=(rint(:,1)./y)*100;
rint_percent(:,2)=(rint(:,2)./y)*100;

%  Plot % residual confidence limits
figure
plot(r_percent)
hold
plot(rint_percent(:,1),'r')
plot(rint_percent(:,2),'g')
title('10% Confidence Interval about % Residual')
xlabel('data point')
ylabel('Residual - %')

% Plot coefficient confidence limits
figure
plot(bint(:,1))
hold
plot(bint(:,2),'r')
plot(b,'g')
title('10% Confidence Interval about Coefficients')
xlabel('Coefficient number')
ylabel('Coefficient - N')
```

```
%  Calculate 10% confidence limits as % of coefficients

bint_percent=abs((bint(:,1)-b)./b);

figure
plot(bint_percent,'x')
hold
plot(bint_percent,'o')
title('10% Confidence limits as % of coefficient')
xlabel('Coefficient number')
axis([1 9 0 10])
```

15.1.3 General linear regression Matlab function

```
function[b,r,bint,rint,stats] = effects(data,alpha)

% Undertakes a linear regression on the 'data' matrix, where
%  cols 1 t0 n-1 are the levels for each of the n-1 factors and interactions
%  and col 10 is the experiment measurements and produces the confidence
%  limits to the level specified by alpha, in the form 100(1-alpha).
%  Returns 4 vectors:
%     b=regression coefficients in the linear model y=bX
%     r=residuals, i.e y-bX
%     bint=the coefficient confidence limits at 100(1-alpha)
%     rint=the residual confidence limits at 100(1-alpha)
%  and the mean, std and max of the residuals.
%  Plots various graphs.

close all

d=size(data)

X=data(:,1:(d(:,2)-1));
y=data(:,d(:,2));

normplot(y)

% Add a col of 1,s to the X matrix to allow for a constant term.

for i=1:length(y);
ones(i)=1;
end

X=[X ones'];

  % Perform regression analysis

[b,bint,r,rint,stats] = regress(y,X,alpha);
```

```
% Plot residuals and confidence limits
figure
plot(r)
hold
plot(rint(:,1),'r')
plot(rint(:,2),'g')
title('10% Confidence Interval about Residual')
xlabel('data point')
ylabel('Residual - N')

% Calculate % residual

r_percent=(r./y)*100;
rint_percent(:,1)=(rint(:,1)./y)*100;
rint_percent(:,2)=(rint(:,2)./y)*100;

% Plot % residual confidence limits
figure
plot(r_percent)
hold
plot(rint_percent(:,1),'r')
plot(rint_percent(:,2),'g')
title('10% Confidence Intervals about % Residual')
xlabel('data point')
ylabel('Residual - %')

% plot residuals against normal distribution

figure
normplot(r)


% Plot coefficient confidence limits
figure
plot(bint(:,1))
hold
plot(bint(:,2),'r')
plot(b,'g')
title('10% Confidence Intervals about Coefficients')
xlabel('Coefficient number')
ylabel('Coefficient')

% Calculate 10% confidence limits as % of coefficients

bint_percent=abs((bint(:,1)-b)./b);

figure
plot(bint_percent,'x')
hold
```

```
plot(bint_percent,'o')
title('10% Confidence limits as % of coefficient')
xlabel('Coefficient number')
axis([1 (d(:,2)-1) 0 10])
```

Chapter 16

DATA LOGGING

M286 Parameters

Col No	Heading	Units
1	'Seconds'	'seconds'
2	'Depth'	'metres'
3	'MsnCtrlStat'	'none'
4	'MCLastEvent'	'none'
5	'RelA_S1Info'	'none'
6	'RelA_S2Info'	'none'
7	'RelB_S2Info'	'none'
8	'EM2000Fault'	'none'
9	'SubbotFault'	'none'
10	'MsnElementCnt'	'none'
11	'SeapamCmd'	'none'
12	'ADCP1Year'	'years'
13	'ADCP1Month'	'month'
14	'ADCP1Date'	'day'
15	'ADCP1Hour'	'hours'
16	'ADCP1Minute'	'minute'
17	'ADCP1Second'	'seconds'
18	'ADCP2Year'	'years'
19	'ADCP2Month'	'month'
20	'ADCP2Date'	'day'
21	'ADCP2Hour'	'hours'
22	'ADCP2Minute'	'minute'
23	'ADCP2Second'	'seconds'
24	'CellIdx0'	'none'
25	'Inten0'	'0.24dB'
26	'Veast0'	'mm/sec'
27	'Vnorth0'	'mm/sec'
28	'Vdown0'	'mm/sec'
29	'Verr0'	'mm/sec'
30	'ADCPVersion'	'none'
31	'ADCPRev'	'none'
32	'HeadBias'	'deg'
33	'NumWatPings'	'none'
34	'CellSize'	'cm'
35	'BlankSize'	'cm'
36	'NumCells'	'none'
37	'MinThresh'	'none'
38	'HeadAlign'	'deg'
39	'Salinity'	'sal_units'
40	'SoundSpeed'	'm/sec'
41	'ADCPTemp'	'degC'
42	'CellIdx1'	'none'
43	'Inten1'	'none'
44	'Veast1'	'mm/sec'
45	'Vnorth1'	'mm/sec'
46	'Vdown1'	'mm/sec'
47	'Verr1'	'mm/sec'

48	'CellIdx2'	'none'
49	'Inten2'	'none'
50	'Veast2'	'mm/sec'
51	'Vnorth2'	'mm/sec'
52	'Vdown2'	'mm/sec'
53	'Verr2'	'mm/sec'
54	'CellIdx3'	'none'
55	'Inten3'	'none'
56	'Veast3'	'mm/sec'
57	'Vnorth3'	'mm/sec'
58	'Vdown3'	'mm/sec'
59	'Verr3'	'mm/sec'
60	'CellIdx4'	'none'
61	'Inten4'	'none'
62	'Veast4'	'mm/sec'
63	'Vnorth4'	'mm/sec'
64	'Vdown4'	'mm/sec'
65	'Verr4'	'mm/sec'
66	'CellIdx5'	'none'
67	'Inten5'	'none'
68	'Veast5'	'mm/sec'
69	'Vnorth5'	'mm/sec'
70	'Vdown5'	'mm/sec'
71	'Verr5'	'mm/sec'
72	'CellIdx6'	'none'
73	'Inten6'	'none'
74	'Veast6'	'mm/sec'
75	'Vnorth6'	'mm/sec'
76	'Vdown6'	'mm/sec'
77	'Verr6'	'mm/sec'
78	'CellIdx7'	'none'
79	'Inten7'	'none'
80	'Veast7'	'mm/sec'
81	'Vnorth7'	'mm/sec'
82	'Vdown7'	'mm/sec'
83	'Verr7'	'mm/sec'
84	'CellIdx8'	'none'
85	'Inten8'	'none'
86	'Veast8'	'mm/sec'
87	'Vnorth8'	'mm/sec'
88	'Vdown8'	'mm/sec'
89	'Verr8'	'mm/sec'
90	'CellIdx9'	'none'
91	'Inten9'	'none'
92	'Veast9'	'mm/sec'
93	'Vnorth9'	'mm/sec'
94	'Vdown9'	'mm/sec'
95	'Verr9'	'mm/sec'
96	'CellIdx10'	'none'
97	'Inten10'	'none'
98	'Veast10'	'mm/sec'
99	'Vnorth10'	'mm/sec'
100	'Vdown10'	'mm/sec'
101	'Verr10'	'mm/sec'
102	'CellIdx11'	'none'
103	'Inten11'	'none'
104	'Veast11'	'mm/sec'
105	'Vnorth11'	'mm/sec'
106	'Vdown11'	'mm/sec'
107	'Verr11'	'mm/sec'

108	'CellIdx12'	'none'
109	'Inten12'	'none'
110	'Veast12'	'mm/sec'
111	'Vnorth12'	'mm/sec'
112	'Vdown12'	'mm/sec'
113	'Verr12'	'mm/sec'
114	'CellIdx13'	'none'
115	'Inten13'	'none'
116	'Veast13'	'mm/sec'
117	'Vnorth13'	'mm/sec'
118	'Vdown13'	'mm/sec'
119	'Verr13'	'mm/sec'
120	'CellIdx14'	'none'
121	'Inten14'	'none'
122	'Veast14'	'mm/sec'
123	'Vnorth14'	'mm/sec'
124	'Vdown14'	'mm/sec'
125	'Verr14'	'mm/sec'
126	'CellIdx15'	'none'
127	'Inten15'	'none'
128	'Veast15'	'mm/sec'
129	'Vnorth15'	'mm/sec'
130	'Vdown15'	'mm/sec'
131	'Verr15'	'mm/sec'
132	'CellIdx0_2'	'none'
133	'Inten0_2'	'0.24dB'
134	'Veast0_2'	'mm/sec'
135	'Vnorth0_2'	'mm/sec'
136	'Vdown0_2'	'mm/sec'
137	'Verr0_2'	'mm/sec'
138	'ADCPVer_2'	'none'
139	'ADCPRev_2'	'none'
140	'HeadBias_2'	'deg'
141	'NumWtPin_2'	'none'
142	'CellSiz_2'	'cm'
143	'BlnkSiz_2'	'cm'
144	'NumCells_2'	'none'
145	'MinThrsh_2'	'none'
146	'HeadAlign_2'	'deg'
147	'Salinity_2'	'sal_units'
148	'SoundSpd_2'	'm/sec'
149	'ADCPTemp_2'	'degC'
150	'CellIdx1_2'	'none'
151	'Inten1_2'	'none'
152	'Veast1_2'	'mm/sec'
153	'Vnorth1_2'	'mm/sec'
154	'Vdown1_2'	'mm/sec'
155	'Verr1_2'	'mm/sec'
156	'CellIdx2_2'	'none'
157	'Inten2_2'	'none'
158	'Veast2_2'	'mm/sec'
159	'Vnorth2_2'	'mm/sec'
160	'Vdown2_2'	'mm/sec'
161	'Verr2_2'	'mm/sec'
162	'CellIdx3_2'	'none'
163	'Inten3_2'	'none'
164	'Veast3_2'	'mm/sec'
165	'Vnorth3_2'	'mm/sec'
166	'Vdown3_2'	'mm/sec'
167	'Verr3_2'	'mm/sec'

168	'Cellldx4_2'	'none'
169	'Inten4_2'	'none'
170	'Veast4_2'	'mm/sec'
171	'Vnorth4_2'	'mm/sec'
172	'Vdown4_2'	'mm/sec'
173	'Verr4_2'	'mm/sec'
174	'Cellldx5_2'	'none'
175	'Inten5_2'	'none'
176	'Veast5_2'	'mm/sec'
177	'Vnorth5_2'	'mm/sec'
178	'Vdown5_2'	'mm/sec'
179	'Verr5_2'	'mm/sec'
180	'Cellldx6_2'	'none'
181	'Inten6_2'	'none'
182	'Veast6_2'	'mm/sec'
183	'Vnorth6_2'	'mm/sec'
184	'Vdown6_2'	'mm/sec'
185	'Verr6_2'	'mm/sec'
186	'Cellldx7_2'	'none'
187	'Inten7_2'	'none'
188	'Veast7_2'	'mm/sec'
189	'Vnorth7_2'	'mm/sec'
190	'Vdown7_2'	'mm/sec'
191	'Verr7_2'	'mm/sec'
192	'Cellldx8_2'	'none'
193	'Inten8_2'	'none'
194	'Veast8_2'	'mm/sec'
195	'Vnorth8_2'	'mm/sec'
196	'Vdown8_2'	'mm/sec'
197	'Verr8_2'	'mm/sec'
198	'Cellldx9_2'	'none'
199	'Inten9_2'	'none'
200	'Veast9_2'	'mm/sec'
201	'Vnorth9_2'	'mm/sec'
202	'Vdown9_2'	'mm/sec'
203	'Verr9_2'	'mm/sec'
204	'Cellldx10_2'	'none'
205	'Inten10_2'	'none'
206	'Veast10_2'	'mm/sec'
207	'Vnorth10_2'	'mm/sec'
208	'Vdown10_2'	'mm/sec'
209	'Verr10_2'	'mm/sec'
210	'Cellldx11_2'	'none'
211	'Inten11_2'	'none'
212	'Veast11_2'	'mm/sec'
213	'Vnorth11_2'	'mm/sec'
214	'Vdown11_2'	'mm/sec'
215	'Verr11_2'	'mm/sec'
216	'Cellldx12_2'	'none'
217	'Inten12_2'	'none'
218	'Veast12_2'	'mm/sec'
219	'Vnorth12_2'	'mm/sec'
220	'Vdown12_2'	'mm/sec'
221	'Verr12_2'	'mm/sec'
222	'Cellldx13_2'	'none'
223	'Inten13_2'	'none'
224	'Veast13_2'	'mm/sec'
225	'Vnorth13_2'	'mm/sec'
226	'Vdown13_2'	'mm/sec'
227	'Verr13_2'	'mm/sec'

228	'CellIdx14_2'	'none'
229	'Inten14_2'	'none'
230	'Veast14_2'	'mm/sec'
231	'Vnorth14_2'	'mm/sec'
232	'Vdown14_2'	'mm/sec'
233	'Verr14_2'	'mm/sec'
234	'CellIdx15_2'	'none'
235	'Inten15_2'	'none'
236	'Veast15_2'	'mm/sec'
237	'Vnorth15_2'	'mm/sec'
238	'Vdown15_2'	'mm/sec'
239	'Verr15_2'	'mm/sec'
240	'ADCPR1'	'metres'
241	'ADCPR2'	'metres'
242	'ADCPR3'	'metres'
243	'ADCPR4'	'metres'
244	'Vel_north'	'mm/sec'
245	'Vel_east'	'mm/sec'
246	'ADCPAlt'	'metres'
247	'ADCPVelMode'	'none'
248	'ADCPErrors'	'none'
249	'ADCPNodeFau'	'none'
250	'Vwater_nort'	'm/s'
251	'Vwater_east'	'm/s'
252	'Vfilt_north'	'm/s'
253	'Vfilt_east'	'metres'
254	'ADCPHeading'	'radians'
255	'ADCPPitch'	'radians'
256	'ADCPRoll'	'radians'
257	'NumADCPCell'	'none'
258	'ADCPSpare1'	'none'
259	'ADCPSpare2'	'none'
260	'ADCP2R1'	'metres'
261	'ADCP2R2'	'metres'
262	'ADCP2R3'	'metres'
263	'ADCP2R4'	'metres'
264	'Vel2_north'	'mm/sec'
265	'Vel2_east'	'mm/sec'
266	'ADCP2Alt'	'metres'
267	'ADCP2VelMod'	'none'
268	'ADCP2Errors'	'none'
269	'ADCP2NodeFl'	'none'
270	'Vwatr_nrth2'	'm/s'
271	'Vwatr_east2'	'm/s'
272	'Vfilt_nrth2'	'm/s'
273	'Vfilt_east2'	'metres'
274	'ADCP2Headin'	'radians'
275	'ADCP2Pitch'	'radians'
276	'ADCP2Roll'	'radians'
277	'NumADCP2Cel'	'none'
278	'ADCP2Spare1'	'none'
279	'ADCP2Spare2'	'none'
280	'dep_ctrl_de'	'metres'
281	'pitch_dem'	'radians'
282	'splane_pos'	'radians'
283	'splane_dem'	'radians'
284	'DepCtldepth'	'metres'
285	'DepCtlADCPa'	'metres'
286	'DepCtlAltit'	'metres'
287	'dep_limit_s'	'none'

288	'DCdepth_sta'	'none'
289	'DCNodeFault'	'none'
290	'AcLatitude'	'degrees'
291	'AcLongitude'	'degrees'
292	'Latitude'	'degrees'
293	'Longitude'	'degrees'
294	'GPSLatitude'	'degrees'
295	'GPSLongitud'	'degrees'
296	'TSLF'	'seconds'
297	'FixType'	'none'
298	'GPSPower'	'none'
299	'GPSNodeFaul'	'none'
300	'Homing1'	'radians'
301	'Homing2'	'metres'
302	'Homing3'	'metres'
303	'Homing4'	'radians'
304	'Homing5'	'radians'
305	'Homing6'	'degrees'
306	'Homing7'	'degrees'
307	'Homing8'	'none'
308	'HomingFault'	'none'
309	'mtr_I_stpnt'	'amperes'
310	'prop_rpm'	'rpm'
311	'prop_torque'	'n/m'
312	'prop_power'	'watts'
313	'vhcl_spd'	'm/s'
314	'mtr_spare1'	'none'
315	'mtr_spare2'	'none'
316	'MtrNodeFaul'	'none'
317	'MtrNodeInfo'	'none'
318	'MtrSpare3'	'none'
319	'Roll'	'radians'
320	'Pitch'	'radians'
321	'Heading'	'radians'
322	'Misalign'	'radians'
323	'YawRate'	'radians'
324	'PitchRate'	'radians'
325	'RollRate'	'radians'
326	'AttStatus'	'none'
327	'AttSpare1'	'none'
328	'AttSpare2'	'none'
329	'INSLat'	'degrees'
330	'INSLong'	'degrees'
331	'INSDepth'	'metres'
332	'INSVz'	'metres/sec'
333	'INSVNorth'	'metres/sec'
334	'INSVEast'	'metres/sec'
335	'MisalignStD'	'radians'
336	'SerNumLow'	'none'
337	'SerNumHi'	'none'
338	'RateOverld'	'none'
339	'RangeToGo'	'metres'
340	'RudderCtrlD'	'radians'
341	'PCRudderPos'	'radians'
342	'PCHeadingCt'	'radians'
343	'PosCtrlDemN'	'degrees'
344	'PosCtrlDemE'	'degrees'
345	'PosHeadingI'	'radians'
346	'PositionGot'	'none'
347	'PositionSta'	'none'

348	'PosFault'	'none'
349	'battery_V'	'volts'
350	'Pwr_total_I'	'amperes'
351	'PwrTemp1'	'amperes'
352	'PwrTemp2'	'amperes'
353	'PwrTemp3'	'amperes'
354	'PwrTemp4'	'degreesC'
355	'PwrLeak1'	'counts'
356	'PwrFault1'	'none'
357	'PwrInfo1'	'none'
358	'PwrSpare1'	'none'
359	'PwrBatI1'	'amperes'
360	'PwrBatI2'	'amperes'
361	'PwrBatI3'	'amperes'
362	'PwrBatI4'	'amperes'
363	'PwrBatI5'	'amperes'
364	'PwrBatI6'	'amperes'
365	'PwrLeak2'	'counts'
366	'PwrFault2'	'none'
367	'PwrInfo2'	'none'
368	'PwrSpare2'	'none'
369	'PwrDCI1'	'amperes'
370	'PwrDCI2'	'amperes'
371	'PwrDCI3'	'amperes'
372	'PwrMotorI'	'amperes'
373	'PwrTemp5'	'amperes'
374	'PwrTemp6'	'degreesC'
375	'PwrHPSNTmp'	'degreesC'
376	'PwrFault3'	'none'
377	'PwrInfo3'	'none'
378	'PwrSpare3'	'none'
379	'Range'	'metres'
380	'FiltRange'	'metres'
381	'RangeTrend'	'metres'
382	'RangeAux1'	'none'
383	'RangeAux2'	'none'
384	'RangeAux3'	'none'
385	'RangeAux4'	'degreesC'
386	'CollisImm'	'none'
387	'CollisFlgCn'	'none'

Chapter 17

TRIALS DATA PRE-PROCESSING

17.1 Matlab script for data drop-out removal

```
% 'remove_data_dropouts' finds and removes any row of a data matrix 'x' that has
%               a '-999' entry.

%______________________________________________________________________

% 1. Find the location of the -999.
%   Note x is transposed as the location is read column by column.

x2=find(x'==-999);

% 2. Calculate the row number of the associated -999s.
%   Size(x,2) used so any sized matrix can be used.

x3=ceil(x2/size(x,2));

% 3. Removes any replication of rows caused by more than 1 -999 in any row.
%   Works by establishing a sliding window.

x4=x3(1:length(x3)-1)-x3(2:length(x3));

% 4. Finds the locations of any zeros generated.
%   Note need to add one as the first number in the vector ignored

x5=find(x4==0)+1;

% 5. Removes the rows associated with duplicate -999 (ie big matrix row numbers)

x3(x5)=[];

% 6. Blitzes the rows of the big matrix with -999 in them

x(x3,:)=[];
```

Chapter 18

TRIAL DATA PROCESSING

18.1 Matlab script for determining the relationship between angle-of-attack, hydroplane angle and speed

```
%       Script 'Angles1' determines the relationship between hull angle-of-attack,
%               hydroplane angle and speed.

%_______________________________________________________________________

% 1. Establish and clean matrix of relevant data, viz:
%               time from start
%               Doppler log bin 0, 1, 2, 3 easterly, northerly and down velocities
%               propeller rotation rate

%       Establish raw time in sec from start.

tr=d(:,1)-d(1,1);

%       Determine raw easterly, northerly and down velocities in m/s from bins 0,1,2,3
of
%               Doppler log,where
%                bin 0 is speed over ground
%                bins 1,2,3 are speed through water with bin 1 closest to vehicle.

Ue0r=d(:,26)/1000; Un0r=d(:,27)/1000; Ud0r=d(:,28)/1000;

Ue1r=d(:,44)/1000; Un1r=d(:,45)/1000; Ud1r=d(:,46)/1000;
Ue2r=d(:,50)/1000; Un2r=d(:,51)/1000; Ud2r=d(:,52)/1000;
Ue3r=d(:,56)/1000; Un3r=d(:,57)/1000; Ud3r=d(:,58)/1000;
Ue4r=d(:,56)/1000; Un4r=d(:,57)/1000; Ud4r=d(:,58)/1000;

%       Establish propeller rotation rate so times for speed change can be determined.

Prr=d(:,310);

% Establish hull angle-of-attack from pitch data from ADCP 1 & 2 and logged 'Pitch'
data.

Aoa1r=d(:,255);
Aoa2r=d(:,275);
Aoapr=d(:,320);

% Establish hydroplane angles.
```

```
Har=d(:,282);

% Clean time data, re allocate to vectors and convert angles from radians to degrees.

data=[tr Ue0r Un0r Ud0r Ue1r Un1r Ud1r Ue2r Un2r Ud2r Ue3r Un3r Ud3r Ue4r
Un4r Ud4r...
    Prr Aoa1r Aoapr Har];

x=data;

remove_data_dropouts;
purge_data;
remove_zeros;

t=x(:,1);
Ue0=x(:,2);
Un0=x(:,3);
Ud0=x(:,4);
Ue1=x(:,5);
Un1=x(:,6);
Ud1=x(:,7);
Ue2=x(:,8);
Un2=x(:,9);
Ud2=x(:,10);
Ue3=x(:,11);
Un3=x(:,12);
Ud3=x(:,13);
Ue4=x(:,14);
Un4=x(:,15);
Ud4=x(:,16);

Pr=x(:,17);
Aoa1=x(:,18)*180/pi;
Aoap=x(:,19)*180/pi;
Ha=x(:,20)*180/pi;

% Plot results

figure
plot(t,Pr)
Title('m286 - Prop rpm')
xlabel('Time from start - secs')


figure
plot(t,Aoa1)
hold
plot(t,Aoap,'g')
legend('Aoa1','Aoap')
title('m286 - Angle-of-attack')
```

```
figure
plot(t,Ha)
title('m286 - Hydroplane angle')
xlabel('Time from start - secs')
ylabel('Degrees')


%______________________________________________________________

% 2. Estimate speed through the water from bins 0 to 4

s0=sqrt(Ue0.^2+Un0.^2+Ud0.^2);
s1=sqrt(Ue1.^2+Un1.^2+Ud1.^2);
s2=sqrt(Ue2.^2+Un2.^2+Ud2.^2);
s3=sqrt(Ue3.^2+Un3.^2+Ud3.^2);
s4=sqrt(Ue4.^2+Un4.^2+Ud4.^2);

%_________________________________________________________________
__
% 3.  Determine times at which speed changes made from propeller rpm data.

figure
plot(t,Pr)
title('Trial m286 - Propeller Rotation Rate')
xlabel('Time from start - sec')
ylabel('Propeller rpm')



% Differentiate prop speed and overlay.

hold
plot(t(1:length(diff(Pr))),((diff(Pr)+230)),'k')
legend('Propeller Rotation Rate','Rate of change of Prop Rotation rate+230')



%       Times for speed steps are:

ta=3500;
tb=3700;
tc=3800;
td=4000;
te=4050;
tf=4300;
tg=4350;
th=4600;
ti=4650;
```

```
tj=4900;
tk=4950;
tl=5200;
tm=5300;
tn=5614;
to=5614;
tp=5800;
tq=5850;
tr=6130;
ts=6150;
tt=6420;
tu=6480;
tv=6720;
tw=6750;
tx=7020;
ty=7050;
tz=7330;
taa=7350;
tab=7630;

%_______________________________________________________________________

% 4. Determine  matrix of mean speed where columns corresponds to bins 0,1,2,3,4
%        and rows correspond to speed steps

% Establish matrix were each row corresponds to the start and stop time
%                 of each speed step.

t_step=[ta tb
  tc td
  te tf
  tg th
  ti tj
  tk tl
  tm tn
  to tp
  tq tr
  ts tt
  tu tv
  tw tx
  ty tz
  taa tab];

  e=size(t_step);

  % Establish matrix i giving rows of indices of time vector corresponding to speed
steps.

  for j=1:e(1,1)
    i(j,1)=max(find(t<t_step(j,1)+1));
```

```
                i(j,2)=max(find(t<t_step(j,2)+1));
  end

%_______________________________________________________________________________

%     5. Establish indices for each speed step after data corrected by removing that
%             associated with change in speed. (See script 'Speed.m' for explanation).

for j=1:e(1,1);
  t_step(j,1)=t_step(j,1)+120;
end

for j=1:e(1,1)
    i(j,1)=max(find(t<t_step(j,1)+1));
                i(j,2)=max(find(t<t_step(j,2)+1));
  end

% Recalculate speed matrices, etc based on reduced data sets.
 % Establish matrices of speed/standard deviation data where each column contains
the mean/std dev
  %   speed for each speed step corresponding to each range bin and send to work
space.

  s=[s0 s1 s2 s3 s4];

  for k=1:5
    for j=1:e(1,1)
    av_speed(j,k)=mean(s(i(j,1):i(j,2),k));
    speed_std(j,k)=std(s(i(j,1):i(j,2),k));
  end
end

% _______________________________________________________________________________

% 6. Establish vectors of average speed & standard deviation using all data in bins
%       1 to 4 for each speed step.

  s_best=[s1 s2 s3 s4];

    for j=1:e(1,1)
   av_speed_step(j)=mean([s_best(i(j,1):i(j,2),1)' s_best(i(j,1):i(j,2),2)'...
       s_best(i(j,1):i(j,2),3)' s_best(i(j,1):i(j,2),4)']);
   std_speed_step(j)=std([s_best(i(j,1):i(j,2),1)' s_best(i(j,1):i(j,2),2)'...
       s_best(i(j,1):i(j,2),3)' s_best(i(j,1):i(j,2),4)']);
     end

% Send speed-per-step data to work space and save it as ascii files

av_speed_step
std_speed_step
```

```
save mean-speed_step av_speed_step -ascii;
save std_speeds_step std_speed_step -ascii;

% Establish speed matrices, with cols corresponding to bin 1 to 4 only and rows to
time,
%       and establish 95% confidence level using Student t test.

%___________________________________________________

% Initialise h,sig,ci. (Script won't work on switch-on without it).

speed_step=s(i(1,1):i(1,2),2:5)
mean_speed_step(1)=mean(mean(speed_step));
for n=1:length(speed_step)
  s1(n)=mean(speed_step(n,:));
end
[h,sig,ci]=ttest(s1,mean_speed_step(1));

%___________________________________________________

speed95=[];
for j=1:e(1,1)
speed_step=s(i(j,1):i(j,2),2:5);
mean_speed_step(j)=mean(mean(speed_step));
[h(j),sig(j),ci]=ttest(mean(s(i(j,1):i(j,2),2:5)),mean_speed_step(j));
speed95=[speed95
  ci];
end

figure
plot(mean_speed_step)
hold
plot(speed95(:,1),'r--')
plot(speed95(:,2),'g--')
xlabel('Speed step')
ylabel('Mean Speed - m/s')
title('m286 - Mean speed per speed step - all bins - acceleration/deceleration data
removed')
legend('mean','+95% conf limit','-95% conf limit')



save mean-speeds av_speed -ascii;
save standard_deviation_speeds speed_std -ascii;

%
______________________________________________________________

%       7. establish relationship between aoa, hydroplane angle and speed.
%       Establish mean aoa and hydroplane angle for each speed step.

for j=1:e(1,1)
  av_aoa_step(j)=mean([Aoap(i(j,1):i(j,2))]);
  std_aoa_step(j)=std([Aoap(i(j,1):i(j,2))]);
  av_ha_step(j)=mean([Ha(i(j,1):i(j,2))]);
  std_ha_step(j)=std([Ha(i(j,1):i(j,2))]);

     end

% Send angle-per-step data to work space, plot it and save it as ascii files

av_aoa_step
std_aoa_step
av_ha_step
std_ha_step

% Establish aoa vectors for each speed step and establish 95% confidence level
%       using Student t test.

aoa95=[];
for j=1:e(1,1)
aoa_step=Aoap(i(j,1):i(j,2));
mean_aoa_step(j)=mean(aoa_step);
[h(j),sig(j),ci]=ttest(aoa_step,mean_aoa_step(j));
aoa95=[aoa95
  ci];
end

figure
plot(mean_aoa_step)
hold
plot(aoa95(:,1),'r--')
plot(aoa95(:,2),'g--')
xlabel('Speed step')
ylabel('Aoa - deg')
title('m286 - Mean aoa per speed step')
legend('mean','+95% conf limit','-95% conf limit')

save mean_aoa_step av_aoa_step -ascii;
save standard_deviation_aoa_step std_aoa_step -ascii;

figure
plot(av_speed_step(1:7), av_aoa_step(1:7))
xlabel('Speed - m/s')
ylabel('Angle of atack - degrees')
title('m286 - Angle-of-attack as a function of speed')
legend('Outward leg',4)

hold
```

```
plot(av_speed_step(7:14), av_aoa_step(7:14),'r--')
xlabel('Speed')
ylabel('Angle of atack')
title('m286 - Angle-of-attack as a function of speed')
legend('Outward leg', 'Return Leg',4)

% Establish ha vectors for each speed step and establish 95% confidence level
%       using Student t test.

Ha95=[];
for j=1:e(1,1)
Ha_step=Ha(i(j,1):i(j,2));
mean_Ha_step(j)=mean(Ha_step);
[h(j),sig(j),ci]=ttest(Ha_step,mean_Ha_step(j));
Ha95=[Ha95
  ci];
end

figure
plot(mean_Ha_step)
hold
plot(Ha95(:,1),'r--')
plot(Ha95(:,2),'g--')
xlabel('Speed step')
ylabel('Hydroplane angle - deg')
title('m286 - Mean hydroplane angle per speed step')
legend('mean','+95% conf limit','-95% conf limit')

save mean_ha_step av_ha_step -ascii;
save standard_deviation_ha_step std_ha_step -ascii;

figure
plot(av_speed_step(1:7), av_ha_step(1:7))
xlabel('Speed - m/s')
ylabel('Hydroplane angle - degrees')
title('m286 - Hydroplane angle as a function of speed')

legend('Outward leg',4)

hold
plot(av_speed_step(7:14), av_ha_step(7:14),'r--')
legend('Outward leg', 'Return Leg',4)

%____________________

figure
plot(av_aoa_step(1:7), av_ha_step(1:7))
xlabel('Angle of atack - degrees')
ylabel('Hydroplane angle - degrees')
title('m286 - Angle-of-attack as a function of hydroplane angle')
```

```
legend('Outward leg',4)

hold
plot(av_aoa_step(7:14), av_ha_step(7:14),'r--')
legend('Outward leg', 'Return Leg',4)

%______________________________________________________________

% 3D plot of Speed, Ha, AoA.

figure
plot3(av_speed_step,av_aoa_step,av_ha_step)
xlabel('Speed - m/s')
ylabel('Aoa - deg')
zlabel('Ha - deg')
```

18.2 Matlab script for determining the speeds at each step

```
% 'Script 'Speed_step1' establishes best way to estimate speed as f(time)
%             using data from trial m286.

%______________________________________________________________

% 1. Establish and clean matrix of relevent data, viz:
%             time from start
%             doppler log bin 0, 1, 2, 3 easterly, northerly and down velocities
%             propeller rotation rate

%      Establish raw time in sec from start.

tr=d(:,1)-d(1,1);

%      Determine raw easterly, northerly and down velocities in m/s from bins 0,1,2,3
of
%             doppler log,where
%              bin 0 is speed over ground
%              bins 1,2,3 are speed through water with bin 1 closest to vehicle.

Ue0r=d(:,26)/1000; Un0r=d(:,27)/1000; Ud0r=d(:,28)/1000;

Ue1r=d(:,44)/1000; Un1r=d(:,45)/1000; Ud1r=d(:,46)/1000;
Ue2r=d(:,50)/1000; Un2r=d(:,51)/1000; Ud2r=d(:,52)/1000;
Ue3r=d(:,56)/1000; Un3r=d(:,57)/1000; Ud3r=d(:,58)/1000;
Ue4r=d(:,56)/1000; Un4r=d(:,57)/1000; Ud4r=d(:,58)/1000;

%      Establish propeller rotation rate so times for speed change can be determined.

Prr=d(:,310);

% Establish data matrix, clean data, demonstrate graphicaly the efect of
%       cleaning the data, and re allocate to vectors for ease of processing

data=[tr Ue0r Un0r Ud0r Ue1r Un1r Ud1r Ue2r Un2r Ud2r Ue3r Un3r Ud3r Ue4r
Un4r Ud4r Prr];

x=data;
%____________
plot(x(:,2))

%____________

remove_data_dropouts;
purge_data;

%___________
plot(x(:,2))

%___________

remove_zeros;

%___________
plot(x(:,2))

%___________

t=x(:,1);
Ue0=x(:,2);
Un0=x(:,3);
Ud0=x(:,4);
Ue1=x(:,5);
Un1=x(:,6);
Ud1=x(:,7);
Ue2=x(:,8);
Un2=x(:,9);
Ud2=x(:,10);
Ue3=x(:,11);
Un3=x(:,12);
Ud3=x(:,13);
Ue4=x(:,14);
Un4=x(:,15);
Ud4=x(:,16);

Pr=x(:,17);

%_____________________________________________________________

% 2. Estimate speed through the water from bins 1 to 4 and plot

s1=sqrt(Ue1.^2+Un1.^2+Ud1.^2);
```

```
figure
plot(t,s1)
axis([0,max(t),0,2.5])
title('Trial m286 - Vehicle Speed Through Water')
xlabel('Time - sec')
ylabel('Speed - m/s')
legend('Bin 1');



s2=sqrt(Ue2.^2+Un2.^2+Ud2.^2);
hold
plot(t,s2,'r')
axis([0,max(t),0,2.5])
legend('Bin 1','Bin 2');



s3=sqrt(Ue3.^2+Un3.^2+Ud3.^2);

plot(t,s3,'g')
axis([0,max(t),0,2.5])
legend('Bin 1','Bin 2','Bin 3');




s4=sqrt(Ue4.^2+Un4.^2+Ud4.^2);

plot(t,s4,'c')
axis([0,max(t),0,2.5])
legend('Bin 1','Bin 2','Bin 3','Bin 4');



plot(t,Pr/200,'k')

legend('Bin 1','Bin 2','Bin 3','Bin 4','Prop rpm/200');



%___________________________________________________________________

% 3. Determine speed over ground and plot.

s0=sqrt(Ue0.^2+Un0.^2+Ud0.^2);
figure
plot(t,s0)
axis([0,max(t),0,2.5])
title('Trial m286 - Vehicle Speed Over ground')
```

```
xlabel('Time - sec')
ylabel('Speed - m/s')


%________________________________________________________________
_

% 4.  Determine times at which speed changes made from propeller rpm data.

figure
plot(t,Pr)
title('Trial m286 - Propeller Rotation Rate')
xlabel('Time - sec')
ylabel('Propeller rpm')


% Differentiate prop speed and overlay.

hold
plot(t(1:length(diff(Pr))),((diff(Pr)+230)),'k')
legend('Propeller Rotation Rate','Rate of change of Prop Rotation rate+230')


%       Times for speed steps are:

ta=3500;
tb=3700;
tc=3800;
td=4000;
te=4050;
tf=4300;
tg=4350;
th=4600;
ti=4650;
tj=4900;
tk=4950;
tl=5200;
tm=5300;
tn=5614;
to=5614;
tp=5800;
tq=5850;
tr=6130;
ts=6150;
tt=6420;
tu=6480;
tv=6720;
```

```
tw=6750;
tx=7020;
ty=7050;
tz=7330;
taa=7350;
tab=7630;

%________________________________________________________________

% 5. Determine  matrix of mean speed where columns corresponds to bins 0,1,2,3,4
%       and rows correspond to speed steps

% Establish matrix were each row corresponds to the start and stop time
%                 of each speed step.

t_step=[ta tb
  tc td
  te tf
  tg th
  ti tj
  tk tl
  tm tn
  to tp
  tq tr
  ts tt
  tu tv
  tw tx
  ty tz
  taa tab];

  e=size(t_step);

  % Establish matrix i giving rows of indices of time vector corresponding to speed
steps.

  for j=1:e(1,1)
    i(j,1)=max(find(t<t_step(j,1)+1));
                 i(j,2)=max(find(t<t_step(j,2)+1));
  end

%____________________________________________________________________

% 6. Establish matrices of speed/standard deviation data where each column contains
the
%                 mean/std dev speed for each speed step corresponding to each range
bin
%                 and send to work space.

  s=[s0 s1 s2 s3 s4];
  for k=1:5
  for j=1:e(1,1)
    av_speed(j,k)=mean(s(i(j,1):i(j,2),k));
    speed_std(j,k)=std(s(i(j,1):i(j,2),k));
  end
end

% Send speed-per-step data to work space, plot it and save it as ascii files

av_speed
speed_std

figure
plot(av_speed(:,1))
xlabel('Speed step')
ylabel('Average Speed')
title('Speed per step')
legend('bin 0')


hold
plot(av_speed(:,2),'r')
legend('bin 0','bin 1')


plot(av_speed(:,3),'y')
legend('bin 0','bin 1','bin 2')

plot(av_speed(:,4),'g')
legend('bin 0','bin 1','bin 2','bin 3')

plot(av_speed(:,5),'k')
legend('bin 0','bin 1','bin 2','bin 3','bin 4')


save mean-speeds av_speed -ascii;
save standard_deviation_speeds speed_std -ascii;



% Plot speed over time-step j from bin (k-1)

j=1; k=1;

figure
plot(t(i(j,1):i(j,2)),s(i(j,1):i(j,2),k))
title('Speed step 1, bin 0')
```

```
%_______________________________________________________________

% 7. Determine whether speed in bins 0 to 4 represent random noise superimposed on
a
%               constant speed. This would be so if distribution is normal.

% Plot normal probability of speed in bin 0 at step 1
figure
normplot(s(i(j,1):i(j,2),k))
title('Normal Probability Plot of Bin O over Speed Step 1')


% Plot speed and normal probability of speed  in bin 3 at step 5

j=5; k=4;
figure
plot(t(i(j,1):i(j,2)),s(i(j,1):i(j,2),k))
title('Speed Step 1, Bin 0')


figure
normplot(s(i(j,1):i(j,2),k))
title('Normal Probability Plot of Bin O over Speed Step 1')


% Both of the above show that distributions are not normal since speed has not
%       had time to settle after step., therefore, need to ignore first 120 secs of data.
%       (In longer term need to repeat experiment with longer time steps to give a
bigger
%       number of data points at constant speed).

%_______________________________________________________________


%       8. Correct dat for each speed step by removing that associated with change in
speed.

% Add 120 to all t_step(:,1) entries and find the corresponding data indices.

for j=1:e(1,1);
  t_step(j,1)=t_step(j,1)+120;
end

for j=1:e(1,1)
    i(j,1)=max(find(t<t_step(j,1)+1));
                i(j,2)=max(find(t<t_step(j,2)+1));
  end

% Recalculate speed matrices, etc based on reduced data sets.
```

```
 % Establish matrices of speed/standard deviation data where each column contains
the mean/std dev
   %    speed for each speed step corresponding to each range bin and send to work
space.

   s=[s0 s1 s2 s3 s4];

   for k=1:5
   for j=1:e(1,1)
     av_speed(j,k)=mean(s(i(j,1):i(j,2),k));
     speed_std(j,k)=std(s(i(j,1):i(j,2),k));
   end
end

% Send speed-per-step data to work space, plot it and save it as ascii files

av_speed
speed_std

figure
plot(av_speed(:,1))
hold
plot(av_speed(:,2),'r')
plot(av_speed(:,3),'y')
plot(av_speed(:,4),'g')
plot(av_speed(:,5),'k')
xlabel('Speed step')
ylabel('Average Speed')
legend('bin 0','bin 1','bin 2','bin 3','bin 4')
title('Mean speed per speed step - acceleration/deceleration data removed')

save mean-speeds av_speed -ascii;
save standard_deviation_speeds speed_std -ascii;


% Plot speed over time-step j from bin (k-1)

j=1; k=1;

figure
plot(t(i(j,1):i(j,2)),s(i(j,1):i(j,2),k))
title('Speed Step 1, Bin 0')


% Plot normal probability of speed in bin 0 at step 1
figure
normplot(s(i(j,1):i(j,2),k))
title('Normal Probability Plot of Bin O over Speed Step 1')
```

```
%______________________________________________________________________

% 9. Check that data now represents constant speed.

% Plot speed and normal probability of speed  in bin 3 at step 5

j=5; k=4;
figure
plot(t(i(j,1):i(j,2)),s(i(j,1):i(j,2),k))
figure
normplot(s(i(j,1):i(j,2),k))
title('Normal Probability Plot of Bin O over Speed Step 1')

%______________________________________________________________________

% 10. Determine whether data in Bins 0 to 4 have a common data set

p1=anova1(s)

title('Boxplot of data in bins 0 to 4')

% Boxplot shows data in bin 0 likely to have significantly different mean from that of
%       bins 1 to 4.

% Check by comparing bins 0 and 1 and bins 1 to 4

p2=anova1(s(:,1:2))
title('Boxplot of data in bins 0 & 1')
p3=anova1(s(:,2:5))
title('Boxplot of data in bins 1 to 4')

% Can, therefore, use all data points in bins 1 to 4 to determine mean speed at each
step.

%______________________________________________________________________

% 11. Establish vectors of speed & standard deviation of bins 1 to 4 for each speed
step

  s_best=[s1 s2 s3 s4];

   for j=1:e(1,1)
  av_speed_step(j)=mean([s_best(i(j,1):i(j,2),1)' s_best(i(j,1):i(j,2),2)'...
      s_best(i(j,1):i(j,2),3)' s_best(i(j,1):i(j,2),4)']);
  std_speed_step(j)=std([s_best(i(j,1):i(j,2),1)' s_best(i(j,1):i(j,2),2)'...
      s_best(i(j,1):i(j,2),3)' s_best(i(j,1):i(j,2),4)']);
    end

% Send speed-per-step data to work space, plot it and save it as ascii files
```

```
av_speed_step
std_speed_step
% 90% confidence level
max_speed_step=av_speed_step+2*(std_speed_step)
min_speed_step=av_speed_step-2*(std_speed_step)

figure
plot(av_speed_step)
xlabel('Speed step')
ylabel('Average Speed')
title('Mean speed per speed step - all bins - acceleration/deceleration data removed')

hold

plot(max_speed_step, 'r')
plot(min_speed_step, 'g')
legend('mean speed','+90% conf limit','-90% conf limit')

save mean-speeds av_speed -ascii;
save standard_deviation_speeds speed_std -ascii;
```

Chapter 19

CALCULATION OF DRAG FROM TRIALS DATA

19.1 Matlab script for the calculation of drag from trials data

```
% 'Script 'Deceleration1' analyses results of decelleration trial to establish
%      vehicle drag coefficient using data from trial m286.

%      Before using, Load data file using instruction '[h,u,d] = lad('m286.ls2');';

%_______________________________________________________________________

% 1. Establish and clean relevent data, viz:
%              time from start
%              doppler log bin 0, 1, 2, 3 easterly, northerly and down velocities
%              propeller rotation rate
%              motor curremt
%              hull attitude
%              temperature
%              salinity

%      Establish raw time in sec from start.

tr=d(:,1)-d(1,1);

%      Determine raw easterly, northerly and down velocities in m/s from bins 0,1,2,3
of
%              doppler log,where
%               bin 0 is speed over ground
%               bins 1,2,3,4,5,6 are speed through water with bin 1 closest to vehicle.

Ue0r=d(:,26)/1000; Un0r=d(:,27)/1000; Ud0r=d(:,28)/1000;

Ue1r=d(:,44)/1000; Un1r=d(:,45)/1000; Ud1r=d(:,46)/1000;
Ue2r=d(:,50)/1000; Un2r=d(:,51)/1000; Ud2r=d(:,52)/1000;
Ue3r=d(:,56)/1000; Un3r=d(:,57)/1000; Ud3r=d(:,58)/1000;
Ue4r=d(:,56)/1000; Un4r=d(:,57)/1000; Ud4r=d(:,58)/1000;
Ue5r=d(:,68)/1000; Un5r=d(:,69)/1000; Ud5r=d(:,70)/1000;
Ue6r=d(:,74)/1000; Un6r=d(:,75)/1000; Ud6r=d(:,76)/1000;

%      Establish propeller rotation rate (rad/sec) and motor current (amps)
%              so times for speed change can be determined.

Prr=d(:,310);
```

```
Imr=d(:,372);

% Establish hull atttitude

Har=d(:,320);

%       Establish water temperature (degC)and salinity (sal units).

Twr=d(:,41);
Swr=d(:,39);

% Establish clean each set of data of interest
%       and re allocate to vectors for ease of processing

x=[tr Ue0r Un0r Ud0r];
remove_data_dropouts;
purge_data;
remove_zeros;
ts0=x(:,1);
%       Speed from bin 0, s0:
s0=sqrt(x(:,2).^2+x(:,3).^2+x(:,4).^2);

x=[tr Ue1r Un1r Ud1r];
remove_data_dropouts;
purge_data;
remove_zeros;
ts1=x(:,1);
%       Speed from bin 1, s1:
s1=sqrt(x(:,2).^2+x(:,3).^2+x(:,4).^2);

x=[tr Ue2r Un2r Ud2r];
remove_data_dropouts;
purge_data;
remove_zeros;
ts2=x(:,1);
%       Speed from bin 2, s2:
s2=sqrt(x(:,2).^2+x(:,3).^2+x(:,4).^2);

x=[tr Ue3r Un3r Ud3r];
remove_data_dropouts;
purge_data;
remove_zeros;
ts3=x(:,1);
%       Speed from bin 3, s3:
s3=sqrt(x(:,2).^2+x(:,3).^2+x(:,4).^2);


x=[tr Ue4r Un4r Ud4r];
remove_data_dropouts;
purge_data;
```

```
remove_zeros;
ts4=x(:,1);
%       Speed from bin 4, s4:
s4=sqrt(x(:,2).^2+x(:,3).^2+x(:,4).^2);

x=[tr Ue5r Un5r Ud5r];
remove_data_dropouts;
purge_data;
remove_zeros;
ts5=x(:,1);
%       Speed from bin 5, s5:
s5=sqrt(x(:,2).^2+x(:,3).^2+x(:,4).^2);

x=[tr Ue6r Un6r Ud6r];
remove_data_dropouts;
purge_data;
remove_zeros;
ts6=x(:,1);
%       Speed from bin 6, s6:
s6=sqrt(x(:,2).^2+x(:,3).^2+x(:,4).^2);

%       Propulsion and water data
x=[tr Prr Imr Twr Swr];
tpw=x(:,1);
Pr=x(:,2);
Im=x(:,3);
Tw=x(:,4);
Sw=x(:,5);

%       Attitude data
x=[tr Har];
remove_data_dropouts;
%purge_data;
%remove_zeros;
tha=x(:,1);
Ha=x(:,2);
% convert rad to deg

Ha=Ha*(180/(2*pi));

figure(1)
plot(tha, Ha)
Title('Hull Attitude')
ylabel('Aoa - deg')

%       Plot processed speed for each bin
figure(2)
plot(ts0,s0,'+')

hold;
```

```
plot(ts1,s1,'r+')
plot(ts2,s2,'g+')
plot(ts3,s3,'y+')
plot(ts4,s4,'k+')
plot(ts5,s5,'c+')
plot(ts6,s6,'x')

Title('Speed by bin')
legend('Bin 0','Bin 1','Bin 2','Bin 3','Bin 4','Bin 5')
xlabel('Time from start - sec')
ylabel('Speed m/s')

pause

%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%       Note: from figure(2) can see that bin 5 suffers from sidelobe contamination
and
%                       that bin 6 is measuring speed over ground.
%                       So can use only bins 1-4 for measuring speed through the water

%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

figure(3)
plot(ts2,s2,'+')
hold
plot(ts3,s3+0.01,'r+')
plot(ts4,s4,'k+')
plot(ts5,s5,'c+')
plot(ts6,s6,'x')
legend('Bin 2','Bin 3+0.01','Bin 4','Bin 5')

%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

%       Note:   from figure(3) speed in bins 2,3 and 4 have no missing data points
%                       for any experiments and speeds 3 and 4 are coincident,
%                       so take bin 3 as providing representative speed data.

%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

% 2. Times for decelleration tial in secs from start

x=[tr Prr Imr];
remove_data_dropouts;
%purge_data;
%remove_zeros;
tpi=x(:,1);
Pr=x(:,2);
Im=x(:,3);
```

```
figure(4)
plot(tpi,Pr)
title('m286 - Propeller Rotation Rate')
xlabel('Time - sec')
ylabel('Propeller rpm')

figure(5)
plot(Pr,Im,'+')
title('m286 - Propeller Rotation Rate vs Motor Current')
xlabel('Propeller rpm')
ylabel('Motor Current - Amps')

ta=3216;
tb=3244;
tc=3398;
td=3428;
te=7638;
tf=7670;
tg=7822;
th=7854;

% Establish matrix were each row corresponds to the start and stop time
%               of each experiment.

t_exp=[ta tb
  tc td
  te tf
  tg th];

% Establish matrix i giving rows of indices of time matrix corresponding to speed
steps
% for data set for bin 3.

  for j=1:length(t_exp)
    i(j,1)=max(find(ts3<t_exp(j,1)+1));
                i(j,2)=max(find(ts3<t_exp(j,2)+1));
  end
% Plot speed profile for bin 3 for experiment 1

figure(6)
   % measure time from beginning of experiment 1
t31=ts3(i(1,1):i(1,2))-ts3(i(1,1));
speed31=s3(i(1,1):i(1,2));
plot(t31,speed31,'+:')
title('Decelleration Trial m286 Exp 1')
xlabel('Time - tsec from start of exp')


% Overlay plot of prop speed/175 and hull angle
% Establish matrix i giving rows of indices of time matrix corresponding to speed
steps
% for data set for bin 3.

hold

tpi1=tpi(i(1,1):i(1,2)+1)-tpi(i(1,1));
Pr1=Pr(i(1,1):i(1,2)+1);
plot(tpi1,Pr1/175,'rx:')


  for j=1:length(t_exp)
    i(j,1)=max(find(tha<t_exp(j,1)+1));
                i(j,2)=max(find(tha<t_exp(j,2)+1));
  end

tha1=tha(i(1,1):i(1,2))-tha(i(1,1));
Ha1=Ha(i(1,1):i(1,2));
plot(tha1,abs(Ha1),'go:')

legend('Bin3 Exp 1 m/s','Prop speed/175 rpm','Hull attitude -deg')

figure(7)
plot(ts3,s3,'+:')
hold
plot(tpi,Pr/175,'r+:')
title('Vehicle and Propeller Speed')
xlabel('Time - s')
legend('Vehicle speed m/s','Prop speed/175 rpm')

%       Repeat for all 4 experiments.
for j=1:length(t_exp)
    i(j,1)=max(find(ts3<t_exp(j,1)+1));
                i(j,2)=max(find(ts3<t_exp(j,2)+1));
  end
t32=ts3(i(2,1):i(2,2))-ts3(i(2,1));
speed32=s3(i(2,1):i(2,2));
t33=ts3(i(3,1):i(3,2))-ts3(i(3,1));
speed33=s3(i(3,1):i(3,2));
t34=ts3(i(4,1):i(4,2))-ts3(i(4,1));
speed34=s3(i(4,1):i(4,2));

figure(8)
plot(t31,speed31,'+:')
hold
plot(t32,speed32,'r+:')
%plot(t33,speed33,'g+:')
%plot(t34,speed34,'k+:')
title('Decelleration Trial m286')
xlabel('Time - tsec from start of exp')
```

```
ylabel('Speed - m/s')
%legend('Exp 1','Exp 2','Exp 3','Exp 4')
legend('Exp 1','Exp 2')

%  Export data as ascii files

data=[t31 speed31];
save decel1_286t1 data -ascii -tabs;
data=[t32 speed32];
save decel1_286t2 data -ascii -tabs;
data=[t33 speed33];
save decel1_286t3 data -ascii -tabs;
data=[t34 speed34];
save decel1_286t4 data -ascii -tabs;
```