THE VEHICLE COAST-DOWN TEST

by

Andrew Laurence Emtage

A Master's Thesis

Submitted in partial fulfilment of the requirements for the award of **Master of Philosophy** of the Loughborough University of Technology
10th October 1983

STATEMENT OF ORIGINALITY

The author hereby takes full responsibility for the work submitted in this thesis and claims originality for all work contained herein except were due acknowledgement is given or specific reference is made.

ABSTRACT

This report investigates the analysis of vehicle coast-down results. Two analytical models are investigated, and one is chosen as the basis for a derivative based correction technique to account for changes in the ambient conditions. A vehicle coefficient extraction algorithm, based on a parameter optimization technique, is developed. A FORTRAN program is developed to implement the correction and coefficient extraction algorithms, and also to carry out a statistical analysis on the extracted vehicle coefficients. The statistical analysis utilizes 'weights' (based on RMS curve fitting error) in order to account for random error in the input coast-down data. An investigation into the effect of measurement error on the extracted values of the coefficients was carried out, and it was recomended that the vehicle speed and wind velocity measurements should be improved. A large number of actual test were obtained and analyzed, but the results were inconclusive, except that it was clear that the measurement accuracy must be improved. Finally, recomendations for future work were made based on experience gained during the course of this work.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the co-operation and assistance he has received from many individuals and organizations, and in particular is indebted to his supervisor, Dr G G Lucas. Dr Lucas' advice and encouragement throughout this project were essential to its successful conclusion. The author also wishes to thank Professor F D Hales for his support of the work in his capacity as Head of Department, and also for his assistance in the development of the correction algorithm. Special thanks must also go to the following companys: Ford Motor Company, for providing the essential test data and giving invaluable advice; Dunlop Ltd, for information on tyre rolling resistance testing methods; Michelin Tyre Company, for tyre rolling resistance data; Numerical Algorithms Group, for permission to use their specialized software. The author is also gratefull to Professor C Storey for his help with the parameter optimization problem, Dr A N Pettitt for his help with the statistical analysis, Dr B Negus for his general help with numerical algorithms, and the Loughborough University Computer Centre staff who helped wherever possible. Finally, the author wishes to acknowledge the support which his wife gave in many ways, and last but by no means least, he wishes to acknowledge his Creator who has blessed him with the priviledge of studying a small part of His universe.

LIST OF CONTENTS

Page

Page

NOTATION

Symbol	Meaning	Units
a_0	drag force equation constant	N
a_1	" " " coefficient in v	N(s/m)
a_2	" " " " " v^2	$N(s/m)^2$
a_n	" " " " " v^n	$N(s/m)^n$
f_s	rolling resistance versus side force coefficient	-
g	gravitational acceleration	m/s^2
h	abberation	
i	track gradient	-
m	observed vehicle mass	kg
m_s	standard vehicle mass	kg
n	arbitary power of velocity	-
r_r	rolling radius of tyre/wheel combination	m
t	time	s
t_o	observed time interval	s
t_s	standard time interval	s
t_n	normalized time ($t_n=t_o/t_s$)	s
v	vehicle velocity	m/s
v_a	air velocity relative to vehicle	m/s
v_{hw}	head wind component of absolute wind velocity	m/s
v_r	head wind component of relative wind velocity	m/s
v_w	absolute wind velocity	m/s
v_{xw}	cross wind component of absolute wind velocity	m/s
x_o	observed value of ambient parameter	
x_s	standard " " " "	
w_{ij}	statistical weight for i^{th} item in j^{th} column	-
A	projected frontal area of vehicle	m^2
A_D	nominal static rolling resistance coefficient	-
A_{Do}	observed static rolling resistance coefficient	-
B_D	nominal rolling resistance coefficient in v	s/m
B_{Do}	observed rolling resistance coefficient in v	s/m
C_D	aerodynamic drag coefficient (shape only)	-
C_{Do}	" " " (shape + attitude)	-
C_L	" lift " (shape only)	-
C_{Lo}	" " " (shape + attitude)	-

Symbol	Meaning	Units
C_{So}	aerodynamic side force coefficient (shape+attitude)	-
D_D	nominal rolling resistance coefficient in v^2	$(s/m)^2$
D_{Do}	observed rolling resistance coefficient in v^2	$(s/m)^2$
F_A	aerodynamic contribution to total drag force	N
F_D	total drag force	N
F_R	rolling resistance contribution to total drag force	N
I_W	total rotating inertia of all wheels and tyres	$kg.m^2$
K_D	constant describing variation of C_{Do} with yaw angle	$(1/rad)^2$
K_L	" " " " C_{Lo} " " "	$(1/rad)^2$
K_S	" " " " C_{So} " " "	(1/rad)
K_T	" " " " A_{Do} " temp.	
K_Y	factor for correcting from C_{Do} to C_D	-
L	aerodynamic lift force	N
M_e	observed effective mass of vehicle	kg
M_e	standard effective mass of vehicle	kg
N_D	nominal rolling resistance coefficient in v^n	$(s/m)^n$
N_{Do}	observed rolling resistance coefficient in v^n	$(s/m)^n$
P	observed ambient atmospheric pressure	kPa
P_s	standard ambient atmospheric pressure	kPa
R	gas constant for air	kJ/kg.K
S	aerodynamic side force	N
T	observed ambient atmospheric temperature	K
T_s	standard ambient atmospheric temperature	K
W	observed vehicle weight	N
W_s	standard vehicle weight	N
α	constant used in coast-down function	
β	" " " " "	
γ	" " " " "	
Δ	time interval additive correction	s
ν	number of degrees of freedom	-
ρ	observed air density	kg/m^3
ρ_s	standard air density	kg/m^3
θ	wind direction rel. to head-on direction of track	rad
σ	standard deviation	
χ^2	statistical quantity	
ψ	aerodynamic yaw angle	rad

CHAPTER 1

INTRODUCTION

The coast-down (or deceleration) test is a well known procedure for determining the aerodynamic drag coefficient (C_D) and the static rolling resistance coefficient (A_D) for a wheeled road vehicle. This test is also used widely to provide performance rankings for modifications to a vehicle such as; the use of different tyres, changes in suspension geometry, and the addition of aerodynamic devices such as spoliers and air dams. In recent years, this test has also become the standard method for providing calibration data for road-load-simulation chassis dynamometers.

The basic principle of the coast-down test is very simple. The vehicle is accelerated on a smooth, level, straight test track, until it attains a speed above the upper limit prescribed for the test. The drive line of the vehicle is then isolated from the engine so as to allow the vehicle to freely coast (For manual transmissions the clutch is simply disengaged and the transmission placed in "neutral". For transmissions with fluid couplings or torque converters, a special clutch, fitted between the drive shaft and the transmission, is released.). When the speed reaches the upper limit of the test a timer is started, and a record of the speed versus time history is taken until the lower speed limit of the test is reached.

The shape of the speed/time characteristic (Fig 1.1) is related to the vehicle's drag force versus speed characteristic (Fig 1.2) and the effective mass of the vehicle. The relationship is a straight foward application of Newton's first law:

$$F_D = M_e \times \left\{- \frac{dv}{dt}\right\} \qquad (1.1)$$

The vehicle's drag-force/speed characteristic is always taken as some function of speed plus a constant. The usual form of the characteristic is as below[1]:

$$F_D = m \times g(A_D + B_D \times v) + 0.5 \times \rho \times A \times C_D \times v^2 \qquad (1.2)$$

Chapter two discusses two models for the drag-force/speed characteristic. The main difference between the two models is that the second one is more comprehensive and takes aerodynamic lift and side forces into account.

Apart from the vehicle characteristics, the total drag force is also affected by the ambient conditions prevailing. Listed below are the main ambient parameters and their effect on the total drag force:

a) track gradient - affects the constant term

b) wind speed - affects the constant and "v" terms

c) atmospheric pressure - affects the air density

d) atmospheric temperature - affects the air density and the rolling resistance

The wind direction is also important since it may affect the apparent value of C_D via a change in the aerodynamic yaw angle, but since it is not really independant of the wind speed it should not really be considered separately.

If a performance ranking only is required, then the effects of changes in the ambient conditions can be negated by adopting a 'control vehicle' test program. The results for the 'test vehicle' are then quoted relative to the results for the 'control vehicle'. If this method was adopted, it would not even be necessary to extract the vehicle coefficients from the data, since the total elapsed times would give adequate ranking.

If the actual values of the coefficients are required, or if a 'control vehicle' is not available, then the coast-down data must be corrected to some standard set of ambient conditions.

This thesis sets out to design an algorithm for coast-down analysis, in order to provide the following information for automotive engineers:

1) Accurate values of the rolling resistance coefficients for corre-

lation with results from tyre test rigs.

2) Accurate values of the aerodynamic drag coefficient for correlation with results from wind tunnels.

3) Accurate values of all the vehicle coefficients for the calibration of chassis dynamometers used for road-load simulation.

4) A strict specification of the coast-down test procedure in terms of data acquisition accuracy.

In view of the above, this thesis investigates the coast-down test with respect to the correction for ambient conditions, the extraction of the vehicle coefficients from the corrected data, and the specification of the test procedure.

Emtage[2] laid the foundation for the current work by investigating briefly, several correction algorithms to account for ambient conditions, and the concept of numerical optimization for extraction of the vehicle coefficients.

At the start of this investigation it was discovered that the supposedly 'raw' data supplied to Emtage[2], had in fact been 'doctored'. In fact it was discovered that the data had been averaged for runs in the same direction. This discovery places in question many of the conclusions made by Emtage[2], but the basic approach is still justified.

In the following sections, the literature concerning the coast-down test is reviewed with emphasis on correction for ambient conditions, extraction of vehicle coefficients, and specification of the coast-down test procedure.

1.1 Correction For Ambient Conditions

The most promising correction algorithm investigated by Emtage[2] consisted of a technique for correcting the length of each time interval of the recorded coast-down characteristic. The correction was in the form of a summation to the uncorrected time interval.

The error term (ie the difference in time between the uncorrected and corrected time intervals) was calculated by means of a partial differential error analysis, and the derivatives needed were calculated from initial estimates of the vehicle coefficients. Chapter three of this thesis investigates this algorithm further. This technique allows the correction of the data before the coefficient extraction process, and the performance of the technique may be assesed by comparing the spreads of the uncorrected and corrected data sets.

The authors of most of the literature[3-15] correct for ambient conditions by simply including the appropriate terms, from the drag-force/speed equation, into the coefficient extraction process. The main advantage with this approach, as apposed to the method described above, is in its simplicity and the ease with which it is implemented and modified. There are two disadvantages with this method however, and these are as follows: Firstly, the extra complexity of the model at the coefficient extraction stage may significantly increase the computational time, especially if an iterative scheme is implemented. Secondly, it is difficult to assess how well the the correction technique has worked. The only possible method of assessment would be to study the correlations between the extracted coefficients and the ambient conditions, but chapter four shows that it is possible for one coefficient to be traded off against another, and this type of error may mask any correlations.

For the above reasons, chapter three investigates further the partial differential correction technique as described by Emtage[2].

In most of the literature account is only taken of ambient temperature and pressure, as they affect air density, and of a slight wind parallel to the test track. Four references[2,6,15,16] also take account of the effect of ambient temperature on tyre rolling resistance, while five references[2,4,6,15,16] take account of moderate cross-winds. Five other references[3,4,6,12,15] take account of aerodynamic lift as it lessens the normal load at the tyre/ground interface, but only Yasin[15] takes account of aerodynamic side force as it affects the rolling resistance. Finally, nine references[3,4,6-12] also take account of track gradient.

The comprehensive model described in chapter two takes account of all the ambient parameters mentioned above, but because of the complexity involved, chapter three investigates a correction algorithm which ignores the effects of lift and side force.

1.2 Extraction Of Vehicle Coefficients

In this section the various methods, described by the literature, for extracting the vehicle coefficients are discussed briefly.

Lucas[1] suggests fitting a least squares, power series polynomial of about 6th or 7th order, to the coast-down data. The polynomial is then differentiated (a trivial analytical process) and multiplied by the effective mass of the vehicle in order to obtain the drag-force/speed characteristic. A second order, least squares, power series polynomial is then fitted to this characteristic in order to obtain the vehicle coefficients. This method at first appears to be very convienent, and it is certainly very efficient in terms of computational time, however, it poses some serious difficulties. Firstly, it is very difficult to obtain a good curve fit for the coast-down data, even with high order polynomials, and it will be shown in chapter four that the accuracy of the extracted coefficients is very sensitive to the curve fitting error. Secondly, any error in the data to begin with, and also any error introduced by the curve fitting process, will be magnified by the differentiation process.

Emtage[2] proposes a parameter optimization procedure to fit the analytical form of the coast-down speed/time characteristic directly to the test data (or corrected test data). This method is an improvement on the previous one since, provided that there is confidence in the model being used, the only source of error is the data. If the model is correct, then the curve fitting error may be used as an indicator of the random error in the data, and this may be used in a sensitivity analysis of the extracted vehicle coefficients. As described by Emtage[2] this procedure still requires development in terms of the model used, and in the scaling of the optimization problem.

Dayman[3,4] describes what is perhaps the most interesting algorithm. It allows the development of a model of any complexity for the drag-force/speed characteristic. This is then integrated numerically, within a parameter optmization process, to give a speed/time coast-down curve. This coast-down curve is then compared with the test data and a value of curve fitting error calculated. The optimization process then optimizes the various parameters of the model until the best fit is achieved. The obvious advantage of this scheme is that the the drag-force/speed characteristic, assumed by the model, is not limited to a second order polynomial, in fact any function may be used. The main disadvantage of the scheme is that it is extremely expensive in terms of computational time.

Five papers[7,8,13,14,15] describe the use of parameter optmization to fit a normalized coast-down curve to the test data. The idea of the normalized coast-down curve is so that the performance of similar cars can be correlated onto a single coast-down curve.

Roussillon et al[9-12] describe a method of extracting the aerodynamic drag coefficient from coast-down tests performed in a light head wind. With the presence of the wind, and by writing the drag equation in terms of the square of the relative head wind velocity, the true aerodynamic drag coefficient may be separated from the rolling resistance. With good instrumentation this technique has proved very successfull, even on tracks with a slight, known gradient.

One reference[16] suggests the numerical differentiation of the coast-down data, and then a second order, least squares, power series polynomial fit to the resulting drag-force/speed data in order to obtain the vehicle coefficients. Once again, the differential process greatly magnifies any error in the observed data.

For the purposes of this thesis it was decided to continue the development of the algorithm described by Emtage[2] for three reasons. Firstly, this method avoided the differentiation process. Secondly, no special test conditions were required (such as a necessary, light head wind). Thirdly, the process is not too costly in terms of computational time. In view of these points, chapter four investigates further this method of vehicle coefficient extraction.

1.3 Specification Of The Coast-Down Test Procedure

In the literature surveyed, there is very little information given concerning the relationship between the procedure for the coast-down test, and the expected accuracy of the extracted vehicle coefficients. Most of the literature simply states that that the vehicle must be warmed up to operating temperatures before the testing begins, and only two papers[16,17] give a definitive criterion for accepting test data. This criterion is simply based on the statistical accuracy of the average total elapsed times for pairs of tests in opposite directions. The most stringent criteria insists that the statistical accuracy be better than two percent error. This may appear to be a reasonable approach, but no relationship is given between this criteria and the necessary accuracy of the instrumentation. Even more importantly, it is possible for data sets with identical total elapsed times to yield quite different values for the vehicle coefficients. To be quite fair, these two references[16,17] do give figures for the required accuracy of the speed and time measurements, but no basis is given for them.

Korst and Funfsinn[8] did carry out an error analysis based on numerically simulated test runs with controlled data randomization levels. From the results of this analysis the authors concluded that "...at least three digit accuracy is required to acheive clear definition of the individual drag contributions.". This type of approach is used in chapter five of this thesis to study the problem even further.

CHAPTER 2

ANALYTICAL MODEL

In this chapter it is proposed to discuss the analytical model for the coast-down test, but in order to do this one must first look at the model for the drag-force/speed characteristic of the vehicle. In its most general form this is given as:

$$F_D = F_R(v) + F_A(v) \tag{2.1}$$

Here the aerodynamic contribution (F_A) to the total drag force is taken as some function of speed, and all other contributions are lumped together as the rolling resistance (F_R), as some function of speed.

In the following two sections the models for these two **separate** contributions are discussed, while the third section deals with the complete drag-force/speed equation and its transformation into the coast-down characteristic. In each of these sections there are sub-sections dealing with the basic model (used for the majority of this investigation), and the comprehensive model (this is developed with a view to future work).

2.1 The Aerodynamic Contribution

At first thought, one might consider the aerodynamics of ground vehicles to be very straight foward compared with today's aerospace vehicles. In one sense this is true because, at the relatively low speeds of ground vehicles, the assumption of incompressible flow is valid. However, in the words of Waters[18], "The simplification afforded by the incompressible flow equation is however more than counterbalanced by the problems arising from (i) ground proximity, (ii) the 'bluff' shapes used, that lead to strong viscous flow seperation effects, and (iii) the cross-flow, velocity gradient and unsteady effects associated with the natural wind."

For the purposes of ground vehicles, the aerodynamic drag is made up of skin friction drag and dynamic pressure drag. For long vehicles, such as trains, the skin friction drag may predominate, but for cars the pressure drag is by far the most important.

It is conventional (MIRA convention) that all aerodynamic forces are defined as the products of the axial component of the dynamic pressure of the air stream (relative to the vehicle), a reference area, and a coefficient. For cars, the reference area is usually taken as the projected frontal area. The coefficient for the longitudinal aerodynamic force is the now familiar C_{Do} (C_x in some texts because the force acts along the 'X' axis of the vehicle according to the MIRA convention).

The aerodynamic contribution to the total drag may then be written as:

$$F_A = (0.5 \times \rho \times v_r^2) \times A \times C_{Do} \qquad (2.2)$$

The other aerodynamic forces are the lift force (L) and the side force (S) which have the same form as equation 2.2, and coefficients C_{Lo} and C_{So} respectively.

So far the picture is fairly simple, however, these three coefficients (C_{Do}, C_{Lo} and C_{So}) are all functions of Reynolds Number, Froude Number, Mach Number, vehicle shape, vehicle attitude and the surface roughness. For the purposes of these relatively slow ground vehicles, the Froude Number and Mach Number effects may be completely ignored.

The Reynolds Number effect for cars is measurable, but is usually ignored for the speed range of the coast-down test, and is usually only considered for very high performance cars such as Formula 1 racing cars.

As has already been mentioned, the pressure drag predominates in the case of cars, and so surface roughness may in general be ignored. (In some cases the effect of surface roughness, at critical places on the car's surface, may have significant consequences for the overall flow patern, and hence the pressure drag. An example of such a critical area might be the point of flow seperation at the back of the car.)

It may be obvious that these coefficients (especially C_{Do}) are functions of shape, but what may not be so obvious is that ground clearance, rotation of wheels, and the velocity/height profile of the ambient wind, all are effectively shape parameters. It is mainly due to these shape parameters, that the wind tunnel derived and test track derived values of the coefficients differ. It is quite difficult to account for these differences when carrying out wind tunnel tests, but it may be argued that the resulting errors are small (< 5%) except for special ground effect cars.

Vehicle attitude is also very important as it can dramatically alter the flow pattern around the vehicle. The changes in attitude are described by the yaw, pitch and roll angles as defined by the MIRA convention. Roll may be ignored and pitch has little effect over the speed range of the coast-down test. The effect of yaw angle, however, is of vital importance if cross-winds are to be considered.

For our purposes, we are only interested in the coefficients which reflect only the shape of the vehicle. For this reason we must introduce another term into the force equation to account for the change in the observed coefficient (coefficient which reflects the shape and attitude of the car) due to the yaw angle.

In the following sub-sections the method of introducing the yaw angle term is discussed with respect to a basic model, which only takes account of the aerodynamic drag force, and with respect to a comprehensive model which also takes account of the lift and side forces.

2.1.1 The Basic Model

This is the same model as that used by Emtage[2], and simply consists of equation 2.2 with the addition of a factor (K_Y) which is a function of yaw angle. For the purposes of computation this function must be stored.

From Fig 2.1, the relative head-wind speed (v_r) may be replaced by the sum of the vehicle speed (v) and the head-wind component of the absolute wind velocity (v_{hw}). Hence equation

2.2 becomes:

$$F_A = 0.5\times\rho\times A\times C_D\times K_Y(\psi)\times(v+v_{hw})^2 \qquad (2.3)$$

Figure 2.2 describes the function K_Y for a typical 'previous generation' passenger car. This figure describes the function for yaw angles far beyond those which are to be expected during coast-down tests. In reality, yaw angles above 10 or 15 degrees are not to be expected.

2.1.2 The Comprehensive Model

The first difference between the two models is that this one includes the lift and side-force terms. The second difference is in the method by which the aerodynamic yaw angle is taken into account.

Yasin[15] claims that, for most cars, plotting the actual value of C_{Do} versus the square of the yaw angle (in radians), over the range 0-15 degrees, yields a linear relationship. The intercept is the nominal or shape dependent only coefficient, C_D, and the slope is a constant (K_D). This obviously gives us a method of taking account of the yaw angle effect by an analytical method, rather than an **empirical** method. The yaw effect is then built into the analysis as follows:

$$F_A = 0.5\times\rho\times A\times(C_D+K_D\times\psi^2)\times(v+v_{hw})^2 \qquad (2.4)$$

From Fig 2.1 we see that:

$$\psi = \mathrm{Tan}^{-1}(v_{xw}/(v+v_{hw})) \qquad (2.5)$$

If we approximate equation 2.5 by assuming that the yaw angles under consideration are small, then:

$$\psi = v_{xw}/(v+v_{hw}) \qquad (2.6)$$

Substituting equation 2.6 into equation 2.4 yields:

$$F_A = 0.5\times\rho\times A\times C_D\times(v+v_{hw})^2 + 0.5\times\rho\times A\times K_D\times v_{xw}^2 \tag{2.7}$$

With reference to Fig 2.1, and writing equation 2.7 in terms of the absolute wind speed and direction, and expanding the air density in terms of temperature and pressure, we have:

$$\begin{aligned} F_A = {} & \frac{P\times A}{2\times R\times T}\times C_D\times v^2 \\ & + \frac{P\times A}{2\times R\times T}\times C_D\times 2\times v_w\times \mathrm{Cos}(\theta)\times v \\ & + \frac{P\times A}{2\times R\times T}\times C_D\times v_w^2\times \mathrm{Cos}^2(\theta) \\ & + \frac{P\times A}{2\times R\times T}\times K_D\times v_w^2\times \mathrm{Sin}^2(\theta) \end{aligned} \tag{2.8}$$

Since the yaw angle characteristic of C_{Lo} is similar to that of C_{Do}, the lift force may be derived by a similar process to give:

$$\begin{aligned} L = {} & \frac{P\times A}{2\times R\times T}\times C_L\times v^2 \\ & + \frac{P\times A}{2\times R\times T}\times C_L\times 2\times v_w\times \mathrm{Cos}(\theta)\times v \\ & + \frac{P\times A}{2\times R\times T}\times C_L\times v_w^2\times \mathrm{Cos}^2(\theta) \\ & + \frac{P\times A}{2\times R\times T}\times K_L\times v_w^2\times \mathrm{Sin}^2(\theta) \end{aligned} \tag{2.9}$$

Over the yaw angle range being considered, the side force coefficient characteristic is linear with yaw angle (in radians) and passes through zero. The side force equation is therefore:

$$S = 0.5\times\rho\times A\times K_S\times\psi\times(v+v_{hw})^2 \tag{2.10}$$

Substituting equation 2.6 into equation 2.10 gives:

$$S = 0.5\times\rho\times A\times K_S\times v_{xw}\times(v+v_{hw}) \tag{2.11}$$

Re-writing equation 2.11 in the form of equations 2.8 and 2.9 yields:

$$S = \frac{P \times A}{2 \times R \times T} \times K_S \times v_W \times \mathrm{Sin}(\theta) \times v$$

$$+ \frac{P \times A}{2 \times R \times T} \times K_S \times v_W^2 \times \mathrm{Sin}(\theta) \times \mathrm{Cos}(\theta) \tag{2.12}$$

In order to confirm that the constants K_D, K_L and K_S can be found, Fig 2.3 gives a graph of C_{Do} and C_{Lo} versus the square of yaw angle for a small saloon car. As can be seen, the graphs indicate good linearity. Fig 2.4 gives the equivalent information for C_{So}. The fact that the curve does not pass through the origin is not relevant since only the slope is important.

2.2 The Rolling Resistance Contribution

The term 'rolling resistance' is usually only associated with the energy losses due to the flexing of the tyres, however for the purposes of this work, we shall define the term to include all energy losses other than the aerodynamic ones mentioned in section 2.1. These other losses include bearing friction, final drive losses, energy dissipated in the suspension, and possibly brake bind. The extra losses are usually all excluded during rig tests (except for some bearing losses), and it is common practice to remove or slacken the brakes before conducting coast-down tests. One reference[6] takes account of these extra losses by evaluating them in independent experiments.

From this point on we shall assume, for the purposes of analysis, that the rolling resistance contribution is purely from the tyres, while at the same time remembering that the extra losses will be included in the results of the coast-down experiments.

At this point it is appropriate to define what is strictly meant by rolling resistance. The most common definition is in terms of energy dissipated per unit distance rolled by the tyre[19]. The units are therefore Nm per m, or N.

2.2.1 Rolling Resistance As A Function Of Normal Load

It is well known that tyre rolling resistance is proportional to its normal loading. For this reason the rolling resistance coefficient is defined as the ratio of the rolling resistance to the normal load. This is a useful measure because, as well as being non-dimensional, it provides a quick method of comparing tyre sizes and inflation pressures for a particular vehicle application. All other things remaining constant, the rolling resistance may then be written as:

$$F_R = m \times g \times (\text{rolling resistance coefficient}) \qquad (2.13)$$

2.2.2 Rolling Resistance As A Function Of Speed

It is well known that the rolling resistance varies with speed from some constant value at rest. What is not clear, however, is the mathematical representation of the speed dependent characteristic.

Lucas[1] and others have suggested that the rolling resistance function be written as follows:

$$F_R = m \times g \times (A_{Do} + B_{Do} \times v) \qquad (2.14)$$

For many years the automotive industry used the above formulation for rolling resistance, and in many cases (for vehicle performance calculations) the 'v' term was neglected.

There is some analytical evidence[20] to indicate that the rolling resistance should be proportional to the square of velocity, so equation 2.14 becomes:

$$F_R = m \times g \times (A_{Do} + B_{Do} \times v + D_{Do} \times v^2) \qquad (2.15)$$

This function is already quite complicated but may still be inadequate, in fact Dayman[3,4] suggests that the following function be used:

$$F_R = m \times g \times (A_{Do} + N_{Do} \times v^n) \quad (2.16)$$

Dayman[3,4] found that a value of n=4 was about optimum for his work. Fig 2.5 shows values of correlation coefficient versus 'n' for a tyre which was used for the coast-down tests described elsewhere in this thesis (tyre type AA). From this graph it can be seen that a value of about n=3.5 is appropriate (Please note that this value of 'n' only applies to the normal speed range of the coast-down test.). Fig 2.6 gives the original rolling resistance versus speed characteristic of this tyre.

It should be noted that the rolling resistance versus speed characteristic described above assumes equilibrium at each point. Of course, during the coast-down test, equilibrium is never attained, and this is obviously a source of error.

2.2.3 Rolling Resistance As A Function Of Temperature

Two references[16,19] suggest that rolling resistance decreases as ambient temperature increases. The most up-to-date figure is 1.3% per deg C. The effect of temperature is usually written as a correction factor from an observed value of rolling resistance to a standard value at a standard temperature:

$$\text{correction factor} = 1/(1-K_T(T-T_S)) \quad (2.17)$$

where $K_T \times 100$ is in % per deg C.

2.2.4 Other Factors Affecting Rolling Resistance

It has been shown[19] that rolling resistance is also affected by the camber and slip angles of the tyre, and also the torque applied to the wheel. For the purposes of the coast-down test the torque effect is non-existant, and the effect of suspension geometry may be important as a development variable. For these reasons it has been decided not to delve into the effects of these parameters.

Yasin[15] suggests that the aerodynamic side force affects the rolling resistance in the following way:

$$\text{side force rolling resistance factor} = (1 + f_S \times S) \qquad (2.18)$$

2.2.5 The Basic Model

The basic model assumes constant and 'v' terms, accounts for vehicle weight, and partially accounts for ambient temperature:

$$F_R = m \times g \times (A_D \times (1 - K_T \times (T - T_S)) + B_D \times v + i) \qquad (2.19)$$

The temperature effect on B_D was neglected because it would have made the complete drag-force/speed equation too complicated, and because B_D is small.

The gradient term is simply included by assuming that the gradient induced force is mg times the Sine of the gradient angle, which for small gradients is taken as being the gradient term (i) itself.

2.2.6 The Comprehensive Model

This model combines equations 2.13, 2.16, 2.17 and 2.18 to give:

$$F_R = (m \times g) \times \{(1 - K_T \times (T - T_S)) \times (1 + f_S \times S) \times (A_D + N_D \times v^n) + i\} \qquad (2.20)$$

2.3 Combining Aerodynamic And Rolling Resistance Contributions

2.3.1 The Basic Model

The basic model combines equations 2.3 and 2.19 to give:

$$F_D = mg(A_D(1-K_T(T-T_S))+B_Dv+i) + 0.5\rho AC_DK_Y(v+v_{hw})^2 \qquad (2.21)$$

2.3.2 The Comprehensive Model

The comprehensive model combines equations 2.8, 2.9, 2.12 and 2.20 to give:

$$F_D = (mg-L)\{(1+f_sS)(1-K_T(T-T_S))(A_D+N_Dv^n) + i\} + F_A \qquad (2.22)$$

When expanded out this gives an equation of the form:

$$F_D = a_0 + a_1v + a_2v^2 + a_3v^3 + a_nv^n + a_{n+1}v^{n+1} + a_{n+2}v^{n+2}$$

$$+ a_{n+3}v^{n+3} \qquad (2.23)$$

2.3.3 The Coast-Down Speed/Time Model

Equation 1.1 may be integrated to give:

$$\int_{v2}^{v1} \frac{M_e}{F_D(v)}dv = \int_{t1}^{t2} dt$$

$$= t_2 - t_1$$

and if $t_1=0$ then:

$$t = M_e\int_{v2}^{v1} \frac{dv}{F_D(v)} \qquad (2.24)$$

Bearing in mind the complexity of equation 2.23, it should be clear that it would be impossible to carry out analytically the integration in equation 2.24. The only possibility is to carry out the integration numerically as described by

Dayman[3].

For the basic model, the form of the drag-force/speed characteristic is simply a quadratic function, and there is a standard analytical solution to the integral in equation 2.24. The solution is described in some detail by Emtage[2], but essentially it is as follows:

if,

$$F_D = a_0 + a_1 \times v + a_2 \times v^2 \tag{2.25}$$

then from equation 2.24:

$$t = \frac{1}{\beta}\left\{\frac{v_1 - v_2}{\alpha^2 + (v_1+\gamma)(v_2+\gamma)}\right\} \tag{2.26}$$

where,

$$\alpha^2 = \frac{a_0}{a_2} - \frac{a_1^2}{4a_2^2} \tag{2.27}$$

and,

$$\beta = \frac{a_2}{M_e} \tag{2.28}$$

and,

$$\gamma = \frac{a_1}{2a_2} \tag{2.29}$$

Now the effective mass of the vehicle is the actual mass, plus the ratio of the total rotating inertia to the square of the tyre rolling radius. ie:

$$M_e = m + \frac{I_w}{r_r^2} \tag{2.30}$$

Substituting equations 2.27, 2.28, 2.29 and 2.30 into equation 2.26 yields:

$$t = \frac{K_1}{K_2} \tag{2.31}$$

where,

$$K_1 = (m + I_w/r_r^2)(v_1 - v_2) \tag{2.32}$$

and,

$$K_2 = mgA_D - mgA_DK_T(T-T_S) + mgi + \frac{PA}{2RT}C_DK_Yv_1v_2$$

$$+ \frac{PA}{2RT}C_DK_Yv_{hw}^2 + \frac{PA}{4RT}C_DK_Y(v_1+v_2)$$

$$+ \frac{PA}{2RT}C_DK_Yv_{hw}(v_1+v_2) \tag{2.33}$$

This is the form of the coast-down function which is the most useful for developing the correction algorithm, and also for comparing coast-down curves which start at the same initial velocity and have identical speed intervals. This is because velocity is the independent variable.

For the curve fitting process used during the vehicle coefficient extraction process, it was found useful to have elapsed time as the independent variable. In view of this, equation 2.26 may be re-arranged to give:

$$v_2 = \alpha\left\{\frac{\text{const} - \text{Tan}(\alpha\beta t)}{1 + \text{const}\times\text{Tan}(\alpha\beta t)}\right\} - \gamma \tag{2.34}$$

where,

$$\text{const} = \frac{v_1 + \gamma}{\alpha} \tag{2.35}$$

It should be noted that there are really three possible solutions to the integral of a quadratic equation, but Emtage[2] has shown that the one described above is always valid for ordinary cars under the conditions of the coast-down test. Appendix A gives a full derivation of equations 2.26 and 2.34, along with any assumptions made.

CHAPTER 3

CORRECTION FOR AMBIENT CONDITIONS

This chapter deals with the design, implementation and testing of a correction algorithm to account for the effects of ambient conditions on the coast-down test. Ideally, this algorithm would be capable of correcting data for one particular vehicle, measured under differing ambient conditions, and arrive at one standard coast-down curve.

3.1 Algorithm Design

3.1.1 Ambient Conditions Under Consideration

Atmospheric temperature and pressure are perhaps the most obvious parameters since they affect the air density, and hence the dynamic pressure of the air stream. As was mentioned in section 2.2.3, the ambient temperature also affects the rolling resistance of the tyres. The standards for these two ambient parameters is usually taken as 20 degrees centigrade and 101.3 kPa for the temperature and pressure respectively.

The head-wind component of the ambient wind velocity, and the yaw angle correction factor (K_Y), must also be accounted for. It may seem odd that both of these parameters have to be corrected for, since they are not really independent. The problem is that the basic model (see section 2.1.1) does not give an analytical relationship between these two parameters, and for this reason they must be treated as being independent of each other. The comprehensive model (see section 2.1.2), on the other hand, gives the yaw effect as an analytical function of the absolute wind velocity, and because of this, it is only necessary to correct for the ambient wind speed. The basic model could be extended to incorporate this feature, but this must now be left for some future development of coast-down analysis. The standard wind speed is of course zero, as is the standard yaw angle.

The track gradient is often overlooked in coast-down analysis, but it could be quite important. The fact is that a gradient of only 1 in 10,000 is equivalent to an error of 1% in the extracted value of the static rolling resistance coefficient (A_D). Of course some test facilities even claim that their test tracks follow the curvature of the earth. A problem which might occur on some test tracks is that, while the overall gradient may be zero, the local value of the gradient along the test track may be unacceptable. If this is the case, one solution may be to accurately survey the track, and set up markers at intervals between which the gradient is constant and known. In this way the local gradient may be incorporated into the coast-down analysis. This type of analysis was carried out by Dayman[4]. Naturally, the standard gradient is taken as zero.

The vehicle mass was chosen to be the final 'ambient' parameter because of its effect on the effective mass and the rolling resistance. This parameter could be ignored if the vehicle mass were strictly controlled during testing.

3.1.2 Correction Method

The correction method used is a differential approach, which corrects the time interval between two consecutive points on the coast-down curve. This method uses the derivatives of the time versus ambient parameter function (calculated by assuming standard ambient conditions), to predict the difference between the values of the function (ie time) at standard, and actual ambient conditions. The time difference thus calculated is then summed to the recorded time interval in order to effect the correction.

For one parameter the correction equation would be:

$$f(x_s) = f(x_s+h) - h \times f'(x_s) - \frac{h^2}{2} \times f''(x_s) - \text{etc} \qquad (3.1)$$

where,

$f(x_s) = t_s$

and,

$f(x_s+h) = t_o$

and,

x_s = datum (or standard) value of parameter

and,

$x_s + h = x_o$ = observed value of parameter

and,

h = the abberation in the ambient parameter

In this particular case time is a function of six variables (T, P, v_{hw}, K_Y, i, m), and so the corrected time is given by:

$$t_s = t_o - \Delta \qquad (3.2)$$

where,

$$\begin{aligned}\Delta = &\left\{\frac{\partial t}{\partial T}\times\delta T + \frac{\partial t}{\partial P}\times\delta P + \frac{\partial t}{\partial v_{hw}}\times\delta v_{hw} + \frac{\partial t}{\partial K_Y}\times\delta K_Y + \frac{\partial t}{\partial i}\times\delta i + \frac{\partial t}{\partial m}\times\delta m\right\}\\
&+ \frac{1}{2}\left\{\frac{\partial^2 t}{\partial T^2}\times\delta T^2 + \frac{\partial^2 t}{\partial P^2}\times\delta P^2 + \frac{\partial^2 t}{\partial v_{hw}^2}\times\delta v_{hw}^2 + \frac{\partial^2 t}{\partial K_Y^2}\times\delta K_Y^2 + \frac{\partial^2 t}{\partial i^2}\times\delta i^2 + \frac{\partial^2 t}{\partial m^2}\times\delta m^2\right\}\\
&+ \left\{\frac{\partial^2 t}{\partial T.\partial P}\times\delta T.\delta P + \frac{\partial^2 t}{\partial T.\partial v_{hw}}\times\delta T.\delta v_{hw} + \frac{\partial^2 t}{\partial T.\partial K_Y}\times\delta T.\delta K_Y\right.\\
&+ \frac{\partial^2 t}{\partial T.\partial i}\times\delta T.\delta i + \frac{\partial^2 t}{\partial T.\partial m}\times\delta T.\delta m + \frac{\partial^2 t}{\partial P.\partial v_{hw}}\times\delta P.\delta v_{hw}\\
&+ \frac{\partial^2 t}{\partial P.\partial K_Y}\times\delta P.\delta K_Y + \frac{\partial^2 t}{\partial P.\partial i}\times\delta P.\delta i + \frac{\partial^2 t}{\partial P.\partial m}\times\delta P.\delta m\\
&+ \frac{\partial^2 t}{\partial v_{hw}.\partial K_Y}\times\delta v_{hw}.\delta K_Y + \frac{\partial^2 t}{\partial v_{hw}.\partial i}\times\delta v_{hw}.\delta i + \frac{\partial^2 t}{\partial v_{hw}.\partial m}\times\delta v_{hw}.\delta m\\
&\left.+ \frac{\partial^2 t}{\partial K_Y.\partial i}\times\delta K_Y.\delta i + \frac{\partial^2 t}{\partial K_Y.\partial_m}\times\delta_{K_Y}.\delta_m + \frac{\partial^2 t}{\partial_i.\partial_m}\times\delta_i.\delta_m\right\}\end{aligned} \qquad (3.3)$$

The first bracketed group represents the first order derivatives, the second group represents the second order derivatives, and the third group represents the second order cross-derivatives. It is, of course, an approximation to ignore the higher order derivatives and cross-derivatives, but it will be demonstrated that, in most cases, the first order terms are adequate.

The six first order derivatives, six second order derivatives, and the fifteen second order cross-derivatives, based on the basic analytical model described by equation 2.31, are given in appendix B.

3.2 Algorithm Implementation

A FORTRAN computer program was developed to analyse the results of the coast-down test. This program is called 'CSTDWN', and is described, in some detail, in appendix C.

The implementation of the correction algorithm in program CSTDWN is very straight foward. The speed/time data for one coast-down test is read in and stored. The various derivatives are then calculated for each speed interval, and the corresponding recorded time interval is corrected according to the algorithm described by equations 3.2 and 3.3. The corrected coast-down data is then stored in readiness for the vehicle coefficient extraction process.

In the initial stages of the development of the correction algorithm, the program was written so as to allow the printing of 'normalized time' and a chosen ambient parameter. The normalized time (t_n) is given by:

$$t_n = \frac{t_o}{t_s} \qquad (3.4)$$

The program printed out the normalized time for the last speed interval. The program gave the choice of using either the first order derivatives only, or first and second order derivatives (but not the second order cross-derivatives). In conjunction with this, the complete analytical coast-down function (see chapter two) was used to calculate 'simulated' values of normalized time for different values of the chosen ambient parameter.

3.3 Algorithm Testing

3.3.1 Derivative Correction Versus Simulated Correction

Figs 3.1 - 3.6 give values of normalized time versus the six ambient parameters. On each graph there is a curve derived from the complete analytical coast-down function, and is basically the result of the ratio of observed elapsed time to standard elapsed time, based on simulations for observed conditions and standard conditions respectively. The other two curves are for corrections based on first order derivatives only, and first and second order derivatives (but not second order cross-derivatives).

In most cases, except for pressure and mass, the error associated with the use of only the first order derivatives is noticeable (but possibly acceptable). This is especially true in the case of wind speed. The use of both first and second order derivatives, however, results in negligible error in all cases.

At this point the question will be raised, as to the reason for not using a correction method based on two simulations (based on the complete analytical coast-down function), one for observed ambient conditions, and the other for standard conditions. This would of course eliminate any errors due to the second order approximation used in the derivative algorithm. The answer to this question is debatable, but it is probabally more efficient to use the derivative method if a first order approximation is adequate. On the other hand, it would be far easier to accomodate any changes in the analytical coast-down function if the simulation approach were used. In view of the last point, any future work might well benefit by adopting a correction algorithm based on the simulation approach.

3.3.2 Performance Of Algorithm By Correlations

One method of testing the performance of the correction algorithm is to plot extracted values of the vehicle coefficients against the ambient parameters. Fig 3.7, for example, gives the extracted values of A_D, for a large number of coast-down experiments, versus ambient temperature. If this type of graph were to indicate a correlation between the coefficient and the ambient parameter, then the algorithm would have failed. In this case there is no significant correlation. (The line drawn is a least squares straight line.)

Fig 3.8 attempts to find a correlation between test direction and A_D. There appears to be a slight increase in A_D for the positive direction, and this might indicate that there was a slight upward gradient in this direction.

The main difficulty in using these graphs is that the vertical scatter is very large compared with the correlation which is being measured, and this therefore puts a question mark over the validity of the exercise. The problem is highlighted again by Figs 3.9 - 3.14. These figures give the probability density bar charts for the extracted vehicle coefficients, which were obtained from both un-corrected and corrected data. As can be seen, the standard deviations do not decrease when the data is corrected.

The explanation for the above problem is that the vertical scatter arises from two possible sources. Firstly, scatter is introduced because of changes in the ambient conditions for which proper correction has not been made (either because of a bad correction algorithm or because of incorrect ambient data). Secondly, scatter is introduced by the coefficient extraction process itself. The second source of error is discussed in detail in chapter four, but briefly, given some random error in the input coast-down data, it is possible for one coefficient to be traded off against another one and yet give almost the same value of curve fitting error. The more the error in the data, the less confident one can be in the resulting coefficients. The large scatter in the values of the coefficients is thus mainly due to random error in the data, and must be reduced before correlation techniques can be usefuly used to test correction algorithms.

3.3.3 Performance Of Algorithm By Comparison Of Data

Due to the problems mentioned in the previous section, we need a better method of testing the correction algorithm. The method described here is based on a comparison of all corrected data sets.

For this method, it is necessary to have the data sets in a form so that speed is the independent variable, and that all the data sets have the same initial speed and speed intervals.

The data sets are then compared by calculating the standard deviations of each time interval, along the curves, as a percentage of the mean times. These percentage standard deviations are then plotted against speed as in Fig 3.15. This provides a clear indication of the performance of the correction algorithm when the results for corrected and uncorrected data are compared.

Another useful measure is the average percentage standard deviation, and these values are also included on Fig 3.15.

3.4 Correction Of Recorded Wind Speed

The coast-down data obtained from one particular test facility was affected by bad wind velocity measurement. The wind velocity was measured at the top of an eight metre high tower, which was at some distance from the test track in use. The real wind velocities associated with the test vehicle were therefore likely to be quite different from those recorded. The method for comparing the corrected coast-down curves, as described in section 3.3.3, provides a useful method for correcting the recorded wind speeds.

The method is based on the following assumptions. Firstly, it is assumed that the data consists of pairs of runs in opposite directions. Secondly, it is assumed that the track gradient is zero. Thirdly, it is assumed that by varying the correction factor for wind speed (WFACT as it is used in program CSTDWN), it should be possible to minimize the average percentage standard deviation for the corrected (ie corrected in every other respect) data sets.

This correction method would be used for groups of data sets for which the recorded ambient conditions were constant (ie the tests were carried out together). Fig 3.16 gives a typical result for this correction technique. The wind factor in this case would be about 3.4. The recorded wind speeds were then corrected as required.

CHAPTER 4

VEHICLE COEFFICIENT EXTRACTION

This chapter deals specifically with the extraction of the vehicle coefficients by the parameter optimization technique. This technique is based on a very simple concept. A function is defined which evaluates, from the relevant variables (parameters), the difference between the current status and the desired status. This difference, for instance, could be the RMS curve fitting error between the analytical coast-down function and actual coast-down data.

Once the error (difference) function has been defined, the heart of the technique utilizes an 'intelligent' trial-and-error algorithm to find the optimum combination of input variables (parameters), which result in the least error (difference). For our purposes, these variables would be the vehicle coefficients (or possibly the coefficients of the drag-force/speed quadratic equation). The final values of the variables would therefore be taken as the best estimate of the actual vehicle coefficients (or coefficients of the drag-force/speed quadratic function).

4.1 Choice Of Optimization Routine

Although the general concept of parameter optimization is very simple, the actual trial-and-error algorithm can be quite sophisticated if the routine is to be efficient in terms of computational time. In view of this, it was decided to make use of one of the commercially available routines.

The only routines available at Loughborough University, at the present time, are those contained in the Numerical Algorithms Group (NAG) FORTRAN library (Mark 10), which is available on the three PRIME and one HONEYWELL (MULTICS) computers at the university. In the NAG documentation, the optimization subroutines are described in chapter E04.

Chapter E04 of the NAG documentation describes routines for solving many types of optimization problem, however, Prof Storey[21] recommends that a least squares, non-linear, unconstrained algorithm be chosen. For this particular problem, no derivatives are available for the error function, so the selection chart for unconstrained problems indicates the use of either the E04FCF routine, or the E04FDF routine. E04FCF is a comprehensive routine which should only be selected when the user has considerable experience with optimization techniques, and so it has been left as a future development prospect. E04FDF is an easy to use routine, and was therefore used for the work in this thesis.

The choice of NAG routine E04FDF was the first major development of the original coefficient extraction method, which is described by Emtage[2], in which the NAG routine E04JAF was used. This routine was not least squares, and required the definition of simple constraints.

4.2 Implementation Of Optimization Routine

The NAG routine E04FDF was implemented in program CSTDWN as described in appendix C. The main point to note is that E04FDF requires a user-defined subroutine which must evaluate the desired least squares function.

When E04FDF calls the user-defined subroutine (which must be named 'LSFUN1'), it supplies some combination of the variables which define the function. LSFUN1 then uses these values of the variables to calculate the error at each data point along the coastdown curve. These values of error are then returned via the subroutine argument list. It should be noted that LSFUN1 does not actually calculate the squares of the errors. This is in fact carried out in NAG routine E04FDF itself.

In program CSTDWN, three variables are used and these are defined as being the coefficients of the drag-force/speed quadratic function. LSFUN1 is called many times during program execution, by NAG routine E04FDF, and must therefore be as efficient, in terms of computational time, as possible. This is one of the main reasons for correcting for ambient conditions before entering the coefficient extraction process. It should be noted that the final version of CSTDWN allows any one or two of the vehicle coefficients to be treated as constants. This is useful when one or two of the coefficients are known.

The error terms are calculated in LSFUN1 by calculating the theoretical speed at every recorded time, via equation 2.34, and then finding the difference between these values and the recorded (corrected) data at every point along the coast-down curve. In the latest version of program CSTDWN the conditions governing the integral solution (see appendix A) are checked and the appropriate analytical function used.

It is important for parameter optimization that the problem is 'scaled' correctly. To quote from NAG documentation, chapter E04FDF, section 11 (Further Comments); "Ideally the poblem should be scaled so that the minimum value of the sum of squares is in the range (0,+1) and so that at points a unit distance away from the solution the sum of squares is approximately a unit value greater than at the minimum." The problem of scaling was mostly ignored by Emtage[2] and this is another reason why this development was required. For the above reasons, program CSTDWN was written so as to incorporate appropriate scale factors.

Most, if not all, of the NAG routines heve an argument called 'IFAIL' which serves as an error severity indicator. (In the output from program CSTDWN, IFAIL is abbreviated as IFL.). Routine E04FDF is called with IFAIL=1, and the most common resulting values of IFAIL are listed below with explanations:

IFAIL=0 - the optimization process has been completely successful

IFAIL=3 - none of the conditions for a minimum have been satisfied but no lower value could be found

IFAIL=5 to 8 - progressively (from 5 to 8) fewer of the conditions for a minimum have been satisfied but no lower point could be found

From experience it has been found that values of IFAIL other than zero are usually the result of poor scaling.

4.3 Choice Of Scale Factors

As has been indicated above, the choice of scale factors is very important in terms of computational time, and in the accuracy of finding the true minimum of the error function.

The problem was tackled by writing a FORTRAN program called 'SCALE'. For reference purposes, appendix D contains a listing of program SCALE.

Program SCALE reads in one coast-down data set, and then allows the user to choose a set of scale factors. The actual coordinates of the minimum (vehicle coefficients) must be known for the data set being used, and these are requested by the program. Program SCALE then calculates and displays the change in the sum of squares for a unit change in each of the three scaled vehicle coefficients. Correct scaling is achieved when the results are all unity.

4.4 The Minimization Problem

With most problems it is usually useful , if not essential, to be able to visualize the problem. With this in mind, it was decided to investigate the 'surfaces' representing the curve fitting error, which is to be minimized.

In order to produce an error surface, an arbitary coast-down data set was chosen, whose optimized coefficients were known (ie coefficients for true minimum). A small FORTRAN program was written, which called subroutine LSFUN1 (see appendix C) in order to evaluate the RMS curve fitting error. The value of RMS curve fitting error could therefore be evaluated for any set of values for the coefficients. By keeping one coefficient constant at its optimized value, a rectangular grid of RMS curve fitting error values could be produced by varying the other two coefficients about their optimized values. Since there is a choice of three coefficients to be held constant, there are three grids of RMS curve fitting error to be produced. Isometric projections and contour maps may then be produced from these grids.

Figs 4.1 - 4.3 give isometric projections of typical RMS error grids, with the coefficients A_D, B_D and C_D being held constant in turn. These figures at once show that the RMS error surfaces are valleys with steep sides, but with quite flat valley floors (along the length of the valley). Figs 4.4 - 4.6 give contour maps for the same RMS error grids, and are just another way of presenting the same information. Figs 4.7 - 4.10 define and display sections taken across and along the valley for one of the grids.

From this initial study of the RMS error surfaces it is possible to come to a few conclusions:

1) It wold be relatively easy for the optimization algorithm to find the bottom of the valley since the sides are steep.

2) It is considerably more difficult for the optimization algorithm to continue on and find the real minimum once the valley floor has been reached. This is because the valley floor is so flat.

3) In view of the above it is clear as to why correct scaling is necessary.

4.5 95% Confidence Limits On Extracted Coefficients

Providing that the input coast-down data is absolutely accurate, and assuming that the analytical coast-down function is appropriate, then the coordinates of the minimum RMS error will be exactly those of the coefficients for the vehicle in question. If, on the other hand, the input data should contain some level of random system error (such as might be caused by instrumentation error, road roughness, wind fluctuations etc.), then it is no longer certain that the minimum of the RMS error function will coincide with the position defined by the actual vehicle coefficients.

It should also be clear that if the optimization process does not quite reach the true minimum (because of poor scaling for example), then, because of the long flat valley floor, the resulting values of the vehicle coefficients could be very much in error.

In view of the above comments, it is necessary to investigate the relationship between the possible error in the extracted coefficients and the amount of random error in the input data. For the purposes of this work, it is now assumed that the analytical model being used is appropriate, and that the scaling is optimum. Therefore the RMS curve fitting error may be assumed to be entirely due to random error in the input data.

Dr Pettitt[22] suggested the following method for determining the 95% confidence limits of the extracted vehicle coefficients:

Let $SS(A_D,B_D,C_D)$ be the sum-of-squares curve fitting error for a trial point (A_D,B_D,C_D), and let $SS(\hat{A}_D,\hat{B}_D,\hat{C}_D)$ be the minimum sum-of-squares curve fitting error which occurs at the 'best-fit' point $(\hat{A}_D,\hat{B}_D,\hat{C}_D)$. Then it may be stated, with 95% confidence, that the true values of the coordinates lie within the limits defined by:

$$SS(A_D,B_D,C_D) \leqslant SS(\hat{A}_D,\hat{B}_D,\hat{C}_D)\times(1 + \frac{\chi^2_\nu}{N-3}) \qquad (4.1)$$

where,

N = the number of error-squared values making up the sum-of-squares value (ie the number of data points)

and,

ν = the number of degrees of freedom (ie the number of variables involved, which would normally be three)

and,

χ^2 = statistical quantity for 95% confidence limits

The actual confidence limits, in terms of RMS curve fitting error, are therefore obtained by replacing the inequality in expression 4.1 with an equality, and by taking the square-root of both sides of the resulting equation. This gives:

$$RMS_{95\%} = \sqrt{(RMS(\hat{A}_D,\hat{B}_D,\hat{C}_D)^2 \times (1 + \frac{\chi^2_\nu}{N-3}))} \qquad (4.2)$$

Therefore, once the minimum RMS curve fitting error is known, then the RMS curve fitting error defining the 95% confidence limits of the coefficients can be calculated.

4.5.1 One Degree Of Freedom System

For one degree of freedom systems (ie where two of the three coefficients are known), the confidence limits for the variable coefficient may be found by plotting RMS error versus the value of that coefficient, in the region of the minimum RMS error. If the horizontal line representing the 95% RMS error is superimposed on this graph, then the intersections of the two lines define the 95% confidence limits for that coefficient. Figs 4.11 - 4.13 give typical examples of such graphs.

4.5.2 Two Degrees Of Freedom System

For two degrees of freedom (ie where only one coefficient is known), the 95% RMS confidence limits are represented by a contour on a plane defined by the two variable coefficients. Examples of these contour maps are given in Figs 4.14 - 4.16. The projections of these contours, onto the axes, define the actual 95% confidence limits of the coefficients.

4.5.3 Three Degrees Of Freedom System

As a natural progression from one and two degrees of freedom, it is clear that, for three degrees of freedom, the 95% RMS error limit will be represented by a three dimensional closed shell. The following scheme was developed in order to afford a visualization of this shell of constant RMS error. Contour maps, similar to those for the two degrees of freedom system, were drawn for different values of B_D (A_D and C_D being treated as variables for each contour). The values of B_D were chosen about the point representing the true minimum of the RMS error function. The resulting contours were then plotted in a three dimensional axes system in order to represent the RMS error shell. Figs 4.17 - 4.21 give different views of the shell.

An interesting point to note is that A_D and C_D decrease as B_D increases, and it will be worthwhile to watch out for this type of trade-off between the extracted coefficients from real test data, because it would indicate that errors in the coefficients were likely to be related to the RMS error only.

Another point to note is that the confidence limits of the coefficients are more-or-less indicated by the two end points of the error shell. This provides an easier way of finding the 95% confidence limits for the three degree of freedom system. The method is as follows:

1) Choose a value of B_D and, treating A_D and C_D as variables, use the coefficient extraction algorithm to find the minimum RMS error, and the corresponding values of A_D and C_D.

2) Repeat (1) for different values of B_D about its true value for the test data in question, and plot the values of minimum RMS error against the values of the three coefficients.

3) Plot the line representing the 95% RMS confidence limits onto the three graphs thus produced, and the intersections between the two lines will define the 95% confidence limits for the three coefficients.

Figs 4.22 - 4.24 give typical examples of such graphs.

4.5.4 Maximum Error In Coefficients Versus RMS Error

So far all of the work has been concerned with one particular data set, which has one particular minimum RMS error value. It would be interesting to see how the confidence limits varied with the RMS error. In fact, although it was a very tedious business, an investigation was carried out, for various values of RMS error between 0.00 and 0.08, and the results are given in Figs 4.25 - 4.33. The vertical axis for each of the three graphs is in terms of the range, between the 95% confidence limits, for the particular coefficient in question.

As can be seen from the figures, the characteristic of the maximum coefficient error, in relation to the RMS error, is a linear one. This means that, given the slope of the characteristic (given on each graph), the value of maximum coefficient error may be calculated for any value of RMS error.

Figs 4.25 - 4.33 clearly indicate that the maximum error in a particular coefficient increases with RMS error. Therefore, if it is possible to find the value of one or two of the coefficients by some other means, then it would be possible to substantially reduce the error in the other coefficient(s).

4.6 Statistical Weights For Extracted Coefficients

Fig 4.34 gives a typical probability density bar chart for RMS error, which was obtained for a large number of test sets. This figure clearly shows the sort of range of RMS values that might be expected. As was indicated in the previous section, we have more confidence in the coefficients corresponding to low RMS error values, and therefore a suitable weighting scheme must be introduced in order to calculate a sensible overall average for each coefficient.

At this point it should be pointed out that the test data falls into natural groups, in which all tests were carried out within a short space of time. This ensured that the ambient conditions were more-or-less constant within a group. The tests within a group shall be designated by subscript 'i' and the groups by subscript 'j'. The following scheme was suggested by Dr Pettitt[22]:

The statistical weight for the i^{th} coefficient in the j^{th} group is given by:

$$w_{ij} = \frac{1}{\sigma^2_{ij}} \tag{4.3}$$

where,

σ_{ij} = standard deviation = 1/4 length of 95% confidence interval

The average for the j^{th} group is given by:

$$(av)_j = \frac{\sum_i x_{ij} w_{ij}}{\sum_i w_{ij}} \tag{4.4}$$

And the standard error in this average is given by:

$$S.E.(av)_j = \surd(1/\sum_i w_{ij}) \tag{4.5}$$

The average for all groups is given by:

$$av = \frac{\sum_j \sum_i x_{ij} w_{ij}}{\sum_j \sum_i w_{ij}} \tag{4.6}$$

And the standard error for this average is given by:

$$S.E.(av) = \surd(1/\sum_j \sum_i w_{ij}) \tag{4.7}$$

The above scheme was implemented in program CSTDWN, and the slopes from Figs 4.25 - 4.33 were stored so that the 95% confidence intervals (and hence the statistical weights) of the coefficients could be calculated for any value of RMS error.

The stored values of the slopes for Figs 4.25 - 4.33 are of course only relevant to one particular vehicle, and it would be very tedious to have to do this work all over again for another vehicle. In view of this, it should be noted that NAG optimization routine E04FCF permits the user to calculate the standard errors for the extracted variables directly. If this were to be implemented as a future development, it would provide a very versatile coefficient extraction algorithm.

Figs 4.35 - 4.37 show how the statistical weights, for the three vehicle coefficients, vary with RMS error.

CHAPTER 5

SPECIFICATION OF THE COAST-DOWN TEST PROCEDURE

As was mentioned in section 1.3, there is very little information available concerning the relationship between the coast-down test procedure, and the expected accuracy of the extracted vehicle coefficients. In view of this it is hoped that this chapter will provide the foundation for a more detailed study in the future.

5.1 Effect Of RMS Error On Coefficient Accuracy

Section 4.5.4 pointed out that the accuracy of the coefficients is directly proportional to the RMS curve fitting error. Section 3.3.3 also suggests that the majority of the error arises from random sources in the system such as wind fluctuations, uneven road surface, instrumentation error, etc. In view of these two points it would be sensible to design the coast-down test procedure so as to minimize the RMS error, and this in turn would mean minimizing the random error in the system.

It is not sufficient to minimize the RMS error because we also need to know what its maximum allowable value is in terms of the random system error.

The largest data set available for this work resulted in an average RMS error of about 0.056 (see Fig 4.34). Referring to Figs 4.31 - 4.33, this value of RMS error corresponds to 95% confidence intervals of 0.00245, 0.000318 and 0.091 for A_D, B_D and C_D respectively. Assuming typical values for A_D, B_D and C_D to be 0.01, 0.0004 and 0.35 respectively, results in errors of ±12.3%, ±39.8% and ±13.0% for A_D, B_D and C_D respectively.

The above errors are unacceptably high, so assuming an error of ±1.0% for C_D, which gives a 95% confidence interval of 0.007035, would require an RMS error of 0.0043. Assuming this value of RMS error, the 95% confidence intervals for A_D and B_D would be 0.0001894 and 0.00002450 respectively, which would result in errors of ±0.95% and ±3.1% for A_D and B_D respectively. These errors are more acceptable, and therefore the design of the coast-down test procedure should aim at keeping the RMS error below 0.004.

5.2 FORTRAN Program CSTSIM

The next step in this study was to investigate the relationship between the random system error and the RMS error. Because of this, it was necessary to develop a computer program which could generate simulated coast-down data with varying degrees of random system error. The FORTRAN program CSTSIM was developed for this purpose, and a listing of CSTSIM is included in appendix E.

Program CSTSIM utilizes the full analytical coast-down function (described in section 2.3.3) to produce a simulated coast-down curve, with speed as the independent variable, and with equal speed intervals.

Program CSTSIM introduces 'normally' distributed random error into the vehicle speed, track gradient, wind speed and wind direction. The standard deviations for these four parameters are chosen by the user (values of zero would result in no error). NAG routine G05CCF was used to initialize the random number generator with a 'seed' based on 'real-time', and NAG routine G05DDF was used to actually generate the random error.

It should be noted that introducing error into the vehicle speed measurement is equivalent to introducing it into the elapsed time measurement.

5.3 Effect Of Random System Error

Program CSTSIM was used to produce a large number of test sets for each of nine values of standard deviation in each of the four parameters. The data sets were then analysed by program CSTDWN and the means and standard deviations of the resulting RMS errors were calculated. The results of this work are given in Figs 5.2 - 5.9.

Fig 5.2 indicates that the standard deviation in the vehicle speed measurement would have to be less than 0.0045 m/s in order to maintain a maximum RMS error of 0.004. Similarly, Figs 5.4, 5.6

and 5.8 indicate that values of 0.00015, 0.15 m/s and 3 deg would be applicable for the standard deviations of track gradient, wind speed and wind direction respectively. Of course, these values are calculated with the assumption that only one source of error is active at any one time, and since this clearly cannot be the case, then the allowable standard deviations in these parameters must be even smaller.

The standard deviations in the wind measurement could probabally be reduced substantially by adopting 'on-board' anemometry. The standard deviation in the track gradient could be reduced by undertaking a detailed survey of the test track and setting up markers at appropriate stations along the test track. It should be noted in passing, that bumps in the road surface will induce energy losses in the suspension, which will probably appear as a gradient induced error. The vehicle speed measurement is perhaps the most critical parameter, and at the same time the most difficult to deal with. One problem is that it would be rare for instrumentation manufacturers to quote standard deviations for measurements. It would therefore be necessary to conduct a detailed investigation into the accuracy of the instrumentation. Special care should be taken with 'fith-wheel' type speed indicators, as their mountings may introduce dynamic problems such as bounce. Also, it is no point in measuring accurately without also recording the results with enough precision.

5.4 Other Considerations

Other considerations concerning the coast-down test procedure are; proper recording of ambient conditions, test vehicle weight, static attitude, road surface moisture, vehicle warm-up and tyre break-in.

The vehicle warm-up period is essential in order for the tyres and the final drive unit to warm up to equilibrium temperatures. One tyre manufacturer[19] suggests a break-in period of about 100 miles be used for new tyres.

Care must be taken to ensure that the specification for the vehicle is rigidly adhered to, except where something is being varied intentionally in order to study its effect on the vehicle coefficients. An exception to this rule concerns the vehicle brakes. In order to eliminate the inconsistencies of brake drag, it is good practice, where safety is not sacrificed, to physically remove (or loosen) the brakes.

CHAPTER 6

RESULTS FROM REAL COAST-DOWN TESTS

This chapter gives the results of coast-down analysis (by FORTRAN program CSTDWN) on test data which was kindly supplied by Ford Motor Company. The total number of individual coast-down tests amounts to a figure of about 550. These consist of results for three different vehicles, and two different test facilities.

Vehicle 'A' was a medium size sports car, and the tests, which involved tyre comparisons, were carried out at a test facility in Europe. Results for nine different tyres ('BB', 'CC', 'D1', 'D2', 'EE', 'FF', 'GG', 'HH' and 'II') were recorded in three groups of approximately ten test sets each. Tyre type 'AA' was used as a 'control' tyre, and involved approximately 200 individual test sets. It should be noted that all of the recorded wind speeds recorded at this test facility were found to be in error and were duly corrected by the method described in section 3.4.

Vehicles 'B' and 'C' were both small family cars, and the tests, which involved comparisons of toe-in settings for the driven front wheels, were carried out at a test facility in the USA. It should be noted that there was also some doubt about the accuracy of the recorded wind speed measurements, but the fact that time was used as the independent variable meant that the correction method described in section 3.4 could not be used. For this reason a wind speed correction factor of approximately 0.6 was suggested by Ford, and this figure was taken account of in the analysis.

Three tables are given below to describe the results for all three vehicles.

Table 6.1 gives the #1 results file for vehicle 'A' with tyre type 'AA' being used. This analysis was carried out for three degrees of freedom.

Table 6.2 gives the summaries of analyses for vehicle 'A' and all tyre types, for both three and two degrees of freedom, B_D=0.0 for the two degrees of freedom system.

Table 6.3 gives the summaries of analyses for vehicles 'B' and 'C', for both three and two degrees of freedom, B_D=0.0 for the two degrees of freedom system.

TABLE 6.1

*
* RESULTS FROM PROGRAM CSTDWN BY A EMTAGE *
*

Vehicle: VEHICLE 'A' TYRE TEST DATA SET AA (WIND SPEED CORRECTED)
Track Gradient For +'ve Direction = 0.000000
Factor For Wind Speed = 1.000
Temperature Correction Factor For Ad = 0.0130

Data has been corrected to the following conditions:-

Standard Atmospheric Pressure   = 101.3 kPa
Standard Atmospheric Temperature = 20.0 deg C
Wind Speed   = 0.0 m/s
Track Gradient   = 0.0
Standard Vehicle Mass   = 1146.0 kg

Description Of Optimization Problem:-

Ad = Variable
Bd = Variable
Cd = Variable
Dd = 0.000D+00

Ad	Bd (s/m)	Cd	RMS ERROR	IFL	VWs (m/s)	VWd (deg)	Temp (deg C)	Press (kPa)	Dir	Mass (kg)
.0120	0.298D-03	.395	.0458	0	0.20	180.	11.	101.7	+	1146.
.0116	0.446D-03	.333	.0495	0	0.20	0.	11.	101.7	-	1146.
.0108	0.452D-03	.337	.0533	0	0.20	180.	11.	101.7	+	1146.
.0115	0.447D-03	.329	.0418	0	0.20	0.	11.	101.7	-	1146.
.0079	0.835D-03	.225	.0609	3	0.20	180.	11.	101.7	+	1146.
.0111	0.485D-03	.326	.0402	0	0.20	0.	11.	101.7	-	1146.
.0109	0.491D-03	.323	.0504	0	0.20	180.	11.	101.7	+	1146.
.0101	0.603D-03	.289	.0416	0	0.20	0.	11.	101.7	-	1146.
.0110	0.437D-03	.352	.0567	0	0.20	180.	11.	101.7	+	1146.
.0100	0.598D-03	.289	.0512	0	0.20	0.	11.	101.7	-	1146.
										2
.0097	0.614D-03	.289	.0511	0	1.15	180.	11.	101.7	+	1147.
.0120	0.310D-03	.385	.0472	0	1.15	0.	11.	101.7	-	1147.
.0102	0.525D-03	.319	.0510	5	1.15	180.	11.	101.7	+	1147.
.0124	0.299D-03	.376	.0436	0	1.15	0.	11.	101.7	-	1147.
.0103	0.638D-03	.276	.0556	5	1.15	180.	11.	101.7	+	1147.
.0112	0.557D-03	.282	.0460	0	1.15	0.	11.	101.7	-	1147.
.0097	0.727D-03	.239	.0593	0	1.15	180.	11.	101.7	+	1147.
.0102	0.676D-03	.261	.0385	0	1.15	0.	11.	101.7	-	1147.
.0110	0.522D-03	.314	.0525	0	1.15	180.	11.	101.7	+	1147.
.0111	0.558D-03	.284	.0442	0	1.15	0.	11.	101.7	-	1147.
										3

Ad	Bd (s/m)	Cd	RMS ERROR	IFL	VWs (m/s)	VWd (deg)	Temp (deg C)	Press (kPa)	Dir	Mass (kg)
.0113	0.327D-03	.403	.0601	0	2.15	180.	16.	101.6	+	1146.
.0108	0.434D-03	.350	.0503	0	2.15	0.	16.	101.6	-	1146.
.0112	0.484D-03	.329	.0563	5	2.15	180.	16.	101.6	+	1146.
.0112	0.386D-03	.370	.0590	5	2.15	0.	16.	101.6	-	1146.
.0114	0.427D-03	.345	.0448	0	2.15	180.	16.	101.6	+	1146.
.0112	0.416D-03	.353	.0490	5	2.15	0.	16.	101.6	-	1146.
.0102	0.491D-03	.330	.0424	0	2.15	180.	16.	101.6	+	1146.
.0118	0.345D-03	.377	.0414	0	2.15	0.	16.	101.6	-	1146.
.0105	0.525D-03	.312	.0514	0	2.15	180.	16.	101.6	+	1146.
.0096	0.732D-03	.247	.0532	5	2.15	0.	16.	101.6	-	1146.
										4
.0103	0.395D-03	.332	.0642	0	2.60	180.	17.	101.5	+	1146.
.0085	0.548D-03	.305	.0620	5	2.60	0.	17.	101.5	-	1146.
.0096	0.495D-03	.311	.0680	0	2.60	180.	17.	101.5	+	1146.
.0114	0.274D-03	.369	.0568	0	2.60	0.	17.	101.5	-	1146.
.0098	0.451D-03	.333	.0723	0	2.60	180.	17.	101.5	+	1146.
.0106	0.356D-03	.352	.0554	5	2.60	0.	17.	101.5	-	1146.
.0081	0.801D-03	.204	.0739	3	2.60	180.	17.	101.5	+	1146.
.0097	0.486D-03	.317	.0507	0	2.60	0.	17.	101.5	-	1146.
.0124	0.164D-03	.407	.0627	0	2.60	180.	17.	101.5	+	1146.
.0111	0.278D-03	.378	.0433	0	2.60	0.	17.	101.5	-	1146.
										5
.0076	0.777D-03	.203	.0979	3	0.90	135.	17.	101.3	+	1146.
.0094	0.533D-03	.292	.0633	5	0.90	45.	17.	101.3	-	1146.
.0083	0.777D-03	.213	.0600	0	0.90	135.	17.	101.3	+	1146.
.0085	0.607D-03	.274	.0575	0	0.90	45.	17.	101.3	-	1146.
.0101	0.431D-03	.330	.0830	0	0.90	135.	17.	101.3	+	1146.
.0078	0.783D-03	.219	.0633	0	0.90	45.	17.	101.3	-	1146.
.0099	0.412D-03	.336	.0699	5	0.90	135.	17.	101.3	+	1146.
.0084	0.698D-03	.246	.0782	5	0.90	45.	17.	101.3	-	1146.
.0096	0.501D-03	.303	.0638	0	0.90	135.	17.	101.3	+	1146.
.0119	0.110D-03	.409	.0675	0	0.90	45.	17.	101.3	-	1146.
										6
.0100	0.456D-03	.319	.0528	0	0.20	180.	13.	101.2	+	1147.
.0098	0.420D-03	.336	.0572	0	0.20	0.	13.	101.2	-	1147.
.0088	0.620D-03	.280	.0465	0	0.20	180.	13.	101.2	+	1147.
.0092	0.609D-03	.278	.0558	0	0.20	0.	13.	101.2	-	1147.
.0087	0.607D-03	.288	.0445	0	0.20	180.	13.	101.2	+	1147.
.0091	0.556D-03	.302	.0456	0	0.20	0.	13.	101.2	-	1147.
.0105	0.399D-03	.352	.0503	0	0.20	180.	13.	101.2	+	1147.
.0100	0.476D-03	.326	.0493	0	0.20	0.	13.	101.2	-	1147.
.0079	0.754D-03	.248	.0469	0	0.20	180.	13.	101.2	+	1147.
.0103	0.434D-03	.341	.0434	0	0.20	0.	13.	101.2	-	1147.
										7
.0118	0.216D-03	.408	.0278	0	0.80	180.	13.	102.1	+	1145.
.0127	0.115D-03	.430	.0409	0	0.80	0.	13.	102.1	-	1145.
.0102	0.390D-03	.363	.0393	0	0.80	180.	13.	102.1	+	1145.
.0111	0.307D-03	.379	.0405	0	0.80	0.	13.	102.1	-	1145.
.0112	0.288D-03	.395	.0397	0	0.80	180.	13.	102.1	+	1145.
.0108	0.360D-03	.364	.0385	0	0.80	0.	13.	102.1	-	1145.
.0107	0.391D-03	.353	.0409	0	0.80	180.	13.	102.1	+	1145.
.0118	0.241D-03	.402	.0367	0	0.80	0.	13.	102.1	-	1145.
.0103	0.407D-03	.349	.0381	5	0.80	180.	13.	102.1	+	1145.
.0107	0.377D-03	.359	.0316	0	0.80	0.	13.	102.1	-	1145.
										8

Ad	Bd (s/m)	Cd	RMS ERROR	IFL	VWs (m/s)	VWd (deg)	Temp (deg C)	Press (kPa)	Dir	Mass (kg)
.0113	0.177D-03	.433	.0375	0	0.95	180.	9.	102.2	+	1144.
.0093	0.439D-03	.357	.0425	0	0.95	0.	9.	102.2	-	1144.
.0111	0.172D-03	.428	.0375	0	0.95	180.	9.	102.2	+	1144.
.0093	0.514D-03	.328	.0383	5	0.95	0.	9.	102.2	-	1144.
.0103	0.392D-03	.354	.0392	0	0.95	180.	9.	102.2	+	1144.
.0108	0.299D-03	.379	.0323	0	0.95	0.	9.	102.2	-	1144.
.0099	0.433D-03	.348	.0424	0	0.95	180.	9.	102.2	+	1144.
.0106	0.316D-03	.382	.0342	0	0.95	0.	9.	102.2	-	1144.
.0105	0.365D-03	.372	.0362	0	0.95	180.	9.	102.2	+	1144.
.0108	0.293D-03	.388	.0405	5	0.95	0.	9.	102.2	-	1144.
										9
.0091	0.535D-03	.317	.0719	5	3.50	180.	15.	102.3	+	1146.
.0101	0.392D-03	.314	.0926	5	3.50	0.	15.	102.3	-	1146.
.0076	0.799D-03	.215	.1012	3	3.50	180.	15.	102.3	+	1146.
.0156	-.285D-03	.518	.0577	0	3.50	0.	15.	102.3	-	1146.
.0087	0.741D-03	.202	.0721	0	3.50	180.	15.	102.3	+	1146.
.0077	0.802D-03	.213	.0431	3	3.50	0.	15.	102.3	-	1146.
.0139	-.127D-03	.487	.0478	0	3.50	180.	15.	102.3	+	1146.
.0083	0.750D-03	.218	.0428	0	3.50	0.	15.	102.3	-	1146.
.0107	0.369D-03	.325	.0471	0	3.50	180.	15.	102.3	+	1146.
.0097	0.403D-03	.361	.0351	0	3.50	0.	15.	102.3	-	1146.
										10
.0116	0.112D-03	.450	.0473	0	2.10	180.	17.	102.3	+	1147.
.0108	0.332D-03	.366	.0363	0	2.10	0.	17.	102.3	-	1147.
.0126	0.447D-04	.448	.0347	0	2.10	180.	17.	102.3	+	1147.
.0105	0.405D-03	.337	.0408	5	2.10	0.	17.	102.3	-	1147.
.0102	0.411D-03	.329	.0440	0	2.10	180.	17.	102.3	+	1147.
.0114	0.238D-03	.390	.0386	0	2.10	0.	17.	102.3	-	1147.
.0114	0.219D-03	.401	.0408	0	2.10	180.	17.	102.3	+	1147.
.0110	0.275D-03	.377	.0249	0	2.10	0.	17.	102.3	-	1147.
.0114	0.248D-03	.398	.0419	0	2.10	180.	17.	102.3	+	1147.
.0102	0.393D-03	.345	.0313	0	2.10	0.	17.	102.3	-	1147.
										11
.0096	0.336D-03	.368	.0644	0	3.30	158.	18.	103.0	+	1146.
.0120	0.175D-03	.389	.0702	0	3.30	23.	18.	103.0	-	1146.
.0100	0.428D-03	.315	.0514	0	3.30	158.	18.	103.0	+	1146.
.0098	0.327D-03	.398	.0432	0	3.30	23.	18.	103.0	-	1146.
.0089	0.424D-03	.347	.0514	5	3.30	158.	18.	103.0	+	1146.
.0110	0.241D-03	.374	.0813	0	3.30	23.	18.	103.0	-	1146.
.0118	0.953D-04	.446	.0829	0	3.30	158.	18.	103.0	+	1146.
.0099	0.360D-03	.354	.0867	0	3.30	23.	18.	103.0	-	1146.
.0144	-.395D-03	.613	.0721	0	3.30	158.	18.	103.0	+	1146.
.0112	0.284D-03	.354	.0611	0	3.30	23.	18.	103.0	-	1146.
										12
.0160	-.467D-03	.573	.0634	0	4.10	135.	21.	103.0	+	1147.
.0119	0.630D-04	.442	.0510	0	4.10	45.	21.	103.0	-	1147.
.0122	0.427D-04	.408	.0658	0	4.10	135.	21.	103.0	+	1147.
.0092	0.232D-03	.413	.0326	0	4.10	45.	21.	103.0	-	1147.
.0152	-.592D-03	.657	.0567	0	4.10	135.	21.	103.0	+	1147.
.0086	0.308D-03	.392	.0497	0	4.10	45.	21.	103.0	-	1147.
.0129	-.103D-03	.501	.0434	0	4.10	135.	21.	103.0	+	1147.
.0098	0.226D-03	.389	.0446	0	4.10	45.	21.	103.0	-	1147.
.0135	-.207D-03	.535	.0692	0	4.10	135.	21.	103.0	+	1147.
.0100	0.329D-03	.350	.0421	0	4.10	45.	21.	103.0	-	1147.
										13

Ad	Bd (s/m)	Cd	RMS ERROR	IFL	VWs (m/s)	VWd (deg)	Temp (deg C)	Press (kPa)	Dir	Mass (kg)
.0104	0.417D-03	.386	.0567	0	2.70	180.	16.	101.6	+	1146.
.0097	0.468D-03	.284	.0456	0	2.70	0.	16.	101.6	-	1146.
.0079	0.813D-03	.215	.0845	3	2.70	180.	16.	101.6	+	1146.
.0085	0.812D-03	.199	.0714	3	2.70	0.	16.	101.6	-	1146.
.0104	0.409D-03	.281	.0766	0	2.70	180.	16.	101.6	+	1146.
.0094	0.467D-03	.378	.0631	0	2.70	0.	16.	101.6	-	1146.
										14
.0094	0.461D-03	.314	.0589	0	0.85	180.	16.	101.3	+	1146.
.0107	0.320D-03	.363	.0604	0	0.85	0.	16.	101.3	-	1146.
.0086	0.599D-03	.280	.0532	0	0.85	180.	16.	101.3	+	1146.
.0079	0.707D-03	.245	.0589	0	0.85	0.	16.	101.3	-	1146.
.0102	0.401D-03	.338	.0600	0	0.85	180.	16.	101.3	+	1146.
.0093	0.461D-03	.325	.0663	0	0.85	0.	16.	101.3	-	1146.
.0106	0.346D-03	.359	.0407	5	0.85	180.	16.	101.3	+	1146.
.0094	0.492D-03	.313	.0596	5	0.85	0.	16.	101.3	-	1146.
										15
.0084	0.555D-03	.292	.0649	0	0.20	180.	13.	101.1	+	1146.
.0087	0.704D-03	.246	.0424	0	0.20	0.	13.	101.1	-	1146.
.0085	0.574D-03	.308	.0481	0	0.20	180.	13.	101.1	+	1146.
.0100	0.473D-03	.323	.0472	5	0.20	0.	13.	101.1	-	1146.
.0094	0.487D-03	.327	.0511	0	0.20	180.	13.	101.1	+	1146.
.0094	0.532D-03	.304	.0484	0	0.20	0.	13.	101.1	-	1146.
.0080	0.726D-03	.234	.0518	0	0.20	180.	13.	101.1	+	1146.
.0173	-.684D-03	.749	.2302	0	0.20	180.	13.	101.1	+	1146.
										16
.0103	0.390D-03	.347	.0797	0	1.50	180.	10.	101.0	+	1146.
.0162	-.362D-03	.474	.1902	0	1.50	0.	10.	101.0	-	1146.
.0111	0.268D-03	.386	.0610	0	1.50	180.	10.	101.0	+	1146.
.0094	0.611D-03	.283	.0457	0	1.50	0.	10.	101.0	-	1146.
.0142	-.181D-03	.511	.1068	0	1.50	180.	10.	101.0	+	1146.
.0106	0.439D-03	.349	.0514	0	1.50	0.	10.	101.0	-	1146.
.0143	-.116D-03	.533	.0743	0	1.50	0.	10.	101.0	-	1146.
										17
.0107	0.350D-03	.349	.0841	0	5.05	180.	17.	101.3	+	1145.
.0077	0.753D-03	.227	.0617	5	5.05	0.	17.	101.3	-	1145.
.0124	0.131D-03	.424	.0927	5	5.05	180.	17.	101.3	+	1145.
.0101	0.302D-03	.381	.0440	0	5.05	0.	17.	101.3	-	1145.
.0084	0.681D-03	.259	.0836	0	5.05	180.	17.	101.3	+	1145.
.0076	0.809D-03	.222	.0765	3	5.05	0.	17.	101.3	-	1145.
.0148	-.138D-03	.458	.0724	0	5.05	180.	17.	101.3	+	1145.
										18
.0080	0.650D-03	.301	.0863	0	3.25	180.	16.	101.6	+	1148.
.0069	0.792D-03	.233	.0581	3	3.25	0.	16.	101.6	-	1148.
.0099	0.373D-03	.347	.0721	0	3.25	180.	16.	101.6	+	1148.
.0096	0.332D-03	.388	.0526	5	3.25	0.	16.	101.6	-	1148.
.0091	0.453D-03	.343	.0616	5	3.25	180.	16.	101.6	+	1148.
.0098	0.346D-03	.365	.0498	5	3.25	0.	16.	101.6	-	1148.
.0123	-.157D-04	.467	.0682	0	3.25	180.	16.	101.6	+	1148.
.0118	0.786D-04	.458	.0538	0	3.25	0.	16.	101.6	-	1148.
										19
.0263	-.159D-02	.875	.1976	0	3.90	180.	19.	101.4	+	1146.
.0094	0.552D-03	.286	.0627	5	3.90	0.	19.	101.4	-	1146.
.0134	-.135D-03	.498	.1137	0	3.90	180.	19.	101.4	+	1146.
.0085	0.575D-03	.302	.0496	0	3.90	0.	19.	101.4	-	1146.
.0165	-.365D-03	.502	.1186	5	3.90	180.	19.	101.4	+	1146.

Ad	Bd (s/m)	Cd	RMS ERROR	IFL	VWs (m/s)	VWd (deg)	Temp (deg C)	Press (kPa)	Dir	Mass (kg)
.0076	0.782D-03	.205	.1203	3	3.90	0.	19.	101.4	-	1146.
.0104	0.289D-03	.386	.0702	0	3.90	180.	19.	101.4	+	1146.
.0087	0.465D-03	.299	.0454	0	3.90	0.	19.	101.4	-	1146.
.0197	-.956D-03	.739	.0799	0	3.90	180.	19.	101.4	+	1146.
										20
.0104	0.371D-03	.342	.0407	0	0.95	180.	11.	102.2	+	1147.
.0098	0.489D-03	.320	.0306	0	0.95	0.	11.	102.2	-	1147.
.0106	0.365D-03	.366	.0410	0	0.95	180.	11.	102.2	+	1147.
.0106	0.409D-03	.328	.0366	0	0.95	0.	11.	102.2	-	1147.
.0116	0.224D-03	.403	.0409	0	0.95	180.	11.	102.2	+	1147.
.0114	0.267D-03	.389	.0361	0	0.95	0.	11.	102.2	-	1147.
.0095	0.513D-03	.317	.0390	0	0.95	180.	11.	102.2	+	1147.
.0107	0.343D-03	.367	.0309	5	0.95	0.	11.	102.2	-	1147.
.0109	0.338D-03	.364	.0385	0	0.95	0.	11.	102.2	-	1147.
										21
.0121	0.836D-04	.456	.0613	5	3.25	180.	18.	102.3	+	1147.
.0099	0.475D-03	.359	.0527	0	3.25	0.	18.	102.3	-	1147.
.0098	0.508D-03	.300	.0459	0	3.25	180.	18.	102.3	+	1147.
.0098	0.480D-03	.325	.0506	5	3.25	0.	18.	102.3	-	1147.
.0132	-.110D-03	.533	.0792	0	3.25	180.	18.	102.3	+	1147.
.0080	0.589D-03	.307	.0588	0	3.25	0.	18.	102.3	-	1147.
.0106	0.443D-03	.304	.0739	0	3.25	180.	18.	102.3	+	1147.
.0139	-.139D-04	.429	.0434	0	3.25	0.	18.	102.3	-	1147.
.0084	0.542D-03	.306	.0685	0	3.25	0.	18.	102.3	-	1147.
										22
.0106	0.223D-03	.397	.0457	0	3.40	180.	7.	103.0	+	1143.
.0098	0.357D-03	.358	.0336	0	3.40	0.	7.	103.0	-	1143.
.0107	0.202D-03	.391	.0371	0	3.40	180.	7.	103.0	+	1143.
.0108	0.216D-03	.405	.0438	5	3.40	0.	7.	103.0	-	1143.
.0110	0.149D-03	.425	.0399	0	3.40	180.	7.	103.0	+	1143.
.0090	0.475D-03	.318	.0397	0	3.40	0.	7.	103.0	-	1143.
.0097	0.406D-03	.353	.0400	0	3.40	180.	7.	103.0	+	1143.
.0112	0.122D-03	.427	.0341	0	3.40	0.	7.	103.0	-	1143.
.0088	0.462D-03	.332	.0320	0	3.40	0.	7.	103.0	-	1143.
										23
.0112	0.253D-03	.383	.0356	0	0.25	180.	21.	102.9	+	1147.
.0087	0.583D-03	.297	.0340	0	0.25	0.	21.	102.9	-	1147.
.0110	0.281D-03	.389	.0482	0	0.25	180.	21.	102.9	+	1147.
.0107	0.304D-03	.386	.0508	0	0.25	0.	21.	102.9	-	1147.
.0106	0.375D-03	.351	.0379	5	0.25	180.	21.	102.9	+	1147.
.0111	0.283D-03	.379	.0308	0	0.25	0.	21.	102.9	-	1147.
.0109	0.315D-03	.377	.0373	0	0.25	180.	21.	102.9	+	1147.
.0072	0.707D-03	.273	.0628	0	0.25	0.	21.	102.9	-	1147.
.0100	0.443D-03	.338	.0330	5	0.25	180.	21.	102.9	+	1147.

GROUP 1	WGTD. AVG.	S.E. AVG.
Ad	0.01084	0.167D-03
Bd (s/m)	0.4976D-03	0.216D-04
Cd	0.3228	0.619D-02

GROUP 2	WGTD. AVG.	S.E. AVG.
Ad	0.01085	0.166D-03
Bd (s/m)	0.5354D-03	0.215D-04
Cd	0.3040	0.618D-02

GROUP 3	WGTD. AVG.	S.E. AVG.
Ad	0.01094	0.173D-03
Bd (s/m)	0.4540D-03	0.223D-04
Cd	0.3419	0.642D-02

GROUP 4	WGTD. AVG.	S.E. AVG.
Ad	0.01032	0.204D-03
Bd (s/m)	0.3987D-03	0.264D-04
Cd	0.3393	0.759D-02

GROUP 5	WGTD. AVG.	S.E. AVG.
Ad	0.00916	0.237D-03
Bd (s/m)	0.5629D-03	0.306D-04
Cd	0.2831	0.879D-02

GROUP 6	WGTD. AVG.	S.E. AVG.
Ad	0.00938	0.169D-03
Bd (s/m)	0.5383D-03	0.219D-04
Cd	0.3059	0.629D-02

GROUP 7	WGTD. AVG.	S.E. AVG.
Ad	0.01116	0.127D-03
Bd (s/m)	0.3051D-03	0.165D-04
Cd	0.3813	0.473D-02

GROUP 8	WGTD. AVG.	S.E. AVG.
Ad	0.01045	0.131D-03
Bd (s/m)	0.3337D-03	0.170D-04
Cd	0.3783	0.487D-02

GROUP 9	WGTD. AVG.	S.E. AVG.
Ad	0.01020	0.181D-03
Bd (s/m)	0.4242D-03	0.235D-04
Cd	0.3254	0.673D-02

GROUP 10	WGTD. AVG.	S.E. AVG.
Ad	0.01109	0.126D-03
Bd (s/m)	0.2717D-03	0.163D-04
Cd	0.3820	0.468D-02

GROUP 11	WGTD. AVG.	S.E. AVG.
Ad	0.01051	0.215D-03
Bd (s/m)	0.2713D-03	0.278D-04
Cd	0.3856	0.798D-02

GROUP 12	WGTD. AVG.	S.E. AVG.
Ad	0.01122	0.167D-03
Bd (s/m)	0.6422D-04	0.217D-04
Cd	0.4456	0.622D-02

GROUP 13	WGTD. AVG.	S.E. AVG.
Ad	0.00956	0.280D-03
Bd (s/m)	0.5261D-03	0.362D-04
Cd	0.3023	0.104D-01

GROUP 14	WGTD. AVG.	S.E. AVG.
Ad	0.00964	0.216D-03
Bd (s/m)	0.4624D-03	0.280D-04
Cd	0.3208	0.803D-02

GROUP 15	WGTD. AVG.	S.E. AVG.
Ad	0.00900	0.205D-03
Bd (s/m)	0.5752D-03	0.266D-04
Cd	0.2923	0.763D-02

GROUP 16	WGTD. AVG.	S.E. AVG.
Ad	0.01102	0.277D-03
Bd (s/m)	0.3450D-03	0.358D-04
Cd	0.3672	0.103D-01

GROUP 17	WGTD. AVG.	S.E. AVG.
Ad	0.01005	0.280D-03
Bd (s/m)	0.4072D-03	0.363D-04
Cd	0.3355	0.104D-01

GROUP 18	WGTD. AVG.	S.E. AVG.
Ad	0.00974	0.234D-03
Bd (s/m)	0.3603D-03	0.303D-04
Cd	0.3675	0.870D-02

GROUP 19	WGTD. AVG.	S.E. AVG.
Ad	0.01067	0.263D-03
Bd (s/m)	0.2778D-03	0.341D-04
Cd	0.3691	0.978D-02

GROUP 20	WGTD. AVG.	S.E. AVG.
Ad	0.01058	0.134D-03
Bd (s/m)	0.3748D-03	0.173D-04
Cd	0.3535	0.498D-02

GROUP 21	WGTD. AVG.	S.E. AVG.
Ad	0.01075	0.206D-03
Bd (s/m)	0.3362D-03	0.266D-04
Cd	0.3636	0.764D-02

GROUP 22	WGTD. AVG.	S.E. AVG.
Ad	0.01011	0.138D-03
Bd (s/m)	0.2977D-03	0.179D-04
Cd	0.3762	0.514D-02

GROUP 23	WGTD. AVG.	S.E. AVG.
Ad	0.01034	0.141D-03
Bd (s/m)	0.3787D-03	0.182D-04
Cd	0.3550	0.523D-02

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01046	0.365D-04
Bd (s/m)	0.3743D-03	0.472D-05
Cd	0.3544	0.136D-02

```
* * * * * * * * * * *
*  END OF RESULTS   *
* * * * * * * * * * *
```

TABLE 6.2

THREE DEGREES OF FREEDOM

TYRE TYPE AA

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01046	0.365D-04
Bd (s/m)	0.3735D-03	0.472D-05
Cd	0.3547	0.136D-02

TYRE TYPE BB

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.00874	0.101D-03
Bd (s/m)	0.4198D-03	0.131D-04
Cd	0.3463	0.377D-02

TYRE TYPE CC

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01013	0.101D-03
Bd (s/m)	0.4291D-03	0.131D-04
Cd	0.3274	0.376D-02

TYRE TYPE D1

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.00887	0.118D-03
Bd (s/m)	0.3131D-03	0.153D-04
Cd	0.3687	0.438D-02

TYRE TYPE D2

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.00850	0.113D-03
Bd (s/m)	0.4372D-03	0.146D-04
Cd	0.3368	0.418D-02

TWO DEGREES OF FREEDOM (B_D = 0.0)

TYRE TYPE AA

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01252	0.813D-05
Bd (s/m)	0.0000D+00	0.692D-16
Cd	0.4501	0.232D-03

TYRE TYPE BB

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01131	0.231D-04
Bd (s/m)	0.0000D+00	0.183D-15
Cd	0.4544	0.658D-03

TYRE TYPE CC

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01262	0.231D-04
Bd (s/m)	0.0000D+00	0.189D-15
Cd	0.4377	0.658D-03

TYRE TYPE D1

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01081	0.244D-04
Bd (s/m)	0.0000D+00	0.189D-15
Cd	0.4490	0.697D-03

TYRE TYPE D2

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01147	0.270D-04
Bd (s/m)	0.0000D+00	0.189D-15
Cd	0.4545	0.769D-03

TYRE TYPE EE

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01090	0.105D-03
Bd (s/m)	0.4143D-03	0.136D-04
Cd	0.3398	0.390D-02

TYRE TYPE FF

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.00991	0.106D-03
Bd (s/m)	0.3287D-03	0.138D-04
Cd	0.3588	0.395D-02

TYRE TYPE GG

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.00997	0.974D-04
Bd (s/m)	0.3583D-03	0.126D-04
Cd	0.3596	0.362D-02

TYRE TYPE HH

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01000	0.938D-04
Bd (s/m)	0.2731D-03	0.121D-04
Cd	0.3877	0.349D-02

TYRE TYPE II

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.00909	0.105D-03
Bd (s/m)	0.3841D-03	0.135D-04
Cd	0.3544	0.388D-02

TYRE TYPE EE

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01321	0.245D-04
Bd (s/m)	0.0000D+00	0.196D-15
Cd	0.4470	0.698D-03

TYRE TYPE FF

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01167	0.228D-04
Bd (s/m)	0.0000D+00	0.200D-15
Cd	0.4410	0.650D-03

TYRE TYPE GG

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01210	0.230D-04
Bd (s/m)	0.0000D+00	0.192D-15
Cd	0.4542	0.657D-03

TYRE TYPE HH

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01146	0.197D-04
Bd (s/m)	0.0000D+00	0.183D-15
Cd	0.4538	0.563D-03

TYRE TYPE II

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01127	0.236D-04
Bd (s/m)	0.0000D+00	0.192D-15
Cd	0.4500	0.674D-03

TABLE 6.3

THREE DEGREES OF FREEDOM

VEHICLE 'B' TOE-IN = 3 MM

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01185	0.482D-04
Bd (s/m)	0.3343D-03	0.623D-05
Cd	0.4432	0.179D-02

VEHICLE 'B' TOE-IN = 8 MM

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01462	0.722D-04
Bd (s/m)	-.1253D-04	0.934D-05
Cd	0.5282	0.268D-02

VEHICLE 'B' TOE-IN = 13 MM

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01267	0.770D-04
Bd (s/m)	0.2824D-03	0.996D-05
Cd	0.4555	0.286D-02

VEHICLE 'C' TOE-IN = -5 MM

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01019	0.620D-04
Bd (s/m)	0.2459D-03	0.802D-05
Cd	0.4274	0.230D-02

VEHICLE 'C' TOE-IN = 0 MM

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01143	0.633D-04
Bd (s/m)	0.9203D-04	0.819D-05
Cd	0.4675	0.235D-02

TWO DEGREES OF FREEDOM (B_D = 0.0)

VEHICLE 'B' TOE-IN = 3 MM

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01349	0.902D-05
Bd (s/m)	0.0000D+00	0.267D-15
Cd	0.4993	0.258D-03

VEHICLE 'B' TOE-IN = 8 MM

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01430	0.128D-04
Bd (s/m)	0.0000D+00	0.267D-15
Cd	0.5098	0.367D-03

VEHICLE 'B' TOE-IN = 13 MM

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01418	0.161D-04
Bd (s/m)	0.0000D+00	0.267D-15
Cd	0.5003	0.459D-03

VEHICLE 'C' TOE-IN = -5 MM

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01155	0.129D-04
Bd (s/m)	0.0000D+00	0.267D-15
Cd	0.4647	0.367D-03

VEHICLE 'C' TOE-IN = 0 MM

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01194	0.118D-04
Bd (s/m)	0.0000D+00	0.267D-15
Cd	0.4801	0.336D-03

VEHICLE 'C' TOE-IN = 5 MM

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.00940	0.553D-04
Bd (s/m)	0.3002D-03	0.716D-05
Cd	0.4108	0.206D-02

VEHICLE 'C' TOE-IN = 5 MM

OVERALL	WGTD. AVG.	S.E. AVG.
Ad	0.01144	0.109D-04
Bd (s/m)	0.0000D+00	0.267D-15
Cd	0.4703	0.312D-03

CHAPTER 7

DISCUSSION OF RESULTS

7.1 Results For Vehicle 'A'

7.1.1 Results For Tyre Type 'AA'

Table 6.1 presents a large mass of information for a large number of individual test sets, and this can become very confusing. It was for this reason that program CSTDWN was written to produce a second results file, which contains no annotation (ie headings etc). This second results file contains the same information as the first one, except that it also includes the statistical weights for each individual extracted coefficient.

During the work on this thesis, the author developed a comprehensive computer graphics package, called TUFTY.PLOT, which was used to produce all of the graphs presented in this thesis. The entire second results file may be read by TUFTY.PLOT, and then graphs of any data column versus any data column may be plotted. This facility greatly simplifies the mass of data which is produced by program CSTDWN.

Figs 7.1 - 7.3 give graphs of the extracted values of the three coefficients versus RMS curve fitting error. These graphs clearly indicate the amount of scatter that exists in the results, and the fact that the scatter increases with RMS error.

Looking through table 6.1, it can be seen that a significant percentage of the test sets resulted in values of IFL other than zero. There is a case for completely excluding these results from the statistical analysis since the best curve fits have probably not been achieved.

The results summaries for each group of test sets are intended to give an insight into the problem of correcting for ambient conditions. Any variations between the group averages will be most likely due to errors when accounting for ambient conditions. The 95% confidence limits for these weighted averages are given by two standard deviations (standard errors) on either side of the average. Figs 7.4 - 7.6 give these confidence limits for all the test groups, and for all three coefficients. It is clear that the 95% confidence limits do not all overlap, and this therefore suggests that all of the ambient conditions have not been corrected for completely.

It should be borne in mind that ambient parameters such as humidity and track surface moisture have not been accounted for, and that the ambient conditions may not all have been accurately recorded (eg wind speed). Problems such as these can usually only be clarified by carrying out one's own carefully conducted test program, and that must now be left to some future work.

7.1.2 Summary Of Results For All Tyre Types

The results given in table 6.2 fall into two columns. The first column is for the three degrees of freedom system, while the second column is for the two degrees of freedom system with B_D=0.0.

The error associated with the three degrees of freedom system is significantly greater than that for the two degrees of freedom system. Also the wind tunnel value of C_D for vehicle 'A' is about 0.385, which is significantly higher than the typical values for the three degrees of freedom system. The values of C_D for the two degrees of freedom system are too high, but it is possible that they also include a rolling resistance term. In any case, the values of C_D, for the two degrees of freedom system, are more consistent for all the tyre types. This is to be expected since the aerodynamic characteristics had not been altered.

Figs 7.7 and 7.8 describe the 95% confidence limits for A_D and C_D respectively, for the overall results of each tyre type. These two graphs are for the three degrees of freedom system, while Figs 7.9 and 7.10 give the same information for the two degrees of freedom system. From these figures it is clear that the two degrees of freedom system provides a better method of determining the ranking for the values of A_D.

7.2 Results For Vehicles 'B' And 'C'

There is not much to discuss concerning the results for vehicles 'B' and 'C' (see table 6.3), except that the change in A_D with toe-in is not as expected. It would be logical if the value of A_D had a 'minimum' as toe-in varied, but the results consistently indicate that a 'maximum' occurs instead. One must remember, however, that there was some doubt as to the accuracy of the wind speed measurement, and the changes in A_D (for the two degrees of freedom system) with toe-in are small.

CHAPTER 8

CONCLUSIONS

1) Two analytical coast-down models have been postulated, but only the 'basic' model was implemented. This 'basic' model appears to be satisfactory, but more carefully recorded data is required in order to establish the model's validity.

2) Three correction techniques were considered for accounting for changes in ambient conditions The derivative method was used, and appears to be satisfactory, but inhibits any modifications to the analytical model. The simulation method is more accurate than the derivative approach, but may be more expensive (in terms of computational time) if first derivatives provide an adequate correction. The simulation method does have more flexibility in terms of allowing modifications to the analytical model. The third method of correction involves the inclusion of the appropriate terms into the analytical function during coefficient extraction, and would therefore be very expensive in terms of computational time. The derivative correction algorithm was tested and proved to be useful.

3) It was found that the scaling of the optimization problem was very important in order for the process to be efficient in finding, or even finding at all, the true minimum RMS curve fitting error.

4) It was found that the error in the extracted coefficients was proportional to the RMS curve fitting error, and a method was developed to calculate the statistical weights for the coefficients.

5) A large number of coast-down test results, for three different vehicles, were analysed, but the results were inconclusive as to the efficiency of the analysis. The reason for this is thought to be the level of error in the input data, especially with regards to the vehicle speed and wind velocity measurements. It was recommended that on-board anemometry be used in the future.

6) Because of the level of error in the data, it was found that the two degrees of freedom system (B_D=0.0) provided a better ranking for the values of A_D than did the three degrees of freedom system.

CHAPTER 9

RECOMMENDATIONS FOR FUTURE WORK

1) If pre-correction of the coast-down data is required, then it is recommended that the simulation based correction method be used.

2) If the comprehensive model is used, then the ambient conditions will have to be accounted for by including the appropriate terms into the analytical function used in the optimization algorithm.

3) It is recommended that more work be done on the basic model before attempting to implement the comprehensive model. This is because the comprehensive model will require numerical integration in order to obtain the coast-down form. The first modifications to the basic model might be to implement the analytical correction for yaw angle effect.

4) It is suggested that the NAG optimization routine E04FCF be implemented, since this would facilitate the automatic calculation of the standard errors in the extracted coefficients..

5) It is imperative that a careful error analysis be carried out for existing measurement systems associated with the coast-down test. This would be with a view to reducing the RMS curve fitting error. It is strongly recommended that on-board anemometry be implemented.

6) Extracted values of the coeficients should only be accepted when there is no doubt that a true minimum has been achieved (ie IFL=0).

LIST OF REFERENCES

1) LUCAS, G. G. - "Vehicle Performance Calculations", Ph.D Thesis, Loughborough University, 1970.

2) EMTAGE, A. L. - "The Deceleration Test - Correction For Ambient Conditions And Extraction Of Drag Coefficients", Final Year Project Thesis, Loughborough University, 1972.

3) DAYMAN, B. (Jr) - "Effects Of Realistic Tire Rolling Resistance Upon The Determination Of Aerodynamic Drag From Road-Vehicle Coast-Down Tests", 2nd AIAA Symp. On The Aerodyn. Of Sports & Competition Automobiles, Los Angeles, Calif., 1974.

4) DAYMAN, B. (Jr) - "Tire Rolling Resistance Measurements From Coast-Down Tests", SAE paper No. 760153, Autom. Eng. Congr. & Expos., Detroit, Michigan, Feb. 23 - 27, 1976.

5) FOGG, A. - "Measurement Of Aerodynamic Drag And Rolling Resistance Of Vehicles", Congr. Fed. Int. Soc. Ing. (FISITA), 1964.

6) Private Communications With Ford Motor Company (Ref. FTB), 1982.

7) KORST, H. H. & WHITE, R. A. - "Aerodynamic And Rolling Resistances Of Vehicles As Obtained From Coast-Down Experiments", Proc. 2nd. Int. Conf. On Veh. Mech., Paris, France, Sept. 6 - 10, 1971.

8) KORST, H. H. & FUNFSINN, M. A. - "Determination Of Effective Rolling Resistance By Coastdown Experiments On Smooth And Rough Roads", SAE Publication P74.

9) ROUSSILLON, G., MARZIN, J. L. & BOURHIS, J. - "Contribution To The Accurate Measurement Of Aerodynamic Drag By The Deceleration Method", Br., Hydrodyn. Res. Assoc., Adv. Road Vehic. Aerodyn., Paper 4, 1975.

10) ROUSSILLON, G. - "Determination Of The Rolling Resistance Of Various Tyres On Different Road Surfaces", MIRA Translation No.

63/79, From Paper Published In Ingenieurs de l'Automobile, Aug. 1978.

11) ROUSSILLON, G., MARZIN, J. L., BOURHIS, J. & LEFEUVRE, J. C. - "Contribution To The Improvement Of On-Road Measurement Of C_x By The Coastdown Method", MIRA Translation No. 62/79, From Paper Published In Ingenieurs de l'Automobile, Aug. / Sept., 1978.

12) ROUSSILLON, G. - "Contribution To Accurate Measurement Of Aerodynamic Drag On A Moving Vehicle From Coast-Down Tests And Determination Of Actual Rolling Resistance", 4th Colloq. Ind. Aerodyn., Aachen, June 18, 1980.

13) WHITE, R. A. & KORST, H. H. - "The Determination Of Vehicle Drag Contributions From Coast-Down Tests", SAE Paper No. 720099, SAE Trans. (81) Sec.1, pp354-9, 1972.

14) WHITE, R. A. & KORST, H. H. - "A Generalized Method For Determining Drag Coefficient Or Rolling Resistances From Coast Down Tests", 2nd Int. Symp. Rd. Veh. Aerodyn., 1974.

15) YASIN, T. P. - "The Analytical Basis Of Automobile Coastdown Testing", SAE Paper No. 780334.

16) INTERNATIONAL STANDARDS ORGANIZATION - "Draft Proposed ISO Standard: Chassis Dynamometer, Road Load Measurement And Simulation".

17) ECONOMIC COMMISION FOR EUROPE - "Resistance To Progress Of A Vehicle: Measurement Method On The Road; Simulation On A Chassis Dynamometer", E.C.E. Regulation No. 15, TRANS/SC1/WP29/R204, Annex 4, Appendix 3.

18) WATERS, D. M. - "Ground Vehicle Aerodynamics Course Notes", Dept. of Transport Technology, Loughborough University, 1981.

19) Private Communications With Dunlop Ltd (Ref. CWB), 1982.

20) SMITH, J. R., TRACY, J. C. & POTTER, D. S. - "Tire Rolling Resis-

tance - A Speed Dependent Contribution", SAE Paper No. 780255, Congr. & Expos., Cobo Hall, Detroit, Feb. 27 - Mar. 3, 1978.

21) STOREY, C. - "Private Communications", Mathematics Department, Loughborough University, 1982 - 1983.

22) PETTITT, A. N. - "Private Communications", Mathematics Department, Loughborough University, 1982 - 1983.

APPENDIX A

THE ANALYTICAL COAST-DOWN FUNCTION

Assuming that the drag-force/speed characteristic may be represented by a quadratic equation of the form,

$$F_D = a_0 + a_1 \times v + a_2 \times v^2 \tag{A1}$$

and also assuming that the drag-force is also given by,

$$F_D = -M_e \times \frac{dv}{dt} \tag{A2}$$

then,

$$\frac{dv}{a_0 + a_1 \times v + a_2 \times v^2} = -\frac{dt}{M_e} \tag{A3}$$

Integrating equation A3 (assuming initial time of zero) gives,

$$\int_{v_2}^{v_1} \left\{\frac{M_e}{a_0 + a_1 \times v + a_2 \times v^2}\right\} dv = t \tag{A4}$$

We may write,

$$\frac{a_0 + a_1 \times v + a_2 \times v^2}{M_e} = \beta\{(v + \gamma)^2 + \alpha^2\} \tag{A5}$$

where,

$$\alpha^2 = \frac{a_0}{a_2} - \frac{a_1^2}{4a_2^2} \tag{A6}$$

and,

$$\beta = \frac{a_2}{M_e} \tag{A7}$$

and,

$$\gamma = \frac{a_1}{2a_2} \tag{A8}$$

Substituting,

$$z = v + \gamma \tag{A9}$$

and therefore,

$$\frac{dz}{dv} = 1 \tag{A10}$$

Equation A4 may then be written as,

$$t = \frac{1}{\beta} \int_{z_2}^{z_1} \frac{dz}{(z^2 + \alpha^2)} \tag{A11}$$

This is a standard integral which has three possible solutions depending on the value of α^2. The three possible solutions (in terms of v) are as follows:

If $\alpha^2 > 0$ then,

$$t = \frac{1}{\alpha \times \beta}\left\{ \mathrm{Tan}^{-1}\left\{\frac{v_1 + \gamma}{\alpha}\right\} - \mathrm{Tan}^{-1}\left\{\frac{v_2 + \gamma}{\alpha}\right\}\right\} \tag{A12}$$

If $\alpha^2 = 0$ then,

$$t = \frac{1}{\beta}\left\{\frac{1}{(v_2 + \gamma)} - \frac{1}{(v_1 + \gamma)}\right\} \tag{A13}$$

If $\alpha^2 < 0$ then,

$$t = \frac{1}{\alpha \times \beta}\left\{ \mathrm{Tanh}^{-1}\left\{\frac{v_1 + \gamma}{\alpha}\right\} - \mathrm{Tanh}^{-1}\left\{\frac{v_2 + \gamma}{\alpha}\right\}\right\} \tag{A14}$$

Equations A12, A13 and A14 may be rearranged to give v_2 as a function of elapsed time:

If $\alpha^2 > 0$ then,

$$v_2 = \alpha\left[\frac{\left\{\frac{v_1 + \gamma}{\alpha}\right\} - \mathrm{Tan}(\alpha\beta t)}{1 + \left\{\frac{v_1 + \gamma}{\alpha}\right\}\mathrm{Tan}(\alpha\beta t)}\right] - \gamma \qquad (A15)$$

If $\alpha^2 = 0$ then

$$v_2 = \frac{1}{\left\{\beta t + \frac{1}{(v_1 + \gamma)}\right\}} - \gamma \qquad (A16)$$

If $\alpha^2 < 0$ then,

$$v_2 = \alpha\left[\frac{\left\{\frac{v_1 + \gamma}{\alpha}\right\} - \mathrm{Tanh}(\alpha\beta t)}{1 - \left\{\frac{v_1 + \gamma}{\alpha}\right\}\mathrm{Tanh}(\alpha\beta t)}\right] - \gamma \qquad (A17)$$

For the purposes of the derivative correction technique equation A12 is appropriate because, in the author's experience, α^2 is always greater than zero for a passenger car. Equation A12 may be rearranged to give:

$$\mathrm{Tan}(\alpha\beta t) = \alpha\left\{\frac{v_1 - v_2}{\alpha^2 + (v_1 + \gamma)(v_2 + \gamma)}\right\} \qquad (A18)$$

It has been shown[2], provided the time interval under consideration is less than seven seconds, that the 'Tan' function in equation A18 may be replaced by its argument. This then gives the function of time as:

$$t = \frac{1}{\beta}\left\{\frac{v_1 - v_2}{\alpha^2 + (v_1 + \gamma)(v_2 + \gamma)}\right\} \qquad (A19)$$

APPENDIX B

DERIVATIVES FOR THE CORRECTION ALGORITHM

Let,

$$K_1 = v_1 - v_2$$

$$K_2 = K_1 \times M_e$$

$$K_3 = m_s g A_D + m_s g B_D v + \frac{P_s A}{2RT_s} C_D v_1 v_2$$

$$K_4 = K_3^2$$

$$K_5 = K_3^3$$

$$C_1 = -\frac{P_s A C_D v_1 v_2}{2RT_s^2} - K_T m_s g A_D$$

$$C_2 = \frac{A C_D v_1 v_2}{2RT_s}$$

$$C_3 = \frac{P_s A C_D v_1 v_2}{2RT_s}$$

$$C_4 = \frac{P_s A C_D (v_1+v_2)}{2RT_s}$$

$$C_5 = m_s g$$

$$C_6 = g A_D + g B_D \frac{(v_1+v_2)}{2}$$

Then the first derivatives are,

$$\frac{\partial t}{\partial T} = -\frac{K_2 C_1}{K_4}$$

$$\frac{\partial t}{\partial P} = -\frac{K_2 C_2}{K_4}$$

$$\frac{\partial t}{\partial K_Y} = -\frac{K_2 C_3}{K_4}$$

$$\frac{\partial t}{\partial v_{hw}} = \frac{K_2 C_4}{K_4}$$

$$\frac{\partial t}{\partial i} = \frac{K_2 C_5}{K_4}$$

$$\frac{\partial t}{\partial m} = \frac{K_1}{K_3} - \frac{K_2 C_6}{K_4}$$

And the second order derivatives are,

$$\frac{\partial^2 t}{\partial T^2} = \frac{2K_2 C_1^2}{K_5} - \frac{K_2 A C_D v_1 v_2 P_s}{K_4 R T_s^2}$$

$$\frac{\partial^2 t}{\partial P^2} = \frac{2K_2 C_2^2}{K_5}$$

$$\frac{\partial^2 t}{\partial K_Y^2} = \frac{2K_2 C_3^2}{K_5}$$

$$\frac{\partial^2 t}{\partial v_{hw}^2} = \frac{2K_2 C_4^2}{K_5} - \frac{K_2 A C_D P_s}{K_4 R T_s}$$

$$\frac{\partial^2 t}{\partial i^2} = \frac{2K_2 C_5^2}{K_5}$$

$$\frac{\partial^2 t}{\partial m^2} = \frac{2K_2 C_6^2}{K_5} - \frac{2K_1 C_6}{K_4}$$

The second order cross-derivatives will not be given here since they are given in the listing of program CSTDWN (Appendix C) in the same format.

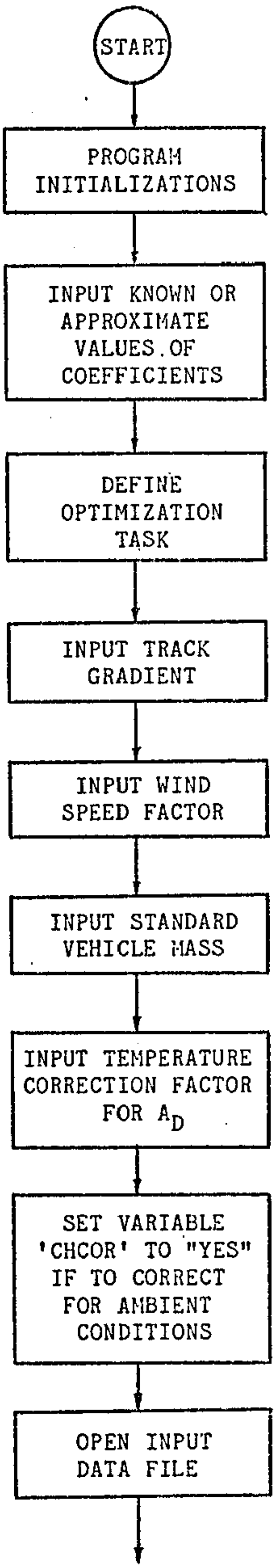
APPENDIX C

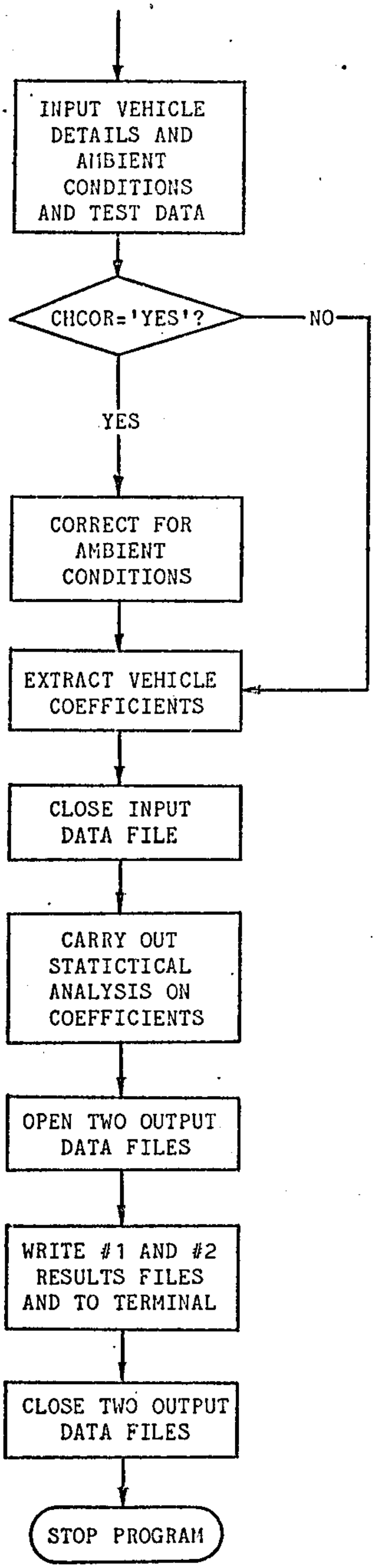
DESCRIPTION OF FORTRAN PROGRAM CSTDWN

Program CSTDWN was originally written to conform to the ansi-77 standard, but no guarantee is made that the current version complies totally to the standard. In fact, the current version of CSTDWN was written to run on the HONEYWELL (MULTICS) system at Loughborough University, and the compiler does not quite conform to the standard either.

The following sections give a general flow chart, a source listing, a description of variable names, a description of the input data file format, and source listings of some miscellaneous subroutines which are required.

General Flow Chart For CSTDWN




INPUT VEHICLE DETAILS AND AMBIENT CONDITIONS AND TEST DATA
CHCOR='YES'?
NO
YES
CORRECT FOR AMBIENT CONDITIONS
EXTRACT VEHICLE COEFFICIENTS
CLOSE INPUT DATA FILE
CARRY OUT STATICTICAL ANALYSIS ON COEFFICIENTS
OPEN TWO OUTPUT DATA FILES
WRITE #1 AND #2 RESULTS FILES AND TO TERMINAL
CLOSE TWO OUTPUT DATA FILES
STOP PROGRAM

Listing Of CSTDWN

```
C      AAA      L         EEEEE M   M TTTTT  AAA   GGG  EEEEE
C     A   A     L         E     MM MM   T   A   A G   G E
C     AAAAA     L         EEEE  M M M   T   AAAAA G ___ EEEE
C     A   A     L         E     M   M   T   A   A G   G E
C     A   A *   LLLLL *   EEEEE M   M   T   A   A  GGG  EEEEE

C     ******************************************************************
C     PROGRAM DECLARATIONS
C     ******************************************************************
      PROGRAM CSTDWN
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DOUBLE PRECISION M

      PARAMETER (NTERM=0, NDATA=5, NRES1=6, NRES2=7)

      CHARACTER *70  FILIN, FILOU1, VEHICL, ANSWER, ANSAD, ANSBD, ANSCD
     *              , FILOU2, CHAD, CHBD, CHCD, CHDD
      CHARACTER *8  DATE(500), STRING, CHCOR
      CHARACTER *5  SPUNIT, TEUNIT
      CHARACTER *1  DIRECT(500)

      COMMON /BLCK1/ OTIME(500), SPEED(500), M(500), WINDV(500)
     *, WINDD(500), PRESS(500), TEMP(500), CTIME(500), NTESTC(500)
     *, AD1(500), BD1(500), CD1(500), RMSER1(500), IFAIL1(500), N2A(500)
     *, SSWA(500), SSWB(500), SSWC(500), SSAWA(500), SSBWB(500)
     *, SSCWC(500), AVGA(500), AVGB(500), AVGC(500), SAVGA(500)
     *, SAVGB(500), SAVGC(500), SWA(500), SWB(500), SWC(500)

      AD = 0.010
      BD = 0.0004
      CD = 0.350
      DD = 0.0
      R = 0.288

      TSSWA = 0.0
      TSSWB = 0.0
      TSSWC = 0.0

      TSSAWA = 0.0
      TSSBWB = 0.0
      TSSCWC = 0.0
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT APPROXIMATE OR KNOWN VALUES OF COEFFICIENTS
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COU('Please enter approximate or known values for the')
      CALL COU('vehicle coefficients, or simply press RETURN to select')
      CALL COU('the default values; Ad=0.01, Bd=0.0004 (s/m), Cd=0.35,')
      CALL COU('Dd=0.0 (sq s/m)')

      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Ad =  ')
      READ (UNIT=NTERM, FMT='(A)') ANSWER
```

```
      CALL LFTJST(ANSWER,NCHAR)
      IF (NCHAR.EQ.0) GOTO 10
      DECODE (ANSWER, 8888) AD

10    WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Bd =  ')
      READ (UNIT=NTERM, FMT='(A)') ANSWER
      CALL LFTJST(ANSWER,NCHAR)
      IF (NCHAR.EQ.0) GOTO 20
      DECODE (ANSWER, 8888) BD

20    WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Cd =  ')
      READ (UNIT=NTERM, FMT='(A)') ANSWER
      CALL LFTJST(ANSWER,NCHAR)
      IF (NCHAR.EQ.0) GOTO 25
      DECODE (ANSWER, 8888) CD

25    WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Dd =  ')
      READ (UNIT=NTERM, FMT='(A)') ANSWER
      CALL LFTJST(ANSWER,NCHAR)
      IF (NCHAR.EQ.0) GOTO 30
      DECODE (ANSWER, 8888) DD
C     ------------------------------------------------------------------

C     ******************************************************************
C     DEFINE PROBLEM TO BE SOLVED
C     ******************************************************************
30    WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Let Ad be a variable?  ')
      READ (UNIT=NTERM, FMT='(A)') ANSAD
      CALL UPCASE(ANSAD)
      CALL LFTJST(ANSAD,NCHAR)
      IF (ANSAD.EQ.'Y'.OR.ANSAD.EQ.'YES'.OR.NCHAR.EQ.0) THEN
         ANSAD = 'Y'
         CHAD = 'Variable'
      ELSE
         WRITE (UNIT=CHAD, FMT='(F5.4)') AD
      END IF

      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Let Bd be a variable?  ')
      READ (UNIT=NTERM, FMT='(A)') ANSBD
      CALL UPCASE(ANSBD)
      CALL LFTJST(ANSBD,NCHAR)
      IF (ANSBD.EQ.'Y'.OR.ANSBD.EQ.'YES'.OR.NCHAR.EQ.0) THEN
         ANSBD = 'Y'
         CHBD = 'Variable'
      ELSE
         WRITE (UNIT=CHBD, FMT='(E9.3)') BD
      END IF

      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Let Cd be a variable?  ')
      READ (UNIT=NTERM, FMT='(A)') ANSCD
      CALL UPCASE(ANSCD)
      CALL LFTJST(ANSCD,NCHAR)
      IF (ANSCD.EQ.'Y'.OR.ANSCD.EQ.'YES'.OR.NCHAR.EQ.0) THEN
```

```
   ANSCD = 'Y'
   CHCD = 'Variable'
ELSE
   WRITE (UNIT=CHCD, FMT='(F5.3)') BD
END IF

WRITE (UNIT=CHDD, FMT='(E9.3)') DD

IF (ANSAD.EQ.'Y'.AND.ANSBD.EQ.'Y'.AND.ANSCD.EQ.'Y') THEN
   IA = 1
   IB = 2
   IC = 3
   NCOF = 3
   CUERAD = 4.404E-2 / 4.0
   CUERBD = 5.697E-3 / 4.0
   CUERCD = 1.636E+0 / 4.0
ELSE IF (ANSAD.NE.'Y'.AND.ANSBD.NE.'Y'.AND.ANSCD.NE.'Y') THEN
   WRITE (UNIT=NTERM, FMT=*)
   CALL COU('***ERROR***  At least one coefficient')
   CALL COU('must be variable.')
   GOTO 30
ELSE IF (ANSBD.EQ.'Y'.AND.ANSCD.EQ.'Y') THEN
   IA = 3
   IB = 1
   IC = 2
   NCOF = 2
   CUERAD = 0.000E+0 / 4.0
   CUERBD = 9.275E-4 / 4.0
   CUERCD = 4.608E-1 / 4.0
ELSE IF (ANSAD.EQ.'Y'.AND.ANSCD.EQ.'Y') THEN
   IA = 1
   IB = 3
   IC = 2
   NCOF = 2
   CUERAD = 7.253E-3 / 4.0
   CUERBD = 0.000E+0 / 4.0
   CUERCD = 2.070E-1 / 4.0
ELSE IF (ANSAD.EQ.'Y'.AND.ANSBD.EQ.'Y') THEN
   IA = 1
   IB = 2
   IC = 3
   NCOF = 2
   CUERAD = 1.256E-2 / 4.0
   CUERBD = 7.126E-4 / 4.0
   CUERCD = 0.000E+0 / 4.0
ELSE IF (ANSAD.EQ.'Y') THEN
   IA = 1
   IB = 2
   IC = 3
   NCOF = 1
   CUERAD = 3.043E-3 / 4.0
   CUERBD = 0.000E+0 / 4.0
   CUERCD = 0.000E+0 / 4.0
ELSE IF (ANSBD.EQ.'Y') THEN
   IA = 2
   IB = 1
   IC = 3
   NCOF = 1
   CUERAD = 0.000E+0 / 4.0
```

```
         CUERBD = 1.735E-4 / 4.0
         CUERCD = 0.000E+0 / 4.0
      ELSE IF (ANSCD.EQ.'Y') THEN
         IA = 2
         IB = 3
         IC = 1
         NCOF = 1
         CUERAD = 0.000E+0 / 4.0
         CUERBD = 0.000E+0 / 4.0
         CUERCD = 8.672E-2 / 4.0
      END IF
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT TRACK GRADIENT
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter track gradient for +''ve direction.  ')
      READ (UNIT=NTERM, FMT=*) GRADEP
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT WIND SPEED FACTOR
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter factor for wind speed. ')
      READ(UNIT=NTERM, FMT=*) WFACT
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT STANDARD VEHICLE MASS
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter standard vehicle mass (kg).  ')
      READ (UNIT=NTERM, FMT=*) SM
      SW = SM * 9.81
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT TEMPERATURE CORRECTION FACTOR FOR Ad
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter temperature correction factor for Ad.  ')
      READ (UNIT=NTERM, FMT=*) ADCORF
C     ------------------------------------------------------------------

C     ******************************************************************
C     CHOOSE WHETHER OR NOT TO CORRECT FOR AMBIENT CONDITIONS
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Correct for ambient conditions?  ')
      READ (UNIT=NTERM, FMT='(A)') CHCOR
      CALL UPCASE(CHCOR)
      CALL LFTJST(CHCOR,NCHAR)
      IF (CHCOR.EQ.'Y'.OR.CHCOR.EQ.'YES'.OR.NCHAR.EQ.0) CHCOR = 'Y'
C     ------------------------------------------------------------------
```

```
C     ******************************************************************
C     OPEN DATA FILE
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter name of data file. ')
      READ (UNIT=NTERM, FMT='(A)') FILIN
      OPEN (UNIT=NDATA, FILE=FILIN, FORM='FORMATTED', CARRIAGE=.FALSE.)
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT VEHICLE NAME
C     ******************************************************************
      READ (UNIT=NDATA, FMT='(A)') VEHICL
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT VEHICLE CONSTANTS
C     ******************************************************************
      READ (UNIT=NDATA, FMT=*) AREA, CAL1
      READ (UNIT=NDATA, FMT=*) RR,   CAL2
      READ (UNIT=NDATA, FMT=*) WI,   CAL3

      AREA = AREA * CAL1
      RR = RR * CAL2
      WI = WI * CAL3
      SEM = SM + WI / RR**2
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT UNITS FOR VEHICLE SPEED AND SET SPEED FACTOR
C     ******************************************************************
      READ (UNIT=NDATA, FMT='(A)') SPUNIT
      CALL UPCASE(SPUNIT)
      IF(SPUNIT.EQ.'KM/H') VCONST = 1.0 / 3.6
      IF(SPUNIT.EQ.'M/S' ) VCONST = 1.0
      IF(SPUNIT.EQ.'MPH' ) VCONST = 1.609344 / 3.6
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT NUMBER OF TEST GROUPS
C     ******************************************************************
      READ (UNIT=NDATA, FMT=*) N3
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT NUMBER OF TEST SETS IN GROUP
C     ******************************************************************
      K = 0

      DO 60, I3 = 1, N3
         READ (UNIT=NDATA, FMT=*) N2
         N2A(I3) = N2

         SSWA(I3) = 0.0
         SSWB(I3) = 0.0
         SSWC(I3) = 0.0

         SSAWA(I3) = 0.0
         SSBWB(I3) = 0.0
```

```
      SSCWC(I3) = 0.0
C     ----------------------------------------------------------------
C     ****************************************************************
C     INPUT NUMBER OF DATA POINTS IN TEST SET
C     ****************************************************************
      DO 50 I2 = 1, N2
         K = K + 1
         READ (UNIT=NDATA, FMT=*) N
C        -------------------------------------------------------------
C        *************************************************************
C        INPUT DATE OF TEST
C        *************************************************************
         READ (UNIT=NDATA, FMT='(A)') DATE(K)
C        -------------------------------------------------------------
C        *************************************************************
C        INPUT AMBIENT CONDITIONS
C        *************************************************************
         READ (UNIT=NDATA, FMT='(A)') DIRECT(K)
         READ (UNIT=NDATA, FMT=*)     M(K),     CAL4
         READ (UNIT=NDATA, FMT=*)     WINDV(K), CAL5
         READ (UNIT=NDATA, FMT=*)     WINDD(K)
         READ (UNIT=NDATA, FMT=*)     PRESS(K), CAL6
         READ (UNIT=NDATA, FMT=*)     TEMP(K)
         READ (UNIT=NDATA, FMT='(A)') TEUNIT

         M(K) = M(K) * CAL4
         DM = SM - M(K)
         W = M(K) * 9.81
         EM = M(K) + WI / RR**2
         WINDV(K) = WINDV(K) * CAL5
         PRESS(K) = PRESS(K) * CAL6
         WINDD(K) = WINDD(K) * 0.01745

         CALL UPCASE(TEUNIT)
         IF(TEUNIT.EQ.'F') THEN
            TEMP(K) = (TEMP(K) - 32.0) / 1.8 + 273.0
         ELSE IF(TEUNIT.EQ.'C') THEN
            TEMP(K) = TEMP(K) + 273.0
         END IF
C        -------------------------------------------------------------
C        *************************************************************
C        INPUT TEST DATA
C        *************************************************************
         DO 40 I=1, N
            READ (UNIT=NDATA, FMT=*) OTIME(I), SPEED(I)
            SPEED(I) = SPEED(I) * VCONST
40       CONTINUE
C        -------------------------------------------------------------
C        *************************************************************
C        CORRECT FOR AMBIENT CONDITIONS
C        *************************************************************
         IF (CHCOR.EQ.'Y') THEN
            CALL CORECT(AD,BD,CD,SM,M(K),SEM,AREA,R,TEMP(K),PRESS(K)
     *      ,WINDV(K),WINDD(K),WFACT,GRADEP,DIRECT(K),ADCORF,N,OTIME
```

```
     *           ,CTIME,SPEED)
            ELSE
               DO 45, I = 1, N
                  CTIME(I) = OTIME(I)
45             CONTINUE
            END IF
C           ----------------------------------------------------------

C           **********************************************************
C           CALCULATE DRAG COEFFICIENTS
C           **********************************************************
            CALL DRGCOF(CTIME,SPEED,N,SW,AREA,SEM,AD1(K),BD1(K),CD1(K)
     *      ,RMSER1(K),IFAIL1(K),AD,BD,CD,DD,IA,IB,IC,NCOF)

            DUM = (CUERAD * RMSER1(K))**2.0
            IF (DUM.GE.1.0E-30) THEN
               SWA(K) = 1.0 / DUM
            ELSE
               SWA(K) = 1.0E+30
            END IF

            DUM = (CUERBD * RMSER1(K))**2.0
            IF (DUM.GE.1.0E-30) THEN
               SWB(K) = 1.0 / DUM
            ELSE
               SWB(K) = 1.0E+30
            END IF

            DUM = (CUERCD * RMSER1(K))**2.0
            IF (DUM.GE.1.0E-30) THEN
               SWC(K) = 1.0 / DUM
            ELSE
               SWC(K) = 1.0E+30
            END IF

            SSWA(I3) = SSWA(I3) + SWA(K)
            SSWB(I3) = SSWB(I3) + SWB(K)
            SSWC(I3) = SSWC(I3) + SWC(K)

            SSAWA(I3) = SSAWA(I3) + (AD1(K) * SWA(K))
            SSBWB(I3) = SSBWB(I3) + (BD1(K) * SWB(K))
            SSCWC(I3) = SSCWC(I3) + (CD1(K) * SWC(K))
50       CONTINUE
60    CONTINUE
C     ----------------------------------------------------------------

C     ****************************************************************
C     CLOSE DATA FILE
C     ****************************************************************
      CLOSE (UNIT=NDATA)
C     ----------------------------------------------------------------

C     ****************************************************************
C     CALCULATE WEIGHTED AVERAGES AND STANDARD ERRORS OF THESE AVERAGES
C     FOR BOTH INDIVIDUAL GROUPS AND THE ENTIRE DATA SET
C     ****************************************************************
      DO 70, I3 = 1, N3
         AVGA(I3) = SSAWA(I3) / SSWA(I3)
         AVGB(I3) = SSBWB(I3) / SSWB(I3)
```

```
         AVGC(I3) = SSCWC(I3) / SSWC(I3)
         SAVGA(I3) = SQRT(1.0 / SSWA(I3))
         SAVGB(I3) = SQRT(1.0 / SSWB(I3))
         SAVGC(I3) = SQRT(1.0 / SSWC(I3))
         TSSWA = TSSWA + SSWA(I3)
         TSSWB = TSSWB + SSWB(I3)
         TSSWC = TSSWC + SSWC(I3)
         TSSAWA = TSSAWA + SSAWA(I3)
         TSSBWB = TSSBWB + SSBWB(I3)
         TSSCWC = TSSCWC + SSCWC(I3)
70    CONTINUE

      TAVGA = TSSAWA / TSSWA
      TAVGB = TSSBWB / TSSWB
      TAVGC = TSSCWC / TSSWC

      TSAVGA = SQRT(1.0 / TSSWA)
      TSAVGB = SQRT(1.0 / TSSWB)
      TSAVGC = SQRT(1.0 / TSSWC)
C     -------------------------------------------------------------------

C     *******************************************************************
C     OPEN RESULTS FILES
C     *******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter name of #1 results file.  ')
      READ (UNIT=NTERM, FMT='(A)') FILOU1
      OPEN (UNIT=NRES1, FILE=FILOU1, FORM='FORMATTED', CARRIAGE=.FALSE.)

      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter name of #2 results file.  ')
      READ (UNIT=NTERM, FMT='(A)') FILOU2
      OPEN (UNIT=NRES2, FILE=FILOU2, FORM='FORMATTED', CARRIAGE=.FALSE.)
C     -------------------------------------------------------------------

C     *******************************************************************
C     OUTPUT RESULTS TO TERMINAL AND RESULTS FILES
C     *******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      WRITE (UNIT=NTERM, FMT=*)

      WRITE (UNIT=NTERM, FMT=1001) VEHICL, GRADEP, WFACT, ADCORF
      WRITE (UNIT=NRES1, FMT=1001) VEHICL, GRADEP, WFACT, ADCORF

      IF (CHCOR.EQ.'Y') THEN
         WRITE (UNIT=NTERM, FMT=1002) SM
         WRITE (UNIT=NRES1, FMT=1002) SM
      ELSE
         WRITE (UNIT=NTERM, FMT=1003)
         WRITE (UNIT=NRES1, FMT=1003)
      END IF

      WRITE (UNIT=NTERM, FMT=1004) CHAD, CHBD, CHCD, CHDD
      WRITE (UNIT=NRES1, FMT=1004) CHAD, CHBD, CHCD, CHDD

      WRITE (UNIT=NTERM, FMT=1100)
      WRITE (UNIT=NRES1, FMT=1100)

      K = 0
```

```
      DO 90, I3 = 1, N3
         N2 = N2A(I3)
         DO 80, I2 = 1, N2
            K = K + 1
            WINDD(K) = WINDD(K) / 0.01745
            TEMP(K) = TEMP(K) - 273.0
            WRITE (UNIT=NTERM, FMT=1200) AD1(K), BD1(K), CD1(K),
     *      RMSER1(K), IFAIL1(K), WINDV(K), WINDD(K), TEMP(K), PRESS(K),
     *      DIRECT(K), M(K)
            WRITE (UNIT=NRES1, FMT=1200) AD1(K), BD1(K), CD1(K),
     *      RMSER1(K), IFAIL1(K), WINDV(K), WINDD(K), TEMP(K), PRESS(K),
     *      DIRECT(K), M(K)
            WRITE (UNIT=NRES2, FMT=1800) AD1(K), SWA(K), BD1(K),
     *      SWB(K), CD1(K), SWC(K), RMSER1(K), IFAIL1(K), WINDV(K),
     *      WINDD(K), TEMP(K), PRESS(K), DIRECT(K), M(K)
80       CONTINUE
         IF (I3.NE.N3)THEN
            WRITE (UNIT=NTERM, FMT=1300) (I3 + 1)
            WRITE (UNIT=NRES1, FMT=1300) (I3 + 1)
         END IF
90    CONTINUE

      WRITE (UNIT=NTERM, FMT=1400)
      WRITE (UNIT=NRES1, FMT=1400)

      DO 100, I3 = 1, N3
         WRITE (UNIT=STRING, FMT='(''GROUP'',I3)') I3
         WRITE (UNIT=NTERM, FMT=1500) STRING,
     *                                AVGA(I3), SAVGA(I3),
     *                                AVGB(I3), SAVGB(I3),
     *                                AVGC(I3), SAVGC(I3)
         WRITE (UNIT=NRES1, FMT=1500) STRING,
     *                                AVGA(I3), SAVGA(I3),
     *                                AVGB(I3), SAVGB(I3),
     *                                AVGC(I3), SAVGC(I3)
100   CONTINUE

      WRITE (UNIT=NTERM, FMT=1600) TAVGA, TSAVGA,
     *                             TAVGB, TSAVGB,
     *                             TAVGC, TSAVGC
      WRITE (UNIT=NRES1, FMT=1600) TAVGA, TSAVGA,
     *                             TAVGB, TSAVGB,
     *                             TAVGC, TSAVGC

      WRITE (UNIT=NTERM, FMT=1700)
      WRITE (UNIT=NRES1, FMT=1700)
C     ------------------------------------------------------------------

C     ******************************************************************
C     CLOSE RESULTS FILES
C     ******************************************************************
      CLOSE (UNIT=NRES1)
      CLOSE (UNIT=NRES2)
C     ------------------------------------------------------------------

C     ******************************************************************
C     TERMINATE PROGRAM
C     ******************************************************************
      STOP
```

```
C     ------------------------------------------------------------------
C
C     ******************************************************************
C     PROGRAM OUTPUT FORMATS
C     ******************************************************************
1001  FORMAT(
     * '* * * * * * * * * * * * * * * * * * * * * * * *'
     */'*  RESULTS FROM PROGRAM CSTDWN BY A EMTAGE  *'
     */'* * * * * * * * * * * * * * * * * * * * * * * *'
     */
     */'Vehicle:  ',A
     */'Track Gradient For +''ve Direction = ',F8.6
     */'Factor For Wind Speed = ',F5.3
     */'Temperature Correction Factor For Ad = ',F6.4)

1002  FORMAT(
     */
     */
     */'Data has been corrected to the following conditions:-'
     */
     */'Standard Atmospheric Pressure     = 101.3 kPa'
     */'Standard Atmospheric Temperature = 20.0 deg C'
     */'Wind Speed                        = 0.0 m/s'
     */'Track Gradient                    = 0.0'
     */'Standard Vehicle Mass             = ',F6.1,' kg')

1003  FORMAT(
     */
     */
     */'Data has NOT been corrected.')

1004  FORMAT(
     */
     */
     */'Description Of Optimization Problem:-'
     */
     */'Ad = ', A
     */'Bd = ', A
     */'Cd = ', A
     */'Dd = ', A//)

1100  FORMAT(
     */'!================================================================'
     *,'=============!'
     */'!   Ad       Bd        Cd     RMS    IFL  VWs    VWd    Temp    Press'
     *,'  Dir  Mass  !'
     */'!           (s/m)            ERROR       (m/s)  (deg) (deg C) (kPa)'
     *,'        (kg)  !'
     */'!================================================================'
     *,'=============!')

1200  FORMAT('! ',F5.4,2X,E9.3,2X,F4.3,2X,F5.4,3X,I1,2X,F5.2,
     *       3X,F4.0,3X,F3.0,3X,F5.1,3X,A,3X,F5.0,' !')

1300  FORMAT(
     * '!----------------------------------------------------------------'
     *,'----------',I3,'!')
```

```
1400  FORMAT(
     * '!================================================================'
     *,'=============!'//)

1500  FORMAT(
     * '              !------------!-----------!'
     */'   ', A8 ,' ! WGTD. AVG. ! S.E. AVG. !'
     */'!-----------!============!===========!'
     */'! Ad         ! ',F8.5,'   ! ', E9.3,' !'
     */'!-----------!------------!-----------!'
     */'! Bd (s/m) ! ', E10.4,' ! ', E9.3,' !'
     */'!-----------!------------!-----------!'
     */'! Cd         ! ',F8.4,'   ! ', E9.3,' !'
     */'!-----------!------------!-----------!'//)

1600  FORMAT(
     * '              !------------!-----------!'
     */'  OVERALL   ! WGTD. AVG. ! S.E. AVG. !'
     */'!-----------!============!===========!'
     */'! Ad         ! ',F8.5,'   ! ', E9.3,' !'
     */'!-----------!------------!-----------!'
     */'! Bd (s/m) ! ', E10.4,' ! ', E9.3,' !'
     */'!-----------!------------!-----------!'
     */'! Cd         ! ',F8.4,'   ! ', E9.3,' !'
     */'!-----------!------------!-----------!')

1700  FORMAT(//
     * '* * * * * * * * * * * *'
     */'*  END OF RESULTS   *'
     */'* * * * * * * * * * * *')

1800  FORMAT(7(E10.4,1X),I1,1X,4(E10.4,1X),A,'1',1X,E10.4)

8888  FORMAT(v)

      END
C     ##################################################################





C     ******************************************************************
C     SUBROUTINE TO DETERMINE THE COEFFICIENTS OF THE DRAG EQUATION
C     ******************************************************************
      SUBROUTINE DRGCOF(TIME,SPEED,NPTS,W,AREA,EMZ,AD1,BD1,CD1,RMSER1
     *,IFAIL1,ADS,BDS,CDS,DDS,IAZ,IBZ,ICZ,NCOFZ)

      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /TUFTY1/ T(500), V(500), COFZ(3)
      COMMON /TUFTY2/ EM, AC, BC, CC, FC, IA, IB, IC
      DIMENSION TIME(NPTS), SPEED(NPTS)
      DIMENSION COF(3), IW(5), WORK(5000)

      IA = IAZ
      IB = IBZ
      IC = ICZ
```

```
      NCOF = NCOFZ
      EM = EMZ
      R = 0.288
      PRESS = 101.3
      TEMP = 293.0
      RHO = PRESS / R / TEMP

      AS = ADS * W
      BS = BDS * W
      CS = (CDS * 0.5 * RHO * AREA) + (DDS * W)

      AC = 0.23
      BC = 4.2
      CC = 85.0
      FC = 1.414

      COF(IA) = AS * AC
      COF(IB) = BS * BC
      COF(IC) = CS * CC

      COFZ(1) = COF(1)
      COFZ(2) = COF(2)
      COFZ(3) = COF(3)

      DO 10, I = 1, NPTS
         T(I) = TIME(I)
         V(I) = SPEED(I)
10    CONTINUE

      ISUM = 0

20    IFAIL = 1
C     OPTIMIZATION ROUTINE
      CALL E04FDF(NPTS,NCOF,COF,FX,IW,5,WORK,5000,IFAIL)
      ISUM = ISUM + 1
      IF (ISUM.EQ.3) GOTO 30
      IF (IFAIL.EQ.2) GOTO 20

30    A = COF(IA) / AC
      B = COF(IB) / BC
      C = COF(IC) / CC

      AD1 = A / W
      BD1 = B / W
      CD1 = (C - (DDS * W)) / (0.5 * RHO * AREA)

      RMSER1 = SQRT(FX / NPTS) / FC
      IFAIL1 = IFAIL

      RETURN
      END
C     ####################################################################
```

```
C     ******************************************************************
C     FUNCTION TO BE MINIMIZED
C     ******************************************************************
      SUBROUTINE LSFUN1(NPTS,NV,COF,ERR)

      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION COF(NV), ERR(NPTS)
      COMMON /TUFTY1/ T(500), V(500), COFZ(3)
      COMMON /TUFTY2/ EM, AC, BC, CC, FC, IA, IB, IC
      COMMON /TUFTY3/ V1, ALFA, BETA, GAMA, CNST1

      V1 = V(1)
      ERR(1) = 0.0

      DO 10, I = 1, NV
         COFZ(I) = COF(I)
10    CONTINUE

      A = COFZ(IA) / AC
      B = COFZ(IB) / BC
      C = COFZ(IC) / CC

      TEST = (A / C) - (B**2 / 4 / C**2)
      ALFA = SQRT(ABS(TEST))
      BETA = C / EM
      GAMA = B / 2.0 / C

      IF (TEST.NE.0.0) CNST1 = (V1 + GAMA) / ALFA

      DO 20, I = 2, NPTS
         IF (TEST.GT.0.0) THEN
            VF = FALFAP(T(I))
         ELSE IF (TEST.EQ.0.0) THEN
            VF = FALFAO(T(I))
         ELSE IF (TEST.LT.0.0) THEN
            VF = FALFAM(T(I))
         END IF
         ERR(I) = (V(I) - VF) * FC
20    CONTINUE

      RETURN
      END
C     ################################################################




C     ******************************************************************
C     FUNCTION TO EVALUATE VELOCITY WHEN ALFA**2 > 0.0
C     ******************************************************************
      FUNCTION FALFAP(T)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /TUFTY3/ V1, ALFA, BETA, GAMA, CNST1

      CNST2 = BETA * ALFA * T
      CNST3 = TAN(CNST2)
```

```
      FALFAP = ALFA * ((CNST1 - CNST3) / (1.0 + CNST1 * CNST3)) - GAMA

      RETURN
      END
C     ##################################################################
```

```
C     ******************************************************************
C     FUNCTION TO EVALUATE VELOCITY WHEN ALFA**2 = 0.0
C     ******************************************************************
      FUNCTION FALFAO(T)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /TUFTY3/ V1, ALFA, BETA, GAMA, CNST1

      FALFAO = 1.0 / ((BETA * T) + (1.0 / (V1 + GAMA))) - GAMA

      RETURN
      END
C     ##################################################################
```

```
C     ******************************************************************
C     FUNCTION TO EVALUATE VELOCITY WHEN ALFA**2 < 0.0
C     ******************************************************************
      FUNCTION FALFAM(T)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /TUFTY3/ V1, ALFA, BETA, GAMA, CNST1

      CNST2 = BETA * ALFA * T
      CNST3 = TANH(CNST2)
      FALFAM = ALFA * ((CNST1 - CNST3) / (1.0 - CNST1 * CNST3)) - GAMA

      RETURN
      END
C     ##################################################################
```

```
C     ******************************************************************
C     SUBROUTINE TO CORRECT COAST-DOWN DATA FOR AMBIENT CONDITIONS
C     ******************************************************************
      SUBROUTINE CORECT(AD,BD,CD,SM,M,SEM,AREA,R,TEMP,PRESS,WINDV,WINDD
     *,WFACT,GRADEP,DIRECT,ADCORF,N,OTIME,CTIME,SPEED)

      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
```

```
      DOUBLE PRECISION M
      CHARACTER *(*)  DIRECT
      DIMENSION Z(9), OTIME(N), CTIME(N), SPEED(N)

      DATA Z(1), Z(2), Z(3), Z(4), Z(5), Z(6), Z(7), Z(8), Z(9)/
     * 0.1651178002D 01, -0.5768188453D 00, -0.4510661009D 00,
     * 0.9379718523D-01, 0.1207860561D 00, -0.2477960124D-01,
     * 0.1212739029D-01, 0.1230558182D-01, -0.1184442607D-01/

C     ****************************************************************
C     CALCULATE DATA FOR CORRECTION METHOD
C     ****************************************************************
      CTIME(1) = OTIME(1)
      GRADE = GRADEP
      IF (DIRECT.EQ.'-') GRADE = -GRADEP
      WNDV = WINDV * WFACT
      HWIND = WNDV * COS(WINDD)
      XWIND = ABS(WNDV * SIN(WINDD))

      DO 20, I = 2, N
         TO = OTIME(I) - OTIME(I-1)
         VO = (SPEED(I-1) + SPEED(I)) / 2.0
         VHR = VO + HWIND
         V1V2 = SPEED(I-1) * SPEED(I)
C        -------------------------------------------------------------

C        *************************************************************
C        CALCULATE YAW FACTOR FOR CD
C        *************************************************************
         IF((VHR.LT.0.000001).AND.(VHR.GT.-0.000001)) PSI = 90.0
         IF((PSI.EQ.90.0).AND.(XWIND.LT.0.001)) PSI = 0.0
         IF((VHR.LT.0.000001).AND.(VHR.GT.-0.000001)) GOTO 10
         PSI = ATAN(XWIND / VHR)
         PSI = PSI * 180.0 / 3.1416
10       XBARZ = ((PSI - (-2.0)) - (92.0 - PSI))
     *           / (92.0 - (-2.0))
         IFAIL = 0
         CALL E02AEF(9, Z, XBARZ, YK, IFAIL)
         YK = YK - 0.0033867702458
C        -------------------------------------------------------------

C        *************************************************************
C        CALCULATE FIRST AND SECOND ORDER DERIVATIVES
C        INCLUDING THE SECOND ORDER CROSS-DERIVATIVES
C        *************************************************************
         PT1 = SPEED(I-1) - SPEED(I)
         PT2 = PT1 * SEM
         PT3 = (SM * 9.81 * AD) + (SM * 9.81 * BD * VO)
     *         + (AREA * CD * V1V2 * 101.3 / (2.0 * R * 293.0))
         PT4 = PT3**2
         PT5 = PT3**3

         PTTM = -(AREA * CD * V1V2 * 101.3
     *          / (2.0 * R * 293.0**2)) - (ADCORF * SM * 9.81 * AD)
         PTPR = (AREA * CD * V1V2 / (2.0 * R * 293.0))
         PTCC = (AREA * CD * V1V2 * 101.3 / (2.0 * R * 293.0))
         PTWN = (AREA * CD * VO * 101.3 / (R * 293.0))
         PTGR = (SM * 9.81)
         PTMA = (9.81 * AD) + (9.81 * BD * VO)
```

```
      T1TM = -(PT2 * PTTM / PT4)
      T1PR = -(PT2 * PTPR / PT4)
      T1CC = -(PT2 * PTCC / PT4)
      T1WN = -(PT2 * PTWN / PT4)
      T1GR = -(PT2 * PTGR / PT4)
      T1MA =  (PT1 / PT3) - (PT2 * PTMA / PT4)

      T2TM2  = (2.0 * PT2 * PTTM**2 / PT5)
     *         - (PT2 * AREA * CD * V1V2 * 101.3
     *         / (PT4 * R * 293.0**3))
      T2TMPR = (2.0 * PT2 * PTTM * PTPR / PT5)
     *         + (PT2 * AREA * CD * V1V2
     *         / (PT4 * 2.0 * R * 293.0**2))
      T2TMCC = (2.0 * PT2 * PTTM * PTCC / PT5)
     *         + (PT2 * AREA * CD *V1V2 * 101.3
     *         / (PT4 * 2.0 * R * 293.0**2))
      T2TMWN = (2.0 * PT2 * PTTM * PTWN / PT5)
     *         - (PT2 * AREA * CD * VO * 101.3
     *         / (PT4 * R * 293.0**2))
      T2TMGR = (2.0 * PT2 * PTTM * PTGR / PT5)
      T2TMMA = (((2.0 * PT2 * PTMA / PT5) - (PT1 / PT4)) * PTTM)
     *         + (PT2 * ADCORF * 9.81 * AD / PT4)
      T2PR2  = (2.0 * PT2 * PTPR**2 / PT5)
      T2PRCC = (2.0 * PT2 * PTPR * PTCC / PT5)
     *         - (PT2 * AREA * CD * V1V2
     *         / (PT4 * 2.0 * R * 293.0))
      T2PRWN = (2.0 * PT2 * PTPR * PTWN / PT5)
     *         - (PT2 * AREA * CD * VO
     *         / (PT4 * R * 293.0))
      T2PRGR = (2.0 * PT2 * PTPR * PTGR / PT5)
      T2PRMA = (((2.0 * PT2 * PTMA / PT5) - (PT1 / PT4)) * PTPR)
      T2CC2  = (2.0 * PT2 * PTCC**2 / PT5)
      T2CCWN = (2.0 * PT2 * PTCC * PTWN / PT5)
     *         - (PT2 * AREA * CD * VO * 101.3
     *         / (PT4 * R * 293.0))
      T2CCGR = (2.0 * PT2 * PTCC * PTGR / PT5)
      T2CCMA = (((2.0 * PT2 * PTMA / PT5) - (PT1 / PT4)) * PTCC)
      T2WN2  = (2.0 * PT2 * PTWN**2 / PT5)
     *         - (PT2 * AREA * CD * 101.3
     *         / (PT4 * R * 293.0))
      T2WNGR = (2.0 * PT2 * PTWN * PTGR / PT5)
      T2WNMA = (((2.0 * PT2 * PTMA / PT5) - (PT1 / PT4)) * PTWN)
      T2GR2  = (2.0 * PT2 * PTGR**2 / PT5)
      T2GRMA = (((2.0 * PT2 * PTMA / PT5) - (PT1 / PT4)) * PTGR)
      T2MA2  = ((2.0 * PT2 * PTMA / PT5) - (2.0 * PT1 / PT4)) * PTMA

      CGTM = TEMP  - 293.0
      CGPR = PRESS - 101.3
      CGCC = YK    - 1.0
      CGWN = HWIND
      CGGR = GRADE
      CGMA = M     - SM
C     ----------------------------------------------------------------

C     ****************************************************************
C     APPLY CORRECTION METHOD
C     ****************************************************************
      TS = TO - (T1TM * CGTM) - (T1PR * CGPR) - (T1CC * CGCC)
```

```
     *              - (T1WN * CGWN) - (T1GR * CGGR) - (T1MA * CGMA)
     *              - (0.5 * T2TM2 * CGTM**2) - (T2TMPR * CGTM * CGPR)
     *              - (T2TMCC * CGTM * CGCC) - (T2TMWN * CGTM * CGWN)
     *              - (T2TMGR * CGTM * CGGR) - (T2TMMA * CGTM * CGMA)
     *              - (0.5 * T2PR2 * CGPR**2) - (T2PRCC * CGPR * CGCC)
     *              - (T2PRWN * CGPR * CGWN) - (T2PRGR * CGPR * CGGR)
     *              - (T2PRMA * CGPR * CGMA) - (0.5 * T2CC2 * CGCC**2)
     *              - (T2CCWN * CGCC * CGWN) - (T2CCGR * CGCC * CGGR)
     *              - (T2CCMA * CGCC * CGMA) - (0.5 * T2WN2 * CGWN**2)
     *              - (T2WNGR * CGWN * CGGR) - (T2WNMA * CGWN * CGMA)
     *              - (0.5 * T2GR2 * CGGR**2) - (T2GRMA * CGGR * CGMA)
     *              - (0.5 * T2MA2 * CGMA**2)

         CTIME(I) = CTIME(I-1) + TS
20    CONTINUE
C     ------------------------------------------------------------------

      RETURN
      END
C     ################################################################
```

Description Of Variable Names Used In CSTDWN

NAMES USED IN MAIN SEGMENT OF CSTDWN

NAME	TYPE OF NAME	MEANING
AD	DOUBLE PRECISION	INITIAL VALUE OF Ad
AD1	DOUBLE PRECISION	EXTRACTED VALUE OF Ad
ADCORF	DOUBLE PRECISION	TEMP CORR FACTOR (Kt)
ANSAD	CHARACTER STRING	(PROGRAM FLOW CONTROL)
ANSBD	CHARACTER STRING	(PROGRAM FLOW CONTROL)
ANSCD	CHARACTER STRING	(PROGRAM FLOW CONTROL)
ANSWER	CHARACTER STRING	GENERAL PURPOSE CHARACTER STRING
AREA	DOUBLE PRECISION	PROJECTED FRONTAL AREA (sq m)
AVGA	DOUBLE PRECISION	GROUP AVERAGE FOR Ad
AVGB	DOUBLE PRECISION	GROUP AVERAGE FOR Bd (s/m)
AVGC	DOUBLE PRECISION	GROUP AVERAGE FOR Cd
BD	DOUBLE PRECISION	INITIAL VALUE OF Bd (s/m)
BD1	DOUBLE PRECISION	EXTRACTED VALUE OF Bd (s/m)
CAL1	DOUBLE PRECISION	UNITS CORRECTION FACTOR FOR AREA
CAL2	DOUBLE PRECISION	UNITS CORRECTION FACTOR FOR RR
CAL3	DOUBLE PRECISION	UNITS CORRECTION FACTOR FOR WI
CAL4	DOUBLE PRECISION	UNITS CORRECTION FACTOR FOR M
CAL5	DOUBLE PRECISION	UNITS CORRECTION FACTOR FOR WINDV
CAL6	DOUBLE PRECISION	UNITS CORRECTION FACTOR FOR PRESS
CD	DOUBLE PRECISION	INITIAL VALUE OF Cd
CD1	DOUBLE PRECISION	EXTRACTED VALUE OF Cd
CHAD	CHARACTER STRING	PROBLEM DESCRIPTION
CHBD	CHARACTER STRING	PROBLEM DESCRIPTION
CHCD	CHARACTER STRING	PROBLEM DESCRIPTION
CHCOR	CHARACTER STRING	PROGRAM FLOW CONTROL
CHDD	CHARACTER STRING	PROBLEM DESCRIPTION
CTIME	DOUBLE PRECISION	CORRECTED TIME (s)
CUERAD	DOUBLE PRECISION	STD ERR IN Ad PER UNIT RMS ERR
CUERBD	DOUBLE PRECISION	STD ERR IN Bd PER UNIT RMS ERR
CUERCD	DOUBLE PRECISION	STD ERR IN Cd PER UNIT RMS ERR
DATE	CHARACTER STRING	TEST DATE
DD	DOUBLE PRECISION	INITIAL VALUE OF Dd (sq (s/m))
DIRECT	CHARACTER STRING	DIRECTION OF TEST
DUM	DOUBLE PRECISION	DUMMY VARIABLE
FILIN	CHARACTER STRING	INPUT FILE NAME
FILOU1	CHARACTER STRING	#1 OUTPUT FILE NAME
FILOU2	CHARACTER STRING	#2 OUTPUT FILE NAME
GRADEP	DOUBLE PRECISION	TRACK GRADIENT FOR +'VE DIR
I	INTEGER	(DO LOOP CONTROL VARIABLE)
I2	INTEGER	(DO LOOP CONTROL VARIABLE)
I3	INTEGER	(DO LOOP CONTROL VARIABLE)
IA	INTEGER	(PROGRAM CONTROL)
IB	INTEGER	(PROGRAM CONTROL)
IC	INTEGER	(PROGRAM CONTROL)
IFAIL1	INTEGER	VALUE RETURNED FROM E04FDF
K	INTEGER	(DO LOOP CONTROL VARIABLE)
M	DOUBLE PRECISION	RECORDED VEHICLE MASS (kg)
N	INTEGER	NO. OF POINTS IN TEST SET
N2	INTEGER	NO. OF TEST SETS IN GROUP
N2A	INTEGER	NO. OF TEST SETS IN GROUP
N3	INTEGER	NUMBER OF GROUPS IN DATA SET
NCHAR	INTEGER	NO. OF LAST NON-BLANK CHARACTER
NCOF	INTEGER	NO. OF VARIABLE COEFFICIENTS

NDATA	NAMED CONSTANT	UNIT NO. OF INPUT DATA FILE
NRES1	NAMED CONSTANT	UNIT NO. OF #1 RESULTS FILE
NRES2	NAMED CONSTANT	UNIT NO. OF #2 RESULTS FILE
NTERM	NAMED CONSTANT	UNIT NO. OF TERMINAL
OTIME	DOUBLE PRECISION	OBSERVED TIME (s)
PRESS	DOUBLE PRECISION	RECORDED AMBIENT PRESSURE (kPa)
R	DOUBLE PRECISION	GAS CONSTANT FOR AIR (kJ/kg.K)
RMSER1	DOUBLE PRECISION	RMS CURVE FITTING ERROR
RR	DOUBLE PRECISION	ROLLING RADIUS OF TYRES (m)
SAVGA	DOUBLE PRECISION	SUM OF AVGA'S
SAVGB	DOUBLE PRECISION	SUM OF AVGB'S (s/m)
SAVGC	DOUBLE PRECISION	SUM OF AVGC'S
SEM	DOUBLE PRECISION	STD EFFECTIVE MASS OF VEHICLE (kg)
SM	DOUBLE PRECISION	STD MASS OF VEHICLE (kg)
SPEED	DOUBLE PRECISION	RECORDED VEHICLE SPEED (m/s)
SPUNIT	CHARACTER STRING	UNITS FOR VEHICLE SPEED
SSAWA	DOUBLE PRECISION	SUM OF (Ad * WGTS) FOR GROUP
SSBWB	DOUBLE PRECISION	SUM OF (Bd * WGTS) FOR GROUP (s/m)
SSCWC	DOUBLE PRECISION	SUM OF (Cd * WGTS) FOR GROUP
SSWA	DOUBLE PRECISION	SUM OF WGTS FOR GROUP
SSWB	DOUBLE PRECISION	SUM OF WGTS FOR GROUP
SSWC	DOUBLE PRECISION	SUM OF WGTS FOR GROUP
STRING	CHARACTER STRING	GENERAL PURPOSE CHARACTER STRING
SW	DOUBLE PRECISION	STANDARD VEHICLE WEIGHT (N)
SWA	DOUBLE PRECISION	STATISTICAL WEIGHT FOR Ad
SWB	DOUBLE PRECISION	STATISTICAL WEIGHT FOR Bd
SWC	DOUBLE PRECISION	STATISTICAL WEIGHT FOR Cd
TAVGA	DOUBLE PRECISION	TOTAL AVG FOR Ad
TAVGB	DOUBLE PRECISION	TOTAL AVG FOR Bd (s/m)
TAVGC	DOUBLE PRECISION	TOTAL AVG FOR Cd
TEMP	DOUBLE PRECISION	RECORDED TEMPERATURE (deg K)
TEUNIT	CHARACTER STRING	RECORDED TEMPERATURE UNITS
TSAVGA	DOUBLE PRECISION	STD ERR IN TOTAL AVG FOR Ad
TSAVGB	DOUBLE PRECISION	STD ERR IN TOTAL AVG FOR Bd (s/m)
TSAVGC	DOUBLE PRECISION	STD ERR IN TOTAL AVG FOR Cd
TSSAWA	DOUBLE PRECISION	SUM OF SSAWA'S
TSSBWB	DOUBLE PRECISION	SUM OF SSBWB'S (s/m)
TSSCWC	DOUBLE PRECISION	SUM OF SSCWC'S
TSSWA	DOUBLE PRECISION	SUM OF SSWA'S
TSSWB	DOUBLE PRECISION	SUM OF SSWB'S
TSSWC	DOUBLE PRECISION	SUM OF SSWC'S
VCONST	DOUBLE PRECISION	UNITS CORRECTION FACTOR FOR SPEED
VEHICL	CHARACTER STRING	VEHICLE DESCRIPTION
W	DOUBLE PRECISION	RECORDED WEIGHT OF VEHICLE (N)
WFACT	DOUBLE PRECISION	WIND CORRECTION FACTOR
WI	DOUBLE PRECISION	TOTAL ROTATING WHEEL INERTIA (kg.sq m)
WINDD	DOUBLE PRECISION	WIND DIR REL TO HEAD ON (deg)
WINDV	DOUBLE PRECISION	WIND VELOCITY (m/s)

NAMES USED IN SUBROUTINE DRGCOF

NAME	TYPE OF NAME	MEANING
A	DOUBLE PRECISION	COEF OF DRAG/SPEED EQN(N)
AC	DOUBLE PRECISION	SCALE FACTOR FOR 'A'
AD1	DOUBLE PRECISION	EXTRACTED VALUE OF Ad
ADS	DOUBLE PRECISION	INITIAL VALUE OF Ad

AREA	DOUBLE PRECISION	PROJECTED FRONTAL AREA (sq m)
AS	DOUBLE PRECISION	INITIAL VALUE OF 'A'
B	DOUBLE PRECISION	COEF OF DRAG/SPEED EQN (N.s/m)
BC	DOUBLE PRECISION	SCALE FACTOR FOR 'B'
BD1	DOUBLE PRECISION	EXTRACTED VALUE OF Bd (s/m)
BDS	DOUBLE PRECISION	INITIAL VALUE OF Bd (s/m)
BS	DOUBLE PRECISION	INITIAL VALUE OF 'B'
C	DOUBLE PRECISION	COEFF OF DRAG/SPEED EQN (N.sq(s/m))
CC	DOUBLE PRECISION	SCALE FACTOR FOR 'C'
CD1	DOUBLE PRECISION	EXTRACTED VALUE OF Cd
CDS	DOUBLE PRECISION	INITIAL VALUE OF Cd
COF	DOUBLE PRECISION	COEFFICIENTS OF DRG/SPEED EQN
COFZ	DOUBLE PRECISION	COEFFICIENTS O DRG/SPEED EQN
CS	DOUBLE PRECISION	INITIAL VALUE OF 'C'
DDS	DOUBLE PRECISION	INITIAL VALUE OF Dd (sq(s/m))
EM	DOUBLE PRECISION	EFFECTIVE MASS OF VEHICLE (kg)
EMZ	DOUBLE PRECISION	EFFECTIVE MASS OF VEHICLE (kg)
FC	DOUBLE PRECISION	SCALE FACTOR FOR CURVE FITTING ERROR
FX	DOUBLE PRECISION	SUM OF SQUARES CURV FIT ERR
I	INTEGER	(DO LOOP CONTROL VARIABLE)
IA	INTEGER	(PROGRAM CONTROL)
IAZ	INTEGER	(PROGRAM CONTROL)
IB	INTEGER	(PROGRAM CONTROL)
IBZ	INTEGER	(PROGRAM CONTROL)
IC	INTEGER	(PROGRAM CONTROL)
ICZ	INTEGER	(PROGRAM CONTROL)
IFAIL	INTEGER	ERROR CODE INDICATOR FOR E04FDF
IFAIL1	INTEGER	ERROR CODE INDICATOR FOR E04FDF
ISUM	INTEGER	NUM OF TIMES E04FDF CALLED
IW	INTEGER	WORK SPACE ARRAY
NCOF	INTEGER	NUMBER OF VARIABLE COEFFICIENTS
NCOFZ	INTEGER	NUMBER OF VARIABLE COEFFICIENTS
NPTS	INTEGER	NUM OF PTS IN TEST SET
PRESS	DOUBLE PRECISION	PRESSURE (kPa)
R	DOUBLE PRECISION	GAS CONSTANT FOR AIR (kJ/kg.K)
RHO	DOUBLE PRECISION	AIR DENSITY (kg/(cu m))
RMSER1	DOUBLE PRECISION	RMS CURVE FITTING ERROR
SPEED	DOUBLE PRECISION	VEHICLE SPEED (m/s)
T	DOUBLE PRECISION	ELAPSED TIME (s)
TEMP	DOUBLE PRECISION	TEMPERATURE (deg K)
TIME	DOUBLE PRECISION	ELAPSED TIME (s)
V	DOUBLE PRECISION	VEHICLE SPEED (m/s)
W	DOUBLE PRECISION	VEHICLE WEIGHT (N)
WORK	DOUBLE PRECISION	WORK SPACE ARRAY

NAMES USED IN SUBROUTINE LSFUN1

NAME	TYPE OF NAME	MEANING
A	DOUBLE PRECISION	COEF OF DRG/SPEED EQN (N)
AC	DOUBLE PRECISION	SCALE FACTOR FOR 'A'
ALFA	DOUBLE PRECISION	COEF OF ANALYTICAL CST-DWN FUNCT
B	DOUBLE PRECISION	COEF OF DRG/SPEED EQN (N.s/m)
BC	DOUBLE PRECISION	SCALE FACTOR FOR 'B'
BETA	DOUBLE PRECISION	COEF OF ANALYTICAL CST-DWN FUNCT
C	DOUBLE PRECISION	COEF OF DRG/SPEED EQN (N.sq(s/m)
CC	DOUBLE PRECISION	SCALE FACTOR FOR 'C'

CNST1	DOUBLE PRECISION	CONST FOR TEST SET
COF	DOUBLE PRECISION	COEFFICIENTS OF DRG/SPEED EQN
COFZ	DOUBLE PRECISION	COEFFICIENTS OF DRG/SPEED EQN
EM	DOUBLE PRECISION	EFFECTIVE MASS OF VEHICLE (kg)
GAMA	DOUBLE PRECISION	COEF OF ANALYTICAL CST-DWN FUNCT
I	INTEGER	(DO LOOP CONTROL VARIABLE)
IA	INTEGER	(PROGRAM CONTROL)
IB	INTEGER	(PROGRAM CONTROL)
IC	INTEGER	(PROGRAM CONTROL)
NPTS	INTEGER	NUM PTS IN TEST SET
NV	INTEGER	NUM OF VARIABLE COEFS
ERR	DOUBLE PRECISION	CURVE FITTING ERROR
T	DOUBLE PRECISION	ELAPSED TIME (s)
TEST	DOUBLE PRECISION	(PROGRAM CONTROL VARIABLE)
V	DOUBLE PRECISION	ANALYTICAL VEHICLE SPEED (m/s)
V1	DOUBLE PRECISION	INITIAL VEHICLE SPEED (m/s)
VF	DOUBLE PRECISION	THEORETICAL VEHICLE SPEED (m/s)

NAMES USED IN SUBROUTINE FALFAP

NAME	TYPE OF NAME	MEANING
ALFA	DOUBLE PRECISION	COEF OF ANALYTICAL CST-DWN FUNCT
BETA	DOUBLE PRECISION	COEF OF ANALYTICAL CST-DWN FUNCT
CNST1	DOUBLE PRECISION	CONST
CNST2	DOUBLE PRECISION	CONST
CNST3	DOUBLE PRECISION	CONST
GAMA	DOUBLE PRECISION	COEF OF ANALYTICAL CST-DWN FUNCT
T	DOUBLE PRECISION	ELAPSED TIME (s)

NAMES USED IN SUBROUTINE FALFAO

NAME	TYPE OF NAME	MEANING
BETA	DOUBLE PRECISION	COEF OF ANALYTICAL CST-DWN FUNCT
GAMA	DOUBLE PRECISION	COEF OF ANALYTICAL CST-DWN FUNCT
T	DOUBLE PRECISION	ELAPSED TIME (s)
V1	DOUBLE PRECISION	INITIAL VEHICLE SPEED (m/s)

NAMES USED IN SUBROUTINE FALFAM

NAME	TYPE OF NAME	MEANING
ALFA	DOUBLE PRECISION	COEF OF ANALYTICAL CST-DWN FUNCT
BETA	DOUBLE PRECISION	COEF OF ANALYTICAL CST-DWN FUNCT
CNST1	DOUBLE PRECISION	CONST
CNST2	DOUBLE PRECISION	CONST
CNST3	DOUBLE PRECISION	CONST
GAMA	DOUBLE PRECISION	COEF OF ANALYTICAL CST-DWN FUNCT
T	DOUBLE PRECISION	ELAPSED TIME (s)

NAMES USED IN SUBROUTINE CORECT

NAME	TYPE OF NAME	MEANING
AD	DOUBLE PRECISION	INITIAL VALUE OF Ad
ADCORF	DOUBLE PRECISION	TEMP CORR FACT FOR RR
AREA	DOUBLE PRECISION	PROJECTED FRONTAL AREA (sq.m)
BD	DOUBLE PRECISION	INITIAL VALUE OF Bd (s/m)
CD	DOUBLE PRECISION	INITIAL VALUE OF Cd
CGCC	DOUBLE PRECISION	CHANGE IN YK
CGGR	DOUBLE PRECISION	CHANGE IN TRACK GRADIENT
CGMA	DOUBLE PRECISION	CHANGE IN VEHICLE MASS (kg)
CGPR	DOUBLE PRECISION	CHANGE IN PRESSURE (kPa)
CGTM	DOUBLE PRECISION	CHANGE IN TEMPERATURE (deg K)
CGWN	DOUBLE PRECISION	CHANGE IN HEAD WIBD SPEED (m/s)
CTIME	DOUBLE PRECISION	CORRECTED ELAPSED TIME (s)
DIRECT	CHARACTER STRING	TEST DIRECTION
GRADE	DOUBLE PRECISION	TRACK GRADIENT REL TO VEH DIR
GRADEP	DOUBLE PRECISION	TRACK GRADIENT FOR +'VE DIR
HWIND	DOUBLE PRECISION	HEAD WIND SPEED (m/s)
I	INTEGER	(DO LOOP CONTROL VARIABLE)
IFAIL	INTEGER	ERROR INDICATOR FOR E02AEF
M	DOUBLE PRECISION	RECORDED VEHICLE MASS (kg)
N	INTEGER	NUM OF PTS IN TEST DATA
OTIME	DOUBLE PRECISION	RECORDED ELAPSED TIME (s)
PRESS	DOUBLE PRECISION	PRESSURE (kPa)
PSI	DOUBLE PRECISION	AERODYNAMIC YAW ANGLE
PT1	DOUBLE PRECISION	CONSTANT
PT2	DOUBLE PRECISION	CONSTANT
PT3	DOUBLE PRECISION	CONSTANT
PT4	DOUBLE PRECISION	CONSTANT
PT5	DOUBLE PRECISION	CONSTANT
PTCC	DOUBLE PRECISION	CONSTANT
PTGR	DOUBLE PRECISION	CONSTANT
PTMA	DOUBLE PRECISION	CONSTANT
PTPR	DOUBLE PRECISION	CONSTANT
PTTM	DOUBLE PRECISION	CONSTANT
PTWN	DOUBLE PRECISION	CONSTANT
R	DOUBLE PRECISION	GAS CONSTANT FOR AIR (kJ/kg.K)
SEM	DOUBLE PRECISION	STANDARD VEHICLE EFFECTIVE MASS (kg)
SM	DOUBLE PRECISION	STANDARD VEHICLE MASS (kg)
SPEED	DOUBLE PRECISION	VEHICLE SPEED (m/s)
TO	DOUBLE PRECISION	RECORDED ELAPSED TIME (s)
T1CC	DOUBLE PRECISION	FIRST ORDER DERIVATIVE
T1GR	DOUBLE PRECISION	FIRST ORDER DERIVATIVE
T1MA	DOUBLE PRECISION	FIRST ORDER DERIVATIVE
T1PR	DOUBLE PRECISION	FIRST ORDER DERIVATIVE
T1TM	DOUBLE PRECISION	FIRST ORDER DERIVATIVE
T1WN	DOUBLE PRECISION	FIRST ORDER DERIVATIVE
T2CC2	DOUBLE PRECISION	SECOND ORDER DERIVATIVE
T2CCGR	DOUBLE PRECISION	SECOND ORDER CROSS DERIVATIVE
T2CCMA	DOUBLE PRECISION	SECOND ORDER CROSS DERIVATIVE
T2CCWN	DOUBLE PRECISION	SECOND ORDER CROSS DERIVATIVE
T2GR2	DOUBLE PRECISION	SECOND ORDER DERIVATIVE
T2GRMA	DOUBLE PRECISION	SECOND ORDER CROSS DERIVATIVE
T2MA2	DOUBLE PRECISION	SECOND ORDER DERIVATIVE
T2PR2	DOUBLE PRECISION	SECOND ORDER DERIVATIVE
T2PRCC	DOUBLE PRECISION	SECOND ORDER CROSS DERIVATIVE
T2PRGR	DOUBLE PRECISION	SECOND ORDER CROSS DERIVATIVE

T2PRMA	DOUBLE PRECISION	SECOND ORDER CROSS DERIVATIVE
T2PRWN	DOUBLE PRECISION	SECOND ORDER CROSS DERIVATIVE
T2TM2	DOUBLE PRECISION	SECOND ORDER DERIVATIVE
T2TMCC	DOUBLE PRECISION	SECOND ORDER CROSS DERIVATIVE
T2TMGR	DOUBLE PRECISION	SECOND ORDER CROSS DERIVATIVE
T2TMMA	DOUBLE PRECISION	SECOND ORDER CROSS DERIVATIVE
T2TMPR	DOUBLE PRECISION	SECOND ORDER CROSS DERIVATIVE
T2TMWN	DOUBLE PRECISION	SECOND ORDER CROSS DERIVATIVE
T2WN2	DOUBLE PRECISION	SECOND ORDER DERIVATIVE
T2WNGR	DOUBLE PRECISION	SECOND ORDER CROSS DERIVATIVE
T2WNMA	DOUBLE PRECISION	SECOND ORDER CROSS DERIVATIVE
TEMP	DOUBLE PRECISION	TEMPERATURE (deg K)
TS	DOUBLE PRECISION	STANDARD ELAPSED TIME (s)
VO	DOUBLE PRECISION	AVG VEHICLE SPEED (m/s)
V1V2	DOUBLE PRECISION	INIT VEL TIMES FIN VEL (sq(m/s))
VHR	DOUBLE PRECISION	RELATIVE HEAD WIND SPEED (m/s)
WFACT	DOUBLE PRECISION	WIND SPEED CORR FACT
WINDD	DOUBLE PRECISION	WIND DIR REL TO HEAD-ON (deg)
WINDV	DOUBLE PRECISION	RECORDED WIND SPEED (m/s)
WNDV	DOUBLE PRECISION	CORRECTED WIND SPEED (m/s)
XBARZ	DOUBLE PRECISION	NORMALIZED YAW ANGLE
XWIND	DOUBLE PRECISION	CROSS-WIND SPEED (m/s)
YK	DOUBLE PRECISION	YAW ANG COR FACT Ky
Z	DOUBLE PRECISION	CHEBYCHEF COEFS FOR YK

Description Of Input Data File Format For CSTDWN

```
LINE 01:  Vehicle/Test Title (up to 70 characters long)
LINE 02:  AREA  CAL1 (AREA*CAL1 = square metres) (free format)
LINE 03:  RR    CAL2 (RR*CAL2 = metres) (free format)
LINE 04:  WI    CAL3 (WI*CAL3 = kilogram metres squared) (free format)
LINE 05:  Vehicle Speed Units ('KM/H','M/S','MPH') (start col 1)
LINE 06:  Number Of Test Groups (free format integer)
LINE 07:  Number Of Test Sets In Group (free format integer)
LINE 08:  Number Of Data Points In Test Set (free format integer)
LINE 09:  Date Of Test (8 characters start col 1)
LINE 10:  Direction Of Test ('+','-') (col 1)
LINE 11:  M     CAL4 (M*CAL4 = kilogram) (free format)
LINE 12:  WINDV CAL5 (WINDV*CAL5 = metres per second) (free format)
LINE 13:  WINDD      (degrees rel to head on) (free format)
LINE 14:  PRESS CAL6 (PRESS*CAL6 = kilo-Pascals) (free format)
LINE 15:  TEMP       (in Centigrade of Fahrenheit) (free format)
LINE 16:  Temperature Units ('C','F') (col 1)
LINE 17:  OTIME SPEED (seconds and speed units) (free format)
  REPEAT (17) FOR ALL DATA POINTS IN TEST SET
  REPEAT FROM (08) FOR ALL TEST SETS IN GROUP
  REPEAT FROM (07) FOR ALL TEST GROUPS
```

Listing Of Supplementary Subroutines For CSTDWN

```
C       ---------------------------------
C       SUBROUTINE TO PRINT OUT STRING AT
C       TERMINAL WITH CARRIAGE RETURN
C       ---------------------------------
        SUBROUTINE COU(A)
        CHARACTER*(*) A

        WRITE (UNIT=*, FMT='(A)') A

        RETURN
        END
```

```
C       ---------------------------------
C       SUBROUTINE TO PRINT OUT STRING AT
C       TERMINAL WITHOUT CARRIAGE RETURN
C       ---------------------------------
        SUBROUTINE COUA(A)
        CHARACTER*(*) A

        WRITE (UNIT=*, FMT='(A,$)') A

        RETURN
        END
```

```
C       ------------------------------------
C       FUNCTION TO RETURN LENGTH OF STRING
C       MINUS TRAILING BLANKS
C       ------------------------------------
        INTEGER FUNCTION LENG(A)
        INTRINSIC LEN
        CHARACTER*(*) A
        INTEGER I, L

        L = LEN(A)
        LENG = 0

        DO 10, I = L, 1, -1
           IF (A(I:I).NE.' ') THEN
              LENG = I
              RETURN
           END IF
   10   CONTINUE

        RETURN
        END
```

```
C     ---------------------------------
C     SUBROUTINE TO LEFT JUSTIFY STRING
C     AND RETURN N=0 IF STRING IS BLANK
C     ---------------------------------
      SUBROUTINE LFTJST(A, N)
      EXTERNAL NONBLK, TRIM
      CHARACTER*(*) A
      INTEGER N

      N = NONBLK(A)
      CALL TRIM(A)

      RETURN
      END
```

```
C     ------------------------------------
C     FUNCTION TO RETURN POSITION OF FIRST
C     NON-BLANK CHARACTER IN STRING.  IF
C     STRING IS BLANK THEN 0 IS RETURNED
C     ------------------------------------
      INTEGER FUNCTION NONBLK(A)
      INTRINSIC LEN
      CHARACTER*(*) A
      INTEGER I, L

      L = LEN(A)
      NONBLK = 0

      DO 10, I = 1, L
         IF (A(I:I).NE.' ') THEN
            NONBLK = I
            RETURN
         END IF
10    CONTINUE

      RETURN
      END
```

```
C     ------------------------------------
C     SUBROUTINE TO PRINT OUT STRING MINUS
C     TRAILING BLANKS AT TERMINAL WITH
C     CARRIAGE RETURN
C     ------------------------------------
      SUBROUTINE SOU(A)
      EXTERNAL LENG
      CHARACTER*(*) A
      INTEGER LG

      LG = LENG(A)

      IF (LG.EQ.0) THEN
         WRITE (UNIT=*, FMT=*)
      ELSE
         WRITE (UNIT=*, FMT='(A)') A(1:LG)
      END IF
```

```
      RETURN
      END
```

```
C     ------------------------------------
C     SUBROUTINE TO PRINT OUT STRING MINUS
C     TRAILING BLANKS AT TERMINAL WITHOUT
C     CARRIAGE RETURN
C     ------------------------------------
      SUBROUTINE SOUA(A)
      EXTERNAL LENG
      CHARACTER*(*) A
      INTEGER LG

      LG = LENG(A)

      IF (LG.EQ.0) THEN
         RETURN
      ELSE
         WRITE (UNIT=*, FMT='(A,$)') A(1:LG)
      END IF

      RETURN
      END
```

```
C     ---------------------------------------
C     SUBROUTINE TO TRIM LEADING SPACES FROM
C     CHARACTER STRING (ie LEFT JUSTIFY)
C     ---------------------------------------
      SUBROUTINE TRIM(A)
      INTRINSIC LEN
      EXTERNAL NONBLK
      CHARACTER*(*) A
      INTEGER N

      N = NONBLK(A)
      L = LEN(A)

      IF (N.NE.0) THEN
         A = A(N:L)
      END IF

      RETURN
      END
```

```
C     --------------------------------------
C     SUBROUTINE TO UPCASE CHARACTER STRING
C     --------------------------------------
      SUBROUTINE UPCASE(A)
      INTRINSIC ICHAR, CHAR
      EXTERNAL LENG, NONBLK
      CHARACTER*(*) A
      INTEGER LG, I, N
```

```
      LG = LENG(A)
      IF (LG.EQ.0) RETURN

      DO 10, I = 1, LG
         N = ICHAR(A(I:I))
         IF ((N.GE.97).AND.(N.LE.122)) THEN
            N = N - 32
            A(I:I) = CHAR(N)
         END IF
10    CONTINUE

      RETURN
      END
```

APPENDIX D

LISTING OF FORTRAN PROGRAM SCALE

```
C      AAA      L         EEEEE M    M TTTTT  AAA    GGG   EEEEE
C     A   A     L         E     MM MM    T    A   A G    G E
C     AAAAA     L         EEEE  M M M    T    AAAAA G  ___ EEEE
C     A   A     L         E     M    M   T    A   A G    G E
C     A   A *   LLLLL *   EEEEE M    M   T    A   A  GGG   EEEEE

C     ******************************************************************
C     PROGRAM DECLARATIONS
C     ******************************************************************
      PROGRAM SCALE
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DOUBLE PRECISION M

      PARAMETER (NTERM=0, NDATA=5, NRES1=6, NRES2=7)

      CHARACTER *70  FILIN, FILOU1, VEHICL, ANSWER, ANSAD, ANSBD, ANSCD
     *               , FILOU2, CHAD, CHBD, CHCD, CHDD
      CHARACTER *8  DATE(500), STRING, CHCOR
      CHARACTER *5  SPUNIT, TEUNIT
      CHARACTER *1  DIRECT(500)

      COMMON /BLCK1/ OTIME(500), SPEED(500), M(500), WINDV(500)
     *, WINDD(500), PRESS(500), TEMP(500), CTIME(500), NTESTC(500)
     *, AD1(500), BD1(500), CD1(500), RMSER1(500), IFAIL1(500), N2A(500)
     *, SSWA(500), SSWB(500), SSWC(500), SSAWA(500), SSBWB(500)
     *, SSCWC(500), AVGA(500), AVGB(500), AVGC(500), SAVGA(500)
     *, SAVGB(500), SAVGC(500), SWA(500), SWB(500), SWC(500)

      R = 0.288
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT VALUES OF COEFFICIENTS FOR MINIMUM RMS
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COU('Please enter values for the vehicle coefficients')
      CALL COU('at minimum RMS.')

      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Ad =  ')
      READ (UNIT=NTERM, FMT=*) AD

      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Bd =  ')
      READ (UNIT=NTERM, FMT=*) BD

      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Cd =  ')
      READ (UNIT=NTERM, FMT=*) CD
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT TRACK GRADIENT
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter track gradient for +''ve direction.  ')
      READ (UNIT=NTERM, FMT=*) GRADEP
C     ------------------------------------------------------------------
```

```
C     ****************************************************************
C     INPUT WIND SPEED FACTOR
C     ****************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter factor for wind speed. ')
      READ(UNIT=NTERM, FMT=*) WFACT
C     ----------------------------------------------------------------

C     ****************************************************************
C     INPUT STANDARD VEHICLE MASS
C     ****************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter standard vehicle mass (kg).  ')
      READ (UNIT=NTERM, FMT=*) SM
      SW = SM * 9.81
C     ----------------------------------------------------------------

C     ****************************************************************
C     INPUT TEMPERATURE CORRECTION FACTOR FOR Ad
C     ****************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter temperature correction factor for Ad.  ')
      READ (UNIT=NTERM, FMT=*) ADCORF
C     ----------------------------------------------------------------

C     ****************************************************************
C     CHOOSE WHETHER OR NOT TO CORRECT FOR AMBIENT CONDITIONS
C     ****************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Correct for ambient conditions?  ')
      READ (UNIT=NTERM, FMT='(A)') CHCOR
      CALL UPCASE(CHCOR)
      CALL LFTJST(CHCOR,NCHAR)
      IF (CHCOR.EQ.'Y'.OR.CHCOR.EQ.'YES'.OR.NCHAR.EQ.0) CHCOR = 'Y'
C     ----------------------------------------------------------------

C     ****************************************************************
C     OPEN DATA FILE
C     ****************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter name of data file. ')
      READ (UNIT=NTERM, FMT='(A)') FILIN
      OPEN (UNIT=NDATA, FILE=FILIN, FORM='FORMATTED', CARRIAGE=.FALSE.)
C     ----------------------------------------------------------------

C     ****************************************************************
C     INPUT VEHICLE NAME
C     ****************************************************************
      READ (UNIT=NDATA, FMT='(A)') VEHICL
C     ----------------------------------------------------------------

C     ****************************************************************
C     INPUT VEHICLE CONSTANTS
C     ****************************************************************
      READ (UNIT=NDATA, FMT=*) AREA, CAL1
      READ (UNIT=NDATA, FMT=*) RR,   CAL2
      READ (UNIT=NDATA, FMT=*) WI,   CAL3
```

```
      AREA = AREA * CAL1
      RR = RR * CAL2
      WI = WI * CAL3
      SEM = SM + WI / RR**2
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT UNITS FOR VEHICLE SPEED AND SET SPEED FACTOR
C     ******************************************************************
      READ (UNIT=NDATA, FMT='(A)') SPUNIT
      CALL UPCASE(SPUNIT)
      IF(SPUNIT.EQ.'KM/H') VCONST = 1.0 / 3.6
      IF(SPUNIT.EQ.'M/S' ) VCONST = 1.0
      IF(SPUNIT.EQ.'MPH' ) VCONST = 1.609344 / 3.6
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT NUMBER OF TEST GROUPS BUT RESET TO 1
C     ******************************************************************
      READ (UNIT=NDATA, FMT=*) N3
      N3 = 1
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT NUMBER OF TEST SETS IN GROUP BUT RESET TO 1
C     ******************************************************************
      K = 0

      DO 60, I3 = 1, N3
         READ (UNIT=NDATA, FMT=*) N2
         N2 = 1
         N2A(I3) = N2
C        ---------------------------------------------------------------

C        ***************************************************************
C        INPUT NUMBER OF DATA POINTS IN TEST SET
C        ***************************************************************
         DO 50 I2 = 1, N2
            K = K + 1
            READ (UNIT=NDATA, FMT=*) N
C           ------------------------------------------------------------

C           ************************************************************
C           INPUT DATE OF TEST
C           ************************************************************
            READ (UNIT=NDATA, FMT='(A)') DATE(K)
C           ------------------------------------------------------------

C           ************************************************************
C           INPUT AMBIENT CONDITIONS
C           ************************************************************
            READ (UNIT=NDATA, FMT='(A)') DIRECT(K)
            READ (UNIT=NDATA, FMT=*)     M(K),      CAL4
            READ (UNIT=NDATA, FMT=*)     WINDV(K), CAL5
            READ (UNIT=NDATA, FMT=*)     WINDD(K)
            READ (UNIT=NDATA, FMT=*)     PRESS(K), CAL6
            READ (UNIT=NDATA, FMT=*)     TEMP(K)
            READ (UNIT=NDATA, FMT='(A)') TEUNIT
```

```
      M(K) = M(K) * CAL4
      DM = SM - M(K)
      W = M(K) * 9.81
      EM = M(K) + WI / RR**2
      WINDV(K) = WINDV(K) * CAL5
      PRESS(K) = PRESS(K) * CAL6
      WINDD(K) = WINDD(K) * 0.01745

      CALL UPCASE(TEUNIT)
      IF(TEUNIT.EQ.'F') THEN
         TEMP(K) = (TEMP(K) - 32.0) / 1.8 + 273.0
      ELSE IF(TEUNIT.EQ.'C') THEN
         TEMP(K) = TEMP(K) + 273.0
      END IF
C     ------------------------------------------------------------

C     ************************************************************
C     INPUT TEST DATA
C     ************************************************************
      DO 40 I=1, N
         READ (UNIT=NDATA, FMT=*) OTIME(I), SPEED(I)
         SPEED(I) = SPEED(I) * VCONST
40    CONTINUE
C     ------------------------------------------------------------

C     ************************************************************
C     CORRECT FOR AMBIENT CONDITIONS
C     ************************************************************
      IF (CHCOR.EQ.'Y') THEN
         CALL CORECT(AD,BD,CD,SM,M(K),SEM,AREA,R,TEMP(K),PRESS(K)
     *   ,WINDV(K),WINDD(K),WFACT,GRADEP,DIRECT(K),ADCORF,N,OTIME
     *   ,CTIME,SPEED)
      ELSE
         DO 45, I = 1, N
            CTIME(I) = OTIME(I)
45       CONTINUE
      END IF
C     ------------------------------------------------------------

C     ************************************************************
C     ENTER AND TEST SCALE FACTORS
C     ************************************************************
10    WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter scale factor for ''a''.  ')
      READ (UNIT=NTERM, FMT=*) AC

      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter scale factor for ''b''.  ')
      READ (UNIT=NTERM, FMT=*) BC

      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter scale factor for ''c''.  ')
      READ (UNIT=NTERM, FMT=*) CC

      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter scale factor for SMSQER.  ')
      READ (UNIT=NTERM, FMT=*) FC

      CALL SCL(CTIME,SPEED,N,SW,AREA,SEM,AD,BD,CD,AC,BC,CC,FC)
```

```
        WRITE (UNIT=NTERM, FMT=*)
        CALL COUA('Do you wish to try other scale factors?  ')
        READ (UNIT=NTERM, FMT='(A)') ANSWER
        CALL UPCASE(ANSWER)
        CALL LFTJST(ANSWER,NCHAR)
        IF (ANSWER.EQ.'Y'.OR.ANSWER.EQ.'YES'.OR.NCHAR.EQ.0) GOTO 10
50      CONTINUE
60    CONTINUE
C     ------------------------------------------------------------------

C     ******************************************************************
C     CLOSE DATA FILE
C     ******************************************************************
      CLOSE (UNIT=NDATA)
C     ------------------------------------------------------------------

C     ******************************************************************
C     TERMINATE PROGRAM
C     ******************************************************************
      STOP
      END
C     ##################################################################




C     ******************************************************************
C     SUBROUTINE TO DETERMINE THE SCALE FACTORS
C     ******************************************************************
      SUBROUTINE SCL(TIME,SPEED,NPTS,W,AREA,EMZ,ADS,BDS,CDS,AC,BC,CC,FC)

      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /TUFTY1/ T(500), V(500), COFZ(3)
      DIMENSION TIME(NPTS), SPEED(NPTS)
      DIMENSION COF(3), IW(5), WORK(5000)

      NCOF = 3
      EM = EMZ
      R = 0.288
      PRESS = 101.3
      TEMP = 293.0
      RHO = PRESS / R / TEMP

      AS = ADS * W
      BS = BDS * W
      CS = (CDS * 0.5 * RHO * AREA)

      COF(1) = AS * AC
      COF(2) = BS * BC
      COF(3) = CS * CC

      DO 10, I = 1, NPTS
         T(I) = TIME(I)
         V(I) = SPEED(I)
10    CONTINUE
```

```
      CALL LSFUN2(NPTS,NCOF,COF,SSQER1,EM,AC,BC,CC,FC)

      COFZ(1) = COF(1) + 1.0
      COFZ(2) = COF(2)
      COFZ(3) = COF(3)
      CALL LSFUN2(NPTS,NCOF,COFZ,SSQER,EM,AC,BC,CC,FC)
      SSQERA = SSQER - SSQER1

      COFZ(1) = COF(1)
      COFZ(2) = COF(2) + 1.0
      COFZ(3) = COF(3)
      CALL LSFUN2(NPTS,NCOF,COFZ,SSQER,EM,AC,BC,CC,FC)
      SSQERB = SSQER - SSQER1

      COFZ(1) = COF(1)
      COFZ(2) = COF(2)
      COFZ(3) = COF(3) + 1.0
      CALL LSFUN2(NPTS,NCOF,COFZ,SSQER,EM,AC,BC,CC,FC)
      SSQERC = SSQER - SSQER1

      PRINT *
      PRINT *, 'AC =', AC
      PRINT *, 'BC =', BC
      PRINT *, 'CC =', CC
      PRINT *, 'FC =', FC
      PRINT *
      PRINT *, 'SMSQER =', SSQER1
      PRINT *
      PRINT *, 'Change in SMSQER for unit change in ''a'' =', SSQERA
      PRINT *, 'Change in SMSQER for unit change in ''b'' =', SSQERB
      PRINT *, 'Change in SMSQER for unit change in ''c'' =', SSQERC

      RETURN
      END
C     ####################################################################




C     ********************************************************************
C     FUNCTION TO BE MINIMIZED
C     ********************************************************************
      SUBROUTINE LSFUN2(NPTS,NV,COF,SMSQER,EM,AC,BC,CC,FC)

      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION COF(NV)
      COMMON /TUFTY1/ T(500), V(500), COFZ(3)
      COMMON /TUFTY3/ V1, ALFA, BETA, GAMA, CNST1

      V1 = V(1)
      ERR = 0.0
      SMSQER = 0.0

      A = COF(1) / AC
      B = COF(2) / BC
```

```
      C = COF(3) / CC

      TEST = (A / C) - (B**2 / 4 / C**2)
      ALFA = SQRT(ABS(TEST))
      BETA = C / EM
      GAMA = B / 2.0 / C

      IF (TEST.NE.0.0) CNST1 = (V1 + GAMA) / ALFA

      DO 20, I = 2, NPTS
         IF (TEST.GT.0.0) THEN
            VF = FALFAP(T(I))
         ELSE IF (TEST.EQ.0.0) THEN
            VF = FALFAO(T(I))
         ELSE IF (TEST.LT.0.0) THEN
            VF = FALFAM(T(I))
         END IF
         ERR = (V(I) - VF) * FC
         SMSQER = SMSQER + ERR**2
20    CONTINUE

      RETURN
      END
C     ###################################################################




C     *******************************************************************
C     FUNCTION TO EVALUATE VELOCITY WHEN ALFA**2 > 0.0
C     *******************************************************************
      FUNCTION FALFAP(T)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /TUFTY3/ V1, ALFA, BETA, GAMA, CNST1

      CNST2 = BETA * ALFA * T
      CNST3 = TAN(CNST2)
      FALFAP = ALFA * ((CNST1 - CNST3) / (1.0 + CNST1 * CNST3)) - GAMA

      RETURN
      END
C     ###################################################################




C     *******************************************************************
C     FUNCTION TO EVALUATE VELOCITY WHEN ALFA**2 = 0.0
C     *******************************************************************
      FUNCTION FALFAO(T)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /TUFTY3/ V1, ALFA, BETA, GAMA, CNST1
```

```
      FALFAO = 1.0 / ((BETA * T) + (1.0 / (V1 + GAMA))) - GAMA

      RETURN
      END
C     ##################################################################




C     ******************************************************************
C     FUNCTION TO EVALUATE VELOCITY WHEN ALFA**2 < 0.0
C     ******************************************************************
      FUNCTION FALFAM(T)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /TUFTY3/ V1, ALFA, BETA, GAMA, CNST1

      CNST2 = BETA * ALFA * T
      CNST3 = TANH(CNST2)
      FALFAM = ALFA * ((CNST1 - CNST3) / (1.0 - CNST1 * CNST3)) - GAMA

      RETURN
      END
C     ##################################################################




C     ******************************************************************
C     SUBROUTINE TO CORRECT COAST-DOWN DATA FOR AMBIENT CONDITIONS
C     ******************************************************************
      SUBROUTINE CORECT(AD,BD,CD,SM,M,SEM,AREA,R,TEMP,PRESS,WINDV,WINDD
     *,WFACT,GRADEP,DIRECT,ADCORF,N,OTIME,CTIME,SPEED)

      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DOUBLE PRECISION M
      CHARACTER *(*)  DIRECT
      DIMENSION Z(9), OTIME(N), CTIME(N), SPEED(N)

      DATA Z(1), Z(2), Z(3), Z(4), Z(5), Z(6), Z(7), Z(8), Z(9)/
     * 0.1651178002D 01, -0.5768188453D 00, -0.4510661009D 00,
     * 0.9379718523D-01, 0.1207860561D 00, -0.2477960124D-01,
     * 0.1212739029D-01, 0.1230558182D-01, -0.1184442607D-01/

C     ******************************************************************
C     CALCULATE DATA FOR CORRECTION METHOD
C     ******************************************************************
      CTIME(1) = OTIME(1)
      GRADE = GRADEP
      IF (DIRECT.EQ.'-') GRADE = -GRADEP
      WNDV = WINDV * WFACT
      HWIND = WNDV * COS(WINDD)
      XWIND = ABS(WNDV * SIN(WINDD))
```

```
      DO 20, I = 2, N
         TO = OTIME(I) - OTIME(I-1)
         VO = (SPEED(I-1) + SPEED(I)) / 2.0
         VHR = VO + HWIND
         V1V2 = SPEED(I-1) * SPEED(I)
C        ---------------------------------------------------------------

C        ***************************************************************
C        CALCULATE YAW FACTOR FOR CD
C        ***************************************************************
         IF((VHR.LT.0.000001).AND.(VHR.GT.-0.000001)) PSI = 90.0
         IF((PSI.EQ.90.0).AND.(XWIND.LT.0.001)) PSI = 0.0
         IF((VHR.LT.0.000001).AND.(VHR.GT.-0.000001)) GOTO 10
         PSI = ATAN(XWIND / VHR)
         PSI = PSI * 180.0 / 3.1416
10       XBARZ = ((PSI - (-2.0)) - (92.0 - PSI))
     *         / (92.0 - (-2.0))
         IFAIL = 0
         CALL E02AEF(9, Z, XBARZ, YK, IFAIL)
         YK = YK - 0.0033867702458
C        ---------------------------------------------------------------

C        ***************************************************************
C        CALCULATE FIRST AND SECOND ORDER DERIVATIVES
C        INCLUDING THE SECOND ORDER CROSS-DERIVATIVES
C        ***************************************************************
         PT1 = SPEED(I-1) - SPEED(I)
         PT2 = PT1 * SEM
         PT3 = (SM * 9.81 * AD) + (SM * 9.81 * BD * VO)
     *       + (AREA * CD * V1V2 * 101.3 / (2.0 * R * 293.0))
         PT4 = PT3**2
         PT5 = PT3**3

         PTTM = -(AREA * CD * V1V2 * 101.3
     *        / (2.0 * R * 293.0**2)) - (ADCORF * SM * 9.81 * AD)
         PTPR = (AREA * CD * V1V2 / (2.0 * R * 293.0))
         PTCC = (AREA * CD * V1V2 * 101.3 / (2.0 * R * 293.0))
         PTWN = (AREA * CD * VO * 101.3 / (R * 293.0))
         PTGR = (SM * 9.81)
         PTMA = (9.81 * AD) + (9.81 * BD * VO)

         T1TM = -(PT2 * PTTM / PT4)
         T1PR = -(PT2 * PTPR / PT4)
         T1CC = -(PT2 * PTCC / PT4)
         T1WN = -(PT2 * PTWN / PT4)
         T1GR = -(PT2 * PTGR / PT4)
         T1MA =  (PT1 / PT3) - (PT2 * PTMA / PT4)

         T2TM2  = (2.0 * PT2 * PTTM**2 / PT5)
     *          - (PT2 * AREA * CD * V1V2 * 101.3
     *          / (PT4 * R * 293.0**3))
         T2TMPR = (2.0 * PT2 * PTTM * PTPR / PT5)
     *          + (PT2 * AREA * CD * V1V2
     *          / (PT4 * 2.0 * R * 293.0**2))
         T2TMCC = (2.0 * PT2 * PTTM * PTCC / PT5)
     *          + (PT2 * AREA * CD *V1V2 * 101.3
     *          / (PT4 * 2.0 * R * 293.0**2))
         T2TMWN = (2.0 * PT2 * PTTM * PTWN / PT5)
     *          - (PT2 * AREA * CD * VO * 101.3
```

```
     *          / (PT4 * R * 293.0**2))
          T2TMGR = (2.0 * PT2 * PTTM * PTGR / PT5)
          T2TMMA = (((2.0 * PT2 * PTMA / PT5) - (PT1 / PT4)) * PTTM)
     *          + (PT2 * ADCORF * 9.81 * AD / PT4)
          T2PR2  = (2.0 * PT2 * PTPR**2 / PT5)
          T2PRCC = (2.0 * PT2 * PTPR * PTCC / PT5)
     *          - (PT2 * AREA * CD * V1V2
     *          / (PT4 * 2.0 * R * 293.0))
          T2PRWN = (2.0 * PT2 * PTPR * PTWN / PT5)
     *          - (PT2 * AREA * CD * VO
     *          / (PT4 * R * 293.0))
          T2PRGR = (2.0 * PT2 * PTPR * PTGR / PT5)
          T2PRMA = (((2.0 * PT2 * PTMA / PT5) - (PT1 / PT4)) * PTPR)
          T2CC2  = (2.0 * PT2 * PTCC**2 / PT5)
          T2CCWN = (2.0 * PT2 * PTCC * PTWN / PT5)
     *          - (PT2 * AREA * CD * VO * 101.3
     *          / (PT4 * R * 293.0))
          T2CCGR = (2.0 * PT2 * PTCC * PTGR / PT5)
          T2CCMA = (((2.0 * PT2 * PTMA / PT5) - (PT1 / PT4)) * PTCC)
          T2WN2  = (2.0 * PT2 * PTWN**2 / PT5)
     *          - (PT2 * AREA * CD * 101.3
     *          / (PT4 * R * 293.0))
          T2WNGR = (2.0 * PT2 * PTWN * PTGR / PT5)
          T2WNMA = (((2.0 * PT2 * PTMA / PT5) - (PT1 / PT4)) * PTWN)
          T2GR2  = (2.0 * PT2 * PTGR**2 / PT5)
          T2GRMA = (((2.0 * PT2 * PTMA / PT5) - (PT1 / PT4)) * PTGR)
          T2MA2  = ((2.0 * PT2 * PTMA / PT5) - (2.0 * PT1 / PT4)) * PTMA

          CGTM = TEMP  - 293.0
          CGPR = PRESS - 101.3
          CGCC = YK    - 1.0
          CGWN = HWIND
          CGGR = GRADE
          CGMA = M     - SM
C         ---------------------------------------------------------------

C         ***************************************************************
C         APPLY CORRECTION METHOD
C         ***************************************************************
          TS = TO - (T1TM * CGTM) - (T1PR * CGPR) - (T1CC * CGCC)
     *            - (T1WN * CGWN) - (T1GR * CGGR) - (T1MA * CGMA)
     *            - (0.5 * T2TM2 * CGTM**2) - (T2TMPR * CGTM * CGPR)
     *            - (T2TMCC * CGTM * CGCC) - (T2TMWN * CGTM * CGWN)
     *            - (T2TMGR * CGTM * CGGR) - (T2TMMA * CGTM * CGMA)
     *            - (0.5 * T2PR2 * CGPR**2) - (T2PRCC * CGPR * CGCC)
     *            - (T2PRWN * CGPR * CGWN) - (T2PRGR * CGPR * CGGR)
     *            - (T2PRMA * CGPR * CGMA) - (0.5 * T2CC2 * CGCC**2)
     *            - (T2CCWN * CGCC * CGWN) - (T2CCGR * CGCC * CGGR)
     *            - (T2CCMA * CGCC * CGMA) - (0.5 * T2WN2 * CGWN**2)
     *            - (T2WNGR * CGWN * CGGR) - (T2WNMA * CGWN * CGMA)
     *            - (0.5 * T2GR2 * CGGR**2) - (T2GRMA * CGGR * CGMA)
     *            - (0.5 * T2MA2 * CGMA**2)

          CTIME(I) = CTIME(I-1) + TS
20    CONTINUE

      RETURN
      END
C     ###################################################################
```

APPENDIX E

LISTING OF FORTRAN PROGRAM CSTSIM

```
C      AAA     L         EEEEE M   M TTTTT  AAA   GGG  EEEEE
C     A   A    L         E     MM MM   T   A   A G   G E
C     AAAAA    L         EEEE  M M M   T   AAAAA G ___ EEEE
C     A   A    L         E     M   M   T   A   A G   G E
C     A   A *  LLLLL *   EEEEE M   M   T   A   A  GGG  EEEEE


C     *******************************************************************
C     PROGRAM DECLARATIONS
C     *******************************************************************
      PROGRAM CSTSIM
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      PARAMETER (NTERM=1, NDATA=5, NRES1=6)
      DOUBLE PRECISION M
      CHARACTER *70 FILIN, FILOU1, VEHICL, ANSWER
      CHARACTER *8 DATE
      CHARACTER *5 SPUNIT, TEUNIT
      CHARACTER *1 DIRECT
      DIMENSION Z(9), TIME(500), SPEED(500), SYK(500)
      COMMON /TUFTY1/ AD, BD, CD, AREA, W, EM
      COMMON /TUFTY2/ ADCORF, TEMP, YK, PRESS, R, HWIND, GRADEZ

      DATA Z(1), Z(2), Z(3), Z(4), Z(5), Z(6), Z(7), Z(8), Z(9)/
     * 0.1651178002D 01, -0.5768188453D 00, -0.4510661009D 00,
     * 0.9379718523D-01, 0.1207860561D 00, -0.2477960124D-01,
     * 0.1212739029D-01, 0.1230558182D-01, -0.1184442607D-01/

      AD = 0.010
      BD = 0.0004
      CD = 0.350
      R = 0.288

      SDSP = 0.0
      SDGR = 0.0
      SDHW = 0.0
      SDXW = 0.0
C     -------------------------------------------------------------------

C     *******************************************************************
C     INPUT REQUIRED VALUES OF COEFFICIENTS
C     *******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COU('Please enter required values for the vehicle')
      CALL COU('coefficients, or simply press RETURN to select the')
      CALL COU('default values; Ad=0.01, Bd=0.0004, Cd=0.35')

      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Ad =  ')
      READ (UNIT=NTERM, FMT='(A)') ANSWER
      CALL LFTJST(ANSWER,NCHAR)
      IF (NCHAR.EQ.0) GOTO 1
      DECODE (ANSWER, 8888) AD

1     WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Bd =  ')
      READ (UNIT=NTERM, FMT='(A)') ANSWER
      CALL LFTJST(ANSWER,NCHAR)
      IF (NCHAR.EQ.0) GOTO 2
      DECODE (ANSWER, 8888) BD
```

```
2     WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Cd =  ')
      READ (UNIT=NTERM, FMT='(A)') ANSWER
      CALL LFTJST(ANSWER,NCHAR)
      IF (NCHAR.EQ.0) GOTO 3
      DECODE (ANSWER, 8888) CD
3     CONTINUE
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT TRACK GRADIENT
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter track gradient.  ')
      READ (UNIT=NTERM, FMT=*) GRADE
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT TEMPERATURE CORRECTION FACTOR FOR Ad
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter temperature correction factor for Ad.  ')
      READ (UNIT=NTERM, FMT=*) ADCORF
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT STANDARD VEHICLE MASS
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter standard vehicle mass (kg).  ')
      READ (UNIT=NTERM, FMT=*) SM
      SW = SM * 9.81
C     ------------------------------------------------------------------

C     ******************************************************************
C     OPEN DATA FILE
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter name of data file. ')
      READ (UNIT=NTERM, FMT='(A)') FILIN
      OPEN (UNIT=NDATA, FILE=FILIN, FORM='FORMATTED', CARRIAGE=.FALSE.)
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT VEHICLE NAME
C     ******************************************************************
      READ (UNIT=NDATA, FMT='(A)') VEHICL
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT VEHICLE CONSTANTS
C     ******************************************************************
      READ (UNIT=NDATA, FMT=*) AREA, CAL1
      READ (UNIT=NDATA, FMT=*) RR,   CAL2
      READ (UNIT=NDATA, FMT=*) WI,   CAL3

      AREA = AREA * CAL1
      RR = RR * CAL2
```

```
      WI = WI * CAL3
      SEM = SM + WI / RR**2
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT UNITS FOR VEHICLE SPEED AND SET SPEED FACTOR
C     ******************************************************************
      READ (UNIT=NDATA, FMT='(A)') SPUNIT
      CALL UPCASE(SPUNIT)
      IF(SPUNIT.EQ.'KM/H') VCONST = 1.0 / 3.6
      IF(SPUNIT.EQ.'M/S' ) VCONST = 1.0
      IF(SPUNIT.EQ.'MPH' ) VCONST = 1.609344 / 3.6
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT NUMBER OF TEST GROUPS
C     ******************************************************************
      READ (UNIT=NDATA, FMT=*) N3
      N3 = 1
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT NUMBER OF TEST SETS IN GROUP
C     ******************************************************************
      READ (UNIT=NDATA, FMT=*) N2
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT NUMBER OF DATA POINTS IN TEST SET
C     ******************************************************************
      NTESTC = 0
      READ (UNIT=NDATA, FMT=*) N
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT DATE OF TEST
C     ******************************************************************
      READ (UNIT=NDATA, FMT='(A)') DATE
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT AMBIENT CONDITIONS
C     ******************************************************************
      READ (UNIT=NDATA, FMT='(A)') DIRECT
      READ (UNIT=NDATA, FMT=*)     M,     CAL4
      READ (UNIT=NDATA, FMT=*)     WINDV, CAL5
      READ (UNIT=NDATA, FMT=*)     WINDD
      READ (UNIT=NDATA, FMT=*)     PRESS, CAL6
      READ (UNIT=NDATA, FMT=*)     TEMYK
      TEMP = TEMYK
      READ (UNIT=NDATA, FMT='(A)') TEUNIT

      M = M * CAL4
      DM = SM - M
      W = M * 9.81
      EM = M + WI / RR**2
      WINDV = WINDV * CAL5
      PRESS = PRESS * CAL6
      WINDD = WINDD * 0.01745
```

```
      CALL UPCASE(TEUNIT)
      IF(TEUNIT.EQ.'F')
     * TEMP = (TEMP - 32.0) / 1.8
      IF (TEUNIT.NE.'K') TEMP = TEMP + 273.0
C     ------------------------------------------------------------------

C     ******************************************************************
C     CLOSE DATA FILE
C     ******************************************************************
      CLOSE (UNIT=NDATA)
C     ------------------------------------------------------------------

C     ******************************************************************
C     OPEN RESULTS FILE
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter name for results file.  ')
      READ (UNIT=NTERM, FMT='(A)') FILOU1
      OPEN (UNIT=NRES1, FILE=FILOU1, FORM='FORMATTED', CARRIAGE=.FALSE.)
C     ------------------------------------------------------------------

C     ******************************************************************
C     WRITE HEADER INFORMATION TO RESULTS FILE
C     ******************************************************************
      WRITE (UNIT=NRES1, FMT='(A)') VEHICL
      AREA = AREA / CAL1
      WRITE (UNIT=NRES1, FMT=*) AREA, CAL1
      RR = RR / CAL2
      WRITE (UNIT=NRES1, FMT=*) RR, CAL2
      WI = WI / CAL3
      WRITE (UNIT=NRES1, FMT=*) WI, CAL3
      WRITE (UNIT=NRES1, FMT='(A)') SPUNIT
      WRITE (UNIT=NRES1, FMT=*) N3
      WRITE (UNIT=NRES1, FMT=*) N2
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT CONSTRAINTS OF COAST-DOWN DATA
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter the initial vehicle speed (m/s).  ')
      READ (UNIT=NTERM, FMT=*) V1
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter the final vehicle speed (m/s).  ')
      READ (UNIT=NTERM, FMT=*) V2
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter size of speed intervals (m/s).  ')
      READ (UNIT=NTERM, FMT=*) VINT
      V2 = V2 - (0.1 * VINT)
C     ------------------------------------------------------------------

C     ******************************************************************
C     INPUT STANDARD DEVIATIONS OF ERRORS
C     ******************************************************************
      WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter SIGMA for vehicle speed (m/s).  ')
      READ (UNIT=NTERM, FMT='(A)') ANSWER
      CALL UPCASE(ANSWER)
```

```
      CALL LFTJST(ANSWER, NCHAR)
      IF (NCHAR.EQ.0) GOTO 4
      DECODE (ANSWER, 8888) SDSP

4     WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter SIGMA for track gradient.  ')
      READ (UNIT=NTERM, FMT='(A)') ANSWER
      CALL UPCASE(ANSWER)
      CALL LFTJST(ANSWER, NCHAR)
      IF (NCHAR.EQ.0) GOTO 5
      DECODE (ANSWER, 8888) SDGR

5     WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter SIGMA for wind speed (m/s).  ')
      READ (UNIT=NTERM, FMT='(A)') ANSWER
      CALL UPCASE(ANSWER)
      CALL LFTJST(ANSWER, NCHAR)
      IF (NCHAR.EQ.0) GOTO 6
      DECODE (ANSWER, 8888) SDWS

6     WRITE (UNIT=NTERM, FMT=*)
      CALL COUA('Please enter SIGMA for wind direction (deg).  ')
      READ (UNIT=NTERM, FMT='(A)') ANSWER
      CALL UPCASE(ANSWER)
      CALL LFTJST(ANSWER, NCHAR)
      IF (NCHAR.EQ.0) GOTO 7
      DECODE (ANSWER, 8888) SDWD

7     CONTINUE
C     ------------------------------------------------------------------

C     ******************************************************************
C     INITIALIZE RANDOM NUMBER GENERATOR WITH SEED BASED ON REAL TIME
C     ******************************************************************
      CALL G05CCF
C     ------------------------------------------------------------------

      DO 50, J = 1, N2

C     ******************************************************************
C     GENERATE SIMULATED COAST DOWN DATA
C     ******************************************************************
      I = 0
      DO 30, SP = V1, V2 - 0.0001, -VINT
         WS = G05DDF(WINDV, SDWS)
         WD = G05DDF(WINDD, SDWD * 0.01745)
         GRADEZ = G05DDF(GRADE, SDGR)
         HWIND = WS * COS(WD)
         XWIND = ABS(WS * SIN(WD))
         I = I + 1
         SPEED(I) = SP
         IF (I.EQ.1) THEN
            TIME(I) = 0.0
            SYK(I) = SP
            GOTO 30
         END IF
         V0 = (SPEED(I-1) + SPEED(I)) / 2.0
         VHR = V0 + HWIND
         IF((VHR.LT.0.000001).AND.(VHR.GT.-0.000001)) PSI = 90.0
```

```
      IF((PSI.EQ.90.0).AND.(XWIND.LT.0.001)) PSI = 0.0
      IF((VHR.LT.0.000001).AND.(VHR.GT.-0.000001)) GOTO 10
      PSI = ATAN(XWIND / VHR)
      PSI = PSI * 180.0 / 3.1416
10    XBARZ = ((PSI - (-2.0)) - (92.0 - PSI)) / (92.0 - (-2.0))
C     EVALUATE POLYNOMIAL FROM CHEBYSHEV SERIES REPRESENTATION
      CALL E02AEF(9, Z, XBARZ, YK, IFAIL)
      YK = YK - 0.0033867702458
      SYK(I) = G05DDF(SP, SDSP)
      TIME(I) = FUNC(SYK(I-1),SYK(I),TIME(I-1))
30    CONTINUE
      N = I
C     ----------------------------------------------------------------

C     ****************************************************************
C     WRITE SIMULATED COAST-DOWN DATA TO RESULTS FILE
C     ****************************************************************
      WRITE (UNIT=NRES1, FMT=*) N
      WRITE (UNIT=NRES1, FMT='(A)') DATE
      WRITE (UNIT=NRES1, FMT='(A)') DIRECT
      M = M / CAL4
      WRITE (UNIT=NRES1, FMT=*) M, CAL4
      WINDV = WINDV / CAL5
      WRITE (UNIT=NRES1, FMT=*) WINDV, CAL5
      WRITE (UNIT=NRES1, FMT=*) WINDD / 0.01745
      PRESS = PRESS / CAL5
      WRITE (UNIT=NRES1, FMT=*) PRESS, CAL6
      WRITE (UNIT=NRES1, FMT=*) TEMYK
      WRITE (UNIT=NRES1, FMT='(A)') TEUNIT

      DO 40, I = 1, N
         SPEED(I) = SPEED(I) / VCONST
         WRITE (UNIT=NRES1, FMT=*) TIME(I), SPEED(I)
40    CONTINUE
C     ----------------------------------------------------------------

50    CONTINUE

C     ****************************************************************
C     CLOSE RESULTS FILE
C     ****************************************************************
      CLOSE (UNIT=NRES1)
C     ----------------------------------------------------------------

C     ****************************************************************
C     PROGRAM TERMINATION
C     ****************************************************************
      STOP
8888  FORMAT(v)
      END
C     ################################################################
```

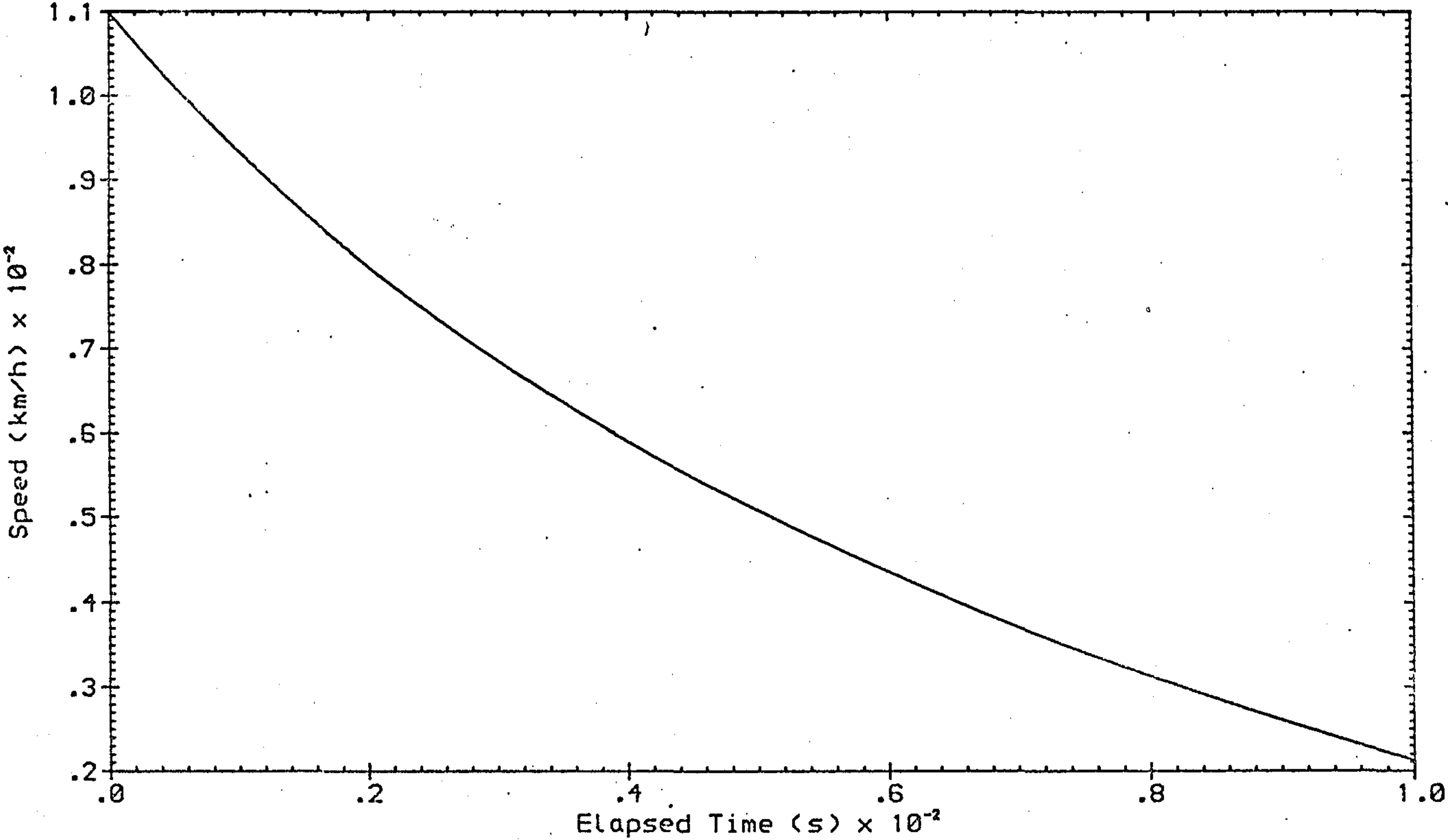
```
C     ******************************************************************
C     FUNCTION DEFINING SPEED AS A FUNCTION OF TIME
C     ******************************************************************
      FUNCTION FUNC(V1,V2,T1)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /TUFTY1/ AD, BD, CD, AREA, W, EM
      COMMON /TUFTY2/ ADCORF, TEMP, YK, PRESS, R, HWIND, GRADE
      COMMON /TUFTY3/ ALFA, BETA, CNST1, CNST2

      A = (W * AD * (1.0 - ADCORF * (TEMP - 293.0))) + (W * GRADE)
     *  + (0.5 * (PRESS / (R * TEMP)) * AREA * CD * YK * HWIND**2)
      B = (W * BD) + ((PRESS / (R * TEMP)) * AREA * CD * YK * HWIND)
      C = (0.5 * (PRESS / (R * TEMP)) * AREA * CD * YK)

      TEST = (A / C) - (B**2 / 4.0 / C**2)

      ALFA = SQRT(ABS(A / C - B * B / 4.0 / C / C))
      BETA = C / EM
      GAMA = B / 2.0 / C

      IF (TEST.NE.0.0) THEN
         CNST1 = (V1 + GAMA) / ALFA
         CNST2 = (V2 + GAMA) / ALFA
      ELSE
         CNST1 = 1.0 / (V1 + GAMA)
         CNST2 = 1.0 / (V2 + GAMA)
      END IF

      IF (TEST.GT.0.0) THEN
         FUNC = FALFAP(T1)
      ELSE IF (TEST.EQ.0.0) THEN
         FUNC = FALFAO(T1)
      ELSE IF (TEST.LT.0.0) THEN
         FUNC = FALFAM(T1)
      END IF

      RETURN
      END
C     ##################################################################




C     ******************************************************************
C     FUNCTION TO EVALUATE TIME WHEN ALFA**2 > 0.0
C     ******************************************************************
      FUNCTION FALFAP(T1)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /TUFTY3/ ALFA, BETA, CNST1, CNST2

      FALFAP = T1 + (ATAN(CNST1) - ATAN(CNST2)) / (ALFA * BETA)

      RETURN
      END
C     ##################################################################
```

```
C     ***************************************************************
C     FUNCTION TO EVALUATE TIME WHEN ALFA**2 = 0.0
C     ***************************************************************
      FUNCTION FALFAO(T1)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /TUFTY3/ ALFA, BETA, CNST1, CNST2

      FALFAO = T1 + (CNST2 - CNST1) / BETA

      RETURN
      END
C     ################################################################
```

```
C     ***************************************************************
C     FUNCTION TO EVALUATE TIME WHEN ALFA**2 < 0.0
C     ***************************************************************
      FUNCTION FALFAM(T1)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /TUFTY3/ ALFA, BETA, CNST1, CNST2

      FALFAM = T1 + (TANHM1(CNST1) - TANHM1(CNST2)) / (ALFA * BETA)

      RETURN
      END
C     ################################################################
```

```
C     ***************************************************************
C     INVERSE TANH FUNCTION
C     ***************************************************************
      FUNCTION TANHM1(X)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)

      IF (X.GE.1.0) THEN
         PRINT *
         PRINT *, '***ERROR***  In function TANHM1.'
         PRINT *, 'Argument > or = 1.0'
         STOP
      END IF
```

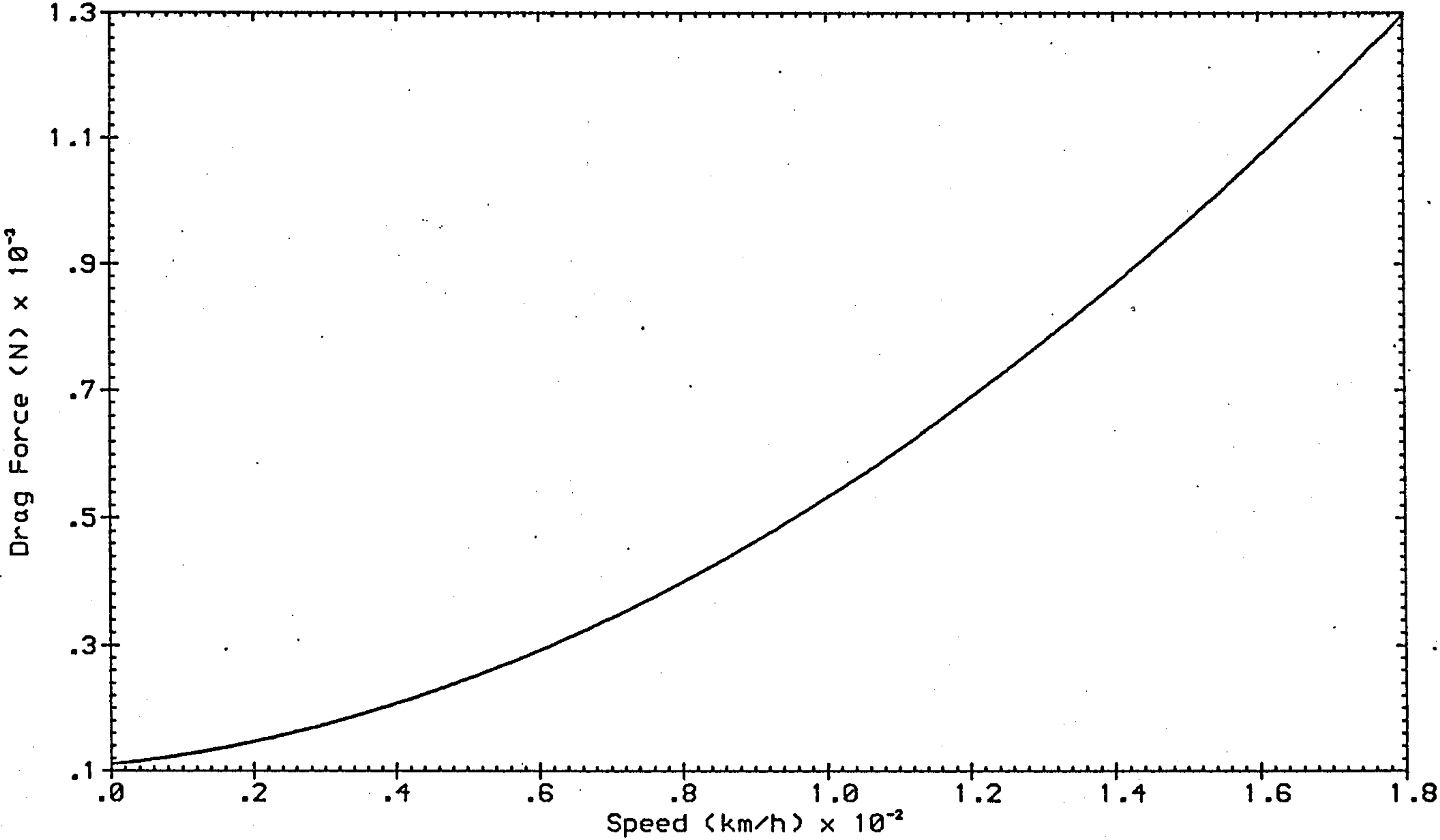
```
      TANHM1 = 0.5 * LOG((1.0 + X) / (1.0 - X))

      RETURN
      END
C     ##################################################################
```



TYPICAL COAST-DOWN CHARACTERISTIC FOR A SMALL CAR (SIMULATED)

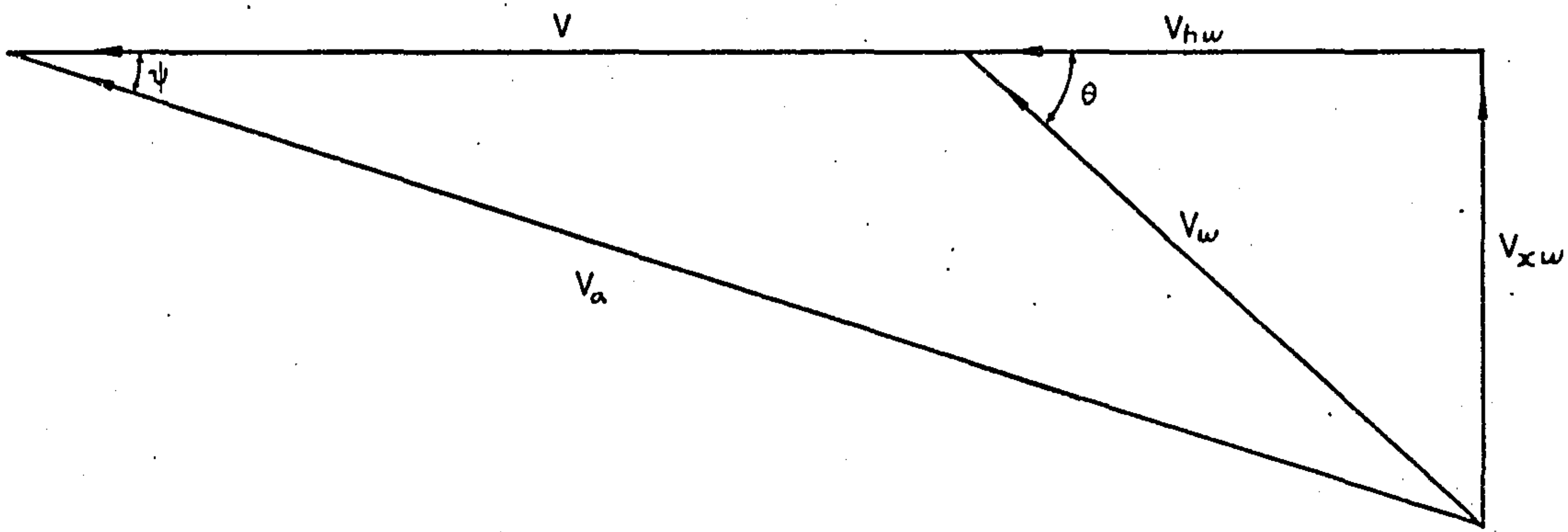
Fig 1.1

Tufty.Plot



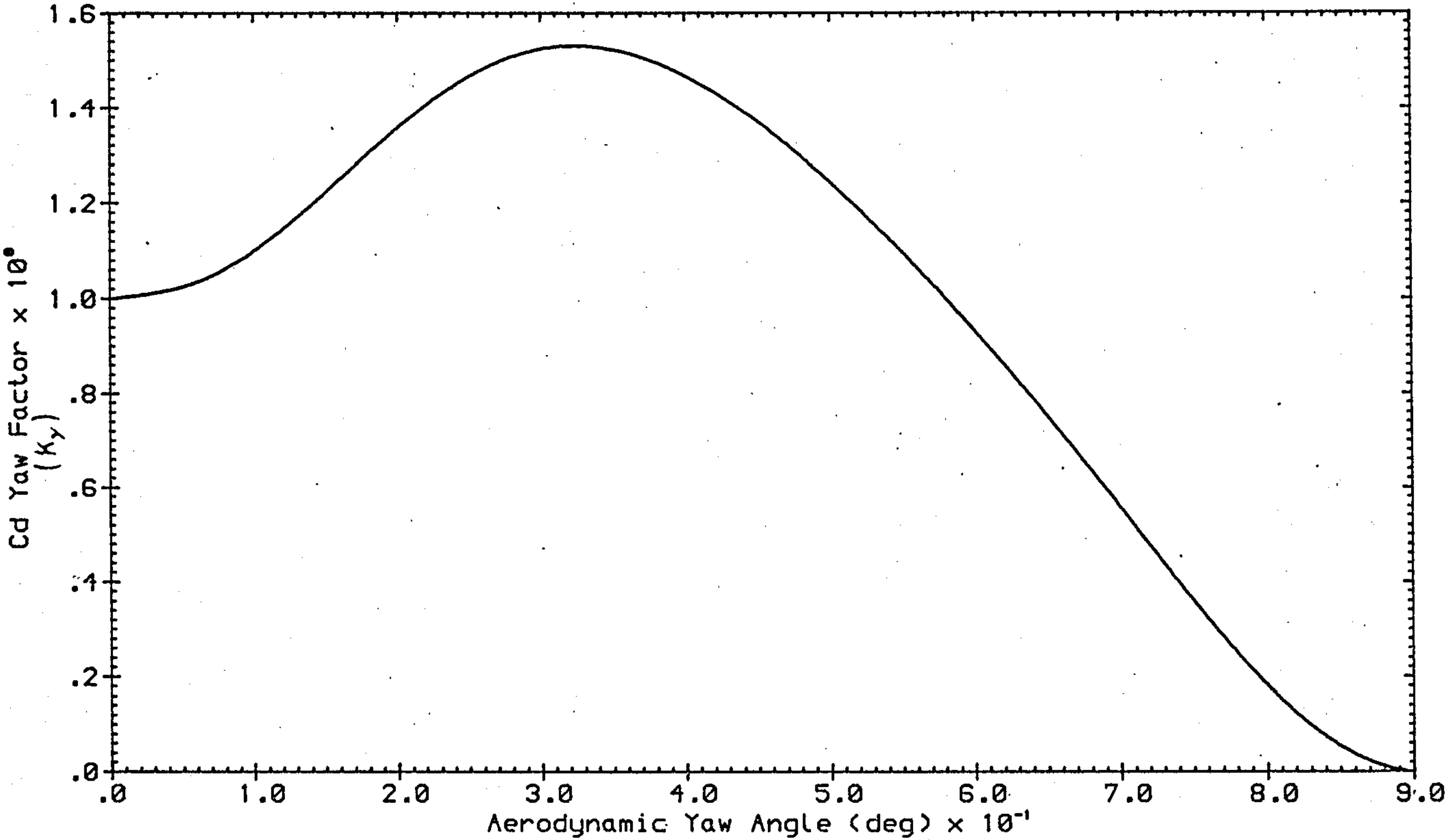
TYPICAL DRAG FORCE vrs SPEED CHARACTERISTIC FOR A SMALL CAR (SIMULATED)

Tufty.Plot



VELOCITY VECTOR DIAGRAM

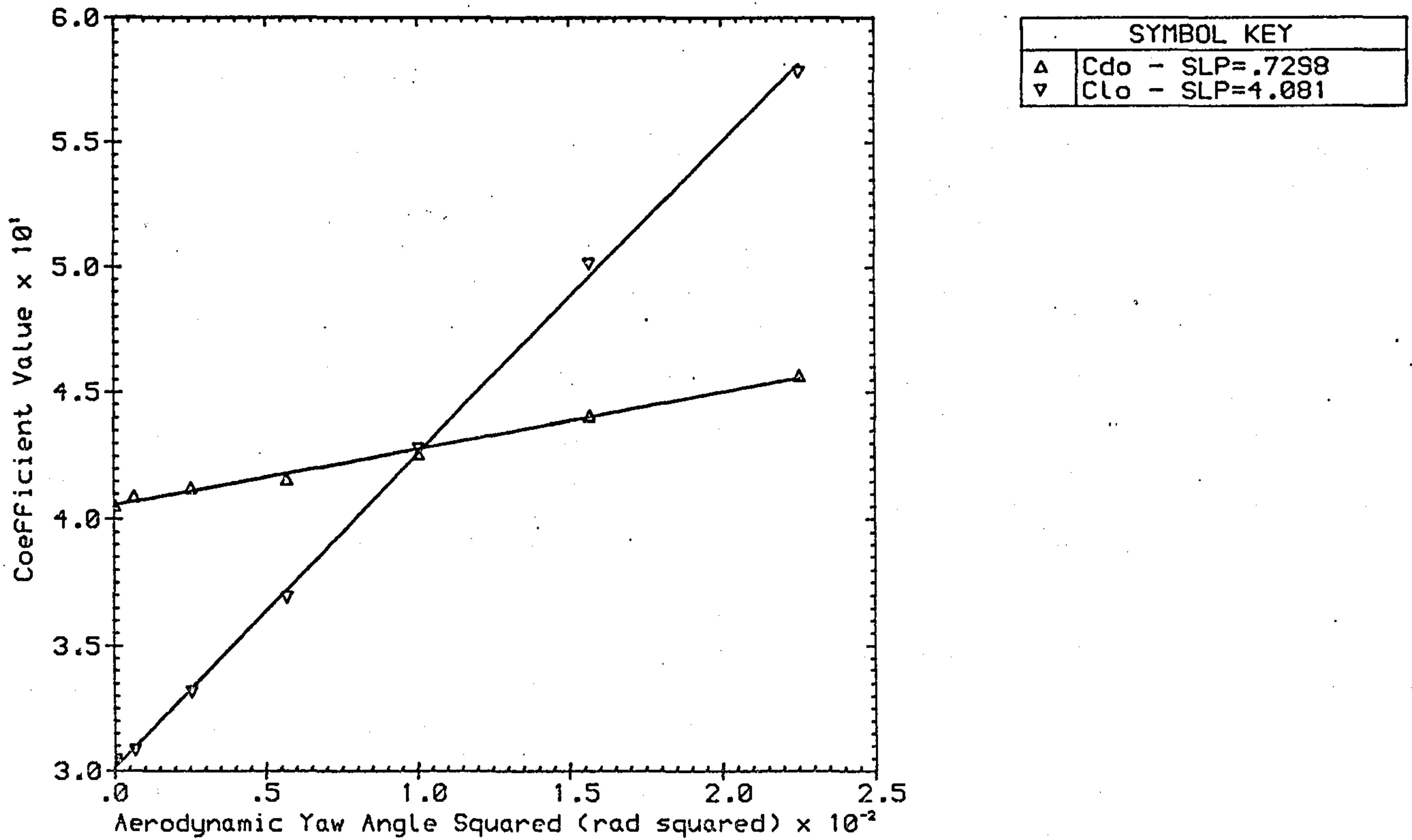
Fig 2.1



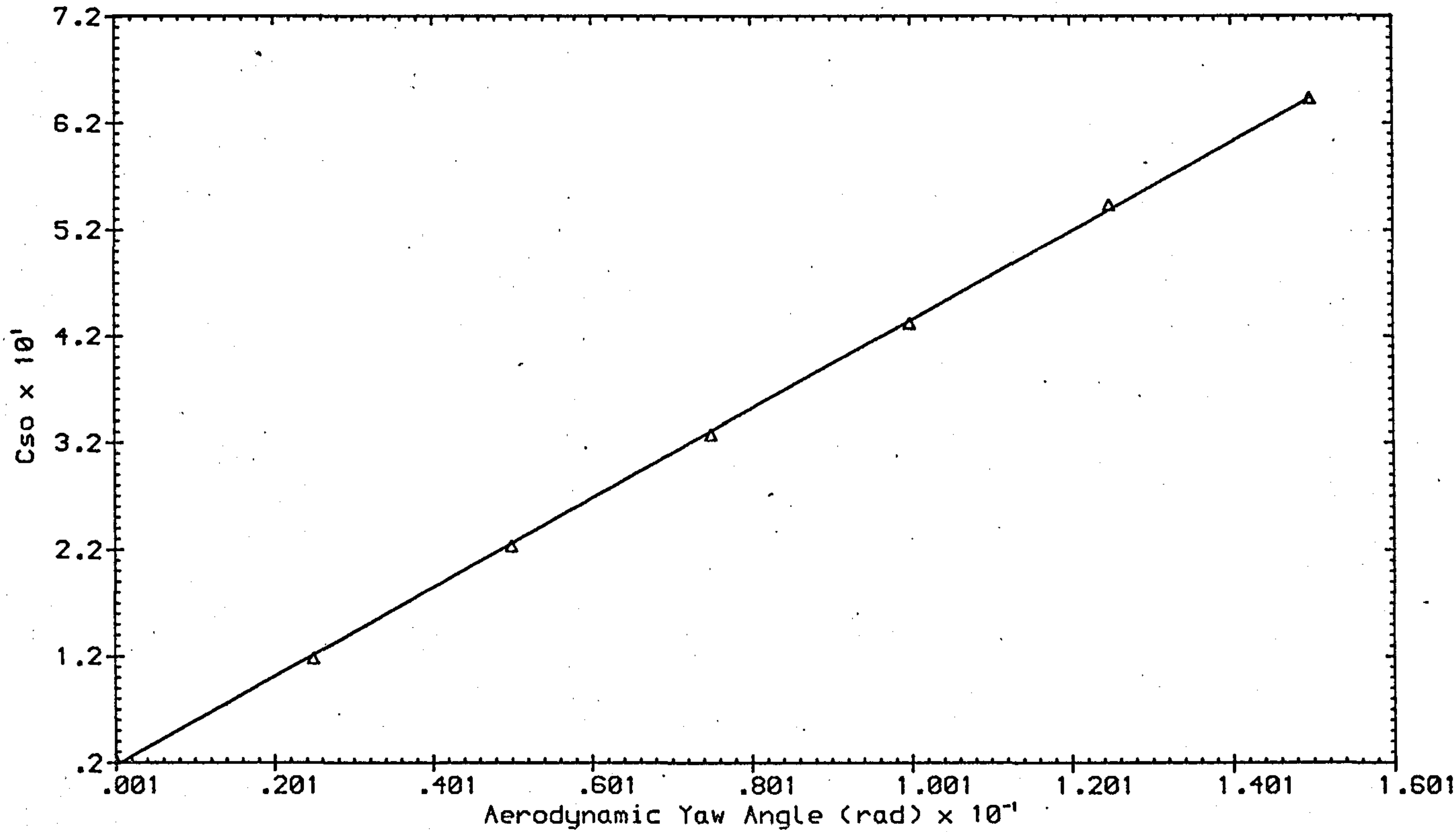
TYPICAL Cd YAW FACTOR vrs YAW ANGLE CHARACTERISTIC

Tufty.Plot

Fig 2.2



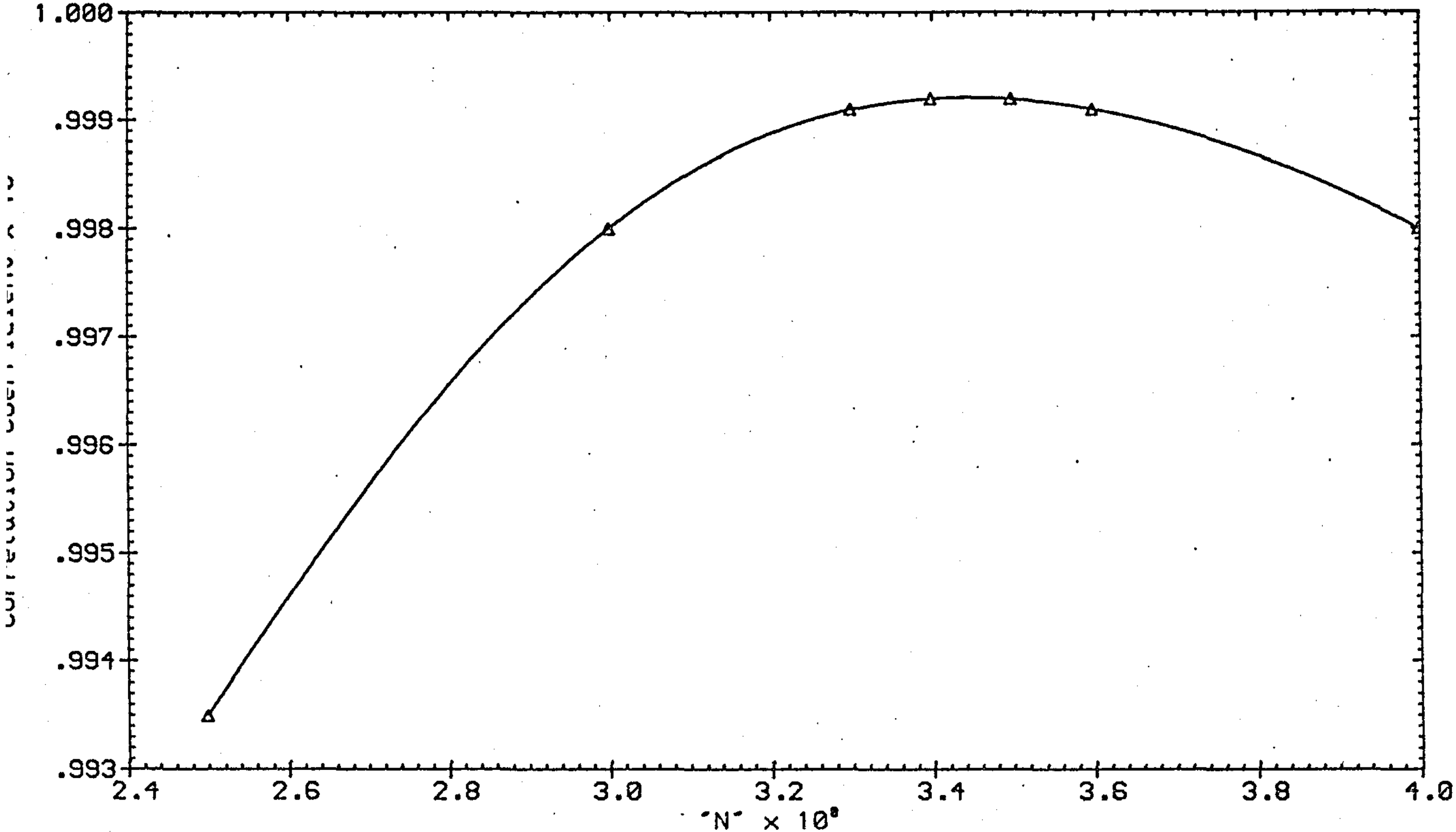
TYPICAL CHARACTERISTICS OF Cdo AND Clo vrs YAW ANGLE SQUARED



TYPICAL CHARACTERISTIC OF Cso vrs YAW ANGLE (SLP=2.392)

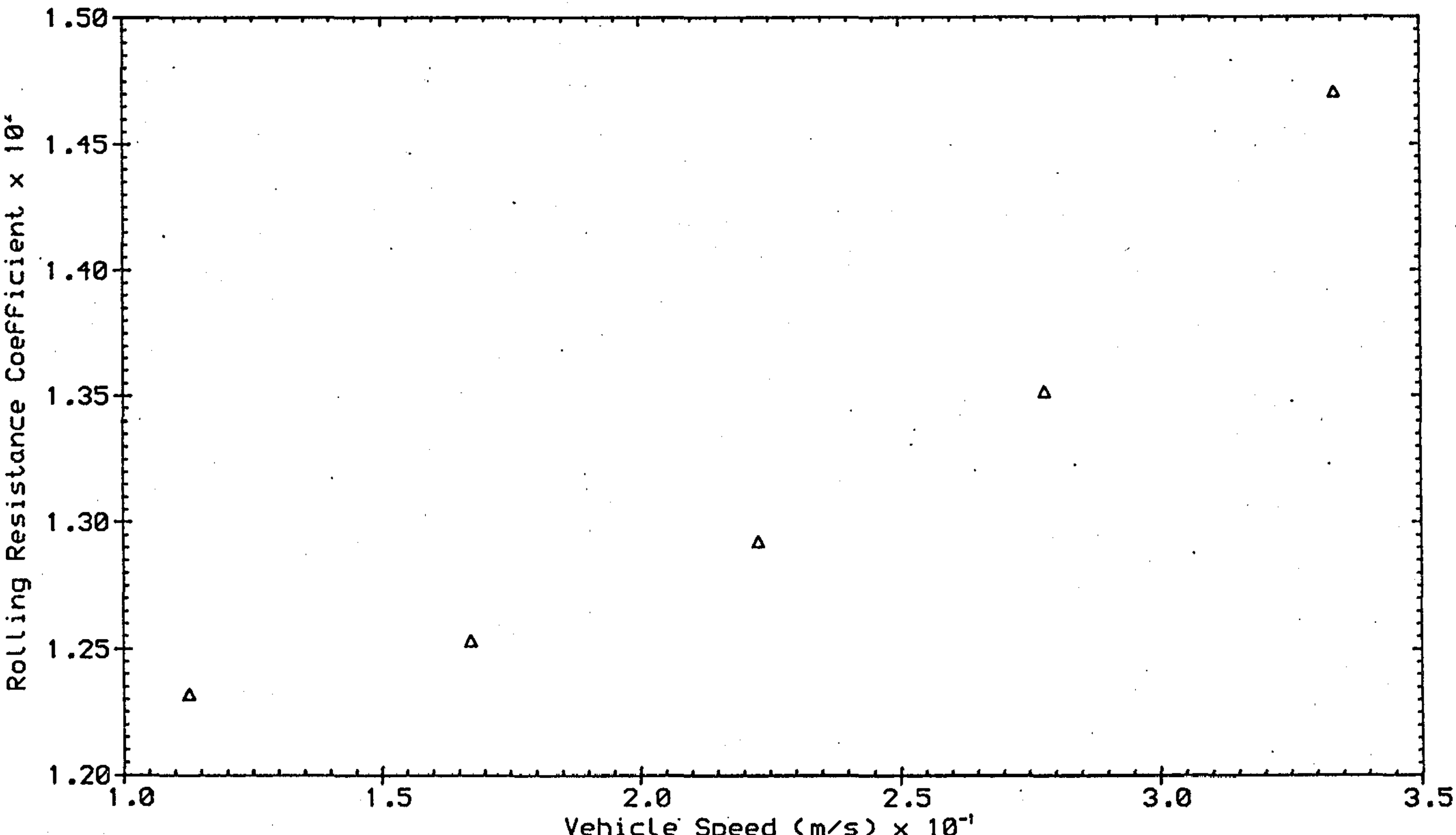
Fig 2.4

TuftyPlot



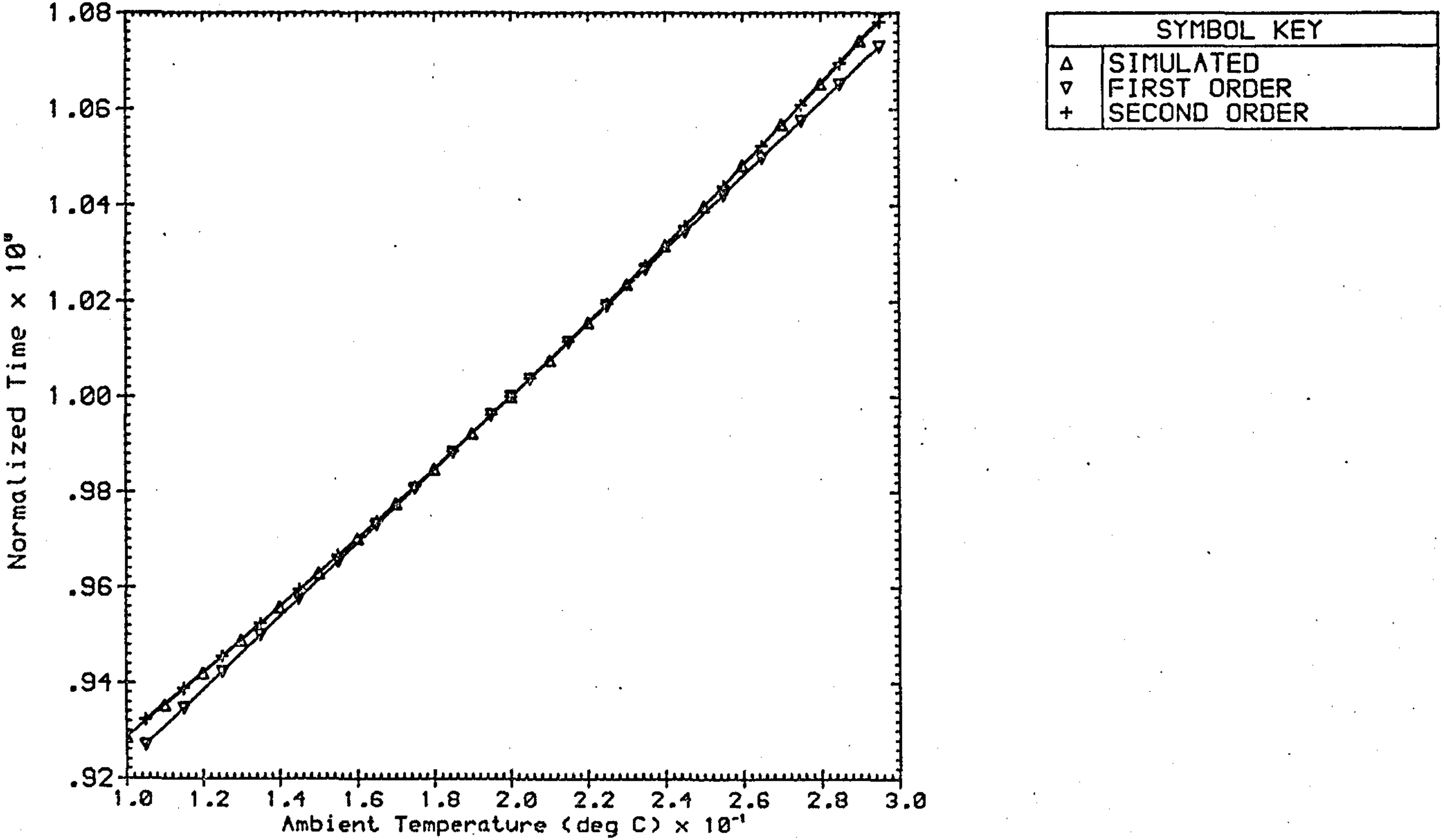
GRAPH OF CORRELATION COEFFICIENT vrs POWER 'N'

Fig 4.5



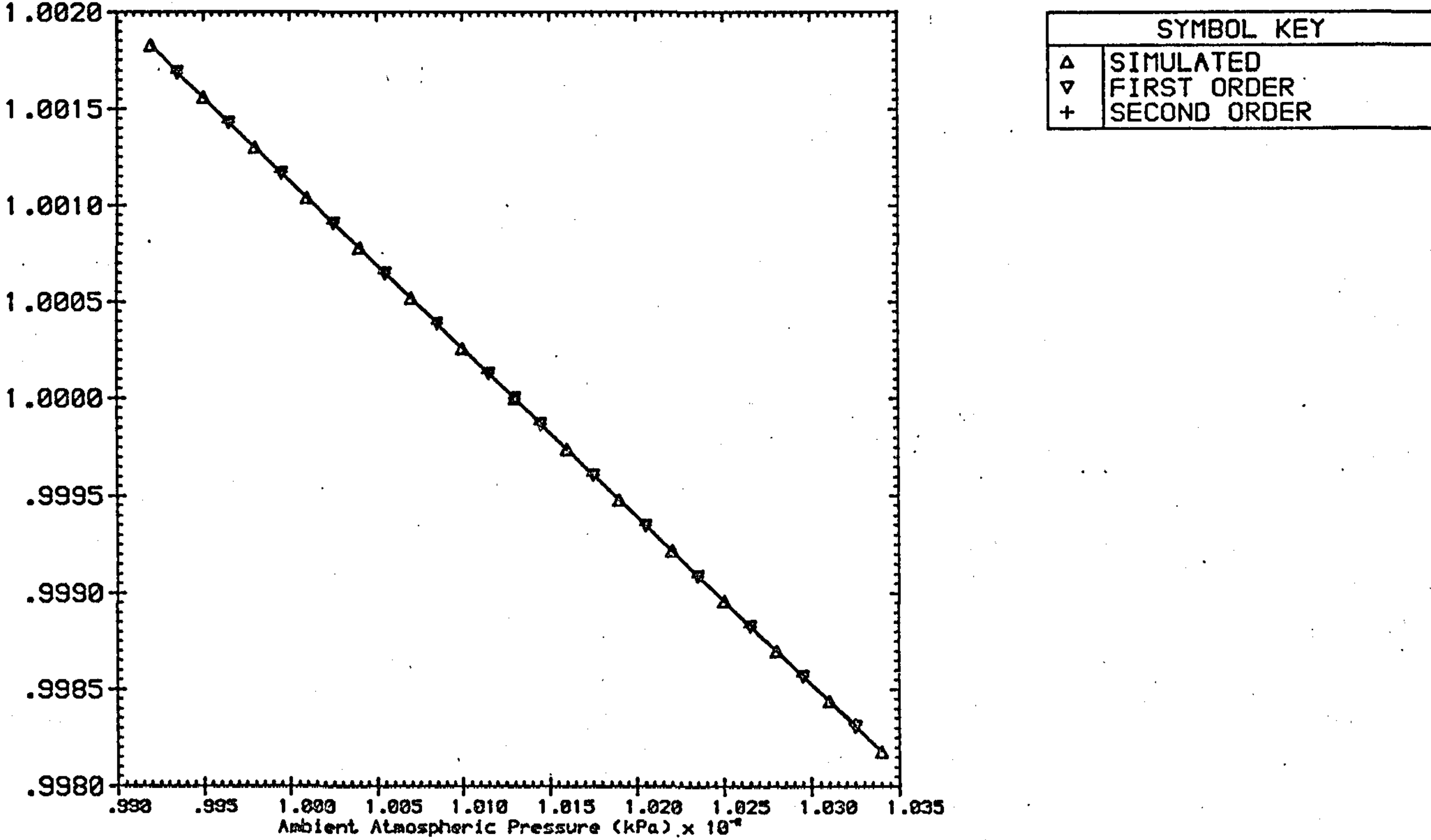
TYPICAL ROLLING RESISTANCE COEFFICIENT vrs VEHICLE SPEED CHARACTERISTIC

Fig 7.6



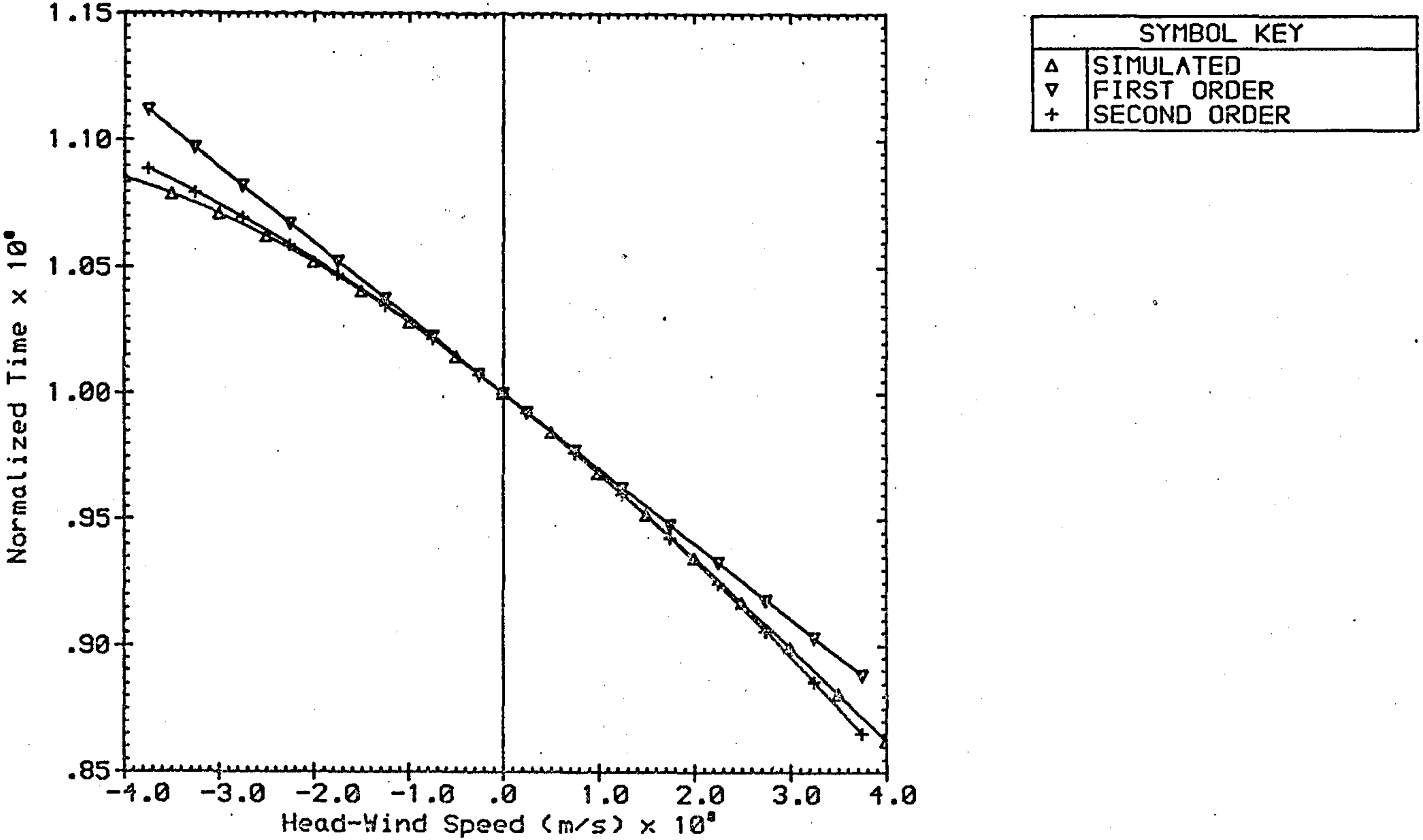
VARIATION OF NORMALIZED TIME WITH AMBIENT TEMPERATURE

Fig 3.1

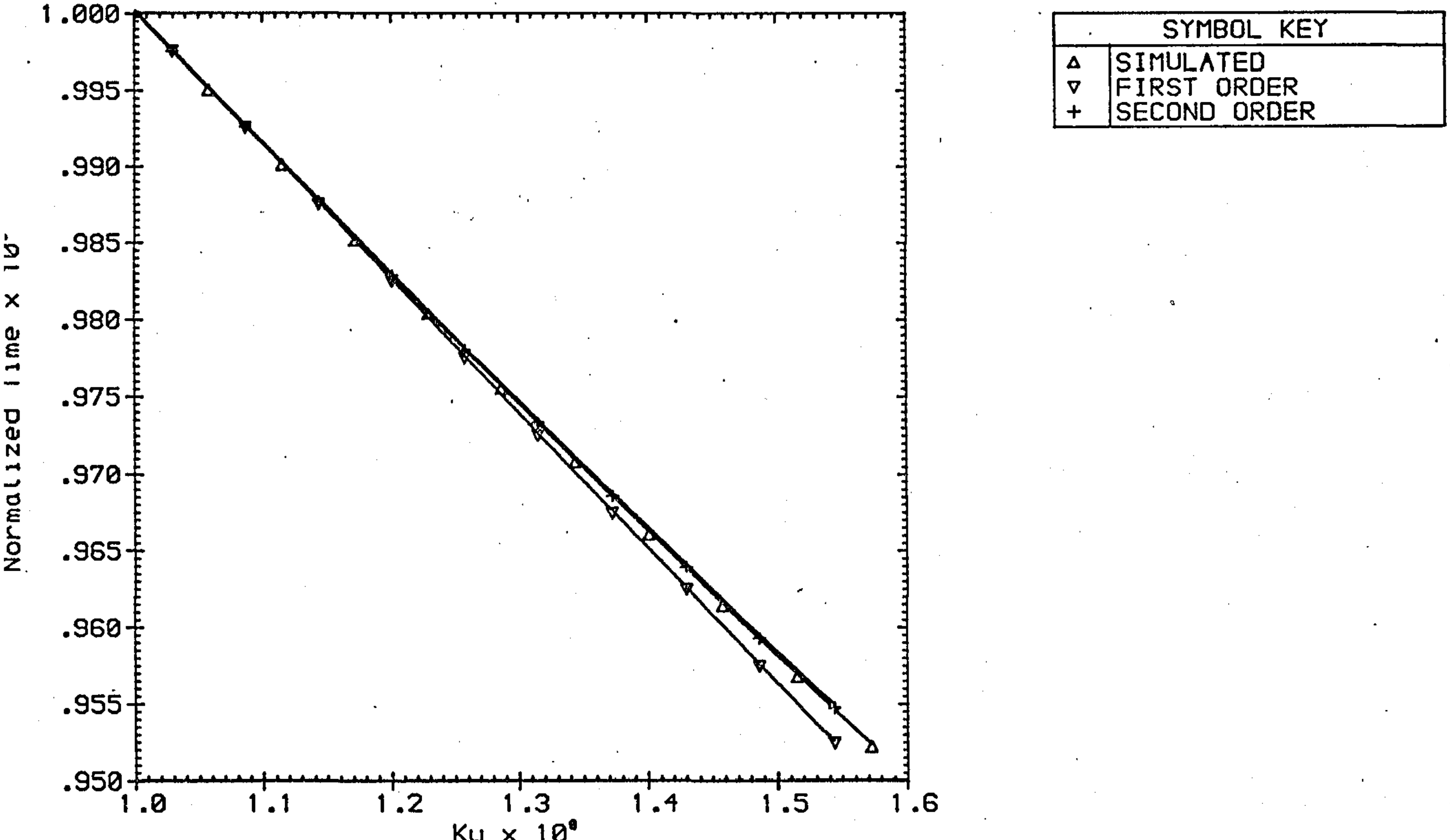


VARIATION OF NORMALIZED TIME WITH AMBIENT ATMOSPHERIC PRESSURE

Fig 3.2

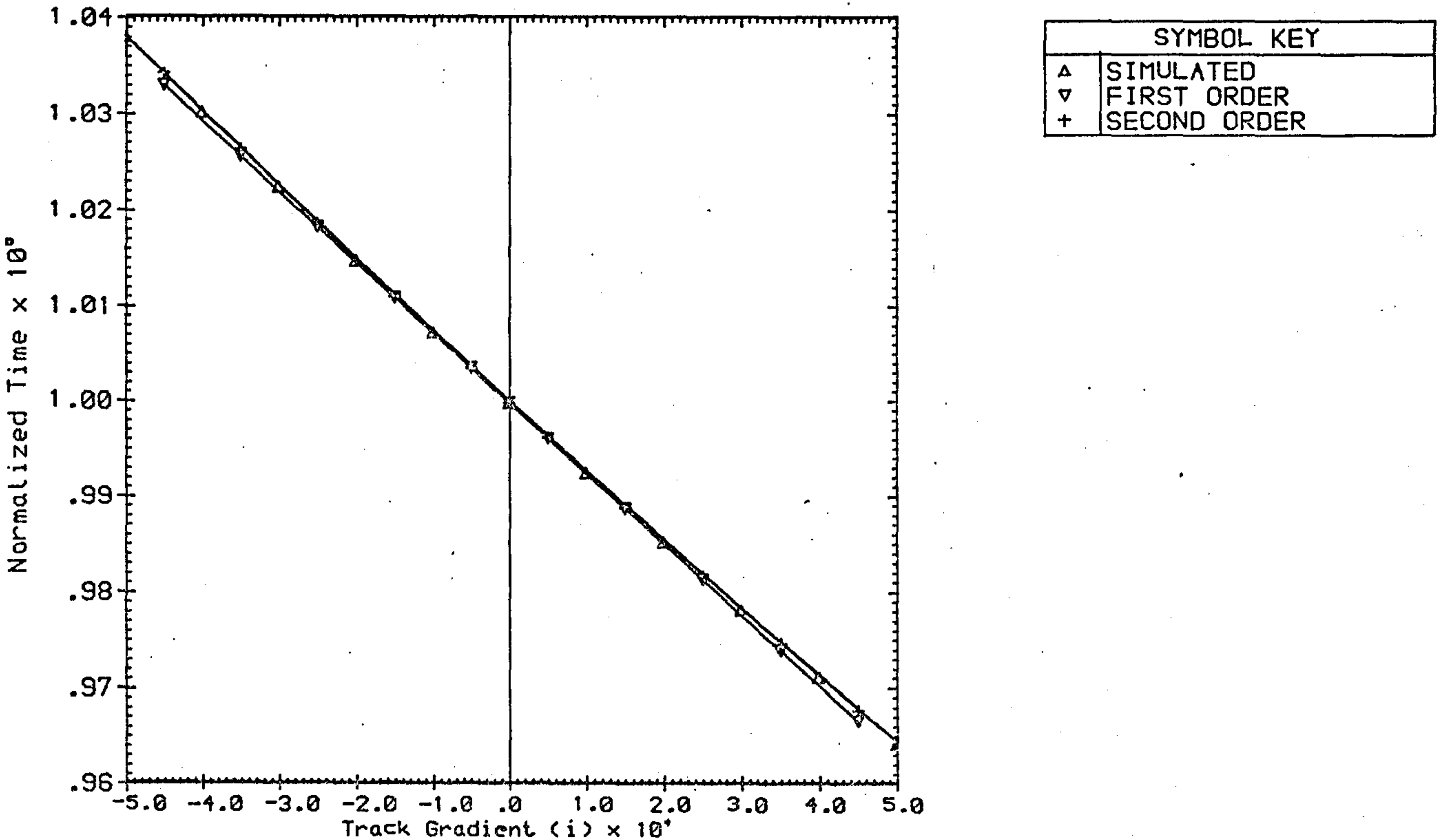


VARIATION OF NORMALIZED TIME WITH HEAD-WIND SPEED

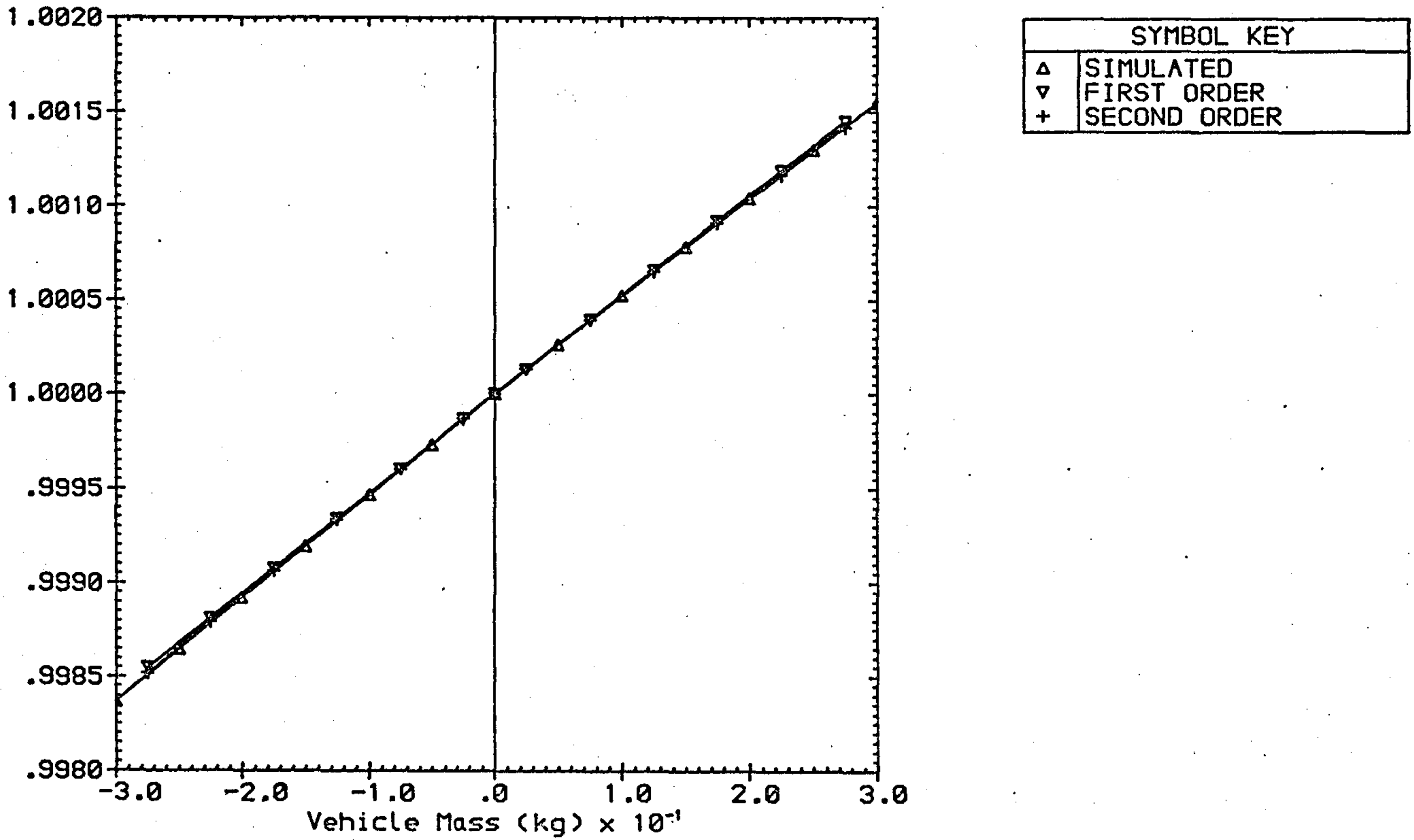


VARIATION OF NORMALIZED TIME WITH Cd YAW FACTOR (Ky)

Fig 5.7

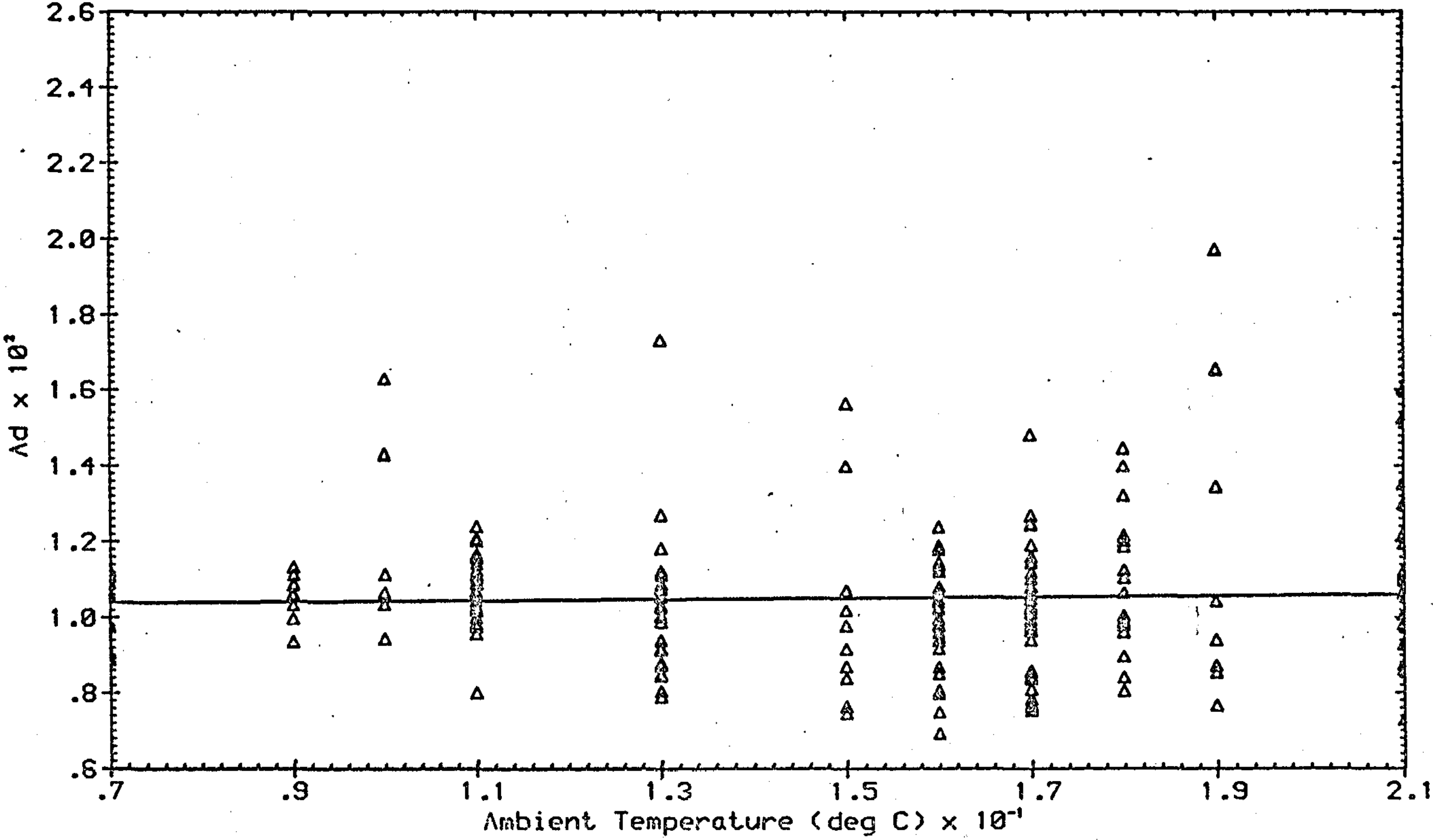


VARIATION OF NORMALIZED TIME WITH TRACK GRADIENT (i)

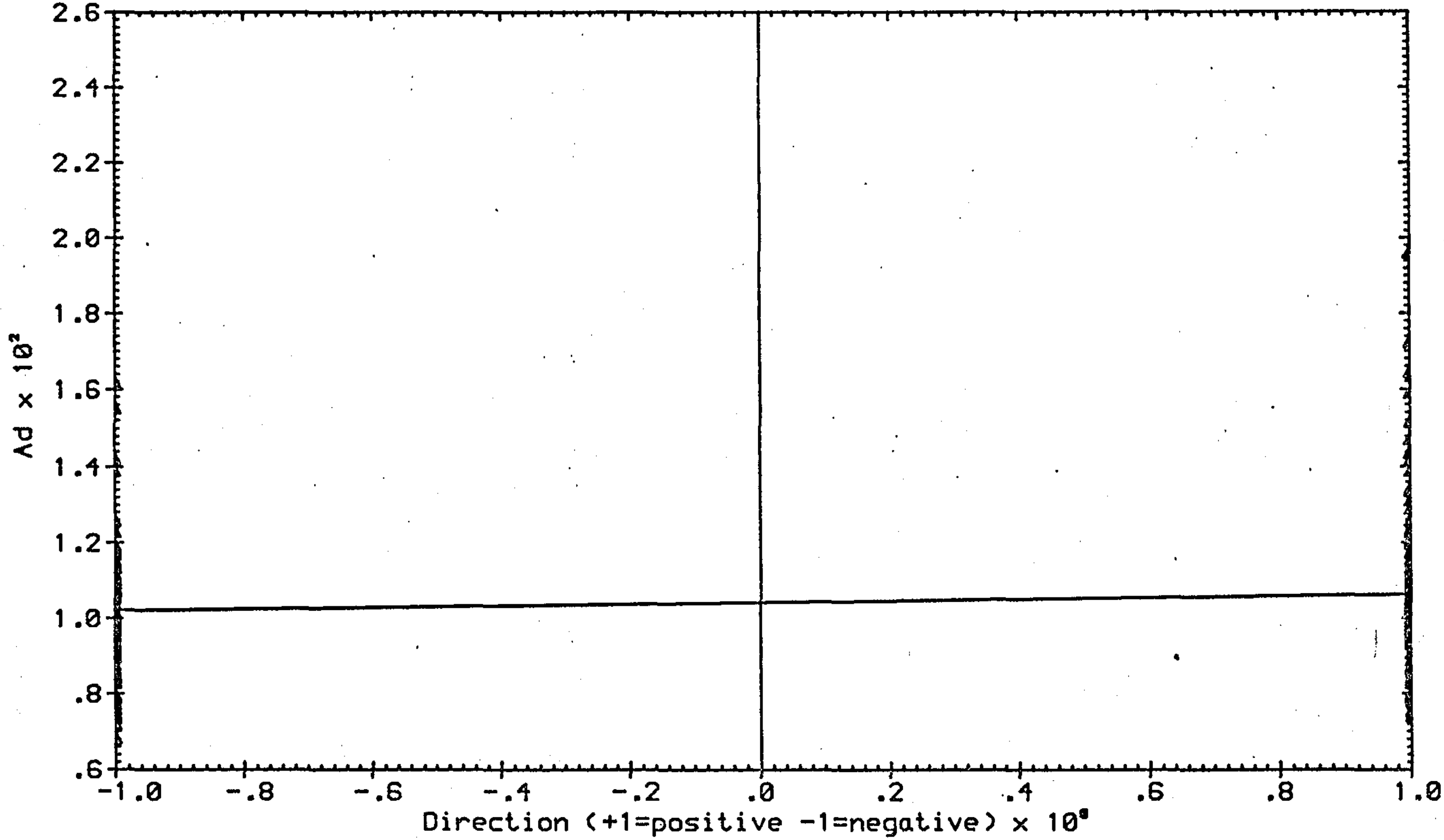


VARIATION OF NORMALIZED TIME WITH VEHICLE MASS

Fig 3.6

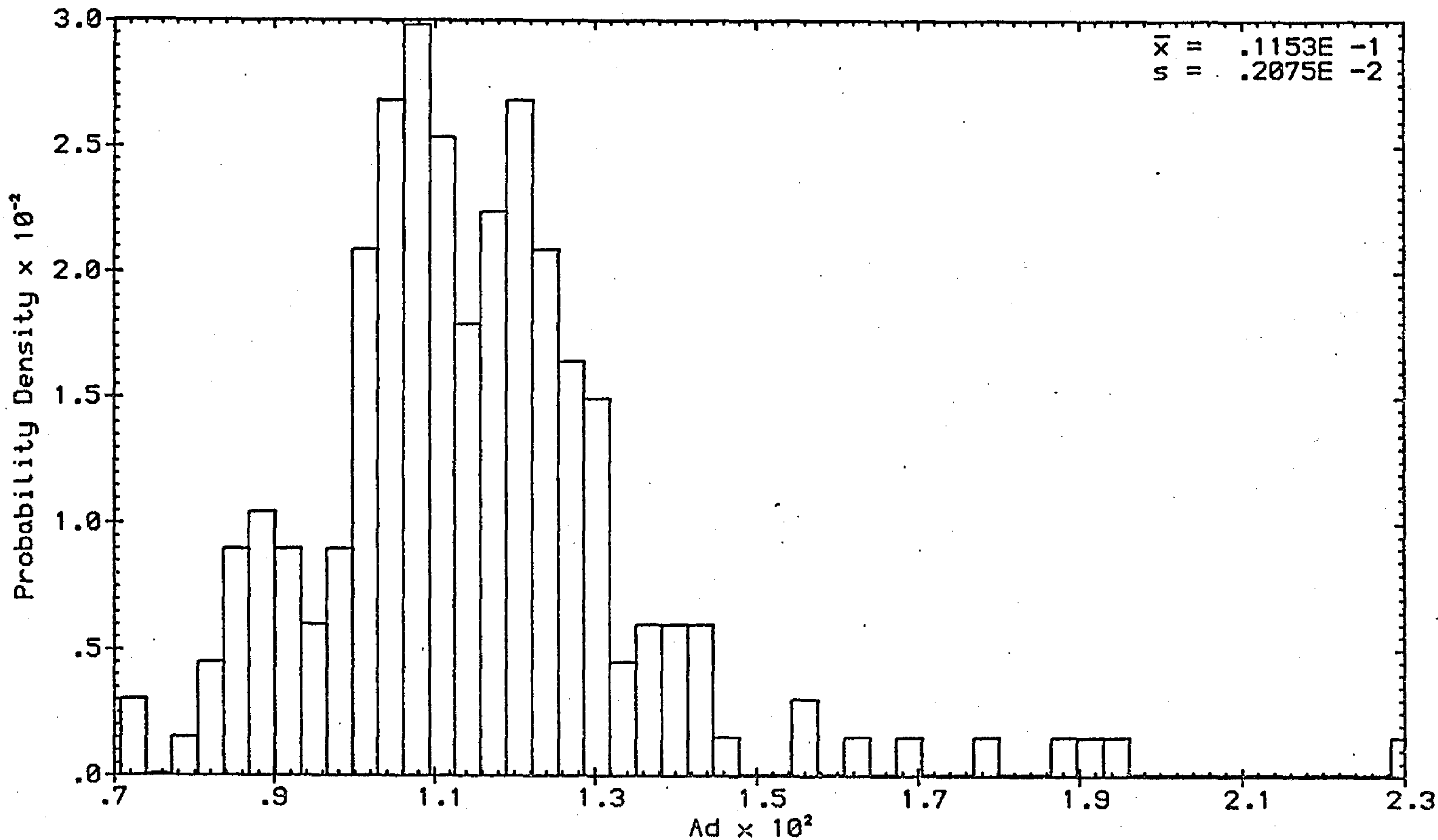


Ad vrs AMBIENT TEMPERATURE FOR CORRECTED DATA

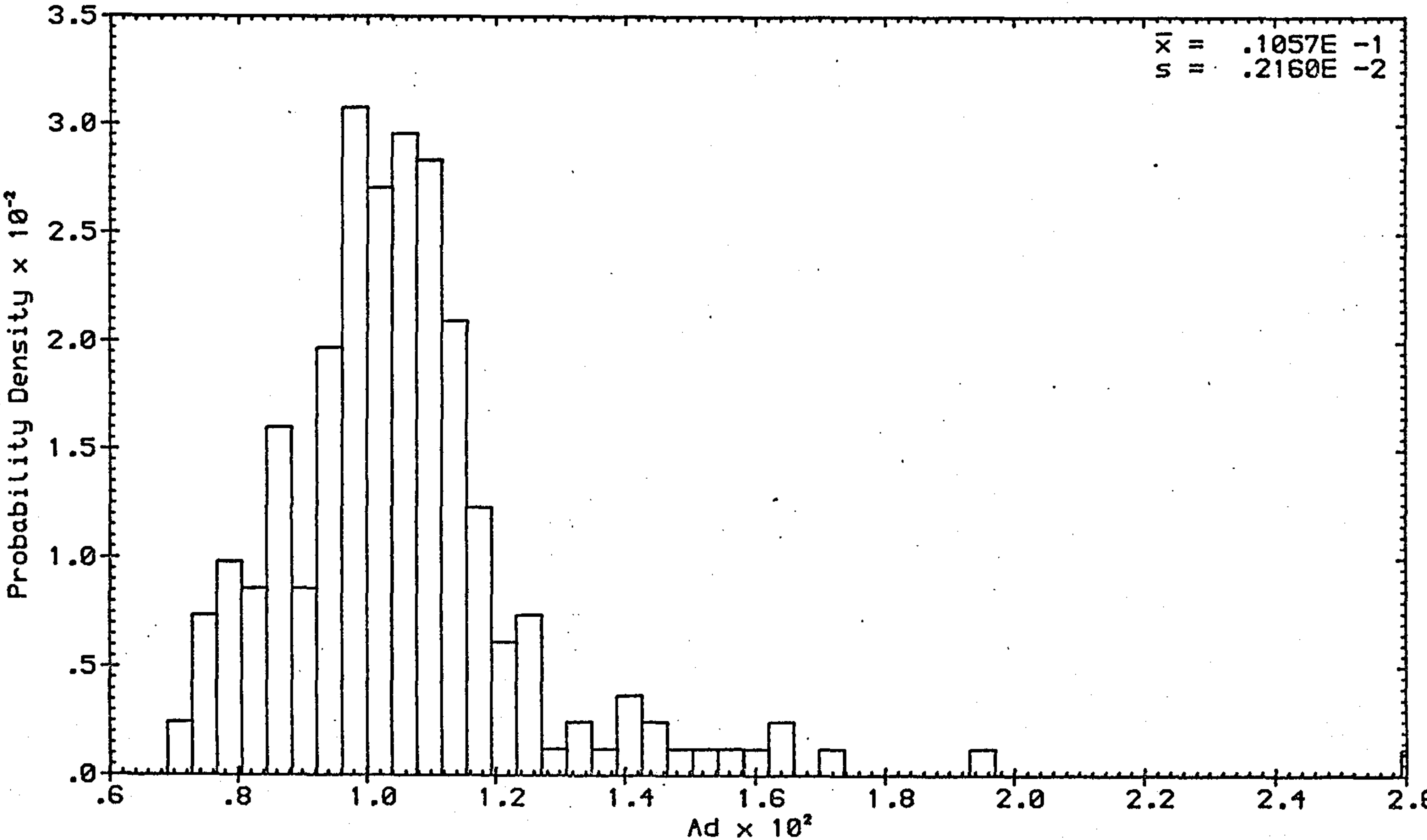


Ad vrs TEST DIRECTION FOR CORRECTED DATA

TuFty.Plot

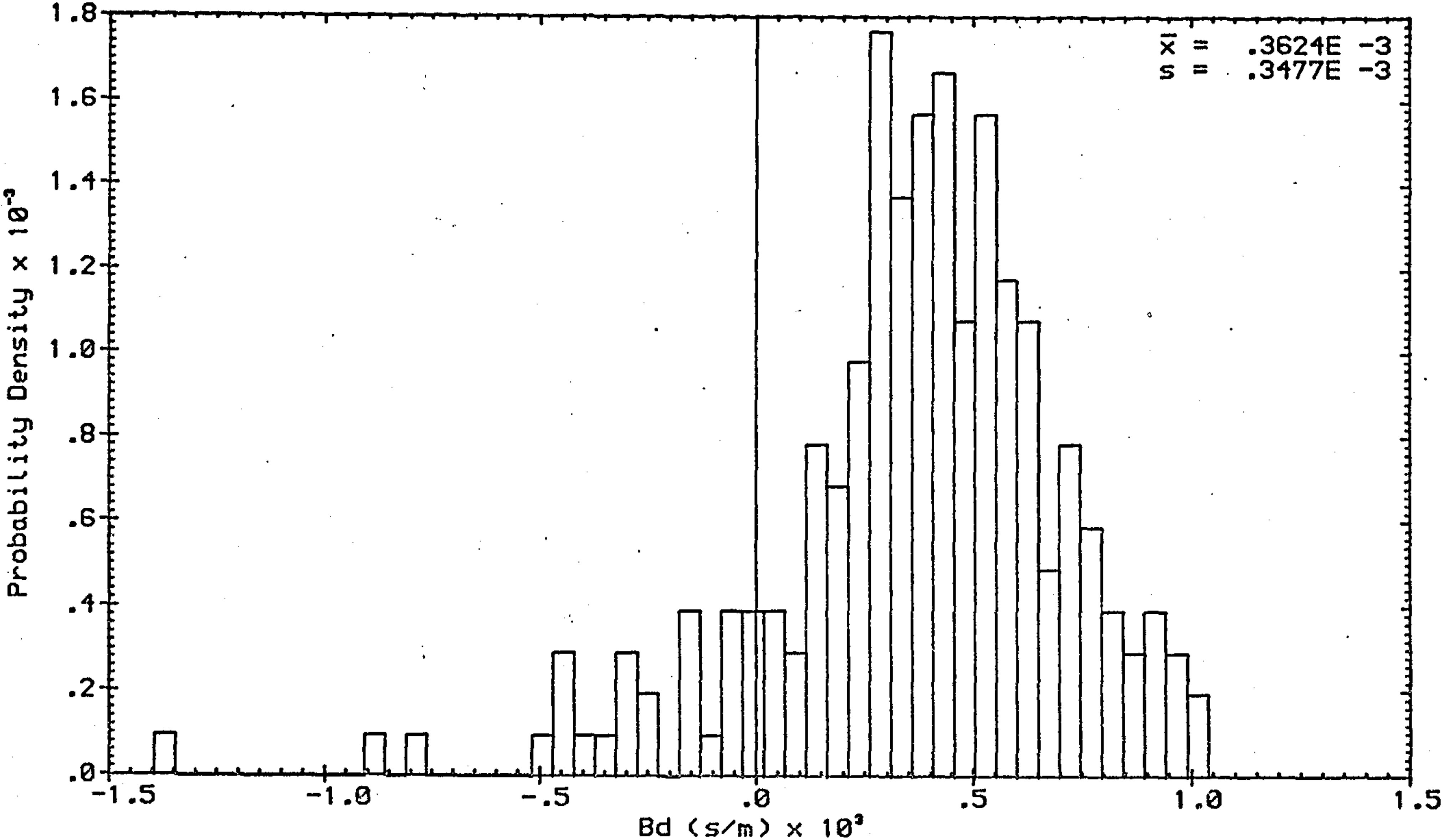


PROBABILITY DENSITY BAR CHART FOR Ad (UNCORRECTED)



PROBABILITY DENSITY BAR CHART FOR Ad (CORRECTED)

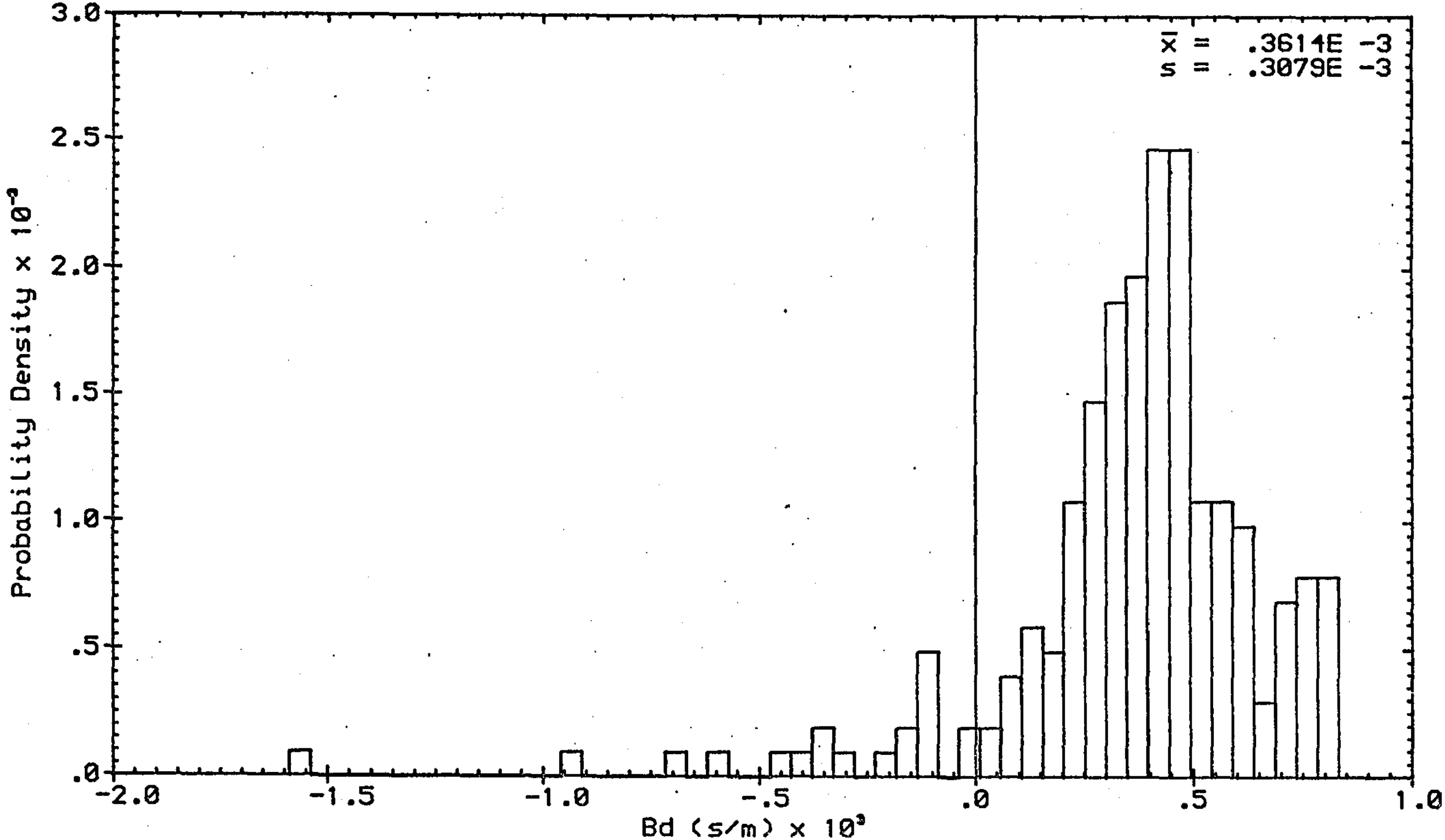
TuFty.Plot



PROBABILITY DENSITY BAR CHART FOR Bd (UNCORRECTED)

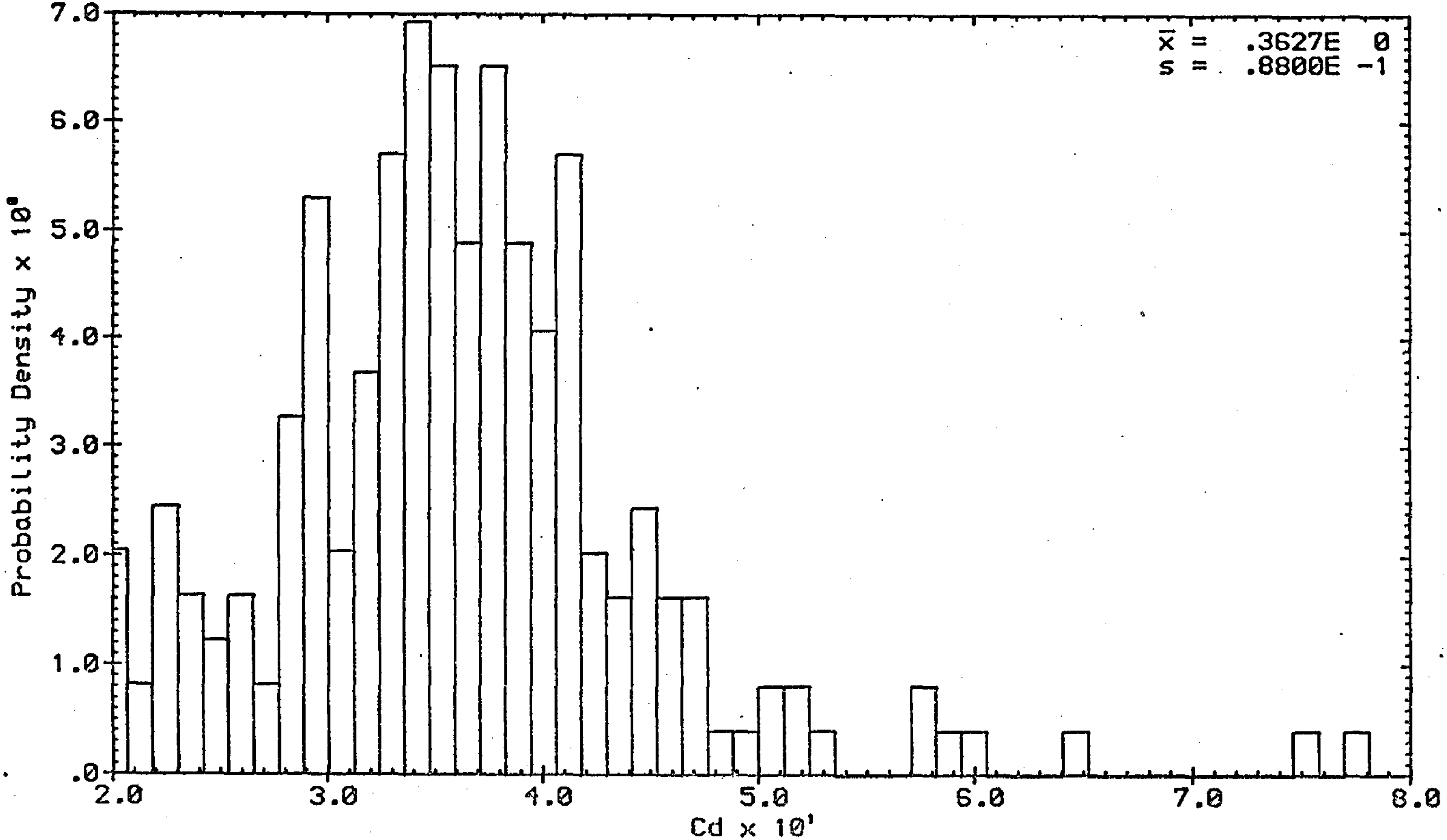
TuFtu.Plot

Fig 3-11



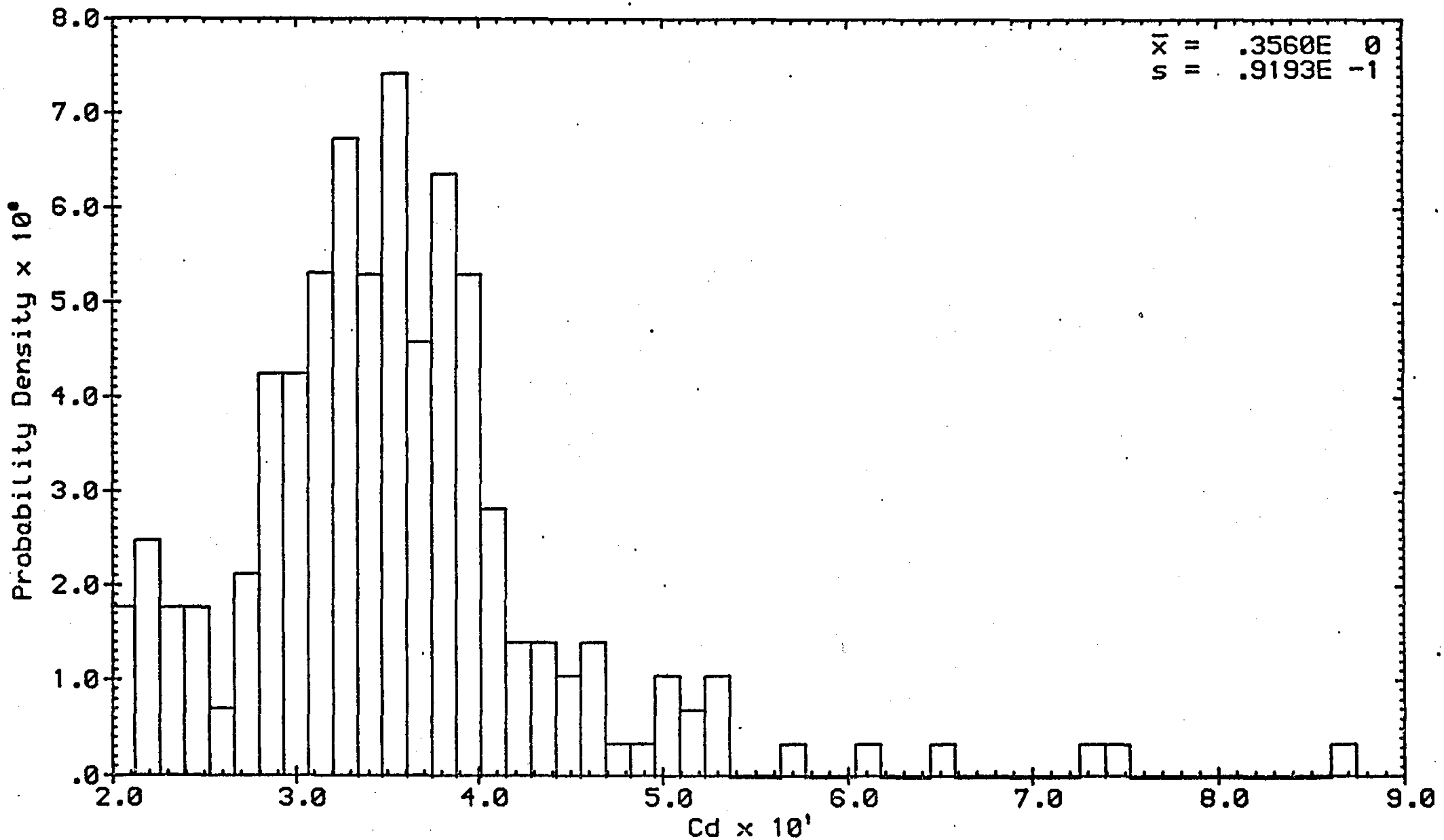
PROBABILITY DENSITY BAR CHART FOR Bd (CORRECTED)

Tufty.Plot



PROBABILITY DENSITY BARCHART FOR Cd (UNCORRECTED)

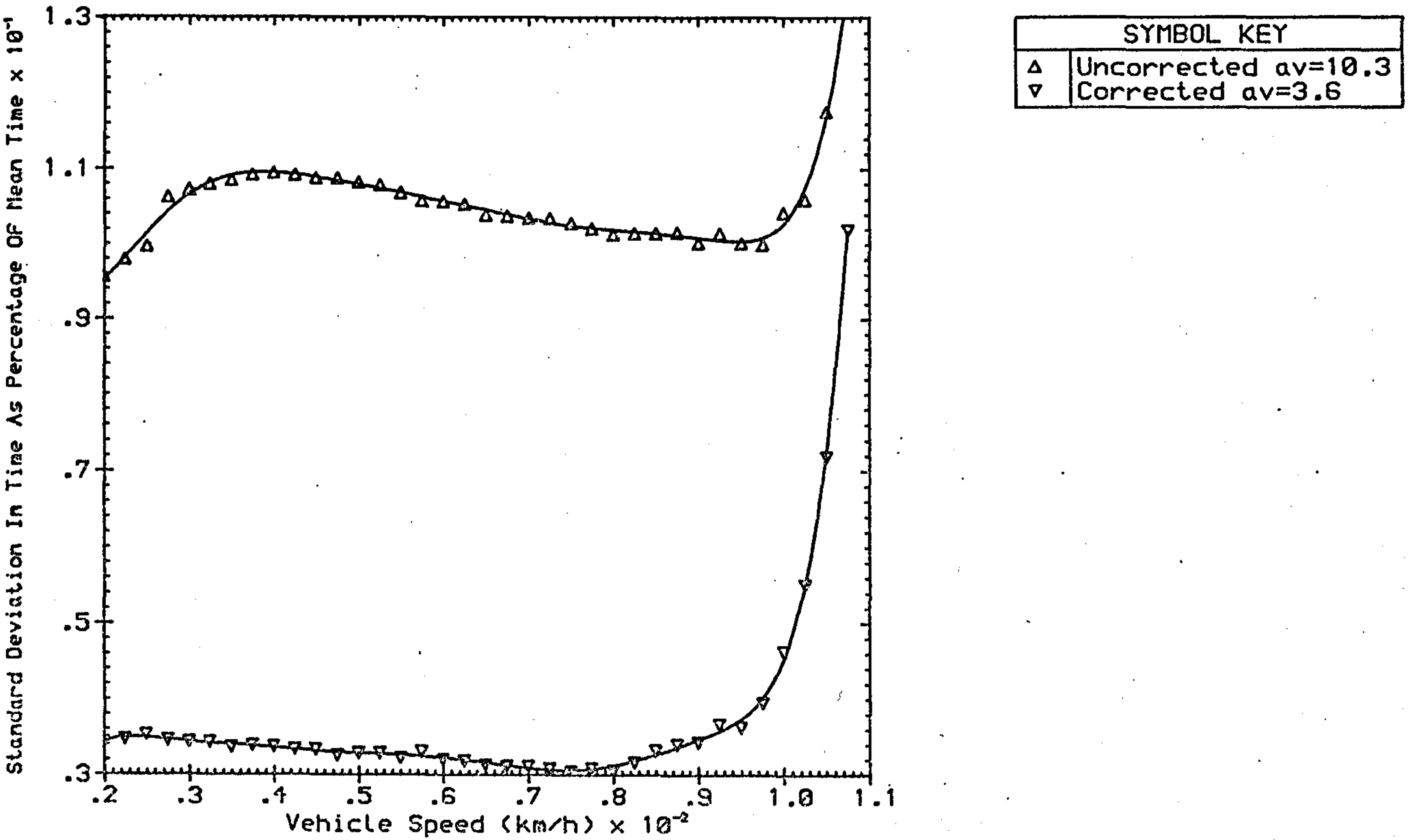
TriFlu.Plot



PROBABILITY DENSITY BAR CHART FOR Cd (CORRECTED)

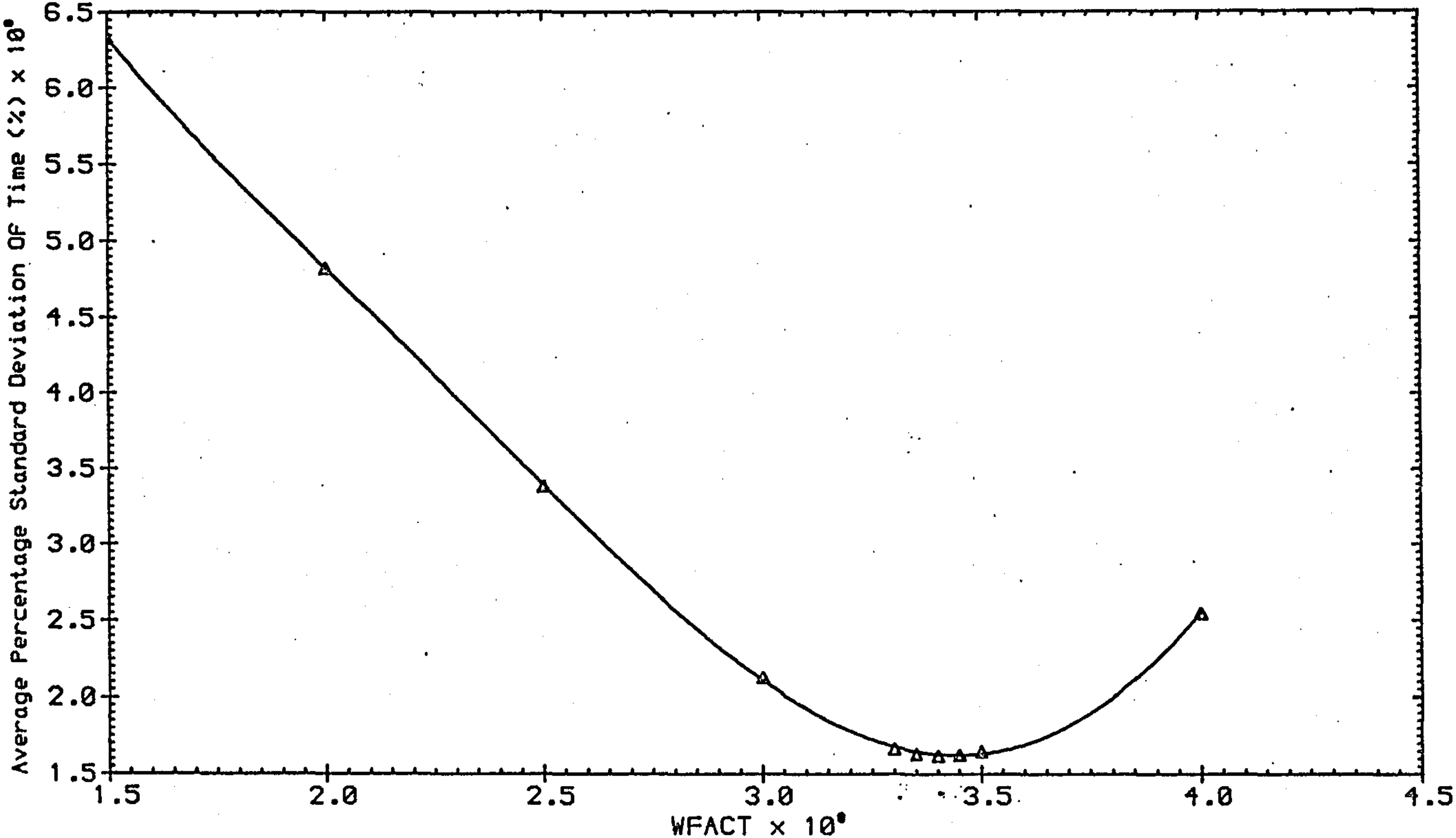
Tufty.Plot

Fig 3.14



COMPARISON OF CORRECTED AND UNCORRECTED SPEED vrs TIME DATA

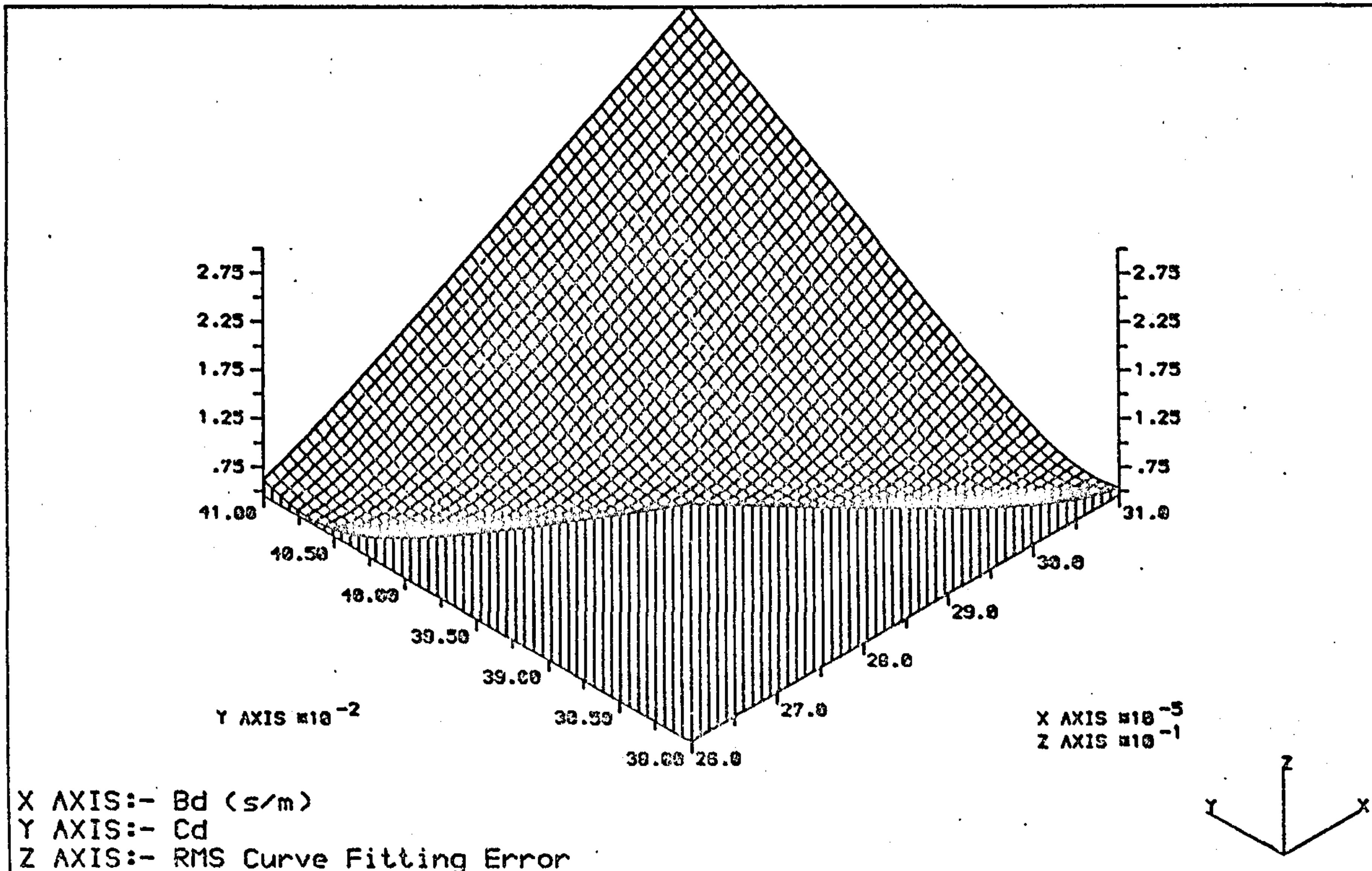
Tufty.Plot



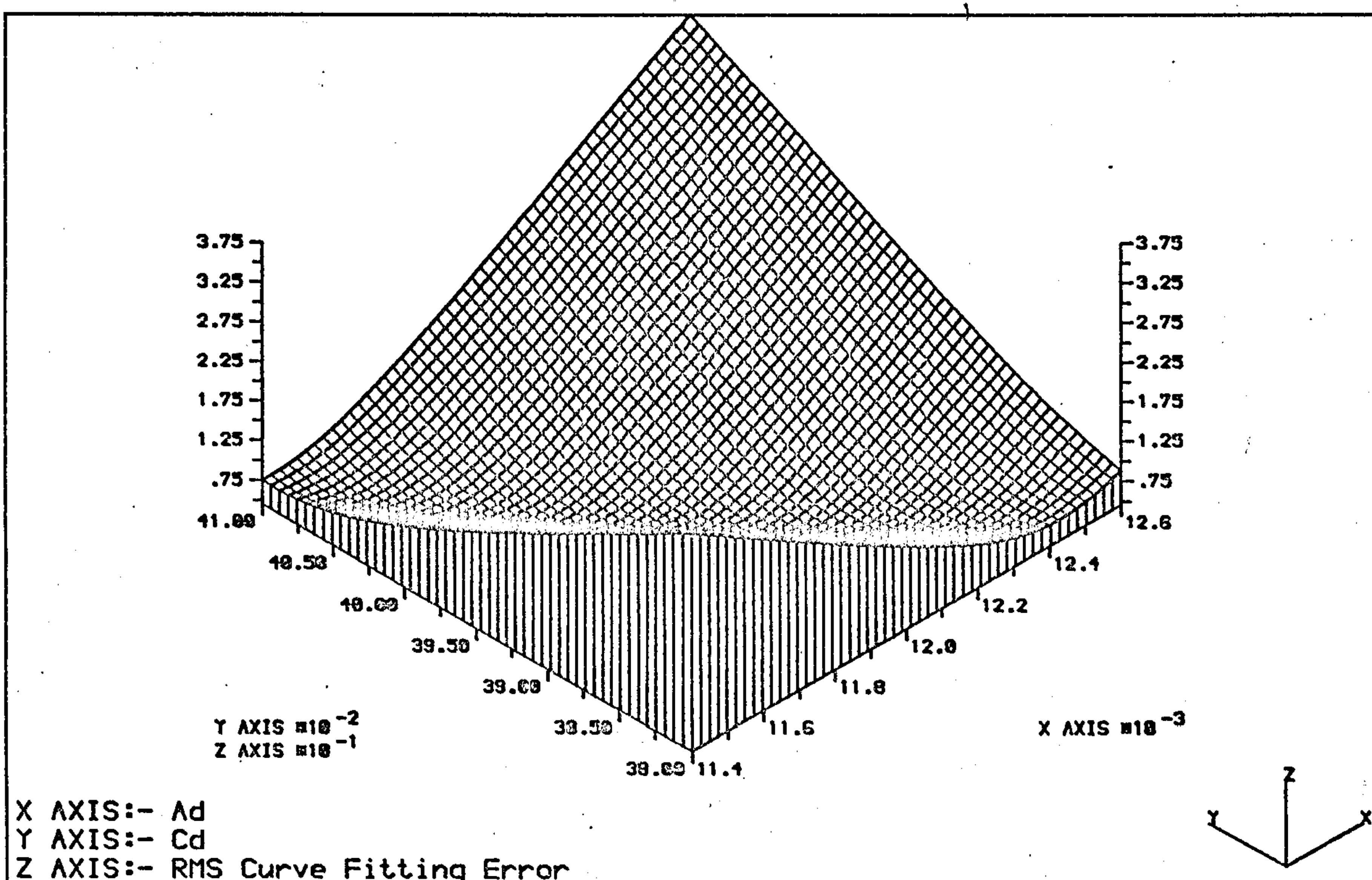
AVERAGE PERCENTAGE STANDARD DEVIATION (TIME) vrs WFACT

Tufty.Plot

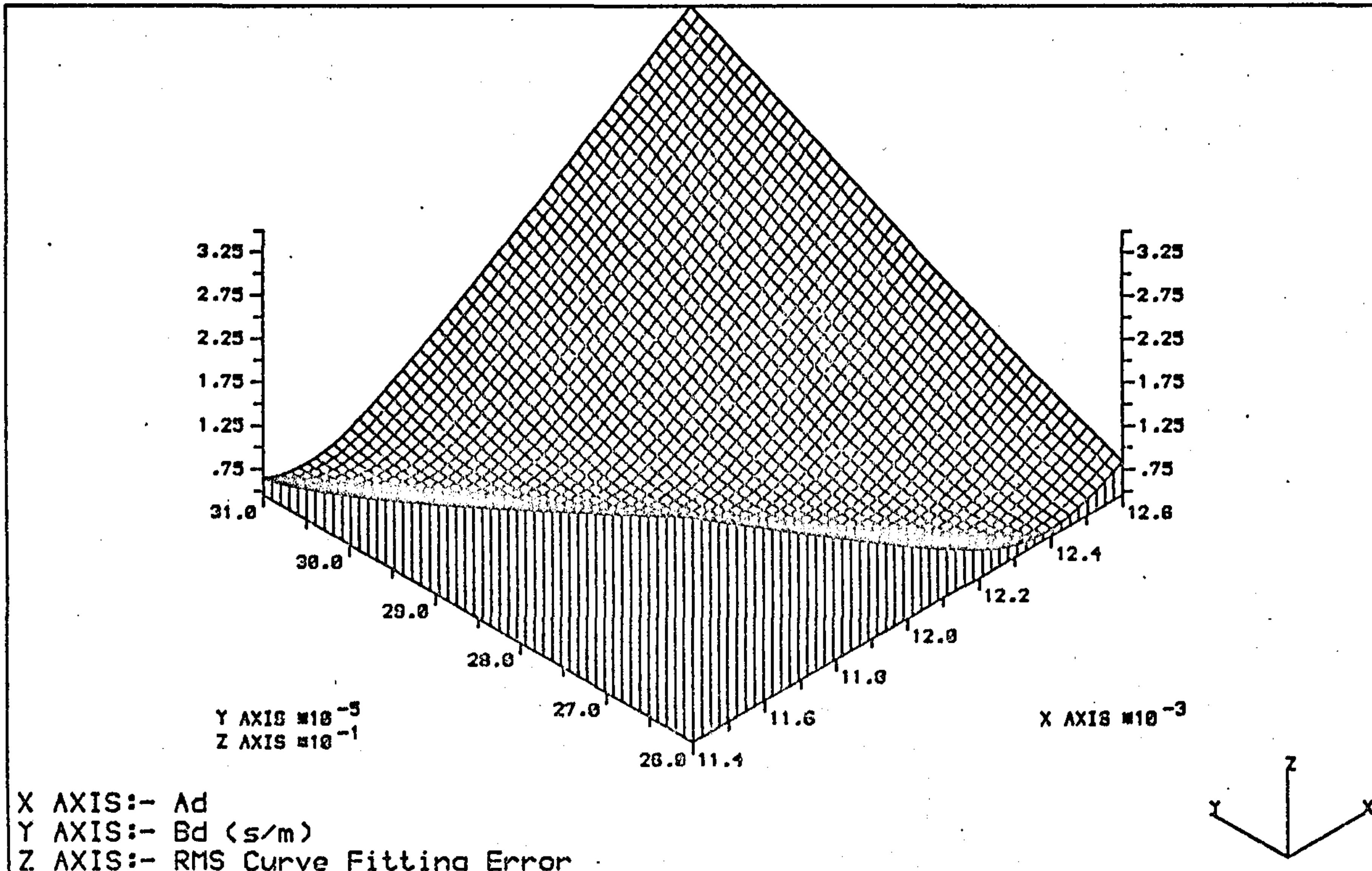
Fig 5.10



ISOMETRIC PROJECTION OF CURVE FITTING ERROR SURFACE (Ad=0.012)

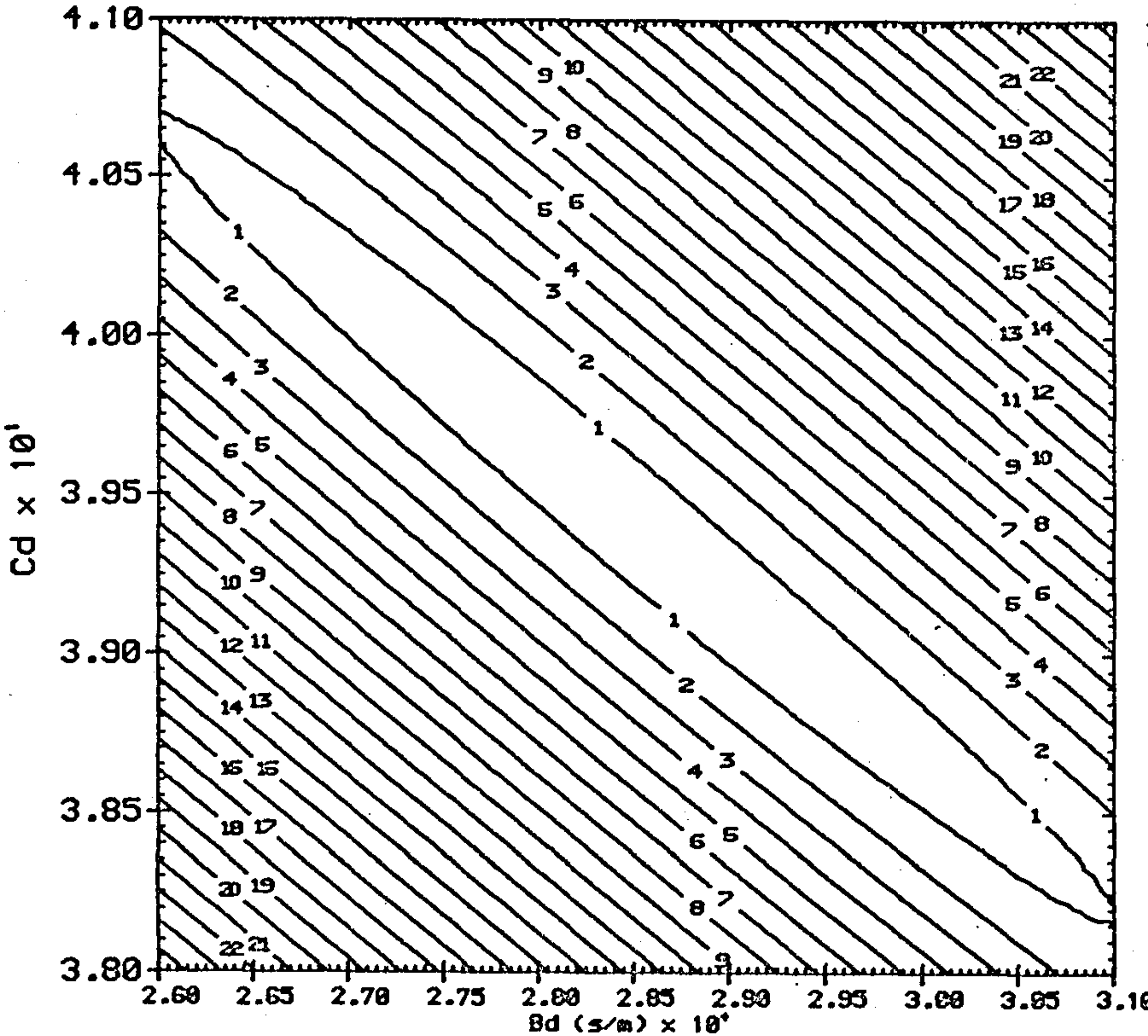


ISOMETRIC PROJECTION OF CURVE FITTING ERROR SURFACE (Bd=0.285E-03)



ISOMETRIC PROJECTION OF CURVE FITTING ERROR SURFACE (Cd=0.394)

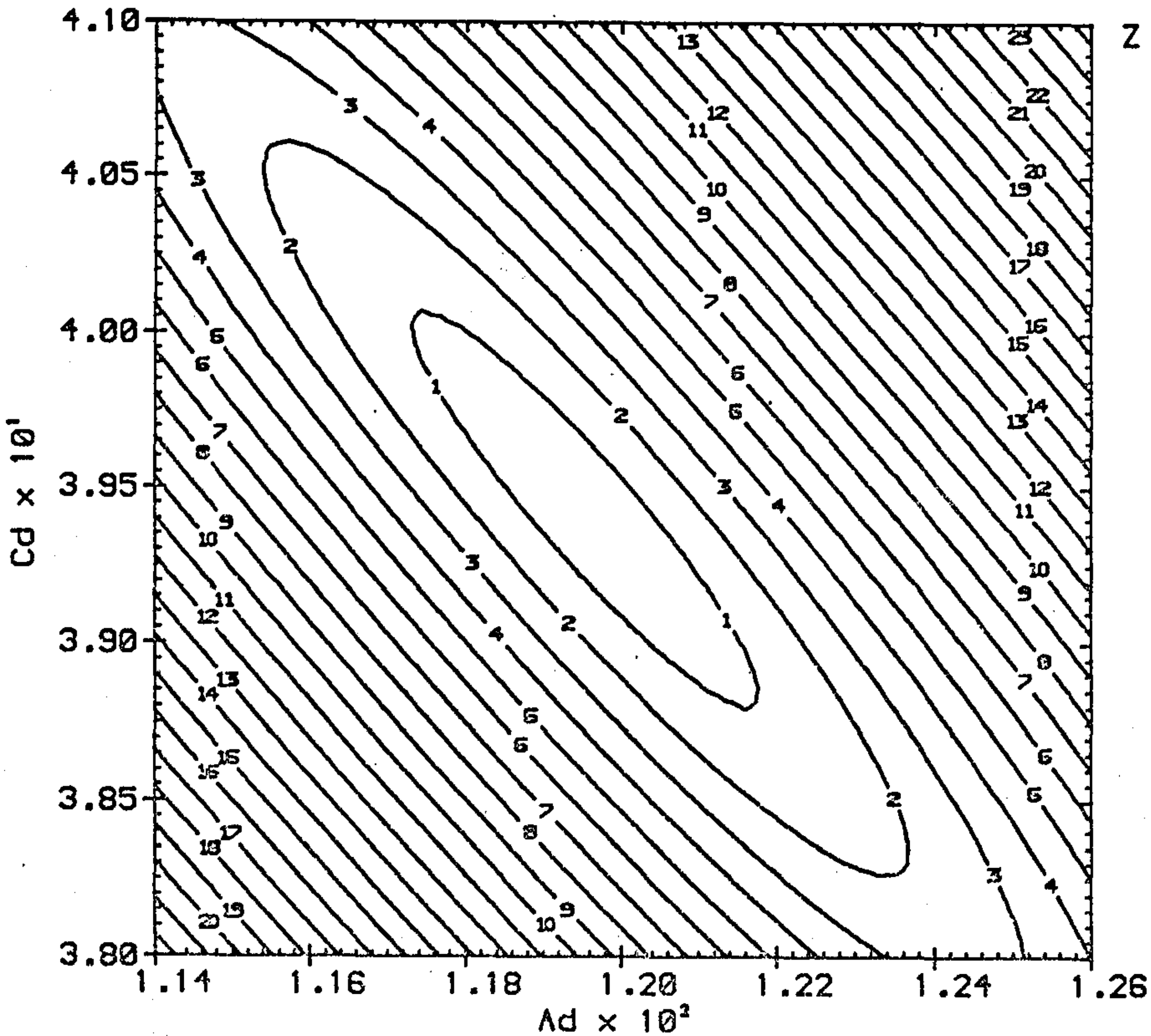
Fig 4.5



Z AXIS:- RMS Curve Fitting Error

CONTOUR KEY	
1	0.5037E-01
2	0.6046E-01
3	0.7055E-01
4	0.8064E-01
5	0.9074E-01
6	0.1008E+00
7	0.1109E+00
8	0.1210E+00
9	0.1311E+00
10	0.1412E+00
11	0.1513E+00
12	0.1614E+00
13	0.1715E+00
14	0.1816E+00
15	0.1917E+00
16	0.2018E+00
17	0.2118E+00
18	0.2219E+00
19	0.2320E+00
20	0.2421E+00
21	0.2522E+00
22	0.2623E+00
23	0.2724E+00
24	0.2825E+00
25	0.2926E+00

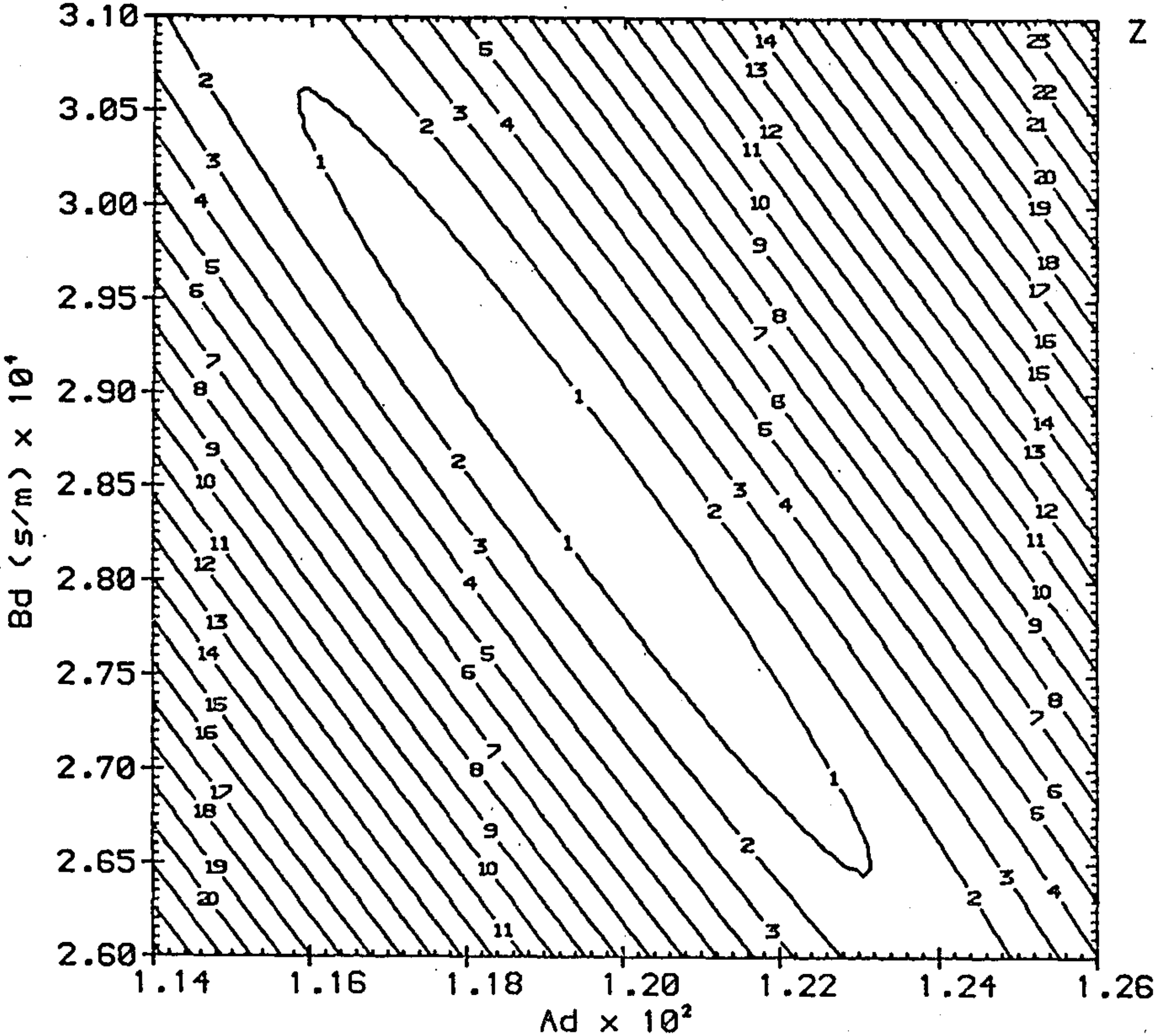
CONTOUR MAP OF CURVE FITTING ERROR SURFACE (Ad=0.012)



Z AXIS:- RMS Curve Fitting Error

CONTOUR KEY	
1	0.5149E-01
2	0.6385E-01
3	0.7621E-01
4	0.8857E-01
5	0.1009E+00
6	0.1133E+00
7	0.1257E+00
8	0.1380E+00
9	0.1504E+00
10	0.1627E+00
11	0.1751E+00
12	0.1875E+00
13	0.1998E+00
14	0.2122E+00
15	0.2246E+00
16	0.2369E+00
17	0.2493E+00
18	0.2616E+00
19	0.2740E+00
20	0.2864E+00
21	0.2987E+00
22	0.3111E+00
23	0.3234E+00
24	0.3358E+00
25	0.3482E+00

CONTOUR MAP OF CURVE FITTING ERROR SURFACE (Bd=0.285E-03)

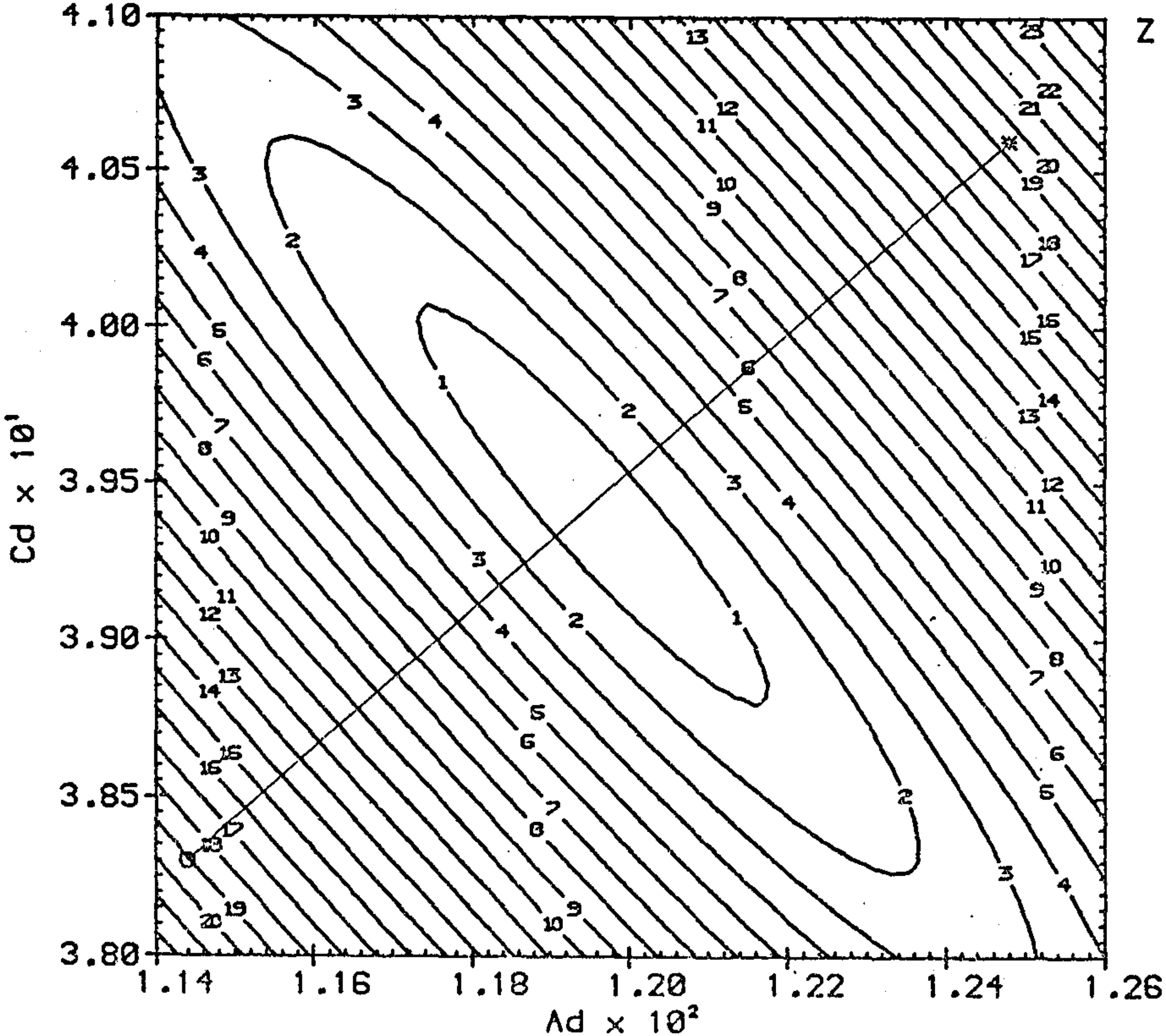


Z AXIS:- RMS Curve Fitting Error

CONTOUR KEY	
1	0.5093E-01
2	0.6219E-01
3	0.7346E-01
4	0.8472E-01
5	0.9598E-01
6	0.1072E+00
7	0.1185E+00
8	0.1298E+00
9	0.1410E+00
10	0.1523E+00
11	0.1636E+00
12	0.1748E+00
13	0.1861E+00
14	0.1973E+00
15	0.2086E+00
16	0.2199E+00
17	0.2311E+00
18	0.2424E+00
19	0.2537E+00
20	0.2649E+00
21	0.2762E+00
22	0.2874E+00
23	0.2987E+00
24	0.3100E+00
25	0.3212E+00

CONTOUR MAP OF CURVE FITTING ERROR SURFACE (Cd=0.394)

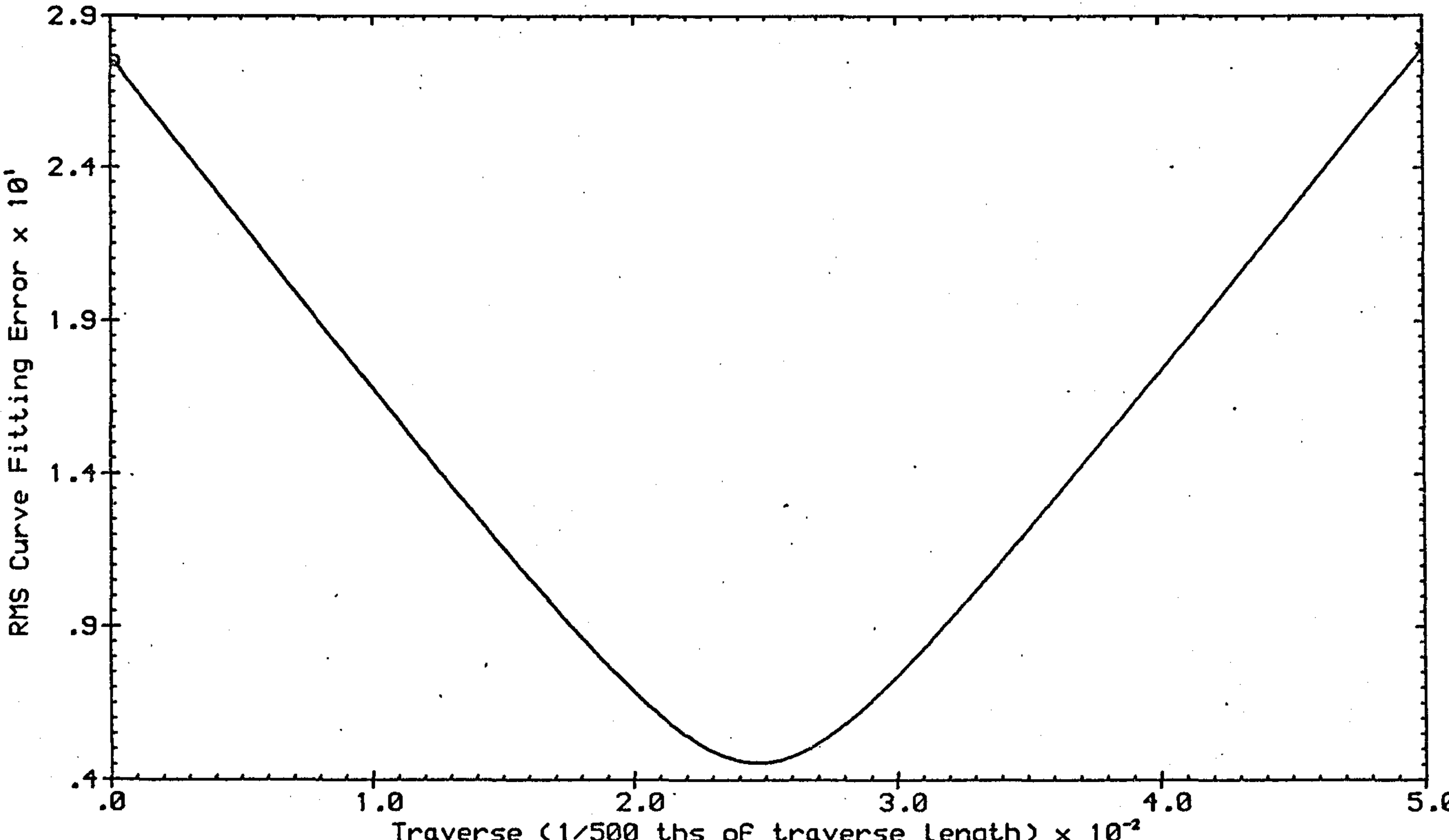
Fig 4.6



Z AXIS:- RMS Curve Fitting Error

CONTOUR KEY	
1	0.5149E-01
2	0.6385E-01
3	0.7621E-01
4	0.8857E-01
5	0.1009E+00
6	0.1133E+00
7	0.1257E+00
8	0.1380E+00
9	0.1504E+00
10	0.1627E+00
11	0.1751E+00
12	0.1875E+00
13	0.1998E+00
14	0.2122E+00
15	0.2246E+00
16	0.2369E+00
17	0.2493E+00
18	0.2616E+00
19	0.2740E+00
20	0.2864E+00
21	0.2987E+00
22	0.3111E+00
23	0.3234E+00
24	0.3358E+00
25	0.3482E+00

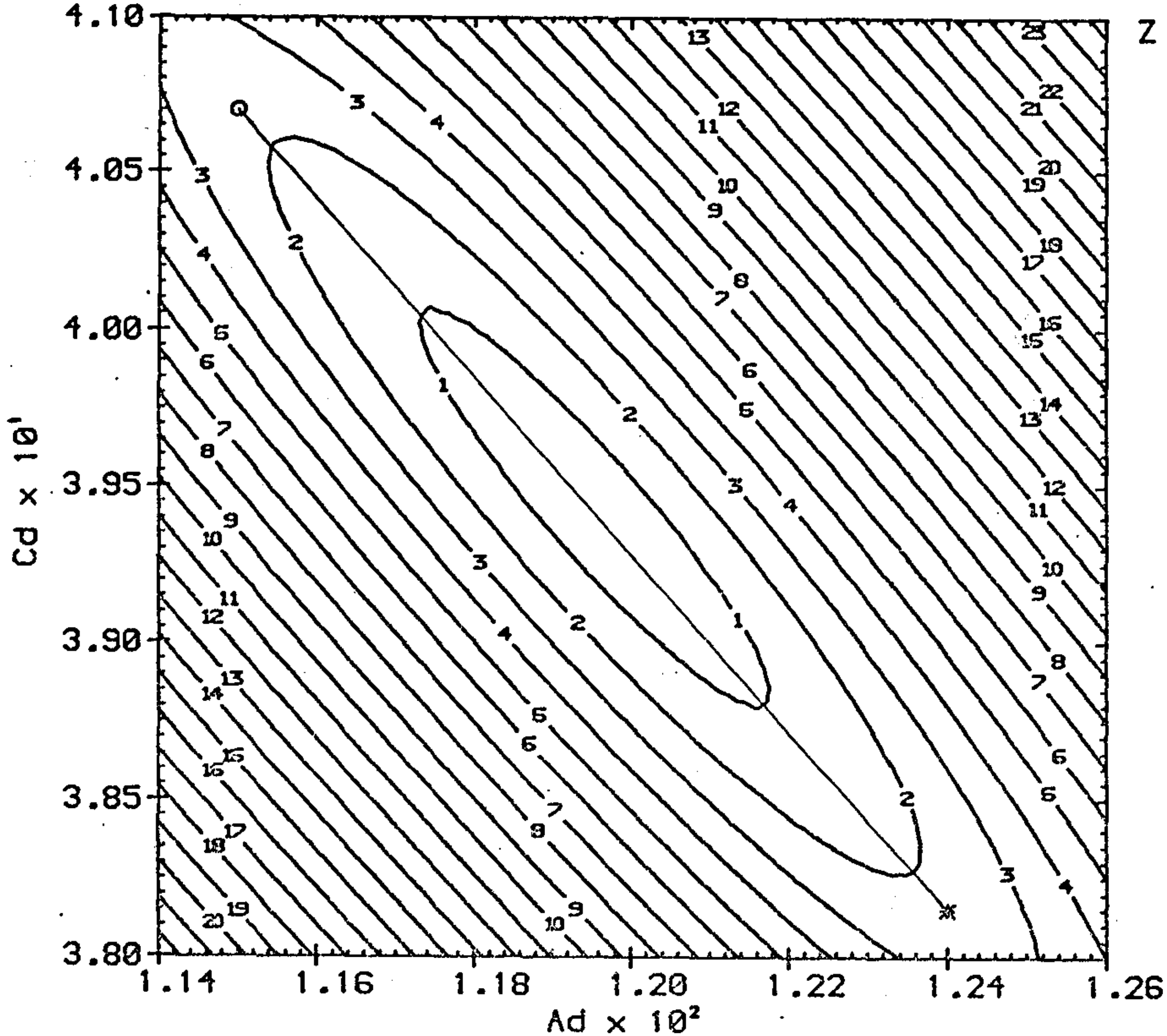
POSITION OF SECTION 'A' THROUGH ERROR SURFACE (Bd=0.285E-03)



SECTION 'A' THROUGH ERROR SURFACE (Bd=0.285E-03)

Fig 4.6

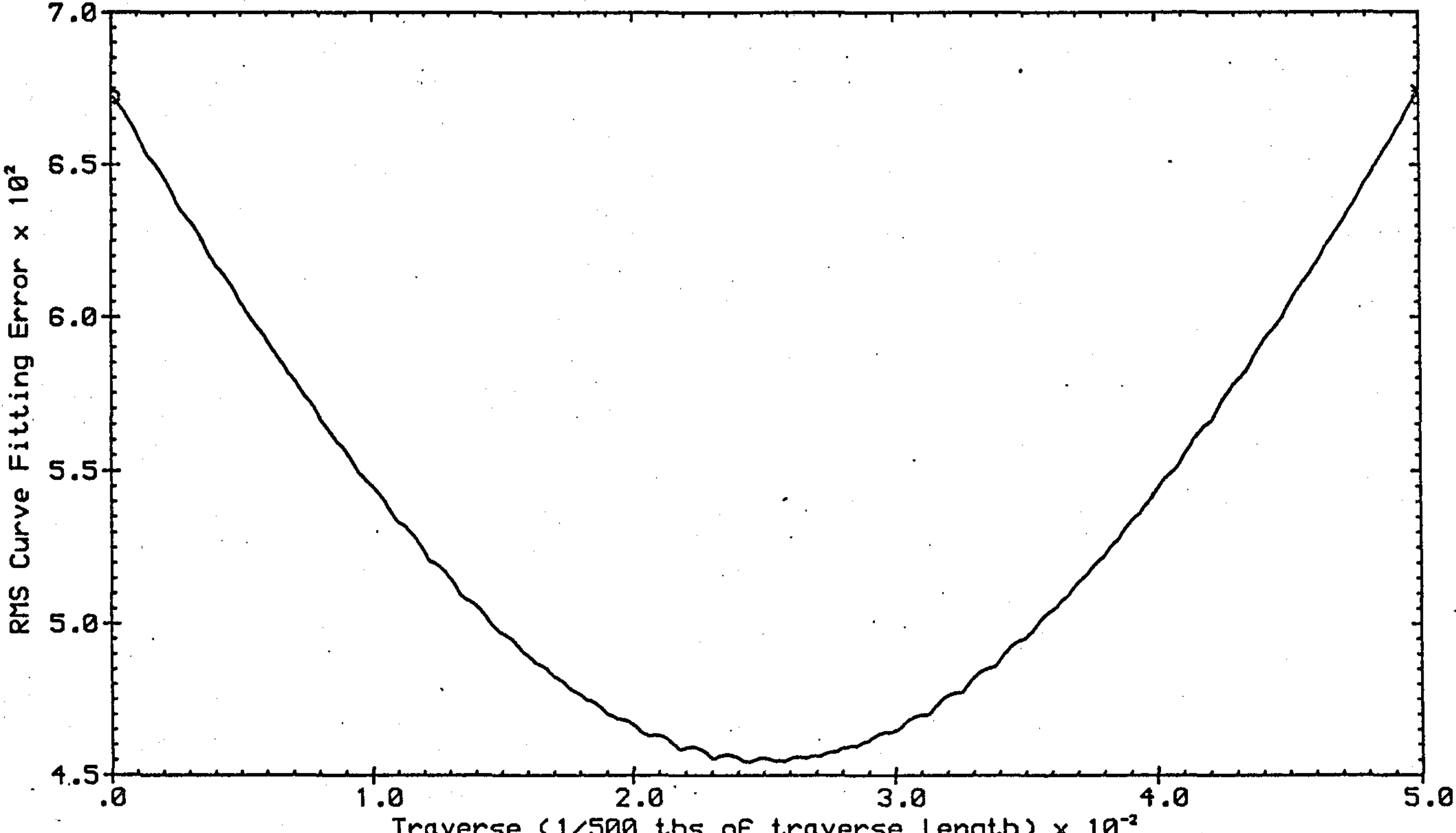
Tufty Plot



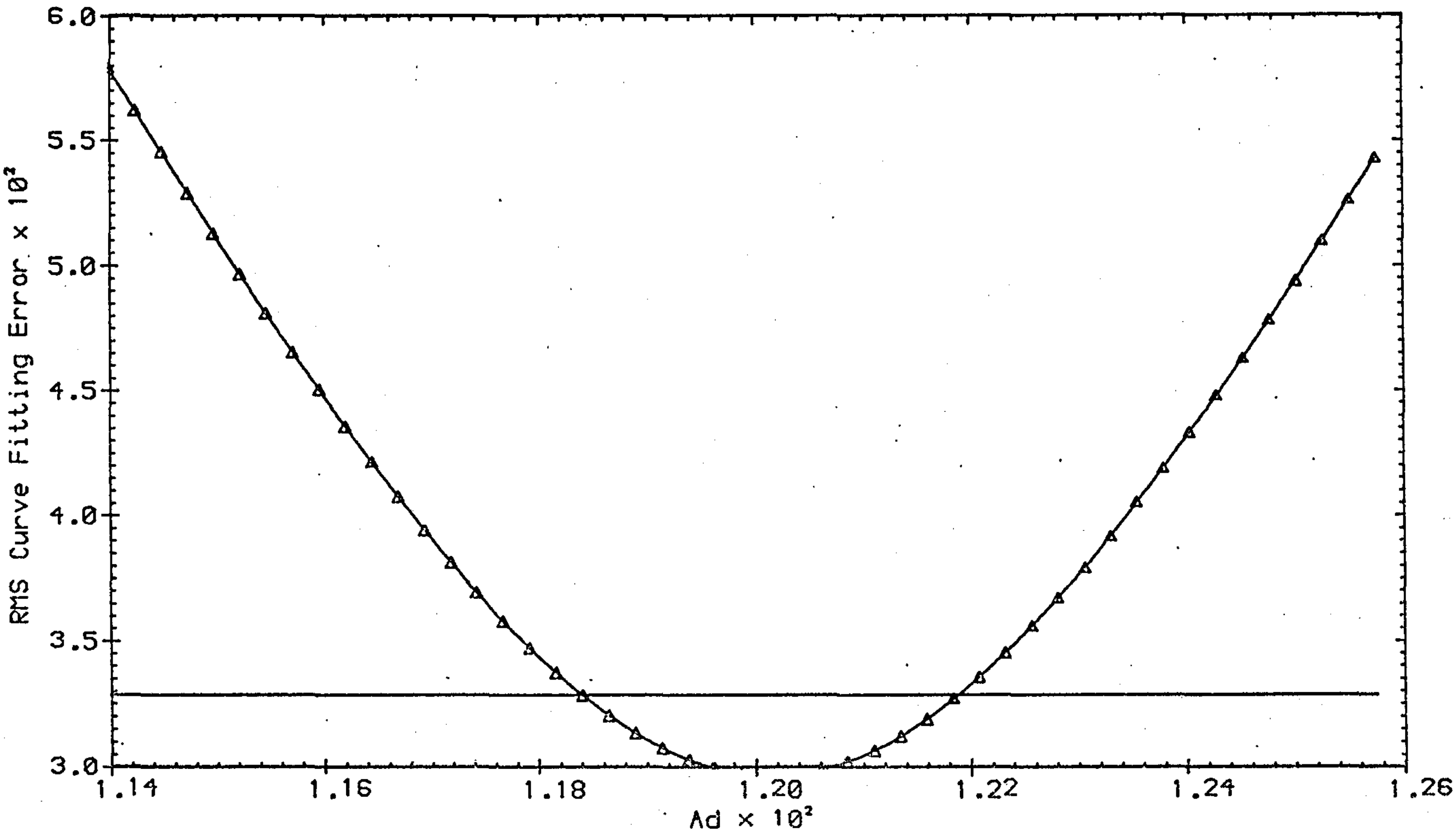
Z AXIS:- RMS Curve Fitting Error

CONTOUR KEY	
1	0.5149E-01
2	0.6385E-01
3	0.7621E-01
4	0.8857E-01
5	0.1009E+00
6	0.1133E+00
7	0.1257E+00
8	0.1380E+00
9	0.1504E+00
10	0.1627E+00
11	0.1751E+00
12	0.1875E+00
13	0.1998E+00
14	0.2122E+00
15	0.2246E+00
16	0.2369E+00
17	0.2493E+00
18	0.2616E+00
19	0.2740E+00
20	0.2864E+00
21	0.2987E+00
22	0.3111E+00
23	0.3234E+00
24	0.3358E+00
25	0.3482E+00

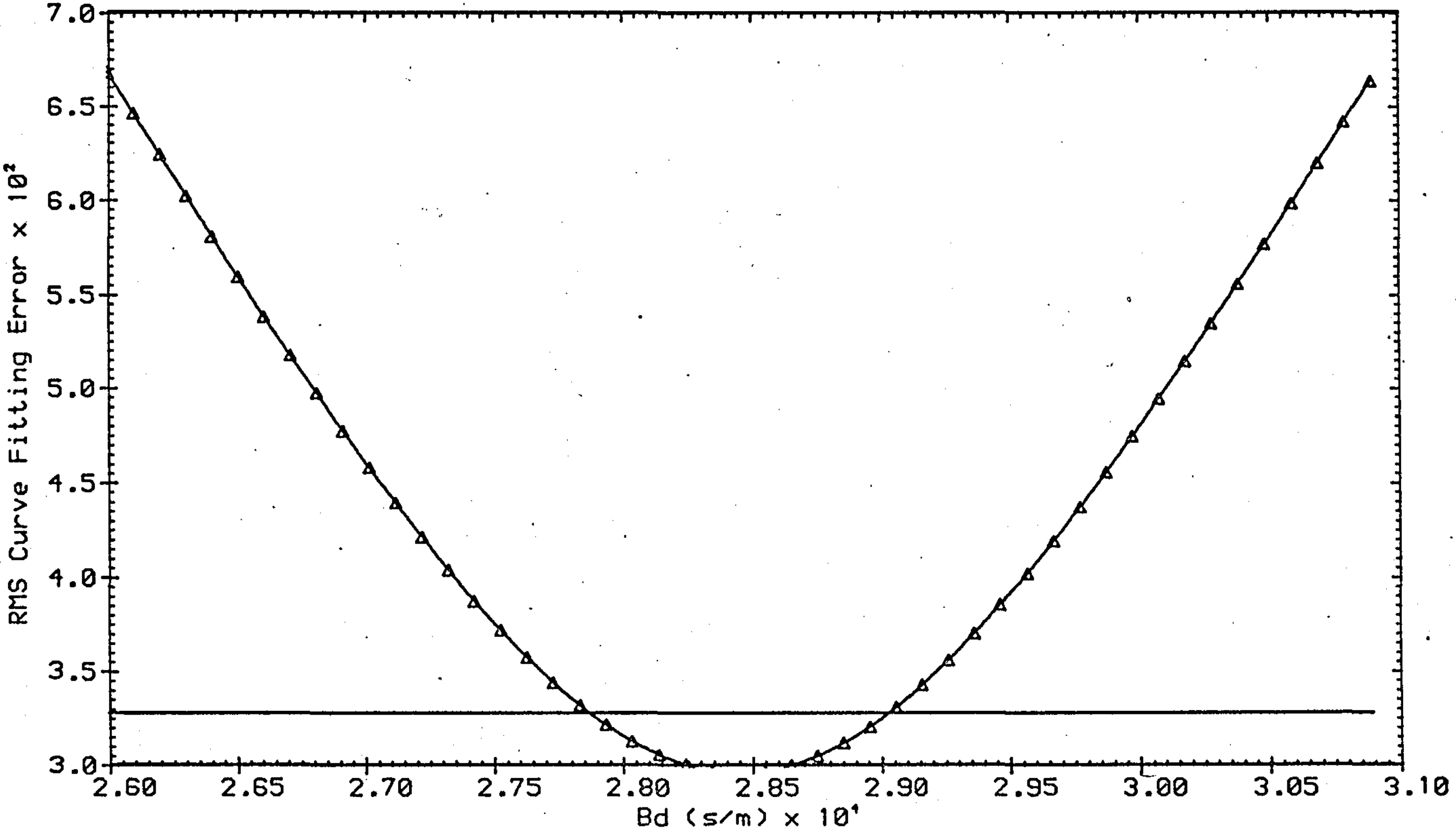
POSITION OF SECTION 'B' THROUGH ERROR SURFACE (Bd=0.285E-03)



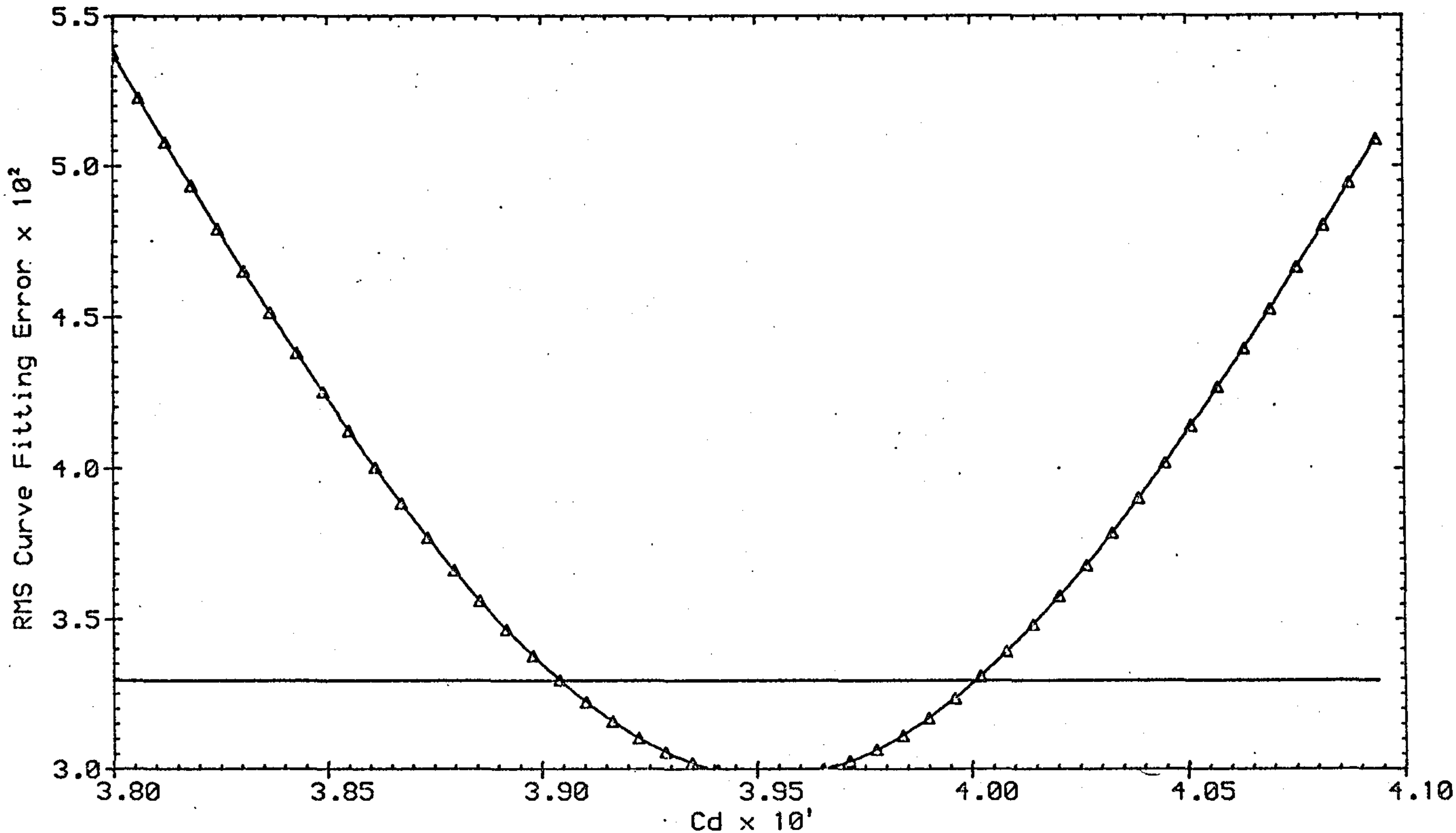
SECTION 'B' THROUGH ERROR SURFACE (Bd=0.285E-03)



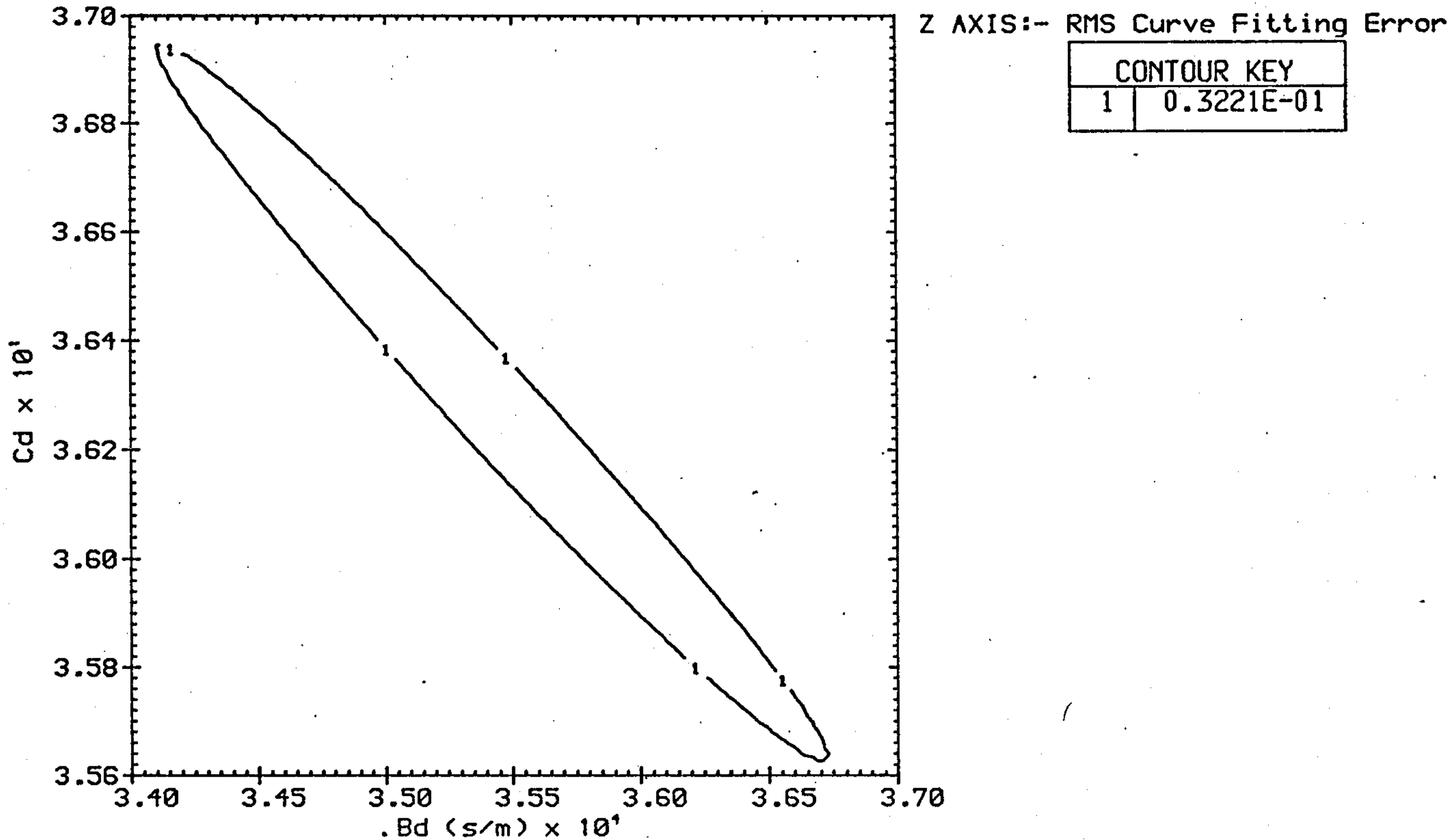
95% CONFIDENCE LIMITS OF Ad FOR RMS=0.02970 AT MINIMUM (nu=1)



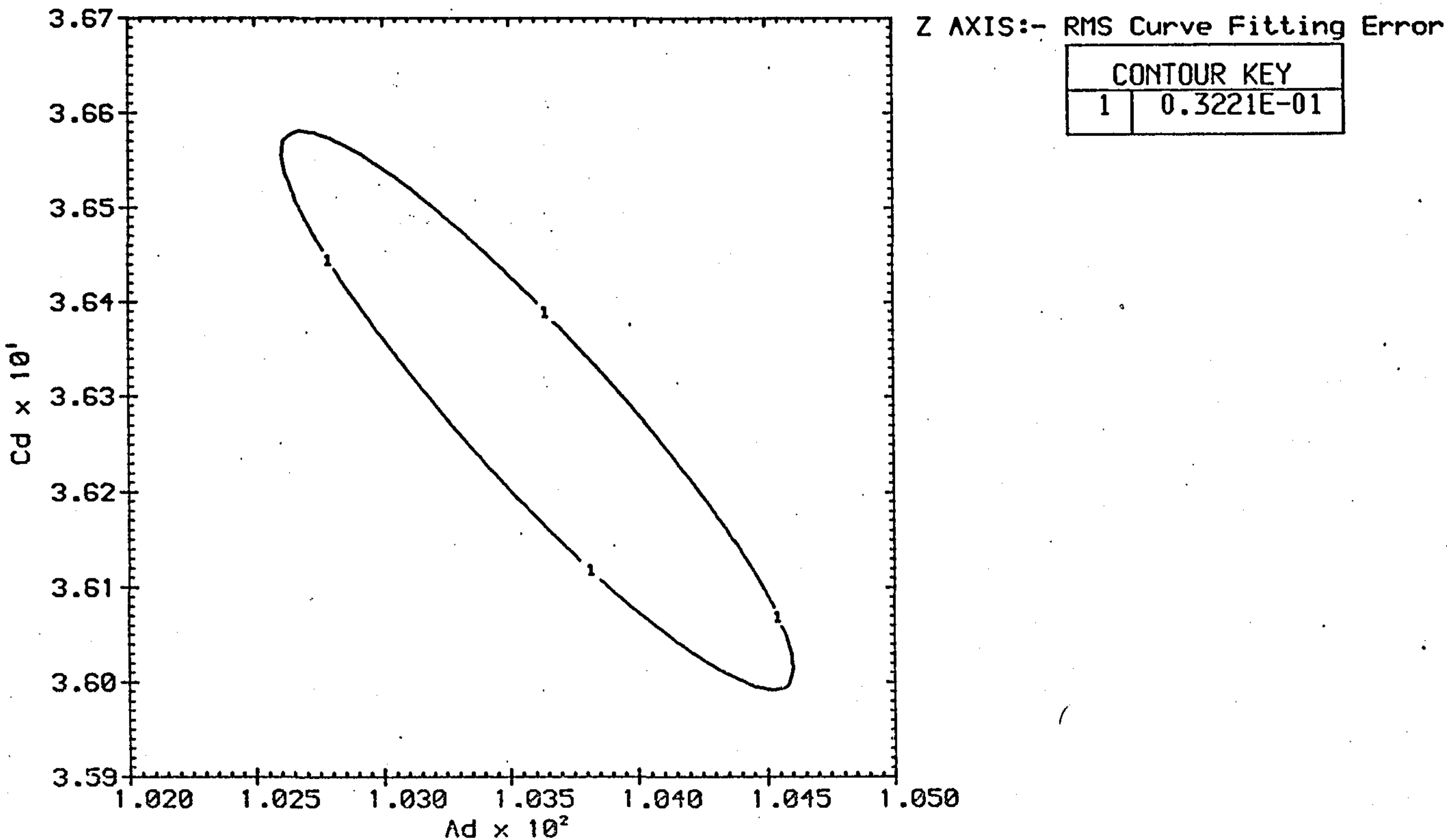
95% CONFIDENCE LIMITS OF Bd FOR RMS=0.02970 AT MINIMUM (nu=1)



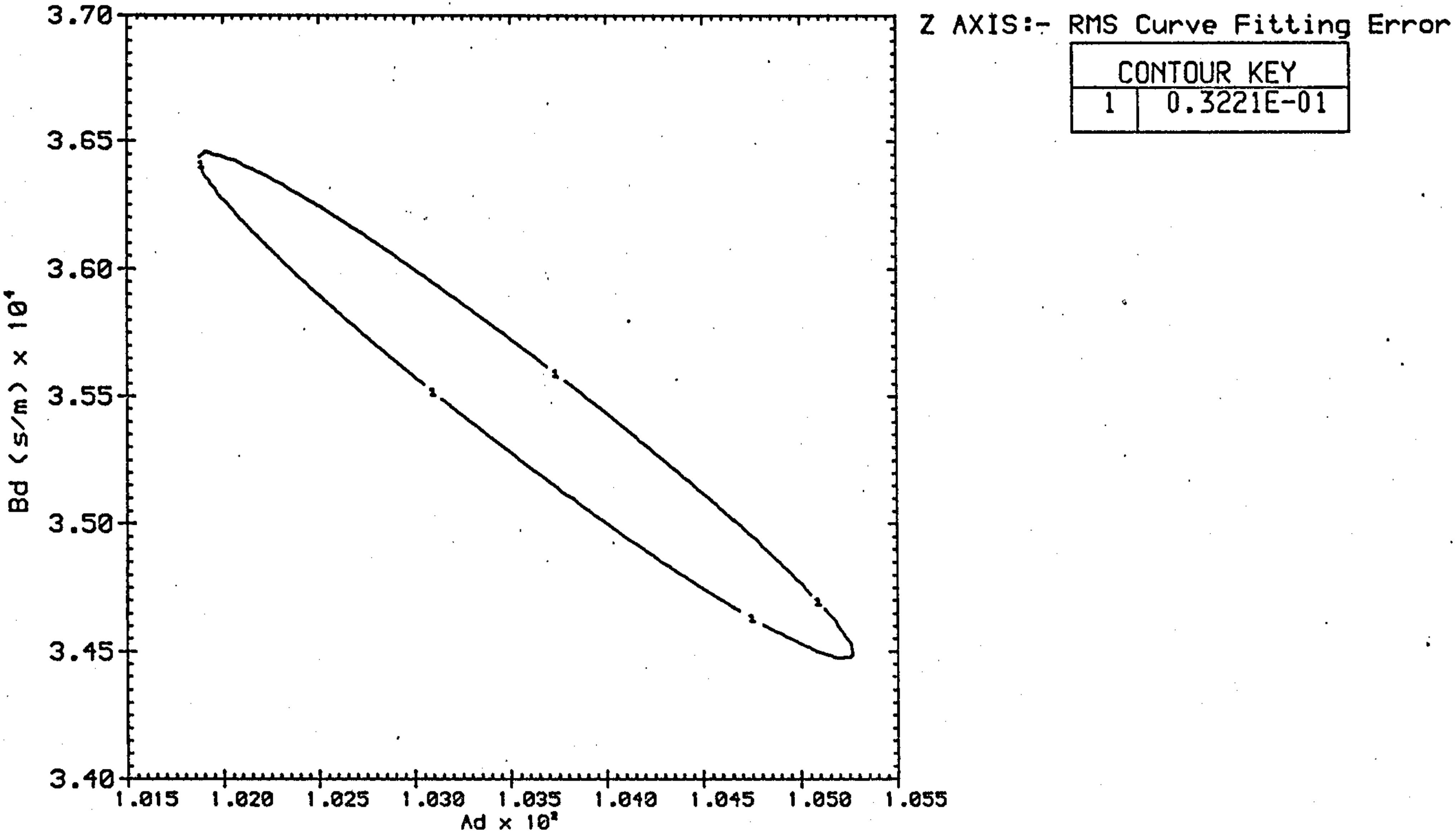
95% CONFIDENCE LIMITS OF Cd FOR RMS=0.02970 AT MINIMUM (nu=1)



95% CONFIDENCE LIMITS FOR RMS=0.02970 AT MINIMUM (nu=2, Ad=const)



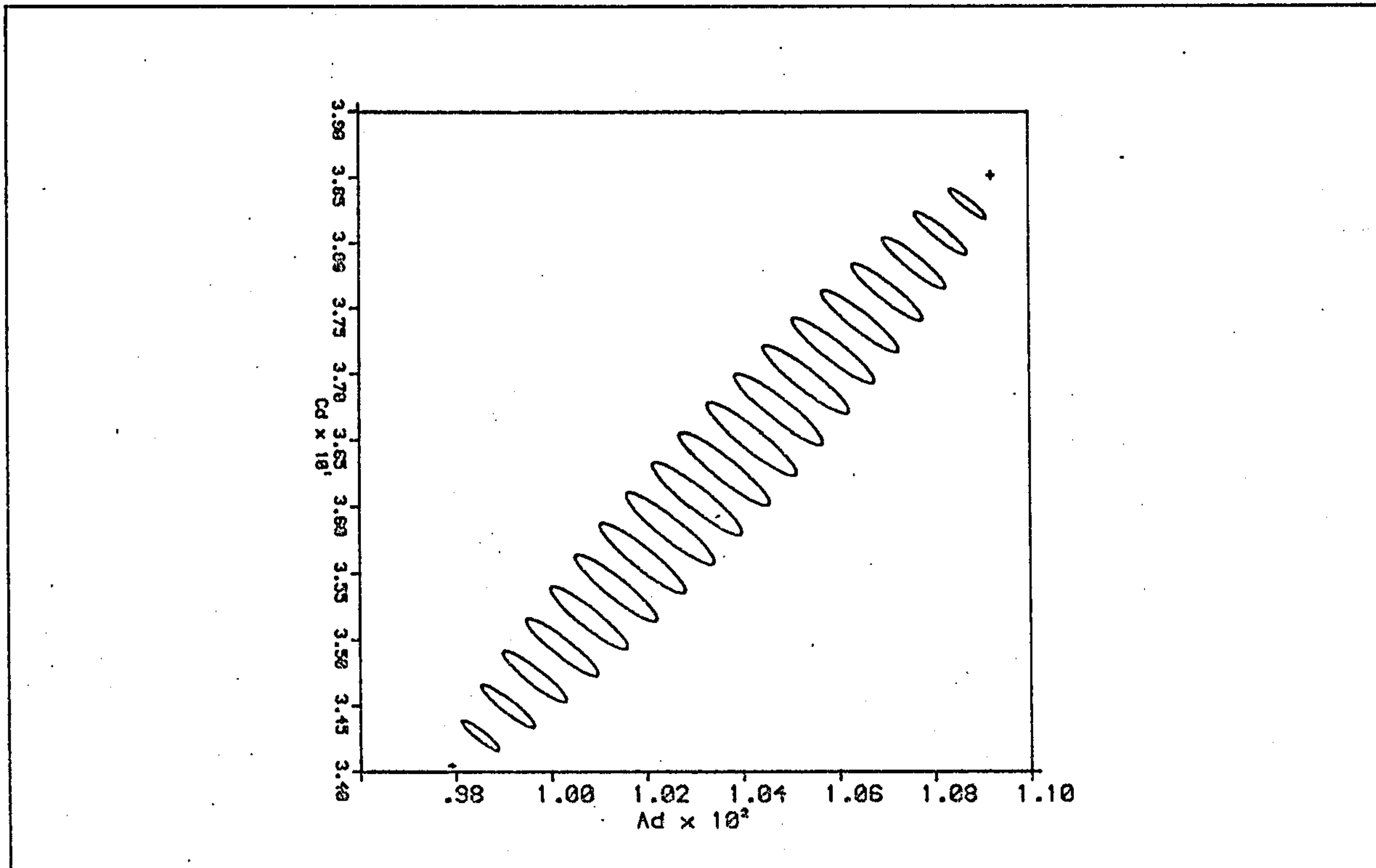
95% CONFIDENCE LIMITS FOR RMS=0.02970 AT MINIMUM (nu=2, Bd=const)



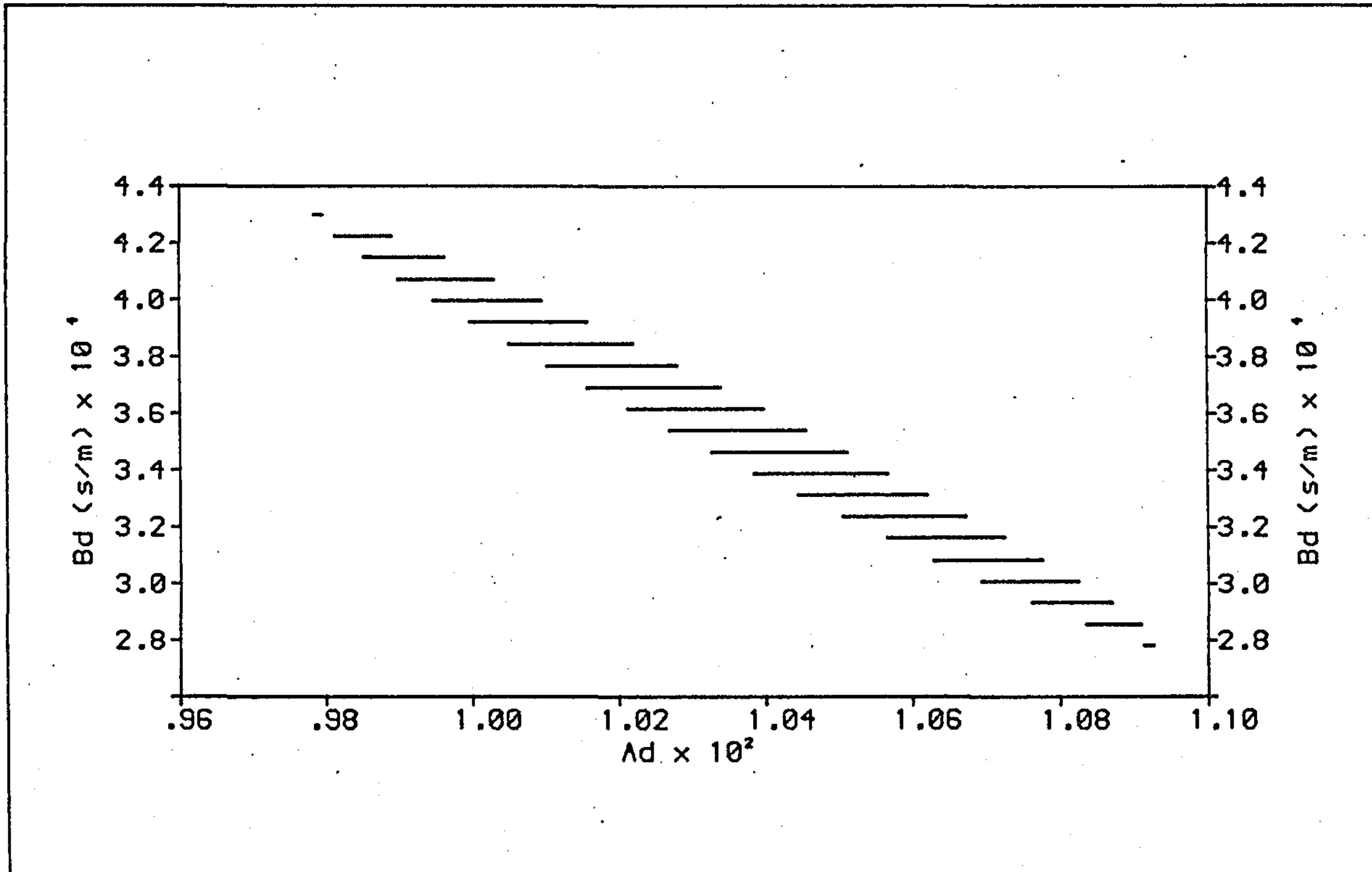
95% CONFIDENCE LIMITS FOR RMS=0.02970 AT MINIMUM (nu=2, Cd=const)

Fig 7.18

TuftyPlot

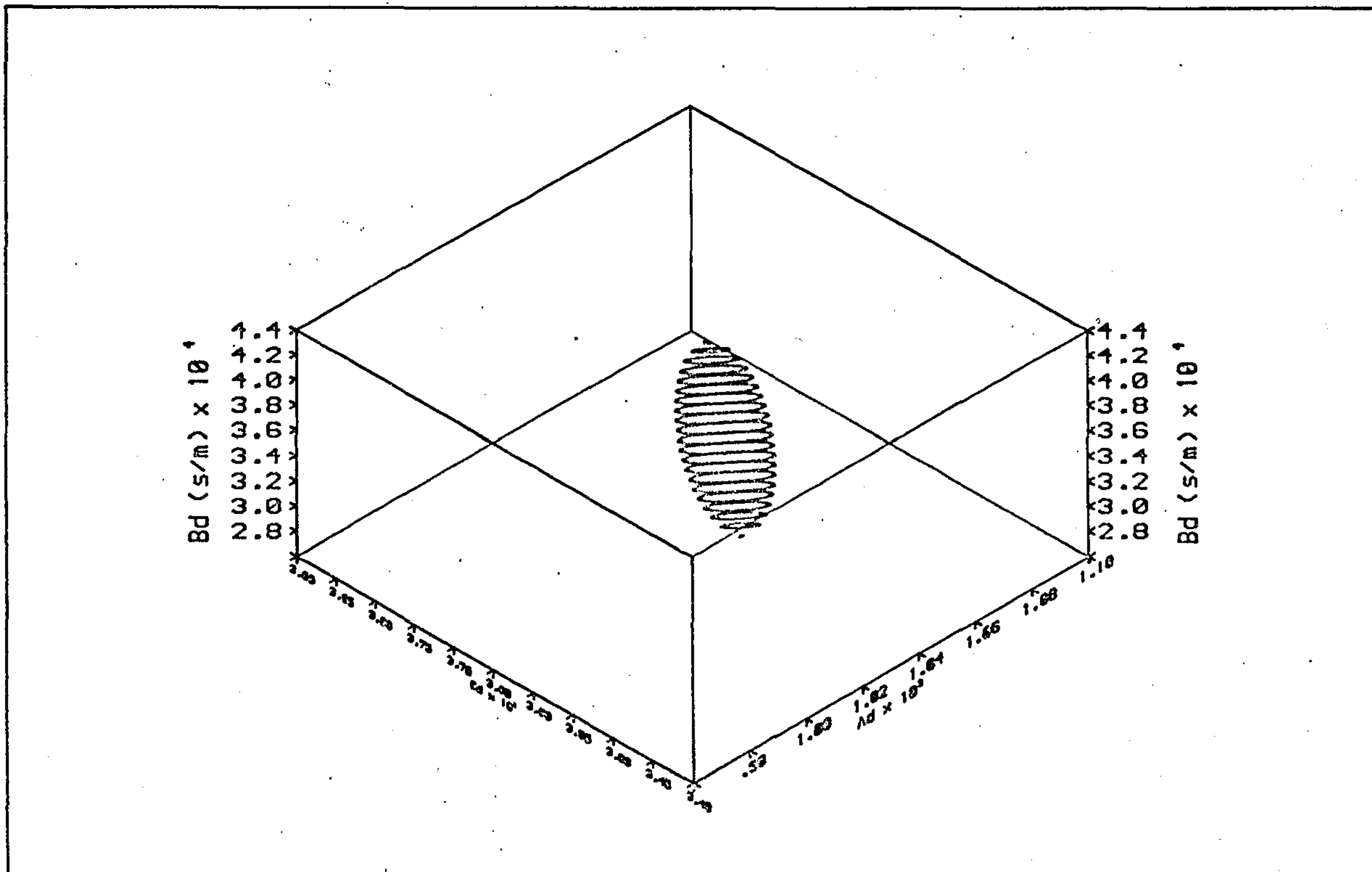


95% CONFIDENCE LIMITS FOR RMS=0.0297 AT MINIMUM

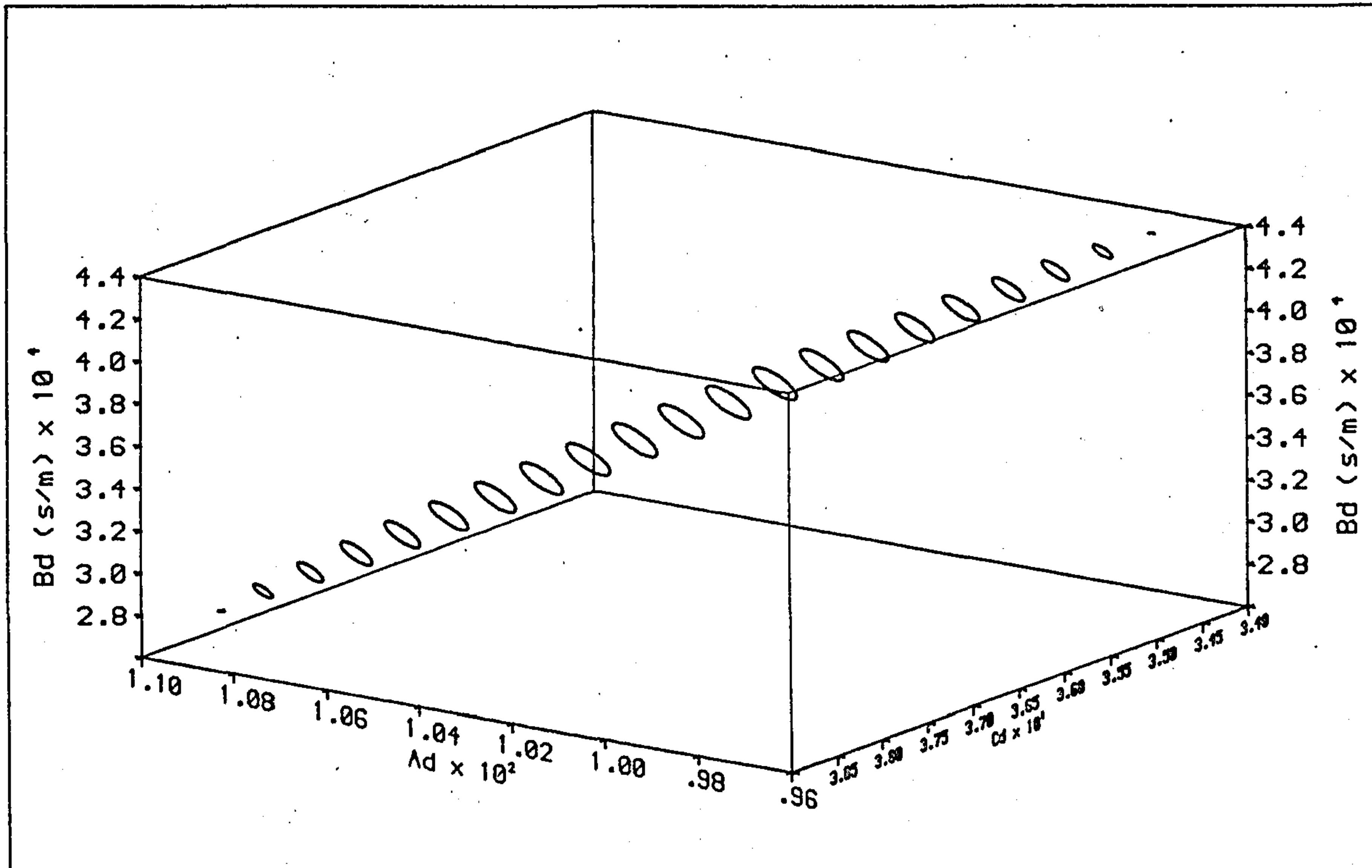


TuFtu.Plot

95% CONFIDENCE LIMITS FOR RMS=0.0297 AT MINIMUM



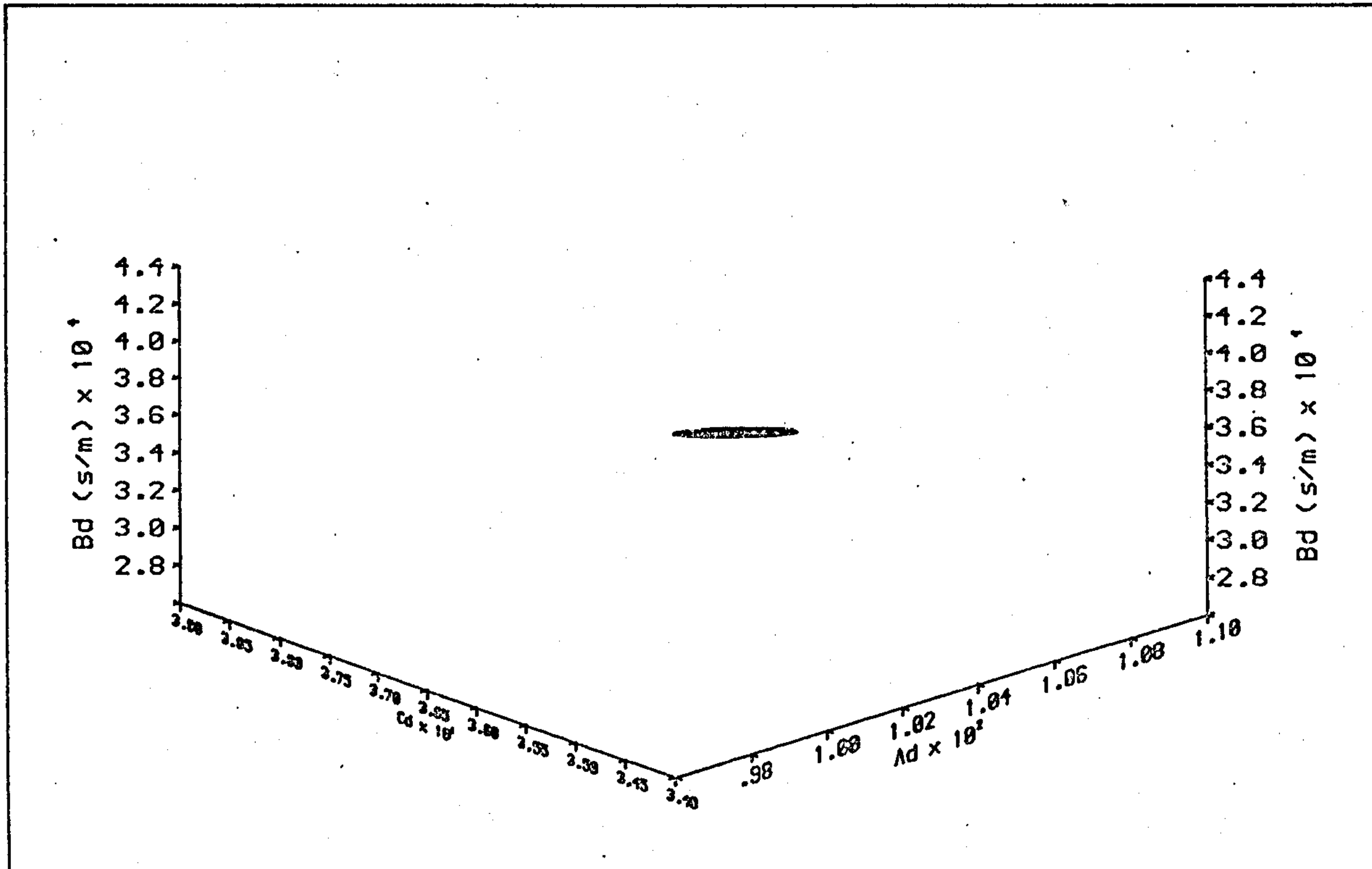
TuFtu.Plot 95% CONFIDENCE LIMITS FOR RMS=0.0297 AT MINIMUM



TufTu.Plot

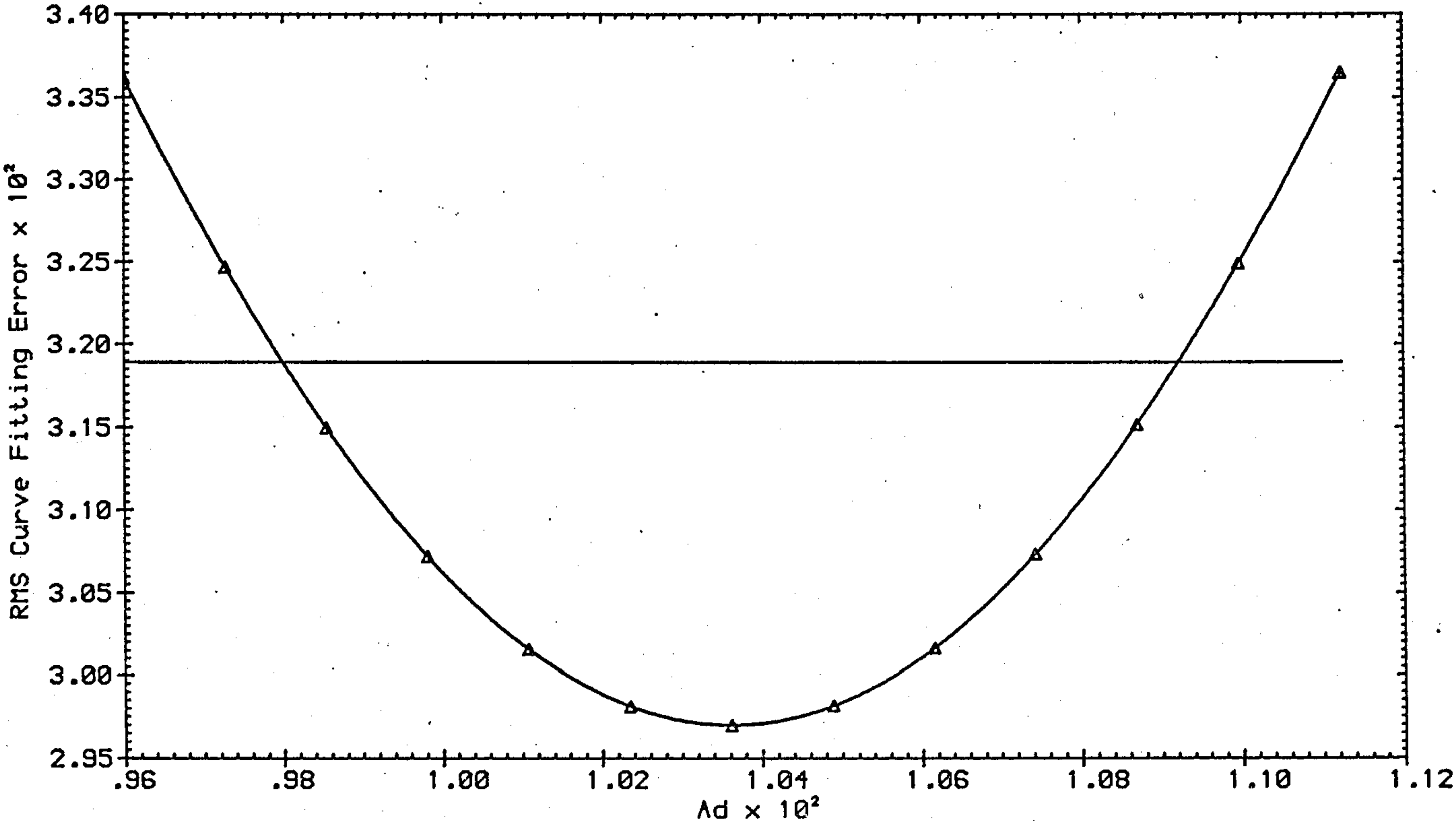
95% CONFIDENCE LIMITS FOR RMS=0.0297 AT MINIMUM

Fig 4.20



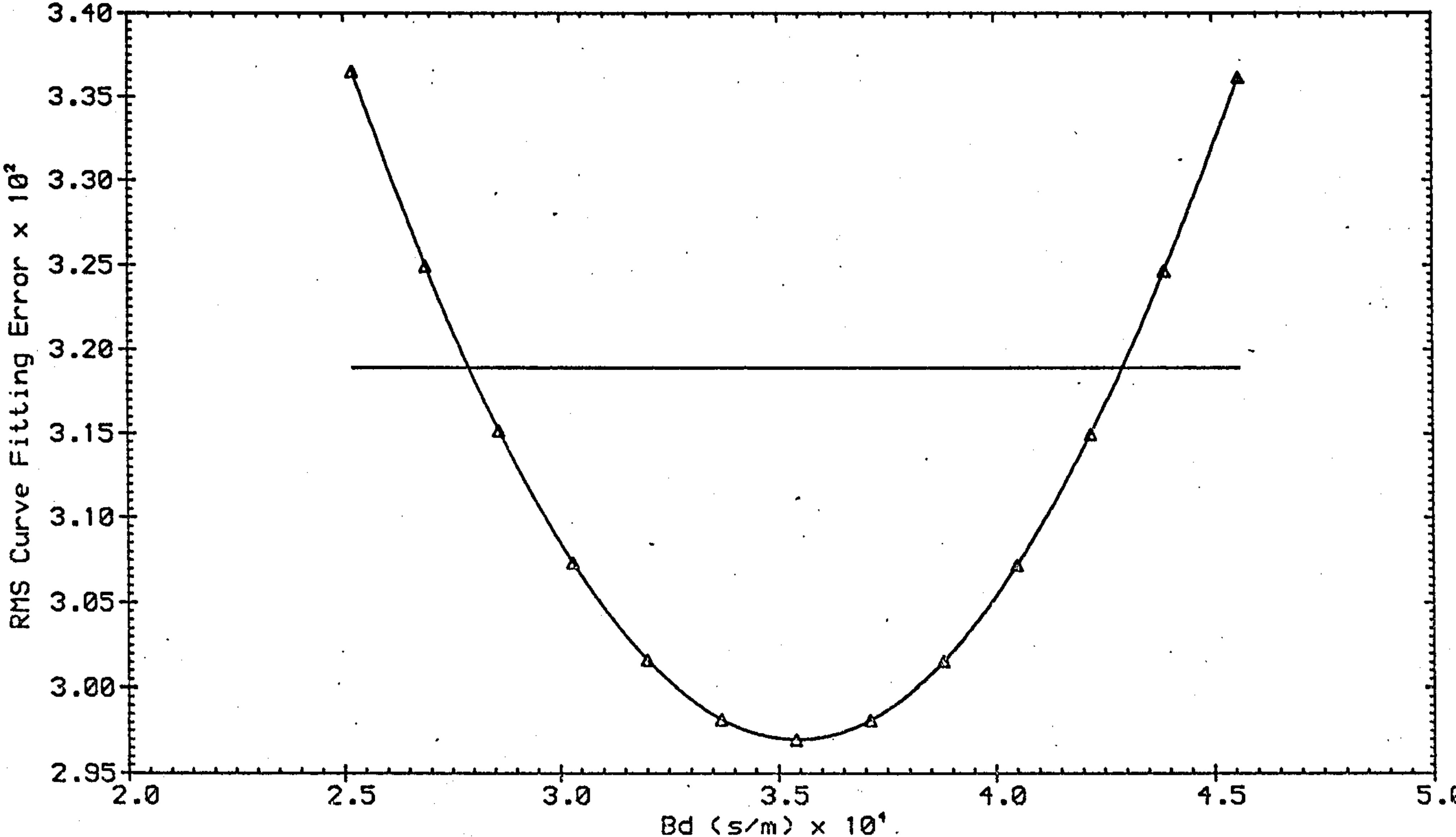
95% CONFIDENCE LIMITS FOR RMS=0.0297 AT MINIMUM

Fig 7.21



95% CONFIDENCE LIMITS OF Ad FOR RMS=0.02970 AT MINIMUM (nu=3)

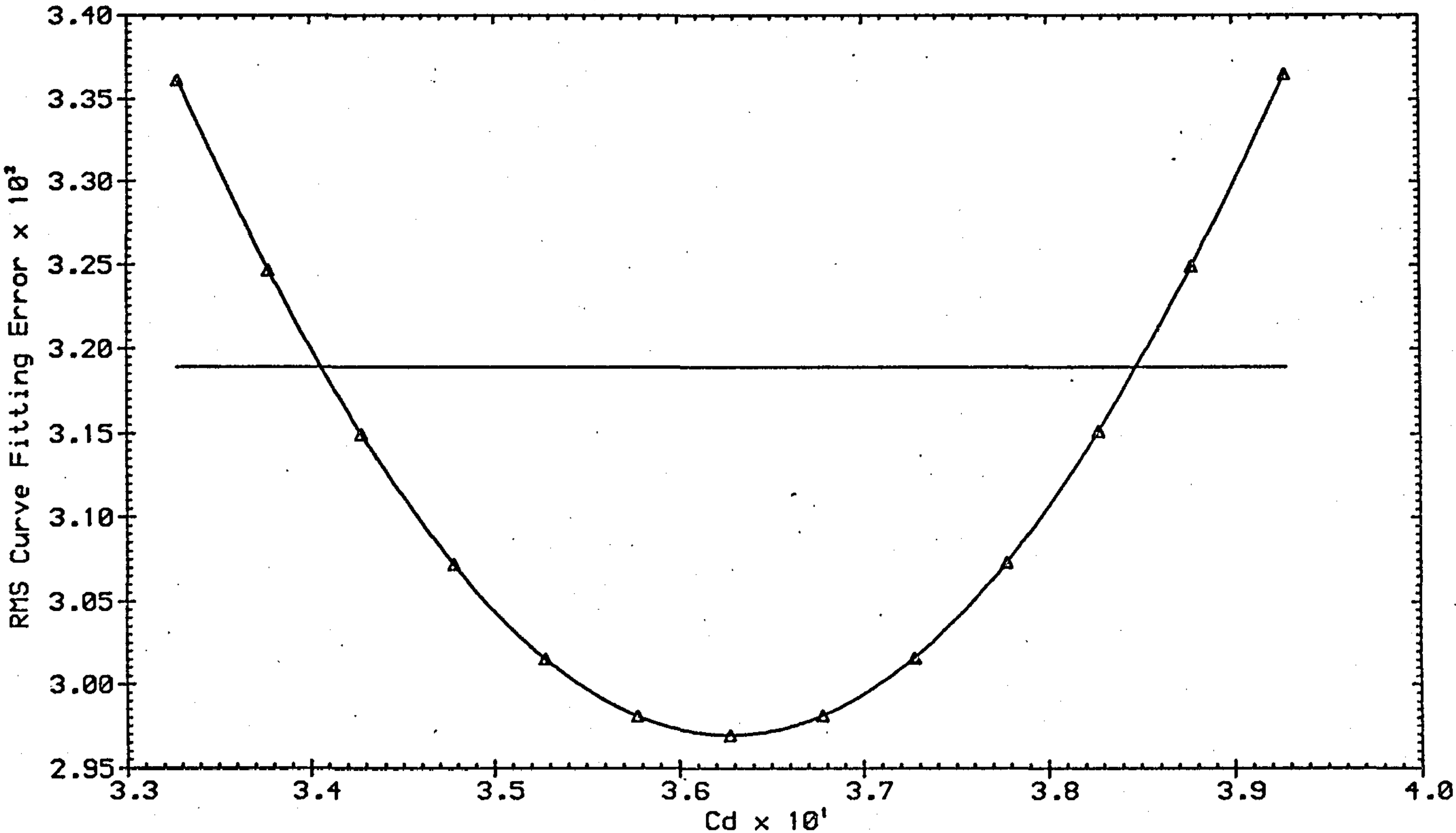
TuFtu.Plot



95% CONFIDENCE LIMITS OF Bd FOR RMS=0.02970 AT MINIMUM (nu=3)

TuFty.Plot

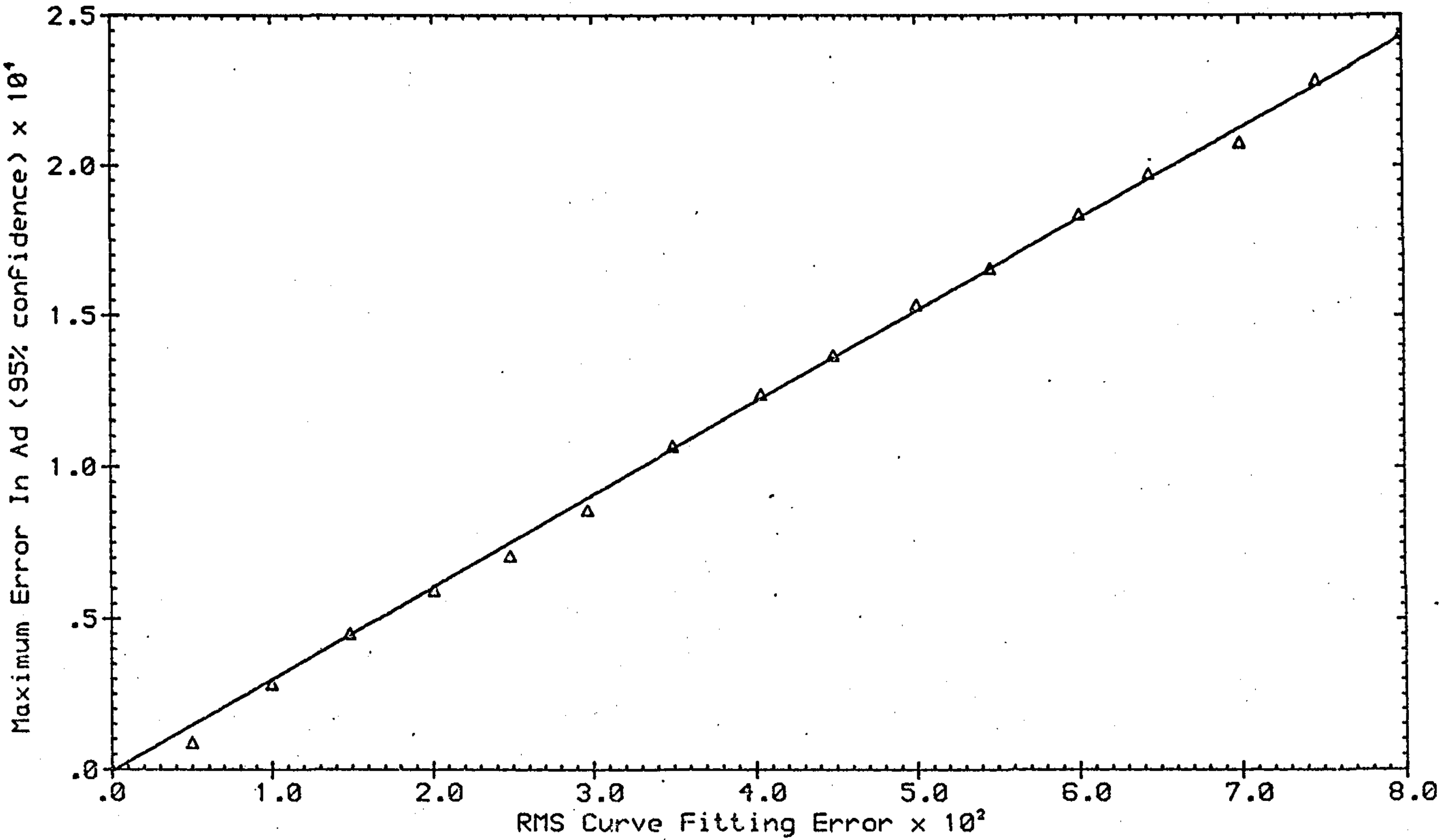
Fig 4.23



95% CONFIDENCE LIMITS OF Cd FOR RMS=0.02970 AT MINIMUM (nu=3)

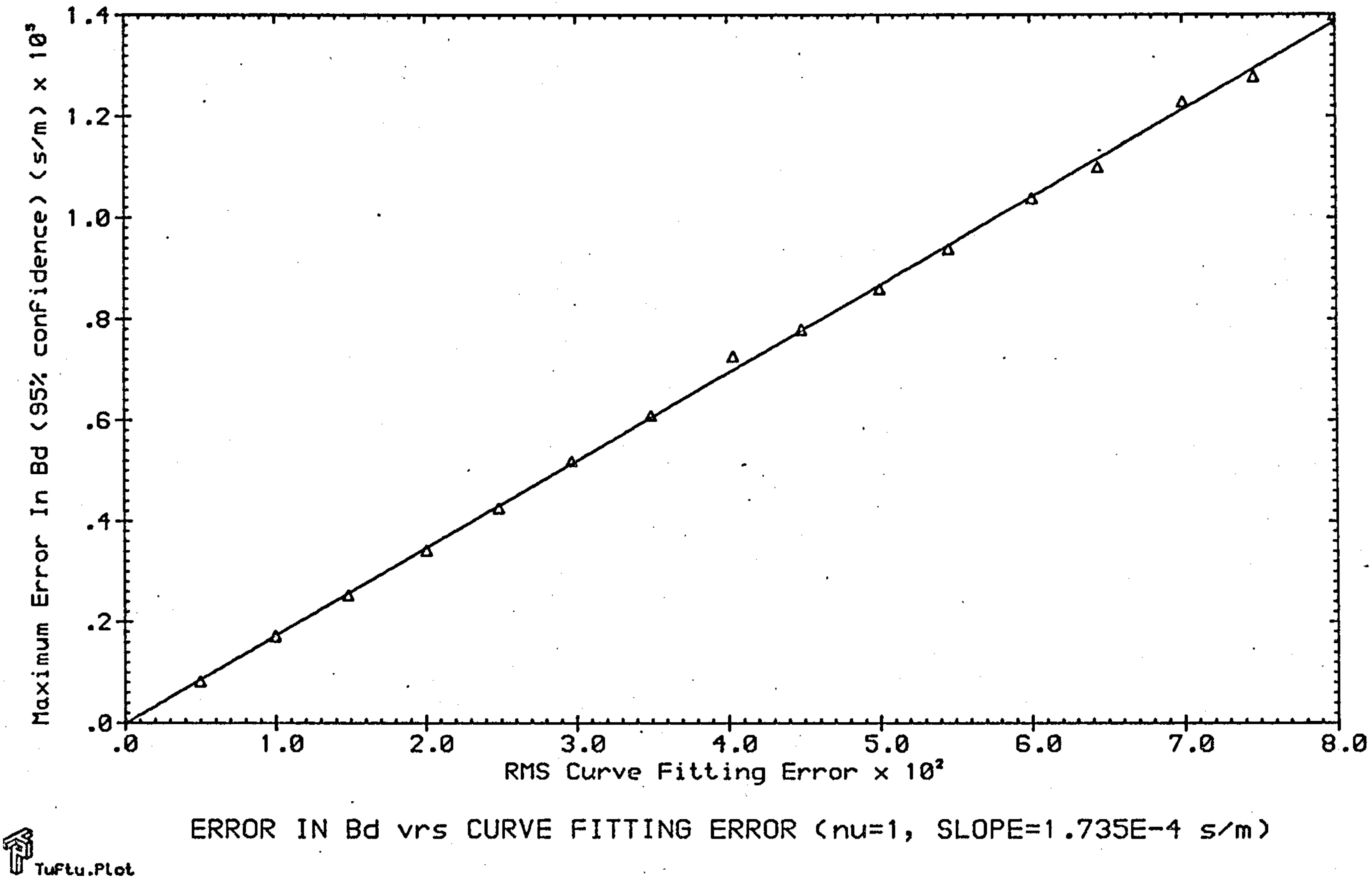
Tufty Plot

Fig 4.24

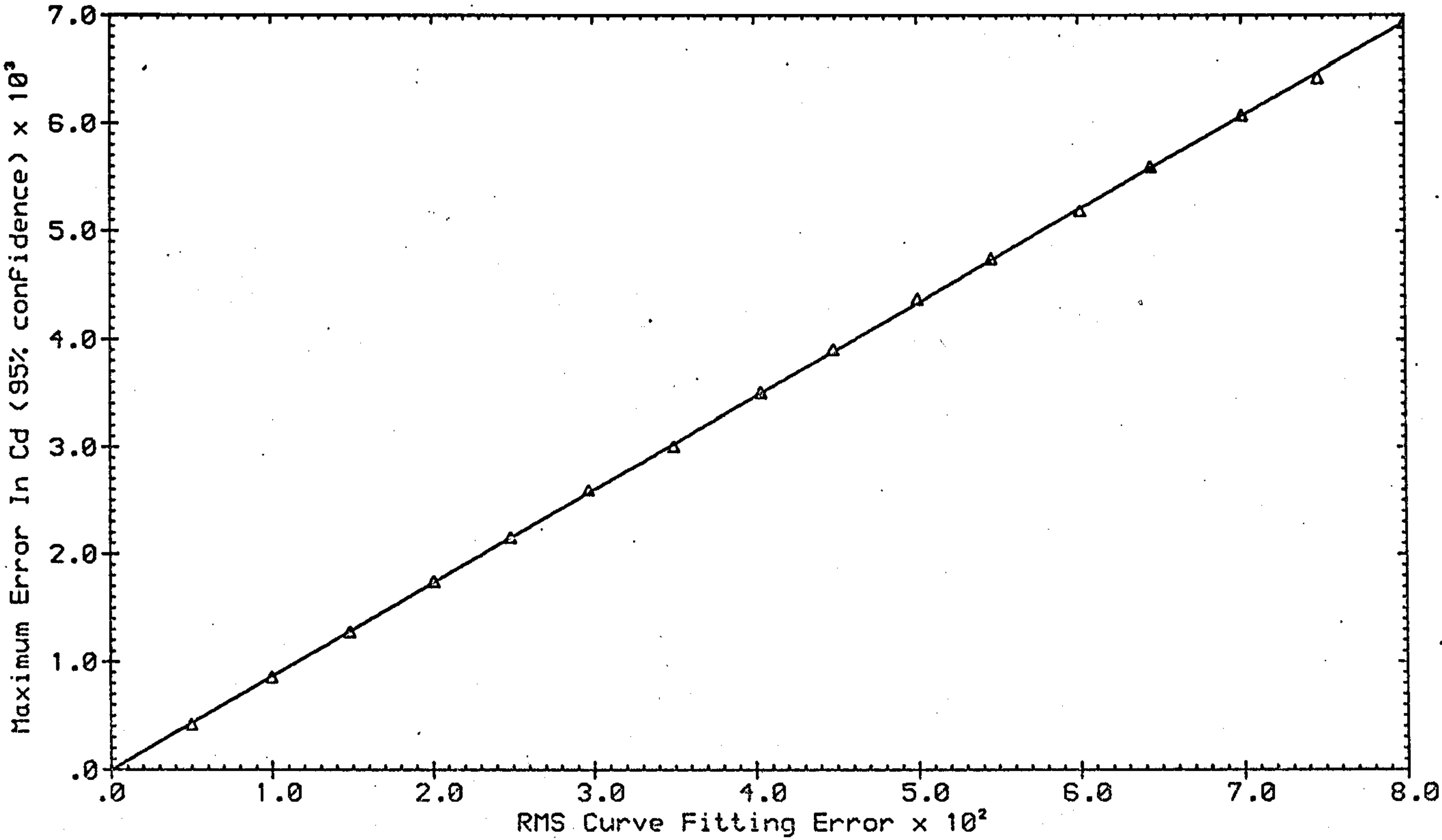


ERROR IN Ad vrs CURVE FITTING ERROR (nu=1, SLOPE=3.043E-3)

TuFtu.Plot

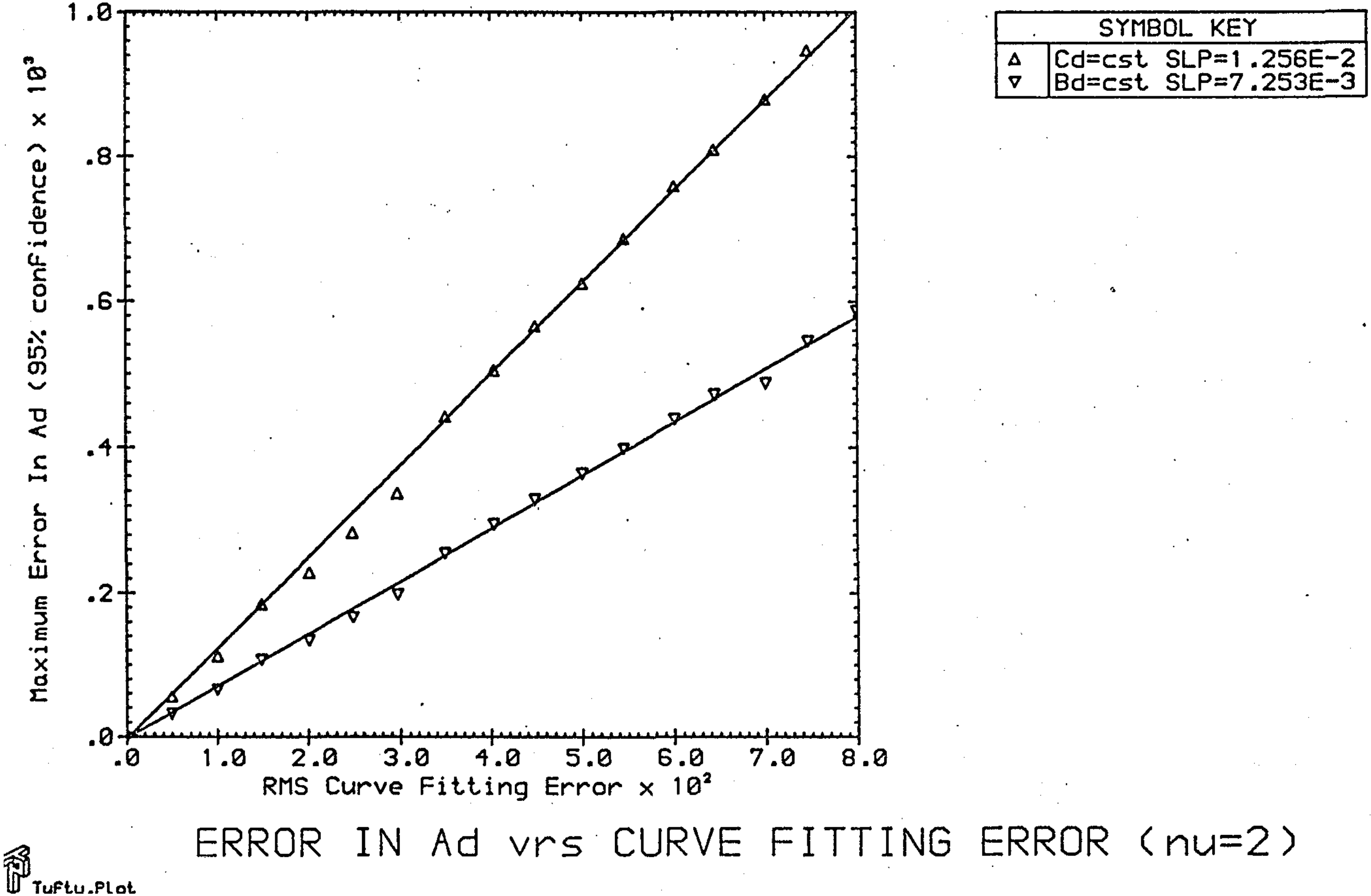


ERROR IN Bd vrs CURVE FITTING ERROR (nu=1, SLOPE=1.735E-4 s/m)

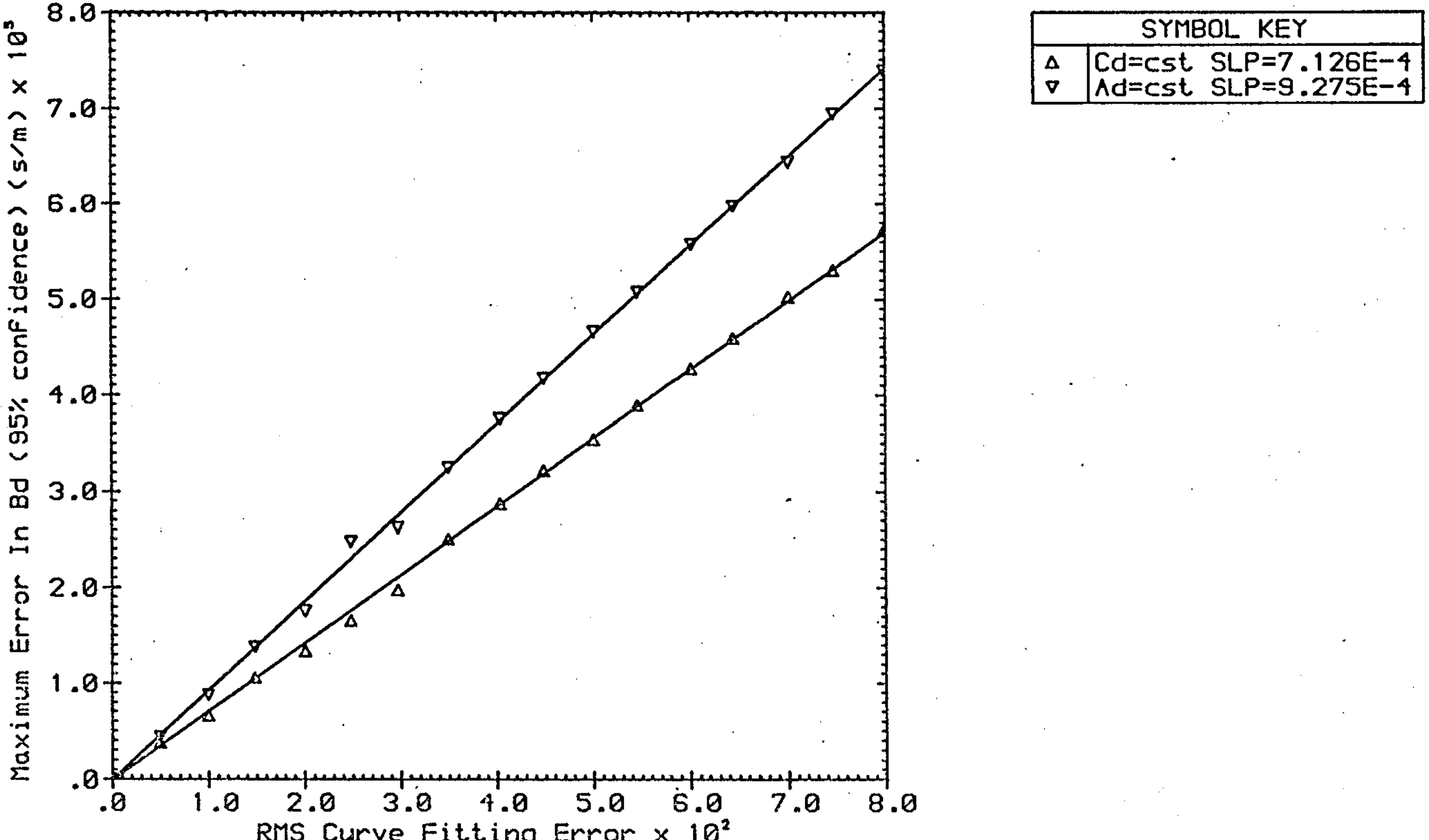


ERROR IN Cd vrs CURVE FITTING ERROR (nu=1, SLOPE=8.672E-2)

TuftyPlot

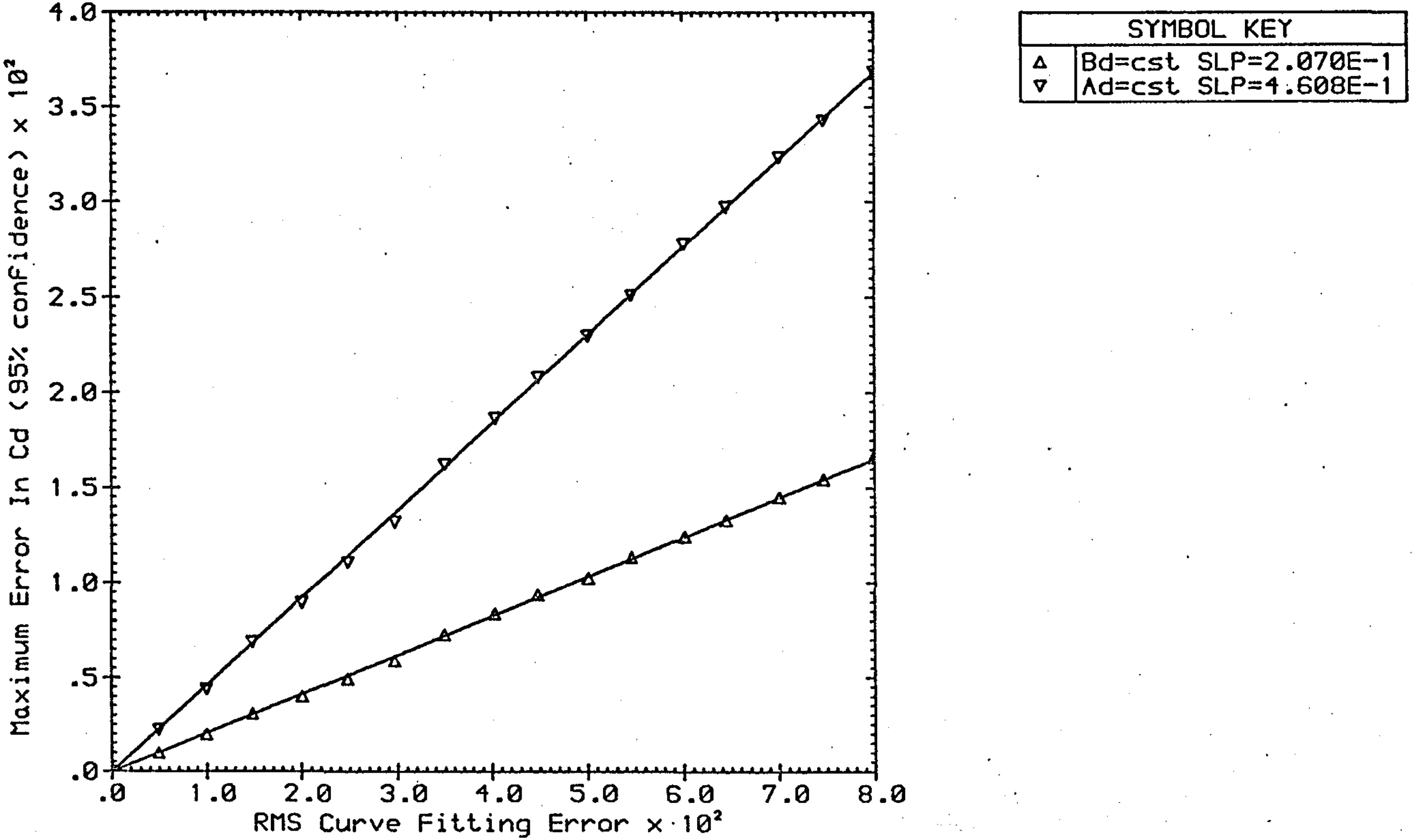


ERROR IN Ad vrs CURVE FITTING ERROR (nu=2)



ERROR IN Bd vrs CURVE FITTING ERROR (nu=2)

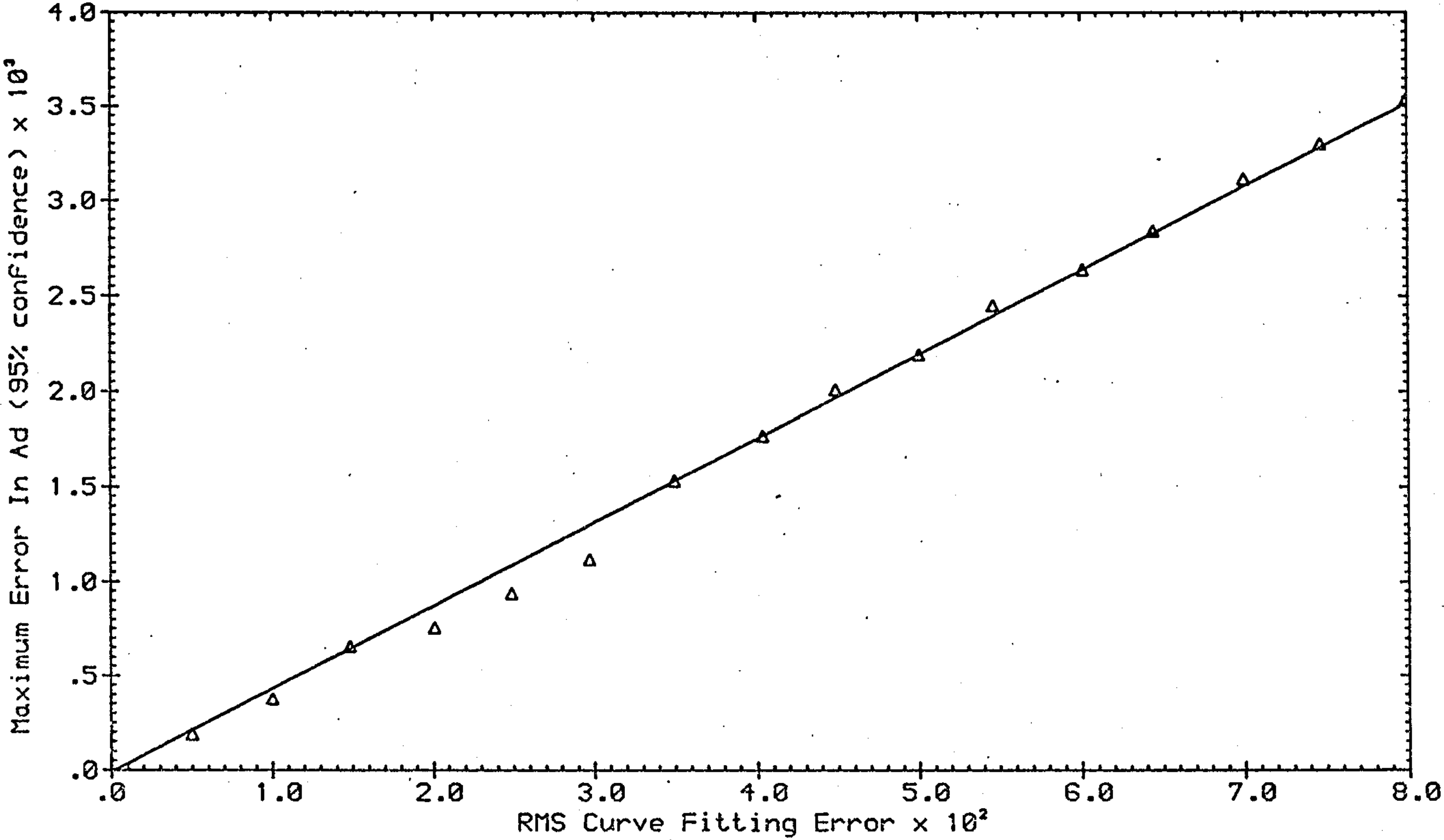
TuFtu.Plot



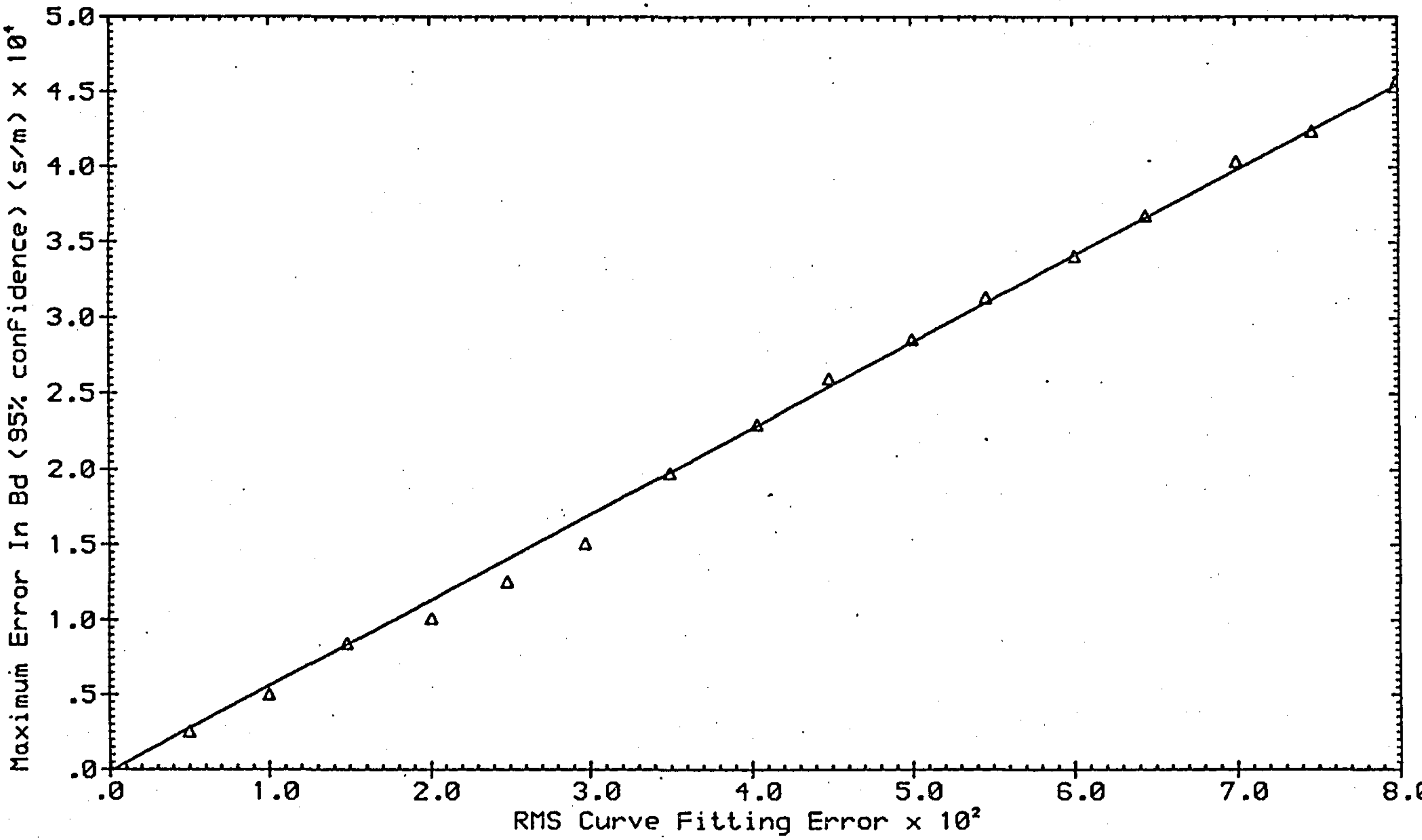
ERROR IN Cd vrs CURVE FITTING ERROR (nu=2)

TuFtu.Plot

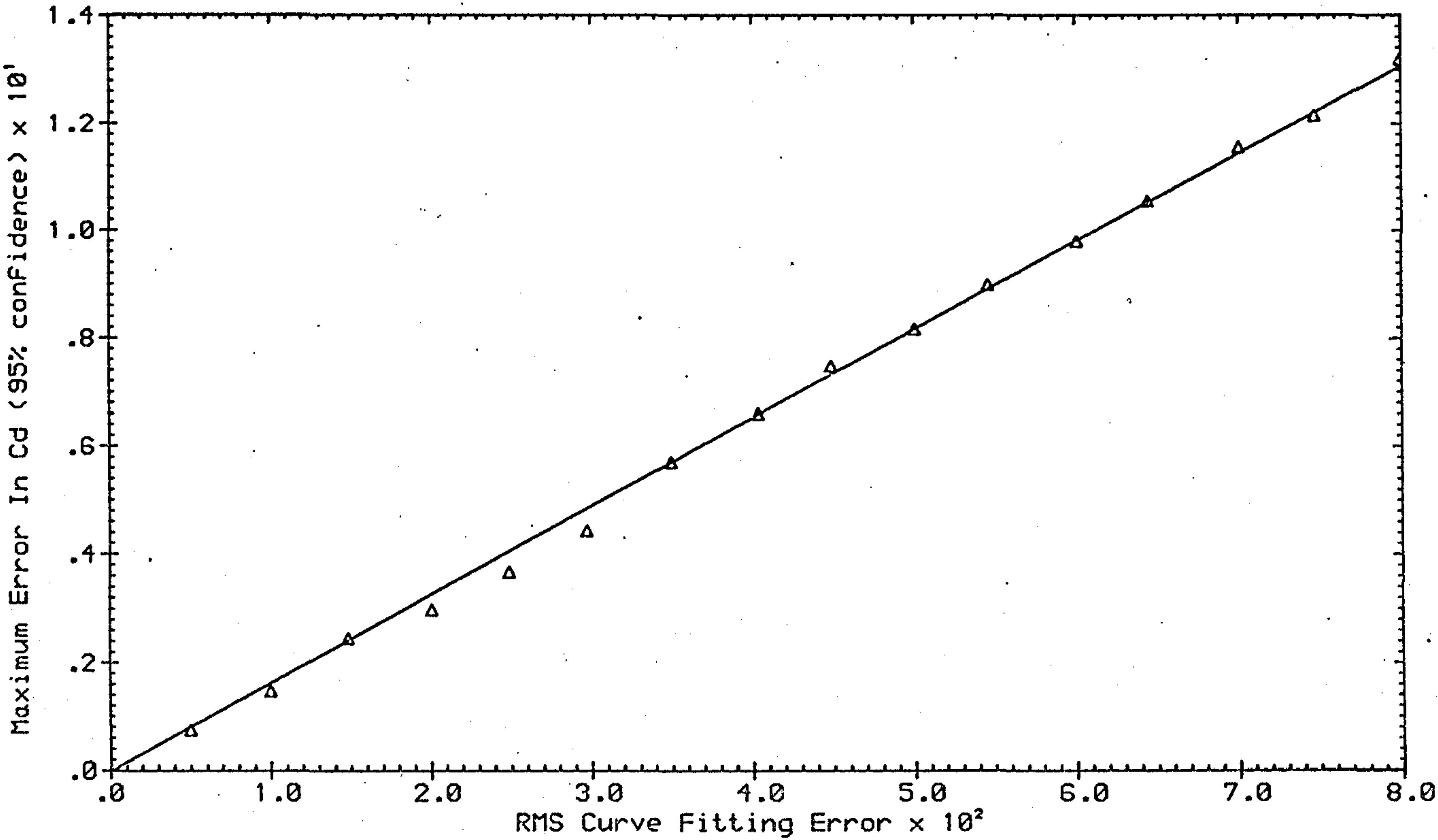
Fig 7.30



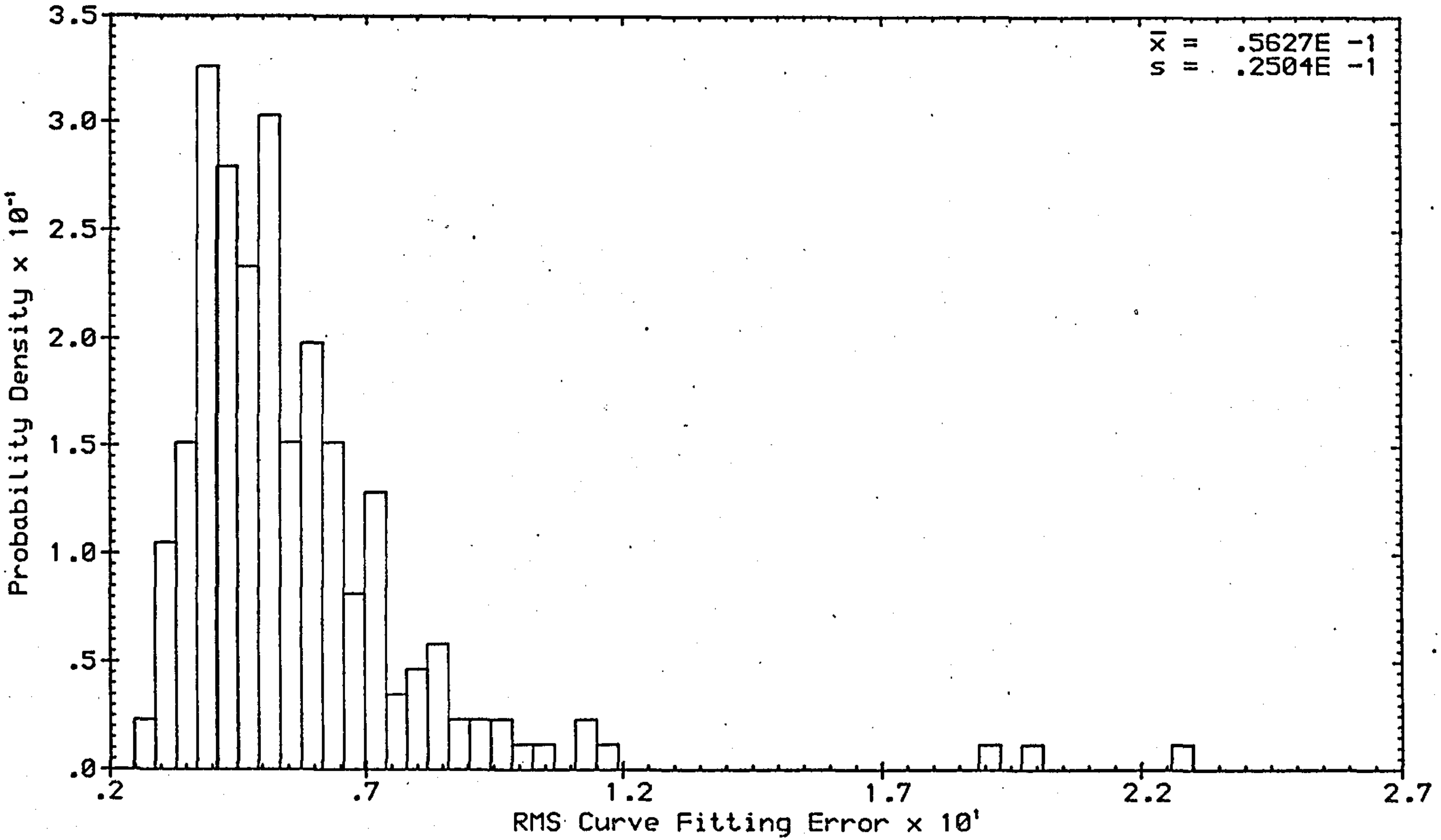
ERROR IN Ad vrs CURVE FITTING ERROR (nu=3, SLOPE=4.404E-2)



ERROR IN Bd vrs CURVE FITTING ERROR (nu=3, SLOPE=5.697E-3 s/m)

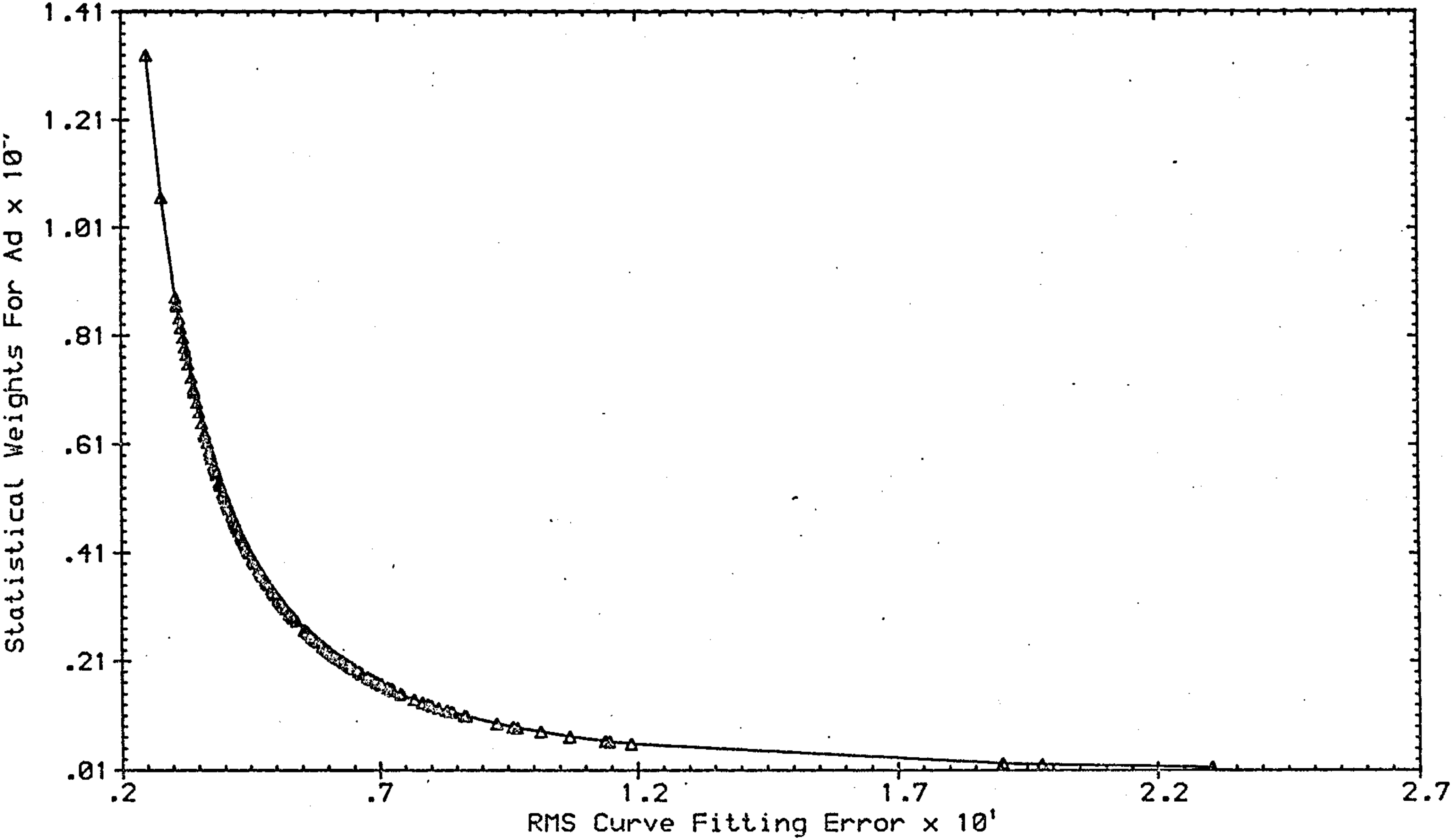


ERROR IN Cd vrs CURVE FITTING ERROR (nu=3, SLOPE=1.636E-0)



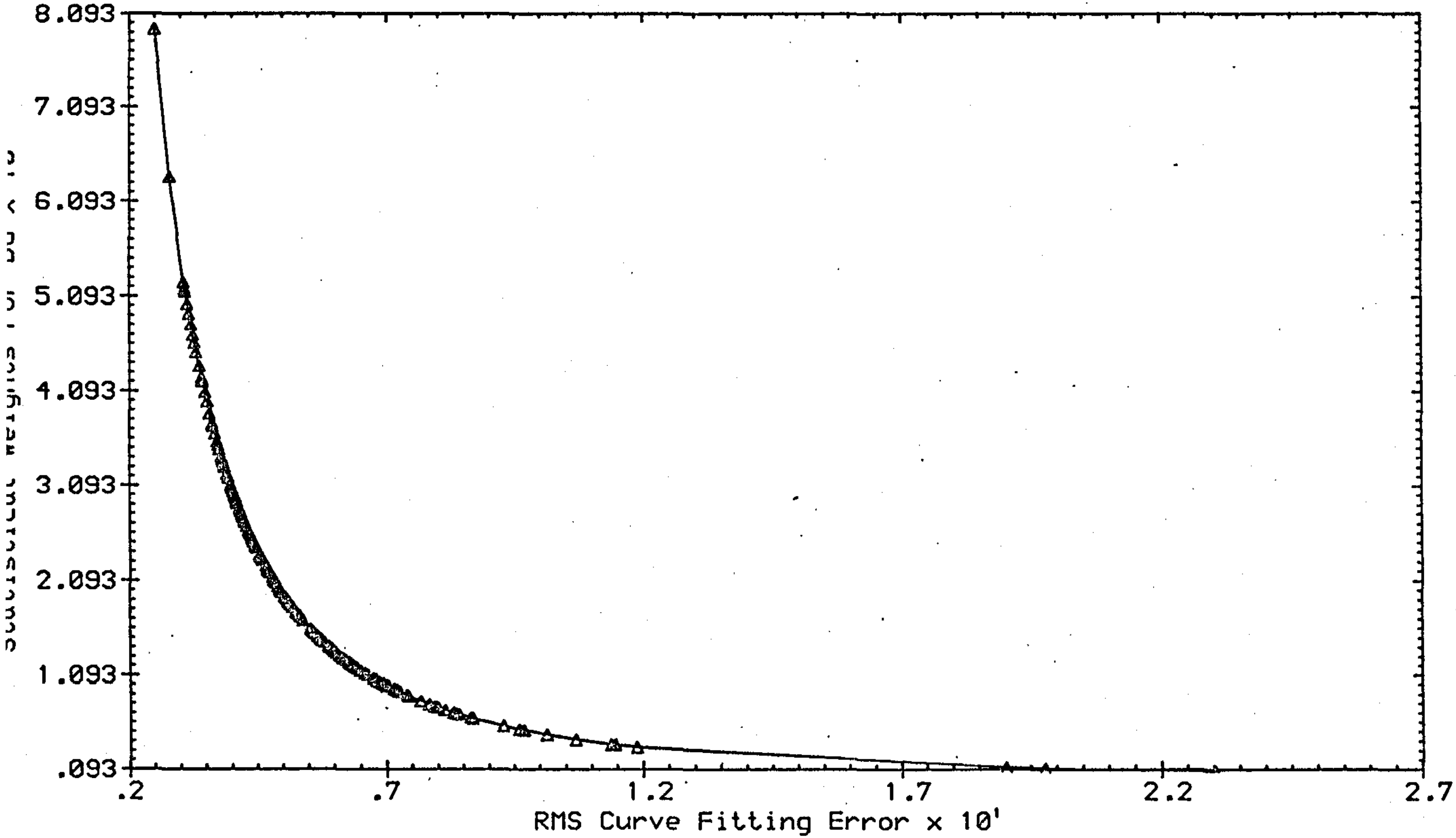
PROBABILITY DENSITY BAR CHART FOR RMS CURVE FITTING ERROR (CORRECTED DATA)

Tufty.Plot



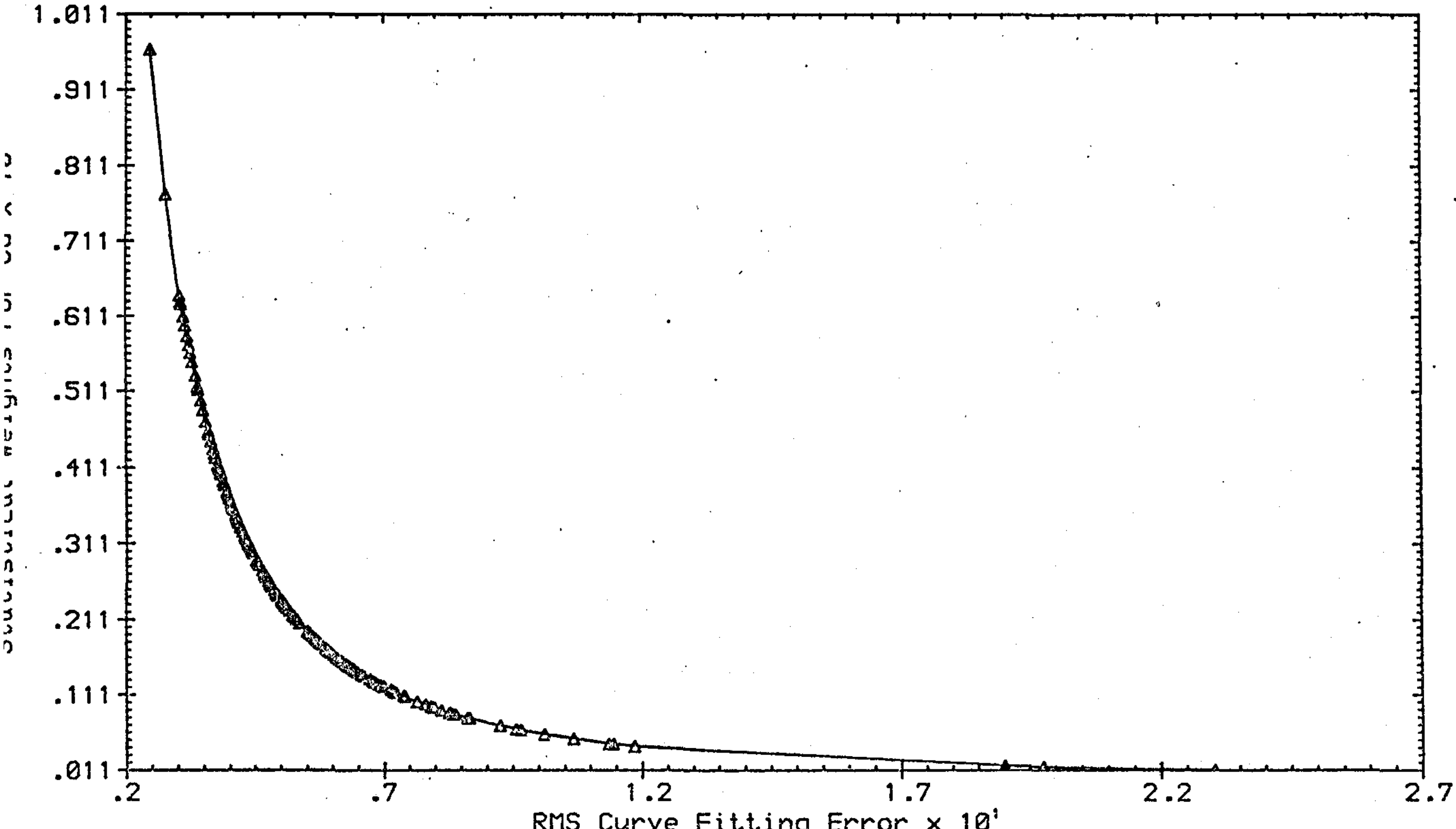
STATISTICAL WEIGHTS FOR Ad vrs RMS CURVE FITTING ERROR

TuFtu.Plot

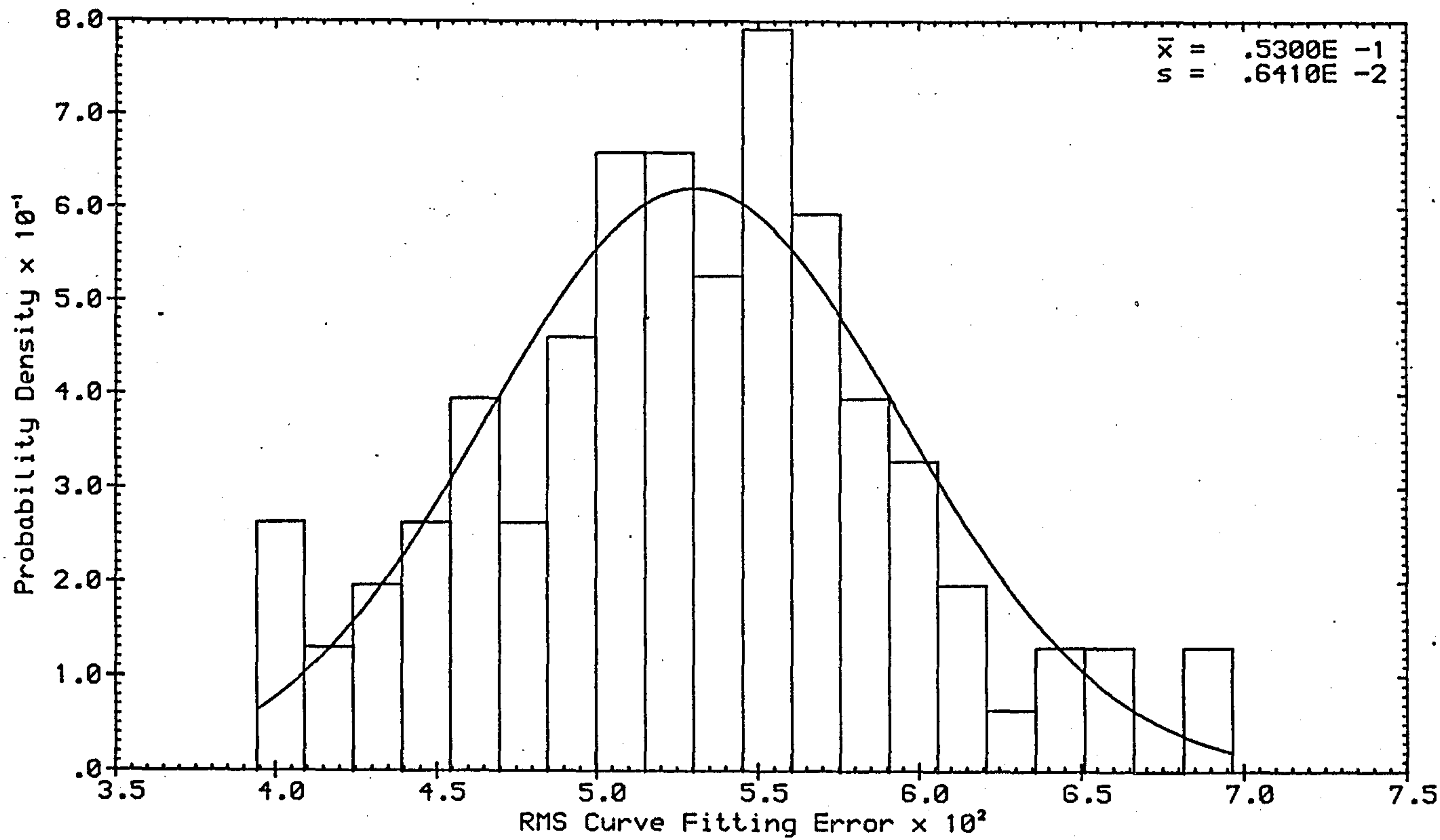


STATISTICAL WEIGHTS FOR Bd vrs RMS CURVE FITTING ERROR

TuFtu.Plot

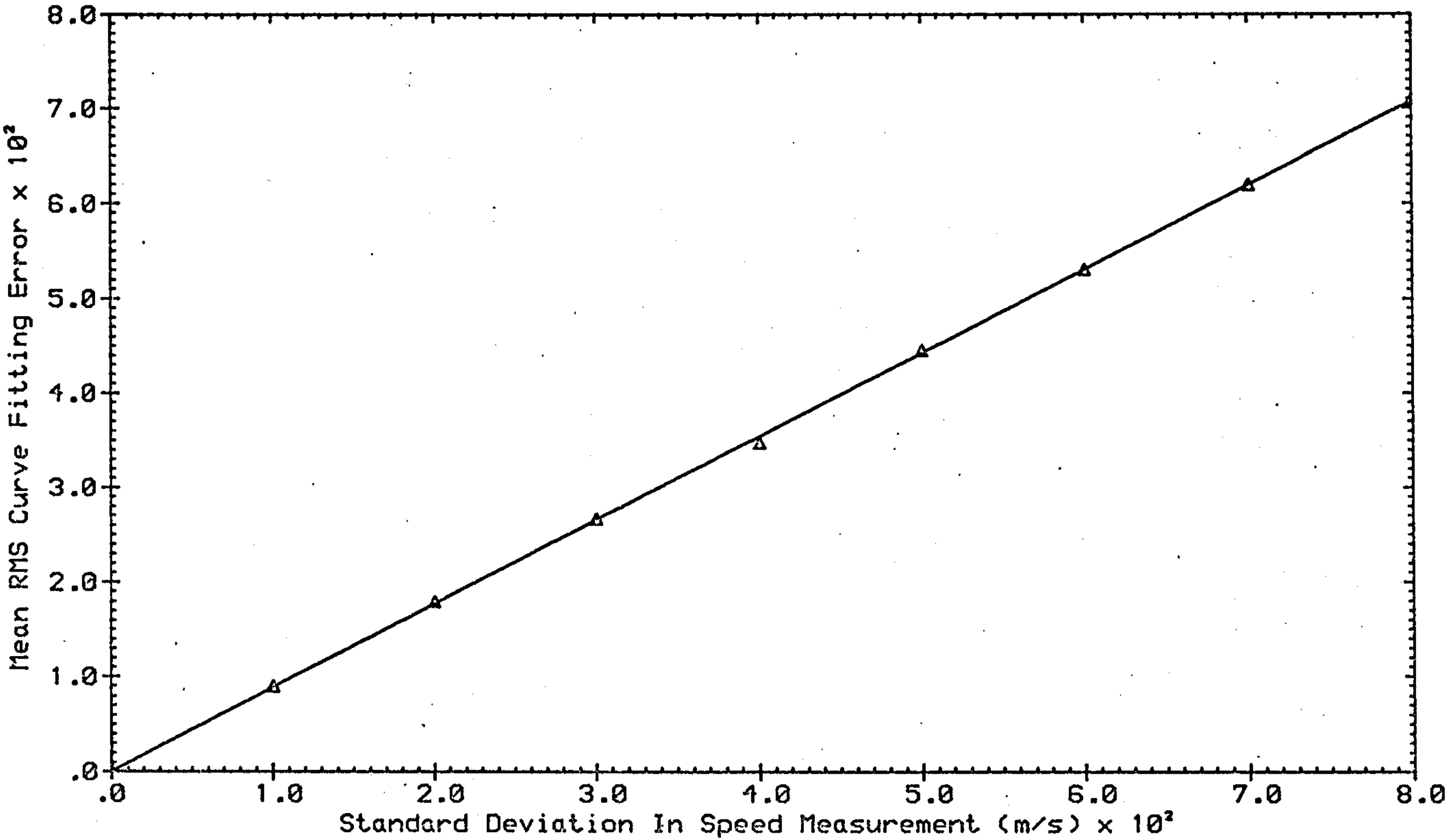


STATISTICAL WEIGHTS FOR Cd vrs RMS CURVE FITTING ERROR



PROBABILITY DISTRIBUTION OF RMS CURVE FITTING ERROR (SDSP=0.06 m/s)

Fig 5.1



MEAN RMS CURVE FITTING ERROR vrs STANDARD DEVIATION IN SPEED MEASUREMENT

Tufty.Plot

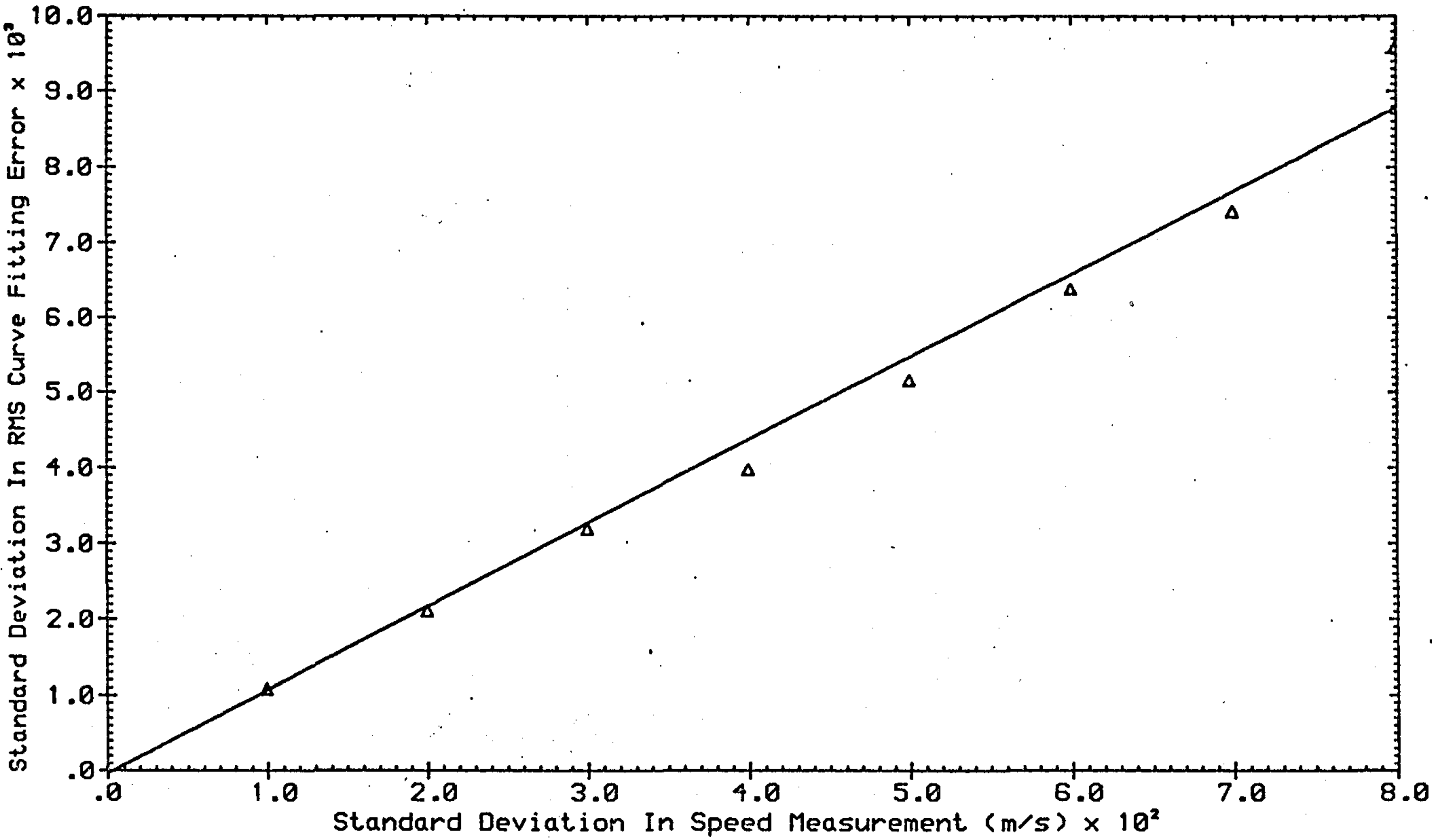
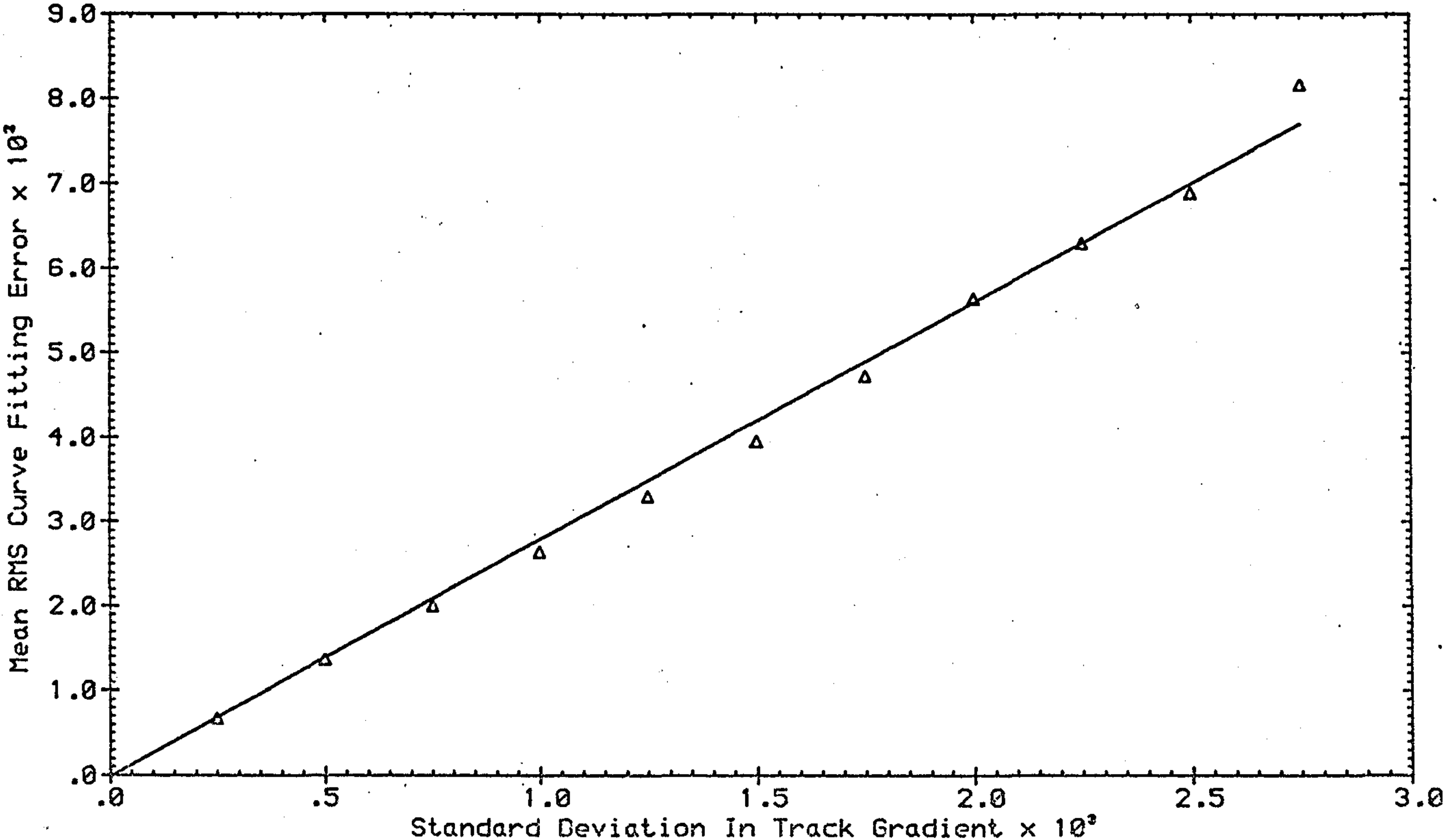


Fig 5.3

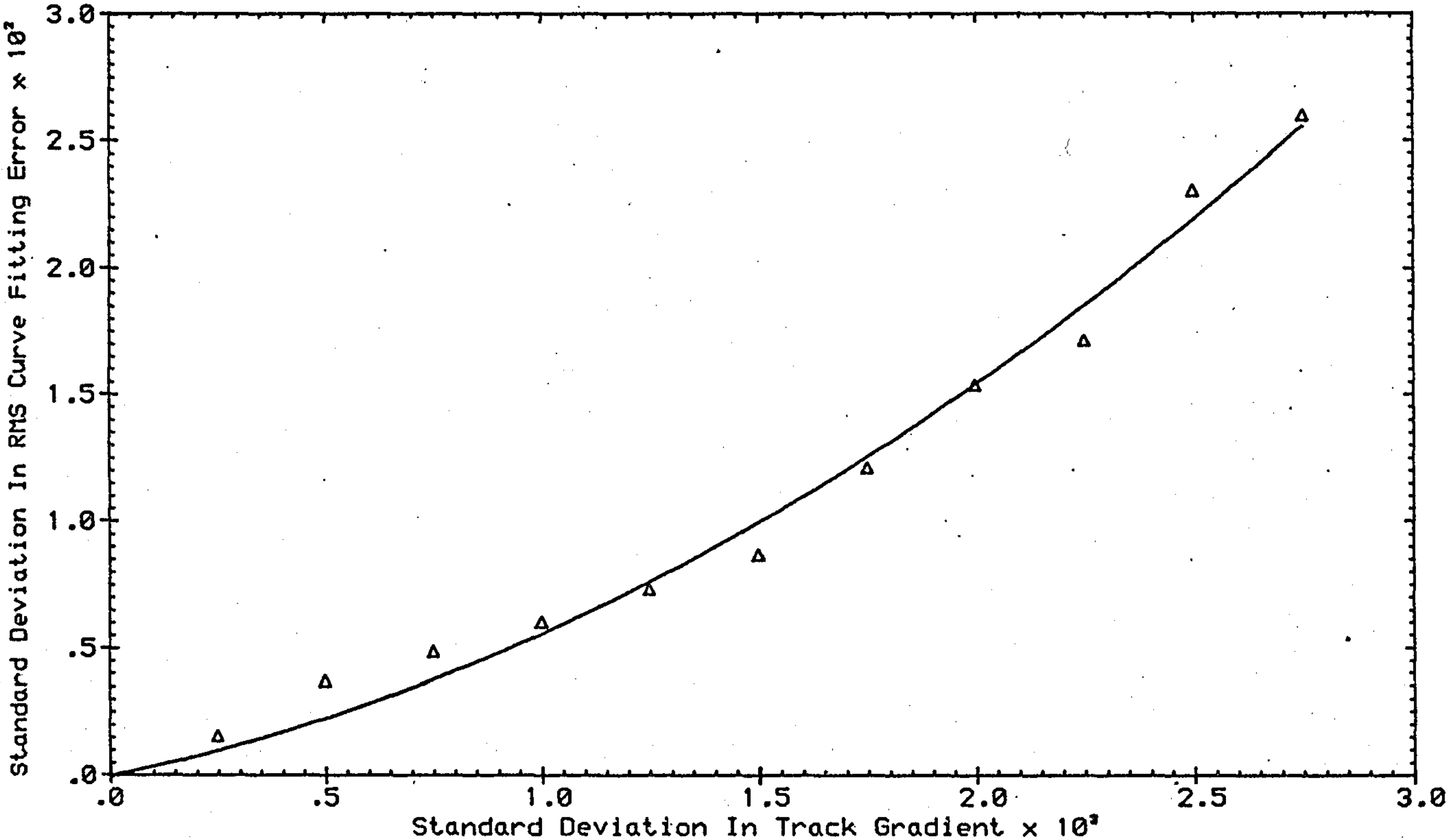
STANDARD DEVIATION IN RMS CURVE FITTING ERROR vrs STANDARD DEVIATION IN SPEED MEASUREMENT

Tufty.Plot



MEAN RMS CURVE FITTING ERROR vrs STANDARD DEVIATION IN TRACK GRADIENT

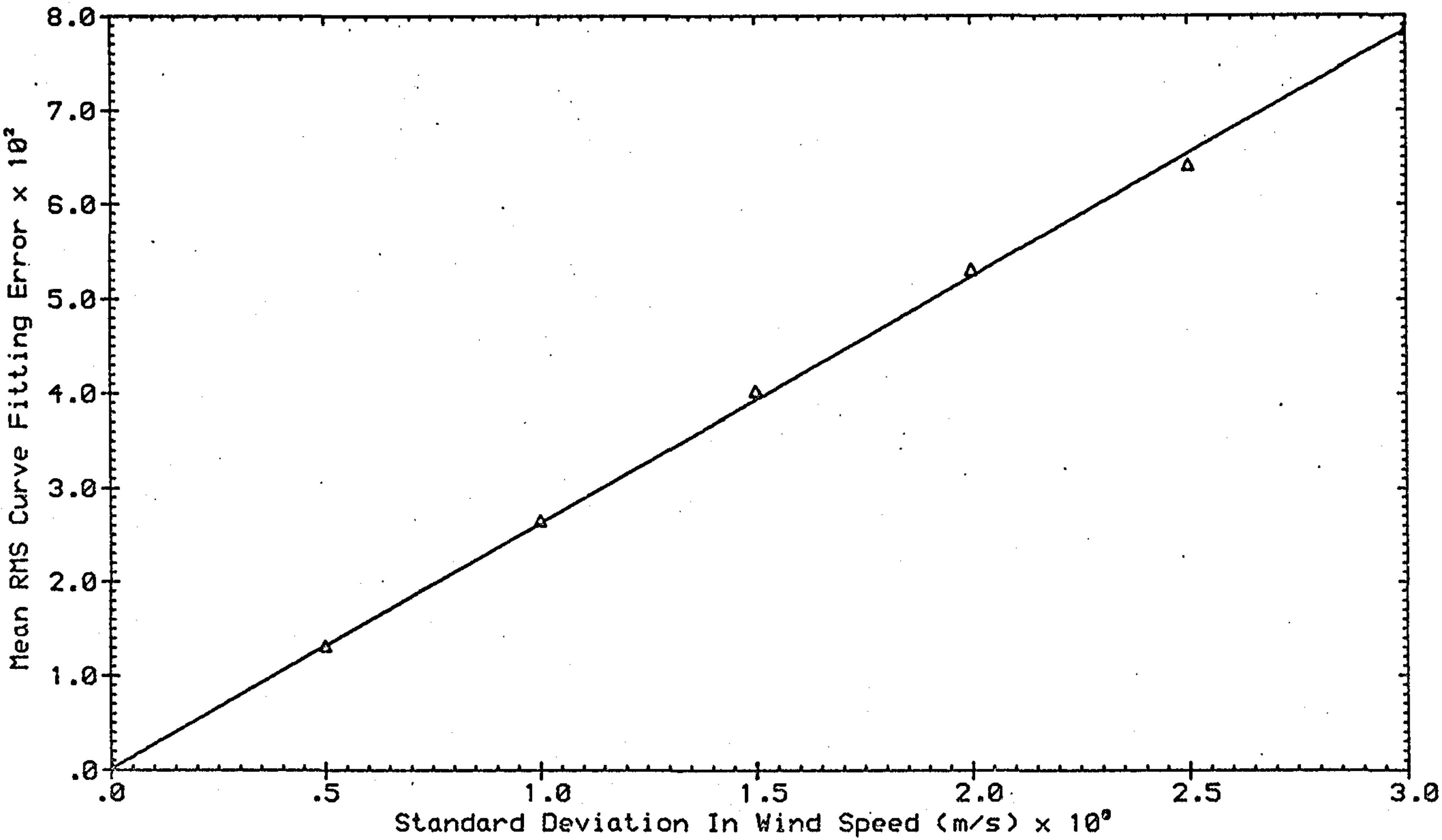
TuFty.Plot



STANDARD DEVIATION IN RMS CURVE FITTING ERROR vrs STANDARD DEVIATION IN TRACK GRADIENT

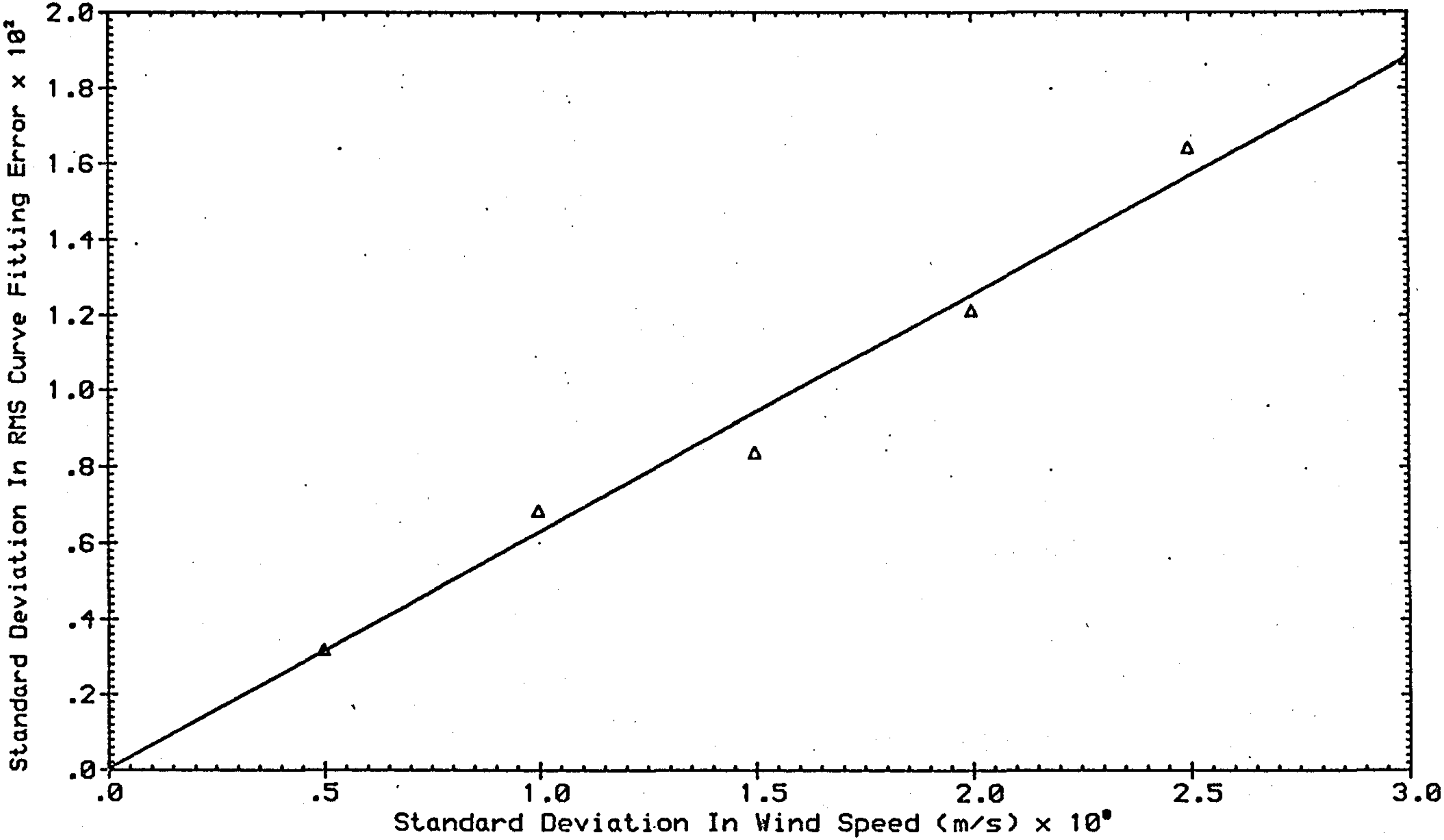
Fig 5.5

Tufty-Plot



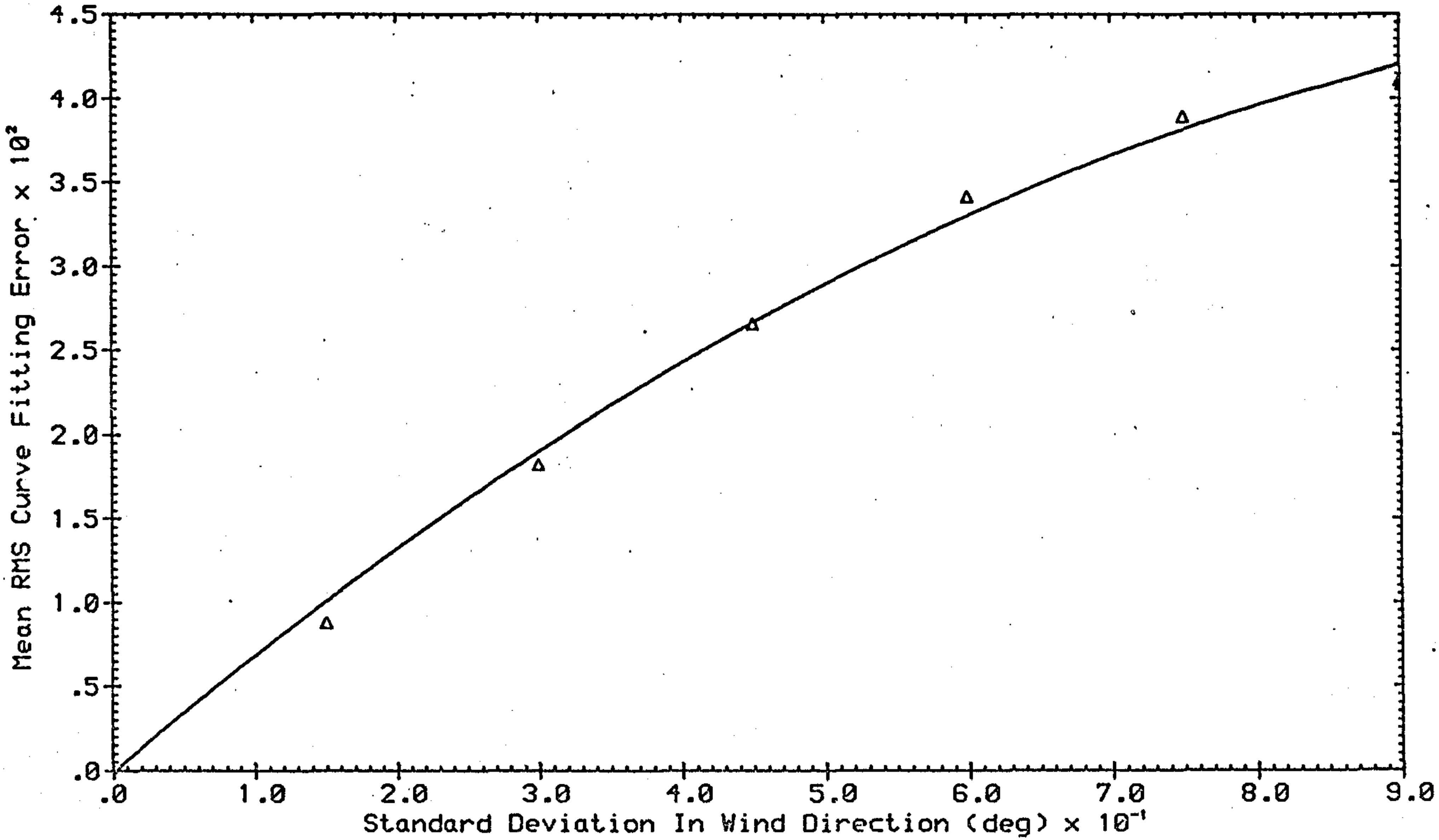
MEAN RMS CURVE FITTING ERROR vrs STANDARD DEVIATION IN WIND SPEED

TuFty.Plot



STANDARD DEVIATION IN RMS CURVE FITTING ERROR vrs STANDARD DEVIATION IN WIND SPEED

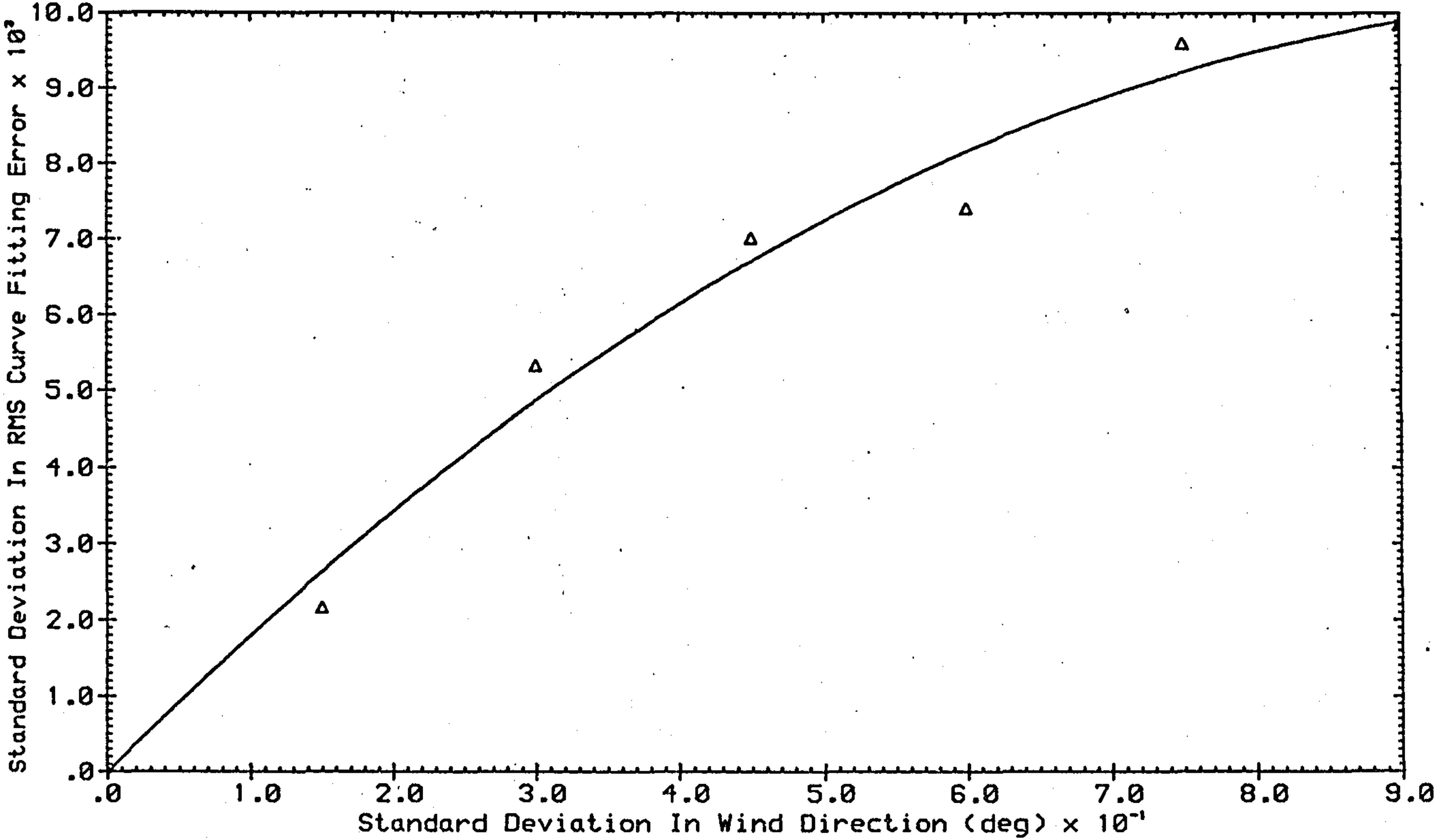
Tufty.Plot



MEAN RMS CURVE FITTING ERROR vrs STANDARD DEVIATION IN WIND DIRECTION

Tufty.Plot

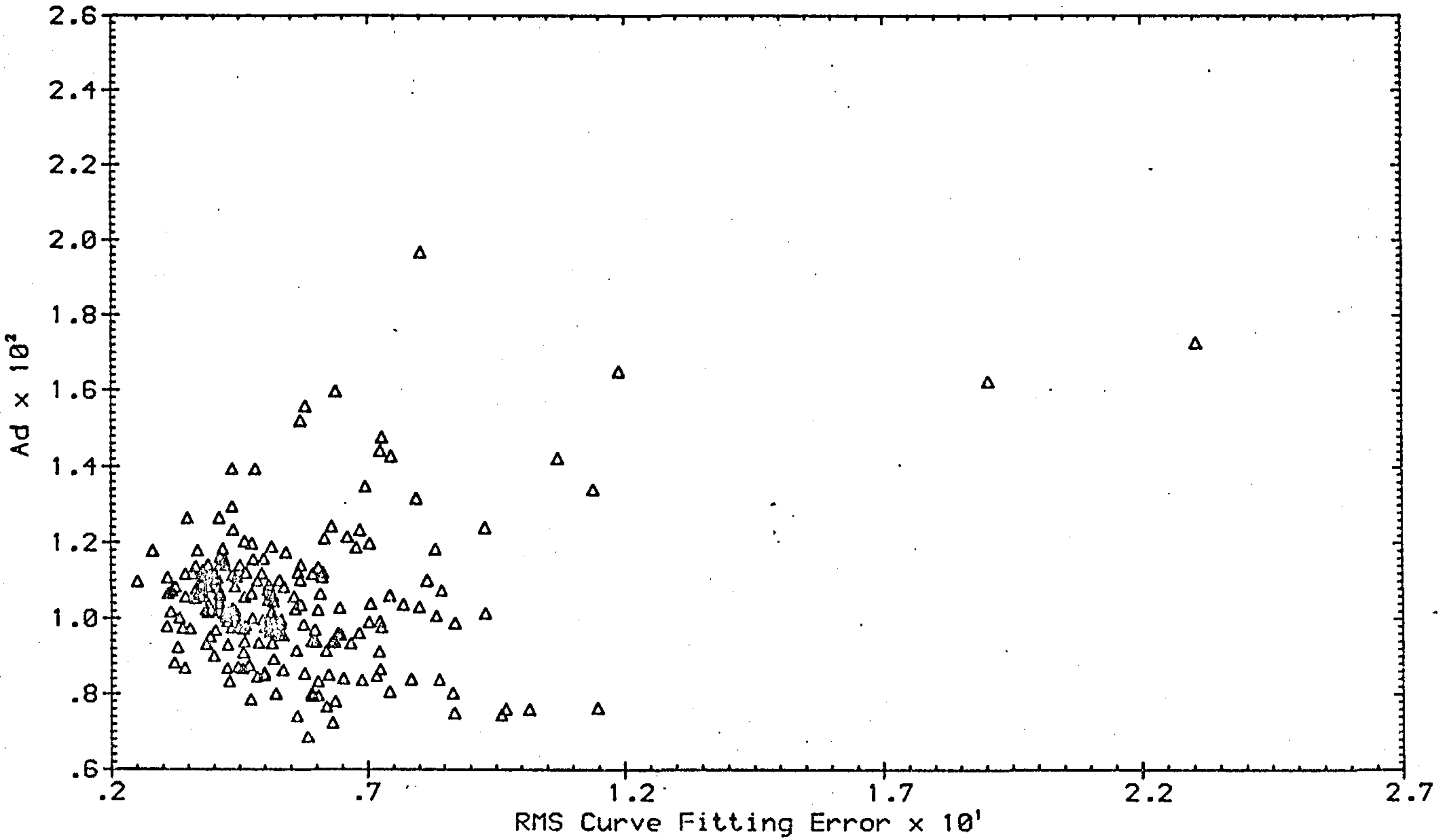
Fig 5.6



STANDARD DEVIATION IN RMS CURVE FITTING ERROR vrs STANDARD DEVIATION IN WIND DIRECTION

Tufty.Plot

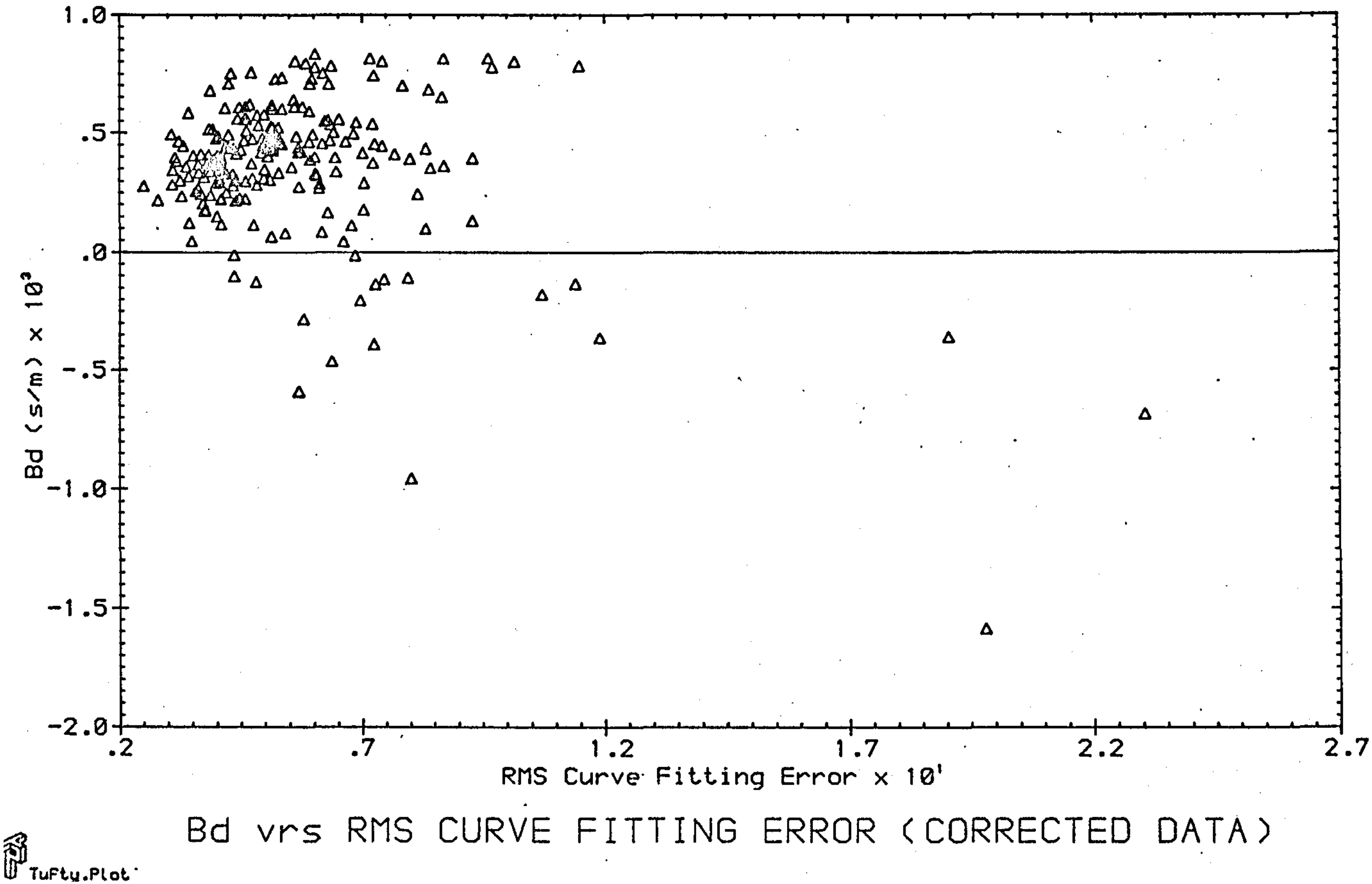
Fig 5.7



Ad vrs RMS CURVE FITTING ERROR (CORRECTED DATA)

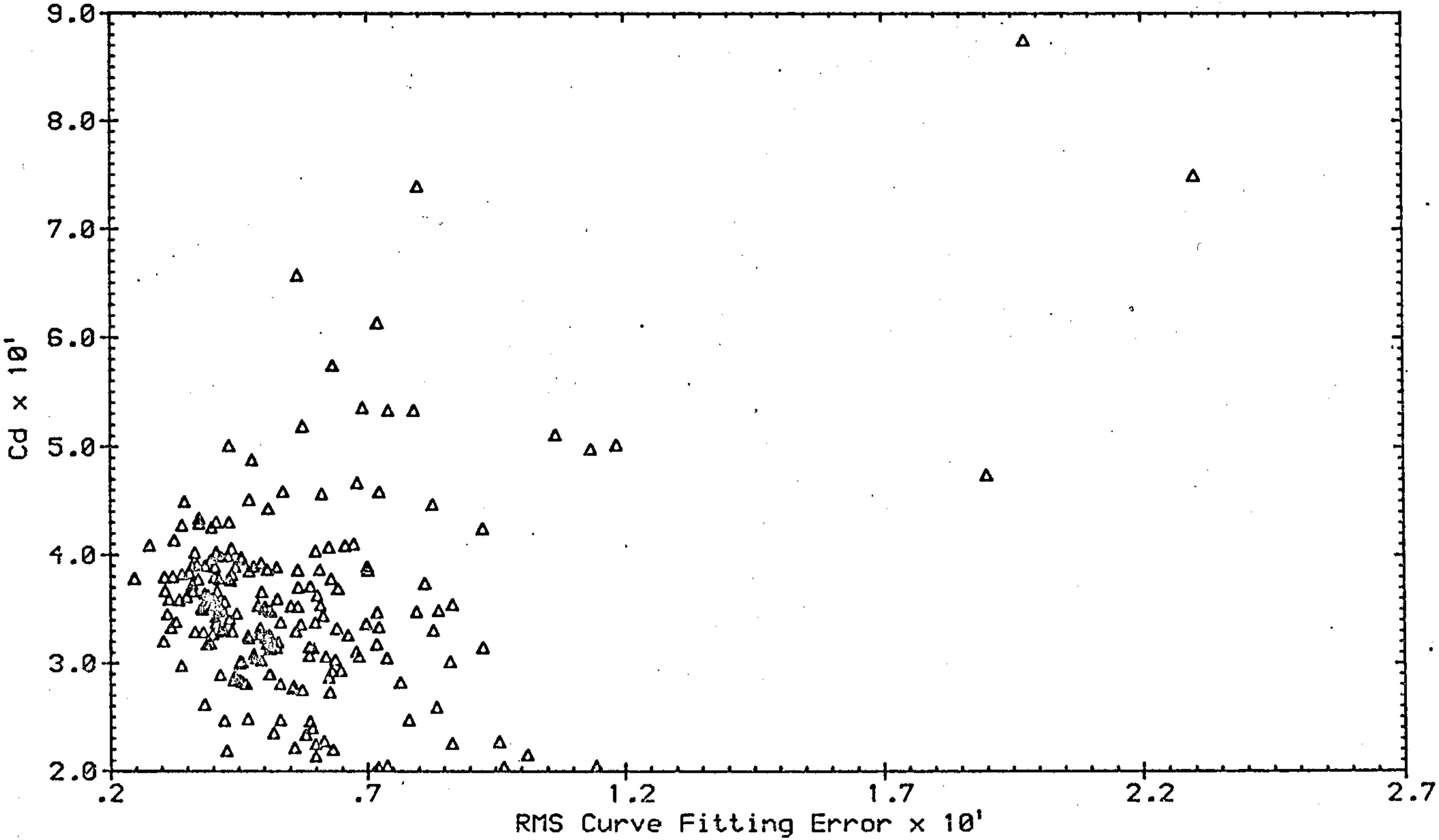
TuFty.Plot

Fig 7.1



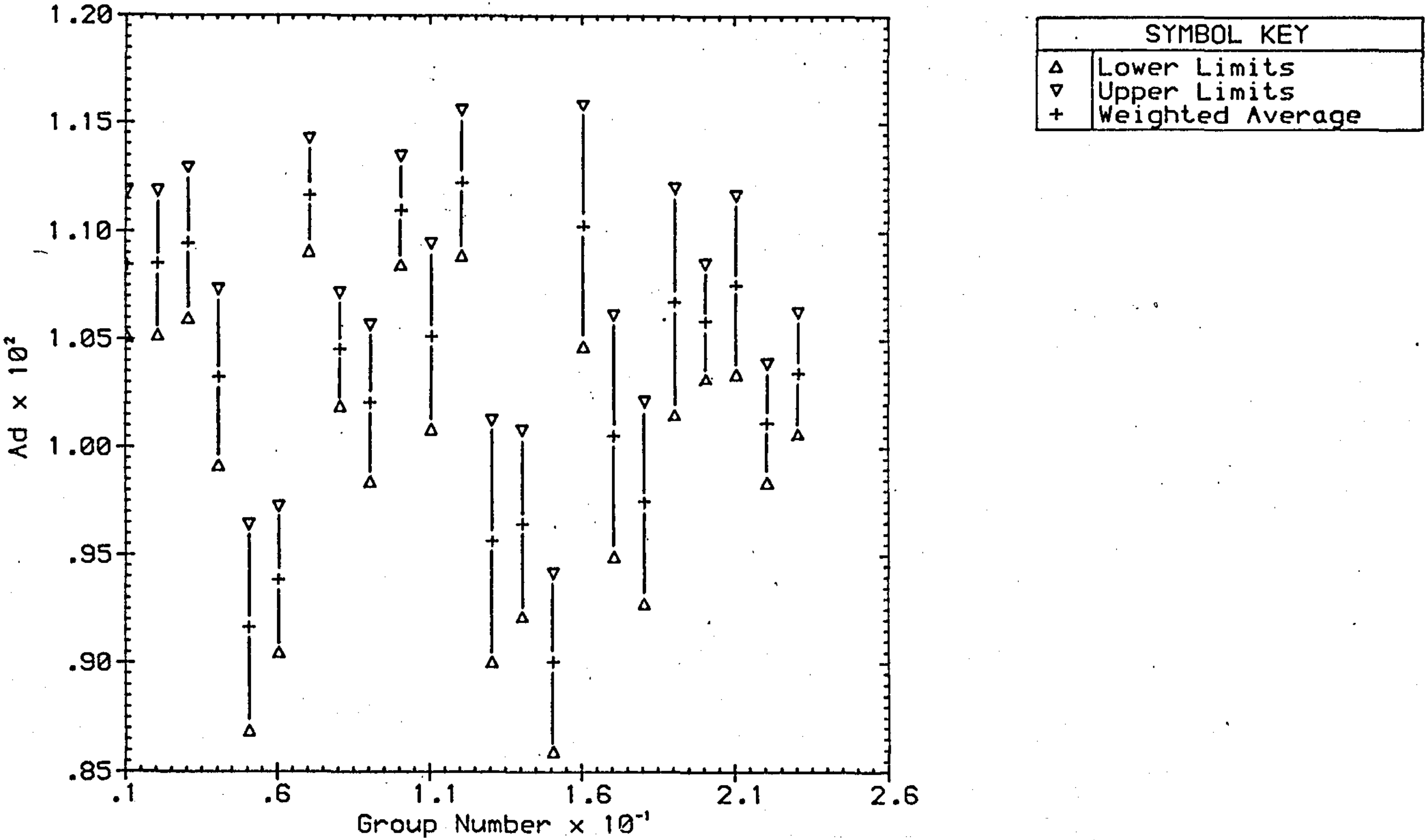
Bd vrs RMS CURVE FITTING ERROR (CORRECTED DATA)

Fig 7.2



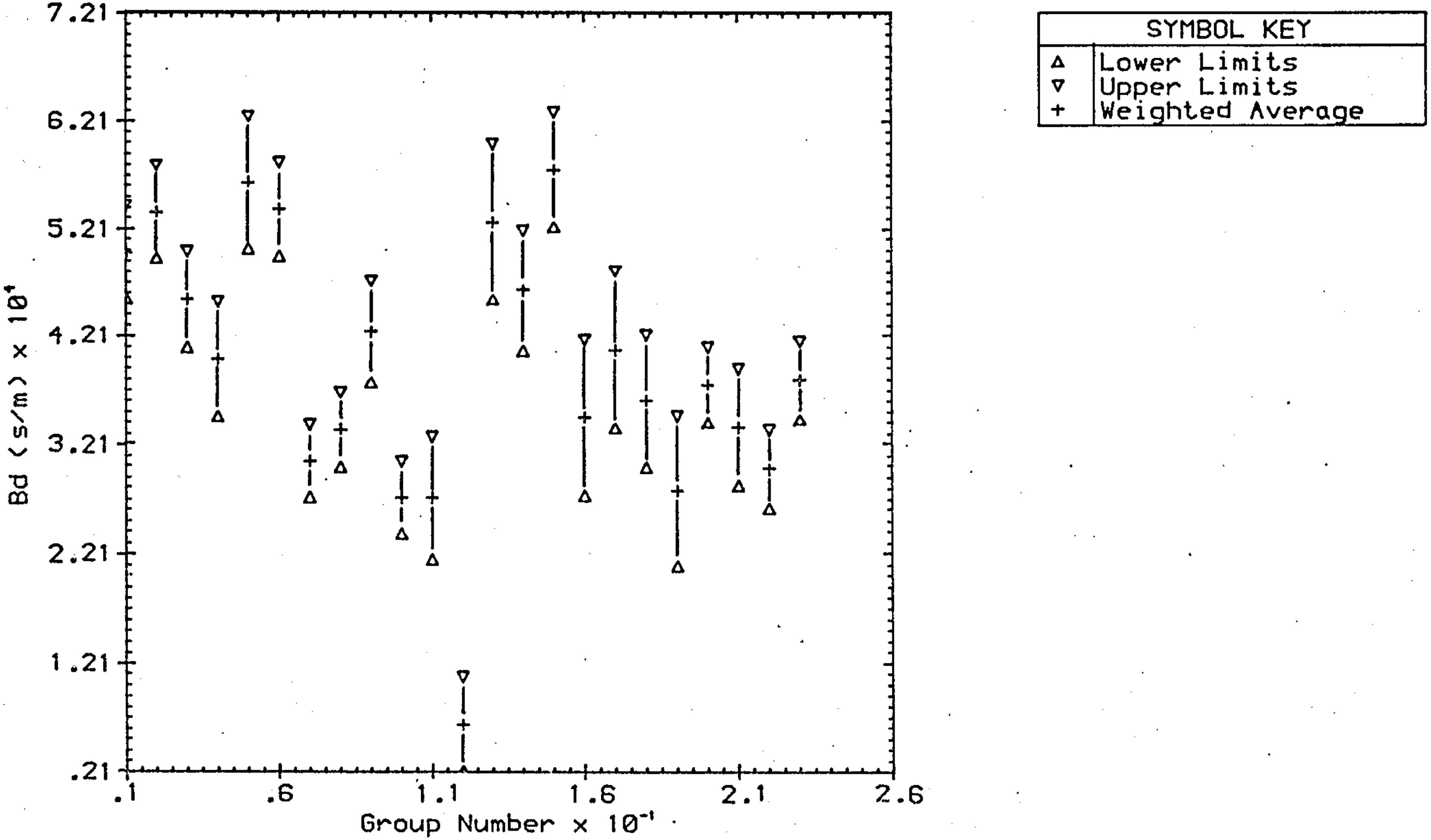
Cd vrs RMS CURVE FITTING ERROR (CORRECTED DATA)

TuFtu.Plot



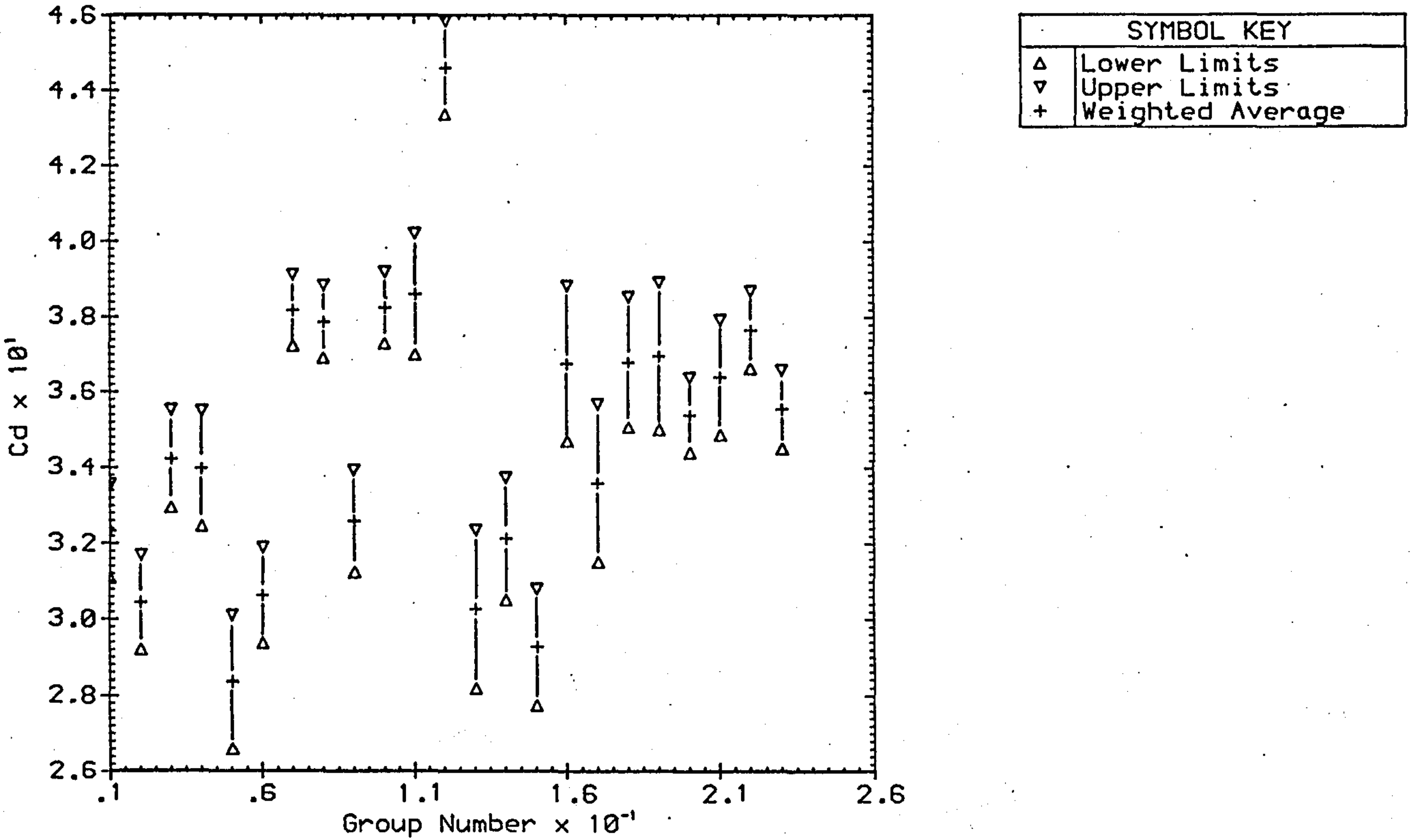
95% CONFIDENCE LIMITS OF Ad vrs GROUP NUMBER

Tufty-Plot



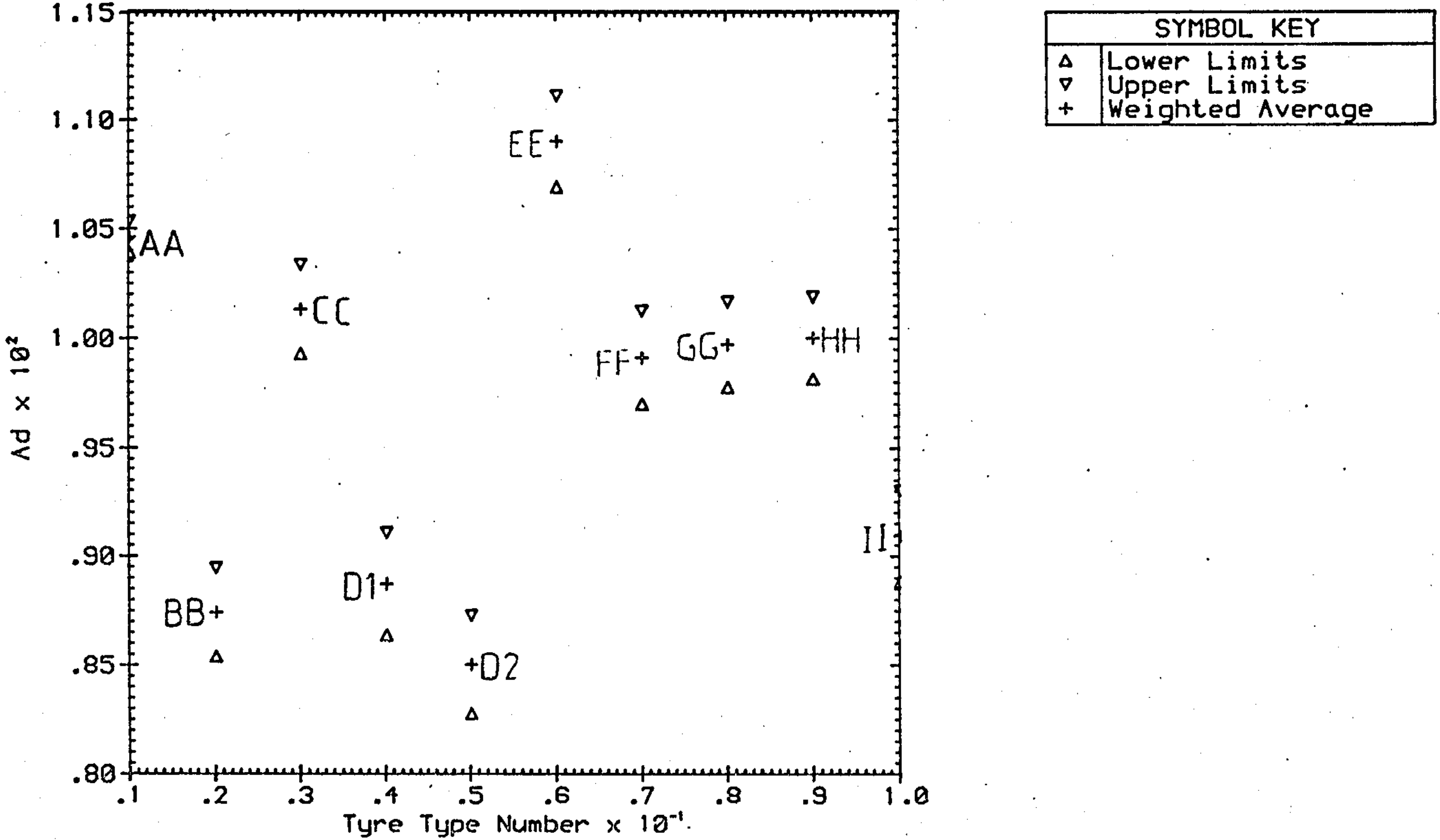
95% CONFIDENCE LIMITS OF Bd vrs GROUP NUMBER

TuFty.Plot



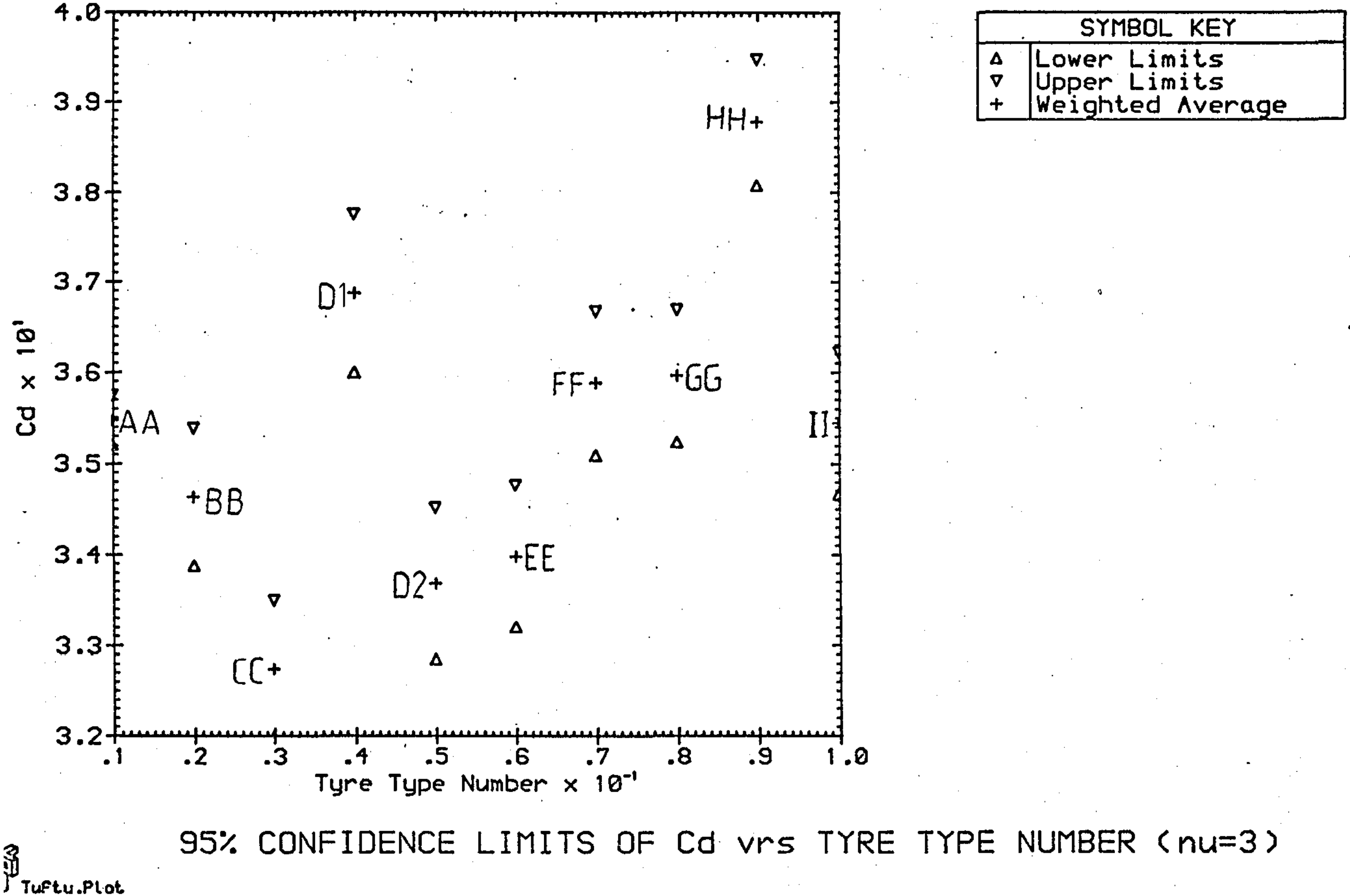
95% CONFIDENCE LIMITS OF Cd vrs GROUP NUMBER

Fig 7.6



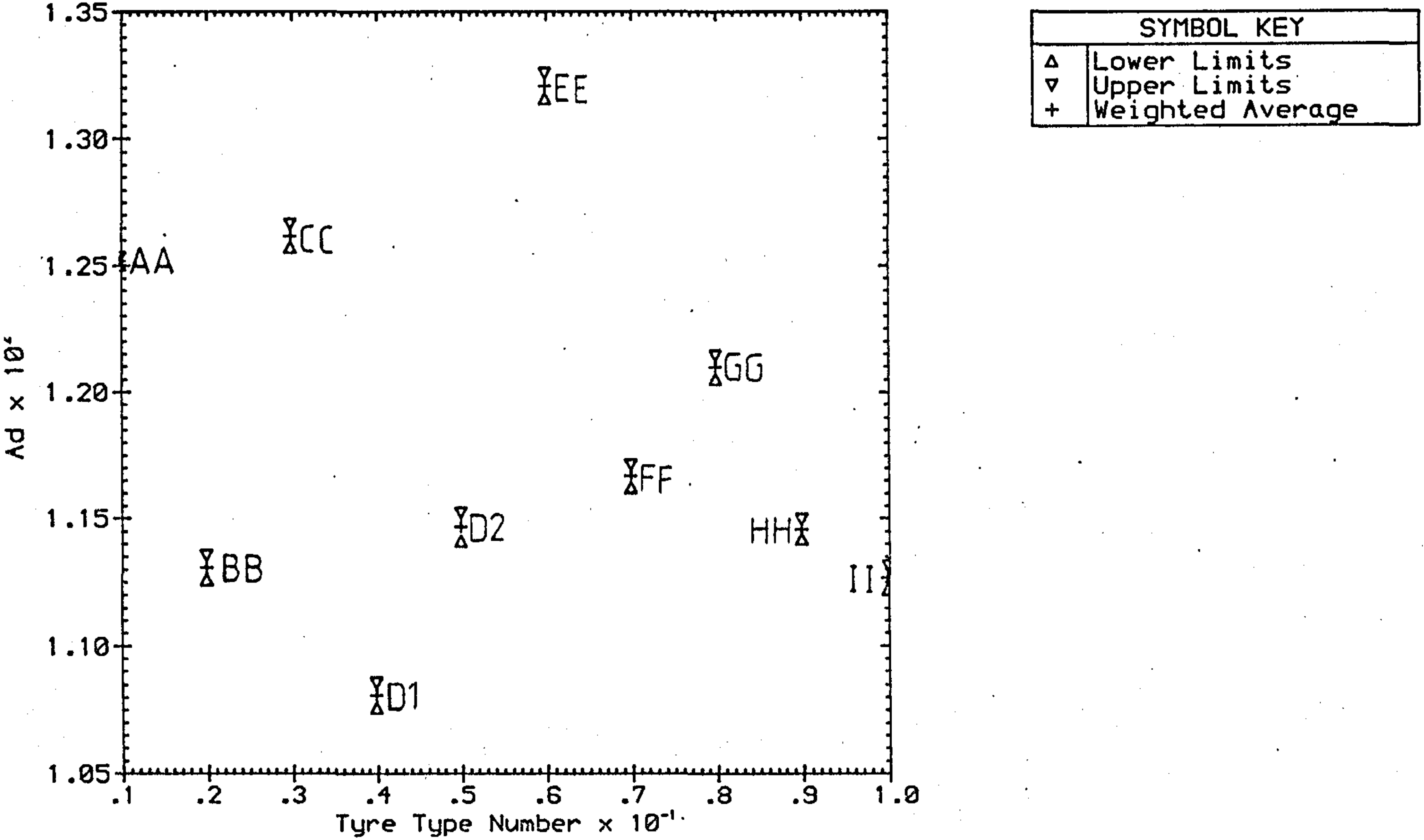
95% CONFIDENCE LIMITS OF Ad vrs TYRE TYPE NUMBER (nu=3)

TuFty.Plot



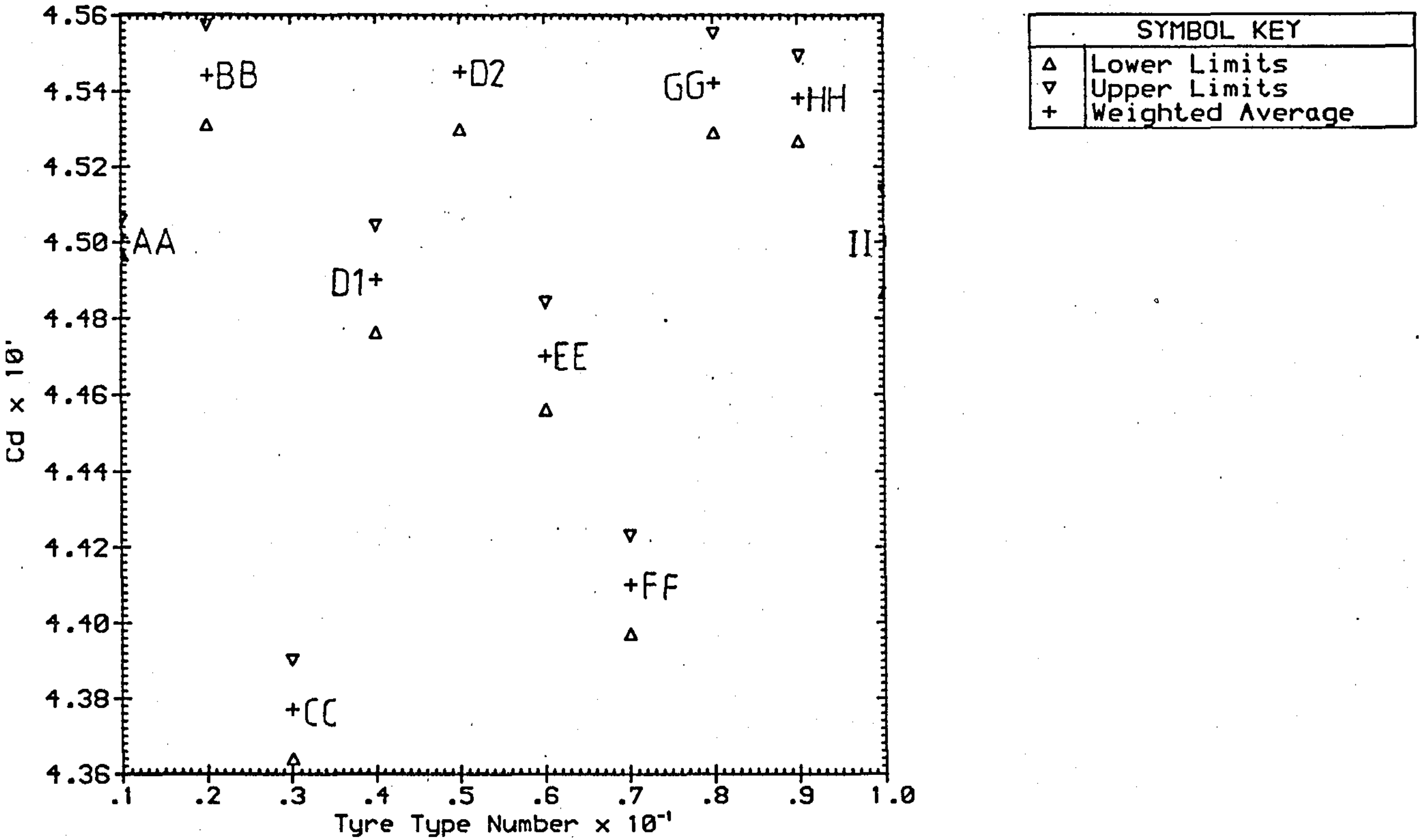
95% CONFIDENCE LIMITS OF Cd vrs TYRE TYPE NUMBER (nu=3)

Fig 7.8



95% CONFIDENCE LIMITS OF Ad vrs TYRE TYPE NUMBER (nu=2)

Fig 7.7



95% CONFIDENCE LIMITS OF Cd vrs TYRE TYPE NUMBER (nu=2)

Tufty.Plot

Fig 7.10