SOFTWARE FOR ME 251

A report written by
Diana Rincon

May 11, 1988

UM-MEAM REPORT #88-04

PREFACE

The work assignment that led to this report, was completed in the Department of Mechanical Engineering at the University of Michigan from January 7, 1988 to May 12, 1988.

I would like to express my appreciation to the following people for assisting me during the course of this assignment.

Professor Kannatey-Asibu for his guidance and support.

Xiangying Liu for his indispensable help on non-linear least squares methods.

Mr. Don Orser from the National Bureau of Standards for sharing his knowledge of phase diagrams.

TABLE OF CONTENTS

I. INTRODUCTION

ME 251, Mechanical Behavior of Solids , is a very important course in the undergraduate program in mechanical engineering since the general concepts in materials science taught in this course are vastly used by mechanical engineers.

Due to the course's importance, learning tools are a welcome commodity. It is therefore the objective of this assignment is to implement various areas covered in ME 251 in software packages that students can use to understand basic concepts. The report on this assignment is organized as follows:

Chapter II discusses the area of Stress-Strain Relations.

Phase Diagrams are reviewed in Chapter III.

Creep and Viscoelasticity are presented in Chapter IV.

Chapter V deals with the successful implementation of Mohr's Circle.

Chapter VI list the conclusions based on the work of this assignment.

II. STRESS-STRAIN RELATION

The software developed for the Stress-Strain Relations computes both the Engineering and True Stress-Strain values and other related parameters from tensile test data. This program also plots the Engineering and True Stress-Strain Curves.

The structure of the program is shown on page 3. The most important sections include inputting the data, computing specific aspects of the deformation process during a tensile test, and graphically showing the relationships between Stress and Strain. Other complementary sections are printing the current data, saving the current data, and editing the data. The complete program listing along with an example demonstrating the capability of this program is included in the appendix.

A. INPUT DATA

A set of data is shown in Table 1. The variable Reading defines a particular data point as either the 'INITIAL' point, the 'YIELD' point, the 'ULTIMATE' point, the 'FRACTURE' point, or simply another point (Reading[DataPoint]='READING'). The program takes into account that all the options for Reading with the exception of 'READING', can only occur once in a particular data set. Consequently the user is not allowed to define more than one Data Point as such. The other variables, Load, Length and Diameter are self explanatory. However, the user should be cautioned that the units for these variables are not arbitrary. The Load, Length and Diameter should be expressed in kN, mm, and mm respectively.

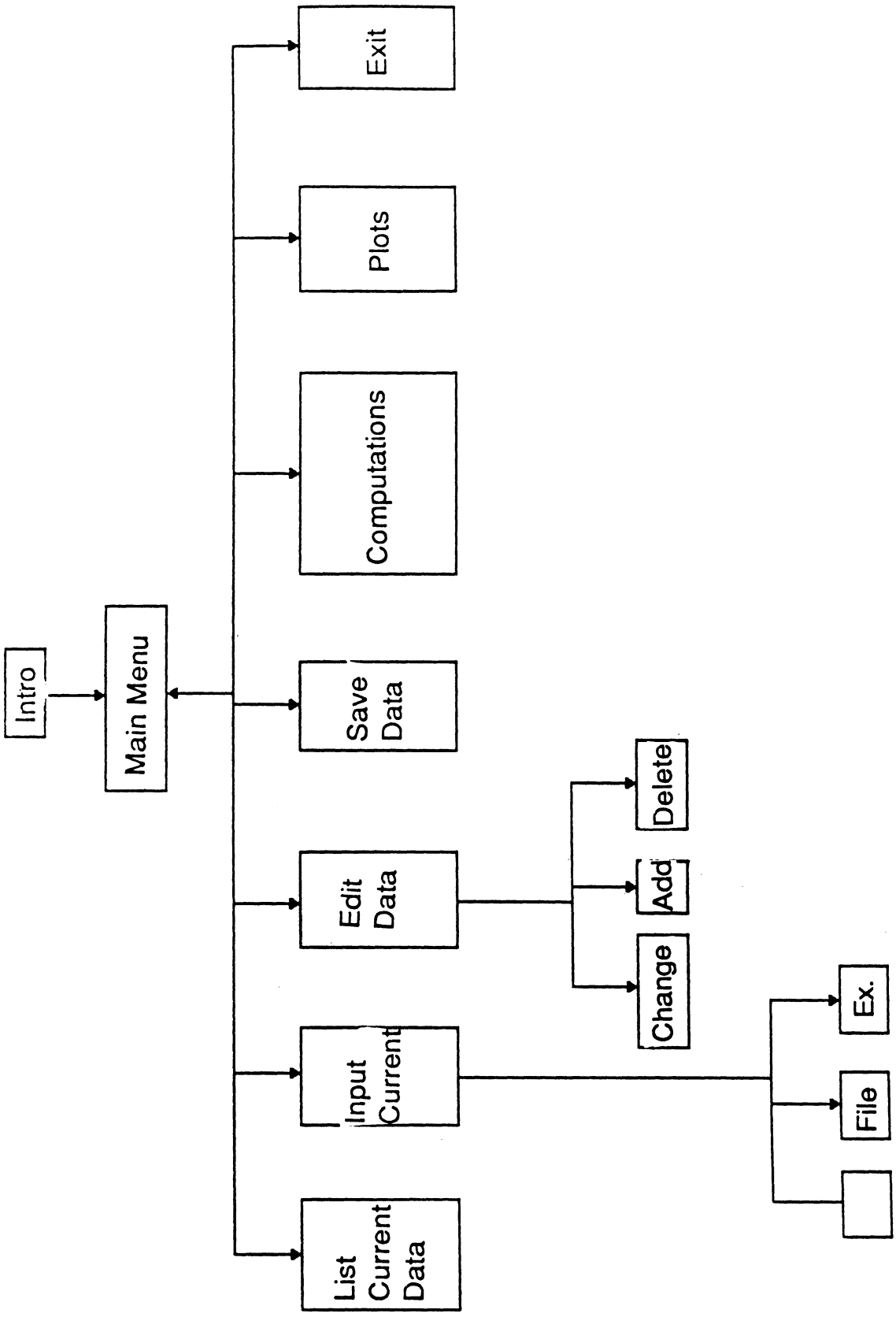
The current values are:

		Load, kN	Length, mm	Diameter, mm
1.	Reading	0.0000	50.0000	12.8000
2.	Reading	40.0000	52.9000	12.6000
3.	Reading	49.8000	56.9000	12.1000
4.	Reading	50.8000	65.3000	11.0000
5.	Fracture	46.3000	69.1000	9.4500

Table 1. Set of Data

The User can input the test data directly or from a file. The format for the input file is one variable input per line, as follows:

PROGRAM STRUCTURE



Line 1: Reading [DataPoint1]
Line 2: Load [DataPoint1]
Line 3: Length [DataPoint1]
Line 4: Diameter[DataPoint1]
Line 5: Reading [DataPoint2]
etc.

The maximum number of DataPoints in each set of data is 20. If a particular variable of a Data Point is unknown, the variable is set equal to 0.1E-50 (or Undefined). When using an input file, any unknowns should be included as either 0.1E-50 or Undefined. When inputting the data directly, the user needs only to press CR (Carriage Return) and the program automatically assumes unknown to be Undefined.

The set of data shown in Table 1, is used as the example data in the program. The user can load this set of data by choosing the "Input Data" selection of the Main Menu and consequently the "Load Example Data option".

B. CALCULATIONS

The different calculations performed by this program are listed below:

1. Initial Conditions
2. Stresses and Strains
3. K and n
4. Stress and Strain at Ultimate
5. Maximum Load
6. Maximum Nominal Strain
7. Modulus of Elasticity

All the calculations make the use of the INITIAL DATA, i.e.,. the load, length, and the diameter before the test is run is used as a reference point in order to measure different aspects of the deformation process. IF no data point has been defined as 'INITIAL', i.e.,. Reading[DataPoint]='INITIAL', the data point with load of zero or the point corresponding to DataPoint=1 will be assumed to define the initial data.

The Stresses and Strains section computes both the engineering and true stresses and strains at each data point. The basic equations used in this section are as follow.

	Stress
Engineering	s=Load[DataPoint]/Area[Initial]
True	σ=Load[DataPoint]/Area[DataPoint]
	Strain
Engineering	e=(Length[DataPoint]-Length[Initial])/Length[Initial]
True (in terms of Diameters)	ε=Ln(Area[Initial]/Area[DataPoint])
True (in terms of Lengths)	ε=Ln(Length[DataPoint]/Length[Initial])

Although both True Strain in terms of Diameters and in terms of Lengths is calculated for each Data point, such calculations are not always appropriate and should be used with discretion.

The K and n parameters used in the mathematical model describing the relationship between the true Stress and true strain in the plastic region, i.e.,. $\sigma=K\varepsilon^n$, are computed from calculated true Stresses and Strains of two different Data Points. The selection of the two data points is crucial in obtaining accurate parameters for the model. The user is reminded that both data points to be selected should occur in the region between Yield and Ultimate. However, the program allows the user to select any data point meeting the following conditions:

-the computed True Stress is defined and not equal to zero
-and the computed True Strain (either in terms of Lengths or Diameters) is defined and not equal to zero

The Stresses and Strains At ULTIMATE section compute the Engineering Stress and both Engineering and True Strains at Ultimate from the calculated values of K and n. The equations used are:

-Engineering Stress= $K*(n/2.718282)^n$
-True Strain = n
-True Stress = $K*n^n$

The same aspects of the deformation process as determined directly from a Data point defined as 'Ultimate', if such data point exists, will be printed with the above results.

The Maximum Load is computed by this equation:

Maximum Load= (Eng. Stress at Ultimate)*Area[Initial]

The Modulus of Elasticity, E, is calculated from the approximation of the Stress-Strain Relation in the Elastic region (s=Ee). The user is asked to select a Data point for this calculation. This point should lie in the Elastic Region, although the program will allow the selection of any point whose Engineering Stress is defined and not equal to zero.

C. PLOTTING

The plotting routine uses graphics. The user is able to fully label the plots or he can press Return to use the default labels. There is also an option to display Grid. In order to print the plots, the word GRAPHICS must be typed before running the program.

The Engineering Stress-Strain Curve section plots only the Data points. The True Stress-Strain Curve section plots the Data Points and the mathematical model using K and n.

D. COMPLEMENTARY SECTIONS

The Print Current Data, Save Current Data, and Edit Current Data sections are self-explanatory. Print Current Data displays the current data on the monitor. Save Current Data allows the user to create and name an input file of the format described above. Edit Current Data allows the user to manipulate his current data. The three subsections here are Edit an Individual Reading, Add a Reading, and Delete a Reading. All of these features provide versatility for the user.

III. PHASE DIAGRAMS

The area of implementing phase diagrams in software was carefully researched. The main conclusions of this research are:

1. Software is scarce, but data needed for such implementation abounds.
2. There is material that describes how to develop software for the implementation of phase diagrams.
3. A particular Software developed by the National Bureau of Standards appears very comprehensive.

Although software does not seem as readily available, the data needed to create the software is plentiful. This information can be found in different handbooks and magazines available at the library.

Among the material describing how to implement phase diagrams in computer software is the book Computer Modelling of Phase Diagrams and the article Computer Calculation of Phase Diagrams of Iron Alloys by Nishizawa and Hasebe. The article in Japanese can be found in the Appendix.

The Software sponsored by the National Bureau of Standards can be described as an "Interactive Computer Graphics Program for Storing Phase Diagrams". The Phase Diagrams and the data used to create them can be found in the Bulletin of Alloy Phase Diagrams. For a more detailed description of this program see the Appendix under PHASE DIAGRAMS.

IV. CREEP AND VISCOELASTICITY

The software developed for this area computes the Modulus of Elasticity, E, and the viscosity coefficients, η, for the Maxwell, the Voigt and the Maxwell-Voigt models. The program also plots the different behaviors associated with each model and with Stress Relaxation for the Maxwell model.

The Structure of the program closely follows the structure of the Stress-Strain Relation Program and will not be discussed in detail. However, the complete program listing along with an example demonstrating the capability of this program is included in the Appendix.

A. MODELS

The MAXWELL Model consists of a spring and damper in series. Its governing equation is $\dot{\varepsilon}=(\tau/\eta_1)^{1/q}+(\dot{\tau}/E_1)$, where q is a constant of non-proportionality. If q=1, the solid is called Newtonian.

The VOIGT Model is a spring and damper in parallel. The governing equation is $\tau=(E_1*\varepsilon)+\eta_1*(\dot{\varepsilon})^q$.

The MAXWELL-VOIGT Model is a combination of both the Maxwell Model and the Voigt Model. It consists of a spring in series with a spring and damper in parallel, all in series with a damper.

B. IMPORTANCE OF FINDING PARAMETERS E AND η

The Importance of finding these parameters can best be observed with the aid of an example. Considering the Maxwell Model for a newtonian solid and assuming a constant stress (τ^*), the governing equation becomes $\varepsilon=t^*(\tau^*/\eta_1)^{1/q}+(\tau^*/E_1)$. The total true strain is the sum of the elastic strain and the viscous (flow) strain, i.e., $\varepsilon_{total}=\varepsilon_{elastic}+\varepsilon_{flow}$. Therefore, it is apparent that $\varepsilon_{elastic}=\tau^*/E_1$ and $\varepsilon_{flow}=(\tau^*/\eta_1)*t$, where t is time. Combining the two equations, $\varepsilon_{flow}=(E_1*t/\eta_1)*\varepsilon_{elastic}$. Now knowing E_1/η_1, the following conclusions can be made.

1. IF E_1/η_1 is large then $\varepsilon_{elastic}<<\varepsilon_{flow}$.
2. IF E_1/η_1 is small then $\varepsilon_{flow}<<\varepsilon_{elastic}$.

3. IF E_1/η_1 is comparable to 1/t then $\varepsilon_{flow} \approx \varepsilon_{elastic}$.

C. INPUT

A set of input data is shown on Table 2. The True Stress is constant at every point. The program only considers the models under constant stress, i.e.,. only a stress step input. The user will only be asked once what the step input is. If it is a negative step input, the user must specify so through the use of a negative sign. The other variables are self-explanatory.

The current values are:

Application	Time,ks	Stress,MPa	Strain
1. Pos Step	0.0000	100.0000	0.2000
2. No Change	0.0200	100.0000	1.0000
3. No Change	0.0400	100.0000	1.2000
4. No Change	0.1000	100.0000	1.4000

Table 2. Set of Data

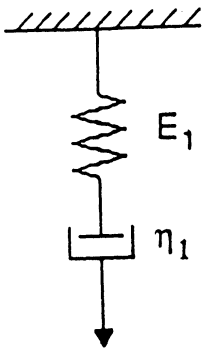
D. CALCULATIONS

The only calculations involved for this software implementation are solving for the E's and τ's. The methods use in the calculations are either the Linear Least Squares Method or the Non-Linear Least Squares Method, depending on which is appropriate.

After the parameters have been computed and shown on the monitor, the behavior of the solid according to the chosen model will be graphically shown.

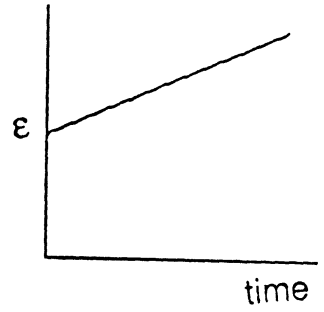
The Menu for the Calculations is shown on page 11. The user can choose to ask the program to solve for any number or all parameters in each model. If the program must compute less parameters than the total number of parameters in the chosen model, the user will be ask to input the rest of the parameters. Stress Relaxation is the behavior of a solid. It eliminates creep, i.e.,. ε=constant or ε=0. The maxwell model has been chosen to illustrate Stress Relaxation for its simplicity. This section does not involve any calculations different from those discussed above.
All equations used in the calculations and graphical representations of the solid's behavior according to each model are summarized on the following page.

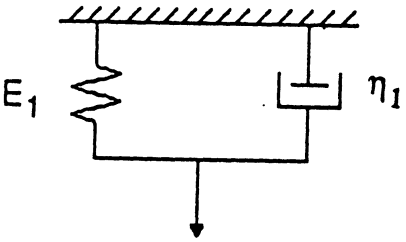
MODELS FOR RATE AND TEMPERATURE DEPENDENT BEHAVIOR



Maxwell Model

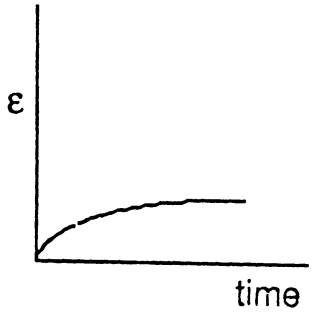
$$\varepsilon = t^*(\tau^*/\eta_1)^{1/q} + (\tau^*/E_1)$$

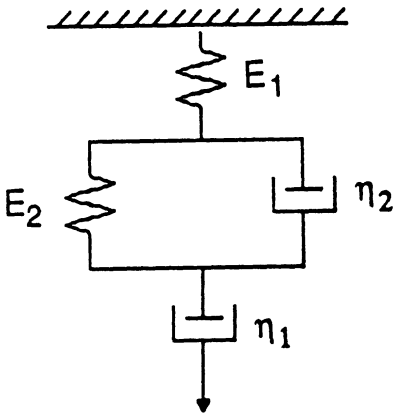




Voigt Model

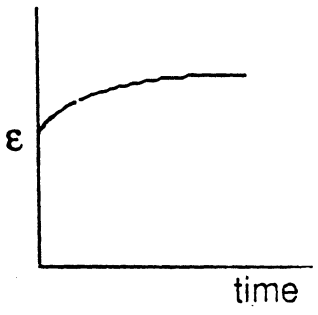
$$\varepsilon = \tau^*/E_1^*(1-EXP(-E_1^*t/\eta_1))$$





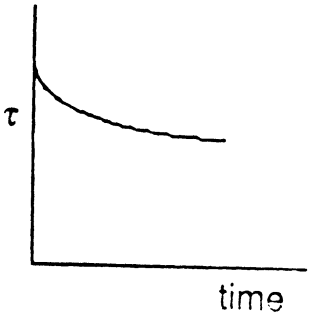
Maxwell-Voigt Model

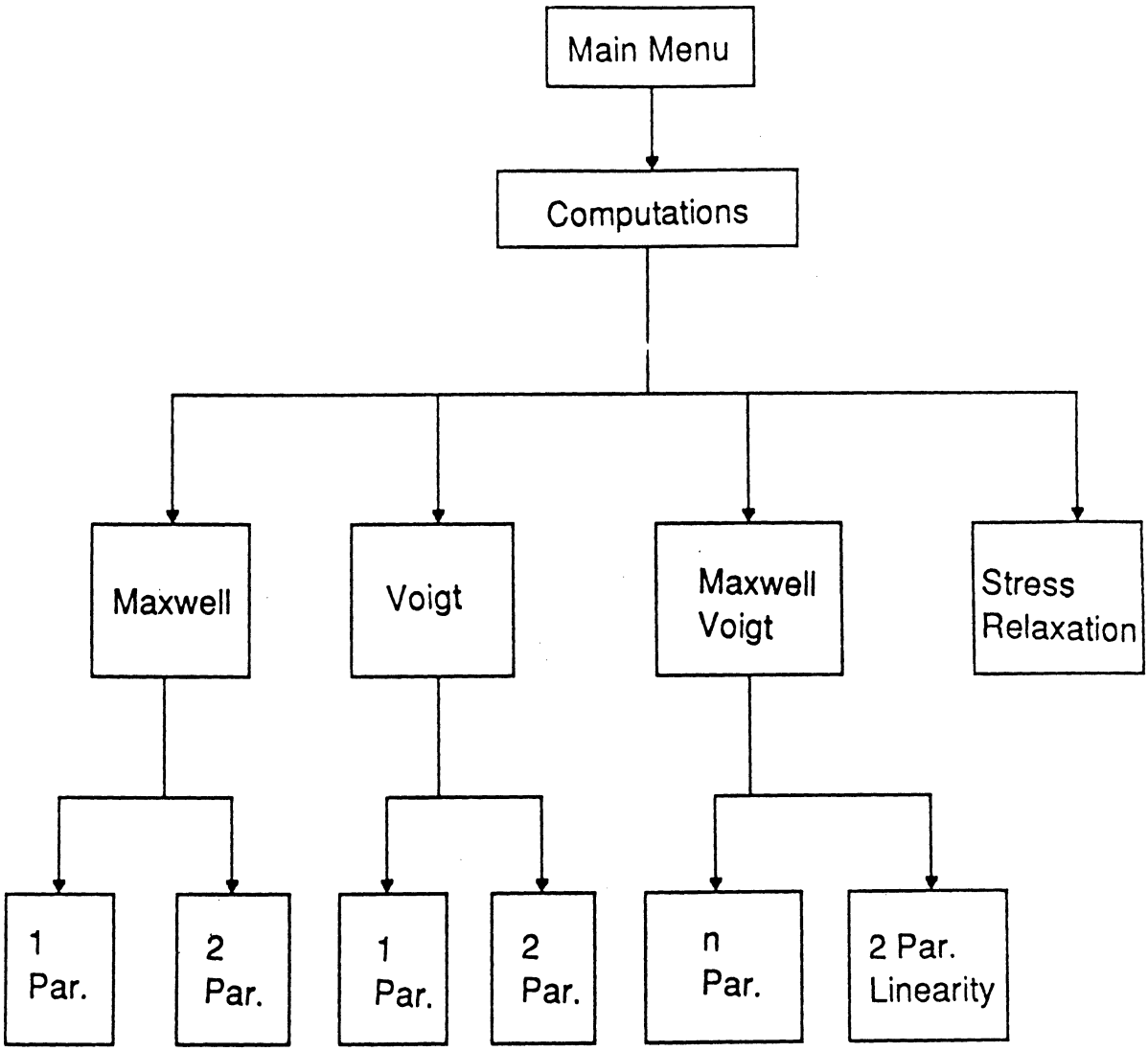
$$\varepsilon = \tau^*/E_1 + (\tau^*/\eta_1)^*t + \tau^*/E_2^*(1-EXP(-E_2^*t/\eta_2))$$



Stress Relaxation
(Maxwell Model)

$$\tau = \tau^{**}EXP(-E_1^*t/\eta_1)$$




Main Menu
Computations
Maxwell
Voigt
Maxwell
Voigt
Stress
Relaxation
1
Par.
2
Par.
1
Par.
2
Par.
n
Par.
2 Par.
Linearity

V. TWO DIMENSIONAL MOHR'S CIRCLE

This software is used to resolve stresses and strains in directions different from those along which stresses and strain are known. The results are graphically shown through the use of Mohr's Circle.

The structure of this program is almost identical to the structure of the Stress-Strain Relation program. However since the amount of data needed to run this program is minimal, the use of input files has been omitted, and the editing features vastly simplified. The complete program listing along with an example is included in the Appendix.

A. INPUT DATA

The data needed to run this program is all the variables, Sx, Sy, τxy of the general two dimensional system. This general two dimensional system is shown in the example in the Appendix. It is also available for viewing on the monitor through the program. The sign convention employed is shear stresses producing Clockwise system rotation are positive, and Counterclockwise rotations of the system define positive angle Theta.

B. CALCULATIONS

The Calculations produced by this program include:

- -the principal stress and its plane directions
- -the maximum shear stress and its plane directions
- -the shear and normal stresses for a specified angle Theta
- -the Mohr's Circle

The Principal Stress and its plane directions are simple to find. Using the rules for resolving stress and noting that the shear stress is zero along the principal planes, the principal stresses are:

$$s_1 = 1/2(Sx+Sy) + 1/2[(Sy-Sx)^2+4\tau xy^2]^{1/2}$$

$$s_2 = 1/2(Sx+Sy) - 1/2[(Sy-Sx)^2+4\tau xy^2]^{1/2}$$

The principal planes are:

$$Theta_1 = (1/2)*ARCTAN(2*\tau xy/(Sy-Sx))$$

$$Theta_2 = Theta_1 + 90$$

The maximum Shear Stress is:

$$\tau max = 1/2[(Sy-Sx)^2+4\tau xy^2]^{1/2}$$

The normal and true stresses on an inclined plane at Theta degrees from the horizontal are:

$$s=1/2(Sy+Sx)+1/2(Sy-Sx)*COS(2*Theta) +\tau xy*SIN(2*Theta)$$

$$\tau=1/2(Sy-Sx)*SIN(2*Theta)-\tau xy*COS(2*Theta)$$

The direction of the planes at which the maximum shear stress occurs is:

$Theta_3 = Theta_1+45$

$Theta_4 = Theta_2+45$

The Mohr's circle is a very useful graphical representation of all the stress states as Theta is varied. The circle has its center at $(1/2(s_1+s_2,0)$ and radius of 1/2 (s_1-s_2). The user is reminded that any angle on the circle corresponds to twice the angle Theta on the real system. The circle is plotted and shown on the monitor. Notice that the most important points are denoted by X. Also the user can opt to connect Sx to Sy. This feature will allow the user to quickly locate the center of the circle.

VI. CONCLUSION

The following conclusions are derived from the work described above.

1. The implementation of the Stress-Strain Relations on a software package was successful. The program is very user-friendly and it will aid any student to comprehend the subject matter.

2. Further research on the area of the implementation of phase diagrams should be conducted.

3. Creep and Viscoelasticity is one area that will greatly benefit from the developed software. Students interested in this area will have a better understanding of the behavior of solids without consuming any time at all on tedious calculations.

4. Perhaps the area where software implementation is useful is Mohr's Circle Transformations. By using this software, any student can visualize the stress states with greater ease.

APPENDIX

STRESS-STRAIN RELATION

EXAMPLE

The current values are:

		Load, kN	Length, mm	Diameter, mm
1.	Reading	0.0000	50.0000	12.8000
2.	Reading	40.0000	52.8000	12.6000
3.	Reading	49.8000	56.9000	12.1000
4.	Reading	50.8000	65.3000	11.0000
5.	Fracture	46.3000	69.1000	9.6500

The Initial Dimensions are:

Length mm	Diameter mm	Area sqmm
50.0000	12.8000	128.6796

Reading	Eng Stress MPa	Eng Strain	True Stress MPa	True Strain ln(Li/Lo)	True Strain 2ln(Do/Di)
1. Reading	0.0000	0.0000	0.0000	0.0000	0.0000
2. Reading	310.8495	0.0560	320.7961	0.0545	0.0315
3. Reading	387.0076	0.1380	433.0806	0.1293	0.1125
4. Reading	394.7789	0.3060	534.5502	0.2670	0.3031
5. Fracture	359.8083	0.3820	633.0478	0.3235	0.5650

The current values are:

	Load, kN	Length, mm	Diameter, mm
1. Reading	0.0000	50.0000	12.8000
2. Reading	40.0000	52.8000	12.6000
3. Reading	49.8000	56.9000	12.1000
4. Reading	50.8000	65.3000	11.0000
5. Fracture	46.3000	69.1000	9.6500

Select First Reading for Calculation of K & n:3

The current values are:

	Load, kN	Length, mm	Diameter, mm
1. Reading	0.0000	50.0000	12.8000
2. Reading	40.0000	52.8000	12.6000
3. Reading	49.8000	56.9000	12.1000
4. Reading	50.8000	65.3000	11.0000
5. Fracture	46.3000	69.1000	9.6500

Select Second Reading for Calculation of K & n:4

n	K
0.2124	688.7700

Stresses & Strains At Ultimate

alculated from K & n:			Determined from Reading:		
ng Stress	True Stress	True Strain	Eng Stress	True Stress	True Strain
400.8212	495.6482	0.2124			

Maximum Load, kN:

Calculated from Su estimate	Calculated from Su reading
51577.5291	

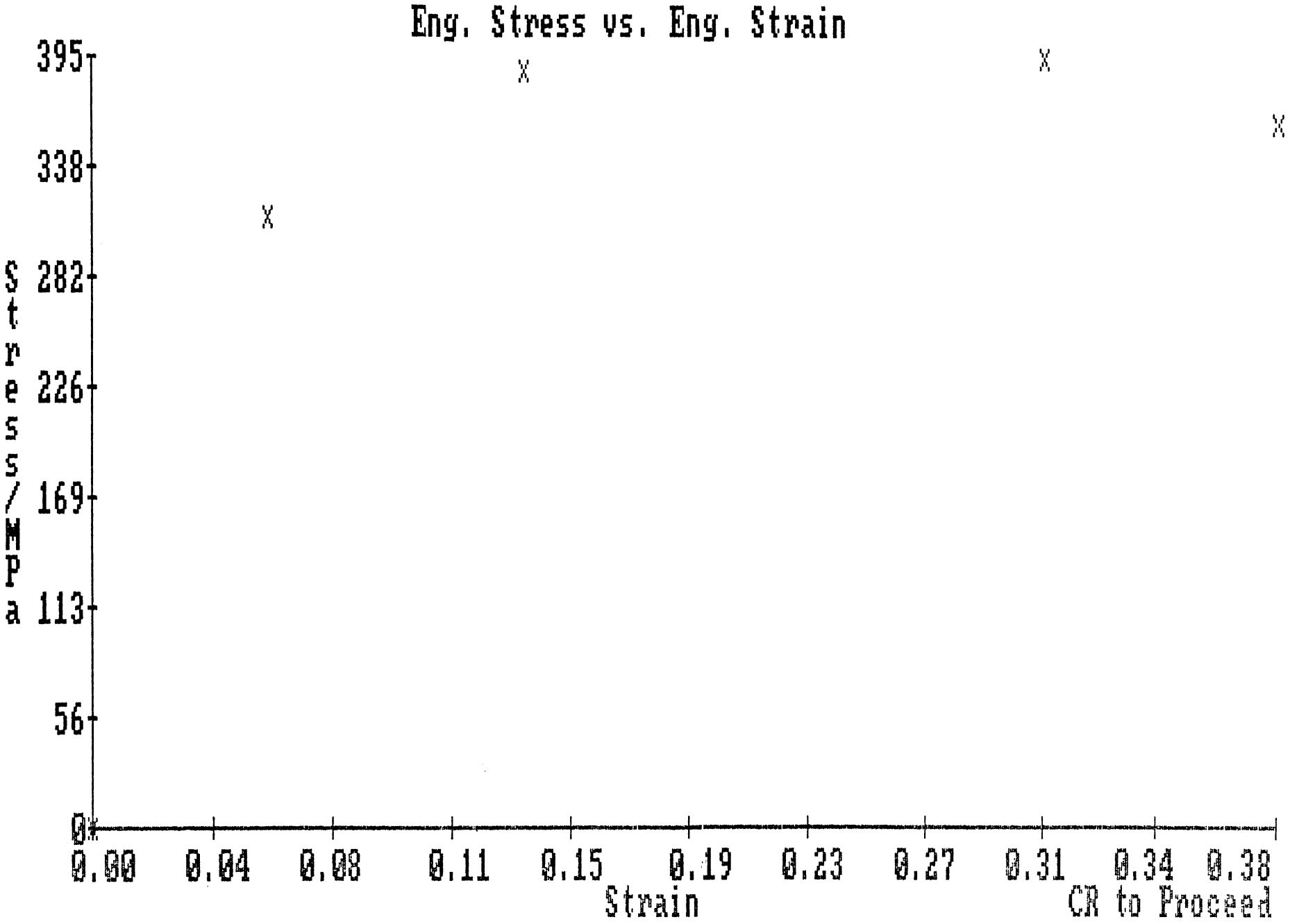
The maximum Engineering Strain is
0.3820

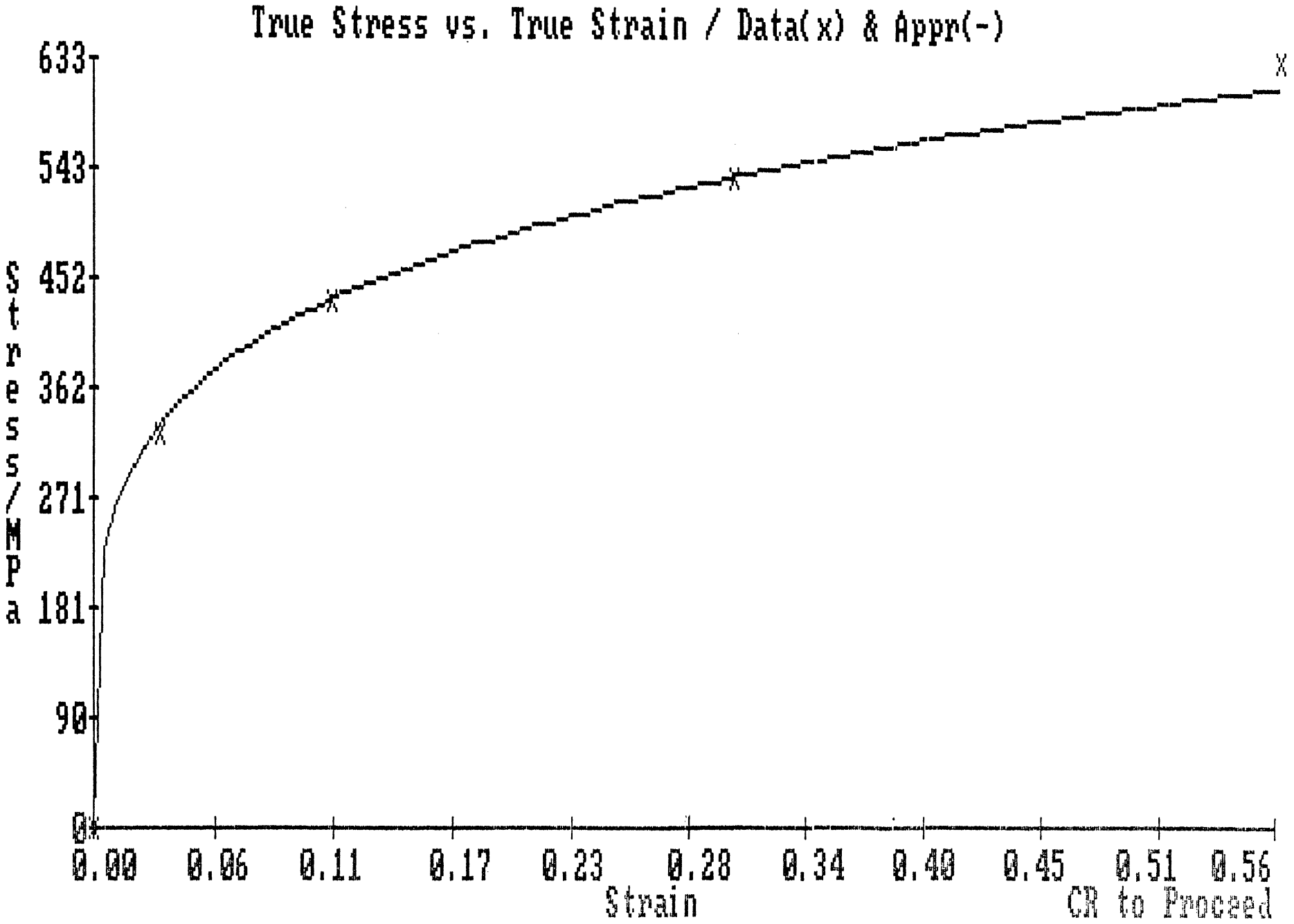
The current values are:

		Load, kN	Length, mm	Diameter, mm
1.	Reading	0.0000	50.0000	12.8000
2.	Reading	40.0000	52.8000	12.6000
3.	Reading	49.8000	56.9000	12.1000
4.	Reading	50.8000	65.3000	11.0000
5.	Fracture	46.3000	69.1000	9.6500

Select Reading for Calculation of E:2

The Modulus of Elasticity is
5550.8839


Eng. Stress vs. Eng. Strain
395
338
282
226
169
113
56
0
Stress/MPa
0.00
0.04
0.08
0.11
0.15
0.19
0.23
0.27
0.31
0.34
0.38
Strain
CR to Proceed


True Stress vs. True Strain / Data(x) & Appr(-)
633
543
452
362
271
181
90
0
Stress/MPa
0.00
0.06
0.11
0.17
0.23
0.28
0.34
0.40
0.45
0.51
0.56
Strain
CR to Proceed

STRESS-STRAIN RELATION

PROGRAM LISTING

LIST OF VARIABLES

Input Data Section

DataPoints	:	Total Number of DataPoints in a Set of Data
I	:	The Number of the individual Data point.
Readings[I]	:	Defines individual Data point as 'Initial', 'Yield', 'Ultimate', 'Fracture' or a regular 'Reading'.
Loads[I]	:	The Load at Data point I.
Lengths[I]	:	The Length at Data point I.
Diameter[I]	:	The Diameter at Data point I.

Calculations Section

EngStress[I]	:	Engineering Stress at Data point I.
EngStrain[I]	:	Engineering Strain at Data point I.
TrueStress[I]	:	True Stress at Data point I.
DiaTrStrain[I]	:	True Strain computed in terms of the Diameters at Data point I.
LenTrStrain	:	True Strain computed in terms of the Lengths at Data point I.
L	:	Initial Length.
D0	:	Initial Diameter.
A0	:	Initial Area.
n,K	:	Parameters in the approximation of the True Stress-Strain relation in the plastic region ($\sigma=K*\varepsilon^n$).
UltEngStressRd	:	Eng. Stress at Ultimate as determined from Data point defined as 'Ultimate'.
UltTrStressRd	:	True Stress at Ultimate as determined from Data point defined as 'Ultimate'.
UltTrStrainRd	:	True Strain at Ultimate as determined from Data point defined as 'Ultimate'.
UltEngStressCal	:	Eng. Stress at Ultimate as calculated from K and n.
UltTrStressCal	:	True Stress at Ultimate as calculated from K and n.
UltTrStrainCal	:	True Strain at Ultimate as calculated from K and n.
MaxLoadRd	:	Maximum Load as determined from Data point defined as 'Ultimate'.
MaxLoadCal	:	Maximum Load as calculated from K and n.
MaxEngStrain	:	Maximum Engineering Strain
E	:	Elastic Modulus.
First	:	First Data point selected by user to calculate K and n.
Second	:	Second Data point selected by user to calculate K and n.
Code	:	Assures selected point will not produce error, ex: +0.

Saving and Loading Input Data Files Section

Indata	:	Declaration of Input File.
Filename	:	Name of Input File.

Plotting Section

Titlename	:	Title for plots.
Titlenames	:	Default Title.
Xlable	:	Lable for X-axis.
Ylable	:	Lable for Y-axis.
GridCode	:	Selection of Grid Option.
Xmax	:	Maximum value needed on the X-axis.
Xmin	:	Minimum value needed on the X-axis.
Ymax	:	Maximum value needed on the Y-axis.
Ymin	:	Minimum value needed on the Y-axis.
Xarray	:	Array containing all X values.
Yarray	:	Array containing all Y values.
XConv	:	Array containing all X values after convertion for plotting.
YConv	:	Array containing all Y values after convertion for plotting.

Main Program

A,Z	:	Integers Used in Selection of Main Menu Options.
Quit	:	Boolean used to quit the program.

```
PROGRAM Stress_Strain;


CONST
  Undefined= 0.1E-50;
  Title    = 14;
  Regular  = 7;
  Highlite = True;
  Normal   = False;
  CR       = True;
  NoCR     = False;
TYPE
  String255 = String[255];
  String80  = String[80];
VAR
  {Input Data}
  Loads,Lengths,Diameters        : Array [1..20] of Real;
  Readings                       : Array [1..20] of string[80];
  I,DataPoints                   : Integer;

  {Calculations}
  EngStress,EngStrain,TrueStress,DiaTrStrain,LenTrStrain  : Array[1..20] of Real;
  L0,D0,A0,n,K,UltEngStressRd,UltTrStressRd,UltTrStrainRd,
  UltEngStressCal,UltTrStressCal,UltTrStrainCal,
  MaxLoadRd,MaxLoadCal,MaxEngStrain,E                    : Real;

  {Selecting Readings for Calculations of K & n and E}
  First,Second                   : Integer;
  Code                           : Boolean;

  {Saving and Loading Data in Files}
  Indata                         : TEXT;
  Filename                       : String[20];

  {Plotting}
  Titlename,Titlenames,Xlable    : String[80];
  Ylable                         : String[20];
  GridCode                       : Integer;
  Xmax,Xmin,Ymax,Ymin            : Real;
  Xarray,Yarray                  : Array [1..20] of Real;
  XConv,YConv                    : Array [1..20] of Integer;

  {Main Program}
  A,Z                            : Integer;
  Quit                           : Boolean;



{Useful procedures throughout the program}

PROCEDURE WriteAt(X,Y : Integer;
                  Highlite,UseCR : Boolean;
                  TheText : string255);
BEGIN
  IF Y < 0 THEN Y := 12;
  IF X < 0 THEN X := (80-Length(TheText)) DIV 2;
  GotoXY(X,Y);
  IF Highlite THEN TextColor(10);
  IF UseCR THEN Writeln(TheText) ELSE Write(TheText);
  TextColor(Regular);
END;

PROCEDURE Proceed;
VAR
  PosY : Integer;
BEGIN
```

```
  WriteAt(-1,25,Highlite,NoCR,'Press Return to Proceed');
  PosY:=WhereY;
  Readln;
  GotoXY(1,PosY);   DelLine;
END;

PROCEDURE Read_Code(VAR Code,Q : Integer);
VAR
  Codestr         : String[80];
  Error,PosX,PosY : Integer;
BEGIN
  PosX:=WhereX;
  PosY:=Wherey;
  REPEAT
  ClrEOL;
  Readln(Codestr);
  IF length(Codestr)=0 THEN BEGIN
  Error:=0;  Q:=45;         END;
  Val(Codestr,Code,Error);
  IF Error<>0  THEN                                BEGIN
    Writeln;
    WriteAt(-1,25,Highlite,NoCR,'Selection type must be an integer');
    Delay(1500);  DelLine;  GotoXY(PosX,PosY);  END;
  UNTIL Error=0;
END;

PROCEDURE Yes_No_Answer(Var Yes_No_Code : Integer);
VAR
  Answer : String[80];
BEGIN
  Readln(Answer);
  IF Length(Answer)=0 THEN Yes_No_Code:=2;
  IF (Answer='yes') OR (Answer='y') THEN Yes_No_Code:=1;
  IF (Answer='no')  OR (Answer='n') THEN Yes_No_Code:=2;
END;


{Header}

PROCEDURE Header;
BEGIN
  Window(15,6,72,25);
  Clrscr;
  TextColor(Title);
  Writeln('************************************************');
  Writeln('*                                              *');
  Writeln('*                                              *');
  Writeln('*            UNIVERSITY OF MICHIGAN            *');
  Writeln('*      Department of Mechanical Engineering    *');
  Writeln('*                  ME  251                     *');
  Writeln('*                                              *');
  Writeln('*          Stress Strain Relationships         *');
  Writeln('*            from Tensile Test Data            *');
  Writeln('*                                              *');
  Writeln('*               by Diana Rincon                *');
  Writeln('*                                              *');
  Writeln('*                                              *');
  Writeln('************************************************');
  TextColor(Regular);
  Window(1,1,80,25);
END;


{Introduction}

PROCEDURE Introduction_choice(Var Intro_Code : Boolean);
```

```
VAR
  Yes_No_Code     : Integer;
  Code_Procedure : Boolean;
BEGIN
  REPEAT
    Intro_Code:=False;
    Write('Do you wish to read the introduction? (no):');
    Yes_No_Answer(Yes_No_Code);
    CASE Yes_No_Code OF
     1: BEGIN   Intro_Code:=True; Code_Procedure:=True;     END;
     2: Code_Procedure:=True;
    ELSE   BEGIN
     WriteAt(-1,25,Highlite,CR,'Must answer yes or no and hit return');
     GotoXY(15,19); DelLine; GotoXY(1,1); InsLine; GotoXY(1,20); InsLine;
     GotoXY(15,20); Code_Procedure:=False;     END;
    END;
  UNTIL Code_Procedure=True;
    Clrscr;
END;

PROCEDURE Introduction;
VAR
  Intro_Code : Boolean;
BEGIN
  Introduction_choice(Intro_Code);
  IF Intro_Code=True THEN   BEGIN
     TextColor(Title);
     Writeln('INTRODUCTION');
     Writeln('This program solves for the ENGINEERING AND TRUE STRESSES AND STRAINS
     Writeln('other important related parameters from Tensile Test Data. ');
     Writeln;
     Writeln('The user can input the Test data directly or from a file. The FORMAT
     Writeln('INPUT FILE is one input per line, as follows: ');
     Writeln('    Line 1-Reading[1]   Line 2-Load[1]   Line 3-Length[1]   Line 4-Di
     Writeln('    Line 6-Reading[2]        etc.');
     Writeln('The VARIABLE READING defines the data point as either the initial poi
     Writeln('yield point, the ultimate point, the fracture point, or simply a non-
     Writeln('point (Reading[DataPoint]=Reading).  The total NUMBER OF DATAPOINTS t
     Writeln('defined in this program is 20.');
     Writeln;
     Writeln('THE OTHER PARAMETERS that can be solved for using this program are:')
     Writeln('the K & N parameters in the mathematical model for the relationship b
     Writeln('true stress and true strain, the Stresses and Strains at Ultimate, th
     Writeln('Load, the Maximum Engineering Strain, and the Modulus of Elasticity.'
     Writeln;
     Writeln('All calculations make use of INITIAL DATA (the load, length, and the
     Writeln('right before the tensile test is run).  IF no data point has been def
     Writeln('Initial, ie. Reading=Initial, the data point with load of zero or the
     Writeln('point first defined will be assumed to be the initial data.');
     Writeln;
     Writeln('To print any graphs, User must type GRAPHICS before running the progr
     Proceed;                    END;
END;


{Initialize Data}

PROCEDURE Initialize_Data_Array;
VAR
  Q : Integer;
BEGIN
  FOR Q:=1 to 20 DO
  BEGIN
  Readings[Q]:='Reading';
  Loads[Q]:=0;
  Lengths[Q]:=Undefined;
```

```
  Diameters[Q]:=Undefined;
  END;
END;


{Input Data}

PROCEDURE Example_Data;
VAR
  Q: Integer;
BEGIN
  FOR Q:=1 to 4 DO BEGIN
  Readings[Q]:='Reading';  END;
  DataPoints:=5;
  Readings[5]:='Fracture';
  Loads[1]:=0; Loads[2]:=40; Loads[3]:=49.8; Loads[4]:=50.8; Loads[5]:=46.3;
  Lengths[1]:=50; Lengths[2]:=52.8; Lengths[3]:=56.9; Lengths[4]:=65.3;
  Lengths[5]:=69.1; Diameters[1]:=12.8; Diameters[2]:=12.6; Diameters[3]:=12.1;
  Diameters[4]:=11; Diameters[5]:=9.65;
END;

PROCEDURE Select_Number_Data_Points;
VAR
  Codestr : String[80];
  Error   : Integer;
BEGIN
  REPEAT
  Error:=0;
  WriteAt(-1,-1,Normal,NoCR,'Select Number of Data Points (3):');
  ClrEOL;
  Readln(Codestr);
  IF Length(Codestr)=0 THEN BEGIN
   Error:=0;   DataPoints:=3;    END;
  Val (Codestr,DataPoints,Error);
  IF DataPoints>20 THEN Error:=1;
  IF Error<>0 THEN   BEGIN
  WriteAt(-1,25,Highlite,NoCr,'Selection type must be an integer less than 20');
  Delay(1000);  DelLine;   END;
  UNTIL Error=0;
END;

PROCEDURE Print_Data;
VAR
  POSY,POSYY : Integer;
BEGIN
  IF DataPoints<=20 THEN BEGIN
  Window(1,1,80,25);
  Clrscr;
  PosY:=(21-DataPoints) DIV 2;
  TextColor(Title);
  WriteAt(-1,PosY,Normal,CR,'The current values are:');
  Writeln; TextColor(Title);
  WriteAt(24,PosY+2,Normal,CR,'Load, kN          Length, mm        Diameter, mm');
  FOR I:=1 to DataPoints DO
  BEGIN
  PosYY:=PosY+3+I;
  GotoXY(6,PosYY);
  TextColor(Title); Write(I,'. ');TextColor(Regular);Write(Readings[I]:9);
  GotoXY(21,PosYY);
  IF Loads[I]<>Undefined   THEN  Write(Loads[I]:10:4);
  GotoXY(41,PosYY);
  IF Lengths[I]<>Undefined THEN  Write(Lengths[I]:10:4);
  GotoXY(60,PosYY);
  IF Diameters[I]<>Undefined THEN Writeln(Diameters[I]:10:4);
  END;                    END
  ELSE
```

```
  Select_Number_Data_Points;
END;

PROCEDURE Select_Reading_Type;
VAR
  Readingstr : string[80];
  Error,Q : integer;
  Duplicate : Boolean;
BEGIN
  REPEAT
  Clrscr;
  Error:=1;
  WriteAt(-1,24,Highlite,NoCR,'Reading Types: '+
                'Initial(I)  Reading(R)  Yield(Y)  Ultimate(U)  Fracture(F)');
  WriteAt(-1,-1,Normal,NoCR,'Select type of Reading input (R):');
  Readln(Readingstr);
  IF Length(Readingstr)=0 THEN
  BEGIN  Error:=0;    Readings[I]:='Reading'     END;
  IF (Readingstr='Reading') OR (Readingstr='R') OR (Readingstr='r') THEN BEGIN
    Error:=0;    Readings[I]:='Reading';   END;
  IF (Readingstr='Yield') OR (Readingstr='Y') OR (Readingstr='y') THEN   BEGIN
    Error:=0;    Readings[I]:='Yield';     END;
  IF (Readingstr='Ultimate') OR (Readingstr='U') OR (Readingstr='u')THEN BEGIN
    Duplicate:=False;
    FOR Q:=1 to DataPoints DO
    BEGIN
    IF Readings[Q]='Ultimate' THEN  Duplicate:=True;
    END;
    IF Duplicate=True  THEN       BEGIN
    GotoXY(1,25); DelLine;
    WriteAt(-1,25,Highlite,NoCR,'An Ultimate Point has already been chosen');
    Delay(1500); DelLine;         END
    ELSE BEGIN
    Readings[I]:='Ultimate';  Error:=0;     END;
  END;
  IF (Readingstr='Fracture') OR (Readingstr='F') OR (Readingstr='f')THEN BEGIN
    Duplicate:=False;
    FOR Q:=1 to DataPoints DO
    BEGIN
    IF Readings[Q]='Fracture' THEN  Duplicate:=True;
    END;
    IF Duplicate=True  THEN       BEGIN
    GotoXY(1,25); DelLine;
    WriteAt(-1,25,Highlite,NoCR,'A Fracture Point has already been chosen');
    Delay(1500); DelLine;         END
    ELSE BEGIN
    Readings[I]:='Fracture';  Error:=0;     END;
  END;
  IF (Readingstr='Initial') OR (Readingstr='I') OR (Readingstr='i')THEN BEGIN
    Duplicate:=False;
    FOR Q:=1 to DataPoints DO
    BEGIN
    IF Readings[Q]='Initial' THEN  Duplicate:=True;
    END;
    IF Duplicate=True  THEN       BEGIN
    GotoXY(1,25); DelLine;
    WriteAt(-1,25,Highlite,NoCR,'An Initial Point has already been chosen');
    Delay(1500); DelLine;         END
    ELSE BEGIN
    Readings[I]:='Initial';  Error:=0;     END;
  END;
  UNTIL Error=0;
 ND;

PROCEDURE Read_Loads;
VAR
```

```
  Loadstr :string[80];
  Error:integer;
BEGIN
  REPEAT
  Readln(Loadstr);
  IF Length(Loadstr)=0 THEN     BEGIN
  Loads[I]:=Undefined;
  WriteAt(-1,25,Highlite,NoCR,'No Data Available for this point Assumed');
  Delay(1500);    DelLine;
  Error:=0;                     END;
  Val(Loadstr,Loads[i],Error);
  IF loads[I]<0 THEN Error:=1;
  IF Error <>0 THEN    BEGIN
    WriteAt(-1,25,Highlite,NoCR,'Invalid data.');
    Delay(1000);   DelLine;
    GotoXY(24,13); Clreol;       END;
  UNTIL Error=0;
END;

PROCEDURE Read_Lengths;
VAR
  Lengthstr :string[80];
  Error:integer;
BEGIN
  REPEAT
  Readln(lengthstr);
  IF Length(Lengthstr)=0 THEN BEGIN
  Lengths[I]:=Undefined;
  WriteAt(-1,25,Highlite,NoCR,'No Data Available for this point Assumed');
  Delay(1500);   DelLine;
  Error:=0;        END;
  Val(Lengthstr,Lengths[i],Error);
  IF Lengths[I]<=0 THEN   Error:=1;
  IF Error <>0 THEN    BEGIN
    WriteAt(-1,25,Highlite,NoCR,'Invalid data.');
    Delay(1000);   DelLine;
    GotoXY(42,13); Clreol;       END;
  UNTIL Error=0;
END;

PROCEDURE Read_Diameters;
VAR
  Diameterstr :string[80];
  Error:integer;
BEGIN
  REPEAT
  Readln(Diameterstr);
  IF Length(Diameterstr)=0 THEN    BEGIN
  Diameters[I]:=Undefined;
  WriteAt(-1,25,Highlite,NoCR,'No Data Available for this point Assumed');
  Delay(1500);   DelLine;
  Error:=0;                        END;
  Val(Diameterstr,Diameters[i],Error);
  IF Diameters[I]<=0   THEN  Error:=1;
  IF Error <>0 THEN    BEGIN
    WriteAt(-1,25,Highlite,NoCR,'Invalid data.');
    Delay(1000);   DelLine;
    GotoXY(60,13); Clreol;       END;
  UNTIL Error=0;
END;

PROCEDURE Input_Data_Format_per_Reading;
BEGIN
  Clrscr;
  TextColor(Title);
  WriteAt(24,-1,Normal,CR,'Load, kN          Length, mm        Diameter, mm');
```

```
  GotoXY(6,13);  TextColor(Title);
  Write(I,'. '); TextColor(Regular); Write(Readings[I]);
  GotoXY(24,13);Read_Loads;
  GotoXY(42,13);Read_Lengths;
  GotoXY(60,13);Read_Diameters;
END;

PROCEDURE Input_Data;
BEGIN
  FOR I:=1 to DataPoints DO
  BEGIN
    Select_Reading_Type;
    Input_Data_Format_per_Reading;
  END;
END;

PROCEDURE Inputing_Data;
BEGIN
  Clrscr;
  WriteAt(-1,-1,Normal,NoCR,'Name of Input File:');
  ReadLn(Filename);
  Assign(Indata,Filename);
  {$I-} Reset(Indata); {$I+}
  IF IOResult<>0 THEN
      WriteAt(1,25,Highlite,NoCR,'This File cannot be opened.')
  ELSE
    BEGIN
      I:=0;
      IF I<20 THEN BEGIN
      WHILE NOT EOF(Indata) DO
      BEGIN
      I:=I+1;
      Readln(Indata,Readings[I]);
      Readln(Indata,Loads[I]);
      Readln(Indata,Lengths[I]);
      Readln(Indata,Diameters[I]);
      DataPoints:=I;
      END;          END;
      Close(Indata);
    END;
END;

PROCEDURE Input_Option;
VAR
  Y,Q  : Integer;
  Stop : Boolean;
BEGIN
  Stop:=False;
  REPEAT
  Clrscr;   TextColor(Title);
  Writeln('**INPUT OPTION**');    TextColor(Regular);
  Writeln('  Choose an option');  TextColor(Title);
  Write('  1');TextColor(Regular);Writeln(' Input Data Directly');TextColor(14);
  Write('  2');TextColor(Regular);Writeln(' Load Input Data File');TextColor(14);
  Write('  3');TextColor(Regular);Writeln(' Load Example Data');
  Writeln;
  Write('Enter selection here:');
  Read_Code(Y,Q);
  CASE Y OF
    1: BEGIN
       Stop:=true;  Window(1,1,80,25);  Clrscr;
       Select_Number_Data_Points;  Input_Data;
       END;
    2: BEGIN
       Stop:=true;  Window(1,1,80,25); Inputing_Data;
       END;
```

```
    3: BEGIN
       Stop:=true; Window(1,1,80,25); Clrscr; Example_Data; Print_Data;
       proceed;
       END;
  ELSE                          BEGIN
  Writeln;
  WriteAt(-1,25,Highlite,NoCR,'Code outside of range 1-3');
  Delay(1500);    DelLine;    END;
  END;
  UNTIL Stop=true;
END;


{Edit Data}

PROCEDURE Edit_Individual_Readings;
VAR
  Q,Code,A,PosY:integer;
  Answer:string[80];
BEGIN
  REPEAT
  Print_Data;
  PosY:=WhereY+1;
  Writeln;
  WriteAt(-1,PosY,Normal,NoCR,'Select variable to be modified, '+
                              'by entering corresponding number:');
  ClrEOL;
  Read_code(I,Q);
  IF I<1 THEN A:=2 ELSE A:=1;
  IF (I>DataPoints)  OR (Q=45) THEN A:=3 ELSE A:=A;
  CASE A OF
    2:  BEGIN
        WriteAt(-1,25,Highlite,NoCR,'Code out of range ');
        Delay(1500);   DelLine;
        END;
    3:  BEGIN
        WriteAt(-1,25,Highlite,NoCR,'Code out of range ');
        Delay(1500);   DelLine;
        END;
  ELSE
    Select_Reading_Type;
    Input_Data_Format_per_Reading;
    REPEAT
    Print_Data;
    Code:=4;
    WriteAt(-1,25,Normal,NoCR,'Do you wish to edit another variable? (no):');
    ClrEOL;
    Yes_No_Answer(code);
    CASE Code OF
      1:  Code:=3;
      2:  BEGIN
          Code:=3; Q:=45;
          END;
    ELSE                          BEGIN
      Writeln;
      WriteAt(-1,25,Highlite,NoCR,'Must answer yes or no and hit return:');
      Delay(1500);   DelLine;     END;
    END;
    UNTIL Code=3;
    END;
  UNTIL Q=45;
 ND;

 ROCEDURE Add_Readings;
VAR
  Q,Code,A,PosY,J:integer;
```

```
  TempReadings : Array[1..20] of String[80];
  TempLoads,TempLengths,TempDiameters: Array[1..20] of Real;
BEGIN
  Q:=4;
  REPEAT
  Print_Data;
  PosY:=WhereY+1;
  Writeln;
  WriteAt(-1,PosY,Normal,NoCR,'Select Reading to be added, '+
                              'by entering corresponding number:');
  ClrEOL;
  Read_code(J,Q);
  IF (J<=DataPoints) AND (J>0) THEN  BEGIN
    FOR I:=J to DataPoints DO
    BEGIN
    TempReadings[I]:=Readings[I];
    TempLoads[I]:=Loads[I];
    TempLengths[I]:=Lengths[I];
    TempDiameters[I]:=Diameters[I];
    END;
    FOR I:=J to DataPoints DO
    BEGIN
    Readings[I+1]:=TempReadings[I];
    Loads[I+1]:=TempLoads[I];
    Lengths[I+1]:=TempLengths[I];
    Diameters[I+1]:=TempDiameters[I];
    END;
    DataPoints:=DataPoints+1;        END;
  IF J>DataPoints THEN DataPoints:=J;
  IF (J<1)  OR  (J>20)  THEN                  BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Negative Numbers or Number greater than 20'+
                              ' not accepted ');
  Delay(1500);   DelLine;     END
  ELSE           BEGIN
  I:=J;
  Select_Reading_Type;
  Input_Data_Format_per_Reading;
  REPEAT
  Print_Data;
  Code:=4;
  WriteAt(-1,25,Normal,NoCR,'Do you wish to add another variable? (no):');
  ClrEOL;
  Yes_No_Answer(Code);
  CASE Code OF
    1:  Code:=3;
    2:  BEGIN
        Code:=3; Q:=45;
        END;
  ELSE                           BEGIN
    Writeln;
    WriteAt(-1,25,Highlite,NoCR,'Must answer yes or no and hit return:');
    Delay(1500);   DelLine;     END;
  END;
  UNTIL Code=3;   END;
  UNTIL Q=45;
END;

PROCEDURE Delete_Readings;
VAR
  Q,Code,PosY,J:integer;
BEGIN
  Q:=4;
  REPEAT
  Print_Data;
  PosY:=WhereY+1;
  Writeln;
```

```
  WriteAt(-1,PosY,Normal,NoCR,'Select Reading to be deleted, '+
                              'by entering corresponding number:');
  ClrEOL;
  Read_code(J,Q);
  IF (J<=DataPoints) AND (J>=1) THEN  BEGIN
    DataPoints:=DataPoints-1;
    FOR I:=J to DataPoints DO
    BEGIN
    Readings[I]:=Readings[I+1];
    Loads[I]:=Loads[I+1];
    Lengths[I]:=Lengths[I+1];
    Diameters[I]:=Diameters[I+1];
    END;
    REPEAT
    Print_Data;
    Code:=4;
    WriteAt(-1,25,Normal,NoCR,'Do you wish to delete another variable? (no):');
    ClrEOL;
    Yes_No_Answer(Code);
    CASE Code OF
      1:  Code:=3;
      2:  BEGIN
          Code:=3; Q:=45;
          END;
    ELSE                              BEGIN
      Writeln;
      WriteAt(-1,25,Highlite,NoCR,'Must answer yes or no and hit return:');
      Delay(1500);   DelLine;     END;
    END;
    UNTIL Code=3;           END
  ELSE                              BEGIN
    WriteAt(-1,25,Highlite,NoCR,'Reading not defined ');
    Delay(1500);   DelLine;       END;
  UNTIL Q=45;
END;

PROCEDURE Menu_Edit_Data;
VAR
  J,Q : Integer;
BEGIN
  REPEAT
  Clrscr;
  Window(29,10,80,25);
  Clrscr;    TextColor(Title);
  Writeln('Edit Data Menu');
  Writeln;    TextColor(14);
  Write('1'); TextColor(Regular); Writeln(' Edit Individual Reading');
  TextColor(14); Write('2'); TextColor(Regular); Writeln(' Add Reading');
  TextColor(14); Write('3'); TextColor(Regular);
  Writeln(' Delete Reading'); TextColor(14);
  Write('4'); TextColor(Regular); Writeln(' Return to Main Menu');
  Writeln;
  Write('Enter selection here:');
  Read_Code(J,Q);
  CASE J OF
    1:  BEGIN
        Clrscr;  Edit_Individual_Readings;
        END;
    2:  BEGIN
        Clrscr;  Add_Readings;
        END;
    3:  BEGIN
        Clrscr;  Delete_Readings;
        END;
    4:  Writeln;
  ELSE                        BEGIN
```

```
    Writeln;
    WriteAt(-1,25,Highlite,NoCR,'Code outside of range 1-4');
    Delay(1500);   DelLine;  END;
  END;
  UNTIL J=4;
END;


{Saving Data in a File}

PROCEDURE Saving_CurrentData;
BEGIN
  WriteAt(-1,-1,Normal,NoCR,'Name Input File to be saved:');
  Readln(Filename);
  Assign(Indata,Filename);
  Rewrite(Indata);
  For I:=1 to DataPoints DO
  BEGIN
  Writeln(Indata,Readings[I]);
  Writeln(Indata,Loads[I]);
  Writeln(Indata,Lengths[I]);
  Writeln(Indata,Diameters[I]);
  END;
  Close(Indata);
END;


{Calculations}

FUNCTION Power(mantissa,Exponent :Real): Real;
BEGIN
  Power:=Exp(Ln(Mantissa)*Exponent);
END;

PROCEDURE Determine_Initial_Conditions;
VAR
  Initial : Integer;
BEGIN
  Initial:=0;
  FOR I:=1 to DataPoints DO
  BEGIN
  IF Readings[I]='Initial' THEN Initial:=I;
  END;
  IF Initial=0 THEN  BEGIN
  FOR I:=1 to DataPoints DO
  BEGIN
  IF Loads[I]=0  THEN Initial:=I;
  END;                END;
  IF Initial=0  THEN  Initial:=1;
  L0:=Lengths[Initial];
  D0:=Diameters[Initial];
  A0:=SQR(D0)*PI/4;
END;

PROCEDURE Print_Initial_Conditions;
BEGIN
  Determine_Initial_Conditions;
  Clrscr;
  TextColor(Title);
  WriteAt(-1,11,Normal,NoCR,'The Initial Dimensions are:');TextColor(14);
  WriteAt(-1,12,Normal,NoCR,'Length mm      Diameter mm     Area sqmm');
  GotoXY(19,13);
  IF L0=Undefined THEN Write('Not defined')  ELSE Write(L0:10:4);
  GotoXY(35,13);
  IF D0=Undefined THEN Write('Not defined')  ELSE Write(D0:10:4);
  GotoXY(50,13);
```

```
  IF D0=Undefined THEN Write('Not defined')  ELSE Writeln(A0:10:4);
  Proceed;
END;

PROCEDURE Determine_Stress_Strain;
BEGIN
  Determine_Initial_Conditions;
  IF  L0<>Undefined   THEN  BEGIN
  FOR I:=1 To DataPoints DO
  BEGIN
  IF Lengths[I]<>Undefined  THEN                  BEGIN
  EngStrain[I]:=(Lengths[I]-L0)/L0;
  LenTrStrain[I]:=Ln(lengths[I]/L0);              END
  ELSE                        BEGIN
  EngStrain[I]:=0.1E-50;
  LenTrStrain[I]:=0.1E-50;   END;
  END;                      END;
  IF D0<>Undefined     THEN  BEGIN
  FOR I:=1 to DataPoints  DO
  BEGIN
  IF Loads[I]<>Undefined  THEN
  EngStress[I]:=Loads[I]*1000/A0
  ELSE  EngStress[I]:=Undefined;
  IF Diameters[I]<>Undefined THEN
  DiaTrStrain[I]:=2*Ln(D0/Diameters[I])
  ELSE  DiaTrStrain[I]:=Undefined;
  END;                        END;
  FOR I:=1 to DataPoints DO
  BEGIN
  IF (Diameters[I]<>Undefined) AND (Loads[I]<>Undefined) THEN
  TrueStress[I]:=Loads[I]*1000/(SQR(Diameters[I])*PI/4)
  ELSE  TrueStress[I]:=Undefined;
  END;
END;

PROCEDURE Print_Stress_Strain;
VAR
  PosY: Integer;
BEGIN
  Determine_Stress_Strain;
  Clrscr;
  PosY:=(22-DataPoints) DIV 2;  TextColor(Title);
  WriteAt(3,PosY,Normal,CR,'Reading      Eng Stress    Eng Strain   '+
          'True Stress         True Strain');  TextColor(Title);
  WriteAt(20,PosY+1,Normal,CR,'MPa');           TextColor(Title);
  WriteAt(46,PosY+1,Normal,CR,'MPa');           TextColor(Title);
  WriteAt(55,PosY+1,Normal,CR,'ln(Li/Lo)');      TextColor(Title);
  WriteAt(68,PosY+1,Normal,CR,'2ln(Do/Di)');
  IF  L0<>Undefined   THEN  BEGIN
  FOR I:=1 To DataPoints DO
  BEGIN
  GotoXY(3,PosY+1+I);
  TextColor(Title);Write(I,'. ');TextColor(Regular);Write(Readings[I]:8);
  IF Lengths[I]<>Undefined  THEN                    BEGIN
  GotoXY(29,PosY+1+I);   Write(EngStrain[I]:10:4);
  GotoXY(54,PosY+1+I);   Write(LenTrStrain[I]:10:4); END;
  END;                           END
  ELSE                             BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Initial Length Not defined. '+
          'Eng. Strain & True Strain dep on length Can not be Calculated');
  Delay(2000);       DelLine;      END;
  IF D0<>Undefined     THEN  BEGIN
  FOR I:=1 to DataPoints  DO
  BEGIN
  IF Loads[I]<>Undefined  THEN                      BEGIN
  GotoXY(16,PosY+1+I);   Write(EngStress[I]:10:4);    END;
```

```
  IF Diameters[I]<>Undefined THEN              BEGIN
  GotoXY(68,PosY+1+I);  Write(DiaTrStrain[I]:10:4); END;
  END;                        END
  ELSE                           BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Initial Diameter Not defined. '+
         'Eng. Stress & True  Strain dep on dia Can not be Calculated');
  Delay(2000);       DelLine;    END;
  FOR I:=1 to DataPoints DO
  BEGIN
  IF (Diameters[I]<>Undefined) AND (Loads[I]<>Undefined) THEN  BEGIN
  GotoXY(42,PosY+1+I);  Write(TrueStress[I]:10:4);              END;
  END;
  Proceed;
END;

PROCEDURE Selecting_Reading(VAR Selected_Reading:Integer;
                                Text            :String80);
{Selection of Reading for Calculation of K & N or E}
VAR
  PosY,Q : Integer;
BEGIN
  REPEAT
  Print_Data;
  PosY:=WhereY+1;
  WriteAt(-1,PosY,Normal,NoCR,Text);
  ClrEOL;
  Read_Code(Selected_Reading,Q);
  IF (Selected_Reading>=1) AND (Selected_Reading<=DataPoints) THEN Q:=10
  ELSE                              BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Reading not defined');
  Delay(1500);      DelLine;        END;
  UNTIL Q=10;
END;


PROCEDURE Are_True_StressStrain_Defined(VAR Reading:Integer);
{Checking Selected Reading to see if appropriate data point for such calculation}
BEGIN
  Code:=True;
  IF (TrueStress[Reading]=0.1E-50) OR (TrueStress[Reading]=0)  THEN  BEGIN
  WriteAt(-1,25,Highlite,NoCR,'True Stress for this point must be defined');
  Delay(1500);    DelLine;  Code:=False; END;
  IF (LenTrStrain[Reading]=0.1E-50) OR (LenTrStrain[Reading]=0)
  AND (DiaTrStrain[Reading]=0.1E-50) OR (DiaTrStrain[Reading]=0)
  THEN BEGIN
  WriteAt(-1,25,Highlite,NoCR,'True Strain for this point must be defined');
  Delay(1500);    DelLine;  Code:=False;                    END;
END;

PROCEDURE NoPrint_True_StressStrain_Defined(VAR Reading:Integer);
{Checking Selected Reading to see if appropriate data point for such calculation}
BEGIN
  Code:=True;
  IF (TrueStress[Reading]=0.1E-50) OR (TrueStress[Reading]=0)  THEN
  Code:=False;
  IF (LenTrStrain[Reading]=0.1E-50) OR (LenTrStrain[Reading]=0)
  AND (DiaTrStrain[Reading]=0.1E-50) OR (DiaTrStrain[Reading]=0)
  THEN
  Code:=False;
END;

PROCEDURE Calculate_K_and_n;
{Asks User to select PROPER readings used in the calculations of k & n and
 performs such calculations.}
VAR
  LNStrains : Real;
```

```
BEGIN
  REPEAT
  Selecting_Reading(First,'Select First Reading for Calculation of K & n:');
  Are_True_StressStrain_Defined(First);
  UNTIL Code=True;
  REPEAT
  Selecting_Reading(Second,'Select Second Reading for Calculation of K & n:');
  Are_True_StressStrain_Defined(Second);
  IF Second=First  THEN                    BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Second reading must differ from the first');
  Delay(1500);    DelLine;  Code:=False    END;
  UNTIL Code=True;
  Clrscr;  TextColor(Title);
  GotoXY(32,12);  Write('n');  GotoXY(50,12);   Write('K'); TextColor(Regular);
  IF (DiaTrStrain[Second]<>0.1E-50) OR (DiaTrStrain[Second]<>0)  THEN  BEGIN
  IF (DiaTrStrain[First]=0.1E-50) OR (DiaTrStrain[First]=0)
  THEN  LNStrains:=LN(LenTrStrain[First]/DiaTrStrain[Second])
  ELSE  LNStrains:=LN(DiaTrStrain[First]/DiaTrStrain[Second]);          END
  ELSE                                                      BEGIN
  IF (DiaTrStrain[First]=0.1E-50) OR (DiaTrStrain[First]=0)
  THEN  LNStrains:=LN(LenTrStrain[First]/LenTrStrain[Second])
  ELSE  LNStrains:=LN(DiaTrStrain[First]/LenTrStrain[Second]); END;
  n:=LN(TrueStress[First]/TrueStress[Second])/LNStrains;
  GotoXY(25,13);    Write(n:10:4);
  IF (DiaTrStrain[First]<>0.1E-50) AND (DiaTrStrain[First]>0)  THEN
  BEGIN
  K:=TrueStress[First]/Power(DiaTrStrain[First],n);
  GotoXY(45,13);     Write(K:10:4);
  END
  ELSE                                    BEGIN
  IF (LenTrStrain[First]<>0.1E-50) AND (LenTrStrain[First]>0)  THEN
  BEGIN
  K:=TrueStress[First]/Power(LenTrStrain[First],n);
  GotoXY(45,13);     Write(K:10:4);
  END
  ELSE                          BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Calculating K with current reading selection'+
                              ' not possible');
  Delay(1500);    DelLine;    END;
                                          END;
  Proceed;
END;

PROCEDURE Call_K_and_n;
{Determines if Proper Data Points are available for calculation of K & N
 before calling Calculate_K_and_n procedure.}
VAR
  J:Integer;
BEGIN
  J:=0;
  Determine_Stress_Strain;
  Clrscr;
  FOR I:=1 to DataPoints DO
  BEGIN
  Code:=True;
  NoPrint_True_StressStrain_Defined(I);
  IF Code=True THEN J:=J+1;
  END;
  IF (DataPoints>=3) AND (J>=2) THEN Calculate_K_and_n
  ELSE              BEGIN
  WriteAt(-1,25,Highlite,NoCR,'No proper data to calculate k & n');
  Delay(1500);         DelLine;
  n:=0;    K:=0;    END;
END;

PROCEDURE Compute_Ultimate_StressStrain;
```

```
VAR
  J: Integer;
BEGIN
  J:=0;
  UltTrStrainRd:=0.1E-50;
  UltEngStressRd:=0.1E-50;
  UltTrStressRd:=0.1E-50;
  FOR I:=1 To DataPoints DO
  BEGIN
  IF Readings[I]='Ultimate'  THEN    BEGIN
  UltEngStressRd:=EngStress[I];
  UltTrStressRd:=TrueStress[I];
  IF DiaTrStrain[I]<>0.1E-50  THEN  UltTrStrainRd:=DiaTrStrain[I]
  ELSE UltTrStrainRd:=LenTrStrain[I];  END;
  END;
  IF (K<>0) OR (n<>0)  THEN   BEGIN
  UltEngStressCal:=K*Power(n/2.718282,n);
  UltTrStressCal:=K*Power(n,n);
  UltTrStrainCal:=n;            END
  ELSE                         BEGIN
  UltEngStressCal:=0;
  UltTrStrainCal:=0;
  UltTrStressCal:=0;            END;
END;

PROCEDURE Print_Ultimate_StressStrain;
BEGIN
  Compute_Ultimate_StressStrain;
  Clrscr;
  Window(1,1,80,25);
  Clrscr;  TextColor(Title);
  WriteAt(-1,10,Normal,NoCR,'Stresses & Strains At Ultimate');
  TextColor(Title);
  WriteAt(1,-1,Normal,NoCR,'Calculated from K & n:                    '+
                      'Determined from Reading:'); TextColor(Title);
  WriteAt(1,13,Normal,NoCR,'Eng Stress   True Stress  True Strain      '+
          'Eng Stress  True Stress  True Strain');  TextColor(Regular);
  GotoXY(1,14);   IF UltEngStressCal<>0      THEN Write(UltEngStressCal:10:4);
  GotoXY(14,14);  IF UltTrStressCal<>0       THEN Write(UltTrStressCal:10:4);
  GotoXY(27,14);  IF UltTrStrainCal<>0       THEN Write(UltTrStrainCal:10:4);
  GotoXY(44,14);  IF UltEngStressRd<>0.1E-50 THEN Write(UltEngStressRd:10:4);
  GotoXY(56,14);  IF UltTrStressRd<>0.1E-50  THEN Write(UltTrStressRd:10:4);
  GotoXY(69,14);  IF UltTrStrainRd<>0.1E-50  THEN Write(UltTrStrainRd:10:4);
  Proceed;
END;

PROCEDURE Calculate_Maximum_Load;
BEGIN
  MaxLoadCal:=0;   MaxLoadRd:=0;
  MaxloadCal:=UltEngStressCal*A0;
  IF UltEngStressRd<>0.1E-50 THEN  MaxloadRd:=UltEngStressRd*A0
  ELSE MaxLoadRd:=0;
END;

PROCEDURE Print_Maximum_Load;
BEGIN
  Calculate_Maximum_Load;
  Clrscr;     TextColor(Title);
  WriteAt(-1,-1,Normal,NoCR,'Maximum Load, kN:'); TextColor(Title);
  WriteAt(12,13,Normal,NoCR,'Calculated from Su estimate   '+
                        'Calculated from Su reading');
  GotoXY(20,14);   IF MaxLoadCal<>0 THEN Write(MaxLoadCal:10:4);
  GotoXY(50,14);   IF MaxLoadRd<>0  THEN Write(MaxLoadRd:10:4);
  Proceed;
END;
```

```
PROCEDURE Max_Nominal_Strain;
BEGIN
  MaxEngStrain:=EngStrain[1];
  FOR  I:=2 To DataPoints DO
  BEGIN
  IF EngStrain[I]>EngStrain[I-1]   THEN   MaxEngStrain:=EngStrain[I];
  END;
  Clrscr;  TextColor(Title);
  WriteAt(-1,-1,Normal,NoCR,'The maximum Engineering Strain is');
  GotoXY(35,13);     Write(MaxEngStrain:10:4);
  Proceed;
END;

PROCEDURE Modulus_of_Elasticity;
{Asks User to select PROPER readings used in the calculations and performs
 such calculations}
VAR
  Reading :Integer;
BEGIN
  REPEAT
  Reading:=0;
  Selecting_Reading(Reading,'Select Reading for Calculation of E:');
  Code:=True;
  IF (EngStress[Reading]=0) OR (EngStress[Reading]=Undefined)  THEN BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Eng Strain for this point must be defined'+
                                  ' and nonzero');
  Delay(1500);  DelLine; Code:=False;    END;
  UNTIL Code=True;
  E:=EngStress[Reading]/EngStrain[Reading];
  Clrscr;        TextColor(Title);
  WriteAt(-1,-1,NOrmal,NoCR,'The Modulus of Elasticity is');
  GoToXY(35,13);    Write(E:10:4);
  Proceed;
END;

PROCEDURE Call_Modulus_Of_Elasticity;
{Determines if Proper Data Points are available for calculation of E
 before calling Modulus_Of_Elasticity procedure.}
VAR
  J:Integer;
BEGIN
  Determine_Stress_Strain;
  J:=0;
  FOR I:=1 to DataPoints DO
  BEGIN
  Code:=True;
  IF (EngStrain[I]=0) OR (EngStrain[I]=Undefined)  THEN Code:=False;
  IF Code=True THEN J:=J+1;
  END;
  IF J>=1 THEN Modulus_Of_Elasticity
  ELSE                          BEGIN
  WriteAt(-1,25,Highlite,NoCR,'No proper data to calculate E');
  Delay(1500);        DelLine;    END;
END;

PROCEDURE Menu_Computations;
VAR
  J,Q : Integer;
BEGIN
  REPEAT
  Clrscr;
  Window(29,10,80,25);
  Clrscr;    TextColor(Title);
  Writeln('Computations Menu');
  Writeln;    TextColor(14);
  Write('1'); TextColor(Regular); Writeln(' Initial Conditions');
```

```
  TextColor(14); Write('2'); TextColor(Regular);
  Writeln(' Stresses and Strains');
  TextColor(14); Write('3'); TextColor(Regular);
  Writeln(' K and n'); TextColor(14);
  Write('4'); TextColor(Regular);
  Writeln(' Stress and Strain at Ultimate'); TextColor(Title);
  Write('5'); TextColor(Regular); Writeln(' Maximum Load');
  TextColor(Title); Write('6'); TextColor(Regular);
  Writeln(' Maximum Nominal Strain');  TextColor(Title); Write('7');
  TextColor(Regular); Writeln(' Calculate the Modulus of Elasticity');
  TextColor(Title); Write('8'); TextColor(Regular);
  Writeln(' All of the Above');
  TextColor(Title); Write('9'); TextColor(Regular);
  Writeln(' Return to Main Menu');
  Writeln;
  Write('Enter selection here:');
  Read_Code(J,Q);
  CASE J OF
    1:  BEGIN
        Clrscr; Window(1,1,80,25);  Print_Initial_Conditions;
        END;
    2:  BEGIN
        Clrscr;  Window(1,1,80,25);  Print_Stress_Strain;
        END;
    3:  BEGIN
        Clrscr;  Window(1,1,80,25);  Call_K_and_n;
        END;
    4:  BEGIN
        Clrscr;  Window(1,1,80,25);
        WriteAt(-1,-1,Normal,NOCR,'K and n'+
        ' needed to calculate stress/strain at ultimate'); Delay(2000); DelLine;
        Call_K_and_n;
        Print_ultimate_StressStrain;
        END;
    5:  BEGIN
        Clrscr; Window(1,1,80,25);  WriteAt(-1,-1,Normal,NOCR,'K and n'+
        ' needed to calculate the maximum load'); Delay(2000); DelLine;
        Call_K_and_n;  Compute_ultimate_StressStrain; Print_maximum_Load;
        END;
    6:  BEGIN
        Clrscr; Window(1,1,80,25);  Determine_Stress_Strain; Max_Nominal_Strain;
        END;
    7:  BEGIN
        Clrscr; Window(1,1,80,25); Call_Modulus_of_Elasticity;
        END;
    8:  BEGIN
        Clrscr; Window(1,1,80,25);
        Print_Initial_Conditions; Print_Stress_Strain; Call_K_and_n;
        Print_Ultimate_StressStrain; Print_maximum_load;
        Max_Nominal_Strain;  Call_Modulus_of_Elasticity;
        END;
    9:  Writeln;
  ELSE                        BEGIN
    Writeln;
    WriteAt(-1,25,Highlite,NoCR,'Code outside of range 1-9');
    Delay(1500);   DelLine;  END;
  END;
  UNTIL J=9;
END;


{Plotting}

PROCEDURE Proceed_Graph;
BEGIN
  WriteAt(67,25,Highlite,NoCR,'CR to Proceed');
```

```
  Readln;
END;

PROCEDURE Title_Lables;
VAR
  Grid : String[80];
  Finish : Boolean;
BEGIN
  Write('Title for graph:');
  Readln(Titlename);
  IF Length(Titlename)=0 THEN Titlename:=Titlenames;
  Write('Lable for the x-axis(Strain):');
  Readln(Xlable);
  IF Length(Xlable)=0 THEN Xlable:='Strain';
  Write('Lable for the y-axis(Stress/MPa):');
  Readln(Ylable);
  IF Length(Ylable)=0 THEN Ylable:='Stress/MPa';
  Repeat
  Write('Grid scale(no)');
  Readln(Grid);
  IF Length(Grid)=0 THEN GridCode:=1;
  IF (Grid='no') or (Grid='n')  THEN GridCode:=1;
  IF (Grid='yes') OR (Grid='y') THEN GridCode:=2;
  Case GridCode of
  1:  Finish:=True;
  2:  Finish:=True;
  ELSE                    BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Must answer yes or no');
  Delay(1500); DelLine;    END;
  END;
  UNTIL Finish=True;
END;

PROCEDURE Print_TitleLables;
VAR
  XlableCent,YlableCent,b : Integer;
  YChar : String[1];
BEGIN
  TextColor(Title);
  WriteAt(-1,1,Normal,NoCR,Titlename);
  XlableCent:=3+((76-Length(Xlable)) DIV 2);
  WriteAt(XlableCent,25,Normal,NoCR,Xlable);
  YlableCent:=2+(20-Length(Ylable)) DIV 2;
  For B:=1 to Length(Ylable) DO
  BEGIN
  YChar:=Copy(Ylable,B,1);
  WriteAt(1,YlableCent+b,Normal,NoCR,YChar);
  END;
END;

PROCEDURE Axis;
BEGIN
Draw(42,11,42,181,1);
Draw(40,179,632,179,1);
END;

PROCEDURE Grid;
VAR
  M,L:Integer;
BEGIN
  L:=11;
  For M:=1 to  7  DO
  BEGIN
  Draw(40,L,632,L,1);
  L:=L+24;
  END;
```

```
  L:=101;
  For M:=1 to 10 DO
  BEGIN
  Draw(L,11,L,181,1);
  L:=L+59;
  END;
END;

PROCEDURE Divisions;
VAR
  M,L:Integer;
BEGIN
  L:=11;
  For M:=1 to  10  DO
  BEGIN
  Draw(40,L,44,L,1);
  L:=L+24;
  END;
  L:=101;
  For M:=1 to 10 DO
  BEGIN
  Draw(L,177,L,181,1);
  L:=L+59;
  END;
END;

PROCEDURE XY_MaxMin;
VAR
  M,L :Integer;
BEGIN
  Xmin:=Xarray[1]; Ymin:=Yarray[1];
  Xmax:=Xarray[DataPoints]; Ymax:=Yarray[DataPoints];
  FOR M:=1 to DataPoints DO
  BEGIN
  IF Xarray[M]<Xmin  THEN Xmin:=Xarray[M];
  IF Yarray[M]<Ymin  THEN Ymin:=Yarray[M];
  IF Xarray[M]>Xmax  THEN Xmax:=Xarray[M];
  IF Yarray[M]>Ymax  THEN Ymax:=Yarray[M];
  END;
END;

PROCEDURE ScalesXY;
VAR
  M,L:Integer;
  Xscalenumbers,Yscalenumbers:Array[1..11] of REAL;
  XIncrement,YIncrement:Real;
BEGIN
  IF GridCode=2 THEN Grid
  ELSE Divisions;
  XIncrement:=(Xmax-Xmin)/10;
  YIncrement:=(Ymax-Ymin)/7;
  Xscalenumbers[1]:=Xmin;
  For L:=2 to 11 DO
  Xscalenumbers[L]:=Xscalenumbers[L-1]+XIncrement;
  Yscalenumbers[1]:=Ymin;
  FOR L:=2 to 8 DO
  Yscalenumbers[L]:=Yscalenumbers[L-1]+YIncrement;
  GotoXY(5,24);
  Write(Xscalenumbers[1]:4:2);
  GotoXY(12,24);
  Write(Xscalenumbers[2]:4:2);
  GotoXY(19,24);
  Write(Xscalenumbers[3]:4:2);
  GotoXY(27,24);
  Write(Xscalenumbers[4]:4:2);
  GotoXY(34,24);
```

```
  Write(Xscalenumbers[5]:4:2);
  GotoXY(42,24);
  Write(Xscalenumbers[6]:4:2);
  GotoXY(49,24);
  Write(Xscalenumbers[7]:4:2);
  GotoXY(56,24);
  Write(Xscalenumbers[8]:4:2);
  GotoXY(63,24);
  Write(Xscalenumbers[9]:4:2);
  GotoXY(70,24);
  Write(Xscalenumbers[10]:4:2);
  GotoXY(76,24);
  Write(Xscalenumbers[11]:4:2);
  M:=2;
  For L:=1 to 8 DO
  BEGIN
  GotoXY(2,M);
  Write(Yscalenumbers[9-L]:4:0);
  M:=M+3;
  END;
END;

PROCEDURE XY_Conversion;
VAR
  M,L : Integer;
BEGIN
  FOR M:=1 to DataPoints DO
  BEGIN
  XConv[M]:=Trunc(42+(590/(Xmax-Xmin)*(Xarray[M]-Xmin)));
  YConv[M]:=Trunc(179-(168/(Ymax-Ymin)*(Yarray[M]-Ymin)));
  END;
END;

PROCEDURE Print_Exes;
VAR
  M,L : Integer;
BEGIN
  FOR M:=1 To DataPoints DO
  BEGIN
  Draw(XConv[M]-2,YConv[M]+2,XConv[M]+2,YConv[M]-2,1);
  Draw(XConv[M]-2,YConv[M]-2,XConv[M]+2,YConv[M]+2,1);
  END;
END;

PROCEDURE TrueRel_Appr;
VAR
  M,L : Integer;
  XapprIncr:Real;
  Xappr,Yappr : Array [1..101] of Real;
  XapprConv,YapprConv : Array [1..101] Of Integer;
BEGIN
  IF (K<>0) AND (n<>0)  THEN     BEGIN
  Xappr[1]:=Xmin;
  XapprIncr:=(Xmax-Xmin)/100;
  FOR M:=2 to 101 DO
  Xappr[M]:=Xappr[M-1]+XapprIncr;
  FOR M:=1 to 101 DO
  BEGIN
  IF Xappr[M]<>0 THEN  Yappr[M]:=K*Power(Xappr[M],n)  ELSE Yappr[M]:=0;
  XapprConv[M]:=Trunc(42+(590/(Xmax-Xmin)*(Xappr[M]-Xmin)));
  YapprConv[M]:=Trunc(179-(168/(Ymax-Ymin)*(Yappr[M]-Ymin)));
  END;
  FOR M:=2 to 101 DO
  BEGIN
  L:=M-1;
  IF XapprConv[L]>=42 THEN
```

```
  Draw(XapprConv[L],YapprConv[L],XapprConv[M],YapprConv[M],1);
  END;                              END
  ELSE
      WriteAt(53,22,Highlite,NoCR,'No approximation calculated');
END;

PROCEDURE Plot_ENGStressStrain;
BEGIN
  Clrscr;
  Determine_Stress_Strain;
  Titlenames:='Eng. Stress vs. Eng. Strain';
  Title_Lables;
  Hires;
  Print_TitleLables;
  FOR I:=1 to DataPoints DO
  BEGIN
  Xarray[I]:=EngStrain[I];
  Yarray[I]:=EngStress[I];
  END;
  Axis;
  XY_MaxMin;
  IF (Xmax<>Xmin) AND (Ymax<>Ymin) THEN BEGIN
  ScalesXY;
  XY_Conversion;
  Print_Exes;
  END ELSE
  WriteAt(-1,-1,Normal,NoCR,'No Proper Data Available');
  Proceed_Graph;
  TextMode;
END;

PROCEDURE Plot_TrueStressStrain;
BEGIN
  Clrscr;
  Call_K_and_n;
  ClrScr;
  Titlenames:='True Stress vs. True Strain / Data(x) & Appr(-)';
  Title_Lables;
  Hires;
  Print_TitleLables;
  FOR I:=1 to DataPoints DO
  BEGIN
  IF DiaTrStrain[I]<>Undefined THEN Xarray[I]:=DiaTrStrain[I];
  Yarray[I]:=TrueStress[I];
  END;
  Axis;
  XY_MaxMin;
  IF (Xmax<>Xmin) AND (Ymax<>Ymin)  THEN BEGIN
  ScalesXY;
  XY_Conversion;
  Print_Exes;
  TrueRel_Appr;
  END ELSE
  WriteAt(-1,-1,Normal,NoCr,'No Proper Data Available');
  Proceed_Graph;
  TextMode;
END;

PROCEDURE Plotting_Menu;
VAR
  Y,Q  : Integer;
  Stop : Boolean;
BEGIN
  Stop:=False;
  REPEAT
  Clrscr;   TextColor(Title);
```

```
  Writeln('**PLOTTING MENU**');    TextColor(Regular);
  Writeln('  Choose an option');  TextColor(Title);
  Write('  1');TextColor(Regular);Writeln(' Eng Stress vs Eng Strain');
  TextColor(14);Write('  2');TextColor(Regular);
  Writeln(' True Stress vs True Strain/Data & Appr.');
  Writeln;
  Write('Enter selection here:');
  Read_Code(Y,Q);
  CASE Y OF
    1: BEGIN
       Stop:=true;  Window(1,1,80,25);
       Plot_EngStressStrain;
       END;
    2: BEGIN
       Stop:=true;  Window(1,1,80,25);
       Plot_TrueStressStrain;
       END;
  ELSE                          BEGIN
  Writeln;
  WriteAt(-1,25,Highlite,NoCR,'Code outside of range 1-3');
  Delay(1500);    DelLine;    END;
  END;
  UNTIL Stop=true;
END;


{Exit}

PROCEDURE Exit;
VAR
    Code    : integer;
    Answer  :String[80];
BEGIN
  REPEAT
  WriteAt(-1,-1,Normal,NoCR,'Exiting program? (yes):');
  ClrEOL;
  Readln(Answer);
  IF Length(Answer)=0 THEN Code:=1;
  IF (Answer='yes') OR (Answer='y') THEN Code:=1;
  IF (Answer='no')  OR (Answer='n') THEN Code:=2;
  CASE Code OF
    1:  BEGIN
        Quit:=true;  Code:=3;
        END;
    2:  Code:=3;
  ELSE                              BEGIN
    WriteAt(-1,25,Highlite,NoCR,'Must answer yes or no and hit return:');
    Delay(1500);   DelLine;       END;
  END;
  UNTIL Code=3;
END;


{Main Menu}

BEGIN
  LowVideo;
  Clrscr;
  Header;
  Introduction;
  Quit:=False;
  Initialize_Data_Array;

  REPEAT              {start main menu}
  Clrscr;
  Window(29,10,80,25);
```

```
  Clrscr; TextColor(Title);
  Writeln('**MAIN MENU**'); TextColor(Regular);
  Writeln('  Choose an Option');
  Writeln;      TextColor(Title);
  Write('  1');TextColor(Regular);Writeln(' List Current Data');TextColor(14);
  Write('  2');TextColor(Regular);Writeln(' Input Data');TextColor(Title);
  Write('  3');TextColor(Regular);Writeln(' Edit Data'); TextColor(Title);
  Write('  4');TextColor(Regular);Writeln(' Save Current Data'); TextColor(14);
  Write('  5');TextColor(Regular);Writeln(' Computations');TextColor(Title);
  Write('  6');TextColor(Regular);Writeln(' Plotting');TextColor(Title);
  Write('  7');TextColor(Regular);Writeln(' Exit Program');
  Writeln;
  WriteAt(-1,21,Normal,NoCR,'Enter selection here:');
  ClrEOL;
  Z:=0;
  Read_Code(Z,A);
  Writeln;
  CASE Z OF
    0:  Writeln;
    1:  BEGIN
        Print_Data;  Proceed;
        END;
    2:  BEGIN
        Input_Option;
        END;
    3:  Menu_Edit_Data;
    4:  BEGIN
        Window(1,1,80,25);  Clrscr; Saving_CurrentData;
        END;
    5:  Menu_Computations;
    6:  Plotting_Menu;
    7:  BEGIN
        Window(1,1,80,25);  ClrScr;   Exit;
        END;
  ELSE                      BEGIN
    WriteAt(-1,25,HighLite,NoCR,'Code outside of range');
    Delay(1500); DelLine;   END;
  END;
  UNTIL Quit=true;
END.
```

PHASE DIAGRAMS

MATERIALS

Interactive Computer Graphics for Storing Phase Diagrams

By Joanne L. Murray — No longer working there ('88)
Center for Materials Research
National Bureau of Standards
and
Don J. Orser — (301) 975-3443 → Dr Pugh (301) 975-5960
Center for Applied Mathematics
National Bureau of Standards → (301) 975-2000

Progress is described on an interactive graphics computer program needed for the creation of an automated phase diagram data base. Automation requires the ability to input literature data in various units and formats and to edit and update a phase diagram without unnecessary reentry of data. A relational data structure makes possible the necessary explicit representation of phase stability data. Examples of binary diagrams are given. Plans for global management of a large data base are described.

Introduction

Nine of the binary phase diagrams presented in this issue have been plotted by an interactive graphics computer program now being developed at NBS. This program is part of a project whose goal is an automated phase stability data base. This article is not a technical description of the computer algorithms, but rather illustrates for the metallurgist the potential for interactive computer graphics in computing phase diagrams. When the computer procedures have become standardized, a more detailed technical account of them, as well as the computer system implementing them, will be provided.

In the first section of this article, the requirements for a truly interactive graphics program are discussed. The nine binary diagrams in this issue are then used to exemplify various features of the program in the second section. Although the program is in the first stages of development, these computer-composed phase diagrams already reveal several advantages over draftsman-drawn figures. Some features of phase diagrams do pose problems for computer graphics, and these are also discussed in the last section. Long-range plans for a fully automated phase diagram data base are described and the influence that a completely automated data base is likely to exert upon the phase diagram field is considered.

Interactive Graphics

Maintaining an up-to-date compilation of phase diagrams and phase stability data has in recent years become a serious problem.[1,2] Phase diagrams can be compiled from recent literature fairly easily, but keeping up with the evaluation and review of data is quite a different matter. After a certain number of supplements to a compilation, complete revision becomes necessary. Because the original data from several experiments often have been combined by the previous evaluator in some unknown way, it may prove necessary in many instances to reread and reevaluate the original papers. A considerable amount of the original work must thus be duplicated. On the other hand, in reevaluating data, it is possible to introduce new judgements (and thus changes) that were not present in earlier versions. An automated phase diagram data base is needed to allow one to store all the older experimental data along with some blend of all the available data in an up-to-date evaluated diagram and with a record of how the current blend was achieved. Thus, a way is needed to introduce new experimental data and to reevaluate and reoptimize the phase diagram.

There has been some work on digitizing phase diagrams, and phase diagrams are routinely plotted by computer (especially thermodynamically calculated

phase diagrams).[3-11] Nash and West[12] have addressed the problems posed by ternary diagrams in interactive computer graphics. We are now primarily interested in the type of interactive graphics needed for the creation of a phase diagram data base. The practical problem we address is that of the initial input of phase diagram data from the literature. Data appear in many formats – for example, in atomic and weight percent scales, in numerical tables or graphical form. The data are a mixture of experimental points, curves fitted to experimental data, and boundaries that are drawn to make the diagram thermodynamically complete. The difference between plotting a phase diagram by computer and using an interactive graphics program to compose a diagram from various sources of data is discussed below.

Recent progress in the automation of thermodynamic data bases is close in spirit to what we have in mind for phase diagram graphics. There are now several thermodynamic data banks in existence; one recent example is the conversion of the data of Kubaschewski and Alcock to a computer data base.[13] Alcock's data base is accessible to users all over the world and is intended to be used as easily and as often as the tables in his book. It is to be updated continuously by means of a file into which participating users can submit new data, which can then be evaluated and incorporated into the data base. We have in mind a similar treatment of phase diagram data.

Let us now define the problem of diagram creation: what functions do we want the computer graphics program to carry out? We divide our tasks into two distinct subtasks:

- Composition of the phase diagram graph
- Generation of the phase diagram data

By "composition", we mean all those processes that go into transforming numerical data and bibliographic messages into a printed graph. In the composition mode, the graphics program is being used as a draftsman's tool. The elements being manipulated are not alloy phases and phase transformations, but for example, axes, tick marks, labels, solid and dotted lines, and character sizes. The manipulations one performs on these elements are, for example, rotations of the lines, repositioning of the labels, and transformations of weight to atomic percent scales. The requirement we make of the composition function of the graphics is that one should be able to do in a straightforward and direct way by computer whatever a draftsman may do to a phase diagram.

By "generation", we mean those aspects of the program that, by an algorithm, mimic scientific common sense and thermodynamic knowledge. Thus, in the process of generation, we determine not how a line is to be graphed, but where it is allowed to lie. We include under the category of generation:

1. The computation of a phase diagram from thermodynamic data.
2. If both thermodynamic and experimental phase diagram data are available, optimal utilization of both sets of data together, by the method of Henig et al. or by a similar technique.[14-17]
3. The qualitative evaluation of phase diagram data by application of, for example, Gibbs' phase rule, van't Hoff's rule, and the 180° rule. A very basic requirement of any phase diagram published in a compilation must be that it not violate any of these simple topological rules.

We want to start with the phase stability data, of which the traditional phase diagram is merely one kind of stylized representation. This information may be thermodynamic data, experimental phase diagram data, or phase boundaries postulated by an evaluator. From these data, one generates the elements of the phase diagram – the positions of the eutectics, peritectics, congruent melts, and phase boundaries. By composing these elements, one can make a printed version of the phase diagram.

We can now see why computer graphics is different from the storage of a digitized graph. The only information explicitly stored in a digitized picture is an array of black/white spots. If we ask for the position of a line or vertex, this information is only available implicitly as a result of processing the stored image. The real objects of interest in a graphical representation of a phase diagram are the lines, vertices, labels, and data points, and these should be referred to explicitly. Only by judicious choice of explicitly represented objects can both composition and generation be carried out on the same data set.

It is our intention that the above process be carried out interactively. By interactive, we mean that one can calculate the data from a graphics terminal and subsequently modify the graphical elements such as lines and vertices during a single session at the terminal. Most graphics applications produce a graph as a result of the execution of a program, and there is no way for the user to intervene between the calculation of the quantities to be graphed and the graphing process itself. Suppose the user wants to shorten a line or move a vertex. There is no way to refer to a line, because its graphical representation has been generated directly by the execution of a procedure, and it is not represented by an explicit data element. Representing it as a data element allows its attributes to be modified and subsequently displayed by the application of a separate display procedure.

The program should thus be divided into three separately functioning parts: the data structure stored in the memory, the procedures that display it in graphical form, and the front-end procedure that interactively permits the user to modify the data structure in order to input new data and to give instructions to produce graphs.

To be able to refer directly to graphical entities such as lines and vertices, we organize the data structure as a "relational data base". The objects in this data base are all of the graphical entities that we might want to refer to by name (i.e., lines, intersections of lines, phase regions they separate, labels that lie within the regions). These objects can be characterized by defining various attributes for them. This is done by assigning a value to an attribute-object pair yielding an attribute-object-value triple. For example, a region of the graph (object) has associated with it a label (attribute) whose value is the character string "LIQUID". In turn, the object label has as attribute a position, whose value is a composition/temperature pair. The relational data base allows one to define and manipulate a diagram by allowing the user to define the triples, e.g., the values associated with an attribute-object pair.

For example, changing the composition of a eutectic means that two lines must have their end points

changed. Without a relational data base, one might, for example, have to first identify the two lines and then reenter two entire lists of points. This procedure is cumbersome and involves the manipulation of much extraneous data. If the vertex can be referred to directly, these other operations can be performed automatically by the computer. Hence, the user can enter and edit data in the most natural possible way.

Furthermore, the relational structure makes possible interactive evaluation of phase diagram data. If the data are organized not just into an arbitrary set of lines to be plotted, but into a set of regions, lines, and vertices, then the objects can be organized according to the underlying thermodynamics of the diagram. Two-phase and single-phase regions can be easily distinguished, and Gibbs' rule can be applied. Topological rules are easily enforced because the topology of the diagram originally determined the organization of the data structure. It is the relational data base that allows one to organize the data in a way specifically appropriate to phase diagrams.

Examples of Binary Diagrams

The problems we are now considering are not typical of those expected in future years, because we are only beginning to acquire a data base of phase diagrams. Our main task at present is not updating or optimizing phase diagrams, but simply entering a large number of diagrams to establish the basic data base. Most of the data to be entered are not in the ideal form of numerical coordinates of points on the phase boundary but are given as a combination of numerical and graphical data. Our starting material is a draftsman's drawing. Temperature and compositions of invariant points are often, but not always, explicitly indicated. To input a curve to the computer directly from a graph, we use a graphics tablet, which is a large magnetically sensitive sheet that can digitize a point indicated on its surface.

In using the tablet, the first thing one must do is to calibrate the table with respect to the scale of the curve to be input. One sets the temperature and composition scales by indicating the temperatures and compositions of the defining rectangle of one's choice. After the scales are set, curves are entered on the tablet. An individual point on a curve is often numerically specified and can be entered directly from the keyboard. The data points are then fit by a numerical spline.[18] If the fit needs to be checked, individual curves can be displayed with the data points superimposed on the fitted spline, or a table of the points may be printed. Minor readjustments, such as making the horizontals exactly horizontal or making lines meet exactly at minima, are now made. If there are readjustments, new lines are entered from the keyboard and old lines are deleted. Positions, angles, and texts of the labels are now entered. Finally, a graph is composed of lines and labels; coordinate limits are given, and we specify which curves are to be plotted. Symbols and line character (dotted, solid, etc.) must be specified. The resulting enumeration of data and plotting parameters comprise a graph that can be stored in a data file.

Two tricky features of the Ag-Al diagram (see tear-out in back of this issue) are the minimum representing the congruent point 25 at. % and the eutectic at 58 at. %. Minima become considerably easier to graph and to match to each other when the numerical data fitting route can distinguish between points through which the curve is constrained to run exactly and points that are treated as approximate. Matching of slopes at end points is needed to deal efficiently with minima.

The Ag-Al peritectic posed problems because part of the data was given in the form of an insert, and it is now difficult to handle input data that are given as an insert. Inevitably, there are occasions when there is disagreement between the draftsman's diagram and his insert. Assuming that the two sets of input data are consistent, it is now impossible (and would in any case be awkward) to combine data for a single curve input under two different calibrations of the graphics tablet. This problem will be solved in part by features that allow one to modify the list of coordinates that make up a curve. Only the judgement of the user can resolve actual discrepancies, however, because the composition stage assumes that the data have already been evaluated.

We can now produce graphs that contain only a small temperature/composition range and hence are effectively enlargements of portions of the full diagram. In the future, we will have a zoom feature that will allow enlargement of any part of the graph as an insert.

A diagram can contain very closely spaced regions that do not show on ordinary composition scales, but that will sometimes be of interest and for which data may be available. For example, the primary solid solubility may be very low. The most accurate and self-consistent way to represent these data is as a part of a diagram that can be enlarged.

The benefit of using the computer as a draftsman's tool can be seen on the Pb side of the Al-Pb diagram (see tear-out in back of this issue). The drawing from which this was taken violated the 180° rule at 98.55 at.% Pb. Because numerical data were included, the curves were spline fit to the numerical data, giving a clear change in curvature at 98.55 at. % in agreement with the 180° rule.

Finally, it is clear that a set of graphical elements can be combined in various ways to form diagrams suitable for different purposes. For example, labels or experimental data points can be included or deleted in different representations of the data. Almost every diagram presents an example of some data or detail that benefits by a separate representation.

Future Developments

We have so far discussed contributions to phase diagram graphics from three quite separate disciplines:

- Interactive computing
- Metallurgical thermodynamics
- Numerical curve fitting

As the phase diagram data base grows, an additional discipline will make more important contributions, namely, scientific data base management. So far, we have discussed a program that can only peruse data within the bounds of an individual phase diagram. When our phase diagram collection has grown to several hundred diagrams, we shall want to be able to use the computer to examine a large collection of diagrams from the point of view of a particular feature. This then becomes data base management on a global level.

When global management is added, the collection will gain new usefulness. For example, one now finds in Hansen several different melting temperatures for Ti in different Ti binaries. With global management, one will be able to require that all the diagrams be consistent with pure metal properties. Phase diagram collection will become more useful as a research tool; thermodynamic parameters can be automatically calculated and optimized self consistently; systems can be searched for a particular phase or property; tables can be constructed, of solid solubilities, for example. Relations between any phase diagram property and the positions of the constituent elements in the periodic table can be studied.

Phase diagrams are almost always studied in connection with mechanical and electrical properties, and metastable phases are important, as well as equilibrium phases. The flexibility of the relational data base is such that one need not be limited to collecting only phase diagram data; any alloy property could potentially be stored in the data base and consequently make use of the graphics capability.

We anticipate that phase diagram compilations will soon be updated immediately upon publication of new data. The new experimental data will be combined with older work and with the best optimized thermodynamic data to produce a reevaluated phase diagram. The usual publication media will be supplemented by an "update file" in the computerized phase diagram collection. Journals may begin to present phase diagram data in a format designed for entry into a computer file.

Much of what we have just described is still in the planning stage and, hence, free to be expanded, modified, or discarded as unforeseen problems or new applications for these methods become apparent.

We welcome ideas and suggestions for new applications of computerized phase diagrams, especially with respect to evaluation methods and thermodynamic calculations.

Acknowledgments

This program is the result of collaboration of many contributors at NBS; Lawrence Bennett, John Cuthill and Russell Kirsch began the dialogue between the Alloy Data Center and the Center for Applied Mathematics, and Lydon Swartzendruber[19] wrote an early version of this program. Christoph Witzgall and Marjorie McClain provided the spline fitting routines. David Redmiles has done the computer programming of our present version of the interactive graphics program. We have had very useful discussions with Lawrence Bennett, John Cuthill, John Simmons and Ted Massalski.

References

1. J.R. Cuthill, D.J. Kahan, L.H. Bennett, H.D. Chafe, A.G. Gray, E.L. Langer and H. Baker, Joint Program of American Society for Metals and National Bureau of Standards for Compilation of Alloy Phase Diagrams, Interamerican Conference on Materials Technology, Sao Paulo, Brazil (1978).

2. W.B. Pearson, Remarks on Producing and Publishing Critically Evaluated Data, Proceedings of Workshop on Applications of Phase Diagrams in Metallurgy and Ceramics, NBS SP-496, p 720 (1978).

3. I. Ansara, Comparison of Methods for Thermodynamic Calculation of Phase Diagrams, *Int. Met. Rev.*, No. 1, p 238 (1979).

4. A.D. Pelton, C.W. Bale and W.T. Thompson, F*A*C*T (Facility for the Analysis of Chemical Thermodynamics) – A Computerized Canadian Thermodynamic Data Treatment Centre, NBS SP-496, p 1077 (1978).

5. L. Kaufman and H. Bernstein, *Computer Calculation of Phase Diagrams, with Special Reference to Refractory Metals*, Academic Press, New York (1970).

6. A.T. Dinsdale, A Computer Program to Calculate Phase Equilibria in Ternary Alloy Systems, National Physical Laboratory, Division of Chemical Standards, Teddington, DCS Internal Report 1/78, February (1978).

7. G.P. Willers, The Construction of Thermodynamic Phase Diagrams through Computer Graphics, M.S. thesis, Iowa State University (1978).

8. Y. Mishima, S. Ishino and S. Iwata, An Approach to Information Processing of Phase Diagrams, *Mater. Sci. Eng.*, *11*, p 163-176 (1973).

9. B. Sundman and J. Agren, A Regular Solution Model for Phases with Several Components and Sublattices, Suitable for Computer Applications, Report No. 6, Institute for Metallograpfi, KTH Stockholm (1978).

10. M.-L. Saboungi and M. Blander, Theoretical Concepts Useful in the Calculation or Storage of Phase Diagrams of Ionic Systems, NBS SP-496, p 1093 (1978).

11. Yu. P. Adler and V.C. Stein, *Composition-Property Diagrams: Representation of Information for Computers, Information Storage and Retrieval*, Vol. 4, p 329-332, Pergamon Press, New York (1969).

12. P. Nash, An Interactive Ternary Phase Diagram Display Program, *Bulletin of Alloy Phase Diagrams*, to be published (1980); P. Nash and D.R.F. West, *Met. Sci.* (in press).

13. O. Kubaschewski and C.B. Alcock, *Metallurgical Thermochemistry*, 5th edition, Pergamon Press, New York (1979).

14. E.-Th. Henig, H.L. Lukas, B.B. Zimmerman and G. Petzow, Optimization of Phase Diagrams by a Least Squares Method Using Simultaneously Different Types of Data, NBS SP-496, p 955 (1978).

15. M. Hillert, Empirical Methods of Predicting and Representing Thermodynamic Properties of Ternary Solution Phases, Report No. TRITA-MAC-0143, Royal Institute of Technology, Stockholm (1979).

16. S.K. Tarby, C.J. Van Tyne and M.L. Boyle, Further Investigation of the Determination of Solute Interaction Parameters by Analysis of Phase Equilibria Using a Linear Programming Technique, *Met. Trans.*, *8B*, p 347 (1977).

17. H. Harvig, T. Nishizawa and B. Uhrenius, Application of Computer Methods for Optimizing the Thermochemical Evaluation of Equilibrium Data in Ternary Fe-C Alloys, in *Metallurgical Chemistry*, Proceedings of a Symposium held at Brunel University and the National Physical Laboratory, U.K., 1971, H.M.S.O., London (1972).

18. C. de Boor, *A Practical Guide to Splines*, Springer-Verlag, New York (1978).

19. R.A. Kirsch and L.J. Swartzendruber, Interactive Graphic Demonstration, NBS SP-496, p 265 (1978).

D ≡ *Bulletin of Alloy Phase Diagrams*, ASM INTERNATIONAL, Metals Park, OH. ***IAEA No. 7*** ≡ *Molybdenum: Physico-Chemical rties of Its Compounds and Alloys, Special Issue No. 7*, International Atomic Energy Agency, Vienna (1980). ***IMR*** ≡ *International Me- Reviews*, The Metals Society, London, and ASM INTERNATIONAL, Metals Park, OH. ***JAPD*** ≡ *Journal of Alloy Phase Diagrams*, In- itute of Metals, Calcutta. ***PDBBA*** ≡ *Phase Diagrams of Binary Beryllium Alloys*, ASM INTERNATIONAL, Metals Park, OH. ≡ *Phase Diagrams of Binary Gold Alloys*, ASM INTERNATIONAL, Metals Park, OH. ***PDBTA*** ≡ *Phase Diagrams of Binary Alloys*, ASM INTERNATIONAL, Metals Park, OH. ***PDTIA*** ≡ *Phase Diagrams of Ternary Iron Alloys*, ASM INTERNATIONAL, ark, OH. ***TIIM*** ≡ *Transactions of the Indian Institute of Metals*, Indian Institute of Metals, Calcutta.

C-Cr-V 82-11-0737
H. Holleck, Ternary Carbide Systems of High-Melting Transition Elements.-II., *Metall.*, *35*(12), p 1246-1253 (1981) in German.

C-Cr-W
See *C-Cr-V* *82-11-0737.*

C-Cr-Zr
See *C-Cr-V* *82-11-0737.*

C-Fe 82-11-0617
Z. Kiedrzynski, Ultrasonic-Thermal Studies of Fe + C Alloys for Eutectic Surroundings and for Hypereutectic Compositions of the Fe–C System, *Bull. Acad. Pol. Sci. Tech.*, *28*(9/10), p 507-514 (1980).

82-11-0618
Z. Kiedrzynski, Ultrasonic-Thermal Studies of Changes in Fe + C Alloys With Compositions Within the Hyperperitectic–Hypoeutectic Range for Conditions Tending to Equilibrium, *Bull. Acad. Pol. Sci. Tech.*, *28*(9/10), p 515-522 (1980).

82-11-0686
B. Chicco and W. R. Thorpe, Experimental Determination of the Austenite + Liquid Phase Boundaries of the Fe–C System, *Metall. Trans. A*, *13A*(7), p 1293-1297 (1982).

82-11-0741
T. Nishizawa and M. Hasebe, Computer Calculation of Phase Diagrams of Iron Alloys, *Tetsu-to-Hagane, J. Iron Steel Inst. Jpn.*, *67*(14), p 2086-2097 (1981) in Japanese.

82-11-0743
T. Okamoto, Z. Morita, A. Kagawa, and T. Tanaka, Partition of Carbon Between Solid and Liquid in Fe–C Binary System, *Tetsu-to-Hagane, J. Iron Steel Inst. Jpn.*, *68*(2), p 244-250 (1982) in Japanese.

C-Fe-Mo
See *C-Fe* *82-11-0741.*

C-Fe-O-Si 82-44-0174
E. K. Borodulin, G. K. Moiseev, and N. A. Vatolin, Modelling the Equilibrium in the Iron–Carbon–Silicon Dioxide System, *Dokl. Akad. Nauk SSSR*, *259*(3), p 596-599 (1981) in Russian.

C-Fe-W
See *C-Fe* *82-11-0741.*

C-Mo-Zr 82-12-1196
O. Matsumoto, Y. Yaguchi, T. Kajiwara, M. Konuma, and Y. Kanzaki, Preparation and Properties of a Ternary Cubic Solid Solution in the Mo–Zr–C System, *High Temp. Sci.*, *14*(3), p 161-169 (1981).

C-Na-O 82-11-0598
A. A. Gokhale and D. L. Johnson, Recomputation of Phase Equilibria in the Sodium–Carbon–Oxygen System: Effect of Oxygen, *Metall. Trans. A*, *13A*(6), p 1101-1102 (1981).

C-Nb-Ta
See *General and Miscellaneous* *82-11-0689, 82-72-0383.*

C-Ni-Pu 82-11-0690
Y. Arai, Y. Suzuki, T. Sasayama, and H. Watanabe, Ternary Compounds $PuNiC_2$ and $PuCoC_2$, *J. Nucl. Sci. Technol.*, *19*(3), p 83-86 (1982).

C-Os-U
See *Be-Os* *82-42-1116.*

C-Os-W
See *Be-Os* *82-42-1116.*

C-Zr
See *General and Miscellaneous* *82-11-0706* and *82-11-0707.*

Cd-Pd 82-11-0682
Y. Maa, A. Mikula, Y. A. Chang, and W. Schuster, Phase Stability Investigations of the Palladium–Cadmium System. I.–Thermodynamic Studies, *Metall. Trans. A*, *13A*(7), p 1115-1121 (1982).

82-11-0683
J. P. Neumann, A. Mikula, and Y. A. Chang, Phase Stability Investigations of the Palladium–Cadmium System. II.–Structural Studies, *Metall. Trans. A*, *13A*(7), p 1123-1126 (1982).

Cd-Se-Tl 82-11-0634
S. K. Karimov, S. Gaforov, and S. Sultonov, Phase Equilibria in the Systems Cd–Tl–C^{VI}(C^{VI}–Se, Te), *Izv. Akad. Nauk SSSR, Neorg. Mater.*, *17*(8), p 1346-1349 (1981) in Russian.

Cd-Te-Tl
See *Cd-Se-Tl* *82-11-0634.*

Ce-Ni-Sn
See *La-Ni-Sn* *82-12-1187.*

Co-Cr-Fe 82-11-0559
C. Allibert, C. Bernard, G. Effenberg, H.-D. Nüssler, and P. J. Spencer, A Thermodynamic Evaluation of the Fe–Co–Cr System, *Calphad*, *5*(4), p 227-237 (1981).

See *C-Fe* *82-11-0741.*

Co-Dy-Sn
See *Co-Sn-Y* *82-33-1668.*

Co-Er-Sn
See *Co-Sn-Y* *82-33-1668.*

Co-Fe-Nb 82-11-0750
L. A. Panteleimonov, O. G. Burtseva, and V. V. Zubenkov, Physico-Chemical Investigations of the Eighth Group Elements Doped by Niobium, *Vestn. Moskovskogo Univ. (Khim.)*, *23*(1), p 62-64 (1982) in Russian.

Co-Fe-Zr 82-11-0747
L. A. Panteleimonov, O. G. Burtseva, and V. V. Zubenkov, The Iron–Cobalt–Zirconium System, *Vestn. Moskovskoga Univ. (Khim.)*, *22*(6), p 609 (1981) in Russian.

Co-Ho-Sn
See *Co-Sn-Y* *82-33-1668.*

Co-Lu-Sn
See *Co-Sn-Y* *82-33-1668.*

Co-Mo-O 82-11-0584
A. Svoboda, S. Windisch, and H. Nowotny, Investigations of the Ternary System Chromium–Molybdenum–Oxygen, *High Temp. High Pressures*, *13*(4), p 427-434 (1981).

Co-Nb-Ni
See *Co-Fe-Nb* *82-11-0750.*

Co-Ni-S 82-11-0563
Y. Y. Chuang, K. C. Hsieh, and Y. A. Chang, Extension of the Associated Solution Model to Ternary Metal–Sulfur Melts: Ni–Co–S at 1273 °K, *Calphad*, *5*(4), p 277-289 (1981).

Co-Re-Y 82-11-0749
S. T. Samyratov, U. M. Makanov, I. G. Sokolova, and E. M. Sokolovskaya, Interaction Between the YCo_2 and YRe_2 Phases, *Vestn. Moskovskoga Univ. (Khim.)*, *23*(1), p 61-62 (1982) in Russian.

技術資料

UDC 669.112 : 681.3

鉄合金の状態図のコンピュータ解析 (2)

西沢 泰二*・長谷部 光弘*

Computer Calculation of Phase Diagrams of Iron Alloys

Taiji NISHIZAWA and *Mitsuhiro* HASEBE

5. 計算状態図の実例

前章までに鉄系状態図のコンピュータ解析についての一般論を述べたので, 本章では, その成果の代表例について説明する.

5·1 2元系状態図

5·1·1 Fe-C および Fe-N 2元系

Fe-C 系の状態図は合金状態図の元祖ともいうべきもので, 古くから多くの研究が行われてきた. この系の"実験状態図"については田中による詳しい集録[32]を参照されたい. 一方, 熱力学的検討もさかんに行われ, とくに DARKEN, GURRY[33] や BENZ, ELLIOTT[34] らによる研究は, 本稿でいうところの"計算状態図"の先駆的なものとして特筆に価する.

Fe-C 系状態図の本格的なコンピュータ解析は CHIPMAN[35] と AGREN[36] によって行われた. 図22は CHIPMAN による計算状態図の全容であり, また図23は αFe に対する Fe_3C と黒鉛の固溶度をアレニウス・プロットの形式で示したものである. αFe に対する Fe_3C の固溶度は, 従来は DIJKSTRA[37] や WERT[38] らの内耗測定にもとづいた経験式;

$$C\% = 2.55\exp\left(-\frac{9\,700\ \mathrm{cal\cdot mol^{-1}}}{RT}\right) \cdots\cdots(10)$$

が広く活用されてきた. しかし CHIPMAN の計算値は図23に破線で示したように, (10)式よりもはるかに低濃度側にある. この相違について筆者らはつぎのように考える. すなわち Fe-C 系では, C原子が侵入型拡散しさえすれば平衡が成立するのではない. この状態は部分平衡

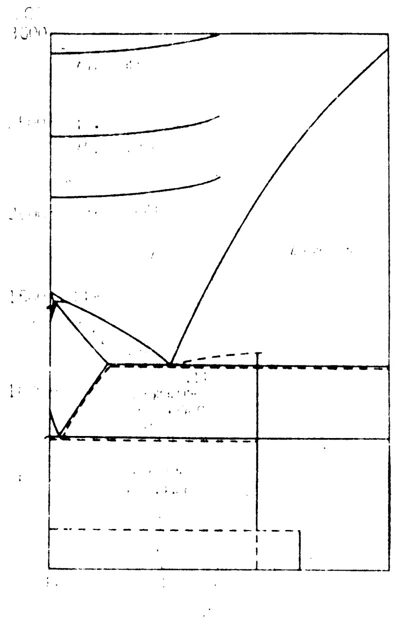


図22 Fe-C 系計算状態図[35]

昭和56年3月4日受付 (Received Mar. 4, 1981) (依頼技術資料)
* 東北大学工学部 工博 (Faculty of Engineering, Tohoku University, Aza-Aoba Aramaki Sendai 980)

鉄合金の状態図のコンピュータ解析 (1) は鉄と鋼 Vol. 67 No. 11 (9月号) に掲載. 目次, 以下のとおり.

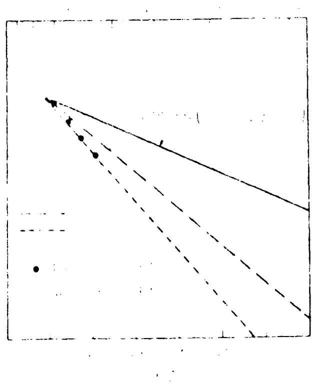


図23　αFe に対する Fe_3C（または黒鉛）の固溶度のアレニウス・プロット

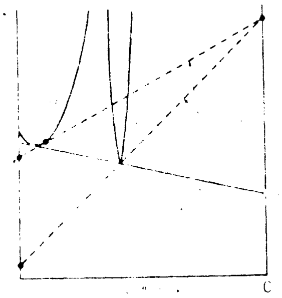


図24　Fe-C 系における部分平衡（説明図）
（C原子については平衡の条件；$\mu_C^\alpha = \mu_C^\beta$ が成立するが，Fe原子については成立していない）

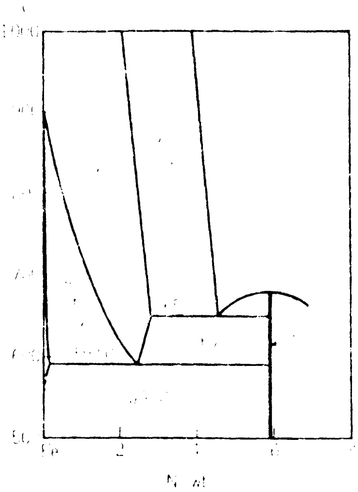


図25　Fe-N 系計算状態図[36)]

図 24 の点線）に相当するのであって，真の平衡に達するには Fe 原子もある程度移動することが必要である．したがって，約 100℃以下の実測値にもとづく WERT の(10)式は部分平衡の状態における Fe_3C の固溶度を示すものと認識すべきである．αFe-C 系の固溶度については，この種の議論はかなり古くから繰り返され，平衡...ていた[37)39)40)]．図 23 に示した実...平衡法によって求めたもので，CHIPMAN の...とよく一致している．

また Fe-N 系も実用上の重要性等の理由で古くから研究されてきた系である．N_2 は1気圧では鉄中に極めて微量（αFe に対してCの約 1/10，γFe に対してCの約 1/100）しか固溶しない．この場合の状態図の熱力学的解析は McLELLAN ら[42)]によってなされている．いわゆる Fe-N の状態図は NH_3 等との反応によって N を...した場合のものであって，HILLERT ら[43)]や AGREN[36)]がコンピュータ解析している．図25に AGREN による計算状態図を示した．

5·1·2　Fe-X 2元系

鋼中に含まれる X元素を考えるとき，...X が α 相を安定化するか，γ 相を安定化するかということである．この点を定量的に調べるには，Fe-X 2元系の状態図における α/γ 平衡を解析することが必要であって，ZENER[44)]をはじめとして多くの研究[45)～47)]が行われた．本邦では和田[15)48)]による優れた研究がある．

4 章の冒頭でも述べたように，鉄の A_3 変態は αFe の強磁性によって引き起こされた異端者的な変態である．したがって Fe-X 系の α/γ 平衡についても強磁性の効果を無視するわけにはいかない．そこで，強磁性を考慮した自由エネルギの近似式（5）～（7）を使って解析してみた[49)]．図 26 は α 安定化型の元素を添加した場合の，いわゆる γ ループである．図中で Fe-Be 系と Fe-Cr 系の γ ループは下ぶくれが著しく，とくに Fe-Cr系はよく知られた極小点をもつが，これらは強磁性の効果にもとづくものである．一方，図 27 は γ 安定化型元素を添加した場合の状態図で，Fe-Co 系の α/γ 平衡線が異様に曲りくねっているのも，やはり強磁性の効果であって，αFe と fcc Co の両者の磁気変態を考慮した計算の結果は実測値とよく一致した．

Fe-X 2元系のうちで相平衡が熱力学的に解析され，状態図が計算されたものを前節の Fe-C，Fe-N 系も含めて表1に掲げた．また，これらの中の代表的なものを図 28，29 に図示した．

5·2　3元系状態図

5·2·1　Fe-C-X 3元系

Fe-C-X 3元系は鉄鋼材料の基本系であって，本邦においても村上，佐藤，武田，武井らによって卓越した研究が行われた．しかし“計算状態図”は表5に見られるように，ほとんどがスウェーデンの HILLERT，UHRENIUS らのグループによって行われたものである．代表例として Fe-C-B[82)]，Fe-C-Cr[83)]，Fe-C-Mo[83)]，Fe-C-W[84)] の各3元系の等温状態図と γ 単相域の温度変化を図30～

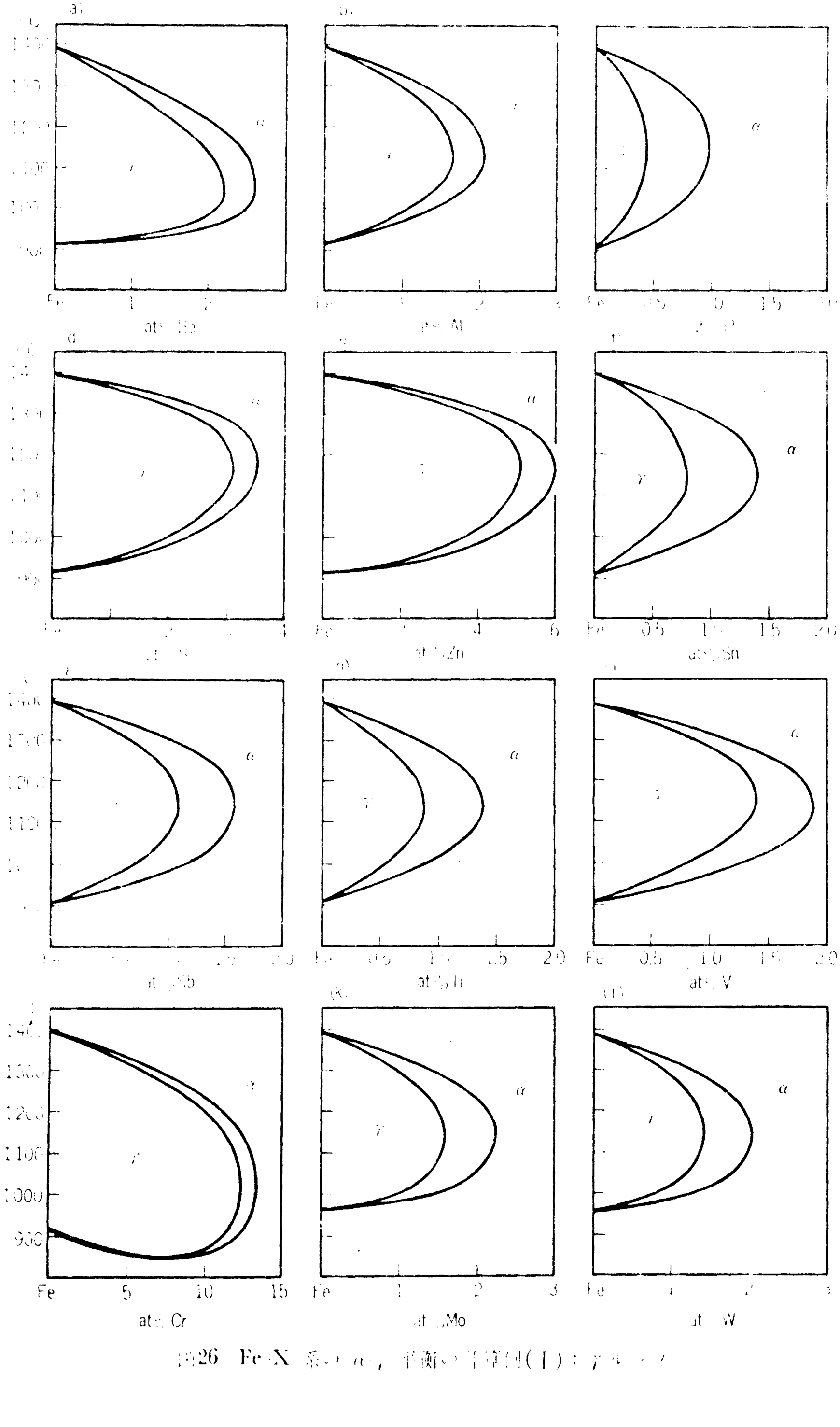


図26 Fe-X 系のα-γ平衡の計算図(I)：γループ

34に示した。

Fe-C-X 3元系の共析点，すなわちγ単相域の最低点の合金元素の添加による変化も最もたいせつな要点である。図34 (a) は1939年，BAIN[96] [illegible]。図34 (b) はUHRENIUS[98]が計算によって求めたもので，両者にわずかながら不一致が見られる。実験と計算の両面からさらに検討を要すると思われる。なお図34 (b) の図中の●印は，γ相と平衡する[illegible]の種類が変わる境界組成を意味している。

[illegible] 5に掲げなかったものの中で重要と思われるも

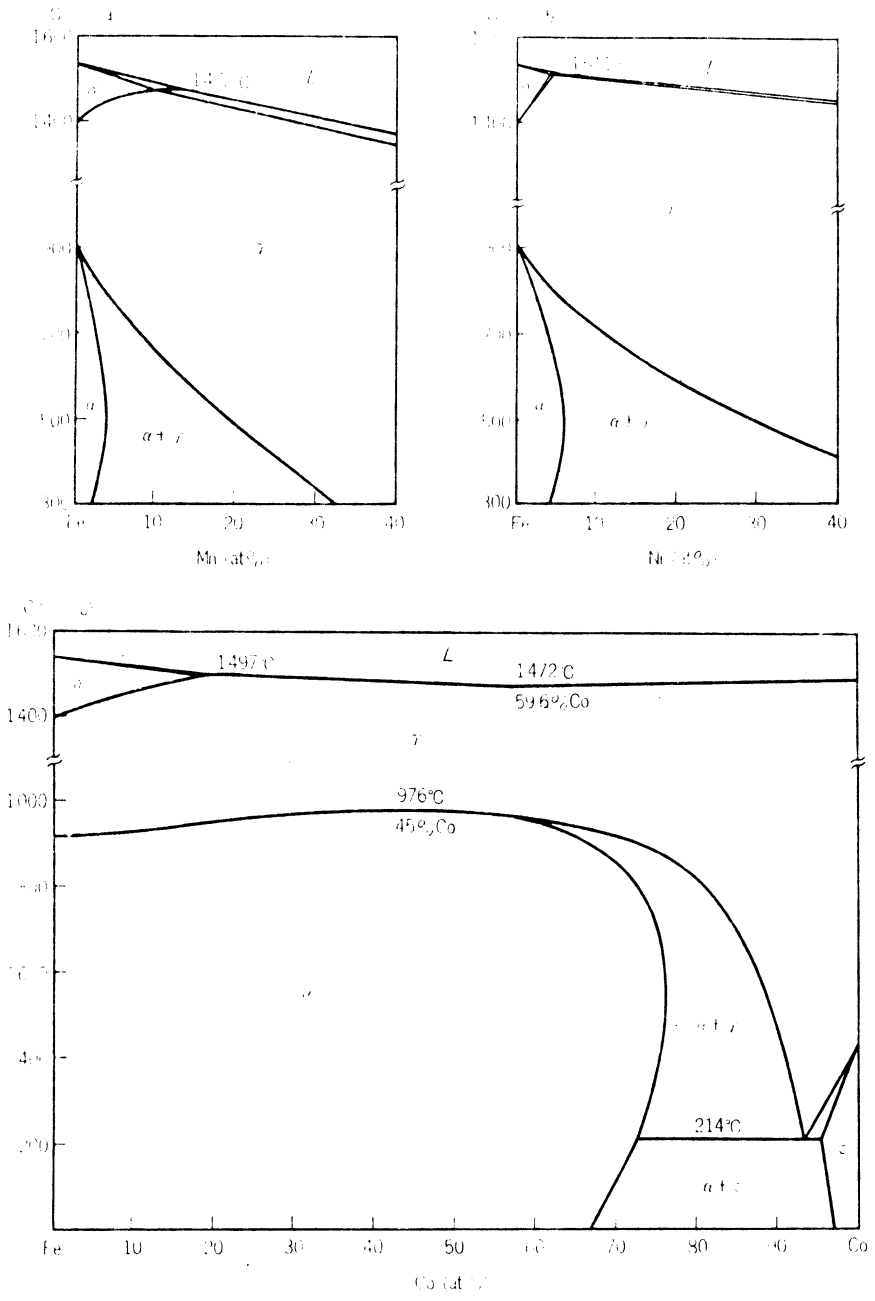


図27　Fe-X 系の α/γ 平衡の計算例（II）
（参照ならびに Fe-Co 系の ε 相（hcp）に関する平衡関係も記載してある）

らに KIRKALDY らの研究がある[97][98]．彼らは Fe-C 系（2 wt%，以下）の状態図が，Mn, Si, Ni, Cr, Mo, Cu, V, Nb, W, Co の 10 種の合金元素の添加によってどのように変化するかを熱力学的に計算した．彼らの計算は希薄な（Mn 3 wt%，Si 1 wt%，Cr（2.5 wt%，Mo 2 wt%，Cu 3 wt%，その他 1 wt% でかつ全体で 5 wt% 以下）の変化を行わせたので合金組成に制限はあるが，そのため逆に，計算が簡明であるという利点があるので参照していただきたい．

5・2・2　Fe-X-Y 3 元系

表 6 に Fe-X-Y 3 元系に関する“計算状態図”のこれまでの成果をまとめた．また代表例として Fe-S-Mn[100]，Fe-Cr-Si[99]，Fe-Cr-Co[104]，Fe-Cu-Mn[4]，Fe-Cu-Ni[4] の各系の等温切断図を図 35～39 に示した．

表4 鉄合金の計算状態図一覧

系	相または平衡系	温度範囲	文献
Fe-Be	Solubility in α	500～1 180°C	14) 図9, 図13,
	γ ループ	910～1 390°C	49) 図26参照
Fe-C	α, γ, Fe_3C, 黒鉛, L	500～1 700°C	34) 35) 36) 50)
	α, γ, Fe_3C, 黒鉛	500～1 150°C	33) 51) 52) 図22
	Solubility in α	910°C以下	41) 図23参照
Fe-N	α, γ, ε, $\gamma'(Fe_2N)$	300～750°C	36) 43) 図25参照
Fe-O	α, γ, L, FeO, Fe_3O_4, Fe_2O_3, Vapor	200～2 100°C	53)
Fe-Al	α, γ, L, Fe_3Al_2, $FeAl_2$, Fe_2Al_5, $FeAl_3$	200～1 700°C	54) 55) 図26参照
	γ ループ	910～1 390°C	49)
Fe-Si	α, γ, L, Fe_2Si, FeSi, Fe_3Si_7, Si	200～1 600°C	56)
	α, L, Fe_2Si, FeSi, Si(液相線)	1 180～1 600°C	57)
	γ ループ	910～1 390°C	49) 図26参照
Fe-P	α, γ, L, Fe_3P, Fe_2P, FeP	200～1 600°C	58)
	Solubility in α	500～1 100°C	14) 図13, 図26
	γ ループ	910～1 390°C	49) 参照
Fe-S	α, γ, L, $Fe_{1-\delta}S$	300～1 600°C	59) 図28参照
	α, γ, L, FeS：<50 at%	870～1 600°C	60)
Fe-Ti	α, γ, ε, L, Fe_2Ti, FeTi	200～2 200°C	61) 62)
	Solubility in α	500～1 310°C	14)
	γ ループ	910～1 390°C	49) 図26参照
Fe-V	α, σ	200～1 200°C	63) 64)
	γ ループ	910～1 390°C	49) 図26参照
Fe-Cr	α, γ, L, σ	200～1 900°C	4) 65)
	α, L, σ	300～1 800°C	66) 図16, 26, 29
	液相線, 固相線	1 520～1 900°C	67) 参照
	γ ループ	840～1 390°C	49) 68)
Fe-Mn	α, γ, L, αMn, βMn	200～1 600°C	69)
	α, γ, L	200～1 600°C	4) 図27参照
	γ, L(液相線, 固相線)	1 250～1 500°C	70)
	α, γ	910°C以下	17) 68)
Fe-Co	α, γ, ε, L	200～1 600°C	49) 62) 65) 71) 図27参照
Fe-Ni	α, γ, L, $FeNi_3$	200～1 700°C	65)
	α, γ, L	200～1 600°C	4) 図27参照
	γ, L(液相線, 固相線)	1 400～1 600°C	67)
	α, γ	910°C以下	17)
Fe-Cu	α, γ, L	500～1 600°C	13) 72) 図13, 29参照
	γ, L	850～1 580°C	73)
	α, γ	200～1 100°C	11)
Fe-Zn	α, γ, L	300～1 400°C	74)
	γ ループ	910～1 400°C	49) 図26参照
Fe-Nb	α, γ, L, σ, Fe_2Nb, Fe_2Nb_3	200～2 500°C	62) 75)
Fe-Mo	α, γ, L, σ, $\mu(Fe_3Mo)$	200～2 700°C	62) 75)
	α, γ, σ, $\mu(Fe_3Mo)$, $R(Fe_{1.7}Mo)$	900～1 500°C	76)
	Solubility in α	500～1 500°C	14)
	γ ループ	910～1 390°C	49) 図26参照
Fe-Ru	α, γ, ε, L	200～2 300°C	77) 78) 図29参照
Fe-Sn	γ, L, Fe_3Sn, Fe_3Sn_2, $Fe_{0.565}Sn_{0.435}$ FeSn,	300～1 600°C	79) 図26, 29参照
	$FeSn_2$ γ ループ	910～1 390°C	17) 49)
Fe-Sb	Solubility in α	500～1 000°C	14)
	γ ループ	910～1 390°C	17) 49) 図26参照
Fe-W	α, γ, L, $\mu(Fe_3W_2)$	200～3 700°C	62) 75) 80)
	α, γ, $\mu(Fe_3W_2)$	930～1 550°C	81) 図26参照
	Solubility in α	500～1 500°C	14)
	γ ループ	910～1 300°C	49)

α: bcc, γ: fcc, ε: hcp, L: liquid

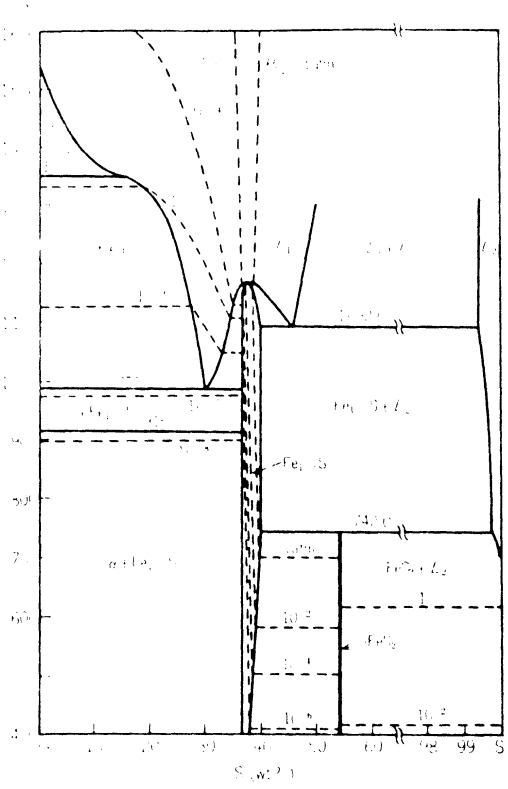


図 28 Fe-S 系状態図[59]

6. おわりに

Kaufman, Hillert, Ansara らによって“計算状態図”を主題とする国際会議 CALPHAD (Calculation of Phase Diagram) が組織されてから約 10 年が経過した[10]。表 4，5，6 に総括した文献のほとんどはこの CALPHAD に心を寄せる人たちによるものである。これまでの CALPHAD の主力は生台づくり——計算方式の確立——に注がれてきたのであって，状態図づくりはこれからである。本邦においてもこの方面への関心が高まることを期待したい。

文　献

32) 田中良平：鉄と鋼，53 (1967)，p. 1586
33) L. S. Darken and R. W. Gurry: Trans. Met. Soc. AIME, 191(1951), p. 1015
34) M. G. Benz and J. F. Elliott: Trans. Met. Soc. AIME, 221(1961), p. 323
35) J. Chipman: Met. Trans., 3(1972), p. 55
36) J. Ågren: Met. Trans., 10A(1979), p. 1847
37) L. J. Dijkstra: Trans. AIME, 185(1949), p. 252
38) C. Wert: Trans. AIME, 188(1950), p. 1242
39) L. S. Darken and R. W. Gurry: Physical Chemistry of Metals, (1953), p. 418[McGraw-Hill Book]
40) J. C. Swartz: Trans. Met. Soc. AIME, 239 (1967), p. 68; 245(1969), p. 1083
41) 大谷博司，長谷部光弘，西沢泰二：日本金属学会講演会（1980 年 4 月）に発表
42) R. B. McLellan and R. J. Farraro: Acta Met., 28(1980), p. 417
43) M. Hillert and M. Jarl: Met. Trans., 6A(1975), p. 553
44) C. Zener: Trans. AIME, 167(1946), p. 513
45) W. Oelsen: Stahl Eisen, 69(1949), p. 468
46) K. W. Andrews: JISI, 184(1956), p. 414
47) R. A. Buckley and W. Hume-Rothery: JISI, 201(1963), p. 227
48) 植田次康：日本金属学会誌，25(1961)，p. 707
49) 大谷博司，長谷部光弘，西沢泰二：日本金属学会講演会（1980 年 10 月）に発表
50) L. Kaufman and H. Nesor: Calphad, 2 (1978), p. 295
51) H. Harvig: Jernkont. Ann., 155(1971), p. 157
52) H. Gaye and C. H. P. Lupis: Met. Trans., 6A(1975), p. 1049
53) L. Kaufman and H. Nesor: Calphad, 2 (1978), p. 35
54) L. Kaufman and H. Nesor: Met. Trans., 5 (1974), p. 1623
55) L. Kaufman and H. Nesor: Calphad, 2 (1978), p. 325
56) L. Kaufman: Calphad, 3(1979), p. 45
57) R. Schmid: Calphad, 4(1980), p. 101
58) P. J. Spencer and O. Kubaschewski: Arch. Eisenhüttenw., 49(1978), p. 225
59) R. C. Sharma and Y. A. Chang: Met. Trans., 10B(1979), p. 103
60) M. Hillert and L. -I. Staffansson: Met. Trans., 6B(1975), p. 37
61) L. Kaufman and H. Nesor: Treatise on Solid State Chemistry, ed. by N. R. Hannay, NY, (1975), p. 179 [Plenum Press]
62) L. Kaufman and H. Nesor: Calphad, 2 (1978), p. 55
63) P. J. Spencer and F. H. Putland: JISI, 211 (1973), p. 293
64) P. J. Spencer and J. F. Counsell: Z. Metallk., 64(1973), p. 662
65) L. Kaufman and H. Nesor: Z. Metallk., 64 (1973), p. 249
66) F. Müller and O. Kubaschewski: High Temp. -High Press., 1(1969), p. 543
67) M. V. Rao and W. A. Tiller: Mater. Sci. Eng., 14(1974), p. 47
68) G. Kirchner, T. Nishizawa, and B. Uhrenius: Met. Trans., 4(1973), p. 167
69) L. Kaufman: Calphad, 2(1978), p. 117
70) M. V. Rao and W. A. Tiller: Mater. Sci. Eng., 15(1974), p. 87
71) L. Kaufman and H. Nesor: Ann. Rev.

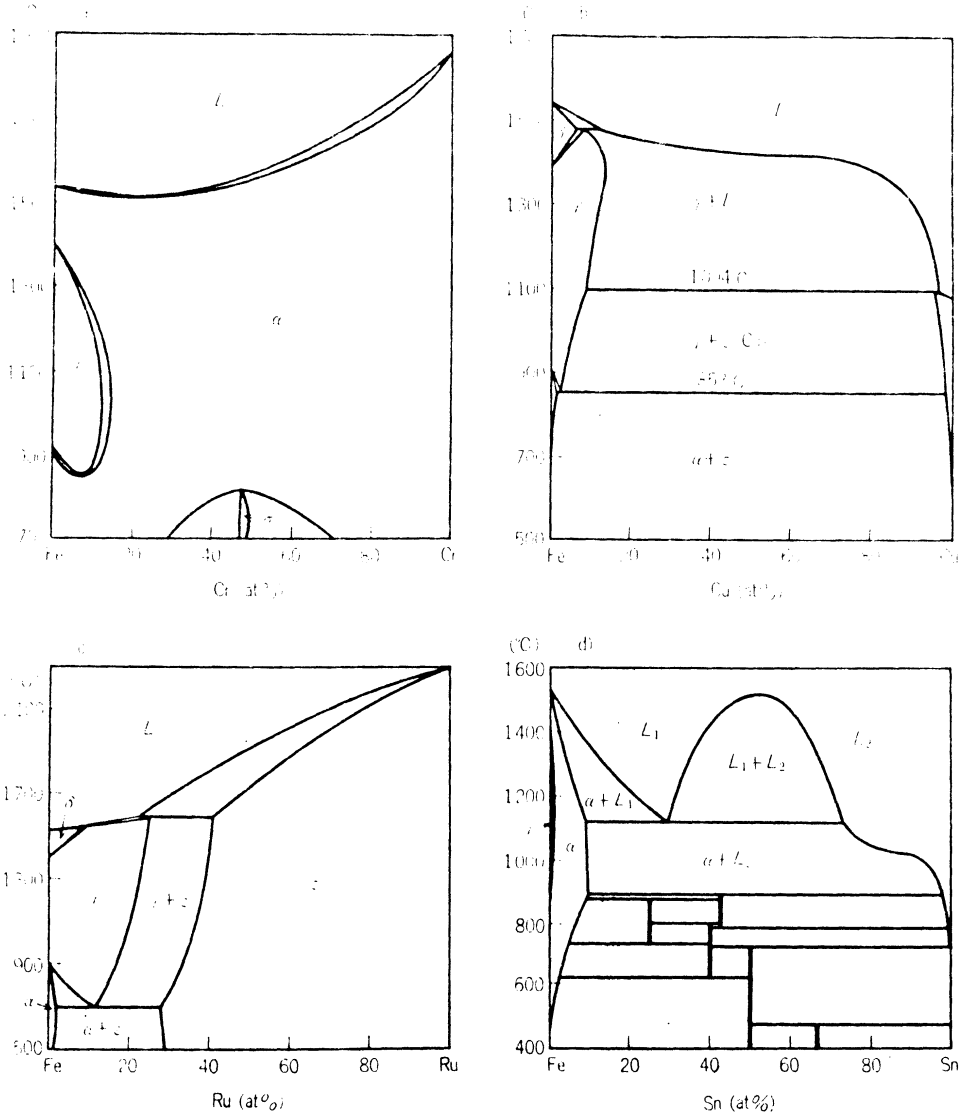


図29 Fe-Cr, Fe-Cu, Fe-Ru ならびに Fe-Sn 系の計算状態図
(Fe-Cr 系については図 16 参照)

Mater. Sci., ed *R.* Huggins, Palo Alto, USA, 3, (1973), p. 1

72) *O.* Kubaschewski, *J. F.* Smith, and *D. M.* Bailey: Z. Metallk., 68(1979), p. 495

73) *P.-A.* Lindqvist and *B.* Uhrenius: Calphad, 4 (1980), p. 193

74) *G.* Kirchner, *H.* Harvig, *K.-R.* Moquist, and *M.* Hillert: Arch. Eisenhüttenw., 44 (1973), p. 227

75) *L.* Kaufman and *H.* Nesor: Met. Trans., 6A (1975), p. 2123

76) *G.* Kirchner, *H.* Harvig, and *B.* Uhrenius: Met. Trans., 4(1973), p. 1059

77) *L.* Kaufman: Physical Metallurgy of Martensite and Bainite, Report No. 93, (1965), p. 48[BISRA and Institute of Metals]

78) *L.* Kamufan and *H.* Bernstein: Comptuer Calculation of Phase Diagrams, (1970), [Academic Press]

79) *H. D.* Nussler, *O.* Goldbeck, and *P. J.* Spencer: Calphad, 3(1979), p. 19

80) *B.* Uhrenius and *L.* Kaufman: Calphad, 3 (1979), p. 223

81) *B.* Uhrenius: Calphad, 4(1980), p. 173

82) 長谷部光弘, 西沢泰二: 日本金属学会誌, 38 (1974), p. 46

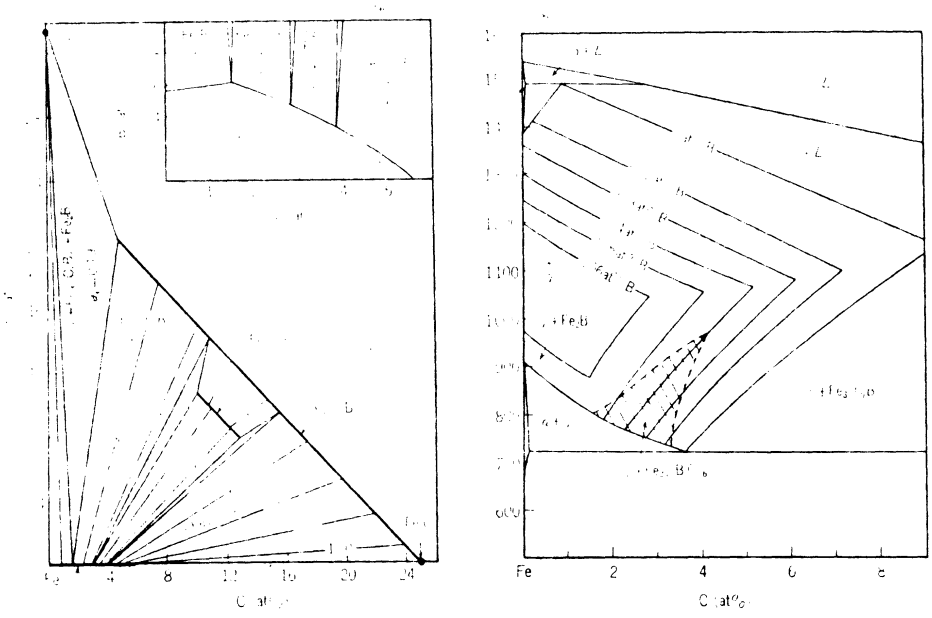
83) *B.* Uhrenius: Hardenability Concepts with Applications to Steel, ed, by *D. V.* Doane and *J. S.* Kirkaldy, (1978), p. 5[AIME]

84) *J.* Agren: Report Royal Inst. Technology,

表5　鉄合金の計算状態図一覧（Fe-C基3元系および4元系）

系	相および平衡関係	温度範囲	文献
Fe-C-B	α, γ, L, θ, G, $M_{23}C_6$, Fe_2B	400～1 100°C	82)　図30参照
Fe-C-Si	α, γ, θ, G	700～1 100°C	83) 84)　図34参照
Fe-C-Ti	α, γ, L, $\lambda(Fe_2Ti)$, $TiC_{0.78}$ 液相線	1 300～3 000°C	61)
Fe-C-V	α, γ, θ, $VC_{0.75}$	800～1 100°C	83)　図34参照
Fe-C-Cr	α, γ, θ, $M_{23}C_6$, M_7C_3	600～1 100°C	83) 85) 86)
	α, γ, θ, $M_{23}C_6$, M_7C_3	1 000°C	87)　図31, 34参照
	α, γ, θ	700～770°C	88)
Fe-C-Mn	α, γ, θ, $M_{23}C_6$	600～1 100°C	83) 89)　図34参照
	α, γ	700～770°C	90)
Fe-C-Ni	α, γ, θ	700～1 100°C	83)
	α, γ	730°C	91)
Fe-C-Cu	α, γ, θ	750～1 100°C	83)　図34参照
Fe-C-Mo	α, γ, θ, M_2C, M_6C, $\xi(M_3C)$, $M_{23}C_6$	700～1 000°C	83) 92)　図32, 34参照
	α, γ, θ, M_2C, M_6C, $\xi(M_3C)$	1 000°C	93) 94)
Fe-C-W	α, γ, θ, Fe_3W_2, WC, M_6C, L, G	1 000～1 500°C	81)　図33, 34参照
	α, γ, θ, WC	700～1 100°C	83)
	α, γ, θ, WC, M_6C, Fe_2W	1 000°C	94)
Fe-C-Cr-W	γ, θ, M_6C, $M_{23}C_6$	900～1 100°C	95)
Fe-C-Mo-W	α, γ, θ, WC, M_2C, M_6C, $\xi(M_3C)$ Fe_2W	1 000°C	94)

α: bcc, γ: fcc, L: 液相, θ: セメンタイト, G: 黒鉛



(a) 等温断面図　(b) オーステナイト相へのBの固溶度

図30　Fe-C-B 系 の 計 算 状 態 図

TRITA-MAC-0125, Stockholm, (1977), [quoted in (83)]

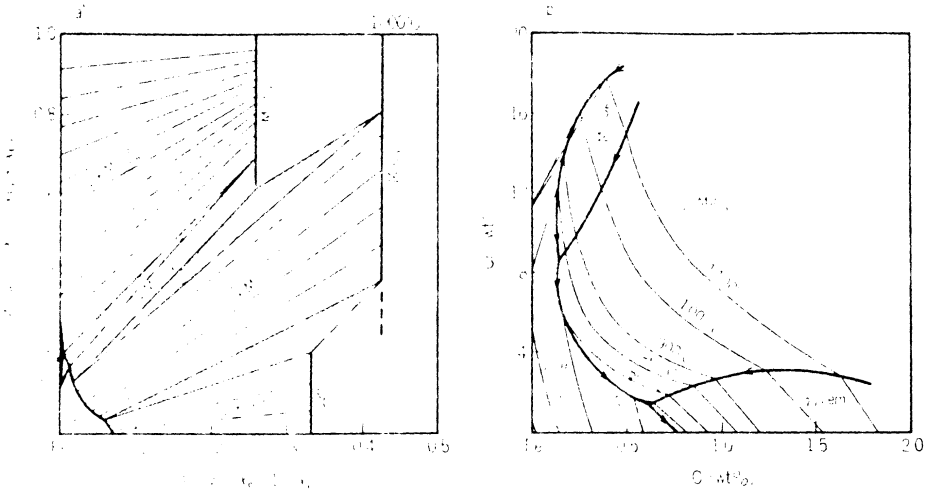
85) *R.* Lundberg, *M.* Waldenström, and *B.* Uhrenius: Calphad, 1 (1977), p. 159

86) *M.* Waldenström and *B.* Uhrenius: Scand. J. Met., 6 (1977), p. 202

87) *T.* Nishizawa and *B.* Uhrenius: Scand. J. Met., 6 (1977), p. 67

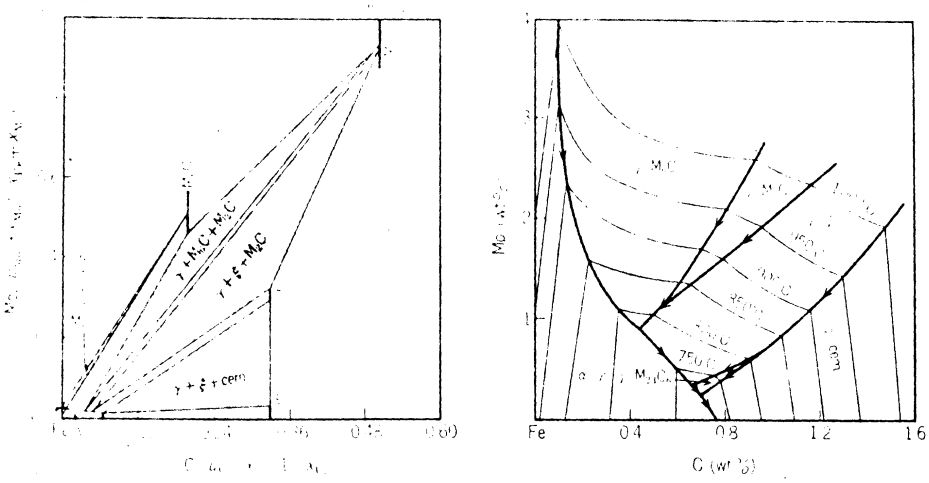
88) *R. C.* Sharma, *G. R.* Purdy, and *J. S.* Kirkaldy: Met. Trans., 10A (1979), p. 1119

89) *M.* Hillert and *M.* Waldenström: Calphad, 1 (1977), p. 97; Met. Trans., 8A (1977), p. 5



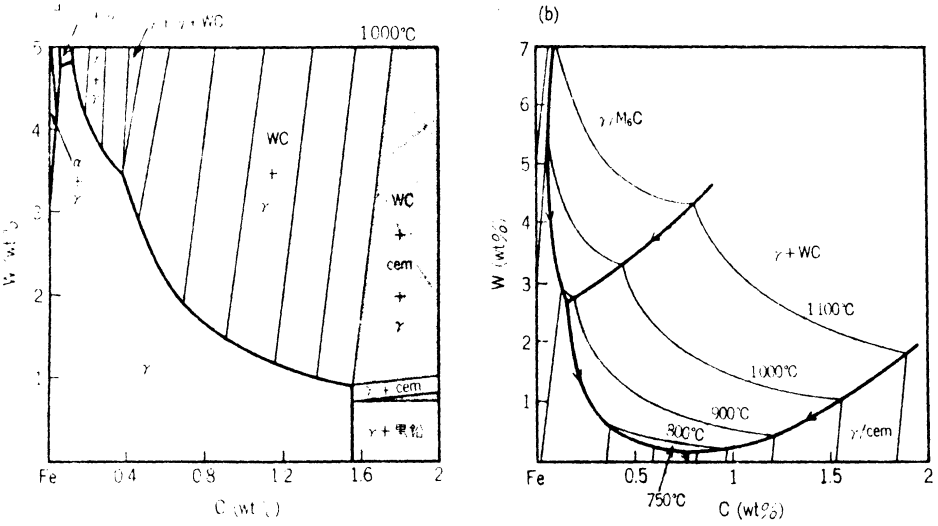
(a) 等温状態図　　(b) γ単相域の温度変化

図31 Fe-C-Cr 系の計算状態図[83]



(a) 等温状態図　　(b) γ単相域の温度変化

図32 Fe-C-Mo 系の計算状態図[83]



(a) 等温状態図　　(b) γ単相域の温度変化

図33 Fe-C-W 系の計算状態図[81][83]

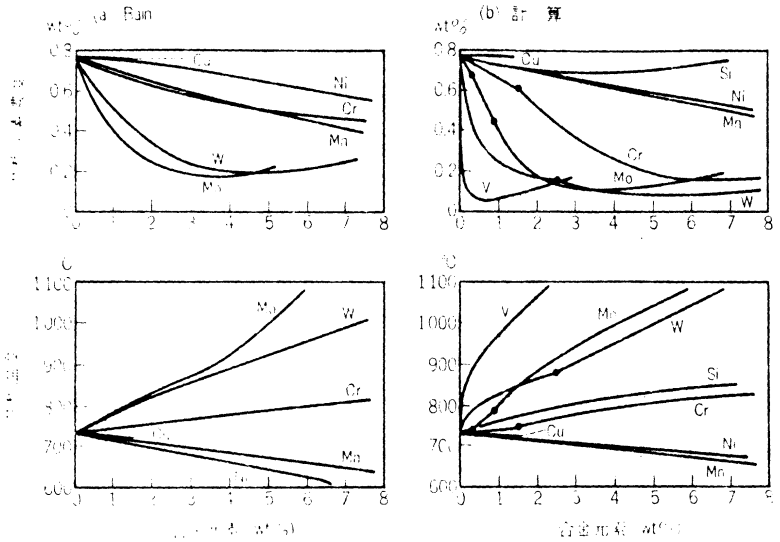
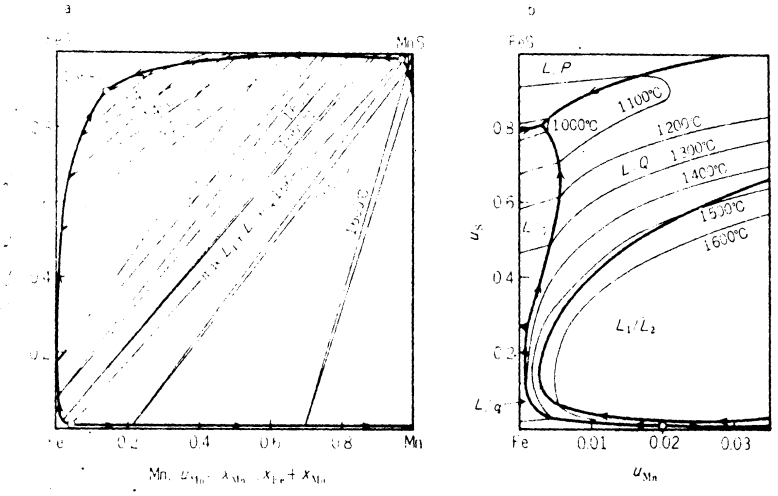


図34　Fe-C 系の共析点に対する合金元素の影響[83]



(a)　Fe-Mn-FeS-MnS 系の2液相分離面と L_1+L_2+MnS 3相3角形
(b)　Fe-FeS 側の拡大図（P：FeS, Q：MnS）
図35　Fe-S-Mn 系の計算状態図[100]

90) *J. B.* Gilmour, *G. R.* Purdy, and *J. S.* Kirkaldy: Met. Trans., 3(1972), p. 1455
91) *R. C.* Sharma and *J. S.* Kirkaldy: Canad. Met. Quart., 12(1973), p. 391
92) *C.* Chatfield and *M.* Hillert: Calphad, 1 (1977), p. 201
93) *T.* Nishizawa: Scand. J. Met., 1(1972), p. 41
94) *B.* Uhrenius and *H.* Harvig: Metal Sci., 9(1975), p. 67
95) *B.* Uhrenius and *S.* Grondell: Metal Sci., 11(1977), p. 73
96) *E. C.* Bain: Function of Alloying Elements in Steel, (1939), [ASM]
97) *J. S.* Kirkaldy and *E. A.* Baganis: Met. Trans., 9A(1978), p. 495
98) *J. S.* Kirkaldy, *B. A.* Thomson, and *E. A.* Baganis: Hardenability Concepts with Applications to Steel, ed. by *D. V.* Doane, and *J. S.* Kirkaldy (1978), p. 82 [AIME]
99) *T.* Chart, *F.* Putland, and *A.* Dinsdale: Calphad, 4(1980), p. 27
100) *M.* Hillert and *L.-I.* Staffanson: Met. Trans., 7B(1976), p. 203
101) *L.* Kaufman and *H.* Nesor: Met. Trans., 6A (1975), p. 2115
102) *I.* Ansara, *C* Bernard, *L.* Kaufamn, and *P. J.* Spencer: Calphad, 2(1978), p. 1
103) *G.* Kirchner and *B.* Uhrenius: Acta Met.,

表6 Fe合金の計算状態図（3元系および4元系）

系	相平衡関係	温度範囲	文献
Fe-Al-Cr	α, L	1 500～1 700°C	75)
Fe-Al-Ni	α, γ, L, Fe_3Al_2, $FeAl_2$, Fe_2Al_5 Ni_3Al, Ni_2Al_3	900～1 400°C	54)
Fe-Si-Cr	α, γ, σ, FeSi, Cr_3Si, CrSi	400～900°C	99)　図36参照
Fe-Si-Ni	α, γ, $FeNi_3$, Ni_3Si, Ni_5Si	400～500°C	99)
Fe-S-Mn	α, γ, L, FeS, MnS	1 000～1 700°C	100)　図37参照
Fe-Ti-Cr	α, γ, L, TiX_2, TiX	1 000～1 400°C	101)
Fe-V-Cr	α, σ	500～900°C	64) 102)
Fe-Cr-Mn	α, γ : 10%Mn	750～950°C	103)
Fe-Cr-Co	α, γ, ε, L, σ	400～1 500°C	104) 105)　図37参照
	α, γ, σ	1 200°C	65) 71) 106)
Fe-Cr-Ni	α, γ, L, σ, $FeNi_3$	400～1 600°C	105) 107)　図7, 8参照
	α, γ, L, σ	800～1 500°C	4)
	α, γ, σ	400～600°C	99)
	α, γ	1 100～1 200K	108) 109)
Fe-Cr-Mo	α, γ, L, σ, μ(Fe_3Mo_2)	650～1 700°C	75)
Fe-Cr-W	〃　μ(Fe_3W_2)	1 100～1 700°C	75)
Fe-Mn-Cu	γ, L	600～1 300°C	4)　図38参照
Fe-Co-Ni	α, γ	600～800°C	65) 71) 105)
Fe-Ni-Cu	γ, L	400～1 400°C	4)　図39参照
	γ : Msibility gap	850～1 400°C	109)
Fe-Mo-W	α, γ, μ(Fe_7X_2), R($Fe_{1.7}X$)	1 100～1 300°C	76)
Fe-Si-Cr-Ni	α, γ, σ : σ の安定領域	527°C	99)

α : bcc, γ : fcd, ε : hcp, σ : シグマ, L : 液相

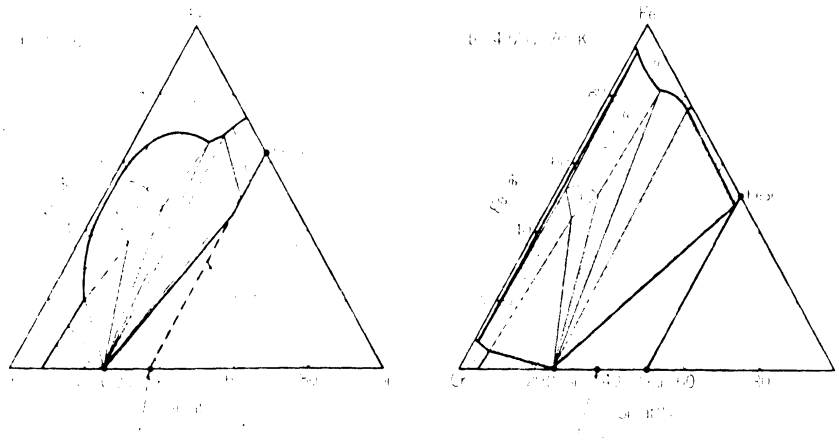


図36 Fe-Cr-Si 系計算状態図[99]

22(1974), p. 523
104) 長谷部光弘, 西沢泰二: 日本金属学会講演会 (1977年10月) に発表
105) L. Kaufman and H. Nesor: Met. Trans., 5 (1974), p. 1617
106) T. G. Chart, J. F. Counsell, G. P. Jones, W. Slough, and P. J. Spencer: Intern. Met. Rev., 20(1975), p. 57
107) L. Kaufman, J. S. Watkin, and A. P. Miodownik: Calphad, 1(1977), p. 281
108) M. Hillert and M. Waldenström: Scand. J. Met., 6(1977), p. 211
109) J. F. Counsell, E. B. Lees, and J. Spencer: Metallurgical Chemistry, ed. by O. Kubaschewski (1972), p. 451 [HMSO]
110) 西沢泰二: 日本金属学会会誌, 18(1979), p. 706

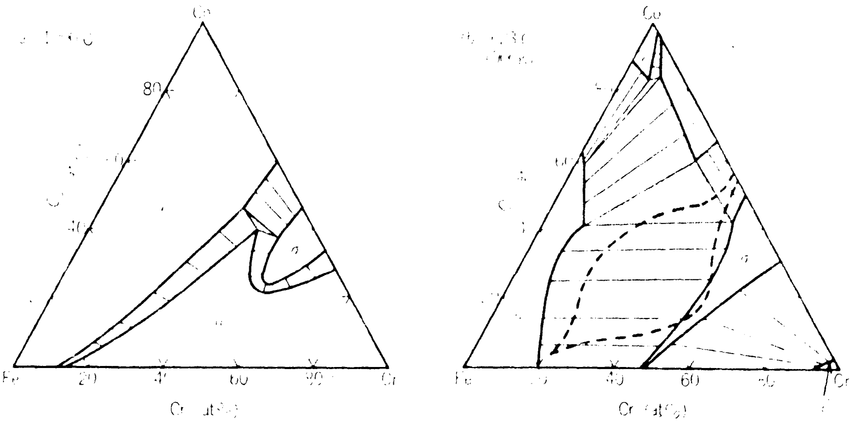
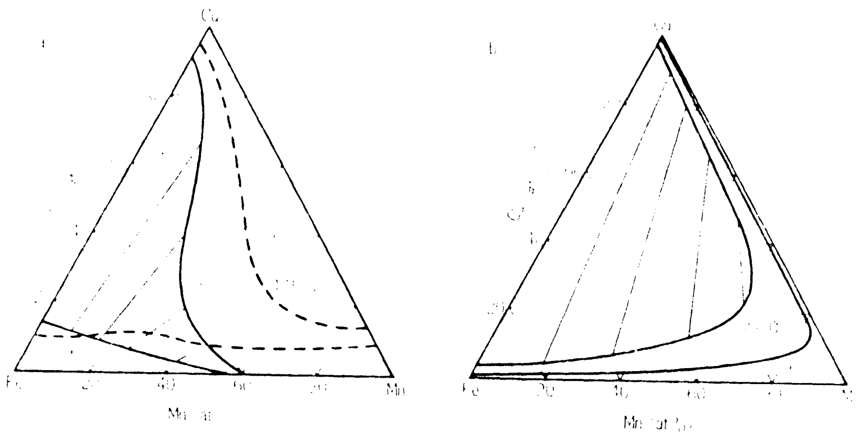
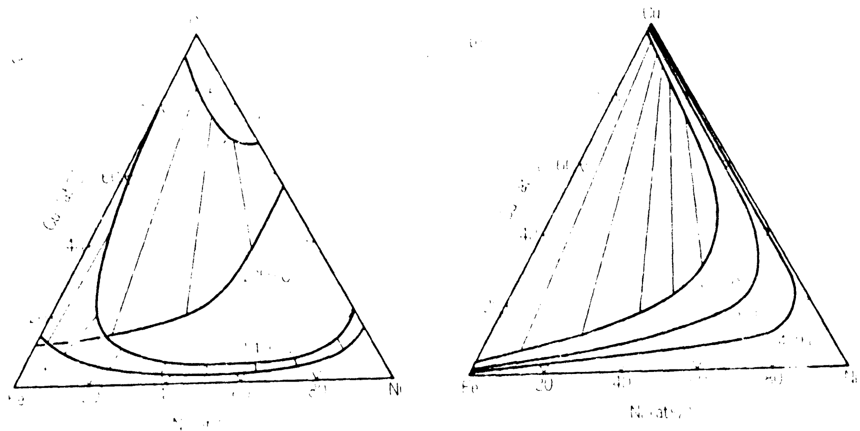


図37 Fe-Cr-Co 系の計算状態図（準安定2相分離面(点線)については図 20, 21 参照）



（a） fcc/液相平衡 （b） fcc 相の2相分離面の等温断面図
図38 Fe-Cu-Mn 系の計算状態図



（a） fcc/液相平衡 （b） fcc 相の2相分離面の等温断面図
図39 Fe-Cu-Ni 系の計算状態図

CREEP AND VISCOELASTICITY

EXAMPLE

The current values are:

	Application	Time,ks	Stress,MPa	Strain
1.	Pos Step	0.0000	100.0000	0.2000
2.	No Change	0.0200	100.0000	1.0000
3.	No Change	0.0400	100.0000	1.2000
4.	No Change	0.1000	100.0000	1.4000

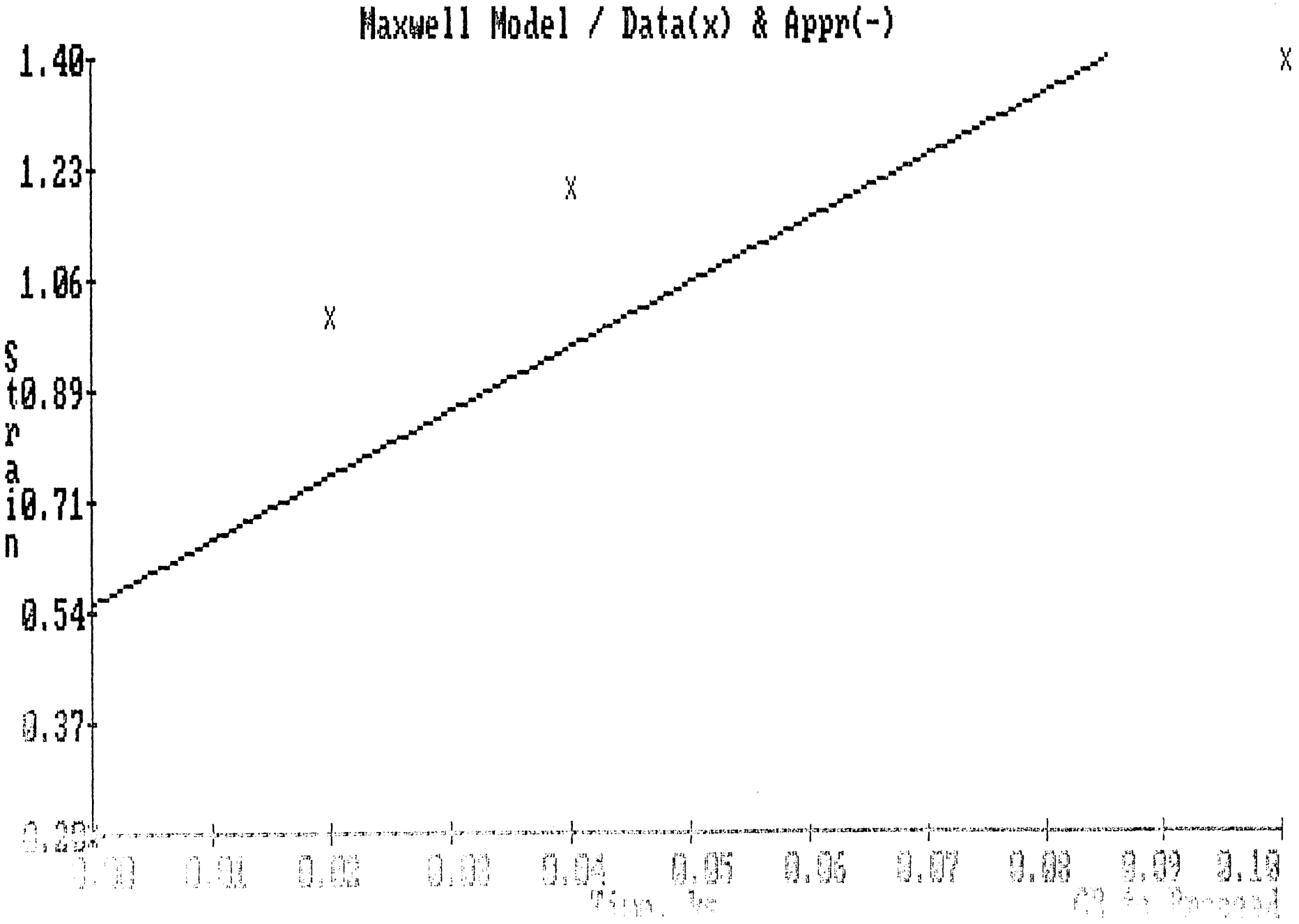
Press Return to Proceed

Select Constant of Nonproportional Behavior,0-(1):

Maxwell Model:	Specified E1,GPa	Calculated n1,GPa-s
	181.8182	10.0000

Maxwell Model:	Calculated E1,GPa	Specified n1,GPa-s
	181.8182	10.0000

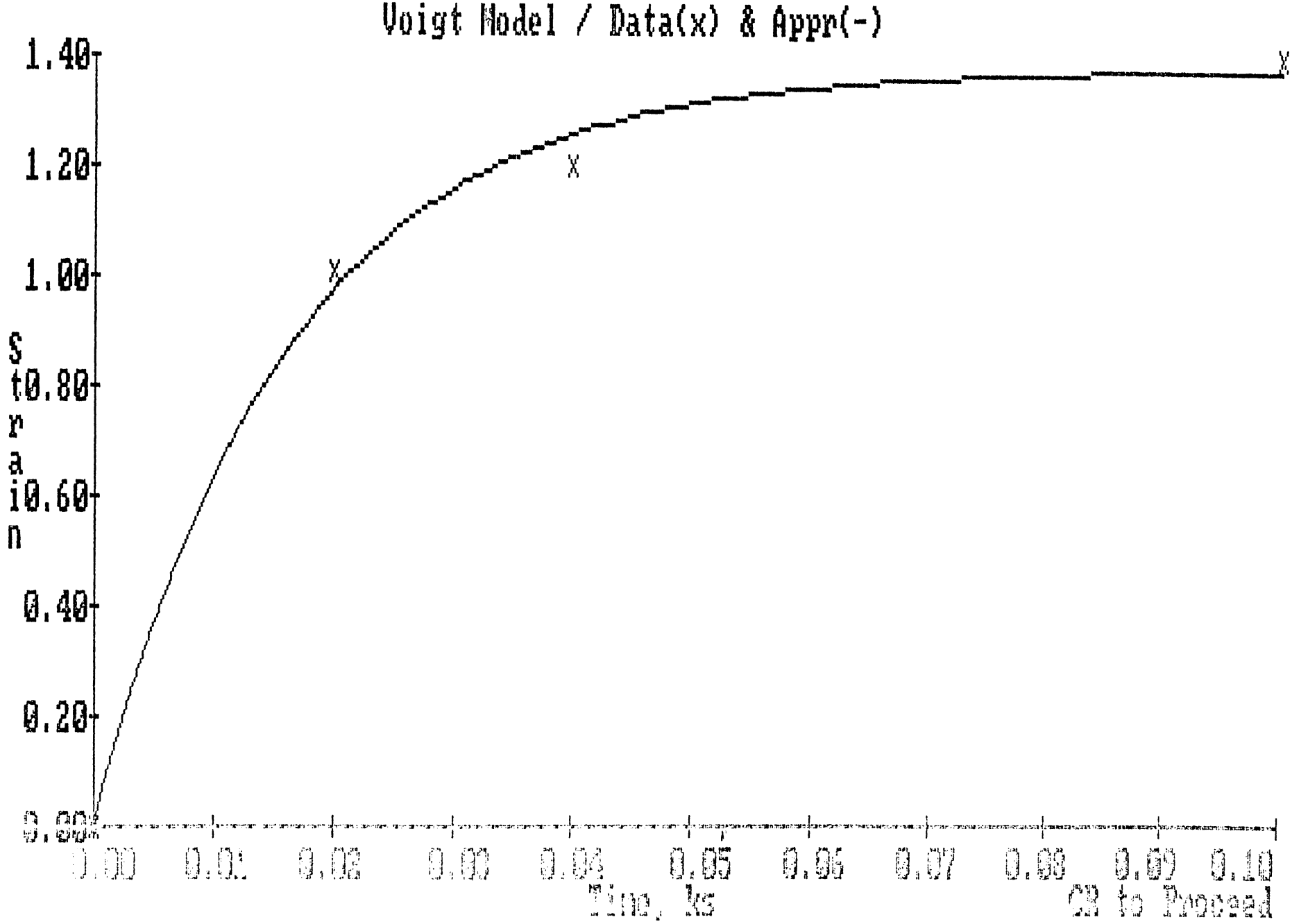
Maxwell Model-Calculated values:	E1,GPa	n1,GPa-s
	181.8182	10.0000


Maxwell Model / Data(x) & Appr(-)
1.40
1.23
1.06
0.89
0.71
0.54
0.37
0.20
Strain
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.10
Time, hr

Voigt Model:	Calculated E1,GPa	Calculated n1,GPa-s
	72.7253	1.193[illegible]

Voigt Model:	Calculated E1,GPa	Specified n1,GPa-s
	72.7248	1.1934

Voigt Model:	Specified E1,GPa	Calculated n1,GPa-s
	72.7248	1.1934


Voigt Model / Data(x) & Appr(-)
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
Strain
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.10
Time, ks
CR to Proceed

Maxwell-Voigt Model:

Calculated E1,GPa	Calculated n1,GPa-s	Calculated E2,GPa	Calculated n2,GPa-s
500.0000	36.3070	108.1282	1.32095

Maxwell-Voigt Model:

Calculated E1,GPa	Specified n1,GPa-s	Calculated E2,GPa	Calculated n2,GPa-s
500.0000	36.3070	108.1281	1.32095

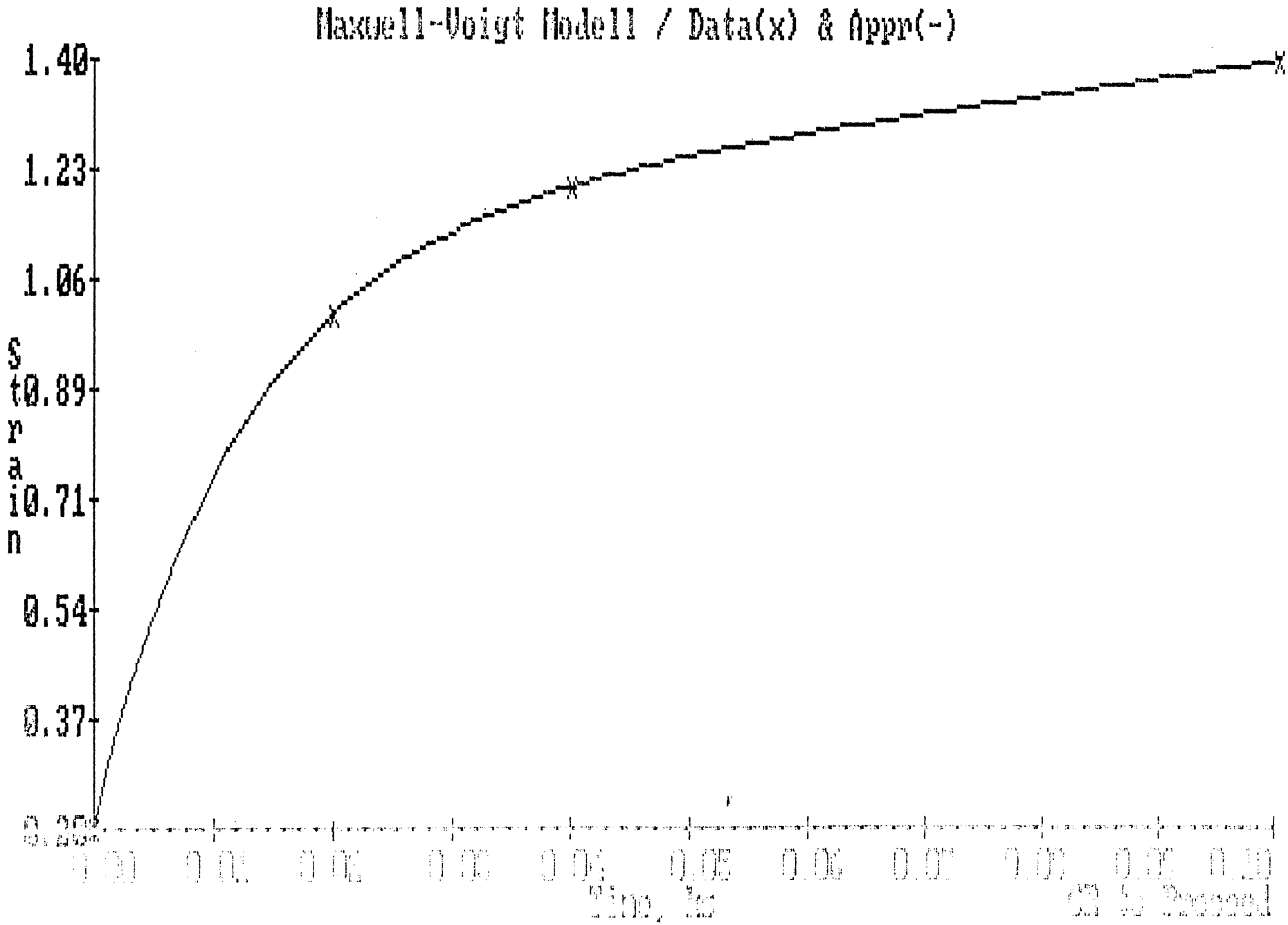
Maxwell-Voigt Model:

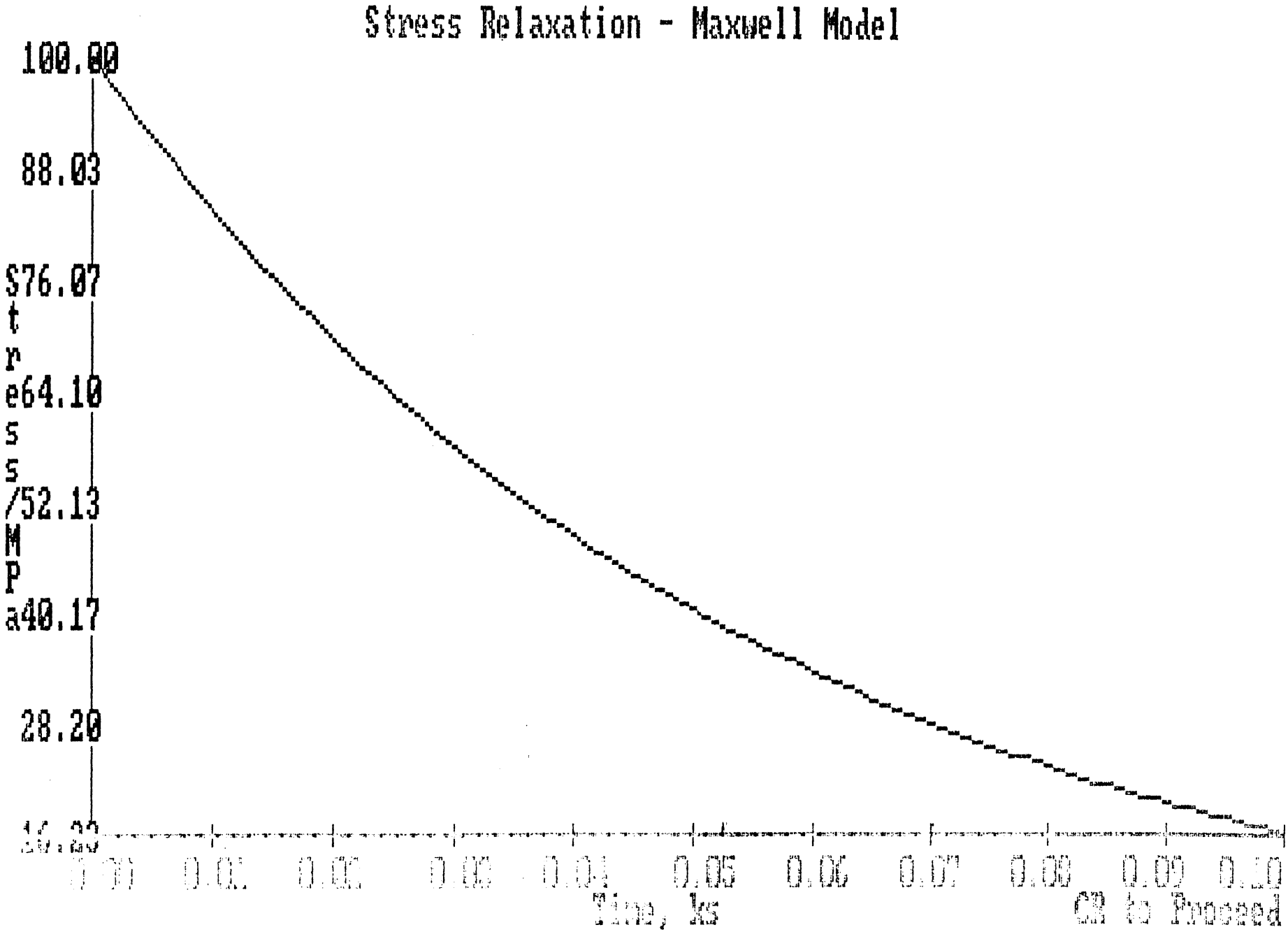
Specified E1,GPa	Specified n1,GPa-s	Calculated E2,GPa	Calculated n2,GPa-s
500.0000	36.3070	108.1281	1.32095

Maxwell-Voigt Model:

Specified E1,GPa	Specified n1,GPa-s	Specified E2,GPa	Calculated n2,GPa-s
500.0000	36.3070	108.1282	1.32095

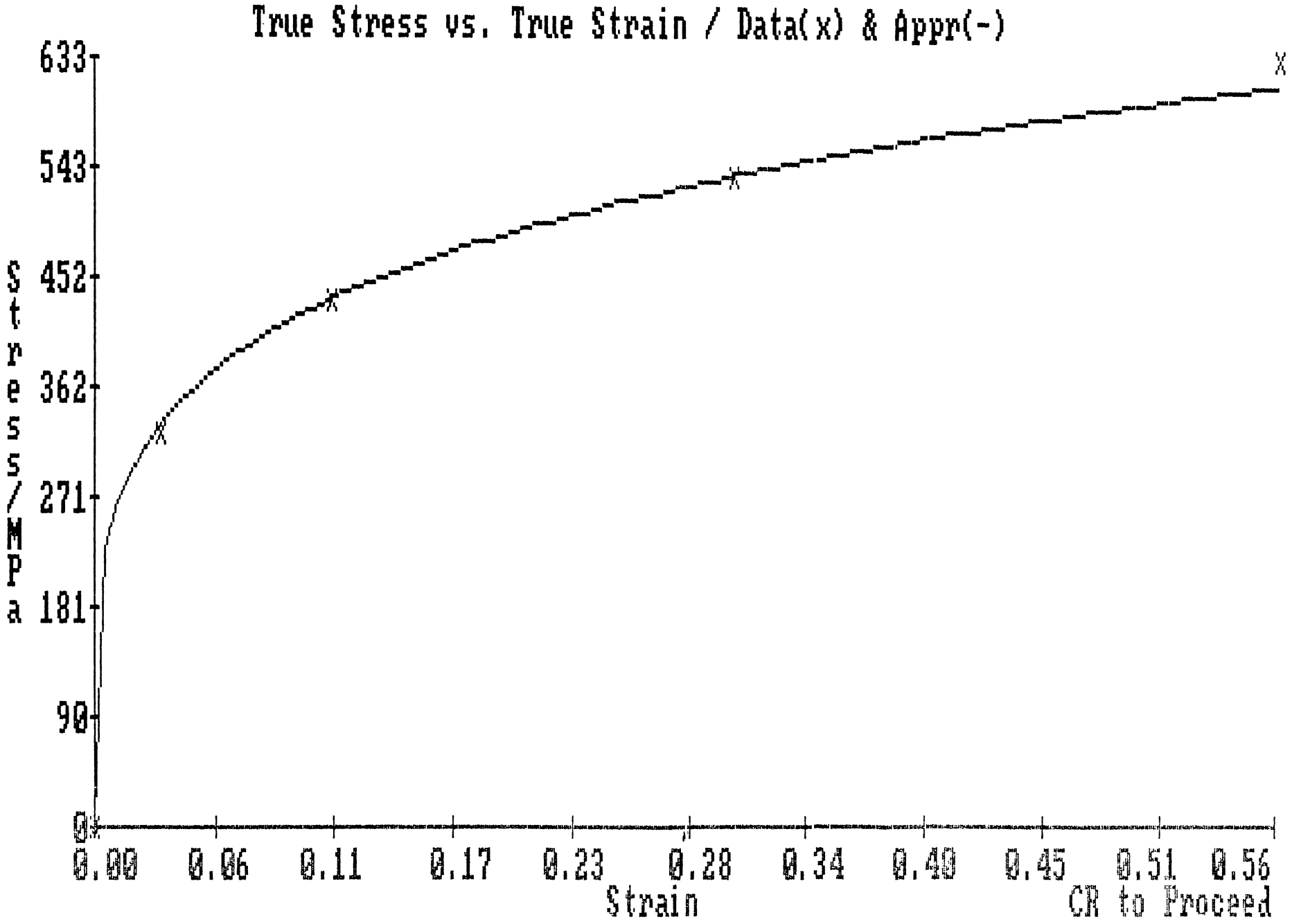
Press Return to Proceed


Maxwell-Voigt Modell / Data(x) & Appr(-)
1.40
1.23
1.06
0.89
0.71
0.54
0.37
Strain
Time, [s]


Stress Relaxation - Maxwell Model
100.00
88.03
S76.07
t
r
e64.10
s
s
/52.13
M
P
a40.17
28.20
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.10
Time, ks
<CR> to Proceed

The current values are:

		Load, kN	Length, mm	Diameter, mm
1.	Reading	0.0000	50.0000	12.8000
2.	Reading	40.0000	52.8000	12.6000
3.	Reading	49.8000	56.9000	12.1000
4.	Reading	50.8000	65.3000	11.0000
5.	Fracture	46.3000	69.1000	9.6500


True Stress vs. True Strain / Data(x) & Appr(-)
633
543
452
362
271
181
90
0
Stress/MPa
0.00
0.06
0.11
0.17
0.23
0.28
0.34
0.40
0.45
0.51
0.56
Strain
CR to Proceed

CREEP AND VISCOELASTICITY

PROGRAM LISTING

```
PROGRAM Viscoelasticity_Models;

CONST
  Undefined=    1.0E-50;
  Title     =    14;
  Regular   =    7;
  Highlite  =    True;
  Normal    =    False;
  CR        =    True;
  NoCR      =    False;
  Maxwell   =    1;
  Voigt     =    2;
  MaxwellVoigt= 3;
  ERR=1.0E-5; KN=0; MAXL=50;
TYPE
  String255 = String[255];
  String80  = String[80];
VAR
  Xarray,Yarray,Y1,B0,XTransY,Times,Stresses,Strains,SpecPar: Array [0..20] of Real
  Partodet,Applications : Array [0..20] of string[80];
  X,XTrans,XTransX,XTransXInv : Array[1..20,1..20] of Real;
  XConv,YConv : Array [1..20] of integer;
  GridCode,Comb,NumofPar,A,C,I,Z,M1,M,Input,DataPoints : Integer;
  Code,Quit : Boolean;
  Xmax,Xmin,Ymax,Ymin,Q1,Det,SingleB0,E1,E2,n1,n2 : Real;
  Indata    : TEXT;
  Appltype,Filename,Ylable : String[20];
  Titlenames,Xlable,Titlename : String[80];
  X1:ARRAY [1..2,1..1000] OF REAL;
  B:ARRAY [1..10] OF REAL;
  G:ARRAY [1..11] OF REAL;
  EE:ARRAY [1..1000] OF REAL;
  FAIL,J,K,N     :INTEGER;
  E,Y,Q     :REAL;


PROCEDURE WriteAt(X,Y : Integer;
                  Highlite,UseCR : Boolean;
                  TheText : string255);
BEGIN
  IF Y < 0 THEN Y := 12;
  IF X < 0 THEN X := (80-Length(TheText)) DIV 2;
  GotoXY(X,Y);
  IF Highlite THEN TextColor(10);
  IF UseCR THEN Writeln(TheText) ELSE Write(TheText);
  TextColor(Regular);
END;

PROCEDURE Proceed;
VAR
  PosY : Integer;
BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Press Return to Proceed');
  PosY:=WhereY;
  Readln;
  GotoXY(1,PosY);   DelLine;
END;

PROCEDURE Header;
BEGIN
  Window(15,6,72,25);
  Clrscr;
  TextColor(Title);
  Writeln('************************************************');
  Writeln('*                                              *');
  Writeln('*                                              *');
```

```
  Writeln('*            UNIVERSITY OF MICHIGAN                *');
  Writeln('*      Department of Mechanical Engineering       *');
  Writeln('*                                                 *');
  Writeln('*                   ME  251                       *');
  Writeln('*                                                 *');
  Writeln('*                  Models for                     *');
  Writeln('*     Rate & Temperature Dependent Behavior       *');
  Writeln('*                                                 *');
  Writeln('*                                                 *');
  Writeln('***************************************************');
  TextColor(Regular);
  Window(1,1,80,25);
END;

PROCEDURE Yes_No_Answer(Var code : Integer);
VAR
  Answer : String[80];
BEGIN
  Readln(Answer);
  IF Length(Answer)=0 THEN Code:=2;
  IF (Answer='yes') OR (Answer='y') THEN Code:=1;
  IF (Answer='no')  OR (Answer='n') THEN Code:=2;
END;


PROCEDURE Introduction_choice(Var W : Integer);
VAR
    Code : Integer;
BEGIN
  REPEAT
    Write('Do you wish to read the introduction? (no):');
    Yes_No_Answer(code);
    CASE Code OF
     1: BEGIN    W:=40; Code:=3;     END;
     2: Code:=3;
    ELSE   BEGIN
     WriteAt(-1,25,Highlite,CR,'Must answer yes or no and hit return');
     GotoXY(15,18); DelLine; GotoXY(1,1); InsLine; GotoXY(1,20); InsLine;
     GotoXY(15,19); Code:=10;     END;
    END;
  UNTIL Code=3;
    Clrscr;
END;


PROCEDURE Introduction;
VAR
  Choice : Integer;
BEGIN
  Introduction_choice(Choice);
  IF Choice=40 THEN   BEGIN
     TextColor(Title);
     Writeln('INTRODUCTION');
     Writeln;
     TextColor(Regular);
     Writeln('This program computes the Modulus of Elasticity, E, and the ');
     Writeln('viscosity coefficients for the Maxwell, the Voigt, and the ');
     Writeln('Maxwell-Voigt models.  ');
     Writeln;
     Writeln('The program also plots the different behaviors associated with');
     Writeln('each model, as well as Stress Relaxation for the Maxwell model');
     Writeln;
     Writeln('The maximum numbers of data points allowed=20');
     Writeln;
     Writeln('User is cautioned to carefully interpret results');
     Proceed;                    END;
```

```
END;

PROCEDURE Initialize_Data_Array;
VAR
  W : Integer;
BEGIN
  FOR W:=0 to 20 DO
  BEGIN
  Applications[W]:='No Change';
  Stresses[W]:=0;
  Strains[W]:=0;
  Times[W]:=0;
  END;
END;

PROCEDURE Stress_Example_Data;
VAR
  W: Integer;
BEGIN
  Appltype:='Stress';
  Applications[1]:='Pos Step';
  FOR W:=2 to 4 DO BEGIN
  Applications[W]:='No Change';
  Stresses[W]:=100;   END;
  DataPoints:=4;
  Stresses[1]:=100;
  Strains[1]:=0.2; Strains[2]:=1.0; Strains[3]:=1.2; Strains[4]:=1.4;
  Times[1]:=0;  Times[2]:=0.02;   Times[3]:=0.04; Times[4]:=0.1;
END;

PROCEDURE Select_Number_Data_Points;
VAR
  Codestr : String[80];
  Error   : Integer;
BEGIN
  REPEAT
  Error:=0;
  WriteAt(-1,-1,Normal,NoCR,'Select Number of Data Points (3):');
  ClrEOL;
  Readln(Codestr);
  IF Length(Codestr)=0 THEN BEGIN
   Error:=0;   DataPoints:=3;    END;
  Val (Codestr,DataPoints,Error);
  IF (DataPoints>20) OR (DataPoints<0) THEN Error:=1;
  IF Error<>0 THEN   BEGIN
  WriteAt(-1,25,Highlite,NoCr,'Selection type must be an integer less than 20');
  Delay(1000);  DelLine;   END;
  UNTIL Error=0;
END;

PROCEDURE Read_Times;
VAR
  Timestr :string[80];
  Error:integer;
BEGIN
  REPEAT
  Readln(Timestr);
  Val(Timestr,Times[i],Error);
  IF Length(Timestr)=0  THEN  Error:=1;
  IF Times[I]<0 THEN Error:=1;
  IF Error <>0 THEN    BEGIN
    WriteAt(-1,25,Highlite,NoCR,'Invalid data.');
    Delay(1000);   DelLine;
      GotoXY(24,13); Clreol;
    END;
  UNTIL Error=0;
```

```
END;

PROCEDURE Read_Stresses;
VAR
  Stressstr :string[80];
  Error:integer;
BEGIN
  REPEAT
  Readln(Stressstr);
  Val(Stressstr,Stresses[i],Error);
  IF Length(Stressstr)=0 THEN Error:=1;
  IF Stresses[I]<0 THEN Error:=1;
  IF Error <>0 THEN    BEGIN
    WriteAt(-1,25,Highlite,NoCR,'Invalid data.');
    Delay(1000);   DelLine;
    GotoXY(42,13); Clreol;       END;
  UNTIL Error=0;
END;

PROCEDURE Read_Strains;
VAR
  Strainsstr :string[80];
  Error:integer;
BEGIN
  REPEAT
  Readln(Strainsstr);
  Val(Strainsstr,Strains[i],Error);
  IF Length(Strainsstr)=0 THEN  Error:=1;
  IF Strains[I]<0 THEN   Error:=1;
  IF Error <>0 THEN    BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Invalid data.');
  Delay(1000);   DelLine;
  GotoXY(60,13); Clreol;  END;
  UNTIL Error=0;
END;

PROCEDURE Select_Application_Type;
VAR
  Applstr : string[80];
  Error : integer;
BEGIN
  REPEAT
  Clrscr;
  Error:=1;
  WriteAt(-1,24,Highlite,NoCR,'Application Types: '+
                'Positive-Step(P)   Negative-Step(N)   ');
  WriteAt(-1,-1,Normal,NoCR,'Select type of Application input (P):');
  Readln(Applstr);
  IF Length(Applstr)=0 THEN
  BEGIN  Error:=0;    Applications[I]:='Pos Step'     END;
  IF (Applstr='Positive') OR (Applstr='P') OR (Applstr='p') THEN BEGIN
    Error:=0;    Applications[I]:='Pos Step';   END;
  IF (Applstr='Negative') OR (Applstr='N') OR (Applstr='n') THEN   BEGIN
    Error:=0;    Applications[I]:='Neg Step';     END;
  UNTIL Error=0;
END;

PROCEDURE Input_Data_Format_per_Reading;
BEGIN
  Appltype:='Stress';
  Clrscr;
  TextColor(Title);
  WriteAt(-1,-1,Normal,CR,'Application     Time,ks          Stress,MPa      '+
                          '  Strain      ');
  GotoXY(6,13);  TextColor(Title);
  Write(I,'. '); TextColor(Regular); Write(Applications[I]);
```

```
  GotoXY(24,13);Read_Times;
  IF Applications[I]<>'No Change'  THEN BEGIN GotoXY(42,13);Read_Stresses; END
  ELSE Stresses[I]:=Stresses[I-1];
  GotoXY(60,13);Read_Strains;
END;

PROCEDURE Input_Data;
BEGIN
  I:=1;
  Select_Application_Type;
  Input_Data_Format_per_Reading;
  FOR I:=2 to DataPoints DO
  BEGIN
    Applications[I]:='No Change';
    Input_Data_Format_per_Reading;
  END;
END;

PROCEDURE Read_Code(VAR Code,Q : Integer);
VAR
  Codestr          : String[80];
  Error,PosX,PosY : Integer;
BEGIN
  PosX:=WhereX;
  PosY:=Wherey;
  REPEAT
  ClrEOL;
  Readln(Codestr);
  IF length(Codestr)=0 THEN BEGIN
  Error:=0;  Q:=45;          END;
  Val(Codestr,Code,Error);
  IF Error<>0  THEN                                BEGIN
    Writeln;
    WriteAt(-1,25,Highlite,NoCR,'Selection type must be an integer');
    Delay(1500);  DelLine;  GotoXY(PosX,PosY);  END;
  UNTIL Error=0;
END;

PROCEDURE Print_Data;
VAR
  POSY,POSYY : Integer;
BEGIN
  IF DataPoints<=20 THEN BEGIN
  Window(1,1,80,25);
  Clrscr;
  PosY:=(21-DataPoints) DIV 2;
  TextColor(Title);
  WriteAt(-1,PosY,Normal,CR,'The current values are:');
  Writeln; TextColor(Title);
  WriteAt(-1,PosY+2,Normal,CR,'Application     Time,ks           Stress,MPa      '+
                          '   Strain     ');
  FOR I:=1 to DataPoints DO
  BEGIN
  PosYY:=PosY+3+I;
  GotoXY(6,PosYY);
  TextColor(Title); Write(I,'. ');TextColor(Regular);Write(Applications[I]:9);
  GotoXY(21,PosYY);
  Write(Times[I]:10:4);
  GotoXY(40,PosYY);
  Write(Stresses[I]:10:4);
  GotoXY(58,PosYY);
  Writeln(Strains[I]:10:4);
  END;
  END   ELSE  Select_Number_Data_Points;
END;
```

```
PROCEDURE Edit_Individual_Readings;
VAR
  P,W,Code,A,PosY:integer;
  Answer:string[80];
BEGIN
  REPEAT
  Print_Data;
  PosY:=WhereY+1;
  Writeln;
  WriteAt(-1,PosY,Normal,NoCR,'Select variable to be modified, '+
                              'by entering corresponding number:');
  ClrEOL;
  Read_code(I,W);
  IF I<1 THEN A:=2 ELSE A:=1;
  IF (I>DataPoints)  OR (W=45) THEN A:=3 ELSE A:=A;
  CASE A OF
    2:  BEGIN
        WriteAt(-1,25,Highlite,NoCR,'Code out of range ');
        Delay(1500);   DelLine;
        END;
    3:  BEGIN
        WriteAt(-1,25,Highlite,NoCR,'Code out of range ');
        Delay(1500);   DelLine;
        END;
  ELSE
    IF I=1 THEN BEGIN
    Select_Application_Type;
    Input_Data_Format_per_Reading;
    END ELSE
    Input_Data_Format_per_Reading;
    IF I<>Datapoints THEN BEGIN
    FOR P:=I+1 to Datapoints  DO
    BEGIN
    IF (Appltype='Stress') AND (Applications[P]='No Change') THEN
    Stresses[P]:=Stresses[P-1];
    END;                    END;
    REPEAT
    Print_Data;
    Code:=4;
    WriteAt(-1,25,Normal,NoCR,'Do you wish to edit another variable? (no):');
    ClrEOL;
    Yes_No_Answer(code);
    CASE Code OF
      1:  Code:=3;
      2:  BEGIN
          Code:=3; W:=45;
          END;
    ELSE                            BEGIN
      Writeln;
      WriteAt(-1,25,Highlite,NoCR,'Must answer yes or no and hit return:');
      Delay(1500);   DelLine;     END;
    END;
    UNTIL Code=3;
    END;
  UNTIL W=45;
END;

PROCEDURE Add_Readings;
VAR
  P,W,Code,A,PosY,J:integer;
  TempApplications : Array[1..20] of String[80];
  TempTimes,TempStrains,TempStresses: Array[1..20] of Real;
BEGIN
  W:=4;
  REPEAT
  Print_Data;
```

```
  PosY:=WhereY+1;
  Writeln;
  WriteAt(-1,PosY,Normal,NoCR,'Select Reading to be added, '+
                              'by entering corresponding number:');
  ClrEOL;
  Read_code(J,W);
  IF (J<=DataPoints) AND (J>0)  THEN  BEGIN
    FOR I:=J to DataPoints DO
    BEGIN
    TempApplications[I]:=Applications[I];
    TempTimes[I]:=Times[I];
    TempStresses[I]:=Stresses[I];
    TempStrains[I]:=Strains[I];
    END;
    FOR I:=J to DataPoints DO
    BEGIN
    Applications[I+1]:=TempApplications[I];
    Times[I+1]:=TempTimes[I];
    Strains[I+1]:=TempStrains[I];
    Stresses[I+1]:=TempStresses[I];
    END;
    DataPoints:=DataPoints+1;        END;
    TempStresses[I]:=Stresses[I];
    TempStrains[I]:=Strains[I];
    Strains[I+1]:=TempStrains[I];
    Stresses[I+1]:=TempStresses[I];
  IF J>DataPoints THEN DataPoints:=J;
  IF (J<1)  OR  (J>20)  OR (W=45)   THEN      BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Negative Numbers or Number greater than 20'+
                              ' not accepted ');
  Delay(1500);   DelLine;     END
  ELSE           BEGIN
  I:=J;
  IF J=1 THEN BEGIN
  Select_Application_Type;
  Input_Data_Format_per_Reading;
  END ELSE BEGIN
  Applications[J]:='No Change'; Input_Data_Format_per_Reading; END;
  IF I<>Datapoints THEN BEGIN
  FOR P:=I+1 to Datapoints  DO
  BEGIN
  IF (Appltype='Stress') AND (Applications[P]='No Change') THEN
  Stresses[P]:=Stresses[P-1];
  END;                   END;
  REPEAT
  Print_Data;
  Code:=4;
  WriteAt(-1,25,Normal,NoCR,'Do you wish to add another variable? (no):');
  ClrEOL;
  Yes_No_Answer(Code);
  CASE Code OF
    1:  Code:=3;
    2:  BEGIN
        Code:=3; W:=45;
        END;
  ELSE                          BEGIN
    Writeln;
    WriteAt(-1,25,Highlite,NoCR,'Must answer yes or no and hit return:');
    Delay(1500);   DelLine;     END;
  END;
  UNTIL Code=3;   END;
  UNTIL W=45;
END;

PROCEDURE Delete_Readings;
VAR
```

```
  P,W,Code,PosY,J:integer;
BEGIN
  W:=4;
  REPEAT
  Print_Data;
  PosY:=WhereY+1;
  Writeln;
  WriteAt(-1,PosY,Normal,NoCR,'Select Reading to be deleted, '+
                              'by entering corresponding number:');
  ClrEOL;
  Read_code(J,W);
  IF (J<=DataPoints) AND (J>=1) OR (W=45) THEN  BEGIN
    DataPoints:=DataPoints-1;
    FOR I:=J to DataPoints DO
    BEGIN
    Applications[I]:=Applications[I+1];
    Times[I]:=Times[I+1];
    Stresses[I]:=Stresses[I+1];
    Strains[I]:=Strains[I+1];
    END;
    FOR P:=J to Datapoints  DO
    BEGIN
    IF (Appltype='Stress') AND (Applications[P]='No Change') THEN
    Stresses[P]:=Stresses[P-1];
    END;
    REPEAT
    Print_Data;
    Code:=4;
    WriteAt(-1,25,Normal,NoCR,'Do you wish to delete another variable? (no):');
    ClrEOL;
    Yes_No_Answer(Code);
    CASE Code OF
      1:  Code:=3;
      2:  BEGIN
          Code:=3; W:=45;
          END;
    ELSE                              BEGIN
      Writeln;
      WriteAt(-1,25,Highlite,NoCR,'Must answer yes or no and hit return:');
      Delay(1500);   DelLine;     END;
    END;
    UNTIL Code=3;            END
  ELSE                               BEGIN
    WriteAt(-1,25,Highlite,NoCR,'Reading not defined ');
    Delay(1500);   DelLine;       END;
  UNTIL W=45;
END;

PROCEDURE Menu_Edit_Data;
VAR
  J,W : Integer;
BEGIN
  REPEAT
  Clrscr;
  Window(29,10,80,25);
  Clrscr;    TextColor(Title);
  Writeln('Edit Data Menu');
  Writeln;    TextColor(14);
  Write('1'); TextColor(Regular); Writeln(' Edit Individual Reading');
  TextColor(14); Write('2'); TextColor(Regular); Writeln(' Add Reading');
  TextColor(14); Write('3'); TextColor(Regular);
  Writeln(' Delete Reading'); TextColor(14);
  Write('4'); TextColor(Regular); Writeln(' Return to Main Menu');
  Writeln;
  Write('Enter selection here:');
  Read_Code(J,W);
```

```
  CASE J OF
    1:  BEGIN
        Clrscr;  Edit_Individual_Readings;
        END;
    2:  BEGIN
        Clrscr;  Add_Readings;
        END;
    3:  BEGIN
        Clrscr;  Delete_Readings;
        END;
    4:  Writeln;
  ELSE                        BEGIN
    Writeln;
    WriteAt(-1,25,Highlite,NoCR,'Code outside of range 1-4');
    Delay(1500);   DelLine;  END;
  END;
  UNTIL J=4;
END;

PROCEDURE Exit;
VAR
    Code    : integer;
    Answer  :String[80];
BEGIN
  REPEAT
  WriteAt(-1,-1,Normal,NoCR,'Exiting program? (yes):');
  ClrEOL;
  Readln(Answer);
  IF Length(Answer)=0 THEN Code:=1;
  IF (Answer='yes') OR (Answer='y') THEN Code:=1;
  IF (Answer='no')  OR (Answer='n') THEN Code:=2;
  CASE Code OF
    1:  BEGIN
        Quit:=true;  Code:=3;
        END;
    2:  Code:=3;
  ELSE                              BEGIN
    WriteAt(-1,25,Highlite,NoCR,'Must answer yes or no and hit return:');
    Delay(1500);   DelLine;         END;
  END;
  UNTIL Code=3;
END;

PROCEDURE Loading_Data_File;
BEGIN
  Clrscr;
  WriteAt(-1,-1,Normal,NoCR,'Name of Input File:');
  ReadLn(Filename);
  Assign(Indata,Filename);
  {$I-} Reset(Indata); {$I+}
  IF IOResult<>0 THEN
      WriteAt(1,25,Highlite,NoCR,'This File cannot be opened.')
  ELSE
    BEGIN
      I:=0;
      Readln(Indata,Appltype);
      IF Appltype='Stress'   THEN  BEGIN
      IF I<20 THEN BEGIN
      WHILE NOT EOF(Indata) DO
      BEGIN
      I:=I+1;
      Readln(Indata,Applications[I]);
      Readln(Indata,Times[I]);
      Readln(Indata,Stresses[I]);
      Readln(Indata,Strains[I]);
      DataPoints:=I;
```

```
      END;          END;
      Close(Indata);  END;
    END;
END;

PROCEDURE Input_Option;
VAR
  Y,W  : Integer;
  Stop : Boolean;
BEGIN
  Stop:=False;
  REPEAT
  Clrscr;   TextColor(Title);
  Writeln('**INPUT OPTION**');    TextColor(Regular);
  Writeln('  Choose an option');  TextColor(Title);
  Write('  1');TextColor(Regular);Writeln(' Input Data Directly');TextColor(14);
  Write('  2');TextColor(Regular);Writeln(' Load Input Data File');TextColor(14);
  Write('  3');TextColor(Regular);Writeln(' Load Example Data');
  Writeln;
  Write('Enter selection here:');
  Read_Code(Y,W);
  CASE Y OF
    1: BEGIN
       Input:=1; Stop:=true;  Window(1,1,80,25);  Clrscr;
       Select_Number_Data_Points;  Input_Data;
       END;
    2: BEGIN
       Input:=2; Stop:=true;  Window(1,1,80,25); Loading_Data_File;
       END;
    3: BEGIN
       Stop:=true; Window(1,1,80,25); Clrscr; Stress_Example_Data; Print_Data;
       proceed;
       END;
  ELSE                          BEGIN
  Writeln;
  WriteAt(-1,25,Highlite,NoCR,'Code outside of range 1-3');
  Delay(1500);    DelLine;    END;
  END;
  UNTIL Stop=true;
END;

PROCEDURE Saving_CurrentData;
BEGIN
  WriteAt(-1,-1,Normal,NoCR,'Name Input File to be saved:');
  Readln(Filename);
  Assign(Indata,Filename);
  Rewrite(Indata);
  Writeln(Indata,Appltype);
  For I:=1 to DataPoints DO
  BEGIN
  Writeln(Indata,Applications[I]);
  Writeln(Indata,Times[I]);
  Writeln(Indata,Stresses[I]);
  Writeln(Indata,Strains[I]);
  END;
  Close(Indata);
END;

FUNCTION Power(mantissa,Exponent :Real): Real;
BEGIN
  Power:=Exp(Ln(Mantissa)*Exponent);
END;

PROCEDURE Nonproportional_Constant_Q;
VAR
  Qstr: String[80];
```

```
  Error: Integer;
BEGIN
  REPEAT
  WriteAt(-1,-1,Normal,NoCr,'Select Constant of Nonproportional Behavior,Q-(1):');
  Readln(Qstr);
  IF Length(Qstr)=0 THEN BEGIN Q1:=1; Error:=0;  END;
  Val(Qstr,Q1,Error);
  IF Q1=0  THEN   Error:=1;
  IF Error<>0  THEN  BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Invalid Q.');
  Delay(1000);  DelLine;
  GotoXY(1,12);  Clreol;  END;
  UNTIL Error=0;
END;

PROCEDURE Select_NumParameters_to_determine;
VAR
  Codestr : String[80];
  Error   : Integer;
BEGIN
  REPEAT
  Error:=0;
  WriteAt(-1,-1,Normal,NoCR,'Select Number of Parameters to determine-(3):');
  ClrEOL;
  Readln(Codestr);
  IF Length(Codestr)=0 THEN BEGIN
   Error:=0;   NumofPar:=3;    END;
  Val (Codestr,NumofPar,Error);
  IF (NumofPar>4) OR (NumofPar<0) THEN Error:=1;
  IF Error<>0 THEN   BEGIN
  WriteAt(-1,25,Highlite,NoCr,'Selection type must be an integer less than 4');
  Delay(1000);  DelLine;   END;
  UNTIL Error=0;
  Clrscr;
END;

PROCEDURE Parameters_to_determine;
VAR
  Partodetstr: String[80];
  W,B,Error: Integer;
  Duplicate: Boolean;
BEGIN
  FOR W:=1 to 20 DO
  BEGIN
  Partodet[W]:='initializing';
  SpecPar[W]:=0;
  END;
  IF C=1 THEN     BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Parameter Selection:   E1    n1');
  TextColor(Title);
  WriteAt(-1,7,Normal,NoCR,'Model: Elastic Modulus E1 in series with viscosity n1')
  IF C=2 THEN     BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Parameter Selection:   E1    n1');
  TextColor(Title);
  WriteAt(-1,7,Normal,NoCR,'Model: Elastic Modulus E1 in parallel with viscosity n1
  IF C=3 THEN     BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Parameter Selection:   E1    n1    E2    n2');
  TextColor(Title);
  WriteAt(-1,7,Normal,NoCR,'Model: E1 in series with'+
                           ' E2 & n2 in parallel, also in series with n1'); END;
  FOR W:=1 to Numofpar DO
  BEGIN
  Error:=1;
  REPEAT
  GotoXY(24,12); Write(W);
  CASE W OF
```

```
    1:Write('st');  2:Write('nd');  3:Write('rd');  ELSE Write('th');  END;
  WriteAt(-1,-1,Normal,NoCr,'Parameter to Determine:');  Clreol;
  Readln(Partodetstr);
  IF (Partodetstr='E1') OR (Partodetstr='e1')  THEN   BEGIN
    Duplicate:=False;
    FOR B:=1 to W DO BEGIN
    IF Partodet[B]='E1'  THEN Duplicate:=True;   END;
    IF Duplicate=True THEN BEGIN
    WriteAt(-1,18,Highlite,NoCR,'E1 has already been chosen');
    Delay(1500);  DelLine; END
    ELSE  BEGIN
    Partodet[W]:='E1';   SpecPar[2]:=Undefined;  Error:=0;   END;
  END;
  IF (Partodetstr='n1') OR (Partodetstr='N1')  THEN   BEGIN
    Duplicate:=False;
    FOR B:=1 to W DO BEGIN
    IF Partodet[B]='n1'  THEN Duplicate:=True;   END;
    IF Duplicate=True THEN BEGIN
    WriteAt(-1,18,Highlite,NoCR,'E1 has already been chosen');
    Delay(1500);  DelLine; END
    ELSE  BEGIN
    Partodet[W]:='n1';   SpecPar[3]:=Undefined;  Error:=0;   END;
  END;
  IF (Partodetstr='E2') OR (Partodetstr='e2') AND (C=3)  THEN   BEGIN
    Duplicate:=False;
    FOR B:=1 to W DO BEGIN
    IF Partodet[B]='E2'  THEN Duplicate:=True;   END;
    IF Duplicate=True THEN BEGIN
    WriteAt(-1,18,Highlite,NoCR,'E1 has already been chosen');
    Delay(1500);  DelLine; END
    ELSE  BEGIN
    Partodet[W]:='E2';   SpecPar[4]:=Undefined;  Error:=0;   END;
  END;
  IF (Partodetstr='n2') OR (Partodetstr='N2') AND (C=3)  THEN   BEGIN
    Duplicate:=False;
    FOR B:=1 to W DO BEGIN
    IF Partodet[B]='n2'  THEN Duplicate:=True;   END;
    IF Duplicate=True THEN BEGIN
    WriteAt(-1,18,Highlite,NoCR,'E1 has already been chosen');
    Delay(1500);  DelLine; END
    ELSE  BEGIN
    Partodet[W]:='n2';   SpecPar[5]:=Undefined;  Error:=0;   END;
  END;
  UNTIL Error=0;
  END;
  Comb:=1;
  FOR W:=2 to 5 DO
  BEGIN
  IF SpecPar[W]=Undefined   THEN  Comb:=Comb*W;
  END;
END;

PROCEDURE Select_SpecPar;
VAR
  W,L : Integer;
  SpecParstr :string[80];
  Error:integer;
BEGIN
  GotoXY(1,25); ClrEOL;
  IF C<>3 THEN L:=3  ELSE  L:=5;
  FOR W:=2 to L DO
  BEGIN
  GotoXY(1,12); ClrEOL;
  IF SpecPar[W]=0 THEN BEGIN
  CASE W OF
  2: BEGIN WriteAt(29,-1,Normal,NoCR,'Specify value of E1 in GPa:'); ClrEOL; END;
```

```
  3: BEGIN WriteAt(27,-1,Normal,NoCR,'Specify value of n1 in GPa-s:'); ClrEOL; END;
  4: BEGIN WriteAt(29,-1,Normal,NoCR,'Specify value of E2 in GPa:'); ClrEOL; END;
  5: BEGIN WriteAt(27,-1,Normal,NoCR,'Specify value of n2 In GPa-s:'); ClrEOL; END;
  END;
  REPEAT
  Readln(SpecParstr);
  IF Length(SpecParstr)=0 THEN BEGIN
  Error:=1;         END;
  Val(SpecParstr,SpecPar[W],Error);
  IF SpecPar[W]<=0 THEN   Error:=1;
  IF Error <>0 THEN    BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Invalid data.');
  Delay(1000);   DelLine;
  GotoXY(56,12); ClrEOL;  END;
  UNTIL Error=0;   END;
  END;
  E1:=SpecPar[2]; n1:=SpecPar[3];  E2:=SpecPar[4];  n2:=SpecPar[5];
END;

PROCEDURE XTranspose;
VAR
  PAR : Integer;
BEGIN
  FOR PAR:=1 to NumofPar DO
  BEGIN
  FOR I:=1 to DataPoints DO
  BEGIN
  XTrans[PAR,I]:=X[I,PAR];
  END;
  END;
END;

PROCEDURE XTransposeX;
VAR
  P,PAR : Integer;
BEGIN
  FOR P:=1 to NumofPar DO
  BEGIN
  FOR PAR:=1 to NumofPar DO
  BEGIN
  XTransX[P,PAR]:=0;
  FOR I:=1 to DataPoints DO
  BEGIN
  XtransX[P,PAR]:=XTransX[P,PAR]+XTrans[P,I]*X[I,PAR];
  END;  END;   END;
END;

PROCEDURE Det_of_XTransX;
VAR
  L,W,P,K,J : Integer;
  NDet,PDet : Array[1..4] of Real;
BEGIN
  IF NumofPar=2 THEN L:=0  ELSE L:=NumofPar-1;
  FOR W:=0 to L DO
 BEGIN
  PDet[W+1]:=1;    NDet[W+1]:=1;
  FOR P:=1 to NumofPar DO
  BEGIN
  K:=P+W;   IF K>NumofPar THEN K:=P-(NumofPar-W);
  PDet[W+1]:=PDet[W+1]*XTransX[P,K];
  J:=NumofPar-(P-1)+W;  IF J>NumofPar THEN  J:=W-(P-1);
  NDet[W+1]:=NDet[W+1]*XTransX[P,J];
  END;
  END;
  Det:=0;
  IF NumofPar=2 THEN L:=1 ELSE L:=NumofPar;
```

```
  FOR W:=1 to L DO
  BEGIN
  Det:=Det+PDet[W]-NDet[W];
  END;
END;

PROCEDURE Inverse_XTransX;
VAR
  Sign,P,PAR,J,J1,W,W1,K,L,M,X,Y,U,Z : Integer;
  Cofactor : Real;
  NDet,PDet : Array[1..4] of Real;
  CofactorMatrix : Array[1..3,1..3] of Real;
BEGIN
  FOR PAR:=1 to NumofPar DO
  BEGIN
  FOR P:=1 to NumofPar DO
  BEGIN
    W1:=1; J1:=0;
    FOR W:=1 to NumofPar DO
    BEGIN
      IF W<>P THEN  BEGIN
      Y:=W;
      FOR J:=1 to NumofPar DO
      BEGIN
      IF J<>PAR THEN                          BEGIN
      X:=J;
      J1:=J1+1;  IF J1>NumofPar-1 THEN BEGIN  W1:=W1+1; J1:=1;  END;
      CofactorMatrix[W1,J1]:=XTransX[X,Y];
                      END;
      END;            END;
      END;
       IF NumofPar<>2   THEN  BEGIN
        IF NumofPar=3 THEN L:=0  ELSE L:=NumofPar-2;
        FOR W:=0 to L DO
        BEGIN
        PDet[W+1]:=1;    NDet[W+1]:=1;
        FOR J:=1 to NumofPar-1 DO
        BEGIN
        K:=J+W;   IF K>NumofPar-1 THEN K:=J-(NumofPar-1-W);
        PDet[W+1]:=PDet[W+1]*CofactorMatrix[J,K];
        M:=NumofPar-1-(J-1)+W;  IF M>NumofPar-1 THEN  M:=W-(J-1);
        NDet[W+1]:=NDet[W+1]*CofactorMatrix[J,M];
        END;
        END;
        Cofactor:=0;
        IF NumofPar=3 THEN L:=1 ELSE L:=NumofPar-1;
        FOR W:=1 to L DO
        BEGIN
        Cofactor:=Cofactor+PDet[W]-NDet[W];
        END;
       END;
    Sign:=1;
    FOR L:=1 to P+PAR DO
    BEGIN
    Sign:=-1*Sign;
    END;
    IF NumofPar<>2 THEN
    XTransXInv[P,PAR]:=Sign*Cofactor/Det   ELSE
    XTransXInv[P,PAR]:=Sign*CofactorMatrix[1,1]/Det;
    END;  END;
END;

PROCEDURE XTransposeY;
VAR
  P : Integer;
BEGIN
```

```
  FOR P:=1 to NumofPar DO
  BEGIN
  XTransY[P]:=0;
  FOR I:=1 to DataPoints DO
  BEGIN
  XTransY[P]:=XTransY[P]+XTrans[P,I]*Y1[I];
  END;
  END;
END;

PROCEDURE B0_XTransXInv_XTransY;
VAR
  P,PAR : Integer;
BEGIN
  FOR P:=1 to NumofPar DO
  BEGIN
  B0[P]:=0;
  FOR PAR:=1 to NumofPar DO
  BEGIN
  B0[P]:=B0[P]+XtransXInv[P,PAR]*XTransY[PAR];
  END;
  END;
END;

PROCEDURE LinearRegression_XY;
BEGIN
  XTranspose; XTransposeX;
  Det_of_XtransX;
  IF Det<>0 THEN BEGIN
  Inverse_XTransX;
  XTransposeY;
  B0_XTransXInv_XTransY;
  END;
END;

PROCEDURE OnePar_XY_LinearRegression;
VAR
  XTXInverse : Real;
BEGIN
  XTranspose;
  XTransposeX;
  IF XTransX[1,1]<>0   THEN    BEGIN
  XTransposeY;
  SingleB0:=1/XTransX[1,1]*XTransY[1];
                               END;
END;

PROCEDURE Proceed_Graph;
BEGIN
  WriteAt(67,25,Highlite,NoCR,'CR to Proceed');
  Readln;
END;

PROCEDURE Title_Lables;
VAR
  Grid : String[80];
  Finish : Boolean;
BEGIN
  Write('Title for graph:');
  Readln(Titlename);
  IF Length(Titlename)=0 THEN Titlename:=Titlenames;
  Write('Lable for the x-axis(Time/ks):');
  Readln(Xlable);
  IF Length(Xlable)=0  THEN Xlable:='Time, ks';
  Write('Lable for the y-axis(Strain):');
  Readln(Ylable);
```

```
  IF (Length(Ylable)=0)  AND (C<>4) THEN Ylable:='Strain';
  IF (Length(Ylable)=0)  AND (C=4) THEN Ylable:='Stress/MPa';
  Repeat
  Write('Grid scale(no)');
  Readln(Grid);
  IF Length(Grid)=0 THEN GridCode:=1;
  IF (Grid='no') or (Grid='n')  THEN GridCode:=1;
  IF (Grid='yes') OR (Grid='y') THEN GridCode:=2;
  Case GridCode of
  1:  Finish:=True;
  2:  Finish:=True;
  ELSE                    BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Must answer yes or no');
  Delay(1500); DelLine;    END;
  END;
  UNTIL Finish=True;
END;

PROCEDURE Print_TitleLables;
VAR
  XlableCent,YlableCent,b : Integer;
  YChar : String[1];
BEGIN
  TextColor(Title);
  WriteAt(-1,1,Normal,NoCR,Titlename);
  XlableCent:=3+((76-Length(Xlable)) DIV 2);
  WriteAt(XlableCent,25,Normal,NoCR,Xlable);
  YlableCent:=2+(20-Length(Ylable)) DIV 2;
  For B:=1 to Length(Ylable) DO
  BEGIN
  YChar:=Copy(Ylable,B,1);
  WriteAt(1,YlableCent+b,Normal,NoCR,YChar);
  END;
END;

PROCEDURE Axis;
BEGIN
Draw(42,11,42,181,1);
Draw(40,179,632,179,1);
END;

PROCEDURE Grid;
VAR
  M,L:Integer;
BEGIN
  L:=11;
  For M:=1 to  7  DO
  BEGIN
  Draw(40,L,632,L,1);
  L:=L+24;
  END;
  L:=101;
  For M:=1 to 10 DO
  BEGIN
  Draw(L,11,L,181,1);
  L:=L+59;
  END;
END;

PROCEDURE Divisions;
VAR
  M,L:Integer;
BEGIN
  L:=11;
  For M:=1 to  10  DO
  BEGIN
```

```
  Draw(40,L,44,L,1);
  L:=L+24;
  END;
  L:=101;
  For M:=1 to 10 DO
  BEGIN
  Draw(L,177,L,181,1);
  L:=L+59;
  END;
END;

PROCEDURE Define_XY_Plotting_Matrices;
BEGIN
  IF C<>4 THEN BEGIN
  FOR I:=1 to DataPoints DO
  BEGIN
  Xarray[I]:=Times[I];
  Yarray[I]:=Strains[I];
  IF (I=1) AND (C=2) AND (Times[I]=0)  THEN  Yarray[I]:=0;
  END;          END
  ELSE BEGIN
  IF Times[DataPoints]<>0 THEN
  Yarray[1]:=Stresses[1]*Exp(-E1*Times[DataPoints]/n1) ELSE Yarray[1]:=0;
  Xarray[1]:=Times[1];
  FOR I:=2 to DataPoints DO
  BEGIN
  Xarray[I]:=Times[I];
  Yarray[I]:=Stresses[I];
  END;   END;
END;

PROCEDURE XY_MaxMin;
VAR
  M,L :Integer;
BEGIN
  Xmin:=Xarray[1]; Ymin:=Yarray[1];
  Xmax:=Xarray[DataPoints]; Ymax:=Yarray[DataPoints];
  FOR M:=1 to DataPoints DO
  BEGIN
  IF Xarray[M]<Xmin  THEN Xmin:=Xarray[M];
  IF Yarray[M]<Ymin  THEN Ymin:=Yarray[M];
  IF Xarray[M]>Xmax  THEN Xmax:=Xarray[M];
  IF Yarray[M]>Ymax  THEN Ymax:=Yarray[M];
  END;
END;

PROCEDURE ScalesXY;
VAR
  M,L:Integer;
  Xscalenumbers,Yscalenumbers:Array[1..11] of REAL;
  XIncrement,YIncrement:Real;
BEGIN
  IF GridCode=2 THEN Grid
  ELSE Divisions;
  XIncrement:=(Xmax-Xmin)/10;
  YIncrement:=(Ymax-Ymin)/7;
  Xscalenumbers[1]:=Xmin;
  For L:=2 to 11 DO
  Xscalenumbers[L]:=Xscalenumbers[L-1]+XIncrement;
  Yscalenumbers[1]:=Ymin;
  FOR L:=2 to 8 DO
  Yscalenumbers[L]:=Yscalenumbers[L-1]+YIncrement;
  GotoXY(5,24);
  Write(Xscalenumbers[1]:4:2);
  GotoXY(12,24);
  Write(Xscalenumbers[2]:4:2);
```

```
  GotoXY(19,24);
  Write(Xscalenumbers[3]:4:2);
  GotoXY(27,24);
  Write(Xscalenumbers[4]:4:2);
  GotoXY(34,24);
  Write(Xscalenumbers[5]:4:2);
  GotoXY(42,24);
  Write(Xscalenumbers[6]:4:2);
  GotoXY(49,24);
  Write(Xscalenumbers[7]:4:2);
  GotoXY(56,24);
  Write(Xscalenumbers[8]:4:2);
  GotoXY(63,24);
  Write(Xscalenumbers[9]:4:2);
  GotoXY(70,24);
  Write(Xscalenumbers[10]:4:2);
  GotoXY(76,24);
  Write(Xscalenumbers[11]:4:2);
  M:=2;
  For L:=1 to 8 DO
  BEGIN
  GotoXY(2,M);
  Write(Yscalenumbers[9-L]:4:2);
  M:=M+3;
  END;
END;

PROCEDURE XY_Conversion;
VAR
  M,L : Integer;
BEGIN
  FOR M:=1 to DataPoints DO
  BEGIN
  XConv[M]:=Trunc(42+(590/(Xmax-Xmin)*(Xarray[M]-Xmin)));
  YConv[M]:=Trunc(179-(168/(Ymax-Ymin)*(Yarray[M]-Ymin)));
  END;
END;

PROCEDURE Print_Exes;
VAR
  M,L : Integer;
BEGIN
  FOR M:=1 To DataPoints DO
  BEGIN
  Draw(XConv[M]-2,YConv[M]+2,XConv[M]+2,YConv[M]-2,1);
  Draw(XConv[M]-2,YConv[M]-2,XConv[M]+2,YConv[M]+2,1);
  END;
END;

PROCEDURE Functions_Appr;
VAR
  M,L : Integer;
  XapprIncr:Real;
  Xappr,Yappr : Array [1..101] of Real;
  XapprConv,YapprConv : Array [1..101] Of Integer;
BEGIN
  Xappr[1]:=Xmin;
  XapprIncr:=(Xmax-Xmin)/100;
  FOR M:=2 to 101 DO
  Xappr[M]:=Xappr[M-1]+XapprIncr;
  FOR M:=1 to 101 DO
  BEGIN
  CASE C OF
  1:  Yappr[M]:=(Stresses[1]/E1)+(Power(Stresses[1]/n1,1/Q1)*Xappr[M]);
  2:  Yappr[M]:=(Stresses[1]/E1)*(1-Exp(-E1*Xappr[M]/n1));
  3:  Yappr[M]:=(Stresses[1]/E1)+(Stresses[1]*Xappr[M]/n1)+(Stresses[1]/E2)*(1-Exp(
```

```
  4:  Yappr[M]:=(Stresses[1])*Exp(-E1*Xappr[M]/n1);
  END;
  XapprConv[M]:=Trunc(42+(590/(Xmax-Xmin)*(Xappr[M]-Xmin)));
  YapprConv[M]:=Trunc(179-(168/(Ymax-Ymin)*(Yappr[M]-Ymin)));
  END;
  FOR M:=2 to 101 DO
  BEGIN
  L:=M-1;
  IF (XapprConv[L]>=42) AND (YapprConv[L]>=11) THEN
  Draw(XapprConv[L],YapprConv[L],XapprConv[M],YapprConv[M],1);
  END;
END;

PROCEDURE Plot_DataPoints_Function;
BEGIN
  ClrScr;
  CASE C OF
  1:  Titlenames:='Maxwell Model / Data(x) & Appr(-)';
  2:  Titlenames:='Voigt Model / Data(x) & Appr(-)';
  3:  Titlenames:='Maxwell-Voigt Modell / Data(x) & Appr(-)';
  4:  Titlenames:='Stress Relaxation - Maxwell Model';
  END;
  Title_Lables;
  Hires;
  Print_TitleLables;
  Axis;
  Define_XY_Plotting_Matrices;
  XY_MaxMin;
  ScalesXY;
  XY_Conversion;
  IF C<>4 THEN Print_Exes;
  Functions_Appr;
  Proceed_Graph;
  TextMode;
END;

PROCEDURE Set_Up;
  BEGIN
    FOR Z:=1 to N DO
    BEGIN
    X1[1,Z]:=Times[Z];
    X1[2,Z]:=Strains[Z]/Stresses[Z];
    END;
    IF C=2 THEN BEGIN
      CASE NumofPar OF
      1: BEGIN
         M:=1;  M1:=2;
         IF Comb=1 THEN  B[1]:=200
         ELSE B[1]:=30;       END;
      2: BEGIN
         M:=2;  M1:=3;
         B[1]:=200;  B[2]:=30;   END;
      END;
    END;
    IF C=3 THEN BEGIN
      CASE Comb OF
      2:  BEGIN
          M:=1;   M1:=2;
          FOR Z:=1 to N DO
          BEGIN
          X[Z,1]:=Stresses[Z];
  Y1[Z]:=Strains[Z]-(Stresses[Z]*Times[Z]/n1)-(Stresses[Z]/E2)*(1-Exp(-E2*Times[Z]/
          END;
          OnePar_XY_LinearRegression;
          B[1]:=1/SingleB0;
          END;
```

```
    3:  BEGIN
        M:=1;   M1:=2;
        FOR Z:=1 to N DO
        BEGIN
        X[Z,1]:=Stresses[Z]*Times[Z];
Y1[Z]:=Strains[Z]-(Stresses[Z]/E1)-(Stresses[Z]/E2)*(1-Exp(-E2*Times[Z]/n2));
        END;
        OnePar_XY_LinearRegression;
        B[1]:=1/SingleB0;
        END;
    4:  BEGIN
        M:=1;   M1:=2;
        B[1]:=200;
        END;
    5:  BEGIN
        M:=1;   M1:=2;
        FOR Z:=1 to N DO
        BEGIN
        X[Z,1]:=-E2*Times[Z];
        Y1[Z]:=Ln(1-(Strains[Z]*E2/Stresses[Z])+(E2/E1)+(E2*Times[Z]/n1));
        END;
        OnePar_XY_LinearRegression;
        B[1]:=1/SingleB0;
        END;
    6:  BEGIN
        M:=2;   M1:=3;
        B[2]:=30;
        B[1]:=200;
        END;
    8:  BEGIN
        M:=2;   M1:=3;
        B[1]:=200;  B[2]:=200;
        END;
   10:  BEGIN
        M:=2;   M1:=3;
        B[1]:=100;  B[2]:=30;  END;
   12:  BEGIN
        M:=2;   M1:=3;
        B[1]:=30;   B[2]:=200;
        END;
   15:  BEGIN
        M:=2;   M1:=3;
        B[1]:=30;  B[2]:=30;  END;
   20:  BEGIN
        M:=2;   M1:=3;
        B[1]:=200;  B[2]:=30;
        END;
   24:  BEGIN
        M:=3;   M1:=4;
        B[1]:=200;  B[2]:=30;  B[3]:=200;
        END;
   30:  BEGIN
        M:=3;   M1:=4;
        B[1]:=200;  B[2]:=30;  B[3]:=30;  END;
   40:  BEGIN
        M:=3;   M1:=4;
        B[1]:=200;   B[2]:=200;   B[3]:=30;
        END;
   60:  BEGIN
        M:=3;   M1:=4;
        B[1]:=30;   B[2]:=200;   B[3]:=30;
        END;
  120:  BEGIN
        M:=4;   M1:=5;
        B[1]:=200;  B[2]:=30;  B[3]:=200;  B[4]:=30;
        END;
```

```
     END;
    END;
  END;

PROCEDURE F;
  VAR
    S:REAL;
  BEGIN
    IF C=2 THEN BEGIN
      CASE NumofPar OF
      1: BEGIN
         IF Comb=2 THEN BEGIN
         S:=(1/B[1])*(1-Exp(-B[1]*X1[1,K]/n1));
         E:=X1[2,K]-S   END
         ELSE   BEGIN
         S:=(1/E1)*(1-Exp(-E1*X1[1,K]/B[1]));
         E:=X1[2,K]-S   END;
         END;
      2: BEGIN
         S:=(1/B[1])*(1-Exp(-B[1]*X1[1,K]/B[2]));
         E:=X1[2,K]-S   END;
      END;
    END;
    IF C=3 THEN BEGIN
      CASE Comb OF
      2: BEGIN
         S:=(1/B[1])+(X1[1,K]/n1)+(1/E2)*(1-Exp(-E2*X1[1,K]/n2));
         E:=X1[2,K]-S   END;
      3: BEGIN
         S:=(1/E1)+(X1[1,K]/B[1])+(1/E2)*(1-Exp(-E2*X1[1,K]/n2));
         E:=X1[2,K]-S   END;
      4: BEGIN
         S:=(1/E1)+(X1[1,K]/n1)+(1/B[1])*(1-Exp(-B[1]*X1[1,K]/n2));
         E:=X1[2,K]-S   END;
      5: BEGIN
         S:=(1/E1)+(X1[1,K]/n1)+(1/E2)*(1-Exp(-E2*X1[1,K]/B[1]));
         E:=X1[2,K]-S   END;
      6: BEGIN
         S:=(1/B[1])+(X1[1,K]/B[2])+(1/E2)*(1-Exp(-E2*X1[1,K]/n2));
         E:=X1[2,K]-S   END;
      8: BEGIN
         S:=(1/B[1])+(X1[1,K]/n1)+(1/B[2])*(1-Exp(-B[2]*X1[1,K]/n2));
         E:=X1[2,K]-S   END;
     10: BEGIN
         S:=(1/B[1])+(X1[1,K]/n1)+(1/E2)*(1-Exp(-E2*X1[1,K]/B[2]));
         E:=X1[2,K]-S   END;
     12: BEGIN
         S:=(1/E1)+(X1[1,K]/B[1])+(1/B[2])*(1-Exp(-B[2]*X1[1,K]/n2));
         E:=X1[2,K]-S   END;
     15: BEGIN
         S:=(1/E1)+(X1[1,K]/B[1])+(1/E2)*(1-Exp(-E2*X1[1,K]/B[2]));
         E:=X1[2,K]-S   END;
     20: BEGIN
         S:=(1/E1)+(X1[1,K]/n1)+(1/B[1])*(1-Exp(-B[1]*X1[1,K]/B[2]));
         E:=X1[2,K]-S   END;
     24: BEGIN
         S:=(1/B[1])+(X1[1,K]/B[2])+(1/B[3])*(1-Exp(-B[3]*X1[1,K]/n2));
         E:=X1[2,K]-S   END;
     30: BEGIN
         S:=(1/B[1])+(X1[1,K]/B[2])+(1/E2)*(1-Exp(-E2*X1[1,K]/B[3]));
         E:=X1[2,K]-S   END;
     40: BEGIN
         S:=(1/B[1])+(X1[1,K]/n1)+(1/B[2])*(1-Exp(-B[2]*X1[1,K]/B[3]));
         E:=X1[2,K]-S   END;
     60: BEGIN
         S:=(1/E1)+(X1[1,K]/B[1])+(1/B[2])*(1-Exp(-B[2]*X1[1,K]/B[3]));
```

```
        E:=X1[2,K]-S   END;
    120: BEGIN
        S:=(1/B[1])+(X1[1,K]/B[2])+(1/B[3])*(1-Exp(-B[3]*X1[1,K]/B[4]));
        E:=X1[2,K]-S   END;
     END;
    END;
  END;

PROCEDURE SUM;
  BEGIN
    Q:=0.0;
    FOR K:=1 TO N DO
      BEGIN
      F;
      EE[K]:=E;
      Q:=Q+E*E
      END;
  END;

PROCEDURE PD;
  BEGIN
    IF C=2 THEN BEGIN
      CASE NumofPar OF
      1: BEGIN
         IF Comb=2 THEN
         G[1]:=(-1/SQR(B[1]))+(1/SQR(B[1])+X1[1,K]/(n1*B[1]))*Exp(-B[1]*X1[1,K]/n1)
         ELSE
         G[1]:=-(X1[1,K]/SQR(B[1]))*Exp(-E1*X1[1,K]/B[1]);
         END;
      2: BEGIN
         G[1]:=(-1/SQR(B[1]))+(1/SQR(B[1])+X1[1,K]/(B[2]*B[1]))*Exp(-B[1]*X1[1,K]/E
         G[2]:=-(X1[1,K]/SQR(B[2]))*Exp(-B[1]*X1[1,K]/B[2]);
         END;
      END;
    END;
    IF C=3 THEN BEGIN
      CASE Comb OF
      2: BEGIN
         G[1]:=-1/SQR(B[1]);
         END;
      3: BEGIN
         G[1]:=-X1[1,K]/SQR(B[1]);
         END;
      4: BEGIN
         G[1]:=(-1/SQR(B[1]))+(1/SQR(B[1])+X1[1,K]/(n2*B[1]))*Exp(-B[1]*X1[1,K]/n2)
         END;
      5: BEGIN
         G[1]:=-(X1[1,K]/SQR(B[1]))*Exp(-E2*X1[1,K]/B[1]);
         END;
      6: BEGIN
         G[1]:=-1/SQR(B[1]);
         G[2]:=-X1[1,K]/SQR(B[2]);
         END;
      8: BEGIN
         G[1]:=-1/SQR(B[1]);
         G[2]:=(-1/SQR(B[2]))+(1/SQR(B[2])+X1[1,K]/(n2*B[2]))*Exp(-B[2]*X1[1,K]/n2)
         END;
     10: BEGIN
         G[1]:=-1/SQR(B[1]);
         G[2]:=-(X1[1,K]/SQR(B[2]))*Exp(-E2*X1[1,K]/B[2]);
         END;
     12: BEGIN
         G[1]:=-X1[1,K]/SQR(B[1]);
         G[2]:=(-1/SQR(B[2]))+(1/SQR(B[2])+X1[1,K]/(n2*B[2]))*Exp(-B[2]*X1[1,K]/n2)
         END;
     15: BEGIN
```

```
        G[1]:=-X1[1,K]/SQR(B[1]);
        G[2]:=-(X1[1,K]/SQR(B[2]))*Exp(-E2*X1[1,K]/B[2]);
        END;
     20: BEGIN
        G[1]:=(-1/SQR(B[1]))+(1/SQR(B[1])+X1[1,K]/(B[2]*B[1]))*Exp(-B[1]*X1[1,K]/E
        G[2]:=-(X1[1,K]/SQR(B[2]))*Exp(-B[1]*X1[1,K]/B[2]);
        END;
     24: BEGIN
        G[1]:=-1/SQR(B[1]);
        G[2]:=-X1[1,K]/SQR(B[2]);
        G[3]:=(-1/SQR(B[3]))+(1/SQR(B[3])+X1[1,K]/(n2*B[3]))*Exp(-B[3]*X1[1,K]/n2)
        END;
     30: BEGIN
        G[1]:=-1/SQR(B[1]);
        G[2]:=-X1[1,K]/SQR(B[2]);
        G[3]:=-(X1[1,K]/SQR(B[3]))*Exp(-E2*X1[1,K]/B[3]);
        END;
     40: BEGIN
        G[1]:=-1/SQR(B[1]);
        G[2]:=(-1/SQR(B[2]))+(1/SQR(B[2])+X1[1,K]/(B[3]*B[2]))*Exp(-B[2]*X1[1,K]/E
        G[3]:=-(X1[1,K]/SQR(B[3]))*Exp(-B[2]*X1[1,K]/B[3]);
        END;
     60: BEGIN
        G[1]:=-X1[1,K]/SQR(B[1]);
        G[2]:=(-1/SQR(B[2]))+(1/SQR(B[2])+X1[1,K]/(B[3]*B[2]))*Exp(-B[2]*X1[1,K]/E
        G[3]:=-(X1[1,K]/SQR(B[3]))*Exp(-B[2]*X1[1,K]/B[3]);
        END;
    120: BEGIN
        G[1]:=-1/SQR(B[1]);
        G[2]:=-X1[1,K]/SQR(B[2]);
        G[3]:=(-1/SQR(B[3]))+(1/SQR(B[3])+X1[1,K]/(B[4]*B[3]))*Exp(-B[3]*X1[1,K]/E
        G[4]:=-(X1[1,K]/SQR(B[4]))*Exp(-B[3]*X1[1,K]/B[4]);
        END;
     END;
    END;
  END;

PROCEDURE GNM;
  LABEL
    1,8,20,21,23,27,28,29,30;
  VAR
    B0:ARRAY [1..5] OF REAL;
    H:ARRAY [1..6] OF REAL;
    A:ARRAY [1..5,1..6] OF REAL;
    A0:ARRAY [1..5,1..6] OF REAL;
    L,I1,I2:INTEGER;
    D,D0,W,Q0: REAL;
  BEGIN
    L:=0; FAIL:=0; D:=0.01;
    SUM;
1:  L:=L+1; Q0:=Q;
    FOR I:=1 TO M DO
      BEGIN
      B0[I]:=B[I] ;
      FOR J:=1 TO M1 DO  A0[I,J]:=0.0 ;
      END;
    FOR K:=1 TO N DO
      BEGIN
      PD;
      G[M1]:=EE[K];
    FOR I:=1 TO M DO
      BEGIN
      Q:=G[I];
    FOR J:=I TO M1 DO A0[I,J]:=A0[I,J]+Q*G[J];
   END;
      END;
```

```
    FOR I:=1 TO M DO
      BEGIN
      Q:=A0[I,I];
      IF Q<=0.0 THEN GOTO 27;
      H[I]:=1.0/SQRT(Q)
      END;
    FOR I:=1 TO M DO
      BEGIN
      Q:=H[I];
      I1:=I+1;
      IF I1<>M1 THEN
       BEGIN
       FOR J:=I1 TO M DO A0[I,J]:=A0[I,J]*Q*H[J]
      END;
    A0[I,M1]:=A0[I,M1]*Q;
      END;
    IF D>=2.0E-7 THEN D:=D/10.0;
    W:=1.0;  D0:=D;
8:  FOR I:=1 TO M DO
      BEGIN
 FOR J:=1 TO M1 DO A[I,J]:=A0[I,J];
      END;
    FOR I:=1 TO M DO   A[I,I]:=D+1.0;
    FOR I:=1 TO M DO
      BEGIN
       I1:=I+1;
      IF (I1<>M1) THEN
       FOR J:=I1 TO M DO A[J,I]:=A[I,J];
      IF A[I,I]=0.0 THEN  GOTO 27;
      Q:=1.0/A[I,I];
     FOR J:=I1 TO M1 DO A[I,J]:=A[I,J]*Q;
     IF  I1<>M1 THEN
       BEGIN
       FOR K:=I1 TO M DO
       BEGIN
       Q:=A[K,I];
     FOR J:=K TO M1 DO A[K,J]:=A[K,J]-Q*A[I,J];
        END;
    END;
      END;
    FOR I2:=1 TO M DO
      BEGIN
   I:=M-I2+1;
    IF (I<>M) THEN
     BEGIN
     I1:=I+1;
     FOR J:=I1 TO M DO A[I,M1]:=A[I,M1]-A[J,M1]*A[I,J];
   END;
     G[I]:=A[I,M1]*H[I];
     B[I]:=B0[I]+G[I]
      END;
    SUM;
    IF D0<D THEN  GOTO 20;
    FOR I:=1 TO M DO
      BEGIN
      IF (ABS(G[I])/(ABS(B[I])+0.001)>ERR) THEN GOTO 20
      END;
    GOTO 30;
20: IF Q>=Q0 THEN GOTO 21;
    IF L < MAXL THEN GOTO 1;
    GOTO 29 ;
21: IF D >= 20.0 THEN GOTO 23;
    D:=D*10.0 ;
    GOTO 8 ;
23: W:=W/4.0 ;
    IF W=0.0 THEN GOTO 28;
```

```
    FOR I:=1 TO M DO  B[I]:=B0[I]+W*G[I];
    SUM;
    IF Q<Q0 THEN  GOTO 1;
     GOTO 23;
27: FAIL:=-1;
28: FAIL:=FAIL-1;
29: FAIL:=FAIL-1;
30: E:=Q;
  END;

PROCEDURE NLSE;
 BEGIN
   N:=DataPoints;
   Set_Up;
   GNM;
   IF C=2 THEN BEGIN
    CASE NumofPar OF
    1: BEGIN
       IF Comb=2 THEN BEGIN
       E1:=B[1];  Clrscr;
       TextColor(Title);
       WriteAt(-1,-1,Normal,NoCR,'  Voigt Model:   Calculated E1,GPa    Specified n
       GotoXY(31,13);Write(E1:10:4);  GotoXY(52,13);  Write(n1:10:4); Proceed;
       Plot_DataPoints_Function; END
       ELSE BEGIN
       n1:=B[1];   Clrscr;
       TextColor(Title);
       WriteAt(-1,-1,Normal,NoCR,'  Voigt Model:   Specified E1,GPa    Calculated n
       GotoXY(31,13);Write(E1:10:4);  GotoXY(52,13);  Write(n1:10:4); Proceed;
       Plot_DataPoints_Function; END;
       END;
    2: BEGIN
       E1:=B[1];    n1:=B[2];    Clrscr;
       TextColor(Title);
       WriteAt(-1,-1,Normal,NoCR,'  Voigt Model:   Calculated E1,GPa    Calculated
       GotoXY(31,13);Write(E1:10:4);  GotoXY(52,13);  Write(n1:10:4); Proceed;
       Plot_DataPoints_Function;
       END;
    END;
   END;
   IF C=3 THEN BEGIN
    CASE Comb OF
    2: BEGIN
       E1:=B[1];  Clrscr;
       TextColor(Title);
       WriteAt(-1,11,Normal,CR,'Maxwell-Voigt Model:');
       WriteAt(-1,-1,Normal,NoCR,'Calculated E1,GPa   Specified n1,GPa-s'+
                         '  Specified E2,GPa   Specified n2,GPa-s');
       GotoXY(4,13);Write(E1:10:4);  GotoXY(26,13);  Write(n1:10:4);
       GotoXY(44,13);Write(E2:10:4);  GotoXY(67,13);  Write(n2:10:5); Proceed;
       Plot_DataPoints_Function;   END;
    3: BEGIN
       n1:=B[1];  Clrscr;
       TextColor(Title);
       WriteAt(-1,11,Normal,CR,'Maxwell-Voigt Model:');
       WriteAt(-1,-1,Normal,NoCR,'Specified E1,GPa   Calculated n1,GPa-s'+
                         '  Specified E2,GPa   Specified n2,GPa-s');
       GotoXY(4,13);Write(E1:10:4);  GotoXY(26,13);  Write(n1:10:4);
       GotoXY(44,13);Write(E2:10:4);  GotoXY(67,13);  Write(n2:10:5); Proceed;
       Plot_DataPoints_Function;   END;
    4: BEGIN
       E2:=B[1];  Clrscr;
       TextColor(Title);
       WriteAt(-1,11,Normal,CR,'Maxwell-Voigt Model:');
       WriteAt(-1,-1,Normal,NoCR,'Specified E1,GPa    Specified n1,GPa-s'+
                         '  Calculated E2,GPa   Specified n2,GPa-s');
```

```
    GotoXY(4,13);Write(E1:10:4);  GotoXY(26,13);  Write(n1:10:4);
    GotoXY(44,13);Write(E2:10:4);  GotoXY(67,13);  Write(n2:10:5); Proceed;
    Plot_DataPoints_Function;   END;
 5: BEGIN
    n2:=B[1];    Clrscr;
    TextColor(Title);
    WriteAt(-1,11,Normal,CR,'Maxwell-Voigt Model:');
    WriteAt(-1,-1,Normal,NoCR,'Specified E1,GPa   Specified n1,GPa-s'+
                      '  Specified E2,GPa   Calculated n2,GPa-s');
    GotoXY(4,13);Write(E1:10:4);  GotoXY(26,13);  Write(n1:10:4);
    GotoXY(44,13);Write(E2:10:4);  GotoXY(67,13);  Write(n2:10:5); Proceed;
    Plot_DataPoints_Function;    END;
 6: BEGIN
    E1:=B[1];  n1:=B[2];
    Clrscr;
    TextColor(Title);
    WriteAt(-1,11,Normal,CR,'Maxwell-Voigt Model:');
    WriteAt(-1,-1,Normal,NoCR,'Calculated E1,GPa   Calculated n1,GPa-s'+
                      '  Specified E2,GPa   Specified n2,GPa-s');
    GotoXY(4,13);Write(E1:10:4);  GotoXY(26,13);  Write(n1:10:4);
    GotoXY(44,13);Write(E2:10:4);  GotoXY(67,13);  Write(n2:10:5); Proceed;
    Plot_DataPoints_Function;    END;
 8: BEGIN
    E1:=B[1];    E2:=B[2];
    Clrscr;
    TextColor(Title);
    WriteAt(-1,11,Normal,CR,'Maxwell-Voigt Model:');
    WriteAt(-1,-1,Normal,NoCR,'Calculated E1,GPa   Specified n1,GPa-s'+
                      '  Calculated E2,GPa   Specified n2,GPa-s');
    GotoXY(4,13);Write(E1:10:4);  GotoXY(26,13);  Write(n1:10:4);
    GotoXY(44,13);Write(E2:10:4);  GotoXY(67,13);  Write(n2:10:5); Proceed;
    Plot_DataPoints_Function;   END;
10: BEGIN
    E1:=B[1]; n2:=B[2];  Clrscr;
    TextColor(Title);
    WriteAt(-1,11,Normal,CR,'Maxwell-Voigt Model:');
    WriteAt(-1,-1,Normal,NoCR,'Calculated E1,GPa   Specified n1,GPa-s'+
                      '  Specified E2,GPa   Calculated n2,GPa-s');
    GotoXY(4,13);Write(E1:10:4);  GotoXY(26,13);  Write(n1:10:4);
    GotoXY(44,13);Write(E2:10:4);  GotoXY(67,13);  Write(n2:10:5); Proceed;
    Plot_DataPoints_Function;   END;
12: BEGIN
    n1:=B[1];  E2:=B[2];  Clrscr;
    TextColor(Title);
    WriteAt(-1,11,Normal,CR,'Maxwell-Voigt Model:');
    WriteAt(-1,-1,Normal,NoCR,'Specified E1,GPa    Calculated n1,GPa-s'+
                       '  Calculated E2,GPa   Specified n2,GPa-s');
    GotoXY(4,13);Write(E1:10:4);  GotoXY(26,13);  Write(n1:10:4);
    GotoXY(44,13);Write(E2:10:4);  GotoXY(67,13);  Write(n2:10:5); Proceed;
    Plot_DataPoints_Function;   END;
15: BEGIN
    n1:=B[1];  n2:=B[2];  Clrscr;
    TextColor(Title);
    WriteAt(-1,11,Normal,CR,'Maxwell-Voigt Model:');
    WriteAt(-1,-1,Normal,NoCR,'Specified E1,GPa   Calculated n1,GPa-s'+
                      '  Specified E2,GPa   Calculated n2,GPa-s');
    GotoXY(4,13);Write(E1:10:4);  GotoXY(26,13);  Write(n1:10:4);
    GotoXY(44,13);Write(E2:10:4);  GotoXY(67,13);  Write(n2:10:5); Proceed;
    Plot_DataPoints_Function;    END;
20: BEGIN
    E2:=B[1];  n2:=B[2];
    Clrscr;
    TextColor(Title);
    WriteAt(-1,11,Normal,CR,'Maxwell-Voigt Model:');
    WriteAt(-1,-1,Normal,NoCR,'Specified E1,GPa   Specified n1,GPa-s'+
                      '  Calculated E2,GPa   Calculated n2,GPa-s');
```

```
      GotoXY(4,13);Write(E1:10:4);  GotoXY(26,13);  Write(n1:10:4);
      GotoXY(44,13);Write(E2:10:4);  GotoXY(67,13);  Write(n2:10:5); Proceed;
      Plot_DataPoints_Function;    END;
   24: BEGIN
      E1:=B[1];    n1:=B[2];    E2:=B[3];
      Clrscr;
      TextColor(Title);
      WriteAt(-1,11,Normal,CR,'Maxwell-Voigt Model:');
      WriteAt(-1,-1,Normal,NoCR,'Calculated E1,GPa   Calculated n1,GPa-s'+
                        '  Calculated E2,GPa   Specified n2,GPa-s');
      GotoXY(4,13);Write(E1:10:4);  GotoXY(26,13);  Write(n1:10:4);
      GotoXY(44,13);Write(E2:10:4);  GotoXY(67,13);  Write(n2:10:5); Proceed;
      Plot_DataPoints_Function;   END;
   30: BEGIN
      E1:=B[1];    n1:=B[2];    n2:=B[3];
      Clrscr;
      TextColor(Title);
      WriteAt(-1,11,Normal,CR,'Maxwell-Voigt Model:');
      WriteAt(-1,-1,Normal,NoCR,'Calculated E1,GPa   Calculated n1,GPa-s'+
                        '  Specified E2,GPa   Calculated n2,GPa-s');
      GotoXY(4,13);Write(E1:10:4);  GotoXY(26,13);  Write(n1:10:4);
      GotoXY(44,13);Write(E2:10:4);  GotoXY(67,13);  Write(n2:10:5); Proceed;
      Plot_DataPoints_Function;   END;
   40: BEGIN
      E1:=B[1];    E2:=B[2];    n2:=B[3];
      Clrscr;
      TextColor(Title);
      WriteAt(-1,11,Normal,CR,'Maxwell-Voigt Model:');
      WriteAt(-1,-1,Normal,NoCR,'Calculated E1,GPa   Specified n1,GPa-s'+
                        '  Calculated E2,GPa   Calculated n2,GPa-s');
      GotoXY(4,13);Write(E1:10:4);  GotoXY(26,13);  Write(n1:10:4);
      GotoXY(44,13);Write(E2:10:4);  GotoXY(67,13);  Write(n2:10:5); Proceed;
      Plot_DataPoints_Function;   END;
   60: BEGIN
      n1:=B[1];    E2:=B[2];    n2:=B[3];
      Clrscr;
      TextColor(Title);
      WriteAt(-1,11,Normal,CR,'Maxwell-Voigt Model:');
      WriteAt(-1,-1,Normal,NoCR,'Specified E1,GPa   Calculated n1,GPa-s'+
                        '  Calculated E2,GPa   Calculated n2,GPa-s');
      GotoXY(4,13);Write(E1:10:4);  GotoXY(26,13);  Write(n1:10:4);
      GotoXY(44,13);Write(E2:10:4);  GotoXY(67,13);  Write(n2:10:5); Proceed;
      Plot_DataPoints_Function;   END;
  120: BEGIN
      E1:=B[1];    n1:=B[2];    E2:=B[3];   n2:=B[4];
      Clrscr;
      TextColor(Title);
      WriteAt(-1,11,Normal,CR,'Maxwell-Voigt Model:');
      WriteAt(-1,-1,Normal,NoCR,'Calculated E1,GPa   Calculated n1,GPa-s'+
                        '  Calculated E2,GPa   Calculated n2,GPa-s');
      GotoXY(4,13);Write(E1:10:4);  GotoXY(26,13);  Write(n1:10:4);
      GotoXY(44,13);Write(E2:10:4);  GotoXY(67,13);  Write(n2:10:5); Proceed;
      Plot_DataPoints_Function;   END;
   END;
  END;
END;



PROCEDURE XY_Maxwell_1Par;
BEGIN
  IF DataPoints>0 THEN BEGIN
  IF Comb=2 THEN BEGIN
  IF Stresses[1]<>0 THEN BEGIN
  FOR I:=1 to DataPoints DO
```

```
  BEGIN
  X[I,1]:=Stresses[I];
  Y1[I]:=Strains[I]-(Power(Stresses[I]/SpecPar[3],1/Q1)*Times[I]);
  END;
  OnePar_XY_LinearRegression;
     E1:=1/SingleB0;
  Clrscr;
  TextColor(Title);
  WriteAt(-1,-1,Normal,NoCR,'Maxwell Model:   Calculated E1,GPa    Specified n1,GPa
  GotoXY(31,13);Write(E1:10:4);  GotoXY(52,13);  Write(n1:10:4); Proceed;
  Plot_DataPoints_Function; END ELSE BEGIN
  WriteAt(-1,25,Highlite,NoCR,'No Appropriate Data To Calculate E1');
  Delay(1000);    DelLine;      END;
                      END
  ELSE                          BEGIN
  IF Stresses[1]<>0 THEN BEGIN
  FOR I:=1 to DataPoints DO
  BEGIN
  X[I,1]:=Power(Stresses[I],1/Q1)*Times[I];
  Y1[I]:=Strains[I]-Stresses[I]/SpecPar[2];
  END;
  OnePar_XY_LinearRegression;
  n1:=1/Power(SingleB0,1/Q1);
  Clrscr;
  TextColor(Title);
  WriteAt(-1,-1,Normal,NoCR,'Maxwell Model:   Specified E1,GPa    Calculated n1,GPa
  GotoXY(30,13);Write(E1:10:4);  GotoXY(53,13);  Write(n1:10:4); Proceed;
  Plot_DataPoints_Function; END ELSE BEGIN
  WriteAt(-1,25,Highlite,NoCR,'No Appropriate Data To Calculate n1');
  Delay(1000);    DelLine;      END;
                                 END;
                        END;
END;

PROCEDURE XY_Maxwell_2Par;
BEGIN
  IF (DataPoints>1) AND (Stresses[1]<>0) THEN BEGIN
  FOR I:=1 to DataPoints DO
  BEGIN
  X[I,1]:=Power(Stresses[I],1/Q1)*Times[I];
  X[I,2]:=Stresses[I];
  Y1[I]:=Strains[I];
  END;
  LinearRegression_XY;
  n1:=Power(1/B0[1],Q1);
  E1:=1/B0[2];
  Clrscr;
  TextColor(Title);
  WriteAt(-1,-1,Normal,NoCR,'Maxwell Model-Calculated values:   E1,GPa        n1,GP
  GotoXY(44,13);Write(E1:10:4);  GotoXY(59,13);  Write(n1:10:4); Proceed;
  Plot_DataPoints_Function;
                        END
  ELSE                          BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Not Enough Available Data To Calculate E1 & n1');
  Delay(1000);    DelLine;      END;
END;

PROCEDURE Stress_Relaxation;
BEGIN
  IF Stresses[1]<>0 THEN BEGIN
  C:=4;
  NumofPar:=2;
  SpecPar[2]:=0;
  SpecPar[3]:=0;
  WriteAt(-1,7,Normal,NoCR,'Maxwell Model: Elastic Modulus E1 in series with viscos
  Select_SpecPar;
```

```
  Plot_DataPoints_Function;
  END ELSE BEGIN
  WriteAt(-1,25,Highlite,NoCR,'No Appropriate Data Available');
  Delay(1000);    DelLine;      END;
END;



PROCEDURE Comp_Submenu12;
VAR
  T,W  : Integer;
  Stop : Boolean;
BEGIN
  Stop:=False;
  REPEAT
  Clrscr;   TextColor(Title);
  Writeln('CHOICE OF PARAMETER DETERMINATION');    TextColor(Regular);
  Writeln('  Choose an option');  TextColor(Title);
  Write('  1');TextColor(Regular);Writeln(' Determine 1 Parameter by specifying the
  TextColor(14);Write('  2');TextColor(Regular);
  Writeln(' Determine both parameters');
  Writeln;
  Write('Enter selection here:');
  Read_Code(T,W);
  CASE T OF
    1: BEGIN
       Stop:=true;  Window(1,1,80,25);  Clrscr;  NumofPar:=1;
       Parameters_to_determine;    Select_SpecPar;
       IF C=1 THEN XY_Maxwell_1Par;
       IF (C=2) AND (Stresses[1]<>0) THEN NLSE;
       IF Stresses[1]=0 THEN BEGIN
       WriteAt(-1,25,Highlite,NoCR,'No Appropriate Data to Calculate Parameters');
       Delay(1000);    DelLine;      END;
       END;
    2: BEGIN
       Stop:=true;  NumofPar:=2;  Window(1,1,80,25);  Clrscr;
       IF C=1 THEN XY_Maxwell_2Par;
       IF (C=2) AND (Stresses[1]<>0) THEN NLSE;
       IF Stresses[1]=0 THEN BEGIN
       WriteAt(-1,25,Highlite,NoCR,'No Appropriate Data to Calculate Parameters');
       Delay(1000);    DelLine;      END;
       END;
  ELSE                          BEGIN
  Writeln;
  WriteAt(-1,25,Highlite,NoCR,'Code outside of range 1-2');
  Delay(1500);    DelLine;    END;
  END;
  UNTIL Stop=true;
END;

PROCEDURE Comp_Submenu3;
VAR
  T,W  : Integer;
  Stop : Boolean;
BEGIN
  Stop:=False;
  REPEAT
  Clrscr;   TextColor(Title);
  Writeln('CHOICE OF PARAMETER DETERMINATION');    TextColor(Regular);
  Writeln('  Choose an option');  TextColor(Title);
  Write('  1');TextColor(Regular);Writeln(' Determine n parameters by specifying th
  TextColor(14);Write('  2');TextColor(Regular);
  Writeln(' Determine Elastic Modulus-1 & viscocity-1 assuming linear response');
  Writeln;
  Write('Enter selection here:');
  Read_Code(T,W);
```

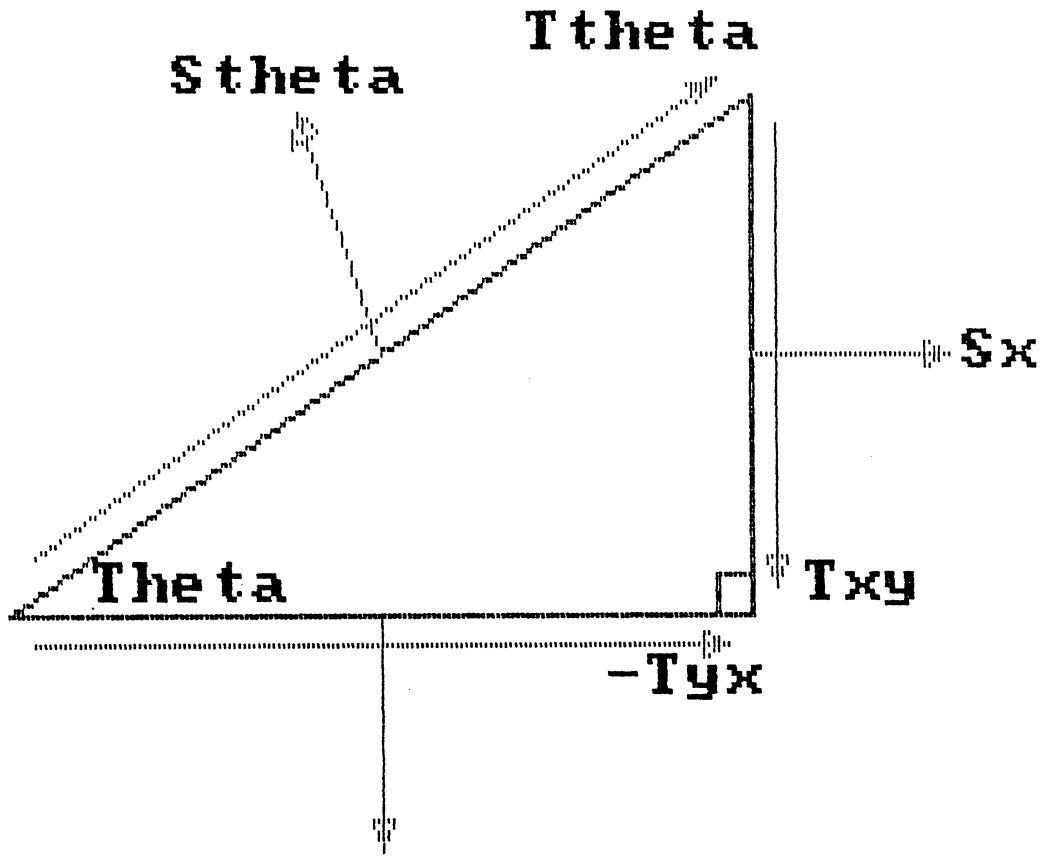
```
  CASE T OF
    1: BEGIN
       IF Stresses[1]<>0  THEN BEGIN
       Stop:=true;  Window(1,1,80,25);  Clrscr;
       Select_NumParameters_to_determine;
       IF (NumofPar<>4) AND (NumofPar<>0) THEN BEGIN
       Parameters_to_determine;  Select_SpecPar; END
       ELSE  Comb:=120;
       IF NumofPar<>0 THEN NLSE;
                                END ELSE BEGIN
       WriteAt(-1,25,Highlite,NoCR,'No Appropriate Data to Calculate Parameters');
       Delay(1000);    DelLine;      END;
       END;
    2: BEGIN
       Stop:=true;  NumofPar:=2; Q1:=1;  Window(1,1,80,25);  Clrscr;
       C:=1;  XY_Maxwell_2Par;
       END;
  ELSE                          BEGIN
  Writeln;
  WriteAt(-1,25,Highlite,NoCR,'Code outside of range 1-2');
  Delay(1500);    DelLine;    END;
  END;
  UNTIL Stop=true;
END;

PROCEDURE Menu_Computations;
VAR
  J,W : Integer;
BEGIN
  REPEAT
  Clrscr;
  Window(29,10,80,25);
  Clrscr;    TextColor(Title);
  Writeln('Computations Menu');
  Writeln;    TextColor(14);
  Write('1'); TextColor(Regular); Writeln(' Maxwell Model (Newtonian or Non-Newtoni
  TextColor(14); Write('2'); TextColor(Regular);
  Writeln(' Voigt Model (Newtonian)');
  TextColor(14); Write('3'); TextColor(Regular);
  Writeln(' Maxwell-Voigt Model Newtonian'); TextColor(14);
  Write('4'); TextColor(Regular);
  Writeln(' Stress Relaxation');
  TextColor(Title); Write('5'); TextColor(Regular);
  Writeln(' Return to Main Menu');
  Writeln;
  Write('Enter selection here:');
  Read_Code(J,W);
  CASE J OF
    1:  BEGIN
        C:=1;
        Clrscr;  Window(1,1,80,25);  Nonproportional_Constant_Q; Clrscr;
        Window(21,10,80,25); Comp_Submenu12;
        END;
    2:  BEGIN
        C:=2;
        Clrscr;  Window(21,10,80,25); Comp_Submenu12;
        END;
    3:  BEGIN
        C:=3;
        Clrscr;  Window(7,10,80,25); Comp_Submenu3;
        END;
    4:  BEGIN
        C:=4;
        Clrscr;  Window(1,1,80,25);
        Stress_Relaxation;
        END;
```

```
    5:  Writeln;
  ELSE                        BEGIN
    Writeln;
    WriteAt(-1,25,Highlite,NoCR,'Code outside of range 1-5');
    Delay(1500);   DelLine;  END;
  END;
  UNTIL J=5;
END;

BEGIN
  LowVideo;
  Clrscr;
  Header;
  Introduction;
  Quit:=False;
  Initialize_Data_Array;

  REPEAT              {start main menu}
  Clrscr;
  Window(29,10,80,25);
  Clrscr; TextColor(Title);
  Writeln('**MAIN MENU**'); TextColor(Regular);
  Writeln('  Choose an Option');
  Writeln;      TextColor(Title);
  Write('  1');TextColor(Regular);Writeln(' List Current Data');TextColor(14);
  Write('  2');TextColor(Regular);Writeln(' Input Data');TextColor(Title);
  Write('  3');TextColor(Regular);Writeln(' Edit Data'); TextColor(Title);
  Write('  4');TextColor(Regular);Writeln(' Save Current Data'); TextColor(14);
  Write('  5');TextColor(Regular);Writeln(' Computations & Plots');TextColor(Title)
  Write('  6');TextColor(Regular);Writeln(' Exit Program');
  Writeln;
  WriteAt(-1,21,Normal,NoCR,'Enter selection here:');
  ClrEOL;
  Z:=0;
  Read_Code(Z,A);
  Writeln;
  CASE Z OF
    0:  Writeln;
    1:  BEGIN
        Print_Data;  Proceed;
        END;
    2:  BEGIN
        Input_Option;
        END;
    3:  Menu_Edit_Data;
    4:  BEGIN
        Window(1,1,80,25);  Clrscr; Saving_CurrentData;
        END;
    5:  Menu_Computations;
    6:  BEGIN
        Window(1,1,80,25);  ClrScr;   Exit;
        END;
  ELSE                        BEGIN
    WriteAt(-1,25,HighLite,NoCR,'Code outside of range');
    Delay(1500); DelLine;   END;
  END;
  UNTIL Quit=true;
END.
```

TWO DIMENSIONAL MOHR'S CIRCLE

EXAMPLE

Positive Sign Convention



CR to Proceed

The current values are:

Normal Stress(x)	Normal Stress(y)	Shear Stress(xy)
Sx	Sy	Txy
25.0000	20.0000	5.0000

The Principal Stresses are:
28.0902 and 16.9098

The Angles between the Principal Planes and the horizontal are:
-31.7175 degrees and 58.2825 degrees

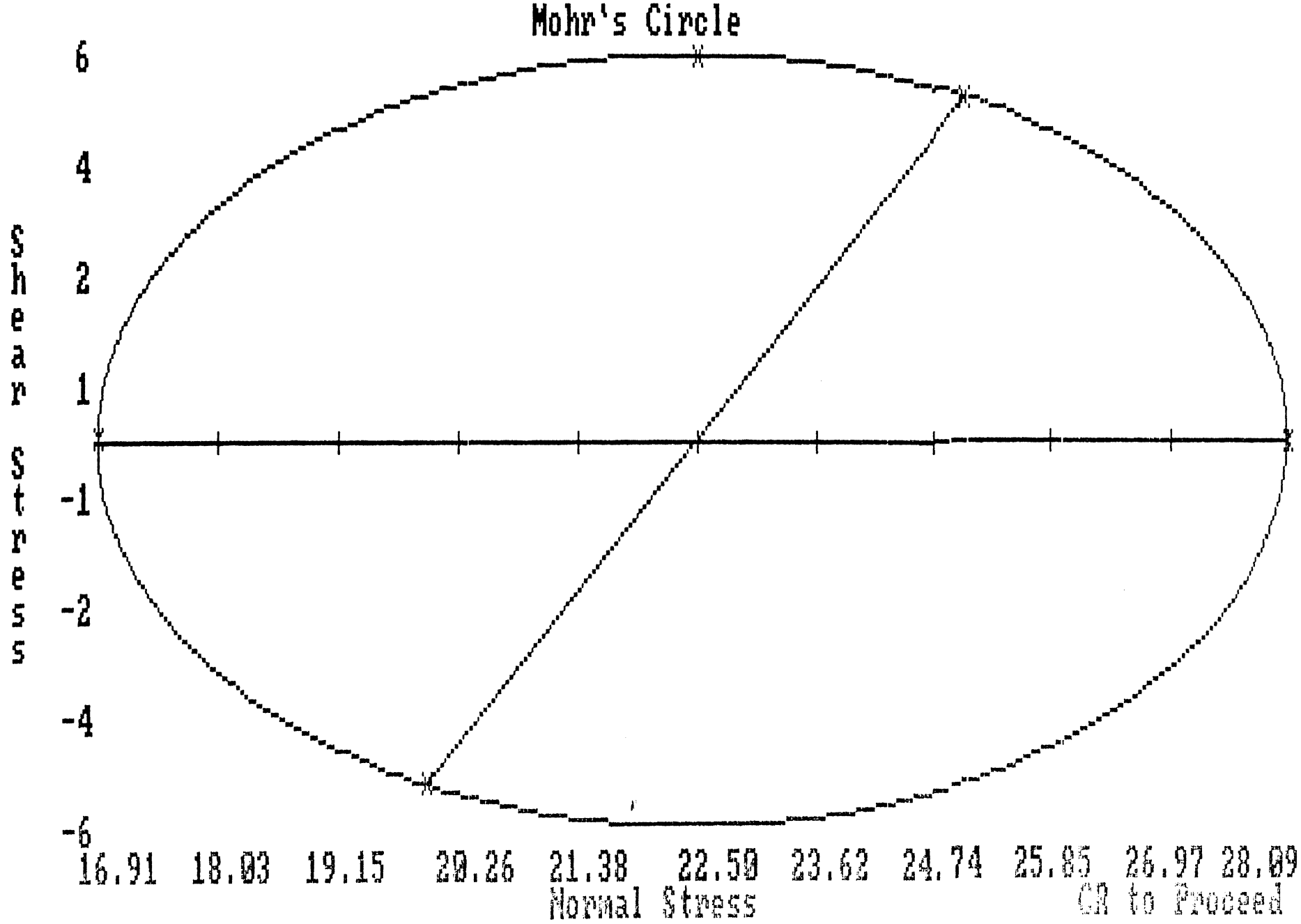
The Maximum Shear Stress is: 5.5902

Angles between the horizontal & the planes where this maximum occurs are:
13.2825 degrees and 103.2825 degrees

The Stresses along the plane defined by angle Theta are:

Theta	Normal Stress Stheta	Shear Stress Ttheta
30.0000	25.5801	-4.6651

```
tle for graph:
ble for the x-axis(Normal Stress):
ble for the y-axis(Shear Stress):
is (yes):
id scale(no):
 ject Sx to Sy(yes):
```


Mohr's Circle
6
4
2
1
-1
-2
-4
-6
Shear Stress
16.91
18.03
19.15
20.26
21.38
22.50
23.62
24.74
25.85
26.97
28.09
Normal Stress
CR to Proceed

TWO DIMENSIONAL MOHR'S CIRCLE

PROGRAM LISTING

```
PROGRAM Mohrs_Circle;


CONST
  Title    = 14;
  Regular  = 7;
  Highlite = True;
  Normal   = False;
  CR       = True;
  NoCR     = False;
TYPE
  String255 = String[255];
  String80  = String[80];
VAR
  {Main Program}
  Z    : Integer;
  Quit : Boolean;

  {Input Data}
  Data,Sx,Sy,Txy                          : Real;

  {Calculations}
  S1,S2,Angle1,Angle2,Angle3,Angle4,Tmax : Real;

  {Graphs}
  Ylable                        : String[20];
  Titlename,Titlenames,Xlable   : String[80];
  SxSyCode,AxisCode,GridCode    : Integer;
  Xmax,Xmin,Ymax,Ymin           : Real;
  Xarray,Yarray                 : Array [0..360] of Real;
  XConv,YConv                   : Array [0..360] of Integer;


{Useful Procedures Throughout the Program}

PROCEDURE WriteAt(X,Y : Integer;
                  Highlite,UseCR : Boolean;
                  TheText : string255);
BEGIN
  IF Y < 0 THEN Y := 12;
  IF X < 0 THEN X := (80-Length(TheText)) DIV 2;
  GotoXY(X,Y);
  IF Highlite THEN TextColor(10);
  IF UseCR THEN Writeln(TheText) ELSE Write(TheText);
  TextColor(Regular);
END;

PROCEDURE Proceed;
VAR
  PosY : Integer;
BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Press Return to Proceed');
  PosY:=WhereY;
  Readln;
  GotoXY(1,PosY);   DelLine;
END;

PROCEDURE Yes_No_Answer(Var code : Integer);
VAR
  Answer : String[80];
BEGIN
  Readln(Answer);
  IF Length(Answer)=0 THEN Code:=2;
  IF (Answer='yes') OR (Answer='y') THEN Code:=1;
  IF (Answer='no')  OR (Answer='n') THEN Code:=2;
END;
```

```
PROCEDURE Read_Code(VAR Code : Integer);
VAR
  Codestr         : String[80];
  Error,PosX,PosY : Integer;
BEGIN
  PosX:=WhereX;
  PosY:=Wherey;
  REPEAT
  ClrEOL;
  Readln(Codestr);
  IF length(Codestr)=0 THEN Error:=0;
  Val(Codestr,Code,Error);
  IF Error<>0  THEN                                BEGIN
    Writeln;
    WriteAt(-1,25,Highlite,NoCR,'Selection type must be an integer');
    Delay(1500);  DelLine;  GotoXY(PosX,PosY);  END;
  UNTIL Error=0;
END;


{Header}

PROCEDURE Header;
BEGIN
  Window(15,6,72,25);
  Clrscr;
  TextColor(Title);
  Writeln('*************************************************');
  Writeln('*                                               *');
  Writeln('*                                               *');
  Writeln('*            UNIVERSITY OF MICHIGAN             *');
  Writeln('*      Department of Mechanical Engineering     *');
  Writeln('*                    ME  251                    *');
  Writeln('*                                               *');
  Writeln('*                Mohr's Circle for              *');
  Writeln('*         a Two Dimensional Stress System       *');
  Writeln('*                                               *');
  Writeln('*                                               *');
  Writeln('*                 by Diana Rincon               *');
  Writeln('*                                               *');
  Writeln('*                                               *');
  Writeln('*************************************************');
  TextColor(Regular);
  Window(1,1,80,25);
END;


{Positive Sign Convention Used}

PROCEDURE Arrows(A,B,Color : Integer;
                 X_dir : Boolean);
VAR
  Inc1,Inc2 : Integer;
BEGIN
  IF X_dir=True THEN BEGIN Inc1:=5; Inc2:=2;
  Draw(A-Inc1,B-Inc2,A,B,Color);
  Draw(A,B,A-Inc1,B+Inc2,Color);
  Draw(A-Inc1,B+Inc2,A-Inc1,B-Inc2,Color); END ELSE
  BEGIN Inc1:=2; Inc2:=5;
  Draw(A-Inc1,B-Inc2,A,B,Color);
  Draw(A,B,A+Inc1,B-Inc2,Color);
  Draw(A+Inc1,B-Inc2,A-Inc1,B-Inc2,Color);
  END;
END;
```

```
PROCEDURE Sign_Convention;
BEGIN
  GraphColorMode;
  WriteAt(8,1,Normal,NoCR,'Positive Sign Convention');
  WriteAt(12,7,Normal,NoCR,'Stheta');
  WriteAt(21,6,Normal,NoCR,'Ttheta');
  WriteAt(10,18,Normal,NoCR,'Theta');
  WriteAt(32,13,Normal,NoCR,'Sx');
  WriteAt(28,18,Normal,NoCR,'Txy');
  WriteAt(23,20,Normal,NoCR,'-Tyx');
  WriteAt(17,25,Normal,NoCR,'Sy');
  Draw(205,55,205,145,2);
  Draw(205,145,55,145,2);
  Draw(205,138,198,138,2);
  Draw(198,138,198,145,2);
  Draw(55,145,205,55,2);
  Draw(210,60,210,140,1);
  Arrows(210,140,1,False);
  Draw(205,100,245,100,1);
  Arrows(245,100,1,True);
  Draw(200,150,60,150,1);
  Arrows(200,150,1,True);
  Draw(130,145,130,185,1);
  Arrows(130,185,1,False);
  Draw(60,135,198,52,1);
  Draw(192,53,198,52,1);
  Draw(198,52,194,56,1);
  Draw(194,56,192,53,1);
  Draw(130,100,113,58,1);
  Draw(112,64,113,58,1);
  Draw(113,58,117,61,1);
  Draw(117,61,112,64,1);
  WriteAt(27,25,Highlite,NoCR,'CR to Proceed');
  Readln;
END;


{Introduction}

PROCEDURE Introduction_choice(Var q : Integer);
VAR
    Code : Integer;
BEGIN
  REPEAT
    Write('Do you wish to read the introduction? (no):');
    Yes_No_Answer(code);
    CASE Code OF
     1: BEGIN    q:=40; Code:=3;     END;
     2: Code:=3;
    ELSE   BEGIN
     WriteAt(-1,25,Highlite,CR,'Must answer yes or no and hit return');
     GotoXY(15,18); DelLine; GotoXY(1,1); InsLine; GotoXY(1,20); InsLine;
     GotoXY(15,19); Code:=10;     END;
    END;
  UNTIL Code=3;
    Clrscr;
END;

PROCEDURE Introduction;
VAR
  Choice : Integer;
BEGIN
  Introduction_choice(Choice);
  IF Choice=40 THEN   BEGIN
     TextColor(Title);
     Writeln('INTRODUCTION');
```

```
    Writeln;
    TextColor(Regular);
    Writeln('This program resolves stresses and strains in directions different');
    Writeln('from those along which stresses and strains are known. ');
    Writeln;
    Writeln('The user, after defining the general two dimensional stress system');
    Writeln('having Sx, Sy, Txy, can ask the program to solve for the following:')
    Writeln('          -the principal stress and its plane directions');
    Writeln('          -the maximum shear stress and its plane directions');
    Writeln('          -the shear and normal stresses for a specified angle Theta'
    Writeln('          -All the stress states, as Theta is varied, through Mohr's'
    Writeln('           Circle. The user is reminded that any angle on the circle'
    Writeln('           corresponds to twice the angle Theta on the real system.')
    Writeln;
    Writeln('The general two dimensional stress system will be graphically shown.'
    Writeln('This will facilitate to describe the sign convention employed, where'
    Writeln('positive shear stresses produce Clockwise system rotation and positiv
    Writeln('angle Theta means CounterClockwise rotation.');
    Writeln;
    Proceed;
    Sign_Convention;          END;
END;


{Initialize Data}

PROCEDURE Initialize_Data;
BEGIN
Sx:=0;  Sy:=0;   Txy:=0;
END;


{Inputing Data}

PROCEDURE Example_Data;
BEGIN
Sx:=25;  Sy:=20;  Txy:=5;
END;

PROCEDURE Read_Data(PosX : Integer);
VAR
  Datastr :string[80];
  Error:integer;
BEGIN
  REPEAT
  Readln(Datastr);
  IF Length(Datastr)=0 THEN  Error:=1;
  Val(Datastr,Data,Error);
  IF Error <>0 THEN    BEGIN
    WriteAt(-1,25,Highlite,NoCR,'Invalid data.');
    Delay(1000);   DelLine;
    GotoXY(PosX,13); Clreol;       END;
  UNTIL Error=0;
END;

PROCEDURE Input_Data;
BEGIN
  Clrscr;
  TextColor(Title);
  WriteAt(24,11,Normal,CR,'Normal Stress(x)   Normal Stress(y)    Shear Stress(xy)'
  TextColor(Title);
  WriteAt(24,12,Normal,CR,'      Sx                Sy                Txy
  GotoXY(6,13); TextColor(Title);  Write('Data');
  GotoXY(24,13); Read_Data(24);  Sx:=Data;
  GotoXY(43,13); Read_Data(43);  Sy:=Data;
  GotoXY(63,13); Read_Data(63);  Txy:=Data;
```

```
END;

PROCEDURE Print_Data;
BEGIN
  Window(1,1,80,25);
  Clrscr;
  TextColor(Title);
  WriteAt(-1,12,Normal,CR,'The current values are:');
  Writeln; TextColor(Title);
  WriteAt(-1,14,Normal,CR,'Normal Stress(x)    Normal Stress(y)    Shear Stress(xy)
  TextColor(Title);
  WriteAt(-1,15,Normal,CR,'      Sx                  Sy                  Txy
  GotoXY(14,16);  Write(Sx:10:4);
  GotoXY(34,16);  Write(Sy:10:4);
  GotoXY(54,16);  Writeln(Txy:10:4);
END;

PROCEDURE Input_Option;
VAR
  Y  : Integer;
  Stop : Boolean;
BEGIN
  Stop:=False;
  REPEAT
  Clrscr;   TextColor(Title);
  Writeln('**INPUT OPTION**');    TextColor(Regular);
  Writeln('  Choose an option');  TextColor(Title);
  Write('  1');TextColor(Regular);Writeln(' Input Data Directly');TextColor(14);
  Write('  2');TextColor(Regular);Writeln(' Load Example Data');
  Writeln;
  Write('Enter selection here:');
  Read_Code(Y);
  CASE Y OF
    1: BEGIN
       Stop:=true;  Window(1,1,80,25);  Clrscr;
       Input_Data;
       END;
    2: BEGIN
       Stop:=true;  Window(1,1,80,25); Example_Data; Print_Data; Proceed;
       END;
  ELSE                          BEGIN
  Writeln;
  WriteAt(-1,25,Highlite,NoCR,'Code outside of range 1-2');
  Delay(1500);    DelLine;    END;
  END;
  UNTIL Stop=true;
END;


{Edit Data}

PROCEDURE Edit_Data;
VAR
  Q,Code,PosY:integer;
  Answer:string[80];
BEGIN
  REPEAT
  Q:=0;
  Print_Data;
  PosY:=WhereY+1;
  Writeln;
  WriteAt(-1,PosY,Normal,NoCR,'Select variable to be modified, '+
                              '(Sx   Sy   Txy):');
  ClrEOL;
  Readln(Answer);
  IF (Answer='Sx') OR (Answer='sx') OR (Answer='SX') THEN
```

```
  BEGIN
  ClrScr;
  WriteAt(-1,13,Normal,NoCR,'Input new value for Sx:');
  ClrEOL;
  Read_Data(WhereX);   Sx:=Data;
  END;
  IF (Answer='Sy') OR (Answer='sy') OR (Answer='SY') THEN
  BEGIN
  ClrScr;
  WriteAt(-1,13,Normal,NoCR,'Input new value for Sy:');
  ClrEOL;
  Read_Data(WhereX);   Sy:=Data;
  END;
  IF (Answer='Txy') OR (Answer='txy') OR (Answer='TXY') THEN
  BEGIN
  ClrScr;
  WriteAt(-1,13,Normal,NoCR,'Input new value for Txy:');
  ClrEOL;
  Read_Data(WhereX);   Txy:=Data;
  END;
    REPEAT
    Print_Data;
    Code:=4;
    WriteAt(-1,25,Normal,NoCR,'Do you wish to edit another variable? (no):');
    ClrEOL;
    Yes_No_Answer(code);
    CASE Code OF
      1:  Code:=3;
      2:  BEGIN
          Code:=3; Q:=45;
          END;
    ELSE                          BEGIN
      Writeln;
      WriteAt(-1,25,Highlite,NoCR,'Must answer yes or no and hit return:');
      Delay(1500);   DelLine;     END;
    END;
    UNTIL Code=3;
  UNTIL Q=45;
END;


{Calculations}

PROCEDURE Stresses_on_Inclined_Plane;
VAR
  Theta,STheta,TTheta : Real;
BEGIN
{Ask User to define the plane by angle(Theta) between plane and horizontal}
  Clrscr;
  TextColor(Title);
  WriteAt(-1,13,Normal,CR,'Angle(Theta) between plane and horizontal:');
  GotoXY(61,13); Read_Data(61);  Theta:=Data;
{Calculate Normal Stress(STheta) & Shear Stress(TTheta) along defined plane}
  STheta:=1/2*(Sy+Sx)+1/2*(Sy-Sx)*COS(PI/180*2*Theta)+Txy*SIN(PI/180*2*Theta);
  TTheta:=1/2*(Sy-Sx)*SIN(PI/180*2*Theta)-Txy*COS(PI/180*2*Theta);
{Print Results on Screen}
  ClrScr;
  TextColor(Title);
  WriteAt(-1,-1,Normal,CR,'The Stresses along the plane defined by angle Theta are:
  TextColor(Title);
  WriteAt(-1,14,Normal,CR,'   Theta           Normal Stress         Shear Stress
  TextColor(Title);
  WriteAt(-1,15,Normal,CR,'                        Stheta               Ttheta
  GotoXY(11,16);  Write(Theta:10:4);
  GotoXY(31,16);  Write(STheta:10:4);
  GotoXY(52,16);  Write(TTheta:10:4);
```

```
  Proceed;
END;

PROCEDURE Calculate_Principal_Stresses_and_Planes;
BEGIN
  IF Sy<>Sx THEN       BEGIN
  Angle1:=180/(2*PI)*ARCTAN(2*Txy/(Sy-Sx));
  Angle2:=Angle1+90;   END;
  S1:=1/2*(Sy+Sx)+1/2*SQRT(SQR(Sy-Sx)+4*SQR(Txy));
  S2:=1/2*(Sy+Sx)-1/2*SQRT(SQR(Sy-Sx)+4*SQR(Txy));
END;

PROCEDURE Printing_Principal_Stresses_and_Planes;
BEGIN
  TextColor(Title);
  WriteAt(-1,-1,Normal,NoCr,'The Principal Stresses are:');
  GotoXY(23,13);  Write(S1:10:4);
  GotoXY(34,13);  Write('and ',S2:10:4);
  TextColor(Title);
  WriteAt(-1,15,Normal,NoCr,'The Angles between the Principal Planes and the horizc
  GotoXY(13,16);  IF Sy<>Sx THEN Write(Angle1:10:4,' degrees');
  GotoXY(34,16);  IF Sy<>Sx THEN Write('and ',Angle2:10:4,' degrees');
  Proceed;
END;

PROCEDURE Calculate_Maximum_Shear_Stress_and_Planes;
BEGIN
  IF Sy<>Sx THEN      BEGIN
  Angle3:=180/(PI*2)*ARCTAN(2*Txy/(Sy-Sx))+45;
  Angle4:=Angle3+90;  END;
  Tmax:=1/2*SQRT(SQR(Sy-Sx)+4*SQR(Txy));
END;

PROCEDURE Printing_Maximum_Shear_Stress_and_Planes;
BEGIN
  Clrscr;
  TextColor(Title);
  WriteAt(-1,-1,Normal,NoCr,'The Maximum Shear Stress is:');
  GotoXY(54,12);  Write(Tmax:10:4);
  TextColor(Title);
  WriteAt(-1,14,Normal,NoCr,'The Angles between the horizontal & the planes where t
                            ' occurs are:');
  GotoXY(18,15);  IF Sy<>Sx THEN Write(Angle3:10:4,' degrees');
  GotoXY(39,15);  IF Sy<>Sx THEN Write('and ',Angle4:10:4,' degrees');
  Proceed;
END;


{Mohr's Circle}

PROCEDURE Proceed_Graph;
BEGIN
  WriteAt(67,25,Highlite,NoCR,'CR to Proceed');
  Readln;
END;

PROCEDURE Title_Lables;
VAR
  Grid,Axis,SxSy : String[80];
  Finish         : Boolean;
BEGIN
  Write('Title for graph:');
  Readln(Titlename);
  IF Length(Titlename)=0 THEN Titlename:=Titlenames;
  Write('Lable for the x-axis(Normal Stress):');
  Readln(Xlable);
```

```
  IF Length(Xlable)=0 THEN Xlable:='Normal Stress';
  Write('Lable for the y-axis(Shear Stress):');
  Readln(Ylable);
  IF Length(Ylable)=0 THEN Ylable:='Shear Stress';
  Repeat
  Write('Axis (yes):');
  Readln(Axis);
  IF Length(Axis)=0 THEN AxisCode:=2;
  IF (Grid='no') or (Grid='n')  THEN GridCode:=1;
  IF (Grid='yes') OR (Grid='y') THEN GridCode:=2;
  Case AxisCode of
  1:  Finish:=True;
  2:  Finish:=True;
  ELSE                    BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Must answer yes or no');
  Delay(1500); DelLine;    END;
  END;
  UNTIL Finish=True;
  Finish:=False;
  Repeat
  Write('Grid scale(no):');
  Readln(Grid);
  IF Length(Grid)=0 THEN GridCode:=1;
  IF (Grid='no') or (Grid='n')  THEN GridCode:=1;
  IF (Grid='yes') OR (Grid='y') THEN GridCode:=2;
  Case GridCode of
  1:  Finish:=True;
  2:  Finish:=True;
  ELSE                    BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Must answer yes or no');
  Delay(1500); DelLine;    END;
  END;
  UNTIL Finish=True;
  Finish:=False;
  Repeat
  Write('Connect Sx to Sy(yes):');
  Readln(SxSy);
  IF Length(SxSy)=0 THEN SxSyCode:=2;
  IF (SxSy='no') or (SxSy='n')  THEN SxSyCode:=1;
  IF (SxSy='yes') OR (SxSy='y') THEN SxSyCode:=2;
  Case SxSyCode of
  1:  Finish:=True;
  2:  Finish:=True;
  ELSE                    BEGIN
  WriteAt(-1,25,Highlite,NoCR,'Must answer yes or no');
  Delay(1500); DelLine;    END;
  END;
  UNTIL Finish=True;
END;

PROCEDURE Print_TitleLables;
VAR
  XlableCent,YlableCent,B : Integer;
  YChar : String[1];
BEGIN
  TextColor(Title);
  WriteAt(-1,1,Normal,NoCR,Titlename);
  XlableCent:=3+((76-Length(Xlable)) DIV 2);
  WriteAt(XlableCent,25,Normal,NoCR,Xlable);
  YlableCent:=2+(20-Length(Ylable)) DIV 2;
  For B:=1 to Length(Ylable) DO
  BEGIN
  YChar:=Copy(Ylable,B,1);
  WriteAt(1,YlableCent+b,Normal,NoCR,YChar);
  END;
END;
```

```
PROCEDURE Grid;
VAR
  M,L,K,N : Integer;
BEGIN
  L:=11;
  For M:=1 to  8  DO
  BEGIN
  FOR K:=40 TO 69 DO
  BEGIN
  N:=K+5;
  Draw(K,L,N,L,1);
  K:=N+14;
  END;
  L:=L+24;
  END;
  L:=42;
  For M:=1 to 11 DO
  BEGIN
  FOR K:=11 TO 29 DO
  BEGIN
  N:=K+2;
  Draw(L,K,L,N,1);
  K:=N+6;
  END;
  L:=L+59;
  END;
END;

PROCEDURE Divisions;
VAR
  M,L,Xaxis : Integer;
BEGIN
  L:=11;
  Xaxis:=Trunc(42+(590/(Xmax-Xmin)*(0-Xmin)));
  For M:=1 to  10  DO
  BEGIN
  Draw(Xaxis-2,L,Xaxis+2,L,1);
  L:=L+24;
  END;
  L:=42;
  For M:=1 to 11 DO
  BEGIN
  Draw(L,93,L,97,1);
  L:=L+59;
  END;
END;

PROCEDURE XY_MaxMin;
BEGIN
  Ymin:=-1*Tmax; Xmin:=S2;
  Ymax:=Tmax; Xmax:=S1;
END;

PROCEDURE Axis;
VAR
  Xaxis : Integer;
BEGIN
  IF AxisCode=2 THEN BEGIN
  Xaxis:=Trunc(42+(590/(Xmax-Xmin)*(0-Xmin)));
  IF (Xmin<0)  AND  (Xmax>0)  THEN  Draw(Xaxis,11,Xaxis,179,1);
  Draw(40,95,632,95,1);  END;
END;

PROCEDURE ScalesXY;
```

```
VAR
  M,L:Integer;
  Xscalenumbers,Yscalenumbers:Array[1..11] of REAL;
  XIncrement,YIncrement:Real;
BEGIN
  IF GridCode=2 THEN Grid
  ELSE Divisions;
  XIncrement:=(Xmax-Xmin)/10;
  YIncrement:=(Ymax-Ymin)/7;
  Xscalenumbers[1]:=Xmin;
  For L:=2 to 11 DO
  Xscalenumbers[L]:=Xscalenumbers[L-1]+XIncrement;
  Yscalenumbers[1]:=Ymin;
  FOR L:=2 to 8 DO
  Yscalenumbers[L]:=Yscalenumbers[L-1]+YIncrement;
  GotoXY(5,24);
  Write(Xscalenumbers[1]:4:2);
  GotoXY(12,24);
  Write(Xscalenumbers[2]:4:2);
  GotoXY(19,24);
  Write(Xscalenumbers[3]:4:2);
  GotoXY(27,24);
  Write(Xscalenumbers[4]:4:2);
  GotoXY(34,24);
  Write(Xscalenumbers[5]:4:2);
  GotoXY(42,24);
  Write(Xscalenumbers[6]:4:2);
  GotoXY(49,24);
  Write(Xscalenumbers[7]:4:2);
  GotoXY(56,24);
  Write(Xscalenumbers[8]:4:2);
  GotoXY(63,24);
  Write(Xscalenumbers[9]:4:2);
  GotoXY(70,24);
  Write(Xscalenumbers[10]:4:2);
  GotoXY(76,24);
  Write(Xscalenumbers[11]:4:2);
  M:=2;
  For L:=1 to 8 DO
  BEGIN
  GotoXY(2,M);
  Write(Yscalenumbers[9-L]:4:0);
  M:=M+3;
  END;
END;

PROCEDURE XY_Conversion;
VAR
  L : Integer;
  M : Real;
BEGIN
  FOR L:=0 to 360 DO
  BEGIN
  M:=L*PI/180;
  Xarray[L]:=1/2*(S1+S2)+1/2*(S1-S2)*COS(M);
  Yarray[L]:=1/2*(S1-S2)*SIN(M);
  XConv[L]:=Trunc(42+(590/(Xmax-Xmin)*(Xarray[L]-Xmin)));
  YConv[L]:=Trunc(179-(168/(Ymax-Ymin)*(Yarray[L]-Ymin)));
  END;
END;

PROCEDURE Print_Exes;
VAR
  M,L : Integer;
BEGIN
  M:=0;
```

```
  Draw(XConv[M]-2,YConv[M]+2,XConv[M]+2,YConv[M]-2,1);
  Draw(XConv[M]-2,YConv[M]-2,XConv[M]+2,YConv[M]+2,1);
  M:=90;
  Draw(XConv[M]-2,YConv[M]+2,XConv[M]+2,YConv[M]-2,1);
  Draw(XConv[M]-2,YConv[M]-2,XConv[M]+2,YConv[M]+2,1);
  M:=180;
  Draw(XConv[M]-2,YConv[M]+2,XConv[M]+2,YConv[M]-2,1);
  Draw(XConv[M]-2,YConv[M]-2,XConv[M]+2,YConv[M]+2,1);
  IF Angle1>0 THEN M:=Trunc(360-Angle1*2) ELSE M:=Trunc(-Angle1*2);
  Draw(XConv[M]-2,YConv[M]+2,XConv[M]+2,YConv[M]-2,1);
  Draw(XConv[M]-2,YConv[M]-2,XConv[M]+2,YConv[M]+2,1);
  IF Angle2>0 THEN L:=Trunc(360-Angle2*2) ELSE L:=Trunc(-Angle2*2);
  Draw(XConv[L]-2,YConv[L]+2,XConv[L]+2,YConv[L]-2,1);
  Draw(XConv[L]-2,YConv[L]-2,XConv[L]+2,YConv[L]+2,1);
  IF SxSyCode=2 THEN Draw(XConv[L],YConv[L],XConv[M],YConv[M],1);
END;

PROCEDURE Trace;
VAR
  M,L : Integer;
BEGIN
  FOR M:=1 to 360 DO
  BEGIN
  L:=M-1;
  IF XConv[L]>=42 THEN
  Draw(XConv[L],YConv[L],XConv[M],YConv[M],1);
  END;
END;

PROCEDURE Plot_Circle;
BEGIN
  ClrScr;
  Titlenames:='Mohr's Circle';
  Title_Lables;
  Hires;
  Print_TitleLables;
  XY_MaxMin;
  IF Xmax<>Xmin THEN BEGIN
  Axis;
  ScalesXY;
  XY_Conversion;
  Print_Exes;
  Trace;              END
  ELSE
  WriteAt(-1,-1,Normal,NoCR,'Mohr's Circle is a point');
  Proceed_Graph;
  Textmode;
END;

PROCEDURE Menu_Computations;
VAR
  J : Integer;
BEGIN
  REPEAT
  Clrscr;
  Window(29,10,80,25);
  Clrscr;    TextColor(Title);
  Writeln('Computations Menu');
  Writeln;    TextColor(14);
  Write('1'); TextColor(Regular); Writeln(' Stresses on an Inclined Plane');
  TextColor(14); Write('2'); TextColor(Regular);
  Writeln(' Principal Stress & Planes'); TextColor(14);
  Write('3'); TextColor(Regular);
  Writeln(' Maximum Shear Stress'); TextColor(Title);
  Write('4'); TextColor(Regular); Writeln(' Mohr's Circle');
  TextColor(Title); Write('5'); TextColor(Regular);
```

```
  Writeln(' Numbers 2-4 Above');
  TextColor(Title); Write('6'); TextColor(Regular);
  Writeln(' Return to Main Menu');
  Writeln;
  Write('Enter selection here:');
  J:=0;
  Read_Code(J);
  CASE J OF
    0:  Writeln;
    1:  BEGIN
        Clrscr; Window(1,1,80,25);
        Stresses_on_Inclined_Plane;
        END;
    2:  BEGIN
        Clrscr;  Window(1,1,80,25);
        Calculate_Principal_Stresses_and_Planes;
        Printing_Principal_Stresses_and_Planes;
        END;
    3:  BEGIN
        Clrscr;  Window(1,1,80,25);
        Calculate_Maximum_Shear_Stress_and_Planes;
        Printing_Maximum_Shear_Stress_and_Planes;
        END;
    4:  BEGIN
        Clrscr;  Window(1,1,80,25);
        Calculate_Principal_Stresses_and_Planes;
        Calculate_Maximum_Shear_Stress_and_Planes;
        Plot_Circle;
        END;
    5:  BEGIN
        Clrscr;  Window(1,1,80,25);
        Calculate_Principal_Stresses_and_Planes;
        Calculate_Maximum_Shear_Stress_and_Planes;
        Printing_Principal_Stresses_and_Planes;
        Printing_Maximum_Shear_Stress_and_Planes;
        Plot_Circle;
        END;
    6:  Writeln;
  ELSE                          BEGIN
    Writeln;
    WriteAt(-1,25,Highlite,NoCR,'Code outside of range 1-6');
    Delay(1500);   DelLine;  END;
  END;
  UNTIL J=6;
END;


{Exit}

PROCEDURE Exit;
VAR
    Code    : integer;
    Answer  :String[80];
BEGIN
  REPEAT
  WriteAt(-1,-1,Normal,NoCR,'Exiting program? (yes):');
  ClrEOL;
  Readln(Answer);
  IF Length(Answer)=0 THEN Code:=1;
  IF (Answer='yes') OR (Answer='y') THEN Code:=1;
  IF (Answer='no')  OR (Answer='n') THEN Code:=2;
  CASE Code OF
    1:  BEGIN
        Quit:=true;  Code:=3;
        END;
    2:  Code:=3;
```

```
  ELSE                          BEGIN
    WriteAt(-1,25,Highlite,NoCR,'Must answer yes or no and hit return:');
    Delay(1500);   DelLine;     END;
  END;
  UNTIL Code=3;
END;


{Main Program}

BEGIN
  LowVideo;
  Clrscr;
  Header;
  Introduction;
  Quit:=False;
  Initialize_Data;

  REPEAT              {start main menu}
  Clrscr;
  Window(29,10,80,25);
  Clrscr; TextColor(Title);
  Writeln('**MAIN MENU**'); TextColor(Regular);
  Writeln('  Choose an Option');
  Writeln;      TextColor(Title);
  Write('  1');TextColor(Regular);Writeln(' Positive Sign Convention Used');
  TextColor(14);
  Write('  2');TextColor(Regular);Writeln(' List Current Data');TextColor(14);
  Write('  3');TextColor(Regular);Writeln(' Input Data');TextColor(Title);
  Write('  4');TextColor(Regular);Writeln(' Edit Data'); TextColor(Title);
  Write('  5');TextColor(Regular);Writeln(' Computations');TextColor(Title);
  Write('  6');TextColor(Regular);Writeln(' Exit Program');
  Writeln;
  WriteAt(-1,21,Normal,NoCR,'Enter selection here:');
  ClrEOL;
  Z:=0;
  Read_Code(Z);
  Writeln;
  CASE Z OF
    0:  Writeln;
    1:  BEGIN
        Window(1,1,80,25); Clrscr;
        Sign_Convention;
        END;
    2:  BEGIN
        Print_Data;  Proceed;
        END;
    3:  BEGIN
        Input_Option;
        END;
    4:  Edit_Data;
    5:  Menu_Computations;
    6:  BEGIN
        Window(1,1,80,25);  ClrScr;   Exit;
        END;
  ELSE                        BEGIN
    WriteAt(-1,25,HighLite,NoCR,'Code outside of range');
    Delay(1500); DelLine;   END;
  END;
  UNTIL Quit=true;
END.
```