

Scilab Textbook Companion for
Materials Science And Engineering: An
Introduction
by W. D. Callister¹

Created by
Akhil Mahawar
B.Tech
Chemical Engineering
Institute of technology, Banaras Hindu University
College Teacher
R.s. Singh
Cross-Checked by

July 31, 2019

¹Funded by a grant from the National Mission on Education through ICT, <http://spoken-tutorial.org/NMEICT-Intro>. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website <http://scilab.in>

Book Description

Title: Materials Science And Engineering: An Introduction

Author: W. D. Callister

Publisher: John Wiley & Sons Inc., USA

Edition: 7

Year: 2007

ISBN: 978-0471736967

Scilab numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

AP Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

Contents

List of Scilab Codes	4
3 The Structure of Crystalline Solids	5
4 Imperfections in Solids	13
5 Diffusion	17
6 Mechanical Properties of Metals	23
7 Dislocations and Strengthening Mechanisms	32
8 Failure	37
9 Phase Diagrams	41
10 Phase Transformations in Metals	47
12 Structures and Properties of Ceramics	50
14 Polymer Structures	55
16 Composites	59
17 Corrosion and Degradation of Materials	63

18 Electrical Properties	67
19 Thermal Properties	73
20 Magnetic Properties	76
21 Optical Properties	80

List of Scilab Codes

Exa 3.1	Determination of FCC Unit Cell Volume . . .	5
Exa 3.2	Computation of the Atomic Packing Factor for FCC	6
Exa 3.3	Theoretical Density Computation for Copper	6
Exa 3.5	Specification of Point Coordinates	7
Exa 3.6	Determination of Directional Indices	8
Exa 3.8	Determination of Directional Indices for a Hexagonal Unit Cell	9
Exa 3.9	Determination of Planar Indices	9
Exa 3.11	Determination of Miller Bravais Indices . . .	10
Exa 3.12.a	Interplanar Spacing	11
Exa 3.12.b	Diffraction Angle Computations	11
Exa 4.1	Number of Vacancies Computation	13
Exa 4.3	Composition Conversion From Weight Percent to Atom Percent	14
Exa 4.4.a	Computations of ASTM Grain Size Number	15
Exa 4.4.b	Number of Grains Per Unit Area	15
Exa 5.1	Diffusion Flux Computation	17
Exa 5.2	Nonsteady State Diffusion Time Computation	18
Exa 5.3	Nonsteady State Diffusion Time Computation II	18
Exa 5.4	Diffusion Coefficient Determination	19
Exa 5.5	Diffusion Coefficient Activation Energy . . .	20
Exa 5.6	Diffusion Temperature Time Heat Treatment Specification	21
Exa 6.1	Elongation Computation	23
Exa 6.2	Computation of Load to Produce Specified Diameter Change	24

Exa 6.3.a	Modulus of elasticity	24
Exa 6.3.c	Maximum load	25
Exa 6.3.d	Change in length	26
Exa 6.4.a	Ductility Computations	27
Exa 6.4.b	True Stress At Fracture Computations	27
Exa 6.5	Calculation of Strain Hardening Exponent	28
Exa 6.6.a	Average Computations	29
Exa 6.6.b	Standard Deviation Computations	29
Exa 6.7	Specification of Support Post Diameter	30
Exa 7.1.a	Resolved Shear Stress	32
Exa 7.1.b	Stress to Initiate Yielding Computations	33
Exa 7.2	Tensile Strength and Ductility Determinations	34
Exa 7.3	Description of Diameter Reduction Procedure	35
Exa 8.1	Maximum Flaw Length Computation	37
Exa 8.2	Rupture Lifetime Prediction	38
Exa 8.3	Estimating theoretical fracture strength	38
Exa 8.4	Computation of critical shear stress	39
Exa 8.5	Determining maximum allowable surface crack length	39
Exa 9.1	Lever Rule Derivation	41
Exa 9.2	Determination of Phases Present	41
Exa 9.3.a	Relative Phase Amount Determinations	42
Exa 9.3.b	Mass and Volume Fractions	43
Exa 9.4.a	fractions of total ferrite and cementite phases	44
Exa 9.4.b	Fractions of the proeutectoid ferrite and pearlite	45
Exa 9.4.c	Fraction of eutectoid ferrite	45
Exa 10.1.a	Computation of Critical Nucleus Radius	47
Exa 10.1.b	Activation Free Energy Computation	48
Exa 10.2	Determination the rate of recrystallization	49
Exa 12.1	Computation of Minimum Cation to Anion Radius Ratio	50
Exa 12.2	Ceramic Crystal Structure Prediction	51
Exa 12.3	Theoretical Density Calculation	51
Exa 12.4	Computation of the Number of Schottky Defects	52
Exa 12.5	Determination of Possible Point Defect Types	53
Exa 14.2.a	Computations of the Density	55
Exa 14.2.b	Computation of Percent Crystallinity	56

Exa 14.3.a	Computations of Diffusion Flux of Carbon Dioxide	57
Exa 14.3.b	Computation of Beverage Shelf Life	57
Exa 16.1.a	Modulus of elasticity	59
Exa 16.1.b	Magnitude of the load	59
Exa 16.1.c	Strain determination	60
Exa 16.2	Elastic Modulus Determination	61
Exa 17.1	Determination of Electrochemical Cell Characteristics	63
Exa 17.2.a	Rate of Oxidation Computation	64
Exa 17.2.b	Corrosion potential determination	65
Exa 17.3	Temperature Computation	65
Exa 18.1	Computation of the Room Temperature Intrinsic Carrier Concentration	67
Exa 18.2	Electrical Conductivity Determination for Intrinsic Silicon	68
Exa 18.3.b	Room Temperature for Extrinsic Silicon	68
Exa 18.3.c	Elevated Temperature Electrical Conductivity Calculations	69
Exa 18.4	Hall Voltage Computation	70
Exa 18.6	Acceptor Impurity Doping in Silicon	71
Exa 19.1	Thermal Stress Created Upon Heating	73
Exa 19.2	Final temperature Calculation	74
Exa 19.3	Computation of specific heats for Al and Fe	74
Exa 20.1.a	Saturation Magnetization	76
Exa 20.1.b	Flux Density Computations for Nickel	77
Exa 20.2	Saturation Magnetization Determination	78
Exa 20.3	Design of a Mixed Ferrite Magnetic Material	78
Exa 21.1	Computation of the Absorption Coefficient for Glass	80
Exa 21.2	Velocity of light in diamond	81
Exa 21.3	Suitable material required	81

Chapter 3

The Structure of Crystalline Solids

Scilab code Exa 3.1 Determination of FCC Unit Cell Volume

```
1 //Determination of FCC Unit Cell Volume
2
3 clear;
4 clc;
5
6 printf("\tExample 3.1\n");
7
8 //For FCC a=2*R*sqrt(2)
9 R=poly([0], 'R');
10
11 //Edge Length
12 a=2*R*sqrt(2);
13
14 //Volume determination
15 V=a^3;
16
17 disp(V, "Volume is");
18
19 //End
```

Scilab code Exa 3.2 Computation of the Atomic Packing Factor for FCC

```
1 //Computation of the Atomic Packing Factor for FCC
2
3 clear;
4 clc;
5
6 printf("\tExample 3.2\n");
7
8 //for FCC no. of atoms are 4
9 n=4;
10
11 //For FCC a=2*R*sqrt(2)
12 R=poly([0], 'R');
13
14 //Edge Length
15 a=2*R*sqrt(2);
16
17 //Volume determination of cube
18 Vc=a^3;
19
20 //Volume of sphere
21 Vs=n*4*%pi*R^3/3;
22
23 //Atomic packing Fraction
24 APF=Vs/Vc;
25
26 disp(APF, "Atomic packing fraction is");
27
28 //End
```

Scilab code Exa 3.3 Theoretical Density Computation for Copper

```

1 //Theoretical Density Computation for Copper
2
3 clear;
4 clc;
5
6 printf("\tExample 3.3\n");
7
8 R=1.28D-08;           //Atomic radius in cm
9 A_Cu=63.5;           //Atomic wt of copper
10 n=4;                 //For FCC
11
12 Na=6.023D23;        //Avogadro no.
13
14 a=2*R*sqrt(2);
15 Vc=a^3;
16
17 den=n*A_Cu/(Vc*Na);
18
19 printf("\nDensity is %.2f g/cm^3\n",den);
20
21 //End

```

Scilab code Exa 3.5 Specification of Point Coordinates

```

1 //Specification of Point Coordinates
2
3 clear;
4 clc;
5
6 printf("\tExample 3.5\n");
7
8 disp("Point coordinates for given positions of BCC
9     cell are");
10 A=['Point_no ', 'x_axis ', 'y_axis ', 'z_axis ', '

```

```

        Coordinates ' ;
11 ' 1', '0', '0', '0', '000' ;
12 ' 2', '1', '0', '0', '100' ;
13 ' 3', '1', '1', '0', '110' ;
14 ' 4', '0', '1', '0', '010' ;
15 ' 5', '1/2', '1/2', '1/2', '1/2 1/2 1/2' ;
16 ' 6', '0', '0', '1', '001' ;
17 ' 7', '1', '0', '1', '101' ;
18 ' 8', '1', '1', '1', '111' ;
19 ' 9', '0', '1', '1', '011' ] ;
20
21 disp(A);
22
23 //End

```

Scilab code Exa 3.6 Determination of Directional Indices

```

1 //Determination of Directional Indices
2
3 clear;
4 clc;
5
6 printf("\tExample 3.6\n");
7
8 printf("\nThe procedure is summarised as :\n");
9
10 A=[ '      ', 'x', 'y', 'z' ;
11 'Projections', 'a/2', 'b', '0c' ;
12 'In terms of a,b,c', '1/2', '1', '0' ;
13 'Reduction', '1', '2', '0' ] ;
14
15 disp(A);
16
17 printf("\nEnclosure      [1 2 0]\n");
18

```

19 //End

Scilab code Exa 3.8 Determination of Directional Indices for a Hexagonal Unit Cell

```
1 //Determination of Directional Indices for a
  Hexagonal Unit Cell
2
3 clear;
4 clc;
5
6 printf("\tExample 3.8\n");
7
8 //From the construction shown in the book
9 du=1;
10 dv=1;
11 dw=1;
12
13 //The above indices are for parallelepiped
14 //To convert it for hexagonal system
15 u=(2*du-dv)/3;
16 v=(2*dv-du)/3;
17 t=-(u+v);
18 w=dw;
19
20 x=[u v t w]*3;
21 disp(x,"The indices for the given directions are");
22
23 //End
```

Scilab code Exa 3.9 Determination of Planar Indices

```
1 //Determination of Planar (Miller) Indices
2
```

```

3 clear;
4 clc;
5
6 printf("\tExample 3.9\n");
7
8 x=[0 -1 2];
9
10 disp(x,"The intercept for the given plane is");
11
12 //End

```

Scilab code Exa 3.11 Determination of Miller Bravais Indices

```

1 //Determination of Miller Bravais Indices for a
   Plane Within a Hexagonal Unit Cell
2
3 clear;
4 clc;
5
6 printf("\tExample 3.11\n");
7
8 //This plane intersects the a1 axis at a distance a
   from the origin of the a1-a2-a3-z coordinate axes
   system.
9
10 //Furthermore, its intersections with the a2 and z
   axes are -a and c.
11
12 //Therefore, in terms of the lattice parameters,
   these intersections are 1, -1 and 1.
13
14 h=1;
15 k=-1;
16 l=1;
17 i=-(h+k);

```

```
18
19 x=[h k i l];
20 disp(x,"The indices of plane are");
21
22 //End
```

Scilab code Exa 3.12.a Interplanar Spacing

```
1 //Interplanar Spacing
2
3 clear;
4 clc;
5
6 printf("\tExample 3.12\n");
7
8 a=0.2866;           //Lattice parameter in nm
9 h=2;
10 k=2;
11 l=0;
12
13 printf("\n\tPart A");
14 d_hkl=a/(sqrt(h^2+k^2+l^2));
15 printf("\nInterplanar spacing is %.4f nm\n",d_hkl);
16
17 //End
```

Scilab code Exa 3.12.b Diffraction Angle Computations

```
1 //Diffraction Angle Computations
2
3 clear;
4 clc;
5
```

```
6 printf("\tExample 3.12\n");
7
8 a=0.2866;           //Lattice parameter in nm
9 h=2;
10 k=2;
11 l=0;
12
13 d_hkl=a/(sqrt(h^2+k^2+l^2));
14
15 printf("\n\tPart B");
16 lambda=0.1790;     //Wavelength in nm
17 n=1;
18
19 theta=asind(n*lambda/(2*d_hkl));
20 printf("\nDiffraction angle is %.2f degree\n",2*
    theta);
21
22 //End
```

Chapter 4

Imperfections in Solids

Scilab code Exa 4.1 Number of Vacancies Computation

```
1 //Number of Vacancies Computation at a Specified
   temperature
2
3 clear;
4 clc;
5
6 printf("\tExample 4.1\n");
7
8 Na=6.023*10^23;           //Avogadro No.
9 den=8.4D+06;             //Density of Copper
10 A=63.5;                 //Atomic weight of Copper
11
12 //No. of atomic site per cubic meter
13 N=Na*den/A;
14
15 //No. of vacancies at 1000 C
16 Qv=0.9;                 //Activation energy in eV
17 k=8.62*10^-5;           //Boltzmann Constant in eV/K
18 T=1000+273;             //Temperature in K
19
20 Nv=N*exp(-Qv/(k*T));
```

```

21 printf("\nNo. of vacancies are %.1f * 10^25 /m^3", Nv
    /10^25);
22
23 //End

```

Scilab code Exa 4.3 Composition Conversion From Weight Percent to Atom Percent

```

1 //Composition Conversion– From weight percent to
  Atom percent
2
3 clear;
4 clc;
5
6 printf("\t Example 4.3\n");
7
8 //Conversion to Atom percent
9 function [C]=conc(C1,C2,A1,A2)
10     C=C1*A2*100/((C1*A2)+(C2*A1));
11     funcprot(0)
12 endfunction
13
14
15 C_Al=97;           //Aluminium wt%
16 C_Cu=3;           //Copper wt%
17 A_Al=26.98;       //Atomic wt of Aluminium
18 A_Cu=63.55;       //Atomic wt of Copper
19
20 CA1=conc(C_Al,C_Cu,A_Al,A_Cu);
21 CCu=conc(C_Cu,C_Al,A_Cu,A_Al);
22
23 printf("\nAtomic %% of Al is %.1f %%",CA1);
24 printf("\nAtomic %% of Cu is %.1f %%\n",CCu);
25
26 //End

```

Scilab code Exa 4.4.a Computations of ASTM Grain Size Number

```
1 //Computations of ASTM Grain Size Number
2
3 clear;
4 clc;
5
6 printf("\tExample 4.4\n");
7
8 printf("\n\tPart A");
9
10 N=45;           //No. of grains per square inch
11
12 //Dterminin grain size no.  $N=2^{(n-1)}$ 
13 n=(log(N)/log(2))+1;
14 printf("\nGrain size no. is %.1f\n",n);
15
16 //End
```

Scilab code Exa 4.4.b Number of Grains Per Unit Area

```
1 //Number of Grains Per Unit Area
2
3 clear;
4 clc;
5
6 printf("\tExample 4.4\n");
7
8 printf("\n\tPart B");
9 N=45;           //No. of grains per square inch
10 n=(log(N)/log(2))+1;
11 M=85;
```

```
12
13 Nm=(100/M)^2*2^(n-1);
14 printf("\nAt magnification of 85x\n");
15 printf("No. of grains per inch square are %.1f\n",Nm
    );
16
17 //End
```

Chapter 5

Diffusion

Scilab code Exa 5.1 Diffusion Flux Computation

```
1 //Diffusion Flux Computation
2
3 clear;
4 clc;
5
6 printf("\tExample 5.1\n");
7
8 Ca=1.2;          //Concentration at A in kg/m^3
9 Cb=0.8;          //Concentration at B in kg/m^3
10
11 xa=5*10^-3;     //Position 1 in m
12 xb=10*10^-3;   //Position 2 in m
13
14 D=3*10^-11;     //Diffusion coefficient in m^2/s
15
16 J=-D*(Ca-Cb)/(xa-xb);
17 printf("\nDiffusion flux is %.1f * 10^-9 kg/m^2-s",J
18         /10^-9);
19 //End
```

Scilab code Exa 5.2 Nonsteady State Diffusion Time Computation

```
1 //Nonsteady-State Diffusion Time Computation I
2
3 clear;
4 clc;
5
6 printf("\tExample 5.2\n");
7
8 Co=0.25;           //Initial Conc. in wt%
9 Cs=1.2;           //Surface conc. in wt%
10 Cx=0.8;          //Conc. at any x in wt%
11
12 x=5*10^-4;       //Position in m
13 D=1.6*10^-11;   //Diffusion coeff in m^2/s
14
15 C=1-((Cx-Co)/(Cs-Co));
16 z=erfinv(C);
17
18 //But C=erf(x/2sqrt(Dt))
19 t=x^2/(4*D*z^2);
20
21 printf("\nTime required is %d s or %.1f h\n",t,t
22       /3600);
23 //End
```

Scilab code Exa 5.3 Nonsteady State Diffusion Time Computation II

```
1 //Nonsteady-State Diffusion Time Computation II
2
3 clear;
```

```

4  clc;
5
6  printf("\tExample 5.3\n");
7
8  D500=4.8*10^-14;      //Diffusion coefficient at 500
   C
9
10 D600=5.3*10^-13;     //Diffusion coefficient at 600
   C
11 t600=10;             //Time in hours to diffuse
12
13 t500=D600*t600/D500;
14
15 printf("\nTime to diffuse at 500 C is %.1f h\n",t500
   );
16
17 //End

```

Scilab code Exa 5.4 Diffusion Coefficient Determination

```

1  //Diffusion Coefficient Determination
2
3  clear;
4  clc;
5
6  printf("\tExample 5.4\n");
7
8  T=550+273;           //in K
9  D0=1.2*10^-4;       //Temperature independent
   preexponential in m^2/s
10 Qd=131000;           //Activation energy in J/mol-K
11 R=8.31;              //Universal Gas constt
12
13 D=D0*exp(-Qd/(R*T));
14

```

```

15 printf("\nDiffusion coefficient is %.1f * 10-13 m
      ^2/s\n",D/10-13);
16
17 //End

```

Scilab code Exa 5.5 Diffusion Coefficient Activation Energy

```

1 //Diffusion Coefficient Activation Energy and
  Preexponential Calculations
2
3 clear;
4 clc;
5
6 printf("\tExample 5.5\n");
7
8 //From graph log D ad 1/T are deducted
9 inv_T1=0.8*10-3; //Reciprocal of temp. in
  K-1
10 inv_T2=1.1*10-3; //Reciprocal of temp. in
  K-1
11 logD1=-12.4;
12 logD2=-15.45;
13
14 R=8.31; //Gas law Constant in J/
  mol-K
15
16 Qd=-2.3*R*(logD1-logD2)/(inv_T1-inv_T2);
17 printf("\nActivation energy is %d kJ/mol",Qd/1000);
18
19 //For calculating Peexponential factor
20 D0=10(logD2+(Qd*inv_T2/(2.3*R)));
21 printf("\nPreexponential factor D0 is %.1f * 10-5 m
      ^2/s\n",D0/10-5);
22
23 //End

```

Scilab code Exa 5.6 Diffusion Temperature Time Heat Treatment Specification

```
1 //Diffusion TemperatureTime Heat Treatment
   Specification
2
3 clear;
4 clc;
5
6 printf("\tDesign Example 5.1\n");
7
8 C0=0.2;           //Initial concentration in wt%
9 Cs=1;            //Surface conc in wt%
10 Cx=0.6;          //Conc at any position X in wt%
11 x=7.5*10^-4;     //Position in m
12 D0=2.3*10^-5;    //Preexponential factor in m^2/s
13 R=8.31;          //Gas law constant in J/mol-K
14 Qd=148000;       //Activation energy in J/mol
15
16 C=1-((Cx-C0)/(Cs-C0));
17 z=erfinv(C);
18 Dt=(x/(2*z))^2;
19
20 //Dt=D0*exp(-Qd/RT)*t = value of variable Dt
21 D=Dt/D0;
22
23 T=[900 950 1000 1050];
24 for i=1:4
25     t(i)=D/exp(-Qd/(R*(T(i)+273)))/3600;
26 end
27
28 printf("\nTemperature(in Celsius) is\n");
29 disp(T);
30 printf("\nTime is (in hours)\n");
31 disp(t);
```

32

33 //End

Chapter 6

Mechanical Properties of Metals

Scilab code Exa 6.1 Elongation Computation

```
1 //Elongation (Elastic) Computation
2
3 clear;
4 clc;
5
6 printf("\tExample 6.1\n");
7
8 E=110*10^3; //Young's modulus of Copper in MPa
9 sigma=276; //Applied stress in MPa
10 lo=305; //Original length in mm
11
12 //Deformation
13 dl=sigma*lo/E;
14
15 printf("\nElongation obtained is %.2f mm \n",dl);
16
17 //End
```

Scilab code Exa 6.2 Computation of Load to Produce Specified Diameter Change

```
1 //Computation of Load to Produce Specified Diameter
   Change
2
3 clear;
4 clc;
5
6 printf("\tExample 6.2\n");
7
8 del_d=-2.5*10^-3; //Deformation in dia in mm
9 d0=10;           //Initial dia in mm
10
11 v=0.34;         //Poisson ratio for brass
12
13 ex=del_d/d0;
14 printf("\nStrain in x-direction is %f",ex);
15
16 ez=-ex/v;
17 printf("\nStrain in z-direction is %f",ez);
18
19 E=97*10^3;      //Modulus of elasticity in MPa
20 sigma=ez*E;
21 F=sigma*pi*(d0^2)/4;
22
23 printf("\nApplied force is %d N",F);
24
25 //End
```

Scilab code Exa 6.3.a Modulus of elasticity

```
1 clear;
```

```

2  clc;
3
4  printf("\tExample 6.3\n");
5
6  //From draph in the question
7  //stress and strain can be obtained
8
9  si2=150;           // in MPa
10 si1=0;
11 e2=0.0016;
12 e1=0;
13 d0=12.8*10^-3;    //Initial Diameter in m
14
15 printf("\n\tPart A");
16 //Young's Modulus = stress/strain
17 E=(si2-si1)/(e2-e1);
18 printf("\nModulus of elasticity is %.2f GPa",E/10^3)
    ;
19
20 //End

```

Scilab code Exa 6.3.c Maximum load

```

1  clear;
2  clc;
3
4  printf("\tExample 6.3\n");
5
6  //From draph in the question
7  //stress and strain can be obtained
8
9  si2=150;           // in MPa
10 si1=0;
11 e2=0.0016;
12 e1=0;

```

```

13 d0=12.8*10^-3;    //Initial Diameter in m
14
15 printf("\n\tPart C");
16 A0=%pi*d0^2/4;
17 sig=450*10^6;    //tensile strength in MPa
18
19 F=sig*A0;
20 printf("\nMaximum load sustained is %d N\n",F);
21
22 //End

```

Scilab code Exa 6.3.d Change in length

```

1 clear;
2 clc;
3
4 printf("\tExample 6.3\n");
5
6 //From draph in the question
7 //stress and strain can be obtained
8
9 printf("\n\tPart D");
10
11 //From stress-strain curve
12 //Strain corresponding to stress of 345 MPa is 0.06
13
14 l0=250;    //Initial lengt in mm
15 e=0.06;    //strain
16
17 dl=e*l0;
18 printf("\nChange in length is %d mm",dl);
19
20
21 //End

```

Scilab code Exa 6.4.a Ductility Computations

```
1 //Ductility
2
3 clear;
4 clc;
5
6 printf("\tExample 6.4\n");
7
8 di=12.8;      //Initial dia in mm
9 df=10.7;      //Final dia in mm
10
11 printf("\n\tPart A");
12
13 //Ductility in terms of Reduction Area
14 RA = ((di^2-df^2)/di^2)*100;
15 printf("\npercent reduction in area is %d%%\n",RA);
16
17 //End
```

Scilab code Exa 6.4.b True Stress At Fracture Computations

```
1 //True-Stress-At-Fracture Computations
2
3 clear;
4 clc;
5
6 printf("\tExample 6.4\n");
7
8 di=12.8;      //Initial dia in mm
9 df=10.7;      //Final dia in mm
10
```

```

11 printf("\n\tPart B");
12 Ao=%pi*di^2*10^-6/4;
13 sig=460*10^6;    //Tensile strength
14
15 F=sig*Ao;
16 printf("\nF = %d N",F);
17
18 Af=%pi*df^2/4;
19 sig_t=F/Af;
20 printf("\nTrue stress is %d MPa",sig_t);
21
22 //End

```

Scilab code Exa 6.5 Calculation of Strain Hardening Exponent

```

1 //Calculation of Strain-Hardening Exponent
2
3 clear;
4 clc;
5
6 printf("\tExample 6.5\n");
7
8 sig_t=415;    //True stress in MPa
9 et=0.1;      //True strain
10 K=1035;     // In MPa
11
12 n=log(sig_t/K)/log(et);
13
14 printf("\nStrain - hardening coefficient is %.2f",n)
15     ;
16 //End

```

Scilab code Exa 6.6.a Average Computations

```
1 //Average Computations
2
3 clear;
4 clc;
5
6 printf("\tExample 6.6a\n");
7
8 //First and Last point are arbitrary to plot the
   required 4 points
9 n=[0 1 2 3 4 5];
10 TS=[510 520 512 515 522 525];
11
12 plot(n,TS, '+');
13 xtitle('Tensile strength data', 'Sample no.', 'Tensile
   strength');
14
15 //Mean Tensile strength
16 i=2;
17 TSmean=0;
18
19 for i=2:5
20     TSmean=TSmean+(TS(i)/4);
21 end
22 printf("\nMean tensile strength is %d MPa\n",TSmean)
   ;
23
24 //End
```

Scilab code Exa 6.6.b Standard Deviation Computations

```
1 //Standard Deviation Computations
2
3 clear;
```

```

4  clc;
5
6  printf("\tExample 6.6b\n");
7
8  //First and Last point are arbitrary to plot the
   required 4 points
9  n=[0 1 2 3 4 5];
10 TS=[510 520 512 515 522 525];
11
12 i=2;
13 TSmean=0;
14
15 for i=2:5
16     TSmean=TSmean+(TS(i)/4);
17 end
18
19 //Standard Deviation
20 j=0;
21 std=0;
22
23 for i=2:5
24     std=std+((TS(i)-TSmean)^2/(4-1));
25 end
26
27 printf("\nStandard deviation is %.1f MPa\n",sqrt(std
   ));
28
29 //End

```

Scilab code Exa 6.7 Specification of Support Post Diameter

```

1  //Specification of Support Post Diameter
2
3  clear;
4  clc;

```

```
5
6 printf("\tDesign Example 6.1\n");
7
8 sig_y=310;          //Minimum yield strength in MPa
9 N=5;              // Conservative factor of safety
10
11 F=220000/2;       //Two rods must support half of the
    total force
12
13 sig_w=sig_y/N;
14 d=2*sqrt(F/(%pi*sig_w));
15
16 printf("\nDiameter of each of the two rods is %.1f
    mm\n",d);
17
18 //End
```

Chapter 7

Dislocations and Strengthening Mechanisms

Scilab code Exa 7.1.a Resolved Shear Stress

```
1 //Resolved Shear Stress Computations
2
3 clear;
4 clc;
5
6 printf("\tExample 7.1\n");
7
8 x=[1 1 0]; //Indices of Plane
9 y=[0 1 0]; //direction of applied tensile stress
10 z=[-1 1 1]; //Direction of shear stress
11
12 function [angle]=dotproduct(a,b)
13     num=(a(1)*b(1))+a(2)*b(2))+a(3)*b(3));
14     den=sqrt((a(1)^2+a(2)^2+a(3)^2)*(b(1)^2+b(2)^2+b
15         (3)^2));
16     angle=acos(num/den);
17     funcprot(0)
18 endfunction
```

```

19 phi=dotproduct(x,y);
20 lambda=dotproduct(y,z);
21
22 printf("\nAngles phi is %.1f degree and lambda is %
    .1f degree\n",phi*180/%pi,lambda*180/%pi);
23
24
25 //When a tensile stress of 52 MPa (7500 psi) is
    applied
26 printf("\n\tPart A\n");
27 sigma=52; //in MPa
28 tr=sigma*cos(phi)*cos(lambda);
29 printf("Resolved shear stress is %.1f MPa\n",tr);
30
31 //End

```

Scilab code Exa 7.1.b Stress to Initiate Yielding Computations

```

1 //Stress-to-Initiate-Yielding Computations
2
3 clear;
4 clc;
5
6 printf("\tExample 7.1\n");
7
8 x=[1 1 0]; //Indices of Plane
9 y=[0 1 0]; //direction of applied tensile stress
10 z=[-1 1 1]; //Direction of shear stress
11
12 function [angle]=dotproduct(a,b)
13     num=(a(1)*b(1))+a(2)*b(2))+a(3)*b(3));
14     den=sqrt((a(1)^2+a(2)^2+a(3)^2)*(b(1)^2+b(2)^2+b
        (3)^2));
15     angle=acos(num/den);
16     funcprot(0)

```

```

17 endfunction
18
19 phi=dotproduct(x,y);
20 lambda=dotproduct(y,z);
21
22 printf("\nAngles phi is %.1f degree and lambda is %
    .1f degree\n",phi*180/%pi,lambda*180/%pi);
23
24 //If slip occurs on a (110) plane and in a [-1 1 1]
    direction , and the critical resolved shear stress
    is 30 MPa
25 printf("\n\tPart B")
26
27 trc=30; //in MPa Critical resolved shear stress
28
29 sy=trc/(cos(phi)*cos(lambda));
30 printf("\nYield strength is %.1f MPa\n",sy);
31
32 //End

```

Scilab code Exa 7.2 Tensile Strength and Ductility Determinations

```

1 //Tensile Strength and Ductility Determinations for
    Cold-Worked Copper
2
3 clear;
4 clc;
5
6 printf("\t Example 7.2\n");
7
8 df=12.2; //Final dia in mm
9 di=15.2; //Initial dia in mm
10
11 CW = ((di^2-df^2)/di^2)*100;
12

```

```

13 printf("\nCold work is %.1f %%\n",CW);
14
15 ts=340; //in Mpa tensile strength
16 duc=7; //in % Ductility
17 printf("\nFrom graph of Fig. 7.19b in book");
18 printf("\nTensile strength is %d MPa",ts);
19 printf("\nDuctility is %d %% EL\n",duc);
20
21 //End

```

Scilab code Exa 7.3 Description of Diameter Reduction Procedure

```

1 //Description of Diameter Reduction Procedure
2
3 clear;
4 clc;
5
6 printf("\tDesign Example 7.1\n");
7
8 di=6.4; //Initial dia in mm
9 df=5.1; //Final dia in mm
10
11 //Cold Work Computation
12 CW = ((di^2-df^2)/di^2)*100;
13
14 printf("\nCold work is %.1f %%\n",CW);
15
16 //From Figures 7.19a and 7.19c,
17 //A yield strength of 410 MPa
18 //And a ductility of 8% EL are attained from this
    deformation
19
20 printf("\nBut required ductility and yield strength
    is not matched at this cold work");
21 printf("\nHence required Cold work is 21.5 %%");

```

```
22
23 //x=poly([0], 'x');
24 dmid = sqrt(df^2/(1-0.215));
25
26 printf("\nHence first draw to %.1fmm and then to %.1
      fmm\n", dmid, df);
27
28 //End
```

Chapter 8

Failure

Scilab code Exa 8.1 Maximum Flaw Length Computation

```
1 //Maximum Flaw Length Computation
2
3 clear;
4 clc;
5
6 printf("\tExample 8.1\n");
7
8 sigma=40*10^6; // in Pa Tensile stress
9 E=69*10^9; //Modulus of elaticity in pa
10 Ys=0.3; //Specific surface energy in N/m^2
11
12 //Maximum length of a surface flaw
13 a=2*E*Ys/(%pi*sigma^2);
14
15 printf("\nMaximum length of a surface falw without
    fracture : %.4f mm\n",a/10^-3);
16
17 //End
```

Scilab code Exa 8.2 Rupture Lifetime Prediction

```
1 //Rupture Lifetime Prediction
2
3 clear;
4 clc;
5
6 printf("\tDesign Example 8.2\n");
7
8 T=800+273; // Temperature in K
9
10 //stress is 140 MPa
11 //From Graph of Fig. 8.32 Larson–Miller Parameter is
    deduced
12 L_M=24*10^3;
13
14 t=10^((L_M/T)-20);
15
16 printf("\nTime to rupture is : %d hours\n",t);
17
18 //End
```

Scilab code Exa 8.3 Estimating theoretical fracture strength

```
1 //Estimating theoretical fracture strength
2
3 clear;
4 clc;
5
6 printf("\tExample 8.3\n");
7
8 a=0.5; //Crack length in mm
9 ro=5D-3; //Crack tip radius of curvature in
    mm
10 sig=1035; //Applied stress in MPa
```

```

11
12 sigm=2*sig*sqrt(a/ro);
13
14 printf("\nFracture strength is %.2e MPa\n",sigm);
15
16 //End

```

Scilab code Exa 8.4 Computation of critical shear stress

```

1 //Computation of critical shear stress
2
3 clear;
4 clc;
5
6 printf("\tExample 8.4\n");
7
8 E=393D9;           //Young's modulus for Al
9 gam=0.9;           //surface energy in J/m^2
10 a=4D-4;           //Crack length in m
11
12 sigc=sqrt(2*E*gam/(%pi*a/2));
13
14 printf("\nCritical shear stress is %.2e N/m^2\n",
15         sigc);
16 //End

```

Scilab code Exa 8.5 Determining maximum allowable surface crack length

```

1 //Determining maximum allowable surface crack length
2
3 clear;
4 clc;

```

```
5
6 printf("\tExample 8.5\n");
7
8 E=225D9;           //Young's modulus
9 gam=1;            //surface energy in N/m
10 sigc=13.5D6;      //Critical shear stress in N
    /m^2
11
12 a=2*E*gam/(%pi*sigc^2);
13
14 printf("\nMaximum allowable crack length is %.1e m\n
    ",a);
15
16 //End
```

Chapter 9

Phase Diagrams

Scilab code Exa 9.1 Lever Rule Derivation

```
1 //Lever Rule derivation
2
3 clear;
4 clc;
5
6 printf("\tExample 9.1\n");
7
8 disp("Since only 2 phases are present");
9 disp("W_alpha + WL = 1");
10 disp("W_alpha*C_alpha + WL*C_L = C0");
11 disp("hence");
12
13 disp("WL = (C_alpha-C0)/(C_alpha-C_L)");
14 disp("W_alpha = (C0-C_L)/(C_alpha-C_L)");
15
16
17 //End
```

Scilab code Exa 9.2 Determination of Phases Present

```

1 //Determination of Phases Present and Computation of
  Phase Compositions
2
3 clear;
4 clc;
5
6 printf("\tExample 9.2\n");
7
8 printf("\nfor given composition and temp. of 150 C")
  ;
9 printf("\ni) both alpha and beta phase coexist\n");
10
11 disp("ii) Constructing the tie line");
12 printf("10 wt%% Sn - 90 wt%% Pb for C_alpha");
13 printf("\n98 wt%% Sn - 2 wt%% Pb for C_beta");
14
15 //End

```

Scilab code Exa 9.3.a Relative Phase Amount Determinations

```

1 //Determination of Phases Present
2
3 clear;
4 clc;
5
6 printf("\tExample 9.3\n");
7
8 printf("\n\tPart A");
9 C1=40; // Overall alloy composition
10 Cb=98;
11 Ca=10;
12
13 Wa=(Cb-C1)/(Cb-Ca);
14 Wb=(C1-Ca)/(Cb-Ca);
15

```

```

16 printf("\nMass fractions for alpha and beta phases
    are : %.2f and %.2f respectively\n",Wa,Wb);
17
18 //End

```

Scilab code Exa 9.3.b Mass and Volume Fractions

```

1 //Computation of Phase Compositions
2
3 clear;
4 clc;
5
6 printf("\tExample 9.3\n");
7
8 C1=40; // Overall alloy composition
9 Cb=98;
10 Ca=10;
11
12 Wa=(Cb-C1)/(Cb-Ca);
13 Wb=(C1-Ca)/(Cb-Ca);
14
15 printf("\n\tPart B");
16 d_Sn=7.24; // in g/cm^3 density of tin
17 d_Pb=11.23; // in g/cm^3 density of lead
18
19 Ca_Sn=10;
20 Ca_Pb=90;
21
22 Cb_Sn=98;
23 Cb_Pb=2;
24
25 d_a=100/((Ca_Sn/d_Sn)+(Ca_Pb/d_Pb));
26 d_b=100/((Cb_Sn/d_Sn)+(Cb_Pb/d_Pb));
27
28 printf("\nDensity of alpha phase is : %.2f g/cm^3",

```

```

    d_a);
29 printf("\nDensity of beta phase is : %.2f g/cm^3",
    d_b);
30
31 Va=Wa/(d_a*((Wa/d_a)+(Wb/d_b)));
32 Vb=Wb/(d_b*((Wa/d_a)+(Wb/d_b)));
33
34 printf("\n\nVolume fraction of alpha phase : %.2f",
    Va);
35 printf("\n\nVolume fraction of beta phase : %.2f",Vb);
36
37 //End

```

Scilab code Exa 9.4.a fractions of total ferrite and cementite phases

```

1 //Determining ferrite and cementite phase
2
3 clear;
4 clc;
5
6 printf("\t Example 9.4\n");
7
8 printf("\n\tPart A");
9 C0=0.35;
10 Ca=0.022;
11 C_Fe3C=6.7;
12
13 Wa=(C_Fe3C-C0)/(C_Fe3C-Ca);
14 W_Fe3C=(C0-Ca)/(C_Fe3C-Ca);
15
16 printf("\nMass fraction of total ferritic phase : %
    .2f",Wa);
17 printf("\nMass fraction of Fe3C : %.2f\n",W_Fe3C);
18
19 //End

```

Scilab code Exa 9.4.b Fractions of the proeutectoid ferrite and pearlite

```
1 //Determining proeutectoid ferrite and pearlite
2
3 clear;
4 clc;
5
6 printf("\t Example 9.4\n");
7
8 C0=0.35;
9 Ca=0.022;
10 C_Fe3C=6.7;
11
12 printf("\n\tPart B");
13 C_p=0.76;
14
15 Wp=(C0-Ca)/(C_p-Ca);
16 W_a=(C_p-C0)/(C_p-Ca);
17
18 printf("\nMass fraction of Pearlite : %.2f",Wp);
19 printf("\nMass fraction of proeutectoid ferrite : %
    .2f\n",W_a);
20
21 //End
```

Scilab code Exa 9.4.c Fraction of eutectoid ferrite

```
1 //Determining eutectoid ferrite
2
3 clear;
4 clc;
```

```

5
6 printf("\t Example 9.4\n");
7
8 C0=0.35;
9 Ca=0.022;
10 C_Fe3C=6.7;
11
12 C_p=0.76;
13
14 Wp=(C0-Ca)/(C_p-Ca);
15 W_a=(C_p-C0)/(C_p-Ca);
16
17 Wa=(C_Fe3C-C0)/(C_Fe3C-Ca);
18 printf("\n\tPart C");
19
20 Wae=Wa-W_a;
21 printf("\nMass fraction of eutectoid ferrite : %.3f\n
      n",Wae);
22
23 //End

```

Chapter 10

Phase Transformations in Metals

Scilab code Exa 10.1.a Computation of Critical Nucleus Radius

```
1 //Computation of Critical Nuclear Radius
2
3 clear;
4 clc;
5
6 printf("\tExample 10.1\n");
7
8 printf("\n\tPart A");
9 Hf=-1.16*10^9; // in J/m^3 latent heat of fusion
10 Y=0.132; // in J/m^2 Surface energy
11 Tm=1064+273; // in K Melting point of gold
12 T=1064+273-230; // in K 230 is supercooling value
13
14 r=-2*Y*Tm/(Hf*(Tm-T));
15
16 printf("\nCritical Radius is : %.2f nm\n",r/10^-9);
17
18 G=16*pi*Y^3*Tm^2/(3*Hf^2*(Tm-T)^2);
19
```

```

20 printf("\nActivation free energy is : %.2e J\n",G);
21
22 //end

```

Scilab code Exa 10.1.b Activation Free Energy Computation

```

1 //Activation Free Energy
2
3 clear;
4 clc;
5
6 printf("\tExample 10.1\n");
7
8 Hf=-1.16*10^9; // in J/m^3 latent heat of fusion
9 Y=0.132; // in J/m^2 Surface energy
10 Tm=1064+273; // in K Melting point of gold
11 T=1064+273-230; // in K 230 is supercooling value
12 r=-2*Y*Tm/(Hf*(Tm-T));
13
14 printf("\n\tPart B");
15 a=0.413*10^-9; // in m Unit Cell edge length
16
17 //unit cells per paticle
18 u_c=4*pi*r^3/(3*a^3);
19
20 printf("\nUnit cells per paricle are : %d",u_c);
21 printf("\nIn FCC there are 4 atoms per unit cell.\n"
22 );
23 printf("\nTotal no. of atoms per critical nucleus
24 are : %d\n",int(u_c)*4);
25 //End

```

Scilab code Exa 10.2 Determination the rate of recrystallization

```
1 //Determination the rate of recrystallization
2
3 clear;
4 clc;
5
6 printf("\tExample 10.2\n");
7 n=5;
8 y=0.3;
9 t=100;           //in min
10
11
12 k=-log(1-y)/t^n;
13
14 thalf=(-log(1-0.5)/k)^(1/n);
15
16 rate=1/thalf;
17
18 printf("\nRate is %.2e (min)^-1\n",rate);
19
20 //End
```

Chapter 12

Structures and Properties of Ceramics

Scilab code Exa 12.1 Computation of Minimum Cation to Anion Radius Ratio

```
1 //Computation of Minimum Caion-to-Anion Radius Ratio
   forCo-ordination No. of 3
2
3 clear;
4 clc;
5
6 printf("\tExample 12.1\n");
7
8 printf("\nFor equilateral triangle after joining
   centres of the atoms \nAngle = 30\n");
9
10 a=30;
11
12 ratio=(1-cos(30*%pi/180))/cos(30*%pi/180);
13
14 printf("\nCation to anion raio is : %.3f\n",ratio);
15
16 //End
```

Scilab code Exa 12.2 Ceramic Crystal Structure Prediction

```
1 //Ceramic Crystal structure prediction
2
3 clear;
4 clc;
5
6 printf("\tExample 12.2\n");
7
8 r_Fe=0.077; // in nm Radius of iron cation Fe++
9
10 r_O=0.140; //in nm Radius of Oxygen anion O—
11
12 ratio=r_Fe/r_O;
13
14 printf("\nRatio is : %0.3f",ratio);
15
16 if ratio>0.414 & ratio<0.732 then
17     printf("\nCo-ordinaton no. is 6");
18     printf("\nStructure is Rock Salt type\n");
19 end
```

Scilab code Exa 12.3 Theoretical Density Calculation

```
1 //Theoretical Density Determination for NaCl
2
3 clear;
4 clc;
5
6 printf("\tExample12.3\n");
7
8 A_Na=22.99; // in g/mol
```

```

 9 A_Cl=35.45; //in g/mol
10
11 r_Na=0.102*10^-7; //in cm Radius of Na+ ion
12 r_Cl=0.181*10^-7; //in cm Radius of Cl- ion
13
14 a=2*(r_Na+r_Cl);
15 V=a^3;
16
17 n=4; //For FCC, no. of atoms are 4 per crystal
18
19 Na=6.023*10^23; //Avogadro number
20
21 density=n*(A_Na+A_Cl)/(V*Na);
22
23 printf("\nDensity is : %0.2f g/cm^3\n",density);
24
25 //End

```

Scilab code Exa 12.4 Computation of the Number of Schottky Defects

```

1 //Computation of the No. of Schottky Defects in KCl
2
3 clear;
4 clc;
5
6 printf("\tExample 12.4\n");
7
8 Na=6.023*10^23; //Avogadro number
9 density=1.955; //in g/cm^3
10
11 A_K=39.1; //in g/mol
12 A_Cl=35.45; //in g/mol
13
14 N=Na*density*10^6/(A_K+A_Cl);
15

```



```

16 printf("\nNo. of lattice points are : %0.2f * 10 ^
    28 /m^3\n",N/10^28);
17
18 Qs=2.6; // in eV
19 k=8.62*10^-5; // in eV/K Boltzmann Constant
20 T=500+273; // in K
21
22 Ns=N*%e^(-Qs/(2*k*T));
23
24 printf("\nSchottky Defects are : %0.2f * 10 ^ 19 /m
    ^3\n",Ns/10^19);
25
26 //End

```

Scilab code Exa 12.5 Determination of Possible Point Defect Types

```

1 //Determination of Possible Point Defect Types
2
3 clear;
4 clc;
5
6 printf("\tExample 12.5\n");
7
8 disp("Replacement of Na+ by a Ca++ ion introduces
    one extra positive charge");
9
10 disp("Removal of a positive charge is accomplished
    by the formation of one Na+ vacancy");
11
12 disp("Alternatively , a Cl- interstitial will supply
    an additional negative charge, negating the
    effect of each Ca++ ion.");
13
14 disp("The formation of this defect is highly
    unlikely.");

```

15

16 //End

Chapter 14

Polymer Structures

Scilab code Exa 14.2.a Computations of the Density

```
1 //Computations of the Density
2
3 clear;
4 clc;
5
6 printf("\tExample 14.2\n");
7
8 printf("\n\tPart A");
9 Ac=12.01; //in g/mol Molecular weight of Carbon
10 Ah=1.008; //in g/mol molecular weight of hydrogen
11 a=7.41*10^-8; //in cm
12 b=4.94*10^-8; //in cm
13 c=2.55*10^-8; //in cm
14 Na=6.023*10^23;
15
16 Vc=a*b*c;
17 n=2;
18 A=(2*Ac)+(4*Ah);
19
20 density_c=n*A/(Vc*Na);
21
```

```

22 printf("\nDensity is : %f g/cm^3\n",density_c);
23
24 //End

```

Scilab code Exa 14.2.b Computation of Percent Crystallinity

```

1 //Percent Crystallinity of Polyethylene
2
3 clear;
4 clc;
5
6 printf("\tExample 14.2\n");
7
8 printf("\n\tPart B");
9 density_a=0.870; // in g/cm^3
10 density_s=0.925; // in g/cm^3
11
12 Ac=12.01; //in g/mol Molecular weight of Carbon
13 Ah=1.008; //in g/mol molecular weight of hydrogen
14 a=7.41*10^-8; //in cm
15 b=4.94*10^-8; //in cm
16 c=2.55*10^-8; //in cm
17 Na=6.023*10^23;
18
19 Vc=a*b*c;
20 n=2;
21 A=(2*Ac)+(4*Ah);
22
23 density_c=n*A/(Vc*Na);
24
25 pc=density_c*(density_s-density_a)*100/(density_s*(
    density_c-density_a));
26
27 printf("\npercentage crystallinity is : %.1f %%\n",
    pc);

```

28
29 //End

Scilab code Exa 14.3.a Computations of Diffusion Flux of Carbon Dioxide

```
1 //Computations of Diffusion Flux of Carbon dioxide
  through Plasic Beverage Container
2
3 clear;
4 clc;
5
6 function [A]= approx(V,n)
7     A= round(V*10^n)/10^n;
8     funcprot(0)
9 endfunction
10
11 printf("\tExample 14.3\n");
12
13 printf("\n\tPart A");
14 P1=400000; // in Pa Pressure inside the bottle
15 P2=400; // in Pa Pressure outside the bottle
16 Pm=0.23*10^-13; //Solubility Cooeffier
17 dx=0.05; // in cm Thickness of wall
18
19 J=approx(-Pm*(P2-P1)/dx,8);
20 printf("\nDiffusion flux is : %.2f * 10 ^ -6 ",J
  /10^-6);
21 printf("cm^3 STP/cm^2-s\n");
22
23 //End
```

Scilab code Exa 14.3.b Computation of Beverage Shelf Life

```

1 //Beverage Shell Life
2
3 clear;
4 clc;
5
6 function [A]= approx(V,n)
7     A= round(V*10^n)/10^n;
8     funcprot(0)
9 endfunction
10
11 printf("\tExample 14.3\n");
12
13 P1=400000; // in Pa Pressure inside the bottle
14 P2=400; // in Pa Pressure outside the bottle
15 Pm=0.23*10^-13; // Solubility Coefficient
16 dx=0.05; // in cm Thickness of wall
17
18 J=approx(-Pm*(P2-P1)/dx,8);
19
20 printf("\n\tPart B");
21 A=500; //surface area of bottle in cm^2
22 V_lose=750; //cm^3 STP
23
24 V=J*A;
25 t=V_lose/V;
26 time=t/(3600*24);
27
28 printf("\nTime to escape is : %.2e sec or %d days\n"
29     ,t,time);
30 //End

```

Chapter 16

Composites

Scilab code Exa 16.1.a Modulus of elasticity

```
1 clear;
2 clc;
3
4 x=poly([0], 'x');
5 printf("\tExample 16.1\n");
6
7 printf("\n\tPart A");
8 E_gf=69; // in GPa Elasticity of glass fibre
9 mf_gf=0.4; //Vol % of glass fibre
10 E_pr=3.4; // in GPa Elasticity of polyester resin
11 mf_pr=0.6; //Vol % of polyester resin
12
13 E_c1=(E_pr*mf_pr)+(E_gf*mf_gf);
14 printf("\nModulus of elasticity of composite is : %f
        GPa\n",E_c1);
15
16 //End
```

Scilab code Exa 16.1.b Magnitude of the load

```

1 clear;
2 clc;
3
4 x=poly([0], 'x');
5 printf("\tExample 16.1\n");
6
7 E_gf=69; // in GPa Elasticity of glass fibre
8 mf_gf=0.4; //Vol % of glass fibre
9 E_pr=3.4; // in GPa Elasticity of polyester resin
10 mf_pr=0.6; //Vol % of polyester resin
11
12 printf("\n\tPart B");
13 Ac=250; //mm^2
14 sigma=50; //MPa
15 f=(E_gf*mf_gf)/(E_pr*mf_pr);
16 Fc=Ac*sigma; //N
17 Fm=roots(f*x+x-Fc); //N
18 printf("\nFm is : %f N\n",Fm);
19
20 Ff=Fc-Fm;
21 printf("\nLoad carried by each of fiber and matrix
    phase is : %f N\n",Ff);
22
23 //End

```

Scilab code Exa 16.1.c Strain determination

```

1 clear;
2 clc;
3
4 x=poly([0], 'x');
5 printf("\tExample 16.1\n");
6
7 E_gf=69; // in GPa Elasticity of glass fibre
8 mf_gf=0.4; //Vol % of glass fibre

```



```

 9 E_pr=3.4; // in GPa Elasticity of polyester resin
10 mf_pr=0.6; //Vol % of polyester resin
11
12 Ac=250; //mm^2
13 sigma=50; //MPa
14 f=(E_gf*mf_gf)/(E_pr*mf_pr);
15 Fc=Ac*sigma; //N
16 Fm=roots(f*x+x-Fc); //N
17
18 Ff=Fc-Fm;
19
20 printf("\n\tPart C");
21 Am=mf_pr*Ac;
22 Af=mf_gf*Ac;
23 sigma_m=Fm/Am;
24 sigma_f=Ff/Af;
25
26 e_m=sigma_m/E_pr; //Strain for matrix phase
27 e_f=sigma_f/E_gf; //Strain for fiber phase
28
29 printf("\nStrain for matrix phase is : %f\n",e_m);
30 printf("\nStrain for fiber phase is : %f\n",e_f);
31
32 //End

```

Scilab code Exa 16.2 Elastic Modulus Determination

```

1 //Elastic Modulus Determination for a Glass Fiber-
  Reinforced Composite Transverse Direction
2
3 clear;
4 clc;
5
6 printf("\tExample 16.2\n");
7 E_gf=69; // in GPa Elasticity of glass fibre

```

```
8 mf_gf=0.4; //Vol % of glass fibre
9 E_pr=3.4; // in GPa Elasticity of polyester resin
10 mf_pr=0.6; //Vol % of polyester resin
11
12 E_ct=E_pr*E_gf/((E_pr*mf_gf)+(E_gf*mf_pr)); //GPa
13
14 printf("\nIn transverse direction , modulus of
    elaticity is : %f GPa\n",E_ct);
15
16 //End
```

Chapter 17

Corrosion and Degradation of Materials

Scilab code Exa 17.1 Determination of Electrochemical Cell Characteristics

```
1 //Determination of Electrochemical Cell
   Characteristics
2
3 clear;
4 clc;
5
6 printf("\tExample 17.1\n");
7
8 V_Cd=-0.403; //Half Cell Potential of Cd++|Cd
9 V_Ni=-0.250; //Half Cell Potential of Ni++|Ni
10 dV=V_Ni-V_Cd;
11 printf("\nStandard Cell potential is : %f V\n",dV);
12
13 C_Ni=10-3;
14 C_Cd=0.5;
15 n=2; //Net electron exchange in Redox reaction
16 V=-dV-(0.0592*log10(C_Ni/C_Cd)/n);
17 printf("\nNet EMF is : %f V\n",V);
18 printf("\nHence\n");
```

```

19
20 if V<0 then
21     printf("\nNi is reduced & Cd is oxidised\n");
22 else
23     printf("\nCd is reduced & Ni is oxidised\n");
24 end
25
26 //End

```

Scilab code Exa 17.2.a Rate of Oxidation Computation

```

1 //Rate of Oxidation Computation
2
3 clear;
4 clc;
5
6 printf("\tExample 17.2 a\n");
7
8 //Activation polarisation data
9 VH_H2=0;
10 VZn_Zn2=-0.763;
11 iZn=10^-7;
12 iH2=10^-10;
13 beta_Zn=0.09;
14 beta_H2=-0.08;
15
16 //Part i
17 ic=10^[(VH_H2-VZn_Zn2-(beta_H2*log10(iH2))+(beta_Zn*
18     log10(iZn)))/(beta_Zn-beta_H2)];
19
20 disp(ic,'ic is ');
21
22 n=2; //Exchange of 2 electrons
23 F=96500; //Faradays constant

```

```

24 r=ic/(n*F);
25 printf("\ni) Rate of oxiadation is %.1f * 10^-10 mol
    /cm^2-s\n",r/10^-10);
26
27
28 //End

```

Scilab code Exa 17.2.b Corrosion potential determination

```

1 clear;
2 clc;
3
4 printf("\tExample 17.2b\n");
5
6 //Activation polarisation data
7 VH_H2=0;
8 VZn_Zn2=-0.763;
9 iZn=10^-7;
10 iH2=10^-10;
11 beta_Zn=0.09;
12 beta_H2=-0.08;
13
14 ic=10^[(VH_H2-VZn_Zn2-(beta_H2*log10(iH2))+(beta_Zn*
    log10(iZn)))/(beta_Zn-beta_H2)];
15
16 //Part ii
17 Vc=VH_H2+(beta_H2*log10(ic/iH2));
18 printf("\nii) Corrosion potential is %.3f V\n",Vc);
19
20 //End

```

Scilab code Exa 17.3 Temperature Computation

```

1 //Temperature Computation
2
3 clear;
4 clc;
5
6 printf("\tExample 17.3\n");
7
8 dV=0.568; //Potential diff. b/w 2
   electrodes
9 V_Pb=-0.126;
10 V_Zn=-0.763;
11
12 C_Zn=0.01;
13 C_Pb=0.0001;
14
15 R=8.31; //Gas constt
16 F=96500; //Faraday's constt
17 n=2; //electron exchange
18
19 T=-n*F*(dV-(V_Pb-V_Zn))/(R*log(C_Zn/C_Pb));
20
21 printf("\nFinal temp is %.1f K\n",T);
22
23 //End

```

Chapter 18

Electrical Properties

Scilab code Exa 18.1 Computation of the Room Temperature Intrinsic Carrier Concentration

```
1 //Computation of the Room-Temperature Intrinsic
   Carrier Concentration for Gallium Arsenide
2
3 clear;
4 clc;
5
6 printf("\t Example 18.1\n");
7
8 sigma=10^-6; // (Ohm-m)^-1 Electrical Conductivity
9
10 e=1.6*10^-19; //Coulomb Charge on electron
11
12 m_e=0.85; //m^2/V-s Mobility of electron
13
14 m_h=0.04; //m^2/V-s Mobility of holes
15
16 //ni is Intrinsic carrier concentration
17 ni=sigma/(e*(m_e+m_h));
18
19 printf("\n Intrinsic Carrier Concentration is: %f m
   ^-3\n",ni);
```

```
20
21 //End
```

Scilab code Exa 18.2 Electrical Conductivity Determination for Intrinsic Silicon

```
1 //Electrical Conductivity Determination for
  Intrinsic Silicon at 150 C
2
3 clear;
4 clc;
5
6 printf("\t Example 18.2\n");
7
8 e=1.6*10^-19; //Coulomb Charge on electron
9
10 ni=4*10^19; //For Si at 423 K (m^-3)
11
12 //Values of m_e and m_h are deduced from graphs at
  page No.689
13
14 m_e=0.06; //m^2/V-s Mobility of electron
15
16 m_h=0.022; ///m^2/V-s Mobility of holes
17
18 //sigma is electrical conductivity
19 sigma=ni*e*(m_e+m_h);
20
21 printf("\nElectrical Conductivity is : %f (Ohm-m)
  ^-1\n",sigma);
22
23 //End
```

Scilab code Exa 18.3.b Room Temperature for Extrinsic Silicon


```

1 //Room-Temperature for Extrinsic Silicon
2
3 clear;
4 clc;
5
6 printf("\tExample 18.3\n");
7 printf("\n\tPart B\n ");
8
9 n=10^23; //m^-3 Carrier Concentration
10
11 e=1.6*10^-19; //Coulomb Charge on electron
12
13 //From graph 18.18 m_e is calculated corresponding
14 //to n=10^23
15 m_e=0.07; //m^2/V-s Mobility of electron
16
17 //For extrinsic n-type, the formula used is:
18
19 sigma=n*e*m_e;
20
21 printf("\nConductivity at n=10^23 is : %d (Ohm-m)
22 ^-1\n",sigma);
23
24 //End

```

Scilab code Exa 18.3.c Elevated Temperature Electrical Conductivity Calculations

```

1 //Elevated-Temperature Electrical Conductivity
2 //Calculations for Extrinsic Silicon
3
4 clear;
5 clc;
6

```

```

6 printf("\tExample 18.3\n");
7
8 n=10^23; //m^-3 Carrier Concentration
9
10 e=1.6*10^-19; //Coulomb Charge on electron
11
12 printf("\n\tPart C\n");
13
14 //From graph 18.19a m_e2 is calculated corresponding
    to 373 K
15
16 m_e2=0.04; //m^2/V-s Mobility of electron
17
18 sigma2=n*e*m_e2;
19
20 printf("\nConductivity at T=373 K becomes : %d (Ohm-
    m)^-1\n",sigma2);
21
22 //End

```

Scilab code Exa 18.4 Hall Voltage Computation

```

1 //Hall Voltage Computation
2
3 clear;
4 clc;
5
6 printf("\tExample 18.4\n");
7
8 sigma=3.8*10^7; //(Ohm-m)^-1 Electrical
    Conductivity
9
10 m_e=0.0012; //m^2/V-s Mobility of electron
11
12 Rh=-m_e/sigma; //Hall coefficient

```

```

13
14 printf("\nHall coefficient is : %f * 10^-11 V-m/A-
    Tesla\n", Rh/10^-11);
15
16 Ix=25; //Ampere(A) Current
17
18 d=15*10^-3; //m Thickness
19
20 Bz=0.6; //Tesla Magnetic field
21
22 Vh=Rh*Ix*Bz/d;
23
24 printf("\nHall Voltage is : %f * 10^-8 V\n", Vh
    /10^-8);
25
26 //End

```

Scilab code Exa 18.6 Acceptor Impurity Doping in Silicon

```

1 //Acceptor Impurity Doping in Silicon
2
3 clear;
4 clc;
5
6 printf("\tDesign Example 18.1\n");
7
8 e=1.6D-19; //Charge on electron in
    Coulomb
9 p=[10^22 10^21 8D21]; //Conc. of holes m^-3
10 uh=[0.04 0.045 0.04]; //Mobility of holes
11
12 for i=1:3
13     sigma(i)=p(i)*e*uh(i);
14 end
15

```

```

16 disp('(Ohm-m)^-1',sigma(3),'conductivity is ','m^2/V-
    s',uh(3),'m^-3',p(3),'For hole conc and mobility'
    );
17
18 Na=6.023D23;           //Avogadro no.
19 den_Si=2.33D6;        //Density of silicon in g/
    m^3
20 A_Si=28.09;           //Atomic weight of silicon
21
22 N_Si=Na*den_Si/A_Si;
23
24
25 Ca=p(3)/(p(3)+N_Si)*100;
26
27 printf("\nThus, Silicon material must contain %.1f *
    10^-5 %% B,Al,Ga or In\n",Ca/10^-5);
28
29 //End

```

Chapter 19

Thermal Properties

Scilab code Exa 19.1 Thermal Stress Created Upon Heating

```
1 // Calculation of maximum temperature
2
3 clear;
4 clc;
5
6 printf("\t Example 19.1\n");
7
8 To=20; // Room Temperature (degree celsius)
9
10 sigma=-172; //Mpa Compressive stress
11
12 E=100*10^3; //Mpa Young's modulus
13
14 a=20*10^-6; //Celsius^-1 Coefficient of thermal
    expansion
15
16 Tf=To-(sigma/(E*a));
17
18 printf("\nFinal Temperature is : %d C\n",Tf);
19
20 //End
```

Scilab code Exa 19.2 Final temperature Calculation

```
1 //Final temperature Calculation
2
3 clear;
4 clc;
5
6 printf("\tExample 19.2\n");
7
8 m=10;           //mass in lbm
9 dQ= 65;        //Heat supplied in Btu
10 To=77;         //Initial temp in F
11
12 Cp=375*2.39*10^-4; //in Btu/lbm - F
13
14 dT=dQ/(m*Cp);
15
16 Tf=To+dT;
17
18 printf("\nFinal temp is %.1f F\n",Tf);
19
20 //End
```

Scilab code Exa 19.3 Computation of specific heats for Al and Fe

```
1 //Computation of specific heats for Al and Fe
2
3 clear;
4 clc;
5
6 printf("\tExample 19.3\n");
```

```

7
8 Cp_Al=900;           //In J/kg-K
9 Cp_Fe=448;          //In j/kg-K
10
11 beta_Al=1.77D-11;
12 beta_Fe=2.65D-12;
13
14 T=273;
15
16 alphas_Al=23.6D-6;
17 alphas_Fe=11.8D-6;
18
19 alphav_Al=3*alphas_Al;
20 alphav_Fe=3*alphas_Fe;
21
22 den_Al=2.71D3;       //in kg/m^3
23 den_Fe=7.87D3;       //in kg/m^3
24
25 vo_Al=1/den_Al;
26 vo_Fe=1/den_Fe;
27
28 Cv_Al=Cp_Al-(alphav_Al^2*vo_Al*T/beta_Al);
29 Cv_Fe=Cp_Fe-(alphav_Fe^2*vo_Fe*T/beta_Fe);
30
31 printf("\nCv (Al) = %d J/kg-K", Cv_Al);
32 printf("\nCv (Fe) = %d T/kg-K\n", Cv_Fe);
33
34 //End

```

Chapter 20

Magnetic Properties

Scilab code Exa 20.1.a Saturation Magnetization

```
1 //Example 20.1 Calculation of saturation
   magnetisation and flux density for Nickel
2
3 clear;
4 clc;
5
6 printf("Example 20.1\n");
7
8 b_m=9.27*10^-24; //ampere*m^2 (Bohr Magneton)
9
10 Na=6.023*10^23; //atoms/mol (Avogadro's No.)
11
12 d=8.9*10^6; //g/m^3 (density)
13
14 uo=4*pi*10^-7; //Permittivity of free space
15
16 A=58.71; //g/mol (Atomic weight of Nickel)
17
18 N=d*Na/A; //No. of atoms per cubic meter
19
20 // M is saturation magnetisation
```



```

21 M=0.6*b_m*N; //0.6= Bohr Magnetron/atom
22
23 printf("\nSaturation Magnetisation is : %d Ampere/m"
24         ,M);
25
26
27 // End

```

Scilab code Exa 20.1.b Flux Density Computations for Nickel

```

1 //Example 20.1 Calculation of saturation
2 magnetisation and flux density for Nickel
3
4 clear;
5 clc;
6
7 printf("Example 20.1\n");
8
9 b_m=9.27*10^-24; //ampere*m^2 (Bohr Magnetron)
10
11 Na=6.023*10^23; //atoms/mol (Avogadro's No.)
12
13 d=8.9*10^6; //g/m^3 (density)
14
15 uo=4*pi*10^-7; //Permittivity of free space
16
17 A=58.71; //g/mol (Atomic weight of Nickel)
18
19 N=d*Na/A; //No. of atoms per cubic meter
20
21 // M is saturation magnetisation
22 M=0.6*b_m*N; //0.6= Bohr Magnetron/atom
23

```

```

24 //B = Saturation Flux Density
25 B=uo*M;
26
27 printf("\nSaturation Flux Density is : %f Tesla\n",B
    );
28
29 // End

```

Scilab code Exa 20.2 Saturation Magnetization Determination

```

1 //Example 20.2 Calculation of saturation
  magnetisation of Fe3O4
2
3 clear;
4 clc;
5
6 printf("Example 20.2\n");
7
8 a=0.839*10^-9; //a is edge length in m
9
10 b_m=9.27*10^-24; //ampere*m^2 (Bohr Magneton)
11
12 nb=8*4; //8 is no. of Fe++ ions per unit cell
13 //4 is Bohr magnetons per Fe++ ion
14
15 M=nb*b_m/a^3; //M is Saturation magnetisation
16
17 printf("\nSaturation Magnetisation is : %f Ampere/m\n
    n",M);
18
19 //End

```

Scilab code Exa 20.3 Design of a Mixed Ferrite Magnetic Material

```

1 //Design Example 20.1: Designing a cubic mixed-
  ferrite magnetic material
2
3 clear;
4 clc;
5
6 printf(" Design Example 20.1\n");
7
8 x=poly([0], 'x'); // Defining X
9
10 Ms_Fe=5.25*10^5; //Required saturation
  Magnetisation
11
12 b_m=9.27*10^-24; //ampere*m^2 (Bohr Magneton)
13
14 a=0.839*10^-9; //a is edge length in m
15
16 M=5*10^5; //From previous question result
17
18 nb=Ms_Fe*a^3/b_m;
19
20 // 'x' represent fraction of Mn++ that have
  substituted Fe++
21
22 n=roots(8*[5*x+4*(1-x)]-nb); //5 is Bohr magnetons
  per Fe++ ion
23
  //4 is Bohr magnetons
  per Mn++ ion
24
25 printf("\nReplacing %f percent of Fe++ with Mn++
  would produce the required saturation
  magnetisation\n",n*100);
26
27 //End

```

Chapter 21

Optical Properties

Scilab code Exa 21.1 Computation of the Absorption Coefficient for Glass

```
1 //Example 20.1 Calculation of absorption coefficient
2
3 clear;
4 clc;
5
6 printf("\tExample 21.1\n");
7
8 // x is thickness of glass (mm)
9 x=200;
10
11 //It is intensity of non-absorbed radiation
12 //Io is intensity of non-relected radiation
13
14 f=0.98; //f=It/Io
15
16 //b is absorption coefficient
17
18 b=-log(f)/x;
19
20 printf("\nAbsorption coefficient is %f mm-1\n",b);
21
```

22 //End

Scilab code Exa 21.2 Velocity of light in diamond

```
1 //Velocity of light in diamond
2
3 clear;
4 clc;
5
6 printf("\tExample 21.2\n");
7
8 er=5.5; //Relative permittivity
9 xm=-2.17D-5; //Magnetic Suseptibility
10
11 eo=8.85D-12; //Permittivity in free
    space
12 uo=4*pi*10^-7; //Permeability
13
14 e=er*eo;
15 u=uo*(1+xm);
16
17 v=1/sqrt(u*e);
18
19 printf("\nVelocity in diamond is %.2e m/s\n",v);
20
21 //End
```

Scilab code Exa 21.3 Suitable material required

```
1 //Suitable material required
2
3 clear;
4 clc;
```

```

5
6 printf("\tExample 21.3\n");
7
8 R=0.05;
9 n=poly([0], 'x');
10
11 //R = (ns-1/ns+1)^2 leading to the equation
12 //      0.95ns^2-2.1ns+0.95 = 0
13 //Hence
14
15 ns=roots((0.95*n^2)-(2.1*n)+0.95);
16
17 disp(ns, 'The values of ns are: ');
18 printf("Thus, soda-lime glass, Pyrex glass, and
        polyprop would be suitable for this application
        .");
19
20 //End

```
