Scilab Textbook Companion for Solid State Physics: Structure And Properties Of Materials by M. A. Wahab[1]

Created by
Pankaj Biswas
Electronics
Physics
Shri Mata Vaishno Devi University
College Teacher
Dr. Kamni
Cross-Checked by
Dr. Jitendra Sharma

August 10, 2013

[1]Funded 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: Solid State Physics: Structure And Properties Of Materials

Author: M. A. Wahab

Publisher: Narosa Publishing House Pvt. Ltd. New Delhi

Edition: 2

Year: 2010

ISBN: 978-81-7319-603-4

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

Chapter 1

Atoms in Crystals

Scilab code Exa 1.1 Relationship among cyrstal elements

```
1 // Scilab Code Ex1.1 Relationship among cyrstal
    elements: Page-2 (2010)
2 f = 18;     // Number of faces of the quartz crystal
3 c = 14;     // Number of angles in the quratz crystal
4 // The relationship amongst the crystal elements can
     be
5 // expressed by the following formula:
6 //     f + c = e + 2;
7 // Solving for e
8 e = f + c - 2;
9 disp (e, "The number of edges of the quartz crystal
    is : ");
10
11 // Result
12 // The number of edges of the quartz crystal is :
13 //     30
```

Scilab code Exa 1.2 Primitive unit cell

```
// Scilab Code Ex1.2 Primitive unit cell: Page-4 (2010)
a = 3, b = 3;      // Lattice translation vectors along X and Y direction, angstrom
c_bar = 3;     // Assumed translation vector along Z direction, angstrom
c = 1.5*(a+b+c_bar);     // Real translation vector along Z direction, angstrom
printf("\n%3.1f is the body centered position of a cubic unit cell defined by the primitive translation vectors a, b and c_bar.", c);
V_con = a^3;     // Volume of conventional unit cell, metre cube
V_primitive = 1/2*V_con;     // Volume of primitive unit cell, metre cube
printf("\nThe volume of conventional unit cell: %2d angstrom cube", V_con);
printf("\nThe volume of primitive unit cell: %4.1f angstrom cube", V_primitive);

// Result
// 13.5 is the body centered position of a cubic unit cell defined by the primitive translation vectors a, b and c_bar.
// The volume of conventional unit cell: 27 angstrom cube
// The volume of primitive unit cell: 13.5 angstrom cube
```

Scilab code Exa 1.3 Number of Lattice points per unit cell

```
// Scilab Code Ex1.3 Number of Lattice points per unit cell Page-9 (2010)
a = 3.60D-10;     // Lattice parameter, m:
M = 63.6;         // Atomic weight, gram per mole
```

```
4 d = 8960D+03;     // Density of copper, g per metre
    cube
5 N = 6.023D+23;     // Avogadro's No.
6 // Volume of the unit cell is given by
7 // a^3 = M*n/(N*d)
8 // Solving for n
9 n = a^3*d*N/M; // Number of lattice points per unit
    cell
10 disp (n, "The number of atoms per unit cell for an
    fcc lattice of copper crystal is :");
11
12 // Result
13 // The number of atoms per unit cell for an fcc
    lattice of copper crystal
14 // 3.9588702
```

Scilab code Exa 1.4 Lattice constant of a unit cell

```
1 // Scilab Code Ex 1.4 Lattice constant of a unit
    cell: Page-9 (2010)
2 M = 58.5;          // Atomic weight of NaCl, gram per
    mole
3 d = 2180D+03;     // Density of rock salt, per metre
    cube
4 n = 4;     // No. of atoms per unit cell for an fcc
    lattice of NaCl crystal
5 N = 6.023D+23;     // Avogadro's No.
6 // Volume of the unit cell is given by
7 // a^3 = M*n/(N*d)
8 // Solving for a
9 a = (n*M/(d*N))^(1/3);     // Lattice constant of
    unit cell of NaCl
10 disp (a/1D-10, "Lattice constant for the rock salt (
    NaCl) crystal, in angstrom, is : ");
11
```

```
12 // Result
13 // Lattice constant for the rock salt (NaCl) crystal
       , in angstrom, is :
14 // 5.6275
```

Scilab code Exa 1.5 Density of diamond

```
1 // Scilab Code Ex 1.5 Density of diamond: Page-9
      (2010)
2 a = 3.57D-10; // Lattice parameter of a diamond
      crystal
3 M = 12D-03;      // Atomic weight of diamond, kg per
      mole
4 n1 = 8;     // No. of corner atoms in the diamond
      cubic unit cell
5 n2 = 6;     // No. of face centered atoms in the
      diamond cubic unit cell
6 n3 = 4;     // No. of atoms completely within the
      unit cell
7 n = 1/8*n1+1/2*n2+1*n3;     // No. of atoms per unit
      cell for an fcc lattice of NaCl crystal
8 N = 6.023D+23;     // Avogadro's No.
9 // Volume of the unit cell is given by
10 // a^3 = M*n/(N*d)
11 // Solving for d
12 d = M*n/(N*a^3);     // Density of diamond cubic unit
       cell
13 disp (round(d), "Density of diamond cubic unit cell,
       in kg per metre cube, is : ");
14
15 // Result
16 // Density of diamond cubic unit cell, in kg per
      metre cube, is :
17 // 3503
```

Scilab code Exa 1.6 Calculating Unit cell dimensions

```
1 // Scilab Code Ex 1.6 Calculating Unit cell
    dimensions: Page-9 (2010)
2 d = 2.7D+03;     // Density of fcc structure of
    aluminium, kg per metre cube
3 M = 26.98D-03;      // Atomic weight of aluminium, kg
     per mole
4 n = 4;        // No. of atoms per unit cell of fcc
    lattice structure of  aluminium
5 N = 6.023D+23;     // Avogadro's No.
6 // Volume of the unit cell is given by
7 // a^3 = M*n/(N*d)
8 // Solving for a
9 a = ((M*n)/(N*d))^(1/3);     // Lattice parameter of
    alumitnium unit cell
10 // For an fcc cryatal lattice,
11 // 2^(1/2) = 4R = 2D
12 // Solving for D
13 D = (a/2^(1/2)); // Diameter of aluminium atom
14 disp (a/1D-10, "The Lattice parameter of aluminium,
    in angstrom, is : ");
15 disp(D/1D-10, "The diameter of aluminium atom, in
    angstrom, is : ");
16
17 // Result
18 // The Lattice parameter of aluminium, in angstrom,
    is :
19 //      4.0486332
20 // The diameter of aluminium atom, in angstrom, is :
21 //      2.862816
```

Scilab code Exa 1.17 Angle between two crystal directions

```
1 // Scilab Code Ex 1.17 Angle between two crystal
    directions: Page-23 (2010)
2 h1 = 1;k1 = 1;l1 = 1; // Miller indices of first set
     of planes
3 h2 = 0;k2 = 0;l2 = 1; // Miller indices of second
    set of planes
4 // We know that
5 // cos(theta) = (h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+
    l1^2)*sqrt(h1^2+k1^2+l1^2))
6 // Solving for theta
7 theta = acos((h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1
    ^2)*sqrt(h2^2+k2^2+l2^2)));
8 printf("\nThe angle between [%d%d%d] and [%d%d%d]
    directions in the cubic crystal, in degrees, is :
     %4.2f", h1,k1,l1,h2,k2,l2, theta*180/%pi);
9
10 // Result
11 // The angle between [111] and [001] directions in
    the cubic crystal, in degrees, is :
12 //      54.74
```

Scilab code Exa 1.18 Angle between two directions of cubic crystal

```
1 // Scilab Code Ex 1.18 Angle between two directions
    of cubic crystal: Page-23(2010)
2 h1 = 1; k1 = 1; l1 = 1 // Miller indices for first
    set of planes
3 h2 = -1; k2 = -1; l2 = 1;     // Miller indices for
    second set of planes
4 // We know that
5 // cos(theta) = (h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+
    l1^2)*sqrt(h2^2+k2^2+l2^2))
6 // Solving for theta
```

```
7 theta = acos((h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1
    ^2)*sqrt(h2^2+k2^2+l2^2)));
8 printf("\nThe angle between [%d%d%d] and [%d %d %d]
    directions in the cubic crystal, in degrees, is :
     %4.1f", h1,k1,l1,h2,k2,l2, theta*180/%pi);
9
10 // Result
11 // The angle between [111] and [-1-1 1] directions
    in the cubic crystal, in degrees, is :
12 //      109.5
```

Scilab code Exa 1.19 Miller indices of the crystal plane

```
1 // Scilab Code Ex 1.19 Miller indices of the crystal
     plane: Page-25 (2010)
2 m = 2; n = 3; p = 6; // Coefficients of intercepts
    along three axes
3 m_inv = 1/m;          // Reciprocate the first
    coefficient
4 n_inv = 1/n;          // Reciprocate the second
    coefficient
5 p_inv = 1/p;          // Reciprocate the third
    coefficient
6 mul_fact = double(lcm(int32([m,n,p]))); // Find l.c.
    m. of m,n and p
7 m1 = m_inv*mul_fact;     // Clear the first fraction
8 m2 = n_inv*mul_fact;     // Clear the second fraction
9 m3 = p_inv*mul_fact;     // Clear the third fraction
10 printf("\nThe required miller indices are : (%d %d
    %d) ", m1,m2,m3);
11
12 // Result
13 // The required miller indices are : (3 2 1)
```

Scilab code Exa 1.20 Indices of lattice plane

```
1 // Scilab Code Ex 1.20 Indices of lattice plane:
    Page-25 (2010)
2 m = 10000; // Coefficient of intercept along x-axis,
    can be taken as some large value
3 n = 2;    // Coefficient of intercept along y-axis
4 p = 1/2;    // Coefficient of intercept along z-axis
5 m_inv = 1/m;    // Reciprocate m
6 n_inv = 1/n;    // Reciprocate n
7 p_inv = 1/p;    // Reciprocate p
8 mul_fact = n;    // multiplicative factor
9 m1 = m_inv*mul_fact;    // Clear the first fraction
10 m2 = n_inv*mul_fact;    // Clear the second fraction
11 m3 = p_inv*mul_fact;    // Clear the third fraction
12 printf("\nThe required miller indices are : %d, %d,
    %d ", m1,m2,m3);
13
14 // Result
15 // The required miller indices are :
16 //     0, 1, 4
```

Scilab code Exa 1.21 Length of the intercepts

```
1 // Scilab Code Ex 1.21 Length of the intercepts:
    Page-26 (2010)
2 a = 1.21D-10;    // Lattice parameter of the unit
    cell, m
3 b = 1.84D-10;    // Lattice parameter of the unit
    cell, m
4 c = 1.97D-10;    // Lattice parameter of the unit
    cell, m
```

```
5 p = 1/2;     // Reciprocal of miller index on x-axis
6 q = 1/3;     // Reciprocal of miller index on y-axis
7 r = 1/(-1);     // Reciprocal of miller index on z-
    axis
8 l1 = 1.21D-10;     // Actual length of the intercept
    along x-axis, m
9 mul_fact = l1/(p*a); // Calculate multiplication
    factor
10 l2 = mul_fact*q*b;     // Actual length of the
    interceptalong y-axis, m
11 l3 = mul_fact*r*c;     // Actual length of the
    intercept along z-axis, m
12 disp(l2/1D-10, "The length of the intercept along y-
    axis, in angstrom, is : ");
13 disp(l3/1D-10, "The length of the intercept along z-
    axis, in angstrom, is : ");
14
15 // Result
16 // The length of the intercept along y-axis, in
    angstrom, is :
17 //      1.2266667
18 // The length of the intercept along z-axis, in
    angstrom, is :
19 //    - 3.94
```

Scilab code Exa 1.22 Miller indices of lattice planes

```
1 // Scilab Code Ex 1.22 Miller indices of lattice
    plane: Page-26 (2010)
2 a = 4;     // Lattice parameter of the unit cell
3 b = 3;     // Lattice parameter of the unit cell
4 c = 2;     // Lattice parameter of the unit cell
5 l1 = 2;     // Length of the intercept along x-axis,
    m
6 l2 = 3;     // Length of the intercept along y-axis,
```

```
    m
l3 = 4;     // Length of the intercept along z-axis,
    m
l = l1/a;     // Intercept per unit translation along
    x-axis
m = l2/b;     // Intercept per unit translation along
    y-axis
n = l3/c;     // Intercept per unit translation along
    z-axis
r1 = 1/l;     // Reciprocal of l
r2 = 1/m;     // Reciprocal of m
r3 = 1/n;     // Reciprocal of n
m1 = 2*r1;     // miller index along x-axis
m2 = 2*r2;     // miller index along y-axis
m3 = 2*r3;     // miller index along z-axis
printf("The required miller indices of the plane are
    : %d %d %d", m1, m2, m3);

// Result
// The required miller indices of the plane are :
//     4, 2, 1
```

Scilab code Exa 1.23 Indices of tetragonal lattice

```
// Scilab Code Ex 1.23 Indices of tetragonal lattice
   : Page-26 (2010)
// For a tetragonal system we have a = b
a = 1;     // Lattice parameter of the unit cell
   along x-axis
b = 1;     // Lattice parameter of the unit cell
   along y-axis
c = 1.5;     // Lattice parameter of the unit cell
   along z-axis
l1 = 3;     // Length of the intercept along x-axis,
   angstrom
```

```
7  l2 = 4;     // Length of the intercept along y-axis,
     angstrom
8  l3 = 3;     // Length of the intercept along z-axis,
     angstrom
9  l = l1/a;     // Intercept per unit translation along
      x-axis
10 m = l2/b;     // Intercept per unit translation along
      y-axis
11 n = l3/c;     // Intercept per unit translation along
      z-axis
12 r1 = 1/l;     // Reciprocal of l
13 r2 = 1/m;     // Reciprocal of m
14 r3 = 1/n;     // Reciprocal of n
15 mul_fact = double(lcm(int32([l,m,n])));
16 m1 = mul_fact*r1;     // miller index along x-axis
17 m2 = mul_fact*r2;     // miller index along y-axis
18 m3 = mul_fact*r3;     // miller index along z-axis
19 printf("The required miller indices of the plane are
     : %d %d %d", m1, m2, m3);
20
21 // Result
22 // The required miller indices of the plane are : 4
     3 6
```

Scilab code Exa 1.24 Miller Bravias indices for Miller indices

```
1  // Scilab Code Ex 1.24 Miller-Bravias indices for
     Miller indices: Page-29 (2010)
2  function [i] = f(h,k)
3      i = -(h + k);
4  endfunction
5  h1 = 1; k1 = 1; l1 = 0 ;  // First set of Miller
     indices
6  h2 = 1; k2 = -1; l2 = 0;     // Second set of miller
     indices
```

```
7 h3 = 3; k3 = 4; l3 = 5;     // Third set of miller
    indices
8 h4 = 3; k4 = -4; l4 = 5;     // Fourth set of miller
    indices
9 printf("\nThe Miller-Bravias indices corresponding
    to the miller indices (%d %d %d), = (%d %d %d %d)
    ", h1, k1, l1, h1, k1, f(h1,k1), l1);
10 printf("\nThe Miller-Bravias indices corresponding
    to the miller indices (%d %d %d), = (%d %d %d %d)
    ", h2, k2, l2, h2, k2, f(h2,k2), l2);
11 printf("\nThe Miller-Bravias indices corresponding
    to the miller indices (%d %d %d), = (%d %d %d %d)
    ", h3, k3, l3, h3, k3, f(h3,k3), l3);
12 printf("\nThe Miller-Bravias indices corresponding
    to the miller indices (%d %d %d), = (%d %d %d %d)
    ", h4, k4, l4, h4, k4, f(h4,k4), l4);
13
14 // Result
15 // The Miller-Bravias indices corresponding to the
    miller indices (1 1 0), = (1 1 -2 0)
16 // The Miller-Bravias indices corresponding to the
    miller indices (1 -1 0), = (1 -1 0 0)
17 // The Miller-Bravias indices corresponding to the
    miller indices (3 4 5), = (3 4 -7 5)
18 // The Miller-Bravias indices corresponding to the
    miller indices (3 -4 5), = (3 -4 1 5)
```

Scilab code Exa 1.25 Miller Bravias indices of lattice plane

```
1 // Scilab Code Ex 1.25 Miller Bravias indices of
    lattice planes: Page-30 (2010)
2 function [h] = fh(H,K)     // Function for
    calculating (2H-K)/3
3     h = (2*H - K)/3;
4 endfunction
```

```
5
6  function [k] = fk(H,K)     // Function for
       calculating (2K-H)/3
7      k = (2*K - H)/3;
8  endfunction
9
10 function [i] = f(h,k)     // Function for calculating
       i
11     i = -(h + k);
12 endfunction
13
14 function [l] = fl(L)     // Function for calculating
       l
15     l = L;
16 endfunction
17
18 H1 = 1; K1 = 0; L1 = 0 ;  // First set of Miller
       indices
19 H2 = 0; K2 = 1; L2 = 0;     // Second set of miller
       indices
20 H3 = 1; K3 = 1; L3 = 0;     // Third set of miller
       indices
21
22 h1 = fh(H1,K1)*3;      // Call function fh
23 k1 = fk(H1,K1)*3;      // Call function fk
24 l1 = fl(L1)*3;         // Call function fl
25 i1 = f(h1,k1);         // Call function
26
27 h2 = fh(H2,K2)*3;      // Call function fh
28 k2 = fk(H2,K2)*3;      // Call function fk
29 l2 = fl(L2)*3;          // Call function l2
30 i2 = f(h2,k2);          // Call function f
31
32 h3 = fh(H3,K3)*3;      // Call function fh
33 k3 = fk(H3,K3)*3;      // Call function fk
34 l3 = fl(L3)*3;         // Call function l3
35 i3 = f(h3,k3);          // Call function f
36
```

```
37 printf("\n The Miller Bravias indices of [%d%d%d]
      are [%d %d %d %d]", H1, K1, L1, h1, k1,i1,l1);
38 printf("\n The Miller Bravias indices of [%d%d%d]
      are [%d %d %d %d]", H2, K2, L2, h2, k2,i2,l2);
39 printf("\n The Miller Bravias indices of [%d%d%d]
      are [%d %d %d %d]", H3, K3, L3, h3, k3,i3,l3);
40
41 // Result
42 // The Miller Bravias indices of [100] are [2 -1 -1
      0]
43 // The Miller Bravias indices of [010] are [-1 2 -1
      0]
44 // The Miller Bravias indices of [110] are [1 1 -2
      0]
```

Scilab code Exa 1.26 Lattice parameter of a cubic crystal

```
1 // Scilab Code Ex 1.26 Lattice parameter of a cubic
     crystal: Page-33 (2010)
2 h = 1; k = 1; l = 1; // Miller Indices for planes in
     a cubic crystal
3 d = 2D-10;     // Interplanar spacing, m
4 // For cubic crystals, the interplanar spacing is
     given by
5 // d = a/(h^2+k^2+l^2)^1/2;
6 // Solving for a
7 a = (h^2+k^2+l^2)^(1/2)*d;    // lattice parameter of
     cubic crystal, m
8 disp(a/1D-10, "The lattice parameter of the cubic
     crystal, in angstrom, is :");
9
10 // Result
11 // The lattice parameter of the cubic crystal, in
     angstrom, is :
12 // 3.4641016
```

Scilab code Exa 1.27 Interplanar spacing in tetragonal crystal

```
1 // Scilab Code Ex 1.27 Interplanar spacing in
    tetragonal crystal: Page-33 (2010)
2 h = 1; k = 0; l = 1; // Miller Indices for planes in
     a cubic crystal
3 a = 2.42D-10;    b = 2.42D-10;    c = 1.74D-10; //
    Lattice parameters of a tetragonal crystal, each
    in m
4 d = [(h^2+k^2)/a^2 + l^2/c^2]^(-1/2); // The
    interplanar spacing for cubic crystal, m
5 disp(d/1D-10, "The interplanar spacing between
    consecutive (101) planes : in angstrom, is :");
6
7 // Result
8 // The interplanar spacing between consecutive (101)
     planes : in angstrom, is :
9 // 1.4127338
```

Scilab code Exa 1.28 Interplanar spacing in cubic crystal

```
1 // Scilab Code Ex 1.28 Interplanar spacing in cubic
    crystal: Page-36 (2010)
2 h = 3; k = 2; l = 1; // Miller Indices for planes in
     a cubic crystal
3 a = 4.21D-10;     // Interatomic spacing, m
4 d = a/(h^2+k^2+l^2)^(1/2);  // The interplanar
    spacing for cubic crystals, m
5 disp(d/1D-10, "The interplanar spacing between
    consecutive (321) planes : in angstrom, is :");
6
```

```
7 // Result
8 // The interplanar spacing between consecutive (321)
      planes : in angstrom, is :
9 // 1.1251698
```

Chapter 2

Atomic Bonding

Scilab code Exa 2.1 Molecular stability based on bond dissociation energy

```
1 // Scilab Code Ex2.1 Stability of molecule based on
    bond dissociation energy: Page-61 (2010)
2 e = 1.6D-19; // Electronic charge, C
3 N = 6.023D+23; // Avogadro's number
4 e0 = 8.854D-12; // Absolute Electrical permitivitty
    of free space, coulomb square per newton per
    metre square
5 Re = 3D-10;     // Equilibrium separation, m
6 IE = 502;     // First ionization energy of A, kJ/mol
7 EA = 335;     // Electron affinity for atom B, kJ/mol
8 IS = 3D-10;     // Interatomic separation between A+
    and B-, m
9 Ue = -(e^2*N)/(4*%pi*e0*Re*1D+3); // Potential
    energy at equilibrium separation of A+B- molecule
    , kJ/mol
10 DE = Ue + IE - EA; // Bond dissociation energy of A+
    B- molecule, kJ/mol
11 printf("\nThe bond dissociation energy of A+B-
    molecule is : %d kJ/mol", DE);
12 if (DE < 0)
13     disp("The molecule A+B- is stable..");
```

```
14 else
15     disp("The molecule A+B- is unstable..");
16 end
17
18 //Result
19 //     The bond dissociation energy of A+B- molecule,
       in kJ/mol, is : -294
20 //    The molecule A+B- is stable..
```

Scilab code Exa 2.2 Conversion of eV into kcal per mol

```
1 // Scilab Code Ex2.2 Conversion of eV into kcal/mol:
      Page-64 (2010)
2 e = 1.6D-19; // Electronic charge, C
3 N = 6.023D+23; // Avogadro's number
4 J = 4.184D+3; // Joule's mechanical equivalent of
     heat
5 V = 1;     // Potential difference, V
6 eV = e*V; // Energy equivalent of 1 electron-volt, J
7 eVpm = eV*N; // Electron-volt per mole, J/mol
8 Ecal = eVpm/J; // Energy equivalent of 1eV, kcal/
     mole
9 printf("\n1 eV is approximately equal to %6.3f kcal/
     mol", Ecal);
10
11 //Result
12 //     1 eV is approximately equal to 23.033 kcal/mol
```

Scilab code Exa 2.3 Potential energy of the ionic solids

```
1 // Scilab Code Ex2.3 Potential energy of the system
     of Na+ and Cl- ions: Page-68 (2010)
2 e = 1.6D-19; // Electronic charge, C
```

```
3 ep_0 = 8.854D-12; // Absolute electrical
    permittivity of free space, coulomb square per
    newton per metre square
4 Re = 2D-10; // Equilibrium separation between Na+
    and Cl- ions, m
5 U = -e/(4*%pi*ep_0*Re); // Potential energy of NaCl
    molecule at equilibrium separation, electron-volt
6 printf("\nThe potential energy of NaCl molecule at
    equilibrium separation5 is : %3.1f eV", U);
7
8 //Result
9 // The potential energy of NaCl molecule at
    equilibrium separation5 is : -7.2 eV
```

Scilab code Exa 2.4 Compressibility and energy of ionic crystal

```
1 // Scilab Code Ex2.4 Compressibility and ionic
    energy of NaCl crystal: Page-68 (2010)
2 e = 1.6D-19; // Electronic charge, C
3 ep_0 = 8.854D-12; // Absolute electrical
    permittivity of free space, coulomb square per
    newton per metre square
4 Re = 2.81D-10; // Equilibrium separation between Na+
     and Cl- ions, m
5 A = 1.7496; // Madelung constant
6 n = 9; // Power of R in the repulsive term of
    potential energy of two particles
7 IP_Na = 5.14; // Ionization potential of sodium, eV
8 EA_Cl = 3.61; // Electron Affinity of chlorine, eV
9 K0 = (72*%pi*ep_0*Re^4)/((n - 1)*A*e^2); //
    Compressibilty of NaCl crystal, metre square
    newton
10 U = -(A*e)/(4*%pi*ep_0*Re)*(1-1/n); // Potential
    energy of NaCl molecule at equilibrium separation
    , electron-volt
```

```
11 U_bar = U/2; // Potential energy per ion, electron-
     volt
12 delta_E = IP_Na - EA_Cl; // Energy required to
     produce the ion-pair, eV
13 E_ion = delta_E/2; // Energy required to produce per
     ion, eV
14 C_E = U_bar + E_ion; // Cohesive energy per ion, eV
15 printf("\nThe compressibility of NaCl crystal is %4
     .2e metre square newton", K0);
16 printf("\nThe cohesive energy of NaCl crystal is %4
     .2f eV", C_E);
17
18 // Result
19 // The compressibility of NaCl crystal is 3.48e-011
     metre square newton
20 // The cohesive energy of NaCl crystal is -3.21 eV
```

Scilab code Exa 2.5 Potential energy and dissociation energy of a diatomic molecule

```
1 // Scilab Code Ex2.5 Potential energy and
     dissociation energy of a diatomic molecule: Page
     -69 (2010)
2 e = 1.6D-19; // Electronic charge, C
3 A = 1.44D-39; // Constant corrsponding to the
     attractive term in potential energy, joule metre
     square
4 B = 2.19D-115; // Constant corresponding to the
     repulsive term in potential energy, joule metre
     raised to power 10
5 Re = (5*B/A)^(1/8); // Equilibrium spacing of
     diatomic molecule, m
6 n = 2;  // Power of R in the attractive term of
     potential energy of two particles
7 m = 10; // Power of R in the repulsive term of
```

```
    potential energy of two particles
D = A/(Re^2*e)*(1-n/m);// Dissociation energy of
    diatomic molecule, eV
printf("\nThe equilibrium spacing of diatomic
    molecule is %4.2e m", Re);
printf("\nThe dissociation energy of diatomic
    molecule is %4.2e eV", D);

//Result
// The equilibrium spacing of diatomic molecule is
    4.08e-010 m
// The dissociation energy of diatomic molecule is
    4.34e-002 eV
```

Scilab code Exa 2.6 Binding force and critical separation of a diatomic molecule

```
// Scilab Code Ex2.6 Binding force and critical
    separation of a diatomic molecule: Page-69 (2010)
Re = 3D-10; // Equilibrium spacing of diatomic
    molecule, m
e = 1.6D-19; // Electronic charge, C
D = 4*e; // Dissociation energy of diatomic molecule
    , eV
n = 2;  // Power of R in the attractive term of
    potential energy of two particles
m = 10; // Power of R in the repulsive term of
    potential energy of two particles
Ue = -D; // Potential energy of diatomic molecule at
     equilibrium separation, joule
A = -(Ue*Re^n)/(1-n/m); // Constant corrsponding to
    the attractive term in potential energy, joule
    metre square
B = A*Re^8/5; // Constant corresponding to the
    repulsive term in potential energy, joule metre
```

```
    raised to power 10
10 Rc = (55/3*B/A)^(1/8); // Critical separation
    between the nuclei, m
11 F_min = -2*A/Rc^3*(1-(Re/Rc)^8); // The minimum
    force required to dissociate the moleule, N
12 disp(A,"The constant A corresponding to the
    attractive potential energy, in joule metre
    square, is :");
13 disp(B,"The constant B corresponding to the
    repulsive potential energy, in joule metre raised
     to power 10, is :");
14 disp(Rc/1d-10, "The critical separation between the
    nuclei, in angstrom, is : ");
15 disp(F_min, "The minimum force required to
    dissociate the molecule, in N, is : ");
16
17 //Result
18 // The constant A corresponding to the attractive
    potential energy, in joule metre square, is :
                    //   7.200D-38
19 // The constant B corresponding to the repulsive
    potential energy, in joule metre raised to power
    10, is :    //   9.44D-115
20 // The critical separation between the nuclei, in
    angstrom, is :
21 // 3.529D-10
22 // The minimum force required to dissociate the
    molecule, in N, is :
23 // -2.383D-09
```

Scilab code Exa 2.7 Bond formation energy of ionic solid

```
1 // Scilab Code Ex2.7  Bond formation Energy for K+
    and Cl- ion pair: Page-70 (2010)
2 eps_0 = 8.854D-12; // Absolute electrical
```

```
    permittivity of free space, coulomb sqaure per
    newton per metre square
e = 1.6D-19; // Electronic charge, C
IP_K = 4.1; // Ionization potential of potassium,
    electron-volt
EA_Cl = 3.6; // Electron affinity of chlorine,
    electron-volt
delta_E = IP_K - EA_Cl; // Net energy required to
    produce the ion-pair, electron-volt
Ec = delta_E; // Coulomb energy equals net energy
    required to produce the ion pair, in electron-
    volt
// Since Ec = -e/(4*%pi*eps_0*R), solving for R
R = -e/(4*%pi*eps_0*Ec); // Separation between K+
    and Cl- ion pair, m
disp(Ec,"The bond formation energy for K+ and Cl-
    ion pair, in eV, is : ");
disp(R/1D-10, "The separation between K+ and Cl- ion
     pair, in angstrom, is : ");

//Result
// The bond formation energy for K+ and Cl- ion pair
    , in eV, is :
// 0.5
// The separation between K+ and Cl- ion pair, in
    angstrom, is :
// - 28.760776
```

Scilab code Exa 2.8 Energy liberation during electron transfer

```
// Scilab Code Ex2.8  Energy liberated during
    electron transfer between ions of a molecule:
    Page-71 (2010)
eps_0 = 8.854D-12; // Absolute electrical
    permittivity of free space, coulomb sqaure per
```

```
    newton per metre square
e = 1.6D-19; // Electronic charge, C
R = 5D-10;    // Separation between the ions M and X, m
IP_M = 5;     // Ionization potential of M, eV
EA_X = 4;     // Electron affinity of X, eV
U = -e/(4*%pi*eps_0*R); // The potential energy of MX molecule, eV
delta_E = IP_M - EA_X; // The net energy required to produce the ion pair, eV
Er = delta_E + U; // Energy required to transfer an electron from M to X atom, eV
printf("\nThe energy required to transfer an electron from M to X atom = %4.2f eV", Er);

//Result
// The energy required to transfer an electron from M to X atom = -1.88 eV
```

Chapter 3

Atomic Packing

Scilab code Exa 3.1 Packing of spheres in 2D square lattice

```
1 // Scilab Code Ex3.1 Packing of equal spheres in two
    dimensional square lattice: Page-88 (2010)
2 // Here we may assume square of unit length i.e. a =
    1 such that radius of sphere, R = a/2 = 0.5
3 a = 1;      // Length of the side of the square, unit
4 R = a/2;      // Radius of the sphere, unit
5 r = (sqrt(2)-1)*R; // Radius of the sphere
    introduced within the void produced by the
    packing of equal spheres on square lattice, unit
6 A = %pi*R^2; // Area associated with a sphere,
    square units
7 FA = a^2-A;     // Free area occupied by void in
    square lattice, square units
8 FA_per = FA*100;     // Percentage free area in
    square lattice
9 printf("\nFree area in square lattice is : %4.1f
    percent", FA_per);
10 //Result
11 // Free area in square lattice is : 21.5 percent
```

Scilab code Exa 3.2 Packing efficiency in diamond structure

```
1 // Scilab Code Ex3.2 Packing efficiency in diamond
    structure: Page-92 (2010)
2 // For simplicity we may take radius of the atom, R
    = 1 unit
3 R = 1;     // Radius of the atom in bcc lattice, unit
4 nc = 8;     // Number of corner atoms in diamond
    structure
5 nfcc = 6;  // Number of face centred atoms in
    diamond structure
6 na = 4;     // Number of atoms completely within the
    unit cell
7 n = 1/8*nc+1/2*nfcc+1*na; // Effective number of
    atoms in the diamond structure
8 V_atom = 8*4/3*%pi*R^3;  // Volume of atoms within
    the unit cell, unit cube
9 // Since for a diamond cubic crystal, the space
    lattice is fcc, with two atos per lattice point,
    such that 8*R = sqrt(3)*a, solving for a
10 a = 8*R/sqrt(3); // lattice parameter of diamond
    structure, unit
11 V_cell = a^3;     // Volume of the unit cell, unit
    cube
12 eta = V_atom/V_cell*100; // Packing efficiency in
    diamond structure
13 printf("\nThe packing efficiency in diamond
    structure is : %2.0f percent", eta);
14 //Result
15 // The packing efficiency in diamond structure is :
    34 percent
```

Scilab code Exa 3.3 Radius of largest sphere at octahedral void

```
1 // Scilab Code Ex3.3 Radius of largest sphere that
    can be placed at the octahedral void: Page-100
    (2010)
2 // For simplicity we may take radius of the atom, R
    = 1 unit
3 R = 1;     // Radius of the atom in bcc lattice, unit
4 // For a bcc lattice, 4*R = a*sqrt(3), solving for a
5 a = 4*R/sqrt(3); // lattice parameter of bcc crystal
    , unit
6 // Since R + Rx = a/2, solving for Rx
7 Rx = a/2 - R;  // Radius of the largest sphere that
    will fit into the octahedral void, unit
8 printf("\nThe radius of the largest sphere that will
     fit into the octahedral void is : %5.3fR", Rx);
9 //Result
10 // The radius of the largest sphere that will fit
    into the octahedral void is : 0.155R
```

Scilab code Exa 3.4 Radius of largest sphere at tetrahedral void

```
1 // Scilab Code Ex3.4 Radius of largest sphere that
    can be placed at the tetrahedral void: Page-100
    (2010)
2 // For simplicity we may take radius of the atom, RL
     = 1 unit
3 RL = 1;     // Radius of the atom in bcc lattice,
    unit
4 // For a bcc lattice, 4*RL = a*sqrt(3), solving for
    a
5 a = 4*RL/sqrt(3); // Lattice parameter of bcc
    crystal, unit
6 // Further RL + Rx = sqrt(5)*a/4, solving for Rx
7 Rx = sqrt(5)*a/4-RL;  // Radius of the largest
```

```
    sphere that will fit into the octahedral void,
    unit
8 printf("\nThe radius of the largest sphere that will
     fit into the tetrahedral void is : %5.3fRL", Rx)
    ;
9 //Result
10 // The radius of the largest sphere that will fit
    into the tetrahedral void is : 0.291RL
```

Scilab code Exa 3.5 Diameter of the largest atom at tetrahedral void

```
1 // Scilab Code Ex3.5 Diameter of the largest atom
    that would fit into the tetrahedral void:5 Page
    -101 (2010)
2 a = 3.52D-10;     // Lattice parameter for Ni, m
3 // For an fcc lattice, sqrt(2)*a = 4*R, solving for
    R
4 R = sqrt(2)*a/4;     // Radius of the atom in fcc
    lattice, m
5 R_oct = 0.414*R;     // Radius of the octahedral void
     in fcc close packing, m
6 D = 2*R_oct;     // Diameter of the octahedral void
    in the fcc structure of nickel, m
7 disp(D/1D-10, "The diameter of the octahedral void
    in the fcc structure of nickel, in angstrom, is :
     ");
8 //Result
9 // The diameter of the octahedral void in the fcc
    structure of nickel, in angstrom, is :
10 // 1.0304526
```

Scilab code Exa 3.6 Void space in cubic close packing

```
// Scilab Code Ex3.6 Void space in cubic close
   packing: Page-101 (2010)
R = 1;    // For simplicity, radius of the sphere, m
// For cubic close packing, side of the unit cell
   and the radius of the sphere is related as
//     sqrt(2)*a = 4*R, solving for a
a = 2*sqrt(2)*R;    // Lattice parameter for cubic
   close packing, m
V_cell = a^3;    // Volume of the unit cell
n = 4;    // Number of lattice points in fcc unit
   cell
V_occupied = 4*4/3*%pi*((1.000)^3+(0.414)
   ^3+2*(0.225)^3); // Volume occupied by the atoms,
    metre cube
void_space = V_cell - V_occupied;     // Void space
   in the close packing
percent_void = void_space/V_cell*100; // Percentage
   void space
printf("\nThe void space in the close packing is :
   %2.0f percent", percent_void);
//Result
// The void space in the close packing is : 19
   percent
```

Scilab code Exa 3.7 The Minimum value of radius ratio in a compound

```
// Scilab Code Ex3.7 The minimum value of radius
   ratio in AX-compound: Page-104 (2010)
// For simplicity we may assume a = 1
a = 1;    // Lattice parameter of the crystal, unit
b = 2/3*a*sin(%pi/3);    // Lattice parameter of the
    crystal, unit
// Here a = 2*Rx, where a is the lattice parameter
   and Rx is the radius of X-ions representing the
   bigger spheres, solving for Rx
```

```
Rx = 0.5*a;
// Also b = RA + Rx, solving for RA
RA = b - Rx;     // Radius of A-ion representing teh smaller sphere, unit
Rad_ratio = RA/Rx;     // Radius ratio in AX compound
printf("\nThe minimum value of radius ratio in AX compound is : %5.3f", Rad_ratio);
// Result
// The minimum value of radius ratio in AX compound is : 0.155
```

Chapter 4

Atomic Shape and Size

Scilab code Exa 4.1 Bohr orbit for the hydrogen atom

```
1 // Scilab Code Ex4.1 Bohr's orbit for the hydrogen
    atom: Page-126 (2010)
2 n = 1;     // The ground state orbit of hydrogen atom
3 Z = 1;     // The atomic number of hydrogen
4 h = 6.626D-34; // Plank's constant, Js
5 eps_0 = 8.85D-12;     // Absolute electrical
    permittivity of free space, coulomb square per
    newton per metre square
6 e = 1.602D-19;     // Electronic charge, C
7 m = 9.1D-31;     // Electronic mass, kg
8 r_B = (n^2*h^2*eps_0)/(%pi*m*Z*e^2);  // Radius of
    first Bohr's orbit (Bohr radius), m
9 disp(r_B/1D-10, "The radius of first Bohr orbit, in
    angstrom, is : ");
10 // Result
11 // The radius of first Bohr orbit, in angstrom, is :
12 // 0.5295779
```

Scilab code Exa 4.2 Ionization potentials of hydrogen atom

```
// Scilab Code Ex4.2 Ionization potentials of
   hydrogen atom: Page-126 (2010)
Z = 1;     // The atomic number of hydrogen
h = 6.626D-34; // Plank's constant, Js
eps_0 = 8.85D-12;  // Absolute electrical
   permittivity of free space, coulomb square per
   newton per metre square
e = 1.602D-19;     // Electronic charge, C
m = 9.1D-31;     // Electronic mass, kg
E = zeros(1, 3); // Initialize three potentials to 0
    value in a vector
for n = 1:1:3
    select n
    case 1 then
        state = "First";
    case 2 then
        state = "Second";
    else
        state = "Third";
    end
E(1,n) = -(m*Z^2*e^4)/(8*eps_0^2*n^2*h^2*e);     //
   Energy of nth bohr orbit, eV
printf("\nThe %s Ionization Potential is : %5.3f eV"
   ,state, E(1,n));
end
// Result
// The First Ionization Potential is : -13.600 eV
// The Second Ionization Potential is : -3.400 eV
// The Third Ionization Potential is : -1.511 eV
```

Scilab code Exa 4.3 Univalent radii of ions

```
// Scilab Code Ex4.3 Univalent radii of ions: Page
   -130 (2010)
S = 4.52;     // Screening constant for neon like
```

```
    configurations
Cn = 1;     // A constant determined by the quantum
    number, m; for simplicity it can be assumed as
    unity
Z_Na = 11;     // Atomic number of sodium
Z_F = 9;     // Atomic number of fluorine
Z_O = 8;     // Atomic number of oxygen
r_Na = Cn/(Z_Na - S);  // Radius of sodium ion, m
r_F = Cn/(Z_F - S);     // Radius of fluorine ion, m
r_ratio = r_Na/r_F;     // Radius ratio
r_Na = r_F*r_ratio;     // Calculating radius of
    sodium ion from r_ratio, m
// Given that r_Na + r_F = 2.31D-10,
// or r_Na + r_Na/0.69 = 2.31D-10,
// or r_Na(1 + 1/0.69) = 2.31D-10, solving for r_Na
r_Na = 2.31D-10/(1+1/0.69);     // Calculating radius
     of sodium, m
r_F = 2.31D-10 - r_Na;     // Calculating radius of
    fluorine from r_Na, m
Cn = r_Na*(Z_Na - S);     // Calculating Cn, m
r_O = Cn/(Z_O - S);     // Radius of oxygen, m
disp(r_Na/1D-10,"Radius of sodium ion, in angstrom,
    is :");
disp(r_F/1D-10, "Radius of fluorine ion, in angstrom
    , is :");
disp(Cn/1D-10, "Constant determined by quantum
    number is : ");
disp(r_O/1D-10, "Radius of oxygen, in angstrom, is :
     ");
// Result
// Radius of sodium ion, in angstrom, is :
// 0.9431361
// Radius of fluorine ion, in angstrom, is :
// 1.3668639
// Constant determined by quantum number, in
    angstrom, is :
// 6.1115219
// Radius of oxygen, in angstrom, is :
```

```
30 // 1.7561845
```

Scilab code Exa 4.4 Ionic Radius of Si ions in silicon dioxide

```
1 // Scilab Code Ex4.4 Ionic Radius of Si ions in
    silicon dioxide: Page-131 (2010)
2 a = 7.12D-10;    //  Lattice parameter of the
    crystal. m
3 d = sqrt(3*a^2/16); // Si-Si distance from (0,0,0)
    to (1/4,1/4,1/4)
4 RO = 1.40D-10;    // Radius of oxygen, m
5 // Distance of oxygen ions between the two Si ions
    is 2*RSi+2*RO = d, solving for RSi
6 RSi = (d - 2*RO)/2;    // Radius of silicon ion, m
7 disp(RSi/1D-10, "The radius of Si4+ ion, in angstrom
    , is : ");
8 //Result
9 // The radius of Si4+ ion, in angstrom, is :
10 // 0.1415252
```

Scilab code Exa 4.5 Ionic Radius occupying an octahedral position

```
1 // Scilab Code Ex4.5 Ionic Radius occupying an
    octahedral position: Page-138 (2010)
2 R_ratio = 0.414;    // Radius ratio for an
    octahedral void in am M+X- ionic lattice
3 R_x = 2.5D-10;    // Critical radius of X- anion, m
4 R_m = R_x*0.414;    // Radius of M+ cation, m
5 disp(R_m/1D-10, "The radius of cation occupying
    octahedral position in an M+X- ionic solid, in
    angstrom, is : ");
6 //Result
```

```
7 // The radius of cation occupying octahedral
    position in an M+X- ionic solid, in angstrom, is
    :
8 // 1.035
```

Scilab code Exa 4.6 Percentage ionic character of a covalent molecule

```
1 // Scilab Code Ex4.7 Percentage ionic character of a
    covalent molecule: Page-142 (2010)
2 x_A = 4.0;     // Electronegativity of fluorine
3 x_B = 2.1;     // Electronegativity of hydrogen
4 P = 16*(x_A - x_B) + 3.5*(x_A - x_B)^2; //Percentage
    ionic character of the covalent bond in HF
    molecule
5 printf("\nThe percentage ionic character in HF
    molecule is %5.2f percent", P);
6 //Result
7 // The percentage ionic character in HF molecule is
    43.03 percent
```

Scilab code Exa 4.7 Metallic radius from unit cell dimension

```
1 // Scilab Code Ex4.8 Calculating metallic radius
    from unit cell dimension: Page-146 (2010)
2 a = 2.81D-10;     // Unit cell dimension of bcc
    structure of iron, m
3 // For bcc structure we have
4 //         sqrt(3)*a = 4*R, solving for R
5 R = sqrt(3)/4*a;     // Metallic radius of iron atom,
    m
6 printf("\nThe metallic radius of iron atom is %4.2f
    angstrom", R/1D-10);
7 //Result
```

```
8 // The metallic radius of iron atom is 1.22 angstrom
```

Scilab code Exa 4.8 Metallic radii from unit cell dimension

```
1 // Scilab Code Ex4.9 Calculating metallic radii from
      unit cell dimensions: Page-146 (2010)
2 a_Au = 4.08e-10;     // Unit cell dimension of fcc
     structure of gold, m
3 a_Pt = 3.91e-10;     // Unit cell dimension of fcc
     structure of platinum, m
4 // For fcc structure we have
5 //         sqrt(2)*a = 4*R, solving for R
6 R_Au = sqrt(2)/4*a_Au;     // Metallic radius of gold
      atom, m
7 R_Pt = sqrt(2)/4*a_Pt;     // Metallic radius of gold
      atom, m
8 printf("\nThe metallic radius of gold atom, in
     angstrom, is : %4.2f", R_Au/1D-10);
9 printf("\nThe metallic radius of platinum atom, in
     angstrom, is : %4.2f", R_Pt/1D-10);
10 //Result
11 // The metallic radius of gold atom, in angstrom, is
      : 1.44
12 // The metallic radius of platinum atom, in angstrom
     , is : 1.38
```

Scilab code Exa 4.9 Metallic diameter and unit cell dimension of aluminium

```
1 // Scilab Code Ex4.10 Calculating metallic diameter
     and unit cell dimension of aluminium: Page-146
     (2010)
2 Z_Al = 13;     // Atomic number of aluminium
3 A_Al = 26.98;     // Atomic mass of aluminium, g
```

```
4  d_Al = 2700D3;     // Density of aluminium, g per
       metre cube
5  n = 4;     // number of atoms in the fcc structure of
        aluminium
6  N = 6.023D+23;     // Avogadro's number
7  // We have number of atoms per fcc unit cell given
       as
8  // n = (V*d_Al*N)/A_Al, solving for V
9  // V = (n*A_Al)/(d_Al*N), V is the volume of the
       unit cell
10 // or a^3 = (n*A_Al)/(d_Al*N), solving for a
11 a = ((n*A_Al)/(d_Al*N))^(1/3);     // unit cell
       parameter of aluminium
12 // For an fcc structure we have
13 // sqrt(2)*a = 4*R = 2*D, solving for D
14 D = a/sqrt(2);     // metallic diameter of aluminium
       having fcc structure
15 printf("\nThe unit cell dimension of aluminium, is :
        %4.2f angstrom", a/1D-10);
16 printf("\nThe metallic diametre of aluminium, is :
       %4.2f angstrom", D/1D-10);
17 //Result
18 // The unit cell dimension of aluminium, is : 4.05
       angstrom
19 // The metallic diametre of aluminium, is : 2.86
       angstrom
```

Chapter 5

Crystal Imperfections

Scilab code Exa 5.1 Variation of atomic fraction with temperature

```
1 // Scilab Code Ex5.1 Variation of fraction of atoms
    in a solid with temperature Page-158 (2010)
2 E = 1.5;     // Energy of the solid, electron-volt
3 T1 = 300;     // First absolute temperature, K
4 T2 = 1500;     // Second absolute temperature, K
5 k = 8.614D-5;     // Boltzmann constant, electron-
    volt/K
6 // Now fraction of atoms = f_atom = n/N = exp(-E/(k*
    T)
7 f_atom_300 = exp(-E/(k*T1));     // Fraction of atoms
     in the solid at 300 K
8 f_atom_1000 = exp(-E/(k*T2));     // Fraction of
    atoms in the solid at 1000 K
9 printf("\nThe fraction of atoms in the solid at 300
    K, is : %5.3e", f_atom_300);
10 printf("\nThe fraction of atoms in the solid at 1000
     K, is : %5.3e", f_atom_1000);
11 //Result
12 // The fraction of atoms in the solid at 300 K, is :
     6.185e-026
13 // The fraction of atoms in the solid at 1000 K, is
```

```
: 9.084e-006
```

Scilab code Exa 5.2 Vacancy formation in copper

```
1 // Scilab Code Ex5.2 Vacancy formation in copper
    Page-159 (2010)
2 E = 1;    // Energy of formation of vacancy in
    copper, electron-volt
3 T = 1356;    // Melting point of copper, K
4 k = 8.614D-5;    // Boltzmann constant, electron-
    volt
5 N = 6.023D23;    // Avogadro's number
6 // Now fraction of vacancies = f_vacancy = n/N = exp
    (-E/(k*T)
7 f = exp(-E/(k*T));    // Fraction of vacancies in
    the solid at 300 K
8 n = N*f;  // Number of vacancy per mole
9 delta_d = n + N;    // Change in the density due to
    creation of vacancy
10 f_d = delta_d/N;    // Relative change in the
    density of copper due to vacancy formation
11 printf("\nThe relative change in the density of
    copper due to vacancy formation (n+N)/N, is : %9
    .7f : 1", f_d);
12 //Result
13 // The relative change in the density of copper due
    to vacancy formation (n+N)/N, is : 1.0001914 : 1
```

Scilab code Exa 5.3 Concentration of Schottky imperfections

```
1 // Scilab Code Ex5.3 Concentration of Schottky
    imperfections Page-159 (2010)
2 N = 6.023D23;    // Avogadro's number
```

```
k = 8.614D-5;     // Boltzmann's constant, eV/K
T1 = 27+273;      // First absolute temperature, K
T2 = 1000;     // Second absolute temperature, K
C_300 = 1D-10;     // Concentration of Schottky
   defects in an fcc crystal at 300 K temperature
n = C_300*N;     // Number of Schottky imperfections
   per mole
d = 1D-10;     // Interatomic spacing assumed to be
   unit angstrom, m
V = d^3;     // Volume of the unit cube, metre cube
V_mole = V*N;     // Volume occupied by one mole of
   atoms in fcc crystal, metre cube
V_per_defect = V_mole/n;     // Volume per defect,
   metre cube
a = (V_per_defect)^(1/3);     // Average separation
   between the defects, m
E_v = 23.03*k*T1;     // Energy of the solid,
   electron-volt
C_1000 = exp(-E_v/(k*T2));     // Schottky defect
   concentration at 1000 K
printf("\nThe average separation between the defects
   , is : %3.1e m", a);
printf("\nThe expected concentration of Schottky
   defect at 1000 K, n/N, is : %3.1e", C_1000);
//Result
// The average separation between the defects, is :
   2.2e-007 m
// The expected concentration of Schottky defect at
   1000 K, n/N, is : 1.0e-003
```

Scilab code Exa 5.4 Number of Schottky imperfections in NaCl crystal

```
// Scilab Code Ex5.4 Number of Schottky
   imperfections in NaCl crystal Page-160 (2010)
N = 6.023D23;     // Avogadro's number
```

```
3 k = 8.614D-5;     // Boltzmann's constant, eV/K
4 T = 27+273;     // Absolute room temperature, K
5 Ep = 2;     // Energy required to remove a pair of Na
    + and Cl- ions, electron-volt
6 // Now Concentration of imperfections in a crystal
    is given by
7 // n/N = exp(-Ep/(2*k*T)), solving for n
8 n = N*exp(-Ep/(2*k*T));     // No. of Schottky
    imperfections present in NaCl crystal
9 printf("\nNo. of Schottky imperfections present in
    NaCl crystal is : %4.2e", n);
10 V = 26.83;     // Volume of one mole of the crystal,
    cm cube
11 n = n/V;     // Number per mole volume of the crystal
    , per cm cube
12 printf("\nConcentration of Schottky imperfections
    present in NaCl crystal is : %4.2e per cm cube",
    n);
13 //Result
14 // No. of Schottky imperfections present in NaCl
    crystal is : 9.42e+006
15 // Concentration of Schottky imperfections present
    in NaCl crystal is : 3.51e+005 per cm cube
```

Scilab code Exa 5.5 Average energy required to create one Schottky defect

```
1 // Scilab Code Ex5.5 Average energy required to
    create one Schottky defect in NaCl Page-160
    (2010)
2 N = 6.023D23;     // Avogadro's number
3 k = 8.614D-5;     // Boltzmann's constant, eV/K
4 T = 27+273;     // Absolute room temperature, K
5 r = 2.82D-10;     // Interatomic separation of NaCl
    cryastal, m
```

```
n = 5D+11;          // Density of defects, per metre
    cube
//Ep = 2;           // Energy required to remove a pair
    of Na+ and Cl- ions, electron-volt
a = 2*r;            // Lattice parameter of unit cell of
    NaCl, m
V = a^3;            // Volume of the unit cell of sodium
    , metre cube
n_ip = 4;       // Number of ion-pairs of NaCl
N = n_ip/V;     // No. of ion-pairs in unit volume of
    an ideal NaCl crystal
// Now n/N = exp(-Ep/(2*k*T)), solving for Ep
Ep = 2*k*T*log(N/n);     // Average energy required
    to create one Schottky defect, electron-volt
printf("\nThe Average energy required to create one
    Schottky defect in NaCl crystal is : %4.2f eV",
    Ep);
//Result
// The Average energy required to create one
    Schottky defect in NaCl crystal is : 1.98 eV
```

Scilab code Exa 5.6 Ratio of Frenkel defects at two different temperatures

```
// Scilab Code Ex5.6 Ratio of Frenkel defects at two
    different temperatures in an ionic crystal Page
    -161 (2010)
k = 8.614D-5;     // Boltzmann's constant, eV/K
Ef = 1.4;     // Average energy required to create a
    Frenkel defect, eV
T1 = 300;     // First absoluite temperature, K
T2 = 600;     // Second absolute temperature, K
// The concentration of Frenkel defect for given Ef
    and absolute temperature T is given by
// n = A*exp(-Ef/(2*k*T)), per metre cube, so that
// n1 = A*exp(-Ef/(2*k*T1)), per metre cube, and
```

```
// n2 = A*exp(-Ef/(2*k*T2)), per metre cube, therefore,
// n1/n2 = exp((-Ef/k)*(1/T1 - 1/T2)), the ratio of Frenkel defects is
n300_r_n600 = exp((-Ef/(2*k))*(1/T1 - 1/T2)); // Frenkel defect ratio
printf("\nThe ratio of Frenkel defect, n300_r_n600, is : %5.3e", n300_r_n600);
//Result
// The ratio of Frenkel defect, n300_r_n600, is : 1.312e-006
```

Scilab code Exa 5.7 Dislocation density of bcc structure of iron

```
// Scilab Code Ex5.7 Dislocation density of bcc structure of iron Page-163 (2010)
L = 0.15;     // Length of the strip, m
t = 0.02;     // Thickness of the iorn strip, m
r = 0.12;     // Radius of curvature of the bent, m
a = 2.81D-10;     // Lattice parameter of the bcc structure of iron, m
b = sqrt(3)*a/2;     // Magnitude of Burger vector, m
// For n positive edge dislocations
// n*b = L*t/r, solving for n/(L*t)
// n/(L*t) = 1/(r*b), Number of dislocation line piercing through a unit area of the plane of the paper, per metre square
d = 1/(r*b);     // Dislocation density in bcc structure of iron, number per metre square
printf("\nThe dislocation density in bcc structure of iron : %4.2e, dislocations per Sq. m", d);
//Result
// The dislocation density in bcc structure of iron : 3.42e+010, dislocations per Sq. m
```

Scilab code Exa 5.8 Minimum dislocation density in aluminium

```
1 // Scilab Code Ex5.8 Minimum dislocation density in
    aluminium Page-164 (2010)
2 b = 3D-10;     // Magnitude of Burgers vector, m
3 r = 0.05;     // Radius of curvatur of the aluminium
    crystal, m
4 // For n positive edge dislocations
5 // n*b = L*t/r, solving for n/(L*t)
6 // n/(L*t) = 1/(r*b), Number of dislocation line
    piercing through a unit area of the plane of the
    paper, per Sq.m
7 d = 1/(r*b);     // Minimum dislocation density in
    aluminium, number per Sq. m
8 printf("\nThe minimum dislocation density in
    aluminium, %4.1e, dislocations per Sq. m", d);
9 //Result
10 // The minimum dislocation density in aluminium, 6.7
    e+010, dislocations per Sq. m
```

Scilab code Exa 5.9 Total force from its resolved component in a given direction

```
1 // Scilab Code Ex5.9 Determining total force from
    its resolved component in a given direction: Page
    -168 (2010)
2 h1 = 1; k1 = -1; l1 = 0 // Miller indices for first
    set of planes
3 h2 = 1; k2 = 0; l2 = 0;     // Miller indices for
    second set of planes
4 F_100 = 130;     // Resolved component of force along
    [100] direction, N
```

```
5 cos_theta = (h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1
    ^2)*sqrt(h2^2+k2^2+l2^2));    // Cosine of angle
    between [1 -1 0] and [100] directions
6 // As F/F_100 = cos_theta, solving for F
7 F_110 = F_100/cos_theta;    // Applied force along
    [1 -1 0] direction, N
8 printf("\nThe applied force along [1-10] direction =
    %3d N", F_110);
9 // Result
10 // The applied force along [1-10] direction = 183 N
```

Scilab code Exa 5.10 Resolved componet of shearing force in a given direction

```
1 // Scilab Code Ex5.10 Determining resolved componet
    of shearing force in a given direction: Page-168
    (2010)
2 h1 = 1; k1 = 1; l1 = 1 // Miller indices for first
    set of planes
3 h2 = 1; k2 = 1; l2 = 0;    // Miller indices for
    second set of planes
4 F_111 = 660;    // Shearing force along [111]
    direction, N
5 cos_theta = (h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1
    ^2)*sqrt(h2^2+k2^2+l2^2));    // Cosine of angle
    between [1 -1 0] and [100] directions
6 // As F_110/F_111 = cos_theta, solving for F_110
7 F_110 = F_111*cos_theta;    // Resolved component of
    shearing force along [110] direction, N
8 printf("\nThe resolved component of shearing force
    along [110] direction, F_110 = %3d N", F_110);
9 // Result
10 // The resolved component of shearing force along
    [110] direction, F_110 = 538 N
```

Scilab code Exa 5.11 Dependence of applied stress on the slip direction

```
1 // Scilab Code Ex5.11 Dependence of applied stress
    on the slip direction of a copper: Page-169
    (2010)
2 tau_critical = 1;     // Critical shear stress for
    the <-110>{111} slip system, mega-pascal (MPa)
3 // For directions [001] and [-111]
4 h1 = 0; k1 = 0; l1 = 1      // Miller indices for
    first set of planes
5 h2 = -1; k2 = 1; l2 = 1;     // Miller indices for
    second set of planes
6 cos_phi = (h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1^2)*
    sqrt(h2^2+k2^2+l2^2));     // Cosine of angle
    between [001] and [-111] directions
7 // For directions [001] and [101]
8 h1 = 0; k1 = 0; l1 = 1      // Miller indices for
    first set of planes
9 h2 = 1; k2 = 0; l2 = 1;     // Miller indices for
    second set of planes
10 cos_lambda = (h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1
    ^2)*sqrt(h2^2+k2^2+l2^2));     // Cosine of angle
    between [001] and [101] directions
11 sigma = tau_critical/(cos_phi*cos_lambda);     //
    Stress along [001] direction, newton per metre
    square
12 printf("\nThe stress required to be applied along
    [001] direction to produce slip in the [101]
    direction on the (-111) plane = %4.2f MPa", sigma
    );
13 // For directions [001] and [110]
14 h1 = 0; k1 = 0; l1 = 1      // Miller indices for
    first set of planes
15 h2 = 1; k2 = 1; l2 = 0;     // Miller indices for
```

```
    second set of planes
16 cos_lambda = (h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1
    ^2)*sqrt(h2^2+k2^2+l2^2));    // Cosine of angle
    between [001] and [110] directions
17 if cos_lambda <> 0 then
18     sigma = tau_critical/(cos_phi*cos_lambda);    //
        Stress along [001] direction, newton per
       metre square
19     printf("\nThe stress required to be applied
       along [001] direction to produce slip in the
       [110] direction on the (-111) plane = %4.2f
       MPa", sigma);
20 else
21     printf("\nSince cos_lambda = 0, this implies
       that slip cannot occur in [110] direction
       when the stress is applied along [001]
       direction");
22 end
23 // Result
24 // The stress required to be applied along [001]
    direction to produce slip in the [101] direction
    on the (-111) plane = 2.45 MPa
25 // Since cos_lambda = 0, this implies that slip
    cannot occur in [110] direction when the stress
    is applied along [001] direction
```

Scilab code Exa 5.12 Resolved stress in a direction from applied stress in other direction

```
1 // Scilab Code Ex5.12 Resolved stress in a direction
    from applied stress in some other direction of
    bcc iron: Page-169 (2010)
2 sigma = 123;    // Axial stress applied in the
    direction [110] of bcc iron, MPa
3 // For directions [010] and [110]
```

```
4 h1 = 0; k1 = 1; l1 = 0      // Miller indices for
    first set of planes
5 h2 = 1; k2 = 1; l2 = 0;     // Miller indices for
    second set of planes
6 cos_phi = (h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1^2)*
    sqrt(h2^2+k2^2+l2^2));     // Cosine of angle
    between [010] and [110] directions
7 // For directions [110s] and [101]
8 h1 = 1; k1 = 0; l1 = 1      // Miller indices for
    first set of planes
9 h2 = 1; k2 = 1; l2 = 0;     // Miller indices for
    second set of planes
10 cos_lambda = (h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1
    ^2)*sqrt(h2^2+k2^2+l2^2));     // Cosine of angle
    between [110] and [101] directions
11 tau = sigma*cos_phi*cos_lambda;     // Resolved shear
     stress in the [101] direction on the (010) plane
    , MPa
12 printf("\nThe resolved shear stress in the [101]
    direction on the (010) plane = %4.1f MPa", tau);
13 // Result
14 // The resolved shear stress in the [101] direction
    on the (010) plane = 43.5 MPa
```

Scilab code Exa 5.13 Critical resolved shear stress from applied stress in a given direction

```
1 // Scilab Code Ex5.13 Determining critical resolved
    shear stress from applied stress in a given
    direction  of aluminium: Page-170 (2010)
2 sigma_critical = 3.5;     // Applied stress in the [1
     -1 1] direction, MPa
3 // For directions [111] and [1 -1 1]
4 h1 = 1; k1 = 1; l1 = 1;      // Miller indices for
    first set of planes
```

```
5 h2 = 1; k2 = -1; l2 = 1;     // Miller indices for
    second set of planes
6 cos_phi = (h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1^2)*
    sqrt(h2^2+k2^2+l2^2));     // Cosine of angle
    between [111] and [1 -1 1] directions
7 // For directions [1 - 1 0] and [1 -1 1]
8 h1 = 1; k1 = -1; l1 = 0      // Miller indices for
    first set of planes
9 h2 = 1; k2 = -1; l2 = 1;     // Miller indices for
    second set of planes
10 cos_lambda = (h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1
    ^2)*sqrt(h2^2+k2^2+l2^2));     // Cosine of angle
    between [1 -1 0] and [1 -1 1] directions
11 tau_c = sigma_critical*cos_phi*cos_lambda;     // The
     critical resolved shear stress in the [1 -1 0]
    direction on the (111) plane, MPa
12 printf("\nThe critical resolved shear stress in the
    [1 -1 0] direction on the (111) plane = %4.2f MPa
    ", tau_c);
13 // Result
14 // The critical resolved shear stress in the [1 -1
    0] direction on the (111) plane = 0.95 MPa
```

Scilab code Exa 5.14 Initiation of slip by the applied stress

```
1 // Scilab Code Ex5.14 Determining the direction in
    which slip is initiated by the applied stress in
    zinc: Page-170 (2010)
2 sigma = 2.3;     // Applied stress when the plastic
    deformation is first observed, MPa
3 phi = 60;     // Angle which the normal to the basal
    plane makes with the tensile axis of zinc, degree
4 // Function to find the value of resolved shear
    stress
5 function[tau] = stress(lambda)
```

```
6       tau = sigma*cosd(phi)*cosd(lambda);
7   endfunction
8   lambda = [38 45 84];      // Angles which the three
       slip directions x1, x2 and x3 respectively makes
       with the tensile axis, degrees
9   t = zeros(1,3);           // Initialize a one-
       dimensional vector of three elements
10  for i = 1:1:3
11      t(i) = stress(lambda(i));     // Calculate the
           value of resolved shear stress by calling
           stress function
12      printf("\ntau%d = %5.3f MPa", i, t(1,i));     //
           Display resloved shear stress for each
           direction, MPa
13  end
14  // Locate for the largest resolved stress value
15  big = t(1,1);
16  for i = 2:1:3
17      if t(1,i) > big then
18          big = t(1,i)     // Set largest valuse of
               resolved stress if the condition meets
19      end
20  end
21  printf("\nThe slip is initiated along direction x1
       at tau_c = %5.3f MPa", big);
22  // Result
23  // tau1 = 0.906 MPa
24  // tau2 = 0.813 MPa
25  // tau3 = 0.120 MPa
26  // The slip is initiated along direction x1 at tau_c
       = 0.906 MPa
```

Scilab code Exa 5.15 Applied tensile stress in a direction to initiate plastic deformation

```
1 // Scilab Code Ex5.15 Determining applied tensile
    stress in a direction to initiate plastic
    deformation: Page-170 (2010)
2 tau_critical = 0.7;     // Critical resolved shear
    stress for fcc crystal, MPa
3 // For directions [100] and [1 1 1]
4 h1 = 1; k1 = 0; l1 = 0;      // Miller indices for
    first set of planes
5 h2 = 1; k2 = 1; l2 = 1;     // Miller indices for
    second set of planes
6 cos_phi = (h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1^2)*
    sqrt(h2^2+k2^2+l2^2));     // Cosine of angle
    between [100] and [1 1 1] directions
7 // For directions [1 0 0] and [1 -1 0]
8 h1 = 1; k1 = 0; l1 = 0      // Miller indices for
    first set of planes
9 h2 = 1; k2 = -1; l2 = 0;     // Miller indices for
    second set of planes
10 cos_lambda = (h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1
    ^2)*sqrt(h2^2+k2^2+l2^2));     // Cosine of angle
    between [1 0 0] and [1 -1 0] directions
11 sigma_c = tau_critical/(cos_phi*cos_lambda);     //
    The critical resolved shear stress in the [1 -1
    0] direction on the (1 1 1) plane, MPa
12 printf("\nThe critical resolved shear stress in the
    [1 -1 0] direction on the (1 1 1) plane = %3.1f
    MPa", sigma_c);
13 // Result
14 // The critical resolved shear stress in the [1 -1
    0] direction on the (1 1 1) plane = 1.7 MPa
```

Scilab code Exa 5.16 Dislocation width in copper

```
1 // Scilab Code Ex5.16 Dislocation width in copper:
    Page-175 (2010)
```

```
mu = 1;     // For simplicity, assume shear modulus
   of copper to be unity, netwon per metre square
tau_PN = mu/1e+05;     // Shear stress to initiate
   plastic deformation, newton per metre square
a = 3.61e-010;     // Lattice parameter of copper, m
b = a/sqrt(2);     // Burger vector magnitude for fcc
    crystal of copper, m
// As stress necessary to move a dislocation in a
   crystal is given by
// tau_PN = mu*exp(-2*%pi*w/b), solving for w
w = b*log(mu/tau_PN)/(2*%pi);     // Width of the
   dislocation in copper, m
printf("\nThe width of dislocation in copper = %4.2e
    angstrom", w/1d-10);
// Result
// The width of dislocation in copper = 4.68e-010
   angstrom
```

Scilab code Exa 5.17 Change in number of vacancies due to disloaction motion

```
// Scilab Code Ex5.17 Change in number of vacancies
   due to disloaction motion: Page-176 (2010)
l = 1e-03;     // Edge dislocation length of simple
   cubic crystal, m
d = 1e-06;     // Distance of dislocation climb in, m
a = 3e-10;     // Lattice parameter of scc, m
A = a^2;     // Area of the unit cell, metre square
A_affected = l*d;     // Affected area when the
   dislocation climbs down, metre square
// N.B.: Area of one unit cell in scc contributes
   one atom
N = A_affected/A;     // Number of vacancies created
   within the affected area
printf("\nThe number of vacancies lost or created =
```

```
    %3.1e", N);
// Result
// The number of vacancies lost or created = 1.1e
    +010
```

Scilab code Exa 5.18 Minimum number of dislocations in motion from shearing rate

```
// Scilab Code Ex5.18 Minimum number of dislocations
    in motion from shearing rate of copper: Page-176
    (2010)
a = 3.61e-010;     // Lattice parameter of copper, m
epsilon_dot = 10/60;     // Strain rate of plastic
    deformation, mm per sec
v_d = 1e+06;     // Velocity of dislocation, mm per
    sec
V = 1e+03;     // Volume of the crystal, mm cube
b = a*1e+03/sqrt(2);     // Burger vector magnitude
    for fcc crystal of copper, mm
// Strain rate of plastic deformation is given by
// epsilon_dot = rho*b*v_d, solving for rho
rho = epsilon_dot/(b*v_d);     // Density of the
    mobile disloacations, per mm cube
N = round(rho*V);     // Number of dislocations in
    motion in the whole cube
printf("\nThe number of dislocations in motion in
    the whole cube = %3d", N);
// Result
// The number of dislocations in motion in the whole
    cube = 653
```

Scilab code Exa 5.19 Elastic energy of line imperfection

```
// Scilab Code Ex5.19 Elastic energy of line imperfection stored in Al: Page-178 (2010)
rho = 1e+010;     // Dislocation density of Al, per metre square
mu = 25.94e+09;     // Shear molulus of aluminium, newton per metre square
a = 4.05e-010;     // Lattice parameter of aluminium, m
b = a/sqrt(2);     // Burger vector magnitude for fcc crystal of Al, m
E_bar = mu*b^2/2;     // Elastic energy per unit length of the dislocation, joule per metre
E = E_bar*rho;     // Elastic energy stored in the crystal, joule per metre cube
printf("\nThe elastic energy stored in the crystal = %5.2f joule per metre cube", E);
// Result
// The elastic energy stored in the crystal = 10.64 joule per metre cube
```

Scilab code Exa 5.20 Spacing between dislocations in a tilt boundary

```
// Scilab Code Ex5.20 Spacing between dislocations in a tilt boundary in fcc Ni: Page-187 (2010)
theta = 2;     // Angle of tilt, degree
a = 3.52e-010;     // Lattice parameter of Al, m
b = a/sqrt(2);     // Burger vector magnitude for fcc Ni, m
h = b/tand(theta);     // The vertical spacing between two neighbouring edge dislocations, m
printf("\nThe spacing between dislocations in a tilt boundary in fcc Ni = %4.1f angstrom", h/1D-10);
// Result
// The spacing between dislocations in a tilt boundary in fcc Ni = 71.3 angstrom
```

Scilab code Exa 5.21 Tilt angle from dislocation spacing in the boundary

```
1 // Scilab Code Ex5.21 Determining tilt angle from
    dislocation spacing in the boundary of Cu: Page
    -188 (2010)
2 a = 3.61e-010;     // Lattice parameter of Cu, m
3 b = a/sqrt(2);     // Burger vector magnitude for fcc
     Cu, m
4 h = 1.5e-06;     // The vertical spacing between two
    neighbouring edge dislocations, m
5 tan_theta = atand(b/h)*(%pi/180);     // tangent of
    tilt angle between two tilt boundaries of Cu,
    radian
6 printf("\nThe tilt angle between two tilt boundaries
     of Cu = %3.1e radian", theta);
7 // Result
8 // The tilt angle between two tilt boundaries of Cu
    = 1.7e-004 radian
```

Scilab code Exa 5.22 Tilt angle from dislocation spacing

```
1 // Scilab Code Ex5.22 Determining tilt angle from
    dislocation spacing in the boundary of Cu: Page
    -188 (2010)
2 b = 0.4e-09;     // Burger vector magnitude for fcc
    Cu, m
3 h = 3.0e-06;     // The vertical spacing between two
    neighbouring edge dislocations, m
4 tan_theta = atand(b/h)*(%pi/180);     // tangent of
    tilt angle between two tilt boundaries of Cu,
    radian
```

```
5 printf("\nThe tilt angle between two tilt boundaries
      of Cu = %4.2e radian", theta);
6 // Result
7 // The tilt angle between two tilt boundaries of Cu
    = 1.33e-004 radian
```

Chapter 6

Atomic Diffusion

Scilab code Exa 6.1 Rate of diffusion of nitrogen through steel wall

```
1 // Scilab Code Ex6.1 Rate of diffusion of nitrogen
    through steel wall: Page-195 (2010)
2 D = 1e-019;     // Diffusion coeffcient of nitrogen
    in steel at room temperature, metre square per
    sec
3 dc = 10;     // Concentration of nitrogen at the
    inner surface of the tank, kg per metre cube
4 dx = 10e-03;     // Thickness of the steel wall, m
5 J = D*(dc/dx);     // Fick's first law giving outward
     flux of nitrogen through steel wall of the tank,
     kg per metre square per second
6 printf("\nThe rate at which nitrogen escapes through
     the tank wall = %1.0e kg per metre square per
    sec", J);
7 // Result
8 // The rate at which nitrogen escapes through the
    tank wall = 1e-016 kg per metre square per sec
```

Scilab code Exa 6.2 Rate of diffusion of copper through pure Al sheet

```
1 // Scilab Code Ex6.2 Rate of diffusion of copper
    through pure Al sheet: Page-196 (2010)
2 a = 4.05e-010;     // Lattice parameter of fcc Al, m
3 N = 4;     // Number of Al atoms per unit cell of fcc
    Al
4 n = N/a^3;     // Number of Al atoms per unit volume,
    per metre cube
5 D = 5.25e-013;     // Diffusion coeffcient of copper
    in Al at 550 degree celsius, metre square per sec
6 c1 = 0.19e-02;     // Atomic percent of copper at the
    surface, per unit volume
7 c2 = 0.18e-02;     // Atomic percent of copper at the
    the depth 1.2 mm from the surface, per unit
    volume
8 dc = (c2 - c1)*n;    // Change in concentration of
    copper at 1.2 mm depth of the surface, per metre
    cube
9 dx = 1.2e-03;     // Thickness of the pure Al sheet,
    m
10 J = -D*(dc/dx);     // Fick's first law giving
    outward flux of copper through the Al sheet, Cu
    atoms per metre square per second
11 printf("\nThe outward flux of copper through the Al
    sheet = %4.2e Cu atoms per metre square per sec",
    J);
12 // Result
13 // The outward flux of copper through the Al sheet =
    2.63e+015 Cu atoms per metre square per sec
```

Scilab code Exa 6.3 Rate of diffusion of carbon through steel bar

```
1 // Scilab Code Ex6.3 Rate of diffusion of carbon
    through steel bar: Page-196 (2010)
2 a = 3.65e-010;     // Lattice parameter of fcc
    structure of iron, m
```

```
3 D = 3e-011;     // Diffusion coeffcient of carbon in
    iron at 1000 degree celsius, metre square per sec
4 n1 = 20;     // Number of unit cells per carbon atom
    at the surface of steel
5 n2 = 30;     // Number of unit cells per carbon atom
    at a depth 1 mm from the surface of steel
6 c1 = 1/(n1*a^3);     // Atomic percent of carbon at
    the surface, per metre cube
7 c2 = 1/(n2*a^3);     // Atomic percent of carbon at a
     depth 1 mm from the surface, per metre cube
8 dx = 1e-03;     // Thickness of the steel bar, m
9 J = -D*((c2-c1)/dx);     // Fick's first law giving
    outward flux of carbon through the Steel bar, C
    atoms per metre square per second
10 J_uc = J*a^2*60;     // The number of carbon atoms
    diffusing through each unit cell per minute
11 printf("\nThe number of carbon atoms diffusing
    through each unit cell per minute = %2d atoms per
     minute", J_uc);
12 // Result
13 // The number of carbon atoms diffusing through each
     unit cell per minute = 82 atoms per minute
```

Scilab code Exa 6.4 Diffusion through a cylinder

```
1 // Scilab Code Ex6.4 Diffusion through a cylinder:
    Page-199 (2010)
2 r = 12;     // Radius of cylindrical crystal, mm
3 A1 = %pi*r^2;     // Cross-sectional area for
    diffusion through the cylinder, milli-metre
    square
4 t = 4e-07;     // Assume effective thickness of the
    surface to be 4 angstrom = two atomic diameters,
    mm
5 A2 = 2*%pi*r*t;     // Cross-sectional area for
```

```
    diffusion along the surface, milli-metre square
ratio = A2/A1;     // Ratio of two cross-sectional
    areas
printf("\nThe ratio of two cross-sectional areas =
    %4.2e", ratio);
// Result
// The ratio of two cross-sectional areas = 6.67e
    -008
```

Scilab code Exa 6.5 Diffusion length of Li in Ge

```
// Scilab Code Ex6.5 Diffusion length of Li in Ge:
    Page-203 (2010)
D = 1e-010;     // Diffusion coefficient for Li in Ge
    , metre square per sec
t = 1*60*60;     // Time taken by diffusing Li to
    travel diffusion depth, sec
T = 500+273;     // absolute temperature of the
    system, kelvin
x = sqrt(D*t);     // Diffusion length of Li in Ge, m
printf("\nThe diffusion length of Li in Ge = %1.0e m
    ", x);
// Result
// The diffusion length of Li in Ge = 6e-004 m
```

Scilab code Exa 6.6 Diffusion time of Li in Ge

```
// Scilab Code Ex6.6 Diffusion time of Li in Ge:
    Page-203 (2010)
D = 1e-010;     // Diffusion coefficient for Li in Ge
    , metre square per sec
T = 500+273;     // Absolute temperature of the
    system, kelvin
```

```
4 x = 0.2e-03;     // Diffusion length of Li in Ge, m
5 // Diffusion length is given by
6 // x = sqrt(D*t), solving for t
7 t = x^2/D;     // Time taken by diffusing Li to
    travel diffusion depth of 0.2 mm, sec
8 printf("\nThe time taken by diffusing Li to travel
    diffusion depth of 0.2 mm = %3d s", t);
9 // Result
10 // The time taken by diffusing Li to travel
    diffusion depth of 0.2 mm = 400 s
```

Scilab code Exa 6.7 Diffusion coefficent of Cu in Al

```
1 // Scilab Code Ex6.7 Diffusion coefficent of Cu in
    Al: Page 206 (2010)
2 D0 = 0.25e-04;     // Pre-exponential diffusion
    constant independent of temperature, metre square
     per second
3 T = 550+273;     // Absolute temperature of the
    system, kelvin
4 R = 8.314;     // Molar gas constant, J/mol/K
5 Q = 121e+03;     // The activation energy for
    diffusion, joule per mole
6 t = 1*60*60;     // Time taken by Cu to diffuse into
    Al, sec
7 D = D0*exp(-Q/(R*T));     // Diffusion coefficient of
     Cu in Al at 550 degree celsius, metre square per
     sec
8 x = sqrt(D*t);     // Diffusion length of Cu in Al, m
9 printf("\nThe diffusion coefficient of Cu in Al at
    550 degree celsius = %4.2e metre square per sec",
     D);
10 printf("\nThe diffusion length of Cu in Al = %5.3f
    mm", x*1000);
11 // Result
```

```
12 // The diffusion coefficient of Cu in Al at 550
   degree celsius = 5.22e-013 metre square per sec
13 // The diffusion length of Cu in Al = 0.043 mm
```

Scilab code Exa 6.8 Activation energy for diffusion of Ag in Si

```
1 // Scilab Code Ex6.8 Activation energy for diffusion
   of silver in silicon: Page 206 (2010)
2 R = 8.314;     // Molar gas constant, J/mol/K
3 T1 = 1350+273;     // First temperature at which
   difuusion of Ag into Si takes place, kelvin
4 T2 = 1100+273;     // Second temperature at which
   difuusion of Ag into Si takes place, kelvin
5 DRR = 8;     // Ratio of diffusion rates of Ag in Si
   at T1 and T2
6 // As diffusion coefficient at temperature T1 is D1
   = D0*exp(-Q/(R*T1))
7 // and that at temperature T2 is D1 = D0*exp(-Q/(R*
   T2)), so that the diffusion rates ratio
8 // D1/D2 = DRR = exp(Q/R*(1/T2-1/T1)), solving for Q
   , we have
9 Q = R*log(DRR)/((1/T2-1/T1)*1000);     // Activation
   energy for diffusion of Ag in Si, kJ/mol
10 printf("\nThe activation energy for diffusion of Ag
   in Si = %3d kJ/mol", Q);
11 // Result
12 // The activation energy for diffusion of Ag in Si =
   154 kJ/mol
```

Scilab code Exa 6.9 Arrhenius rate law

```
// Scilab Code Ex6.9 Activation energy and diffusion constant of a diffusion system obeying Arrhenius rate law: Page 207 (2010)
R = 1.987;    // Molar gas constant, cal/mol/K
D_1100 = 8e-013;    // Diffusivity of Ga in Si at 1100 degree celsius, cm square per sec
D_1300 = 1e-010;    // Diffusivity of Ga in Si at 1300 degree celsius, cm square per sec
T1 = 1100+273;    // First temperature at which diffusion of Ga into Si takes place, kelvin
T2 = 1300+273;    // Second temperature at which diffusion of Ga into Si takes place, kelvin
// Arrehenius equation in log10 form is given by
// log10(D) = log10(D0)-Q/(2.303*R*T)    --- (a)
// Thus log10(D_1100) = log10(D0)-Q/(2.303*R*T1)    --- (i)
// log10(D_1300) = log10(D0)-Q/(2.303*R*T2)    --- (ii),
// On subtracting (ii) from (i), we get
// log10(D_1100/D_1300) = -Q/(2.303*R)*(1/T2-1/T1), solving for Q
Q = (2.303*log10(D_1100/D_1300)*R)/(1/T2-1/T1);    // Activation energy for diffusion of Ga in Si, cal/mol
// Putting Q in (ii) and solving for D0
D0 = exp(2.303*log10(D_1100)+Q/(R*T1))
// D0 = exp(2.303*log10(D_1300)+Q/(R*T2));    // Pre-exponential diffusion constant independent of temperature, cm square per sec
T = 1200+273;    // Temperature at which diffusion of Ga into Si is to be calculated, kelvin
// Substituting D0, Q, R and T in (a) and solving for D, we have
D = exp(2.303*log10(D0)-Q/(R*T));    // Diffusivity of the system, cm square per sec
printf("\nThe activation energy for diffusion of Ga in Si = %3d kcal/mol", Q/1000);
printf("\nThe pre-exponential diffusion constant, D0
```

```
    = %5d cm square per sec", D0);
22 printf("\nThe diffusivity of the system = %4.2e cm
   square per sec", D);
23 // Result
24 // The activation energy for diffusion of Ga in Si =
    103 kcal/mol
25 // The pre-exponential diffusion constant, D0 =
   24893 cm square per sec
26 // The diffusivity of the system = 1.05e-011 cm
   square per sec
```

Scilab code Exa 6.10 Activation energy for diffusion rates at different temperatures

```
1 // Scilab Code Ex6.10 Activation energy for
   diffusion rates at different temperatures: Page
   208 (2010)
2 R = 8.314;     // Molar gas constant, J/mol/K
3 T1 = 500+273;     // First temperature at which
   diffusion of A into B takes place, kelvin
4 T2 = 850+273;     // Second temperature at which
   diffusion of A into B takes place, kelvi
5 PDR = 1/4;     // Penetration depth ratio at 500
   degree celsius and 850 degree celsius
6 // x1/x2 = sqrt(D1/D2) i.e. PDR = sqrt(DRR), DRR is
   the diffusion rate ratio
7 // solving for DRR
8 DRR = PDR^2;     // Diffusion rate ratio D1/D2 of A
   in B
9 // As diffusion coefficient at temperature T1 is D1
   = D0*exp(-Q/(R*T1))
10 // and that at temperature T2 is D1 = D0*exp(-Q/(R*
   T2)), so that the diffusion rates ratio
11 // D1/D2 = DRR = exp(Q/R*(1/T2-1/T1)), solving for Q
   , we have
```

```
12 Q = R*log(DRR)/((1/T2-1/T1)*1000);     // Activation
   energy for diffusion of A in B, kJ/mol
13 printf("\nThe activation energy for diffusion of A
   in B = %5.2f kJ/mol", Q);
14 // Result
15 // The activation energy for diffusion of A in B =
   57.17 kJ/mol
```

Scilab code Exa 6.11 Time required for carburizing of steel

```
1 // Scilab Code Ex6.11 Time required for carburizing
   of steel: Page 209 (2010)
2 C0 = 0.0018;     // Intial carbon concentration of
   steel
3 Cx = 0.0030;     // Carbon concentration of steel at
   0.60 mm below the surface of the gear
4 Cs = 0.01;     // Carbon concentration of steel at
   the surface
5 x = 0.6e-03;     // Diffusion depth below the surface
    of the gear, m
6 D_927 = 1.28e-011;     // Diffusion coefficient for
   carbon in iron, metre square per sec
7 erf_Z = (Cs-Cx)/(Cs-C0);     // Error function of Z
   as a solution to Fick's second law
8 Z1 = 1.0, Z2 = 1.1;     // Preceding and succeeding
   values about Z from error function table
9 erf_Z1 = 0.8427, erf_Z2 = 0.8802;     // Preceding
   and succeeding values about erf_Z from error
   function table
10 Z = poly(0,'Z');
11 Z = roots((Z-Z1)/(Z2-Z1)-(erf_Z-erf_Z1)/(erf_Z2-
   erf_Z1));
12 // As Z = x/(2*sqrt(D_927*t)), where Z is a constant
    argument of error function as erf(Z)
13 // Solving for t, we have
```

```
14 t = (x/(2*Z))^2/D_927;     // Time necessary to
   increase the carbon content of steel, sec
15 printf("\nThe time necessary to increase the carbon
   content of steel = %3d minutes", t/60);
16 // Result
17 // The time necessary to increase the carbon content
    of steel = 110 minutes
```

Scilab code Exa 6.12 Carbon concentration of carburized steel at certain depth

```
1 // Scilab Code Ex6.12 Carbon concentration of
   carburized steel at certain depth: Page 210
   (2010)
2 C0 = 0.0020;     // Initial carbon concentration of
   steel
3 Cs = 0.012;     // Carbon concentration of steel at
   the surface
4 t = 10*60*60;     // Carburizing time of steel, sec
5 x = 0.06*25.4*1e-03;     // Diffusion depth below the
    surface of the gear, mm
6 D_927 = 1.28e-011;     // Diffusion coefficient for
   carbon in iron, metre square per sec
7 Z = x/(2*sqrt(D_927*t)), // A constant argument of
   error function as erf(Z)
8 Z1 = 1.1, Z2 = 1.2;     // Preceding and succeeding
   values about Z from error function table
9 erf_Z1 = 0.8802, erf_Z2 = 0.9103;     // Preceding
   and succeeding values about erf_Z from error
   function table
10 efZ = poly(0,'efZ');
11 efZ = roots((efZ-erf_Z1)/(erf_Z2-erf_Z1)-(Z-Z1)/(Z2-
   Z1)); // Error function of Z as a solution to
   Fick's second law
12 Cx = poly(0,'Cx');
```

```
13 Cx = roots(efZ-(Cs-Cx)/(Cs-C0)); // Carbon
   concentration of carburized steel at 0.06 inch
   depth
14 printf("\nThe carbon concentration of carburized
   steel at 0.06 inch depth = %4.2f percent", Cx
   *100);
15 // Result
16 // The carbon concentration of carburized steel at
   0.06 inch depth = 0.31 percent
```

Scilab code Exa 6.13 Depth of decarburization below the surface of steel

```
1 // Scilab Code Ex6.13 Depth of decarburization below
    the surface of steel: Page 211 (2010)
2 C2 = 0.012;    // Initial carbon concentration of
   steel
3 Cx = 0.008;    // Carbon concentration of carburized
    steel at x metre depth
4 Cs = 0;    // Carbon concentration of steel at the
   surface
5 t = 5*60*60;    // Carburizing time of steel, sec
6 D_927 = 1.28e-011;    // Diffusion coefficient for
   carbon in iron, metre square per sec
7 erf_Z = abs((Cs-Cx)/(C2-Cs));    // Error function
   of Z as a solution to Fick's second law
8 Z1 = 0.65, Z2 = 0.70;    // Preceding and succeeding
    values about Z from error function table
9 erf_Z1 = 0.6420, erf_Z2 = 0.6778;    // Preceding
   and succeeding values about erf_Z from error
   function table
10 Z = poly(0,'Z');
11 Z = roots((Z-Z1)/(Z2-Z1)-(erf_Z-erf_Z1)/(erf_Z2-
   erf_Z1));
12 // As Z = x/(2*sqrt(D_927*t)), where Z is a constant
    argument of error function as erf(Z)
```

```
13 // Solving for x, we have
14 x = Z*2*sqrt(D_927*t); // Depth of decarburization
      below the surface of steel, m
15 printf("\nThe minimum depth upto which post
      machining is to be done = %4.2f mm", x*1000);
16 // Result
17 // The minimum depth upto which post machining is to
       be done = 0.66 mm
```

Scilab code Exa 6.14 Diffusion depth of P type semiconductor

```
1  // Scilab Code Ex6.14 Diffusion depth of P-type
      semiconductor (B into Si): Page 212 (2010)
2  C0 = 0;     // Initial boron concentration of silicon
3  Cx = 1e+17;     // Boron concentration at depth x
      below the silicon surface
4  Cs = 1e+18;     // Boron concentration of silicon at
      the surface
5  T = 1100+273;     // Absolute temperature of the
      system, kelvin
6  t = 2*60*60;     // Time taken to diffuse boron into
      silicon, sec
7  D_1100 = 4e-013;     // Diffusion coefficient for
      boron in silicon, cm square per sec
8  erf_Z = abs((Cs-Cx)/(Cs-C0));     // Error function
      of Z as a solution to Fick's second law
9  Z1 = 1.1, Z2 = 1.2;     // Preceding and succeeding
      values about Z from error function table
10 erf_Z1 = 0.8802, erf_Z2 = 0.9103;     // Preceding
      and succeeding values about erf_Z from error
      function table
11 Z = poly(0,'Z');
12 Z = roots((Z-Z1)/(Z2-Z1)-(erf_Z-erf_Z1)/(erf_Z2-
      erf_Z1));
13 // As Z = x/(2*sqrt(D_927*t)), where Z is a constant
```

```
    argument of error function as erf(Z)
// Solving for x, we have
x = Z*2*sqrt(D_1100*t); // Diffusion depth of boron
    into silicon
printf("\nThe diffusion depth of boron into silicon
    = %4.2e cm", x);
// Result
// The diffusion depth of boron into silicon = 1.25e
    -004 cm
```

Chapter 7

Lattice or Atomic Vibrations

Scilab code Exa 7.1 Cut off frequency of the linear lattice of a solid

```
1 // Scilab Code Ex7.1 Cut-off frequency of the linear
      lattice of a solid: Page-238 (2010)
2 v = 3e+03;     // Velocity of sound in the solid, m/s
3 a = 3e-010;     // Interatomic distance, m
4 // As cut-off frequency occurs at k = %pi/a and k =
     2*%pi/lambda, this gives
5 lambda = 2*a;     // Cut-off wavelength for the solid
     , m
6 f = v/lambda;     // Cut-off frequency (v = f*lambda)
      for the linear lattice, hertz
7 printf("\nThe cut-off frequency for the linear
     lattice of a solid = %1.0e Hz", f);
8 // Result
9 // The cut-off frequency for the linear lattice of a
      solid = 5e+012 Hz
```

Scilab code Exa 7.2 Comparison of frequency of waves in a monoatomic and diatomic linear systems

```
1 // Scilab Code Ex7.2 Comparison of frequency of
    waves in a monoatomic and diatomic linear systems
    : Page-238 (2010)
2 a = 2.5e-010;     // Interatomic spacing between two
    identical atoms, m
3 v0 = 1e+03;     // Velocity of sound in the solid, m/
    s
4 lambda = 10e-010;     // Wavelength of the sound wave
    , m
5 omega = v_0*2*%pi/lambda;     // Angular frequency of
     sound wave in a monoatomic lattice, rad per sec
6 printf("\nThe frequency of sound waves in a
    monoatomic lattice = %4.2e rad/sec", omega);
7 // For acoustic waves in a diatomic lattice (M = m),
     the angular frequency, omega = 0 at k = 0 and
8 // omega = (2*K/m)^(1/2)     --- (i)     at k = %pi
    /(2*a)
9 // As v0 = a*(2*K/m)^(1/2)     --- (ii)
10 // From (i) and (ii), we have
11 omega_min = 0;     // Angular frequency of acoustic
    waves at k = 0, rad per sec
12 omega_max = v0/a;     // Angular frequency of
    acoustic waves at k = %pi/(2*a), rad per sec
13 printf("\n\nThe frequency of acoustic waves wave in
    a diatomic lattice :\n %d rad/sec for k = 0 \n %1
    .0e rad/sec for k = pi/(2*a)", omega_min,
    omega_max);
14 // For optical waves in a diatomic lattice (M = m),
    the angular frequency
15 // omega = sqrt(2)*(2*K/m)^(1/2)     --- (iii)     at
    k = 0
16 // As v0 = a*(2*K/m)^(1/2)     --- (iv)
17 // From (iii) and (iv), we have
18 omega_max = sqrt(2)*v_0/a;     // Angular frequency
    of optical waves at k = 0, rad per sec
19 // For optical waves in a diatomic lattice (M = m),
    the angular frequency
20 // omega = (2*K/m)^(1/2)     --- (iii)     at k = %pi
```

```
    /(2*a)
21 // As v0 = a*(2*K/m)^(1/2)    --- (iv)
22 // From (iii) and (iv), we have
23 omega_min = v_0/a;     // Angular frequency of
    optical waves at k = %pi/(2*a), rad per sec
24 printf("\n\nThe frequency of optical swaves wave in
    a diatomic lattice :\n %4.2e rad/sec for k = 0 \n
     %1.0e rad/sec for k = pi/(2*a)", omega_max,
    omega_min);
25 // Result
26 // The frequency of sound waves in a monoatomic
    lattice = 6.28e+012 rad/sec
27
28 // The frequency of acoustic waves wave in a
    diatomic lattice :
29 // 0 rad/sec for k = 0
30 // 4e+012 rad/sec for k = pi/(2*a)
31
32 // The frequency of optical swaves wave in a
    diatomic lattice :
33 // 5.66e+012 rad/sec for k = 0
34 // 4e+012 rad/sec for k = pi/(2*a)
```

Scilab code Exa 7.3 Reflection of electromagentic radiation from a crystal

```
1 // Scilab Code Ex7.3 Reflection of electromagentic
    radiation from a crystal: Page-239(2010)
2 c = 3.0e+08;     // Speed of electromagnetic wave in
    vacuum, m/s
3 a = 5.6e-010;     // Lattice parameter of NaCl
    crystal, m
4 Y = 5e+010;     // Modulus of elasticity along [100]
    direction of NaCl, newton per metre square
5 m = 23;     // Atomic weight of sodium, amu
6 M = 37;     // Atomic weight of chlorine, amu
```

```
amu = 1.67e-027;     // Kg equivalent of 1 amu
K = a*Y;     // Force constant of springs when the extension along [100] direction is neglected, N/m
omega_plus_max = (2*K*(1/(M*amu)+1/(m*amu)))^(1/2);     // The maximum angular frequency of the reflected electromagnetic radiation, rad per sec
lambda = 2*%pi*c/omega_plus_max;     // The wavelength at which the electromagnetic radiation is strongly reflected, m
printf("\nThe wavelength at which the electromagnetic radiation is strongly reflected by the crystal = %4.2e m", lambda);
// Result
// The wavelength at which the electromagnetic radiation is strongly reflected by the cystal = 3.88e-005 m
```

Chapter 8

Diffraction of Waves and Particles by Crystals

Scilab code Exa 8.1 Shortest wavelength and frequency of X rays from accelerating potential

```
1 // Scilab Code Ex08.1 Determination of shortest
    wavelength and frequency of X-rays from
    accelerating potential Page-250 (2010)
2 V = 50e+03;     // Accelerating potential, volt
3 c = 3e+08;     // Speed of light in free space
4 Lambda_min = 1.24e-06/V;     // Minimum wavelength,
    metre
5 F_max = c/Lambda_min;     // Maximum frequency, Hz
6 printf("\nThe shortest wavelength present in X-rays
    = %4.2f angstrom", Lambda_min/1D-10);
7 printf("\nThe maximum frequency present in X-rays =
    %3.1e Hz", F_max);
8 // Result
9 // The shortest wavelength present in X-rays = 0.25
    angstrom
10 // The maximum frequency present in X-rays = 1.2e+19
     Hz
```

Scilab code Exa 8.2 Impinging electrons on the target and characteristics of X rays

```
1 // Scilab Code Ex8.2 Calculation of impinging
    electrons on the target and characteristics of X-
    rays Page-253 (2010)
2 I = 2.5e-03;     // Current through X-ray tube,
    ampere
3 V = 6e+03;     // Potential across the X-ray  tube,
    volt
4 e = 1.6e-19;     // Charge on an electron, coulomb
5 m = 9.1e-031;     // mass of an electron, kg
6 t = 1;     // Transit time, second
7 Q = I*t;     // Total charge flowing per second
    through the x-ray tube, coulomb
8 n = Q/e;     // Number of electrons striking the
    target per second
9 // We have eV = 1/2*m*v^2 (stopping potential =
    maximum Kinetic energy)
10 // Solving for v
11 v = sqrt(2*e*V/m);     // speed of electrons striking
     the target, m/s
12 Lambda_min = 1.24e-06/V;     // Minimum wavelength of
     X-rays produced, metre
13 printf("\nThe number of electrons striking the
    target = %4.2e",n);
14 printf("\nThe velocity of electrons striking the
    target = %4.2e m/s",v);
15 printf("\nThe shortest wavelength present in X-rays
    = %4.2e m", Lambda_min);
16 // Result
17 // The number of electrons striking the target =
    1.56e+016
18 // The velocity of electrons striking the target =
```

```
    4.59e+007 m/s
19 // The shortest wavelength present in X-rays = 2.07e
    -010 m
```

Scilab code Exa 8.3 Wavelength of characteristic X rays

```
1  // Scilab Code Ex8.3 Calculation of wavelength of
       characteristic X-rays Page-253 (2010)
2  h = 6.626e-034;     // Planck's constant, Js
3  c = 3e+08;     // Speed of light in free space, m/s
4  e = 1.602e-019;     // Charge on an electron, coulomb
5  E_K = -78;     // Energy of K shell for platinum, keV
6  E_L = -12;     // Energy of L shell for platinum, keV
7  E_M = -3 ;     // Energy of M shell for platinum, keV
8  E_K_alpha = E_L - E_K;     // Energy of K_alpha line,
       keV
9  E_K_beta = E_M - E_K;     // Energy of K_beta line,
       keV
10 // We have E = h*f, where f = c/Lambda this implies
       E = h*c/lambda
11 // Solving for Lambda
12 // Lambda = h*c/E
13 lambda_K_alpha = h*c/(E_K_alpha*e*1e+03);     //
       Wavelength of K_alpha line, metre
14 lambda_K_beta = h*c/(E_K_beta*e*1e+03);     //
       Wavelength of K_beta line, metre
15 printf("\nThe wavelength of K_alpha line = %4.2f
       angstrom", lambda_K_alpha/1D-10);
16 printf("\nThe wavelength of K_beta line  = %4.2f
       angstrom", lambda_K_beta/1D-10);
17 // Result
18 // The wavelength of K_alpha line = 0.19 angstrom
19 // The wavelength of K_beta line  = 0.17 angstrom
```

Scilab code Exa 8.4 Atomic number of an unknown element

```
1 // Scilab Code Ex8.4 Calculation of atomic number of
     an unknown element Page-255 (2010)
2 lambda_Pt = 1.321e-010;     // Wavelength of L_alpha
    line of Pt, m
3 Z_Pt = 78;     // Atomic number of platinum
4 b = 7.4;     // Constant
5 lambda_x = 4.174e-010;     // Wavelength of unknown
    element, m
6 // We have f = [a*(Z-b)]^2          (Moseley's law)
7 // As f_Pt = c/lambda_Pt = [a*(Z_Pt-b)]^2
8 // Similarly f_x = c/lambda_x = [a*(Z_x-b)]^2
9 // Dividing f_Pt by f_x and solving for x
10 Z_x = b + sqrt(lambda_Pt/lambda_x)*(Z_Pt-b);     //
    Atomic number of unknown element
11 printf("\nThe atomic number of unknown element = %4
    .1f", Z_x);
12 // Result
13 // The atomic number of unknown element = 47.1
```

Scilab code Exa 8.5 Wavelength of copper using Moseley law

```
1 // Scilab Code Ex8.5 Calculation of wavelength of
    copper using Moseley's law Page-256 (2010)
2 c = 3.0e+08;     // Speed of light, m/s
3 lambda_W = 210e-010;     // Wavelength of K_alpha
    line of W, m
4 Z_W = 74;     // Atomic number of tungsten
5 Z_Cu = 29;     // Atomic number of copper
6 b = 1;     // Constant for K-series
```

```
7 // f_W = c/lambda_W = (a*73)^2, The frequency
    K_alpha line for tungsten, Hz
8 // f_Cu = c/lambda_Cu = (a*28)^2, The frequency
    K_alpha line for copper, Hz
9 // Dividing f_W by f_Cu and solving for lambda_Cu
10 lambda_Cu = ((Z_W-b)/(Z_Cu-b))^2*lambda_W; //
    Wavelength of K_alpha line of Cu, m
11 printf("\nThe wavelength of K_alpha line of copper =
    %4.0f angstrom", lambda_Cu/1D-10);
12 // Result
13 // The wavelength of K_alpha line of copper = 1427
    angstrom
```

Scilab code Exa 8.6 Atomic number from wavelength using Moseley law

```
1 // Scilab Code Ex8.6 Calculation of atomic number
    from wavelength using Moseley's law Page-256
    (2010)
2 c = 3.0e+08;     // Speed of light, m/s
3 h = 6.626e-034;     // Planck's constant, Js
4 epsilon_0 = 8.85e-012;     // Absolute electrical
    permittivity of free space, coulomb square per
    newton per metre square
5 m = 9.1e-031;     // Mass of an electron, kg
6 e = 1.6e-019;     // Charge on an electron, C
7 lambda = 0.7185e-010;     // Wavelength of K_alpha
    line of unknown element
8 b = 1;     // Mosley's constant for K-series
9 n_1 = 1; n_2 = 2;     // Lower and upper energy
    levels
10 // We know that f = c/lambda = m*e^4*(Z-b)^2/(8*
    epsilon_0^2*h^3)*(1/n_2^2-1/n_1^2)
11 // This implies that lambda = (8*epsilon_0^2*c*h^3/(
    m*e^4*(Z-b)^2*(1/n_2^2-1/n_1^2))
12 // Solving for Z
```

```
13 Z = sqrt(8*epsilon_0^2*c*h^3/(m*e^4*lambda*(1/n_1
   ^2-1/n_2^2)))+b; // Atomic number unknown element
14 printf("\nThe atomic number unknown element = %2d",
   Z);
15 // Result
16 // The atomic number unknown element = 42
```

Scilab code Exa 8.7 Wavelengths of tin and barium using Moseley law

```
1 // Scilab Code Ex8.7 Calculation of wavelengths of
   tin and barium using Moseley's law Page-257
   (2010)
2 Z_Fe = 26;     // Atomic number of iron
3 Z_Pt = 78;     // Atomic number of platinum
4 Z_Sn = 50;     // Atomic number of tin
5 Z_Ba = 56;     // Atomic number of barium
6 b = 1;    // Mosley's constant for K-series
7 lambda_Fe = 1.93e-010;     // Wavelength of K_alpha
   line of Fe
8 lambda_Pt = 0.19e-010;     // Wavelength of K_alpha
   line of Pt
9 // From Moseley's Law,
10 // f = a*(Z-1)^2. This implies lambda  = C*1/(Z-1)^2
11 // so that lambda_Fe  = C*1/(Z_Fe-1)^2 and lambda_Sn
     = C*1/(Z_Sn-1)^2
12 // Dividing lambda_Sn by lambda_Fe and solving for
   lambda_Sn
13 lambda_Sn = (Z_Fe-1)^2/(Z_Sn-1)^2*lambda_Fe;     //
   Wavelength of K_alpha line for tin, m
14 lambda_Ba = (Z_Pt-1)^2/(Z_Ba-1)^2*lambda_Pt;     //
   Wavelength of K_alpha line for barium, m
15 printf("\nThe wavelengths of tin and barium = %3.1f
   angstrom and %4.2f angstrom respectively",
   lambda_Sn/1D-10, lambda_Ba/1D-10);
16 // Result
```

```
17 // The wavelengths of tin and barium = 0.5 angstrom
   and 0.37 angstrom respectively
```

Scilab code Exa 8.8 Percentage transmitted energy of X rays

```
1 // Scilab Code Ex8.8 Percentage transmitted energy
   of X-rays: Page 259 (2010)
2 mu = 139;     // Attenuation co-efficient of
   aluminium, per metre
3 x = 0.005;     // Thickness of aluminium sheet, m
4 // If X% is the intensity of the X-ray transmitted
   through the aluminium sheet then
5 // X% = I/I_0
6 // or X/100 = exp(-absorb_coeff*x)
7 // Solving for X
8 X = 100*exp(-mu*x);     // Transmitted percentage of
   X-rays
9 printf("\nThe intensity of the X-ray transmitted
   through the aluminium sheet = %g percent", round(
   X));
10 // Result
11 // The intensity of the X-ray transmitted through
   the aluminium sheet = 50 percent
```

Scilab code Exa 8.9 Thickness of lead piece by using two equal intensity X ray wavelengths

```
1 // Scilab code Ex8.9 : Determination of thickness of
    lead piece by using two equal intensity X-ray
   wavelengths : Page 259 (2010)
2 lambda_1 = 0.064e-010;     // First wavelength of X-
   ray, metre
```

```
3 lambda_2 = 0.098e-010;     // Second wavelength of X-
    ray, metre
4 I1_ratio_I2 = 3;     // Ratio of attenuated beam
    intensity
5 mu_m1 = 0.164;     // Mass absorption coefficient for
     first wavelength, metre square per kg
6 mu_m2 = 0.35;     // Mass absorption coefficient for
    second wavelength, metre square per kg
7 d = 0.164;     // Density of the lead, kg per metre
    cube
8 mu1 = mu_m1*d;     // absorption co-efficient of the
    lead for first wavelength, per metre
9 mu2 = mu_m2*d;     // absorption co-efficient of the
    lead for second wavelength, per metre
10 x = poly(0,"x");     // Declare 'x' as the thickness
    variable
11 // Now I = exp(-ac*x) thus
12 // I1_ratio_I2 = exp(-ac_1*x)/exp(-ac_2*x)
13 // or 3 = exp(2109.24)*x this implies
14 // 2104.24*x = log(3) and assume
15 p = 2104.24*x-log(3);
16 printf("\nThe thickness of lead piece = %4.2e m",
    roots(p));
17 // Result
18 // The thickness of lead piece = 5.22e-004 m
```

Scilab code Exa 8.10 Angle of reflection by using wavelength of X rays

```
1 // Scilab code Ex8.10: Determining angle of
    reflection by using wavelength of X-ray Page 261
    (2010)
2 lambda = 0.440e-010;     // Wavelength of X-rays, m
3 d = 2.814e-010;     // Interplanar spacing of
    rocksalt crystal, m
4 // 2*d*sin(theta) = n*lambda     **Bragg's law, n is
```

```
    the order of diffraction
// Solving for theta, we have
// theta = asin(n*lambda/(2*d))
// Declare a function for converting angle into
    degrees and minutes
function [d,m] = degree_minute(n)
          d = int(n);
          m = (n-int(n))*60;
endfunction
for n = 1:1:5    // For diffraction order from 1 to
    5
     theta = asind(n*lambda/(2*d));    // Bragg's
        angle
     [deg, mint] = degree_minute(theta);    // Call
        conversion function
     printf("\nTheta%d = %2d degree(s), %2d minute(s)
        ", n, deg, mint);
end
// Result
// Theta1 =  4 degree(s), 29 minute(s)
// Theta2 =  8 degree(s), 59 minute(s)
// Theta3 = 13 degree(s), 33 minute(s)
// Theta4 = 18 degree(s), 13 minute(s)
// Theta5 = 23 degree(s),  0 minute(s)
```

Scilab code Exa 8.11 Wavelength of diffracted X rays

```
// Scilab code Ex8.11: Determining the wavelength of
    diffracted X-rays Page 262 (2010)
d = 2.814e-010;    // Interplanar spacing of
    rocksalt crystal, m
theta = 9;    // Bragg's angle, degree
// 2*d*sin(theta) = n*lambda    **Bragg's law, n is
    the order of diffraction
// Solving for lambda, we have
```

```
// lambda = 2*d*sin(theta)/n;
printf("\nThe first four wavelengths of diffracted beam are:");
for n = 1:1:5    // For diffraction order from 1 to 5
    lambda = 2*d*sind(theta)/n;    // Wavelength of X-rays, m
    if lambda >= 0.2e-010 & lambda <= 1.0e-010 then
        printf("\nLambda%d = %6.4e angstrom", n, lambda/1D-10);
    end
end
// Result
// The first four wavelengths of diffracted beam are:
// Lambda1 = 8.8041e-001 angstrom
// Lambda2 = 4.4021e-001 angstrom
// Lambda3 = 2.9347e-001 angstrom
// Lambda4 = 2.2010e-001 angstrom
```

Scilab code Exa 8.12 Reciprocal lattice parameters from 2D direct lattice parameters

```
// Scilab code Ex8.12: Reciprocal lattice parameters from 2-D direct lattice parameters Page 277 (2010)
a = 3e-010;    // First lattice parameter of direct lattice
b = 5e-010;    // Second lattice parameter of direct lattice
theta = 60;    // Angle between two lattice vectors of the direct lattice
// if a_prime and b_prime are the lattice vectors for the reciprocal lattice, then
// a_prime*a = 2*%pi and a_prime*b = 0
```

```
7 // Similarly , b_prime*b = 2*%pi and b_prime*a = 0
8 // Solving for a_prime and b_prime , we have
9 a_prime = 2*%pi/(a*cosd(90-theta)); // Lattice
    vector for reciprocal lattice , per metre
10 b_prime = 2*%pi/(b*cosd(90-theta)); // Lattice
    vector for reciprocal lattice , per metre
11 printf("\nThe reciprocal lattice vectors are:\n
    a_prime = %5.2f per angstrom and b_prime = %5.2f
    per angstrom", a_prime*1e-010, b_prime*1e-010);
12 // Result
13 // The reciprocal lattice vectors are:
14 // a_prime =  2.42 per angstrom and b_prime =  1.45
    per angstrom
```

Scilab code Exa 8.13 Bragg angle and the indices of diffraction of Powder Lines

```
1 // Scilab code Ex8.13: Bragg angle and the indices
    of diffraction of Powder Lines Page 285 (2010)
2 n = 1;    // Cosider first order diffraction
3 a = 6e-010;    // First lattice parameter of direct
    lattice , m
4 lambda = 1.54e-010;    // Wavelength used in
    diffraction of X-rays by Powder Method, m
5 // Declare a function for converting angle into
    degrees and minutes
6 function [d,m] = degree_minute(n)
7         d = int(n);
8         m = (n-int(n))*60;
9 endfunction
10 // Calculate the hkl and hence interpalnar spacing '
    d' for three lowest powder lines
11 printf("\nThe Bragg angles and the indices of
    diffraction for the three lowest powder lines are
    :");
```

```
for h = 0:1:2
    for k = 0:1:2
        for l = 0:1:1
            if (modulo(h,2) == 1 & modulo(k,2) == 1
               & modulo (l,2) == 1) | (modulo(h,2)
               == 0 & modulo(k,2) == 0 & modulo (l
               ,2) == 0) then
                if (h <> 0) then
                    N = h^2+k^2+l^2;
                    d = a/sqrt(N);     // Interplanar
                        spacing, metre
                    theta = asind(n*lambda/(2*d));
                    [deg, mint] = degree_minute(
                       theta);     // Call conversion
                        function
                    printf("\nd[%d%d%d] = %4.2e and
                       theta[%d%d%d] = %d deg %d min
                       ", h, k, l, d, h, k, l, deg,
                       mint);
                end
            end
        end
    end
end
// Result
// The Bragg angles and the indices of diffraction
   for the three lowest powder lines are:
// d[111] = 3.46e-010 and theta[111] = 12 deg 50 min
// d[200] = 3.00e-010 and theta[200] = 14 deg 52 min
// d[220] = 2.12e-010 and theta[220] = 21 deg 17 min
```

Scilab code Exa 8.14 Minimum distance from the centre of the Laue pattern

```
// Scilab code Ex8.14: Minimum distance from the
```

```
    centre of the Laue pattern  of an fcc crystal
    Page 289 (2010)
n = 1;    // Consider the first order diffraction
a = 4.5e-010;    // Lattice parameter for fcc
    lattice , m
V = 50e+03;    // Potential difference across the X-
    ray tube , volt
D = 5;    // Crystal to film distance , cm
h = 1, k = 1, l = 1;    // Incides for the planes of
     maximum spacing
lambda_min = 1.24e-06/V;    // The cut-off
    wavelength of X-rays , m
d_111 = a/sqrt(h^1+k^2+l^2);
theta_111 = asind(n*lambda_min/(2*d_111));
// As tan(2*theta_111) = x/D, solving for x
x = D*tand(2*theta_111);    // // Minimum distance
    from the centre of Laue pattern
printf("\nThe minimum distance from the centre of
    the Laue pattern at which reflections can occur
    from the planes of maximum spacing = %4.2f cm", x
    );
// Result
// The minimum distance from the centre of the Laue
    pattern at which reflections can occur from the
    planes of maximum spacing = 0.48 cm
```

Scilab code Exa 8.15 Unit cell height along the axis of a rotation photograph

```
// Scilab code Ex8.15: Calculating unit cell height
    along the axis of a rotation photograph Page 291
    (2010)
n = 1;    // Consider the first order diffraction of
     X-rays
S = [0.29,0.59,0.91,1.25,1.65,2.12];    // An array
```

```
    of heights of first six layers above(below) the
    zero layer, cm
R = 3;          // Radius of the camera, cm
lambda = 1.54e-08;     // Wavelength of the X-rays,
    cm
// For an a-axis rotation photograph, the unit cell
    parameter is given by
// a = n*lambda/S(n)*(R^2 + S(n)^2)^(1/2)
// Calculate 'a' for six different values of n from
    1 to 6
for n = 1:1:6
    a = (n*lambda/S(n))*(R^2 + S(n)^2)^(1/2);
end
printf("\nThe unit cell height of the crystal = %2.0
    f angstrom", a/1D-8);

// Result
// The unit cell height of the crystal = 16 angstrom
```

Scilab code Exa 8.16 Diffraction of thermal neutrons from planes of Ni crystal

```
// Scilab code Ex8.16: Diffraction of thermal
    neutrons from planes of Ni crystal  Page 294
    (2010)
k = 1.38e-023;     // Boltzmann constant, J/mol/K
h = 6.626e-034;     // Planck's constant, Js
theta = 28.5;     // Bragg's angle, degree
a = 3.52e-010;     // Lattice parameter of fcc
    structure of nickel, m
m_n = 1.67e-027;     // Rest mass of neutron, kg
// For fcc lattice, the interplanar spacing is given
    by
d = a/sqrt(3);     // Interplanar spacing of Ni, m
// Bragg's equation for first order diffraction (n =
```

```
    1) is
lambda = 2*d*sind(theta);     // Bragg's law, m
// From kinetic interpretaion of temperature, we
   have
// (1/2)*m*v^2 = (3/2)*k*T    -- (a)
// Further from de-Broglie relation
// lambda = h/(m*v)           -- (b)
// From (a) and (b), solving for T, we have
T = h^2/(3*m_n*k*lambda^2);    // Effective
   temperature of the neutrons, K
printf("\nThe effective temperature of neutrons = %d
    K", T);
// Result
// The effective temperature of neutrons = 168 K
```

Scilab code Exa 8.17 Diffraction of electrons from fcc crystal planes

```
// Scilab code Ex8.17: Diffraction of electrons from
    fcc crystal planes Page 295 (2010)
// Declare a function for converting angle into
   degrees and minutes
function [d,m] = degree_minute(n)
          d = int(n);
          m = (n-int(n))*60;
endfunction
h = 6.626e-034;     // Planck's constant, Js
m = 9.1e-031;     // Rest mass of electron, kg
e = 1.602e-019;     // charge on an electron, coulomb
a = 3.5e-010;     // Lattice parameter of fcc crystal
   , m
V = 80;     // Accelerating potential for electrons,
   volt
lambda = h/sqrt(2*m*e*V);     // de-Broglie
   wavelength of electrons, m
d_111 = a/sqrt(3);     // Interplanar spacing for
```

```
    (111) planes of fcc crystal, m
// Bragg's equation for first order diffraction (n = 1) is
// lambda = 2*d_111*sind(theta_111);     // Bragg's law, m
theta_111 = asind(lambda/(2*d_111));     // Bragg's angle, degree
[deg, mint] = degree_minute(theta_111);     // Call conversion function
printf("\nThe Bragg angle for electron diffraction = %d deg %d min", deg, mint);
// Result
// The Bragg angle for electron diffraction = 19 deg 50 min
```

Chapter 9

Thermal Properties of Materials

Scilab code Exa 9.1 Exception of Dulong Petit law at room temperature

```
1 // Scilab Code Ex9.1 Exception of Dulong-Petit law
    at room temperature: Page-303(2010)
2 h = 6.626e-034;     // Planck's constant, joule
    second
3 k = 1.38e-023;     // Boltzmann constant, joule/mol/
    kelvin
4 T = 300;     // Room temperature, kelvin
5 f_Ag = 4.0e+012;     // Vibrational frequency for
    silver, cycles/second
6 f_Dia = 2.4e+013;     // Vibrational frequency for
    diamond, cycles/second
7 E_Ag = h*f_Ag;     // Vibrational Energy for silver,
    joule
8 E_Dia = h*f_Dia;     // Vibrational Energy for
    diamond, joule
9 E_th = k*T;     // Thermal energy at room temperature
    , joule
10 if E_th > E_Ag & E_th < E_Dia then
11     printf("\nSince E_Ag < kT and E_Dia > kT,
```

```
        therefore,");
12      printf("\nSilver metal obeys the Dulong Petit
        law at room temperature while diamond does
        not.");
13  end
14  // Result
15  //Since E_Ag < kT and E_Dia > kT, therefore,
16  // Silver metal obeys the Dulong Petit law at room
    temperature while diamond does not.
```

Scilab code Exa 9.2 Specific heat of copper from Debye temperature

```
1  // Scilab Code Ex9.2 Specific heat of copper from
   Debye temperature: Page-311(2010)
2  h = 6.626e-034;     // Planck's constant, joule
   second
3  k = 1.38e-023;     // Boltzmann constant, joule/mol/
   kelvin
4  T = 30;     // Given temperature, kelvin
5  N = 6.023e+023;     // Avogadro's number
6  R = N*k;     // Universal gas constant, joule/kelvin
7  v_l = 4.76e+03;     // Longitudinal velocity of
   lattice waves, m/s
8  v_t = 2.32e+03;     // Tranverse velocity of lattice
   waves,
9  rho = 8.9e+03;     // Density of copper, kg per metre
    cube
10 A_Cu = 63.5;     // Gram atomic mass of Cu, g
11 M = A_Cu*1e-03;     // Mass of 1 mole of Cu-atoms, kg
12 V = M/rho;     // Volume of copper, metre cube
13 theta_D = (h/k)*((9*N)/((4*%pi*V)*((1/v_l^3)+(2/v_t
   ^3))))^(1/3);     // Debye temperature of copper,
   K
14 C_v = 12/5*%pi^4*R*(T/theta_D)^3;     // Specific
   heat of copper, kJ/kmol/kelvin
```

```
15 printf("\nThe specific heat of copper = %4.2f kJ/
     kmol/kelvin", C_v);
16 // Result
17 // The specific heat of copper = 1.33 kJ/kmol/kelvin
```

Scilab code Exa 9.3 Vibrational frequency and molar heat capacity of diamond

```
1 // Scilab Code Ex9.3 Vibrational frequency and molar
     heat capacity of diamond: Page-312(2010)
2 h = 6.626e-034;     // Planck's constant, joule
     second
3 k = 1.38e-023;     // Boltzmann constant, joule/mol/
     kelvin
4 T = 10;     // Given temperature, kelvin
5 N = 6.023e+023;     // Avogadro's number
6 R = N*k;     // Universal gas constant, joule/kelvin
7 theta_D = 2230;     // Debye temperature for diamond,
     kelvin
8 f_D = k*theta_D/h;     // Debye frequency of diamond,
     hertz
9 C_v = 12/5*%pi^4*R*1e+03*(T/theta_D)^3;     //
     Specific heat of diamond, J/kmol/kelvin
10 printf("\nThe highest possible vibrational frequency
     of diamond = %4.2e per second", f_D);
11 printf("\nThe molar specific heat of diamond = %5.3f
     J/kmol/kelvin", C_v);
12 // Result
13 // The highest possible vibrational frequency of
     diamond = 4.64e+013 per second
14 // The molar specific heat of diamond = 0.175 J/kmol
     /kelvin
```

Scilab code Exa 9.4 Debye temperature of copper at low temperature

```
1 // Scilab Code Ex9.4 Debye temperature of copper at
    low temperature: Page-312(2010)
2 k = 1.38e-023;     // Boltzmann constant, joule/mol/
    kelvin
3 N = 6.023e+023;     // Avogadro's number
4 R = N*k;     // Universal gas constant, joule/kelvin
5 C_vl = 4.6e-02;     // Lattice specific heat, J/kmol/
    K
6 // Lattice specific heat C_vl = Molar lattice
    specific heat, C_v
7 // or     12/5*%pi^4*R/(5*theta_D^3) = C_vl
8 // solving for theta_D, we have
9 theta_D = (12*%pi^4*R*1e+03/(5*C_vl))^(1/3);     //
    Debye temperature of copper at low temperature, K
10 printf("\nDebye temperature of copper at low
    temperature = %3d K", theta_D);
11 // Result
12 // Debye temperature of copper at low temperature =
    348 K
```

Scilab code Exa 9.5 Debye temperature for gold

```
1 // Scilab Code Ex9.5 Debye temperature for gold :
    Page-313(2010)
2 h = 6.626e-034;     // Planck's constant, Js
3 k = 1.38e-023;     // Boltzmann constant, joule/mol/
    kelvin
4 N = 6.023e+023;     // Avogadro's number
5 R = N*k;     // Universal gas constant, joule/kelvin
6 M = 197e-03;     // Gram atomic weight of gold, g
7 rho = 1.9e+04;     // Density of gold, kg per metre
    cube
8 V = M/rho;     // Volume of gold, metre cube
```

```
9  v = 2100;     // Velocity of sound in gold medium, m/
     s
10 theta_D = h*v/k*(9*N/(12*%pi*V))^(1/3);     // Debye
     temperature for gold, K
11 printf("\nDebye temperature of gold = %3d K",
     theta_D);
12 // Result
13 // Debye temperature of gold = 242 K
```

Scilab code Exa 9.6 Heat transference into rock salt at low temperature

```
1  // Scilab Code Ex9.6 Heat transference into rock
     salt at low temperature: Page-313(2010)
2  A = 464;     // Atomic specific heat of rock salt,
     cal g/mol/kelvin
3  theta_D = 281;     // Debye temperature of rock salt,
      K
4  delta_T = 10;     // Rise in temperature in each
     class interval, K
5  // Define a function which returns lattice specific
     heat at constant volume
6  function[C_vl] = lattice_SH(T)
7      C_vl = A*(T/theta_D)^3;
8  endfunction
9  Q = 0;     // Initialize heat accumulator to zero,
     cal
10 for t = 10:10:40
11     mean_temp = (t + (t + 10))/2;     // Calculate
         mean temperature of each class interval, K
12     Q = Q + 2*delta_T*lattice_SH(mean_temp);     //
         Acuumulate heat for each step
13 end
14 printf("\nThe amount of heat required to raise the
     temperature of 2 gmol of Rock salt from 10K to 50
     K = %5.2f cal", Q);
```

```
15 // Result
16 // The amount of heat required to raise the
      temperature of 2 gmol of Rock salt from 10K to 50
      K = 63.99 cal
```

Chapter 10

Free Electrons in Crystals

Scilab code Exa 10.1 Particle moving in one dimensional potential well

```
1 // Scilab Code Ex10.1 Particle Moving in One-
    Dimensional Potential Well: Page-328 (2010)
2 a = 10^-3; //Separation between the walls of the
    well, m
3 m = 10^-9; // Mass of the dust particle, kg
4 t = 100; // Average time for successive collisons
    with the wall, s
5 h = 6.626*10^-34; // Plank's constant, Js
6 v = a/t; // Velocity of the particle inside the
    potential well, m/s
7 E = 1/2*m*v^2; // Kinetic energy of the particle, J
8 // For one-dimensional potential well, the energy
    eigen value is given by
9 //              E = h^2*n^2/(8*m*a^2)
10 // Solving for n
11 n = sqrt((8*m*a^2*E)/h^2) // Quantum number
    corresponding to the energy eigen value E
12 disp (n, "The quantum number described by this
    motion is:")
13 // Result
14 // The quantum number described by this motion is:
```

```
15 //     3.018D+16
```

Scilab code Exa 10.2 Motion of a ground state electron in a 3D potential well

```
1 // Scilab Code Ex 10.2 Motion of a ground state
     Electron in a 3-D Potential Well: Page-329 (2010)
2 a = 0.5*10^-10;     // length of the potential box, m
3 h = 6.626*10^-34; // Plank's Constant, Js
4 m = 9.1*10^-31;  // Mass of an Electron, kg
5 // In 3-D, the three quantum numbers nx, ny and nz
     each will have value equal to 1 for lowest energy
      state
6 nx = 1;     // Quantum number corresponding to x-
     direction
7 ny = 1;     // Quantum number corresponding to y-
     direction
8 nz = 1;     // Quantum number corresponding to z-
     direction
9 EG = h^2*(nx^2+ny^2+nz^2)/(8*m*a^2); // Energy eigen
      value for 3-D potential, J
10 EeV = EG/1.6D-19; // Convert energy from joule to eV
11 disp (EeV, "The lowest energy of an electron
     confined to move in a 3D-potential box, in eV, is
      : ")
12 //Result
13 // The lowest energy of an electron confined to move
      in a 3D-potential box, in eV, is :
14 // 452.30641
```

Scilab code Exa 10.3 Motion of an electron excited next to the ground state in a 3D potential well

```
// Scilab Code Ex 10.3 Motion of an Electron excited next to the ground state in a 3-D Potential Well : Page-329 (2010)
a = 1D-10;     // length of the cubic potential box, m
h = 6.626*10^-34; // Plank's Constant, Js
m = 9.1*10^-31;  // Mass of an Electron, kg
k = 1.38D-23;     // Boltzmann Constant, J/mol-K
// In 3-D, the three quantum numbers nx, ny and nz will have values 1, 1 and 2 respectively for first excited energy state
nx = 1;     // Quantum number corresponding to x-direction
ny = 1;     // Quantum number corresponding to y-direction
nz = 2;     // Quantum number corresponding to z-direction
EE = h^2*(nx^2+ny^2+nz^2)/(8*m*a^2); // Energy eigen value for 3-D potential for first excited state, J
// As EE(next to the lowest) = 3/2 (k/T), where T is the absolute temperature
// Solving for T
T = 2/3*1/k*EE;     // Absolute temperature at which energy next to the lowest energy state = 3/2 (k/T), K
EeV = EE/1.6D-19; // Convert energy from joule to eV
disp (EeV, "The first excited state energy of the electron confined to move in a 3D-potential box, in eV, is : ")
disp (T, "The temperature at which the average energy becomes equal to first excited state energy, in K, is : ")
//
//Result
// The first excited state energy of the electron confined to move in a 3D-potential box, in eV, is :
```

```
20 //     226.15321
21 // The temperature at which the average energy
      becomes equal to first excited state energy, in K
      , is :
22 //      1748044.1
```

Scilab code Exa 10.4 Degeneracy of energy level

```
1  // Scilab Code Ex 10.4 Degeneracy of Energy Level:
      Page-332 (2010)
2  // Function to find the factorial of a number
3  function[f] = fact(num)
4      f = 1;
5      for i = 1:1:num
6          f = f*i;
7      end
8  endfunction
9
10 // Fucntion to determine degenerate energy states
11 function[degstates] = degno(a, b, c)// degno takes
      three arguments
12     if  a == b & b == c then // check if all the
         values are same
13         degeneracy = 3;
14         degstates = fact(3)/fact(degeneracy); //
            calculate degenerate states
15     end
16     if a == b | b == c | c == a then    // check if
         any two values are equal
17         degeneracy = 2;
18         degstates = fact(3)/fact(degeneracy); //
            calculate degenerate states
19     end
20     if a ~= b & b ~= c then    // check if all the
         values are different
```

```
21          degeneracy = 1;
22          degstates = fact(3)/fact(degeneracy); //
               calculate degenerate states
23      end
24  endfunction
25  //
26  clc
27  coef = 38;      // Coefficient of H^2/(8*m*a^2)
28  nx = zeros(1,5);      // Quantum number corresponding
       to x-direction
29  ny = zeros(1, 5);     // Quantum number corresponding
       to y-direction
30  nz = zeros(1,5);      // Quantum number corresponding
       to z-direction
31  deg = zeros(1,5);      // Variable to store the
       degeneracy of states
32  count = 1;      // set the counter
33  sum = 0;       // initialize the sum
34  // Look for all the possible set of values for nx,
       ny and ny
35  for i = 1:1:10
36      for j = 1:1:10
37          for k = 1:1:10
38        // Check for the condition and avoid
             repetition of set of values
39              if ((i^2+j^2+k^2==coef) & (i+j+k)> sum)
                   then
40                  nx(1,count)=i; // Save current i
                       value
41                  ny(1,count)=j; // Save current j
                       value
42                  nz(1,count)=k; // Save current k
                       value
43                  deg(1,count) = degno(i, j, k); //
                       Save degeneracy for given set of
                       values
44                  count = count + 1; // Increment the
                       counter
```

```
45                  sum = i + j + k; // Add the three
                        values of quantum numbers
46              end
47          end
48      end
49  end
50  printf("\nThe %d set(s) of values of quantum number
        are : \n", count-1);
51  deg_states = 0; // Intialize the variable
52  for i = 1:1:count-1
53      printf("\nnx = %d, ny = %d, nz = %d\n", nx(1,i),
            ny(1,i), nz(1,i));
54      deg_states = deg_states + deg(1,i); //
          Accumulate the degeneracy
55  end
56      printf("\nThe given energy level is %d-fold
          degenerate.", deg_states);
57  //Result
58  // The 2 set(s) of values of quantum number are :
59  //          nx = 1, ny = 1, nz = 6
60  //          nx = 2, ny = 3, nz = 5
61  // The energy level is 9-fold degenerate
```

Scilab code Exa 10.5 Fermi energy of zinc at absolute zero

```
1  //Scilab Code Ex 10.5 Fermi energy of zinc at
       absolute zero: Page-335 (2010)
2  d = 7.13D+3;    // Density of Zn, in kg per m cube
3  M = 65.4D-3;       // Atomic weight of Zn, kg/mol
4  me = 9.1D-31;   // Mass of an electron, kg
5  meff = 0.85*me;  // Effective mass of the electron
       in zinc, kg
6  v = 2;     // valency of divalent (Zn) metal
7  N = 6.023D+23;      // Avogadro's Number
8  h = 6.626D-34;      //Plank's constant, in Js
```

```
n = v*d*N/M;     // Number of electrons per unit
   volume
Ef = h^2/(2*meff)*(3*n/(8*%pi))^(2/3);  //Fermi
   energy in zinc at absolute zero, J
EfeV = Ef/1.6D-19;     // Fermi energy in eV
Ebar = (3/5)*EfeV;     // Average energy of an
   electron at 0K, eV
disp(EfeV,"The fermi energy in zinc at absolute zero
   ,in eV, is : ");
disp(Ebar,"The average energy of an electron at 0K,
   in eV, is : ");
//Result
// The fermi energy in zinc at absolute zero,in eV,
   is :
// 11.110065
// The average energy of an electron at 0K,in eV, is
    :
// 6.6660389
```

Scilab code Exa 10.6 Electron probability above Fermi energy

```
// Scilab Code Ex 10.6 Electron probability above
   Fermi energy: Page-336 (2010)
k = 1.38D-23; // Boltzmann constant, in J/mol-K
FD = 0.10;     // Fermi-Dirac distribution
   probability for electrons
Efermi = 5.5;     // Fermi Energy of silver, in eV
E = Efermi + 0.01*Efermi;     // Allowed energy for
   electrons
dE = E - Efermi;     //Deviation of allowed energy
   from Fermi energy, in eV
DEeV = dE*1.6D-19; //Convert into joule
// The Fermi-Dirac distribution function as at any
   temperature T is given by
//             F(E) = FD = 1/(exp((E-Efermi)/kT)+1
```

```
10 // Solving for T
11 T = DEeV/(k*log(1/FD-1)); // Absolute temperature at
       which result follows, in K
12 disp(DEeV, dE, E);
13 disp(T, "The temperature at which the given
       probability is expected, in K, is :");
14 //Result
15 //  The temperature at which the given probability
       is expected, in K, is :
16 //  290.2212
```

Scilab code Exa 10.7 The electroic specific heat of Cu

```
1 // Scilab Code Ex 10.7 The Electroic Specific Heat
       of Cu: Page-341 (2010)
2 k = 1.38D-23;     //Boltzmann constant, in J/mol-K
3 N = 6.023D+23;     // Avogadro's Number
4 Efermi = 7.05;         // Fermi energy of copper, in
       eV
5 EFeV = Efermi*1.6D-19; // Fermi energy conversion,
       in J
6 T1 = 4;     //Lower value of temperature, in K
7 T2 = 300;     //Upper value of temperature, in K
8 Ce4 = (%pi^2*k^2*T1)/(2*EFeV)*N;     // Electronic
       specific heat at 4K, J/mol/K
9 Ce100 = (%pi^2*k^2*T2)/(2*EFeV)*N;     // Electronic
       specific heat at 100K, J/mol/K
10 disp(Ce4, "The Electronic specific heat at 4K, in J/
       mol/K is :");
11 disp(Ce100, "The Electronic specific heat at 100K,
       in J/mol/K is :");
12 //Result
13 // The Electronic specific heat at 4K, in J/mol/K is
       :
14 // 0.0020072
```

```
15 // The Electronic specific heat at 100K, in J/mol/K
   is :
16 // 0.1505404
```

Scilab code Exa 10.8 Electrical resitivity of sodium metal

```
1  // Scilab Code Ex 10.10 Electron mobility inside
   conductors : Page-346 (2010)
2  e = 1.6D-19;       // Electronic charge, in C
3  m = 9.1D-31;       // Eelctronic mass, in kg
4  res = 1.54D-8;     // Electrical resistivity of
   silver, in ohm metre
5  E = 100;           // Electric field applied along
   the length of the wire, V/m
6  n = 5.8D+28;       // Number of conduction electrons
   per unit volume, per metre cube
7  mu = 1/(res*n*e); // Mobility of electron through
   silver, metre square per volt-sec
8  vd = mu*E;         // Average drift velocity of
   electrons, m/s
9  t = mu*m/e;        // Relaxation time of the electron
   , s
10 disp(mu, "The mobility of electron through silver,
   in metre square per V-s, is : ");
11 disp(vd, "The average drift velocity of electrons,
   in m/s, is : ");
12 disp(t, "c ");
13 // Result
14 // The mobility of electron through silver, in metre
    square per V-s, is :
15 //         0.0069973
16 // The average drift velocity of electrons, in m/s,
   is :
17 //         0.6997313
18 // The average drift velocity of electrons, in m/s,
```

```
    is :
19 //          3.980D-14
```

Scilab code Exa 10.9 Electrical conductivity of Cu

```
1  // Scilab Code Ex 10.9 Electrical Conductivity of Cu
     : Page-345 (2010)
2  e = 1.6D-19;      // Electronic charge, C
3  N = 6.023D+23;      // Avogardro's number
4  d = 8920;         // Density of Copper, kg per metre
     cube
5  A = 63.5;          // Atomic weight of copper, g/mole
6  I = 10;          // Current through uniform copper wir
     , A
7  D = 16D-4;          //Diameter of circular cross-
     section of copper wire, m
8  R = D/2;             // Radius of circular cross-
     section of copper wire, m
9  n = d*N/63.5*1D+3;     // The number of electrons per
      unit volume in copper, per metre cube
10 J = I/(%pi*R^2);     // Current density of electrons
     in copper, ampere per metre square
11 vd = J/(n*e);          // Drift velocity of electrons
     in copper, metre per second
12 disp(J,"The current density of electrons in copper,
     in ampere per metre square, is : ");
13 disp(vd,"The drift velocity of electrons in copper,
     in metre per second, is : ");
14 //Result
15 //The current density of electrons in copper, in
     ampere per metre square, is :
16 //          4973592
17 // The drift velocity of electrons in copper, in
     metre per second, is :
18 //           0.0003674
```

Scilab code Exa 10.10 Electron mobility inside conductors

```
1 // Scilab Code Ex 10.10 Electron mobility inside
    conductors : Page-346 (2010)
2 e = 1.6D-19;       // Electronic charge, in C
3 m = 9.1D-31;       // Eelctronic mass, in kg
4 res = 1.54D-8;     // Electrical resistivity of
    silver, in ohm metre
5 E = 100;           // Electric field applied along
    the length of the wire, V/m
6 n = 5.8D+28;       // Number of conduction electrons
    per unit volume, per metre cube
7 mu = 1/(res*n*e); // Mobility of electron through
    silver, metre square per volt-sec
8 vd = mu*E;         // Average drift velocity of
    electrons, m/s
9 t = mu*m/e;        // Relaxation time of the electron
    , s
10 disp(mu, "The mobility of electron through silver,
    in metre square per V-s, is : ");
11 disp(vd, "The average drift velocity of electrons,
    in m/s, is : ");
12 disp(t, "c ");
13 // Result
14 // The mobility of electron through silver, in metre
     square per V-s, is :
15 //         0.0069973
16 // The average drift velocity of electrons, in m/s,
    is :
17 //         0.6997313
18 // The average drift velocity of electrons, in m/s,
    is :
19 //         3.980D-14
```

Scilab code Exa 10.11 Lorentz number calculation of a solid

```
1 // Scilab Code Ex 10.11 Lorentz number calculation
    of a solid: Page-347 (2010
2 e = 1.6D-19;          // Electronic charge, in C
3 k = 1.38D-23;         // boltzmann constant, J/mol-K
4 T = 293;              // Absolute temperature of the
    solid
5 K = 390;      // Thermal conductivity of copper at 293
     K, W/m-K
6 l = 0.5;      // Lenght of the copper wire, m
7 d = 0.3D-3;       // Diameter of cross-section of Cu, m
8 r = d/2;      // Radius of copper wire, m
9 R = 0.12;       // Resistance of copper wire, ohm
10 // As R = 1/con*l/(%pi*r^2)
11 // Solving for R
12 con = l/(%pi*r^2*R);      // Conductance of copper,
    per ohm per metre
13 // The Lorentz number is defined as the ratio of the
     Thermal conductivity to the
14 // Electrical conductivity of a solid per degree
    rise in temperature
15 Lexp = K/(con*T);      // Experimental value of
    Lorentz number, watt ohm per kelvin square
16 Lth = %pi^2/3*(k/e)^2;  // Thoeretical value of
    Lorentz number value, watt ohm per kelvin square
17 disp(Lexp,"The experimetal value of Lorentz number,
    in watt ohm per kelvin square, is :");
18 disp(Lth,"The theoretical value of Lorentz number,
    in watt ohm per kelvin square, is :");
19 printf("\nThe theoretical value of Lorentz number is
     %f times higher than the experimental one.\n",
    Lth/Lexp);
20 // Result
```

```
21 // The experimetal value of Lorentz number, in watt
     ohm per kelvin square, is :
22 //     2.258D-08
23 // The theoretical value of Lorentz number, in watt
     ohm per kelvin square, is :
24 //     2.447D-08
25 // The theoretical value of Lorentz number is  times
      higher than the experimental one.
26 //     1.083817
```

Scilab code Exa 10.12 Increase in electrical resistivity of a metal with temperature

```
1 // Scilab Code Ex 10.12 Increase in electrical
     resistivity of a metal with temperature: Page-349
      (2010)
2 function [res] = final_res(T)
3     alpha = 0.0001;     // Temperature co-efficient
        of resistance
4     resi = 0;         // Initial resistivity of the
        nichrome which is an arbitray
5     //constant and can be taken to be zero
6 res = resi + alpha*T;     // Final resistivity of the
     nichrome as function of T
7 endfunction
8 T1 = 300;     // Initial temperature of nichrome, K
9 T2 = 1000;     // Final temperature of nichrome, K
10 res300 = final_res(T1); // Final resistivity of the
     nichrome at 300 K
11 res1000 = final_res(T2); // Final resistivity of the
     nichrome at 1000 K
12 percent_res = (res1000 - res300)*100;     //
     Percentage increase in resistivity
13 printf("\nThe percentage increase in the resistivity
     of nichrome is %d percent", percent_res);
```

```
14 // Result
15 // The percentage increase in the resistivity of
      nichrome is 7 percent
```

Scilab code Exa 10.13 Thermionic emission of a filament

```
1  // Scilab Code Ex 10.13 Thermionic emission of a
      filament: Page-352 (2010)
2  e = 1.6D-19;     // Electronic charge, C
3  m = 9.1D-31;     // Mass of the electron, kg
4  k = 1.38D-23;    // Boltzmann constant, J/mol-K
5  h = 6.626D-34;     // Plank's constant, Js
6  W = 4.5;     // Work function of tungsten filament,
      eV
7  D = 1D-4;     // Diameter of the filament, m
8  r = D/2;     // Radius of the filament, m
9  T = 2400;     // Temperature of the filament, K
10 l = 0.05;     // Length of the filament, m
11 A =  4*%pi*e*m*k^2/h^3;      // A constant expressed
      in ampere per metre square
12                                 // per kelvin square
13 a = 2*%pi*r*l;          // Surface area of the
      filament, meter square
14 J = A*T^2*exp(-e*W/(k*T));     // Electronic current
      density of the filament,
15                                 // ampere per metre
                                      square
16 I = a*J;     // Electric current due to thermionic
      emission, ampere
17 disp(I,"The electric current due to thermionic
      emission, in A, is : ");
18 // Result
19 // The electric current due to thermionic emission,
      in A, is :
20 //      0.0392404
```

Scilab code Exa 10.14 Hall coefficient of sodium based on free electron model

```
1 // Scilab Code Ex 10.14 Hall coefficient calculation
      of sodium based on free electron model: Page-353
      (2010)
2 e = 1.6D-19;     // Electronic charge, C
3 a = 4.28D-10;     // lattice parameter (side) of the
    unit cell of sodium crystal, m
4 N = 2;          // Number of atoms per unit cell in
    bcc structure of sodium
5 n = N/a^3;     // Number of electrons per unit volume
      for the sodium crystal, per metre cube
6 RH = -1/(n*e);     // Hall coefficient of sodium,
    metre cube per coulomb
7 disp(RH,"The Hall coefficient of sodium , in metre
    cube per coulomb, is : ");
8 // Result
9 // The Hall coefficient of sodium , in metre cube
    per coulomb, is :
10 //      -2.450D-10
```

Chapter 11

Band Theory

Scilab code Exa 11.2 Ratio between kinetic energy of an electron in 2D square lattice

```
1 // Scilab Code Ex11.2 Determining ratio between K.E.
     of an electron in 2D square lattice: Page-370
    (2010)
2 h = 6.626e-034;     // Planck's constant, Js
3 m = 9.1e-031;     // Mass of an electron, kg
4 a = 1;     // For simplicity assuming lattice
    parameter to be unity, m
5 // Case-I when k_x = k_y = %pi/a
6 k_x = %pi/a, k_y = %pi/a;     // Wave numbers in X-
    and Y- directions, rad per metre
7 E1 = h^2/(8*%pi^2*m)*(k_x^2 + k_y^2);     // Energy
    of the electron inside a Brilliouin Zone, J
8 // Case-II when k_x = %pi/a and k_y = 0
9 k_x = %pi/a, k_y = 0;     // Wave numbers in X- and Y
    - directions, rad per metre
10 E2 = h^2/(8*%pi^2*m)*(k_x^2 + k_y^2);     // Energy
    of the electron inside a Brilliouin Zone, J
11 E_ratio = E1/E2;     // Ratio between K.E. of an
    electron in 2D square lattice
12 printf("\nThe ratio between K.E. of an electron in 2
```

```
    D square lattice = %1d", E_ratio);
13 // Result
14 // The ratio between K.E. of an electron in 2D
    square lattice = 2
```

Chapter 13

Semiconducting Properties of Materials

Scilab code Exa 13.3 Intrinsic concentration of charge carriers in semiconductors

```
1 // Scilab Code Ex13.3 Intrinsic concentration of
    charge carriers in semiconductors: Page-432
    (2010)
2 k = 1.38e-023;     // Boltzmann constant, J/mol/K
3 h = 6.626e-034;     // Planck's constant, Js
4 eV = 1.6e-019;     // Joule equivalent of 1 eV
5 T = 300;     // Room temperature, kelvin
6 m_0 = 9.1e-031;     // Rest mass of an electron, kg
7 m_e = 0.12*m_0;     // Effective mass of electron, kg
8 m_h = 0.28*m_0;     // Effective mass of electron, kg
9 E_g = 0.67;     // Energy gap of Ge, eV
10 n_i = 2*(2*%pi*k*T/h^2)^(3/2)*(m_e*m_h)^(3/4)*exp(-
    E_g*eV/(2*k*T));     // Intrinsic carrier
    concentration of Ge, per metre cube
11 printf("\nThe intrinsic carrier concentration of Ge
    = %3.1e per metre cube", n_i);
12 // Result
13 // The intrinsic carrier concentration of Ge = 4.7e
```

```
+018 per metre cube
```

Scilab code Exa 13.4 Comparison of intrinsic carrier densities of two semiconductors

```
// Scilab Code Ex13.4 Comparison of intrinsic
   carrier densities of two semiconductors at room
   temperature  Page-433 (2010)
eV = 1.6e-019;     // Joule equivalent of 1 eV
m = 9.1e-031;     // Rest mass of an electron, kg
m_e = m;     // Effective mass of electron, kg
m_h = m;     // Effective mass of electron, kg
Eg_A = 0.36;     // Energy gap of A, eV
Eg_B = 0.72;     // Energy gap of B, eV
k = 1.38e-023;     // Boltzmann constant, J/mol/K
h = 6.626e-034;     // Planck's constant, Js
k_T = 0.052/2;     // Thermal energy, eV
// As n_i_ratio = ni_A/ni_B = exp(-Eg_A/(2*k_T))/exp
   (-Eg_A/(2*k_T))
n_i_ratio = exp(-Eg_A/(2*k_T))/exp(-Eg_B/(2*k_T));
      // Intrinsic carrier density ratio of A and B
printf("\nThe ratio of intrinsic carrier density =
   %4d ", n_i_ratio);
// Result
// The ratio of intrinsic carrier density = 1015
```

Scilab code Exa 13.5 Shift in fermi level with change in concentration of impurities

```
// Scilab Code Ex13.5 Shift in position of fermi
   level with change in concentration of impurities:
     Page-436 (2010)
k_T = 0.03;     // Thermal energy, eV
```

```
dE_Fv = 0.4;    // Energy difference between fermi
    level and topmost valence level, eV
// The hole concentration in P-type material is
// p = N_A = N_v*exp(-EF-Ev)/(k_T) = N_v*exp(-dE_Fv)
    /(k_T)
// The new value of hole concentration in P-type
    material is
// p_prime = 3*N_A = N_v*exp(-EF_prime-Ev)/(k_T)= =
    N_v*exp(-dE_F_primev)/(k_T)
// Solving for dE_F_primev by removing exponetial
    term
dE_F_primev = dE_Fv - k_T*log(3);    // Energy
    difference between new fermi level and topmost
    valence level, eV
printf("\nThe energy difference between new fermi
    level and topmost valence level = %5.3f eV",
    dE_F_primev);
// Result
// The energy difference between new fermi level and
    topmost valence level = 0.367 eV
```

Scilab code Exa 13.6 Electrical resistivity of Ge

```
// Scilab Code Ex13.6 Electrical resistivity of Ge:
    Page-439 (2010)
e = 1.602e-019;    // Charge on an elctron, C
n_i = 2.37e+019;    // Intrinsic carrier density of
    Ge at room temperature, per metre cube
mu_e = 0.38;    // Mobility of electrons, metre
    square per volt per second
mu_h = 0.18;    // Mobility of holes, metre square
    per volt per second
T = 300;    // Room temperature, kelvin
sigma_i = n_i*e*(mu_e + mu_h);    // Intrinsic
    electrical conductivity, per ohm per metre
```

```
8 rho_i = 1/sigma_i;     // Intrinsic electrical
    resistivity , ohm-metre
9 printf("\nThe intrinsic electrical resistivity = %4
    .2f ohm-metre", rho_i);
10 // Result
11 // The intrinsic electrical resistivity = 0.47 ohm-
    metre
```

Scilab code Exa 13.7 Electrical conductivity of intrinsic and extrinsic Si

```
1 // Scilab Code Ex13.7 Electrical conductivity of
    intrinsic and extrinsic Si: Page-439 (2010)
2 NA = 6.023e+23;     // Avogadro's number
3 A_Si = 28.09e-03;     // Kilogram atomic mass of Si,
    kg
4 e = 1.602e-019;     // Charge on an elctron, C
5 n_impurity = 1/1e+08;     // Donor impurity atoms per
     Si atom
6 n_i = 1.5e+016;     // Intrinsic carrier density of
    Si at room temperature, per metre cube
7 mu_e = 0.13;     // Mobility of electrons, metre
    square per volt per second
8 mu_h = 0.05;     // Mobility of holes, metre square
    per volt per second
9 T = 300;          // Room temperature, kelvin
10 sigma_i = n_i*e*(mu_e + mu_h);     // Intrinsic
    electrical conductivity, per ohm per metre
11 Si_density = 2.23e+03;     // Density of silicon, kg
    per metre cube
12 N_Si = NA * Si_density/A_Si;     // Number of Si
    atoms, per metre cube
13 N_D = N_Si*n_impurity;     // Density of donor
    impurity, per metre cube;
14 sigma_ext = ceil(N_D)*e*mu_e;     // Extrinsic
    electrical conductivity of Si, per ohm per metre
```

```
15 printf("\nThe intrinsic electrical conductivity of
     Si = %5.3e per ohm per metre", sigma_i);
16 printf("\nThe extrinsic electrical conductivity of
     Si = %4.1f per ohm per metre", sigma_ext);
17 // Result
18 // The intrinsic electrical conductivity of Si =
     4.325e-004 per ohm per metre
19 // The extrinsic electrical conductivity of Si =
     10.0 per ohm per metre
```

Scilab code Exa 13.8 Resistance of intrinsic Ge Rod

```
1  // Scilab Code Ex13.8 Resistance of intrinsic Ge Rod
     : Page-440 (2010)
2  e = 1.602e-019;     // Charge on an elctron, C
3  T = 300;            // Room temperature, kelvin
4  l = 1e-02;      // Length of the Ge rod, m
5  b = 1e-03;      // Width of the Ge rod, m
6  t = 1e-03;      // Thickness of the Ge rod, m
7  n_i = 2.5e+019;     // Intrinsic carrier density of
     Ge, per metre cube
8  mu_e = 0.39;     // Mobility of electrons, metre
     square per volt per second
9  mu_h = 0.19;     // Mobility of holes, metre square
     per volt per second
10 sigma_i = n_i*e*(mu_e + mu_h);     // Intrinsic
     electrical conductivity, per ohm per metre
11 A = b*t;     // Surface area of the Ge rod, metre
     square
12 rho = 1/sigma_i;     // Electrical resistivity of Ge
     Rod, ohm-metre
13 R = rho*l/A;     // Resistance of Ge Rod, ohm
14 printf("\nThe resistance of Ge Rod = %3.1e ohm", R);
15 // Result
16 // The resistance of Ge Rod = 4.3e+003 ohm
```

Scilab code Exa 13.9 Hall effect in Si semiconductor

```
1 // Scilab Code Ex13.9 Hall effect in Si
    semiconductor: Page-442 (2010)
2 e = 1.602e-019;     // Charge on an elctron, C
3 T = 300;          // Room temperature, kelvin
4 R_H = -7.35e-05;     // Hall co-efficeint of Si
    specimen, metre cube per coulomb
5 sigma = 200;     // Electrical conductivity of Si,
    per ohm per metre
6 n = -1/(e*R_H);     // Electron density in the Si
    specimen
7 mu_e = sigma/(n*e);     // Electron mobility in the
    Si specimen, metre cube per volt per second
8 printf("\nThe density of electron = %3.1e metre cube
    ", n);
9 printf("\nThe mobility of electron = %4.2e metre
    cube per volt per second", mu_e);
10 // Result
11 // The density of electron = 8.5e+022 metre cube
12 // The mobility of electron = 1.47e-002 metre cube
    per volt per second
```

Scilab code Exa 13.10 Forward current of a pn diode using diode equation

```
1 // Scilab Code Ex13.10 Forward current of a p-n
    diode in terms of reverse saturation current
    using diode equation: Page-450 (2010)
2 e = 1.6e-019;     // Charge on an electron, coulomb
3 k = 1.38e-023;     // Boltzmann constant, J/mol/K
4 V = 0.35;     // Potential difference applied across
    a Ge diode, volt
```

```
5 T = 300;     // Room temperature, kelvin
6 Io = 1;          // Reverse saturation current, micro-
    ampere, for simplicity assume I0 = 1
7 Iv = Io*(exp(e*V/(k*T))-1);     // "Diode Equation"
    for net forward current, milliamperes
8 printf("\nThe net forward current = %4.2e Io", Iv);
9 // Result
10 // The net forward current = 7.49e+005 Io
```

Scilab code Exa 13.11 Voltage from net forward current using Diode Equation

```
1 // Scilab Code Ex13.11 Finding voltage from net
    forward current using Diode Equation: Page-450
    (2010)
2 e = 1.6e-019;     // Charge on an electron, coulomb
3 k = 1.38e-023;     // Boltzmann constant, J/mol/K
4 T = 300;     // Room temperature, kelvin
5 Io = 1;          // Reverse saturation current, micro-
    ampere, for simplicity assume I0 = 1
6 Iv = 0.9*Io;     // "Diode Equation" for net forward
    current, milliamperes
7 // As Iv = Io*(exp(e*V/(k*T))-1), solving for V
8 V = log(Iv/Io+1)*k*T/e;     // Potential difference
    applied across p-n junction, volt
9 printf("\nThe potential difference applied across p-
    n junction = %6.4f volt", V);
10 // Result
11 // The potential difference applied across p-n
    junction = 0.0166 volt
```

Chapter 14

Dielectric Properties of Materials

Scilab code Exa 14.1 Polarization of water molecule

```
1 // Scilab Code Ex14.1 Polarization of water molecule
    : Page-456 (2010)
2 NA = 6.023e+23;     // Avogadro's number
3 p = 6e-030;     // Dipole moment of water molecule, C
    -m
4 r = 1e-03;     // Radius of water molecule, m
5 M = 18e-03;         // Molecular weight of water, kg
6 d = 1e+03;     // Density of water, kg per metre cube
7 V = M/d;     // Volume of water, metre cube
8 // Now M/d metre cube volume will contain NA = 6.023
    e+023 water molecules, so that 4*%pi/3*(r^3)
    metre cube volume will contain
9 N = NA*d*4*%pi*r^3/(M*3);     // Number of water
    molecules per metre cube
10 P = N*p;     // Polarization of water molecules,
    coulomb per metre square
11 printf("\nThe polarization of water molecules = %3.1
    e coulomb per metre square", P);
12 // Result
```

```
13 // The polarization of water molecules = 8.4e-010
   coulomb per metre square
```

Scilab code Exa 14.2 Dielectric constant from electric polarizability of the atom

```
1 // Scilab Code Ex14.2 Calculating dielectric
   constant from electric polarizability of the atom
   : Page-464 (2010)
2 alpha_Kr = 2.18e-040;     // Electric polarizability
   of the Kr-atom, farad-metre square
3 NA = 6.023e+023;     // Avogadro's number
4 epsilon_0 = 8.85e-012;     // Electrical permittivity
    of free space, coulomb square per newton per
   metre square
5 N = NA/(22.4e-03);     // Number of Kr atoms per
   metre cube
6 epsilon_r = N*alpha_Kr/epsilon_0 + 1;     // Relative
    electrical permittivity of Kr specimen
7 printf("\nThe diectric constant of Kr specimen = %7
   .5f", epsilon_r);
8 // Result
9 // The diectric constant of Kr specimen = 1.00066
```

Scilab code Exa 14.3 Electric polarizability of a molecule from its susceptibility

```
1 // Scilab Code Ex14.3 Calculating electric
   polarizability of a molecule from its
   susceptibility: Page-464 (2010)
2 NA = 6.023e+023;     // Avogadro's number
```

```
epsilon_0 = 8.85e-012;    // Electrical permittivity
    of free space, coulomb square per newton per
    metre
chi = 0.985e-03;    // Electrical susceptibility of
    carbon-dioxide molecule
rho = 1.977;    // Density of carbon-dioxide, kg per
    metre cube
M = 44e-03;    // Molecular weight of CO2, kg
N = NA*rho/M;    // Number of molecules per unit
    volume, per metre cube
alpha = epsilon_0*chi/N;    // Total electric
    polarizability of carbon-dioxide, farad-metre
    square
printf("\nThe total electric polarizability of
    carbon-dioxide = %4.2e farad-metre square", alpha
    );
// Result
// The total electric polarizability of carbon-
    dioxide = 3.22e-040 farad-metre square
```

Scilab code Exa 14.4 Electric polarizability of oxygen atom

```
// Scilab Code Ex14.4 Calculating electric
    polarizability of Oxygen atom: Page-465 (2010)
e = 1.602e-019;    // Charge on an electron, coulomb
p = 0.5e-022;    // Dipole moment of oxygen atom, C-
    m
d = 4e-017;    // Distnace of the centre of negative
    charge cloud from the nucleus, m
epsilon_0 = 8.85e-012;    // Electrical permittivity
    of free space, coulomb square per newton per
    metre
// In equilibrium, Coulomb interaction = Lorentz
    force
// i.e.    8*e*E = (8*e)*(8*e)/(4*%pi*epsilon_0*d^2)
```

```
8 // Solving for E
9 E = 8*e/(4*%pi*epsilon_0*d^2);     // The strength of
      local electric field, volt per metre
10 // As p = alpha*E, solving for alpha
11 disp(E);
12 alpha = p/E;     // Atomic polarizability of oxygen,
     farad-metre square
13 printf("\nThe atomic polarizability of oxygen = %3.1
     e farad-metre square", alpha);
14 // Result
15 // The atomic polarizability of oxygen = 6.9e-048
     farad-metre square
```

Scilab code Exa 14.5 Dipolar polarization of HCl molecule

```
1 // Scilab Code Ex14.5 Dipolar polarization of HCl
     molecule: Page-470 (2010)
2 k = 1.38e-023;     // Boltzmann constant, J/mol/K
3 T = 300;     // Temperature of the HCl vapour, kelvin
4 N = 1e+027;     // Number of HCL molecuels per unit
     volume, per metre cube
5 E = 1e+06;     // Electric field strength to which
     the HCL vapour is subjected, volt/m
6 p = 3.46e-030;     // The dipole moment of HCl
     molecule,C-m
7 alpha_d = p^2/(3*k*T);     // Dipolar polarizability
     of HCl molecule, farad-metre square
8 // As P = N*p = N*alpha_d*E
9 P = N*alpha_d*E;          // Orientational or Dipolar
     polarization of HCl molecule, coulomb per metre
     square
10 E_M = p*E;     // Magnetic energy stored in the
     dipole-field system, joule
11 E_Th = k*T;     // Thermal energy of the HCl molecule
     , joule
```

```
12 a = E_M/E_Th;     // Ratio of magnetic energy to the
    thermal energy
13 printf("\nThe orientational polarization of
    molecules in HCl vapour = %4.2e coulomb per metre
     square", P);
14 printf("\nThe ratio of magnetic energy to the
    thermal energy = %f << 1", a);
15 // Result
16 // The orientational polarization of molecules in
    HCl vapour = 9.64e-007 coulomb per metre square
17 // The ratio of magnetic energy to the thermal
    energy = 0.000836 << 1
```

Scilab code Exa 14.6 Effect of molecular deformation on polarizability

```
1 // Scilab Code Ex14.6 Effect of molecular
    deformation on polarizability: Page-471 (2010)
2 alpha_309 = 2.42e-039;     // Polarizability of
    ammonia molecule at 309 K, farad-metre square
3 alpha_448 = 1.74e-039;     // Polarizability of
    ammonia molecule at 448 K, farad-metre square
4 k = 1.38e-023;     // Boltzmann constant, J/mol/K
5 T1 = 309;     // First temperature of the experiment,
     kelvin
6 T2 = 448;     // Second temperature of the experiment
    , kelvin
7 // As alpha = alpha_i + alpha_d = alpha_i + p^2/(3*k
    *T) = alpha_i + bta/T
8 // where bta = p^2/(3*k)
9 // Thus alpha_309 = alpha_i + bta/309 and alpha_448
    = alpha_i + bta/448
10 // Solving for bta
11 // bta(1/309-1/448) = alpha_309 - alpha_448
12 bta = poly(0, "bta");
13 bta = roots(bta*(1/309 - 1/448) - alpha_309 +
```

```
    alpha_448);     // bta = p^2/(3*k), farad-kelvin
    metre square
// Solving for alpha_i
alpha_i = alpha_309 - bta/309;     // Polarizability
    due to permanent dipole moment, farad-metre
    square
// Polarizability due to deformation of molecules =
    bta/T, bta = p^2/(3*k)
alpha_d_309 = bta/T1;     // Orientational
    polarizability at 309 K, farad-metre square
alpha_d_448 = bta/T2;     // Orientational
    polarizability at 448 K, farad-metre square
printf("\nThe polarizability due to permanent dipole
     moment = %4.1e farad-metre square", alpha_i);
printf("\nThe orientational polarization of ammonia
    at 309 K = %4.2e farad-metre square", alpha_d_309
    );
printf("\nThe orientational polarization of ammonia
    at 448 K = %4.2e farad-metre square", alpha_d_448
    );
// Result
// The polarizability due to permanent dipole moment
     = 2.3e-040 farad-metre square
// The orientational polarization of ammonia at 309
    K = 2.19e-039 farad-metre square
// The orientational polarization of ammonia at 448
    K = 1.51e-039 farad-metre square
```

Chapter 15

Optical Properties of Materials

Scilab code Exa 15.1 Photon count from Planck quantum law

```
1 // Scilab Code Ex15.1 Determining Photon number by
    using Planck quantum law: Page-486 (2010)
2 h = 6.626e-034;     // Planck's constant, Js
3 f = 1760e+03;     // Frequency of the radio
    transmitter, Hz
4 P = 10e+03;     // Power of radio transmitter, W
5 E = h*f;     // Energy carried by one photon from
    Planck's law, J
6 N = P/E;     // Number of photons emitted per second,
     number per second
7 printf("\nThe number of photons emitted per second =
    %4.2e", N);
8 // Result
9 // The number of photons emitted per second = 8.58e
    +030
```

Scilab code Exa 15.2 Inicient energy of photon in photoelectric effect

```
// Scilab Code Ex15.2 Finding suitable energy for Photoelectric Effect from Na metal: Page-486 (2010)
e = 1.602e-019;     // Charge on an electron, C
h = 6.626e-034;     // Planck's constant, Js
c = 3.0e+08;     // Speed of light in vacuum, m/s
W = 2.3*e;         // Work function of Na metal, J
lambda = 2800e-010;     // Wavelength of incident light, m
f = c/lambda;     // Frequency of the incident light, Hz
E = h*f;     // Energy carried by one photon from Planck's law, J
printf("\nThe energy carried by each photon of radiation = %4.2f eV", E/e);
if E > W then
    printf("\nThe photoelectric effect is possible..");
else
    printf("\nThe photoelectric effect is impossible..");
end
// Result
// The energy carried by each photon of radiation = 4.43 eV
// The photoelectric effect is possible..
```

Scilab code Exa 15.3 photon count for green wavelength of Hg

```
// Scilab Code Ex15.3 Finding number of photons for green wavelength of Hg: Page-487 (2010)
h = 6.626e-034;     // Planck's constant, Js
c = 3.0e+08;     // Speed of light in vacuum, m/s
lambda = 496.1e-09;     // Wavelength of green light of mercury, m
```

```
E_total = 1;     // Work done by photons from green
   light , J
f = c/lambda;     // Frequency of the green light , Hz
E = h*f;     // Energy carried by one photon from
   Planck's law , J
N = E_total/E;     // Number of photons of green
   light of Hg
printf("\nThe number of photons of green light of Hg
    = %3.1e", N);
// Result
// The number of photons of green light of Hg = 2.5e
   +018
```

Scilab code Exa 15.4 Photoelectric effect in a photocell

```
// Scilab Code Ex15.4 Photoelectric effect in a
   photocell: Page-487 (2010)
e = 1.602e-019;     // Charge on an electron , C
h = 6.626e-034;     // Planck's constant , Js
c = 3.0e+08;     // Speed of light in vacuum, m/s
lambda = 1849e-010;     // Wavelength of incident
   light , m
V_0 = 2.72;     // Stopping potential for emitted
   electrons , V
f = c/lambda;     // Frequency of incident radiation
   , Hz
E = h*f;     // Energy carried by one photon from
   Planck's law , J
T_max = e*V_0;     // Maximum kinetic energy of
   electrons , J
// We have, T_max = E - h*f_0 = h*f - W
f_0 = poly(0, "f_0");     // Declare f_0 as variable
f_0 = roots(T_max - E + h*f_0);     // Threshold
   frequency for Cu metal, Hz
W = h*f_0/e;     // Work function of Cu metal, eV
```

```
14 printf("\nThe threhold frequency for Cu metal = %4.2
     e Hz", f_0);
15 printf("\nThe work function of Cu metal = %g eV",
     round(W));
16 printf("\nThe maximum kinetic energy of
     photoelectrons = %4.2f eV", T_max/e);
17 // Result
18 // The threhold frequency for Cu metal = 9.65e+014
     Hz
19 // The work function of Cu metal = 4 eV
20 // The maximum kinetic energy of photoelectrons =
     2.72 eV
```

Scilab code Exa 15.5 Energy required to stimulate the emission of Na doublets

```
1 // Scilab Code Ex15.5 Energy required to stimulate
     the emission of Na d-lines: Page-497 (2010)
2 e = 1.6e-019;     // Charge on an electron, C
3 h = 6.626e-034;     // Planck's constant, Js
4 c = 3.0e+08;     // Speed of light in vacuum, m/s
5 lambda_mean = 5893e-010;     // Wavelength of
     incident light, m
6 delta_E = h*c/(lambda_mean*e);     // The energy of
     the electron which must be transferred to the
     atoms of Na
7 printf("\nThe energy which must be transferred to
     stimulate the emission of Na d-lines = %5.3f eV",
      delta_E);
8 // Result
9 // The energy which must be transferred to stimulate
      the emission of Na d-lines = 2.108 eV
```

Chapter 16

Magnetic Properties of Materials

Scilab code Exa 16.1 Response of copper to magnetic field

```
1 // Scilab Code Ex16.1 Response of Cu to magnetic
    field: Page-503 (2010)
2 H = 1e+06;          // Applied magnetic field in
    copper, A/m
3 chi = -0.8e-05;     // Magnetic susceptibility of
    copper
4 mu_0 = 4*%pi*1e-07;     // Magnetic permeability of
    free space, henry/metre
5 M = chi*H;     // Intesity of magnetization in copper
    , A/m
6 B = mu_0*(H + M);     // Magnetic flux density in
    copper, tesla
7 printf("\nThe magnetization of copper = %d A/m", M);
8 printf("\nThe magnetic flux density of copper = %5.3
    f T", B);
9 // Result
10 // The magnetization of copper = -8 A/m
11 // The magnetic flux density of copper = 1.257 T
```

Scilab code Exa 16.2 Diamagnetic susceptibility of copper

```
1 // Scilab Code Ex16.2 Diamagnetic susceptibility of
    copper: Page-512 (2010)
2 e = 1.6e-019;     // Charge on an electron, C
3 m = 9.1e-031;     // Mass of an electron, kg
4 mu_0 = 4*%pi*1e-07;     // Magnetic permeability of
    free space, henry/metre
5 Z = 1;     // Number of electrons contributing to the
     magnetic moment
6 r = 1e-010;     // Radius of copper atom, m
7 a = 3.608e-010;     // Lattice parameter of copper, m
8 // For FCC lattice of Cu, there are 4 atoms per unit
     cell
9 n = 4;     // Number of atoms per unit cell
10 N = n/a^3;     // Number of electrons per unit volume
    , per metre cube
11 chi_dia = -mu_0*Z*e^2*N*r^2/(6*m);     // Diamagnetic
     susceptibility of copper
12 printf("\nThe diamagnetic susceptibility of copper =
    %3.1e", chi_dia);
13 // Result
14 // The diamagnetic susceptibility of copper = -5.0e
    -006
```

Scilab code Exa 16.3 Magnetic induction from orientational energy equivalent of thermal energy

```
1 // Scilab Code Ex16.3 Calculating magnetic induction
     from orientational energy equivalent of thermal
    energy: Page-514 (2010)
```

```
k = 1.38e-023;     // Boltzmann constant, joule per
   mole per kelvin
mu_B = 9.27e-024;     // Bohr's magneton, joule per
   tesla
mu_m = 5*mu_B;         // Magnetic moment of
   paramagnetic sample, joule per tesla
T = 300;     // Thermal energy of specimen, joule
// At equilibrium, mu_m*B = k*T, solving for B
B = k*T/mu_m;     // Magentic induction of
   paramagnetic sample, weber per metre square
printf("\nThe magentic induction of paramagnetic
   sample = %5.2f weber per metre square", B);
// Result
// The magentic induction of paramagnetic sample =
   89.32 weber per metre square
```

Scilab code Exa 16.4 Behaviour of paramagnetic salt when placed in uniform magnetic field

```
// Scilab Code Ex16.4 Behaviour of paramagnetic salt
    when placed in uniform magnetic field: Page-514
   (2010)
k = 1.38e-023;     // Boltzmann constant, joule per
   mole per kelvin
T = 300;     // Thermal energy of specimen, joule
mu_B = 9.27e-024;     // Bohr's magneton, ampere per
   metre square
mu_0 = 4*%pi*1e-07;     // Magnetic permeability of
   free space, henry per metre
N = 1e+28;     // Concentration of paramagnetic ions
   in paramagnetic salt, per metre cube
mu_m = mu_B;
H = 1e+06;     // Applied magnetic field, A/m
chi = mu_0*N*mu_m^2/(3*k*T);     // Paramagnetic
   susceptibility of salt at room temperature
```

```
10 M = chi*H;     // Intensity of magnetization at room
   temperature, A/m
11 printf("\nThe paramagnetic susceptibility of salt at
    room temperature = %3.1e", chi);
12 printf("\nThe intensity of magnetization of salt =
   %d A/m", round(M));
13 // Result
14 // The paramagnetic susceptibility of salt at room
   temperature = 8.7e-005
15 // The intensity of magnetization of salt = 87 A/m
```

Chapter 17

Superconductivity

Scilab code Exa 17.1 Variation of critical magnetic field with temperature

```
1 // Scilab Code Ex17.1 Variation of critical magnetic
     field with temperature Page-537 (2010)
2 T_c = 3.7;     // Critical temperature of
    superconducting transition, kelvin
3 H_c0 = 0.0306;     // Critical magnetic field to
    destroy superconductivity, tesla
4 T = 2;     // Temperature at which critical magnetic
    field is to be found out, kelvin
5 H_cT = H_c0*(1-(T/T_c)^2);
6 printf("\nThe critical magnetic field at %d K = %f T
    ", T, H_cT);
7 // Result
8 // The critical magnetic field at 2 K = 0.021659 T
```

Scilab code Exa 17.2 Temperature variation of critical magnetic field for tin

```
1 // Scilab Code Ex17.2 Variation of critical magnetic
     field with temperature for tin Page-537 (2010)
```

```
2 T_c = 3.69;     // Critical temperature of
    superconducting transition, kelvin
3 B_c0 = 3e+5/(4*%pi);     // Critical magnetic field
    intensity to destroy superconductivity at zero
    kelvin, tesla
4 B_cT = 2e+5/(4*%pi);     // Critical magnetic field
    at temperature T kelvin
5 // T = 2;     // Temperature at which critical
    magnetic field is to be found out, kelvin
6 // since B_cT = B_c0*(1-(T/T_c)^2); // Critical
    magnetic field intensity as a function of
    temperature
7 // Solving for T
8 T = sqrt(1-B_cT/B_c0)*T_c;     // Temperature at
    which critical magnetic field becomes B_cT,
    kelvin
9 printf("\nThe temperature at which critical magnetic
     field becomes %4.2e T = %4.2f K",B_cT,T); //
    Display result
10 // Result
11 // The temperature at which critical magnetic field
    becomes 1.59e+04 T = 2.13 K
```

Scilab code Exa 17.3 Critical current for a lead wire from its critical temperature

```
1 // Scilab Code Ex17.3 Calculating critical current
    for a lead wire from critical temperature of lead
     Page-537 (2010)
2 T_c = 7.18;     // Critical temperature of
    superconducting transition for Pb, kelvin
3 H_c0 = 6.5e+4;     // Critical magnetic field
    intensity to destroy superconductivity at zero
    kelvin, A/m
4 T = 4.2;     // Temperature at which critical
```

```
    magnetic field becomes H_cT, kelvin
d = 1e-03;     // Diameter of lead wire, m
H_cT = H_c0*(1-(T/T_c)^2);;     // Critical magnetic
    field intensity at temperature T kelvin, A/m
I_c = %pi*d*H_cT;     // Critical current through the
    lead wire, A
printf("\nThe critical current through the lead wire
    = %6.2f A", I_c);

// Result
// The critical current through the lead wire =
    134.33 A
```

Scilab code Exa 17.4 Dependence of London penetration depth on temperature

```
// Scilab Code Ex17.4 Dependence of London
    penetration depth on temperature Page-548 (2010)
N = 6.02e+023;     // Avogadro's number
rho = 13.55e+03;     // Density of mercury, kg per
    metre cube
M = 200.6e-03;     // Molecular mass of mercury, kg
lambda_T = 750e-010;     // Penetration depth of
    mercury at T kelvin, m
T_c = 4.12;     // Critical temperature of
    superconducting transition for Hg, kelvin
T = 3.5;     // Temperature at which penetration
    depth for Hg becomes lambda_T, kelvin
lambda_0 = lambda_T*(1-(T/T_c)^4)^(1/2);     //
    Penetration depth of mercury at 0 kelvin, m
n_0 = N*rho/M;     // Normal electron density in
    mercury, per metre cube
n_s = n_0*(1-(T/T_c)^4);     // Superelectron density
    in mercury, per metre cube
printf("\nThe penetration depth at 0 K = %4.2e m",
```

```
    lambda_0);
12 printf("\nThe superconducting electron density = %4
    .2e per metre cube", n_s);
13
14 // Result
15 // The penetration depth at 0 K = 5.19e-008 m
16 // The superconducting electron density = 1.95e+028
    per metre cube
```