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

May 24, 2016

---

[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

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

// Result
// The number of edges of the quartz crystal is :
//     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
```

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

// Result
// The number of atoms per unit cell for an fcc lattice of copper crystal
// 3.9588702
```

Scilab code Exa 1.4 Lattice constant of a unit cell

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

```

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

Scilab code Exa 1.5 Density of diamond

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

// Result
// Density of diamond cubic unit cell, in kg per metre cube, is :
// 3503
```

Scilab code Exa 1.6 Calculating Unit cell dimensions

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

// Result
// The Lattice parameter of aluminium, in angstrom, is :
//      4.0486332
// The diameter of aluminium atom, in angstrom, is :
//      2.862816
```

Scilab code Exa 1.17 Angle between two crystal directions

```
// Scilab Code Ex 1.17 Angle between two crystal directions: Page-23 (2010)
h1 = 1;k1 = 1;l1 = 1; // Miller indices of first set of planes
h2 = 0;k2 = 0;l2 = 1; // Miller indices of second set of planes
// We know that
// cos(theta) = (h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1^2)*sqrt(h1^2+k1^2+l1^2))
// Solving for theta
theta = acos((h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1^2)*sqrt(h2^2+k2^2+l2^2)));
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);

// Result
// The angle between [111] and [001] directions in the cubic crystal, in degrees, is :
//     54.74
```

Scilab code Exa 1.18 Angle between two directions of cubic crystal

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

```
theta = acos((h1*h2+k1*k2+l1*l2)/(sqrt(h1^2+k1^2+l1^2)*sqrt(h2^2+k2^2+l2^2)));
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);

// Result
// The angle between [111] and [-1-1 1] directions in the cubic crystal, in degrees, is :
//     109.5
```

Scilab code Exa 1.19 Miller indices of the crystal plane

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

// Result
// The required miller indices are : (3 2 1)
```

Scilab code Exa 1.20 Indices of lattice plane

```
// Scilab Code Ex 1.20 Indices of lattice plane: Page-25 (2010)
m = 10000; // Coefficient of intercept along x-axis, can be taken as some large value
n = 2;    // Coefficient of intercept along y-axis
p = 1/2;    // Coefficient of intercept along z-axis
m_inv = 1/m;    // Reciprocate m
n_inv = 1/n;    // Reciprocate n
p_inv = 1/p;    // Reciprocate p
mul_fact = n;    // multiplicative factor
m1 = m_inv*mul_fact;    // Clear the first fraction
m2 = n_inv*mul_fact;    // Clear the second fraction
m3 = p_inv*mul_fact;    // Clear the third fraction
printf("\nThe required miller indices are : %d, %d, %d ", m1,m2,m3);

// Result
// The required miller indices are :
//      0, 1, 4
```

Scilab code Exa 1.21 Length of the intercepts

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

```
p = 1/2;     // Reciprocal of miller index on x-axis
q = 1/3;     // Reciprocal of miller index on y-axis
r = 1/(-1);     // Reciprocal of miller index on z-axis
l1 = 1.21D-10;     // Actual length of the intercept along x-axis, m
mul_fact = l1/(p*a); // Calculate multiplication factor
l2 = mul_fact*q*b;     // Actual length of the interceptalong y-axis, m
l3 = mul_fact*r*c;     // Actual length of the intercept along z-axis, m
disp(l2/1D-10, "The length of the intercept along y-axis, in angstrom, is : ");
disp(l3/1D-10, "The length of the intercept along z-axis, in angstrom, is : ");

// Result
// The length of the intercept along y-axis, in angstrom, is :
//     1.2266667
// The length of the intercept along z-axis, in angstrom, is :
//   - 3.94
```

Scilab code Exa 1.22 Miller indices of lattice planes

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

```
l2 = 4;    // Length of the intercept along y-axis, angstrom
l3 = 3;    // Length of the intercept along z-axis, angstrom
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
mul_fact = double(lcm(int32([l,m,n])));
m1 = mul_fact*r1;    // miller index along x-axis
m2 = mul_fact*r2;    // miller index along y-axis
m3 = mul_fact*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 3 6
```

Scilab code Exa 1.24 Miller Bravias indices for Miller indices

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

```
h3 = 3; k3 = 4; l3 = 5;      // Third set of miller indices
h4 = 3; k4 = -4; l4 = 5;      // Fourth set of miller indices
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);
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);
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);
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);

// Result
// The Miller-Bravias indices corresponding to the miller indices (1 1 0), = (1 1 -2 0)
// The Miller-Bravias indices corresponding to the miller indices (1 -1 0), = (1 -1 0 0)
// The Miller-Bravias indices corresponding to the miller indices (3 4 5), = (3 4 -7 5)
// 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

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

```

function [k] = fk(H,K)    // Function for calculating (2K-H)/3
    k = (2*K - H)/3;
endfunction

function [i] = f(h,k)    // Function for calculating i
    i = -(h + k);
endfunction

function [l] = fl(L)    // Function for calculating l
    l = L;
endfunction

H1 = 1; K1 = 0; L1 = 0 ;  // First set of Miller indices
H2 = 0; K2 = 1; L2 = 0;    // Second set of miller indices
H3 = 1; K3 = 1; L3 = 0;    // Third set of miller indices

h1 = fh(H1,K1)*3;    // Call function fh
k1 = fk(H1,K1)*3;    // Call function fk
l1 = fl(L1)*3;       // Call function fl
i1 = f(h1,k1);       // Call function

h2 = fh(H2,K2)*3;    // Call function fh
k2 = fk(H2,K2)*3;    // Call function fk
l2 = fl(L2)*3;        // Call function l2
i2 = f(h2,k2);        // Call function f

h3 = fh(H3,K3)*3;    // Call function fh
k3 = fk(H3,K3)*3;    // Call function fk
l3 = fl(L3)*3;       // Call function l3
i3 = f(h3,k3);        // Call function f

```

```
printf("\n The Miller Bravias indices of [%d%d%d] are [%d %d %d %d]", H1, K1, L1, h1, k1,i1,l1);
printf("\n The Miller Bravias indices of [%d%d%d] are [%d %d %d %d]", H2, K2, L2, h2, k2,i2,l2);
printf("\n The Miller Bravias indices of [%d%d%d] are [%d %d %d %d]", H3, K3, L3, h3, k3,i3,l3);

// Result
// The Miller Bravias indices of [100] are [2 -1 -1 0]
// The Miller Bravias indices of [010] are [-1 2 -1 0]
// The Miller Bravias indices of [110] are [1 1 -2 0]
```

Scilab code Exa 1.26 Lattice parameter of a cubic crystal

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

// Result
// The lattice parameter of the cubic crystal, in angstrom, is :
// 3.4641016
```

Scilab code Exa 1.27 Interplanar spacing in tetragonal crystal

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

// Result
// The interplanar spacing between consecutive (101)
    planes : in angstrom, is :
// 1.4127338
```

Scilab code Exa 1.28 Interplanar spacing in cubic crystal

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

```

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

Chapter 2

Atomic Bonding

Scilab code Exa 2.1 Molecular stability based on bond dissociation energy

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

```
else
    disp("The molecule A+B- is unstable..");
end

//Result
//    The bond dissociation energy of A+B- molecule, in kJ/mol, is : -294
//   The molecule A+B- is stable..
```

Scilab code Exa 2.2 Conversion of eV into kcal per mol

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

//Result
//    1 eV is approximately equal to 23.033 kcal/mol
```

Scilab code Exa 2.3 Potential energy of the ionic solids

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

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

//Result
// The potential energy of NaCl molecule at equilibrium separation5 is : -7.2 eV
```

Scilab code Exa 2.4 Compressibility and energy of ionic crystal

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

```
U_bar = U/2; // Potential energy per ion, electron-volt
delta_E = IP_Na - EA_Cl; // Energy required to produce the ion-pair, eV
E_ion = delta_E/2; // Energy required to produce per ion, eV
C_E = U_bar + E_ion; // Cohesive energy per ion, eV
printf(”\nThe compressibility of NaCl crystal is %4.2e metre square newton”, K0);
printf(”\nThe cohesive energy of NaCl crystal is %4.2f eV”, C_E);

// Result
// The compressibility of NaCl crystal is 3.48e-011 metre square newton
// The cohesive energy of NaCl crystal is -3.21 eV
```

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

```
// Scilab Code Ex2.5 Potential energy and dissociation energy of a diatomic molecule: Page-69 (2010)
e = 1.6D-19; // Electronic charge, C
A = 1.44D-39; // Constant corrsponding to the attractive term in potential energy, joule metre square
B = 2.19D-115; // Constant corresponding to the repulsive term in potential energy, joule metre raised to power 10
Re = (5*B/A)^(1/8); // Equilibrium spacing of diatomic molecule, m
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
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
Rc = (55/3*B/A)^(1/8); // Critical separation between the nuclei, m
F_min = -2*A/Rc^3*(1-(Re/Rc)^8); // The minimum force required to dissociate the moleule, N
disp(A,"The constant A corresponding to the attractive potential energy, in joule metre square, is :");
disp(B,"The constant B corresponding to the repulsive potential energy, in joule metre raised to power 10, is :");
disp(Rc/1d-10, "The critical separation between the nuclei, in angstrom, is : ");
disp(F_min, "The minimum force required to dissociate the molecule, in N, is : ");

//Result
// The constant A corresponding to the attractive potential energy, in joule metre square, is :                  //    7.200D-38
// The constant B corresponding to the repulsive potential energy, in joule metre raised to power 10, is :      //    9.44D-115
// The critical separation between the nuclei, in angstrom, is :
// 3.529D-10
// The minimum force required to dissociate the molecule, in N, is :
// -2.383D-09
```

Scilab code Exa 2.7 Bond formation energy of ionic solid

```
// Scilab Code Ex2.7  Bond formation Energy for K+ and Cl- ion pair: Page-70 (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
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

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

Scilab code Exa 3.2 Packing efficiency in diamond structure

```
// Scilab Code Ex3.2 Packing efficiency in diamond structure: Page-92 (2010)
// For simplicity we may take radius of the atom, R = 1 unit
R = 1;     // Radius of the atom in bcc lattice, unit
nc = 8;     // Number of corner atoms in diamond structure
nfcc = 6;   // Number of face centred atoms in diamond structure
na = 4;     // Number of atoms completely within the unit cell
n = 1/8*nc+1/2*nfcc+1*na; // Effective number of atoms in the diamond structure
V_atom = 8*4/3*%pi*R^3;  // Volume of atoms within the unit cell, unit cube
// 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
a = 8*R/sqrt(3); // lattice parameter of diamond structure, unit
V_cell = a^3;    // Volume of the unit cell, unit cube
eta = V_atom/V_cell*100; // Packing efficiency in diamond structure
printf(”\nThe packing efficiency in diamond structure is : %2.0f percent”, eta);
//Result
// The packing efficiency in diamond structure is : 34 percent
```

Scilab code Exa 3.3 Radius of largest sphere at octahedral void

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

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

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

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

```
// Scilab Code Ex4.1 Bohr's orbit for the hydrogen atom: Page-126 (2010)
n = 1;     // The ground state orbit of hydrogen atom
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
r_B = (n^2*h^2*eps_0)/(%pi*m*Z*e^2);  // Radius of first Bohr's orbit (Bohr radius), m
disp(r_B/1D-10, "The radius of first Bohr orbit, in angstrom, is : ");
// Result
// The radius of first Bohr orbit, in angstrom, is :
// 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 :
```

```
// 1.7561845
```

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

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

Scilab code Exa 4.5 Ionic Radius occupying an octahedral position

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

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

Scilab code Exa 4.6 Percentage ionic character of a covalent molecule

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

Scilab code Exa 4.7 Metallic radius from unit cell dimension

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

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

Scilab code Exa 4.8 Metallic radii from unit cell dimension

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

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

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

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

Chapter 5

Crystal Imperfections

Scilab code Exa 5.1 Variation of atomic fraction with temperature

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

```
: 9.084e-006
```

Scilab code Exa 5.2 Vacancy formation in copper

```
// Scilab Code Ex5.2 Vacancy formation in copper Page-159 (2010)
E = 1;    // Energy of formation of vacancy in copper, electron-volt
T = 1356;    // Melting point of copper, K
k = 8.614D-5;    // Boltzmann constant, electron-volt
N = 6.023D23;    // Avogadro's number
// Now fraction of vacancies = f_vacancy = n/N = exp(-E/(k*T))
f = exp(-E/(k*T));    // Fraction of vacancies in the solid at 300 K
n = N*f;  // Number of vacancy per mole
delta_d = n + N;    // Change in the density due to creation of vacancy
f_d = delta_d/N;    // Relative change in the density of copper due to vacancy formation
printf("\nThe relative change in the density of copper due to vacancy formation (n+N)/N, is : %9.7f : 1", f_d);
//Result
// 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

```
// Scilab Code Ex5.3 Concentration of Schottky imperfections Page-159 (2010)
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
```

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

```
// Scilab Code Ex5.5 Average energy required to create one Schottky defect in NaCl Page-160 (2010)
N = 6.023D23;    // Avogadro's number
k = 8.614D-5;    // Boltzmann's constant, eV/K
T = 27+273;    // Absolute room temperature, K
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

```
// Scilab Code Ex5.8 Minimum dislocation density in aluminium Page-164 (2010)
b = 3D-10;     // Magnitude of Burgers vector, m
r = 0.05;     // Radius of curvatur of the aluminium crystal, 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 Sq.m
d = 1/(r*b);     // Minimum dislocation density in aluminium, number per Sq. m
printf("\nThe minimum dislocation density in aluminium, %4.1e, dislocations per Sq. m", d);
//Result
// The minimum dislocation density in aluminium, 6.7e+010, dislocations per Sq. m
```

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

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

```
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
// As F/F_100 = cos_theta, solving for F
F_110 = F_100/cos_theta;    // Applied force along [1 -1 0] direction, N
printf("\nThe applied force along [1-10] direction = %3d N", F_110);
// Result
// The applied force along [1-10] direction = 183 N
```

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

```
// Scilab Code Ex5.10 Determining resolved componet of shearing force in a given direction: Page-168 (2010)
h1 = 1; k1 = 1; l1 = 1 // Miller indices for first set of planes
h2 = 1; k2 = 1; l2 = 0;    // Miller indices for second set of planes
F_111 = 660;    // Shearing force along [111] direction, N
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
// As F_110/F_111 = cos_theta, solving for F_110
F_110 = F_111*cos_theta;    // Resolved component of shearing force along [110] direction, N
printf("\nThe resolved component of shearing force along [110] direction, F_110 = %3d N", F_110);
// Result
// 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

```
// Scilab Code Ex5.11 Dependence of applied stress on the slip direction of a copper: Page-169 (2010)
tau_critical = 1;    // Critical shear stress for the <-110>{111} slip system, mega-pascal (MPa)
// For directions [001] and [-111]
h1 = 0; k1 = 0; l1 = 1      // Miller indices for first set of planes
h2 = -1; k2 = 1; l2 = 1;     // Miller indices for second set of planes
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
// For directions [001] and [101]
h1 = 0; k1 = 0; l1 = 1      // Miller indices for first set of planes
h2 = 1; k2 = 0; l2 = 1;     // Miller indices for second set of planes
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
sigma = tau_critical/(cos_phi*cos_lambda);    // Stress along [001] direction, newton per metre square
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);
// For directions [001] and [110]
h1 = 0; k1 = 0; l1 = 1      // Miller indices for first set of planes
h2 = 1; k2 = 1; l2 = 0;     // Miller indices for
```

```
second set of planes
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
if cos_lambda <> 0 then
    sigma = tau_critical/(cos_phi*cos_lambda);     // Stress along [001] direction, newton per metre square
    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);
else
    printf("\nSince cos_lambda = 0, this implies that slip cannot occur in [110] direction when the stress is applied along [001] direction");
end
// Result
// The stress required to be applied along [001] direction to produce slip in the [101] direction on the (-111) plane = 2.45 MPa
// 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

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

```
h1 = 0; k1 = 1; l1 = 0      // Miller indices for
    first set of planes
h2 = 1; k2 = 1; l2 = 0;     // Miller indices for
    second set of planes
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
// For directions [110s] and [101]
h1 = 1; k1 = 0; l1 = 1      // Miller indices for
    first set of planes
h2 = 1; k2 = 1; l2 = 0;     // Miller indices for
    second set of planes
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
tau = sigma*cos_phi*cos_lambda;     // Resolved shear
     stress in the [101] direction on the (010) plane
    , MPa
printf(”\nThe resolved shear stress in the [101]
    direction on the (010) plane = %4.1f MPa”, tau);
// Result
// 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

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

```
h2 = 1; k2 = -1; l2 = 1;    // Miller indices for second set of planes
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
// For directions [1 - 1 0] and [1 -1 1]
h1 = 1; k1 = -1; l1 = 0     // Miller indices for first set of planes
h2 = 1; k2 = -1; l2 = 1;    // Miller indices for second set of planes
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
tau_c = sigma_critical*cos_phi*cos_lambda;    // The critical resolved shear stress in the [1 -1 0] direction on the (111) plane, MPa
printf("\nThe critical resolved shear stress in the [1 -1 0] direction on the (111) plane = %4.2f MPa", tau_c);
// Result
// 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

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

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

```
// Scilab Code Ex5.15 Determining applied tensile stress in a direction to initiate plastic deformation: Page-170 (2010)
tau_critical = 0.7;    // Critical resolved shear stress for fcc crystal, MPa
// For directions [100] and [1 1 1]
h1 = 1; k1 = 0; l1 = 0;     // Miller indices for first set of planes
h2 = 1; k2 = 1; l2 = 1;    // Miller indices for second set of planes
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
// For directions [1 0 0] and [1 -1 0]
h1 = 1; k1 = 0; l1 = 0     // Miller indices for first set of planes
h2 = 1; k2 = -1; l2 = 0;    // Miller indices for second set of planes
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
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
printf("\nThe critical resolved shear stress in the [1 -1 0] direction on the (1 1 1) plane = %3.1f MPa", sigma_c);
// Result
// 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

```
// 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

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

Scilab code Exa 5.22 Tilt angle from dislocation spacing

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

```
printf("\nThe tilt angle between two tilt boundaries of Cu = %4.2e radian", theta);
// Result
// 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

```
// Scilab Code Ex6.1 Rate of diffusion of nitrogen
   through steel wall: Page-195 (2010)
D = 1e-019;     // Diffusion coeffcient of nitrogen
   in steel at room temperature, metre square per
   sec
dc = 10;     // Concentration of nitrogen at the
   inner surface of the tank, kg per metre cube
dx = 10e-03;     // Thickness of the steel wall, m
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
printf("\nThe rate at which nitrogen escapes through
    the tank wall = %1.0e kg per metre square per
   sec", J);
// Result
// 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

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

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

```
D = 3e-011;    // Diffusion coeffcient of carbon in iron at 1000 degree celsius, metre square per sec
n1 = 20;    // Number of unit cells per carbon atom at the surface of steel
n2 = 30;    // Number of unit cells per carbon atom at a depth 1 mm from the surface of steel
c1 = 1/(n1*a^3);    // Atomic percent of carbon at the surface, per metre cube
c2 = 1/(n2*a^3);    // Atomic percent of carbon at a depth 1 mm from the surface, per metre cube
dx = 1e-03;    // Thickness of the steel bar, m
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
J_uc = J*a^2*60;    // The number of carbon atoms diffusing through each unit cell per minute
printf("\nThe number of carbon atoms diffusing through each unit cell per minute = %2d atoms per minute", J_uc);
// Result
// 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

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

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

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

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

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

```
// Scilab Code Ex6.8 Activation energy for diffusion of silver in silicon: Page 206 (2010)
R = 8.314;    // Molar gas constant, J/mol/K
T1 = 1350+273;    // First temperature at which difuusion of Ag into Si takes place, kelvin
T2 = 1100+273;    // Second temperature at which difuusion of Ag into Si takes place, kelvin
DRR = 8;    // Ratio of diffusion rates of Ag in Si at T1 and T2
// As diffusion coefficient at temperature T1 is D1 = D0*exp(-Q/(R*T1))
// and that at temperature T2 is D1 = D0*exp(-Q/(R*T2)), so that the diffusion rates ratio
// D1/D2 = DRR = exp(Q/R*(1/T2-1/T1)), solving for Q, we have
Q = R*log(DRR)/((1/T2-1/T1)*1000);    // Activation energy for diffusion of Ag in Si, kJ/mol
printf("\nThe activation energy for diffusion of Ag in Si = %3d kJ/mol", Q);
// Result
// 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);
printf("\nThe diffusivity of the system = %4.2e cm square per sec", D);
// Result
// The activation energy for diffusion of Ga in Si = 103 kcal/mol
// The pre-exponential diffusion constant, D0 = 24893 cm square per sec
// 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

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

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

Scilab code Exa 6.11 Time required for carburizing of steel

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

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

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

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

```
// Scilab Code Ex6.13 Depth of decarburization below the surface of steel: Page 211 (2010)
C2 = 0.012;    // Initial carbon concentration of steel
Cx = 0.008;    // Carbon concentration of carburized steel at x metre depth
Cs = 0;    // Carbon concentration of steel at the surface
t = 5*60*60;    // Carburizing time of steel, sec
D_927 = 1.28e-011;    // Diffusion coefficient for carbon in iron, metre square per sec
erf_Z = abs((Cs-Cx)/(C2-Cs));    // Error function of Z as a solution to Fick's second law
Z1 = 0.65, Z2 = 0.70;    // Preceding and succeeding values about Z from error function table
erf_Z1 = 0.6420, erf_Z2 = 0.6778;    // Preceding and succeeding values about erf_Z from error function table
Z = poly(0,'Z');
Z = roots((Z-Z1)/(Z2-Z1)-(erf_Z-erf_Z1)/(erf_Z2-erf_Z1));
// 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_927*t); // Depth of decarburization below the surface of steel, m
printf("\nThe minimum depth upto which post machining is to be done = %4.2f mm", x*1000);
// Result
// 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

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

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

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

```
/(2*a)
// As v0 = a*(2*K/m)^(1/2)     --- (iv)
// From (iii) and (iv), we have
omega_min = v_0/a;     // Angular frequency of optical waves at k = %pi/(2*a), rad per sec
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);
// Result
// The frequency of sound waves in a monoatomic lattice = 6.28e+012 rad/sec

// The frequency of acoustic waves wave in a diatomic lattice :
// 0 rad/sec for k = 0
// 4e+012 rad/sec for k = pi/(2*a)

// The frequency of optical swaves wave in a diatomic lattice :
// 5.66e+012 rad/sec for k = 0
// 4e+012 rad/sec for k = pi/(2*a)
```

Scilab code Exa 7.3 Reflection of electromagentic radiation from a crystal

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

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

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

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

Scilab code Exa 8.3 Wavelength of characteristic X rays

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

Scilab code Exa 8.4 Atomic number of an unknown element

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

Scilab code Exa 8.5 Wavelength of copper using Moseley law

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

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

Scilab code Exa 8.6 Atomic number from wavelength using Moseley law

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

```
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
printf("\nThe atomic number unknown element = %2d", Z);
// Result
// The atomic number unknown element = 42
```

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

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

```
// 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

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

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

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

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

```
// Scilab code Ex8.10: Determining angle of reflection by using wavelength of X-ray Page 261 (2010)
lambda = 0.440e-010;     // Wavelength of X-rays, m
d = 2.814e-010;     // Interplanar spacing of rocksalt crystal, m
// 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
```

```
// Similarly, b_prime*b = 2*%pi and b_prime*a = 0
// Solving for a_prime and b_prime, we have
a_prime = 2*%pi/(a*cosd(90-theta)); // Lattice vector for reciprocal lattice, per metre
b_prime = 2*%pi/(b*cosd(90-theta)); // Lattice vector for reciprocal lattice, per metre
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);
// Result
// The reciprocal lattice vectors are:
// 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

```
// Scilab code Ex8.13: Bragg angle and the indices of diffraction of Powder Lines Page 285 (2010)
n = 1;    // Cosider first order diffraction
a = 6e-010;    // First lattice parameter of direct lattice, m
lambda = 1.54e-010;    // Wavelength used in diffraction of X–rays by Powder Method, m
// 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
// Calculate the hkl and hence interpalnar spacing 'd' for three lowest powder lines
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 = %2.0f 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

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

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

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

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

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

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

Scilab code Exa 9.4 Debye temperature of copper at low temperature

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

Scilab code Exa 9.5 Debye temperature for gold

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

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

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

```
// Scilab Code Ex9.6 Heat transference into rock salt at low temperature: Page-313(2010)
A = 464;    // Atomic specific heat of rock salt, cal g/mol/kelvin
theta_D = 281;    // Debye temperature of rock salt, K
delta_T = 10;    // Rise in temperature in each class interval, K
// Define a function which returns lattice specific heat at constant volume
function[C_vl] = lattice_SH(T)
    C_vl = A*(T/theta_D)^3;
endfunction
Q = 0;    // Initialize heat accumulator to zero, cal
for t = 10:10:40
    mean_temp = (t + (t + 10))/2;    // Calculate mean temperature of each class interval, K
    Q = Q + 2*delta_T*lattice_SH(mean_temp);    // Acuumulate heat for each step
end
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);
```

```
// Result
// 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

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

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

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

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

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

Scilab code Exa 10.4 Degeneracy of energy level

```
// Scilab Code Ex 10.4 Degeneracy of Energy Level: Page-332 (2010)
// Function to find the factorial of a number
function[f] = fact(num)
    f = 1;
    for i = 1:1:num
        f = f*i;
    end
endfunction

// Fucntion to determine degenerate energy states
function[degstates] = degno(a, b, c)// degno takes three arguments
    if  a == b & b == c then // check if all the values are same
        degeneracy = 3;
        degstates = fact(3)/fact(degeneracy); // calculate degenerate states
    end
    if a == b | b == c | c == a then    // check if any two values are equal
        degeneracy = 2;
        degstates = fact(3)/fact(degeneracy); // calculate degenerate states
    end
    if a ~= b & b ~= c then    // check if all the values are different
```

```
        degeneracy = 1;
        degstates = fact(3)/fact(degeneracy); // calculate degenerate states
    end
endfunction
//
clc
coef = 38;    // Coefficient of H^2/(8*m*a^2)
nx = zeros(1,5);    // Quantum number corresponding to x-direction
ny = zeros(1, 5);   // Quantum number corresponding to y-direction
nz = zeros(1,5);    // Quantum number corresponding to z-direction
deg = zeros(1,5);    // Variable to store the degeneracy of states
count = 1;    // set the counter
sum = 0;     // initialize the sum
// Look for all the possible set of values for nx, ny and ny
for i = 1:1:10
    for j = 1:1:10
        for k = 1:1:10
      // Check for the condition and avoid repetition of set of values
            if ((i^2+j^2+k^2==coef) & (i+j+k)> sum) then
                nx(1,count)=i; // Save current i value
                ny(1,count)=j; // Save current j value
                nz(1,count)=k; // Save current k value
                deg(1,count) = degno(i, j, k); // Save degeneracy for given set of values
                count = count + 1; // Increment the counter
```

```
                sum = i + j + k; // Add the three values of quantum numbers
            end
        end
    end
end
printf(”\nThe %d set(s) of values of quantum number are : \n”, count-1);
deg_states = 0; // Intialize the variable
for i = 1:1:count-1
    printf(”\nnx = %d, ny = %d, nz = %d\n”, nx(1,i), ny(1,i), nz(1,i));
    deg_states = deg_states + deg(1,i); // Accumulate the degeneracy
end
    printf(”\nThe given energy level is %d-fold degenerate.”, deg_states);
//Result
// The 2 set(s) of values of quantum number are :
//          nx = 1, ny = 1, nz = 6
//          nx = 2, ny = 3, nz = 5
// The energy level is 9-fold degenerate
```

Scilab code Exa 10.5 Fermi energy of zinc at absolute zero

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

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

Scilab code Exa 10.7 The electroic specific heat of Cu

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

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

Scilab code Exa 10.8 Electrical resitivity of sodium metal

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

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

Scilab code Exa 10.9 Electrical conductivity of Cu

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

Scilab code Exa 10.10 Electron mobility inside conductors

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

Scilab code Exa 10.11 Lorentz number calculation of a solid

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

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

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

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

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

Scilab code Exa 10.13 Thermionic emission of a filament

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

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

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

Chapter 11

Band Theory

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

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

```
    D square lattice = %1d", E_ratio);
// Result
// 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

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

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

Scilab code Exa 13.7 Electrical conductivity of intrinsic and extrinsic Si

```
// Scilab Code Ex13.7 Electrical conductivity of
    intrinsic and extrinsic Si: Page-439 (2010)
NA = 6.023e+23;    // Avogadro's number
A_Si = 28.09e-03;    // Kilogram atomic mass of Si,
    kg
e = 1.602e-019;    // Charge on an elctron, C
n_impurity = 1/1e+08;    // Donor impurity atoms per
     Si atom
n_i = 1.5e+016;    // Intrinsic carrier density of
    Si at room temperature, per metre cube
mu_e = 0.13;    // Mobility of electrons, metre
    square per volt per second
mu_h = 0.05;    // 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
Si_density = 2.23e+03;    // Density of silicon, kg
    per metre cube
N_Si = NA * Si_density/A_Si;    // Number of Si
    atoms, per metre cube
N_D = N_Si*n_impurity;    // Density of donor
    impurity, per metre cube;
sigma_ext = ceil(N_D)*e*mu_e;    // Extrinsic
    electrical conductivity of Si, per ohm per metre
```

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

Scilab code Exa 13.8 Resistance of intrinsic Ge Rod

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

Scilab code Exa 13.9 Hall effect in Si semiconductor

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

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

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

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

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

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

```
// 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

```
// Scilab Code Ex14.2 Calculating dielectric constant from electric polarizability of the atom : Page-464 (2010)
alpha_Kr = 2.18e-040;     // Electric polarizability of the Kr-atom, farad-metre square
NA = 6.023e+023;     // Avogadro's number
epsilon_0 = 8.85e-012;     // Electrical permittivity of free space, coulomb square per newton per metre square
N = NA/(22.4e-03);     // Number of Kr atoms per metre cube
epsilon_r = N*alpha_Kr/epsilon_0 + 1;     // Relative electrical permittivity of Kr specimen
printf("\nThe diectric constant of Kr specimen = %7.5f", epsilon_r);
// Result
// The diectric constant of Kr specimen = 1.00066
```

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

```
// Scilab Code Ex14.3 Calculating electric polarizability of a molecule from its susceptibility: Page-464 (2010)
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)
```

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

Scilab code Exa 14.5 Dipolar polarization of HCl molecule

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

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

Scilab code Exa 14.6 Effect of molecular deformation on polarizability

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

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

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

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

```
// Scilab Code Ex15.5 Energy required to stimulate the emission of Na d-lines: Page-497 (2010)
e = 1.6e-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_mean = 5893e-010;    // Wavelength of incident light, m
delta_E = h*c/(lambda_mean*e);    // The energy of the electron which must be transferred to the atoms of Na
printf("\nThe energy which must be transferred to stimulate the emission of Na d-lines = %5.3f eV", delta_E);
// Result
// 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

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

Scilab code Exa 16.2 Diamagnetic susceptibility of copper

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

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

```
// 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
```

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

Chapter 17

Superconductivity

Scilab code Exa 17.1 Variation of critical magnetic field with temperature

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

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

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

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

```
// Scilab Code Ex17.3 Calculating critical current for a lead wire from critical temperature of lead Page-537 (2010)
T_c = 7.18;    // Critical temperature of superconducting transition for Pb, kelvin
H_c0 = 6.5e+4;    // Critical magnetic field intensity to destroy superconductivity at zero kelvin, A/m
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);
printf("\nThe superconducting electron density = %4
   .2e per metre cube", n_s);

// Result
// The penetration depth at 0 K = 5.19e-008 m
// The superconducting electron density = 1.95e+028
   per metre cube
```