

Scilab Textbook Companion for  
Engineering Physics  
by S. I. Gupta, Sanjeev Gupta<sup>1</sup>

Created by

Akhil

B.sc

Physics

Cochin University of Science and Technology

College Teacher

None

Cross-Checked by

None

January 13, 2018

<sup>1</sup>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:** Engineering Physics

**Author:** S. I. Gupta, Sanjeev Gupta

**Publisher:** Dhanapat Rai Publications

**Edition:** 1

**Year:** 2012

**ISBN:** 9789383182206

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

**Exa** Example (Solved example)

**Eqn** Equation (Particular equation of the above book)

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

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

# Contents

List of Scilab Codes	4
1 Bonding in Solids	5
2 Crystal Structure	8
3 X ray Diffraction	24
4 Defects in Crystals	35
5 Elements of Statistical Mechanics	37
6 Principles of Quantum Mechanics	43
8 Semiconductor Physics	69
9 Physics of Semiconductor Devices	77
10 Dielectric Properties	81
11 Magnetic Properties	94
12 Lasers	101
13 Fibre Optics	107



# List of Scilab Codes

Exa 1.1	potential energy . . . . .	5
Exa 1.2	atomic cohesive energy . . . . .	5
Exa 1.3	binding energy . . . . .	6
Exa 2.1	volume density . . . . .	8
Exa 2.2	the lattice . . . . .	9
Exa 2.3	number of unit cells . . . . .	9
Exa 2.8	intercept on Z axis . . . . .	10
Exa 2.9	interplanar spacing in 3rd plane . . . . .	11
Exa 2.10	interplanar spacing in 3rd plane . . . . .	12
Exa 2.11	interplanar spacing . . . . .	13
Exa 2.14	density of crystal . . . . .	13
Exa 2.15	minimum radius . . . . .	14
Exa 2.16	maximum radius . . . . .	14
Exa 2.17	percent volume change . . . . .	15
Exa 2.18	density of diamond . . . . .	16
Exa 2.20	density of zinc . . . . .	16
Exa 2.21	density of GaAs . . . . .	17
Exa 2.22	lattice constant . . . . .	18
Exa 2.23	density of copper . . . . .	18
Exa 2.24	lattice constant . . . . .	19
Exa 2.25	radius of atom . . . . .	20
Exa 2.26	distance between ions . . . . .	20
Exa 2.27	distance between ions . . . . .	21
Exa 2.28	packing factor . . . . .	22
Exa 2.29	packing fraction . . . . .	22
Exa 2.30	lattice constant . . . . .	23
Exa 3.1	wavelength . . . . .	24
Exa 3.2	wavelength . . . . .	24

Exa 3.3	wavelengths . . . . .	25
Exa 3.4	separation between lattice planes . . . . .	26
Exa 3.5	wavelength . . . . .	26
Exa 3.6	velocity . . . . .	27
Exa 3.7	longest wavelength . . . . .	28
Exa 3.8	glancing angle . . . . .	28
Exa 3.9	interplanar spacing . . . . .	29
Exa 3.10	avagadro number . . . . .	30
Exa 3.11	spacing . . . . .	31
Exa 3.12	the crystal . . . . .	32
Exa 3.13	spacing of crystal . . . . .	32
Exa 3.14	radius of atom . . . . .	33
Exa 4.1	fraction of vacancy sites at 1000 C . . . . .	35
Exa 4.2	energy required . . . . .	36
Exa 5.1	temperature . . . . .	37
Exa 5.2	temperature . . . . .	37
Exa 5.3	wavelength . . . . .	38
Exa 5.4	wavelength . . . . .	39
Exa 5.5	temperature of moon . . . . .	39
Exa 5.6	temperature of moon . . . . .	40
Exa 5.7	maximum kinetic energy . . . . .	40
Exa 5.8	fermi energy . . . . .	41
Exa 6.1	de broglie wavelength of electron . . . . .	43
Exa 6.2	de broglie wavelength of proton . . . . .	44
Exa 6.3	de broglie wavelength of proton . . . . .	45
Exa 6.4	energy of neutron . . . . .	46
Exa 6.5	energy of electron . . . . .	46
Exa 6.6	kinetic energy of electron . . . . .	47
Exa 6.7	voltage . . . . .	48
Exa 6.8	kinetic energy of neutron . . . . .	48
Exa 6.9	wavelength of electron . . . . .	49
Exa 6.10	wavelength of photo electron . . . . .	50
Exa 6.11	wavelength of electron . . . . .	50
Exa 6.12	de broglie wavelength of neutron . . . . .	51
Exa 6.13	de broglie wavelength of neutron . . . . .	52
Exa 6.14	de broglie wavelength of proton . . . . .	52
Exa 6.15	wavelength of thermal neutron . . . . .	53
Exa 6.16	interplanar spacing . . . . .	54

Exa 6.17	interplanar spacing . . . . .	54
Exa 6.18	uncertainty in momentum . . . . .	55
Exa 6.19	uncertainty in position of electron . . . . .	56
Exa 6.21	uncertainty in velocity . . . . .	56
Exa 6.22	uncertainty in velocity . . . . .	57
Exa 6.23	smallest possible uncertainty in position of electron . . . . .	58
Exa 6.24	minimum uncertainty in velocity of electron	58
Exa 6.25	time required . . . . .	59
Exa 6.26	uncertainty in position of electron . . . . .	59
Exa 6.27	uncertainty in position of dust particle . . . . .	60
Exa 6.28	uncertainty in energy . . . . .	61
Exa 6.29	minimum uncertainty in frequency . . . . .	61
Exa 6.30	minimum energy . . . . .	62
Exa 6.31	least energy . . . . .	63
Exa 6.32	3rd least energy . . . . .	63
Exa 6.33	3rd least energy . . . . .	64
Exa 6.34	energy difference between ground state and 1st excited state . . . . .	65
Exa 6.35	energy levels . . . . .	66
Exa 6.36	quantum state . . . . .	67
Exa 6.37	probability of finding the particle . . . . .	68
Exa 8.1	ratio of density of electrons . . . . .	69
Exa 8.3	density of holes and electrons . . . . .	70
Exa 8.4	position of Fermi level . . . . .	70
Exa 8.5	position of Fermi level . . . . .	71
Exa 8.6	hall coefficient . . . . .	71
Exa 8.7	hall coefficient . . . . .	72
Exa 8.8	hall coefficient . . . . .	73
Exa 8.9	hall coefficient . . . . .	73
Exa 8.10	mobility . . . . .	74
Exa 8.11	charge carrier concentration . . . . .	74
Exa 8.12	hall angle . . . . .	75
Exa 9.1	value of current . . . . .	77
Exa 9.2	reverse saturation current . . . . .	77
Exa 9.3	voltage applied . . . . .	78
Exa 9.4	rectification ratio . . . . .	79
Exa 9.5	voltage applied . . . . .	79



Exa 10.1	relative permittivity . . . . .	81
Exa 10.2	polarisation . . . . .	82
Exa 10.3	induced dipole moment . . . . .	82
Exa 10.4	dipole moment . . . . .	83
Exa 10.5	atomic polarizability . . . . .	84
Exa 10.6	dipole moment . . . . .	84
Exa 10.7	permittivity . . . . .	85
Exa 10.8	dipole moment . . . . .	86
Exa 10.9	relative permittivity . . . . .	86
Exa 10.10	electronic polarizability . . . . .	87
Exa 10.11	dielectric constant . . . . .	88
Exa 10.12	electronic polarizability . . . . .	88
Exa 10.13	displacement . . . . .	89
Exa 10.14	dielectric constant . . . . .	90
Exa 10.15	dielectric constant . . . . .	90
Exa 10.16	electronic polarizability . . . . .	91
Exa 10.17	ratio between electronic and ionic polarizability . . . . .	92
Exa 10.18	percentage of ionic polarizability . . . . .	92
Exa 11.1	susceptibility . . . . .	94
Exa 11.2	absolute permeability . . . . .	95
Exa 11.3	relative permeability . . . . .	95
Exa 11.4	relative permeability . . . . .	96
Exa 11.5	permeability of rod . . . . .	96
Exa 11.6	flux density . . . . .	97
Exa 11.7	magnetic susceptibility . . . . .	98
Exa 11.8	magnetic moment . . . . .	99
Exa 11.9	bohrrs magneton . . . . .	99
Exa 12.1	momentum of photon . . . . .	101
Exa 12.2	energy of laser pulse . . . . .	102
Exa 12.3	coherence time . . . . .	102
Exa 12.4	intensity of beam . . . . .	103
Exa 12.5	number of ions . . . . .	103
Exa 12.6	population ratio . . . . .	104
Exa 12.7	coherence length . . . . .	105
Exa 12.8	intensity of image . . . . .	105
Exa 13.1	maximum entrance angle . . . . .	107
Exa 13.2	acceptance angle . . . . .	108

Exa 13.3	acceptance angle . . . . .	108
Exa 13.4	critical angle . . . . .	109
Exa 13.5	cladding refractive index . . . . .	110
Exa 13.6	critical angle . . . . .	110
Exa 13.7	numerical aperture . . . . .	111
Exa 13.9	critical angle . . . . .	112
Exa 13.10	velocity of light in fibre core . . . . .	112
Exa 13.11	diameter of core . . . . .	113
Exa 13.12	maximum radius for fibre . . . . .	114
Exa 13.13	diameter of fibre core . . . . .	114
Exa 13.14	number of guided modes . . . . .	115
Exa 13.15	fibre loss . . . . .	115
Exa 14.1	total absorption in hall . . . . .	117
Exa 14.2	reverberation time with audience in cushioned chairs . . . . .	117
Exa 14.3	reverberation time . . . . .	119
Exa 14.4	acoustic power . . . . .	120

# Chapter 1

## Bonding in Solids

Scilab code Exa 1.1 potential energy

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 e=1.6*10^-19; //charge(coulomb)
8 x=9*10^9;
9 r0=2.81*10^-10; //equilibrium distance(m)
10 A=1.748; //madelung constant
11 n=9; //repulsive exponent value
12
13 //Calculations
14 U0=-(x*A*e/r0)*(1-1/n); //potential energy(eV)
15
16 //Result
17 printf("\n potential energy is %0.3f eV",U0/2)
```

---

Scilab code Exa 1.2 atomic cohesive energy

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 e=1.6*10^-19; //charge (coulomb)
8 x=9*10^9;
9 r0=3.56*10^-10; //equilibrium distance (m)
10 A=1.763; //madelung constant
11 n=10.5; //repulsive exponent value
12 IE=3.89; //ionisation energy (eV)
13 EA=-3.61; //electron affinity (eV)
14
15 //Calculations
16 U0=-(x*A*e/r0)*(1-1/n); //ionic cohesive
    energy (eV)
17 U=U0+IE+EA; //atomic cohesive energy (
    eV)
18
19 //Result
20 printf("\n ionic cohesive energy is %0.2f eV",U0)
21 printf("\n atomic cohesive energy is %0.2f eV",U)

```

---

### Scilab code Exa 1.3 binding energy

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 N=6.02*10^26; //Avagadro Number
8 e=1.6*10^-19; //charge (coulomb)
9 x=9*10^9;

```

```
10 r0=0.324*10^-9;    //equilibrium distance(m)
11 A=1.748;          //madelung constant
12 n=9.5;           //repulsive exponent value
13
14 // Calculations
15 U0=(A*e*x/r0)*(1-1/n);
16 U=(U0)*N*e*10^-3;    //binding energy(kJ/kmol)
17
18
19 // Result
20 printf("\n binding energy is %0.0f *10^3 kJ/kmol",U
    /10^3)
21 printf("\n answer in the book varies due to rounding
    off errors")
```

---

# Chapter 2

## Crystal Structure

Scilab code Exa 2.1 volume density

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 M=28; //atomic weight of Si
8 N=6.023*10^23; //avagadro number
9 a=5.3*10^-10; //lattice constant(m)
10 n=4;
11
12 //Calculations
13 V=a^3; //volume(m^3)
14 m=M/(N*10^3); //mass(kg)
15 rho=n*m/V; //volume density(kg/m^3)
16
17 //Result
18 printf("\n volume density is %e kg/m^3",rho)
19 printf("\n answer in the book is wrong")
```

---

### Scilab code Exa 2.2 the lattice

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 M=55.85; //atomic weight
8 N=6.023*10^23; //avagadro number
9 a=2.9*10^-8; //lattice constant(m)
10 rho=7.87; //volume density(gm/cc)
11
12 //Calculations
13 n=rho*N*a^3/M; //number of atoms per unit cell
14
15 //Result
16 printf("\n number of atoms per unit cell is %0.3f ",
17 n)
17 printf("\n the lattice is BCC")
```

---

### Scilab code Exa 2.3 number of unit cells

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 M=120; //atomic mass
8 N=6.023*10^23; //avagadro number
9 n=2;
```

```

10 g=20;          //mass (gm)
11
12 // Calculations
13 u=n*M/N;
14 nu=g/u;        //number of unit cells
15
16 //Result
17 printf("\n number of unit cells is %0.3f *10^22",nu
        /10^22)

```

---

Scilab code Exa 2.8 intercept on Z axis

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 a=1.2;
8 b=1.8;
9 c=2.0;        //crystal primitives
10 x=2;
11 y=3;
12 z=1;         //intercepts
13 h=1.2;       //intercept on X axis
14
15 //Calculations
16 h1=a/x;
17 k1=b/y;
18 l1=c/z;
19 k=h*h1/k1;   //intercept on Y axis
20 l=h*l1/h1;   //intercept on Z-axis
21
22 //Result
23 printf("\n intercept on Y axis is %0.3f angstrom",k)

```



```
24 printf("\n intercept on Z axis is %0.3f angstrom",l)
```

---

Scilab code Exa 2.9 interplanar spacing in 3rd plane

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 r=1.246; //atomic radius(angstrom)
8 h1=2; //intercept on X axis
9 k1=0; //intercept on Y axis
10 l1=0; //intercept on Z-axis
11 h2=2; //intercept on X axis
12 k2=2; //intercept on Y axis
13 l2=0; //intercept on Z-axis
14 h3=1; //intercept on X axis
15 k3=1; //intercept on Y axis
16 l3=1; //intercept on Z-axis
17
18 //Calculations
19 a=2*sqrt(2)*r; //lattice constant
20 d1=a/sqrt(h1^2+k1^2+l1^2); //interplanar
 spacing in 1st plane(angstrom)
21 d2=a/sqrt(h2^2+k2^2+l2^2); //interplanar
 spacing in 2nd plane(angstrom)
22 d3=a/sqrt(h3^2+k3^2+l3^2); //interplanar
 spacing in 3rd plane(angstrom)
23
24 //Result
25 printf("\n interplanar spacing in 1st plane is %0.3f
 angstrom",d1)
26 printf("\n interplanar spacing in 2nd plane is %0.3f
 angstrom",d2)
```

```
27 printf("\n interplanar spacing in 3rd plane is %0.4f
    angstrom",d3)
```

---

### Scilab code Exa 2.10 interplanar spacing in 3rd plane

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h1=0; //intercept on X axis
8 k1=1; //intercept on Y axis
9 l1=1; //intercept on Z-axis
10 h2=1; //intercept on X axis
11 k2=0; //intercept on Y axis
12 l2=1; //intercept on Z-axis
13 h3=1; //intercept on X axis
14 k3=1; //intercept on Y axis
15 l3=2; //intercept on Z-axis
16
17 //Calculations
18 d1=h1^2+k1^2+l1^2; //interplanar spacing in 1
    st plane(angstrom)
19 d2=h2^2+k2^2+l2^2; //interplanar spacing in 2
    nd plane(angstrom)
20 d3=h3^2+k3^2+l3^2; //interplanar spacing in 3
    rd plane(angstrom)
21
22 //Result
23 printf("\n interplanar spacing in 1st plane is a/
    sqrt( %0.3f ) angstrom",d1)
24 printf("\n interplanar spacing in 2nd plane is a/
    sqrt( %0.3f ) angstrom",d2)
25 printf("\n interplanar spacing in 3rd plane is a/
```

```
sqrt( %0.3f ) angstrom",d3)
```

---

### Scilab code Exa 2.11 interplanar spacing

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 a=4.2; //lattice constant(angstrom)
8 h=3; //intercept on X axis
9 k=2; //intercept on Y axis
10 l=1; //intercept on Z-axis
11
12 //Calculations
13 d=a/sqrt(h^2+k^2+l^2); //interplanar spacing(
    angstrom)
14
15 //Result
16 printf("\n interplanar spacing is %0.2f angstrom",d)
```

---

### Scilab code Exa 2.14 density of crystal

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 M=63.5; //atomic weight(gm/mol)
8 N=6.023*10^23; //avagadro number
9 r=1.278*10^-8; //atomic radius(cm)
```

```

10 n=4;
11
12 // Calculations
13 m=M/N;           //mass(g)
14 a=4*r/sqrt(2);  //lattice constant(cm)
15 V=a^3;          //volume(m^3)
16 rho=n*m/V;     //density of crystal(g/cm^3)
17
18 //Result
19 printf("\n density of crystal is %0.3f gm/cm^3",rho)

```

---

#### Scilab code Exa 2.15 minimum radius

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 r=1; //assume
8
9 //Calculations
10 a=4*r/sqrt(3); //lattice constant
11 R=(a-(2*r))/2; //minimum radius
12
13 //Result"
14 printf("\n minimum radius is %0.3f r",R)

```

---

#### Scilab code Exa 2.16 maximum radius

```

1 clear
2 //
3 //

```

```

4 //
5
6 //Variable declaration
7 r=1; //assume
8
9 //Calculations
10 a=4*r/sqrt(2); //lattice constant
11 R=(a/2)-r; //maximum radius
12
13 //Result"
14 printf("\n maximum radius is %0.3f r",R)

```

---

Scilab code Exa 2.17 percent volume change

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 r1=1.258*10^-10; //atomic radius (m)
8 r2=1.292*10^-10; //atomic radius (m)
9 n1=2;
10 n2=4;
11
12 //Calculations
13 a1=4*r1/sqrt(3); //lattice constant (m)
14 V1=a1^3/n1; //volume (m^3)
15 a2=2*sqrt(2)*r2; //lattice constant (m)
16 V2=a2^3/n2; //volume (m^3)
17 V=(V1-V2)*100/V1; //percent volume change
18
19 //Result"
20 printf("\n percent volume change is %0.1f percentage
    ",V)

```

---

Scilab code Exa 2.18 density of diamond

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 a=0.356*10^-9; //cube edge(m)
8 M=12.01; //atomic weight
9 N=6.023*10^26; //avagadro number
10
11 //Calculations
12 n=8/a^3; //number of atoms
13 m=M/N; //mass(kg)
14 rho=m*n; //density of diamond(kg/m^3)
15
16 //Result"
17 printf("\n number of atoms is %0.2f *10^29",n/10^29)
18 printf("\n density of diamond is %0.1f kg/m^3",rho)
19 printf("\n answer in the book is wrong")
```

---

Scilab code Exa 2.20 density of zinc

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 a=0.27*10^-9; //lattice constant(m)
```

```

8 c=0.494*10^-9; //height of cell(m)
9 M=65.37; //atomic weight
10 N=6.023*10^26; //avagadro number
11 n=6; //number of atoms
12
13 //Calculations
14 V=3*sqrt(3)*a^2*c/2; //volume of unit cell(m^3)
15 rho=n*M/(N*V); //density of zinc(kg/m^3)
16
17 //Result"
18 printf("\n volume of unit cell is %0.3f *10^-29 m^3"
, V*10^29)
19 printf("\n density of zinc is %0.0f kg/m^3", rho)
20 printf("\n answer in the book is wrong")

```

---

### Scilab code Exa 2.21 density of GaAs

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 a1=5.43*10^-8; //lattice constant(cm)
8 M1=28.1; //atomic weight
9 N=6.023*10^23; //avagadro number
10 n1=8; //number of atoms
11 a2=5.65*10^-8; //lattice constant(cm)
12 M2=144.6; //atomic weight
13 n2=4; //number of atoms
14
15 //Calculations
16 rho1=n1*M1/(N*a1^3); //density of Si(gm/cm^3)
17 rho2=n2*M2/(N*a2^3); //density of GaAs(gm/cm^3)
18

```

```
19 //Result"
20 printf("\n density of Si is %0.2f gm/cm^3",rho1)
21 printf("\n density of GaAs is %0.3f gm/cm^3",rho2)
22 printf("\n answer in the book varies due to rounding
    off errors")
```

---

### Scilab code Exa 2.22 lattice constant

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 rho=6250;          //density (kg/m^3)
8 M=60.2;           //molecular weight
9 N=6.02*10^26;     //avagadro number
10 n=4;             //number of atoms
11
12 //Calculations
13 a=(n*M/(rho*N))^(1/3);      //lattice constant(m)
14
15 //Result
16 printf("\n lattice constant is %0.0f angstrom",a
    *10^10)
```

---

### Scilab code Exa 2.23 density of copper

```
1 clear
2 //
3 //
4 //
5
```



```

6 //Variable declaration
7 r=1.278*10^-8;           //atomic radius(cm)
8 M=63.54;                 //molecular weight(g/mol)
9 N=6.02*10^23;           //avagadro number
10 n=4;                    //number of atoms
11
12 //Calculations
13 a=4*r/sqrt(2);          //lattice constant(cm)
14 rho=n*M*10^3/(N*a^3);   //density(kg/m^3)
15
16 //Result
17 printf("\n density of copper is %0.0f kg/m^3",rho)
18 printf("\n answer in the book is wrong")

```

---

#### Scilab code Exa 2.24 lattice constant

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 rho=7870;                //density(kg/m^3)
8 M=55.8;                  //molecular weight
9 N=6.02*10^26;           //avagadro number
10 n=2;                    //number of atoms
11
12 //Calculations
13 a=(n*M/(rho*N))^(1/3);   //lattice constant(m)
14
15 //Result
16 printf("\n lattice constant is %0.3f angstrom",a
        *10^10)

```

---

### Scilab code Exa 2.25 radius of atom

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 rho=6.23;           //density (gm/cc)
8 M=60;              //molecular weight
9 N=6.023*10^23;     //avagadro number
10 n=4;              //number of atoms
11
12 //Calculations
13 a=(n*M/(rho*N))^(1/3); //lattice constant (cm)
14 r=a*sqrt(2)*10^8/4;   //radius of atom(
    angstrom)
15
16 //Result
17 printf("\n radius of atom is %0.3f angstrom",r)
```

---

### Scilab code Exa 2.26 distance between ions

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 rho=2.48;           //density (gm/cc)
8 M=58;              //molecular weight
9 N=6.023*10^23;     //avagadro number
```

```

10 n=4;      //number of atoms
11
12 // Calculations
13 a=(n*M/(rho*N))^(1/3);      //lattice constant(cm)
14 r=a*sqrt(2)*10^8/4;      //radius of atom(
    angstrom)
15 d=2*r;      //distance between ions(angstrom)
16
17 //Result
18 printf("\n distance between ions is %0.1f angstrom",
    d)

```

---

#### Scilab code Exa 2.27 distance between ions

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 rho=8.96;      //density (gm/cc)
8 M=63.5;      //molecular weight
9 N=6.02*10^23;      //avagadro number
10 n=4;      //number of atoms
11
12 // Calculations
13 a=(n*M/(rho*N))^(1/3);      //lattice constant(cm)
14 d=a/sqrt(2)*10^8;      //distance between ions(
    angstrom)
15
16 //Result
17 printf("\n distance between ions is %0.2f angstrom",
    d)

```

---

### Scilab code Exa 2.28 packing factor

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 rho=5.96;          //density (gm/cc)
8 M=50;             //molecular weight
9 N=6.023*10^23;    //avagadro number
10 n=2;             //number of atoms
11
12 //Calculations
13 a=(n*M/(rho*N))^(1/3); //lattice constant(cm)
14 r=a*sqrt(3)/4;     //radius of atom(angstrom)
15 pf=n*(4/3)*%pi*r^3/a^3; //packing factor
16
17 //Result
18 printf("\n packing factor is %0.2f ",pf)
```

---

### Scilab code Exa 2.29 packing fraction

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 a=1;             //assume
8 n=2;             //number of atoms
9
```

```

10 // Calculations
11 r=a*sqrt(3)/4;           //radius of atom
12 V=4*%pi*r^3/3;         //volume
13 f=n*V*100/a^3;        //packing fraction
14
15 //Result
16 printf("\n packing fraction is %0.0f percentage",f)

```

---

### Scilab code Exa 2.30 lattice constant

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 Vd=3*10^22;           //density (gm/cc)
8 n=8*(1/8);           //number of atoms
9
10 // Calculations
11 a=(n/Vd)^(1/3);      //lattice constant (cm)
12
13 //Result
14 printf("\n lattice constant is %0.2f angstrom",a
        *10^8)

```

---

# Chapter 3

## X ray Diffraction

Scilab code Exa 3.1 wavelength

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 d=2.82*10^-10; //lattice spacing(m)
8 theta=10; //glancing angle(degree)
9 n=1; //order
10
11 //Calculation
12 theta=theta*%pi/180; //angle(radian)
13 lamda=2*d*sin(theta)/n; //wavelength(m)
14
15 //Result
16 printf("\n wavelength is %0.5f angstrom",lamda
    *10^10)
```

---

Scilab code Exa 3.2 wavelength

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 d=3.035*10^-10; //lattice spacing(m)
8 theta=12; //glancing angle(degree)
9 n=1; //order
10
11 //Calculation
12 theta=theta*%pi/180; //angle(radian)
13 lamda=2*d*sin(theta)/n; //wavelength(m)
14
15 //Result
16 printf("\n wavelength is %0.3f angstrom",lamda
    *10^10)

```

---

### Scilab code Exa 3.3 wavelengths

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 d=2.81; //lattice spacing(angstrom)
8 theta1=15.1; //glancing angle(degree)
9 theta2=17.1; //glancing angle(degree)
10
11 //Calculation
12 theta1=theta1*%pi/180; //angle(radian)
13 lamda1=2*d*sin(theta1); //wavelength(angstrom)
14 theta2=theta2*%pi/180; //angle(radian)
15 lamda2=2*d*sin(theta2); //wavelength(angstrom)

```

```
16
17 //Result
18 printf("\n wavelengths are %0.3f angstrom and %0.4f
        angstrom",lamda1,lamda2)
19 printf("\n answer varies due to rounding off errors"
        )
```

---

#### Scilab code Exa 3.4 separation between lattice planes

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 lamda=1.54; //wavelength(angstrom)
8 theta=11; //glancing angle(degree)
9
10 //Calculation
11 theta=theta*%pi/180; //angle(radian)
12 d=lamda/(2*sin(theta)); //separation between
    lattice planes(angstrom)
13
14 //Result
15 printf("\n separation between lattice planes is %0.3
        f angstrom",d)
```

---

#### Scilab code Exa 3.5 wavelength

```
1 clear
2 //
3 //
4 //
```



```

5
6 //Variable declaration
7 lamdaB=0.92; //wavelength(angstrom)
8 theta1=30; //glancing angle(degree)
9 theta2=60; //glancing angle(degree)
10
11 //Calculation
12 theta1=theta1*pi/180; //angle(radian)
13 theta2=theta2*pi/180; //angle(radian)
14 lamdaA=2*lamdaB*sin(theta1)/sin(theta2); //
    wavelength of line A(angstrom)
15
16 //Result
17 printf("\n wavelength is %0.3f angstrom",lamdaA)
18 printf("\n answer in the book is wrong")

```

---

### Scilab code Exa 3.6 velocity

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 d=0.4086*10^-10; //lattice spacing(m)
8 theta=65; //glancing angle(degree)
9 h=6.6*10^-34; //plank's constant(Js)
10 m=9.1*10^-31; //mass(kg)
11 n=1;
12
13 //Calculation
14 theta=theta*pi/180; //angle(radian)
15 lamda=2*d*sin(theta)/n; //debroglie wavelength(m)
16 v=h/(m*lamda); //velocity(m/sec)
17

```

```
18 //Result
19 printf("\n debroglie wavelength is %0.4f *10^-10
    metre",lamda*10^10)
20 printf("\n velocity is %0.3f *10^6 m/sec",v/10^6)
21 printf("\n answer in the book is wrong")
```

---

### Scilab code Exa 3.7 longest wavelength

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 d=2.82*10^-10; //lattice spacing(m)
8 sintheta=1;
9 n=1;
10
11 //Calculation
12 lamda_max=2*d*sintheta/n; //longest wavelength(m)
13
14 //Result
15 printf("\n longest wavelength is %0.3f angstrom",
    lamda_max*10^10)
```

---

### Scilab code Exa 3.8 glancing angle

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
```

```

7 d=0.842*10^-10;    //lattice spacing(m)
8 theta1=8+(35/60); //glancing angle(degree)
9 n1=1;    //order
10 n2=3;    //order
11
12 //Calculation
13 theta1=theta1*pi/180; //angle(radian)
14 theta3=asin(n2*sin(theta1)); //glancing angle(
    radian)
15 theta3=theta3*180/pi ; //glancing angle(
    degree)
16
17 //Result
18 printf("\n glancing angle is %0.3f degree",theta3)

```

---

### Scilab code Exa 3.9 interplanar spacing

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 lamda=0.58; //wavelength(angstrom)
8 theta1=6+(45/60); //glancing angle(degree)
9 theta2=9+(15/60); //glancing angle(degree)
10 theta3=13; //glancing angle(degree)
11
12 //Calculation
13 theta1=theta1*pi/180; //angle(radian)
14 theta2=theta2*pi/180; //angle(radian)
15 theta3=theta3*pi/180; //angle(radian)
16 x1=lamda/(2*sin(theta1));
17 x2=lamda/(2*sin(theta2));
18

```

```

19 //Result
20 printf("\n interplanar spacing is %0.3f angstrom",x2
   )
21 printf("\n answer varies due to rounding off errors"
   )

```

---

### Scilab code Exa 3.10 avagadro number

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 lamda=1.3922; //wavelength(angstrom)
8 n=1;
9 theta=14+(27/60)+(26/(60*60)); //glancing angle(
   degree)
10 M=58.454; //molecular weight
11 rho=2163; //density(kg/m^3)
12
13 //Calculation
14 theta=theta*%pi/180; //angle(radian)
15 d=n*lamda/(2*sin(theta)); //lattice spacing(
   angstrom)
16 d_m=d*10^-10; //lattice spacing(m)
17 N=M/(2*rho*d_m^3); //avagadro number(mol/k-mole)
18
19 //Result
20 printf("\n lattice spacing is %0.4f angstrom",d)
21 printf("\n avagadro number is %0.4f *10^26 mol/k-
   mole",N/10^26)
22 printf("\n answer varies due to rounding off errors"
   )

```

---

### Scilab code Exa 3.11 spacing

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 lamda=0.586*10^-10; //wavelength(m)
8 theta1=5+(58/60); //glancing angle(degree)
9 theta2=12+(10/60); //glancing angle(degree)
10 theta3=18+(12/60); //glancing angle(degree)
11
12 //Calculation
13 theta1=theta1*pi/180; //angle(radian)
14 theta2=theta2*pi/180; //angle(radian)
15 theta3=theta3*pi/180; //angle(radian)
16 x1=sin(theta1);
17 x2=sin(theta2);
18 x3=sin(theta3);
19 d1=lamda/(2*sin(theta1)); //spacing for 1st
    order(m)
20 d2=2*lamda/(2*sin(theta2)); //spacing for 2nd
    order(m)
21 d3=3*lamda/(2*sin(theta3)); //spacing for 3rd
    order(m)
22 d=(d1+d2+d3)/3; //spacing(m)
23
24 //Result
25 printf("\n ratio of angles of incidence are %0.3f :
    %0.4f : %0.4f which is nothing but %0.1f : %0.1f
    : %0.1f ",x1,x2,x3,x1,x2,x3)
26 printf("\n angles of incidence should be 1st , 2nd
    and 3rd orders")
```

```
27 printf("\n spacing is %0.3f *10-10 m",d*1010)
28 printf("\n answer varies due to rounding off errors"
    )
```

---

### Scilab code Exa 3.12 the crystal

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 theta1=5+(23/60); //glancing angle(degree)
8 theta2=7+(37/60); //glancing angle(degree)
9 theta3=9+(25/60); //glancing angle(degree)
10
11 //Calculation
12 theta1=theta1*pi/180; // angle(radian)
13 theta2=theta2*pi/180; // angle(radian)
14 theta3=theta3*pi/180; // angle(radian)
15 x1=sin(theta1);
16 X1=1/(10*x1);
17 x2=sin(theta2)/x1;
18 x3=sin(theta3)/x1;
19
20 //Result
21 printf("\n ratio of angles of incidence are %0.3f :
    %0.3f : %0.3f ",x1,x2,x3)
22 printf("\n the crystal is a simple cubic crystal")
```

---

### Scilab code Exa 3.13 spacing of crystal

```
1 clear
```

```

2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.62*10^-34; //planck's constant(J sec)
8 e=1.6*10^-19; //charge(coulomb)
9 m=9*10^-31; //mass(kg)
10 E=344; //energy(volts)
11 n=1;
12 theta=60; //angle(degrees)
13
14 //Calculation
15 lamda=h/sqrt(2*m*e*E); //wavelength(m)
16 theta=theta*%pi/180; //angle(radian)
17 d=n*lamda*10^10/(2*sin(theta)); //spacing
    of crystal(angstrom)
18
19 //Result
20 printf("\n spacing of crystal is %0.2f angstrom",d)

```

---

Scilab code Exa 3.14 radius of atom

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=2;
8 k=2;
9 l=0;
10 n=1;
11 theta=32; //angle(degrees)
12 lamda=1.54*10^-10; //wavelength(m)

```

```
13
14 // Calculation
15 theta=theta*%pi/180; //angle (radian)
16 d=n*lamda*10^10/(2*sin(theta)); //spacing
    of crystal(angstrom)
17 a=d*sqrt(h^2+k^2+l^2); //lattice parameter(
    angstrom)
18 r=a/(2*sqrt(2)); //radius of atom(angstrom)
19
20 // Result
21 printf("\n lattice parameter is %0.1f angstrom",a)
22 printf("\n radius of atom is %0.2f angstrom",r)
```

---



# Chapter 4

## Defects in Crystals

Scilab code Exa 4.1 fraction of vacancy sites at 1000 C

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 T1=773; //temperature(K)
8 T2=1273; //temperature(K)
9 n=1*10^-10; //fraction of vacancy sites
10
11 //Calculations
12 X=(T1*log(n)/T2);
13
14 x=exp(X); //fraction of vacancy sites at
15 // 1000 C
16 //Result
17 printf("\n fraction of vacancy sites at 1000 C is %0
18 .4f *10^-7",x*10^7)
19 printf("\n answer varies due to rounding off errors"
20 )
```

---

Scilab code Exa 4.2 energy required

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 T=273+25; //temperature (K)
8 m=4;
9 n=5*10^11; //density (per m^3)
10 V=(2*2.82*10^-10)^3; //volume(m^3)
11 kB=8.625*10^-5;
12
13 // Calculations
14 N=m/V;
15 Ep=2*kB*T*log(N/n);
16
17 // Result
18 printf("\n energy required is %0.3f eV",Ep)
```

---

# Chapter 5

## Elements of Statistical Mechanics

Scilab code Exa 5.1 temperature

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 b=2.92*10^-3; //value of b(mK)
8 lamda=4900*10^-10; //wavelength(m)
9
10 //Calculations
11 T=b/lamda; //temperature(K)
12
13 //Result
14 printf("\n temperature is %0.0f K",T)
15 printf("\n answer in the book is wrong")
```

---

### Scilab code Exa 5.2 temperature

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 T=1500; //temperature(K)
8 lamda=5500; //wavelength(m)
9 lamda_m=20000; //wavelength(m)
10
11 //Calculations
12 T_dash=lamda_m*T/lamda; //temperature of
    sun(K)
13
14 //Result
15 printf("\n temperature is %0.0f K",T_dash)
```

---

### Scilab code Exa 5.3 wavelength

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 T=327+273; //temperature(K)
8 b=2.897*10^-3; //value of b(mK)
9
10 //Calculations
11 lamda_m=b/T; //wavelength(m)
12
13 //Result
14 printf("\n wavelength is %0.0f angstrom",lamda_m)
```

\*10<sup>10</sup>)

---

#### Scilab code Exa 5.4 wavelength

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 T=107; //temperature (K)
8 b=0.292; //value of b(cmK)
9
10 //Calculations
11 lamda_m=b/T; //wavelength (cm)
12
13 //Result
14 printf("\n wavelength is %0.3f angstrom",lamda_m
    *108)
```

---

#### Scilab code Exa 5.5 temperature of moon

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 T=1127+273; //temperature (K)
8 lamda_m=2*10-6; //wavelength (m)
9 lamda=14*10-6; //wavelength (m)
10
11 //Calculations
```

```

12 Tm=lamda_m*T/lamda;      //temperature of moon(K)
13
14 //Result
15 printf("\n temperature of moon is %0.0f K",Tm)

```

---

#### Scilab code Exa 5.6 temperature of moon

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 lamda_m=4753*10^-10;      //wavelength(m)
8 lamda=14*10^-6;          //wavelength(m)
9 b=0.2898*10^-2;          //value of constant(mK)
10
11 //Calculations
12 Ts=b/lamda_m;            //temperature of sun(K)
13 Tm=b/lamda;              //temperature of moon(K)
14
15 //Result
16 printf("\n temperature of sun is %0.0f K",Ts)
17 printf("\n temperature of moon is %0.0f K",Tm)

```

---

#### Scilab code Exa 5.7 maximum kinetic energy

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration

```

```

7 e=1.6*10^-19; //charge (coulomb)
8 m=9*10^-31; //mass(kg)
9 h=6.624*10^-34; //plank's constant(Js)
10 n=5.86*10^28; //density(electrons/m^3)
11 k=8.6*10^-5;
12
13 //Calculations
14 ef=(h^2/(8*m))*(3*n/%pi)^(2/3); //energy(J)
15 ef=ef/e; //energy(eV)
16 theta_f=ef/k; //maximum kinetic energy(K)
17
18 //Result
19 printf("\n maximum kinetic energy is %0.2f *10^4 K",
theta_f/10^4)

```

---

#### Scilab code Exa 5.8 fermi energy

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 e=1.6*10^-19; //charge (coulomb)
8 m=9*10^-31; //mass(kg)
9 h=6.62*10^-34; //plank's constant(Js)
10 rho=970; //density(kg/m^3)
11 N0=6.02*10^26; //avagadro number
12 A=23; //atomic weight
13
14 //Calculations
15 n=rho*N0/A; //concentration(electrons/m^3)
16 ef=(h^2/(8*m))*(3*n/%pi)^(2/3); //fermi energy(J
)
17 ef=ef/e; //fermi energy(eV)

```

```
18
19 //Result
20 printf("\n fermi energy is %0.3f eV",ef)
21 printf("\n answer varies due to rounding off errors"
    )
```

---



# Chapter 6

## Principles of Quantum Mechanics

Scilab code Exa 6.1 de broglie wavelength of electron

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.625*10^-34; //planck's constant(J-sec)
8 m=0.05; //mass(kg)
9 v=20; //velocity(m/sec)
10 mp=1.67*10^-27; //mass of proton(kg)
11 vp=2200; //velocity of proton(m/sec)
12 me=9.11*10^-31; //mass of electron(kg)
13 E=10*1.602*10^-19; //kinetic energy(J)
14
15 //Calculations
16 lamda_ball=h/(m*v); //de-broglie wavelength
    of ball(m)
17 lamda_p=h*10^10/(mp*vp); //de-broglie
    wavelength of proton(angstrom)
```

```

18 lamda_e=h/(2*me*E);           //de-broglie wavelength
    of electron (m)
19
20 //Result
21 printf("\n de-broglie wavelength of ball is %e m",
    lamda_ball)
22 printf("\n de-broglie wavelength of proton is %0.2f
    angstrom",lamda_p)
23 printf("\n de-broglie wavelength of electron is %0.2
    f *10^14 m",lamda_e/10^14)
24 printf("\n answer for de-broglie wavelength of
    electron in the book is wrong")

```

---

Scilab code Exa 6.2 de broglie wavelength of proton

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.63*10^-34;           //planck 's constant(J-sec)
8 m=1.673*10^-27;        //mass of proton(kg)
9 v=10^4;                 //velocity of proton(m/sec)
10 V1=100;                 //potential difference in 1st case(V)
11 V2=10000;              //potential difference in 2nd case(V
    )
12 V3=6400;               //potential difference in 3rd case(V)
13
14
15 //Calculations
16 lamda1=12.25/sqrt(V1)   //de-broglie wavelength
    in 1st case(angstrom)
17 lamda2=12.25/sqrt(V2)  //de-broglie wavelength
    in 2nd case(angstrom)

```

```

18 lamda3=12.25/sqrt(V3)           //de-broglie wavelength
    in 3rd case(angstrom)
19 lamda4=12.25/sqrt(V2)           //de-broglie wavelength
    in 4th case(angstrom)
20 lamda5=h/(m*v);                 //de-broglie wavelength of
    proton(m)
21
22 //Result
23 printf("\n de-broglie wavelength in 1st case is %0.3
    f angstrom",lamda1)
24 printf("\n de-broglie wavelength in 2nd case is %0.3
    f angstrom",lamda2)
25 printf("\n de-broglie wavelength in 3rd case is %0.3
    f angstrom",lamda3)
26 printf("\n de-broglie wavelength in 4th case is %0.3
    f angstrom",lamda4)
27 printf("\n de-broglie wavelength of proton is %0.4 f
    angstrom",lamda5*10^10)

```

---

Scilab code Exa 6.3 de broglie wavelength of proton

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.62*10^-34;                 //planck's constant(J-sec)
8 m=1.67*10^-27;                 //mass of proton(kg)
9 vc=3*10^8;                      //velocity of light(m/sec)
10
11 //Calculations
12 v=vc/20;                       //velocity of proton(m/sec)
13 lamda=h/(m*v);                 //de-broglie wavelength of
    proton(m)

```

```

14
15 //Result
16 printf("\n de-broglie wavelength of proton is %0.2f
    *10-14 m",lamda*1014)

```

---

#### Scilab code Exa 6.4 energy of neutron

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.60*10-34; //planck's constant(J-sec)
8 m=1.674*10-27; //mass of proton(kg)
9 lamda=10-10; //de-broglie wavelength(m)
10 e=1.6*10-19; //charge of electron(c)
11
12 //Calculations
13 E=h2/(2*m*lamda2); //energy of neutron(J)
14 E=E/e; //energy of neutron(eV)
15
16 //Result
17 printf("\n energy of neutron is %0.2f *10-2 eV",E
    *102)

```

---

#### Scilab code Exa 6.5 energy of electron

```

1 clear
2 //
3 //
4 //
5

```

```

6 //Variable declaration
7 h=6.62*10^-34; //planck's constant(J-sec)
8 m=9.1*10^-31; //mass of electron(kg)
9 lamda=3*10^-12; //de-broglie wavelength(m)
10 e=1.6*10^-19; //charge of electron(c)
11
12 //Calculations
13 E=h^2/(2*m*lamda^2); //energy of electron(J)
14 E=E/e; //energy of electron(eV)
15
16 //Result
17 printf("\n energy of electron is %0.1f eV",E)
18 printf("\n answer in the book is wrong")

```

---

#### Scilab code Exa 6.6 kinetic energy of electron

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.63*10^-34; //planck's constant(J-sec)
8 m=9.1*10^-31; //mass of electron(kg)
9 lamda=5896*10^-10; //de-broglie wavelength(m)
10 e=1.6*10^-19; //charge of electron(c)
11
12 //Calculations
13 K=h^2/(2*m*lamda^2); //energy of electron(J)
14 K=K/e; //kinetic energy of electron(eV)
15
16 //Result
17 printf("\n kinetic energy of electron is %0.2f
    *10^-6 eV",K*10^6)

```

---

Scilab code Exa 6.7 voltage

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.6*10^-34; //planck's constant(J-sec)
8 m=9.1*10^-31; //mass of electron(kg)
9 lamda=0.4*10^-10; //de-broglie wavelength(m)
10 e=1.6*10^-19; //charge of electron(c)
11
12 //Calculations
13 V=h^2/(2*m*e*lamda^2); //voltage(V)
14
15 //Result
16 printf("\n voltage is %0.1f V",V)
17 printf("\n answer in the book is wrong")
```

---

Scilab code Exa 6.8 kinetic energy of neutron

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.63*10^-34; //planck's constant(J-sec)
8 m=1.67*10^-27; //mass of neutron(kg)
9 lamda=10^-10; //de-broglie wavelength(m)
```

```

10 e=1.6*10^-19;          //charge of electron(c)
11
12 //Calculations
13 v=h/(m*lamda);          //velocity of neutron(m/sec)
14 E=m*v^2/(2*e);          //kinetic energy of neutron(eV)
15
16 //Result
17 printf("\n velocity of neutron is %0.2f *10^3 m/sec"
    ,v/10^3)
18 printf("\n kinetic energy of neutron is %0.5f eV",E)
19 printf("\n answer for kinetic energy in the book is
    wrong")

```

---

#### Scilab code Exa 6.9 wavelength of electron

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.6*10^-34;          //planck's constant(J-sec)
8 m=9.1*10^-31;          //mass of electron(kg)
9 c=3*10^8;              //velocity of light(m/sec)
10 e=1.6*10^-19;          //charge of electron(c)
11 E=1000;                //energy of electron(eV)
12
13 //Calculations
14 lamda_p=h*c*10^10/(E*e);          //wavelength of photon
    (angstrom)
15 lamda_e=h*10^10/sqrt(2*m*E*e);      //wavelength of
    electron(angstrom)
16
17 //Result
18 printf("\n wavelength of photon is %0.1f angstrom",

```

```
    lamda_p)
19 printf("\n wavelength of electron is %0.2f angstrom"
    ,lamda_e)
```

---

#### Scilab code Exa 6.10 wavelength of photo electron

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.6*10^-34;           //planck's constant(J-sec)
8 m=9.1*10^-31;         //mass of electron(kg)
9 c=3*10^8;             //velocity of light(m/sec)
10 lamda=0.82*10^-10;    //wavelength(m)
11
12 //Calculations
13 E=h*c/lamda;          //energy(J)
14 lamda=h*10^10/sqrt(2*m*E); //wavelength of photo
    -electron(angstrom)
15
16 //Result
17 printf("\n wavelength of photo-electron is %0.1f
    angstrom",lamda)
```

---

#### Scilab code Exa 6.11 wavelength of electron

```
1 clear
2 //
3 //
4 //
5
```



```

6 //Variable declaration
7 h=6.6*10^-34; //planck's constant(J-sec)
8 m=9.1*10^-31; //mass of electron(kg)
9 c=3*10^8; //velocity of light(m/sec)
10
11 //Calculations
12 lamda=h*10^10/(m*c); //wavelength of electron
    (angstrom)
13
14 //Result
15 printf("\n wavelength of electron is %0.4f angstrom"
    ,lamda)
16 printf("\n answer in the book varies due to rounding
    off errors")

```

---

#### Scilab code Exa 6.12 de broglie wavelength of neutron

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.625*10^-34; //planck's constant(J-sec)
8 m=1.675*10^-27; //mass of neutron(kg)
9 e=1.6*10^-19; //charge of electron(c)
10 E=10^14; //energy of neutron(eV)
11
12 //Calculations
13 v=sqrt(2*E*e/m); //velocity(m/sec)
14 lamda=h/(m*v); //de-broglie wavelength of
    neutron(m)
15
16 //Result
17 printf("\n de-broglie wavelength of neutron is %0.2f

```

```
*10-18 m", lamda*1018)
```

---

Scilab code Exa 6.13 de broglie wavelength of neutron

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.625*10-34; //planck's constant(J-sec)
8 m=1.675*10-27; //mass of neutron(kg)
9 e=1.6*10-19; //charge of electron(c)
10 E=12.8*106; //energy of neutron(eV)
11
12 //Calculations
13 v=sqrt(2*E*e/m); //velocity(m/sec)
14 lamda=h/(m*v); //de-broglie wavelength of
    neutron(m)
15
16 //Result
17 printf("\n de-broglie wavelength of neutron is %0.3 f
    *10-15 m", lamda*1015)
18 printf("\n answer in the book is wrong")
```

---

Scilab code Exa 6.14 de broglie wavelength of proton

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
```

```

7 h=6.62*10^-34;           //planck's constant(J-sec)
8 m=9.1*10^-31;           //mass of electron(kg)
9 mp=1836*m;              //mass of proton(kg)
10 c=3*10^8;              //velocity of light(m/sec)
11 e=1.6*10^-19;          //charge of electron(c)
12
13 // Calculations
14 E=m*c^2;                //energy(J)
15 v=sqrt(2*E/mp);         //velocity(m/sec)
16 lamda=h*10^10/(mp*v);   //de-broglie wavelength
                           //of proton(angstrom)
17
18 // Result
19 printf("\n de-broglie wavelength of proton is %0.4f
        angstrom",lamda)
20 printf("\n answer in the book is wrong")

```

---

Scilab code Exa 6.15 wavelength of thermal neutron

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.60*10^-34;           //planck's constant(J-sec)
8 m=1.67*10^-27;          //mass of neutron(kg)
9 k=8.6*10^-5;            //boltzmann constant(eV/deg)
10 e=1.6*10^-19;          //charge of electron(c)
11 T=300;                  //temperature(K)
12
13 // Calculations
14 lamda=h*10^10/sqrt(2*m*k*e*T); //wavelength of
        thermal neutron(angstrom)
15

```

```

16 //Result
17 printf("\n wavelength of thermal neutron is %0.3 f
    angstrom",lamda)

```

---

#### Scilab code Exa 6.16 interplanar spacing

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.62*10^-34; //planck's constant(J-sec)
8 mn=1.67*10^-27; //mass of neutron(kg)
9 k=1.38*10^-23; //boltzmann constant(eV/deg)
10 T=300; //temperature(K)
11
12 //Calculations
13 E=k*T; //energy(J)
14 p=sqrt(2*mn*E); //momentum
15 d=h*10^10/p; //interplanar spacing(angstrom)
16
17 //Result
18 printf("\n interplanar spacing is %0.2 f angstrom",d)

```

---

#### Scilab code Exa 6.17 interplanar spacing

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration

```

```

7 h=6.62*10^-34;           //planck's constant(J-sec)
8 m=9*10^-31;           //mass of neutron(kg)
9 e=1.6*10^-19;         //charge of electron(c)
10 V=344;               //potential difference(V)
11 theta=60*%pi/180;    //angle(radian)
12
13 //Calculations
14 d=h*10^10/(2*sin(theta)*sqrt(2*m*e*V));           //
    interplanar spacing(angstrom)
15
16 //Result
17 printf("\n interplanar spacing is %0.2 f angstrom",d)
18 printf("\n answer in the book is wrong")

```

---

#### Scilab code Exa 6.18 uncertainty in momentum

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.6*10^-34;         //planck's constant(J-sec)
8 deltax=4*10^-10;     //uncertainty in position of
    electron(m)
9
10 //Calculations
11 delta_px=h/deltax;   //uncertainty in momentum(
    kg m/sec)
12
13 //Result
14 printf("\n uncertainty in momentum is %e kg m/sec",
    delta_px)

```

---

Scilab code Exa 6.19 uncertainty in position of electron

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.6*10^-34; //planck's constant(J-sec)
8 m=9.1*10^-31; //mass of electron(kg)
9 v=600; //speed(m/sec)
10 a=0.005/100; //accuracy(%)
11
12 //Calculations
13 deltav=v*a; //uncertainty in speed(kg m/sec)
14 delta_px=m*deltav; //uncertainty in momentum(
    kg m/sec)
15 deltax=h/delta_px; //uncertainty in position
    of electron(m)
16
17 //Result
18 printf("\n uncertainty in position of electron is
    %0.5f m",deltax)
19 printf("\n answer in the book is wrong")
```

---

Scilab code Exa 6.21 uncertainty in velocity

```
1 clear
2 //
3 //
4 //
5
```

```

6 //Variable declaration
7 h=6.63*10^-34; //planck's constant(J-sec)
8 m0=9.1*10^-31; //mass of electron(kg)
9 deltax=0.1*10^-10; //uncertainty in position
    of electron (m)
10
11 //Calculations
12 delta_p=h/deltax; //uncertainty in momentum(
    kg m/sec)
13 delta_v=delta_p/m0; //uncertainty in velocity(
    m/sec)
14
15 //Result
16 printf("\n uncertainty in momentum is %e kg m/sec",
    delta_p)
17 printf("\n uncertainty in velocity is %0.3f *10^7 m
    /sec",delta_v/10^7)

```

---

#### Scilab code Exa 6.22 uncertainty in velocity

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 me=9.10*10^-31; //mass of electron(kg)
8 mp=1.67*10^-27; //mass of electron(kg)
9
10 //Calculations
11 uv=mp/me; //uncertainty in velocity
12
13 //Result
14 printf("\n uncertainty in velocity is %0.3f ",uv)

```

---

Scilab code Exa 6.23 smallest possible uncertainty in position of electron

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.62*10^-34;           //planck's constant(J-sec)
8 m0=9*10^-31;           //mass of electron(kg)
9 v=3*10^7;              //velocity of electron(m/sec)
10 c=3*10^8;              //velocity of light(m/sec)
11
12 //Calculations
13 deltax_min=h*10^10*sqrt(1-(v^2/c^2))/(4*pi*m0*v);
14                       //smallest possible uncertainty in
15                       position of electron(angstrom)
16 //Result
17 printf("\n smallest possible uncertainty in
18 position of electron is %0.0f angstrom",
19 deltax_min)
```

---

Scilab code Exa 6.24 minimum uncertainty in velocity of electron

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.6*10^-34;           //planck's constant(J-sec)
```



```

8 m=9*10^-31; //mass of electron(kg)
9 deltax_max=10*10^-10; //length of box(m)
10
11 //Calculations
12 deltavx_min=h/(deltax_max*m); //minimum
    uncertainty in velocity of electron(m/s)
13
14 //Result
15 printf("\n minimum uncertainty in velocity of
    electron is %0.0f *10^5 m/s",deltavx_min/10^5)

```

---

Scilab code Exa 6.25 time required

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 dlamda=10^-4*10^-10; //width(m)
8 lamda=6000*10^-10; //wavelength(m)
9 c=3*10^8; //velocity of light(m/sec)
10
11 //Calculations
12 delta_t=lamda^2/(2*%pi*c*dlamda); //time
    required(second)
13
14 //Result
15 printf("\n time required is %0.1f *10^-8 second",
    delta_t*10^8)

```

---

Scilab code Exa 6.26 uncertainty in position of electron

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.63*10^-34; //planck's constant(J-sec)
8 m=9.1*10^-31; //mass of electron(kg)
9 v=3.5*10^7; //speed(cm/sec)
10 a=0.0098/100; //accuracy(%)
11
12 //Calculations
13 deltav=v*a; //uncertainty in speed(kg m/sec)
14 delta_p=m*deltav; //uncertainty in momentum(
    kg m/sec)
15 deltax=h/(4*pi*delta_p); //uncertainty in
    position of electron(m)
16
17 //Result
18 printf("\n uncertainty in position of electron is
    %0.4f *10^-8 m",deltax*10^8)
19 printf("\n answer in the book is wrong")

```

---

Scilab code Exa 6.27 uncertainty in position of dust particle

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.62*10^-34; //planck's constant(J-sec)
8 m=10^-6; //mass of electron(kg)
9 deltav=5.5*10^-20; //speed(m/sec)
10

```

```

11 // Calculations
12 delta_p=m*deltav;           //uncertainty in momentum(
    kg m/sec)
13 deltax=h/(4*pi*delta_p);    //uncertainty in
    position of dust particle(m)
14
15 //Result
16 printf("\n uncertainty in position of dust particle
    is %0.2f *10^-10 m",deltax*10^10)

```

---

Scilab code Exa 6.28 uncertainty in energy

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 delta_t=10^-12;           //life time(s)
8 hby2pi=1.054*10^-34;
9 e=1.6*10^-19;           //charge of electron(c)
10
11 //Calculations
12 deltaE=hby2pi/(2*e*delta_t);    //uncertainty in
    energy(eV)
13
14 //Result
15 printf("\n uncertainty in energy is %0.1f *10^-4 eV
    ",deltaE*10^4)

```

---

Scilab code Exa 6.29 minimum uncertainty in frequency

```

1 clear

```

```

2 //
3 //
4 //
5
6 //Variable declaration
7 delta_t=10^-8;           //life time(s)
8
9 //Calculations
10 deltav=1/(4*pi*delta_t);           //minimum
    uncertainty in frequency (s-1)
11
12 //Result
13 printf("\n minimum uncertainty in frequency is %0.0
    f *10^6 s-1",deltav/10^6)

```

---

#### Scilab code Exa 6.30 minimum energy

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.63*10^-34;           //planck's constant(J-sec)
8 e=1.6*10^-19;           //charge of electron(c)
9 delta_t=2.5*10^-14*10^-6;           //life time(s)
10
11 //Calculations
12 deltaE=h*10^-3/(4*pi*delta_t*e);           //minimum
    energy (keV)
13
14 //Result
15 printf("\n minimum energy is %0.5f keV",deltaE)
16 printf("\n answer in the book varies due to rounding
    off errors")

```

---

Scilab code Exa 6.31 least energy

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.63*10^-34;           //planck's constant(J-sec)
8 e=1.602*10^-19;        //charge of electron(c)
9 L=10^-10;              //width(m)
10 m=9.11*10^-31;        //mass of electron(kg)
11
12
13 //Calculations
14 E1=h^2/(8*m*e*L^2);    //least energy(eV)
15
16 //Result
17 printf("\n least energy is %0.3f eV",E1)
18 printf("\n answer in the book is wrong")
```

---

Scilab code Exa 6.32 3rd least energy

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.63*10^-34;           //planck's constant(J-sec)
8 e=1.6*10^-19;           //charge of electron(c)
```

```

 9 L=2.5*10^-10;           //width(m)
10 m=9.1*10^-31;         //mass of electron(kg)
11 n1=1;
12 n2=2;
13 n3=3;
14
15 // Calculations
16 E=h^2/(8*m*e*L^2);     //energy(eV)
17 E1=n1^2*h^2/(8*m*e*L^2); //1st least energy(eV)
18 E2=n2^2*h^2/(8*m*e*L^2); //2nd least energy(eV)
19 E3=n3^2*h^2/(8*m*e*L^2); //3rd least energy(eV)
20
21 // Result
22 printf("\n 1st least energy is %0.0f eV",E1)
23 printf("\n 2nd least energy is %0.0f eV",E2)
24 printf("\n 3rd least energy is %0.0f eV",E3)

```

---

### Scilab code Exa 6.33 3rd least energy

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.63*10^-34;           //planck's constant(J-sec)
8 e=1.6*10^-19;           //charge of electron(c)
9 L=10^-9;                 //width(m)
10 m=9.1*10^-31;          //mass of electron(kg)
11 n1=1;
12 n2=2;
13 n3=3;
14
15 // Calculations
16 lamda1=2*L*10^10/n1;    //wavelength in 1st

```

```

    energy state(angstrom)
17 lamda2=2*L*10^10/n2;      //wavelength in 2nd
    energy state(angstrom)
18 lamda3=2*L*10^10/n3;      //wavelength in 3rd
    energy state(angstrom)
19 E=h^2/(8*m*e*L^2);      //energy(eV)
20 E1=n1^2*h^2/(8*m*e*L^2);    //1st least energy(eV)
21 E2=n2^2*h^2/(8*m*e*L^2);    //2nd least energy(eV)
22 E3=n3^2*h^2/(8*m*e*L^2);    //3rd least energy(eV)
23
24 //Result
25 printf("\n wavelength in 1st energy state is %0.0f
    angstrom",lamda1)
26 printf("\n wavelength in 2nd energy state is %0.0f
    angstrom",lamda2)
27 printf("\n wavelength in 3rd energy state is %0.2f
    angstrom",lamda3)
28 printf("\n 1st least energy is %0.2f eV",E1)
29 printf("\n 2nd least energy is %0.4f eV",E2)
30 printf("\n 3rd least energy is %0.3f eV",E3)
31 printf("\n answers for 2nd and 3rd least energies
    varies due to rounding off errors")

```

---

Scilab code Exa 6.34 energy difference between ground state and 1st excited state

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.626*10^-34;      //planck 's constant(J-sec)
8 e=1.60*10^-19;      //charge of electron(c)
9 L=10^-10;           //width(m)
10 m=9.1*10^-31;      //mass of electron(kg)

```

```

11 n1=1;
12 n2=2;
13
14 // Calculations
15 E=h^2/(8*m*e*L^2); //energy (eV)
16 E1=n1^2*h^2/(8*m*e*L^2); //1st least energy (eV)
17 E2=n2^2*h^2/(8*m*e*L^2); //2nd least energy (eV)
18 Ed=E2-E1
19 // Result
20 printf("\n 1st least energy is %0.2f eV",E1)
21 printf("\n 2nd least energy is %0.0f eV",E2)
22 printf("\n energy difference between ground state
    and 1st excited state is %0.2f eV",Ed)
23 printf("\n answer in the book varies due to rounding
    off errors")

```

---

### Scilab code Exa 6.35 energy levels

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.6*10^-34; //planck's constant (J-sec)
8 e=1.6*10^-19; //charge of electron (c)
9 L=10^-1; //width (m)
10 m=10^-2; //mass of electron (kg)
11 n1=1;
12 n2=2;
13 n3=3;
14
15 // Calculations
16 E=h^2/(8*m*e*L^2); //energy (eV)
17 E1=n1^2*h^2/(8*m*e*L^2); //1st least energy (eV)

```



```

18 E2=n2^2*h^2/(8*m*e*L^2);      //2nd least energy (eV)
19 E3=n3^2*h^2/(8*m*e*L^2);      //3rd least energy (eV)
20
21 //Result
22 printf("\n 1st least energy is %0.1f *10^-45 eV",E1
    *10^45)
23 printf("\n 2nd least energy is %0.1f *10^-45 eV",E2
    *10^45)
24 printf("\n 3rd least energy is %0.1f *10^-45 eV",E3
    *10^45)
25 printf("\n energy levels are so close to each other
    that the energy states cannot be observed")

```

---

#### Scilab code Exa 6.36 quantum state

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 h=6.63*10^-34;      //planck's constant(J-sec)
8 e=1.602*10^-19;    //charge of electron(c)
9 L=0.2*10^-9;      //width(m)
10 n5=5;
11 En=10^3;          //energy (eV)
12 E5=230;          //energy of particle(eV)
13
14 //Calculations2
15 E5=230*e;        //energy (J)
16 E1=E5/n5^2;      //energy in 1st state(J)
17 m=h^2/(8*E1*L^2); //mass of particle(kg)
18 n=sqrt(En*e/E1); //quantum state
19
20 //Result

```

```
21 printf("\n mass of particle is %0.1f *10-31 kg",m
    *1031)
22 printf("\n quantum state is %0.1f ",n)
```

---

Scilab code Exa 6.37 probability of finding the particle

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 L=25*10-10;           //width (m)
8 deltax=5*10-10;       //interval (m)
9
10 //Calculations2
11 P=2*deltax/L;         //probability of finding the
    particle
12
13 //Result
14 printf("\n probability of finding the particle is %0
    .3f ",P)
```

---

# Chapter 8

## Semiconductor Physics

Scilab code Exa 8.1 ratio of density of electrons

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 ni=2.5*10^19; //concentration(per m^3)
8 d=4.4*10^28; //density(per m^3)
9 n=4*10^8; //number of Ge atoms
10
11 //Calculation
12 Na=d/n; //density of acceptor atoms
13 np=ni^2/Na;
14 npbyni=np/ni; //ratio of density of electrons
15
16 //Result
17 printf("\n ratio of density of electrons is %0.3f ",
    npbyni)
```

---

### Scilab code Exa 8.3 density of holes and electrons

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 me=9.1*10^-31;           //mass of electron(kg)
8 kb=1.38*10^-23;         //boltzmann constant
9 T=300;                   //temperature(K)
10 h=6.62*10^-34;          //planck's constant
11 Eg=0.7;                 //band gap(eV)
12 e=1.6*10^-19;           //charge(c)
13
14 //Calculation
15 x=2*%pi*me*kb*T/(h^2);
16 n=2*(x^(3/2))*exp(-Eg*e/(2*kb*T)); //density
    of holes and electrons(per m^3)
17
18 //Result
19 printf("\n density of holes and electrons is %0.3f
    *10^19 per m^3",n/10^19)
```

---

### Scilab code Exa 8.4 position of Fermi level

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 kb=1.38*10^-23;         //boltzmann constant
8 T=300;                   //temperature(K)
9 m=6;
```

```

10 Eg=0.7;      //band gap(eV)
11
12 //Calculation
13 x=3*kb*T*log(m)/4;
14 EF=(Eg/2)+x;      //position of Fermi level(eV
    )
15
16 //Result
17 printf("\n position of Fermi level is %0.3f eV",EF)
18 printf("\n answer in the book is wrong")

```

---

#### Scilab code Exa 8.5 position of Fermi level

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 T1=300;      //temperature (K)
8 T2=330;      //temperature (K)
9 E=0.3;      //band gap(eV)
10
11 //Calculation
12 Ec_Ef=T2*E/T1;      //position of Fermi level(eV)
13
14 //Result
15 printf("\n position of Fermi level is %0.3f eV",
    Ec_Ef)

```

---

#### Scilab code Exa 8.6 hall coefficient

```

1 clear

```

```

2 //
3 //
4 //
5
6 //Variable declaration
7 n=2.05*10^22;           //charge carrier density
8 e=1.602*10^-19;       //charge of electron
9
10 //Calculation
11 RH=1/(n*e);           //hall coefficient (m^3/C)
12
13 //Result
14 printf("\n hall coefficient is %0.3f *10^-4 m^3/C",
        RH*10^4)

```

---

#### Scilab code Exa 8.7 hall coefficient

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 n=5*10^28;           //charge carrier density
8 e=1.6*10^-19;       //charge of electron
9
10 //Calculation
11 RH=-1/(n*e);        //hall coefficient (m^3/C)
12
13 //Result
14 printf("\n hall coefficient is %0.3f *10^-9 m^3/C",
        RH*10^9)

```

---

### Scilab code Exa 8.8 hall coefficient

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 a=4.28*10^-10;           //side (m)
8 e=1.6*10^-19;           //charge of electron
9
10 //Calculation
11 n=2/(a^3);
12 RH=-1/(n*e);           //hall coefficient (m^3/C)
13
14 //Result
15 printf("\n hall coefficient is %0.3f *10^-9 m^3/C",
        RH*10^9)
```

---

### Scilab code Exa 8.9 hall coefficient

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 rho=9*10^-3;           //resistivity (ohm m)
8 mew=0.03;             //mobility (m^2/Vs)
9
10 //Calculation
11 sigma=1/rho;
12 RH=mew/sigma;         //hall coefficient (m^3/C)
13
14 //Result
```

```
15 printf("\n hall coefficient is %0.3f *10-4 m3/C",  
        RH*104)
```

---

#### Scilab code Exa 8.10 mobility

```
1 clear  
2 //  
3 //  
4 //  
5  
6 //Variable declaration  
7 rho=9*10-3;           //resistivity (ohm m)  
8 RH=3.6*10-4;         //hall coefficient (m3/C)  
9 e=1.6*10-19;         //charge of electron  
10  
11 //Calculation  
12 sigma=1/rho;  
13 rho=1/RH;  
14 n=rho/e;             //density of charge carrier (per m3)  
15 mew=sigma*RH;       //mobility (m2/Vs)  
16  
17 //Result  
18 printf("\n density of charge carrier is %0.5f *1022  
        per m3",n/1022)  
19 printf("\n mobility is %0.3f m2/Vs",mew)
```

---

#### Scilab code Exa 8.11 charge carrier concentration

```
1 clear  
2 //  
3 //  
4 //  
5
```



```

6 //Variable declaration
7 e=1.6*10^-19;           //charge of electron
8 z=0.3*10^-3;           //thickness(m)
9 VH=1*10^-3;            //hall voltage(V)
10 Ix=10*10^-3;           //current(A)
11 Bz=0.3;                 //magnetic field(T)
12
13 //Calculation
14 n=Ix*Bz/(VH*z*e);       //charge carrier
    concentration(m^-3)
15
16 //Result
17 printf("\n charge carrier concentration is %e m^-3",
    n)

```

---

#### Scilab code Exa 8.12 hall angle

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 rho=0.00912;           //resistivity(ohm m)
8 RH=3.55*10^-4;         //hall coefficient(m^3/C)
9 B=0.48;                 //flux density(Wb/m^2)
10
11 //Calculation
12 sigma=1/rho;
13 theta_H=atan(sigma*B*RH); //hall angle(radian)
14 theta_H=theta_H*180/%pi ; //hall angle(degrees
    )
15
16 //Result
17 printf("\n hall angle is %0.4f degrees",theta_H)

```



# Chapter 9

## Physics of Semiconductor Devices

Scilab code Exa 9.1 value of current

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 I0=0.3;      //current (micro A)
8 V=0.15;     //voltage (V)
9
10 //Calculations
11 I=I0*(exp(40*V)-1);      //value of current (micro A)
12
13 //Result
14 printf("\n value of current is %0.2f micro A",I)
```

---

Scilab code Exa 9.2 reverse saturation current

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 I=10*10^-3; //current (A)
8 V=0.75; //voltage (V)
9 T=300; //temperature (K)
10 eta=2;
11
12 //Calculations
13 VT=T/11600;
14 I0=I*10^9/(exp(V/(eta*VT))-1); //reverse
    saturation current (nA)
15
16 //Result
17 printf("\n reverse saturation current is %0.3f nA",
    I0)
18 printf("\n answer in the book is wrong")

```

---

### Scilab code Exa 9.3 voltage applied

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 J=10^5; //current density (amp/m^2)
8 T=300; //temperature (K)
9 eta=1;
10 J0=250*10^-3; //saturation current density (A/m
    ^2)
11

```

```

12 // Calculations
13 VT=T/11600;
14 x=(J/J0)+1;
15 V=log(x)*VT;      // voltage applied (V)
16
17 // Result
18 printf("\n voltage applied is %0.4f V",V)

```

---

#### Scilab code Exa 9.4 rectification ratio

```

1 clear
2 //
3 //
4 //
5
6 // Variable declaration
7 I0=4*10^-6;      // current (A)
8 T=273+25;       // temperature (K)
9 V=0.15;         // voltage (V)
10 eta=1;
11
12 // Calculations
13 VT=T/11600;
14 IF=I0*(exp(V/VT)-1); // forward current (A)
15 IR=I0*(exp(-V/VT)-1); // reverse current (A)
16 r=-IF/IR;      // rectification ratio
17
18 // Result
19 printf("\n rectification ratio is %0.3f ",r)
20 printf("\n answer in the book is wrong")

```

---

#### Scilab code Exa 9.5 voltage applied

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 T=300; //temperature(K)
8 eta=1;
9 I0=1;
10 I=-0.9*I0; //saturation current density (A/m^2)
11
12 //Calculations
13 VT=T/11600;
14 x=(I/I0)+1;
15 V=log(x)*VT; //voltage applied (V)
16
17 //Result
18 printf("\n voltage applied is %0.2f Volt",V)
```

---

# Chapter 10

## Dielectric Properties

Scilab code Exa 10.1 relative permittivity

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 P=4.3*10^-8; //polarisation(per cm^2)
8 epsilon0=8.85*10^-12; //relative permeability
   (F/m)
9 E=1000; //electric field (V/m)
10
11 //Calculations
12 epsilon_r=1+(P/(epsilon0*E)); //relative
   permittivity
13
14 //Result
15 printf("\n relative permittivity is %0.2f ",epsilon_r
   )
```

---

### Scilab code Exa 10.2 polarisation

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 k=4;
8 epsilon0=9*10^-12;          //relative permeability(F/
   m)
9 E=10^6;                    //electric field(V/m)
10
11 //Calculations
12 D=k*epsilon0*E;           //electric displacement(C/m^2)
13 P=epsilon0*E*(k-1);      //polarisation(C/m^2)
14
15 //Result
16 printf("\n electric displacement is %0.0f *10^-6 C/m
   ^2",D*10^6)
17 printf("\n polarisation is %0.0f *10^-6 C/m^2",P
   *10^6)
```

---

### Scilab code Exa 10.3 induced dipole moment

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 k=5;
8 epsilon0=8.86*10^-12;      //relative permeability
   (F/m)
9 D=5*10^-12;              //electric displacement(C/m^2)
```



```

10 V=0.5*10^-6;
11
12 // Calculations
13 E=D/(k*epsilon0); //electric field (N/C)
14 P=D*(1-(1/k)); //polarisation (C/m^2)
15 dm=P*V; //induced dipole moment(cm)
16
17 //Result
18 printf("\n electric field is %e N/C",E)
19 printf("\n polarisation is %e C/m^2",P)
20 printf("\n induced dipole moment is %e cm",dm)

```

---

#### Scilab code Exa 10.4 dipole moment

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 k=1.000074;
8 epsilon0=8.85*10^-12; //relative permeability
   (F/m)
9 E=1; //electric field (N/C)
10 n=2.69*10^25; //molecular density
11
12 //Calculations
13 p=epsilon0*E*(k-1)/n; //dipole moment(coulx metre
   )
14
15 //Result
16 printf("\n dipole moment is %0.2f *10^-41 coul x
   metre",p*10^41)

```

---

### Scilab code Exa 10.5 atomic polarizability

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 k=1.000134;
8 epsilon0=8.85*10^-12;           //relative permeability
   (F/m)
9 E=90000;                       //electric field (N/C)
10 N=6.023*10^26;                 //avagadro number
11
12 //Calculations
13 n=N/22.4;
14 p=epsilon0*E*(k-1)/n;         //dipole moment(coul-metre)
15 alpha=p/E;                   //atomic polarizability (coul-m^2/volt)
16
17 //Result
18 printf("\n dipole moment is %0.2f *10^-36 coul-metre
   ",p*10^36)
19 printf("\n atomic polarizability is %0.1f *10^-41
   coul-m^2/volt",alpha*10^41)
```

---

### Scilab code Exa 10.6 dipole moment

```
1 clear
2 //
3 //
4 //
5
```

```

6 //Variable declaration
7 k=7;
8 epsilon0=8.9*10^-12;          //relative permeability(
    F/m)
9 V0=100;          //potential difference (V)
10 d=10^-2;        //displacement (m)
11
12 //Calculations
13 E0=V0/d;        //electric field intensity (volt/m)
14 E=E0/k;         //electric field (N/C)
15 D=k*E*epsilon0; //electric displacement (C/m^2)
16 p=epsilon0*E*(k-1); //dipole moment (coul-metre)
17
18 //Result
19 printf("\n electric field is %0.2f *10^3 volt/m",E
    /10^3)
20 printf("\n electric displacement is %e C/m^2",D)
21 printf("\n dipole moment is %0.1f *10^-8 C/m^2",p
    *10^8)

```

---

#### Scilab code Exa 10.7 permittivity

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 epsilon0=8.85*10^-12;          //relative permeability
    (F/m)
8 chi=35.4*10^-12;          //electric susceptibility (coul
    ^2/nt-m^2)
9
10 //Calculations
11 k=1+(chi/epsilon0);          //dielectric constant

```

```

12 epsilon=epsilon0*k;           //permittivity (coul^2/nt-m
    ^2)
13
14 //Result
15 printf("\n dielectric constant is %0.3f ",k)
16 printf("\n permittivity is %0.2f *10^-12 coul^2/nt-m
    ^2",epsilon*10^12)

```

---

#### Scilab code Exa 10.8 dipole moment

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 epsilon0=8.85*10^-12;           //relative permeability
    (F/m)
8 E=100;                          //electric field (N/C)
9 epsilonR=1.000074;             //dielectric constant
10 n=2.68*10^27;                  //density
11
12 //Calculations
13 p=epsilon0*E*(epsilonR-1)/n;    //dipole moment(coul
    -metre)
14
15 //Result
16 printf("\n dipole moment is %0.4f *10^-41 C/m^2",p
    *10^41)
17 printf("\n answer in the book is wrong")

```

---

#### Scilab code Exa 10.9 relative permittivity

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 epsilon0=8.85*10^-12;           //relative permeability
   (F/m)
8 R=0.053*10^-9;                 //radius (nm)
9 N=9.8*10^26;                   //number of atoms
10
11 //Calculations
12 alphae=4*pi*epsilon0*R^3;      //electronic
   polarizability (Fm^2)
13 epsilon_r=1+(4*pi*N*R^3);     //relative
   permittivity
14
15 //Result
16 printf("\n electronic polarizability is %0.4f
   *10^-41 Fm^2", alphae*10^41)
17 printf("\n relative permittivity is %0.4f ", epsilon_r
   )

```

---

#### Scilab code Exa 10.10 electronic polarizability

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 epsilon0=8.85*10^-12;           //relative permeability
   (F/m)
8 epsilon_r=1.0000684;           //dielectric constant
9 N=2.7*10^25;                   //number of atoms

```

```

10
11 // Calculations
12 alphae=epsilon0*(epsilon0r-1)/N; // electronic
    polarizability (Fm^2)
13
14 // Result
15 printf("\n electronic polarizability is %e Fm^2",
    alphae)
16 printf("\n answer varies due to rounding off errors"
    )

```

---

#### Scilab code Exa 10.11 dielectric constant

```

1 clear
2 //
3 //
4 //
5
6 // Variable declaration
7 epsilon0=8.854*10^-12; // relative
    permeability (F/m)
8 alphae=10^-40; // dielectric polarizability (Fm
    ^2)
9 N=3*10^28; // number of atoms
10
11 // Calculations
12 epsilon0r=1+(N*alphae/epsilon0); // dielectric
    constant
13
14 // Result
15 printf("\n dielectric constant is %e ",epsilon0r)

```

---

#### Scilab code Exa 10.12 electronic polarizability

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 epsilon0=8.85*10^-12;           //relative permeability
   (F/m)
8 epsilon_r=1.0024;             //dielectric constant
9 N=2.7*10^25;                 //number of atoms
10
11 //Calculations
12 alphae=epsilon0*(epsilon_r-1)/N; //electronic
   polarizability (Fm^2)
13
14 //Result
15 printf("\n electronic polarizability is %0.1f
   *10^-40 Fm^2",alphae*10^40)
16 printf("\n answer in the book is wrong")

```

---

### Scilab code Exa 10.13 displacement

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 epsilon_r=1.0000684;         //dielectric constant
8 N=2.7*10^25;                 //number of atoms
9 X=1/(9*10^9);
10 E=10^6;                       //electric field (V/m)
11 Z=2;                          //atomic number
12 e=1.6*10^-19;                 //electron charge(coulomb)
13

```

```

14 // Calculations
15 R=((epsilon_r-1)/(4*pi*N))^(1/3); //radius of
    electron cloud (m)
16 x=X*E*R^3/(Z*e); //displacement (m)
17
18 //Result
19 printf("\n radius of electron cloud is %0.2f *10^-11
    m",R*10^11)
20 printf("\n displacement is %0.5f *10^-17 m",x*10^17)

```

---

#### Scilab code Exa 10.14 dielectric constant

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 epsilon0=8.85*10^-12; //dielectric constant
8 N=3*10^28; //number of atoms
9 alphae=10^-40; //dielectric polarizability (Fm
    ^2)
10
11 //Calculations
12 x=N*alphae/(3*epsilon0);
13 epsilon_r=(1+(2*x))/(1-x); //dielectric
    constant
14
15 //Result
16 printf("\n dielectric constant is %0.2f ",epsilon_r)

```

---

#### Scilab code Exa 10.15 dielectric constant



```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 epsilon0=8.85*10^-12; //dielectric constant
8 Na=6.023*10^26; //number of atoms
9 M=32; //atomic mass
10 alphae=3.28*10^-40; //dielectric
    polarizability (Fm^2)
11 rho=2.08*10^3; //density (kg/m^3)
12
13 //Calculations
14 x=Na*rho*alphae/(M*3*epsilon0);
15 epsilon_r=(1+(2*x))/(1-x); //dielectric
    constant
16
17 //Result
18 printf("\n dielectric constant is %0.1f ",epsilon_r)

```

---

#### Scilab code Exa 10.16 electronic polarizability

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 epsilon0=8.85*10^-12; //dielectric constant
8 Na=6.02*10^26; //number of atoms
9 epsilon_r=3.75; //dielectric constant
10 M=32; //atomic mass
11 rho=2050; //density (kg/m^3)
12 gama=1/3; //internal field constant

```

```

13
14 // Calculations
15 N=Na*rho/M; //number of atoms
16 alphae=((epsilon_r-1)/(epsilon_r+2))*(3*epsilon_0/N);
           //electronic polarizability (Fm^2)
17
18 //Result
19 printf("\n electronic polarizability is %0.2f
           *10^-40 Fm^2",alphae*10^40)

```

---

Scilab code Exa 10.17 ratio between electronic and ionic polarizability

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 epsilon_r=4.94; //dielectric constant
8 n2=2.69;
9
10 //Calculations
11 x=(epsilon_r-1)/(epsilon_r+2);
12 y=(n2-1)/(n2+2);
13 alpha=1/((x/y)-1); //ratio between electronic
           and ionic polarizability
14
15 //Result
16 printf("\n ratio between electronic and ionic
           polarizability is %0.3f ",alpha)

```

---

Scilab code Exa 10.18 percentage of ionic polarizability

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 epsilon_r=5.6;           //dielectric constant
8 n=1.5;
9
10 //Calculations
11 x=(epsilon_r+2)/(epsilon_r-1);
12 y=(n^2-1)/(n^2+2);
13 alpha=(1-(x*y))*100;    //percentage of ionic
    polarizability
14
15 //Result
16 printf("\n percentage of ionic polarizability is %0
    .1f percentage",alpha)
```

---

# Chapter 11

## Magnetic Properties

Scilab code Exa 11.1 susceptibility

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 mew0=4*%pi*10^-7;
8 B=0.2; //magnetic induction(web/m^2)
9 H=500; //magnetic field intensity(amp/m)
10
11 //Calculation
12 mewr=B/(mew0*H); //relative permeability
13 chi=mewr-1; //susceptibility
14
15 //Result
16 printf("\n relative permeability is %0.1f ",mewr)
17 printf("\n susceptibility is %0.1f ",chi)
18 printf("\n answer in the book varies due to rounding
    off errors")
```

---

### Scilab code Exa 11.2 absolute permeability

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 mew0=4*%pi*10^-7;
8 chi=948*10^-11;           //susceptibility
9
10 //Calculation
11 mewr=1+chi;             //relative permeability
12 mew=mewr*mew0;         //absolute permeability
13
14 //Result
15 printf("\n relative permeability is %0.3f ",mewr)
16 printf("\n absolute permeability is %0.3f *10^-6",
    mew*10^6)
```

---

### Scilab code Exa 11.3 relative permeability

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 H=6.5*10^-4;           //magnetizing force (amp/m)
8 M=1.4;                 //magnetic field (T)
9
10 //Calculation
```

```

11 chi=M/H;
12 mewr=1+chi;          //relative permeability
13
14 //Result
15 printf("\n relative permeability is %0.3f ",mewr)
16 printf("\n answer in the book is wrong")

```

---

#### Scilab code Exa 11.4 relative permeability

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 H=220;          //magnetizing force(amp/m)
8 M=3300;        //magnetic field(T)
9
10 //Calculation
11 chi=(M/H)+1;   //relative permeability
12
13 //Result
14 printf("\n relative permeability is %0.3f ",chi)

```

---

#### Scilab code Exa 11.5 permeability of rod

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 H=1600;        //magnetizing force(amp/m)

```

```

8 phi=4*10^-4;      // flux (weber)
9 A=4*10^-4;      // area (m^2)
10
11 // Calculation
12 B=phi/A;
13 mew=B/H;      // permeability of rod (weber/amp.m)
14
15 // Result
16 printf("\n permeability of rod is %0.3f *10^-3 weber
    /amp.m",mew*10^3)

```

---

#### Scilab code Exa 11.6 flux density

```

1 clear
2 //
3 //
4 //
5
6 // Variable declaration
7 H=10^6;      // magnetizing force (amp/m)
8 mew0=4*%pi*10^-7;
9 chi=1.5*10^-3;      // susceptibility
10
11 // Calculation
12 M=chi*H;      // magnetisation of material (A/m)
13 B=mew0*(M+H);      // flux density (T)
14
15 // Result
16 printf("\n magnetisation of material is %0.3f *10^3
    A/m",M/10^3)
17 printf("\n flux density is %0.3f T",B)
18 printf("\n answer in the book varies due to rounding
    off errors")

```

---

### Scilab code Exa 11.7 magnetic susceptibility

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 mew0=4*%pi*10^-7;
8 phi=2*10^-6; //flux (weber)
9 A=10^-4; //area (m^2)
10 N=300; //number of turns
11 l=30*10^-2; //length (m)
12 i=0.032; //current (ampere)
13
14 //Calculation
15 B=phi/A; //flux density (weber/metre^2)
16 n=N/l;
17 H=n*i; //magnetic intensity (amp-turn/metre)
18 mew=B/H; //permeability of ring (weber/amp-metre
19 )
20 mewr=mew/mew0; //relative permeability
21 chi=mewr-1; //magnetic susceptibility
22
23 //Result
24 printf("\n flux density is %0.3f *10^-2 weber/metre
25 ^2",B*10^2)
26 printf("\n magnetic intensity is %0.0f amp-turn/
27 metre",H)
28 printf("\n permeability of ring is %0.3f *10^-7
29 weber/amp-metre",mew*10^7)
30 printf("\n relative permeability is %0.1f ",mewr)
31 printf("\n magnetic susceptibility is %0.3f ",chi)
32 printf("\n answer in the book is wrong")
```

---



### Scilab code Exa 11.8 magnetic moment

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 new=6.5*10^15;           //frequency (Hz)
8 r=0.54*10^-10;         //radius (m)
9 e=1.6*10^-19;          //charge (coulomb)
10
11 //Calculation
12 mew_m=e*new*%pi*r^2;   //magnetic moment(A-m^2)
13
14 //Result
15 printf("\n magnetic moment is %0.2f *10^-24 A-m^2",
        mew_m*10^24)
16 printf("\n answer in the book varies due to rounding
        off errors")
```

---

### Scilab code Exa 11.9 bohrs magneton

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 e=1.6*10^-19;           //charge (coulomb)
8 m=9.1*10^-31;          //mass (kg)
9 h=6.64*10^-34;         //plank 's constant (Js)
```

```
10
11 // Calculation
12 mewb=e*h/(4*%pi*m); //bohr's magneton(J/T)
13
14 // Result
15 printf("\n bohrs magneton is %0.2f *10^-24 J/T",mewb
    *10^24)
```

---

# Chapter 12

## Lasers

Scilab code Exa 12.1 momentum of photon

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 e=1.6*10^-19; //charge(coulomb)
8 c=3*10^8; //velocity of matter wave(m/s)
9 h=6.62*10^-34; //plank's constant(Js)
10 lamda=6328*10^-10; //wavelength(m)
11
12 //Calculation
13 E=h*c/(lamda*e); //energy of photon(eV)
14 p=h/lamda; //momentum of photon(kg m/s)
15
16 //Result
17 printf("\n energy of photon is %0.2f eV",E)
18 printf("\n momentum of photon is %0.2f *10^-27 kg m/
    s",p*10^27)
```

---

### Scilab code Exa 12.2 energy of laser pulse

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 c=3*10^8; //velocity of matter wave(m/s)
8 h=6.62*10^-34; //plank's constant(Js)
9 lamda=7000*10^-10; //wavelength(m)
10 n=2.8*10^19; //number of ions
11
12 //Calculation
13 E=n*h*c/lamda; //energy of laser pulse(joule)
14
15 //Result
16 printf("\n energy of laser pulse is %0.2f joule",E)
```

---

### Scilab code Exa 12.3 coherence time

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 c=3*10^8; //velocity of matter wave(m/s)
8 l=2.945*10^-2;
9 lamda=5890*10^-10; //wavelength(m)
10
11 //Calculation
```

```

12 n=1/lamda;           //number of oscillations
13 tow_c=1/c;          //coherence time(s)
14
15 //Result
16 printf("\n number of oscillations is %0.0f *10^4",n
    /10^4)
17 printf("\n coherence time is %0.2f *10^-11 s",tow_c
    *10^11)

```

---

#### Scilab code Exa 12.4 intensity of beam

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 P=10*10^-3;          //power (W)
8 d=1.3*10^-3;         //diameter (m)
9
10 //Calculation
11 I=4*P/(%pi*d^2);    //intensity of beam(W/m^2)
12
13 //Result
14 printf("\n intensity of beam is %0.1f kW/m^2",I
    /10^3)

```

---

#### Scilab code Exa 12.5 number of ions

```

1 clear
2 //
3 //
4 //

```

```

5
6 //Variable declaration
7 c=3*10^8; //velocity of matter wave(m/s)
8 h=6.62*10^-34; //plank's constant(Js)
9 lamda=6940*10^-10; //wavelength(m)
10 P=1; //power(J)
11
12 //Calculation
13 n=P*lamda/(h*c); //number of ions
14
15 //Result
16 printf("\n number of ions is %0.2f *10^18",n/10^18)

```

---

#### Scilab code Exa 12.6 population ratio

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 c=3*10^8; //velocity of matter wave(m/s)
8 h=6.62*10^-34; //plank's constant(Js)
9 lamda=6*10^-7; //wavelength(m)
10 e=1.6*10^-19; //charge(coulomb)
11 k=8.6*10^-5;
12 T=300; //temperature(K)
13
14 //Calculation
15 E=h*c/(lamda*e); //energy(eV)
16 N=-E/(k*T); //population ratio
17
18 //Result
19 printf("\n population ratio is e^ %0.3f ",N)

```

---

### Scilab code Exa 12.7 coherence length

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 lamda=10.66*10^-6; //wavelength (m)
8 delta_lamda=10^-5*10^-9; //line width(m)
9
10 //Calculation
11 cl=lamda^2/delta_lamda; //coherence length (m)
12
13 //Result
14 printf("\n coherence length is %0.2f km",cl/10^3)
15 printf("\n answer varies due to rounding off errors"
    )
```

---

### Scilab code Exa 12.8 intensity of image

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 lamda=7000*10^-10; //wavelength (m)
8 d=5*10^-3; //aperture (m)
9 f=0.2; //focal length (m)
10 P=50*10^-3; //power (W)
11
```

```
12 // Calculation
13 d_theta=1.22*lamda/d;           // angular speed (radian)
14 A=(d_theta*f)^2;                 // areal speed (m^2)
15 I=P/A;                           // intensity of image (watt/m^2)
16
17 // Result
18 printf("\n areal speed is %0.3f *10^-8 m^2",A*10^8)
19 printf("\n intensity of image is %0.2f *10^5 watt/m
    ^2",I/10^5)
20 printf("\n answer given in the book is wrong")
```

---



# Chapter 13

## Fibre Optics

Scilab code Exa 13.1 maximum entrance angle

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 n1=1.5; //core refractive index
8 n2=1.48; //cladding refractive index
9 n=1;
10
11 //Calculations
12 NA=sqrt(n1^2-n2^2); //numerical aperture
13 i0=asin(NA/n); //maximum entrance angle(
    radian)
14 i0=i0*180/%pi ; //maximum entrance angle(
    degrees)
15
16 //Result
17 printf("\n numerical aperture is %0.5f ",NA)
18 printf("\n maximum entrance angle is %0.2f degrees",
    i0)
```

---

Scilab code Exa 13.2 acceptance angle

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 n0=1.33; //water refractive index
8 n2=1.59; //cladding refractive index
9 NA=0.2; //numerical aperture
10
11 //Calculations
12 n1=sqrt(NA^2+n2^2); //core refractive index
13 NA=sqrt(n1^2-n2^2)/n0; //numerical aperture
14 i0=asin(NA); //acceptance angle(radian)
15 i0=i0*180/%pi ; //acceptance angle(degrees)
16
17 //Result
18 printf("\n core refractive index is %0.4f ",n1)
19 printf("\n acceptance angle is %0.1f degrees",i0)
```

---

Scilab code Exa 13.3 acceptance angle

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 n1=1.36; //core refractive index
```

```

8 delta=0.025;    //relative difference
9
10 // Calculations
11 NA=n1*sqrt(2*delta);    //numerical aperture
12 i0=asin(NA);    //acceptance angle(radian)
13 i0=i0*180/%pi ;    //acceptance angle(degrees)
14
15 //Result
16 printf("\n numerical aperture is %0.3f ",NA)
17 printf("\n acceptance angle is %0.1f degrees",i0)

```

---

#### Scilab code Exa 13.4 critical angle

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 n1=1.5;    //core refractive index
8 n2=1.45;    //cladding refractive index
9
10 // Calculations
11 delta=(n1-n2)/n1;    //relative difference
12 NA=n1*sqrt(2*delta);    //numerical aperture
13 i0=asin(NA);    //acceptance angle(radian)
14 i0=i0*180/%pi ;    //acceptance angle(
    degrees)
15 theta_c=asin(n2/n1);    //critical angle(radian)
16 theta_c=theta_c*180/%pi ;    //critical
    angle(degrees)
17
18 //Result
19 printf("\n numerical aperture is %0.4f ",NA)
20 printf("\n acceptance angle is %0.2f degrees",i0)

```

```
21 printf("\n critical angle is %0.2f degrees",theta_c)
```

---

### Scilab code Exa 13.5 cladding refractive index

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 NA=0.22; //numerical aperture
8 delta=0.012; //relative difference
9
10 //Calculations
11 N=1-delta;
12 n1=sqrt(NA^2/(1-N^2)); //core refractive index
13 n2=N*n1; //cladding refractive index
14
15 //Result
16 printf("\n core refractive index is %0.2f ",n1)
17 printf("\n cladding refractive index is %0.3f ",n2)
```

---

### Scilab code Exa 13.6 critical angle

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 NA=0.40; //numerical aperture
8 delta=1/100; //relative difference
9
```

```

10 // Calculations
11 i0=asin(NA); //acceptance angle(radians)
12 i0=i0*180/%pi ; //acceptance angle(degrees)
13 N=1-delta;
14 thetac=asin(N); //critical angle(radians)
15 thetac=thetac*180/%pi ; //critical angle(degrees
    )
16
17 //Result
18 printf("\n acceptance angle is %0.1f degrees",i0)
19 printf("\n critical angle is %0.1f degrees",thetac)

```

---

#### Scilab code Exa 13.7 numerical aperture

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 vf=3*10^8; //velocity of light in free space(m/
    s)
8 vc=2*10^8; //velocity of light in core(m/s)
9 thetac=60*%pi/180; //critical angle(radians)
10
11 //Calculations
12 n1=vf/vc; //core refractive index
13 n2=n1*sin(thetac); //cladding refractive index
14 NA=sqrt(n1^2-n2^2); //numerical aperture
15
16 //Result
17 printf("\n core refractive index is %0.3f ",n1)
18 printf("\n cladding refractive index is %0.1f ",n2)
19 printf("\n numerical aperture is %0.3f ",NA)
20 printf("\n answer for numerical aperture varies due

```

to rounding off errors”)

---

### Scilab code Exa 13.9 critical angle

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 delta=0.03; //relative difference
8 i0=22*pi/180; //acceptance angle(radians)
9
10 //Calculations
11 NA=sin(i0); //numerical aperture
12 N=1-delta;
13 thetac=asin(N); //critical angle(radians)
14 theta_c=thetac*180/pi ; //critical angle(
    degrees)
15
16 //Result
17 printf("\n numerical aperture is %0.3f ",NA)
18 printf("\n critical angle is %0.2f degrees",theta_c)
```

---

### Scilab code Exa 13.10 velocity of light in fibre core

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 delta=0.0045; //relative difference
```

```

8 i0=0.115;          //acceptance angle(radians)
9 v=3*10^8;         //velocity of light(m/s)
10
11 //Calculations
12 NA=sin(i0);       //numerical aperture
13 n1=NA/sqrt(2*delta); //core refractive index
14 vcore=v/n1;       //velocity of light in fibre core(m
    /s)
15
16 //Result
17 printf("\n velocity of light in fibre core is %0.3f
    *10^8 m/s",vcore/10^8)

```

---

#### Scilab code Exa 13.11 diameter of core

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 V=2.405;          //V-number
8 lamda=8500*10^-10; //wavelength(m)
9 n1=1.48;          //core refractive index
10 n2=1.47;         //cladding refractive index
11
12 //Calculations
13 d=V*lamda/(%pi*sqrt(n1^2-n2^2)); //diameter of
    core(m)
14
15 //Result
16 printf("\n diameter of core is %0.2f *10^-6 m",d
    *10^6)

```

---

**Scilab code Exa 13.12** maximum radius for fibre

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 V=2.405; //V-number
8 lamda=1300*10^-3; //wavelength(micro m)
9 n1=1.466; //core refractive index
10 n2=1.46; //cladding refractive index
11
12 //Calculations
13 r=V*lamda/(2*%pi*sqrt(n1^2-n2^2)); //maximum
    radius for fibre(micro m)
14
15 //Result
16 printf("\n maximum radius for fibre is %0.2f micro m
    ",r)
17 printf("\n answer varies due to rounding off errors"
    )
```

---

**Scilab code Exa 13.13** diameter of fibre core

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 lamda=1.3; //wavelength(micro m)
```



```

8 n1=1.5;           //core refractive index
9 Nm=1100;         //number of modes
10 delta=0.01;    //refractive index difference
11
12 //Calculations
13 d=lamda*sqrt(Nm/delta)/(%pi*n1);    //diameter of
    fibre core(micro m)
14
15 //Result
16 printf("\n diameter of fibre core is %0.1f micro m",
    d)

```

---

Scilab code Exa 13.14 number of guided modes

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 lamda=1.1*10^-6;    //wavelength(m)
8 r=60/2*10^-6;      //radius(m)
9 NA=0.25;           //numerical aperture
10
11 //Calculations
12 V=2*%pi*r*NA/lamda;
13 Nm=V^2/4;         //number of guided modes
14
15 //Result
16 printf("\n number of guided modes is %0.3f ",Nm)

```

---

Scilab code Exa 13.15 fibre loss

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 L=500/1000; //length(km)
8 P0byPi=25/100; //optical power
9
10 //Calculations
11 dB=-10*log10(P0byPi)/L; //fibre loss (dB/km)
12
13 //Result
14 printf("\n fibre loss is %0.4f dB/km",dB)
```

---

# Chapter 14

## Acoustics of Buildings and Acoustic Quieting

Scilab code Exa 14.1 total absorption in hall

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 V=7500; //volume(m^3)
8 T=1.5; //time(sec)
9
10 //Calculations
11 aS=0.165*V/T; //total absorption in hall(OWU)
12
13 //Result
14 printf("\n total absorption in hall is %0.3f OWU",aS
    )
```

---

## Scilab code Exa 14.2 reverberation time with audience in cushioned chairs

```
1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 V=1500; //volume(m^3)
8 A1=112; //area of plastered walls(m^2)
9 A2=130; //area of wooden floor(m^2)
10 A3=170; //area of plastered ceiling(m^2)
11 A4=20; //area of wooden door(m^2)
12 n=100; //number of cushioned chairs
13 A5=120; //area of audience(m^2)
14 C1=0.03; //coefficient of absorption in plastered
    walls
15 C2=0.06; //coefficient of absorption in wooden
    floor
16 C3=0.04; //coefficient of absorption in plastered
    ceiling
17 C4=0.06; //coefficient of absorption in wooden
    door
18 C5=1.0; //coefficient of absorption in cushioned
    chairs
19 C6=4.7; //coefficient of absorption in audience
20
21 //Calculations
22 a1=A1*C1; //absorption due to plastered walls
23 a2=A2*C2; //absorption due to wooden floor
24 a3=A3*C3; //absorption due to plastered ceiling
25 a4=A4*C4; //absorption due to wooden door
26 a5=n*C5; //absorption due to cushioned chairs
27 a6=A5*C6; //absorption due to audience
28 aS=a1+a2+a3+a4+a5; //total absorption in hall
29 T1=0.165*V/aS; //reverberation time when hall
    is empty(sec)
30 T2=0.165*V/(aS+a6); //reverberation time with
```

```

    full capacity of audience(sec)
31 T3=0.165*V/((n*C6)+aS);    //reverberation time with
    audience in cushioned chairs(sec)
32
33 //Result
34 printf("\n reverberation time when hall is empty is
    %0.2f sec",T1)
35 printf("\n reverberation time with full capacity of
    audience is %0.3f sec",T2)
36 printf("\n reverberation time with audience in
    cushioned chairs is %0.2f sec",T3)

```

---

#### Scilab code Exa 14.3 reverberation time

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 V=1200;    //volume(m^3)
8 a1=220;    //area of wall(m^2)
9 a2=120;    //area of floor(m^2)
10 a3=120;    //area of ceiling(m^2)
11 C1=0.03;    //coefficient of absorption in wall
12 C2=0.80;    //coefficient of absorption in floor
13 C3=0.06;    //coefficient of absorption in ceiling
14
15 //Calculations
16 A1=a1*C1;    //absorption due to plastered walls
17 A2=a2*C2;    //absorption due to wooden floor
18 A3=a3*C3;    //absorption due to plastered ceiling
19 aS=a1+a2+a3;    //total absorption in hall
20 abar=(A1+A2+A3)/aS;    //average sound absorption
    coefficient

```

```

21 AS=abar*aS;           //total absorption of room(metric
    sabines)
22 T=0.165*V/AS;        //reverberation time(sec)
23
24 //Result
25 printf("\n average sound absorption coefficient is
    %0.2f ",abar)
26 printf("\n reverberation time is %0.1f sec",T)

```

---

#### Scilab code Exa 14.4 acoustic power

```

1 clear
2 //
3 //
4 //
5
6 //Variable declaration
7 I0=10^-12;           //standard intensity level(watt/m^2)
8 A=1.4;               //area(m^2)
9 il=60;               //intensity level(decibels)
10
11 //Calculations
12 x=10^(il/10);
13 I=x*10^-12;         //intensity level(watt/m^2)
14 Ap=I*A;             //acoustic power(watt)
15
16 //Result
17 printf("\n acoustic power is %e watt",Ap)
18 printf("\n answer in the book is wrong")

```

---