Scilab Textbook Companion for Engineering Physics by S. M. Naidu[1]

Created by
Siddharth
M.tech
Electronics Engineering
NITK
College Teacher
None
Cross-Checked by
Reshma

July 31, 2019

[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: Engineering Physics

Author: S. M. Naidu

Publisher: Pearson, New Delhi

Edition: 1

Year: 2009

ISBN: 9788131730928

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 STRUCTURES 9

3 Crystal planes X ray diffraction and defects in solids 13

4 PRINCIPLES OF QUANTUM MECHANICS 23

5 ELECTRON THEORY OF METALS 31

6 DIELECTRIC PROPERTIES 39

7 Magnetic properties 47

8 Semiconductors 53

9 Superconductivity 62

10 Lasers 66

11 Fibre Optics 68

14 Optics 76

List of Scilab Codes

Exa 1.1	bond energy	5
Exa 1.2	total cohesive energy	5
Exa 1.3	cohesive energy	6
Exa 1.4	potential energy	7
Exa 1.5	cohesive energy	7
Exa 1.6	cohesive energy	8
Exa 2.1	free volume per unit cell	9
Exa 2.2	lattice constant	9
Exa 2.3	lattice constant	10
Exa 2.4	number of atoms per m3	10
Exa 2.5	number of atoms per sq mm	11
Exa 2.6	Calculate density	11
Exa 3.1	glancing angle	13
Exa 3.2	maximum order of diffraction	14
Exa 3.3	fraction of vacancy sites	14
Exa 3.4	ratio	15
Exa 3.5	lattice parameter of nickel	15
Exa 3.6	order of diffraction	16
Exa 3.7	radius of the atom	16
Exa 3.8	order of diffraction	17
Exa 3.9	glancing angle	18
Exa 3.10	cube edge of unit cell	18
Exa 3.11	lattice parameter of nickel	19
Exa 3.12	interplanar spacing	20
Exa 3.13	glancing angle	20
Exa 3.14	glancing angle	21
Exa 4.1	wavelength	23
Exa 4.2	velocity	23

Exa 4.3 nergy value in states 24
Exa 4.4 minimum energy 25
Exa 4.5 wavelength 25
Exa 4.6 minimum energy 26
Exa 4.8 minimum energy 26
Exa 4.9 de broglie wavelength 27
Exa 4.10 energy value in states 27
Exa 4.11 interplanar spacing 28
Exa 4.12 energy required to pump an electron 28
Exa 4.13 minimum energy 29
Exa 4.14 wavelength 30
Exa 5.1 temperature 31
Exa 5.2 total number of free electrons 32
Exa 5.3 relaxation time 32
Exa 5.4 relaxation time 33
Exa 5.5 drift velocity 33
Exa 5.6 mobility of electrons 34
Exa 5.7 mobility of electrons 35
Exa 5.8 relaxation time 35
Exa 5.9 thermal velocity 36
Exa 5.10 mean free path 37
Exa 5.11 fermi energy 37
Exa 5.12 drift velocity of free electrons 38
Exa 6.1 dielectric constant of material 39
Exa 6.2 charge on plates 39
Exa 6.3 electronic polarizability 40
Exa 6.4 voltage 40
Exa 6.5 polarization 41
Exa 6.6 electronic polarizability 41
Exa 6.7 orientational polarization 42
Exa 6.8 displacement 43
Exa 6.9 voltage 43
Exa 6.10 polarizability 44
Exa 6.11 electronic polarizability 45
Exa 6.12 electronic polarizability 45
Exa 7.1 magnetic moment 47
Exa 7.2 flux density 47
Exa 7.3 magnetisation and flux density 48

Exa 7.4 magnetisation and flux density 48
Exa 7.5 magnetic moment 49
Exa 7.6 magnetizing force and relative permeability 49
Exa 7.7 change in magnetic moment 50
Exa 7.8 change in magnetic moment 51
Exa 7.9 susceptibility 51
Exa 7.10 relative permeability 52
Exa 8.1 resistivity 53
Exa 8.2 number of donor atoms 53
Exa 8.3 diffusion coefficient 54
Exa 8.4 mobility of holes 54
Exa 8.5 intrinsic concentration 55
Exa 8.6 resistivity 55
Exa 8.7 conductivity 56
Exa 8.8 carrier concentration 57
Exa 8.9 conductivity 57
Exa 8.10 conductivity 58
Exa 8.11 resistivity 58
Exa 8.12 hole concentration 59
Exa 8.13 density of donor atoms 60
Exa 8.14 energy gap 60
Exa 8.15 diffusion coefficient 60
Exa 8.16 energy gap 61
Exa 9.1 transition temperature 62
Exa 9.2 frequency 62
Exa 9.3 critical field 63
Exa 9.4 temperature 63
Exa 9.5 critical field 64
Exa 9.6 frequency 64
Exa 9.7 critical field 65
Exa 10.1 energy gap 66
Exa 10.2 wavelength 66
Exa 11.1 fractional refractive indices change 68
Exa 11.2 Acceptance angle 69
Exa 11.3 fractional refractive indices change 69
Exa 11.4 Numerical aperture 70
Exa 11.5 refractive index 70
Exa 11.6 Numerical aperture 71

Exa 11.7 Acceptance angle 71
Exa 11.8 refractive index 72
Exa 11.9 Acceptance angle 72
Exa 11.10 fractional refractive indices change 73
Exa 11.11 Critical angle 73
Exa 11.12 Numerical aperture 74
Exa 14.1 ratio of maximum intensity to minimum intensity 76
Exa 14.2 angular position of 1st minimum 76
Exa 14.3 visible region 77
Exa 14.4 wavelength 78
Exa 14.5 radius of curvature 79
Exa 14.6 least distance of the point 79
Exa 14.7 thickness 80
Exa 14.8 fringe width 80
Exa 14.9 separation between slits 81
Exa 14.10 ratio of maximum intensity to minimum intensity 81
Exa 14.11 diameter of 25th ring 82

Chapter 1

Bonding in Solids

Scilab code Exa 1.1 `bond energy`

```

//Variable declaration
e=1.6*10**-19;   //charge of electron(c)
epsilon0=8.85*10**-12;    //permittivity(C/Nm)
r0=236*10**-12;     //seperation(m)
IE=5.14;     //ionisation energy of Na(eV)
Ea=-3.65;     //electron affinity(eV)

//Calculation
V=-e**2/(4*e*%pi*epsilon0*r0);
BE=IE+Ea+(V);       //bond energy(eV)

//Result
printf('bond energy is %0.3f    eV    \n',(BE))
```

Scilab code Exa 1.2 `total cohesive energy`

```

```

```
//Variable declaration
e=1.602*10**-19;    //charge of electron(c)
epsilon0=8.85*10**-12;     //permittivity(C/Nm)
r0=0.314*10**-9;      //seperation(m)
A=1.75;     //madelung constant
n=5.77;     //repulsive exponent value
IE=4.1;      //ionisation energy of K(eV)
Ea=3.6;      //electron affinity(eV)

//Calculation
E=-A*e**2*(1-(1/n))/(4*e*%pi*epsilon0*r0);       //
   energy(eV)
Ce=E/2;       //cohesive energy per atom(eV)
x=IE-Ea;      //energy(eV)
CE=Ce+(x/2);     //total cohesive energy per atom(eV)

//Result
printf('total cohesive energy per atom is %0.3f
   eV   \n',(CE))
printf('answer varies due to ing off errors')
```

Scilab code Exa 1.3 cohesive energy

```

//Variable declaration
e=1.602*10**-19;    //charge of electron(c)
epsilon0=8.85*10**-12;     //permittivity(C/Nm)
r0=0.281*10**-9;      //seperation(m)
alpham=1.748;     //madelung constant
n=9;     //repulsive exponent value

//Calculation
E=-alpham*e**2*(1-(1/n))/(4*e*%pi*epsilon0*r0);
        //cohesive energy(eV)

```

```
//Result
printf('cohesive energy is %0.3f  eV       \n',(E))
```

Scilab code Exa 1.4 potential energy

```

//Variable declaration
e=1.6*10**-19;    //charge of electron(c)
epsilon0=8.85*10**-12;     //permittivity(C/Nm)
r0=2.5*10**-10;      //seperation(m)

//Calculation
PE=e**2/(4*e*%pi*epsilon0*r0);      //potential
   energy(eV)

//Result
printf('potential energy is %0.3f   eV  \n',(PE))
```

Scilab code Exa 1.5 cohesive energy

```

//Variable declaration
m=1;
n=9;      //repulsive exponent value
a=1.748*10**-28;
r0=0.281*10**-9;      //seperation(m)
e=1.6*10**-19;
//Calculation
Ur0=-a*(1-(m/n))/(e*r0**m);     //cohesive energy(eV)

//Result
printf('cohesive energy is %0.3f   eV  \n',(Ur0))
```

Scilab code Exa 1.6 `cohesive energy`

```

//Variable declaration
e=1.6*10**-19;    //charge of electron(c)
epsilon0=8.85*10**-12;     //permittivity(C/Nm)
r0=0.281*10**-9;      //seperation(m)
IE=5.14;      //ionisation energy of Na(eV)
Ea=-3.61;      //electron affinity(eV)

//Calculation
V=-e**2/(4*e*%pi*epsilon0*r0);
CE=IE+Ea+(V);        //cohesive energy(eV)

//Result
printf('cohesive energy is %0.3f   eV  \n',CE)
```

Chapter 2

CRYSTAL STRUCTURES

Scilab code Exa 2.1 `free volume per unit cell`

```

//Variable declaration
r=0.1249;       //atomic radius(nm)
n=2;       //number of atoms

//Calculation
a=4*r/sqrt(3);       //edge length(m)
V=a**3;                    //volume(nm**3)
v=4*%pi*r**3*n/3;      //volume of atoms(nm**3)
Fv=V-v;       //free volume/unit cell(nm**3)

//Result
printf('free volume/unit cell is %0.3f nm**3 \n'
   ,(Fv))
```

Scilab code Exa 2.2 `lattice constant`

```
//Variable declaration
```

```
n=2;      //number of atoms
M=6.94;    //atomic weight(kg)
rho=530;    //density(kg/m**3)
Na=6.02*10**26;     //avagadro number

//Calculation
a3=n*M/(rho*Na);
a=a3**(1/3);      //lattice constant(m)

//Result
printf('lattice constant is %0.3f   angstrom   \n',(a
   *10**10))
```

Scilab code Exa 2.3 `lattice constant`

```

//Variable declaration
n=2;      //number of atoms
M=55.85;    //atomic weight(kg)
rho=7860;    //density(kg/m**3)
Na=6.02*10**26;     //avagadro number

//Calculation
a3=n*M/(rho*Na);
a=a3**(1/3);      //lattice constant(m)

//Result
printf('lattice constant is %0.3f  angstrom   \n',(a
   *10**10))
```

Scilab code Exa 2.4 `number of atoms per m3`

```

```

```
//Variable declaration
a=0.356*10**-9;      //lattice constant(m)
n=8;      //number of atoms

//Calculation
N=n/a**3;      //number of atoms per m**3

//Result
printf('//number of atoms per m**3 is %0.3f  *10**27
      \n',(N/10**27))
```

Scilab code Exa 2.5 number of atoms per sq mm

```

//Variable declaration
a=3.5;      //lattice constant(angstrom)

//Calculation
A=a**2;
N=10**7*10**7/A;     //number of atoms per sq. mm

//Result
printf('number of atoms per sq. mm is %0.3f  *10**12
      \n',(N/10**12))
```

Scilab code Exa 2.6 Calculate density

```

//Variable declaration
n=8;      //number of atoms
a=5.62*10**-10;      //lattice constant(m)
M=72.59;    //atomic weight(kg)
Na=6.02*10**26;     //avagadro number
```

```

//Calculation
rho=n*M/(a**3*Na);    //density(kg/m**3)

//Result
printf('density is %0.3f   kg/m**3  \n',(rho))
```

Chapter 3

Crystal planes X ray diffraction and defects in solids

Scilab code Exa 3.1 `glancing angle`

```

//Variable declaration
lamda=0.071*10**-9;      //wavelength(m)
a=0.28*10**-9;     //lattice constant(m)
h=1;
k=1;
l=0;
n=2;     //order of diffraction

//Calculation
d=a/sqrt(h**2+k**2+l**2);
x=n*lamda/(2*d);
theta=asin(x);     //angle(radian)
theta=theta*180/%pi;     //glancing angle(degrees)

//Result
printf('glancing angle is %0.3f   degrees  \n',int(
   theta))
```

Scilab code Exa 3.2 maximum order of diffraction

```
//Variable declaration
n=1;     //order of diffraction
theta1=8+(35/60);     //angle(degrees)
d=0.282;      //spacing(nm)
theta2=90;

//Calculation
theta1=theta1*%pi/180;     //angle(radian)
lamda=2*d*sin(theta1)/n;     //wavelength(nm)
theta2=theta2*%pi/180;     //angle(radian)
nmax=2*d/lamda;     //maximum order of diffraction

//Result
printf('wavelength is %0.3f    nm \n',(lamda))
printf('maximum order of diffraction is %0.3f     \n
   ',(nmax))
```

Scilab code Exa 3.3 fraction of vacancy sites

```
//Variable declaration
T1=500+273;     //temperature(K)
T2=1000+273;    //temperature(K)
f=1*10**-10;    //fraction

//Calculation
x=(T1/T2);
y=(log(f));
w=(x*y);
F=exp(w);     //fraction of vacancy sites

```

```
//Result
printf('fraction of vacancy sites is %0.3f  *10**-7
    \n',(F*10**7))
```

Scilab code Exa 3.4 ratio

```
//Variable declaration
a=1;     //assume
h1=1;
k1=0;
l1=0;
h2=1;
k2=1;
l2=0;
h3=1;
k3=1;
l3=1;

//Calculation
d100=a*6/(h1**2+k1**2+l1**2);
d110=a*6/(h2**2+k2**2+l2**2);
d111=a*(6)/(h3**2+k3**2+l3**2);

//Result
printf('ratio is %0.3f:%0.3f:%0.3f',sqrt( d100),
   sqrt(d110), sqrt( d111))
```

Scilab code Exa 3.5 lattice parameter of nickel

```
//Variable declaration
n=1;     //order of diffraction
theta=38.2;     //angle(degrees)
lamda=1.54;     //wavelength(angstrom)
```

```
h=2;
k=2;
l=0;

//Calculation
theta=theta*%pi/180;    //angle(radian)
d=n*lamda/(2*sin(theta));
a=d*sqrt(h**2+k**2+l**2);    //lattice parameter of
   nickel(angstrom)

//Result
printf('lattice parameter of nickel is %0.3f
   angstrom   \n',(a))
```

Scilab code Exa 3.6 order of diffraction

```
//Variable declaration
theta=90;    //angle(degrees)
lamda=1.5;    //wavelength(angstrom)
d=1.6;    //spacing(angstrom)

//Calculation
theta=theta*%pi/180;    //angle(radian)
n=2*d*sin(theta)/lamda;    //order of diffraction

//Result
printf('order of diffraction is %0.3f     \n',int(n)
   )
```

Scilab code Exa 3.7 radius of the atom

```
//Variable declaration
h=1;
```

```
k=1;
l=0;
d=0.203*10**-9;     //spacing(m)

//Calculation
a=d*sqrt(h**2+k**2+l**2);     //length of unit cell(m
    )
V=a**3;     //volume of unit cell(m**3)
r=sqrt(3)*a/4;     //radius of the atom(m)

//Result
printf('length of unit cell is %0.3f   *10**-9 m  \n
    ',(a*10**9))
printf('volume of unit cell is %0.3f  *10**-27 m**3
     \n',(V*10**27))
printf('radius of the atom is %0.3f  *10**-9 m   \n'
    ,(r*10**9))
```

Scilab code Exa 3.8 `order of diffraction`

```
//Variable declaration
theta=90;     //angle(degrees)
lamda=1.5;     //wavelength(angstrom)
d=1.6;     //spacing(angstrom)

//Calculation
theta=theta*%pi/180;     //angle(radian)
n=2*d*sin(theta)/lamda;     //order of diffraction

//Result
printf('order of diffraction is %0.3f     \n',int(n)
    )
```

Scilab code Exa 3.9 `glancing angle`

```

//Variable declaration
lamda=0.065;     //wavelength(nm)
a=0.26;       //edge length(nm)
h=1;
k=1;
l=0;
n=2;

//Calculation
d=a/sqrt(h**2+k**2+l**2);
x=n*lamda/(2*d);
theta=asin(x);        //glancing angle(radian)
theta=theta*180/%pi;     //glancing angle(degrees)
theta_d=int(theta);
theta_m=(theta-theta_d)*60;
theta_s=(theta_m-int(theta_m))*60;

//Result

printf('glancing angle is %2d degrees %2d minutes
   %2d seconds',theta_d,int(theta_m),int(theta_s))
printf('answer varies due to approximating off
   errors')
```

Scilab code Exa 3.10 `cube edge of unit cell`

```

//Variable declaration
lamda=1.54;     //wavelength(angstrom)
h=1;
k=1;
l=1;
```

```
n=1;
theta=19.2;     //angle(degrees)

//Calculation
theta=theta*%pi/180;     //angle(radian)
d=n*lamda/(2*sin(theta));
a=d*sqrt(h**2+k**2+l**2);       //cube edge of unit
   cell(angstrom)

//Result
printf('cube edge of unit cell is %0.3f    angstrom
   \n',(a))
```

Scilab code Exa 3.11 `lattice parameter of nickel`

```

//Variable declaration
lamda=1.54;     //wavelength(angstrom)
h=2;
k=2;
l=0;
n=1;
theta=38.2;     //angle(degrees)

//Calculation
theta=theta*%pi/180;     //angle(radian)
d=n*lamda/(2*sin(theta));
a=d*sqrt(h**2+k**2+l**2);     //lattice parameter of
   nickel(angstrom)

//Result
printf('lattice parameter of nickel is %0.3f
   angstrom     \n',(a))
```

Scilab code Exa 3.12 interplanar spacing

```

//Variable declaration
a=0.36;        //edge length(nm)
h1=1;
k1=1;
l1=1;
h2=3;
k2=2;
l2=1;

//Calculation
d1=a/sqrt(h1**2+k1**2+l1**2);     //interplanar
   spacing for (111)(nm)
d2=a/sqrt(h2**2+k2**2+l2**2);     //interplanar
   spacing for (321)(nm)

//Result
printf('interplanar spacing for (111) is %0.3f     nm
    \n',(d1))
printf('interplanar spacing for (321) is %0.3f   nm
     \n',(d2))
```

Scilab code Exa 3.13 glancing angle

```

//Variable declaration
lamda=0.675;     //wavelength(angstrom)
n=3;     //order of diffraction
theta=5+(25/60);     //angle(degrees)

```

```
//Calculation
theta=theta*%pi/180;    //angle(radian)
d=lamda/(2*sin(theta));
theta3=asin(3*lamda/(2*d));     //glancing angle(
   radian)
theta3=theta3*180/%pi;     //glancing angle(degrees)
theta_d=int(theta3);
theta_m=(theta3-theta_d)*60;

//Result
printf('glancing angle is %0.3f    degrees %0.3f
   minutes \n',theta_d,int(theta_m))
printf('glancing angle is %2d degrees %2d minutes ',
   theta_d,int(theta_m))
printf('answer varies due to approximating off
   errors\n')
```

Scilab code Exa 3.14 glancing angle

```

//Variable declaration
lamda=0.79;     //wavelength(angstrom)
n=3;    //order of diffraction
d=3.04;     //spacing(angstrom)

//Calculation
x=(n*lamda/(2*d));
theta=asin(x);          //glancing angle(radian)
theta=theta*180/%pi;     //glancing angle(degrees)
theta_d=int(theta);
theta_m=(theta-theta_d)*60;
theta_s=(theta_m-int(theta_m))*60;

//Result
printf('glancing angle is %2d degrees %2d minutes
    %2d seconds',theta_d,int(theta_m),int(theta_s))
printf('answer given in the book is wrong\n')
```

Chapter 4

PRINCIPLES OF QUANTUM MECHANICS

Scilab code Exa 4.1 `wavelength`

```

//Variable declaration
e=1.6*10**-19;
m=9.1*10**-31;      //mass(kg)
h=6.63*10**-34;     //planck's constant
E=2000;       //energy(eV)

//Calculation
lamda=h/sqrt(2*m*E*e);       //wavelength(m)

//Result
printf('wavelength is %0.4f nm\n ',(lamda*10**9))
```

Scilab code Exa 4.2 `velocity`

```

```

```
//Variable declaration
e=1.6*10**-19;
m=9.1*10**-31;     //mass(kg)
h=6.626*10**-34;     //planck's constant
lamda=1.66*10**-10;     //wavelength(m)

//Calculation
v=h/(m*lamda);      //velocity(m/s)
E=h**2/(2*m*e*lamda**2);     //kinetic energy(eV)

//Result
printf('velocity is %0.3f   *10**4 m/s  \n',(v
   /10**4))
printf('answer varies due to approximating off
   errors\n')
printf('kinetic energy is %0.3f  eV   \n',(E))
```

Scilab code Exa 4.3 nergy value in states

```

//Variable declaration
n=1;
e=1.6*10**-19;
m=9.1*10**-31;     //mass(kg)
h=6.63*10**-34;     //planck's constant
L=1*10**-10;        //width(m)

//Calculation
E1=n**2*h**2/(8*m*e*L**2);       //energy value in
   ground state(eV)
E2=4*E1;       //energy value in 1st state(eV)
E3=9*E1;       //energy value in 2nd state(eV)

//Result
printf('energy value in ground state is %0.4f eV",(
    E1))
printf('\nenergy value in 1st state is %0.2f eV",(E2
    ))
printf('\nenergy value in 2nd state is %0.4f eV",(E3
    ))
```

Scilab code Exa 4.4 `minimum energy`

```

//Variable declaration
n=1;
e=1.6*10**-19;
m=9.1*10**-31;      //mass(kg)
h=6.63*10**-34;      //planck's constant
L=4*10**-10;         //width(m)

//Calculation
E1=n**2*h**2/(8*m*e*L**2);        //energy value in g
    state(eV)

//Result
printf('minimum energy is %0.3f eV   \n',(E1))
```

Scilab code Exa 4.5 `wavelength`

```

//Variable declaration
V=15*10**3;      //voltage(V)

//Calculation
lamda=1.227/sqrt(V);       //wavelength(nm)

//Result
```

```
printf('wavelength is %0.3f  nm   \n',(lamda))
```

Scilab code Exa 4.6 `minimum energy`

```
//Variable declaration
n=1;
e=1.6*10**-19;
m=9.1*10**-31;     //mass(kg)
h=6.63*10**-34;     //planck's constant
L=0.05*10**-9;         //width(m)

//Calculation
E1=n**2*h**2/(8*m*e*L**2);        //energy value in g
    state(eV)

//Result
printf('minimum energy is %0.3f  eV   \n',(E1))
```

Scilab code Exa 4.8 `minimum energy`

```
//Variable declaration
n=1;
e=1.6*10**-19;
m=9.1*10**-31;     //mass(kg)
h=6.63*10**-34;     //planck's constant
L=3*10**-10;        //width(m)

//Calculation
E1=n**2*h**2/(8*m*e*L**2);        //energy value in g
    state(eV)

//Result
printf('minimum energy is %0.3f   eV  \n',(E1))
```

Scilab code Exa 4.9 `de broglie wavelength`

```
//Variable declaration
me=1.676*10**-27;     //mass(kg)
mn=9.1*10**-31;     //mass(kg)
h=6.63*10**-34;     //planck's constant

//Calculation
lamda_n=h/sqrt(4*mn*me);        //de broglie
    wavelength(m)

//Result
printf('de broglie wavelength is %0.3f  nm   \n',int
    (lamda_n*10**9))
```

Scilab code Exa 4.10 `energy value in states`

```
//Variable declaration
n=1;
e=1.6*10**-19;
m=9.1*10**-31;     //mass(kg)
h=6.63*10**-34;     //planck's constant
L=2*10**-10;        //width(m)

//Calculation
E1=n**2*h**2/(8*m*e*L**2);        //energy value in g
    state(eV)
E2=2**2*E1;        //energy value in 2nd quantum state
    (eV)
E4=4**2*E1;        //energy value in 2nd quantum state
    (eV)
```

```

//Result
printf('energy value in 2nd quantum state is  %0.3f
      \n',(E2))
printf('energy value in 4th quantum state is %0.3d
   eV\n ',(E4))
printf('answer varies due to approximating off
   errors\n')
```

Scilab code Exa 4.11 `interplanar spacing`

```

//Variable declaration
e=1.6*10**-19;
m=9.1*10**-31;      //mass(kg)
h=6.63*10**-34;     //planck's constant
V=344;              //potemtial(V)
n=1;
theta=60;       //angle(degrees)

//Calculation
theta=theta*%pi/180;        //angle(radian)
d=n*h/(2*sin(theta)*sqrt(2*m*V*e));       //
   interplanar spacing(m)

//Result
printf('interplanar spacing is %0.3f  angstrom   \n'
   ,(d*10**10))
```

Scilab code Exa 4.12 `energy required to pump an electron`

```

//Variable declaration
```

```
n=1;
e=1.6*10**-19;
m=9.11*10**-31;     //mass(kg)
h=6.63*10**-34;     //planck's constant
L=1*10**-10;        //width(m)

//Calculation
E1=n**2*h**2/(8*m*e*L**2);       //energy value in g
    state(eV)
E3=3**2*E1;       //energy value in 2nd quantum state
    (eV)
E=E3-E1;       //energy required to pump an electron
    (eV)

//Result
printf('energy required to pump an electron is %0.3f
    eV   \n',(E))
printf('answer varies due to approximating off
    errors\n')
```

Scilab code Exa 4.13 `minimum energy`

```

//Variable declaration
n=1;
e=1.6*10**-19;
m=9.11*10**-31;     //mass(kg)
h=6.63*10**-34;     //planck's constant
L=2*10**-10;        //width(m)

//Calculation
E1=n**2*h**2/(8*m*e*L**2);       //energy value in g
    state(eV)

//Result
```

```
printf('minimum energy is %0.3f   eV  \n',(E1))
printf('answer varies due to approximating off
    errors\n')
```

Scilab code Exa 4.14 `wavelength`

```

//Variable declaration
V=1600;     //voltage(V)

//Calculation
lamda=1.227/sqrt(V);      //wavelength(nm)

//Result
printf('wavelength is %0.3f  angstrom   \n',(lamda
    *10))
```

Chapter 5

ELECTRON THEORY OF METALS

Scilab code Exa 5.1 `temperature`

```

//Variable declaration
fE=1/100;     //probability(%)
E_EF=0.5;     //fermi energy(eV)
Kb=1.38*10**-23;     //boltzmann constant
e=6.24*10**18;       //conversion faction from J to
   eV

//Calculation
x=E_EF/(Kb*e);
y=log(1/fE);
T=x/y;          //temperature(K)

//Result
printf('temperature is %0.3f    K \n',(T))
printf('answer varies due to approximating off
   errors\n')
```

Scilab code Exa 5.2 `total number of free electrons`

```

//Variable declaration
Ef=7*1.602*10**-19;      //fermi energy(J)
h=6.63*10**-34;      //planck's constant
m=9.11*10**-31;      //mass(kg)

//Calculation
x=h**2/(8*m);
y=(3/%pi)**(2/3);
n23=Ef/(x*y);
n=n23**(3/2);        //total number of free electrons
   (per m**3)

//Result
printf('total number of free electrons is %0.3f
   **10**28 per m**3\n',(n/10**28))
printf('answer varies due to approximating off
   errors\n')
```

Scilab code Exa 5.3 `relaxation time`

```

//Variable declaration
rho=1.54*10**-8;      //resistivity of metal(ohm m)
n=5.8*10**28;      //number of free electrons(per m
   **3)
e=1.602*10**-19;      //charge(c)
m=9.11*10**-31;      //mass(kg)

//Calculation
```

```
tow=m/(n*e**2*rho);         //relaxation time(s)

//Result
printf('relaxation time is %0.3f  *10**-15 s   \n',(
   tow*10**15))
printf('answer varies due to approximating off
   errors\n')
```

Scilab code Exa 5.4 `relaxation time`

```

//Variable declaration
rho=1.43*10**-8;       //resistivity of metal(ohm m)
n=6.5*10**28;      //number of free electrons(per m
   **3)
e=1.6*10**-19;      //charge(c)
m=9.1*10**-31;      //mass(kg)

//Calculation
tow=m/(n*e**2*rho);         //relaxation time(s)

//Result
printf('relaxation time is %0.3f   *10**-14 s  \n',(
   tow*10**14))
```

Scilab code Exa 5.5 `drift velocity`

```

//Variable declaration
L=5;     //length(m)
R=0.06;     //resistance(ohm)
I=15;        //current(A)
ne=3;     //number of electrons
```

```
rho=2.7*10**-8;    //resistivity(ohm m)
w=26.98;    //atomic weight
D=2.7*10**3;    //density(kg/m**3)
Na=6.025*10**26;    //avagadro number(per k mol)
e=1.6*10**-19;
//Calculation
n=ne*Na*D/w;        //number of conduction electrons(
   per m**3)
mew=1/(n*e*rho);     //mobility of electrons(m**2/Vs)
vd=I*R/(L*rho*n*e);      //drift velocity(m/s)

//Result
printf('number of conduction electrons is %0.3f
   *10**29 per m**3   \n',(n/10**29))
printf('mobility of electrons is %0.3f   m**2/Vs  \n
   ',(mew))
printf('drift velocity is %0.3f    *10**-4 m/s\n',(
   vd*10**4))
```

Scilab code Exa 5.6 `mobility of electrons`

```

//Variable declaration
ne=1;    //number of electrons
rho=1.73*10**-8;    //resistivity(ohm m)
w=63.5;    //atomic weight
e=1.6*10**-19;     //charge(c)
D=8.92*10**3;    //density(kg/m**3)
Na=6.02*10**26;    //avagadro number(per k mol)

//Calculation
n=ne*Na*D/w;
mew=1/(n*e*rho);      //mobility of electrons(m**2/Vs
   )

```

```
//Result
printf('mobility of electrons is %0.3f   m**2/Vs  \n
   ',(mew))
printf('answer in the book is wrong\n')
```

Scilab code Exa 5.7 `mobility of electrons`

```

//Variable declaration
ne=1;     //number of electrons
rho=1.721*10**-8;    //resistivity(ohm m)
w=63.54;    //atomic weight
e=1.6*10**-19;      //charge(c)
D=8.95*10**3;    //density(kg/m**3)
Na=6.025*10**26;    //avagadro number(per k mol)

//Calculation
n=ne*Na*D/w;
mew=1/(n*e*rho);      //mobility of electrons(m**2/Vs
   )

//Result
printf('mobility of electrons is %0.3f   m**2/Vs  \n
   ',(mew))
printf('answer in the book is wrong\n')
```

Scilab code Exa 5.8 `relaxation time`

```

//Variable declaration
rho=1.5*10**-8;      //resistivity of metal(ohm m)
n=6.5*10**28;      //number of free electrons(per m
   **3)
```

```
e=1.602*10**-19;      //charge(c)
m=9.11*10**-31;      //mass(kg)

//Calculation
tow=m/(n*e**2*rho);        //relaxation time(s)

//Result
printf('relaxation time is %0.3f   *10**-14 s  \n',(
    tow*10**14))
```

Scilab code Exa 5.9 thermal velocity

```

//Variable declaration
rho=1.54*10**-8;      //resistivity of metal(ohm m)
n=5.8*10**28;      //number of free electrons(per m
    **3)
e=1.602*10**-19;      //charge(c)
m=9.11*10**-31;      //mass(kg)
E=1*10**2;    //electric field(V/m)
Kb=1.381*10**-23;     //boltzmann constant
T=300;     //temperature(K)

//Calculation
tow=m/(n*e**2*rho);        //relaxation time(s)
vd=e*E*tow/m;     //drift velocity(m/s)
mew=vd/E;         //mobility(m**2/Vs)
Vth=sqrt(3*Kb*T/m);     //thermal velocity(m/s)

//Result
printf('relaxation time is %0.3f  *10**-14 s   \n',(
    tow*10**14))
printf('drift velocity is %0.3f    m/s \n',(vd))
printf('mobility is %0.3f   *10**-2 m**2/Vs  \n',(
    mew*10**2))
```

```
printf('thermal velocity is %0.3f   *10**5 m/s  \n'
   ,(Vth/10**5))
```

Scilab code Exa 5.10 `mean free path`

```

//Variable declaration
EF=5.5*1.602*10**-19;      //fermi energy of silver(J
   )
tow=3.97*10**-14;     //relaxation time(s)
m=9.11*10**-31;      //mass(kg)

//Calculation
vf=sqrt(2*EF/m);       //fermi velocity(m/s)
lamda=vf*tow;      //mean free path(m)

//Result
printf('fermi velocity is %0.3f  *10**6 m/s   \n',(
   vf/10**6))
printf('mean free path is %0.3f   *10**-8 m  \n',(
   lamda*10**8))
```

Scilab code Exa 5.11 `fermi energy`

```

//Variable declaration
ne=1;    //number of electrons
M=107.9;    //atomic weight
D=10500;    //density(kg/m**3)
Na=6.025*10**26;    //avagadro number(per k mol)
m=9.11*10**-31;      //mass(kg)
h=6.63*10**-34;      //planck's constant

```

```
//Calculation
n=ne*Na*D/M;
x=h**2/(8*m);
y=(3/%pi)**(2/3);
Ef=x*y*n**(2/3);          //fermi energy(eV)

//Result
printf('fermi energy is %0.3f  *10**-19 J   \n',(Ef
   *10**19))
```

Scilab code Exa 5.12 `drift velocity of free electrons`

```

//Variable declaration
A=10*10**-6;       //area(m**2)
ne=1;      //number of electrons
I=100;      //current(amperes)
w=63.5;     //atomic weight
e=1.6*10**-19;       //charge(c)
D=8.92*10**3;     //density(kg/m**3)
Na=6.02*10**26;     //avagadro number(per k mol)

//Calculation
n=ne*Na*D/w;
J=I/A;
vd=J/(n*e);        //drift velocity of free electrons(
   m/s)

//Result
printf('drift velocity of free electrons is %0.3f
   *10**-3 m/s  \n',(vd*10**3))
```

Chapter 6

DIELECTRIC PROPERTIES

Scilab code Exa 6.1 dielectric constant of material

```

//Variable declaration
N=3*10**28;      //number of atoms(per m**3)
alpha_e=10**-40;     //electronic polarizability(F m
    **2)
epsilon0=8.85*10**-12;

//Calculation
epsilonr=(alpha_e*N/epsilon0)+1;       //dielectric
    constant of material

//Result
printf('dielectric constant of material is %0.3f
        \n',(epsilonr))
```

Scilab code Exa 6.2 charge on plates

```

```

```
//Variable declaration
epsilon0=8.85*10**-12;
A=100*10**-4;      //area(m**2)
d=1*10**-2;      //seperation(m)
V=100;      //potential(V)

//Calculation
C=epsilon0*A/d;     //capacitance(F)
Q=C*V;       //charge on plates(C)

//Result
printf('capacitance is %e  F   \n',C)
printf('charge on plates is %e  C  \n',Q)
```

Scilab code Exa 6.3 electronic polarizability

```

//Variable declaration
epsilon0=8.85*10**-12;
epsilonr=1.0000684;      //dielectric constant of
   material
N=2.7*10**25;     //number of atoms(per m**3)

//Calculation
alpha_e=epsilon0*(epsilonr-1)/N;      //electronic
   polarizability(F m**2)

//Result
printf('electronic polarizability is %e  F m**2   \n
   ',alpha_e)
```

Scilab code Exa 6.4 voltage

```

//Variable declaration
epsilon0=8.85*10**-12;
A=650*10**-6;       //area(m**2)
d=4*10**-3;       //seperation(m)
Q=2*10**-10;      //charge(C)
epsilonr=3.5;

//Calculation
V=Q*d/(epsilon0*epsilonr*A);        //voltage(V)

//Result
printf('voltage is %0.3f   V  \n',(V))
```

Scilab code Exa 6.5 polarization

```

//Variable declaration
epsilon0=8.85*10**-12;
A=6.45*10**-4;       //area(m**2)
d=2*10**-3;       //seperation(m)
V=12;     //voltage(V)
epsilonr=5;

//Calculation
P=epsilon0*(epsilonr-1)*V/d;      //polarization(C m)

//Result
printf('polarization is %0.3f   *10**-9 C m  \n',P
   *10**9)
```

Scilab code Exa 6.6 electronic polarizability

```
//Variable declaration
epsilon0=8.85*10**-12;
epsilonr=3.75;    //dielectric constant
gama=1/3;     //internal field constant
D=2050;    //density(kg/m**3)
Na=6.02*10**26;    //avagadro number
M=32;    //atomic weight

//Calculation
N=Na*D/M;     //number of atoms(per m**3)
alphae=((epsilonr-1)/(epsilonr+2))*3*epsilon0/N;
   //electronic polarizability(F m**2)

//Result
printf('electronic polarizability is %0.3f  *10**-40
    F m**2   \n',(alphae*10**40))
printf('answer varies due to approximating off
   errors\n')
```

Scilab code Exa 6.7 orientational polarization

```
//Variable declaration
N=1.6*10**20;      //number of molecules(/m**3)
T=300;      //temperature(K)
E=5*10**5;     //electric field(V/m)
x=0.25*10**-9;    //separation(m)
Kb=1.381*10**-23;     //boltzmann constant
e=1.6*10**-19;

//Calculation
Pd=N*e**2*x**2*E/(3*Kb*T);       //orientational
   polarization

```

```
//Result
printf('orientational polarization is %0.3f *10**-11
    C m    \n',(Pd*10**11))
```

Scilab code Exa 6.8 displacement

```

//Variable declaration
epsilon0=8.85*10**-12;
epsilonr=1.0000684;       //dielectric constant of
    material
N=2.7*10**25;     //number of atoms(per m**3)
E=10**6;      //electric field(V/m)
e=1.6*10**-19;
Z=2;      //atomic number

//Calculation
alpha_e=epsilon0*(epsilonr-1)/N;      //electronic
    polarizability(F m**2)
r=(alpha_e/(4*%pi*epsilon0))**(1/3);      //radius(m)
d=alpha_e*E/(Z*e);      //displacement(m)

//Result
printf('radius is %0.3f    *10**-11 m \n',(r*10**11)
    )
printf('answer varies due to approximating off
    errors\n')
printf('displacement is %0.3f *10**-16 m    \n',(d
    *10**16))
```

Scilab code Exa 6.9 voltage

```

```

```
//Variable declaration
epsilon0=8.85*10**-12;
A=750*10**-6;       //area(m**2)
d=5*10**-3;       //seperation(m)
Q=2.5*10**-10;      //charge(C)
epsilonr=3.5;

//Calculation
V=Q*d/(epsilon0*epsilonr*A);        //voltage(V)

//Result
printf('voltage is %0.3f   V  \n',(V))
```

Scilab code Exa 6.10 polarizability

```

//Variable declaration
N=3*10**25;     //number of atoms(per m**3)
r=0.2*10**-9;     //radius(m)
epsilon0=8.85*10**-12;
E=1;      //electric field

//Calculation
p=4*%pi*epsilon0*r**3;     //dipole moment(F m**2)
P=N*p;      //polarization(C m)
epsilonr=(P/(epsilon0*E))+1;      //dielectric
   constant
alpha_e=epsilon0*(epsilonr-1)/N;     //polarizability
   (F m**2)

//Result
printf('dipole moment is %0.3f  *10**-40 F m**2   \n
   ',(p*10**40))
printf('polarization is %0.3f   *10**-15 C m  \n',(P
   *10**15))
```

```
printf('dielectric constant is %0.3f      \n',(
   epsilonr))
printf('polarizability is %0.3f     *10**-40 F m**2 \
   n',(alpha_e*10**40))
```

Scilab code Exa 6.11 electronic polarizability

```

//Variable declaration
epsilon0=8.85*10**-12;
epsilonr=1.000435;       //dielectric constant of
   material
N=2.7*10**25;     //number of atoms(per m**3)

//Calculation
alpha_e=epsilon0*(epsilonr-1)/N;      //electronic
   polarizability(F m**2)

//Result
printf('electronic polarizability is %0.3f
   *10**-40 F m**2  \n',(alpha_e*10**40))
```

Scilab code Exa 6.12 electronic polarizability

```

//Variable declaration
epsilon0=8.85*10**-12;
epsilonr=4;     //dielectric constant
D=2.08*10**3;     //density(kg/m**3)
Na=6.02*10**26;     //avagadro number
M=32;     //atomic weight

//Calculation
```

```
N=Na*D/M;       //number of atoms(per m**3)
alphae=epsilon0*(epsilonr-1)/N;     //atomic
   polarizability(F m**2)

//Result
printf('electronic polarizability is %0.3f  *10**-40
    F m**2   \n',(alphae*10**40))
```

Chapter 7

Magnetic properties

Scilab code Exa 7.1 `magnetic moment`

```

//Variable declaration
chi=-0.4*10**-5;        //magnetic susceptibility
H=5*10**5;        //magnetic field(A/m)
mew0=4*%pi*10**-7;

//Calculation
B=mew0*H*(1+chi);       //flux density(Wb/m**2)
M=chi*H;        //magnetic moment(A/m)

//Result
printf('flux density is %0.3f    Wb/m**2 \n',(B))
printf('magnetic moment is %0.3f   A/m  \n',M)
```

Scilab code Exa 7.2 `flux density`

```

//Variable declaration
```

```
chi=-0.25*10**-5;        //magnetic susceptibility
H=1000;          //magnetic field(A/m)
mew0=4*%pi*10**-7;

//Calculation
M=chi*H;          //magnetisation(A/m)
B=mew0*(H+M);        //flux density(Wb/m**2)

//Result
printf('magnetisation is %0.3f  *10**-2 A/m   \n',M
   *10**2)
printf('flux density is %0.3f  *10**-3 Wb/m**2   \n'
   ,(B*10**3))
```

Scilab code Exa 7.3 `magnetisation and flux density`

```

//Variable declaration
H=250;          //magnetic field(A/m)
mewr=15;      //relative permeability
mew0=4*%pi*10**-7;

//Calculation
M=H*(mewr-1);        //magnetisation(A/m)
B=mew0*(H+M);        //flux density(Wb/m**2)

//Result
printf('magnetisation is %0.3f   A/m  \n',M)
printf('flux density is %0.3f    *10**-3 Wb/m**2 \n'
   ,(B*10**3))
```

Scilab code Exa 7.4 `magnetisation and flux density`

```

//Variable declaration
chi=-0.42*10**-3;        //magnetic susceptibility
H=1000;        //magnetic field(A/m)
mew0=4*%pi*10**-7;

//Calculation
M=chi*H;        //magnetisation(A/m)
B=mew0*(H+M);        //flux density(Wb/m**2)

//Result
printf('magnetisation is %0.3f     A/m \n',M)
printf('flux density is %0.3f    *10**-3 Wb/m**2 \n'
    ,(B*10**3))
```

Scilab code Exa 7.5 magnetic moment

```

//Variable declaration
r=10/2;        //radius(cm)
i=500*10**-3;        //current(A)

//Calculation
mew=%pi*(r*10**-2)**2*i;        //magnetic moment(Am**2)

//Result
printf('magnetic moment is %0.3f   *10**-3 Am**2    \n
    ',(mew*10**3))
```

Scilab code Exa 7.6 magnetizing force and relative permeability

```

//Variable declaration
```

```
mew0=4*%pi*10**-7;
B=0.0044;      //flux density(Wb/m**2)
M=3300;        //magnetic moment(A/m)

//Calculation
H=(B/mew0)-M;     //magnetizing force(A/m)
mewr=1+(M/H);     //relative permeability

//Result
printf('magnetizing force is %0.3f   A/m  \n',int(H)
    )
printf('relative permeability is %0.3f     \n',(mewr
    ))
printf('answer varies due to approximating off
    errors\n')
```

Scilab code Exa 7.7 change in magnetic moment

```

//Variable declaration
r=0.052*10**-9;      //radius(m)
B=3;      //flux density(Wb/m**2)
e=1.6*10**-19;
m=9.1*10**-31;     //mass(kg)

//Calculation
delta_mew=e**2*r**2*B/(4*m);      //change in
    magnetic moment(A m**2)

//Result
printf('change in magnetic moment is %0.3f
    *10**-29 Am**2  \n',(delta_mew*10**29))
printf('answer given in the book is wrong\n')
```

Scilab code Exa 7.8 `change in magnetic moment`

```

//Variable declaration
r=5.29*10**-11;       //radius(m)
B=2;       //flux density(Wb/m**2)
e=1.6*10**-19;
m=9.1*10**-31;     //mass(kg)

//Calculation
d_mew=e**2*r**2*B/(4*m);       //change in magnetic
   moment(A m**2)

//Result
printf('change in magnetic moment is %0.3f *10**-29
   Am**2     \n',(d_mew*10**29))
```

Scilab code Exa 7.9 `susceptibility`

```

//Variable declaration
N=10**28;       //number of atoms(per m**3)
chi1=2.8*10**-4;    //susceptibility
T1=350;     //temperature(K)
T2=300;     //temperature(K)

//Calculation
chi2=chi1*T1/T2;     //susceptibility

//Result
printf('susceptibility is %0.3f    *10**-4  \n',(chi2
   *10**4))
```

Scilab code Exa 7.10 relative permeability

```

//Variable declaration
B=1.4;      //flux density(Wb/m**2)
B0=6.5*10**-4;     //magnetic field(Tesla)

//Calculation
mewr=B/B0;     //relative permeability

//Result
printf('relative permeability is %0.3f      \n',(mewr
    ))
```

Chapter 8

Semiconductors

Scilab code Exa 8.1 resistivity

```

//Variable declaration
ni=2.5*10**19;       //intrinsic concentration(per m
    **3)
mewn=0.4;       //mobility of electrons(m**2/Vs)
mewp=0.2;      //mobility of holes(m**2/Vs)
e=1.6*10**-19;

//Calculation
sigma_i=ni*e*(mewn+mewp);
rhoi=1/sigma_i;       //resistivity(ohm m)

//Result
printf('resistivity is %0.3f   ohm m  \n',(rhoi))
```

Scilab code Exa 8.2 number of donor atoms

```

```

```
//Variable declaration
mewn=0.3;       //mobility of electrons(m**2/Vs)
rho=0.25;       //resistivity(ohm m)
e=1.6*10**-19;

//Calculation
n=1/(rho*e*mewn);     //number of donor atoms(per m
   **3)

//Result
printf('number of donor atoms is %0.3f    *10**19 per
    m**3  \n',(n/10**19))
```

Scilab code Exa 8.3 diffusion coefficient

```

//Variable declaration
mewn=0.21;       //mobility of electrons(m**2/Vs)
e=1.6*10**-19;
Kb=1.38*10**-23;      //boltzmann constant
T=300;      //temperature(K)

//Calculation
Dn=mewn*Kb*T/e;       //diffusion coefficient of
   electrons(m**2/s)

//Result
printf('diffusion coefficient of electrons is %0.3f
    *10**-4 m**2/s    \n',(Dn*10**4))
```

Scilab code Exa 8.4 mobility of holes

```

```

```
//Variable declaration
Rh=3.22*10**-4;      //hall coefficient(m**3/C)
e=1.6*10**-19;
rho=8.5*10**-3;      //resistivity(ohm m)

//Calculation
p=1/(Rh*e);        //carrier concentration(per m**3)
mewp=Rh/rho;       //mobility of holes(m**2/Vs)

//Result
printf('carrier concentration is %0.3f     *10**21
   per m**3 \n',(p/10**21))
printf('//mobility of holes is %0.3f      m**2/Vs\n'
   ,(mewp))
```

Scilab code Exa 8.5 `intrinsic concentration`

```

//Variable declaration
mewe=0.36;       //mobility of electrons(m**2/Vs)
mewh=0.17;      //mobility of holes(m**2/Vs)
e=1.6*10**-19;
rhoi=2.12;      //resistivity(ohm m)

//Calculation
ni=1/(rhoi*e*(mewe+mewh));       //intrinsic
   concentration(per m**3)

//Result
printf('intrinsic concentration is %0.3f     *10**16
   per m**3 \n',(ni/10**16))
```

Scilab code Exa 8.6 `resistivity`

```
//variable declaration
mewe=0.39;      //mobility of electrons(m**2/Vs)
mewh=0.19;     //mobility of holes(m**2/Vs)
e=1.6*10**-19;
ni=2.4*10**19;      //intrinsic concentration(per m
    **3)

//Calculation
rhoi=1/(ni*e*(mewe+mewh));          //resistivity(
    ohm m)

//Result
printf('resistivity is %0.3f  ohm m   \n',(rhoi))
```

Scilab code Exa 8.7 conductivity

```

//Variable declaration
mewe=0.135;      //mobility of electrons(m**2/Vs)
mewh=0.048;     //mobility of holes(m**2/Vs)
e=1.6*10**-19;
ni=1.5*10**16;      //intrinsic concentration(per m
    **3)
Nd=10**23;          //doping concentration(per m**3)

//Calculation
sigma=ni*e*(mewe+mewh);      //conductivity(per ohm m
    )
p=ni**2/Nd;       //hole concentration(per m**3)
sigman=Nd*e*mewe;      //conductivity(per ohm m)

//Result
printf('conductivity is %0.3f   *10**-3 per ohm m  \
    n',(sigma*10**3))
printf('hole concentration is %0.3f  *10**9 per m**3
      \n',p/10**9)
printf('conductivity is %0.3f   *10**3 per ohm m  \n
   ',sigman/10**3)
```

Scilab code Exa 8.8 carrier concentration

```

//Variable declaration
Rh=3.66*10**-4;     //hall coefficient(m**3/C)
e=1.6*10**-19;
rhoh=8.93*10**-3;     //resistivity(ohm m)

//Calculation
p=1/(Rh*e);       //carrier concentration(per m**3)
mewp=Rh/rhoh;      //mobility of holes(m**2/Vs)

//Result
printf('carrier concentration is %0.3f *10**22 per m
   **3    \n',(p/10**22))
printf('//mobility of holes is %0.3f   *10**-2 m**2/
   Vs  \n',(mewp*10**2))
```

Scilab code Exa 8.9 conductivity

```

//Variable declaration
mewe=0.13;      //mobility of electrons(m**2/Vs)
mewh=0.05;     //mobility of holes(m**2/Vs)
e=1.6*10**-19;
ni=1.5*10**16;      //intrinsic concentration(per m
   **3)

//Calculation
```

```
sigma=ni*e*(mewe+mewh);      //conductivity(per ohm m
   )

//Result
printf('conductivity is %0.3f  *10**-4 per ohm m   \
   n',sigma*10**4)
```

Scilab code Exa 8.10 conductivity

```

//Variable declaration
mewe=0.14;      //mobility of electrons(m**2/Vs)
mewh=0.05;     //mobility of holes(m**2/Vs)
e=1.6*10**-19;
ni=1.5*10**16;      //intrinsic concentration(per m
   **3)
A=28.09;      //atomic weight
D=2.33*10**3;     //density(kg/m**3)
Na=6.025*10**26;     //avagadro number

//Calculation
N=Na*D/A;     //number of atoms(per m**3)
n=N/10**8;     //electron concentration(per m**3)
p=ni**2/n;     //hole concentration(per m**3)
sigma=e*((n*mewe)+(p*mewh));      //conductivity(per
   ohm m)

//Result
printf('conductivity is %0.3f   per ohm m\n  \n',(
   sigma)  )
```

Scilab code Exa 8.11 resistivity

```
//Variable declaration
mewe=0.36;      //mobility of electrons(m**2/Vs)
mewh=0.18;     //mobility of holes(m**2/Vs)
e=1.6*10**-19;
ni=2.5*10**19;      //intrinsic concentration(per m
    **3)
N=4.2*10**28;     //avagadro number

//Calculation
n=N/10**6;     //electron concentration(per m**3)
p=ni**2/n;     //hole concentration(per m**3)
rhoi=1/(e*((n*mewe)+(p*mewh)));      //resistivity(
    per ohm m)

//Result
printf('resistivity is %0.3f   *10**-4 per ohm m  \n
    ',(rhoi*10**4))
```

Scilab code Exa 8.12 `hole concentration`

```
//Variable declaration
np=2.4*10**9;     //carrier concentration(per m**3)
N=4.2*10**28;     //avagadro number

//Calculation
p=np/2;      //hole concentration(per m**3)

//Result
printf('hole concentration is %0.3f  *10**9 per m**3
      \n',p/10**9)
```

Scilab code Exa 8.13 density of donor atoms

```

//Variable declaration
mewn=0.35;      //mobility of electrons(m**2/Vs)
e=1.602*10**-19;
rho=0.2;     //resistivity(ohm m)

//Calculation
n=1/(rho*e*mewn);       //density of donor atoms

//Result
printf('density of donor atoms is %0.3f  *10**19
   electrons/m**3   \n',(n/10**19))
```

Scilab code Exa 8.14 energy gap

```

//Variable declaration
Kb=1.38*10**-23;     //boltzmann constant
T1=300;     //temperature(K)
T2=320;     //temperature(K)
rho1=5;     //resistivity(ohm m)
rho2=2.5;     //resistivity(ohm m)
e=1.6*10**-19;
//Calculation
Eg=2*Kb*log(rho1/rho2)/((1/T1)-(1/T2));      //energy
    gap(J)

//Result
printf('energy gap is %0.3f   eV  \n',(Eg/e))
```

Scilab code Exa 8.15 diffusion coefficient

```

//Variable declaration
Kb=1.38*10**-23;      //boltzmann constant
T=300;      //temperature(K)
mewe=0.19;       //mobility of electrons(m**2/Vs)
e=1.6*10**-19;

//Calculation
Dn=mewe*Kb*T/e;        //diffusion coefficient(m**2/
    sec)

//Result
printf('diffusion coefficient is %0.3f  *10**-3 m
    **2/sec   \n',(Dn*10**3))
```

Scilab code Exa 8.16 energy gap

```

//Variable declaration
Kb=1.38*10**-23;      //boltzmann constant
T1=293;      //temperature(K)
T2=305;      //temperature(K)
rho1=4.5;      //resistivity(ohm m)
rho2=2.0;      //resistivity(ohm m)
e=1.6*10**-19;
//Calculation
Eg=2*Kb*log(rho1/rho2)/((1/T1)-(1/T2));      //energy
    gap(J)

//Result
printf('energy gap is %0.3f  eV   \n',(Eg/e))
```

Chapter 9

Superconductivity

Scilab code Exa 9.1 `transition temperature`

```

//Variable declaration
T=8;     //temperature(K)
Hc=1*10**5;    //critical field(amp/m)
H0=2*10**5;    //critical field(amp/m)

//Calculation
Tc=T/sqrt(1-(Hc/H0));      //transition temperature(K
   )

//Result
printf('transition temperature is %0.3f   K  \n',(Tc
   ))
```

Scilab code Exa 9.2 `frequency`

```

//Variable declaration
```

```
h=6.626*10**-34;      //plancks constant
e=1.6*10**-19;
V=8.5*10**-6;      //voltage(V)

//Calculation
new=2*e*V/h;       //frequency(Hz)

//Result
printf('frequency is %0.3f    *10**9 Hz  \n',(new
   /10**9))
```

Scilab code Exa 9.3 `critical field`

```

//Variable declaration
T=2;      //temperature(K)
H0=0.0306;     //critical field(amp/m)
Tc=3.7;       //transition temperature(K)

//Calculation
Hc=H0*(1-(T/Tc)**2);     //critical field(Tesla)

//Result
printf('critical field is %0.3f     Tesla \n',(Hc))
```

Scilab code Exa 9.4 `temperature`

```

//Variable declaration
H0=250*10**3;     //critical field(amp/m)
Tc=12;       //transition temperature(K)
Hc=200*10**3;     //critical field(Tesla)

```

```
//Calculation
T=Tc*sqrt(1-(Hc/H0)**2);      //temperature(K)

//Result
printf('temperature is %0.3f    K   \n',(T))
```

Scilab code Exa 9.5 critical field

```

//Variable declaration
T=2.5;       //temperature(K)
H0=0.03;     //critical field(amp/m)
Tc=3.7;       //transition temperature(K)

//Calculation
Hc=H0*(1-(T/Tc)**2);    //critical field(Tesla)

//Result
printf('critical field is %0.3f     Tesla \n',(Hc))
```

Scilab code Exa 9.6 frequency

```

//Variable declaration
h=6.625*10**-34;      //plancks constant
e=1.6*10**-19;
V=650*10**-6;       //voltage(V)

//Calculation
new=2*e*V/h;        //frequency(Hz)

//Result
```

```
printf('frequency is %0.3f *10**9 Hz \n',(new
   /10**9))
```

Scilab code Exa 9.7 `critical field`

```

//Variable declaration
T=5;      //temperature(K)
H0=6.5*10**3;    //critical field(amp/m)
Tc=7.2;       //transition temperature(K)

//Calculation
Hc=H0*(1-(T/Tc)**2);    //critical field(Tesla)

//Result
printf('critical field is %0.3f *10**3 A/m \n',(
   Hc/10**3))
```

Chapter 10

Lasers

Scilab code Exa 10.1 `energy gap`

```

//Variable declaration
h=6.63*10**-34;        //planck's constant
c=3*10**8;        //velocity of light(m/s)
lamda=1.55*10**-6;        //wavelength(m)
e=1.6*10**-19;

//Calculation
Eg=h*c/(lamda*e);        //energy gap(eV)

//Result
printf('energy gap is %0.3f    eV   \n',(Eg))
```

Scilab code Exa 10.2 `wavelength`

```

//Variable declaration
h=6.63*10**-34;        //planck's constant
```

```
c=3*10**8;        //velocity of light(m/s)
Eg=1.44;      //energy gap(eV)
e=1.6*10**-19;

//Calculation
lamda=h*c/(Eg*e);     //wavelength(m)

//Result
printf('wavelength is %0.3f    angstrom   \n',(lamda
   *10**10))
```

Chapter 11

Fibre Optics

Scilab code Exa 11.1 `fractional refractive indices change`

```

//Variable declaration
n1=1.48;        //refractive index of core
n2=1.45;        //refractive index of cladding

//Calculation
NA=sqrt((n1**2)-(n2**2));      //numerical aperture
theta0=asin(NA);      //acceptance angle(radian)
theta0=theta0*180/%pi;     //acceptance angle(degrees
   )
theta0_m=60*(theta0-int(theta0));
thetac=asin(n2/n1);      //critical angle(radian)
thetac=thetac*180/%pi;     //critical angle(degrees)
thetac_m=60*(thetac-int(thetac));
delta=(n1-n2)/n1;       //fractional refractive
   indices change

//Result
printf('numerical aperture is %0.3f      \n',(NA))
printf('acceptance angle is %0.3f   degrees %0.3f
   minutes \n',int(theta0),(theta0_m))
```

```
printf('critical angle is %0.3f   degrees %0.3f
   minutes  \n',int(thetac),int(thetac_m))
printf('fractional refractive indices change is %0.3
   f     \n',(delta))
```

Scilab code Exa 11.2 Acceptance angle

```

//Variable declaration
n1=1.563;       //refractive index of core
n2=1.498;       //refractive index of cladding

//Calculation
NA=sqrt((n1**2)-(n2**2));      //numerical aperture
theta0=asin(NA);      //acceptance angle(radian)
theta0=theta0*180/%pi;     //acceptance angle(degrees
   )
theta0_m=60*(theta0-int(theta0));

//Result
printf('numerical aperture is %0.3f     \n',(NA))
printf('acceptance angle is %0.3f   degrees %0.3f
   minutes\n',int(theta0),(theta0_m))
printf('answer varies due to approximating off
   errors\n')
```

Scilab code Exa 11.3 fractional refractive indices change

```

//Variable declaration
n1=1.563;       //refractive index of core
n2=1.498;       //refractive index of cladding

```

```
//Calculation
delta=(n1-n2)/n1;       //fractional refractive
   indices change

//Result
printf('fractional refractive indices change is %0.3
   f      \n',(delta))
```

Scilab code Exa 11.4 `Numerical aperture`

```

//Variable declaration
n1=1.55;       //refractive index of core
n2=1.50;       //refractive index of cladding

//Calculation
NA=sqrt((n1**2)-(n2**2));      //numerical aperture

//Result
printf('numerical aperture is %0.3f      \n',(NA))
```

Scilab code Exa 11.5 `refractive index`

```

//Variable declaration
NA=0.39;      //numerical aperture
n1_n2=0.05;     //difference in refractive indices

//Calculation
x=NA**2/n1_n2;
n2=(x-n1_n2)/2;      //refractive index of cladding
n1=n2+n1_n2;       //refractive index of core

```

```
//Result
printf('refractive index of core is %0.3f      \n',n1
   )
printf('refractive index of cladding is %0.3f      \n
   ',n2)
```

Scilab code Exa 11.6 Numerical aperture

```

//Variable declaration
n1=1.55;       //refractive index of core
n2=1.50;       //refractive index of cladding

//Calculation
NA=sqrt((n1**2)-(n2**2));      //numerical aperture

//Result
printf('numerical aperture is %0.3f      \n',(NA))
```

Scilab code Exa 11.7 Acceptance angle

```

//Variable declaration
n1=1.48;       //refractive index of core
n2=1.45;       //refractive index of cladding

//Calculation
NA=sqrt((n1**2)-(n2**2));      //numerical aperture
theta0=asin(NA);      //acceptance angle(radian)
theta0=theta0*180/%pi;     //acceptance angle(degrees
   )
theta0_m=60*(theta0-int(theta0));

```

```
//Result
printf('numerical aperture is %0.3f    \n',(NA))
printf('acceptance angle is %0.3f    degrees %0.3f
   minutes \n',int(theta0),(theta0_m))
```

Scilab code Exa 11.8 `refractive index`

```

//Variable declaration
NA=0.33;      //numerical aperture
delta=0.02;       //fractional refractive indices
   change

//Calculation
x=1-delta
y=sqrt(1-x**2);
n1=NA/y;        //refractive index of core
n2=n1*x;        //refractive index of cladding

//Result
printf('refractive index of core is %0.3f    \n',(
   n1))
printf('refractive index of cladding is %0.3f    \n
   ',(n2))
```

Scilab code Exa 11.9 `Acceptance angle`

```

//Variable declaration
NA=0.20;      //numerical aperture
n2=1.59;      //refractive index of cladding
n0=1.33;      //refractive index of water

```

```
//Calculation
n1=sqrt(NA**2+n2**2);        //refractive index of
   core
theta0=asin(NA/n0);        //acceptance angle(radian)
theta0=theta0*180/%pi;     //acceptance angle(degrees
   )
theta0_m=60*(theta0-int(theta0));
theta0_s=60*(theta0_m-int(theta0_m));

//Result
printf('acceptance angle is %0.3f   degrees %0.3f
   minutes %0.3f seconds  \n',int(theta0),int(
   theta0_m),(theta0_s))
printf('answer varies due to approximating off
   errors\n')
```

Scilab code Exa 11.10 fractional refractive indices change

```

//Variable declaration
n1=1.45;       //refractive index of core
n2=1.44;       //refractive index of cladding

//Calculation
delta=(n1-n2)/n1;        //fractional refractive
   indices change

//Result
printf('fractional refractive indices change is %0.3
   f  *10**-3   \n',(delta*10**3))
```

Scilab code Exa 11.11 Critical angle

```

//Variable declaration
n1=1.50;        //refractive index of core
delta=4/100;     //fractional refractive indices
   change

//Calculation
n2=n1-(n1*delta);        //refractive index of
   cladding
NA=sqrt((n1**2)-(n2**2));       //numerical aperture
theta0=asin(NA);       //acceptance angle(radian)
theta0=theta0*180/%pi;     //acceptance angle(degrees
   )
theta0_m=60*(theta0-int(theta0));
thetac=asin(n2/n1);       //critical angle(radian)
thetac=thetac*180/%pi;     //critical angle(degrees)
thetac_m=60*(thetac-int(thetac));

//Result
printf('refractive index of cladding is %0.3f      \n
   ',n2)
printf('numerical aperture is %0.3f      \n',(NA))
printf('acceptance angle is %0.3f  degrees %0.3f
   minutes  \n',int(theta0),int(theta0_m))
printf('critical angle is %0.3f    degrees %0.3f
   minutes\n',int(thetac),int(thetac_m))
```

Scilab code Exa 11.12 Numerical aperture

```

//Variable declaration
n1=1.563;       //refractive index of core
n2=1.498;       //refractive index of cladding

//Calculation
```

```
NA=sqrt((n1**2)-(n2**2));      //numerical aperture
theta0=asin(NA);      //acceptance angle(radian)
theta0=theta0*180/%pi;     //acceptance angle(degrees
    )
theta0_m=60*(theta0-int(theta0));

//Result
printf('numerical aperture is %0.3f      \n',(NA))
printf('acceptance angle is %0.3f  degrees %0.3f
    minutes  \n',int(theta0),(theta0_m))
printf('answer varies due to approximating off
    errors\n')
```

Chapter 14

Optics

Scilab code Exa 14.1 `ratio of maximum intensity to minimum intensity`

```

//Variable declaration
I1=10;      //intensity(w/m**2)
I2=25;      //intensity(w/m**2)

//Calculation
a1bya2=sqrt(I1/I2);
I=((1+a1bya2)**2)/((a1bya2-1)**2);     //ratio of
    maximum intensity to minimum intensity

//Result
printf('ratio of maximum intensity to minimum
    intensity is %0.3f     \n',(I))
printf('answer varies due to approximating off
    errors\n')
```

Scilab code Exa 14.2 `angular position of 1st minimum`

```

//Variable declaration
lamda=5460*10**-10;     //wavelength(m)
d=1*10**-4;      //seperation(m)
D=2;       //distance(m)
n=10;      //position

//Calculation
Xmax10=n*lamda*D/d;
tan_phi=Xmax10/D;
phi_max10=atan(tan_phi);
phi_max10=phi_max10*180/%pi;      //angular position
   of 10th maximum(degrees)
phim=60*(phi_max10-int(phi_max10));
phis=60*(phim-int(phim));
xmin1=lamda*D/(2*d);
tan_phi1=xmin1/D;
phi_min1=atan(tan_phi1);
phi_min1=phi_min1*180/%pi;      //angular position of
    1st minimum(degrees)
phi_m=60*(phi_min1-int(phi_min1));
phi_s=60*(phi_m-int(phi_m));

//Result
printf('angular position of 10th maximum is %0.3f
   degrees %0.3f  minutes %0.3f  seconds \n',int(
   phi_max10),int(phim),(phis))
printf('answer varies due to approximating off
   errors\n')
printf('angular position of 1st minimum is %0.3f
   degrees %0.3f  minutes %0.3f seconds \n',int(
   phi_min1),int(phi_m),int(phi_s))
```

Scilab code Exa 14.3 `visible region`

```

//Variable declaration
mew=1.33;       //refractive index of soap
t=5000*10**-10;     //thickness(m)
n0=0;
n1=1;
n2=2;
n3=3;

//Calculation
x=4*mew*t;
lamda1=x/((2*n0)+1);        //for n=0
lamda2=x/((2*n1)+1);        //for n=1
lamda3=x/((2*n2)+1);        //for n=2
lamda4=x/((2*n3)+1);        //for n=3

//Result
printf('%0.3f angstrom lies in the visible region',
   lamda3*10**10)
```

Scilab code Exa 14.4 wavelength

```

//Variable declaration
D15=0.59*10**-2;      //diameter of 15th ring(m)
D5=0.336*10**-2;      //diameter of 5th ring(m)
R=1;     //radius(m)
m=10;

//Calculation
lamda=((D15**2)-(D5**2))/(4*m*R);      //wavelength
   of light(m)

//Result
printf('wavelength of light is %0.3f   angstrom  \n'
    ,int(lamda*10**10))
```

Scilab code Exa 14.5 `radius of curvature`

```

//Variable declaration
D10=0.5*10**-2;        //diameter of 10th ring(m)
lamda=5900*10**-10;      //wavelength of light(m)
n=10;

//Calculation
R=D10**2/(4*n*lamda);       //radius of curvature(m)

//Result
printf('radius of curvature is %0.3f   m  \n',(R))
```

Scilab code Exa 14.6 `least distance of the point`

```

//Variable declaration
lamda1=650*10**-9;        //wavelength(m)
lamda2=500*10**-9;        //wavelength(m)
D=1;      //distance(m)
d=0.5*10**-3;       //seperation(m)
n=10;

//Calculation
x=n*lamda1*D/d;      //least distance of the point(m)

//Result
printf('least distance of the point is %0.3f   mm  \
    n',int(x*10**3))
```

Scilab code Exa 14.7 thickness

```

//Variable declaration
lamda=500*10**-9;      //wavelength(m)
n=10;
D10=2*10**-3;       //diameter(m)

//Calculation
r10=D10/2;     //radius(m)
R=D10**2/(4*n*lamda);
t=r10**2/(2*R);         //thickness(m)

//Result
printf('thickness is %0.3f     micro m \n',t*10**6)
```

Scilab code Exa 14.8 fringe width

```

//Variable declaration
d=0.2*10**-3;       //seperation(m)
lamda=550*10**-9;      //wavelength(m)
D=1;      //diameter(m)

//Calculation
beta=lamda*D/d;       //fringe width(m)

//Result
printf('fringe width is %0.3f   mm  \n',beta*10**3)
```

Scilab code Exa 14.9 `separation between slits`

```

//Variable declaration
lamda=500*10**-9;     //wavelength(m)
D=2;      //diameter(m)
beta=(5/100)*10**-2;      //fringe width(m)

//Calculation
d=lamda*D/beta;       //separation between slits(m)

//Result
printf('separation between slits is %0.3f   mm  \n',
   int(d*10**3))
```

Scilab code Exa 14.10 `ratio of maximum intensity to minimum intensity`

```

//Variable declaration
a12=36;       //intensity 1
a22=1;        //intensity 2

//Calculation
a1=sqrt(a12);
a2=sqrt(a22);
Imin=(a1-a2)**2;      //minimum intensity
Imax=(a1+a2)**2;      //maximum intensity
r=Imax/Imin;

//Result
printf('ratio of maximum intensity to minimum
   intensity is %0.3f     \n',(r))
```

Scilab code Exa 14.11 `diameter of 25th ring`

```

//Variable declaration
D5=0.3;        //diameter of 5th ring(cm)
D15=0.62;      //diameter of 15th ring(cm)

//Calculation
D_25=2*(D15**2)-(D5**2);
D25=sqrt(D_25);        //diameter of 25th ring(cm)

//Result
printf('diameter of 25th ring is %0.3f    cm \n',(
   D25))
```