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

Created by
Ebby
Mathematics
Mathematics
Cusat
College Teacher
None
Cross-Checked by
None

April 12, 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:** Applied Physics

**Author:** S. M. Naidu

**Publisher:** Pearson

**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

# Chapter 1

# bonding in solids

**Scilab code Exa 1.1** `bond energy of molecule`

```
clear
//
//
//

//Variable declaration
e=1.6*10**-19       //charge(coulomb)
epsilon0=8.85*10**-12
r0=236*10**-12      //equilibrium distance(m)
I=5.14              //ionisation energy(eV)
EA=-3.65            //electron affinity(eV)

//Calculation
V=-(e**2)/(4*e*%pi*epsilon0*r0)    //potential(eV)
BE=I+EA+V           //bond energy of molecule(eV)

//Result
```

**Scilab code Exa 1.3** `fractional index change`

```
clear
//
//
//

//Variable declaration
e=1.602*10**-19       //charge(coulomb)
epsilon0=8.85*10**-12
r0=0.281*10**-9       //equilibrium distance(m)
alphaM=1.748          //madelung constant
n=9                   //born constant

//Calculation
CE=-alphaM*e**2*((n-1)/n)/(4*e*%pi*epsilon0*r0)
   //cohesive energy per molecule(eV)

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

---

**Scilab code Exa 1.4** numerical aperture

```
clear
//
//
//

//Variable declaration
e=1.6*10**-19       //charge(coulomb)
epsilon0=8.85*10**-12
r0=2.5*10**-10      //equilibrium distance(m)

//Calculation
PE=e**2/(4*e*%pi*epsilon0*r0)

```

```
//Result
```

**Scilab code Exa 1.5** cohesive energy of NaCl

```
clear
//
//
//

//Variable declaration
e=1.6*10**-19        //charge(coulomb)
r0=0.281*10**-9      //equilibrium distance(m)
a=1.748*10**-28      //madelung constant(J m**2)
n=9                  //repulsive exponent value
m=1

//Calculations
Ur0=-a*(1-m/n)/(e*r0**m)       //cohesive energy of
   NaCl(eV)

//Result
```

**Scilab code Exa 1.6** cohesive energy of molecule

```
clear
//
//
//

//Variable declaration
e=1.6*10**-19        //charge(coulomb)
epsilon0=8.85*10**-12
r0=0.281*10**-9      //equilibrium distance(m)
```

```
I=5.14                  //ionisation energy(eV)
EA=-3.61                //electron affinity(eV)

//Calculation
V=-(e**2)/(4*e*%pi*epsilon0*r0)    //potential(eV)
CE=I+EA+V               //cohesive energy of molecule(eV)

//Result
printf("\n cohesive energy of molecule is %0.2f  eV"
   ,CE)
```

# Chapter 2

# crystal structures

**Scilab code Exa 2.1** `free volume per unit cell`

```
clear
//
//
//

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

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

//Result
```

**Scilab code Exa 2.2** `lattice constant`

```
clear
//
//
//

//Variable declaration
N=6.02*10**26            //Avagadro Number
n=2
rho=530     //density(kg/m**3)
M=6.94      //atomic weight(amu)

//Calculation
a=(n*M/(rho*N))**(1/3)*10**10     //lattice constant(
    angstrom)

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

**Scilab code Exa 2.3** `lattice constant`

```
clear
//
//
//

//Variable declaration
N=6.02*10**23            //Avagadro Number
n=2
rho=7860     //density(kg/m**3)
M=55.85     //atomic weight(amu)

//Calculation
a=(n*M/(rho*N))**(1/3)*10**9     //lattice constant(
    angstrom)

//Result
printf("\n lattice constant is %0.2f  angstrom",a)
```

**Scilab code Exa 2.5** `number of atoms per sq mm`

```
clear
//
//
//

//Variable declaration
a=3.5          //lattice constant(angstrom)
n=10**7        //1mm in angstrom

//Calculation
N=n**2/a**2   //number of atoms per sq mm

//Result
printf("\n number of atoms per sq mm is %0.2f
    *10**12",N/10**12)
```

**Scilab code Exa 2.6** `density`

```
clear
//
//
//

//Variable declaration
N=6.02*10**26        //Avagadro Number
n=8                  //number of atoms
```

```
a=5.62*10**-10     //lattice constant(m)
M=72.59            //atomic weight(amu)

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

//Result
```

# Chapter 3

# fects in solids

**Scilab code Exa 3.1** glancing angle

```
clear
//
//
//

//Variable declaration
a=0.28          //lattice spacing(nm)
lamda=0.071     //wavelength of X-rays(nm)
h=1
k=1
l=0
n=2

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

//Result
printf("\n glancing angle is %0.0f  degrees",theta)
```

**Scilab code Exa 3.2** `maximum order of diffraction`

```
clear
//
//
//

//Variable declaration
d=0.282         //lattice spacing(nm)
theta=(8+(35/60))*%pi/180      //glancing angle(
   radian)
n=1             //order

//Calculation
lamda=2*d*sin(theta)/n     //wavelength of X-rays(nm)
n=2*d/lamda                        //maximum order of
   diffraction

//Result
```

**Scilab code Exa 3.5** `lattice parameter`

```
clear
//
//
//

//Variable declaration
n=1      //order
theta=38.2*%pi/180      //glancing angle(radian)
lamda=1.54                  //wavelength(angstrom)
```

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

//Calculation
a=sqrt(h**2+k**2+l**2)
d=n*lamda*a/(2*sin(theta))        //lattice parameter
    (angstrom)

//Result
```

**Scilab code Exa 3.6** `maximum order of diffraction`

```
clear
//
//
//

//Variable declaration
d=1.6        //lattice spacing(angstrom)
theta=90*%pi/180      //glancing angle(radian)
lamda=1.5     //wavelength of X-rays(angstrom)

//Calculation
n=2*d*sin(theta)/lamda        //maximum order of
    diffraction

//Result
```

**Scilab code Exa 3.7** `radius of atom`

```
clear
//
```

```
//
//

//Variable declaration
d=0.203*10**-9        //lattice spacing(m)
h=1
k=1
l=0            //miller indices of (110)
lamda=1.5     //wavelength of X-rays(angstrom)

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

//Result
printf("\n length is %0.3f  *10**-9 m",a*10**9)
printf("\n volume of unit cell is %0.5f  *10**-27 m
   **3",V*10**27)
printf("\n answer for volume given in the book
   varies due to rounding off errors")
printf("\n radius of atom is %0.4f  *10**-9 m",r
   *10**9)
```

**Scilab code Exa 3.8** `maximum order of diffraction`

```
clear
//
//
//

//Variable declaration
d=1.6        //lattice spacing(angstrom)
theta=90*%pi/180      //glancing angle(radian)
lamda=1.5     //wavelength of X-rays(angstrom)
```

```

//Calculation
n=2*d*sin(theta)/lamda        //maximum order of
    diffraction

//Result
```

**Scilab code Exa 3.9** glancing angle

```
clear
//
//
//

//Variable declaration
a=0.26          //lattice spacing(nm)
lamda=0.065     //wavelength of X-rays(nm)
h=1
k=1
l=0
n=2

//Calculation
d=a/sqrt(h**2+k**2+l**2)
sintheta=n*lamda/(2*d)
theta=asin(sintheta)*180/%pi            //glancing
    angle(degrees)
thetad=int(theta)                               //
    glancing angle(degrees)
thetam=(theta-thetad)*60                        //
    glancing angle(minutes)
thetas=60*(thetam-int(thetam))                  //
    glancing angle(seconds)

//Result
```

```
printf("\n glancing angle is %0.3f degrees %0.3f
   minutes %0.3f seconds",thetad,thetam,thetas)
printf("\n answer in the book is wrong")
```

**Scilab code Exa 3.10** `cube edge of unit cell`

```
clear
//
//
//

//Variable declaration
n=1      //order
theta=19.2*%pi/180      //glancing angle(radian)
lamda=1.54                  //wavelength(angstrom)
h=1
k=1
l=1

//Calculation
d=n*lamda/(2*sin(theta))        //lattice parameter(
   angstrom)
a=d*sqrt(h**2+k**2+l**2)        //cube edge of unit
   cell(angstrom)

//Result
```

**Scilab code Exa 3.11** `lattice parameter`

```
clear
//
//
//
```

```

//Variable declaration
n=1       //order
theta=38.2*%pi/180      //glancing angle(radian)
lamda=1.54                    //wavelength(angstrom)
h=2
k=2
l=0

//Calculation
d=n*lamda/(2*sin(theta))        //lattice parameter(
   angstrom)
a=d*sqrt(h**2+k**2+l**2)        //lattice parameter(
   angstrom)

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

**Scilab code Exa 3.12** interplanar spacing

```
clear
//
//
//

//Variable declaration
a=0.36       //cube edge of unit cell(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
```

**Scilab code Exa 3.13** `glancing angle for 3rd order`

```
clear
//
//
//

//Variable declaration
theta=(5+(25/60))*%pi/180      //glancing angle(
    radian)
lamda=0.675    //wavelength of X-rays(angstrom)
n1=1                    //order
n3=3                    //order

//Calculation
d=n1*lamda/(2*sin(theta))        //lattice spacing(
    angstrom)
d=(d)

theta3=asin(n3*lamda/(2*d))*180/%pi      //glancing
    angle for 3rd order(degrees)
theta3d=int(theta3)                             //
    glancing angle for 3rd order(degrees)
theta3m=(theta3-theta3d)*60                     //
    glancing angle for 3rd order(minutes)

//Result
printf("\n lattice spacing is %0.3f  angstrom",d)
printf("\n glancing angle for 3rd order is %0.3f
    degrees %0.1f minutes",theta3d,theta3m)
printf("\n answer for minutes given in the book
    varies due to rounding off errors")
```

**Scilab code Exa 3.14** glancing angle

```
clear
//
//
//

//Variable declaration
d=3.04          //interplanar spacing(angstrom)
lamda=0.79     //wavelength of X-rays(angstrom)
n=3

//Calculation
sintheta=n*lamda/(2*d)
theta=(5+(25/60))*%pi/180;      //glancing angle(
    radian)
thetad=asin(sintheta)*180/%pi            //glancing
    angle(degrees)
thetam=(theta-int(theta))*60
    //glancing angle(minutes)
thetas=60*(thetam-int(thetam))                  //
    glancing angle(seconds)

//Result
printf("\n glancing angle is %0.0f degrees %0.3f
    minutes %0.3f seconds",thetad,thetam,thetas)
printf("\n answer given in the book is wrong")
```

# Chapter 4

# elements of statistical mechanics and principles of quantum mechanics

**Scilab code Exa 4.1** `average energy of oscillator`

```
clear
//
//
//

//Variable declaration
new=5.6*10**12      //frequency(Hz)
h=6.625*10**-34     //plank constant
kB=1.38*10**-23     //boltzmann constant
T=330               //temperature(K)

//Calculation
x=h*new/(kB*T)
E=h*new/(exp(x)-1)       //average energy of
   oscillator(joule)

//Result
```

```
printf("\n average energy of oscillator is %0.3f
    *10**-21 joule",E*10**21)
printf("\n answer given in the book varies due to
    rounding off errors")
```

**Scilab code Exa 4.2** energy density per unit wavelength

```
clear
//
//
//

//Variable declaration
h=6.63*10**-34    //plank constant
kB=1.38*10**-23   //boltzmann constant
T=1500            //temperature(K)
c=3*10**8         //velocity of light(m/sec)
lamda=6000*10**-10    //wavelength(m)

//Calculation
new=c/lamda
x=h*new/(kB*T)
y=exp(x)-1      //average energy of oscillator(joule)
Ulamda=8*%pi*h*new/(y*lamda**4)     //energy density
    per unit wavelength(Jm-4)

//Result
```

**Scilab code Exa 4.4** kinetic energy

```
clear
//
//
```

```
//

//Variable declaration
lamda=1.66*10**-10     //wavelength(m)
m=9.1*10**-31     //mass(kg)
e=1.6*10**-19     //charge(c)
h=6.63*10**-34    //plank constant

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

//Result
```

**Scilab code Exa 4.5** `energy of second excited state`

```
clear
//
//
//

//Variable declaration
L=1*10**-10     //length(m)
n2=2
n3=3
m=9.1*10**-31     //mass(kg)
e=1.6*10**-19     //charge(c)
h=6.63*10**-34    //plank constant

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

E2=n2**2*E1       //energy of 1st excited state(eV)
E3=n3**2*E1       //energy of 2nd excited state(eV)

```

```
//Result
printf("\n ground state energy is %0.4f  eV",E1)
printf("\n energy of 1st excited state is %0.2f  eV"
   ,E2)
printf("\n energy of 2nd excited state is %0.4f  eV"
   ,E3)
```

**Scilab code Exa 4.6** minimum energy

```
clear
//
//
//

//Variable declaration
L=4*10**-10     //length(m)
m=9.1*10**-31     //mass(kg)
e=1.6*10**-19     //charge(c)
h=6.63*10**-34    //plank constant

//Calculation
E1=h**2/(8*m*e*L**2)     //minimum energy(eV)

//Result
```

**Scilab code Exa 4.7** wavelength of electron waves

```
clear
//
//
//

//Variable declaration
```

```
V=15*10**3      //accelerated voltage(V)

//Calculation
lamda=1.227/sqrt(V)     //wavelength of electron
   waves(nm)

//Result
printf("\n wavelength of electron waves is %0.2f  nm
   ",lamda)
```

**Scilab code Exa 4.9** `minimum energy`

```
clear
//
//
//

//Variable declaration
L=3*10**-10      //length(m)
m=9.1*10**-31      //mass(kg)
e=1.6*10**-19      //charge(c)
h=6.63*10**-34     //plank constant

//Calculation
E1=h**2/(8*m*e*L**2)     //minimum energy(eV)

//Result
printf("\n minimum energy is %0.1f  eV",E1)
```

**Scilab code Exa 4.10** `de broglie wavelength`

```
clear
//
```

```
//
//

//Variable declaration
me=9.1*10**-31     //mass(kg)
h=6.63*10**-34    //plank constant
mn=1.676*10**-27     //mass(kg)

//Calculation
lamdan=h*10**9/sqrt(4*mn*me)      //de broglie
   wavelength(nm)

//Result
```

**Scilab code Exa 4.11** energy of 2nd excited state

```
clear
//
//
//

//Variable declaration
L=2*10**-10     //length(m)
n2=2
n4=4
m=9.1*10**-31     //mass(kg)
e=1.6*10**-19     //charge(c)
h=6.63*10**-34    //plank constant

//Calculation
E1=h**2/(8*m*e*L**2)     //minimum energy(eV)
E2=n2**2*E1       //energy of 1st excited state(eV)
E4=n4**2*E1       //energy of 2nd excited state(eV)

//Result
```

```
printf("\n ground state energy is %0.2f  eV",E1)
printf("\n energy of 1st excited state is %0.3f  eV"
    ,E2)
printf("\n energy of 2nd excited state is %0.2f  eV"
    ,E4)
printf("\n answers for energy of 1st and 2nd states
    given in the book are wrong")
```

**Scilab code Exa 4.13** `energy required to pump an electron`

```
clear
//
//
//

//Variable declaration
L=1*10**-10     //length(m)
n3=3
m=9.11*10**-31     //mass(kg)
e=1.6*10**-19     //charge(c)
h=6.63*10**-34    //plank constant

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

E3=n3**2*E1        //energy of 2nd excited state(eV)
E=E3-E1            //energy required to pump an
    electron(eV)

//Result
printf("\n ground state energy is %0.3f  eV",E1)
printf("\n energy of 2nd excited state is %0.2f  eV"
    ,E3)
printf("\n energy required to pump an electron is %0
    .2f  eV",E)
```

```
printf("\n answer given in the book is wrong")
```

**Scilab code Exa 4.15** `wavelength of electron waves`

```
clear
//
//
//

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

//Calculation
lamda=1.227*10/sqrt(V)     //wavelength of electron
   waves(angstrom)

//Result
```

# Chapter 5

# electron theory of metals

**Scilab code Exa 5.1** temperature

```
clear
//
//
//

//Variable declaration
E_EF=0.5      //fermi energy(eV)
FE=1/100      //probability
Kb=1.381*10**-23      //boltzmann constant(J/k)
x=6.24*10**18

//Calculation
KB=Kb*x
y=E_EF/KB
T=y/log(1/FE)       //temperature(K)

//Result
```

**Scilab code Exa 5.2** total number of free electrons

```
clear
//
//
//

//Variable declaration
e=1.602*10**-19        //charge(c)
m=9.11*10**-31         //mass(kg)
h=6.63*10**-34         //plancks constant(Js)
Ef=7*e                 //fermi energy(J)

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

//Result
printf("\n total number of free electrons is %0.4f
   *10**28 electrons/m**3",n/10**28)
printf("\n answer in the book varies due to rounding
    off errors")
```

**Scilab code Exa 5.3** relaxation time

```
clear
//
//
//

//Variable declaration
rho=1.54*10**-8        //resistivity(ohm m)
n=5.8*10**28           //number of electrons
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("\n relaxation time is %0.3f  *10**-15 s",tow
   *10**15)
printf("\n answer in the book varies due to rounding
    off errors")
```

**Scilab code Exa 5.5** `drift velocity`

```
clear
//
//
//

//Variable declaration
D=2.7*10**3          //density(kg/m**3)
rho=2.7*10**-8       //resistivity(ohm m)
w=26.98              //atomic weight
Na=6.025*10**26      //avagadro number
e=1.6*10**-19        //charge(c)
L=5                  //length(m)
R=0.06               //resistance(ohm)
I=15                 //current(A)
n=3                  //number of electrons

//Calculation
N=n*D*Na/w              //number of conduction
   electrons(/m**3)
mew=1/(rho*N*e)         //mobility(m**2/Vs)
vd=I*R/(L*rho*N*e)      //drift velocity(m/s)

//Result
```

```
printf("\n number of conduction electrons is %0.4f
   *10**29 /m**3",N/10**29)
printf("\n mobility is %0.5f m**2/Vs",mew)
printf("\n drift velocity is %0.1f *10**-4 m/s",vd
   *10**4)
```

**Scilab code Exa 5.6** mobility

```
clear
//
//
//

//Variable declaration
D=8.92*10**3       //density(kg/m**3)
rho=1.73*10**-8     //resistivity(ohm m)
W=63.5      //atomic weight
Na=6.02*10**26     //avagadro number
e=1.6*10**-19       //charge(c)

//Calculation
n=D*Na/W
mew=1/(rho*n*e)       //mobility(m**2/Vs)

//Result
printf("\n mobility is %0.5f m**2/Vs",mew)
printf("\n answer given in the book is wrong")
```

**Scilab code Exa 5.8** relaxation time

```
clear
//
//
```

```
//

//Variable declaration
rho=1.50*10**-8     //resistivity(ohm m)
n=6.5*10**28        //conduction 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(sec)

//Result
```

**Scilab code Exa 5.9** `thermal velocity`

```
clear
//
//
//

//Variable declaration
m=9.11*10**-31        //mass(kg)
rho=1.54*10**-8       //resistivity(ohm m)
e=1.602*10**-19       //charge(c)
E=10**2               //electric field(V/m)
n=5.8*10**28          //number of electrons
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
```

**Scilab code Exa 5.10** `mean free path`

```
clear
//
//
//

//Variable declaration
m=9.11*10**-31        //mass(kg)
e=1.602*10**-19       //charge(c)
E=5.5                 //fermi energy(V/m)
tow=3.97*10**-14      //relaxation time(s)

//Calculation
Vf=sqrt(2*E*e/m)      //fermi velocity(m/s)
lamda=Vf*tow               //mean free path(m)

//Result
printf("\n fermi velocity is %0.2f  *10**6 m/s",Vf
   /10**6)
printf("\n mean free path is %0.2f  *10**-8 m",lamda
   *10**8)
```

**Scilab code Exa 5.11** `fermi energy`

```
clear
//
//
//

//Variable declaration
```

```
n=1                     //number of electrons
NA=6.025*10**26         //avagadro number
D=10500                 //density(kg/m**3)
M=107.9                 //atomic weight(kg)
m=9.11*10**-31          //mass(kg)
h=6.63*10**-34          //plancks constant(Js)

//Calculation
n=n*NA*D/M              //electronic concentration(per
    m**3)
x=(3*n/%pi)**(2/3)
Ef=h**2*x/(8*m)         //fermi energy(J)

//Result
printf("\n electronic concentration is %0.3f
    *10**28 per m**3",n/10**28)
printf("\n fermi energy is %0.2f  *10**-19 J",Ef
    *10**19)
```

**Scilab code Exa 5.12** `drift velocity`

```
clear
//
//
//

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

```

```
//Calculation
J=I/A              //current density(amp/m**2)
n=n*Na*D/w
vd=J/(n*e)         //drift velocity(m/s)

//Result
```

# Chapter 6

# dielectric properties

**Scilab code Exa 6.1** `dielectric constant`

```
clear
//
//
//

//Variable declaration
alpha_e=10**-40       //polarisability(Fm**2)
N=3*10**28            //density of atoms
epsilon0=8.85*10**-12

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

//Result
```

**Scilab code Exa 6.2** `charge on plates`

```
clear
//
//
//

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

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

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

**Scilab code Exa 6.3** polarisability

```
clear
//
//
//

//Variable declaration
epsilonr=1.0000684      //dielectric constant
N=2.7*10**25    //number of atoms
epsilon0=8.85*10**-12

//Calculation
alpha_e=epsilon0*(epsilonr-1)/N     //polarisability(
    Fm**2)

```

```
//Result
printf("\n polarisability is %e  Fm**2",alpha_e)
```

**Scilab code Exa 6.5** polarisation

```
clear
//
//
//

//Variable declaration
epsilonr=5      //relative permittivity
V=12            //potential(V)
d=2*10**-3      //separation(m)
epsilon0=8.85*10**-12

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

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

**Scilab code Exa 6.6** electronic polarisability

```
clear
//
//
//

//Variable declaration
epsilonr=3.75        //relative dielectric constant
gama=1/3      //internal field constant
```

```
D=2050          //density(kg/m**3)
M=32            //atomic weight(amu)
Na=6.02*10**26       //avagadro number
epsilon0=8.85*10**-12

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

//Result
printf("\n electronic polarisability is %0.2f
   *10**-40 Fm**2",alpha_e*10**40)
printf("\n answer in the book varies due to rounding
    off errors")
```

**Scilab code Exa 6.8** displacement

```
clear
//
//
//

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

//Calculation
alphae=epsilon0*(epsilonr-1)/N     //polarisability(
   Fm**2)
```

```
r=(alphae/(4*%pi*epsilon0))**(1/3)     //radius of
   electron cloud(m)
d=alphae*E/(Z*e)                      //displacement(m)

//Result
printf("\n polarisability is %e  Fm**2",alphae)
printf("\n radius of electron cloud is %0.3f
   *10**-11 m",r*10**11)
printf("\n answer for radius given in the book
   varies due to rounding off errors")
printf("\n displacement is %0.1f  *10**-16 m",d
   *10**16)
```

**Scilab code Exa 6.9** voltage across plates

```
clear
//
//
//

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

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

//Result
```

**Scilab code Exa 6.10** `polarisability`

```
clear
//
//
//

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

//Calculation
p=4*%pi*epsilon0*r**3         //dipole moment per
   unit electric field(F-m**2)
P=N*p                                  //polarisation(C-m)
epsilonr=1+(4*%pi*r**3*N/E)   //dielectric constant
alphae=epsilon0*(epsilonr-1)/N    //polarisability(Fm
   **2)

//Result
```

**Scilab code Exa 6.11** `polarisability`

```
clear
//
//
//

//Variable declaration
N=2.7*10**25    //number of atoms
epsilon0=8.85*10**-12
epsilonr=1.000435   //dielectric constant

```

```
//Calculation
alphae=epsilon0*(epsilonr-1)/N    //polarisability(Fm
    **2)

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

**Scilab code Exa 6.12** `polarisability`

```
clear
//
//
//

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

//Calculation
N=NA*D/M             //number of atoms
alphae=epsilon0*(epsilonr-1)/N    //polarisability(Fm
    **2)

//Result
```

# Chapter 7

# magnetic properties

**Scilab code Exa 7.1** magnetic moment

```
clear
//
//
//

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

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

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

**Scilab code Exa 7.2** magnetic flux density

```
clear
//
//
//

//Variable declaration
chi=-0.25*10**-5          //magnetic susceptibility
H=1000                //magnetic field intensity(amp/m)
mew0=4*%pi*10**-7

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

//Result
```

---

**Scilab code Exa 7.3** magnetic flux density

```
clear
//
//
//

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

//Calculation
M=H*(mewr-1)          //magnetisation(A/m)
```

```
B=mew0*(H+M)          //magnetic flux density(wb/m**2)

//Result
```

**Scilab code Exa 7.4** magnetic flux density

```
clear
//
//
//

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

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

//Result
```

**Scilab code Exa 7.5** magnetic moment

```
clear
//
//
//

//Variable declaration
d=0.1                //diameter(m)
i=0.5                //current(ampere)

```

```
//Calculation
r=d/2                //radius of atom(m)
mew=i*%pi*r**2         //magnetic moment(A-m**2)

//Result
printf("\n magnetic moment is %0.2f  *10**-3 A-m**2"
    ,mew*10**3)
```

**Scilab code Exa 7.6** `relative permeability`

```
clear
//
//
//

//Variable declaration
mew0=4*%pi*10**-7
B=0.0044           //magnetic flux density(wb/m**2)
M=3300             //magnetisation(A/m)

//Calculation
H=(B/mew0)-M     //magnetising force(amp/m)
mewr=1+(M/H)     //relative permeability

//Result
printf("\n magnetising force is %0.1f  A/m",H)
printf("\n relative permeability is %0.2f  ",mewr)
printf("\n answers given in the book are wrong")
```

**Scilab code Exa 7.7** `change in magnetic moment`

```
clear
//
```

```
//
//

//Variable declaration
r=0.52*10**-10      //radius(m)
B=3      //magnetic induction(web/m**2)
e=1.6*10**-19       //charge(c)
m=9.1*10**-31       //mass(kg)

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

//Result
```

**Scilab code Exa 7.8** `change in magnetic moment`

```
clear
//
//
//

//Variable declaration
r=5.29*10**-11      //radius(m)
B=2      //magnetic induction(web/m**2)
e=1.6*10**-19       //charge(c)
m=9.1*10**-31       //mass(kg)

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

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

**Scilab code Exa 7.9** `susceptibility at 300K`

```
clear
//
//
//

//Variable declaration
chi1=2.8*10**-4        //susceptibility
T1=350                 //temperature(K)
T2=300                 //temperature(K)

//Calculation
chi2=(chi1*T1)/T2      //susceptibility at 300K

//Result
```

**Scilab code Exa 7.10** `relative permeability of iron`

```
clear
//
//
//

//Variable declaration
B0=6.5*10**-4        //magnetic field(Tesla)
B=1.4                //magnetic field(Tesla)

//Calculation
mewr=B/B0            //relative permeability of iron

//Result
```

# Chapter 8

# semiconductors and physics of semiconductor devices

**Scilab code Exa 8.2** number of donor atoms

```
clear
//
//
//

//Variable declaration
e=1.6*10**-19     //charge(c)
mew_n=0.3         //electron mobility(m**2/Vs)
rho=0.25          //resistivity(ohm m)

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

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

**Scilab code Exa 8.3** `diffusion coefficient`

```
clear
//
//
//

//Variable declaration
e=1.6*10**-19      //charge(c)
mewn=0.21          //electron mobility(m**2/Vs)
T=300              //temperature(K)
KB=1.38*10**-23    //boltzmann constant

//Calculation
Dn=mewn*KB*T/e     //diffusion coefficient(m**2/sec)

//Result
```

---

**Scilab code Exa 8.4** `hole mobility`

```
clear
//
//
//

//Variable declaration
e=1.6*10**-19      //charge(c)
RH=3.22*10**-4     //hall coefficient(m**3C-1)
rho=8.5*10**-3     //resistivity(ohm m)

//Calculation
p=1/(RH*e)         //hole concentration(m-3)
```

```
mewp=RH/rho          //hole mobility(m**2/Vs)

//Result
printf("\n hole concentration is %0.1f  *10**21 m-3"
   ,p/10**21)
printf("\n hole mobility is %0.5f  m**2/Vs",mewp)
```

**Scilab code Exa 8.5** intrinsic concentration

```
clear
//
//
//

//Variable declaration
e=1.6*10**-19       //charge(c)
mew_e=0.36          //electron mobility(m**2/Vs)
mew_h=0.17          //hole mobility(m**2/Vs)
rhoi=2.12           //resistivity(ohm m)

//Calculation
ni=1/(rhoi*e*(mew_e+mew_h))      //intrinsic
   concentration(per m**3)

//Result
printf("\n intrinsic concentration is %0.2f  *10**16
    per m**3",ni/10**16)
```

**Scilab code Exa 8.6** resistivity

```
clear
//
//
```

```
//

//Variable declaration
e=1.6*10**-19      //charge(c)
mew_e=0.39         //electron mobility(m**2/Vs)
mew_h=0.19         //hole mobility(m**2/Vs)
ni=2.4*10**19      //intrinsic concentration(per m**3)

//Calculation
rhoi=1/(ni*e*(mew_e+mew_h))     //resistivity(ohm m)

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

**Scilab code Exa 8.7** conductivity

```
clear
//
//
//

//Variable declaration
ni=1.5*10**16       //charge carriers(per m**3)
e=1.6*10**-19       //charge(c)
mew_e=0.135         //electron mobility(m**2/Vs)
mew_h=0.048         //hole mobility(m**2/Vs)
N=10**23            //number of atoms(per m**3)

//Calculation
sigma=ni*e*(mew_e+mew_h)
p=ni**2/N           //hole concentration(per m**3)
sigman=N*e*mew_e    //conductivity(per ohm m)

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

**Scilab code Exa 8.8** hole mobility

```
clear
//
//
//

//Variable declaration
e=1.6*10**-19        //charge(c)
RH=3.66*10**-4       //hall coefficient(m**3C-1)
rho=8.93*10**-3      //resistivity(ohm m)

//Calculation
p=1/(RH*e)           //hole concentration(m-3)
mew=RH/rho          //hole mobility(m**2/Vs)

//Result
printf("\n hole concentration is %0.1f  *10**22 m-3"
    ,p/10**22)
printf("\n hole mobility is %0.3f  *10**-2 m**2/Vs",
    mew*10**2)
```

**Scilab code Exa 8.9** conductivity

```
clear
//
//
//

```

```
//Variable declaration
e=1.6*10**-19     //charge(c)
ni=1.5*10**16     //particle density(per m**3)
mew_e=0.13        //electron mobility(m**2/Vs)
mew_h=0.05        //hole mobility(m**2/Vs)

//Calculation
sigma=ni*e*(mew_e+mew_h)        //conductivity(per
   ohm m)

//Result
```

**Scilab code Exa 8.10** `conductivity`

```
clear
//
//
//

//Variable declaration
e=1.6*10**-19     //charge(c)
ni=1.5*10**16     //particle density(per m**3)
mew_e=0.14        //electron mobility(m**2/Vs)
mew_h=0.05        //hole mobility(m**2/Vs)
D=2.33*10**3      //density(kg/m**3)
A=28.09           //atomic weight(kg)
NA=6.025*10**26   //avagadro number

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

```

```
//Result
printf("\n conductivity is %0.1f  per ohm m",sigma)
```

**Scilab code Exa 8.13** density of donor atoms

```
clear
//
//
//

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

//Calculation
n=1/(rho*mewn*e)      //density of donor atoms(
   electrons/m**3)

//Result
```

**Scilab code Exa 8.15** diffusion coefficient

```
clear
//
//
//

//Variable declaration
e=1.6*10**-19      //charge(c)
mew_e=0.19         //electron mobility(m**2/Vs)
T=300           //temperature(K)
KB=1.38*10**-23    //boltzmann constant
```

```

//Calculation
Dn=mew_e*KB*T/e     //diffusion coefficient(m**2/sec)

//Result
```

**Scilab code Exa 8.16** `energy gap`

```
clear
//
//
//

//Variable declaration
KB=1.38*10**-23        //boltzmann constant
e=1.602*10**-19        //charge(c)
rho1=4.5
rho2=2.0
T1=293            //temperature(K)
T2=305            //temperature(K)

//Calculation
Eg=2*KB*log(rho1/rho2)/((1/T1)-(1/T2))          //
   energy gap(J)
Eg=Eg/e
   //energy gap(eV)

//Result
```

**Scilab code Exa 8.17** `peak output voltage`

```
clear
//
```

```
//
//

//Variable declaration
Vm=20         //voltage(V)
RL=500        //load resistance(ohm)
rf=10         //forward resistance(ohm)
VB=0.7        //bias voltage(V)

//Calculation
Im=(Vm-VB)*10**3/(rf+RL)      //peak current(mA)
Vo=Im*RL/10**3                //peak output voltage(V
   )

//Result
printf("\n peak current is %0.1f  mA",Im)
printf("\n peak output voltage is %0.1f  V",Vo)
```

**Scilab code Exa 8.18** `ripple factor`

```
clear
//
//
//

//Variable declaration
Vrms=200       //voltage(V)
RL=1000        //load resistance(ohm)

//Calculation
Im=Vrms*sqrt(2)/RL       //peak current(A)
Idc=2*Im/%pi               //average DC current(A)
Vdc=int(Idc*RL)                    //dc voltage(V)
x=(Vrms/Vdc)**2
gama=sqrt(x-1)*Vdc       //ripple factor(V)
```

```

//Result
```

# Chapter 9

# superconductivity

**Scilab code Exa 9.2** `frequency`

```
clear
//
//
//

//Variable declaration
e=1.6*10**-19     //charge(c)
h=6.626*10**-34   //plank constant
V=8.5*10**-6      //voltage(V)

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

//Result
printf("\n frequency is %0.1f  *10**9 Hz",new/10**9)
```

**Scilab code Exa 9.3** `critical field`

```
clear
//
//
//

//Variable declaration
T=2            //temperature(K)
Tc=3.7         //critical temperature(K)
H0=0.0306      //critical magnetic field(A/m)

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

//Result
printf("\n critical field is %0.5f  Tesla",Hc)
```

**Scilab code Exa 9.4** `maximum critical temperature`

```
clear
//
//
//

//Variable declaration
Hc=200*10**3      //critical magnetic field(A/m)
Tc=12             //critical temperature(K)
H0=250*10**3      //critical magnetic field(A/m)

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

//Result
printf("\n maximum critical temperature is %0.3f  K"
   ,T)
```

**Scilab code Exa 9.5** `critical field`

```
clear
//
//
//

//Variable declaration
T=2.5           //temperature(K)
Tc=3.7          //critical temperature(K)
H0=0.03         //critical magnetic field(A/m)

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

//Result
printf("\n critical field is %0.4f  Tesla",Hc)
```

**Scilab code Exa 9.6** `frequency`

```
clear
//
//
//

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

//Calculation
```

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

//Result
```

# Chapter 10

# lasers

**Scilab code Exa 10.1** `band gap`

```
clear
//
//
//

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

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

//Result
printf("\n band gap is %0.1f  eV",Eg)
```

**Scilab code Exa 10.2** `wavelength`

```
clear
//
//
//

//Variable declaration
c=3*10**8                  //velocity of light(m/s)
h=6.63*10**-34             //plank's constant(Js)
e=1.6*10**-19              //charge(coulomb)
Eg=1.44*e                  //band gap(eV)

//Calculation
lamda=h*c*10**10/Eg        //wavelength(angstrom)

//Result
printf("\n wavelength is %0.0f  angstrom",lamda)
```

# Chapter 11

# fibre optics

**Scilab code Exa 11.1** `fractional index change`

```
clear
//
//
//

//Variable declaration
n1=1.48          //Core refractive index
n2=1.45          //Cladding refractive index

//Calculation
NA=sqrt(n1**2-n2**2)     //numerical aperture
theta0=asin(NA)          //acceptance angle(radian)
theta0=theta0*180/%pi      //acceptance angle(degrees
   )
theta0m=60*(theta0-int(theta0)) //acceptance angle(
   minutes)
thetac=asin(n2/n1)       //critical angle(radian)
thetac=thetac*180/%pi      //critical angle(degrees)
thetacm=60*(thetac-int(thetac)) //critical angle(
   minutes)
delta=(n1-n2)/n1               //fractional index
```

```
    change

//Result
printf("\n numerical aperture is %0.4f  ",NA)
printf("\n acceptance angle is %0.3f degrees %0.0f
    minutes",theta0,theta0m)
printf("\n critical angle is %0.3f degrees %0.3f
    minutes",thetac,thetacm)
printf("\n fractional index change is %0.2f  ",delta
    )
```

**Scilab code Exa 11.2** `acceptance angle`

```
clear
//
//
//

//Variable declaration
n1=1.563           //Core refractive index
n2=1.498           //Cladding refractive index

//Calculation
NA=sqrt(n1**2-n2**2)     //numerical aperture
theta0=asin(NA)          //acceptance angle(radian)
theta0=theta0*180/%pi       //acceptance angle(degrees
    )
theta0m=60*(theta0-int(theta0)) //acceptance angle(
    minutes)

//Resul"
```

**Scilab code Exa 11.3** `fractional index change`

```
clear
//
//
//

//Variable declaration
n1=1.563          //Core refractive index
n2=1.498          //Cladding refractive index

//Calculation
delta=(n1-n2)/n1     //fractional index change

//Result
```

**Scilab code Exa 11.4** `numerical aperture`

```
clear
//
//
//

//Variable declaration
n1=1.55           //Core refractive index
n2=1.50           //Cladding refractive index

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

//Result
printf("\n numerical aperture is %0.4f  ",NA)
```

**Scilab code Exa 11.5** `numerical aperture`

```
clear
//
//
//

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

//Calculation
n1n2=NA**2/n1_n2
n2=(n1n2-n1_n2)/2        //Cladding refractive index
n1=n2+n1_n2              //Core refractive index

//Result
```

**Scilab code Exa 11.6** `numerical aperture`

```
clear
//
//
//

//Variable declaration
n1=1.55          //Core refractive index
n2=1.50          //Cladding refractive index

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

//Result
printf("\n numerical aperture is %0.4f  ",NA)
```

**Scilab code Exa 11.7** `acceptance angle`

```
clear
//
//
//

//Variable declaration
n1=1.48           //Core refractive index
n2=1.45           //Cladding refractive index

//Calculation
NA=sqrt(n1**2-n2**2)     //numerical aperture
theta0=asin(NA)          //acceptance angle(radian)
theta0=theta0*180/%pi      //acceptance angle(degrees
   )
theta0m=60*(theta0-int(theta0)) //acceptance angle(
   minutes)

//Result
```

---

**Scilab code Exa 11.8** `refractive index of cladding`

```
clear
//
//
//

//Variable declaration
NA=0.33    //numerical aperture
delta=0.02      //refractive index of cladding

//Calculation
x=1-delta
n1=sqrt(NA**2/(1-x**2))     //refractive index of
```

```
   core
n2=x*n1                            //refractive index
   of cladding

//Result
printf("\n refractive index of core is %0.4f  ",n1)
printf("\n refractive index of cladding is %0.3f  ",
   n2)
```

---

**Scilab code Exa 11.9** `acceptance angle`

```
clear
//
//
//

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

//Calculation
n1=sqrt((NA**2)+(n2**2))      //core refractive index
x=sqrt((n1**2)-(n2**2))/n0
theta0=asin(x)           //acceptance angle(radian)
theta0=theta0*180/%pi      //acceptance angle(degrees
   )
theta0m=60*(theta0-int(theta0)) //acceptance angle(
   minutes)
theta0s=60*(theta0m-int(theta0m)) //acceptance angle
   (seconds)

//Resul"
printf("\n acceptance angle is %0.3f degrees %0.3f
   minutes %0.3f seconds",theta0,theta0m,theta0s)
```

```
printf("\n answer for angle in seconds given in the
   book varies due to rounding off errors")
```

**Scilab code Exa 11.10** `fractional index change`

```
clear
//
//
//

//Variable declaration
n1=1.45           //Core refractive index
n2=1.44           //Cladding refractive index

//Calculation
delta=(n1-n2)/n1     //fractional index change

//Result
printf("\n fractional index change is %0.4f  *10**-3
   ",delta*10**3)
```

**Scilab code Exa 11.11** `critical angle`

```
clear
//
//
//

//Variable declaration
n1=1.50           //Core refractive index
delta=4/100       //fractional index change

//Calculation
```

```
n2=n1-(delta*n1)                //Cladding refractive
    index
NA=sqrt(n1**2-n2**2)     //numerical aperture
theta0=asin(NA)          //acceptance angle(radian)
theta0=theta0*180/%pi      //acceptance angle(degrees
    )
theta0m=60*(theta0-int(theta0)) //acceptance angle(
    minutes)
thetac=asin(n2/n1)       //critical angle(radian)
thetac=thetac*180/%pi      //critical angle(degrees)
thetacm=60*(thetac-int(thetac)) //critical angle(
    minutes)

//Result
```

**Scilab code Exa 11.12** acceptance angle

```
clear
//
//
//

//Variable declaration
n1=1.563           //Core refractive index
n2=1.498           //Cladding refractive index

//Calculation
NA=sqrt(n1**2-n2**2)     //numerical aperture
theta0=asin(NA)          //acceptance angle(radian)
theta0=theta0*180/%pi      //acceptance angle(degrees
    )
theta0m=60*(theta0-int(theta0)) //acceptance angle(
    minutes)

//Result
```

```
printf("\n numerical aperture is %0.3f  ",NA)
printf("\n acceptance angle is %0.3f degrees %0.0f
   minutes",theta0,theta0m)
printf("\n answer for angle in minutes given in the
   book varies due to rounding off errors")
```

# Chapter 13

# acoustics of buildings and acoustic quieting

**Scilab code Exa 13.1** `reverberation time of hall with audience`

```
clear
//
//
//

//Variable Declaration
A=92.9          //absorption(m**2)
V=2265          //volume(m**3)

//Calculation
T1=0.161*V/A        //reverberation time of hall
   without audience(seconds)
T2=0.161*V/(A*2)    //reverberation time of hall
   with audience(seconds)

//Result
printf("\n reverberation time of hall without
   audience is %0.1f  seconds",T1)
printf("\n reverberation time of hall with audience
    is %0.3f  seconds",T2)
```