Scilab Textbook Companion for Introduction To Modern Physics Volume 1 by R. B. Singh[1]

Created by
Shomit Goyal
B. Tech
Others
IIT-Bombay
College Teacher
None
Cross-Checked by
Mukul Kulkarni

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: Introduction To Modern Physics Volume 1

Author: R. B. Singh

Publisher: New Age International, New Delhi

Edition: 2

Year: 2009

ISBN: 978-81-224-2922-0

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

The Special Theory of Relativity

Scilab code Exa 1.1.3 Ex 3

```
1 //variable initialization
2 x=50,y=20,z=10                                    //x,y,z
     cordinates in meters(frame s)
3 t=5*10^(-8);                                      //time in
     seconds(frame s)
4 velocity=0.6*3*10^8;                              //velocity
     of observer in s' frame relative to s in meter/
     second
5 c=3*10^8;                                         //speed of
     light in meter/second
6 Beta=0.6;
7 Gamma=1/((1-Beta^2)^(1/2));
8
9 //calculation of cordinates in s' frame
10 xdash=Gamma*(x-(velocity*t));                     //value of
     x cordinate in frame s' in meters
11 ydash=y;                                         //value of
     y cordinate in frame s' in meters
12 zdash=z;                                         //value of
```

```
    z cordinate in frame s' in meters
13 tdash=Gamma*(t-((velocity*x)/(c^2)));      //value of
    t in frame s' in seconds
14
15 printf("\nValue of space time cordinates in frame s
   ':\n\t x'=%.2f meter\n\t y'=%.0f meter\n\t z'=%.0
   f meter\n\t t'=%.2e second",xdash,ydash,zdash,
   tdash);
```

Scilab code Exa 1.1.4 Ex 4

```
1 //variable initialization
2 x1=20                                              //
   position of event 1 in meters(frame s)
3 t1=2*10^(-8);                                      //time
   of event 1 in seconds(frame s)
4 x2=60                                              //
   position of event 2 in meters(frame s)
5 t2=3*10^(-8);                                      //time
   of event 2 in seconds(frame s)
6 c=3*10^8;                                          //speed
   of light in meter/second
7 v=0.6*c;                                           //speed
   of frame s' relative to frame s (meter/second)
8 Beta=0.6
9 Gamma=1/((1-Beta^2)^(1/2));
10
11 //part(i):spatial separation of the events in frame
   s'
12 separation=Gamma*((x2-x1)-v*(t2-t1));             //
   spatial separation of the events in frame s' (
   meter)
13
14 //part(ii):time interval between the two events in
   frame s'
```

```
15 interval=Gamma*((t2-t1)-(v*(x2-x1))/(c^2)); //time
     interval between the two events in frame s' (
     second)
16
17 printf("\nIn frame s':\n\t (i)spatial separation=%.2
     f meter\n\t (ii)time interval=%.2e second",
     separation,interval);
```

Scilab code Exa 1.1.5 Ex 5

```
1 //variable initialization
2 x1=24                                              //
     position of event 1 in meters(frame s)
3 t1=8*10^(-8);                                      //time
     of event 1 in seconds(frame s)
4 x2=48                                              //
     position of event 2 in meters(frame s)
5 t2=4*10^(-8);                                      //time
     of event 2 in seconds(frame s)
6 c=3*10^8;                                          //
     speed of light in meter/second
7
8 //calculation of velocity of the frame s' so that
     both the events occur simultaneously
9 v=((c^2)*(t2-t1))/(x2-x1);                         //
     velocity of the frame s' (meter/second)
10
11 printf("\nvelocity of the frame s' = %.1e meter/
     second",v);
```

Scilab code Exa 1.1.6 Ex 6

```
1 //variable initialization
```

```
2 interval_s=1
                                                    //
    time difference between two events in frame s (
    second)
3 interval_sdash=4
                                                  //time
    difference between two events in frame s' (second
    )
4 separation_s=0
                                                    //
    spatial separation of two events in frame s (
    meter)
5 c=3*10^8;
                                                      //
    speed of light (meter/second)
6 v=rand();

    //assign a random value to unknown velocity(meter
    /second)
7
8 //calculation of spatial separation of the events in
     frame s':
9 Gamma=interval_sdash/(interval_s-(v*(separation_s))
    /(c^2)); //calculating gamma
10
11 separation=-2*(((Gamma^2)-1)^(1/2))*c;
                          //spatial separation in s' (
    meter)
12
13 printf("\nspatial separation of the events in frame
    s' = %.2e meter",separation);
```

Scilab code Exa 1.1.7 Ex 7

```
1 //variable initialization
```

```
2  interval_s=0
     //time difference between two events in frame s (
     second)
3  separation_s=1;
                                                        //
     spatial separation of two events in frame s (
     meter)
4  separation_sdash=2;
                                                      //
     spatial separation of two events in frame s' (
     meter)
5  c=3*10^8;
     //speed of light (meter/second)
6  v=rand();
     //assign a random value to unknown velocity of
     frame s' with respect to frame s (meter/second)
7
8  //calculation of time interval between the events in
       frame s'
9  Gamma=separation_sdash/(separation_s-(v*interval_s))
     ;          //calculating value of Gamma
10 Beta=(1-1/(Gamma^2))^(1/2);
                                             //calculating
     value of Beta
11 v=Beta*c;
     //velocity of s' with respect to s (meter/second)
12 interval_sdash=Gamma*(interval_s-((v*separation_s)/(
     c^2)));    //time interval between the events in
     frame s' (second)
13
14 printf("\nThe time interval between the events in
     frame s' = %.2e X0",interval_sdash);
```

Scilab code Exa 1.1.8 Ex 8

```
1 //variable initialization
2 IbyI_not=.99                          //ratio of moving
      length and rest length
3 c=3*10^8;                             //speed of light
     (meter/second)
4
5 //calculation of velocity of rocket ship
6 Beta=(1-IbyI_not^2)^(1/2);            //calculating
     value of Beta
7 v=Beta*c;                             //velocity of
     rocket ship (meter/second)
8
9 printf("\nThe velocity of the rocket ship = %.2e
     meter/second",v);
```

Scilab code Exa 1.1.9 Ex 9

```
1 //variable initialization
2 l_dash=1

     //length of the rod in frame s' (meter)
3 Theta_dash_degree=45

     //angle of the rod with x-axis in frame s' (
     degree)
4 Beta=1/2

     //value of Beta
5
```

```
6 //calculation of the length of the rod and its
     inclination with x-axis in the frame s
7 Theta_dash_radian=Theta_dash_degree*(%pi/180);

     //conversion of angle Theta in radian from degree
      (radian)
8 l=((l_dash^2)*((sin(Theta_dash_radian))^2+((1-(Beta
     ^2))*((cos(Theta_dash_radian))^2))))^(1/2);
               //length of the rod in frame s (meter)
9 tan_theta=tan(Theta_dash_radian)/((1-Beta^2)^(1/2));


     //tan of angle of rod with x-axis in frame s
10 theta=atand(tan_theta);

     //angle of rod with x-axis in frame s (degree)
11
12 printf("\n\t The length of the rod = %f meter\n\t
     Inclination of rod with x-axis = %f degree",l,
     theta);
```

Scilab code Exa 1.1.10 Ex 10

```
1 //variable initialization
2 c=3*10^8;                                    //speed
     of light (meter/second)
3
4 //calculation of velocity of the reference frame s'
5 Beta=(1-((1/1.25)^2))^(1/2);                  //
     calculating Beta (1.25 comes from the fact that
     in frame s' density of bloc is 25% greater than
     frame s)
6 v=Beta*c;                                     //
     velocity of the reference frame s'
7
```

```
8 printf("\nBeta = %.1e\nThe velocity of the frame s'
    = %.1e meter/second",Beta,v);
```

Scilab code Exa 1.1.11 Ex 11

```
1 //Variable initialization
2 E=0.25
    //Energy of photon (MeV)
3 Theta=(120*%pi)/180;
    //Scattering angle of photon (radian)
4 a=0.51
    //Value of m0*c^2 (Mev)
5
6 //Calculation of energy of the photon
7 E_dash=E/(1+(E/a)*(1-cos(Theta)));
                                                    //
    Energy of the scattered photon (MeV)
8
9 printf("\nEnergy of the scattered photon = %.3f Mev"
    ,E_dash);
```

Scilab code Exa 1.1.12 Ex 12

```
1 //variable initialization
2 deltaTow=1*10^(-6);                              //mean
    proper lifetime of particle (second)
3 Beta=0.9                                         //value
     of Beta
4 v=2.7*10^8;                                      //
    velocity of particle (meter/second)
```

```
5
6 //part(i):lifetime of the particle in the laboratory
      frame
7 deltaT=deltaTow/((1-Beta^2)^(1/2));          //
     lifetime of the particle in the laboratory frame
     (second)
8
9 //part(ii):distance traversed by the particle in the
      laboratory frame
10 d=v*deltaT;                                    //
     distance traversed by the particle in the
     laboratory before disintegration (meter)
11
12 printf("\nIn laboratory frame:\n\t lifetime of the
     particle = %.2e second\n\t distance traversed by
     the particle = %.1e meter",deltaT,d);
```

Scilab code Exa 1.1.13 Ex 13

```
1 //variable initialization
2 Theta=120

     //Scattering angle of photon (degree)
3 T=0.45

     //Kinetic energy of electron (MeV)
4 a=0.51

     //Value of m0*c^2 (Mev)
5
6 //Calculation of the energy of the incident photon
7 E=0.5*T*(1+(1+(2*a)/(T*(sind(Theta/2))^2))^(1/2));
                                    //Energy of the
     incident photon (MeV)
8
```

```
9 printf("\nEnergy of the incident photon = %.2f Mev",
    E);
```

Scilab code Exa 1.1.14 Ex 14

```
1 //variable initialization
2 c=3*10^8;                                    //speed of
    light (meter/second)
3 u1=0.6*c;                                    //speed of
    Beta particle 1 in lab frame (meter/second)
4 u2=-0.8*c;                                   //speed of
    Beta particle 2 in lab frame (meter/second)
5 v=u1;                                        //velocity
    of frame s' where frame s' is attached to the
    first Beta particle (meter/second)
6
7 //velocity of 2nd Beta particle relative to the 1st
    Beta particle(assuming frame s' is attached to
    the first Beta particle)
8 u2_dash=(u2-v)/(1-((u2*v)/c^2));              //velocity
    of 2nd Beta particle relative to the 1st Beta
    particle (meter/second)
9
10 printf("\n\tThe velocity of 2nd Beta particle
    relative to the 1st Beta particle = %.2e meter/
    second",u2_dash);
```

Scilab code Exa 1.1.15 Ex 15

```
1 //variable initialization
2 m0=1                                         //let rest
    mass of particle to be 1 (kg)
```

```
3 m=3*m0;                                    //moving
     mass of particle (kg)
4 c=3*10^8;                                  //speed of
     light (meter/second)
5
6 //calculation of speed of particle
7 Beta=(1-(m0/m)^2)^(1/2);                   //
     Calculation fo Beta
8 v=Beta*c;                                  //speed of
     particle (meter/second)
9
10 printf("\n\tThe speed of The particle = %.2e meter/
     second",v);
```

Scilab code Exa 1.1.23 Ex 23

```
1 //variable initialization
2 RestEnergy=0.51                            //energy
     of electron if the electron is at rest (Mev)
3 KineticEnergy=2                            //kinetic
      energy of electron (Bev)
4 c=3*10^8;                                  //speed
     of light (meter/second)
5
6 //calculation of momentum of electron
7 p=(KineticEnergy*10^9)/c;                  //
     momentum of electron neglecting rest energy
     relative to kinetic energy (Bev*second/meter)
8
9 printf("\n\tThe momentum of the electron = %.2f ev*
     second/meter",p);
```

Scilab code Exa 1.1.24 Ex 24

```
1 //variable initialization
2 n=0.01                                  //fractional
     increase in momentum
3 c=3*10^8;                              //speed of light (
     meter/second)
4
5 //calculation of velocity of particle
6 Beta=(n*(2-n))^(1/2);                   //calculation of
     Beta
7 v=Beta*c;                              //velocity of
     particle (meter/second)
8
9 printf("\nBeta = %.1e\nThe velocity of the particle
     = %.2e meter/second",Beta,v);
```

Scilab code Exa 1.1.26 Ex 26

```
1 //variable initialization
2 RestEnergy=0.51                                    //
     rest energy of electron (Mev)
3 T=1                                                //
     potential difference i.e. kinetic energy (Mev)
4 c=3*10^8;                                          //
     speed of light (meter/second)
5
6 //calculation of speed of electron
7 Beta=(1-(RestEnergy/(T+RestEnergy))^2)^(1/2);      //
     calculation of Beta
8 v=Beta*c;                                          //
     speed of electron (meter/second)
9
10 printf("\n\tThe speed of the electron = %.2e meter/
     second",v);
11
12 //Note: In the book answer of Beta is slightly wrong
```

Scilab code Exa 1.1.27 Ex 27

```
1 //variable initialization
2 RestEnergy=0.51                                    //
     rest energy of electron (Mev)
3 T=2000                                             //
     potential difference i.e. kinetic energy (Mev)
4
5 //part(i)effective mass of electron in terms of its
     rest mass
6 EffectiveMass=1+(T/RestEnergy);                    //
     ratio of effective mass of electron and rest mass
7
8 //part(ii)speed of electron in terms of the speed of
     light
9 Beta=(1-(1/EffectiveMass)^2)^(1/2);                //
     Calculatio of Beta
10
11 printf("\n\t The effective mass of electron = %.2f*
     mo    ;mo is rest mass of electron\n\t The speed
     of electron = %.10f*c    ;c is speed of light",
     EffectiveMass,Beta);
12
13 //Note: In the book answer of m/mo is slightly wrong
```

Scilab code Exa 1.1.28 Ex 28

```
1 //variable initialization
2 c=3*10^8;

     //speed of light(meter/second)
```

```
3 v1=0.6*c;
     //initial velocity of particle (meter/second)
4 v2=0.8*c;
     //final velocity of particle (meter/second)
5
6 //Calculation of work required to increase the
     velocity from v1 to v2:
7
8 //Classically
9 W_Classic=0.5*((v2/c)^2-(v1/c)^2);
                                        //ratio of work
     and m0*c^2 (mo is the rest mass of particle and c
      is the speed of light)
10
11 //Relativistically
12 W_Relative=(1/(1-(v2/c)^2)^(1/2))-(1/(1-(v1/c)^2)
     ^(1/2));          //ratio of work and m0*c^2 (mo
     is the rest mass of particle and c is the speed
     of light)
13
14 printf("\nWork required:\n\t Classically: Work = %.2
     f*m0*c^2\n\t Relativistically: Work = %.3f*m0*c
     ^2\nWhere m0:rest mass of particle & c:speed of
     light",W_Classic,W_Relative);
```

Scilab code Exa 1.1.29 Ex 29

```
1 //variable initialization
2 h=6.63*10^-34                                    //
     planck's constant (joule*second)
3 c=3*10^8                                         //
     speed of light (meter/second)
4 lambda1=5000*10^-10                              //
```

```
    wavelength (meter)
5 lambda2=0.1*10^-10                                    //
    wavelength (meter)
6
7 //Calculation of effective mass of photon:
8
9 //part(i): wavelength=5000
10 m1=h/(lambda1*c);                                     //
    effective mass of photon of wavelength 5000
11
12 //part(ii): wavelength=0.1
13 m2=h/(lambda2*c);                                     //
    effective mass of photon of wavelength 0.1
14
15 printf("\neffective mass of photon:\n\t(i) mass = %
    .2e kg\n\t(ii) mass = %.2e kg",m1,m2);
```

Scilab code Exa 1.1.30 Ex 30

```
1 //variable initialization
2 RestEnergy=0.51                                       //
    rest energy of electron (Mev)
3
4 //calculation of minimum energy of a gamma ray
    photon which is required to produce an electron
    positron pair
5 E=2*RestEnergy;                                       //
    minimum energy of gamma ray photon (Mev)
6
7 printf("\nMinimum energy required = %.2f Mev",E);
```

Scilab code Exa 1.1.33 Ex 33

```
1 //variable initialization
2 c=3*10^8;                                          //
     Speed of sound (meter/second)
3 M=1.97*10^30;                                      //
     Mass of sun (kg)
4 R=1.5*10^11;                                       //
     Mean radius of the earth orbit (meter)
5 sigma=1.4*10^3;                                    //
     Solar energy received by the earth (joule/meter
     ^2*second)
6
7 //calculation of the fractional loss of mass of the
     sun per second
8 loss=(4*%pi*R^2*sigma)/(M*c^2);                    //
     Fractional loss of mass of the sun per second
9
10 printf("\nThe fractional loss of mass of the sun per
     second = %.0e",loss);
```

Chapter 2

Origin of Quantum Concepts

Scilab code Exa 2.1.1 Ex 1

```
1 //variable initialization
2 c1=0.01
3 c2=0.1
4 c3=1
5 c4=10
6 b=2.898*10^-3;

    //Wien's constant (meter-kelvin)
7 h=(6.625*10^-34)/(2*%pi);

    //Planck's constant (joule-second)
8 c=3*10^8;

    //speed of light (meter/second)
9 k=1.38*10^-23;

    //Boltzmann constant (joule/kelvin)
10 T=3000

    //Temperature of black body (kelvin)
11 Delta_lembda=1*10^-9;
```

```
    //wavelength interval (meter)
12
13 //(a)Average energy of Planck's oscillator:
14 E1=c1/(%e^c1-1);

    //Average energy of Planck's oscillator
15 E2=c2/(%e^c2-1);

    //Average energy of Planck's oscillator
16 E3=c3/(%e^c3-1);

    //Average energy of Planck's oscillator
17 E4=c4/(%e^c4-1);

    //Average energy of Planck's oscillator
18
19 //(b)Power radiated by a unit area of a black body
20 P=(4*%pi^2*h*c^2*T^5*Delta_lembda)/(b^5*((%e^((2*%pi
   *h*c)/(b*k)))-1));          //The power radiated
   per unit area (watt/meter^2)
21
22 printf("\n(a) The average energy of Planck's
   oscillator:\n\t (i) Energy = %.3f kT\n\t (ii)
   Energy = %.2f kT\n\t (iii) Energy = %.2f kT\n\t (
   iv) Energy = %.5f kT\n\t k: Boltzmann constant =
   %.2e joule/kelvin    T: Temperature\n(b) The
   power radiated per unit area = %.2f watt/meter^2"
   ,E1,E2,E3,E4,k,P);
```

Scilab code Exa 2.1.2 Ex 2

```
1 //variable initialization
2 v=2*10^-2;
```

```
    //side of the cube (meter)
lembda=5000*10^-10;

    //wavelength (meter)
delta_lembda=10*10^-10;

    //range of wavelength (meter)
k=1.38*10^-23;

    //Boltzmann constant (joule/kelvin)
T=1500

    //Temperature of the cavity (kelvin)

//(i)Number of modes:
N=(8*%pi*v^3*delta_lembda)/lembda^4;
                                                    //
    number of modes

//(ii)Total radiant energy in the cavity:
U=N*k*T;

    //energy density (joule)

printf("\n(a) Number of modes = %.3e\n(b) Energy
    density = %.2e joule",N,U);

//Note: In book the answers of both the parts are
    wrong by one order of magnitude
```

Scilab code Exa 2.1.3 Ex 3

```
//variable initialization
m=0.1
```

```
    //mass of a spring-mass system (kg)
3 k=10

    //spring constant of a spring-mass system (newton
    /meter)
4 A=0.1

    //Amplitude of system oscillation (meter)
5 h=(6.625*10^-34)/(2*%pi);

    //Planck's constant (joule-second)
6 delta_n=1

    //change in quantum number
7
8 //(a) Quantum number n associated with the energy of
     the oscillator
9 f=(k/m)^(1/2);

    //frequency of oscillator (radian/second)
10 E=0.5*f*A^2;

    //Energy of oscillator (joule)
11 n=E/(h*f);

    //Quantum number of the oscillator
12
13 //(b) Fractional change in energy
14 change_E=delta_n/n;

    //fractional change in energy
15
16 //(c) Conclusion: This example illustrates that the
    energy levels of macroscopic oscillators are so
    close together that even most delicate
    instruments cannot reveal the quantized nature of
     energy levels. All this is due to smallness of
     Plancks constant h. In the limit h->0, the
```

```
    energy levels become continuous.
17
18 printf("\n(a) Quantum number of the oscillator = %.1
    e\n(b) Fractional change in energy = %.1e\n(c)
    This example illustrates that the energy levels
    of macroscopic oscillators are so close together
     that even most delicate instruments cannot reveal
     the quantized nature of energy levels. All this
    is due to smallness of Plancks constant h. In
    the limit h->0, the energy levels become
    continuous.",n,change_E);
19
20 // The answer given in the book for quantum number
    is just the order of it as it is a very large
    number. But the answer generated by the code is
    the exact value of it.
```

Scilab code Exa 2.1.4 Ex 4

```
1 //variable initialization
2 e=1.6*10^-19;

    //Charge of electron (coulombs)
3 h=(6.625*10^-34)/(2*%pi);

    //Planck's constant (joule-second)
4 c=3*10^8;

    //Speed of light (meter/second)
5
6 //calculating ch by using ch = 2*pi*h*c
7 ch=(2*%pi*h*c*10^9)/e;

    //Value of ch (eV nm)
8
```

```
9 printf("\nch = %.0f eV nm",ch);
```

Scilab code Exa 2.1.5 Ex 5

```
1 //variable initialization
2 h=(6.625*10^-34)/(2*%pi);
     //Planck's constant (joule-second)
3 c=3*10^8;
     //speed of light (meter/second)
4 lembda=2000;
     //Wavelength of the light (  )
5 phi=4.2
     //work function of aluminium surface (eV)
6 ch=12400
     //constant (eV   )
7
8 //(a) maximum kinetic energy of photoelectrons
9 Tmax=(ch/lembda)-phi;
     //maximum kinetic energy of photoelectrons (eV)
10
11 //(b) minimum kinetic energy of photoelectrons
12 Tmin=0
13
14 //(c) cut-off wavelength
15 lembda_cut=ch/phi;
     // cut-off wavelength (  )
16
17 //(d) stopping potential
```

```
18 v=2

    //stopping potential (volt)
19
20 printf("\n(a) Maximum kinetic energy of
    photoelectrons = %.0f eV\n(b) Minimum kinetic
    energy of photoelectrons = %.0f\n(c) Cut-off
    wavelength = %.0f    \n(d) Stopping potential = %
    .0f volt",Tmax,Tmin,lembda_cut,v);
21
22 //Note: In book answer of cut_off wavelength is
    wrong
```

Scilab code Exa 2.1.6 Ex 6

```
1 //variable initialization
2 lembda1=4000

    //wavelength of light (  )
3 V1=2

    //stopping potential (volt)
4 lembda2=6000

    //wavelength of light (  )
5 V2=1

    //stopping potential (volt)
6 e=1.6*10^-19;

    //Charge of electron (coulombs)
7 c=3*10^8;

    //speed of light (meter/second)
8 ch=12400
```

```
    //constant (eV    )
//(i) Planck's constant
h=(e*(V1-V2)*lembda1*10^-10*lembda2*10^-10)/(c*((
    lembda2*10^-10)-(lembda1*10^-10)));  //Planck's
    constant (joule-second)

//(ii) Work function
phi=(ch/lembda1)-V1;

    //work function of the material (eV)

printf("\n(i) Planck's constant = h = %.1e joule-
    second\n(ii) Work function of the material = %.1f
     eV",h,phi);
```

Scilab code Exa 2.1.7 Ex 7

```
//variable initialization
ch=12400

    //constant (eV    )
phi_Tantalum=4.2

    //work function of Tantalum (eV)
phi_Tungsten=4.5

    //work function of Tungsten (eV)
phi_Aluminium=4.2

    //work function of Aluminium (eV)
phi_Barium=2.5

    //work function of Barium (eV)
phi_Lithium=2.3
```

```
    //work function of Lithium (eV)
8
9  //Calculation
10 lembda_Tantalum=ch/phi_Tantalum;
                                                    //
    Threshold wavelength of Tantalum (  )
11 lembda_Tungsten=ch/phi_Tungsten;
                                                    //
    Threshold wavelength of Tungsten (  )
12 lembda_Aluminium=ch/phi_Aluminium;
                                                  //
    Threshold wavelength of Aluminium (  )
13 lembda_Barium=ch/phi_Barium;

    //Threshold wavelength of Barium (  )
14 lembda_Lithium=ch/phi_Lithium;

    //Threshold wavelength of Lithium (  )
15
16 if(lembda_Tantalum < 8000 &  lembda_Tantalum > 4000)
                                    //Checking whether
     Threshold wavelength of Tantalum lies in visible
     range or not
17     disp("Tantalum can be used for designing
         photocell")
18 end
19
20 if(lembda_Tungsten < 8000 & lembda_Tungsten > 4000)
                                   //Checking whether
    Threshold wavelength of Tungsten lies in visible
    range or not
21     disp("Tungsten can be used for designing
         photocell")
22 end
23
24 if(lembda_Aluminium < 8000 &  lembda_Aluminium >
    4000)                              //Checking
```

```
    whether Threshold wavelength of Aluminium lies in
     visible range or not
25      disp("Aluminium can be used for designing
           photocell")
26  end
27
28  if(lembda_Barium < 8000 &  lembda_Barium > 4000)
                                          //Checking
    whether Threshold wavelength of Barium lies in
    visible range or not
29      disp("Barium can be used for designing photocell
           ")
30  end
31
32  if(lembda_Lithium < 8000 & lembda_Lithium > 4000)
                                        //Checking
    whether Threshold wavelength of Lithium lies in
    visible range or not
33      disp("Lithium can be used for designing
           photocell")
34  end
```

Scilab code Exa 2.1.8 Ex 8

```
1  //variable initialization
2  lembda_c=0.024

     //Compton wavelength of electron (  )
3  lembda=1

     //Wavelength of X-rays (  )
4  Theta1=(60*%pi)/180;

     //angle (radian)
5  Theta2=(90*%pi)/180;
```

```
    //angle (radian)
Theta3=(180*%pi)/180;

    //angle (radian)
ch=12400

    //constant (eV    )

//(a) Compton shift
shift1=lembda_c*(1-cos(Theta1));
                                                    //
    Compton shift (   )
shift2=lembda_c*(1-cos(Theta2));
                                                    //
    Compton shift (   )
shift3=lembda_c*(1-cos(Theta3));
                                                    //
    Compton shift (   )

//(b) Kinetic energy imparted to the recoil electron
T1=(ch*shift1)/(lembda*(lembda+shift1));
                                                //Kinetic
    energy imparted to the electron (eV)
T2=(ch*shift2)/(lembda*(lembda+shift2));
                                                //Kinetic
    energy imparted to the electron (eV)
T3=(ch*shift3)/(lembda*(lembda+shift3));
                                                //Kinetic
    energy imparted to the electron (eV)

printf("\n(a) Compton shift:\n\t (i) %.3f   \n\t (ii
    ) %.3f   \n\t (iii) %.3f   \n(b) Kinetic energy
    imparted to the recoil electron:\n\t (i) %.0f eV\
    n\t (ii) %.0f eV\n\t (iii) %.0f eV",shift1,shift2
    ,shift3,T1,T2,T3);
```

Scilab code Exa 2.1.9 Ex 9

```
1 //variable initialization
2 lembda_c=0.024

    //Compton wavelength of electron (  )
3 Theta=(45*%pi)/180;

    //Scattering angle (radian)
4
5 //Calculation of wavelength of incident photon
6 lembda=lembda_c*(1-cos(Theta));

    //Wavelength of incident photon (  )
7
8 printf("\n(a) Wavelength of incident photon = %.4f
      \n(b) Photon lies in the gamma ray spectrum",
    lembda);
```

Scilab code Exa 2.1.10 Ex 10

```
1 //variable initialization
2 E=1

    //Energy of photon (MeV)
3 eta=0.25

    //Relative change in photon's wavelength
4
5 //Calculation of kinetic energy of electron
6 T=(E*eta)/(1+eta);
```

```
    //Kinetic energy of recoil electron (MeV)
7
8 printf("\nThe kinetic energy of recoil electron = %
    .1f MeV",T);
```

Scilab code Exa 2.1.11 Ex 11

```
1 //Variable initialization
2 E=0.25
    //Energy of photon (MeV)
3 Theta=(120*%pi)/180;
    //Scattering angle of photon (radian)
4 a=0.51
    //Value of m0*c^2 (Mev)
5
6 //Calculation of energy of the photon
7 E_das=E/(1+(E/a)*(1-cos(Theta)));
                                        //
    Energy of the scattered photon (MeV)
8
9 printf("\nEnergy of the scattered photon = %.3f Mev"
    ,E_das);
```

Scilab code Exa 2.1.12 Ex 12

```
1 //variable initialization
2 p=1.02
    //momentum of the photon (MeV/c)
```

```
3 p_dash=0.255
    //momentum of the photon after scattering (MeV/c)
4 a=0.51
    //Value of m0*c^2 (Mev)
5
6 //Calculation of the angle of the photon after
    scattering
7 Theta=2*asind(((0.5*a*(p-p_dash))/(p*p_dash))^(1/2))
    ;                                  //Angle of the photon
     after scattering (degree)
8
9 printf("\nAngle of the photon after scattering = %.0
    f degree",Theta);
```

Scilab code Exa 2.1.13 Ex 13

```
1 //variable initialization
2 Theta=120
    //Scattering angle of photon (degree)
3 T=0.45
    //Kinetic energy of electron (MeV)
4 a=0.51
    //Value of m0*c^2 (Mev)
5
6 //Calculation of the energy of the incident photon
7 E=0.5*T*(1+(1+(2*a)/(T*(sind(Theta/2))^2))^(1/2));
                                       //Energy of the
    incident photon (MeV)
8
9 printf("\nEnergy of the incident photon = %.2f Mev",
```

```
E);
```

Scilab code Exa 2.1.14 Ex 14

```
1 //variable initialization
2 lembda0=2536*10^-10;
    //wavelength of exciting line (meter)
3 lembda=2612*10^-10;
    //wavelength of Raman line (meter)
4
5 //Calculation of the Raman shift
6 v0=1/lembda0;
    //wave number of exciting line (1/meter)
7 v=1/lembda;
    //wave number of Raman line (1/meter)
8 shift=v0-v;
    //the Raman shift (1/meter)
9
10 printf("\nThe Raman shift = %.0f m-1",shift);
11
12 //Note: Answer given in the book is an approximate
    answer
```

Scilab code Exa 2.1.15 Ex 15

```
1 //variable initialization
```

```
lembda0=5000*10^-10;

   //Wavelength of radiation (meter)
lembda=5050.5*10^-10;

   //Wavelength of Raman line (meter)

//(a) Raman frequency
v0=1/lembda0;

   //Wave number of radiation (1/meter)
v=1/lembda;

   //Wave number of Raman line (1/meter)
shift=v0-v;

   //Raman shift (1/meter)
va=v0+shift;

   //Frequency of antistoke's line (1/meter)

//(b) Position of the antistokes' line
lembdaa=10^10/va;

   //Wavelength of antistoke's line (  )

printf("\n(a) Raman frequency = %.2e m-1\n(b)
   Wavelength of antistoke's line = %.2f    ",va,
   lembdaa);
```

Chapter 3

Wave Nature of Material Particles

Scilab code Exa 2.2.1 Ex 1

```
1 //variable initialization
2 m=9.1*10^-31;

    //mass of electron (kg)
3 h=6.6*10^-34;

    //planck's constant (joule-second)
4 e=1.6*10^-19;

    //charge of electron (coulomb)
5
6 //calculation of wavelength of electron
7 a=(h*10^10)/(2*m*e)^(1/2);

    //wavelength of electron = h/(2*m*e*v)^(1/2) (  )
8
9 printf("\n Wavelength of electron accelerated
    through a potential difference V = %.1f/V^0.5   "
    ,a);
```

Scilab code Exa 2.2.3 Ex 3

```
1 //variable initialization
2 m_e=9.1*10^-31;
     //mass of electron (kg)
3 m=100*10^-3;
     //mass of object (kg)
4 v=1000
     //velocity of electron and object (meter/second)
5 h=6.63*10^-34;
     //planck's constant (joule-second)
6 //(i) de Broglie wavelength of electron
7 lembda_e=h/(m_e*v);
     //de Broglie wavelength of electron
8
9 //(ii) de Broglie wavelength of object
10 lembda=h/(m*v);
     //de Broglie wavelength of object
11
12 printf("\n(i) de Broglie wavelength of electron = %
     .3e meter\n(ii) de Broglie wavelength of object =
     %.2e meter\nOwing to extremely short wavelength
     of the object its wave behavior cannot be
     demonstrated.",lembda_e,lembda);
13
14 //Note: In the book the answer of part(ii) is wrong
```

Scilab code Exa 2.2.4 Ex 4

```
1 //variable initialization
2 e=1.6*10^-19;
      //charge of electron (coulomb)
3 T=100*e;
      //kinetic energy (joule)
4 m_e=9.1*10^-31;
      //mass of electron (kg)
5 m_p=1.67*10^-27;
      //mass of proton (kg)
6 m_alpha=4*m_p;
      //mass of alpha particle (kg)
7 h=6.63*10^-34;
      //planck's constant (joule-second)
8
9 //calculation of de Broglie wavelengths
10 lembda_e=(h*10^10)/(2*m_e*T)^(1/2);
                                        //de
      Broglie wavelength of electron (  )
11 lembda_p=(h*10^10)/(2*m_p*T)^(1/2);
                                        //de
      Broglie wavelength of proton (  )
12 lembda_alpha=(h*10^10)/(2*m_alpha*T)^(1/2);
                                      //de Broglie
      wavelength of alpha particle (  )
13
14 printf("\nde Broglie wavelength of electron = %.2f
```

```
    \nde Broglie wavelength of proton = %.3f    \nde
    Broglie wavelength of alpha particle = %.3f    ",
    lembda_e,lembda_p,lembda_alpha);
```

Scilab code Exa 2.2.5 Ex 5

```
1 //variable initialization
2 h=6.63*10^-34;

    //planck's constant (joule-second)
3 m=9.1*10^-31;

    //mass of electron (kg)
4 lembda=5896*10^-10;

    //wavelength of yellow spectral line of sodium (
    meter)
5 e=1.6*10^-19;

    //charge of electron (coulomb)
6
7 //calculation of kinetic energy of electron
8 T_j=h^2/(2*m*lembda^2);

    //kinetic energy of the electron (joule)
9 T_eV=T_j/e;

    //kinetic energy of the electron (eV)
10
11 printf("\n kinetic energy of electron = %.2e joule =
    %.1e eV",T_j,T_eV);
```

Scilab code Exa 2.2.6 Ex 6

```
1 //variable initialization
2 h=6.63*10^-34;

    //planck's constant (joule-second)
3 m_n=1.67*10^-27;

    //mass of neutron (kg)
4 T=300

    //Temperature (kelvin)
5 k=1.38*10^-23;

    //Boltzmann constant (joule/kelvin)
6
7 //calculation of the wavelength of thermal neutron
8 E=(3*k*T)/2;

    //Kinetic energy of thermal neutron (joule)
9 lembda=(h*10^10)/(2*m_n*E)^(1/2);
                                                    //
    Wavelength of thermal neutron (  )
10
11 printf("\nThe wavelength of thermal neutron = %.2f
     ",lembda);
```

Scilab code Exa 2.2.7 Ex 7

```
1 //variable initialization
2 h=6.63*10^-34;

    //planck's constant (joule-second)
3 m_H2=2*1.67*10^-27;

    //mass of hydrogen molecule (kg)
4 T=27+273;
```

```
    //room temperature (kelvin)
5 k=1.38*10^-23;

    //Boltzmann constant (joule/kelvin)
6
7 //calculation of de Broglie wavelength of hydrogen
    molecule
8 lembda=(h*10^10)/(2*m_H2*k*T)^(1/2);
                                                    //de
    Broglie wavelength of hydrogen molecule (  )
9
10 printf("\nThe de Broglie wavelength of hydrogen
    molecules at their most probable speed = %.2f    "
    ,lembda);
```

Scilab code Exa 2.2.8 Ex 8

```
1 //variable initialization
2 a=0.51

    //Value of m0*c^2 (Mev)
3
4 //Calculation of kinetic energy of electron
5 T=a*(sqrt(2)-1);

    //Kinetic energy (MeV)
6
7 printf("\n Kinetic energy of electron = %.2f MeV",T)
    ;
```

Scilab code Exa 2.2.9 Ex 9

```
1 //variable initialization
2 a=0.51

    //Value of m0*c^2 (MeV)
3 b=0.0124

    //Value of h*c (MeV    )
4 lembda_X=0.1

    //Short wavelength limit of continuous X-ray
    spectrum (   )
5
6 //calculation of de Broglie wavelength of electron
7 lembda=lembda_X/(1+(2*a*lembda_X)/b)^(1/2);
                                                    //de
    Broglie wavelength of relativistic electrons
8
9 printf("de Broglie wavelength of relativistic
    electrons = %.3f    ",lembda);
```

Scilab code Exa 2.2.10 Ex 10

```
1 //variable initialization
2 r=0.53

    //Radius of the first Bohr orbit in hydrogen atom
     (   )
3
4 //calculation of de Broglie wavelength of electron
5 lembda=2*%pi*r;

    //de Broglie wavelength of electron in first Bohr
     orbit in hydrogen atom
6
7 printf("\nde Broglie wavelength of electron in first
```

```
Bohr orbit in hydrogen atom = %.1f    ",lembda);
```

Scilab code Exa 2.2.12 Ex 12

```
1 //variable initialization
2 v=10000
    //speed of object (meter/second)
3 accu_v=0.0001
    //accuracy of speed of object
4 m_b=0.05
    //mass of the bullet (kg)
5 h=1.054*10^-34;
    //planck's constant (joule-second)
6 m_e=9.1*10^-31;
    //mass of electron (kg)
7
8 //(a) fundamental accuracy of position for bullet
9 p_b=m_b*v;
    //momentum of bullet (kg m/s)
10 p_uncer_b=p_b*accu_v;
    //uncertainty in momentum of bullet (kg m/s)
11 x_uncer_b=h/p_uncer_b;
    //minimum uncertainty in position of bullet (
    meter)
12
13 //(b) fundamental accuracy of position for electron
14 p_e=m_e*v;
```

```
    //momentum of electron (kg m/s)
15 p_uncer_e=p_e*accu_v;

    //uncertainty in momentum of electron (kg m/s)
16 x_uncer_e=h/p_uncer_e;

    //uncertainty in position of electron (meter)
17
18 printf("\n(a) uncertainty in position of bullet = %
    .1e meter\n(b) uncertainty in position of
    electron = %.6f meter\n   The uncertainty in
    bullets position is so small that it is far
    beyond the possibility of measurement.",x_uncer_b
    ,x_uncer_e);
19
20 //Note:The answers given in the book are wrong. Also
     in the solution they have used speed=1000 while
    in the question it is given to be equal to 10000.
```

Scilab code Exa 2.2.13 Ex 13

```
1 //variable initialization
2 h=1.054*10^-34;

    //planck's constant (joule-second)
3 m=9.1*10^-31;

    //mass of electron (kg)
4 x_uncer=1*10^-10;

    //uncertainty in the position of elctrons (meter)
5 e=1.6*10^-19;

    //charge of electron (coulomb)
```

```
6
7 //(i) uncertainty in the momentum of electron
8 p_uncer=h/x_uncer;

     //The uncertainty in the momentum of electron (kg
     m/s)
9
10 //(ii) kinetic energy of electron
11 T=p_uncer^2/(2*m*e);

     //kinetic energy of electron (eV)
12
13 printf("\n(i) The uncertainty in the momentum of
     electron = %.3e kg m/s\n(ii) Kinetic energy of
     electron = %.1f eV\n   The ionization potential
     of atoms is of this order and hence the
     uncertainty in momentum is consistence with the
     binding energy of electrons in atoms.",p_uncer,T)
     ;
```

Scilab code Exa 2.2.14 Ex 14

```
1 //variable initialization
2 h=1.054*10^-34;

     //planck's constant (joule-second)
3 x=10^-14;

     //dimension of the nucleus (meter)
4 c=3*10^8;

     //speed of light (meter/second)
5 e=1.6*10^-19;

     //charge of electron (coulomb)
```

```
6
7 //(i) Uncertainty in the momentum of electron
8 p_uncer=h/x;

     //The uncertainty in the momentum of electron (kg
     m/s)
9
10 //(ii) kinetic energy of electron
11 T=(p_uncer*c)/(e*10^6);

     //kinetic energy of electron (MeV)
12
13 printf("\n(i) The uncertainty in the momentum of
     electron = %.3e kg m/s\n(ii) Kinetic energy of
     electron = %.0f MeV\n   Experiments show that
     energy of electrons in nuclear disintegration (
     beta decay) is very much less than %.0f MeV.
     Hence the uncertainty principle rules out the
     possibility of electrons being a nuclear
     constituent.",p_uncer,T,T);
```

Scilab code Exa 2.2.15 Ex 15

```
1 //variable initialization
2 h=1.054*10^-34;

     //planck's constant (joule-second)
3 x=10^-14;

     //dimension of the nucleus (meter)
4 e=1.6*10^-19;

     //charge of electron (coulomb)
5 m=1.67*10^-27;
```

```
    //mass of proton (kg)
6
7 //(i) Uncertainty in the momentum of electron
8 p_uncer=h/x;

    //The uncertainty in the momentum of electron (kg
     m/s)
9
10 //(ii) kinetic energy of proton
11 T=(p_uncer^2)/(2*m*e*10^6);

    //kinetic energy of proton (MeV)
12
13 printf("\n(i) The uncertainty in the momentum of
    electron = %.3e kg m/s\n(ii) Kinetic energy of
    proton = %.2f MeV\n   The binding energies of
    nuclei are of this order.",p_uncer,T);
```

Scilab code Exa 2.2.17 Ex 17

```
1 //variable initialization
2 h=1.054*10^-34;

    //planck's constant (joule-second)
3 delta_t=10^-12;

    //time for which nucleus remains in excited state
     (second)
4
5 //calculation of uncertainty in the energy of the
    gamma ray photon emitted by the nucleus
6 delta_E=h/delta_t;

    //uncertainty in the energy of the gamma ray
    photon (joule)
```

```
7
8 printf("\nThe uncertainty in the energy of the gamma
      ray photon = %.3e joule",delta_E);
```

Scilab code Exa 2.2.18 Ex 18

```
1 //variable initialization
2 delta_t=10^-8;

     //life-time of the average excited atom (second)
3
4 //calculation of the minimum uncertainty in the
     frequency of photon
5 delta_f=1/delta_t;

     //minimum uncertainty in the frequency of photon
     (radian/second)
6
7 printf("\nminimum uncertainty in the frequency of
     photon = %.0e radian/second",delta_f);
```

Scilab code Exa 2.2.19 Ex 19

```
1 //variable initialization
2 h=1.054*10^-34;

     //planck's constant (joule-second)
3 e=1.6*10^-19;

     //charge of electron (coulomb)
4 m=9.1*10^-31;

     //mass of electron (kg)
```

```
5  E0=8.8542*10^-12;
       //permittivity of free space (C^2/N*m^2)
6
7  //(i) radius of ground state of hydrogen atom
8  r=(4*%pi*E0*h^2)/(m*e^2);
       //radius of ground state of hydrogen atom (meter)
9
10 //(ii) Binding energy of electron in hydrogen atom
       in the ground state
11 E=(-0.5*m*e^4)/(4*%pi*E0*h)^2;
       //binding energy of electron in hydrogen atom in
       the ground state (joule)
12
13 printf("\n(i) Radius of ground state of hydrogen
       atom = %.3e meter\n(ii) Binding energy of
       electron in ground state of hydrogen atom = %.2e
       joule",r,E);
```

Chapter 7

Particle in a Box

Scilab code Exa 2.6.1 Ex 1

```
1 //variable initialization
2 x=(1/3):.01:(2/3);
3 x0=1/3;
4 x1=2/3;
5
6 //calculation
7 P=2*integrate('(sin(%pi*x))^2','x',x0,x1);
8
9 printf("The required probability = %.2f",P);
```

Scilab code Exa 2.6.2 Ex 2

```
1 //variable initialization
2 x=0:.001:(1/3);
3 y=0:.001:(1/3);
4 x0=0;
5 x1=1/3;
6 y0=0;
```

```
y1=1/3;

p=4*integrate('(sin(%pi*x))^2','x',x0,x1)*integrate(
    '(sin(%pi*y))^2','y',y0,y1);

printf("The required probability = %.2f",p);
```

Scilab code Exa 2.6.3 Ex 3

```
//variable initialization
L = 10;
n = 2;

//Calculation
X = L/2;
X_square = ((L^2)/3)-((L^2)/(2*(n^2)*(3.14^2)));

printf("\nExpectation of X = %.2f",X);
printf("\nExpectation of X^2 = %.2f",X_square);
```

Chapter 10

Particle in a Central Force Field

Scilab code Exa 2.9.2 Ex 2

```
1 //variable initialization
2 x=0:0.1:9999;
3 x0=0;
4 x1=9999;
5
6 //calculation
7 I=integrate('x^2*exp(-x)','x',x0,x1);
8 A=sqrt(1/(I*(%pi/2)));
9 r=(1/4)*integrate('x^3*exp(-x)','x',x0,x1);
10
11 printf("\n A = %f*a0^-1.5\n r = %.1f*a0",A,r);
```

Scilab code Exa 2.9.3 Ex 3

```
1 //variable initialization
2 r=0:.01:1;
```

```
3 r0=0;
4 r1=1;
5
6 //calculation
7 P=4*integrate('r^2*exp(-2*r)','r',r0,r1);
8
9 printf("\n Probability = %.2f",P);
```

Scilab code Exa 2.9.4 Ex 4

```
1 //variable initialization
2 r=0:.01:9999;
3 theta=0:.01:%pi;
4 phi=0:.01:2*%pi;
5
6 //calculation
7 I1=integrate('r^4*exp(-r)','r',0,9999);
8 I2=integrate('sin(theta)*(cos(theta))^2','theta',0,
    %pi);
9 I3=integrate('1','phi',0,2*%pi);
10 A=sqrt(1/(2*%pi*I1*I2*I3));
11
12 printf("\n A = %f*a0^(-5/2)",A);
```

Chapter 11

Preliminary Concepts

Scilab code Exa 3.1.1 Ex 1

```
1 //variable initialization
2 N=2
     //no. of particles
3 n1=2
     //occupation no. of particles
4 g1=3
     //degeneracy of particles
5
6 //(i) particles are distinguishable
7 state1=(factorial(N)*g1^n1)/factorial(n1);
                                          //possible
     microstates of distinguishable particles
8
9 //(ii) particles are indistinguishable bosons
10 state2=factorial(n1+g1-1)/(factorial(n1)*factorial(
     g1-1));                          //possible
     microstates of indistinguishable bosons
11
```

```
12 //(iii) particles are indistinguishable fermions
13 state3=factorial(g1)/(factorial(n1)*factorial(g1-n1)
   );                              //possible
   microstates of indistinguishable fermions
14
15 printf("\n(i)   (distinguishable) = %.0f\n(ii)   (
   indistinguishable bosons) = %.0f\n(iii)   (
   indistinguishable fermions) = %.0f",state1,state2
   ,state3);
```

Scilab code Exa 3.1.2 Ex 2

```
1 //variable initialization
2 N=4
     //no. of particles
3 A1=[2 0 0 2];
     //possible macrostate
4 A2=[1 1 1 1];
     //possible macrostate
5 A3=[0 3 0 1];
     //possible macrostate
6 A4=[1 0 3 0];
     //possible macrostate
7 A5=[0 2 2 0];
     //possible macrostate
8 g1=1
     //degeneracy of particles
9 g2=2
```

```
    //degeneracy of particles
g3=2

    //degeneracy of particles
g4=1

    //degeneracy of particles

//(i) particles are distinguishable
printf("\n(i)Possible macrostates are\n ");
disp(A5,A4,A3,A2,A1);
micro1=((factorial(N)*g1^A1(1)*g2^A1(2)*g3^A1(3)*g4^
    A1(4))/(factorial(A1(1))*factorial(A1(2))*
    factorial(A1(3))*factorial(A1(4))));     //The
    number of microstates
micro2=((factorial(N)*g1^A2(1)*g2^A2(2)*g3^A2(3)*g4^
    A2(4))/(factorial(A2(1))*factorial(A2(2))*
    factorial(A2(3))*factorial(A2(4))));     //The
    number of microstates
micro3=((factorial(N)*g1^A3(1)*g2^A3(2)*g3^A3(3)*g4^
    A3(4))/(factorial(A3(1))*factorial(A3(2))*
    factorial(A3(3))*factorial(A3(4))));     //The
    number of microstates
micro4=((factorial(N)*g1^A4(1)*g2^A4(2)*g3^A4(3)*g4^
    A4(4))/(factorial(A4(1))*factorial(A4(2))*
    factorial(A4(3))*factorial(A4(4))));     //The
    number of microstates
micro5=((factorial(N)*g1^A5(1)*g2^A5(2)*g3^A5(3)*g4^
    A5(4))/(factorial(A5(1))*factorial(A5(2))*
    factorial(A5(3))*factorial(A5(4))));     //The
    number of microstates
printf("\nMost probable macrostates are\n ");
if(micro1>=micro2 & micro1>=micro3 & micro1>=micro4
    & micro1>=micro5) then
      disp(A1);
end
if(micro2>=micro1 & micro2>=micro3 & micro2>=micro4
```

```
    & micro2>=micro5) then
26      disp(A2);
27  end
28  if(micro3>=micro1 & micro3>=micro2 & micro3>=micro4
    & micro3>=micro5) then
29      disp(A3);
30  end
31  if(micro4>=micro1 & micro4>=micro2 & micro4>=micro3
    & micro4>=micro5) then
32      disp(A4);
33  end
34  if(micro5>=micro1 & micro5>=micro2 & micro5>=micro3
    & micro5>=micro4) then
35      disp(A5);
36  end
37
38  //(ii) particles are indistinguishable bosons
39  printf("\n(ii)Possible macrostates are\n ");
40  disp(A5,A4,A3,A2,A1);
41  micro1=(factorial(A1(1)+g1-1)*factorial(A1(2)+g2-1)*
    factorial(A1(3)+g3-1)*factorial(A1(4)+g4-1))/(
    factorial(A1(1))*factorial(A1(2))*factorial(A1(3)
    )*factorial(A1(4))*factorial(g1-1)*factorial(g2
    -1)*factorial(g3-1)*factorial(g4-1));
42  micro2=(factorial(A2(1)+g1-1)*factorial(A2(2)+g2-1)*
    factorial(A2(3)+g3-1)*factorial(A2(4)+g4-1))/(
    factorial(A2(1))*factorial(A2(2))*factorial(A2(3)
    )*factorial(A2(4))*factorial(g1-1)*factorial(g2
    -1)*factorial(g3-1)*factorial(g4-1));
43  micro3=(factorial(A3(1)+g1-1)*factorial(A3(2)+g2-1)*
    factorial(A3(3)+g3-1)*factorial(A3(4)+g4-1))/(
    factorial(A3(1))*factorial(A3(2))*factorial(A3(3)
    )*factorial(A3(4))*factorial(g1-1)*factorial(g2
    -1)*factorial(g3-1)*factorial(g4-1));
44  micro4=(factorial(A4(1)+g1-1)*factorial(A4(2)+g2-1)*
    factorial(A4(3)+g3-1)*factorial(A4(4)+g4-1))/(
    factorial(A4(1))*factorial(A4(2))*factorial(A4(3)
    )*factorial(A4(4))*factorial(g1-1)*factorial(g2
```

```
    -1)*factorial(g3-1)*factorial(g4-1));
45 micro5=(factorial(A5(1)+g1-1)*factorial(A5(2)+g2-1)*
    factorial(A5(3)+g3-1)*factorial(A5(4)+g4-1))/(
    factorial(A5(1))*factorial(A5(2))*factorial(A5(3)
    )*factorial(A5(4))*factorial(g1-1)*factorial(g2
    -1)*factorial(g3-1)*factorial(g4-1));
46 printf("\nMost probable macrostate is\n ");
47 if(micro1>=micro2 & micro1>=micro3 & micro1>=micro4
    & micro1>=micro5) then
48     disp(A1);
49 end
50 if(micro2>=micro1 & micro2>=micro3 & micro2>=micro4
    & micro2>=micro5) then
51     disp(A2);
52 end
53 if(micro3>=micro1 & micro3>=micro2 & micro3>=micro4
    & micro3>=micro5) then
54     disp(A3);
55 end
56 if(micro4>=micro1 & micro4>=micro2 & micro4>=micro3
    & micro4>=micro5) then
57     disp(A4);
58 end
59 if(micro5>=micro1 & micro5>=micro2 & micro5>=micro3
    & micro5>=micro4) then
60     disp(A5);
61 end
62
63 //(iii) Particles are indistinguishable fermions
64 printf("\n(iii)Possible macrostates are\n ");
65 disp(A5,A2);
66 micro2=4/(factorial(A2(1))*factorial(A2(2))*
    factorial(A2(3))*factorial(A2(4))*factorial(g1-A2
    (1))*factorial(g2-A2(2))*factorial(g3-A2(3))*
    factorial(g4-A2(4)));
67 micro5=4/(factorial(A5(1))*factorial(A5(2))*
    factorial(A5(3))*factorial(A5(4))*factorial(g1-A5
    (1))*factorial(g2-A5(2))*factorial(g3-A5(3))*
```

```
    factorial(g4-A5(4)));
printf("\nMost probable macrostate is\n ");
if(micro2>=micro5) then
    disp(A2);
end
if(micro5>=micro2) then
    disp(A5);
end
```

Scilab code Exa 3.1.3 Ex 3

```
//vairable initialization
N=4
    //no. of particles
A1=[4 0];
    //possible macrostate
A2=[3 1];
    //possible macrostate
A3=[2 2];
    //possible macrostate
A4=[1 3];
    //possible macrostate
A5=[0 4];
    //possible macrostate

//calculation
printf("\nPossible macrostates are\n ");
disp(A5,A4,A3,A2,A1);
micro1=factorial(N)/(factorial(A1(1))*factorial(A1
```

```
    (2)));                          //no. of microstate
     corresponding to macrostate1
micro2=factorial(N)/(factorial(A2(1))*factorial(A2
    (2)));                          //no. of microstate
     corresponding to macrostate2
micro3=factorial(N)/(factorial(A3(1))*factorial(A3
    (2)));                          //no. of microstate
     corresponding to macrostate3
micro4=factorial(N)/(factorial(A4(1))*factorial(A4
    (2)));                          //no. of microstate
     corresponding to macrostate4
micro5=factorial(N)/(factorial(A5(1))*factorial(A5
    (2)));                          //no. of microstate
     corresponding to macrostate5
printf("\nTotal no. of microstates are %.0f",micro1+
    micro2+micro3+micro4+micro5);
printf("\nMost probable macrostate is\n ");
if(micro1>=micro2 & micro1>=micro3 & micro1>=micro4
    & micro1>=micro5) then
    disp(A1);
end
if(micro2>=micro1 & micro2>=micro3 & micro2>=micro4
    & micro2>=micro5) then
    disp(A2);
end
if(micro3>=micro1 & micro3>=micro2 & micro3>=micro4
    & micro3>=micro5) then
    disp(A3);
end
if(micro4>=micro1 & micro4>=micro2 & micro4>=micro3
    & micro4>=micro5) then
    disp(A4);
end
if(micro5>=micro1 & micro5>=micro2 & micro5>=micro3
    & micro5>=micro4) then
    disp(A5);
end
```

Scilab code Exa 3.1.4 Ex 4

```
1 //variable initialization
2 N=4

    //no. of particles
3 A1=[1 0 1 2];

    //possible macrostate
4 A2=[0 2 0 2];

    //possible macrostate
5 A3=[0 1 2 1];

    //possible macrostate
6 A4=[0 0 4 0];

    //possible macrostate
7
8 //calculation
9 printf("\nPossible macrostates are\n ");
10 disp(A4,A3,A2,A1);
11 micro1=factorial(N)/(factorial(A1(1))*factorial(A1
    (2))*factorial(A1(3))*factorial(A1(4)));
                          //no. of microstate
    corresponding to macrostate1
12 micro2=factorial(N)/(factorial(A2(1))*factorial(A2
    (2))*factorial(A2(3))*factorial(A2(4)));
                          //no. of microstate
    corresponding to macrostate2
13 micro3=factorial(N)/(factorial(A3(1))*factorial(A3
    (2))*factorial(A3(3))*factorial(A3(4)));
                          //no. of microstate
    corresponding to macrostate3
```

```
14 micro4=factorial(N)/(factorial(A4(1))*factorial(A4
   (2))*factorial(A4(3))*factorial(A4(4)));
                          //no. of microstate
   corresponding to macrostate4
15 printf("\nThe number of microstates belonging to the
    above macrostates is:%.0f, %.0f, %.0f, %.0f",
   micro1,micro2,micro3,micro4);
```

Scilab code Exa 3.1.5 Ex 5

```
1 function[probability]=p(A)

   //function to calculate probability
2     probability=1;
3     i=1
4     for i=1:7
5         probability=probability*(factorial(A(i)+2)
             /(2*factorial(A(i))));
6     end
7 endfunction
8
9 //variable initialization
10 A1=[5 0 0 0 0 0 1];

   //possible macrostate
11 A2=[4 1 0 0 0 1 0];

   //possible macrostate
12 A3=[4 0 1 0 1 0 0];

   //possible macrostate
13 A4=[3 2 0 0 1 0 0];

   //possible macrostate
14 A5=[4 0 0 2 0 0 0];
```

```
    //possible macrostate
15 A6=[3 1 1 1 0 0 0];

    //possible macrostate
16 A7=[2 3 0 1 0 0 0];

    //possible macrostate
17 A8=[3 0 3 0 0 0 0];

    //possible macrostate
18 A9=[2 2 2 0 0 0 0];

    //possible macrostate
19 A10=[1 4 1 0 0 0 0];

    //possible macrostate
20 A11=[0 6 0 0 0 0 0];

    //possible macrostate
21
22 //calculation
23 p1=p(A1);

    //Thermodynamic probability of macrostate 1
24 p2=p(A2);

    //Thermodynamic probability of macrostate 2
25 p3=p(A3);

    //Thermodynamic probability of macrostate 3
26 p4=p(A4);

    //Thermodynamic probability of macrostate 4
27 p5=p(A5);

    //Thermodynamic probability of macrostate 5
28 p6=p(A6);
```

```
    //Thermodynamic probability of macrostate 6
29 p7=p(A7);

    //Thermodynamic probability of macrostate 7
30 p8=p(A8);

    //Thermodynamic probability of macrostate 8
31 p9=p(A9);

    //Thermodynamic probability of macrostate 9
32 p10=p(A10);

    //Thermodynamic probability of macrostate 10
33 p11=p(A11);

    //Thermodynamic probability of macrostate 11
34
35 printf("\nP1 = %.0f, P2 = %.0f, P3 = %.0f, P4 = %.0f
    , P5 = %.0f, P6 = %.0f, P7 = %.0f, P8 = %.0f, P9
    = %.0f, P10 = %.0f, P11 = %.0f",p1,p2,p3,p4,p5,p6
    ,p7,p8,p9,p10,p11);
36 printf("\nThermodyanmic probability of the system =
    %.0f",p1+p2+p3+p4+p5+p6+p7+p8+p9+p10+p11);
```

Scilab code Exa 3.1.6 Ex 6

```
1 function[micro]=p(A)

    //function to calculate no. of microstates
2     micro=1;
3     i=1
4     for i=1:5
5         micro=micro*(6/(factorial(A(i))*factorial(3-
              A(i))));
```

```
6       end
7   endfunction
8
9   //variable initialization
10  A1=[3 2 0 0 1];

        //possible macrostate
11  A2=[3 1 1 1 0];

        //possible macrostate
12  A3=[2 3 0 1 0];

        //possible macrostate
13  A4=[3 0 3 0 0];

        //possible macrostate
14  A5=[2 2 2 0 0];

        //possible macrostate
15
16  //calculation
17  p1=p(A1);

        //no. of microstates
18  p2=p(A2);

        //no. of microstates
19  p3=p(A3);

        //no. of microstates
20  p4=p(A4);

        //no. of microstates
21  p5=p(A5);

        //no. of microstates
22
23  printf("\nPossible microstates are : %.0f, %.0f, %.0
```

```
    f, %.0f, %.0f",p1,p2,p3,p4,p5);
24 printf("\nThe thermodynamic probability of the
    system = %.0f",p1+p2+p3+p4+p5);
```

Chapter 18

Atomic Spectra I

Scilab code Exa 4.1.1 Ex 1

```
1 //variable initialization
2 z=2

    //atomic no. of He
3 a0=0.529

    //radius of first Bohr orbit of H atom (  )
4 n=1

    //no. of Bohr orbit
5 A=2.19*10^6;

    //velocity of e in first Bohr orbit of H atom (m/
    s)
6 B=4.14*10^15;

    //orbital frequency in the first Bohr orbit of H
    atom (rad/s)
7 E0=13.6

    //energy of electron in ground state of H atom (
```

```
    eV)
n1=1;
n2=2;
R=1.097*10^7;
    //Rydberg constant (m-1)

//(i) radius of first Bohr orbit
r=a0/2;
    //radius of first Bohr  orbit (  )

//(ii) velocity of electron in the first orbit
v=A*(z/n);
    //velocity of electron in the first orbit (m/s)

//(iii) orbital frequency in the first orbit
omega=B*(z^2/n^3);
    //orbital frequency in the first orbit (rad/s)

//(iv) kinetic and binding energy
KE=E0*(z^2/n^2);
    //kinetic energy of electron in the ground state
    (eV)
BE=KE;
    //binding energy of electron in the ground state
    (eV)

//(v) ionization potential and first excitation
    potential
IP=KE;
    //ionization potential (eV)
EE=E0*z^2*((1/n1^2)-(1/n2^2));
```

```
    //first excitation potential (eV)
28
29 //(vi) wavelength of the resonance line emitted in
    the transition n=2 to n=1
30 lembda=(1/(R*z^2*((1/n1^2)-(1/n2^2))))*10^10;
                                        //wavelength of
    the resonance line emitted in the transition n=2
    to n=1 (  )
31
32 printf("\n(i) radius = %.3f   \n(ii) velocity = %.2e
    m/s\n(iii) orbital frequency = %.3e rad/s\n(iv)
   KE = %.2f eV    BE = %.2f eV\n(v) IP = %.2f eV
      EE = %.2f eV\n(vi) wavelength = %.1f   ",r,v,
   omega,KE,BE,IP,EE,lembda);
```

Scilab code Exa 4.1.2 Ex 2

```
1 //variable initialization
2 z=1

    //atomic no. of H atom
3 m=1.68*10^-27;

    //mass of H atom (kg)
4 h=1.054*10^-34;

    //Planck's constant (joule second)
5 R=10967800;

    //Rydberg constant (m-1)
6 e=1.6*10^-19;

    //Charge of electron (coulombs)
7 c=3*10^8;
```

```
    //speed of light (m/s)
8
9  //(i) recoil velocity
10 v=(3*%pi*h*R*z^2)/(2*m);

    //recoil velocity of H atom (m/s)
11
12 //(ii) recoil kinetic energy
13 Er=(9/8)*((%pi*h*R*z^2)^2/(m*e));
                                                    //
    recoil kinetic energy of H atom (eV)
14
15 //(iii) energy of emitted photon
16 E=(1.5*%pi*h*c*R*z^2)/e;

    //energy of emitted photon (eV)
17
18 printf("\n(i) recoil velocity = %.2f m/s\n(ii)
    recoil kinetic energy = %.1e eV\n(iii) energy of
    emitted photon = %.2f eV",v,Er,E);
```

Scilab code Exa 4.1.3 Ex 3

```
1  //variable initialization
2  z=2

    //atomic no. of He atom
3  h=1.054*10^-34;

    //Planck's constant (joule second)
4  R=10967800;

    //Rydberg constant (m-1)
5  e=1.6*10^-19;
```

```
    //Charge of electron (coulombs)
c=3*10^8;

    //speed of light (m/s)

//calculation
E=1.5*%pi*h*c*R*z^2;

    //The energy of the emitted photon (J)
IE=2*%pi*h*c*R;

    //Ionization energy of H atom (J)
KE=(E-IE)/e;

    //Kinetic energy of the photoelectron (eV)

printf("\nKinetic energy of photoelectron = %.1f eV"
    ,KE);
```

Scilab code Exa 4.1.4 Ex 4

```
//variable initialization
ratio=4

    //ratio of wavelengths
z1=1

    //atomic no. of hydrogen atom

//calculation
z2=sqrt(ratio*z1^2);

    //atomic no. of unknown element

```

```
8 printf("\natomic no. = %.0f",z2);
```

Scilab code Exa 4.1.5 Ex 5

```
1 //variable initialization
2 lembda1=108.5*10^-9;
     //wavelength (m)
3 lembda2=30.4*10^-9;
     //wavelength (m)
4 R=1.097*10^7;
     //Rydberg constant (m-1)
5 z=2
     //atomic no. of He
6 n0=1
     //ground state
7
8 //calculation
9 n=sqrt(1/((1/n0^2)-(((1/lembda1)+(1/lembda2))/(R*z
     ^2))));                          //quantum no.
     corresponding to the excited state of He+
10
11 printf("\nn = %.0f",n);
```

Scilab code Exa 4.1.6 Es 6

```
1 //variable initialization
```

```
2 z=2

    //atomic no. of He+ ion
3 lembda=133.7*10^-9;

    //difference b/w the first lines of the Balmer
    and Lyman series (m)
4 n1=1
5 n2=2
6 n3=3
7
8 //calculation
9 R=(1/(lembda*z^2))*((1/((1/n2^2)-(1/n3^2)))-(1/((1/
    n1^2)-(1/n2^2))));              //Rudberg constant (
    m-1)
10
11 printf("\nR = %.3e m-1",R);
```

Scilab code Exa 4.1.7 Ex 7

```
1 //variable initialization
2 R=1.097*10^7;

    //Rydberg constant (m-1)
3 lembda=59.3*10^-9;

    //wavelength difference b/w first lines of Balmer
     and Lyman series (m)
4
5 //calculation
6 z=sqrt(88/(15*R*lembda));

    //atomic no.
7
8 printf("\nZ = %.0f",z);
```

Scilab code Exa 4.1.8 Ex 8

```
1 //variable initialization
2 R=1.097*10^7;
     //Rydberg constant (m-1)
3 ratio=1836;
     //ratio of maas of tritium and hydrogen
4
5 //calculation
6 lembda=(36*2*10^10)/(5*R*3*ratio);
                                                    //
     separation of the first line of the Balmer series
      (  )
7
8 printf("\n     = %.1f    ",lembda);
```

Chapter 19

Atomic Spectra II

Scilab code Exa 4.2.4 Ex 4

```
1  //variable initialization
2  rP = 4;
3  rD = 5;
4  LP = 1;
5  LP = 2;
6  jP = [5/2, 3/2, 1/2];
7  jD = [4, 3, 2, 1, 0];
8
9  //Calculation
10 SP = (rP-1)/2;
11 SD = (rD-1)/2;
12 i=1;
13 for i=1:3
14     JP(i) = sqrt(jP(i)*(jP(i)+1));
15 end
16 i=1;
17 for i=1:5
18     JD(i) = sqrt(jD(i)*(jD(i)+1));
19 end
20
21 printf("\nAngular moments allowed for 4P : %.2f",JP)
```

```
    ;
22 printf("\nAngular moments allowed for 5D : %.2f",JD)
    ;
```

Scilab code Exa 4.2.9 Ex 9

```
1 //variable initialization
2 l=1
3 s=1/2
4 j=3/2
5
6 //calculation
7 angle=((j*(j+1))-(l*(l+1))-(s*(s+1)))/(2*sqrt(l*s*(l
    +1)*(s+1)));                    //value of cos
8
9 printf("\n cos     = %.3f",angle);
```

Scilab code Exa 4.2.10 Ex 10

```
1 //variable initialization
2 e=1.6*10^-19;

    //charge of electron (C)
3 m=9.1*10^-31;

    //mass of electron (kg)
4 B=0.1

    //external magnetic field (Wb/m^2)
5 g=4/3
6 mu=9.27*10^-24;

    //(J/T)
```

```
7
8 //calculation
9 E=g*mu*B;

    //The spacing of adjacent sub-levels (J)
10 v=(e*B)/(4*%pi*m);

    //Larmor frequency (Hz)
11
12 printf("\n The spacing of adjacent sub-levels = %e J
    \n Larmor frequency = %.1e Hz",E,v);
```

Scilab code Exa 4.2.11 Ex 11

```
1 //variable initialization
2 mu=9.27*10^-24;

    //(J/T)
3 g=2;
4 ms=1/2;
5 dB=2*10^2;

    //gradient of magnetic field (T/m)
6 m=1.67*10^-27;

    //maas of hydrogen atom (kg)
7 l=0.2;

    //distance travelled by hydrogen atom (m)
8 v=2*10^5;

    //speed of hydrogen atom (m/s)
9
10 //calculation
11 muz=g*mu*ms;
```

```
    //Resolved part of magnetic moment in the
    direction of magnetic field (J/T)
12 Fz=muz*dB;

    //Force on the atom (N)
13 z=0.5*(Fz/m)*(l/v)^2;

    //Displacement of beam (m)
14 sep=2*z;

    //Total separation (m)
15
16 printf("\n Total separation = %.2e m",sep);
```

Scilab code Exa 4.2.12 Ex 12

```
1  //variable initialization
2  l=1
3  s=1/2
4  j1=1/2
5  j2=3/2
6
7  //calculation
8  L=sqrt(l*(l+1));

    //orbital angular momenta
9  S=sqrt(s*(s+1));

    //spin angular momenta
10 J1=sqrt(j1*(j1+1));

    //total angular momenta
11 J2=sqrt(j2*(j2+1));
```

```
    //total angular momenta
12 theta1=(180/%pi)*acos(((j1*(j1+1))-(l*(l+1))-(s*(s
    +1)))/(2*sqrt(l*(l+1))*sqrt(s*(s+1))));     //
    angle b/w l and s (degree)
13 theta2=(180/%pi)*acos(((j2*(j2+1))-(l*(l+1))-(s*(s
    +1)))/(2*sqrt(l*(l+1))*sqrt(s*(s+1))));     //
    angle b/w l and s (degree)
14
15 printf("\n L = %f*h\n S = %f*h\n J = %f*h, %f*h\n
     1  = %.0f degree\n  2  = %.0f degree",L,S,J1,J2,
    theta1,theta2);
```

Scilab code Exa 4.2.13 Ex 13

```
1 //variable initialization
2 B=0.5

    //magnetic field (T)
3 s=1/2;
4 g=2;
5
6 //calculation
7 S=sqrt(s*(s+1));

    //Magnitude of spin vector
8 theta1=(180/%pi)*acos(0.5/S);

    //Orientation of spin vector (degree)
9 theta2=(180/%pi)*acos(-0.5/S);

    //Orientation of spin vector (degree)
10 E=2*g*B;

    //Separation of the energy levels (in terms of
        )
```

```
11
12 printf("\n     = %.1f degree and %.1f degree\n  E  =
     %.0f*     ",theta1,theta2,E);
```

Scilab code Exa 4.2.15 Ex 15

```
1 //variable initialization
2 zH=1
     //atomic no. of H
3 zHe=2
     //atomic no. of He
4 deltaHe=5.84
     //doublet splitting of the first excited state of
      He (cm-1)
5
6 //calculation
7 deltaH=deltaHe*(zH/zHe)^4;
     //doublet splitting for hydrogen atom (cm-1)
8
9 printf("\n doublet splitting for H atom = %.3f cm-1"
     ,deltaH);
```

Scilab code Exa 4.2.16 Ex 16

```
1 //variable initialization
2 z=1
     //atomic no. of hydrogen atom
3 n=2
```

```
4 l=1
5
6 //calculation
7 delta=(5.84*z^4)/(n^3*l*(l+1));

    //spin-orbit interaction splitting (cm-1)
8
9 printf("\n spin-orbit interaction splitting = %.3f
    cm-1",delta);
```

Chapter 20

Atomic Spectra III

Scilab code Exa 4.3.4 Ex 4

```
1 //variable initialization
2 R=109729

    //(cm-1)
3 T1=43487

    //(cm-1)
4 T2=28583

    //(cm-1)
5 n=2
6
7 //calculation
8 delta=n-sqrt(R/T2);

    //quantum defect
9
10 printf("\nquantum defect = %.4f",delta);
```

Scilab code Exa 4.3.5 Ex 5

```
1 //variable initialization
2 R=109737
     //(cm-1)
3 n=1.805
     //effective quantum number for the ground state
     of rubidium
4
5 //calculation
6 T=R/(8065*n^2);
     //ionization potential of rubidium (eV)
7
8 printf("\nThe ionization potential of rubidium = %.3
     f eV",T);
```

Scilab code Exa 4.3.6 Ex 6

```
1 //variable initialization
2 ratio=2.5
     //ratio of ionization potential of hydrogen and
     sodium
3 n=3
4
5 //calculation
6 z=sqrt(n^2/ratio);
     //effective atomic number of sodium
7
8 printf("\nEffective atomic number of sodium = %.2f",
     z);
```

Scilab code Exa 4.3.7 Ex 7

```
1 //variable initialization
2 hc=12400
    //value of product of plank's constant and speed
    of light (eV    )
3 E1=3.18
    //separation of 4s and 3s level (eV)
4 lembda=5890
    //wavelength of the first member of principal
    series of sodium (   )
5
6 //calculation
7 E2=hc/lembda;
    //separation of 3s and 3p levels (eV)
8 deltaE=E1-E2;
    //separation of 4s and 3p level (eV)
9 lembda1=hc/deltaE;
    //wavelength of the first member of sharp series
    (   )
10
11 printf("\n    = %.0 f     ",lembda1);
```

Scilab code Exa 4.3.8 Ex 8

```
1 //variable initialization
2 lembda1=5890*10^-10;

    //wavelength of doublet (  )
3 lembda2=5896*10^-10;

    //wavelength of doublet (  )
4 h=6.63*10^-34;

    //Plank's constant (Js)
5 c=3*10^8;

    //speed of light (m/s)
6 e=1.6*10^-19;

    //Charge of electron (coulombs)
7
8 //calculation
9 deltaV=(lembda2-lembda1)/(lembda1*lembda2);
                                        //wave no. (m
    -1)
10 deltaE=(h*c*deltaV)/e;

    //separation of energy levels (eV)
11
12 printf("\n E = %.2e eV",deltaE);
```

Scilab code Exa 4.3.9 Ex 9

```
1 //variable initialization
2 deltaT=2.1*10^-3;

    //(eV)
3 lembda=5893*10^-8;
```

```
    //(   )
4
5 //calculation
6 deltaV=deltaT*8065;

    //(cm-1)
7 deltalembda=deltaV*lembda^2;

    //(cm)
8
9 printf("\n      = %.2e cm",deltalembda);
```

Scilab code Exa 4.3.10 Ex 10

```
1 //variable initialization
2 E1=16960

    //mean position of the level (cm-1)
3 E2=24490

    //convergence limit of sharp series (cm-1)
4
5 //calculation
6 I=(E1+E2)/8065;

    //ionization energy of sodium atom (eV)
7
8 printf("\nI = %.4f eV",I);
```

Scilab code Exa 4.3.11 Ex 11

```
1 //variable initialization
```

```
2 E1=41450
    //principal series for sodium atom (cm-1)
3 E2=24477
    //sharp series for sodium atom (cm-1)
4
5 //calculation
6 I=(E1+E2)/8065;
    //ionization energy of sodium atom (eV)
7
8 printf("\nI = %.4f eV",I);
```

Scilab code Exa 4.3.12 Ex 12

```
1 //variable initialization
2 E1=14904
    //mean position of the level (cm-1)
3 E2=28583
    //convergence limit of sharp series (cm-1)
4
5 //calculation
6 I=(E1+E2)/8065;
    //ionization energy of Li atom (eV)
7
8 printf("\nI = %.2f eV",I);
```

Scilab code Exa 4.3.13 Ex 13

```
1 //variable initialization
2 R=109734

    //(cm-1)
3 T=24477

    //(cm-1)
4 Zeff=1
5 n=3
6
7 //calculation
8 delta=n-(Zeff*sqrt(R/T));

    //quantum defect for 3p configuration of sodium
9
10 printf("\n    = %.3f",delta);
```

Scilab code Exa 4.3.14 Ex 14

```
1 //variable initialization
2 z1=1

    //atomic no.
3 z2=2

    //atomic no.
4 deltaT2=5.84

    //doublet splitting of the first excited state
    for z=2 (cm-1)
5
6 //calculation
7 deltaT1=deltaT2*(z1/z2)^4;

    //separation in hydrogen atom (cm-1)
```

```
8
9 printf("\n separation in hydrogen atom = %.3f cm-1",
    deltaT1);
```

Chapter 21

Magneto Optic and Electro Optic Phenomena

Scilab code Exa 4.4.1 Ex 1

```
1 //variable initialization
2 e=1.6*10^-19;

    //charge of electron (Coulomb)
3 B=0.5

    //magnetic field (Tesla)
4 lembda=6438*10^-10;

    //wavelength of the line (m)
5 m=9.1*10^-31;

    //mass of electron (kg)
6 c=3*10^8;

    //speed of light (m/s)
7
8 //calculation
9 dlembda=(e*B*lembda^2*10^10)/(4*%pi*m*c);
```

```
                                                    //normal
    Zeeman splitting (  )
10
11 printf("\nZeeman shift = %.3f    ",dlembda);
```

Scilab code Exa 4.4.2 Ex 2

```
1 //variable initialization
2 e=1.6*10^-19;
    //charge of electron (Coulomb)
3 B=1
    //magnetic field (Tesla)
4 lembda=612*10^-9;
    //wavelength of the line (m)
5 m=9.1*10^-31;
    //mass of electron (kg)
6 c=3*10^8;
    //speed of light (m/s)
7
8 //calculation
9 dlembda1=(e*B*lembda^2*10^10)/(4*%pi*m*c);
                                                    //normal
    Zeeman splitting (  )
10 dlembda2=2*dlembda1;
    //Separation of outer lines (  )
11
12 printf("\nSeparation of outer lines = %.2f    ",
    dlembda2);
```

Scilab code Exa 4.4.3 Ex 3

```
1 //variable initialization
2 mu=9.27*10^-24;
    //(J/T)
3 B=1*10^-1;
    //external magnetic field (T)
4 h=1.054*10^-34;
    //Plank's constant (Js)
5 J=3/2;
6 L=1;
7 S=1/2;
8
9 //calculation
10 g=1+(((J*(J+1))+(S*(S+1))-(L*(L+1)))/(2*J*(J+1)));
                                    //Lande g-factor
11 omega=(g*mu*B)/h;
    //angular velocity of precession (rad/s)
12
13 printf("\n   = %.1e rad/sec",omega);
```

Scilab code Exa 4.4.4 Ex 4

```
1 //(i)
2 printf("\n(i) Energy level does not split");
3
4 //(ii)
5 J=5/2;
```

```
6 sub=2*J+1;
7 printf("\n(ii) number of sub-shells = %.0f",sub);
8
9 //(iii)
10 printf("\n(iii) Energy level does not split");
```

Scilab code Exa 4.4.5 Ex 5

```
1 //variable initialization
2 S1=0
3 L1=2
4 J1=2
5 g1=1
6 S2=1
7 L2=3
8 J2=4
9 g2=5/4
10 B=0.25

    //magnetic field (T)
11 mu=5.79*10^-5;

    //mass (eV/T)
12
13 //(i)
14 E1=4*g1*mu*B;

    //total splitting (eV)
15
16 //(ii)
17 E2=8*g2*mu*B;

    //total splitting (eV)
18
19 printf("\n(i) total splitting = %.2e eV\n(ii) total
```

```
splitting = %.4e eV",E1,E2);
```

Scilab code Exa 4.4.7 Ex 7

```
1 //variable initialization
2 mu=9.27*10^-24;
    //(J/T)
3 B=0.45;
    //magnetic field ( b /m^2)
4 h=1.054*10^-34;
    //Plank's constant (Js)
5 k=[5/3 1 1/3 -1/3 -1 -5/3];
    //value of g'Mj'-gMj
6
7 //calculation
8 c=(mu*B)/h;
    //constant (rad/s)
9 deltaomega1=c*k(1);
    //displacement of Zeeman component (rad/s)
10 deltaomega2=c*k(2);
    //displacement of Zeeman component (rad/s)
11 deltaomega3=c*k(3);
    //displacement of Zeeman component (rad/s)
12 deltaomega4=c*k(4);
    //displacement of Zeeman component (rad/s)
13 deltaomega5=c*k(5);
```

```
    //displacement of Zeeman component (rad/s)
14 deltaomega6=c*k(6);

    //displacement of Zeeman component (rad/s)
15
16 printf("\ndisplcement of Zeeman component = %.2e, %
    .2e, %.2e, %.2e, %.2e, %.2e rad/s",deltaomega1,
    deltaomega2,deltaomega3,deltaomega4,deltaomega5,
    deltaomega6);
```

Scilab code Exa 4.4.8 Ex 8

```
1 //variable initialization
2 m=9.1*10^-31;

    //mass of electron (Kg)
3 h=1.054*10^-34;

    //Plank's constant (Js)
4 B=1.2

    //magnetic field (mu*b/m^2)
5 gs=2

    //for a pure spin system
6 J=0.5;

    //for a pure spin system
7
8 //calculation
9 mub=h/(2*m);

    //(eV/T)
10 deltaE=2*gs*mub*B*J;
```

```
    //energy difference b/w electrons(eV)
11
12 printf("\n  E  = %.2e eV",deltaE);
```

Scilab code Exa 4.4.9 Ex 9

```
1 //variable initialization
2 m=9.1*10^-31;
    //mass of electron (Kg)
3 h=1.054*10^-34;
    //Plank's constant (Js)
4 B=5
    //magnetic field (T)
5 lembda=1210
    //wavelength of spectral line (  )
6 M=[1 0 -1 1 0 -1];
    //value of Ml+2*Ms
7 ch=12400
    //product of speed of light and Plank's constant
    (eV*  )
8
9 //calculation
10 dE=(h/(2*m))*B*M;
    //value of dE(upper)-dE(lower) (eV)
11 dlembda=(lembda^2/ch)*dE;
    //wavelengths of the spectral lines in the
```

```
    pattern ( )
12
13 printf("\nwavelengths of the line = %.0f+%.3f, %.0f+
    %.0f, %.0f%.3f    ",lembda,dlembda(1),lembda,
    dlembda(2),lembda,dlembda(3));
```

Chapter 22

X Rays and X Ray Spectra

Scilab code Exa 4.5.1 Ex 1

```
1 //variable initialization
2 ch=12400
    //product of speed of light and Plank's constant
    (eV*  )
3 lembda1=0.024;
    //Compton wavelength of X-ray (  )
4 lembda2=1
    //wavelength of X-ray (  )
5
6 //(i)
7 x1=ch/lembda1;
    //minimum voltage across X-ray tube (V)
8
9 //(ii)
10 x2=ch/(lembda2*10^3);
    //minimum voltage across X-ray tube (kV)
```

```
11
12 //(iii)
13 x3=1.02
      //minimum energy of X-ray photon (M*eV)
14
15 printf("\n(i) voltage = %.2e V\n(ii) voltage = %.1f
      KV\n(iii) energy = %.2f MeV",x1,x2,x3);
```

Scilab code Exa 4.5.2 Ex 2

```
1 //variable initialization
2 n=3/2;
3 dlembda=26*10^-2;
      //shifting in short wave limit of X-ray spectrum
      (  )
4 ch=12400
      //product of speed of light and Plank's constant
      (eV*  )
5 e=1.6*10^-19;
      //charge of electron (Coulomb)
6
7 //solution
8 V=((n-1)/n)*(ch/(dlembda*10^3));
                                                    //
      initial voltage applied to the tube (KV)
9
10 printf("\nInitial voltage = %.1f KV",V);
```

Scilab code Exa 4.5.3 Ex 3

```
1 //variable initialization
2 R=10972900
    //(m-1)
3 lembda=1.54*10^-10;
    //wavelength of K line (m)
4
5 //calculation
6 z=1+sqrt(4/(3*lembda*R));
    //atomic number of the target element
7
8 printf("\nZ = %.0f",z);
```

Scilab code Exa 4.5.4 Ex 4

```
1 //variable initialization
2 z1=29
    //atomic no. of Copper
3 z2=26
    //atomic no. of Iron
4 lembda1=193
    //wavelength of K line in Iron (pm)
5
6 //calculation
7 lembda=((z2-1)/(z1-1))^2*lembda1;
                                                    //
    wavelength of K line in Copper (pm)
8
9 printf("\n    = %.0f pm",lembda);
```

Scilab code Exa 4.5.5 Ex 5

```
1 //variable initialization
2 z1=13
    //atomic no. of Al
3 z2=27
    //atomic no. of Co
4 R=1.097*10^7;
    //(m-1)
5
6 //calculation
7 lembda1=(4*10^12)/(3*R*(z1-1)^2);
                                                    //
    wavelength of K line in Al (pm)
8 lembda2=(4*10^12)/(3*R*(z2-1)^2);
                                                    //
    wavelength of k line in Co (pm)
9
10 printf("\n wavelength of Al = %.0f pm\n wavelength
    of Co = %.0f pm",lembda1,lembda2);
```

Scilab code Exa 4.5.6 Ex 6

```
1 //variable initialization
2 lembda1=250*10^-12;
    //wavelength of K-alpha line (m)
3 lembda2=179*10^-12;
```

```
    //wavelength of K-alpha line (m)
R=10972900

    //(m-1)

//calculation
z1=1+sqrt(4/(3*lembda1*R));

    //atomic number
z2=1+sqrt(4/(3*lembda2*R));

    //atomic number
printf("\nThe required elements are: Z =");
for i=z1+1:1:z2-1
    printf(" %.0f",i);
end
```

Scilab code Exa 4.5.7 Ex 7

```
//variable initialization
ch=12400

    //product of speed of light and Plank's constant
    (eV*  )
Rch=13.6

    //product of speed of light, Plank's constant and
     R (eV)
z=23

    //atomic no. of vanadium
lembda=24

    //wavelength of L absorption edge (  )

```

```
7 //calculation
8 El=ch/(lembda*10^3);

    //binding energy of L electron (KeV)
9 Ek=((3/(4*10^3))*Rch*(z-1)^2)+El;
                                                    //
    binding energy of K electron (KeV)
10
11 printf("\nBinding energy of K-electron = %.2f KeV",
    Ek);
```

Scilab code Exa 4.5.8 Ex 8

```
1 //variable initialization
2 ch=12.4

    //product of speed of light and Plank's constant
    (KeV*  )
3 lembda1=0.178

    //wavelength of K-alpha line (  )
4 lembda2=0.210

    //wavelength of K line (  )
5
6 //calculation
7 Ek=ch/lembda1;

    //binding energy of K electron (KeV)
8 El=Ek-(ch/lembda2);

    //binding energy of K-alpha electron (KeV)
9 lembda=ch/El;

    //wavelength of L absorption edge (  )
```

```
10
11 printf("\nWavelength of L absorption edge = %.2f    "
   ,lembda);
```

Scilab code Exa 4.5.9 Ex 9

```
1 //variable initialization
2 ch=12.4
     //product of speed of light and Plank's constant
     (KeV*  )
3 lembdak=0.18
     //wavelength of K absorption edge (  )
4 lembda=0.1
     //wavelength of incident photon (  )
5
6 //calculation
7 Ek=ch/lembdak;
     //binding energy of K electron (KeV)
8 E=ch/lembda;
     //energy of incident photon (KeV)
9 K=E-Ek;
     //maximum kinetic energy of ejected electron (KeV
     )
10
11 printf("\n KE = %.2f KeV",K);
```

Scilab code Exa 4.5.10 Ex 10

```
1 //variable initialization
2 ch=12.4

    //product of speed of light and Plank's constant
    (KeV*  )
3 Rch=13.6/10^3;

    //product of speed of light , Plank's constant and
     R (KeV)
4 lembdak=1.74

    //K band absorption edge wavelength of iron (  )
5 z=30;

    //atomic no. of zinc
6
7 //calculation
8 Ek=ch/lembdak;

    //binding energy of K electron in iron (KeV)
9 E=(3/4)*Rch*(z-1)^2;

    //energy of photon of K-alpha radiation (KeV)
10 K=E-Ek;

    //kinetic energy of the photoelectrons liberated
    from iron (KeV)
11
12 printf("\n KE = %.3f KeV",K);
```

Chapter 27

Raman Spectra

Scilab code Exa 5.5.1 Ex 1

```
1 //function for calculating the wave number
2 function[wave]=F(j)
3     wave=B*j*(j+1);
4 endfunction
5
6 //variable initialization
7 r=1.21*10^-10;

     //internuclear distance (meter)
8 m=2.7*10^-26;

     //mass of oxygen atom (kg)
9 h=6.626*10^-34;

     //Plank's constant (joule second)
10 c=3*10^8;

     //speed of light (meter/second)
11
12 //(a) moment of inertia
13 mu=m/2;
```

```
    //reduced mass (kg)
I=mu*r^2;

    //moment of inertia (kg m^2)

//(b) rotational constant
B=h/(8*%pi^2*I*c);

    //rotational constant (m-1)

//(c) wave number
waveno=F(1)-F(0);

    //wave no. of the line corresponding to the
    transition J=0 to J=1 (m-1)

printf("\n(a) I = %.3e kg m^2\n(b) B = %.1f m-1\n(c)
    wave number = %.0f m-1",I,B,waveno);
```

Scilab code Exa 5.5.2 Ex 2

```
//function for calculating the energy level
function[energy]=F(j)
    energy=a*j*(j+1);
endfunction

//variable initialization
m=1.6738*10^-27;

    //mass of hydrogen atom (kg)
r=0.74*10^-10;

    //intermolecular distance of hydrogen molecule (
    meter)
```

```
9  h=1.054*10^-34;
       //Planck's constant (joule second)
10 e=1.6*10^-19;
       //Charge of electron (coulombs)
11
12 //calculation of rotational energy levels
13 mu=m/2;
       //reduced mass of hydrogen atom (kg)
14 I=mu*r^2;
       //moment of inertia of molecule (kg meter^2)
15 a=h^2/(2*I*e);
       //constant (eV)
16 E0=F(0);
       //energy of level 0 (eV)
17 E1=F(1);
       //energy of level 1 (eV)
18 E2=F(2);
       //energy of level 2 (eV)
19 E3=F(3);
       //energy of level 3 (eV)
20
21 printf("\nE0 = %.0f\nE1 = %.2e eV\nE2 = %.2e eV\nE3
       = %.2e eV",E0,E1,E2,E3);
```

Scilab code Exa 5.5.3 Ex 3

```
1  //variable initialization
2  u=1.68*10^-27;

      //mass of hydrogen atom (kg)
3  m1=16;

      //mass of oxygen atom in terms u
4  m2=1;

      //mass of hydrogen atom in terms of u
5  I=1.48*10^-47;

      //moment of inertia of OH-radical (kg m^2)
6  h_bar=1.054*10^-34;

      //Planck's constant (joule second)
7  j=5;

      //energy level of OH-radical
8  c=3*10^8;

      //speed of light (meter/second)
9  h=6.626*10^-34;

      //Plank's constant (joule second)
10
11 //(a) internuclear distance
12 mu=((m1*m2)/(m1+m2))*u;

      //reduced mass of the molecule (kg)
13 r=(sqrt(I/mu))*10^10;

      //internuclear distance of molecule (  )
14
15 //(b) angular momentum
16 P=h_bar*sqrt(j*(j+1));

      //angular momentum of molecule (joule second)
```

```
17
18 //(c) angular velocity
19 omega=P/I;

      //angular velocity of molecule (radian/second)
20
21 //(d) wave number
22 B=h/(8*%pi^2*I*c);

      //rotational constant (m-1)
23 no=2*B*(j+1);

      //wave no. of line corresponding to transition j
      =5 to j=6 (m-1)
24
25 //(e) energy absorbed
26 E=c*h*no;

      //energy absorbed in the transition j=6 to j=5 (
      joule)
27
28 printf("\n(a) r = %.3f    \n(b) J = %.2e joule second
      \n(c)     = %.2e radian/second\n(d) wave number =
      %.2e m-1\n(e) E = %.1e joule",r,P,omega,no,E);
```

Scilab code Exa 5.5.4 Ex 4

```
1 //variable initialization
2 h=6.63*10^-34;

      //Plank's constant (joule second)
3 v=1.153*10^11;

      //Frequency of absorption line (Hz)
4 mu=11.38*10^-27;
```

```
    //Recuced mass of the molecule (kg)
5
6 //Calculation of the internuclear distance
7 I=h/(4*%pi^2*v);

    //moment of inertia of CO molecule (kg m^2)
8 r=sqrt(I/mu)*10^10;

    //Internuclear distance (  )
9
10 printf("\n Internuclear distance = %.2f    ",r);
```

Scilab code Exa 5.5.5 Ex 5

```
1 //variable initialization
2 mu=1.62*10^-27;

    //Reduced mass of HCL (kg)
3 c=3*10^8;

    //Velocity of light (m/s)
4 h=6.62*10^-34;

    //Plank's constant (joule second)
5 v1_P=2906.3

    //Wave no. of P branch (cm-1)
6 v2_P=2927.5

    //Wave no. of P branch (cm-1)
7 v3_P=2948.7

    //Wave no. of P branch (cm-1)
8 v4_P=2969.9
```

```
    //Wave no. of P branch (cm-1)
v1_R=3012.2

    //Wave no. of R branch (cm-1)
v2_R=3033.4

    //Wave no. of R branch (cm-1)
v3_R=3054.6

    //Wave no. of R branch (cm-1)
v4_R=3078.8

    //Wave no. of R branch (cm-1)

//(i) Equilibrium internuclear distance
delta_v=v2_P-v1_P;

    //Separation of successive line of P and R branch
     (cm-1)
B=delta_v/2;

    //rotational constant (cm-1)
I=h/(8*%pi^2*B*10^2*c);

    //Moment of inertia (kg m^2)
r=sqrt(I/mu)*10^10;

    //Equilibrium internuclear distance (  )

//(ii) Force constant
v0=(v4_P+v1_R)/2;

    //Equlibrium frequency (cm-1)
k=4*%pi^2*mu*c^2*v0^2*10^4;

    //Force constant of HCl (N/m)

```

```
24 printf("\n(i) Equilibrium internuclear distance = %
     .2f    \n(ii) Force constant = %.0f N/m",r,k);
25
26 //Note: the answer of (ii) part is wrong in the book
```

Scilab code Exa 5.5.6 Ex 6

```
1 //variable initialization
2 mu=8.37*10^-28;
     //Reducec mass of hydrogen molecule (kg)
3 h=6.58*10^-16;
     //Plank's constant (eV s)
4 E0=0.273
     //Ground state vibrational energy of hydrogen
     molecule (eV)
5
6 //calculation of force constant of the molecule
7 k=mu*((2*E0)/h)^2;
     //force constant of hydrogen molecule (N/m)
8
9 printf("\n Force constant = %.0f N/m",k);
```

Scilab code Exa 5.5.7 Ex 7

```
1 //variable initialization
2 m1=1;
     //molar mass of H atom (amu)
```

```
3 m2=35;

    //molar mass of Cl atom (amu)
4 u=1.68*10^-27;

    //atomic mass unit (kg)
5 v=2885.9*100;

    //wave no. of line (m-1)
6 c=3*10^8;

    //Velocity of light (m/s)
7
8 //calculation of force constant
9 mu=((m1*m2)/(m1+m2))*u;

    //reduced mass of HCl molecule (kg)
10 k=4*(%pi*c*v)^2*mu;

    //force constant of HCl molecule (N/m)
11
12 printf("\n force constant = %.2f N/m",k);
```

Scilab code Exa 5.5.8 Ex 8

```
1 //function for calculating the various vibrational
    energy levels of CO molecule
2 function[energy]=F(V)
3     energy=((V+.5)*h*v)/e;
4 endfunction
5
6 //function for converting eV to cm-1
7 function[energy]=G(V)
8     energy=((V+.5)*h*v*8065)/e;
9 endfunction
```

```
10
11 //variable initialization
12 m1=12;
      //molar mass of C atom (amu)
13 m2=16;
      //molar mass of O atom (amu)
14 u=1.68*10^-27;
      //atomic mass unit (kg)
15 k=1870
      //force constant of CO molecule (N/m)
16 h=6.6*10^-34;
      //Plank's constant (joule second)
17 e=1.602*10^-19;
      //charge of electron (Coulomb)
18
19 //calculation of energy levels
20 mu=((m1*m2)/(m1+m2))*u;
      //reduced mass of CO molecule (kg)
21 v=(1/(2*%pi))*sqrt(k/mu);
      //frequency of vibration of CO molecule (Hz)
22 e1=F(0);
      //energy of 1st level of CO molecule (eV)
23 E1=G(0);
      //energy of 1st level of CO molecule (cm-1)
24 e2=F(1);
      //energy of 2nd level of CO molecule (eV)
25 E2=G(1);
```

```
    //energy of 2nd level of CO molecule (cm-1)
26 e3=F(2);

    //energy of 3rd level of CO molecule (eV)
27 E3=G(2);

    //energy of 3rd level of CO molecule (cm-1)
28
29 printf("\nE = %.3f eV, %.3f eV, %.3f eV
    .....\n  = %.1f cm-1, %.1f
    cm-1, %.1f cm-1.....",e1,e2,e3,E1,E2,
    E3);
```

Scilab code Exa 5.5.9 Ex 9

```
1 //variable initialization
2 mu=1.61*10^-27;

    //reduced mass of HCl molecule (kg)
3 c=3*10^8;

    //speed of light (m/s)
4 lembda=3.465*10^-6;

    //wavelength of infrared radiation (m)
5
6 //calculation of force constant
7 v=c/lembda;

    //frequency of radiation (s-1)
8 k=4*(%pi*v)^2*mu;

    //force constant of HCl molecule (N/m)
9
```

```
10 printf("\nforce constant = %.1f N/m",k);
```

Scilab code Exa 5.5.10 Ex 10

```
1 //variable initialization
2 h=6.6*10^-34;
    //Plank's constant (joule second)
3 mu=1.62*10^-27;
    //reduced mass of HCl molecule (kg)
4 c=3*10^8;
    //speed of light (m/s)
5 v=2.886*10^5;
    //wave no. of absorption line in infrared
    spectrum (m-1)
6
7 //calculation of amplitude of vibration
8 k=4*(%pi*c*v)^2*mu;
    //force constant of HCl molecule (N/m)
9 amp=sqrt((h*c*v)/k)*10^10;
    //amplitude of vibration in the ground state (  )
10
11 printf("\namplitude of vibration = %.2f   ",amp);
```

Scilab code Exa 5.5.11 Ex 11

```
1 //variable initialization
```

```
2 v1=214330

    //fundamental band for CO molecule (m-1)
3 v2=425970

    //first overtone for CO molecule (m-1)
4 A=[1 -2;2 -6];

    //coefficient matrix
5 b=[v1;v2];

    //right hand side matrix
6
7 //calculation
8 x=inv(A)*b;

    //values of omega and x*omega (m-1)
9
10 printf("\n e = %.0f m-1\nxe* e = %.0f m-1",x(1),x
    (2));
```

Scilab code Exa 5.5.12 Ex 12

```
1 //variable initialization
2 v1=288600

    //intense absorption (m-1)
3 v2=566800

    //intense absorption (m-1)
4 v3=834700

    //intense absorption (m-1)
5 A=[1 -2;2 -6];
```

```
    //coefficient matrix
6 b=[v1;v2];

    //right hand side matrix
7 mu=1.61*10^-27;

    //reduced mass (kg)
8 c=3*10^8;

    //speed of light (m/s)
9
10 //calculation
11 x=inv(A)*b;

    //values of omega and x*omega (m-1)
12 k=4*(%pi*c*x(1))^2*mu;

    //force constant (N/m)
13
14 printf("\n e = %.0f m-1\nxe* e = %.0f m-1\nforce
    constant = %.1f N/m",x(1),x(2),k);
```

Scilab code Exa 5.5.13 Ex 13

```
1 //variable initialization
2 v1=8.657*10^13;

    //frequency of rotation absorption spectrum (Hz)
3 v2=8.483*10^13;

    //frequency of rotation absorption spectrum (Hz)
4 h=6.6*10^-34;

    //Plank's constant (joule second)
5 mu=1.544*10^-27;
```

```
    //Recuced mass of CH molecule (kg)
6
7 //(i) equilibrium separation
8 I=h/(2*%pi^2*(v1-v2));

    //Moment of inertia (kg m^2)
9 r=sqrt(I/mu);

    //equilibrium internuclear distance (m)
10
11 //(ii) force constant of molecule
12 v0=(v1+v2)/2;

    //Central frequency (Hz)
13 k=4*mu*(%pi*v0)^2;

    //Force constant of CH molecule (N/m)
14
15 printf("\n (i) equilibrium separation = %.2e meter\n
    (ii) force constant = %.0f N/m",r,k);
```

Scilab code Exa 5.5.14 Ex 14

```
1 //variable initialization
2 k=448

    //force constant of CH molecule (N/m)
3 mu=4.002*10^-27;

    //reduced mass of CH molecule (kg)
4 r=0.112*10^-9;

    //internuclear distance (m)
5 h=6.6*10^-34;
```

```
    //Plank's constant (joule second)
6
7 //Calculation of peak frequencies
8 v0=(1/(2*%pi))*sqrt(k/mu);

    //central frequency (s-1)
9 I=mu*r^2;

    //moment of inertia of molecule (kg m^2)
10 x=h/(4*%pi^2*I);

    //additional frequency (s-1)
11 v1=v0+x;

    //peak frequency (Hz)
12 v2=v0-x;

    //peak frequency (Hz)
13
14 printf("\n Peak frequencies = %.3e Hz, %.3e Hz",v1,
    v2);
```

Scilab code Exa 5.5.15 Ex 15

```
1 //variable initialization
2 v1=2174.07

    //peak wave number (cm-1)
3 v2=2166.35

    //peak wave number (cm-1)
4 h=6.6*10^-34;

    //Plank's constant (joule second)
```

```
5  c=3*10^8;
       //Speed of light (m/s)
6  mu=1.145*10^-26;
       //Reduced mass of CO molecule (kg)
7
8  //(a) central frequency
9  B=(v1-v2)/4;
       //Rotational constant (cm-1)
10 v0=(v1+v2)/2;
       //Central frequency (cm-1)
11
12 //(b) internuclear distance
13 I=h/(8*%pi^2*B*100*c);
       //moment of inertia of molecule (kg m^2)
14 r=sqrt(I/mu)*10^10;
       //equilibrium internuclear distance (  )
15
16 //(c) force constant
17 k=4*mu*(%pi*c*v0*100)^2;
       //force constant (N/m)
18
19 printf("\n(a) central frequency = %.2f cm-1\n(b)
       internuclear distance = %.2f   \n(c) force
       constant = %.0f N/m",v0,r,k);
```

Scilab code Exa 5.5.16 Ex 16

```
1  //variable initialization
```

```
2  mu=3.142*10^-27;
      //reduced mass of the molecule (kg)
3  r=1.288*10^-10;
      //internuclear distance (m)
4  h=6.6*10^-34;
      //Plank's constant (joule second)
5  c=3*10^8;
      //Speed of light (m/s)
6  v0=201100
      //central frequency (m-1)
7
8  //Calculation
9  I=mu*r^2;
      //moment of inertia of molecule (kg m^2)
10 B=h/(8*%pi^2*I*c);
      //Rotational constant (m-1)
11 vR0=v0+(2*B);
      //wave no. of 1st line of R-branch (m-1)
12 vR1=v0+(4*B);
      //wave no. of 2nd line of R-branch (m-1)
13 vP1=v0-(2*B);
      //wave no. of 1st line of P-branch (m-1)
14 vP2=v0-(4*B);
      //wave no. of 2nd line of P-branch (m-1)
15
16 printf("\n V_R(0) = %.0f m-1\n V_R(1) = %.0f m-1\n
      V_P(1) = %.0f m-1\n V_P(2) = %.0f m-1",vR0,vR1,
```

```
vP1,vP2);
```

Scilab code Exa 5.5.17 Ex 17

```
1 //variable initialization
2 mu=1.62*10^-27;
    //reduced mass of HCl molecule (kg)
3 r=1.293*10^-10;
    //internuclear distance (m)
4 h=6.6*10^-34;
    //Plank's constant (joule second)
5 c=3*10^8;
    //Speed of light (m/s)
6
7 //Calculation of separation between lines R(0) and P
    (1) of the fundamental band of HCl 35
8 I=mu*r^2;
    //moment of inertia of molecule (kg m^2)
9 B=h/(8*%pi^2*I*c);
    //Rotational constant (m-1)
10 sep=4*B;
    //separation b/w lines R(0) and P(1) (m-1)
11
12 printf("\n      = %.0f m-1",sep);
```

Scilab code Exa 5.5.18 Ex 18

```
1 //variable initialization
2 a=214.6*100;

    //(m-1)
3 b=0.6*100;

    //(m-1)
4 h=6.6*10^-34;

    //Plank's constant (joule second)
5 c=3*10^8;

    //Speed of light (m/s)
6 no=1/(%e);

    //number of molecules in state with respect to
    ground state
7 k=1.38*10^-23;

    //Boltzmann constant (J K-1)
8
9 //Calculation
10 deltaE=h*c*(a-2*b);

    //difference in the energies of state 0 and state
     1 (J)
11 T1=deltaE/k;

    //temperature at which number of molecules in
    state 1 is 1/e times of state 0 (K)
12 T2=deltaE/(k*log(10));

    //temperature at which number of molecules in
    state 1 is 10% of state 0 (K)
13
14 printf("n(i) T = %.0f K\n(ii) T = %.0f K",T1,T2);
```

Scilab code Exa 5.5.19 Ex 19

```
1 //variable initialization
2 vexc=4358*10^-10;
     //wavelength of exciting line (m)
3 vsto=4458*10^-10;
     //wavelength of Stokes line (m)
4
5 //calculation of wavelength of Anti-stokes line
6 vbar_exc=1/vexc;
     //wave number of exciting line (m-1)
7 vbar_sto=1/vsto;
     //wave number of Stokes line (m-1)
8 delta_vbar=vbar_exc-vbar_sto;
     //Raman shift (m-1)
9 vbar_antistoke=vbar_exc+delta_vbar;
                                                //
     Wave number of Anti-Stokes line (m-1)
10 lembda_antistoke=(1/vbar_antistoke)*10^10;
                                              //Wavelength
      of Anti-Stokes line (  )
11
12 printf("\nwavelength of Anti-stokes line = %.1f   ",
     lembda_antistoke);
```

Scilab code Exa 5.5.20 Ex 20

```
1 //variable initialization
2 h=6.62*10^-34;

    //Plank's constant (joule second)
3 c=3*10^8;

    //Speed of light (m/s)
4 x=62.4*100;

    //(m-1)
5 y=41.6*100;

    //(m-1)
6
7 //calculation of the moment of inertia of HCl
    molecule
8 B=y/4;

    //Rotational constant of HCl (m-1)
9 I=h/(8*%pi^2*B*c);

    //Moment of inertia (kg m^2)
10
11 printf("\n I = %.1e kg m^2",I);
```

Scilab code Exa 5.5.21 Ex 21

```
1 //function for calculating the vibrational energy of
    O2 molecule
2 function[energy]=F(v)
3     energy=(((v+.5)*a)-(((v+.5)^2)*b))*h*c;
4 endfunction
5
6 //variable initialization
7 h=6.62*10^-34;
```

```
    //Plank's constant (joule second)
c=3*10^8;

    //Speed of light (m/s)
a=1580.36*100;

    //value of  e  (m-1)
b=12.07*100;

    //value of  exe  (m-1)

//Calculation of zero point energy
E0=F(0);

    //Zero point energy of the molecule (J)

//Calculation of vibrational Raman shift
shift=(F(1)-F(0))/(h*c);

    //Expected vibrational Raman shift (m-1)

printf("\nZero point energy = %.3e J\nExpected
    vibrational Raman shift = %.0f m-1",E0,shift);
```