Scilab Textbook Companion for Modern Physics by R. A. Serway[1]

Created by
Nusrat Ali
Modern Physics
Electronics Engineering
Model Institute of Engineering and Technology
College Teacher
Ms. Bhavna Sharma
Cross-Checked by
Lavitha Pereira

January 23, 2014

[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: Modern Physics

Author: R. A. Serway

Publisher: Thomson Learning, USA

Edition: 3

Year: 2005

ISBN: 0-534-49339-4

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

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

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

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

Contents

List of Scilab Codes

Chapter 1

Relativity I

Scilab code Exa 1.2 Period of the pendulum wrt different frames of references

```
// Scilab code Ex1.2: Pg.18 (2005)
clc; clear;
c = 3e+08;        // Velocity of light , m/s
v = 0.95*c;       // Velocity of observer , m/s
T_proper = 3;     //  Proper time period of pendulum
   in rest frame , s
gama = 1/(sqrt(1 - (v/c)^2));        // Multiplying
   factor
// From time-dilation formula , we have
T = gama*T_proper;                   // Time period
   of pendulum w.r.t to moving observer , s
printf("\nTime period of pendulum w.r.t to moving
   observer = %3.1f s", T);

// Result
// Time period of pendulum w.r.t to moving observer
   = 9.6 s
```

Scilab code Exa 1.3 Contraction of spaceship

```
// Scilab code Ex1.3: Pg 20 (2005)
clc; clear;
c = 3e+08;                          // Velocity of light, m/s
L_p = 100;                          // Proper length of spaceship, m
v = 0.99*c;                         // Velocity of spaceship, m/s
// Using length contracction formula,
L = L_p*sqrt(1 - (v/c)^2);                          // Observed length of spaceship, m
printf("Observed length of spaceship = %2d m", L);

// Result
// Observed length of spaceship = 14 m
```

Scilab code Exa 1.4 Altitude of spaceship wrt different frames of references

```
// Scilab code Ex1.4: Pg 20 (2005)
clc; clear;
c = 3e+08;                          // Velocity of light, m/s
L_p = 435;                          // Proper altitude of spaceship, m
v = 0.970*c;                        // Velocity of spaceship, m/s
// Using length contracction formula,
L = L_p*sqrt(1 - (v/c)^2);                          // Observed altitude of spaceship, m
printf("Observed altitude of spaceship = %2d m", ceil(L));

```

```
// Result
// Observed altitude of spaceship = 106 m
```

Scilab code Exa 1.5 Shape of spaceship seen from different frames of references

```
// Scilab code Ex1.5: Pg 20 (2005)
clc; clear;
c = 3e+08;                    // Velocity of light, m/s
L_p = 50;                     // Proper distance between points x & y of spaceship, m
v = 0.950*c;                  // Velocity of spaceship, m/s
// Using length contracction formula,
L = L_p*sqrt(1 - (v/c)^2);                    // Observed distance between points x & y of spaceship, m
printf("\nObserved distance between points x and y of spaceship = %4.1f m", L);
printf("\nThe spaceship will get contracted in the direction of motion");

// Result
// Observed distance between points x and y of spaceship = 15.6 m
// The spaceship will get contracted in the direction of motion
```

Scilab code Exa 1.6 Speed of recession of the galaxy Hydra

```
// Scilab code Ex1.6: Pg 25 (2005)
clc; clear;
```

```
// For simplification assume velocity of light equal
    to unity
c = 1                        // Velocity of light , m/s
lamda_obs = 474e-09;          // Wavelength measured
   by observer , m
lamda_source = 394e-09;        // Wavelength measured
    in the source 's rest frame , m
v = ((lamda_obs^2 - lamda_source^2)/(lamda_obs^2 +
   lamda_source^2))*c;                              //
   Receding velocity of Hydra, m/s
printf("\nReceding velocity of Hydra = %5.3fc m/s",
   v);

// Result
// Receding velocity of Hydra = 0.183c m/s
```

Scilab code Exa 1.8 Relative velocity of spaceships

```
// Scilab code Ex1.8: Pg 30 (2005)
clc; clear;
// For simplification assume velocity of light equal
    to unity
c = 1;                        // Velocity of light , m/s
v = 0.750*c;                  // Velocity of spaceship A
   relative to S frame , m/s
u_x = (-0.850)*c;             // Velocity of spaceship B
   relative to S frame , m/s
// Using Lorentz velocity transformation
U_x = (u_x - v)/(1 - u_x*v/c^2);                     //
   Velocity of spaceship B with respect to spaceship
    A, m/s
printf("\nVelocity of spaceship B with respect to
   spaceship A = %6.4fc m/s", U_x);

// Result
```

```
// Velocity of spaceship B with respect to spaceship
    A = −0.9771c m/s
```

Scilab code Exa 1.9 Velocity of ball wrt stationary observer

```
// Scilab code Ex1.9: Pg 30 (2005)
clc; clear;
// For simplification assume velocity of light equal
    to unity
c = 1;                          // Velocity of light, m/s
v = 0.800*c;                    // Velocity of motorcycle w
   .r.t stationary observer, m/s
U_x = 0.700*c;                  // Velocity of ball in the
   reference frame of motorcyclist, m/s
// Using  inverse Lorentz velocity transformation
u_x = (U_x + v)/(1 + U_x*v/ c^2);                    //
    Velocity of ball relative to stationary observer
   , m/s
printf("\nVelocity of ball relative to stationary
   observer = %6.4fc m/s", u_x);

// Result
// Velocity of ball relative to stationary observer
   = 0.9615c m/s
```

Scilab code Exa 1.10 Relative velocity of recession of two gang leaders

```
// Scilab code Ex1.10: Pg 30−31 (2005)
clc; clear;
// For simplification assume velocity of light equal
    to unity
c = 1;                            // Velocity of light, m/s
```

```
ux = 0.75*c;                    // Velocity of pack
    leader alpha, m/s
gama = 1/(sqrt(1 - (ux/c)^2));
u_x = 0;                        // Velocity component of
    beta measured in S frame, m/s
U_x = (u_x - ux)/(1 - u_x*ux/c^2);          //
    Velocity component of beta along X-axis measured
    in S' frame, (Velocity Addition Rule), m/s
u_y = -0.90*c;                          // Velocity
    component of beta long Y-axis measured in S frame
U_y = u_y/(gama*(1 - u_x*ux/c^2));    // Velocity
    component of beta along Y-axis measured in S'
    frame, m/s
U = sqrt(U_x^2+U_y^2);        // Relative velocity of
    recession of two gang leaders, m/s
printf("\nThe relative velocity of recession of two
    gang leaders = %4.2fc", U);

// Result
// The relative velocity of recession of two gang
    leaders = 0.96c
```

Chapter 2

Relativity II

Scilab code Exa 2.1 Momentum of an electron

```
// Scilab code Ex2.1: Pg.44 (2005)
clc; clear;
c = 3e08;   // Velocity of light, m/s
u = 0.750*c;   // Velocity of electron, m/s
m = 9.11e-31;   // Rest mass of electron, kg
p_r = m*u/(sqrt(1 - (u/c)^2));   // Relativistic
    momentum of electron, kgm/s
p = m*u;   // Classical momentum of electron, kg-m/s
printf("\nThe relativistic momentum of electron = %4
    .2fe-22 kg-m/s", p_r*1e+22);
printf("\nThe classical momentum of electron = %4.2
    fe-22 kg-m/s", p*1e+22);
printf("\nThe relativistic result is 50 percent
    greater than the classical result.");

//Result
// The relativistic momentum of electron = 3.10e-22
    kg-m/s
// The classical momentum of electron = 2.05e-22 kg-
    m/s
// The relativistic result is 50 percent greater
```

```
than the classical result.
```

Scilab code Exa 2.3 Energy of a speedy electron

```
// Scilab code Ex2.3: Pg.47 (2005)
clc; clear;
c = 3e+08;      // Velocity of light, m/s
u = 0.85*c;    // Velocity of electron, m/s
E_0 = 0.511;    // Rest energy of electron, MeV
E = E_0/(sqrt(1-(u/c)^2));    // Total energy of
    electron, MeV
K = E - E_0;    // Kinetic energy of electron, MeV
printf("\nThe total energy of electron = %5.3f MeV",
     E);
printf("\nThe kinetic energy of electron = %5.3f MeV
    ", K);

// Result
// The total energy of electron = 0.970 MeV
// The kinetic energy of electron = 0.459 MeV
```

Scilab code Exa 2.4 Energy of a speedy proton

```
// Scilab code Ex2.4: Pg.47 (2005)
clc; clear;
c = 3e+08;     // Velocity of light, m/s
u = 0.85*c;     // Velocity of electron, m/s
m_p = 1.67e-27;    // Rest mass of proton, kg

// Part (a)
E_o = m_p*c^2/1.602e-019;    // Rest energy of proton
    , MeV
```

```
printf("\nRest energy of proton = %3d MeV", E_o/1e+06);

// Part (b)
// Since given that E = 3*E_o = (3*m_p*c^2)/sqrt(1-(u/c)^2), solving for u
u = sqrt(8/9)*c;    // Velocity of proton, m/s
printf("\nVelocity of proton = %4.2fe+08 m/s", u*1e-08);

// Part (c)
// Since K = E - m_p*(c^2) = 3*m_p*(c^2) - m_p*(c^2) = 2*m_p*(c^2)
K = 2*E_o;    // Kinetic energy of proton, MeV
printf("\nThe kinetic energy of proton = %4d MeV", K*1e-06);

// Part (d)
p = sqrt(8)*(E_o);
printf("\nThe momentum of proton = %4d MeV/c", p*1e-06);

// Result
// Rest energy of proton = 938 MeV
// Velocity of proton = 2.83e+08 m/s
// The kinetic energy of proton = 1876 MeV
// The momentum of proton = 2653 MeV/c
```

Scilab code Exa 2.5 Increase in mass of colliding balls

```
// Scilab code Ex2.5: Pg.49 (2005)
clc; clear;
u = 450;     // Velocity ofeach ball, m/s
m = 5;     // Mass of each ball, kg
c = 3e+08;      // Velocity of light, m/s
```

```
// Since (u/c)^2 << 1, therefore, substituting 1/sqrt(1 - (u/c)^2) = 1 + (1/2)*(u/c)^2
delta_M = m*(u/c)^2;    // Increase in the mass of balls, kg
printf("\nIncrease in the mass of the balls = %3.1fe-11 kg", delta_M*1e+11);

// Result
// Increase in the mass of the balls = 1.1e-11 kg
```

Scilab code Exa 2.7 A Fission Reaction

```
// Scilab code Ex2.7:  Pg.50 (2005)
clc; clear;
u = 1.660e-27;    // Atomic mass unit
M_U = 236.045563;    // Atomic mass of Uranium, u
M_Rb = 89.914811;    // Atomic mass of Rubidium, u
M_Cs = 142.927220;    // Atomic mass of Caesium, u
m_n = 1.008665;     // Mass of neutons, u

// Part (a)
printf("\nU(92,235) --> Rb(37,90) + Cs(55,143) + 3n(0,1)");
printf("\nSo three neutrons are produced per fission.\n");

// Part (b)
delta_M = (M_U - (M_Rb + M_Cs + 3*m_n))*u;    // Combined mass of all products, kg
printf("\nCombined mass of all products = %6.4fe-28 kg\n", delta_M*1e+28);

// Part (c)
// For simplification let velocity of light = 1 m/s
c = 1;    // Velocity of light, m/s
```

```
// Since 1u = 931.5 MeV/(c^2), therefore
Q = (delta_M/u)*931.5*(c^2);     // Energy given out per fission event, MeV
printf("\nEnergy given out per fission event = %5.1f MeV\n", Q);

// Part (d)
N = ((6.02e23)*1000)/236;    // Number of nuclei present
efficiency = 0.40;
E = efficiency*N*Q*(4.45e-20);    // Total energy released, kWh
printf("\nTotal energy released = %4.2fe+06 kWh\n", E*1e-06);

printf("\nThis amount of energy will keep a 100-W lightbulb burning for %d years", E*1000/(100*24*365));

// Result
// U(92,235) --> Rb(37,90) + Cs(55,143) + 3n(0,1)
// So three neutrons are produced per fission.
// Combined mass of all products = 2.9471e-28 kg
// Energy given out per fission event = 165.4 MeV
// Total energy released = 7.51e+06 kWh
// This amount of energy will keep a 100-W lightbulb burning for 8571 years
```

Scilab code Exa 2.8 Energy conservation

```
// Scilab code Ex2.8: Pg.52 (2005)
clc; clear;

// Part (a)
// Since 1 eV = 1.6e-19 J, therefore 3 eV = 3*1.6e
```

```
−19
BE = 3*1.6e-19;    // Binding energy of water, J
c = 3e+08;    // Velocity of light, m/s
delta_m = BE/(c^2);    // Mass difference of water molecule & it constituents, kg
printf("\nMass difference of water molecule & it constituents = %3.1fe−36 kg", delta_m*1e+36);

// Part (b)
M = 3.0e-26;    // Mass of water molecule, kg
M_f = delta_m/M;    // Fractional loss of mass per molecule
printf("\nThe fractional loss of mass per molecule = %3.1fe−10", M_f*1e+10);

// Part (c)
E = M_f*(c^2);    // Energy released when 1 g of water is formed, kJ
printf("\nEnergy released when 1 g of water is formed = %2.0f kJ", E*1e-06);

// Result
// Mass difference of water molecule & it constituents = 5.3e−36 kg
// The fractional loss of mass per molecule = 1.8e−10
// Energy released when 1 g of water is formed = 16 kJ
```

Scilab code Exa 2.9 Mass of Pion

```
// Scilab code Ex2.9: Pg.53 (2005)
clc; clear;
K_mu = 4.6;    // Kinetic energy of muon, MeV
// For convinience let m_mew*(c^2) = E_mew
```

```
E_mu = 106;        // Energy of muon, MeV
E_pion = sqrt((E_mu^2) + (K_mu^2) + (2*K_mu*E_mu)) +
    sqrt((K_mu^2) + (2*K_mu*E_mu));
m_pion = E_pion;    // Mass of pion, MeV/(c^2)
printf("\nMass of Pion = %3.0f MeV/(c^2)", m_pion);

// Result
// Mass of Pion = 142 MeV/(c^2)
```

Chapter 3

The Quantum theory of light

Scilab code Exa 3.1 Temperature of sun

```
// Scilab code Ex3.1: Pg 69 (2005)
clc; clear;
e_total = 1400;   // Total power per unit area, W/(m
    ^2)
sigma = 5.6e-08;    // Stefan-Boltzmann constant
R = 1.5e+11;    // Earth-sun distance, m
R_s = 7.0e+08;   // Radius of sun, m
// Using Stefan's law & solving for T, we get
T = (e_total*R^2/(sigma*R_s^2))^0.25;   //
    Temperature of sun, K
printf("\nThe temperature of sun = %4d K", T);

// Result
// The temperature of sun = 5820 K
```

Scilab code Exa 3.2 Quantum oscillator vs classical oscillator

```
// Scilab code Ex3.2: Pg 75 (2005)
```

```
clc; clear;

// Part (a)
h = 6.63e-34;    // Plank's constant, Js
c = 3e+08;    // Velocity of light, m/s
lamda_green = 540e-09;    // Wavelength of green light, nm
delta_E_green = h*c/lamda_green/1.602e-19;    // Minimum energy change in green light, eV
lamda_red = 700e-09;    // Wavelength of red light, nm
delta_E_red = h*c/lamda_red/1.602e-19;    // Minimum energy change in red light, eV

printf("\nMinimum energy change in green light = %4.2f eV", delta_E_green);
printf("\nMinimum energy change in red light = %4.2f eV", delta_E_red);

// Part (b)
f = 0.50;    // Frequency, Hz
m = 0.1;     // Mass of pendulum, kg
l = 1;       // Length of pendulum, m
theta = %pi/180*10;    // Angle, radians
g = 9.8;    // Acceleration due to gravity, m/s^2
E = m*g*l*(1-cos(theta));
delta_E = (h*f)/(1.6e-19);    //  Minimum energy change in pendulum, eV
delta_E_f = (delta_E*1.6e-19)/E ;    // Fractional energy change
printf("\nFractional energy change = %3.1fe-32", delta_E_f*1e+32);

// Result
// Minimum energy change in green light = 2.30 eV
// Minimum energy change in red light = 1.77 eV
// Fractional energy change = 2.2e-32
```

Scilab code Exa 3.3 Stefan law from Planck distribution

```
// Scilab code Ex3.3: Pg 80 (2005)
clc; clear;
k_B = 1.381e-23;    // Boltzmann's constant, J/K
c = 3e+08;     // Velocity of light, m/s
h = 6.626e-34;     // Plank's constant, Js
// Since e_total = sigma*(T^4) = (2*(%pi)^5*(k_B)^4)/(15*(c^2)*(h^3))*T^4
sigma = (2*(%pi)^5*(k_B)^4)/(15*(c^2)*(h^3));
printf("\nThe value of sigma = %3.2fe-08 W/Sq.m/K^4", sigma*1e+08);

// Result
// The value of sigma = 5.67e-08 W/Sq.m/K^4
```

Scilab code Exa 3.4 Time lag between start of illumination and photocurrent generation

```
// Scilab code Ex3.4: Pg 83 (2005)
clc; clear;
phi = 2.38;     // Work function for sodium, eV
I = 1e-07;     // Absorbed light intensity, mJcm^2/s
A = %pi*1e-16;     // Cross-sectional area, m^2
t = phi*1.6e-16/(I*A)    // Time lag, days
printf("\nTime lag between start of illumination and photocurrent generation = %3.1fe+07 s", t*1e-07);

// Result
// Time lag between start of  illumination and photocurrent generation = 1.2e+07 s
```

Scilab code Exa 3.5 Time lag between start of illumination and photocurrent generation

```
// Scilab code Ex3.5: Pg 85 (2005)
clc; clear;
e = 1.6e-19;     // Electric charge, C
V_s = 4.3;    // Stopping potential, V
K_max = e*V_s;   // Maximum kinetic energy attained by photoelectrons, J
m_e = 9.11e-31   // Mass of electron, kg
// Since K.E = eV_s = 0.5m_e(v_max^2), therefore
v_max = sqrt((2*K_max)/m_e);     // Maximum velocity attained by photoelectron, m/s
printf("\nMaximum velocity attained by photoelectron = %3.1fe+06 m/s", v_max*1e-06);

// Result
// Maximum velocity attained by photoelectron = 1.2e+06 m/s
```

Scilab code Exa 3.6 Photoelectric effect for iron

```
// Scilab code Ex3.6: Pg 85 (2005)
clc; clear;

// Part (a)
I_o = 1;     // Total intensity of light, micro-W/cm^2
I = (0.030)*(.040)*I_o;     // Intensity available to produce photoelectric effect, nW/cm^2
printf("\nIntensity available to produce photoelectric effect = %3.1f nW/cm^2", I*1e+03);
```

```

// Part (b)
h = 6.6e-34;    // Planck's constant, Js
c = 3e+08;     // Velocity of light, m/s
lamda = 250e-09;     // Wavelength, m
e_per_sec = (I*lamda*1e-06)/(h*c);    // Number of electrons emitted per second
printf("\nNumber of electrons emitted per second = %3.1e", e_per_sec);

// Part (c)
e = 1.6e-019;    // Energy equivalent of 1 eV, C
i = (e_per_sec)*e;      // Electric current in phototube, A
printf("\nElectric current in phototube = %3.1e A", i);

// Part (d)
f_o = 1.1e+15;    // Cut-off frequeny, Hz
phi = (h*f_o)/e;    // Work function for iron, eV
printf("\nWork function for iron = %3.1f eV", phi);

// Part (e)
V_s = (h*c/(e*lamda))-phi;         // Stopping voltage, V
printf("\nStopping voltage = %4.2f V", V_s);

// Result
// Intensity available to produce photoelectric effect = 1.2 nW/cm^2
// Number of electrons emitted per second = 1.5e-09
// Electric current in phototube = 2.4e-10 A
// Work function for iron = 4.5 eV
// Stopping voltage = 0.41 V
```

Scilab code Exa 3.7 Compton shift for carbon

```
// Scilab code Ex3.7: Pg 93 (2005)
clc; clear;
h = 6.63e-34;    // Plank's constant, Js
m_e = 9.11e-31;    // Mass of electron, kg
c = 3e+08;     // Velocity of light, m/s
theta = ((%pi)/180)*45;     // Angle, radians
delta_lamda = (h/(m_e*c)*(1-cos(theta)));    // Compton shift, nm
lamda_o = 0.200e-09;     // Wavelength of X-ray, nm
lamda = delta_lamda+lamda_o    // Increased wavelength of scattered X-ray, nm
printf("\nIncreased wavelength of scattered X-ray = %8.6f nm", lamda*1e+09);

// Result
// Increased wavelength of scattered X-ray = 0.200711 nm
```

Scilab code Exa 3.8 Xray photons vs visible photons

```
// Scilab code Ex3.8: Pg 93 (2005)
clc; clear;

// Part (a)
h = 6.63e-34;     // Plank's constant, Js
q = 1.6e-19;     // Electric charge, C
m_e = 9.11e-31;     // Mass of electron, kg
c = 3e+08;      // Velocity of light, m/s
theta = ((%pi)/180)*90;     // Angle, radians
delta_lamda = (h/(m_e*c)*(1-cos(theta)));                // Compton shift, Angstrom
lamda_C = 0.0106;    // Wavelength of gamma-rays from Cobalt,
```

```
f_dl_C = delta_lamda/ lamda_C;    // Fractional change in wavelength of gamma rays from cobalt
printf("\nFractional change in wavelength of gamma rays from Cobalt = %4.2f", f_dl_C*1e+10);
lamda_Mo = 0.712;     // Wavelength of gamma-rays from Molybdenum, Angstrom
f_dl_Mo = delta_lamda/ lamda_Mo;                    // Fractional change in wavelength of gamma rays from Molybdenum
printf("\nFractional change in wavelength of gamma rays from Molybdenum = %6.4f", f_dl_Mo*1e+10);
lamda_Hg = 5461;     // Wavelength of gamma-rays from Mercury, Angstrom
f_dl_Hg = delta_lamda/ lamda_Hg;                    // Fractional change in wavelength of gamma rays from mercury
printf("\nFractional change in wavelength of gamma rays from Mercury = %4.2fe-06", f_dl_Hg*1e+16);

// Part (b)
lamda = 0.712e-10;    // Wavelength of X-rays, Angstrom
E = (h*c)/(q*lamda);    // Energy of X-rays' photon, eV
printf("\nEnergy of X-rays photon = %5.0f eV\n", E);

// Result
// Fractional change in wavelength of gamma rays from Cobalt = 2.29
// Fractional change in wavelength of gamma rays from Molybdenum = 0.0341
// Fractional change in wavelength of gamma rays from Mercury = 4.45fe-06
// Energy of X-rays photon = 17460 eV
```

Scilab code Exa 3.9 Gravitational redshift for a white dwarf

```
// Scilab code Ex3.9: Pg 96 (2005)
clc; clear;
M = 1.99e+30;    // Mass of sun, kg
R_s = 6.37e+06;     // Radius of earth, m
G = 6.67e-11;    // Gravitational constant, Nm^2/kg^2
lamda = 300e-09;    // Wavelength, m
c = 3e+08;      // Velocity of light, m/s
delta_lamda = lamda*((G*M)/(R_s*c^2));    //
    Gravitational redshift, angstrom
printf("\nGravitational redshift = %3.1f angstrom",
    delta_lamda*1e+10);

// Result
// Gravitational redshift = 0.7 angstrom
```

Chapter 4

The particle nature of matter

Scilab code Exa 4.1 Electrolysis of barium chloride

```
// Scilab code Ex4.1: Pg 109 (2005)
clc; clear;
I = 10;    // Electric current, A
t = 3600;    // Time, s
q = I*t;    // Electric charge liberated, C
mm_Ba = 137;    // Molar mass of Barium, g
mm_Cl = 35.5;    // Molar mass of Chlorine, g
valence_Ba = 2;    // Valence electrons of Barium
valence_Cl = 1;    // Valence electrons of Chlorine
// Using Faraday s law of electrolysis, we have
m_Ba = (q*mm_Ba)/(96500*valence_Ba);    // Mass of Barium obtained, g
m_Cl = (q*mm_Cl)/(96500*valence_Cl);    // Mass of Chlorine obtained, g
printf("\nMass of Barium obtained = %4.1f g", m_Ba);
printf("\nMass of Chlorine obtained = %4.1f g", m_Cl);

// Result
// Mass of Barium obtained = 25.6 g
```

```
// Mass of Chlorine obtained = 13.2 g
```

Scilab code Exa 4.2 Deflection of electron beam by E and B Fields

```
// Scilab code Ex4.2: Pg 113 (2005)
clc; clear;
 V = 200;     // Electrc potential, V
theta = 0.20;    // Angle, radians
l = 0.050;     // Length of plates, m
d = 1.5e-02;     // Distance between two plates, m
c_m_r = 1.76e+11;     // Charge-to-mass ratio, C/kg
// Since e/m_e = (V*theta)/(B^2*l*d), solving for B
B = sqrt((V*theta)/(l*d*c_m_r));     // Magnetic field, T
printf("\nThe magnetic field required to produce the deflection of %4.2f rad = %3.1e T", theta, B);

// Result
// The magnetic field required to produce the deflection of 0.20 rad = 5.5e-04 T
```

Scilab code Exa 4.3 Experimental determination of e

```
// Scilab code Ex4.3: Pg 117 (2005)
clc; clear;

// Part (a)
delta_y = 0.600;     // Distance of rise or fall of a droplet, cm
t_av = 21.0;     // Average time of fall of droplet, s
delta_t = [46.0, 15.5, 28.1, 12.9, 45.3, 20.0];    // Rise time of the droplet in succession, s
```

```
v = delta_y/t_av;    // Average speed of the falling droplet, cm/s
v_prime = zeros(6);
for i = 1:1:6
    v_prime(i) = delta_y/delta_t(i);    // Successive speeds of the rising drops, cm/s
end

// Calculate charge ratios
q1byq2 = (v+v_prime(1))/(v + v_prime(2));
q2byq3 = (v+v_prime(2))/(v + v_prime(3));
q3byq4 = (v+v_prime(3))/(v + v_prime(4));
q4byq5 = (v+v_prime(4))/(v + v_prime(5));
q5byq6 = (v+v_prime(5))/(v + v_prime(6));
printf("\nq1/q2 = %5.3f", q1byq2);
printf("\nq2/q3 = %5.3f", q2byq3);
printf("\nq3/q4 = %5.3f", q3byq4);
printf("\nq4/q5 = %5.3f", q4byq5);
printf("\nq5/q6 = %5.3f", q5byq6);
printf("\nThe charge ratios are ratios of small whole numbers\n");

// Part (b)
eta = 1.83e-05;                             // Viscosity of air, kg/ms
rho = 858;                                  // Oil density, kg/m^3
g = 9.81;                                   // Acceleration due to gravity, m/s^2
a = sqrt((9*eta*v*1e-02)/(2*rho*g));    // Radius of oil droplet, m
V = 4/3*(%pi)*a^3;                          // Volume of oil droplet, m^3
m = rho*V;                                  // Mass of oil droplet, kg
printf("\nRadius of oil droplet = %4.2e m", a);
printf("\nVolume of oil droplet = %4.2e m^3", V);
printf("\nMass of oil droplet = %4.2e kg", m);
```

```
// Part (c)
V = 4550;    // Potential difference across the plates of the capacitor, volt
d = 0.0160;    // Distance between the plates
E = V/d;    // Electric field between plates, V/m
q = zeros(6), e = zeros(6);
for i=1:1:6
    q(i) = m*g/E*((v+v_prime(i))/v);    // Charge on first drop, C
    printf("\nq%d = %4.2e V/m", i, q(i));
end
e(1) = q(1)/5;
e(2) = q(2)/8;
e(3) = q(3)/6;
e(4) = q(4)/9;
e(5) = q(5)/5;
e(6) = q(6)/7;
e_tot = 0;
for i = 1:1:6
    e_tot = e_tot + e(i);
end
e = e_tot/6;
printf("\nThe average charge on an electron = %5.3e C", e);

// Result
// q1/q2 = 1.105
// q2/q3 = 0.958
// q3/q4 = 1.053
// q4/q5 = 0.899
// q5/q6 = 1.086
// The charge ratios are ratios of small whole numbers

// Radius of oil droplet = 1.67e-06 m
// Volume of oil droplet = 1.96e-17 m^3
// Mass of oil droplet = 1.68e-14 kg
```

```
// q1 = 8.44e-019 V/m
// q2 = 1.36e-018 V/m
// q3 = 1.01e-018 V/m
// q4 = 1.52e-018 V/m
// q5 = 8.48e-019 V/m
// q6 = 1.19e-018 V/m
// The average charge on an electron = 1.694e-019 C
```

Scilab code Exa 4.4 Collision of alpha particle with proton

```
// Scilab code Ex4.4: Pg 121 (2005)
clc; clear;

// Part (b)
// For easy calculations, assume all variables to be
    unity
m_p = 1;      // Mass of proton, a.m.u
m_a = 4*m_p;     // Mass of alpha particle, a.m.u
Valpha = 1;    // Velocity of alpha particle before
    collision, m/s
v_p = (2*m_a*Valpha)/(m_a + m_p);     // Velocity of
    proton after collision, m/s
v_a = ((m_a - m_p)*(Valpha))/(m_a + m_p);      //
    Velocity of alph particle after collision, m/s
p_change = ((v_a - Valpha)/(Valpha))*100;      //
    Percentage change in velocity of alpha particle
printf("\nVelocity of proton after collision = %4.2
    fVa m/s", v_p);
printf("\nVelocity of alpha particle after collision
    = %4.2fVa m/s", v_a);
printf("\nPercentage change in velocity of alpha
    particle = %2d percent", p_change);

// Result
```

```
// Velocity of proton after collision = 1.60 V_a m/s
// Velocity of alph particle after collision = 0.60 V_a m/s
// Percentage change in velocity of alpha particle = -40 percent
```

Scilab code Exa 4.5 Radius of Aluminium Nucleus

```
// Scilab code Ex4.5: Pg 124 (2005)
clc; clear;
Z = 13;     // Atomic number of Aluminium
e = 1.6e-19;      // Charge on electron, C
k = 8.99e+09;      // Coulomb constant, Nm^2/C^2
K_a = 7.7e+06*e;     // Since K_a = (k*Z_e*2*e)/d_min, solving for d_min
d_min = (k*2*Z*e^2)/K_a;    // Radius of Aluminum, m
printf("\nRadius of Aluminum = %3.1e m", d_min);

// Result
// Radius of Aluminum = 4.9e-15 m
```

Scilab code Exa 4.7 Collision of alpha particle with proton

```
// Scilab code Ex4.7: Pg 135 (2005)
clc; clear;
// Part (a)
n_i = 2;    // Initial level of electron
n_f = 1;    // Final level of electron
R = 1.097e+07;      // Rydberg constant, per metre
c = 3e+08;      // Velocity of light, m/s
h = 4.136e-15;        // Planck's constant, eV
lamda = n_i^2*n_f^2/((n_i^2-n_f^2)*R);    // Wavelength of emitted photon, m
```

```
f = c/lamda;        // Frequency of emitted photon, Hz
E = h*f;      // Energy of emitted photon, eV
printf("\nThe wavelength of emitted photon = %5.1f nm", lamda/1e-09);
printf("\nThe frequency of emitted photon = %4.2e Hz", f);
printf("\nEnergy of emitted photon = %4.1f eV", E);

// Part (b)
mc_square = 938.8e+06;    // Energy of recoil of hydrogen atom, eV
K = 0.5*(E^2/mc_square);      // Recoil kinetic energy of H atom, eV
E_difference = K/E;    // Energy difference
printf("\nRecoil kinetic energy of H atom = %4.2e eV", K);
printf("\nThe fraction of energy difference = %3.1e", E_difference);

// Result
// The wavelength of emitted photon = 121.5 nm
// The frequency of emitted photon = 2.47e+15 Hz
// Energy of emitted photon = 10.2 eV
// Recoil kinetic energy of H atom = 5.55e-08 eV
// The fraction of energy difference = 5.4e-09
```

Scilab code Exa 4.8 series for Hydrgen

```
// Scilab code Ex4.8: Pg 136 (2005)
clc; clear;

// Part (a)
n_i = 3;      // Initial level of electron
n_f = 2;      // Final level of electron
R = 1.097e+07;    // Rydberg constant, per metre
```

```
c = 3e+08;        // Velocity of light, m/s
h = 6.626e-34;    // Plank's constant, Js
lamda_max = (n_i^2*n_f^2)/((n_i^2-n_f^2)*R);       // Maximum wavelength of emitted photon, m
E_photon = (h*c)/(lamda_max*1.6e-19);         // Energy of emitted photon, eV
printf("\nThe maximum wavelength of emitted photon = %5.1f nm", lamda_max/1e-09);
printf("\nEnergy of emitted photon = %4.2f eV", E_photon);

// Part (b)
n_i = %inf;     // Initial level of electron
lamda_min = 1/(R*(1/n_f^2-1/n_i^2));
printf("\nThe wavelength corresponding to the series limit = %5.1f nm which is in the ultraviolet region", lamda_min/1e-09);

// Result
// The maximum wavelength of emitted photon = 656.3 nm
// Energy of emitted photon =1.89 eV
//// The wavelength corresponding to the series limit = 364.6 nm which is in the ultraviolet region
```

Scilab code Exa 4.9 Hydrogen in its first excited state

```
// Scilab code Ex4.9: Pg 137 (2005)
clc; clear;
k_B = 8.62e-05;    // Boltzmann constant, eV/K
delta_E = 10.2;     // Average thermal energy, eV
// Since (3/2)*k_B*T = averge thermal energy per atom = 10.2eV, solving for T
T = 10.2/(3/2*k_B);        // Temperature at which H-
```

```
atoms jump to first excited state, K
printf("\nThe temperature at which H-atoms jump to first excited state = %5d K", T);
N_ratio = 0.10;     // Number ratio of population of first excited state relative to the ground state
// As N_ratio = exp(-delta_E/(k_B*T)), solving for T
T = -delta_E/(k_B*log(N_ratio));     // Temperature at which H-atoms jump to first excited state, K
printf("\nThe temperature of excitation from Boltzmann distribution = %5d K", T);

// Result
// The temperature at which H-atoms jump to first excited state = 78886 K
// The temperature of excitation from Boltzmann distribution = 51389 K
```

Chapter 5

Matter waves

Scilab code Exa 5.1 Wave properties of a baseball

```
// Scilab code Ex5.1: Pg 154 (2005)
clc; clear;
h = 6.63e-34;      // Plank's constant, Js
m = 140e-03;      // Mass of baseball, kg
v = 27;      // Velocity of baseball, m/s
p = m*v;    // Momentum of baseball, kgm/s
lamda = h/p;    // de Broglie wavelength associated
    with baseball, m
printf("\nde-Broglie wavelength associated with
    baseball = %3.1e m", lamda);

// Result
// de-Broglie wavelength associated with baseball =
    1.8e-34 m
```

Scilab code Exa 5.2 de Broglie wavelength of an electron

```
// Scilab code Ex5.2: Pg 154 (2005)
```

```
clc; clear;

// Part (b)
h = 6.63e-34;      // Plank's constant, Js
m_e = 9.11e-31;    // Mass of electron, kg
q = 1.6e-19;      // Charge on electron, C
V = 50;     // Electric potential applied, V
lamda = h/(sqrt(2*m_e*q*V));     // de Broglie wavelength of an electron, m
printf("\nde Broglie wavelength of an electron = %3.1f angstrom", lamda/1e-10);

// Result
// de Broglie wavelength of an electron = 1.7 angstrom
```

Scilab code Exa 5.3 Diffraction of neutrons at the crystal lattice

```
// Scilab code Ex5.3: Pg 158 (2005)
clc; clear;
h = 6.63e-34;     // Plank's constant, J-s
lamda = 1e-10;      // de Broglie wavelength of neutron, m
p = h/lamda;      // Momentum associated with neutron, kg-m/s
m_n = 1.66e-27;      // Mass of neutron, kg
e = 1.6e-19;     // Energy equivalent of 1 eV, J/eV
K = p^2/(2*m_n);     // Kinetic energy of neutron, eV
printf("\nThe momentum of neutrons = %4.2e kg-m/s", p)
printf("\nThe kinetic energy of neutrons = %4.2fe-20 J = %6.4f eV", K*1e+20, K/e);

// Result
// The momentum of neutrons = 6.63e-24 kg-m/s
```

```
// The kinetic energy of neutrons = 1.32e-20 J =
   0.0828 eV
```

Scilab code Exa 5.8 Uncertainity principle for macroscopic objects

```
// Scilab code Ex5.8: Pg 177 (2005)
clc; clear;
h_cross = 1.05e-34;    // Reduced Plank's constant ,
   J-s
delta_x = 15;        // Uncertainity in position , m
v_x = 2;        // Velocity of ball , m/s
m = 100e-03;      // Mass of ball , kg
delta_p_x = h_cross/(2*delta_x);    // Uncertainity
   in momentum, kg-m/s
delta_v_x = delta_p_x/m;      // Minimum spread in
   velcoity , m/s
U_r = delta_v_x/v_x;      // Relative uncertainity in
    velocity of ball
printf("\nThe minimum spread in velcoity of ball =
   %3.1e m/s", delta_v_x);
printf("\nThe relative uncertainity in velocity of
   ball = %4.2e", U_r);

// Result
// The minimum spread in velcoity of ball = 3.5e-35
   m/s
// The relative uncertainity in velocity of ball =
   1.75e-35
```

Scilab code Exa 5.9 Kinetic energy of electron confined within the nucleus

```
// Scilab code Ex5.9: Pg 178 (2005)
clc; clear;
```

```
delta_x = 1.0e-14/2;                              // Uncertainity in position of electron, m
q = 1.6e-19;                                      // Charge on electron, C
h_cross = 1.05e-34;                               // Reduced Plank's constant, J-s
c = 3e+08;                                        // Velocity of light, m/s
delta_p_x = (h_cross*c)/(2*delta_x*q);            // Uncertainity in momentum, eV/c
E_r = 0.551e+06;                                  // Rest mass energy if electron, eV
E = sqrt((delta_p_x)^2 + (E_r)^2);
K = E - E_r;                                      // Kinetic energy of electron within nucleus, eV
printf("\nKinetic energy of electron within nucleus = %4.1f MeV", K/1e+06);

// Result
// Kinetic energy of electron within nucleus = 19.1 MeV
```

Scilab code Exa 5.10 Width of spectral lines

```
// Scilab code Ex5.10: Pg 178 (2005)
clc; clear;

// Part (a)
h_cross = 1.05e-34;    // Reduced Plank's constant, J-s
h = 6.63e-34;     // Plank's constant, J-s
delta_t = 1.0e-08;     // Average time to measure the excited state, s
delta_E = h_cross/(2*delta_t);    // Uncertainty in
```

```
energy of the excited state, J
// Since delta_E = h*delta_f, solving for delta_f
delta_f = delta_E/h;    // Line width of emitted light, Hz
printf("\nLine width of emitted light = %2.0e Hz", delta_f);

// Part (b)
c = 3e+08;    // Velocity of light, m/s
lamda = 500e-09;    // Wavelength of spectral line, m
f_o = c/lamda;    // Center frequency of spectral line, Hz
f_b = delta_f/f_o;    // Fractional broadening of spectral line
printf("\nFractional broadening of spectral line = %3.1e", f_b);

// Result
// Line width of emitted light = 8.0e+06 Hz
// Fractional broadening of spectral line = 1.3e-08
```

Chapter 6

Quantum mechanics in one dimension

Scilab code Exa 6.2 Probability from wave function

```
// Scilab code Ex6.2: Pg 193 (2005)
clc; clear;
x0 = 1;    // For simplicity assume x0 = 1
C = 1/sqrt(x0);    // Normalization constant
P = 2*C^2*integrate('exp(-2*x/x0)', 'x', 0, x0);
printf("\nThe probability that the particle will be
    found in the interval -x0 <= x <= x0 is %6.4f or
    %4.1f percent", P, P*100);

// Result
// The probability that the particle will be found
    in the interval -x0 <= x <= x0 is 0.8647
```

Scilab code Exa 6.4 Dispersion of matter waves

```
// Scilab code Ex6.4: Pg 197 (2005)
```

```
clc; clear;
delta_x0 = 1e-010;      // Initial width of the localized space, m
delta_xt = 10*delta_x0;      // Final width at which the wave packet is dispersed, m
h_cross = 1.055e-034;      // Reduced Planck's constant, Js
m = 9.11e-031;      // Mass of the electron, kg
// From Dispersion relation, delta_xt^2 - delta_x0^2 = sqrt(h_cross*t/(2*m*deltax0)^2), solving for t
t = 2*m*sqrt(delta_xt^2 - delta_x0^2)*delta_x0/h_cross;   // Time which elapses before delocalization
printf("\nThe time which elapses before the localization of electron destroys = %3.1e s", t);
m = 1e-03;      // Mass of marble, kg
delta_x0 = 1e-004;      // Initial width of the localized space, m
delta_xt = 10*delta_x0;      // Final width at which the wave packet is dispersed, m
t = 2*m*sqrt(delta_xt^2 - delta_x0^2)*delta_x0/h_cross;   // Time which elapses before delocalization
printf("\nThe time which elapses before the localization of marble destroys = %3.1e s", t);
printf("\nFor all the practical purposes, the marble will remain localized for ever");
// Result
//
```

Scilab code Exa 6.5 Energy Quantization for Macroscopic Object

```
// Scilab code Ex6.5: Pg 202 (2005)
clc; clear;
h = 6.626e-034;      // Planck's constant, Js
```

```
m = 1e-06;      // Mass of the object , kg
n = 1;      // Quantum number for minimum energy level
L = 1e-02;       // Distance between two rigid walls ,
    m
E1 = n^2*h^2/(8*m*L^2);   // Minimum energy of the
    object , J
v1 = sqrt(2*E1/m);     // Minimum speed of the object
    , m/s
v = 3.00e-02;      // Given speed of the objct , m/s
E = 1/2*m*v^2;   // Energy of the object for given
    speed , J
n = sqrt(8*m*L^2*E)/h;   // Quantum number
    corresponding to the given speed
printf("\nThe minimum speed of the object = %4.2e m/
    s", v1);
printf("\nThe quantum number corresponding to the
    speed of %4.2e m/s is n = %4.2e", v1, n);

// Result
//  The minimum speed of the object = 3.31e-26 m/s
// The quantum number corresponding to the speed of
    3.31e-26 m/s is n = 9.06e+23
```

Scilab code Exa 6.6 Model of an Atom

```
// Scilab code Ex6.6: Pg 203 (2005)
clc; clear;
c = 1;        // Assume speed of light to be unity , m/
    s
h_cross = 197.3;      // Reduced Planck's constant , eV
    .nm/c^2
m_e = 511e+03;            // Mass of an electron , eV/c
    ^2
L = 0.200;   // Length of the box , nm
E1 = %pi^2*(h_cross/c)^2/(2*m_e*L^2);    // Ground
```

```
    state energy of atomic electron , eV
E2 = 2^2*E1;    // Excited state energy of the
    atomic electron , eV
delta_E = E2- E1;   // Energy that must be applied
    to the electron  to raise it from ground to the
    first excited state , eV
h = 2*%pi*h_cross;  // Planck's constant , Js
lambda = h*c/delta_E;   // Wavelength of the photon
    to cause the electron transition , nm
printf("\nThe energy that must be applied to the
    electron  to raise it from ground to the first
    excited state = %4.1f eV", delta_E);
printf("\nThe wavelength of the photon to cause this
     electron transition = %4.1f nm", lambda);
printf("\nThis wavelength is in the far ultraviolet
    region.");

// Result
// The energy that must be applied to the electron
    to raise it from ground to the first excited
    state = 28.2 eV
// The wavelength of the photon to cause this
    electron transition = 44.0 nm
// This wavelength is in the far ultraviolet region.
```

Scilab code Exa 6.7 Probabilities for a particle in a Box

```
// Scilab code Ex6.7: Pg 205 (2005)
clc; clear;
L = 1;  // For simplicity assume length of finite
    square well to be unity , m
P = 2/L*integrate('sin(%pi*x/L)^2', 'x', L/4, 3*L/4)
    ; // Probability that the particle will be found
    in the middle half of the well
printf("\nThe probability that the particle will be
```

```
found in the middle half of the well = %5.3f", P)
;

// Result
// The probability that the particle will be found
in the middle half of the well = 0.818
```

Scilab code Exa 6.8 Ground state energy of an electron confined to a potential well

```

// Scilab code Ex6.8: Pg 211 (2005)
clc; clear;
c = 1;      // Assume speed of light to be unity, m/
s
L = 0.200;  // Width of the potential well, nm
h_cross = 197.3;    // Reduced Planck's constant, eV
.nm/c^2
m = 511e+03;            // Mass of an electron, eV/c^2
U = 100;    // Height of potential well, eV
delta = h_cross/sqrt(2*m*U);    // Decay length of
electron, nm
L = L + 2*delta;    // Effective length of the
infinite potential well, nm
E = %pi^2*(h_cross/c)^2/(2*m*L^2);  // Ground state
energy of the electron with effective length, eV
U = U - E;  // New potential energy, eV
delta = h_cross/sqrt(2*m*U);    // New decay length
of electron, nm
printf("\nThe ground state energy of an electron
confined to the potential well = %4.2f eV", E);
printf("\nThe new decay length of the electron = %6
.4f nm", delta);

// Result
```

```
// The ground state energy of an electron confined
    to the potential well = 6.58 eV
// The new decay length of the electron = 0.0202 nm
```

Scilab code Exa 6.12 The quantum oscillator in nonclassical region

```
// Scilab code Ex6.12: Pg 214 (2005)
clc; clear;
P = 2/sqrt(%pi)*integrate('exp(-z^2)', 'z', 1, 100);
        // Probability that the quantum oscillator in
    its ground state will be found in the
    nonclassical region
printf("\nThe probability that the quantum
    oscillator in its ground state will be found in
    the nonclassical region = %5.3f", P);

// Result
// The probability that the quantum oscillator in
    its ground state will be found in the
    nonclassical region = 0.157
```

Scilab code Exa 6.13 Quantization of vibrational energy

```
// Scilab code Ex6.13: Pg 211 (2005)
clc; clear;
h_cross = 6.582e-016;    // Reduced Planck's constant
    , eV-s
// For spring-mass system
K = 0.100;   // Force constant of the spring-mass
    system, N/m
m = 0.0100; // Mass attached to the spring, kg
omega = sqrt(K/m);   // Angular frequency of
    oscillations, rad/s
```

```
delta_E = h_cross*omega;     // Energy spacing
    between quantum levels , eV
printf("\nThe energy spacing between quantum levels
    for spring-mass system = %4.2e eV\nwhich is far
    below present limits of detection", delta_E);
// For vibrating hydrogen molecule
K = 510.5;   // Force constant of the hydrogen
    molecule system , N/m
mu = 8.37e-028; // Reduced mass of the hydrogen
    molecule , kg
omega = sqrt(K/mu);   // Angular frequency of
    oscillations , rad/s
delta_E = h_cross*omega;     // Energy spacing
    between quantum levels , eV
printf("\nThe energy spacing between quantum levels
    for hydrogen molecule = %5.3f eV\nwhich can be
    measured easily", delta_E);

// Result
// The energy spacing between quantum levels for
    spring-mass system = 2.08e-15 eV
// which is far below present limits of detection
// The energy spacing between quantum levels for
    hydrogen molecule = 0.514 eV
// which can be measured easily
```

Scilab code Exa 6.14 Standard Deviations from Averages

```
// Scilab code Ex6.14: Pg 219 (2005)
clc; clear;
x = [2.5, 3.7, 1.4, 7.9, 6.2, 5.4, 8.0, 6.4, 4.1,
    5.4, 7.0, 3.3, 4.2, 8.8, 6.2, 7.1, 5.4, 5.3]; //
    Data entries
sum_x = 0;       // Initialize the accumulator
sum_x_sq = 0;    // Initialize the second accumulator
```

```
N = 18;      // Total number of data points
for i = 1:1:N
    sum_x = sum_x + x(i);    // Sum of data
    sum_x_sq = sum_x_sq + x(i)^2; // Sum of square of data
end
x_av = sum_x/N;      // Average of data
x_sq_av = sum_x_sq/N;    // Mean square value
sigma = sqrt(x_sq_av-x_av^2);    // Standard deviation from averages
printf("\nThe standard deviation from averages = %4.2f", sigma);


// Result
// The standard deviation from averages = 1.93
```

Scilab code Exa 6.15 Location of a particle in the box

```
// Scilab code Ex6.15: Pg 219 (2005)
clc; clear;
L = 1;  // For simplicity assume length of the box to be unity, unit
x_av = 2*L/%pi^2*integrate('theta*sin(theta)^2', 'theta', 0, %pi);  // Average value of x
x_sq_av = L^2/%pi^3*(integrate('theta^2', 'theta', 0, %pi)-integrate('theta^2*cos(2*theta)', 'theta', 0, %pi)); // Average value of x square
delta_x = sqrt(x_sq_av - x_av^2);   // Uncertainty in the position for this particle, unit
printf("\nThe average position of the particle in the box = L/%1d", x_av*4);
printf("\nThe uncertainty in the position for the particle = %5.3fL", delta_x);

```

```
// Result
// The average position of the particle in the box =
    L/2
// The uncertainty in the position for the particle
    = 0.181L
```

Chapter 7

Tunnelling phenomena

Scilab code Exa 7.1 Transmission coefficient for an oxide layer

```
// Scilab code Ex7.1: Pg 235 (2005)
clc; clear;
c = 3e+08;    // Velocity of light, m/s
m_e = 511e+03/(c^2);    // Mass of electron, eV
U = 3.00;    // Ground state energy neglecting E, eV
h_cross = (1.973e+03)/c;    // Reduced planck's constant, eV
alpha = sqrt(2*m_e*U)/h_cross;
L = 50;    // Thickness of the layer, angstrom
T = 1/(1+1/4*10^2/(7*3)*sinh(alpha*L)^2);
printf("\nThe transmission coefficient for the layer thickness of");
printf("\n%2d angstrom = %5.3e", L, T);
L = 10;    // // Thickness of the layer, angstrom
T = 1/(1+1/4*10^2/(7*3)*sinh(alpha*L)^2);
printf("\n%2d angstrom = %5.3e", L, T);

// Result
// The transmission coefficient for the layer thickness of
// 50 angstrom = 9.628e-39
```

```
// 10 angstrom = 6.573e-08
```

Scilab code Exa 7.2 Tunnelling current through an oxide layer

```
// Scilab code Ex7.2: Pg 236 (2005)
clc; clear;
e = 1.60e-19;     // Electrc charge, C
i = 1.00e-03;       // Electron current, A
N = i/e;        // Electrons per second
T = 0.657e-07;      // Fraction of electrons
    transmitted
T_e = N*T;     // Number of electrons transmitted per
     second
T_i = T_e*e;     // Transmitted current, A
printf("\nThe transmitted current through the oxide
    layer = %4.1f pA", T_i*1e+12);

// Result
// The transmitted current through the oxide layer =
     65.7 pA
```

Scilab code Exa 7.5 Tunnelling in a parallel plate capacitor

```
// Scilab code Ex7.5: Pg 241 (2005)
clc; clear;
epsilon_c = 5.5e+10;     // Characteristic field
    strength, V/m
epsilon = 1.0e+09;     // Electric field, V/m
f = 1.0e+30;      // Collision frequency, s(-1)cm(-2)
lamda = f*exp(-epsilon_c/epsilon);    // Electron
    emission rate, electrons/sec
e = 1.60e-19;     // Electrc charge, C
I = lamda*e;     // Tunelling current, A
```

```
printf("\nTunelling current in parallel plate
    capacitor = %4.2f pA", I/1e-12);
printf("\n");

// Result
// Tunelling current in parallel plate capacitor =
    0.21 pA
```

Scilab code Exa 7.6 Estimating halflives of Thorium and Polonium

```
// Scilab code Ex7.6: Pg 244 (2005)
clc; clear;
Z_T = 88;       // Atomic number of daughter nucleus
E_T = 4.05e+06;       // Energy of ejected alphas, eV
R = 9.00e-15;       // Nuclear radius, m
r_o = 7.25e-15;       // Bohr radius, m
E_o = 0.0993e+06;       // Energy analogous to the
    Rydberg in Atomic Physics
T_T = exp(-4*%pi*Z_T*sqrt(E_o/E_T) + 8*sqrt((Z_T*R)/
    r_o));     // Transmission factor in case of
    Thorium
f = 1e+21;     // Frequency of collisions, Hz
lamda_T = f*T_T;     // Decay rate in case of Thorium
    , s^(-1)
t_T = 0.693/lamda_T;       // Half-life time of
    Thorium, s
Z_P = 82;     // Atomic number of daughter nucleus
E_P = 8.95e+06;       // Energy of ejected alphas, eV
R = 9.00e-15;       // Nuclear radius, m
r_o = 7.25e-15;       // Bohr radius, m
E_o = 0.0993e+06;       // Energy unit, eV
T_P = exp(-4*%pi*Z_P*sqrt(E_o/E_P) + 8*sqrt((Z_P*R)/
    r_o));     // Transmission factor in case of
    Polonium
f = 1e+21;     // Frequency of collisions, Hz
```

```
lamda_P = f*T_P;        // Decay rate in case of Thorium, s^(-1)
t_P = 0.693/lamda_P;     // Half-life time of Polonium, s

printf("\nHalf-life time of Thorium = %3.1e s = %3.1e yrs", t_T, t_T/(365*24*60*60));
printf("\nHalf-life time of Polonium = %3.1e s", t_P);

// Result
// Half-life time of Thorium = 5.3e+17 s = 1.7e+10 yrs
// Half-life time of Polonium = 8.4e-10 s
```

Chapter 8

Quantum Mechanics in Three Dimensions

Scilab code Exa 8.4 Orbital quantum number for a stone

```
// Scilab code Ex8.4: Pg 270 (2005)
clc; clear;
R = 1.00;       // Radius of circle, m
T = 1.00;       // Time period of revolution, s
v = (2*%pi*R)/T;      // Speed of stone in its orbit, m/s
m = 1.00;       // Mass of stone, kg
L = m*v*R;      // Angular momentum of stone, kg-m^2/s
h_cross = 1.055e-34;      // Reduced Planck's constant, kg-m^2/s
l = L/h_cross;      // Orbtal quantum number
printf("\nThe orbtal quantum number for stone = %4.2fe+34", l*1e-34);

// Result
// Orbtal quantum number for stone = 5.96e+34
```

Scilab code Exa 8.6 Space quantisation for an atomic electron

```
// Scilab code Ex8.6: Pg 272 (2005)
clc; clear;
// For simplicity let h_cross = 1
h_cross = 1;    // Reduced planck's constant
l = 3;   // Given orbital quantum number
L = sqrt(l*(l+1)*h_cross);     // Magnitude of total angular momentum, in h_cross units
m_l = [-3, -2, -1, 0, 1, 2, 3];
L_z = m_l*h_cross;   // Allowed values of L_z
cos_theta = L_z/L;
theta = acosd(L_z/L);   // Orientations of L_z, degrees
for i = 1:1:7
    if theta(i) > 90 then
        theta(i) = theta(i)-180;
    end
end
printf("\nThe magnitude of total angular momentum = 2*sqrt(%d)*h_cross\n", L^2/4);
printf("\nThe allowed values of L_z in units of h_cross are :");
disp(L_z);
printf("\nThe orientations of L_z in degrees are:");
disp(theta);

// Result
// The magnitude of total angular momentum = 2*sqrt(2)*h_cross

// The allowed values of L_z in units of h_cross are :
//   -3.   -2.   -1.     0.     1.     2.     3.

// The orientations of L_z in degrees are:
//   -30.   -54.73561   -73.221345     90.    73.221345     54.73561     30.
```

Scilab code Exa 8.7 Energy of Hydrogen atom at first excited state

```
// Scilab code Ex8.7: Pg 281 (2005)
clc; clear;
k = 9e+09;  // Coulomb constant, N/Sq.m/C
e = 1.6e-019;   // Electronic charge, C
a_0 = 0.529e-010;   // Bohr's radius, m
n = 2;  // Principal quantum number
l = [0, 1]; // Orbital quantum number
m_l = [-1, 0, 1];     // Orbital magnetic quantum
    number
Z = 1;  // Atomic number of hydrogen
E2 = -k*e^2/(2*a_0)*Z^2/n^2;    // Energy of first
    excited level of hydrogen,
printf("\nThe energy of first excited level of
    hydrogen = %3.1f eV", E2/e);

// Result
// The energy of first excited level of hydrogen =
    -3.4 eV
```

Scilab code Exa 8.8 Probabilities for the Electron in Hydrogen

```
// Scilab code Ex8.8: Pg 284 (2005)
clc; clear;
P = 1/2*integrate('z^2*exp(-z)', 'z', 2, 100); //
    Take some large value of upper limit
printf("\nP(electron in the ground state of hydrogen
     will be found outside the first Bohr radius) =
    %4.1f percent", P*100);

```

```
// Result
// P(electron in the ground state of hydrogen will
   be found outside the first Bohr radius) = 67.7
   percent
```

Chapter 9

Atomic Structure

Scilab code Exa 9.1 Magnetic energy of electron in Hydrogen

```
// Scilab code Ex9.1: Pg 300 (2005)
clc; clear;
// Since mu_B = (e*h_cross)/(2*m_e)
mu_B = 9.27e-24;      // Bohr magneton, J/T
B = 1.00;      // Magnetic flux, T
// Since 1 eV = 1.6e-19 J
eV = 1.6e-19;    // Energy, J
h_cross = 6.58e-16;    // Reduced Plank's constant,
    eV-s
omega_L = (mu_B*B)/(eV*h_cross);      // Larmor
    frequency, rad/s
printf("\nLarmour frequency at n = 2 is %4.2fe+10
    rad/s", omega_L*1e-10);

// Result
// Larmour frequency at n = 2 is 8.81e+10 rad/s
```

Scilab code Exa 9.2 Angles between z axis and the spin angular momentum vector

```
// Scilab code Ex9.2: Pg 307 (2005)
clc; clear;
h_cross = 6.58e-16;     // Reduced Plank's constant, eV-s
S = h_cross*sqrt(3)/2;    // Spin angular momentum, eV-s
S_z = h_cross/2;       // Z-component of spin angular momentum, eV-s
theta_up = acosd(S_z/S);
theta_down = acosd(-S_z/S);
printf("\nFor up spin state, theta = %4.2f degrees", theta_up);
printf("\nFor down spin state, theta = %5.1f degrees", theta_down);

// Result
// For up spin state, theta = 54.74 degrees
// For down spin state, theta = 125.3 degrees
```

Scilab code Exa 9.3 Zeeman Spectrum of Hydrogen Including Spin

```
// Scilab code Ex9.3: Pg 311 (2005)
clc; clear;
e = 1.6e-019;    // Energy equivalent of 1 eV, J/eV
B = 1.00;    // Magnitude of magnetic field, tesla
n = 2;   // Initial state of the hydrogen atom
mu_B = 9.27e-024;    // Bohr's magneton, J/T
E_Z = mu_B*B/e;      // Zeeman energy, eV
E2 = -13.6/n^2;      // Energy of first excited state, eV
m_l = [-2, -1, 0, 1, 2];  // Orbital magnetic quantum number for l = 2
printf("\nThe energies of the electron (in eV) in n = 2 state are:\n");
for i = 1:1:5
```

```
    if m_l(i) < 0 then
        sig = '-';
    else
        sig = '+';
    end
    printf(" (%4.2f %s %4.2e) ", E2, sig, abs(E_Z*
        m_l(i)));
end

// Result
// The energies of the electron (in eV) in n = 2
    state are:
// (-3.40 - 1.16e-04) (-3.40 - 5.79e-05) (-3.40 +
    0.00e+00) (-3.40 + 5.79e-05) (-3.40 + 1.16e-04
```

Scilab code Exa 9.4 Spin orbit energy of Sodium doublet

```
// Scilab code Ex9.4: Pg 311 (2005)
clc; clear;
hc = 1240;    // Product of plank's constant &
    velocity of light, eV
lamda_1 = 588.995;    // Wavelength of first doublet
     of Na lines, nm
lamda_2 = 589.592;    // Wavelength of second
    doublet of Na lines, nm
delta_E = hc*(lamda_2 - lamda_1)/(lamda_1*lamda_2);
        // Spin orbit energy, eV
printf("\nSpin orbit energy from doublet spacing =
    %4.2fe-03 eV", delta_E*1e+03);

// Result
// Spin orbit energy from doublet spacing = 2.13e-03
     eV
```

Scilab code Exa 9.5 Ground state of Helium atom

```
// Scilab code Ex9.5: Pg 316 (2005)
clc; clear;
n = 1;      // Principal quantum number
Z = 2;      // Atomic number of Helium
E_a = (-13.6*Z^2)/n^2;         // Energy of the
    electron in state 'a', eV
E_b = (-13.6*Z^2)/n^2;         // Energy of the
    electron in state 'b', eV
E = E_a + E_b;      // Total electronic energy of
    Helium, eV
printf("\nTotal electronic energy of Helium = %5.1f
    eV", E);

// Result
// Total electronic energy of Helium = -108.8 eV
```

Scilab code Exa 9.6 Effective atomic number for 3s electron in Na

```
// Scilab code Ex9.6: Pg 317 (2005)
clc; clear;
E_i = 5.14;     // Ionisation energy of Na, eV
n = 3;     // Principal quantum number
Z_eff = sqrt((n^2*E_i)/13.6);     // Effective atmic
    number
printf("\nEffective atomic number for 3s electron in
     Na = %4.2f", Z_eff);

// Result
// Effective atomic number for 3s electron in Na =
    1.84
```

Chapter 10

Statistical Physics

Scilab code Exa 10.1 Population of excited states with respect to ground states in Hydrogen

```
// Scilab code Ex10.1: Pg 340 (2005)
clc; clear;
// Part (a)
E1 = -13.6;     // Energy of ground state, eV
E2 = -3.40;     // Energy of first excited state, eV
E3 = -1.51;    // Energy of second excited state, eV
g1 = 2;    // Degeneracy for ground state
g2 = 8;    // Degeneracy for first excited state
g3 = 18;    // Degeneracy for second excited state
kB = 8.617e-05;    // Boltzmann constant, eV/K
Ta = 300;    // Temperature, K
// As n_2/n_1 = (g_2*A*e^(-E_2/(k_B*T)))/(g_1*A*e^(-E_1/(k_B*T))), on simplifying we get
N21 = (g2/g1)*exp((E1 - E2)/(kB*Ta));    // The population of first excited state w.r.t ground state
printf("\nThe population of first excited state w.r.t. ground state at %3d K = %1d", Ta, N21);

// Part (b)
```

```
Tb = 20000;   // Temperature, K
n21 = (g2/g1)*exp((E1 - E2)/(kB*Tb));    // The
    population of first excited state w.r.t ground
    state
n31 = (g3/g1)*exp((E1 - E3)/(kB*Tb));     // The
    population of second excited state w.r.t ground
    state
printf("\nThe population of first excited state w.r.
    t. ground state at %4d K = %6.4f", Tb, n21);
printf("\nThe population of second excited state w.r
    .t ground state at %4d K = %6.4f", Tb, n31);

// Part (c)
E_strength = (g3/g2)*exp((E2 - E3)/(kB*Tb));    //
    Emission strength
printf("\nEmission strength of spectral lines = %3.2
    f", E_strength);

// Result
// The population of first excited state w.r.t.
    ground state at 300 K = 0
// The population of first excited state w.r.t.
    ground state at 20000 K = 0.0108
// The population of second excited state w.r.t
    ground state at 20000 K = 0.0081
// Emission strength of spectral lines = 0.75
```

Scilab code Exa 10.2 Validity of Maxwell Boltzmann Statistics

```
// Scilab code Ex10.2: Pg 345 (2005)
clc; clear;
// Part (a)
N = 6.02e+23;     // Number of molecules at STP
m = 3.34e-27;    // Mass of H-molecule, kg
h_cross = 1.055e-34;    // Reduced Plank's constant,
```

```
J-s
V = 22.4e-03;       // Volume occupied by molecules at STP, m^3
T = 273;    // Absolute temperature, K
k_B = 13.8e-24;     // Boltzmann constant, J/K
x_H = N/V*h_cross^3/(8*(m*k_B*T)^(3/2));     // Particle concentration at STP
printf("\nx_H = %4.2e", x_H);
if (x_H < 1)
printf("\nThe criterion for the validity of Maxwell Boltzmann Statistics is satisfied in hydrogen.");

// Part (b)
d_Ag = 10.5;    // Density of silver, g/m^3
M_Ag = 107.9;    // Molar weight of silver, g
NV_Ag = (d_Ag/M_Ag)*(6.02e+023)*1e+06;     // Density of free electrons in silver, electrons/m^3
me = 9.109e-031;     // Mass of an electron, kg
T = 300;    // Room temperature, K
x_Ag = ((NV_Ag)*h_cross^3)/(8*(me*k_B*T)^(3/2));    // Particle concentration at STP
printf("\nx_Ag = %4.2f", x_Ag);
if (x_Ag > 1)
printf("\nThe criterion for the validity of Maxwell Boltzmann Statistics does not hold for electrons in silver");

// Result
// x_H = 8.84e-08
// The criterion for the validity of Maxwell Boltzmann Statistics is satisfied in hydrogen.
// x_Ag = 37.13
// The criterion for the validity of Maxwell Boltzmann Statistics does not hold for electrons in silver
```

Scilab code Exa 10.3 Photons in a box

```
// Scilab code Ex10.3: Pg 352 (2005)
clc; clear;
// Part (b)
I = integrate('z^2/(exp(z)-1)', 'z', 0, 100); //
    Integral value
k_B = 8.62e-05;    // Boltzmann constant, eV/K
T = 3000;     // Temperature, K
h = 4.136e-15;     // Plank's constant, eV
c = 3e+10;     // Velocity of light, cm/s
N_V = 8*%pi*((k_B*T)/(h*c))^3*I;     // Number of
    photons/cc
printf("\nThe density of photons inside the cavity =
    %4.2fe+11 photons/cc", N_V*1e-11);

// Result
// The density of photons inside the cavity = 5.47e
    +11 photons/cc
```

Scilab code Exa 10.4 Specific Heat of Diamond

```
// Scilab code Ex10.4: Pg 356 (2005)
clc; clear;

// Part (a)
k_B = 8.62e-05;    // Boltzmann constant, eV/K
T_E = 1300;     // Temperature, K
h_cross = 6.58e-16;    // Reduced plank's constant,
    eV-s
omega = (k_B*T_E)/h_cross;     // Frequency of
    vibration of carbon atom in diamond, Hz
```

```
spacing = (h_cross*omega);    // Spacing between adjacent oscillator energy level, eV
printf("\nFrequency of vibration of carbon atom in diamond = %4.2e Hz", omega);
printf("\nSpacing between adjacent oscillator energy level = %5.3f eV", spacing);

// Part (b)
T_R = 300;     // Room temperature, K
p = exp((h_cross*omega)/(k_B*T_R));  // For simplication
E_R = (h_cross*omega)/(p-1);   // Average energy of oscillator at room temperature, eV
T = 1500;    // Temperature, K
q = exp((h_cross*omega)/(k_B*T));     // For simplication
E_bar = (h_cross*omega)/(q-1);     // Average energy at 1500 K, eV
printf("\nAverage energy of oscillator at room temperature = %7.5f eV", E_R);
printf("\nAverage oscillator energy at %4d K = %7.5f eV", T, E_bar);


// Result
// Frequency of vibration of carbon atom in diamond = 1.70e+14 Hz
// Spacing between adjacent oscillator energy level = 0.112 eV
// Average energy of oscillator at room temperature = 0.00149 eV
// Average oscillator energy at 1500 K = 0.0813 eV
```

Scilab code Exa 10.5 Fermi Energy of Gold

```
// Scilab code Ex10.5: Pg 360 (2005)
clc; clear;

// Part (a)
h = 6.625e-34;    // Plank's constant, J-s
m_e = 9.11e-31;    // Mass of electron, kg
density = 19.32/(1e-02)^3;    // Density of gold, g/m^3
weight = 197;      // Molar weight, g/mol
N_V = (density/weight)*6.02e+23;    // Number of electrons per mole
E_F = (h^2/(2*m_e*1.6e-19))*((3*(N_V))/(8*%pi))^(2/3);    // Fermi energy of Gold at 0 K
printf("\nFermi energy of Gold at 0 K = %4.2f eV", E_F);

// Part (b)
v_F = sqrt((2*E_F*1.6e-19)/m_e);    // Fermi speed of Gold at 0 K
printf("\nFermi speed of Gold at 0 K = %4.2fe+06 m/s", v_F*1e-06);

// Part (c)
k_B = 8.62e-05;    // Boltzmann constant, eV/K
T_F = (E_F)/(k_B);     // Fermi temperature for Gold at 0 K, K
printf("\nFermi temperature for Gold at 0 K = %5d K", T_F);

// Result
// Fermi energy of Gold at 0 K = 5.53 eV
// Fermi speed of Gold at 0 K = 1.39fe+06 m/s
// Fermi temperature for Gold at 0 K = 64201 K
```

Chapter 11

Molecular Structure

Scilab code Exa 11.1 Rotation of CO molecule

```
// Scilab code Ex11.1: Pg 380 (2005)
clc; clear;
// Part (a)
f = 1.15e+11;   // Frequency of transitions, Hz
omega = 2*(%pi)*f;    // Angular frequency of
   absorbed radiations, Hz
h_cross = 1.055e-34;   // Reduced planks constant, J
   -s
// Since E = (h_cross)^2/I_CM = h_cross*omega,
   solving for I_CM
I_CM = h_cross/omega;   // Moment of inertia of
   molecule about its center of mass, kg-m^2
printf("\nThe moment of inertia of molecule about
   its center of mass = %4.2e kg-m^2", I_CM);

// Part (b)
m_O = 16;   // Mass of oxygen atom, a.m.u
m_C = 12;    // Mass of carbon atom, a.m.u
mu = ( m_O * m_C *0.166e-26)/(m_O + m_C);       //
   Reduced mass, kg
// Since I_CM = mew*R_o^2, solving for R_o
```

```
R_0 = sqrt(I_CM/mu);     // Bond length of carbon
    monoxide molecule, m
printf("\nThe bond length of carbon monoxide
    molecule = %5.3f nm", R_0/1e-09);

// Result
// The moment of inertia of molecule about its
    center of mass = 1.46e-046 kg-m^2
// The bond length of carbon monoxide molecule =
    0.113 nm
```

Scilab code Exa 11.2 Variation of CO molecule

```
// Scilab code Ex11.2: Pg 383 (2005)
clc; clear;

// Part (a)
f = 6.42e+13;    // Frequency of absorption, Hz
omega = 2*(%pi)*f;    // Angular frequency of
    absorbed radiations, Hz
mu = 1.14e-26;     // Reduced mass of CO molecule, kg
K = mu*(omega^2);     // Effective force constant of
    CO molecule, N/m
printf("\nThe effective force constant of CO
    molecule = %4.2e N/m", K);

// Part (b)
h_cross = 1.055e-34;    // Reduced Planck's constant,
     J-s
A = sqrt(h_cross/(mu*omega));    // Amplitude of
    vibrations, m
printf("\nThe amplitude of vibrations = %7.5f nm", A
    /1e-09);

// Result
```

```
// The effective force constant of CO molecule =
    1.85e+003 N/m
// The amplitude of vibrations = 0.00479 nm
```

Chapter 12

The Solid State

Scilab code Exa 12.1 Classical free electron model

```
// Scilab code Ex12.1: Pg 418 (2005)
clc; clear;
// Part (a)
k_B = 1.38e-23;    // Boltzmann constat, J/K
m_e = 9.11e-31;     // Mass of electron, kg
T = 300;    // Temperature, K
N_A = 6.023e+023;     // Avogadro's number
v_rms = sqrt((3*k_B*T)/m_e);        // Root mean
   square velocity of electrons, m/s
I = 10;    // Electric current, A
A = 4e-06;     // Area of cross-section of copper
   wire, m^2
J = I/A;    // Current density, A-m^(-2)
d = 8.96;     // Density of copper at room
   temperature, g/cc
M = 63.5;     // Atomic mass of Cu, g
n = d*N_A/M*1e+06;     // Number of electrons per
   metre cube
e = 1.6e-19;     // Charge on electron, C
v_d = J/(n*e);     // Drift velocity, m/s
v_d_rms  = v_d/v_rms;     // Ratio of drift speed to
```

```
    rms speed
printf("\nThe ratio of drift speed to rms speed is =
    %3.1e", v_d_rms);

// Part (b)
L = 2.6e-10;
tau = L/v_rms;    // Average time between two
    collisions , s
printf("\nAverage time between two collisions = %2.2
    e s", tau);

// Part (c)
sigma = (n*e^2*L)/sqrt(3*k_B*T*m_e);   //
    Conductivity of copper , per ohm-m
printf("\nConductivity of copper at room temperature
    = %3.1e per ohm-m", sigma);


// Result
// The ratio of drift speed to rms speed is = 1.6e
    -009
// Average time between two collisions = 2.23e-015 s
// Conductivity of copper at room temperature = 5.3e
    +006 per ohm-m
```

Scilab code Exa 12.2 Conduction in diamond

```
// Scilab code Ex12.2: Pg 429 (2005)
clc; clear;
V = 7;     // Energy gap , V
L = 5e-08;    // Mean free path  , m
E = V/L;    // Electric field , V/m
printf("\nThe electric field strength required to
    produce conduction in diamond = %3.1fe+08 V/m", E
    *1e-08);
```

```
printf("\n");

// Result
// The electric field strength required to produce conduction in diamond = 1.4e+08 V/m
```

Scilab code Exa 12.3 Forward and reverse currents in diode

```
// Scilab code Ex12.3: Pg 436 (2005)
clc; clear;
e_V = 1;     // Energy applied to diode, eV
k_B_T = 0.025;    // Product of Boltzmann constant and temperature, eV
// For simplicity let (q*V)/(k_B*T) = x
x = (e_V/(k_B_T));
I_f_r = (exp(x)-1)/(exp(-x)-1);    // Ratio of forward current to reverse current in diode
printf("\nThe ratio of forward current to reverse current in diode = %3.1fe+17", I_f_r*1e-17);

// Result
// The ratio of forward current to reverse current in diode = -2.4e+17
```

Chapter 13

Nuclear Structure

Scilab code Exa 13.1 The Atomic Mass Unit

```
// Scilab code Ex13.1: Pg 466 (2005)
clc; clear;
M = 0.012;  // Atomic mass of carbon, kg
N_A = 6.02e+023; // Avogadro's number
m = M/N_A;   // Mass of one Carbon-12 atom, kg
// As m = 12*u, twelve mass units, solving for u
u = m/12;   // The atomic mass unit, kg
printf("\nThe atomic mass unit = %4.2e kg", u);

// Result
// The atomic mass unit = 1.66e-27 kg
```

Scilab code Exa 13.2 The Volume and Density of Nucleus

```
// Scilab code Ex13.2: Pg 468 (2005)
clc; clear;
r0 = 1.2e-015;  // Nuclear mean radius, m
m = 1.67e-027;  // Mass of the nucleon, kg
```

```
rho_0 = 3*m/(4*%pi*r0^3);    // Density of the
    nucleus, kg per metre cube
printf("\nThe mass of the nucleus = Am approx.");
printf("\nThe volume of the nucleus = 4/3*pi*r0^3*A"
    );
printf("\nThe density of the nucleus = %3.1e kg per
    metre cube", rho_0);

// Result
// The mass of the nucleus = Am approx.
// The volume of the nucleus = 4/3*pi*r0^3*A
// The density of the nucleus = 2.3e+17 kg per metre
    cube
```

Scilab code Exa 13.3 Binding energy of the Deuteron

```
// Scilab code Ex13.3: Pg 473 (2005)
clc; clear;
M2 = 2.014102;   // Atomic mass of deuteron, u
M_H = 1.007825;  // Atomic mass of hydrogen, u
m_n = 1.008665;  // Mass of a neutron, u
E_b = (M_H + m_n - M2)*931.494;      // Binding
    energy of the deuteron, MeV/u
printf("\nThe binding energy of the Deuteron = %5.3f
    MeV", E_b);

// Result
// The binding energy of the Deuteron = 2.224 MeV
```

Scilab code Exa 13.4 Left out sample during radioactive decay

```
// Scilab code Ex13.4: Pg 482 (2005)
clc; clear;
```

```
T = 5730;    // Half life of the carbon-14 isotope, years
N0 = 1000;   // Initial number of carbon-14 isotope
t = 22920;   // Time of decay, years
n = t/T;     // Total number of half lives
N = (1/2)^n*N0; // Sample remains after 22920 years
printf("\nNumber of C-14 isotopes remained after %d years = %d", t, N);

// Result
// Number of C-14 isotopes remained after 22920 years = 62
```

Scilab code Exa 13.5 The Activity of Radium

```
// Scilab code Ex13.5: Pg 483 (2005)
clc; clear;
T_half = 1.6e+03*3.16e+07;     // Half life of radioactive nucleus Ra-226, s
lambda = 0.693/T_half;   // Decay constant of Ra-226, per second
N0 = 3.0e+016;   // Number of radioactive nuclei at t = 0
R0 = lambda*N0; // Activity of sample at t = 0, decays/s
t = 2.0e+003*3.16e+07;     // Time during which the radioactive disintegration takes place, s
R = R0*exp(-1*lambda*t);   // Decay rate after 2.0e+003 years, decay/s
printf("\nThe decay constant of Ra-226 = %3.1e per second", lambda);
printf("\nThe activity of sample at t = 0 = %4.1f micro-Ci", R0/(3.7e+010*1e-006));
printf("\nThe activity of sample after %3.1e years = %3.1e decays/s", t, R);
```

```

// Result
// The decay constant of Ra-226 = 1.4e-11 per second
// The activity of sample at t = 0 = 11.1 micro-Ci
// The activity of sample after 6.3e+10 years = 1.7e+05 decays/s
```

Scilab code Exa 13.6 The Activity of Carbon

```
// Scilab code Ex13.6: Pg 483 (2005)
clc; clear;
M = 11.0;    // Atomic mass of C-11 isotope, g
NA = 6.02e+023; // Avogadro's number
m = 3.50e-06;    // Given mass of Cabon-11, g

// Part (a)
N = m/M*NA;       // Number of C-11 atoms in 3.50 micro-g of sample
printf("\nThe number of C-11 atoms in %4.2f micro-g of sample = %4.2e nuclei", m/1e-06, N);

// Part (b)
T_half = 20.4*60;     // Half life of radioactive nucleus C-11, s
lambda = 0.693/T_half;   // Decay constant of C-11, per second
R0 = lambda*N; // Activity of sample at t = 0, decays/s
t = 8.00*60*60;    // Time during which the radioactive disintegration takes place, s
R = R0*exp(-1*lambda*t);   // Decay rate after 2.0e+003 years, decay/s

printf("\nThe activity of C-11 sample at t = 0 is %4.2e decays/s", R0);
```

```
printf("\nThe activity of sample after %4.2f hours =
    %4.2e decays/s", t/3600, R);

// Result
// The number of C-11 atoms in 3.50 micro-g of
   sample = 1.92e+17 nuclei
// The activity of C-11 sample at t = 0 is 1.08e+14
   decays/s
// The activity of sample after 8.00 hours = 8.99e
   +06 decays/s
```

Scilab code Exa 13.7 The Radiactive Isotope of Iodine

```
// Scilab code Ex13.7: Pg 484 (2005)
clc; clear;
R0 = 5; // Activity of I-131 isotope at the time of
   shipment, mCi
R = 4.2;    // Activity of I-131 isotope at the time
    of receipt by the medical laboratory, mCi
T_half = 8.04;    // Half life of radioactive
   nucleus I-131, days
lambda = 0.693/T_half;  // Decay constant of C-11,
   per second
// As log(R/R0) = -lambda*t, solving for t
t = -1/lambda*log(R/R0);    // Time that has elapsed
    between two measurements, days
printf("\nThe time that has elapsed between two
   measurements = %4.2f days", t);

// Result
// The time that has elapsed between two
   measurements = 2.02 days
```

Scilab code Exa 13.8 Energy Liberated during Decay of Radium

```
// Scilab code Ex13.8: Pg 486 (2005)
clc; clear;
M_X = 226.025406;    // Atomic mass of Ra-226, u
M_Y = 222.017574;    // Atomic mass of Rn-222, u
M_alpha = 4.002603; // Mass of alpha particle, u
Q = (M_X - M_Y - M_alpha)*931.494;   // Q-value for
   Radium Decay, MeV/u
printf("\nThe Q-value for Radium Decay = %4.2f MeV",
    Q);

// Result
// The Q-value for Radium Decay = 4.87 MeV
```

Scilab code Exa 13.9 Probability of Alpha Decay

```
// Scilab code Ex13.9: Pg 487 (2005)
clc; clear;
Z = 86;       // Atomic number of radon
A = 222;      // Mass number of radon
k = 9e+09;    // Coulomb constant, N-metre square per
   C-square
e = 1.6e-019;    // Charge on an electron, C
r0 = 7.25e-015; // Bohr radius for alpha particle, m
E0 = k*e^2/(2*r0*1e+06*e);  // Rydberg energy, MeV
R = 1.2e-015*A^(1/3);    // Radius of radon nucleus,
    fm
E = 5;   // Disintegration energy during alpha decay,
    MeV
T_E = exp(-4*%pi*Z*sqrt(E0/E)+8*sqrt(Z*R/r0));  //
   Decay probability for alpha disintegration
printf("\nThe decay probability for alpha
   disintegration at %d MeV energy = %4.2e", E, T_E)
   ;
```

```

// Result
// The decay probability for alpha disintegration at 5 MeV energy = 1.29e-34
```

Scilab code Exa 13.11 Radioactive Dating

```
// Scilab code Ex13.11: Pg 490 (2005)
clc; clear;
T_half = 5370*3.6e+07;  // Half life of C-14, s
lambda = 0.693/T_half;  // // Decay constant for C-14 disintegration , per sec
N_C12 = 6.02e+023/12*25;    // Number of C-12 nuclei in 25.0 g of carbon
N0_C14 = 1.3e-012*N_C12;    // Number of C-14 nuclei in 25.0 g of carbon before decay
R0 = N0_C14*3.83e-012*60;  // Initial activty of the sample , decays/min
R = 250;    // Present activity of the sample
// As R = R0*exp(-lambda*t), solving for t
t = -1/lambda*log(R/R0);    // Time during which the tree dies , s
printf("\nThe lifetime of the tree = %3.1e yr", t/(365*24*60*60));

// Result
// The lifetime of the tree = 3.6e+03 yr
```

Chapter 14

Nuclear Physics Applications

Scilab code Exa 14.1 Energy released in Fission

```
// Scilab code Ex14.1: Pg 513 (2005)
clc; clear;
// Part (a)
u = 931.5;      // Atomic mass unit, Mev
M_Li = 7.016003;    // Mass of Lithium, kg
M_H = 1.007825;    // Mass of Hydrogen, kg
M_He = 4.002603;    // Mass of Helium, kg
Q = (M_Li + M_H - 2*M_He)*u;    // Q-value of the
    reaction, MeV
// Part (b)
K_incident = 0.6;     // Kinetic energy of the
    incident protons, MeV
K_products = Q + K_incident;    // Kinetic energy of
     the products
printf("\nThe Q value of the reaction = %4.1f MeV",
    Q);
printf("\nThe kinetic energy of the products (two
    alpha particles) = %4.1f MeV", K_products);

// Result
// The Q value of the reaction = 17.3 MeV
```

```
// The kinetic energy of the products (two alpha
   particles) = 17.9 MeV
```

Scilab code Exa 14.2 Neutron capture by Al

```
// Scilab code Ex14.2: Pg 509 (2005)
clc; clear;
roh = 2.7e+06;   // Density of Al, g/cm^3
A = 27;   // Mass number of Al
n = (6.02e+23*roh)/A;    // Number of nuclei/m^3
sigma = 2.0e-31;    // Effective area of nucleas
   normal to motion, m^2
R_0 = 5.0e+12;       // Rate of incident particles
   per unit area, neutrons/cm^2-s
x = 0.30e-03;        // Thickness of foil, m
R = (R_0*sigma*n*x) // Number of neutrons captured
   by foil, neutrons/cm^2-s
printf("\nThe number of neutrons captured by foil =
   %3.1fe+07 neutrons/Sq.cm-s", R*1e-07);

// Result
// The number of neutrons captured by foil = 1.8e+07
   neutrons/Sq.cm-s
```

Scilab code Exa 14.4 Energy released in the Fission of U235

```
// Scilab code Ex14.4: Pg 513 (2005)
clc; clear;
m = 1;   // Mass of Uranium taken, kg
Q = 208;   // Disintegration energy per event, MeV
A = 235;   // Mass number of Uranium
N = (6.02e+23*m)/A;    // Number of nuclei
E = N*Q;    // Disintegration energy, MeV
```

```
printf("\nThe total energy released if %1d kg of Uranium undergoes fission = %4.2fe+26 MeV", m, E*1e-23);

// Result
// The total energy released if 1 kg of Uranium undergoes fission = 5.33e+26 MeV
```

Scilab code Exa 14.5 A Rough Mechanism for Fission Process

```
// Scilab code Ex14.5: Pg 513 (2005)
clc; clear;
A_Ba = 141;     // Mass number of Barium
A_Kr = 92;    // Mass number of Barium
r_0 = 1.2e-15;    // Separation constant, m
r_Ba = r_0*A_Ba^(1/3);   // Nuclear radius of Barium, m
r_Kr = r_0*A_Kr^(1/3);   // Nuclear radius of Krypton, m
r = r_Ba + r_Kr;   // Separation between two atoms, m
Z_1 = 56;     // Atomic number of Barium
Z_2 = 36;   // Atomic number of Barium
k = 1.440e-09;   // Coulomb constant, eV-m
U = k*Z_1*Z_2/r                    // Coulomb Potential energy of two charges, MeV
printf("\nThe Coulomb potential energy for two charges = %3d MeV" , U/1e+06);
printf("\nThis shows that the fission mechanism is plausible");

// Result
// The Coulomb potential energy for two charges = 248 MeV
// This shows that the fission mechanism is
```

```
plausible
```

Scilab code Exa 14.6 The Fusion of Two Deutrons

```
// Scilab code Ex14.6: Pg 519 (2005)
clc; clear;
// Part (a)
e = 1.6e-19;    // Charge on electron, C
k = 8.99e-09;    // Coulomb constant, N-m^2/C^2
r = 1.0e-14;    // Distance between two duetrons, m
// We have U = (k*q1*q2)/r, for duetrons q1 = q2 = e
    , therefore we get
U = (k*e^2)/r;    // Potential energy of duetrons, J
E_C = 1.1e-014;    // The coulomb energy per deutron
    , J
k_B = 1.38e-023;    // Boltzmann constant, J/mol/K
T = 2/3*E_C/k_B;    // Effective temperature
    required for deutron to overcome the potential
    barrier, K
printf("\nThe potential energy of two duetrons
    separated by the distance of %1.0de-14 m = %4.2f
    MeV", r*1e+14, (U*1e+12)/e);
printf("\nThe effective temperature required for
    deutron to overcome the potential barrier = %3.1e
    K", T);

// Result
// The potential energy of two duetrons separated by
    the distance of 1e-14 m = 0.14 MeV
// The effective temperature required for deutron to
    overcome the potential barrier = 5.3e+008 K
// Result
// The potential energy of two duetrons separated by
    the distance of 1e-14 m = 0.14 MeV
```

Scilab code Exa 14.7 Half value thickness

```
// Scilab code Ex14.7: Pg 530 (2005)
clc; clear;
mew = 55e-02;     // Linear absoption coefficient ,
    per m
// In equation I(x) = I_o*exp(-mew*x), replacing I(x
    ) by I_o/2 & solving for x, we get
x = log(2)/mew;    // Half value thickness , m
printf("\nThe half value thickness for lead = %4.2fe
    -02 cm", x);

// Result
// The half value thickness for lead = %1.26e-02 cm
```

Chapter 15

Elementary Particle

Scilab code Exa 15.2 Checking Baryon Numbers

```
// Scilab code Ex15.2: Pg 560 (2005)
clc; clear;
// Data for Reaction 1
R1 = cell(6,2);  // Declare a 6X2 cell
R1(1,1).entries = 'p';
R1(2,1).entries = 'n';
R1(3,1).entries = 'p';
R1(4,1).entries = 'p';
R1(5,1).entries = 'n';
R1(6,1).entries = 'p_bar';
R1(1,2).entries = 1;
R1(2,2).entries = 1;
R1(3,2).entries = 1;
R1(4,2).entries = 1;
R1(5,2).entries = 1;
R1(6,2).entries = -1;
// Data for reaction 2
R2 = cell(5,2);  // Declare a 5X2 cell
R2(1,1).entries = 'p';
R2(2,1).entries = 'n';
R2(3,1).entries = 'p';
```

```
R2(4,1).entries = 'p';
R2(5,1).entries = 'p_bar';
R2(1,2).entries = 1;
R2(2,2).entries = 1;
R2(3,2).entries = 1;
R2(4,2).entries = 1;
R2(5,2).entries = -1;
// Check baryon number conservation for first reaction
if (R1(1,2).entries+R1(2,2).entries) == (R1(3,2).entries+R1(4,2).entries+R1(5,2).entries+R1(6,2).entries) then
    printf("\nThe reaction %s + %s --> %s + %s + %s + %s can occur (B is conserved)", R1(1,1).entries, R1(2,1).entries, R1(3,1).entries, R1(4,1).entries, R1(5,1).entries, R1(6,1).entries);
else
    printf("\nThe reaction %s + %s --> %s + %s + %s + %s cannot occur (B is not conserved)", R1(1,1).entries, R1(2,1).entries, R1(3,1).entries, R1(4,1).entries, R1(5,1).entries, R1(6,1).entries);
end
// Check baryon number conservation for second reaction
if R2(1,2).entries+R2(2,2).entries == R2(3,2).entries+R2(4,2).entries+R2(5,2).entries then
    printf("\nThe reaction %s + %s --> %s + %s + %s can occur (B is conserved)", R2(1,1).entries, R2(2,1).entries, R2(3,1).entries, R2(4,1).entries, R2(5,1).entries);
else
    printf("\nThe reaction %s + %s --> %s + %s + %s cannot occur (B is not conserved)", R2(1,1).entries, R2(2,1).entries, R2(3,1).entries, R2(4,1).entries, R2(5,1).entries);
end
```

```
// Result
// The reaction p + n --> p + p + n + p_bar can occur (B is conserved)
// The reaction p + n --> p + p + p_bar cannot occur (B is not conserved)
```

Scilab code Exa 15.3 Checking Lepton Numbers

```
// Scilab code Ex15.3: Pg 561 (2005)
clc; clear;
// Data for Reaction 1
R1 = cell(4,3);  // Declare a 4X3 cell
R1(1,1).entries = 'mu';
R1(2,1).entries = 'e-';
R1(3,1).entries = 'nue_bar';
R1(4,1).entries = 'nu_mu';
R1(1,2).entries = 1;    // Muon number for mu
R1(2,2).entries = 0;    // Muon number for e-
R1(3,2).entries = 0;    // Muon number for nue_bar
R1(4,2).entries = 1;    // Muon number for nu_mu
R1(1,3).entries = 0;    // Lepton number for mu
R1(2,3).entries = 1;    // Lepton number for e-
R1(3,3).entries = -1;    // Lepton number for nue_bar
R1(4,3).entries = 0;    // Lepton number for nu_mu
// Data for Reaction 2
R2 = cell(4,3);  // Declare a 4X3 cell
R2(1,1).entries = 'Pi+';
R2(2,1).entries = 'mu+';
R2(3,1).entries = 'nu_mu';
R2(4,1).entries = 'nu_e';
R2(1,2).entries = 0;    // Muon number for Pi+
R2(2,2).entries = -1;    // Muon number for mu+
R2(3,2).entries = 1;    // Muon number for nu_mu
```

```
R2(4,2).entries = 0;     // Muon number for nu_e
R2(1,3).entries = 0;     // Lepton number for Pi+
R2(2,3).entries = 0;     // Lepton number for mu+
R2(3,3).entries = 0;     // Lepton number for nu_mu
R2(4,3).entries = 1;     // Lepton number for nu_e
// Check lepton number conservation for first reaction
if (R1(1,2).entries== R1(2,2).entries+R1(3,2).entries+R1(4,2).entries) & (R1(1,3).entries == R1(2,3).entries+R1(3,3).entries+R1(4,3).entries) then
    printf("\nThe reaction %s --> %s + %s + %s can occur (Both L_mu and L_e are conserved)", R1(1,1).entries, R1(2,1).entries, R1(3,1).entries, R1(4,1).entries);
else
    printf("\nThe reaction %s + %s --> %s + %s + %s + %s cannot occur (L_mu and L_e are not conserved)", R1(1,1).entries, R1(2,1).entries, R1(3,1).entries, R1(4,1).entries);
end
// Check lepton number conservation for second reaction
if (R2(1,2).entries== R2(2,2).entries+R2(3,2).entries+R2(4,2).entries) & (R2(1,3).entries == R2(2,3).entries+R2(3,3).entries+R2(4,3).entries) then
    printf("\nThe reaction %s --> %s + %s + %s can occur (Both L_mu and L_e are conserved)", R2(1,1).entries, R2(2,1).entries, R2(3,1).entries, R2(4,1).entries);
else
    printf("\nThe reaction %s --> %s + %s + %s cannot occur (L_mu is conserved but L_e is not conserved)", R2(1,1).entries, R2(2,1).entries, R2(3,1).entries, R2(4,1).entries);
end
```

```
// Result
// The reaction mu --> e- + nue_bar + nu_mu can occur (Both L_mu and L_e are conserved)
// The reaction Pi+ --> mu+ + nu_mu + nu_e cannot occur (L_mu is conserved but L_e is not conserved0
```

Scilab code Exa 15.4 Conservation of strangeness

```
// Scilab code Ex15.4: Pg 563 (2005)
clc; clear;
// Data for Reaction 1
R1 = cell(4,2);  // Declare a 4X2 cell
R1(1,1).entries = 'Pi0';
R1(2,1).entries = 'n';
R1(3,1).entries = 'K+';
R1(4,1).entries = 'sigma-';
R1(1,2).entries = 0;    // Strangeness number for Pi0
R1(2,2).entries = 0;    // Strangeness number for n
R1(3,2).entries = 1;    // Strangeness number for K+
R1(4,2).entries = -1;    // Strangeness number for sigma-
// Data for Reaction 2
R2 = cell(4,2);  // Declare a 4X2 cell
R2(1,1).entries = 'Pi-';
R2(2,1).entries = 'p';
R2(3,1).entries = 'Pi-';
R2(4,1).entries = 'sigma+';
R2(1,2).entries = 0;    // Strangeness number for Pi-
R2(2,2).entries = -1;    // Strangeness number for p
R2(3,2).entries = 1;    // Strangeness number for pi-
R2(4,2).entries = 0;    // Strangeness number for
```

```
sigma+
// Check strangeness number conservation for first reaction
if R1(1,2).entries + R1(2,2).entries == R1(3,2).entries+R1(4,2).entries then
    printf("\nThe reaction %s + %s --> %s + %s can occur (Strangness is conserved)", R1(1,1).entries, R1(2,1).entries, R1(3,1).entries, R1(4,1).entries);
else
    printf("\nThe reaction %s + %s --> %s + %s cannot occur (Strangness is not conserved)", R1(1,1).entries, R1(2,1).entries, R1(3,1).entries, R1(4,1).entries);
end
// Check strangeness number conservation for second reaction
if R2(1,2).entries + R2(2,2).entries == R2(3,2).entries+R2(4,2).entries then
    printf("\nThe reaction %s + %s --> %s + %s can occur (Strangness is conserved)", R2(1,1).entries, R2(2,1).entries, R2(3,1).entries, R2(4,1).entries);
else
    printf("\nThe reaction %s + %s --> %s + %s cannot occur (Strangness is not conserved)", R2(1,1).entries, R2(2,1).entries, R2(3,1).entries, R2(4,1).entries);
end

// Result
// The reaction Pi0 + n --> K+ + sigma- can occur (Strangness is conserved)
// The reaction Pi- + p --> Pi- + sigma+ cannot occur (Strangness is not conserved)
```

Scilab code Exa 15.5 Making virtual particle real

```
// Scilab code Ex15.5: Pg 570 (2005)
clc; clear;
m_pi = 135;      // Mass of pion, MeV/c^2
m_p = 938.3;      // Mass op proton, MeV/c^2
// For simplification, let velocity of light be
   unity
c = 1;        // Velocity of light, m/s
// Simplifying  K_th = (m_3 + m_4 + m_5 + ....)^2*c
   ^2 - (m_1 + m_2)^2*c^2, we get
K_th = 2*m_pi*c^(2) + ((m_pi*c)^2/(2*m_p));      //
   Required kinetic energy of proton, MeV
printf("\nRequired kinetic energy of proton = %3d
   MeV", ceil(K_th));

// Result
// Required kinetic energy of proton = 280 MeV
```

Chapter 16

Cosmology

Scilab code Exa 16.1 Hubbles law

```
// Scilab code Ex16.1: Pg 15 (2005)
clc; clear;
c = 3e+05;    // Velocity of light , km/s
v = c/4;     // Recessional velocity , km/s
H_0 = 20e-06;     // Hubble's constant , km/s/lightyear
// From Hubble's law , v = H_o*R_max, solving for R_max
R_max = v/H_0;   // Maximum distance at which Hubble's law applies without relativistic correction , lightyears
printf("\nThe maximum distance at which Hubbles law applies without relativistic correction = %1.0e ly", R_max);
printf("\n");

// Result
// The maximum distance at which Hubbles law applies without relativistic correction = 4e+09 ly
```

Scilab code Exa 16.2 Critical density of universe

```
// Scilab code Ex16.2: Pg 22 (2005)
clc; clear;
H = 23e-03/(9.46e15);     // Hubble's constant, km/s/ly
G = 6.67e-11;       // Gravitational constant, N-m^2/kg^2
//  Since H^2 = (8*%pi*G*p_c)/3, solving for p_c
p_c = (3*H^2)/(8*%pi*G);     // Critical mass density of universe, kg/m^3
printf("\nCritical mass density of universe = %4.2e kg per metre cube", p_c);


// Result
// Critical mass density of universe = %1.06e-27 kg/m^3
```