Scilab Textbook Companion for Introduction To Nuclear And Particle Physics by V. K. Mittal, R. C. Verma And S. C. Gupta[1]

Created by
Arjun Singh
M.Sc.
Physics
Shri Mata Vaishno Devi Univeristy
College Teacher
Mr. Pankaj Biswas
Cross-Checked by
Dr. Jitendra Sharma

August 10, 2013

[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 Nuclear And Particle Physics

Author: V. K. Mittal, R. C. Verma And S. C. Gupta

Publisher: PHI Learning Pvt. Ltd., New Delhi

Edition: 2

Year: 2011

ISBN: 978-81-203-4311-5

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 Nucleus

Scilab code Exa 1.3.1 de Broglie relation

```
// Scilab code Exa1.3.1 Momentum determination for a neutron using de-Broglie relation : Page 31 (2011)
h = 6.626e-034;     // Planck's constant, Js
e = 1.602e-019;     // Charge on an electron, C
red_h = h/(2*%pi*e*1e+06);     // Reduced Planck's constant, MeV
lambda = 5.0e-015;     // de_Broglie wavelength of neutron, m
p = red_h/lambda;     // Momentum of the neutron, MeV-s/m
printf("\nThe momentum of the neutron from de-Broglie relation : %5.3e MeV-s/m", p);

// Result
// The momentum of the neutron from de-Broglie relation : 1.317e-007 MeV-s/m
```

Scilab code Exa 1.3.2 Isotopes Isotones and Isobars

```
// Scilab code Exa1.3.2 : Grouping the nuclides as isotopes, isotones and isobars : Page 32 (2011)
E = cell(3,3);    // Declare a cell array of empty matrices for nuclides information
E(1,1).entries = 'C';    // Assign element 'C' to (1,1) cell
E(2,1).entries = 'N';    // Assign element 'N' to (2,1) cell
E(3,1).entries = 'O';    // Assign element 'o' to (3,1) cell
E(1,2).entries = 6;    // Assign atomic No. 6 to (1,2) cell
E(2,2).entries = 7;    // Assign atomic No. 7 to (2,2) cell
E(3,2).entries = 8;    // Assign atomic No. 8 to (3,2) cell
E(1,3).entries = [12,13,14,16];    // Assign mass numbers for 'C' to (1,3) cell
E(2,3).entries = [14,15,16,17];    // Assign mass numbers for 'N' to (2,3) cell
E(3,3).entries = [14,15,16,17];    // Assign mass numbers for 'O' to (3,3) cell
// Isotopes
printf("\nIsotopes:");
printf("\n==========");
for i = 1:1:3    // Search for the three elements one-by-one
   printf("\n(Z = %d)\n",E(i,2).entries);
   for j= 1:1:4
      printf("\t%s(%d)",E(i,1).entries,E(i,3).entries(j));
   end
end
// Isotones
printf("\n\nIsotones:");
printf("\n==========");
for N = 6:1:9    // Search for the neutron numbers from 6 to 9
```

```
    printf("\n(N = %d)\n",N);
    for i = 1:1:3
        for j= 1:1:4
            if  E(i,3).entries(j)- E(i,2).entries == N then // N = A-Z
                printf("\t%s(%d)",E(i,1).entries,E(i,3).entries(j));
            end
        end
    end
end
// Isobars
printf("\n\nIsobars:");
printf("\n==========");
for A = 14:1:17     // Search for the mass numbers from 14 to 17
    printf("\n(A = %d)\n",A);
    for i = 1:1:3
        for j= 1:1:4
            if  E(i,3).entries(j) == A then
                printf("\t%s(%d)",E(i,1).entries,E(i,3).entries(j));
            end
        end
    end
end
//
// Result
//
// Isotopes:
// ==========
// (Z = 6)
//    C(12)     C(13)     C(14)     C(16)
// (Z = 7)
//    N(14)     N(15)     N(16)     N(17)
// (Z = 8)
//   O(14)     O(15)     O(16)     O(17)
//
```

```
// Isotones:
// =========
// (N = 6)
//  C(12)      O(14)
// (N = 7)
//  C(13)      N(14)      O(15)
// (N = 8)
//  C(14)      N(15)      O(16)
// (N = 9)
//  N(16)      O(17)
//
// Isobars:
// ========
// (A = 14)
//  C(14)       N(14)      O(14)
// (A = 15)
//  N(15)      O(15)
// (A = 16)
//  C(16)       N(16)       O(16)
// (A = 17)
//  N(17)      O(17)
```

Scilab code Exa 1.4.1 Rest mass energy of electron

```
// Scilab code Exa1.4.1: To calculate the energy of
   electron at rest : Page 33 (2011)
m = 9.1e-031; // Mass of the electron , Kg
C = 3e+08; // Velocity of the light ,m/s
E = m*C^2/1.6e-013; // Energy of the electron at
   rest , MeV
printf("\nEnergy of the electron at rest : %5.3f MeV
   ", E)

//   Result
//    Energy of the electron at rest : 0.512 MeV
```

Scilab code Exa 1.4.2 Nuclear radius

```
// Scilab code Exa1.4.2 : Estimation of the Nucleus
   type from its radius : Page 33 (2011)
r = 3.46e-015; // Radius of the nucleus, m
r0 = 1.2e-015; // Distance of closest approach of
   the nucleus, m
A = round((r/r0)^3); // Mass number of the nucleus
if A == 23 then
   element = "Na";
elseif A == 24 then
   element = "Mg";
elseif  A == 27 then
   element = "Al";
elseif  A == 28 then
   element = "Si";
end
printf("The mass number of the nucleus is %d and the
    nucleus is of %s", A, element);

//  Result
// The mass number of the nucleus is 24 and the
   nucleus is of Mg
```

Scilab code Exa 1.4.3 Nuclear density

```
// Scilab code Exa1.4.3 : Estimate the density of
   nuclear matter : Page 34 (2011)
m = 40*(1.66e-027); // Mass of the nucleus, kg
r0 = 1.2e-015; // Distance of the closest approach,
   m
```

```
A = 40; // Atomic mass of the nucleus
r = r0*A^(1/3); //Radius of the nucleus, m
V = 4/3*(%pi*r^3); // Volume of the nucleus, m^3
density = m/V; // Density of the nucleus, kg/m^3
printf("\nRadius of the nucleus: %3.1e m\nVolume of
    the nucleus: %5.3e m^3\nDensity of the nucleus:
    %3.1e kg/m^3",r,V,density);

// Result
// Radius of the nucleus: 4.1e-015 m
// Volume of the nucleus: 2.895e-043 m^3
// Density of the nucleus: 2.3e+017 kg/m^3
```

Scilab code Exa 1.4.4 Density of uranium 235

```
// Scilab code Exa1.4.4 : To determine the density
    of U-235 nucleus : Page 34 (2011)
m = 1.66e-027;     // Mass of a nucleon, kg
A = 235; // Atomic mass of U-235 nucleus
M = A*m;     //Mass of the U-235 nucleus, kg
r0 = 1.2e-015; // Distance of closest approach, m
r = r0*(A)^(1/3); // Radius of the U-235 nucleus
V = 4/3*(%pi*r^3); // Volume of the U-235 nucleus,m
    ^3
d = M/V; // Density of the U-235 nucleus,kg/m^3
printf("\nThe density of U-235 nucleus : %4.2e kg
    per metre cube",d)

// Result
// The density of U-235 nucleus : 2.29e+017 kg per
    metre cube
```

Scilab code Exa 1.4.5 Variation of nuclear density with radius

```
// Scilab code Exa1.4.5 : To calculate densities of O and Pb whose radii are given: Page 35 (2011)
m_O = 2.7e-026; // Mass of O nucleus, kg
r_O = 3e-015; // Radius of O nucleus, m
V_O = 4/3*(%pi*(r0)^3); // Volume of O nucleus, metre cube
d_O = m_O/V_O; // Density of O nucleus, kg/metre cube
m_Pb = 3.4e-025; // Mass of Pb nucleus, kg
r_Pb = 7.0e-015; // Radius of Pb nucleus, m
V_Pb = 4/3*(%pi*(r_Pb)^3); // Volume of Pb nucleus, metre cube
d_Pb = m_Pb/V_Pb; //Density of Pb nucleus,kg/metre cube
printf("\nThe density of oxygen nucleus : %4.2e in kg/metre cube",d_O);
printf("\nThe density of Pb nucleus : %4.2e in kg/metre cube",d_Pb);

// Result
// The density of oxygen nucleus : 3.73e+018 in kg/metre cube
// The density of Pb nucleus : 2.37e+017 in kg/metre cube
```

Scilab code Exa 1.4.6 Distance of closest approach

```
// Scilab code Exa1.4.6 : Determination of distance of closest approach for alpha-particle : Page 35 (2011)
E = 5.48*1.6e-013; // Energy of alpha particle, J
e = 1.6e-019; // Charge of an electron, C
Z = 79; // Mas number of Au nucleus,
epsilon_0 = 8.85e-012; // Permittivity of free space,
```

```
D = (2*Z*e^2)/(4*%pi*epsilon_0*E); // Distance of closest approach, m
printf("\nThe distance of closest appproach of alpha particle : %4.2e m", D)

// Result
// The distance of closest appproach of alpha particle : 4.15e-014 m
```

Scilab code Exa 1.4.7 Radius of Pb 208

```
// Scilab code Exa1.4.7 : Determination of radius of Pb-208 : Page 36 (2011)
A = 208; // Mass number of Pb-208
r0 = 1.2e-015; // Distance of closest approach, m
r = r0*((A)^(1/3)); // Radius of Pb-208, m
printf("\nThe radius of Pb-208 : %4.2e m", r)

// Result
//    The radius of Pb-208 : 7.11e-015 m
```

Scilab code Exa 1.5.1 Binding energy of alpha particle

```
// Scilab code Exa1.5.1 : Calculation of binding energy of alpha particle and express in MeV and joule : Page 36 (2011)
amu = 931.49; // Atomic mass unit, MeV
M_p = 1.00758; // Mass of proton, amu
M_n = 1.00897;  // Mass of neutron, amu
M_He = 4.0028; // Mass of He nucleus, amu
Z = 2; // Atomic number
N = 2; // Number of neutron
M_defect = Z*M_p+N*M_n-M_He;    // Mass defect, amu
```

```
BE_MeV = M_defect*amu; // Binding energy, MeV
BE_J = M_defect*1.49239e-010;     // Binding energy,
    J
printf("\nThe binding energy (in MeV): %5.2f",
    BE_MeV)
printf("\nThe binding energy (in J): %4.2e", BE_J)

// Result
// The binding energy (in MeV): 28.22
// The binding energy (in J): 4.52e-012
```

Scilab code Exa 1.5.2 Dissociation energy of C12

```
// Scilab code Exa1.5.2 : Calculation of energy
    required to break C-12 into 3-alpha particle :
    Page 37 (2011)
amu = 1.49239e-010; // Atomic mass unit, J
M_C = 12; // Mass of C-12, amu
M_a = 4.0026;     // Mass of alpha particle, amu
M_3a = 3*M_a; // Mass of 3 alpha particle, amu
D = M_C-M_3a; // Difference in two masses, amu
E = D*amu; // Required energy,J
printf("\nThe energy required to break 3 alpha
    particles : %4.2e J",E)

// Result
// The energy required to break 3 alpha particles :
    -1.16e-012 J
```

Scilab code Exa 1.5.3 Dissociation energy of helium nucleus

```
// Scilab code Exa1.5.3 : Calculation of energy
   required to knock out nucleon from He nucleus :
   Page 37 (2011)
M_p = 1.007895; // Mass of proton, amu
M_n = 1.008665; // Mass of neutron, amu
M_He = 4.0026; // Mass of He-nucleus, amu
Z = 2; // Number of proton
N = 2; // Number of neutron
D_m = [(Z*M_p)+(N*M_n)-M_He]; // Mass defect, amu
amu = 931.49; // Atomic mass unit, MeV
E = D_m*amu; // Required energy, MeV
printf("\nThe energy required to knock out nucleons
   from the He nucleus = %5.2f MeV", E);

// Result
// The energy required to knock out nucleons from
   the He nucleus = 28.43 MeV
```

Scilab code Exa 1.5.4 Binding energy of Fe 56

```
// Scilab code Exa1.5.4 : To calculate binding
   energy of Fe-56 : Page 38 (2011)
M_Fe = 55.934939; // Mass of Fe-56, amu
M_p = 1.007825; // Mass of proton, amu
M_n = 1.008665; // Mass of neutron, amu
Z = 26; // Atomic number of Fe-56
N = 30; // Number of neutron in Fe-56
amu = 931.49; // Atomic mass unit, MeV
BE = [(Z*M_p)+(N*M_n)-M_Fe]*amu; // Binding energy
   of Fe-56, MeV
printf("\nThe binding energy of Fe-56 : %6.4f MeV",
   BE)

// Result
// The binding energy of Fe-56 : 492.2561 MeV
```

Scilab code Exa 1.5.5 Mass defect and packing fraction

```
// Scilab code Exa1.5.5 : Calculation of mass defect
    and packing fraction from given data Page : 38
   (2011)
amu = 931.49; // Atomic mass unit, MeV
M_p = 1.007825; // Mass of proton, amu
M_n = 1.008663;  // Mass of neutron, amu
A = 2;        // Mass number of deutron, amu
M_D = 2.014103;    // Mass of deuteron nucleus, amu
M_Defect = (M_p+M_n-M_D)*amu;     // Mass defect of
   the nucleus, MeV
P_fraction = (M_D - A)/A;     // Packing fraction of
    nucleus
printf("\n Mass defect      %4.2f MeV\n Packing
   fraction    %7.5f",M_Defect,P_fraction);

// Result
//   Mass defect       2.22 MeV
//   Packing fraction    0.00705
```

Scilab code Exa 1.5.6 Average binding energy

```
// Scilab code Exa1.5.6 : To calculate binding
   energy per nucleon of He-4 nucleus : Page 38
   (2011)
m_p = 1.007825; // Mass of proton, amu
m_n = 1.008665; // Mass of neutron, amu
m_He = 4.002634; // Mass of He-4 nucleus, amu
amu = 931.47; // Atomic mass unit, MeV
A = 4, // Mass number of He-4 nucleus
```

```
BE = [2*m_p+2*m_n-m_He]*amu; // Binding energy of He
    -4 nucleus , MeV
Av_BE = BE/A; // Average binding energy or binding
    energy per nucleon , MeV
printf("\nThe binding energy per nucleon : %4.2f MeV
    ", Av_BE);

// Result
// The binding energy per nucleon of He-4 is
// The binding energy per nucleon : 7.07 MeV
```

Scilab code Exa 1.6.1 Orbital angular momentum of coupled nucleons

```
// Scilab code Exa1.6.1 : Orbital angular momentum
    of coupled nucleons : Page 39 (2011)
l1 = 1;     // Orbital qunatum number for p-state
    nucleon
l2 = 2;     // Orbital qunatum number for d-state
    nucleon
// Display the value of L within the for loop
disp("The possible L values will be");
for i = abs(l1-l2):1:abs(l1+l2)         // Coupling
    of l-orbitals
    printf("\t %1d",i);
end

// Result
// The possible L values will be
//   1     2     3
```

Scilab code Exa 1.6.2 Total angular momentum of proton

```
// Scilab code Exa1.6.2 : Total angular momentum of proton : Page 40 (2011)
// Get the l value from the user
l = 3;     // Orbital qunatum number for f-state proton
s = 1/2;     // Magnitude of spin quantum number
// Display the value of j within the for loop
disp("The j values will be between");
for i = abs(l-s):1:abs(l+s)          // l-s Coupling
    printf("\t %3.1f",i);
end

// Result
// The j values will be between
//  2.5     3.5
```

Scilab code Exa 1.11.1 Ion accelerated in a mass spectrograph

```
// Scilab code Exa1.11.1 : To find the speed, mass and mass number of the ion which is accelerated in a mass spectrograph : Page 40 (2011)
V = 1000; // Potential difference, volts
R = 0.122; // Radius of the circular path, m
B = 1500e-04; // Magnetic field, tesla
e = 1.602e-019; // Charge of the electron, C
amu = 1.673e-027; // Atomic mass unit, kg
v = (2*V)/(R*B); // Speed of the ion, m/s
M = 2*e*V/v^2; // Mass of the ion, kg
A = M/amu; // Mass number
printf("\n    Speed    >  %5.3e m/s  \n    Mass    > %5.3e kg  \n    Mass number    > %5.2f ",v, M, A);

// Result
//
```

```
//      Speed     >   1.093e+005 m/s
//     Mass      >   2.682e-026 kg
//       Mass number     >   16.03
```

Scilab code Exa 1.11.2 Distance between isotopic Ar ions

```
// Scilab code Exa 1.11.2 : To determine distances
    between the isotopic Ar ions in Bainbridge mass
    spectrograph : Page 41 (2011)
amu = 1.673e-027; // Atomic mass unit, kg
E = 5e+04; // Electric field, V/m
B1 = 0.4; // Magnetic field, tesla
v = E/B1; // Velocity of ions, m/s
B = 0.8; // Magnetic field, tesla
e = 1.602e-019; //charge of electron ,C
m_Ar = zeros(1,3);    // Array of masses of three Ar
     ions, amu
m_Ar(1,1) = 36,m_Ar(1,2) = 38,m_Ar(1,3) = 40; //
    Masses of three isoptopes of Ar, amu
r_Ar = zeros(1,3);    // Array of radii of three Ar
    ions, mm
for i = 1:1:3
    r_Ar(1,i) = (m_Ar(1,i)*amu*v)/(B*e)*1e+03; //
        Radius of Ar ion orbit, mm
    disp(r_Ar(1,i));
end
d1 = 2*(r_Ar(1,2)-r_Ar(1,1));    // Distance b/w
    first and second line, mm
d2 = 2*(r_Ar(1,3)-r_Ar(1,2));    // Distance b/w
    second and third line, mm
printf("\nThe distance between successive lines due
    to three different isotopes : %3.1f mm and %3.1f
    mm", d1,d2);

// Result
```

```
// The distance between successive lines due to
   three different isotopes : 6.5 mm and 6.5 mm
```

Chapter 2

Nuclear Models

Scilab code Exa 2.2.1 Binding energy and percentage discrepancy

```
// Scilab code Exa2.2.1 To calculate the binding
   energy of Ca(20,40) and %_age discrepancy : Page
   66 (2011)
// For Ca(20,40), actual binding energy is ......
m_p = 1.007825; // Mass of proton, amu
m_n = 1.008665; // Mass of neutron, amu
Z = 20; // Number of protons
N = 20;  // Number of neutrons
M_n = 39.962591; // Mass of the nucleus, amu
B_actual = (M_n-Z*m_p-N*m_n)*931.49; // Actual
   binding energy, MeV
// For Ca(20,40), Binding energ as per semiemperical
    mas formula......
Z = 20; // Number of protons
a_v = 15.5; // Volume constant, MeV
a_s = 16.8; // Surface constant, MeV
a_a = 23.0; // Asymmetric constant, MeV
a_c = 0.7; // Coulomb constant, MeV
a_p = 34.0; // Paring constant, MeV
A = 40; // Mass number
B_semi = [a_v*A-(a_s*A^(2/3))-(a_c*Z*(Z-1)/A^(1/3))
```

```
-(a_a*(A-2*Z)^2/A)-(a_p*A^(-3/4))]; // Binding energy as per semiemperical mass formula
// Percentage discrepancy between actual and semiemperical mass formula values are.......
Per_des = -(B_semi+B_actual)/B_actual*100; // Percentage discrepancy
printf("\nActual binding energy = %6.2f MeV\nBinding energy as per semiemperical mass formula = %6.2f MeV\nPercentage discrepancy = %3.1f percent", B_actual, B_semi, Per_des);

// Result
// Actual binding energy = -342.05 MeV
// Binding energy as per semiemperical mass formula = 343.59 MeV
// Percentage discrepancy = 0.4 percent
```

Scilab code Exa 2.2.2 Coulomb energies and nucelon masses of mirror nuclei

```
// Scilab code Exa2.2.2 To calculate the difference in coulomb energy and nucleons' mass difference for mirror nuclei and show in agreement with actual mass difference Page 67 (2011)
 // Calculation of coulomb energy for mirror nuclei : N-7 and O-8
 // For N-7 nucleus
a_c = 0.7; // Coulomb energy constant, MeV
Z_N = 7;  //  Atpmic no.
A = 15; // Atomic mass
E_C_N = a_c*Z_N*(Z_N-1)/(A^(1/3)); // Coulomb energy for N-7, MeV
// For O-8 nucleus
a_c = 0.7; // Coulomb energy constant, MeV
Z_O = 8;  //  Atpmic no.
```

```
A = 15; // Atomic mass
E_C_O = a_c*Z_O*(Z_O-1)/(A^(1/3)); // Coulomb energy for O-8, MeV
C_E_d = E_C_O-E_C_N; // Coulomb energy difference, MeV
m_p = 1.007276*931.49; // Mass of proton, MeV
m_n = 1.008665*931.49; // Mass of neutron, MeV
M_d = m_n-m_p; // Mass difference of nucleons, MeV
D_C_M = round(C_E_d-M_d); // Difference in coulomb energy and nucleon mass difference, MeV
M_O = 15.003070*931.49; // Mass of O-8, MeV
M_N = 15.000108*931.49; // Mass of N-7, MeV
D_A = round(M_O-M_N); // Actual mass difference, MeV
printf("\nDifference in Coulomb energy = %5.3f MeV\nNucleon mass difference  = %6.4f MeV\nDifference in Coulomb energy and nucleon mass difference = %5.3f MeV\nActual mass difference = %5.3f MeV", C_E_d, M_d ,D_C_M, D_A);
if D_A == D_C_M then printf("\nResult is verified")
end
// Result
// Difference in Coulomb energy = 3.974 MeV
// Nucleon mass difference  = 1.2938 MeV
// Difference in Coulomb energy and nucleon mass difference = 3.000 MeV
// Actual mass difference = 3.000 MeV
// Result is verified
```

Scilab code Exa 2.2.3 Neutron binding energy for isotopes of krypton

```
// Scilab code Exa2.2.3 To calculate the energy required to remove a neutron from Kr-81, Kr-82, Kr-83 : Page 68 (2011)
// For Kr-80,
m_p = 1.007825; // Mass of proton, amu
```

```
m_n = 1.008665; // Mass of neutron , amu
Z = 36; // Number of protons
N_80 = 44;  // Number of neutrons
M_n_80 = 79.91628; // Mass of Kr nucleus
BE_Kr_80 = (Z*m_p+N_80*m_n-M_n_80)*931.49; // Binding energy for Kr-80, MeV
// For Kr-81,
N_81 = 45; // Number of neutrons
M_n_81 = 80.91661; // Mass of Kr-81 nucleus
BE_Kr_81 = (Z*m_p+N_81*m_n-M_n_81)*931.49; // Binding energy for Kr-81 nucleus
// For Kr-82
N_82 = 46;  // Number of neutrons
M_n_82 = 81.913482; // Mass of Kr nucleus
BE_Kr_82 = (Z*m_p+N_82*m_n-M_n_82)*931.49;  // Binding energy for Kr-82,MeV
// For Kr-83
N_83 = 47; // Number of protons
M_n_83 = 82.914134; // Mass of Kr-83 nucleus
BE_Kr_83 = (Z*m_p+N_83*m_n-M_n_83)*931.49; // Binding energy for Kr-83, MeV
E_sep_81 = BE_Kr_81-BE_Kr_80; // Energy seperation of neutron for Kr-81, MeV
E_sep_82 = BE_Kr_82-BE_Kr_81; // Energy seperation of neutron for Kr-82, MeV
E_sep_83 = BE_Kr_83-BE_Kr_82; // Energy seperation of neutron for Kr-83, MeV
,
printf("\nEnergy seperation of neutron for Kr-81 = %4.2f MeV\nEnergy seperation of neutron for Kr-82 = %4.2f MeV\nEnergy seperation of  neutron for Kr-83 = %5.2f MeV", E_sep_81, E_sep_82, E_sep_83);

// Result
// Energy seperation of neutron for Kr-81 = 7.76 MeV
// Energy seperation of neutron for Kr-82 = 10.99 MeV
```

```
// Energy seperation of  neutron for Kr−83 =  7.46
   MeV
```

Scilab code Exa 2.2.4 Isotopic stability

```
// Scilab code Exa2.2.4 To determine the most stable
    isotope of A = 75 : Page 68 (2011)
a_v = 15.5; // Volume energy coefficient , MeV
a_s = 16.8; // Surface energy coefficient MeV
a_c = 0.7; // Coulomb energy coefficient , MeV
a_a = 23.0; // Asymmetric energy coefficient , MeV
a_p = 34.0; // Pairing energy coefficient , MeV
A = 75; //  Given atomic mass
z = poly(0, 'z'); // z declares  a polynomial
B = -a_c*z*(z-1)/A^(1/3)-a_a*(A-2*z)^2/A ; //
   Binding energy as per liquid drop model
dB = derivat(B); // Differentiate B w.r.t. z
z = roots(dB); // Isotope of A = 75
z_i = round(z); // Most stable isotope of A = 75
printf("\nMost stable isotope of A = 75 corresponds
   to Z = %d ", z_i)

//    Result
//  Most stable isotope of A = 75 corresponds to Z =
    33
```

Scilab code Exa 2.2.5 Stable isotopes for different mass numbers

```
// Scilab code Exa2.2.5 To determine the most stable
    isotopes for A = 27, A = 118, A = 238 : Page 69
   (2011)
a_v = 15.5; // Volume energy , MeV
a_s = 16.8; // Surface energy , MeV
```

```
a_c = 0.7; // Coulomb energy, MeV
a_a = 23.0; // Asymmetric energy, MeV
a_p = 34.0; // Pairing energy, MeV
z = poly(0, 'z')
// For A = 27;
B_27 = -a_c*z*(z-1)/27^(1/3)-a_a*(27-2*z)^2/27 ; // Binding energy as per liquid drop model
dB_27 = derivat(B_27) // Differentiate B w.r.t. z
z_27 = roots(dB_27)   // Isotope of A = 27
z_i_27 = round(z_27)  // Most stable isotope of A = 27
// For A = 118
B_118 = -a_c*z*(z-1)/118^(1/3)-a_a*(118-2*z)^2/118 ; // Binding energy as per liquid drop model
dB_118 = derivat(B_118)  // Differentiate B w.r.t. z
z_118 = roots(dB_118)    // Isotope of A = 118
z_i_118 = round(z_118)   // Most stable isotope of A = 118
// For A = 238
B_238 = -a_c*z*(z-1)/238^(1/3)-a_a*(238-2*z)^2/238 ; // Binding energy as per liquid drop model
dB_238 = derivat(B_238); // Differentiate B w.r.t. z
z_238 = roots(dB_238); // Isotope of A = 238
z_i_238 = round(z_238); // Most stable isotope of A = 238
printf("\nMost stable isotopes for A = 27, A = 118, A = 238 corresponds to z = %d, %d and %d respectively", z_i_27, z_i_118, z_i_238);

// Result
// Most stable isotopes for A = 27, A = 118, A = 238 corresponds to z = 13, 50 and 92 respectively
```

Scilab code Exa 2.2.6 Coulomb energy coefficient of mirror nuclei

```
// Scilab code Exa2.2.6 : To calculate the coulomb coefficient and estimate nuclear radius for mirror nuclei: Page no. 69 (2011)
// Mirror nuclei : Na-11 and Mg-12
m_p = 1.007276; // Mass of proton, amu
m_n = 1.008665; // Mass of neutron, amu
M_Mg = 22.994124; // Atomic mass of Mg-12, amu
M_Na = 22.989768; // Atomic mass of Na-11, amu
A = 23; // Mass number
Z_Mg = 12; // Atomic number of Mg-12
e = 1.6e-019; // Charge of the electron, C
K = 8.98e+09; // Coulomb force constant
a_c = A^(1/3)/(2*Z_Mg-1)*[(M_Mg-M_Na)+(m_n-m_p)]*931.47; // Coulomb coefficient, MeV
r_0 = 3/5*K*e^2/(a_c*1.6e-013); // Nuclear radius, m
printf("\nCoulomb coefficient = %4.2f MeV\nNuclear radius = %3.1e m", a_c, r_0)
// Result
// Coulomb coefficient = 0.66 MeV
// Nuclear radius = 1.3e-015 m
```

Scilab code Exa 2.2.7 Coulomb and surface energies of uranium

```
// Scilab code Exa2.2.7 To calculate coulomb energy and surface energy for U(92,236) : Page 71 (2011)
Z = 92; // Atomic number of U-236
e = 1.6e-019; // Charge of an electron, C
A = 236; // Mass number of U-236
K = 8.98e+09; // Coulomb constant,
r_o = 1.2e-015; // Distance of closest approach, m
a_s = -16.8; // Surface constant
E_c = -(3*K*Z*(Z-1)*e^2)/(5*r_o*A^(1/3)*1.6e-013); // Coulomb energy, MeV
E_s = a_s*A^(2/3); // Surface energy, MeV
printf("\nCoulomb energy for U(92,236) = %5.1f MeV
```

```
    \nSurface energy for U(92,236)    = %5.1f MeV  ",
    E_c, E_s)
//   Result
//  Coulomb energy for U(92,236)   = -973.3 MeV
//   Surface energy for U(92,236)    = -641.6 MeVS
```

Scilab code Exa 2.3.1 Mass of decayed radioactive material

```
// Scilab code Exa2.3.1 To calculate the mass of
    decayed radioactive material: Page 126 (2011)
t_prime =  1600; // Half life of radioactive
    material, years
t = 2000; // Total time, years
lambda = 0.6931/t_prime; // Decay constant, years
    ^(-1)
m0 = 1; // The mass of radioactive substance at t0,
    mg
m = m0* %e^(-(lambda*t)); // Ratio of total number
    of atoms and number of atoms disintegrat, mg
a = 1-m; // The amount of radioactive substance
    decayed, mg
printf("\nThe amount of radioactive substance
    decayed : %6.4f mg", a)

//   Result
//  The amount of radioactive substance decayed :
    0.5795 mg
```

Scilab code Exa 2.3.4 Magnetic moment of nuclei

```
// Scilab code Exa2.3.4 : To calculate the magnetic
    moment of given nuclei  : Page no. 74 : (2011)
// For Ne(10.19) nucleus
```

```
j_Ne_9 = 5/2; // Total angular momentum for Ne-19 nucleus
u_Ne_9 = j_Ne_9+2.29; // Magnetic moment of Ne-19 nucleus , nuclear magneton
// For Ne(10,20) nucleus
j_Ne_10 = 0;//  Total angular momentum for Ne-20 nucleus
u_Ne_10 = j_Ne_10+2.29; // Magnetic moment of Ne-20 nucleus, nuclear magneton
// For Ne(10,21) nucleus
j_Ne_11 = 5/2;//  Total angular momentum for Ne-21 nucleus
u_Ne_11 = j_Ne_11+2.29; // Magnetic moment of Ne-21 nucleus, nuclear magneton
printf("\nMagnetic moment of Ne-19 nucleus = %4.2f nuclear magneton\nMagnetic moment of Ne-20 nucleus = %4.2f nuclear magneton\nMagnetic moment of Ne-21 nucleus = %4.2f nuclear magneton", u_Ne_9, u_Ne_10, u_Ne_11);
// Result
// Magnetic moment of Ne-19 nucleus = 4.79 nuclear magneton
// Magnetic moment of Ne-20 nucleus = 2.29 nuclear magneton
// Magnetic moment of Ne-21 nucleus = 4.79 nuclear magneton
```

Chapter 3

Radioactivity

Scilab code Exa 3.2.1 Curie becquerel relation

```
// Scilab code Exa3.2.1: To determine how many curie in 10^10 Bq : Page 124 (2011)
Bq = 1/3.7e+010; // Number of curie in one Bq, Ci
N = 10^10*Bq; // The number of curie in 10^10 Bq, Ci
printf("\nThe number of curie in 10^10 Bq : %4.2f Ci", N)
//   Result
//  The number of curie in 10^10 Bq : 0.27 Ci
```

Scilab code Exa 3.2.2 Activity of thorium

```
// Scilab code Exa3.2.2: To calculate the activity of 10g of Th-232 : Page 125 (2011)
lambda_232 = 1.58e-018; // Decay constant, s^-1
N = 2.596e+022; // Number of atoms in 10g Th-232
A = N*lambda_232; // The activity of 10g of Th-232, dps
printf("\nThe activty of 10g of Th-232 : %5.3e dps", A)
```

```
//    Result
//    The activty of 10g of Th-232 : 4.102e+004 dps
```

Scilab code Exa 3.2.3 Mass of radiactive sample

```
// Scilab code Exa3.2.3: Calculation of mass of 1 Ci sample of radioactive sample : Page 125 (2011)
A = 3.7e+010; // Activity of 1Ci sample, dps
t = 1608; // Half life of radioactive substance, s
N = 6.023e+023/214; // Number of atoms in 1g of substance having atomic mass 214
lambda = 0.6931/t; // Decay constant, s^-1
m = A/(lambda*N); // The mass of radoiactive substance, g
printf("\nThe mass of radioactive substance : %4.2e g", m)
//    Result
//      The mass of radioactive substance : 3.05e-008 g
```

Scilab code Exa 3.2.4 Activity of 1 kg of uranium

```
// Scilab code Exa3.2.4: To calculate the activity of 1kg of U-238: Page 125 (2011)
t =  1.419e+017; // Half life of U-238, s
N = 6.023e+023/238; // Number of atoms in 1g of U-238
lambda = 0.6931/t; // Decay constant, s^-1
A = (lambda*N)*1000/(3.7e+010); // The activity of 1 kg of U-238, Ci
printf("\nThe activity of 1kg of U-238 : %4.2e Ci", A)
//    Result
```

```
// The activity of 1kg of U-238 : 3.34e-004 Ci
```

Scilab code Exa 3.2.6 Half life of radioactive material

```
// Scilab code Exa3.2.6 Determination of half life
   of radioactive material Page 127 (2011)
t = 10; // Total period of radioactive material,
   days
lambda = log(6.6667)/10; //Decay constant, day^-1
t_h = 0.6931/(lambda); // Half life of radioactive
   substance, days
printf("\nThe half life of radioactive substance :
   %4.2f days", t_h)
// Result
// The half life of radioactive substance :
   3.65 days
```

Scilab code Exa 3.2.7 Mass of Ra 226

```
// Scilab code Exa3.2.7 : To calculate the mass of
   Ra-226 :Page no. 127 (2011)
t_h = 1620*31536000; // Half life of Ra-226, S
D = 0.6931/t_h; // Decay constant, S^-1
A_Ci = 3.7e+010; // Activity, Ci
N_Ci = A_Ci/D; // Number of atoms decayed
m = 0.226; // Mass of 6.023e+023 atoms, kg
M_Ci = m*N_Ci/6.023e+023; // Mass of 1-Ci sample of
   Ra-226, kg
A_rf = 10^6; // Activity, Rf
N_rf = A_rf/D; // Number of atoms decayed
M_rf = m*N_rf/6.023e+023; // Mass of 1-Rf sample of
   Ra-226, kg
```

```
printf("\n Mass of 1-Ci sample of Ra-226   = %5.3e
    kg and \n Mass of 1-Rf sample of Ra-226  =  %4.2e
     kg ",M_Ci, M_rf )
//   Result
// Mass of 1-Ci sample of Ra-226   = 1.023e-003 kg
    and
// Mass of 1-Rf sample of Ra-226  =  2.77e-008 kg
```

Scilab code Exa 3.2.8 Activity and weight of radiactive material

```
// Scilab code Exa3.2.8 To calculate the activity
    and weight of radioactive material : Page 128
    (2011)
N_o = 7.721e+018; // Number of atoms in 3 mg of U
    -234
t_h = 2.5e+05; // Half life of U-234, years
T = 150000; // Total time, years
lambda = 0.6931/t_h; // Decay constant, year^-1
N = N_o*(%e^-(lambda*T)); // Number of atoms left
    after T years
m = 234000; // Mass of 6.023e+023 atoms of U-234, mg
M = m*N/(6.023e+023); // Weight of sample left after
    t years,
L = 8.8e-014; // Given decay constant, S^-1
A = N*L*10^6/(3.7e+010); // Activity, micro Ci
printf("\nThe weight of sample  =  %5.3f mg  \n
    Activity   =  %5.2f micro Ci ", M, A)
//   Result
//     The weight of sample  =  1.979 mg
//      Activity   =  12.12 micro Ci
```

Scilab code Exa 3.2.9 Activity of K 40

```
// Scilab code Exa3.2.9 : To calculate the activity
    of K-40 : Page no. 129 (2011)
N = 6.324e+020; // Number of atoms in 4.2e-05 kg of
   K-40
t_h = 1.31e+09*31536000; // Half life of K-40, s
D = 0.693/t_h; // Decay constant , s^-1
A = N*D/(3.7e+010)*10^6; // Activity of K-40,
    microCi
printf("\nThe activity of K-40 : %5.3f micro Ci", A
    )
//    Result
//     The activity of K-40 : 0.287 micro Ci
```

Scilab code Exa 3.2.10 Power in radioactive decay

```
// Scilab code Exa3.2.10 : To calculate the power
    produced by 10 mg of Po-210  : Page no. 130
    (2011)
N = 2.87e+019; // Number of atoms in 10e-10kg of Po
    -210
t_h = 138*24*3600; // Half life of Po-210, s
D = 0.693/t_h; // Decay constant , s^-1
A = N*D; // Activity of K-40, dps
E = 5.3*1.6e-013; // Power produce by one dps , MeV
P = A*E; // Power produced by 1.667e+012 dps , W
printf("\nThe Power produced by 1.667e+012 dps : %3
    .1f W", P)
//    Result
//    The Power produced by 1.667e+012 dps : 1.4 W
```

Scilab code Exa 3.3.1 Emitted particles during nuclear disintegration

```
// Scilab code Exa 3.3.1 : Finding particles in the
    given reactions : page no. 131 (2011)
// Declare three cells (for three reactions)
R1 = cell(4,3);
R2 = cell(4,3);
R3 = cell(3,3);

// Enter data for first cell (Reaction)
R1(1,1).entries = "Pb";
R1(1,2).entries = 82;
R1(1,3).entries = 211;
R1(2,1).entries = 'Bi';
R1(2,2).entries = 83;
R1(2,3).entries = 211;
R1(3,1).entries = 'Tl';
R1(3,2).entries = 81;
R1(3,3).entries = 207;
R1(4,1).entries = 'Pb';
R1(4,2).entries = 82;
R1(4,3).entries = 207;

// Enter data for second cell (Reaction)
R2(1,1).entries = "U";
R2(1,2).entries = 92;
R2(1,3).entries = 238;
R2(2,1).entries = 'Th';
R2(2,2).entries = 90;
R2(2,3).entries = 234;
R2(3,1).entries = 'Pa';
R2(3,2).entries = 91;
R2(3,3).entries = 234;
R2(4,1).entries = 'U';
R2(4,2).entries = 92;
R2(4,3).entries = 234;

// Enter data for third cell (Reaction)
R3(1,1).entries = "Bi";
R3(1,2).entries = 83;
```

```
R3(1,3).entries = 211;
R3(2,1).entries = 'Pa';
R3(2,2).entries = 84;
R3(2,3).entries = 211;
R3(3,1).entries = 'Pb';
R3(3,2).entries = 82;
R3(3,3).entries = 207;

// Declare a function returning the type of particle emitted
function particle = identify_particle(d_Z, d_A)
      if d_Z == 2 & d_A == 4 then
       particle = "Alpha";
   elseif d_Z == -1 & d_A == 0 then
       particle = "Beta minus";
   elseif d_Z == 1 & d_A == 0 then
       particle = "Beta plus";
      end
endfunction

// Display emitted particles for first reaction
printf("\n\n\nReaction-I:");
for i = 1:1:3
        dZ = R1(i,2).entries-R1(i+1,2).entries;
        dA = R1(i,3).entries-R1(i+1,3).entries;
        p = identify_particle(dZ,dA);
        printf("\n%s(%d) - (%s) --> %s(%d)", R1(i,1).entries, R1(i,2).entries, p, R1(i+1,1).entries, R1(i+1,2).entries);
end

// Display emitted particles for second reaction
printf("\n\n\nReaction-II:");
for i = 1:1:3
        dZ = R2(i,2).entries-R2(i+1,2).entries;
        dA = R2(i,3).entries-R2(i+1,3).entries;
        p = identify_particle(dZ,dA);
        printf("\n%s(%d) - (%s) --> %s(%d)", R2(i,1)
```

```
            .entries, R2(i,2).entries, p, R2(i+1,1).entries, R2(i+1,2).entries);
end

// Display emitted particles for third reaction
printf("\n\n\nReaction-III:");
for i = 1:1:2
        dZ = R3(i,2).entries-R3(i+1,2).entries;
        dA = R3(i,3).entries-R3(i+1,3).entries;
        p = identify_particle(dZ,dA);
        printf("\n%s(%d) - (%s) --> %s(%d)", R3(i,1).entries, R3(i,2).entries, p, R3(i+1,1).entries, R3(i+1,2).entries);
end

// Result
//
// Reaction-I:
// Pb(82) - (Beta minus) --> Bi(83)
// Bi(83) - (Alpha) --> Tl(81)
// Tl(81) - (Beta minus) --> Pb(82)


// Reaction-II:
// U(92) - (Alpha) --> Th(90)
// Th(90) - (Beta minus) --> Pa(91)
// Pa(91) - (Beta minus) --> U(92)


// Reaction-III:
// Bi(83) - (Beta minus) --> Pa(84)
// Pa(84) - (Alpha) --> Pb(82)
```

Scilab code Exa 3.3.2 Energy of Pb decay

```
// Scilab code Exa 3.3.2 To calculate mass number of Pb isotope and energy emitted : Page no : 132 (2011)
M_U = 238.050786; // Atomic mass of U-238, amu
M_Pb = 205.9744550; // Atomic mass of Pb-205, amu
M_He = 4.002603; // Atomic mass of He-4, amu
M_e = 5.486e-04; // Atomic mass of electron , amu
M = M_Pb+(8*M_He)+(6*M_e); // Total mass of products , amu
D = M_U-M; // Decrease in mass , amu
E = D*931.47; // Energy evolved , MeV
printf("\nTotal mass of products  = %1.7f amu \n Decrease in mass   = %9.7f amu and \n Energy evolved   = %4.1f MeV", M, D,  E)
// Result
//     Total mass of products  = 237.9985706 amu
//  Decrease in mass   =  0.0522154 amu and
//   Energy evolved   = 48.6 MeV
```

Scilab code Exa 3.4.1 Atomic and mass numbers of daughter nuclei

```
// Finding atomic No. and mass No. of daughter nuclei in the given reactions : Page No. 133(2011)
// Declare  cell (for given reaction)
R1 = cell(5,4);
// Enter data for cell (Reaction-I)
R1(1,1).entries = "A";
R1(1,2).entries = 90;
R1(1,3).entries = 238;
R1(1,4).entries = "Alpha";
R1(2,1).entries = 'B';
R1(2,4).entries = "Beta minus";
R1(3,1).entries = 'C';
R1(3,4).entries = "Alpha";
```

```
R1(4,1).entries = 'D';
R1(4,4).entries = "Beta minus";
R1(5,1).entries = 'E';

// Declare a function returning the type of particle emitted
function [Z, A] = daughter_nucleus(particle_emitted)
      if particle_emitted == "Alpha" then
          Z = 2, A = 4;
   elseif particle_emitted == "Beta minus" then
          Z = -1, A = 0;
   elseif particle_emitted == "Beta plus" then
          Z = 1, A = 0;
      end
endfunction

// Display emitted particles for first reaction
printf("\n\n\nReaction-I:");
for i = 1:1:4
        [Z, A] = daughter_nucleus(R1(i,4).entries);
        R1(i+1,2).entries = R1(i,2).entries-Z;
        R1(i+1,3).entries = R1(i,3).entries-A;
        printf("\n%s(%d,%d) - (%s) --> %s(%d,%d)", R1(i,1).entries, R1(i,2).entries, R1(i,3).entries, R1(i,4).entries, R1(i+1,1).entries, R1(i+1,2).entries, R1(i+1,3).entries)
        ;
end
// Result
//
// Reaction-I:
// A(90,238) - (Alpha) --> B(88,234)
// B(88,234) - (Beta minus) --> C(89,234)
// (89,234) - (Alpha) --> D(87,230)
// D(87,230) - (Beta minus) --> E(88,230)
```

Scilab code Exa 3.4.2 Number of half lives of Rn 222

```
// Scilab code Exa 3.4.2 : To determine the number of Rn-222 half lives elapsed when it reaches 99% of its equilibrium concentration : Page no. 133 : (2011)
 D = log(2); // Decay constant , s^-1
 t = log(100); // Half life , s
 n = t/D; // Number of half-lives
printf("\n Number of half-lives : %4.2f ", n)
// Result
//       Number of half-lives : 6.64
```

Scilab code Exa 3.4.3 Decay constant for alpha and beta decays

```
// Scilab code Exa 3.4.3 : To calculate the  decay constant for alpha and beta decays : Page no. 133 : (2011)
 H_t = 60.5*60; // Total half life period , s
 T_d = 0.693/H_t; // Total decay constant , s^-1
 A_d = 34/100*T_d; // Decay constant for alpha decays , s^-1
 B_d = 66/100*T_d; // Decay constant for beta decay , s^-1
printf("\n Alpha decay    =  %4.2e s^-1     \n Beta decay     = %4.2e s^-1", A_d, B_d)
// Result
//       Alpha decay    =  6.49e-005 s^-1
//       Beta decay     = 1.26e-004 s^-1
```

Scilab code Exa 3.4.4 Half life of uranium 234

```
// Scilab code Exa 3.4.4 : To calculate the half
   life of U(92,234): Page no. 134 : (2011)
A_r = 1.8e+04; // Atomic ratio of U(92,238) and U
   (92,234)
T_238  = 2.5e+05; // Half life of U(92,238), years
T_234 = A_r*T_238; // Half life of U(92,234), years
printf("\n Half life of U(92,234): %3.1e years",
   T_234)
// Result
//        Half life of U(92,234): 4.5e+009 years
```

Scilab code Exa 3.4.5 Decayed amount of radioactive matter

```
// Scilab code Exa3.2.5 To calculate the mass of
   decayed radioactive material: Page 126 (2011)
t_h =  1600; // Half life of radioactive material,
   years
t = 2000; // Totaltime, years
lambda = 0.6931/t_h; // Decay constant, years^-1
m0 = 1; // The mass of radioactive substance at t0,
   mg
m = m0* %e^(-(lambda*t)); // Ratio of total number
   of atoms and number of atoms disintegrat, mg
A = 1-m; // The amount of radioactive substance
   decayed, mg
printf("\nThe amount of radioactive substance
   decayed : %6.4f mg",A)
//   Result
//     The amount of radioactive substance decayed :
     0.5795 mg
```

Scilab code Exa 3.5.2 Kinetic energy of alpha particle

```
// Scilab code Exa 3.5.2 : To calculate the K.E. of alpha particle in following decay   Pu-239 to U-235+He-4
M_239 = 239.052158; // Atomic mass of Pu-239, amu
M_235 = 235.043925; // Atomic mass of U-235, amu
M_4 = 4.002603; // Atomic mass of He-4, amu
Q = (M_239-M_235-M_4)*931.47; // Difference in masses, MeV
A = 241; // Mass number
K_alpha = Q*(A-4)/A; // Kinetic energy of alpha particle, MeV
printf("\nKinetic energy of alpha particle %5.2f MeV", K_alpha)
// Result
//          Kinetic energy of alpha particle  5.16 MeV
```

Scilab code Exa 3.5.3 Height of barrier faced by alpha particle

```
// Scilab code Exa 3.5.3 : To calculate the height of barrier faced by alpha particle of Ra-226 : Page no. : 136 (2011)
Z = 88; // Atomic number of Ra-226 nucleus,
A = 226; // Atomic mass of Ra-226 nucleus
R_0 = 1.3e-015; // Distance of closest approach, m
E_0 = 8.854e-012; // Permittivity of free space, C^2/Nm^2
e = 1.6e-019; // Charge of an electron, C
B = 2/(1.6e-013)*(Z-2)*e^2/(4*%pi*E_0*R_0*A^(1/3)); // The barrier height faced by alpha particle, MeV
printf("\nThe barrier height faced by alpha particle : %4.1f MeV", B)
// Result
```

```
//          The barrier height faced by alpha particle : 31.2 MeV
```

Scilab code Exa 3.5.4 Height of coulomb barrier

```
// Scilab code Exa 3.5.4 : To calculate the height of coulomb barrier faced by alpha particle : Page no. : 136 (2011)
Z_1 = 2; //Atomic number of He-4,
Z_2 = 7; // Atomic number of N-14,
A_1 = 4; // Atomis mass of He-4 nucleus
A_2 = 14; // Atomic mass of N-14 nucleus
R_0 = 1.5e-015; // Distance of closest approach, m
E_0 = 8.854e-012; // Permittivity of free space, C^2/Nm^2
e = 1.6e-019; // Charge of an electron, C
B = Z_1/(1.6e-013)*Z_2*e^2/(4*%pi*E_0*R_0*(A_1^(1/3)+A_2^(1/3))); // The coulomb barrier faced by alpha particle, MeV
printf("\nThe coulomb barrier  faced by alpha particle : %4.2f MeV", B)
// Result
//     The coulomb barrier  faced by alpha particle : 3.36 MeV
```

Scilab code Exa 3.5.5 KE of a proton to penetrate the barrier

```
// Scilab code Exa 3.5.5 : To calculate the K.E. of a proton to penetrate the barrier of H nucleus : Page no. : 137 (2011)
R_0 = 1.2; // Distance of closest approach, m
E_b = 197/(R_0*137); // The K.E. of proton to penetrate the berrier of H nucleus, Mev
```

```
printf("\nThe K.E. of proton to penetrate the
    berrier of H nucleus : %3.1f MeV", E_b)
// Result
//          The K.E. of proton to penetrate the
    berrier of H nucleus : 1.2 MeV
```

Scilab code Exa 3.6.1 Mass of daughter nucleus

```
// Scilab code Exa 3.6.1 : To determine the mass of
    daughter nucleus for given reaction : Page no.
    138 : (2011)
 M_C = 14.007685; // Mass of C-14 nucleus, amu
 E_e = 0.156/931.47; // Kinetic energy of emitted
     electron, amu
 M_N = M_C-E_e; // Mass of N-14 nucleus, amu
printf("\n Mass of N-14 nucleus : %9.6f amu", M_N)
// Result
//       Mass of N-14 nucleus : 14.007518 amu
```

Scilab code Exa 3.6.3 Number of proton decayed per year from water

```
// Scilab code Exa. 3.6.3 : To determine the number
    of proton decayed  per year from H2O in a
    reservior : Page no. 139 : (2011)
N_p = 6.70e+033;// Number of protons
T_p = 10^32; // Mean life of proton, years
D_p = N_p/T_p*0.5; // Number of proton decays per
    year, decays/year
printf("\n Number of proton decays per year,: %4.1f
    decays/year", D_p)
// Result
//       Number of proton decayed per year: 33.5
    decays/year
```

Scilab code Exa 3.7.1 Energy of gamma photons from excited Ni 60

```
// Scilab code Exa. 3.7.1 : To determine the energies of two gamma rays emitted  during de-excitation of Ni-60: Page no. 141 : (2011)
E_2 = 2505; // Second excited state of Ni-60, KeV
E_1 = 1332; // First excited state of Ni-60, KeV
E_0 = 0; // Ground state of Ni-60 , KeV
E_G_2 = E_2-E_1; // Energy of gamma rays emitted when transition from 2 to 1, KeV
E_G_1 = E_1-E_0; // Energy of gamma rays emitted when transition from 1 to 0, KeV
printf("\n Energies of two gamma rays emitted : %d KeV and %d KeV", E_G_2, E_G_1)
// Result
//      Energy of two gamma rays emitted : 1173 KeV and 1332 KeV
```

Scilab code Exa 3.7.2 Conversion energies for K and L shell electrons

```
// Scilab code Exa. 3.7.2 : To determine the energies conversion for K and L-shell electrons for reaction Cs(55,137) = Ba(56,137)+e(-1,0): Page no. 141 : (2011)
E = 662; // Energy available with the nucleus, KeV
I_b_K = 37.4; // Binding energy for K-shell, KeV
I_b_L = 6.0; // Binding energy for L-shell, KeV
E_c_K = E-I_b_K; // Energy conversion for K-shell, KeV
E_c_L = E-I_b_L; // Energy conversion for L-shell, KeV
```

```
printf("\n Energies conversion for K and L-shell
    electrons : %5.1f KeV and %d KeV", E_c_K, E_c_L)
// Result
//       Energies conversion for K and L-shell
    electrons : 624.6 KeV and 656 KeV
```

Scilab code Exa 3.9.1 Age of uranium mineral

```
// Scilab code Exa. 3.9.1 : To calculate the age of
    uranium mineral: Page no. 143 : (2011)
t_h = 4.5e+09; // Half life of mineral, years
D_c = 0.6931/t_h; // Decay constant of minerals,
    years^-1
N_1 = 6.023e+023/238; // Number of nuclei in 1g of
    Uranium
N = 6.023e+023*0.093/206; // Number of nuclei in
    0.093g of lead
t = log(1+N/N_1)/D_c; // Age of the mineral, years
printf("\n Age of the mineral : %6.4e years ", t)
// Result
//          Age of the mineral : 6.6261e+008 years
```

Scilab code Exa 3.9.2 Age of boat from its half life

```
// Scilab code Exa. 3.9.2 : To determine the age of
    boat whose half life is given : Page no. 145 :
    (2011)
t_h = 5760; // Half life of boat, years
D_c = 0.6931/t_h; // Decay constant of boat, years
    ^-1
N_1 = 16; // Number of atoms decay per min. per gram
     initially
```

```
N = 5; // Number of atoms decay per min per gram
    presently
t = log(N_1/N)*1/D_c; // Age of the boat, years
printf("\n Age of the boat : %d years ", t)
// Result
//          Age of the boat : 9666 years
```

Scilab code Exa 3.9.4 radioactive disintegration of Pu 239

```
// Scilab code Exa. 3.9.4 : To calculate the number
    of nuclei at t = 0, initial activity and age of
    Pu-239 which emit alpha particle : Page no. 145 :
     (2011)
t_h = 24000*365*24*3600; // Half life of Pu-239, s
D_c = 0.6931/t_h; // Decay constant of Pu-239, s^-1
N = 6.023e+023*10/239; // Number of nuclei at t = 0,
     nuclei
A_0 = D_c*N; // Initial activity, disintegrations/
    sec
A = 0.1; // Activity after time t, disintegrations/
    sec
t = log(A_0/A)*1/D_c; // Age of the Pu-239, years
printf("\nThe number of nuclei at t = 0,  = %4.2e
    nuclei \nInitial activity  = %4.2e
    disintegrations/s and \nAge  of Pu-239  = %4.2e
    years ", N, A_0, t)
// Result
//   The number of nuclei at t = 0,  = 2.52e+022
    nuclei
// Initial activity  = 2.31e+010 disintegrations/s
    and
//  Age of Pu-239  = 2.86e+013 years
```

Chapter 4

Nuclear Reactions

Scilab code Exa 4.3.1 Cross section of lithium

```
// Scilab code Exa4.3.1: To calculate the cross
    section of Li(3,7) : Page 179(2011)
t = 10^-5; // Thickness of Li(3,7), m
d = 500; // Density, Kg/m^3
N = 6.023e+026; // Number of  nuclei in 7-Kg of Li-7
M = 7 ; // Molar mass of Li
n = d*N*t/M; // Number of Li(3,7) nuclei/area
N_p = 10^8; // Number of neutron produced/s
N_0 = 10^13; // Number of incident particle striking
    /unit area of target
C_s = N_p/(N_0*n*10^(-028)); //  Cross section, b
printf("\n Cross section : %5.3f b", C_s)
 // Result
//  Cross section : 0.232 b
```

Scilab code Exa 4.3.2 Neutron absorption ratio

```
// Scilab code Exa4.3.2: To calculate the fraction
```

```
    of neutron absorbed by Cd sheet of given
    thickness : Page 180 (2011)
t = 0.2e-03; // Thickness of Cd sheet , m
d = 8.64e+03; // Density , Kg/m^3
N = 6.023e+026; // Number of  nuclei in 7-Kg of Li-7
M = 112 ; // Atomic mass of Cd-113, amu
C_s = 20000e-028; //  Cross section of neutron for
    Cd-113, m^2
n = 0.12*d*N/M; // Number of Cd atoms/volume, atoms/
    m^3
F_inc_absorb = [1-%e^(-n*C_s*t)]*100; // Fraction of
    neutron absorbed
printf("\n Fraction of neutron absorbed by Cd sheet
    : %4.2f percent",F_inc_absorb )
// Result
//   Fraction of neutron absorbed by Cd sheet :
    89.25 percent
```

Scilab code Exa 4.4.1 Nuclear reactions

```
// Scilab code Exa test : Checking the possibility
    of occurence of reactions : page no. 181 (2011)
// Declare three cells (for three reactions)
R1 = cell(4,4);
R2 = cell(5,4);
R3 = cell(4,4);
// Enter data for first cell (Reaction)
R1(1,1).entries = 'Al'; // Element
R1(1,2).entries = 13;   // Atomic number
R1(1,3).entries = 27;    // Mass number
R1(1,4).entries = 0;    // Lepton number
R1(2,1).entries = 'He';
R1(2,2).entries = 2;
R1(2,3).entries = 4;
R1(2,4).entries = 0;
```

```
R1(3,1).entries = 'Si';
R1(3,2).entries = 14;
R1(3,3).entries = 30;
R1(2,4).entries = 0;
R1(4,1).entries = 'n';
R1(4,2).entries = 0;
R1(4,3).entries = 1;
R1(2,4).entries = 0;
// Enter data for second cell (Reaction)
R2(1,1).entries = "U";
R2(1,2).entries = 92;
R2(1,3).entries = 235;
R2(1,4).entries = 0;
R2(2,1).entries = 'n';
R2(2,2).entries = 0;
R2(2,3).entries = 1;
R2(2,4).entries = 0;
R2(3,1).entries = 'Ba';
R2(3,2).entries = 56;
R2(3,3).entries = 143;
R2(3,4).entries = 0;
R2(4,1).entries = 'Kr';
R2(4,2).entries = 36;
R2(4,3).entries = 90;
R2(4,4).entries = 0;
R2(5,1).entries = '2n';
R2(5,2).entries = 0;
R2(5,3).entries = 1;
R1(5,4).entries = 0;
// Enter data for third cell (Reaction)
R3(1,1).entries = 'P';
R3(1,2).entries = 15;
R3(1,3).entries = 32;
R3(1,4).entries = 0;
R3(2,1).entries = 'S';
R3(2,2).entries = 16;
R3(2,3).entries = 32;
R3(2,4).entries = 0;
```

```
R3(3,1).entries = 'e';
R3(3,2).entries = -1;
R3(3,3).entries = 0;
R3(3,4).entries = 0;
R3(4,1).entries = 'v_e';
R3(4,2).entries = 0;
R3(4,3).entries = 0;
R3(4,4).entries = 0;
// Declare a function returning equality status of
   nucleon number
function f = check_nucleon(nr_sum,np_sum)
        if nr_sum == np_sum then
            f = 1;
        else
            f = 0;
        end
endfunction

// Declare a function returning equality status of
   proton number
function f = check_proton(pr_sum,pp_sum)
        if pr_sum == pp_sum then
            f = 1;
        else
            f = 0;
        end
endfunction

// Declare a function returning equality status of
   lepton number
function f = check_lepton(lr_sum,lp_sum)
        if lr_sum == lp_sum then
            f = 1;
        else
            f = 0;
        end
endfunction

```

```
// Reaction-I
printf("\n\n\nReaction-I:\n\n");
        pr_sum = R1(1,2).entries+R1(2,2).entries;
        pp_sum = R1(3,2).entries+R1(4,2).entries;
        nr_sum = R1(1,3).entries+R1(2,3).entries;
        np_sum = R1(3,3).entries+R1(4,3).entries;
        lr_sum = R1(1,4).entries+R1(2,4).entries;
        lp_sum = R1(3,4).entries+R1(4,4).entries;
        if (check_nucleon(nr_sum,np_sum)&check_proton(pr_sum,pp_sum)&check_lepton(lr_sum,lp_sum) == 1) then
            printf("The Reaction\n")
            printf("\t%s(%d) + %s(%d) --> %s(%d)+%s(%d)\nis possible", R1(1,1).entries, R1(1,3).entries, R1(2,1).entries, R1(2,3).entries, R1(3,1).entries, R1(3,3).entries, R1(4,1).entries, R1(4,3).entries);
        elseif (check_proton(pr_sum,pp_sum) == 0) then
            printf("The Reaction\n")
            printf("\t%s(%d) + %s(%d) --> %s(%d)+%s(%d)\nis impossible", R1(1,1).entries, R1(1,3).entries, R1(2,1).entries, R1(2,3).entries, R1(3,1).entries, R1(3,3).entries, R1(4,1).entries, R1(4,3).entries);
            R1(4,1).entries = 'H'; R1(4,3).entries = 1;
            printf("\nThe correct reaction is:\n")
            printf("\t%s(%d) + %s(%d) --> %s(%d)+%s(%d)\n", R1(1,1).entries, R1(1,3).entries, R1(2,1).entries, R1(2,3).entries, R1(3,1).entries, R1(3,3).entries, R1(4,1).entries, R1(4,3).entries);
        end
//  Display for reaction-II
```

```
printf("\n\n\nReaction-II:\n\n");
        pr_sum = R2(1,2).entries+R2(2,2).entries;
        pp_sum = R2(3,2).entries+R2(4,2).entries+R2(5,2).entries;
        nr_sum = R2(1,3).entries+R2(2,3).entries;
        np_sum = R2(3,3).entries+R2(4,3).entries+R2(5,3).entries;
        lr_sum = R2(1,4).entries+R2(2,4).entries;
        lp_sum = R2(3,4).entries+R2(4,4).entries+R2(5,4).entries;
        if (check_nucleon(nr_sum,np_sum)&check_proton(pr_sum,pp_sum)&check_lepton(lr_sum,lp_sum) == 1) then
          printf("The Reaction\n")
          printf("\t%s(%d) + %s(%d) --> %s(%d)+%s(%d)+%s(%d)\nis possible", R2(1,1).entries, R2(1,3).entries, R2(2,1).entries, R2(2,3).entries, R2(3,1).entries, R2(3,3).entries, R2(4,1).entries, R2(4,3).entries, R2(5,1).entries, R2(5,3).entries);
        elseif (check_nucleon(nr_sum,np_sum) == 0) then
          printf("The Reaction\n")
          printf("\t%s(%d) + %s(%d) --> %s(%d)+%s(%d)+%s(%d)\nis impossible", R2(1,1).entries, R2(1,3).entries, R2(2,1).entries, R2(2,3).entries, R2(3,1).entries, R2(3,3).entries, R2(4,1).entries, R2(4,3).entries, R2(5,1).entries, R2(5,3).entries);
           R2(5,1).entries = '3n';
          printf("\nThe correct reaction is:\n")
          printf("\t%s(%d) + %s(%d) --> %s(%d)+%s(%d)+%s(%d)\n", R2(1,1).entries, R2(1,3).entries, R2(2,1).entries, R2(2,3).entries, R2(3,1).entries, R2(3,3).entries, R2(4,1).entries, R2
```

```
                (4,3).entries, R2(5,1).entries, R2
                (5,3).entries);
        end
//      Reaction-III
            printf("\n\n\nReaction-III:\n\n");
        pr_sum = R3(1,2).entries+R3(2,2).entries;
        pp_sum = R3(3,2).entries+R3(4,2).entries;
        nr_sum = R3(1,3).entries+R3(2,3).entries;
        np_sum = R3(3,3).entries+R3(4,3).entries;
        lr_sum = R3(1,4).entries+R3(2,4).entries;
        lp_sum = R3(3,4).entries+R3(4,4).entries;
        if (check_nucleon(nr_sum,np_sum)&
           check_proton(pr_sum,pp_sum)&check_lepton(
           lr_sum,lp_sum) == 1) then
            printf("The Reaction\n")
            printf("\t%s(%d) + %s(%d) --> %s(%d)+%s(
                %d)\nis possible", R3(1,1).entries,
                R3(1,3).entries, R3(2,1).entries, R3
                (2,3).entries, R3(3,1).entries, R3
                (3,3).entries, R3(4,1).entries, R2
                (4,3).entries);
        elseif (check_lepton(nr_sum,np_sum) == 0)
           then
            printf("The Reaction\n")
            printf("\t%s(%d) + %s(%d) --> %s(%d)+%s(
                %d)\nis impossible", R3(1,1).entries,
                 R3(1,3).entries, R3(2,1).entries, R3
                (2,3).entries, R3(3,1).entries, R3
                (3,3).entries, R3(4,1).entries, R3
                (4,3).entries);
             R3(4,1).entries = 'v_e_a'
            printf("\nThe correct reaction is:\n")
            printf("\t%s(%d) + %s(%d) --> %s(%d)+%s(
                %d)\n", R3(1,1).entries, R3(1,3).
                entries, R3(2,1).entries, R3(2,3).
                entries, R3(3,1).entries, R3(3,3).
                entries, R3(4,1).entries, R3(4,3).
                entries);
```

```
        end

// Reaction-I:

// The Reaction
//  Al(27) + He(4) --> Si(30)+n(1)
// is impossible
// The correct reaction is:
//  Al(27) + He(4) --> Si(30)+H(1)



// Reaction-II:

// The Reaction
//  U(235) + n(1) --> Ba(143)+Kr(90)+2n(1)
// is impossible
// The correct reaction is:
//  U(235) + n(1) --> Ba(143)+Kr(90)+3n(1)



// Reaction-III:

// The Reaction
//  P(32) + S(32) --> e(0)+v_e(0)
// is impossible
// The correct reaction is:
//  P(32) + S(32) --> e(0)+v_e_a(0)
```

Scilab code Exa 4.5.1 Q value for reaction

```
// Scilab code Exa4.5.1: To calculate Q-value for
   given reaction  : Page 182 (2011)
M_n = 1.00866501; // Mass of neutron, amu
```

```
M_Hp = 2.014102; // Mass of proton , amu
M_Hd = 3.016049; // Mass of deutron , amu
M_He = 4.002603; // Mass of alpha particle , amu
Q = [M_Hp+M_Hd-M_He-M_n]*931.49; // Q-value , MeV
printf("\nThe Q-value for the reaction : %4.1f MeV",
    Q)
// Result
//     The Q-value for the reaction : 17.6 MeV
```

Scilab code Exa 4.5.2 Energy emitted in nuclear reaction

```
// Scilab code Exa4.5.2: To calculate Q-value for
   the reaction : Page 183 (2011)
M_Cf = 252.081621; // Mass of califronium , amu
M_Cm = 248.072343; // Mass of curium , amu
M_He = 4.002603; // Mass of alpha particle , amu
Q = [M_Cf-M_Cm-M_He]*931.49; // Q-value , MeV
printf("\nThe Q-value for the reaction : %4.2f MeV",
    Q)
// Result
// The Q-value for the reaction : 6.22 MeV
```

Scilab code Exa 4.5.3 Threshold energy and Q value for nuclear reaction

```
// Scilab code Exa4.5.3: To calculate Q-value and
   threshold energy for the given reaction : Page
   183 (2011)
// Pb_208(Fe_56 , Fe_54)Pb_210
M_Pb_208 = 207.976641; // Mass of Pb-208, amu
M_Fe_56 = 55.934939; // Mass of Fe-56, amu
M_Pb_210 = 209.984178; // Mass of Pb-210, amu
M_Fe_54 = 53.939612; // Mass of Fe-54, amu
```

```
Q = [M_Pb_208+M_Fe_56-M_Pb_210-M_Fe_54]*931.49; // Q-value, MeV
E_th = -Q*(M_Fe_56+M_Pb_208)/M_Pb_208; // Threshold energy, MeV
printf("\nThe Q-value for the reaction    =  %5.2f MeV \n Threshold energy    = %5.2f MeV ", Q,E_th)
// Result
//  The Q-value for the reaction    =   -11.37 MeV
//  Threshold energy    = 14.43 MeV
```

Scilab code Exa 4.5.4 Mass of neutron from nuclear reaction

```
// Scilab code Exa4.5.4: To calculate  the mass of neutron for  given reaction : P.No. 184 (2011)
// H(1,1)+n(0,1)  = H(1,2)+G is the reaction
M_H_2 = 2.014735; // Mass of H-2, amu
M_H_1 = 1.008142 ; // Mass of H-1, amu
E_g = 2.230; // Energy of gamma rays, MeV
M_n_1 = [(M_H_2*931.47+E_g)-(M_H_1*931.47)]/931.47; //Mass of neutron, amu
printf("\nThe  mass of the neutron   : %8.6f MeV ", M_n_1)
// Result
//      The mass of the neutron   : 1.008987 MeV
```

Scilab code Exa 4.5.5 Q value sign for nuclear reaction

```
// Scilab code Exa 4.5.5 : Checking given reaction condition : page no. 184 (2011)
// Li-6 + n-1      > He-4 + H-3 is the given reaction
M_Li = 6.0151234; // Atomic mass of Li, amu
M_n = 1.0086654; //  Atomic mass of neutron, amu
M_He = 4.0026034; //  Atomic mass of He, amu
```

```
M_H = 3.0160294; //  Atomic mass of H, amu
r_sum = M_Li+M_n; //  Sum of reactant , amu
p_sum = M_He+M_H; //  Sum of product , amu
// Declare a function returning equality status of nucleon number
function Q = check_Qvalue(r_sum,p_sum)
        if r_sum >= p_sum then
            Q = 1;
        else
            Q = 0;
        end
endfunction

// Reaction
        if (check_Qvalue(r_sum,p_sum)  == 1) then
            printf("\n Reaction : \n\n\t Li(6)+n(1) ----> He(4)+H(3)")
            printf("\n\n\t\tThis reaction is exoergic")
        elseif (check_Qvalue(r_sum,p_sum) == 0) then
            printf("\n Reaction : \n\n\t Li(6)+n(1) ----> He(4)+H(3)")
            printf("\n\n\t\tThis reaction is endoergic")
           end
// Reaction :

//  Li(6)+n(1) ----> He(4)+H(3)

    //  This reaction is exoergic
```

Scilab code Exa 4.5.6 Spontaneity of Q value for nulclear reaciton

```
// Scilab code Exa 4.5.5 : Checking whether the reaction  is spontaneous or exoergic : page no.
```

```
    185 (2011)
// Cf-252    > Zr-98 +Ce-145 + 9*n-1 is the given
    reaction
M_Cf = 252.081621; // Atomic mass of Cf, amu
M_Zr = 97.912735; //  Atomic mass of Zr, amu
M_Ce = 144.917230; //  Atomic mass of Ce, amu
M_n = 3.0160294; //  Atomic mass of neutron, amu
r_sum = M_Cf+M_Zr; //  Sum of reactant, amu
p_sum = M_Ce+M_n; //  Sum of product, amu
// Declare the function which check the Q-value
function Q = check_Qvalue(r_sum,p_sum)
        if r_sum >= p_sum then
            Q = 1;
        else
            Q = 0;
        end
endfunction

// Reaction
        if (check_Qvalue(r_sum,p_sum)  == 1) then
            printf("\n Reaction : \n\n\t Cf(256)
              ----> Zr(98)+Ce(145)+9*n(1)")
            printf("\n\n\t\tThis reaction is
              spontaneous")
        elseif (check_Qvalue(r_sum,p_sum) == 0) then
            printf("\n Reaction : \n\n\t Cf(256)
              ----> Zr(98)+Ce(145)+9*n(1)")
            printf("\n\n\t\tThis reaction is not
              spontaneous")
           end
 // Reaction :
   //  Cf(256) ----> Zr(98)+Ce(145)+9*n(1)

    //  This reaction is spontaneous
```

Scilab code Exa 4.5.7 Nuclear reaction Q value

```
// Scilab code Exa4.5.7: To calculate Q-value  for given reaction : Page 185 (2011)
// O(8,16)      > N(7,15)+ H(1,1) is the given reaction
M_N_15 = 15.000108; // Mass of N-15, amu
M_O_16 = 16; // Mass of O-16, amu
M_H_1 = 1.007825; // Mass of H-1, amu
Q = [M_O_16-M_N_15-M_H_1]*931.49; // Q-value, MeV
printf("\nThe Q-value for the reaction  : %3.1f MeV ", Q)
// Result
//The Q-value for the reaction  : -7.4 MeV
```

Scilab code Exa 4.5.8 Threshold energy for given reaction

```
// Scilab code Exa4.5.8: To determine the threshold energy for given reaction : P.no. 185 (2011)
// Na(11,23)+ n     >   F(9,20)+ He(2,4) is the reaction
M_Na_23 = 22.99097; // Mass of Na-23, amu
M_n_1 =1.00866 ; // Mass of n-1, amu
Q = -5.4; // Q-value, MeV
E_th = -Q*(M_Na_23+M_n_1)/M_Na_23; // Threshold energy, MeV
printf("\nThe threshold energy for the reaction  : %4.2f MeV ", E_th)
// Result
//   The threshold energy for the reaction  : 5.64 MeV
```

Scilab code Exa 4.5.9 Q value of nuclear reaction

```
// Scilab code Exa4.5.9: To calculate Q-value for the reaction : Page 187 (2011)
// He(2,4)+ N(7,14) =  O(8,17)+ H(1,1) is the given reaction
M_N_14 = 14.00755; // Mass of N-14, amu
M_He_4 = 4.00388; // Mass of He-4, amu
M_O_17 = 17.00452; // Mass of O-17, amu
M_H_1 = 1.00815; // Mass of H-1, amu
Q = [M_N_14+M_He_4-M_O_17-M_H_1]*931.49; // Q-value , MeV
printf("\nThe Q-value for the reaction  : %4.2f MeV", Q)
// Result
//The Q-value for the reaction  : -1.16 MeV
```

Scilab code Exa 4.5.10 Energy of gamma rays

```
// Scilab code Exa4.5.10: To determine the energy of gamma ray for reaction :: P.no. 186 (2011)
//    H(1,2)+G  =  H(1,1)+ n(0,1) is the given reaction
M_H_2 = 2.014735; // Mass of H-2, amu
M_H_1 = 1.008142 ; // Mass of H-1, amu
M_n_1 = 1.008987; // Mass of M_n_1 , amu
Q = -5.4; // Q-value , MeV
E_g = (M_H_1*931.47+M_n_1*931.47)-(M_H_2*931.47); // Energy of the gama rays , MeV
printf("\nThe  energy of the gama rays  : %6.4f MeV", E_g)
// Result
//   The  energy of the gama rays  : 2.2299 MeV
```

Scilab code Exa 4.7.1 Energy and power released during fission of U 235

```
// Scilab code Exa4.7.1: To calculate the energy
    and power released during fission of U-235 : Page
     189 (2011)
m = 0.001; // Mass of U-235 lost during fission , Kg
c = 3e+08; // Velocity of light , m/s
E = m*c^2; // Energy released during fission , J
E_t = E/(4e+09*1000); // Energy requires TNT, Kt
printf("\n Energy released during fission    = %1.0e
     J  \n Destructive power of bomb    =  %4.1f Kt
    of TNT", E, E_t)
// Result
  //      Energy released during fission     = 9e+013
      J
//  Destructive power of bomb    =  22.5 Kt of TNT
```

Scilab code Exa 4.7.2 Fission rate induced in the uranium foil by neutron

```
// Scilab code Exa4.7.2: To determine the fission
    rate induced in the foil by neutron  : Page 190
    (2011)
t = 0.15; // Thickness of the foil , Kg
N = 6.023e+026; // Number of nuclei in 1Kg of U-235,
     nuclei
N_1 = N/235*t; // Number of nuclei in 0.15Kg of U
    -235, nuclei
A = 2e-026; // Area present in each nucleus , m^2
I = 10^6; // Intensity ,s^-1
F_r = N_1*A; // Rate of fissions induced in the foil
     by the neutrons , s^-1
printf("\n Rate of fissions induced in the foil by
    the neutrons: %5.3e per sec", F_r)
// Result
//       Rate of fissions induced in the foil by the
    neutrons: 7.689e-003 per sec
```

Scilab code Exa 4.7.3 Power in fission process

```
// Scilab code Exa4.7.3: To determine the fission power produced by one microgram of Fm-256 : Page 190 (2011)
N = 6.023e+023/256*10^-6; // Number of nuclei in 1ug of Fm-256
t_h = 158*60; // Half life of Fm-256, s
D_c = log(2)/t_h; // Decay constant, s^-1
F_r = N*D_c; // Fission rate, fissions/s
E = 220*1.6e-013; // Energy released during fission of one nucleus, J
P = E*F_r; // Power released in fission of 1 microgram of Fm-256, W
printf("\n Power released in fission of 1 microgram of Fm-256 = %d W", P)
// Result
//          Power released in fission of 1 microgram of Fm-256 = 6 W
```

Scilab code Exa 4.7.4 Power released in fission

```
// Scilab code Exa4.7.4: To determine the power produced by 100 milligram of Cf-252 : Page 191 (2011)
N = 6.023e+023/252*0.1; // Number of nuclei in 100mg of Cf-252
t_h = 2.62*365*24*3600; // Half life of Cf-252, s
D_c = log(2)/t_h; // Decay constant, s^-1
F_r = N*D_c; // Fission rate, fissions/s
E = 210*1.6e-013; // Energy released during fission of one nucleus, J
```

```
P = E*F_r; // Power released in fission of 100
    milligram of Cf-252, W
printf("\n Power released in fission of 100
    milligram of Cf-252: %4.1f W", P)
// Result
//          Power released in fission of 100
    milligram of Cf-252: 67.4 W
```

Scilab code Exa 4.7.5 Fission counts and mass reduction of fissile material

```
// Scilab code Exa4.7.5: To determine the number of
    nuclear fission and decrease in mass during
    explosion at hiroshima : Page 191 (2011)
E = 200*1.6e-013; // Energy released during fission
    of one nucleus, J
E_t = 20000*4.18e+09; // Energy released in
    detonation of 20000 tons of TNT, J
N_f = E_t/E; // Number of fission occured during
    eplosion, fissions
c = 3e+08; // Velocity of light, m/s
m = E_t/(c)^2*10^6; // Decrease in mass during
    explosion, mg
m_r = round(m)
printf("\n Number of fissions occured during
    explosion    = %4.2e fissions \n Decrease in mass
     during explosion    = %d mg ", N_f, m_r)
// Result
//          Number of fissions occured during
    explosion    = 2.61e+024 fissions
//             Decrease in mass during explosion    =
    929 mg
```

Scilab code Exa 4.8.1 Energy liberated in fusion reaction

```
// Scilab code Exa4.8.1: To calculate the energy liberated during fusion reaction: Page 194 (2011)
//  5*H(1,2)= He(2,3)+He(2,4)+H(1,2)+2*n(0,1)+25MeV is the given reaction
N = 6.023e+026/2*10; // Number of atoms in 10Kg of H-2, atoms
E = 25/5*1.6e-013; // Energy liberate during fusion of 1 atom of H-2, J
E_l = E*N; // Energy liberate during fusion of 10 Kg of H-2, J
printf("\n Energy liberated during fusion of 10 Kg of H-2 = %4.2e J", E_l)
// Result
//   Energy liberated during fusion of 10 Kg  of H-2 = 2.41e+015 J
```

Scilab code Exa 4.8.2 Energy produced by helium carbon fusion

```
// Scilab code Exa4.8.2: To calculate the energy produced by the fusion reaction He(2,4)+C(6,12)=O(8,16) : Page 194 (2011)
M_r = 16.002603; // Mass of the reactant, amu
M_p = 15.994915; // Mass of reactant, amu
M_d = 7.688e-03; // Difference in masses, amu
E_p = M_d*931.49; // Energy produced, MeV
printf("\n Energy produced by the fusion reaction : %4.2f MeV", E_p)
// Result
// Energy produced by the fusion reaction :7.16 MeV
```

Scilab code Exa 4.8.3 Energy released and temperature required for fusion of gases

```
// Scilab code Exa4.8.3: To calculate the energy released and temperature required for fusion of given gases : Page 194 (2011)
// Firstly calculate for B-10
Z_B = 5; // Atomic number of B-10
r_B = 5.17; //  Seperation of two nuclei , fm
K = 1.38e-023; // Boltzmann's constant
F = 1/137; // Fine structure constant
E = 197.5*1.6e-013; // Energy , J
V_c_B = F*Z_B^2*E/r_B; // Coulomb barrier for B-10, J
T_B = 2/3*V_c_B/K; // Temperature required to overcome the barrier for B-10, K
// Now calculate for Mg-24
Z_Mg = 12; // Atomic number of Mg-24
r_Mg = 6.92; //  Seperation of two nuclei , fm
K = 1.38e-023; // Boltzmann's constant
F = 1/137; // Fine structure constant
E = 197.5*1.6e-013; // Energy , J
V_c_Mg = F*Z_Mg^2*E/r_Mg; // Coulomb barrier for Mg-24, J
T_Mg = 2/3*V_c_Mg/K; // Temperature required to overcome the barrier for Mg-24, K
printf("\nFor B-10 \n Energy released   = %4.2e J \n Temperature required   = %4.1e K   \nFor Mg-24 \n Energy released    = %4.2e J \n Temperature required  =  %4.2e K", V_c_B,T_B, V_c_Mg,  T_Mg)
// Result
 //      For B-10
 // Energy released   = 1.12e-012 J
 // Temperature required    = 5.4e+010 K
// For Mg-24
//  Energy released    = 4.80e-012 J
//  Temperature required  =  2.32e+011 K
```

Scilab code Exa 4.8.4 Life time of sun

```
// Scilab code Exa4.8.4: To calculate the life time
    of sun for given reaction : Page 196 (2011)
// 4*H(1,1)= He(2,4)+2*e(1,0)+2*v+G is the reaction
E_r = 3.9e+026; // Energy releasd in 1s, J
N = 1.2e+057; // Number of hydrogen atoms in the sun
    , atoms
M_d = 0.027599;// Mass difference, amu
E = M_d*931.47; // In terms of energy, MeV
E_t = N/4*E*1.6e-013; // Total energy available in
    the sun, J
t = E_t/(E_r*365*24*3600*10^9); // Life time of the
    sun, billion years
printf("\n Life time of the sun : %5.1f billion
    years", t)
// Result
//         Life time of the sun : 100.3 billion years
```

Scilab code Exa 4.8.5 Particle identification in the nuclear reaction

```
// Scilab code Exa 4.8.5 : Identifying the nucleus
    and energy released in the given reaction : page
    no. 197 (2011)
// Declare three cells (for three reactions)
R = cell(4,3);
// Enter data for first cell (Reaction)
R(1,1).entries = 'H'; // Element
R(1,2).entries = 1;    // Atomic number
R(1,3).entries = 2;     // Mass number
R(2,1).entries = 'H';
R(2,2).entries = 1;
```

```
R(2,3).entries = 3;
R(3,1).entries = 'n'
R(3,2).entries = 0;
R(3,3).entries = 1;
R(4,1).entries = 'He'
R(4,2).entries = 2;
R(4,3).entries = 3;
// Declare a function returning equality status of nucleon number

        p_sum = R(1,2).entries+R(2,2).entries;
                if (p_sum == 2) then

            printf("\n The particle is : %s(%d,%d) ",R(4,1).entries,R(4,2).entries,R(4,3).entries )
             end
//  Calculate the energy released
m_n = 1.008665; // Mass of neutron , amu
m_d = 2.014102; // Mass of deutron , amu
m_He = 3.0160293; // Mass of He-3, amu
E = [2*m_d-(m_n+m_He)]*931.47; // Energy released in this reaction , MeV
printf("\n The energy released in this reaction : %4.2f MeV", E )
// Result
//          The particle is : He(2,3)
 //        The energy released in this reaction : 3.27 MeV
```

Scilab code Exa 4.8.6 Mass defect and q value for fusion reaction

```
// Scilab code Exa4.8.6: To calculate the mass defect and Q-value for the fusion reactions : Page 197 (2011)
```

```
//  Reaction-1 = H(1,2)+H(1,2)= He(2,3)+n(0,1)
m_p = 1.007825; // Mass of proton, amu
m_n = 1.008665; // Mass of neutron, amu
m_H = 2.014102; // Mass of H(1,2), amu
m_He = 3.016029; // Mass of He(2,3), amu
m_d_1 = 2*m_H-m_He-m_n; // Mass defect for reaction first, amu
Q_1 = m_d_1*931.47; // Q-value for reaction first, MeV
// Reaction-2 = H(1,2)+H(1,2)= H(1,3)+p(1,1)
m_p = 1.007825; // Mass of proton, amu
m_n = 1.008665; // Mass of neutron, amu
m_H = 2.014102; // Mass of H(1,2), amu
m_H_3 = 3.016049; // Mass of H(1,3), amu
m_d_2 = 2*m_H-m_H_3-m_p; // Mass defect for reaction second, amu
Q_2 = m_d_2*931.47; // Q-value for reaction second, MeV
printf("\nFor first reaction \n Mass defect    = %7.5f amu \n Q-value    =   %7.5f amu    \nFor second reaction \n Mass defect    =  %7.5f MeV \n Q-value     = %4.2f MeV ", m_d_1,Q_1, m_d_2,  Q_2)
// Result
//   For first reaction
//  Mass defect     = 0.00351 amu
//  Q-value     =   3.26946 amu
// For second reaction
//  Mass defect    =  0.00433 MeV
//  Q-value     = 4.03 MeV
```

Chapter 5

Interaction of Radiations with Matter

Scilab code Exa 5.2.1 Energy lost during collision

```
// Scilab code Exa5.2.1: To calculate the energy and
    no. of collision required to stop collision : P.
   no. 223 (2011)
m = 511; // Mass of electron, KeV
M = 938*10^3; // Mass of incident charged particle,
   KeV
E = 10*10^3; // Energy of proton, KeV
E_l = 4*m*E/M; // Energy lost during collison, KeV
n = E/E_l; // Number of collisions,
N = round(n)
printf("\n The energy lost during collision    = %5.2f
    KeV \n Number of collision required    = %d
   collisions",E_l, N )
// Result
//      The energy lost during collision   = 21.79 KeV
//  Number of collision required    = 459 collisions
```

Scilab code Exa 5.5.1 Half value thickness of aluminium

```
// Scilab code Exa5.5.1: To calculate the half value thickness of Al for given radiation : P.no. 225 (2011)
x = 0.2; // Thickness of Al material , m
I_r = 3/100; // Intensity ratios ,
x_h = log(2)*x/log(1/I_r); // Half value thickness of Al, m
printf("\n  Half value thickness of Al : %6.4f m",x_h )
// Result
//       Half value thickness of Al : 0.0395 m
```

Scilab code Exa 5.5.2 Thickness of lead

```
// Scilab code Exa5.5.2: To calculate the thickness of Pb: P.no. 226 (2011)
u = 0.75; // Absorption coefficient , cm^-1
I_r = 1/100; // Intensity ratios ,
x = log(1/I_r)*u; // Thckness of Pb, cm
printf("\n Thickness of Pb : %5.3f cm",x )
// Result
//       Thickness of Pb : 6.140 m
```

Scilab code Exa 5.5.3 Percentage loss of intensity of gamma rays

```
// Scilab code Exa5.5.3: To calculate the percentage loss of intensity of gamma rays  : P.no. 226 (2011)
x_h = 5; // Half thickness of an absorber, mm
u = log(2)/x_h; // Absorption coefficient , mm^-1
x = 20; // Thickness of an absorber , mm
```

```
I_r = %e^(-u*x); // Intensity ratios,
P_loss = I_r*100; // Percentage loss in intensity, percent
printf("\n Percentage loss in intensity : %4.2f percent",P_loss )
// Result
//         Percentage loss in intensity : 6.25 percent
```

Scilab code Exa 5.6.1 Velocity of ejected photoelectron

```
// Scilab code Exa5.6.1: To calculate the velocity of ejected photoelectron : P.no. 230 (2011)
 C = 3e+08; // Speed of light, m/s
 h = 6.626e-034; // Planck's constant, Js
 lambda = 2500e-010; //   wavelength of light, m
e = 1.602e-019; // Charge of electron, C
 w = 1.9; // Work function, J
m = 9.1e-031; // Mass of the electron, kg
E_c = h*C/(lambda*e); // Calculated energy, J
E_e = E_c-w; // Energy of photoelectron, J
v = sqrt((2*E_e*e)/m); // Velocity of photoelectron, m/s
printf("\nThe velocity of photoelectron : %4.2e m/s", v )
// Result
//       The velocity of photoelectron : 1.04e+006 m/s
```

Scilab code Exa 5.6.2 Rate of photoelectron emission

```
// Scilab code Exa5.6.2: To calculate the kinetic energy of photoelectron and rate at which photoelectron emitted : P.no. 231 (2011)
 C = 3e+08; // Speed of light, m/s
 h = 6.626e-034; // Planck's constant, Js
 lambda = 250e-09; // Wavelength of light, m
 w = 2.30; // Work function, eV
A = 2e-04; // Area of the surface, m^2
I = 2; // Intensity of light, W/m^2
e = 1.6e-019; // Charge of the electron, C
E_p = h*C/(lambda*e); // Energy of photoelectron, eV
E_max = E_p-w; // Maximum kinetic energy of photoelectron, eV
n_p = I*A/(E_p*e); // Number of photons reaching the surface per second, photons/s
R_p = 0.2/100*n_p; // Rate at which photoelectrons are emitted, photoelectrons/s
printf("\n The maximum kinetic energy  = %4.2f eV \n The rate at which photoelectrons are emitted = %4.2e photoelectrons/s ", E_max, R_p)
// Result
//        The maximum kinetic energy  = 2.67 eV
 // The rate at which photoelectrons are emitted = 1.01e+012 photoelectrons/s
```

Scilab code Exa 5.6.3 Kinetic energy of photoelectron

```
// Scilab code Exa5.6.3: To calculate the wavelength of light whose kinetic energy is given : P. No. 232 (2011)
 C = 3e+08; // Speed of light, m/s
 h = 6.626e-034; // Planck's constant, Js
 T_lambda = 190e-09; // Threhold wavelength of light, m
e = 1.6e-019; // Charge of the electron, C
```

```
E_max = 1.1; // Maximum kinetic energy of
    photoelectron , eV
w = h*C/(T_lambda*e); // Work function , eV
E_t = E_max+w; // threshold energy , eV
lambda = h*C/(E_t*e); // Wavelength of light used , m
printf("\nThe wavelength of light used  : %5.3e m",
    lambda)
// Result
//     The wavelength of light used  : 1.626e-007 m
```

Scilab code Exa 5.7.1 Compton shift

```
// Scilab code Exa5.7.1: To calculate the Compton
    shift : P.no. 233 (2011)
h = 6.62e-034; // Value of Planck's constant , J
m_e = 9.11e-031; // Mass of the electron ,Kg
c = 3e+08; // Velocity of light , m/s
A = 65; // Angle between scattered radiation and
    incident radiation , degree
C_s =  h/(m_e*c)*(1-cosd(A)); // Compton shift , m
printf("\nCompton shift  : %4.2e m",C_s )
// Result
//      Compton shift  : 1.40e-012 m
```

Scilab code Exa 5.7.2 Wavelength of the scattered gamma rays

```
// Scilab code Exa5.7.2: To calculate the
    wavelength of the scattered gamma rays: P.no. 233
     (2011)
h = 6.626e-034; // Value of Planck's constant , J
m_e = 9.11e-031; // Mass of the electron ,Kg
c = 3e-04; // Velocity of light , m/s
```

```
A = 135; // Angle between scattered radiation and incident radiation, degree
W_i = 1.87; // Wavelength of incident radiation, pm
W_s = W_i + [h*(1-cosd(A))]/(m_e*c); // Wavelength of scattered radiation, pm
printf("\nWavelength of scattered radiation   : %4.2f pm",W_s )
// Result
//        Wavelength of scattered radiation   : 6.01 pm
```

Scilab code Exa 5.7.3 Wavelength of the incident beam of X rays

```
// Scilab code Exa5.7.3: To calculate the wavelength of the incident beam of X-rays : P.no. 234 (2011)
h = 6.626e-034; // Value of Planck's constant, J
m_e = 9.11e-031; // Mass of the electron,Kg
c = 3e-04; // Velocity of light, pm/s
A = 90; // Angle between scattered radiation and incident radiation, degree
W_s = 3.8; // Wavelength of scattered radiation, pm
W_i = [W_s - h/(m_e*c)*(1-cosd(A))]; // Wavelength of incident beam of Xrays, pm
printf("\nWavelength of incident beam of X-rays : %4.2f pm", W_i )
// Result
//        Wavelength of incident beam of X-rays : 1.38 pm
```

Scilab code Exa 5.7.4 Frequency of the scattered photon

```
// Scilab code Exa 5.7.4 : To calculate the frequency of the scattered photon  Page.no. 234 (2011)
h = 6.626e-034; // Value of Planck's constant, J
m_e = 9.11e-031; // Mass of the electron,Kg
c = 3e+08; // Velocity of light, pm/s
A = 60; // Angle between scattered radiation and incident radiation, degree
v_0 = 3.2e+019; // Frequency of the incident photon, Hz
V = 1/v_0 + h/(m_e*c^2)*(1-cosd(A));
v =(1/V); // Frequency of the scattered photon, Hz
printf("\n Frequency of the scattered photon: %4.2e Hz", v )
// Result
//       Frequency of the scattered photon: 2.83e+019 Hz
```

Scilab code Exa 5.7.5 Energy of scattered photon and recoil electron

```
// Scilab code Exa 5.7.5 : To calculate the  energy of the scattered photon and the energy of recoil electron : P.no. 235 (2011)
h = 6.626e-034; // Value of Planck's constant, J
m_e = 9.11e-031; // Mass of the electron,Kg
c = 3e+08; // Velocity of light, pm/s
A = 180; // Angle between scattered radiation and incident radiation, degree
E_i = 1836; // Energy of the incident electron, KeV
E = 1/E_i + 1/511*(1-cosd(A));
E_s = round(1/E); // Energy of the sscattered photon, KeV
E_r = E_i-E_s; // Energy of the recoil electron, KeV
printf("\n Energy of the scattered photon    = %d KeV  \n Energy of the recoil electron     = %d KeV
```

```
", E_s, E_r )
// Result
//        Energy of the scattered photon    = 224 KeV
 //       Energy of the recoil electron    = 1612 KeV
```

Scilab code Exa 5.7.6 Scattering angle of X rays

```
// Scilab code Exa 5.7.6 : To calculate the scattering angle of X-rays  Page.no. 235 (2011)
E_s = 180; // Energy of the scattered X-rays, KeV
E_i = 200; // Energy of the incident X-rays, KeV
a = acosd(1-[{1/E_s-1/E_i}*511]); //
A = round(a); // Scattering angle of X-rays, degree
printf("\n Scattering angle of X-rays: %d degree", A )
// Result
//          Scattering angle of X-rays: 44 degree
```

Scilab code Exa 5.8.1 Kinetic energy of electron and positron

```
// Scilab code Exa5.8.1: To calculate the kinetic energy of electron and positron :P.no. 236 (2011)
 M_e = 0.511; // Rest mass of electron, MeV
 M_p = 0.511; // Rest mass of positron, MeV
 E_c = M_e+M_p; // Energy consumed, Mev
 E_g = 5.0; // Given energy, MeV
 E_l = E_g-E_c; // Energy left, Mev
 E_k = E_l/2; // Kinetic energy of electron and positron, MeV
 printf("\n The kinetic energy of electron and positron  : %5.3f Mev", E_k)
```

```
// Result
//          The kinetic energy of electron and
   positron  : 1.989 Mev
```

Chapter 6

Particle Accelerators

Scilab code Exa 6.2.1 Kinetic energy of protons

```
// Scilab code Exa6.2.1 : To calculate the kinetic energy of protons : Page 264 (2011)
q = 1;  //   Number of proton,
V = 800;    //  Voltage applied to the dome, kV
E = q*V; // The kinetic energy of proton,keV
printf("\nThe kinetic energy of proton : %d keV", E);
// Result
// The kinetic energy of proton : 800 keV
```

Scilab code Exa 6.3.1 Protons in Van de Graff accelerator

```
// Scilab code Exa6.3.1 : To calculate the kinetic energy of protons in Van de Graff accelerator: Page 265 (2011)
q = 1;  //   Number of proton,
V = 7;    //  Voltage applied to the dome, MV
E = q*V; // The kinetic energy of proton,MeV
```

```
printf(”\nThe kinetic energy of proton : %d MeV”, E)
;
// Result
//      The kinetic energy of proton : 7 MeV
```

Scilab code Exa 6.3.2 Reactions at different particle energies

```
// Scilab code Exa6.3.2 : To calculate the kinetic energy of protons and no. of  possibile reactions : Page 265 (2011)
V = 5; // Voltage of accelerator , MV
// Declare three cells (for three reactions): Page no. : 133(2011)
 R1 = cell(3,2)
 R2 = cell(10,2)
// Enter data for first cell (Reaction)
R1(1,1).entries = ”p”;
R1(1,2).entries = 1;
R1(2,1).entries = 'd';
R1(2,2).entries = 1;
R1(3,1).entries = 'He';
R1(3,2).entries = 2;
E_p =  (R1(1,2).entries)*V
E_d =  (R1(2,2).entries)*V
E_He =  (R1(3,2).entries)*V
 // Enter data for second cell (Reaction)
 R2(1,1).entries = ”p”
 R2(1,2).entries = 1
 R2(2,1).entries = ”N”
 R2(2,2).entries = 14
 R2(3,1).entries = ”O”
 R2(3,2).entries = 15
 R2(4,1).entries = ”y”
 R2(4,2).entries = 0
 R2(5,1).entries = ”d”
```

```
R2(5,2).entries = 1
R2(6,1).entries = "n"
R2(6,2).entries = 0
R2(7,1).entries = "He"
R2(7,2).entries = 3
R2(8,1).entries = "C"
R2(8,2).entries = 13
R2(9,1).entries = "He"
R2(9,2).entries = 4
R2(10,1).entries = "C"
R2(10,2).entries = 12
printf("\nProtons energy   = -%d MeV \n Deuterons energy   = -%d MeV  \n Double charged He-3  = -%d MeV", E_p, E_d, E_He)
printf("\n Possible reaction at these energies are")
printf("\n %s + %s(%d) --->   %s(%d)+ %s", R2(1,1).entries,R2(2,1).entries,R2(2,2).entries,R2(3,1).entries,R2(3,2).entries,R2(4,1).entries)
printf("\n %s + %s(%d) --->   %s(%d) + %s ", R2(5,1).entries,R2(2,1).entries,R2(2,2).entries,R2(3,1).entries,R2(3,2).entries,R2(6,1).entries)
printf("\n %s(%d) +%s(%d)  --->   %s(%d)+ %s", R2(7,1).entries,R2(7,2).entries,R2(8,1).entries,R2(8,2).entries,R2(3,1).entries,R2(3,2).entries,R2(6,1).entries)
printf("\n %s(%d) + %s(%d)  --->  %s(%d) +%s", R2(9,1).entries,R2(9,2).entries,R2(10,1).entries,R2(10,2).entries,R2(3,1).entries,R2(3,2).entries,R2(6,1).entries)

// Result
// Protons energy   = -5 MeV
// Deuterons energy   = -5 MeV
// Double charged He-3  = -10 MeV
// Possible reaction at these energies are
// p + N(14) --->   O(15)+ y
// d + N(14) --->   O(15) + n
```

```
// He(3) +C(13)  --->   O(15)+ n
// He(4) + C(12)  --->  O(15) +n
```

Scilab code Exa 6.4.1 Protons passing through the carbon stripper foil

```
// Scilab code Exa6.4.1 : To calculate the kinetic
   energy of protons passing through the carbon
   stripper foil : Page 266 (2011)
q = 2;  //   Number of proton,
V = 15;    //  Voltage applied to the dome, MV
E = q*V; // The kinetic energy of proton,MeV
printf("\nThe kinetic energy of proton : %d MeV", E)
   ;
// Result
//    The kinetic energy of proton : 30 MeV
```

Scilab code Exa 6.5.1 Electron at relativistic energy

```
// Scilab code Exa6.5.1 : To calculate the
   difference between the electron's speed and speed
    of light. Page 265 (2011)
v = 2.999999997e+08;  //  Velocity of the electron,
  m/s
c = 3e+08;  // Velocity of light,m/s
D = c-v; // difference between electron's speed and
   speed of light,m/s
printf("\nThe difference between electron speed and
   speed of light : %3.1f m/s", D);
// Result
//  The difference between electron speed and speed
   of light : 0.3 m/s
```

Scilab code Exa 6.5.2 Protons accelerating through drift tubes

```
// Scilab code Exa6.5.2 : To calculate the length of
    the first and last drift tubes which accelerate
    the protons whose frequency and energies are
    given. Page 268 (2011)
f = 200e+06; //  Frequency of applied the voltage,
    Hz
V_0 = 750e+03;  // Applied potential difference, V
q = 1.6e-019; // Charge of proton, C
m = 1.67e-027; // Mass of proton, Kg
n_1 = 1; // For first tube
L_1 = sqrt(2*n_1*q*V_0/m)/(2*f);    // Length of the
    first tube, m
n_n = 128; //  For last tube
L_n = 1/(2*f)*sqrt(2*n_n*q*V_0/m); // Length of the
    last tube,m
printf("\n Length of the first tube = %4.2f m \n
    Length of the last tube = %4.2f m ", L_1,L_n);
// Result
//     Length of the first tube = 0.03 m
//     Length of the last tube = 0.34 m
```

Scilab code Exa 6.5.3 Electron speed at relativistic energies

```
// Scilab code Exa6.5.3 : To calculate the velocity
    of the electrons using relativistic
    considerations . Page 269 (2011)
K_E = 1.17; // Kinetic energy of the electron, MeV
E_r = 0.511; // Rest mass energy of the electron,
    MeV
```

```
v = [1-1/(K_E/E_r+1)^2]; // Velocity of the electron
    , m/s
printf("\nVelocity of the electron : %4.2fc", v)
// Result
//    Velocity of the electron : 0.91c
```

Scilab code Exa 6.7.1 Proton accelerating in a cyclotron

```
// Scilab code Exa6.7.1 : To calculate the maximum
    energy, oscillator frequency and number of
    revolutions of proton accelerated in a cyclotron.
     Page 270(2011)
V = 20e+03; // Potential difference across the dees,
     V
r = 0.28; // Radius of the dees, m
B = 1.1; // Magnetic field, tesla
q = 1.6e-019; // Charge of the proton, C
m = 1.67e-027; // Mass of the proton, Kg
E_max = B^2*q^2*r^2/(2*m*1.6e-013); // Maximnum
    energy acquired by protons,MeV
f = B*q/(2*%pi*m*10^06); // Frequecy of the
    oscillator,MHz
N = E_max*1.6e-013/(q*V); // Number of revolutions,
disp(N)
printf("\n Maximum energy acquired by proton    = %4
    .2f MeV \n Frequency of the oscillator    = %4.2f
    MHz \n Number of revolutions     = %d revolutions
     ", E_max,f,N)
// Result
//    Maximum energy acquired by proton    =  4.54
    MeV
//  Frequency of the oscillator    = 16.77 MHz
//  Number of revolutions     =  227 revolutions
```

Scilab code Exa 6.7.2 Frequency of deutron accelerated in a cyclotron

```
// Scilab code Exa6.7.2 : To calculate the frequency
    of deutron accelerated in a cyclotron. Page
   271(2011)
B= 2.475; // Magnetic field , tesla
q = 1.6e-019; // Charge of the deutron , C
m = 2*1.67e-027; // Mass of the deutron , Kg
f = B*q/(2*%pi*m*10^06); // Frequency of the deutron
   ,MHz
printf("\nFrequency of the deutron: %4.2f MHz ", f)
// Result
//   Frequency of the deutron: 18.87 MHz
```

Scilab code Exa 6.7.3 Relation between magnetic field and cyclotron frequency

```
// Scilab code Exa6.7.3 : To calculate the magnetic
   field applied to cyclotron whose frequency is
   given. Page 271(2011)
q = 1.6e-019; // Charge of the proton , C
r = 0.60; // radius of the dees , m
m = 1.67e-027; // Mass of the proton , Kg
f = 10^6; // Frequency of the proton ,Hz
B = 2*%pi*m*f/q; // Magnetic field applied to
   cyclotron , tesla
printf("\nMagnetic field applied to cyclotron : %6
   .4f tesla ", B)
// Result
//    Magnetic field applied to cyclotron : 0.0656
   tesla
```

Scilab code Exa 6.7.4 Frequency of alternating field

```
// Scilab code Exa6.7.4 : To calculate the frequency
    of alternating field applied to dees. Page
   272(2011)
q = 1.6e-019; // Charge of the proton, C
m = 1.67e-027; // Mass of the proton, Kg
B = 1.4; // Magnetic field , tesla
f = B*q/(2*%pi*m*10^06); // Frequency of the applied
    field, tesla
printf("\n Frequency of the applied field :  %4.2f
   MHz", f)
// Result
//     Frequency of the applied field :  21.35 MHz
```

Scilab code Exa 6.8.1 Energy gained by an electron in the magnetic field

```
// Scilab code Exa6.8.1. : To calculate the energy
   gained per turn of an electron present in given
   magnetic field. Page 273(2011)
e = 1.6e-019 ; // Charge of an electron, C
f = 60; // Frequency of variation magnetic field, Hz
B_0 = 1; // Magnetic field , tesla
r_0 = 1; // Radius of doughnut, m
E = 4*e*2*%pi*f*r_0^2/(1.6e-019); // Energy gained
   by electron per turn, eV
E_g = round(E)
printf("\n Energy gained by electron per turn:  %d
   eV", E_g)
// Result
//      Energy gained by electron per turn:  1508 eV
```

Scilab code Exa 6.9.1 Ratio of highest to the lowest frequency of accelerating proton

```
// Scilab code Exa6.9.1 : To determine the ratio of
   highest to the lowest frequency of cyclotron
   accelerating protons whose energy is given. Page
   273(2011)
K = 500; // Kinetic energy of the proton, MeV
E_r = 938; // Rest mass energy of the proton, MeV
R_f = E_r/(K+E_r); // The ratio of highest to the
   lowest frequency,
printf("\nThe ratio of highest to the lowest
   frequency : %4.2f ", R_f)
// Result
//    The ratio of highest to the lowest frequency :
     0.65
```

Scilab code Exa 6.9.2 W B ration of completely stripped nitrogen

```
// Scilab code Exa6.9.2 : To calculate the w/B ratio
    for a completely stripped nitrogen to move in a
   stable orbit : Page 274(2011)
E_k = 1200; // Kinetic energy of the proton, MeV
q = 7; // Number of proton in nitrogen
E_r = 13040 //  Rest mass energy of the electron,
   MeV
E = (E_k+E_r)*1.6e-013; // Total energy,j
c = 3e+08; // Velocity of light, m/s
R_w_B = q*1.6e-019*c^2/E; // Ratio of w/B, m^2/W
printf("\nThe ratio of w/B : %4.2e m^2/W ", R_w_B)
// Result
//      The ratio of w/B :  4.42e+007 m^2/W
```

Scilab code Exa 6.10.1 Magnetic field of the electron

```
// Scilab code Exa6.10.1 : To calculate the value of
    magnetic field of the electron whose energy is
   given   Page 274(2011)
q = 1.602e-019; // Charge of an electron , C
r = 0.28; // Radius of stable orbit ,m
E = 70*1.6e-013;  //  Energy of the electron , j
c = 3e+08; // Velocity of light , m/s
B = E/(e*r*c); // Magnetic field , T
printf("\nThe magnetic field of the electron :  %4.2
   f T", B)
// Result
//   The magnetic field of the electron :  0.83 T
```

Scilab code Exa 6.10.2 Radius of proton orbit in synchrotron

```
// Scilab code Exa6.10.2 : To calculate the radius
   of proton orbit in synchrotron of given energy
   Page 275(2011)
c= 3e+08; // Speed of light in vacuum, m/s
q = 1.602e-019; // Charge on proton , coulomb
amu = 931;     // Energy equivalent of 1 amu, MeV
m = 938;   //   Rest mass of a proton , MeV
KE = 12e+03;     // Kinetic energy of proton , MeV
B = 1.9; // Magnetic field , T
E = m + KE;     // Total energy of proton , MeV
// As E = m*amu, solving for m, the mass of proton
m = E/amu*1.672e-027;     // Proton mass in motion ,
   kg
v = 0.9973*c; // Velocity of the proton , m/s
```

```
r = m*v/(B*q); // Radius of the proton, m
printf("\nRadius of the proton orbit : %4.2f m", r)
// Result
//         Radius of the proton orbit:  22.84 m
```

Chapter 7

Radiation Detectors

Scilab code Exa 7.2.1 Energy of alpha particle

```
// Scilab code Exa7.2.1: To calculate the  energy of
    alpha particle :P.no. 308 (2011)
  E_p = 30; // Energy required for one pair , eV
  n = 150000; // Number of pairs
 E_a = n*E_p/10^6; // Energy of alpha particle , Mev
  printf("\n The energy of alpha particle  : %3.1f
    Mev", E_a)
// Result
//     The  energy of alpha particle  : 4.5 Mev
```

Scilab code Exa 7.3.1 Pulse height of ionising particle

```
// Scilab code Exa7.3.1: To calculate the  pulse
   height of ionising particle :P.no. 308 (2011)
  E = 5.48e+06; // Energy of alpha particle , eV
  C = 50e-012; // Capacitance of the chamber , F
  R = 10^6; // Resistance , ohm
  E_p = 35; // Energy required to produced an ion
    pair , eV
```

```
n = E/E_p; // Number of ion pair produced
e = 1.6e-019; // Charge of an electron, C
V =( n*e)/C; // Pulse height, V
I = V/R; // current produced, A
printf("\n The pulse height  =  %4.3e V \n Current produced  = %5.3e A", V,I)
// Result
// The pulse height     =   5.010e-004 V
//Current produced    = 5.010e-010 A
```

Scilab code Exa 7.3.2 Charge deposited on detector plate

```
// Scilab code Exa7.3.2: To calculate the  kinetic energy and amount of charge collected on plate :P.no. 309 (2011)
E_p = 35; // Energy required to produced an ion pair, eV
n = 10^5; // Number of ion pair produced
e = 1.6e-019; // Charge of an electron, C
E_k = E_p*n/10^6; // Kinetic energy of the proton, MeV
A = n*e; // The amount of charge collected on each plate, C
printf("\n The kinetic energy of the proton    = %3.1f MeV  \n The amount of charge collected on each plate      =  %3.1e C ", E_k, A)
// Result
//   The kinetic energy of the proton = 3.5 MeV
//   The amount of charge collected on each plate  =  1.6e-014 C
```

Scilab code Exa 7.4.1 Height of voltage pulses

```
// Scilab code Exa7.4.1: To calculate the charge flow in a counter and height of voltage pulses :P.no. 310 (2011)
E_p = 30; // Energy required to produced an ion pair, eV
M = 1000; // Multiplication factor
   e = 1.6e-019; // Charge of an electron, C
   t = 10^-3; // Time, s
   R = 10^5; // Resistance, ohm
E_k = 20*10^6; // Kinetic energy of the proton, eV
n = E_k/E_p; // Number of ion pairs produced
n_a = n*M; // Number of ion-pair after multiplication
Q = n_a*e; // Charge carried by these ion, C
 I = Q/t; //  The current through 100-ohm resistance, A
 A = I*R; // ,The amplitude of voltage pulse, V
printf("\n The current through 100-ohm resistance = %6.4e A \n The amplitude of voltage pulse = %6.4e V ", I, A)
// Result
// The current through 100-ohm resistance    = 1.0667e-007 A
// The amplitude of voltage pulse   = 1.0667e-002 V
```

Scilab code Exa 7.4.2 Electric field at the surface of wire

```
// Scilab code Exa7.4.2: To calculate the electric field at the surface of wire :P.no. 310 (2011)
V = 1500; // Potential difference, V
a = 0.0001; // Radius of the wire, m
b = 0.02; // Radius of the cylinderical tube, m
r = 0.0001; // Distance of electric field from the surface, m
E_r = V/(r*log(b/a)); // the electric field at the
```

```
surface, V/m
printf("\n The electric field at the surface : %4.2e V/m", E_r)
// Result
//    The electric field at the surface : 2.83e+006 V/m
```

Scilab code Exa 7.5.1 Electric filed in G M counter

```
// Scilab code Exa7.5.1: To calculate the electric field at the surface of wire of G.M. counter :P. no. 311 (2011)
V = 2000; // Potential difference, V
a = 0.01; // Radius of the wire, cm
b = 2; // Radius of the cylinderical tube, cm
r = 0.01; // Radius of the wire, m
E_r = V/(r*log(b/a)); // the electric field at the surface, V/m
printf("\n The electric field at the surface : %d V/cm", E_r)
// Result
//    The electric field at the surface : 37747 V/cm
```

Scilab code Exa 7.5.2 Life of G M counter

```
// Scilab code Exa7.5.2: To calculate the life of G.M. counter :P.no. 312 (2011)
n_t = 10^9; // Total number of counts
n_d = 2000*3*60; // Count recorded per day
n_y = n_d*365; // Counts recorded in 365-days
t = n_t/n_y; // The life of G.M. counter, year
printf("\nThe life of G.M. counter : %4.2f year", t)
// Result
```

```
//     The life of G.M. counter : 7.61 year
//
```

Scilab code Exa 7.5.3 Amplitude of voltage pulses in G M counter

```
// Scilab code Exa7.5.3: To calculate the voltage pulse of G.M. counter :P.no. 312 (2011)
E_p = 30; // Energy required for one electron pair, eV
E = 10e+06 ; // Energy lost by alpha particle, eV
n = E/E_p; // Number of ion-pairs produced
M = 5000; // Multiplication factor
C = 50e-012; // Capacitance, F
n_M = n*M; // Number of ion-pairs after multiplication
e = 1.6e-019; // Charge of an electron, C
Q = n_M*e; // Charge present in each ion
A = Q/C; // Amplitude of voltage pulse, V
printf("\n Amplitude of voltage pulse : %3.1f V", A)
// Result
//      Amplitude of voltage pulse : 5.3 V
```

Scilab code Exa 7.5.4 Estimating true count rate of G M counter

```
// Scilab code Exa7.5.4: To estimate the true count rate of G.M. counter :P.no. 312 (2011)
n = 30000; // Count per minute
n_o = n/60; // Observed count rate, count/s
t = 2e-04; // Dead time, s
n_t = round(n_o/(1-n_o*t)); // The true count rate, count/s
printf("\n The true count rate : %d counts/s", n_t)
```

```
// Result
//   The true count rate : 556 counts/s
```

Scilab code Exa 7.6.1 Energy resolution of gamma rays

```
// Scilab code Exa7.6.1: To calculate the energy resolution of gamma rays emitted by Na-22 for channel first and second :P.no. 313 (2011)
// For 511 KeV gamma rays (for channel first)
F_W_H_M_1 = 97; //  Frequency width at half maximum for channel first
P_pos_1 = 1202; // Peak position for channel first
Res_KeV_1 = F_W_H_M_1/P_pos_1*511; // Resolution in KeV for channel first
// For 1275 KeV gamma rays (for channel second)
F_W_H_M_2 = 82; // Frequency width at half maximum for channel second
P_pos_2 =  1202; // Peak position for channel second
Res_KeV_2 = round(F_W_H_M_2/P_pos_2*1275); // Resolution in KeV for channel second
  printf("\n Resolution  for channel first  = %d KeV \n Resolution  for channel second  = %d KeV ",Res_KeV_1, Res_KeV_2)
  // Result
//   Resolution  for channel first = 41 KeV
//   Resolution  for channel second = 87 KeV
```

Scilab code Exa 7.6.2 Amplitude of output voltage pulse

```
// Scilab code Exa7.6.2 : To calculate the amplitude of output voltage pulse for NaI(Tl) :P.no. 314 (2011)
e = 1.6e-019; // Charge of an electron, C
```

```
n = 4.2e+08; // Number of photoelectrons
C = 200e-012; // Capacitance , F
A = n*e/C; // Amplitude of output voltage pulse , V
printf("\n Amplitude of output voltage pulse : %4.2f
    V ",A)
// Result
//            Amplitude of output voltage pulse :
   0.34 V
```

Scilab code Exa 7.6.3 Resolution of scintillation detector

```
// Scilab code Exa7.6.3 : To calculate the %-
   resolution and resolution in KeV for
   scintillation detector for Cs-137 :P.no. 315
   (2011)
F_W_H_M = 0.72; // Full width at half maximum, V
P_p = 6.0; // Peak position , V
E = 662; // Energy of photopeak , KeV
%_resolution = F_W_H_M/P_p*100; // Percentage
   resolution in percent
Res_KeV = %_resolution/100*E; // Resolution in KeV
   for Cs-137
printf("\n The percentage resolution    = %d percent
      \n Resolution in KeV   = %4.1f KeV   ",
   %_resolution, Res_KeV)
// Result
//       The percentage resolution    = 12 percent
//         Resolution in KeV   = 79.4 KeV
```

Scilab code Exa 7.7.1 Silicon pulse detector

```
// Scilab code Exa7.7.1 : To calculate the thickness
    of depletion layer of silicon detector and
   amplitude of voltage pulse :P.no. 316 (2011)
E_r = 12; // Relative permittivity
E_o = 8.85e-012; // Permittivity of free space
E = E_r*E_o; // Absolute dielectric constant
C = 100e-012; // Capacitance of the dielectric , F
A = 1.6e-04; // Area of the detector , m^2
e = 1.602e-019; // Charge of an electrin , C
E_p = 3.2; // Energy required to create an ion pair ,
    eV
E_s = 12e+06; // Energy required to stopped ion pair
   , eV
n = E_s/E_p; // Number of ion-pair produced
Q = n*e; // Charge of these ion pair , C
d = A*E/(C*10^-6); // The thickness of the depletion
    layer , micron
A = Q/C*1000; // The amplitude of voltage pulse , mV
printf("\n The thickness of the depletion layer    =
    %d micron \n  The amplitude of voltage pulse:
      = %6.4f mV  ", d, A)
// Result
//       The thickness of the depletion layer    =
   169 micron
//       The amplitude of voltage pulse:    = 6.0075
   mV
```

Scilab code Exa 7.7.2 Detector characteristics

```
// Scilab code Exa7.7.2 : To calculate the
   capacitance and the amplitude of voltage pulse
   across the detector :Page 316 (2011)
E_r = 12; // Relative permittivity
E_o = 8.85e-012; // Permittivity of free space
E = E_r*E_o; // Absolute dielectric constant
```

```
A = 2e-04; // Area of the detector , m^2
e = 1.602e-019; // Charge of an electron , C
d = 100e-06; // The thickness of the depletion layer
    , m
C = E*A/d; // The capacitance of the dielectric , F
E_p = 3.0; // Energy required to create an ion pair ,
     eV
E_s = 5.48e+06; // Energy required to stopped ion
    pair , eV
n = E_s/E_p; // Number of ion-pair produced
Q = n*e; // Charge of these ion pair , C
A = Q/C*1000; // The amplitude of voltage pulse , mV
printf("\n The capacitance of dielectric    = %5.3e F
     \n The amplitude of voltage pulse    = %5.3f mV
     ", C, A)
// Result
//     The capacitance of dielectric    = 2.124e-010 F
//    The amplitude of voltage pulse    = 1.378 mV
```

Chapter 8

Particle Physics

Scilab code Exa 8.5.1 Average kinetic energy of pion

```
// Scilab code Exa8.5.1: To calculate the average kinetic energy of each pion:P.No.360 (2011)
// Proton and antiproton annihilate to produced three pions
E_p = 938; // Energy of proton, MeV
E_pi = 139.5; // Energy of pions, MeV
E_pi_0 = 134.9; // Energy of pi_0_ion, MeV
E_KE = [2*E_p-(2*E_pi+E_pi_0)]/3; // The average kinetic energy of each pions, MeV
printf("\n  The average kinetic energy of each pions : %5.1f MeV", E_KE)
// Result
// The average kinetic energy of each pions : 487.4 MeV
```

Scilab code Exa 8.5.2 Inherent uncertainity in mass of the particle

```
// Scilab code Exa8.5.2: To calculate the inherent
```

```
uncertainity in mass of the given particle : P.no
    . 360 (2011)
// Here r_1 and r_2 are two decay rates are given
// Declare the cell
R1 = cell(1,2)
R1(1,1).entries = 'r_1'
R1(1,2).entries = 'r_2'
   printf("\n The inherent uncertainity in mass of
      particle = h(%s + %s) ", R1(1,1).entries, R1
      (1,2).entries)
// Result
//   The inherent uncertainity in mass of particle =
    h(r_1 + r_2)
```

Scilab code Exa 8.7.3 Sub nuclear reactions

```
// Scilab code Exa8.7.3: Determine the possibility
    of the given reaction:P.no. 362 (2011)
// Declare cell for the given reaction
R1 = cell(7,5)
// Enter data for the cell
R1(1,1).entries = 'p'
R1(1,2).entries = 1
R1(1,3).entries = 1
R1(1,4).entries = 0
R1(1,5).entries = 1/2
R1(2,1).entries = 'K_+'
R1(2,2).entries = 1
R1(2,3).entries = 0
R1(2,4).entries = 1
R1(2,5).entries = 1/2
R1(3,1).entries = 'S_+'
R1(3,2).entries = 1
R1(3,3).entries = 1
R1(3,4).entries = -1
```

```
R1(3,5).entries = 1
R1(4,1).entries = 'pi_-'
R1(4,2).entries = -1
R1(4,3).entries = 0
R1(4,4).entries = 0
R1(4,5).entries = 1
R1(5,1).entries = 'S_0'
R1(5,2).entries = 0
R1(5,3).entries = 1
R1(5,4).entries = -1
R1(5,5).entries = 0
R1(6,1).entries = 'p_-'
R1(6,2).entries = -1
R1(6,3).entries = -1
R1(6,4).entries = 0
R1(6,5).entries = 1/2
R1(7,1).entries = 'n_0'
R1(7,2).entries = 0
R1(7,3).entries = 0
R1(7,4).entries = 0
R1(7,5).entries = 0

function f = check_Isotopic_no(Ir_sum,Ip_sum)
        if Ir_sum == Ip_sum then
            f = 1;
        else
            f = 0;
        end
endfunction
1
// Declare a function returning equality status of proton number
function f = check_strangeness(sr_sum,sp_sum)
        if sr_sum == sp_sum then
            f = 1;
        else
            f = 0;
        end
```

```
endfunction
function f = check_charge(cr_sum,cp_sum)
        if cr_sum == cp_sum then
            f = 1;
        else
            f = 0;
        end
endfunction
// Declare a function returning equality status of lepton number

//      Reaction-I
printf("\n\n\nReaction-I:\n\n");
        Ir_sum = R1(1,5).entries+R1(1,5).entries;
        Ip_sum = R1(2,5).entries+R1(3,5).entries;
       if (check_Isotopic_no(Ir_sum,Ip_sum) == 0) then
            printf("The Reaction\n")
            printf("\t%s + %s --> %s + %s \nis  not possible", R1(1,1).entries,  R1(1,1).entries, R1(2,1).entries, R1(3,1).entries)
//      Reaction-II
            printf("\n\n\nReaction-II")
        sr_sum = R1(1,4).entries+R1(4,4).entries;
        sp_sum = R1(5,4).entries+R1(7,4).entries;
          if (check_strangeness(sr_sum,sp_sum)== 0) then
            printf("\n\nThe Reaction\n")
            printf("\t%s + %s --> %s + %s \nis not possible", R1(1,1).entries, R1(4,1).entries, R1(5,1).entries, R1(7,1).entries)
//      Reaction-III
            printf("\n\n\nReaction-III:\n\n");
        cr_sum = R1(1,2).entries+R1(1,2).entries;
        cp_sum = R1(1,2).entries+R1(1,2).entries+R1(1,2).entries+R1(6,2).entries;
```

```
                if (check_charge(cr_sum,cp_sum) == 1) then
            printf("The Reaction\n")
            printf("\t%s + %s --> %s + %s + %s \nis possible", R1(1,1).entries, R1(1,1).entries, R1(1,1).entries, R1(1,1).entries, R1(6,1).entries)
        end
        // Reaction-I:

// The Reaction
//  p + p --> K_+ + S_+
// is  not possible


// Reaction-II

// The Reaction
//  p + pi_- --> S_0 + n_0
// is not possible


// Reaction-III:

// The Reaction
//  p + p --> p + p + p_-
// is possible
```