SIMULATION OF CONTINUOUS SYSTEMS

3.1 Introduction

In Chapter 1, the classification of systems into continuous and discrete was discussed. In a continuous system, the state of the system changes continuously generally with time. But the time is not an essential parameter of a continuous system. In many situations, some parameter, other than time, is continuously changed to determine the behavior of the system. The recursive procedures for solving problems do not involve time, but are examples of continuous systems. Similarly, determining the value of π by simulation and solving an integral by simulation are examples of continuous systems. But many of the continuous systems, as the progress of a chemical reaction, flight of a missile or a servo system are time dependent, and may be called dynamic systems. Such systems are generally described by means of a set of differential equations. If the differential equations are ordinary, linear and time-invariant, their solution can easily be obtained analytically. However, when the equations are complex and more involved, having variable coefficients, their solutions can be obtained only by employing numerical methods, which involve a very large amount of computations, which can be carried out only by using computers. Simulation of continuous systems, is the process of carrying out the computations with a detailed insight into the system. There is hardly any difference between the numerical solution and a continuous simulation of a problem.

In this chapter, we will discuss some examples of continuous systems.

3.2 A Pure Pursuit Problem

In a pure pursuit problem, there is a target, which moves along a predetermined path, and there is a pursuer, who follows the target, redirecting itself towards the target at fixed intervals of time. The target moves along its predetermined path and does not make any effort to evade the pursuer. A fighter aircraft following an enemy bomber is an example of pure pursuit problem. The aircraft flies towards the bomber to catch up with it and to destroy it. The bomber continues to fly along a predetermined path and the fighter has to change its direction to keep pointing towards the bomber. Now, the typical question is whether the fighter will be able to destroy the bomber or not, if so in how much time. The answer depends upon a number of factors, out of which one is the path of the bomber. If it is a straight line, the problem can easily be solved by employing analytical techniques. But if the target follows a curved path, the solution of the problem becomes difficult, and simulation is the technique, which can be employed.

Let us consider the situation, where the target and pursuer fly in the same horizontal plane, and stay in that plane during the period of pursuit. The fighter's speed S is constant at 20 km/min, while the target's path is specified as a function of time as below:

Time (t) :	0	1	2	3	4	5	6	7	8	9
xb (t) :	100	100	120	129	140	149	158	168	179	188
yb (t) :	0	3	6	10	15	20	26	32	37	34

Time (t) :	10	11	12	13	14	15
xb (t) :	198	209	219	226	234	240
yb (t) :	30	27	23	19	16	14

The fighter aircraft is at position xf, yf (0, 50) when it sights the bomber that is at time $t = 0$, the time at which pursuit begins. The fighter corrects its direction after a fixed interval of one minute, so as to point towards the bomber, and shoots the bomber by firing a missile as soon as it is within a distance of 10 km. The pursuit ends. In case the bomber is not shot within 12 minutes of the pursuit, the pursuit is abandoned.

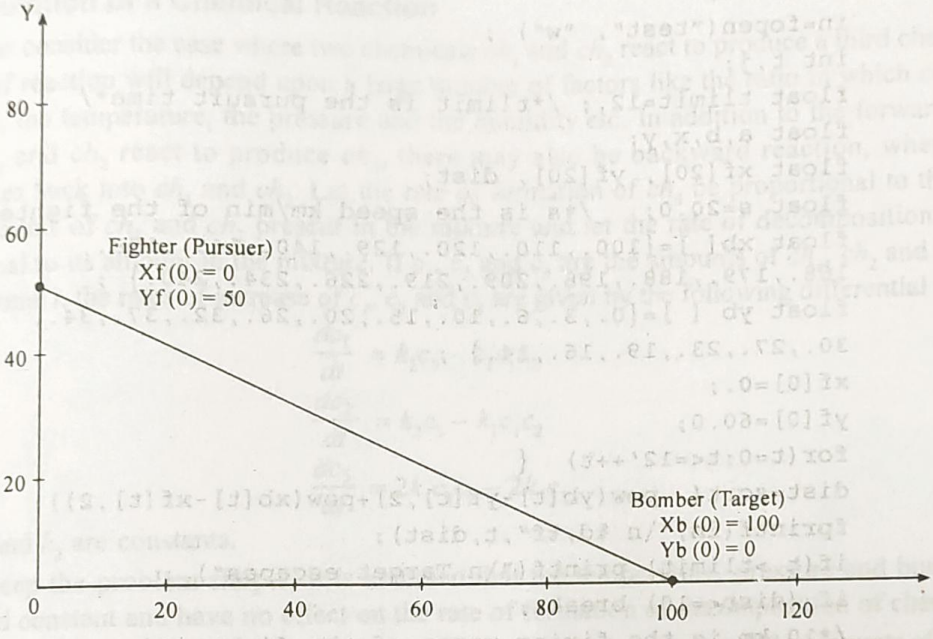

Fig. 3.1

The pursuit begins at time $t = 0$, when the bomber is at $(xb, yb) = (100, 0)$ and the fighter is at $(xf, yf) = (0, 50)$. The fighter aligns its velocity vector with the point of sight towards the bomber. It continues to fly in that direction for one minute and at time $(t + 1)$, it looks at the target again and realigns itself.

At a time t, the distance Dist (t) between fighter and the bomber is given by.

$$\text{Dist}\,(t) = \sqrt{\left(yb\,(t) - yf\,(t)\right)^2 + \left(xb\,(t) - xf\,(t)\right)^2}$$

If θ is the angle which the line connecting bomber and fighter makes with X-direction then,

$$\sin\theta = \frac{yb\,(t) - yf\,(t)}{Dist\,(t)}; \quad \cos\theta = \frac{xb\,(t) - xf\,(t)}{Dist\,(t)}$$

The co-ordinates of the position of the fighter at time $(t + 1)$ can be determined as,

$$xf\,(t+1) = xf\,(t) + S\cos\theta$$
$$yf\,(t+1) = yf\,(t) + S\sin\theta$$

With these new coordinates of the fighter, its distance from the target is again computed. If the distance is 10 km or less, the pursuit ends, otherwise the values of sin θ, cos θ and the new co-ordinates of the fighter are recomputed. If the pursuit does not end within the specified time, the bomber is considered as escaped and pursuit is abandoned.

A computer program for this simulation written in C language is given below :

```
#include<stdio.h>
#include <math.h>
void main ( )
    {
        /*Pursuit-Simulation of a pursuit problem.*/
        /* define and inialise variables*/
        FILE *in;
        in=fopen("test", "w") ;
        int t,j;
        float tlimit=12.; /*tlimit is the pursuit time*/
        float a,b,x,y;
        float xf[20], yf[20], dist;
        float s=20.0;   /*s is the speed km/min of the fighter*/
        float xb[ ]={100.,110.,120.,129.,140.,149.,158.,
        168.,179.,188.,198.,209.,219.,226.,234.,240.} ;
        float yb [ ]={0.,3.,6.,10.,15.,20.,26.,32.,37.,34.,
        30.,27.,23.,19.,16.,14.} ;
        xf[0]=0.;
        yf[0]=60.0;
        for(t=0;t<=12'++t)  {
        dist=sqrt(  pow(yb[t]-yf[t],2)+pow(xb[t]-xf[t],2));
        fprintf(in,"\n %d,%f",t,dist);
        if(t >tlimit) printf("\n Target escapes") ;
        if (dist<=10) break ;
        /*10 km is the firing range of the fighter*/
        else {
            xf[t+1]=xf[t]+s*(xb[t]-xf[t])/dist;
            yf[t+1]=xf[t]+s*(yb[t]-yf[t])/dist;
        }}
        fprintf(in,"\n Persuit ends,shot at %d minutes and at
        %5.2f kms.",t,dist);

        fprintf(in,"\n vlaue of s");
        scanf("%f",s);
        fclose(in);
```

The output of this program for the given data is,

0, 116.619041

1, 103.937393

2, 91.803185

3, 78.942284

4, 68.461441

5, 56.794758

6, 45.858341
7, 36.668800
8, 28.603636
9, 17.205299
10, 8.210612

Pursuit ends, shot at 10 minutes and at 8.21 km.

3.3 Simulation of a Chemical Reaction

Let us consider the case where two chemicals ch_1 and ch_2 react to produce a third chemical ch_3. The rate of reaction will depend upon a large number of factors like the ratio in which ch_1 and ch_2 are mixed, the temperature, the pressure and the humidity etc. In addition to the forward reaction where ch_1 and ch_2 react to produce ch_3, there may also be backward reaction, where by, ch_3 decomposes back into ch_1 and ch_2. Let the rate of formation of ch_3 be proportional to the product of the amounts of ch_1 and ch_2 present in the mixture and let the rate of decomposition of ch_3 be proportional to its amount in the mixture. If c_1, c_2 and c_3 are the amounts of ch_1, ch_2 and ch_3 at any instant of time t, the rates of increase of c_1, c_2 and c_3 are given by the following differential equations:

$$\frac{dc_1}{dt} = k_2c_3 - k_1c_1c_2$$

$$\frac{dc_2}{dt} = k_2c_3 - k_1c_1c_2$$

$$\frac{dc_3}{dt} = 2k_1c_1c_2 - 2k_2c_3$$

where k_1 and k_2 are constants.

To keep the problem simple, it is assumed that the temperature, pressure and humidity are maintained constant and have no effect on the rate of formation or decomposition of chemicals.

As soon as the chemicals ch_1 and ch_2 are mixed, the reaction starts and the amounts of c_1, c_2 and c_3 in the mixture go on changing as the time progresses. The simulation of the reaction will determine the state of the system, that is the value of quantities c_1, c_2 and c_3, at different points in time. Starting at zero time, a very small increment of time is taken in each step. It is assumed to be so small, that all changes in the mixture can be taken to occur at the end of each increment. If $c_1(t)$, $c_2(t)$ and $c_3(t)$ are the quantities of the three chemicals at time t, then at time $t + \Delta t$, the quantities are;

$$c_1(t + \Delta t) = c_1(t) + \frac{dc_1(t)}{dt}\,\Delta t$$

Identical equations can be written for

$c_2(t + \Delta t)$ and $c_3(t + \Delta t)$.

Taking $c_1(0)$, $c_2(0)$ and $c_3(0)$ as the quantities of ch_1, ch_2 and ch_3 at time zero, that is when the reaction starts the state of the system at time Δt will be,

$$c_1(\Delta t) = c_1(0) + [k_2c_3(0) - k_1c_1(0)\cdot c_2(0)]\,\Delta t$$

$$c_2(\Delta t) = c_2(0) + [k_2c_3(0) - k_1c_1(0)\cdot c_2(0)]\,\Delta t$$

$$c_3(\Delta t) = c_3(0) + [2k_1c_1(0)\cdot c_2(0) - 2k_2c_3(0)]\,\Delta t$$

Using the state of the system at time $2\Delta t$, the state of the system at time $3\Delta t$ and so on will be computed. This will continue till the specified time of reaction T is reached. At each increment of time, we can either count the number of steps taken or check the attained time with the prescribed simulation run time.

A computer program in C language for the simulation of above said chemical reaction is given below.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
main()
    {       /*   Simulation of a chemical reaction*/
            int i,n,k;
             float c1[200],c2[200],c3[200];
             float k1=0.025,k2=0.01,delta=0.1;
             c1[0]=50.0,c2[0]=25.0,c3[0]=0.0;
             float t=0.0, time=8.0;

             i=0
             printf("\n Time    C1      C2      C3 ");
             while(t<=time)  {
             printf("\n%5.2f %6.2f %6.2f %6.2f ",t,c1[i],c2[i],c3[i]);
             c1[i+1]=c1[i]+(k2*c3[i]-k1*c1[i]*c2[i])*delta;
             c2[i+1]=c2[i]+(k2*c3[i]-k1*c1[i]*c2[i])*delta;
             c3[i+1]=c3[i]+2.0*(k1*c1[i]*c2[i]-k2*c3[i])*delta;
             i=i+1;
             t=t+delta;
             if(t >= 2.0) delta=0.2;
             if(t >= 6.0) delta=0.4;
             }
            printf("\n Any digit");
            scanf("%d",k);
}
```

The output of this program for the given values of constants and initial values of variables, for a reaction of 15 minutes, is given below. It can be noticed that as the time progresses, the changes in the mixture become slow. It may not be of much use to continue the reaction beyond a certain time.

Chemical Reactor

$k_1 = 0.025$ $k_2 = 0.01$ time = 15.00

$c_1[0] = 50.0$ $c_2[0] = 25.0$ $c_3[0] = 0.0$

Time	c_1	c_2	c_3
0.00	50.00	25.00	0.00
0.1	46.88	21.88	6.25
0.2	44.32	19.32	11.36
0.3	42.19	17.19	15.62
0.4	40.39	15.39	19.22
0.5	38.86	13.86	22.29
0.6	37.53	12.53	24.93
0.7	36.38	11.38	27.24

0.8	35.37	10.37	29.25
0.9	34.49	9.49	31.03
1.0	33.70	8.70	32.60
1.1	33.80	8.00	34.00
1.2	32.37	7.37	35.25
1.3	31.81	6.81	36.38
1.4	31.31	6.31	37.39
1.5	30.85	5.85	38.30
1.6	30.44	5.44	39.13
1.7	30.06	5.06	39.88
1.8	29.72	4.72	40.56
1.9	29.41	4.41	41.18
2.0	29.13	4.13	41.74
2.2	28.61	3.61	42.78
2.4	28.18	3.18	43.64
2.6	27.82	2.82	44.36
2.8	27.52	2.52	44.97
3.0	27.26	2.26	45.48
3.2	27.04	2.04	45.92
3.4	26.86	1.86	46.28
3.6	26.70	1.70	46.60
3.8	26.57	1.57	46.87
4.0	26.45	1.45	47.09
4.2	26.35	1.35	47.29
4.4	26.27	1.27	47.46
4.6	26.20	1.20	47.60
4.8	26.14	1.14	47.73
5.0	26.08	1.08	47.83
5.2	26.04	1.04	47.92
5.4	26.00	1.00	48.00
5.6	25.97	0.97	48.07
5.8	25.94	0.94	48.13
6.0	25.91	0.91	48.18
6.2	25.89	0.89	48.22
6.6	25.85	0.85	48.30
7.0	25.82	0.82	48.35
7.4	25.81	0.81	48.39
7.8	25.79	0.79	48.42
8.2	25.78	0.78	48.44
9.0	25.77	0.77	48.46
10.20	25.76	0.76	48.48
12.20	25.75	0.75	48.49
15.00	25.75	0.75	48.49

3.4 The Runge-Kutta Method

This method of solving the differential equations was devised by C. Runge about the year 1894 and was extended by W. Kutta in 1901. It is a method designated to approximate the Taylor series without having to evaluate the higher order derivatives. In a simpler Euler's formula, the value of a variable y at $(i+1)$th instant is estimated in the form of its value and shape at ith instant. That is,

$$y_{i+1} = y_i + \frac{dy}{dt} \cdot \Delta t.$$

But in Runge-Kutta method, the first derivative $f(t, y)$ of y is computed at several carefully chosen points in the interval $(t, t+\Delta t)$, and these values are combined in such a way as to obtain good accuracy in the computed increment $y_{i+1} - y_i$. Following this technique, a family of formulas have been derived, which are known as Runge-Kutta formulas. The most commonly used is the fourth order Runge-Kutta formula. If $\frac{dy}{dx} = f(x, y)$ represents any first order equation and h denotes the interval between equally distant values of x, then with the initial values x_0, y_0, the first increment in y is computed from the formula, which is a set of equations given below.

$$k_1 = f(x_0, y_0)\, h,$$

$$k_2 = f\left(x_0 + \frac{h}{2}, y_0 + \frac{k_1}{2}\right) h,$$

$$k_3 = f\left(x_0 + \frac{h}{2}, y_0 + \frac{k_2}{2}\right) h,$$

$$k_4 = f(x_0 + h, y_0 + k_3)\, h,$$

$$\Delta y = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Then

$$x_1 = x_0 + h$$
$$y_1 = y_0 + \Delta y$$

The increment in y for the second interval is computed in a similar manner by means of the formula,

$$k_1 = f(x_1, y_1)\, h,$$

$$k_2 = f\left(x_1 + \frac{h}{2}, y_1 + \frac{k_1}{2}\right) h,$$

$$k_3 = f\left(x_1 + \frac{h}{2}, y_1 + \frac{k_2}{2}\right) h,$$

$$k_4 = f(x_1 + h, y_1 + k_3)\, h,$$

$$\Delta y = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

and so on for the succeeding intervals.

The derivation and any discussion of the Runge-Kutta formulas is beyond the scope of this book. Any book on Numerical Analysis may be referred for further details of this formula. To

illustrate the use of this formula in continuous simulation, let us consider the simulation of an amplifier circuit.

3.5 Simulation of an Amplifier Circuit

Fig. 3.2 shows an electronic circuit, where 'Amp' is a voltage amplifier with amplification factor A. The input impedance to the amplifier is infinite, while output impedance is zero. The input capacitor C_1 is a coupling capacitor of large value, $C_1\,\mu f$. Output capacitor C_2 is of small size, $c_2\mu f$; R_1, R_2, R_3 are resistances.

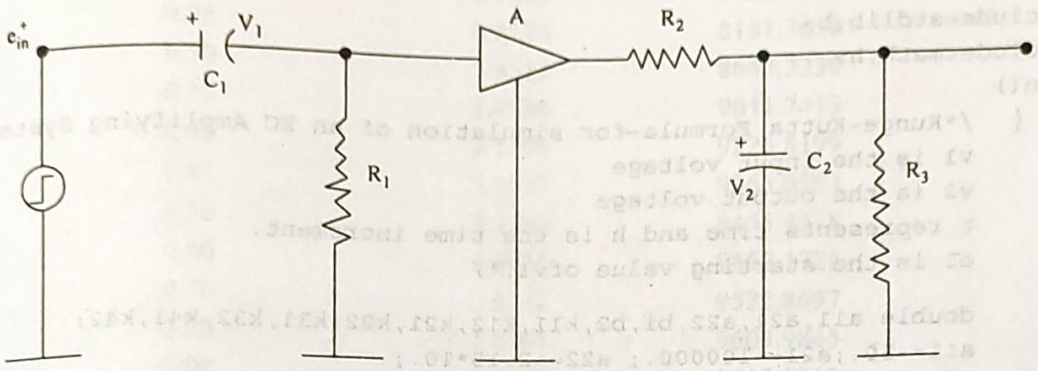

Fig. 3.2

The currents entering the capacitors C_1 and C_2 are given by the equations.

$$C_1 \frac{dv_1}{dt} = (e_{in} - v_1)\frac{1}{R_1}$$

$$C_2 \frac{dv_2}{dt} = (e_{in} - v_1)\frac{A}{R_2} - \frac{v_2\,(R_2 + R_3)}{R_2 R_3}$$

where e_{in} is the input voltage.

These equations can be written as:

$$\frac{dv_1}{dt} = \frac{1}{R_1 C}(e_{in} - v_1)$$

$$\frac{dv_2}{dt} = \frac{A}{R_2 C_2}(e_{in} - v_1) - \frac{(R_2 + R_3)}{R_2 R_3 C_2} \cdot V_2 \quad \text{or} \quad \frac{dv_1}{dt} = A_{11} v_1 + B_1 e_{in}$$

$$\frac{dv_2}{dt} = A_{21} v_1 + A_{22} v_2 + B_2 e_{in}$$

where constants $A_{11} = \dfrac{1}{R_1 C_1} = -B_1$

$$A_{21} = \frac{A}{R_2 C_2} = -B_2$$

$$A_{22} = \frac{R_1 + R_2}{R_1 R_2 C_3}$$

Let us take the values of constants as under:

$$A_{11} = -50 \text{ sec}^{-1}$$
$$A_{21} = -10{,}000 \text{ sec}^{-1}$$
$$A_{22} = 21.5 \text{ sec}^{-1}$$

For simulating the system, let us assume the starting conditions, as time $t = 0$, $v_1 = 0$, $v_2 = 0$. The voltage applied at the input terminal $e_{in} = 1.5$ for $t \geq 0$ and $e_{in} = 0$ for $t < 0$.

The simulation program written in C language is given below.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
main()
   {  /*Runge-Kutta Formula-for simulation of an RC Amplifying System
      v1 is the input voltage
      v2 is the output voltage
      t represents time and h is the time increment.
      e1 is the starting value ofv1.*/

      double a11,a21,a22,b1,b2,k11,k12,k21,k22,k31,k32,k41,k42;
      a11=-50.;a21=-100000.; a22=-2.15*10.;
      b1=-a11;b2=-a21;
      double t=0.,v1=0., v2=0.;
      double h=.05,e1=1.5;
      int i,k;
      int n=10;
      printf("\n a11=%8.2f a21=%8.2f  a22=%8.2f  h=%4.2f e1=%4.2f
      ",a11,a21,a22,h,e1);
      printf("\n i   Time(u sec.)   Voltage,v1   Voltage v2");

      for(i=1;i<=n;++i) {
      k11=h*((a11*v1)+e1);
      /*printf("\n %6.2f",k11);*/
      k12=h*((a21*v1)+(a22*v2)+b2);
      k21=h*(a11*(v1+.5*k11)+b1);
      k22=h*(a21*(v1+.5*k11)+a22*(v2+.5*k12)+b2);
      k31=h*(a11*(v1+.5*k21)+b1);
      k32=h*(a21*(v1+.5*k21)+a22*(v2+.5*k22)+b2);
      k41=h*(a11*(v1+k31)+b1);
      k42=h*(a21*(v1+k31)+a22*(v2+k32)+b2);
      /*printf("\n %6.2f %6.2f %6.2f %6.2f %6.2f
      %6.2f",k11,k12,k21,k22,k31,k32);*/
      v1=v1+(k11+2.*k21+2.*k31+k41)/6.;
      v2=v2+(k12+2.*k22+2.*k32+k42)/6.;
      t=t+h;
      printf("\n %d %8.2f      %12.4f    %12.4f",i,t,v1,v2);
      }
      printf("\n any digit");
      scanf("%d",k);
      }
```

The output of this program is as under.

$a_{11} = -50.00$ $a_{21} = -10,0000.00$ $a_{22} = -21.50$

$h = 0.05$ $e_1 = 1.50$

Time (μ/*sec*)	*Voltage* (v_1)	*Voltage* (v_2)
0.050	1.2736	2465.5143
0.10	2.1034	4659.2401
0.15	2.6349	6290.8198
0.20	2.9821	7422.2709
0.25	3.2173	8181.7658
0.30	3.3533	8683.3239
0.35	3.4580	9011.7413
0.40	3.5194	9225.8199
0.45	3.5592	9365.0303
0.50	3.5750	9455.4378
0.60	3.6026	9552.1725
0.70	3.6142	9522.8697
0.80	3.6191	9609.9845
0.90	3.6211	9617.1812
1.00	3.6220	9620.2072
1.20	3.6225	9622.0146
1.40	3.6226	9622.3582
1.50	3.6226	9622.3739

3.6 A Serial Chase Problem

In this chase problem, there are a number of moving objects, which chase each other in a serial order like A chases B; B chases C and C chases D etc. Each object moves towards its target unmindful of the fact that it itself is being targeted by some other object. Depending upon the original location, and speed of motion of the objects, each object will take its own time to hit its target. As soon as a hit occurs, the chase ends.

Let us consider the objects A, B, C and D located at the four corners of a square field. A is to kill B, B is to kill C and C is to kill D. The velocities of A, B, C and D are *va*, *vb*, *vc* and *vd* respectively. Let θ*a* be the angle which the velocity vector of A makes with the direction AB. Similarly let θ*b* be the angle which velocity vector of B makes with the direction BC. Though C and D move in straight lines in the present case, to generalize the problem, let θ*c* and θ*d* be the angles, which velocity vectors of C and D make with the direction CD. As A runs towards B, it continuously changes its direction so that its velocity vector is in the direction of B. Similarly B continuously changes its direction, so that its velocity vector is always in the direction of its target C. Since D is not chasing any object, it will move in a straight line, so as to keep maximum distance from its hitter C. Object C has to run towards D. This again will move in a straight line, as its target is running straight. Which of the object will be hit first ? is to be determined. This can be determined by simulating the chase. The location of all the four objects is determined after every small increment in time. The direction of motion at new points is determined. The distances between objects are computed. As soon as the distance between any set of objects that is distance AB, BC or CD becomes zero, hit occurs and chase ends. It can be assumed that when the distance becomes very small, say less than 0.005, hit is taken to occur.

The simulation program, written in C language is given below.

Following values for velocities and initial locations of the objects have been taken.

Va = 35 km/hr
Vb = 25 km/hr
Vc = 15 km/hr
Vd = 10 km/hr

Location of A is (10, 10), of B is (30, 10), of C is (30, 30) and of D is (10, 30).

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
main()
    { /*It is a serial chase problem. Four animals A,B,C,D are there
      in a field. A chases B, B chases C and C chases D.as soon as
      the target is within say 0.005km, it is killed and the chase
      ends.
      Speeds of A,B,C,D are given say 35,25,15 and10 km /hour.
      The initial positions are also fixed.It is assumed that A will
      always run in the direction of B and B straight towards C and C
      straight towards D. D runs straight away from C in first case
      and towards A in the second case.
      xa and xb are the coordinates of A.
      thab is the angle with direction ab.
      dab is the distance between A and B.
      va is the velocity of A.
       */
     float xa=10.,ya=10.,xb=30.,yb=10.,xc=30.,yc=30.,xd=10.,yd=30.;
     float thab=0.,thbc=0.,thcd=0.,thd=0.;
     float dab,dbc,dcd,t,delt,k;
     float va=35.,vb=25.,vc=15.,vd=10.;
    t=0.;delt=.001;
    dab=sqrt((xa-xb)*(xa-xb)+(ya-yb)*(ya-yb));
    dbc=sqrt((xb-xc)*(xb-xc)+(yb-yc)*(yb-yc));
    dcd=sqrt((xc-xd)*(xc-xd)+(yc-yd)*(yc-yd));
    while(t<= 1.70) {
    xa=xa+va*delt*(xb-xa)/dab;
    ya=ya+va*delt*(yb-ya)/dab;
    xb=xb-vb*delt*(xb-xc)/dbc;
    yb=yb+vb*delt*(yc-yb)/dbc;
    xc=xc-vc*delt*(xc-xd)/dcd;
    yc=yc-vc*delt*(yc-yd)/dcd;
    xd=xd;
    yd=yd-vd*delt;
    dab=sqrt((xa-xb)*(xa-xb)+(ya-yb)*(ya-yb));
    dbc=sqrt((xb-xc)*(xb-xc)+(yb-yc)*(yb-yc));
    dcd=sqrt((xc-xd)*(xc-xd)+(yc-yd)*(yc-yd));
```

```
printf("\nt=%6.4f   A=%5.2f,%5.2f   B=%5.2f,%5.2f   C=%5.2f,%5.2f
D=%5.2f,%5.2f",
t,xa,ya,xb,yb,xc,yc,xd,yd);
/* printf("\nt=%6.3f dab=%6.2f  dbc=%6.2f
dcd=%6.2f",t,dab,dbc,dcd);
*/
t=t+delt;
if(dab<=0.005) {
t=4.0; printf("\n B killed, chase ends");}
if(dbc<=0.005) {
t=4.0; printf("\n C killed,chase ends");}
if(dcd<=0.005) {t=4.0; printf("\n D killed,chase ends");}
}
printf("\n any digit");
scanf("%d",k);
}
```

For the given data, the path traced by the four objects will be as in Fig. 3.3. The chase ends at 0.897 hrs, when object A hits object B, at location X. (19.87, 28.57). At that instant object C is at C′ and D is at D′. The location of the four objects at different points in time is given in the following table.

A		*B*		*C*		*D*		*T*
x	*y*	*x*	*y*	*x*	*y*	*x*	*y*	
10.00	10.00	30.00	10.00	30.00	30.00	10.00	30.00	0.000
12.69	10.13	29.94	11.92	28.85	30.00	9.23	30.00	0.076
13.18	10.19	29.92	12.27	28.64	30.00	9.09	30.00	0.090
13.49	10.23	29.90	12.50	28.50	30.00	9.00	30.00	0.099
13.84	10.28	29.88	12.75	28.35		8.90		0.109
14.18	10.34	29.86	13.00	28.20		8.80		0.119
14.53	10.40	29.83	13.24	28.05		8.70		0.129
15.21	10.54	29.78	13.74	27.75		8.50		0.149
16.23	10.79	29.67	14.48	27.30		8.20		0.179
18.55	11.59	29.55	16.20	26.25		7.50		0.249
21.61	13.33	28.67	18.63	24.74		6.49		0.350
24.06	15.81	27.71	20.94	23.25		5.50		0.450
25.51	18.97	26.45	23.10	21.74		4.49		0.550
25.51	22.44	24.89	29.05	20.24		3.49		0.650
24.92	24.08	24.00	25.92	19.49		2.99		0.700
23.98	25.55	23.04	26.72	18.74		2.49		0.750
22.76	26.81	22.01	27.44	17.99		1.99		0.800
21.34	27.82	20.95	28.06	17.24		1.49		0.850
19.87	28.57	19.87	28.57	16.50		1.02		0.897

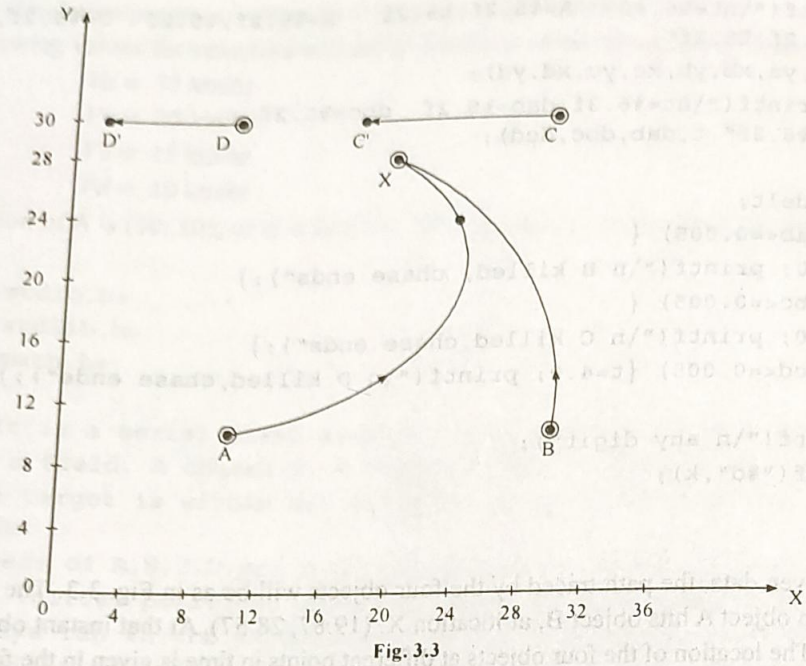

Fig. 3.3

D moves towards A

If the object D decides to run towards point A, its hitter object C will move in a curved path and may be it is hit by the object B. When the simulation is run with this modification, keeping all other data same, the paths traced by the four objects will be as shown in Fig. 3.4. The chase ends, when object A hits its target B; at location X (19.52, 26.64) and at time 0.87 hours. Thus, as compared to the earlier case, the chase ends in a lesser time.

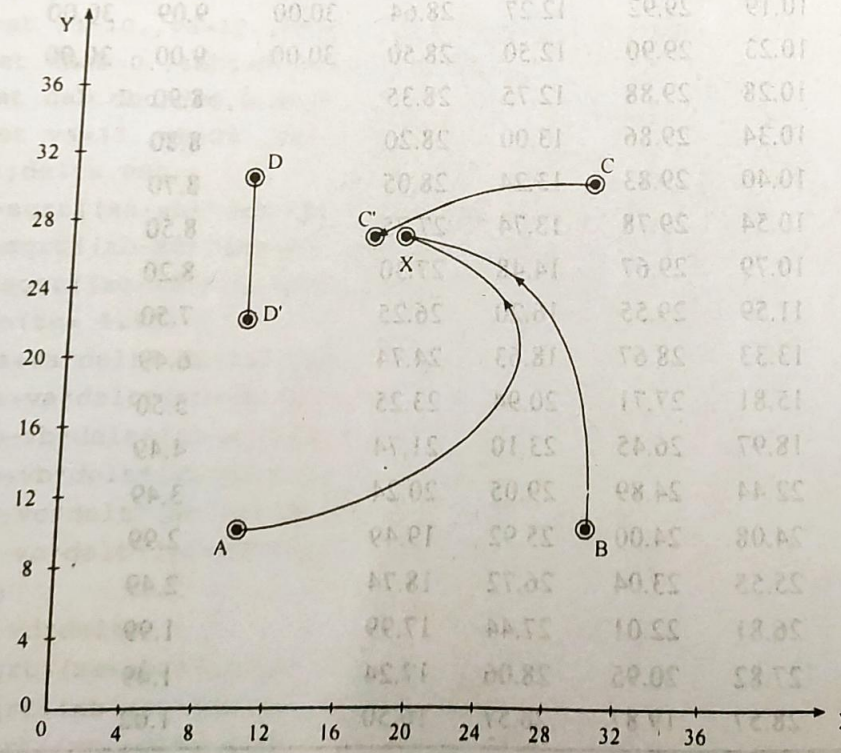

Fig. 3.4

3.7 Simulation of Exterior Ballistics

The exterior ballistics is the science, which deals with the motion of a projectile after it leaves the gun. The projectile is the material particle, which is acted upon by the force of gravity, by a tangential drag and the retarding force due to the resistance of air. In the simplest case, the acceleration due to gravity is assumed to be constant in magnitude and direction, which means that we are assuming a flat earth and that the projectile does not reach a great height. The air resistance is taken proportional to some power of velocity, like kv^n, where k is a constant. The value of n, which further depends upon velocity, can be taken as,

$0 < v < 790$ m/sec;	$n = 2$
320	3
410	5
450	3
600	2
860	1.7
1200	1.55

In simple cases, the problem can be analyzed to form differential equations, which can further be solved by some numerical method. In case of complicated problems, the solution of differential equations becomes difficult and simulation can be employed to solve the problems.

Let a projectile of mass m be fired with an initial velocity and an angle of elevation ϕ. Let v denote the velocity of projectile at any point in its path and let θ denote the inclination of the velocity vector at that point. Let kv^n be the drag due to air resistance, and ρ the radius of curvature of the trajectory at that point. (Fig. 3.5)

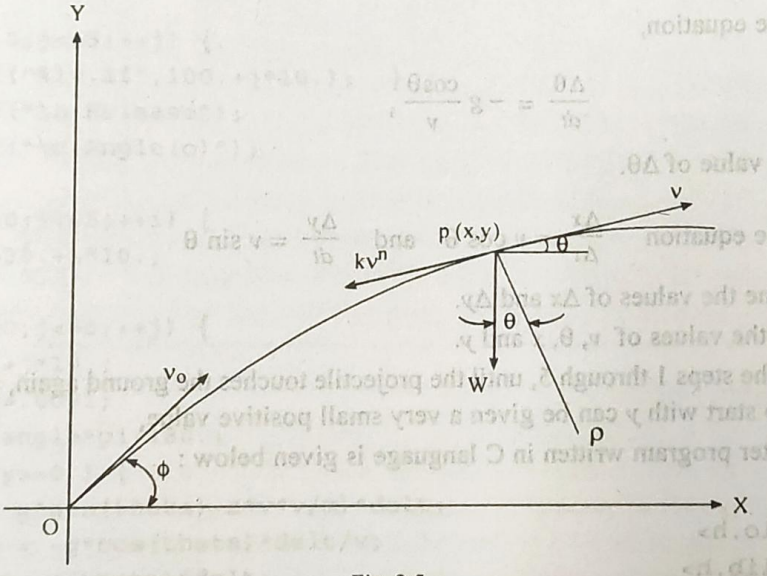

Fig. 3.5

Resolving the forces along the tangent and the normal at point p. We obtain:

$$m\frac{dv}{dt} + mg \sin\theta + kv^n = 0 \qquad ...(i)$$

$$m\frac{v^2}{\rho} + mg \cos\theta = 0 \qquad ...(ii)$$

If at that point $p\ (x, y)$, ds is the distance moved in a small time interval dt and $d\theta$ is the change in angle θ,

$$\rho = \frac{ds}{d\theta} \quad \text{and} \quad v = \frac{ds}{dt}.$$

Equation (*ii*) can be written as

$$mv\frac{d\theta}{dt} + mg\cos\theta = 0$$

Also we can write $$\frac{dx}{dt} = v\cos\theta$$

and $$\frac{dy}{dt} = v\sin\theta$$

The simulation of the projectile trajectory, will be the determination of a large number of points on its path. A very small interval of time Δt is taken, and the velocity and θ are assumed to remain constant over this small duration. Smaller the value of Δt, more accurate will be the simulation results. The simulation will proceed through the following steps:

1. Start with initial values $v = v_0$, $\theta = \phi$, $x = 0$, $y = 0$ and initialize k, m and n etc.
2. From the equation,

$$\frac{\Delta v}{\Delta t} = -g\sin\theta - \frac{kv^n}{m}$$

find the value of Δv.

3. From the equation,

$$\frac{\Delta\theta}{dt} = -g\frac{\cos\theta}{v},$$

find the value of $\Delta\theta$.

4. From the equation $\frac{\Delta x}{\Delta t} = v\cos\theta$ and $\frac{\Delta y}{dt} = v\sin\theta$

determine the values of Δx and Δy.

5. Update the values of v, θ, x and y.
6. Repeat the steps 1 through 5, until the projectile touches the ground again, that is y becomes zero. To start with y can be given a very small positive value.

The computer program written in C language is given below :

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
main()
        {  /* Projectile Trajectory simulation
        Release velocity (v) changes from 100 to 150m/sec in steps of 10
m/sec.
        Angle of release (angle) changes from 30 to 80 degree in steps of
10 degree.
```

```
Mass of the projectile(m) =30 kg.,
Coefficient of air resistance(c)=.001
theta: angle in radians,
delv: change in velocity in one iteration,
delt: change in angle in one iteration,
delx, dely and dtheta: changse in values of x, y and theta
            when time changes through delt,

*/
float v=100.,angle,theta,delv,pi=22./
7.,m=30.,c=.001,delt=.01,g=9.81;
float x,y;
float vel[10] ,range[10];
float delx,dely,dtheta,xx,yy;
int i,j,k;
printf("\n Enter the value of mass m");
scanf("%f",&m);

printf("\n                    RANGE TABLE");
printf("\n          Mass of projectile=%5.2fKg",m);
printf("\n          Release Velocity(m/sec.) \n");
printf("\n       ------------------------------------------
-\n
");

for(j=0;j<=5;++j) {
printf("%10.2f",100.+j*10.);  }
printf("\n Release");
printf("\n Angle(o)");

for(i=0;i<=5;++i) {
angle=30.+i*10.;

for(j=0;j<=5;++j) {
v=100.+j*10.;
x=0.;y=.0001;
theta=angle*pi/180.;
while(y>=0.) {
delv=(-g*sin(theta)-c*v*v/m)*delt;
dtheta = -g*cos(theta)*delt/v;
delx=v*cos(theta)*delt;
dely=v*sin(theta)*delt;

v+=delv;
theta+=dtheta;
x+=delx;
y+=dely;
/*printf("\n %5.2f %5.2f %5.2f %5.2f",v,theta,x,y); */
```

```
}
range[j]=x;
}
printf("\n\n %4.1f",angle);
for(j=0;j<=5;++j) {
printf("%10.2f",range[j]);  }

}
}
```

The program can be further modified to find different parameter of the projectile, as the maximum height reached, the distance traveled and the time of flight. A range table for the above projectile, with θ varying from 30° to 80° and velocity varying from 100 to 150 m/sec and range in meters is given below:

Range Table

Mass of Projectile = 30.00 kg

Release Velocity (m/sec)

Release Angle	*100*	*110*	*120*	*130*	*140*	*150*
30	866.18	1043.64	1235.99	1442.74	1665.32	1899.93
35	937.00	1198.65	1336.01	1559.50	1797.63	2050.94
40	980.20	1179.72	1396.40	1628.88	1877.52	2140.80
45	993.84	1195.35	1411.88	1649.55	1900.39	2165.94
50	976.95	1175.91	1390.44	1621.30	1867.19	2128.35
55	931.29	1120.59	1325.38	1545.08	1779.07	2027.56
60	857.94	1032.20	1220.72	1423.01	1638.49	1866.62
65	758.38	912.52	1079.35	1258.39	1448.63	1650.64
70	635.84	765.00	905.20	1055.31	1215.76	1385.29
75	494.04	594.48	703.54	820.69	945.27	1077.31
80	336.97	405.68	480.11	560.28	645.61	735.83

3.8 Analog Simulation

The analog simulation in general, is the simulation performed by using the analog models of the system. There are many examples of physical analog models, where the properties of the system are represented by a set of another properties, and an analogy between the two is established. However, when we talk of the system simulation, the simulation carried on by employing the analog computers is called the analog system simulation. Simulation with an analog computer is based on a mathematical model, than being on a physical model. The electronic analog computers came into existence much before the digital computers, and were extensively used for simulating the engineering systems, which were too complex to be handled by the analytical techniques.

In analog simulation, various mathematical functions and operations are represented by some hardware devices whose behavior is equivalent to the mathematical functions like addition, multiplication or integration. The most widely used form of analog computer is the electronic analog computer, based on the use of high gain direct current amplifiers, called operational amplifiers.

Voltages in the computer are equated to mathematical variables, and the operational amplifiers can add or integrate the voltages. In each simulation, depending upon the system being simulated, some amount of hardware devices is required. These hardware elements are interconnected to imitate the system. The elements performing mathematical operations like integration, multiplication, addition, subtraction, square root etc. can be expressed as blocks, which are like subroutines. Any continuous system, can be simulated by suitably combining together a number of blocks. Different systems will have different number of blocks and their interconnections. Since, the same blocks can be organized in different ways, it is advantageous to have packages of standard subroutines to perform the operation of blocks. Hence, it is more convenient to have a special purpose block oriented programming language. A large number of such continuous system simulation languages have been designed and implemented. However, the discussion of these languages, is beyond the scope of this book.

3.9 Disadvantages of Analog Simulation

Though analog computers came into existence much before the digital computers, and have played a major role in the simulation of continuous dynamic systems, but they are giving way to digital computers at a fast pace. Following are some of the important disadvantages of analog computers.

1. *Limited accuracy*: The results obtained from analog simulation has limited accuracy, while much more accurate results can be obtained by employing digital simulation. Thus, where, high accuracy in results is required, as in space vehicles, guided missiles and fusion, the analog simulation cannot be used.
2. *Magnitude scaling*: In analog simulation, values of various variables are represented by voltages, which have a fixed and limited range. The variables must remain within the limited range, otherwise the results become inaccurate. Thus, all program variables have to be scaled, so that none exceeds the voltage range. This is a very difficult task, especially when the number of variables is large, and the range of their variation is not known. On the other hand, in digital computers, the problem of magnitude scaling does not arise, as they have a very large range. With floating point arithmetic, they have a very large precision. Hence, no magnitude scaling is normally required in digital simulation.
3. *Hardware set up required*: In analog simulation, hardware elements have to be combined to simulate the system, they have to be tested and calibrated while no such set up is required in digital simulation. Switching from one simulation to other takes time in analog simulation, while it requires no time in digital simulation. A simulation program on a digital computer can be easily stored, for reuse.

Analog simulation is considered to have two advantages over digital simulation. One is higher speed of solution, and second is immediate display of simulation results. However, these two advantages are also now becoming historical with the advent of super speed digital computers having on-line CRT displays.

3.10 Exercises

1. Solve the following employing continuous simulation

$$I = \int_0^{\pi} \sin x dx.$$

2. A bullet is fired at an angle of 40° with horizontal and with an initial velocity of 350 meter/sec. Assuming that the air resistance varies as the square of the velocity of the bullet and that the resistance coefficient is –0.00005, find the range, time of flight, and the angle of fall of the bullet.

3. Simulate the Pure Pursuit Problem (Fighter Bomber Problem). The initial position of the fighter is (0, 50) and the bomber in following the path :

Time (t) :	0	1	2	3	4	5	6	7	8	9	10	11	12
$xb(t)$:	80	90	99	108	116	125	133	141	151	160	169	179	180
$yb(t)$:	0	−2	−5	−9	−15	−18	−23	−29	−28	−25	−21	−20	−17

Assume the speed of the fighter as constant at 20 km/min. Calculate whether the fighter will hit the bomber (the time for chase is 10 min). The range of the fighter to hit the bomber is 10 km. Specify any other assumptions you made. **[PTU, B.Tech (Elect.) 3rd Sem. 2002]**

4. A cannon fires spherical balls of mass m, a range table that gives ranges for various values of muzzle velocity and gun elevation (firing angle) is to be produced. The sensitivity of range to small changes in muzzle velocity and gun elevation is also to be investigated. Assume that the drag is proportional to the square of the instantaneous velocity of the cannon ball, and it is purely along the direction of flight. Making use of the following four equations, write a computer program to produce the range table:

$$m\frac{dv}{dt} + mg \sin \theta + cv^2 = 0$$

$$mv\frac{d\theta}{dt} + mg \cos \theta = 0$$

$$\frac{dx}{dt} = v \cos \theta$$

$$\frac{dy}{dt} = v \sin \theta$$

Constant c is the drag coefficient for the cannon ball. Variables v and θ are the instantaneous velocity and angle of elevation of the cannon ball, and x, y are the coordinates of its instantaneous position. For various values of the starting conditions, i.e., v_0, θ_0, and $x_0 = y_0 = 0$, the program should yield the range.

5. A missile is fired vertically up. It has a rocket motor that produces a certain constant upward thrust F for as long as the fuel-burning motor is on. The motor burns the fuel at a constant rate b. Assume that the missile does not go so high as to vary the gravity constant g nor does the drag coefficient vary during the flight. Set up the equation of the vertical motion. Write a program, which gives the vertical distance y at any time t.

6. Suppose a 50-kg heat-seeking missile is fired at a jet bomber flying at a constant speed of 1,500 km/hour. The missile has a rocked motor that produces a thrust of 6000 kg for a period of 5 minutes. Simulate the missile system to test its effectiveness and to estimate how close the target should be before a missile is fired. Assume the drag coefficient $c = .04$ kg/km^2 and the weight of liquid fuel is 20 kg which is burned in 5 minutes at a constant rate ?

7. In the pure pursuit problem of Section 3.2, plot the path traced by the target and the pursuer. What should be the speed of the fighter to ensure that the bomber is hit in 8 minutes? Determine this value by selecting different values of the time increment. How do this effect the results?

8. In the pure pursuit problem of Section 3.2, at time zero the (x, y) coordinates of the bomber (target) are (0, 0) and of the fighter are $(0, d)$. The target moves in a straight line along the x-axis at constant speed of Vt. The fighter moves at a velocity of Vf and continuously corrects its direction to point in the direction of its target. In this simple case the path traced by the fighter, can be described by the equation,

$$x = \frac{1}{2}\left[\frac{y^{1+r}}{(1+r)\,dr} - \frac{y^{1-r}}{(1-r)\,d^{-r}}\right] + \frac{rd}{1-r^2}, \qquad \text{where } r = \frac{vt}{vf}$$

First trace the path of the fighter using the simulation technique and then the above analytic expression. Discuss if there are any differences between the two results.

9. Design a three-dimensional pure pursuit problem, and develop a simulation model for the same.

10. In the serial chase problem of Section 3.6, the initial locations of the objects A, B, C and D are (0, 0), (0, 10), (0, 20) and (0, 30) respectively and their velocities are 30 km/hr, 25 km/hr, 20 km/hr and 15 km/hr respectively. Object D moves towards a fixed target at (30, 50), while C moves always in a direction pointing towards D, The object B moves always pointing towards C and object A always pointing towards B. Which object will hit its target first and in what time?

11. An environment consists of two populations, the predators and prey, which interact with each other. The prey are passive, but the predators depend upon prey as their source of food, let $x(t)$ and $y(t)$ denote, respectively, the numbers of individuals in the prey and predator populations at time t. There is an ample supply of food for the prey and in the absence of predators, their growth rate is $rx(t)$ for some positive r. The death rate of the prey due to interaction with predator can be assumed to be proportioned to the product of two population sizes, $x(t)$, $y(t)$. Thus, the overall rate of change of the prey population is given by,

$$\frac{dx}{dt} = rx(t) - ax(t)\,y(t)$$

where a is a positive constant of proportionality.
The predators depend upon prey, and in the absence of prey, their rate of change is $-sy(t)$ for some positive value of s. Due to interaction between the populations, the predators increase at a rate, which is proportionate to $x(t)\,y(t)$. Thus, overall rate of change of the predator population is,

$$\frac{dy}{dt} = -sy(t) + bx(t)\,y(t)$$

where b is a positive constant.

Considering the following values for constants and the initial starting conditions, plot the growth of two populations with time, for a period of 5000 time units.

Constants:

$r = .001$ $\quad a = 2 \times 10^{-6}$ $\quad s = 0.01$ $\quad b = 10^{-6}$

Initial conditions;

$x(0) = 12{,}000$ $\quad y(0) = 600.$

12. In the chemical reaction simulation of Section 3.3, take the value of delta as 0.05 for time up to 2 minutes and 0.15 for time up to 4 minutes. In the second run, the corresponding values of delta are 0.1 and 0.2. Run the simulations and discuss if there are any differences in the results.