Aim : To solve the equation of radioactive decay using fourth order Runge-Kutta method of solving first order ordinary differential equation.

Theory :

Equation of Radioactive Decay

Suppose a radioactive substance has X_0 number of initial atoms at time $(t=0)$ and there are X atoms at any instant t (ie. $X_0 - X$ number of atoms have disintegrated.

If dN atoms disintegrate in time dt, then the decay equation has the form

$$\frac{dX}{dt} = -\lambda X \qquad [\lambda \text{ being the decay constant}]$$

The solution to this equation is

$$X = X_0 e^{-\lambda t}$$

Runge-Kutta Method (4^{th} order)

To solve the first order ordinary differential equation

$$\frac{dX}{dt} = f(t, X)$$

at a time-point t_n using the intial condition ($X = X_0$ at $t = t_0$) we divide the interval (t_0, t_n) into n number of steps with step-size h.

i.e. $h = \frac{t_n - t_0}{n}$

To calculate X_n at t_n we calculate the following for all steps successively till $m+1 = n$

$$X_{m+1} = X_m + \frac{h}{6}\left[k_{1(m+1)} + 2k_{2(m+1)} + 2k_{3(m+1)} + k_{4(m+1)}\right]$$

where $t_{m+1} = t_m + h \qquad m = 0, 1, \ldots, m$

$$k_{1(m+1)} = f(t_m, X_m)$$

$$k_{2(m+1)} = f\left(t_m + \frac{h}{2}, X_m + k_{1(m+1)}\frac{h}{2}\right)$$

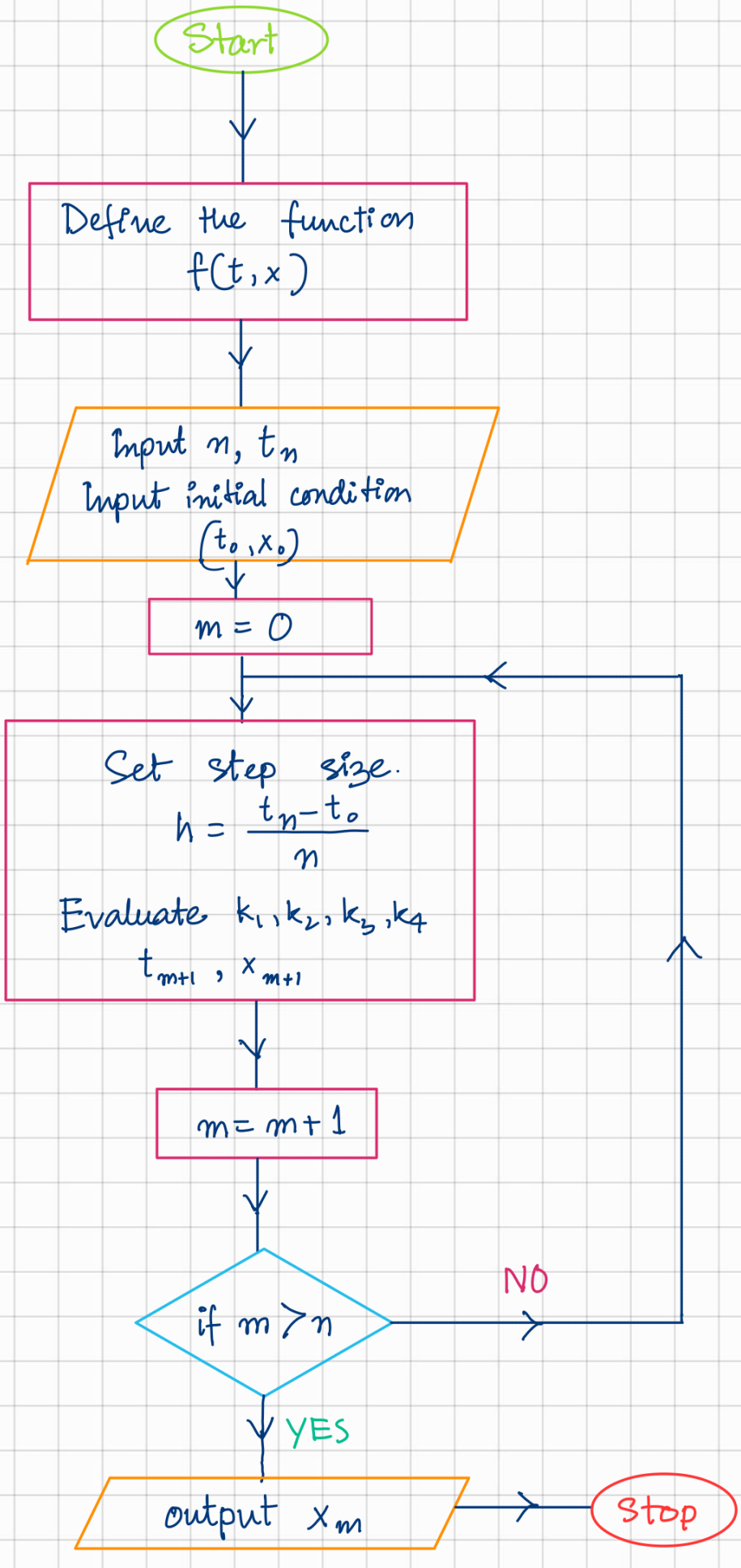
$$k_{3(m+1)} = f\left(t_m + \frac{h}{2}, X_m + k_{2(m+1)}\frac{h}{2}\right)$$

$$k_{4(m+1)} = f\left(t_m + h, X_m + k_{3(m+1)}\, h\right)$$

For the radioactive decay equation,

$$f(t, X) = -\lambda X$$

Flowchart :


Start
Define the function
f(t, x)
Input n, t_n
Input initial condition
(t_0, x_0)
m = 0
Set step size.
$h = \frac{t_n - t_0}{n}$
Evaluate k_1, k_2, k_3, k_4
t_{m+1} , x_{m+1}
m = m + 1
if m > n
NO
YES
output x_m
Stop

```c
#include <stdio.h>
#include <math.h>


float fn(float t, float x)
{
    float f;
    f = -1 * (2 * x);
    return(f);
}


int main()
{
    int m,n,i,j;
    float lambda, t0, x0, tn, xn, h, result ;


    printf("\n \nRunge-Kutta Diff. Eqn. Solver for Radioactive Decay Equation ");

    printf("\n \nEnter the decay constant lambda \n");
    printf("lambda: ");
    scanf("%f",&lambda);

    float fn(float t, float x)
    {
        float f;
        f = -1 * (lambda * x);
        return(f);
    }

    printf("\n \nEnter the initial condition of the form x(t0)=x0 \n");
    printf("t0: ");
    scanf("%f",&t0);
    printf("x0: ");
    scanf("%f",&x0);


    printf("\n \nEnter t where you want to find out x(t)  \n");
    printf("tn: ");
    scanf("%f",&tn);


    printf("\n \nEnter no. of steps n in the interval (t0,tn) \n");
    printf("n: ");
    scanf("%d",&n);


    h = (tn - t0)/(n);
    printf("\n \nstep-size h = %f \n \n",h);

    float t[n], x[n], k1[n], k2[n], k3[n], k4[n];

    t[0]=t0;
    x[0]=x0;


    for ( m = 0; m < n; ++m)
    {
```

```
        printf("\n \n  ---Step %d--- \n \n",m+1);
        printf(" t%d = %f \n",m,t[m]);
        printf(" x%d = %f \n",m,x[m]);


        k1[m+1] = fn( t[m]  ,  x[m] );
        printf("\nk1%d = %f",m+1, k1[m+1]);


        k2[m+1] = fn( t[m]+(h/2)  ,  x[m]+k1[m+1]*(h/2) );
        printf("\nk2%d = %f",m+1, k2[m+1]);


        k3[m+1] = fn( t[m]+(h/2)  ,  x[m]+k2[m+1]*(h/2) );
        printf("\nk3%d = %f",m+1, k3[m+1]);


        k4[m+1] = fn( t[m]+h  ,  x[m]+k3[m+1]*h  );
        printf("\nk4%d = %f",m+1, k4[m+1]);

        t[m+1]  = t[m]+h;
        printf("\n \n t%d = %f \n",m+1, t[m+1]);

        x[m+1]  = (x[m])+ ((h/6) * (k1[m+1] + 2*k2[m+1] + 2*k3[m+1] + k4[m+1]));
        printf(" x%d = %f \n",m+1, x[m+1]);

        result = x[m+1];

    }


    printf("\n \nThe result is : x(%f) = %f \n \n",tn,result);
    return 0;
}
```

Observations : Decay constant , λ =

Initial condition , t_0 =

x_0 =

Evaluation at , t_n =

steps, 'n'	x_n according to analytic solution $X = x_0 e^{-\lambda t}$	X_n according to Runge-Kutta method
10		
100		
1000		

Result :- Using 4th order Runge-Kutta method,

x_n =

Discussion :-

1. The accuracy increases as number of steps increases.