Project C10

# ENUME

Mazen Ibrahim

295924

## Table of Contents

# Approximation

## Theory

The purpose of approximation is to find a simpler function close to the original function. This is achieved by finding the coefficients of the approximated function **F** such that the norm ||f-F||is minimized where **f** is the original function.

### Discrete least-squares approximation

Assume that for a given finite number of points **$x_0, x_1, \ldots, x_n$** ($x_i \neq x_j$) the values $y_j = f(x_j)$ , j = 0,1,2,…,N are known.

Let $\phi_i(x)$ , i = 0,1,..,n be a basis of a space $X_n \subseteq X$ of interpolating function

$$\forall F \in X_n \;\; F(x) = \sum_{i=0}^{n} a_i \varphi_i(x)$$

To find values of the parameters $a_0, a_1, \ldots, a_n$ defining the approximating function mentioned above which minimize the least-squares error defined by

$$H(a_0 a_1, \ldots, a_n) = \sum_{j=0}^{N} [f(x_j) - \sum_{i=0}^{n} a_i \phi_i\,(x_j)]\text{^}2$$

The weighting function is not present to simplify the presentation.
The formula for the coefficients $a0$, $a1$, … , $an$ can be derived from the necessary condition for a minimum)being here also the sufficient condition, as the function is convex

$$\frac{\partial H}{\partial a_k} = -2 \sum_{j=0}^{N} \left[ f(x_j) - \sum_{i=0}^{n} a_i \phi_i(x_j) \right] . \phi_k(x_j) = 0 \,, \quad k = 0,1..,n$$

The system of linear equations with the unknowns $a0$, $a1$, … , $an$
Is called the *set of normal equations*, and its matrix is known as **the Gram's matrix.**

The performance function of the approximation problem is written as
H(a) = (|||$y - Aa$||2)^2. Where **A** is our gram matrix and **a** is the transposed coefficients that were found and **y** is the set transposed values that were given to me in the task.

Therefore, the problem of the least-squares approximation is a linear least- squares problem(LLSP),

**Note:**

All columns of the matrix A are linearly independent(which follows from linear independence of the basis functions).Hence, the matrix has full rank.
The set of normal equations can be written in the following form:
$A^T A a = A^T y$.

Since, the matrix A has full rank, then the Gram's matrix $A^T A$ is nonsingular. This implies uniqueness of the solution of the set of normal equations. But, even being nonsingular, the matrix $A^T A$ can be badly conditioned – its condition number is a square of the condition number of **A**, that's why it is recommended to solve the approximation problem using the method based on the QR factorization of A.

## Results

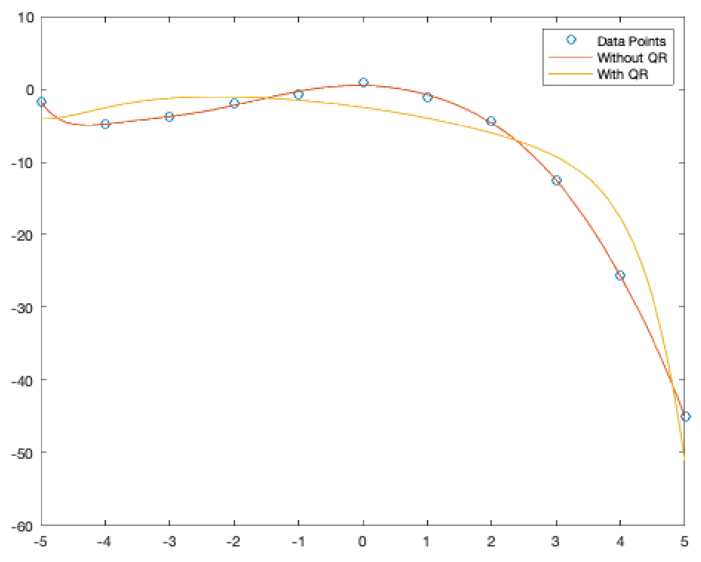


Figure 1

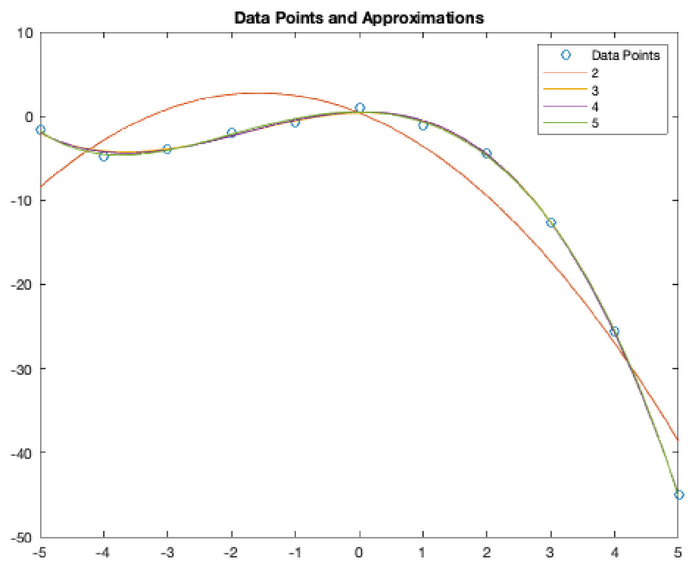


Figure 2

| Order | Without QR | With QR |
|---|---|---|
| 1 | 44.5891 | 3.55E-15 |
| 2 | 31.3988 | 5.02E-15 |
| 3 | 14.106 | 5.02E-15 |
| 4 | 1.1297 | 7.94E-15 |
| 5 | 1.0857 | 3.55E-15 |
| 6 | 0.9059 | 6.15E-15 |
| 7 | 0.8342 | 5.02E-15 |
| 8 | 0.8285 | 5.02E-15 |
| 9 | 0.7883 | 8.70E-15 |
| 10 | 0.7721 | 1.12E-14 |

Where each row corresponds to the power of the x row one is x^0 row two is x^1 and so on

## Conclusions

As we can see with QR decomposition we get much lower errors  but it is more time demanding while without QR it is much quicker .

# Solving Differential equations

## Theory

Usually systems of differential equations describing real world problems are nonlinear and analytic solutions are not possible to obtain and this where numerical methods become useful as they help to obtain those solutions. Now to solve ODEs we have two types of methods one is **single step** and the other is **multistep**. We will be investigating **RK4** as **single step** method and **Adams predicator-correct method** as **multistep** method.

### RK4

Family of Runge-Kutta methods are defined by the following formula

$$y_{n+1} = y_n + h \cdot \sum_{i=1}^{m} w_i k_i,$$

where

$$\begin{aligned} k_1 &= f\left(x_n, y_n\right), \\ k_i &= f(x_n + c_i h, y_n + h \cdot \sum_{j=1}^{i-1} a_{ij} k_j), \quad i = 2, 3, ..., m, \end{aligned}$$

and also

$$\sum_{j=1}^{i-1} a_{ij} = c_i, \quad i = 2, 3, ..., m.$$

And in order to perform a single iteration the values on the right side of equations should be calculated **m** times. The parameters **$w_i$,$a_{ij}$,$c_i$** are not unique. The coefficients are often chosen in a way to assure a higher order of the method for a given m. if p(m) denotes a maximal possible order of RK method

$$\begin{aligned} &p\left(m\right) = m && \text{for } m = 1, 2, 3, 4, \\ &p\left(m\right) = m - 1 && \text{for } m = 5, 6, 7, \\ &p\left(m\right) \le m - 2 && \text{for } m \ge 8. \end{aligned}$$

in my project I will be using method with m = 4 of order p = 4 because they are balanced in terms of accuracy of approximation and number of iterations

This is how the algorithm looks like for RK4 (m= 4 ,p=4)

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{6} h \left(k_1 + 2k_2 + 2k_3 + k_4\right), \\ k_1 &= f\left(x_n, y_n\right), \\ k_2 &= f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1), \\ k_3 &= f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2), \\ k_4 &= f\left(x_n + h, y_n + hk_3\right). \end{aligned}$$

And this is how RK4 looks like based on the equations I was given in my task

$k_{1,1} = x_2 + x_1(0.5 - x_1^2 - x_2^2)$
$k_{1,2} = -x_1 + x_2(0.5 - x_1^2 - x_2^2)$

For $k_2$
$x_1 = x_1 + ½ * h * k_{1,1}$
$x_2 = x_2 + ½ * h * k_{1,2}$

$k_{2,1} = x_2 + x_1(0.5 - x_1^2 - x_2^2)$
$k_{2,2} = -x_1 + x_2(0.5 - x_1^2 - x_2^2)$

For $k_3$
$x_1 = x_1 + ½ * h * k_{2,1}$
$x_2 = x_2 + ½ * h * k_{2,2}$

$k_{3,1} = x_2 + x_1(0.5 - x_1^2 - x_2^2)$
$k_{3,2} = -x_1 + x_2(0.5 - x_1^2 - x_2^2)$

For $k_4$
$x_1 = x_1 + h * k_{3,1}$
$x_2 = x_2 + h * k_{3,2}$

$k_{4,1} = x_2 + x_1(0.5 - x_1^2 - x_2^2)$
$k_{4,2} = -x_1 + x_2(0.5 - x_1^2 - x_2^2)$

$(x_1)_{n+1} = (x_1)_n + (⅙)h_n(k_{1,1} + 2k_{2,1} + 2k_{3,1} + k_{4,1})$
$(x_2)_{n+1} = (x_2)_n + (⅙)h_n(k_{1,2} + 2k_{2,2} + 2k_{3,2} + k_{4,2})$

The crucial part of the single step method is determining the value of step size **h** choosing small **h** results in lower errors but using  excessively small **h** results in more athematic operations which eventually affects the roundoff errors as well so choosing optimal **h** step size is necessary to achieve the best results.

## Adams PC (P5EC5E)

Adams methods are based on the idea of approximating the integrand with a polynomial within the interval ($t_n$, $t_{n+1}$). Using a *k*th order polynomial results in a *k*+1th order method. There are two types of Adams methods, the explicit and the implicit types. The explicit type is called the Adams-Bashforth (AB) methods and the implicit type is called the Adams-Moulton (AM) methods.

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} \frac{dy}{dt} dt = y_n + \int_{t_n}^{t_{n+1}} f(y,t) dt.$$

Now considering the following formula

$$y_n = y_{n-1} + h \sum_{j=1}^{k} \beta_j f(x_{n-j}, y_{n-j}),$$

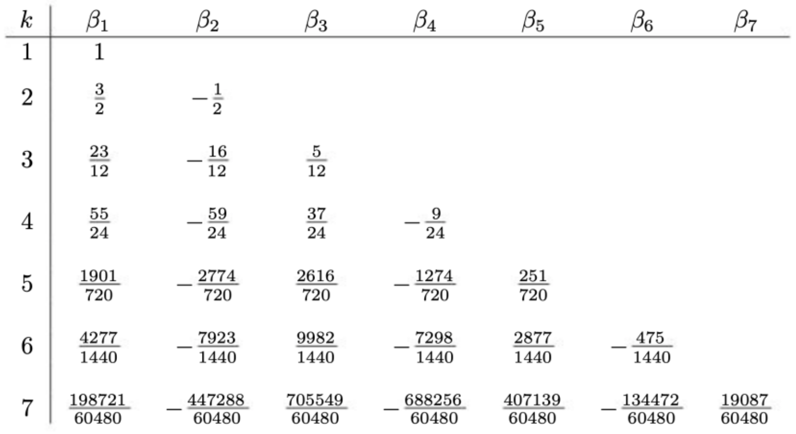
If $\beta_0 = 0$ then we are dealing with explicit Adams and the value $y_n$ will depend explicitly on the values of the solution y and its derivative f(x,y) taken at previously calculated points meaning $y_n$ will depend on $y_{n-1}, \dots, y_{n-k}$. if $\beta_0 \neq 0$ then we are dealing with implicit Adams and $y_n$ depend on $y_{n-1}, \dots, y_{n-k}$ and $f$ ,...,$f$ ,where$f$ =f($x$,$y$),but also on$f$ =f($x$ ,$y$ ).

Now one of the practical implementation of multistep methods is the PC (predicator-corrector) algorithm which combines both explicit and implicit Adams methods according to the following form

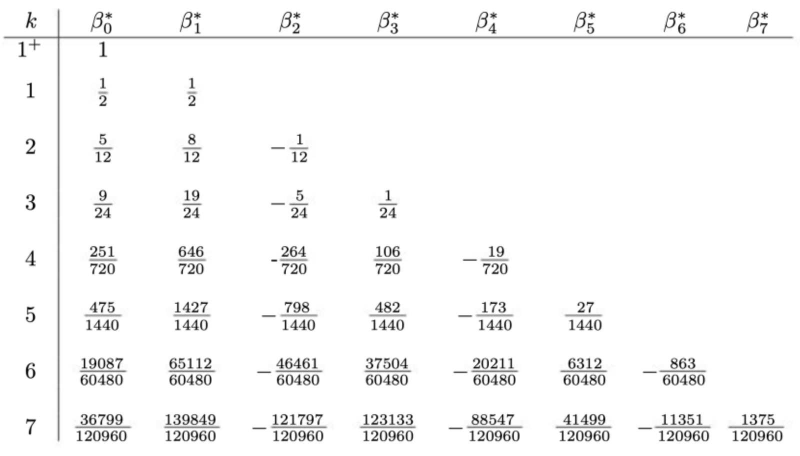
$$\begin{array}{lll}
\text{P:} & y_n^{[0]} = \sum\limits_{j=1}^{k} \alpha_i y_{n-j} + h \sum\limits_{j=1}^{k} \beta_j f_{n-j}\,, & \text{(P – prediction)} \\
\text{E:} & f_n^{[0]} = f(x_n, y_n^{[0]}), & \text{(E – evaluation)} \\
\text{C:} & y_n = \sum\limits_{j=1}^{k} \alpha_j^* y_{n-j} + h \sum\limits_{j=1}^{k} \beta_j^* f_{n-j} + h\beta_0^* f_n^{[0]}\,, & \text{(C – correction)} \\
\text{E:} & f_n = f\left(x_n, y_n\right). & \text{(E – evaluation)}
\end{array}$$

And we will be using the following tables to pick our beta values at k = 5 for explicit and implicit methods

Table 7.6. Parameters of the explicit Adams methods (Adams-Bashforth methods)

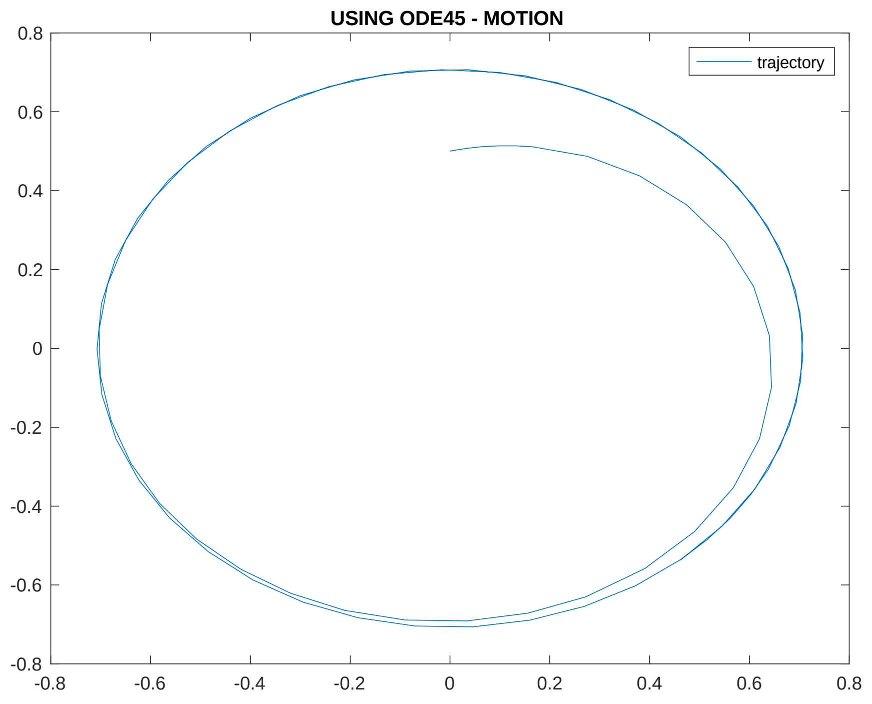


| $k$ | $\beta_1$ | $\beta_2$ | $\beta_3$ | $\beta_4$ | $\beta_5$ | $\beta_6$ | $\beta_7$ |
|---|---|---|---|---|---|---|---|
| 1 | $1$ | | | | | | |
| 2 | $\frac{3}{2}$ | $-\frac{1}{2}$ | | | | | |
| 3 | $\frac{23}{12}$ | $-\frac{16}{12}$ | $\frac{5}{12}$ | | | | |
| 4 | $\frac{55}{24}$ | $-\frac{59}{24}$ | $\frac{37}{24}$ | $-\frac{9}{24}$ | | | |
| 5 | $\frac{1901}{720}$ | $-\frac{2774}{720}$ | $\frac{2616}{720}$ | $-\frac{1274}{720}$ | $\frac{251}{720}$ | | |
| 6 | $\frac{4277}{1440}$ | $-\frac{7923}{1440}$ | $\frac{9982}{1440}$ | $-\frac{7298}{1440}$ | $\frac{2877}{1440}$ | $-\frac{475}{1440}$ | |
| 7 | $\frac{198721}{60480}$ | $-\frac{447288}{60480}$ | $\frac{705549}{60480}$ | $-\frac{688256}{60480}$ | $\frac{407139}{60480}$ | $-\frac{134472}{60480}$ | $\frac{19087}{60480}$ |

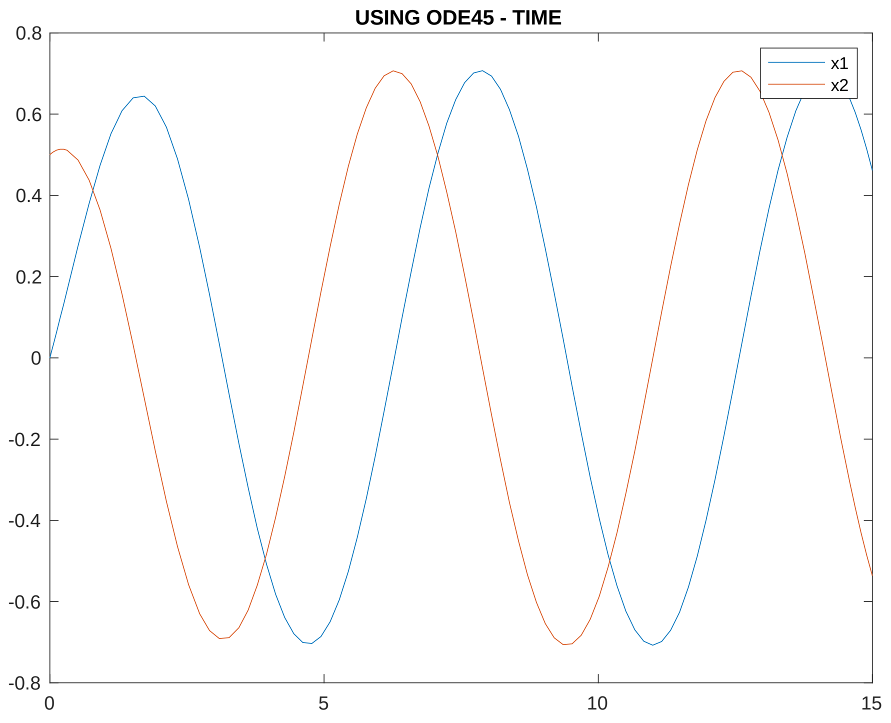


| $k$ | $\beta_0^*$ | $\beta_1^*$ | $\beta_2^*$ | $\beta_3^*$ | $\beta_4^*$ | $\beta_5^*$ | $\beta_6^*$ | $\beta_7^*$ |
|---|---|---|---|---|---|---|---|---|
| $1^+$ | $1$ | | | | | | | |
| 1 | $\frac{1}{2}$ | $\frac{1}{2}$ | | | | | | |
| 2 | $\frac{5}{12}$ | $\frac{8}{12}$ | $-\frac{1}{12}$ | | | | | |
| 3 | $\frac{9}{24}$ | $\frac{19}{24}$ | $-\frac{5}{24}$ | $\frac{1}{24}$ | | | | |
| 4 | $\frac{251}{720}$ | $\frac{646}{720}$ | $-\frac{264}{720}$ | $\frac{106}{720}$ | $-\frac{19}{720}$ | | | |
| 5 | $\frac{475}{1440}$ | $\frac{1427}{1440}$ | $-\frac{798}{1440}$ | $\frac{482}{1440}$ | $-\frac{173}{1440}$ | $\frac{27}{1440}$ | | |
| 6 | $\frac{19087}{60480}$ | $\frac{65112}{60480}$ | $-\frac{46461}{60480}$ | $\frac{37504}{60480}$ | $-\frac{20211}{60480}$ | $\frac{6312}{60480}$ | $-\frac{863}{60480}$ | |
| 7 | $\frac{36799}{120960}$ | $\frac{139849}{120960}$ | $-\frac{121797}{120960}$ | $\frac{123133}{120960}$ | $-\frac{88547}{120960}$ | $\frac{41499}{120960}$ | $-\frac{11351}{120960}$ | $\frac{1375}{120960}$ |

## Results

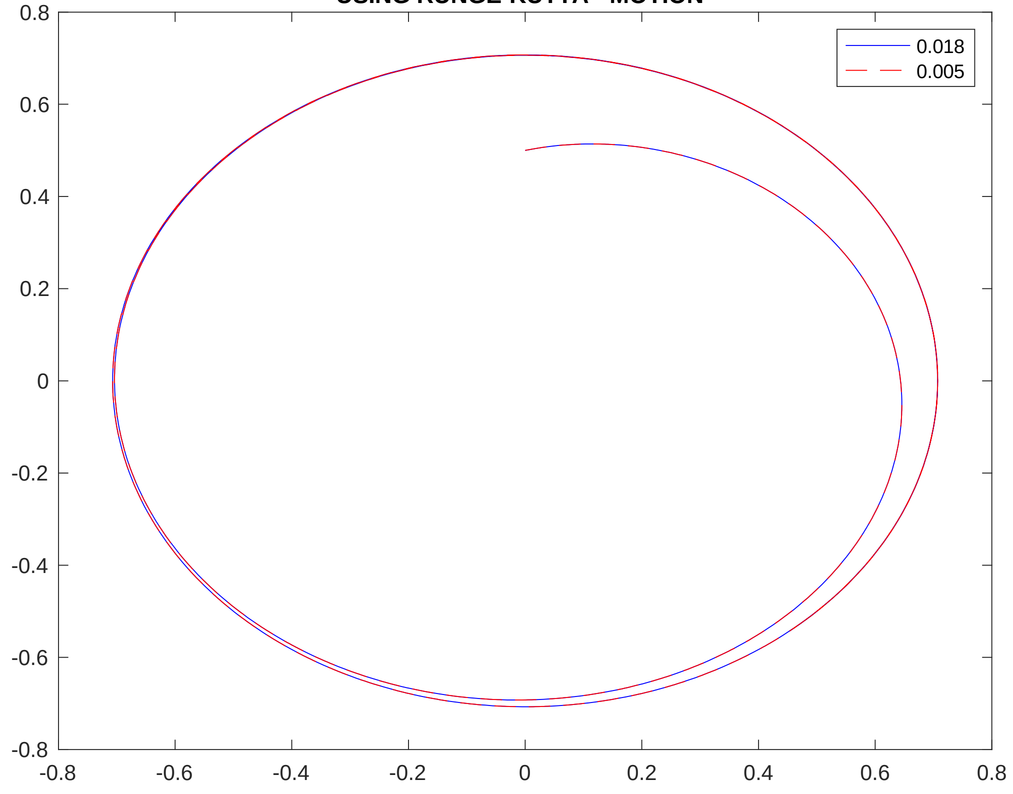


USING ODE45 - MOTION
trajectory
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8

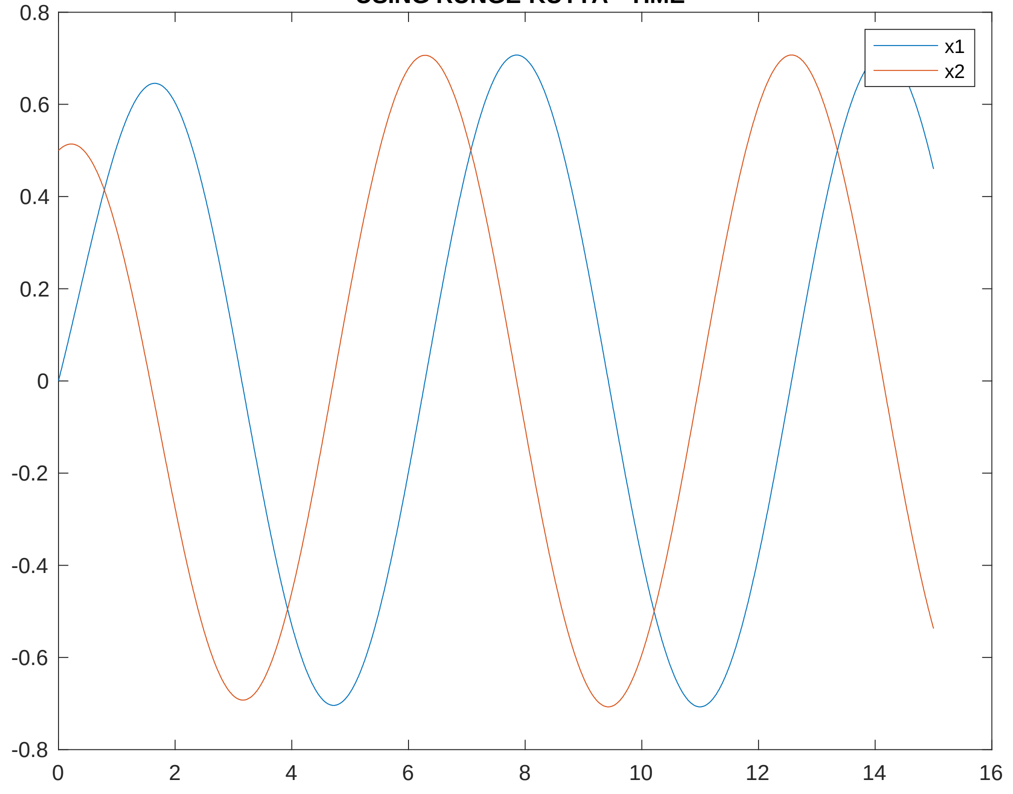


USING ODE45 - TIME
x1
x2
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
5
10
15

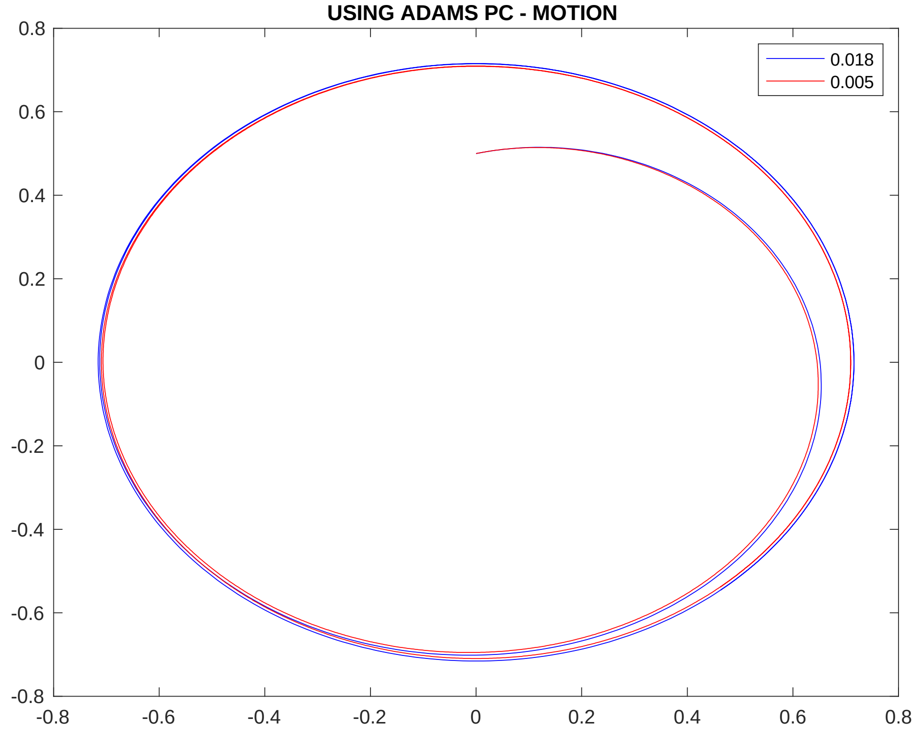
USING RUNGE-KUTTA - MOTION



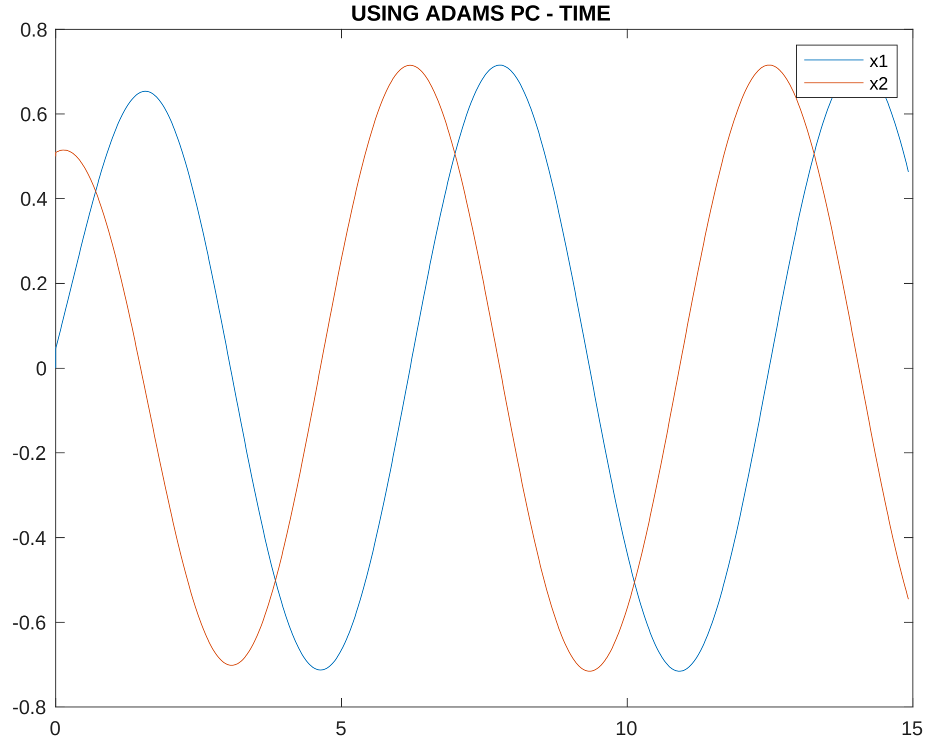
USING RUNGE-KUTTA - TIME



USING ADAMS PC - MOTION



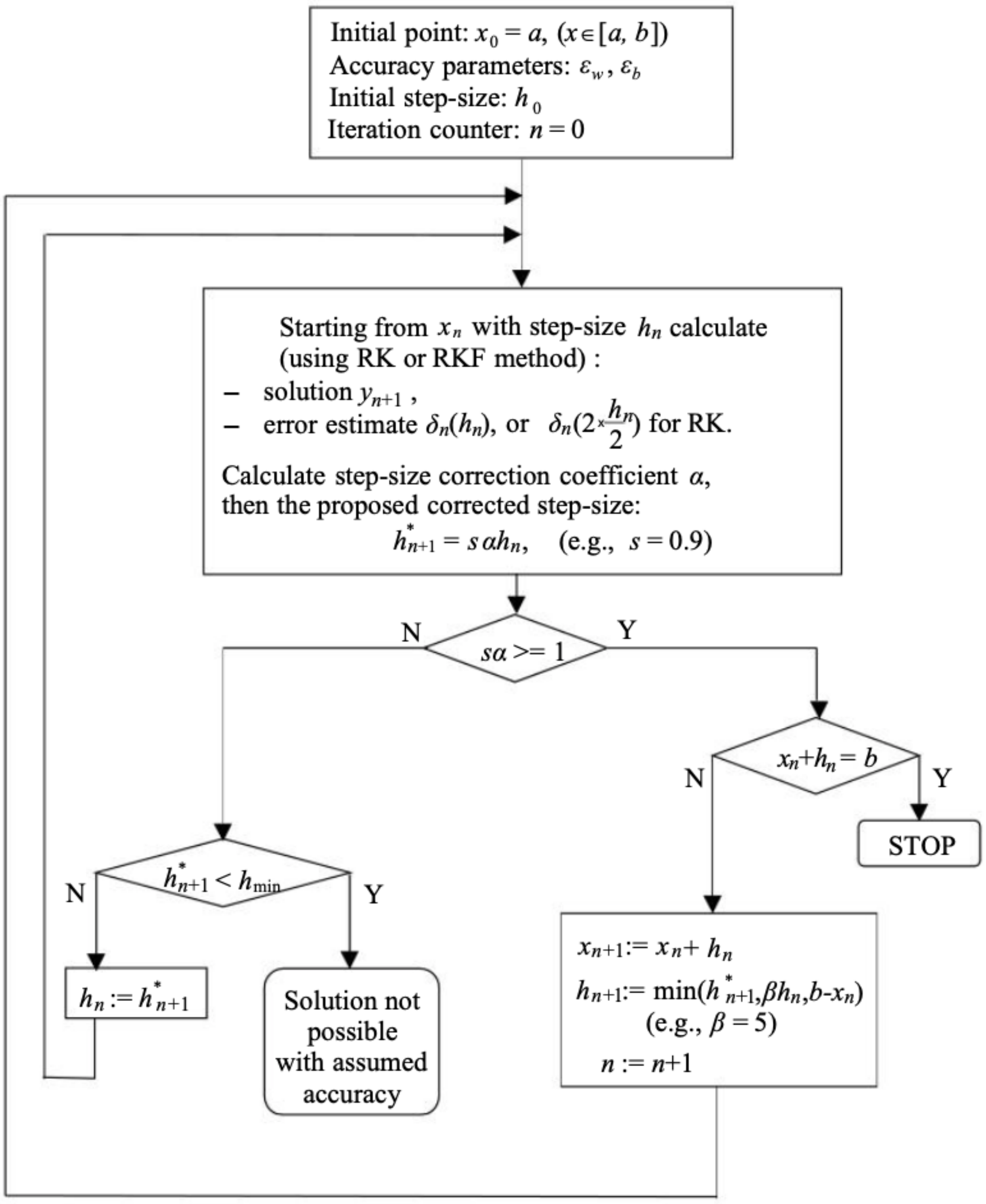
USING ADAMS PC - TIME



## Conclusions

As we can see both methods are reliable but Adams PC method seems to be more sensitive to step change compared to RK4.

## Task2b



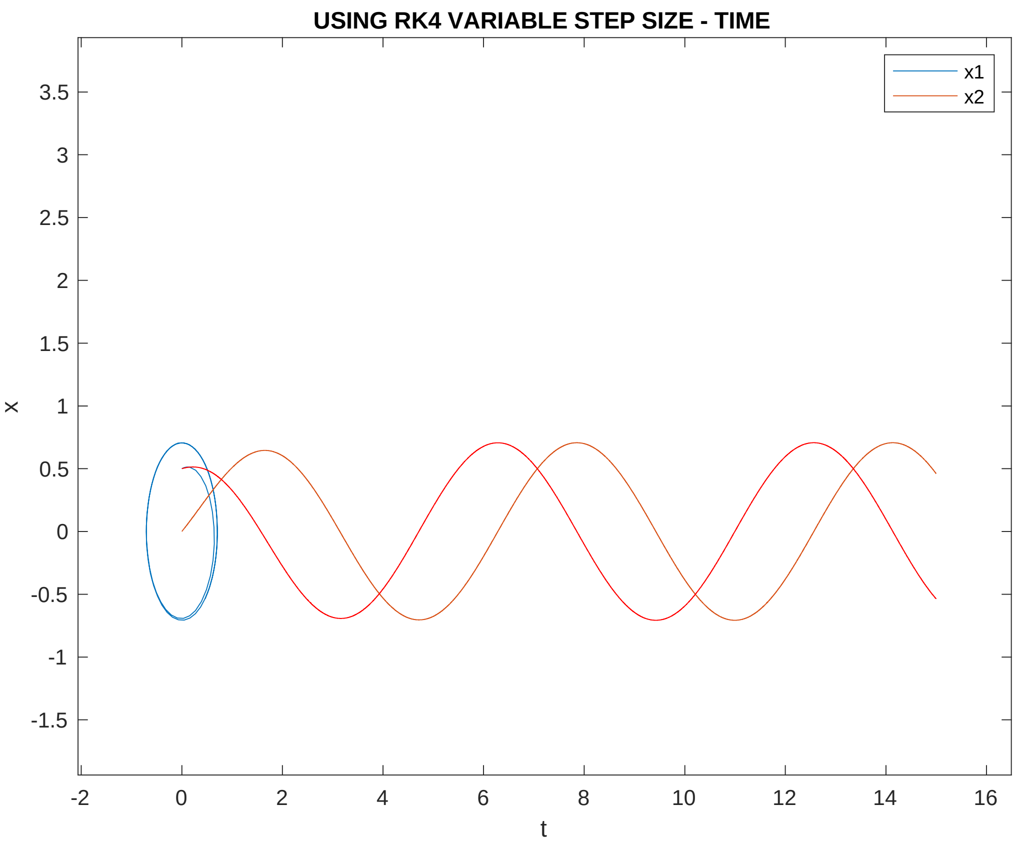
Initial point: $x_0 = a$, $(x \in [a, b])$
Accuracy parameters: $\varepsilon_w$, $\varepsilon_b$
Initial step-size: $h_0$
Iteration counter: $n = 0$
Starting from $x_n$ with step-size $h_n$ calculate (using RK or RKF method) :
– solution $y_{n+1}$ ,
– error estimate $\delta_n(h_n)$, or $\delta_n(2\times\frac{h_n}{2})$ for RK.
Calculate step-size correction coefficient $\alpha$, then the proposed corrected step-size:
$h^*_{n+1} = s\alpha h_n$, (e.g., $s = 0.9$)
$s\alpha >= 1$
N
Y
$x_n+h_n = b$
STOP
$h^*_{n+1} < h_{\min}$
$h_n := h^*_{n+1}$
Solution not possible with assumed accuracy
$x_{n+1} := x_n + h_n$
$h_{n+1} := \min(h^*_{n+1}, \beta h_n, b-x_n)$
(e.g., $\beta = 5$)
$n := n+1$

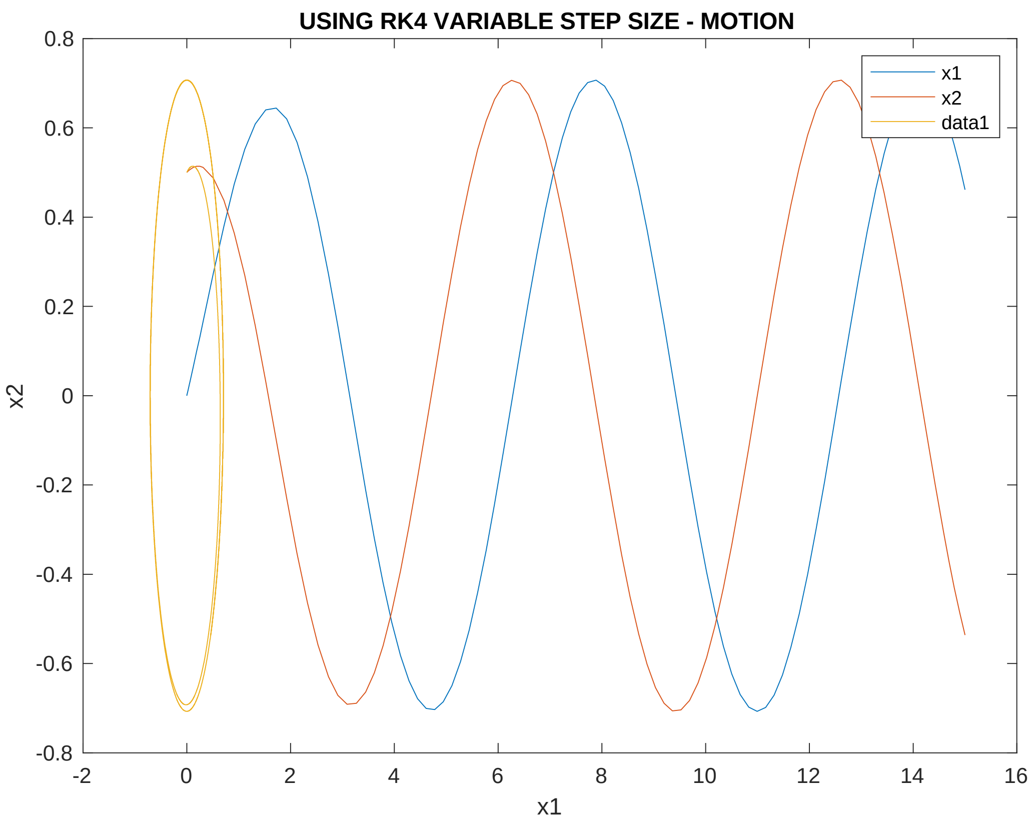
Now RK4 with variable step size according to the step-doubling rule is as follows

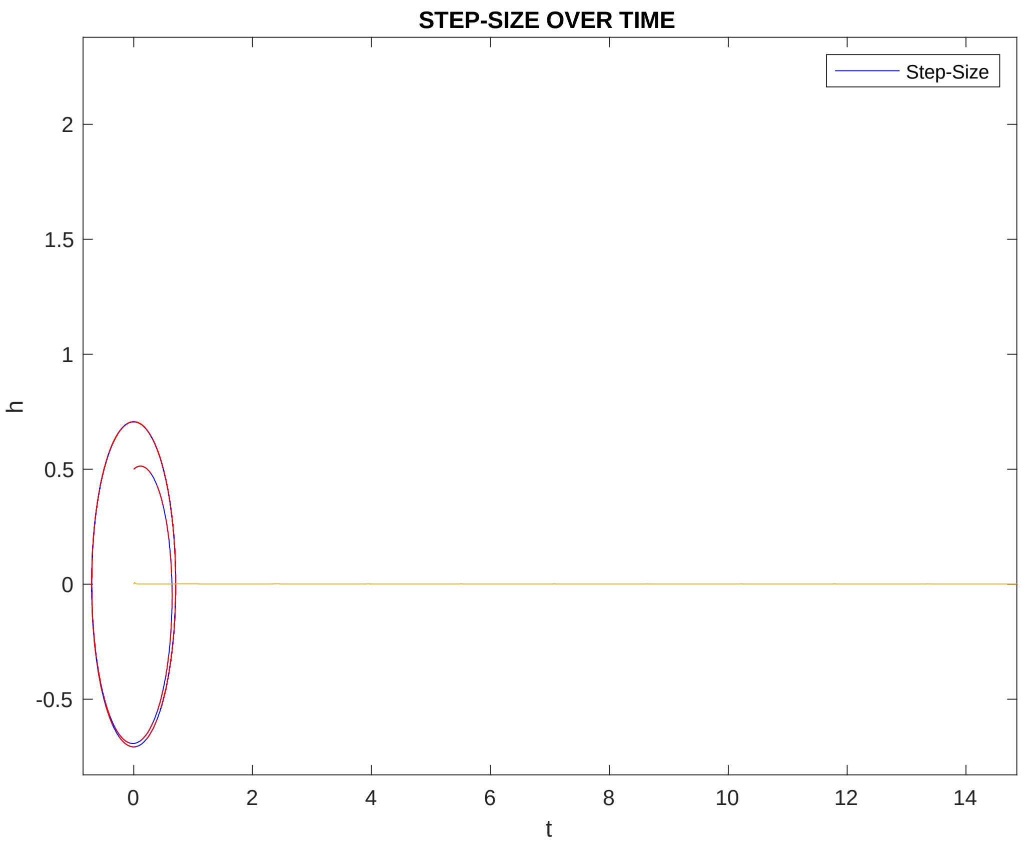
1) $H_{min}$ is selected by the user to obtain the required accuracy with low number of iterations
2) The absolute and relative tolerances are set by the user as well but they should be higher than $H_{min}$ otherwise the desired accuracy will not be achieved

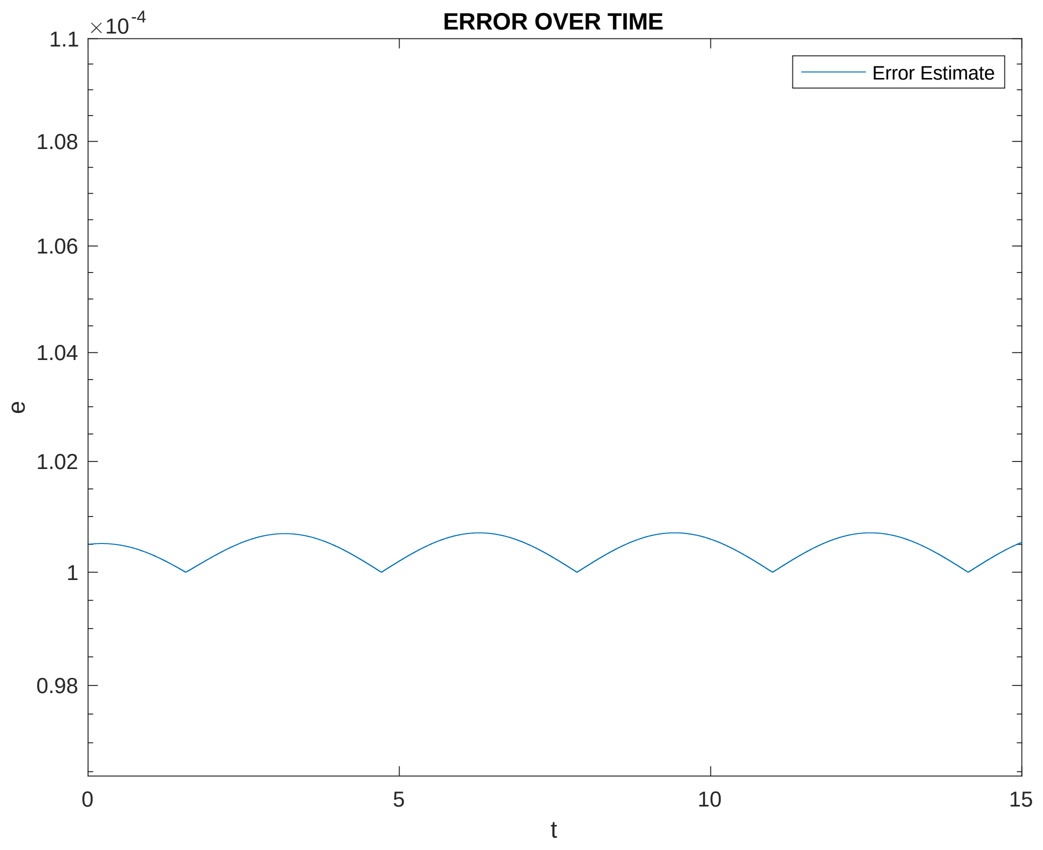
RK4 with variable step size has two stages one is full length and the second one is half length. If large steps keep the desired accuracy then its chosen, otherwise small steps will be taken.

## Results

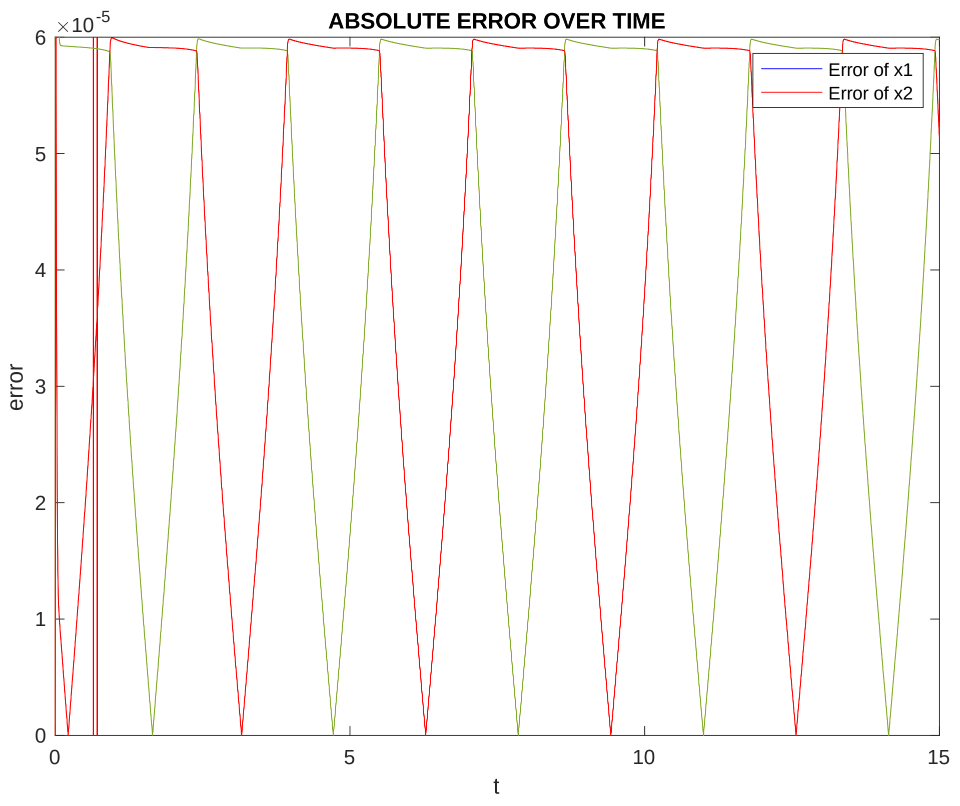



USING RK4 VARIABLE STEP SIZE - MOTION
x1
x2
data1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-2
0
2
4
6
8
10
12
14
16
x2
x1


STEP-SIZE OVER TIME
Step-Size
2
1.5
1
0.5
0
-0.5
0
2
4
6
8
10
12
14
h
t



ERROR OVER TIME
×10^-4
1.1
1.08
1.06
1.04
1.02
1
0.98
e
Error Estimate
0
5
10
15
t



ABSOLUTE ERROR OVER TIME
×10^-5
6
5
4
3
2
1
0
error
Error of x1
Error of x2
0
5
10
15
t

# Appendix

## Task 1

```
clear all;
clc;
x = [-5:5]';
y = [-1.6889 -4.7689 -3.8259 -2.0068 -0.6884 0.9391 -1.1556 -4.4293 -12.5746 -
25.6768 -45.0598]';
    H = [];
    HQR = [];
    A = [];
    maxDegree = 10;
    a = zeros(maxDegree);
    aQR = zeros(maxDegree);
    % Filling Gram matrix
    for i = 1:maxDegree
        A(:,i) = x.^(i-1);
    end
    % Without QR approx
    for i = 1:maxDegree
        tempA = A(:,1:i);
        linA = (tempA'* tempA);
        linb = (tempA' * y);
        coeff = linA\linb;
          a(1:i,i) = coeff';
        H(i) = (norm(y - A(:, 1:i)*coeff,2));
    end
    disp(H);
    disp(a);
    for i = 1:maxDegree
% find A for linear equation
        tempA = A(:, 1:i);
%find QR facotrization
       [Q, R] = qrfact(tempA);
        d = Q' * y;
%back substituaiton
        coeff = R^-1 * d;
        aQR(1:i, i) = coeff';
        HQR(i) = norm(R * coeff - d,2);

    end
    disp(HQR);
    disp(aQR);
x2 = [-5:0.1:5];
% QR vs without QR
figure();
plot(x, y, 'o', x2, polyval(fliplr(a(:, end)'), x2), x2, polyval(fliplr(aQR(:,
end)'), x2))
legend("Data Points", "Without QR", "With QR")
figure();
plot(x, y, 'o', x2, polyval(fliplr(aQR(:, 3)'), x2), x2, polyval(fliplr(aQR(:,
4)'), x2), x2, polyval(fliplr(aQR(:, 5)'), x2), x2, polyval(fliplr(aQR(:, 6)'),
x2))
title ("Data Points and Approximations")
legend("Data Points", "2", '3', '4', '5')
```

## QR function

```
function [Q, R] = qrfact(A)
[m, n] = size(A);
Q = zeros(m,n);
R = zeros(n,n);
d = zeros(1,n);
for i = 1:n
    Q(:,i) = A(:,i);
    R(i,i) = 1;
    d(i) = Q(:,i)' * Q(:,i);
    for j = i+1:n
    R(i,j) = (Q(:,i)' * A(:,j)) / d(i);
    A(:,j) = A(:,j) - R(i,j) * Q(:,1);
    end
end
for i = 1:n
    dtemp = norm(Q(:,i));
    Q(:,i) = Q(:,i) / dtemp;
    R(i,i:n) = R(i,i:n) * dtemp;
end
end
```

## Task 2a

```
%initialising
clc;clear;
%declaring the set of ordinary differential equations
f1 = @(t,x1,x2) (x2+(x1*(0.5-(x1)^2-(x2)^2)));
f2 = @(t,x1,x2) (-x1+(x2*(0.5-(x1)^2-(x2)^2)));
%initial conidtions
x0 = [0, 0.5];
a = 0;b = 15;
interval = [a, b];

%using ODE45---------------------------
func=@(t,y) [y(2)+y(1)*(0.5-y(1).^2-y(2).^2);-y(1)+y(2)*(0.5-y(1).^2-y(2).^2)];
[t, y] = ode45(func, interval, x0);

%plotting trajectory
figure(1);
plot(y(:,1),y(:,2));
hold on;
grid on;
title('USING ODE45 - MOTION');
legend ('trajectory');
%plotting time
figure(2);
plot(t,y);
hold on;
grid on;
title('USING ODE45 - TIME');
legend ('x1','x2');

%plotting for RK4 different constant step sizes------------------------
figure(3);
for h = [0.018,0.005]
    [x1,x2] = RK4CStep(a,b,h,f1,f2,x0(1),x0(2));

    switch h
        case 0.018
```

```
            plot(x1,x2,'b');
        case 0.005
            plot(x1,x2,'r--');
    end
    grid on;
    hold on;
end
legend('0.018', '0.005');
title('USING RUNGE-KUTTA - MOTION');
%plotting with time
figure(4);
[x1,x2, t] = RK4CStep(a,b,0.005,f1,f2,x0(1),x0(2));
plot(t,x1);
hold on;
grid on;
plot(t,x2);
title('USING RUNGE-KUTTA - TIME');
legend ('x1','x2');

%using adams PC------------------------------------
figure(5);
for h = [0.018,0.005]
    [x1,x2] = AdamsPC(a,b,h,f1,f2,x0(1),x0(2));
    switch h
        case 0.018
            plot(x1,x2,'b');
        case 0.005
            plot(x1,x2,'r');
    end
    grid on;
    hold on;
end
legend('0.018','0.005');
title('USING ADAMS PC - MOTION');
%plotting with time
figure(6);
[x1,x2, t] = AdamsPC(a,b,0.018,f1,f2,x0(1),x0(2));
plot(t,x1);
hold on;
grid on;
plot(t,x2);
title('USING ADAMS PC - TIME');
legend ('x1','x2');
```

## Task2b

```
%initialising
clc;clear;
%initial conidtions
x0 = [0, 0.5];
a = 0;b = 15;
interval = [a, b];


h0 = 0.01; %initial step
%declaring the functions
func=@(t,y) [y(2)+y(1)*(0.5-y(1)^2-y(2)^2);-y(1)+y(2)*(0.5-y(1)^2-y(2)^2)];
%error tolerances
eps_rel = 10^-6;%relative tolerance
eps_abs = 10^-4;%absolute tolerance
hmin = 10^-7;%minimum step size
```

```
h(2) = h0;
t(1) = a;
n = (b-a)/h(2);
x = x0';
t(2) = t(1) + h(2);
xtmp(:,1)=x(:,1);
i = 2;

while t<= b %while we are in range keep looping algorithm
    incrh = true;

    %Runge kutta method
    tj = t(i);
    k1 = h(i)*feval(func,tj,x(:,i-1));
    k2 = h(i)*feval(func,tj+h(i)/2,x(:,i-1)+k1/2);
    k3 = h(i)*feval(func,tj+h(i)/2,x(:,i-1)+k2/2);
    k4 = h(i)*feval(func,tj+h(i),x(:,i-1)+k3);
    x(:,i) = x(:,i-1) + (k1+2*k2+2*k3+k4)/6;
    %error calculation
    d1(i) = (x(1,i)-xtmp(1,1))/((2^4)-1);
    d2(i) = (x(2,i)-xtmp(2,1))/((2^4)-1);
    xtmp(:,1) = x(:,i);

    %calculation sa1 = s*alpha
    e(i) = (abs(x(1,i)))*eps_rel+eps_abs;
    sa1 = (e(i)/abs(d1(i)))^(1/5)*0.9;
    e(i) = (abs(x(2,i)))*eps_rel+eps_abs;
    sa2 = (e(i)/abs(d2(i)))^(1/5)*0.9;

    %selecting smaller sa as the new step size
    if sa1 < sa2
        nh = h(i)*sa1;
    else
        nh = h(i)*sa2;
    end
    %increasing until sa is smaller than 1
    while((sa1<1)&&(sa2<1))
        incrh = false;
        if nh>hmin
            h(i) = nh;
            break;
        else %if step size is smaller than hmin
            disp('Accuracy not possible to satisfy');
            break;
        end
    end
    if incrh == true %assigning new step sizes
        h(i+1) = min(nh,3*h(i));
    else
        h(i+1) = h(i);
    end
    if t(i)+h(i+1)>b
        break %if interval end is met then break
    end
    i = i+1;
    t(i) = t(i-1)+h(i);
end


%All figures

figure(1);
plot(t,x(1,:));
axis([0 15 -1 8])
hold on
grid on
title('USING RK4 VARIABLE STEP SIZE - TIME');
xlabel('t');ylabel('x');
```

```
plot(t,x(2,:),'r');
legend('x1','x2');


figure(2);
plot(x(1,:),x(2,:));
hold on
grid on
xlabel('x1'); ylabel('x2');
title('USING RK4 VARIABLE STEP SIZE - MOTION');



figure(3);
semilogy(t,h(1:length(t)));
axis([0 15 0 2.5])
grid on
title('STEP-SIZE OVER TIME');
xlabel('t');ylabel('h');
legend('Step-Size');

figure(4);
semilogy(t,e(1:length(t)));
axis([0 15 0 1.1*10^(-4)])
grid on
title('ERROR OVER TIME');
xlabel('t');ylabel('e');
legend('Error Estimate');

figure(5);
plot(t,abs(d1));
axis([0 15 0 6*10^(-5)])
hold on
grid on
title('ABSOLUTE ERROR OVER TIME');
xlabel('t');ylabel('error');
plot(t,abs(d2),'r');
legend('Error of x1','Error of x2');
```

## RK4

```
%runge kutta constant step
%start, stop = interval
%h = step size
%f1, f2 = function 1 and 2
%y1, y2 = initial conditions
%x1, x2 = values of x1, x2
%t = time
function [x1, x2, t] = RK4CStep(start,stop,h,f1,f2,y1,y2)

%number of steps in interval
iter = ceil(stop/h);

%preallocation
t = zeros(1,iter);
x1 = zeros(1,iter);
x2 = zeros(1,iter);

%initial conditions
t(1) = start;
x1(1) = y1;
x2(1) = y2;

%loops through all points
for i = 1:iter
```

```
    t(i+1) = t(i) + h;
    %runge kutta of 4th order
    k11 = f1(t(i),x1(i),x2(i));
    k21 = f2(t(i),x1(i),x2(i));
    k12 = f1(t(i)+h/2,x1(i)+(h/2)*k11,x2(i)+(h/2)*k21);
    k22 = f2(t(i)+h/2,x1(i)+(h/2)*k11,x2(i)+(h/2)*k21);
    k13 = f1(t(i)+h/2,x1(i)+(h/2)*k12,x2(i)+(h/2)*k22);
    k23 = f2(t(i)+h/2,x1(i)+(h/2)*k12,x2(i)+(h/2)*k22);
    k14 = f1(t(i)+h,x1(i)+h*k13,x2(i)+h*k23);
    k24 = f2(t(i)+h,x1(i)+h*k13,x2(i)+h*k23);

    x1(i+1) = x1(i) + (h/6)*(k11 + 2*k12 + 2*k13 + k14);
    x2(i+1) = x2(i) + (h/6)*(k21 + 2*k22 + 2*k23 + k24);
end

end
```

## Adams implicit

```
function sum = implicitSum(k, f,t,i,x1,x2)
sum = 0;
B = [1427, -798, 482, -173, 27];
B = B/1440;
for j=1:k
    sum = sum + B(j) * f(t(i-j),x1(i-j),x2(i-j));
end
end
```

## Adams explicit

```
function sum = explicitSum(k,f,t,i,x1,x2)
sum = 0;
%b = [1901/720, -2774/720, 2616/720, -1274/720, 251/720];
B = [1901, -2774, 2616, -1274, 251];
B = B/720;
for j=1:k
    sum = sum + B(j)*f(t(i-j),x1(i-j),x2(i-j));
end
end
```

## Adams PC

```
%Adams PC Method
%start, stop = interval
%h = step size
%f1, f2 = function 1 and 2
%y1, y2 = initial conditions
%x1, x2 = values of x1, x2
%t = time
function [x1, x2, t] = AdamsPC(start, stop, h, f1, f2, y1, y2)
B0 = 475/1440; %B0 from table
k = 5; %P5EC5E, k = 5
tfin = (k*h)-start; %for 5point runge-kutta approximation
iter = ceil((stop - start)/h);
t = zeros(1,iter);%preallocation
[x1, x2] = RK4CStep(start,tfin,h,f1,f2,y1,y2);%initial points from RK4
for i=6:iter
```

```
    t(1+i) = t(i)+h;
    %prediction - explicit sum
    x1(i+1) = x1(i) + h*explicitSum(k,f1,t,i,x1,x2);
    x2(i+1) = x2(i) + h*explicitSum(k,f2,t,i,x1,x2);
    %evaluation
    fn1 = f1(t(i+1),x1(i+1),x2(i+1));
    fn2 = f2(t(i+1),x1(i+1),x2(i+1));
    %correction - implicit sum
    x1(i+1) = x1(i) + h*implicitSum(k,f1,t,i,x1,x2)+h*B0*fn1;
    x2(i+1) = x2(i) + h*implicitSum(k,f2,t,i,x1,x2)+h*B0*fn2;
end
end
```