[ENUME] Report of a project C no.30

Jan Świerczyński

293660

Task no.1

In task number one, we were assigned to determine the polynomial function y = f(x) that best fits our given experimental data:

X:	Y:
-5.0000	-1.6889
-4.0000	-4.7689
-3.0000	-3.8259
-2.0000	-2.0068
-1.0000	-0.6884
0	0.9391
1.0000	-1.1556
2.0000	-4.4293
3.0000	-12.5746
4.0000	-25.6768
5.0000	-45.0598

We were obliged to use the least-square approximation and test polynomials of various degrees.

The method of **least squares** is an approach to approximate the solution of sets of equations in which there are more equations than unknowns, by minimizing the sum of squares of the residuals, made in the result of every single equation.

Discrete least-squares approximation

For a given finite number of points $x_0, x_1, \dots, x_N$ ($x_i \neq x_j$), 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 $X_n \subset X$ interpolating functions, i. e.,

$$\forall F \in X_n \quad F(x) = \sum_{i=0}^{n} a_i \phi_i(x) .$$

The approximation problem to find values of the parameters $a_0, a_1, \dots, a_n$:

$$H(a_0, \dots, a_n) \stackrel{\text{df}}{=} \sum_{j=0}^{N} \left[f(x_j) - \sum_{i=0}^{n} a_i \phi_i(x_j) \right]^2 .$$

The formula for the coefficients a_0, a_1, … , a_n can be derived from the necessary condition for a minimum.

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

The system of linear equations with the unknowns a_0, a_1, … , a_n is called the set of **normal equations**, its matrix is known as **the Gram's matrix**. The normal equations can be written in a simple form if the scalar product is defined:

$$\langle\phi_i, \phi_k\rangle \stackrel{\text{df}}{=} \sum_{j=0}^{N}\phi_i\left(x_j\right)\phi_k\left(x_j\right).$$

Then the set of normal equations takes also simple form:

$$\begin{bmatrix} \langle\phi_0, \phi_0\rangle & \langle\phi_1, \phi_0\rangle & \cdots & \langle\phi_n, \phi_0\rangle \\ \langle\phi_0, \phi_1\rangle & \langle\phi_1, \phi_1\rangle & \cdots & \langle\phi_n, \phi_1\rangle \\ \vdots & \vdots & \vdots & \vdots \\ \langle\phi_0, \phi_n\rangle & \langle\phi_1, \phi_n\rangle & \cdots & \langle\phi_n, \phi_n\rangle \end{bmatrix}\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \langle\phi_0, f\rangle \\ \langle\phi_1, f\rangle \\ \vdots \\ \langle\phi_n, f\rangle \end{bmatrix}$$

Now we can define the following matrix **A**:

$$\mathbf{A} = \begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) & \cdots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_n(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_n(x_2) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_n(x_N) \end{bmatrix}$$

And:

$$\begin{aligned} \mathbf{a} &= [a_0\ a_1 \cdots a_n]^{\mathrm{T}}, \\ \mathbf{y} &= [y_0\ y_1 \cdots y_N]^{\mathrm{T}}, \quad y_j = f(x_j), \quad j = 0, 1, ..., N. \end{aligned}$$

Hence, the performance function mentioned before:

$$H(a_0, ..., a_n) \stackrel{\text{df}}{=} \sum_{j=0}^{N}\left[f(x_j) - \sum_{i=0}^{n} a_i\phi_i(x_j)\right]^2.$$

Can be written as:

$$H(\mathbf{a}) = (\parallel \mathbf{y} - \mathbf{A}\mathbf{a} \parallel_2)^2.$$

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

Utilizing the definition of the matrix **A**, the set of normal equations can be written in the form :

$$\mathrm{A}^{\mathrm{T}}\mathrm{A}\ \mathrm{a} = \mathrm{A}^{\mathrm{T}}\mathrm{y}.$$

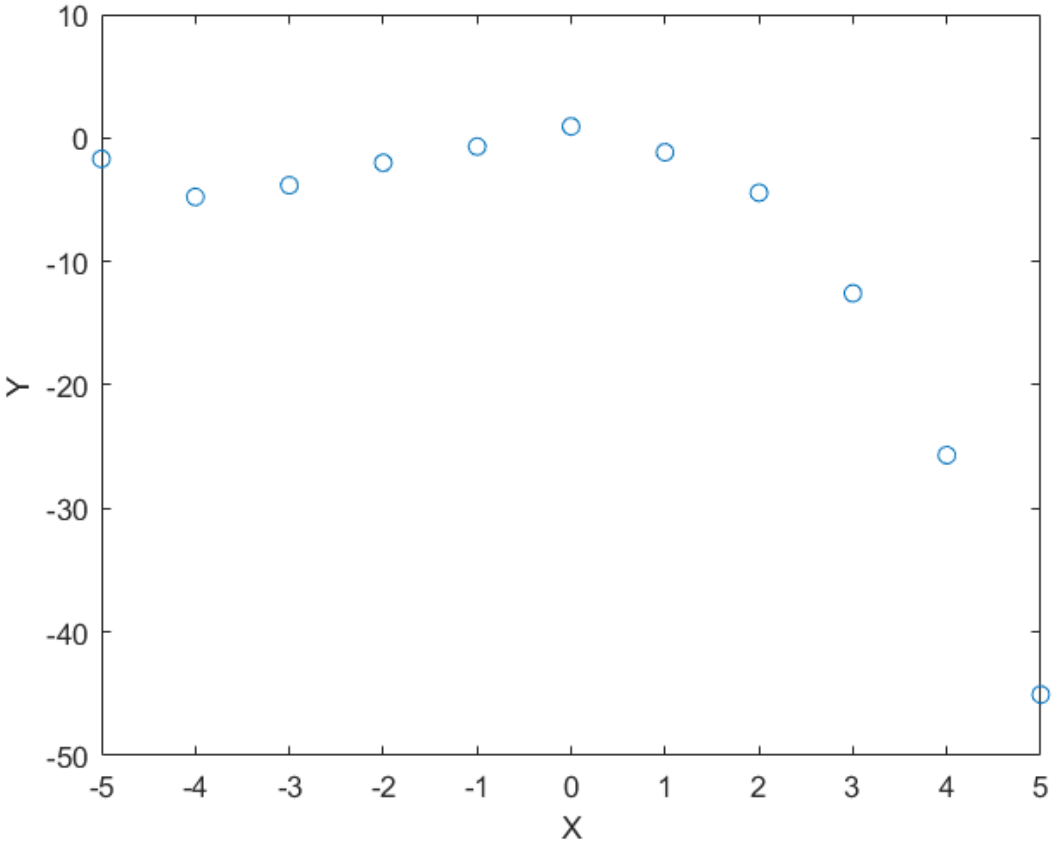
Because the matrix **A** has full rank, then **the Gram's** matrix $\mathbf{A^TA}$ is non-singular. This implies uniqueness of the solution of the set of normal equations. But, even being non-singular, the matrix $\mathbf{A^TA}$ can be badly conditioned – its condition number is a square of the condition number of **A** (which I am also checking in my program). That is why it is recommended to use QR factorization of matrix **A** (LLSP).

.

.

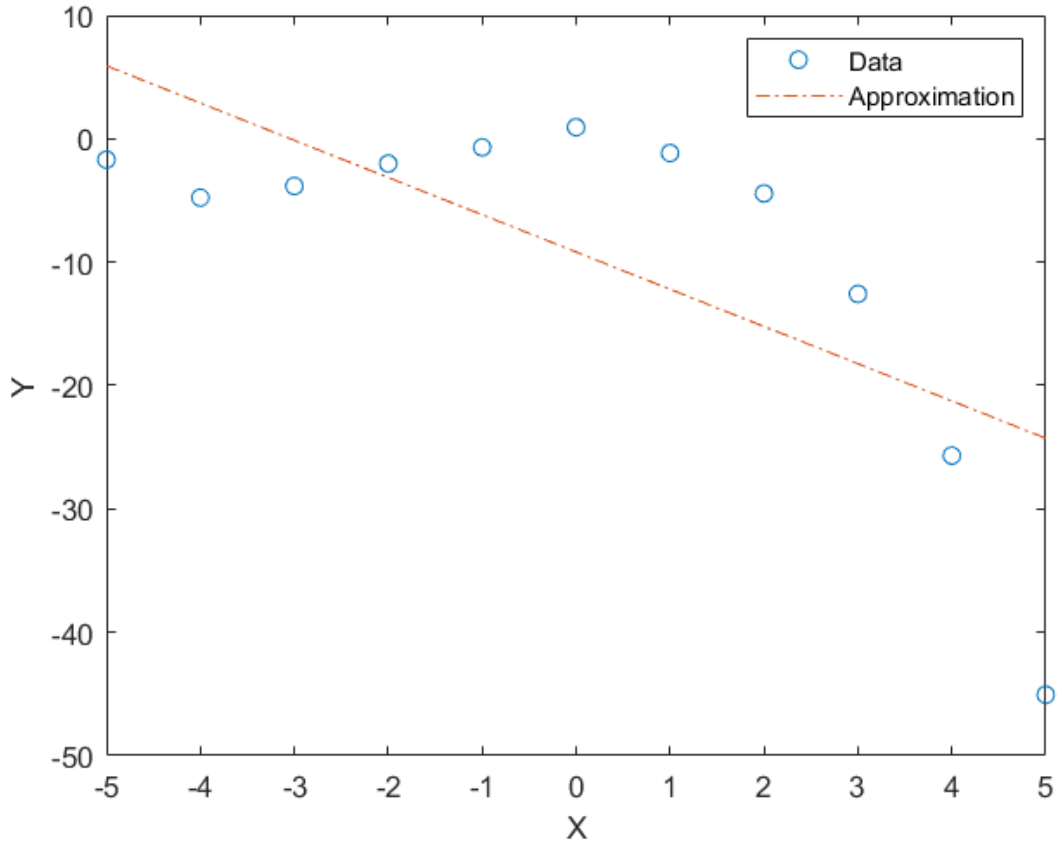
.

Experimental data on a plot:

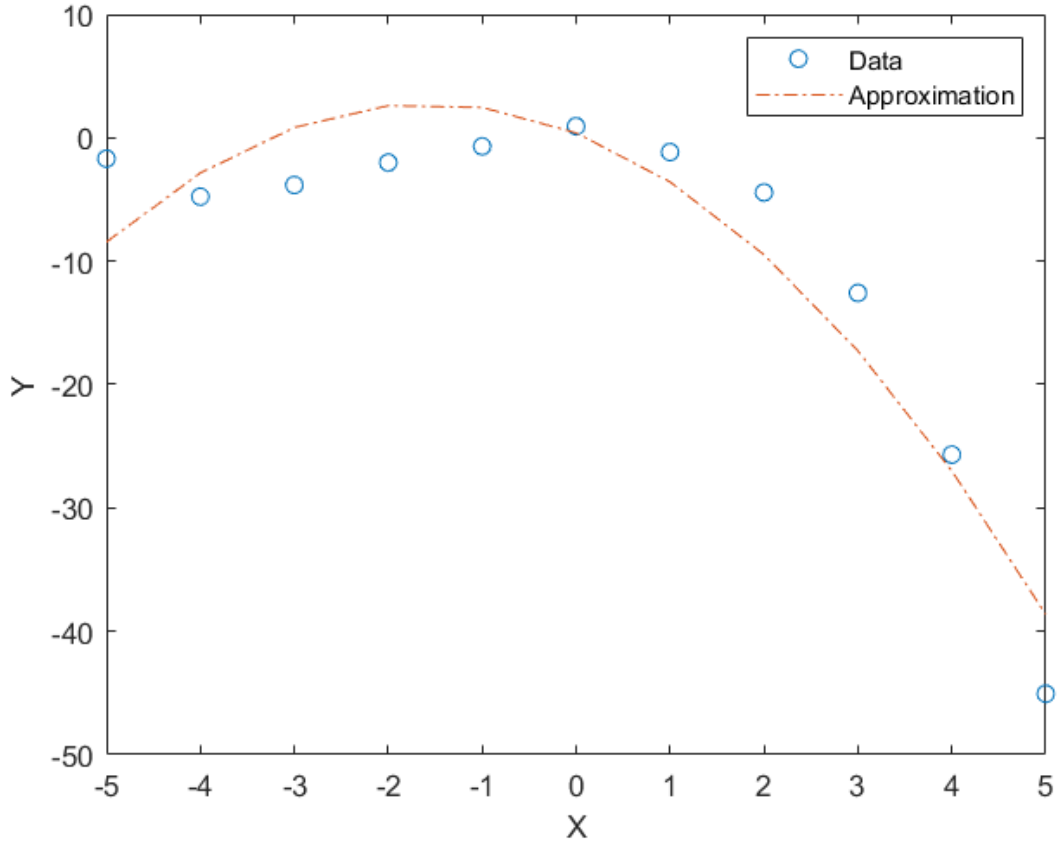


Legend: *"n" stands for polynomial degree, condA is the condition number of matrix A, condG is a calculated condition number of matrix G (by rising the value of condA to the second power) and true_condG is the calculated condition number of matrix G the same way as condA; "a" are our coefficients of polynomial and "r" is the error defined as the Euclidean norm of the vector of residuum.*

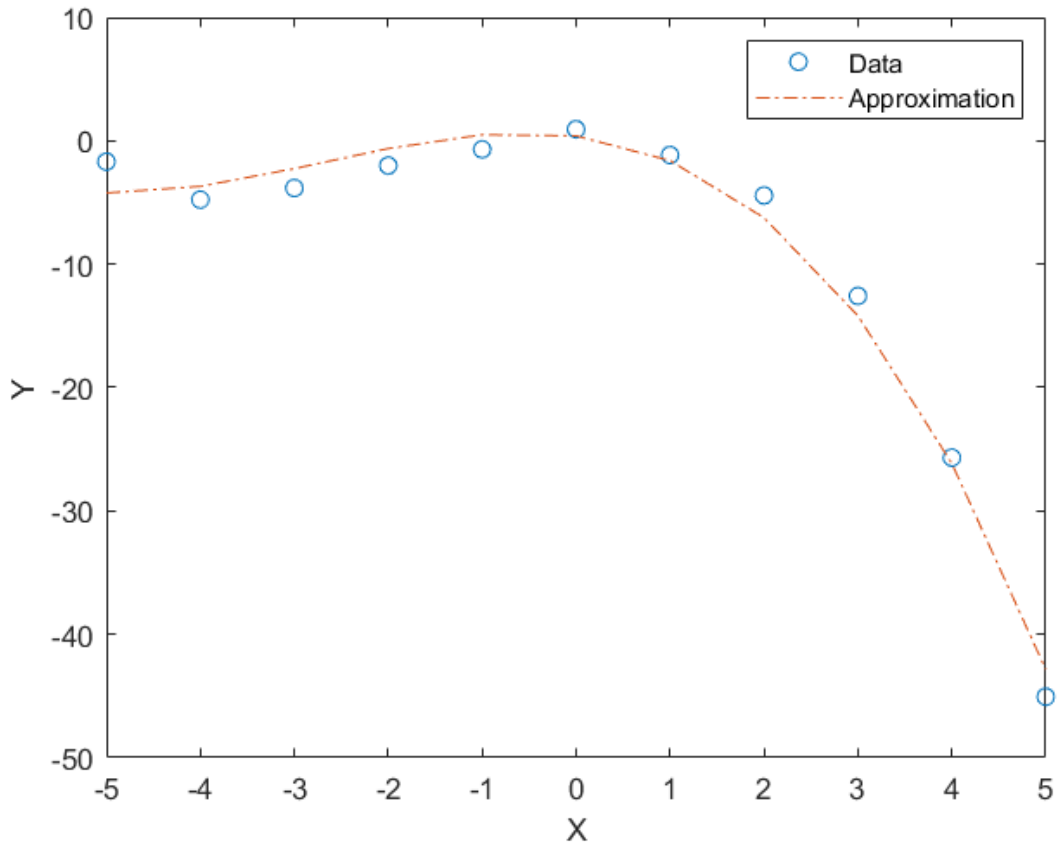
```
n = 1
condA = 3.1623
condG = 10.0000
true_condG = 10
a = 2×1
     -9.1760
     -3.0186

r = 31.3988
```



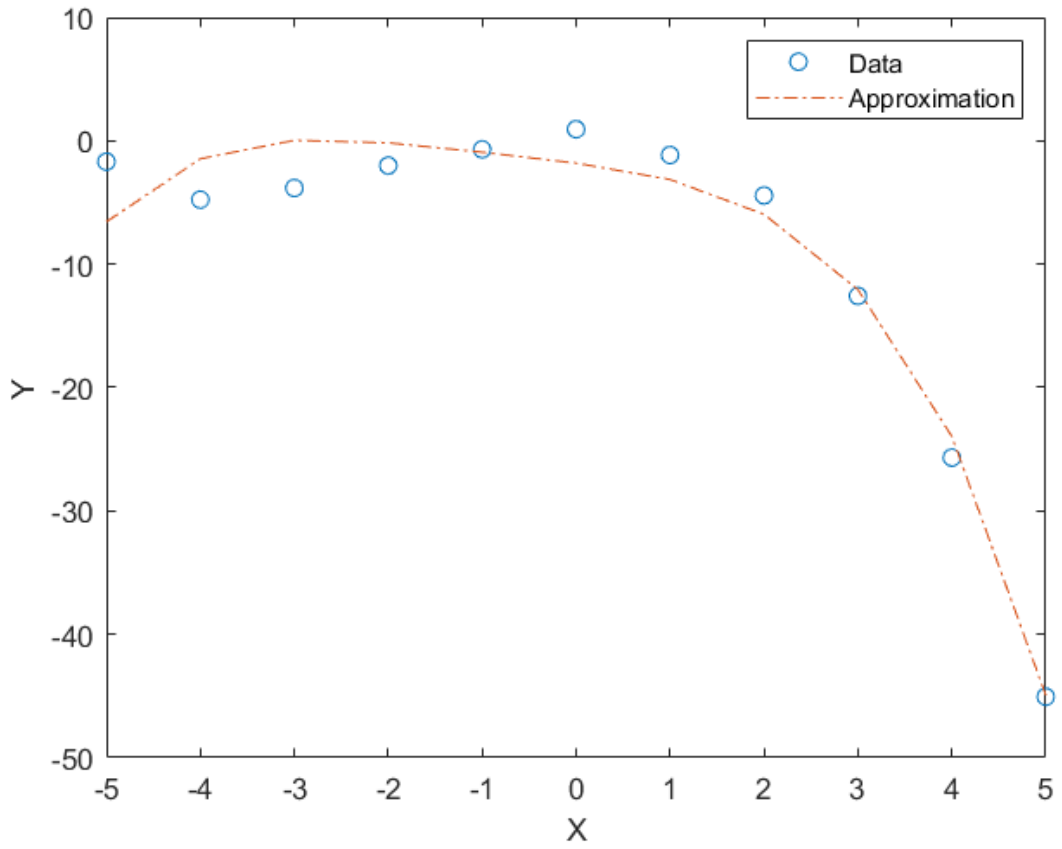
```
n = 2
condA = 20.2183
condG = 408.7796
true_condG = 408.7796
a = 3×1
       0.4007
      -3.0186
      -0.9577

r = 14.1060
```



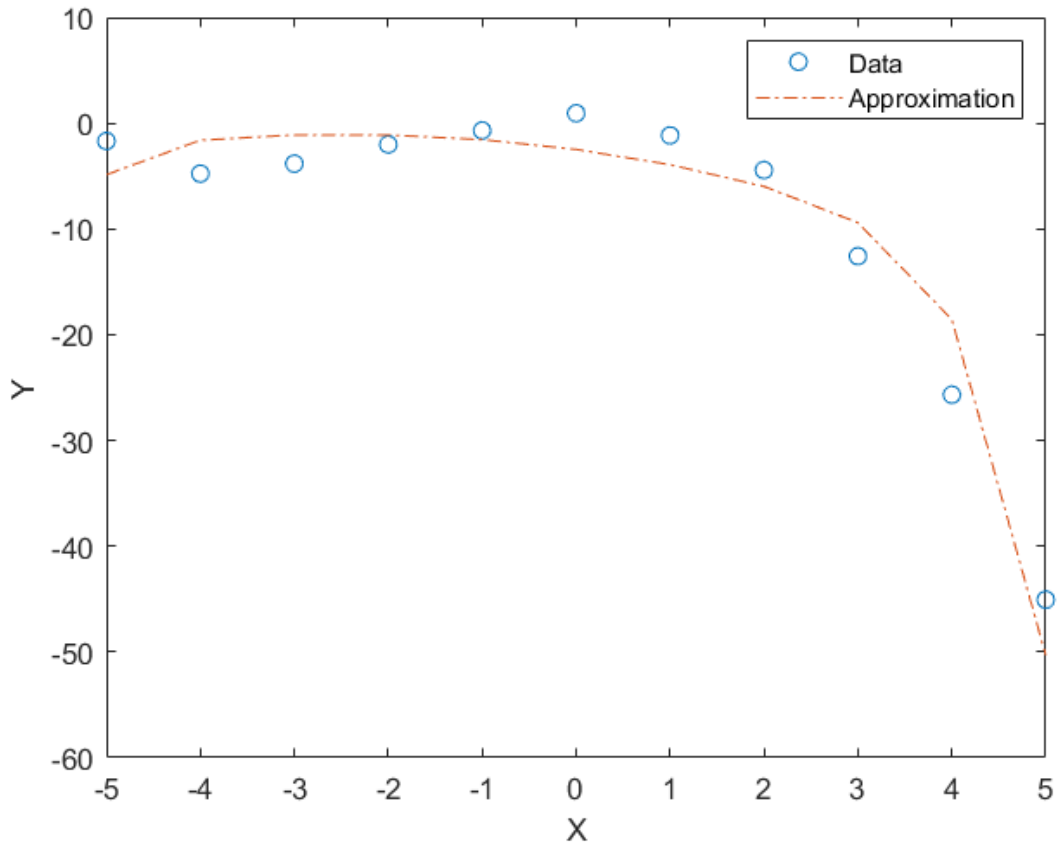
```
n = 3
condA = 92.5118
condG = 8.5584e+03
true_condG =
8.5584e+03
a = 4×1
       0.4007
      -0.9318
      -0.9577
      -0.1172

r = 4.9760
```



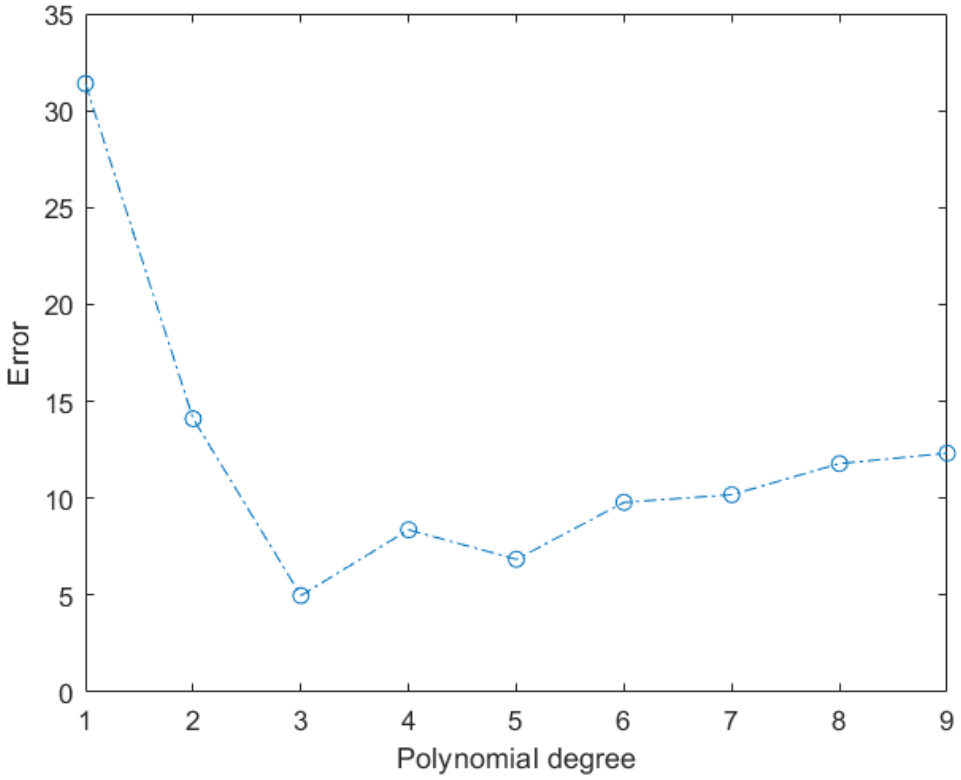
```
n = 4
condA = 563.8984
condG = 3.1798e+05
true_condG =
3.1798e+05
a = 5×1
      -1.8102
      -0.9919
      -0.1900
      -0.1139
      -0.0307

r = 8.3742
```



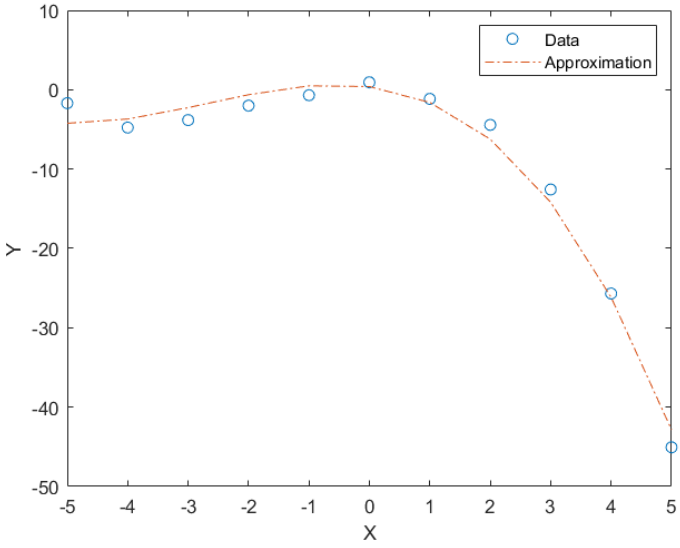
```
n = 8
condA = 5.7493e+05
condG = 3.3055e+11
true_condG =
3.3055e+11
a = 9×1
      -2.4829
      -1.1800
      -0.2529
      -0.0053
      -0.0025
      -0.0001
      -0.0000
      -0.0002
      -0.0000

r = 11.7917
```



Error to polynomial degree:



As we can observe the lowest error appeard for the polynomial of degree equal to three. Which means that this plot:



Represents the best approximation of a function, for a given data, the function equation looks like this: $f(x) = -0.1172*x^3-0.9577*x^2-0.9318*x+0.4007$.

For the task, matrix **A** was calculated from the natural polynomial basis (the power basis).

Here for n = 7:

	1	2	3	4	5	6	7	8	9
1	1	-5	25	-125	625	-3125	15625	-78125	390625
2	1	-4	16	-64	256	-1024	4096	-16384	65536
3	1	-3	9	-27	81	-243	729	-2187	6561
4	1	-2	4	-8	16	-32	64	-128	256
5	1	-1	1	-1	1	-1	1	-1	1
6	1	0	0	0	0	0	0	0	0
7	1	1	1	1	1	1	1	1	1
8	1	2	4	8	16	32	64	128	256
9	1	3	9	27	81	243	729	2187	6561
10	1	4	16	64	256	1024	4096	16384	65536
11	1	5	25	125	625	3125	15625	78125	390625

In addition our plot, showing error to polynomial degree, proves us that the order n of the approximating polynomial is usually much lower than the number of points (our data). When "n" value is near the value of number of points, the error value oscillates between 10 and 15.

Task no.2

In task no.2 we were given equations of a motion of a point:

$x_1' = x_2 + x_1*(0.5 - x_1^2 - x_2^2)$,

$x_2' = - x_1 + x_2*(0.5 - x_1^2 - x_2^2)$

According to them we had to determine the trajectory of the motion on the interval [0, 15] from the initial point $x_1(0) = 0$ and $x_2(0) = 0.5$.

2a)

For this part of the task we are asked to use Runge-Kutta method of 4^{th} order (RK4) and Adams PC (P_5EC_5E) to determine previously mentioned trajectory of a point.

Runge-Kutta methods:

A family of Runge-Kutta methods can be defined by the following formula:

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

Where

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

And also

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

To perform a single step of the method the values of the right-hand sides of the equations must by calculated precisely m times, that is why the method can be described as an "m-stage one".

If p(m) denotes a maximal possible order of the RK method, then it can be shown that:

$$\begin{aligned} p(m) &= m \quad \text{for } m = 1, 2, 3, 4, \\ p(m) &= m - 1 \quad \text{for } m = 5, 6, 7, \\ p(m) &\leq m - 2 \quad \text{for } m \geq 8. \end{aligned}$$

The methods with m = 4 and of order p = 4 are most important in practice – they constitute a good trade-off between the approximation accuracy and the number of arithmetic operations performed at one iteration.

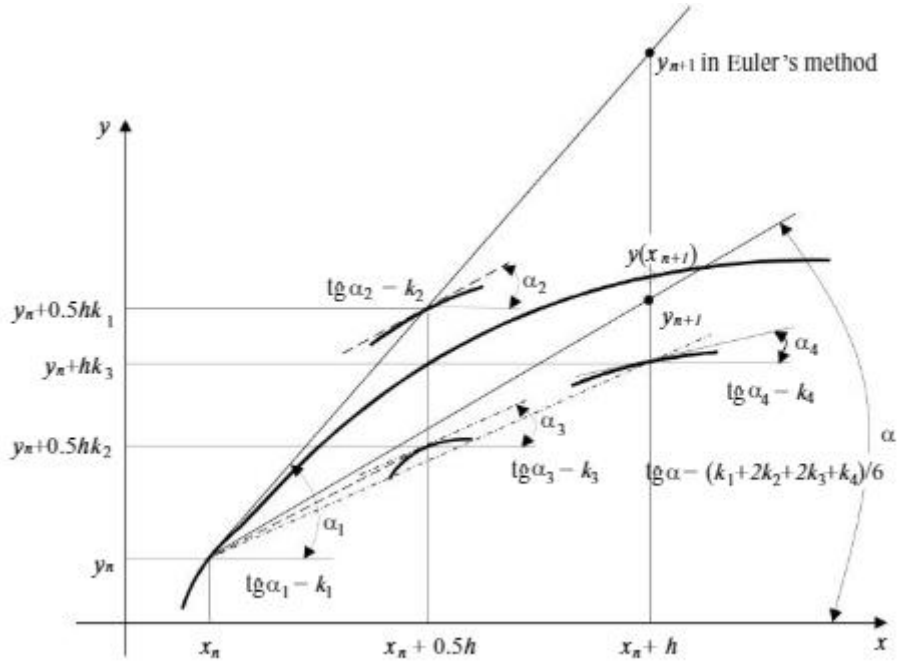
RK method of 4th order (classical) :

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

Is described with such equations.

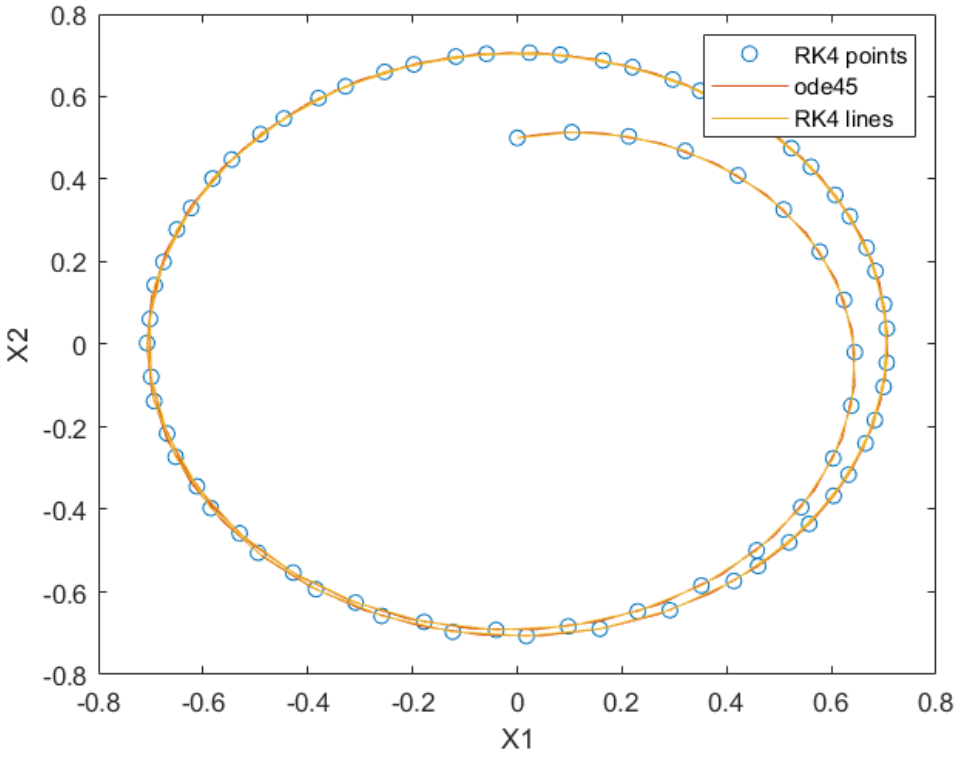
First equation stands for calculating the approximated value at a subsequent point, rest of the equations are the calculations of the new value in steps, used in first equation to get a score.

Graphical representation for one equation (m=1) :



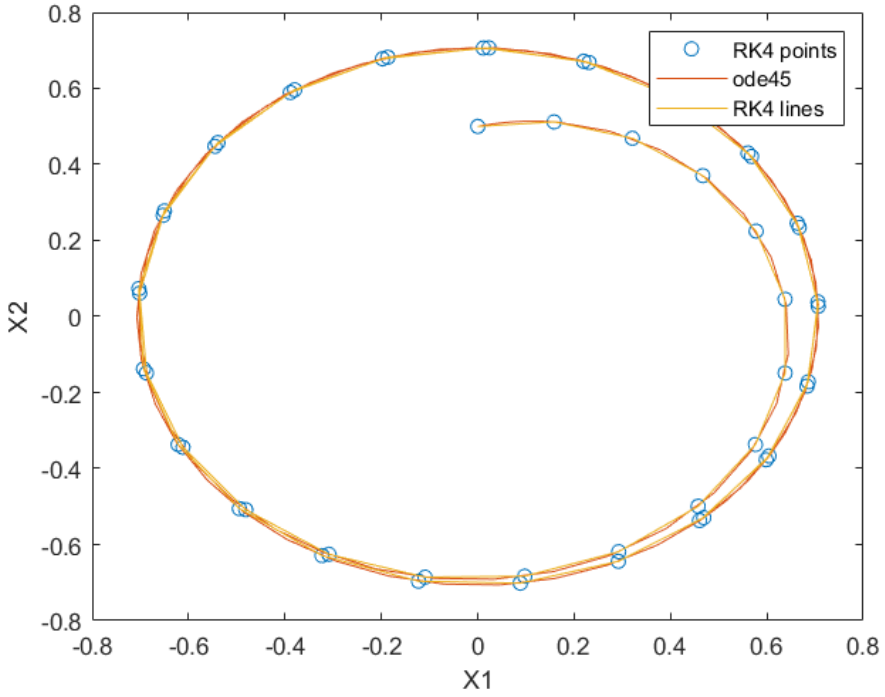

```
t = [0 15];
x0 = [0 0.5];
h = 0.2;
```

For such a data, where h is the “optimal” constant step size we obtained:



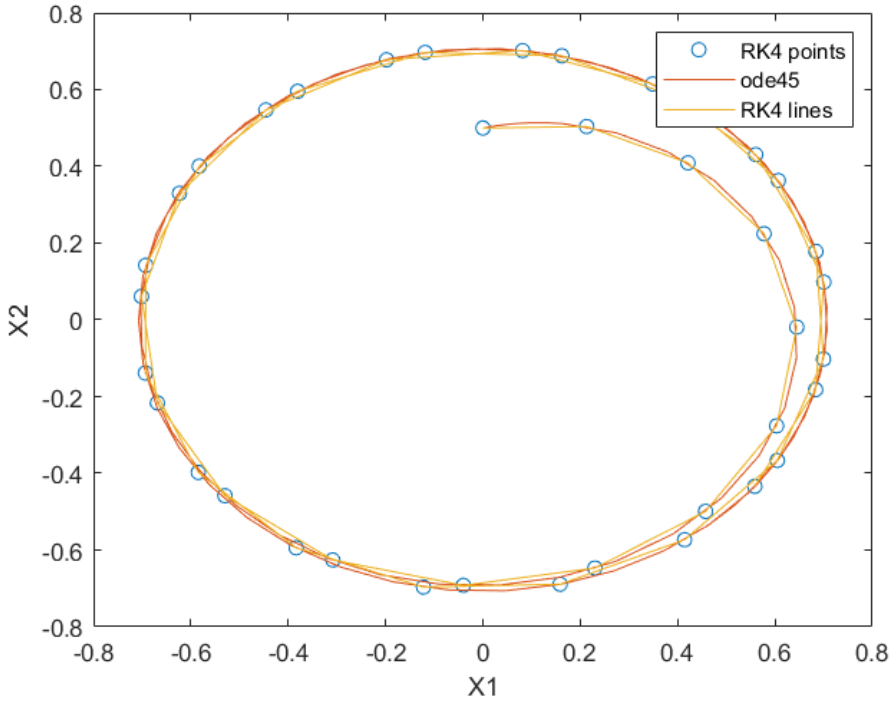
For:

```
t = [0 15];
x0 = [0 0.5];
h = 0.3;
```



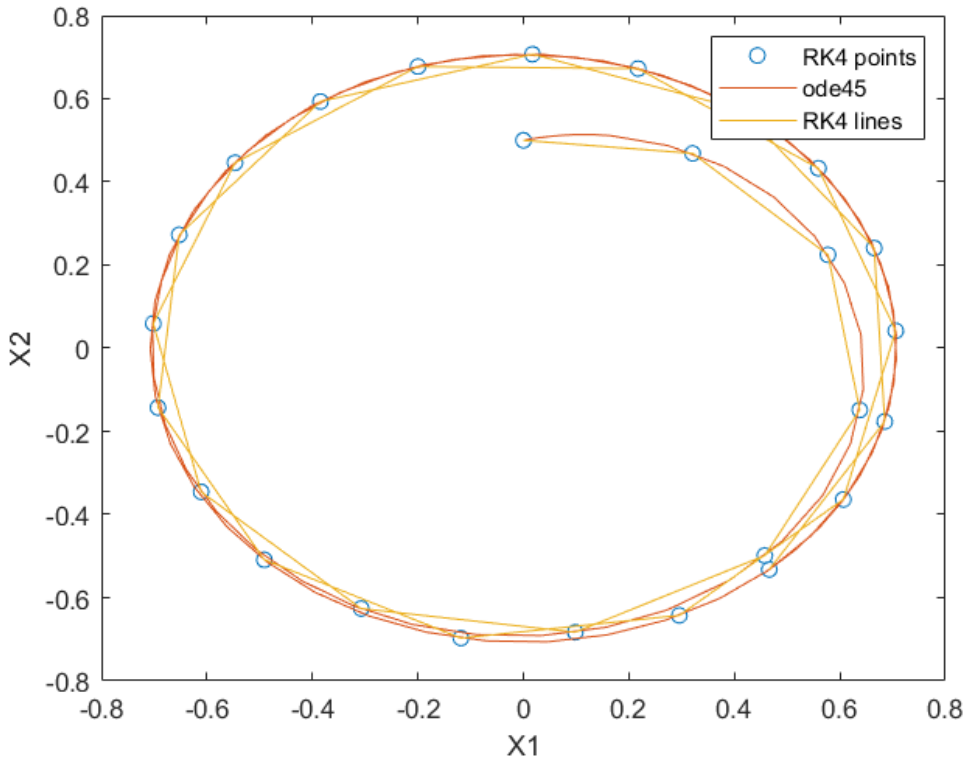
For:

```
t = [0 15];
x0 = [0 0.5];
h = 0.4;
```

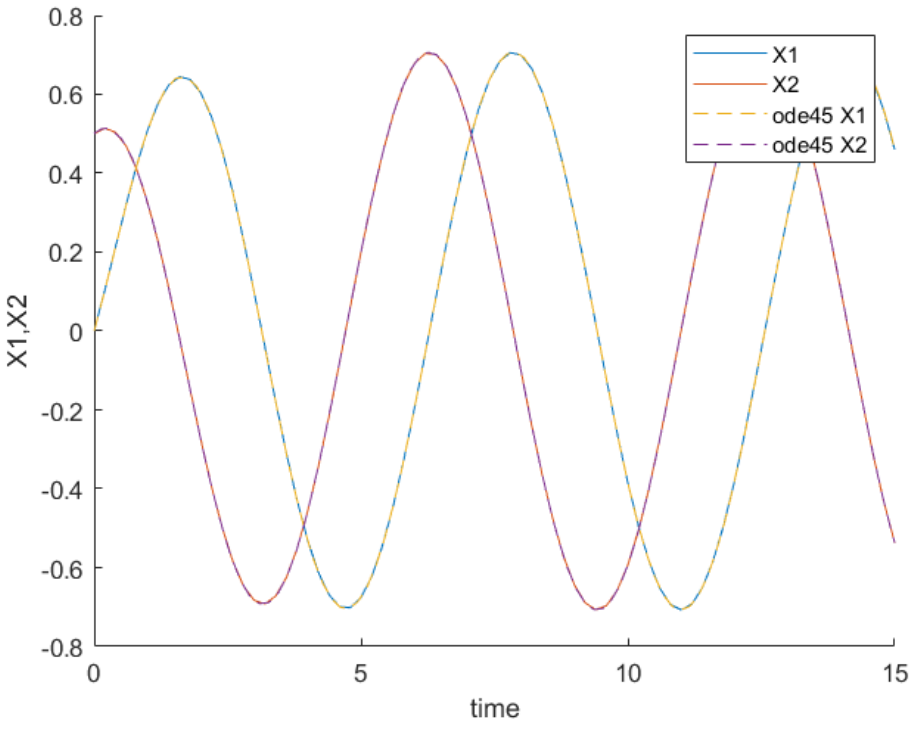


And finally for:

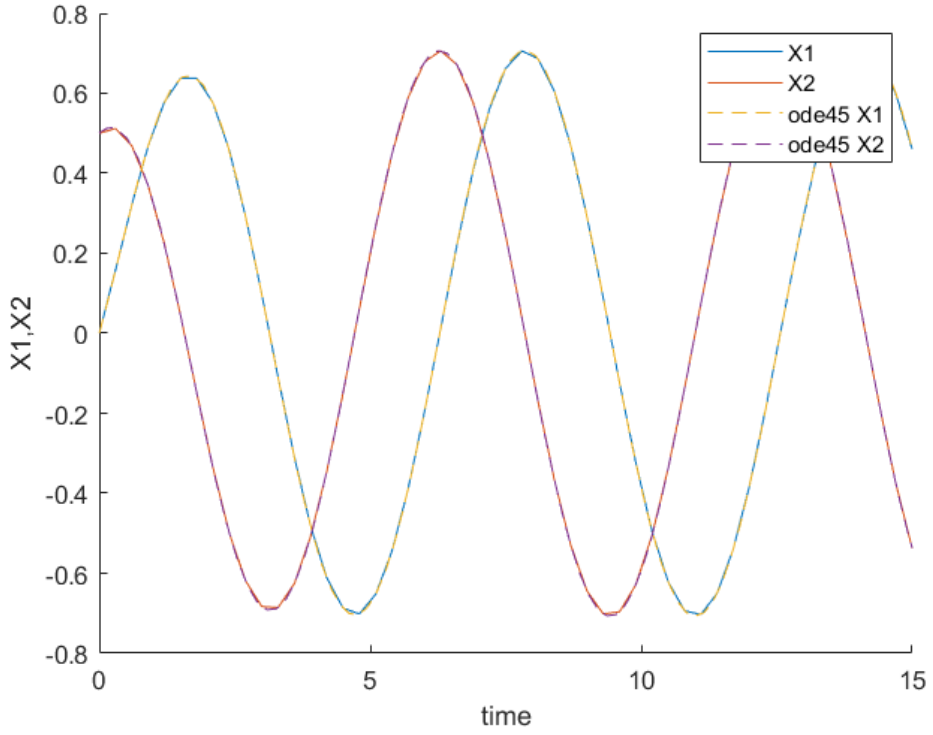
```
t = [0 15];
x0 = [0 0.5];
h = 0.6;
```



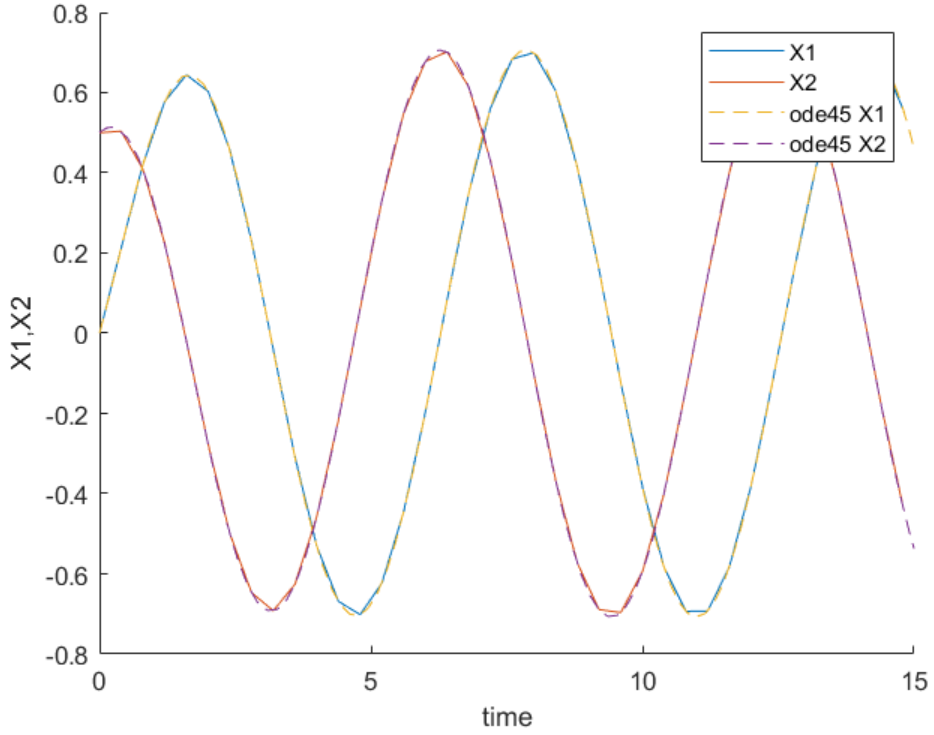
For h = 0.2: (Plots of problem solutions vs time)



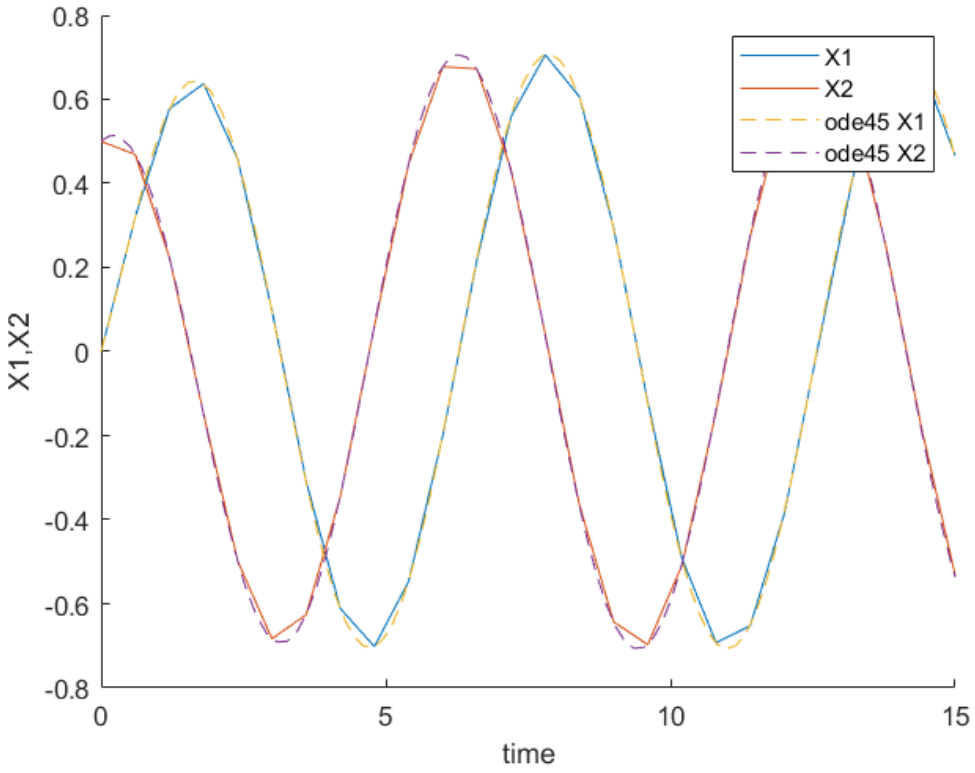
For h = 0.3:



For h = 0.4:



For h = 0.6:



Legend: *time stands for parameter t.*

Adams methods:

An initial value problem:

$$y'(x) = f(x, y(x)),$$
$$y(a) = y_a, \quad x \in [a, b],$$

Can be equivalently formulated in the form of an integral equation

$$y(x) = y(a) + \int_a^x f(t, y(t))\, dt.$$

Considering this integral on the interval $[x_{n-1}, x_n]$,

$$y(x_n) = y(x_{n-1}) + \int_{x_{n-1}}^{x_n} f(t, y(t))\, dt,$$

- leads to Adams methods:

Adams PC (P_5EC_5E):

PC – stands for predictor-corrector

It means that the first step in our algorithm is a prediction step, for which we are using Adams-Bashforth method:

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

k	β_1	β_2	β_3	β_4	β_5	β_6	β_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	$\frac{198721}{60480}$	$-\frac{447288}{60480}$	$\frac{705549}{60480}$	$-\frac{688256}{60480}$	$\frac{407139}{60480}$	$-\frac{134472}{60480}$	$\frac{19087}{60480}$

Which is described in equation above. To obtain a solution from it, we need to use a shown table and choose appropriate values for coefficient β.

For the second step, a correction one, we are using Adams-Moulton method:

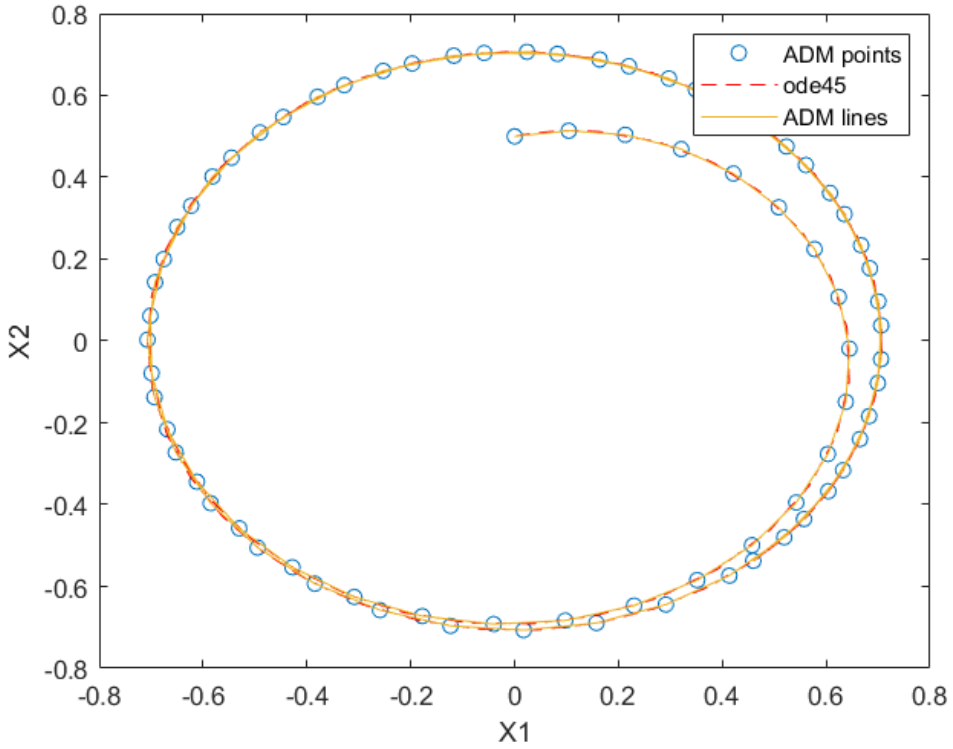
$$y_n = y_{n-1} + h \sum_{j=0}^{k} \beta_j^* \cdot f\left(x_{n-j}, y_{n-j}\right)$$

$$= y_{n-1} + h \cdot \beta_0^* \cdot f\left(x_n, y_n\right) + h \sum_{j=1}^{k} \beta_j^* \cdot f\left(x_{n-j}, y_{n-j}\right),$$

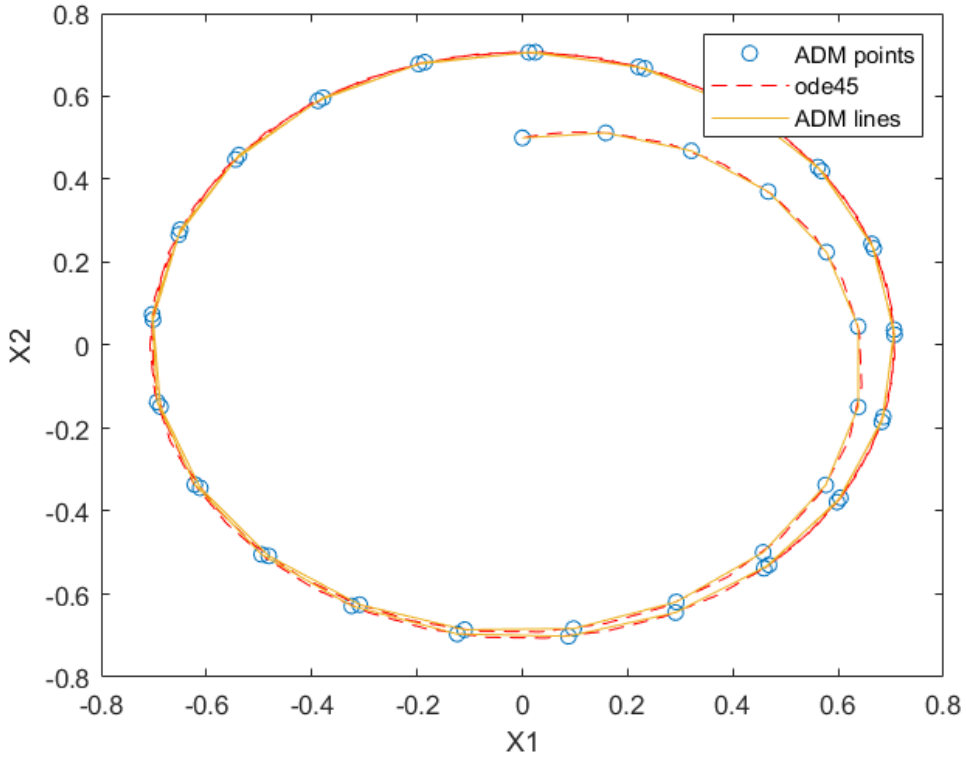
k	β_0^*	β_1^*	β_2^*	β_3^*	β_4^*	β_5^*	β_6^*	β_7^*
1^+	1							
1	$\frac{1}{2}$	$\frac{1}{2}$						
2	$\frac{5}{12}$	$\frac{8}{12}$	$-\frac{1}{12}$					
3	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$				
4	$\frac{251}{720}$	$\frac{646}{720}$	$-\frac{264}{720}$	$\frac{106}{720}$	$-\frac{19}{720}$			
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	$\frac{36799}{120960}$	$\frac{139849}{120960}$	$-\frac{121797}{120960}$	$\frac{123133}{120960}$	$-\frac{88547}{120960}$	$\frac{41499}{120960}$	$-\frac{11351}{120960}$	$\frac{1375}{120960}$

With the usage of a table, to choose appropriate coefficients. Like in previous method.

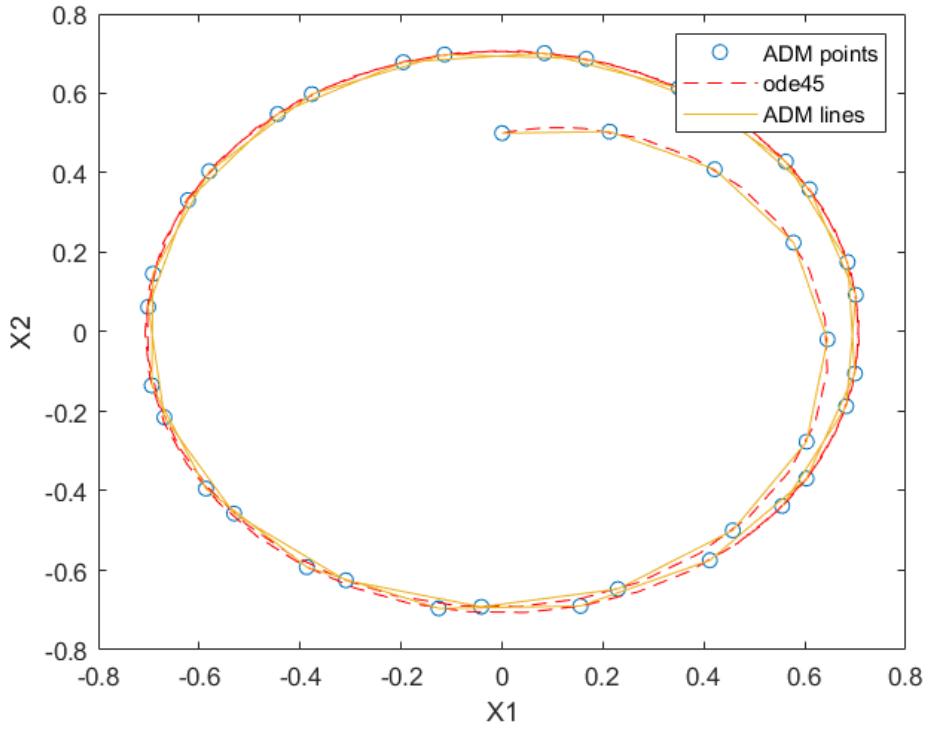
For h = 0.2:



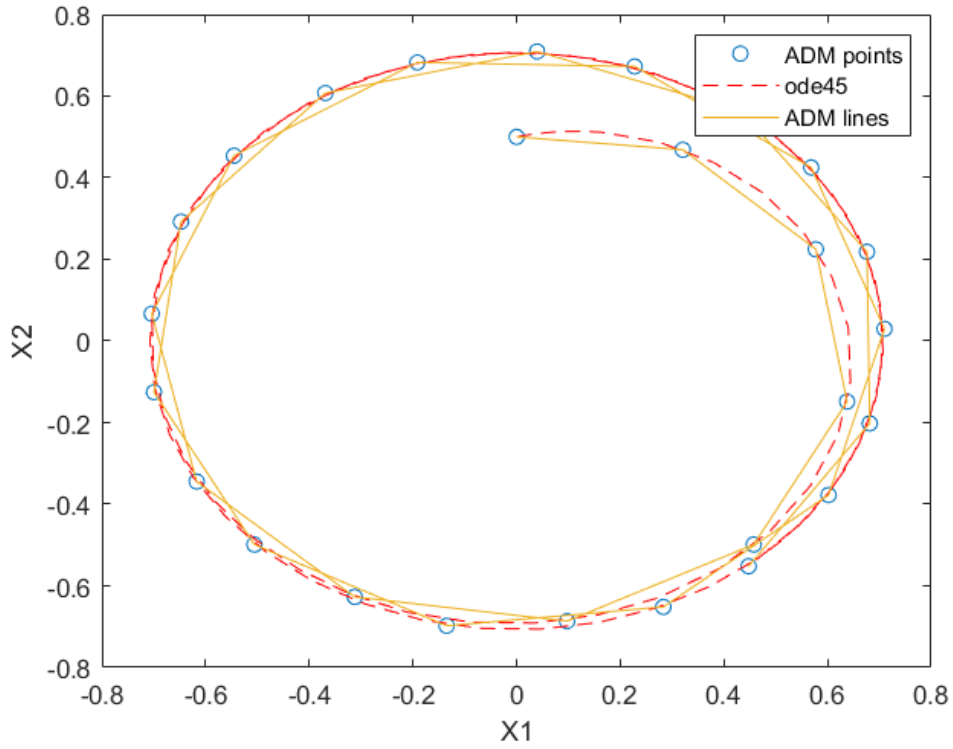
For h = 0.3:



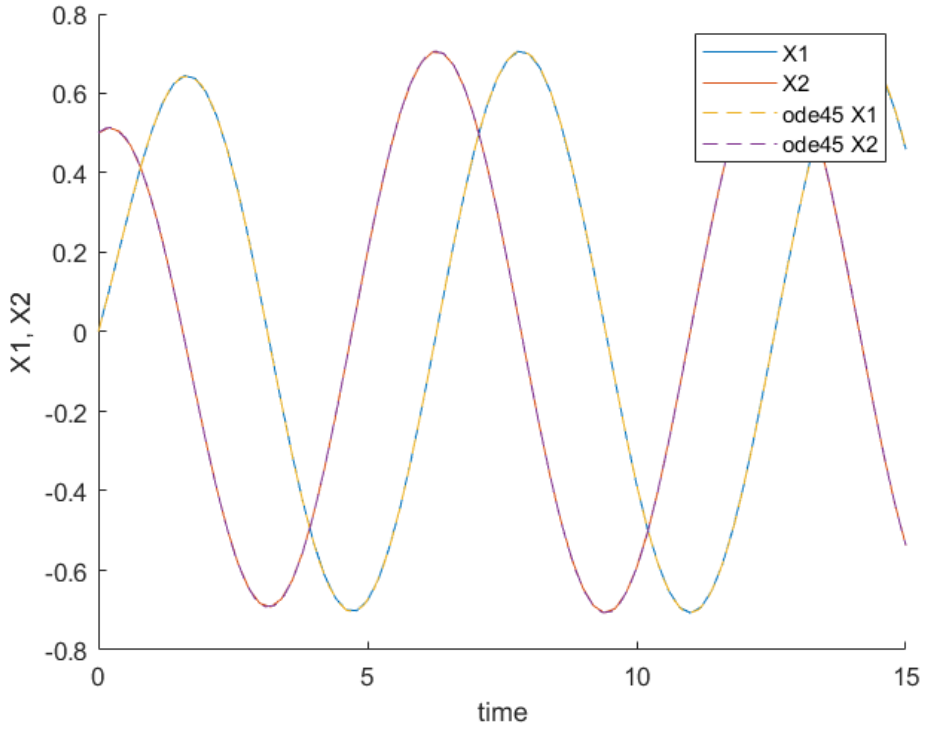
For h = 0.4:



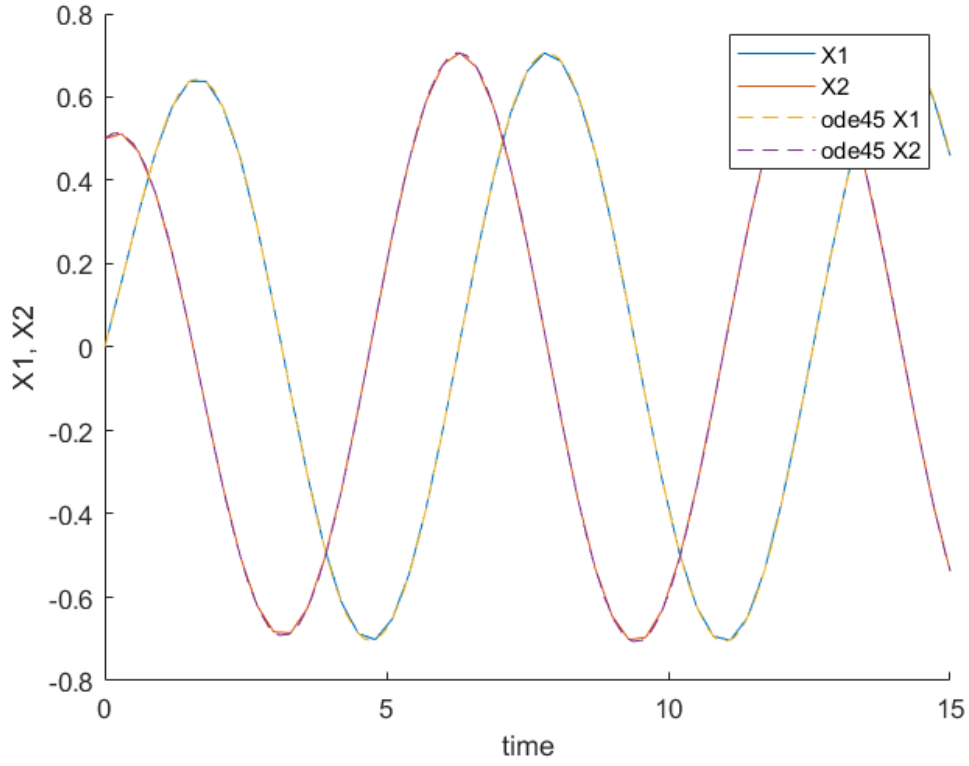
For h = 0.6:



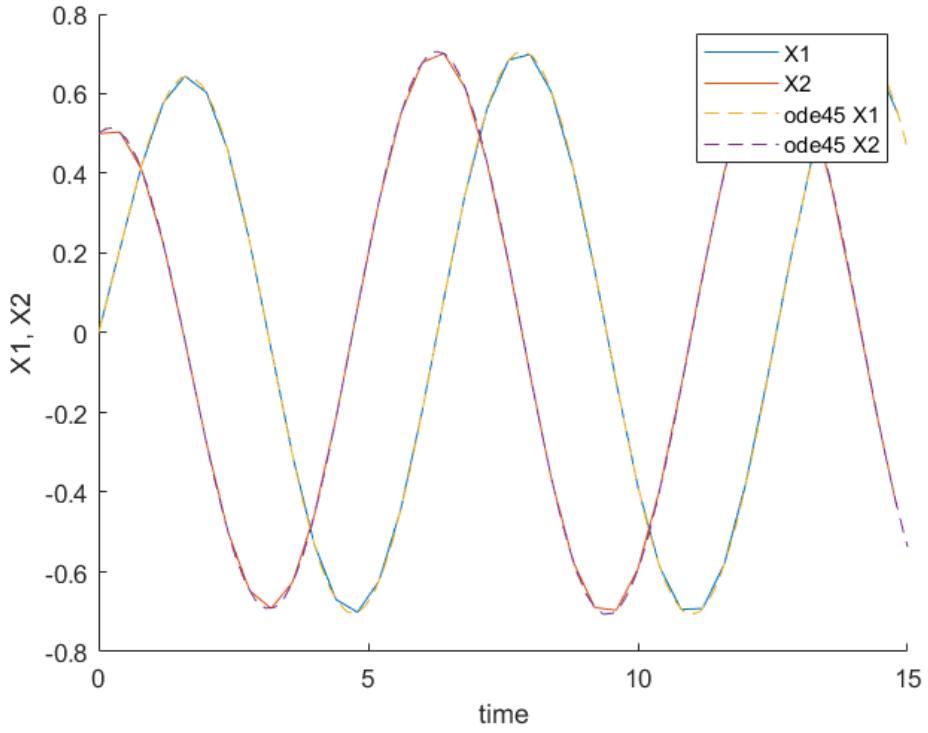
For h = 0.2:



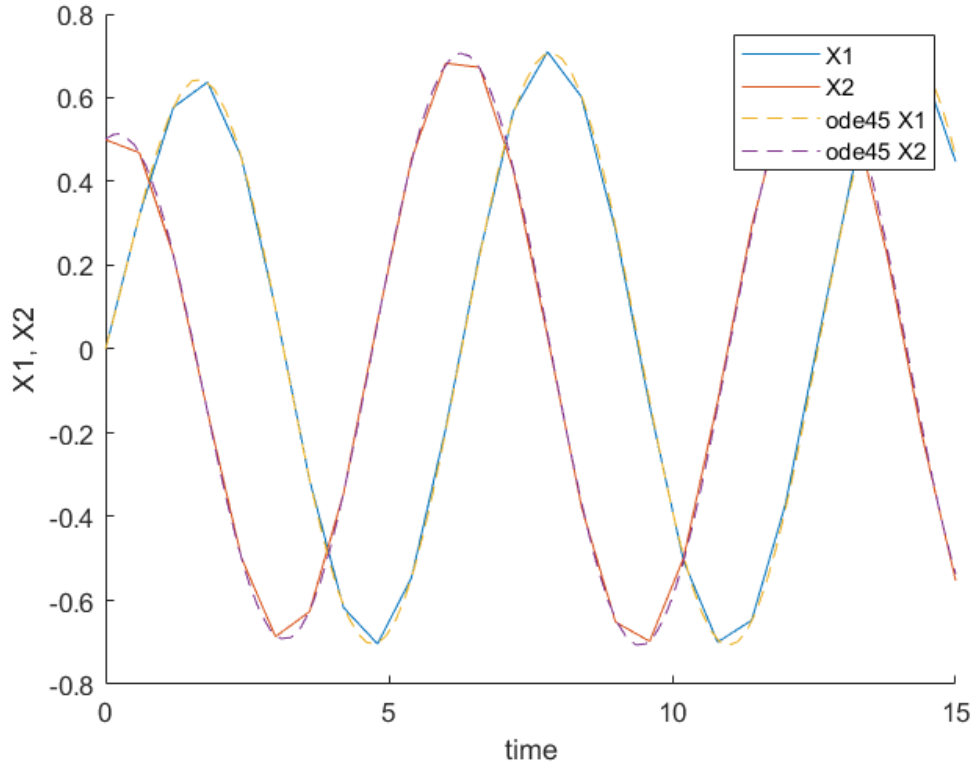
For h = 0.3:



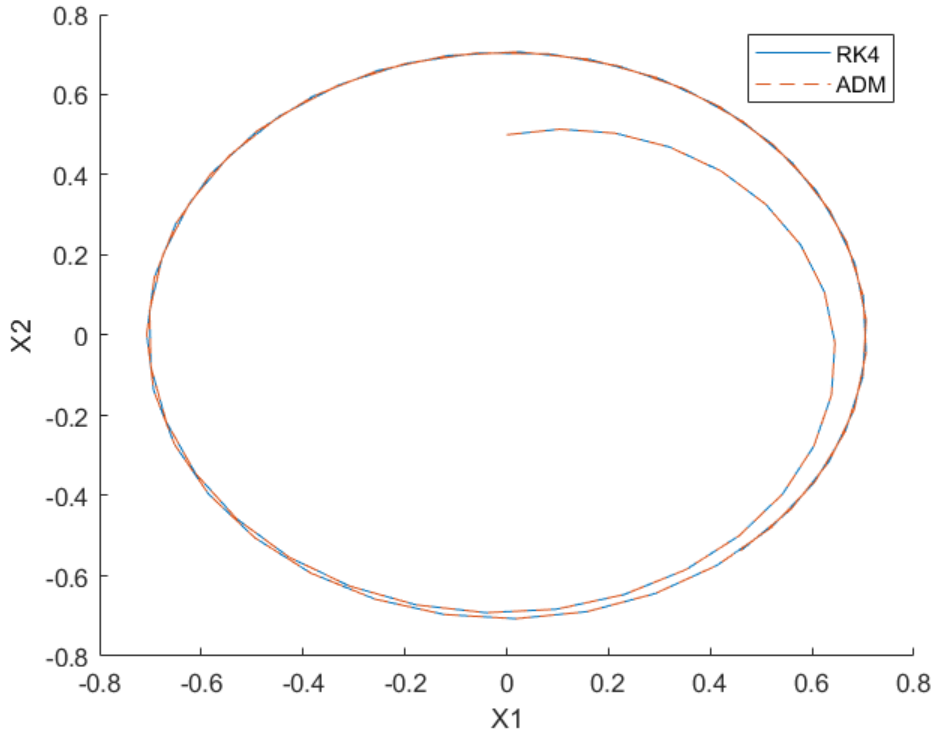
For h = 0.4:

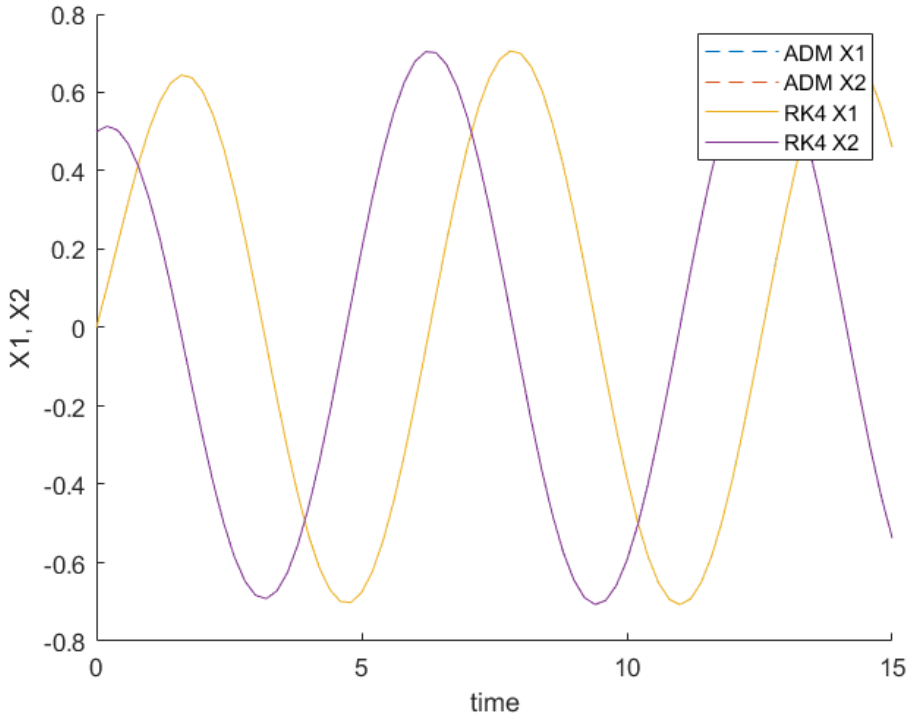


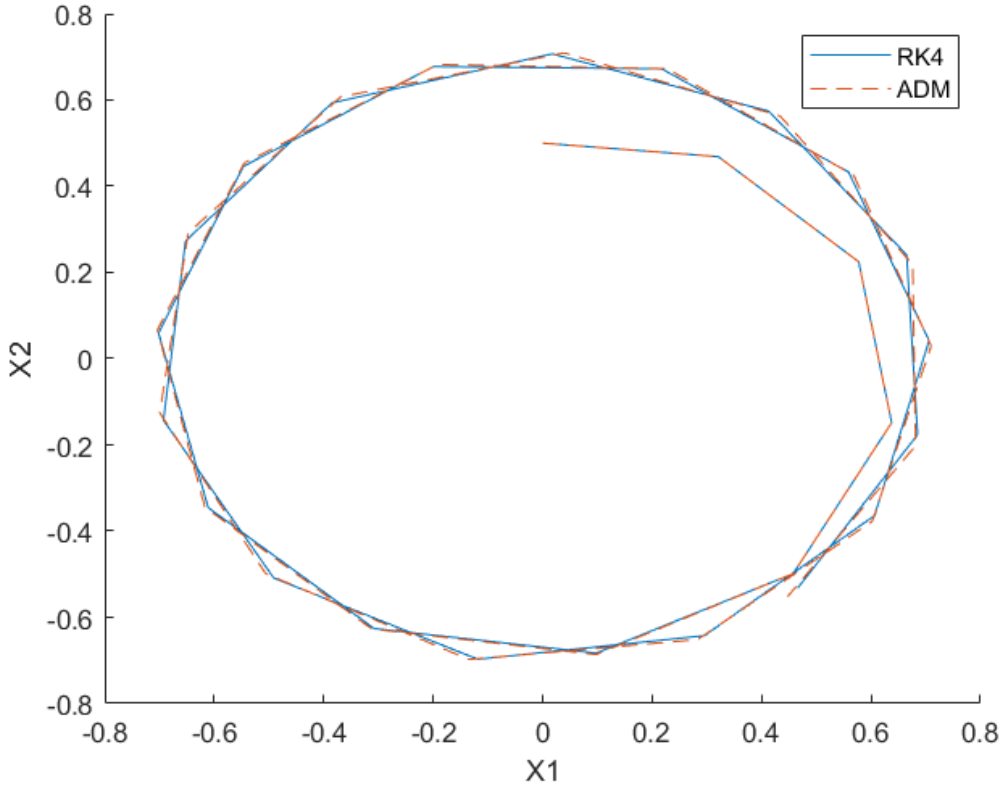
For h = 0.6:

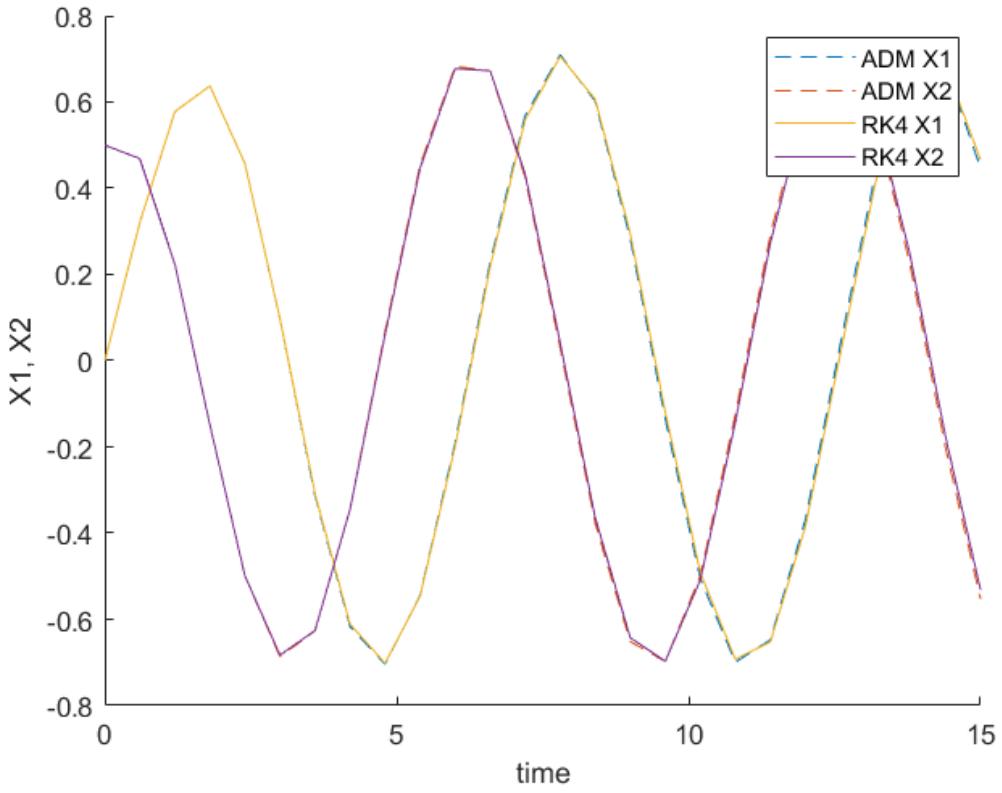


Comparison of Adams PC and RK4, for h = 0.2, and for h = 0.6, respectively:



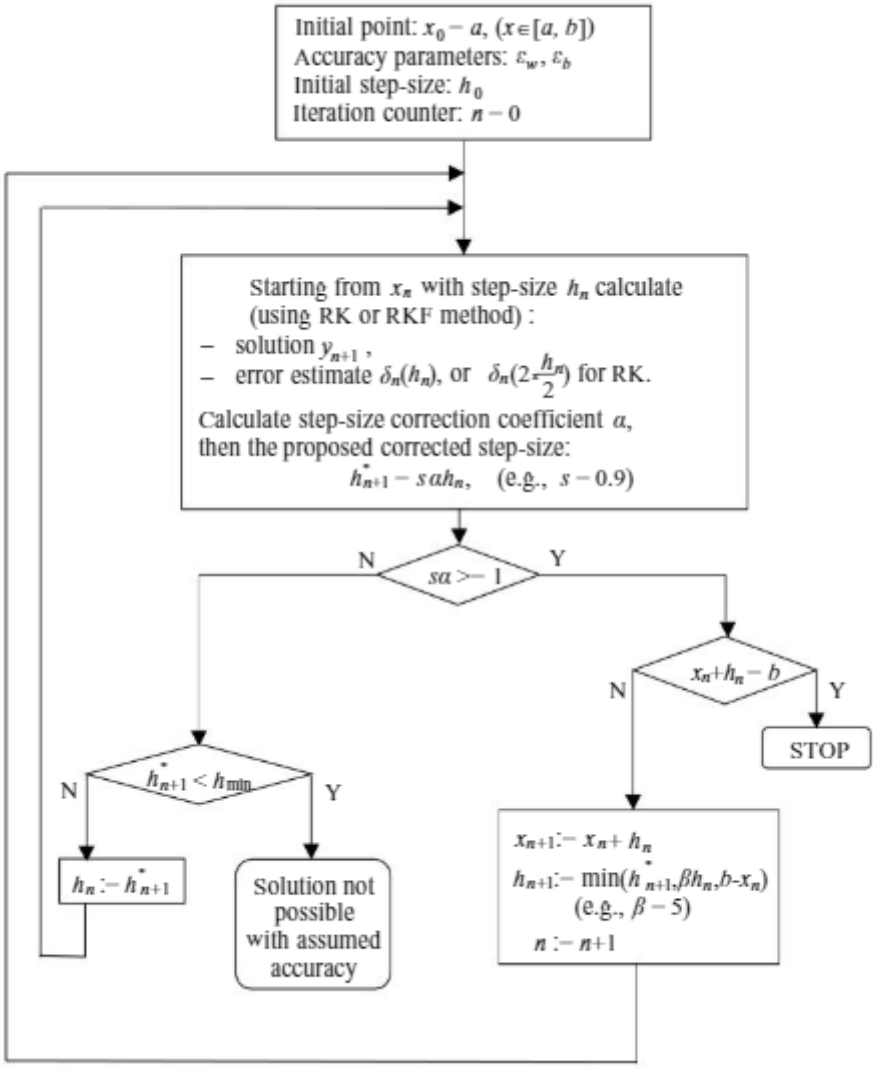



RK4
ADM
X2
X1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8


ADM X1
ADM X2
RK4 X1
RK4 X2
X1, X2
time
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
5
10
15

We can observe that for 'optimal' h there is no difference between plot of the RK4 or Adams PC method, however, when we increase h, RK4 method is becoming a little bit slower than Adams PC.

2b) Correction of the step-size



The program was written according to provided above algorithm. Moreover those equations were used:

$$\delta_n(h)_i = \frac{(y_i)_n^{(2)} - (y_i)_n^{(1)}}{2^p - 1}, \qquad i = 1, 2, ..., m,$$

$$\varepsilon_i = \left|(y_i)_n^{(2)}\right| \cdot \varepsilon_r + \varepsilon_a,$$

$$\alpha = \min_{1 \le i \le m} \left(\frac{\varepsilon_i}{|\delta_n(h)_i|}\right)^{\frac{1}{p+1}}.$$

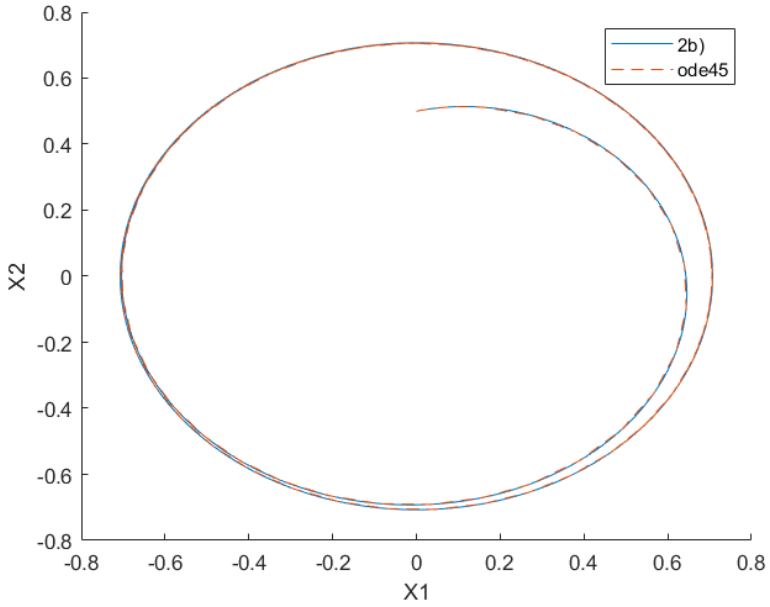
ε_r – a relative tolerance,

ε_a – an absolute tolerance.

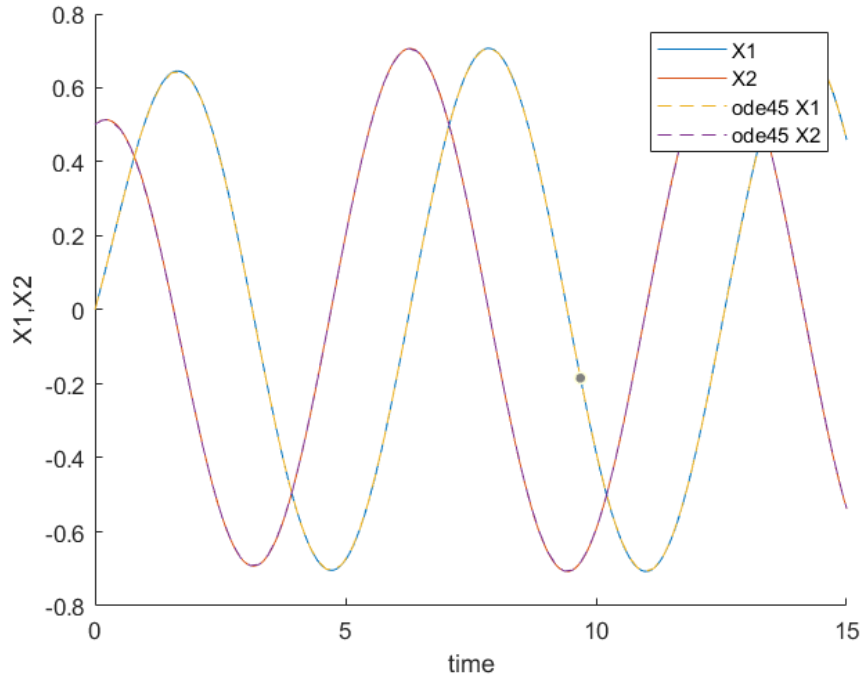
$h_{n+1} = s \cdot \alpha \cdot h_n$, where $s < 1$ (for RK4, RKF45: $s \approx 0.9$).

Results:

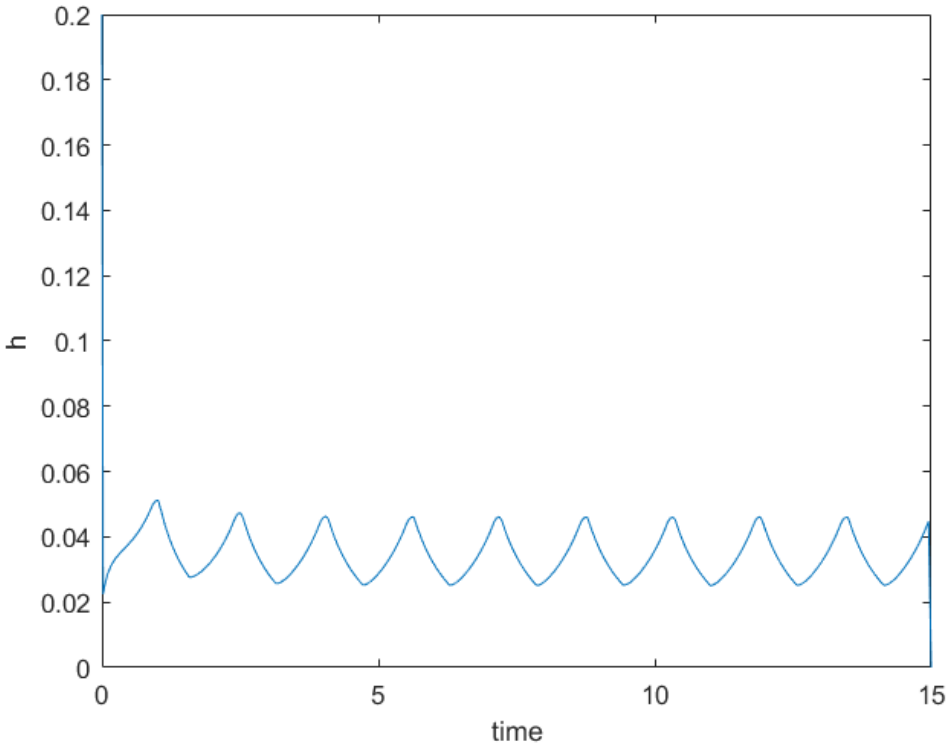
X1 versus X2 plot:



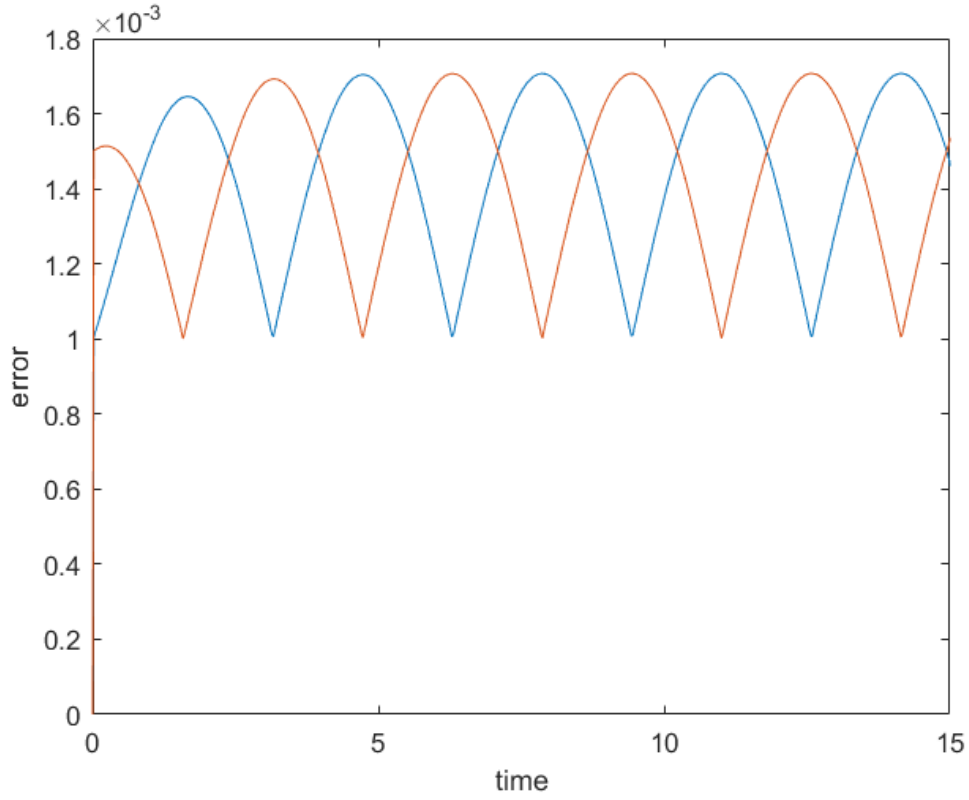
X1,X2 and Ode45 versus time:



Change of step ‘h’ versus time:



Change of error versus time:



CODE:

Task no.1 :

```
clear;
    x = [-5 -4 -3 -2 -1 0 1 2 3 4 5]';
    y = [-1.6889 -4.7689 -3.8259 -2.0068 -0.6884 ...
        0.9391 -1.1556 -4.4293 -12.5746 -25.6768 -45.0598]';
    [x y]


    plot(x, y,"o")
    xlabel('X');
    ylabel('Y');


    for n = 1:9

        A = zeros(length(x), n+1);


        for i = 1:size(A,2)

            A(:,i) = x.^(i-1);

        end
        A
        disp("***NEW ITERATION***");
        n
        condA = cond(A)
        condG = condA^2
        G = A'*A;
        true_condG = cond(G) %checking with previous postulate
                        % that condA^2 = condG
        [Q, R] = qrs(A);
        a = (R^(-1))*(Q'*y);
        a
        r = norm(A*a - y);
        r
        vec_r(n) = r;
        vec_n(n) = n;

        figure;
        plot(x,y,'o');
        hold on;
        plot(x,polyval(flipud(a),x),"LineStyle","-.")
        xlabel('X');
        ylabel('Y');
        legend('Data','Approximation');

    end

    figure;
```

```
    disp("Error to polynomial degree:");
    plot(vec_n, vec_r, "o","LineStyle","-.")
    xlabel('Polynomial degree');
    ylabel('Error');
```

```
function [Q, R] = qrs(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
         dd=norm(Q(:,i));
         Q(:,i)=Q(:,i)/dd;
         R(i,i:n)=R(i,i:n)*dd;
     end
end
```

Task no.2 :

```
clear;
```

RK4

```


    t = [0 15];
    x0 = [0 0.5];
    h = 0.2; %h = 0.4
    t_rk4 = (t(1):h:t(2))';
    x_rk4 = zeros(length(t_rk4),2);
    x_rk4(1,:) = x0;

    for n = 2:length(t_rk4)

        k1 = odef(x_rk4(n-1,:));
        k2 = odef(x_rk4(n-1,:)+.5*h*k1);
        k3 = odef(x_rk4(n-1,:)+.5*h*k2);
        k4 = odef(x_rk4(n-1,:)+h*k3);
```

```
    x_rk4(n,:) = x_rk4(n-1,:)+1/6*h*(k1+2*k2+2*k3+k4);

end


[t_ode45,x_ode45] = ode45(@(t,x) odef(x)', t, x0);

figure;
plot(x_rk4(:,1),x_rk4(:,2), "o");
hold on;
plot(x_ode45(:,1),x_ode45(:,2),'--');
xlabel('X1');
ylabel('X2');
plot(x_rk4(:,1),x_rk4(:,2));
xlabel('X1');
ylabel('X2');
legend("RK4 points", "ode45", "RK4 lines");


figure;
hold on;
plot(t_rk4,x_rk4(:,1));
plot(t_rk4,x_rk4(:,2));
plot(t_ode45,x_ode45(:,1),'--');
plot(t_ode45,x_ode45(:,2),'--');
xlabel('time');
ylabel('X1,X2');
legend("X1", "X2", "ode45 X1", "ode45 X2");
```

Adams

```
s = 5;
h = 0.2;
t_adm = (t(1):h:t(2))';
x_adm = zeros(length(t_adm),2);
f = zeros(length(t_adm),2);
x_adm(1,:) = x0;

for n = 2:s

    k1 = odef(x_adm(n-1,:));
    k2 = odef(x_adm(n-1,:)+.5*h*k1);
    k3 = odef(x_adm(n-1,:)+.5*h*k2);
    k4 = odef(x_adm(n-1,:)+h*k3);
    x_adm(n,:) = x_adm(n-1,:)+1/6*h*(k1+2*k2+2*k3+k4);

end

for i = 1:s
```

```
    f(i,:) = odef(x_adm(i,:));

end

for n = s+1:length(t_adm)

    x_ab = ab_method(x_adm(n-1,:),f(1:n-1,:),h);
    f(n,:) = odef(x_ab);
    x_am = am_method(x_adm(n-1,:),f(1:n,:),h);
    f(n,:) = odef(x_am);
    x_adm(n,:) = x_am;

end

figure;
plot(x_adm(:,1),x_adm(:,2), "o");
hold on;
plot(x_ode45(:,1),x_ode45(:,2),'--',"Color",'r');
xlabel('X1');
ylabel('X2');
plot(x_adm(:,1),x_adm(:,2));
hold on;
xlabel('X1');
ylabel('X2');
legend("ADM points", "ode45", "ADM lines");

figure;
hold on;
plot(t_adm,x_adm(:,1));
plot(t_adm,x_adm(:,2));
plot(t_ode45,x_ode45(:,1),'--');
plot(t_ode45,x_ode45(:,2),'--');
xlabel('time');
ylabel('X1, X2');
legend("X1", "X2", "ode45 X1", "ode45 X2");


figure;
hold on;
plot(t_adm,x_adm(:,1),'--');
plot(t_adm,x_adm(:,2),'--');
plot(t_rk4,x_rk4(:,1));
plot(t_rk4,x_rk4(:,2));
xlabel('time');
ylabel('X1, X2');
legend("ADM X1", "ADM X2", "RK4 X1", "RK4 X2");

figure;
hold on;
plot(x_rk4(:,1),x_rk4(:,2));
plot(x_adm(:,1),x_adm(:,2),'--');
xlabel('X1');
ylabel('X2');
legend("RK4", "ADM");
```

```
%% RK4 2b)

t = [0 15];
x0 = [0 0.5];
p = 4; % order of RK, do not change
s = 0.9; % may be lowered, do not increase over 1
n = 1;
ew = 1e-3;
eb = 1e-3;
B = 4;
h0 = 0.02;
hn = h0;
h_max = 0.6;
h_min = 0.01;
t_temp = t(1);
xh = zeros(2);
delta=zeros(1,2);

while t_temp(n) ~= t(2)

    for hh = [hn,hn/2]

        k1=odef(x0(end,:));
        k2=odef(x0(end,:)+.5*hh*k1);
        k3=odef(x0(end,:)+.5*hh*k2);
        k4=odef(x0(end,:)+hh*k3);

        if hh==hn/2

            xh_temp=x0(end,:)+1/6*hh*(k1+2*k2+2*k3+k4);
            k1=odef(xh_temp);
            k2=odef(xh_temp+.5*hh*k1);
            k3=odef(xh_temp+.5*hh*k2);
            k4=odef(xh_temp+hh*k3);

        end
         xh(hn/hh,:)=x0(end,:)+1/6*hh*(k1+2*k2+2*k3+k4);
    end

    delta(n,:) = abs((xh(2,:) - xh(1,:))) / (2.^p - 1);
    e = abs(xh(2,:)) * ew + eb;
    e_vec(n+1,:) = e;
    a_temp = min(e./delta(n,:));
    a = a_temp.^(1/(p+1));
    hn1_star = s * a * hn;

    if s * a >= 1

    t_temp(n+1) = t_temp(n) + hn;
    x0(n+1,:) = xh(1,:);
    hn1 = min([hn1_star, B * hn, t(2) - t_temp(n+1)]);
    hn = hn1;
    h(n+1) = hn;
```

```
        n = n + 1;

        else
            if hn1_star < h_min

                disp('Obtaining a solution is not possible');
                break;

            else
            hn = hn1_star;
            end
        end
    end

    figure;
    hold on;
    plot(x0(:,1), x0(:,2));
    plot(x_ode45(:,1),x_ode45(:,2),'--');
    xlabel('X1');
    ylabel('X2');
    legend("2b)", "ode45");

    figure;
    hold on;
    plot(t_temp, x0(:,1));
    plot(t_temp, x0(:,2));
    plot(t_ode45, x_ode45(:,1),'--');
    plot(t_ode45, x_ode45(:,2),'--');
    xlabel('time');
    ylabel('X1,X2');
    legend("X1", "X2", "ode45 X1", "ode45 X2");

    figure;
    plot(t_temp, h);
    xlabel('time');
    ylabel('h');

    figure;
    plot(t_temp, e_vec);
    xlabel('time');
    ylabel('error');


function dx = odef(x)


 dx = zeros(1,2);
 dx(1) = x(2)+x(1)*(0.5-x(1).^2-x(2).^2);
 dx(2) = -x(1)+x(2)*(0.5-x(1).^2-x(2).^2);


end
```

```
function x_am = am_method(y, f, h)


x_am = y+h/720*(251*f(end,:)+646*f(end-1,:)-264*f(end-2,:)+106*f(end-3,:)-
19*f(end-4,:));


end
function x_ab = ab_method(y,f,h)


x_ab = y+h/720*(1901*f(end,:)-2774*f(end-1,:)+2616*f(end-2,:)-1274*f(end-
3,:)+251*f(end-4,:));


end
```