5. Let $u(x) = \frac{1}{x+1}$. Approximate u on [0,2] using n equally spaced subintervals for n=2,5,10 using

a) polynomial interpolation with the Vandermonde matrix and a polynomial of degree n

b) polynomial interpolation with the the Lagrange formula and a polynomial of degree n

Plot your results and discuss.

a) By using the Vandermonde martix

In order to find polynomial that pas throug all the point are given, we need to first set up a polynomial:

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_k x^k \qquad (4)$$

Then plug all the points given into equation (4)

Suppose the points are:

$(x_0, y_0)(x_2, y_2)(x_3, y_3)\ldots(x_k, y_k)$

As a result we will get:

$$\begin{cases} P(x_0) = a_0 + a_1 x_0 + a_2 {x_0}^2 + \ldots + a_k {x_0}^k = y_0 \\ P(x_1) = a_0 + a_1 x_1 + a_2 {x_1}^2 + \ldots + a_k {x_1}^k = y_1 \\ \vdots \\ P(x_k) = a_0 + a_1 x_k + a_2 {x_k}^2 + \ldots + a_k {x_k}^k = y_k \end{cases}$$

And if we write them as a matrix:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 & \cdots & x_0^k \\ 1 & x_1 & x_1^2 & x_1^3 & \cdots & x_1^k \\ 1 & x_2 & x_2^2 & x_2^3 & \cdots & x_2^k \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_k & x_k^2 & x_k^3 & \cdots & x_k^k \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_k \end{bmatrix} \qquad (5)$$

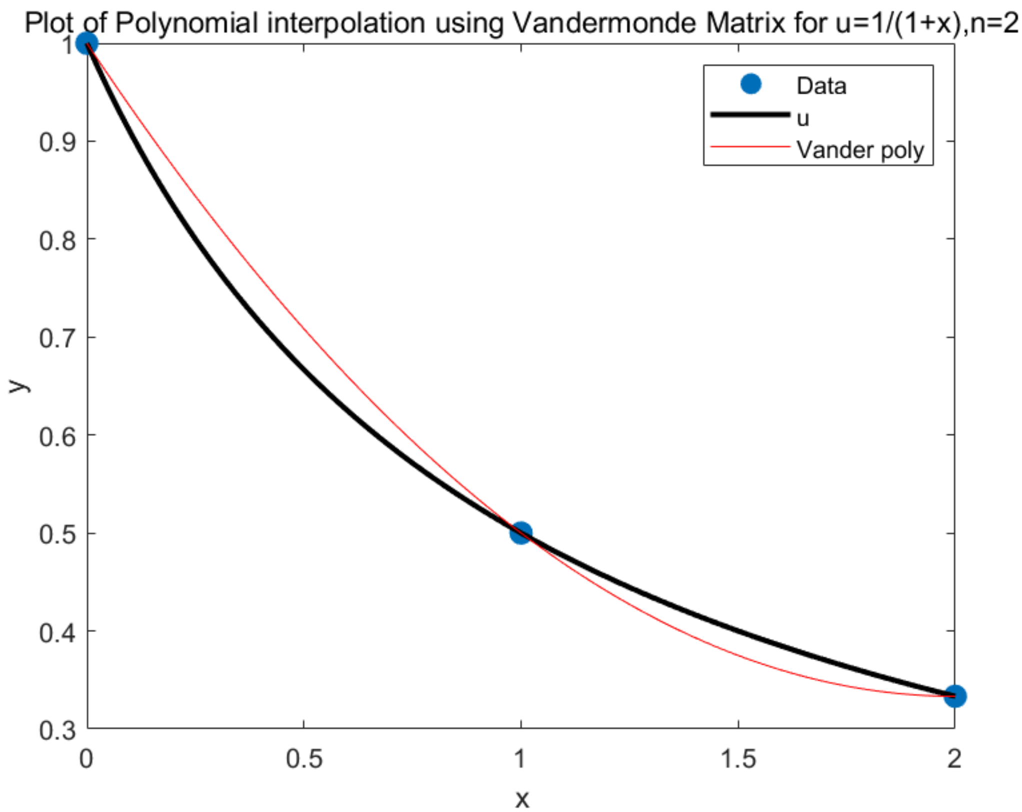
Let

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 & \cdots & x_0^k \\ 1 & x_1 & x_1^2 & x_1^3 & \cdots & x_1^k \\ 1 & x_2 & x_2^2 & x_2^3 & \cdots & x_2^k \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_k & x_k^2 & x_k^3 & \cdots & x_k^k \end{bmatrix}, \ a = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_k \end{bmatrix} \text{ and } y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_k \end{bmatrix}$$

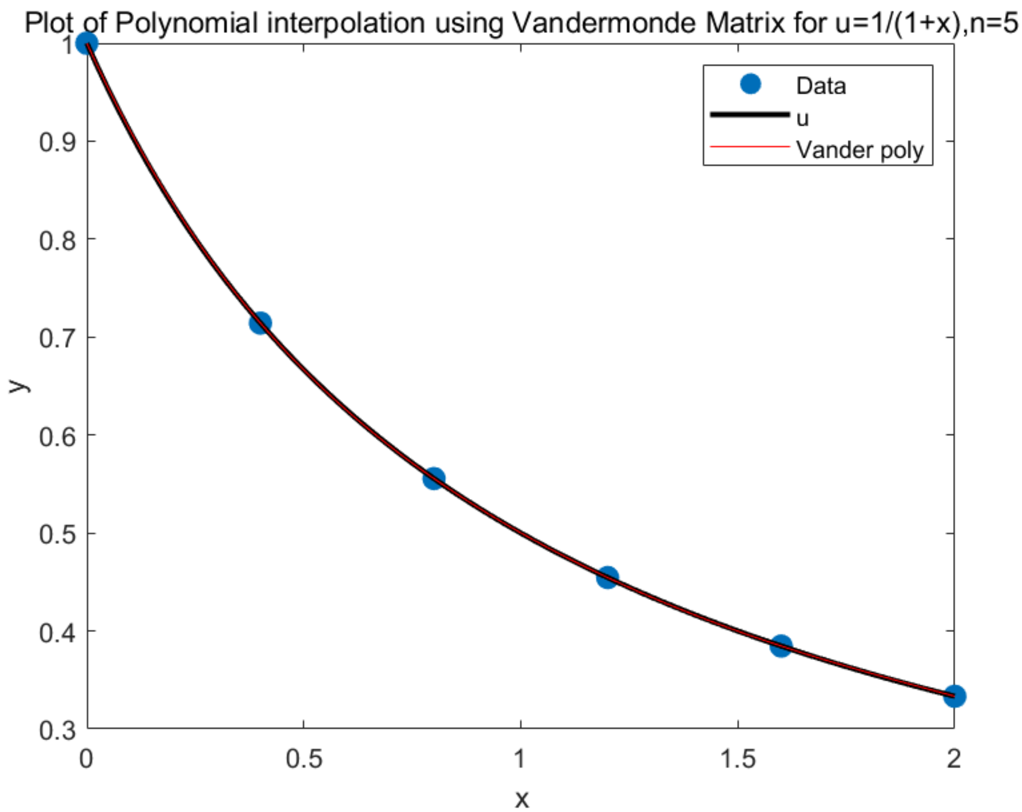
equation (5) can be written as $\mathrm{Va} = y$

V is called Vandermonde

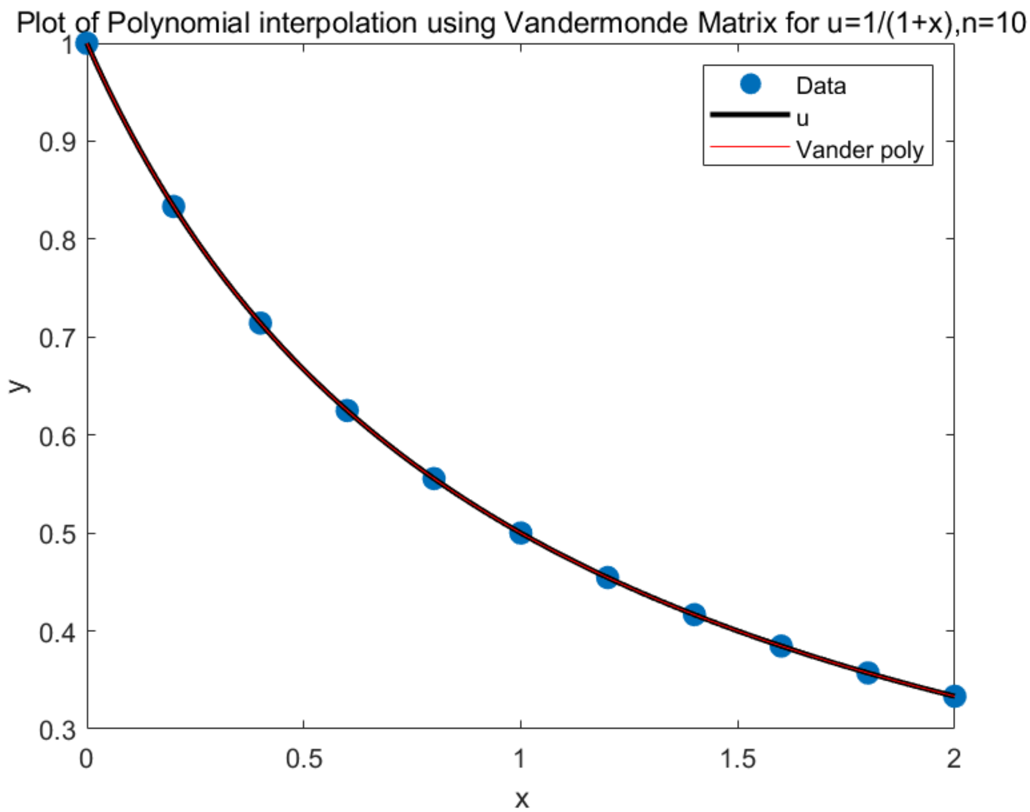
```
clear
%define the interval
a=0;
b=2;

%Vandermonde matrix
n = 2; % number of intervals
[x2V,y2V,xfine2V,u_Vander_fine2V,a_vec2V]=van(a,b,n);
plotA(x2V,y2V,xfine2V,u_Vander_fine2V,n)
```




```
n = 5; % number of intervals
[x5V,y5V,xfine5V,u_Vander_fine5V,a_vec5V]=van(a,b,n);
plotA(x5V,y5V,xfine5V,u_Vander_fine5V,n)
```




```
n = 10; % number of intervals
[x10V,y10V,xfine10V,u_Vander_fine10V,a_vec10V]=van(a,b,n);
plotA(x10V,y10V,xfine10V,u_Vander_fine10V,n)
```



From the graph above we could see as the interval point increasing the polynomial are almost become coincide with the original function u(x).

b)By using the Lagrange fomular

The Lagrange fomular is:

$$y_k = p(x_k) \qquad (6)$$

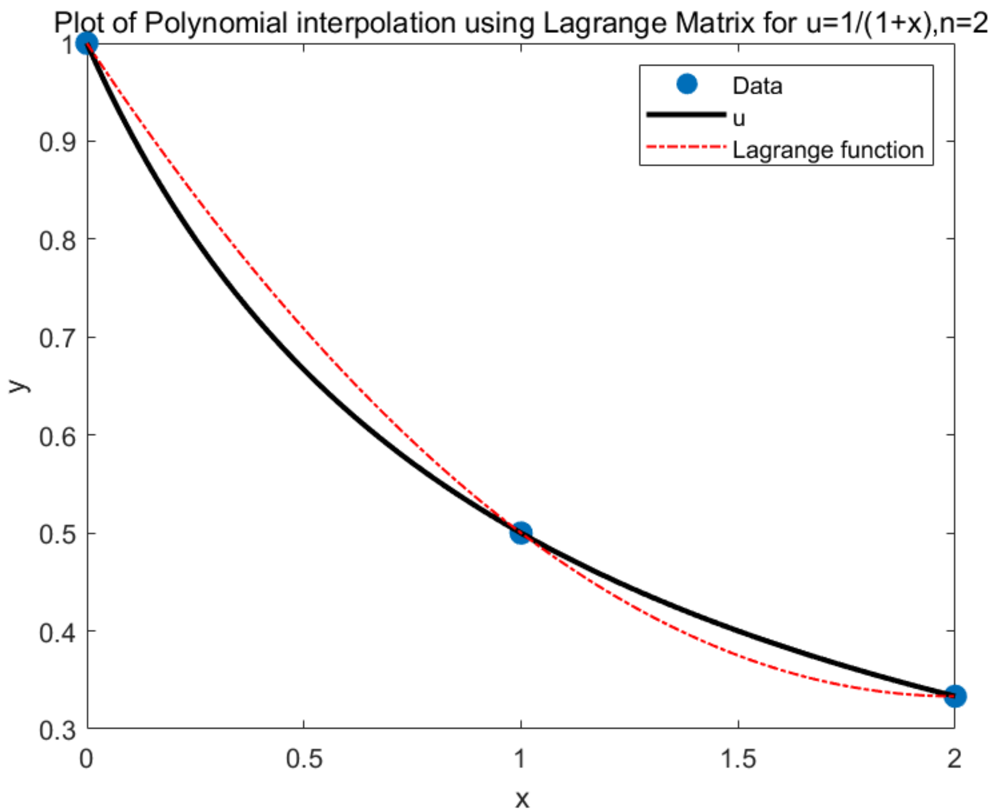
$$p(x_k) = y_0\ell_0(x) + y_1\ell_1(x) + \cdots + y_n\ell_n(x) \qquad (7)$$

$$\ell_0(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}$$

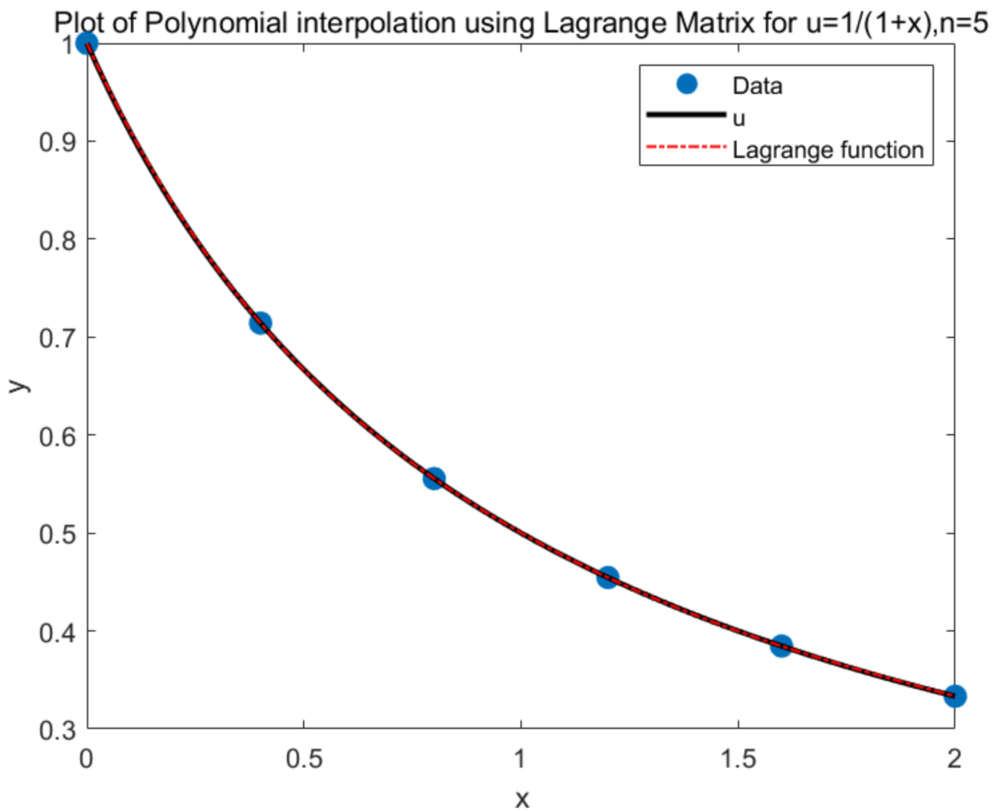
where can be write as

$$\ell_j(x) = \prod_{k=0,k\neq j}^{n} \frac{(x-x_k)}{(x_j-x_k)} \qquad (8)$$

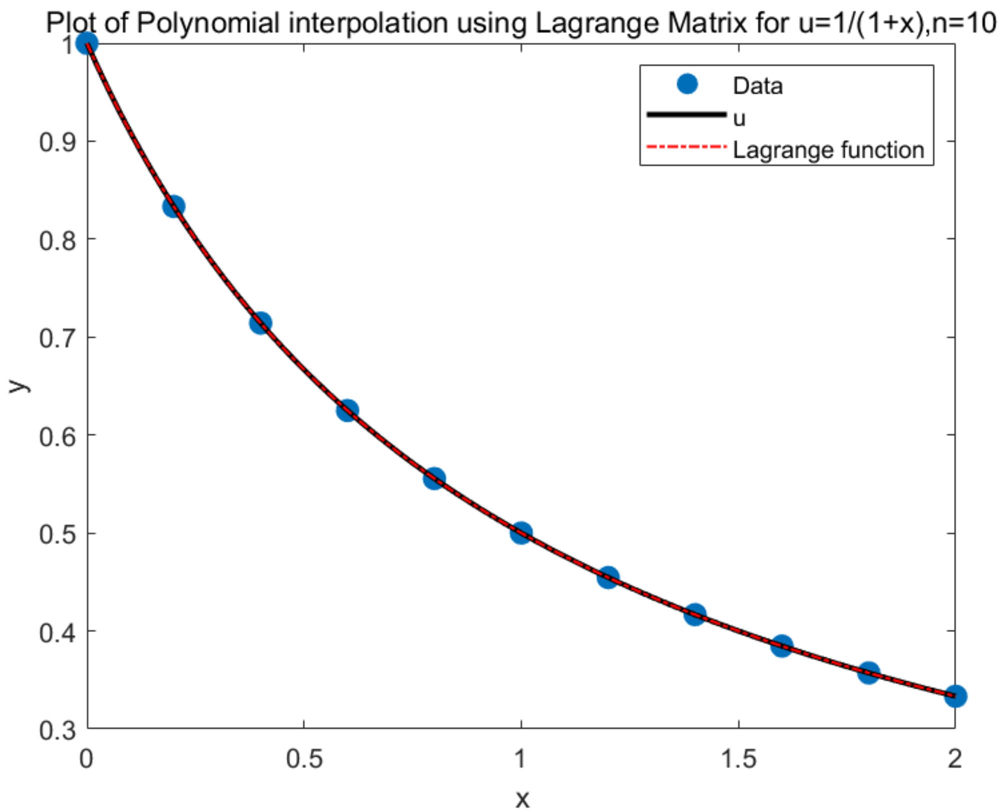
```
%Lagrange fomular
n = 2; % number of intervals
[x2L,y2L,xfine2L,u_Lagrange2L]=lan(a,b,n);
plotAN(x2L,y2L,xfine2L,u_Lagrange2L,n)
```




```
n = 5; % number of intervals
[x5L,y5L,xfine5L,u_Lagrange5L]=lan(a,b,n);
plotAN(x5L,y5L,xfine5L,u_Lagrange5L,n)
```




```
n = 10; % number of intervals
[x10L,y10L,xfine10L,u_Lagrange10L]=lan(a,b,n);
plotAN(x10L,y10L,xfine10L,u_Lagrange10L,n)
```



From the graph above we could see as that the polynomial compute from Lagrange method is also looks like coinside with the true value;

By compute the error of them

```
err_Vandermonde_n_2 = sum(abs(u_Vander_fine2V-u(xfine2V)),'all')
```

```
err_Vandermonde_n_2 = 4.5646
```

```
err_Vandermonde_n_5 = sum(abs(u_Vander_fine5V-u(xfine5V)),'all')
```

```
err_Vandermonde_n_5 = 0.0643
```

```
err_Vandermonde_n_10 = sum(abs(u_Vander_fine10V-u(xfine10V)),'all')
```

```
err_Vandermonde_n_10 = 1.3101e-04
```

```
err_Lagrange_n_2 = sum(abs(u_Lagrange2L-u(xfine2L)),'all')
```

```
err_Lagrange_n_2 = 4.5646
```

```
err_Lagrange_n_5 = sum(abs(u_Lagrange5L-u(xfine5L)),'all')
```

```
err_Lagrange_n_5 = 0.0643
```

```
err_Lagrange_n_10 = sum(abs(u_Lagrange10L-u(xfine10L)),'all')
```

```
err_Lagrange_n_10 = 1.3101e-04
```

As result, we could find that, as n increading the error of both approximation decreased.

```
function [x,y,xfine,u_Vander_fine,a_vec]=van(a,b,n)
%function of computing the Vandermonde martix and correspounding
%coefficients
%take input a,b,and n
%out put x: the x coodinate of data point according to n
%out put y: the y coodinate of data point that compute from x
%xfine: a finer grid of x
%u_Vander_fine: approximate the polynomial using the coefficient compute from V and y
%a_vec: vector a
    h = (b-a)/n;
    x = a + (0:n)*h; % define the x values
    y = u(x); % define the corresponding y values as y_j = u(x_j)
    Vmat = zeros(length(x));
    for ii=1:length(x)
        % loop over the data points
        for jj=1:length(x)
            % loop over the columns of the Vandermonde martix
            Vmat(ii,jj) = x(ii)^(jj-1);
            % since matlab indices starts at 1, so we need to subtract 1
        end
    end
    rr_form     = rref([Vmat,y(:)]);
    a_vec       = rr_form(:,end); % peeling off the last column
    xfine         = a:0.01:b;%define a finer grid x
    u_Vander_fine = a_vec(1)*ones(size(xfine));
    for jj=1:n
        %compute the polynomial from vector a
        u_Vander_fine = u_Vander_fine + a_vec(jj+1)*xfine.^(jj);
    end
end



function [x,y,xfine,u_Lagrange]=lan(a,b,n)
%function of compute the Lagrange fomular
%input a, b and n
%outpout
%out put x: the x coodinate of data point according to n
%out put y: the y coodinate of data point that compute from x
%xfine: a finer grid of x
%u_Lagrange:  approximate the polynomial
% using the coefficient compute Lagrange fomular
    h = (b-a)/n;
    x = a + (0:n)*h; % define the x values
    y = u(x); % define the corresponding y values as y_j = u(x_j)
    N          = length(x);
    xfine       = a:0.01:b; % define the finer x values
    u_Lagrange  = zeros(size(xfine));
    for k = 1:N
        %loop each k from equation (6)
```

```
        w = ones(size(xfine));
        for j = [1:k-1 k+1:N] % This loop the index from equation (8)
            w = (xfine-x(j))./(x(k)-x(j)).*w;
        end
        u_Lagrange = u_Lagrange + w*y(k);
        % This adds up each y_k*l_k(x) term, equation (7)
    end
end

function []=plotA(x,y,xfine,u_Vander_fine,n)
%function of plot the polynomial from Vandermonde martix
%take input:
%x: the x coodinate of data point according to n
%y: the y coodinate of data point that compute from x
%xfine: a finer grid of x
%u_Vander_fine: approximate the polynomial using the coefficient compute from V and y
%n
    plot(x,y,'.','MarkerSize', 30)%plot the data
    hold on;
    plot(xfine,u(xfine),'k', 'LineWidth',2)%plot the true value
    plot(xfine, u_Vander_fine, 'r')%plot the polynomial
    xlabel('x');ylabel('y');
    legend('Data','u', 'Vander poly')
    title(['Plot of Polynomial interpolation using ' ...
        'Vandermonde Matrix for u=1/(1+x),n=',num2str(n)])
    hold off
end

function []=plotAN(x,y,xfine,u_Lagrange,n)
%function of plot the polynomial from Lagrange fomular
%take input:
%x: the x coodinate of data point according to n
%y: the y coodinate of data point that compute from x
%xfine: a finer grid of x
%u_Lagrange: approximate the polynomial using the coefficient compute Lagrange
%n
    plot(x,y,'.','MarkerSize',30);%plot the data
    hold on;
    plot(xfine,u(xfine),'k', 'LineWidth',2);%plot the true value
    plot(xfine,u_Lagrange,'-.r','LineWidth',1)%plot the polynomial
    xlabel('x');ylabel('y');
    legend('Data','u','Lagrange function')
    title(['Plot of Polynomial interpolation using' ...
        ' Lagrange Matrix for u=1/(1+x),n=',num2str(n)])
    hold off
end

function [ut]=u(x)
%true function
    ut=1./(x+1);
end
```