```
%{
    ENG 180 Project 3
    File:           ENG180_PJ3_main_Hieu_Bui.m
    Author:         Hieu Bui
    Date:           10/12/2022
    Description:    Finished problem 1 and 3 for project 3

%}


clear, clc; %clear all variables in work place and command window
close all;  %close all figures

%====Create grid====%
kstr = 1.1;
delta = .2573;
n = 11;
x = ones(n,1);
x(6) = 0;
for i = 6:n-1
    x(i+1) = x(i) + delta*kstr^(i-5); %stretch from 0 to positive side
    x(n-i) = -1.*(x(i) + delta*kstr^(i-5));
end

%Given weighted matrices for P3
%{
    first number indicates which given weighted matrix
    second and third number indicates size (IE 2x2)
%}
w122 = weightedM(2,2,-1);
w133 = weightedM(3,2,-1);
w144 = weightedM(4,2,-1);
w222 = weightedM(2,4/6,1/6);
w233 = weightedM(3,4/6,1/6);
w244 = weightedM(4,4/6,1/6);

%Given function
F = tanh(x)+sin(x);
G = cosh(x)+cos(x);

%Results for crammer least square method
yf = linearLeastSqr(x,F,n);
yg = linearLeastSqr(x,G,n);
yf2 = quadLeastSqr(x,F,n);
yg2 = quadLeastSqr(x,G,n);
yf3 = cubicLeastSqr(x,F,n);
yg3 = cubicLeastSqr(x,G,n);

%Results for LU least square method
yluf1 = leastSqrLU(x,F,n,1);
yluf2 = leastSqrLU(x,F,n,2);
yluf3 = leastSqrLU(x,F,n,3);
```

```
ylug1 = leastSqrLU(x,G,n,1);
ylug2 = leastSqrLU(x,G,n,2);
ylug3 = leastSqrLU(x,G,n,3);

%Results for weighted variants
%first weighted matrix
ywf1 = leastSqrWeight(x,F,n,1,w122);
ywf2 = leastSqrWeight(x,F,n,2,w133);
ywf3 = leastSqrWeight(x,F,n,3,w144);
ywg1 = leastSqrWeight(x,G,n,1,w122);
ywg2 = leastSqrWeight(x,G,n,2,w133);
ywg3 = leastSqrWeight(x,G,n,3,w144);
%second weighted matrix
yw2f1 = leastSqrWeight(x,F,n,1,w222);
yw2f2 = leastSqrWeight(x,F,n,2,w233);
yw2f3 = leastSqrWeight(x,F,n,3,w244);
yw2g1 = leastSqrWeight(x,G,n,1,w222);
yw2g2 = leastSqrWeight(x,G,n,2,w233);
yw2g3 = leastSqrWeight(x,G,n,3,w244);

%Plotting Problem 1 for F (P1F)
figure(1);
subplot(2,2,1);
plot(x,F,'k',x,F,'bo',x,yf,'--b',x,yf2,'r--',x,yf3,'g--')
legend('Exact','Data','Linear','Quadratic','Cubic','Location','southeast')
hold on;grid on;
title('PJ3 Problem 1 F function')
xlabel('x')
ylabel('y','Rotation',0)

subplot(2,2,2);
plot(x,G,'k',x,G,'bo',x,yg,'--b',x,yg2,'r*',x,yg3,'g--')
legend('Exact','Data','Linear','Quadratic','Cubic','Location','north')
hold on;grid on;
title('PJ3 Problem 1 G function')
xlabel('x')
ylabel('y','Rotation',0)

subplot(2,2,3)
plot(x,F,'k',x,F,'bo',x,yluf1,'--b',x,yluf2,'r--',x,yluf3,'g--')
legend('Exact','Data','Linear','Quadratic','Cubic','Location','southeast')
hold on;grid on;
title('PJ3 Problem 1 F function using LU')
xlabel('x')
ylabel('y','Rotation',0)

subplot(2,2,4)
plot(x,G,'k',x,G,'bo',x,ylug1,'--b',x,ylug2,'r*',x,ylug3,'g--')
legend('Exact','Data','Linear','Quadratic','Cubic','Location','north')
hold on;grid on;
title('PJ3 Problem 1 G function using LU')
xlabel('x')
ylabel('y','Rotation',0)
```

```
%Plotting Problem 3 for different weighted
figure(2);
subplot(2,2,1);
plot(x,F,'k',x,F,'bo',x,ywf1,'--b',x,ywf2,'g*',x,ywf3,'m--')
legend('Exact','Data','Linear','Quadratic','Cubic','Location','southeast')
hold on;grid on;
title('PJ3 Problem 3 F function first weight')
xlabel('x')
ylabel('y','Rotation',0)

subplot(2,2,2);
plot(x,G,'k',x,G,'bo',x,ywg1,'--b',x,ywg2,'r*',x,ywg3,'g--')
legend('Exact','Data','Linear','Quadratic','Cubic','Location','north')
hold on;grid on;
title('PJ3 Problem 3 G function first weight')
xlabel('x')
ylabel('y','Rotation',0)

subplot(2,2,3)
plot(x,F,'k',x,F,'bo',x,yw2f1,'--b',x,yw2f2,'r--',x,yw2f3,'g--')
legend('Exact','Data','Linear','Quadratic','Cubic','Location','southeast')
hold on;grid on;
title('PJ3 Problem 3 F function second weight')
xlabel('x')
ylabel('y','Rotation',0)

subplot(2,2,4)
plot(x,G,'k',x,G,'bo',x,yw2g1,'--b',x,yw2g2,'r*',x,yw2g3,'g--')
legend('Exact','Data','Linear','Quadratic','Cubic','Location','north')
hold on;grid on;
title('PJ3 Problem 3 G function second weight')
xlabel('x')
ylabel('y','Rotation',0)

%====Project 3 P1====%
%Performing linear least square fit with crammer
function y = linearLeastSqr(x,F,n)
    xsum = sum(x);
    xsum2 = sum(x.^2);
    ysum = sum(F);
    xysum = sum(x.*F);

    %Matrix form for linear least square
    A = [n, xsum; xsum, xsum2];
    b = [ysum; xysum];

    %Solving the matrix with cramer's rule supporting function
    a0 = cramer2(A,b,1);
    a1 = cramer2(A,b,2);

    %output y
    y = a0 + a1.*x;
end
%Performing quadratic least square fit with crammer
```

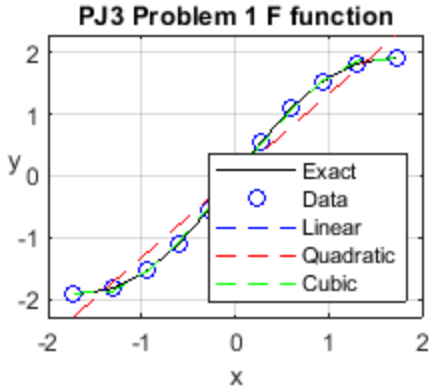
```
function y = quadLeastSqr(x,F,n)
    xsum = sum(x);
    xsum2 = sum(x.^2);
    xsum3 = sum(x.^3);
    xsum4 = sum(x.^4);
    ysum = sum(F);
    xysum = sum(x.*F);
    xysum2 = sum(x.^2.*F);

    %Matrix form for quadratic least square
    A = [n,xsum,xsum2;xsum,xsum2,xsum3;xsum2,xsum3,xsum4];
    b = [ysum;xysum;xysum2];

    %Solving for coefficients with crammer's rule
    a0 = cramer3(A,b,1);
    a1 = cramer3(A,b,2);
    a2 = cramer3(A,b,3);

    %output y
    y = a0 + a1.*x + a2.*x.^2;

end
%Performing cubic least square fit with crammer
function y = cubicLeastSqr(x,F,n)

    xsum = sum(x);
    xsum2 = sum(x.^2);
    xsum3 = sum(x.^3);
    xsum4 = sum(x.^4);
    xsum5 = sum(x.^5);
    xsum6 = sum(x.^6);
    ysum = sum(F);
    xysum = sum(x.*F);
    xysum2 = sum(x.^2.*F);
    xysum3 = sum(x.^3.*F);

    A = [n,xsum,xsum2,xsum3;xsum,xsum2,xsum3,xsum4;xsum2,xsum3,xsum4,xsum5...
        ;xsum3,xsum4,xsum5,xsum6];
    b = [ysum;xysum;xysum2;xysum3];

    a0 = cramer4(A,b,1);
    a1 = cramer4(A,b,2);
    a2 = cramer4(A,b,3);
    a3 = cramer4(A,b,4);

    %output y
    y = a0 + a1.*x + a2.*x.^2 +a3.*x.^3;

end

%Performing linear least square fit with LU decomp
%Parameter dim is the order of function (linear = 1, quad = 2, etc.)
function y = leastSqrLU(x,F,n,dim)
    xsum = sum(x);
```

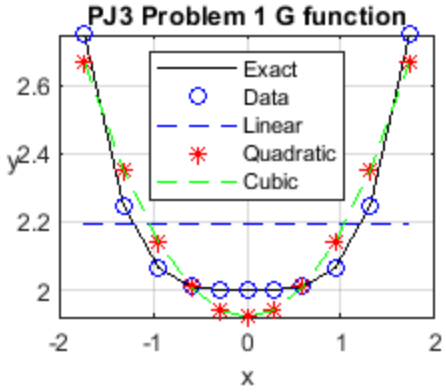
```
    xsum2 = sum(x.^2);
    xsum3 = sum(x.^3);
    xsum4 = sum(x.^4);
    xsum5 = sum(x.^5);
    xsum6 = sum(x.^6);
    ysum = sum(F);
    xysum = sum(x.*F);
    xysum2 = sum(x.^2.*F);
    xysum3 = sum(x.^3.*F);

    %Computes results based on the order (1 = linear; 2 = quad; 3 = cubic)
    if dim == 1
        A = [n, xsum; xsum, xsum2];
        b = [ysum; xysum];
        a = doolittle(A,b);
        y = a(1) + a(2).*x;
    elseif dim == 2
        A = [n,xsum,xsum2;xsum,xsum2,xsum3;xsum2,xsum3,xsum4];
        b = [ysum;xysum;xysum2];
        a = doolittle(A,b);
        y = a(1) + a(2).*x + a(3).*x.^2;
    else
        A =
 [n,xsum,xsum2,xsum3;xsum,xsum2,xsum3,xsum4;xsum2,xsum3,xsum4,xsum5...
            ;xsum3,xsum4,xsum5,xsum6];
        b = [ysum;xysum;xysum2;xysum3];
        a = doolittle(A,b);
        y = a(1) + a(2).*x + a(3).*x.^2 +a(4).*x.^3;
    end

end

%Supporting function perform LU decomposition to solve for x
function x = doolittle(A,b)
    dim = length(A);
    L = zeros(dim,dim);
    U = zeros(dim,dim);

    %Step 1
    %Set up initial conditions for L and U
    for i=1:dim
        U(1,i) = A(1,i);
        L(i,i) = 1;
    end

    for i=2:dim
        L(i,1) = A(i,1)/U(1,1);
    end

    %Solving for the rest of matrix elements
    for s = 2:dim
        i = s;
        for j = i:dim
            U(i,j) = A(i,j)-L(i,1:i-1)*U(1:i-1,j);
```

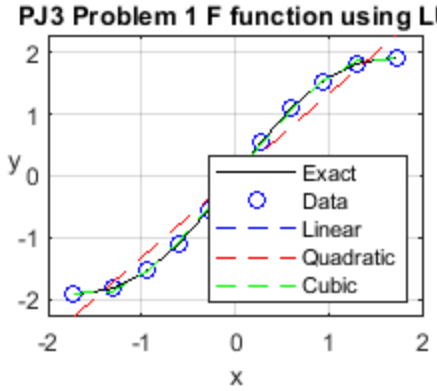
```
        end
        j = s;
        for i = j+1:dim
            L(i,j) = (A(i,j)-L(i,1:j-1)*U(1:j-1,j))/U(j,j);
        end
    end

    %Step 2 solving for y
    y = ones(dim,1);
    y(1) = b(1);
    for i=2:dim
           y(i) = b(i)-L(i,1:i-1)*y(1:i-1);
    end

    %Step 3 solving for x
    x = ones(dim,1);
    x(dim) = y(dim)/U(dim,dim);
    for i=dim-1:-1:1
            x(i) = (y(i)-U(i,i+1:dim)*x(i+1:dim))/U(i,i);
    end
end

%====Project 3 P3====%
% w is the weighted matrix
function y = leastSqrWeight(x,F,n,dim,w)
    xsum = sum(x);
    xsum2 = sum(x.^2);
    xsum3 = sum(x.^3);
    xsum4 = sum(x.^4);
    xsum5 = sum(x.^5);
    xsum6 = sum(x.^6);
    ysum = sum(F);
    xysum = sum(x.*F);
    xysum2 = sum(x.^2.*F);
    xysum3 = sum(x.^3.*F);

    [l,lt] = cholesky(w);

    %Computes results based on the order (1 = linear; 2 = quad; 3 = cubic)
    if dim == 1
        A = [n, xsum; xsum, xsum2];
        b = [ysum; xysum];
        abar = inv(lt)*A;
        bbar = inv(lt)*b;
        a = inv(abar'*abar)*abar'*bbar;
        y = a(1) + a(2).*x;
    elseif dim == 2
        A = [n,xsum,xsum2;xsum,xsum2,xsum3;xsum2,xsum3,xsum4];
        b = [ysum;xysum;xysum2];
        abar = inv(lt)*A;
        bbar = inv(lt)*b;
        a = inv(abar'*abar)*abar'*bbar;
        y = a(1) + a(2).*x + a(3).*x.^2;
    else
```

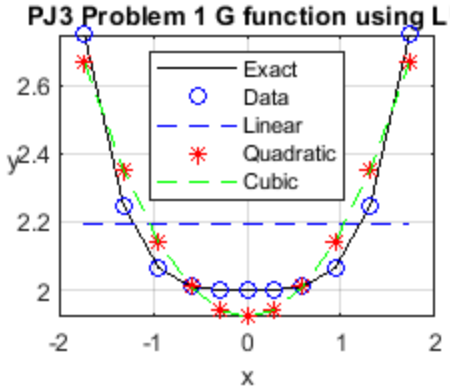
```
        A =
 [n,xsum,xsum2,xsum3;xsum,xsum2,xsum3,xsum4;xsum2,xsum3,xsum4,xsum5...
            ;xsum3,xsum4,xsum5,xsum6];
        b = [ysum;xysum;xysum2;xysum3];
        abar = inv(lt)*A;
        bbar = inv(lt)*b;
        a = inv(abar'*abar)*abar'*bbar;
        y = a(1) + a(2).*x + a(3).*x.^2 +a(4).*x.^3;
    end
end

function [L,LT] = cholesky(A)
    n = length(A);
    b = diag(A); %extract main diagonal
    c = diag(A,1); %extract diagonal one deviation from main

    beta = zeros(n,1);
    gamma = zeros(n-1,1);

    beta(1) = sqrt(b(1));
    gamma(1) = c(1)/beta(1);

    %finding beta
    for i=1:n-1
        beta(i+1) = sqrt(b(i+1)-gamma(i)^2);
        gamma(i) = c(i)/beta(i);
    end
    L = diag(beta)+diag(gamma,-1);
    LT = L';
end

%supporting function to create weighted matrix to size
function w = weightedM(n,main,second)
    w = zeros(n,n);
    a = second.*ones(n-1,1);
    b = main.*ones(n,1);
    w = diag(b)+diag(a,-1)+diag(a,1);
end

%supporting function that calculate determinant of 2x2 matrix
function determ = det2(a)
    determ = (a(1,1).*a(2,2))-(a(1,2).*a(2,1));
end

%supporting function that calculate determinant of 3x3 matrix
function determ = det3(a)
    determ = a(1,1)*det2([a(2,2),a(2,3);a(3,2),a(3,3)])...
            - a(1,2)*det2([a(2,1),a(2,3);a(3,1),a(3,3)])...
            + a(1,3)*det2([a(2,1),a(2,2);a(3,1),a(3,2)]);
end

%supporting function that calculate determinant of 4x4 matrix
function determ = det4(a)
```

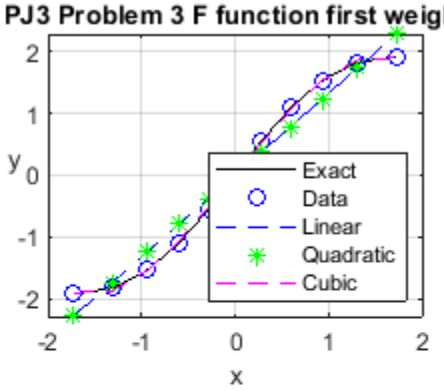
```
    determ =
 a(1,1)*det3([a(2,2),a(2,3),a(2,4);a(3,2),a(3,3),a(3,4);a(4,2),a(4,3),a(4,4)])...
             -
a(1,2)*det3([a(2,1),a(2,3),a(2,4);a(3,1),a(3,3),a(3,4);a(4,1),a(4,3),a(4,4)])...

 +a(1,3)*det3([a(2,1),a(2,2),a(2,4);a(3,1),a(3,2),a(3,4);a(4,1),a(4,2),a(4,4)])...
             -
a(1,4)*det3([a(2,1),a(2,2),a(2,3);a(3,1),a(3,2),a(3,3);a(4,1),a(4,2),a(4,3)]);
end

%Supporting function that perform cramer's rule for 2x2
function cram = cramer2(a,b,col)
    deterA = det2(a);
    ai = a;
    ai(:,col) = b;
    deterAI = det2(ai);
    cram = deterAI/deterA;
end

%Supporting function that perform cramer's rule for 3x3
function cram = cramer3(a,b,col)
    deterA = det3(a);
    ai = a;
    ai(:,col) = b;
    deterAI = det3(ai);
    cram = deterAI/deterA;
end

%Supporting function that perform cramer's rule for 4x4
function cram = cramer4(a,b,col)
    deterA = det4(a);
    ai = a;
    ai(:,col) = b;
    deterAI = det4(ai);
    cram = deterAI/deterA;
end
```

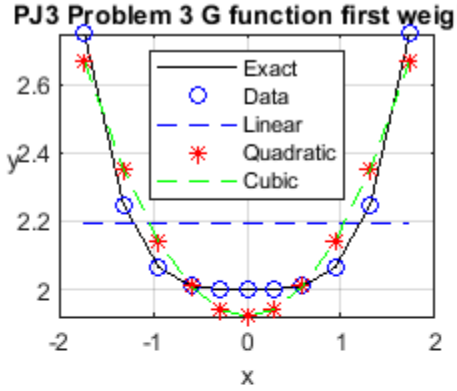

PJ3 Problem 1 F function
Exact
Data
Linear
Quadratic
Cubic
x
y

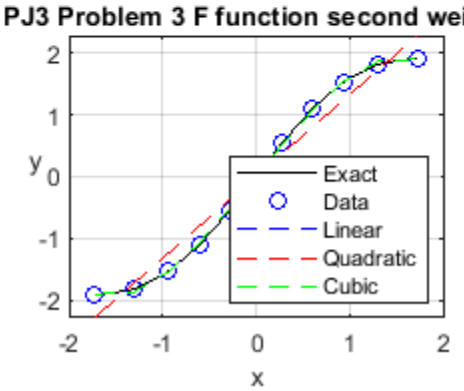

PJ3 Problem 1 G function
Exact
Data
Linear
Quadratic
Cubic
x
y

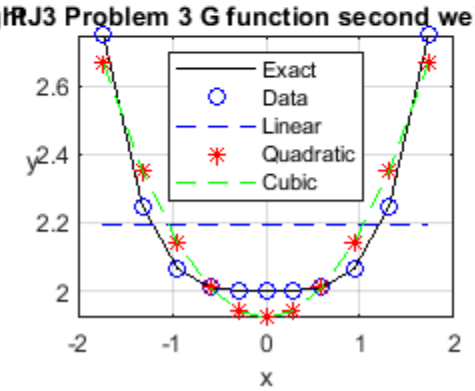

PJ3 Problem 1 F function using LU
Exact
Data
Linear
Quadratic
Cubic
x
y


PJ3 Problem 1 G function using LU
Exact
Data
Linear
Quadratic
Cubic
x
y


PJ3 Problem 3 F function first weight
Exact
Data
Linear
Quadratic
Cubic
x
y


PJ3 Problem 3 G function first weight
Exact
Data
Linear
Quadratic
Cubic
x
y


PJ3 Problem 3 F function second weight
Exact
Data
Linear
Quadratic
Cubic
x
y


PJ3 Problem 3 G function second weight
Exact
Data
Linear
Quadratic
Cubic
x
y

Published with MATLAB® R2022b