Solution

전기공학전공 홍종혁

Problem 1 – Approximation of Datas in Two Dimension

(1) Least Square Method

전체 코드

```
clear       %모든 변수 초기화
clc         %명령 창 초기화
close all;  %모든 figure 닫음
load census %census 불러오기

a=[1 2 4];  %order 수를 저장한 행렬
for i=1:3   %i가 1에서 3이 될 때까지 반복하는 for문
    least_square(a(i),500*(i-1),cdate,pop);    %least_square 함수에 a(i), 500*(i-1), cdate, pop 입력
end         %for문 종료

function least_square(n,p,cdate,pop)        %n,p,cdate,pop을 입력받는 함수를 least_square 라는 이름으로 선언
figure('Position', [p 200 560 420])         %Figure의 위치와 크기 지정
plot(cdate,pop,'o')                         %cdate를 x축, pop을 y축으로 하여 o 모양으로 plot
hold on                                     %그래프 유지
titlenum=num2str(n);                        %입력 받은 차수를 string 형태로 바꿈
title("Least Square method when order is "+titlenum)    %제목 출력
mat1=zeros(n+1,n+1);                        %mat1을 n+1 by n+1 size로 선언
mat2=zeros(n+1,1);                          %mat2를 n+1 by 1 size로 선언
for i=1:n+1                                 %i가 1에서 n+1이 될때까지 반복하는 for문 선언
    mat2(i)=sum((cdate.^(n+1-i)).*pop);     %mat2의 i번째 행에 cdate의 n+1-j제곱과 pop의 곱을 전부 합한 값을 저장
    for j=1:n+1                             %j가 1에서 n+1이 될 때까지 반복하는 for문 선언
        mat1(i,j)=sum(cdate.^(2*(n+1)-i-j));%mat1(i,j)에 cdate의 2*(n+1)-i-j제곱을 전부 합한 값을 저장
    end                                     %j를 사용한 for문 종료
end                                         %i를 사용한 for문 종료
resmat=mat1\mat2;                           %inverse(mat1)*mat2를 수행한 값을 resmat에 저장
y=0;                                        %변수 y 선ㄴ언
for k=1:n+1                                 %k가 1에서 n+1이 될 때까지 반복하는 for문 선언
    y=y+resmat(k)*(cdate.^(n+1-k));         %y에 resmat(k)에 cdate를 n+1-k 제곱한 것을 곱한 값을 더해줌
end                                         %k를 사용한 for문 종료
plot(cdate,y,"LineWidth",1.5)               %x를 x축으로, y를 y축으로, 선 굵기를 1.5으로 plot
hold on                                     %그래프 유지
end                                         %function 마침
```

알고리즘 설명

```
function least_square(n,p,cdate,pop)

figure('Position', [p 200 560 420])
plot(cdate,pop,'o')
hold on
titlenum=num2str(n);
title("Least Square method when order is "+titlenum)

mat1=zeros(n+1,n+1);
mat2=zeros(n+1,1);

for i=1:n+1
    mat2(i)=sum((cdate.^(n+1-i)).pop);
        for j=1:n+1
            mat1(i,j)=sum(cdate.^(2(n+1)-i-j));
        end
end
```

Least Square Method 를 수행하는 함수를 선언했다. 그 후에는 원본 데이터를 plot 하였다.
$y = Ax + B$에서 A 에 해당하는 mat1 을, B 에 해당하는 mat2 를 각각 선언해준 후, for 문을 이용하여 데이터 값을 저장하였다. 일반화를 통해서, n 값에 대응하여 nth order least square method 를 수행할 수 있게끔 작성하였다.

```
resmat=mat1\mat2;
y=0;
for k=1:n+1
    y=y+resmat(k)*(cdate.^(n+1-k));
end
plot(cdate,y,"LineWidth",1.5)
hold on
end
end
```

그 후 우리가 원하는 A 와 B 를 구하기 위해 mat1 의 inverse matrix 와 mat2 를 곱해준 후,
$y = Ax + B$식을 만들어 데이터 값을 저장하도록 해준다. 그 후 Approximation 값을 plot 하였다.

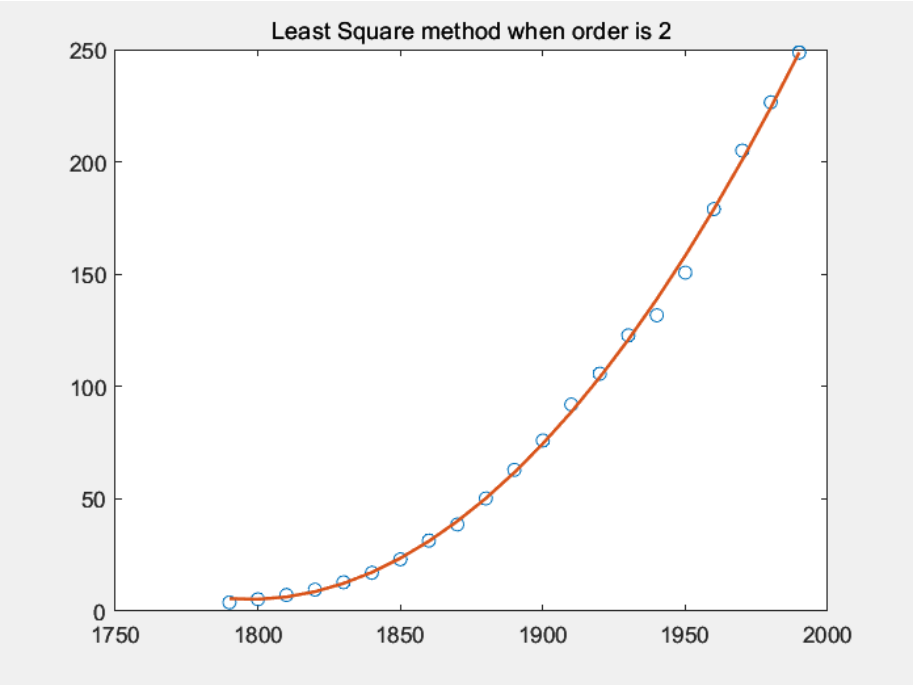
```
clear
clc
close all;
load census

a=[1 2 4];
for i=1:3
    least_square(a(i),500*(i-1),cdate,pop);
end
```

함수를 실행한 부분의 코드이다. a 행렬은 First Order, Second Order, Fourth Order 를 수행하기 위한 값들이 저장되어 있고, for 문을 이용하여 함수를 실행하였다.

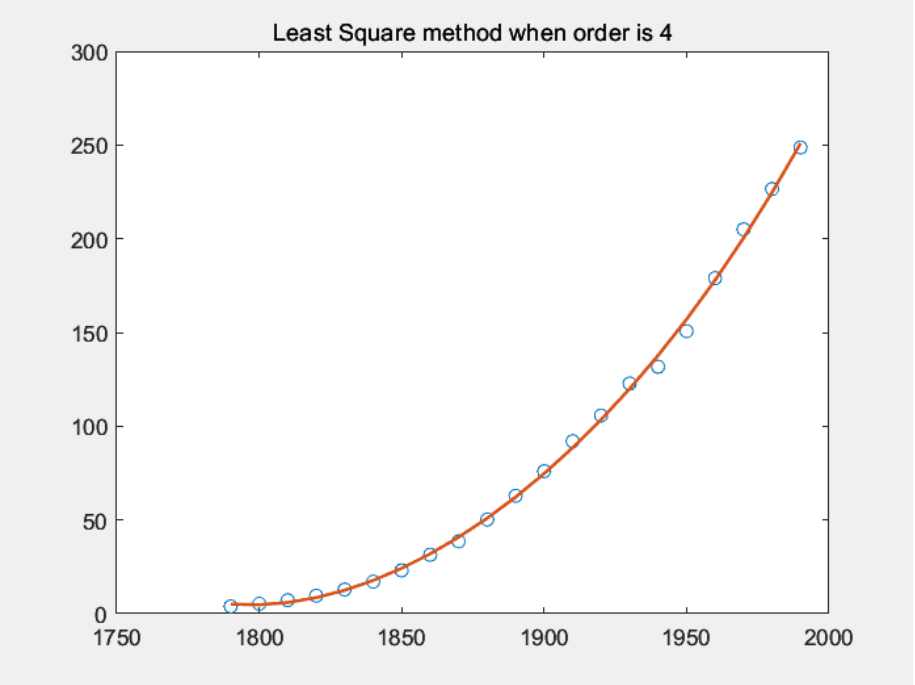
각각의 결과는 다음과 같았다.

- First Order Least Square

- Second Order Least Square



- Fourth Order Least Square



(2) Spline Method

전체 코드

```
clear       %모든 변수 초기화
clc         %명령 창 초기화
close all;  %모든 figure 닫음
load census %census 불러오기

spline_first(cdate,pop)     %spline_first 함수 실행
spline_second(cdate,pop)    %spline_second 함수 실행

function spline_first(cdate,pop)        %spline_first 함수 선언
figure('Position', [200 200 560 420])   %figure 생성 관련 세팅
plot(cdate,pop,'o')                     %원본 데이터 plot
title("First Order Spline Fit")         %제목 설정
hold on                                 %그래프 유지
temp_y=zeros(1,length(cdate)-1);        %y값을 담아둘 행렬 생성
m=zeros(1,length(cdate));               %기울기 값을 저장할 행렬 생성
for j=1:length(cdate)-1                 %j가 1에서 length(cdate)-1이 될때까지 반복
    m(j)=((pop(j+1)-(pop(j)))/((cdate(j+1))-(cdate(j)))); %pop변화값을 cdate 변화값으로 나눈 값을 m의 j 열에 저장
    x=linspace(cdate(j),cdate(j+1),length(cdate)-1);    %cdate(j)와 cdate(j+1)사이를 length(cdate)-1등분 한 x 선언
    for k=1:length(cdate)-1     %k가 1에서 length(cdate)-1이 될때까지 반복
        temp_y(k)=m(j)*(x(k)-cdate(j))+pop(j);  %temp_y의 k번째 열에 해당하는 범위에 spline을 시행한 값을 저장
    end                                 %반복문 종료
    plot(x,temp_y,"LineWidth",1.5)      %해당 그래프 plot
    hold on                             %그래프 유지
end                                     %for문 종료
end                                     %함수 닫음

function spline_second(cdate,pop)       %spline_second 함수 선언
figure('Position', [800 200 560 420])   %figure 생성, 위치 지정
plot(cdate,pop,'o')                     %원본 데이터 plot
title("Second Order Spline Fit")         %제목 설정
hold on                                 %그래프 유지
a=zeros((length(cdate)-1)*3);           %(length(cdate)-1)*3 by (length(cdate)-1)*3 size 행렬 선언
b=zeros((length(cdate)-1)*3,1);         %(length(cdate)-1)*3 by 1 size 행렬 선언

%% f(x)값 처리
for i=1:2:(length(cdate)-1)*2-1             %i가 1에서 (length(cdate)-1)*2-1이 될때까지 2씩 더하며 반복
    k=(3*(i-1))/2;                          %k 값 선언
    for j=1:3                               %j가 1에서 3 될때까지 반복
        a(i,k+j)=(cdate((i+1)/2))^(3-j);    %a 행렬의 i행의 k+j열에 cdate((i+1)/2)의 (3-j)제곱을 저장
        a(i+1,k+j)=(cdate((i+3)/2))^(3-j);  %a 행렬의 i+1행의 k+j열에 cdate((i+3)/2)의 (3-j)제곱을 저장
    end                                     %for문 종료
    b(i)=pop((i+1)/2);                      %b 행렬의 i번째 행에 pop((i+1)/2)저장
    b(i+1)=pop((i+3)/2);                    %b 행렬의 i+1번째 행에 pop((i+3)/2) 저장
end                                         %for문 종료

%% f'(x)값 처리 - smoothing
for i=1:length(cdate)-2                     %i가 1에서 legnth(cdate)-2가 될때까지 반복
    k=3*(i-1);                              %k에 3*(i-1)값 저장
    for j=1:3                               %j가 1에서 3이 될때까지 반복
        if j==1                                         %j가 1이라면
            a(i+(length(cdate)-1)*2,k+j)=2*cdate(i+1);      %a(i+(length(cdate)-1)*2,k+j)에 2*cdate(i+1)저장
            a(i+(length(cdate)-1)*2,3+j+k)=-2*cdate(i+1);   %a(i+(length(cdate)-1)*2,3+k+j)에 -2*cdate(i+1)저장
        elseif j==2                                     %j가 2라면
           a(i+(length(cdate)-1)*2,j+k)=1;              %a(i+(length(cdate)-1)*2,k+j)에 1저장
           a(i+(length(cdate)-1)*2,3+j+k)=-1;           %a(i+(length(cdate)-1)*2,3+k+j)에 -1저장
        elseif j==3                                     %j가 3이라면
            a(i+(length(cdate)-1)*2,j+k)=0;             %a(i+(length(cdate)-1)*2,k+j)에 0저장
            a(i+(length(cdate)-1)*2,3+j+k)=0;           %a(i+(length(cdate)-1)*2,3+k+j)에 0저장
        end                                             %if문 종료
    end                                                 %for문 종료
end                                                     %for문 종료

%% f''(x)값 처리
a(end,1)=1;     %a(end,1)=1에 1 저장
c=a\b;          %c에 inv(a)*b 저장

for j=1:3:length(c)                                     %j가 1에서 length(c)가 될때까지 3씩 더해가며 반
    y=zeros(1,length(cdate)-1);                         %y값 저장할 행렬 선언
    x=linspace(cdate((j+2)/3),cdate((j+5)/3),length(cdate)-1);  %x는 구간 cdate((j+2)/3),cdate((j+5)/3)를 length(cdate)-1등분한 행렬
    for k=1:length(cdate)-1                             %k가 1에서 length(cdate)-1이 될때까지 반복
        y(k)=c(j)*(x(k)^2)+c(j+1)*x(k)+c(j+2);          %y(k)에 c(j)*(x(k)^2)+c(j+1)*x(k)+c(j+2)라는 2차식 저장
    end                                                 %for문 종료
    plot(x,y,"LineWidth",1.5)                           %approximation graph plot
    hold on                                             %그래프 유지
end                                                     %for문 종료
end             %함수 닫음
```

알고리즘 설명

```
function spline_first(cdate,pop)
figure('Position', [200 200 560 420])
plot(cdate,pop,'o')
title("First Order Spline Fit")
hold on
temp_y=zeros(1,length(cdate)-1);
m=zeros(1,length(cdate));
for j=1:length(cdate)-1
    m(j)=((pop(j+1)-(pop(j)))/((cdate(j+1))-(cdate(j))));
    x=linspace(cdate(j),cdate(j+1),length(cdate)-1
    for k=1:length(cdate)-1
        temp_y(k)=m(j)*(x(k)-cdate(j))+pop(j);
    end
    plot(x,temp_y,"LineWidth",1.5)
    hold on
end
end
```

First Order Spline Method를 수행하기 위한 함수를 선언하였다. 원본 데이터를 plot하는 부분과, 특정 구간 사이를 plot 하기 위한 변수와, 기울기를 저장할 변수를 선언하였다. 그 후 for문을 이용하여 기울기를 계산하여 m에 저장, 각 구간 사이를 $y = ax + b$ 형태로 이어준다. 그 후 Approximation graph를 plot한다.

```
function spline_second(cdate,pop)
figure('Position', [800 200 560 420])
plot(cdate,pop,'o')
title("Second Order Spline Fit")
hold on
a=zeros((length(cdate)-1)*3);
b=zeros((length(cdate)-1)*3,1
for i=1:2:(length(cdate)-1)*2-1
    k=(3*(i-1))/2;
    for j=1:3
        a(i,k+j)=(cdate((i+1)/2))^(3-j);
        a(i+1,k+j)=(cdate((i+3)/2))^(3-j);
    end
    b(i)=pop((i+1)/2);
    b(i+1)=pop((i+3)/2);
end
```

Second Order Spline Method를 수행하기 위한 함수를 선언하였다. 원본 데이터를 plot하는 부분과, Approximation에 사용할 행렬들을 선언하였다. 그 후 cdate의 특정 인덱스 값을 a행렬의 특정 인덱스에 저장하는 과정을 반복한다. 이 과정은 당연히 Spline Method의 공식을 따른다. 그 후 마지막에는 b에 pop 값을 저장해주는 식으로 for문을 이용하여 반복을 수행한다. f(x)의 처리가 완료되었다.

```
 for i=1:length(cdate)-2
    k=3*(i-1);
    for j=1:3
        if j==1
            a(i+(length(cdate)-1)*2,k+j)=2*cdate(i+1);
            a(i+(length(cdate)-1)*2,3+j+k)=-*cdate(i+1);
        elseif j==2
           a(i+(length(cdate)-1)*2,j+k)=1;
           a(i+(length(cdate)-1)*2,3+j+k)=-1;
        elseif j==3
            a(i+(length(cdate)-1)*2,j+k)=0;
            a(i+(length(cdate)-1)*2,3+j+k)=0;
        end
    end
end
a(end,1)=1;
c=a\b;
```

또 다시 cdate의 특정 인덱스 값을 a행렬의 특정 인덱스에 저장하는 과정을 반복한다. 이 과정은 f′(x)의 값들을 처리하는 과정이다. 그리고 마지막 행에서 f″(x)의 값을 처리해주고, 행렬방정식을 풀어준다.

```
for j=1:3:length(c)
    y=zeros(1,length(cdate)-1
    x=linspace(cdate((j+2)/3),cdate((j+5)/3),length(cdate)-1);
    for k=1:length(cdate)-1
        y(k)=c(j)*(x(k)^2)+c(j+1)*x(k)+c(j+2);
    end
    plot(x,y,"LineWidth",1.5)
    hold on
end
end
```

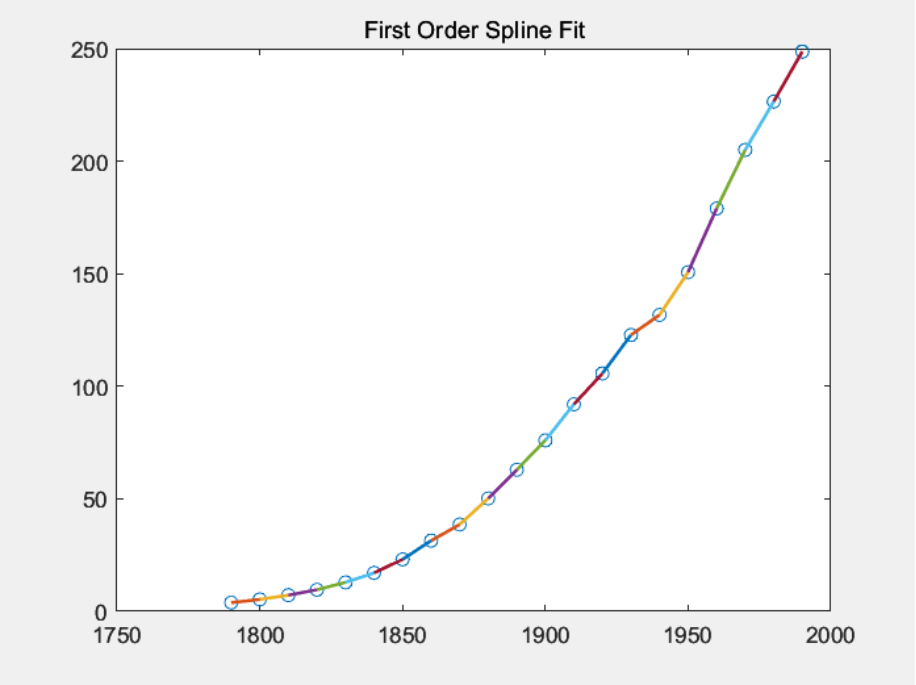
마지막으로 for문 반복을 이용하여 Approximation Graph를 plot해 준다.

```
clear
clc
close all;
load census

spline_first(cdate,pop)
spline_second(cdate,pop)
```

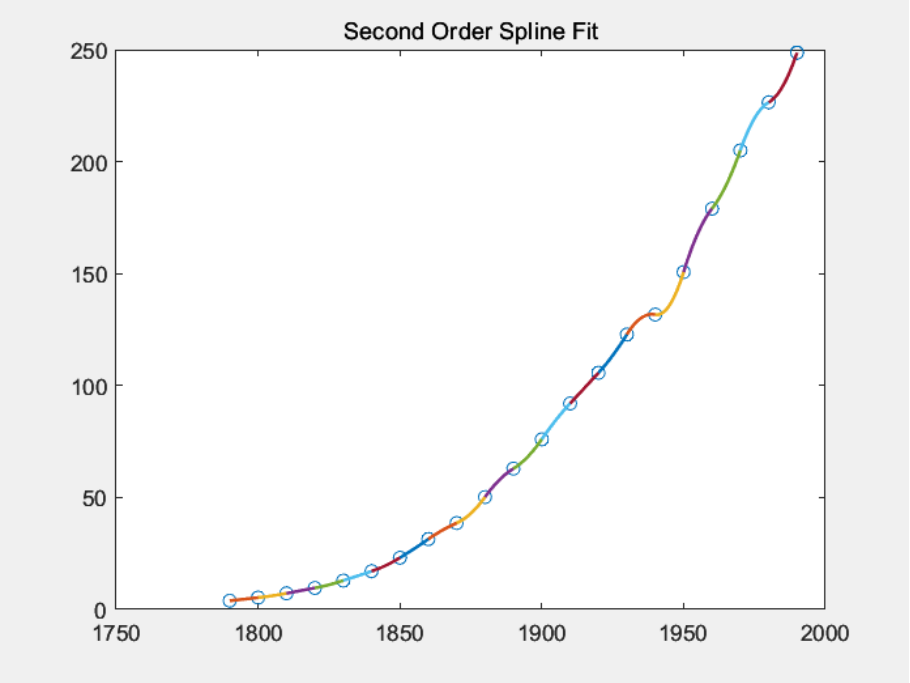
위와 같은 코드로 함수를 호출하여 spline method를 수행하였다.

각각의 plot 결과는 다음과 같았다.

- First Order Spline Method



- Second Order Spline Method



(3) Exponential Fit Method

전체 코드

```
clear       %모든 변수 초기화
clc         %명령 창 초기화
close all;  %모든 figure 닫음
load census %census 불러오기
plot(cdate,pop,'o')          %cdate를 x축, pop을 y축으로 하여 o 모양으로 plot
hold on    %그래프 유지
title("Exponential Fit")    %제목 작성
log_pop=log(pop);    %log를 취한 pop을 log_pop에 저장
exponential(cdate,log_pop)  %exponential 함수에 cdate,log_pop을 입력

function exponential(cdate,pop) %두 가지 입력값을 받는 함수 exponential을 선언
mat1=zeros(2,2);    %2 by 2 size의 matrix 생성
mat2=zeros(2,1);    %2 by 1 size의 matrix 생성
for i=1:2           %i가 1에서 2가 될 때까지 반복
    mat2(i)=sum((cdate.^(2-i)).*pop);   %mat2의 i번째 행에 cdate의 2-i제곱과 pop을 곱한 값의 합을 저장
    for j=1:1:2     %j가 1에서 2가 될 때까지 반복
        mat1(i,j)=sum(cdate.^(4-i-j));  %mat1의 i번째 행 j번째 열에 cdate의 4-i-j 제곱의 합을 저장
    end             %반복문 종료
end                 %반복문 종료
resmat=mat1\mat2;   %inv(mat1)*mat2를 resmat에 저장
resmat(2)=exp(resmat(2));               %resmat(2)를 exp에 대입해 C를 찾아줌
y=resmat(2).*exp(cdate.*resmat(1));     %resmat(2)와 exp(cdate와 resmat(1)을 곱한 값)을 곱한값을 y에 저장
plot(cdate,y,"LineWidth",1.5)           %cdate를 x축, y를 y축으로 하여 plot
end                 %함수 닫음
```

알고리즘 설명

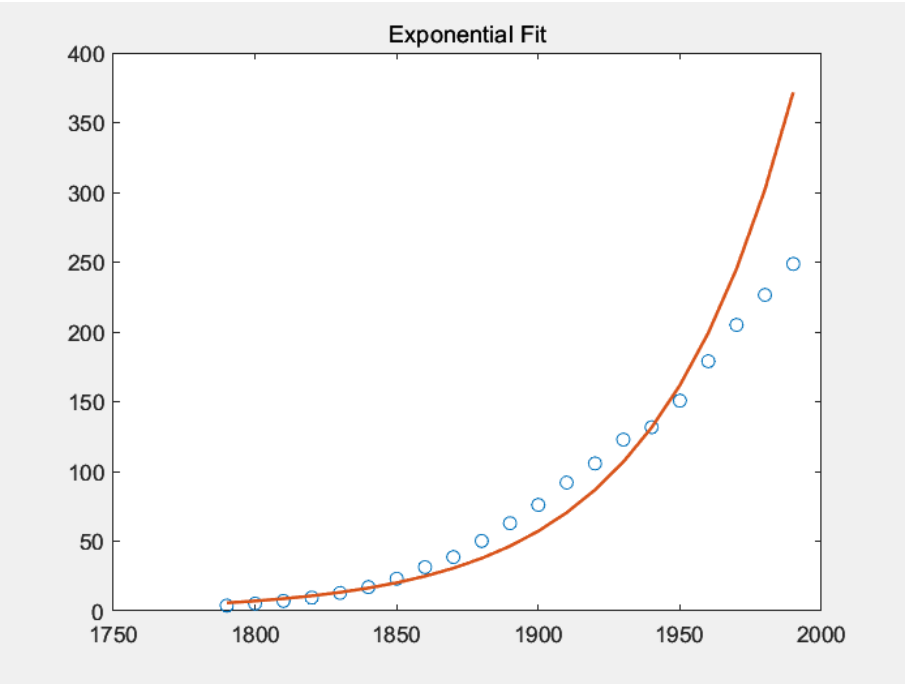
```
function exponential(cdate,pop)
mat1=zeros(2,2);
mat2=zeros(2,1);
for i=1:2
    mat2(i)=sum((cdate.^(2-i)).*pop);
    for j=1:1:2
        mat1(i,j)=sum(cdate.^(4-i-j));
    end
end
resmat=mat1\mat2;
resmat(2)=exp(resmat(2));
y=resmat(2).*exp(cdate.*resmat(1));
plot(cdate,y,"LineWidth",1.5)
end
```

Exponential Fit의 기본적인 알고리즘 동작 방식은 Least Square Method와 전반적으로 동일하다. 다만 행렬 방정식을 이용하여 구한 A와 B값 중, B값은 $e^B = C$를 이용, C값을 구해서 $y = Ce^{Ax}$의 형태로 Approximation을 수행해야 한다. 아래 코드는 함수를 호출하는 코드이다.

```
clear
clc
close all;
load census
plot(cdate,pop,'o')
hold on
title("Exponential Fit")
log_pop=log(pop);
exponential(cdate,log_pop)
```

결과는 다음과 같았다.

- Exponential Fit



(4) Curve-Fitting Function of MATLAB

전체 코드

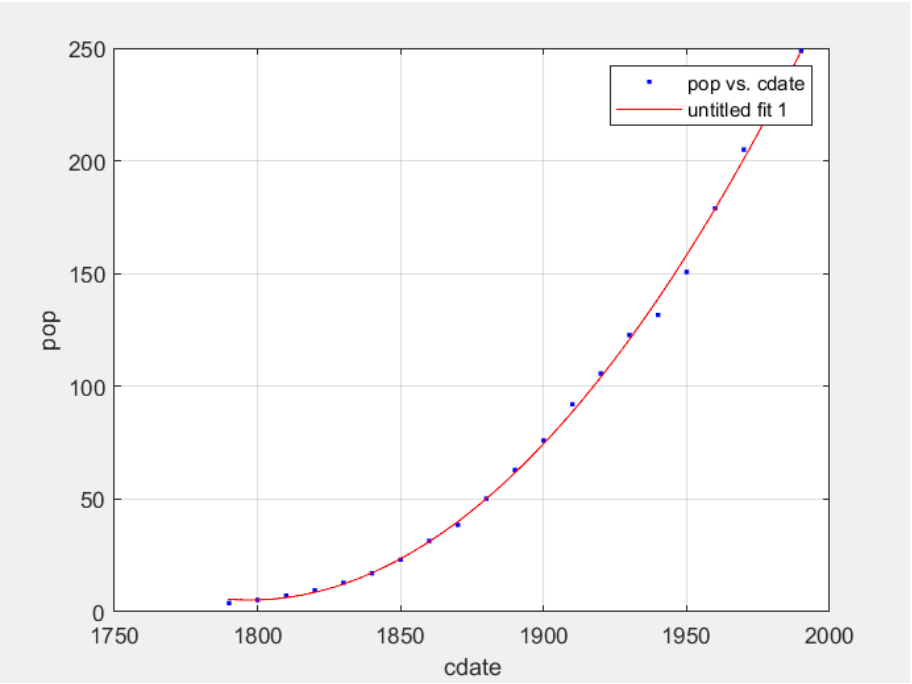
```
clear       %모든 변수 초기화
clc         %명령 창 초기화
load census %census 불러오기
close all;  %모든 figure 닫음
%% curveFitter(cdate,pop) 사용
createFit(cdate,pop) %createFit 함수 cdate,pop을 입력값으로 실행
%Fourier Curve Fitting used

function [fitresult, gof] = createFit(cdate, pop)   %함수 선언
%CREATEFIT(CDATE,POP)
%  피팅을 생성하십시오.
%
%  'untitled fit 1' 피팅의 데이터:
%      X 입력값: cdate
%      Y 출력값: pop
%  출력값:
%      fitresult: 피팅을 나타내는 피팅 객체.
%      gof: 피팅의 적합도 정보를 포함한 구조체.
%
%  참고 항목 FIT, CFIT, SFIT.

%  MATLAB에서 23-Apr-2023 12:21:42에 자동 생성됨


%% 피팅: 'untitled fit 1'.
[xData, yData] = prepareCurveData( cdate, pop );    %피팅용 데이터를 [xData,yData]에 저장

% fittype과 옵션을 설정하십시오.
ft = fittype( 'fourier1' );     %fit type 설정
opts = fitoptions( 'Method', 'NonlinearLeastSquares' ); %옵션 설정
opts.Display = 'Off';   %최적화 출력 메시지 off
opts.StartPoint = [0 0 0 0.0314159265358979];   %최적화 과정 초기값 설정

% 데이터에 모델을 피팅하십시오.
[fitresult, gof] = fit( xData, yData, ft, opts );   %데이터 피팅

% 데이터의 피팅을 플로팅하십시오.
figure( 'Name', 'untitled fit 1' );     %figure 생성
h = plot( fitresult, xData, yData );    %toolbox 사용한 결과를 받아 plot
legend( h, 'pop vs. cdate', 'untitled fit 1', 'Location', 'NorthEast', 'Interpreter', 'none' );
%데이터 종류 표시
% 좌표축에 레이블을 지정하십시오.
xlabel( 'cdate', 'Interpreter', 'none' );   %x label 지정
ylabel( 'pop', 'Interpreter', 'none' );     %y label 지정
grid on     %격자 켬
end     %함수 닫음
```

알고리즘 설명

많은 Curve Fitting Function 중, 푸리에 표현을 이용한 근사 방법을 선택하였다. 이유는 전기전자공학에서 푸리에 급수, 푸리에 변환 등이 널리 쓰이기 때문이었다.

결과는 다음과 같았다.

- Fourier Approximation



Comparison for Problem 1

Least Square Method는 차수가 1차, 2차, 4차로 높아짐에 따라 주어진 원본 Data 값에 더 근사하게 되는 것을 볼 수 있었다. Spline Method는 각 Data의 사이 사이를 이었는데, 1차와 2차간의 Smoothness 차이가 눈에 띄었다. Exponential Method는 잘 들어맞는 부분도 있었지만, 데이터가 커짐에 따라 오차가 커졌다. Curve Fitter를 이용한 푸리에 근사는 전체적으로 Least Square의 2차 이상의 그래프 개형과 유사하게 plot 되었다. 정리하자면 가장 Curve Fitting이 원본 Data의 분포와 유사하게 된 것은 2차 이상의 Least Square였다. Spline Method는 1차는 그래프의 꺾임이, 2차는 너무 과한 곡률이 눈에 띄었고, Exponential은 오차가 컸다.

Problem 2 – Approximation of Datas in Three Dimension

- Least Square Methods

전체 코드

```
clear       %변수 초기화
clc         %명령창 초기화
load census_3d.mat      %census_3d 불러오기
close all;  %모든 figure 닫음

a=[1 2];    %order 수를 저장한 행렬
for i=1:2   %i가 1에서 2가 될 때까지 반복하는 for문
    least_square3D(a(i),800*(i-1)+100,census_3D)
    %least_square3D 함수에 a(i), 800*(i-1)+100, census_3D 입력
end         %for문 종료
```

```
%% 함수

function least_square3D(n,p,census_3D)
%n,p,census_3D를 입력받는 함수를 least_square3D 라는 이름으로 선언
cx1=census_3D(:,1);     %census_3d의 첫번째 열을 cx1에 저장
cx2=census_3D(:,2);     %census_3d의 두번째 열을 cx2에 저장
cy=census_3D(:,3);      %census_3d의 세번째 열을 y에 저장
figure('Position', [p 200 560 420])  %figure 위치와 크기 지정
plot3(cx1,cx2,cy,"o")   %cx1,cx2를 x1,x2축으로, cy를 y축으로 하여 o 모양으로 plot
hold on                 %그래프 유지
titlenum=num2str(n);    %입력 받은 차수를 string 형태로 바꿈
title("Least Square method when order is "+titlenum)    %제목 출력
mat1=zeros(2*n+1,2*n+1);    %mat1을 2n+1 by 2n+1 size로 선언
k=n+1;                      %경계값 k를 n+1로 선언
mat2=zeros(2*n+1,1);        %mat2를 2n+1 by 1 size로 선언
for i=1:2*n+1               %i가 1에서 2n+1이 될때까지 반복하는 for문 선언
    if i<=k                 %i가 k보다 작거나 같을 때
        mat2(i)=sum((cx1.^(i-1)).*cy);  %mat2(i)에 (cx1.^(i-1)).*cy의 합 저장
        for j=1:2*n+1       %j가 1에서 2n+1이 될 때까지 반복하는 for문 선언
            if j<=k         %j가 k보다 작거나 같을 때
                mat1(j,i)=sum(cx1.^(i+j-2));    %mat1(j,i)에 cx1.^(i+j-2)의 합 저장
            else            %j가 k보다 크면
                mat1(j,i)=sum((cx2.^(j-k)).*(cx1.^(i-1))); %mat1(j,i)에 (cx2.^(j-k)).*(cx1.^(i-1))의 합 저장
            end             %if문 종료
        end                 %for문 종료
    else                    %i가 k보다 크면
        mat2(i)=sum((cx2.^(i-k)).*cy);      %mat2(j,i)에 (cx2.^(i-k)).*cy의 합 저장
        for j=1:2*n+1       %j가 1에서 2n+1이 될때까지 반복하는 for문 선언
            if j<=k         %j가 k보다 작거나 같으면
                mat1(j,i)=sum((cx1.^(j-1)).*(cx2.^(i-k)));  %mat1(j,i)에 (cx1.^(j-1)).*(cx2.^(i-k))합 저장
            else            %j가 k보다 크면
                mat1(j,i)=sum(cx2.^(i+j-2*k));  %mat1(j,i)에 cx2.^(i+j-2*k) 합 저장
            end             %if문 종료
        end                 %for문 종료
    end                     %if문 종료
end                         %for문 종료
resmat=mat1\mat2;           %resmat에 inv(mat1)*mat2 값 저장
y=0;                        %변수 y 선언
for i=1:2*n+1               %i가 1에서 2n+1이 될 때까지
    if i<=k                 %i가 k보다 작거나 같으면
        y=y+resmat(i)*(cx1.^(i-1));     %y에 resmat(i)와 cx1.^(i-1)곱한 값 저장
    else                    %i가 k보다 크면
        y=y+resmat(i)*(cx2.^(i-k));     %y에 resmat(i)와 cx2.^(i-k)곱한 값 저장
    end                     %if문 종료
end                         %for문 종료
plot3(cx1,cx2,y,"LineWidth",1.5)        %Approximation 결과 plot
end                         %함수 닫음
```

알고리즘 설명

```
function least_square3D(n,p,census_3D)
cx1=census_3D(:,1);
cx2=census_3D(:,2);
cy=census_3D(:,3);
figure('Position', [p 200 560 420])
plot3(cx1,cx2,cy,"o")
hold on
titlenum=num2str(n);
title("Least Square method when order is "+titlenum)
mat1=zeros(2*n+1,2*n+1);
mat2=zeros(2*n+1,1);
```

census_3D를 입력 받아 x1, x2, y값을 각각 할당해준 뒤, 원본 Data를 plot 해준다. n값을 입력 받아 nth order approximation을 수행할 수 있도록 작성하였다. $y = Ax + B$에서 A에 해당하는 mat1은 2n+1 by 2n+1 size로, B에 해당하는 mat2는 2n+1 by 1 size로 생성하였다.

```
k=n+1;
for i=1:2*n+1
    if i<=k
        mat2(i)=sum((cx1.^(i-1)).*cy);
        for j=1:2*n+1
            if j<=k
                mat1(j,i)=sum(cx1.^(i+j-2));
            else
                mat1(j,i)=sum((cx2.^(j-k)).*(cx1.^(i-1)));
            end
        end
    else
        mat2(i)=sum((cx2.^(i-k)).*cy);
        for j=1:2*n+1
            if j<=k
                mat1(j,i)=sum((cx1.^(j-1)).*(cx2.^(i-k)));
            else
                mat1(j,i)=sum(cx2.^(i+j-2*k));
            end
        end
    end
end
resmat=mat1\mat2;
```

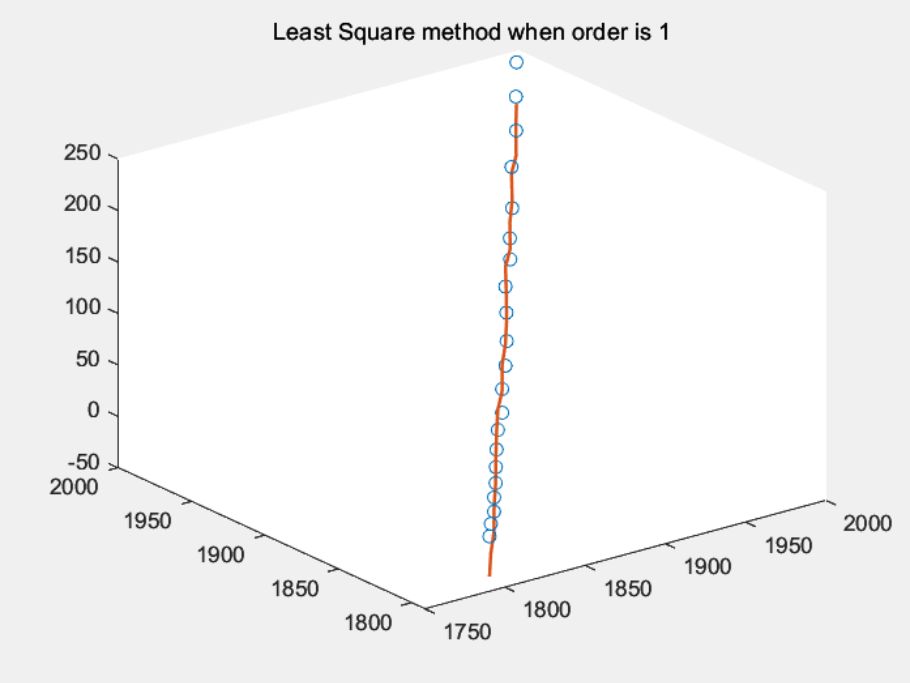
Least Square Method를 수행하기 위해 행렬을 Build하는 코드이다. k라는 임계값을 설정하였는데, mat1과 mat2 모두 k번째 인덱스를 기점으로 연산의 방식이 결정된다. 4개의 for문과 2개의 if문을 거치면 2n+1 by 2n+1 행렬과 2n+1 by 1 행렬이 완성된다. 그 후 resmat에 행렬방정식의 결과값을 저장한다.

```
y=0;
for i=1:2*n+1
    if i<=k
        y=y+resmat(i)*(cx1.^(i-1));
    else
        y=y+resmat(i)*(cx2.^(i-k));
    end
end
plot3(cx1,cx2,y,"LineWidth",1.5)
end
```

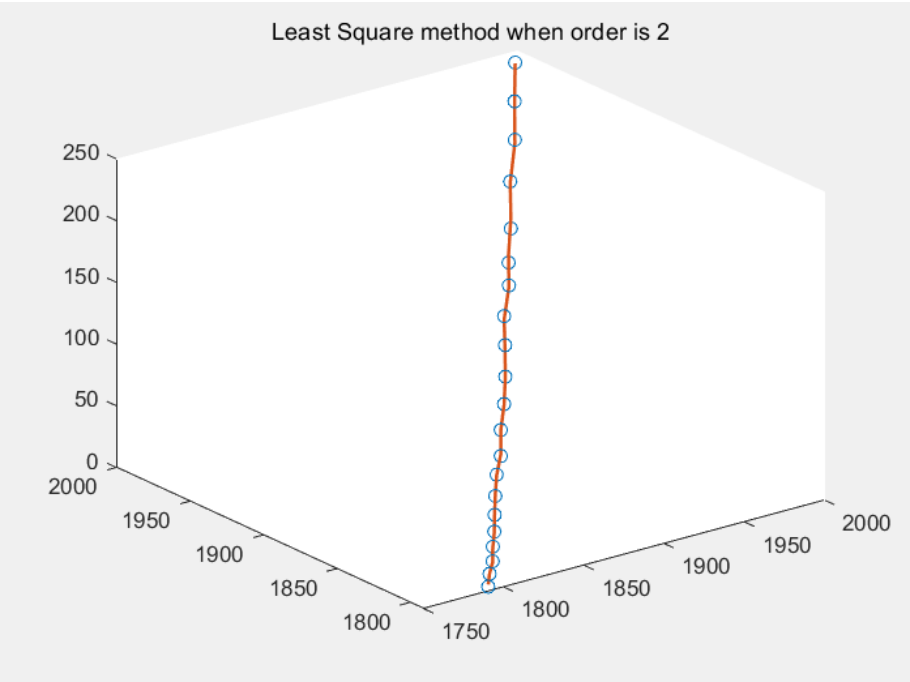
그 후 plot을 위해 변수 y를 선언해주고, 마찬가지로 임계값 k를 이용하여 Least Square Approximation 방정식을 세운다. for문을 통해서 완성된 식을 cx1,cx2,y를 이용하여 plot한다.

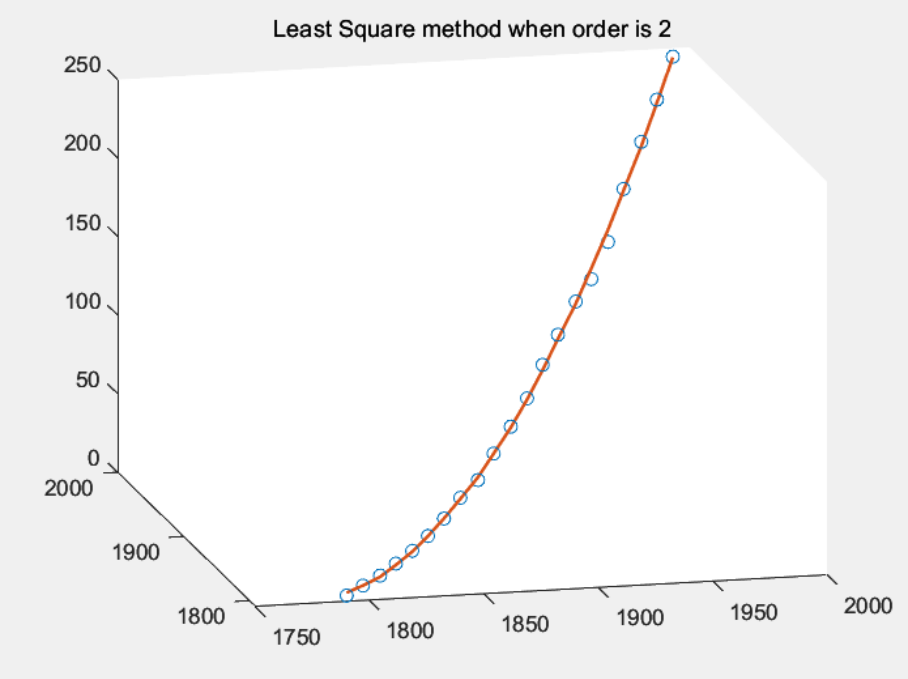
결과는 다음과 같았다.

- First Order Least Square



- Second Order Least Square





Comparison for Problem 2

3차원에서의 First Order Least Square Method와 Second Order Least Square Method를 구현하였다. First Order는 그래프의 꺾임에 더해, 직선의 한계로 인해서 Second Order보다 잘 Fitting 되지는 않았다. n을 입력 받아 Fitting 하는 식으로 코드를 작성했기에 추가로 더욱 높은 차수의 Least Square Method 결과를 plot 해보았으나, Second Order와 큰 차이는 없었다.