MATH 211.3 Winter Term 2024 Assignment

Assignment #06

Name: Kanwar Pannu

Student Number: 11336732

NSID: eza147

**PROBLEM 1**

**A close-up of math equations

Description automatically generated**

**1**

clear;

clc;

% Initial plot setup

figure;

plot([-1 1],[0,0],'k',[0 0],[-1 1],'k'); hold on;

t = 0:.02:1;

% Predefined points and control points

x = [-1; -1; 1; 1];

y = [0; 2; 2; 0];

% Plot initial spline with control points

plot([x(1) x(2)], [y(1) y(2)], 'r:', x(2), y(2), 'rs');

plot([x(3) x(4)], [y(3) y(4)], 'r:', x(3), y(3), 'rs');

plot(x(1), y(1), 'bo', x(4), y(4), 'bo');

% Calculate and plot the initial spline

bx = 3\*(x(2)-x(1)); by = 3\*(y(2)-y(1));

cx = 3\*(x(3)-x(2))-bx; cy = 3\*(y(3)-y(2))-by;

dx = x(4)-x(1)-bx-cx; dy = y(4)-y(1)-by-cy;

xp = x(1) + t.\*(bx + t.\*(cx + t.\*dx));

yp = y(1) + t.\*(by + t.\*(cy + t.\*dy));

plot(xp, yp, 'b');

% Loop for additional user input to extend or modify the curve

while true

[xnew, ynew] = ginput(3); % get three new mouse clicks

if length(xnew) < 3

break; % if fewer than 3 points, exit loop

end

x = [x(4); xnew]; % update points, keeping the last as first

y = [y(4); ynew];

% Plot new control points and line segments

plot([x(1) x(2)], [y(1) y(2)], 'r:', x(2), y(2), 'rs');

plot([x(3) x(4)], [y(3) y(4)], 'r:', x(3), y(3), 'rs');

plot(x(1), y(1), 'bo', x(4), y(4), 'bo');

% Calculate and plot new spline segment

bx = 3\*(x(2)-x(1)); by = 3\*(y(2)-y(1));

cx = 3\*(x(3)-x(2))-bx; cy = 3\*(y(3)-y(2))-by;

dx = x(4)-x(1)-bx-cx; dy = y(4)-y(1)-by-cy;

xp = x(1) + t.\*(bx + t.\*(cx + t.\*dx));

yp = y(1) + t.\*(by + t.\*(cy + t.\*dy));

plot(xp, yp, 'b');

end

hold off;

A screen shot of a graph

Description automatically generated

**PROBLEM 2**

**A close-up of a paper

Description automatically generated**

**A close-up of a paper with writing

Description automatically generated**

**5**

%A

% Data

P = [0.59; 0.80; 0.95; 0.45; 0.79; 0.99; 0.90; 0.65; 0.79; 0.69; 0.79; 0.49; 1.09; 0.95; 0.79; 0.65; 0.45; 0.60; 0.89; 0.79; 0.99; 0.85];

S = [3980; 2200; 1850; 6100; 2100; 1700; 2000; 4200; 2440; 3300; 2300; 6000; 1190; 1960; 2760; 4330; 6960; 4160; 1990; 2860; 1920; 2160];

% Least Squares Fit

A = [ones(size(P)), P]; % Design matrix

coeffs = A\S; % Solving normal equations for c1 and c2

c1 = coeffs(1);

c2 = coeffs(2);

% Plotting

figure;

scatter(P, S, 'filled');

hold on;

P\_fit = linspace(min(P), max(P), 100);

S\_fit = c1 + c2\*P\_fit;

plot(P\_fit, S\_fit, '-r');

xlabel('Price ($)');

ylabel('Sales per Week');

title('Demand Curve: Sales vs. Price');

legend('Actual Sales', 'Least Squares Fit', 'Location', 'Best');

% Calculate RMSE

S\_pred = c1 + c2\*P;

RMSE = sqrt(mean((S - S\_pred).^2));

disp(['Root Mean Square Error: ', num2str(RMSE)]);

% Part (b)

% Define the Profit Function as a function of P

Profit = @(P) (c1 + c2\*P) .\* (P - 0.23);

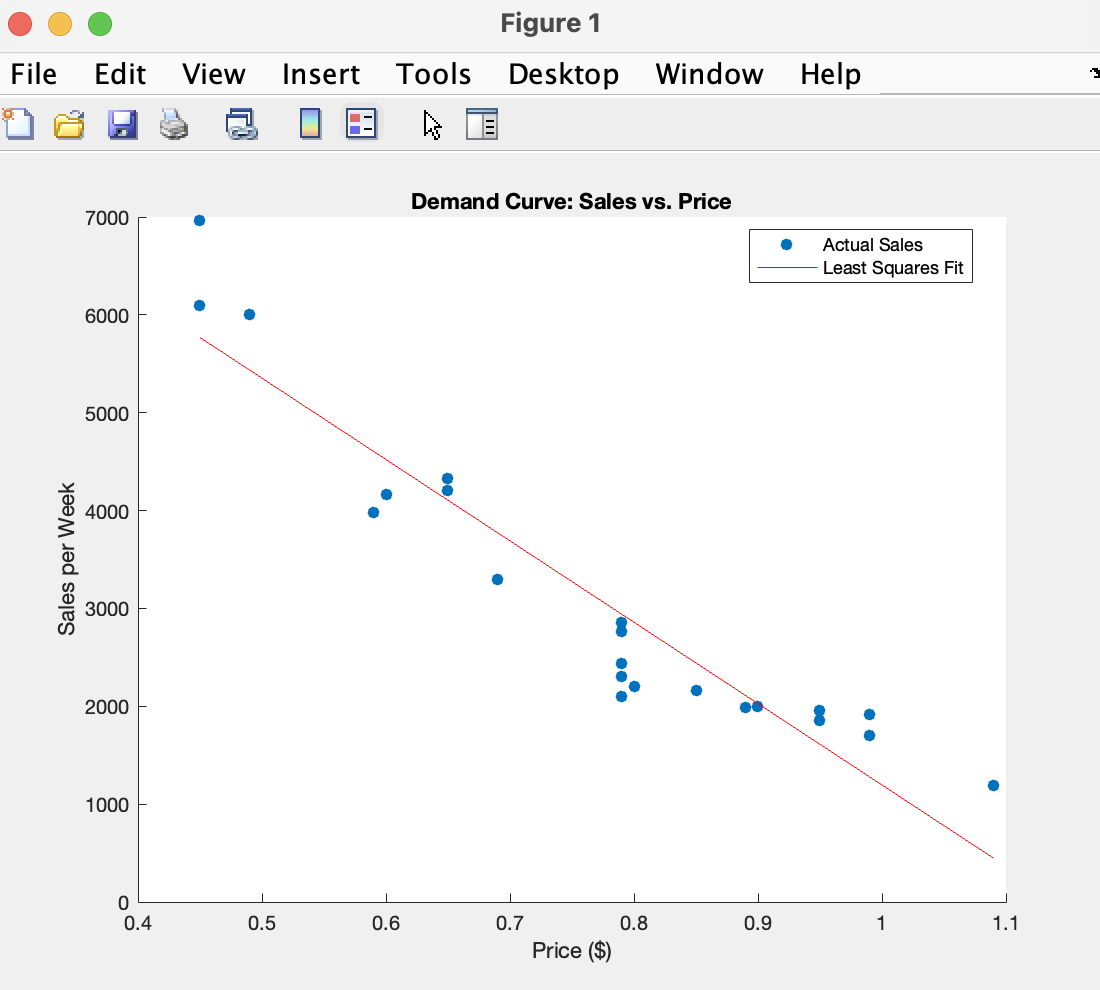
% Find the selling price to maximize profit

[P\_optimal, Profit\_max] = fminbnd(@(P) -Profit(P), 0, 2);

P\_optimal = -P\_optimal;

disp(['Optimal Selling Price: $', num2str(P\_optimal)]);

disp(['Maximum Profit per City per Week: $', num2str(-Profit\_max)]);

****

**PROBLEM 3**

**A sheet of paper with mathematical equations

Description automatically generated**

**A close-up of a paper with writing

Description automatically generated**

**3**

clear;

clc;

% Given data

years = [1960; 1970; 1990; 2000];

populations = [3039585530; 3707475887; 5281653820; 6079603571];

X = years;

Y = log(populations);

p = polyfit(X, Y, 1);

b = p(1);

ln\_a = p(2);

a = exp(ln\_a);

year\_1980 = 1980;

pop\_1980\_est = a \* exp(b \* year\_1980);

% Actual population for 1980

actual\_pop\_1980 = 4452584592;

% Calculate estimation error

error\_1980 = abs(pop\_1980\_est - actual\_pop\_1980);

% Display results

fprintf('The estimated population for 1980 is approximately %.0f.\n', pop\_1980\_est);

fprintf('The actual population for 1980 was %d.\n', actual\_pop\_1980);

fprintf('The estimation error for 1980 is %.0f.\n', error\_1980);

**A screenshot of a computer

Description automatically generated**

**PROBLEM 4**

**A close-up of a paper with mathematical equations

Description automatically generated**

**A close-up of a paper with mathematical equations

Description automatically generated**

**5 a,b**

clear;

clc;

% System (a)

A\_a = [1 1; 2 1; 1 2; 0 3];

b\_a = [3; 5; 5; 5];

[Q\_a, R\_a] = qr(A\_a,0);

x\_a = R\_a \ (Q\_a' \* b\_a);

error\_a = norm(A\_a \* x\_a - b\_a, 2);

disp('System (a):');

disp('Least squares solution:');

disp(x\_a);

disp('2-norm error:');

disp(error\_a);

% System (b)

A\_b = [1 2 2; 2 -1 2; 3 1 1; 1 1 -1];

b\_b = [10; 5; 10; 3];

[Q\_b, R\_b] = qr(A\_b,0);

x\_b = R\_b \ (Q\_b' \* b\_b);

error\_b = norm(A\_b \* x\_b - b\_b, 2);

disp('System (b):');

disp('Least squares solution:');

disp(x\_b);

disp('2-norm error:');

disp(error\_b);

**A screenshot of a computer

Description automatically generated**

**PROBLEM 5**

**A close-up of a paper

Description automatically generated**

**1**

clear;

clc;

f = @(x) sin(x) - cos(x);

f\_prime\_true = @(x) cos(x) + sin(x);

h\_values = 10.^-(1:12)';

errors = zeros(length(h\_values), 1);

for i = 1:length(h\_values)

h = h\_values(i);

f\_prime\_approx = (f(h) - f(-h)) / (2 \* h);

errors(i) = abs(f\_prime\_approx - f\_prime\_true(0));

end

error\_table = table(h\_values, errors)

% Plot the results

loglog(h\_values, errors, '-o');

xlabel('h');

ylabel('Error');

title('Error of Three-Point Centered-Difference Formula');

grid on;

% Analyze the minimum error

[min\_error, min\_idx] = min(errors);

fprintf('The minimum error is %e for h = %e\n', min\_error, h\_values(min\_idx));

**A screenshot of a computer

Description automatically generated**

**A screen shot of a graph

Description automatically generated**

**5**

clear;

clc;

f = @(x) x.^(-1);

f\_double\_prime\_true = @(x) 2.\*x.^(-3);

true\_value = f\_double\_prime\_true(1);

% h values

h\_values = [0.1, 0.01, 0.001];

errors = zeros(size(h\_values));

for i = 1:length(h\_values)

h = h\_values(i);

approximation = (f(1+h) - 2\*f(1) + f(1-h)) / h^2;

errors(i) = abs(approximation - true\_value);

end

for i = 1:length(h\_values)

fprintf('h = %f: Approximation = %f, True Value = %f, Error = %f\n', ...

h\_values(i), (f(1+h\_values(i)) - 2\*f(1) + f(1-h\_values(i))) / h\_values(i)^2, ...

true\_value, errors(i));

end

**A screenshot of a computer

Description automatically generated**