MATH 211.3 Winter Term 2024 Assignment

Assignment #05

Name: Kanwar Pannu

Student Number: 11336732

NSID: eza147

**Problem 1**

A close-up of a paper with mathematical equations

Description automatically generated

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

Description automatically generated**

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

Description automatically generated**

**1**

clear;

clc;

x = [0.6, 0.7, 0.8, 0.9, 1.0];

y = [1.433329, 1.632316, 1.896481, 2.247908, 2.718282];

n = length(x);

% Part(a)

coeffs = newtdd(x, y, n);

% Part(b)

P4\_082 = evalNewtPoly(coeffs, x, 0.82);

P4\_098 = evalNewtPoly(coeffs, x, 0.98);

f = @(x) exp(x.^2);

x\_plot = linspace(0, 2, 1000);

y\_interp = arrayfun(@(x0) evalNewtPoly(coeffs, x, x0), x\_plot);

y\_actual = f(x\_plot);

y\_error = y\_interp - y\_actual;

figure;

subplot(1,2,1); plot(x\_plot, y\_error); title('Error on interval [0, 2]');

xlabel('x'); ylabel('Interpolation Error P(x) - e^{x^2}');

xlim([0.5, 1]);

subplot(1,2,2); plot(x\_plot, y\_error); title('Error on interval [0, 2]');

xlabel('x'); ylabel('Interpolation Error P(x) - e^{x^2}');

xlim([0, 2]);

function coeffs = newtdd(x, y, n)

v = zeros(n, n);

v(:,1) = y';

for i = 2:n

for j = 1:n+1-i

v(j,i) = (v(j+1,i-1) - v(j,i-1)) / (x(j+i-1) - x(j));

end

end

coeffs = v(1, :);

end

function val = evalNewtPoly(coeffs, x, x0)

n = length(coeffs);

val = coeffs(n);

for i = n-1:-1:1

val = val \* (x0 - x(i)) + coeffs(i);

end

end

**3**

clear;

clc;

years = 1994:2003;

bbl\_day = [67.052, 68.008, 69.803, 72.024, 73.400, 72.063, 74.669, 74.487, 74.065, 76.777];

% Convert years to a more manageable range for numerical stability

x = years - 1993;

y = bbl\_day;

% Fit a degree 9 polynomial to the data

p = polyfit(x, y, 9);

%Points for the curve

x\_fit = linspace(min(x), max(x)+16, 1000);

y\_fit = polyval(p, x\_fit);

figure;

plot(x, y, 'o', 'MarkerFaceColor', 'b');

plot(x\_fit, y\_fit, '-r');

xlabel('Year');

ylabel('Oil Production (Millions of barrels per day)');

title('Oil Production and Degree 9 Polynomial Fit');

legend('Original Data', 'Degree 9 Fit', 'Location', 'NorthWest');

year\_2010 = 2010 - 1993; % Adjusted year for 2010

prod\_2010 = polyval(p, year\_2010);

fprintf('Estimated oil production for 2010: %.3f million barrels per day\n', prod\_2010);

**Problem 2**

**A close-up of a piece of paper

Description automatically generated**

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

Description automatically generated**

**4**

clear;

clc;

for n = [10, 20]

x\_even = linspace(-1, 1, n+1);

y\_even = exp(abs(x\_even));

% Chebyshev points

x\_cheb = chebyshevNodes(n, -1, 1);

y\_cheb = exp(abs(x\_cheb));

% Calculate coefficients without adjusting n inside newtdd

c\_even = newtdd(x\_even, y\_even, n+1);

c\_cheb = newtdd(x\_cheb, y\_cheb, n+1);

% Plotting and error

x\_plot = -1:0.01:1;

y\_actual = exp(abs(x\_plot));

y\_even\_poly = arrayfun(@(x) nest(n, c\_even, x, x\_even), x\_plot);

y\_cheb\_poly = arrayfun(@(x) nest(n, c\_cheb, x, x\_cheb), x\_plot);

figure;

subplot(2,1,1);

plot(x\_plot, y\_actual, 'k-', x\_plot, y\_even\_poly, 'r--', x\_plot, y\_cheb\_poly, 'b-.');

title(sprintf('Interpolation with n = %d', n));

legend('Actual', 'Evenly Spaced', 'Chebyshev', 'Location', 'best');

%Empirical interpolation errors

error\_even = abs(y\_actual - y\_even\_poly);

error\_cheb = abs(y\_actual - y\_cheb\_poly);

subplot(2,1,2);

plot(x\_plot, error\_even, 'r--', x\_plot, error\_cheb, 'b-.');

title(sprintf('Interpolation Error with n = %d', n));

legend('Error Evenly Spaced', 'Error Chebyshev', 'Location', 'best');

end

function coeffs = newtdd(x, y, n)

% Adjusted newtdd function, assuming n is now correctly the number of points

v = zeros(n, n);

v(:,1) = y(:); % Ensure y is a column vector

for i = 2:n

for j = 1:n-i

v(j,i) = (v(j+1,i-1) - v(j,i-1)) / (x(j+i-1) - x(j));

end

end

coeffs = v(1, :);

end

% Function to generate Chebyshev base points

function nodes = chebyshevNodes(n, a, b)

k = 0:n;

nodes = cos((2\*k+1)\*pi/(2\*n+2));

nodes = 0.5\*(b-a)\*nodes + 0.5\*(b+a);

end

function y = nest(d, c, x, b)

% Nested multiplication

y = c(d+1);

for i = d:-1:1

y = y\*(x - b(i)) + c(i);

end

end

**Problem 3**

**A close-up of a paper

Description automatically generated**

**A close-up of math equations

Description automatically generated**

1b

clear;

clc;

% data points for part (b)

x\_b = [-1, 0, 3, 4, 5];

y\_b = [3, 5, 1, 1, 1];

% Calculate coefficients

coeff\_b = splinecoeff(x\_b, y\_b);

%part (b) using the splineplot function

figure;

splineplot(x\_b, y\_b, coeff\_b);

title('Natural Cubic Spline for Part (b)');

xlabel('x');

ylabel('y');

function splineplot(x, y, coeff)

n = length(x); % Number of data points

k = 100; % Number of points to plot per segment for smooth curves

hold on; % Hold the plot for multiple plots

for i = 1:n-1

xs = linspace(x(i), x(i+1), k);

dx = xs - x(i);

% Assume coeff structure is [b,c,d] per segment, and a\_i = y\_i

ys = y(i) + coeff(i,1)\*dx + coeff(i,2)\*dx.^2 + coeff(i,3)\*dx.^3;

plot(xs, ys, 'r-', 'LineWidth', 1.5); % Plot spline segment

end

plot(x, y, 'bo', 'MarkerFaceColor', 'b'); % Plot original data points

hold off; % Release plot hold

xlabel('x');

ylabel('y');

title('Natural Cubic Spline Interpolation');

end

%Program 3.5 Calculation of spline coefficients

%Calculates coefficients of cubic spline

%Input: x,y vectors of data points

% plus two optional extra data v1, vn

%Output: matrix of coefficients b1,c1,d1;b2,c2,d2;...

function coeff=splinecoeff(x,y)

n=length(x);v1=0;vn=0;

A=zeros(n,n); % matrix A is nxn

r=zeros(n,1);

for i=1:n-1 % define the deltas

dx(i)= x(i+1)-x(i); dy(i)=y(i+1)-y(i);

end

for i=2:n-1 % load the A matrix

A(i,i-1:i+1)=[dx(i-1) 2\*(dx(i-1)+dx(i)) dx(i)];

r(i)=3\*(dy(i)/dx(i)-dy(i-1)/dx(i-1)); % right-hand side

end

% Set endpoint conditions

% Use only one of following 5 pairs:

A(1,1) = 1; % natural spline conditions

A(n,n) = 1;

%A(1,1)=2;r(1)=v1; % curvature-adj conditions

%A(n,n)=2;r(n)=vn;

%A(1,1:2)=[2\*dx(1) dx(1)];r(1)=3\*(dy(1)/dx(1)-v1); %clamped

%A(n,n-1:n)=[dx(n-1) 2\*dx(n-1)];r(n)=3\*(vn-dy(n-1)/dx(n-1));

%A(1,1:2)=[1 -1]; % parabol-term conditions, for n>=3

%A(n,n-1:n)=[1 -1];

%A(1,1:3)=[dx(2) -(dx(1)+dx(2)) dx(1)]; % not-a-knot, for n>=4

%A(n,n-2:n)=[dx(n-1) -(dx(n-2)+dx(n-1)) dx(n-2)];

coeff=zeros(n,3);

coeff(:,2)=A\r; % solve for c coefficients

for i=1:n-1 % solve for b and d

coeff(i,3)=(coeff(i+1,2)-coeff(i,2))/(3\*dx(i));

coeff(i,1)=dy(i)/dx(i)-dx(i)\*(2\*coeff(i,2)+coeff(i+1,2))/3;

end

coeff=coeff(1:n-1,1:3);

end

**2b**

clear;

clc;

% Data points for part (b)

x\_b = [-1, 0, 3, 4, 5];

y\_b = [3, 5, 1, 1, 1];

coeff\_b = splinecoeff(x\_b, y\_b);

% Plot

splineplot(x\_b, y\_b, coeff\_b);

function coeff = splinecoeff(x, y)

n = length(x); % Number of data points

dx = diff(x); % Differences in x

dy = diff(y) ./ dx; % Differences in y divided by differences in x

% Initialize the matrix A and the vector r

A = zeros(n, n);

r = zeros(n, 1);

% Set interior rows of A and r (tridiagonal part of A)

for i = 2:n-1

A(i, i-1:i+1) = [dx(i-1), 2\*(dx(i-1) + dx(i)), dx(i)];

r(i) = 3\*(dy(i) - dy(i-1));

end

% Not-a-knot conditions

A(1, 1:3) = [dx(2), -(dx(1) + dx(2)), dx(1)];

A(n, n-2:n) = [dx(n-2), -(dx(n-2) + dx(n-1)), dx(n-1)];

% Solve the system A\*c = r for c

c = A\r;

% Calculate b and d

b = dy - dx.\*(2\*c(1:end-1) + c(2:end))/3;

d = (c(2:end) - c(1:end-1)) ./ (3\*dx);

% Output coefficients

coeff = zeros(n-1, 3); % Each row will have [b, c, d] for each interval

for i = 1:n-1

coeff(i, :) = [b(i), c(i), d(i)];

end

end

function splineplot(x, y, coeff)

n = length(x); % Number of data points

xx = linspace(min(x), max(x), 1000); % Fine grid for plotting

yy = zeros(size(xx));

for i = 1:n-1

% Indices for the current interval in xx

idx = xx >= x(i) & xx <= x(i+1);

% Calculate spline value for this interval

dx = xx(idx) - x(i);

yy(idx) = y(i) + coeff(i,1)\*dx + coeff(i,2)\*dx.^2 + coeff(i,3)\*dx.^3;

end

% Plot the spline

plot(x, y, 'bo', 'MarkerFaceColor', 'b'); hold on; % Original data points

plot(xx, yy, 'r-', 'LineWidth', 1.5); % Spline

hold off;

title('Not-a-Knot Cubic Spline Interpolation');

xlabel('x');

ylabel('y');

legend('Data Points', 'Spline', 'Location', 'Best');

end