Q2

14 / 14

(Programming by MATLAB) The US census data from 1900 to 2010 are as follows (numbers are in million):

x	1900	1910	1920	1930	1940	1950	1960
y	75.995	91.972	105.711	123.203	131.669	150.697	179.323
	1970	1980	1990	2000	2010		
	203.212	226.505	249.633	281.422	308.746		

(a) (8 points) Spline interpolation

Find the natural cubic spline function S to interpolate the data. Then use the spline function to estimate the population for the years 1975 and 2020.

You do not need to write an explicit expression for $S(x)$. Just plot $y = S(x)$, the given 12 data points, and the 2 population estimate points on the same graph. Print your MATLAB codes.

Note: For reference, see `us-census.m` on the course website: https://www.cs.mcgill.ca/~chang/teaching/cs350/doc.php

(b) (6 points) LS approximation

Suppose we use the model $y = c_1 e^{c_2 x}$ to represent the population in the year x. The challenge is that y is a nonlinear function of c_1 and c_2, making it difficult to apply the linear LS method directly to estimate c_1 and c_2. However, we can use a simple transformation to address this issue. Taking the natural logarithm of both sides, we have $\log_e y = \log_e c_1 + c_2 x$. Let $\bar{y} = \log_e y$ and $\bar{c}_1 = \log_e c_1$. Then $\bar{y} = \bar{c}_1 + c_2 x$. This linear form allows us to fit the transformed data points $(x_i, \bar{y}_i)$ (where $\bar{y}_i = \log_e y_i$) by the LS method, yielding estimates for $\bar{c}_1$ and c_2. Then we can find the estimate of c_1 using $c_1 = e^{\bar{c}_1}$.

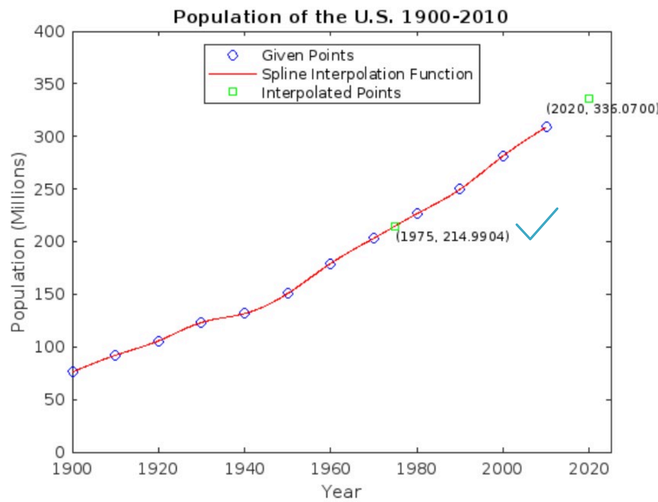
Use the method described above to estimate the values of c_1 and c_2. Then use the function $y = c_1 e^{c_2 x}$ to estimate the population for the years 1975 and 2020. Plot the function $y = c_1 e^{c_2 x}$, the given 12 data points, and the 2 population estimate points on the same graph. Print your MATLAB codes.

Note: You are NOT allowed to use MATLAB built-in functions `polyfit`, `polyval`, and `spline`.

```
% ASSIGNMENT 5 QUESTION 2(a)
% Given data from 1900 to 2010, 12 data points (n=11).
t = 1900:10:2010;
y = [75.995 91.972 105.711 123.203 131.669 150.697 179.323 203.212 226.505
249.633 281.422 308.746];

p1 = plot(t, y, 'bo');
hold on;
axis([1900 2025 0 400]);
title('Population of the U.S. 1900-2010');
xlabel('Year');
ylabel('Population (Millions)');


% Natural cubic spline is piecewise S(x) such that:
% For i = 1:n-1, Si(x) = ai + bi(x-ti) + ci(x-ti)^2 + di(x-ti)^3
n = length(t)-1;

% Compute the coefficients of the natural cubic spline (4 n-vectors)
[a, b, c, d] = natural_cubic_spline(t, y);


% Plot spline interpolation for every year in the interval 1900 - 2010
x_spline = 1900:0.1:2010;
y_spline = zeros(1, length(x_spline));

start_index = 1;

% For each of the intervals, keep track of which piece of S(x) we're using
% Note: we are taking advantage of the fact that I know what interval each
% x-value is in by the loop index so I am not using another loop to search and
% re-compute the interval that each point is found in for efficiency.

for i = 1:n
    % Find expression for S_i(x)
    % Note: using nested, element-wise multiplication for more efficient
computation
    S = @(x) a(i) + (x-t(i)) .* (b(i) + (x-t(i)) .* (c(i) + (x-t(i)) .*
d(i)));

    % Incrementing by 100 since each interval plots 100 interpolated points
    y_spline(start_index:start_index+100) =
S(x_spline(start_index:start_index+100));
    start_index = start_index + 100;
end

p2 = plot(x_spline, y_spline, 'r-');

% Now to plot the estimated population values for 1975 and 2020
% Again, we know that for 1975, we will use S_7(x) and for 2020 we will use
S_10(x)
% But, for the sake of generalization of the algorithm, we will use a loop
```

1

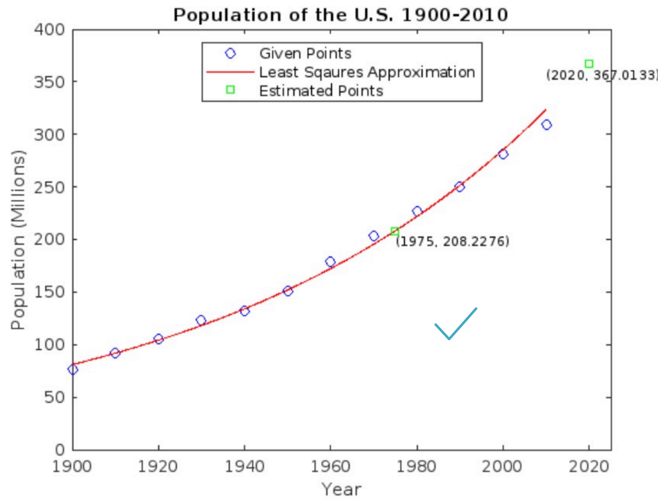
```
% to compute the interval these points are found in and evaluate S(x) that way

x_1 = 1975;
for i = 1:n
    if x_1 - t(i+1)< 0
        break
    end
end
y_1 = a(i)+(x_1-t(i))*(b(i)+(x_1-t(i))*(c(i)+(x_1-t(i))*d(i)));

p3 = plot(x_1,y_1,'gs');
p1_label = sprintf("(%d, %3.4f)", x_1, y_1);
text(1975, y_1-10, p1_label, "FontSize", 8);

x_2 = 2020;
for i = 1:n
    if x_2 - t(i+1)< 0
        break
    end
end
y_2 = a(i)+(x_2-t(i))*(b(i)+(x_2-t(i))*(c(i)+(x_2-t(i))*d(i)));

plot(x_2,y_2,'gs');
p2_label = sprintf("(%d, %3.4f)", x_2, y_2);
text(2020-10, y_2-10, p2_label, "FontSize", 8);
legend([p1, p2, p3], 'Given Points', 'Spline Interpolation Function',
'Interpolated Points', 'Location', 'best');
```



Published with MATLAB® R2024b

```
% ASSIGNMENT 5 QUESTION 2(b)
% Given data from 1900 to 2010, 12 data points
x = 1900:10:2010;
y = [75.995 91.972 105.711 123.203 131.669 150.697 179.323 203.212 226.505
249.633 281.422 308.746];

% Using m+1 for number of given points
m = length(x)-1;

% Plot given points
p1 = plot(x, y, 'bo');
hold on;
axis([1900 2025 0 400]);
title('Population of the U.S. 1900-2010');
xlabel('Year');
ylabel('Population (Millions)');

% Enter basis functions
% Note: g_1 is trivial and won't actually be called to evaluate
% since we know the result it will give
g_1 = @(x) 1;
g_2 = @(x) x;

% Construct the matrix A, trivially g_1(x) = 1 for all x
A = [ones(m+1,1), g_2(x)'];

% Transform the RHS vector y to match the LHS
log_y = log(y);

% Solve for c using MATLAB built-in command
c = A\log_y';

% Transform c1 back to original form and assign c2
c1 = exp(c(1));
c2 = c(2);

% Plot actual given function y
fun_y = @(x) c1.*exp(c2.*x);
x_ls = 1900:0.1:2010;
y_ls = fun_y(x_ls);
p2 = plot(x_ls, y_ls, 'r-');

% Estimate population for 1975 and 2020
est_1975 = fun_y(1975);
p3 = plot(1975,est_1975,'gs');
label = sprintf("(1975, %3.4f)", est_1975);
text(1975, est_1975-10, label, "FontSize", 8);

est_2020 = fun_y(2020);
plot(2020,est_2020,'gs');
label = sprintf("(2020, %3.4f)", est_2020);
text(2010, est_2020-10, label, "FontSize", 8);
```

```
legend([p1, p2, p3], 'Given Points', 'Least Sqaures Approximation',
'Estimated Points', 'Location', 'best');
hold off;
```



Published with MATLAB® R2024b