MATH 3620: Computer Homework 06

Student: Peter Koncelik

Signature:

Contents

Matlab output

Reduce space between output

```
format compact
```

Question 1:

Q1 a)

```
% saved gauss.txt to working folder
```

Q1 b)

```
% loads data from gauss.txt
Data = load('gauss.txt');
```

Q1 c)

```
% stores first and second data columns as 'tdat' and 'ydat' respectively
tdat = Data(:,1);
ydat = Data(:,2);
```

Question 2:

Q2 a)

```
% determines the number of elements 'n'
n = numel(tdat);

fprintf('Number of data elements n = %d\n', n);
```

```
Number of data elements n = 21
```

Q2 b)

```
% Fix the period T: note that tDelt comes out to 0.5 no matter which values
% are chosen (see gauss.txt data spacing)
tDelt = tdat(2)-tdat(1);
T = tdat(n) - tdat(1) + tDelt;

fprintf('Fixed Period T = %.2f\n', T);
```

```
Fixed Period T = 10.50
```

Question 3

Q3)

```
% Calculates the Discrete Fourier Transform of 'y'
yhat = fft(ydat);
```

Question 4

Q4)

```
nhat = floor(n/2);

fprintf('The value of nhat = %d\n', nhat);

% The use of nhat as the floor of the number of elements 'n' over 2 is due
% to the symmetry of the DFT. In particular, we proved in class that:
% yhat(n-k) = yhat(k) complex conjugate (or. yhat(k) bar) for k = 1,...,n-1.
% This symmetry allows the above calculation, and reduces the ammount of
% work done in computing/extracting the trigonometric polynomial and its
% values below
%
% Additionally the floor function ensures that the value being used in
% evaluation/interpolation is an integer as needed (since we can't sum from
% j = 1 to n/2 = 10.5)
```

```
The value of nhat = 10
```

Question 5

Q5)

```
a0 = yhat(1) / n;

a = zeros(n, 1);
b = zeros(n, 1);

% Extracting trigonometric polynomial coefficients from DFT
for k = 1:nhat
    a(k) = (2*real(yhat(k+1))) / n;
    b(k) = -(2*imag(yhat(k+1))) / n;
end
```

Question 6

Q6)

```
% Fine grid of points on interval [0,10]
tlo = 0;
thi = 10;
tplt = linspace(tlo, thi, 801);
```

Question 7

Q7)

```
% Evaluating the trigonometric polynomial at the fine grid of points
yplt = a0 * ones(size(tplt));
for k = 1:n
   tmp = (2*pi*k*tplt)/T;
```

```
    yplt = yplt + a(k)*cos(tmp) + b(k)*sin(tmp);
end
```

Question 8

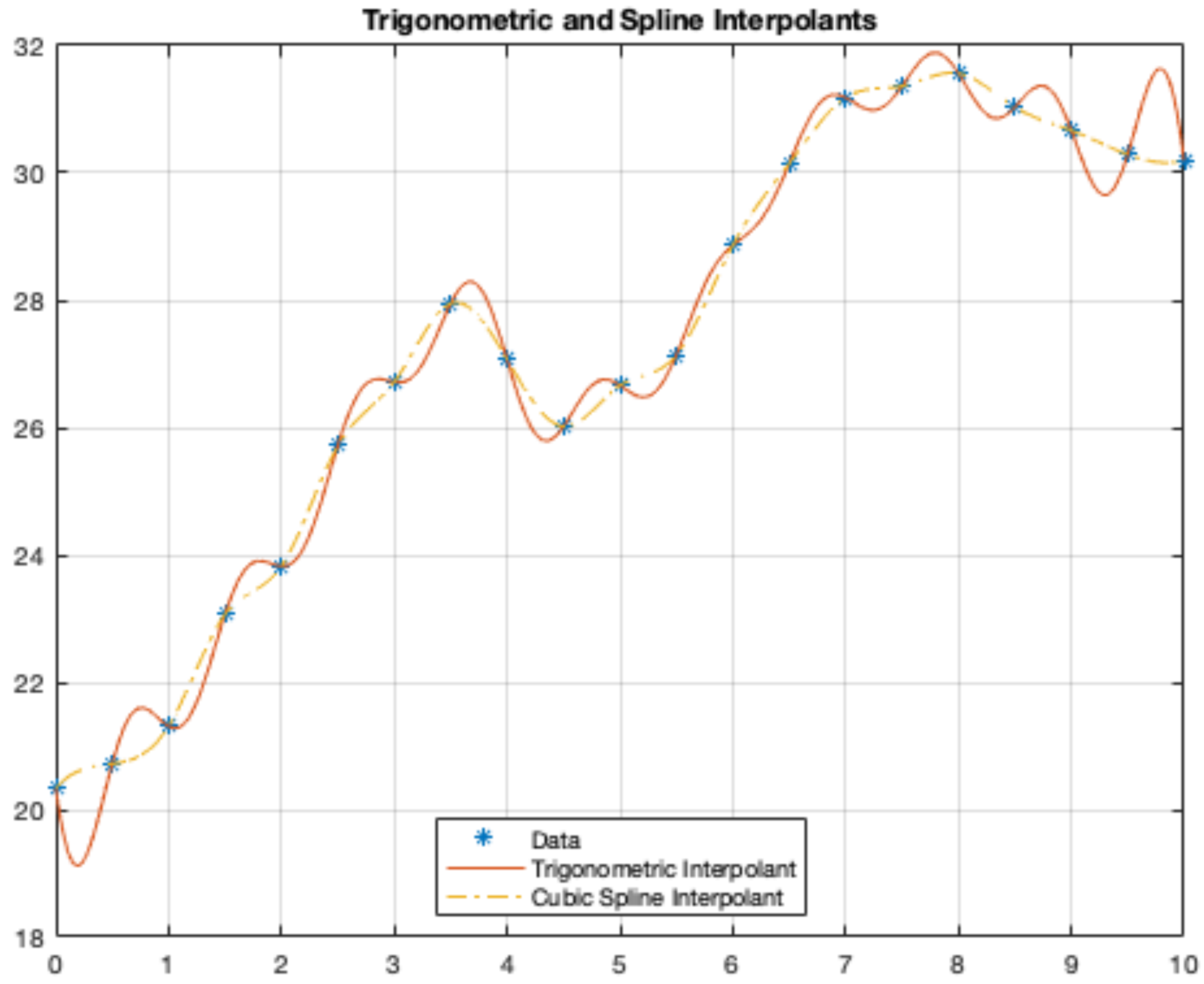
Q8)

```
% Evaluating the cubic spline interpolant, default not-a-knot end
% conditions.
s1 = spline(tdat, ydat, tplt);
```

Question 9

Q9)

```
% Plotting data, trig interpolant, and spline interpolant on [0,10]
figure(1)
p = plot(tdat, ydat, '*', tplt, yplt, tplt, s1, '-.');
grid on
title('Trigonometric and Spline Interpolants');
legend('Data', 'Trigonometric Interpolant', 'Cubic Spline Interpolant', 'Location', 'South');
p(2).LineWidth = 1;
p(3).LineWidth = 1;
```



Question 10

Q10 a)

```
% Repeating the above plot (ie. Questions 7,8,9) on an interval of
% [-0.5, 10.5] rather than [0,10]

% Q6) again

tlo_2 = -0.5;
thi_2 = 10.5;
tplt_2 = linspace(tlo_2, thi_2, 801);

% Q7) again

% Evaluating the trigonometric polynomial at the fine grid of points on new
% interval [-0.5, 10.5]
yplt_2 = a0 * ones(size(tplt_2));
for k = 1:n
   tmp = (2*pi*k*tplt_2)/T;
   yplt_2 = yplt_2 + a(k)*cos(tmp) + b(k)*sin(tmp);
end

% Q8) again

% cubic spline interpolation on new interval [-0.5, 10.5]
s2 = spline(tdat, ydat, tplt_2);

% Q9) again

% plotting data, trig interpolant, spline interpolant on [-0.5, 10.5]
figure(2)
p2 = plot(tdat, ydat, '*', tplt_2, yplt_2, tplt_2, s2, '-.');
grid on
title('Trigonometric and Spline Interpolants beyond interval of data');
legend('Data', 'Trigonometric Interpolant', 'Cubic Spline Interpolant', 'Location', 'South');
p2(2).LineWidth = 1;
p2(3).LineWidth = 1;
```

Q10 b)

```
% The behavior of both the spline and trigonometric interpolants sees
% massive seperation from the data in the positive and negative directions.
% Specifically, as the input values (ie. x-axis) move in the negative
% direction, the trig interpolant shoots up in the positive direction
% rapidly, while the spline interpolant shoots down in the negative
% direction. Conversely, as  input values move in the positive direction,
% the trig interpolant shoots down rapidly in the negative
% direction, and the spline interpolant shoots up in the positive
% direction.
%
% (while trig behavior in both cases appears infinite, upon
% further inspection it is more oscillatory (see below)).
%
% This wild/random behavior can be attributed to the
% interpolants attempting to plot/interpolate outside of the given data set
% of [0,10]. For this reason, both the trig and spline interpolations can
```

```
% not be treated as an accurate approximation outside of the bounds of
% [0,10].

% Note: that upon expanding the interval even further (ie. to [-5,15]), the
% spline interpolant moves to negative and positive infinity, respectively,
% while the aforementioned rapid increase/decrease of the trig interpolant
% is actually an apparent rapid increase, followed by a general "flattening
% out" of the function as it continues to negative and
% positive infinity. Upon inspection, it appears that the function
% oscillates infinitely in either direction around function values of
% 20-30 (This is just based on my surface level visualization and no
% computation.
```