Rutgers CS323 (04), Spring 2017, Homework 4

Due at 11:55pm on March 31, 2017, submitted via Sakai

Cubic Spline Interpolation

You are asked to implement the following four functions:

```
1. function B = NCS(x,y)
2. function B = NNCS(x,y)
3. function TestNCS(n)
4. function TestNNCS(n)
```

for implementing/testing natural cubic splines and "not-a-knot" cubic splines.

1 Natural Cubic Splines

The TestNCS.m code is almost entirely written for you. You will just need to figure out the coefficients for plotting the two boundary segments. You don't need to submit the figures this time however your output should be identical to the figures as provided. We will test your code for any n values.

As shown in lecture notes pages 79-81, there are in total $(n-1)$ cubic curves. Each curve needs 4 coefficients, for example $ax^3 + bx^2 + cx + d$. In the code, B stores the vector of $(n-1)4$ coefficients $B = [a\ b\ c\ d\ ...]^{\mathrm{T}}$. You will need $(n-1)4$ equations and organize them in a matrix form: $AB = Z$ and solve for $B = A^{-1}Z$. You are suggested to follow the similar order as in the lecture note example to organize A and Z. But ultimately, as long as you obtain the correct B vector, you can organize A and Z any way you like. Note that you will need to figure out the two linear equations at the two boundaries, when you plot the curves.

```
function TestNCS(n)

dx = 0.001; maxX = 5; minX = -5;
X = minX:dx:maxX;
Y = 1./(1+X.^2);
t = round(length(X)/(n+1));
ind = t:t:n*t;

x = X(ind); y = Y(ind);
B = NCS(x,y);
figure;
plot(X,Y,'--','linewidth',2);hold on; grid on;
```

```
plot(x,y,'go','linewidth',3);
for i = 1:n-1
    xx = x(i):dx:x(i+1);
    b = B((i-1)*4+1:i*4);
    yy = 0;
    for j = 1:4
        yy = b(j)*xx.^(4-j)+yy;
    end
    plot(xx,yy,'r-','linewidth',2);
end

xx = min(X):dx:x(1);
A = %% you need to figure out the expression for A
yy = A.*(xx-x(1))+y(1);
plot(xx,yy,'r-','linewidth',2);
set(gca,'fontsize',20);xlabel('x');ylabel('y');
set(gca,'xtick',min(X):1:max(X));
xx = x(end):dx:maxX;
A = %% you need to figure out the expression for A
yy = A.*(xx-x(end))+y(end);
plot(xx,yy,'r-','linewidth',2);
title(['NCS: y = 1/(1+x^2) : n = ' num2str(n)]);
```

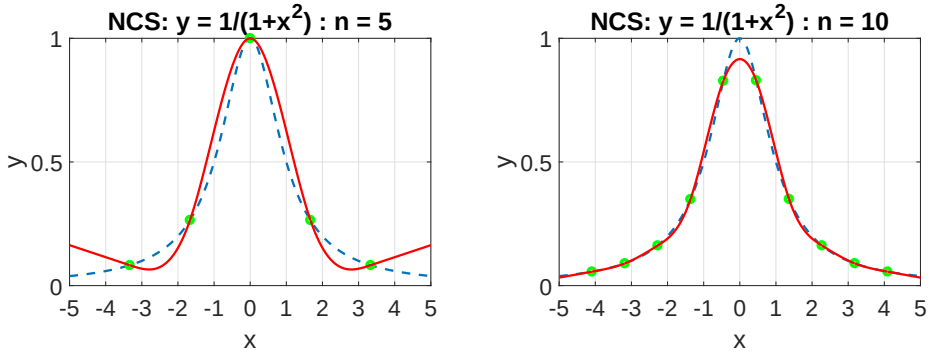

Figure 1: Natural cubic splines

2 Not-A-Knot Cubic Splines

The TestNNCS.m code is almost entirely written for you. You will just need to add the left and right boundary curves.

Given n data points (x_i, y_i), $i = 1, ..., n$, this algorithm first delete x_2 and x_{n-1} for constructing the $(n-3)$ cubic curves, i.e., there are $(n-3)4$ coefficients in the vector B. Unlike natural cubic splines which assume linear functions outside x_1 and x_n, this "not-a-knot" method uses two constraints: $y_2 = s(x_2)$ and $y_{n-1} = s(x_{n-1})$, in order to have in total $(n-1)4$ equations.

```
function TestNNCS(n)

dx = 0.001; maxX = 5; minX = -5;
X = minX:dx:maxX;
Y = 1./(1+X.^2);

t = round(length(X)/(n+1));
ind = t:t:n*t;
x = X(ind); y = Y(ind);
B = NNCS(x,y);

figure;
plot(X,Y,'--','linewidth',2);hold on; grid on;
plot(x,y,'go','linewidth',3);

xx = minX:dx:x(3);
%% figure out yy for plotting the left bounary curve
plot(xx,yy,'k-','linewidth',2);

for i = 3:length(x)-2
    xx = x(i):dx:x(i+1);
    b = B((i-2)*4+1:(i-1)*4);
    yy = 0;
    for j = 1:4
        yy = b(j)*xx.^(4-j)+yy;
    end
    plot(xx,yy,'k-','linewidth',2);
end

xx = x(end-2):dx:maxX;
%% figure out yy for plotting the right bounary curve
plot(xx,yy,'k-','linewidth',2);

set(gca,'fontsize',20);xlabel('x');ylabel('y');
set(gca,'xtick',min(X):1:max(X));

title(['NNCS: y = 1/(1+x^2) : n = ' num2str(n)]);

%Y = spline(x,y,X);plot(X,Y,'r-.','linewidth',2);
%your curve should be idential to matlab own implementation.
```

3 Submission Instruction

Your submission should include the required four matlab files to Sakai in one zipped file.

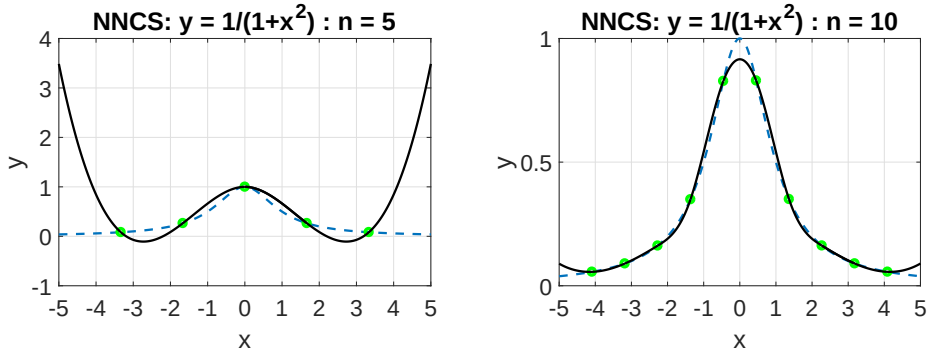

Figure 2: Not-A-Knot cubic spline