Math 4NA3 - Assignment 1

January 26, 2021

Question 1:

Since f is a polynomial it is infinitely differentiable and in particular it is of degree at most N so its derivatives of order greater than N are zero. Now the error of the Lagrange polynomial P_N for any x is given by

$$E_N(x) = f(x) - P_N(x) = \frac{f^{(N+1)}(\xi)}{(N+1)!} \prod_{i=0}^{N} (x - x_i)$$

Therefore, the error is identically zero as $f^{(N+1)} = 0$ and so $f(x) = P_N(x)$ for all x.

Question 2:

a) First we derive the quadratic Lagrange polynomial.

$$P_2(x) = f(x_0)\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f(x_1)\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + f(x_2)\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$$

$$= (1)\frac{(x-1.25)(x-1.5)}{(1-1.25)(1-1.5)} + (1.3217)\frac{(x-1)(x-1.5)}{(1.25-1)(1.25-1.5)} + (1.8371)\frac{(x-1)(x-1.25)}{(1.5-1)(1.5-1.25)}$$

$$= (8x^2 - 22x + 15) + (-21.1472x^2 + 52.868x - 31.7208) + (14.6968x^2 - 33.0678x + 18.371)$$

$$= 1.5496x^2 - 2.1998x + 1.6502$$

b) The average value of $f(x) = x^x$ over $[1, 1.5]$ is approximately

$$\frac{1}{1.5-1}\int_1^{1.5}(1.5496x^2 - 2.1998x + 1.6502)dx = 1.35398$$

c) The error estimate of approximating $f(x)$ by $P_2(x)$ is given by

$$E_2(x) = f(x) - P_2(x) = \frac{f^{(3)}(\xi)}{(3)!}\prod_{i=0}^{2}(x - x_i)$$

Note that $f^{(3)}(x) = x^x(\log(x)+1)^3 + x^{x-1}\left(\log(x) + \frac{x-1}{x}\right) + 2x^{x-1}(\log(x)+1)$ which is monotonically increasing so we use $\xi = 1.5$ as an upper bound. Then we have

$$E_2(x) = \frac{9.4478}{6}(x-1)(x-1.25)(x-1.5) = 1.5746(x^3 - 3.75x^2 + 4.625x - 1.875)$$

$$= 1.5746x^3 - 5.90475x^2 + 7.282525x - 2.952375$$

Question 3:

a) Here is my function to perform Newton Interpolation. One could argue to use Gregory Newton Interpolation since the x values are equally spaced, however, I wanted to make a more general function.

```
function [val,p] = inter(x, y, q)
% Performs Newton Interpolation
% INPUT:
% x,y: Input data as row vecotrs
% q: Row vector of values to approximate using the polynomial
%
% OUTPUT:
% val: Row vector of polynomial approximations at values in q
% p: Row vector of Newton coefficents in increasing order

p = zeros(1,length(x));
p(1) = y(1);
fprev = (x(2:end) - x(1:end-1)).\(y(2:end) - y(1:end-1));
p(2) = fprev(1);

for i = 3:length(x)
    fnext = (x(i:end) - x(1:end-i+1)).\diff(fprev);
    p(i) = fnext(1);
    fprev = fnext;
end

val = zeros(1,length(q));
for l = 1:length(q)
s = ones(1,length(x));
    for k = 2:length(x)
        s(k) = s(k-1)*(q(l)-x(k-1));
    end
val(l) = sum(p.*s);
end
end
```

b) Here is my code to calculate the errors. I present the coefficients, the approximated values, and the relative errors.

```
x =[-5,-4,-3,-2,-1,0,1,2,3,4,5];
y =
    [-0.1923,-0.2353,-0.30,-0.40,-0.50,0.00,0.50,0.40,0.30,0.2353,0.1923];

n=1:10;
q = (-5.5+1*n);

```

```
[val,p] = inter(x,y,q);

exact = arrayfun(@(x) x/(1+x^2),q);
error = zeros(1,length(val));

for t = 1:length(val)
    error(t) = abs((val(t) - exact(t))/exact(t));
end

p
val
error
```

```
p =

  Columns 1 through 8

   -0.1923   -0.0430   -0.0108   -0.0023    0.0020    0.0043   -0.0032    0.0010

  Columns 9 through 11

   -0.0002    0.0000         0


val =

  Columns 1 through 8

    0.1286   -0.3503   -0.2985   -0.5098   -0.3132    0.3132    0.5098    0.2985

  Columns 9 through 10

    0.3503   -0.1286


error =

  Columns 1 through 8

    1.6073    0.3263    0.1344    0.1045    0.2170    0.2170    0.1045    0.1344

  Columns 9 through 10

    0.3263    1.6073
```

The relative errors are quite small except for the points $x = \pm 4.5$. The errors are reasonable since they are small close to the centre but tend to grow larger as you approach farther regions; could be a case of polynomial wiggle. Of course you could adjust the interpolation if the regions near $x = \pm 4.5$ are important. Also notice that the errors are symmetric since the underlying function is odd.

c) Here is the requested plot.

```
plot(x,y, "r*")
hold on
grid on
n = 0:100;
q = (-5+0.1*n);
[val,p] = inter(x,y,q);
plot(q,val)
fplot(@(x) x/(1+x^2),"b")
```

As seen from the previous output, the error is relatively small for $x \in [-4, 4]$ but the plot suffers from polynomial wiggle as the error dramatically increases outside this range. We can also see the safest region is clearly $x \in [-1, 1]$.

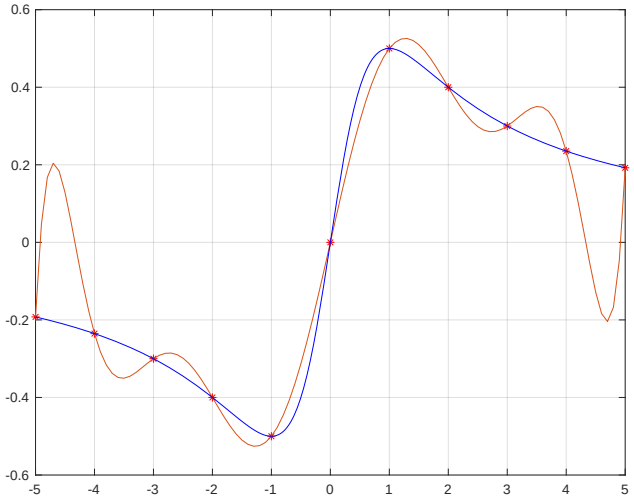

Figure 1: Original (Blue) and Interpolant (Orange)

Question 4:

a) To compute the requested clamped spline I used the function csfit.m from avenue. Here is the code and plot.

```
X = [0.5  1  1.5  2];
Y = arrayfun(@(x) x + 2/x, X);
type = 1;
dx0 = -7;
dxn  = 0.5;

S = csfitt(X,Y,dx0,dxn,type)
```

```
S =

   -7.0667   11.5333   -7.0000    4.5000
   -0.1333    0.9333   -0.7667    3.0000
   -0.4000    0.7333    0.0667    2.8333
```

```
plot(X,Y,"r*")
hold on
grid on
fplot(@(x) x + 2/x,[0.5,2],"r")
fplot(@(x) (S(1,1))*((x-0.5)^3)+(S(1,2))*((x-0.5)^2)+S(1,3)*(x-0.5)+S(1,4),[0.5,1],"--b")
fplot(@(x) (S(2,1))*(x-1)*(x-1)*(x-1)+(S(2,2))*(x-1)*(x-1)+S(2,3)*(x-1)+S(2,4),[1,1.5],"--b")
fplot(@(x) (S(3,1))*((x-1.5)^3)+(S(3,2))*((x-1.5)^2)+S(3,3)*(x-1.5)+S(3,4),[1.5,2],"--b")
```

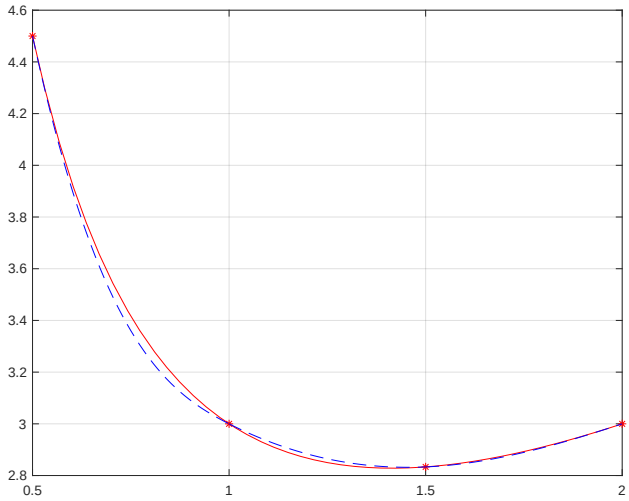

Figure 2: Original (Red) and Clamped Spline (Blue)

b) Again using the same process for the natural cubic spline.

```
X = [0.5  1  1.5  2];
Y = arrayfun(@(x) x + 2/x, X);
type = 0;
dx0 = 0;
dxn  = 0;

S = csfitt(X,Y,dx0,dxn,type)
```

```
S =
    2.6667         0   -3.6667    4.5000
   -2.6667    4.0000   -1.6667    3.0000
   -0.0000    0.0000    0.3333    2.8333
```

```
plot(X,Y,"r*")
hold on
grid on
fplot(@(x) x + 2/x,[0.5,2],"r")
fplot(@(x) (S(1,1))*((x-0.5)^3)+(S(1,2))*((x-0.5)^2)+S(1,3)*(x-0.5)+S
    (1,4),[0.5,1],"--b")
```

```
fplot(@(x) (S(2,1))*(x-1)*(x-1)*(x-1)+(S(2,2))*(x-1)*(x-1)+S(2,3)*(x-1)
    +S(2,4),[1,1.5],"--b")
fplot(@(x) (S(3,1))*((x-1.5)^3)+(S(3,2))*((x-1.5)^2)+S(3,3)*(x-1.5)+S
    (3,4),[1.5,2],"--b")
```

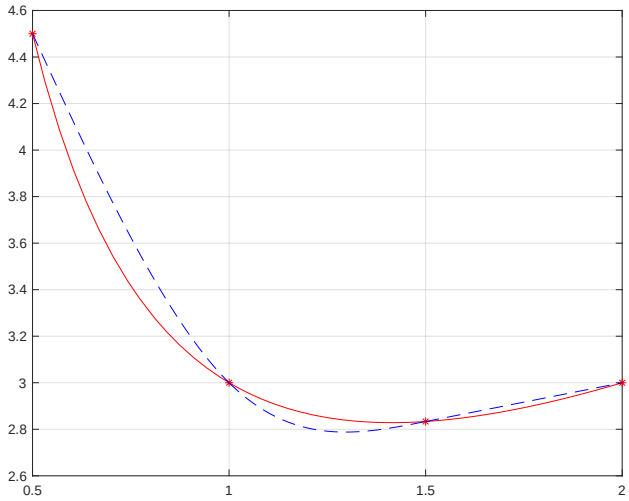

Figure 3: Original (Red) and Natural Spline (Blue)

c) From the plots, it is clear that the clamped cubic spline travels much tighter to the original function when compared to the natural cubic spline so the error is smaller for the clamped spline. Another interesting factor is the clamped spline travels on the opposite side of the true curve, that is, when the clamped spline overshoots the true curve, the natural spline undershoots it. One last comment is the coefficients for the clamped spline were all nonzero, however, the natural spline had several zero coefficients.