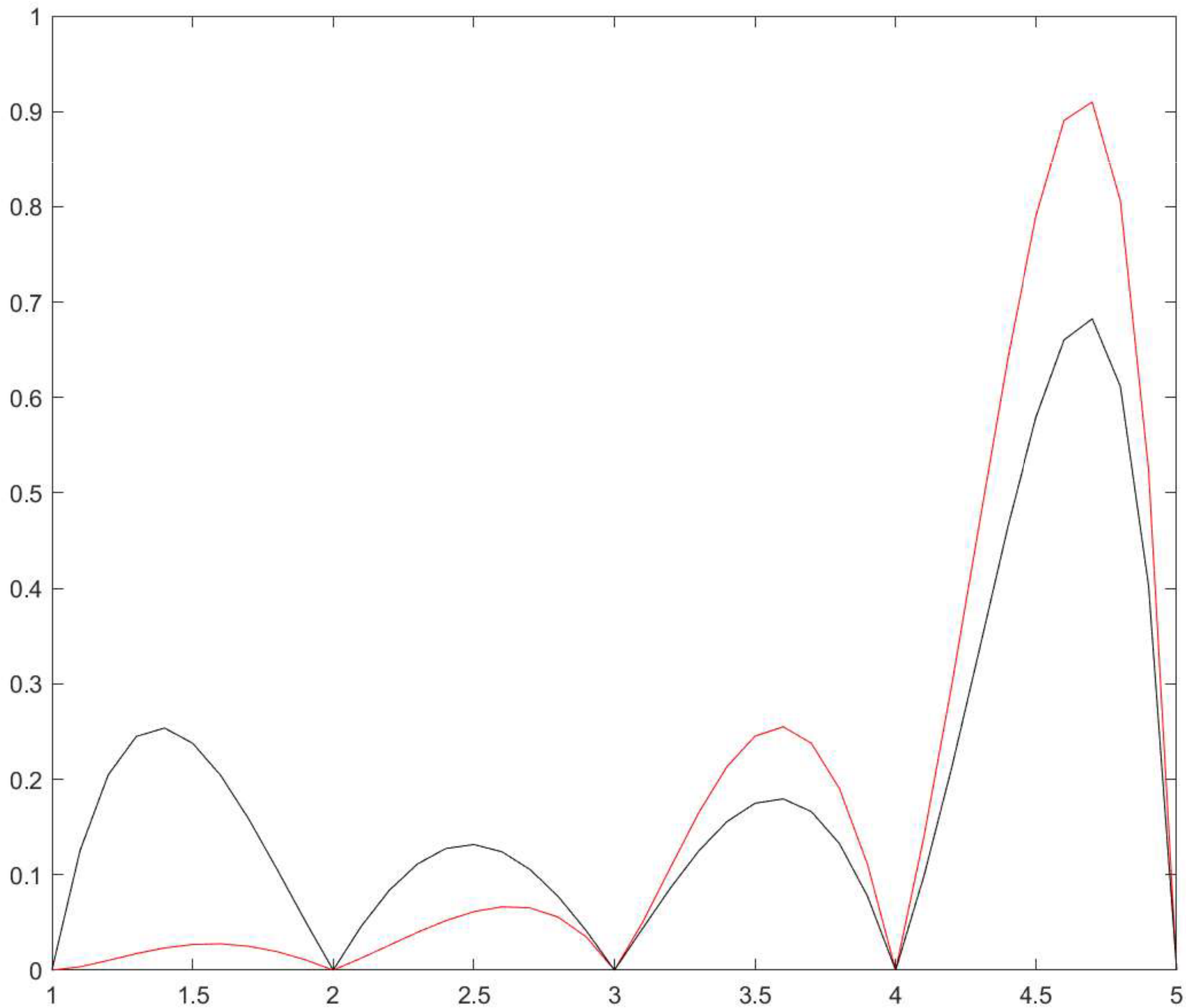
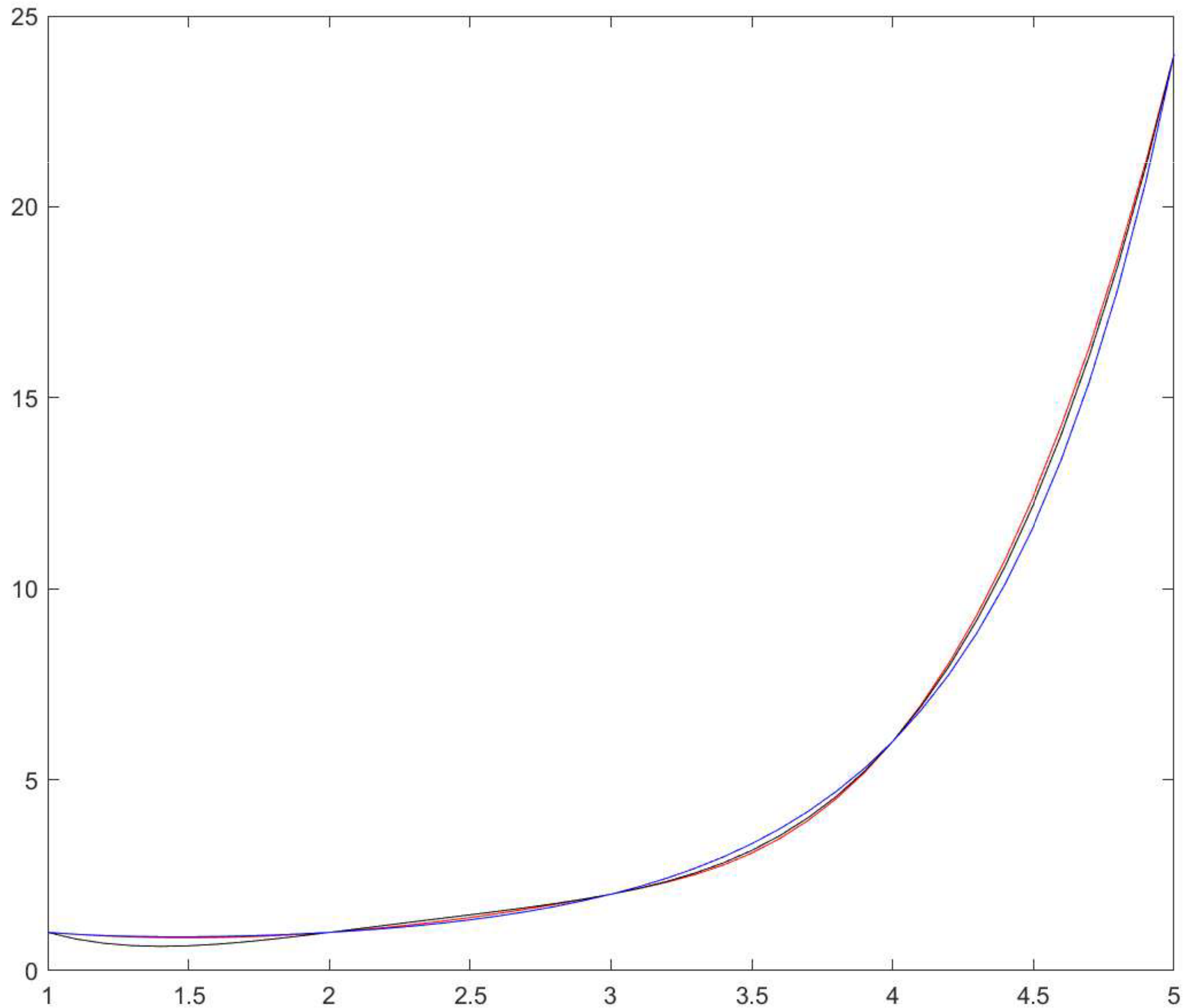
```
>> a = [1,2,3,4,5];
>> b = [1,1,2,6,24];
>> c = 1:0.1:5;
>> cs = cubicspline(a,b,c);
>> ndd = NDD(a,b,c);
>> g = gamma(c);
>> plot(c,cs,'red',c,ndd,'black',c,g,'blue')
>> plot(c,abs(cs-g),'red',c,abs(ndd-g),'black')
>> %the ndd had a more consistent error for the first 3 sub intervals, only really ↙
blowing up after the 4
>> %this is makes sense because this area should be close to linear which doesn't ↙
mesh well with how ndd forms its polynomial
>> %for the cubic spline, the last interval caught me off gaurd intially but when i ↙
thought back to what not a knot is doing
>> %it makes more sense because it would limit the third derivative to 0 which isn't ↙
going to be the case for the underlying function
>> a = [0.1,0.15,0.2,0.3,0.35,0.5,0.75];
>> b = [3,2,1.2,2.1,2.0,2.5,2.5];
>> c = 0.05:0.01:0.8;
>> cs = cubicspline(a,b,c);
>> ndd = NDD(a,b,c);
>> plot(c,cs,'red',c,ndd,'black')
>> plot(c,ndd,'black')
>> plot(c,cs,'black')
>> plot(c,ndd,'black')
>> plot(c,ndd,'black',a,b,'o')
>> ndd = NDD(a,b,c);
File: NDD.m Line: 17 Column: 186
Invalid expression. When calling a function or indexing a variable, use parentheses. ↙
Otherwise, check for mismatched delimiters.

>> ndd = NDD(a,b,c);
File: NDD.m Line: 17 Column: 186
Invalid expression. When calling a function or indexing a variable, use parentheses. ↙
Otherwise, check for mismatched delimiters.

>> ndd = NDD(a,b,c);
>> plot(c,ndd,'black',a,b,'o')
>> plot(c,cs,'red',c,ndd,'black')
>> % the ndd varies wildly between our data points compared to cubic splines, showing ↙
the benefits of cubic splines, and why generally they are the better approach for ↙
looking at more realistic data
>> %the near linear nature between some of the points highlighted the struggle of ↙
polynomial interpolants have to fit these types of data
>> %I am making assumptions about the data here, but it is not unreasonable given the ↙
lack of derivative information
>>
```

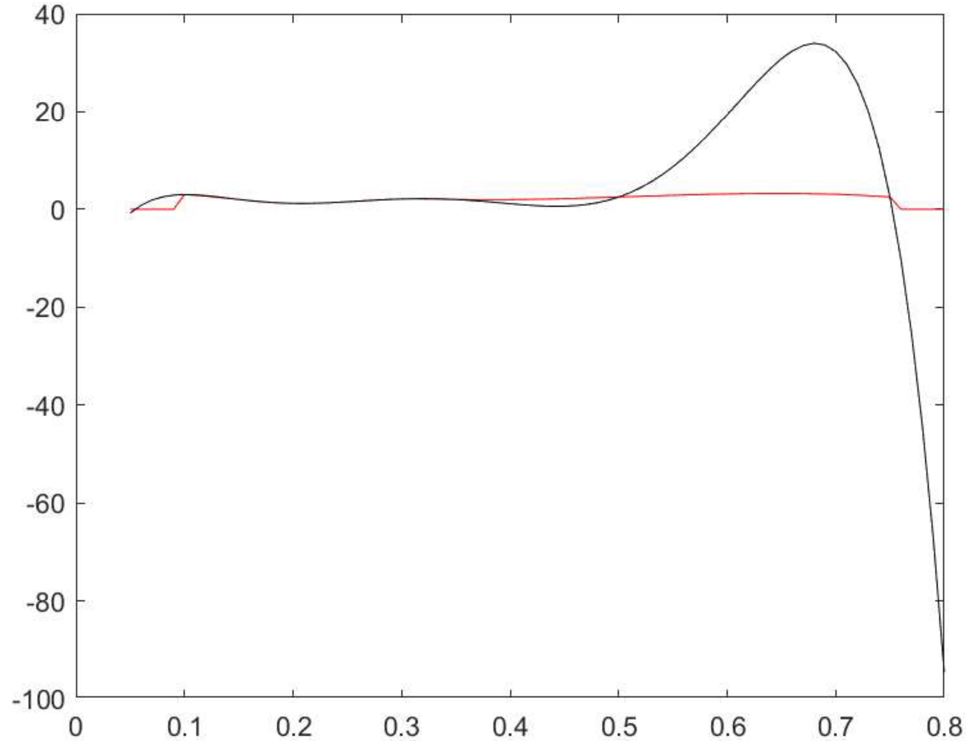


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
1.5
2
2.5
3
3.5
4
4.5
5

```
function Y=cubicspline(x,f,X)

n = length(x);

b = zeros(4*(n-1),1);
A = zeros(4*(n-1),4*(n-1));

%solves for a not a knot case;
%I decided to go for not a knot, knot just because it followed the example
%code you provided us with, but because i think it is more prudent to focus
%on the the center portions of the interval
for i = 1:n-1
    %next two lines are setting up the basic polynomial system via monomial
    A(2*i-1, 4*(i-1)+1:4*(i)) = [1,x(i),x(i).^2,x(i).^3];
    A(2*i, 4*(i-1)+1:4*(i)) = [1,x(i+1),x(i+1).^2,x(i+1).^3];
    %these two lines
    b(2*i-1) = f(i);
    b(2*i) = f(i+1);
    if i==n-1
        A(4*(n-1)-1,1:8) = [0 0 0 6 0 0 0 -6];
        A(4*(n-1),4*(n-3)+1:4*(n-1)) = [0 0 0 6 0 0 0 -6];
    else
        A(2*(n-1)+i,4*(i-1)+1:4*(i+1)) = [0,1,2*x(i+1),3*x(i+1).^2,0,-1,-2*x(i+1),-3 ↙
*x(i+1).^2];
        A(3*(n-1)+i-1,4*(i-1)+1:4*(i+1)) = [0,0,2,6*x(i+1),0,0, -2,-6*x(i+1)];
    end
end

C = A\b;

m = length(X);
Y = zeros(1,m);


for i = 1:m
    for j = 1:n-1
        if X(i) <= x(j+1) && X(i) >= x(j)
            Y(i) = C(1+4*(j-1)) + C(2+4*(j-1))*(X(i)) + C(3+4*(j-1))*(X(i)).^2 +C ↙
(4+4*(j-1))*(X(i)).^3;
        else
        end
    end
end

%plot(X,Y,x,f,'o')
```

```
function Y = NDD(x,f,X)

n = length(x);

C = zeros(n);

C(:,1) = f';
for i = 2:n
    for j = i:n
        C(j,i) = (C(j,i-1)-C(j-1,i-1))/(x(j)-x(j-(i-1)));
    end
end

p = diag(C);

%Y = p(1) + p(2)*(X-x(1)) + p(3)*(X-x(1)).*(X-x(2)) + p(4)*(X-x(1)).*(X-x(2)).*(X-x
(3)) + p(5)*(X-x(1)).*(X-x(2)).*(X-x(3)).*(X-x(4));
Y = p(1) + p(2)*(X-x(1)) + p(3)*(X-x(1)).*(X-x(2)) + p(4)*(X-x(1)).*(X-x(2)).*(X-x
(3)) + p(5)*(X-x(1)).*(X-x(2)).*(X-x(3)).*(X-x(4))+p(6)*(X-x(1)).*(X-x(2)).*(X-x(3)).
*(X-x(4)).*(X-x(5))+p(7)*(X-x(1)).*(X-x(2)).*(X-x(3)).*(X-x(4)).*(X-x(5)).*(X-x(6));
```


25
20
15
10
5
0
1
1.5
2
2.5
3
3.5
4
4.5
5


40
20
0
-20
-40
-60
-80
-100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8