Math 104A - Homework 4

Due 7/14

1 Write a function that takes as input an interval $[a, b]$, a number subintervals N, and a function f, and outputs the approximate derivative of f at the nodes $x_i = a + i \cdot h$, for $i = 0, \ldots, N$, where $h = \frac{b-a}{N}$. Use the three point formulas:

$$f'(x_0) = \frac{1}{2h}[-3f(x_0) + 4f(x_1) - f(x_2)], \tag{1}$$

$$f'(x_i) = \frac{1}{2h}[-f(x_{i-1}) + f(x_{i+1})], \quad \text{for} \quad i = 1, \ldots, N-1, \tag{2}$$

$$f'(x_N) = \frac{1}{2h}[f(x_{N-2}) - 4f(x_{N-1}) + 3f(x_N)]. \tag{3}$$

An example code template in MatLab is:

```
function yp = approx_deriv(a,b,N,fstring)

    f = inline(fstring);
    x = linspace(a,b,N+1);
    y = f(x);

    % code for approximating the derivative goes here

end
```

See the attached code (`approx_deriv.m`).

2 Write another function that plots the results from problem 1. It should take as input the interval $[a, b]$, the number of subintervals N, the function f, and the derivative f'. The function should plot both the exact derivative and the approximate derivative at the nodes $x_i = a + i \cdot h$. An example code template is:

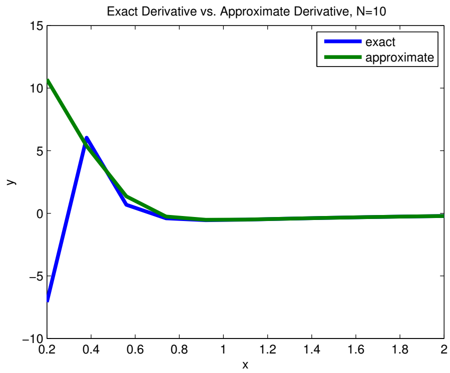
```
function [] = plot_approx_deriv(a,b,N,fstring,fpstring)

    fp = inline(fpstring);
    x = linspace(a,b,N+1);
    yp = fp(x);
    yp_approx = approx_deriv(a,b,N,fstring);

    % code for plotting results goes here

end
```

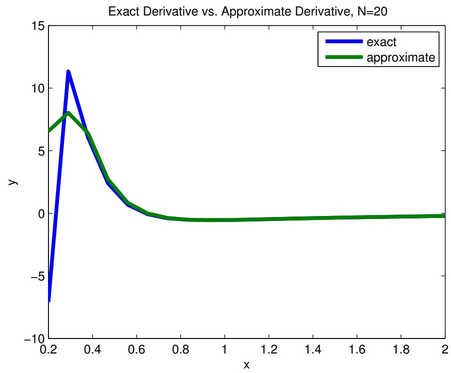
Test your code for the interval $[0.2, 2]$ on the function $f(x) = \sin(1/x)$, for $N = 10, 20, 40, 80, 160$. **Label your plots clearly** (i.e. include a legend, label the axes, give a title) and include them in your homework.

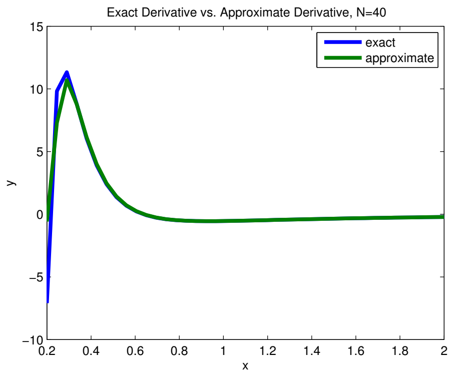
Using the attached code (`plot_approx_deriv.m`) and the following commands:

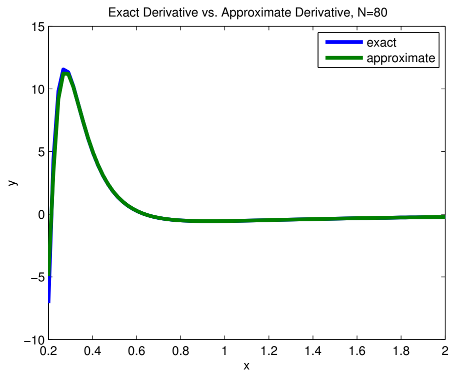
```
>> plot_approx_deriv(0.2,2,10,'sin(1./x)','-cos(1./x)./(x.^2)');
>> plot_approx_deriv(0.2,2,20,'sin(1./x)','-cos(1./x)./(x.^2)');
>> plot_approx_deriv(0.2,2,40,'sin(1./x)','-cos(1./x)./(x.^2)');
>> plot_approx_deriv(0.2,2,80,'sin(1./x)','-cos(1./x)./(x.^2)');
>> plot_approx_deriv(0.2,2,160,'sin(1./x)','-cos(1./x)./(x.^2)');
```

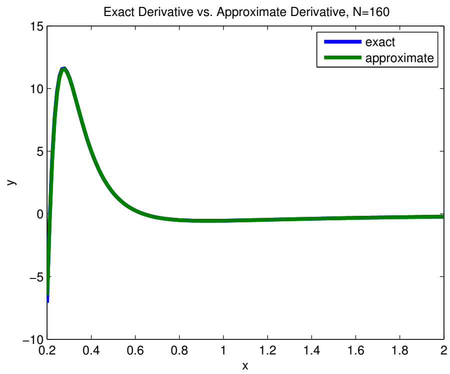
we get the following plots:











3 Write a code that implements composite Simpson's rule (algorithm 4.1 in the book). Test your code for the interval $[0, 1]$ on the function $f(x) = \frac{e^x+e^{-x}}{2}$ with $N = 10, 20, 40, 80, 160$. An example code template is:

```
function I = comp_Simpsons(a,b,N,fstring)

    f = inline(fstring);

    % code for composite Simpson's rule goes here.
```

```
end
```

Using the attached code (`composite_Simpsons.m`), we get

```
>> composite_Simpsons(0,1,10,'(exp(x)+exp(-x))/2')
ans =
   1.175201845756919

>> composite_Simpsons(0,1,20,'(exp(x)+exp(-x))/2')
ans =
   1.175201234437257

>> composite_Simpsons(0,1,40,'(exp(x)+exp(-x))/2')
ans =
   1.175201196193961

>> composite_Simpsons(0,1,80,'(exp(x)+exp(-x))/2')
ans =
   1.175201193803196

>> composite_Simpsons(0,1,160,'(exp(x)+exp(-x))/2')
ans =
   1.175201193653764
```

all of which are close to the true solution $\sinh(1) \approx 1.17520119$.

4 Write a code that has as input an interval $[a, b]$, a function f, the antiderivative F, and a *vector* of N's, and plots the error for each of the N's versus the step sizes h. An example code template is:

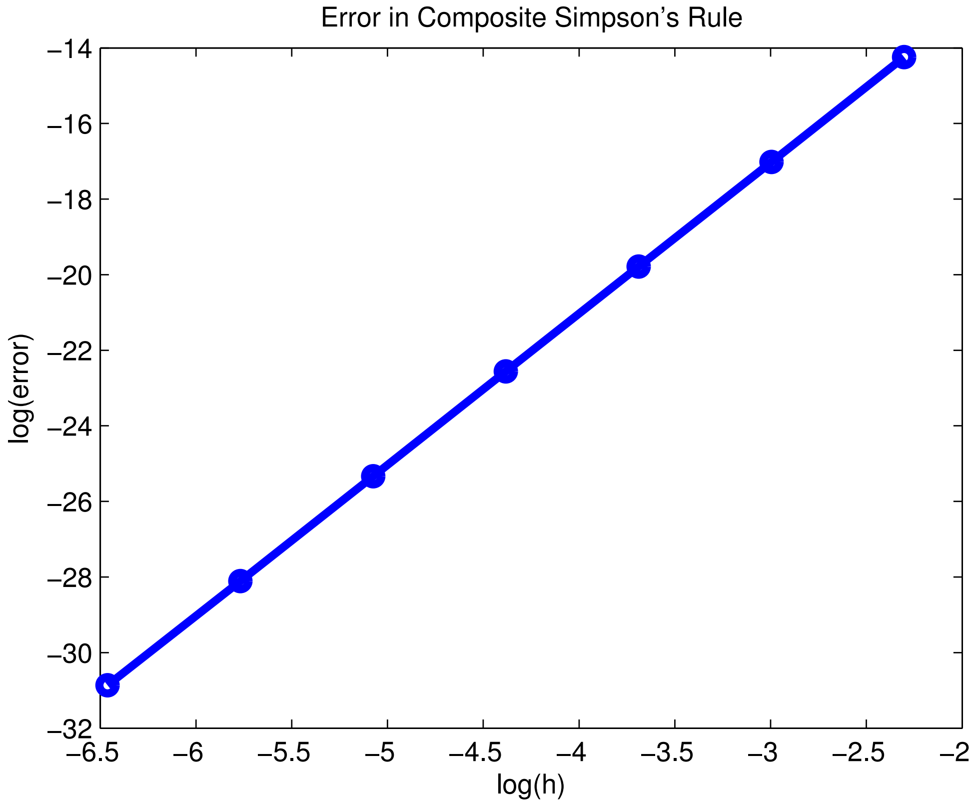
```
function [] = plot_comp_Simpsons_error(a,b,fstring,Fstring,Nvec)

    f = inline(fstring);
    F = inline(Fstring);

    n = length(Nvec);
    hvec = zeros(n,1);
    errorvec = zeros(n,1)

    % code for calculating the error and h for each N goes here.

    % code for plotting the error goes here.

end
```

Now modify your code to plot the *logarithm* of the error versus the *logarithm* of h. Use this modified code to plot the errors for $N = 10, 20, 40, 80, 160, 320, 640$ using the same interval and function as in problem 3. As before, make sure to **label your plots clearly**. How does the slope of your log-log plot compare to the error term given for composite Simpson's rule?

Using the attached code (`plot_comp_Simpsons_error.m`) and using the following command:

```
>> plot_comp_Simpsons_error(0,1,'(exp(x)+exp(-x))/2',...
       '(exp(x)-exp(-x))/2',[10,20,40,80,160,320,640]);
```

we get the following plot:



We see that the plot is nearly linear; finding the slope between the first point ($h = 1/640$) and last point ($h = 1/10$), we find that

$$\text{slope} \approx \frac{-30.867503656543327 + 14.243047796621255}{-6.461468176353717 + 2.302585092994045} = 3.9973 \approx 4$$

This number is significant. The error term in composite Simpson's rule is of the form $E(h) = Ch^4$. Then letting $x = \ln(h)$ and $y = \ln(E(h))$, we see that

$$\begin{aligned} E(h) &= Ch^4 \\ \ln(E(h)) &= 4\ln(h) + \ln(C) \\ y &= 4x + C, \end{aligned}$$

that is, we expected to see a line with slope approximately 4.

Code

```
%%%%% begin approx_deriv.m %%%%%

function yp = approx_deriv(a,b,N,fstring)

    f = inline(fstring);
    x = linspace(a,b,N+1);
    y = f(x);

    h = (b-a)/N;

    yp = zeros(size(x));
    yp(1) = (-3*y(1)+4*y(2)-y(3))/(2*h);
    yp(2:N) = (-y(1:N-1)+y(3:N+1))/(2*h);
    yp(N+1) = (y(N-1)-4*y(N)+3*y(N+1))/(2*h);

end

%%%%% end of approx_deriv.m %%%%%



%%%%% begin plot_approx_deriv.m %%%%%

function [] = plot_approx_deriv(a,b,N,fstring,fpstring)

    fp = inline(fpstring);
    x = linspace(a,b,N+1);
    yp = fp(x);
    yp_approx = approx_deriv(a,b,N,fstring);

    plot(x,yp,x,yp_approx,'Linewidth',3);
    legend('exact','approximate');
    title(sprintf('Exact Derivative vs. Approximate Derivative, N=%i',N));
    xlabel('x');
    ylabel('y');

end

%%%%% end of plot_approx_deriv.m %%%%%



%%%%% begin composite_Simpsons.m %%%%%

function I = composite_Simpsons(a,b,N,fstring)
```

```
    f = inline(fstring);

    h = (b-a)/N;
    I0 = f(a)+f(b);
    I1 = 0;
    I2 = 0;

    for i=1:N-1
        X = a+i*h;
        if( mod(i,2) == 0 )
            I2 = I2 + f(X);
        else
            I1 = I1 + f(X);
        end
    end

    I = h/3*(I0+4*I1+2*I2);

end

%%%%% end of composite_Simpsons.m %%%%%



%%%%% begin plot_comp_Simpsons_error.m %%%%%

function [hvec,errorvec] = plot_comp_Simpsons_error(a,b,fstring,Fstring,Nvec)

    F = inline(Fstring);
    n = length(Nvec);
    hvec = zeros(n,1);
    errorvec = zeros(n,1);

    for i=1:n
        hvec(i) = (b-a)/Nvec(i);
        I_approx = composite_Simpsons(a,b,Nvec(i),fstring);
        I_exact = F(b)-F(a);
        errorvec(i) = abs(I_exact-I_approx);
    end

    plot(log(hvec),log(errorvec),'bo-','Linewidth',3);
    title('Error in Composite Simpson''s Rule');
    xlabel('log(h)');
    ylabel('log(error)');

end

%%%%% end of plot_comp_Simpsons_error.m %%%%%
```