Math 104A - Homework 5

Due 7/21

1 Write a code that implements Gaussian quadrature with 3 nodes. The quadrature is given by

$$\int_1^{-1} f(x)dx \approx a_0 f(x_0) + a_1 f(x_1) + a_2 f(x_2),$$

where $x_0 = -\sqrt{\frac{3}{5}}, x_1 = 0, x_2 = \sqrt{\frac{3}{5}}$ and $a_0 = \frac{5}{9}, a_1 = \frac{8}{9}, a_2 = \frac{5}{9}$. The code should take as input the interval $[a, b]$ and the integrand f. An example code template:

```
function I = Gaussian_quadrature(a,b,fstring)

    f = inline(fstring);

    % code for Gaussian quadrature goes here.

end
```

Note: you will need to do the change of variables described in lecture/the book in order to get an approximation over the *arbitrary* interval $[a, b]$.

See attached code (`Gaussian_quadrature.m`).

2 Write a code that implements a *composite* Gaussian quadrature. What this means is the code should break the total interval into smaller subintervals, perform Gaussian quadrature on each subinterval, and sum the results. The code should take as input the interval $[a, b]$, the number of subintervals N, and the integrand f. An example code template:

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

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

    % Code for composite Gaussian quadrature goes here.
    % This should involve the code from the previous problem.

end
```

See attached code (`composite_Gaussian_quadrature.m`). As a check to see if the code is working correctly, we test it on the integral $\int_0^{\pi} \sin(x)dx$:

```
>> composite_Gaussian_quadrature(0,pi,10,'sin(x)')

ans =
   2.000000000956971
```

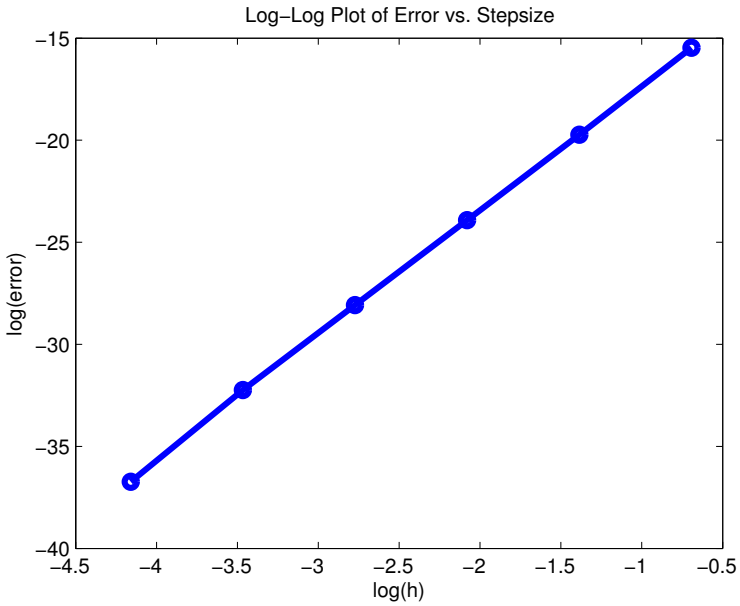
which agrees closely with the actual value 2.

3 Test your composite Gaussian quadrature using the function $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \tanh(x)$ on the interval $[0, 1]$ (the antiderivative of $\tanh(x)$ is $\ln(\cosh(x))$) for values $N = 2, 4, 8, 16, 32, 64$. Make a log-log plot of the absolute error versus the step-size $h = \frac{b-a}{N}$. As before, **be sure to label your plot clearly**. How does this quadrature compare to composite Simpson's method from the previous homework?

Using the attached code (`plot_composite_Gaussian_quadrature_error.m`), with the following arguments:

```
>> [h,E] = plot_composite_Gaussian_quadrature_error ...
    (0,1,'tanh(x)','log(cosh(x))',[2 4 8 16 32 64]);
```

we get the following plot:



We find the slope of the line via

```
>> ( log(E(6))-log(E(1)) )/( log(h(6))-log(h(1)) )

ans =
   6.135422302774067
```

Thus, the order of convergence of composite Gaussian quadrature is approximately $O(h^6)$, which is two orders higher than that of composite Simpson's rule, which was found to be $O(h^4)$.

4 Write a code that implements Euler's method (algorithm 5.1 in the book). The code should take as input the interval $[a, b]$, the number of subintervals N, the initial condition α, and the r.h.s. of the differential equation $f(t, y)$. The code should return the approximate solution w_i at the meshpoints $t_i = a + ih$. An example code template:

```
function w = Eulers_method(a,b,N,alpha,fstring)

    % This converts the string representation of f into
    % an actual function. An example function string would
    % be 'y - t.^2 + 1'
    f = inline(fstring,'t','y');

    % This sets up the mesh points. N+1 because if there are
    % N subintervals, there are N+1 meshpoints.
    t = linspace(a,b,N+1);

    % This initializes the output vector. Initially, it
    % contains all zeros for each of the mesh points. The
    % first solution point is set to the initial condition.
    w = zeros(size(t));
    w(1) = alpha;

    %%%%% Code for Euler's method goes here. %%%%%

end
```

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

5 Test your code on the initial value problem

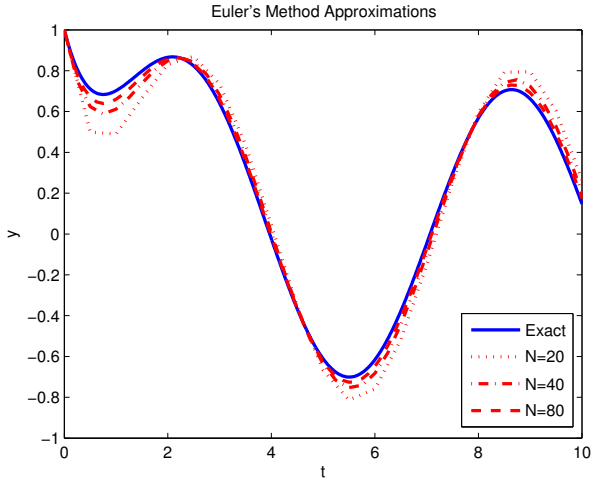
$$y'(t) = -y + \sin(t), 0 \leq t \leq 10, y(0) = 1.$$

Plot your approximation with number of subintervals $N = 20, 40, 80$, along with a plot of the exact solution $y(t) = \frac{1}{2}(3e^{-t} + \sin(t) - \cos(t))$. Again, **be sure to label your plot clearly**.

Using the attached code (`plot_Eulers_method.m`), and using the following inputs:

```
>> plot_Eulers_method(0,10,1,'-y+sin(t)','(3*exp(-t)+sin(t)-cos(t))/2');
```

we get the following plot:



Code

```
%%%%% begin Gaussian_quadrature.m %%%%%

function I = Gaussian_quadrature(a,b,fstring)

    f = inline(fstring);
    nodes = [-sqrt(3/5) 0 sqrt(3/5)];
    weights = [5/9 8/9 5/9];

    I = 0;
    for i=1:3
        I = I + weights(i)*f((b-a)/2*nodes(i)+(a+b)/2);
    end
    I = (b-a)/2*I;

end

%%%%% end of Gaussian_quadrature.m %%%%%



%%%%% begin composite_Gaussian_quadrature.m %%%%%

function I = composite_Gaussian_quadrature(a,b,N,fstring)

    intervals = linspace(a,b,N+1);

    I = 0;
    for i=1:N

        I = I + Gaussian_quadrature(intervals(i),intervals(i+1),fstring);

    end

end

%%%%% end of composite_Gaussian_quadrature.m %%%%%



%%%%% begin plot_composite_Gaussian_quadrature_error.m %%%%%

function [hvec,Evec] = plot_composite_Gaussian_quadrature_error ...
  (a,b,fstring,Fstring,Nvec)

    F = inline(Fstring);
    hvec = (b-a)./Nvec;
    Evec = zeros(size(hvec));
```

```
    for i=1:length(Nvec)

        Evec(i) = abs(composite_Gaussian_quadrature(a,b,Nvec(i),fstring) ...
                        - (F(b) - F(a)) );

    end

    plot(log(hvec),log(Evec),'bo-','LineWidth',3);
    title('Log-Log Plot of Error vs. Stepsize');
    xlabel('log(h)');
    ylabel('log(error)');

end

%%%%% end of plot_composite_Gaussian_quadrature_error.m %%%%%



%%%%% begin Eulers_method.m %%%%%

function [t,w] = Eulers_method(a,b,N,alpha,fstring)

    f = inline(fstring,'t','y');

    h = (b-a)/N;
    t = linspace(a,b,N+1);
    w = zeros(size(t));

    w(1) = alpha;
    for i=1:N
        w(i+1) = w(i) + h*f(t(i),w(i));
    end

end

%%%%% end of Eulers_method.m %%%%%



%%%%% begin plot_Eulers_method.m %%%%%

function [t,w] = plot_Eulers_method(a,b,alpha,fstring,solstring)

    sol = inline(solstring,'t');

    t = linspace(a,b,500);
    y = sol(t);
```

```
    [t1,w1] = Eulers_method(a,b,20,1,fstring);
    [t2,w2] = Eulers_method(a,b,40,1,fstring);
    [t3,w3] = Eulers_method(a,b,80,1,fstring);

    plot(t,y,'b-',t1,w1,'r:',t2,w2,'r-.',t3,w3,'r--','LineWidth',2);
    legend('Exact','N=20','N=40','N=80','Location','Best');
    xlabel('t');
    ylabel('y');
    title('Euler''s Method Approximations');

end

%%%%% end of plot_Eulers_method.m %%%%%
```