# ME140A - Homework 5

Due by 11:59PM, December 7th, by email to ameiburg@ucsb.edu.

Collaboration is encouraged! This homework is very short.

The following MATLAB code implements a simple iterative algorithm for solving the boundary value problem

$$x'' + \sin(x(t)) + 0.1x'(t) = 0$$

$$x(0) = 0, \quad x(10) = 1$$

It does solve it, and plot the solution, but it doesn't do a very good job. Your homework is just to make three simple modifications to improve this code.

1. Implement error tracking. As the values of $x[i]$ change, keep track of the total of how much they change in each pass. Add an automatic termination condition, to stop the passes when the total change is small; this way the user doesn't need to pass in a number of passes. Try a few numbers and make sure that it still converges to a correct value.

2. Improve the first-derivative estimation. This is lines 63-69. Currently it uses
$$x[i] \approx \frac{x[i+1] - x[i]}{dt}$$
which is a forward first-difference. A better expression is the five-point stencil,
$$f'(x) \approx \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h}$$
Use this instead to improve the accuracy of the method.

3. Currently each pass goes
$$i = 1, 2, \ldots N-1, N, 1, 2, \ldots N, 1 \ldots$$
which is not very efficient, as we talked about in class. It's preferrable to go "back and forth", like
$$i = 1, 2, \ldots N-1, N, N-1, N-2 \ldots 3, 2, 1, 2 \ldots$$
Modify the passes to use this better order.

The below code is also available on the website next to the link to the homework.

```
% solve x'' = -sin(x) - 0.1x', with boundary conditions x(0) = 5 and
% x(10)=6.


%Run algorithm with 100 points and 40 sweeps
npts = 100;
x_arr_40 = iterative(0, 1, npts, 40);
%Compare with 100 sweeps
x_arr_100 = iterative(0, 1, npts, 100);
%Compare with 1000 sweeps
x_arr_1k = iterative(0, 1, npts, 1000);
%Compare with 2000 sweeps
x_arr_2k = iterative(0, 1, npts, 2000);
%Compare with 4000 sweeps
x_arr_4k = iterative(0, 1, npts, 4000);

clf
hold on
plot(x_arr_40)
plot(x_arr_100)
plot(x_arr_1k)
plot(x_arr_2k)
plot(x_arr_4k)
legend('40','100','1k','2k','4k')
hold off

%Iterative solver. Takes number of points to discretize with, and a number
%of passes to do.
function x_arr = iterative(x0, x10, npts, npasses)

    %Array of x and x' values
    dt = 10/npts;
    x_arr = zeros(1,npts);
    xp_arr = zeros(1,npts); %xp for "x prime"

    %Set boundary values
    x_arr(1) = x0;
    x_arr(end) = x10;

    %do a number of passes
    for i_pass = 1:npasses
        %step through each point and update x_arr.
        %would be 1:npts, but we skip the endpoints, so just 2:npts-1.
```

```
        for ix = 2:npts-1
            %x[i-1], x[i], x[i+1]
            xi0 = x_arr(ix - 1);
            xi1 = x_arr(ix);
            xi2 = x_arr(ix + 1);
            %x'[i]
            xpi = xp_arr(ix);

            newxval = (1/2)*(xi0 + xi2 - dt*dt*(-sin(xi1)-0.1*xpi));
            x_arr(ix) = newxval;
        end

        %step through an update the derivative estimates using a forward
        %difference rule, x'[i] = (x[i+1]-x[i])/dt. This prevents us from
        %updating the last point though, so that one we use we
        %(x[i]-x[i-1])/dt instead.

        for ix = 1:npts-1 %skip last point
            %x[i], x[i+1]
            xi1 = x_arr(ix);
            xi2 = x_arr(ix + 1);

            xprime_val = (xi2 - xi1)/dt;
            xp_arr(ix) = xprime_val;
        end
        %last point ends up getting same value as previous one
        xp_arr(end) = xp_arr(end-1);
    end
end
```