Lab 12:
Numerical Differentiation and Integration

12.1 Introduction

This lab is aimed at calculating derivatives and integrals of discrete data in MATLAB. At the end of the lab, you should be able to write code that loads a data file and approximates its first derivative and second derivative as well as calculating a cumulative integral using two different methods.

12.2 Resources

The additional resources required for this assignment include:

- Books: None
- Pratt Pundit Pages: `X-Win 32` (in case X-Win asks for a new floating node license), `PuTTY` (in case the terminal says something about not being able to lock the `.Xauthority` file)

12.3 Getting Started

1. Log into one of the PCs in the lab using your NET ID. Be sure it is set to log on to acpub.
2. Start a browser and point it to `http://pundit.pratt.duke.edu/wiki/Lab:B209`. Check out the `Using the PCs to Run MATLAB Remotely` section for how to get connected and make sure the connection is working.
3. Once connected to a machine you believe will also display graphics, switch into your `EGR53` directory and create a `lab12` directory inside it:

   ```
   cd␣EGR53
   mkdir␣lab12
   ```

4. Switch to your `~/EGR53/lab12` directory:

   ```
   cd␣lab12
   ```

5. Copy all relevant files from Dr. G's public `lab12` directory:

   ```
   cp␣-i␣~mrg/public/EGR53/lab12/*␣.
   ```

   Do not forget the space and the "." at the end.

6. Open MATLAB by typing `matlab␣&` at the prompt that appears in your terminal window. It will take MATLAB a few seconds to start up.

12.4 Assignment

Rather than turning anything in, you will be graded by an automated script. Be sure to follow all the instructions closely, including putting files in the proper directory and calling variables the proper name. You have been given the checker code for yourself, so you can work on your program until you have it perfect.

12.5 Derivatives

Recall the following equations for estimating derivatives assuming a constant spacing in the x direction, denoted below as Δx. There are N total points.

2-pt Forward 1st Derivative	$\left.\frac{dy}{dx}\right\|_{x_k} = \frac{y_{k+1} - y_k}{\Delta x}$
Exception: k=N	$\left.\frac{dy}{dx}\right\|_{x_N} = \frac{y_N - y_{N-1}}{\Delta x}$

2-pt Backward 1st Derivative	$\left.\frac{dy}{dx}\right\|_{x_k} = \frac{y_k - y_{k-1}}{\Delta x}$
Exception: k=1	$\left.\frac{dy}{dx}\right\|_{x_1} = \frac{y_2 - y_1}{\Delta x}$

3-pt Centered 1st Derivative	$\left.\frac{dy}{dx}\right\|_{x_k} = \frac{y_{k+1} - y_{k-1}}{2\ \Delta x}$
Exception: k=1	$\left.\frac{dy}{dx}\right\|_{x_1} = \frac{-y_3 + 4y_2 - 3y_1}{2\ \Delta x}$
Exception: k=N	$\left.\frac{dy}{dx}\right\|_{x_N} = \frac{3y_N - 4y_{N-1} + y_{N-2}}{2\ \Delta x}$

3-pt Centered 2nd Derivative	$\left.\frac{d^2y}{dx^2}\right\|_{x_k} = \frac{y_{k+1} - 2y_k + y_{k-1}}{\Delta x^2}$
Exception: k=1	$\left.\frac{d^2y}{dx^2}\right\|_{x_1} = \frac{y_3 - 2y_2 + y_1}{\Delta x^2}$
Exception: k=N	$\left.\frac{d^2y}{dx^2}\right\|_{x_N} = \frac{y_N - 2y_{N-1} + y_{N-2}}{\Delta x^2}$

12.6 Integrals

Recall the following equations for integration assuming a constant spacing in the x direction Δx and using the Trapezoidal rule:

Trapezoidal Rule for 2 Points
$$\int_{x_1}^{x_2} y\ dx = \frac{\Delta x}{2}\left(y_1 + y_2\right)$$

Trapezoidal Rule for 3 Points
$$\int_{x_1}^{x_3} y\ dx = \frac{\Delta x}{2}\left(y_1 + 2y_2 + y_3\right)$$

Trapezoidal Rule for k >3 Points
$$\int_{x_1}^{x_k} y\ dx = \frac{\Delta x}{2}\left(y_1 + 2\left(\sum_{i=2}^{k-1} y_i\right) + y_k\right)$$

Also, you can use combinations of the Trapezoidal and Simpson's rules to get higher accuracy. For integration between two, three, or four points, you will use Trapezoidal, Simpson's 1/3rd, and Simpson's 3/8ths Rule, respectively:

Trapezoidal Rule for 2 Points
$$\int_{x_1}^{x_2} y\ dx = \frac{\Delta x}{2}\left(y_1 + y_2\right)$$

Simpson's 1/3rd Rule for 3 Points
$$\int_{x_1}^{x_3} y\ dx = \frac{\Delta x}{3}\left(y_1 + 4y_2 + y_3\right)$$

Simpson's 3/8ths Rule for 4 Points
$$\int_{x_1}^{x_4} y\ dx = \frac{3\ \Delta x}{8}\left(y_1 + 3y_2 + 3y_3 + y_4\right)$$

For five or more (odd) points you can use successive applications of Simpson's 1/3rd rule:

$k = 5$ or greater and odd - Simpson's 1/3rd Rule

$$\int_{x_1}^{x_k} y\ dx = \frac{\Delta x}{3}\left(y_1 + 4\left(\sum_{i=2,\ \text{even}}^{k-1} y_i\right) + 2\left(\sum_{j=3,\ \text{odd}}^{k-2} y_j\right) + y_k\right)$$

For six or more (even) points you can use Simpson's 3/8ths Rule for the last three intervals (the last four points) and successive applications of Simpson's 1/3rd Rule for the rest of the intervals:

$k = 6$ or greater and even - Simpson's 1/3rd Rule with one Simpson's 3/8ths Rule

$$\int_{x_1}^{x_k} y\ dx = \frac{\Delta x}{3}\left(y_1 + 4\left(\sum_{i=2,\ \text{even}}^{k-4} y_i\right) + 2\left(\sum_{j=3,\ \text{odd}}^{k-5} y_j\right) + y_{k-3}\right) + \frac{3\ \Delta x}{8}\left(y_{k-3} + 3y_{k-2} + 3y_{k-1} + y_k\right)$$

12.7 Help with Summations and Vectors

In several of the above equations, there are summations over a range of points - and sometimes excluding some of the points. Remember that you can use the index operator : to pick out particular points; you can then use the `sum` command to add up those particular points. For example, assuming you have some vector y with data in it and have defined k as the index of y to which you want to integrate and N as the total number of points, the following table will demonstrate what the code will look like for some of the specific integrals above:

$\sum_{i=2}^{k-1} y_i$	`sum(y(2:(k-1)))`
$\sum_{i=2,\ \text{even}}^{k-1} y_i$	`sum(y(2:2:(k-1)))`
$\sum_{j=3,\ \text{odd}}^{k-5} y_j$	`sum(y(3:2:(k-5)))`

Also note that you will be wanting to store an entire vector of integrals. The easiest way to do this for most of the derivatives and integrals will be to run a `for` loop. And for the hybrid integration, you will need to program in the exceptional cases. The structure for that code might therefore start as:

```
for k=1:N
     if k==1
          simpint(k)=0;
     elseif k==2
          simpint(k)=0;
     elseif k==3
          simpint(k)=0;
     elseif k==4
          simpint(k)=0;
     elseif 0 % replace 0 with code to see if k is odd
          simpint(k)=0;
     else % if you are here, k is even and >4
          simpint(k)=0;
     end
end
```

Note the use of the variable k to make sure that the calculated integral goes into the correct spot in the vector `simpint`. You will want to use a similar structure for the derivatives, keeping in mind the exceptional cases for them as well.

12.8 Assignment

Run the `MakeData` script in MATLAB using your NET ID. This will produce a data file called `Data_ID.m` where `ID` is your NET ID. This file has two vectors in it - independent values `t` and dependent values `y`.

Next, make a copy of the `Script_skel.m` file and call it `Script_ID.m`, where ID is your NET ID. You will finish the `Script_ID` file to perform the following:

- Clear the workspace.
- Load your data file.
- Calculate an array of the 2-point forward first derivatives for each time and call this array `dydt2F`. Be sure to include any values that must be calculated by other means.
- Calculate an array of the 2-point backward first derivatives for each time and call this array `dydt2B`. Be sure to include any values that must be calculated by other means.
- Calculate an array of the 3-point centered first derivatives for each time and call this array `dydt3C`.
- Calculate an array of the 3-point forward second derivatives for each time and call this array `d2ydt23C`.
- Calculate an array of the accumulated integral of y using the Trapezoidal Rule and call this array `trapint`. Assume that the value of the integral at the first time is 0.
- Calculate an array of the accumulated integral of y using the Simpson's Hybrid Rule and call this array `simpint`. Assume that the value of the integral at the first time is 0.

When you are ready to get check your program, simply type `RunCheckerCode` in MATLAB. The code will tell you what your current grade is as well as indicate where there are incorrect values. Note in the skeleton that the other variables are set to 0 so that the checker program can run even if a particular vector has not been completed. The checker program will break if the solution data is in the wrong directory, if the solution data is in the wrong file, if permissions are not set correctly, if any of the variables do not exist, or if any of the variables are the wrong size (i.e. not of length 1 or N).

12.9 Important Policy Items

Note - this is an **individual assignment**. You can of course ask the TAs for help, but not each other. As with all assignments, failure to comply with the Honor Code will lead to a severely reduced grade and possible judicial action, not to mention that working through this problem *yourself* will lead to better success on Test III while cheating your way through life is a horrible idea in general, and specifically will leave you under-prepared for the test.