Queen's University | Faculty of Engineering and Applied Science

Lab 3: Numerical Integration

The objective of this week's laboratory is to give you some experience using numerical integration techniques including the Trapezoidal Rule and Simpson's 1/3 Rule.

Before You Start

MATLAB can only use code that is saved in its 'Current Folder'. To set the Current Folder in MATLAB, press the 'Browse for Folder' button ( ). To keep your code organized, it is strongly suggested that you **create a new folder for each lab**. Create a folder for Laboratory 1, and select it as your Current Folder. Save all files to this folder for this lab, and make sure you upload the correct files to onQ after finishing the lab. Each lab should be saved in its own folder in future weeks to ensure the proper files are submitted.

Part A – Due at end of the lab period

In the ENPH 253 Magnetic Hysteresis lab you will be asked calculate the area enclosed by a hysteresis loop based on experimental measurements. This is an ideal application for the Trapezoidal Rule.

Task A1

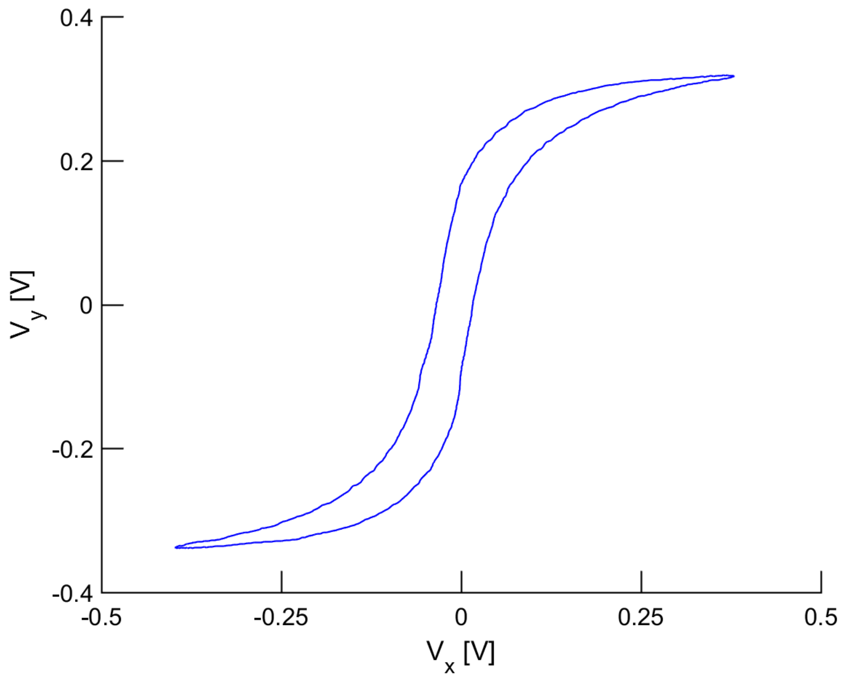
Before using the Trapezoidal Rule, plotting some hysteresis data will give you a good visualization tool as you move forward.

Create a function `plotHysteresisData` with the following inputs/outputs in the following order:

INPUTS	
`voltageDataFilename`	Filename, hysteresis voltage data

OUTPUTS	
`figure(1)`	Plot of the hysteresis data, y-component of the voltage against the x-component, similar to the plot shown below

Assume the filetype is '.csv' and that the first column in the file contains the time data, the second column contains the x-component data and the third column contains the y-component data. The data in the spreadsheet file can be read into Matlab using the function `csvread`. Use the test file **test_plotHysteresisData.m** from onQ to test your function.



Task A2

To find the area within the hysteresis loop, you will write the function `integrateTrapRule` with the following inputs/outputs in the following order:

INPUTS	
`x`	1D array of x data
`y`	1D array of y data

OUTPUTS	
`integral`	Integral calculated using the Trapezoid Rule

The Trapezoid Rule is used for this data instead of Simpson's Rule because the x point (Voltage (x)) data is not evenly spaced. The data you will use for testing is the same as for Task A1, where the data is ordered so that the data points start at the lower left corner and progress clock-wise along the loop. This means that as you integrate the data, for the top part of the loop the integral is evaluated as usual, but for the bottom part of the loop, the integral is being evaluated backwards, and so will be negative. You'll note that to integrate the area in the loop, this fact is beneficial.

This should be done without using `for` or `while` loops, and should only take one line of code.

Use the test file **test_integrateTrapRule.m** from onQ to test your function. Also make sure you download the **checkEqual.m** file!

Task A3

Write the function `integrateSimpsonRule1D` with the following inputs/outputs in the following order:

INPUTS	
`f`	Inline function of one independent variable, integrand
`xL`	Lower integration limit
`xU`	Upper integration limit
`xN`	Number of intervals

OUTPUTS	
`integral`	Integral calculated using Simpson's Rule

This should be done without using `for` or `while` loops.

Use the test file **test_ integrateSimpsonRule1D.m** from onQ to test your function.

Task A4

Write another function `integrateSimpsonRule2D` with the following inputs/outputs in the following order:

INPUTS	
`f`	Inline function of two independent variables, integrand
`xL`	Lower integration limit in first independent variable
`xU`	Upper integration limit for first independent variable
`xN`	Number of intervals for first independent variable
`yL`	Lower integration limit for second independent variable
`yU`	Upper integration limit for second independent variable
`yN`	Number of intervals for second independent variable

OUTPUTS	
`integral`	Integral calculated using Simpson's Rule

This should be done by vectorizing your code from Task A3. Use the test file **test_ integrateSimpsonRule2D.m** files from onQ to test your function.

Part A: Submission List

When complete, please submit the following files to the **Lab 3: Part A** dropbox onQ for marking:

- plotHysteresisData.m
- integrateTrapRule.m
- integrateSimpsonRule1D.m
- integrateSimpsonRule2D.m
- Any other functions you wrote that the above files use

Part B: Due Sunday @ 9PM

Task B1: Error Analysis for the ENPH 253 Impedance Experiment

You wish to calculate the force between two cylindrical magnets as part of the design of a magnetic levitation system. As a first step, you will use the following expression, which gives the magnetic field at location (*x1, y1, z1*) due to a cylindrical magnet centered on the z-axis with its base in the x-y plane.

$$\vec{B}(x1, y1, z1) = \frac{\mu_0}{4\pi} \iint_{0\,0}^{D\,2\pi} \frac{\vec{K} \times \vec{r}}{|r|^3} R d\phi dz \quad (1)$$

The magnet has a thickness *D* in the z-direction, radius *R*, and uniform magnetization parallel to the z-axis. The magnetization produces a uniform bound surface current $\vec{K}$ on the outside surface of the cylinder in the $\hat{\phi}$ direction. It is this bound surface current which is responsible for the magnetic field produced by the magnet. The above equation for the magnetic field (Biot-Savart's Law) is written using Griffith's notation where $\vec{r}$ is the separation vector and points from where the current is located to where we are evaluating the field:

$$\vec{r} = \big(x1 - Rcos(\phi)\big)\hat{x} + \big(y1 - Rsin(\phi)\big)\hat{y} + (z1 - z)\hat{z}$$

Similarly, the surface current density can be written as:

$$\vec{K} = K(-sin(\phi)\,\hat{x} + \cos(\phi)\,\hat{y}\,)$$

By expressing the separation vector and surface current density using unit vectors in Cartesian coordinates, you can break the vector integral into three scalar integrals, one for each component of the magnetic field. For example, the x-component of the magnetic field (B_x) can be expressed as a ordinary double integral of a scalar function *fnToIntegrateX*(*x1, y1, z1, ϕ, z*) :

$$B_x(x1, y1, z1) = \int_0^D \int_0^{2\pi} fnToIntegrateX(x1, y1, z1, \phi, z)\, d\phi dz$$

where

$$fnToIntegrateX(x1, y1, z1, \phi, z) = \frac{\mu_0 R}{4\pi |r|^3} \big(\vec{K} \times \vec{r}\big) \cdot \hat{x}$$

Don't try to use MATLAB's `cross` function for calculating the cross product. Instead, just write out the expression for each component of the cross product. For example, the x-component of $\big(\vec{K} \times \vec{r}\big)$ is:

$$\big(\vec{K} \times \vec{r}\big) \cdot \hat{x} = k_y r_z - k_z r_y$$

Where:

$$k_y = Kcos(\phi);\ k_z = 0$$
$$r_z = z1 - z;\ r_y = y1 - Rsin(\phi)$$

We can similarly use the functions:

$$fnToIntegrateY(x1, y1, z1, \phi, z) = \frac{\mu_0 R}{4\pi|r|^3}\left(\vec{K} \times \vec{r}\right) \cdot \hat{y}$$
$$fnToIntegrateZ(x1, y1, z1, \phi, z) = \frac{\mu_0 R}{4\pi|r|^3}\left(\vec{K} \times \vec{r}\right) \cdot \hat{z}$$

Where we also know that:

$$k_x = -Ksin(\phi)$$
$$r_x = x1 - Rcos(\phi)$$
$$|r| = \sqrt{r_x^2 + r_y^2 + r_z^2}$$

To compute the y and z components of the magnetic field (*B_x* and *B_y*).

You might be a little frightened at this point! And that is perfectly normal! Solving complex problems are exactly that, complex. In the following tasks, you'll break the problem into smaller easier problems, and by testing each section separately and then combining them, you can solve this larger task.

Task B1

To give you some exposure to global variables, first write a function `setupGlobalVariables`, with no inputs or outputs, that declares the variables in the table below as global and sets their values according to the table below:

Variable	MATLAB name	Values
Magnet Radius, *R*	`R`	0.025 [m]
Magnet Thickness, *D*	`D`	0.05 [m]
Surface Current, *K*	`K`	0.95×10^6 [A/m]
Magnetic Permeability, μ_0	`u_0`	$4\pi \times 10^{-7}$ [N/A^2]
Number of Regions, *n*	`n`	10

The value of a global variable is shared across separate functions and the base workspace. That is, a change in the value of a global variable is visible to all other functions that declare the same variable as global. You will need to use the `global` command in both `setupGlobalVariables` and any other functions that use the global variables in the above table. **Refer to the help page for `global`.**

Task B2

Before you can calculate the forces at play, you'll first have to be able to calculate the magnetic field *B*. To do so, write three functions: `fnToIntegrateX`, `fnToIntegrateY`, `fnToIntegrateZ` each with the following inputs/outputs in the following order:

INPUTS	
`x1`	x-coordinate at point of measurement, or at location that you want to calculate the magnetic field *B* at
`y1`	y-coordinate at point of measurement
`z1`	z-component at point of measurement
`phi`	Angular position along the magnet at source point
`z`	Height along the magnet at source point

OUTPUTS	
`val`	Value of the integrand function used to find the magnetic field *B*

It is critical that these functions are fully vectorized in order to be integrated!

To test your functions, check their outputs for the coordinates:

```
x1 = 0.01;
y1 = 0.02;
z1 = 0.03;
phi = pi/6;
z = 0.05;
```

You should get the following values:

Function	**Expected value**
`fnToIntegrateX`	-2.8560
`fnToIntegrateY`	-1.6489
`fnToIntegrateZ`	1.0454

Task B3

Now that you have the function to integrate for Eqn. (2) (`fnToIntegrateX`), you can use the 2D Simpson's Rule function from Task A4 to calculate the value of B_x. The values for B_y and B_z can be found in a similar way. Create a function `B` with the following inputs/outputs in the following order:

INPUTS	
`x1`	x-coordinate at point of measurement
`y1`	y-coordinate at point of measurement
`z1`	z-component at point of measurement

OUTPUTS	
`Bx`	x-component of magnetic field at point of measurement
`By`	y-component of magnetic field at point of measurement
`Bz`	z-component of magnetic field at point of measurement

Note that to find each component of the *B* field, you have to integrate over ϕ and *z,* and so you will have to take your `fnToIntegrate` functions and use inline functions in MATLAB to turn them from 5 variable functions to 2 variable functions. To get you started, for the x-component I used the following inline function definition:

```
fnX = @(phi, z) fnToIntegrateX(x1, y1, z1, phi, z);
```

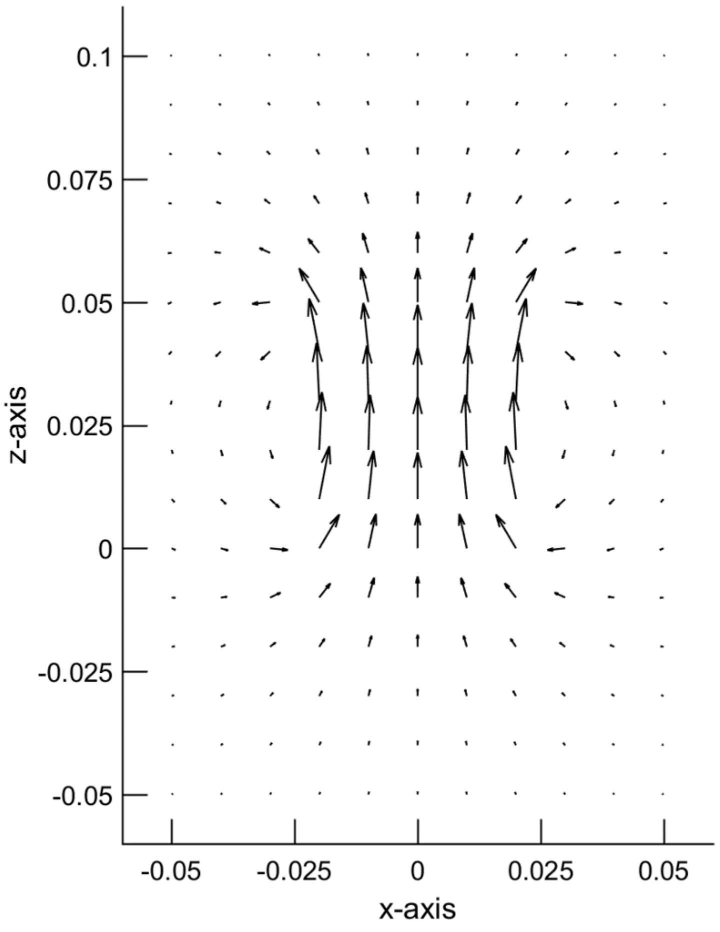
To test your function, check its outputs for the same coordinate as Task B2:

```
x1 = 0.01;
y1 = 0.02;
z1 = 0.03;
```

You should get the following values:

Output	Expected value
`Bx`	1.9844×10^{-2}
`By`	3.9659×10^{-2}
`Bz`	8.2693×10^{-1}

Another useful way to evaluate whether or not a function is working correctly is to visualize the data it is outputting. Another file from onQ, **test_B_byPlotting.m**, allows you to plot multiple values of *B*. to see what the magnetic field looks like. If everything is working correctly, the plot should look similar to the below.



Task B4

Now that you can calculate the magnetic field, *B,* from a cylindrical magnetic at any point, you can calculate the force between two magnets separated by some distance, *d.* Assume that the first magnets sits with its base in the x-y plane and is centered on the z-axis, and that a second identical cylindrical magnet is placed directly above the first so that the bottom face of the top magnet is a distance *d* above the top face of the bottom magnet (e.g. the air gap between the magnets is *d*). You can use the Lorentz force law to calculate the force exerted on the second magnet. The expression is given below:

$$\vec{F}(d) = \int_{D+d}^{2D+d} \int_{0}^{2\pi} (\vec{K} \times \vec{B}) R \, d\phi dz \tag{3}$$

Here, *B* is the magnetic field produced by the first magnet (bottom) on the surface of the second magnet, and *K* is the current density on the surface of the second magnet (top). Note the limits of *z* go from *D+d* to *2D+d*.

The expression for *K* on the surface of the top magnet is:

$$\vec{K} = -K(-sin(\phi)\,\hat{x} + \cos(\phi)\,\hat{y}\,) \tag{4}$$

Note: The sign of *K* is flipped compared to the first magnet. If you don't flip one of the magnets, then they will attract each other, which wouldn't be very useful for a magnetic levitation system.

In order to evaluate Eqn. (3), you can use the same technique that was used to evaluate Eqn. (1). First, Eqn. (3) can split into its components, such that:

$$F_x(d) = \int_{D+d}^{2D+d} \int_0^{2\pi} (\vec{K} \times \vec{B}) R \cdot \hat{x}\ d\phi dz$$

The term within the integral can then be replaced with a function:

$$F_x(d) = \int_{D+d}^{2D+d} \int_0^{2\pi} fn2ToIntegrateX(\phi, z)\, d\phi dz \tag{5}$$

Where:

$$fn2ToIntegrateX(\phi, z) = (\vec{K} \times \vec{B}) R \cdot \hat{x} = R(k_y B_z - k_z B_y)$$

From Eqn. (4) we know that:

$$k_x = Ksin(\phi); k_y = -Kcos(\phi); k_z = 0$$

And the values for B_x, B_y, B_z can be found using the `B` function you made in Task B3.

To break this problem down, start by making the integration functions (`fn2ToIntegrateX`, `fn2ToIntegrateY`, `fn2ToIntegrateZ`) for this problem, with the following inputs/outputs in the following order:

INPUTS	
`phi`	Angular position along the second magnet
`z`	Height along the second magnet

OUTPUTS	
`val`	Value of the integrand function used to find the force on the second magnet *F*

It is <u>critical</u> that each of these functions be vectorized, as matrices of ϕ and *z* values will be passed into them. The `B` function you wrote for Task B3 <u>is not vectorized</u> and so you will have to use `for` loops to evaluate `B` at the given matrices of ϕ and *z* values. Test your functions with the test files **test_fn2ToIntegrateX.m, test_fn2ToIntegrateY.m**, and **test_fn2ToIntegrateZ.m** available on onQ.

Task B5

Now that you have functions for the integrals in the force component calculations, you simply need to evaluate the integrals according to Eqn. (5) to find the force. Write a function `F` with the following inputs/outputs in the following order:

INPUTS	
`d`	Separation between first and second magnets along the z-axis

OUTPUTS	
`Fx`	x-component of force exerted on the second magnet
`Fy`	y-component of force exerted on the second magnet
`Fz`	z-component of force exerted on the second magnet

The functions that will be given your `integrateSimpsonRule2D` function are the same that you wrote in Task B4, for example I use the following command to define `fn2ToIntegrateX` as an inline function:

```
fnX = @fn2ToIntegrateX;
```

Use the test file **test_F.m** from onQ to test your function. If everything goes correctly, you should find that for the test case, the F_z would support a 60 kg mass! The bound surface current density and magnet dimensions specified in the tests corresponds to the magnetization that would be typical for a neodymium magnet.

Part B: Submission List

When complete, please submit the following files to onQ for marking:

- setupGlobalVariables.m
- fnToIntegrateX.m
- fnToIntegrateY.m
- fnToIntegrateZ.m
- B.m
- fn2ToIntegrateX.m
- fn2ToIntegrateY.m
- fn2ToIntegrateZ.m
- F.m
- Any other functions you wrote that the above files use