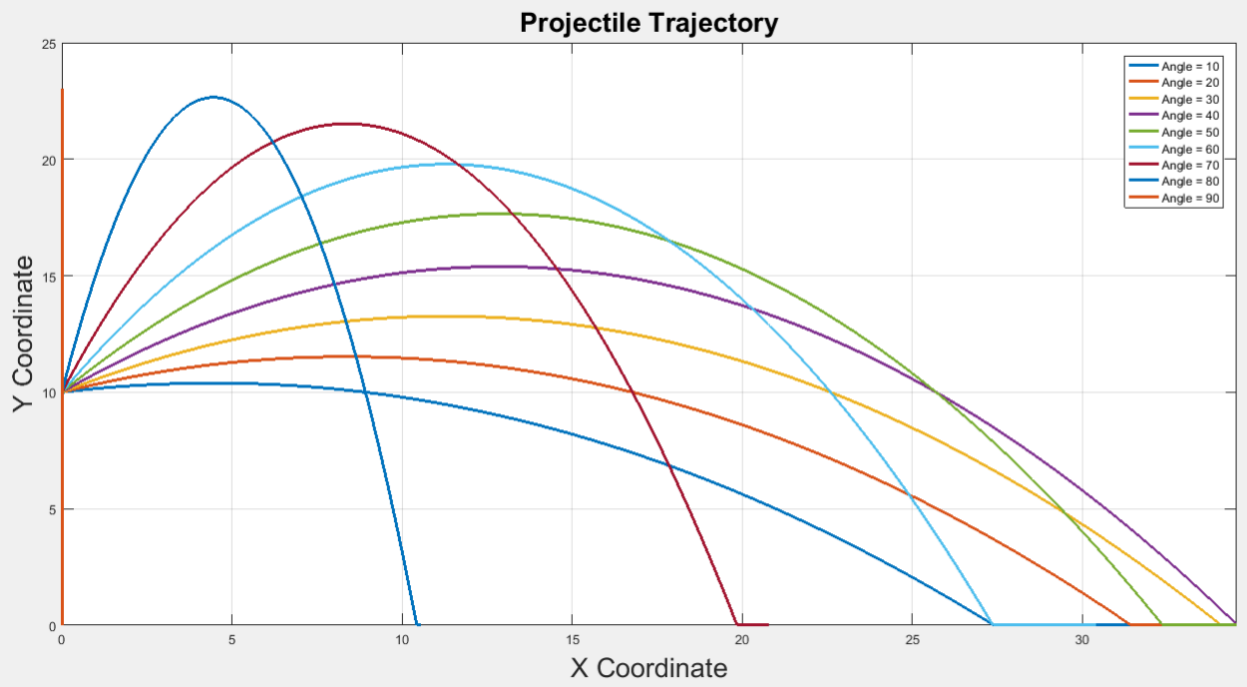
UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING
First Year Program – Core 8 and TrackOne

FIRST YEAR PROGRAM
ENGINEERING PROBLEM SOLVING LABS

MAT188: Laboratory #6
Approximating Derivatives



APPROXIMATING DERIVATIVES

In this lab, you will approximate derivatives of functions numerically, and compare the approximation with the exact symbolic derivative. You will also use these approaches to solve scientific and engineering problems.

Learning Outcomes

By the end of this lab students will…

1) Synthesize and apply an existing knowledge of finite differences and derivatives in the context of numeric computation
2) Engage in critical thinking to frame, scope, develop, and numerically-compute an authentic engineering problem by developing and defending meaningful assumptions and calculations

Preparation *(Required to do* ***before*** *you come to the laboratory session)*

1. Read through this lab document.

Course Connection (MAT186: Calculus I)

As you read through this lab document, work along with the module and enter each example command in the MATLAB command window to see the result.

You have learned about the *derivative* (rate of change) of a function with respect to an independent variable using the definition of a limit. In this exercise, you will approximate the derivative of a function using the method of *finite differences*.

First, you'll need to understand how the `diff` command works. If given a numeric vector input, `diff` returns a vector of the differences between each pair of consecutive numbers. For example:

```
>> x=1:10
>> y=x.^2
>> diff(y)
```

If given a symbolic function input, `diff` returns the derivative as a symbolic function. For example:

```
>> syms x
>> f=tan(x.^2+3*x)
>> diff(f)
```

Estimating a Derivative from First Principles

The instantaneous rate of change of a function can be expressed as an exact derivative as:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h)-f(x)}{h}$$

When using numeric computation tools we work with discrete values of a function, so we cannot take an exact limit of this quotient. Instead, we can only estimate this limit and derivative through a *finite difference* calculation over small intervals:

$$\frac{df(x)}{dx} \approx \frac{\Delta f(x)}{\Delta x} = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

When $f(x)$ is differentiable at x_1, this approximation becomes more accurate as we make these differences smaller, as x_2 becomes closer to x_1.

For example, consider the function $\sin(x)$. We can represent this function at discrete intervals using `linspace` to generate equidistant points with `n` used to define the precision:

```
>> n = 101;
>> x = linspace(0,2*pi,n);
>> y = sin(x);
```

To approximate the first derivative, use the `diff` command to obtain a vector containing the difference of a function evaluated at adjacent points, and use element-wise division:

```
>> y_prime = diff(y)./diff(x)
```

This is the same as calculating $\frac{\Delta y}{\Delta x}$ for each interval as an approximation. This value is closest to the true value of the derivative at the midpoint of each interval.

What is the size of `y_prime` ***(use*** `whos` ***or the Workspace panel to find out)? Can you explain why?***

Now plot the derivative:

```
>> delta_x=x(2)-x(1);
>> xd=linspace(x(1)+delta_x/2, x(end)-delta_x/2, n-1);
>> plot(xd,y_prime);
>> grid on
```

Considering the size of `y_prime`***, why did we define*** `xd` ***in the way that we did? Observe how this approach plots the values of the "derivative of y" at the midpoint of each interval of*** `x`**.**

Now let's add the exact symbolic derivative to the plot to compare it with the approximation.

```
>> hold on
>> syms sym_x
>> sym_y=sin(sym_x)
>> sym_y_prime=diff(sym_y)
>> fplot(sym_y_prime)
```

By zooming into the plot, you can observe the small differences that exist between the numerical and symbolic approaches.

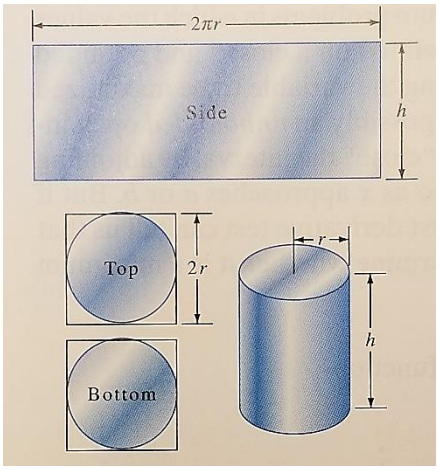
You might be wondering why we computed the derivative numerically first since doing it symbolically was much easier. If you know the expression of a function, doing it symbolically is easier. But often, you will have a set of experimental data and no expression to model it, so it is impossible to do symbolically. In those cases, you must do it numerically to approximate the rate of change between data points. Here we generated the set of numeric data points as an example, but normally these data points would come from observation or experimentation.

Engineering Exercise: PLEASE SELECT *ONE* OF THE TWO EXERCISES BELOW for submission. You may do both – this is excellent practice -- but please submit only one.

1) Lee is participating in a pumpkin-throwing contest where participants launch pumpkins for maximum distance. Equipped with Matlab and a computer, and as Lee's only U of T-Engineering friends, we would like to engineer a well thought-out computational strategy, so he wins. All we know is that there is no resistance due to wind, and that there will be a number of pumpkins, weighing no more than 10kg each, and there is no powered-machinery used. Using what you know about Matlab (and any other engineering courses) so far, demonstrate your mastery by mathematically-optimizing Lee's chances of throwing pumpkins the farthest. It is strongly recommended that you use plots, approximations, derivatives, and a variety of tools to frame, understand, engineer, and evaluate the strategy. In your plot(s) create a legend to label each line, and approximate clearly. Some of this work will require you to make reasonable assumptions, which you need to identify and articulate.

   For submission:
   - 1 PDF page showing your work/calculations (some hand-sketches are fine), etc.
   - Recommended plots:
     - Position/Time; Velocity/Time; Acceleration/Time, others?
       i. (numerically AND/OR symbolically)
   - Write: explain what your calculations/plots actually mean. Please also explain your assumptions and why they are reasonable, articulate your thought process, and any other considerations an engineer should take into account for this problem.

2) Emma the Engineer is coming home after a night of trick-or-treating with her friends, and they pool together their candies and determine that they need to make a cylindrical can that holds a volume of 2 Litres (equal to 2000 cm^3) to store the candy. They have access to sheets of metal; the top and bottom circle can be cut from square sheets, whereas the side of the can made is by bending a sheet, like a tube. The top and bottom circles are identical in size. What radius, r, and height, h, of the can will minimize the total amount of material required for the rectangle and the two squares?
   Area = $8r^2 + 2\pi rh$
   Volume = $\pi r^2 h$ = 2000 cm^3 = 0.002 m^3



CONTINUED…

For submission:

- 1 PDF page showing your work/calculations, etc
- Must include a figure of your function, with Area as the vertical axis and r as the horizontal axis;
- Must clearly communicate your thought process, the justification for the measurements you suggest, and discuss any other considerations an engineer must take into account for this problem.

Deadlines:

Monday Practical Sections (PRA 113, 114, 116): **Due by Saturday October 27th by 11:59pm**

All other Practical Sections, unaffected by Thanksgiving Monday holiday: **Due by Saturday October 20th, by 11:59pm.**