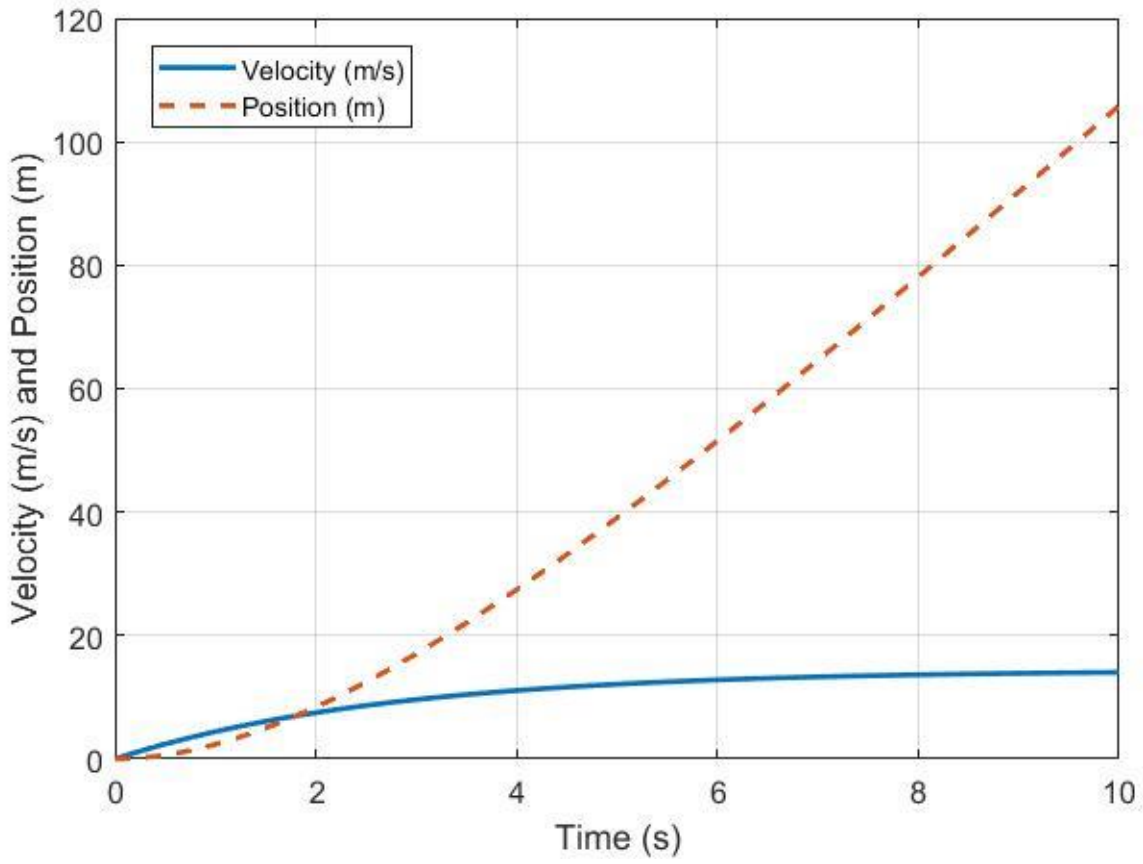
UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING
First Year Program – Core 8 and TrackOne

FIRST YEAR PROGRAM
ENGINEERING PROBLEM SOLVING LABS

MAT188: Laboratory #4
Symbolic Computation



SYMBOLIC COMPUTATION

In this lab, you will be introduced to the symbolic computation capabilities in MATLAB, which is a useful complement to the numeric computation techniques that we have been working with in the first two labs.

Learning Outcomes

By the end of this lab you should…

1) Have the ability to do basic symbolic calculations including equation solving, differentiation, and integration, and
2) Understand how to apply the symbolic calculations to determine and plot position and velocity of a moving object.

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

1) Read through the entire lab handout.
2) Review questions **P5.1-9** and **P5.2-75** from your MAT186: Calculus I textbook (see below).

Related Reference Materials *(Not required, but may be a helpful resource)*

Video demonstration, *Symbolic Math Toolbox Overview (2:12)*
https://www.mathworks.com/videos/symbolic-math-toolbox-overview-61212.html

MATLAB Skills and Knowledge: Symbolic Computation

Symbolic Computation - Introduction

Up until now, you have been using MATLAB for ***numeric computation***, in which values are stored and approximated using decimal numbers in the computer's memory. There is another type of computation called ***symbolic computation***, in which data is stored as exact expressions rather than numerical values. This type of computation more closely aligns with analytic approaches to solving problems.

From high school math, you know that $\sin \pi = 0$. ***If you evaluate this in MATLAB, what value is produced?***

```
>> sin(pi)
```

This happens because the value of `pi` has a small rounding error when being stored as a decimal. You can represent the value of π exactly as a symbolic value using the `sym` function.

```
>> theta=sym(pi)
```

What do you notice about the value of `theta`? How is it represented in the command window and workspace panel?

```
>> sin(theta)
```

What is the result?

Numeric computation leads to small approximation errors, but symbolic computation can only be used if you know the exact expression for a value, such as a function. Remember that numeric computation deals with numbers (and thus approximations), while symbolic computation deals with expressions (and thus can be considered to be "exact").

For example, you would have to use numeric computation if you had a set of experimental data but did not have a general expression or function to represent it.

Normally, when you define a variable you give it an exact numerical value. Instead, you can define a ***symbolic variable*** without a specific value using `syms`.

```
>> syms a
>> syms b c   % Can define multiple variables on the same line
```

Now, you can define variables in terms of symbolic variables. This is useful for representing functions as general expressions.

```
>> f(a)=a^2+3*a
```

How does MATLAB represent the value of `f`? Here we have defined `f(a)` to be a `symfun` which allows us to evaluate the value of this function at any point, such as `f(-1)` or `f(10)`. Try this.

If you have a known expression, symbolic computation also makes it easier to plot functions. You can use `fplot` to plot a function using a symbolic expression. For example, to plot $f(a) = a^2 + 3a$:

```
>> fplot(f);grid on
```

Notice how MATLAB automatically sets the limits of the axes for this figure. Try adjusting these using the command:

```
>> axis([-10 10 -10 100])
```

Observe how MATLAB fills in the full figure even though we have not explicitly defined the function over this range, as we have previously using the numeric computation approach (i.e., using the `linspace` command and then calculating the values of the function of that domain).

You can also control the limits of the `x` or `y` axes through the commands `xlim` or `ylim`. As an example, try:

```
>> ylim([-10 60])
```

and this can be "reset" to the default limits using:

```
>> ylim('auto')
```

Finally, you can also set the limits of the x-axis directly in `fplot`:

```
>> fplot(f(a),[-10 10])
```

and MATLAB will automatically adjust the limits for the y-axis.

Symbolic Computation – Solving Equations

You can use the symbolic computation tools in MATLAB to carry out a wide variety of analytic calculations, including solution solving, differentiation, and integration.

Consider the drone flight data function that you worked with in Lab #2:

$$h(t) = -2(t-2)^3 + 3(t-2) + 1$$

To find values of time for which the drone achieves its proper hover height we can use the following set of commands:

```
>> syms t
>> equ_sol=solve(-2*(t-2)^3+3*(t-2)+1==0)
>> eval(equ_sol)
```

Because the `solve` function returns the solutions as exact mixed fractions, the `eval` function is used to determine the numeric solutions (which are *approximate* solutions).

Symbolic Computation – Differentiation and Integration

Now let us use MATLAB to evaluate simple symbolic derivatives and integrals through the `diff` and `int` commands.

Exercise 1 (practice in the lab):
Use the `diff` and `fplot` commands to create a figure that includes both the drone flight position, `h(t)` and its velocity, $v(t) = \frac{dh(t)}{dt}$ for the range, $0 \le t \le 4$ s.

Exercise 2 (practice in the lab):
In MAT186, you have recently been introduced to the basic idea of finding the area under a curve through the use of Reimann Sums or definite integration. Use MATLAB to solve these problems:

a) **P5.1-9 (MAT186):** Find the area under the curve $v(t) = 3t^2 + 1$ between $t = 0$ and $t = 4$. *How does this result compare to your Reimann sum calculations with 4 and 8 subintervals? You may find the command* `rsums` *interesting to try.*

b) **P5.2-75 (MAT186)**: Find the area under the curve $f(x) = \sqrt{24 - 2x - x^2}$ between $x = -6$ and $x = 4$. *How does this result compare to your analytic calculations based on geometry?*

Exercise 3 (for submission): *A Model of Competitive Runners*[1]
One way to represent the velocity for a short-distance runner is to use the function $v(t) = A(1 - e^{-t/b})$, where A and b are positive constants and $v(t)$ is measured in m/s. Using $A = 14.4$ m/s and $b = 2.72$, use `fplot` to visualize the velocity and position, $s(t)$, function over the range $0 \le t \le 15$ s. You should assume that the runner starts at $s(0) = 0$ m. *How long does it take this runner to reach 100 m? What was their instantaneous speed at that time in km/h?*

Exercise 4 (optional): *Bungee Jumper*[2]
A women attached to a bungee cord jumps from a bridge that is 30 m above a river. Her height in meters, above the river t seconds after the jump is $y(t) = 15(1 + e^{-t}\cos t)$, for $t \ge 0$ s.

a) Plot her velocity and determine the velocity values at $t = 1$ s and $t = 3$ s.
b) Determine over what time periods she is moving downwards and upwards.
c) What is the maximum upward velocity?
d) How close does she come to the surface of the river?

If you would like to explore further examples of how to use the Symbolic Toolbox in MATLAB click here.

[1] Adapted from P6.9-40, W. Briggs, L. Cochran, and B. Gillett, *Calculus for Scientists and Engineers, Early Trancendentals*, 2013, pg. 480
[2] Taken from P3.5-51, W. Briggs, L. Cochran, and B. Gillett, *Calculus for Scientists and Engineers, Early Trancendentals*, 2013, pg. 181

Submission Guideline:

As part of this lab, each student is required to submit the solution to **Exercise 3** in the form of:

A one-page PDF file containing: 1) properly labeled position and velocity plots 2) descriptive answers (i.e. numbers and how you got them) to the two questions asked in that exercise.

You will submit these two items through the MAT188 PRA Portal page:
Winter-2018-MAT188H1-S-PRA0101: APP. LINEAR ALGEBRA

Ideally, you will submit this assignment by the end of your Lab #4 session, but at the very latest you can submit this by Wednesday, February 7th at 11:59PM.