ME 2004 Homework Exercises

Compiled by Jaisohn Kim (jais0hn at vt dot edu).

Version: Spring 2024

Here are the homework problems for ME 2004, *Engineering Analysis Using Numerical Methods.* Problems are organized by topic.

Instructions
You may work in a team of up to **3 students in your Recitation section (same CRN).** Everyone will receive the same grade. Please ensure everyone's name is clearly visible on the front page.

Answers to analysis questions can be written as a MATLAB comment. You may also use `fprintf(), disp(),` etc. to print your answers. Create a published PDF of each problem when finished.

Remember to conform to the *Homework Formatting Guidelines* and *Plot Formatting Guidelines* on Canvas.

Submission
Submit ONE PDF to Canvas. You will need to use a PDF merger. If you work in a team, only one member needs to submit.

Terminology Note
Visual learning is emphasized throughout the course. Most homework problems require either a *plot* or a *sketch*. A *plot* refers to a clean, MATLAB-generated image. A *sketch* is a hand-drawn image which primarily emphasizes qualitative behavior due to its rough, imperfect nature. They are not equivalent or interchangeable. Both a plot and a sketch must adhere to the course's Plot Formatting Guidelines (located on Canvas). The author has carefully chosen the diction in each problem in accordance with this distinction.

Table of Contents

1. MATLAB Fundamentals (MF)

Objective

The purpose of these problems is to get you familiarized with MATLAB and some useful built-in functions you will likely use throughout the course.

Statement on Built-In Functions

Some students may have prior MATLAB experience. In order to get everyone on the same page, we are imposing usage restrictions on built-in functions. The ONLY built-in functions that may be used for the *MATLAB Fundamentals* problems are:

- Functions specifically mentioned in the problem statement (such as `min()` in Problem MF-4)
- Any plot-specific functions: `plot()`, `fplot()`, `subplot()`, `xlabel()`, `ylabel()`, `title()`, `legend()`, `sgtitle()`, etc.

You may NOT use `if` statements, `for` loops, etc.! Every problem in this set can be solved without them. Those may be used from the *Advanced MATLAB* problems onward.

It is in your best interest to always read the MATLAB documentation for built-in functions, even if you think you know how to use them.

MF-1: Terminology

MATLAB stands for MATrix LABoratory. Storing, manipulating, and operating on data to achieve results is the identity of the entire language.

We typically use the words *scalar, vector,* and *matrix* to qualitatively communicate a variable's dimensions. Classify each quantity as a scalar, row vector, column vector, or matrix. Also give the variable's dimensions (number of rows and columns).

Do entirely by hand, although you may verify your answers in MATLAB via the `size()` command once finished.

Case	**Quantity**	**Type**	**# Rows**	**# Columns**
example	$\begin{bmatrix}1 & 0\\ 2 & 2\end{bmatrix}$	Matrix	2	2
A	$\begin{bmatrix}4 & 8\end{bmatrix}$			
B	$\begin{bmatrix}2 & 2 & 1 & 6 & 12 & 22\end{bmatrix}$			
C	$\begin{bmatrix}3 & -2\\ 0 & 0.6\\ 4 & 0\end{bmatrix}$			
D	-15.5			
E	$\begin{bmatrix}0 & 0\end{bmatrix}$			
F	0.000001			
G	$\begin{bmatrix}3\\ 30\\ 300\end{bmatrix}$			
H	$\begin{bmatrix}0 & 0 & 1\\ 2 & -2 & 2\end{bmatrix}$			
I	$\begin{bmatrix}-1 & 10\\ 0.5 & \pi\end{bmatrix}$			
J	$\begin{bmatrix}-1 & -2 & -3\\ 4 & 5 & 6\\ -7 & 8 & -9\end{bmatrix}$			

MF-2: Matrix Basics

Consider three quantities: $A = \begin{bmatrix} 7 & 13 \\ 11 & 6 \\ 8 & 2 \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, and $C = \begin{bmatrix} 2 & 1 \\ 4 & 4 \\ 1 & 3 \end{bmatrix}$.

a) What are the dimensions of A? What are the dimensions of B?
b) Compute $3 * A$
c) Compute $C + A$
d) Compute $C.* A$
e) Compute $A * B$
f) Identify the **illegal** operations and explain why they're illegal. Recall the apostrophe means transpose.
    a. $A * B'$
    b. $C * A$
    c. $B' * A'$
    d. $B * A'$
    e. $\frac{A}{B}$

Do entirely by hand, although you may verify your answers in MATLAB once finished.

MF-3: Hookian Springs

For a so-called *Hookian* spring, an applied force F results in a displacement x according to the equation:

$$F = kx$$

where k is the spring constant $\left(\frac{N}{m}\right)$. Despite its name, k does vary under certain circumstances; for instance, as the result of a change in temperature.

A fixed load of $17\ N$ was applied to a spring over a range of temperatures T $(°C)$, and the resulting displacements x were measured (m). The data are contained in the `ME2004_HookianSpringsData.mat` file located here.

Develop a script .m-file which:

1. Loads the `.mat` file
2. Computes k for each temperature.
3. Plots k vs. T. Refer to the course FAQ on Canvas for detailed requirements defining plot formatting (titles, labels, etc.).
4. Briefly describes how the spring constant k changes with temperature.
5. Why do you think the spring behaves this way?

MF-4: Projectile Motion

The equation for projectile motion gives the height of a projectile y (m):

$$y = (tan\theta_0)x - \frac{gx^2}{2v_0^2 cos^2\theta_0} + y_0$$

where x (m) is the horizontal position, v_0 (m/s) is the initial projectile speed, θ_0 (deg) is the initial angle, y_0 (m) is the initial height, and $g = 9.81$ (m/s^2) is the acceleration due to gravity.

a) Create an `x` vector ranging from 0 to 80 m in steps of 5 m. Also create a vector for $\theta_0 = 15°$ to $75°$ in steps of $15°$.
b) In *only one statement*, calculate the projectile's height for all cases of `x` and θ_0. Take $v_0 = 25$ m/s and $y_0 = 0$ m.
c) On one figure, plot `y` vs `x` for all five θ_0. Don't forget to add a legend (and the other plot annotations, such as labels).
d) What is y when $x = 30\ m$ and $\theta = 60°$? (The `find()` function may be useful.)
e) Suppose we only want to view positive y values on the plot. Set the lower y-limit on the plot to 0.
f) Analyze your results. Are there any launch angles which produce the same end point? Which launch angle produces the farthest range? Are your results consistent with what you learned in basic physics?

MF-5: Terminal Velocity

The terminal velocity v_t (m/s) of a skydiver is:

$$v_t = \sqrt{\frac{mg}{c_d}}$$

where m (kg) is the skydiver's mass, $g = 9.81$ (m/s^2) is the acceleration due to gravity, and c_d (kg/m) is the drag coefficient. A skydiver with $c_d = 0.3\,\frac{kg}{m}$ jumped seven times, wearing different gear every trial to artificially change her mass:

Trial	1	2	3	4	5	6	7
$m\ (kg)$	60.2	65.3	72.1	80.9	83.6	91.1	92.9

a) Compute each trial's v_t.
b) Create a 2-by-1 subplot and plot v_t vs. m on the upper subplot using magenta diamonds with a connecting line.

She then fixed her mass at $m = 60.2\ kg$ and then jumped seven more times with different drag coefficients:

Trial	1	2	3	4	5	6	7
$c_d\ \left(\frac{kg}{m}\right)$	0.25	0.28	0.33	0.41	0.47	0.55	0.70

c) Compute each trial's v_t.
d) Plot v_t vs. c_d on the lower subplot using blue squares with a connecting line.
e) Analyze your results. How do m and c_d affect v_t? Are your results consistent with basic physics and common sense?

MF-6: Average Daily Temperature

The average daily air temperature for an area over a time span t can be approximated by the following function:

$$T = T_{mean} + \left(T_{peak} - T_{mean}\right)\cos\left(\omega\left(t - t_{peak}\right)\right)$$

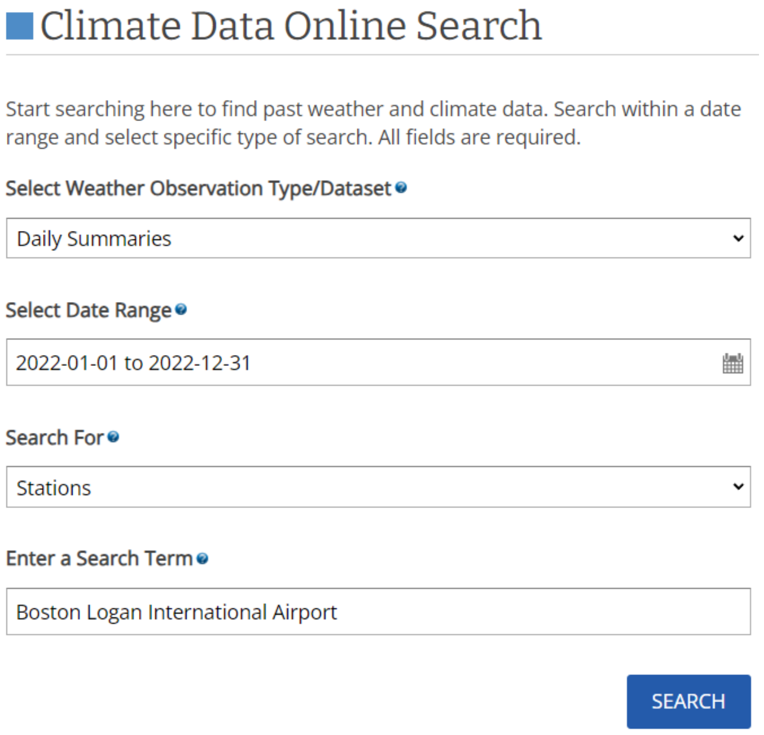
where:

- t = time span in calendar days (1-365)
- T = average daily temperature for each day in t
- T_{mean} = average annual temperature
- T_{peak} = peak average temperature
- t_{peak} = day of the peak average temperature (1-365)
- $\omega = \frac{2\pi}{365}$ = frequency of annual variation

Our friend Matt Labb, who is stationed near Boston Logan International Airport, recorded the average daily air temperature for the entire 2022 year using a homemade thermometer. He obtained $T_{mean} = 51\ °F, T_{peak} = 86\ °F$, and $t_{peak} = 203$ days. He wants to compare his data from July-August ($t = 180$ to 242) with official weather station data.

First, you need to obtain the weather data from the National Oceanic and Atmospheric Administration (NOAA).

1) Go here and fill in the fields as follows:



Climate Data Online Search

Start searching here to find past weather and climate data. Search within a date range and select specific type of search. All fields are required.

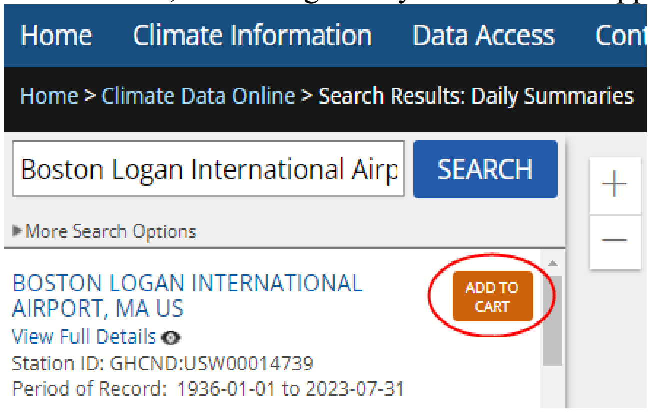
Select Weather Observation Type/Dataset: Daily Summaries

Select Date Range: 2022-01-01 to 2022-12-31

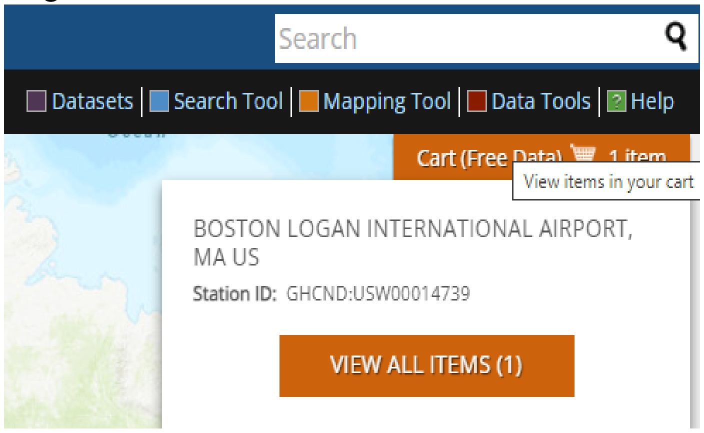
Search For: Stations

Enter a Search Term: Boston Logan International Airport

1) On the next screen, locate the Boston Logan International Airport station. Press “Add to Cart”, then navigate to your cart in the upper right corner of the screen.

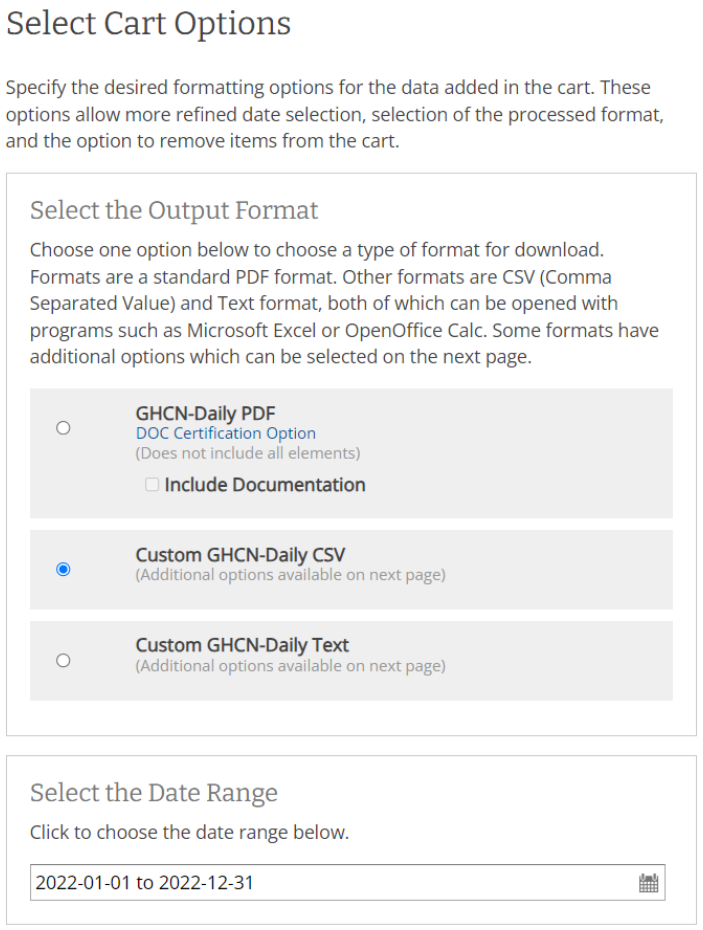

Home
Climate Information
Data Access
Cont
Home > Climate Data Online > Search Results: Daily Summaries
Boston Logan International Airp
SEARCH
More Search Options
ADD TO CART

BOSTON LOGAN INTERNATIONAL AIRPORT, MA US
View Full Details
Station ID: GHCND:USW00014739
Period of Record: 1936-01-01 to 2023-07-31



BOSTON LOGAN INTERNATIONAL AIRPORT, MA US
Station ID: GHCND:USW00014739

2) On the next screen, select the “Custom GHCN-Daily CSV” option in the “Select the Output Format” box. Ensure the date range is: 2022-01-01 to 2022-12-31. Then, press the Continue button at the bottom of the screen.



Select Cart Options

Specify the desired formatting options for the data added in the cart. These options allow more refined date selection, selection of the processed format, and the option to remove items from the cart.

Select the Output Format

Choose one option below to choose a type of format for download. Formats are a standard PDF format. Other formats are CSV (Comma Separated Value) and Text format, both of which can be opened with programs such as Microsoft Excel or OpenOffice Calc. Some formats have additional options which can be selected on the next page.

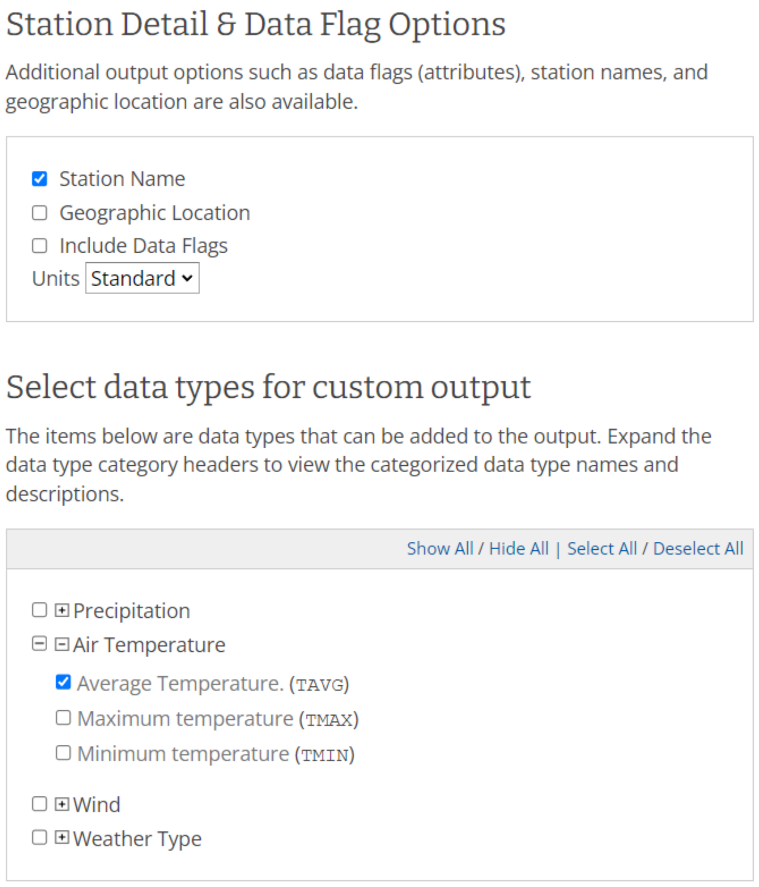
- GHCN-Daily PDF
  DOC Certification Option
  (Does not include all elements)
  Include Documentation
- [✓ Custom GHCN-Daily CSV
  (Additional options available on next page)]
- Custom GHCN-Daily Text
  (Additional options available on next page)

Select the Date Range

Click to choose the date range below.

2022-01-01 to 2022-12-31

3) Leave the "Station Name" option in the "Station Detail & Data Flag Options" section checked. Check only the "Average Temperature" button in the "Select data types for custom output" section. Press Continue.



Station Detail & Data Flag Options

Additional output options such as data flags (attributes), station names, and geographic location are also available.

- [✓] Station Name
- [ ] Geographic Location
- [ ] Include Data Flags

Units Standard

Select data types for custom output

The items below are data types that can be added to the output. Expand the data type category headers to view the categorized data type names and descriptions.

- [ ] Precipitation
- [ ] Air Temperature
  - [✓] Average Temperature. (TAVG)
  - [ ] Maximum temperature (TMAX)
  - [ ] Minimum temperature (TMIN)
- [ ] Wind
- [ ] Weather Type

4) Enter your email address and press Submit Order. A CSV containing the air temperature data will be emailed to you in a few minutes.

Now that you have NOAA's official air temperature data, develop a script .m-file which:

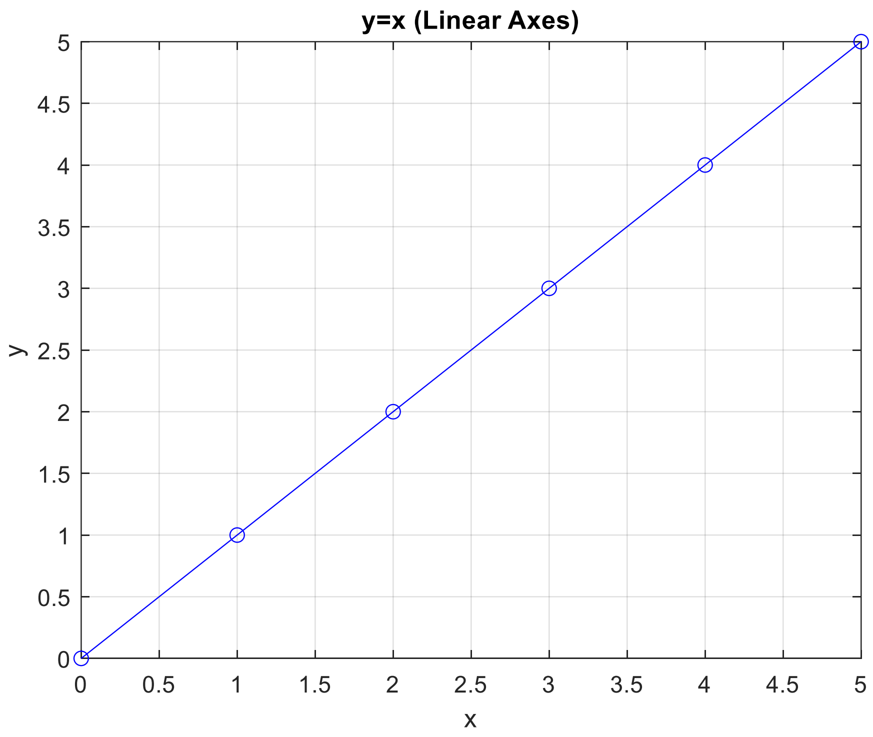
1. Reads the CSV data. The `readmatrix()` function may be useful.
2. Computes T_{mean}, T_{peak}, and t_{peak}. The `max()` function may be useful.
3. Computes T from July-August ($t = 180$ to 242).
4. Computes T from July-August using Matt Labb's T_{mean}, T_{peak}, and t_{peak} values.
5. Plots Matt Labb's T and NOAA's T on a single figure. Please remember to properly annotate your plots (axis labels, legends, etc.). Play around with the marker sizes, colors, etc.
6. Describes the discrepancies, if any, between NOAA's data and Matt Labb's data.

Remember to generalize your code to accept any values of T_{mean}, T_{peak}, and t (no hard coding!)

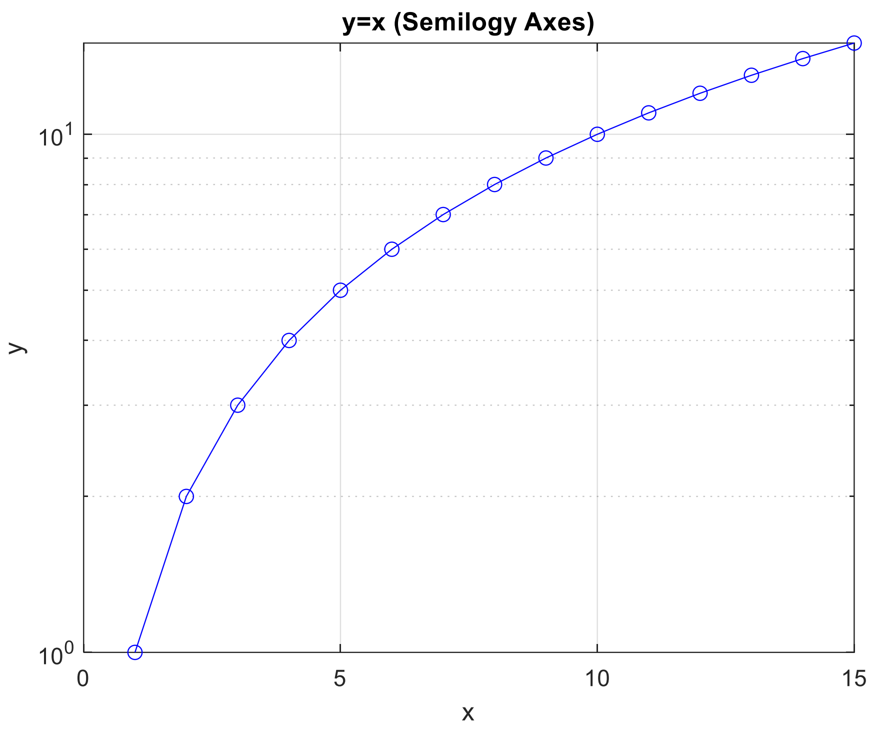
MF-7: Semilog and LogLog Plots

The focus of this exercise is creating and comprehending semilog and loglog plots. If you are unfamiliar with logarithms, please review them before you attempt this problem.

The majority of plots we'll make in this class contain linear axes. However, it may sometimes be advantageous to plot a dataset on logarithmic axes. For instance, an exponential function (such as the function modeling the spread of COVID-19), can grow quickly over a short period of time. Not only is this visually unappealing on a standard linear scale, but plotting on a linear scale may obscure trends easily noticeable if one or both axes are plotted on a logarithmic scale. Plotting the simple function $y = x$ on linear axes yields the (hopefully) familiar graph:

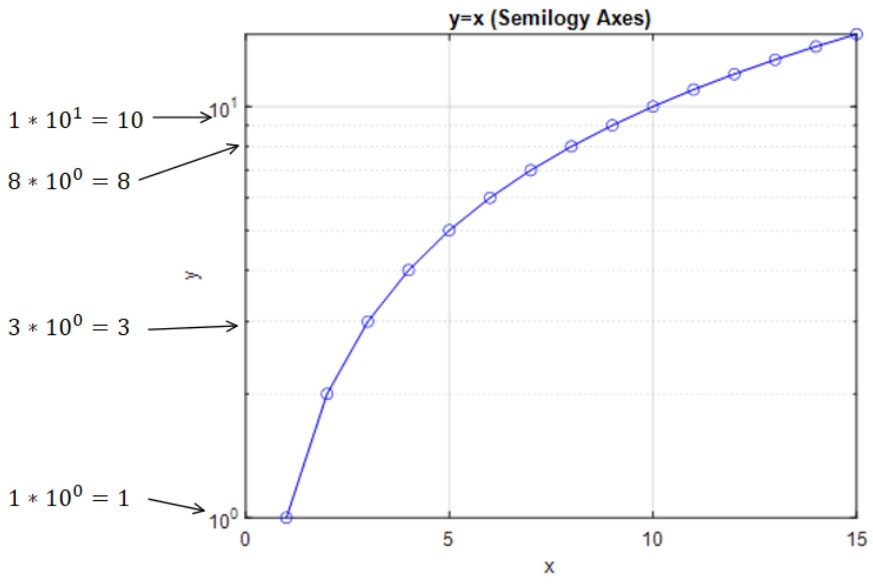


This plot has linear axes on both the x- and y-axes because each axis tick is incremented by 1. The following figure plots the same function but with a logarithmic scale on the y-axis:



The x-axis is unchanged, but the y-axis's ticks are unevenly spaced. This is referred to as a *semilogy* plot because the y-axis is logarithmically scaled. Notice how the very linear function $y = x$ looks exponential; this is a byproduct of the y-axis scaling. Also note that there are no negative y-values, as a logarithmic never yields negative numbers (by definition).

How do you read a semilogy plot? On a linear scale, each tick represents 1. But on this particular y-axis, each tick represents $(tick\ value) * 10^{some\ power}$:



You can also plot the x-axis exclusively logarithmically. Finally, you can plot both axes logarithmically; this is called a *loglog* plot.

Consider a microbial population. The number of microbes, P, over different temperatures is:

T (°C)	-2	-1	0	1	2	3	4
P	0.0204	0.143	1	7	49	343	2401

The population grows by a factor of 7 for every unit temperature increase, so the data can be modeled by the function $P = 7^T$.

1) Plot the data using linearly scaled x and y axes. Why is this graph problematic?
2) Create a semilogy plot using MATLAB's `semilogy()` function. Be sure you know how to read the individual data points.
3) Create a semilogx plot using MATLAB's `semilogx()` function. Be sure you know how to read the individual data points.
4) Create a loglog plot using MATLAB's `loglog()` function. Be sure you know how to read the individual data points.
5) Which of the logarithmic plots is the "best"? Why?

2. Advanced MATLAB (AM)

Objective

MATLAB’s ease of use can be attributed to several things:

1) A vast library of built-in functions accompanied by comprehensive documentation.
2) High-level data structures and handling of data types.
3) It’s an *interpreted* programming language, as opposed to a *compiled* language like C++.

The advantage of interpreted languages comes from their ability to execute code “on the fly”. Unlike a C++ program, a MATLAB script is simply a sequence of instructions that could very well have been typed by the user and executed line-by-line in the MATLAB terminal. A programmer can simply piece together a program by experimenting in the terminal and examining the outputs after each piece of code (the visualization and plotting functions in MATLAB makes the task of examining these outputs even easier in certain cases). Functional lines of code can then be built into a MATLAB script or function.

The disadvantage of interpreted languages is slow performance. In MATLAB’s case, this issue can be circumvented by taking advantage of its built-in functions, many of which are highly optimized. Vector and matrix operations are also well-optimized, and thus, vectorization is preferred instead of loop-based, scalar-oriented code (`for` loops and `if` statements). Loop-based, scalar-oriented methods can be used to solve problems, but vectorization is highly preferred.

AM-1: Relational and Logical Operators

Given the following:

```
a = 3;
b = -2;
c = 0:2:8;
d = 3:3:15;
```

Complete each row of the table by hand. If a MATLAB statement produces an error, put `N/A` in the *Dimensions of Output* column and briefly explain why the error occurs in the *MATLAB Output* column. Check your answers in MATLAB.

Case	**MATLAB Statement**	**Dimensions of Output**	**MATLAB Output**
example	`[1 2 3] < [4 5 6]`	1x3 row vector	`[0 0 0]`
a	`a < 3`		
b	`a >= b`		
c	`(a < b) && (b >= 2)`		
d	`(a-b) <= 0`		
e	`a ~= b`		
f	`a = b`		
g	`c <= b`		
h	`c ~= d`		
i	`d > [a b]`		
j	`c < (d-2*a)`		
k	`(a == 3) && (d < 2)`		
l	`(a ~= 0) | (c == d)`		
m	`((a < b) | (b < 3)) && (b ~= -1)`		
n	`(~(c-d < 0)) | (c > a)`		
o	`~((~(a~=b)) & (d./c > 1))`		

AM-2: Estimating the Inverse of e

The inverse of the mathematical constant e can be approximated as follows:

$$e^{-1} \approx \left(1 - \frac{1}{n}\right)^{n}$$

Write a script that will loop through values of n until the absolute value of the difference between the approximation and the analytical value is less than 0.0001. The script should then print out the analytical value of e^{-1} and the approximation to 4 decimal places and also print the value of n required for such accuracy.

Hint: e is not a built-in constant in MATLAB! Use `exp()` instead.

AM-3: Time Response Characteristics

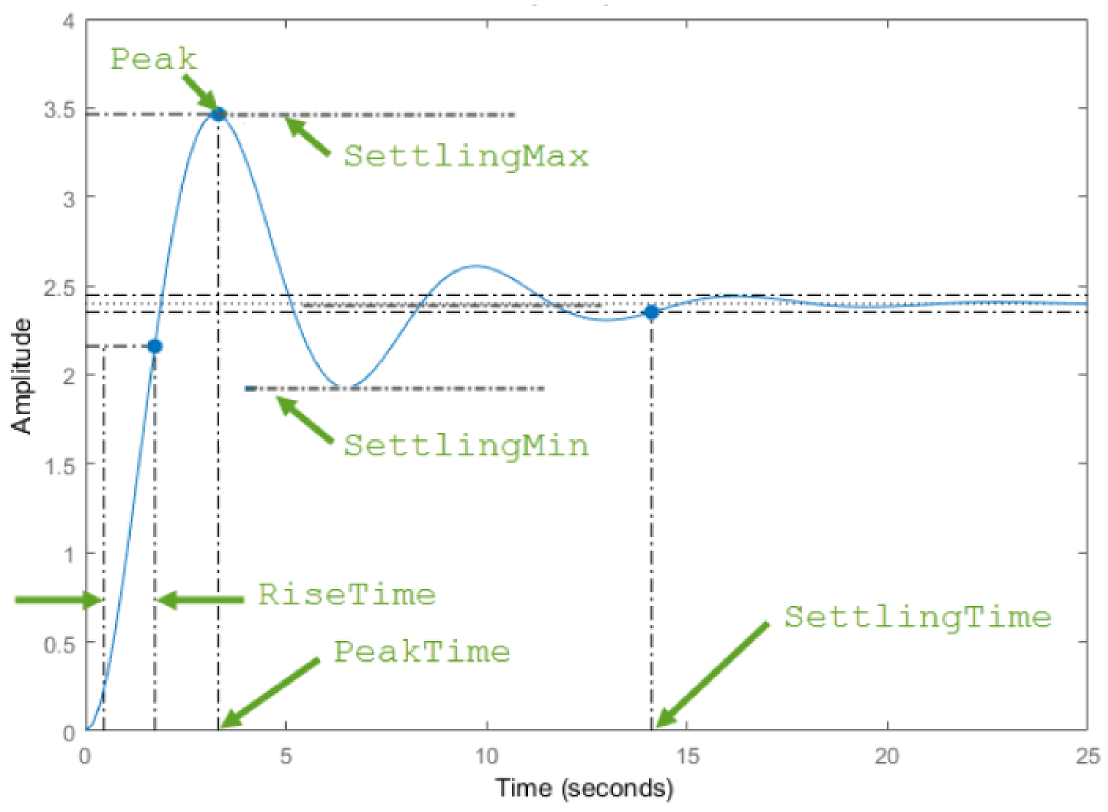
The voltage of an electrical component over time is stored in the `ME2004_VoltageData.mat` file located here.

1) Download the .mat file to your Working Directory and load the dataset in a new script by issuing:

```
load('ME2004_VoltageData.mat');
```

2) If successful, you will see two variables appear in the Workspace: V and t. These are the component's voltage over time, respectively. These two vectors comprise the system's *time response*. Plot voltage over time.

3) There are many metrics of interest when analyzing a system's time response. Consider the time response (amplitude y vs time t) of an arbitrary system.



As seen in the figure, there are many metrics of interest:

- `Peak`: Peak absolute value of $y(t)$
- `PeakTime`: Time at which the peak value occurs
- `Overshoot`: Percentage overshoot, relative to y_{final}
- `RiseTime`: Time it takes for the response to rise from 10% to 90% of the steady-state response, y_{final} (the last element of the y vector)
- `SettlingTime`: Time it takes for the error $e(t) = |y(t) - y_{final}|$ between the response $y(t)$ and the steady-state response y_{final} to fall below and stay within 2% of the peak value of $e(t)$
- `SettlingMin`: Minimum value of $y(t)$ once the response has risen
- `SettlingMax`: Maximum value of $y(t)$ once the response has risen

Compute each of the seven metrics for the supplied dataset. You may not use the `stepinfo()` function in your solution, but you may use it to validate your answers. Do not use the `lsiminfo()` function whatsoever (either in your solution or to validate).

For an additional challenge, compute each metric without using any decision or repetition structures. If you're really ambitious, try to recreate the annotations (labels, arrows, etc.) seen in the figure. The `annotation()` function may be handy.

AM-4: Piecewise Functions

A *piecewise function* is a function defined by multiple sub-functions, where each sub-function applies to a different interval in the domain. The unit step function is a classic piecewise function. It's commonly used in control systems engineering to apply a force at a specified time, then apply another force at a different time.

Piecewise functions manifest themselves in real life. Cell phone carriers offer an amalgam of monthly data packages. In many plans, you pay a fixed price for a fixed amount of data, then pay more once you exceed your usage. The monthly cost C (\$) for using g gigabytes (GB) of data is computed as:

$$C(g) = \begin{cases} 25 & 0 \leq g < 2 \\ 25 + 10(g - 2) & g \geq 2 \end{cases}$$

This formula states that you pay \$25 for using up to 2 GB of data, then pay proportionally to your data usage afterwards.

a) Create a g vector ranging from 0 GB to 5 GB. Choose a sufficiently small step size.
b) Construct $C(g)$ using a `for` loop and `if` statements.
c) Construct $C(g)$ using *vectorized code*, i.e., using relational operators ($<$, $>$, $\leq$, etc…no `for, while, if, elseif, heaviside`, etc.).
d) Construct $C(g)$ using the `heaviside()` function.
e) Plot (b)-(d) on one figure. Differentiate your lines by varying the line markers, colors, thicknesses, etc.

<u>Analysis Questions</u>

1) Looking at the plot, do your lines overlap? Why or why not?
2) In December, our friend Matt Labb didn't use his phone much; he only used 0.8 GB of data. How much did he pay?
3) In January, he obsessively checked GameStop's stock prices and accidentally used 4.4 GB of data. How much did he pay?
4) In February, he attentively monitored his data usage and used exactly 2 GB of data. How much did he pay?

AM-5: Deflection of a Cantilever Beam

The deflection of a cantilevered beam with a point load is:

$$y = -\frac{Wx^2}{6EI}(3a - x) \quad , \quad 0 \le x \le a$$

$$y = -\frac{Wa^2}{6EI}(3x - a) \quad , \quad a \le x \le L$$

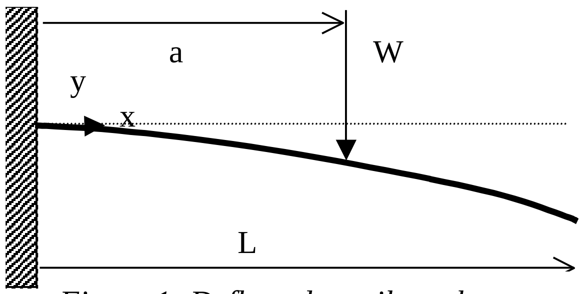


Figure 1: Deflected cantilever beam.

where:

- E = Young's Modulus (psi)
- I = area moment of inertia (in^4)
- L = beam length (in)
- a = location of point load (in)
- W = point load magnitude (lbf)
- x = location along the beam (in)
- y = beam deflection as a function of x (in). y is positive in the upwards direction.

We wish to study the deflection y as function of x for a given set of system parameters. Ordinarily, this would be done by creating two standalone m-files: one *function m-file* and one *script file*. The function file would be used only to compute and plot y versus x; the script file would call the function file repeatedly to perform a parameter study (see how the deflection changes when $E, I, W,$ etc. are changed). In these applications, the script file is referred to as a *driver program* or *main program*.

To consolidate this, we have provided a single script file (template), `BeamCantPointStudy_template.m`, located here. This template merges both of the aforementioned .m-files into one file in which you do all of your coding. Everything is already laid out in place; all you need to do is code.

`BeamCantPoint` Function

First, we must write the function to compute the beam's deflection. Skip to the end of the `BeamCantPointStudy_template.m` file. You will see the following function definition (and the corresponding `end` keyword):

```
function y = BeamCantPoint(E,I,a,W,x)
```

Write the code to compute and generate a plot of y versus x. You may use `for/if` statements, vectorized code, or the `heaviside()` function.

Calling the `BeamCantPoint` Function

After the function is written, we need to call the function in a driver program. Create a new *script* file. Perform the following tasks:

1. Specify the five input parameters `E,I,L,W,` and `x` (this is already done for you):

```
E = 2*10^7;          % Young's Modulus [psi]
I = 0.163;           % Moment of inertia [in^4]
L = 12;              % Beam length [in]
W = 750;             % Point load magnitude [lb]
x = 0:0.01:L;        % Distance along beam [in]
```

2. Call the `BeamCantPoint` function to create a plot with $a = 3$ inches. Indicate the location of the point load (W) on your graph with a downwards-pointing triangle.

3. Use a `for` or `while` loop to repeatedly call the `BeamCantPoint` function to produce a **separate** graph from $a = 0$ to L inches in 2-inch increments. Place every curve on the **same figure** (after this step, you should have 2 figure windows). Indicate the location of each point load (W) on your graph with a downwards-pointing triangle. For this problem (and only this problem), you are not required to add a legend.

Analysis

1. How does the deflection change as a increases?
2. For any value of a, where does the maximum deflection occur?
3. Briefly describe the difference between the overall shape of the beam deflection: from the wall to the point load location vs. from the point load location to the end of the beam.
4. Young's Modulus (E) is a material property representing the beam's stiffness. A higher E represents a stiffer beam. Play around with a range of E values (of your choosing) and hold the other parameters constant. How does the deflection change as you increase/decrease E? **Provide a plot to support your explanation.**
5. The area moment of inertia (I) is a geometric property representing the distribution of matter relative to an axis. A high moment of inertia means the mass is concentrated away from the centroid (think of an I-beam). Play around with a range of I values (of your choosing) and hold the other parameters constant. How does the deflection change as you increase/decrease I? **Provide a plot to support your explanation.**
6. Play around with a range of W values (of your choosing) and hold the other parameters constant. How does the deflection change as you increase/decrease W? **Provide a plot to support your explanation.**
7. Are all of your observations consistent with elementary physics and/or common sense?

Studying how the beam's deflection changes with varying parameter values is known as a *parameter study* or *sensitivity analysis*. Parameter studies are a vital to analyzing a system's limitations and will be performed in nearly every future homework.

AM-6: Analyzing Heart Rate Data

Heart rate is a critical metric in running training as it measures a run's intensity and the resulting cardiovascular adaptation. Athletes often train in five heart rate zones:

Description	Zone	Heart Rate (bpm)
Talking is extremely difficult, if not impossible	5	180+
Hard to talk; can only speak in 2-3 word sentences	4	160-179
Can have a broken conversation	3	140-159
Can have a fluid conversation	2	120-139
Light recovery/warm up	1	0-119

Monitoring heart rate helps athletes optimize their workouts by ensuring they are training within target zones, promoting efficient endurance gains, and preventing overexertion that could lead to injury or burnout. Athletes are typically interested in how much time they spend in each zone during a run. The more time athletes spend in higher zones, the more recovery they likely need.

The file **HRData.xlsx** (located here) contains three sheets, each of which records a runner's heart rate every second during a run. Your task is to write a MATLAB function and a driver script to analyze the heart rate data in each sheet.

To consolidate this, we have provided a single script file (template), `HRAnalysis_template.m`, located here. Everything is already laid out in place; all you need to do is code.

`analyzeData` Function

This function processes the time and heart rate data. At the end of the `HRAnalysis_template.m` file, you will see the following function definition (and the corresponding `end` keyword):

```
function [zone_time,is_easy] = analyzeData(HR)
```

The input is `HR`, the vector of heart rate data. The first output is the five-element vector `zone_time`, which stores the total time spent in each zone (for instance, `zone_time(2)=44` means 44 seconds of the run was spent in Zone 2). The second output is the Boolean `is_easy`, which determines if the run was "easy." A run is "easy" if the (combined) time spent in Zones 1 and 2 comprise at least 75% of the total run time.

You may not use the `histcounts()`, `discretize()`, or any other related functions to obtain `zone_time`.

Calling the Function

After the function is written, we need to call the function. Use a `for` or `while` loop to repeatedly call both functions to analyze all three sheets of the Excel file. Within the body of the loop, you will need to:

1. Use the `readmatrix()` command to extract the time and heart rate data. Read the documentation carefully to learn how to extract data from a specific sheet within a spreadsheet.
2. Call the `analyzeData()` function
3. Produce one figure with two subplots:
   a. Plot the heart rate over time, and
   b. Create a histogram displaying the time spent in each zone. Set the bin edges to the vector `[0 120:20:200]`.

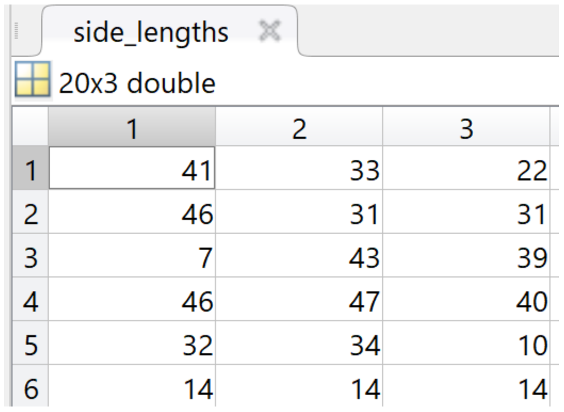
Ultimately, you will end up with three figures (with two subplots each), one for each sheet in the Excel file.

AM-7: Triangle Classifier

This program will determine and print information about triangles. Your job is to determine if the side lengths form a triangle, classify each triangle according to the side lengths, compute its area, and relay that information to the user (aka, print to the Command Window). Because this problem is fairly complex, you will write four user-defined functions to solve individual parts of the problem. You will then write a driver script which invokes the functions.

Data File

The `ME2004_TriangleData.mat` file (located here) holds the variable `side_lengths`:



side_lengths

20x3 double

	1	2	3
1	41	33	22
2	46	31	31
3	7	43	39
4	46	47	40
5	32	34	10
6	14	14	14

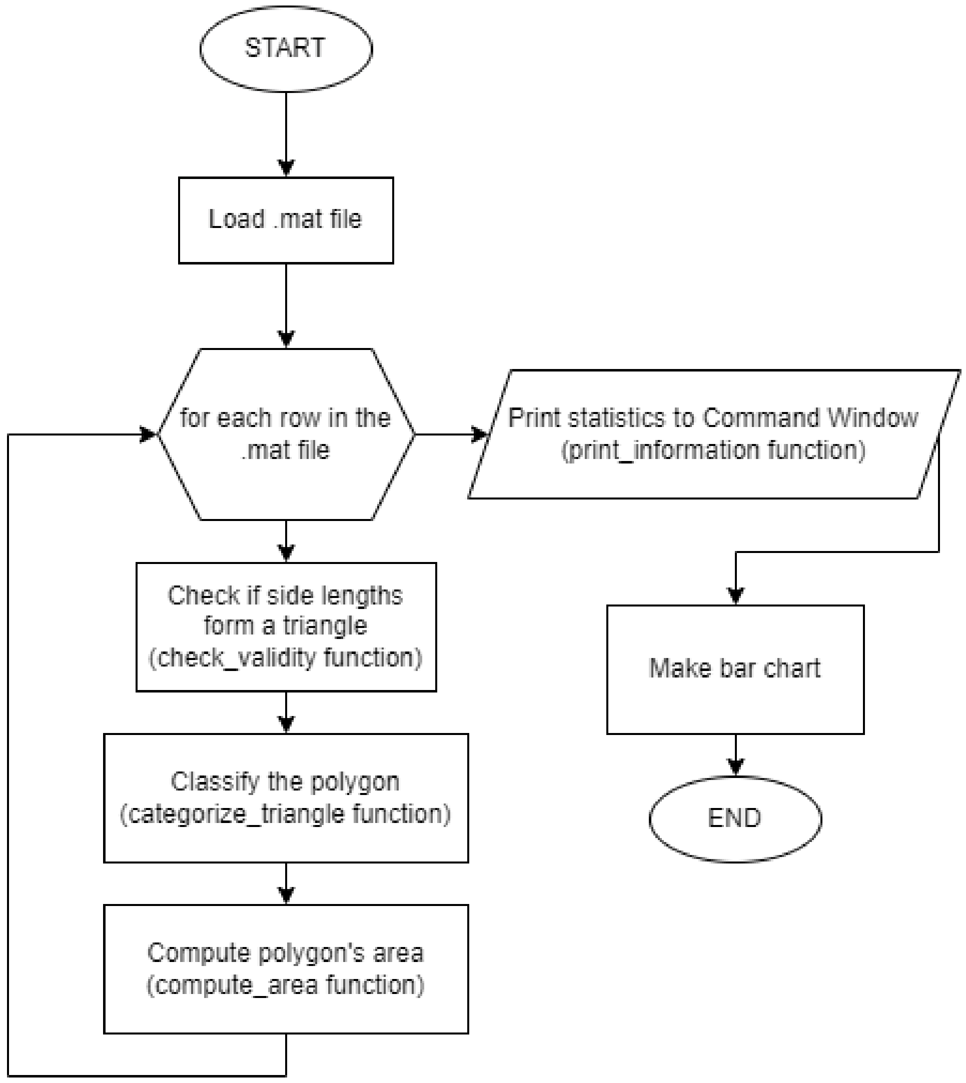
There are 20 three-element row vectors. Each row vector represents the side lengths of a polygon. For instance, the first polygon has side lengths of 41, 33, and 22 units. This is the data you will be using.

Program Structure

The driver script should accomplish the following tasks:

- Load the .mat file
- For each row in the `side_lengths` variable:
  - Determine if the polygon's side lengths form a triangle
  - Classify the polygon (equilateral triangle, isosceles triangle, scalene triangle, or not a triangle)
  - Compute the area
- Print the number of valid and invalid triangles plus the mean area of the valid triangles
- Create a bar chart showing the number of equilateral, isosceles, and scalene triangles (done for you)

The flowchart is given below:


START
Load .mat file
for each row in the .mat file
Print statistics to Command Window (print_information function)
Check if side lengths form a triangle (check_validity function)
Classify the polygon (categorize_triangle function)
Compute polygon's area (compute_area function)
Make bar chart
END

<u>Functions</u>

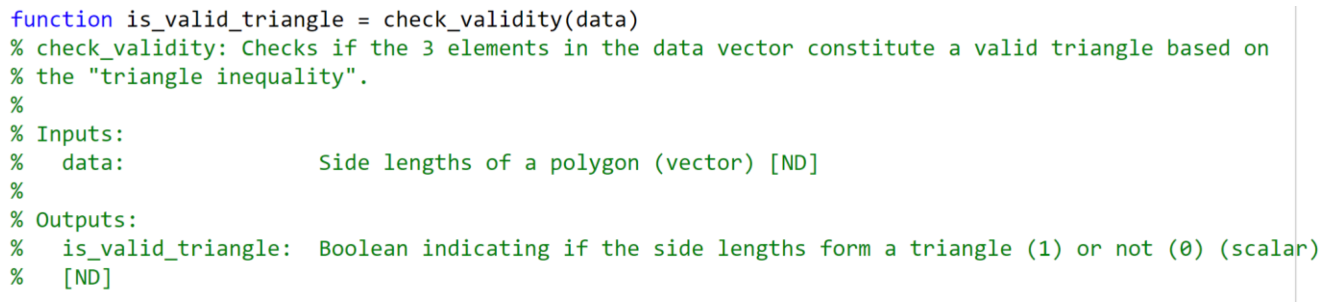
As illustrated in the flowchart, this problem is decomposed into four main subparts, which you will write as standalone function .m files and call in the driver script.

1) `check_validity`: This function tests if the side lengths of one polygon forms a triangle. For any triangle, the length of the largest side (z) must be less than or equal to the sum of the other two sides (x and y). This is known as the Triangle Inequality:

$$z \leq x + y$$

If the Triangle Inequality is satisfied, the polygon forms a triangle. If not, the polygon does not form a triangle (it is likely a self-intersecting polygon, or may not actually be a polygon).

The function header is:



```
function is_valid_triangle = check_validity(data)
% check_validity: Checks if the 3 elements in the data vector constitute a valid triangle based on
% the "triangle inequality".
%
% Inputs:
%   data:               Side lengths of a polygon (vector) [ND]
%
% Outputs:
%   is_valid_triangle:  Boolean indicating if the side lengths form a triangle (1) or not (0) (scalar)
%   [ND]
```

2) `categorize_triangle`: This function classifies the polygon as an equilateral, isosceles, or scalene triangle (if the polygon forms a triangle), or 'not a triangle' (if the polygon does not form a triangle). A triangle is equilateral if all 3 sides have the same length, isosceles if 2/3 sides have the same length, and scalene if all 3 sides are different lengths.

The function header is:

```
function triangle_type = categorize_triangle(data,is_valid_triangle)
% categorize_triangle: Classifies a polygon as an equilateral triangle, isosceles triangle, scalene
% triangle, or not a triangle based on the three side lengths.
%
% Inputs:
%   data:               Side lengths of a polygon (vector) [ND]
%   is_valid_triangle:  Boolean indicating if the side lengths form a triangle (1) or not (0) (scalar)
%   [ND]
%
% Outputs:
%   triangle_type:      The type of triangle ('equilateral', 'isosceles', or 'scalene') for a valid
%   triangle, or 'not a triangle' for an invalid triangle (character vector) [ND]
```

This function accepts the `is_valid_triangle` argument, which is the output of the `check_validity` function.

3) `compute_area`: Like the name suggests, this function calculates the area of the triangle if the polygon's side lengths form a triangle. The area (A) is computed via Heron's Formula:

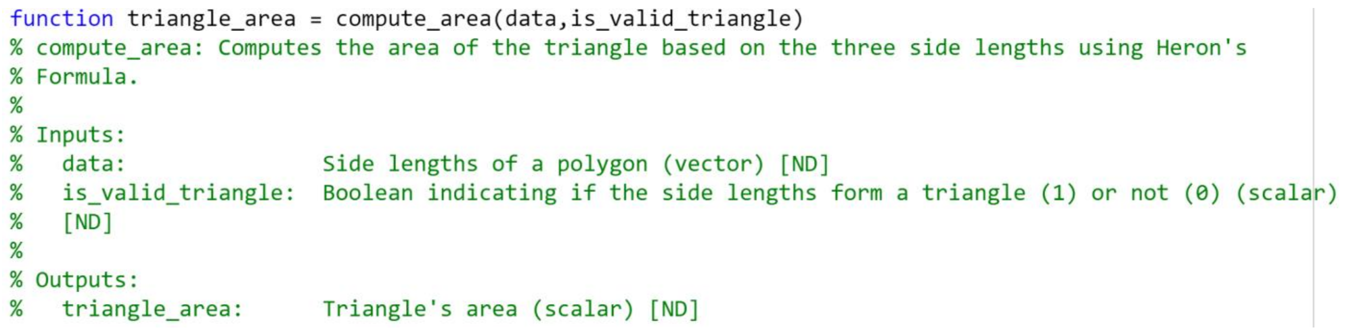
$$A = \sqrt{s(s-x)(s-y)(s-z)}$$

where s is the semi-perimeter, defined as:

$$s = \frac{x+y+z}{2}$$

Heron's Formula only applies to (valid) triangles. If the polygon does not form a triangle, set $A = 0$.

The function header is:

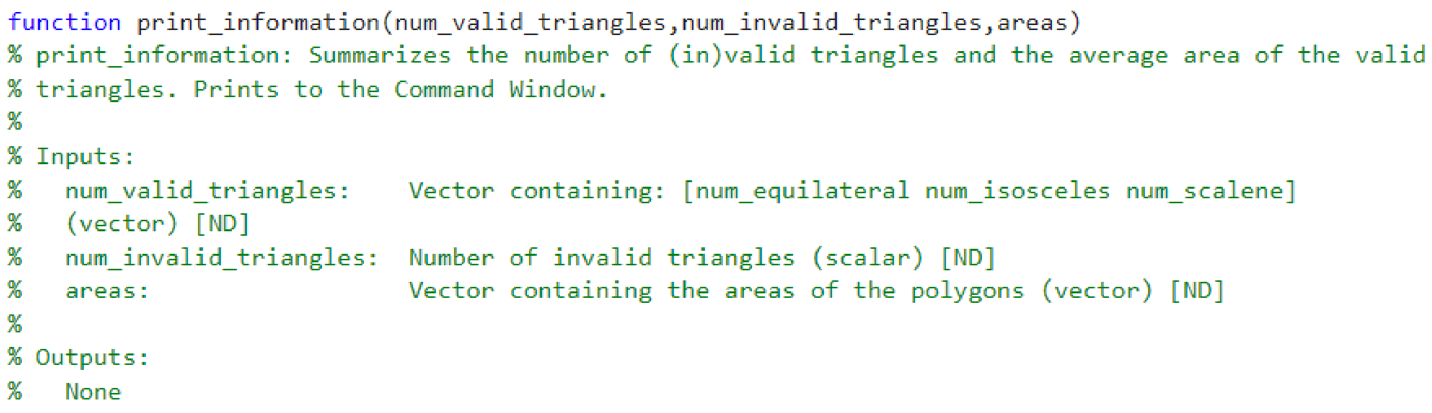


```
function triangle_area = compute_area(data,is_valid_triangle)
% compute_area: Computes the area of the triangle based on the three side lengths using Heron's
% Formula.
%
% Inputs:
%   data:                Side lengths of a polygon (vector) [ND]
%   is_valid_triangle:   Boolean indicating if the side lengths form a triangle (1) or not (0) (scalar)
%   [ND]
%
% Outputs:
%   triangle_area:       Triangle's area (scalar) [ND]
```

This function accepts the `is_valid_triangle` argument, which is the output of the `check_validity` function.

4) `print_information`: This function prints the number of valid and invalid triangles to the Command Window, along with a breakdown of how many of each triangle (equilateral, isosceles, scalene) comprise the total number of valid triangles. Finally, the function also calculates and prints the average area of the valid triangles.

The function header is:



```
function print_information(num_valid_triangles,num_invalid_triangles,areas)
% print_information: Summarizes the number of (in)valid triangles and the average area of the valid
% triangles. Prints to the Command Window.
%
% Inputs:
%   num_valid_triangles:    Vector containing: [num_equilateral num_isosceles num_scalene]
%   (vector) [ND]
%   num_invalid_triangles:  Number of invalid triangles (scalar) [ND]
%   areas:                  Vector containing the areas of the polygons (vector) [ND]
%
% Outputs:
%   None
```

This function accepts `num_valid_triangles`, `num_invalid_triangles`, and `areas`, which need to be initialized somewhere within your code before the function is invoked. The Command Window output should mirror the following:

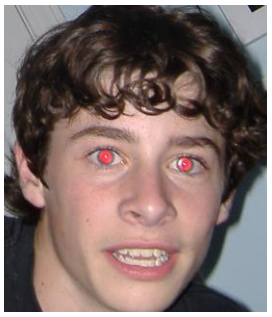
```
In total, there are AA valid triangles and BB invalid triangles.
Of the valid triangles, there are:
     CC equilateral triangles,
     DD isosceles triangles,
 and EE scalene triangles.
The average value of the valid triangles' area is FF units^2.
```

where ***AA***-***FF*** are values to be computed.

AM-8: Red Eye Removal

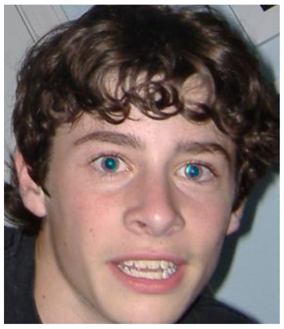
Review the following discussion of how an RGB image can be represented and in MATLAB: https://www.geeksforgeeks.org/matlab-rgb-image-representation/.

The image shown below, *redeye.jpg*, can be downloaded here.



The red eyes in this photo are caused by strong camera flash reflections from the retina. This is particularly common in low light conditions (pupils are wider and let in more light) and when the axes of the flash and the camera lens are closely aligned (i.e. just about any compact digital camera). Fortunately, because of improved sensors, LED-based flashes, embedded post-processing software, etc., this effect is less common with smartphones.

Use MATLAB to "remove" the "red eye" in this photo. Be careful not to significantly affect the color of the other portions of the image. Use MATLAB's `imshow()` command to display the corrected image. It should look something like the following:



NOTE: This "red-eye" correction is far from perfect, e.g. notice the red rings around the pupils?
Can you do better?

AM-9: Pi Estimation

We all know that $\pi = 3.14159265\ldots$, but how might we approximate π ourselves? Assume we have a square which spans both axes from -1 to 1, and a unit circle which falls within this square. We know that $A_{circle} = \pi r^2$, and in this case we know that $r = 1$. Therefore, the area of the circle should be π.

Prove this is true by creating a Monte Carlo simulation using random numbers in the range of -1:1 in both axes, and by determining the ratio of the number of points that fall on or in the unit circle to the total number of points. This *Area Ratio* tells us the percent of the total area that was occupied by the circle. Multiplying the Area Ratio by the total area of the square will yield the estimation of π. No prior knowledge of Monte Carlo simulations is expected or even necessary to complete this assignment. However, some insight from *Appendix A* may prove useful.

1. Create a *function* m-file with the following definition:

```
function [estimate,pcerror] = EstimatePi(numPoints)
```

This function should estimate π from the last paragraph. `estimate` is the π estimation, `pcerror` is the percent error between the estimate and the actual value of π, and `numPoints` is the number of randomly generated points to be used in the simulation. The function should also create a plot formatted as follows:

- Plot the unit circle in black. You may find the `viscircles()` function handy.
- Plot points outside the unit circle as blue X's. You may find the `scatter()` function handy.
- Plot points on or inside the unit circle as red O's. You may find the `scatter()` function handy.

2. Create a driver script m-file to study the accuracy of the estimate of π versus `numPoints`. Call your function created in (1) to complete the following statistical study. Use the `table()` command to neatly print your results to the Command Window.

`numPoints`	`Pi Estimate`	`Percent Error`
10^2		
10^3		
10^4		
10^5		
10^6		
10^7		

The percent error is defined as:

$$\%error = \left|\frac{approx - exact}{exact}\right| * 100 = \left|\frac{Pi\ Estimate - \pi}{\pi}\right| * 100$$

<u>Analysis</u>

1) Re-run your code numerous times. Why does the output change after each run?
2) Explain the results of your statistical study. How does `numPoints` affect the π estimate and the percent error?
3) Examine each plot (you should have 6, one for each value of `numPoints`). Are there any plots which appear to be flawed due to a dearth of points? Are there any plots which appear to contain too many points? What's the minimum number of points needed to estimate π without over-cluttering the plot? (These are mostly opinion-based questions)

AM-10: Two Aces

The scene from an old cowboy movie, where four aces is beat by a straight flush in a game of 5-card stud, almost never happens in real life. First, 5-card stud is almost never played anymore and second, it is extraordinarily unlikely that two such powerful hands would be dealt at the same time – especially with only 5 cards to work with.

A more typical hand of poker would involve much less dramatic confrontations – such as two pair beating one pair. Also, the most popular game played at both casino and private poker games throughout the world is a game called holdem. The main event at the world series of poker, formally at Binions Horseshoe and now at the Rio in Las Vegas, is no-limit holdem with a first prize up to 12 million dollars (larger than first place money in any tennis or golf championship). In holdem, each player is first dealt two individual cards. Then 5 community cards are dealt face up. The best hand using the two individual cards and 5 community cards wins. The best starting hand in holdem is two aces.

Because of the computer and all the simulation software available today, in addition to TV coverage, there is an explosion of information about poker strategies and correct play based on probability. The objective of this assignment is to begin to understand some of the methods that might be required to simulate the play and statistical analysis of poker on the computer. The task of writing a complete computer program to simulate poker (or any other game like chess or backgammon) is complex and beyond the scope of the course. However, we will have the computer help us with a much simpler task – computing the probability that a player will be dealt two aces as starting cards in holdem.

1. Derive the theoretical probability that you will be dealt two aces as your starting cards in holdem. *Hint:* there are 52 cards in a deck, so what are the odds that the first card you are dealt is an ace?
2. Write a *function* m-file which accepts a specified number of hands and returns the percent of hands that a player is dealt two aces. The function definition should be:

```
function pcTwoAces = ProbTwoAces(NHands)
```

where:

- `NHands` = number of hands or trials
- `pcTwoAces` = percentage of hands consisting of 2 aces

This type of analysis applied to engineering problems is called a *Monte Carlo Method*.

3. Write a separate *driver script* which repeatedly calls the `ProbTwoAces` function to perform the following statistical study:

`NHands`	`pcTwoAces`
10^2	
10^3	
10^4	
10^5	
10^6	
10^7	
10^8	

Use the `table()` command to print this table to the Command Window.

4. How do your results in (3) compare to the theoretical probability derived in (1)?
5. (Optional) For the extra ambitious: generate a loop-based `ProbTwoAces` function and a fully vectorized `ProbTwoAces` function. Run the statistical study for both functions. Use `tic` and `toc` (read the documentation) to compare the runtimes. At what value of `NHands` does vectorized code run significantly faster than loop-based code?

Remark. There are 52 cards in a deck. If each unique card is represented by a number (1-52), the aces can be represented as cards 1-4. Use the `randperm()` function which generates a vector of exclusionary random numbers (no number is selected more than once).

3. Numerical Differentiation (ND)

Objective

Analytical differentiation rules (chain rule, product rule, etc.) require a smooth function $f(x)$. In practice, you likely obtain a set of data points from a digital readout machine, rendering it difficult to construct the $f(x)$ necessary for analytical differentiation. You could curve fit the data (which we will do later in the semester), but the required justification is so tedious and extensive that doing so is rarely worthwhile*.

Thankfully, numerical differentiation provides a direct means of computing $\frac{dy}{dx}$ given a dataset. We have learned three main techniques: the forward, backward, and central difference methods. In these problems, you will apply the three numerical differentiation techniques.

*Much of engineering revolves around *why* you made a particular design decision, not *what* the decision is.

ND-1: Numerical Differentiation Step Size Experimentation

In class, we experimented with the step size for the forward, backward, and central differences. This problem is another spin on step size experimentation. Consider the function:

$$y(x) = \cos{(x)}$$

a) Analytically compute $\frac{dy}{dx}$ at $x = \frac{\pi}{6}$.
b) Compute $\frac{dy}{dx}$ at $x = \frac{\pi}{6}$ using the backward difference method with step sizes $h = 10^{-n}$, where $n = -1{:}\,14$. What value of n minimizes the percent error with respect to the analytical value of $\frac{dy}{dx}$ at $x = \frac{\pi}{6}$?
c) Repeat (b) for the central and forward difference methods.
d) Create a single `loglog()` plot of $\%error$ vs. h containing the results for all 3 differentiation schemes. Explain your results, particularly at extremely small step sizes.

Remarks:

1) The percent error is defined as:

$$\%error = \left|\frac{approx - exact}{exact}\right| * 100\%$$

2) In class, we saw that the central difference is generally more accurate than the forward/backward differences. You should hopefully verify this (or at least see this through another lens) here.
3) Before you explain your plot in (d), you may want to revisit the Numerical Errors 1 lecture.

ND-2: Numerical Differentiation of Data

Consider the following data. We wish to approximate the derivative, $\frac{dy}{dx}$.

x	y
0	2
1.5	0.45
3	0.10
4.5	0.002

a) Compute the numerical derivatives of this data using the forward difference method. Why can you not determine the forward difference approximation at the last point?
b) Compute the numerical derivatives of this data using the backward difference method. Why can you not determine the backward difference approximation at the first point?
c) Compute the numerical derivatives of this data using the central difference method. Why can you not determine the centered difference approximation at the first and last points?
d) Plot your results on separate figures.

ND-3: Heat Transfer

Heat Transfer: "The transfer of energy between systems due to a temperature difference"

Heat transfer is a very important phenomenon in engineering and affects the performance of jet engines, internal combustion engines, heat exchangers, etc. You will take a class dedicated to heat transfer (ME 3304) in your junior year. Consider Figure 1 below:

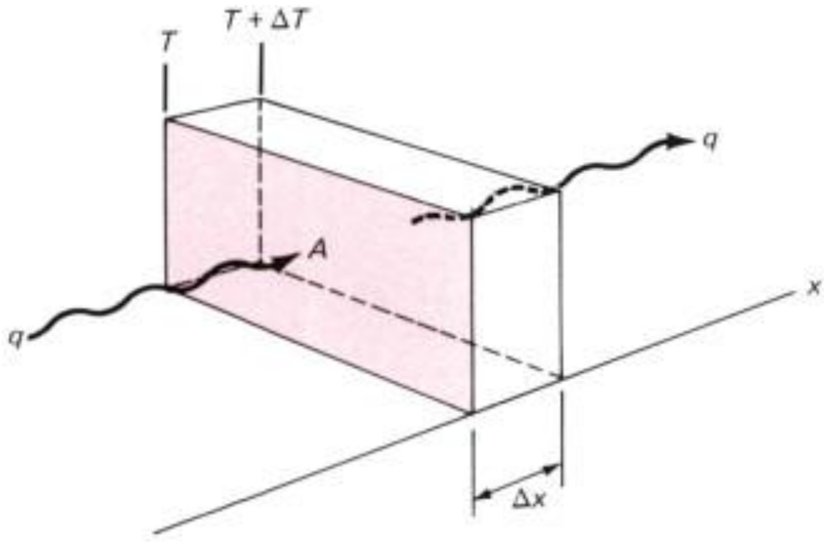


Figure 1. Heat transfer through a material

Here, heat is entering the material from the left face of an unknown material and exits from the right face. Assume the left face is hotter than the right face. One metric quantifying the amount of heat transferred is the *heat flux, $q"$*. Heat flux is the power per unit area through non-moving materials and is given by Fourier's Law of Conduction:

$$q" = -k\frac{dT}{dx} \qquad \text{Equation 1}$$

where:

- $q"$ = heat flux $\left(\frac{W}{m^2}\right)$
- $\frac{dT}{dx}$ = gradient (slope) of the temperature within the material $\left(\frac{°C}{m}\right)$
- k = thermal conductivity (material property) $\frac{W}{m\,°C}$

For semantics, the above diagram uses q instead of $q"$. The heat flux ($q"$) is the heat transferred (q) per unit area, so $q" = q/A$. Thus, the heat flux is just the heat transfer normalized with respect to the material's size (A). In heat transfer, using $q"$ over q is customary.

Observing Equation 1 reveals the greater the temperature drop $\frac{dT}{dx}$ through a material, the higher the rate of heat flux q". The negative sign reflects the flow of heat from hot to cold. Suppose you want to investigate heat transfer within a wall (like the above image). You place thermocouples at various locations within the wall and record the temperatures at a particular time. This data is compiled in the `ME2004_HTdata.mat` file, located here.

1a) Load the data into your code by issuing:

```
load('ME2004_HTdata.mat');
```

Upon execution, this should load two vectors T $(°C)$ and x (m) into the Workspace. What are the dimensions of each vector?

1b) Plot the temperature distribution (T vs x) using black circles with a dashed connecting line.

2a) Compute the heat flux q" within the wall using Equation 1. Use a forward difference at the first point, backward difference at the last data point, and central difference at the interior points. Assume the thermal conductivity $k = 0.25\frac{W}{m\,°C}$.

2b) On a new figure, plot the heat flux distribution (q" vs x) using blue pentagrams with a linewidth of 1.5 and a dashed connecting line. *(You should have 2 figures after this step)*

3a) At any position in the wall, the time rate of change of temperature $\left(\frac{dT}{dt}\right)$ is proportional to the gradient of the heat flux as given by the formula:

$$\frac{dT}{dt} = \left(\frac{-1}{\rho C_p}\right)\left(\frac{dq"}{dx}\right) \qquad \text{Equation 2}$$

where:

- $\frac{dT}{dt} =$ time rate of change of temperature $\left(\frac{℃}{\mathrm{s}}\right)$
- $\rho = 700 =$ wall's density $\left(\frac{kg}{m^3}\right)$
- $C_p = 1200 =$ specific heat $\left(\frac{J}{kg\ °C}\right)$

Using Equation 2, compute $\frac{dT}{dt}$.

3b) On a new figure, plot $\frac{dT}{dt}$ vs x using magenta downwards-pointing triangles with a linewidth of 2 and a dashed connecting line. *(You should have 3 figures after this step)*

4a) We preach the value of thoroughly testing your code. We will check $q"$ and $\frac{dT}{dt}$ three ways. One quick check is comparing the average value of $q"$ and $\frac{dT}{dt}$ ($q^{"}_{avg}$ and $\frac{dT}{dt}_{avg}$ respectively). *Average* can hold many meanings depending on the context. First, inspect the $q"$ and $\frac{dT}{dt}$ vectors. What are $q"$ and $\frac{dT}{dt}$ at the center of the wall? This is our first estimate of $q^{"}_{avg}$ and $\frac{dT}{dt}_{avg}$.

4b) Next, use the `mean()` function to estimate $q^{"}_{avg}$ and $\frac{dT}{dt}_{avg}$.

4c) Finally, check your results by performing a "back of the envelope" calculation (very common in practice) to estimate the average heat flux and rate of temperature change:

$$q''_{avg} \approx -k\frac{\Delta T}{L} \qquad \text{Equation 3}$$

$$\frac{dT}{dt}_{avg} \approx \left(\frac{-1}{\rho C_p}\right)\left(\frac{\Delta q''}{L}\right) \qquad \text{Equation 4}$$

4d) You should now be able to fill in the following table:

	q''_{avg}	$\frac{dT}{dt}_{avg}$
Center of the wall (4a)		
`mean()` function (4b)		
Equations 3-4 (4c)		

Recreate this table in MATLAB and print it the Command Window using the `table()` function.

Analysis

1) Which direction is the heat flux q'' within the wall? Where is the maximum q''?
2) Does your estimated heat flux distribution appear "smooth"? If not, why not? What about the plot of the rate of temperature change?
3) Consider the table you created in Problem 4. How do your results compare across the 3 methods? Which method(s) might be better than other? Why?

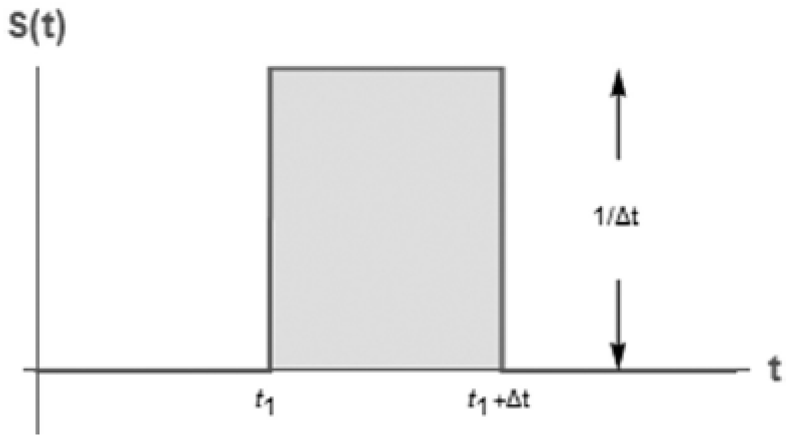
4. Numerical Integration (NI)

Objective

Functions to be integrated numerically will typically be of two forms: a table of values or a function. The form of the data has an important influence on the approaches that can be used to evaluate the integral. For tabulated information, you are limited by the number of points that are given. In contrast, if the function is available, you can generate as many values of $f(x)$ as are required to attain acceptable accuracy. In these problems, you will apply numerical integration techniques to real engineering problems.

NI-1*: *Pulse Integral

Consider the pulse function $S(t)$.



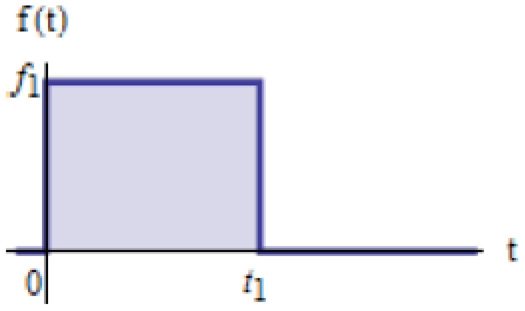
a) Determine (analytically or graphically) the symbolic solution and sketch the function $F(t) = \int_{t_0=0}^{t} S(t_0)\ dt_0$. (*Hint:* The integrand is defined in 3 parts, so think of the integral that way.)

b) Analytically determine $\lim_{\Delta t \to 0} F(t)$.

c) Create a MATLAB plot of $F(t)$ using $t_1 = 2\ s$ and $dt = 10^{-n}$, where $n = -1{:}\,3$. Place all curves on the same figure. Does this verify your answer to (b)?

NI-2: ***Another Pulse***

Consider the function $f(t)$ shown:



a) Determine an expression for $f(t)$ in terms of terms of f_1 and t_1.

b) Determine an expression for the integral $I(t) = \int_{t_o=0}^{t} f(t_o)\ dt_o$ in terms of f_1 and t_1.

c) Determine an expression for the integral $I(t) = \int_{t_o=0}^{t} f(t_o)e^{\frac{-(t-t_0)}{\tau}}\ dt_o$, where τ is an arbitrary constant, in terms of f_1, t_1 and τ.

d) Create a 3x1 subplot. Plot $f(t)$ on the upper subplot, $I(t)$ from (b) on the middle subplot, and $I(t)$ from (c) on the lower subplot. Take $f_1 = \tau = 1$ and $t_2 = 5$.

NI-3: Basics of Numerical Integration

Consider the integral $I = \int_{x=0}^{4} (1 - e^{-2x}) dx$.

a) Plot the integrand $f(x) = (1 - e^{-2x})$ versus x. Estimate the integral I from your plot.
b) Evaluate the exact value of this integral, I_{exact}.
c) Approximate the integral using the methods listed. Sketch or plot each approximation on a graph of $f(x)$. The percent error is defined by:

$$\%error = \left| \frac{I_{approx} - I_{exact}}{I_{exact}} \right| * 100\%$$

	Rule	**Numerical Approximation** $\boldsymbol{I_{approx}}$	**% Error**
a	Trapezoid, $n = 1$		
b	Trapezoid, $n = 2$		
c	Trapezoid, $n = 4$		
d	MATLAB `integral()` command		

Print this table to the Command Window.

To clarify what is meant by "plot each approximation on a graph of $f(x)$", Fig. 4.19a in the Vick text would be an example for case (a), Fig. 4.19b is an example for case (d), and Fig. 4.22 is an example of the trapezoid rule for n = 5.

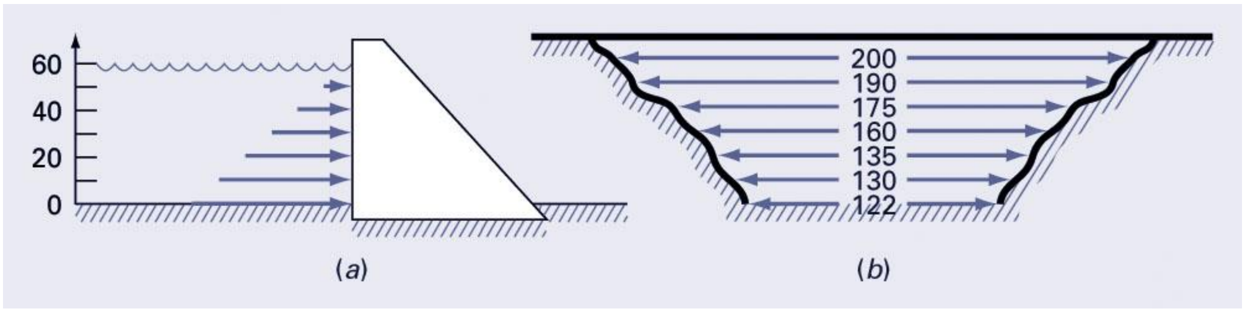
NI-4*: *Numerical Integration Step Size Exploration
Consider the function:

$$y(x) = -sin(x)$$

a) Analytically compute $\int_0^{\frac{\pi}{6}} y(x)\,dx$.
b) Compute $\int_0^{\frac{\pi}{6}} y(x)\,dx$ using MATLAB's `trapz()` function with step sizes of $h = 10^{-n}$, where $n = 1:8$. *NOTE: Your script may require a bit of time to run, depending on how it is designed. Be patient.*
c) Create a single `loglog()` plot of $\%error$ vs. h.

NI-5: Forces on a Dam

Water exerts pressure on the upstream face of a dam as shown.



The pressure increases linearly with depth and can be characterized by $p(z) = \rho g(D - z)$ where $p(z)$ is the pressure (in $\frac{N}{m^2}$) exerted at an elevation z meters above the reservoir bottom; ρ is density of water, which for this problem is assumed to be a constant $10^3\,\frac{kg}{m^3}$; g is the acceleration due to gravity $\left(9.81\,\frac{m}{s^2}\right)$; and D is elevation (in m) of the water surface above the reservoir bottom. Omitting atmospheric pressure (because it works against both sides of the dam face and essentially cancels out), the total force f_t can be determined by multiplying pressure times the area of the dam face. Because both pressure and area vary with elevation, the total force is obtained by evaluating:

$$f_t = \int_0^D \rho g \cdot (D - z)w \cdot dz.$$

where $w(z)$ is width of the dam face (m) at elevation z. The line of action can also be obtained by evaluating:

$$d = \frac{\int_0^D z\rho g \cdot (D - z)w \cdot dz}{\int_0^D \rho g \cdot (D - z)w \cdot dz} = \frac{\int_0^D z\rho g \cdot (D - z)w \cdot dz}{f_t}$$

a) The following table contains data of stream width at various elevations.

Elevation, z (m)	0	10	20	30	40	50	60
Width, w (m)	122	130	135	160	175	190	200

Write a function that accepts vectors containing the z locations and the corresponding w values on the dam face. The function should perform numerical integration using `trapz()` and return the values f_t and d. Test your function using the specific data given.

b) Now you have the following data:

Elevation, z (m)	0	10	20	50
Width, w (m)	122	130	135	190

(Note that this is the same as the table from (a) with the 4th, 5th, and 7th columns omitted)

Call the function you wrote in (a) using the new data and return the new f_t and d values. Compare these to the f_t and d values obtained in (a).

5. Linear Algebra and Applications (LA)

Objective

Many of the fundamental equations of engineering and science are based on conservation laws. Some familiar quantities that conform to such laws are mass, energy, and momentum. In mathematical terms, these principles lead to balance or continuity equations that relate system behavior as represented by the levels or response of the quantity being modeled to the properties or characteristics of the system and the external stimuli or forcing functions acting on the system.

These systems are often multifaceted. Multicomponent systems result in a coupled set of mathematical equations that must be solved simultaneously. The equations are coupled because the individual parts of the system are influenced by other parts. For example, applying a force at one node in a truss affects the forces in the rest of the truss.

Aside from physical systems, simultaneous linear algebraic equations also arise in a variety of mathematical problem contexts. These result when mathematical functions are required to satisfy several conditions simultaneously. Each condition results in an equation that contains known coefficients and unknown variables. The techniques discussed in class can be used to solve for the unknowns when the equations are linear and algebraic. Some important numerical techniques that employ simultaneous equations are regression and interpolation. Linear algebra is arguably the most important class topic due to its versatility and span of problems. MATLAB itself is founded on solving linear systems. A rigorous understanding of linear algebra concepts and applications translates well to the rest of the class.

Do not use `equationsToMatrix()`, `solve()`, `linsolve()`, or any related functions for these problems.

LA-1: Linear Algebra Basics

Consider the special cases of the systems of equation $Ax = b$ listed:

Case	A	b
a	$\begin{bmatrix}1&0\\1&1\end{bmatrix}$	$\begin{bmatrix}2\\1\end{bmatrix}$
b	$\begin{bmatrix}1&1\\0&0\end{bmatrix}$	$\begin{bmatrix}2\\1\end{bmatrix}$
c	$\begin{bmatrix}1&1\\0&0\end{bmatrix}$	$\begin{bmatrix}0\\0\end{bmatrix}$
d	$\begin{bmatrix}1&0\\0&0\end{bmatrix}$	$\begin{bmatrix}0\\0\end{bmatrix}$
e	$\begin{bmatrix}1&1\\1&0\\0&1\end{bmatrix}$	$\begin{bmatrix}2\\1\\1\end{bmatrix}$
f	$\begin{bmatrix}1&1\\1&0\\0&1\end{bmatrix}$	$\begin{bmatrix}1\\-1\\-1\end{bmatrix}$
g	$\begin{bmatrix}1&2&0&2\\0&1&3&4\\1&0&1&2\end{bmatrix}$	$\begin{bmatrix}4\\4\\4\end{bmatrix}$
h	$\begin{bmatrix}1&2&-3&1\\0&1&-2&1\\1&0&1&-1\end{bmatrix}$	$\begin{bmatrix}-1\\2\\1\end{bmatrix}$

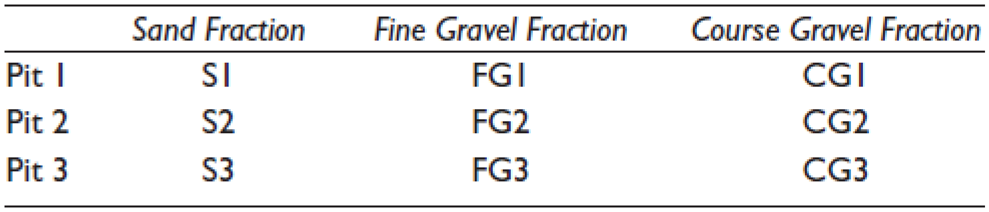
For each case:

a) Generate (by hand or in MATLAB) the row and column interpretation (only for whichever one of Case (a)-(d) you pick)
b) Compute the rank of the coefficient matrix, A
c) Compute the rank of the augmented matrix, $\tilde{A} = [A\ b]$
d) Determine whether the system is consistent. If the system is consistent, give the solution(s).
e) Compute the determinant of the coefficient matrix if it exists.

You may use MATLAB for the entirety of the problem, but you should know how to obtain the rank, consistency, and determinant of a 2x2 system by hand. Some parts (such as sketching the row/column interpretations) may be easiest by hand.

LA-2: Gravel Pits

A civil engineer involved in a construction process requires a volume V_s of sand, a volume V_{fg} of fne gravel, and a volume V_{cg} of course gravel. There are three pits from which these materials can be obtained. The composition of these pits is:



	Sand Fraction	Fine Gravel Fraction	Course Gravel Fraction
Pit 1	S1	FG1	CG1
Pit 2	S2	FG2	CG2
Pit 3	S3	FG3	CG3

a) Develop the equations needed to determine the volumes V_1, V_2, and V_3 that must be hauled from pits 1, 2, and 3 to exactly meet the construction needs. Arrange in matrix form.

b) For the following special case, compute V_1, V_2, and V_3 given $V_s = 4800\ m^3, V_{fg} = 5800\ m^3, V_{cg} = 5700\ m^3$:

	Sand Fraction	**Fine Gravel Fraction**	**Course Gravel Fraction**
Pit 1	0.55	0.30	0.15
Pit 2	0.25	0.45	0.30
Pit 3	0.25	0.20	0.55

LA-3: Cooling of an IC Package

Consider the cooling of an IC package from B. Vick *Applied Engineering Mathematics*, Section 5.3.1. The purpose of this exercise is to write a function to solve for the unknown heat flows $(Q_1 - Q_4)$ and temperatures* (T_c, T_p, T_w) and test this function. You are requested to:

a) Write a function called `CoolingIC` that computes the heat flow rates and temperatures in the IC package. The inputs should be:
- R = vector containing the five thermal resistance values
- T_a = ambient temperature
- Q_c = power dissipated.

The function returns two vectors:

- T = vector containing the three unknown temperatures
- Q = vector containing the four unknown heat flow rates

b) Write a separate M-file to test your `CoolingIC` function using the test cases below. This m-file simply assigns the input values, invokes `CoolingIC` to compute the T and Q vectors, and displays the output. No `assert()` statements are required, but you should include some for your own sake.

Case	R_1 (°C/W)	R_2	R_3	R_4	R_5	Q_c (W)	T_a (°C)
1	2	2	2	2	2	0	25
2	2	2	2	2	2	10	25
3	2	0.5	35	0.7	1	10	25
4	1000	0.5	35	0.7	1	10	25

*The Vick textbook defines the temperatures in K, but we will use $°C$ here.

The Command Window output for each case should mimic the following:

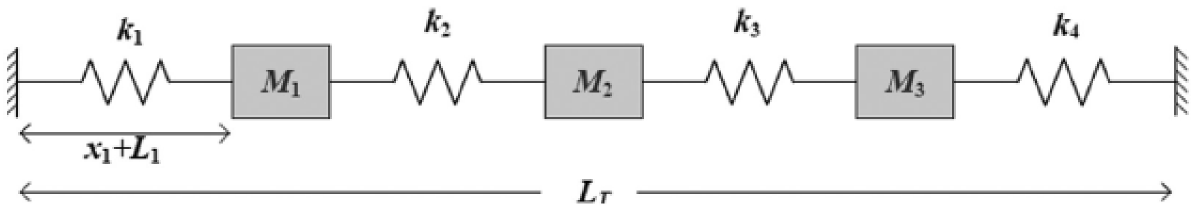
```
--- Case 3 Inputs ---
 Ta = 25.0 degC
 Qc =10.00 W
 R1 = 2.00 degC/W
 R2 = 0.50 degC/W
 R3 =35.00 degC/W
 R4 = 0.70 degC/W
 R5 = 1.00 degC/W

 --- Case 3 Outputs ---
 Tc = XXX degC
 Tp = XXX degC
 Tw = XXX degC
 Q1 = XXX W
 Q2 = XXX W
 Q3 = XXX W
 Q4 = XXX W
```

We will now use the function to perform a parameter study on the cooling arrangement of the IC system. The most important engineering result of this analysis is the chip temperature, T_c. This is particularly crucial, since electronic chips can fail if overheated. The cooling arrangement must maintain the chip temperature below a critical failure temperature, T_{crit}. As electrical circuits become smaller and draw more power, overheating becomes a limiting constraint in their design.

c) Study the chip temperature as a function of power. Create a plot of T_c versus Q_c with Q_c ranging from 0 to $100\ W$. On a single graph, put curves for $R_4 = R_5 = 0, 2, 4,$ and $6\ °\frac{C}{W}$. Take other parameter values from Case 3 in the above table.
d) What can you conclude about T_c versus Q_c? If $T_{crit} = 150\ °C$, what is the maximum allowable chip power, Q_c, for each case?
e) Study the chip temperature as a function of air flow (R_4 and R_5). The resistances R_4 and R_5 represent the effect of the air cooling; high resistance corresponds to low air velocity, and low resistance corresponds to high air velocity. Assume $R_4 = R_5$ and create a plot of T_c versus R_4 with $R_4 = R_5$ ranging from 0 (hurricane) to $100\ °\frac{C}{W}$ (stagnant air). On a single graph, put curves for $Q_c = 0, 5, 10,$ and $15\ W$. Take other parameter values from Case 3 in the above table.
f) If $T_{crit} = 150\ °C$, what is the allowable range of resistance values for $Q_c = 10\ W$?

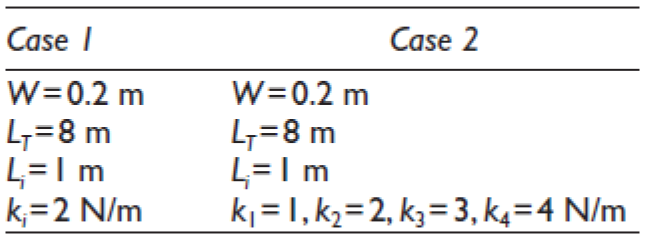
LA-4: Equilibrium Position of a System of Linear Springs and Masses
Consider the system of masses and linear springs:



The total length of each spring is equal to the unstretched length (L_i) plus the stretched length (x_i). The width of each block is W, and the total length of the system is L_T. For a linear spring, the force is directly proportional to elongation, $F_{spring} = kx$. The equations governing the system are:

Total Length	$L_T = (L_1 + x_1 + W) + (L_2 + x_2 + W) + (L_3 + x_3 + W) + (L_4 + x_4)$
Mass 1	$-k_1x_1 + k_2x_2 = 0$
Mass 2	$-k_2x_2 + k_3x_3 = 0$
Mass 3	$-k_3x_3 + k_4x_4 = 0$

a) Write a MATLAB function to solve the above system of equations.
b) Validate your function by developing at least 2 tests using `assert()`.
c) Use the function developed in (a) to compute each x_i for the following cases:

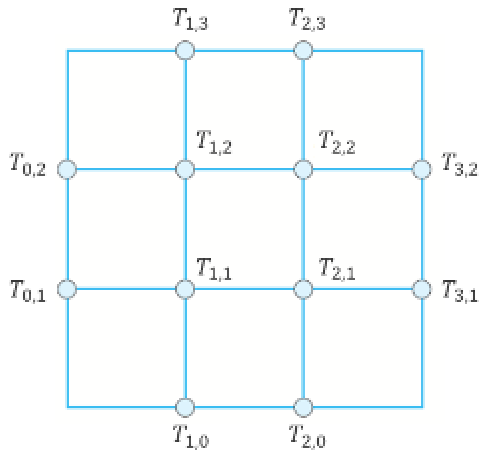


Case 1	*Case 2*
W=0.2 m	W=0.2 m
L_T=8 m	L_T=8 m
L_i=1 m	L_i=1 m
k_i=2 N/m	k_1=1, k_2=2, k_3=3, k_4=4 N/m

Note: The tests cases should be rooted in the physical context of the problem. All problems have some clues you can exploit to generate `assert()` test cases, even if the nature of the problem is related to a course you haven't yet taken.

LA-5: Plate Temperature

A square plate, modeled as a grid of nodes in the diagram below, is heated and is allowed to come to a steady-state temperature.



The temperatures of the exterior nodes are known constants. Each of the interior node temperatures can be found via a finite-difference equation, as follows:

$$0 = \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta^2}$$

Where $T_{i,j}$ is the temperature of the node located at position (i, j) on the plate, and Δ is the spacing between nodes (equal in the vertical and horizontal directions). For example, the temperature of $T_{1,1}$ is:

$$0 = \frac{T_{2,1} - 2T_{1,1} + T_{0,1}}{\Delta^2} + \frac{T_{1,2} - 2T_{1,1} + T_{1,0}}{\Delta^2}$$

which can be simplified to:

$$0 = T_{2,1} - 4T_{1,1} + T_{0,1} + T_{1,2} + T_{1,0}$$

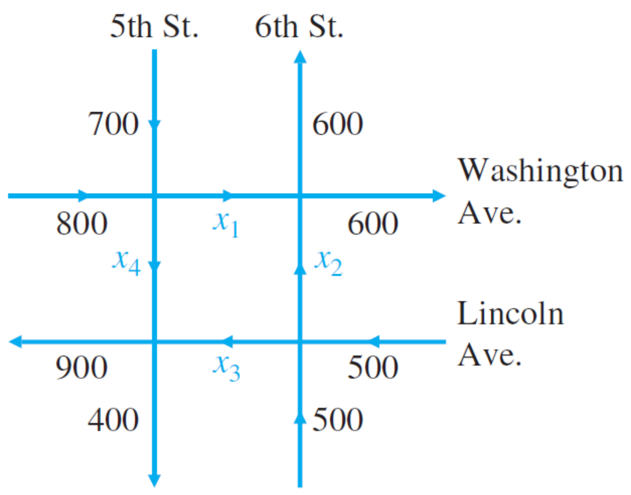
a) Build a complete system of linear equations by repeating the above computation for the remaining three interior nodes.
b) Using the MATLAB `inv()` command, solve the system of equations for the interior temperatures given the exterior temperatures below:

   $T_{1,0} = T_{2,0} = 75\ °C; T_{3,1} = T_{3,2} = 0\ °C; T_{1,3} = T_{2,3} = 25\ °C; T_{0,1} = T_{0,2} = 100\ °C$

c) Download the function `PlateTemperatureHeatMap.m` here. Call it from your script to calculate the mean plate temperature and create a "heat map" plot.

LA-6: Traffic Network

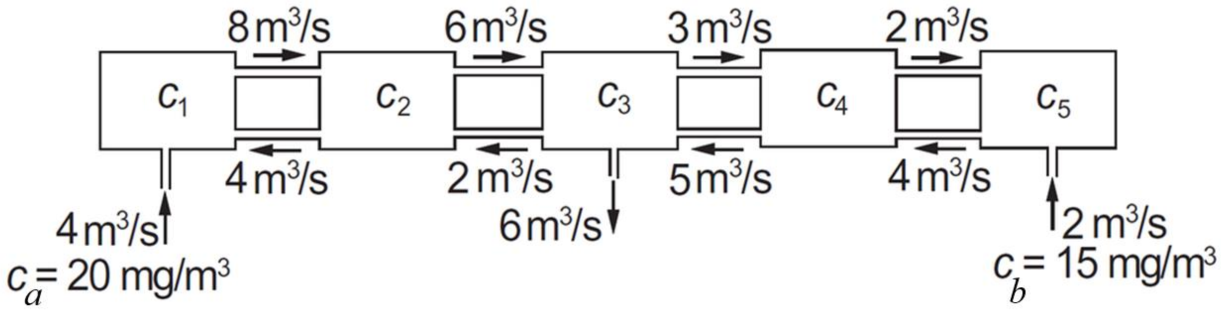
The rush-hour traffic flow for a network of four one-way streets in a city is shown below. The numbers next to each street indicate the number of vehicles per hour that enter and/or leave the network on that street.



a) Determine the system of equations governing this network.
b) What is the maximum number of vehicles per hour that can travel from Washington Ave. to Lincoln Ave. on 5th St.?
c) If traffic lights are adjusted so that 1,000 vehicles per hour travel from Washington Ave. to Lincoln Ave. on 5th St., determine the flow around the rest of the network.

LA-7: Tank Concentrations

Five mixing tanks containing chemicals of unknown concentrations (c_1 to c_5; units of $\frac{mg}{m^3}$) are connected by pipes. Water in the system is pumped through the pipes at the volumetric flowrates shown in the figure (units of $\frac{m^3}{s}$). The fluid entering Tank 1 contains a chemical of concentration c_a as indicated; the fluid entering the system from Tank 5 contains a chemical of concentration c_b.



Conservation of mass can be used to write mass balances for each tank:

$$\Sigma\dot{m}_{in} = \Sigma\dot{m}_{out}$$

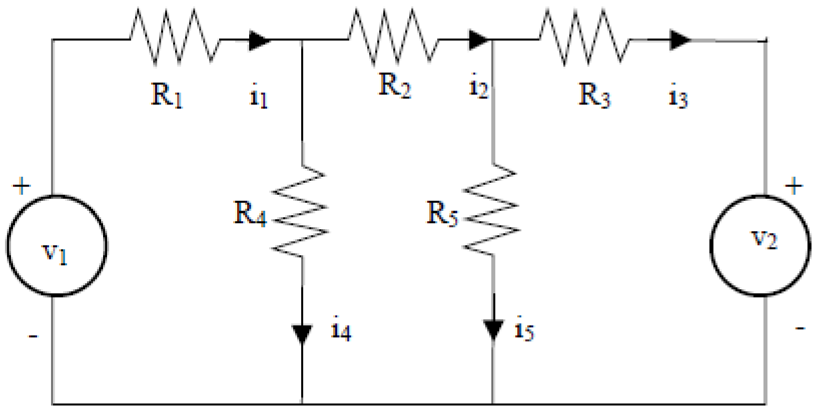
where $\Sigma\dot{m}_{in}$ and $\Sigma\dot{m}_{out}$ are the total inbound and outbound mass flow rates, respectively (units of $\frac{mg}{s}$). Note that the mass flow rate is obtained by multiplying the volumetric flow rate of the water by the concentration. For example, the mass balance for Tank 1 is:

$$4c_a + 4c_2 = 8c_1$$

a) Perform mass balances for the remaining 4 tanks.
b) Place the equations into $Ax = b$ form and solve for the 5 unknown concentrations via the matrix inverse (in MATLAB).
c) Study the effect of c_a and c_b on the system. On one figure, plot the concentrations when c_a ranges from 0 to $30\frac{mg}{m^3}$ (keep $c_b = 15\frac{mg}{m^3}$). On another figure, plot the concentrations when c_b ranges from 0 to $30\frac{mg}{m^3}$ (keep $c_a = 20\frac{mg}{m^3}$). Interpret your results.
d) <u>Without re-solving the system,</u> determine how much will the concentrations in each tank change if c_a is reduced by $5\frac{mg}{m^3}$ and c_b is increased by $10\frac{mg}{m^3}$.

LA-8: Circuit

Consider the circuit shown in the figure.



Parameter (Unit)	Value
R_1 (Ω)	5
R_2 (Ω)	100
R_3 (Ω)	200
R_4 (Ω)	150
R_5 (Ω)	250
v_1 (V)	100
v_2 (V)	100

The equations governing the currents in the circuit are:

$$-v_1 + R_1 i_1 + R_4 i_4 = 0$$

$$-R_4 i_4 + R_2 i_2 + R_5 i_5 = 0$$

$$-R_5 i_5 + R_3 i_3 + v_2 = 0$$

$$i_1 = i_2 + i_4$$

$$i_2 = i_3 + i_5$$

a) Briefly describe the physical system, forcing function, and system response.
b) Symbolically express the system in $Ax = b$ form.
c) Use the matrix inverse to compute the five currents.
d) If each resistor in the circuit is rated to carry a current of no more than 1 Amp, calculate the allowable range of v_2 values, assuming v_1 stays constant.

LA-9: Matrix Norms

Consider the matrix $A = \begin{bmatrix} 1 & b \\ b^2 & 0 \end{bmatrix}$, where b is a positive, nonzero integer. What are the values of b such that $\|A\|_1 > \|A\|_\infty$?

LA-10: Lateral Offset

A study was conducted to determine the correlation between vehicle speed v $\left(\frac{m}{s}\right)$ and lateral offset L (distance from the centerline of the lane, m) in a hairpin turn. Seven vehicles' speeds and lateral offsets are compiled in the `ME2004_LateralOffsetData.mat` dataset (located here).

Upon loading the data, two variables populate the Workspace: `v` (vector containing the vehicles' speeds) and `L` (vector containing the vehicles' lateral offsets).

a) Fit a 0th -order polynomial (straight line) to the data. Don't forget to plot!
b) Compute the R^2.
c) Compute the predicted lateral offset for a vehicle negotiating the turn at $v = 2.35\,\frac{m}{s}$.
d) Repeat (a)-(c) using 1st-order and 2nd-order polynomials.

Analysis

1) What are the units of each curve fit coefficient?
2) Which of the three fits is the best? Why?

LA-11: NASA Challenger

When the space shuttle Challenger exploded in 1986, one of the criticisms made of NASA's decision to launch was in the way they did the analysis of number of O-ring failures versus temperature (O-ring failure caused the explosion). Four O-ring failures would be fatal. NASA had data from 24 previous flights, which is stored in the `ME2004_NASAData.mat` file (located here). This file contains two vectors: `T` (the temperature on the day of a flight, $°F$) and `f` (the number of O-ring failures in that flight).

a) NASA based the decision to launch partially on a chart showing only the flights that had at least one O-ring failure. Find the line that best fits these flights and return the R^2.
b) On the basis of this data, predict the number of O-ring failures when the temperature is $31°F$ (the temperature on the day of that fateful launch), and when the number of failures will exceed four.
c) Repeat (a)-(b) but using the data from all 24 flights.
d) Which do you think is the more accurate method of predicting? An excellent discussion is located here.

LA-12: Bacteria Growth Rate

An investigator has reported the data tabulated below for an experiment to determine the growth rate of bacteria k (per day) as a function of oxygen concentration c $\left(\frac{mg}{L}\right)$. It is known that such data can be modeled by the following equation:

$$k = \frac{k_{max}c^2}{c_s + c^2}$$

where c_s and k_{max} are parameters to be determined.

a) Linearize the equation.
b) Estimate c_s and k_{max} via linear regression (using the `fit()` function and any associated postprocessing commands) given the following data:

$c\ \left(\frac{mg}{L}\right)$	0.5	0.8	1.5	2.5	4
$k\ \left(\frac{1}{day}\right)$	1.1	2.4	5.3	7.6	8.9

c) Predict the growth rate at $c = 2.10\frac{mg}{L}$.
d) Estimate the oxygen concentration at $k = 8.03\frac{1}{day}$.

LA-13: Walther Equation

The viscosity of a fluid represents its resistance to deformation. Examples of highly viscous fluids include oils, corn syrup, honey, molasses, etc. The relationship between fluid viscosity and temperature is an inverse: as temperature increase, fluids become less viscous (thinner) and vice versa. This relationship is quantified by the Walther Equation, which is used as a scientific standard by the American Society for Testing and Materials (ASTM).

Before starting, read the documentation for `readmatrix()` and `text()`.

a) The `temp_viscosity_data.xls` spreadsheet (located here) contains temperatures T (in Kelvin) in the first column and viscosities ν (in "centi-Stokes", cS) in the second column from an experiment. Load this data using `readmatrix()` and produce a plot of ν vs. T.

b) The relationship between ν and T can be approximated by the *Walther Equation*:

$$\log_{10}(\log_{10}(\nu + 0.7)) = A + B\log_{10}(T)$$

where A and B are empirical constants. Note that $\log_{10}(\log_{10}(\nu + 0.7))$ is linear in $\log_{10}(T)$. Plot $\log_{10}(\log_{10}(\nu + 0.7))$ vs. $\log_{10}(T)$ on a separate figure.

c) Using the `fit()` built-in function and necessary postprocessing commands, produce and plot the best-fit line on top of your data from the part (b) plot. Extract the values of the constants A and B along with the R^2 value. Print the A and B values to the Command Window using a neatly formatted `fprintf()` statement similar to the following:

```
 The value of A is A = xxxxxx cS
The value of B is B = xxxxxx cS/K
```

The units of A is cS and B is cS/K. Do not print R^2 to the Command Window.

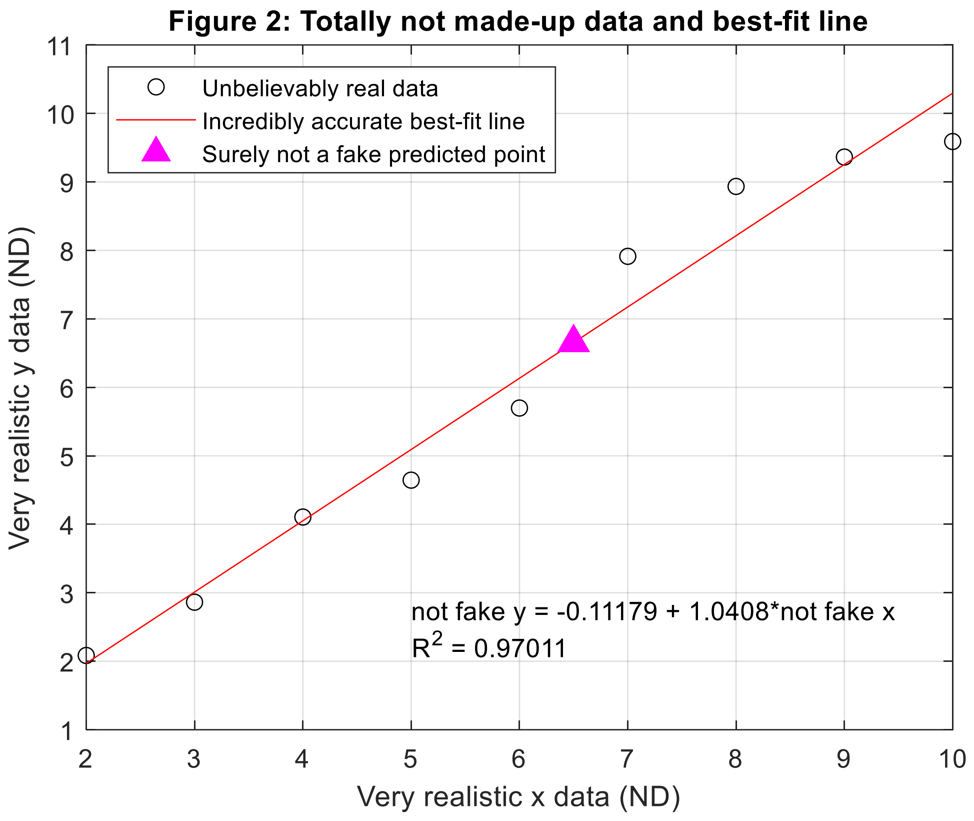
d) Now that you have the best-fit line equation, you can confidently* predict values in between points in the spreadsheet. Predict the viscosity ($\nu_{predict}$) for $T_{predict} = 345\ K$. Overlay this point using a unique color and marker style on your plot from (b) and (c). Use `fprintf()` to print $\nu_{predict}$ to the Command Window:

```
For T_predict = 345 K, v_predict = xxxxxx cS
```

(*How confidently? You'll learn more about error analysis in your statistics class!)

e) Use the `text()` command to annotate your plot with a string containing the Walther equation, including values of the coefficients (do not hard-code in the coefficients!). Repeat with just the R^2 value. You may place your text strings anywhere as long as they don't obscure the data and/or the best-fit line.

Your final second figure should look something like:



The $(T_{predict}, \nu_{predict})$ marker does not have to be filled in like above, but it should be clearly distinguishable from the original data points.

LA-14: Thermodynamic Interpolation

Engineer commonly use standardized tables to look up values necessary for calculations, such as probabilities, material properties, and stress limits. Figure 1, which comes from VT's Thermodynamics (ME 2124) textbook, tabulates some properties of superheated water vapor.

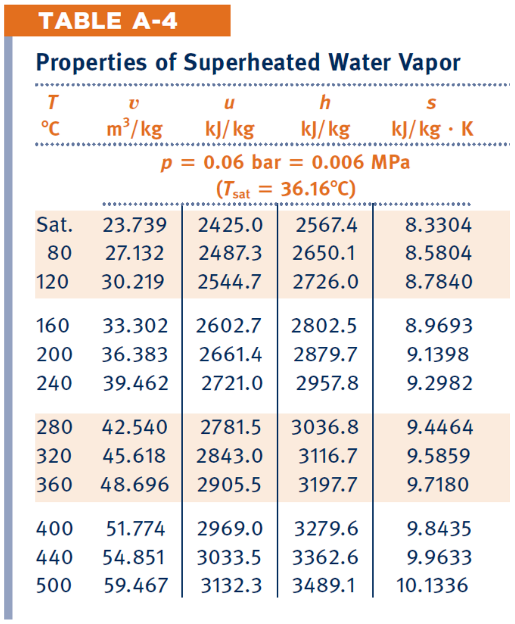


TABLE A-4

Properties of Superheated Water Vapor

T °C	v m^3/kg	u kJ/kg	h kJ/kg	s kJ/kg · K
	$p = 0.06$ bar $= 0.006$ MPa ($T_{sat} = 36.16$°C)			
Sat.	23.739	2425.0	2567.4	8.3304
80	27.132	2487.3	2650.1	8.5804
120	30.219	2544.7	2726.0	8.7840
160	33.302	2602.7	2802.5	8.9693
200	36.383	2661.4	2879.7	9.1398
240	39.462	2721.0	2957.8	9.2982
280	42.540	2781.5	3036.8	9.4464
320	45.618	2843.0	3116.7	9.5859
360	48.696	2905.5	3197.7	9.7180
400	51.774	2969.0	3279.6	9.8435
440	54.851	3033.5	3362.6	9.9633
500	59.467	3132.3	3489.1	10.1336

Figure 1. Superheated water vapor properties.

If we want to look up $v, u, h,$ or s at $T = 500\ °C$, we can simply read across the bottom row because the table happens to list the properties at that exact temperature. However, if we want to want to find properties at $T = 312.4\ °C$, we need to use a technique called *linear interpolation* to estimate the properties because the table doesn't directly contain the properties at our requested temperature.

Linear interpolation consists of "filling in the gaps" between known points. It is a form of polynomial regression. The logic behind linear interpolation is as follows:

1) Identify two known points, A and B. A and B have coordinates (x_A, y_A) and (x_B, y_B), respectively.
2) Fit a straight line through A and B.
3) Determine a point of interest, C, that lies in between A and B. We want to know the properties at this point, but we can't directly look it up because it lies between A and B. C has coordinates (x, y).
4) Because C lies in between A and B, we can use similar triangles and geometry to find the equation of C knowing the slope of the line that connects the three points.

Figure 2 is a graphical representation of interpolation:

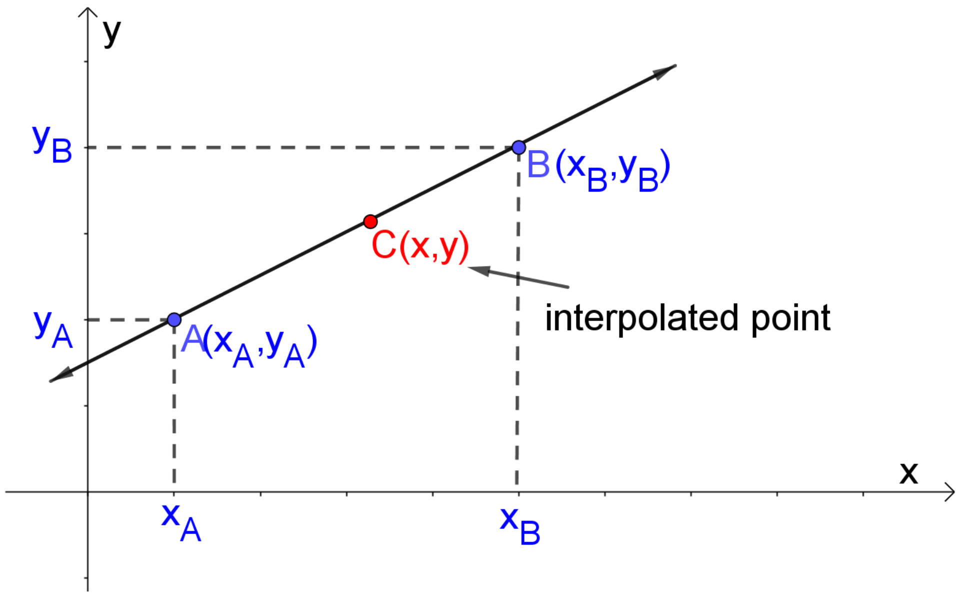


Figure 2. Point C is interpolated by drawing a line that passes through 2 known points, A and B.

a) Using Figure 2 as a reference, derive the formula to find the y-coordinate of Point C (y) given its location (x) and the coordinates of two known points A and B (with coordinates (x_A, y_A) and (x_B, y_B)).

The `ME2004_entropy.mat` file (located here) contains tabulated values of *specific entropy* $s\ \left(\frac{kJ}{kg\ K}\right)$ for air as a function of temperature $T\ (K)$, pulled directly from Figure 1.

b) Load and plot the data.
c) Construct a MATLAB function called `myInterp` with the following definition:

```
function s_interp = myInterp(T,s,T_interp)
```

`T` and `s` are the temperature and specific entropy vectors, `T_interp` is the interpolated temperature, and `s_interp` is the interpolated entropy.

Your function should:

- Implement your equation from (a) to linearly interpolate between (x_A, y_A) and (x_B, y_B). This inherently means you have to determine (x_A, y_A) and (x_B, y_B) from the elements in the `T` and `s` vectors.
- Be able to handle erroneous requests. If x is outside the dataset, your code should terminate without errors but with some suitable message on the command line:

```
Interpolation not found: Program will stop
```

- Graphically indicate the interpolation line used as well as the interpolated entropy:

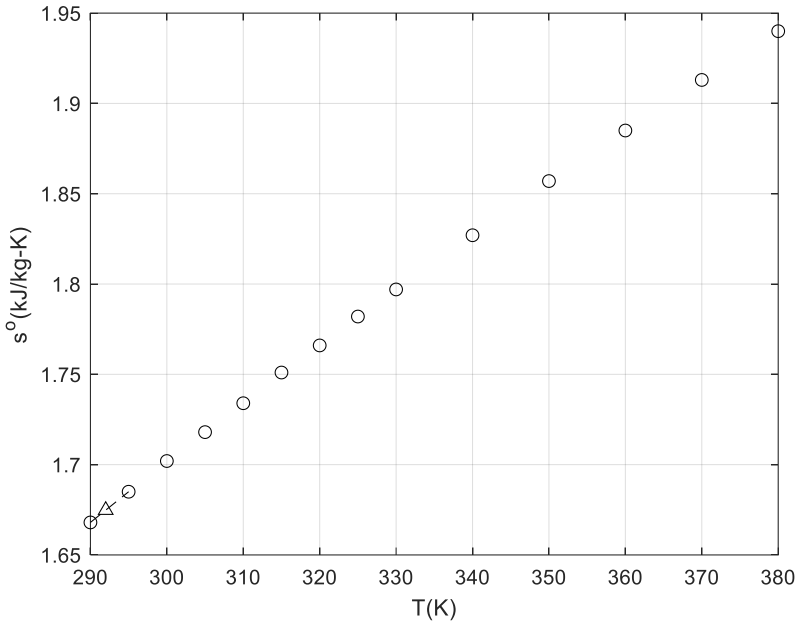
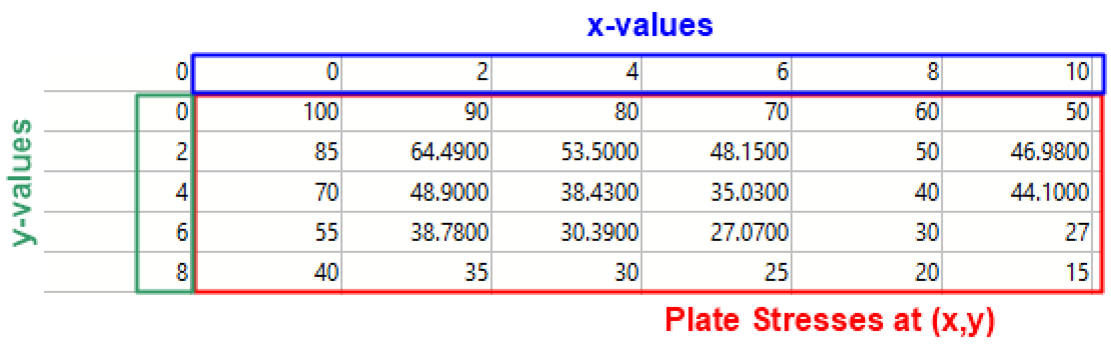


Figure 3. The interpolated entropy (triangle) appended to the plot of the raw data. Your plot may look different.

d) Repeatedly call your function to determine the interpolated specific entropy at $T_{interp} = 302, 375, 414,$ and $575\ K$. Check your answers using the built-in `interp1()` function.

LA-15: Stress Interpolation

A rectangular concrete plate, measuring 10 meters in the x-direction and 8 meters in the y-direction, was subjected to a compression test. The stresses σ (in Megapascals) at equally spaced points (nodes) along the slab are given in the file `ME2004_stressData.mat` (located here). When loaded into MATLAB, you will see the following matrix contained in the `data` variable:



0	0	2	4	6	8	10
0	100	90	80	70	60	50
2	85	64.4900	53.5000	48.1500	50	46.9800
4	70	48.9000	38.4300	35.0300	40	44.1000
6	55	38.7800	30.3900	27.0700	30	27
8	40	35	30	25	20	15

The top row and leftmost column represent various x-distances and y-distances along the plate, respectively. The rest of the entries in the matrix are the plates stresses σ. For example, $\sigma(2,6) = 38.78\ MPa$. There is an extraneous 0 in the upper left corner just to make the dimensions of the grid fully rectangular.

a) Develop a MATLAB function which computes σ given arbitrary x and y coordinates (x_i and y_i).
b) Write a MATLAB driver to call your function to estimate σ for:
    a. $(x_i, y_i) = (4,3.2)$.
    b. $(x_i, y_i) = (2.73,6.18)$.
c) Download the function `StressPlotter.m` here. Call it from your script to generate a 3-D surface plot of the stress data and interpolated point.

You may not use a loop or the `interp2()` function (or any other interpolation function except `interp1()`) in your solution, but you may use it to validate your answers. Think outside the box!

6. Root Finding (RF)

Objective

“Find x” is the quintessential engineering problem. Usually, this involves some algebraic manipulation to arrive at a closed-form solution, i.e., "$x =$ _____". While many engineering problems can be solved with relative ease, some expressions do not have closed-form solutions. In such instances, the only alternative is an approximation.

One method to obtain an approximate solution is to plot the function and determine where it crosses the x-axis. This point, which represents the x value for which $f(x) = 0$, is called the *root*. Although graphical methods are useful for obtaining rough estimates of roots, they are limited because of their lack of precision. Plotting is always encouraged to qualitatively analyze a system, but it is a haphazard and inadequate method for obtaining a precise quantitative answer. Numerical methods represent alternatives that are also approximate but employ systematic strategies to home in on the true root. As elaborated in class, methods such as *Bisection* and *Newton-Raphson* makes the solution of most applied roots-of-equations problems a simple and efficient task.

One peculiar application of root finding is optimization. Recall from calculus that *optimal* solutions can be obtained analytically by determining the value at which the function is flat; that is, where its derivative is 0. Although such analytical solutions are sometimes feasible, most practical optimization problems require numerical, computer solutions. From a numerical standpoint, such optimization methods are similar in spirit to the root-finding methods mentioned above.

RF-1: Root Finding Basics 1

Consider the equation $e^{-x} = x$. Estimate the real root of this equation in the following ways:

a) Graphically by plotting the functions e^{-x} and x on one figure.
b) Graphically by plotting the function $f(x) = e^{-x} - x$ on another figure.
c) Using 3 iterations of the Bisection Method with initial guesses $x_l = 0$ and $x_u = 2$. Fill in the following table:

Iteration	x_l	x_u	x_r	$f(x_l)$	$f(x_u)$	$f(x_r)$	$\|e_a\|$ (%)
1	0	2					-
2							
3							

d) Using 3 iterations of the Bisection Method with initial guesses $x_l = 0.5$ and $x_u = 1$. Fill in the following table:

Iteration	x_l	x_u	x_r	$f(x_l)$	$f(x_u)$	$f(x_r)$	$\|e_a\|$ (%)
1	0.5	1					-
2							
3							

e) Using 3 iterations of the Newton-Raphson Method with initial guess $x_0 = 2$. Fill in the following table:

Iteration	x_{i+1}	$\|e_a\|$ (%)
1		
2		
3		

The purpose of this problem is to ensure you understand the mechanics of each algorithm. Therefore, you may use MATLAB only for plotting and for simple number crunching. Do not use the `bisection_xtol`, `bisection_ftol`, `BisectionMethod`, `NewtonRaphson_xtol`, `NewtonRaphson_ftol`, or `NewtonRaphson` functions (or any derivatives/modifications you made to the functions) in your solution. You may only use them to validate your work.

RF-2: Root Finding Basics 2

Consider the function $f(x) = -14 - 20x + 19x^2 - 3x^3$.

a) Determine every root of $f(x)$ graphically. How many roots are there? Approximately where do they occur?
b) Determine the first root of the function via the Bisection Method.
c) Determine the first root of the function via the Newton-Raphson Method.

For (b) and (c), use your own initial guess(es). Iterate until the relative error falls below 0.001%.

RF-3: Thermal Stress

You are tasked with analyzing the thermal stress in a complex mechanical component subjected to high temperatures. The component is made of a material with nonlinear thermal expansion behavior. You need to determine the critical temperature at which the component will experience a sudden change in stress. The thermal stress in the component can be calculated using the following equation:

$$\sigma(T) = \left(\frac{E}{1-\nu}\right)\left[\alpha\,(T - T_0) - \beta(T - T_0)^2\right]$$

where:

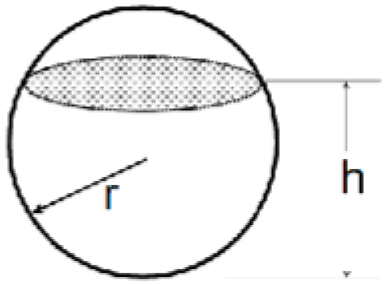
- $\sigma(T)$ = thermal stress in the component at temperature T
- $E = 2e11\,\frac{N}{m^2}$ = Young's modulus of the material
- $\nu = 0.3$ = Poisson ratio
- $(a, \beta) = \left(1.2e-5\,\frac{1}{°C}, 2.8e-9\,\frac{1}{°C^2}\right) =$ material-specific coefficients
- $T_0 = 25\;°C$ = reference temperature.

a) Given $T = 300\;°C$, calculate the thermal stress.
b) Implement the Bisection Method to find the *critical temperature*: the temperature at which the stress is $\sigma_{limit} = 75e7\,\frac{N}{m^2}$. Iterate until the absolute error falls below 0.001.
c) Plot the thermal stress as a function of temperature. Indicate the critical temperature and stress limit on the plot.

RF-4: Water Tank

You are designing a spherical tank to hold water. The volume of liquid in the tank is:

$$V = \frac{\pi h^2 (3R - h)}{3}$$



where:

- $V =$ volume (m^3)
- $h =$ depth of water in the tank (m)
- $R =$ tank radius (m)

a) All root finding methods require at least 1 initial guess. No matter the method, you can elicit some clues as to what the initial guess(es) can be (or more importantly, cannot be) from physical constraints imposed by the underlying physics. What is the range of h values that would physically make sense for an initial guess(es)?

b) Compute h given $R = 3\ m$ and $V = 30\ m^3$ using either the Bisection Method or the Newton-Raphson Method. Iterate until the percent relative error falls below 0.0001%. Supply your own initial guess(es) based on (b).

c) Create a plot of h vs. V for $R = 2, 3,$ and 4 m (place all of the curves on one figure). Briefly comment on your findings.

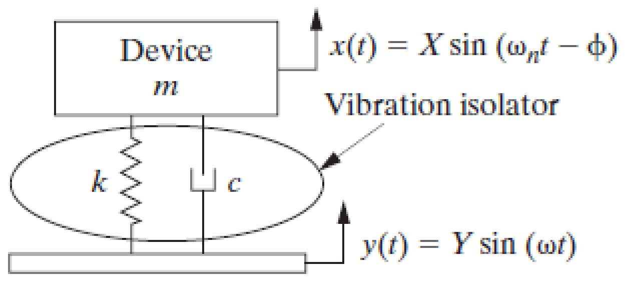
RF-5: Thermal Radiation

Consider a metal plate exposed to the sun with an insulated (no heat flow) bottom surface, as described in Vick Chapter 6.2.3. The surface emissivity is $\varepsilon = 0.9$, the solar absorptivity is $\alpha_s = 0.8$, and the air temperature is $T_\infty = 300\ K$.

a) Determine the plate temperature, T, on a day when $G_s = 900 \frac{W}{m^2}$ and $h = 15 \frac{W}{m^2 K}$.
b) Compute a table of the plate temperature T for values of G_s ranging from 0 to 1200 W/m^2 in increments of 100 when h = 15 W/m^2 K. Plot T vs. G_s.
c) Compute a table of the plate temperature, T, for values of the heat transfer coefficient, h, ranging from 10 to 200 W/m^2 K in increments of 10 when $G_s = 900 \frac{W}{m^2}$. Plot of T vs. h. As h becomes extremely large, what is the plate temperature?
d) Compute $q"_{conv}$ and $q"_{rad}$ for the h values used in (c). Plot these quantities versus h, and comment on your observations.

RF-6: Transmissibility Ratio

An important concept in vibrations and dynamics is mitigating unwanted vibrations via *vibration isolators*. A common single degree of freedom vibration model is a device of mass m mounted to a moving base by a spring of stiffness k and a damper c:



Moving base (source of vibration)

Assuming the base's motion is sinusoidal, its position over time is given by $y(t) = Y\sin(\omega t)$, where Y is the amplitude of the sinusoidal motion and ω is the frequency of the base's oscillation. This induces sinusoidal motion in the device: $x(t) = X\sin(\omega_n t - \phi)$, where X is the device's amplitude, $\omega_n = \sqrt{\frac{k}{m}}$ is the device's natural frequency, and ϕ is a phase shift.

Many parameters can influence the device's vibration. Two important quantities are the damping ratio $\zeta = \frac{c}{2\sqrt{km}}$ and the frequency ratio $r = \frac{\omega}{\omega_n}$. Vibration engineers study how ζ and r affect X and Y. After plenty of physics and algebra, they developed the *Transmissibility Ratio*:

$$TR = \frac{X}{Y} = \sqrt{\frac{1 + (2\zeta r)^2}{(1 - r^2)^2 + (2\zeta r)^2}}$$

The Transmissibility Ratio expresses the ratio of the maximum response magnitude (X) to the input displacement magnitude (Y). It describes how motion is transmitted from the base to the device as a function of r. By varying ω, and consequently r, the amount of motion transferred from the base to the device can be tuned.

A classic example of vibration analysis is in vehicle suspensions. Suppose a car drives over a particularly bumpy road. Assume $m = 100\ kg, c = 30\frac{kg}{s}, k = 2000\frac{N}{m}$, and that the "base" can be modeled as the bumpy road.

a) Compute ζ.
b) What are the units of ζ and r? Recall $1\ N = 1\ kg * \frac{m}{s^2}$.
c) Plot TR versus r.
d) It was found that $Y = 0.02\ m$ through a series of road profile measurements. Determine the minimum allowable ω if X cannot exceed $0.015\ m$.
e) Plot the road profile $y(t)$ from $t = 0\ s$ to $t = 2.5\ s$.
f) How many bumps occur in this timespan?

You conduct a series of experiments using a vehicle with the same $m, c,$ and k as above. Each trial includes multiple parameter constraints:

Trial	**ζ Constraint**	**r Constraint**	**X Constraint**
1	$\zeta = 0.1$	$r < 1$	$0.02 \le X \le 0.05$
2	$\zeta = 0.4$	$r > 1$	$0.006 \le X \le 0.007$
3	$\zeta = 0.7$	$r > 1$	$0.0103 \le X \le 0.011$
4	$\zeta = 1$	$r > 1$	$0.0082 \le X \le 0.0088$

Your proving grounds contains five roads, all of which have $Y = 0.01\ m$:

Road	**Oscillation Frequency ω $\left(\frac{rad}{s}\right)$**
1	7.82
2	3.21
3	9.47
4	8.28
5	5.76

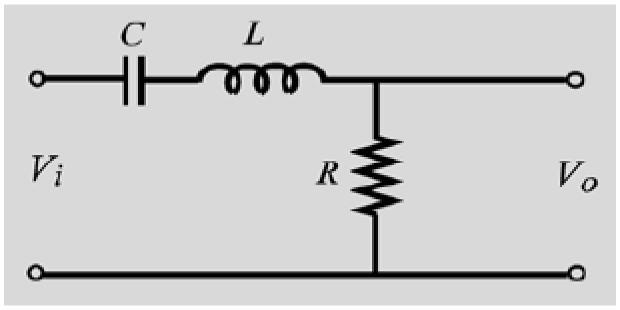
g) Plot TR when ζ ranges from 0.1 to 1 in steps of 0.3 on a single figure.
h) Which road can you use in each trial? Are there any roads which can't be used in any trial? Any there any roads which can be used in multiple trials?

<u>Analysis</u>

1) If $TR > 1$, does the base's motion amplify or attenuate the device's motion?
2) Give an example of when $TR > 1$ is desirable.
3) Give an example of when $TR < 1$ is desirable.

RF-7: Bandpass Filter

Consider the circuit below:

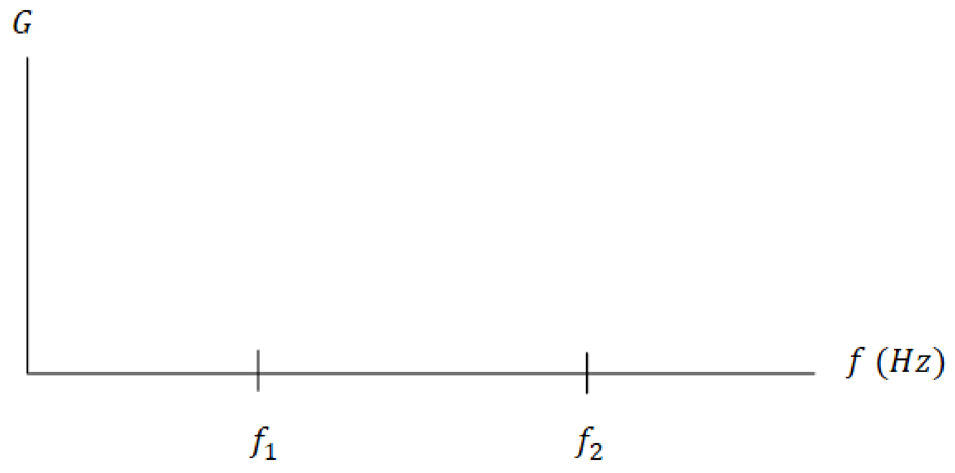


An input voltage source with frequency f (Hz) is applied at V_i. A capacitor C (F), inductor L (H), and resistor R (Ω) are connected in series. The voltage across R (output voltage, V_o) is measured. The *gain*, G, is the ratio of the magnitude of the output voltage to the magnitude of the input voltage:

$$G = \left|\frac{V_o}{V_i}\right| = \frac{(2\pi f)RC}{\sqrt{(1-(2\pi f)^2 LC)^2 + (2\pi f RC)^2}}$$

This circuit represents a *bandpass filter*. A bandpass filter allows input voltage signals with frequencies that are within a certain band, hence the name. There are two main frequencies that define a passband filter: the *low cutoff frequency* f_1 and the *high cutoff frequency* f_2. These frequencies are formally defined as the frequencies where $G = \frac{1}{\sqrt{2}}$. (This is a very prominent number which should be recognizable if you've taken a prior digital circuits, filtering, or signal processing class). If a signal's frequency is below f_1 or above f_2, it will be attenuated (nullified). Bandpass filters are analogous to the Unit Pulse Function, which passes function values within a certain range.

a) Qualitatively sketch G versus f.



A well-known application of bandpass filters is in audio engineering. The analog ("smooth") sound waves captured by microphones often need to be filtered for noise removal, etc. before converting them to digital (discrete) signals for further processing. Tweaking the human voice needs to be done carefully, as changing the frequency alters the pitch. The human voice spans a range of roughly $125\ Hz$ to $8\ kHz$. Different regions of this frequency spectrum fulfill different functions in our speech:

- The *fundamental frequency* is the lowest frequency of any voice signal. It ranges from $125\ Hz$ (adult male) to $300\ Hz$ (child). It conveys a lot of information about the pitch of the voice at any given point and therefore about the overall intonation of speech.
- *Vowels* are found from $500\ Hz$ to $2\ kHz$. Interestingly, each vowel is characterized by its own resonance pattern created by the speaker's mouth and tongue positions.
- *Consonants* occupy the region between $2\ kHz$ and $5\ kHz$. These sounds pass quickly and can help make speech more intelligible. Poor articulation makes it difficult to discern *S*, *T* and *F* sounds.

Suppose you receive a sound sample which you need to filter. You have a capacitor $C = 8\ \mu F$, inductor $L = 11\ mH$, and resistor $R = 1\ k\Omega$.

b) Plot G versus f on a logarithmic scale (use the `semilogx()` function). Compare/contrast with your plot from (a).
c) Estimate f_1 and f_2 from the plot. Numerically confirm your results.

Suppose you receive a sound sample which you need to filter. You have the same L as above, but now you must select C and R. Given the following:

$$f_0 = \sqrt{f_1 f_2}$$

$$\omega_0 = 2\pi f_0$$

$$L = \frac{1}{\omega_0^2 C}$$

$$f_1 = \frac{-\frac{R}{2L} + \sqrt{\left(\frac{R}{2L}\right)^2 + \frac{1}{LC}}}{2\pi}$$

$$f_2 = \frac{\frac{R}{2L} + \sqrt{\left(\frac{R}{2L}\right)^2 + \frac{1}{LC}}}{2\pi}$$

d) What R do you need to filter fundamental frequencies?
e) What R do you need to filter vowels?
f) What R do you need to filter consonants?

For (c)-(e), plot G vs f (with f on a logarithmic scale) on a separate figure.

RF-8: WIND Spacecraft

In 1994, NASA launched the WIND spacecraft. Its primary purpose was to study plasma processes occurring near Earth. The trajectory of the spacecraft was designed such that it would stay at a fixed position along a line from the Earth to the Sun so the solar wind passes around the satellite on its way to Earth:

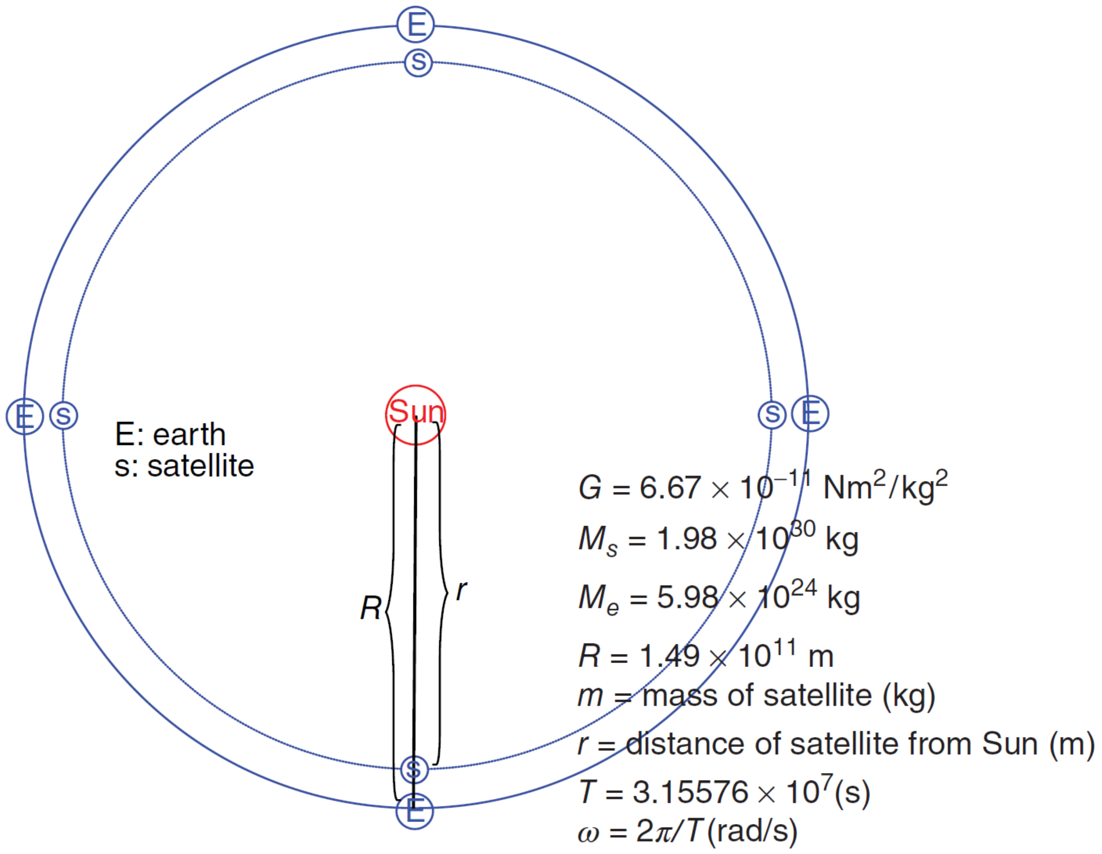

E
s
Sun
E: earth
s: satellite
R
r
$G = 6.67 \times 10^{-11}$ Nm^2/kg^2
$M_s = 1.98 \times 10^{30}$ kg
$M_e = 5.98 \times 10^{24}$ kg
$R = 1.49 \times 10^{11}$ m
m = mass of satellite (kg)
r = distance of satellite from Sun (m)
$T = 3.15576 \times 10^{7}$(s)
$\omega = 2\pi/T$(rad/s)

Fig. 1: WIND satellite trajectory relative to the Earth and Sun.

Newton's Laws of Motion can be used to find the distance of the satellite from Earth $(R - r)$:

$$\frac{GM_s m}{r^2} = \frac{GM_e m}{(R-r)^2} + mr\omega^2$$

$$\rightarrow \boxed{G\left(\frac{M_s}{r^2} - \frac{M_e}{(R-r)^2}\right) - r\omega^2 = 0} \qquad \text{(Eq. 1)}$$

where G is the universal gravitational constant, M_s is the satellite's mass, $M_e$is Earth's mass, R is the distance from the Sun to the Earth, T is the orbital period, and ω is orbital frequency. m and r are defined in Fig. 1. In order to compute $(R - r)$, we first need to compute r. The bulk of this problem focuses on finding r in a variety of ways. One of the major objectives of this problem is to see firsthand how different root finding methods produce different answers and how varying parameters can affect your answers.

Task 0) It should be obvious that we need to employ a root finding method to find r. For this assignment, we will use the *Newton-Raphson Method* and the built-in MATLAB function `fzero()`. Both methods require a user-defined starting guess, r_0. What is the range of r_0 (lowest and highest possible r_0 values) that would be appropriate based on the context of the problem? Why?

Task 1a) We will compute the root in many ways. As you progress through the problem, fill out this table. A brief description of each method is also given in the table.

Method	Description	r	$\varepsilon_{residual}$	%error (to be done in the Analysis section)
1	Baseline. Uses Newton-Raphson			
2	Uses `fzero()` instead of Newton-Raphson			
3	Same as (2) but different r_0			
4	Same as (1) but uses Eq. 2			
5	Same as (2) but uses Eq. 2			

Compute the r for Method 1 via the Newton-Raphson Method. Use Eq. 1 and the parameters from Fig. 1. Use an initial guess of $r_0 = 1e6\ m$ and an *absolute tolerance* (see Eqn. 6.26 in the textbook) with $tol = 1e-4$. Record your results in the above table.

(This r_0 isn't the answer to Task 0)

Task 1b) Compute the *residual error*:

$$\varepsilon_{residual} = |f(r_{Method1})|$$

Record your results in the above table.

Task 2) For Method 2, repeat Method 1 (Task 1) but with `fzero()` instead. Add this line just before calling `fzero()`:

```
options = optimset('TolX',es);
```

where `es` is the tolerance from the previous part. This ensures `fzero()` and your Newton-Raphson Method implementation use the same tolerance. Add `options` as a third argument to `fzero()`:

```
r_method2 = fzero(func,r0,options)
```

Read the `fzero()` documentation if you are unsure of what `options` does. Then, compute the residual error for this root. Don't forget to tabulate your answer.

Task 3) For Method 3, repeat Task 2a but with $r_{0_{new}} = 1e10\ m$. (this also isn't the answer to Task 0.) Make sure to include `options` in your `fzero()` call. Then, compute/tabulate the residual error.

Task 4) Eq. 1 may incur a division-by-zero error, which will derail Newton-Raphson Method. An alternative form is obtained by multiplying both sides by $r^2(R-r)^2$:

$$r^3(R-r)^2\omega^2 - GM_s(R-r)^2 + GM_e r^2 = 0 \qquad \text{(Eq. 2)}$$

For Method 4, compute r via Newton-Raphson with Eq. 2. Use the initial guess r_0 from Methods 1 and 2 (not $r_{0_{new}}$ from Method 3).

Compute/tabulate the residual error **using Eq. 1.** Do NOT use Eq. 2 to compute the residual error; you'll end up with a huge error! Make sure you use Eq. 1.

Task 5) For Method 5, the last method, repeat Method 4 but with `fzero()`. Use `options` like Method 2. Compute/tabulate the residual error **using Eq. 1.**

Task 6) Finally, compute $(R - r)$ using an r of your choice. Why did you select your particular r?

Analysis

1) Which of the Method 1-3 roots has the lowest residual error? How does $r_{0_{new}}$ affect $r_{Method3}$ compared to $r_{Method1}$ and $r_{Method2}$? Do your results make sense in the context of the problem?

2) How do $r_{Method1}$'s and $r_{Method4}$'s residual errors compare? What about $r_{Method5}$ versus $r_{Method2}/r_{Method3}$? Why do you think the changes (or lack thereof) occurred? (Think about what happens to the *shape* of Eq. 2 compared to Eq. 1 and how Newton-Raphson/`fzero()` will respond).

3) You computed r five ways, all of which should be "reasonably close" to each other. Each should be on the order of 10^{11}. Compute the percent error for each r w.r.t (with respect to) to the r with the lowest residual error. Fill in the "%error" column in the table.

4) You should find that every r returns a low percent error. This gives us confidence in our answers, so the next step is to conduct a quick sanity check. Do all of the computed r values fall within the range you specified in Task 0? If there are any outliers, why do they occur?

7. Ordinary Differential Equations (ODE)

Objective

The fundamental laws of physics, mechanics, electricity, and thermodynamics are usually based on empirical observations that explain variations in physical properties and states of systems. Rather than describing the state of physical systems directly, the laws are usually couched in terms of spatial and temporal changes. These laws define mechanisms of change. When combined with continuity laws for energy, mass, or momentum, ordinary differential equations (ODEs) result. Subsequent integration of these differential equations results in mathematical functions that describe the spatial and temporal state of a system in terms of energy, mass, or velocity variations.

Like all functions we have studied in this class, it is often undesirable or impossible to solve ODEs analytically. A variety of numerical solutions have cropped up over the years, with distinct advantages and disadvantages. `ode45()`, MATLAB's hallmark ODE solver, is used in many of these problems.

ODE-1: ODE Sketches

For each of the ODEs:

a) Sketch the phase portrait
b) Find all the fixed points and classify their stability
c) Sketch the anticipated solution

Clearly label your sketches.

Problem	ODE	Note
a	$\frac{d\theta}{dt} = b, b$ is a constant	Draw sketches for $b < 0, b = 0,$ and $b > 0$
b	$\frac{d\theta}{dt} = -\theta$	
c	$\frac{d\theta}{dt} = -a\theta$	Draw sketches corresponding to a zero, a low, and a high value of the parameter a.
d	$\frac{d\theta}{dt} = -a\theta + b$	Draw sketches corresponding to a zero, a low, and a high value of the parameter b.
e	$\frac{d\theta}{dt} = r\theta(1-\theta), r$ is a positive constant	Draw the anticipated solution starting from several initial conditions.
f	$\frac{d\theta}{dt} = r\theta(1-\theta) + S, r$ is a positive constant	Draw the anticipated solution starting from several initial conditions.

ODE-2: Euler's Method

Consider the differential equation:

$$\frac{dy}{dt} = -y + 2$$

a) Sketch the phase portrait and anticipated solution.
b) Manually (by hand) perform 3 iterations of Euler's Method using a step size of $h = 0.5$ and $y_0 = 0$.
c) Using Euler's Method, compute and plot the solution starting from initial conditions: $y_0 = -2{:}\,2{:}\,6$. Put all the curves on a single graph. Experiment with the required step size and time range in order to get accurate and smooth looking plots. Report the step size and time range you decided upon.

ODE-3: Gompertz Law

The growth of cancerous tumors can be modeled by the Gompertz Law:

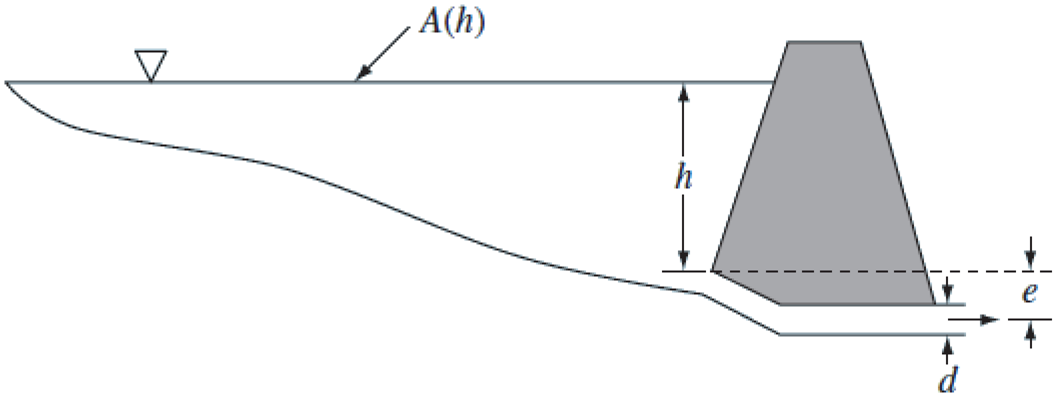
$$\frac{dN}{dt} = -aN\ln(bN)$$

where $N(t)$ is proportional to the number of cells in the tumor, and $a, b > 0$ are parameters.

a) Sketch (by hand) the phase portrait for a low, medium, and high value of a while holding b constant. Then, sketch (by hand) the anticipated solution for various initial values.
b) Verify your hand sketches by generating the phase portrait and anticipated solution in MATLAB via `ode45()` with at least two initial conditions of your choosing. Set $a =$ [1 3 5] and $b = 0.75$.
c) Repeat (a) with a low, medium, and high value of b while holding a constant.
d) Repeat (b). Set $a = 3$ and $b =$ [0.5 1 1.5].
e) Briefly describe how your hand sketches and computer-generated plots compare.

ODE-4: Emptying a Lake

A team of civil engineers installed a dam near a lake.



However, they failed to remove a circular duct underneath the dam. As a result, water leaks out from under the dam. The following differential equation describes how the depth of the lake changes over time:

$$\frac{dh}{dt} = \left(\frac{-1}{A(h)}\right)\left(\frac{\pi d^2 \sqrt{2g(h+e)}}{4}\right)$$

where h is the lake depth (m), t is the time (s), $d = 0.25\ m$ is the diameter of the circular duct (m), $A(h)$ is the lake's surface area as a function of depth (m^2), g is the gravitational constant (m/s^2), and $e = 1\ m$ is the depth of pipe outlet below the lake bottom (m). Furthermore, the following area-depth data is available:

$h, (m)$	6	5	4	3	2	1	0
$A(h), (m^2)$	11700	9700	6700	4500	3200	1800	0

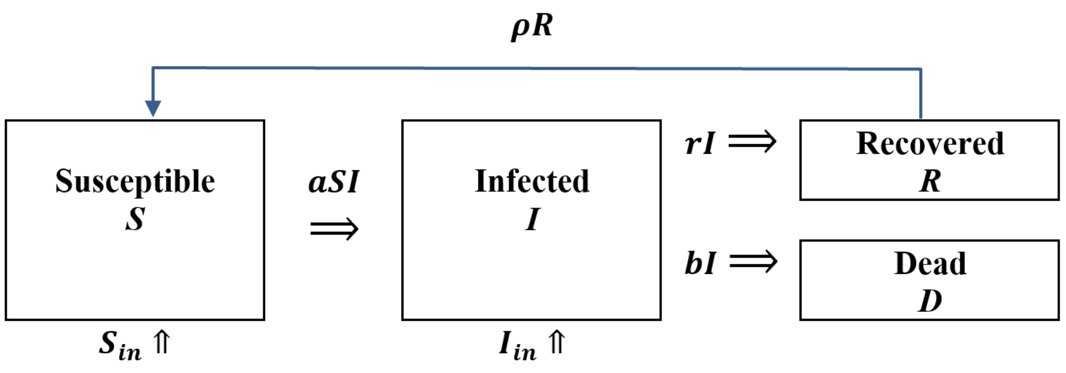
1) Fit a polynomial to the tabulated data. You may use any order polynomial as long as it is not overconstrained/ill-conditioned and $R^2 > 0.9995$.
2) Based on the fitted polynomial, solve the ODE to determine how long it takes for the lake depth to fall under $h_{crit} = 2.5\ m$. Use `ode45()` with an initial condition of $h_0 = 6\ m$ and a time vector with a step size of $1\ s$. Change the absolute and relative tolerances to `1e-6` each. Don't forget to plot!

ODE-5: Epidemic Problem Revisited

Recall the Epidemic problem. Now, consider a new model with two significant improvements:

1) Suppose that after recovery, there is a loss of immunity that causes recovered individuals to become susceptible. This reinfection mechanism can be modeled as ρR, where $\rho =$ the reinfection rate $= \ 0.03 \frac{1}{day}$.
2) Suppose that some of the infected people recover and some die. D is the number of dead people. The death mechanism can be modeled as bI, where $b \ =$ the death rate $=$ $0.15 \frac{1}{day}$. Also consider that there is an influx of susceptible individuals moving to the city at the rate of $S_{in} = 0.2 \ \frac{people}{day}$ and an influx of infected individuals moving to the city at the rate of $I_{in} = 0.3 \ \frac{people}{day}$.

The new model can be represented by:



Modify your model to include these mechanisms. Choose a contextually sensible initial value for D. Explore various values for each parameter and physically interpret your results.

(This is meant to be an open-ended question. There is no single correct answer, per se. Rather, the emphasis is on self-experimentation and seeing what information you can elicit from jockeying around with your updated model.)

ODE-6: Mass-Spring Damper Revisited

Recall the mass-spring-damper system described by the following differential equation and initial conditions:

$$m\ddot{x} + c\dot{x} + kx = u(t)$$
$$x = x_0, \dot{x} = \dot{x}_0 \text{ when } t = 0$$

A fundamental mechanical law is Conservation of Energy:

$$E(t) = KE(t) + PE(t)$$

where $E(t)$ is the total energy of the system, $KE(t) = \frac{1}{2}m\dot{x}^2$ is the mass's kinetic energy, and $PE(t) = \frac{1}{2}kx^2$ is the elastic potential energy of the spring. If a system obeys CoE, then $\frac{dE}{dt} = 0$ always.

In class, we said the damper dissipates energy from the system. This means the mass-spring-damper system does not adhere to CoE, but a mass-spring system (i.e., the special case of the mass-spring-damper system when $c = 0 \frac{N \cdot s}{m}$) does. Let's see if we can verify these claims!

1) On one subplot, plot the system's kinetic, potential, and total energy over time. Simulate long enough to see at least 5 periods. Then, obtain $\frac{dE}{dt}$ and plot it on another subplot.
2) Repeat (1) but for the mass-spring-damper system.
3) Analyze your results. Were you able to verify that a mass-spring system obeys CoE but a mass-spring-damper system does not? Why or why not?

Change the absolute and relative tolerances to `1e-6` each. Use the following parameters:

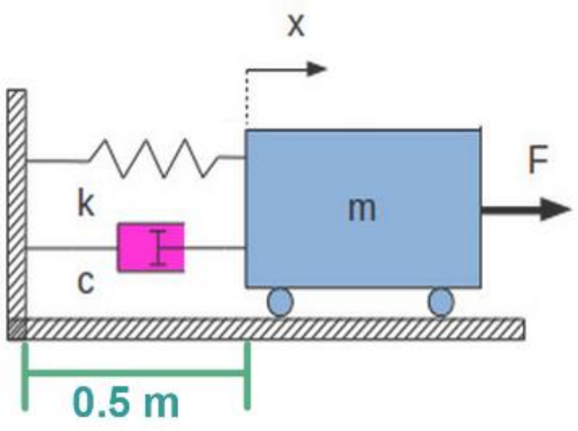
Parameter	$m\ (kg)$	$c\ \left(\frac{N \cdot s}{m}\right)$	$k\ \left(\frac{N}{m}\right)$	$u(t)\ (N)$	$x_0\ \ (m)$	$\dot{x}_0\ \ \left(\frac{m}{s}\right)$
Value	5	1 (when applicable)	3	0	1	0

ODE-7: Mass-Spring-Damper Design Problem Revisited

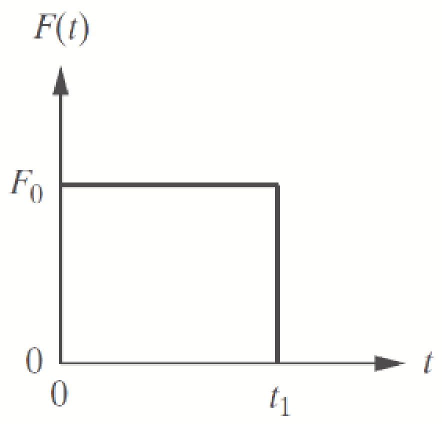
Consider the spring-mass-damper system described by the following differential equation and initial conditions:

$$m\ddot{x} + c\dot{x} + kx = F(t)$$

$$x = x_0, \dot{x} = \dot{x}_0 \text{ when } t = 0$$



The forcing function, $F(t)$, is a rectangular-pulse excitation of magnitude F_0 lasting for t_1 seconds, as illustrated by the figure below:



Find the values of c which satisfy the following requirements:

1. The cart must not hit the left wall, which is located 0.5 m behind its back (left) edge. (Note that the cart's left edge is the point where $x = 0$.)
2. The cart's maximum forward (positive) velocity must exceed $0.72\frac{m}{s}$.
3. $c < 2\sqrt{mk}$

Use `ode45()` and a time vector with a step size of $0.01\ s$. Change the absolute and relative tolerances to `1e-6` each. Use the following parameters:

```
m = 25;                 % Cart's mass [kg] (scalar)
k = 10;                 % Spring stiffness [N/m] (scalar)
F0 = 30;                % Magnitude of the pulsed force [N] (scalar)
t1 = 1.5;               % Duration of the pulsed force [s] (scalar)
IC = [0 0];             % Initial position and velocity (vector).
```

ODE-8: Ingenuity Helicopter

NASA's Jet Propulsion Laboratory (JPL) is designing a new system to control their Mars-based helicopter, Ingenuity.

The vertical motion in time of Ingenuity is governed by the following equations:

$$\frac{dy}{dt} = v$$

$$\frac{dv}{dt} = \frac{8P}{(vm + A)} - \frac{cv^2}{m} - g$$

$$\frac{dQ}{dt} = e = h - y$$

where the dependent variables are: y, the vertical position of the helicopter; v, its vertical velocity, and Q, the integral over time of the vertical position error, e. The vertical position error (e) is the difference between the desired altitude (h) and the vertical position (y), i.e. $e = h - y$. The system parameters are provided below:

- Helicopter mass, $m = 1.8\ kg$
- Motor power, $P = k_p e + k_d \frac{de}{dt} + k_i Q$, where k_p, k_d, and k_i are the known PID controller parameters
- Length of blades, $L = 1.2\ m$
- Atmospheric density, $\rho = 0.02 \frac{kg}{m^3}$
- Lift constant, $A = 2m\sqrt{\frac{mg}{\pi \rho L^2}}$
- Drag constant, $c = 0.0012$
- Acceleration due to gravity on Mars, $g = 3.72 \frac{m}{s^2}$
- Desired helicopter altitude, $h = 18\ m$

a) Develop a MATLAB function that uses `ode45()` to solve this system of ODEs for given values of the PID controller parameters: k_p, k_d, and k_i. Use a time (t) span from 0 to 100 s in steps of 0.1 s. Assume zero initial conditions.
b) Develop a MATLAB driver script that calls your function for the following sets of PID controller parameters:

	k_p	k_d	k_i
Case 1	3	0	0
Case 2	3	0.6	0
Case 3	3	0.6	0.2

c) Graph the solution (y) as a function of t for each case in Part (b). Place all plots on the same figure.
d) Briefly compare the three plots.

Appendix A: Probability Distributions and Monte Carlo Methods

Let's say you are given a coin and are asked to flip it $N = 100$ times. Each flip is independent, i.e., the outcome of the current flip is not affected by the outcome of the previous flip. The results are as follows:

THHTTHTHTTHTHTHTHHHTHTHTTHTTHHTHTTHTHTHTTHHTTHHTTHTTTTHTHTT
HTTHTTHHHTTTTHHTTTHTHHHTTTTHTHHHTTTHHTHTH

If we count the occurrence of heads and tails we obtain:

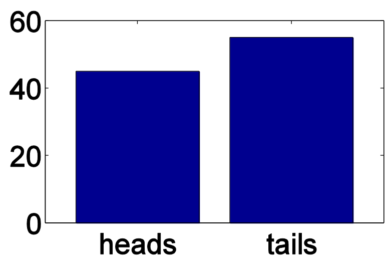
H = 45
T = 55

Based on this knowledge we can estimate the probability of the coin landing as "heads", P(H), and "tails", P(T), as follows:

$$P(H) = \frac{\text{\# of observed heads}}{\text{\# of total trials}} = \frac{45}{100} = 0.45$$

$$P(T) = \frac{\text{\# of observed tails}}{\text{\# of total trials}} = \frac{55}{100} = 0.55$$

Note that P(H) + P(T) = 1, since the coin must land either on "heads" or on "tails"; there is no other possible outcome. Based on this evidence, the possible outcomes are distributed as follows:



According to our results, the coin does not appear to be fair. A biased opinion of coins is that P(H) = P(T) = 0.5, i.e., "heads" and "tails" are equally likely. Given our data, we cannot make such a claim without further evidence, i.e., more flips. In the context of this assignment, increasing `Nhands` (Two Aces problem) or `numPoints` (Pi Problem) is done for this very reason.

This coin can be simulated given the observed occurrences of “heads” and “tails” with the following Monte Carlo simulation in MATLAB:

```
mu = 0.45;  %probability of heads
N = 100; %number of trials
rr = rand(1, N); %random numbers to simulate each trial
outcome = zeros(1, N);
display = 'HT'; %"H" is denoted by 1, "T" is denoted by 2

for ii = 1:length(rr)
    if rr(ii) <= mu
        %The current trial resulted in heads
        outcome(ii) = 1;
    else
        %The current trial resulted in tails
        outcome(ii) = 2;
    end
end

disp(outcome)
```

Output:

```
ans =

HHTTTHTTHHTTTHHHTHHHTTTTTTHTTTTTTHHTHTTHHHTTTHTTTTTTTTHTTTHHHHHH
HTTTTHTHHTTHTTHHTTTHTTHHTTHHHHHTHHHT

>> heads = sum(outcome == 1) % Tally number of heads
heads =
    44

>> tails = sum(outcome == 2) % Tally number of tails
tails =
    56
```