MATLAB for Beginners

The course website is located at http://uiuc-cse.github.io/matlab-fa13/.

Session 1

Example: Area of a circle & volume of a sphere (functions) • Mathematics • 5min

Given the radius r, what is the area A of a circle? The classic formula is written

$$A(r) = \pi r^2$$

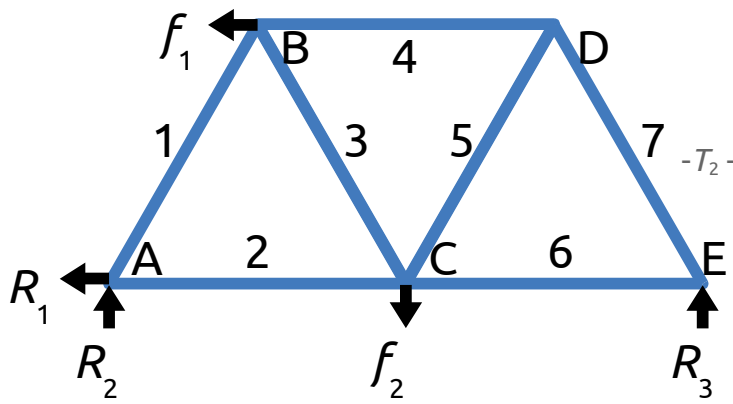
- To make this reproducible, let us create a function to store it. Open a new file and enter the following:

```
function [A] = areaOfCircle(r)
A = pi * r^2;
```

- Save this file as areaOfCircle.m. Test it on a few values. Now try it on an array `1:1:5`. What do you need to change to make this work?
- Now compose a new function, `volOfSphere`, and make it work.

Example: Truss forces (Elementwise & matrix operators) • Mathematics • 5min

A common problem in mechanics is the solution of forces in a truss. This is solved statically by the method of joints, in which you write an equation for each node of the truss and solve the linear set of equations which results.



$$0.5\ T_1 + T_2 = R_1 = f_1$$
$$0.866\ T_1 = -R_2 = -0.433\ f_1 - 0.5\ f_1$$
$$-0.5\ T_1 + 0.5\ T_3 + T_4 = -f_1$$
$$0.866\ T_1 + 0.866\ T_3 = 0$$
$$-T_2 - 0.5\ T_3 + 0.5\ T_5 + T_6 = 0$$
$$0.866\ T_3 + 0.866\ T_5 = f_2$$
$$-T_4 - 0.5\ T_5 + 0.5\ T_7 = 0$$

Let us write the matrix form of this equation as $\mathbf{T}x = f$. You could write the solution to this formally as $x = \mathbf{T}^{-1}f$. While attractive formally, it is often far too expensive to calculate and store an inverse matrix in memory for large problems.

In practice, matrix inversion is a brute-force solution to a linear algebra problem. MATLAB has a number of clever ways to solve matrices built into it. The most frequent is not to invert the matrix, but instead to use what is called *right division*, written $x = \mathbf{T}\backslash f$.

- Let $f_1 = 1000$ and $f_2 = 2000$. Write the governing equations in matrix form.

$$\begin{pmatrix} 0.5 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0.866 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.5 & 0 & 0.5 & 1 & 0 & 0 & 0 \\ 0.866 & 0 & 0.866 & 0 & 0 & 0 & 0 \\ 0 & -1 & -0.5 & 0 & 0.5 & 1 & 0 \\ 0 & 0 & 0.866 & 0 & 0.866 & 0 & 0 \\ 0 & 0 & 0 & -1 & -0.5 & 0 & 0.5 \end{pmatrix} x = \begin{pmatrix} f_1 \\ -0.433 f_1 - 0.5 f_2 \\ -f_1 \\ 0 \\ 0 \\ f_2 \\ 0 \end{pmatrix}$$

- Define the matrix using the MATLAB *New Variable...* interface.
- Write the solution vector and call it *f*.
- Solve the matrix in MATLAB using right division:

```
x = T\f;
```

Example: Falling ballistic object (vectorization, control flow) • Physics • 15min

You are familiar with the equation for an object moving in a gravitational field (a.k.a. falling). The formula for vertical position y as a function of time t may be written

$$y(t) = y_0 + v\,t + a\,t^2$$

where y_0 is the initial position (typically ground level, $y_0 = 0$); v is the initial y-velocity; and a is the acceleration (in this case due to gravity, a = -9.8 m/s^2). The classic ballistics example would be an object fired into the air such that a bullet or shell has a high initial velocity. Let's update this by using a railgun, with a projectile velocity of 2.52km/s.

- We will first calculate the values directly.

```
a=-9.8; %m/s^2
v=2520; %m/s
x0=0;
t=1;
y=a*t^2+v*t+x0;
```

- Now let's vectorize the formula, or allow it to work on many values simultaneously (or at least, in one statement).

```
t=linspace(0,5,101);
y=a*t^2+v*t+x0;      % comment on this error or ask for input
y=a*t.^2+v*t+x0;
```

- Plot the vectorized formula: `plot(t,y);`
- Finally, abstract the formula out into its own function. This which will allow us to reuse it over and over again as necessary. Open a new blank file and write the function definition: `function y = calc(t,v,x0)`
- Test the function:
- Modify the function to not yield y-values less than zero. (if statement, max to find max)

Example: Dam rate of flow (plotting, vectorization) • Civil Eng. • 10min

One equation for the volume rate of flow of water Q over the spillway of a dam is the formula

$$Q = C\sqrt{2g}\,B\left(H + \frac{v^2}{2g}\right)^{3/2}$$

where C is the discharge coefficient; B is the spillway width; and H is the depth of water passing over the spillway.

- Open the file `computedamrateofflow.m`. This has skeleton code for a function you will define.
- Define a discharge coefficient of 1.946 and the acceleration due to gravity in SI units.
- Calculate the value of Q. This should be in the form `Q = <something>`.
- Plot your function for B = 1m, v = 0.536m/s (the flow rate of the Mississippi river) against a vector H ranging from 1 to 10 at increments of 0.1.
- What would you change to make this also a function of discharge coefficient C?

Example: Fahrenheit/Celsius (Scripts, file operations) • Mathematics • 25min

You are familiar with the conversion formula from Fahrenheit to Celsius:

$$T_F(T_C) = T_C \frac{180}{100} + 32$$

- Write a function which performs this conversion and save it in an appropriate file.

```
function Tf = TempC2F(Tc)
... your code here ...
```

- Write a script templist.m which generates a table of conversions for the range 0℃ to 200℃ and then writes that table to a file templist.txt.

```
cList = linspace(0,200,101);
fList = TempC2F(cList);

fileID = fopen('templist.txt','w');
fprintf(fileID,'%6s %12s\n','Tc','Tf');
fprintf(fileID,'%6.2f %6.2f\n',cList,fList);
fclose(fileID);
```

- Examine templist.txt. Is the output what you expect? Change the output line to the following:

```
tList = [cList; fList];
fprintf(fileID,'%6.2f %6.2f\n',tList);
```

(Note that the row-major order of MATLAB translates into a transposed column-major order in the output.)

- Next, let's read the data in and plot them.

```
tData = importdata('templist.txt')
plot(tData.data(:,1),tData.data(:,2))
```