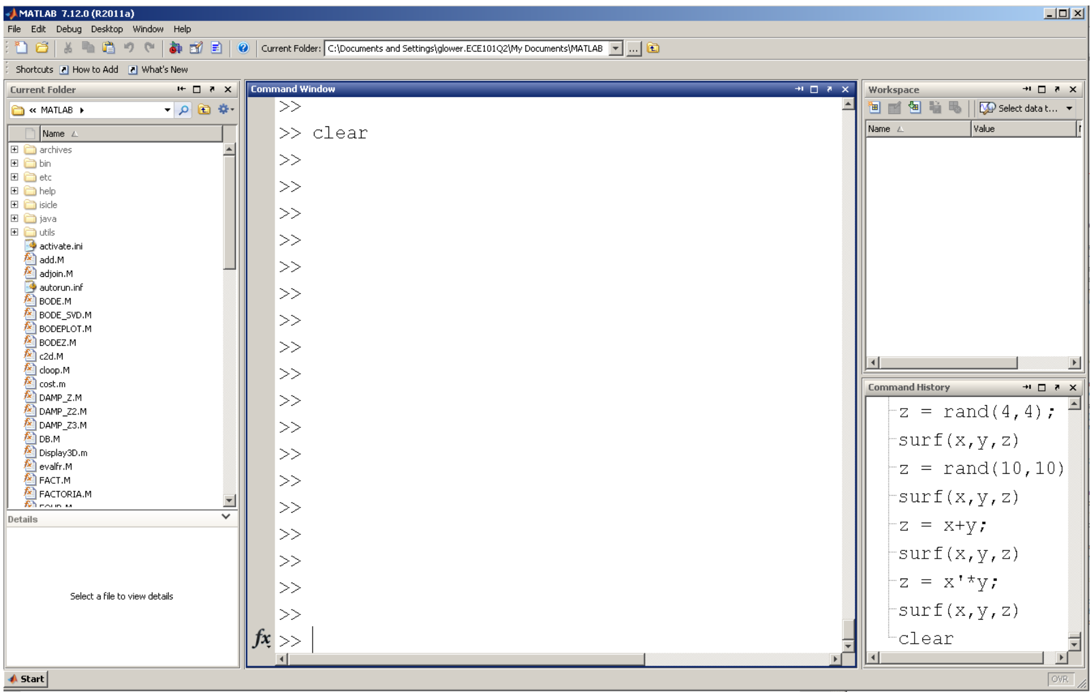
Matlab Review

Becoming familiar with MATLAB

- The console
- The editor
- The graphics windows
- The help menu
- Saving your data (diary)

General environment and the console



Simple numerical calculations

```
>> x = 17/3

    5.6667

>> y = (3+4)*5

    35
```

Particular numbers

```
>> e = exp(1)

    2.7183

>> pi

    3.1416
```

```
>> i

        0 + 1.0000i

>> j

        0 + 1.0000i
```

Do and don't display results

```
>> x = 2*pi

    6.2832

>> x = 2*pi;
```

Displaying number of decimal places

```
>> format short
>> pi

    3.1416

>> format long
>> pi

   3.141592653589793

>> format longe
>> pi^30

    8.212893304027486e+014

>> format shorteng
>> pi^30

   821.2893e+012
```

Matrices

```
[  start of matrix
]  end of matrix
,  next element
;  next row


>> A = [1,2,3]

    1     2     3


>> B = [1,2,3;4,5,6]

     1     2     3
     4     5     6

>> C = A'

     1
     2
     3
```

```
>> D = zeros(1,3)

     0     0     0
```

Random Numbers: Uniform distribution from (0, 1)

```
>> rand(2,4)

    0.8147    0.1270    0.6324    0.2785
    0.9058    0.9134    0.0975    0.5469
```

Normal distrubution: N(0,1)

```
>> randn(2,4)

    3.5784   -1.3499    0.7254    0.7147
    2.7694    3.0349   -0.0631   -0.2050
```

for loops:

```
>> x = zeros(1,5);
>> for i=1:5
   x(i) = i*i;
   end
>> x

x =

     1     4     9    16    25

while

if

if else end
```

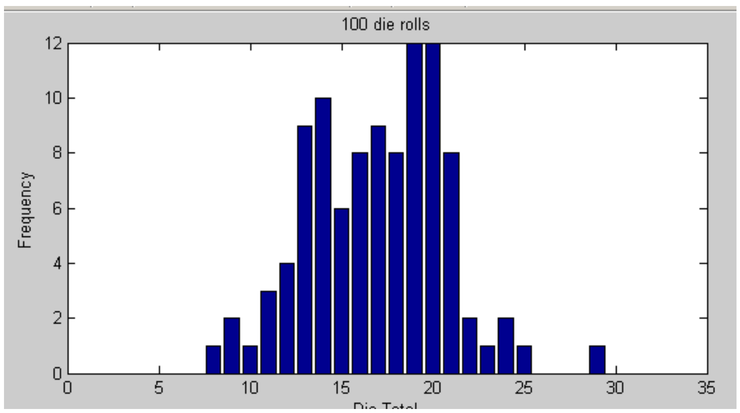
Example: Roll five 6-sided dice

```
>> x = [1:5]

     1     2     3     4     5

>> rand(1,5)

    0.7298    0.8908    0.9823    0.7690    0.5814

>> dice = ceil( 6*rand(1,5) )

     3     6     2     4     4

>> dice = ceil( 6*rand(1,5) )

     1     4     3     1     3

>> sum( ceil( 6*rand(1,5) ) )

     8

>> sum( ceil( 6*rand(1,5) ) )
```

```
    16
```

Roll 5d6 one-hundred times and record how many rolls you get for each total:

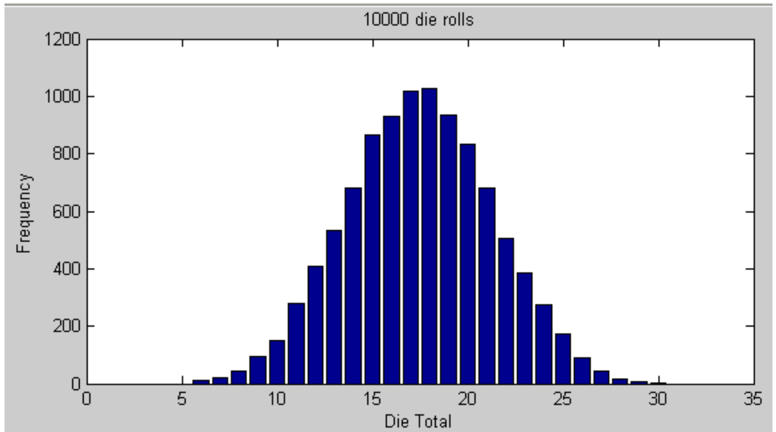
```
>> X = zeros(30,1);
>> for i=1:100
   D = sum( ceil( 6*rand(1,5) ) );
   X(D) = X(D) + 1;
   end
>> bar(X)
>> >> xlabel('Die Total');
>> ylabel('Frequency');
>> title('100 die rolls')
```



Result from rolling 5d6 100 times

Roll 5d6 10,000 times and record the frequency of each outcome:

```
>> X = zeros(30,1);
>> for i=1:10000
   D = sum( ceil( 6*rand(1,5) ) );
   X(D) = X(D) + 1;
   end
>> bar(X)
>> xlabel('Die Total');
>> ylabel('Frequency');
>> title('10000 die rolls')
```



Result of rolling 5d6 10,000 times

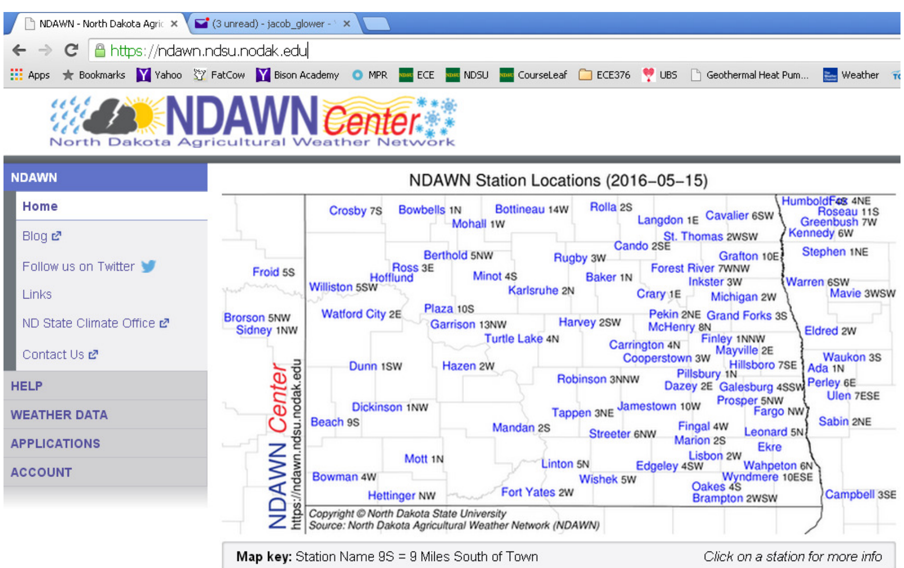
Numerical Integration:

Simplest (and least accurate) is Euler integration

Area = Width * height

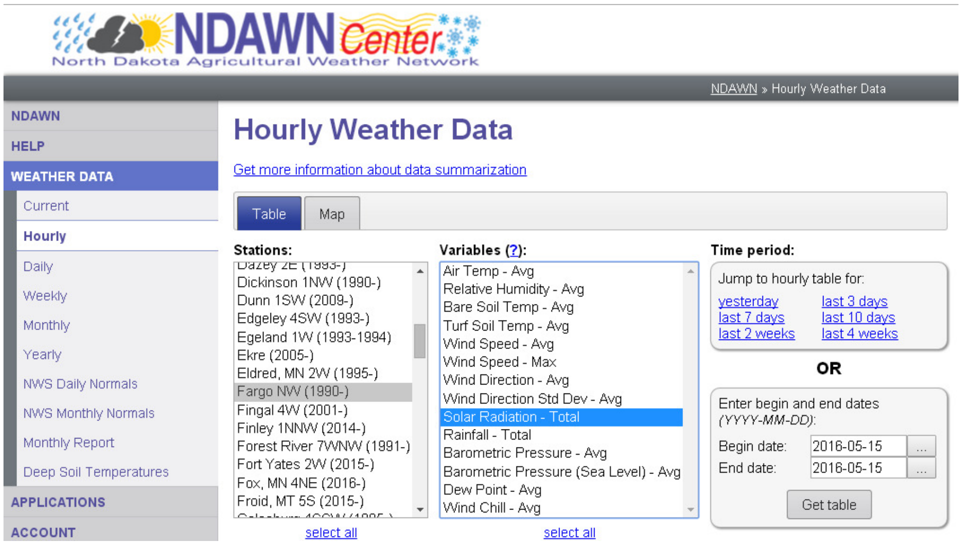
Example: Determine how much energy a 1.5m2 solar panel will generate in Fargo, ND over the past two weeks. Assume the efficiency of the solar panel is 20%

Solution: Get solar data from NDAWN:



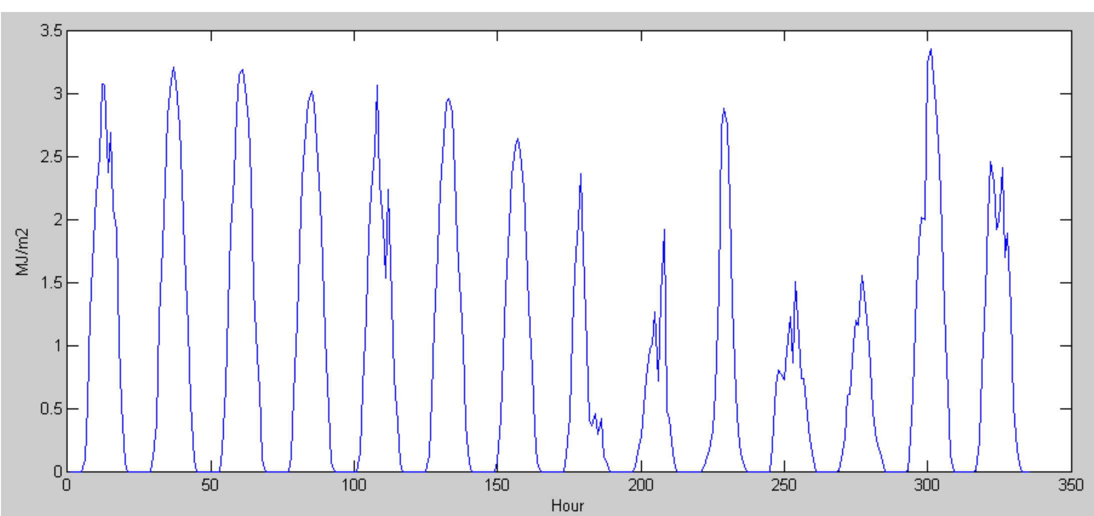
https://ndawn.ndsu.nodak.edu/

Select Weather Data - Hourly - Fargo - Solar Total



Export to a CVS file and copy the data to the clip board. From Matlab

```
>> Sun = [
     paste in the data
   ];

>> size(Sun)

   336      1

>> h = [1:336]';
>> plot(h,Sun);
>> xlabel('Hour')
>> ylabel('MJ/m2')
>>
```



Solar Radiation in Fargo for the last two weeks

This is hourly data. To convert to Joules, integrate

height =data * 1,000,000 (MJ total over an hour)

width = 1 hour

Area = Width * height = Joules

```
>> MJ = sum(Sun)

MJ =

  280.2130
```

To convert that to kWh

$$1MJ = 0.2778\ kWh$$

```
>> kWh = MJ * 0.2778

kWh =

   77.8432
```

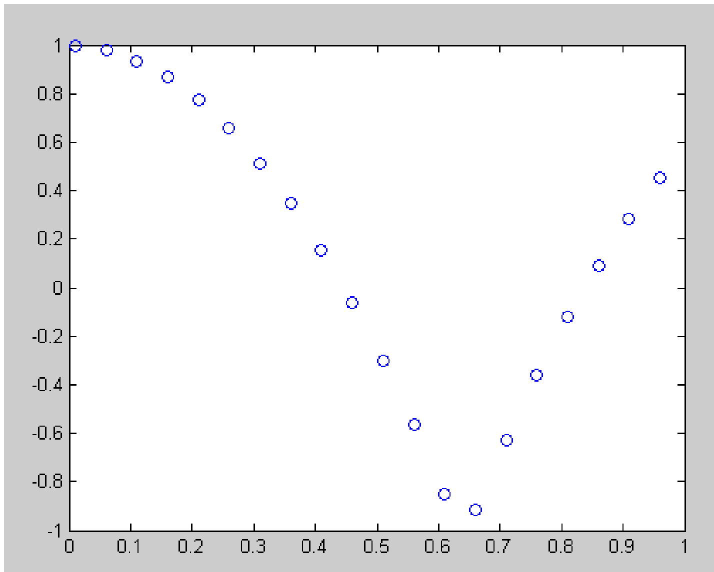
At 20% efficiency, a solar panel would generate 15.5kWh over this 2 week span. This is worth about $1.55

```
>> kWh * 0.2

   15.5686
```

Bouncing Ball

```
x = 0;
y = 1;

dx = 1;
dy = 0;

ddx = 0;
ddy = 0;

dt = 0.01;

for i=1:1000
   ddy = -9.8;

   dx = dx + ddx*dt;
   dy = dy + ddy*dt;

   x = x + dx*dt;
   y = y + dy*dt;

   if (x > 1)
      dx = -abs(dx);
      end
   if (x < -1)
      dx = abs(dx);
      end
   if (y < -1)
```

```
      dy = abs(dy);
      end

   hold off
   plot([-1,1],[-1,1],'.');
   hold on
   plot(x,y,'o')
   pause(0.01);
   end
```



Bouncing Ball

Vectors, Dot Products, and Cross Products

Vectors

Vectors in 3-space are represented with a 4x1 matrix:

$$v = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

The point in 3-space is [x, y, z]'.

Zoom is the scaling factor:

- 0 vector at infinity (the size of the image is zero when you're infinitely far away)
- 1 scale = 1 (normal scaling)
- 2 Zoom in 2x

Magnitude

The magnitude of a vector is

$$|a| = \begin{vmatrix} a_x \\ a_y \\ a_z \end{vmatrix} = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

Dot product

A dot producto is a scalar: the length of vector a projected on vector b

$$a \cdot b = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \cdot \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = a_x b_x + a_y b_y + a_z b_z$$

Cross Product

A cross product is a vector

- The direction is perpindiculat to vector a and b
- The magnitude is a measure of how orthogonal the vector are

$$a \times b = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \times \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{vmatrix} i_x & i_y & i_z \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}$$

$$a \times b = \begin{bmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{bmatrix}$$

Planes

A plane is a row vector

P = [px, py, pz, pw]

The dot product of a plane and a point is the distance from the plane to the point. A dot-product of zero means the point is on the plane.

A positive dot-product indicates you're above the plane, a negative dot-product indicates the point is below the plane.

Transformation Matricies

A transform matrix is a way to

- Shift a point by the vector (x, y, z)
- Rotate the coordinate frame, and
- Zoom in and out with a scaling factor of w.

Since each point is defined by a 4x1 vector, the transformation matrix needs to be a 4x4 matrix:

$$a_{4x1} = T_{4x4} b_{4x1}$$

T is composed of three parts:

- A 3x3 rotation matrix (identity in this example)
- A 3x1 translation matrix ([bx, by, bz]T)
- A 1x1 scalar (w) defining the zoom in / zoom out factor.

$$\begin{bmatrix} a_x \\ a_y \\ a_z \\ \cdots \\ a_w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \vdots & x \\ 0 & 1 & 0 & \vdots & y \\ 0 & 0 & 1 & \vdots & z \\ \cdots & \cdots & \cdots & & \cdots \\ 0 & 0 & 0 & \vdots & w \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \\ \cdots \\ b_w \end{bmatrix}$$

Example 1: Shift the point [1,2,3] by [x, y, z] Use a scaling factor of one (w=1).

$$b = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

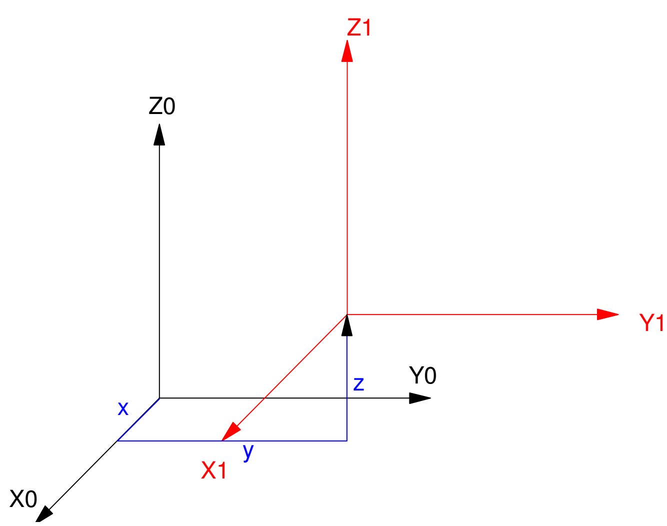
$$a = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1+x \\ 2+y \\ 3+z \\ 1 \end{bmatrix}$$

Point b has been shifted by [x,y,z].

Zoom in with a scaling factor of 2

$$a = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 2 \end{bmatrix}$$

This means if you plot the point (1,2,3), it will be doubled (zoomed in with a factor of 2)



$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

Matlab Commands

Analysis

- sqrt(x) square root of x
- log(x) log base e
- log10(x) log base 10
- exp(x) e^x
- exp10(x) 10^x
- abs(x) |x|
- round(x) round to the nearest integer
- floor(x) round down (integer value of x)
- ceil(x) round up to the next integer
- real(x) real part of a complex number
- imag(x) imaginary part of a complex number
- abs(x) absolute value of x, magnitude of a complex number
- angle(x) angle of a complex number (answer in radians)
- unwrap(x) remove the discontinuity at pi (180 degrees) for a vector of angles

Polynomials

- poly(x)
- roots(x)
- conv(x,y)

Trig Functions

- sin(x) sin(x) where x is in radians
- cos(x) cos()
- tan(x) tan()
- asin(x) arcsin(x)
- acos(x) arccos(x)
- atan(x) arctan(x)
- atan2(y,x) angle to a point (x,y)

Probability and Statistics

- factorial(x) (x-1)!
- gamma(x) x!
- rand(n,m) create an nxm matrix of random numbers between 0 and 1
- randn(n,m) create an nxm matrix of random numbers with a normal distribution
- sum(x) sum the columns of x
- prod(x) multiply the columns of x
- sort(x) sort the columns of x from smallest to largest
- length(x) return the dimensions of x
- mean(x) mean (average) of the columns of x
- std() standard deviation of the columns of x

Display Functions

- plot(x) plot x vs sample number
- plot(x,y) plot x vs. y
- semilogx(x,y) log(x) vs y
- semilogy(x,y) x vs log(y)

- loglog(x,y) log(x) vs log(y)
- mesh(x) 3d plot where the height is the value at x(a,b)
- contour(x) contour plot
- bar(x,y) draw a bar graph
- xlabel('time') label the x axis with the word 'time'
- ylabel() label the y axis
- title() put a title on the plot
- grid() draw the grid lines

Useful Commands

- hold on don't erase the current graph
- hold off do erase the current graph
- diary create a text file to save whatever goes to the screen
- linepace(a, b, n) create a 1xn array starting at a, increment by b
- logspace(a,b,n) create a 1xn array starting at 10^a going to 10^b, spaced logarithmically
- subplot() create several plots on the same screen
- disp('hello') display the message *hello*

Utilities

- format set the display format
- zeros(n,m) create an nxm matrix of zeros
- eye(n,m)create an nxm matrix with ones on the diagonal
- ones(n,m) create an nxm matrix of ones
- help help using different functions
- pause(x) pause x seconds (can be a fraction). Show the graph as well
- clock the present time
- etime the difference between to times
- tic start a stopwatch
- toc the number of seconds since tic