Introduction to MATLAB

Kamesh Subbarao
Associate Professor
Department of Mechanical & Aerospace Engineering
The University of Texas at Arlington, Arlington, TX 76019
subbarao@uta.edu
Off: (817) 272 7467

MATLAB

- Founded in 1984 by Jack Little (President) and Dr Cleve Moler (Chief Scientist) in Natick, MA
- Before joining The Mathworks Inc, Dr Cleve Moler was a professor in University of New Mexico's Mathematics department. He conceptualized MATLAB in Fortran

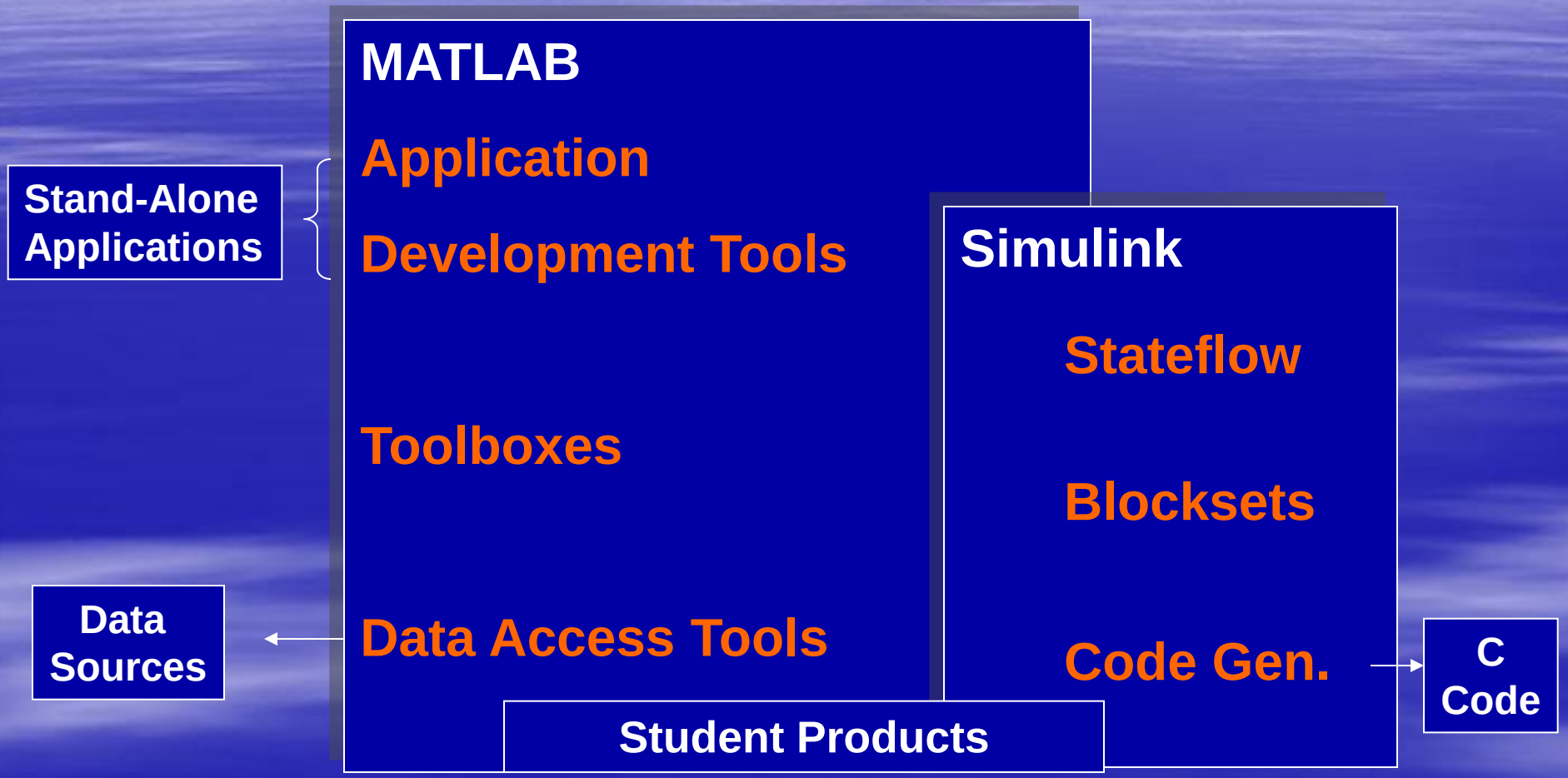
(author of "*Nineteen Dubious Ways to Compute the Exponential of a Matrix*" – Last visited 09/26/2007)

- Product Website www.mathworks.com
- Complete list of products (*Concept of Toolboxes*)

http://www.mathworks.com/products/product_listing/index.html

- From the product website, *"…Our two core products are MATLAB®, used for performing mathematical calculations, analyzing and visualizing data, and writing new software programs; and Simulink®, used for modeling and simulating complex dynamic systems, such as a vehicle's automatic transmission system…"*
- Additional web-based resources to learn MATLAB:
  - *http://www.mathworks.com/matlabcentral/*
  - *http://www.mathworks.com/academia/student_center/index.html*
  - Peter Acklam's (MATLAB Guru) resource on Matrix Manipulation Tricks: *http://home.online.no/~pjacklam/matlab/doc/mtt/doc/mtt.pdf*

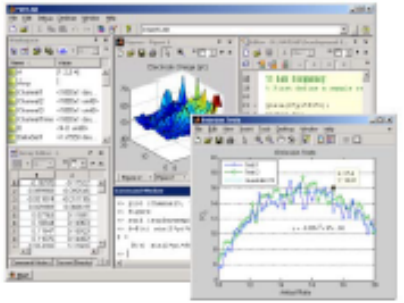
The MATLAB Product Family



What is MATLAB

- High level language for technical computing
  - Interprets. Uses high level commands to pass down arguments to the underlying Numerical Engines.
- Stands for **MAT**rix **LAB**oratory
- Everything is a matrix - easy to do linear algebra

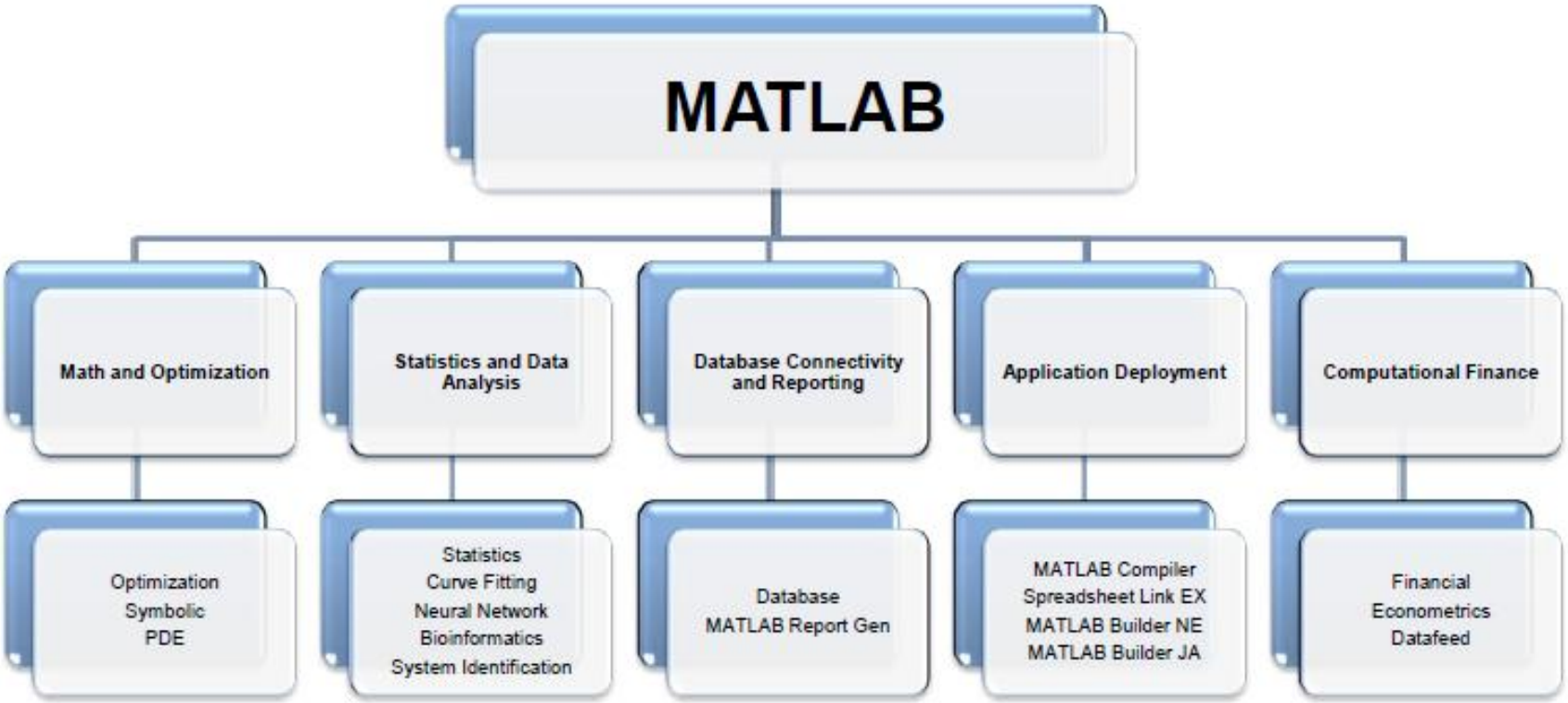
MATLAB
The Language for Technical Computing



- Key Features
  - High-level language of technical computing
  - Development environment for engineers, scientists
  - Interactive tools for design, problem solving
  - Mathematical function libraries
  - Graphics and data visualization tools
  - Custom GUIs
  - External Interfaces: C, C++, Fortran, Java, COM, Excel, .NET

With permission from Todd Atkins, The MathWorks Inc.

MATLAB Product Family

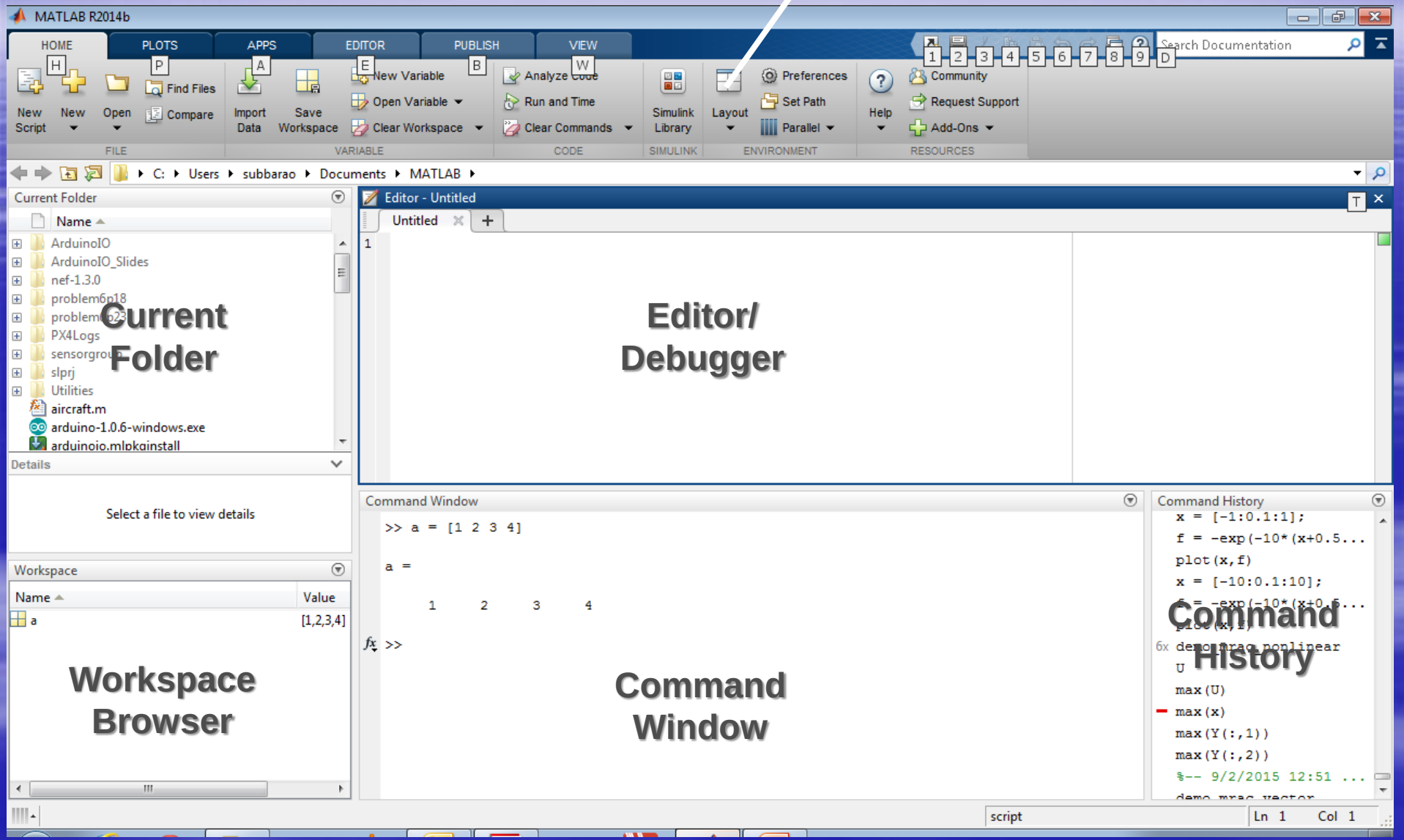


With permission from Todd Atkins, The MathWorks Inc.

Outline

- MATLAB Desktop
- Computing in MATLAB
- Problem Solving with MATLAB
- Working with data in MATLAB
- Programming in MATLAB

Desktop Layout – 2014b



- Clear workspace using >> clear
- Clear Screen using >> clc
- Close all open figures using >> close all;
- Close specific figure window using >> close(#)
- Create some variables
- t = 0:0.1:10;
- y = sin(t);
- z = cos(t);
- Plot y vs t, z vs t, z vs y

Getting HELP in MATLAB

>> doc (Opens product documentation page)
>> doc “cmd” (Opens the page for a specific command)
Example:

```
>> doc plot
```

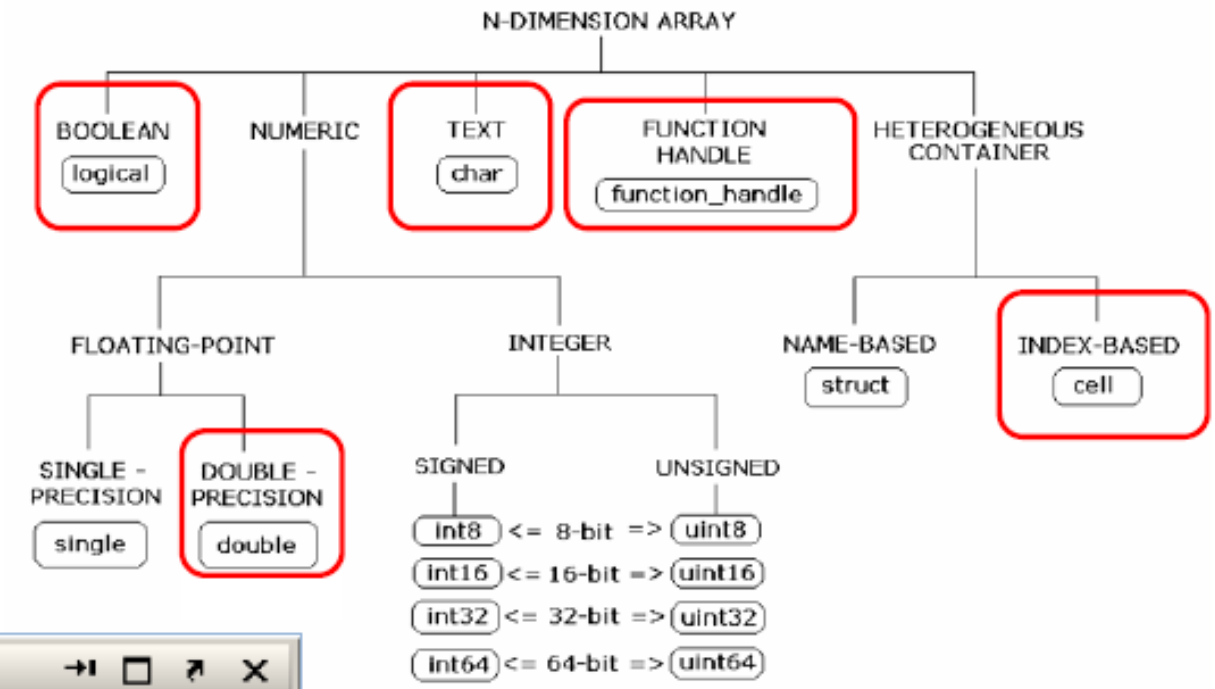
>> help “cmd” (provides command line help – limited)

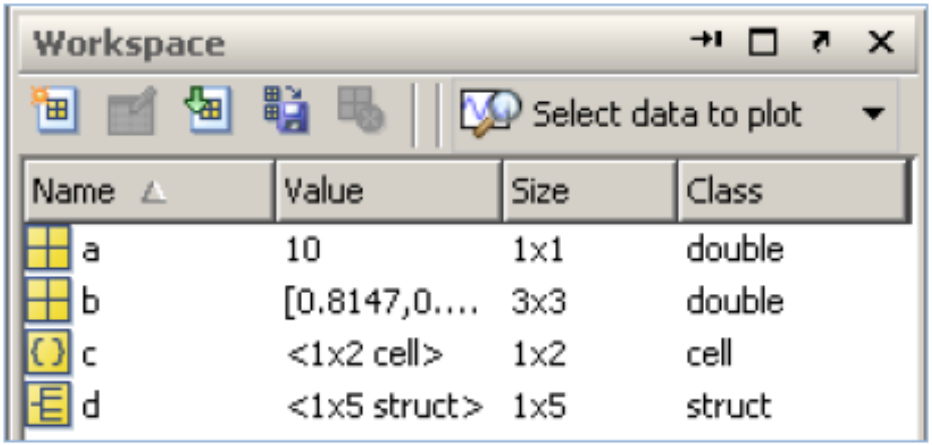
Some commands (functions) are common to several toolboxes and are over-loaded functions.

http://www.mathworks.com/help/index.html

Basics: Data and Variables

- Class
- Size
- Value

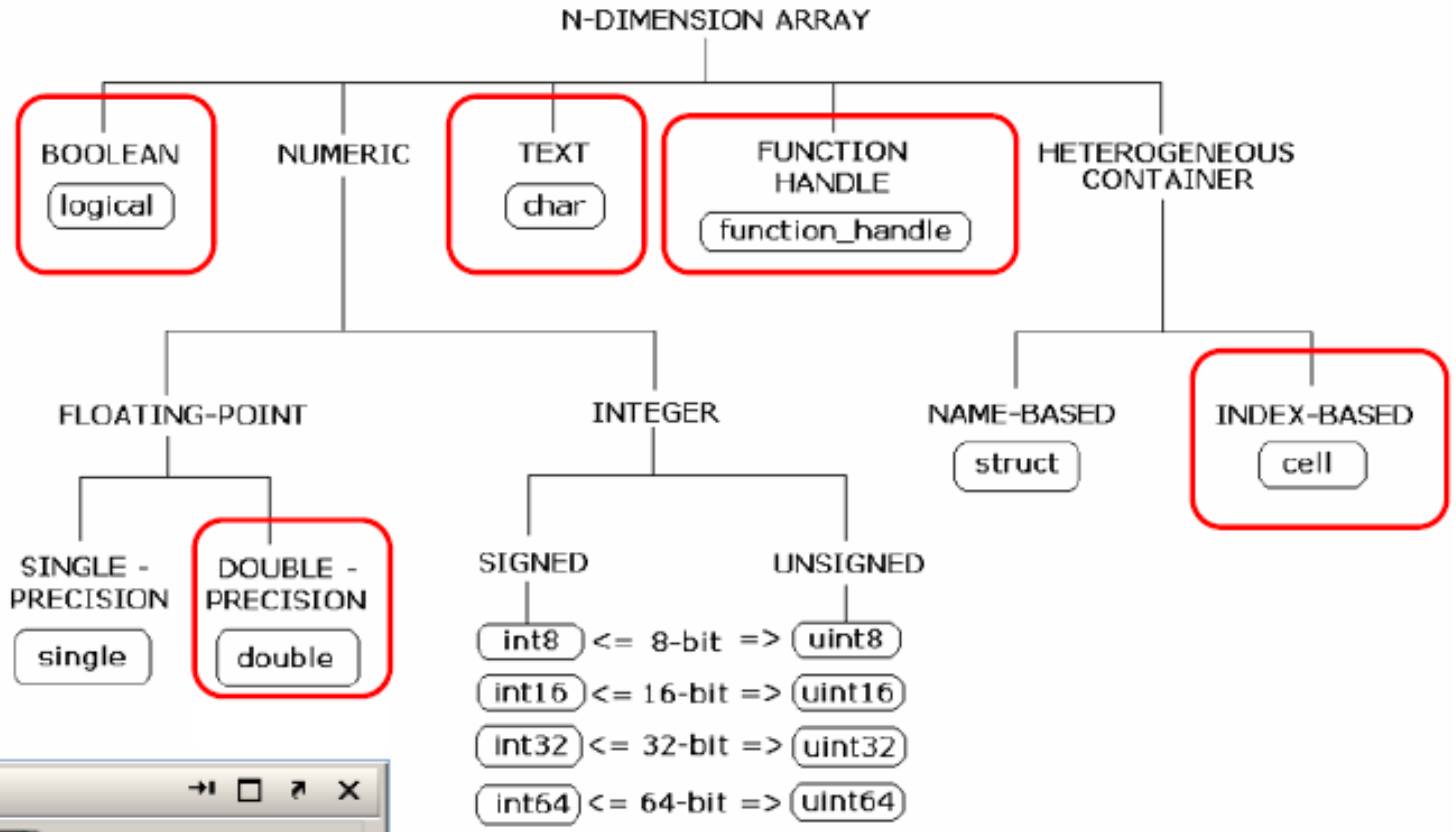


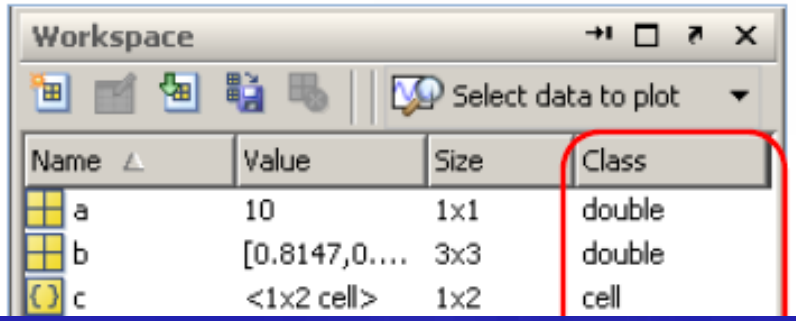


With permission from Todd Atkins, The MathWorks Inc.

Basics: Data and Variables

- Class
- Size
- Value

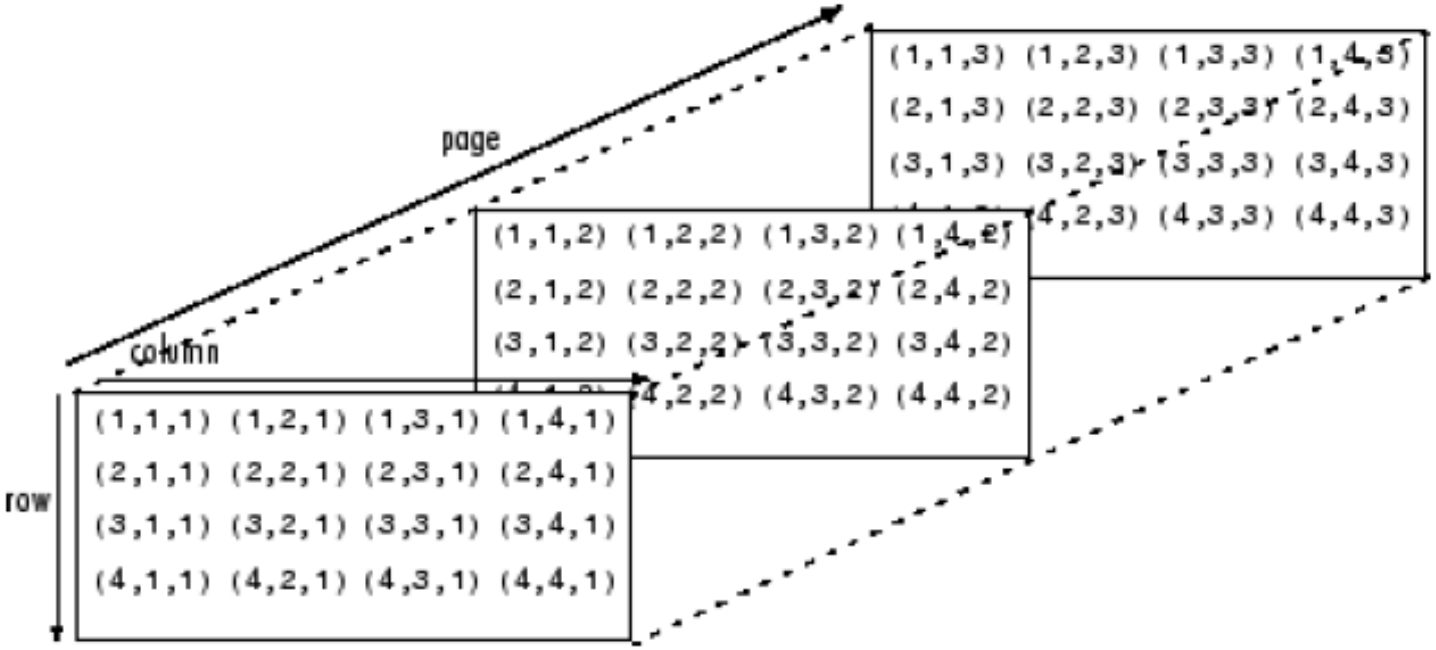


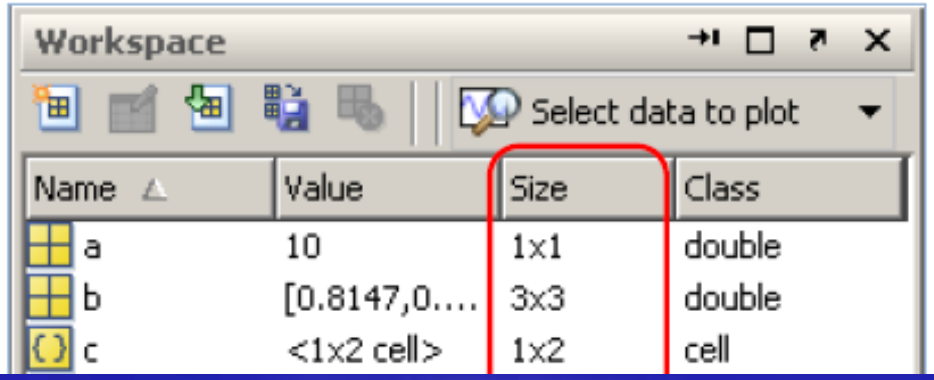


With permission from Todd Atkins, The MathWorks Inc.

Basics: Data and Variables

- Class
- Size
- Value





$m*n$

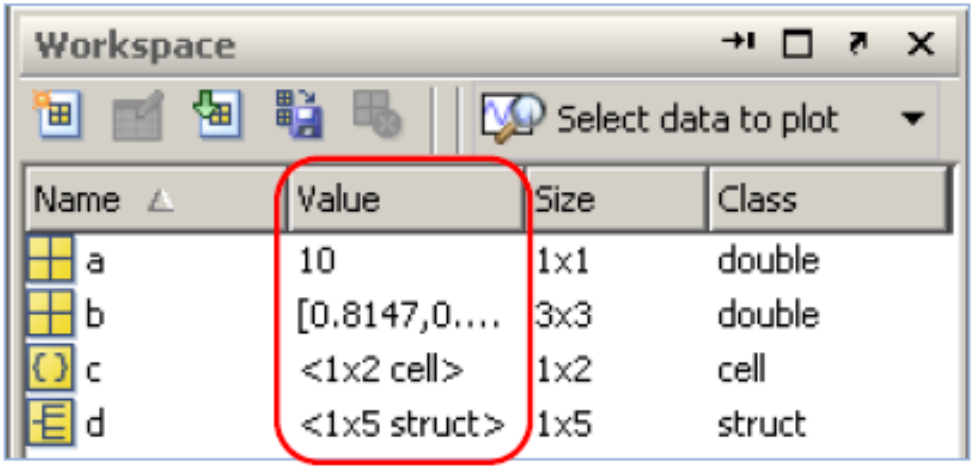
$m*n*\ldots*z$

With permission from Todd Atkins, The MathWorks Inc.

Basics: Data and Variables

- Class
- Size
- Value

```
>> magic(4)
ans =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
>>
```



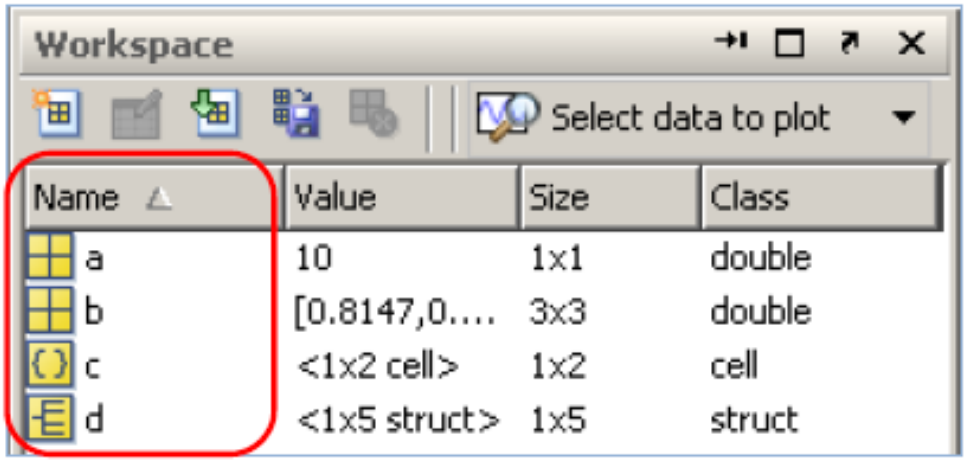
With permission from Todd Atkins, The MathWorks Inc.

Basics: Data and Variables

- Class
- Size
- Value
- Name (“variable”)

foo

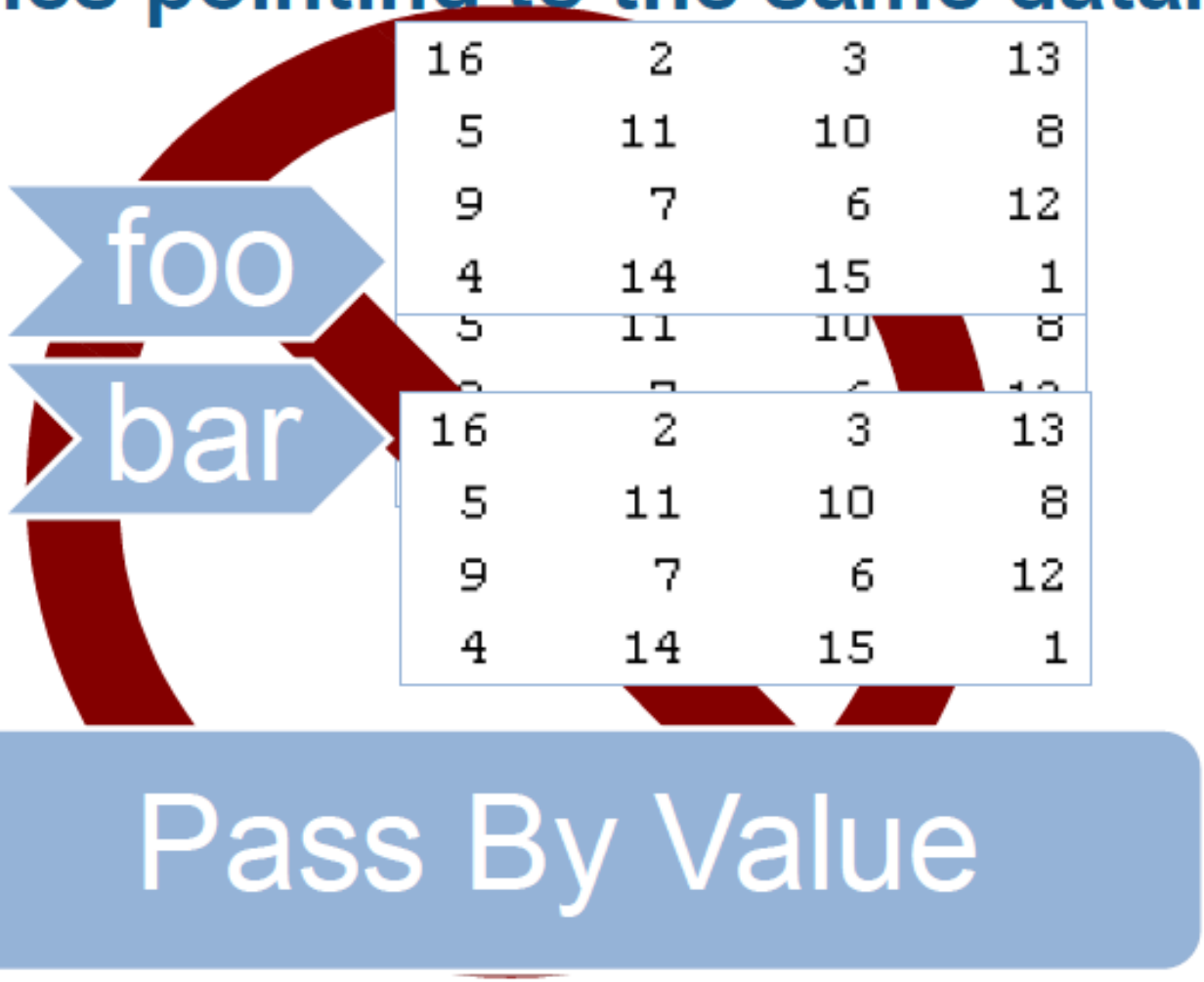
16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1



With permission from Todd Atkins, The MathWorks Inc.

Can We Do This?

Two names pointing to the same data.



With permission from Todd Atkins, The MathWorks Inc.

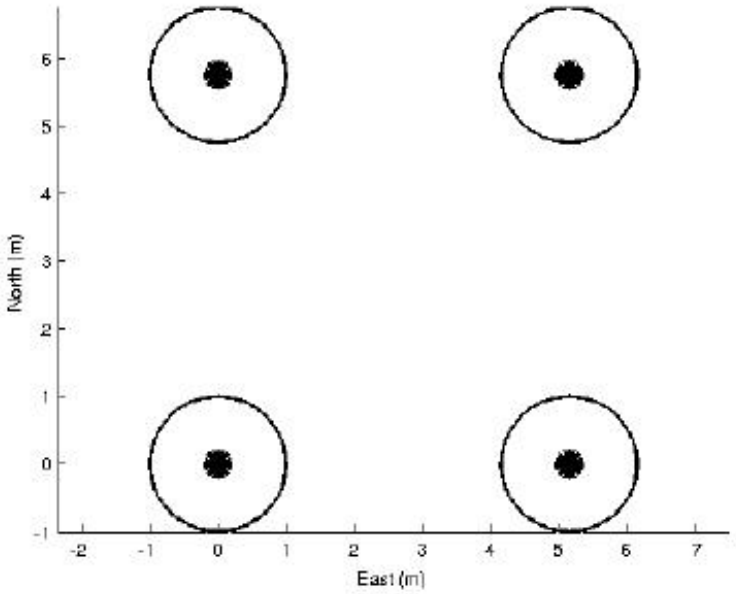
UVS SYSTEM (Spring 2015)

Wild Thumper 6WD Chassis

DIYDrones APMRover Autopilot System

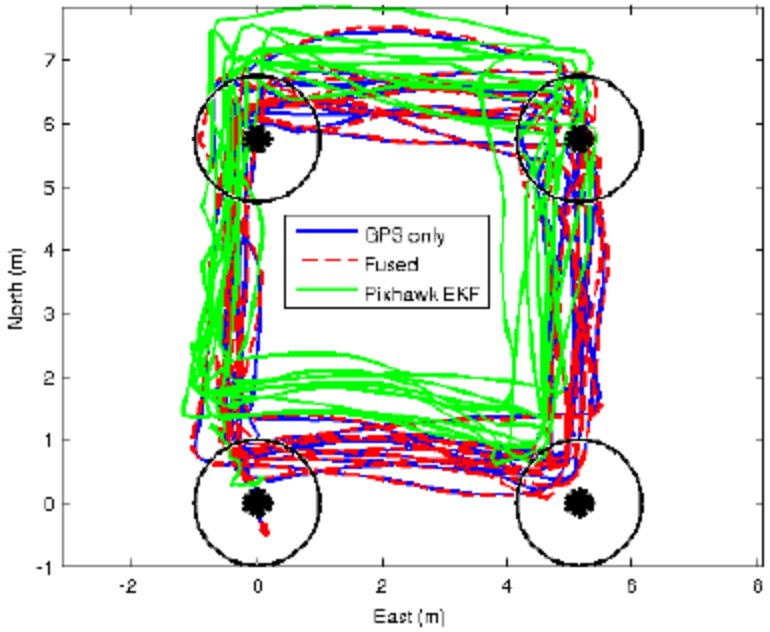
SPRING 2015 TEAM 5 Experiment Results

Our Experiment

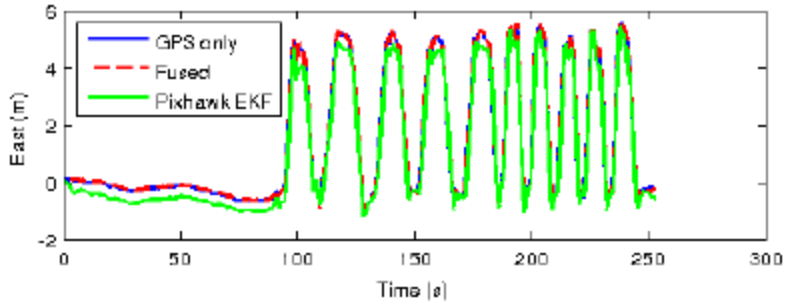


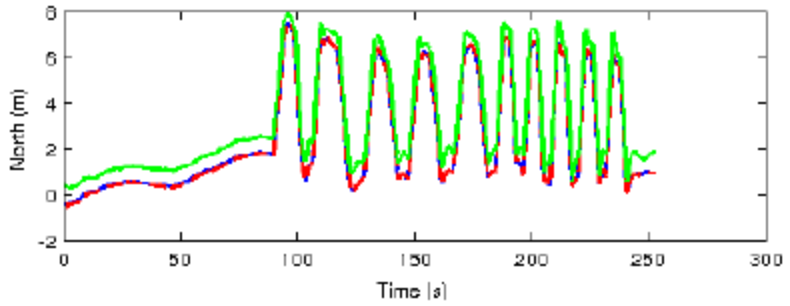
SPRING 2015 TEAM 5 Experiment Results

Position in NED



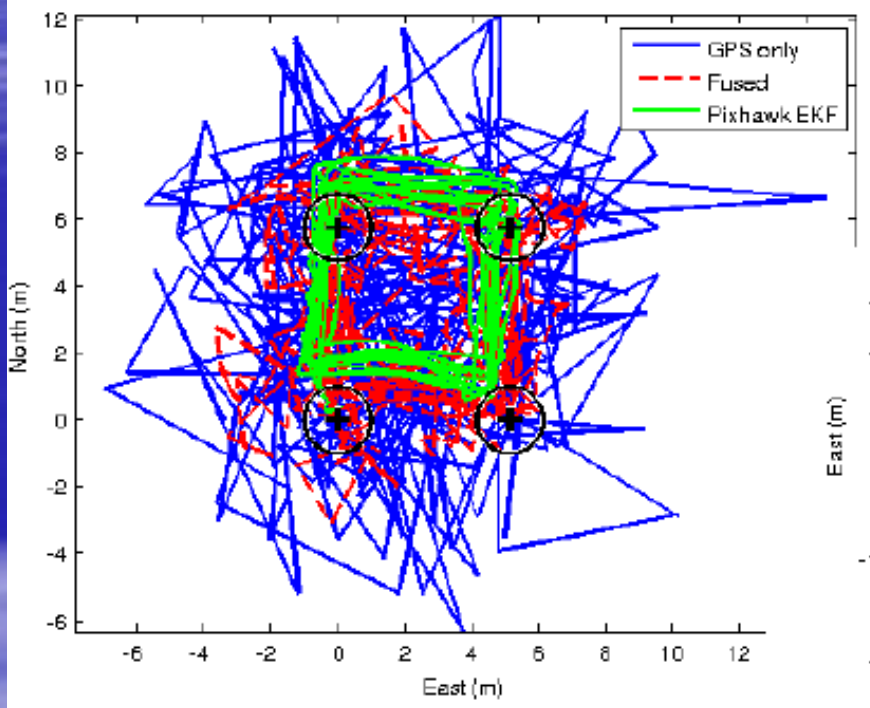
Data was recorded on the processor. After the experiment the Sensor Logs were downloaded via the Mavlink (USB link from Pixhawk/APMRover)



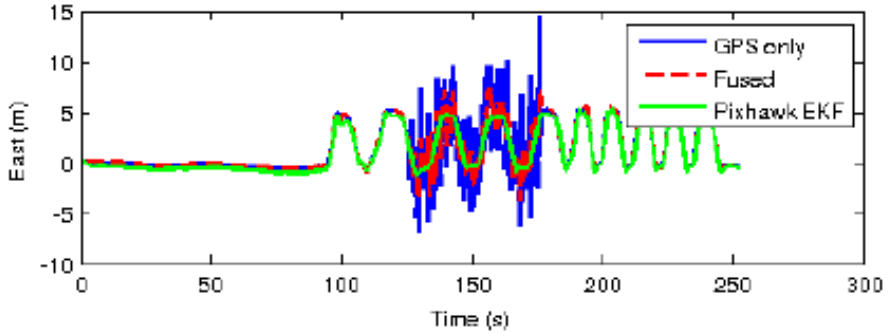


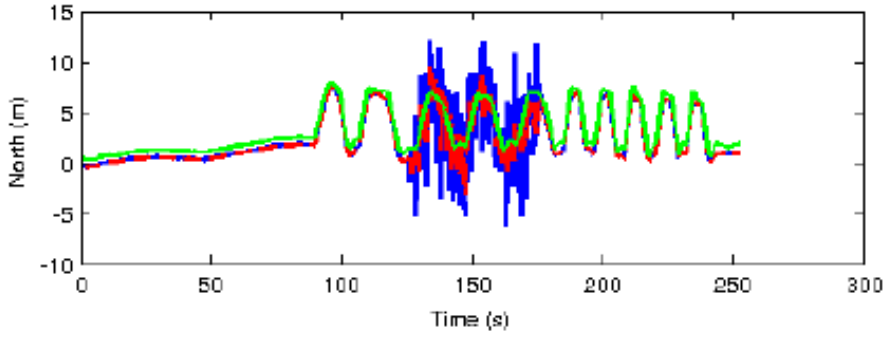
SPRING 2015 TEAM 5 Experiment Results

Position in NED

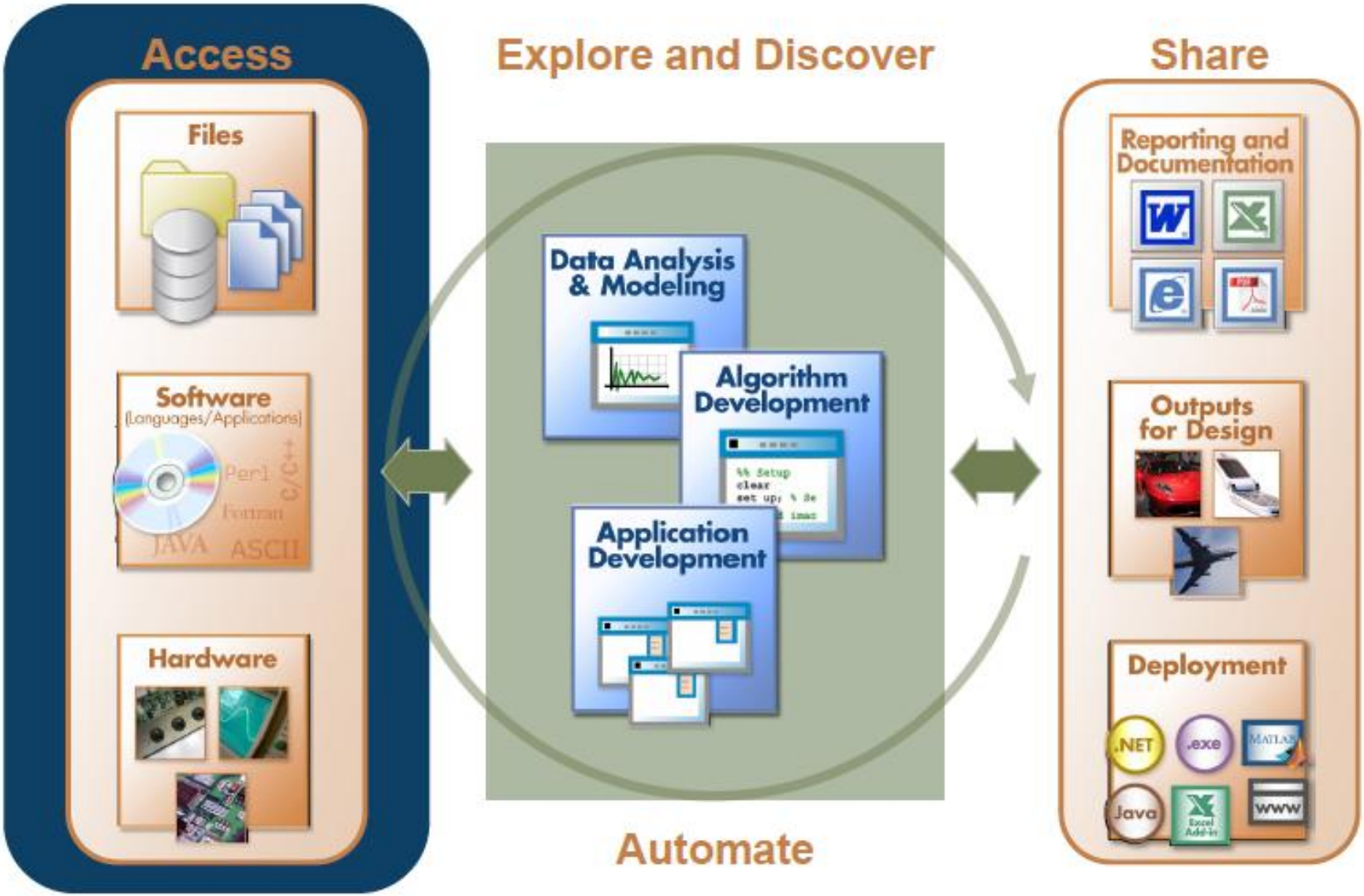


Let's see how DATA can be read into MATLAB before processing it.





Technical Computing Workflow



With permission from Todd Atkins, The MathWorks Inc.

Technical Computing Workflow: Access

Files

- File->Import Data
- `uiimport`
- `importdata`
- `csvread`
- `xlsread`
- `aviread`
- `wavread`
- `fread`
- `fscanf`
- `load/save`
- `doc fileformats`
- …

Software

- COM/.NET
- Java
- C/C++
- SOAP/WSDL
- Database
- OPC
- `ftp`
- `urlread`
- …

Hardware

- Data Acquisition
- Instrument Control
- TCP/IP
- Serial
- Image Acquisition
- …

Excel Sheets, Text, Numeric, Audio and Video Files

With permission from Todd Atkins, The MathWorks Inc.

IMPORT DATA

```
>> [ldata, text, alldata] = xlsread('2015-02-27_07-20-05.csv');

>> load mydata.mat                          % MATLAB DATA

>> imdata = imread('ngc6543a.jpg');         % Image Files
>> imshow(imdata)

>> load handel;
>> p = audioplayer(y, Fs);                  % Audio Files
>> play(p, [1 (get(p, 'SampleRate') * 3)]);

>> help videoreader                         % Video Files
```

Basics: Manipulating Data

- Creation
- Extraction (subset)
- Union (merge)
- Deletion

```
>> a = rand(2,4)
a =
   0.957506835434298   0.157613081677548   0.957166948242946   0.800280468888800
   0.964888535199277   0.970592781760616   0.485375648722841   0.141886338627215
>> a([1,3,5])
ans =
   0.957506835434298   0.157613081677548   0.957166948242946
>> a(1,3)
ans =
   0.957166948242946
>> a(3)
ans =
   0.157613081677548
>>
```

Basics: Manipulating Data

- Creation
- Extraction (subset)
- Union (merge)
- Deletion

```
:
rand
zeros
ones
diag
magic
;
end
linspace
logspace
...
```

Basics: Manipulating Data

- Creation
- Extraction (subset)
- Union (merge)
- Deletion

```
>> a = [-1 2 1 -3 4 7 0.1 0.5];

Find 0 < a < 1

>> i0 = find(a > 0);
>> i1 = find(a < 1);
>> in = intersect(i0, i1);
>> a(in)
```

Subscript
Linear
Logical
()
sub2ind
ind2sub
...

Basics: Manipulating Data

- Creation
- Extraction (subset)
- Union (merge)
- Deletion

Subscript

```
<name>(row, col)
<name>(row, col, …, z)
```

Indices may themselves be arrays.

Basics: Manipulating Data

- Creation
- Extraction (subset)
- Union (merge)
- Deletion

```
[]
repmat
strvcat
Expansion
cat
horzcat
vertcat
...
```

Basics: Manipulating Data

- Creation
- Extraction (subset)
- Union (merge)
- Deletion

```
clear
clearvars
```
Assign to empty

Basics: Manipulating Data

() versus []

Indexing
Order of operations
Argument list

Matrix/Vector creation
Concatenation
Multiple outputs

With permission from Todd Atkins, The MathWorks Inc.

Basics: Manipulating Data

:

:

```
1:5
0:5:25
25:-3:2
25:5:0
0:.5:4
```

end

```
a(1:end)
b(end, end)
```

Student Learning Exercise

- Create a 4x3 matrix of random numbers
- Extract the elements at locations 1,2 and 2,3
- Extract the element in the lower right
- Extract every value >.5
- Create a vector of every other value from 3 to 10
- Create a diagonal matrix of size 4x4 with 3s on the diagonal

Basics: Math

- Matrix operations

```
>> a = [1 2; 3 4]
>> b = [5 6; 7 8]
>> c = a*b
```

- Element operations (dot)

```
>> a = [1 2; 3 4]
>> b = [5 6; 7 8]
>> c = a.*b
>> d = a.^3 + b^2
```

- Others
  - ^ - Power
  - \ - Left divide
  - ' - Transpose

With permission from Todd Atkins, The MathWorks Inc.

Basics: Math

- Scalar expansion

```
>> a = [1 2; 3 4]
>> a+[1 2]
>> a+1
```

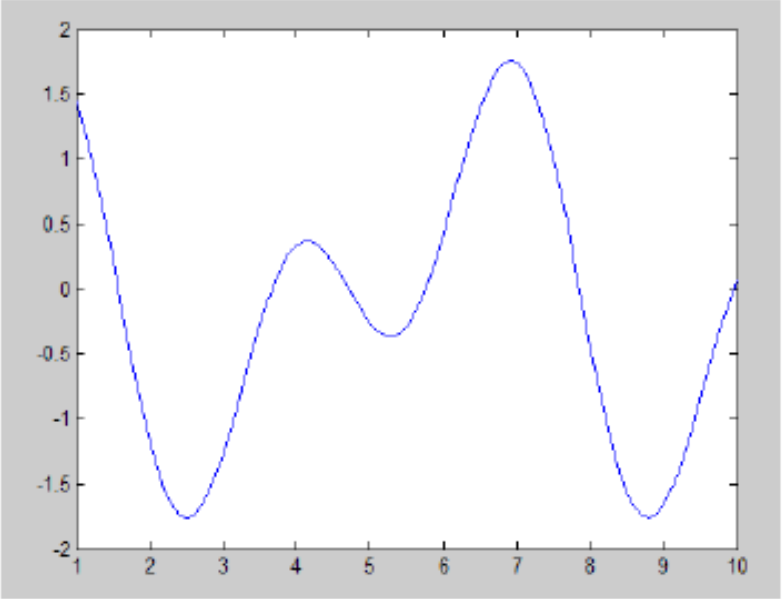
- What happens?

With permission from Todd Atkins, The MathWorks Inc.

Demo

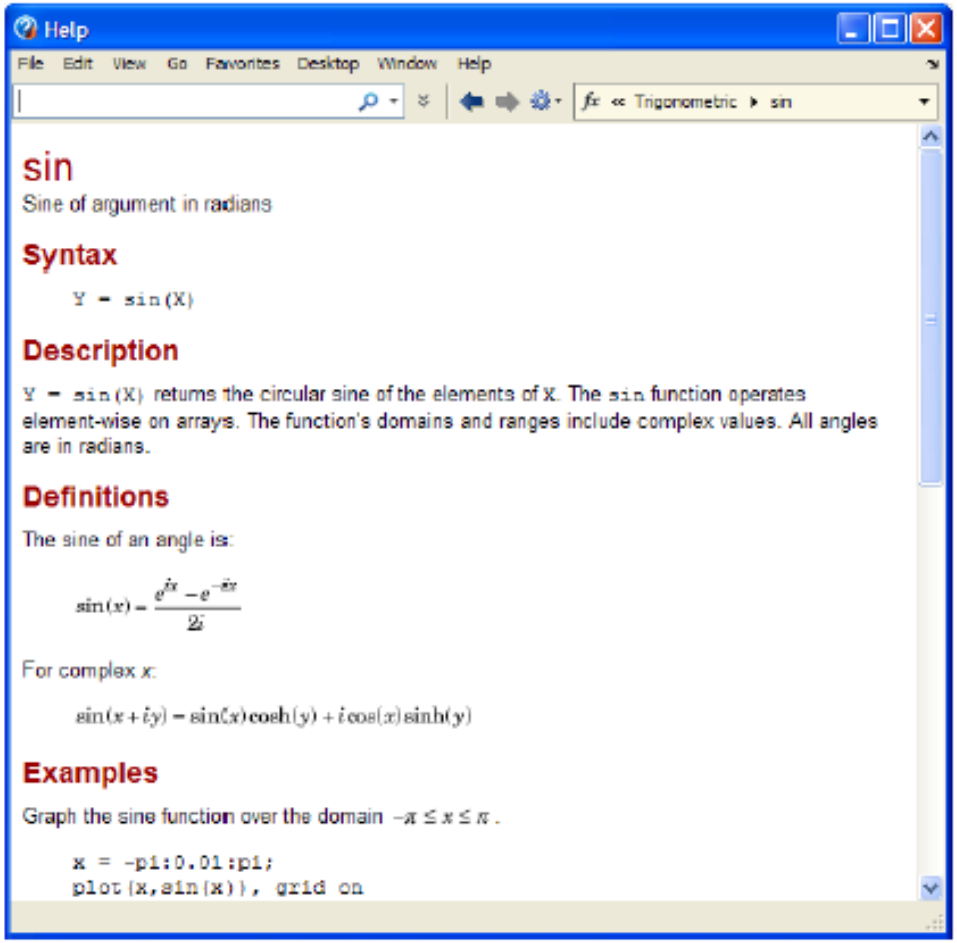
- Compute `y = sin(2t) + cos(t)` where `t` is from 1 to 10 seconds.

- MATLAB Functions
  - `sin`
  - `cos`
  - `plot`



With permission from Todd Atkins, The MathWorks Inc.

MATLAB Functions



With permission from Todd Atkins, The MathWorks Inc.

Calling Functions

Function calling syntax:

```
[out1, out2, ..., outN] = functionname(in1, in2, ..., inN)
```

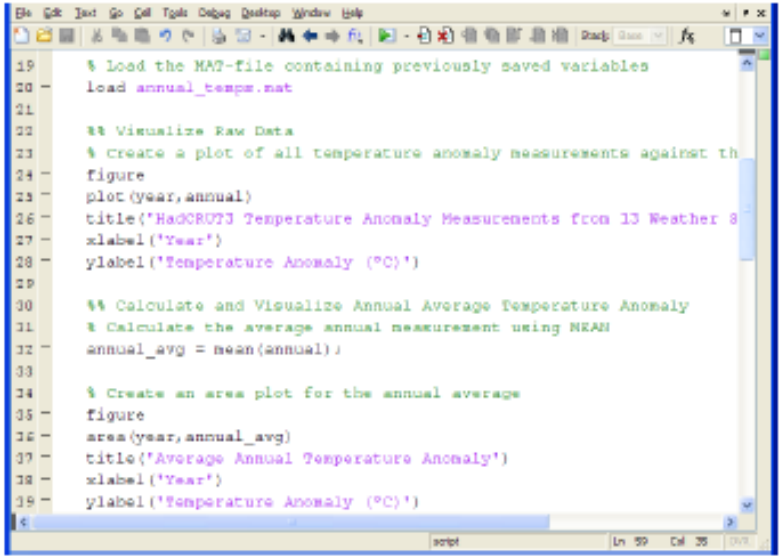
Aside - Command syntax:

```
functionname string1 string2 string3 … stringN
```

Revisit the Help.

Writing MATLAB Programs

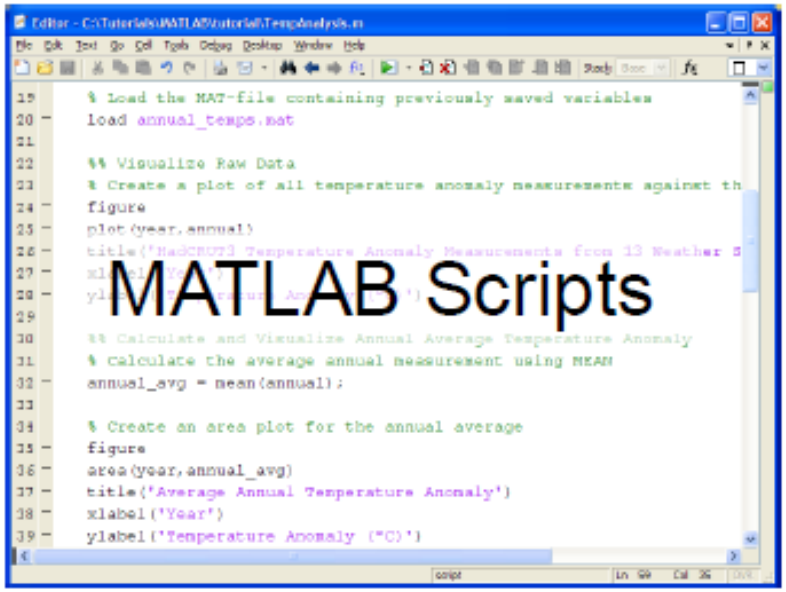
- Using MATLAB Editor
- Executing MATLAB script
- Reusing MATLAB programs

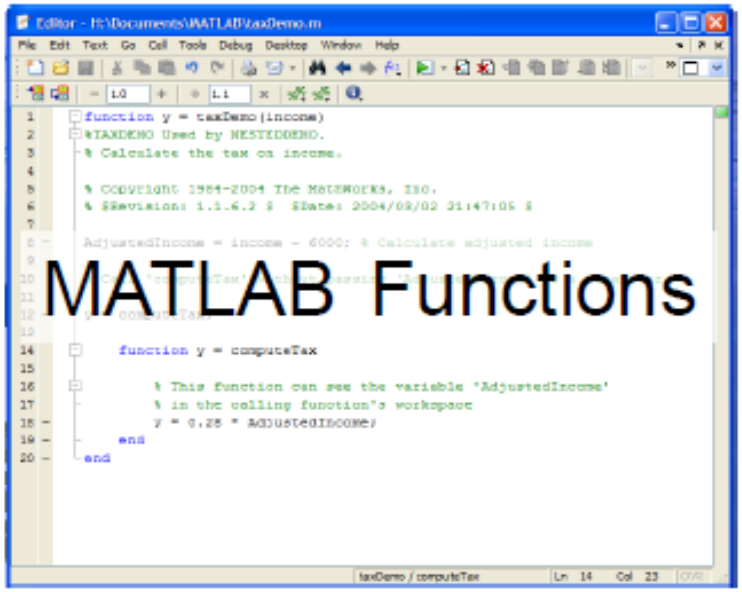


With permission from Todd Atkins, The MathWorks Inc.

MATLAB Program Files

- Why?
  - Automating
  - Editing/Debugging
  - Deploying as applications





With permission from Todd Atkins, The MathWorks Inc.

Basics of a MATLAB Program File

```
function f = fact(n)
% Compute a factorial value.
% FACT(N) returns the factorial of N,
% usually denoted by N!

% Put simply, FACT(N) is PROD(1:N).
f = prod(1:n);
```

```
function [y1, y2] = functionName(x1,x2,…)
```

```
>> help fact
  Compute a factorial value.          H1 line
  FACT(N) returns the factorial of N,  Help text
  usually denoted by N!
```

```
>> fact(3)
ans =
     6
```

With permission from Todd Atkins, The MathWorks Inc.

Types of Functions

- Primary MATLAB-file Functions

```
function [avg, med] = newstats(u) % Primary function
% NEWSTATS Find mean and median with internal functions.
n = length(u);
avg = mean(u, n);
med = median(u, n);
```

With permission from Todd Atkins, The MathWorks Inc.

Types of Functions

- Primary MATLAB-file Functions
- Subfunctions

```
function [avg, med] = newstats(u) % Primary function
% NEWSTATS Find mean and median with internal functions.
n = length(u);
avg = mean(u, n);
med = median(u, n);

function a = mean(v, n)              % Subfunction
% Calculate average.
a = sum(v)/n;

function m = median(v, n)            % Subfunction
% Calculate median.
w = sort(v);
if rem(n, 2) == 1
   m = w((n+1) / 2);
else
   m = (w(n/2) + w(n/2+1)) / 2;
end
```

Types of Functions

- Primary MATLAB-file Functions
- Subfunctions
- Nested Functions

```
function x = A(p1, p2)
...
   function y = B(p3)
   ...
   end
...
end
```

```
function x = A(p1, p2)
...
   function y = B(p3)
   ...
   end

   function z = C(p4)
   ...
   end
...
end
```

With permission from Todd Atkins, The MathWorks Inc.

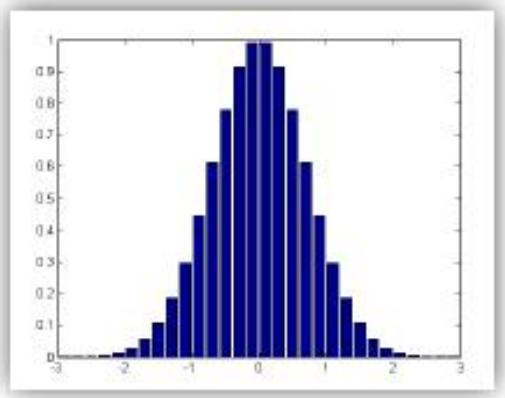
Types of Functions

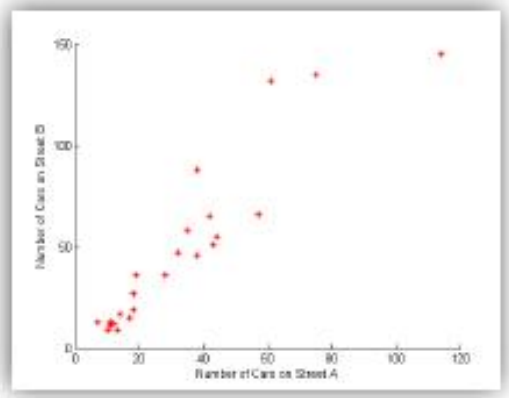
- Primary MATLAB-file Functions
- Subfunctions
- Nested Functions

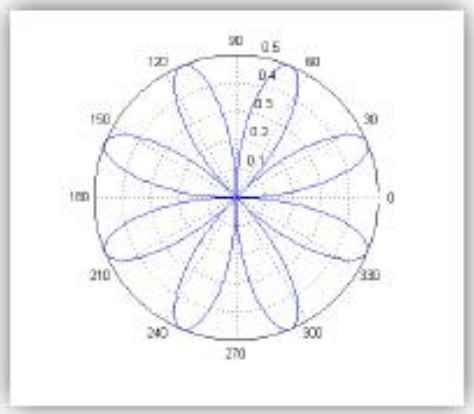
- Anonymous Functions
- Overloaded Functions
- Private Functions

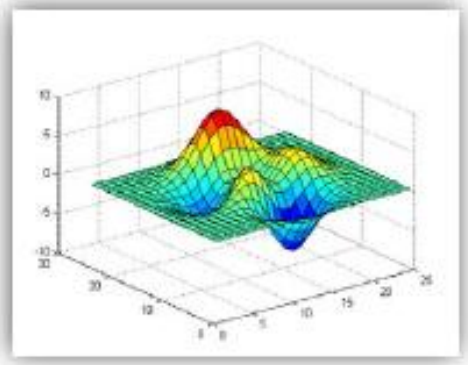
Visualization Tools

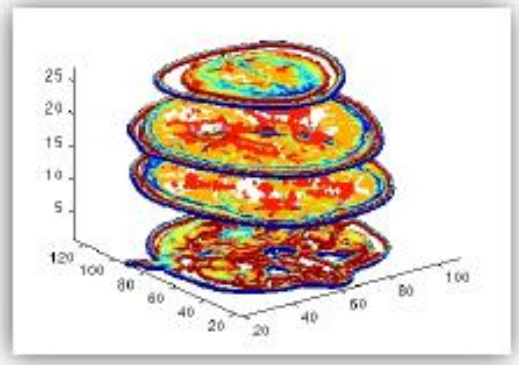
- 2-d plots
- 3-d plots

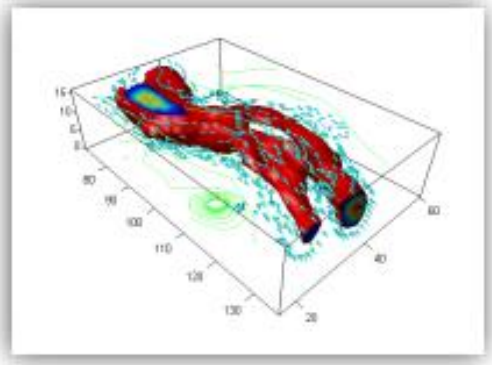








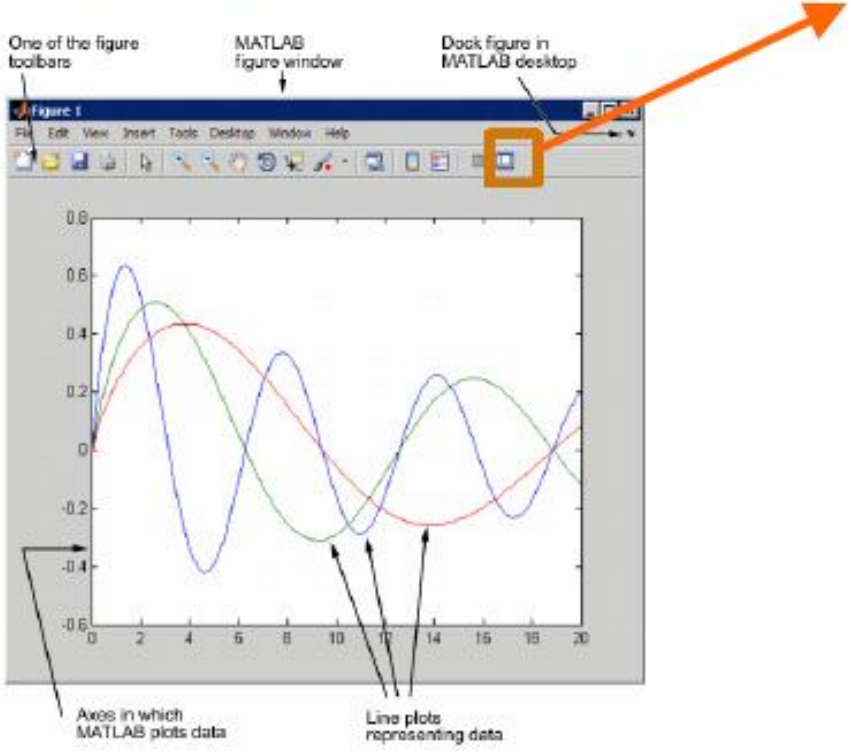


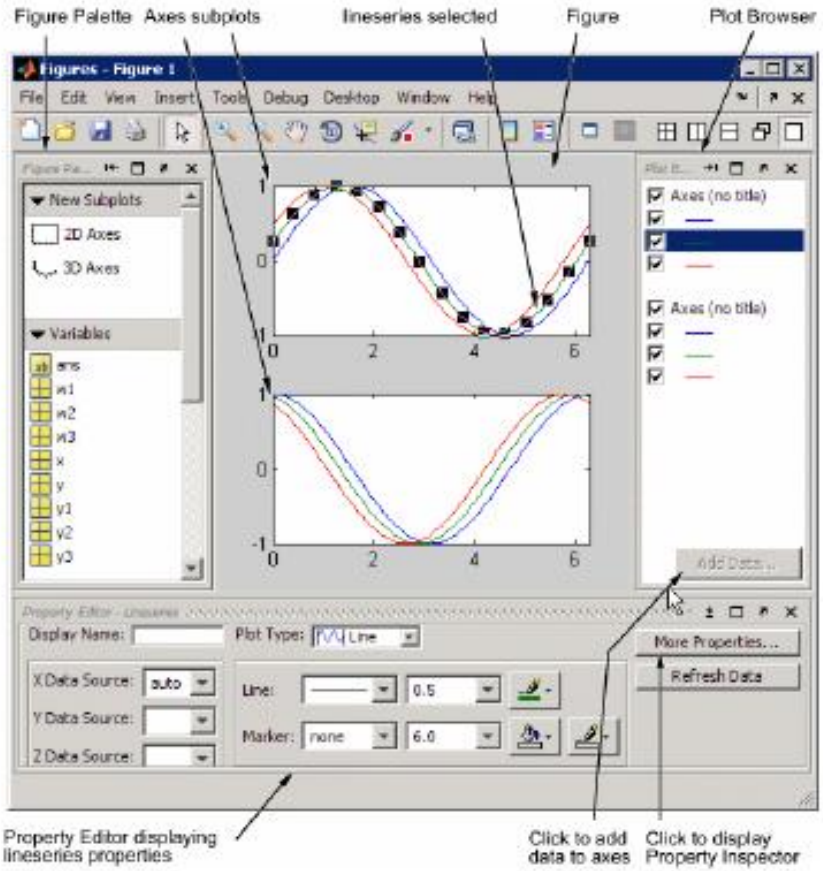


With permission from Todd Atkins, The MathWorks Inc.

Plotting Tools

```
>> x = [0:0.2:20];
>> y = sin(x).sqrt(x+1);
>> y(2,:) = sin(x/2).sqrt(x+1);
>> y(3,:) = sin(x/3).sqrt(x+1);
>> plot(x,y);
```





With permission from Todd Atkins, The MathWorks Inc.

Processing and Analyzing Data

- Data Analysis functions in MATLAB
  - Statistics
    - `cov , max, mean, median, std`
  - Filtering and Convolution
    - `conv, deconv, filter, filter2`
  - Interpolation and Regression
    - `interp1, interpn, mldivide, polyfit, polyval`
  - Fourier Transforms
    - `fft, fftn, fftshift, ifft, umwrap`
  - Derivatives and Integrals
    - `del2, diff, gradient, polyint, trapz`

With permission from Todd Atkins, The MathWorks Inc.

Program Control Statements

- Conditional Control
  - `if/elseif/if, switch/case`
- Loop Control
  - `while, for, break`
- Error Control
  - `try, catch`

With permission from Todd Atkins, The MathWorks Inc.

Conditional Control Statements

- if, elseif and else

```
if n < 0                % If n negative, display error message.
    disp('Input must be positive');
elseif rem(n,2) == 0 % If n positive and even, divide by 2.
    A = n/2;
else
    A = (n+1)/2;      % If n positive and odd, increment and divide.
end
```

- switch, case and otherwise

```
switch input_num
    case -1
        disp('negative one');
    case 0
        disp('zero');
    case 1
        disp('positive one');
    otherwise
        disp('other value');
end
```

With permission from Todd Atkins, The MathWorks Inc.

Loop Control Statements

- `while` (Conditional Loop)

```
n = 1;
while prod(1:n) < 1e100
    n = n + 1;
end
```

- `for` (Iterative Loop)

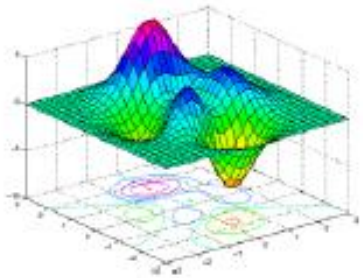
```
for index = start:increment:end
    statements
end
```

```
for m = 1:5
    for n = 1:100
        A(m, n) = 1/(m + n - 1);
    end
end
```

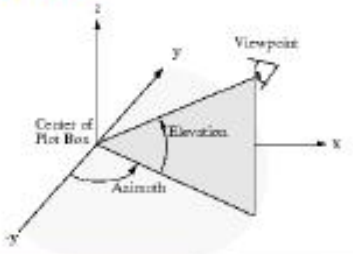
- `continue, break`

3-D Visualization Features

- **Surface and Mesh plots**
  - peaks, surf, mesh, meshgrid
  - colorbar, colormap, shading

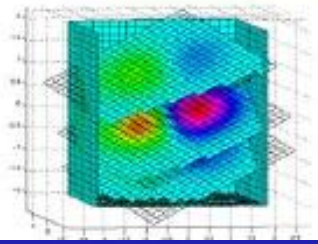


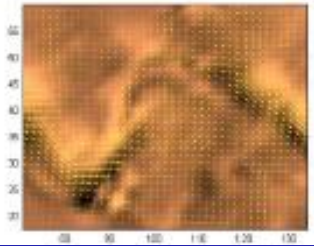
- **View Control**
  - campos, view, daspect, rotate3d



- **Lighting and Transparency**
  - camlight, diffuse, alpha

- **Volume Visualization**
  - curl, isosurface, slice





With permission from Todd Atkins, The MathWorks Inc.

Object-Oriented Programming

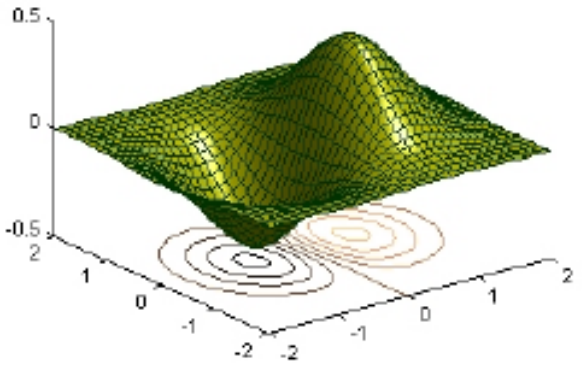
Editor - C:\Program Files\MATLAB\R2009b\help\techdoc\matlab_oop\examples\@topo\...

File Edit Text Go Cell Tools Debug Desktop Window Help

```
1  classdef topo < handle % Plots function of 2 vars
2     properties
3        FigHandle        % Store figure handle
4        FofXY;           % Function handle to fcn being evaluated
5        Lm = [-2*pi 2*pi]; % Default range if not specified
6     end
7
8     properties (Dependent = true, SetAccess = private)
9        Data % Data property depends on current value of FofXY
10    end
11
12    methods
13       function obj = topo(fnc,limits)
14          obj.FofXY = fnc;
15          obj.Lm = limits;
16       end
17
18     function set.Lm(obj,lim)
19     % Lm property set after checking limits
20       if  -(lim(1) < lim(2))
21          disp('Bad limits, using [-2pi 2pi]')
```

topo Ln 1

```
tobj = topo(@(x,y) x.*exp(-x.^2-y.^2),[-2 2]);
a = tobj;
surflight(a) % Call class method to create a graph
```



With permission from Todd Atkins, The MathWorks Inc.

External Interfaces

- Shared libraries (`.dll, .so, .dylib`)
- C, Fortran interface
- C, Fortran MEX-files (`.mex`)
- Sun Java classes
- COM/.NET support
- Web services
- Serial Port I/O

More on MATLAB

- MATLAB Tutorials

- Demos and Webinars

- Documentation

- MATLAB Central
  (User Community)

With permission from Todd Atkins, The MathWorks Inc.

Summary

- MATLAB is a **high level-language** for **technical computing**
- Interactive tool with **mathematical and graphical** functions
- MATLAB provides features to **access, compute, analyze and visualize data**
- MATLAB also provides capabilities to **interface with external languages**

With permission from Todd Atkins, The MathWorks Inc.

Dynamical Systems

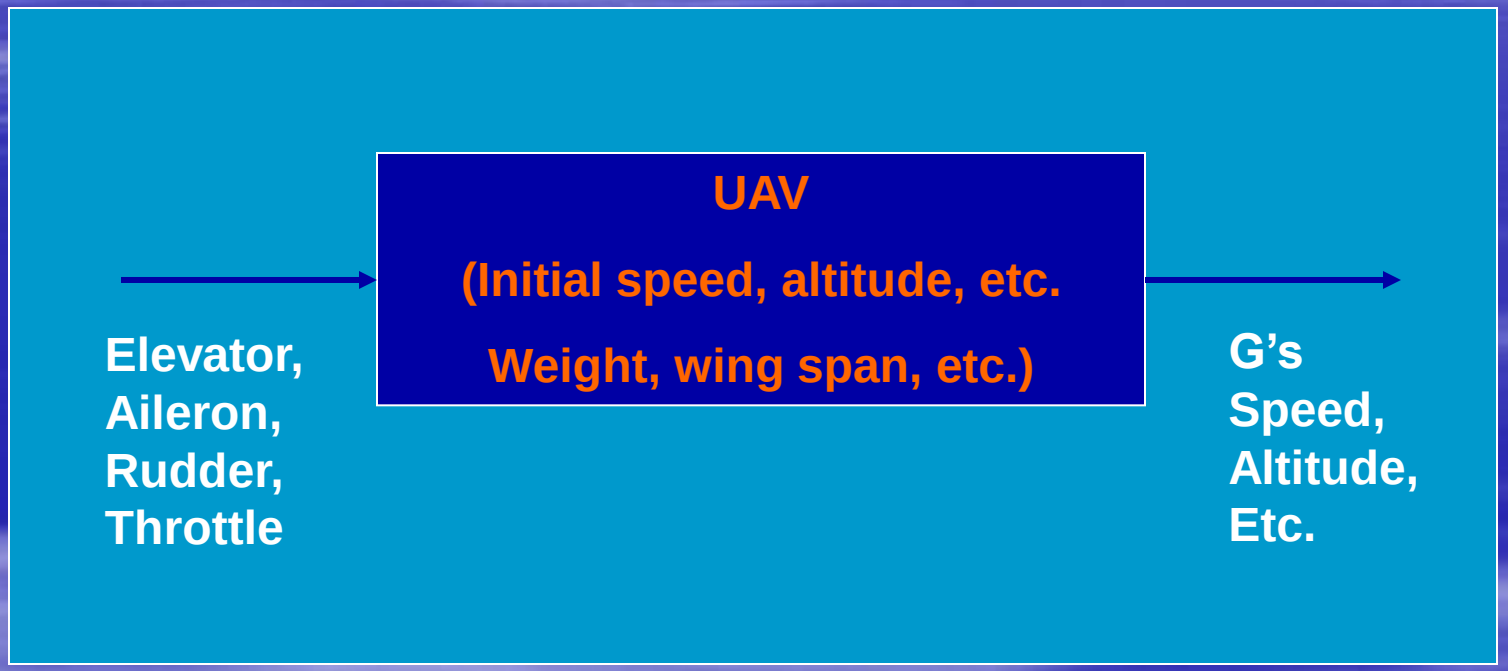
Working Definition:

"A dynamical system is a mathematical representation of a physical phenomenon whose state evolves over time."

These are used in almost all branches of Engineering, Economics, Medicine, etc.

Dynamical Systems

e. g. Airplane



Outputs are typically (linear/nonlinear) combinations of states

Mathematical Modeling of Dynamical Systems

AUTONOMOUS VEHICLE SYSTEM DEFINITION

(Source: http://avl.uta.edu)

"An autonomous vehicle system thinks, decides and operates in an unstructured (dull, dirty, or dangerous) environment and significantly outperforms an equivalent manned system under similar conditions" – Arthur A. Reyes

Augmentation of Normal Systems with Autonomy

- Design and test on real hardware is expensive
- The real operating scenario is not always reproducible in the laboratory
- Mathematical modeling and simulation are powerful tools at our disposal

Dynamical Systems

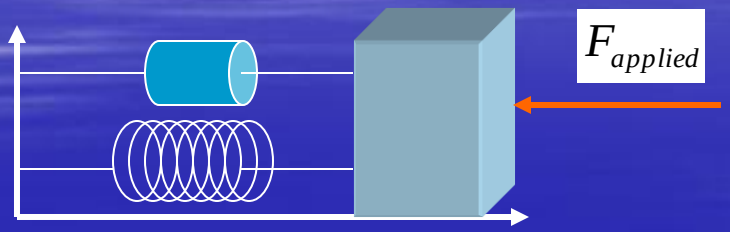
Examples: Aircraft, Spacecraft, Robotic vehicles, Spring-Mass-Dampers

Greatly Simplified Equations of Motion (esp. Kinematics)

Translation	Rotation	
$\frac{dx}{dt} = v$	$\frac{d\theta}{dt} = \omega$	KINEMATICS
$m\frac{dv}{dt} = F_{external}$	$J\frac{d\omega}{dt} + \omega \times J\omega = \tau_{external}$	DYNAMICS

Dynamical Systems

Example: Mathematical Model for a Spring-Mass-Damper system acted upon by a force



Kinematics:

Let position be described by 'x' & speed by 'v'

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

Dynamics:

$$m\frac{dv}{dt} = F_{damping} + F_{restoring} + F_{applied}$$

$$\text{or } \frac{dv}{dt} = \frac{1}{m}F_{damping} + \frac{1}{m}F_{restoring} + \frac{1}{m}F_{applied}$$

Dynamical Systems

Remarks:

- The motion of the mass was specified via two differential equations.
- It was enough to use two "*states*" to complete the description of the dynamics.
- Evolution of these "*states*" tells us the dynamic behavior of the mass.
- The exact nature of $F_{damping}$, $F_{restoring}$ is determined by experiment
- Irrespective of the nature of the above, we can design $F_{applied}$ to tailor the evolution of the "*states*". *(you need to check other things but we will save that for later)*

Dynamical Systems

Assume $F_{damping} = -cv$ & $F_{restoring} = -kx$

This implies :

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

$$\frac{dv}{dt} = -cv - kx + F_{applied}$$

Dynamical systems represented by differential equations such as above i.e., constant coefficient, linear differential equations are termed as Linear Time Invariant (LTI) Systems.

Dynamical Systems

A very convenient Matrix representation of these Linear Time Invariant (LTI) systems follows:

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

$$\frac{dv}{dt} = -cv - kx + F_{applied}$$

$$\frac{d}{dt}\begin{Bmatrix} x \\ v \end{Bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -k & -c \end{bmatrix}}_{A} \begin{Bmatrix} x \\ v \end{Bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} F_{applied}$$

X A B

$Y = \begin{bmatrix} 1 & 0 \end{bmatrix} X \Rightarrow$ Output is the position x

$Y = \begin{bmatrix} 0 & 1 \end{bmatrix} X \Rightarrow$ Output is the velocity v

$Y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} X \Rightarrow$ Output is the position x & velocity v

Introduce Output Function

Y = C X

Let $U = F_{applied}$

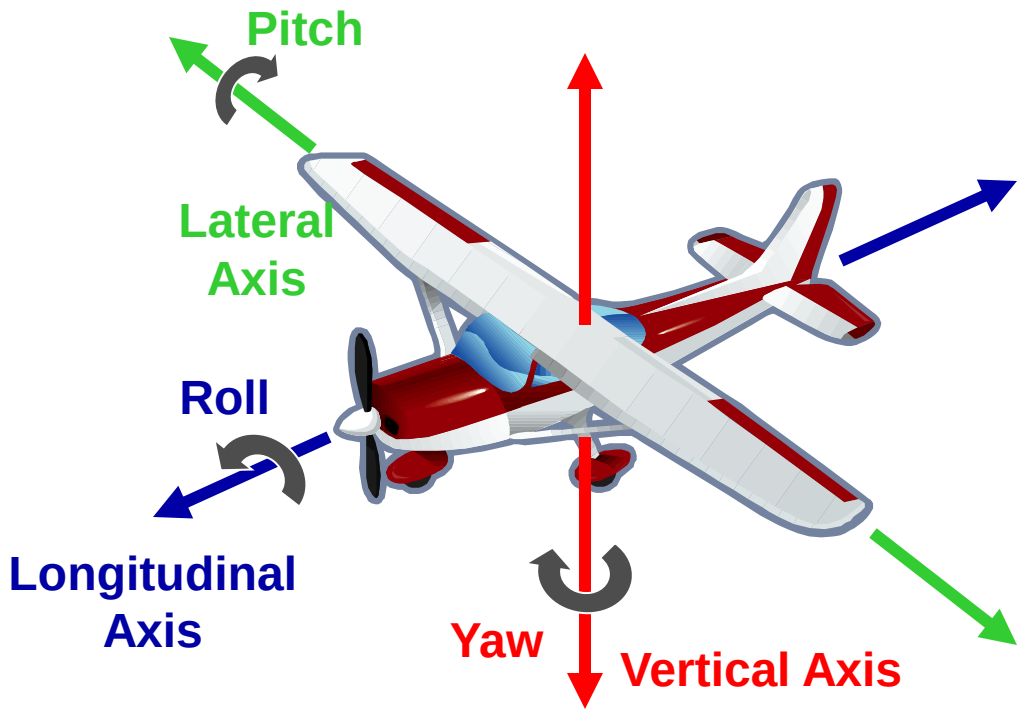
$\therefore \dot{X} = AX + BU$

$$\dot{X} = AX + BU$$

$$Y = CX + DU$$

Dynamical Systems

Example: Aircraft Dynamics



Equations of motion for such a system can be derived, then at specific flight conditions (altitude, speed), LTI models as shown previously can be obtained.

Dynamical Systems

$\dot{X} = AX + BU$	$\dot{X} = AX + BU$
$Y = CX$	$Y = CX + DU$

State Space Representation of LTI Dynamical Systems

Remarks:

- **Notice the use of Matrices in the system descriptions**
- **Some questions of interest – Stability of the LTI system? Solution for different initial conditions, inputs. Simulation for arbitrary inputs.**
- **This is done so as to leverage a whole bunch of analysis tools available from Linear Algebra and Matrix Theory**
- **This also motivates us to use MATLAB (MATrix LABoratory)**

MATLAB for Dynamical Systems

Complete step-by-step guide to understanding the functionality of MATLAB

http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html

Linear Algebra Tools of specific interest:

```
>> A = [0 1; -2 -1];        % 2 X 2 matrix with k = 2 and c = 1
>> eig(A)                   % Eigenvalues of A – What do these tell us?
>> damp(A)                  % Properties of the eigenvalues of A
```

RECALL

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

$$\frac{dv}{dt} = -cv - kx + F_{applied}$$

$$\therefore \frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = F_{applied}$$

MATLAB for Dynamical Systems

RECALL

$$\therefore \frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = F_{applied}$$

Assume zero initial conditions i.e., $x(0) = 0, v(0) = 0$

Taking Laplace Transforms on both sides

$$s^2x(s) + csx(s) + kx(s) = F_{applied}(s)$$

$$\frac{x(s)}{F_{applied}(s)} = \frac{1}{s^2 + cs + k}$$

Transfer Function Representation

Further, recall a little bit of your differential equations class. The solution for different $F_{applied}$ (constant, sine, etc) can be obtained from an inverse Laplace Transform.

If $F_{applied}$ was zero and the initial-conditions were non-zero, the evolution of x(t) indicates the stability of the dynamical system (Zero Input/Free Response)

This in turn depends on the roots of the denominator on the RHS

MATLAB for Dynamical Systems

For $c = 1$ and $k = 2$

$$D(s) = s^2 + s + 2$$

```
>> Den = [1 1 2];          % Specify coefficients of the polynomial
>> roots(Den)              % List roots of the polynomial
>> damp(Den)               % Properties of the roots of D
>> damp(roots(Den))        % Properties of the roots of D
>> damp(A)                 % Properties of eigenvalues of A
>> A = [0 1; -2 -1];
>> B = [0; 1];
>> C = [1 0];
>> D = 0;
```

MATLAB for Dynamical Systems

Dynamical Systems can be represented in several different forms

```
>> sys_ss  = ss(A, B, C, D);        % State Space Representation
>> sys_tf   =  tf(sys_ss);          % Transfer Function Representation
>> Num      = 1;                    % Numerator polynomial
>> Den      = [1 1 2];              % Denominator polynomial
>> sys_tf1  = tf(Num, Den);         % Transfer Function Representation
>> sys_zpk = zpk(sys_tf);           % zero-pole-gain format from Trans Fun.
>> sys_zpk1= zpk(sys_ss);           % zero-pole-gain format from State Space
```

MATLAB will do the necessary conversions for you automatically, once some representation is obtained.

MATLAB for Dynamical Systems

Time Evolution of Dynamical Systems

LTI Systems: $\dot{X} = AX + BU$

$$Y = CX + DU$$

Let's say we want to study how the dynamical system under consideration behaves for a zero input, but non-zero initial conditions i.e.

$$X(0) = \begin{Bmatrix} 1 \\ 2 \end{Bmatrix}, \qquad U = 0$$

```
>> sys_ss = ss(A, B, C, D); % create the state space system
>> initial(sys_ss,X0)
```

Let's say we want to study how the dynamical system under consideration behaves for a non-zero input, but zero initial conditions i.e.

$$X(0) = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}, \qquad U \neq 0$$

```
>> t   = 0:0.1:4;          % create a time vector
>> m = length(t);          % length of the time vector
>> U  = ones(m,1);         % Input column vector of ones (step input)
```

MATLAB for Dynamical Systems

Time Evolution of Dynamical Systems

LTI Systems: $\dot{X} = AX + BU$

$$Y = CX + DU$$

>> help lsim

>> y = lsim(sys_ss,U,t) % Simulate the system sys_ss for the time specified in length of the time vector t for the applied input U.

>> y = step(sys_ss) % The control system toolbox provides this function

The above simulation functions are helpful only for Linear Time Invariant Systems. For linear time varying systems and nonlinear systems (most real systems are nonlinear) we take recourse to NUMERICAL INTEGRATION of the differential equations.