SysEng 5212/EE 5370
Introduction to Neural Networks and Applications

Week 2: Linear Algebra Review, Introduction to MatLab TM

Cihan H Dagli, PhD

Professor of Engineering Management and Systems Engineering
Professor of Electrical and Computer Engineering
Founder and Director of Systems Engineering Graduate Program

dagli@mst..edu

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY
Rolla, Missouri, U.S.A.

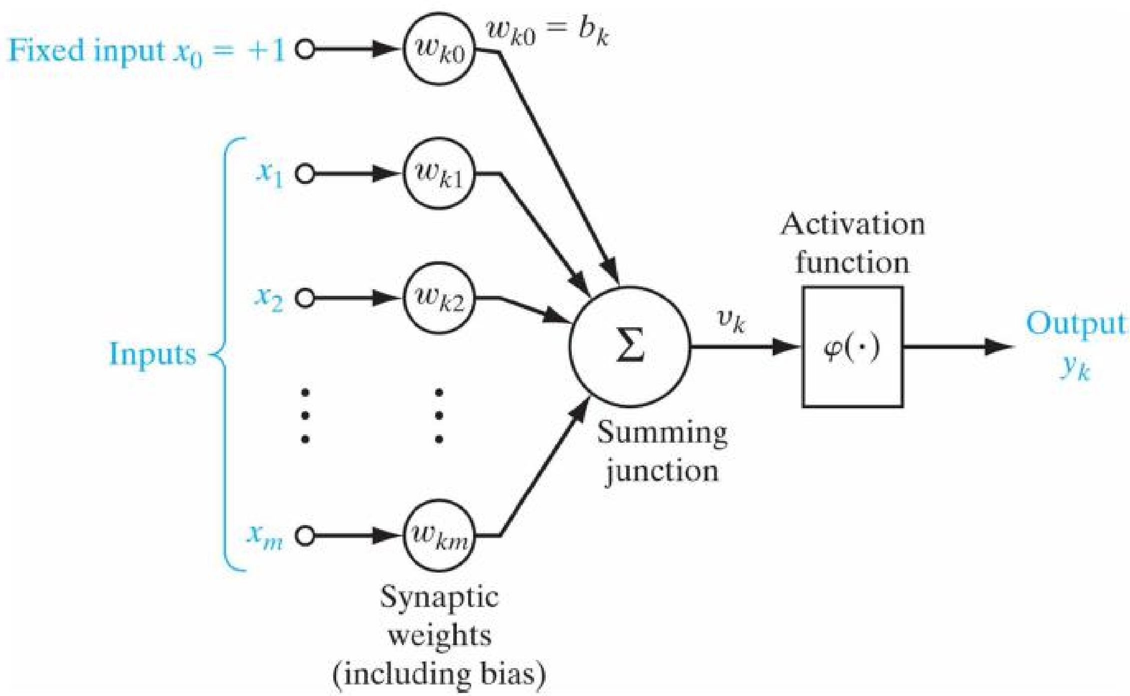
SYSTEMS ENGINEERING
Research Center
Department of Defense UARC

Smart Engineering Systems Lab
Engineering Management and Systems Engineering Department

Lecture outline

- Linear Algebra Review
- Introduction to MATLAB
- MatLab Demos
- Homework Part 1 and 2 due next week.

Nonlinear Model of the Neuron



Output at the summing junction is $v_k = w_{kj}x_j$. If we consider a network with n neurons, we can express the output as a system of linear equations.

NN Output as a System of Linear Equations

$$v_k = w_{kj}x_j = \begin{pmatrix} v_1 = w_{11}x_1 + w_{12}x_2 + \ldots w_{1m}x_m \\ v_2 = w_{21}x_1 + w_{22}x_2 + \ldots w_{1m}x_{2m} \\ \vdots \\ v_n = w_{n1}x_1 + w_{n2}x_2 + \ldots w_{nm}x_m \end{pmatrix}$$

In vector representation, the expression becomes,

$$v = Fx$$

It is very helpful to think of NN inputs, outputs and weights as vectors and matrices. This allows us to frame all computations in terms of vector-matrix operations.

Systems Engineering Research Center
Department of Defense UARC
Smart Engineering Systems Lab
Engineering Management and Systems Engineering Department

Vectors

A vector can be loosely defined as an ordered collection of components of the vector. Vector components can be real numbers, complex numbers, functions or matrices.

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix}$$

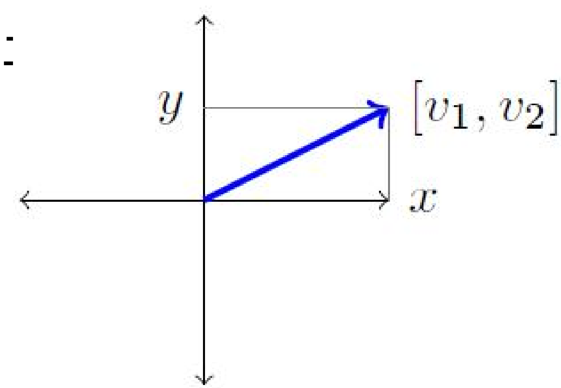
N is the dimensionality of the vector

Visualizing Vectors

Two and three dimensional vectors can be graphically visualized as

arrows with the tail at the origin (0; 0) and the head at the coordinate

location specif [v_1; v_2].



Most vectors we will deal with will be ordered n-tuples or columns of

real numbers, i.e., vectors in R^n n-dimensional Euclidean space

L-2 Norm

The norm of a mathematical object is a quantity that in some (possibly abstract) sense describes the length, size, or extent of the object.

This norm is denoted by ||x|| and gives the length of an n-vector x=(x_1,x_2,...,x_n). It can be computed as

||x|| = sqrt(x_1^2+x_2^2+...+x_n^2).

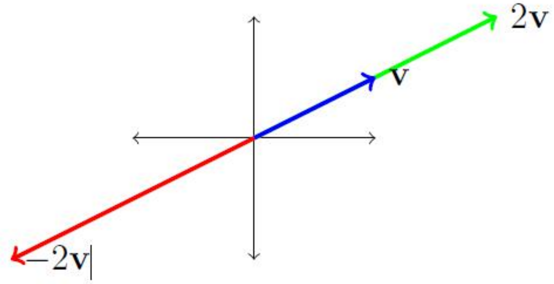
- A vector with all zero components (e.g.,[0; 0; 0; 0]) is called a zero
- vector. The norm of a zero vector is zero.
- A unit vector is a vector of length one.

Scaling a Vector

Multiplying a vector by a scalar changes the length of the vector by that factor.

$$||a\mathbf{v}|| = a||\mathbf{v}||$$

The set of all scaled versions of a vector lie on a straight line.



To scale a vector to unit length, divide by its norm,

$$\ddot{u} = \mathbf{v}/||\mathbf{v}||$$

Summing Vectors

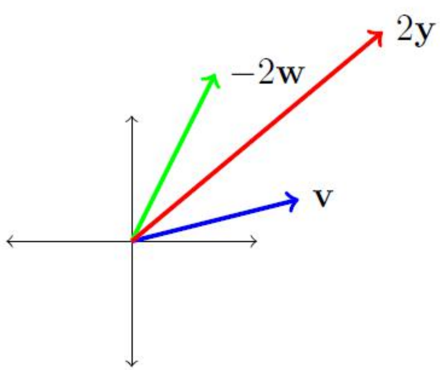
To add (or subtract) two vectors they must be of the same dimension. The sum of two vectors, w = $[w_1;w_2]^T$ and v = $[v_1; v_2]^T$ is

given by,

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} w_1 + v_1 \\ w_2 + v_2 \end{bmatrix}$$

$$\mathbf{w} + \mathbf{v} = \mathbf{y}$$

Geometrically, when we add two vectors, we stack them head to foot,

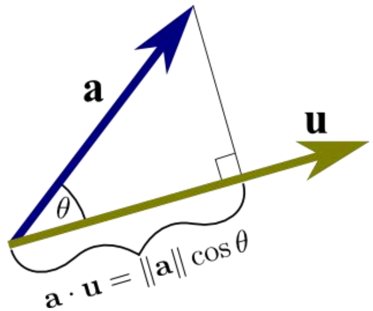


Inner Product

- The inner product is the sum of the pairwise product of the vector's components. It is also known as the dot product. The result of the operation is a scalar.

$$v \cdot w = \sum_n v_n w_n = v_1 w_1 + v_2 w_2 + : : : + v$$

Dot product as projection onto a unit vector



The dot product of vectors **a** and unit vector **u** is the projection of **a** onto **u**, i.e.,**a** · **u**= // **a** // // **u** // cosϑ,

where ϑ is the angle between **a** and **u**. When the vectors are perpendicular, ϑ = 90, and the inner product is zero, **a** · **u** = 0.

When the vectors are parallel, ϑ = 0, and the inner product is the product of the vector norms.

a · **u**= // **a** // // **u** //

Vector Spaces

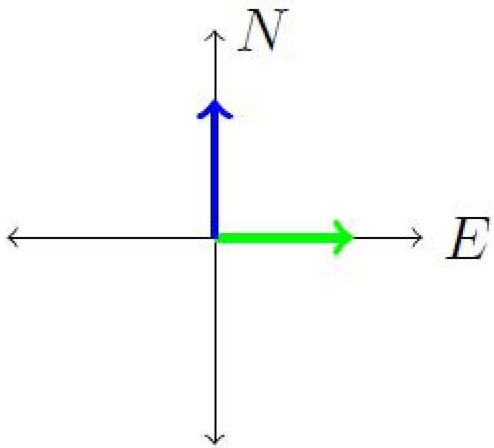
A linear vector space, V, is a set of elements defined over a scalar field, F, that satisfies the following conditions.

1) If **u** and **v** are objects in V, then **u** + **v** is in V.
2) **u** + **v** = **v** + **u**
3) **u** + (**v** + **w**) = (**u** + **v**) + **w**
4) There is an object **0** in V, called a zero vector for V, such that **0** + **u** = **u** + **0** = **u** for all **u** in V.
5) For each **u** in V, there is an object –**u** in V, called a negative of **u**, such that **u** + (-**u**) = (-**u**) + **u** = **0**.
6) If *k and l* are any two scalars in F and **u** is any object in V, then *k***u** is in V.
7) *k* (**u** + **v**) = *k***u** + *k***v**
8) (*k* + *l*) **u** = *k***u** + *l***u**
9) *k* (*l***u**) = (*kl*) (**u**)
10) 1**u** = **u**

Vector Spaces Example

A vector space is n-dimensional space with vector objects closed
under linear combination.

Example: On a map, any two directions represent a 2D space which
describes any point on the earth - real 2D space, R^2 Given 2 vectors
you can create all that space using linear
combination.



Vector Span

Say we have a collection of vectors,

$\mathbf{v}$ = [v_1; v_2 …. ,v_n]

Then, the set W of all possible linear combinations of **v** is called the vector span.

A set of vectors is said to span a vector space, if we can write any vector in the vector space as a linear combination of the set. A

spanning set can be redundant.

Linear Independence

If S={$\mathbf{v}_1$, $\mathbf{v}_2$, …, $\mathbf{v}_r$} is a nonempty set of vector, then the vector equation

$$k_1\mathbf{v}_1+k_2\mathbf{v}_2+\ldots+k_r\mathbf{v}_r=0$$

has at least one solution, namely (where all coefficients are zero)

$$k_1=0,\ \ k_2=0,\ \ldots,\ k_r=0$$

If this is the only solution, then S is called linearly independent set. If there are other solutions, then S is called a linear dependent set.

A basis of a vector space is a linearly independent set of vectors that span the vector space. In 2D space one basis set would be formed by the x and y coordinate axes.

As a rule, a vector space of dimensions N requires a basis of size N.

Orthogonality

Two vectors x Є X and y Є Y are orthogonal, if their inner product is zero,

x.y = 0

Gram-Schmidt Orthogonalization: Used to convert a set of independent vectors into a set of orthogonal vectors that span the same vector space. Start with n independent vectors $\{y_1, y_2, \ldots, y_m\}$, to obtain n orthogonal vectors $\{v_1, v_2, \ldots, v_m\}$.

Step 1: Choose $v_1 = y_1$

Step 2: Subtract the component of y_2 that lies in the direction of v_1,

$$v_2 = y_2 - av_1$$

a is chosen such that v_2 is $\perp$ to v_1.

In general:

$$v_k = y_k - \Sigma_{i=1}^{k-1} \frac{v_i, y_k}{v_i, v_i} v_i$$

Linear Transformations

In a linear transformation, we have,

- A set of elements $X = \chi_i$, called the domain.
- A set of elements $Y = y_i$, called the range.
- A rule relating each $\chi_i \in X$ to an element of $y_i \in Y$.

Matrix multiplication is an example of a linear transformation.

A transformation $\mathbb{A}$ is linear if, for all $\chi_1, \chi_2 \in X$,

- $\mathbb{A}(\chi_1 + \chi_2) = \mathbb{A}(\chi_1) + \mathbb{A}(\chi_2)$
- $\mathbb{A}(a\chi_1) = a\mathbb{A}(\chi_1)$ where a is a real number.

Linear transformations in Neural Networks

Let A be a linear transformation with domain X and range Y , such that,

$$A(X) = y$$

In a neural network, the input vector is the domain and the output is

the range. The weight matrix is the linear transformation that relates

an element of the input with an element of the output.

The characteristic equation,

$$|A - \lambda \boldsymbol{I}| = 0$$

where λ is the eigenvalue of the linear transformation and **I** is the identity matrix.

The eigenvalues of the transformation can help us determine the

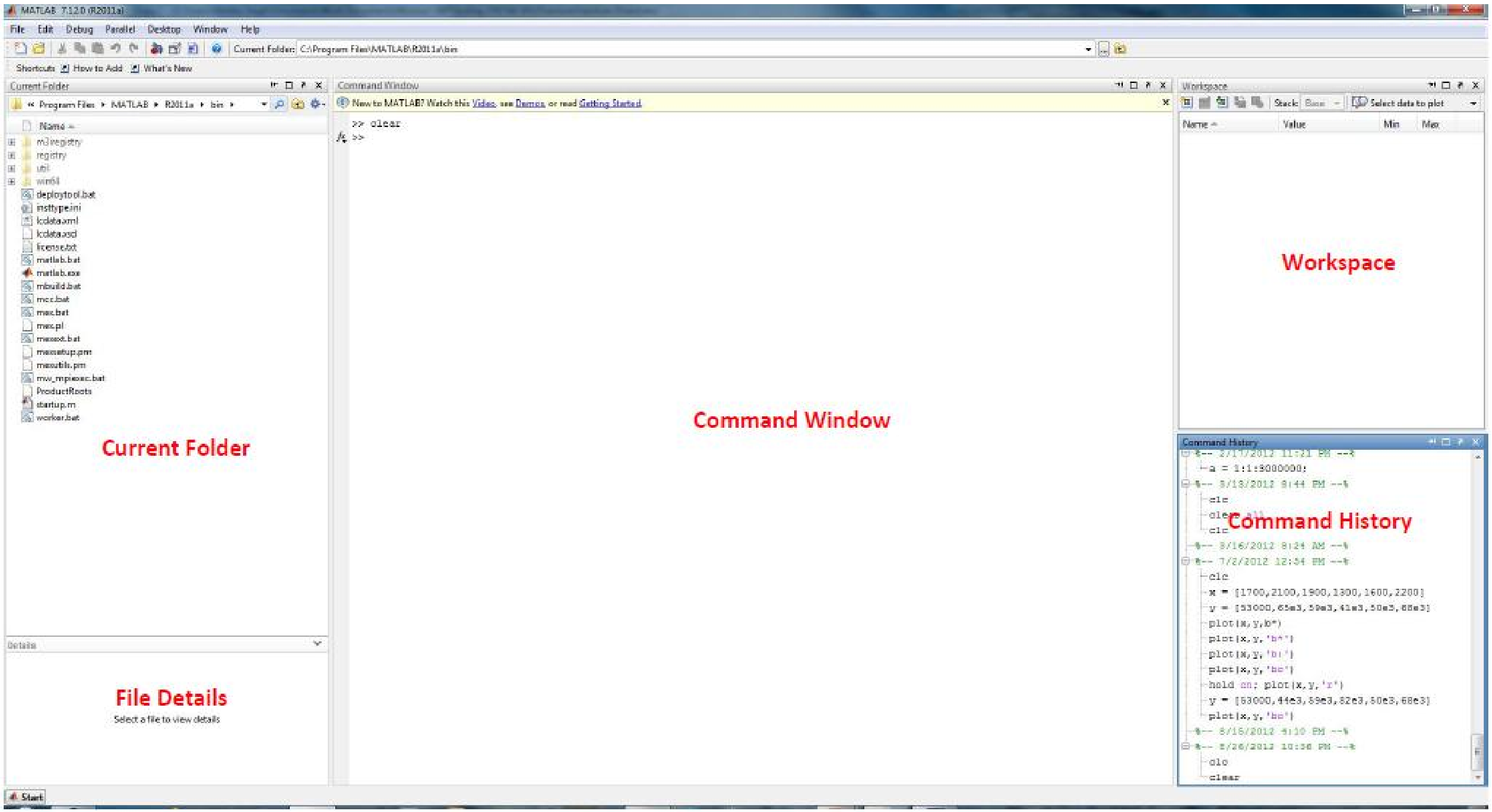
behavior of the output- whether it will converge, go to infinity, or oscillate.

Introduction to MATLAB

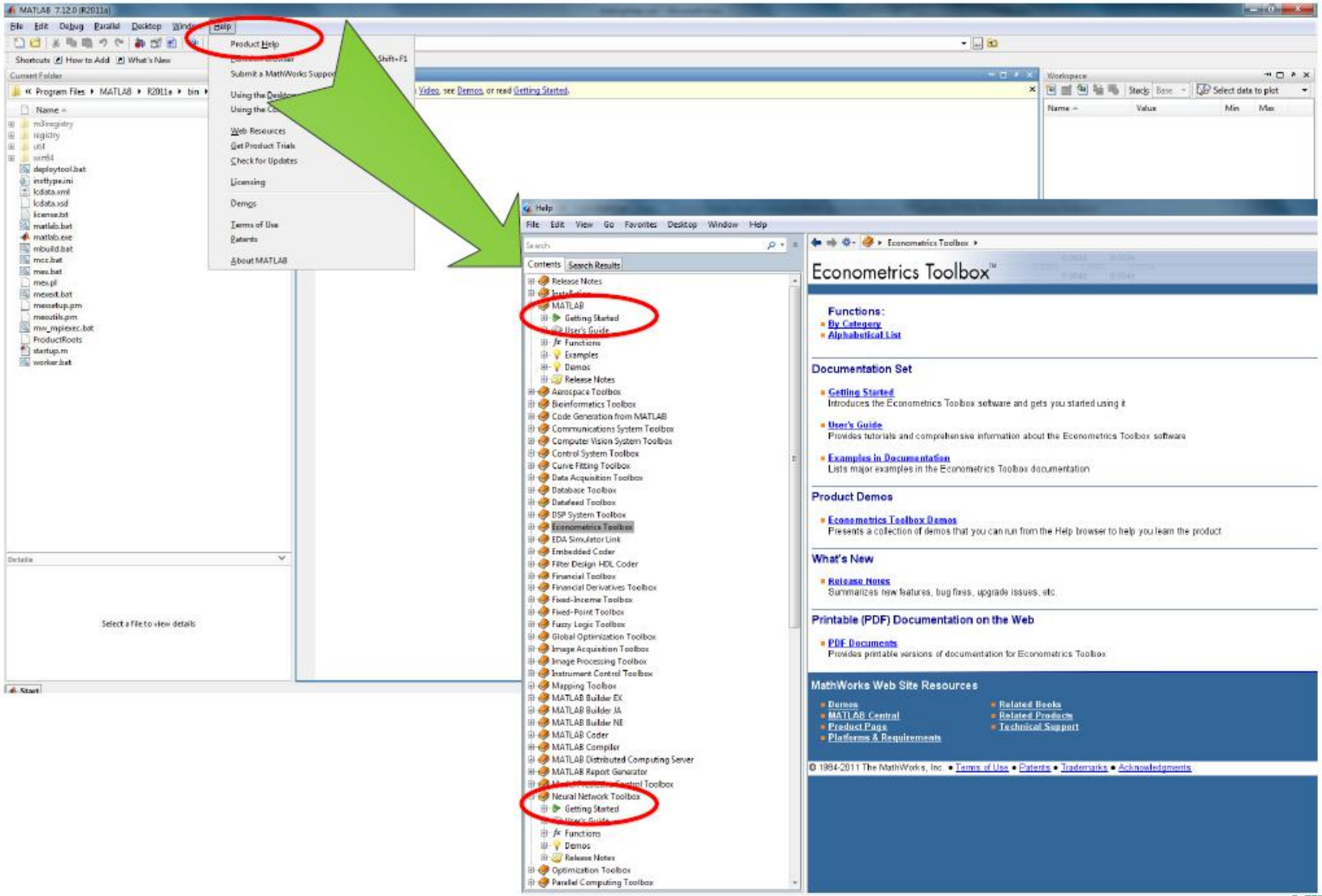
MATLAB™: MATrix LABoratory

- Numerical computing environment
- Developed by Mathworks , Inc.
- 4GL programming language designed for manipulating matrices
- Extended to incorporate numerous specialized toolboxes
- neural networks, statistics, control systems, power systems etc.
- Case-sensitive, starting R2011b

The MATLAB Interface

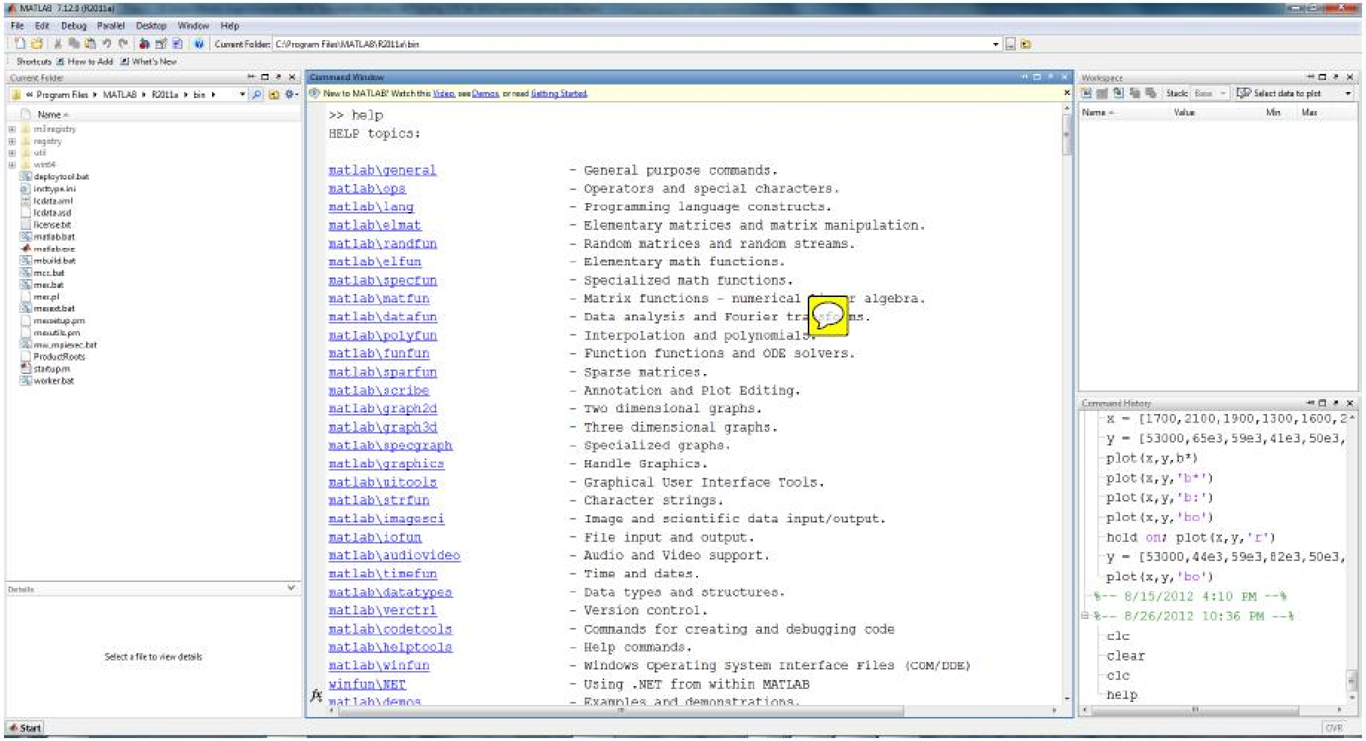


Getting Help



The Help Command

- Type help at the command prompt



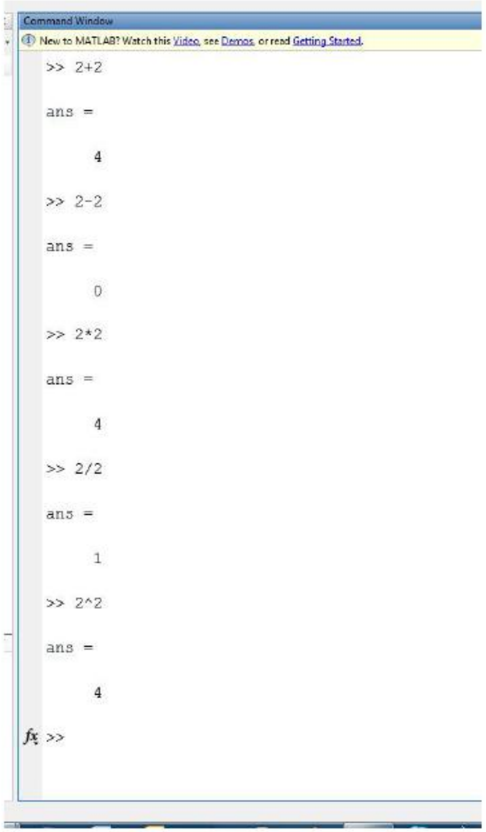
For help with specific functions, e.g. sin, type help sin

Variables

- Supported types include int, long, oat, double, char, string
- Dynamically typed
  - No need to declare type
- Variables names cannot begin with a number
- The underscore( ) is the only special character permitted in variable names
- Keyword and function names are reserved, and should not be used as variable names.

Arithmetic Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
\	left division
^	Power
()	Order of operation



Standard Functions

MATLAB has a large library of standard mathematic

- sqrt,sin,cos,abs,exp

To view a list of elementary mathematical functio

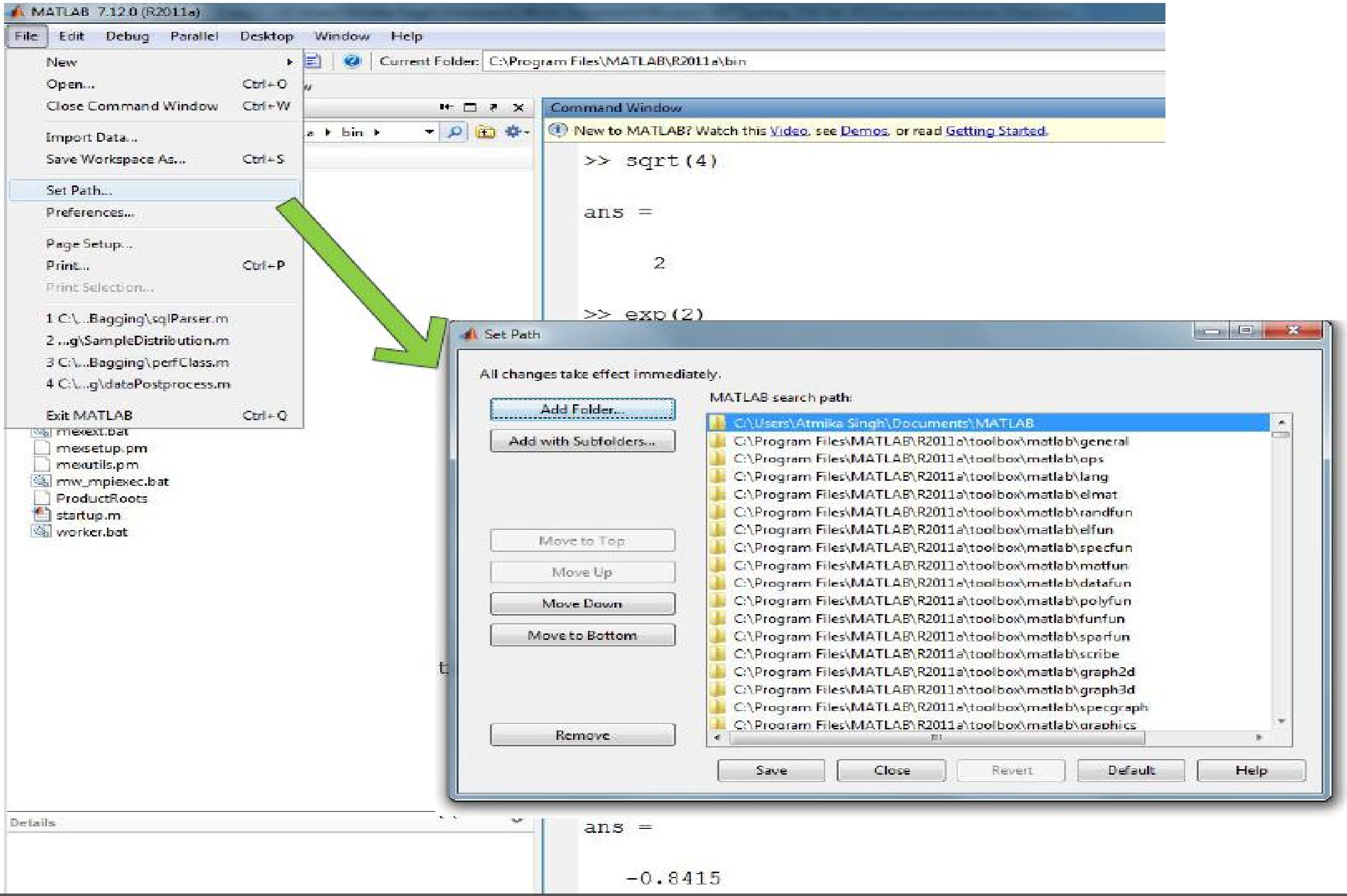
- help elfun
- help specfun: specialized mathematical functions
- help elmat: elementary mathematical functions

```
>> sqrt(4)

ans =

     2

>> exp(2)

ans =

    7.3891

>> sin(pi/2)

ans =

     1

>> sin(5)+sqrt(42)

ans =

    5.5218

>> sin(cos(pi))

ans =

   -0.8415
```

A Few Useful Commands

- clc: clear command window
- clear: clear workspace
- close: close all open gure windows
- who: returns the list of variables in the workspace

Accessing your Directory



Working with Arrays and Matrices

Generating Matrices:

Command Window

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>> mat1 = [1 3;4 5]

mat1 =

     1     3
     4     5

>> mat2 = [2 5]

mat2 =

     2     5

>> mat3 = 0:0.2:1

mat3 =

         0    0.2000    0.4000    0.6000    0.8000    1.0000

>> mat4 = sin(mat3)

mat4 =

         0    0.1987    0.3894    0.5646    0.7174    0.8415
```

Elements in the same row are separated by spaces or commas

Multiple rows are separated by semicolons

Generate arrays automatically by specifying a range and a step size value

Matrix Operations

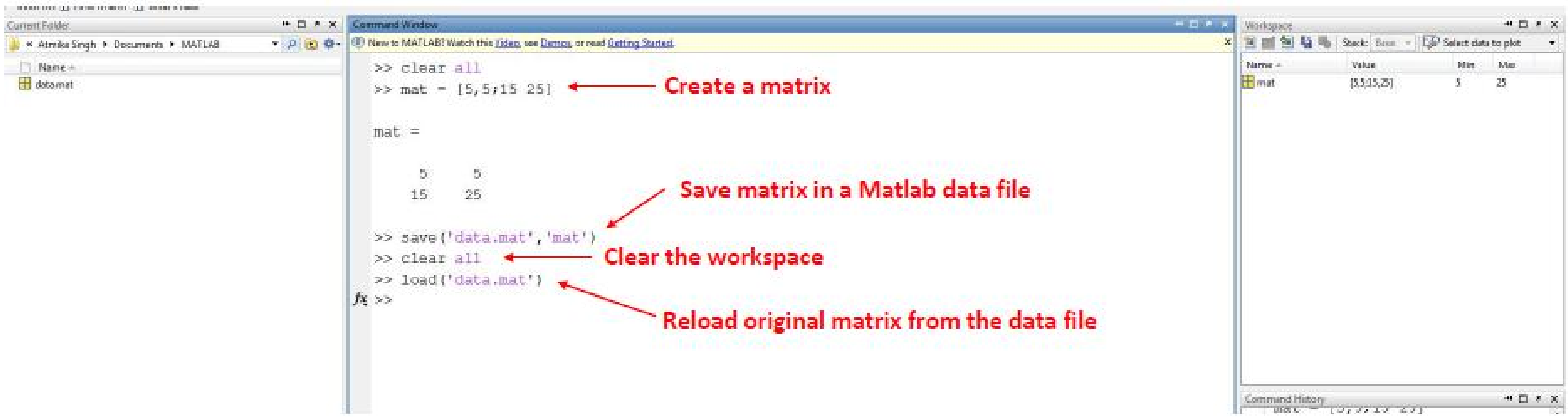
```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> mat1 = [2 3;1 4]

mat1 =

     2     3
     1     4

>> mat2 = [1 3;2 3]

mat2 =

     1     3
     2     3

>> mat3 = mat1+mat2

mat3 =

     3     6
     3     7

>> mat4 = mat1*mat2

mat4 =

     8    15
     9    15

fx >>
```

Arithmetic operations on matrices

```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> mat5 = 5*mat1

mat5 =

    10    15
     5    20

>> mat6 = mat5/5

mat6 =

     2     3
     1     4

>> mat7 = sin(mat1)

mat7 =

    0.9093    0.1411
    0.8415   -0.7568

>> mat8 = exp(mat2)

mat8 =

    2.7183   20.0855
    7.3891   20.0855

fx >>
```

Applying functions to matrices

Loading and Saving Matrices



Indexing and Concatenation

Indexing

```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> mat = [1 3 5; 2 3 7; 4 2 9]

mat =

     1     3     5
     2     3     7
     4     2     9

>> mat1 = mat(2,3)

mat1 =

     7

>> mat2 = mat(1:3,2)

mat2 =

     3
     3
     2

>> mat2 = mat(3,:)

mat2 =

     4     2     9
```

Concatenation

```
>> mat = [5,5;15 25]

mat =

     5     5
    15    25

>> array1 = [15,5]

array1 =

    15     5

>> mat1 = [mat;array1]

mat1 =

     5     5
    15    25
    15     5

>> mat2 = [mat1 [1; 5; 1]]

mat2 =

     5     5     1
    15    25     5
    15     5     1

fx >>
```

Deletion

```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> mat = [1 3 5 4; 2 3 7 2; 4 2 9 6; 9 5 7 3]

mat =

     1     3     5     4
     2     3     7     2
     4     2     9     6
     9     5     7     3

>> mat1 = mat;
>> mat1(:,2) = []

mat1 =

     1     5     4
     2     7     2
     4     9     6
     9     7     3

>> mat1(3,:) = []

mat1 =

     1     5     4
     2     7     2
     9     7     3

fx >>
```

Some Linear Algebra Functions

```
>> mat

mat =

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

>> mat1 = mat'

mat1 =

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

>> det(mat)

ans =

  1.0871e-012

>> inv(mat)
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 9.796086e-018.
```

Transpose

Determinant

Inverse

```
>> eig(mat)

ans =

   34.0000
    8.0000
    0.0000
   -8.0000

fx >>
```

Eigenvalues

Useful Matrix Functions

```
>> ones

ans =

     1

>> ones(2,2)

ans =

     1     1
     1     1

>> zeros(2,2)

ans =

     0     0
     0     0

>> eye(2,2)

ans =

     1     0
     0     1

>> rand

ans =

    0.8147
```

Matrix of ones

Zero matrix

Identity matrix

Random number from the interval (0,1)

```
>> rand(2,2)

ans =

    0.9436    0.9577
    0.6377    0.2407

>> randn(2)

ans =

    0.5265    0.6001
   -0.2603    0.5939

>> randi(100)

ans =

     7

>> randi(100,2,1)

ans =

    26
    23
```

Random number matrix

Normally distributed random numbers

Uniformly distributed integers

Array Operators

+	Addition
-	Subtraction
.*	element by element multiplication
./	element by element division
.\	element by element left division
.^	element by element power

Array Operators

```
>> array1 = 1:1:10

array1 =

     1     2     3     4     5     6     7     8     9    10

>> array2 = 0.1:0.1:1

array2 =

  Columns 1 through 9

    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000    0.9000

  Column 10

    1.0000

>> array1.*array2        <-- Element-wise multiplication

ans =

  Columns 1 through 9

    0.1000    0.4000    0.9000    1.6000    2.5000    3.6000    4.9000    6.4000    8.1000

  Column 10

   10.0000
```

```
>> array1./array2        <-- Element-wise division

ans =

  Columns 1 through 9

   10.0000   10.0000   10.0000   10.0000   10.0000   10.0000   10.0000   10.0000   10.0000

  Column 10

   10.0000

>> array1.^2        <-- Element-wise power

ans =

     1     4     9    16    25    36    49    64    81   100
```

Multivariate Data

- Neural network inputs frequently consist of multivariate data. Each column in a data set represents a variable or feature and each row represents a sample or instance or observation.
- Example:

Heart rate

Weight

Hours of exercise per week

Heart Rate	Weight	Hours of Exercise per week
72	134	3.2
81	201	3.5
69	156	7.1
82	148	2.4
75	170	1.2

Data Analysis Functions

```
D =

   72.0000  134.0000    3.2000
   81.0000  201.0000    3.5000
   69.0000  156.0000    7.1000
   82.0000  148.0000    2.4000
   75.0000  170.0000    1.2000

>> mu = mean(D)

mu =

   75.8000  161.8000    3.4800

>> sigma = std(D)

sigma =

    5.6303   25.4990    2.2107
```

Columns represent data variables

Gives the mean of each column

Gives the standard deviation of each column

- Type `datafun` at the prompt for more data processing functions.

Displaying Data Graphically

- MATLAB provides a powerful graphics library for creating and annotating graphs. With MATLAB we can,
  - Create 2D and 3D plots
  - Plot multiple datasets on one graph
  - Display multiple plots in one gure
  - Customize lines styles and colors
  - Add axis labels and titles
  - Save and export gures in various formats (.g,.eps,.bmp,.jpg etc.)
- Most useful plotting functions: plot,mesh
- MATLAB PLOTTING DEMO: plotDemo.m

Flow Control

- Four types of control statements can be used to selectively execute blocks of code in a MATLAB file.
  - Conditional control: if, else, switch
  - Loop Control: for, while, continue, break
  - Error Control: try, catch
  - Program Termination: return

Conditional Flow

```
%% if Statement
% Generate a random number
a = randi(100, 1);
% If it is even, divide by 2
if rem(a, 2) == 0
    disp('a is even')
    b = a/2;
end

disp('The value of a is ')
a
%% elseif or else Statement
a = randi(100, 1);
if a < 30
    disp('small')
elseif a < 80
    disp('medium')
else
    disp('large')
end
disp('The value of a is ')
a
```

Loop Control

```
%% for Statement
for n = 3:32
    r(n) = rank(magic(n));
end

%% while Statement
a = 0; fa = -Inf;
b = 3; fb = Inf;
while b-a > eps*b
    x = (a+b)/2;
    fx = x^3-2*x-5;
    if sign(fx) == sign(fa)
        a = x; fa = fx;
    else
        b = x; fb = fx;
    end
end
```

Testing Equality

- Comparing two variables,
- `if A == B, ...`
- This works perfectly for scalars. Does not work for arrays and matrices! To compare arrays use,
- `if isequal(A, B), ...`
- Returns a logical 1 (true) or 0 (false).
- For more on programming, see MATLAB > Getting Started >Programming

Data sets for neural network training

- UC Irvine Machine Learning Repository

http://archive.ics.uci.edu/ml/index.html