Electrical Engineering Matlab Tutorial
Part I: Computation Environment

Michael D. DeVore

mdd2@essrl.wustl.edu

M. Ali Rauf

ar2@ee.wustl.edu

Tutorial online at: **http://cis.wustl.edu/~mdd2/matlab/matlab.html**

Outline

- Starting a Matlab Session
- Matlab Fundamentals
  - Creating Matrices
  - Matrix and Array Operations
  - Basic Functions
- Scripts and Functions
  - m-Files
  - Conditional Execution
- Visualization
  - Figures and Plotting
  - Axes and Labeling
  - Printing and Saving
- Other Data Types
- File I/O

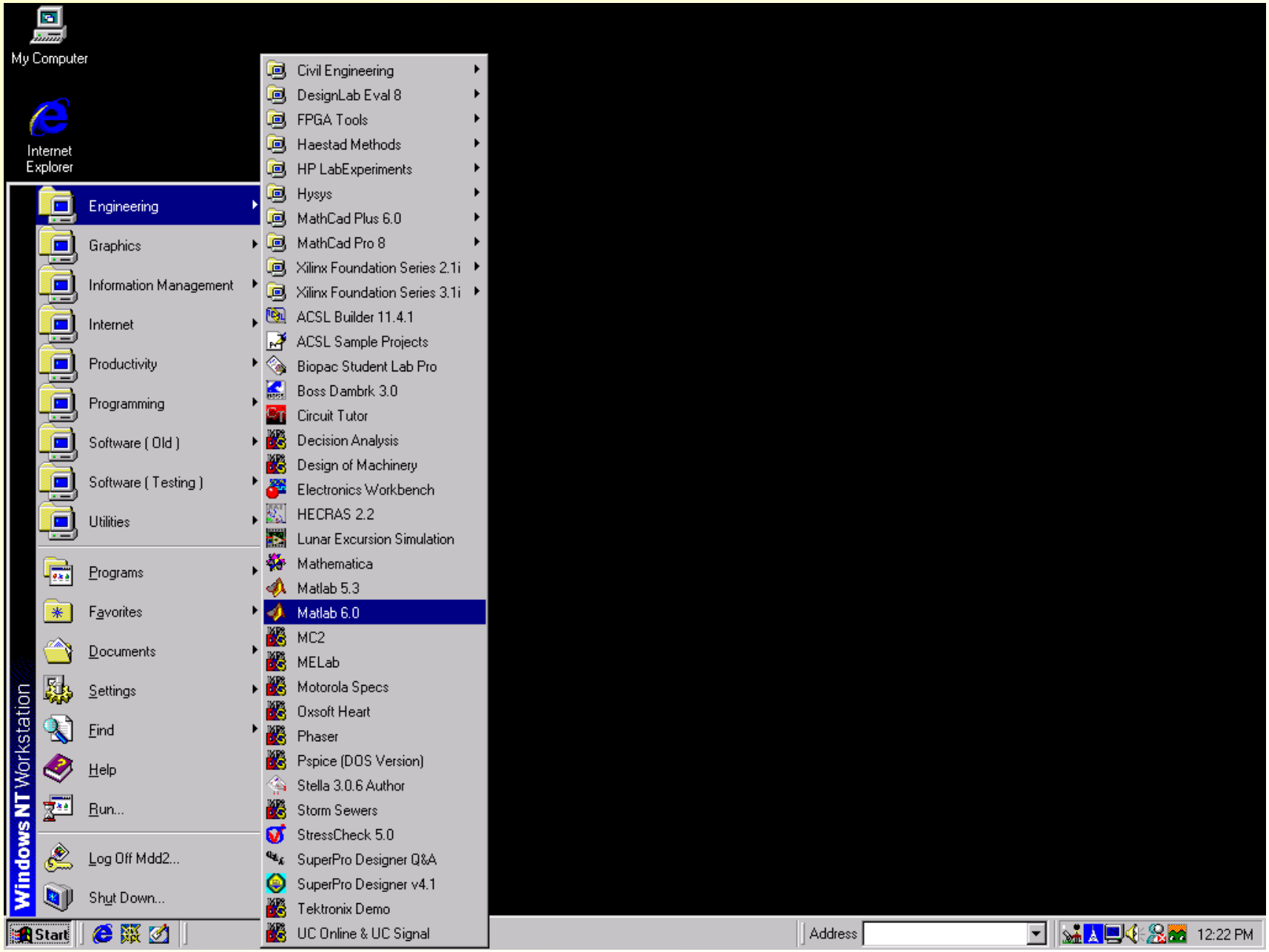
Starting Matlab

- On the Unix platform
  - Enter your CEC username and password
  - In a Terminal window, type (first time only)

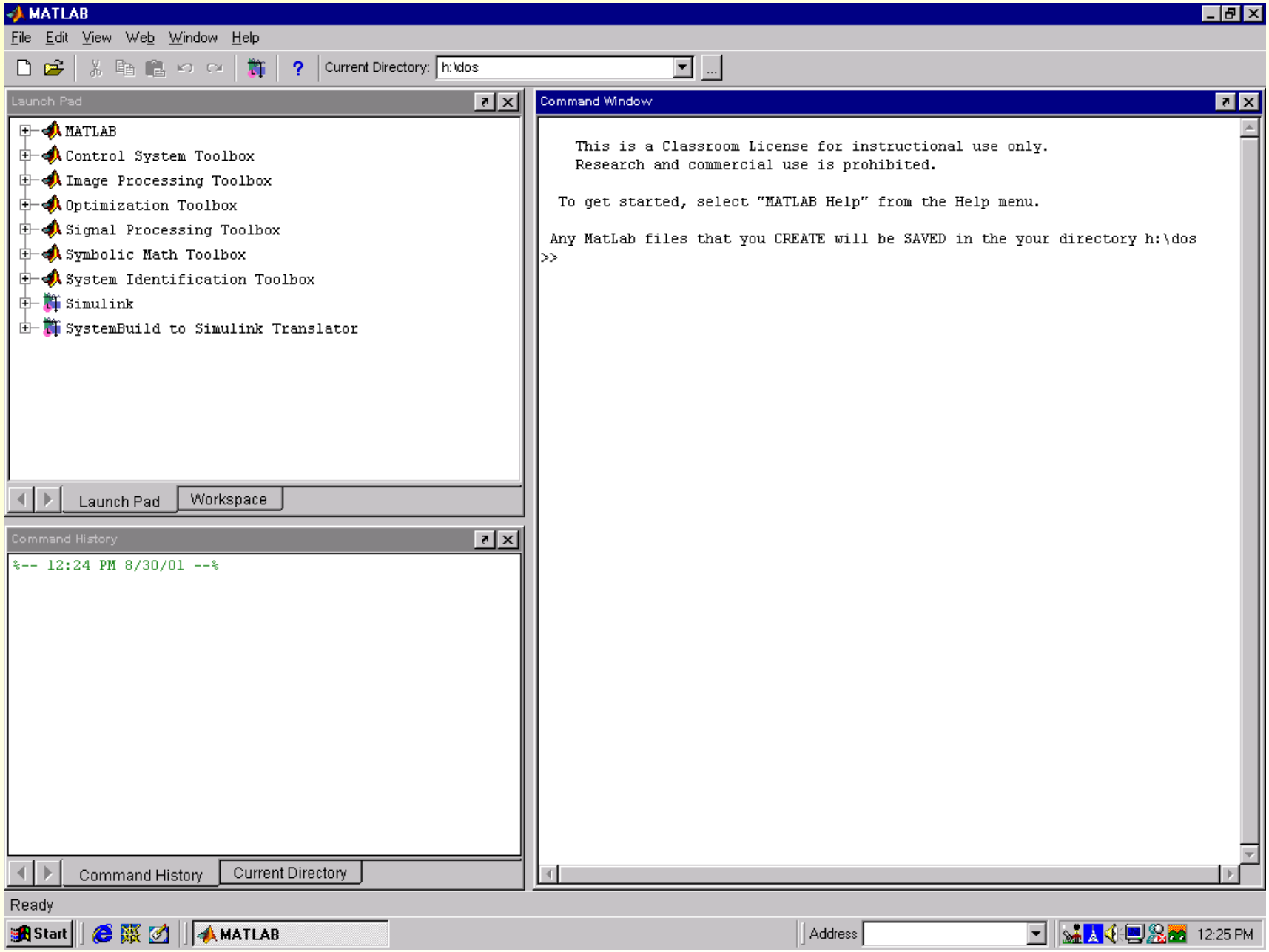
    `pkgaddperm matlab`
  - In a Terminal window, launch Matlab by typing

    `matlab`
- On the Windows platform
  - Press **Ctrl-Alt-Delete** to bring the login prompt
  - Enter your CEC username and password
  - Launch Matlab from the Start menu

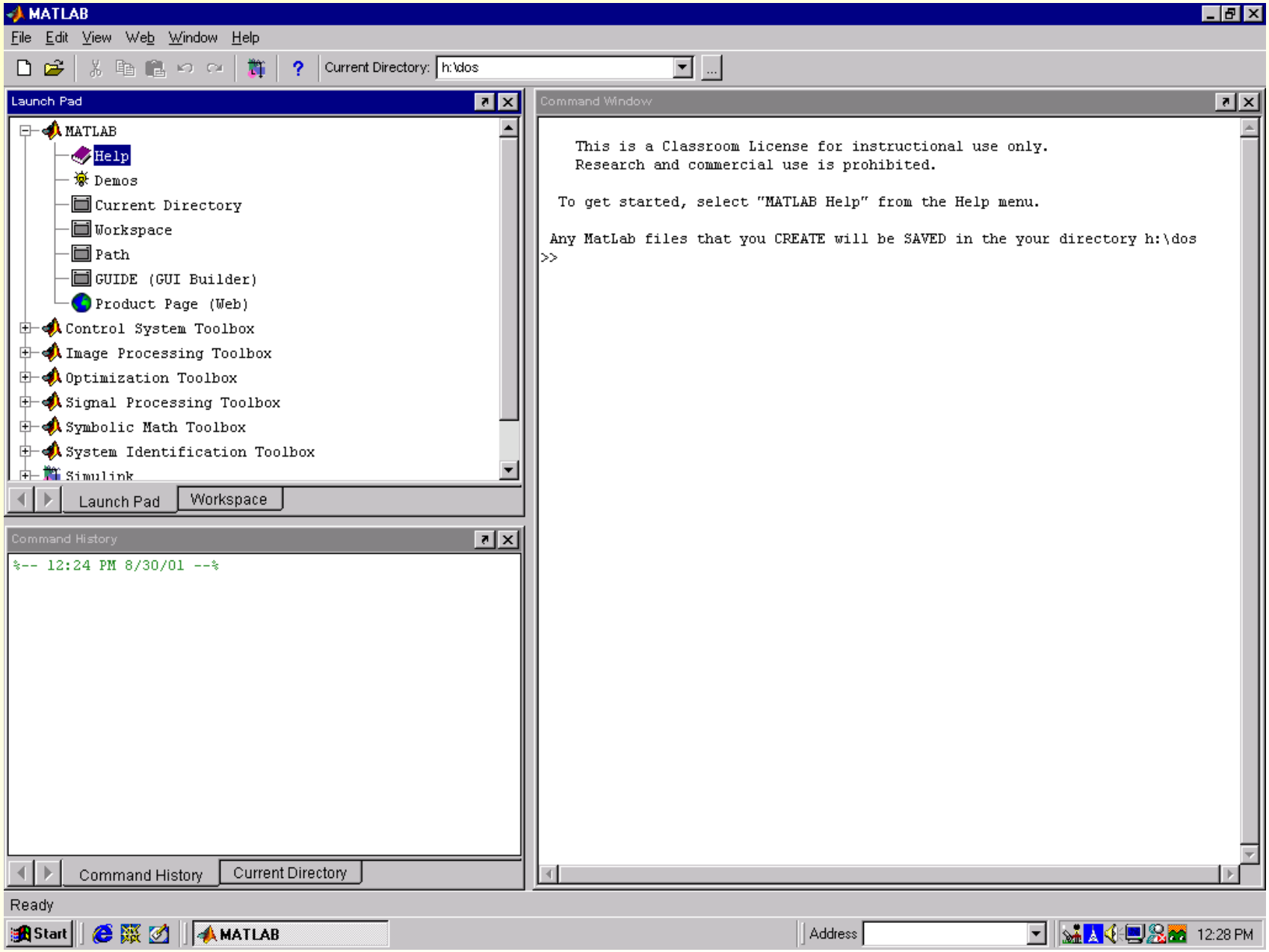
Starting Matlab



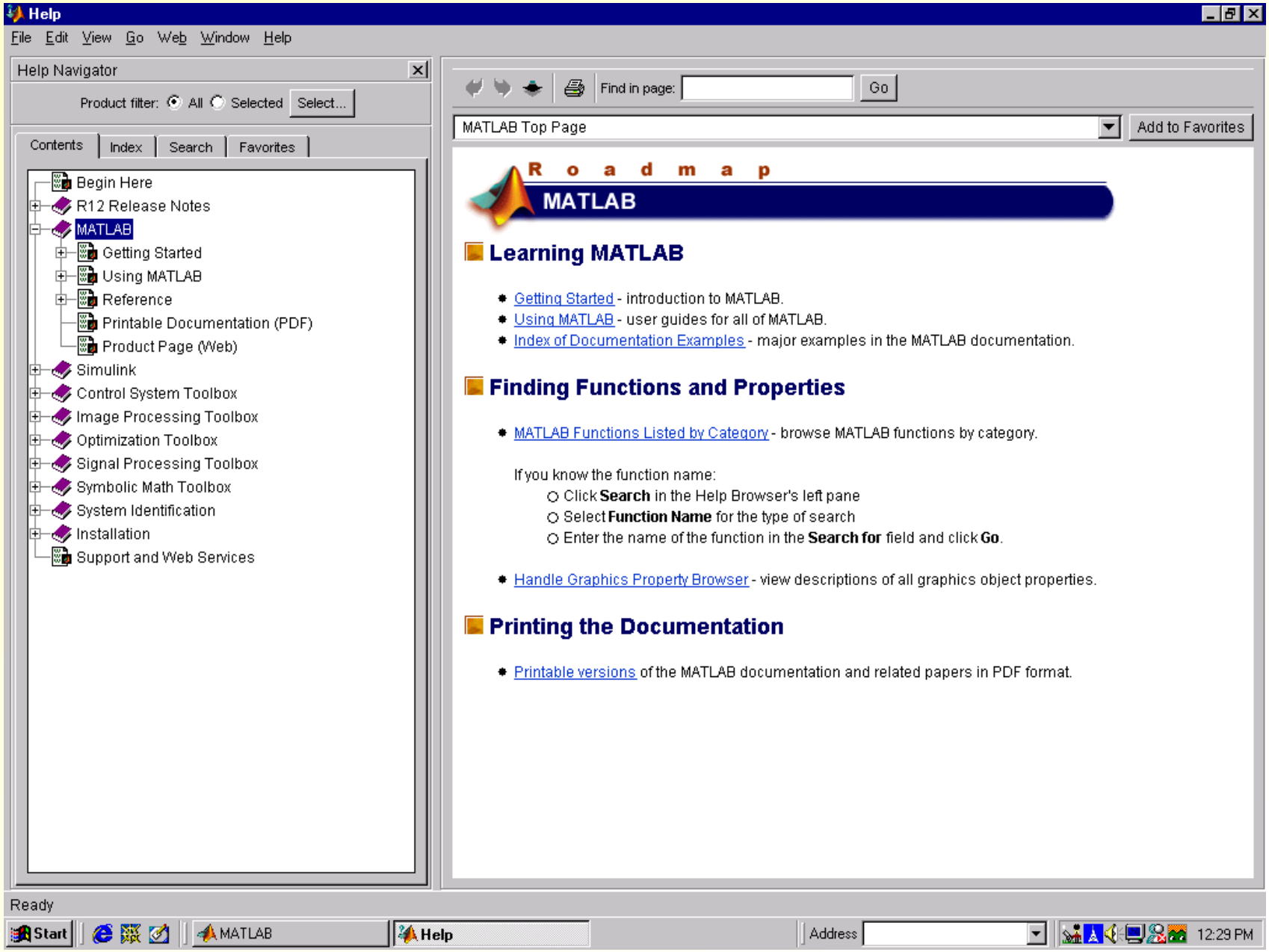
Starting Matlab



Starting Matlab



Starting Matlab



Starting Matlab

- Enter Matlab commands into the "Command Window"
- Matlab output appears under your command
- To capture all input and output, use the diary command

```
>> diary matlab_tutorial.txt
```

- To insert comments in the diary (and on screen) preceed them with a % sign

```
>> % We can capture all our activities in a file with
'diary filename'
```

Matlab Fundamentals - Creating Matrices

- The result of the last operation is stored as a variable with the name “ans” (as in answer)
- Typing a variable name shows its value in the command window

```
>> 5*5

ans =

    25
>> ans

ans =

    25
```

Matlab Fundamentals - Creating Matrices

- All variables in Matlab hold **arrays**, rectangular arrangements of data elements
- The **size** of an array with *m* rows and *n* columns is denoted by *m* x *n*
- A **matrix** is an array of numbers
- A **scalar** is a 1 x 1 matrix holding a single number

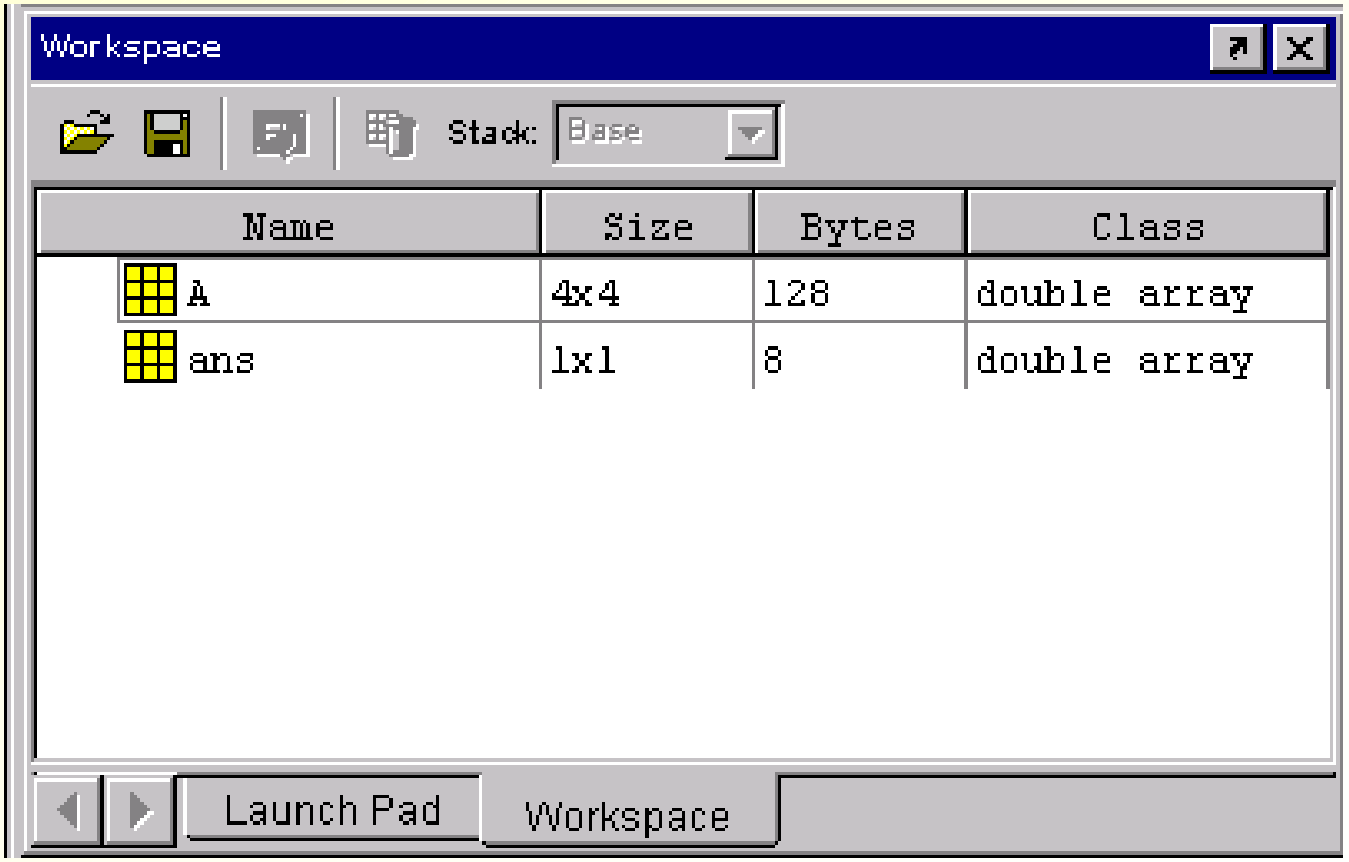
```
>> A=[1,2,3,4; 4,2,1,7; 9,8,9,3; 5,2,8,6]

A =

     1     2     3     4
     4     2     1     7
     9     8     9     3
     5     2     8     6
```

Matlab Fundamentals - Creating Matrices

- The workspace window lists current variables, their size, and the amount of memory they occupy



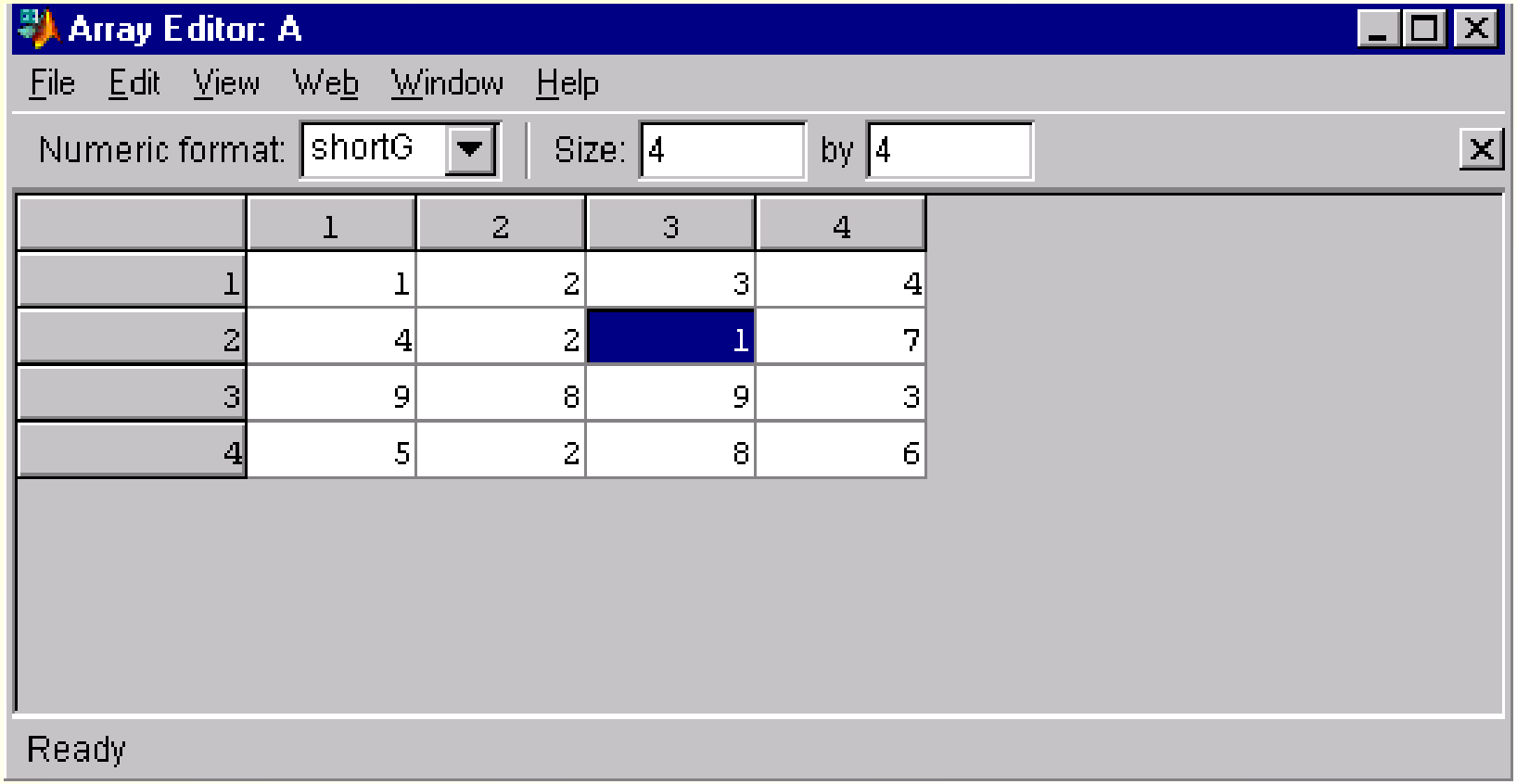
Matlab Fundamentals - Creating Matrices

- The screen output of a command will be suppressed if the command ends with a semi-colon
- Of course, the command is still evaluated

```
>> B=[1,4,9,5; 2,2,8,2; 3,1,9,8; 4,7,3,6];
```

Matlab Fundamentals - Creating Matrices

- Double-clicking on a variable name in the workspace window opens the **Array Editor**
- The Array Editor lets you directly enter or change array contents
- Click in an element and type any Matlab command



	1	2	3	4
1	1	2	3	4
2	4	2	1	7
3	9	8	9	3
4	5	2	8	6

Matlab Fundamentals - Creating Matrices

- A 1 x *n* or *n* x 1 matrix is called a **vector**
- The colon operator generates vectors with evenly spaced elements
- The format is *start*:*step*:*stop*

```
>> x=1:0.3:2

x =

    1.0000    1.3000    1.6000    1.9000
```

Matlab Fundamentals - Creating Matrices

- Matlab can work with **complex numbers** which have the form $a + b\sqrt{-1}$
- Use the Matlab symbols i or j to denote the imaginary number $i = j = \sqrt{-1}$
- Matlab is case sensitive
- Continue long lines by typing '…' before the return

```
>> 5+7i

ans =

   5.0000 + 7.0000i

>> C=[1+2i, 2+3i, 3+4i; 4+2j, 2+j, 1+7j; ...
i 1 j]

C =

   1.0000 + 2.0000i   2.0000 + 3.0000i   3.0000 + 4.0000i
   4.0000 + 2.0000i   2.0000 + 1.0000i   1.0000 + 7.0000i
   0      + 1.0000i   1.0000                   0 + 1.0000i
```

Matlab Fundamentals - Creating Matrices

- A number of other constants are already defined, including pi, eps, Inf, and NaN

```
>> [pi eps Inf NaN]

ans =

    3.1416    0.0000       Inf       NaN
```

Matlab Fundamentals - Creating Matrices

- Several functions exist to create matrices of size *m* x *n*
  - ones(m,n) makes a matrix with 1 in every element
  - zeros(m,n) makes a matrix with 0 in every element
  - rand(m,n) produces pseudo-random values uniformly between zero and 1
  - randn(m,n) produces Normal(0,1) random values

```
>> ones(3,5)

ans =

     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1

>> rand(3,5)

ans =

    0.9501    0.4860    0.4565    0.4447    0.9218
    0.2311    0.8913    0.0185    0.6154    0.7382
    0.6068    0.7621    0.8214    0.7919    0.1763
```

Matlab Fundamentals - Creating Matrices

- Variables can be saved in ".mat" files for later retrieval
- The memory occupied by a variable can be released with the command clear
- The command load will retrieve the value
- Saving, clearing, and loading without naming any variables operates on all of them in the workspace

```
>> save myfile.mat x
>> clear x
>> whos
  Name        Size              Bytes  Class

  A           4x4                 128  double array
  B           4x4                 128  double array
  C           3x3                 144  double array (complex)
  ans         1x4                  64  double array (complex)

Grand total is 49 elements using 464 bytes

>> load myfile x
```

Matlab Fundamentals - Operators

- Individual elements of arrays are **referenced** (**subscripted**) with parentheses ( )
- Arrays are **indexed** from 1
- The colon operator means select all matching elements

```
>> A(1,3)

ans =

     3

>> A(1,:)

ans =

     1     2     3     4
```

Matlab Fundamentals - Operators

- Arrays can be **concatenated** with brackets [ ]
- To concatenate horizontally, use a comma

```
>> [A, B]

ans =

     1     2     3     4     1     4     9     5
     4     2     1     7     2     2     8     2
     9     8     9     3     3     1     9     8
     5     2     8     6     4     7     3     6
```

Matlab Fundamentals - Operators

- Arrays can be **concatenated** with brackets [ ]
- To concatenate vertically, use a semi-colon

```
>> [A; B]

ans =

     1     2     3     4
     4     2     1     7
     9     8     9     3
     5     2     8     6
     1     4     9     5
     2     2     8     2
     3     1     9     8
     4     7     3     6
```

Matlab Fundamentals - Operators

- **Submatrices** can be extracted with the colon operator

```
>> A(1:3,2:4)

ans =

     2     3     4
     2     1     7
     8     9     3
```

Matlab Fundamentals - Operators

- Matrix addition and subtraction are defined for two matrices of the same size

```
>> A+B

ans =

     2     6    12     9
     6     4     9     9
    12     9    18    11
     9     9    11    12

>> A-B

ans =

     0    -2    -6    -1
     2     0    -7     5
     6     7     0    -5
     1    -5     5     0
```

Matlab Fundamentals - Operators

- Scalar operations on matrices apply to each element within the matrix
- We can compute quantities like 5+A, 5-A, A-5, 5*A, and A/5

```
>> 5+A

ans =

     6     7     8     9
     9     7     6    12
    14    13    14     8
    10     7    13    11

>> A/5

ans =

    0.2000    0.4000    0.6000    0.8000
    0.8000    0.4000    0.2000    1.4000
    1.8000    1.6000    1.8000    0.6000
    1.0000    0.4000    1.6000    1.2000
```

Matlab Fundamentals - Operators

- With matrix multiplication and division, there are two possibilities
  - * and / operate on entire matrices,
    A*B forms AB and A/B forms AB^{-1}
  - .* and ./ operate element-by-element
    A.*B forms $[a_{ij}\ b_{ij}]$ and A./B forms $[a_{ij} / b_{ij}]$

```
>> A/B

ans =

    0.3519   -0.3677    0.2573    0.1529
   -0.7233   -0.3374    0.9126    0.6650
   -1.8981    3.0073   -0.2427    1.4029
   -0.8981    1.0073    0.7573    0.4029

>> A./B

ans =

    1.0000    0.5000    0.3333    0.8000
    2.0000    1.0000    0.1250    3.5000
    3.0000    8.0000    1.0000    0.3750
    1.2500    0.2857    2.6667    1.0000
```

Matlab Fundamentals - Operators

- A variety of output formats are available
  - Use “format long” to see more detail
  - Use “format short” to hide detail

```
>> format long
>> A./B

ans =

   1.00000000000000   0.50000000000000   0.33333333333333
   0.80000000000000
   2.00000000000000   1.00000000000000   0.12500000000000
   3.50000000000000
   3.00000000000000   8.00000000000000   1.00000000000000
   0.37500000000000
   1.25000000000000   0.28571428571429   2.66666666666667
   1.00000000000000

>> format short
```

Matlab Fundamentals - Operators

- The **transpose** operator interchanges the rows and columns of a matrix
  - Use ' (a single quote) for **conjugate transpose**
  - Use .' (with a period) for transpose
  - These are equivalent for arrays of real numbers

```
>> C'

ans =

   1.0000 - 2.0000i   4.0000 - 2.0000i   0 - 1.0000i
   2.0000 - 3.0000i   2.0000 - 1.0000i   1.0000
   3.0000 - 4.0000i   1.0000 - 7.0000i   0 - 1.0000i

>> C.'

ans =

   1.0000 + 2.0000i   4.0000 + 2.0000i   0 + 1.0000i
   2.0000 + 3.0000i   2.0000 + 1.0000i   1.0000
   3.0000 + 4.0000i   1.0000 + 7.0000i   0 + 1.0000i
```

Matlab Fundamentals - Operators

- With matrix power, there are two possibilities
  - ^ operates on entire matrices, A^2 forms AA
  - .^ operates element-by-element, A.*2 forms $[a_{ij}^{\,2}]$

```
>> A^2

ans =

    56    38    64    51
    56    34    79    75
   137   112   140   137
   115    90   137    94

>> A.^2

ans =

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

Matlab Fundamentals - Operators

- With matrix power, there are two possibilities
  - ^ operates on entire matrices, A^2 forms AA
  - .^ operates element-by-element, A.*2 forms $[a_{ij}^{\,2}]$

```
>> A^2

ans =

    56    38    64    51
    56    34    79    75
   137   112   140   137
   115    90   137    94

>> A.^2

ans =

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

Matlab Fundamentals - Functions

- The function sum( ) adds elements in a matrix
  - sum(A) adds up the columns
  - sum(A,2) adds up the rows
- If A is a vector, sum( ) adds its elements so sum(sum(A)) adds up all elements in matrix A

```
>> sum(A)

ans =

    19    14    21    20

>> sum(A,2)

ans =

    10
    14
    29
    21
```

Matlab Fundamentals - Functions

- The functions prod( ) and mean( ) work similarly
  - prod(A) multiplies along the columns
  - prod(A,2) multiplies along the rows
  - mean(A) takes averages along the columns
  - mean(A,2) takes averages along the rows

```
>> prod(A)

ans =

   180    64   216   504

>> mean(A)

ans =

    4.7500    3.5000    5.2500    5.0000
```

Matlab Fundamentals - Functions

- The function max( ) computes maximum values
  - max(A) finds the maximum of each column
  - max(A,5) finds the maximum of each element and the number 5
  - max(A,[],2) finds the maximum of each row

```
>> max(A)

ans =

     9     8     9     7

>> max(A,5)

ans =

     5     5     5     5
     5     5     5     7
     9     8     9     5
     5     5     8     6
```

Matlab Fundamentals - Functions

- The function diag( ) extracts the diagonal elements of a matrix
- The function det( ) computes the determinant $|A|$ of a square matrix A

```
>> diag(A)

ans =

     1
     2
     9
     6
>> det(A)

ans =

  -824
```

Matlab Fundamentals - Functions

- A large number of functions from trigonometry are available
- The functions sin( ), cos( ), tan( ), etc. all assume angles measured in radians
- The functions are applied element-by-element

```
>> sin(A)

ans =

    0.8415    0.9093    0.1411   -0.7568
   -0.7568    0.9093    0.8415    0.6570
    0.4121    0.9894    0.4121    0.1411
   -0.9589    0.9093    0.9894   -0.2794
```

Matlab Fundamentals - Functions

- A large number of exponential related functions are available
  - exp( ) raises *e* to a power, element-by-element
  - log( ) computes the natural logarithm
  - log10( ) computes logarithms, base 10
  - sqrt( ) computes square roots

```
>> log(A)

ans =

         0    0.6931    1.0986    1.3863
    1.3863    0.6931         0    1.9459
    2.1972    2.0794    2.1972    1.0986
    1.6094    0.6931    2.0794    1.7918
```

Matlab Fundamentals - Functions

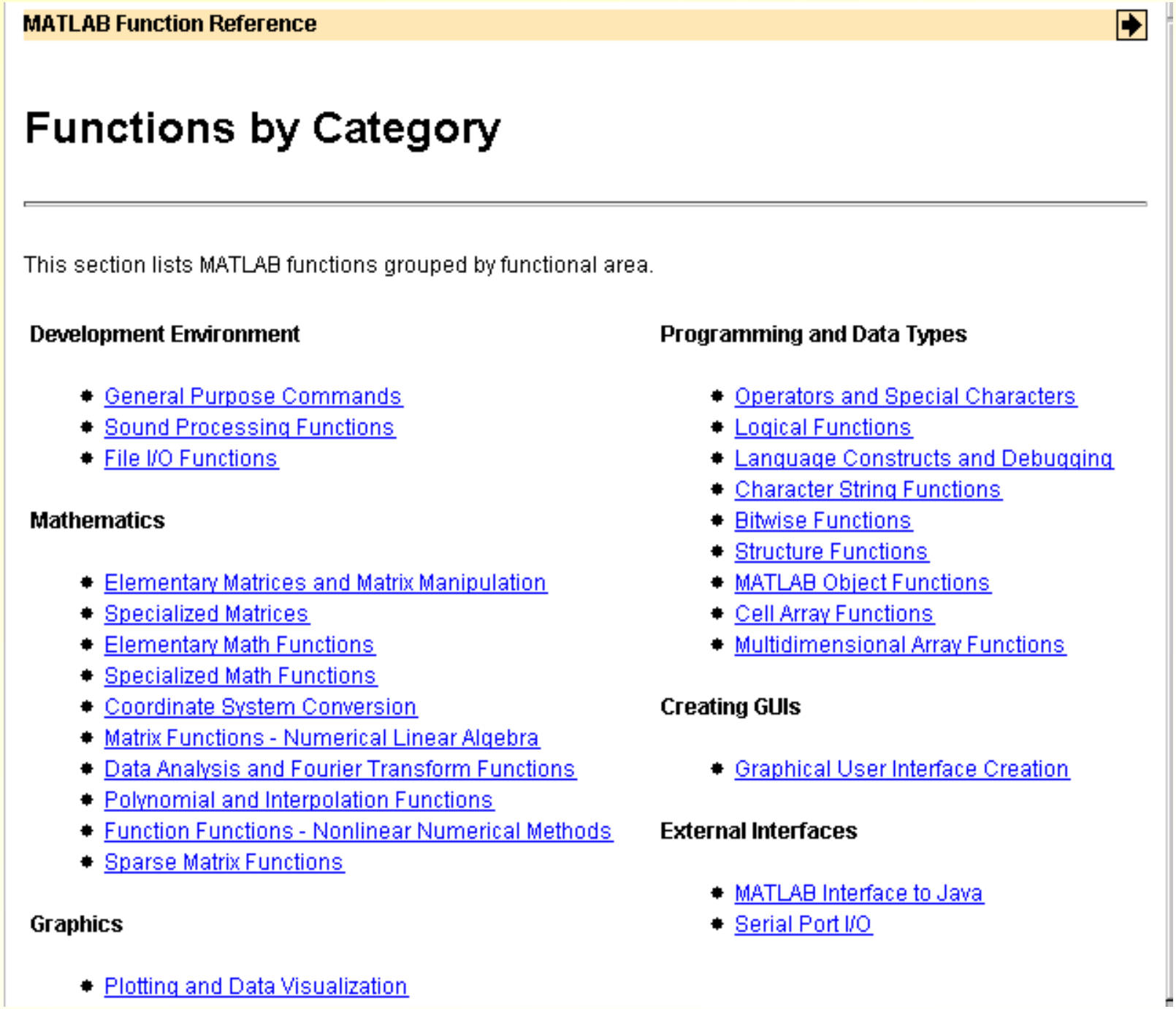
- A large number of functions for complex numbers are available and operate element-by-element
  - abs( ) computes the magnitude
  - phase( ) computes the phase angle in radians
  - imag( ) returns the imaginary part
  - real( ) returns the real part

```
>> phase(C)

ans =

    1.1071    0.9828    0.9273
    0.4636    0.4636    1.4289
    1.5708         0    1.5708
```

Matlab Fundamentals - Functions



MATLAB Function Reference

Functions by Category

This section lists MATLAB functions grouped by functional area.

Development Environment

- General Purpose Commands
- Sound Processing Functions
- File I/O Functions

Mathematics

- Elementary Matrices and Matrix Manipulation
- Specialized Matrices
- Elementary Math Functions
- Specialized Math Functions
- Coordinate System Conversion
- Matrix Functions - Numerical Linear Algebra
- Data Analysis and Fourier Transform Functions
- Polynomial and Interpolation Functions
- Function Functions - Nonlinear Numerical Methods
- Sparse Matrix Functions

Graphics

- Plotting and Data Visualization

Programming and Data Types

- Operators and Special Characters
- Logical Functions
- Language Constructs and Debugging
- Character String Functions
- Bitwise Functions
- Structure Functions
- MATLAB Object Functions
- Cell Array Functions
- Multidimensional Array Functions

Creating GUIs

- Graphical User Interface Creation

External Interfaces

- MATLAB Interface to Java
- Serial Port I/O