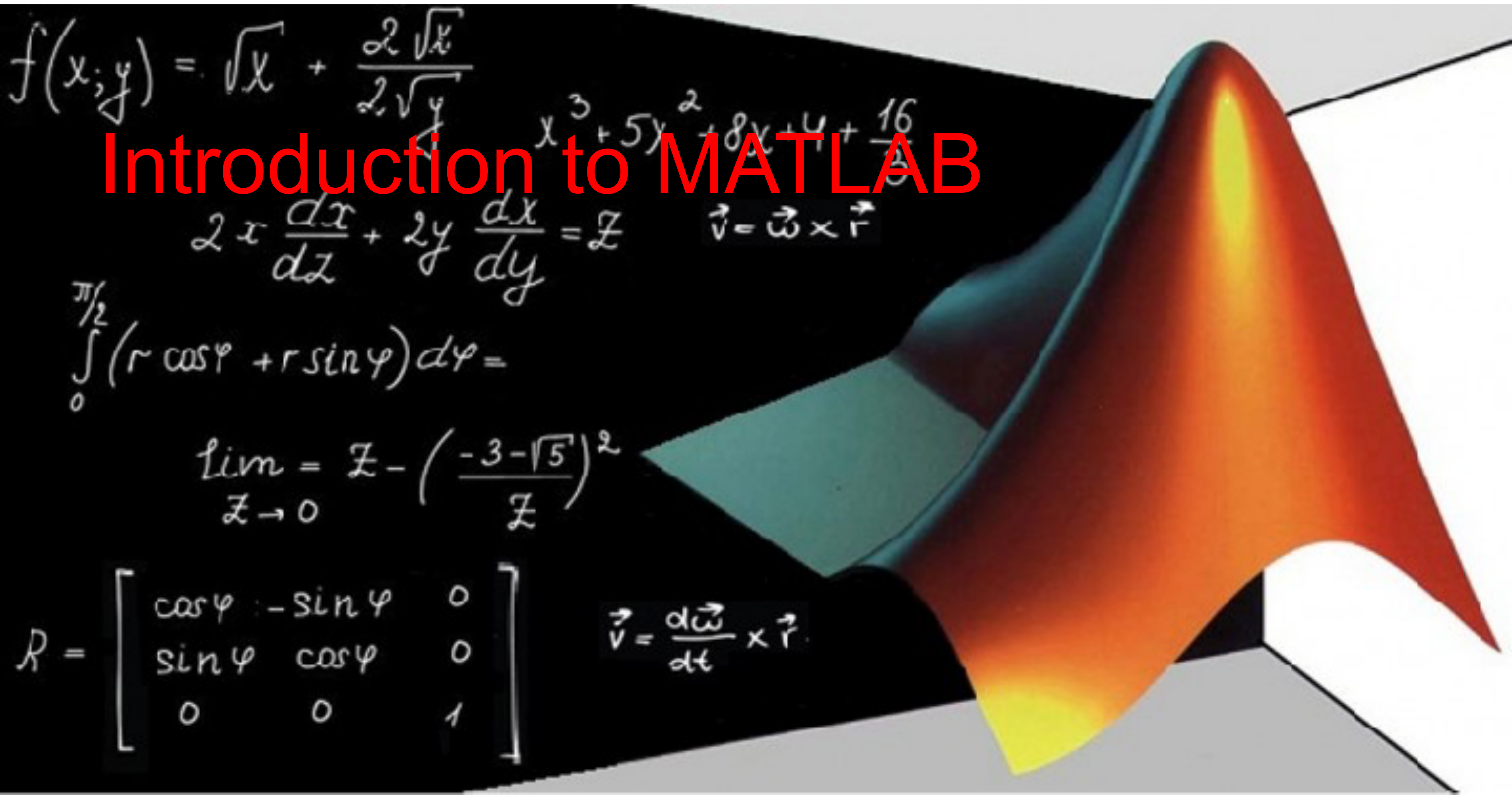
Introduction to MATLAB



Instructor: Dr. Peter Beerli

(slides: Dr. Ming Ye)

History of MATLAB: Fortran and Scientific Computing

- Engineering and scientific applications involve a lot of "number crunching".
- For many years, the main language for this was FORTRAN -- first "high level" programming language, and especially designed for numerical computing.
- Here's a Fortran code to solve a $x^2 + b x + c = 0$:

```
C      Solve a quadratic equation (this is a comment).
       DESC = B*B - 4*A*C
       IF ( DESC .LT. 0.0 ) GOTO 10
         DESC = SQRT(DESC)
         X1 = (-B + DESC)/(2.0*A)
         X2 = (-B - DESC)/(2.0*A)
         WRITE(6,*) "SOLUTIONS ARE ",X1," AND ", X2
         RETURN
    10 WRITE(6,*) "EQUATION HAS COMPLEX ROOTS"
       RETURN
```

Problems using FORTRAN

"Number crunching" on a computer can be tricky.

Problems that occur are:

- loss of precision and inaccurate results:

  ```
  X = 0.1
  Y = 1.0 - 10*X
  ```

  Y "should" equal 0, but probably does not!
- underflow and overflow: `X = 1.0E20, X*X` **-->** too big!
- efficient coding of algorithms not always obvious
- programming errors!

Numerical Libraries

- The U.S. government recognized these problems, and the inefficiency of many engineers all writing the *same* algorithms... again and again.
- So, they commissioned *numerical analysts* to write good quality algorithms for common tasks.
- Make the results freely available as "libraries" of subroutines that anyone can use in their programs.
- Libraries are available at: www.netlib.org

Examples of Numerical Libraries

- BLAS (Basic Linear Algebra Subroutines): operations on vectors, like adding to vectors, dot product, norm.
- LINPACK: linear algebra subroutines for vector-matrix operations, solving linear systems, factoring a matrix, inverting a matrix. Later replaced by LAPACK.
- EISPACK: compute eigenvalues and eigenvectors of matrices.
- Example: solve A*x = b using LINPACK

```
C.... factor the A matrix
      CALL SGEFA(A, N, N, IPVT, INFO)
C.... copy B vector into X vector
      CALL SCOPY(N, B, 1, X, 1)
C.... solve the system of equations
      CALL SGESL(A, N, N, IPVT, X, 0)
```

MATLAB (**Mat**rix **Lab**oratory)

History of MATLAB (mainly from Wikipedia)

- Ancestral software to MATLAB
  - Fortran subroutines for solving linear (LINPACK) and eigenvalue (EISPACK) problems
  - Developed primarily by Cleve Moler in the 1970's, mathematician, once chairman of the computer science department at the University of New Mexico
- When teaching courses in mathematics, Moler wanted his students to be able to use LINPACK and EISPACK without requiring knowledge of Fortran
- MATLAB developed as an interactive system to access LINPACK and EISPACK
- It soon spread to other universities and found a strong audience within the applied mathematics community.
- MATLAB gained popularity primarily through word of mouth because it was not officially distributed

History of MATLAB

- Jack Little, an engineer, was exposed to it during a visit Moler made to Stanford University in 1983. Recognizing its commercial potential, he joined with Moler and Steve Bangert. They rewrote MATLAB in C with more functionality (such as plotting routines) and founded The MathWorks in 1984 to continue its development.
- The Mathworks is now responsible for development, sale, and support for MATLAB
- MATLAB was first adopted by control design engineers, Little's specialty, but quickly spread to many other domains. It is now also used in education, in particular the teaching of linear algebra and numerical analysis.
- A standard tool in both professional and academic use with millions of users

Software Principles

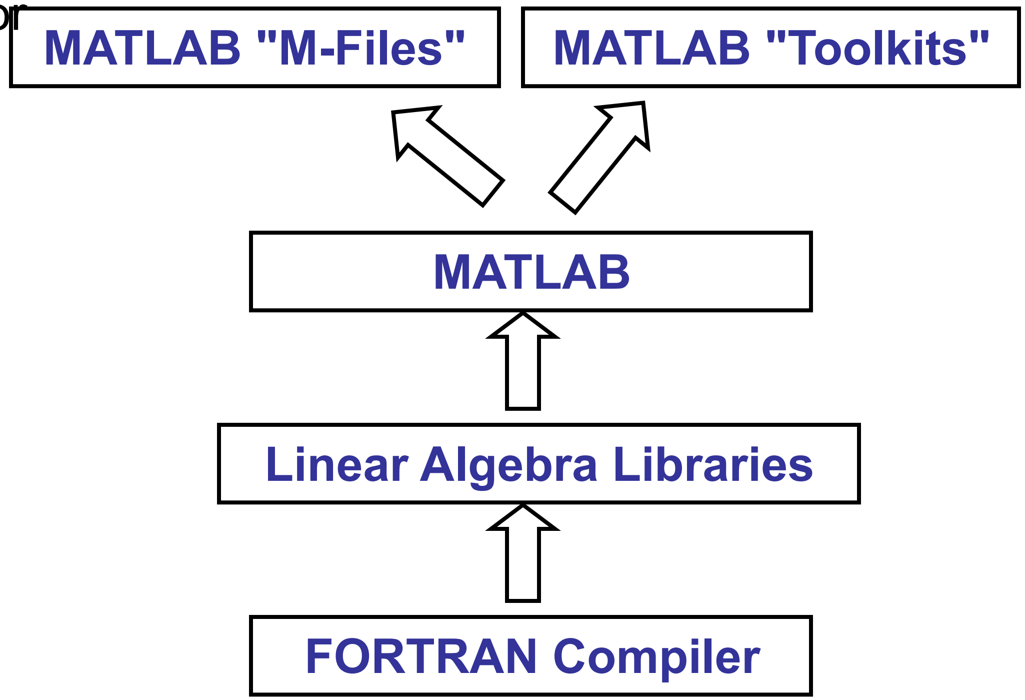
MATLAB illustrates some useful design concepts for software.

Extensible using "Toolkits" or user-contributed programs called M-files.

Interactive user interface; hides boring details

Modular, reusable software components

Standard base platform



MATLAB Toolbox

"Toolboxes" providing functions for many applications:

- Symbolic Math Toolbox: mathematical manipulation of symbols
- Partial Differential Equation Toolbox: tools for solving PDEs in 2-D
- Statistics Toolbox: statistical data analysis
- Image processing toolbox: visualization and image analysis
- Bioinformatics toolbox: computational molecular biology
- Compiler: application development
- Many many more.

Running MATLAB

- MATLAB is installed on the classroom machines University license can be purchase at http://its.fsu.edu/Software/SoftwareLicensing/Mathworks-MATLAB

  We will spend about three weeks to introduce MATLAB.
- You will master MATLAB by practicing (homework and projects) and self-learning (let MATLAB help you)
- Useful tutorial material:
  - Links http://www.mathworks.com/academia/student_center/tutorials/launchpad.html
  - Many other online resources

https://octave-online.net

- Online octave to experiment with Matlab commands

MATLAB Desktop

- Command window
  - ✓ Type the commands (2+2;x=3;sin(100))
  - ✓ MATLAB is case sensitive
  - ✓ Virtually all numerical computations in MATLAB are performed by typing commands, not by manipulating menus.
- Current Directory Browser and Workspace Browser There are tabs for alternating between the two browsers
  - ✓ Workspace: the complete collection of defined variables
  - ✓ Clear/Save/Load workspace by typing
- Command History Window
  - ✓ Save time of typing commands
  - ✓ Right click a command to view all options

matlab onramp has a tutorial and video instructions, ideally we could acces this though the site license — but I need a student ID to check

Exercise

Step 1:

```
>>fun=sin(pi/4)
```

Step 2:

```
>>format long
```

Step 3:

```
>>fun
```

What do you observe?

Can you explain?

Let MATLAB Help YOU!

- MATLAB has extensive online help.
- Quick launch by typing "doc" or clicking the (?) symbol on the menu bar
- Type "help" in the command prompt

  A long list of topics for which help is available
- Type "help plot" and "doc plot"

  Help/document file for the "plot" command
- Type "lookfor plot"

  Search the first line of every MATLAB help files for a specified string (plot here)
- "More on" and "more off" for better display

  Tell me what these two commands are for

Let OCTAVE Help YOU!

- Octave has extensive online help.
- Quick launch by typing "doc" or clicking the (?) symbol on the menu bar
- Type "help" in the command prompt

  A long list of topics for which help is available
- Type "help plot" and "doc plot"

  Help/document file for the "plot" command
- Type "lookfor plot"

  Search the first line of every MATLAB help files for a specified string (plot here)
- "More on" and "more off" for better display

  Tell me what these two commands are for

Why MATLAB?

- tons of basic "libraries" or functions available
- many more complicated "toolboxes" can be added
- Interpreted and interactive, no compiling

```
>>a=3
>>b=2
>>c=a+b
```

- In a sequence of commands, the intermediate values may not be interesting or the echoing of values to the command window might be distraction.
- The output of individual command may be suppressed by appending a semicolon to the end of the expression:

```
>>a=3;
>>b=2;
>>c=a+b
```

or a=3; b=2; c=a+b

Why MATLAB?

- Easy debugging and errors are easier to fix
  >>c=a+b+d
- Any variable appearing at the right-hand side of the equals sign must already be defined.
- A variable is created whenever it appears on the left-hand side of the equal sign.
- The value stored in a variable can be changed by a subsequent assignment
  >> a=5; a=a+2
- Watch the value of “a” in the workspace.
- The names of variables can be up to 31 characters long.

Why MATLAB?

- Everything is a matrix

Square brackets,[ ], are used to delimit vector and matrices.

>>a=[1 2 3 4 5] (try a=(1 2 3 4 5))

or a=1:5, or a=[1:5], or a=[1:1:5] (colon notation)

>>a(3) (try a[3])

>>length(a)

>>b='some string'

>>b(3)

>>length(b)

- MATLAB code is optimized to be relatively quick and easy when performing matrix operations

>>b=3*a

>>c=a'*a

Vector and Matrix: Basics

- A *vector* is an ordered list of numbers
  X=[2 4 6 8]
  X(3)=?
- A *matrix* is a rectangular array of members
  A=[1 2 3 4;
  5 6 7 8;
  9 10 11 12]
  A(2,3)=?

Why MATLAB?

- In-house graphics capabilities for visualization

```
>>plot(a,b)
>> xlabel('Icecream consumption (g)')
>>ylabel('Weight (Pounds)')
>>title('Relationship between Y and X')
>>figure(2)
```

Why MATLAB?

- Easy to learn and fast development times
- tri-development: interactive, scripts, and functions
- Exercise

show $\sum_{i=1}^{n} 1/i^2 \rightarrow \frac{\pi^2}{6}$ as $n \rightarrow \infty$

Consider n=100
Demonstration for each n can be done in several lines.

Why MATLAB?

```
>>exact=(pi^2)/6;
>>n=100;
>>i=[1:n];
>>i=i.^2;
>>i=1./i;
>>approx=sum(i);
>>exact-approx
```

I want to try n=1000, but I do not want to type all these lines. What should I do?

MATLAB Programming

- There are two different kinds of MATLAB programs: script and functions.
- There are stored in plain text files that end with the extension ".m", called m-files.
- You can create and edit an m-file within MATLAB by typing "edit *filename*" in the command window or outside MATLAB using any text editor.

Generate an m-file from the Command History Window

- You can highlight commands in the Command History window, right click, and choose Create m-file.
- As with other applications, use Shift-click to add items to the selection and Ctrl-click to remove items from the selection.
- Save the file as “Exercise_1.m” or any file of your preference.
- Run the script m-file in the command window by typing >>Exercise_1
- Change n from 100 to 1000 and then 10000, and run the program. What do you observe?

More on Script m-files and Programming Style

- Scripts are just sequences of interactive statements stored in a file.
- Typing the name of the script at the command prompt has the same effect as typing the contents of the script file at the command prompt.
- Script files have no input and output parameters; hence they are most useful for those tasks that never change.
- Do you like or are you content about the m-file?
- A programming style consists of
  - Visual appearance of the code
  - Conventions used for variable names
  - Documentation with comment statements

Why MATLAB?

- The Good:
- The Bad:
  - small coding mistakes can result in slow code
  - loops are extremely computationally intensive
  - language is limited: no templates, classes etc.
  - as an interpreted language, MATLAB is slower than a compiled language such as C++
- The Ugly:
  - proprietary (but the language format is open)
  - expensive
  - the open source substitute, GNU Octave, is not fully compatible (http://www.gnu.org/software/octave/index.html)

Basics of Linear Algebra

Instructor: Dr. Ming Ye

Vectorize!

- The single most effective aspect in Matlab in order to build efficient code.

  Run test functions test1 and test 2.
- Vectorize whenever you can.
- Avoid loop whenever you can.
- All built-in functions in MATLAB are vectorized, meaning that, if given a vector as inputs, the operation denoted by the name of the function is applied to all elements of the vector.
- The array operations are VERY important. Be careful about location of the period.

Vector Operations

- Addition and Subtraction
- Multiplication by a scalar
- Transpose
- Linear Combinations of Vectors
- Inner Product
- Outer Product

Define Vectors in MATLAB

- Assign any expression that evaluates to a vector

```
>> v = [1 3 5 7]
>> w = [2; 4; 6; 8]
>> x = linspace(0,10,5);
>> y = 0:30:180
>> z = sin(y*pi/180);
```

- Distinquish between row and column vectors

```
>> r = [1 2 3];        % row vector
>> s = [1 2 3]';       % column vector
>> r - s
??? Error using ==> -
Matrix dimensions must agree.
```

Although `r` and `s` have the same elements, they are not the same vector. Furthermore, operations involving `r` and `s` are bound by the rules of linear algebra.

Vector Addition and Subtraction

Addition and subtraction are element-by-element operations

$$c = a + b \iff c_i = a_i + b_i \quad i = 1, \ldots, n$$
$$d = a - b \iff d_i = a_i - b_i \quad i = 1, \ldots, n$$

Example:

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

$$a + b = \begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix} \qquad a - b = \begin{bmatrix} -2 \\ 0 \\ 2 \end{bmatrix}$$

Multiplication by a Scalar

Multiplication by a scalar involves multiplying each element in the vector by the scalar:

$$b = \sigma a \quad \Longleftrightarrow \quad b_i = \sigma a_i \quad i = 1, \ldots, n$$

Example:

$$a = \begin{bmatrix} 4 \\ 6 \\ 8 \end{bmatrix} \qquad b = \frac{a}{2} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$

Vector Transpose

The *transpose* of a row vector is a column vector:

$$u = [1, 2, 3] \quad \text{then} \quad u^T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Likewise if v is the column vector

$$v = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \quad \text{then} \quad v^T = [4, 5, 6]$$

Linear Combinations (1)

Combine scalar multiplication with addition

$$\alpha \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} + \beta \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} = \begin{bmatrix} \alpha u_1 + \beta v_1 \\ \alpha u_2 + \beta v_2 \\ \vdots \\ \alpha u_m + \beta v_m \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

Example:

$$r = \begin{bmatrix} -2 \\ 1 \\ 3 \end{bmatrix} \qquad s = \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}$$

$$t = 2r + 3s = \begin{bmatrix} -4 \\ 2 \\ 6 \end{bmatrix} + \begin{bmatrix} 3 \\ 0 \\ 9 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 15 \end{bmatrix}$$

Vector Inner Product (1)

In physics, analytical geometry, and engineering, the **dot product** has a geometric interpretation

$$\sigma = x \cdot y \quad \Longleftrightarrow \quad \sigma = \sum_{i=1}^{n} x_i y_i$$

$$x \cdot y = \|x\|_2 \, \|y\|_2 \cos \theta$$

Vector Inner Product (2)

The rules of linear algebra impose compatibility requirements on the inner product.

The inner product of x and y *requires* that x be a row vector y be a column vector

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = x_1y_1 + x_2y_2 + x_3y_3 + x_4y_4$$

Vector Outer Product

The inner product results in a scalar.
The *outer product* creates a rank-one matrix:

$$A = uv^T \iff a_{i,j} = u_i v_j$$

Example: *Outer product of two 4-element column vectors*

$$uv^T \quad \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 & v_4 \end{bmatrix}$$

$$= \begin{bmatrix} u_1v_1 & u_1v_2 & u_1v_3 & u_1v_4 \\ u_2v_1 & u_2v_2 & u_2v_3 & u_2v_4 \\ u_3v_1 & u_3v_2 & u_3v_3 & u_3v_4 \\ u_4v_1 & u_4v_2 & u_4v_3 & u_4v_4 \end{bmatrix}$$

Matrices

- Columns and Rows of a Matrix are Vectors
- Addition and Subtraction
- Multiplication by a scalar
- Transpose
- Matrix–Vector Product
- Matrix–Matrix Product

Notation

The matrix A with m rows and n columns looks like:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ \vdots & & & \vdots \\ a_{m1} & & \cdots & a_{mn} \end{bmatrix}$$

a_{ij} = element in **row** i, and **column** j

In MATLAB we can define a matrix with

```
>> A = [ ... ; ... ; ... ]
```

where semicolons separate lists of row elements.
The $a_{2,3}$ element of the MATLAB matrix `A` is `A(2,3)`.

Matrices Consist of Row and Column Vectors

As a collection of column vectors

$$A = \left[\begin{array}{c|c|c|c} a_{(1)} & a_{(2)} & \cdots & a_{(n)} \end{array}\right]$$

As a collection of row vectors

$$A = \left[\begin{array}{c} a'_{(1)} \\ \hline a'_{(2)} \\ \vdots \\ \hline a'_{(m)} \end{array}\right]$$

A prime is used to designate a row vector on this and the following pages.

Matrix Operations

Addition and subtraction

$$C = A + B$$

or

$$c_{i,j} = a_{i,j} + b_{i,j} \quad i = 1, \ldots, m; \quad j = 1, \ldots, n$$

Multiplication by a Scalar

$$B = \alpha A$$

or

$$b_{i,j} = \alpha a_{i,j} \quad i = 1, \ldots, m; \quad j = 1, \ldots, n$$

Note

Commas in subscripts are necessary when the subscripts are assigned numerical values. For example, $a_{2,3}$ is the row 2, column 3 element of matrix A, whereas a_{23} is the 23rd element of vector a. When variables appear in indices, such as a_{ij} or $a_{i,j}$, the comma is optional

Matrix Transpose

$$B = A^T$$

or

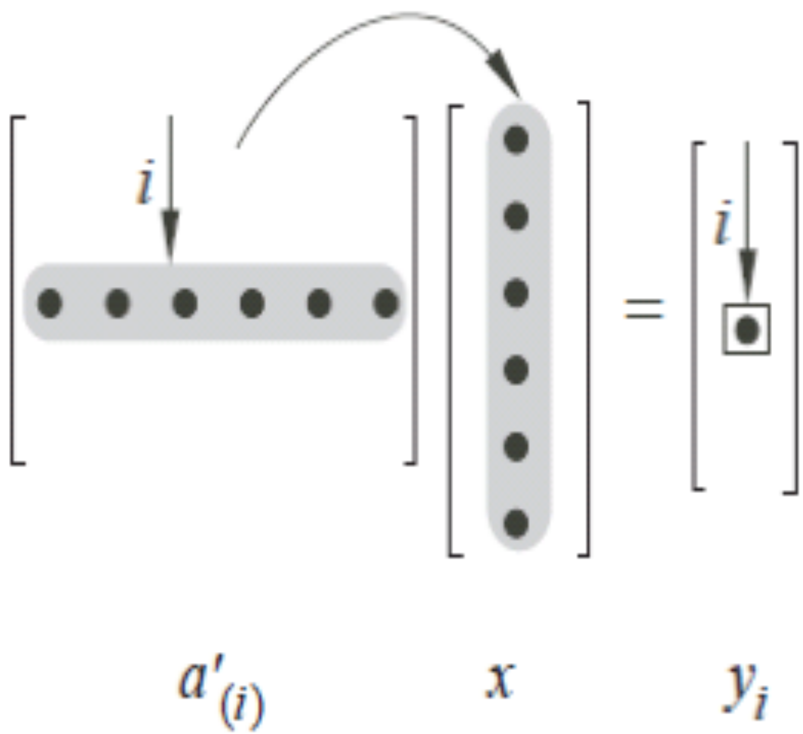
$$b_{i,j} = a_{j,i} \quad i = 1, \ldots, m; \;\; j = 1, \ldots, n$$

In MATLAB

```
>> A = [0 0 0; 0 0 0; 1 2 3; 0 0 0]
A =
 0      0      0
 0      0      0
 1      2      3
 0      0      0

>> B = A'
B =
 0      0      1      0
 0      0      2      0
 0      0      3      0
```

Row View of Matrix-Vector Product



Matrix–vector product

$$\underset{m \times n}{\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}} \underset{n \times 1}{\begin{bmatrix} \\ \\ \end{bmatrix}} = \underset{m \times 1}{\begin{bmatrix} \\ \\ \\ \end{bmatrix}}$$

Vector–Matrix product

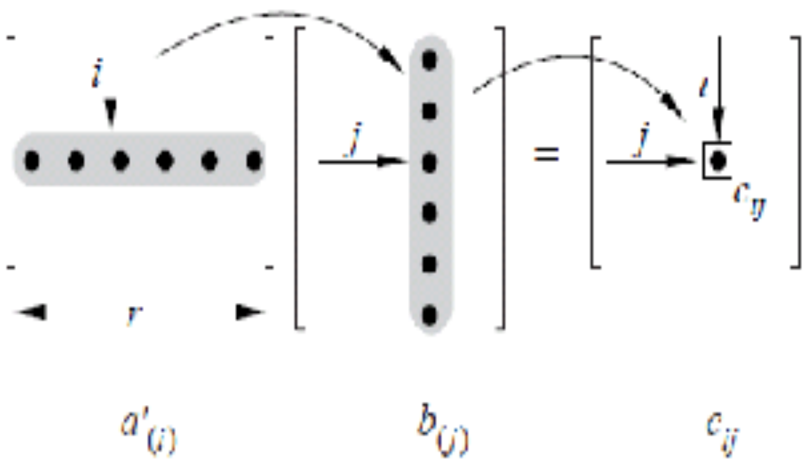
$$\underset{1 \times m}{\begin{bmatrix} & & \end{bmatrix}} \underset{m \times n}{\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}} = \underset{1 \times n}{\begin{bmatrix} & & \end{bmatrix}}$$

Row View of Matrix-Matrix Product

The product AB produces a matrix C. The c_{ij} element is the *inner product* of row i of A and column j of B.

$$AB = C \qquad \Longleftrightarrow \qquad c_{ij} = a'_{(i)} b_{(j)}$$

$a'_{(i)}$ is a row vector, $b_{(j)}$ is a column vector.



The inner product view of the matrix–matrix product is easier to use for hand calculations.

$$\begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{bmatrix} \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix} = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix}$$

Compatibility Requirement

$$\begin{array}{cccc} A & B & = & C \\ [m \times r] & [r \times n] & = & [m \times n] \end{array}$$

Inner dimensions must agree

Also, in general

$$AB \neq BA$$

Exercise

- Write a MATLAB statement to manually enter matrix A

  1 2 3

  4 5 6

  7 8 9

- Obtain the matrix B

  7 8 9

  4 5 6

  1 2 3

- Do not enter matrix B manually but use “lookfor flip” to find the MATLAB function for this operation.

Exercise

- Manually compute C=AB for

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 3 & -1 \\ -2 & 1 \end{bmatrix}$$

and check your result using MATLAB

Exercise

- Create the following vector using MATLAB function *ones*
  [2 2 2 2]
- Create the following diagonal matrix using MATLAB functions *ones* and *diag* or *eye*

$$\begin{matrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{matrix}$$

- Create the following diagonal matrix using MATLAB functions *ones* and *diag*

$$\begin{matrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{matrix}$$