Introduction to Matlab

Andrew T. Schneider

Temple University

August 26, 2015
Outline

- Matlab Introduction
- Data Types
- Matrices
- Operators
- Examples
- Appendix: Get your own version of Matlab
MAT-LAB - Introduction

Matlab is a programming language and environment primarily used for numerical computing.
MAT-LAB - Introduction

- Matlab is a programming language and environment primarily used for numerical computing.
- Matlab was originally developed to facilitate linear algebra calculations, that is matrix operations.
MAT-LAB - Introduction

- Matlab is a programming language and environment primarily used for numerical computing
- Matlab was originally developed to facilitate linear algebra calculations, that is matrix operations.
- Hence the derivation of its name: Matrix laboratory
We use the term Matlab to refer to:
We use the term Matlab to refer to:

1. Matlab the programming environment
We use the term Matlab to refer to:

1. Matlab the programming environment
2. Matlab the programming language
We use the term Matlab to refer to:

1. Matlab the programming environment
2. Matlab the programming language

Broadly speaking, we use Matlab\(^1\) to run code written in Matlab\(^2\)
Matlab v. Matlab - Introduction

- We use the term Matlab to refer to:
  1. Matlab the programming environment
  2. Matlab the programming language
- Broadly speaking, we use Matlab$^1$ to run code written in Matlab$^2$
- It should be clear from context which one we mean
Matlab v. Matlab - Introduction

- We use the term Matlab to refer to:
  1. Matlab the programming environment
  2. Matlab the programming language
- Broadly speaking, we use Matlab\(^1\) to run code written in Matlab\(^2\)
- It should be clear from context which one we mean
- e.g., *double click on the Matlab icon* ...
Data Types

Matlab has the following datatypes:

- strings,
  e.g., hello world
Data Types

Matlab has the following datatypes:

- strings, e.g., hello world
- scalars (or numbers), e.g., 5
Matlab has the following datatypes:

- **strings,**
  e.g., *hello world*
- **scalars** (or numbers),
  e.g., *5*
- **vectors,**
  e.g., [*11 12 13 14*]
Data Types

Matlab has the following datatypes:

- **strings**, e.g., hello world
- **scalars** (or numbers), e.g., 5
- **vectors**, e.g., [11 12 13 14]
- **matrices**, e.g.,
  \[
  \begin{bmatrix}
  5 & 6 & 7 \\
  11 & 12 & 13 \\
  1 & 4 & 2
  \end{bmatrix}
  \]
Matlab has the following datatypes:

- strings, e.g., hello world
- scalars (or numbers), e.g., 5
- vectors, e.g., \([11 \ 12 \ 13 \ 14]\)
- matrices,
  \[
  \begin{bmatrix}
  5 & 6 & 7 \\
  11 & 12 & 13 \\
  1 & 4 & 2
  \end{bmatrix}
  \]

  In fact, vectors can be treated as matrices with only 1 row or 1 column
Data Types

Matlab has the following datatypes:

- strings,
  e.g., hello world
- scalars (or numbers),
  e.g., 5
- vectors,
  e.g., [11 12 13 14]
- matrices,
  e.g., \[
  \begin{bmatrix}
  5 & 6 & 7 \\
  11 & 12 & 13 \\
  1 & 4 & 2 \\
  \end{bmatrix}
  \]
- In fact, vectors can be treated as matrices with only 1 row or 1 column
- Unlike other languages you may have used (e.g., Java, C), these types are *implicitly* assigned
Matrices (the plural form of 'matrix') can be written explicitly using the following syntax:

\[
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{bmatrix}
\]
Matrices (the plural form of 'matrix') can be written explicitly using the following syntax:

- \[ [1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 9] \]
- which yields:

\[
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{bmatrix}
\]
Matrices (the plural form of 'matrix') can be written explicitly using the following syntax:

- \[ [1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 9] \]
- which yields:  
  \[
  \begin{bmatrix}
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 
  \end{bmatrix}
  \]
- Note the semicolons start a new line
Operators

Matlab has many of the typical operators you are familiar with:

- Assignment: =
  e.g., \( a = [1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 9] \) (numerical assignment)
  e.g., \( a = 'hello \ world' \) (string assignment)
Matlab has many of the typical operators you are familiar with:

- **Assignment**: =
  
  e.g., \(a = [1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 9]\) (*numerical assignment*)
  
  e.g., \(a = 'hello world'\) (*string assignment*)

- (Matrix) addition, multiplication, etc.: +, *, ...
  
  e.g., \(5 * 3\) (*yields 15*)
  
  e.g., \([1 \ 2 \ 3] + [4 \ 5 \ 6]\) (*yields [5 7 9]*)
  
  e.g., \([3 \ 2 \ 1; \ 4 \ 5 \ 6] * [1 \ 2; \ 1 \ 2; \ 1 \ 2]\)
  
  (*yields \[
  \begin{bmatrix}
  6 & 12 \\
  15 & 30
  \end{bmatrix}
  \]*)
Component-wise Operations

Component-wise operations enable us to perform operations on the individual components of vectors (matrices). Written as ’dot’ + operator.
E.g.:
Component-wise Operations

Component-wise operations enable us to perform operations on the individual components of vectors (matrices). Written as 'dot' + operator.

E.g.:

\[
\begin{bmatrix} a & b & c \end{bmatrix} \cdot \begin{bmatrix} e & f & g \end{bmatrix} = \begin{bmatrix} a*e & b*f & c*g \end{bmatrix}
\]
Component-wise Operations

Component-wise operations enable us to perform operations on the individual components of vectors (matrices). Written as 'dot’ + operator.

E.g.:

- \([a \ b \ c] \ .* \ [e \ f \ g] = [a*e \ b*f \ c*g]\)
- \([a \ b \ c] \ ./ \ [e \ f \ g] = [a/e \ b/f \ c/g]\)
The colon operator allows to create a range of values

- **Syntax**: \( a : b \); Gives a range of values from \( a \) to \( b \) (inclusive)

- **Syntax**: \( a : z : b \); Gives a range of values from \( a \) to \( b \) by increments of \( z \)

  - \( 1:4 \) \( = \) \( 1 \ 2 \ 3 \ 4 \)
  - \( 1:2:10 \) \( = \) \( 1 \ 3 \ 5 \ 7 \ 9 \)
  - \( 20:-5:0 \) \( = \) \( 20 \ 15 \ 10 \ 5 \ 0 \)
Colon Operator

The colon operator allows to create a range of values

- Syntax: \( a : b; \) Gives a range of values from \( a \) to \( b \) (inclusive)
- e.g., \( 1:4 \) \(~= 1 \ 2 \ 3 \ 4\)
Colon Operator

The colon operator allows to create a range of values

- Syntax: \( a : b \); Gives a range of values from \( a \) to \( b \) (inclusive)
- e.g., \( 1:4 \) \((= 1 \ 2 \ 3 \ 4)\)
- Syntax: \( a : z : b \); Gives a range of values from \( a \) to \( b \) by increments of \( z \)
The colon operator allows to create a range of values

- **Syntax:** \( a : b; \) Gives a range of values from \( a \) to \( b \) (inclusive)
  - e.g., \( 1 : 4 = 1 \ 2 \ 3 \ 4 \)

- **Syntax:** \( a : z : b; \) Gives a range of values from \( a \) to \( b \) by increments of \( z \)
  - e.g., \( 1 : 2 : 10 = 1 \ 3 \ 5 \ 7 \ 9 \)
Colon Operator

The colon operator allows to create a range of values

- **Syntax:** `a : b;` Gives a range of values from `a` to `b` (inclusive)
  
  - e.g., `1:4` (\(= 1 \ 2 \ 3 \ 4\))

- **Syntax:** `a : z : b;` Gives a range of values from `a` to `b` by increments of `z`

  - e.g., `1:2:10` (\(= 1 \ 3 \ 5 \ 7 \ 9\))

  - e.g., `20:-5:0` (\(= 20 \ 15 \ 10 \ 5 \ 0\))
The colon operator can be used to create a vector of numbers in a certain range.
Colon Operator cont’d

- The colon operator can be used to create a vector of numbers in a certain range.
- e.g., \([1:10]\) \(= [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10]\)
The transpose operation (denoted by a superscript $T$) flips the matrix along the diagonal.
**Transpose Operator ’**

- The transpose operation (denoted by a superscript $T$) flips the matrix along the diagonal.
- e.g., $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$
The transpose operation (denoted by a superscript $T$) flips the matrix along the diagonal.

- e.g., $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$

The transpose operation is written in Matlab as `.'.
Transpose Operator ‘

- The transpose operation (denoted by a superscript $T$) flips the matrix along the diagonal
- e.g., $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$
- The transpose operation is written in Matlab as ‘
- e.g., $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}’ = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$
The transpose operation (denoted by a superscript $T$) flips the matrix along the diagonal.

- e.g., $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$

The transpose operation is written in Matlab as '.

- e.g., $[1 \ 2 \ 3; \ 4 \ 5 \ 6]' = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$

The transpose operation is handy for turning row vectors into column vectors.
Transpose Operator ‘

- The transpose operation (denoted by a superscript $T$) flips the matrix along the diagonal.

- e.g., $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$

- The transpose operation is written in Matlab as ‘.

- e.g., $\begin{bmatrix} 1 & 2 & 3; \\ 4 & 5 & 6 \end{bmatrix}' = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$

- The transpose operation is handy for turning row vectors into column vectors.

- e.g., $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}' = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$
Summation

- We can sum over the elements of a vector with the aptly named `sum()` function
Summation

- We can sum over the elements of a vector with the aptly named `sum()` function
- e.g., `sum([1 2 3 4 5]) (= 15)`
Accessing Elements of a Matrix

- \( A(i,j) \) gives us the element of \( A \) in the \( ith \) row, \( jth \) column
Accessing Elements of a Matrix

1. $A(i,j)$ gives us the element of $A$ in the $ith$ row, $jth$ column
2. e.g., for $A = \begin{bmatrix} 11 & 12 & 13 \\ 14 & 15 & 16 \\ 17 & 18 & 19 \end{bmatrix}$; $A(2,3)$ yields 16
Accessing Elements of a Matrix

- $A(i,j)$ gives us the element of $A$ in the $ith$ row, $jth$ column
- e.g., for $A = \begin{bmatrix} 11 & 12 & 13; & 14 & 15 & 16; & 17 & 18 & 19 \end{bmatrix}$; $A(2,3)$ yields 16
- $A(i,:)$ gives all the elements of the $ith$ row of $A$
Accessing Elements of a Matrix

- \( A(i,j) \) gives us the element of \( A \) in the \( ith \) row, \( jth \) column
- e.g., for \( A = [11 \ 12 \ 13; \ 14 \ 15 \ 16; \ 17 \ 18 \ 19]; \)
  \( A(2,3) \) yields \( 16 \)
- \( A(i,:) \) gives all the elements of the \( ith \) row of \( A \)
- e.g., \( A(2,:) \) yields \([14 \ 15 \ 16]\)
Accessing Elements of a Matrix

- $A(i,j)$ gives us the element of $A$ in the $ith$ row, $jth$ column
- e.g., for $A = \begin{bmatrix} 11 & 12 & 13; 14 & 15 & 16; 17 & 18 & 19 \end{bmatrix}$; $A(2,3)$ yields 16
- $A(i,:)$ gives all the elements of the $ith$ row of $A$
- e.g., $A(2,:)$ yields $[14 \ 15 \ 16]$
- for vectors (one dimensional matrices) we need only 1 index
Accessing Elements of a Matrix

- \( A(i,j) \) gives us the element of \( A \) in the \( ith \) row, \( jth \) column
- e.g., for \( A = \begin{bmatrix} 11 & 12 & 13; & 14 & 15 & 16; & 17 & 18 & 19 \end{bmatrix} \); \( A(2,3) \) yields 16
- \( A(i,:) \) gives all the elements of the \( ith \) row of \( A \)
- e.g., \( A(2,:) \) yields \([14 15 16]\)
- for vectors (one dimensional matrices) we need only 1 index
- e.g., \( [1 2 3 4 5]' \) (2) yields 2
Examples

Let \( A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 6 & 7 & 8 \\ 10 & 11 & 12 & 16 & 15 \\ 5 & 4 & 3 & 2 & 1 \\ 8 & 9 & 1 & 2 & 3 \end{bmatrix} \)

Make a submatrix of \( A \) consisting of rows 2 and 3 and columns 1-4

- \( A(2:3,1:4) \)
- Multiply the 2nd row by the 2nd column componentwise
  \( A(:,2)' .* A(2,:) \)
- Add \( A \) to \( A^T \)
  \( A + A' \)
Examples

Let \( A = \begin{bmatrix}
1 & 2 & 3 & 4 & 5 \\
4 & 5 & 6 & 7 & 8 \\
10 & 11 & 12 & 16 & 15 \\
5 & 4 & 3 & 2 & 1 \\
8 & 9 & 1 & 2 & 3
\end{bmatrix} \)

- Make a submatrix of \( A \) consisting of rows 2 and 3 and columns 1-4
  \( A(2:3,1:4) \)

- \( A(2:3,1:4) \)
Examples

Let \( A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 6 & 7 & 8 \\ 10 & 11 & 12 & 16 & 15 \\ 5 & 4 & 3 & 2 & 1 \\ 8 & 9 & 1 & 2 & 3 \end{bmatrix} \)

- Make a submatrix of \( A \) consisting of rows 2 and 3 and columns 1-4
- \( A(2:3,1:4) \)
- Multiply the 2nd row by the 2nd column componentwise
- \( A(:,2)' \cdot A(2,:) \)
Let \( A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 6 & 7 & 8 \\ 10 & 11 & 12 & 16 & 15 \\ 5 & 4 & 3 & 2 & 1 \\ 8 & 9 & 1 & 2 & 3 \end{bmatrix} \)

- Make a submatrix of \( A \) consisting of rows 2 and 3 and columns 1-4
  \( A(2:3,1:4) \)
- Multiply the 2nd row by the 2nd column componentwise
  \( A(:,2)' \times A(2,:) \)
- Add \( A \) to \( A^T \)
  \( A + A' \)
Examples

Let \( A = \begin{bmatrix}
1 & 2 & 3 & 4 & 5 \\
4 & 5 & 6 & 7 & 8 \\
10 & 11 & 12 & 16 & 15 \\
5 & 4 & 3 & 2 & 1 \\
8 & 9 & 1 & 2 & 3
\end{bmatrix} \)

- Make a submatrix of \( A \) consisting of rows 2 and 3 and columns 1-4
- \( A(2:3,1:4) \)
- Multiply the 2nd row by the 2nd column componentwise
- \( A(:,2)' .* A(2,:) \)
- Add \( A \) to \( A^T \)
Examples

Let \( A = \begin{bmatrix}
1 & 2 & 3 & 4 & 5 \\
4 & 5 & 6 & 7 & 8 \\
10 & 11 & 12 & 16 & 15 \\
5 & 4 & 3 & 2 & 1 \\
8 & 9 & 1 & 2 & 3
\end{bmatrix} \)

- Make a submatrix of \( A \) consisting of rows 2 and 3 and columns 1-4
  \( A(2:3,1:4) \)
- Multiply the 2nd row by the 2nd column componentwise
  \( A(:,2)' \times A(2,:) \)
- Add \( A \) to \( A^T \)
  \( A + A' \)
Get your own copy of Matlab

- Matlab is proprietary licensed software
- Temple students can get access through the university by going to https://download.temple.edu/
- GNU Octave is an open source alternative which aims to be fully compatible with Matlab. For more information go to https://gnu.org/software/octave/