MATLAB for beginners

KiJung Yoon,¹

¹Center for Learning and Memory, University of Texas at Austin, Austin, TX 78712, USA
MATLAB Tutorial II

• Functions for matrix analysis

i) For creating a vector of evenly spaced entries.

\[ u = 1:20 \]
\[ u = \text{linspace}(1,20,20) \]
\[ v = 0:20:100 \]
\[ v = \text{linspace}(0,100,6) \]

ii) For \( \mathbf{v} \in R^{n \times 1} \), \( \| \mathbf{v} \| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2} \)

\[ \text{norm}(v) \]
\[ \text{sqrt} (v' \ast v) \]

iii) For creating \( X \in R^{3 \times 2} \) of all zeros:

\[ X = [0 \ 0; 0 \ 0; 0 \ 0] \]
\[ X = \text{zeros}(3,2) \]

iv) For creating \( X \in R^{3 \times 2} \) of all ones:

\[ X = [1 \ 1; 1 \ 1; 1 \ 1] \]
\[ X = \text{ones}(3,2) \]

v) For creating \( X \in R^{3 \times 2} \) of random values sampled from a uniform distribution of the interval \([0 \ 1]\):

\[ X = \text{rand}(3,2) \]
\[ X = 2 \times \text{rand}(3,2) \ % \text{ random values sampled from a uniform distribution of the interval \([0 \ 2]\)} \]

vi) For creating diagonal matrix \( X \) whose diagonal elements are \([1 \ 2 \ 3]\):

\[ X = [1 \ 0 \ 0; 2 \ 0; 0 \ 0 \ 3] \]
\[ X = \text{diag}([1 \ 2 \ 3]) \]
\[ X = \text{diag}([1;2;3]) \]
vii) For getting diagonal elements from a square matrix $X$:

$$ [X(1,1); X(2,2); X(3,3)]$$

diag(X)

viii) For creating identity matrix $I_{3\times3}$:

$$ I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$ I = \text{eye}(3)$$

ix) Sum of diagonal elements:

$$ X(1,1)+X(2,2)+X(3,3)$$

$\text{trace}(X)$

x) For computing the inverse of a square matrix $X$: *(we will learn the concept of matrix inverse in next class)*

$$ Y = \frac{1}{X(1,1)*X(2,2)-X(1,2)*X(2,1)} \cdot \begin{bmatrix} X(2,2) & -X(1,2) \\ -X(2,1) & X(1,1) \end{bmatrix} \text{ only when } X \text{ is 2 by 2 matrix}$$

$$ Y = \text{inv}(X)$$

xi) For computing the determinant of a square matrix $X$:

$$ d = X(1,1)*X(2,2)-X(1,2)*X(2,1) \text{ only when } X \text{ is 2 by 2 matrix}$$

$$ d = \text{det}(X)$$

3
• Conditional statements

======== Syntax ========

if expression (e.g. \( x > 2 \))
 statements (e.g. \( y = 3 \))

elseif expression (e.g. \( x < 1 \))
 statements (e.g. \( y = 1 \))

else
 statements (e.g. \( y = 2 \))

end

\[
\begin{array}{c}
  y \\
  3 \\
  2 \\
  1 \\
  0 \\
\end{array}
\]

• Exercise: conditional statements

% Assign two row vectors [1 2] and [3 4] to u and v, and write a matlab code that performs
% the addition of the two vectors if a conditional variable 'cond' is equal to 1,
% or that performs the inner (dot) product of them.

u = [1 2];
v = [3 4];

cond = 0; % 'cond' needs to be pre-assigned by a value. You can try many different values.
if cond == 1
    u + v
else
    u*v'
end
• **Loop control statements**

i) **for** statements loop a specific number of times, and keep track of each iteration with an incrementing/decrementing index variable.

\[
\begin{align*}
x(1) &= 0; \\
x(2) &= 1; \\
\text{for } n = 3:10 & \% \text{ the following statement is executed until } n \text{ becomes 10 by incrementing 3 in step of 1} \\
x(n) &= x(n-1)+x(n-2) \\
\text{end}
\end{align*}
\]

ii) **while** statements loop as long as a condition remains true.

\[
\begin{align*}
x(1) &= 0; \\
x(2) &= 1; \\
n &= 3; \\
\text{while } n \leq 10 & \% \text{ the following statements are executed as long as } n \text{ is less than or equal to 10} \\
x(n) &= x(n-1)+x(n-2) \\
n &= n + 1; \\
\text{end}
\end{align*}
\]

The above two examples i) and ii) generate Fibonacci numbers.

• **Loop vs. Vectorization**

MATLAB is optimized for operations involving matrices and vectors. The process of revising loop-based, scalar-oriented code to use MATLAB matrix and vector operations is called vectorization. Vectorizing your code is worthwhile for several reasons: \[doc\ vectorization\]

i) **Appearance:** Vectorized mathematical code appears more like the mathematical expressions found in textbooks, making the code easier to understand.

ii) **Less Error Prone:** Without loops, vectorized code is often shorter. Fewer lines of code mean fewer opportunities to introduce programming errors.

iii) **Performance:** Vectorized code often runs much faster than the corresponding code containing loops.
% Create a vector of one cycle of a sine wave (t from 0 to 2*pi in step of 0.001)
% by using for-loop and vectorized form.

% For loop form
tic; % tic starts a stopwatch timer to measure the internal time at execution of the tic command
n = 1;
for t = 0:0.001:2*pi;
    x(n,1) = sin(t);
    n = n + 1;
end
toc; % toc reads the elapsed time from the stopwatch timer started by the tic function

% Vectorized form
tic;
t = 0:0.001:2*pi;
y = sin(t);
toc;

• Q: Construct 3 by 3 identity matrix in the following four different ways:
  
  i) for loop
  
  ii) while loop
  
  ii) diag and ones
  
  iii) eye

============================================
i) \textit{for} loop

\begin{verbatim}
X = zeros(3,3);
for n = 1:3
    X(n,n) = 1;
end
\end{verbatim}

ii) \textit{while} loop

\begin{verbatim}
X = zeros(3,3);
n = 1;
while n <= 3
    X(n,n) = 1;
    n = n + 1;
end
\end{verbatim}

iii) \textit{diag} and \textit{ones}

\begin{verbatim}
X = diag(ones(3,1));
% OR
X = diag(ones(1,3));
\end{verbatim}

iv) \textit{eye}

\begin{verbatim}
X = eye(3);
\end{verbatim}