

VECTORS AND MATRICES IN MATLAB

Vectors

The most direct way is to put the values that you want in the vector in square brackets, separated by either spaces or commas. Example:

```
>> v = [1, 2, 3, 4]
v =
    1    2    3    4
```

The **colon operator** can be used to **iterate** through the regularly spaced values. Example:

```
>> vec = 1:5
vec =
    1    2    3    4    5
```

With the colon operator, a **step value** can also be specified with another colon, in the form (first: step: last). Example:

```
>> nv = 1:2:9
nv =
    1    3    5    7    9
```

Similarly, the **linspace** function creates a linearly spaced vector; **linspace(x,y,n)** creates a vector with n values in the inclusive range from x to y. For example, the following creates a vector with five values linearly spaced between 3 and 15, including the 3 and 15:

```
>> ls = linspace(3,15,5)
ls =
    3    6    9   12   15
```

Vector variables can also be created using existing variables. For example, a new vector is created here consisting first of all the values from nv followed by all values from ls:

```
>> newvec = [nv ls]
newvec =
    1    3    5    7    9    3    6    9   12   15
```

Putting two vectors together like this to create a new one is called **concatenating** the vectors.

Matrices

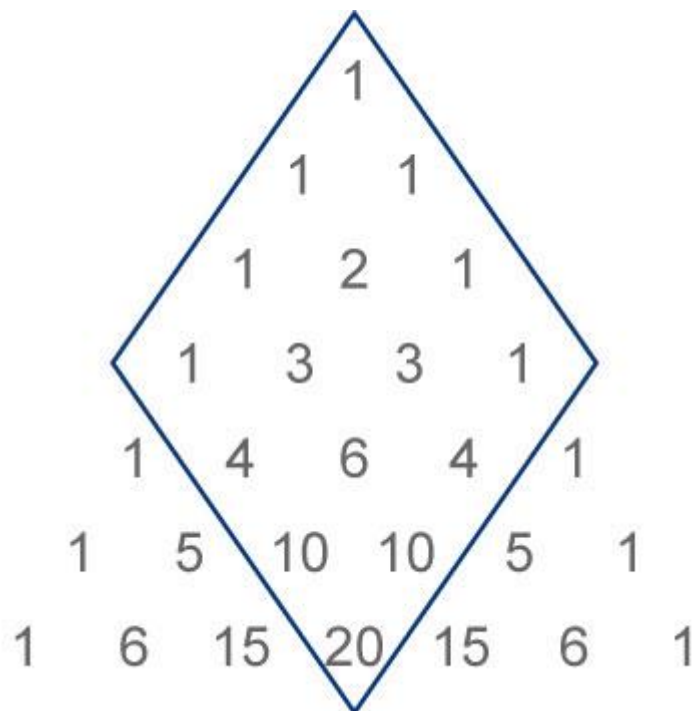
MATLAB has dozens of functions that create different kinds of matrices. Two of them can be used to create a pair of 3-by-3 example matrices for use throughout this lecture. The first example is symmetric.

```
A = pascal(3)
```

```
A =
```

```
1 1 1
1 2 3
1 3 6
```

Pascal's triangle is a triangle formed by rows of numbers. The first row has entry 1. Each succeeding row is formed by adding adjacent entries of the previous row, substituting a 0 where no adjacent entry exists. The pascal function forms Pascal's matrix by selecting the portion of Pascal's triangle that corresponds to the specified matrix dimensions, as outlined in the graphic. The matrix outlined corresponds to the MATLAB® command `pascal(4)`.



The second example is not symmetric.

```
B = magic(3)
```

B =

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

- `M=magic(n)` returns an n -by- n matrix constructed from the integers 1 through n^2 with equal row and column sums. The order n must be a scalar greater than or equal to 3.

Another example is a 3-by-2 rectangular matrix of random integers.

```
C = fix(10*rand(3,2))
```

C =

```
9 4
2 8
6 7
```

- `X = rand` returns a single uniformly distributed random number in the interval (0,1).
- `X = rand(n)` returns an n -by- n matrix of random numbers.
- `Y = fix(X)` rounds each element of X to the nearest integer toward zero. For positive X , the behavior of `fix` is the same as `floor`. For negative X , the behavior of `fix` is the same as `ceil`.

Matrices of random numbers can be created using the **rand** and **randint** functions. The first two arguments to the **randint** function specify the size of the matrix of random integers. For example, the following will create a 2×4 matrix of random integers, each in the range from 10 to 30:

```
>> randint (2,4,[10,30])
```

ans =

```
29 22 28 19
14 20 26 10
```

For the **rand** function, if a single value n is passed to it, an $n \times n$ matrix will be created, or passing two arguments will specify the number of rows and columns:

```
>> rand(2)
```

ans =

```
0.2311 0.4860
0.6068 0.8913
```

```
>> rand(1,3)
```

```
ans =  
    0.7621  0.4565  0.0185
```

```
>> zeros(3)
```

```
ans =  
    0  0  0  
    0  0  0  
    0  0  0
```

```
>> zeros(2,4)
```

```
ans =  
    0  0  0  0  
    0  0  0  0
```

```
>> ones(3)
```

```
ans =  
    1  1  1  
    1  1  1  
    1  1  1
```

```
>> ones(2,4)
```

```
ans =  
    1  1  1  1  
    1  1  1  1
```

Vector Products and Transpose

A *column vector* is an m -by-1 matrix, a *row vector* is a 1-by- n matrix and a *scalar* is a 1-by-1 matrix. The statements

```
u = [3; 1; 4]
```

```
v = [2 0 -1]
```

```
s = 7
```

produce a column vector, a row vector, and a scalar.

```
u =
```

```
    3
```

```
    1
```

```
    4
```

```
v =
```

```
    2    0   -1
```

S =
7

For real matrices, the *transpose* operation interchanges a_{ij} and a_{ji} . MATLAB uses the apostrophe (or single quote) to denote transpose. Our example matrix A is *symmetric*, so A' is equal to A. But B is not symmetric.

B = magic(3)

B =
8 1 6
3 5 7
4 9 2

X = B'

X =
8 3 4
1 5 9
6 7 2

Transposition turns a row vector into a column vector.

x = v'
x =
2
0
-1

If

z = [1+2i 3+4i]

then z' is (*complex conjugate transpose*)

1-2i
3-4i

while z.' is

1+2i (*unconjugated complex transpose*)
3+4i

The Identity Matrix

Identity matrices are matrices of various sizes with ones on the main diagonal and zeros elsewhere.

The function

eye(m,n)

returns an m -by- n rectangular identity matrix and

`eye(n)`

returns an n -by- n square identity matrix.

The Kronecker Tensor Product

The Kronecker product, $\text{kron}(X,Y)$, of two matrices is the larger matrix formed from all possible products of the elements of X with those of Y . If X is m -by- n and Y is p -by- q , then $\text{kron}(X,Y)$ is mp -by- nq . The elements are arranged in the following order:

$$\begin{bmatrix} X(1,1)*Y & X(1,2)*Y & \dots & X(1,n)*Y \\ \vdots & \vdots & \ddots & \vdots \\ X(m,1)*Y & X(m,2)*Y & \dots & X(m,n)*Y \end{bmatrix}$$

$X =$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$I = \text{eye}(2)$

$I =$

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

$\text{kron}(X,I)$ is

$$\begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 3 & 0 & 4 & 0 \\ 0 & 3 & 0 & 4 \end{bmatrix}$$

$\text{kron}(I,X)$ is

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 3 & 4 \end{bmatrix}$$

Solving System of Linear Equations

Suppose the following system of equations:

$$\begin{aligned} x_1 - x_2 + 2x_3 - x_4 &= -8 \\ 2x_1 - 2x_2 + 3x_3 - 3x_4 &= -20 \\ x_1 + x_2 + x_3 &= -2 \\ x_1 - x_2 + 4x_3 + 3x_4 &= 4 \end{aligned}$$

$$A = \begin{bmatrix} 1 & -1 & 2 & -1 \\ 2 & -2 & 3 & -3 \\ 1 & 1 & 1 & 0 \\ 1 & -1 & 4 & 3 \end{bmatrix}$$

$$B = \begin{bmatrix} -8 \\ -20 \\ -2 \\ 4 \end{bmatrix}$$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$AX=B$$

To solve the above system of equations we use the command:

$$X=A \backslash B$$

Or we can use the command:

$$X=\text{inv}(A)*B$$

Where $\text{inv}(A)$ function returns the inverse of the matrix A.

We get the following values for the vector X

X=

-7
3
2
2