Basic Matrix Algebra

MATLAB is an abbreviation for "matrix laboratory." While other programming languages mostly work with numbers one at a time, MATLAB® is designed to operate primarily on whole matrices and arrays.

All MATLAB variables are multidimensional *arrays*, no matter what type of data. A *matrix* is a two-dimensional array often used for linear algebra.

Array Creation

To create an array with four elements in a single row, separate the elements with either a comma (`,`) or a space

```
x = [1 2 3 4]
```

```
x = 1×4
     1     2     3     4
```

This type of array is a *row vector*.

To create the 3-by-3 matrix A=[1 2 -1;2 -1 2;3 2 -1] that has multiple rows, separate the rows with semicolons.

```
A=[1 2 -1;2 -1 2;3 2 -1]
```

```
A = 3×3
     1     2    -1
     2    -1     2
     3     2    -1
```

Another way to create a matrix is to use a function, such as `ones`, `zeros`, or `rand`. For each of these functions, create a 5-by-1 column vector.

```
a = ones(5,1)
```

```
a = 5×1
     1
     1
     1
     1
     1
```

```
b = zeros(5,1)
```

```
b = 5×1
     0
     0
     0
     0
     0
```

```
c = rand(5,1)
```

```
c = 5×1
    0.2858
    0.7572
    0.7537
    0.3804
    0.5678
```

Next try the function eye() to create a 3-by-3 matrix.

```
eye(3,3)
```

```
ans = 3×3
     1     0     0
     0     1     0
     0     0     1
```

Matrix eye() is the unity element in matrix algebra and is known as the identity matrix.

Matrix and Array Operations

Addition: Create another 3-by-3 matrix, B=[1 0 1;0 1 1;1 1 0], and find the sum A + B.

```
A
```

```
A = 3×3
     1     2    -1
     2    -1     2
     3     2    -1
```

```
B = [1 0 1;0 1 1;1 1 0]
```

```
B = 3×3
     1     0     1
     0     1     1
     1     1     0
```

```
A + B
```

```
ans = 3×3
     2     2     0
     2     0     3
     4     3    -1
```

How is the sum calculated?

The sum was calculated by adding each component of one matrix to the same indexed component on the other matrix.

Now create the matrix A + 10*eye.

```
A
```

```
A = 3×3
     1     2    -1
     2    -1     2
     3     2    -1
```

```
A + 10*eye
```

```
ans = 3×3
    11    12     9
    12     9    12
    13    12     9
```

Multiplication: Create the matrix A*B.

```
A*B
```

```
ans = 3×3
     0     1     3
     4     1     1
     2     1     5
```

Using A^2, create the square of A.

```
A^2
```

```
ans = 3×3
     2    -2     4
     6     9    -6
     4     2     2
```

Verify that A^2 = A*A

```
verify = A^2 == A*A
```

```
verify = 3×3 logical array
   1   1   1
   1   1   1
   1   1   1
```

Find the matrix A*eye()

```
A*eye()
```

```
ans = 3×3
     1     2    -1
     2    -1     2
     3     2    -1
```

```
A
```

```
A = 3×3
     1     2    -1
     2    -1     2
     3     2    -1
```

Did this operation change the matrix A?

No. Multiplying a matrix by its idendity matrix will just return the same matrix back.

Matrix inversion:

Use inv(A) to create the matrix A^{-1}

```
inv(A)
```

```
ans = 3×3
   -0.5000         0    0.5000
    1.3333    0.3333   -0.6667
    1.1667    0.6667   -0.8333
```

Find the product inv(A)*A

```
inv(A)*A
```

```
ans = 3×3
    1.0000    0.0000   -0.0000
    0.0000    1.0000    0.0000
         0   -0.0000    1.0000
```

Use the backslash \ to calculate the same quantity as A \ A. This operation is more accurate.

```
A \ A
```

```
ans = 3×3
    1.0000    0.0000         0
         0    1.0000         0
         0   -0.0000    1.0000
```

What is the result equal to?

This result is equal to its identity matrix.

Solving Systems of Linear Equations

These equations arise in many robotics and control applications as $Ax = b$. The unknown vector x can be solved as $x = A^{-1}b$. Create a 3-by-1 vector b and solve for the unknown vector x. Use b=[1; 0; 1].

```
b = [1;0;1]
```

```
b = 3×1
     1
     0
     1
```

```
x = inv(A)*b
```

```
x = 3×1
    0.0000
    0.6667
    0.3333
```

Alternatively, x=A \ b

```
x = A \ b
```

```
x = 3×1
    0.0000
    0.6667
    0.3333
```

Eigenvalues and Eigenvectors are central in the analysis of linear equations. Use eig(A) to find the eigenvalues of A.

```
eig(A)
```

```
ans = 3×1
    2.4495
   -1.0000
   -2.4495
```

Find the eigenvalues of the matrix [0 -1;1 0].

```
eig([0 -1;1 0])
```

```
ans = 2×1 complex
   0.0000 + 1.0000i
   0.0000 - 1.0000i
```

Are these eigenvalues real?

No. They are imaginary.

Prof Kiriakos Kiriakidis (Jan 10, 2024).