Crashcourse Matlab - Part II: Inbuilt Matlab functions

Matlab provides a large number od inbuild functions that help you do things. Here we just highlight some of the very commonly used ones. The best way to learn about functions is the Matlab documentation. The hyperlinks below will take to the documentation of the functions we discuss.

Basics

In general, a function has an input and an output: out = function(in);

For example you can determine the length of the vector as follows

```
a = [-10:5:20];
len_of_a = length(a)
```

```
len_of_a = 7
```

Here length() is the function, the vector "a" the input and the scalar "len_of_a" the out put. Functions can have multiple inputs.

```
A = rand(6,8)
```

```
A = 6×8
    0.8147    0.2785    0.9572    0.7922    0.6787    0.7060    0.6948    0.7655
    0.9058    0.5469    0.4854    0.9595    0.7577    0.0318    0.3171    0.7952
    0.1270    0.9575    0.8003    0.6557    0.7431    0.2769    0.9502    0.1869
    0.9134    0.9649    0.1419    0.0357    0.3922    0.0462    0.0344    0.4898
    0.6324    0.1576    0.4218    0.8491    0.6555    0.0971    0.4387    0.4456
    0.0975    0.9706    0.9157    0.9340    0.1712    0.8235    0.3816    0.6463
```

Here the function rand() produces a random matrix with 6 rows and 9 columns. Functions can also have multiple outputs.

```
[r, c] = size(A)
```

```
r = 6
c = 8
```

Here the function size() tels you that the matrix A has 6 rows and 8 columns. There are also functions that don't require an input

```
today = date()
```

```
today =
'16-Jan-2021'
```

Here the function date() returns a string, i.e., a number of letters and numbers, giving todays date.

In general, inbuilt matlab functions can have variable numbers of inputs and outputs.

Learning more about functions

- Your power in Matlab correlates directly with your knowledge of functions! So try to explore the documentation. At the bottom of every documentation page you are given a list of related functions, browsing those is a good way to explore potentially useful functions.
- The documentation takes some getting used to, but I expect that you look up the dcumentation by yourself. In class, I will only highlight certain features of functions we are using that are important in the context of the class.

If you know the function name:

Typically you can just google "Matlab Name" and you will get the documentation page.

If you know what you want to do:

Try to google something like "Matlab my activity". For example if you you want to integrate a function but don't know how. Google "Matlab numerical integration". The first hit I get is to this to the documentation page of the Matlab function integral, which not surprisingly intrgrates a function.

Basic Mathematical functions

Matlab has a function for everything you need. Typically the names don't need much explanation. For example all trigonometric and transcendental functions are availabe. For a full list see here and scroll down to Mathematics. A few simple examples are

```
sin(2*pi)
```

```
ans = -2.4493e-16
```

```
e = exp(1) % Euler's number
```

```
e = 2.7183
```

```
log(e)
```

```
ans = 1
```

```
sqrt(2)
```

```
ans = 1.4142
```

```
2^(1/2)
```

```
ans = 1.4142
```

Note that you can use "pi" to refer to the trigonometric constant $\pi = 3.14...$ in Matlab. Note, that log() is the natural logarithm in Matlab, often otherwise denoted ln(). After getting familiar with Matlab you can guess the function name in 80% of the cases.

Basic vector functions

Matlab is particularly good with functions for vectors

```
a = round((rand(5,1)-.5)*10)
```

```
a = 5×1
     2
     3
    -2
     2
     2
```

```
length(a)   % Gives the length of the vector
```

```
ans = 5
```

```
max(a)      % Maximum element of the vector
```

```
ans = 3
```

```
min(a)      % Minimum element of the vector
```

```
ans = -2
```

```
abs(a)'     % takes absolute value of all elements of the vector
```

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

```
norm(a)     % magnitude of the vector
```

```
ans = 5
```

```
mean(a)     % Computes the mean of the vector elements
```

```
ans = 1.4000
```

```
std(a)      % Computes the standard deviation of the vector elements
```

```
ans = 1.9494
```

```
sum(a)      % Computes the sum of all vector elements, returns a scalar
```

```
ans = 7
```

```
cumsum(a)'  % Cumulative sum of all vector elements, returns a vector
```

```
ans = 1×5
     2     5     3     5     7
```

Basic Matrix functions

Matlab is even better with matrix operations

```
A = round((rand(5)-.5)*10);
A = A*A'                % SPD A with real integer entries
```

```
A = 5×5
    39    13     7     0    -8
    13    61    -3   -22   -22
     7    -3    47    -6    -7
     0   -22    -6    42   -10
    -8   -22    -7   -10    33
```

```
cond(A)                 % Condition number
```

```
ans = 13.8975
```

```
rank(A)                 % Rank of matrix
```

```
ans = 5
```

```
Ainv = inv(A)           % computes inverse
```

```
Ainv = 5×5
    0.0296   -0.0104   -0.0065   -0.0072   -0.0033
   -0.0104    0.0513    0.0171    0.0406    0.0476
   -0.0065    0.0171    0.0289    0.0182    0.0215
   -0.0072    0.0406    0.0182    0.0589    0.0471
   -0.0033    0.0476    0.0215    0.0471    0.0800
```

```
[L,U] = lu(A)           % LU factorization
```

```
L = 5×5
    1.0000         0         0         0         0
    0.3333    1.0000         0         0         0
    0.1795   -0.0941    1.0000         0         0
         0   -0.3882   -0.1784    1.0000         0
   -0.2051   -0.3412   -0.1632   -0.5879    1.0000
U = 5×5
   39.0000   13.0000    7.0000         0   -8.0000
         0   56.6667   -5.3333  -22.0000  -19.3333
         0         0   45.2416   -8.0706   -7.3837
         0         0         0   32.0191  -18.8231
         0         0         0         0   12.4923
```

```
[V,D] = eig(A)          % Eigenvalue decomposition
```

```
V = 5×5
   -0.0935    0.8600   -0.0771    0.3883   -0.3080
    0.5011   -0.1558   -0.1718   -0.2088   -0.8071
    0.2389   -0.2534    0.6566    0.6594   -0.1131
    0.5246   -0.0782   -0.6195    0.4576    0.3543
    0.6387    0.4071    0.3868   -0.4018    0.3395
D = 5×5
    6.0768         0         0         0         0
         0   30.7957         0         0         0
         0         0   48.5008         0         0
         0         0         0   52.1741         0
         0         0         0         0   84.4525
```

```
null(A)                 % Basis of nullspace of A
```

```
ans =

  5×0 empty double matrix
```

```
[U,S,V] = svd(A)        % Singular value decomposition of A
```

```
U = 5×5
   -0.3080    0.3883   -0.0771    0.8600    0.0935
   -0.8071   -0.2088   -0.1718   -0.1558   -0.5011
   -0.1131    0.6594    0.6566   -0.2534   -0.2389
    0.3543    0.4576   -0.6195   -0.0782   -0.5246
    0.3395   -0.4018    0.3868    0.4071   -0.6387
S = 5×5
   84.4525         0         0         0         0
         0   52.1741         0         0         0
         0         0   48.5008         0         0
         0         0         0   30.7957         0
         0         0         0         0    6.0768
V = 5×5
   -0.3080    0.3883   -0.0771    0.8600    0.0935
   -0.8071   -0.2088   -0.1718   -0.1558   -0.5011
   -0.1131    0.6594    0.6566   -0.2534   -0.2389
    0.3543    0.4576   -0.6195   -0.0782   -0.5246
    0.3395   -0.4018    0.3868    0.4071   -0.6387
```

```
[Q,R] = qr(A)           % QR factorization
```

```
Q = 5×5
   -0.9185    0.3040    0.1888    0.1655   -0.0309
   -0.3062   -0.8312    0.0297   -0.1228    0.4466
   -0.1649    0.1423   -0.9537   -0.0506    0.2013
         0    0.3620    0.1896   -0.7988    0.4415
    0.1884    0.2558    0.1343    0.5630    0.7511
R = 5×5
  -42.4617  -34.2661  -14.5778    5.8406   21.4546
         0  -60.7687    7.3480   30.0791   19.6785
         0         0  -45.6672   11.6906    7.0474
         0         0         0  -36.1727   28.2977
         0         0         0         0    9.3825
```

Matlab's best feature is the backslash for direct solution of linear systems $\mathbf{Ax} = \mathbf{b}$.

```
b = a                   % r.h.s. vector
```

```
b = 5×1
     2
     3
    -2
     2
     2
```

```
x = A\b
```

```
x = 5×1
    0.0200
    0.2753
```

```
    0.0599
    0.2831
    0.3475
```

```
x2 = Ainv*b
```

```
x2 = 5×1
    0.0200
    0.2753
    0.0599
    0.2831
    0.3475
```

Please NEVER solve a linear system by computing the inverse of $\mathbf{A}$. For large systems this is very slow, very inaccurate and requires enormous memory. The latter is because $\mathbf{A}^{-1}$ is generally a full matrix, even if $\mathbf{A}$ only has a few entries. Instead the backslash uses Gaussian elimination to solve the linear system.