Matlab Cheatsheet

Marcel Neidinger
m.neidinger@unibas.ch

Basics

Defining variables and functions

```
% Variables
y = 42;
% Function with one return
function y = SomeFunc(p1,p2)
% function definiton goes here
% Assign return val by assigning
% the variable
    y = 42;
end
% Function with multiple returns
function [A,z] = SomeFunc(p1,p2)
    % Function body
end
```

Defining vectors with steps

```
x = 1:0.5:10;
```

Simple loops with while and for

```
for i=1:10
% loops i from 1 to 10
end
while n > 10
    n = n - 1;
end
```

Writing for loops as vector operations

Instead of looping over a vector and applying some operation to every entry, one can perform vector operations

```
% Normal with for-loops (slow!)
y = 0:0.3:10;
for i=1:size(y) %
    y(i) = y(i) * 2;
end
% Without loop
y = 2.*y;
```

This works for every basic mathematic operation($+, -, *, /, \hat{}$).

Useful commands and functions

`clc`	Clear command window
`clear`	Clear all variables
`ans`	Variable holding last value
`num2str(num)`	Converts *num* to a string

Built in (math) functions

`abs(x)`	absolute value of x
`pi`	3.1415926...
`exp(`x`)`	Exponential value of x
`eps`	Numerical floating-point precission
`sum(`x`)`	Sums all elements in x
`diff(`x`)`	Returns (a vector) with the difference of adjacent elements
`prod(`x`)`	Returns the product of all elements in x
`round(`x`)`	Rounds a number
`log(`x`)/sqrt(`x`)`	Returns the log or sqrt of x
`max(`x`)/min(`x`)`	Returns max or min value of vector x
`linspace(a,b,n)`	Generates a n dimensional vector of linearily spaced points between a and b
`plot(`x`,`y`)`	Plots vector y against vector x
`subplot(`w`,`h`,`pos`)`	Positions next plot at *pos* in $w \times h$ grid

Operations on Matrixes

Declaring matrixes

`j:k`	Generates a row vector $[j, j+1, \ldots, k]$
`j:i:k`	Generates a row vector $[i, j+i, \ldots, k]$
ones(a,b)	Generates a $a \times b$ matrix of ones
zeros(a,b)	Generates a $a \times b$ matrix of zeros
eye(n)	Generates a $n \times n$ identity matrix

Accessing and manipulating

`x = [0 1 1 2 3]`	1×4 column vector
`x = [0; 1; 1; 2; 3]`	4×1 row vector
`x'`	Transposes a vector(or matrix). Use to convert between row and column vectors
`A = [0 1; 1 2]`	2×2 Matrix
`x(2)` or `A(2,2)`	Access specific elements
`x(j:k)`	All entries from j to k. For k one can use *end*. Works with Matrices too
`A(j,:)`	All elements in j-th row
`A(:,j)`	All elements in j-th column
`diag(A,n)`	Diagonal entries of A. Use additional parameter to get n-th diagonal above or below main diagonal
`inv(A)`	Inverse of A
`size(A)`	Returns rows and columns of A
`[vals,vecs] = eig(A)`	Returns all eigenvalues and eigenvectors
`x(x>10) = 42`	Conditional reassignment
`x(x=42)`	All elements matching condition
`find(A > 42)`	All indices of elements matching condition
`[A,B]`	Concats A and B horizontally
`[A;B]`	Concats A and B vertically
`arrayfun(func,x)`	Applies `func` to every element in x

Using `arrayfun` and `anonymous functions`

You have to define `f` as a function handler

```
x=1:5;
f = @(x) 2*x;
y = arrayfun(f, x);
% Using arrayfun with function in f.m
y = arrayfun(@f, x);
```

Examples

Series 5

Calculate an integral based on quadtrapez

```
function int = quadtrapez(f,a,b,h)
  x = a:h:b;
  int = h/2*(f(a)+f(b)+2*sum(f(x(2:end-1))));
end
```

Series 7

Calculate the lu decomposition for a given matrix

```
function [ L,U ] = ludecomp( A,p,q )
    n = size(A,1);
    L = eye(n);
    for k = 1:n-1
     for i = k+1:min(n,k+q)
       L(i,k) = A(i,k) / A(k,k);
       A(i,k) = 0;
       for j = k+1:min(n,k+p)
         A(i,j) = A(i,j) - L(i,k) * A(k,j);
       end
     end
    end
    U = A;
end
```

General example

Solve a system of equations

```
A = [0 1 1; 2 3 5; 8 13 21];
b = [10 12 13];
x = A/b
```