Julia & IJulia Cheat-sheet (for 18.xxx at MIT)

Basics:

`julialang.org` documentation

`github.com/stevengj/julia-mit` installation & tutorial

`ipython notebook --profile-julia` start IJulia browser

shift-return execute input cell in IJulia

Defining/changing variables:

`x = 3` define variable *x* to be 3

`x = [1,2,3]` array/“column”-vector (1,2,3)

`y = [1 2 3]` 1×3 row-vector (1,2,3)

`A = [1 2 3 4; 5 6 7 8; 9 10 11 12]`
—set *A* to 3×4 matrix with rows 1,2,3,4 etc.

`x[2] = 7` change *x* from (1,2,3) to (1,7,3)

`A[2,1] = 0` change $A_{2,1}$ from 5 to 0

`u, v = (15.03, 1.2e-27)` set u=15.03, $v=1.2\times10^{-27}$

`f(x) = 3x` define a function f(x)

`x -> 3x` an “anonymous” function

Constructing a few simple matrices:

`rand(12), rand(12,4)` random length-12 vector or 12×4 matrix with uniform random numbers in [0,1)

`randn(12)` Gaussian random numbers (mean 0, std. dev. 1)

`eye(5)` 5×5 identity matrix *I*

`linspace(1.2,4.7,100)` 100 equally spaced points from 1.2 to 4.7

`diagm(x)` matrix whose diagonal is the entries of *x*

Portions of matrices and vectors:

`x[2:12]` the 2nd to 12th elements of *x*

`x[2:end]` the 2nd to the last elements of *x*

`A[5,1:3]` row vector of 1st 3 elements in 5th row of A

`A[5,:]` row vector of 5th row of *A*

`diag(A)` vector of diagonals of *A*

Arithmetic and functions of numbers:

`3*4, 7+4, 2-6, 8/3` mult., add, sub., divide numbers

`3^7, 3^(8+2im)` compute 3^7 or 3^{8+2i} power

`sqrt(-5+0im)` $\sqrt{-5}$ as a complex number

`exp(12)` e^{12}

`log(3), log10(100)` natural log (ln), base-10 log ($\log_{10}$)

`abs(-5), abs(2+3im)` absolute value $|-5|$ or $|2+3i|$

`sin(5pi/3)` compute $\sin(5\pi/3)$

`besselj(2,6)` compute Bessel function $J_2(6)$

Arithmetic and functions of vectors and matrices:

`x * 3, x + 3` multiply/add every element of *x* by 3

`x + y` element-wise addition of two vectors *x* and *y*

`A*y, A*B` product of matrix *A* and vector *y* or matrix *B*

`x * y` not defined for two vectors!

`x .* y` element-wise product of vectors *x* and *y*

`x .^ 3` every element of *x* is cubed

`cos(x), cos(A)` cosine of every element of *x* or *A*

`exp(A), expm(A)` exp of each element of *A*, matrix exp e^A

`x′, A′` conjugate-transpose of vector or matrix

`x′*y, dot(x,y), sum(conj(x).*y)` three ways to compute $x \cdot y$

`A \ b, inv(A)` return solution to Ax=b, or the matrix A–1

`λ, V = eig(A)` eigenvals λ and eigenvectors (columns of *V*) of *A*

Plotting (type `using PyPlot` first)

`plot(y), plot(x,y)` plot *y* vs. 0,1,2,3,… or versus *x*

`loglog(x,y), semilogx(x,y), semilogy(x,y)` log-scale plots

`title(“A title”), xlabel(“x-axis”), ylabel(“foo”)` set labels

`legend([“curve 1”, “curve 2”], “northwest”)` legend at upper-left

`grid(), axis(“equal”)` add grid lines, use equal *x* and *y* scaling

`title(L”the curve $e^\sqrt{x}$”)` title with LaTeX equation

`savefig(“fig.png”), savefig(“fig.eps”)` save as PNG or EPS image

18.303 Linear Partial Differential Equations: Analysis and Numerics
Fall 2014