

- \* AI languages
  - o LISP (List Processing) – 2nd oldest high-level language (only Fortran)
  - o Prolog ("PRogrammation en LoGique")
  - o Matlab (Matrix laboratory)
- \* Why Matlab?
  - o In general good at:
    - + Math and computation
    - + Data analysis
    - + operations working with arrays (vectors, matrixes, multi-D arrays)
  - \* Arrays don't require dimensioning
    - + Algorithm development
  - \* Interactive environment
    - o Why use it here?
      - + overall, we'll be working with symbolic AI, LISP and Prolog more traditional and suited
      - + The "AI" language most likely to be used today
      - + Still good, dynamic data structures that are useful
  - \* Arrays like lists, since dynamic
    - + Still encourages functional programming
    - + Interactive environment makes testing easy
  - \* Not unusual to prototype in Matlab and then convert to C++
    - + Will make 151 easier if you know it
  - \* Matrices
    - o Fundamental data structure
      - + 0x0 array = [] = empty matrix
  - \* Can also have 0xn, or nx0 or nxmx0
    - + 1x1 arrays = scalars
    - + 1xn, nx1 arrays = vectors
    - o A = [...;...]
    - o Indexing
      - + A(i, j) accesses row i, column j
      - + Starts at (1,1) not (0,0)
      - + A(:)
      - + A(1, :)
      - + A(1, 1:2)
      - + A(1:2, 1:2)
      - + A(1:2:end)
      - + Can delete rows, columns with A(:, 1) = [];
    - o Variables
      - + Dynamic variables, just define

- + Numbers, letters, \_ (start with letter)
- \* Can over ride functions (such as i)
  - + Dynamic resizing
- \*  $A = [1\ 2\ 3]$ ;  $A(4) = 4$ ;
- \* Inefficient
  - o Can combine, append
    - +  $[A\ A]$  or  $[A;A]$
  - o Defaults to rational numbers
  - o operators
    - +  $+$ ,  $-$ ,  $*$ ,  $/$ ,  $,$ ,  $'$ ,  $()$ ,  $.*$ ,  $./$ , etc
  - o size
  - o Sort, sum
    - + Use number to indicate which dim
  - o Initializers
    - + Zeros()
    - + ones()
    - + Rand()
  - o Strings – matrix of chars, often pad and such (cells)
- \* Rather than work with one value at a time, work with array
  - o Example:  $A + 1$ ;
- \* Logic
  - o 0 is false
  - o non-zero (typically 1) is true
  - o functions true and false f\*\*r readability
  - o  $\&$ ,  $|$ ,  $\sim$ , xor
  - o  $||$ ,  $\&\&$  (short circuit forms, useful in control)
  - o  $==$ ,  $\sim=$ ,  $<$ ,  $>$ ,  $\leq$ ,  $\geq$
  - o any, all (for vects)
    - + If 1's and 0's can use sum to get count, or sum of sums
  - o all is\* functions (isempty, isscalar)
- \* Logical subscripting
  - o  $X = [2\ \text{Inf}\ 3]$
  - o  $X = X(\text{isfinite}(X)) \Rightarrow X = [2\ 3]$
  - o  $X(X < 0) = 0$
- \* Find returns indexes (vector form)
- \* Control
  - o If, else, elseif, end
  - o For
    - + For  $n=1:n$
    - + often can vectorize
  - o Return (no arg)
  - o other standard control (break, switch, continue)

- \* Interactive environment

- o With no semi-colon, outputs value
  - + Semi-colon suppresses
- o Up down arrows to get previous, can edit
  - + Also history to left
- o Tab completion
- o If no destination var set, goes to 'ans'
- o Can turn on debugging from menu
- o Editor
- o Profiler
- o Save and load variables
- o Type 'help <fname>' for help on a function
- o Can be run from command line rather than full environment

(matlab -nodisplay)

- \* Scripts

- o Toss in .m file

- \* Functions

- o Generally each in own .m file
  - + Function [a, b] = foo(x)
- o Comments right after (%)
- o Variables local to function (global vars possible, but discouraged)
- o Lots of variants and tricky stuff, mostly unnecessary
- o Can be passed varying number of args
- o Can return varying number of things
  - + Dropped if not assigned

- \* More advanced data structs

- \* Cell arrays

- o C = {a b c}
- o Useful for combining things of different sizes

- \* Structs