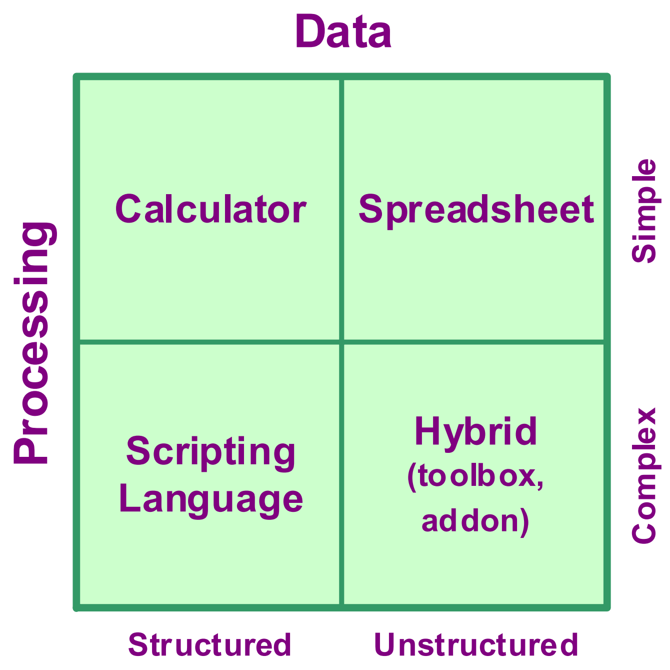
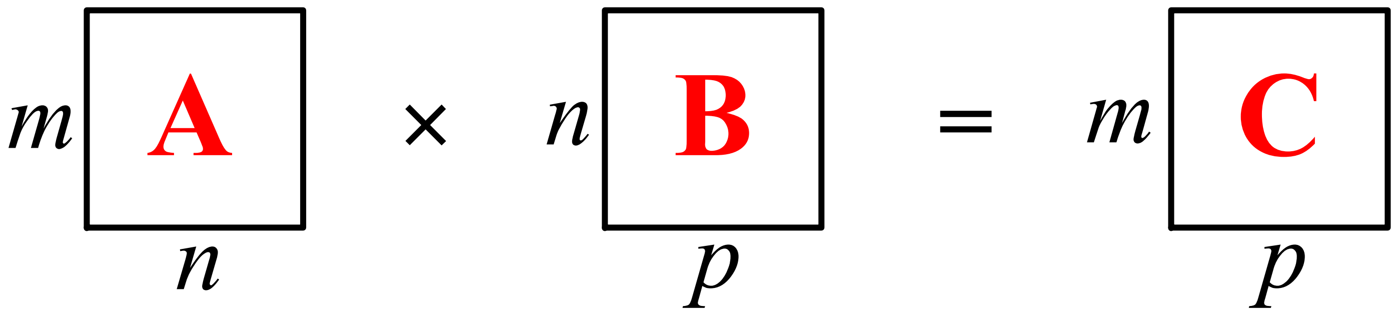
Computational Tools



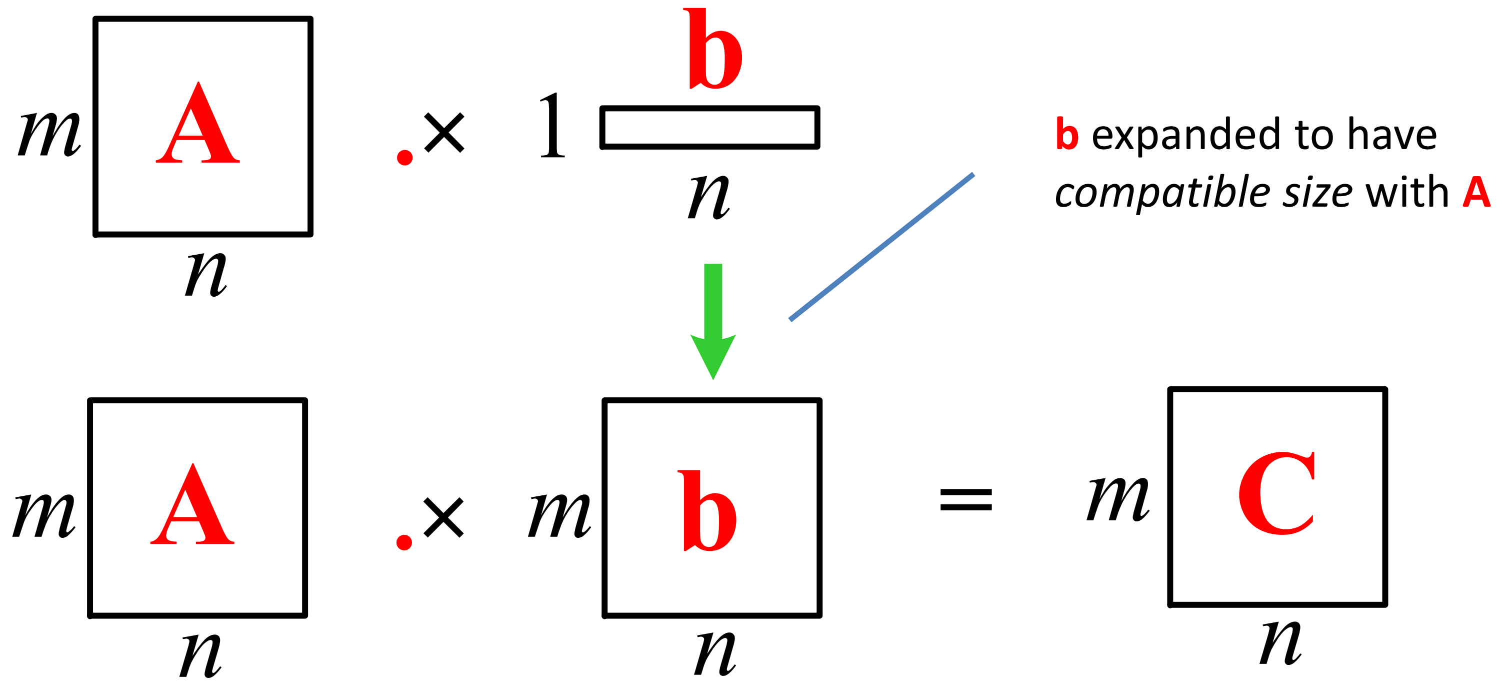
Matrix Multiplication



$$(m\times \cancel{n})\times(\cancel{n}\times p) \quad = \quad (m\times p)$$

Arrays must have same
inner dimensions

Element-by-Element Multiplication



$$\left(m\times n\right)\times\left(1\times n\right) \quad = \quad \left(m\times n\right)$$

Compatible Sizes

- Two arrays have compatible sizes if, for *each respective dimension*, either
  - has the same size, or
  - size of one of arrays is one, in which case it is automatically duplicated so that it matches the size of the other array

$$\mathbf{A}_{m\times n} \;.\!*\; \mathbf{B}_{m\times n} = \mathbf{C}_{m\times n}$$

$$\mathbf{A}_{m\times n} \;.\!*\; \mathbf{b}_{1\times n} = \mathbf{C}_{m\times n}$$

$$\mathbf{a}_{m\times 1} \;.\!*\; \mathbf{B}_{m\times n} = \mathbf{C}_{m\times n}$$

$$\mathbf{a}_{m\times 1} \;.\!*\; \mathbf{b}_{1\times n} = \mathbf{C}_{m\times n}$$

$$\mathbf{a}_{1\times n} \;.\!*\; \mathbf{b}_{m\times 1} = \mathbf{C}_{m\times n}$$

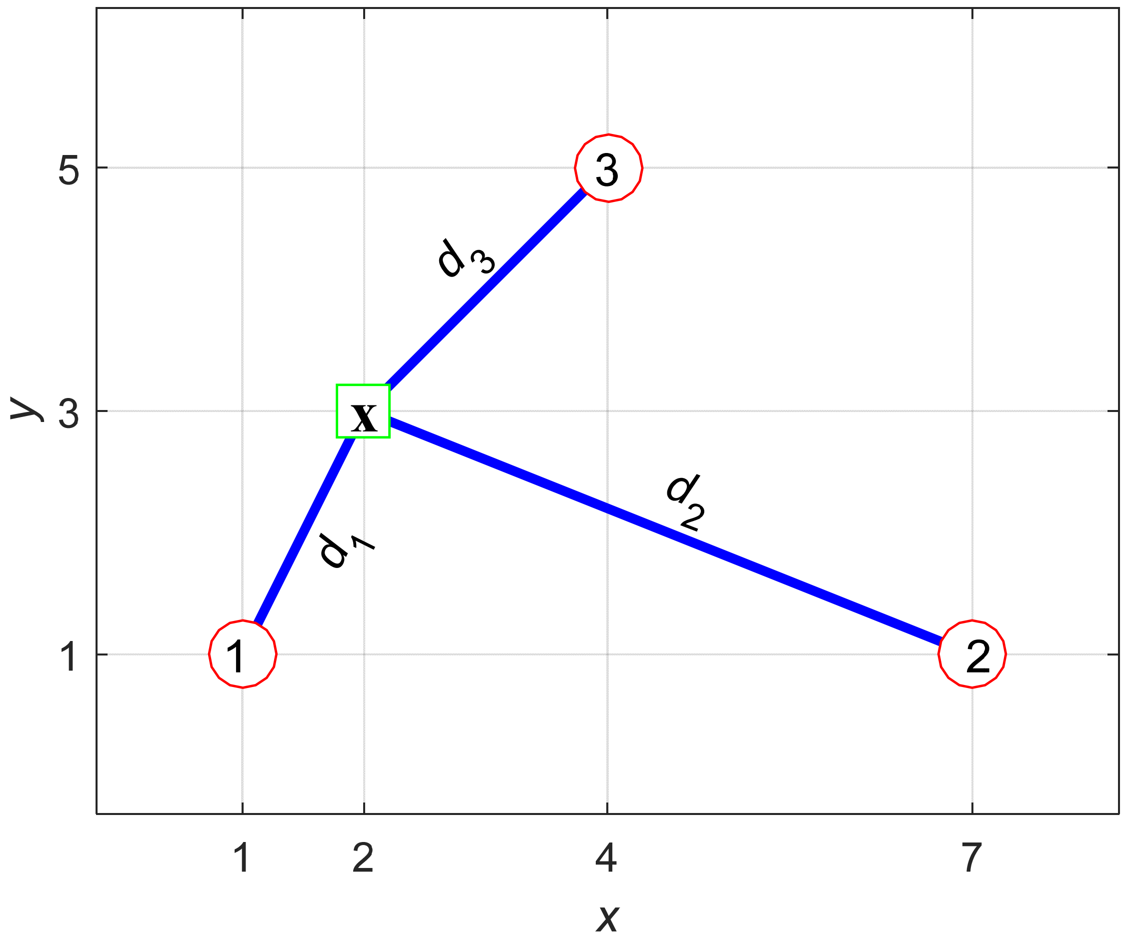
$$\cancel{\mathbf{A}_{m\times n} \;.\!*\; \mathbf{B}_{m\times p}}$$

$$\cancel{\mathbf{A}_{m\times n} \;.\!*\; \mathbf{b}_{n\times 1}}$$

$$\cancel{\mathbf{a}_{m\times 1} \;.\!*\; \mathbf{B}_{p\times n}}$$

$$\cancel{\mathbf{a}_{m\times 1} \;.\!*\; \mathbf{b}_{n\times 1}}$$

2-D Euclidean Distance

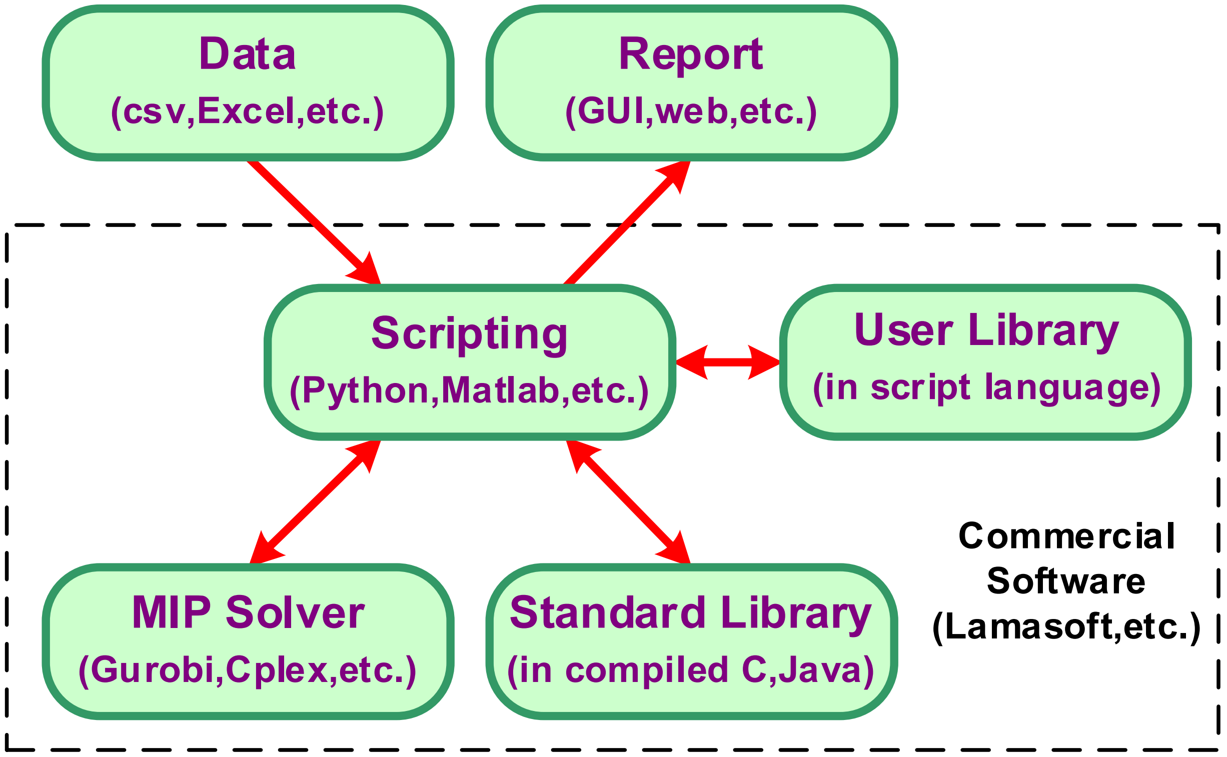


$$\mathbf{x} = \begin{bmatrix} 2 & 3 \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} 1 & 1 \\ 7 & 1 \\ 4 & 5 \end{bmatrix}$$

$$\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} \sqrt{\left(x_1 - p_{1,1}\right)^2 + \left(x_2 - p_{1,2}\right)^2} \\ \sqrt{\left(x_1 - p_{2,1}\right)^2 + \left(x_2 - p_{2,2}\right)^2} \\ \sqrt{\left(x_1 - p_{3,1}\right)^2 + \left(x_2 - p_{3,2}\right)^2} \end{bmatrix}$$

$$\mathbf{d} = \sqrt{\sum \left(\begin{bmatrix} 2 & 3 \\ 2 & 3 \\ 2 & 3 \end{bmatrix} - \begin{bmatrix} 1 & 1 \\ 7 & 1 \\ 4 & 5 \end{bmatrix} \right)^2}$$

Logistics Software Stack



- New *Julia* (1.0) scripting language
  - (almost?) as fast as C and Java (but not FORTRAN)
  - does not require compiled standard library for speed
  - uses *multiple dispatch* to make type-specific versions of functions

Basic Matlab Workflow

- Given problem to solve:
  1. Test critical steps at Command Window
  2. Copy working critical steps to a cell (&&) in script file (myscript.m) along with supporting code (can copy selected lines from Command History)
  - Repeat using new cells for additional problems
- Once all problems solved, report using:
  - `>> diary hw1soln.txt`
  - Evaluate each cell in script:
    - To see code + results: select text then Evaluate Selection on mouse menu (or F9)
    - To see results: position cursor in cell then Evaluate Current Section (Cntl+Enter)
  - `>> diary off`
- Can also report using Publish (see Matlab menu) as html or Word
- Submit all files created, which may include additional
  - Data files (myscript.mat) or spreadsheet files (myexcel.xlsx)
  - Function files (myfun.m) that can allow use to re-use same code used in multiple problems
    - All code inside function isolated from other code except for inputs/outputs: `[out1,out2] = myfun(input1,input2)`