MATLAB Basics

- a calculator
  >> 3+4
- but much more!
  x = [1 2 3 4]
  x = [1 ; 2 ; 3 ; 4]
  y = x'
  A = [1 2 ; 3 4]
- useful commands:
  >> whos
- too many variables?
  >> clear
- don't know how a function works?
  >> help sin

Vectors & matrices

eg) x = 0:10 ≡ use colon ":"

eg) x = 0:0.5:10

Want to index only part of a vector

x(1:3)

Matrices:

A = [1 2 3 ; 4 5 6 ; 7 8 9]

indexing:

A(2,3)

row, column

colon can be used to address an entire row or column:

>> A(2,:)

>> A(:,3)

Matlab's power is in it's ease of matrix & vector manipulation.

eg) B = [3 -2 1 ; 10 -8 4 ; 2 0

S = [4 2 3]

t = [-1 ; 0 ; 8]

>> A + B

>> A - 1

>> S - t : get error

>> S - t' : OK

>> A * B

>> A * t

>> A * S : get error

>> S * A : OK

>> y = B\t

Back Slash operator!!

"Matrix division"

want By = t

use y = B\t

Also:

yB = S

y = S/B

element wise operations: use "."

eg) S.*S
S.^2
A./B

Built in functions:

- the usual transcendentals and abs, rem, round, floor, ceil,

Note that all these functions can take vector inputs:

>> exp([0 1 2 3])

Useful vector functions:

max
min
length
size
sort

Horner's method for polynomials.

eg) $a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$

Q) How many operations?

n "+"

$n+(n-1)+(n-2)+\cdots+2+1$ "*"

⇒ lots of room for truncation/round-off error

Solution:

eg) $a_0 + a_1x + a_2x^2 + a_3x^3$

$= a_0 + x(a_1 + x(a_2 + xa_3))$

How how many operations?

n (3) "+"
n (3) "*"

in general:

$i = n-1$
$p = a_3$
while $i \geq 0$
$p \leftarrow px + a_i$

$p \leftarrow a_3$
$p \leftarrow px + a_2$
$p \leftarrow px + a_1$
$p \leftarrow px + a_0$

~~$p \leftarrow x^2a_3 + xp + a_ix$~~

in general:

$p = a_n$
$i = n-1$
while $i \geq 0$
$p \leftarrow px + a_i$
$i \leftarrow i-1$
end

* Now code it *

Bisection method: first as script

- need function handle

  eg) f(x) = @(x) $x^2 - 1$

- interval

  [a, b] = [0, 3]

  * → test ~~[illegible]~~ sign(f(a)) ≠ sign(f(b))

- tolerance

  Tol = $1e^{-4}$

- max iterations

  nmax = 100

```
n = 1
while n ≤ nmax
    c = (a+b)/2
    if f(c) = 0 or (b-a)/2 ≤ Tol
        return c
    end
    n = n+1
    if sign(f(c)) = sign(f(a))
        a = c
    else
        b = c
    end
end
error("Method failed.")
```

- now convert to function

- run with

  f(x) = @(x) $x^3 - x - 2$

  on [1, 2]