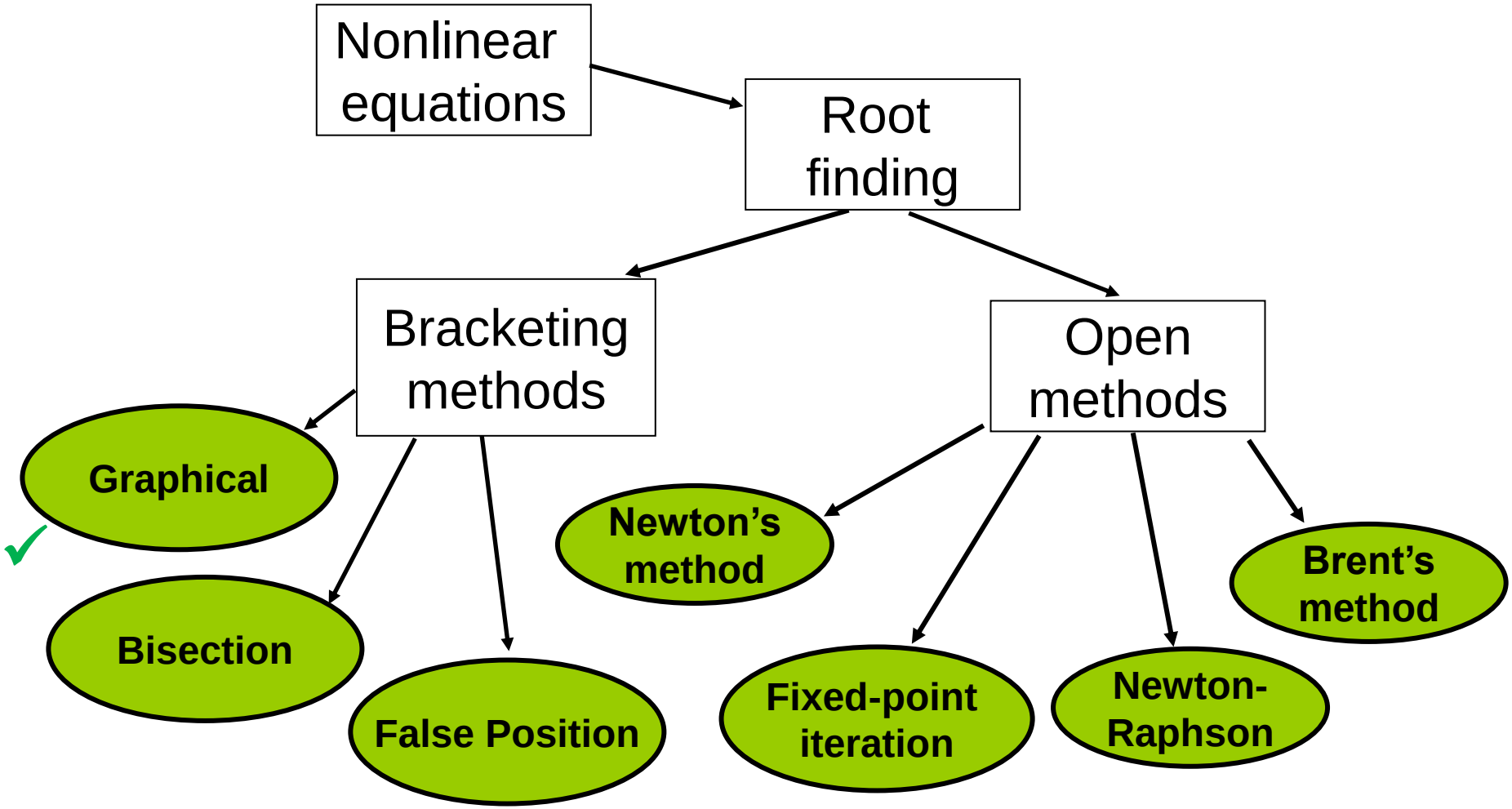
Numerical Modeling in Biomedical Systems

BME 125:305

Lecture 7

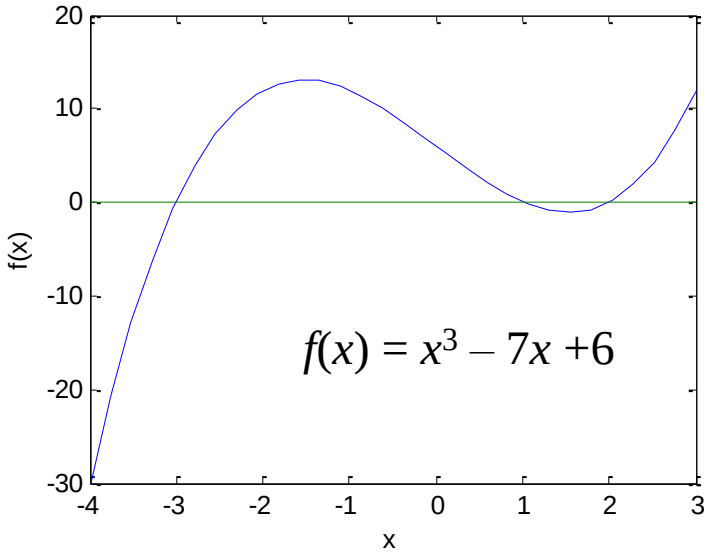
9/26/17

Nonlinear Systems – Dunn Chapter 5



- How do we solve this nonlinear equation to find x? $x^3 - 7x + 6 = 0$

$\rightarrow$ Plot $f(x) = x^3 - 7x + 6$ versus x and see where it crosses zero?



Roots near x = -3, 2, 1

- Graphical methods are not very accurate but they can give us a clue about the region where the root(s) lie

- Guess a value for x, evaluate whether $f(x) = 0$?
  - This will take a long time

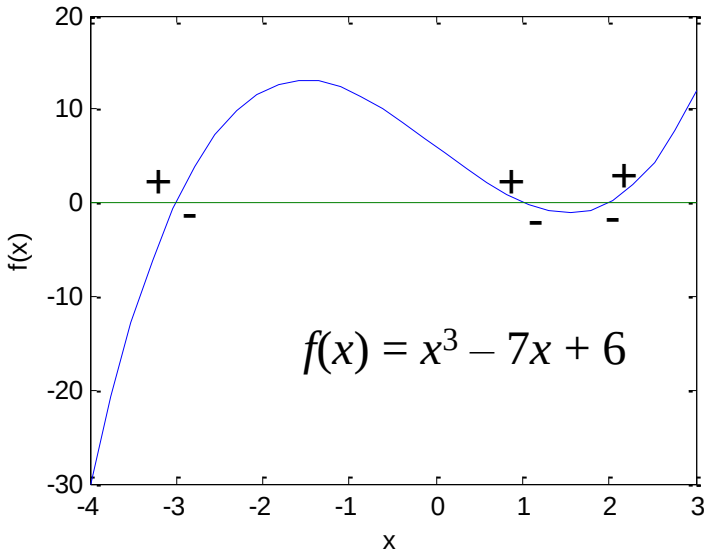
- Better to develop **systematic methods** to search for $f(x) = 0$ until our required level of accuracy is reached

Bracketing Methods

We want to find the roots of $f(x)$, i.e. the values of x where $f(x) = 0$

Where to start looking?

Bracketing methods use the fact that the value of $f(x)$ *changes sign* at each root



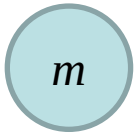
Strategy: Find two values of x, (x_1,x_2) giving *positive* and *negative* values for $f(x)$

The root must lie somewhere between these values of x

Narrow the distance between (x_1,x_2) to close in on the true root

Example - Falling object from Lecture 2

F_{up} = drag force



mass, m

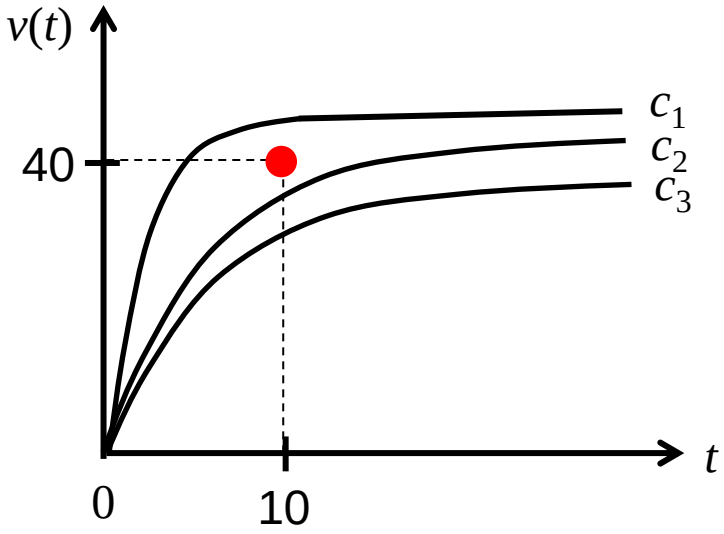
F_{down} = gravitational force

Using Newton's 2nd law leads to an analytical expression for velocity v as a function of time t and system parameters mass (m) and drag coefficient (c):

$$v(t) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right)$$

$\rightarrow$ Given g,m,c,t we can calculate v

- But what if we are given g,m,v,t and we want to calculate c ?



If an object has mass m = 68.1 kg, what drag coefficient (c) is necessary in order to reach a velocity of v = 40 m/s at time t = 10 seconds?

Problem: We can't rearrange the expression in the form $c = f(g,m,v,t)$

Recap

- How to solve a nonlinear equation? $v(t) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right)$

  Eg., find c when g = 9.8 m/s^2, m = 68.1 kg, v = 40 m/s, t = 10 s

- Solve this nonlinear equation for c: $40 = \frac{9.8(68.1)}{c}\left(1 - e^{-(c/68.1)10}\right)$

- What are the value(s) of c which satisfy: $0 = \frac{9.8(68.1)}{c}\left(1 - e^{-(c/68.1)10}\right) - 40$

- Where does the function $f(c)$ equal zero? $f(c) = \frac{9.8(68.1)}{c}\left(1 - e^{-(c/68.1)10}\right) - 40$

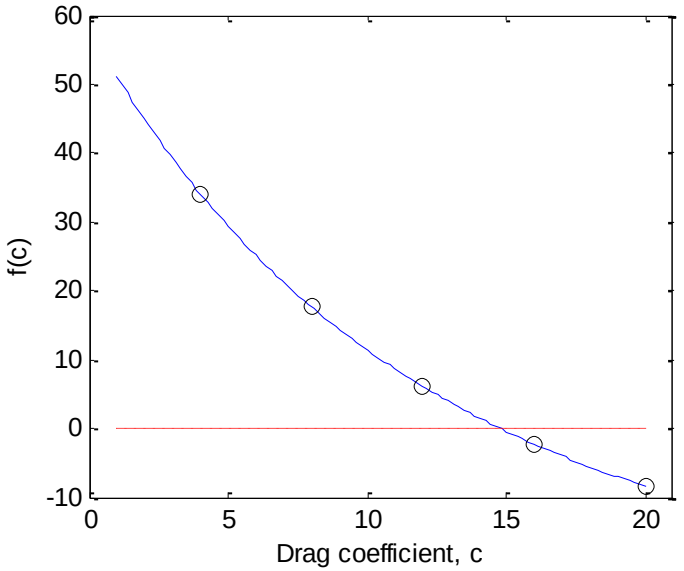
$\rightarrow$ These are just different ways of asking the same question

- We need to find the root(s) of $f(c) = \frac{9.8(68.1)}{c}\left(1 - e^{-(c/68.1)10}\right) - 40$

  I.e. the value(s) of c which lead to $f(c) = 0$

- We can start to make some guesses for c, and see what we get for $f(c)$:

Guess for c	$f(c)$
4	34.115
8	17.653
12	6.067
16	-2.269
20	-8.401



$c \approx 14.75$ looks close to zero. $v(t) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right)$ gives v = 40.059 m/s when c = 14.75

$\rightarrow$ Graphical methods are not very accurate, but they can provide us with useful initial guesses to use with other methods

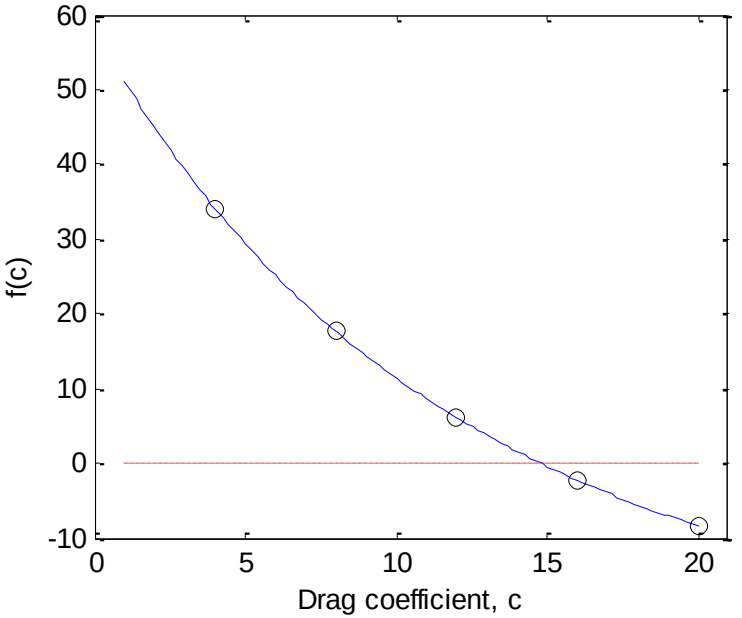
Bisection Method

- In general:

If $f(x)$ is *real* and *continuous* in the range between x_l and x_u, and

$$f(x_l)\, f(x_u) < 0,$$

then there is at least one root in the range x_l to x_u



Step 1: Choose lower (x_l) and upper (x_u) guesses for the location of the root.

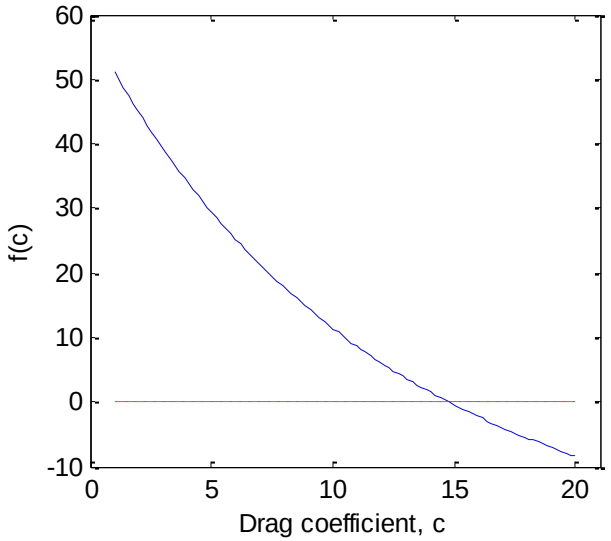
Step 2: If $f(x_l)\, f(x_u) < 0$, then the root is estimated to lie at $x_r = (x_l + x_u) / 2$

Step 3: If $f(x_l)\, f(x_r) < 0$, then the true root must lie between x_l and x_r
If $f(x_l)\, f(x_r) > 0$, then the true root must lie between x_r and x_u

Return to step 2 with the new guesses for the interval, repeat until the interval width is sufficiently small (i.e. the estimate for the root is within an acceptable tolerance)

Example - Bisection Method

$$f(c)=\frac{9.8(68.1)}{c}\left(1-e^{-(c/68.1)10}\right)-40$$



- Repeat the previous (falling mass) example using the bisection method:

Step 1: Make initial guesses for x_l and x_u

$x_l = 12$, $x_u = 16$

Check: $f(12) \times f(16) = 6.067 \times -2.269$ (i.e. < 0)

$\rightarrow$ a root exists in this range

Step 2: Estimate the root as lying <u>at the midpoint</u> of this range: $x_r = (x_l + x_u) / 2 = 14$

Step 3: Calculate $f(x_l)\, f(x_r) = f(12)\, f(14) = 9.517$. I.e. > 0, the true root does not lie between x_l and x_r

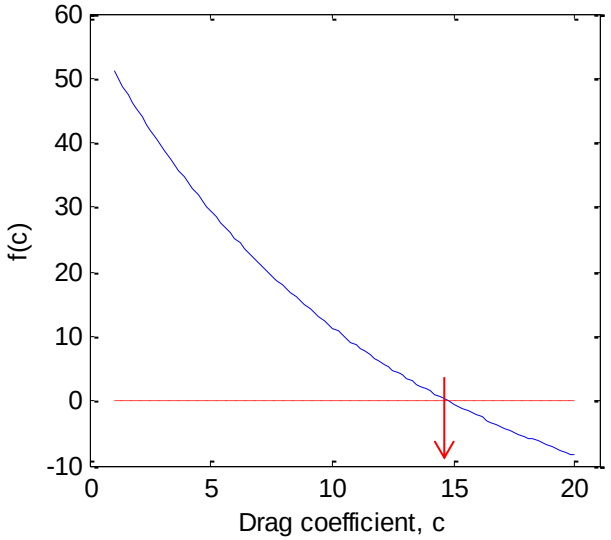
$\rightarrow$ the true root must lie between x_r and x_u (between 14 and 16)

Return to step 2 with the new guesses for the interval:

$\rightarrow$ new estimate for $x_r = (14+16)/2 = 15$

$\rightarrow$ test whether the root lies between 14-15 or 15-16

The true value for the root is $c = 14.7802$



We can evaluate the error in the value generated by the bisection method at each iteration:

As previously when evaluating the error in iterative methods, we can use the *current* and *previous* values to generate an approximate relative error, ε_a :

$$\varepsilon_a = \left| \frac{x_r^{new} - x_r^{old}}{x_r^{new}} \right| \times 100\%$$

Iteration	x_l	x_u	x_r	ε_a [%]	ε_t [%]
1	12	16	14	-	5.279
2	14	16	15	6.667	1.487
3	14	15	14.5	3.448	1.896
4	14.5	15	14.75	1.695	0.204
5	14.75	15	14.875	0.840	0.641
6	14.75	14.875	14.8125	0.422	0.219

- After 6 iterations, the approximate relative error $\varepsilon_a < 0.5\%$ (and ε_t is even lower)

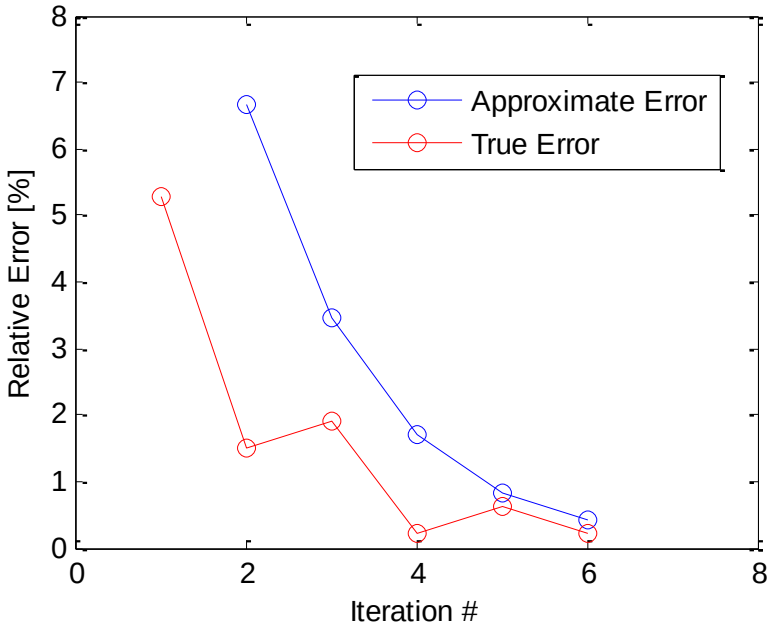
- Compare the approximate relative error to the true relative error:

Iteration	ε_a [%]	ε_t [%]
1	-	5.279
2	6.667	1.487
3	3.448	1.896
4	1.695	0.204
5	0.840	0.641
6	0.422	0.219

- With the bisection method, the true relative error is always less than the approximate relative error:

(This is a good thing for engineering design / problem solving)

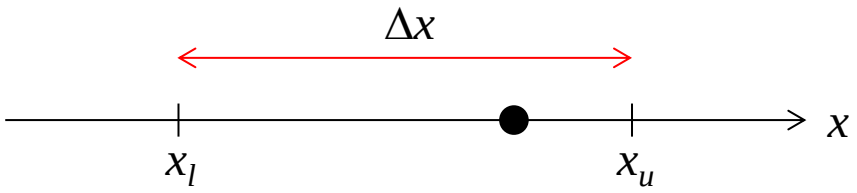
If we terminate the calculation when $\varepsilon_a < \varepsilon_s$, we can be confident that the true value is within our stopping criterion



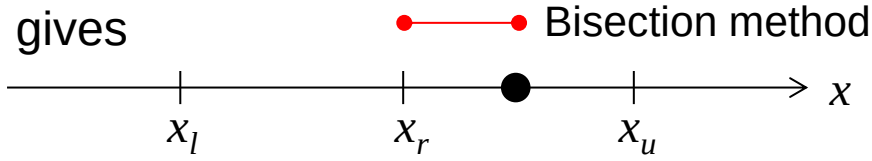
• With the bisection method, "*the true relative error is always less than the approximate relative error*". Why is this the case?

Choose lower and upper guesses for the root:

True root: ●



The bisection method gives a new estimate at x_r



→ This true error *must* always be less than $\pm \Delta x/2$:

Our estimated (approximate) error ε_a is always equal to $\Delta x/2$: $x_r^{new} - x_r^{old} = \frac{x_u - x_l}{2}$

So our relative approximate error will always <u>overestimate</u> the true error:

$$\varepsilon_a = \left| \frac{x_r^{new} - x_r^{old}}{x_r^{new}} \right| \times 100\%$$

Functions in Matlab

- Matlab has many built-in functions eg. `sin( ), exp( ), mean( ), bin2dec( ), plot( )..,`

- We can also define our own functions which can be called (by name) from the command line, or from within other scripts / m-files

function

Declare function

Syntax

```
function [out1, out2, ...] = myfun(in1, in2, ...)
```

Description

`function [out1, out2, ...] = myfun(in1, in2, ...)` declares the function `myfun`, and its inputs and outputs. The function declaration must be the first executable line of any MATLAB function.

The existing commands and functions that compose the new function reside in a text file that has a `.m` extension to its filename.

Example 1

The existence of a file on disk called `stat.m` containing this code defines a new function called `stat` that calculates the mean and standard deviation of a vector:

```
function [mean,stdev] = stat(x)
n = length(x);
mean = sum(x)/n;
stdev = sqrt(sum((x-mean).^2/n));
```

Call the function, supplying two output variables on the left side of the equation:

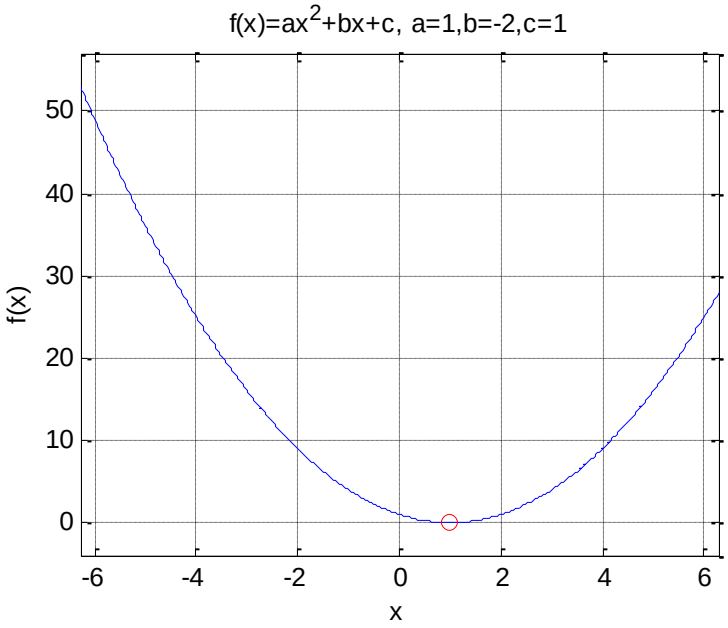
```
[mean stdev] = stat([12.7 45.4 98.9 26.6 53/1])
mean =
   47.3200
stdev =
   29.4085
```

Anonymous Functions in Matlab

- Suppose we want to find the minimum of the following function

$$f(x) = ax^2 + bx + c$$

when a = 1, b = -2, c = 1



Create an ***anonymous function*** (@):

```
clear all; close all;

a = 1; b = -2; c = 1;

f = @(x)(a*x.^2 + b*x + c);

x = [-2*pi:0.1:2*pi];
plot(x,f(x));                    % Plots function f(x) over [-2pi < x < 2pi]

title('f(x)=ax^2+bx+c, a=1,b=-2,c=1');
xlabel('x'); ylabel('f(x)');
hold on;

 % Find and plot the minimum
minimum = fminbnd(f,-2,2);      % We can pass our function directly to the minimization routine

plot(minimum,f(minimum),'ro'); % We can evaluate our function at the value "minimum"
```

Anonymous functions can have more than one variable eg: `f = @(x,y)(a*x.^2+b*y.^2)`

Example:

- A reversible chemical reaction involving reactants A and B forming product C can be written as:

$$2A + B \Leftrightarrow C \qquad \text{(Eg. } 2H_2 + CO \Leftrightarrow CH_3OH\text{)}$$

and characterized by the equilibrium relationship: , $K = \frac{C_c}{C_a^2 C_b}$

where C_i represents the concentration of constituent i.

- If the variable x represents the number of moles of product C generated by this reaction, we can write the concentration of product C as $(C_{c,0} + x)$, where $C_{c,0}$ is the *initial* concentration of C_c,.

- We can reformulate the equilibrium relationship as $K = \frac{(C_{c,0} + x)}{(C_{a,0} - 2x)^2 (C_{b,0} - x)}$

$$K = \frac{\left(C_{c,0} + x\right)}{\left(C_{a,0} - 2x\right)^2 \left(C_{b,0} - x\right)}$$

- Question:
  If the equilibrium constant for this reaction is K = 0.016, and the initial molar concentrations of constituents a, b, and c are $C_{a,0}$ = 42, $C_{b,0}$ = 28, $C_{c,0}$ = 4, respectively, determine the number of moles of product C_c (represented by variable x) generated under these conditions.

$\rightarrow$ We would like to write $x = \ldots$ and plug in our values for K, $C_{a,0}$, $C_{b,0}$, $C_{c,0}$,

But we can't.

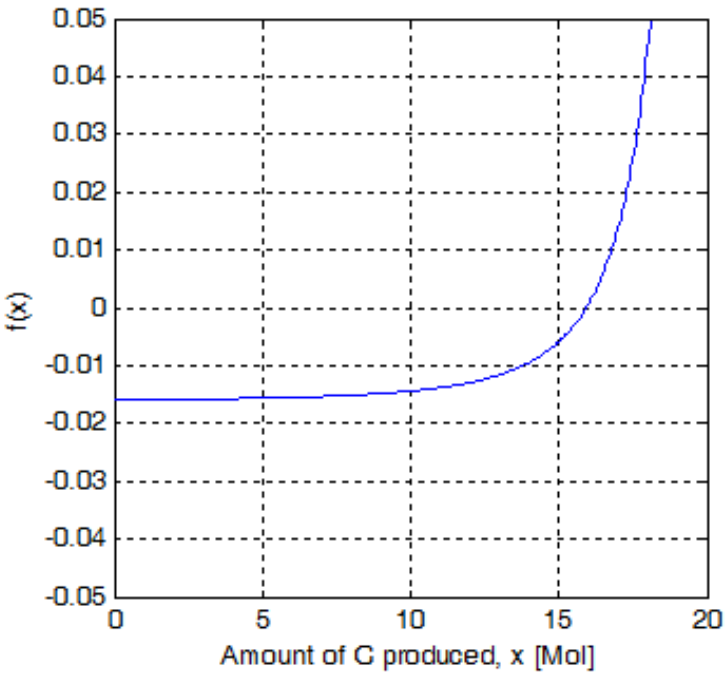
Equivalently, our problem is to find the values of x which satisfy:

$$f(x) = \frac{\left(C_{c,0} + x\right)}{\left(C_{a,0} - 2x\right)^2 \left(C_{b,0} - x\right)} - K = 0 \qquad \frac{\left(4 + x\right)}{\left(42 - 2x\right)^2 \left(28 - x\right)} - 0.016 = 0$$

I.e. "find the root(s) of $f(x) = 0$"

Step 1: Plot the function and estimate where the root lies

```
K = 0.016;
Ca = 42;
Cb = 28;
Cc = 4;

x = linspace(0,20,500);

        % This is the function we are interested in (after rearranging):
f = @(x) (Cc+x)./(((Ca-2*x).^2).*(Cb-x)) - K;

figure; plot(x,f(x),'b-');
xlabel('Amount of C produced, x [Mol]');
ylabel('f(x)');
ylim([-0.05 0.05]);
```



→ It looks like the root is around 16

Step 2: Implement the bracketing method to get an accurate estimate for the root

```
Xl = 10;                              % Initial lower and upper guesses for the root
Xu = 20;
i = 1;                                % Iteration counter starts at one
i_max = 100;                          % Max # iterations allowed
Es = 1;                               % Stopping error (when % relative approx error < Es)

while (1)
Xr = (Xl + Xu)/2;                     % New root estimate from Bracketing Method

if i > 1                              % Ensure that i > 1 (can't calculate Ea on 1st loop)
    Ea = abs((Xr - X_old)/Xr)*100;    % Calculate approx. relative error (%)
    Ea_store(i) = Ea;                 % Store the approx. relative error
end

test = f(Xl)*f(Xr);                   % Test whether f(x) changes sign between Xl and Xr

if test < 0                           % If test < 0, root must be between Xl and Xr
    Xu = Xr;                          % The calculated root becomes our new upper guess (Xu)
else                                  % If test is not < 0, root must be between Xr and Xu
    Xl = Xr;                          % The calculated root becomes our new lower guess (Xl)
end

X_old = Xr;                           % Assign the latest root to X_old for use in the next loop
root_store(i) = Xr;                   % Store the latest root for plotting later

if i > 1                              % If we are on the 2,3,4 iteration, check the approx. error
    if Ea < Es | i > i_max
        break
    end                               % If either condition is true, exit the loop
end

i = i + 1;                            % Increment the iteration counter for the next loop
end
```

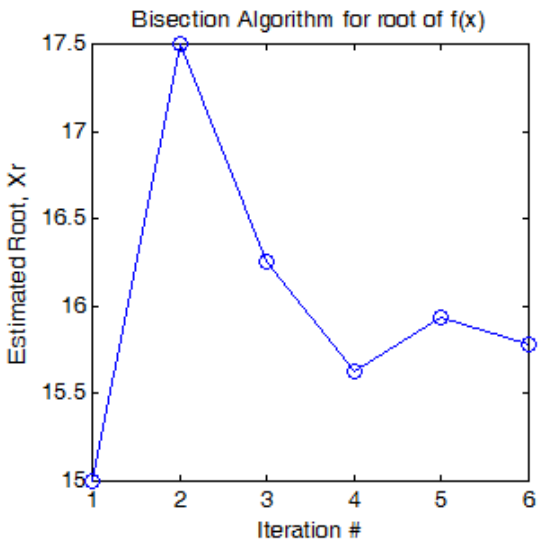
Step 3: Plot out the results

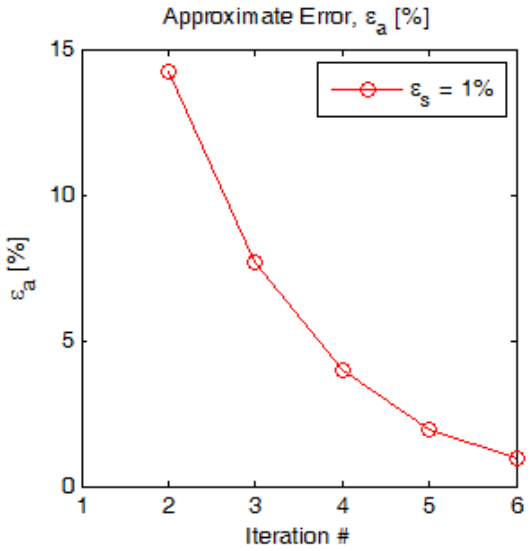
```
disp('Calculated root'); Xr
disp('# of Iterations'); i
disp('Approximate (Relative) Error'); Ea

figure;
subplot(1,3,1); plot([1:i],root_store,'-bx');
xlabel('Iteration #'); ylabel('Estimated Root, Xr'); axis square;
title('Bisection Algorithm for root of f(x)'); xlim([1 i]);

subplot(1,3,2); plot([2:i],Ea_store(2:end),'b-');
xlabel('Iteration #'); ylabel('\epsilon_a [%]'); axis square;
title('Approximate Error, \epsilon_a [%]'); xlim([1 i]);
legend(['\epsilon_s = ',num2str(Es),'%']);

subplot(1,3,3); plot([2:i],Ea_store(2:end),'b-');
xlabel('Iteration #'); ylabel('\epsilon_a [%]'); axis square;
title('Approximate Error, \epsilon_a [%]'); ylim([0 5]); xlim([3.5 7]);
legend(['\epsilon_s = ',num2str(Es),'%']);
```

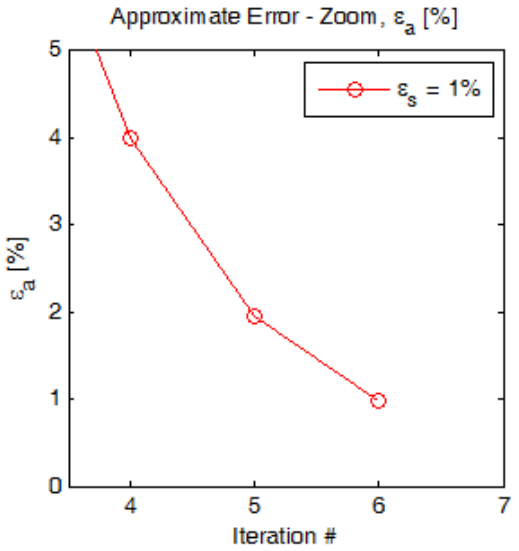
Command Window

New to MATLAB? Watch this Video, see Demos, or

```
Calculated root

Xr =

   15.7813

# of Iterations

i =

     6

Approximate (Relative) Error

Ea =

    0.9901
```

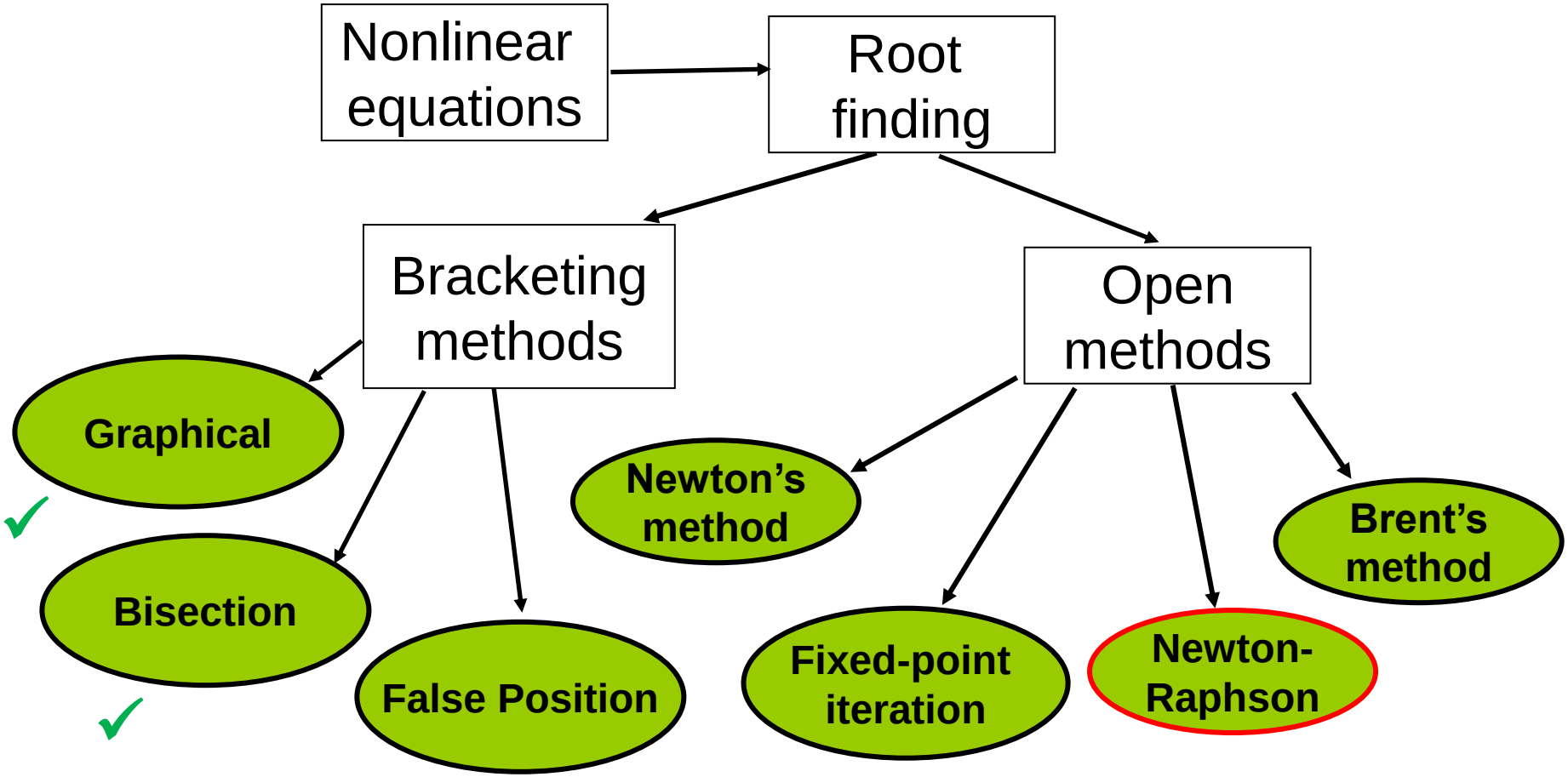






Nonlinear Systems – Dunn Chapter 5

Next time: False-position, Newton's method, Fixed point iteration



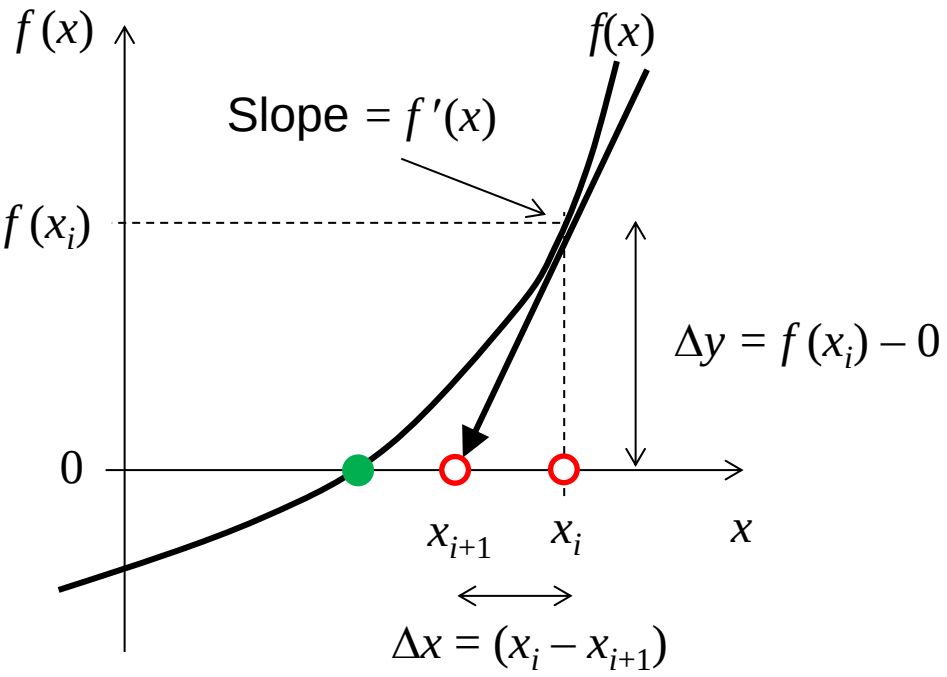
Open Methods for Root Finding: Newton-Raphson

- The most widely used method for root finding

1) Start with one initial guess for the root (x_i)

2) Draw a line tangent to the point at [x_i, $f(x_i)$]

3) Our new estimate for the root corresponds to the x-value where the tangent from [x_i, $f(x_i)$] crosses the x-axis



→ The slope of the tangent line $f'(x_i)$ equals $$f'(x_i) = \frac{\Delta y}{\Delta x} = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

Rearranging this in terms of the new root estimate (x_{i+1}): $$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Example:

- Use the Newton-Raphson method to find the root of $f(x) = e^{-x} - x$

Given a function $f(x)$, its derivative $f'(x)$, and an initial guess for the root (x_i, $i = 0$), the Newton-Raphson equation gives us a new root estimate (x_{i+1}):

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Function: $f(x) = e^{-x} - x$

Derivative: $f'(x) = -e^{-x} - 1$

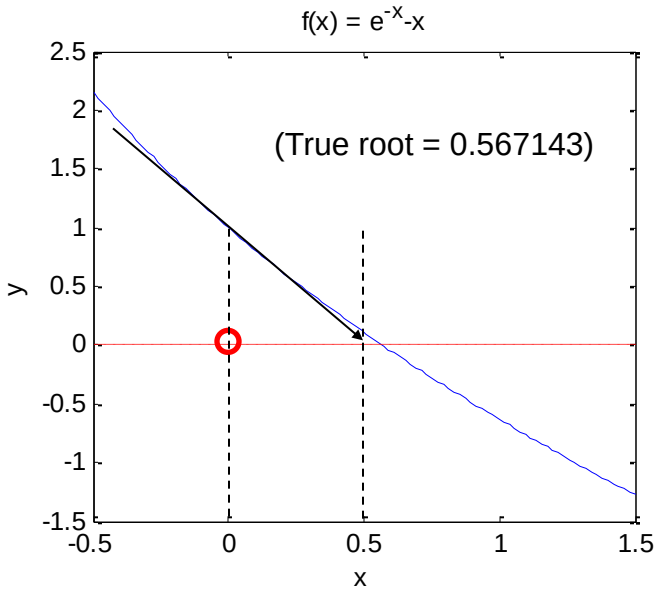
Initial guess: $x_0 = 0$

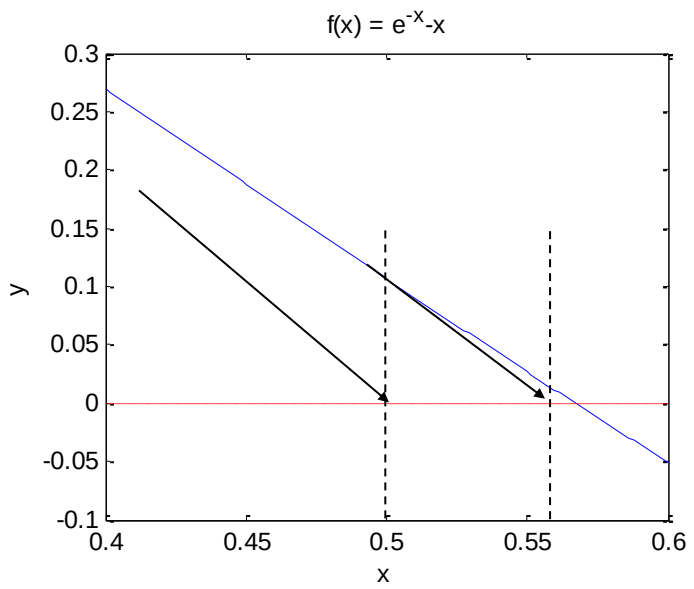
Use these expressions in the N-R equation:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$x_{i+1} = x_i - \frac{e^{-x_i} - x_i}{-e^{-x_i} - 1}$$

Original problem: Find the root of $f(x) = e^{-x} - x$

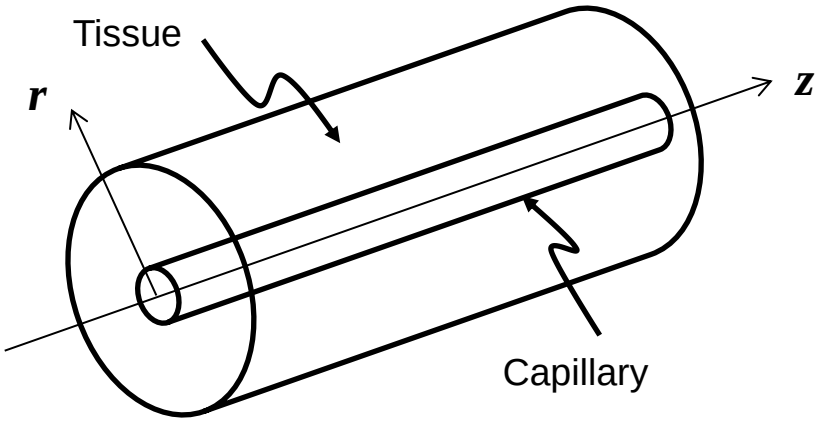




	Newton-Raphson	
Iteration	x_i	ε_t [%]
0	0	100
1	0.500000	11.8
2	0.566311	0.147
3	0.567143	2.2×10^{-5}
4	0.567143	$< 10^{-8}$

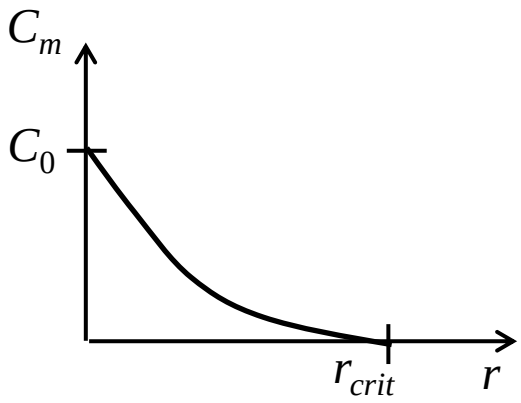
Transport Example: Krogh Tissue Cylinder

- Model for studying transport of metabolites from capillaries to surrounding tissue



Property	Value
Capillary inner radius, r_c	0.0005 cm
Capillary wall thickness, t_m	5×10^{-5} cm
Average blood velocity, V	0.041 cm/s
Entering concentration, C_0	5 μm/cm^3
Tissue glucose consumption rate, R_0	0.03 μm/cm^3.s
Glucose tissue diffusivity, D_T	8×10^{-4} cm^2/s
Mass transfer coefficient, K_0	5.75×10^{-5} cm/s

- The critical radius r_{crit} is the distance at which the metabolite concentration is reduced to zero:



- r_{crit} varies with distance along the capillary according to:

$$R^2 \ln\left(R^2\right) - R^2 + 1 - \left[\frac{4D_T C_0}{R_0\left(r_c + t_m\right)^2}\right] + 4\frac{D_T}{Vr_c^2}\left[R^2 - 1\right]z + \frac{2D_T}{r_c K_0}\left[R^2 - 1\right] = 0$$

where $R = \dfrac{r_{crit}}{r_c + t_m}$.

Question (i) : What is the critical tissue radius (r_{crit}) where glucose is no longer supplied to cells, at a distance z = 0.025 cm under the conditions listed in the table?

$$R^2 \ln\left(R^2\right) - R^2 + 1 - \left[\frac{4D_T C_0}{R_0\left(r_c + t_m\right)^2}\right] + 4\frac{D_T}{Vr_c^2}\left[R^2 - 1\right]z + \frac{2D_T}{r_c K_0}\left[R^2 - 1\right] = 0 \qquad \text{where} \quad R = \frac{r_{crit}}{r_c + t_m}.$$

We cannot rearrange this equation in the form r_{crit} = …

We need to "find the root" of $f(r_{crit}) = 0$. (The value of r_{crit} which satisfies $f(r_{crit}) = 0$).

Let's simplify this nonlinear expression:

$$R^2 \ln\left(R^2\right) = A + \left(B - Dz\right)\left(R^2 - 1\right)$$

where $A = \dfrac{4D_T C_0}{R_0\left(r_c + t_m\right)^2}$ $B = 1 - \dfrac{2D_T}{r_c K_0}$ $D = \dfrac{4D_T}{Vr_c^2}$

(A, B, and D are just constants which can be found from values given in the table)

…and z is given in the question.

Property	Value
Capillary inner radius, r_c	0.0005 cm
Capillary wall thickness, t_m	5×10^{-5} cm
Average blood velocity, V	0.041 cm/s
Entering concentration, C_0	5 μm/cm³
Tissue glucose consumption rate, R_0	0.03 μm/cm³.s
Glucose tissue diffusivity, D_T	8×10^{-4} cm²/s
Mass transfer coefficient, K_0	5.75×10^{-5} cm/s

- Let's make this expression even simpler: Let $x = R^2$

$$R^2 \ln\left(R^2\right) = A + \left(B - Dz\right)\left(R^2 - 1\right) \longrightarrow x\ln\left(x\right) = A + \left(B - Dz\right)\left(x - 1\right)$$

If we solve this nonlinear equation for x (find the roots of $f(x) = x\ln(x) - A - E(x-1) = 0$) then we can easily convert the answer from $x \to R \to r_{crit}$:

$$E = B - Dz$$

Write Matlab code to implement the Newton-Raphson method to solve this problem:

First steps:

1) Define the relevant constants
2) Define the function we want to solve (find the root(s) of)
3) Plot out the function to obtain a rough idea of where the root(s) lie

Next steps:

4) Define additional constants (initial guess for the root (x_1), stopping error (ε_s),…)
5) Set up a loop to keep generating and storing new estimates for the root, using the Newton-Raphson method

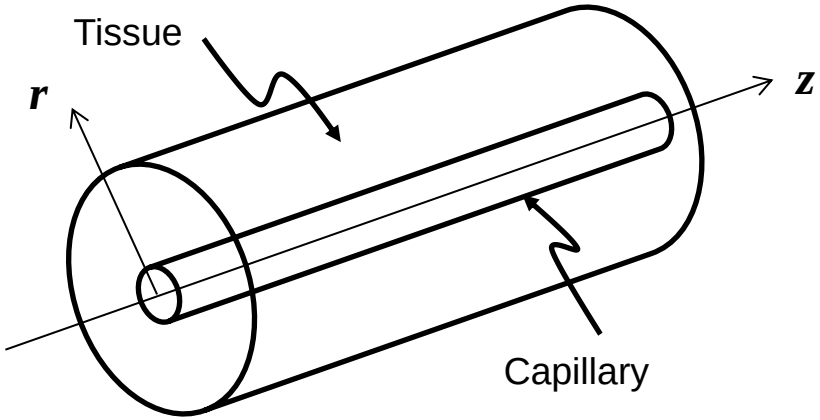
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

6) Estimate and store the error (ε_a) at each iteration of the loop (if i >1). Exit the loop if this error is small enough ($|\varepsilon_a| < \varepsilon_s$)
7) Plot out the root estimates and error estimates as a function of iteration number

Question (part ii):

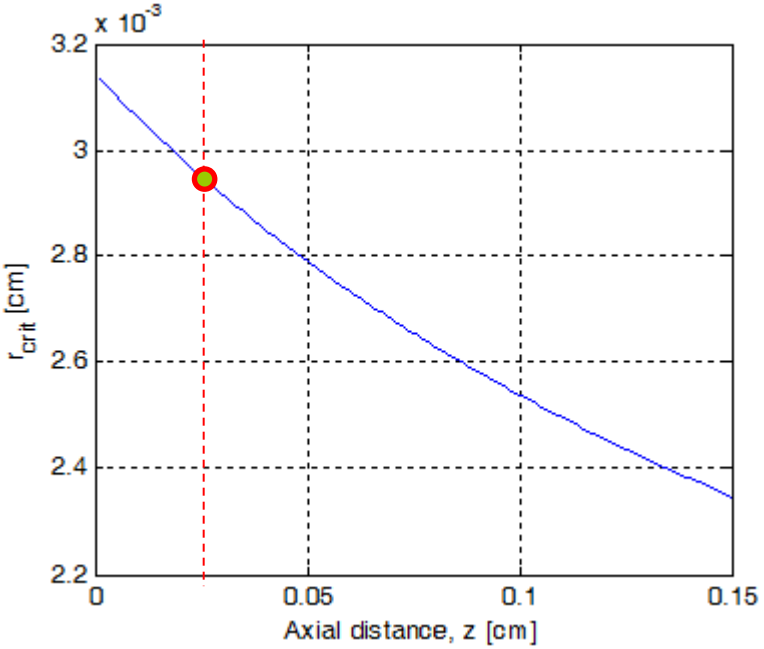
You just calculated r_{crit} for a single axial location of z = 0.025 cm.

How does r_{crit} vary as z ranges from 10 μm to 0.15 cm?



- Modify your code from part (i) to calculate r_{crit} at lots of z values between 10 μm and 0.15 cm

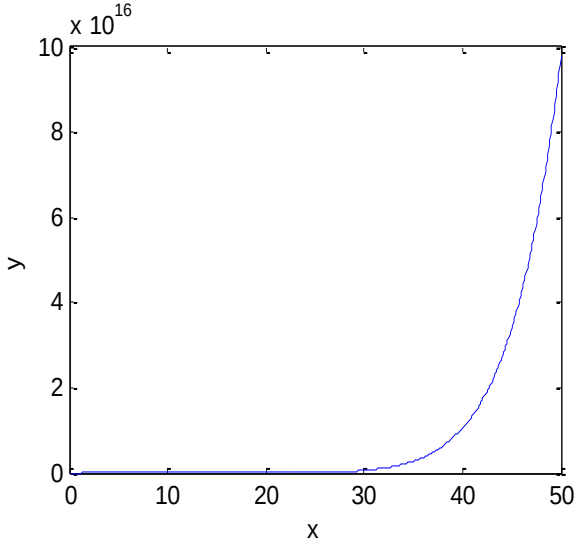
- Store the values calculated for r_{crit} at each z value. Plot r_{crit} versus distance z.

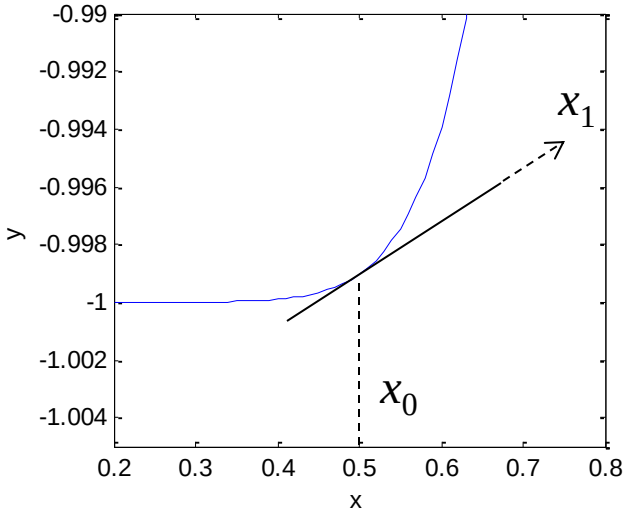


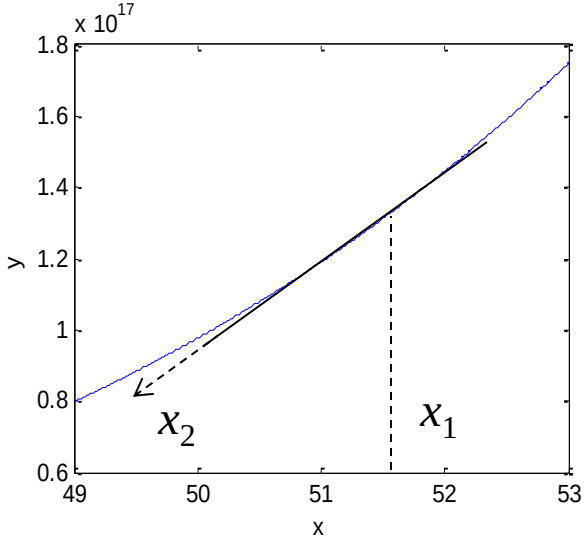
Problems with the Newton-Raphson method: (1) Slow Convergence

Example: $f(x) = x^{10} - 1$

Find the root from an initial guess of $x_0 = 0.5$







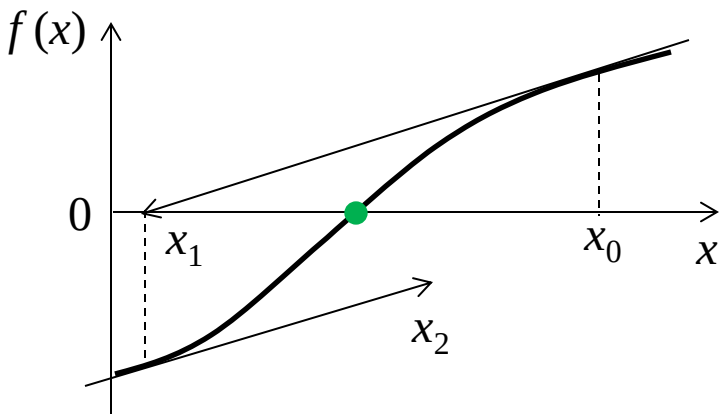
The new guess is the x value where the tangent from $f(x_0)$ crosses the x-axis

Iteration	x_i
0	0.5
1	51.65
2	46.485
3	41.8365

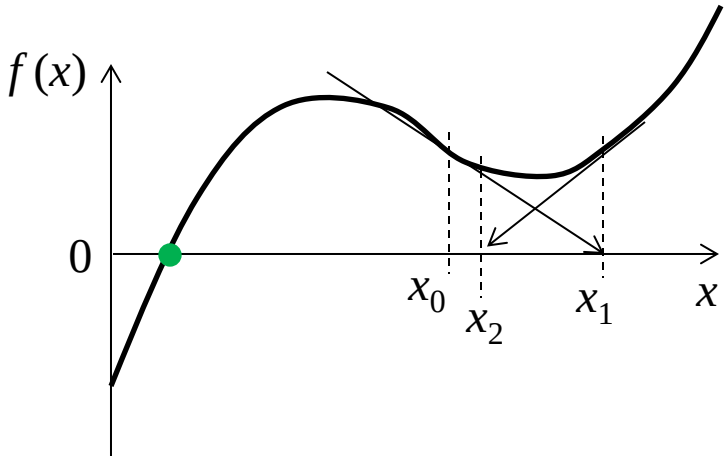
The algorithm is slowly converging on the true root ($x_r = 1$)

Due to
(1) the nature of the function $f(x)$
(2) our initial guess for the root, x_0

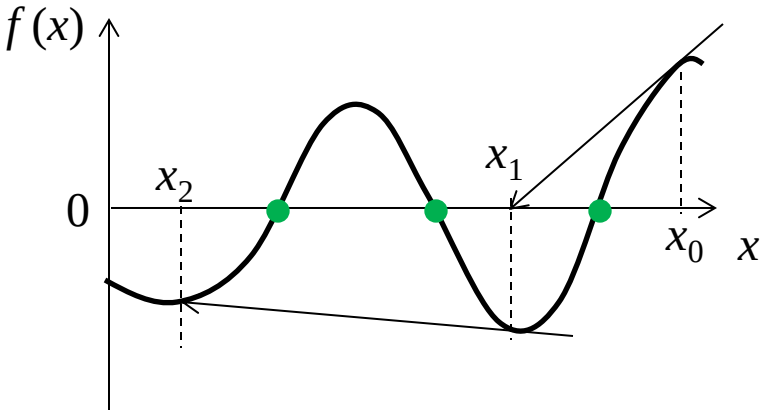
Problems with the Newton-Raphson method: (2) No Convergence



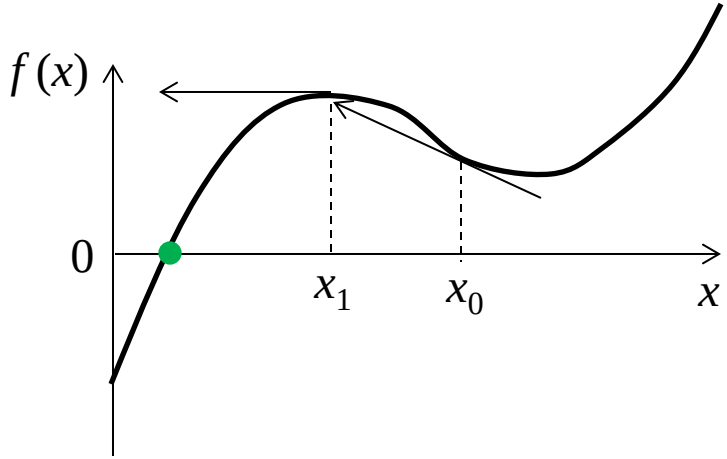
- Point of inflection near the root



- Trapped at local max / min



- Slope of $f(x) \approx 0$ near root



- Slope of $f(x) = 0$ at estimated root

Problems with the Newton-Raphson method: No Convergence

- There is no general convergence rule for the Newton-Raphson method
- Convergence depends on the nature of $f(x)$ and the initial guess x_0

$\rightarrow$ Computational methods using N-R should include some safety features:

1. Substitute the final estimate for the root back into $f(x)$. Check whether the result is close to zero
2. Limit the number of iterations to some maximum value
3. Check whether the slope $f'(x) = 0$ at any iteration
4. Make the program plot the estimated root at each iteration for inspection

Nonlinear Systems – Dunn Chapter 5