CS 3200
Introduction to Scientific Computing

Instructor: **Martin Berzins**

Topic: **Solving Nonlinear Equations**

Nonlinear Equations $\mathbf{f(x) = 0}$

- Find a root **x** of **f** where both **x** and **f(x)** are n-vectors
- There may be one, none or multiple solutions (roots) !
- The notation **f(x)** is short-hand for the vector function

$$\begin{bmatrix} f_1\left(x_1, x_2, \ldots, x_n\right) \\ f_2\left(x_1, x_2, \ldots, x_n\right) \\ \vdots \\ f_n\left(x_1, x_2, \ldots, x_n\right) \end{bmatrix} = 0$$

We consider only a scalar case
f(x) = 0

Bisection Method

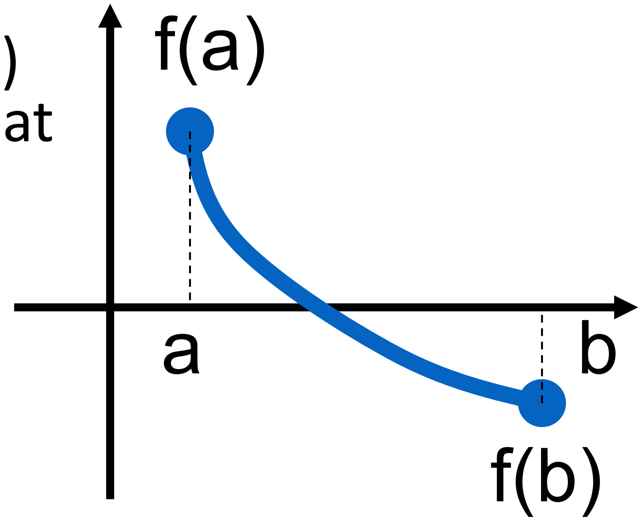
- The **Bisection method** is one of the simplest methods to find a zero of a nonlinear function $f(x) = 0$.
- Start with an initial interval that is known to contain a zero of the function.
- Reduce the interval by dividing it into two equal parts, perform a simple test and based on the result of the test, half of the interval is thrown away.
- The procedure is repeated until the interval size is small enough for accuracy purposes.

Intermediate Value Theorem

- Let f(x) be defined on the interval [a,b].

- **Intermediate value theorem:**

  if a function is continuous and f(a) and f(b) have different signs then the function has at least one zero in the interval [a,b].



Hence we can sub-divide the interval and apply the process recursively.

Note each iteration narrows down the interval containing the root by a factor of 2

After n iterations the interval is $$\frac{(b-a)}{2^n}$$

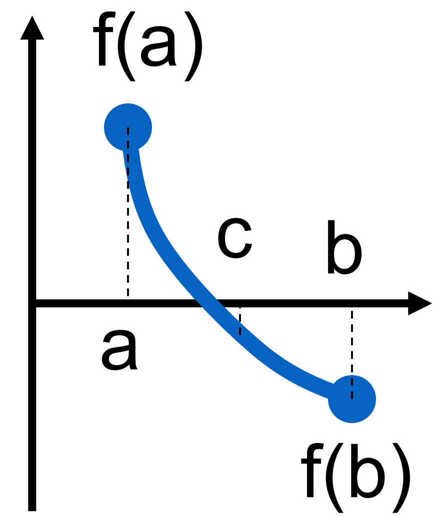
Bisection Algorithm

Loop

1. Compute the mid point c=(a+b)/2
2. Evaluate f(c)
3. If   f(a) f(c) < 0 then new interval [a, c]

   If   f(a) f(c) > 0 then new interval [c, b]

End loop



$$if\ \frac{(b-a)}{2^n} < tol$$

$$then$$

$$n > \log_2\left(\frac{(b-a)}{tol}\right),$$

$$e.g. \log_2(10^9) \approx 30$$

After n steps subdivide
Interval by 2^n
How big does n have to be?

Bisection Summary

- Bisection is foolproof
- Bisection is slow

Newton (1671) -Raphson (1690) Method for a Single Equation

Raphson’s method is closer to the one we use today.

Suppose we are at the m th iteration of solving f(x)= 0

1. For each guess of x, $x^{(m)}$, define

   $\Delta x^{(m)} = x - x^{(m)}$

2. Represent $f(x)$ by a Taylor series about $f(x^{(m)})$

$$f(x) = f(x^{(m)}) + \frac{df(x^{(m)})}{dx}\Delta x^{(m)} + \frac{1}{2}\frac{d^2 f(x^{(m)})}{dx^2}\left(\Delta x^{(m)}\right)^2 + h.o.t.$$

Newton-Raphson Method,

3. Approximate $f(x)$ by neglecting higher order terms (h.o.t.)

$$f(x) \;=\; 0 \;\approx\; f(x^{(m)}) + \frac{df(x^{(m)})}{dx}\Delta x^{(m)}$$

4. Use this approximation to solve for $\Delta x^{(m)}$

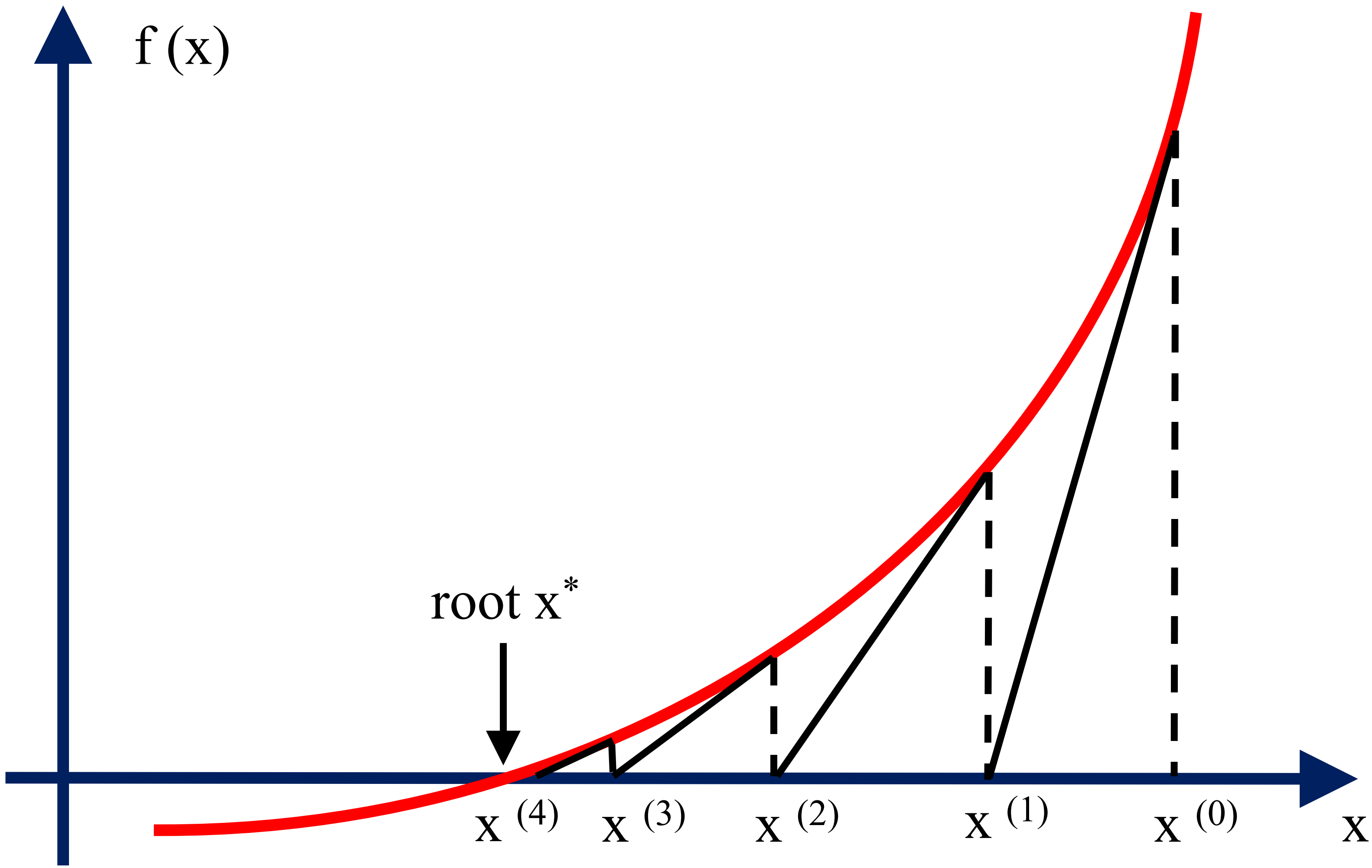
$$\Delta x^{(m)} \;=\; -\left[\frac{df(x^{(m)})}{dx}\right]^{-1} f(x^{(m)})$$

5. Solve for a new estimate of x

$$x^{(m+1)} = x^{(m)} + \Delta x^{(m)}$$

6. Continue until convergence

Newton's Method for a Single Equation



Newton-Raphson Example

Solve $f(x) = x^2 - 2 = 0$

The equation we use to update our estimate is

$$\Delta x^{(m)} = -\left[\frac{df(x^{(m)})}{dx}\right]^{-1} f(x^{(m)})$$

$$\Delta x^{(m)} = -\left[\frac{1}{2x^{(m)}}\right]((x^{(m)})^2 - 2)$$

$$x^{(m+1)} = x^{(m)} + \Delta x^{(m)}$$

$$x^{(m+1)} = x^{(m)} - \left[\frac{1}{2x^{(m)}}\right]((x^{(m)})^2 - 2)$$

Newton-Raphson Example

$$x^{(m+1)} = x^{(m)} - \left[\frac{1}{2x^{(m)}}\right]((x^{(m)})^2 - 2)$$

Guess $x^{(0)} = 1$. The iteration gives

m	$x^{(m)}$	$f(x^{(m)})$	$\Delta x^{(m)}$
0	1	-1	0.5
1	1.5	0.25	-0.08333
2	1.41667	6.953×10^{-3}	-2.454×10^{-3}
3	1.41422	6.024×10^{-6}	

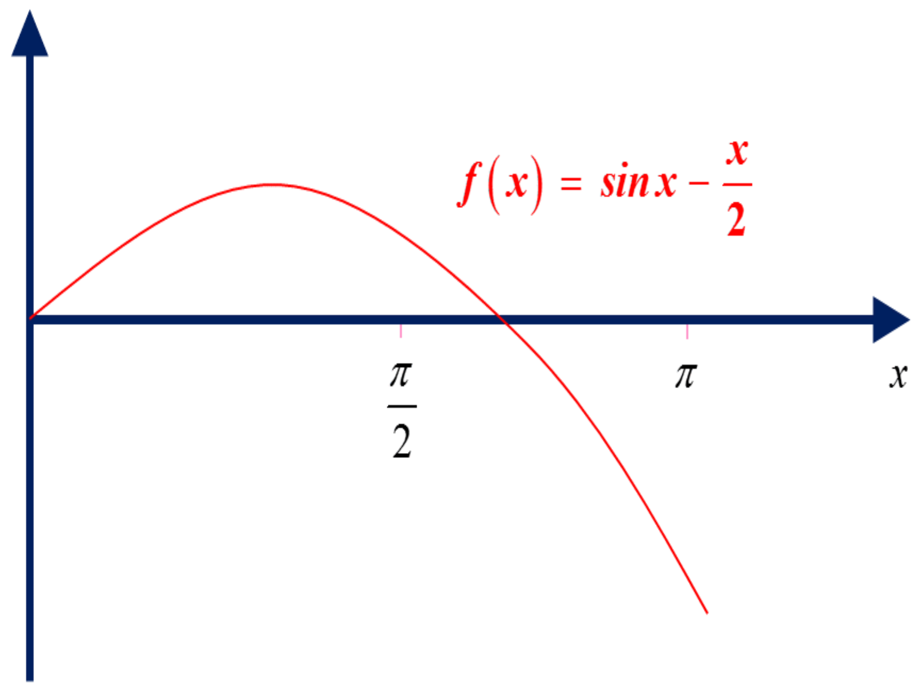
Example 2

- Find the positive root of $\sin x - 0.5x = 0$ using Newton's method starting $x^{(0)} = \pi/2$

$$x^{(1)} = x^{(0)} - \frac{f\left(x^{(0)}\right)}{f'\left(x^{(0)}\right)}$$

$$= 1.57079 - \frac{1.0 - 0.78539}{0 - 0.5}$$

$$= 2.00001$$



iteration number m	
2	1.90100
3	1.89551
4	1.89549

Newton's Method may

Converge

Or converge to the wrong root

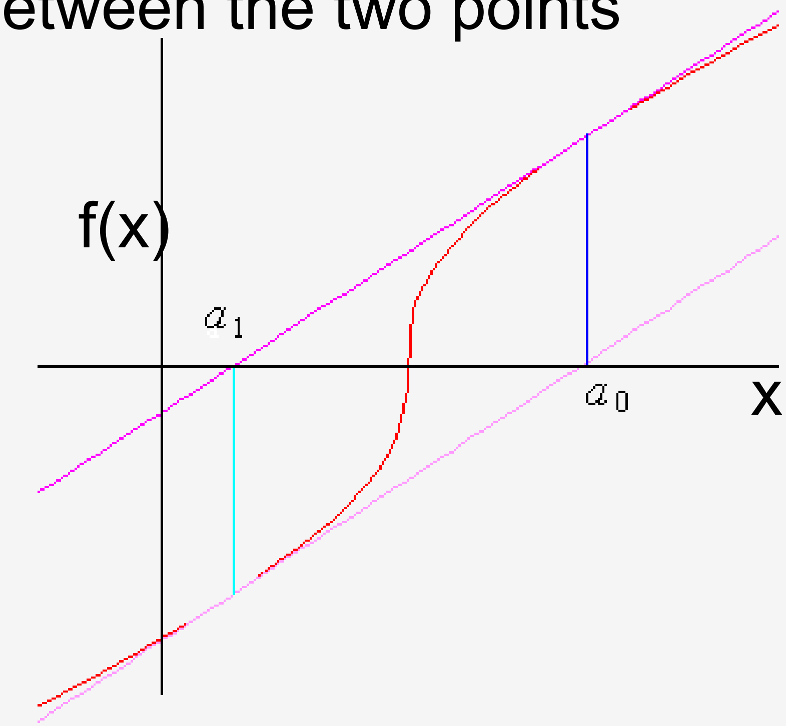
Diverge

Or get stuck

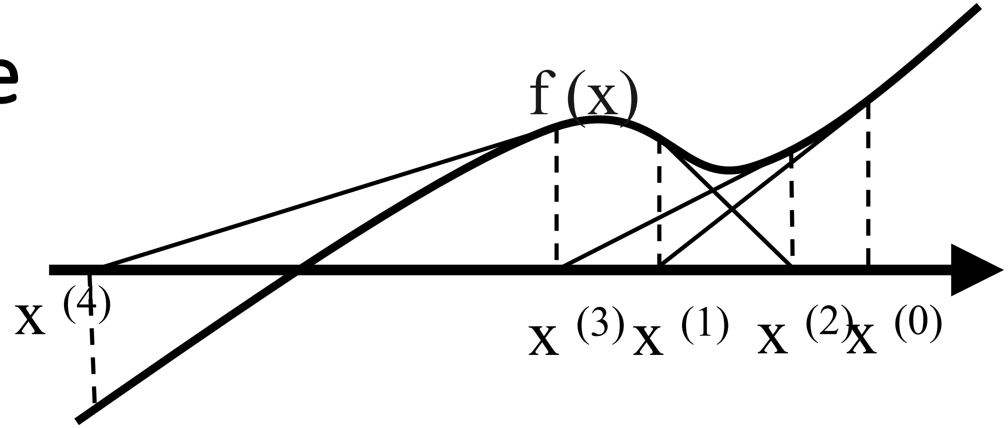
Convergence only if we start "close enough"

There are globally convergent extensions

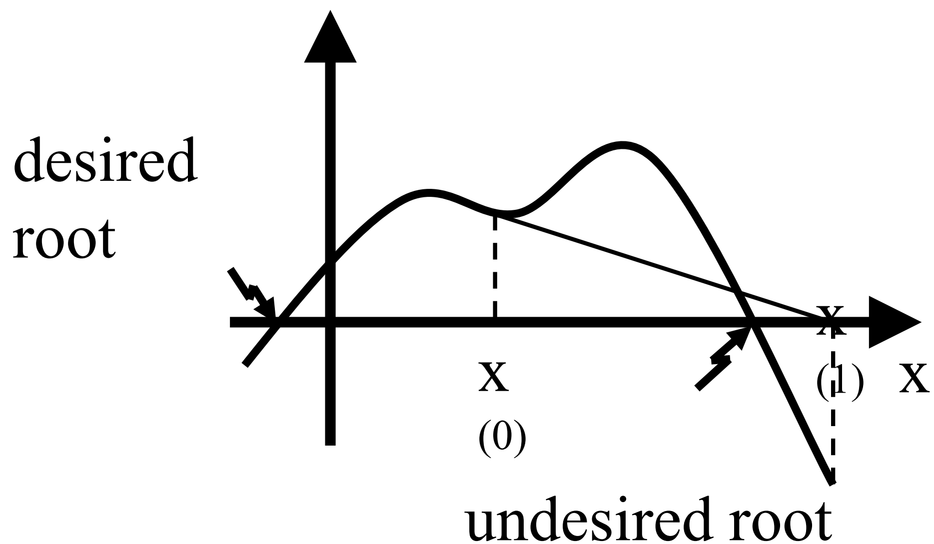
Method cycles endlessly between the two points



Oscillatory Convergence



Convergence to an Unwanted Root



Newton Raphson Secant Method

This is really just using a finite difference approximation to the derivative

$$\frac{df}{dx}\left(x^{(m)}\right) \approx \frac{f\left(x^{(m)}\right) - f\left(x^{(m-1)}\right)}{\left(x^{(m)} - x^{(m-1)}\right)}$$

$$\frac{df}{dx}(x^{(m)}) \approx \frac{f(x^{(m)}) - f(x^{(m-1)})}{(x^{(m)} - x^{(m-1)})}$$

As

$$f(x^{(m-1)}) = f(x^{(m)}) - (x^{(m)} - x^{(m-1)}) \left.\frac{df}{dx}\right|_{x^{(m)}} + \frac{(x^{(m)} - x^{(m-1)})^2}{2} \left.\frac{d^2 f}{dx^2}\right|_{x^{(m)}} + \ldots.$$

$$f(x^{(m)}) - f(x^{(m-1)}) = (x^{(m)} - x^{(m-1)}) \left.\frac{df}{dx}\right|_{x^{(m)}} - \frac{(x^{(m)} - x^{(m-1)})^2}{2} \left.\frac{d^2 f}{dx^2}\right|_{x^{(m)}} + \ldots.$$

then

$$\frac{df}{dx}(x^{(m)}) = \frac{f(x^{(m)}) - f(x^{(m-1)})}{(x^{(m)} - x^{(m-1)})} + \frac{(x^{(m)} - x^{(m-1)})}{2} \left.\frac{d^2 f}{dx^2}\right|_{x^{(m)}} + \ldots.$$

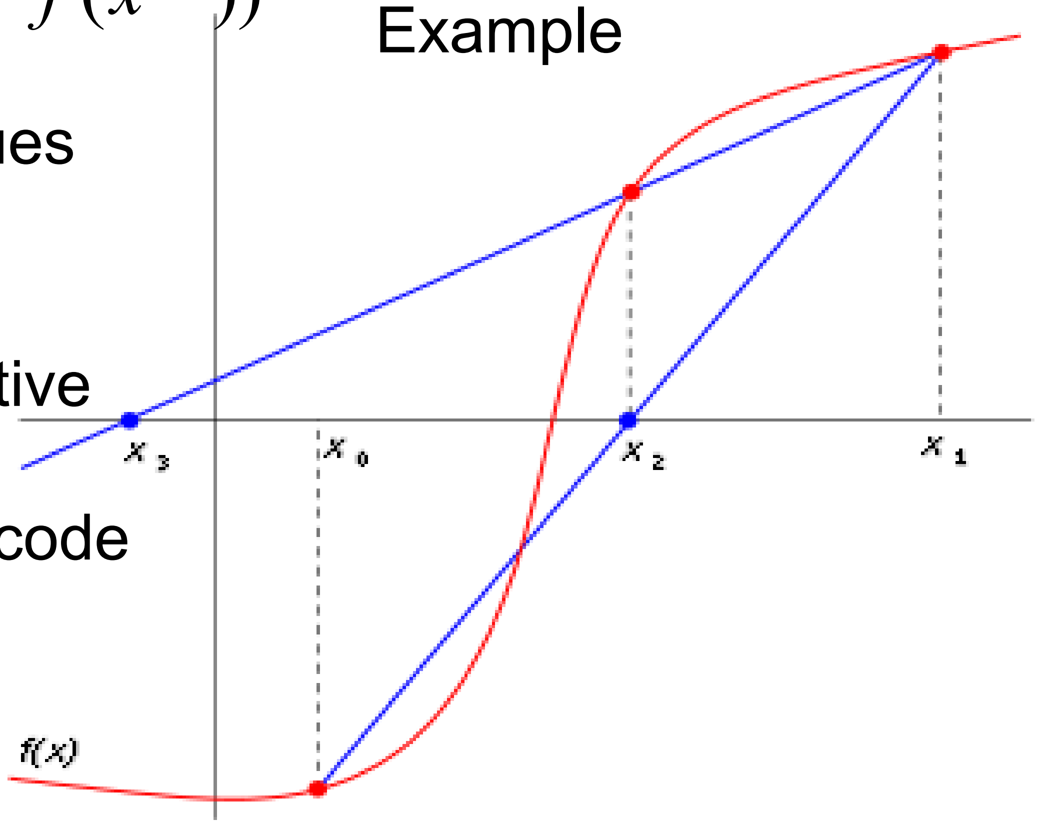
Error

Newton Raphson Secant Method

$$x^{(m+1)} = x^{(m)} - f(x^{(m)})\frac{(x^{(m)} - x^{(m-1)})}{(f(x^{(m)}) - f(x^{(m-1)}))}$$

$$x^{(2)} = x^{(1)} - f(x^{(1)})\frac{(x^{(1)} - x^{(0)})}{(f(x^{(1)}) - f(x^{(0)}))}$$

Example



Need to pick two starting values

This is useful when the derivative
Is not available ?
E.G. *f(x)* defined by complex code

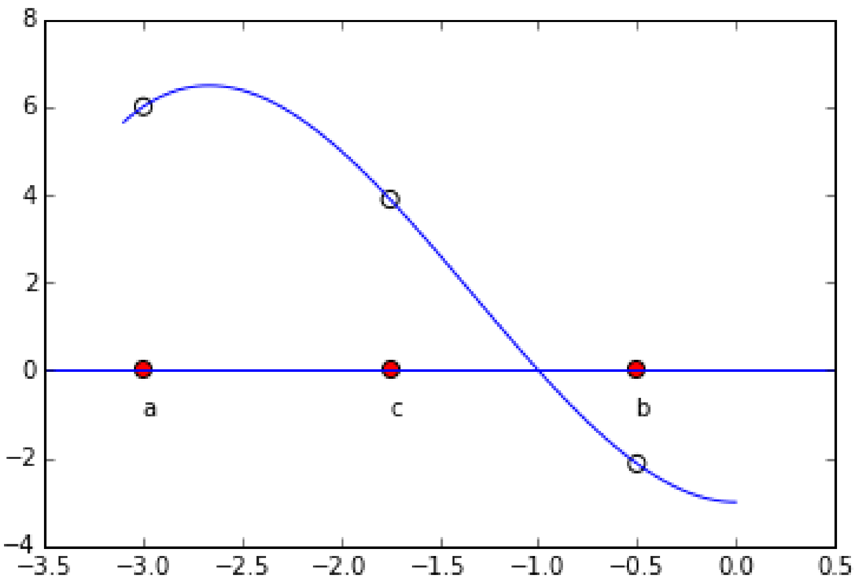
Fzero – A globally convergent polyalgorithm

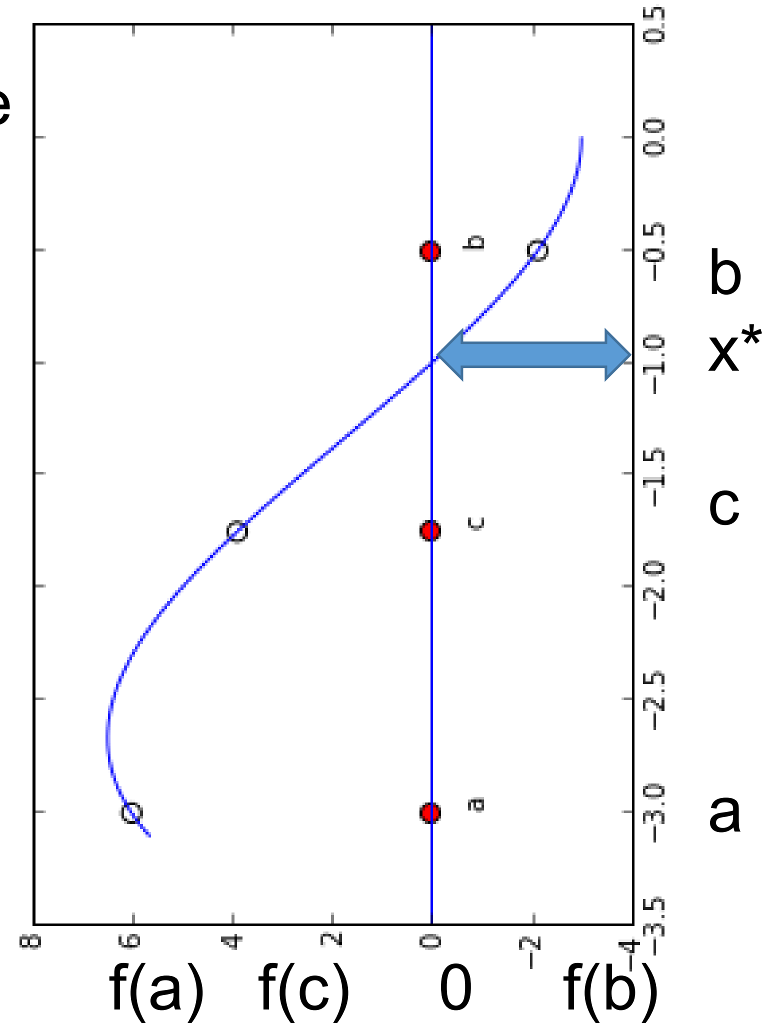
- Fzero uses a combination of three algorithms to get a foolproof method that always finds a root
  - Bisection
  - Newton Secant
  - Inverse Quadratic Interpolation

Inverse Quadratic Interpolation (IQI)

Use the data points (f(x),x) (f(a),a) (f(b),b) and (f(c),c)
To define a quadratic polynomial

Evaluate at f(x)=0 to get where
The root x* is





Inverse Quadratic Interpolation (IQI) Code

```
k = 0; while abs(c-b) > eps*abs(c)
   x = polyinterp([f(a),f(b),f(c)],[a,b,c],0)
   a = b;
   b = c;
   c = x;
    k = k + 1;
 end
```

Problem - needs f(a) f(b) and f(c) to be distinct
So cannot always be used

In other words

$$x = a\frac{(x-f(b))(x-f(c))}{(f(a)-f(b))(f(a)-f(c))}$$

$$+b\frac{(x-f(a))(x-f(c))}{(f(b)-f(a))(f(b)-f(c))}$$

$$+c\frac{(x-f(a))(x-f(b))}{(f(c)-f(a))(f(c)-f(b))}$$

Method breaks down if any two of f(a), f(b) and f(c) are identical

Fzero Algorithm

Start with a and b so that f(a) and f(b) have opposite signs, perhaps using bisection .

- Use a secant step to give c between a and b.

- Repeat the following steps until $|b - a| < \varepsilon|b|$ or $f(b) = 0$.

- Arrange a, b, and c so that
  - f(a) and f(b) have opposite signs,
  - $|f(b)| \le |f(a)|$,
  - c is the previous value of b.

- If $c \ne a$, consider an IQI step.
- If $c = a$, consider a secant step.

- If the IQI or secant step is in the interval [a, b], take it.

- If the step is not in the interval, use bisection.

Matlab fzero function

You can use the `fzero` function to find the zero of a function of a single variable, which is denoted by `x`. One form of its syntax is

```
fzero('function', x0)
```

where `function` is a string containing the name of the function, and `x0` is a user-supplied guess for the zero.

The `fzero` function returns a value of `x` that is near `x0`.
It identifies only points where the function crosses the *x*-axis, not points where the function just touches the axis.

For example, `fzero('cos',2)` returns the value 1.5708.

Using `fzero` with User-Defined Functions

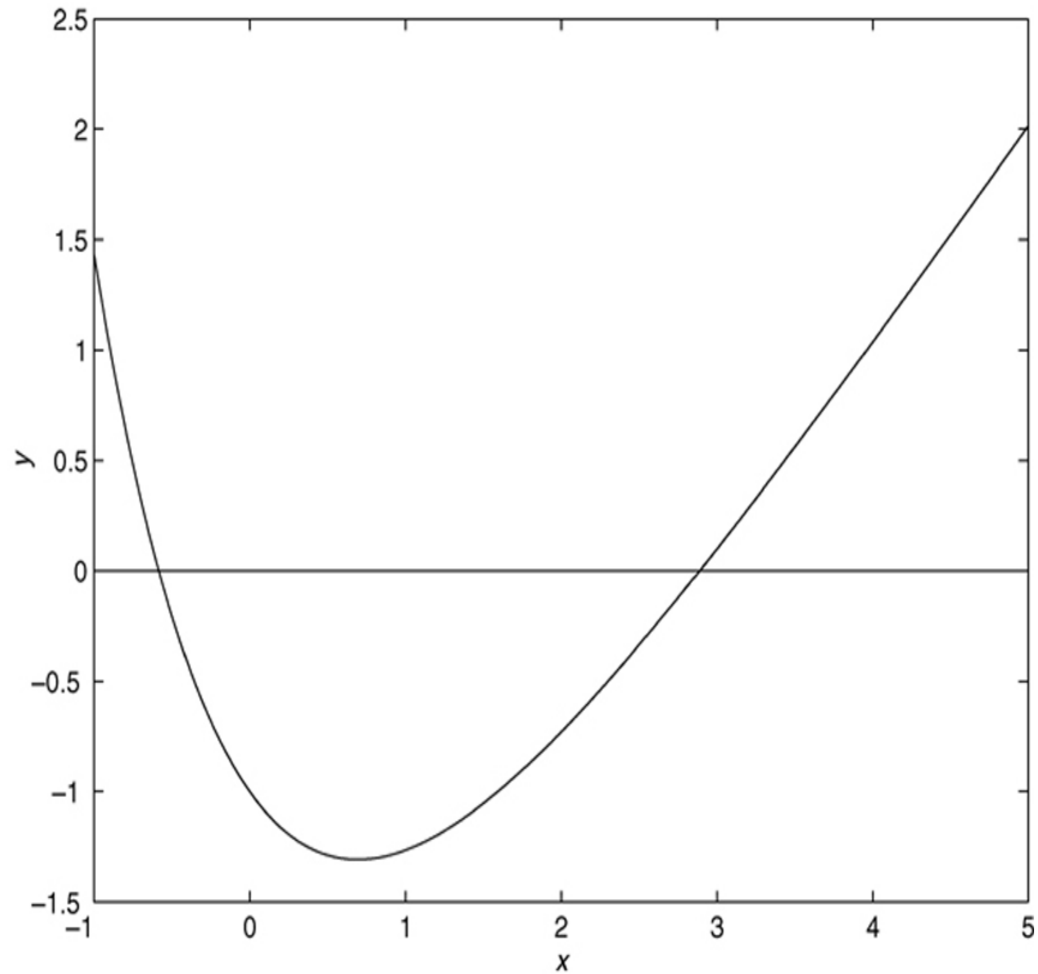
To use the `fzero` function to find the zeros of more complicated functions, it is more convenient to define the function in a function file.

For example, if $y = x + 2e^{-x} - 3$, define the following function file:

```
function y = f1(x)
y = x + 2*exp(-x) - 3;
```

Plot of the function $y = x + 2e^{-x} - 3$

There is a zero near x = -0.5 and one near x = 3.



Multi-Variable Newton-Raphson

Consider the case where $\mathbf{x}$ is an n-dimension vector, and $\mathbf{f}(\mathbf{x})$ is an n-dimension function

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{bmatrix}$$

Define the solution $\hat{\mathbf{x}}$ so $\mathbf{f}(\hat{\mathbf{x}}) = 0$ and

$$\Delta\mathbf{x} = \hat{\mathbf{x}} - \mathbf{x}$$

Instead of dividing by the derivative we have to solve a system of equations. This will not be covered any further here

Matlab Fsolve Routine Later version

- This routine incorporates many features to improve the robustness of Newton's method.

- One of the key ideas is that of a "**trust region**" in which the next point chosen is only used if the value of the function goes down. This is done by approximating the function f by a simpler function Q(x) which is a quadratic approximation. This provides a different, more robust and more complex search direction than the standard Newton step.
- Alternatives in fsolve are the **trust region dogleg** and the **Levenberg-Marquardt algorithm**
- See the fsolve documentation for more details https://www.mathworks.com/help/optim/ug/fsolve.html