Part 2
Chapter 6

Roots: Open Methods

Chapter Objectives

- Recognizing the difference between bracketing and open methods for root location.
- Knowing how to solve a roots problem with the Newton-Raphson method
- Knowing how to implement both the secant method.
- Knowing how to use MATLAB's `fzero`, `roots`, and `poly` functions.

Newton-Raphson Method – An open method

only need 1 point
? - bracketing needs 2

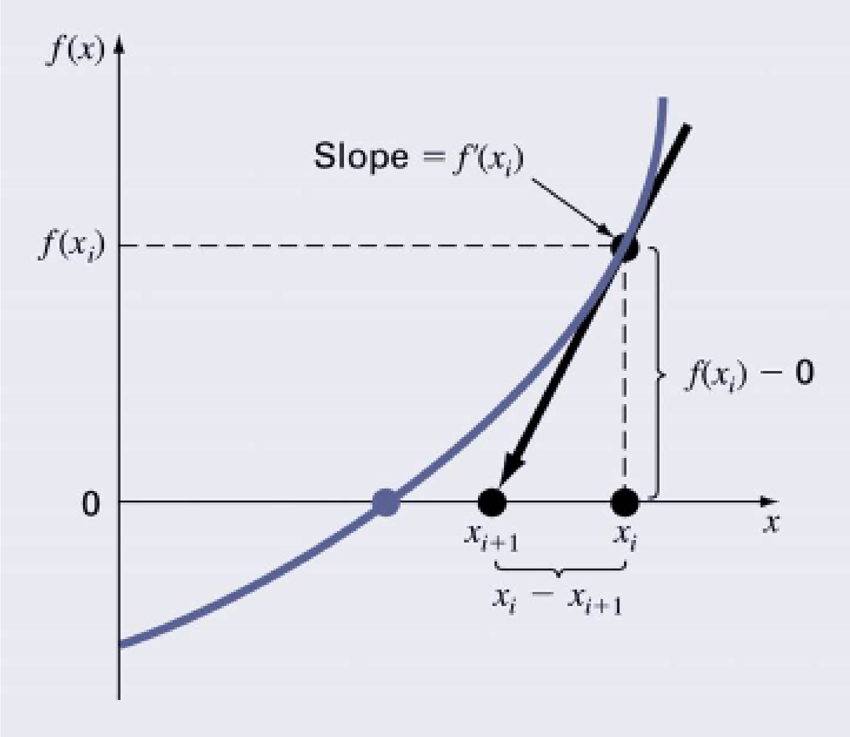
- *Open methods* differ from bracketing methods, in that open methods require only a single starting value

- Open methods may diverge as the computation progresses, but when they do converge, they usually do so much faster than bracketing methods.

Newton Raphson method

$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

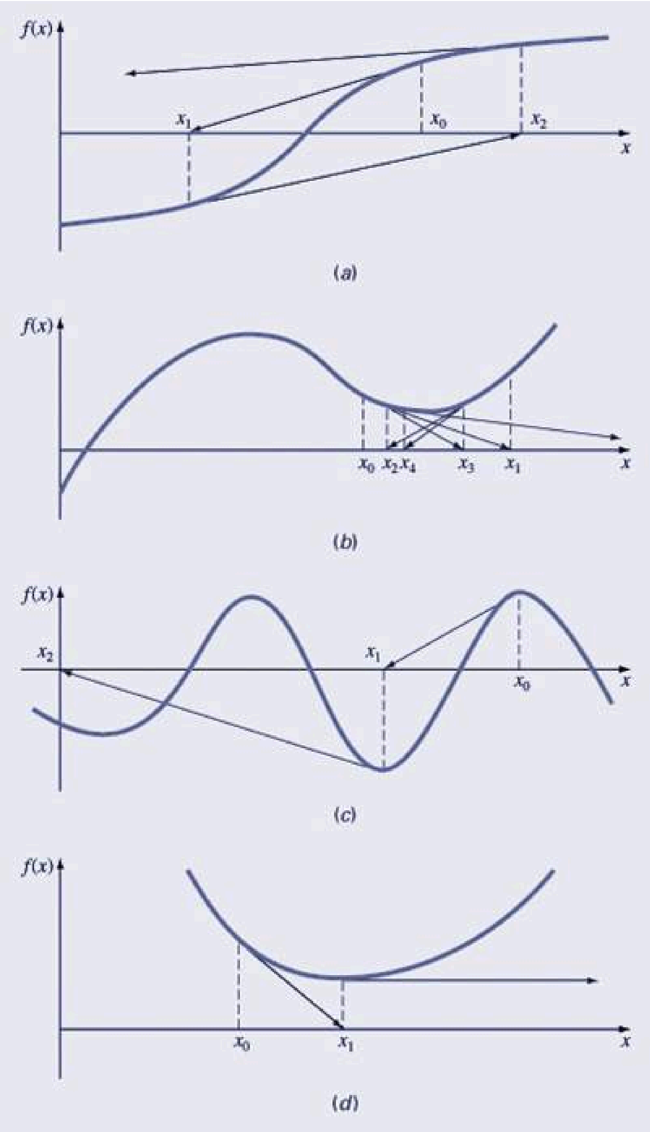
$$x_{i+1} = x_i - \frac{f(xi)}{f'(xi)}$$



Pros and Cons

Pro: The error of the i+1th iteration is roughly proportional to the square of the error of the ith iteration - this is called *quadratic convergence*

Con: Some functions show slow or poor convergence or divergence!



Secant Method (open method)

A potential problem in implementing the Newton-Raphson method is the evaluation of the derivative - there are certain functions whose derivatives may be difficult or inconvenient to evaluate.

For these cases, the derivative can be approximated as,

$$f'(x_i) \cong \frac{f(x_i + \delta x) - f(x_i)}{\delta x}$$

while (abs(f(xnew)) > 10^-4)

By using this approximation in the Newton-Raphson method we have

$$x_{new} = x_{old} - \frac{f(x_{old})\,\delta x}{f(x_{old} + \delta x) - f(x_{old})}$$

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

end

This method is called the Secant method

MATLAB's `fzero` Function

MATLAB's `fzero` provides the best qualities of both bracketing methods and open methods.

- Using an initial guess:
  `x = fzero(function, x0)`
  `[x, fx] = fzero(function, x0)`
  - *`function`* is a function handle to the function being evaluated
  - *`x0`* is the initial guess
  - *`x`* is the location of the root
  - *`fx`* is the function evaluated at that root
- Using an initial bracket:
  `x = fzero(function, [x0 x1])`
  `[x, fx] = fzero(function, [x0 x1])`
  - As above, except *`x0`* and *`x1`* are guesses that *must* bracket a sign change

Polynomials, 1

MATLAB has a built in program called `roots` to determine all the roots of a polynomial - including imaginary and complex ones.

```
x = roots(c)
```

- `x` is a column vector containing the roots
- `c` is a row vector containing the polynomial coefficients

Example:

- Find the roots of
  $f(x) = x^5 - 3.5x^4 + 2.75x^3 + 2.125x^2 - 3.875x + 1.25$
- `x = roots([1 -3.5 2.75 2.125 -3.875 1.25])`

Polynomials, 2

MATLAB's `poly` function can be used to determine polynomial coefficients if roots are given:

- `b = poly([0.5 -1])`
  - Finds $f(x)$ where $f(x) = 0$ for $x=0.5$ and $x=-1$
  - MATLAB reports `b = [1.000 0.5000 -0.5000]`
  - This corresponds to $f(x)=x^2+0.5x-0.5$

MATLAB's `polyval` function can evaluate a polynomial at one or more points:

```
>> a = [1 -3.5 2.75 2.125 -3.875 1.25];
```

  - If used as coefficients of a polynomial, this corresponds to $f(x)=x^5-3.5x^4+2.75x^3+2.125x^2-3.875x+1.25$

- `polyval(a, 1)`
  - This calculates $f(1)$, which MATLAB reports as -0.2500

`Poly` and `roots` commands

Use the `Poly` and `roots` commands to determine the roots of

$$f(x) = x^5 - 16.05\,x^4 + 88.75x^3 - 192.0375x^2 + 116.35x + 31.6875$$