Part 2
Chapter 6

Roots: Open Methods
Chapter Objectives

• Recognizing the difference between bracketing and open methods for root location.
• Understanding the fixed-point iteration method and how you can evaluate its convergence characteristics.
• Knowing how to solve a roots problem with the Newton-Raphson method and appreciating the concept of quadratic convergence.
• Knowing how to implement both the secant and the modified secant methods.
• Knowing how to use MATLAB’s `fzero` function to estimate roots.
• Learning how to manipulate and determine the roots of polynomials with MATLAB.
Open Methods

• *Open methods* differ from bracketing methods, in that open methods require only a single starting value or two starting values that do not necessarily bracket a root.

• Open methods may diverge as the computation progresses, but when they do converge, they usually do so much faster than bracketing methods.
Graphical Comparison of Methods

a) Bracketing method
b) Diverging open method
c) Converging open method - note speed!
Simple Fixed-Point Iteration

• Rearrange the function \( f(x)=0 \) so that \( x \) is on the left-hand side of the equation: \( x=g(x) \)

• Use the new function \( g \) to predict a new value of \( x \) - that is, \( x_{i+1}=g(x_i) \)

• The approximate error is given by:

\[
\varepsilon_a = \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| \times 100 \%
\]
Examples

• Solve \( f(x) = e^{-x} - x \)

• Re-write as \( x = g(x) \) by isolating \( x \) (example: \( x = e^{-x} \))

• Start with an initial guess (here, 0)

| \( i \) | \( x_i \) | \( |\varepsilon_a| \) % | \( |\varepsilon_t| \) % | \( |\varepsilon_t|/|\varepsilon_t|_{i-1} \) |
|-----|--------|----------------|----------------|------------------|
| 0   | 0.0000 | 100.000        |                 |                  |
| 1   | 1.0000 | 100.000        | 76.322          | 0.763            |
| 2   | 0.3679 | 171.828        | 35.135          | 0.460            |
| 3   | 0.6922 | 46.854         | 22.050          | 0.628            |
| 4   | 0.5005 | 38.309         | 11.755          | 0.533            |

• Continue until some tolerance is reached. True value = 0.56714329
Convergence

- Convergence of the simple fixed-point iteration method requires that the derivative of $g(x)$ near the root has a magnitude less than 1.
  a) Convergent, $0 \leq g' < 1$
  b) Convergent, $-1 < g' \leq 0$
  c) Divergent, $g' > 1$
  d) Divergent, $g' < -1$
Newton-Raphson Method

- Based on forming the tangent line to the $f(x)$ curve at some guess $x$, then following the tangent line to where it crosses the $x$-axis.

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

\[
x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}
\]
Pros and Cons

- **Pro:** The error of the $i+1^{\text{th}}$ iteration is roughly proportional to the square of the error of the $i^{\text{th}}$ iteration - this is called *quadratic convergence*.

- **Con:** Some functions show slow or poor convergence.
Secant Methods

• 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 by a backward finite divided difference:

\[ f'(x_i) \approx \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i} \]
Secant Methods (cont)

- Substitution of this approximation for the derivative to the Newton-Raphson method equation gives:

\[ x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)} \]

- Note - this method requires two initial estimates of \( x \) but does not require an analytical expression of the derivative.
Modified Secant Method

• Rather than use two arbitrary values to estimate the derivative, alternate method is to use a fractional perturbation of the independent variable

\[ f'(x_i) \approx \frac{f(x_i + \delta x_i) - f(x_i)}{\delta x_i} \]

\[ x_{i+1} = x_i - \frac{\delta x_i f(x_i)}{f(x_i + \delta x_i) - f(x_i)} \]
Example of modified secant method

• Use modified secant method to determine the mass of the bungee jumper with a drag coefficient of 0.25 kg/m to have a velocity of 36 m/s after 4 s of free fall. Use initial guess of 50 kg and a $10^{-6}$ for perturbation fraction.

$$v(t) = \sqrt{\frac{g m}{c_d}} \tanh \left( \sqrt{\frac{g c_d}{m}} t \right)$$

$$f(m) = 0 = \sqrt{\frac{g m}{c_d}} \tanh \left( \sqrt{\frac{g c_d}{m}} t \right) - v(t)$$

• First iteration

$x_0 = 50$ \hspace{0.5cm} $f(x_0) = -4.57938708$

$x_0 + \delta x_0 = 50.00005$ \hspace{0.5cm} $f(x_0) = -4.579381118$

$$x_1 = 50 = \frac{10^{-6} (50)(-4.57938708)}{-4.579381118 - (-4.57938708)}$$

$$= 88.39931 (|\varepsilon_t| = 38.1\% ; |\varepsilon_a| = 43.4\%)$$
• Second iteration

\[ x_0 = 88.39931 \quad f(x_0) = -1.69220771 \]

\[ x_0 + \delta x_0 = 88.39940 \quad f(x_0) = -1.692203516 \]

\[ x_2 = 88.39931 - \frac{10^{-6} (88.39931)(-1.69220771)}{-1.692203516 - (-1.69220771)} \]

\[ = 124.08970 (|\varepsilon_t| = 13.1\% ; |\varepsilon_a| = 28.76\%) \]
| \(i\) | \(x_i\)      | \(|\varepsilon_t|\)% | \(|\varepsilon_a|\)% |
|------|--------------|----------------------|----------------------|
| 0    | 50.0000      | 64.971               |                      |
| 1    | 88.3993      | 38.069               | 43.438               |
| 2    | 124.0897     | 13.064               | 28.762               |
| 3    | 140.5417     | 1.538                | 11.706               |
| 4    | 142.7072     | 0.021                | 1.517                |
| 5    | 142.7376     | 4.1 \times 10^{-6}  | 0.021                |
| 6    | 142.7376     | 3.4 \times 10^{-12} | 4.1 \times 10^{-6}  |
MATLAB’s \texttt{fzero} Function

MATLAB’s \texttt{fzero} provides the best qualities of both bracketing methods and open methods.

- Using an initial guess:
  \[
  x = \texttt{fzero}(function, \ x0)
  \]
  \[
  [x, \ fx] = \texttt{fzero}(function, \ x0)
  \]
  - \textit{function} is a function handle to the function being evaluated
  - \textit{x0} is the initial guess
  - \textit{x} is the location of the root
  - \textit{fx} is the function evaluated at that root

- Using an initial bracket:
  \[
  x = \texttt{fzero}(function, \ [x0 \ x1])
  \]
  \[
  [x, \ fx] = \texttt{fzero}(function, \ [x0 \ x1])
  \]
  - As above, except \textit{x0} and \textit{x1} are guesses that \textit{must} bracket a sign change
Options may be passed to fzero as a third input argument - the options are a data structure created by the `optimset` command.

```matlab
options = optimset('par1', val1, 'par2', val2,...)
```

- `par_n` is the name of the parameter to be set.
- `val_n` is the value to which to set that parameter.

The parameters commonly used with `fzero` are:

- **display**: when set to `iter` displays a detailed record of all the iterations.
- **tolx**: A positive scalar that sets a termination tolerance on `x`. 
**fzero Example**

- \( \text{options} = \text{optimset}('\text{display}', '\text{iter}'); \)
  - Sets options to display each iteration of root finding process
- \([x, \, fx] = \text{fzero}(@(x) \, x^{10} - 1, \, 0.5, \, \text{options}) \)
  - Uses fzero to find roots of \( f(x) = x^{10} - 1 \) starting with an initial guess of \( x=0.5 \).
- MATLAB reports \( x=1, \, fx=0 \) after 35 function counts
Polynomials

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

• \( x = \text{roots}(c) \)
  - \( x \) is a column vector containing the roots
  - \( c \) is a row vector containing the polynomial coefficients

• Example:
  - Find the roots of
    \( f(x) = x^5 - 3.5x^4 + 2.75x^3 + 2.125x^2 - 3.875x + 1.25 \)
  - \( x = \text{roots}([1 -3.5 2.75 2.125 -3.875 1.25]) \)
Polynomials (cont)

- MATLAB’s `poly` function can be used to determine polynomial coefficients if roots are given:
  \[ b = \text{poly}(\{0.5, -1\}) \]
  - Finds \( f(x) \) where \( f(x) = 0 \) for \( x=0.5 \) and \( x=-1 \)
  - MATLAB reports \( b = [1.0000 \ 0.5000 \ -0.5000] \)
  - 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 \( f(x)=x^5-3.5x^4+2.75x^3+2.125x^2-3.875x+1.25 \)
  - \( \text{polyval}(a, 1) \)
  - This calculates \( f(1) \), which MATLAB reports as -0.2500