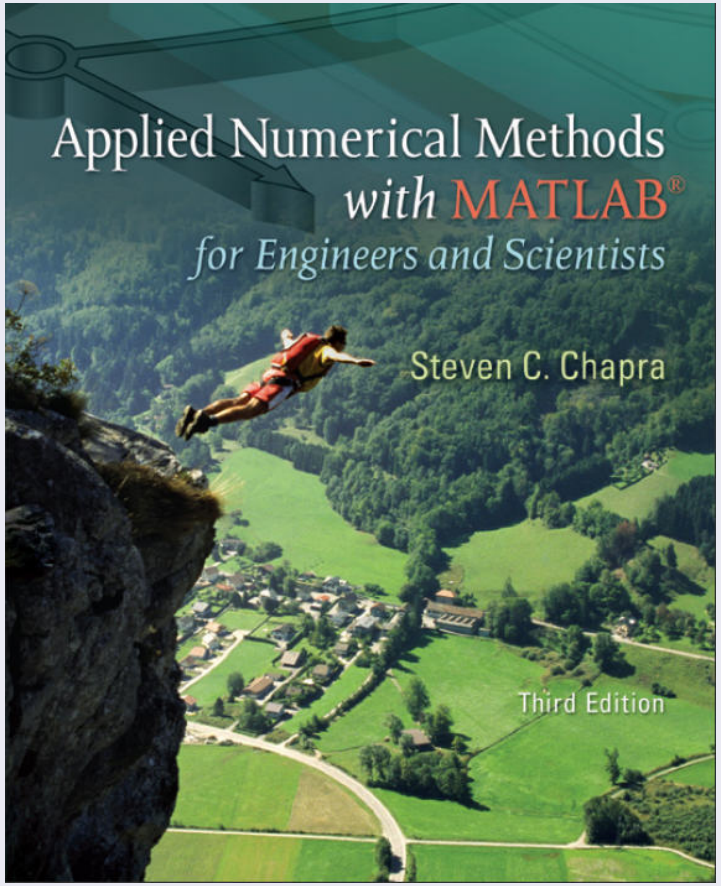


Part 2
Chapter 5

Roots: Bracketing Methods

PowerPoints organized by Dr. Michael R. Gustafson II, Duke University

Chapter Objectives

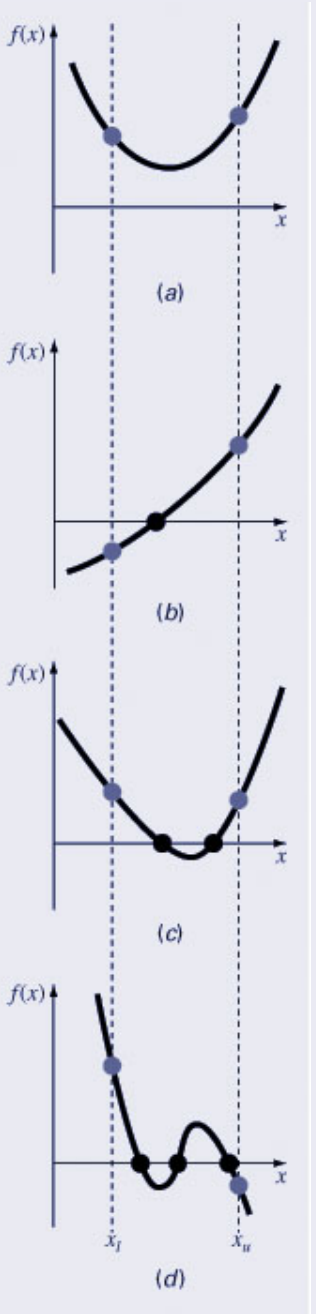
- Understanding what roots problems are and where they occur in engineering and science.
- Knowing how to determine a root graphically.
- Understanding the incremental search method and its shortcomings.
- Knowing how to solve a roots problem with the bisection method.
- Knowing how to estimate the error of bisection and why it differs from error estimates for other types of root location algorithms.
- Understanding false position and how it differs from bisection.

Roots

- “Roots” problems occur when some function *f* can be written in terms of one or more dependent variables *x*, where the solutions to *f(x)=0* yields the solution to the problem.
- These problems often occur when a design problem presents an implicit equation for a required parameter.

Graphical Methods

- A simple method for obtaining the estimate of the root of the equation *f(x)*=0 is to make a plot of the function and observe where it crosses the *x*-axis.
- Graphing the function can also indicate where roots may be and where some root-finding methods may fail:
  a) Same sign, no roots
  b) Different sign, one root
  c) Same sign, two roots
  d) Different sign, three roots



Bracketing Methods

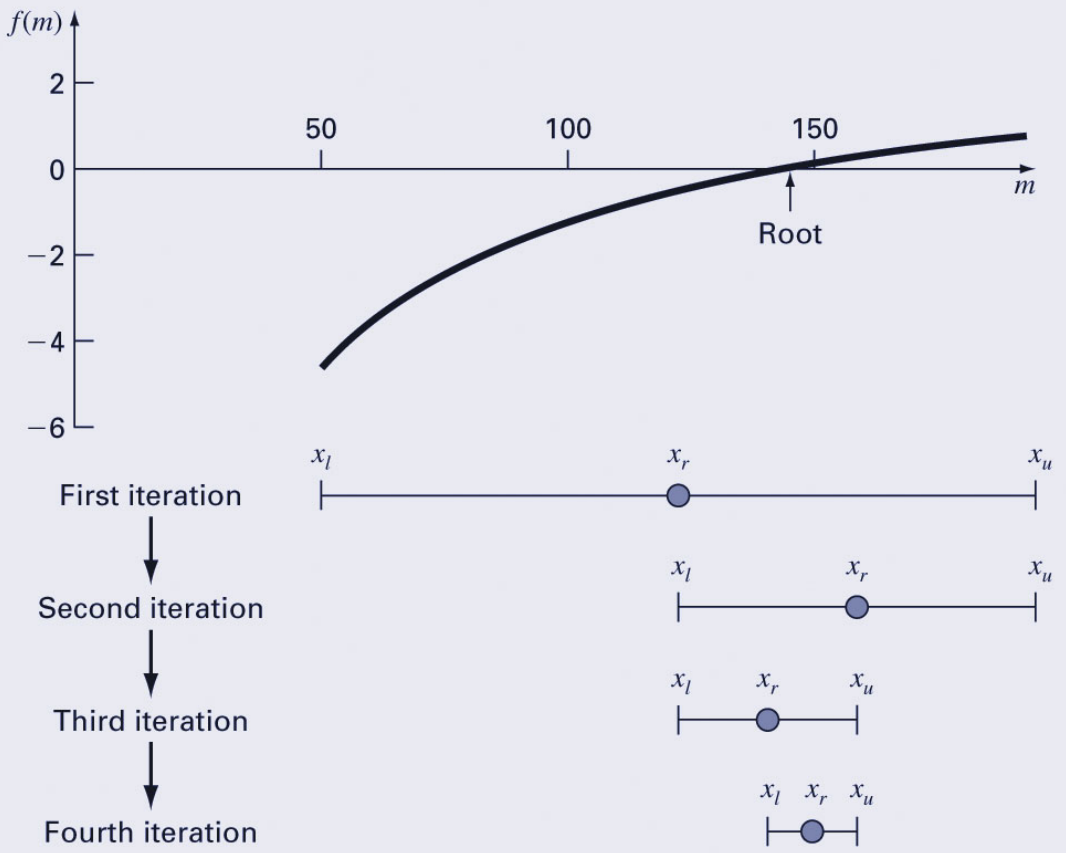
- *Bracketing methods* are based on making two initial guesses that “bracket” the root - that is, are on either side of the root.
- Brackets are formed by finding two guesses x_l and x_u where the sign of the function changes; that is, where $f(x_l)\, f(x_u) < 0$
- The *incremental search* method tests the value of the function at evenly spaced intervals and finds brackets by identifying function sign changes between neighboring points.

Incremental Search Hazards

- If the spacing between the points of an incremental search are too far apart, brackets may be missed due to capturing an even number of roots within two points.
- Incremental searches cannot find brackets containing even-multiplicity roots regardless of spacing.

Bisection

- The *bisection method* is a variation of the incremental search method in which the interval is always divided in half.
- If a function changes sign over an interval, the function value at the midpoint is evaluated.
- The location of the root is then determined as lying within the subinterval where the sign change occurs.
- The absolute error is reduced by a factor of 2 for each iteration.



Programming Bisection

```
function [root,ea,iter]=bisect(func,xl,xu,es,maxit,varargin)
% bisect: root location zeroes
%   [root,ea,iter]=bisect(func,xl,xu,es,maxit,p1,p2,...):
%      uses bisection method to find the root of func
% input:
%   func = function handle
%   xl, xu = lower and upper guesses
%   es = desired relative error (default = 0.0001%)
%   maxit = maximum allowable iterations (default = 50)
%   p1,p2,... = additional parameters used by func
% output:
%   root = real root
%   ea = approximate relative error (%)
%   iter = number of iterations

if nargin<3,error('at least 3 input arguments required'),end
test = func(xl,varargin{:})*func(xu,varargin{:});
if test>0,error('no sign change'),end
if nargin<4||isempty(es), es=0.0001;end
if nargin<5||isempty(maxit), maxit=50;end
iter = 0; xr = xl;
while (1)
  xrold = xr;
  xr = (xl + xu)/2;
  iter = iter + 1;
  if xr ~= 0,ea = abs((xr - xrold)/xr) * 100;end
  test = func(xl,varargin{:})*func(xr,varargin{:});
  if test < 0
    xu = xr;
  elseif test > 0
    xl = xr;
  else
    ea = 0;
  end
  if ea <= es || iter >= maxit,break,end
end
root = xr;
```

Bisection Error

- The absolute error of the bisection method is solely dependent on the absolute error at the start of the process (the space between the two guesses) and the number of iterations:

$$E_a^n = \frac{\Delta x^0}{2^n}$$

- The required number of iterations to obtain a particular absolute error can be calculated based on the initial guesses:

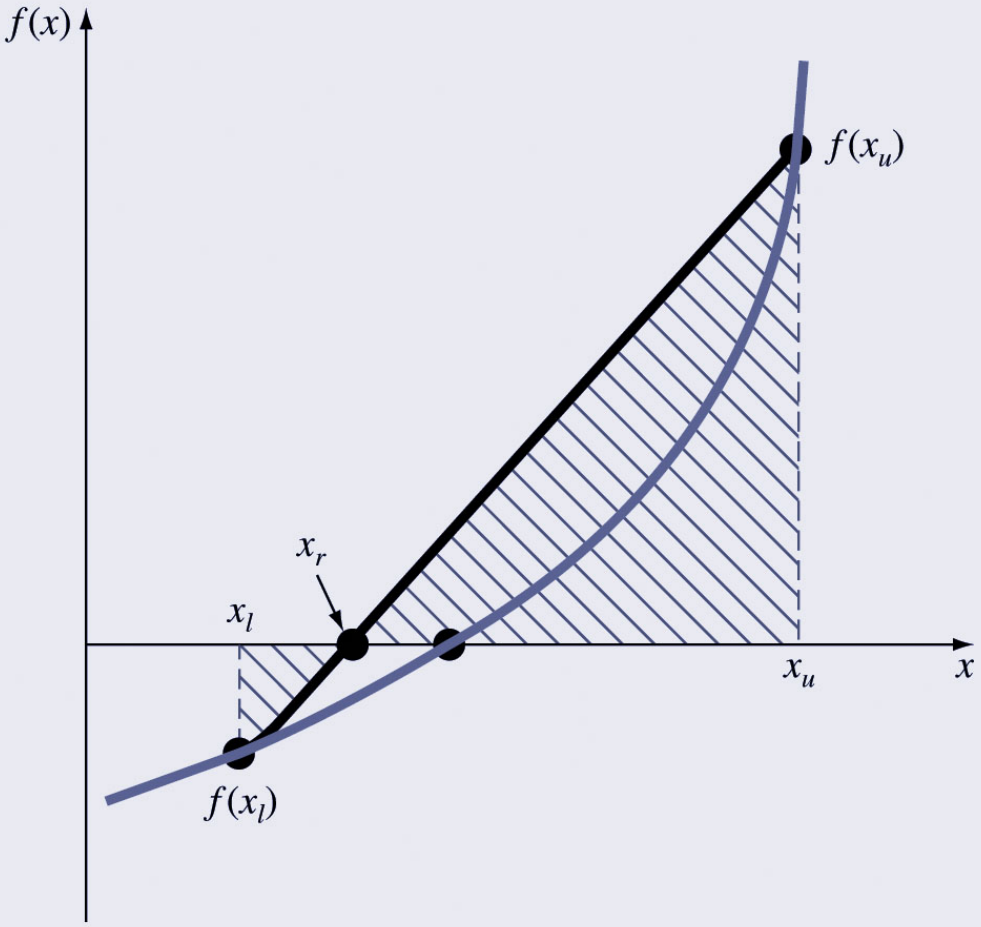
$$n = \log_2\left(\frac{\Delta x^0}{E_{a,d}}\right)$$

False Position

- The *false position* method is another bracketing method.
- It determines the next guess not by splitting the bracket in half but by connecting the endpoints with a straight line and determining the location of the intercept of the straight line (x_r).
- The value of x_r then replaces whichever of the two initial guesses yields a function value with the same sign as $f(x_r)$.

False Position Illustration

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$$



Bisection vs. False Position

- Bisection does not take into account the shape of the function; this can be good or bad depending on the function!
- Bad: $f(x) = x^{10} - 1$