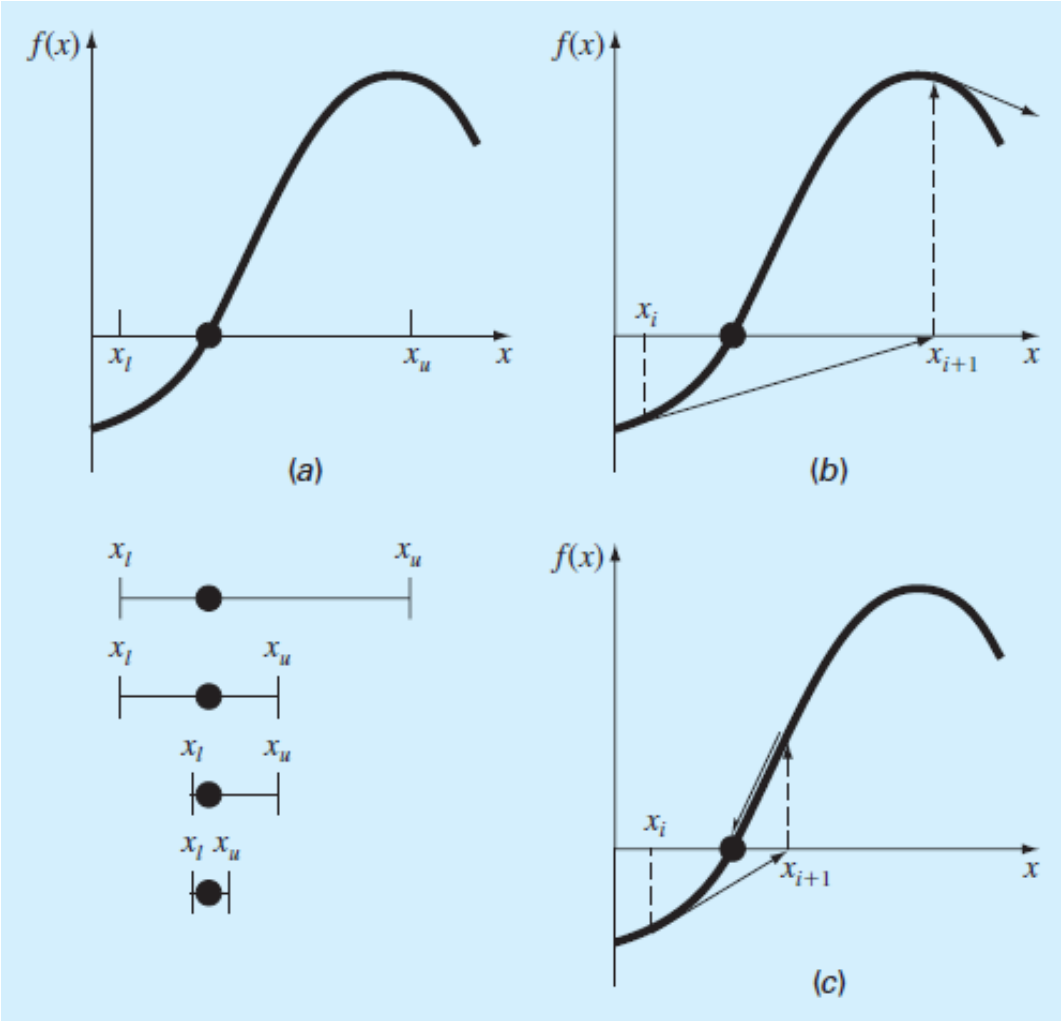
# Roots by Open Method

Hsiao-Lung Chan
Dept Electrical Engineering
Chang Gung University, Taiwan
chanhl@mail.cgu.edu.tw

# Graphic comparison of root-finding methods

Bracket method

Open method



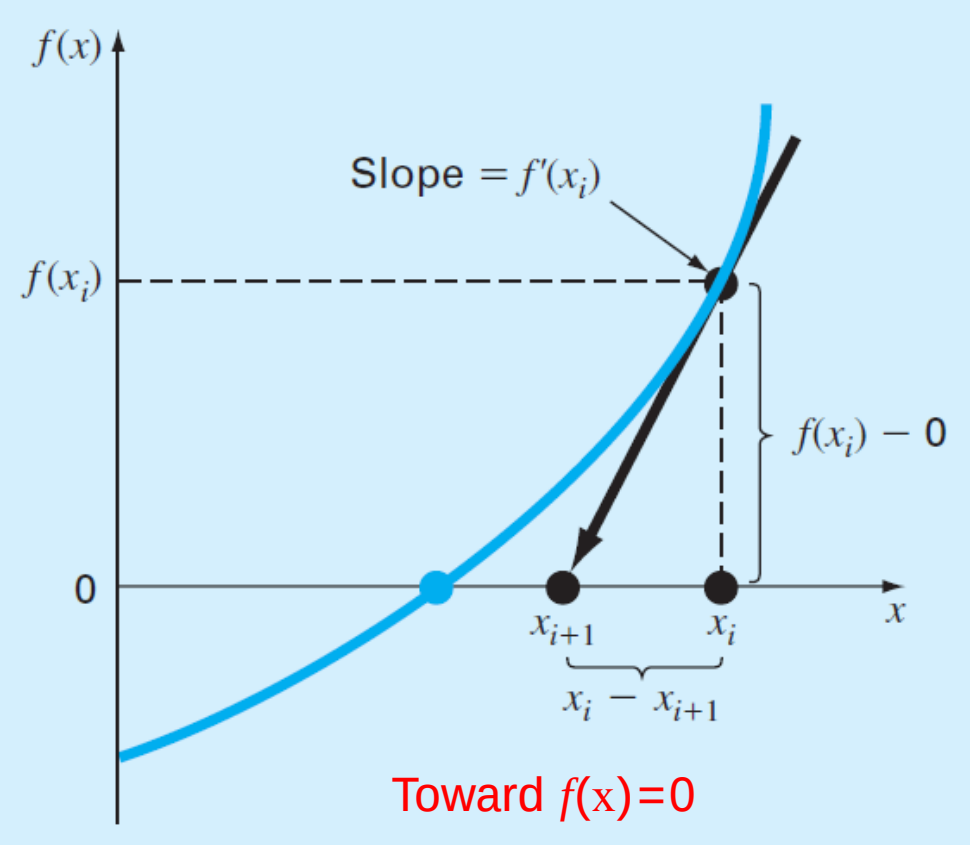

Diverging

Converging

# Newton-Raphson method

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

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




Extrapolated down to the *x* axis

# Example: Use the Newton-Raphson method to estimate the root of $f(x) = e^{-x} - x$

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

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

Starting with an initial guess of $x_0 = 0$

| $i$ | $x_i$ | $\lvert\varepsilon_t\rvert$, % |
|---|---|---|
| 0 | 0 | 100 |
| 1 | 0.500000000 | 11.8 |
| 2 | 0.566311003 | 0.147 |
| 3 | 0.567143165 | 0.0000220 |
| 4 | 0.567143290 | $<10^{-8}$ |

$$|\varepsilon_t| = \left|\frac{x_{root} - x_i}{x_{root}}\right| \times 100\%$$

# M-function for Newton-Raphson method

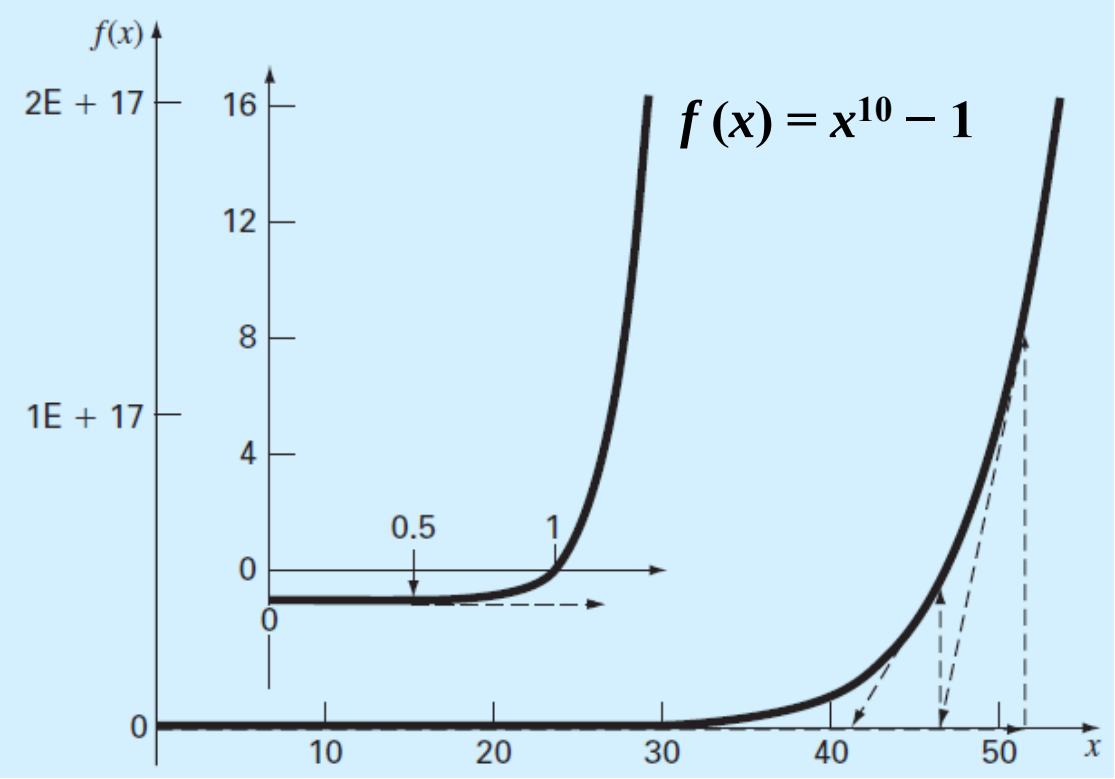
```
function [root,fx,ea,iter]=newtraph(func,dfunc,xr,es,maxit)
  if nargin<4|isempty(es), es=0.0001; end
  if nargin<5|isempty(maxit), maxit=50; end
  iter = 0;
  while (1)
    xrold = xr;
    xr = xr - func(xr)/dfunc(xr);
    fx=func(xr);
    iter = iter + 1;
    if xr ~= 0, ea = abs((xr - xrold)/xr) * 100; end
    if ea <= es | iter >= maxit, break, end
  end
  root = xr;
```

**Calling code:**

```
newtraph(@(x) exp(-x)-x, @(x) –exp(-x)-1, 0,0.0001,2)
```

# A slowly converging function with Newton-Raphson

$$x_{i+1} = x_i - \frac{x_i^{10} - 1}{10x_i^9}$$




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

| $i$ | $x_i$ | $\lvert\varepsilon_a\rvert$, % |
|---|---|---|
| 0 | 0.5 | |
| 1 | 51.65 | 99.032 |
| 2 | 46.485 | 11.111 |
| 3 | 41.8365 | 11.111 |
| 4 | 37.65285 | 11.111 |
| • | | |
| • | | |
| • | | |
| 40 | 1.002316 | 2.130 |
| 41 | 1.000024 | 0.229 |
| 42 | 1 | 0.002 |

# Newton-Raphson bungee jumper problem:

## A drag coefficient of 0.25 kg/m to have a velocity of 36 m/s after 4 s of free fall

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

$$\frac{df(m)}{dm} = \frac{1}{2}\sqrt{\frac{g}{mc_d}}\tanh\left(\sqrt{\frac{gc_d}{m}}t\right) - \frac{g}{2m}t\,\mathrm{sech}^2\left(\sqrt{\frac{gc_d}{m}}t\right)$$

```
y = @(m) sqrt(9.81*m/0.25)*  ...
      tanh(sqrt(9.81*0.25/m)*4)-36;
dy = @(m) 1/2*sqrt(9.81/(m*0.25))* ...
       tanh((9.81*0.25/m)^(1/2)*4) ...
       -9.81/(2*m)*sech(sqrt(9.81*0.25/m)*4)^2;
newtraph(y,dy,140,0.00001)
```

# Secant method

- Potential problem in Newton-Raphson: the derivative may be difficult or inconvenient to evaluate.
- Secant method: the derivative is approximated by a backward finite divided difference:

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

Root finding by

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

# Modified Secant method

- A fractional perturbation of $x$ to estimate $f(x)$

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

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

# MATLAB's `fzero` function

- *x* = fzero(*function*, *x0*); % *x0*, initial guess
  [*x*, *fx*] = fzero(*function*, *x0*);
- *options* = optimset('$par_1$', $val_1$, '$par_2$', $val_2$,…)
  *x* = fzero(*function*, *x0, options*);
- Example

```
options = optimset('display', 'iter');
% display each iteration of root finding process
options = optimset('tolx', 1E-6);
% a termination tolerance on x
```

# Polynomial

$f(x)=x^5 - 3.5x^4 + 2.75x^3 + 2.125x^2 - 3.875x + 1.25$

- Coefficient vector  a = [1 -3.5 2.75 2.125 -3.875 1.25];
- Roots  x=roots(a);
- Evaluation  r=polyval(a,1);  % $f(1)$

# Reference


- Steven C. Chapra "Applied Numerical Methods with MATLA B", 3rd ed., McGraw Hill, 2012.