Note on Root Finding and Nonlinear Sets of Equations

References: Numerical Recipes https://numerical.recipes/

Bracketing and Bisection

A root is bracketed in the interval (a, b) if

$$f(a) \cdot f(b) < 0$$

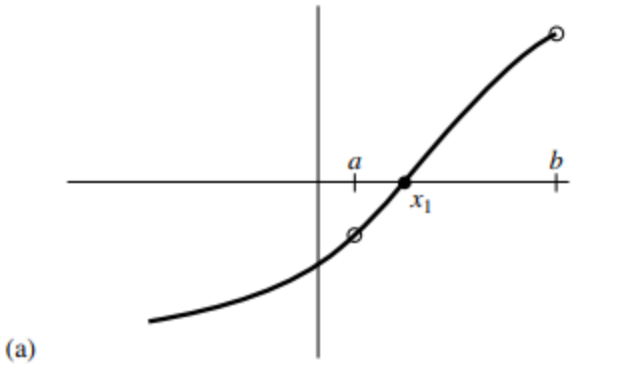
If $f(x)$ is continuous, at least one root must lie in that interval (the intermediate value theorem).

If $f(x)$ is discontinuous, but bounded, then instead of a root there might be a step discontinuity that crosses zero.

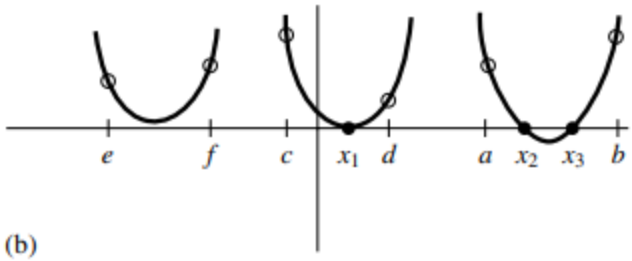
eg:

$$f(x) = \frac{1}{x - c}$$

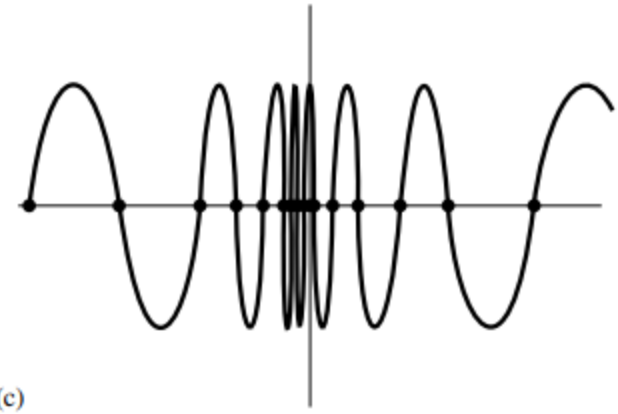
some root-finding algorithms like bisection will readily converge to c.



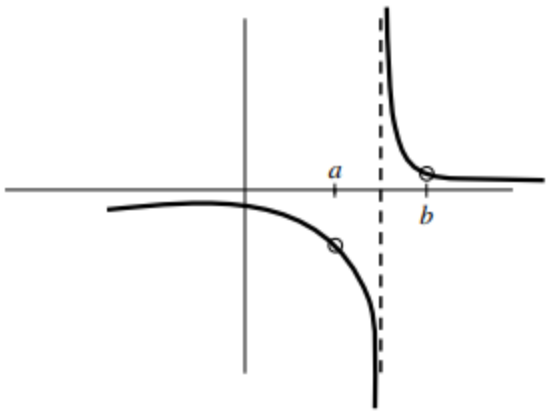
(a)



(b)



(c)



(d)

> Go downhill, taking steps of increasing size, until your function starts back uphill.

Bisection Method

Evaluate the function at the interval's midpoint and examine its sign.

Use the midpoint to replace whichever limit has the same sign.

After each iteration the bounds containing the root decrease by a factor of two.

$$\epsilon_{n+1} = \frac{\epsilon_n}{2}$$

eg: if $f(a) \cdot f(b) < 0$,

$$\epsilon_0 = b - a$$
$$\epsilon_1 = \frac{\epsilon_0}{2}$$

evaluate

$$f(a + \frac{\epsilon_0}{2})$$
$$\dots$$

tolerance:

$$n = log_2 \frac{\epsilon_0}{\epsilon}$$

ϵ_0 : the size of the initially bracketing interval

ϵ : the desired ending tolerance

When a method converges as a factor (less than 1) times the previous uncertainty to the first power (as is the case for bisection), it is said to converge linearly.

Methods that converge as a higher power,

$$\epsilon_{n+1} = constant \times (\epsilon_n)^m \quad m > 1$$

are said to converge superlinearly.

```
%Simple Bisection Method of Root Finding
xp = 1;
xn = 0.4;
xmp = (xn + xp) / 2;
ymp = input_function(xmp);
for iteration = 1:10
    if ymp < 0
        xn = xmp;
    else
        xp = xmp;
    end
    xmp = (xn + xp) / 2;
    ymp = input_function(xmp);
end
xvals = linspace(0.1,0.5);
yvals = 0.5 - sin(xvals) - xvals.^2;
plot(xvals,yvals)
hold on
plot([0.1 0.5],[0 0])
plot(xmp,ymp,"r.","MarkerSize",30)
hold off
grid on
```

Secant method, False Position method, and Ridders'method

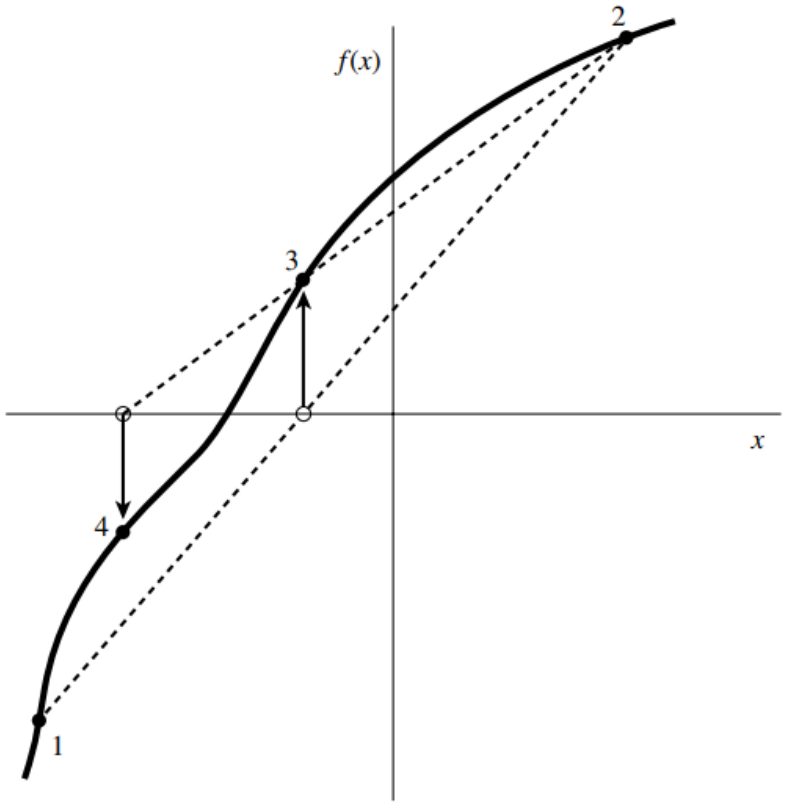
For functions that are smooth near a root

Secant method:

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

Convergence:

$$1.|x_{n+1} - x_n| < \epsilon?$$
$$or$$
$$2.f(x_{n+1}) \to 0?$$

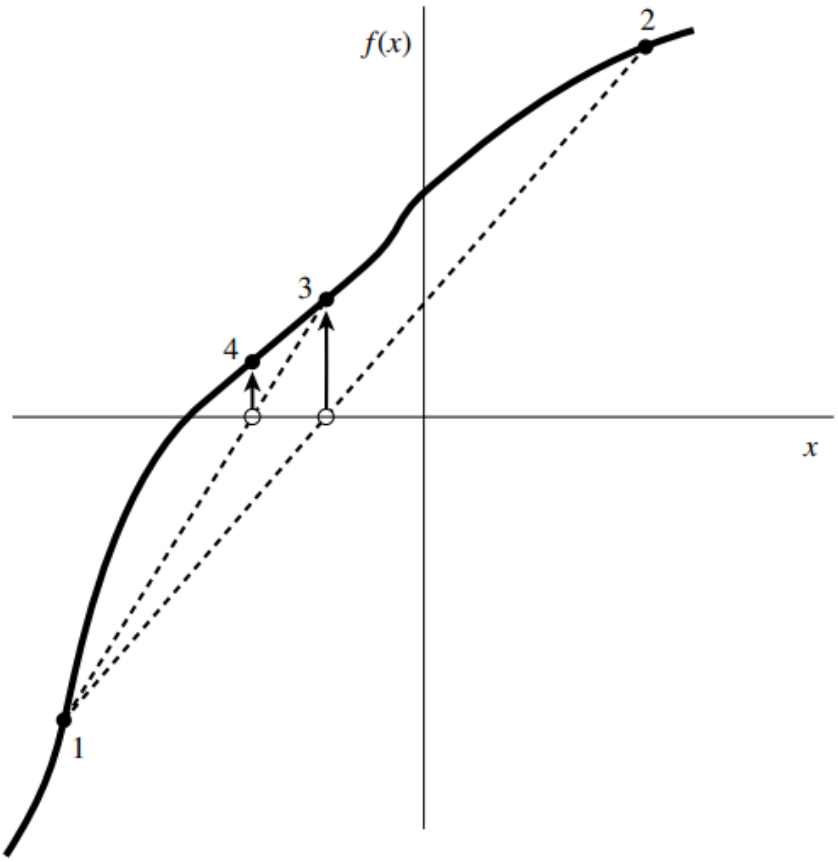


False position method:

In the interval (x_1, x_2), $f(x_1) \cdot f(x_2) < 0$

$$x_3 = x_1 - \frac{f(x_1) \cdot (x_2 - x_1)}{f(x_2) - f(x_1)}$$

$$if\ f(x_3) = 0 \to x_3 \approx\ root$$
$$if\ f(x_3) \cdot f(x_1) > 0 \to (x_3, x_2)$$
$$f(x_3) \cdot f(x_2) > 0 \to (x_1, x_3)$$



Ridder's method

When a root is bracketed between x_1 and x_2, Ridder's method first evaluates the function at the midpoint

$$x_3 = \frac{x_1 + x_2}{2}$$

It then factors out that unique exponential function that turns the residual function into a straight line.:

$$f(x_1) - 2f(x_3)e^Q + f(x_2)e^{2Q} = 0$$
$$e^Q = \frac{f(x_3) + sgn[f(x_2)]\sqrt{f(x_3)^2 - f(x_1)f(x_2)}}{f(x_2)}$$

$$x_4 = x_3 + (x_3 - x_1)\frac{sgn[f(x_1) - f(x_2)]f(x_3)}{\sqrt{f(x_3)^2 - f(x_1)f(x_2)}}$$

$$sgn(x) = \begin{cases} -1, \ x < 0 \\ 0, \ x = 0 \\ 1, \ x > 0 \end{cases}$$

Newton-Raphson method using derivative

It requires the evaluation of both the function $f(x)$ and the derivative $f'(x)$ at arbitrary x

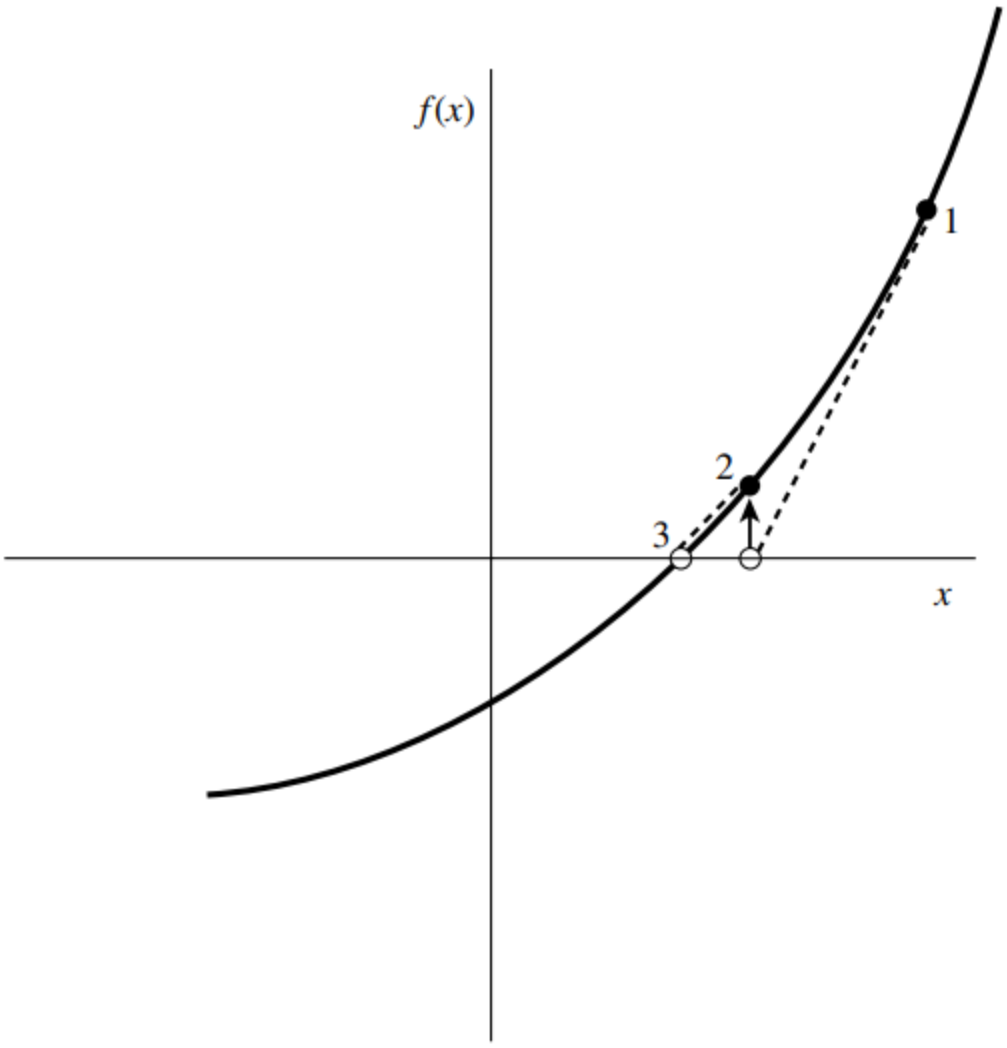
$$f(x + \delta) \approx f(x) + f'(x)\delta + \frac{f''(x)}{2}\delta^2 + \cdots$$

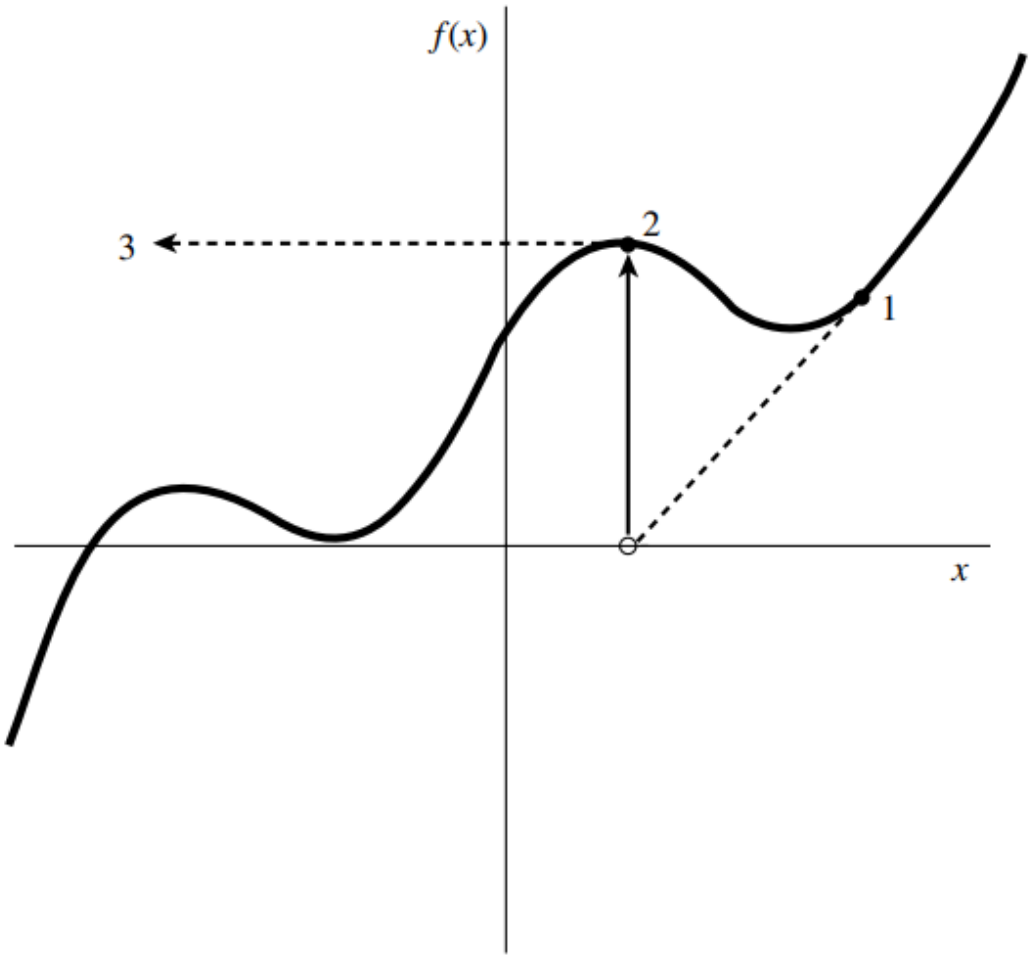
for small enough δ and well-behaved functions :

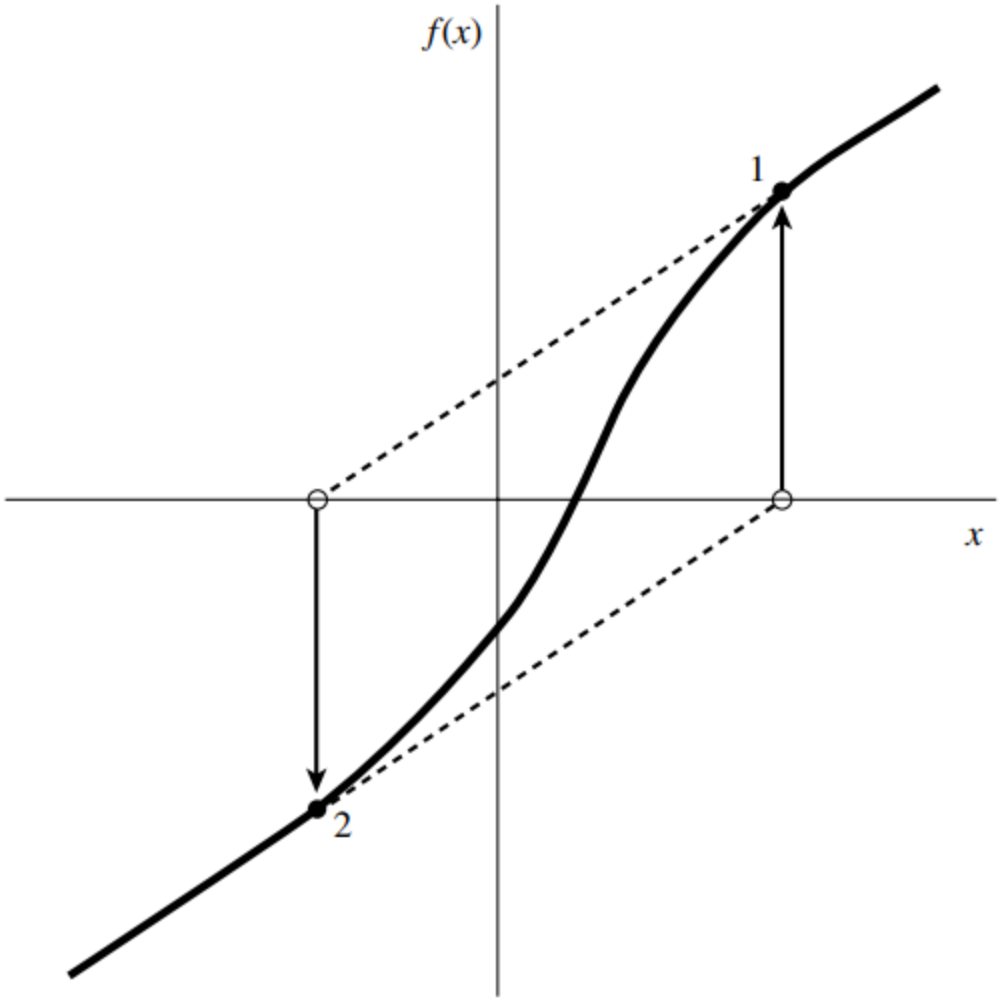
$$f(x + \delta) = 0$$

$$\delta = -\frac{f(x)}{f'(x)}$$

The Newton-Raphson formula consists geometrically
of extending the tangent line at a current point
x_i until it crosses zero, then setting the next guess x_{i+1} to the abscissa of that zero crossing:


f(x)
1
2
3
x


f(x)
3
2
1
x



Within a small distance ϵ of x :

$$f(x+\epsilon) = f(x) + \epsilon f'(x) + \epsilon^2 \frac{f''(x)}{2} + \cdots$$
$$f'(x+\epsilon) = f'(x) + \epsilon f''(x) + \cdots$$

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

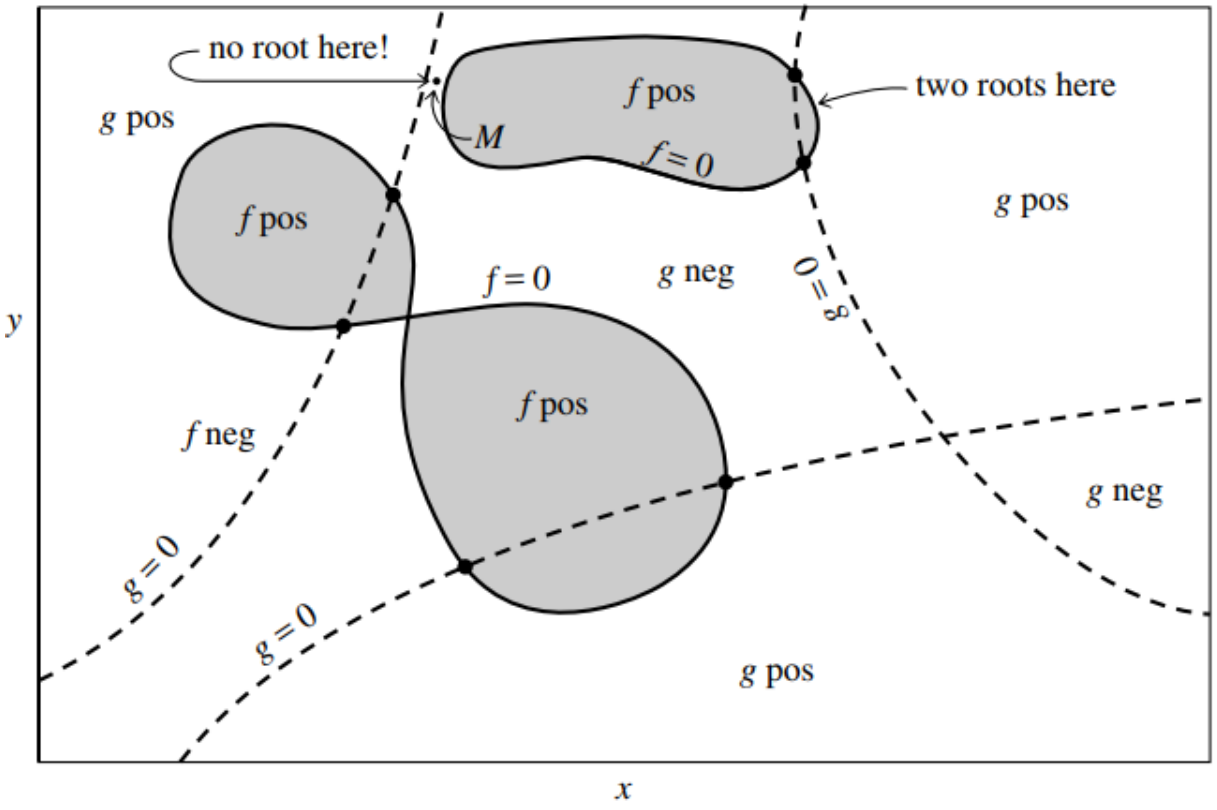
$$\epsilon_{i+1} = \epsilon_i - \frac{f(x_i)}{f'(x_i)}$$

$$\epsilon_{i+1} = -\epsilon_i^2 \frac{f''(x)}{2f'(x)}$$

Newton-Raphson Method for Nonlinear Systems of Equations

Consider the case of two dimensions where we want to solve simultaneously:

$$f(x, y) = 0$$
$$g(x, y) = 0$$



A typical problem gives N functional relations to be zeroed, involving variables $x_i, \quad i = 0, 1, 2, \cdots, N-1$:

$$F_i(x_0, x_1, \cdots, x_{N-1}) = 0, \quad i = 0, 1, 2, \cdots, N-1$$

denote $\mathbf{x} = \{x_i\}$ and $\mathbf{F} = \{F_i\}$

In the neighborhood of $\mathbf{x}$:

$$F_i(\mathbf{x} + \delta\mathbf{x}) = F_i(\mathbf{x}) + \sum_{j=0}^{N-1} \frac{\partial F_i}{\partial x_j}\delta x_j + \mathcal{O}(\delta\mathbf{x}^2)$$

Jacobian matrix $\mathbf{J}$:

$$J_{ij} = \frac{\partial F_i}{\partial x_j}$$

→

$$\mathbf{F}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{F}(\mathbf{x}) + \mathbf{J} \cdot \delta\mathbf{x} + \mathcal{O}(\delta\mathbf{x}^2)$$

Set

$$\mathbf{F}(\mathbf{x} + \delta\mathbf{x}) = 0$$

$$\mathbf{J} \cdot \delta\mathbf{x} = -\mathbf{F}(\mathbf{x})$$

$$\begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \cdots & \frac{\partial F_n}{\partial x_n} \end{pmatrix} \begin{pmatrix} x_2 - x_1 \\ x_3 - x_2 \\ \cdots \\ x_n - x_{n-1} \end{pmatrix} = \begin{pmatrix} -F_1 \\ -F_2 \\ \cdots \\ -F_n \end{pmatrix}$$

The matrix equation can be solved by LU decomposition.

The corrections are then added to the solution vector,

$$\mathbf{x}_{new} = \mathbf{x}_{old} + \delta\mathbf{x}$$

and the process is iterated to convergence.

eg: Solve the nonlinear multi-variable equation:

$$F_1(x,y) = x^2 - 4y = 0$$
$$F_2(x,y) = y - x = 0$$

```
% A simple example of Newton iteration
while ((abs(x1-x0)/x0>1.0e-8) || abs(y1-y0)/y0>1.0e-8) % Conve
  % Initial guesses of x and y
  x0 = 1; y0 = 1;
  a11=dF1dx(x0,y0);
  a22=dF1dy(x0,y0);
  a21=dF2dx(x0,y0);
  a12=dF2dy(x0,y0);
  j=[[a11 a12];[a21 a22]]; % Jacobian matrix
  f1=F1(x0,y0);
  f2=F2(x0,y0);
  f=[f1 f2]';
  dx = j\f; % inverse of j * f

  x1 = x0*(1.0-dx(1));
  y1 = y0*(1.0-dx(2));
end
x_final=x1; y_final=y1;
```

Quasi-Newton Method

$$f(\mathbf{x}) = f(\mathbf{P}) + \sum_i \frac{\partial f}{\partial x_i} x_i + \frac{1}{2} \sum_{i,j} \frac{\partial^2 f}{\partial x_i \partial x_j} x_i x_j + \cdots$$
$$\approx c - \mathbf{b} \cdot \mathbf{x} + \frac{1}{2} \mathbf{x} \cdot \mathbf{A} \cdot \mathbf{x}$$

where

$$c = f(\mathbf{P}) \quad \mathbf{b} = -\nabla f|_{\mathbf{P}} \quad A_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}|_{\mathbf{P}}$$

Hessian Matrix : $\mathbf{A}$

$$\nabla f = \mathbf{A} \cdot \mathbf{x} - \mathbf{b}$$

the gradient vanishes when the function is at an extremum by solving $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$

To build up, iteratively, a good approximation to the inverse Hessian matrix $\mathbf{A}^{-1}$, , that is, to construct a sequence of matrices $\mathbf{H}_i$ with the property

$$lim_{i \to \infty} \mathbf{H}_i = \mathbf{A}^{-1}$$

Consider finding a minimum:

$$f(\mathbf{x}) = f(\mathbf{x}_i) + (\mathbf{x} - \mathbf{x}_i) \cdot \nabla f(\mathbf{x}_i) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_i) \cdot \mathbf{A} \cdot (\mathbf{x} - \mathbf{x}_i)$$

$$\nabla f(\mathbf{x}) = \nabla f(\mathbf{x}_i) + \mathbf{A} \cdot (\mathbf{x} - \mathbf{x}_i) = 0$$

Determine the next iteration point:

$$\mathbf{x} - \mathbf{x}_i = -\mathbf{A}^{-1} \cdot \nabla f(\mathbf{x}_i)$$

the right hand side is known once we have

$$\mathbf{H} \approx \mathbf{A}^{-1}$$