Poh Leong Hien
ceeplh@nus.edu.sg

2 Solution of nonlinear equations

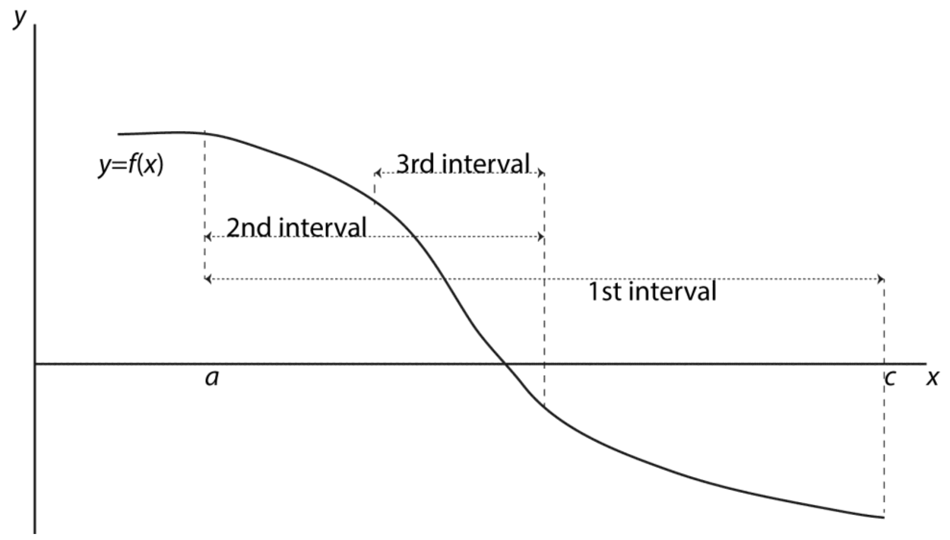
It can be difficult to solve a nonlinear function $f(x)=0$ in closed-form (if it exists), hence the motivation for numerical approximations.

2.1 Bisection method

This is one of the simplest method for finding a root in a given interval.

Idea

- Consider an interval $a \le x \le c$ such that $f(a)\times f(c) = -1$.
- The root for $f(x)=0$ must lie within this interval.
- Bisect the original interval into two halves, i.e. $b = 0.5(a+c)$.
- Consider $f(a)\times f(b)$ and $f(b)\times f(c)$. The root must lie in the (new) interval where the function *f* changes sign at the two end points.
- Update the new end points. Repeat the iteration steps until solution is within a tolerance level.



As this procedure is repeated, the size of the interval with the root becomes smaller and smaller. The interval size after *n* iterations becomes $(c-a)_0 / 2^n$ where the subscript denotes the iteration number.

At each step *n*, the midpoint of the interval is taken as the most updated approximated for the root. The iteration stops when the function value at half interval is within a tolerance level, i.e. $f(b_n) \le \text{tol}$

Poh Leong Hien
ceeplh@nus.edu.sg

2.2 Newton Raphson method

The Newton Raphson method is perhaps the most widely used method in solving nonlinear functions.

<u>Idea</u>

- Start with a trial solution x_1. Typically $f(x_1) \neq 0$, i.e. x_1 is not a root to the function.
- To improve the trial solution with $x_2 = x_1 + \Delta x_1$, such that $f(x_2)$ is closer to zero.
- Continue with the iteration until $f(x_k) \approx 0$. Solution is thus approximated by x_k.

Given that the current trial solution x_i is not satisfactory, i.e. $f(x_1) \neq 0$, we hope that the next trial value $x_{i+1} = x_i + \Delta x_i$ will lead to $f(x_{1+1}) = 0$. The objective then is to find the suitable correction term Δx_i.

Linearize the nonlinear function around x_i with the Taylor's expansion

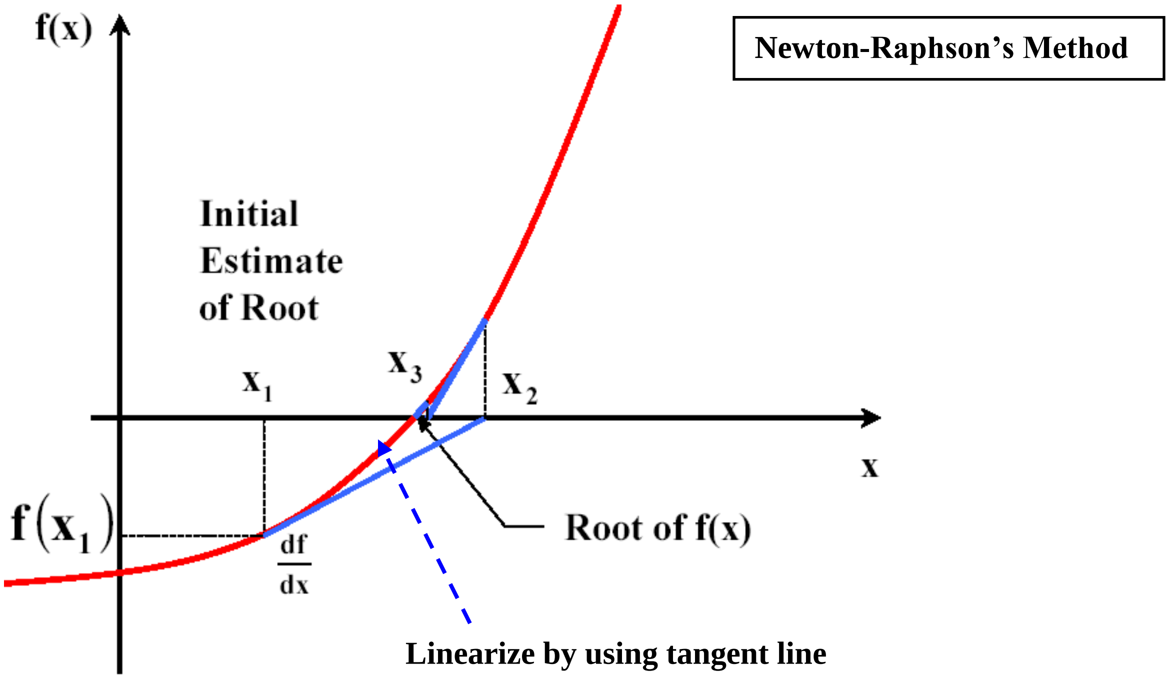
$$f(x_{i+1}) = f(x_i) + \Delta x_i \frac{df(x_i)}{dx} \tag{2.1}$$

where the higher order terms are ignored.

Setting the linearized equation to zero, the correction term is obtained as

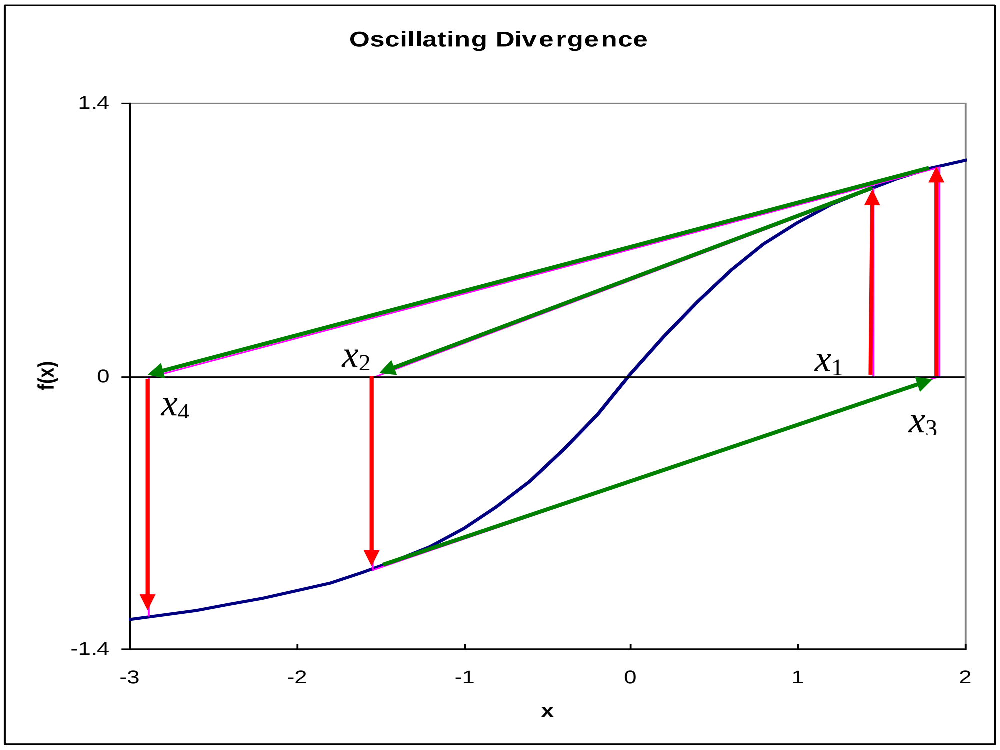
$$\Delta x_i = -\frac{f(x_i)}{df(x_i)/dx} \tag{2.2}$$

Check whether $f(x_{i+1}) \leq \mathrm{tol}$. If so, convergence is achieved and the solution to the nonlinear function is approximated by x_{i+1}. Otherwise, repeat (2.1) – (2.2).



Note: convergence is not guaranteed, i.e. it depends on the function and initial guess.

For example: $f(x) = \tan^{-1}(x)$ with initial guess x_1 = 1.45



Poh Leong Hien
ceeplh@nus.edu.sg

2.3 Newton Raphson method for a system of equations

Extend the concept in 1D to multi-dimensional space. Consider the following two nonlinear equations,

$$\begin{aligned} u &= f(x,y) \\ v &= g(x,y) \end{aligned} \tag{2.3}$$

which can be considered as a transformation (or mapping) from the *xy*-plane to the *uv*-plane.

We are interested in this transformation near a point $(x_k \, , \, y_k)$ which represents a trial (guess) point. Assume that the two functions have continuous partial derivatives. Based on Taylor series expansion about $(x_k \, , \, y_k)$, the differential can be used to write a system of <u>linear</u> approximations:

$$\begin{aligned} u_{k+1} &= u_k + \left.\frac{\partial f}{\partial x}\right|_{(x_k \, , \, y_k)} (x_{k+1} - x_k) + \left.\frac{\partial f}{\partial y}\right|_{(x_k \, , \, y_k)} (y_{k+1} - y_k) \\ v_{k+1} &= v_k + \left.\frac{\partial g}{\partial x}\right|_{(x_k \, , \, y_k)} (x_{k+1} - x_k) + \left.\frac{\partial g}{\partial y}\right|_{(x_k \, , \, y_k)} (y_{k+1} - y_k) \end{aligned} \tag{2.4}$$

In matrix form:

$$\begin{bmatrix} u_{k+1} - u_k \\ v_{k+1} - v_k \end{bmatrix} = \underbrace{\begin{bmatrix} \left.\frac{\partial f}{\partial x}\right|_{(x_k , y_k)} & \left.\frac{\partial f}{\partial y}\right|_{(x_k , y_k)} \\ \left.\frac{\partial g}{\partial x}\right|_{(x_k , y_k)} & \left.\frac{\partial g}{\partial y}\right|_{(x_k , y_k)} \end{bmatrix}}_{\text{Jacobian matrix } \mathbf{J}(x_k \, , \, y_k)} \begin{bmatrix} x_{k+1} - x_k \\ y_{k+1} - y_k \end{bmatrix} \tag{2.5}$$

The aim is to solve

$$\begin{aligned} f(x,y) &= 0 \\ g(x,y) &= 0 \end{aligned} \tag{2.6}$$

Referring to (2.3) and (2.6), we let $u_{k+1} = v_{k+1} = 0$ in (2.5). Thus,

$$\begin{bmatrix} -u_k \\ -v_k \end{bmatrix} = \mathbf{J}(x_k , \, y_k) \begin{bmatrix} x_{k+1} - x_k \\ y_{k+1} - y_k \end{bmatrix} \tag{2.7}$$

Poh Leong Hien
ceeplh@nus.edu.sg

Substitute $\mathbf{f} = [f \quad g]^T$ and $\mathbf{x} = [x \quad y]^T$ into (2.7) to get

$$\underbrace{-\mathbf{f}\,(\mathbf{x}_k)}_{\text{"error"}} = \mathbf{J}(\mathbf{x}_k)\,\underbrace{(\mathbf{x}_{k+1} - \mathbf{x}_k)}_{\text{correction}} \tag{2.8}$$

The next trial is obtained by solving (2.8) for $\mathbf{x}_{k+1}$. Mathematically, at *k*-th iteration:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{J}^{-1}(\mathbf{x}_k)\;\mathbf{f}(\mathbf{x}_k) \tag{2.9}$$

Note: numerically, avoiding solving (2.9) by computing the inverse of **J**.

The iteration continues until the solution has converged.

Example
Solve the following nonlinear system by Newton-Raphson method

$$f_1(x_1, x_2) = x_1^{\;2} - x_2 - 0.2 \;= 0$$
$$f_2(x_1, x_2) = x_2^{\;2} - x_1 - 0.3 \;= 0$$

Graphically, these two equations are parabolas and they intersect at two points which are the solutions. Start with the following two initial points: (1.2, 1.2) and (-0.2, -0.2).

To use Newton-Raphson method, Jacobian matrix is needed:

$$\frac{\partial f_1}{\partial x_1} = 2x_1 \;\;,\;\; \frac{\partial f_1}{\partial x_2} = -1 \;\;,\;\; \frac{\partial f_2}{\partial x_1} = -1 \;\;,\;\; \frac{\partial f_2}{\partial x_2} = 2x_2$$

Hence, $\mathbf{J} = \begin{bmatrix} 2x_1 & -1 \\ -1 & 2x_2 \end{bmatrix}$

% Define the nonlinear system in Matlab M-file F.m

```
Function Z=F(X)

x=X(1); y=X(2);
Z=zeros(1,2);
Z(1)=x^2-y-0.2;
Z(2)=y^2-x-0.3;
```

% Define the Jacobian function for the nonlinear system in Matlab M-file JF.m

```
function W=JF(X)

x=X(1); y=X(2);
W=[2*x   -1; -1   2*y];
```

Poh Leong Hien
ceeplh@nus.edu.sg

% Newton Raphson iteration in Matlab M-file NR.m

```
function [P,iter,err,Y,relerr]=NR(F,JF,X,tol,epsilon,maxI)

%F is the system saved as the M-file F.m
%JF is the Jacobian of F saved as the M-file JF.m
%X is the initial approximation to the solution
%tol is the tolerence for P
%epsilon is the tolerence for F(P)
%maxI is the maximum number of iterations
%P is the approximation to the solution
%iter is the number of iterations required
%err is the error estimate for P
%relerr is the relative error estimate for P
%Y is the evaluated function

P=X;
Y=feval('F',P);

for k=1:maxI
    J=feval('JF',P);
    Q=P-(J\Y');
    G=feval('F',Q);
    err=norm(Q-P,inf);
    relerr=err/(norm(Q)+eps);
    P=Q;
    Y=G;
    iter=k;
    if (err<tol)|(relerr<tol)|(abs(Y)<epsilon)
        break
    end
end
```

<u>Output 1</u>:

```
>> X=[1.2; 1.2];   % Solving using initial point (1.2,1.2)
>> Z=F(X)   % Solving for the error term associated with trial values
Z =
   0.0400   -0.0600

>> W=JF(X)   % Solving for the Jacobian at the trial values
W =
   2.4000   -1.0000
  -1.0000    2.4000

>> tol=1.0e-10;
>> epsilon=1.0e-10;
>> maxI=10;
>> [P,iter,err,Y,relerr]=NR(Z,W,X,tol,epsilon,maxI)   % NR iteration
P =
    1.1923      Solution (approx.)
    1.2216
iter =
     3
err =
  3.3691e-008
Y =
  1.0e-014 *
    0.0389    0.1166
relerr =
  1.9737e-008
```

<u>Output 2</u>:

```
>> X=[-0.2; -0.2];   % Solving using initial point (-0.2,-0.2)
>> Z=F(X);
>> W=JF(X);
>> [P,iter,err,Y,relerr]=NR(Z,W,X,tol,epsilon,maxI)
P =
   -0.2860      Solution (approx.)
   -0.1182
iter =
     4
err =
  1.1262e-009
Y =
     0     0
relerr =
  3.6389e-009
```