이동현상 특수연구

Objectives: Understand the fundamental
numerical computation for DEs
stability analysis of chem. & bio. Processes
nonlinear dynamics and bifurcation

References: S.H. Strogatz, Nonlinear dynamics and chaos (1994)
R. Seydel, Practical bifurcation and stability analysis (1994)
M.M. Denn, Stability of reaction and transport processes (1975)
B.W. Bequette, Process dynamics (1998)
P.G. Drazin & W.H. Reid, Hydrodynamic stability (1982)
S. Chandrasekhar, Hydrodynamic and hydromagnetic stability (1961)
G. Ioose and D.D. Joseph, Elementary stability and bifurcation theory (1980)
D.D. Perlmutter, Stability of chemical reactors (1972)
P.G. Drazin, Introduction to hydrodynamic stability (2002)
D.Y. Hsieh & S.P. Ho, Wave and stability in fluids (1994)
…

이동현상 특수연구

Process Dynamics: Modeling, Analysis, and Simulation
By B. W. Bequette (1998)

Objective: understand the dynamic behavior of chemical and biological processes

Modeling: 1D, 2D, 3D
Simulation: FDM, FEM, FVM, OCM, etc.
Analysis: Linear & nonlinear analysis
Steady & transient responses
Stability/ sensitivity analysis
Chaotic motions
Bifurcation analysis
…

→ ***Processability, productivity***

Section I. Process Modeling
Chapter 1. Introduction

Process modeling: differential equation systems
time-dependent mathematical models of the chemical and biological processes

- **Models**: A set of equations (including input data) that allows us to predict the behavior of chemical and biological processes
  Approximate representation of an actual processes

Fundamental models: conservation of mass, momentum, energy.
constitutive equations (Newton & viscoelatic models)
reaction kinetics, crystallization, transport phenomena,
thermodynamic relationships (phase equilibrium), etc.

Empirical models: least square fit of experimental data, overall heat transfer coeff.
useful for “interpolation”, not “extrapolation”

The complexity of a process model depends on the final use of the model.

How models are used:

- Marketting, allocation, synthesis, design, operation, control, etc.

- **Steady state**: variables does not change with time
  Dynamic: variables change with time

- **Lumped parameter systems**: variables change only with one independent variable (time) – ODE systems, CSTR, etc.
  Distributed parameter systems: ~ with more than one independent variable (time and space) – PDE systems, tubular reactor, etc.

  - **Algebraic equations**
    ODEs
    PDEs

  - **Example 1.1.** A lumped parameter system
    Perfect insulated CSTR
    Uniform temperature in the tank
    Temp. does not change with space.

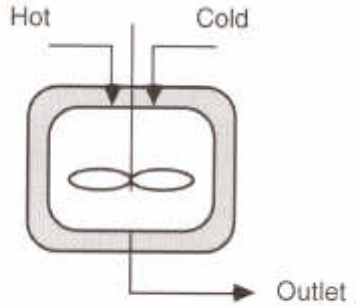


FIGURE 1.1 Stirred tank.

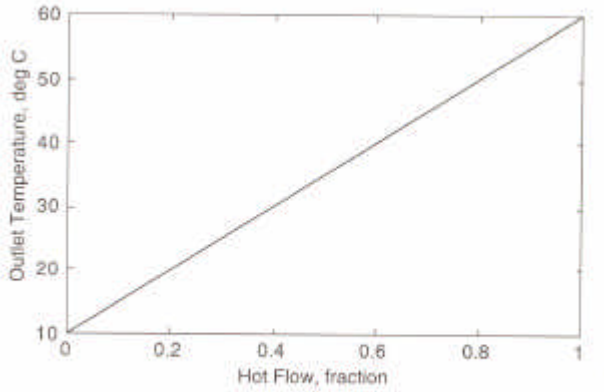


FIGURE 1.2 Relationship between hot flow and outlet temperature.

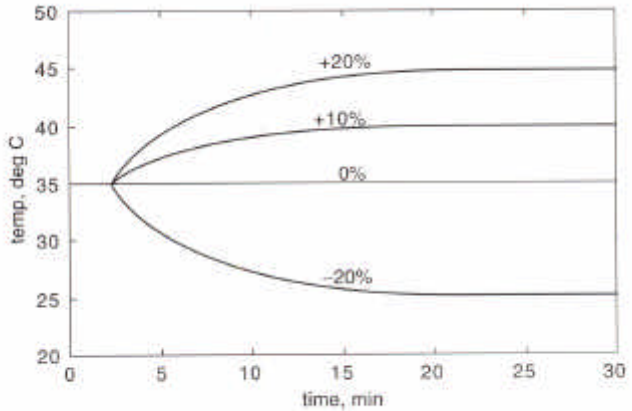


FIGURE 1.3 Response of temperature to various changes in the hot flow fraction.

- **Example 1.2.** A distributed parameter system
  Counterflow heat exchanger

FIGURE 1.4 Counterflow heat exchanger.

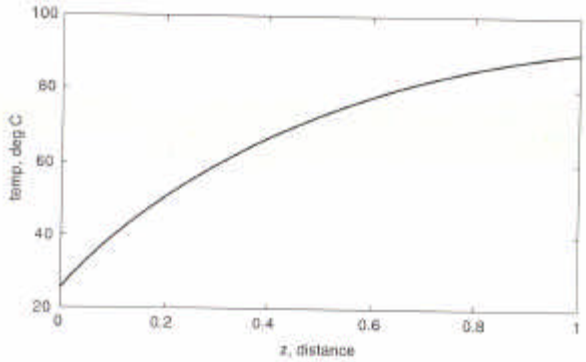


FIGURE 1.5 Water temperature as a function of position.

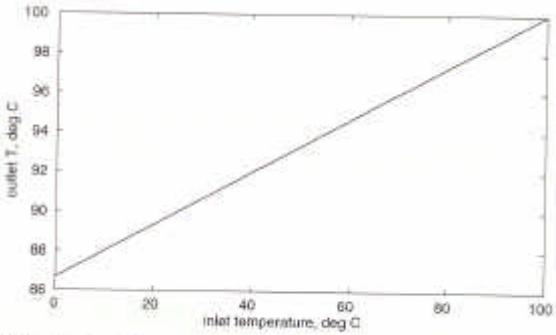


FIGURE 1.6 Outlet water temperature as a function of inlet water temperature.

Temp. of the water system changes with time and space (position).

Systems

- **System**: a combination of several pieces of equipment integrated to perform a specific function.
  (composed of chemical unit operations such as chemical reactors, heat exchangers, separation devices, etc.)

- **Simulation**: steady-state simulation of lumped parameter systems: algebraic eqn.
  dynamic simulation: ODEs
  (PDE can be converted to ODEs by appropriate numerical techniques, i.e., method of lines)

- **Linear system analysis**: Laplace transforms
  Eigenvalue and eigenvector analysis

- **A broader view of analysis**: Understand how the response of system variable changes when a parameter or input changes.

Qualitative change of the systems

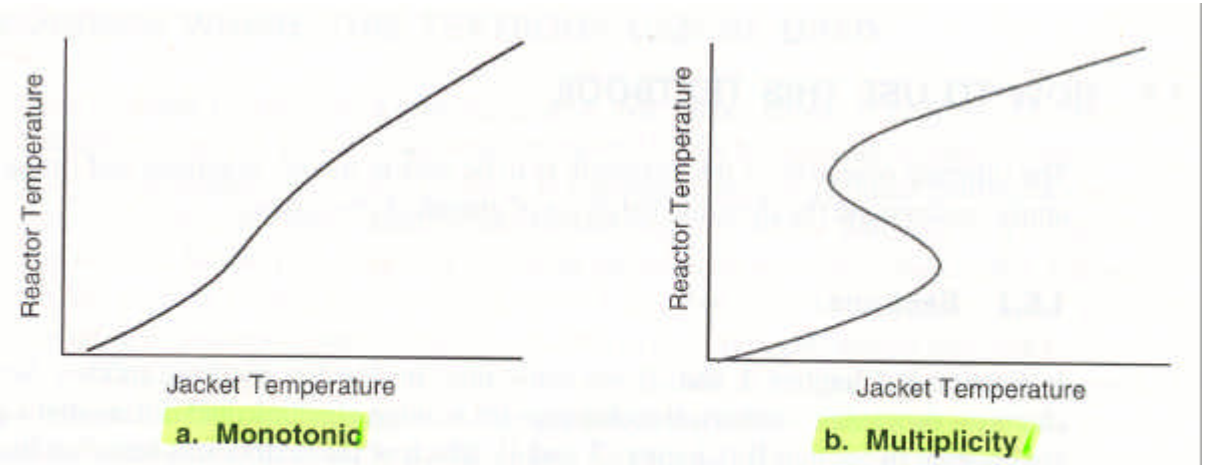


FIGURE 1.7 Two qualitatively different types of input/output behavior. Steady-state reactor temperature as a function of steady-state jacket temperature.

Chapter 2. Process Modeling

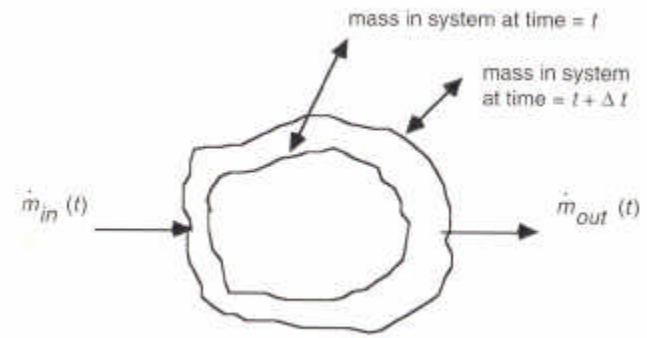
- Methodology for developing dynamic models of chemical and biological processes

- **Balance equations** *(see Books by Bird et al., Welty et al., and Deen)*

- **Steady state balance equations**:
  (mass (M) or energy (E) entering a system) – (~ leaving~) =0
  Generation and consumption of species by reaction can be included…

- **Dynamic balances**:
  (rate of M or E accumulation in a system)
  = (rate of M or E entering ~) – (rate of M or E leaving ~)
  dM/dt, dE/dt, dNi/dt, …

- **Integral balances**:
  (M or E inside the system at t+Δt) – (M or E inside the system at t) =
  (M or E entering ~ from t to t+Δt) – (M or E leaving ~ from t to t+Δt)



→ ODE systems by mean-value theorem and differential calculus.

Example) Total mass balance equation

$$M\big|_{t+\Delta t} - M\big|_t = \int_t^{t+\Delta t} \dot{m}_{in}\,dt - \int_t^{t+\Delta t} \dot{m}_{out}\,dt = \int_t^{t+\Delta t} (\dot{m}_{in} - \dot{m}_{out})\,dt$$

By mean-value theorem,

$$M\big|_{t+\Delta t} - M\big|_t = (\dot{m}_{in} - \dot{m}_{out})\big|_{t+a\Delta t}\,\Delta t \quad (0 < a < 1)$$

$$\Rightarrow \frac{dM}{dt} = \dot{m}_{in} - \dot{m}_{out} \quad or \quad \frac{dVr}{dt} = F_{in}\,r_{in} - F_{out}\,r$$

(V: volume, ρ: mass density, F: volumetric flowrate)

- **Instantaneous balances**:

(rate of accumulation of M in the system) = (rate of M entering) – (rate of M leaving)

$$\Rightarrow \frac{dM}{dt} = \dot{m}_{in} - \dot{m}_{out} \quad or \quad \frac{dVr}{dt} = F_{in}\, r_{in} - F_{out}\, r$$

Material balances

- **Example 2.1.**: Liquid surge tank

$$\frac{dVr}{dt} = F_{in}\, r - Fr \;\rightarrow\; (const.\ r)\ \frac{dV}{dt} = F_i - F$$

ODE system – lumped parameter system

(initial condition V(0) is required)

State variable: V

Input variables: F_i, F

$$\frac{dh}{dt} = -\frac{b\sqrt{h}}{A} + \frac{F_i}{A} \quad (V = Ah,\ F = b\sqrt{h})$$

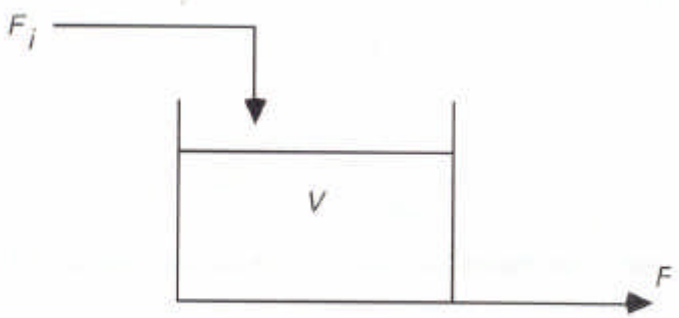


FIGURE 2.2 Liquid surge tank.

- **Example 2.2.**: An isothermal chemical reactor

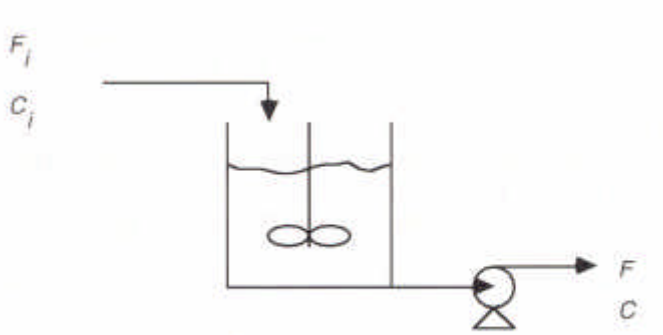


FIGURE 2.3 Isothermal chemical reactor.

Overall material balance:

$$\frac{dV}{dt} = F_i - F$$

Component material balance: (in molar units)
(A+2B → P)

$$\frac{dVC_A}{dt} = F_i C_{Ai} - FC_A + Vr_A, \quad \frac{dVC_B}{dt} = F_i C_{Bi} - FC_B + Vr_B$$

$$\frac{dVC_P}{dt} = -FC_P + Vr_P$$ (r_i: rate of species i per unit volume, C_{Ai}, C_{Bi}: inlet conc.)

Assume: $r_A = -kC_A C_B$ *then* $r_B = 2r_A = -2kC_A C_B$, $r_P = -r_A = kC_A C_B$

$$\Rightarrow \frac{dC_A}{dt} = \frac{F_i}{V}(C_{Ai} - C_A) - kC_A C_B, \quad \frac{dC_B}{dt} = \frac{F_i}{V}(C_{Bi} - C_B) - 2kC_A C_B$$

$$\frac{dC_P}{dt} = -\frac{F_i}{V}C_P + kC_A C_B$$

State variables: V, C_A, C_B, C_P
Input variables: F_i, C_{Ai}, C_{Bi}
Parameter: k
Initial conditions: V(0), $C_A(0)$, $C_B(0)$, $C_P(0)$

- **Example 2.3.**: Gas surge drum

Pressure variation in the tank with time ?

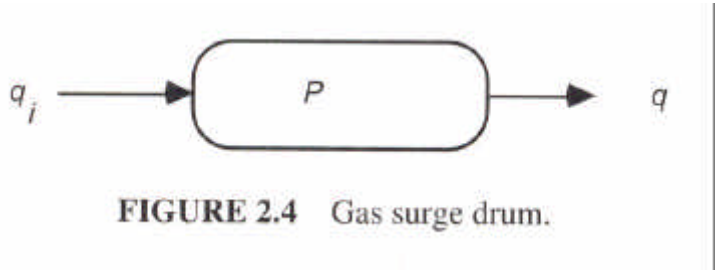


FIGURE 2.4 Gas surge drum.

Ideal gas law: $\frac{1}{\hat{V}} = \frac{P}{RT}$

Total amount of gas in the tank: $\frac{V}{\hat{V}} = \frac{PV}{RT}$

Rate of accumulation of gas: d(PV/RT)/dt

$$\frac{V}{RT}\frac{dP}{dt} = q_i - q \quad (q : molar \quad flowrate)$$

State variable: P; inputs: q_i, q; parameter: R, T, V; initial conditions: P(0)

- **Constitutive relationships**

Gas law, chemical reactions, equilibrium relationships, heat transfer, flow-through valves, etc.

Material and energy balances

- Review of thermodynamics

T_E (total energy) = U (internal) + K_E (kinetic) + P_E (potential)

(K_E = ½ mv^2, P_E=mgh)

$$\hat{T}_E = \hat{U} + \hat{K}_E + \hat{P}_E \ \text{(energy/mole)}, \quad \overline{T}_E = \overline{U} + \overline{K}_E + \overline{P}_E \ \text{(energy/mass)}$$

(In many chemical processes where there are thermal effects, the kinetic and potential energy terms can be neglected.)

$$H = U + PV \rightarrow \overline{H} = \overline{U} + P\overline{V} = \overline{U} + P/r$$

- **Example 2.4.**: Stirred tank heater

Assumptions: kinetic and potential energy effects neglected.
changes in PV term neglected.

Material balance: $$\frac{dVr}{dt} = F_i r_i - Fr$$

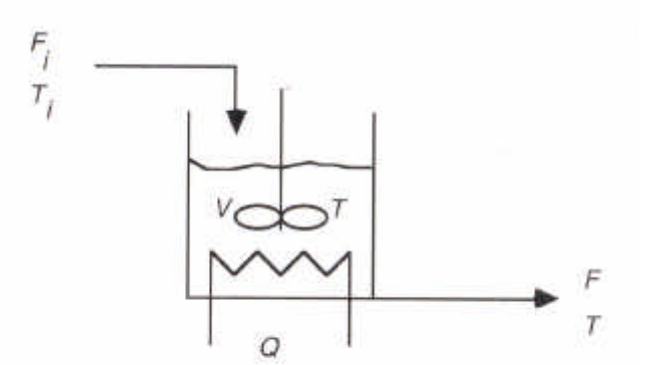


FIGURE 2.6 Stirred tank heater.

Energy balance:

Accumulation=in by flow-out by flow
+ in by heat transfer
+ work done on system

$$\frac{dT_E}{dt} = F_i\rho_i\overline{T}_{Ei} - F\rho\overline{T}_E + Q + W_T \rightarrow \frac{dU}{dt} = F_i\rho_i\overline{U}_i - F\rho\overline{U} + Q + W_T$$

$$W_T = W_s + F_i p_i - Fp$$

$$\frac{dU}{dt} = F_i r_i \left(\overline{U}_i + \frac{p_i}{r_i} \right) - Fr \left(\overline{U} + \frac{p}{r} \right) + Q + W_s \Rightarrow \quad \frac{dH}{dt} = F_i r_i \overline{H}_i - Fr\overline{H} + Q + W_s$$

- **Distributed parameter systems**

  State variables change with time and space (PDE systems)

- **Tubular reactor modeling** under the convection flow

  Balance equations:

  Material balance:

$$(\Delta V) C_A\big|_{t+\Delta t} - (\Delta V) C_A\big|_t = \int_t^{t+\Delta t} \left(F C_A\big|_V - F C_A\big|_{V+\Delta V} - k C_A \Delta V \right) dt$$

  By mean-value theorem and $\Delta t, \Delta V \rightarrow 0$:

$$\frac{\partial C_A}{\partial t} = -\frac{\partial F C_A}{\partial V} - k C_A \Rightarrow \frac{\partial C_A}{\partial t} = -\frac{\partial v_z C_A}{\partial z} - k C_A \quad (dV = A dz,\ F = A v_z)$$

  Overall material balance: $\frac{\partial r}{\partial t} = -\frac{\partial v_z r}{\partial z}$ (for const. density) $v_z = const$

$$\Rightarrow \frac{\partial C_A}{\partial t} = -v_z \frac{\partial C_A}{\partial z} - k C_A$$ (initial condition & one boundary condition)

$$C_A(z, t=0) = C_{A0}(z)$$

$$C_A(0, t) = C_{Ain}(t)$$

Dimensionless models

(simple constant volume, isothermal CSTR model)

$$\frac{dC_A}{dt} = \frac{F}{V}(C_{Af} - C_A) - kC_A$$

Let $x = \frac{C}{C_{Af}}, t = \frac{t}{t^*}, a = \frac{Vk}{F} \quad \Rightarrow \frac{dx}{dt} = 1 - x + ax$

General form of dynamic models

$$\frac{dx_1}{dt} = \dot{x}_1 = f_1(x_1,...,x_n,u_i,...,u_m,p_1,...,p_r)$$

$$\vdots$$

$$\frac{dx_n}{dt} = \dot{x}_n = f_n(x_1,...,x_n,u_i,...,u_m,p_1,...,p_r)$$

vector notation: $\underline{\dot{x}} = \underline{f}(\underline{x},\underline{u},\underline{p})$

steady state: $\underline{f}(\underline{x},\underline{u},\underline{p}) = \underline{0}$

Section II. Numerical Techniques
Chapter 3. Algebraic Equations

- **Introduction**

  From steady state of $\dot{\underline{x}} = \underline{f}(\underline{x}) = \underline{0}$ find $\underline{x}$ (fixed points, equilibrium points)

- **General form for a linear system of equations**

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$
$$\vdots$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$$

$$vector\ \ form: \ \underline{\underline{A}}\,\underline{x} = \underline{b}$$

$\underline{x}$ is obtained by inverse of matrix A
LU decomposition
Gauss or Gauss-Jordan elimination
Others…

- **Nonlinear functions of a single variable**

  Single solution & multiple solutions
  Convergence tolerance: absolute tolerance $\left|x_k - x_{k-1}\right| \le e_a$

  relative tolerance $\dfrac{\left|x_k - x_{k-1}\right|}{\left|x_{k-1}\right|} < e_r$

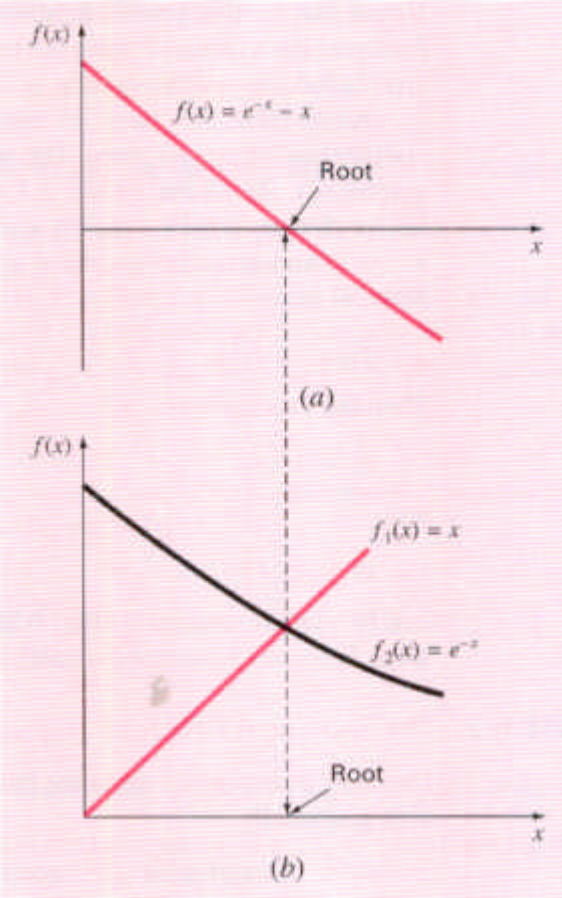
Iterative methods for finding solutions or roots

→ Fixed-point iteration, bisection, false position, Newton's method

(1) Simple fixed-point iteration

One-point iteration or successive iteration

x=g(x) by rearranging the function f(x)=0, then $x_{i+1}=g(x_i)$



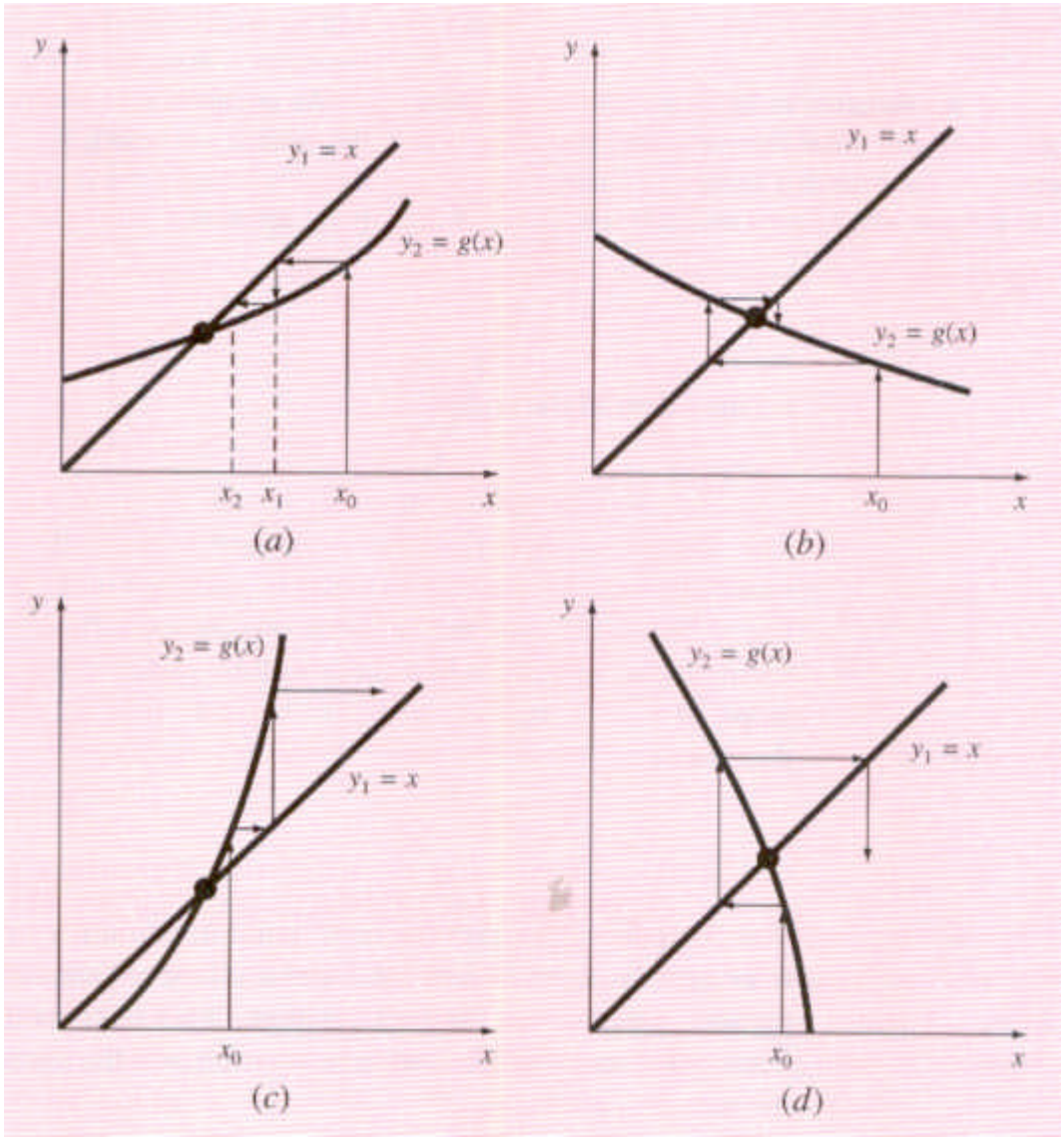
EXAMPLE

```
c ... EXAMPLE 6.1
c ... f(x)=e(-x) - x

      double precision x,f
      x = 0.d0
      iter = 1
      error = 1.d-5
100   f = dexp(-x)
      gap = dabs(x - f)
      print *, 'iter:',iter, ' x:',x
      if(gap.lt.error) then
      print *,' root:',x
      stop
      else
      x = f
      iter = iter + 1
      goto 100
      endif

      stop
      end
```

```
iter:   1  x:  0.000000000000000E+000
iter:   2  x:     1.000000000000000
iter:   3  x:  3.678794411714423E-001
iter:   4  x:  6.922006275553464E-001
iter:   5  x:  5.004735005636368E-001
                       ...
iter:  20  x:  5.671570440012975E-001
iter:  21  x:  5.671354902062784E-001
iter:  22  x:  5.671477142601192E-001
```

Linear convergent !!!

Converging and diverging cases of fix-point iteration



Convergence condition

Fixed-point iteration converges, if $|g'(x)| < 1$

If we let

$$\Delta_{n+1} = \left|x_{n+1} - x^*\right| = \left|g(x_n) - x^*\right|$$

Putting $x_n = x^* + \Delta_n$

We get

$$\Delta_{n+1} = \left|g(x^* + \Delta_n) - x^*\right| = \left|\frac{dg(x^*)}{dx^*}\right|\Delta_n$$

For convergence $\left|\frac{\Delta_n}{\Delta_{n+1}}\right| > 1$

(2) Bisection method

If f(xL) and f(xU) have opposite signs, i.e., f(xL) f(xU) < 0,
(*f(x) is real and continuous*)
then, there is at least one real root between xL and xU.

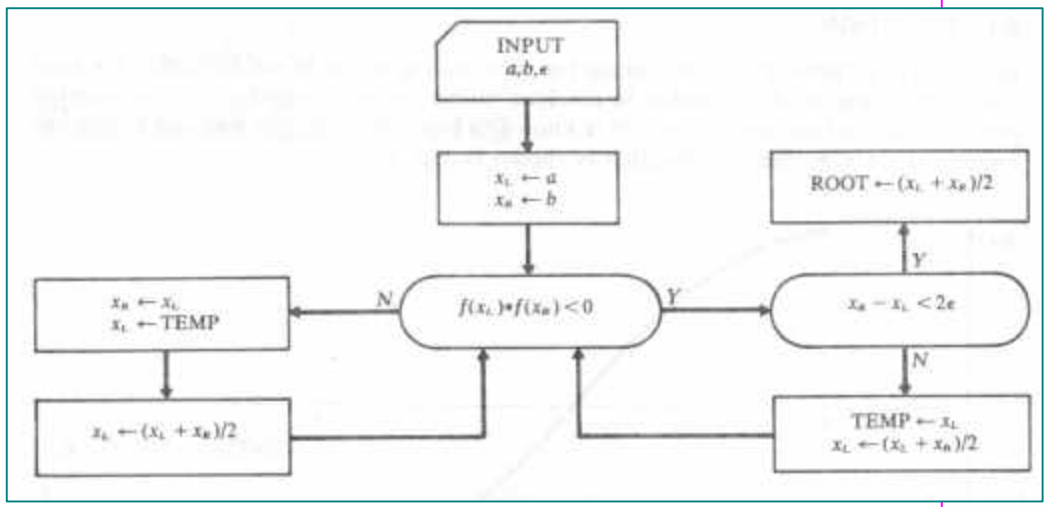
Bisection method: x interval is always divided in half.

Termination criteria and error estimates:

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

"Brute-force" method (inefficient)

Flow chart for bisection method



EXAMPLE

```
c ... Bisection method for finding a root
      implicit double precision (a-h,o-z)
      parameter (error=1.d-5)
      external f
      print *,'x1 and x2'
      read(*,*) x1,x2
      iter = 1
100   f1 = f(x1)
      f2 = f(x2)
      gap = x2-x1
      fmulti = f1*f2
      print *,' iter:',iter,' xroot',(x1+x2)/2.d0
      if(fmulti.lt.0.d0) then
         if(gap.lt.error) then
         xroot = (x1+x2)/2.d0
         else
         temp = x1
         x1 = (x1+x2)/2.d0
         iter = iter+1
         goto 100
         endif
      else
      x2 = x1
      x1 = temp
      x1 = (x1+x2)/2.d0
      iter = iter+1
      goto 100
      endif

      stop
      end

      double precision function f(x)
      implicit double precision (a-h,o-z)
      f = 667.38d0/x*(1.d0-dexp(-0.146843d0*x)) - 40.d0
      return
      end
```

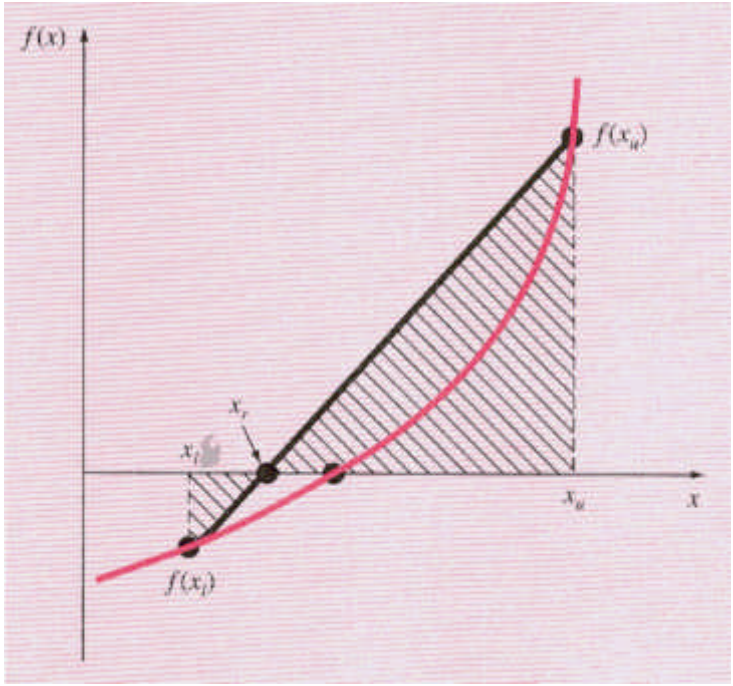
(3) False-position method

Alternative based on a graphical insight, instead of bisection method.

→ Find a root from straight line connecting f(xL) and f(xU)

Replacement of the curve by a straight line → "*false position*"

$$\frac{f(x_L)}{(x_R - x_L)} = \frac{f(x_U)}{(x_R - x_U)} \quad \Rightarrow \quad x_R = x_U - \frac{f(x_U)(x_L - x_U)}{f(x_L) - f(x_U)}$$

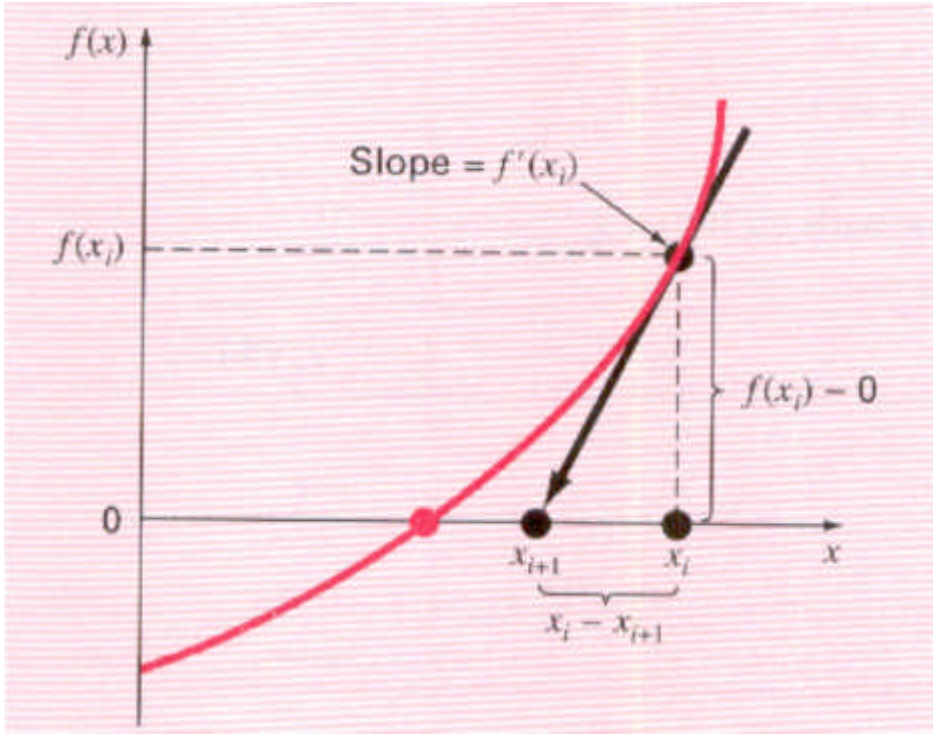


False-position is more efficient than bisection method

(4) Newton-Raphson method

Most wisely used for finding roots

Initial guess of a root, old x_i → Find new x_{i+1} from tangent at old x_i.



Formula for N-R method

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2 + \ldots$$

$$0 = f(x_0) + f'(x_0)(x - x_0)$$

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

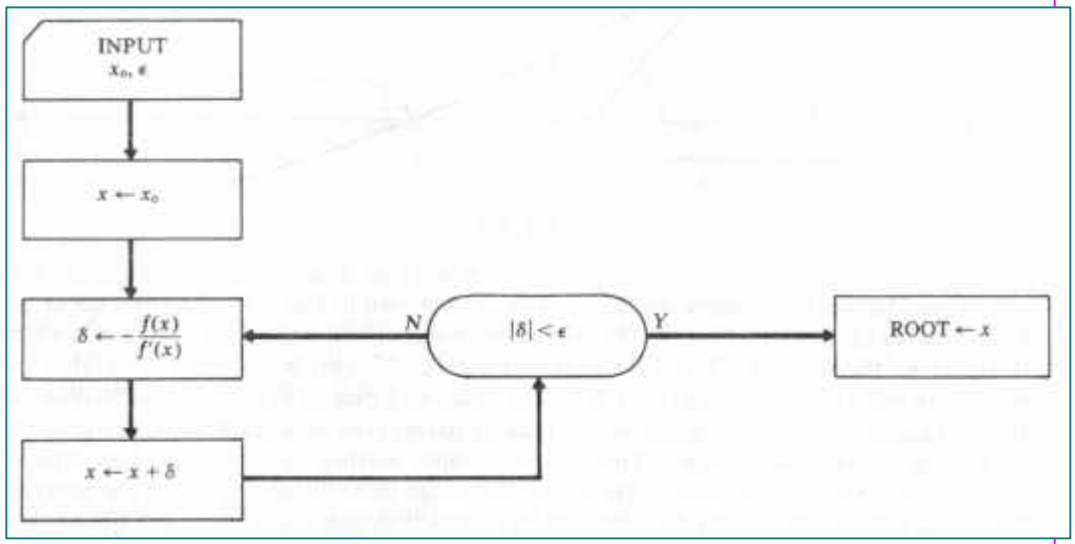
$$\Rightarrow \quad x^{n+1} - x^n = d^{n+1} = -\frac{f(x^n)}{f'(x^n)}$$

Residual

Jacobian

Quadratic convergent !!!

Flow chart for N-R method



EXAMPLE

```
c ... EXAMPLE 6.3. Newton-Raphson Method
c ... f(x) = e(-x) - x

      implicit double precision (a-h,o-z)
      parameter (error=1.d-10)
      external f,fp

      print *,'initial guess of x:'
      read(*,*) x0
      iter = 1
100   delta = -f(x0)/fp(x0)
      x = x0 + delta
      print *,'iter:',iter,' delta:',delta
      if(delta.lt.error) then
      print *,'root is',x
      else
      x0 = x
      iter = iter + 1
      goto 100
      endif

      stop
      end

      double precision function f(x0)
      implicit double precision (a-h,o-z)
      f = dexp(-x0) - x0
      return
      end

      double precision function fp(x0)
      implicit double precision (a-h,o-z)
      fp = -dexp(-x0) - 1.d0
      return
      end
```

```
iter:        1 delta:  5.000000000000000E-001
iter:        2 delta:  6.631100319721815E-002
iter:        3 delta:  8.321618376440470E-004
iter:        4 delta:  1.253749188553222E-007
iter:        5 delta:  2.808428767121099E-015
root is  5.671432904097838E-001
```

(5) Newton's method for multivariable problems

$$\underline{f}(\underline{x}) = \underline{0}$$

A set of n equations with n unknowns $\begin{bmatrix} f_1(x_1, x_2, ..., x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, ..., x_n) = 0 \end{bmatrix}$

Taylor's series:

$$f_i(x + \Delta x) = f_i(x) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j} \Delta x_j + h.o.t.$$

$$\underline{f}(\underline{x} + \underline{\Delta x}) = \underline{f}(\underline{x}) + \underline{\underline{J}}\,\underline{\Delta x} \quad (\underline{\underline{J}} : Jacobian)$$

$$\Rightarrow \underline{\underline{J}}_k (\underline{x}_{k+1} - \underline{x}_k) = \underline{\underline{J}}_k \Delta x_k = -\underline{f}(\underline{x}_k)$$

$$\underline{\underline{J}} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

Example 3.3.

```
c ... Newton's method for multivariable problems
      implicit double precision (a-h,o-z)
      parameter (n=2,tol=1.d-9)
      common /newt1/ ajac(n,n),res(n)
      common /soln1/ x(n),xold(n),dx(n)
      integer indx(n)

c ... initial guess of x vector
      iter = 0
      do i=1,n
      x(i) = -1.d0
      enddo

100   iter = iter + 1
      do i=1,n
      xold(i) = x(i)
      enddo

c ... jacobian matrix and residuals
      call newt

c ..  solve dx vector (LU decomposition)
      call ludcmp(ajac,n,indx,dtemp)
      call lubksb(ajac,n,indx,res)

      do i=1,n
      dx(i) = res(i)
      enddo

c ... find solutions
      do i=1,n
      x(i) = xold(i) + dx(i)
      enddo

      errsum=0.d0
      do i=1,n
      errsum = errsum + dx(i)*dx(i)
      enddo
      errsum = dsqrt(errsum)
      print *,iter,errsum
      if (errsum.lt.tol) then
      write(*,*) (x(i),i=1,n)
      else
      goto 100
      endif

      stop
      end

c ---------------------------------------
      subroutine newt
      implicit double precision (a-h,o-z)
      parameter (n=2)
      common /newt1/ ajac(n,n),res(n)
      common /soln1/ x(n),xold(n),dx(n)

      do 100 i=1,nn
      res(i) = 0.d0
      do 100 j=1,nn
      ajac(i,j) = 0.d0
100   continue

c ... jacobian
      ajac(1,1) = 1.d0 - 8.d0*xold(1) - xold(2)
      ajac(1,2) = -xold(1)
      ajac(2,1) = 3.d0*xold(2)
      ajac(2,2) = 2.d0 - 2.d0*xold(2) + 3.d0*xold(1)

c ... residuals
      res(1) = -(xold(1)-4.d0*x(1)*x(1)-x(1)*x(2))
      res(2) = -(2.d0*x(2)-x(2)*x(2)+3.d0*x(1)*x(2))

      return
      end
```