KMU 255
Computer Programming

Hacettepe University
Department of Chemical Engineering
Fall Semester

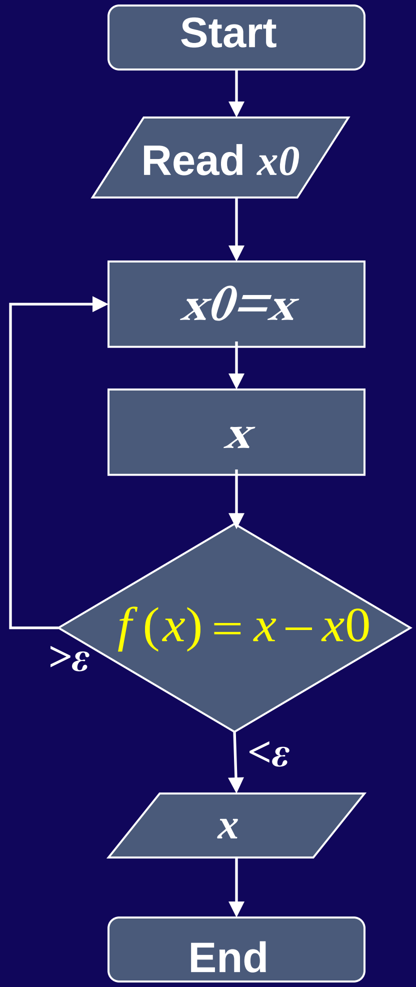
Iterating methods

Selis Önel, PhD

What are we going to learn today?

- Review of iteration
- Various iterative techniques to solve equations
- Jacobi iteration
- Least squares method

A x = y Solution by Iteration: Convergence



1. Initial guess values are used to calculate new guess values
2. New estimates of x are calculated
3. Iteration continues until convergence is satisfied, i.e. $f(x)<\varepsilon$

ε : convergence criteria (tolerance)

Selis Önel, PhD

Jacobi (Simple) Iteration

(1) $a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n = y_1$

(2) $a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n = y_2$

..

(n) $a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n = y_n$

$\sum_{j=1}^{n} a_{i,j}x_j = y_i$, where $i = 1, 2, \ldots, n$. Extracting x_i yields $a_{i,i}x_i + \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j}x_j = y_i$

Solving for x_i gives: $$x_i = \frac{1}{a_{i,i}}\left(y_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j}x_j \right)$$

Consequently, the iterative scheme should be $$x_i \leftarrow \frac{1}{a_{i,i}}\left(y_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j}x_j \right)$$

Jacobi (Simple) Iteration Cycle

1. Choose a starting vector xo (Initial guesses)
2. If a good guess for solution is not available, choose x randomly
3. Use $$x_i \leftarrow \frac{1}{a_{i,i}}\left(y_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j} x_j \right)$$ with x_j =xo to recompute each value of x
4. Check if |x-xo|<ε (tolerance), if so x=xo
5. If |x-xo|>ε, assign new values to xo
6. Repeat this cycle until changes in x between successive iteration cycles become sufficiently small, i.e, |x-xo|<ε

Jacobi (Simple) Iteration

$$x_i^{(t)} = \frac{1}{a_{i,i}}\left(y_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j} x_j^{(t-1)} \right),$$ where t is the iteration count

for t=1 $\rightarrow$ $$x_i^{(1)} = \frac{1}{a_{i,i}}\left(y_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j} x_j^{(0)} \right),$$ where $x_j^{(0)}$ is the initial guess x0

if $\left| x_{\mathrm{i}}^{(1)} - x_i^{(0)} \right| > \varepsilon$ $\rightarrow$ $$x_i^{(2)} = \frac{1}{a_{i,i}}\left(y_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j} x_j^{(1)} \right)$$

continue iteration until $\left| x_{\mathrm{i}}^{(t)} - x_i^{(t-1)} \right| \le \varepsilon$ or $\left| \mathrm{y}_{\mathrm{i}} - \left(a_{i,i} x_i^{(t)} - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j} x_j^{(t)} \right) \right| \le \delta$

Ex: Jacobi (Simple) Iteration

(1) $4x_1 - 2x_2 + x_3 = 3$

(2) $3x_1 - 7x_2 + 3x_3 = -2$

(3) $x_1 + 3x_2 - 5x_3 = -8$

→

$$\begin{pmatrix} 4 & -2 & 1 \\ 3 & -7 & 3 \\ 1 & 3 & -5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ -2 \\ -8 \end{pmatrix}$$

→ $ax = y$

$$x_1 = \frac{3-(-2x_2 + x_3)}{4}$$

$$x_2 = \frac{-2-(3x_1 + 3x_3)}{-7}$$

$$x_3 = \frac{-8-(x_1 + 3x_2)}{-5}$$

```
>> x0=zeros(n,1)
x0 =
     0
     0
     0
```

```
t =      1
x =   0.75000000000000
                     0
                     0
t =      1
x =   0.75000000000000
      0.28571428571429
                     0
t =      1
x =   0.75000000000000
      0.28571428571429
      1.60000000000000
```

```
t =      2
x =   0.49285714285714
      0.28571428571429
      1.60000000000000
t =      2
x =   0.49285714285714
      1.29285714285714
      1.60000000000000
t =      2
x =   0.49285714285714
      1.29285714285714
      1.92142857142857
```

Ex: Jacobi (Simple) Iteration

```
%Solve 3 strictly diagonally dominant linear equations for 3 unknowns:
a=[4 -2 1;3 -7 3;1 3 -5];     %Coefficient matrix
y=[3;-2;-8];                  %Vector for values of f(x)=ax
n=length(y);
x=zeros(n,1);                 %Create an empty matrix for x
xo=x;                         %Initial guess values for x
tmax=50;     %Set max iteration no to stop iteration if system does not converge
tol=10^-3;                    %Set the tolerance to end iteration before t=tmax
for t=1:tmax,                 %Start iteration
   for j=1:n,     x(j)=(y(j)-a(j,[1:j-1,j+1:n])*xo([1:j-1,j+1:n]))/a(j,j);
   end
   error=abs(x-xo);   xo=x;
   if error<=tol,    ' Convergence is good. Iteration ended before tmax '
      break
   end
end
display('Iteration no=');  display(t-1);
x
```

Ex: Jacobi (Simple) Iteration

Results of the Jacobi iteration in the command window:

```
ans =
Convergence is good. Iteration ended before tmax
Iteration no=
ans =
   18
x =
  1.00011187524906
  1.99949883459545
  2.99983186316654
```

Direct solution by Gauss elimination in the command window:

```
>> x=a\y
x =
     1
     2
     3
```

Falling Parachutist Problem

Analytical Solution

&

Numerical Solution

Falling Parachutist Problem

Newton's 2nd law of motion $F = m.a$ describes a natural process or system in mathematical terms.

F : net force acting on the body (N or kg/ms^2)

m : mass of object (kg)

a : acceleration (m/s^2)

Therefore, $a = \frac{F}{m}$ or $\frac{dV}{dt} = \frac{F}{m}$ = time rate of change of velocity

V : velocity (m/s)

t : time (s)

If downward is the (+) direction: $F = F_D - F_U$

Falling Parachutist Problem

$F = F_D - F_U$

- F_U → upward force of air resistance = cV
- F_D → downward pull of gravity = mg

g : gravitational constant = 9.8 m/s^2

c : drag coefficient = 12. 5 kg/s

m : mass of parachutist = 70 kg

Then

$$\frac{dV}{dt} = \frac{mg - cV}{m} = g - \frac{c}{m}V$$

Model relating acceleration of falling object to forces acting on it

Falling Parachutist Problem

$$\frac{dV}{dt} = g - \frac{c}{m}V$$

Dependent variable : V(t)
Independent variable : t
Constant parameters : g, c, m

Solving analytically using the initial condition:
If V = 0 @ t = 0 gives:

$$V(t) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right)$$

Falling Parachutist Problem

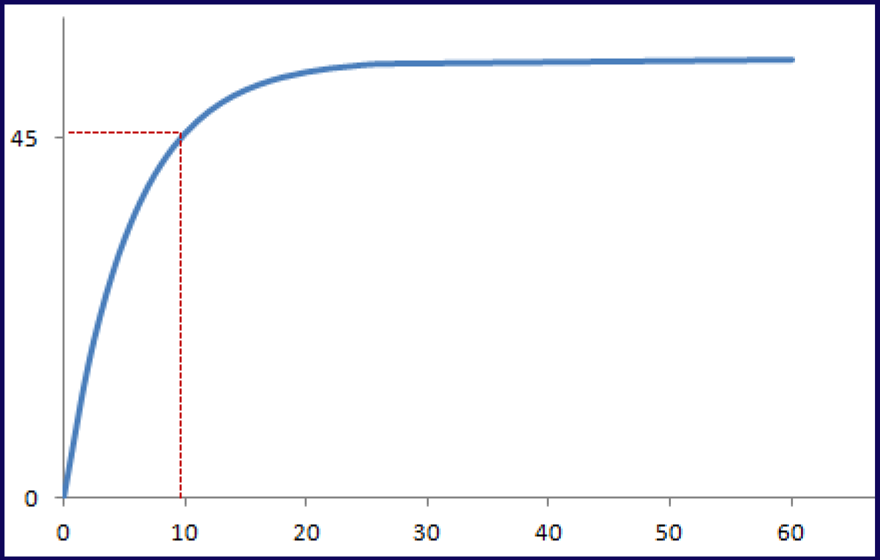
t (s)	V (m/s)
0	0
2	16.4820
4	28.0140
6	36.0826
8	41.7280
10	45.6779
12	48.4415
14	50.3752
16	51.7281
18	52.6747
20	53.3370
22	53.8004
24	54.1246
26	54.3515
.	.
.	.
60	54.8788

Analytical Solution:

$$V(t) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right)$$

$$V(t) = \frac{9.8\,\frac{kg}{ms^2} * 70kg}{12.5\,\frac{kg}{s}}\left(1 - e^{-\frac{12.5}{70}t}\right)$$

V (m/s)



t (s)

As t → ∞

V → Terminal velocity

Falling Parachutist Problem: Numerical

Remember:

$$\frac{dV}{dt} \cong \frac{\Delta V}{\Delta t} = \frac{V(t_{i+1}) - V(t_i)}{t_{i+1} - t_i} = \lim_{\Delta t \to 0} \frac{\Delta V}{\Delta t}$$

"finite divided difference" approximation of derivative at t_i

$$\frac{dV}{dt} = g - \frac{c}{m} V(t_i) = \frac{V(t_{i+1}) - V(t_i)}{t_{i+1} - t_i}$$

↓

So for

$$\underbrace{V(t_{i+1})}_{\text{New value}} = \underbrace{V(t_i)}_{\text{Old value}} + \underbrace{\left[g - \frac{c}{m} V(t_i) \right]}_{\text{slope}} \underbrace{(t_{i+1} - t_i)}_{\text{step size}}$$

If given an initial value for velocity at some time t_i, velocity at a later time t_{i+1} can be called, then, use $V(t_{i+1})$ to call $V(t_{i+2})$

Falling Parachutist Problem: Numerical

New Value = Old Value + (slope * step size)

i = 0, we know that @ $t_{i=0}=0 \rightarrow V_{i=0} = 0$

i = 1, $t_1 = 2$ →

$$V_1 = 0 + \left[9.8 - \frac{12.5}{70}(0)\right] 2 = 19.60\ m/s$$

i = 2, $t_2 = 4$ →

$$V_2 = V_1 + \left[g - \frac{c}{m}V_1\right](t_2 - t_1)$$

$$= 19.60 + \left[9.8 - \frac{12.5}{70}19.60\right] 2 = 32.2\ m/s$$

. .

. .

. .

. .

. .

Homework (for groups of two students)

Write a Matlab code using an iterative technique to calculate the falling velocity of a parachustist of mass 70 kg at any time prior to opening the chute and the terminal velocity. Drag coefficient is 12.5 kg/s and gravitational acceleration is 9.8 m/s^2.

The program should ask the user to enter the time and output should display the following on the command window:

t, V(t), Terminal Velocity, number of iterations used for calculation, error,

The program should tell if the parachutist has reached terminal velocity or not at time t.