Introduction

- Time domain analysis can be realized in the transient regime or in the steady-state regime
- Calculating the transient response of a circuit implies solving a system of differential equations
- A number of methods can be used to calculate the time response of electrical circuits:
  - Linear Multi-Step (LMS) formulae
  - Runge-Kutta (R-K) formulae
  - Harmonic-Balance (HB)
Solving Differential Equations

\[ x' = \frac{dx}{dt} = f(x, t) \quad x(t_0) = x_0 \quad x_i' = f(x_i, t_i) \]

\[ h = t_{i+1} - t_i \]

\[ x_i' \approx \frac{x_{i+1} - x_i}{h} \]

\[ x_{i+1} \approx x_i + hx_i' \]

Forward Euler formula

Solving Differential Equations (cont.)

\[ x_{i+1}' \approx \frac{x_{i+1} - x_i}{h} \]

\[ x_{i+1} \approx x_i + hx_{i+1}' \]

Backward Euler formula
Solving Differential Equations (cont.)

\[
\frac{x'_{i+1} + x'_i}{2} \approx \frac{x_{i+1} - x_i}{h}
\]

\[
x_{i+1} \approx x_i + \frac{h}{2} (x'_{i+1} + x'_i)
\]

Trapezoidal formula

Generalizing to Systems of Differential Eq.

\[x' = f(x, t)\]
\[x(t_0) = x_0\]
\[x'_i = f(x_i, t_i)\]

\[x, x', f \in \mathbb{R}^n\]

- Forward Euler:
  \[x_{i+1} = x_i + hx'_i\]

- Backward Euler:
  \[x_{i+1} = x_i + hx'_{i+1}\]

- Trapezoidal:
  \[x_{i+1} = x_i + \frac{h}{2} (x'_{i+1} + x'_i)\]
Using Predictors and Correctors

- Backward Euler and Trapezoidal formulae require $x'_{i+1}$ to calculate $x_{i+1}$, but this value is not known at the $i$th iteration.
- Forward Euler can be used as a predictor for $x_{i+1}$, which can be later inserted into a corrector using Backward Euler or Trapezoidal formulae.

A Forward Euler Matlab Implementation

```matlab
function [t,x] = F_Euler(fun,x0,t0,tf,h)
N = round((tf-t0)/h); % Number of points.
N = round((tf-t0)/h); % Number of points.
n = length(x0); % Number of variables.
x = zeros(N+1,n); % Solution of the differential equations (an N by n matrix).
t = linspace(t0,tf,N+1); % t: Independent variable (row vector of length N).

x(1,:) = x0; % x0: initial condition (row vector, of length n).
t(1) = t0; % t0: initial value of the independent variable.

i = 1; % h: step used for the independent variable.

while i<=N
    f = feval(fun,x(i,:),t(i)); % Evaluates the differential equations. This function takes two arguments: a row vector x of length n and a scalar independent variable t. It returns a row vector f of length n.
    x(i+1,:) = x(i,:) + h*f; % Backward Euler formula.
    t(i+1) = t(i) + h;
    i = i+1;
end
```

% This function solves a system of n first-order ordinary differential equations $x' = f(x,t)$ using the Forward Euler formula.
% Usage: [t,x] = F_Euler(fun,x0,t0,tf,h)
% t: Independent variable (row vector of length N).
% x: Solution of the differential equations (an N by n matrix).
% fun: name of the multidimensional vector function (string) that evaluates the differential equations. This function takes two arguments: a row vector x of length n and a scalar independent variable t. It returns a row vector f of length n.
% x0: initial condition (row vector, of length n). x0 = x(t=t0).
% t0: initial value of the independent variable.
% tf: final value of the independent variable.
% h: step used for the independent variable.
A Backward Euler Matlab Implementation

```matlab
function [t,x] = B_Euler(fun,x0,t0,tf,h)
N = round((tf-t0)/h); % Number of points.
n = length(x0);
x = zeros(N+1,n);
t = zeros(1,N+1);
x(1,:) = x0;
t(1) = t0;
i = 1;
while i<=N
    f = feval(fun,x(i,:),t(i));
    xp = x(i,:) + h*f;
    t(i+1) = t(i) + h;
    fn = feval(fun,xp,t(i+1));
    x(i+1,:) = x(i,:) + h*fn;
    i = i+1;
end
```

A Trapezoidal Matlab Implementation

```matlab
function [t,x] = Trapezoidal(fun,x0,t0,tf,h)
N = round((tf-t0)/h); % Number of points.
n = length(x0);
x = zeros(N+1,n);
t = zeros(1,N+1);
x(1,:) = x0;
t(1) = t0;
i = 1;
while i<=N
    f = feval(fun,x(i,:),t(i));
    xp = x(i,:) + h*f;
    t(i+1) = t(i) + h;
    fn = feval(fun,xp,t(i+1));
    x(i+1,:) = x(i,:) + (h/2)*(f+fn);
    i = i+1;
end
```
Example 1 - Forward Euler

\[ x' = x + t^2 \]
\[ x_0 = 1 \]
\[ h = 0.025 \]

Exact solution:
\[ x = 3e^t - t^2 - 2t - 2 \]

Example 1 - Backward Euler

\[ x' = x + t^2 \]
\[ x_0 = 1 \]
\[ h = 0.025 \]

Exact solution:
\[ x = 3e^t - t^2 - 2t - 2 \]
Example 1 - Trapezoidal

\[ x' = x + t^2 \]
\[ x_0 = 1 \]
\[ h = 0.025 \]

Exact solution:
\[ x = 3e^t - t^2 - 2t - 2 \]

---

Example 1 - Comparison

\[ x' = x + t^2 \]
\[ x_0 = 1 \]
\[ h = 0.025 \]

Exact solution:
\[ x = 3e^t - t^2 - 2t - 2 \]
Example 1 - Comparison (Decreasing $h$)

$x' = x + t^2$
$x_0 = 1$
$h = 0.01$

Exact solution:
$x = 3e^t - t^2 - 2t - 2$

Example 2 - Forward Euler

$x' = -40x$
$x_0 = 10$
$h = 0.01$

Exact solution:
$x = 10e^{-40t}$
Example 2 - Backward Euler

\[ x' = -40x \]
\[ x_0 = 10 \]
\[ h = 0.01 \]

Exact solution:
\[ x = 10e^{-40t} \]

Example 2 - Trapezoidal

\[ x' = -40x \]
\[ x_0 = 10 \]
\[ h = 0.01 \]

Exact solution:
\[ x = 10e^{-40t} \]
Example 2 - Comparison

\[ x' = -40x \]
\[ x_0 = 10 \]
\[ h = 0.01 \]

Exact solution:
\[ x = 10e^{-40t} \]

![Graph comparing methods](image)

Example 2 - Comparison (Decreasing \( h \))

\[ x' = -40x \]
\[ x_0 = 10 \]
\[ h = 0.001 \]

Exact solution:
\[ x = 10e^{-40t} \]

![Graph comparing methods](image)
Stability of Integration

- Test differential equation: \( x' = \lambda x \) solution: \( x = x_0 e^{\lambda t} \)
- Forward Euler:
  \[
  x_i = (1 + \lambda h) x_0 \quad |1 + \lambda h| \leq 1
  \]
- Backward Euler:
  \[
  x_i = \left( \frac{1}{1 - \lambda h} \right)^i x_0 \quad \left| \frac{1}{1 - \lambda h} \right| \leq 1
  \]
- Trapezoidal:
  \[
  x_i = \left( \frac{1 + \lambda h / 2}{1 - \lambda h / 2} \right)^i x_0 \quad \left| \frac{1 + \lambda h / 2}{1 - \lambda h / 2} \right| \leq 1
  \]

Linear Systems of Differential Equations

\[
x' = f(x, t) \quad x(t = 0) = x(t_0) = x_0 \quad x'_i = f(x_i, t_i)
\]
\[
x' = Bx + w(t)
\]

Backward Euler:
\[
x_{i+1} = x_i + h x'_{i+1} = x_i + h (Bx_{i+1} + w_{i+1})
\]
\[
(1 - hB)x_{i+1} = x_i + hw_{i+1}
\]

Trapezoidal:
\[
x_{i+1} = x_i + \frac{h}{2} (x'_{i+1} + x'_i) = x_i + \frac{h}{2} (Bx_{i+1} + w_{i+1} + Bx_i + w_i)
\]
\[
(1 - \frac{h}{2} B)x_{i+1} = (1 + \frac{h}{2} B)x_i + \frac{h}{2} (w_{i+1} + w_i)
\]
Transient Solution of Linear Circuits

Let the system equations in the Laplace domain be

\[(H_1 + sH_2)X = W\]

Taking the inverse Laplace Transform,

\[H_1 x + H_2 x' = w\]
\[H_2 x' = w - H_1 x\]

Transient Solution of Linear Circuits (cont.)

\[H_2 x' = w - H_1 x\]

Forward Euler:

\[x_{i+1} = x_i + hx'_i\]
\[H_2 x_{i+1} = H_2 x_i + hH_2 x'_i\]
\[H_2 x_{i+1} = H_2 x_i + h(w_i - H_1 x_i)\]
\[H_2 x_{i+1} = (H_2 - hH_1) x_i + hw_i\]

\[(H_2 \text{ might be singular})\]
Transient Solution of Linear Circuits (cont.)

\[ H_2 x' = w - H_1 x \]

Backward Euler:

\[
\begin{align*}
    x_{i+1} &= x_i + h x'_{i+1} \\
    H_2 x_{i+1} &= H_2 x_i + h H_2 x'_{i+1} \\
    (H_2 + h H_1) x_{i+1} &= H_2 x_i + h w_{i+1}
\end{align*}
\]

Trapezoidal:

\[
\begin{align*}
    x_{i+1} &= x_i + \frac{h}{2} (x'_{i+1} + x'_{i}) \\
    H_2 x_{i+1} &= H_2 x_i + \frac{h}{2} (H_2 x'_{i+1} + H_2 x'_{i}) \\
    H_2 x_{i+1} &= H_2 x_i + \frac{h}{2} (w_{i+1} - H_1 x_{i+1} + w_i - H_1 x_i) \\
    (H_2 + \frac{h}{2} H_1) x_{i+1} &= (H_2 - \frac{h}{2} H_1) x_i + \frac{h}{2} (w_{i+1} + w_i)
\end{align*}
\]
MNA Formulation for Transient Analysis

- The MNA equation
  \[
  \begin{bmatrix}
  A_1YA_1^T & A_1 \\
  Y_2A_2^T & Z_2
  \end{bmatrix}
  \begin{bmatrix}
  V_n \\
  I_2
  \end{bmatrix}
  =
  \begin{bmatrix}
  -A_1J_1 \\
  W_2
  \end{bmatrix}
  \quad HX = W
  \]
  can be formulated without using oriented graphs or incidence matrices \(A_1\) and \(A_2\)

- \(H\) and \(W\) can be directly formulated by inspection, using stamps

- Once \(H\) is known, the L and C elements can be separated so that
  \[
  (H_1 + sH_2)X = W
  \]
  which can be solved in the time domain using the previous methods