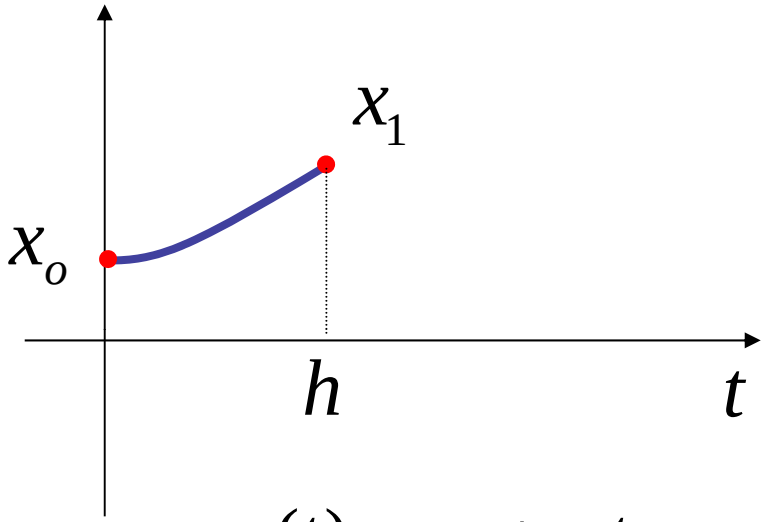
Transient Analysis-II

Designing Your Own Integration Formula



Generating polynomial of order 1:

$$x(t) = a_o + a_1 t \Rightarrow x_1 = a_0 + a_1 h$$

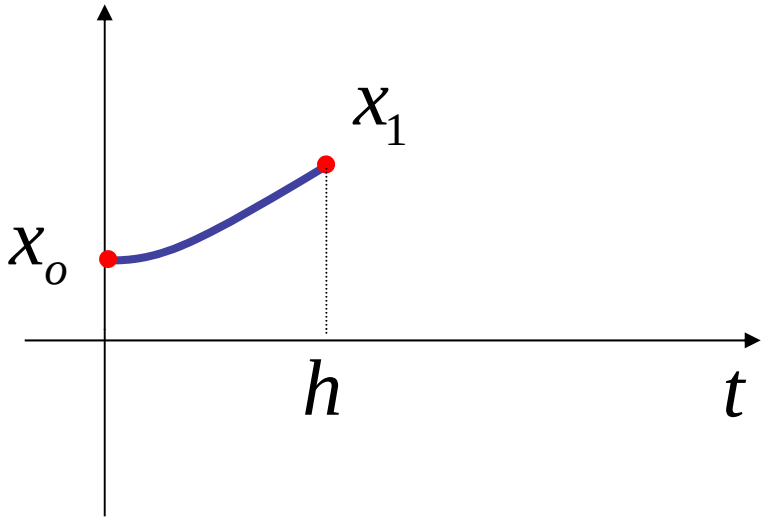
to find a_0 *and* a_1 :

$$x_o = x(t=0) = a_o \text{.....(1)}$$

$$\dot{x}_o = \dot{x}(t=0) = a_1 \text{.....(2)}$$

$$x_1 = x(t=h) = x_o + \dot{x}_o h$$ ➡ Forward Euler!!

Designing Your Own Integration Formula



Generating polynomial of order 1:

$$x(t) = a_o + a_1 t \Rightarrow x_1 = a_0 + a_1 h$$

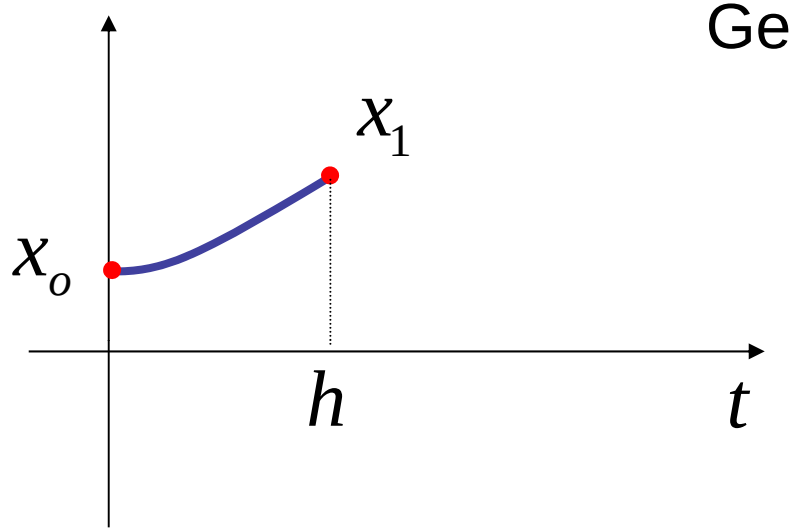
$$to\ \ find\ a_0\ and\ a_1:$$

$$x_o = x(t=0) = a_o \text{.....(1)}$$

$$\dot{x}_1 = \dot{x}(t=h) = a_1 \text{.....(2)}$$

$$x_1 = x(t=h) = x_o + \dot{x}_1 h$$ ➡ Backward Euler.

Second Order Methods



Generating polynomial of order 2:

$$x(t) = a_o + a_1 t + a_2 t^2$$

$$\dot{x}(t) = a_1 + 2a_2 t$$

$$\begin{cases} x_o = x(t=0) = a_o & ① \\ \dot{x}_o = \dot{x}(t=0) = a_1 & ② \\ \dot{x}_1 = \dot{x}(t=h) = a_1 + 2a_2 h & ③ \end{cases}$$

Second Order Methods

$$\begin{cases} x_o = x(t=0) = a_o & \text{①} \\ \dot{x}_o = \dot{x}(t=0) = a_1 & \text{②} \\ \dot{x}_1 = \dot{x}(t=h) = a_1 + 2a_2 h & \text{③} \end{cases}$$

① → $x_o = a_o$

② → $\dot{x}_o = a_1$

③ & ② → $a_2 = \dfrac{\dot{x}_1 - a_1}{2h} = \dfrac{\dot{x}_1 - \dot{x}_o}{2h}$

Second Order Methods

① → $x_o = a_o$

② → $\dot{x}_o = a_1$

③ & ② → $a_2 = \dfrac{\dot{x}_1 - a_1}{2h} = \dfrac{\dot{x}_1 - \dot{x}_o}{2h}$

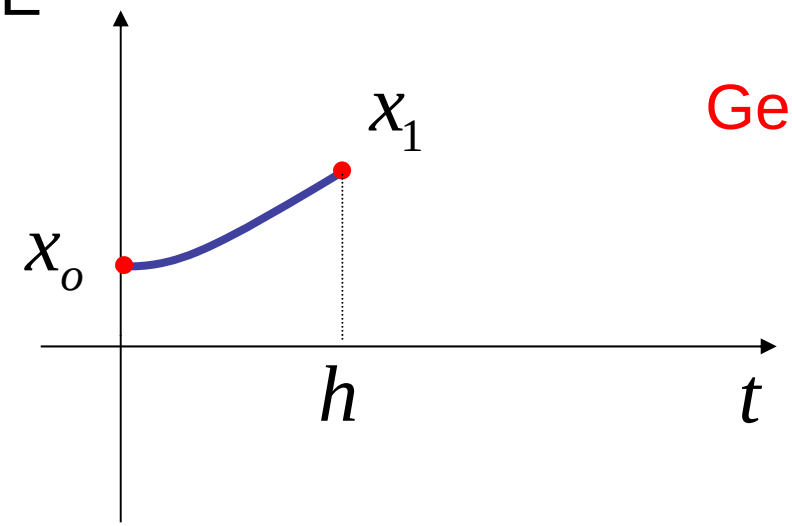
$$x(t) = a_o + a_1 t + a_2 t^2$$

$$x_1 = x(t = h) = a_o + a_1 h + a_2 h^2 = x_o + \dot{x}_o h + \left(\frac{\dot{x}_1 - \dot{x}_o}{2h} \right) h^2$$

$$x_1 = x_o + \frac{h}{2}\left(\dot{x}_0 + \dot{x}_1\right) \quad \Rightarrow \quad \text{Trapezoidal Rule}$$

Order of the Integration Formula

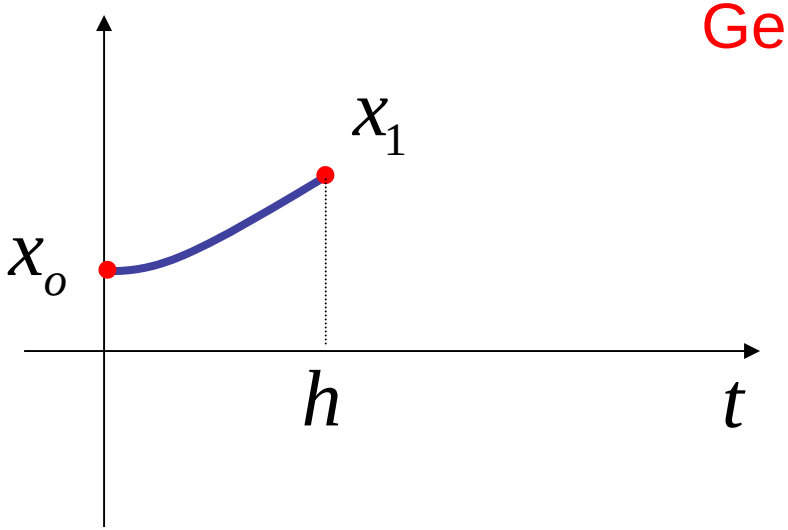
FE,BE



Generating polynomial of order 1:

$$x(t) = a_o + a_1 t$$

TR



Generating polynomial of order 2:

$$x(t) = a_o + a_1 t + a_2 t^2$$

Local Truncation Error

First Order Methods

$$x_1 = x(t = h) = a_o + a_1 h$$

Second Order Methods

$$x_1 = x(t = h) = a_o + a_1 h + a_2 h^2$$

For forward and backward Euler (1st order):

→ L.T.E proportional to h^2

For Trapezoidal Rule (2st order):

→ L.T.E proportional to h^3

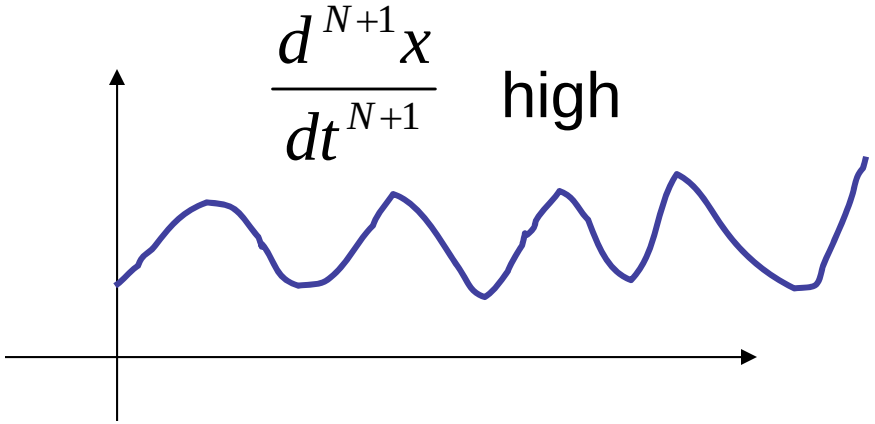
Local Truncation Error

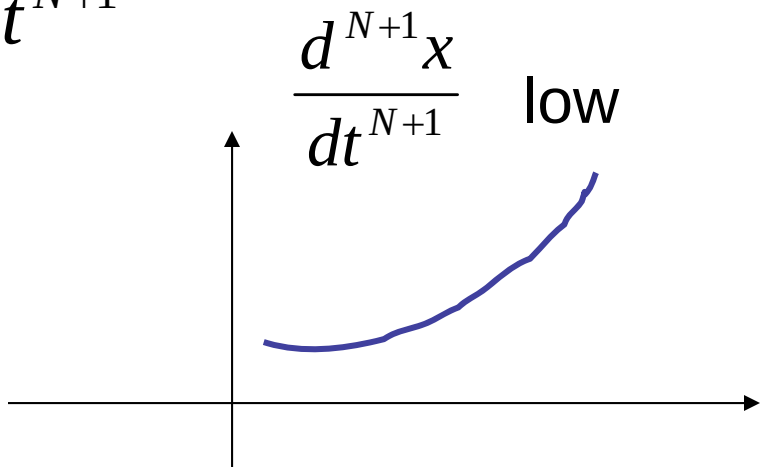
Integration method were derived from a truncated Taylor approximation of the function

→ Truncation error is derived from Taylor theory

→ For a method of order N

$$LTE \cong Ch^{N+1}\frac{d^{N+1}x}{dt^{N+1}}$$





Local Truncation Error

Backward Euler:

$$LTE \cong \frac{1}{2} h^2 \frac{d^2 x}{dt^2}$$

Trapezoidal Rule:

$$LTE \cong \frac{-1}{12} h^3 \frac{d^3 x}{dt^3}$$

Local Truncation Error

Example: Backward Euler

$$\text{if} \quad h = 0.1 \quad \text{error} \approx 10^{-2}, \ \text{find } h \text{ to} \quad \text{achieve} \quad \text{error} \approx 10^{-4}$$

$$LTE \cong \frac{1}{2} h^2 \frac{d^2 x}{dt^2}$$

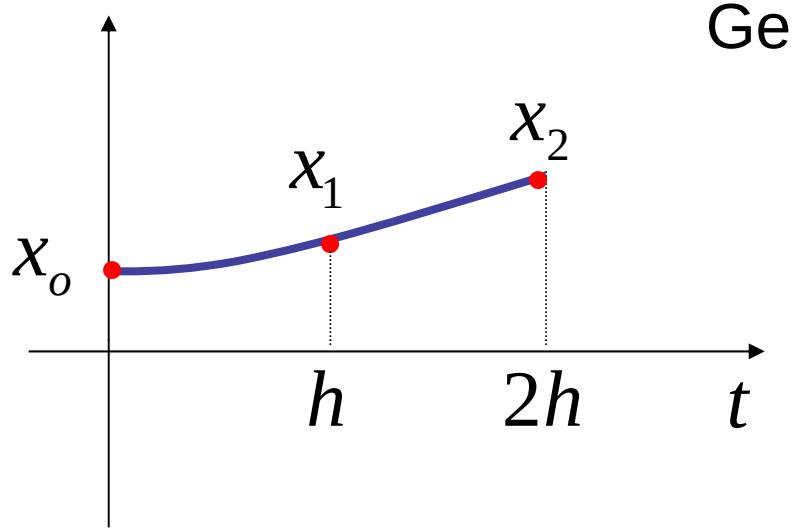
$$10^{-2} = k\left(0.1\right)^2 \ldots\ldots(1)$$

$$10^{-4} = k\left(h\right)^2 \ldots\ldots\ldots(2)$$

$$\left(\frac{h}{0.1}\right)^2 = \frac{10^{-4}}{10^{-2}}$$

$$h = 0.01$$

Multi-Step Methods



Generating polynomial of order 2:

$$x(t) = a_o + a_1 t + a_2 t^2$$

$$\dot{x}(t) = a_1 + 2a_2 t$$

$$\begin{cases} x_o = x(t=0) = a_o & ① \\ \dot{x}_o = \dot{x}(t=0) = a_1 & ② \\ \dot{x}_2 = \dot{x}(t=2h) = a_1 + 4a_2 h & ③ \end{cases}$$

Multi-Step Methods

$$\begin{cases} x_o = x(t=0) = a_o & ① \\ \dot{x}_o = \dot{x}(t=0) = a_1 & ② \\ \dot{x}_2 = \dot{x}(t=2h) = a_1 + 4a_2 h & ③ \end{cases} \rightarrow$$

$$a_2 = \frac{\dot{x}_2 - a_1}{4h} = \frac{\dot{x}_2 - \dot{x}_o}{4h}$$

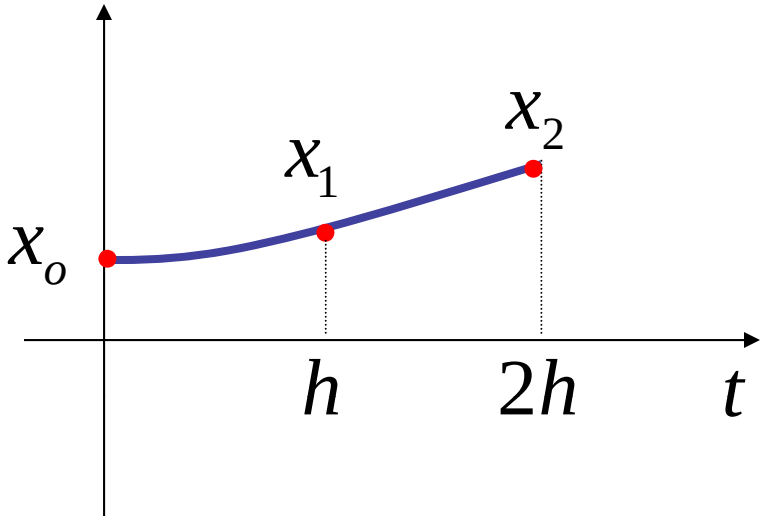
$$x(t) = a_o + a_1 t + a_2 t^2$$

$$x_2 = x(t=2h) = x_o + 2a_1 h + 4a_2 h^2$$

$$= x_o + 2\dot{x}_o h + 4a_2 h^2 = x_o + 2\dot{x}_o h + 4\left(\frac{\dot{x}_2 - \dot{x}_o}{4h}\right)h^2$$

$$\boxed{x_2 = x_o + \dot{x}_o h + \dot{x}_2 h}$$

Gear's Method



Generating polynomial of order 2:

$$x(t) = a_o + a_1 t + a_2 t^2$$

$$\dot{x}(t) = a_1 + 2a_2 t$$

Derivative ***only*** at current point

$$\begin{cases} x_o = x(t=0) = a_o & ① \\ x_1 = x(t=h) = a_o + a_1 h + a_2 h^2 & ② \\ \dot{x}_2 = \dot{x}(t=2h) = a_1 + 4a_2 h & ③ \end{cases}$$

Gear's Method

$$\begin{cases} x_o = x(t=0) = a_o & ① \\ x_1 = x(t=h) = a_o + a_1 h + a_2 h^2 & ② \\ \dot{x}_2 = \dot{x}(t=2h) = a_1 + 4a_2 h & ③ \end{cases}$$

① & ② $\quad x_1 = x_o + a_1 h + a_2 h^2$

③ $\quad \dot{x}_2 = a_1 + 4a_2 h \longrightarrow a_2 = \dfrac{\dot{x}_2 - a_1}{4h}$

$$x_1 = x_o + a_1 h + \left(\frac{\dot{x}_2 - a_1}{4h}\right) h^2 = x_o + \frac{h}{4}\dot{x}_2 + \frac{3h}{4} a_1$$

Gear's Method

$$x_1 = x_o + \frac{h}{4}\dot{x}_2 + \frac{3h}{4}a_1$$

$$a_1 = \frac{4}{3h}\left(-x_o + x_1 - \frac{h}{4}\dot{x}_2\right) = \frac{-4}{3h}x_o + \frac{4}{3h}x_1 - \frac{1}{3}\dot{x}_2$$

$$a_2 = \frac{\dot{x}_2 - a_1}{4h} = \frac{1}{4h}\left(\dot{x}_2 + \frac{4}{3h}x_o - \frac{4}{3h}x_1 + \frac{1}{3}\dot{x}_2\right)$$

$$a_2 = \frac{1}{3h^2}x_o - \frac{1}{3h^2}x_1 + \frac{1}{3h}\dot{x}_2$$

Gear's Method

$$\begin{cases} a_o = x_o \\ a_1 = \dfrac{-4}{3h} x_o + \dfrac{4}{3h} x_1 - \dfrac{1}{3} \dot{x}_2 \\ a_2 = \dfrac{1}{3h^2} x_o - \dfrac{1}{3h^2} x_1 + \dfrac{1}{3h} \dot{x}_2 \end{cases}$$

$$x(t) = a_o + a_1 t + a_2 t^2$$

$$x_2 = x(t = 2h) = a_o + 2a_1 h + 4a_2 h^2$$

$$= x_o + 2h\left(\frac{-4}{3h} x_o + \frac{4}{3h} x_1 - \frac{1}{3} \dot{x}_2 \right) + 4h^2 \left(\frac{1}{3h^2} x_o - \frac{1}{3h^2} x_1 + \frac{1}{3h} \dot{x}_2 \right)$$

Gear's Method

$$x_2 = x_o + 2h\left(\frac{-4}{3h}x_o + \frac{4}{3h}x_1 - \frac{1}{3}\dot{x}_2\right) + 4h^2\left(\frac{1}{3h^2}x_o - \frac{1}{3h^2}x_1 + \frac{1}{3h}\dot{x}_2\right)$$

$$x_2 = -\frac{1}{3}x_o + \frac{4}{3}x_1 + \frac{2h}{3}\dot{x}_2$$

Derivative at *current* time point *only*

→ Gear's method

Example

Gear's Method :

Prove that

$$\left(\mathbf{G}+\frac{3\mathbf{C}}{2h}\right)\mathbf{x}_{n+2}=\frac{2\mathbf{C}}{h}\mathbf{x}_{n+1}-\frac{\mathbf{C}}{2h}\mathbf{x}_{n}+\mathbf{b}_{n+2}$$

$$\mathbf{x}_0=\begin{bmatrix}0\\0\end{bmatrix} \qquad \mathbf{x}_1=\begin{bmatrix}0.1875\\0.125\end{bmatrix}$$

Find $\mathbf{x}_2$ Compare with BE

Linear Multistep Methods

$$x_{n+1} = x_n + \tfrac{h}{2}\left(x'_n + x'_{n+1}\right) \qquad \textit{implicit} \quad (\textit{one step})$$

$$x_{n+1} = x_n + h x'_n \qquad \textit{explicit} \quad (\textit{one step})$$

$$x_{n+1} = x_n + \tfrac{h}{2}\left(3x'_n - x'_{n-1}\right) \qquad \textit{explicit} \quad (\textit{two steps})$$

$$x_{n+1} = x_n + \tfrac{h}{12}\left(5x'_{n+1} + 8x'_n - x'_{n-1}\right) \qquad \textit{implicit} \quad (\textit{two steps})$$

$$x_2 = -\frac{1}{3}x_o + \frac{4}{3}x_1 + \frac{2h}{3}\dot{x}_2 \qquad \textit{implicit} \quad (\textit{two steps})$$

Linear Multistep Methods (LMS)

$$x_{n+1} = x_n + \frac{h}{2}\left(x'_n + x'_{n+1}\right) \qquad Order \quad 2$$

$$x_{n+1} = x_n + h x'_n \qquad Order \quad 1$$

$$x_{n+1} = x_n + \frac{h}{2}\left(3x'_n - x'_{n-1}\right) \qquad Order \quad 2$$

$$x_{n+1} = x_n + \frac{h}{12}\left(5x'_{n+1} + 8x'_n - x'_{n-1}\right) \qquad Order \quad 3$$

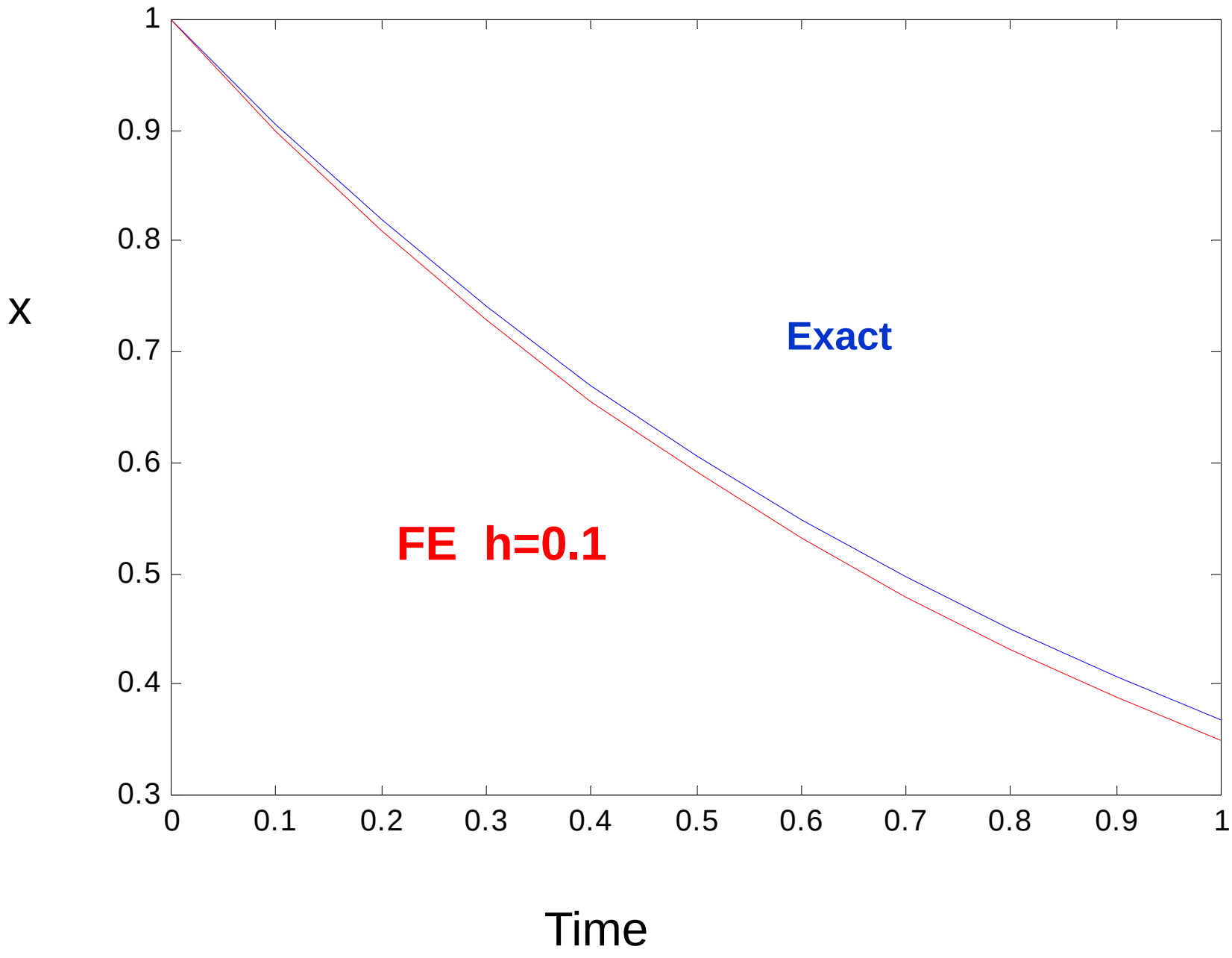
$$x_2 = -\frac{1}{3}x_o + \frac{4}{3}x_1 + \frac{2h}{3}\dot{x}_2 \qquad Order \quad 2$$

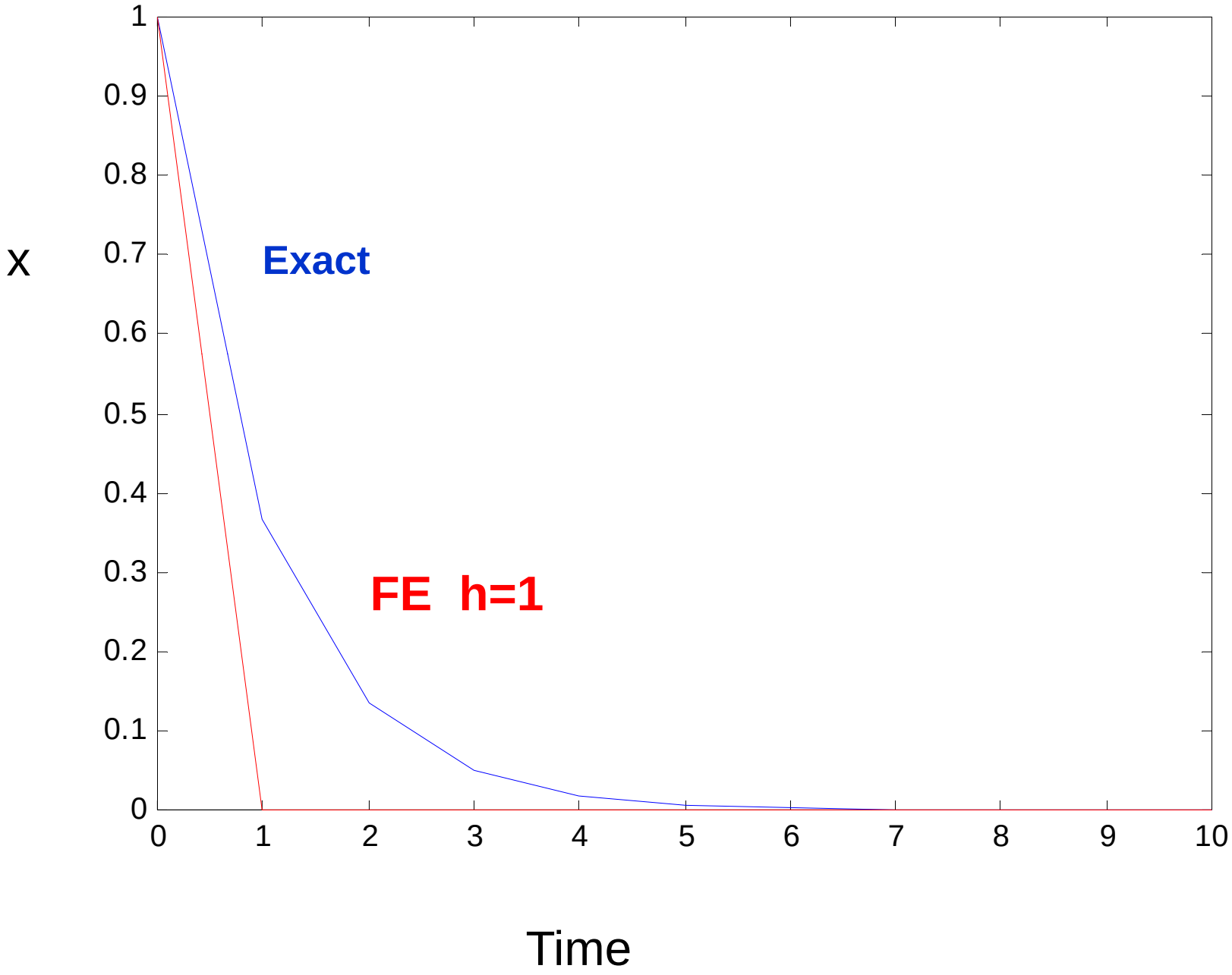
Note: LMS are also called in the literature "Backward differentiation formulas"

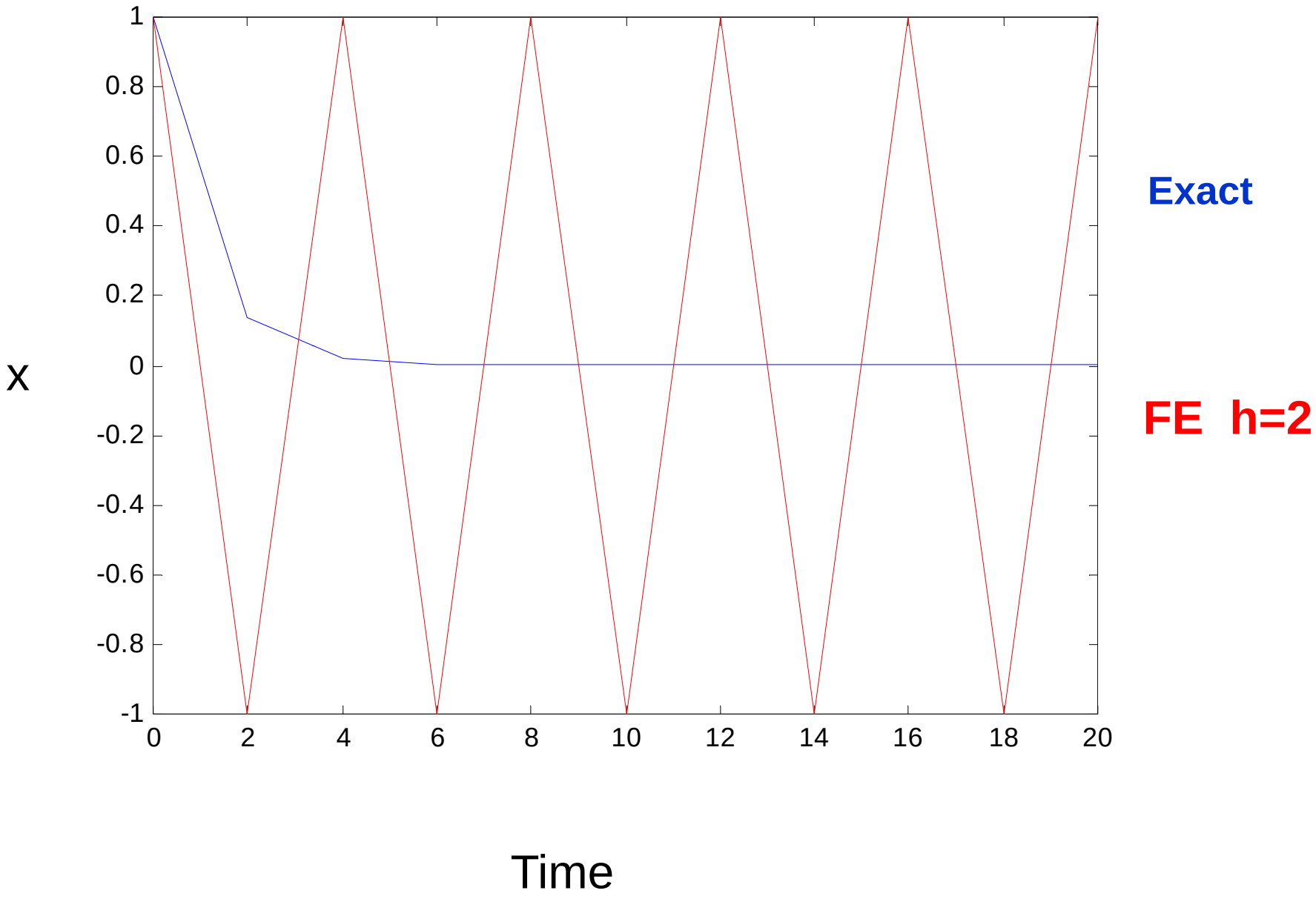
Numerical Stability of Forward-Euler (FE) is illustrated by the simple example ;

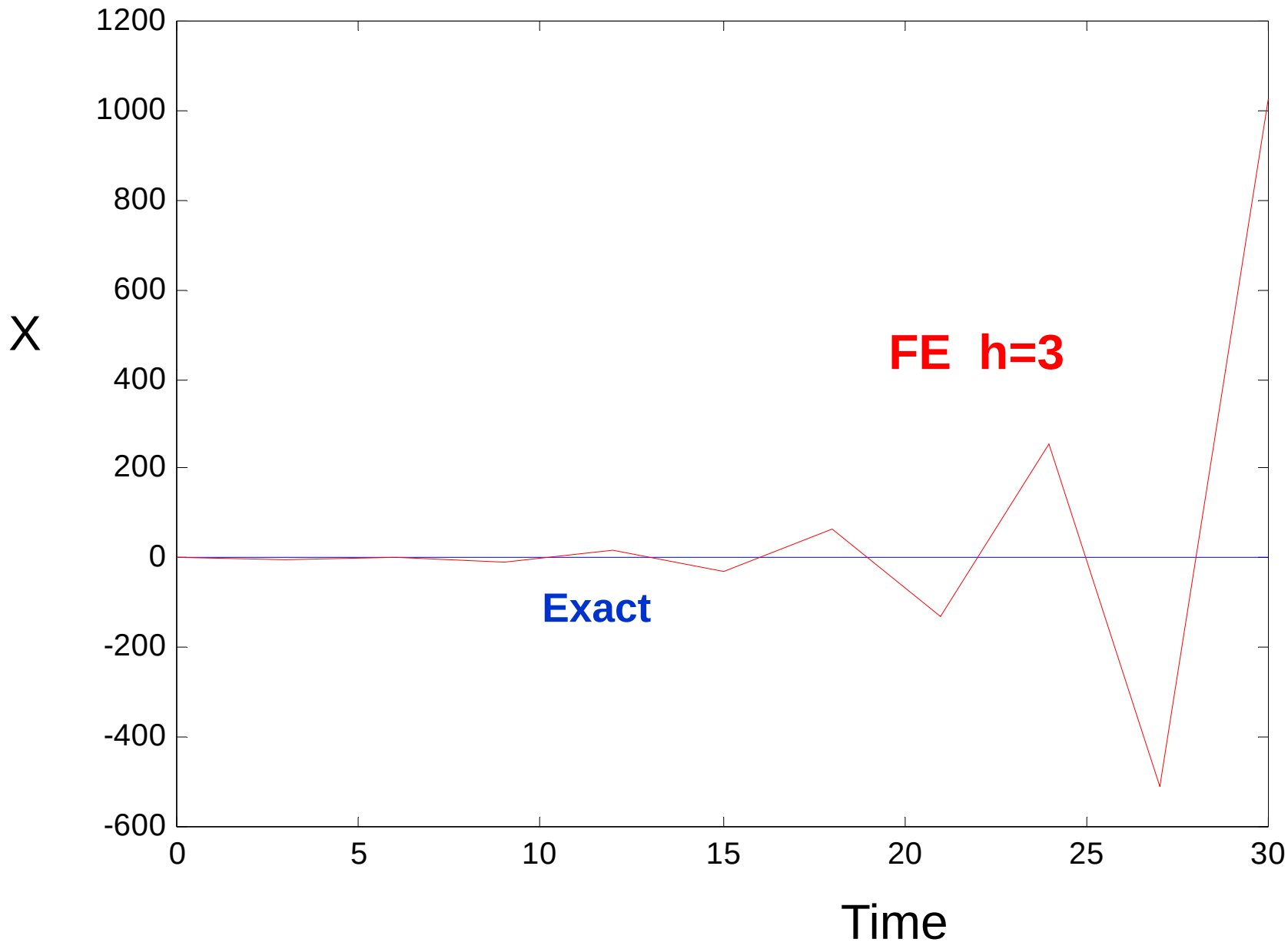
$dx/dt=-x$ with $x(0)=1$

Exact solution $x=\exp(-t)$ is compared with FE for different values of the step size h.

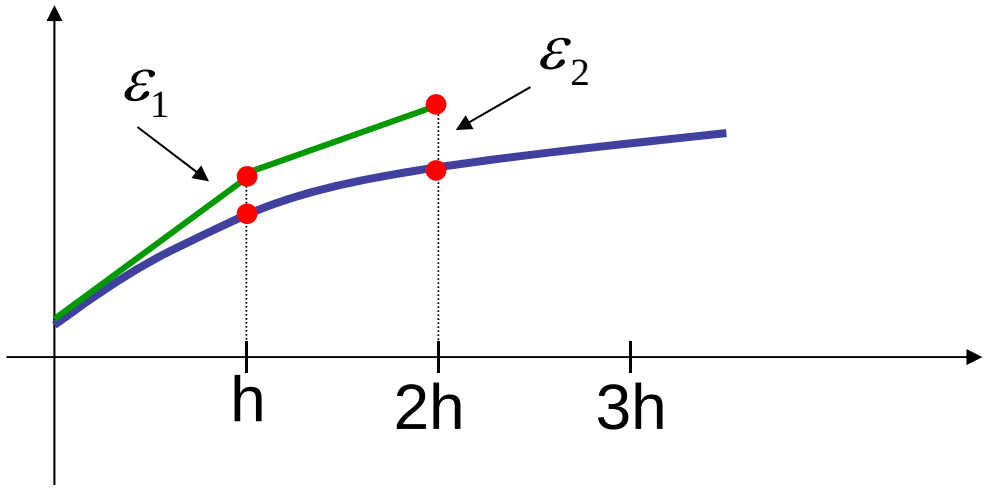

x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Exact
FE h=0.1
Time


x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1
2
3
4
5
6
7
8
9
10
Exact
FE h=1
Time


x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
2
4
6
8
10
12
14
16
18
20
Time
Exact
FE h=2


1200
1000
800
600
400
200
0
-200
-400
-600
X
0
5
10
15
20
25
30
Time
FE h=3
Exact

Error Mechanisms



ε_1 = Local Truncation Error (L.T.E)

ε_2 = L.T.E + effects of ε_1

< ε_1
Numerically stable

> ε_1
unstable

Numerical Stability

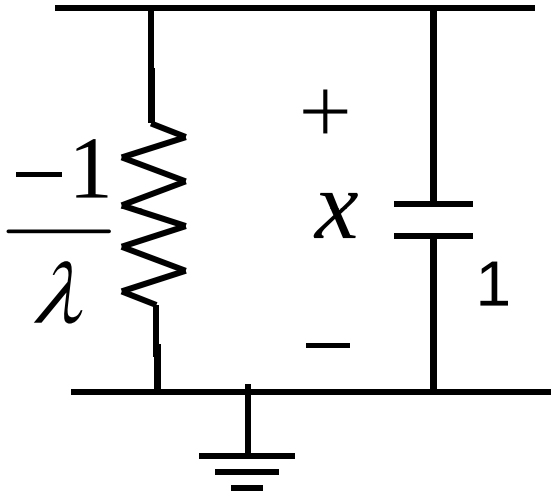
1. Numerical stability related to the *propagation* of the local truncation error. If the LTE is monotonously decreasing then the approximation is numerically stable. If it is monotonously increasing then the approximation is unstable.
2. Numerical stability is a function of the differential equation, the step size, and the integration formula.

Stability Test

How to test an integration formula for stability.

Test equation:

$$\dot{x} = \lambda x$$



Apply Forward Euler

$$x_{n+1} = x_n + h\dot{x}_n$$

$$x_{n+1} = x_n + h\lambda x_n = (1 + h\lambda)x_n$$

Stability Test (Forward Euler)

$$x_{n+1} = (1 + h\lambda)x_n$$

$$t = 0 \longrightarrow x_o$$

$$t = h \longrightarrow x_1 = (1 + h\lambda)x_0$$

$$t = 2h \longrightarrow x_2 = (1 + h\lambda)x_1 = (1 + h\lambda)^2 x_o$$

$$\vdots$$

$$t = nh \longrightarrow x_n = (1 + h\lambda)^n x_o$$

Stability Test (Forward Euler)

$$x_n = (1 + h\lambda)^n x_o$$

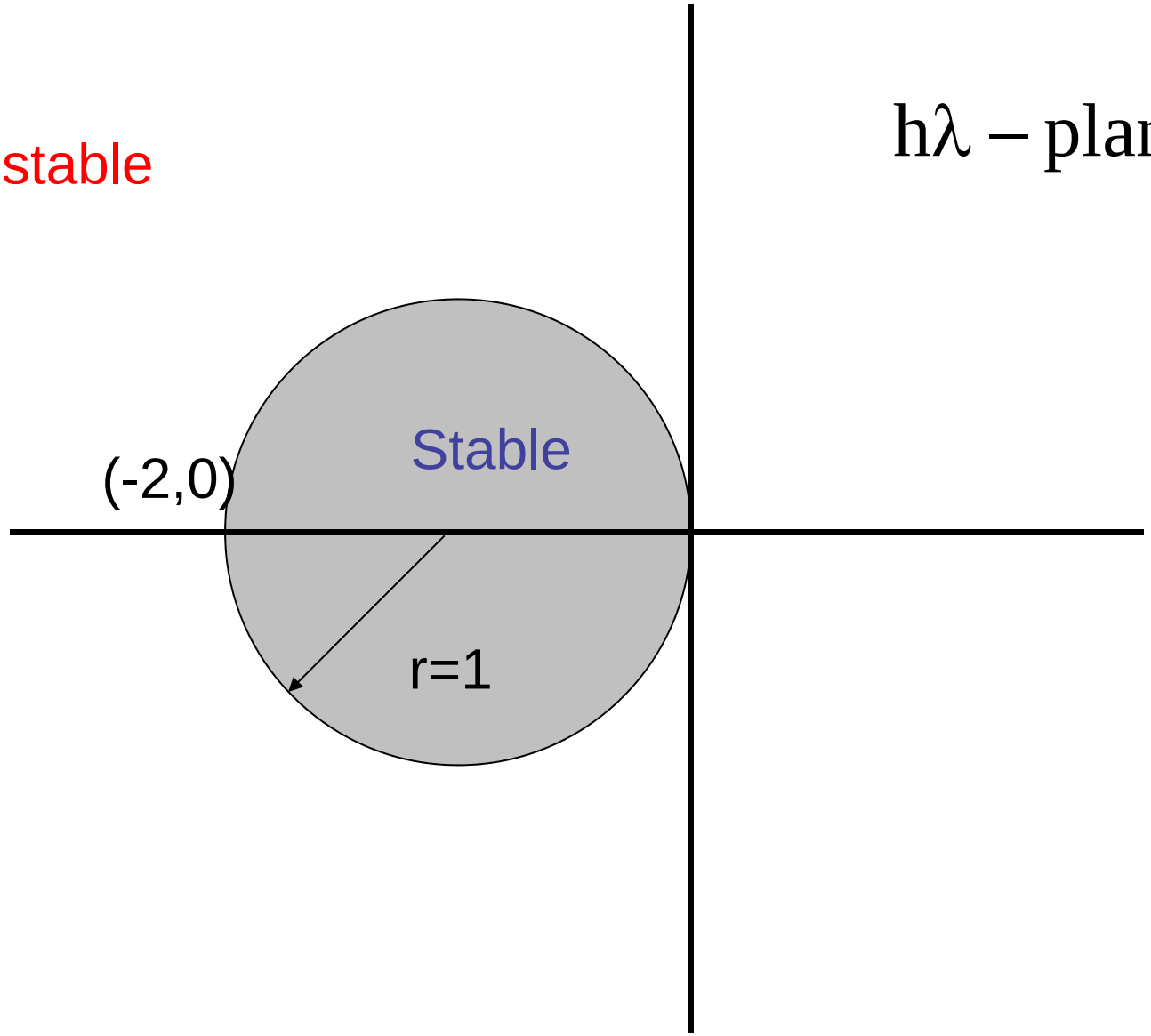
Conditions for stability:

$$n \to \infty \Rightarrow x_n \to 0$$

➔ $\lambda < 0$ i.e. system has stable poles

➔ $|1 + h\lambda| < 1$

Numerical Stability-FE



System with Multiple Poles

➔ All poles must satisfy stability condition.

Example: $\lambda_1 = -1$

$\lambda_2 = -1000 \longrightarrow$ Stray pole

→ Stiff system (widely separated poles)

→ For Forward Euler

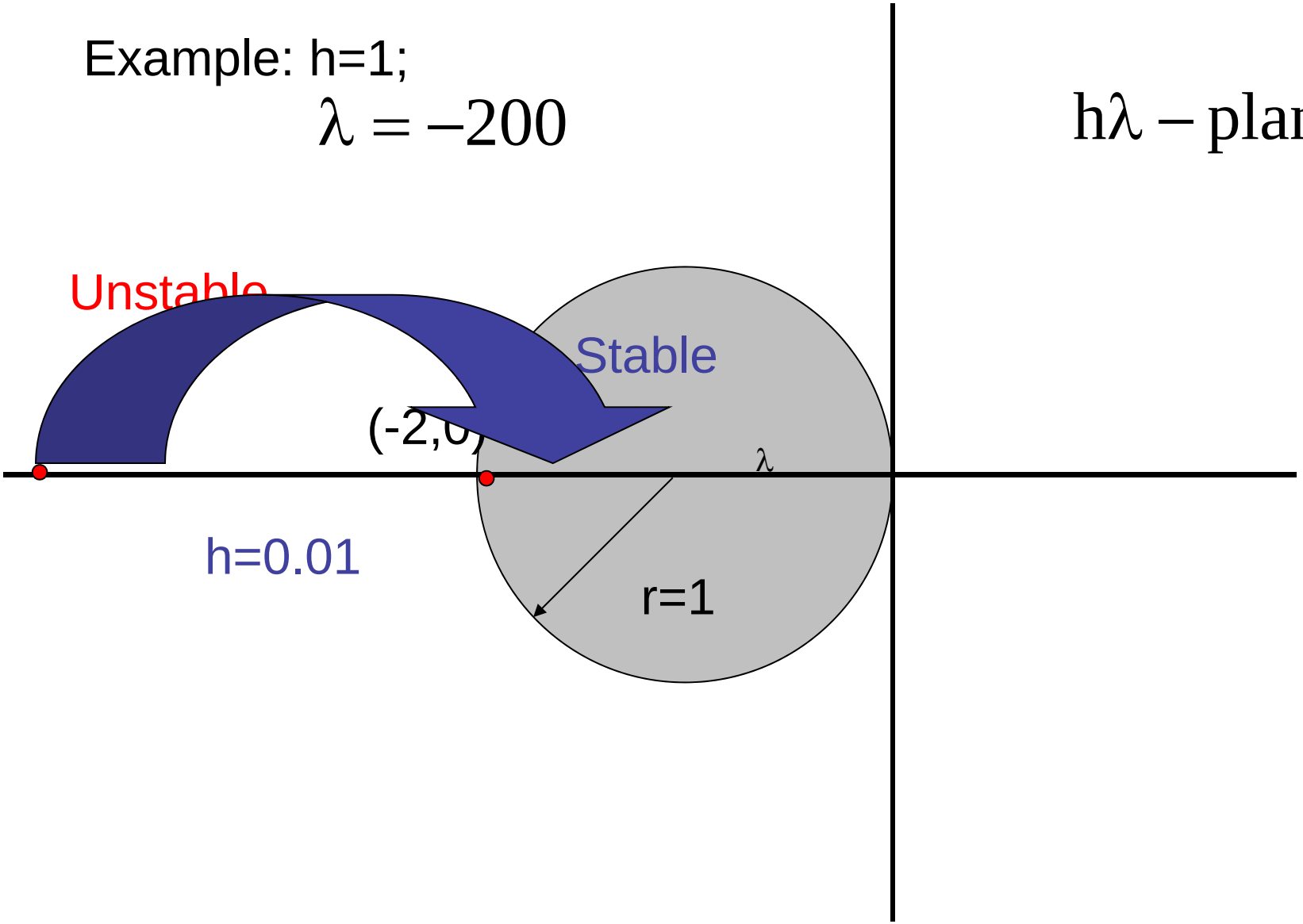
$$\left.\begin{array}{l} \lambda_1 = -1 \rightarrow h < 2 \\ \lambda_2 = -1000 \rightarrow h < 0.002 \end{array}\right\} \quad h < 0.002$$

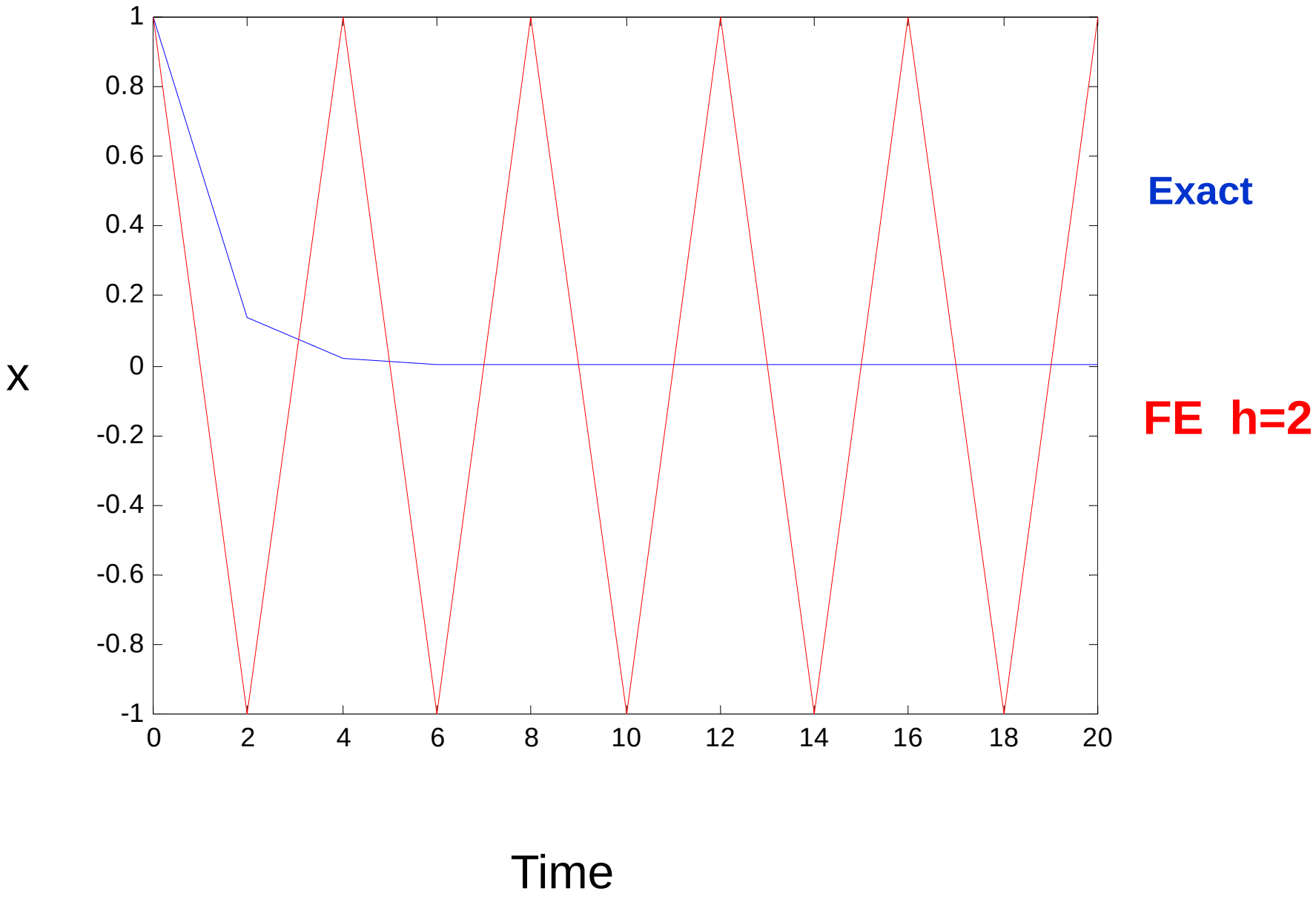
Numerical Stability-FE

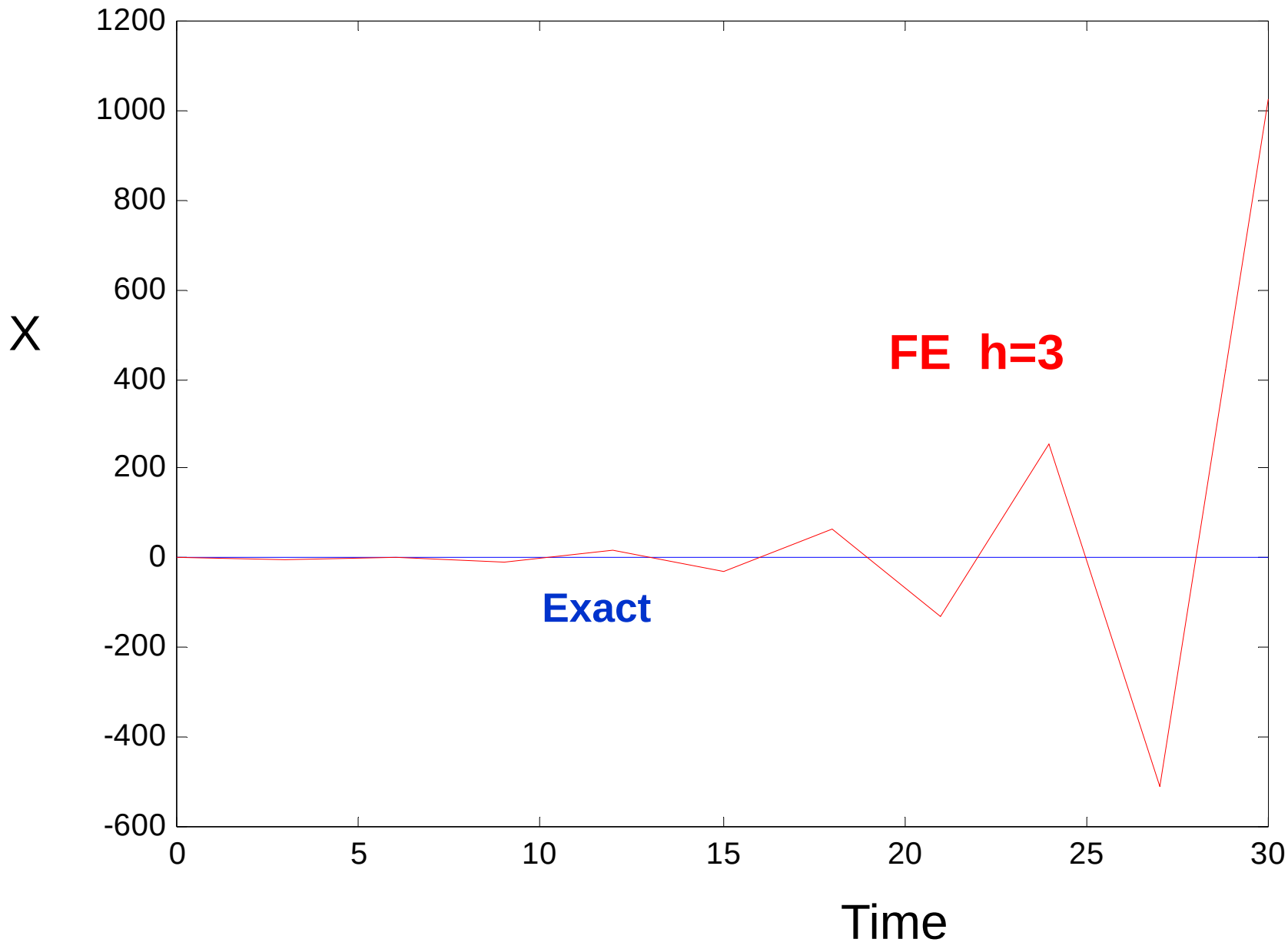
Example: h=1;

$$\lambda = -200$$

$h\lambda$ – plane

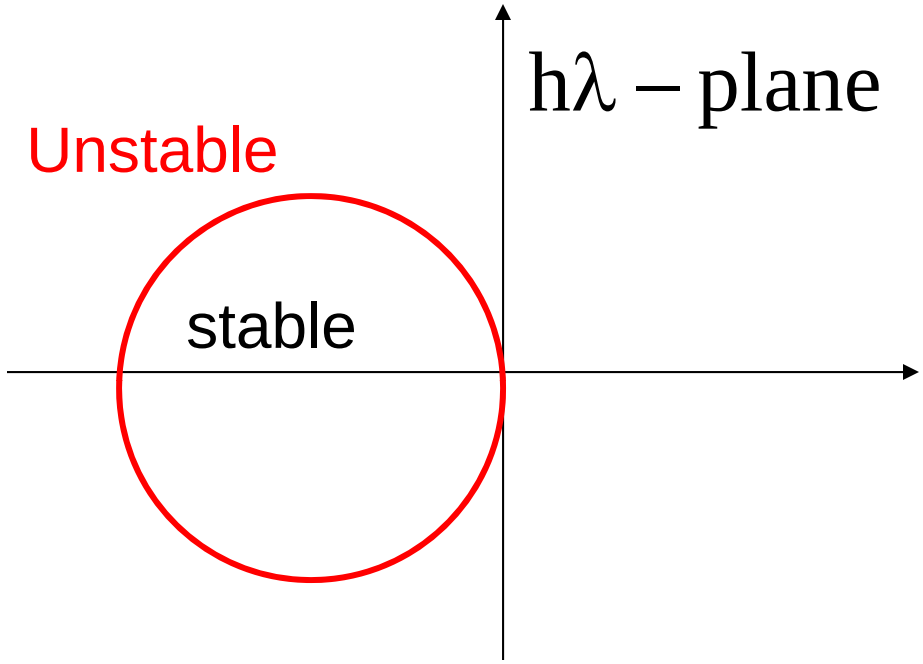



x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
2
4
6
8
10
12
14
16
18
20
Time
Exact
FE h=2


1200
1000
800
600
400
200
0
-200
-400
-600
X
0
5
10
15
20
25
30
Time
FE h=3
Exact

A- Stability

Absolute Stability A-Stability:
A numerical integration method is A-Stable if all its solutions tend to zero for all values of h when the method is applied to a differential equation of the form



$$\frac{dx}{dt} = \lambda x \qquad \mathrm{Re}[\lambda] < 0$$

i.e. the method has to be stable in the whole left half-plane

Forward Euler is not A-stable

Dalquist's Barriers

Germund Dahlquist
1925-2005

- All explicit LMS are not absolutely stable

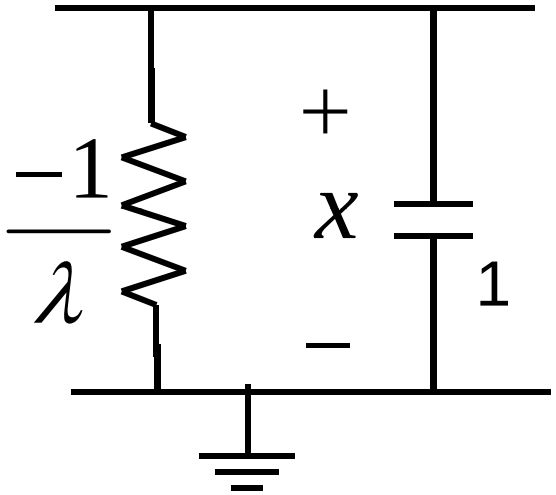
  - Are all implicit LMS methods absolutely stable?

Stability Test: Backward Euler

How to test an integration formula for stability.

Test equation:

$$\dot{x} = \lambda x$$



Apply Backward Euler

$$x_{n+1} = x_n + h\dot{x}_{n+1}$$

$$x_{n+1} = x_n + h\lambda x_{n+1}$$

$$x_{n+1} = (1 - h\lambda)^{-1} x_n$$

Stability Test: Backward Euler

$$x_{n+1} = (1 - h\lambda)^{-1} x_n$$

$$t = 0 \longrightarrow x_o$$

$$t = h \longrightarrow x_1 = (1 - h\lambda)^{-1} x_0$$

$$t = 2h \longrightarrow x_2 = (1 - h\lambda)^{-1} x_1 = (1 - h\lambda)^{-2} x_o$$

$$\vdots$$

$$t = nh \longrightarrow x_n = (1 - h\lambda)^{-n} x_o$$

Stability Test: BE

$$x_n = (1 - h\lambda)^{-n} x_o$$

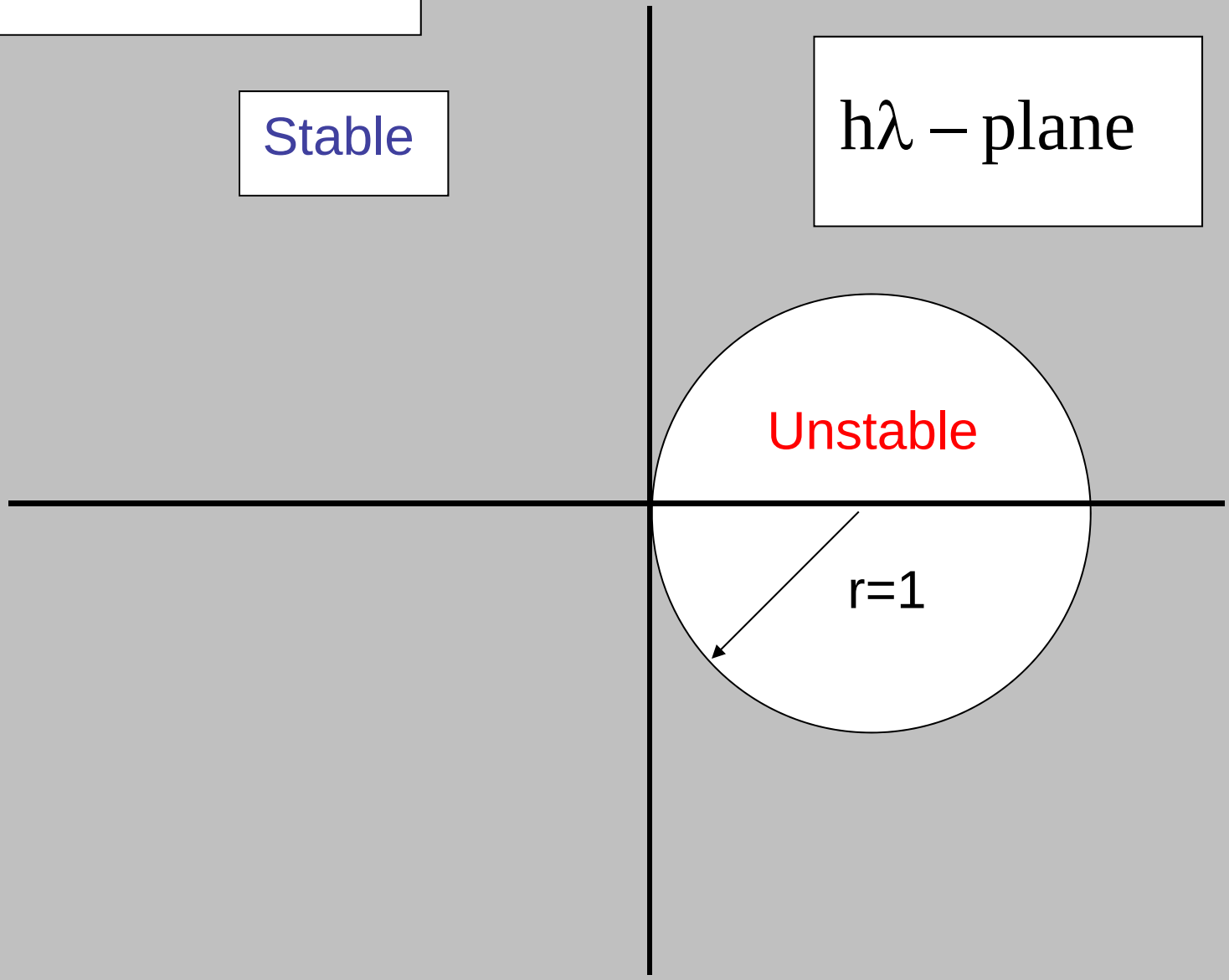
Conditions for stability:

$n \to \infty \Rightarrow x_n \to 0$

➔ $\lambda < 0$ i.e. system has stable poles

➔ $|1 - h\lambda| > 1$

Numerical Stability-BE

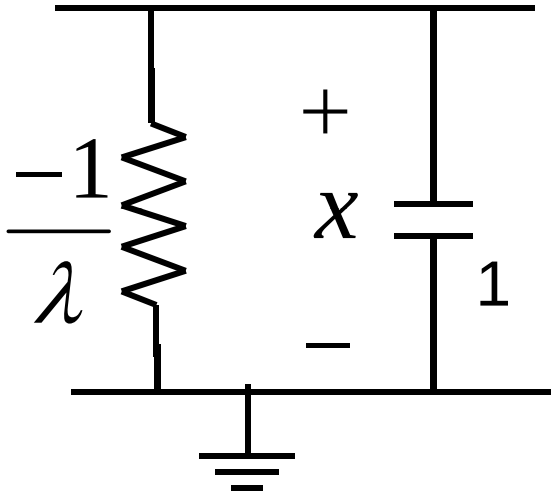


Stability Test: TR Rule

How to test an integration formula for stability.

Test equation:

$$\dot{x} = \lambda x$$



Apply TR

$$x_{n+1} = x_n + \frac{h}{2}\left(\dot{x}_n + \dot{x}_{n+1}\right)$$

$$\dot{x}_n = \lambda x_n$$

$$\dot{x}_{n+1} = \lambda x_{n+1}$$

$$x_{n+1} = x_n + \frac{h}{2}\left(\lambda x_n + \lambda x_{n+1}\right)$$

Stability Test: TR Rule

$$x_{n+1} = x_n + \frac{h}{2}\left(\lambda x_n + \lambda x_{n+1}\right)$$

$$\Rightarrow \quad \left(1 - \frac{h\lambda}{2}\right) x_{n+1} = \left(1 + \frac{h\lambda}{2}\right) x_n$$

$$\Rightarrow \quad x_{n+1} = \left(1 - \frac{h\lambda}{2}\right)^{-1} \left(1 + \frac{h\lambda}{2}\right) x_n$$

Stability Test: TR Rule

$$x_{n+1} = \left(1 - \frac{h\lambda}{2}\right)^{-1} \left(1 + \frac{h\lambda}{2}\right) x_n$$

$$n \to \infty \quad \Rightarrow \quad x_n \to 0$$

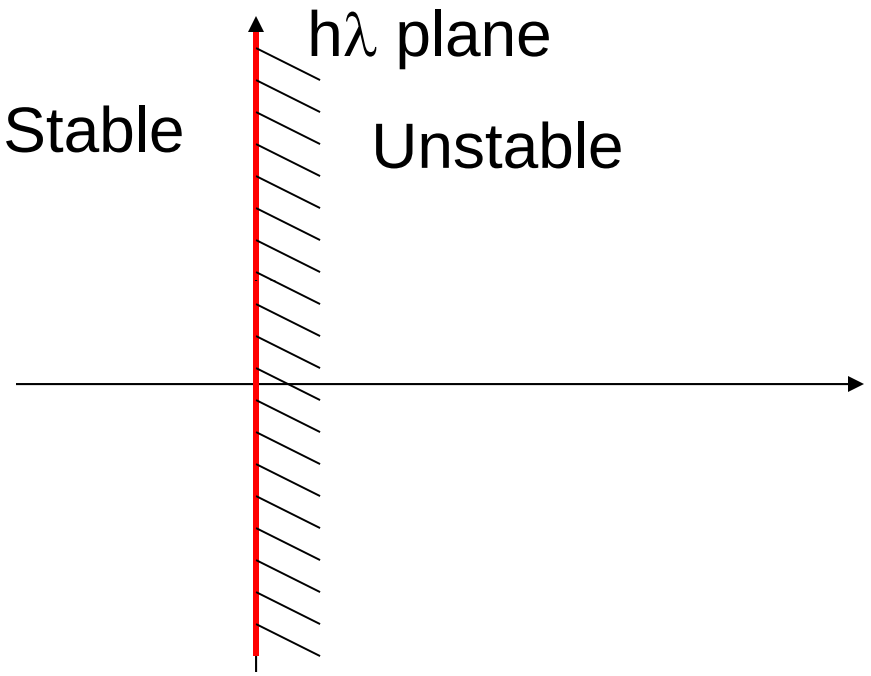
$$\left| \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} \right| < 1 \quad \Rightarrow \quad \begin{cases} \left| \frac{1+z}{1-z} \right| < 1 \\ z = \frac{h\lambda}{2} \end{cases}$$

Stability Region of TR-Rule

Stable when:

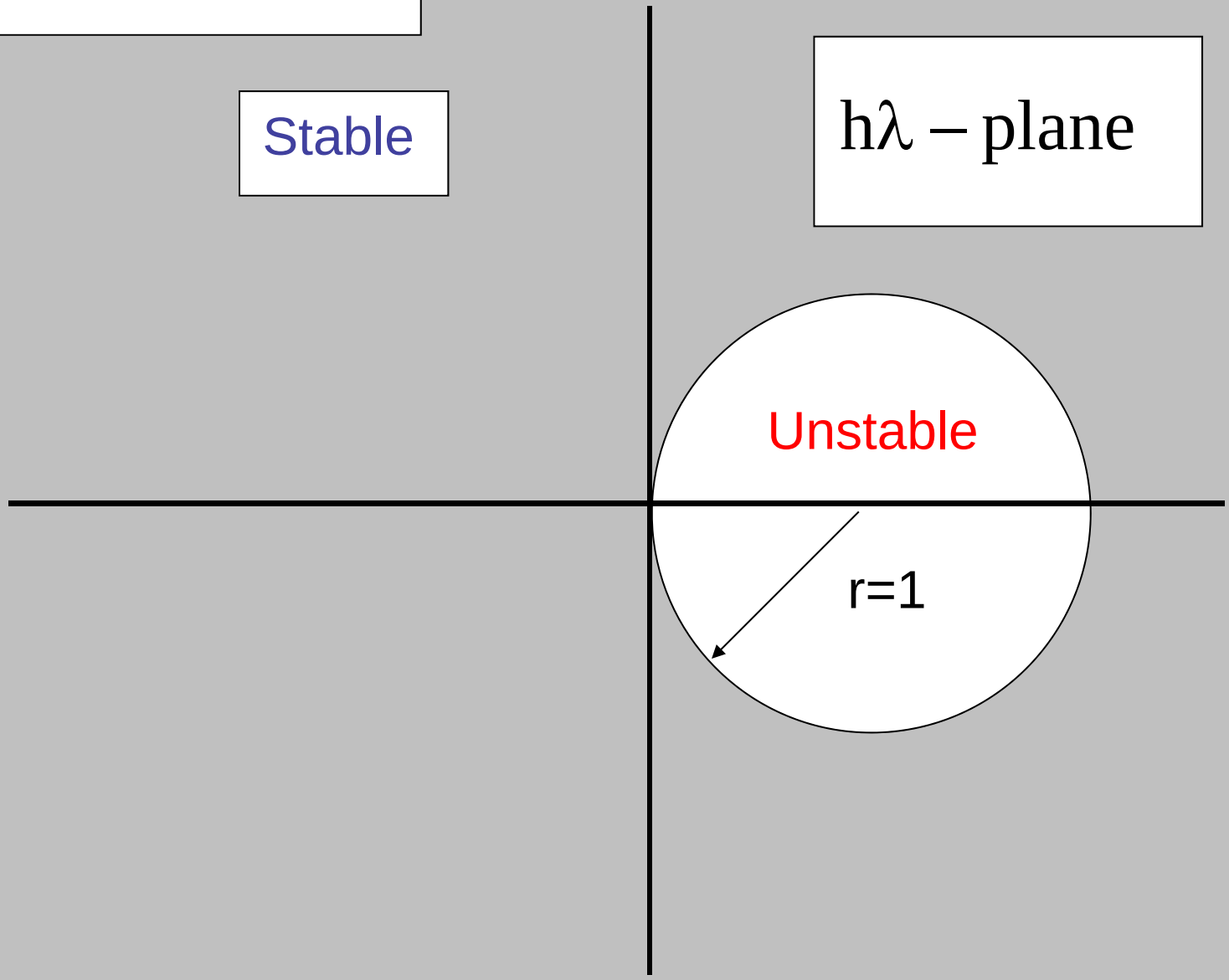
$$\mathrm{Re}[h\lambda] < 0$$

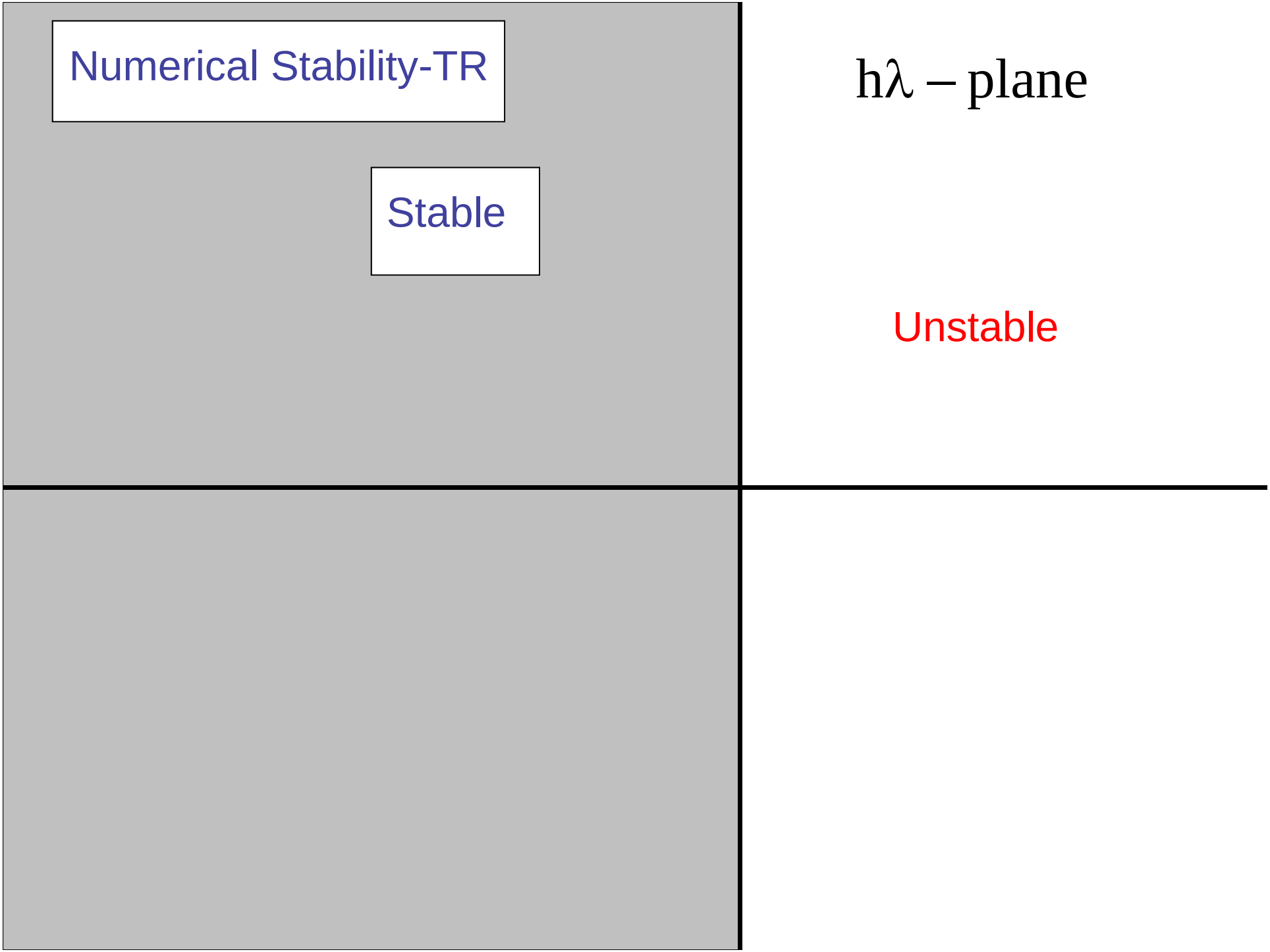
For stable systems Re(λ)<0 the TR rule is stable for all values of h.



TR-Rule
is A-stable

Numerical Stability-BE




Numerical Stability-TR
Stable
hλ – plane
Unstable

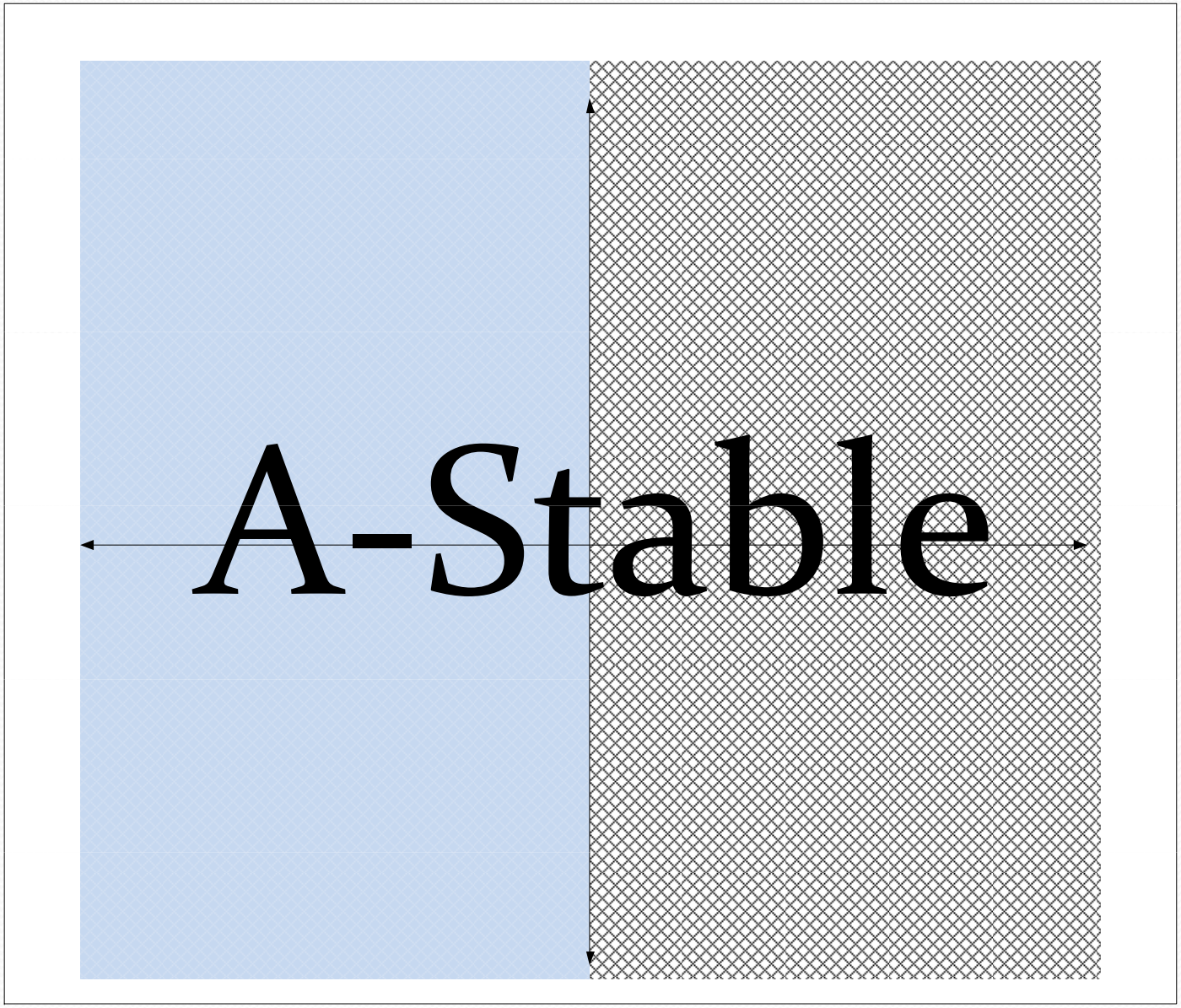
Dalquist's Barriers

Germund Dahlquist
1925-2005

- All explicit LMS are not absolutely stable
- The highest order of absolutely stable implicit LMS method is 2

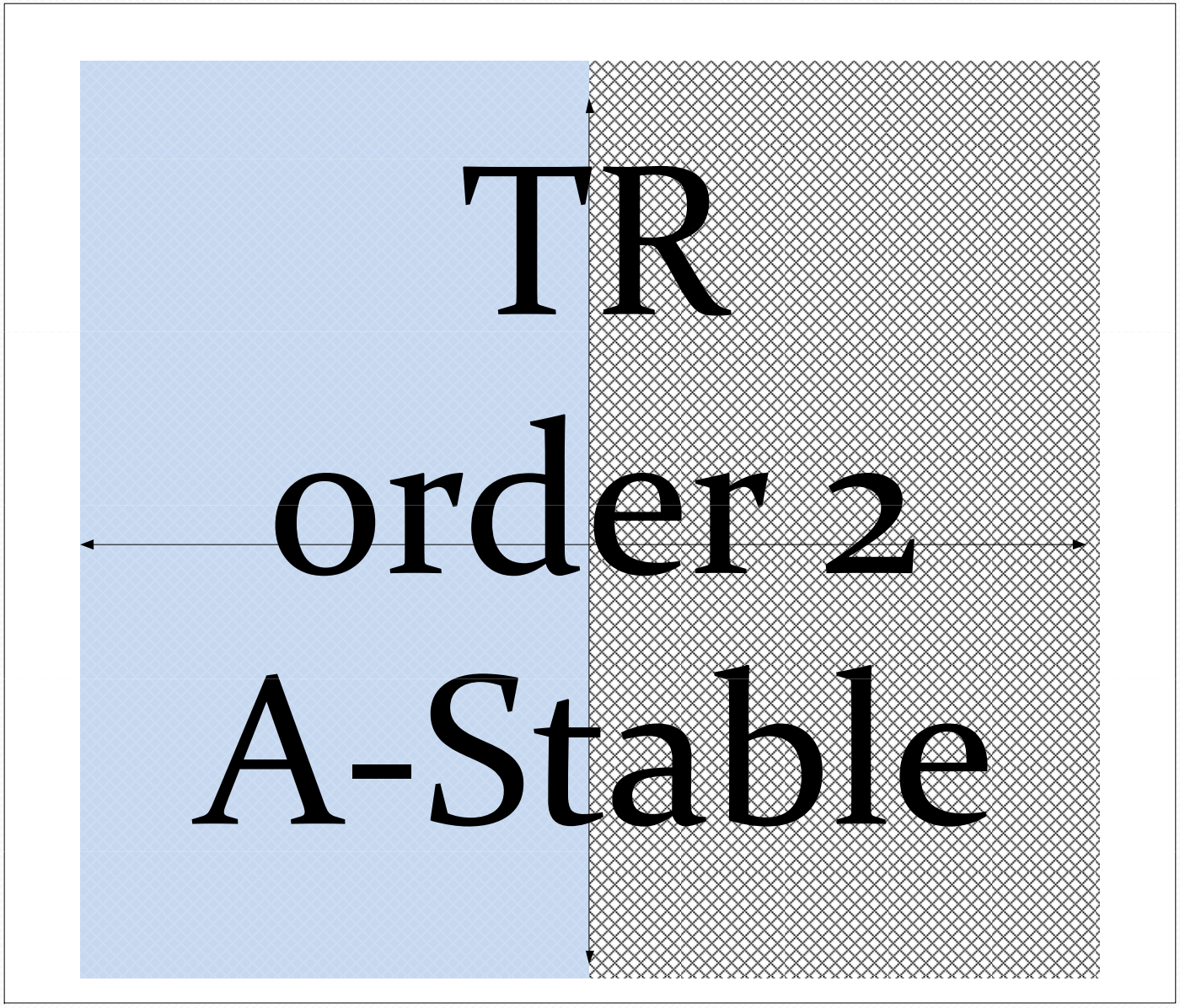
Stability in DE solvers

Laplace s-domain



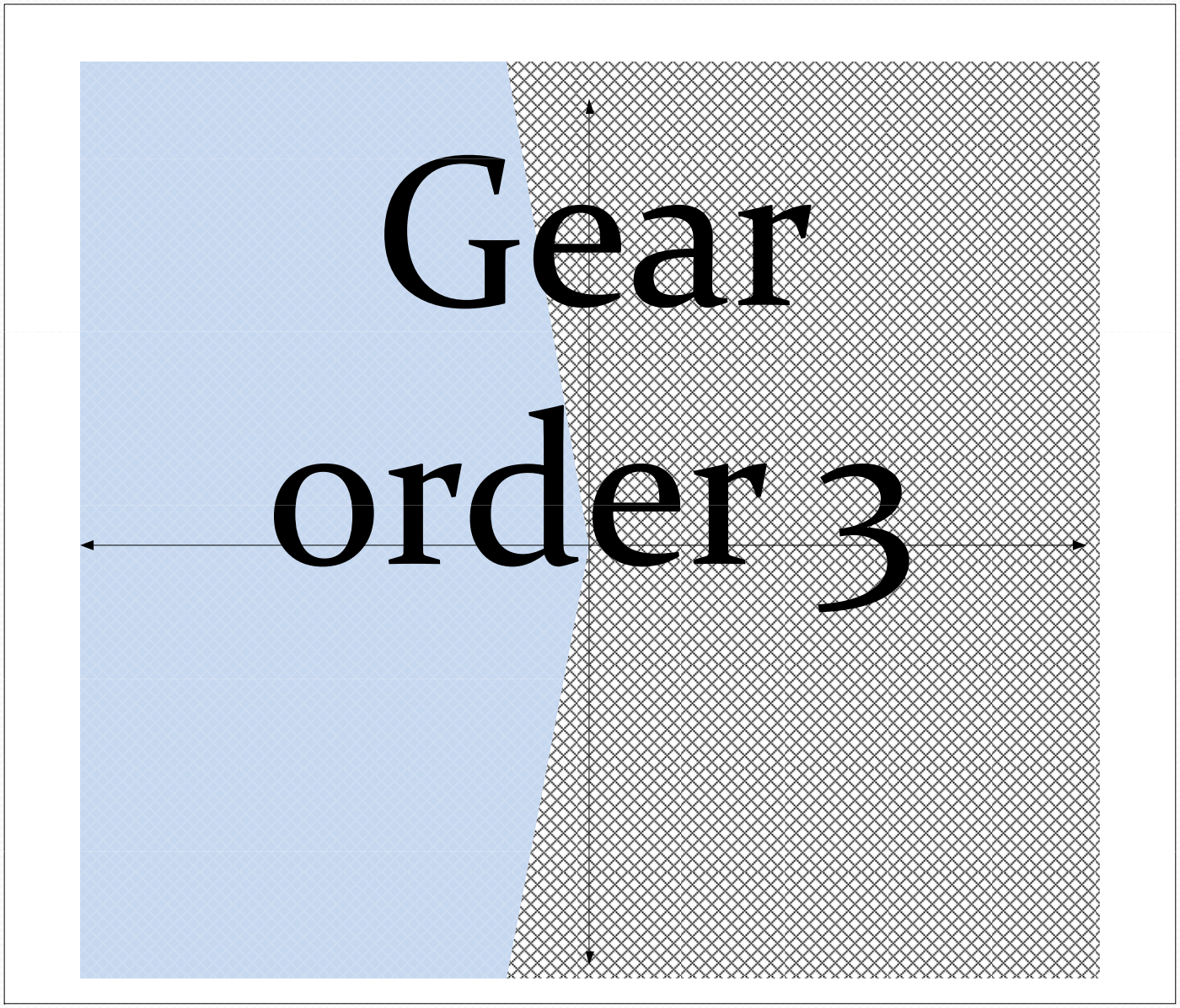
Stability in DE solvers

Laplace s-domain



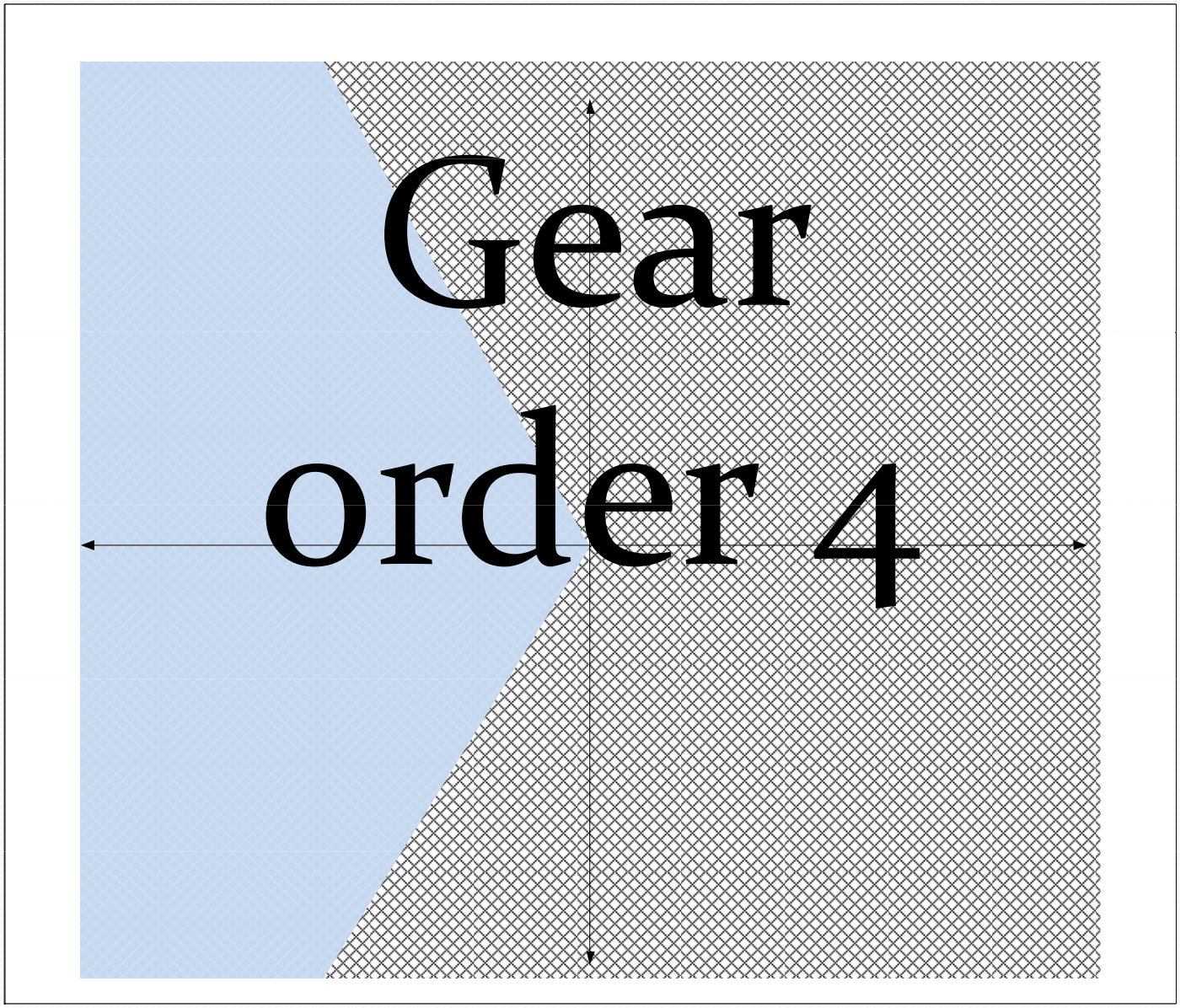
Stability in DE solvers

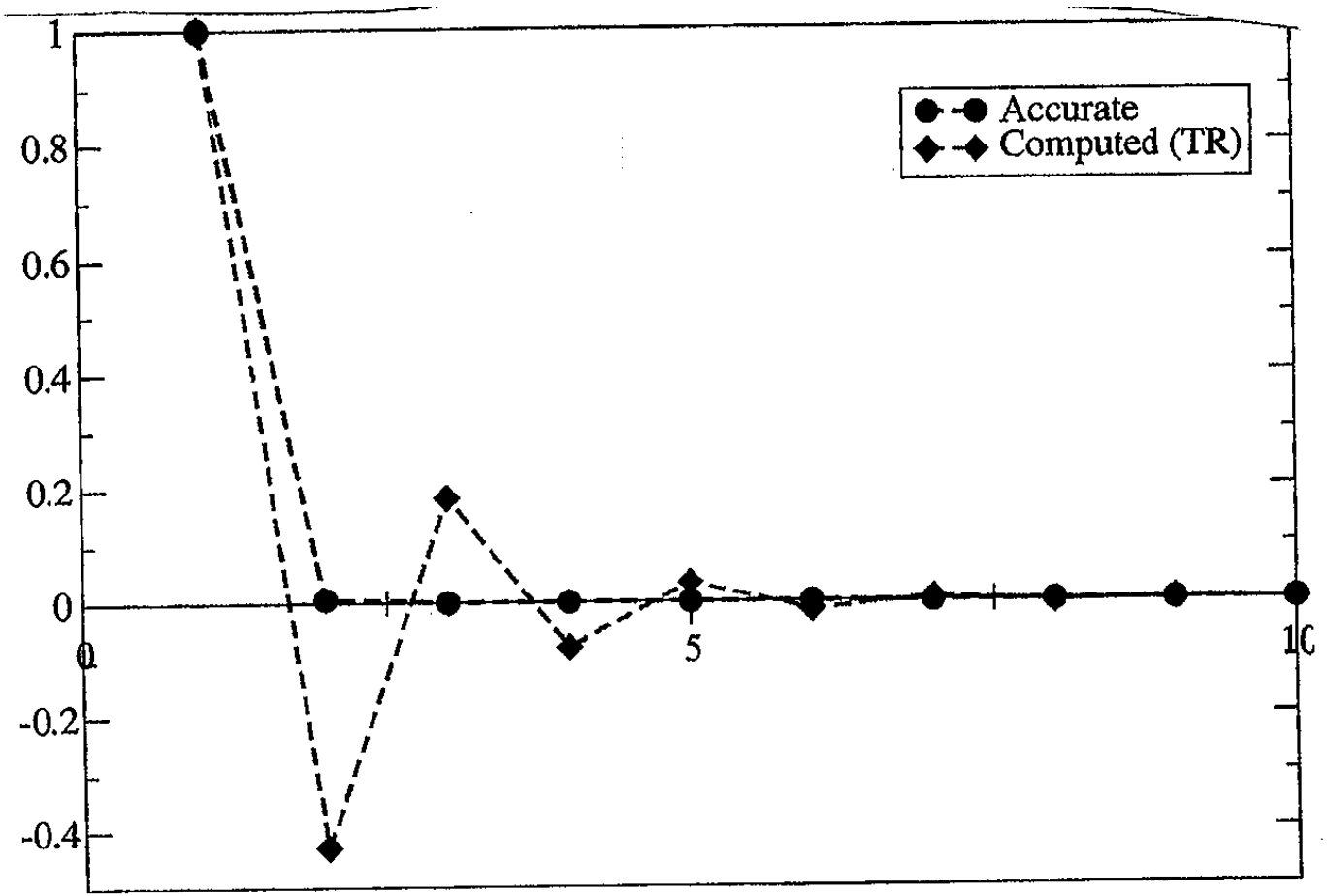
Laplace s-domain



Stability in DE solvers

Laplace s-domain




1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
5
10
Accurate
Computed (TR)

Revisiting the TR Rule

$$x_{n+1} = \left(1 - \frac{h\lambda}{2}\right)^{-1} \left(1 + \frac{h\lambda}{2}\right) x_n$$

$$n \to \infty \quad \Rightarrow \quad x_n \to 0$$

$$\Rightarrow \quad \mathrm{Re}[h\lambda] < 0$$

What will happen when $|\lambda| \gg 1$??

Revisiting the TR Rule: L Stability

Integration method is L-Stable iff:

$$\left| \frac{x_{n+1}}{x_n} \right| \to 0 \qquad |\lambda| \to \infty$$

Revisiting the TR Rule: L Stability

TR:

$$x_{n+1} = \left(1 - \frac{h\lambda}{2}\right)^{-1} \left(1 + \frac{h\lambda}{2}\right) x_n$$

BE :

$$x_{n+1} = \left(1 - h\lambda\right)^{-1} x_n$$

- TR is NOT L –Stable
- BE is L-stable
- L- Stable method is A-Stable

High-Order Integration

$$\alpha_0 x_{n+1} + \alpha_1 h \dot{x}_{n+1} + \alpha_2 h^2 \ddot{x}_{n+1} = \beta_0 x_n + \beta_1 h \dot{x}_n + \beta_2 h^2 \ddot{x}_n$$

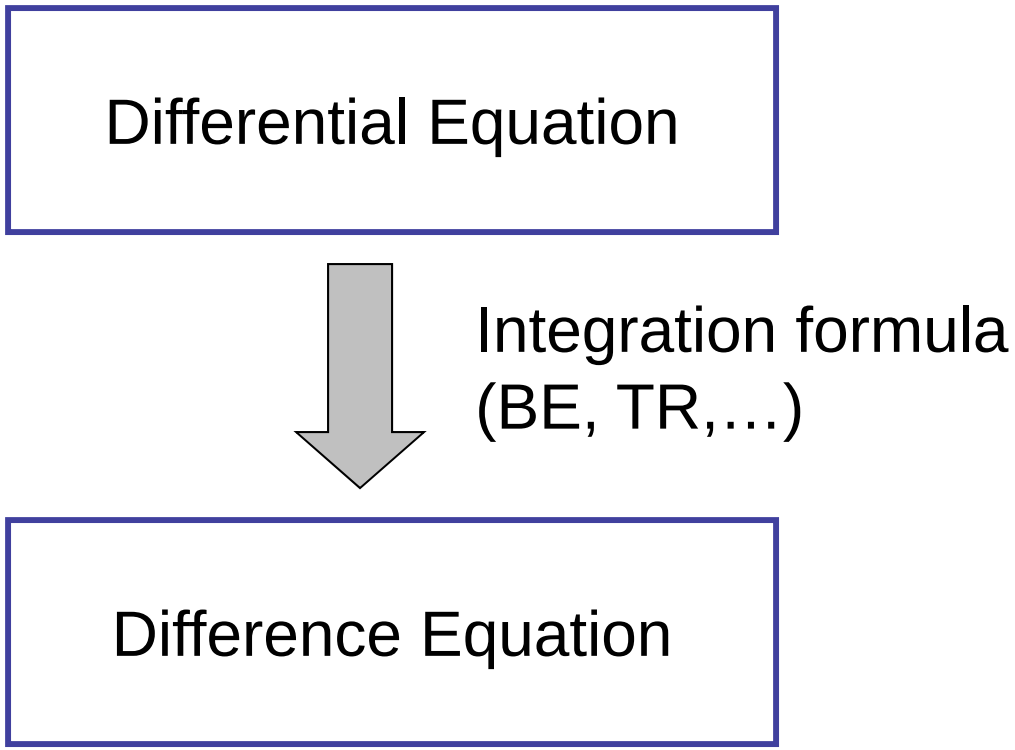
Single step
LMS : No
Order: 4

A-stable ?

α_0	1
α_1	-0.5
α_2	0.0833
β_0	1
β_1	0.5
β_2	0.0833

Numerical Integration

The solution of the differential equation is transformed into a solution of a system of linear equation at each time point.



Difference Equations-LMS

Backward Euler

$$\left(\mathbf{G}+\frac{\mathbf{C}}{h}\right)\mathbf{x}_{n+1}=\frac{\mathbf{C}}{h}\mathbf{x}_{n}+\mathbf{b}_{n+1}$$

Trapezoidal Rule

$$\left(\mathbf{G}+\frac{2\mathbf{C}}{h}\right)\mathbf{x}_{n+1}=\left(\frac{2\mathbf{C}}{h}-\mathbf{G}\right)\mathbf{x}_{n}+\mathbf{b}_{n}+\mathbf{b}_{n+1}$$

Difference Equations-High Order

Eliminate $\mathbf{x}''_{n+1}$

$$\begin{pmatrix} & \end{pmatrix}\begin{pmatrix} \mathbf{x}_{n+1} \\ \mathbf{x}'_{n+1} \end{pmatrix} = \begin{pmatrix} & \end{pmatrix}\begin{pmatrix} \mathbf{x}_{n} \\ \mathbf{x}'_{n} \end{pmatrix} + \begin{pmatrix} & \end{pmatrix}$$

Eliminate $\mathbf{x}'''_{n+1}$

$$\begin{pmatrix} & \end{pmatrix}\begin{pmatrix} \mathbf{x}_{n+1} \\ \mathbf{x}'_{n+1} \\ \mathbf{x}''_{n+1} \end{pmatrix} = \begin{pmatrix} & \end{pmatrix}\begin{pmatrix} \mathbf{x}_{n} \\ \mathbf{x}'_{n} \\ \mathbf{x}''_{n} \end{pmatrix} + \begin{pmatrix} & \end{pmatrix}$$

The HiSPICE Algorithm

A linear circuit with N unknown waveforms is represented by

$$C\frac{\mathrm{d}x(t)}{\mathrm{d}t} + Gx(t) = u(t)$$

- C and G are N x N matrices
- $u(t)$ are the independent sources

Assumed Known

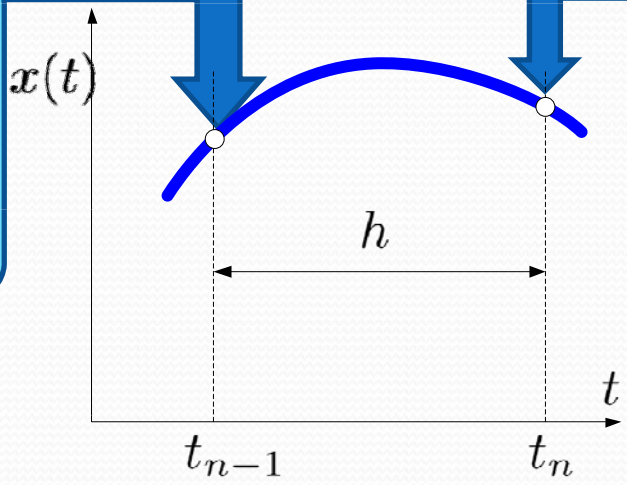
$$x(t_{n-1}) \equiv x_{n-1}$$

$$h\left.\frac{\mathrm{d}x(t)}{\mathrm{d}t}\right|_{t_{n-1}} \equiv x_{n-1}^{(1)}$$

$$\vdots$$

$$h^i\left.\frac{\mathrm{d}^i x(t)}{\mathrm{d}t^i}\right|_{t_{n-1}} \equiv x_{n-1}^{(i)}$$

Time Stepping



?

Need to be computed

$$x(t_n) \equiv x_n$$

Added Unknowns

$$h\left.\frac{\mathrm{d}x(t)}{\mathrm{d}t}\right|_{t_n} \equiv x_n^{(1)}$$

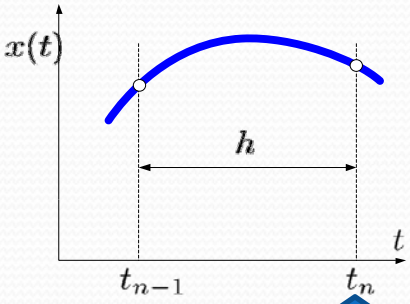
$$\vdots$$

$$h^j\left.\frac{\mathrm{d}^j x(t)}{\mathrm{d}t^j}\right|_{t_n} \equiv x_n^{(j)}$$

1)E. Gad, M. Nakhla, R. Achar, Y. Zhou: A-Stable and L-Stable High-Order Integration Methods for Solving Stiff Differential Equations. *IEEE Trans. on CAD (2009)*.

2)Y. Zhou, E. Gad, M. Nakhla, R. Achar: Structural Characterization and Efficient Implementation Techniques for A-Stable High-Order Integration Methods. *IEEE Trans. on CAD (2012)*.

At each time step, we will need to solve an enlarged system of algebraic equations: *j* times the size of the original system (*Nj*)

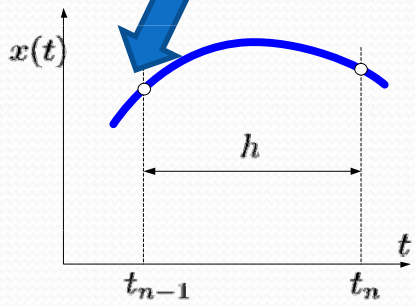


unknowns $\tilde{x}_n \in \mathbb{R}^{Nj}$

$$\left(\tilde{C} + \tilde{G}\right) \tilde{x}_n = \tilde{u}_n$$

$$\tilde{C}, \tilde{G} \in \mathbb{R}^{Nj \times Nj}$$

Obtained from $u(t)$, its derivatives, and the past time point



HiSPICE-Obreshkov formula

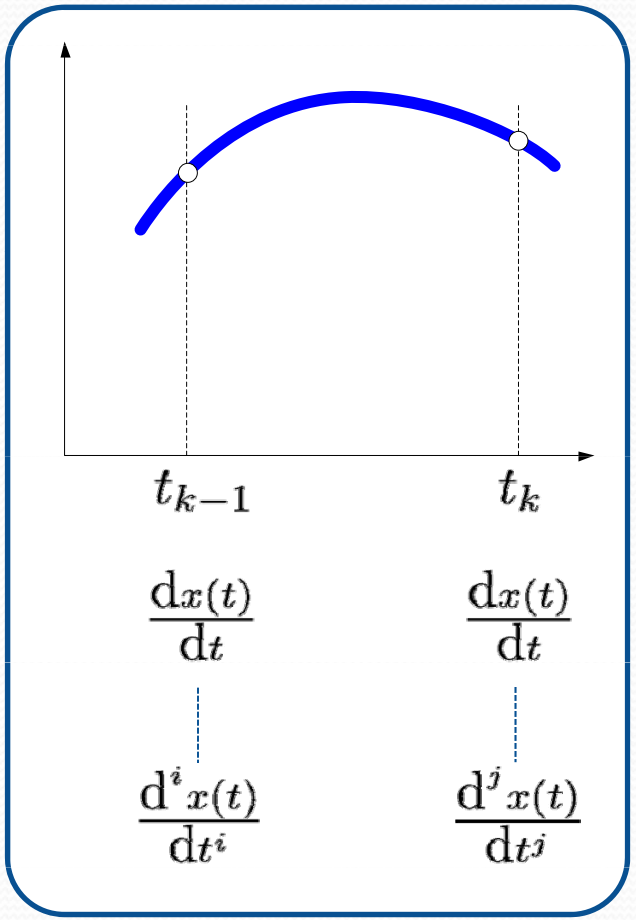
REF[1]: Eqs: (10) and (17)-(22)

REF[2]:Eqs: [4]-[6]

(HiSPICE-Obreshkov formula):

The main concept

- Order? $i + j$
- Stability?
  - Necessary and sufficient condition for A-stability is $j - 2 \leq i \leq j$
  - Necessary and sufficient condition for L-stability is $j - 2 \leq i < j$



$$\sum_{k=0}^{j} \alpha_k h^k x_{n+1}^{(k)} = \sum_{k=0}^{i} \beta_k h^k x_n^{(k)}$$

i=1; j=2: order 3, A,L

i=1;j=4: order 5, not A, not L

i=2;j=2: order 4, A, not L

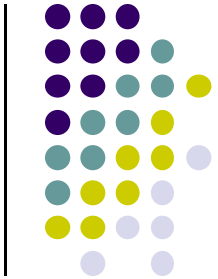
i=4; j=5: order 9, A, L

$Order \quad i + j$

$A - Stable: \quad j - 2 \le i \le j$

$L - Stable: \quad j - 2 \le i < j$

Example



i=2 , j=2

$$\sum_{k=0}^{j} \alpha_k h^k x_{n+1}^{(k)} = \sum_{k=0}^{i} \beta_k h^k x_n^{(k)}$$

$$\alpha_0 x_{n+1} + \alpha_1 h x_{n+1}^{(1)} + \alpha_2 h^2 x_{n+1}^{(2)} = \beta_0 x_n + \beta_1 h x_n^{(1)} + \beta_2 h^2 x_n^{(2)}$$

$$\frac{dx}{dt} = \lambda x$$

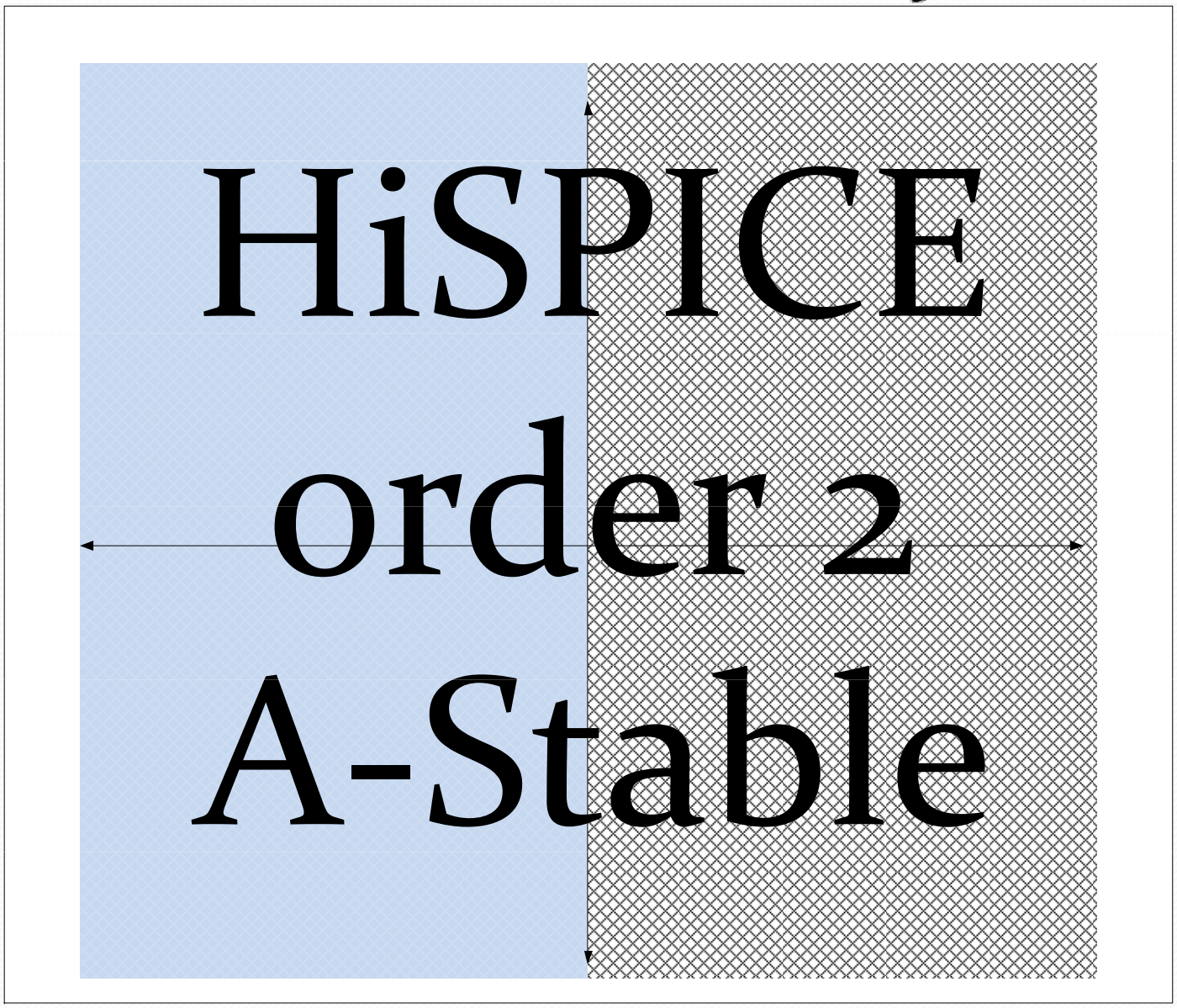
$$\left(\alpha_0 + \alpha_1 h\lambda + \alpha_2 h^2\lambda^2\right) x_{n+1} = \left(\beta_0 + \beta_1 h\lambda + \beta_2 h^2\lambda^2\right) x_n$$

$$A-stable \quad \Leftrightarrow \left|\frac{\beta_0 + \beta_1 h\lambda + \beta_2 h^2\lambda^2}{\alpha_0 + \alpha_1 h\lambda + \alpha_2 h^2\lambda^2}\right| < 1 \;\forall\; \Re\left(h\lambda\right) \le 0$$

$$L-Stable \quad \Leftrightarrow \lim_{h\lambda\to\infty} \frac{\beta_0 + \beta_1 h\lambda + \beta_2 h^2\lambda^2}{\alpha_0 + \alpha_1 h\lambda + \alpha_2 h^2\lambda^2} = 0$$

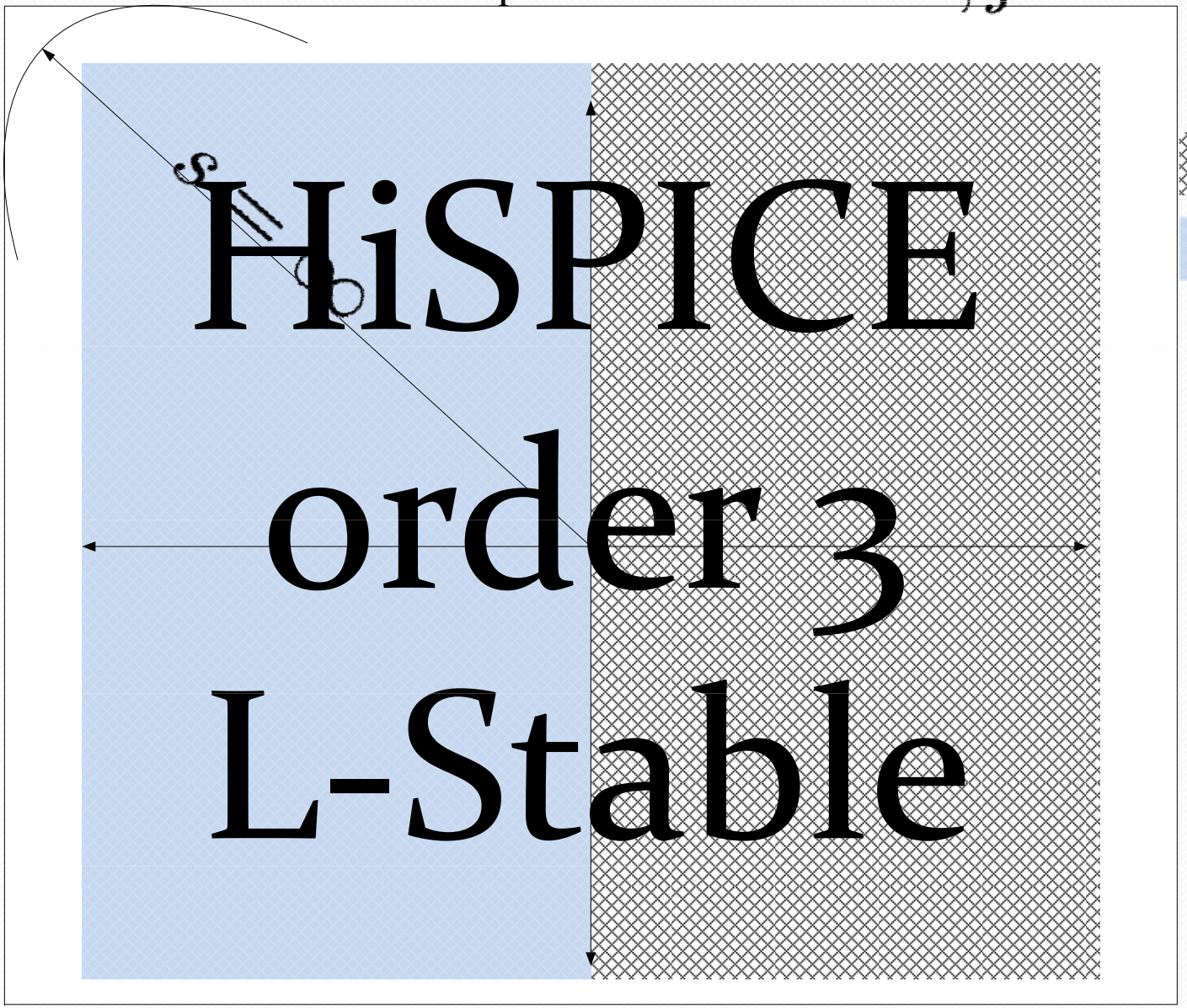
Stability in HiSPICE

Laplace s-domain $i = j = 1$



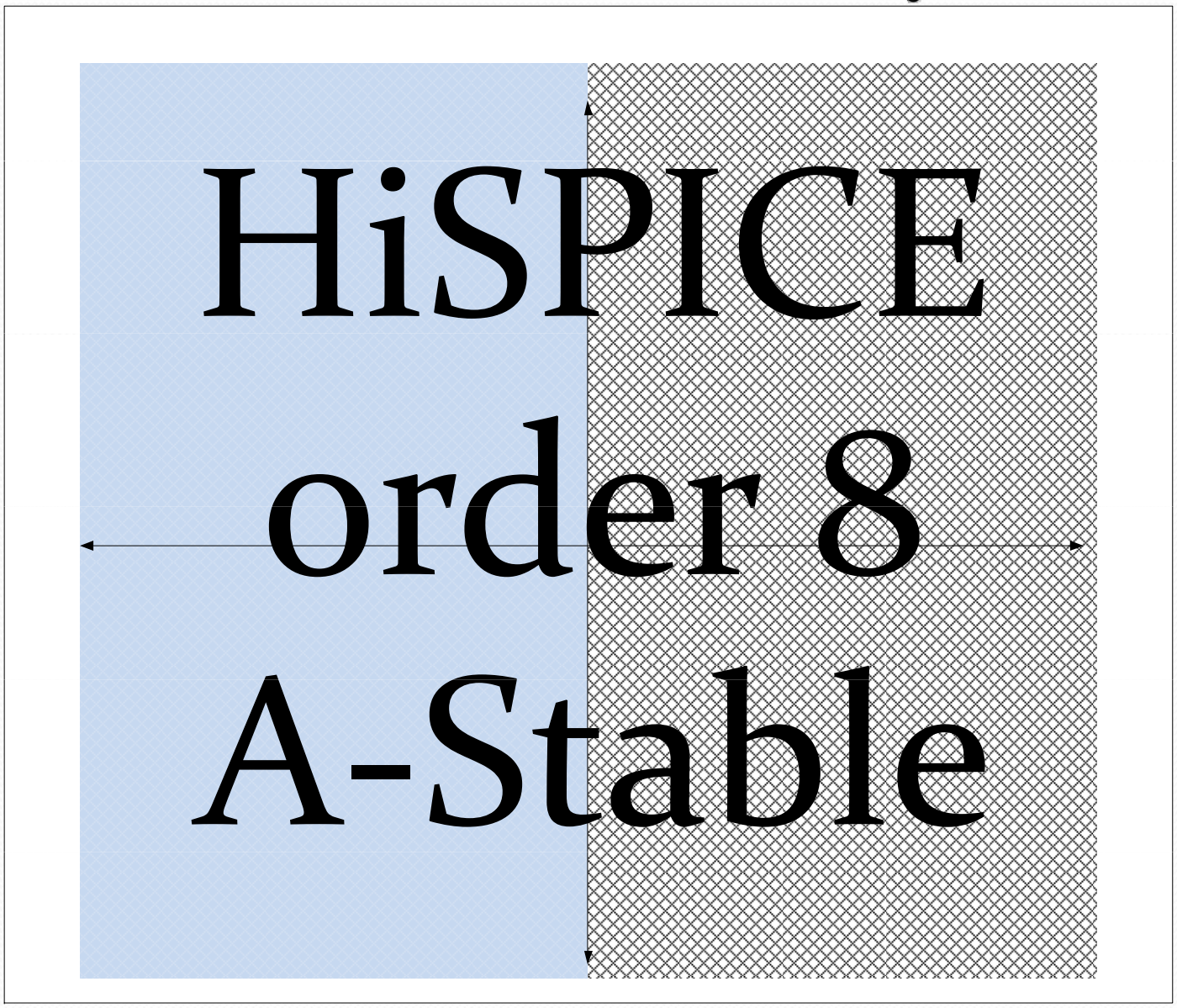
Stability in HiSPICE

Laplace s-domain $i = 1, j = 2$



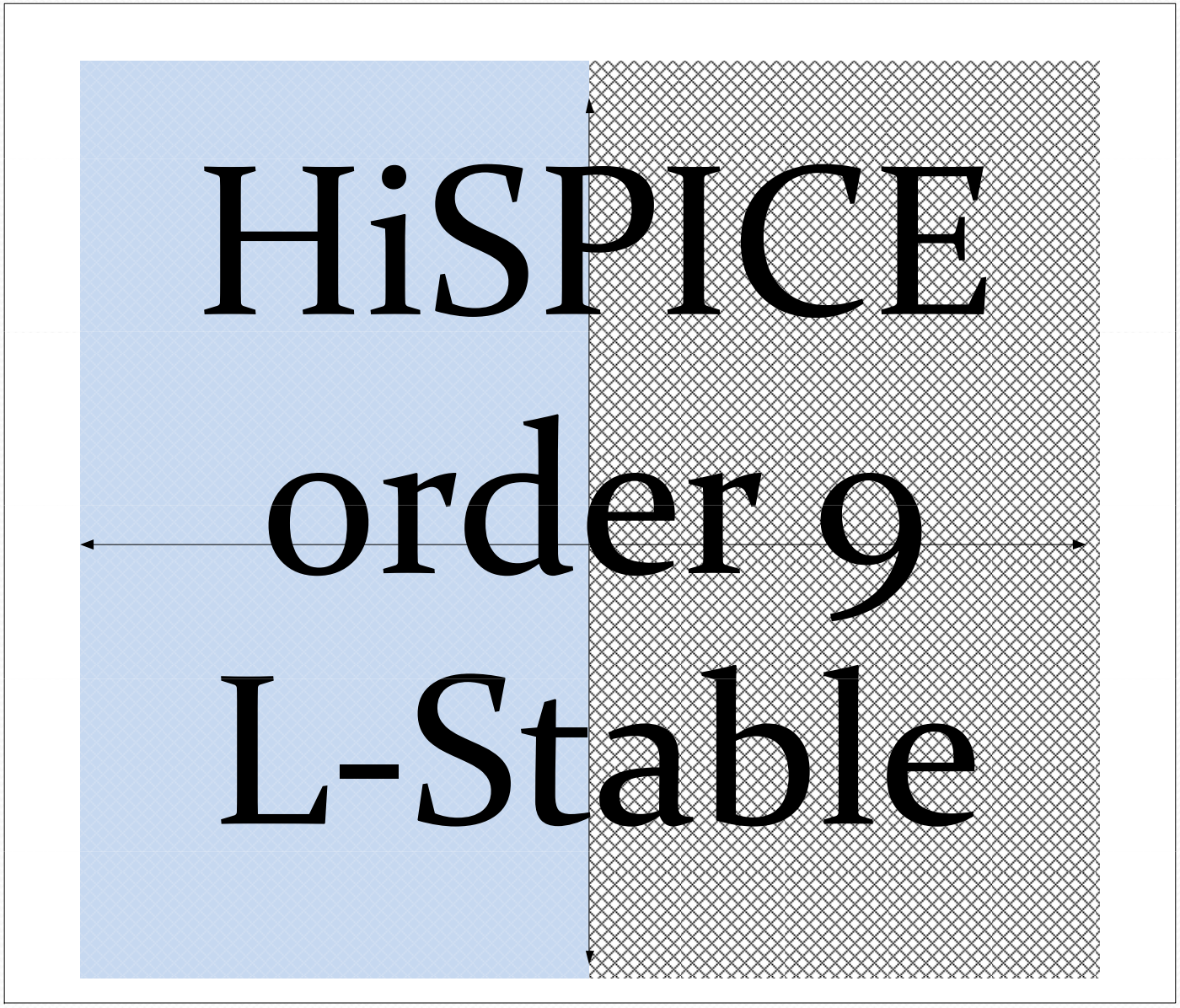
Stability in HiSPICE

Laplace s-domain $i = j = 4$



Stability in HiSPICE

Laplace s-domain i=4; j=5



The HiSPICE Algorithm- Alternative Ordering

$$x = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \Leftrightarrow \quad \tilde{x} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \frac{dv_1}{dt} \\ \frac{dv_2}{dt} \\ \frac{dv_3}{dt} \end{bmatrix}$$

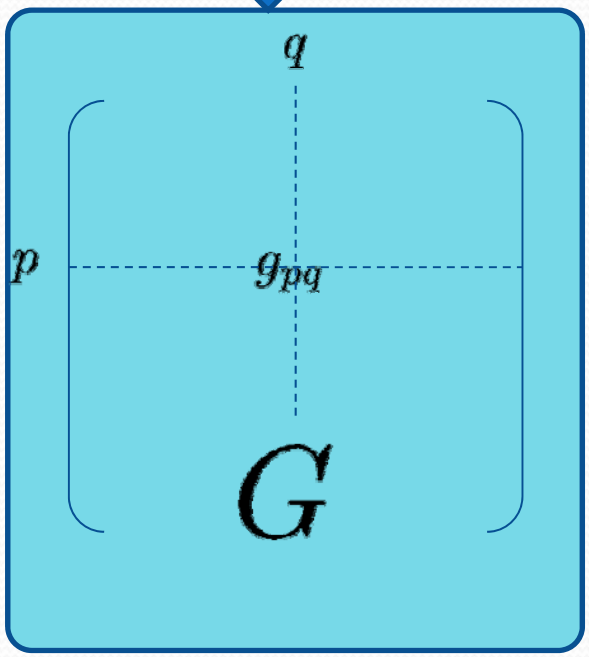
$$x = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \Leftrightarrow \quad \tilde{x} = \begin{bmatrix} v_1 \\ \frac{dv_1}{dt} \\ v_2 \\ \frac{dv_2}{dt} \\ v_3 \\ \frac{dv_3}{dt} \end{bmatrix}$$

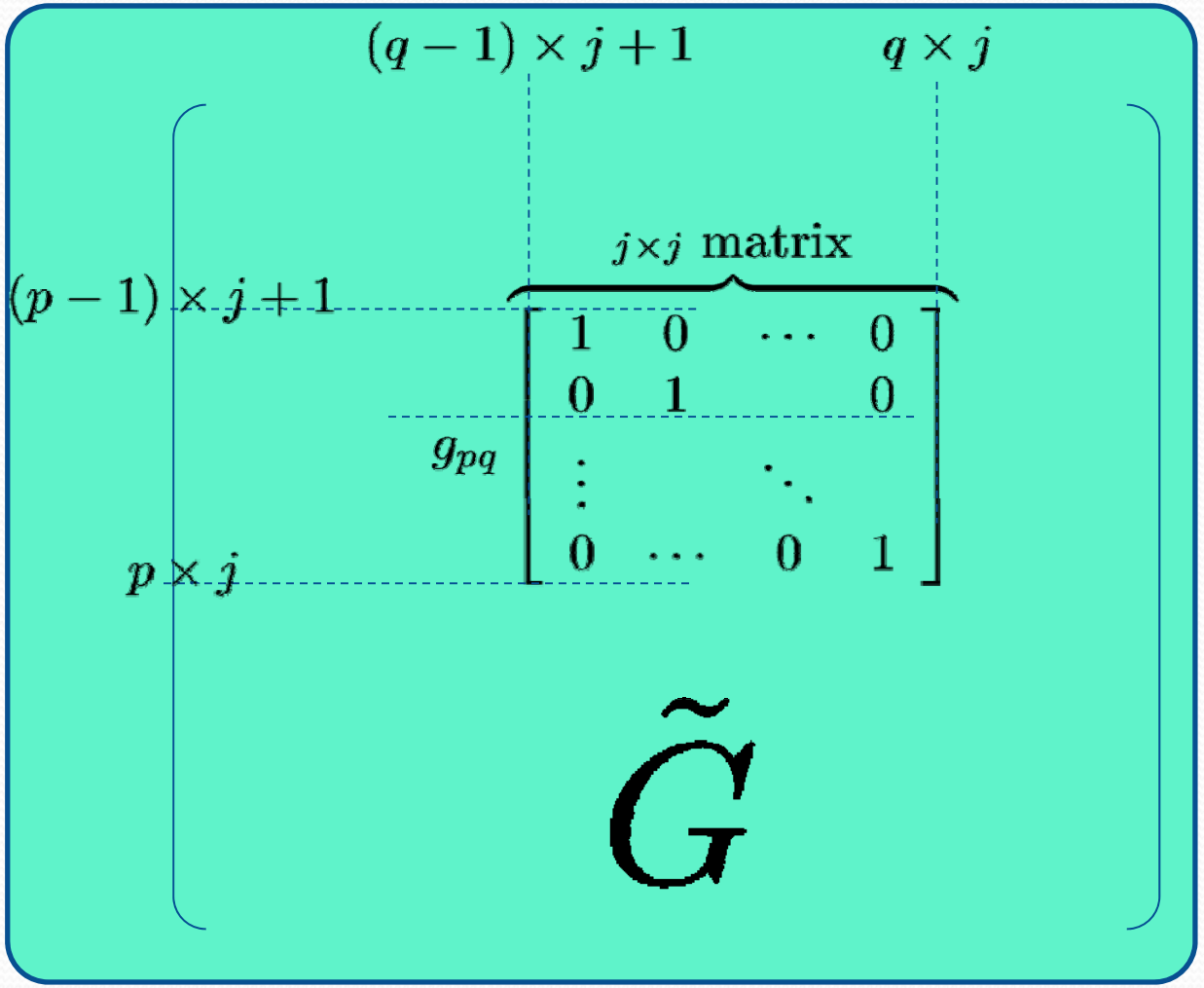
Ref[3]; Eqs(2)-(6)

The HiSPICE Algorithm- Alternative Ordering

The block structure of the augmented matrices $\tilde{C}, \tilde{G} \in \mathbb{R}^{Nj \times Nj}$

$$C\frac{\mathrm{d}x(t)}{\mathrm{d}t} + Gx(t) = u(t)$$

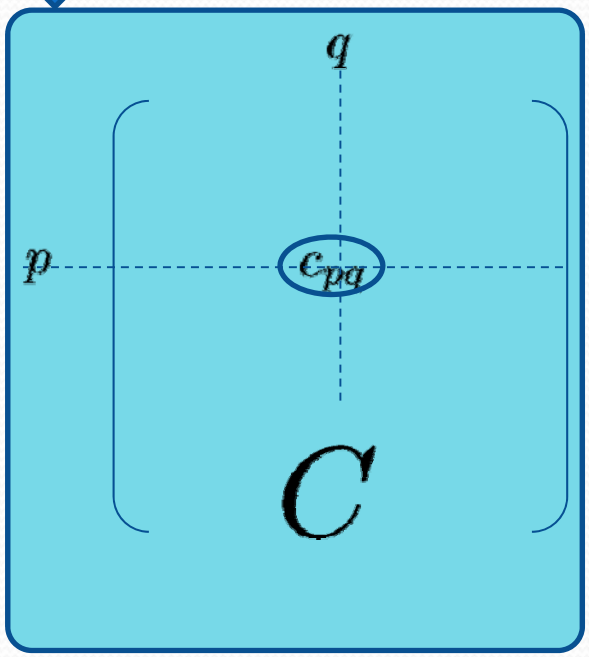


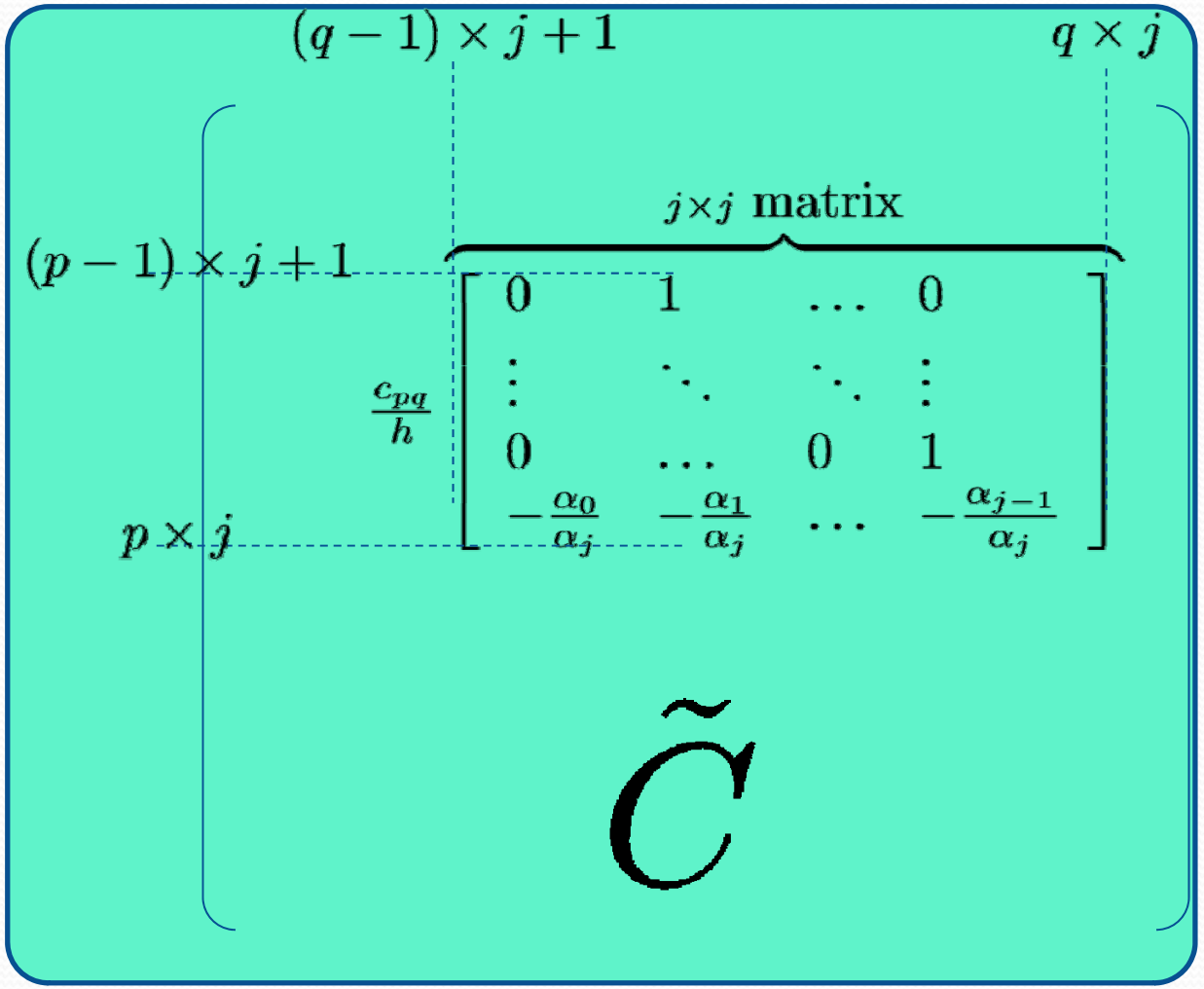


The HiSPICE Algorithm

The block structure of the augmented matrices $\tilde{C}, \tilde{G} \in \mathbb{R}^{Nj \times Nj}$

$$C\frac{\mathrm{d}x(t)}{\mathrm{d}t} + Gx(t) = u(t)$$





$$\alpha_k = (-1)^{(j)} \frac{(i+j-k)!j!}{i!(i+j)!(j-k)!}$$

Alternative Formulation

$$G=\begin{bmatrix} g & -g & 1 \\ -g & g & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Ref[3]; Eqs(2)-(6)

$$\tilde{G}=\begin{bmatrix} \begin{bmatrix} g & 0 \\ 0 & g \end{bmatrix} & \begin{bmatrix} -g & 0 \\ 0 & -g \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ \begin{bmatrix} -g & 0 \\ 0 & -g \end{bmatrix} & \begin{bmatrix} g & 0 \\ 0 & g \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix}$$

$$,\tilde{x}_1=\begin{bmatrix} \begin{bmatrix} v_1 \\ v_1^{(1)} \end{bmatrix} \\ \begin{bmatrix} v_2 \\ v_2^{(1)} \end{bmatrix} \\ \begin{bmatrix} i \\ i^{(1)} \end{bmatrix} \end{bmatrix}$$

Example

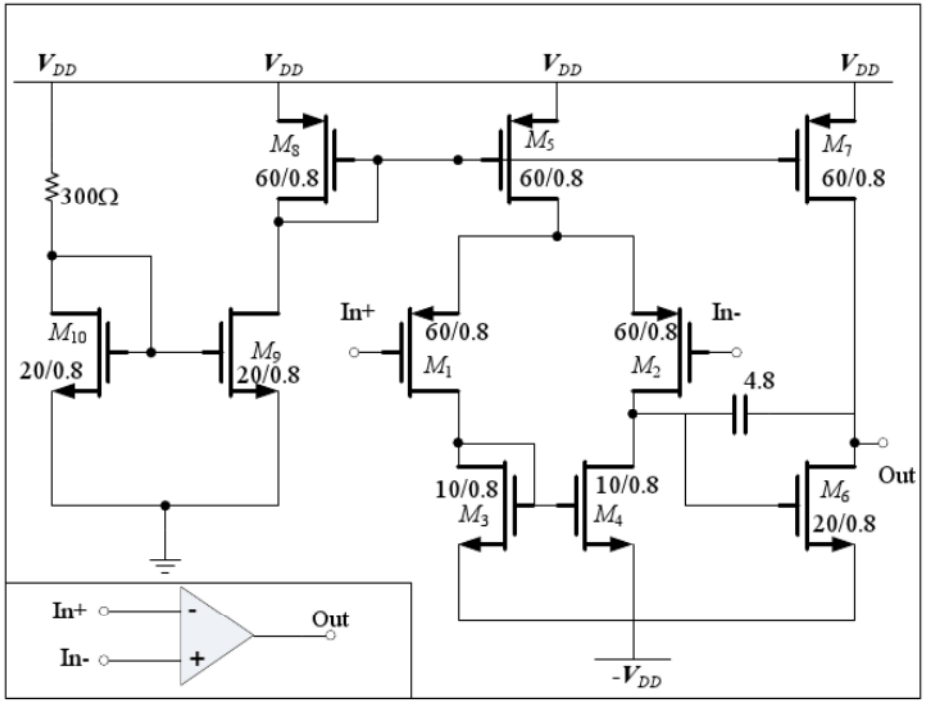
$$C = \begin{bmatrix} 0 & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\tilde{C} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & \frac{c}{h} \\ \frac{c}{h}\frac{\alpha_0}{\alpha_2} & \frac{c}{h}\frac{\alpha_1}{\alpha_2} \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix}$$

Example

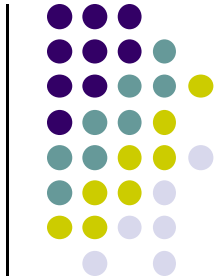
Cauer Low-pass filter consists of 8 OpAmps

Total Number of transistors = 180



Gad, E.; Nakhla, M.; Achar, R.; Yinghong Zhou; , " A-Stable and L-Stable High-Order Integration Methods for Solving Stiff Differential Equations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2009)*

Example



Cauer Low-pass filter consists of 8 OpAmps

Order	# time points	Speedup	Error
SPICE (2)	**1771**	-	< 0.001
HiSPICE (4)	**101**	**6.5**	< 0.001
HiSPICE (6)	**36**	**10.3**	< 0.001

Gad, E.; Nakhla, M.; Achar, R.; Yinghong Zhou; , " A-Stable and L-Stable High-Order Integration Methods for Solving Stiff Differential Equations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2009)*

Simulation Results

CPU time for TR method is 743.59 sec

Order	CPU (seconds)	Speedup
HiSPICE (6)	53.22	13.97
HiSPICE (8)	42.12	**17.65**

(single core)

REFERENCES

1. E. Gad, M. Nakhla, R. Achar and Y. Zhou, "A-Stable and L-Stable High-Order Integration Methods for Stiff Differential Equations", *IEEE Trans. on Computer Aided Design*, vol. 28, pp. 1359-1372, Sept. 2009.

2. Y. Zhou; E. Gad, M. Nakhla and R. Achar, R., “Structural Characterization and Efficient Implementation Techniques for -Stable High-Order Integration Methods , *IEEE Trans. CAD*, vol. 31 , Issue: 1, pp. 101 – 108, Jan. 2012

3. M. Farhan, E. Gad, M. Nakhla and R. Achar, “A New Method for Fast Transient Simulation of Large Linear Circuits using High-Order Stable Methods ” *IEEE Trans Advanced Packaging,* vol. 3, pp. 661-669, April 2013.

4. M. Farhan, E. Gad, M. Nakhla and R. Achar, “ Fast Simulation of Microwave Circuits with Nonlinear Terminations using High-Order Stable Methods,” *IEEE Trans. Microwave Theory and Techniques,* vol. 51, pp. 362-371, January 2013.