Modern Taylor Series Method: Motivation examples in MATLAB

Václav Šátek et.al.

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2, 612 66 Brno - Královo Pole
Václav Šátek, satek@fit.vutbr.cz

TU Vienna, 25.04.2017

Overview

- Modern Taylor Series Method (MTSM)
- Example – circuit test
- Example – wave equation

Extremely Accurate Solutions of Systems of Differential Equations

The development project deals with extremely exact, stable and fast numerical solutions of systems of differential equations.

The project is based on a mathematical method which uses the Taylor series method for solving differential equations.

By a numerical solution of an ordinary differential equation

$$y' = f(t, y), \quad y(t_0) = y_0$$

we understand the finding of a sequence:

$$y(t_1) = y_1,$$
$$y(t_2) = y_2,$$
$$\dots,$$
$$y(t_n) = y_n$$

The best-known and most accurate method of calculating a new value of a numerical solution of a differential equation is to construct the Taylor series in the form

$$y_{n+1} = y_n + h * f(t_n, y_n) + \frac{h^2}{2!} * f'(t_n, y_n) + \cdots + \frac{h^p}{p!} * f^{[p-1]}(t_n, y_n)$$

$$y' = f(t, y) \qquad y(0) = y_0$$

$$y_{n+1} = y_n$$

$$+ h \cdot y'_n \qquad ORD = 1$$

$$+ \frac{h^2}{2!} \cdot y''_n \qquad ORD = 2$$

$$+ \frac{h^3}{3!} \cdot y'''_n \qquad ORD = 3$$

$$\vdots$$

$$y' = f(t, y) \qquad y(0) = y_0$$

$$y_{n+1} = y_n$$

$$+ h \cdot y'_n \qquad ORD = 1$$

Let us define *ORD* as the order of Taylor series method, respectively the highest Taylor series term used in computation

$$y' = f(t, y) \qquad y(0) = y_0$$

$$y_{n+1} = y_n$$

$$+ h \cdot y'_n \qquad ORD = 1$$

$$+ \frac{h^2}{2!} \cdot y''_n \qquad ORD = 2$$

$$y' = f(t, y) \qquad y(0) = y_0$$

$$
\begin{aligned}
y_{n+1} = y_n & \\
+ h \cdot y'_n & \qquad ORD = 1 \\
+ \frac{h^2}{2!} \cdot y''_n & \qquad ORD = 2 \\
+ \frac{h^3}{3!} \cdot y'''_n & \qquad ORD = 3 \\
\vdots &
\end{aligned}
$$

The Modern Taylor Series is based on a **recurrent calculation of the Taylor series terms** for each time interval. Thus the complicated calculation of higher order derivatives (much criticised in the literature) need not be performed but rather the value of each Taylor series term is numerically calculated.

$$y_{n+1} = y_n + h \cdot y'_n + \frac{h^2}{2!} \cdot y''_n + \frac{h^3}{3!} \cdot y'''_n + \cdots$$

$$y_{n+1} = DY0_n + DY1_n + DY2_n + DY3_n + \cdots$$

Very effective computation of linear systems of differential equations (only **matrix-vector multiplications** are needed for each Taylor series term calculation).

$$\vec{y}' = A \cdot \vec{y}, \quad \vec{y}(0) = D\vec{Y}0_0$$

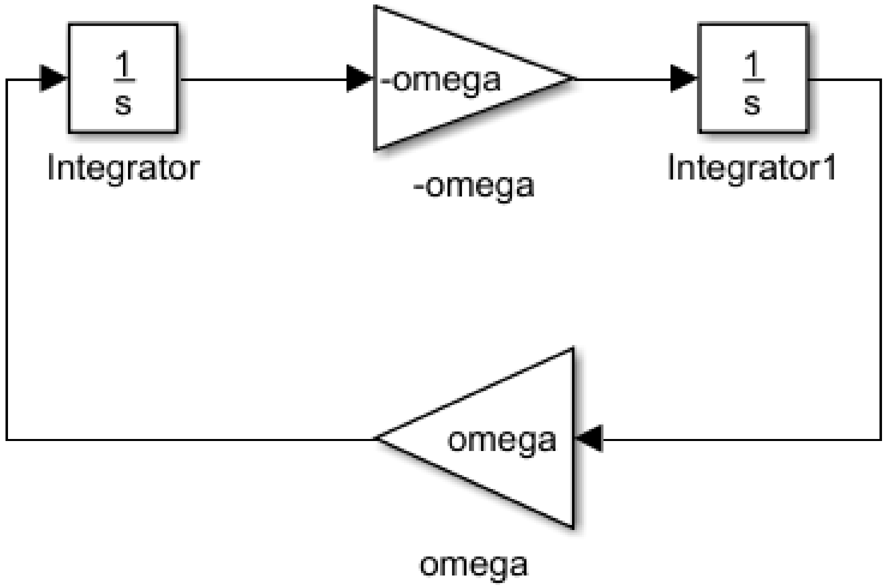
$$\vec{y}_{n+1} = D\vec{Y}0_n + D\vec{Y}1_n + D\vec{Y}2_n + D\vec{Y}3_n + \cdots$$

$$D\vec{Y}i_n = \frac{h}{i} \cdot A \cdot D\vec{Y}(i-1)_n, \quad i = 1 \cdots k, \quad ORD = k$$

An important part of the method is an automatic integration order setting, i.e. using as many Taylor series terms as the defined accuracy requires.

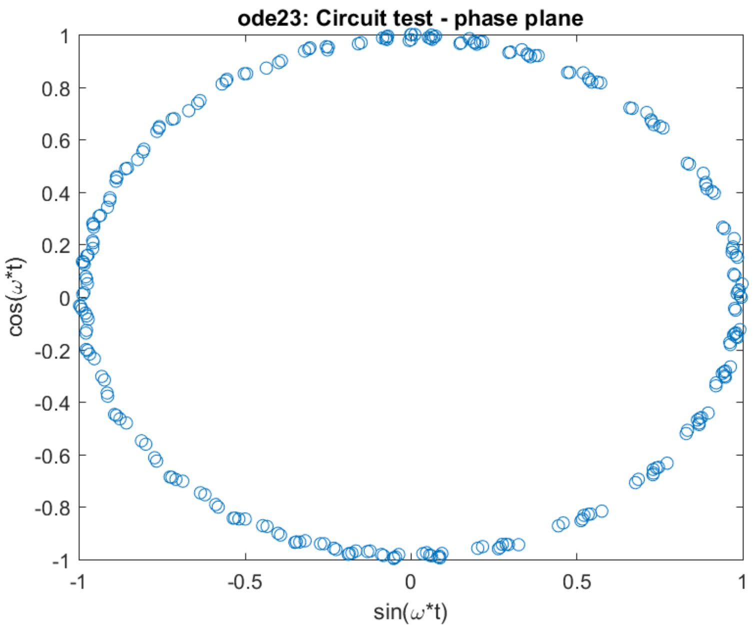
Example 1 – circuit test

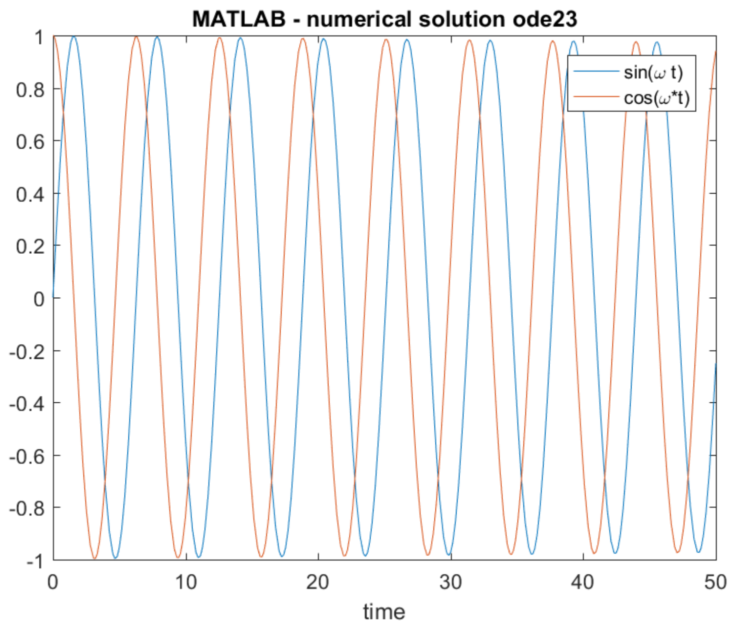
- Lets consider the following functions
  - $u = \sin(\omega t) \longrightarrow u' = \omega\, v, \quad u(0) = 0$
  - $v = \cos(\omega t) \longrightarrow v' = -\omega\, u, \quad v(0) = 1$
- These functions can be represented by the following block scheme:



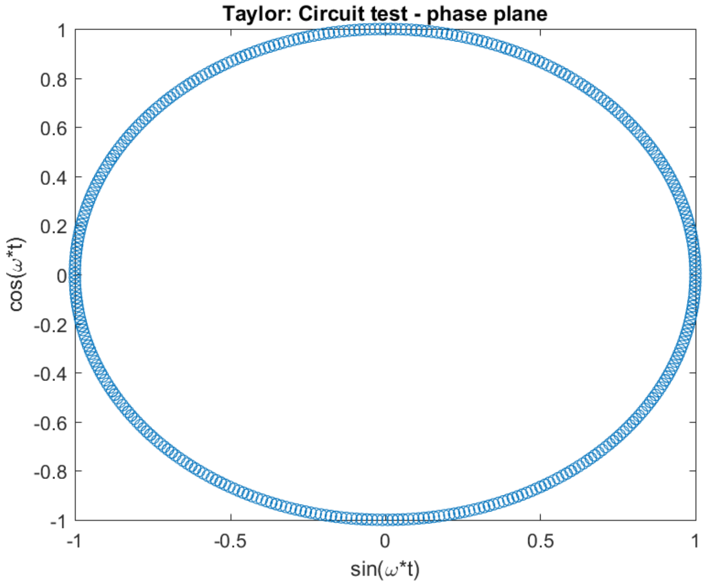
- The behavior of the system depends on ω

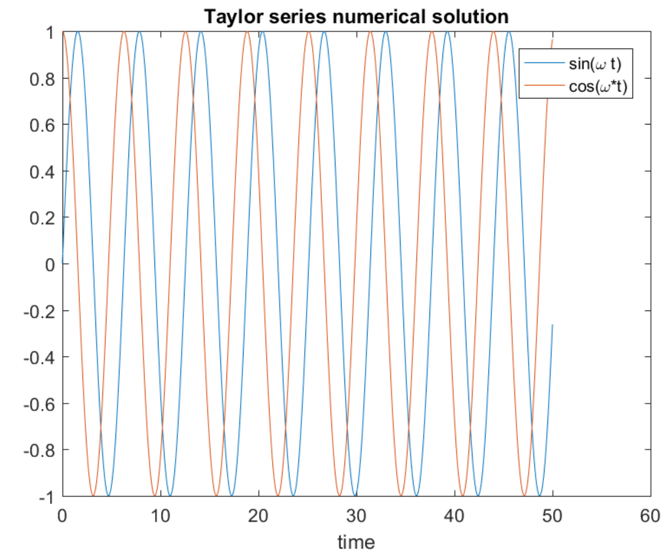
- MATLAB ode23 solver (default settings)

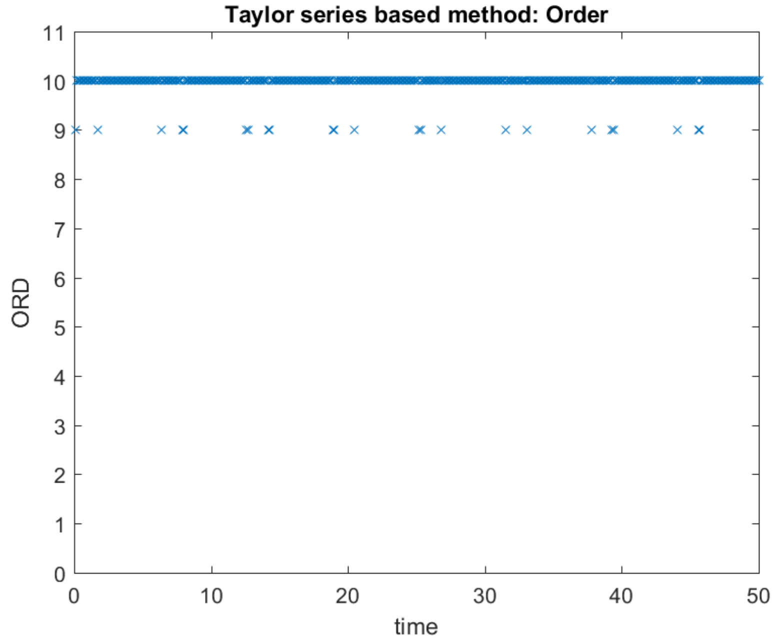




Example 1 – circuit test, $\omega = 1$







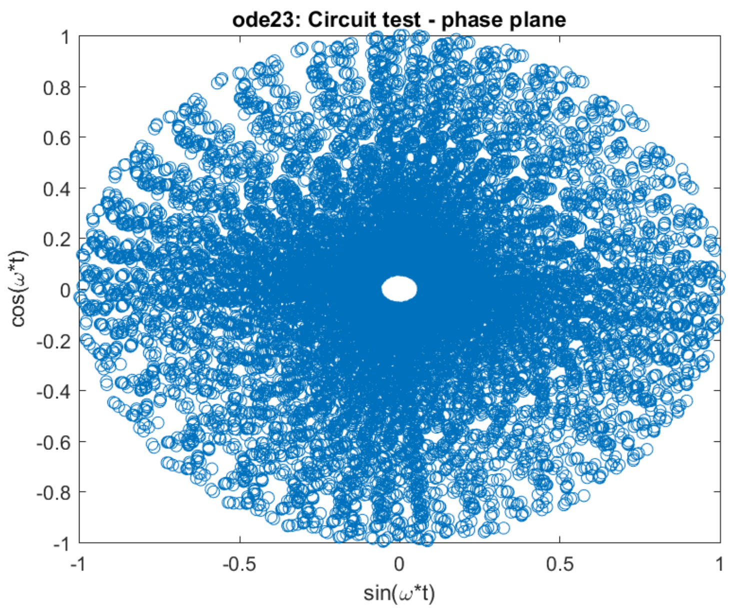
- MTSM (h=0.1)
- ORD≈10

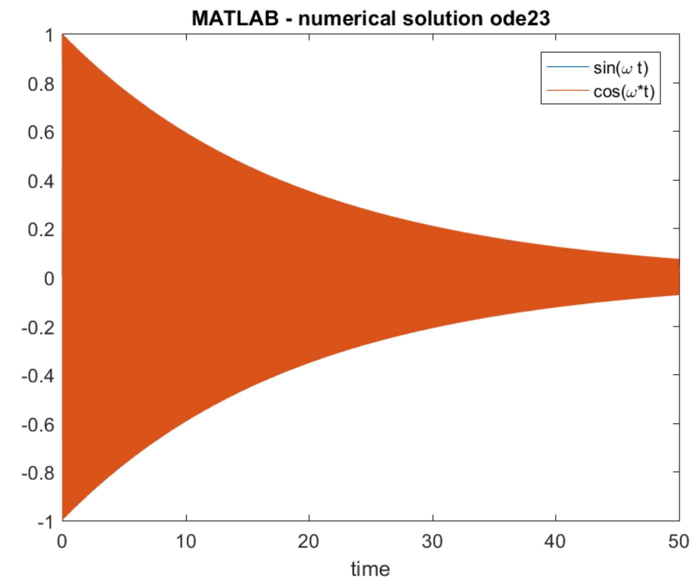
stable and fast solution

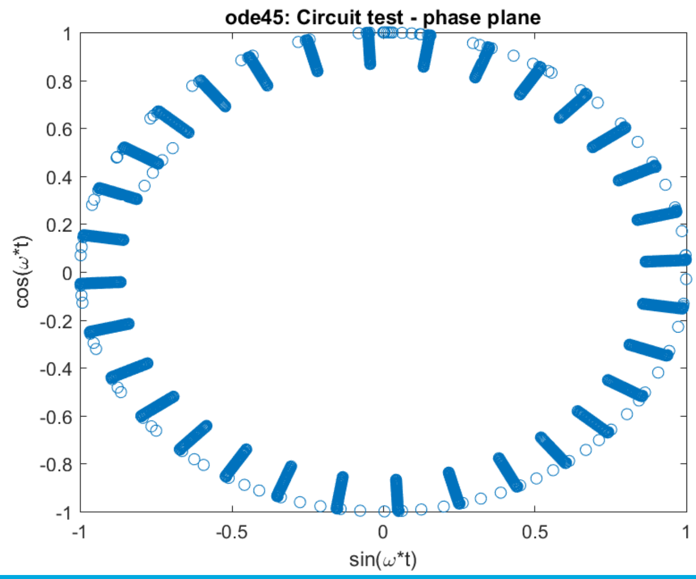
- $\omega = 1, tmax = 50, dt = 0.1$

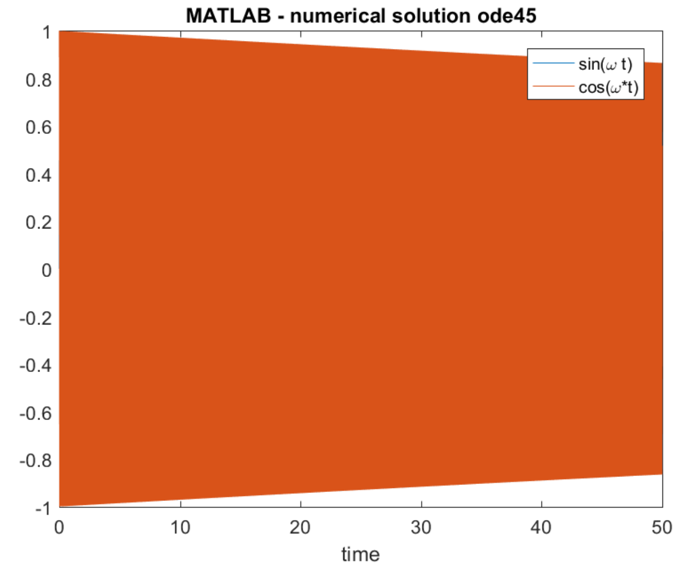
Method	Steps	||Error||
ode23	245	0.0365738
ode45	277	0.00731112
MTSM	500	6.99885e-13

- Lets increase ω to 100

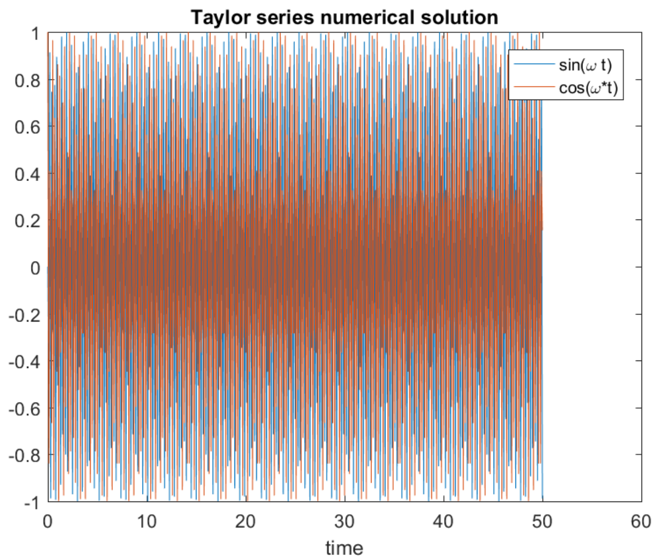

ode23: Circuit test - phase plane
cos(ω*t)
sin(ω*t)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

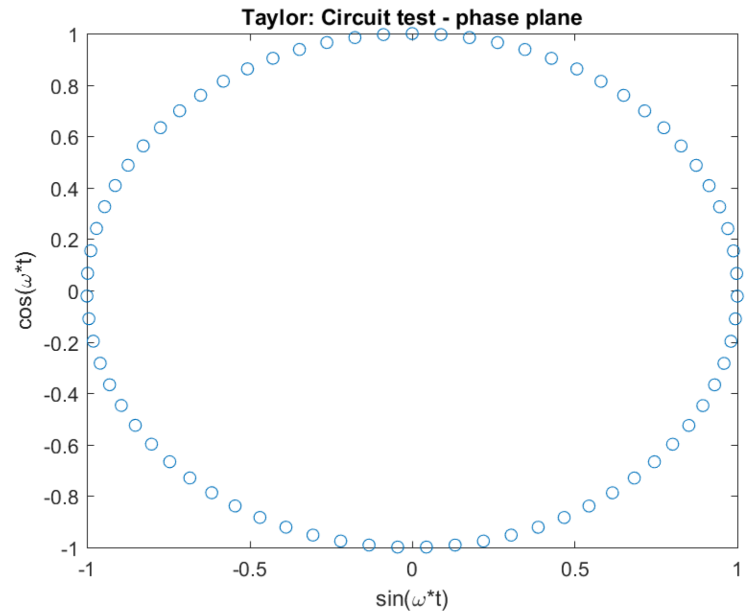

MATLAB - numerical solution ode23
sin(ω t)
cos(ω*t)
time
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
10
20
30
40
50

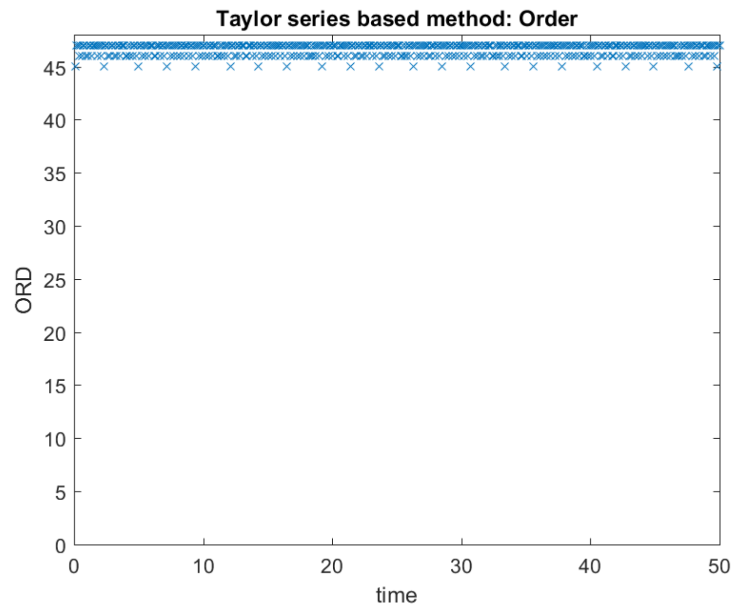

ode45: Circuit test - phase plane
cos(ω*t)
sin(ω*t)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1


MATLAB - numerical solution ode45
sin(ω t)
cos(ω*t)
time
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
10
20
30
40
50

Example 1 – circuit test, $\omega = 100$







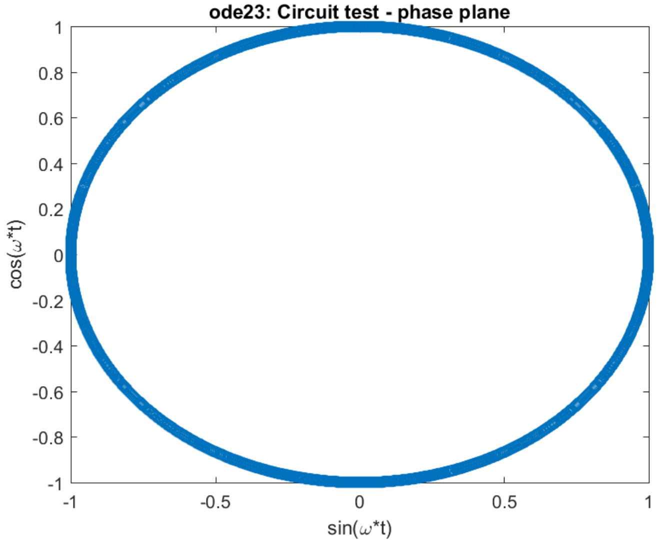
- MTSM (h=0.1)
- ORD≈47

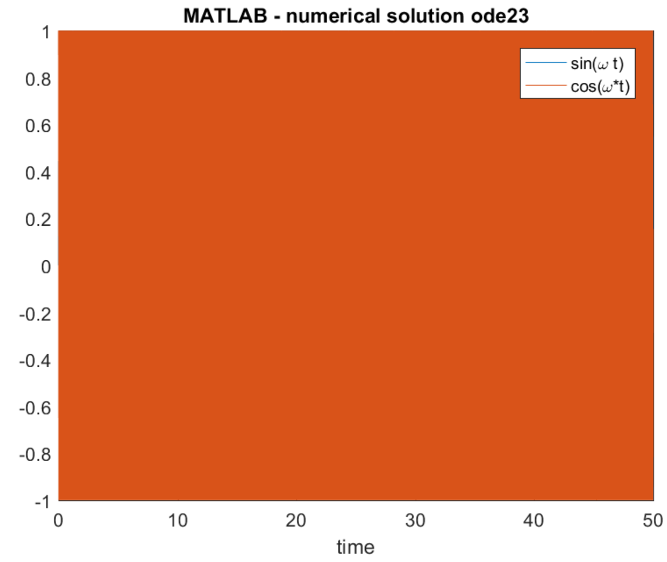
stable and fast solution

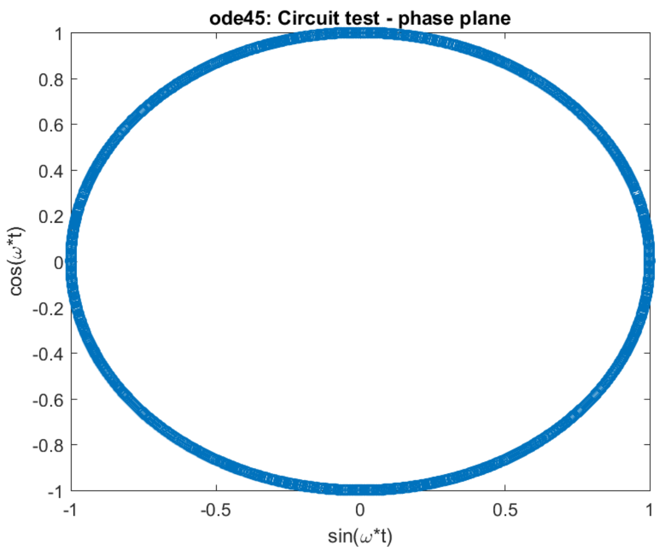
- Matlab solvers (with default settings) don't get accurate solution

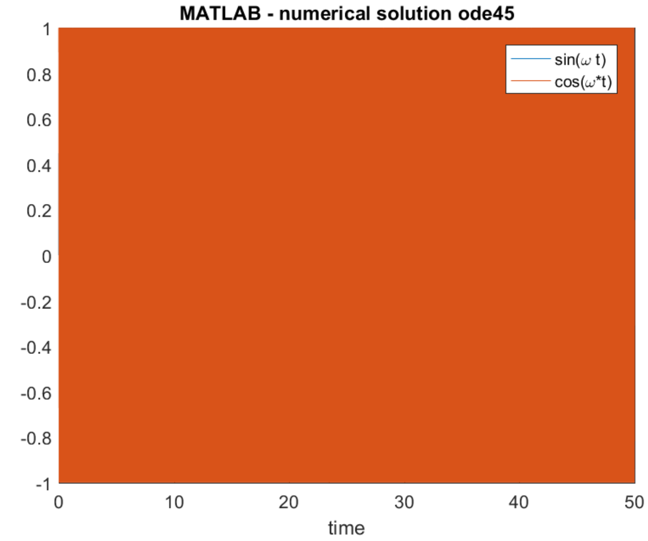
- Let's try to increase the precision of the MATLAB solvers

Example 1 – circuit test, $\omega = 100$, $RelTol = 10^{-10}$







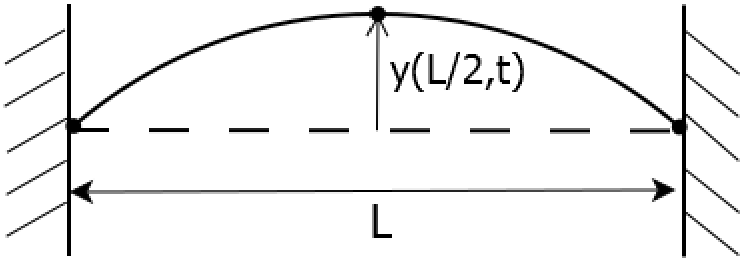


Example 1 – circuit test, $\omega = 100, RelTol = 10^{-10}$

- $\omega = 100,\ tmax = 50,\ RelTol = 10^{-10}$

Method	Time [s]	Steps	||Error||
ode23	153	3 573 706	8.12405e-07
ode45	15	888 829	8.93866e-08
MTSM	0.102145	500	4.88108e-10

Example 2 – wave equation



- Hyperbolic partial diff.eq. (1D)

$$\frac{\partial^2 y(x,t)}{\partial t^2} - \frac{\partial^2 y(x,t)}{\partial x^2} = 0$$

$$0 \leq x \leq L,\ L = 1$$
$$0 \leq t \leq Tmax$$

- Dirichlet boundary conditions:

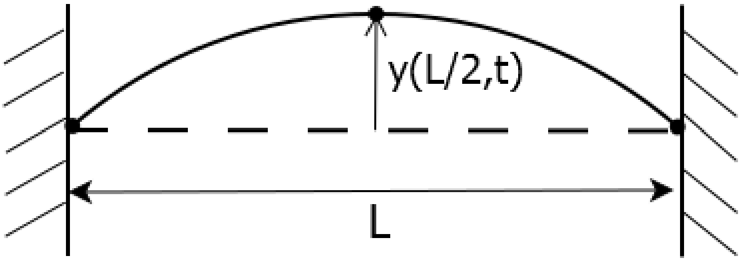
$$y(0,t) = y(L,t) = 0\,,\ \ 0 \leq t \leq Tmax$$

- Initial conditions:

$$\begin{aligned} y(x,0) &= \sin(\pi x)\,,\ \ 0 \leq x \leq L \\ \frac{\partial y(x,0)}{\partial t} &= 0 \end{aligned}$$

Example 2 – wave equation

- Hyperbolic partial diff.eq. (1D)



$$\frac{\partial^2 y(x,t)}{\partial t^2} - \frac{\partial^2 y(x,t)}{\partial x^2} = 0$$

- Dirichlet boundary conditions:

$$0 \le x \le L,\ L = 1$$
$$0 \le t \le Tmax$$

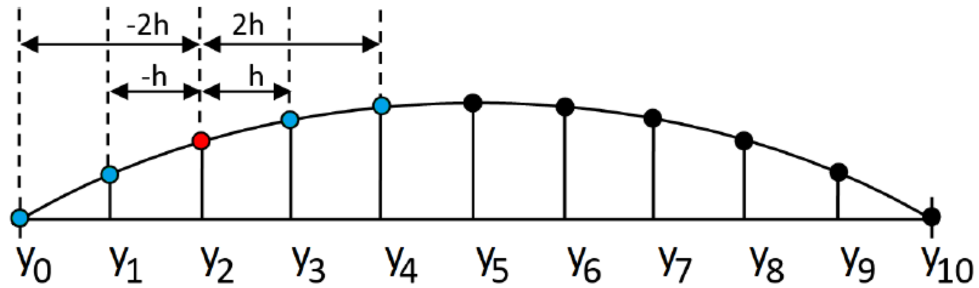
$$y(0,t) = y(L,t) = 0\,,\ \ 0 \le t \le Tmax$$

- Initial conditions:

$$\begin{aligned} y(x,0) &= \sin(\pi x)\,,\ \ 0 \le x \le L \\ \frac{\partial y(x,0)}{\partial t} &= 0 \end{aligned}$$

Analytic solution:

$$\begin{aligned} y &= \sin(\pi x)\cos(\pi t) \\ \frac{\partial y}{\partial t} &= -\pi \sin(\pi x)\sin(\pi t) \\ \frac{\partial^2 y}{\partial t^2} &= -\pi^2 \sin(\pi x)\cos(\pi t) \end{aligned}$$



- Central difference formula for $y_k = y_2$

$$
\begin{aligned}
y_{k-2} &= y_k + (-2h)y'_k + \frac{(-2h)^2}{2!}y''_k + \frac{(-2h)^3}{3!}y'''_k + \frac{(-2h)^4}{4!}y''''_k, \\
y_{k-1} &= y_k + (-h)y'_k + \frac{(-h)^2}{2!}y''_k + \frac{(-h)^3}{3!}y'''_k + \frac{(-h)^4}{4!}y''''_k, \\
y_{k+1} &= y_k + hy'_k + \frac{h^2}{2!}y''_k + \frac{h^3}{3!}y'''_k + \frac{h^4}{4!}y''''_k, \\
y_{k+2} &= y_k + 2hy'_k + \frac{(2h)^2}{2!}y''_k + \frac{(2h)^3}{3!}y'''_k + \frac{(2h)^4}{4!}y''''_k.
\end{aligned}
$$

$$\begin{pmatrix} DY1 \\ DY2 \\ DY3 \\ DY4 \end{pmatrix} = \begin{pmatrix} -2 & (-2)^2 & (-2)^3 & (-2)^4 \\ -1 & (-1)^2 & (-1)^3 & (-1)^4 \\ 1 & 1^2 & 1^3 & 1^4 \\ 2 & 2^2 & 2^3 & 2^4 \end{pmatrix}^{-1} \cdot \begin{pmatrix} y_{k-2} - y_k \\ y_{k-1} - y_k \\ y_{k+1} - y_k \\ y_{k+2} - y_k \end{pmatrix}$$

$$y_k{}'' = DY2_k(y_{k-2}, y_{k-1}, y_k, y_{k+1}, y_{k+2}) \cdot \frac{2}{h^2}$$

$$\vec{y}'' = A_{approx} \cdot \vec{y}$$

$$\begin{pmatrix} DY1 \\ DY2 \\ DY3 \\ DY4 \end{pmatrix} = \begin{pmatrix} -2 & (-2)^2 & (-2)^3 & (-2)^4 \\ -1 & (-1)^2 & (-1)^3 & (-1)^4 \\ 1 & 1^2 & 1^3 & 1^4 \\ 2 & 2^2 & 2^3 & 2^4 \end{pmatrix}^{-1} \cdot \begin{pmatrix} y_{k-2} - y_k \\ y_{k-1} - y_k \\ y_{k+1} - y_k \\ y_{k+2} - y_k \end{pmatrix}$$

$$y_k'' = DY2_k(y_{k-2}, y_{k-1}, y_k, y_{k+1}, y_{k+2}) \cdot \frac{2}{h^2}$$

$$\vec{y}'' = A_{approx} \cdot \vec{y} \qquad \left\|Error_{space}\right\| = \left\|\frac{\partial^2 \vec{y}}{\partial x^2} - \vec{y}''\right\|$$

- Solution in time – initial value problem

$$\overrightarrow{uy}' = \mathrm{A} \cdot \overrightarrow{uy}, \qquad \overrightarrow{uy}(x, 0) = (0, \sin(\pi x))^T$$

$$A = \begin{bmatrix} \mathbf{0} & A_{approx} \\ I & \mathbf{0} \end{bmatrix}$$

- Solution in time – initial value problem

$$\overrightarrow{uy}' = \mathrm{A} \cdot \overrightarrow{uy}, \qquad \overrightarrow{uy}(x,0) = (0, \sin(\pi x))^T$$

$$A = \begin{bmatrix} \mathbf{0} & A_{approx} \\ I & \mathbf{0} \end{bmatrix}$$

$$\left\|Error_{position}\right\| = \|\vec{y}_{anal} - \vec{y}\|$$

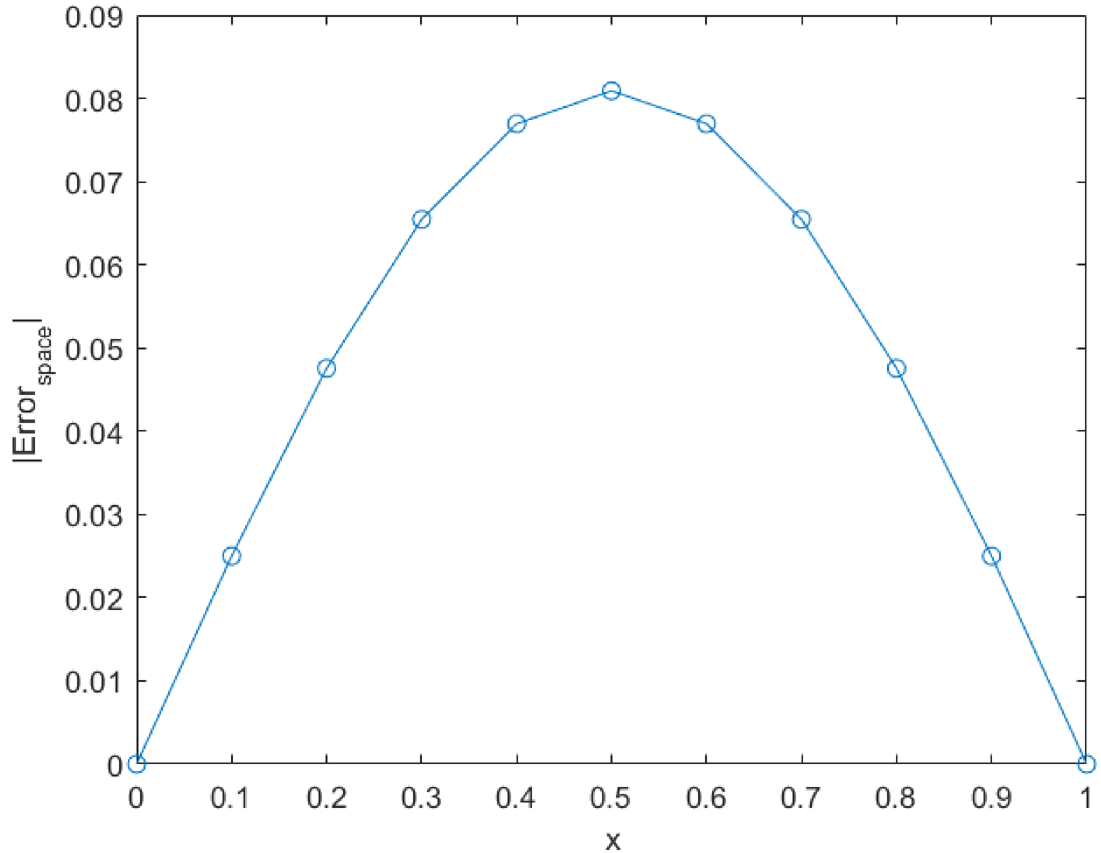
$$\left\|Error_{velocity}\right\| = \left\|\frac{\partial \vec{y}}{\partial t} - \vec{u}\right\|$$

Numerical experiments:

- Fixed number of cuts in space domain $N_{cuts} = 10$ $(h = 0.1)$
- Time of simulation $T_{max} = 10\,000$
- Compare MTSM and MATLAB ode solvers

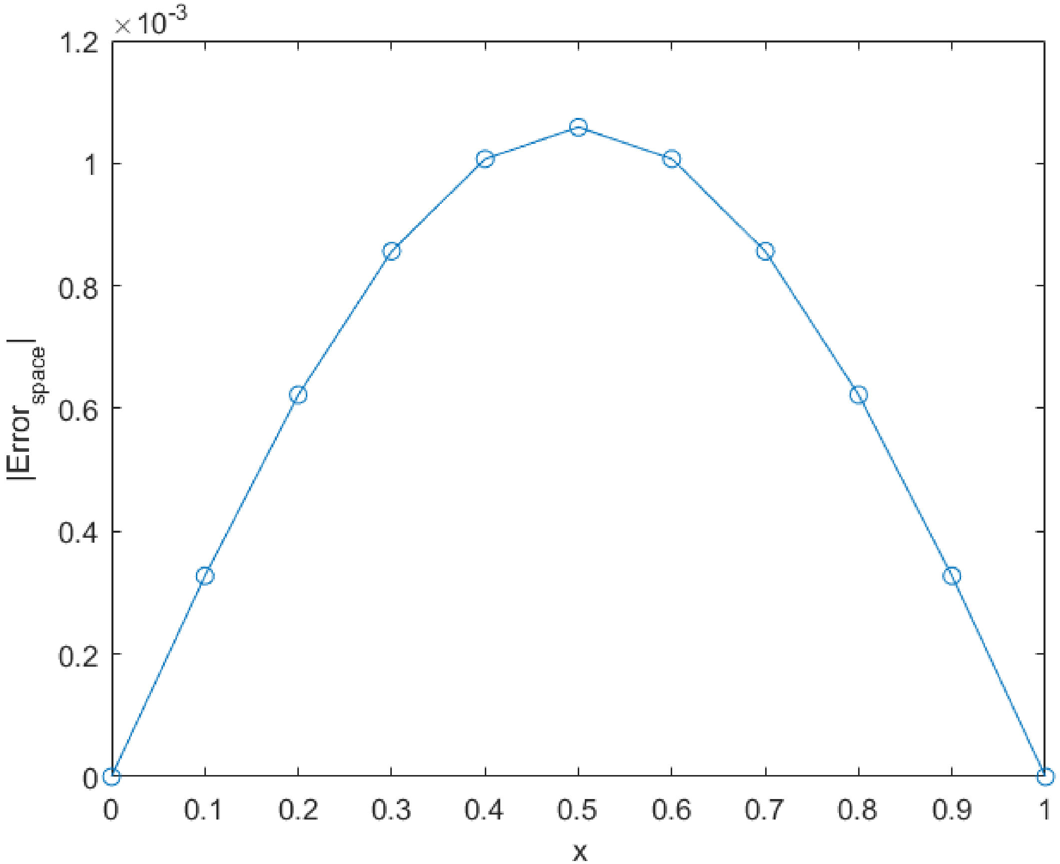
- Note: fully **explicit scheme** was used -> for spatial approximation and for solution in time

- 3-point approximation of $\frac{\partial^2 \vec{y}}{\partial x^2}$



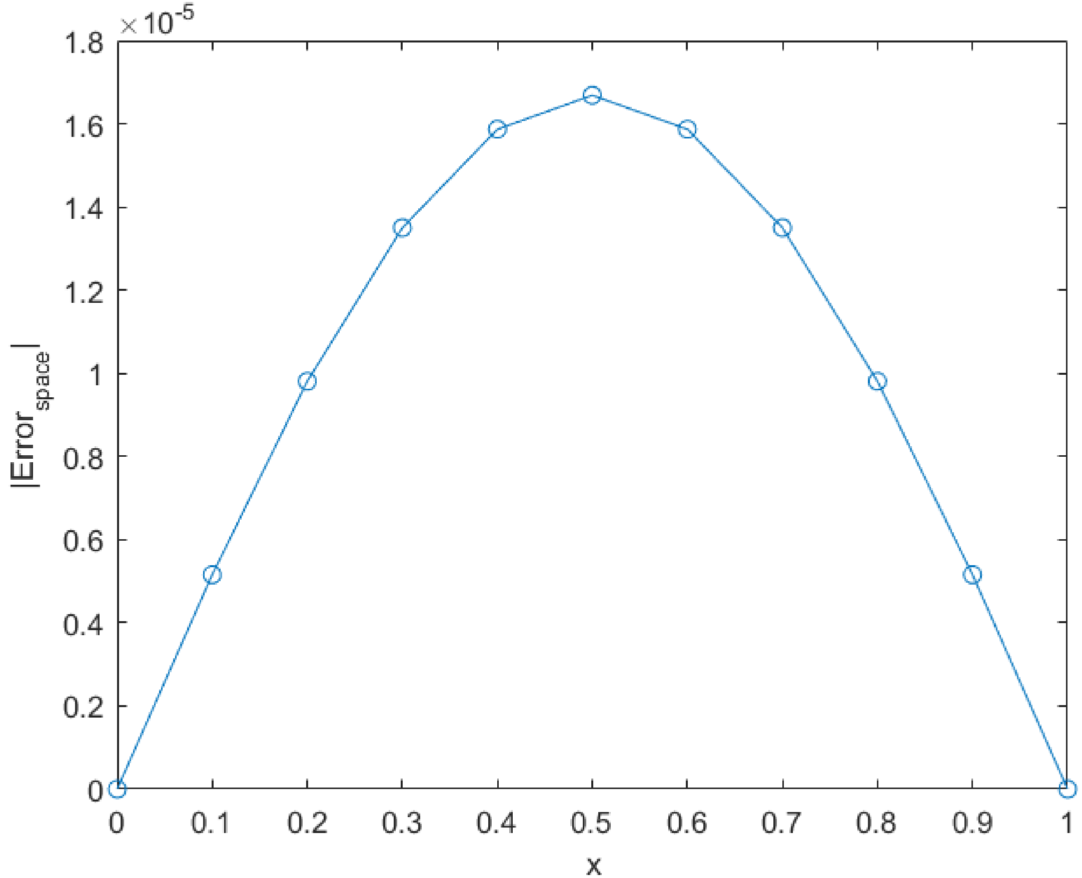
$$\left\| Error_{space} \right\| = 0.0809 \approx 10^{-1}$$

- 5-point approximation of $\frac{\partial^2 \vec{y}}{\partial x^2}$



$$\left\|Error_{space}\right\| = 0.0011 \approx 10^{-3}$$

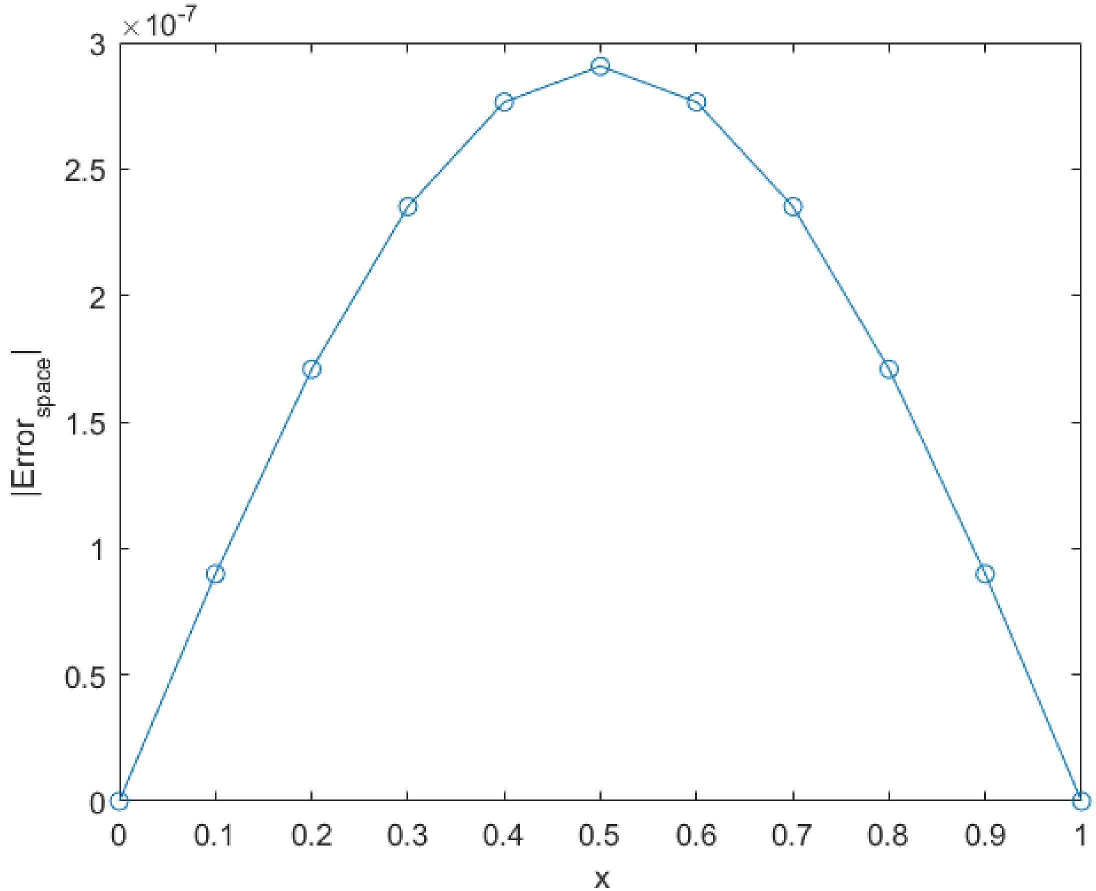
- 7-point approximation of $\frac{\partial^2 \vec{y}}{\partial x^2}$



$$\left\| Error_{space} \right\| \approx 10^{-5}$$

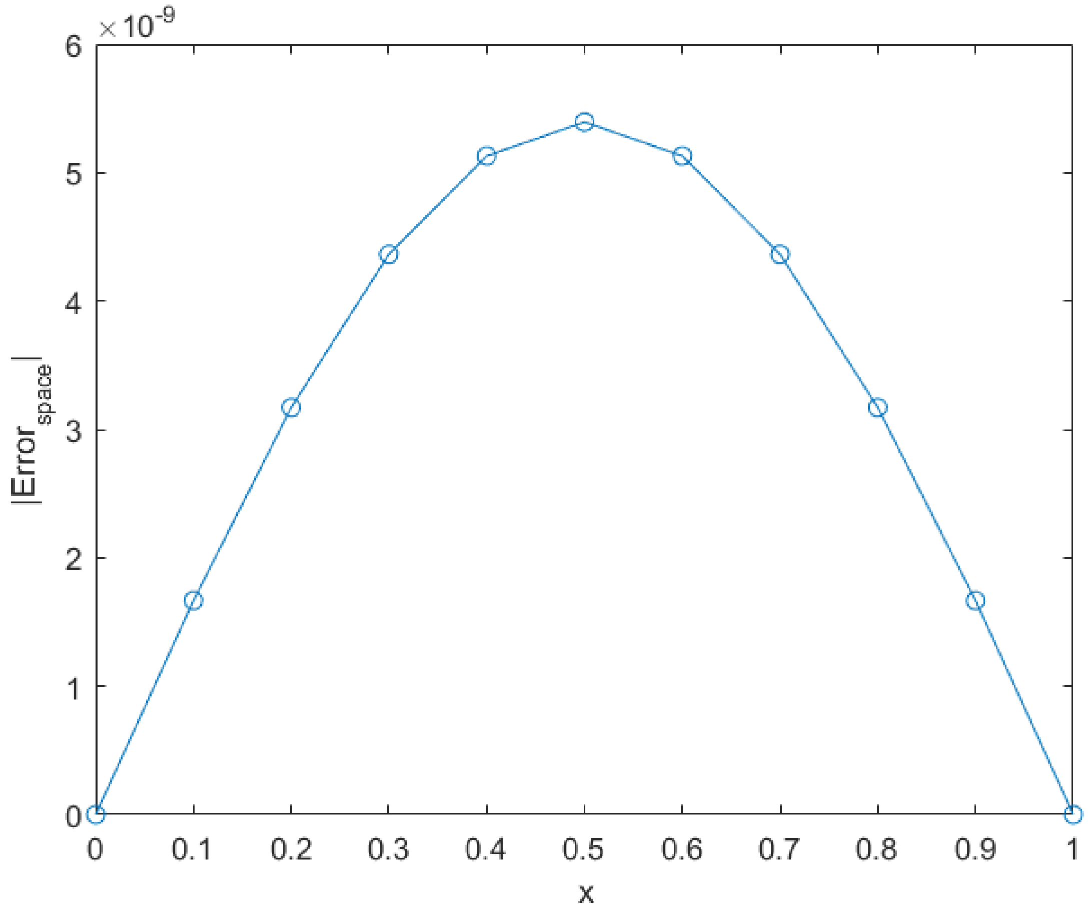
Example 2 – error in space domain

- 9-point approximation of $\frac{\partial^2 \vec{y}}{\partial x^2}$



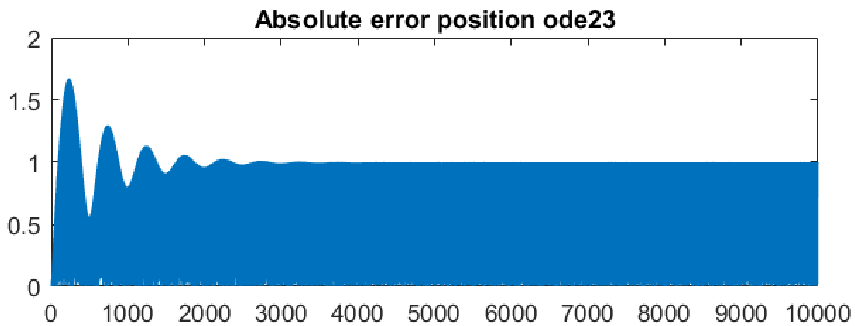
$$\left\| Error_{space} \right\| \approx 10^{-7}$$

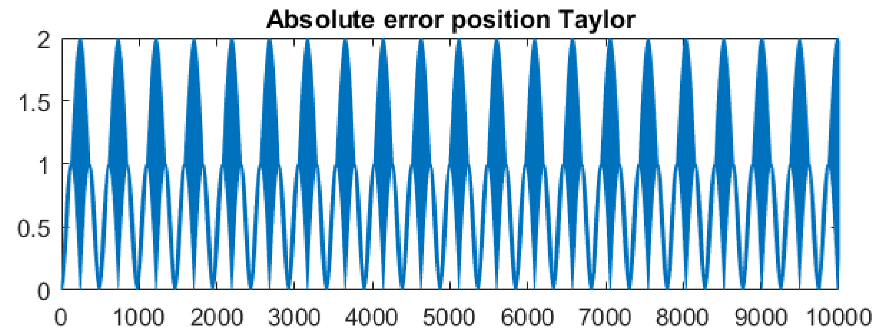
- 11-point approximation of $\frac{\partial^2 \vec{y}}{\partial x^2}$

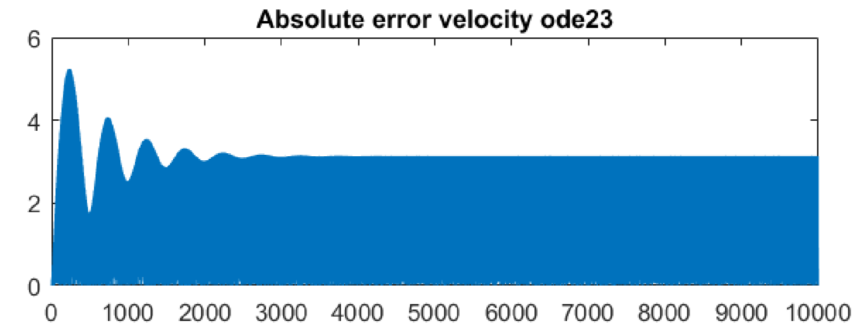


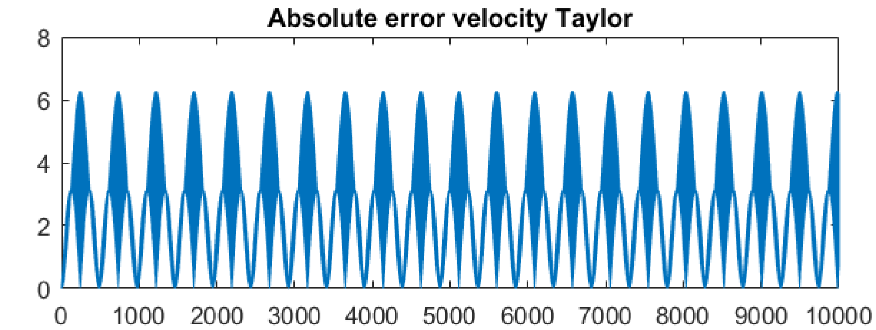
$$\left\| Error_{space} \right\| \approx 10^{-9}$$

- 3-point approximation







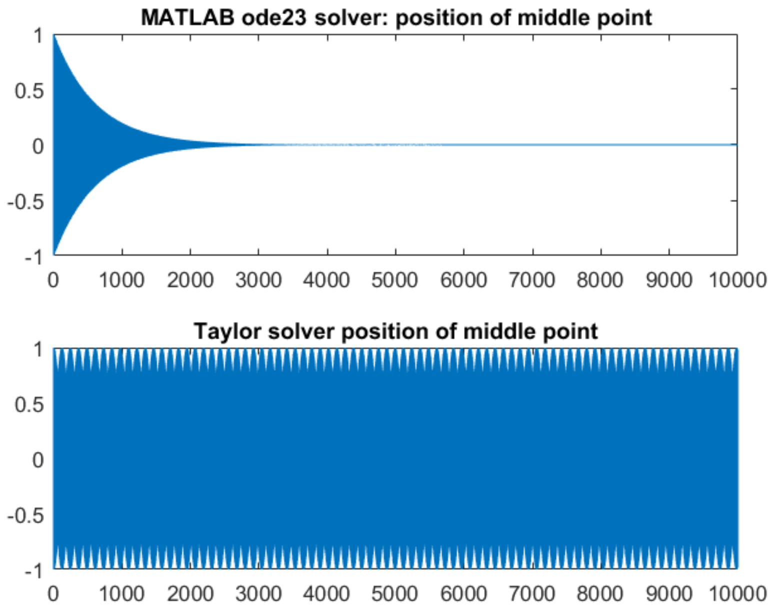


$$\|Error_{position}\|_{ode23} = 1.67 \qquad \|Error_{position}\|_{MTSM} = 2$$

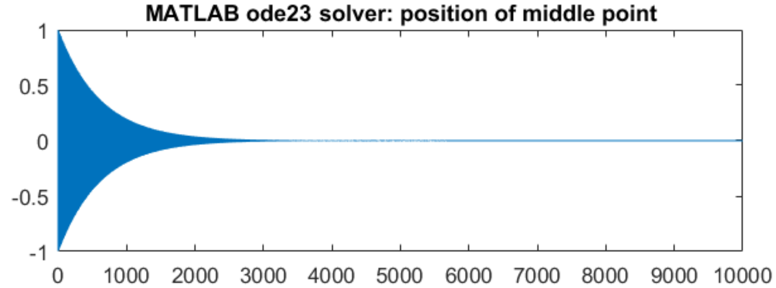
$$\|Error_{velocity}\|_{ode23} = 5.25 \qquad \|Error_{velocity}\|_{MTSM} = 6.27$$

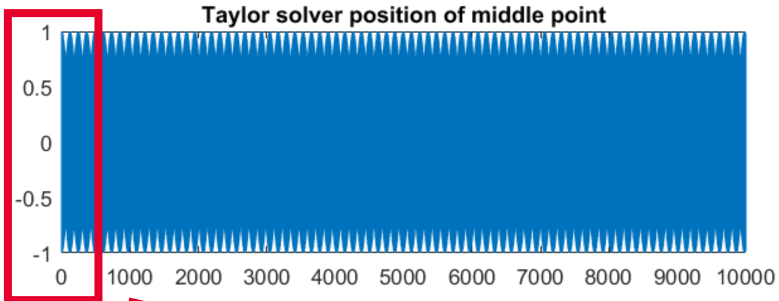
Example 2 – error in time domain

- 3-point approximation



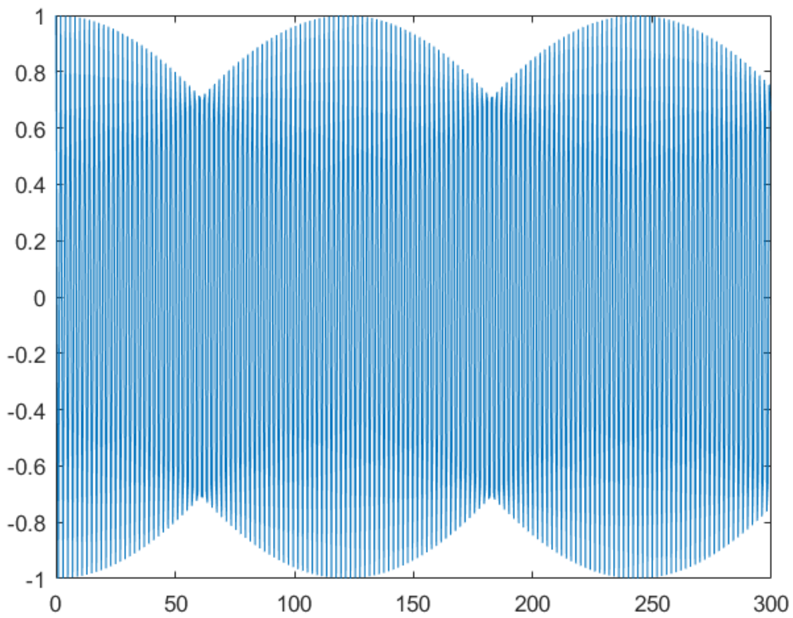
- 3-point approximation



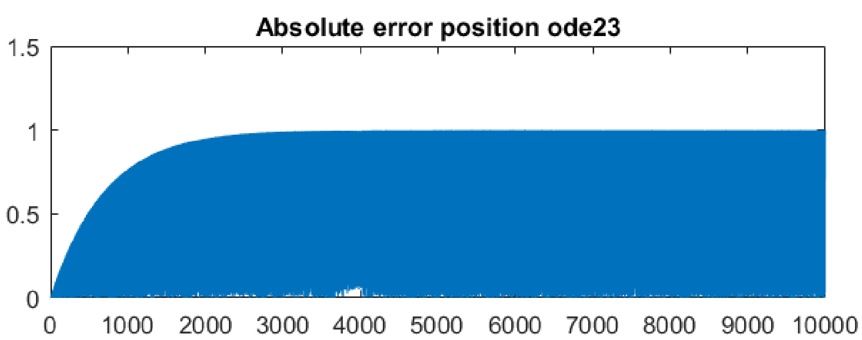


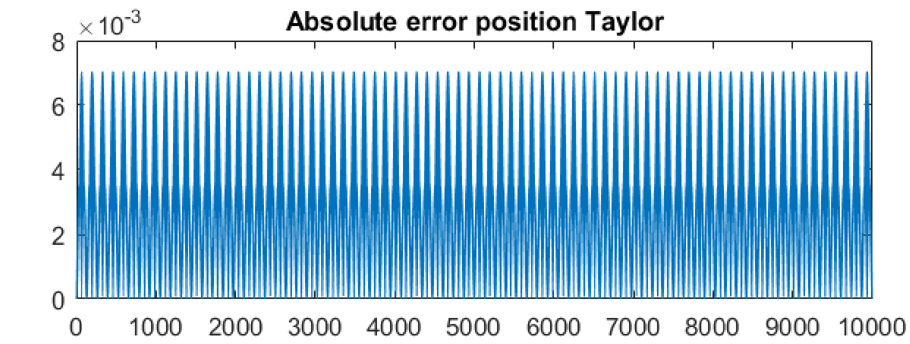
- MTSM ($dt = 0.1$)
- ORD≈30

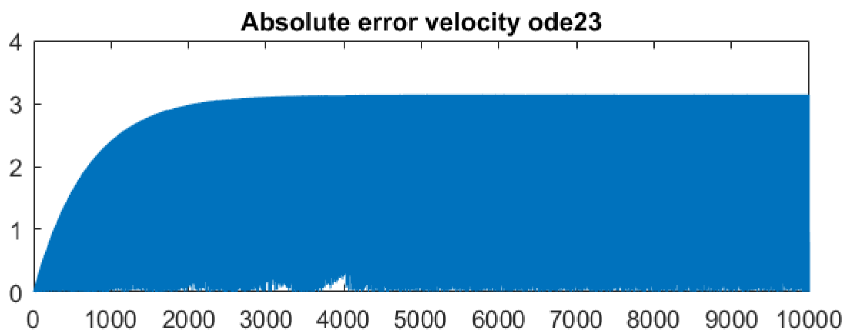
stable and fast solution

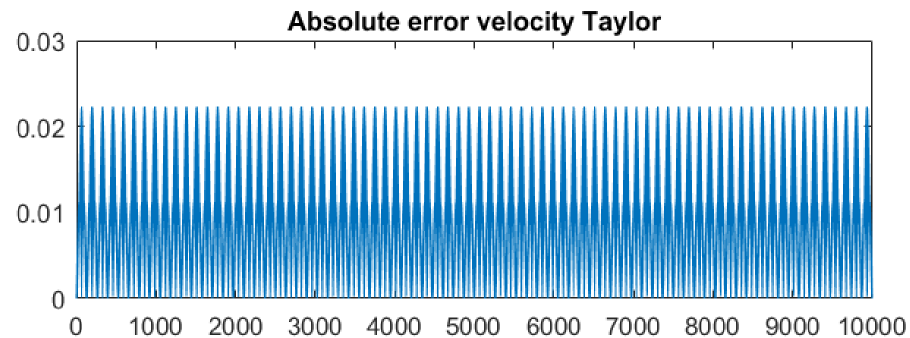


- 5-point approximation





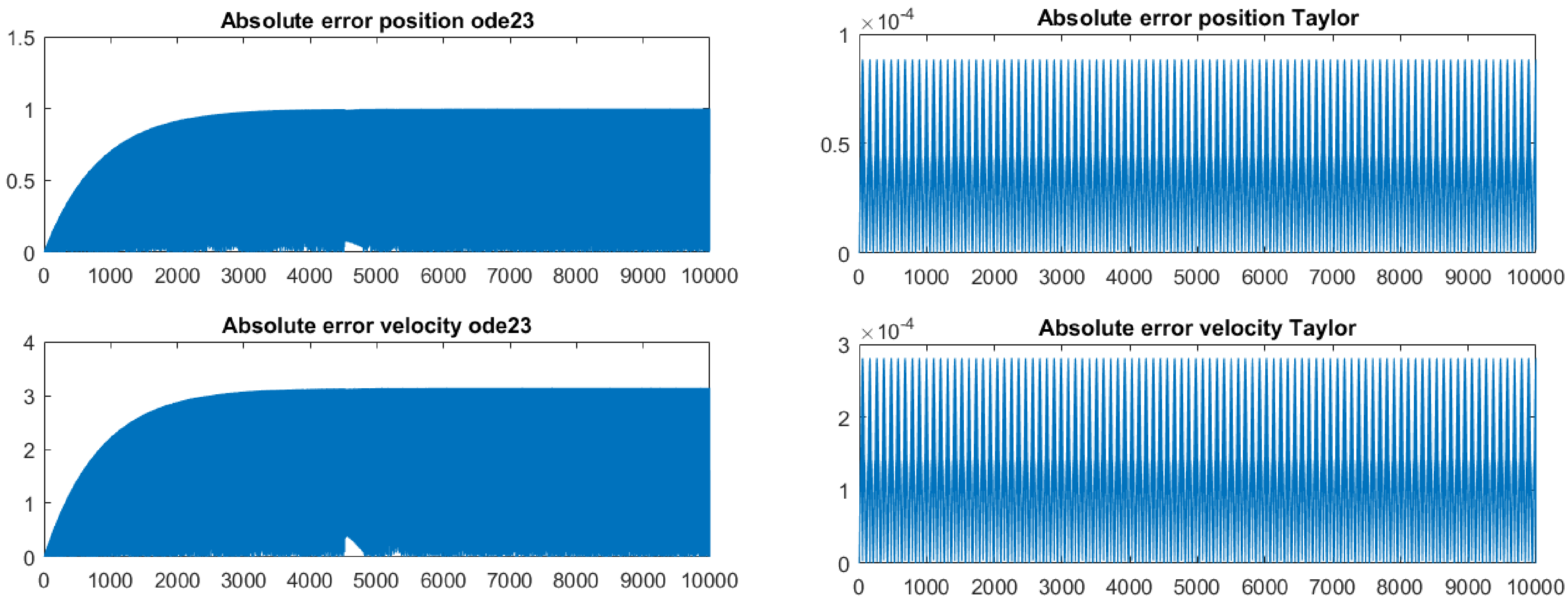




$$\left\|Error_{position}\right\|_{ode23} = 1 \qquad \left\|Error_{position}\right\|_{MTSM} = 7 \cdot 10^{-3}$$

$$\left\|Error_{velocity}\right\|_{ode23} = 3.14 \qquad \left\|Error_{velocity}\right\|_{MTSM} = 2.2 \cdot 10^{-2}$$

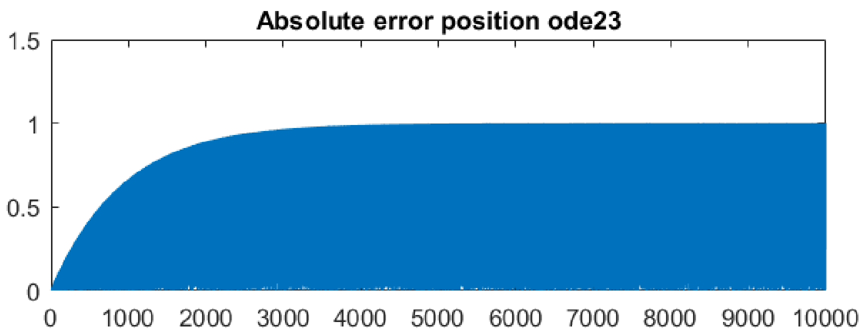
- 7-point approximation

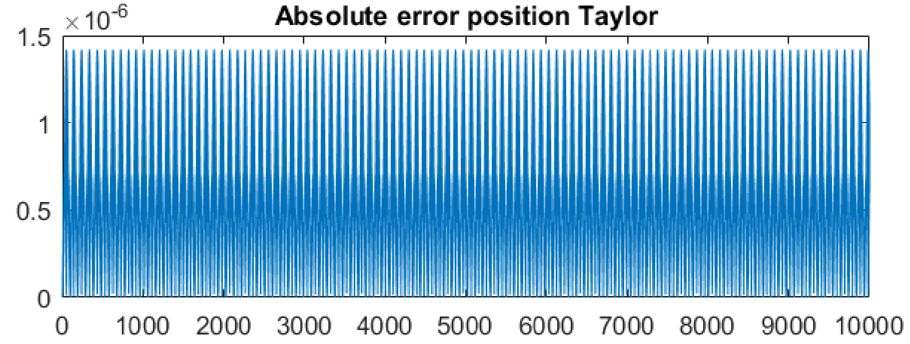


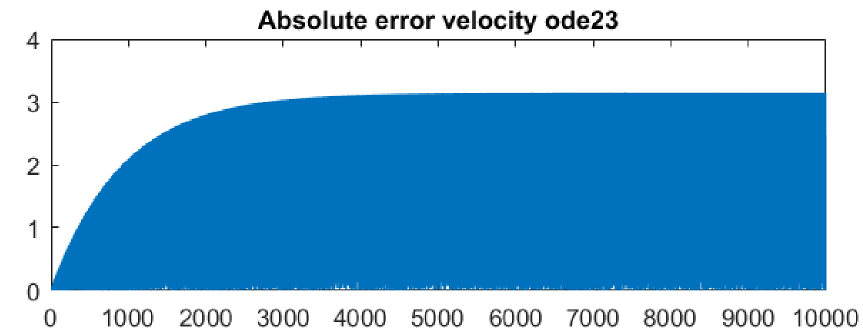
$$\left\|Error_{position}\right\|_{ode23} = 1 \qquad \left\|Error_{position}\right\|_{MTSM} = 8.8 \cdot 10^{-5}$$

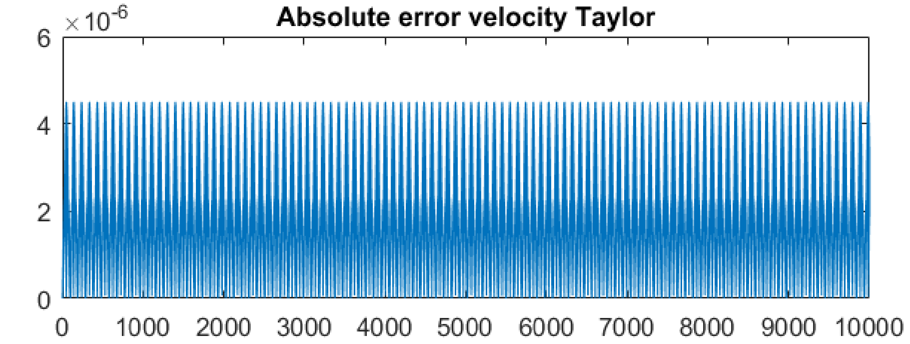
$$\left\|Error_{velocity}\right\|_{ode23} = 3.14 \qquad \left\|Error_{velocity}\right\|_{MTSM} = 2.8 \cdot 10^{-4}$$

- 9-point approximation





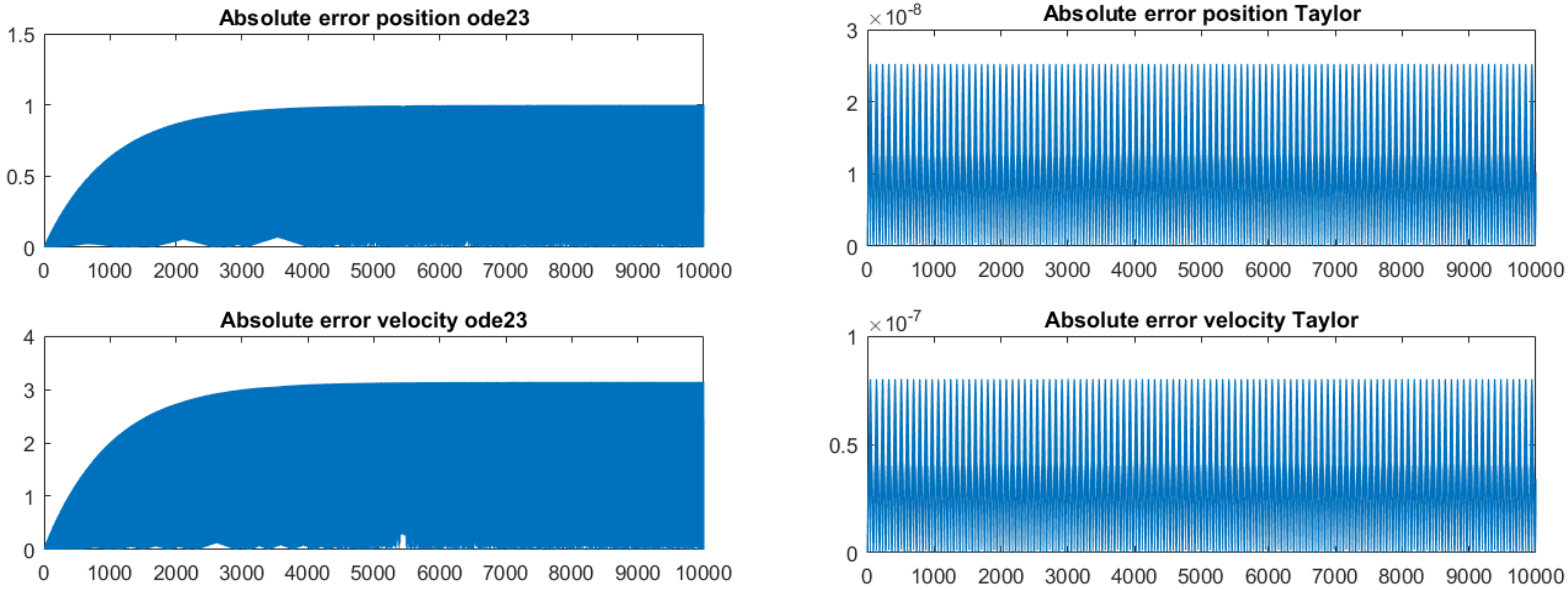




$$\left\|Error_{position}\right\|_{ode23} = 1 \qquad \left\|Error_{position}\right\|_{MTSM} = 1.4 \cdot 10^{-6}$$

$$\left\|Error_{velocity}\right\|_{ode23} = 3.14 \qquad \left\|Error_{velocity}\right\|_{MTSM} = 4.5 \cdot 10^{-6}$$

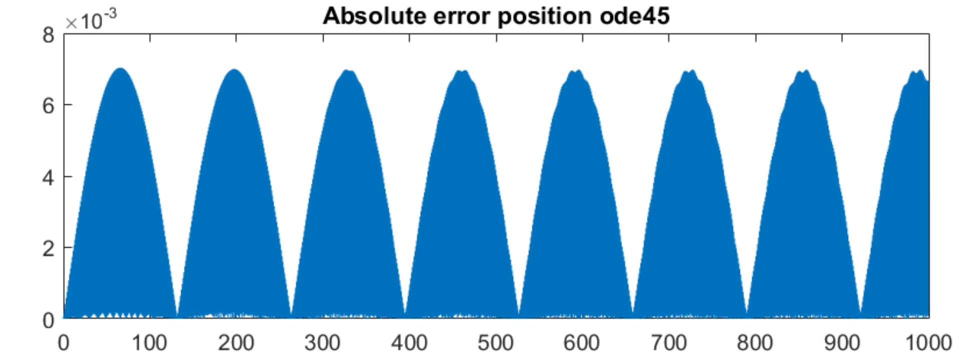
- 11-point approximation

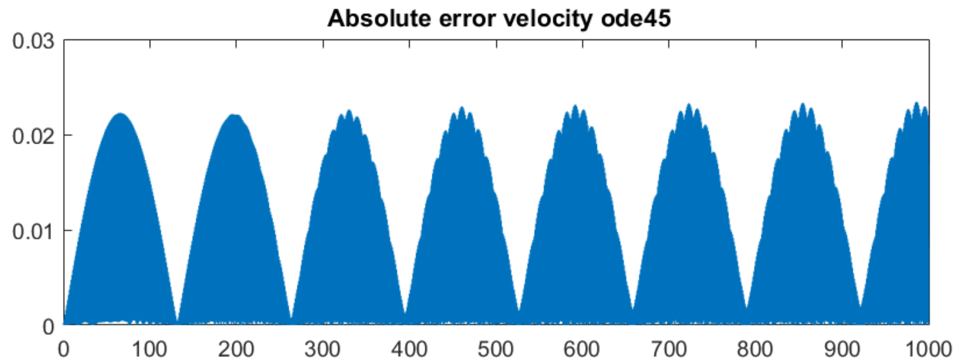


$$\|Error_{position}\|_{ode23} = 1 \qquad \|Error_{position}\|_{MTSM} = 2.5 \cdot 10^{-8}$$

$$\|Error_{velocity}\|_{ode23} = 3.14 \qquad \|Error_{velocity}\|_{MTSM} = 8 \cdot 10^{-8}$$

- 5-point approximation (ode45, $Tol = 10^{-6}$)





$$\left\|Error_{position}\right\|_{ode45} = 0.007$$

$$\left\|Error_{velocity}\right\|_{ode45} = 0.023$$

- 5-point approximation (ode45, $Tol = 10^{-6}$)

$$T_{max} = 100$$

Method	Time [s]	Steps	$\|\|Error_{position}\|\|$	$\|\|Error_{velocity}\|\|$
ode45	12.8	6364	0.007	0.022
MTSM	0.07	200	0.007	0.022

Example 2 – stable solution in MATLAB

- 5-point approximation (ode45, $Tol = 10^{-6}$)

$$T_{max} = 100$$

Method	Time [s]	Steps	$\|\|Error_{position}\|\|$	$\|\|Error_{velocity}\|\|$
ode45	12.8	6364	0.007	0.022
MTSM	0.07	200	0.007	0.022

$$T_{max} = 1000$$

Method	Time [s]	Steps	$\|\|Error_{position}\|\|$	$\|\|Error_{velocity}\|\|$
ode45	66012.4 (18.3h)	83756	0.007	0.023
MTSM	0.863	2000	0.007	0.022

Conclusion

- Implementation of MTSM in MATLAB for linear systems of ODEs - fast and stable solution

- Next step will be implementation of nonlinear systems of ODEs in MATLAB using matrix-vector computation in MTSM

Thank you for your attention!