Mixed precision implicit numerical methods for solving large systems of ordinary differential equations

M. Al Sayed Ali, S. Bernard, A. Marzorati, J. Rouzaud Cornabas

IRMAR-Université de Rennes

CANUM2026, Saint-Jacut-de-la-mer, 4 Juin 2026

Introduction

Comparison of floating-point precisions	
Single precision (FP32)	**Double precision (FP64)**
· 32 bit memory to represent a value.	· 64 bit memory to represent a value.
· 7 decimal digits of precision	· 16 decimal digits of precision
· Machine precision $\approx 10^{-8}$	· Machine precision $\approx 10^{-16}$

Introduction

Comparison of floating-point precisions	
Single precision (FP32)	**Double precision (FP64)**
· 32 bit memory to represent a value.	· 64 bit memory to represent a value.
· 7 decimal digits of precision	· 16 decimal digits of precision
· Machine precision $\approx 10^{-8}$	· Machine precision $\approx 10^{-16}$

- Mixed precision = Single precision + Double precision

Introduction

Comparison of floating-point precisions	
Single precision (FP32)	**Double precision (FP64)**
· 32 bit memory to represent a value.	· 64 bit memory to represent a value.
· 7 decimal digits of precision	· 16 decimal digits of precision
· Machine precision $\approx 10^{-8}$	· Machine precision $\approx 10^{-16}$

- Mixed precision = Single precision + Double precision
- Numerical methods with mixed precision run faster and use less memory but with lost of accuracy

Introduction

Comparison of floating-point precisions	
Single precision (FP32)	**Double precision (FP64)**
· 32 bit memory to represent a value.	· 64 bit memory to represent a value.
· 7 decimal digits of precision	· 16 decimal digits of precision
· Machine precision $\approx 10^{-8}$	· Machine precision $\approx 10^{-16}$

- Mixed precision = Single precision + Double precision
- Numerical methods with mixed precision run faster and use less memory but with lost of accuracy
- Domain of applications : AI, Climate/weather modeling, Computational fluid dynamics, ...

Ordinary differential equation (ODE)

Consider the following ordinary differential equation (ODE) :

$$\begin{cases} \dot{y}(t) = f(t, y(t)) \quad \forall \quad t \in [t_0, T] \\ y(t_0) = y^{(0)} \end{cases} \tag{1}$$

where

- $y^{(0)}$ is a vector of $\mathbb{R}^n$.
- $f : [t_0, T] \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is a smooth function.

Numerical methods

- Discretization in time $\Longrightarrow$

$$y_{i+1} = a_i + h\beta_0 f(t_{i+1}, y_{i+1}) \qquad i = q, \ldots, N-1.$$

Numerical methods

- Discretization in time $\Longrightarrow$

$$y_{i+1} = a_i + h\beta_0 f(t_{i+1}, y_{i+1}) \qquad i = q, \ldots, N-1.$$

where

- h is the time step.
- $y_i \approx y(t_i)$, with $t_i = t_0 + ih$.
- $a_i = F(y_{i-q}, y_{i-(q-1)}, \ldots, y_i, f) \in \mathbb{R}^n$.
- β_0 is a **nonzero** constant.
- $y_N \approx y(T)$.
- p is the order of the numerical method.

Implicit methods for solving the ODE

- Numerical method $\Longrightarrow G_i(y_{i+1}) = 0, \qquad i = q, \ldots, N-1,$

where

$$G_i(y) = y - a_i - \beta_0 h f(t_{i+1}, y).$$

Mixed precision approach

We define the function $\mathrm{fl}(y)$ by

$$\begin{aligned} \mathrm{fl} : \mathbb{R}^n &\to \mathbb{S}^n \\ y &\to \mathrm{fl}(y) \end{aligned}$$

such that

$$\|\mathrm{fl}(y) - y\| \leq \varepsilon \|y\|, \tag{2}$$

where ε the machine epsilon ($\approx 10^{-8}$), and $\mathbb{S}^n$ is the set of single precision vectors in $\mathbb{R}^n$.

Mixed Method MAD :

This mixed method is defined by the equation

$$\tilde{G}_i(\tilde{y}_{i+1}) = 0, \quad i = q, \ldots, N-1, \tag{3}$$

where

$$\tilde{G}_i(y) = y - a_i - \beta_0 h f^{(L)}(t_{i+1}, y),$$

with

$$f^{(L)}(t, x) = \mathrm{fl}(f)(\mathrm{fl}(t), \mathrm{fl}(x))$$

which is computed by performing all arithmetic operations in the function f in a single precision.

SINGLE and DOUBLE methods

- **SINGLE method :** The numerical method y_i is entirely computed in **single** precision.

- **DOUBLE method ($\equiv y_i$):** The numerical method y_i is entirely computed in **double** precision.

Theorem (Error analysis of the mixed method MAD)

Assume that $f(t, y)$ is Lipschitz with constant c_0, and that

$$h \leq \alpha_0 < \frac{1}{|\beta_0|\, c_0(1+\varepsilon)}, \quad \alpha_0 > 0.$$

Then :

1. *The solution $\tilde{y}_i$ computed by MAD satisfies*

$$\max_{0 \leq i \leq N} \|y_i - \tilde{y}_i\| = \mathcal{O}\big(\varepsilon(1 + h^p)\big).$$

Theorem (Error analysis of the mixed method MAD)

Assume that $f(t, y)$ is Lipschitz with constant c_0, and that

$$h \leq \alpha_0 < \frac{1}{|\beta_0|\, c_0(1+\varepsilon)}, \quad \alpha_0 > 0.$$

Then :

1. *The solution $\tilde{y}_i$ computed by MAD satisfies*

$$\max_{0 \leq i \leq N} \|y_i - \tilde{y}_i\| = \mathcal{O}\big(\varepsilon(1 + h^p)\big).$$

2. *The solution $\tilde{y}_i$ computed by SINGLE method satisfies*

$$\max_{0 \leq i \leq N} \|y_i - \tilde{y}_i\| = \mathcal{O}\left(\frac{\varepsilon}{h}(1 + h^p)\right).$$

Newton method

1. The nonlinear system $\tilde{G}_i(\tilde{y}_{i+1}) = 0$ is solved by the Newton method.

Newton method

1. The nonlinear system $\tilde{G}_i(\tilde{y}_{i+1}) = 0$ is solved by the Newton method.
2. The Newton method computes a sequence $(y_i^{(m)})$ such that :
   - $y_i^{(m+1)} = y_i^{(m)} + z_i^{(m)}$

Newton method

1. The nonlinear system $\tilde{G}_i(\tilde{y}_{i+1}) = 0$ is solved by the Newton method.
2. The Newton method computes a sequence $(y_i^{(m)})$ such that :
   - $y_i^{(m+1)} = y_i^{(m)} + z_i^{(m)}$
   - $\tilde{G}_i'(y_i^{(m)}) z_i^{(m)} = -\tilde{G}_i(y_i^{(m)})$,

Newton method

1. The nonlinear system $\tilde{G}_i(\tilde{y}_{i+1}) = 0$ is solved by the Newton method.
2. The Newton method computes a sequence $(y_i^{(m)})$ such that :
   - $y_i^{(m+1)} = y_i^{(m)} + z_i^{(m)}$
   - $\tilde{G}_i'(y_i^{(m)}) z_i^{(m)} = -\tilde{G}_i(y_i^{(m)})$,
   - $\lim_{m \to +\infty} y_i^{(m)} = \tilde{y}_{i+1}$ when $y_i^{(0)} \approx \tilde{y}_{i+1}$.

Newton method

1. The nonlinear system $\tilde{G}_i(\tilde{y}_{i+1}) = 0$ is solved by the Newton method.
2. The Newton method computes a sequence $(y_i^{(m)})$ such that :
   - $y_i^{(m+1)} = y_i^{(m)} + z_i^{(m)}$
   - $\tilde{G}_i'(y_i^{(m)}) z_i^{(m)} = -\tilde{G}_i(y_i^{(m)})$,
   - $\lim_{m \to +\infty} y_i^{(m)} = \tilde{y}_{i+1}$ when $y_i^{(0)} \approx \tilde{y}_{i+1}$.
3. To compute good initial guess $y_i^{(0)}$ we use the Line-search or trust region algorithm.

Line-search and trust-region algorithms for computing $y_i^{(0)}$

- They compute a **local** minimum of $h_i(y) = \frac{1}{2}\|\tilde{G}_i(y)\|^2$. (**global** minimum is $\tilde{y}_{i+1}$)

Line-search and trust-region algorithms for computing $y_i^{(0)}$

- They compute a **local** minimum of $h_i(y) = \frac{1}{2}\|\tilde{G}_i(y)\|^2$. (**global** minimum is $\tilde{y}_{i+1}$)

- They compute a sequence $(u_i^{(k)})_{k\geq 0}$ s.t. $\nabla h_i(u_i^{(k)}) \underset{k\to+\infty}{\to} 0$.

Line-search and trust-region algorithms for computing $y_i^{(0)}$

- They compute a **local** minimum of $h_i(y) = \frac{1}{2}\|\tilde{G}_i(y)\|^2$. (**global** minimum is $\tilde{y}_{i+1}$)
- They compute a sequence $(u_i^{(k)})_{k\geq 0}$ s.t. $\nabla h_i(u_i^{(k)}) \underset{k\to+\infty}{\to} 0$.
- $y_i^{(0)} = u_i^{(Kmax)}$ where $Kmax \in \mathbb{N}$.

Line-search algorithm for computing $y_i^{(0)}$

- It computes the sequence $u_i^{(k)}$ given by $u_i^{(k+1)} = u_i^{(k)} + \lambda_k p_i^{(k)}$ where $p_i^{(k)}$ solves

$$(\tilde{G}_i'(u_i^{(k)}))p_i^{(k)} = -\tilde{G}_i(u_i^{(k)}).$$

Line-search algorithm for computing $y_i^{(0)}$

- It computes the sequence $u_i^{(k)}$ given by $u_i^{(k+1)} = u_i^{(k)} + \lambda_k p_i^{(k)}$ where $p_i^{(k)}$ solves

$$(\tilde{G}_i'(u_i^{(k)}))p_i^{(k)} = -\tilde{G}_i(u_i^{(k)}).$$

- λ_k is a positive scalar chosen to satisfy the Goldstein-Armijo condition:

$$h_i(u_i^{(k+1)}) \leq h_i(u_i^{(k)}) + \alpha\lambda_k \nabla h_i(u_i^{(k)})^T p_i^{(k)} \qquad (4)$$

where $\alpha \in (0, 1/2)$ is a parameter typically set to 10^{-4}.

Solution of the linear systems $\tilde{G}_i'(y_i^{(m)})z_i^{(m)} = -\tilde{G}_i(y_i^{(m)})$ and $(\tilde{G}_i'(u_i^{(k)}))p_i^{(k)} = -\tilde{G}_i(u_i^{(k)})$.

Solution of the linear systems $\tilde{G}_i'(y_i^{(m)})z_i^{(m)} = -\tilde{G}_i(y_i^{(m)})$ and $(\tilde{G}_i'(u_i^{(k)}))p_i^{(k)} = -\tilde{G}_i(u_i^{(k)})$.

- We use the GMRES method, which computes $\tilde{x}$, of $Ax_* = b$, s. t.

$$\tilde{x} = x_0 + Vz,$$

where $V \in \mathbb{R}^{n\times p}$ with $p \ll n$ such that $V^T \cdot V = I_p$. The vector z solves the small system

$$\left((AV)^T AV\right) z = (AV)^T b.$$

Mixed Precision Generalized Minimal RESidual (m)–MPGMRES(m) for solving $Ax = b$ I

Input x_0 : initial guess, ϵ : convergence criterion

1: $r_0 = b - Ax_0$ **and** $\beta_0 = \|r_0\|$
2: $\beta_0^{(L)} = \mathrm{fl}(\beta_0)$ and $v_1^{(L)} = \mathrm{fl}(v_1)$ where $v_1 = r_0/\beta_0$
3: Compute $V_{m+1}^{(L)}$ by the m-step Arnoldi process with A and $v_1^{(L)}$.
4: Compute $y_m^{(L)}$. {using low precision}
5: $z_m^{(L)} = V_m^{(L)} y_m^{(L)}$ {using low precision}
6: $x_m = x_0 + Std(z_m^{(L)})$ {convert to standard precision}
7: $x_0 = x_m$
8: **EndIf**
9: **until** attain the maximum number of total inner iterations

Output: $x^{(M)} := x_0$

Since $A = \tilde{G}_i'(u_i^{(k)})$ then :

$$A \cdot v \approx \frac{\mathrm{fl}(\tilde{G}_i(\mathrm{fl}(u_i^{(k)} + \sigma v)) - \mathrm{fl}(\tilde{G}_i(\mathrm{fl}(u_i^{(k)})))}{\sigma}$$

Trust-region algorithms for computing $y_i^{(0)}$

- It computes $(u_i^{(k)})$ given by $u_i^{(k+1)} = u_i^{(k)} + s_i^{(k)}$ where $s_i^{(k)}$ solves

$$arg \min_{\|s\| \leq \delta_i^{(k)}} T(s), \tag{5}$$

with $T(s) = \frac{1}{2}\|\tilde{G}_i(u_i^{(k)}) + \tilde{G}_i'(u_i^{(k)})s\|^2$ and $\delta_i{}^{(k)} > 0$.

Trust-region algorithms for computing $y_i^{(0)}$

- It computes $(u_i^{(k)})$ given by $u_i^{(k+1)} = u_i^{(k)} + s_i^{(k)}$ where $s_i^{(k)}$ solves

$$arg \min_{\|s\| \leq \delta_i^{(k)}} T(s), \tag{5}$$

with $T(s) = \frac{1}{2}\|\tilde{G}_i(u_i^{(k)}) + \tilde{G}_i'(u_i^{(k)})s\|^2$ and $\delta_i{}^{(k)} > 0$.

- The point $u_i^{(k+1)}$ satisfies

$$h_i(u_i^{(k+1)}) \leq h_i(u_i^{(k)}) + \alpha \nabla h_i(u_i^{(k)})^T (u_i^{(k+1)} - u_i^{(k)}), \tag{6}$$

where $\alpha \in (0, 1/2)$.

The solution of (5) is given by

$$s_i^{(k)} = \begin{cases} -(\tilde{G}_i'(u_i^{(k)}))^{-1}\tilde{G}_i(u_i^{(k)}) & \text{if } \|(\tilde{G}_i'(u_i^{(k)}))^{-1}\tilde{G}_i(u_i^{(k)})\| \leq \delta_i{}^{(k)}, \\ -(H_i{}^{(k)} + \mu_i{}^{(k)} I_n)^{-1}\nabla h_i(u_i^{(k)}) & \text{otherwise,} \end{cases}$$

where

$$H_i{}^{(k)} = (\tilde{G}_i'(u_i^{(k)}))^T \tilde{G}_i'(u_i^{(k)}), \nabla h_i(u_i^{(k)}) = (\tilde{G}_i'(u_i^{(k)}))^T \tilde{G}_i(u_i^{(k)})$$

The solution of (5) is given by

$$s_i^{(k)} = \begin{cases} -(\tilde{G}_i'(u_i^{(k)}))^{-1}\tilde{G}_i(u_i^{(k)}) & \text{if } \|(\tilde{G}_i'(u_i^{(k)}))^{-1}\tilde{G}_i(u_i^{(k)})\| \leq \delta_i{}^{(k)}, \\ -(H_i{}^{(k)} + \mu_i{}^{(k)} I_n)^{-1}\nabla h_i(u_i^{(k)}) & \text{otherwise,} \end{cases}$$

where

$$H_i{}^{(k)} = (\tilde{G}_i'(u_i^{(k)}))^T \tilde{G}_i'(u_i^{(k)}), \nabla h_i(u_i^{(k)}) = (\tilde{G}_i'(u_i^{(k)}))^T \tilde{G}_i(u_i^{(k)})$$

and $\mu_i{}^{(k)} \geq 0$ is the unique solution of

$$\|s(\mu_i{}^{(k)})\| = \delta_i{}^{(k)}, \tag{7}$$

with $s(\mu)$ given by

$$s(\mu) = -(H_i{}^{(k)} + \mu I_n)^{-1}\nabla h_i(u_i^{(k)}).$$

"Hook step" for solving $\|s(\mu_i^{(k)})\| = \delta_i^{(k)}$

This algorithm computes (γ_n) such that $\lim_{n\to+\infty} \gamma_n = \mu_i^{(k)}$,

where

$$\gamma_{n+1} = \gamma_n - \frac{\|s(\gamma_n)\|}{\delta_i^{(k)}} \frac{\Phi(\gamma_n)}{\Phi'(\gamma_n)}$$

with $\Phi(\mu) = \|s(\mu)\| - \delta_i^{(k)}$.

"doglegstep" for solving $\|s(\mu_i{}^{(k)})\| = \delta_i{}^{(k)}$

It computes $s_i^{(k)} \approx s(\mu_i{}^{(k)})$ such that $\|s_i^{(k)}\| = \delta_i^{(k)}$.

- Let $u_{CP} = u_i^{(k)} - \lambda_* \nabla h_i(u_i^{(k)})$, where

$$\lambda_* = \frac{\left\|\nabla h_i(u_i^{(k)})\right\|^2}{\nabla\left(h_i(u_i^{(k)})\right)^T H_i^{(k)} \nabla\left(h_i(u_i^{(k)})\right)} = arg\min f(\lambda),$$

with

$$f(\lambda) = T(-\lambda \nabla h_i(u_i^{(k)}))$$
$$= h_i(u_i^{(k)}) - \lambda \|\nabla h_i(u_i^{(k)})\|^2 + \frac{1}{2}\lambda^2 \nabla\left(h_i(u_i^{(k)})\right)^T H_i^{(k)} \nabla\left(h_i(u_i^{(k)})\right).$$

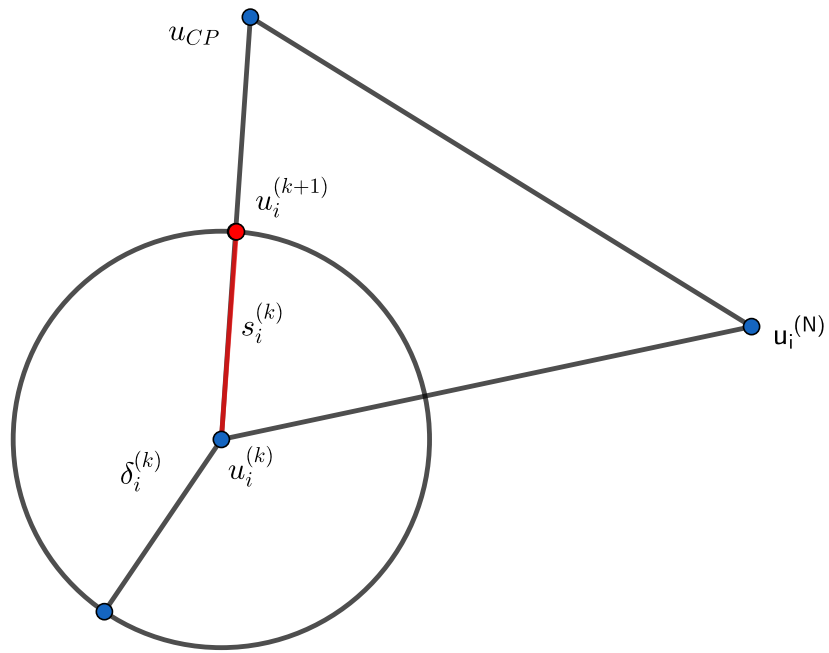
- Let $u_i^{(N)}$ be the Newton direction :

$$u_i^{(N)} = u_i^{(k)} - (\tilde{G}_i'(u_i^{(k)}))^{-1}\,\tilde{G}_i(u_i^{(k)}).$$

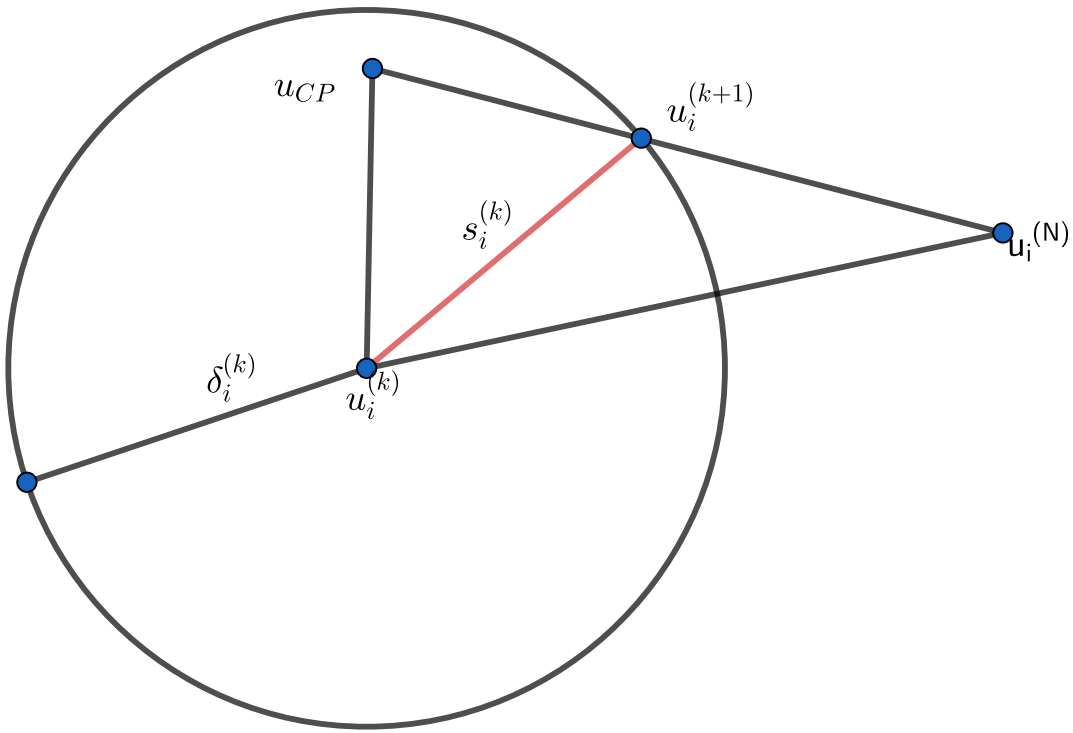
- Let $u_i^{(N)}$ be the Newton direction :

$$u_i^{(N)} = u_i^{(k)} - (\tilde{G}_i'(u_i^{(k)}))^{-1}\,\tilde{G}_i(u_i^{(k)}).$$

- If $\|u_{CP} - u_i^{(k)}\| \geq \delta_i^{(k)}$ then $u_i^{(k+1)} = u_i^{(k)} \underbrace{-\delta_i^{(k)} \frac{\nabla h_i(u_i^{(k)})}{\|\nabla h_i(u_i^{(k)})\|}}_{s_i^{(k)}}$.



- If $\|u_{CP} - u_i^{(k)}\| < \delta_i^{(k)}$ then $u_i^{(k+1)} = u_i^{(k)} + s_i^{(k)}$, with $u_i^{(k+1)} = [u_{CP}; u_i^{(N)}] \cap D(u_i^{(k)}, \delta_i^{(k)})$.



- **The computation of $s_i^{(k)}$ is expensive**. Then, we use T_0 where

$$T_0(d) := T(V_i d) = \frac{1}{2}\|\tilde{G}_i(u_i^{(k)}) + \tilde{G}_i'(u_i^{(k)})V_i d\|^2, \ d \in \mathbb{R}^r$$

where $V_i = vect\{f(t_{i-k}, y_{i-k}), \ k = 0, \ldots, r-1\}$. [1], $r \ll n$.

- Then $u_i^{(k+1)} = u_i^{(k)} + V_i d_i^{(k)}$, where $d_i^{(k)}$ is the solution of

$$\min_{\|d\| \leq \delta_i^{(k)}} T_0(d). \tag{8}$$

[1] M. Al Sayed Ali, M. Sadkane : *Acceleration of implicit schemes for large systems of nonlinear ODEs*, Electronic Transactions on Numerical Analysis, Vol. $35(2009), 104 - 117$

Mixed Precision Accelerated Implicit Numerical Scheme (MPAINS)

(MP1) Set $i = 0$. For $i \leq N$, compute y_{i+1} as follows :

A) Computing an initial guess $y_i^{(0)}$, by :

Mixed Precision Accelerated Implicit Numerical Scheme (MPAINS)

(MP1) Set $i = 0$. For $i \leq N$, compute y_{i+1} as follows :

A) Computing an initial guess $y_i^{(0)}$, by :

A1) Line-search (LS) by computing $p_i^{(k)}$ using **MP GMRES.**

Mixed Precision Accelerated Implicit Numerical Scheme (MPAINS)

(MP1) Set $i = 0$. For $i \leq N$, compute y_{i+1} as follows :

A) Computing an initial guess $y_i^{(0)}$, by :

A1) Line-search (LS) by computing $p_i^{(k)}$ using **MP GMRES.**

A2) Trust-region (TR) by computing $s_i^{(k)}$ using **HOOKSTEP** or **DOGLEGSTEP**.

B) Inexact Newton (IN):

(IN1) Let $m = 0$ and repeat until $\|\tilde{G}_i(y_i^{(m)})\| \leq \varepsilon$:

(IN2) Compute an approximation $\tilde{z}_i^{(m)} \approx z_i^{(m)}$ by **MP GMRES.**

(IN3) $y_i^{(m+1)} = y_i^{(m)} + \tilde{z}_i^{(m)}$.

(MP2) Set $y_{i+1} = y_i^{(m)}$.

(MP3) $i = i + 1$.

OUTPUT: $y_M = y_N \approx y(T)$.

Notations :

- (DPAINS, T_D, y_D) for Double precision algorithm.
- (MPAINS, T_M, y_M) for Mixed precision algorithm.
- (SPAINS, T_S, y_S) for Single precision algorithm.

Computational environment

- Compiler : Fortran
- Parallel computing : MPI, Ethernet, 25 Gbps
- Cluster : Gros on Grid5000, Intel Xeon Gold 5220, 2.20GHz
- # processors (hosts) : 144(8)
- The runtime of each method, is calculated as the average of 4 technical replicates of the simulations.

Benchmark 1

Let's consider the following benchmark model, a large nonlinear densely coupled ODEs[2], $(S_i)_{i=1,\dots,N_h}$, where (S_i) is given by:

$$\frac{dy_1^{(i)}}{dt} = \frac{1}{\tau}\left(\frac{\nu_{1b}(y_7^{(i)}+\psi_{1i})}{k_{1b}(1+\left(\frac{y_3^{(i)}}{k_{1i}}\right)^{p_0})+y_7^{(i)}+\psi_{1i}} - k_{1d}y_1^{(i)}\right)$$

$$\frac{dy_2^{(i)}}{dt} = \frac{1}{\tau}\left(k_{2b}(y_1^{(i)})^q - k_{2d}y_2^{(i)} - k_{2t}y_2^{(i)} + k_{3t}y_3^{(i)}\right)$$

$$\frac{dy_3^{(i)}}{dt} = \frac{1}{\tau}\left(k_{2t}y_2^{(i)} - k_{3t}y_3^{(i)} - k_{3d}y_3^{(i)}\right)$$

[2] R. El cheikh, S. Bernard, N. El Khatib, *A multiscale modelling approach for the regulation of the cell cycle by the circadian clock*, Journal of Theoretical Biology 426 (2017) 117-125

$$
\begin{aligned}
\frac{dy_4^{(i)}}{dt} &= \frac{1}{\tau}\left(\frac{\nu_{4b}(y_3^{r_0})^{(i)}}{k_{4b}^{r_0}+(y_3^{r_0})^{(i)}} - k_{4d}y_4^{(i)}\right)\\
\frac{dy_5^{(i)}}{dt} &= \frac{1}{\tau}\left(k_{5b}y_4^{(i)} - k_{5d}y_5^{(i)} - k_{5t}y_5^{(i)} + k_{6t}y_6^{(i)}\right)\\
\frac{dy_6^{(i)}}{dt} &= \frac{1}{\tau}\left(k_{5t}y_5^{(i)} - k_{6t}y_6^{(i)} - k_{6d}y_6^{(i)} + k_{7a}y_7^{(i)} - k_{6a}y_6^{(i)}\right)\\
\frac{dy_7^{(i)}}{dt} &= \frac{1}{\tau}\left(k_{6a}y_6^{(i)} - k_{7a}y_7^{(i)} - k_{7d}y_7^{(i)}\right)\\
\frac{dy_8^{(i)}}{dt} &= \lambda\left(\frac{(k_{impf} + k_{0mpf}\exp(-\eta\psi_2))k_{1mpf}^{n_0}}{(k_{1mpf}^{n_0} + (y_8^{(i)})^{n_0} + sy_{10}^{n_0})(1-y_8^{(i)})} - d_{wee1}y_9^{(i)}y_8^{(i)}\right)
\end{aligned}
$$

$$\frac{dy_9^{(i)}}{dt} = \lambda\left(\frac{k_{actw}}{k_{actw}+dw_1}(cw + C(y_7 - bbmal_0) + bbmal_0) + \ldots \right.$$
$$\left. \left(\frac{k_{actw}}{k_{actw}+dw_1} - 1\right)\frac{kin_{actw}(y_8^{(i)})^{n_0} y_9^{(i)}}{k^{n_0}{}_{1wee_1} + (y_8^{(i)})^{n_0}} - dw_2 y_9^{(i)}\right)$$

$$\frac{dy_{10}^{(i)}}{dt} = \lambda\left(kk_{act}(y_8^{(i)} - y_{10}^{(i)})\right)$$

where $t \in [0, 120]$ and ψ_{1i}, ψ_{2i}, for $i = 1, \ldots, N_h$, are given by

$$\psi_{1i} = \frac{ks}{N_h}\sum_{j=1}^{N_h} \arctan(y_2^{(j)} - y_2^{(i)}) + \frac{\pi}{2}ks$$

$$\psi_{2i} = N_h = 10^4 \Rightarrow n = 10^5.$$

- This ODE is solved by several implicit numerical methods with different timesteps.

- This ODE is solved by several implicit numerical methods with different timesteps.
- The (relative) error is expressed as $\left\| \frac{y_D - y_I}{y_D} \right\|_\infty$, $I \in \{S, M\}$.

- This ODE is solved by several implicit numerical methods with different timesteps.
- The (relative) error is expressed as $\left\| \frac{y_D - y_I}{y_D} \right\|_\infty$, $I \in \{S, M\}$.
- Runtime speedup : $\frac{T_D}{T_I}$, $I \in \{S, M\}$.
- *nb. LS iterations* $= 5$,
- *nb. GMRES iterations* $= 10$,
- *nb. Newton iterations* $= 5$,
- The number of vectors r in V_i is 2.

Implicit numerical methods

The s-step BDFs of orders $1, 2, 3, 4, 5, 6$ are given by :

BDF1: $y_{i+1} = y_i + hf(t_{i+1}, y_{i+1})$

BDF2: $y_{i+2} = \frac{4}{3}y_{i+1} - \frac{1}{3}y_i + \frac{2}{3}hf(t_{i+2}, y_{i+2})$

BDF3: $y_{i+3} = \frac{18}{11}y_{i+2} - \frac{9}{11}y_{i+1} + \frac{2}{11}y_i + \frac{6}{11}hf(t_{i+3}, y_{i+3})$

BDF4: $y_{i+4} = \frac{48}{25}y_{i+3} - \frac{36}{25}y_{i+2} + \frac{16}{25}y_{i+1} - \frac{3}{25}y_i + \frac{12}{25}hf(t_{i+4}, y_{i+4})$

BDF5: $$y_{i+5} = \frac{300}{137}y_{i+4} - \frac{300}{137}y_{i+3} + \frac{200}{137}y_{i+2} - \frac{75}{137}y_{i+1} + \frac{12}{137}y_i + \frac{60}{137}hf(t_{i+5}, y_{i+5})$$

BDF6: $$y_{i+6} = \frac{360}{147}y_{i+5} - \frac{450}{147}y_{i+4} + \frac{400}{147}y_{i+3} - \frac{225}{147}y_{i+2} + \frac{72}{147}y_{i+1} - \frac{10}{147}y_i + \frac{60}{147}hf(t_{i+6}, y_{i+6}).$$

and the Crank-Nicolson (AM2) method of order 2

AM2: $$y_{i+1} = y_i + \frac{1}{2}h\left(f(t_{i+1}, y_{i+1}) + f(t_i, y_i)\right).$$

Implicit Runge Kutta methods

The IRK methods are given by :

$$\begin{aligned} Y_{ik} - y_i - h\sum_{j=1}^{s} a_{kj} f(t_i + c_j h, Y_{ij}) &= 0, \qquad \text{for } k = 1, \ldots s, \\ y_{i+1} &= y_i + h\sum_{k=1}^{s} d_k f(t_i + c_k h, Y_{ik}), \end{aligned}$$

where a_{kj}, c_j, d_k are the IRK coefficients.

Let $Y_i = [Y_{i1}^T, \ldots, Y_{is}^T]^T$. Then, we get

$$G_i(Y_i) = 0,$$

where $G_i : \mathbb{R}^{n.s} \longrightarrow \mathbb{R}^{n.s}$ is given by

$$G_i(X) = X - 1_s \otimes y_i - h(A_0 \otimes I_n)F_i(X), \qquad X \in \mathbb{R}^{n.s}$$

with

- $1_s = (1, \ldots, 1)^T \in \mathbb{R}^s$,
- $A_0 = (a_{ij}) \in \mathbb{R}^{s \times s}$,
- $X = [X_1^T, \ldots, X_s^T]^T$,
- $F_i(X) = \left[f(t_i + c_1 h, X_1)^T, \ldots, f(t_i + c_s h, X_s)^T\right]^T$.

Numerical Experiments for Benchmark 1

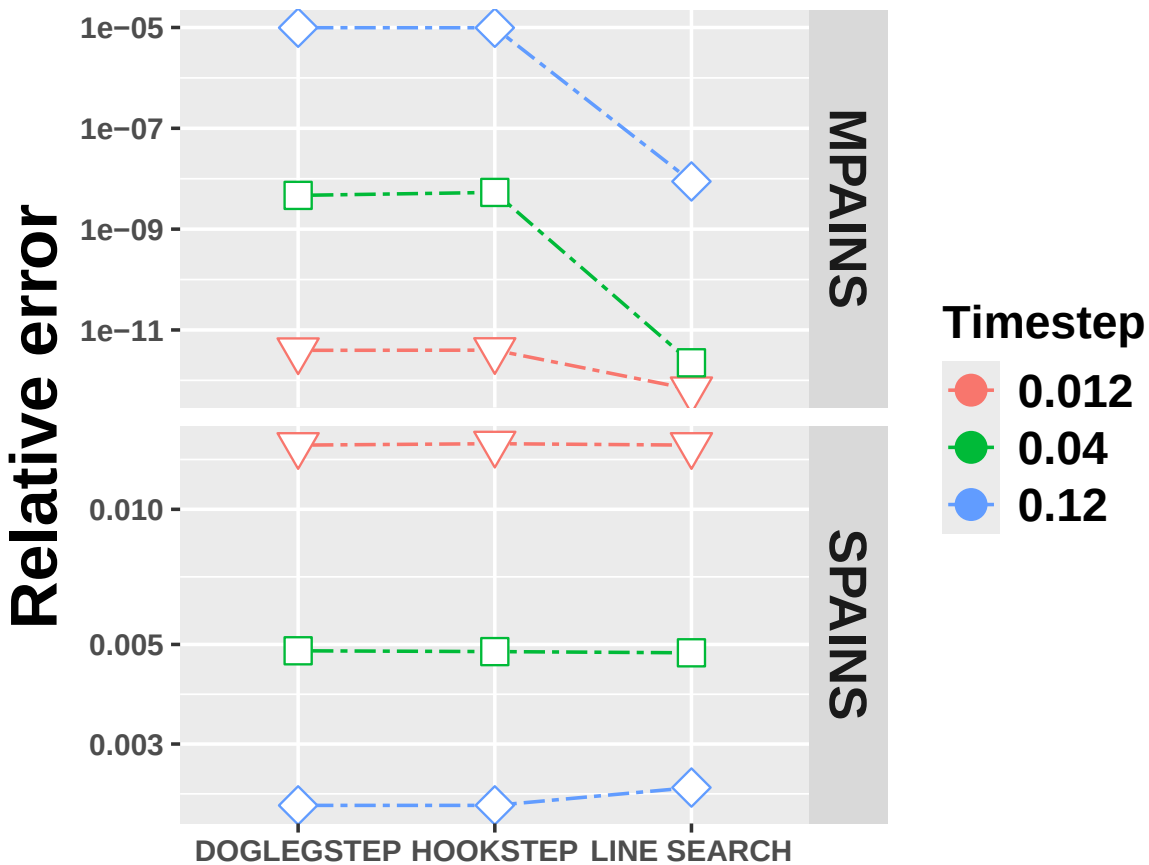


Figure: **Benchmark 1**. Numerical method **BDF1**.

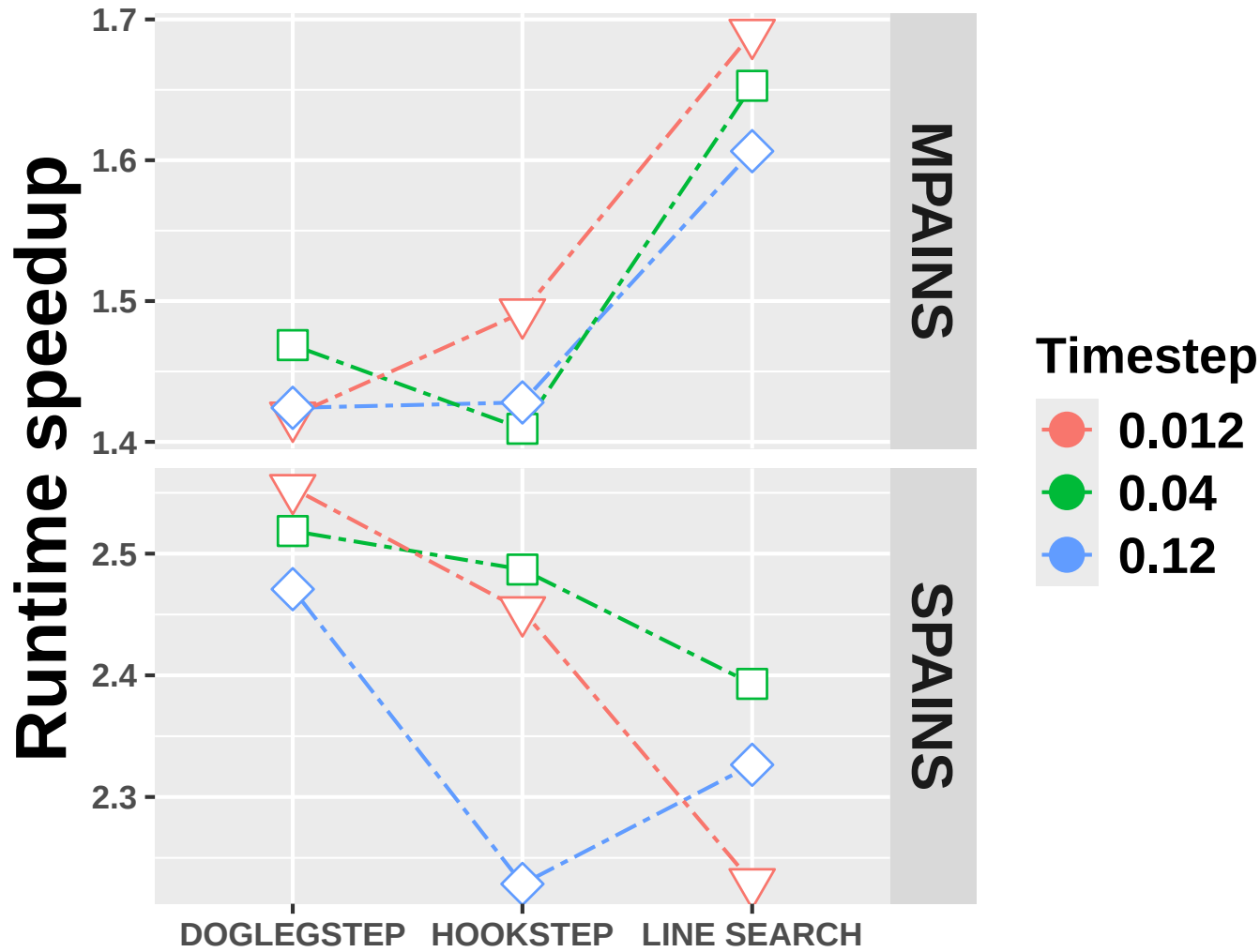


Figure: **Benchmark** 1. Numerical method **BDF1**.

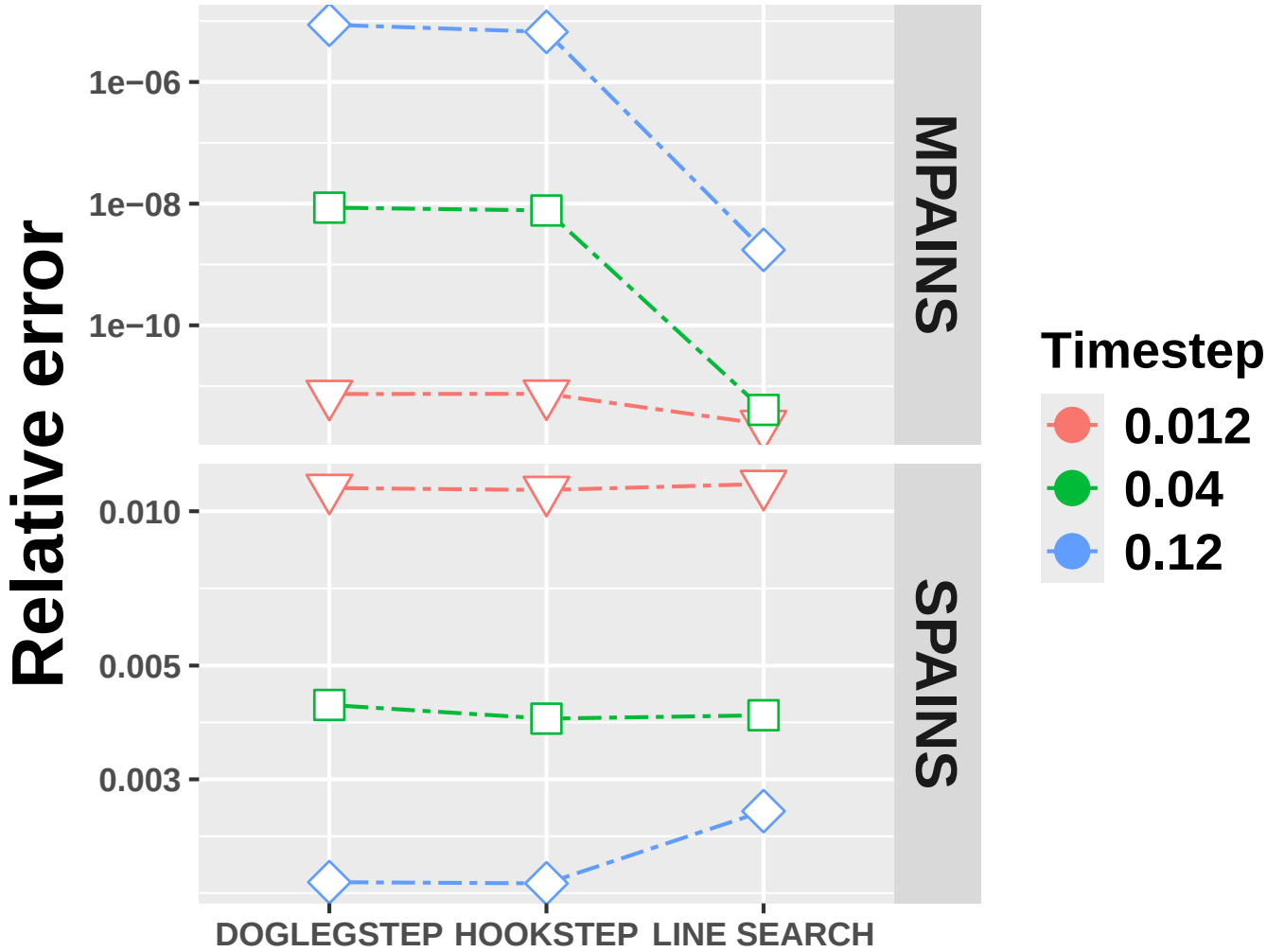


Figure: **Benchmark** 1. Numerical method **BDF2**.

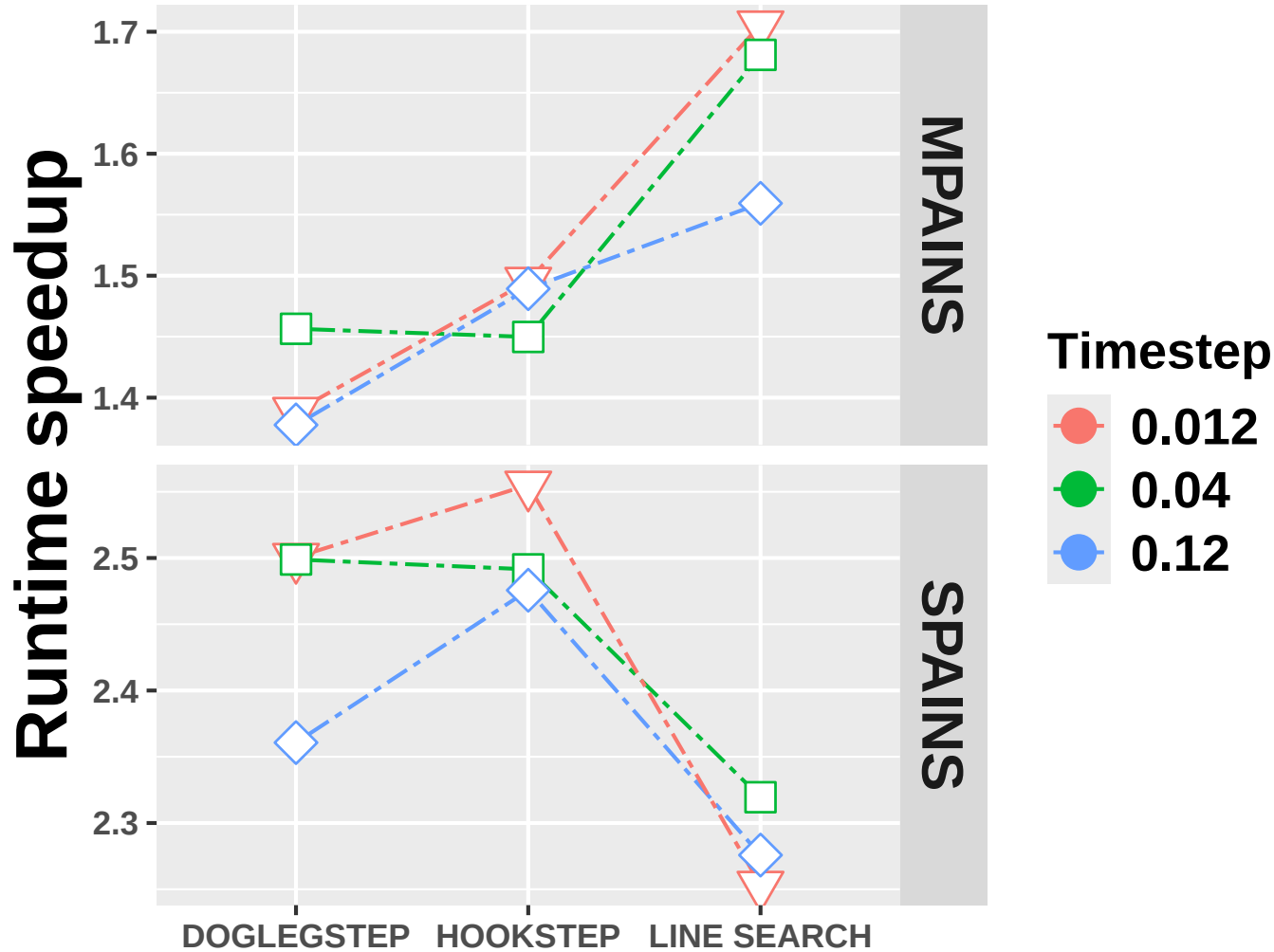


Figure: **Benchmark** 1. Numerical method **BDF2**.

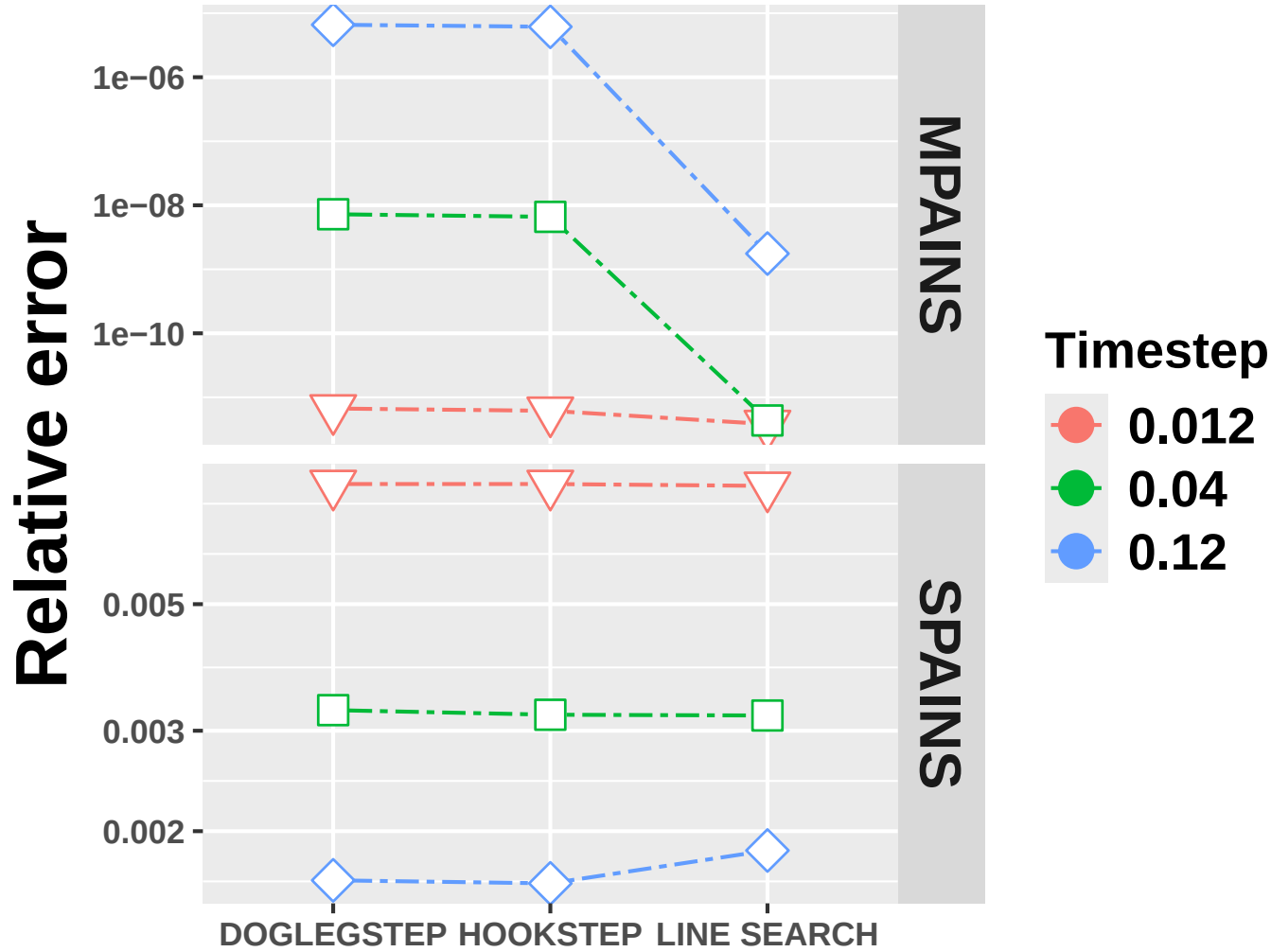


Figure: **Benchmark** 1. Numerical method **BDF3**.

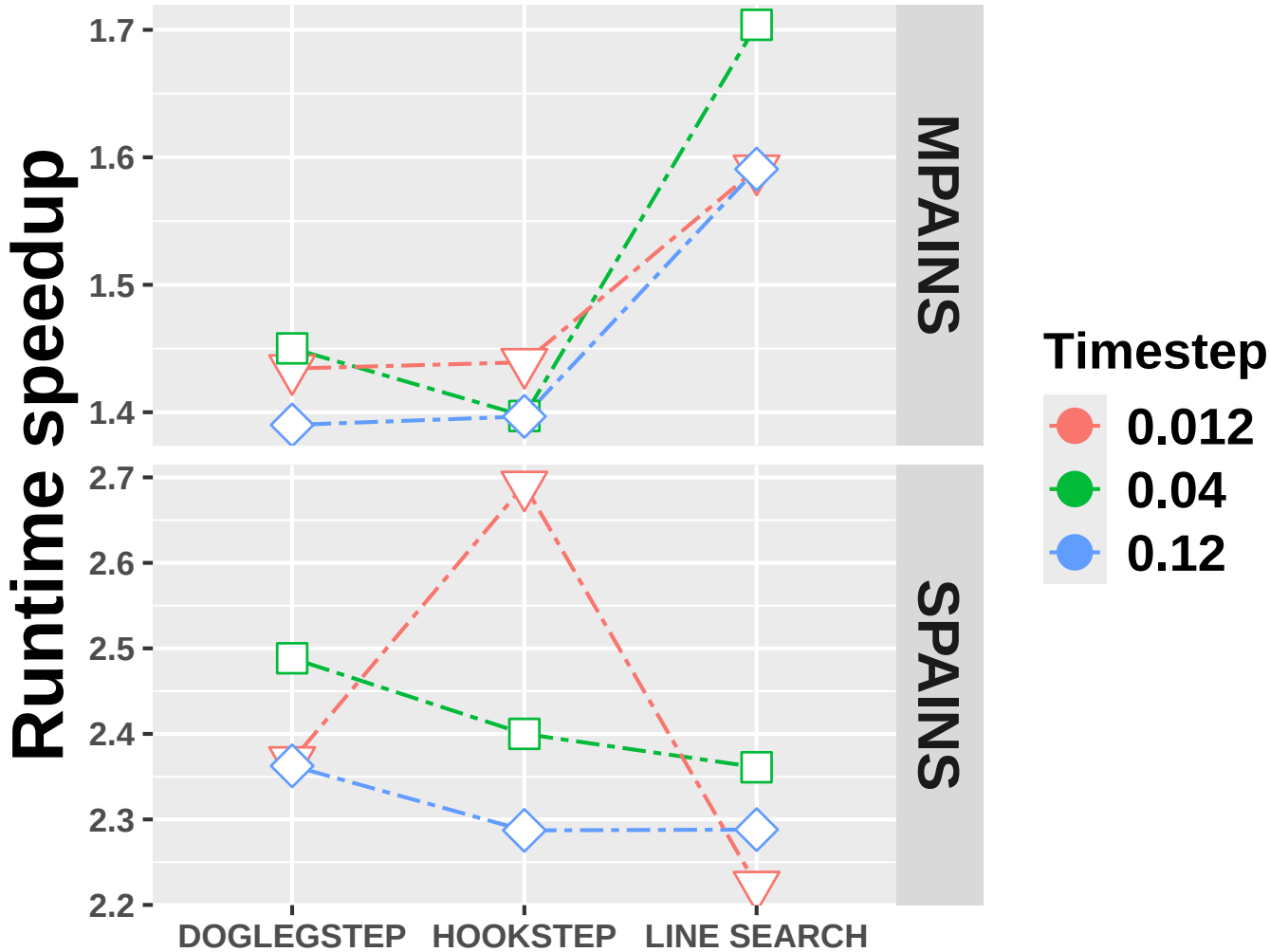


Figure: **Benchmark** 1. Numerical method **BDF3**.

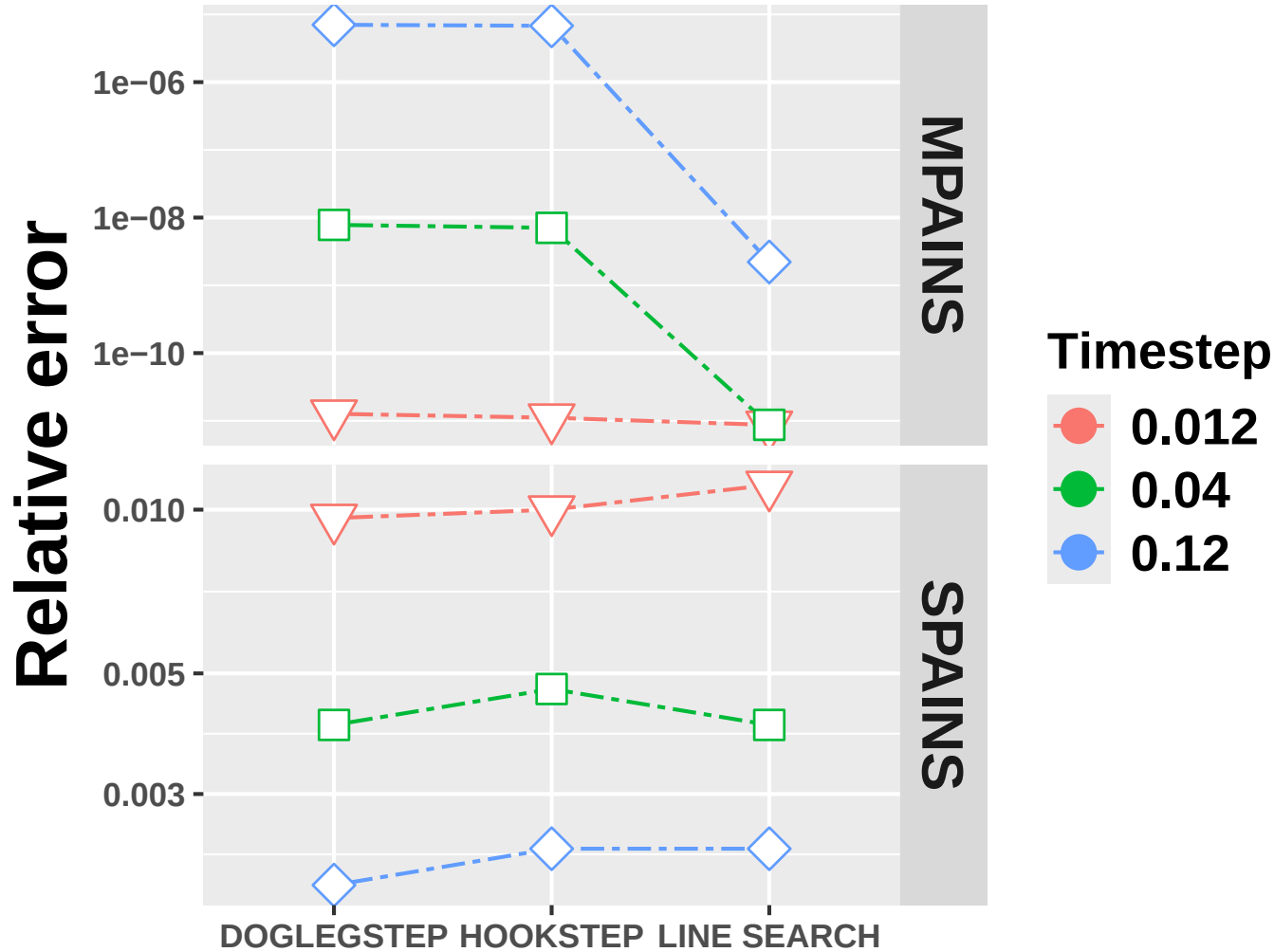


Figure: **Benchmark** 1. Numerical method **BDF4**.

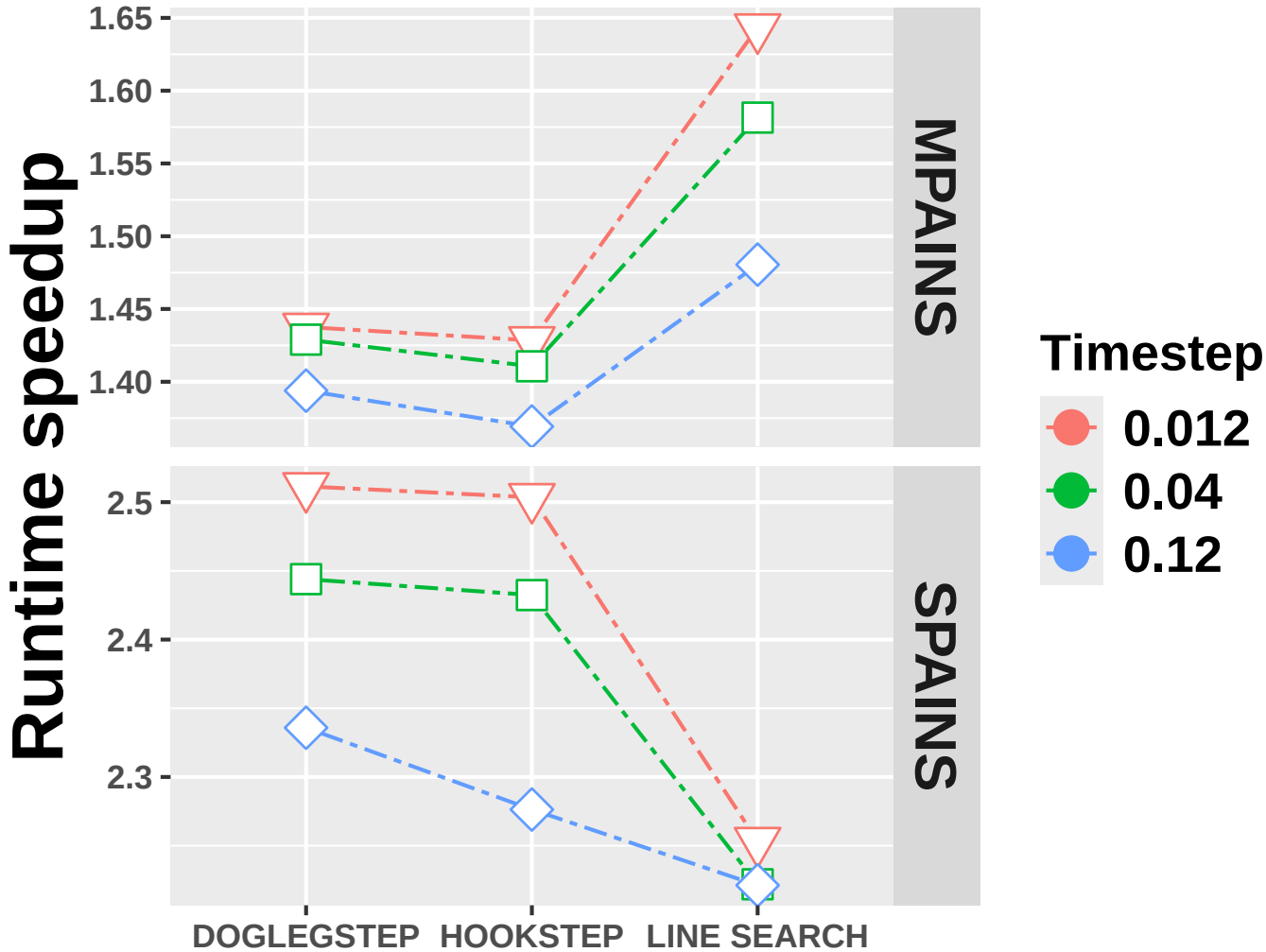


Figure: **Benchmark** 1. Numerical method **BDF4**.

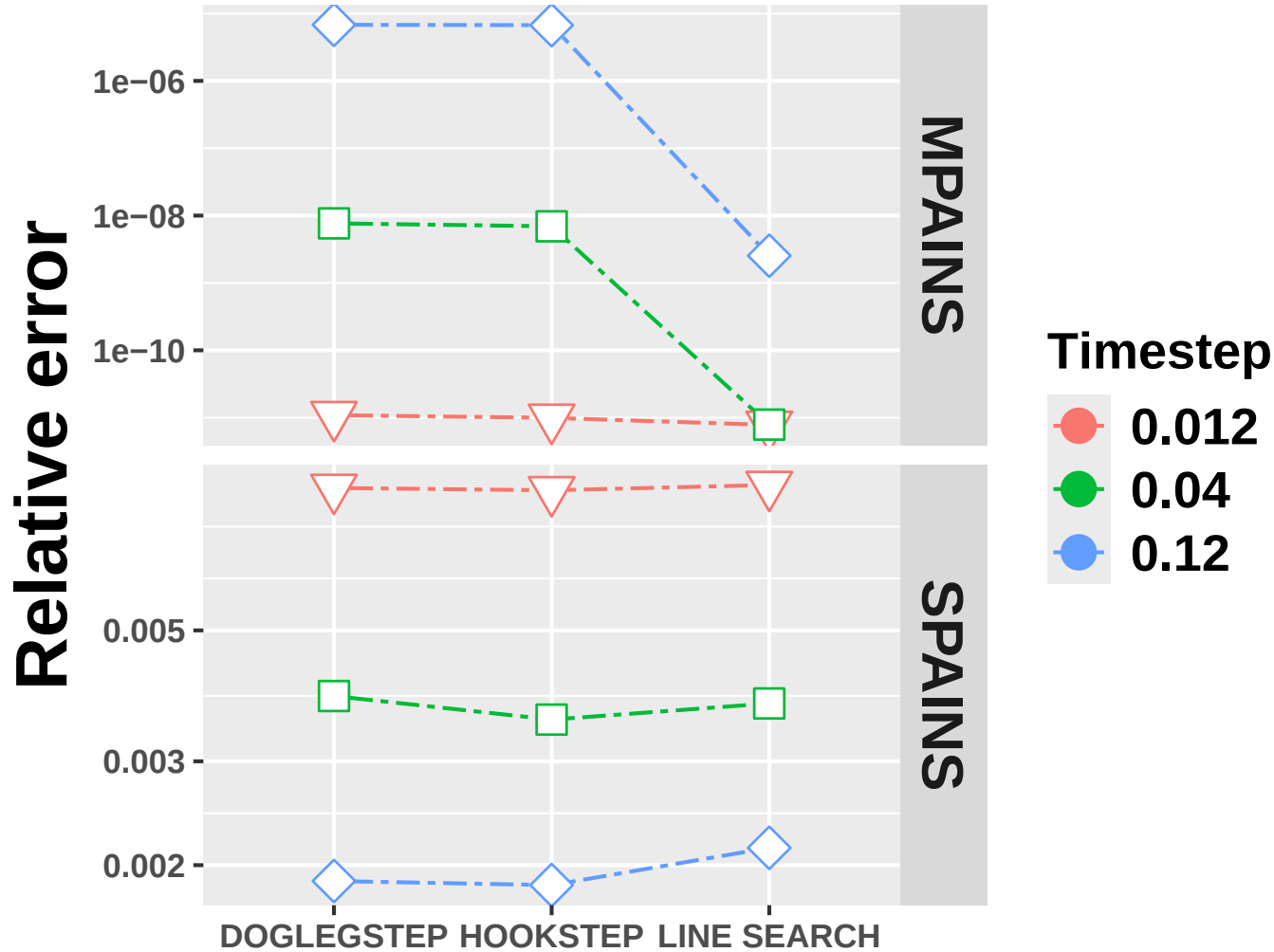


Figure: **Benchmark** 1. Numerical method **BDF5**.

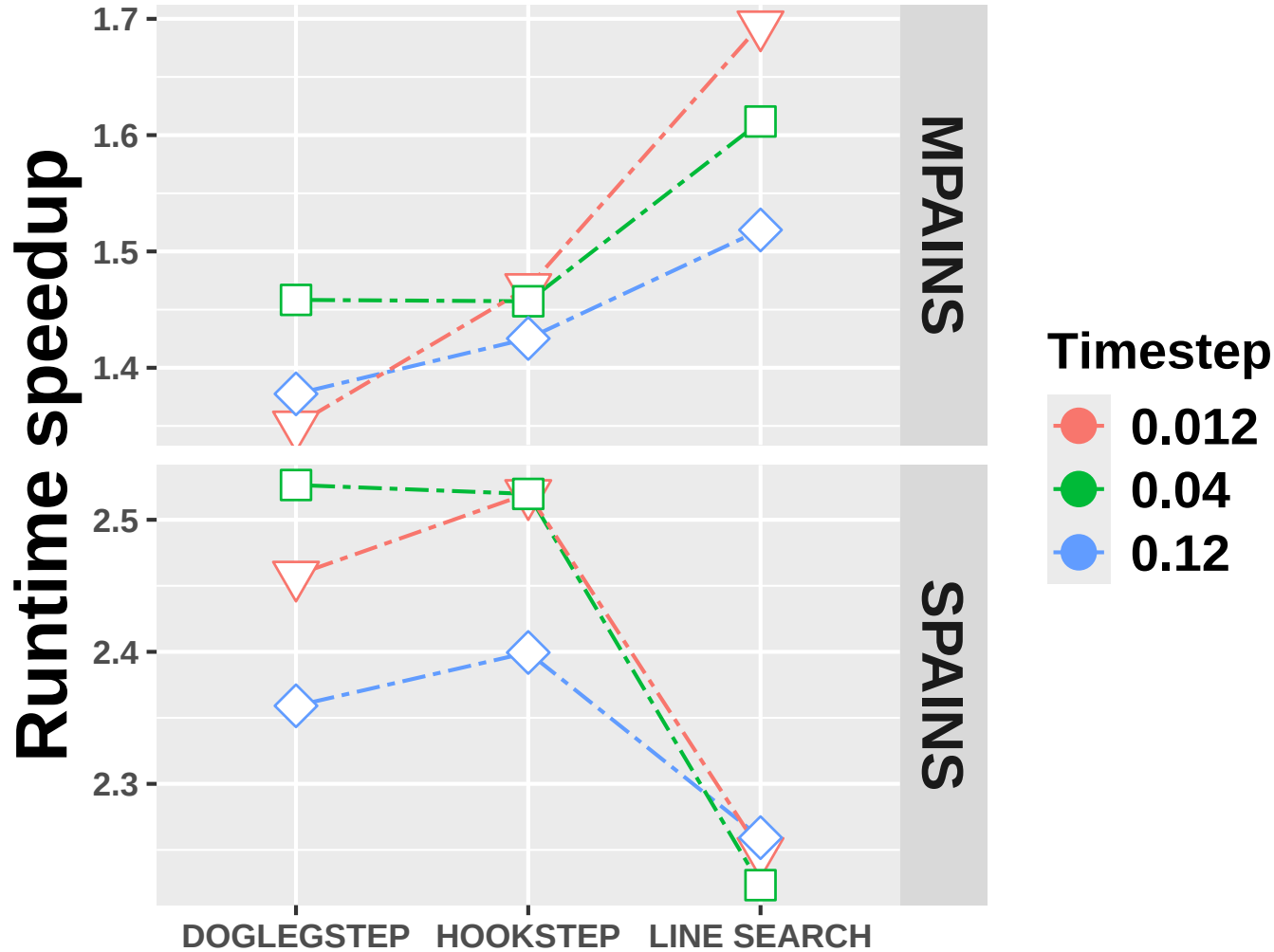


Figure: **Benchmark** 1. Numerical method **BDF5**.

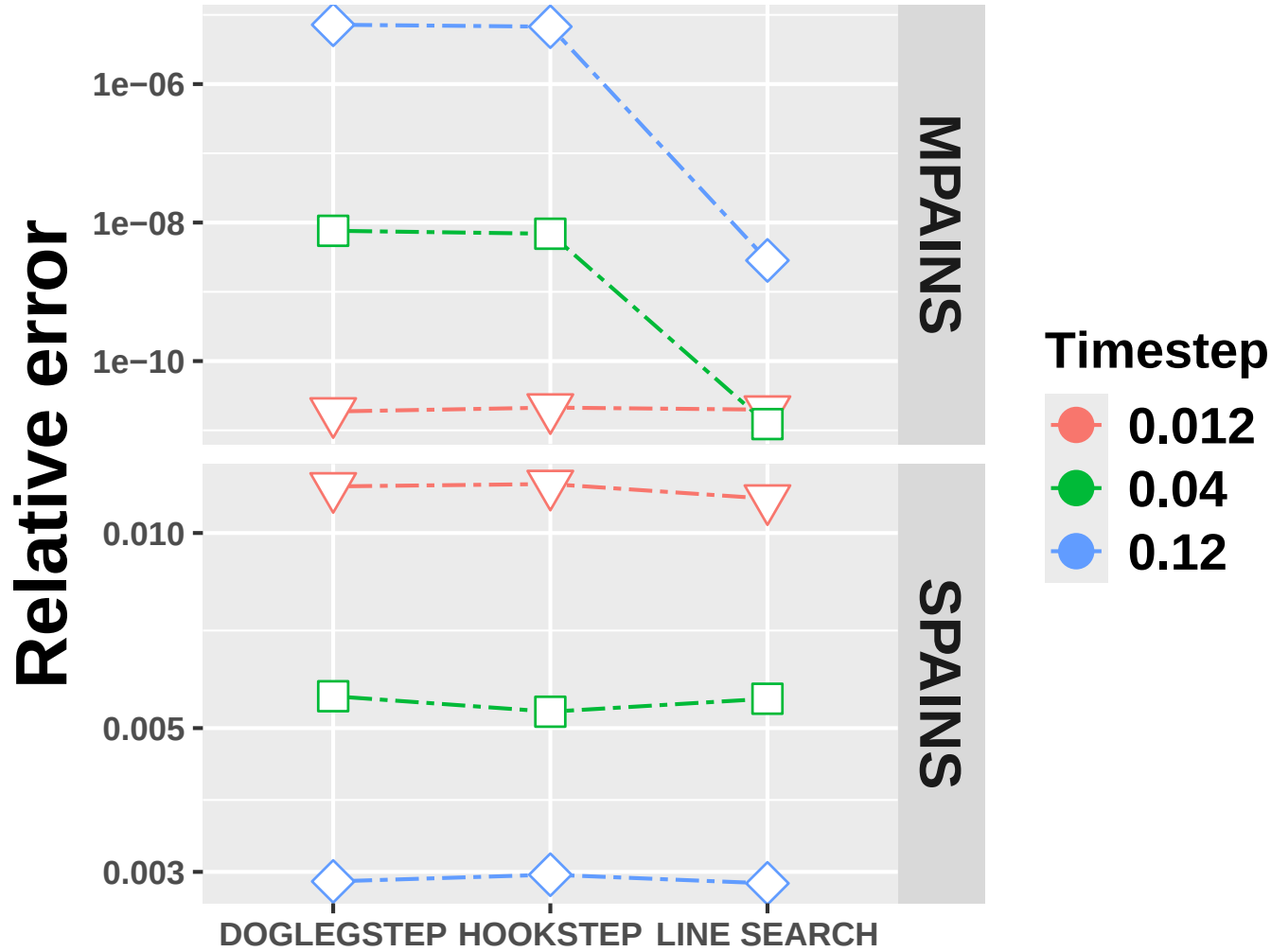


Figure: **Benchmark** 1. Numerical method **BDF6**.

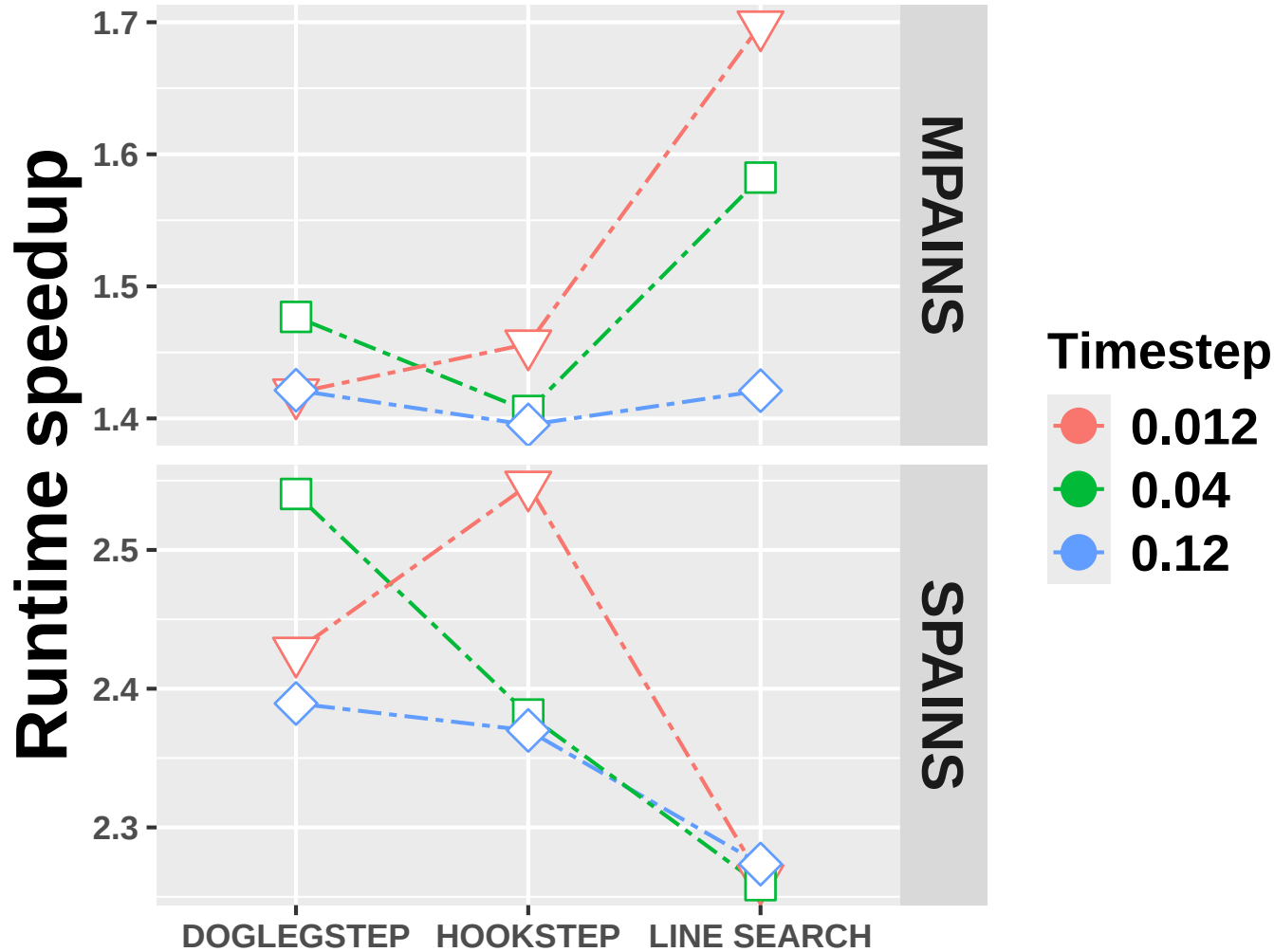


Figure: **Benchmark** 1. Numerical method **BDF6**.

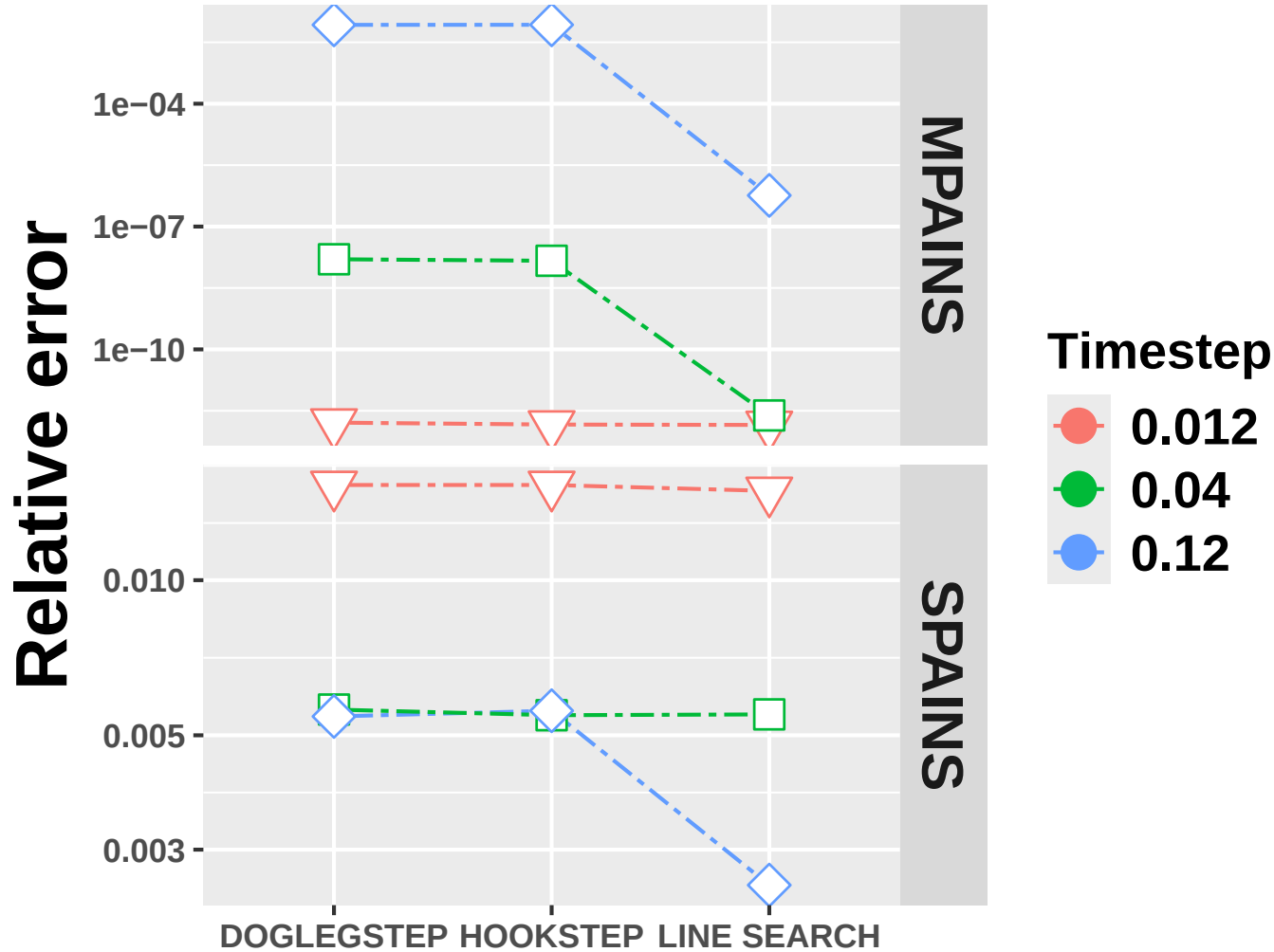


Figure: **Benchmark** 1. Numerical method **AM2**.

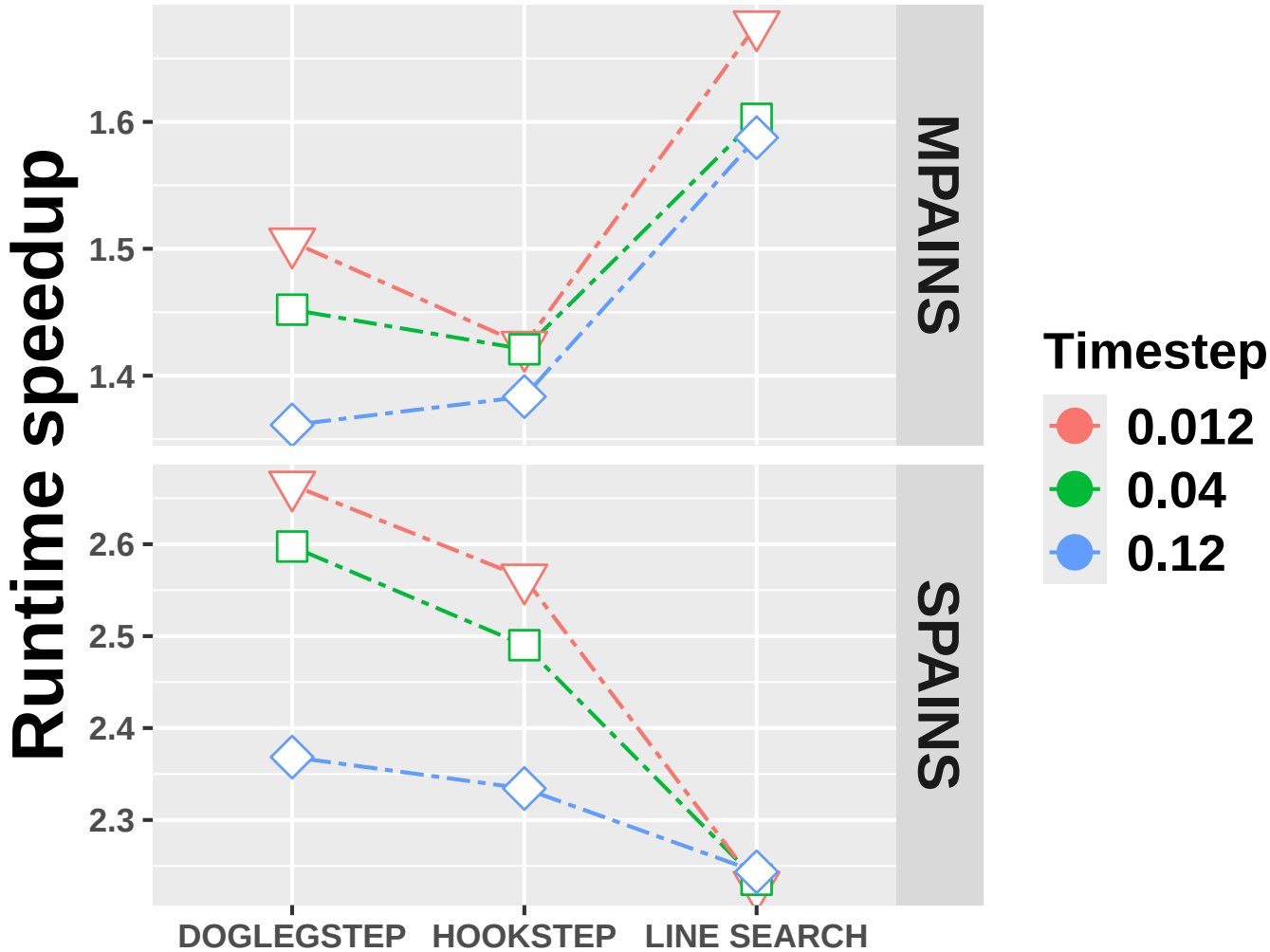


Figure: **Benchmark** 1. Numerical method **AM2**.

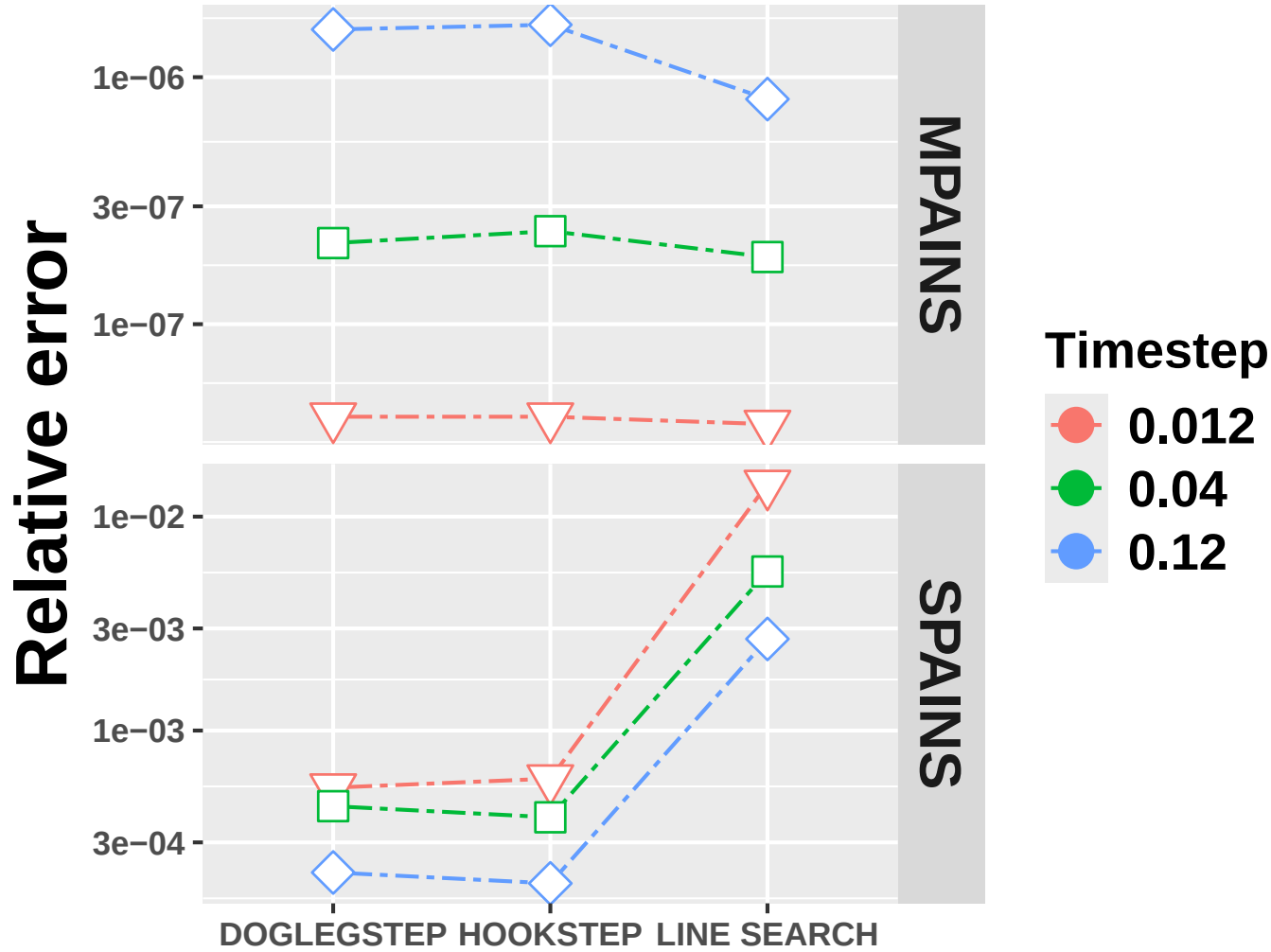


Figure: **Benchmark** 1. Numerical method **IRK4**.

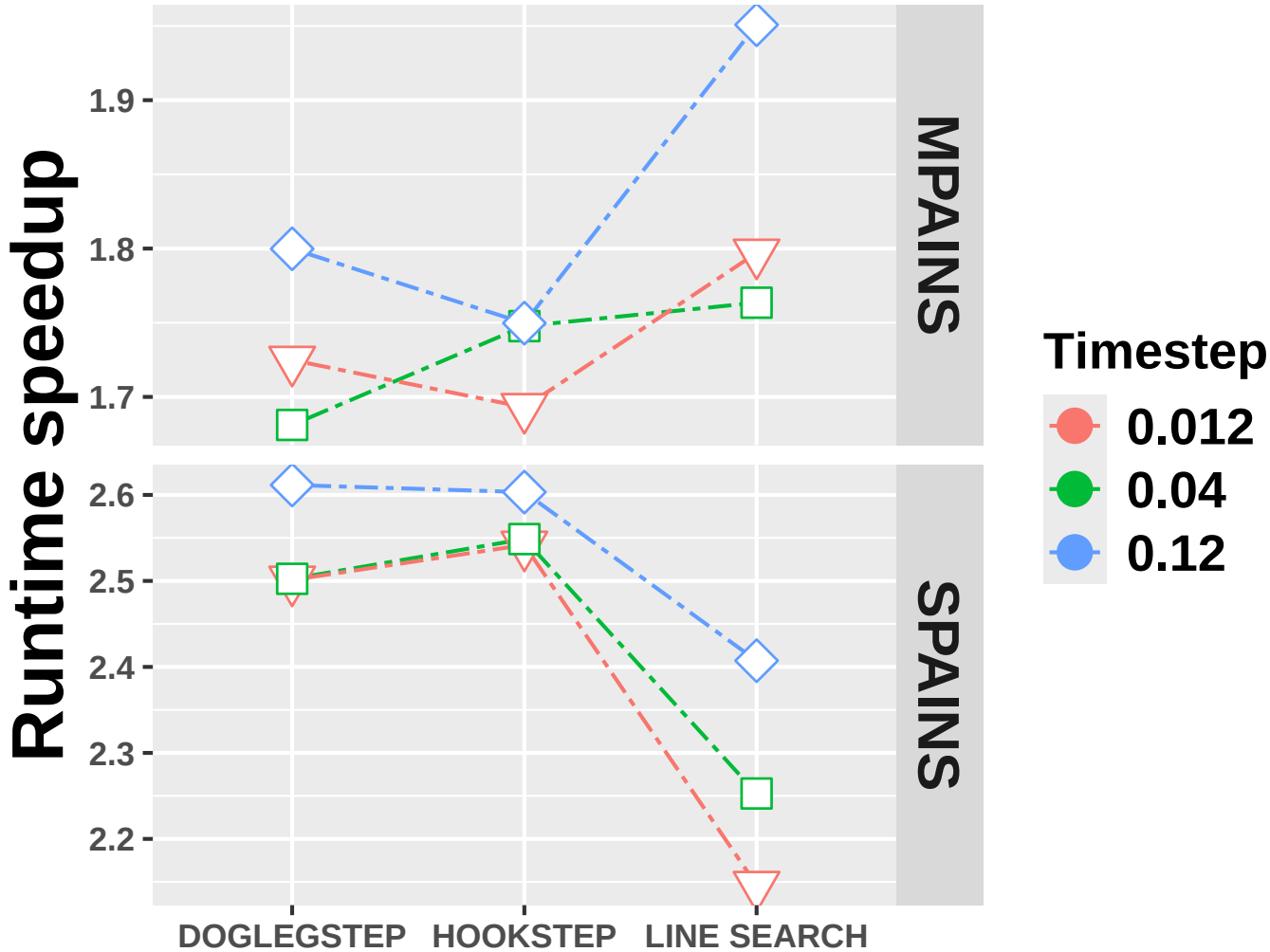


Figure: **Benchmark** 1. Numerical method **IRK4**.

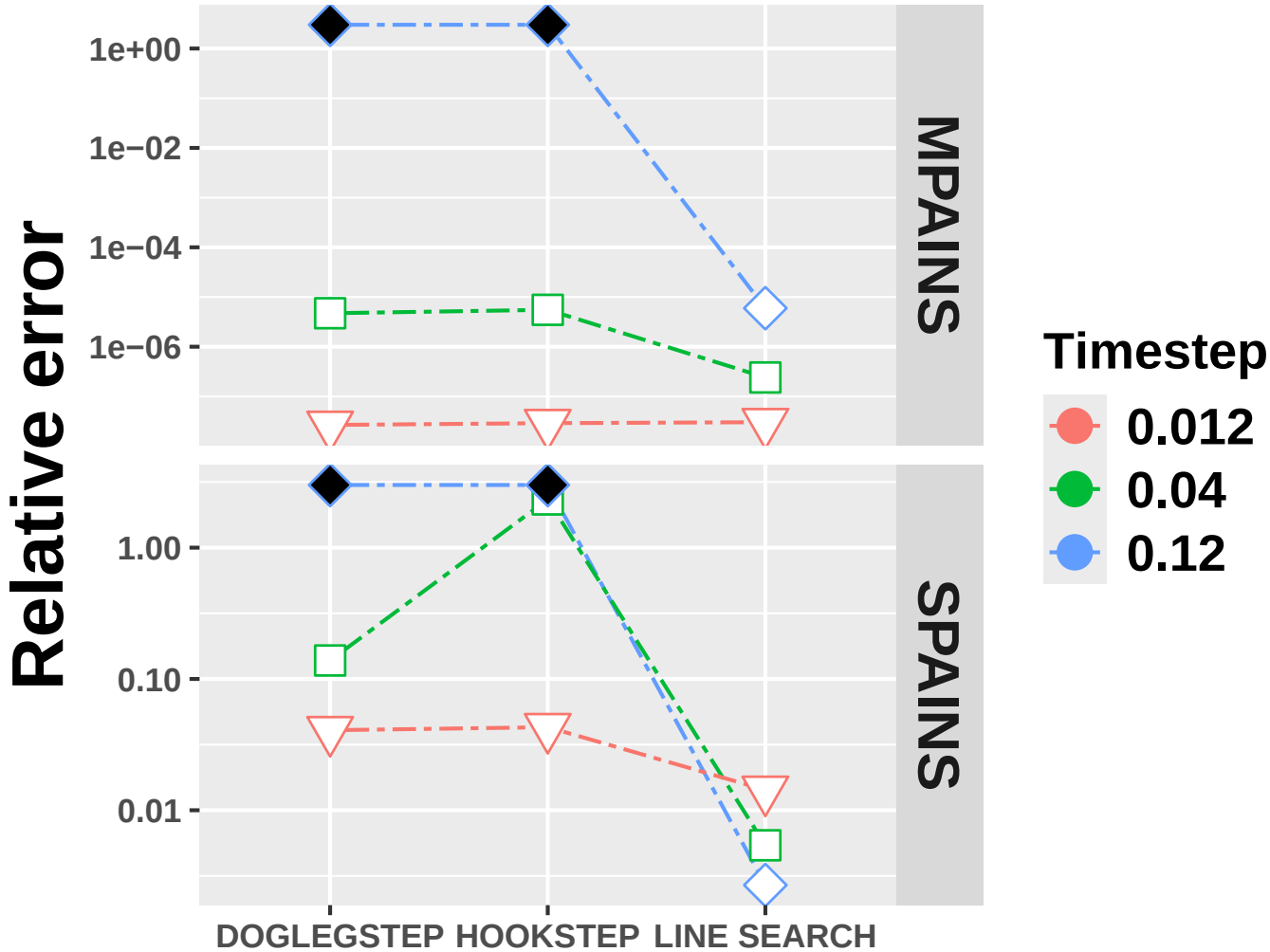


Figure: **Benchmark** 1. Numerical method **IRK6**.

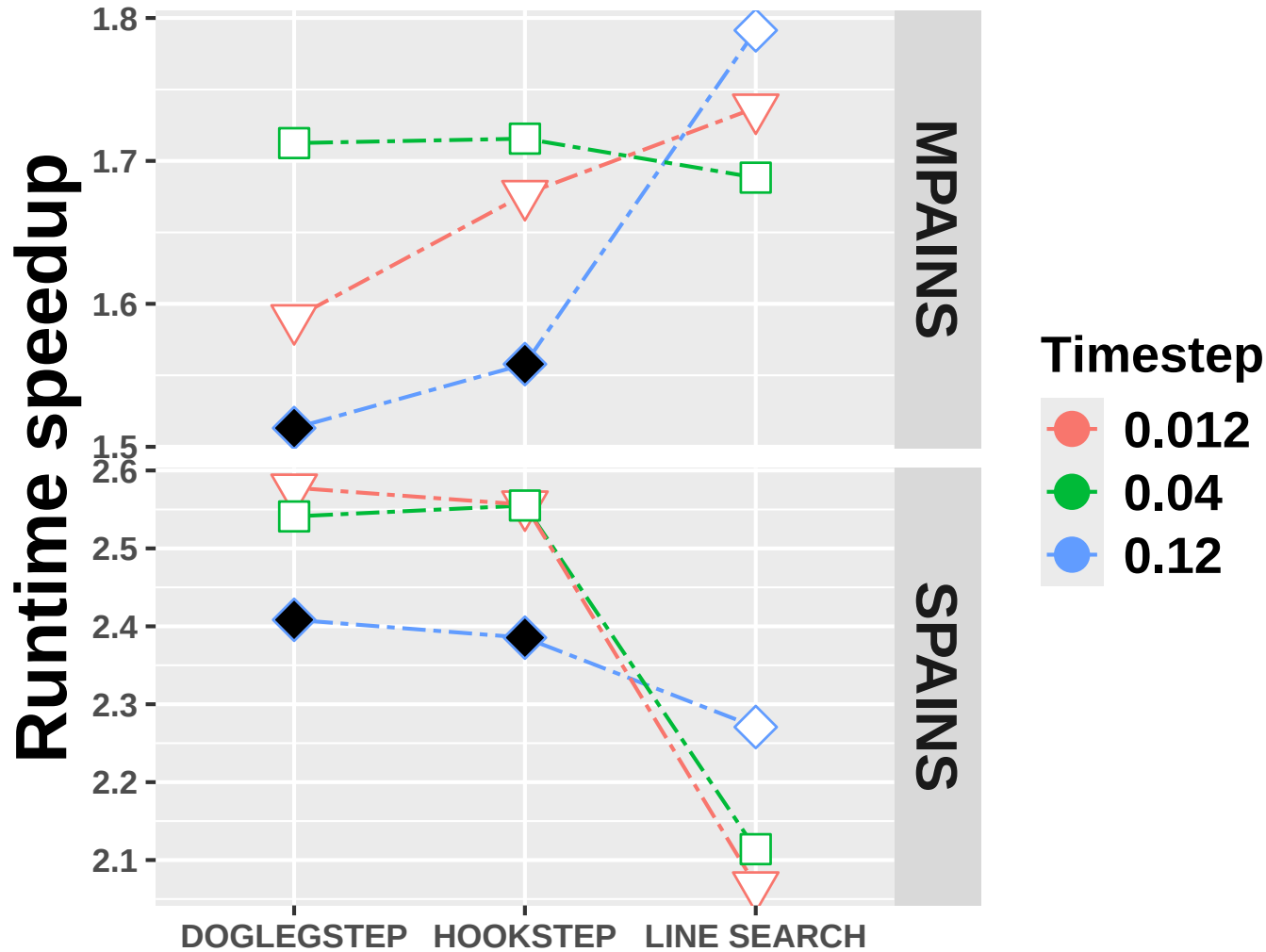


Figure: **Benchmark** 1. Numerical method **IRK6**.

Benchmark 2

Let's consider the Neural Field Equation (NFE) [3]

$$\frac{\partial V}{\partial t}(x,t) = I(x,t) - V(x,t) + \int_{-1}^{1} K(|x-z|) S(V(z,t)) dz \qquad t \in [0,1]$$

where $K(x) = e^{-x^2}, S(x) = I(x,t) = \tanh(x)$.

[3] S.L. Amari, *Dynamics of pattern formation in lateral-inhibition type neural fields*, Biol. Cybernet. 27(2)(1977)77 – 87.

- Discretizing the integral, with $n = 10^6$ points, we get an ODE of the form $\dot{y} = f(t, y)$ of size n.

Numerical Experiments for Benchmark 2

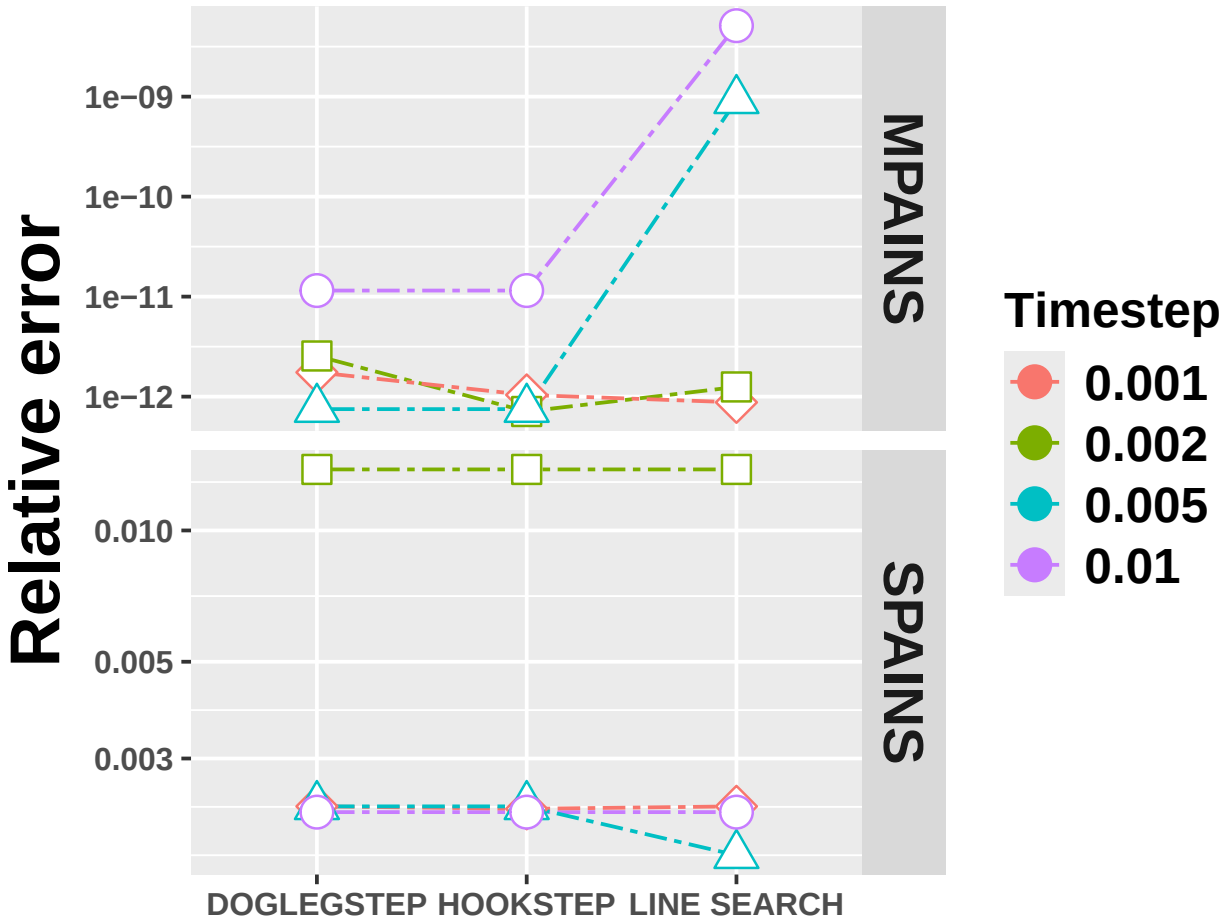


Figure: **Benchmark** 2. Numerical method **BDF1**.

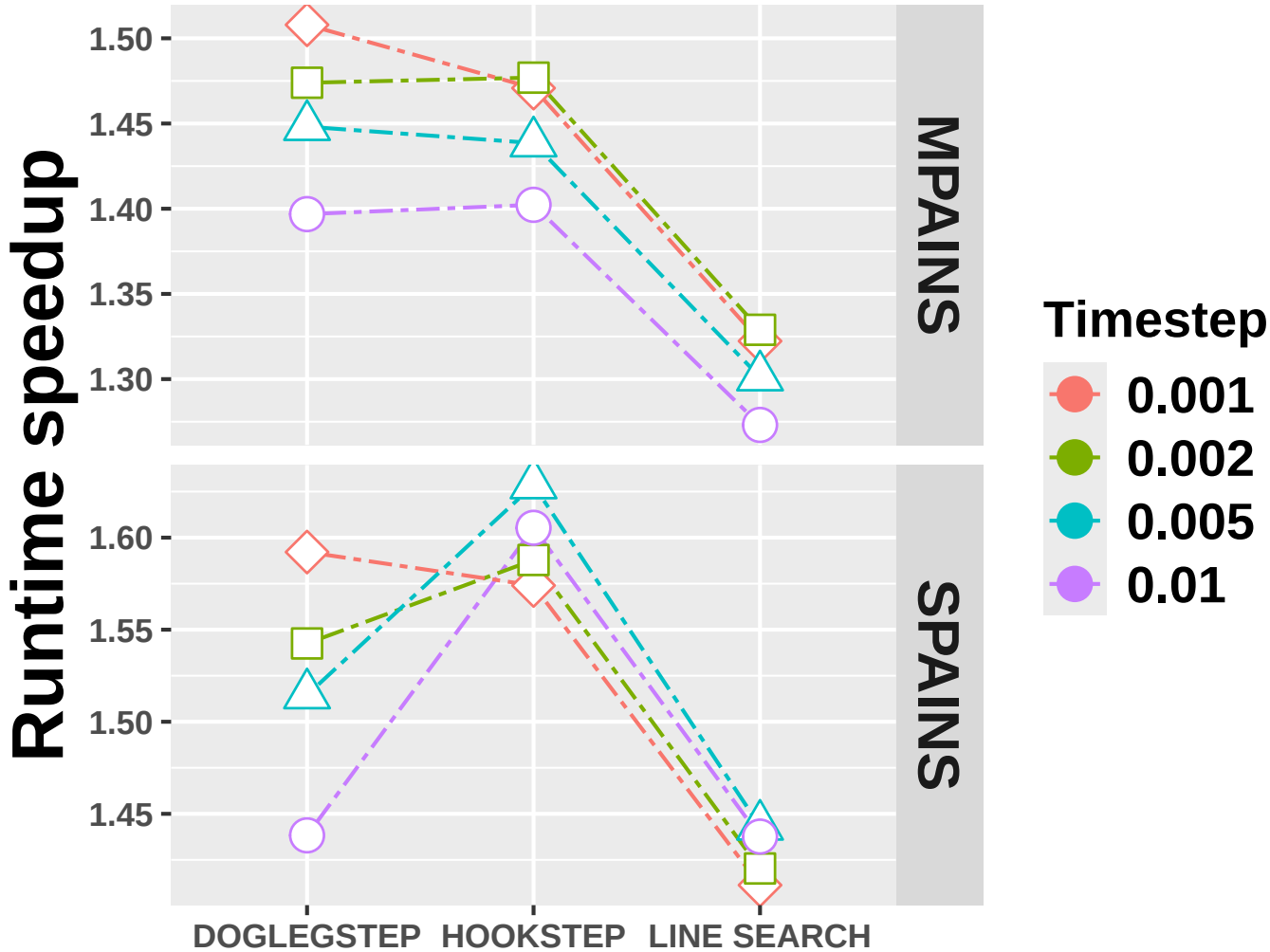


Figure: **Benchmark** 2. Numerical method **BDF1**.

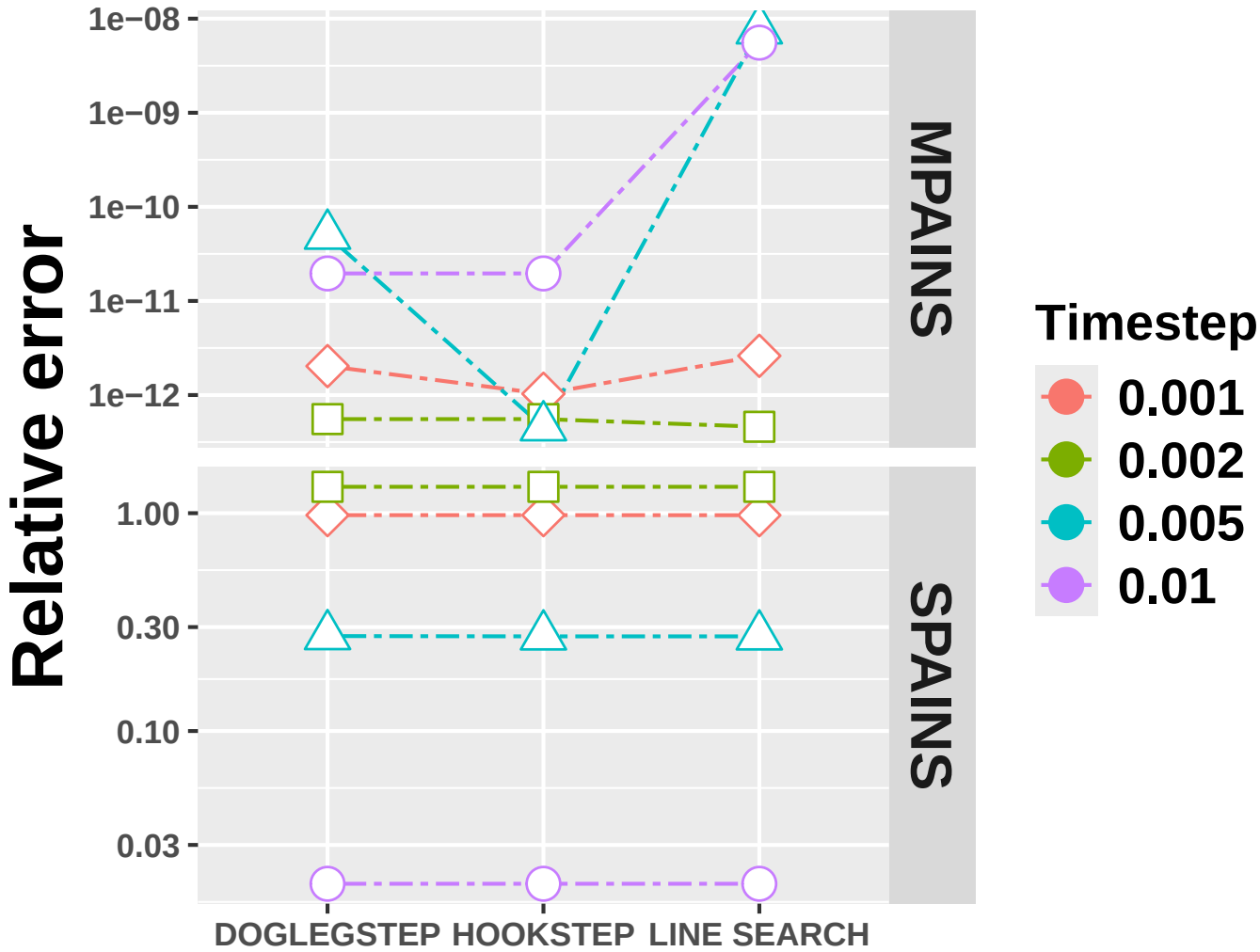


Figure: **Benchmark** 2. Numerical method **BDF2**.

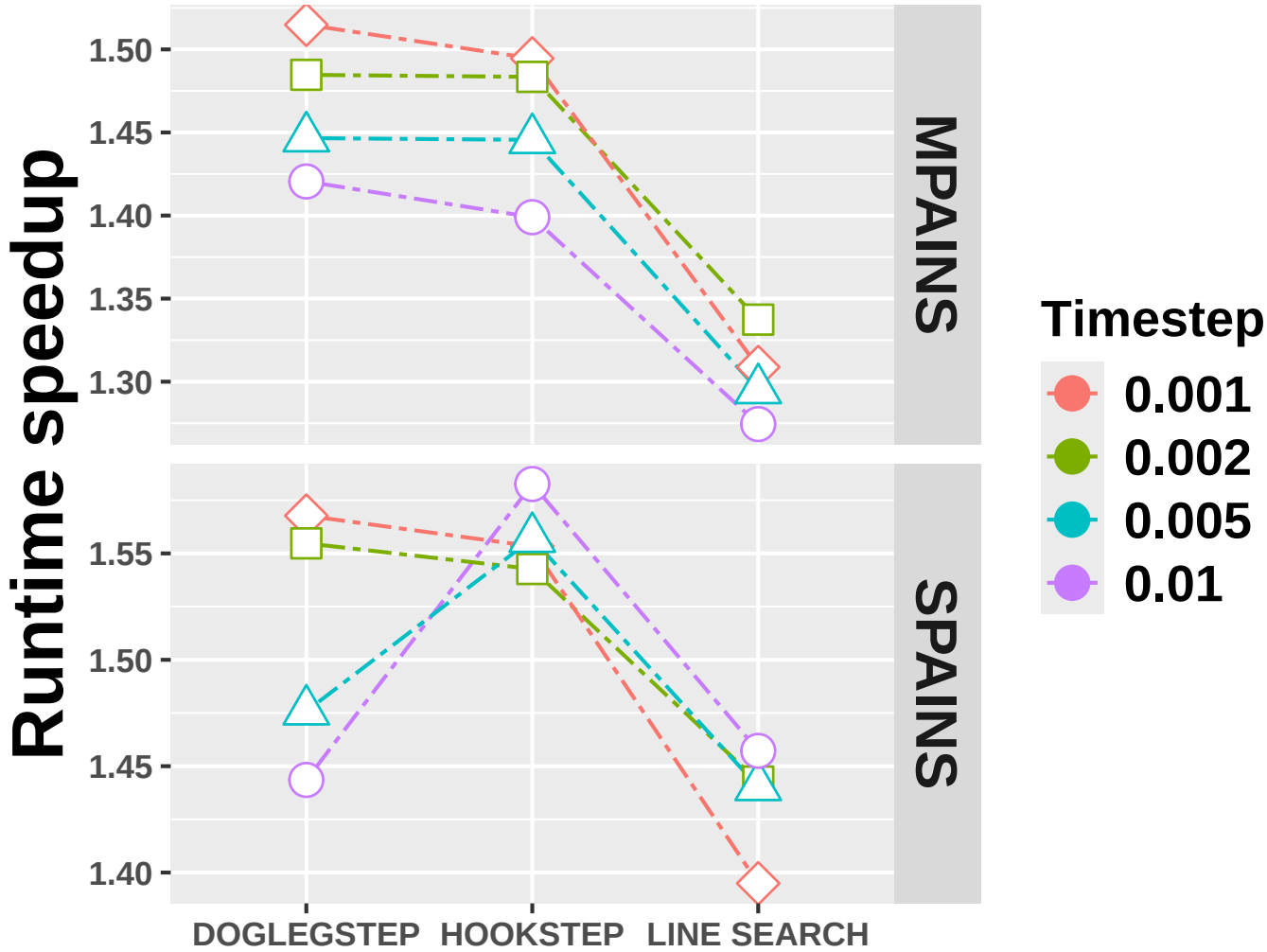


Figure: **Benchmark** 2. Numerical method **BDF2**.

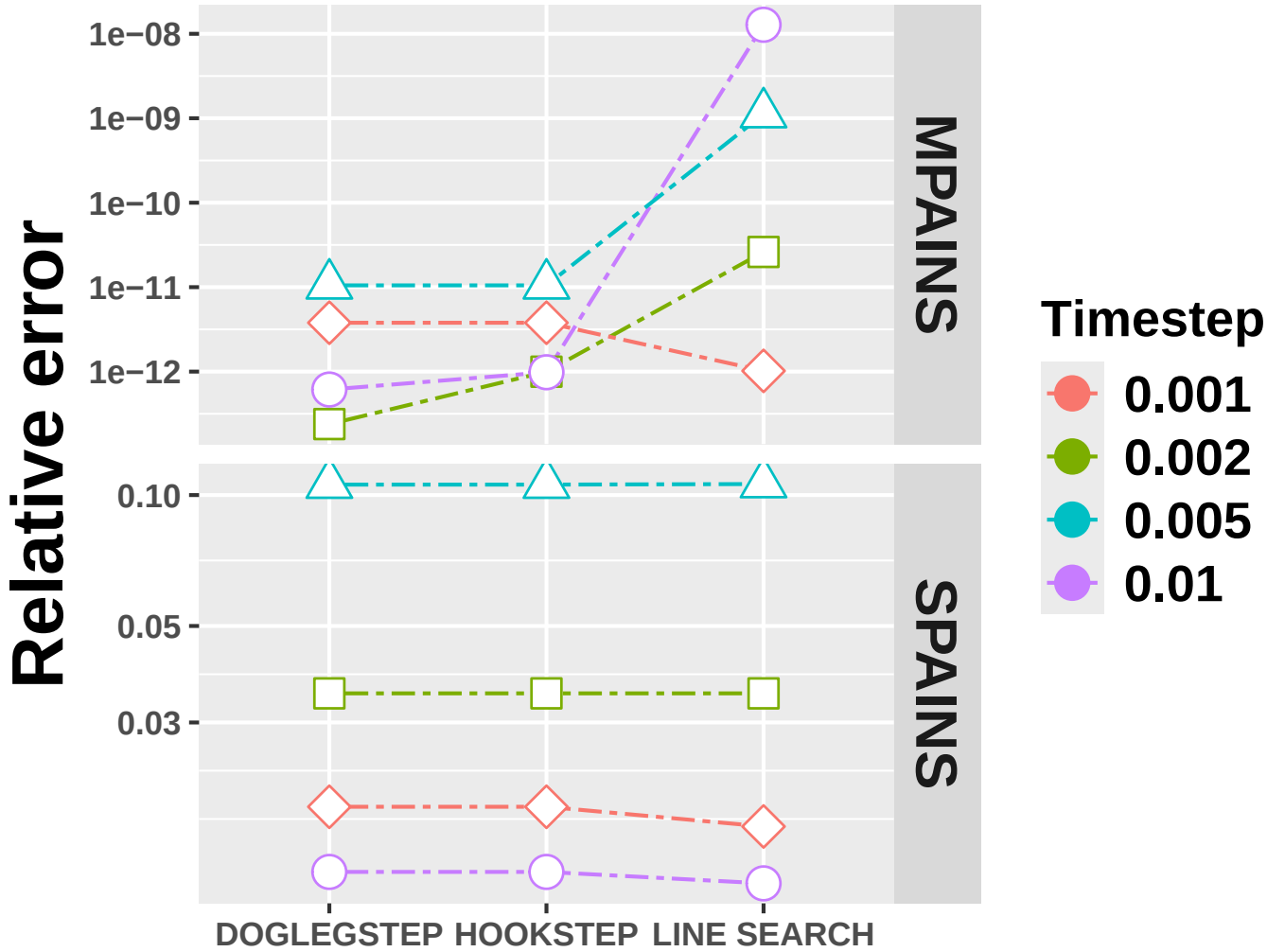


Figure: **Benchmark** 2. Numerical method **BDF3**.

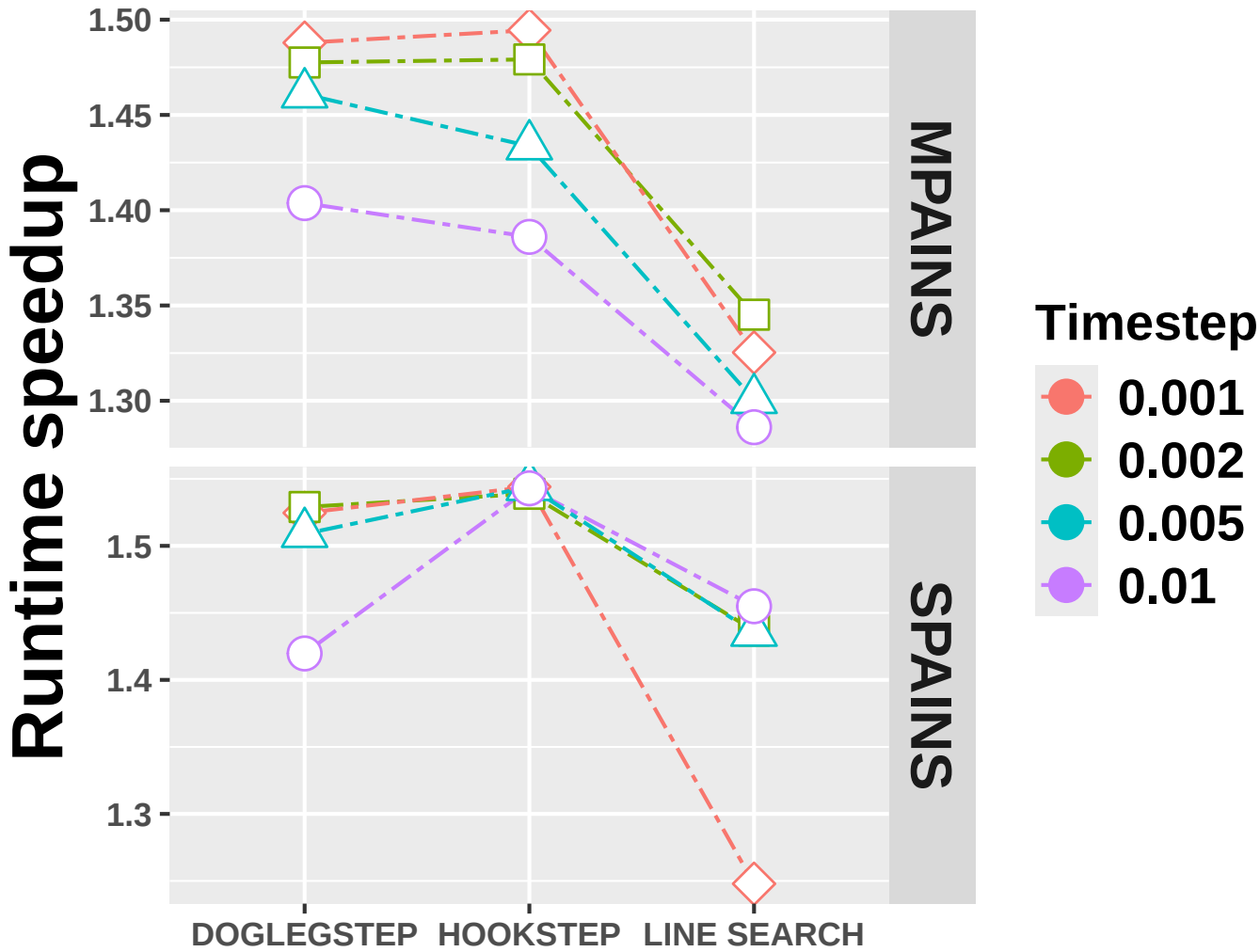


Figure: **Benchmark** 2. Numerical method **BDF3**.

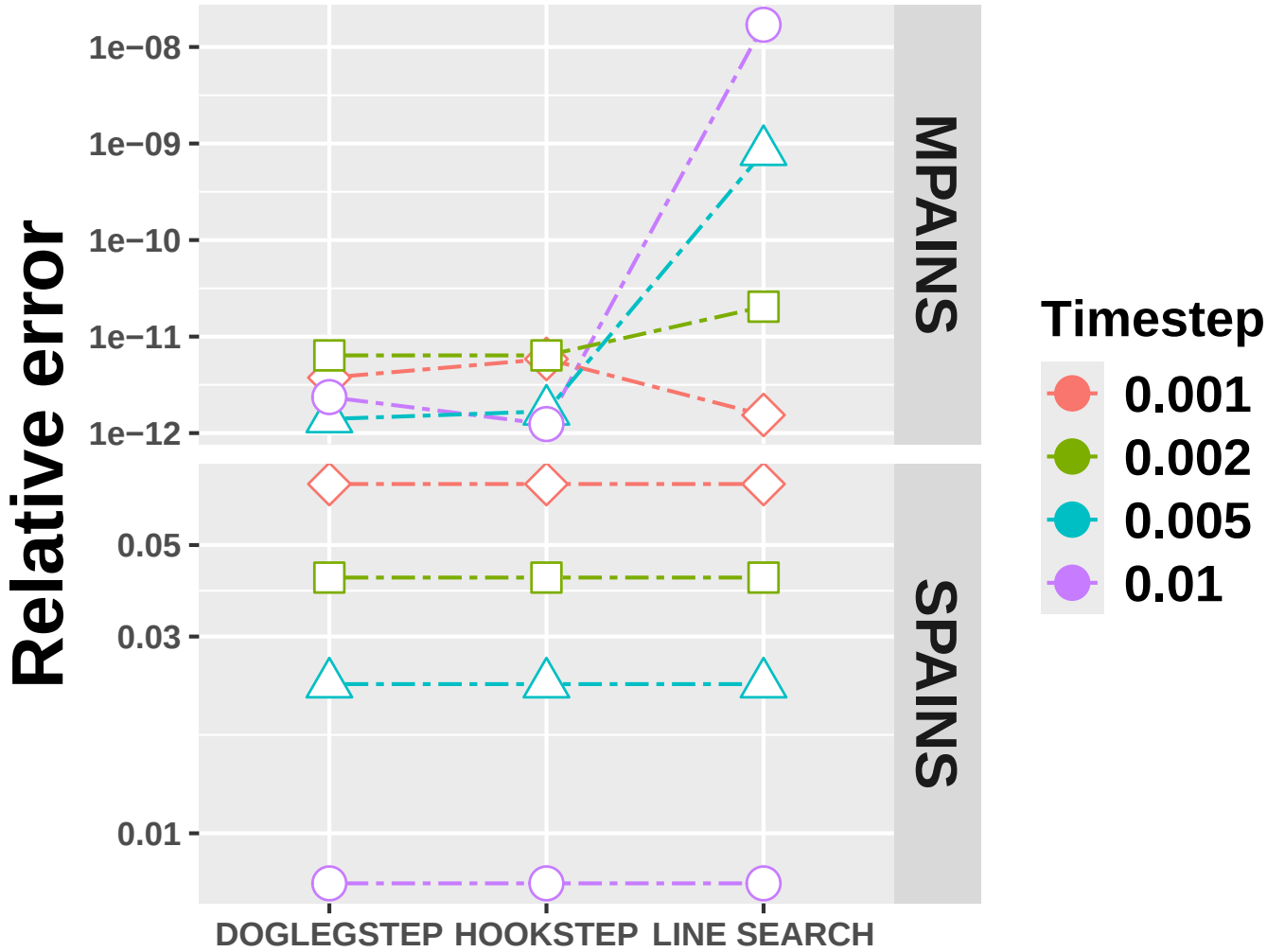


Figure: **Benchmark** 2. Numerical method **BDF4**.

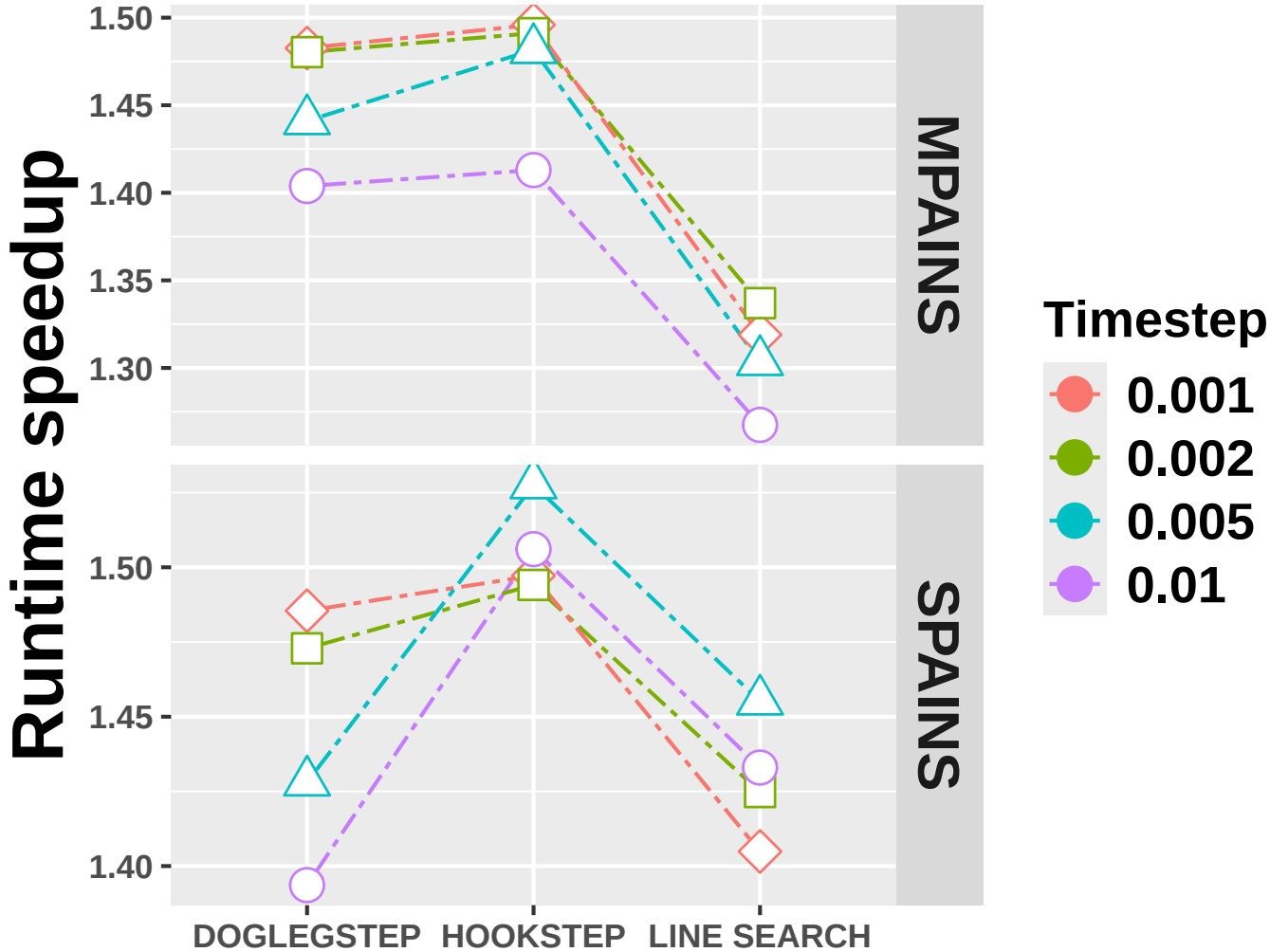


Figure: **Benchmark** 2. Numerical method **BDF4**.

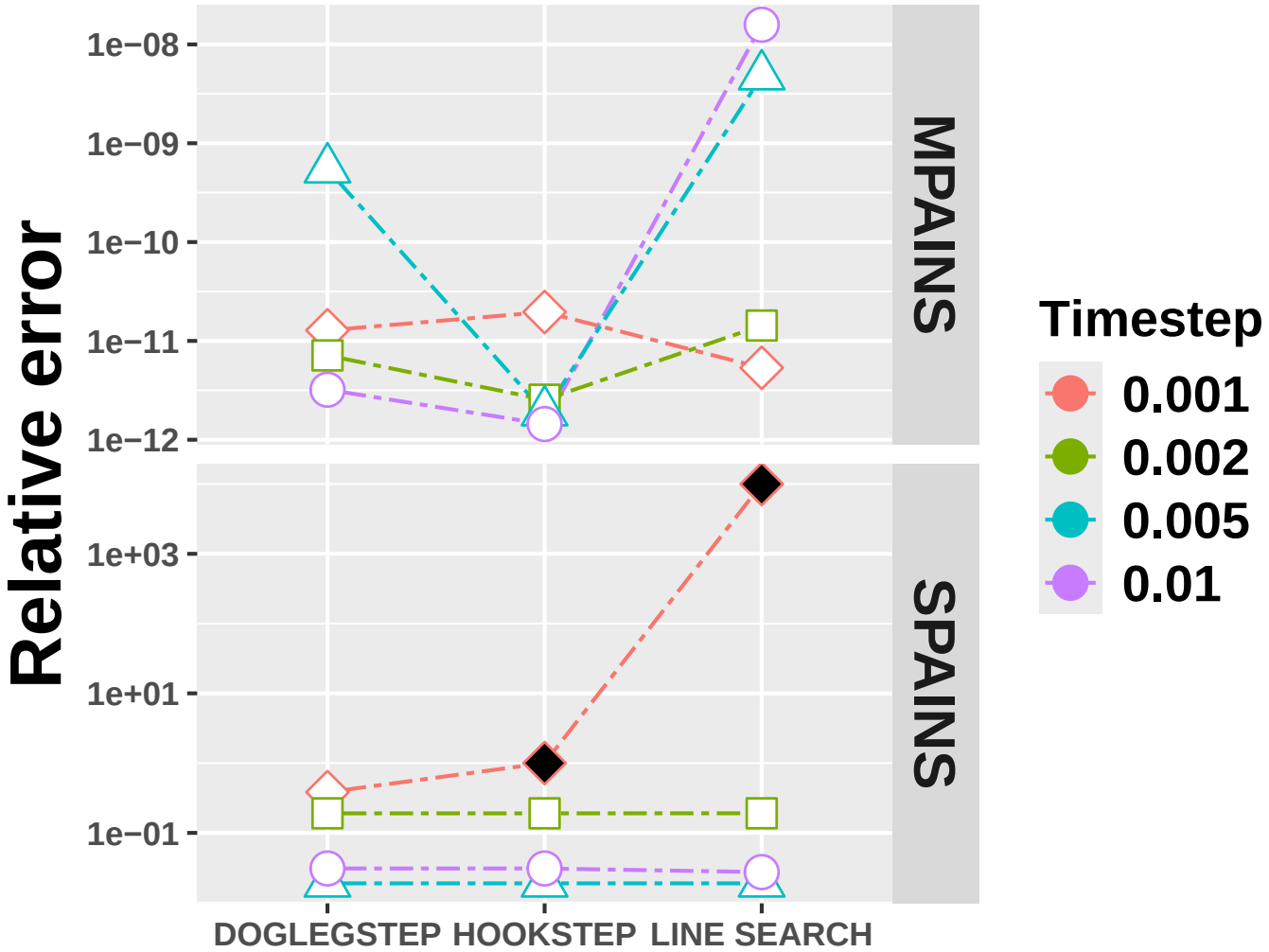


Figure: **Benchmark** 2. Numerical method **BDF5**.

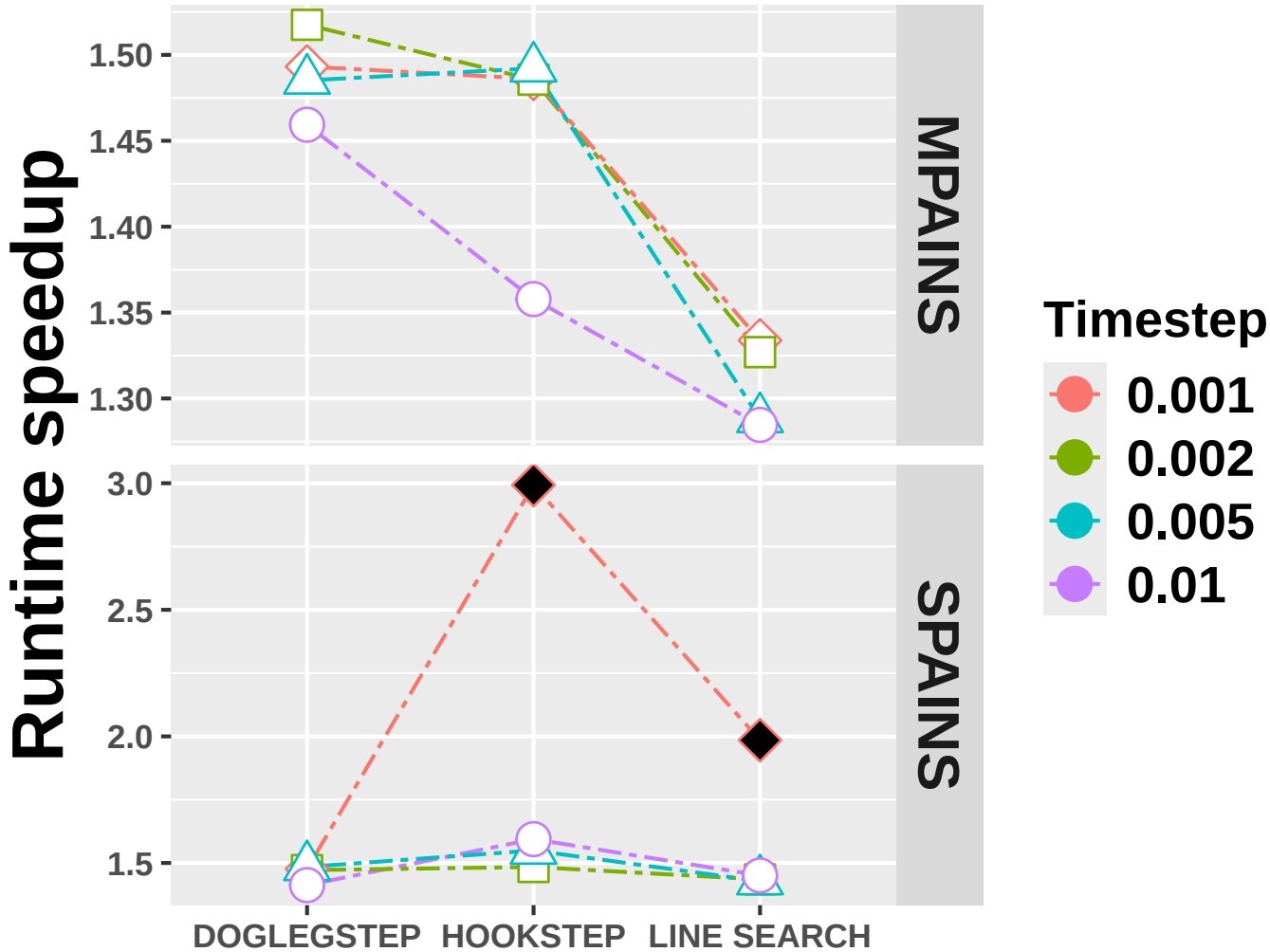


Figure: **Benchmark** 2. Numerical method **BDF5**.

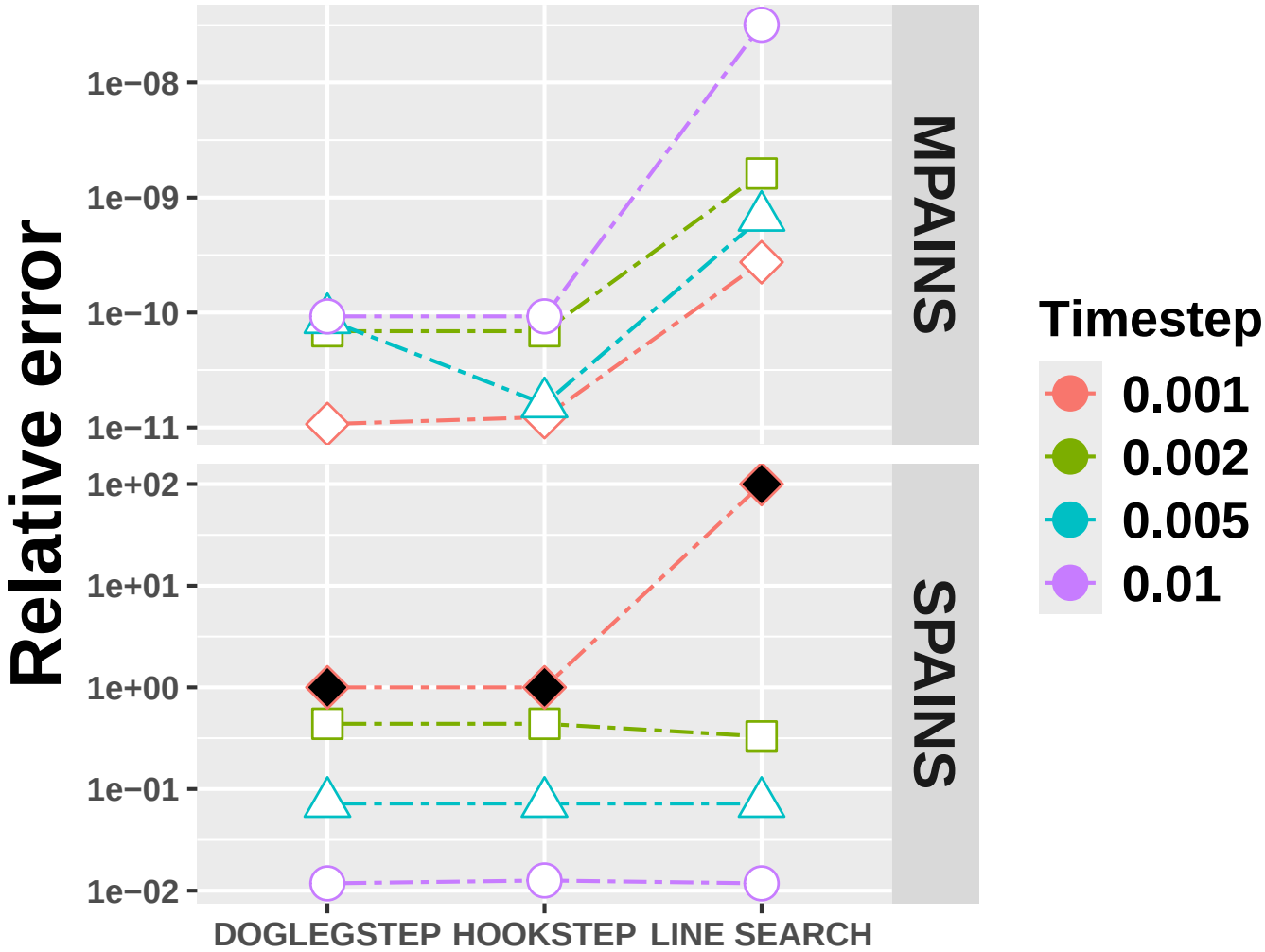


Figure: **Benchmark** 2. Numerical method **BDF6**.

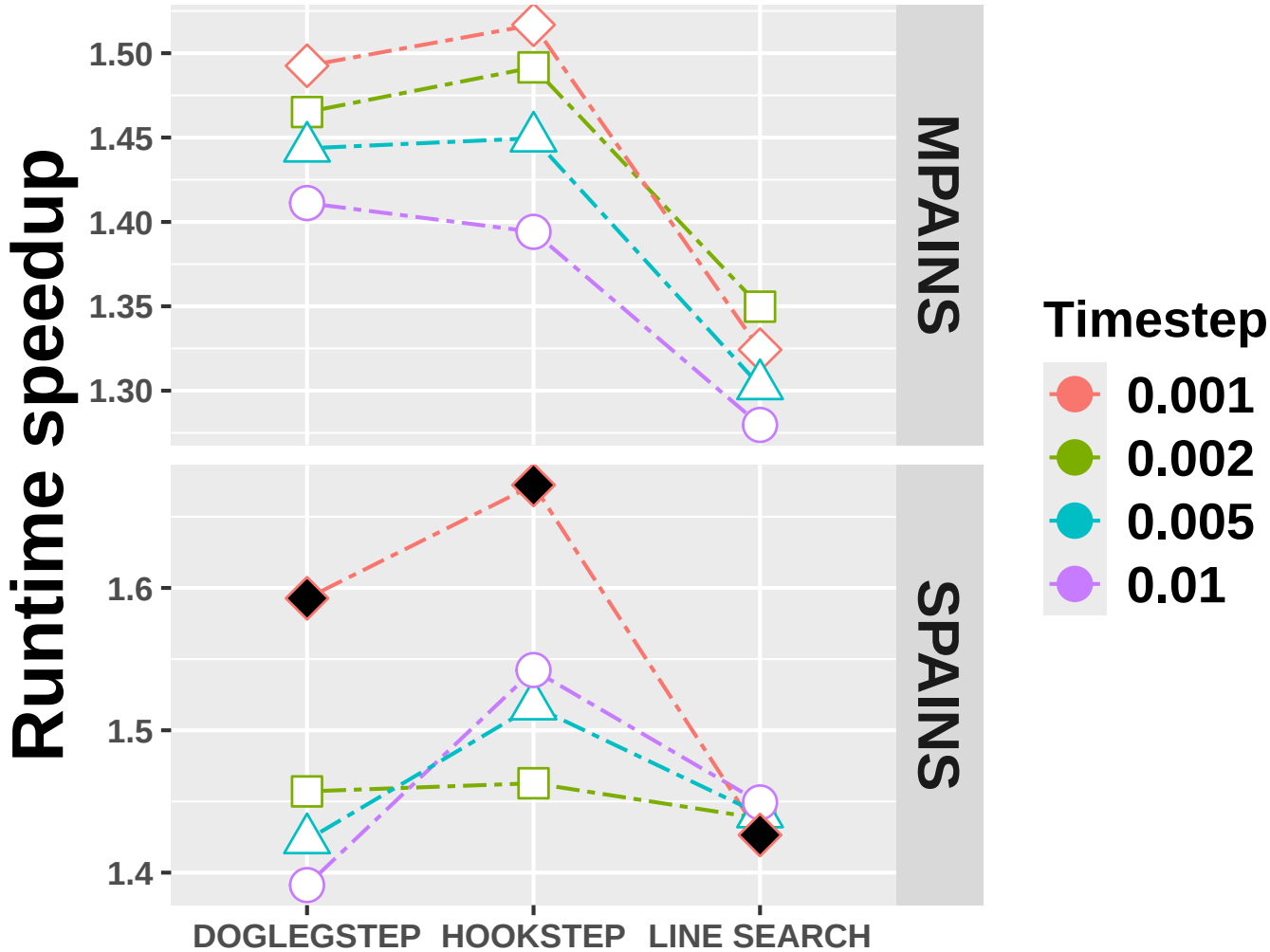


Figure: **Benchmark** 2. Numerical method **BDF6**.

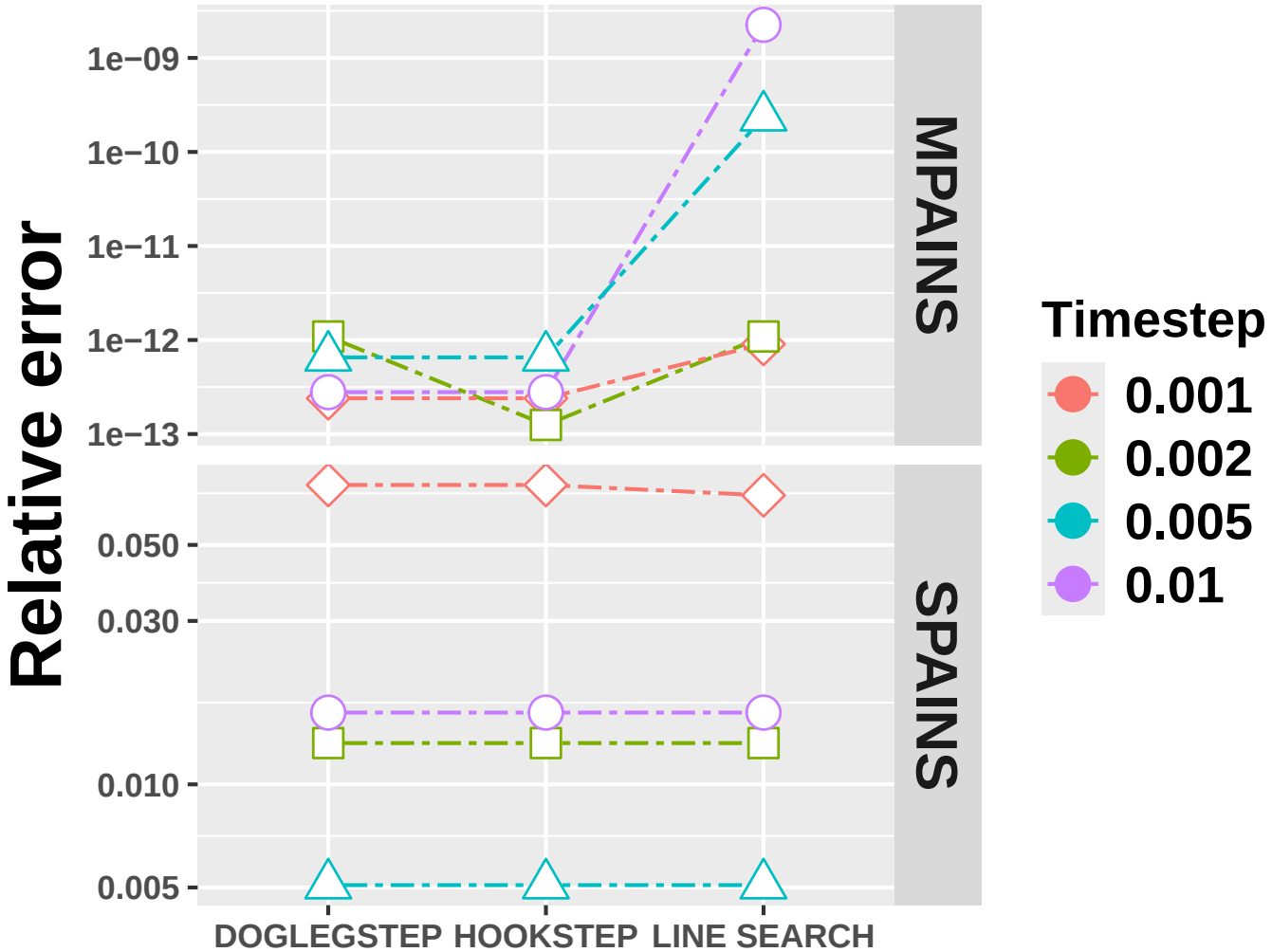


Figure: **Benchmark** 2. Numerical method **AM2**.

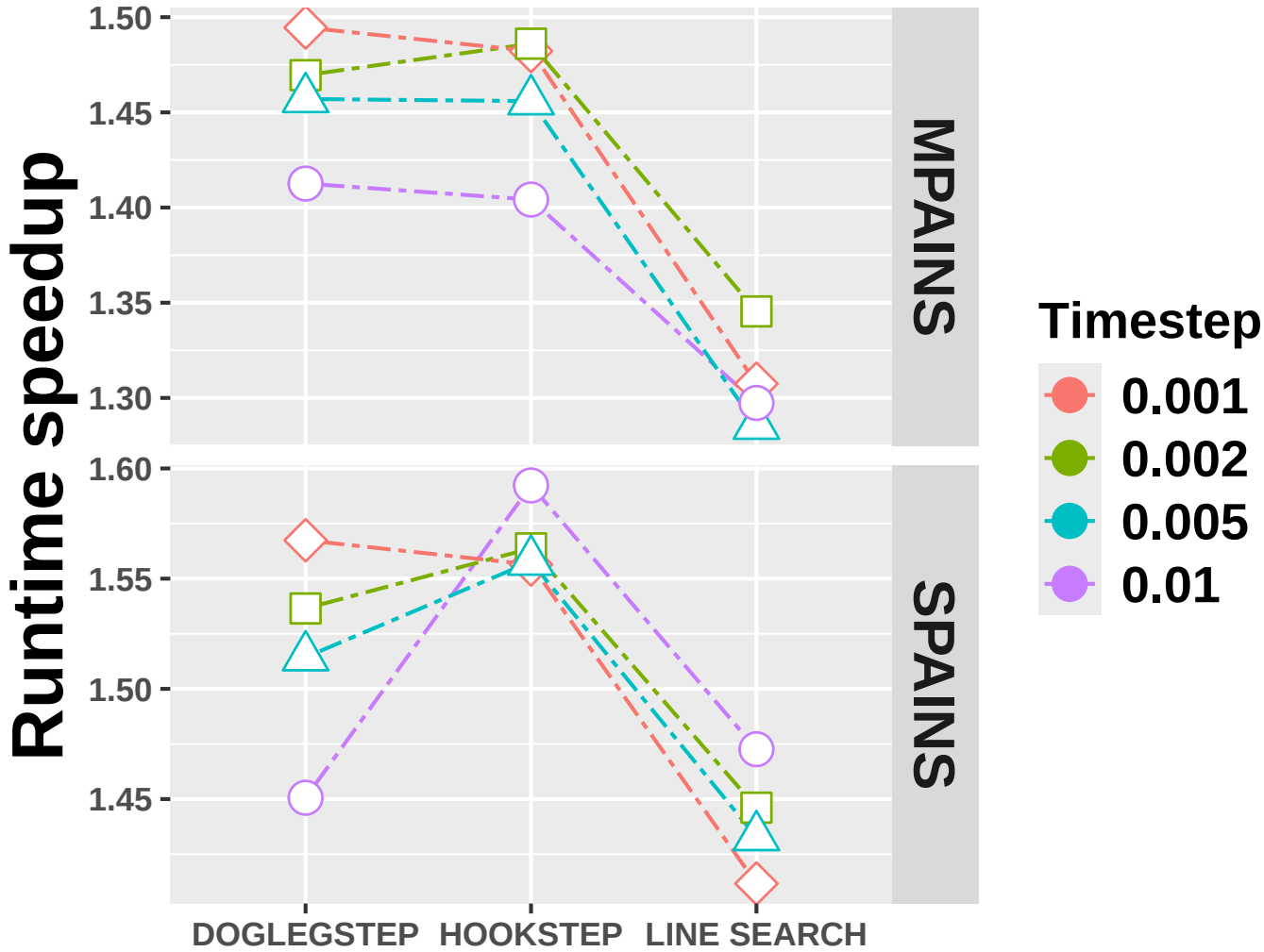


Figure: **Benchmark** 2. Numerical method **AM2**.

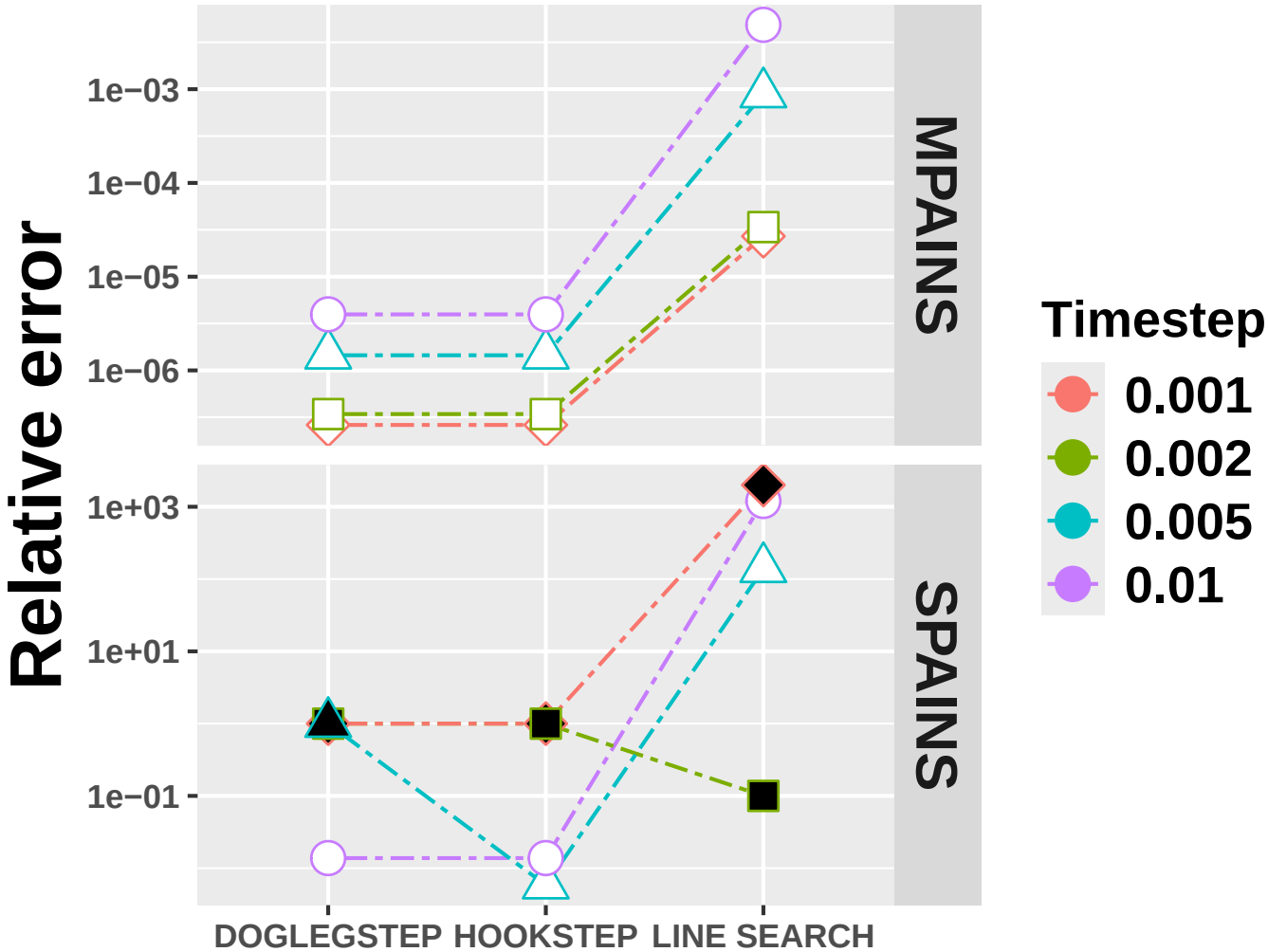


Figure: **Benchmark** 2. Numerical method **IRK4**.

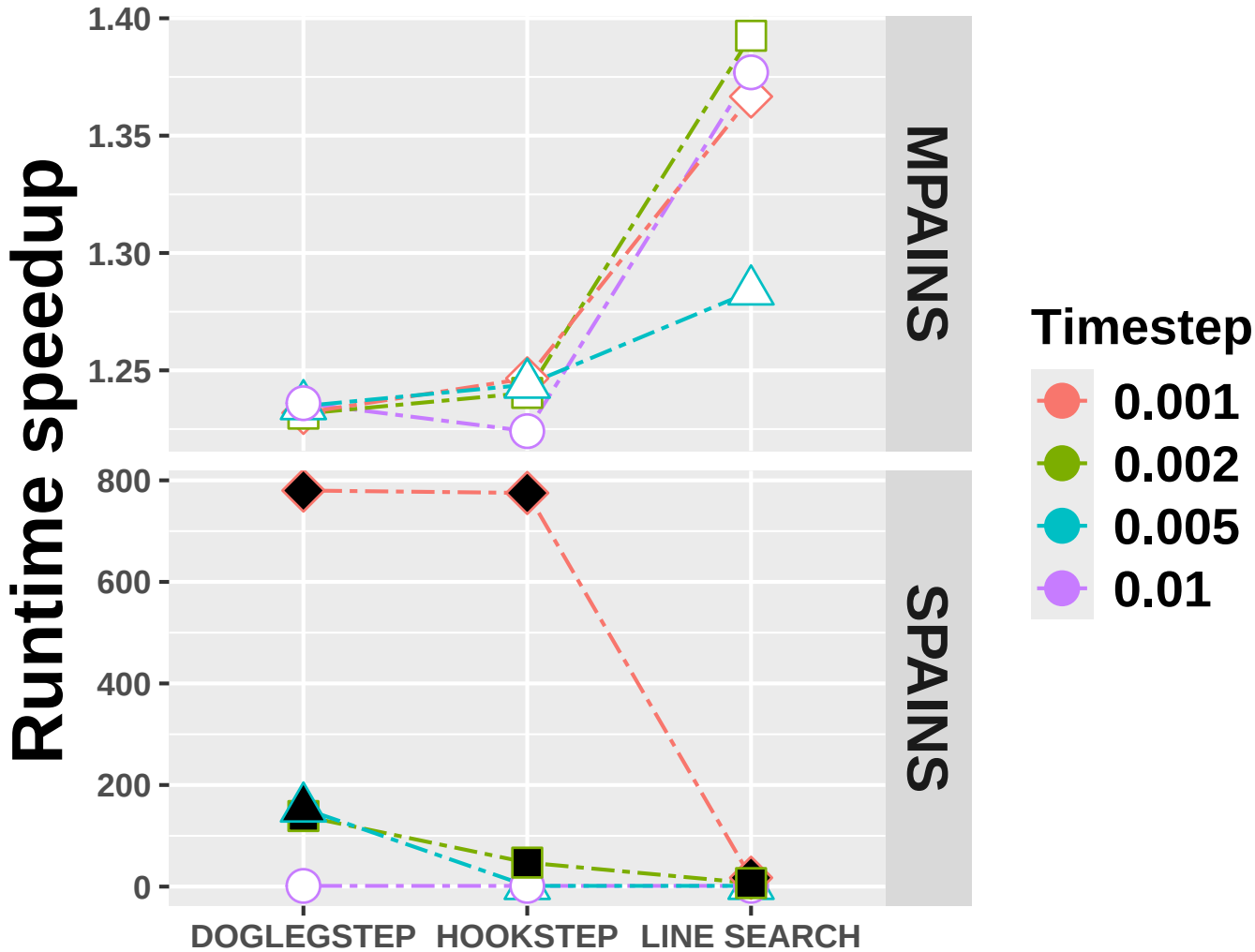


Figure: **Benchmark** 2. Numerical method **IRK4**.

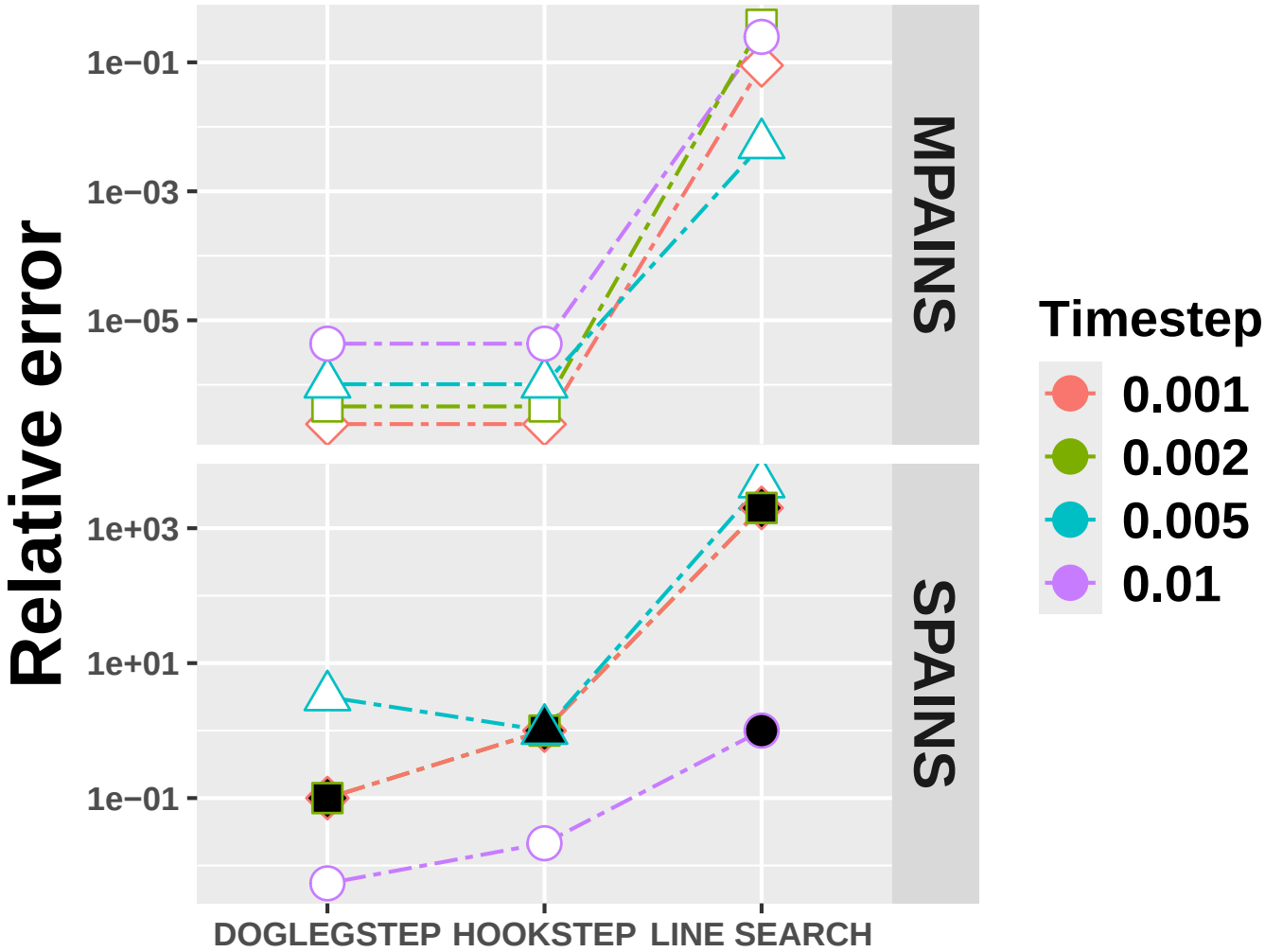


Figure: **Benchmark** 2. Numerical method **IRK6**.

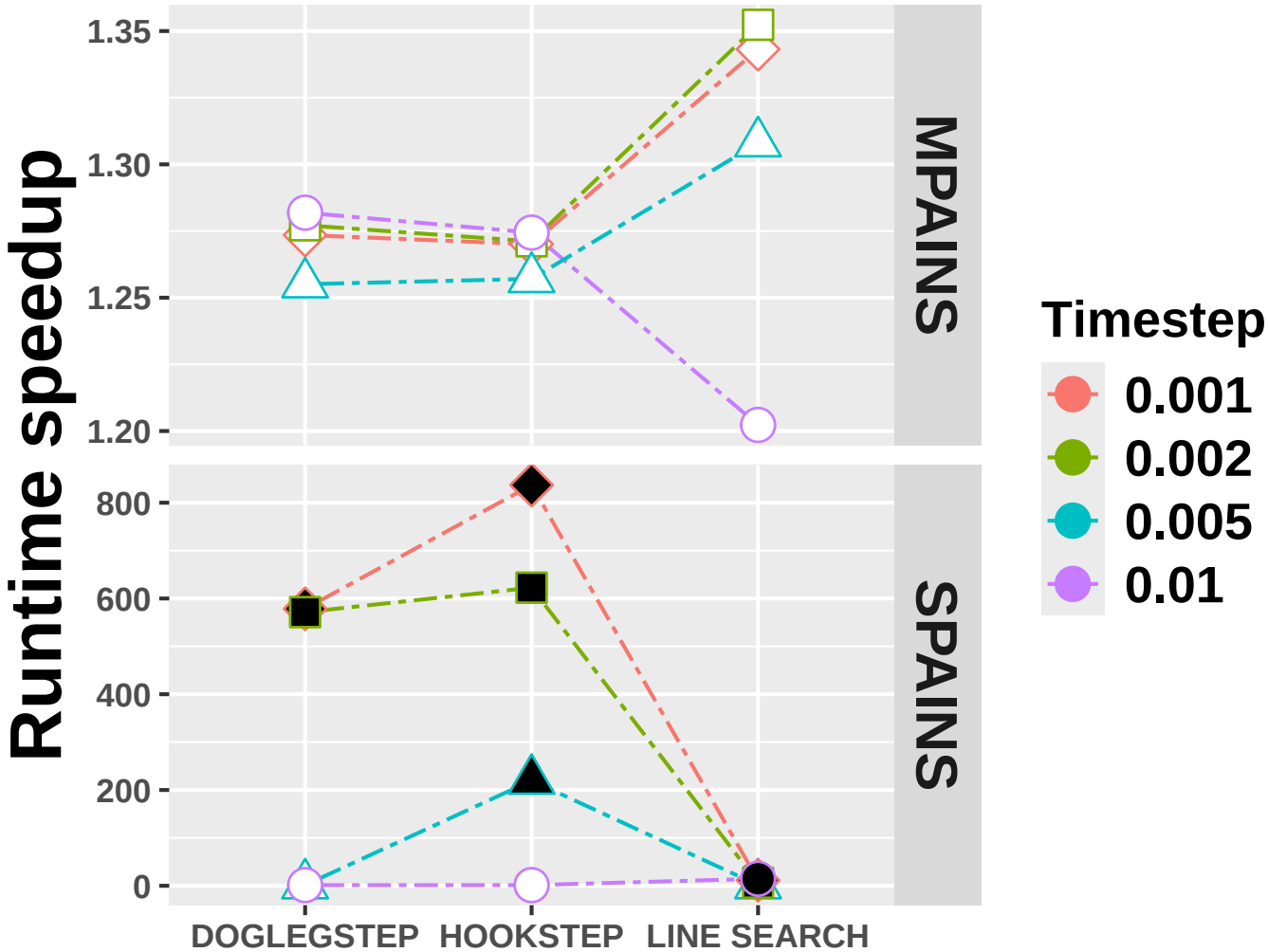


Figure: **Benchmark** 2. Numerical method **IRK6**.

Conclusions

- We have implemented, using mixed precisions, implicit numerical methods for solving large ODE's.
- The **mixed methods are up to** 2 **times faster** than those running in double precision with sufficient accuracy.

Future works

- Implemented the mixed methods in other language (C, C++,...)
- Implemented Mixed precision's numerical methods on a Tensor Processing Unit (matrix processor)
- Implemented our approaches on a GPU(Graphics Processing Unit)
- Use other type of precisions (half, quadruple,....)

Mixed Precision Approach

- **M. Al Sayed Ali, S. Bernard, A. Marzorati, J. Rouzaud-Cornabas:** *Mixed-Precision implicit numerical schemes for systems of ordinary differential equations*, Numer Algor (2025).

- **M. Al Sayed Ali, S. Bernard, A. Marzorati, J. Rouzaud-Cornabas:** *Mixed-Precision in adaptive Runge-Kutta method for large ODE systems*, **submitted** in SIAM Journal on Scientific Computing, (2025).
- **M. Al Sayed Ali, S. Bernard, A. Marzorati, J. Rouzaud-Cornabas:** *Mixed precision explicit numerical methods for ordinary differential equations*, **DRAFT**.

Galerkin Approach

Replace $F(x) = 0,\ x \in \mathbb{R}^n$ by

$$V^T F(Vz) = 0,\ z \in \mathbb{R}^p$$

where $dim(Vect(V)) = p \ll n$ and $x \approx Vz$.

Galerkin Approach

Replace $F(x) = 0,\ x \in \mathbb{R}^n$ by

$$V^T F(Vz) = 0,\ z \in \mathbb{R}^p$$

where $dim(Vect(V)) = p \ll n$ and $x \approx Vz$.

- **M. Al Sayed Ali**: *Accélération de schémas d'intégration temporelle pour les équations différentielles algébriques linéaires*, CRAS, Ser. I 346(2008), 229 – 232.

- **M. Al Sayed Ali, M. Sadkane :** *Acceleration of implicit schemes for large systems of nonlinear ODEs,* Electronic Transactions on Numerical Analysis, Vol. 35(2009), 104 – 117.

Galerkin Approach

- **M. Al Sayed Ali, M. Sadkane :** *Improved predictor schemes for large systems of linear ODEs,* Electronic Transactions on Numerical Analysis, Vol. 39(2012), 253 – 270.

- **M. Al Sayed Ali, M. Sadkane :** *Acceleration of implicit schemes for solving large systems of nonlinear differential-algebraic equations,* AIMS Mathematics, 5(1)(2019), 603 – 618.

Galerkin Approach

- **M. Al Sayed Ali, M. Sadkane :** *Acceleration of implicit schemes for large linear systems of differential-algebraic equations,* J. Comput. Appl. Math. *Vol*.389, 113364, (2021).

- **M. Al Sayed Ali, M. Sadkane :** *Acceleration of implicit schemes for large systems of delay differential equations*, J. Comput. Appl. Math., *Vol*.473, 116863, (2026).

Thank you for your attention.

Scaling for benchmark 1

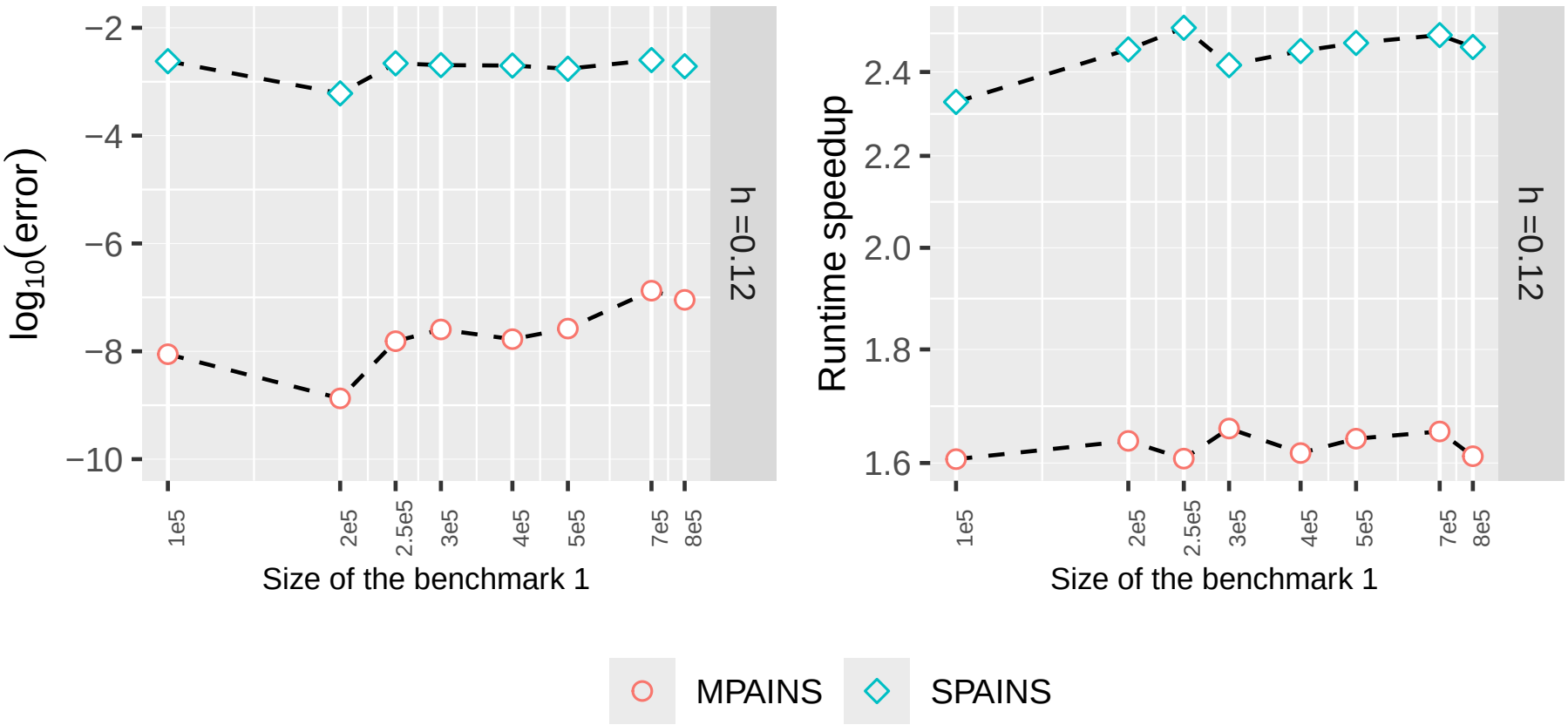


Figure: Benchmark 1 solved by **BDF1**. The x-axis shows the size of the benchmark. Relative error (left) and runtime speedup (right).

The tolerance ε is fixed

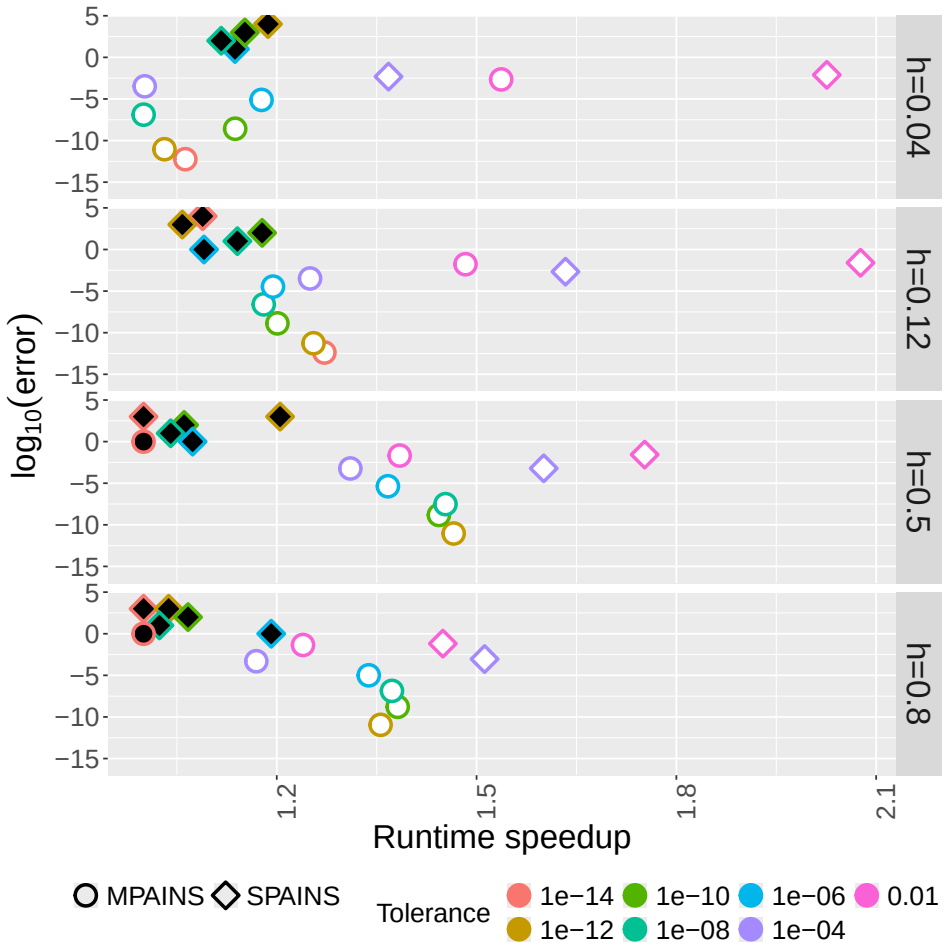


Figure: **Benchmark 1-LS** with multiple values of ε **(BDF1)**

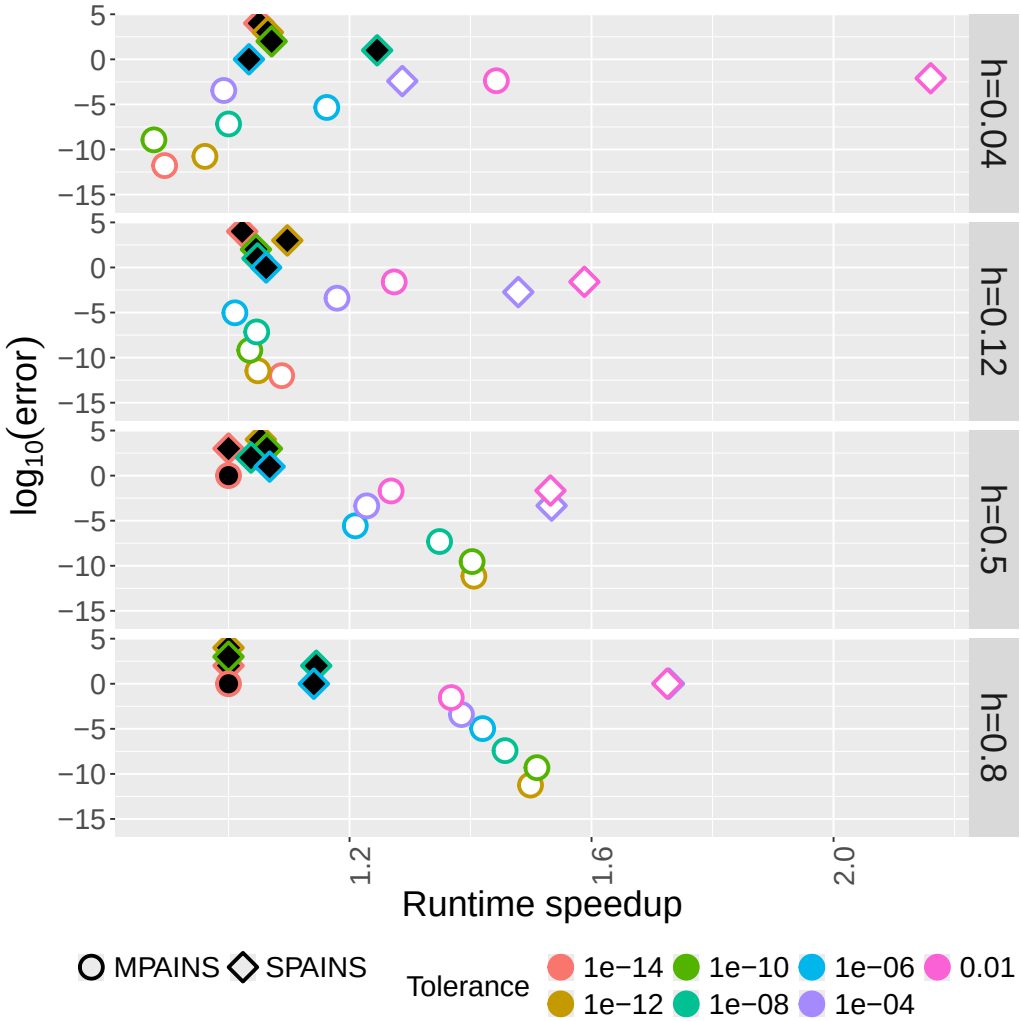


Figure: **Benchmark 1-LS** with multiple values of ε **(BDF2)**

Thank you for your attention.