

Linear Time Complexity Analysis of Time-Periodic Delayed Systems with Multiplication Free Semi-Discretization Method

Journal Title
XX(X):1–15
©The Author(s) 0000
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/

SAGE

Daniel Bachrathy¹

Abstract

This study introduces the Multiplication Free Semi-Discretization Method (MFSD), a novel approach for the efficient stability and fixed-point analysis of time-periodic Delayed Differential Equations (DDEs). Efficient stability prediction is critical for mitigating instabilities in mechanical systems, such as machine tool chatter in milling and vibration in delayed control systems, where high-accuracy modelling often leads to prohibitive computational costs. Unlike traditional semi-discretization methods that are burdened by high computational demands due to repeated matrix multiplication, the MFSD method leverages a composite mapping structure inspired by collocation techniques to achieve linear time complexity ($\mathcal{O}(p^1)$). The performance of the method is validated through extensive spectral analysis of the delayed Mathieu equation and further demonstrated on three practical engineering cases: a seasonal biological model, a multi-cutter turning system with spindle speed variation, and a finite-element-method-based longitudinal vibration of an elastic beam under delayed boundary feedback. Results demonstrate that MFSD maintains linear scaling even for system resolutions up to 10^6 , enabling high-fidelity stability mapping even for high-dimensional structural dynamics that were previously computationally inaccessible. The algorithm is provided as an open-source tool via the `SemiDiscretizationMethod.jl` package to support the vibration and control research community.

Keywords

time-delay, semi-discretization, time complexity, stability, periodic solution

Introduction

Delayed Differential Equations (DDEs) play a pivotal role in modelling dynamics where time delays are intrinsic, such as in control systems Åström and Murray (2021); Habib et al. (2016); Bartfai and Dombovari (2022), biological processes Stepan (2009); Insperger et al. (2022); Kovacs and Insperger (2018), vehicle dynamics Li et al. (2024); Mi et al. (2024) and mechanical vibrations Bartfai et al. (2024); Horvath and Takacs (2023); Sanz-Calle et al. (2024); Zhang and Xi (2022). The stability analysis of DDEs is crucial for predicting system behaviour and ensuring reliability Ali et al. (1998); Kiss et al. (2022); Borgioli et al. (2020). However, calculating the stability of DDEs poses significant computational challenges, especially when it is critical to capture the system's dynamics accurately Zatarain and Dombovari (2014); Toth et al. (2017); Lehotzky et al. (2016); Lichtner et al. (2011); Itovich et al. (2024).

There are several available methods, most of them in frequency- and time domains. In the frequency domain, conventional approaches, such as calculating stability boundaries, involve the critical frequency as an additional parameter Stepan (2009); Ozturk and Budak (2008); Bachrathy and Stepan (2013). These methods often require complex integration along the frequency parameter, limiting their practicality. Alternatively,

time-domain methods, including collocation-based and integration-based strategies, offer solutions but with their own sets of challenges. Collocation methods and time-finite element methods Butcher and Bobrenkov (2011); Lehotzky and Insperger (2016); Lehotzky et al. (2018); Vyasarayani et al. (2014); Breda et al. (2005); Bayly et al. (2003), known for their excellent convergence properties, face difficulties with discontinuities, making them less robust and more complex to implement for general cases.

Despite these limitations, collocation-based approaches continue to be applied successfully in structural dynamics problems involving delay, particularly when combined with finite element discretisation in space. For example, recent studies have applied Chebyshev-based discretisation to analyse the delayed stability of delaminated composite beams subjected to follower forces Szekrényes (2023, 2025),

¹Department of Applied Mechanics, Faculty of Mechanical Engineering, Budapest University of Technology and Economics, Műgyetem rkp. 3., H-1111 Budapest, Hungary.
Tel.: +36-1-463-1369
Email: bachrathy@mm.bme.hu

Corresponding author:

Daniel Bachrathy, Department of Applied Mechanics, Faculty of Mechanical Engineering, Budapest University of Technology and Economics, Budapest, Hungary.

demonstrating the practical relevance of such methods in vibration problems with distributed parameters and time-delay feedback.

Our focus is on an integration-based strategy, specifically the semi-discretization method Insperger and Stépán (2002); Insperger and Stepan (2004); Insperger and Stépán (2011); Sykora and Bachrathy (2020) and its alternatives Ozoegwu (2018); Insperger (2010); Liu et al. (2024); Yang et al. (2023), such as full-discretizations. Despite their limited convergence, these methods are favoured for their ease of implementation. The trade-off between convergence limitations and implementation ease is often acceptable, especially when handling systems with non-smooth behaviour.

Traditionally, these methods involve repeated matrix multiplication, significantly increasing CPU time demands. In Henninger and Eberhard (2008), the method has been optimized based on the special matrix structure leading to quadratic time complexity. Current literature primarily concentrates on improving the eigenvalue convergence rate rather than computational efficiency, attempting to achieve more accurate eigenvalues at lower resolution Ozoegwu (2018).

However, it's crucial to note the trend in newer techniques that promise better time complexity of the eigenvalues of the mapping with fewer resolution steps. Often, these methods are demonstrated with a very limited resolution range (<100) Ozoegwu (2018); Liu et al. (2024); Yang et al. (2023), and usually not in a log-log scale, making it challenging to discern true convergence rates and the time complexity. At lower resolution levels, the nuances in implementation can significantly influence outcomes, potentially resulting in inaccurate conclusions. Furthermore, these higher-order methods have similar difficulties with discontinuities as the collocation methods.

We have to note that there is a new class of time-domain methods, which are based on direct simulation of the investigated DDE Zatarain and Dombovari (2014); Zatarain et al. (2015); Toth et al. (2017). These methods iteratively extract the eigenvalue/eigenvectors of the underlying mappings from the simulated timeseries and promise excellent work-precision diagrams. Unfortunately, their iterative nature makes it hard to compare them based on time complexity to the above-mentioned time domain methods where exact resolution-CPU time can be determined.

This study extends the integration-based time-domain methods by introducing a novel approach inspired by collocation techniques, enabling a linear time complexity implementation. The main aim is to overcome the computational limitations of traditional semi-discretization methods by eliminating the need for repeated matrix multiplications. The development is specifically presented for the semi-discretization method but is applicable to any approach that involves step-wise matrix multiplication, significantly reducing the computational cost.

Section 2 outlines the fundamental steps of the semi-discretization method and analyzes its computational

time complexity. Additionally, it briefly reviews prior advancements aimed at improving its efficiency. In Section 3, we introduce the Multiplication Free Semi-Discretization (MFSD) method, detailing the formulation of left and right mapping matrices for various time-period and time-delay ratios. Section 4 evaluates the computational performance of the proposed approach through high-resolution work-precision diagrams for both the conventional Semi-Discretization (SD) method and the introduced MFSD. The results clearly demonstrate that the new method achieves linear time complexity.

We've implemented our method in the publicly available `SemiDiscretizationMethod.jl` package, aiming to contribute to the broader community's ability to analyze and predict the behaviour of systems governed by DDEs more efficiently.

Main Contributions

While the algebraic concept of direct matrix composition has been previously mentioned for specific cases (e.g., in Hamann and Eberhard (2018) for $T = \tau$), its profound algorithmic implications have remained unrecognized. This study establishes MFSD as a comprehensive computational framework with the following core contributions:

- **Discovery of Linear Complexity:** We demonstrate that MFSD, when coupled with sparse matrix data structures, reduces the time complexity of stability analysis from nearly cubic $\mathcal{O}(p^{2.8})$ to strictly linear $\mathcal{O}(p^1)$.
- **Generalization to Arbitrary Ratios:** We develop the mathematical framework necessary to handle arbitrary period-to-delay ratios ($T \neq \tau$), providing a unified solution for general time-periodic systems.
- **Algorithmic Equivalence:** We demonstrate that the method is an exact algebraic rearrangement of the traditional semi-discretization mapping, ensuring identical numerical results without loss of accuracy.
- **Open-Source Tool:** We provide a highly optimized implementation in the publicly available `SemiDiscretizationMethod.jl` package, capable of evaluating previously impossible discretization resolutions (e.g., $p = 10^6$).

Basics of the Semi-discretization method

Consider the linear delayed dynamical system Insperger and Stépán (2011) of the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{x}(t - \tau), \quad (1)$$

where $\mathbf{x}(t) \in \mathbb{R}^d$ is the time dependent state vector of the d dimensional first order system. The system matrices $\mathbf{A}(t)$ and $\mathbf{B}(t) \in \mathbb{R}^{d \times d}$ are the T -periodic time-varying coefficient matrices, and τ is the time delay. The initial state of delay systems is described via a function $\mathbf{x}(t) = \phi(t)[- \tau, 0] \mapsto \mathbb{R}^d$. As the derivative $\dot{\mathbf{x}}(t)$ requires not only the current state $\mathbf{x}(t)$ but the delay state $\mathbf{x}(t -$

τ) as well, therefore the notation $\mathbf{x}^{(t)}(\vartheta)$, $\vartheta \in [t - \tau, t]$ will be used to denote a solution segment that includes the entire state history between times $t - \tau$ and t .

Finding the stability of the system (1) requires the spectral properties of the mapping that maps $\mathbf{x}^{(t)}(\vartheta)$ a full period T forward to $\mathbf{x}^{(t+T)}(\vartheta)$. In the semi-discretization method, the continuous state-space is sampled over a uniform time-grid with resolution Δt , resulting in the sampled states $\mathbf{x}_i = \mathbf{x}(t_i)$ at times $t_i = i\Delta t$. Fig. 1 shows the sketch of corresponding discretised one-step- and one-period mappings.

Using the discretised values \mathbf{x}_i , the solution segment $\mathbf{x}^{(t_i)}(\theta)$ is represented by the vector

$$\mathbf{y}_i = [\mathbf{x}_i \quad \mathbf{x}_{i-1} \quad \mathbf{x}_{i-2} \quad \dots \quad \mathbf{x}_{i-r}]^\top, \quad (2)$$

where

$$r \geq \tau/\Delta t. \quad (3)$$

The usual selection for the number of the sampled points is $r = \lceil \tau/\Delta t \rceil \in \mathbb{Z}_+$.

In order to capture the T -periodic mapping, the time increment Δt is required to be an integer fraction $p \in \mathbb{Z}_+$ of period T , i.e. $T = p\Delta t$, therefore, in practice p is selected first, leading to $\Delta t = T/p$. In the following description, for the sake of simplicity, the time delay τ is assumed to be $\tau = r\Delta t$.

The main idea of the semi-discretization method is that the time-varying components: $\mathbf{A}(t)$, $\mathbf{B}(t)$ and $\mathbf{x}(t - \tau)$ are considered to be constant for each Δt time segment. With this approximation, an analytical formula can be provided for the next sampled state \mathbf{x}_{i+1} using the elements of \mathbf{y}_i (see Eq. (2)):

$$\mathbf{x}_{i+1} = \mathbf{P}_i \mathbf{x}_i + \mathbf{R}_i \mathbf{x}_{i-r}. \quad (4)$$

In case Δt is chosen in a manner that $\tau = r\Delta t$ and $T = p\Delta t$, and $\mathbf{A}^{-1}(t_i)$ exists, then for zeroth-order semi-discretization Insperger and Stépán (2002) the coefficient matrices \mathbf{P}_i and \mathbf{R}_i have closed form solutions, i.e. $\mathbf{P}_i = e^{\mathbf{A}(t_i)\Delta t}$ and $\mathbf{R}_i = (e^{\mathbf{A}(t_i)} - \mathbf{I})\mathbf{A}^{-1}(t_i)\mathbf{B}(t_i)$. For the definition of \mathbf{P}_i and $\mathbf{R}_i \in \mathbb{R}^{d \times d}$ in higher-order cases*, see Insperger et al. (2008). To cover a full period, the mapping (4) should be repeated for p times. Without loss of generality, we set $t = 0$ as the starting point; in this case, we need to repeat the mapping (4) for $i = 0, 1, \dots, p-1$.

Using Eq.(4), the one-step mapping from \mathbf{y}_i to \mathbf{y}_{i+1} is defined by the one-step matrix $\mathbf{C}_i \in \mathbb{R}^{(r+1)d \times (r+1)d}$ as:

$$\mathbf{C}_i = \begin{bmatrix} \mathbf{P}_i & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{R}_i \\ \mathbf{I}_d & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_d & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_d & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & & & \ddots & & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I}_d & \mathbf{0} \end{bmatrix}, \quad (5)$$

leading to the state-space representation:

$$\mathbf{y}_{i+1} = \mathbf{C}_i \mathbf{y}_i, \quad (6)$$

where $\mathbf{I}_d \in \mathbb{R}^{d \times d}$ are identity matrices. The subdiagonal blocks in \mathbf{C}_i represent the temporal shift of the sampled state history between \mathbf{y}_i and \mathbf{y}_{i+1} .

Starting from the initial discretized state \mathbf{y}_0 , p recursive applications of the one-step mapping in Eq.(6) yield the state-space after one period:

$$\mathbf{y}_p = \mathbf{C}_{p-1} \dots \mathbf{C}_1 \mathbf{C}_0 \mathbf{y}_0. \quad (7)$$

The transition over a full period T is governed by the Monodromy matrix (or Floquet transition matrix) $\Phi \in \mathbb{R}^{(r+1)d \times (r+1)d}$, defined as:

$$\Phi := \mathbf{C}_{p-1} \dots \mathbf{C}_1 \mathbf{C}_0, \quad (8)$$

such that the one-period mapping is represented by $\mathbf{y}_p = \Phi \mathbf{y}_0$.

The stability properties of the trivial solution can be determined based on the spectral radius of the monodromy matrix (or Floquet transition matrix) Φ . The fixed point is asymptotically stable if the spectral radius ρ is smaller than 1, that is, the absolute value of the largest eigenvalue of Φ is smaller than 1 ($|\mu_{max}| < 1$).

Time complexity of the semi-discretization method

The time complexity of the traditional semi-discretization method was thoroughly analysed in Henninger and Eberhard (2008). Here we will focus on the effect of the time-step Δt on the spectral radius ($|\mu_{max}|$) only. Note that both p and r are inversely proportional to Δt . The semi-discretization algorithm has the following key steps:

- The computation of \mathbf{P}_i and \mathbf{R}_i requires the calculation of matrix exponents, which can take a significant portion of the time spent on computation for high-dimensional systems ($d \gg 1$). This step needs to be performed p times, once for each time step, therefore, it has linear time complexity $\mathcal{O}(\Delta t^{-1})$.
- The one-step matrices \mathbf{C}_i have $(r+1)^2 d^2$ elements. However, there are simple optimisation, that reduce the memory required. For example, the array containing \mathbf{C}_0 can be reused for \mathbf{C}_1 , \mathbf{C}_2 , Hence, after its use, it is no longer needed. Using sparse matrices further reduce memory requirements, as there are $(r+2)d^2$ elements that change for each \mathbf{C}_i . Combining sparse matrices with so-called "lazy arrays", where some elements of the matrix are computed upon indexing the array can further reduce memory requirements. With this technique, there is no need to store the identity matrices \mathbf{I} , as the location of the non-zero elements are distributed along an off-diagonal (easy-to-capture pattern). Therefore, the complexity of this step has linear complexity $\mathcal{O}(\Delta t^{-1})$ when implemented properly.

*In practice higher-order version should be used as they have a better convergence rate. Based on Insperger et al. (2008): "if piecewise constant approximation of the periodic coefficients is used, then the first-order approximation of the delayed term is the optimal choice", which leads to second-order convergence of the spectral radius in the time resolution Δt .

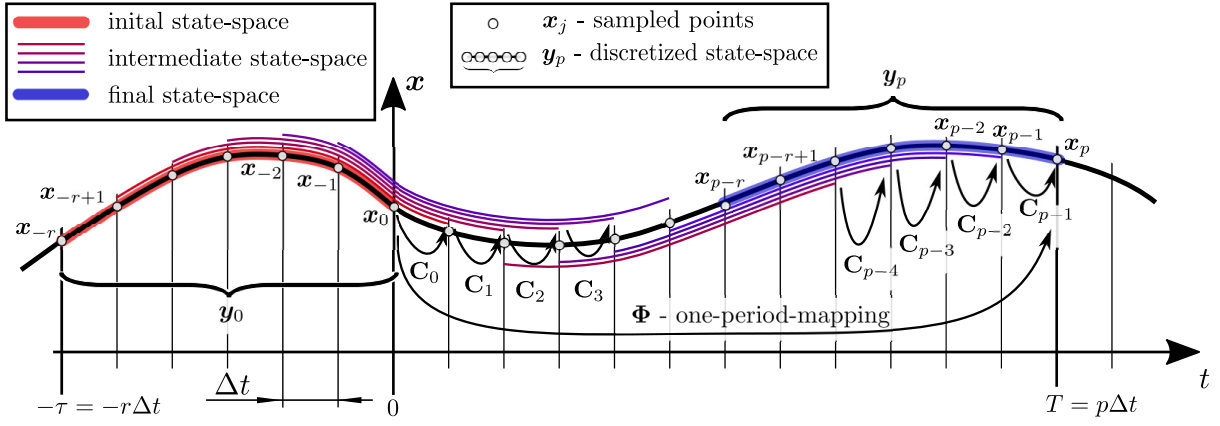


Figure 1. Schematic representation of the semi-discretization method. The vector \mathbf{y}_0 of the sampled \mathbf{x}_i ($i \in [-r, 0]$) states are mapped one time step ahead by the one-step matrices \mathbf{C}_j ($j \in [0, p-1]$) sequentially. The one-period mapping can be performed directly by the monodromy matrix Φ based on Eq.(7).

- The generation of monodromy matrix Φ can be the most time-consuming step, hence it requires p multiplications of $\mathbb{R}^{(r+1)d \times (r+1)d}$ sized matrices. The schoolbook multiplication has cubic complexity leading to $\mathcal{O}(p((r+1)d)^3)$, however, high-end algorithms can reach ~ 2.37 complexity Le Gall (2014). In the practical range of p values, mostly the Strassen-algorithm Strassen (1969) is used (e.g. in Matlab, BLAS implementation), which has ~ 2.8 complexity. In our case, it leads to approximately $\mathcal{O}(p((r+1)d)^{2.8}) = \mathcal{O}(\Delta t^{-3.8})$.
- The final step is to compute the eigenvalue with the largest magnitude of $\Phi \in \mathbb{R}^{(r+1)d \times (r+1)d}$. In case of full matrix representation of Φ this can be solved with $\mathcal{O}((r+1)^3 d^3) = \mathcal{O}(\Delta t^{-3})$ (through SVD factorization), or $\mathcal{O}(s(r+1)d^2) = \mathcal{O}(\Delta t^{-1})$ for sparse matrices, where s is the average number of non-zeros in Φ (through a row based on Lanczos algorithm Saad (2011)). Note, Φ is typically similar to a full upper triangular matrix, thus $s \sim p$ leading to quadratic time complexity $\mathcal{O}(\Delta t^{-2})$.

The most critical part is the recursive matrix multiplication in Eq.(7), therefore, in the literature, many different methods were proposed to speed up this step. In the following section, we will present the most common ones.

Following the History of the Necessary State Components Only In many problems, not all the component of the delayed state $\mathbf{x}(t - \tau)$ is used in the governing equation, i.e. the rank of $\mathbf{B}(t)$ is not full. This is more evident in the following form, which is commonly used in control theory:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \hat{\mathbf{B}}(t)\mathbf{u}(t - \tau) \quad (9)$$

$$\mathbf{u}(t) = \mathbf{D}(t)\mathbf{x}(t), \quad (10)$$

where $\mathbf{u}(t) \in \mathbb{R}^c$ is the control input, \mathbf{D} is $\mathbb{R}^{c \times d}$ and $\hat{\mathbf{B}}$ is $\mathbb{R}^{d \times c}$. These coefficients generate $\mathbf{B}(t) = \mathbf{D}(t)\hat{\mathbf{B}}(t)$ for Eq. (1). Based on Eqs. (9) and (10), we can reduce the size of \mathbf{y}_i by storing only the relevant components, yielding

$$\hat{\mathbf{y}}_i = [\mathbf{x}_i \quad \mathbf{u}_{i-1} \quad \mathbf{u}_{i-2} \quad \dots \quad \mathbf{u}_{i-r}]^T, \quad (11)$$

where $\hat{\mathbf{y}}_i \in \mathbb{R}^{d+cr}$. In this case, the one-step mapping $\hat{\mathbf{y}}_{i+1} = \hat{\mathbf{C}}_i \hat{\mathbf{y}}_i$ is formulated based:

$$\hat{\mathbf{C}}_i = \begin{bmatrix} \mathbf{P}_i & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \hat{\mathbf{R}}_i \\ \mathbf{D}_i & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_c & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_c & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & & & \ddots & & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I}_c & \mathbf{0} \end{bmatrix}, \quad (12)$$

where \mathbf{I}_c is the identity matrix with size $\mathbb{R}^{c \times c}$. The size reduction can be significant if $c \ll d$, leading to the size of the monodromy matrix $\mathbb{R}^{(d+cr) \times (d+cr)}$. However, this will not improve the complexity of the calculation as a function of the time-step (it remains $\mathcal{O}(p(d+cr)^{2.8}) = \mathcal{O}(\Delta t^{-3.8})$), but it does provide a constant improvement of approximately $(c/d)^{2.8}$.

Exploitation of the Matrix Structure In Henninger and Eberhard (2008), the recursive matrix multiplication of \mathbf{C}_i (or $\hat{\mathbf{C}}_i$) matrices is greatly reduced by exploiting the special structure of the matrix. In this method, the step matrix is considered to comprise three main blocks: $[\mathbf{P}_i \quad \mathbf{D}_i]^T$, $[\hat{\mathbf{R}}_i]$ and a diagonal block $[\mathbf{I}_{(r-1)c}]$. The multiplication is performed by the first two blocks with the corresponding row of the preceding step matrix. The multiplication with the unit-block matrix provides a shift in the final result, so it is not performed numerically. Note that with proper memory management, even the movement of the data (the shift) can be eliminated, hence, the final position of the elements in $\hat{\Phi}$ can be determined a priori. With this formulation, the time complexity is reduced to $\mathcal{O}(\Delta t^{-2})$ Henninger and Eberhard (2008).

Sparse matrix representation It is observed that most of the elements of the matrix \mathbf{C}_i are zero, suggesting that a sparse matrix data structure can substantially enhance computational speed. Following a detailed numerical analysis (discussed later), it has been determined that the complexity associated with constructing Φ

can be approximated as $\mathcal{O}(\Delta t^{-2.5}) \dots \mathcal{O}(\Delta t^{-2.8})$. This strategic optimization, which is a straightforward one-line implementation, achieves a comparable level of efficiency improvement to that outlined in the preceding subsection; however, the sparse matrix manipulation has a slight overhead.

Note that this sparse representation is not so efficient at the last few steps of the multiplication, where the resulting matrix typically becomes an almost full triangular matrix.

The improvements discussed above represent the most commonly adopted strategies to enhance the computational efficiency of the semi-discretization method. However, the authors are unaware of any other numerically validated optimisation that achieves a linear computational complexity.

Multiplication Free One-period Mapping.

In the following sections, the "Multiplication Free Semi-Discretization" (MFSD) is introduced and analysed for different T/τ ratios. The linear time complexity is presented through a detailed numerical test, and it is compared to the Semi-Discretization method, both combined with a sparse matrix representation of the corresponding mapping matrices.

In the Semi-Discretization method, each multiplication with the one-step matrices \mathbf{C}_i approximates an integration step of length Δt of the state variable \mathbf{y}_i , while the multiplication with the Φ matrix approximates a one-period integration of length T . In the proposed method, our viewpoint is different: it is sufficient to represent the dynamics by means of Eq. (4) for each time step to find the spectral properties, similarly to the collocation methods Butcher and Bobrenkov (2011); Lehotzky and Insperger (2016); Bayly et al. (2003).

This type of direct composition was briefly presented for SD in Hamann and Eberhard (2018) for $T = \tau$ only. However, it was viewed primarily as an alternative algebraic representation with a disadvantageously larger state vector. Its banded structure and very sparse properties, its generalisation to arbitrary period-to-delay ratios, and its profound effect on reducing the computational time complexity to $\mathcal{O}(p^1)$ were completely unrecognised and unanalyzed—aspects which form the primary focus of the following sections.

The first step is to rewrite the one-step mapping (4) to the form:

$$\mathbf{0} = \mathbf{I}_d \mathbf{x}_{i+1} - \mathbf{P}_i \mathbf{x}_i - \mathbf{R}_i \mathbf{x}_{i-r}, \quad (13)$$

leading to the matrix form:

$$\mathbf{0} = [\mathbf{I}_d \quad -\mathbf{P}_i \quad \mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad -\mathbf{R}_i] \times [\mathbf{x}_{i+1} \quad \mathbf{x}_i \quad \mathbf{x}_{i-1} \quad \mathbf{x}_{i-2} \quad \dots \quad \mathbf{x}_{i-r}]^\top, \quad (14)$$

as shown in the visual representation in the top section of Fig. 2.

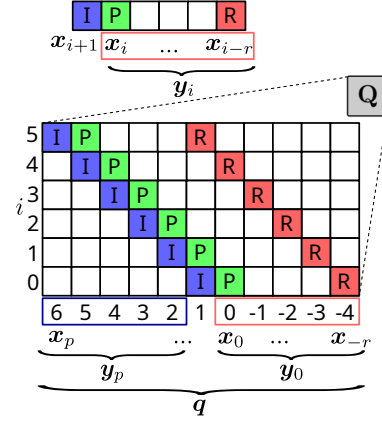


Figure 2. (top) Schematic representation of one-step equation of the semi-discretization method; (bottom) representation of all the equations of each time step (red and blue rectangle represent the collections of states which belong to the initial and final sampled state-space, respectively).

Repeating (14) for each index $i \in 0, 1, \dots, p-1$ yields the discretised

$$\mathbf{0} = \begin{bmatrix} \mathbf{I}_d & -\mathbf{P}_{p-1} & \mathbf{0} & \dots & -\mathbf{R}_{p-1} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_d & -\mathbf{P}_{p-2} & \mathbf{0} & \dots & -\mathbf{R}_{p-2} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_d & -\mathbf{P}_{p-3} & \dots & \mathbf{0} & -\mathbf{R}_{p-3} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & & & \ddots & \ddots & & & & & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I}_d & -\mathbf{P}_0 & \dots & \mathbf{0} & -\mathbf{R}_0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_p & \mathbf{x}_{p-1} & \mathbf{x}_{p-2} & \mathbf{x}_{p-3} & \dots & \mathbf{x}_0 & \dots & \mathbf{x}_{-r} \end{bmatrix}^\top. \quad (15)$$

Note, that the vector containing the discredited states from \mathbf{x}_{-r} to \mathbf{x}_p now covers the time interval $t \in [t - \tau, t + T]$. This extended state vector will be denoted by \mathbf{q} and the corresponding matrix coefficient by \mathbf{Q} . The bottom section of Fig. 2 shows the sketch of this composite description, with the coloured table depicting the matrix structure that emerges according to Eq. (15).

The goal of the spectral analysis is to perform a mapping of the state-space one period later. Therefore, the extended state vector needs to be split into two components that correspond to \mathbf{y}_p and \mathbf{y}_0 . However, the period T and the time delay τ are only equal in special cases, thus, the resolutions p and r are usually not equal. The following sections discuss the cases where $p-1 = r$, $p-1 > r$ and $p-1 < r$. The schematic representations of these cases are shown Fig. 3.

Partitioning of the equations for $p-1 = r$

In this case, all the necessary elements of the sampled vector \mathbf{y}_p and \mathbf{y}_0 is found in the extended state vector \mathbf{q} without overlap or missing elements (see Fig. 3 left). Therefore, Eq. (15) can be separated into two

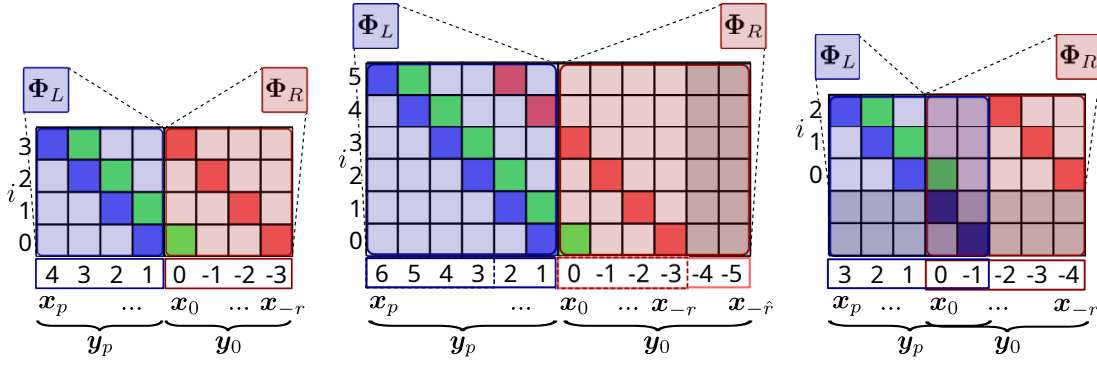


Figure 3. Schematic representation of the aggregated matrix structure for different p and r values. (left) $p - 1 = r$, (middle) $p - 1 > r$ and (right) $p - 1 < r$. \hat{r} represents the extended states (see: SubSection Partitioning of the equations for $p - 1 < r$). Red and blue rectangles represent the collections of states that belong to the initial and final sampled state space, respectively. Grey regions show the necessary extension to generate square left and right mapping matrices.

components:

$$\mathbf{0} = \begin{bmatrix} \mathbf{I}_d & -\mathbf{P}_{p-1} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_d & -\mathbf{P}_{p-2} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_d & -\mathbf{P}_{p-3} & \dots & \mathbf{0} & & \\ \vdots & & & \ddots & \ddots & & & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I}_d & -\mathbf{P}_1 & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{I}_d & \end{bmatrix} + \begin{bmatrix} \mathbf{x}_p & \mathbf{x}_{p-1} & \mathbf{x}_{p-2} & \mathbf{x}_{i-2} & \dots & \mathbf{x}_1 \end{bmatrix}^\top + \begin{bmatrix} -\mathbf{R}_p & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & -\mathbf{R}_{p-1} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathbf{R}_{p-1} & \mathbf{0} & \dots & \mathbf{0} & & \\ \vdots & & & \ddots & \ddots & & & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{R}_1 & \mathbf{0} & \\ -\mathbf{P}_0 & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & -\mathbf{R}_0 & \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 & \dots & \mathbf{x}_{-1} & \dots & \mathbf{x}_{-r} \end{bmatrix}^\top, \quad (16)$$

where rearranging the equation yields

$$\Phi_L \mathbf{y}_p = \Phi_R \mathbf{y}_0. \quad (17)$$

The partitioned 'left' component of \mathbf{Q} is denoted by Φ_L , while the negative of the 'right' component is denoted by Φ_R . The monodromy matrix Φ can be found by $\Phi = \Phi_L^{-1} \Phi_R$.

However, in softwares specialised for numeric computations (e.g. Matlab, Julia), there is an option to compute the eigenvalues of $\Phi = \Phi_L^{-1} \Phi_R$ via a generalised eigenvalue calculation function based on Φ_R and Φ_L , avoiding the matrix inversion. This results in the same eigenvalues and eigenvectors (disregarding the numerical errors) as the original Eq.(7). Because Φ_L essentially has identity matrices on its block diagonal (due to the \mathbf{I} terms in the left-hand side of the step-wise mapping), it is typically very well-conditioned, making the solution of the linear system $\Phi_L \mathbf{x} = \mathbf{b}$ (which is what iterative solvers do internally instead of explicitly forming $\Phi_L^{-1} \Phi_R$) numerically stable. The iterative solver evaluates the action of $\Phi_L^{-1} \Phi_R$ on a vector, and the sparsity pattern of Φ_L allows this to be computed using highly efficient sparse forward/backward substitutions. Therefore, the overall conditioning and the convergence rate of the eigenvalue

solver remain comparable to the traditional SD method, while the cost per iteration is vastly reduced.

Note, that constructing the sparse $\Phi_L, \Phi_R \in \mathbb{R}^{pd \times pd}$ matrix with a banded structure has linear time complexity and can be directly built after the computation of the submatrices \mathbf{P}_i and \mathbf{R}_i .

Partitioning of the equations for $p - 1 > r$

If the time period of the system is larger than the time delay $T > \tau$, then $p - 1 > r$. In this case, the extended state vector \mathbf{q} can be split into two separate vectors \mathbf{y}_p and \mathbf{y}_0 . However, \mathbf{y}_p and \mathbf{y}_0 do not capture the intermediate states $\mathbf{x}_i : i \in \{p - 1 - r, \dots, 2, 1\}$, and the corresponding equations. To remedy this, the value r should be updated to $\hat{r} = p - 1$, as this choice still satisfies the inequality (3). With this choice, there are more points to store, and at the corresponding locations in the matrix \mathbf{Q} of Eq. (15) must be padded with zeros (see the grey columns in the middle panel of the schematic representation in Fig. 3). With these considerations, the partitioned representation of the system of equations is

$$\mathbf{0} = \begin{bmatrix} \mathbf{I}_d & -\mathbf{P}_{p-1} & \mathbf{0} & \dots & -\mathbf{R}_{p-1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_d & -\mathbf{P}_{p-2} & \mathbf{0} & \dots & -\mathbf{R}_{p-2} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_d & -\mathbf{P}_{p-3} & \dots & \mathbf{0} & & \\ \vdots & & & \ddots & \ddots & & & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I}_d & -\mathbf{P}_1 & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{I}_d & \end{bmatrix} + \begin{bmatrix} \mathbf{x}_p & \mathbf{x}_{p-1} & \mathbf{x}_{p-2} & \mathbf{x}_{i-2} & \dots & \mathbf{x}_1 \end{bmatrix}^\top + \begin{bmatrix} \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & & & \vdots & & & \vdots \\ -\mathbf{R}_r & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \\ \vdots & \ddots & & & & & \vdots \\ \mathbf{0} & \dots & -\mathbf{R}_1 & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ -\mathbf{P}_0 & \dots & \mathbf{0} & -\mathbf{R}_0 & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 & \dots & \mathbf{x}_{-1} & \dots & \mathbf{x}_{-(p-1)} \end{bmatrix}^\top. \quad (18)$$

Although we use more states than the original method requires, thus $\Phi_L, \Phi_R \in \mathbb{R}^{(\hat{r}+1)d \times (\hat{r}+1)d} = \mathbb{R}^{pd \times pd}$ which might initially seem to lead to higher computational time if $T \gg \tau$. However, in sparse representation, the zero padding in the matrix \mathbf{Q} is 'free'. As we will quantitatively demonstrate in Section 4 (where ratios

of $T/\tau = 0.1, 1$, and 10 are tested), the combined CPU time exhibits almost no difference between the cases where $T/\tau = 1$ and $T/\tau = 10$. Even for $T/\tau = 0.1$, the total computation time remains practically identical. Therefore, the linear time complexity of the algorithm results in vastly more efficient computations than the original algorithm across a wide range of practical parameters.

Note that expanding the state vector \mathbf{y}_0 to store more historical points than the original delay τ requires does not alter the dynamics of the discretized system, nor does it introduce spurious numerical modes. The additional points are simply used to "store" the data for internal points. If $T > \tau$, the state in $[-\tau, 0]$ does not fully describe the time-periodic solution; it merely serves as the initial state necessary for a simulation to eventually provide the periodic solution. With this expanded state vector, the full periodic solution over T is provided directly. Due to the sparse matrix structure, this extended capacity comes for "free" without increasing the CPU time.

Partitioning of the equations for $p - 1 < r$

If $T < \tau$ and $p - 1 < r$, then there is an overlap between the \mathbf{y}_p and \mathbf{y}_0 (right panel of Fig. 3). In this case, the nonzero elements of matrix \mathbf{Q} corresponding to the overlapping elements in \mathbf{y}_p and \mathbf{y}_0 are free to be assigned to either $\hat{\Phi}_L$ or $\hat{\Phi}_R$, a decision not influencing the end result. In this work, the coefficient matrix \mathbf{Q} in Eq. (15) is partitioned as follows: the left side of matrix \mathbf{Q} goes into $\hat{\Phi}_L \in \mathbb{R}^{pd \times pd}$, chosen to be a square matrix, while the remaining elements go into $\hat{\Phi}_R \in \mathbb{R}^{pd \times (r+1)d}$ chosen to be rectangular (see the Fig. 3). Finally, the corresponding system of equations must be completed by identity matrices that account for the equality of the overlapping elements, resulting in the system:

$$\begin{bmatrix} \hat{\Phi}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{(r+1-p)d} \end{bmatrix} \mathbf{y}_p = \begin{bmatrix} \hat{\Phi}_R & \mathbf{0} \\ \mathbf{I}_{(r+1-p)d} & \mathbf{0} \end{bmatrix} \mathbf{y}_0, \quad (19)$$

$$\hat{\Phi}_L \mathbf{y}_p = \hat{\Phi}_R \mathbf{y}_0,$$

where $\mathbf{I}_{(r+1-p)d}$ is the identity matrix $\in \mathbb{R}^{(r+1-p)d \times (r+1-p)d}$, thus both $\hat{\Phi}_L$ and $\hat{\Phi}_R \in \mathbb{R}^{(r+1)d \times (r+1)d}$.

Computation of the periodic solution

In many cases, the investigated system has periodic external forcing:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{x}(t - \tau) + \mathbf{c}(t), \quad (20)$$

where $\mathbf{c}(t) = \mathbf{c}(t + T)$, which leads to a non-zero periodic solution. Then applying the semi-discretisation yields the affine map

$$\mathbf{x}_{i+1} = \mathbf{P}_i \mathbf{x}_i + \mathbf{R}_i \mathbf{x}_{i-r} + \mathbf{v}_i \quad i \in 0, 1, \dots, p-1, \quad (21)$$

see the definition of \mathbf{v}_i in Insperger and Stépán (2011). The corresponding multiplication-free scheme is

$$\hat{\Phi}_L \mathbf{y}_p = \hat{\Phi}_R \mathbf{y}_0 + \mathbf{V}, \quad (22)$$

where $\mathbf{V} = [-\mathbf{v}_{p-1} \quad -\mathbf{v}_{p-2} \quad \dots \quad -\mathbf{v}_0]^\top$. Note that in the case when $p - 1 < r$, the vector \mathbf{V} should be padded with zeros as well.

Then the discretized time-periodic solution will be the fixed point $\bar{\mathbf{y}}$ of the map (22). Exploiting that $\mathbf{y}_p = \mathbf{y}_0 = \bar{\mathbf{y}}$ the fix point is given by

$$\bar{\mathbf{y}} = (\hat{\Phi}_L - \hat{\Phi}_R)^{-1} \mathbf{V}. \quad (23)$$

The stability of this fixed point can be determined based on the spectral radius $\hat{\Phi}_L^{-1} \hat{\Phi}_R$, as described previously.

The construction of the matrices $\hat{\Phi}_L$ and $\hat{\Phi}_R$, along with the extended excitation vector \mathbf{V} , involves a number of non-zero elements directly proportional to the resolution parameter p . This structural property implies not only linear CPU time for their generation but also linearly proportional memory usage when employing sparse matrix implementations. In contrast, the traditional monodromy matrix $\hat{\Phi}$ typically evolves into a dense, almost full upper-triangular structure as it accumulates the effects of p recursive multiplications. Consequently, the memory requirement for the traditional approach scales quadratically ($\mathcal{O}(p^2)$). While this quadratic consumption is often manageable for standard engineering resolutions, it becomes a definitive bottleneck in high-accuracy scenarios. The MFSD method remains superior in these regimes, as its linear memory scaling enables the exploration of extremely high resolutions (e.g., $p = 10^6$) that are practically impossible to test or store using traditional dense-matrix formulations.

Moreover, due to the special banded structure of these matrices, we anticipate that computing both the dominant Floquet multiplier and the periodic solution can be done with linear complexity when iterative methods such as Krylov subspace methods or Arnoldi iterations are utilised. These methods are particularly efficient for large, sparse matrices, as they construct solutions within Krylov subspaces, which reduces computational costs. These assumptions are substantiated by the numerical case studies presented in the subsequent section.

Numerical case studies

The introduced Multiplication Free Semi-Discretization Method (MFSD) has been added to the SemiDiscretizationMethod.jl package [†], which is an efficient implementation of the Semi-Discretization method Insperger and Stépán (2011) in Julia Bezanson et al. (2017). This package allows stability and periodic solution computations on time-periodic dynamical systems with multiple time delays and subjected to external forcing, even using higher-order semi-discretisation methods. In the following sections, the computations were executed using these implementations, thus minimising the number of confounding factors when investigating algorithm performance and complexity.

[†]<https://github.com/HTSykora/SemiDiscretizationMethod.jl>

Note, we will analyse the CPU time only. Since the MFSD method is an exact algebraic rearrangement of the traditional SD mapping, the two methods are mathematically equivalent and MFSD introduces no new discretization errors. The numerical output of the built-in eigenvalue calculation (eigs with the default parameters) provides a numerically equivalent spectral radius. Throughout all the numerical results, the difference is bounded by floating-point round-off errors, i.e., $|\mu_{SD} - \mu_{MFSD}| < 3 \times 10^{-15}$ for any parameter combination. Note that the convergence analysis for different order semi-discretizations is out of scope here, as it is discussed in detail in Insperger and Stépán (2011).

Delayed Mathieu Equation with Forcing

Common test cases for the validation of numerical methods for time-periodic DDEs are variations of the delayed Mathieu equation Kovacic et al. (2018). Therefore, this section is dedicated to a variation of this model: the Mathieu Equation with a delayed state and subjected to external periodic forcing, which has the form of

$$\ddot{x}(t) + 2\zeta\dot{x}(t) + (\delta + \epsilon \cos(2\pi t/T))x(t) = bx(t - \tau) + \cos(8\pi t/T), \quad (24)$$

where the time period of the parametric excitation is T .

We compare the MFSD method to the highly-optimized, open-source implementation of the traditional Semi-Discretization, which performs the successive matrix multiplication (Eq.(7)) based on sparse matrices and calculates the largest eigenvalues and the fixed point. The time measurements are based on the BenchmarkTools.jl, which performs multiple evaluations and provides the median and the standard deviation of the CPU time. The benchmark tests are stopped after 1000 evaluations or after 2 seconds, whichever comes first.

The variation of the measured evaluation times are also recorded, and are presented using the standard deviation with coloured areas for the cases where a sufficient number (>10) of evaluations was available. While the relative timing trends are of primary interest, for reproducibility, we specify the hardware configuration used in the tests: an Intel(R) Xeon(R) Gold 6154 CPU @ 3.00 GHz (36 cores) with 192 GB of RAM. All computations were performed using the Julia programming language and the SemiDiscretizationMethod.jl package. To numerically approximate the time complexity, a power function is fitted to the data for $p > 100$. In most cases, the fitted coefficients (the order of time complexity) are close to integer values; the small difference is related to the influence of the small p values and the fluctuation of the CPU time measurements.

In Fig. 4a, we plot the CPU time as a function of the discretization resolution p for each step listed in Section Time complexity of the semi-discretization method (note: $p \sim 1/\Delta t$). As discussed previously, the construction of all the submatrices \mathbf{P}_i and \mathbf{R}_i , $i =$

$1, 2, \dots, p$ has linear-time complexity $\mathcal{O}(p^1)$, see the thin bright blue curve labelled as “SD- $\mathbf{P}_i, \mathbf{R}_i - \mathcal{O}(p^{1.03})$ ”. The implementation of the \mathbf{C}_i step matrix calculation is sub-optimal because each of them is stored separately, leading to quadratic time complexity $\mathcal{O}(p^2)$ (bright, thin red curve labelled as “SD- $\mathbf{C}_i - \mathcal{O}(p^{1.96})$ ”). This part is probably not optimised further in the original code because it is still one order of magnitude faster than the creation of the monodromy matrix Φ which has an almost cubic time complexity $\mathcal{O}(p^{2.79})$ (bright, thin green curve labelled as “SD- $\Phi - \mathcal{O}(p^{2.79})$ ”) due to the dense structure of Φ . The last step of the spectral analysis is the calculation of the largest eigenvalue (μ_1), which is also much faster and has approximately quadratic time complexity (bright, thin magenta curve labelled as “SD- $\mu_1 - \mathcal{O}(p^{1.69})$ ”). The CPU time of the fix point calculation is also tested: it has approximately quadratic time complexity $\mathcal{O}(p^2)$ (bright, thin pink curve labelled as “SD- $\bar{y} - \mathcal{O}(p^{2.08})$ ”). Figure 4a confirms the findings of Section Time complexity of the semi-discretization method, namely, that the bottleneck of the traditional semi-discretisation method is the computation of the monodromy matrix with its complexity of $\sim \mathcal{O}(p^{2.8})$.

The CPU time of the MFSD method has four components. The first is identical to the computation of the \mathbf{P}_i and \mathbf{R}_i , which has linear-time complexity (thin bright blue curve labelled as “SD- $\mathbf{P}_i, \mathbf{R}_i - \mathcal{O}(p^{1.03})$ ”, the same as in case of semi-discretisation). The second is the construction of the Φ_L and Φ_R matrices. In this process, we create a sparse matrix from the previously computed submatrices and the identity matrices, which also have linear-time complexity (thick brown curve labelled as “MFSD- $\Phi_L, \Phi_R - \mathcal{O}(p^{1.03})$ ”). The final (third) component of the stability analysis is the eigenvalue computation: in this case, the computation of the largest multiplier is faster than for the typically full upper triangular matrix Φ for SD, as it has linear-time complexity for the very sparse banded-structure of the Φ_L and Φ_R matrices (see the thick green curve labelled as “MFSD- $\mu_1 - \mathcal{O}(p^{0.91})$ ”). The computation complexity of fixed point calculation is also reduced to linear based on the numerical analysis (see the thick bright brown curve labelled as “MFSD- $\bar{y} - \mathcal{O}(p^{1.01})$ ”).

Fig. 4a confirms that all the steps of the MFSD method have linear time complexity $\mathcal{O}(p^1)$, and even for small p values, the computation time is similar to the original implementation. The figure also shows that we had to stop the CPU time measurement after around $p \approx 2000$ for the original implementation of the SD method due to the computational demand. Still, in the case of the MFSD method, the CPU time measurements could be continued even for $p = 10^6$.

Figs. 4b and 4c compare the computation times for different T/τ ratios (10 and 0.1, respectively) to analyse the cases $p > r$ and $p < r$ discussed in Subsections Partitioning of the equations for $p - 1 > r$ and Partitioning of the equations for $p - 1 < r$. The MFSD maintains linear time complexity, similarly to the case $T = \tau$ presented in Fig. 4a because the number of \mathbf{P}_i and \mathbf{R}_i matrices is the same for a

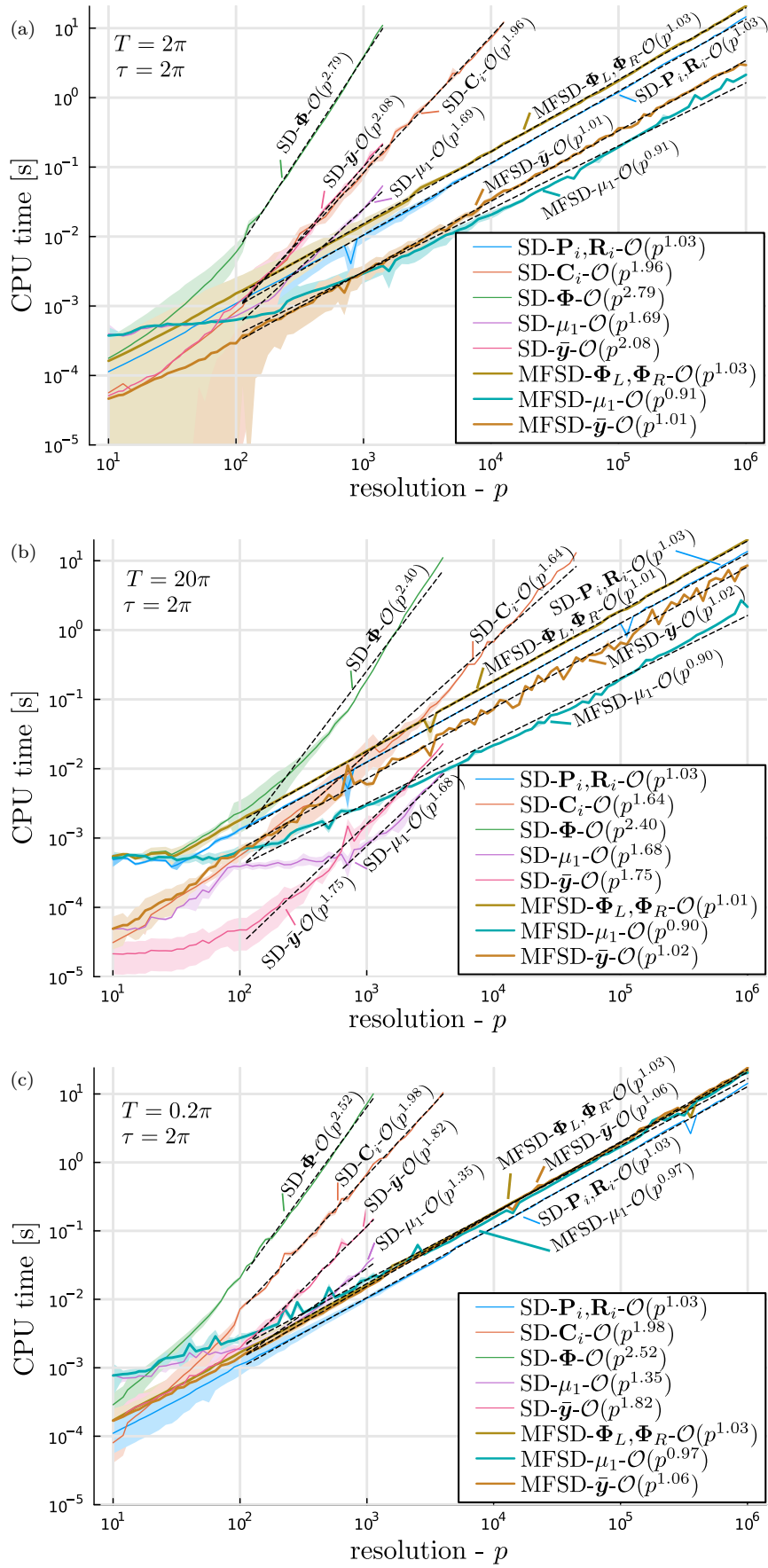


Figure 4. Computation time of the traditional Semi-Discretization (SD) Method (thin lines), and the Multiplication Free Semi-Discretization (MFSD) method (thick lines). The lines denote the mean values, and the coloured area denotes the corresponding ± 1 standard deviation. Thin dashed black lines show the fitted curve. The parameters used during the tests are $\zeta = 0.2$, $\delta = 3$, $\epsilon = 3$, $b = -0.5$, $\tau = 2\pi$,

selected p value. The time complexity of the traditional SD method is also basically the same, with the worst being the time complexity $\mathcal{O}(p^{2.4})$ - $\mathcal{O}(p^{2.8})$ of computing the monodromy matrix Φ . However, there is a slight decrease in the complexity of some computation components due to the different density and non-zero element distribution of the Φ matrix.

Further numerical examples for different system sizes (different values of d) and multiple delays are provided in the Appendix. All cases exhibit similar trends in the graphs; the different CPU time relative to Figs. 4a, 4b and 4c is attributed to the different number of nonzero elements in the system matrices.

All numerical tests support the fact that the Multiplication Free Semi-Discretization has a shorter CPU time for every p parameter and has linear time complexity, leading to orders of magnitude faster computations for scenarios where accurate results are critical.

Conclusion

The development and analysis of the Multiplication Free Semi-Discretization Method (MFSD) presented in this study marks a significant advancement in numerical methods for delayed differential equations. The MFSD method is integrated into the existing open-source `SemiDiscretization.jl` Julia package, allowing the demonstration of the method's superior computational efficiency through the excited Mathieu Equation. This study is driven by the hypothesis that it is possible to achieve linear time complexity in the numerical treatment of delayed dynamical systems, and the implemented MFSD showcases linear time complexity across different scenarios, including spectral analysis and calculation of time-periodic solution for systems with external excitation, making it a potent tool for handling complex dynamical systems with delay elements.

As the method maintains results in mapping matrices Φ_L and Φ_R with well-defined banded structures, it significantly reduces the computational time and resources necessary for the corresponding eigenvalue computations. This effect is especially prominent in cases where the traditional semi-discretization method exhibits quadratic or cubic time complexity.

The open-source implementation of the MFSD opens the way for the research community to make detailed calculations, where high resolution of the time period is essential to capture a system's stability and steady-state behaviour accurately. This capability is particularly valuable in applications such as:

- Machine tool vibrations in milling processes: In milling operations, the regenerative effect introduces delays due to the periodic engagement of cutting teeth, leading to complex dynamics that can cause chatter. Accurate modeling of the stability and periodic motions of these systems is crucial for predicting and mitigating such instabilities Kiss et al. (2022); Sanz-Calle et al. (2024); Ding et al. (2011).
- Networked control systems: In modern control systems, especially those distributed over networks, communication delays are inherent. These delays can significantly affect system stability and performance. Modelling and analysing these systems as time-delay systems help in designing robust controllers that can handle such delays effectively Pan and Peng (2024); Sun and Li (2017).
- The semi-discretization method continues to be actively utilized as a robust tool for stability analysis in complex recent mechanical engineering applications Kumar et al. (2025)
- Biological systems: delays often represent gestation or maturation periods. For example, in population dynamics, the reproduction rate at a given time may depend on the population size at an earlier time. Additionally, these models can exhibit time-periodic behaviour as a result of environmental cycles such as day/night rhythms or seasonal changes. Modeling these systems with delay differential equations allows for a more accurate representation of the biological processes involved Smith (2011); Arino et al. (2007).
- Traffic systems: delays are inherent due to driver reaction times, which can lead to complex dynamics such as stop-and-go waves and phantom traffic jams. These phenomena have been effectively modelled using DDES Molnar and Orosz (2024); Martinovich et al. (2025). Moreover, traffic systems often exhibit time-periodic behaviour due to factors such as traffic light cycles and daily commuting patterns. Modelling these periodicities is crucial for understanding and optimising traffic flow Jiang¹ et al. (2024); Bartfai et al. (2024).

These examples illustrate the broad applicability of the MFSD method in accurately capturing the dynamics of systems where delays and periodicity play a critical role.

In future work, the MFSD method will be extended to distributed delays; however, the linear time complexity of the MFSD method is expected to degrade. When addressing systems with distributed delays, the number of elements - and consequently the computational time - can increase quadratically with the discretisation parameter p . However, in the case of distributed delays, the traditional implementation is also expected to exhibit a similar degradation in computational complexity, transitioning to a higher time complexity (e.g., from $\mathcal{O}(p^{2.8})$ to $\mathcal{O}(p^{3.8})$). However, a detailed analysis of this behaviour is beyond the scope of the present work and is suggested as a direction for future research.

While the current formulation of the MFSD method is tailored for retarded delay differential equations, extending its applicability to neutral and advanced types — based on the work presented in Bartfai et al. (2022) — is a promising direction for future research. This extension is compatible with the proposed MFSD method, so implementing it in the `SemiDiscretization.jl` package would be a straightforward step. However,

integrating this approach into the Julia framework would necessitate additional inputs and structural modifications within the package, which would require considerable implementation effort. Therefore, this task is left for a future project.

The MSFD approach has the potential to generalise well to the stochastic semi-discretisation method Sykora and Bachrathy (2020), for which the actual quartic time complexity ($\mathcal{O}(p^4)$) could be reduced to quadratic ($\mathcal{O}(p^2)$). In stochastic cases, handling the matrix \mathbf{D}_i is crucial (see Eqs. (9) and (10)); based on the further refinement of the MFSD, the constant improvement of $(c/d)^2$ could greatly improve the computational time. However, these method extensions are beyond this paper's scope.

Furthermore, it is important to emphasise that the steps of the MFSD method are similar for the higher-order variations of the semi-discretisation method, therefore resulting in similar performance.

Declaration of Conflicting Interests

The author declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The research leading to these results was supported by the Hungarian Scientific Research Fund (NKFIH 138500 and NKFIH-152125).

Acknowledgements

We thank Henrik Sykora for creating the original implementation of the SemiDiscretization.jl package and for allowing me to access and enhance it.

Appendix

This appendix presents three practical test cases from the literature, of increasing complexity, on which timing results are reported.

Appendix A introduces a minimal population model with seasonal effects: a first-order DDE featuring constant maturation delay, time-periodic mortality and recruitment rates, and external forcing.

The second example, in Appendix B, involves a two-degree-of-freedom turning system with two distinct time-periodic delays, inspired by a practical application: a parallel cutting configuration combined with spindle-speed variation.

The third example, in Appendix C, models the longitudinal vibration of a beam with delayed boundary feedback, discretised using finite element methods. This case represents a high-dimensional system with constant delay and time-periodic excitation.

Note that the aim of the appendix is not to derive detailed models or analyse system behaviour in depth, but rather to present CPU time results across models of varying size and structure using the

proposed MFSD method. We refer readers to the cited literature for complete model formulations and theoretical background.

Appendix A – Population model with seasonal effects

In dynamical population models, reproduction time is often represented as a time delay (see predator–prey models in Ruan (2009); Martin and Ruan (2001)).

We consider the simplest single-species biological model as a scalar DDE with time-periodic coefficients, constant explicit delay, and external seasonal forcing:

$$\dot{N}(t) = -\mu(t)N(t) + \beta(t)N(t - \tau) + A(1 + \cos(\omega t)), \quad (25)$$

where:

- $\mu(t) = \mu_0 + \mu_1 \cos(\omega t)$ is the time-varying mortality rate,
- $\beta(t) = \beta_0 + \beta_1 \cos(\omega t)$ is the time-varying recruitment rate,
- $\tau > 0$ is the maturation delay,
- $A(1 + \cos(\omega t))$ represents seasonal change of the resources.

This linear scalar DDE serves as the minimal model capturing delay, periodic coefficients, and external periodic input.

This model has dimension $d = 1$ and contains a single constant delay. We analyze it numerically using the proposed MFSD method. The numerical results in Fig. 5 show that the traditional SD method exhibits approximately cubic time complexity in forming the monodromy matrix Φ , while the MFSD method achieves linear time complexity in all steps.

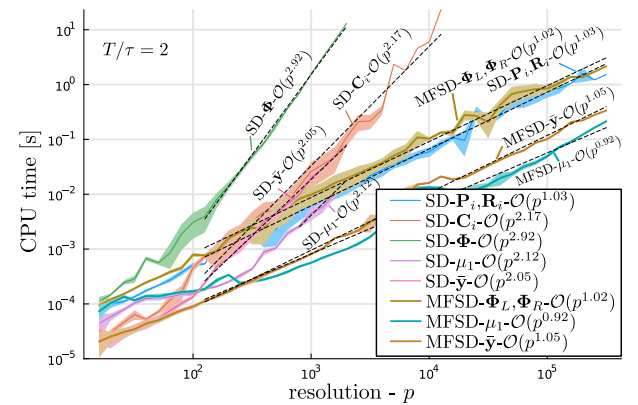


Figure 5. Numerical results for the single-species biological model. Computation time of the traditional Semi-Discretization (SD) Method (thin lines), and the Multiplication Free Semi-Discretization (MFSD) method (thick lines). The lines denote the mean values, and the coloured area denotes the corresponding ± 1 standard deviation. Thin dashed black lines show the fitted curve. The physical parameters used in the tests are: $\mu_0 = 1.2$, $\mu_1 = 0.2$, $\beta_0 = 1.0$, $\beta_1 = 0.3$, $\tau = 0.5$, $A = 0.1$, $\omega = 2\pi$.

Appendix B - Multi-cutter turning with spindle speed variation

As an additional demonstration case, we consider a two-cutter parallel turning system, inspired by the model introduced and measured in Reith et al. (2017, 2016). The dynamic behaviour of the cutting process is captured by a two-degree-of-freedom model, in which both tools are assumed to vibrate in the feed direction. The model incorporates regenerative effects due to the time-delayed interaction between the tool positions and the previously cut surface.

The equation of motion for the system can be written as (for details see Reith et al. (2017)):

$$\mathbf{M}\ddot{\mathbf{x}}(t) + \mathbf{C}\dot{\mathbf{x}}(t) + \mathbf{K}\mathbf{x}(t) = k_w \begin{bmatrix} v_f \tau_1 + x_2(t - \tau_1) - x_1(t) \\ v_f \tau_2 + x_1(t - \tau_2) - x_2(t) \end{bmatrix} \quad (26)$$

where $\mathbf{x} = [x_1, x_2]^T$ denotes the displacements of the two tools, and \mathbf{M} , \mathbf{C} , and \mathbf{K} are the diagonal mass, damping, and stiffness matrices, respectively. The delays τ_1 and τ_2 represent the time required for the surface cut by one tool to reach the position of the other tool.

To introduce time-periodic excitation, we modulate the delay using spindle-speed-variation (SSV) technique Insperger and Stepan (2004); Albertelli et al. (2012); Otto and Radons (2013):

$$\tau_1(t) = \delta \tau_0 (1 + \text{RVA} \cos(\text{RVF} \Omega t)), \quad (27)$$

$$\tau_2(t) = (1 - \delta) \tau_0 (1 + \text{RVA} \cos(\text{RVF} \Omega t)), \quad (28)$$

where Ω is the nominal spindle speed, $\tau_0 = 2\pi/2\Omega$, and RVA, RVF are the relative amplitude and frequency of the speed variation and δ is the delay ratio created by an asymmetric positioning of the tools.

This leads to a time-periodic delay differential equation of Mathieu-type with external parametric excitation through the modulated delay term. The resulting first-order form has dimension $d = 4$ and contains two periodically changing delays. The numerical results in Fig. 6 show that the SD method still has approximately cubic time-complexity in the generation of the monodromy matrix Φ , while the proposed MFSD method has linear time complexity in all steps.

Appendix C - Longitudinal beam vibration with delayed boundary feedback

As a further demonstration case for a large system, we analyse the longitudinal vibration of an elastic beam subject to delayed boundary feedback (see Fig.7), based on the model presented in Zhang and Stepan (2016). The beam has length L , uniform cross-sectional area A , density ρ , and Young's modulus E . The governing equation is a 1D wave equation with damping:

$$\ddot{u}(x, t) - \eta c^2 \dot{u}''(x, t) - c^2 u''(x, t) = 0, \quad (29)$$

where $c = \sqrt{E/\rho}$ is the wave speed and η is the internal damping coefficient.

The right end of the beam is fixed at $x = L$, while the left side at $x = 0$ is free and subject to a delayed feedback force, which is proportional to the

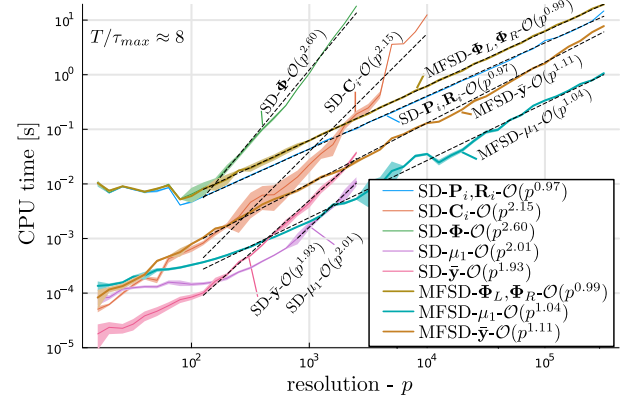


Figure 6. Numerical results for the multi-cutter turning model with SSV. Computation time of the traditional Semi-Discretization (SD) Method (thin lines), and the Multiplication Free Semi-Discretization (MFSD) method (thick lines). The lines denote the mean values, and the coloured area denotes the corresponding ± 1 standard deviation. Thin dashed black lines show the fitted curve. The physical parameters used in the tests are: $m_{diag} = 11$ kg, $k_{diag} = 14.5 \times 10^6$ N/m, $\zeta = 0.0066$, $\omega_n = \sqrt{k/m}$, $c_{diag} = 2m\zeta\omega_n$, $k_w = 237 \times 10^3$ N/m², $v_f = 0.1$ mm/s, $\text{RVA} = 0.05$, $\text{RVF} = 0.25$, $\Omega = 1000$ rpm, $\tau_0 = 2\pi/\Omega$ and delay ratio $\delta = 0.6$.

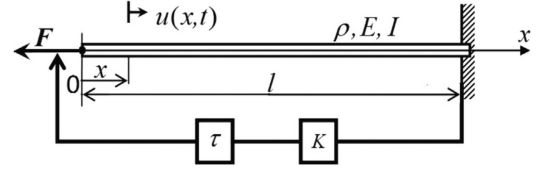


Figure 7. Mechanical model of elastic beam subjected to delayed feedback (copied from Zhang and Stepan (2016))

normal stress resultant measured at the fixed end. The corresponding boundary conditions are

$$u(L, t) = 0, \quad u'(0, t) = K u'(L, t - \tau), \quad (30)$$

where K is the dimensionless feedback gain, and τ is the delay time arising from the control.

We apply a simple finite element discretisation to approximate the continuum model using $N = 15$ linear elements, resulting in the corresponding mass, damping, and stiffness matrices. The dimension of the corresponding first-order systems is $d = 30$. The feedback delay is represented as a fraction of the wave travel time ($T_{\text{prop}} = L/c$) as follows: $\tau = \hat{\tau} T_{\text{prop}}$. We select $\hat{\tau} = 0.2$ in the numerical test case and feedback gain $K = 0.5$.

We use the act-and-wait strategy (see.: Insperger (2006)) with 80% active and 20% inactive control, with time period $T_{a.w.} = 0.4 T_{\text{prop}}$. In addition, a periodic actuator is considered at $x = 0$, applying a harmonic force $F_{\text{exc}}(t) = \cos(2\pi t/T_{a.w.})$.

The CPU time of the SD and MFSD methods are shown in Fig. 8 with a similar tendency as before. It is worth noting that the CPU time increased significantly, as expected, due to the high dimensionality of the system.

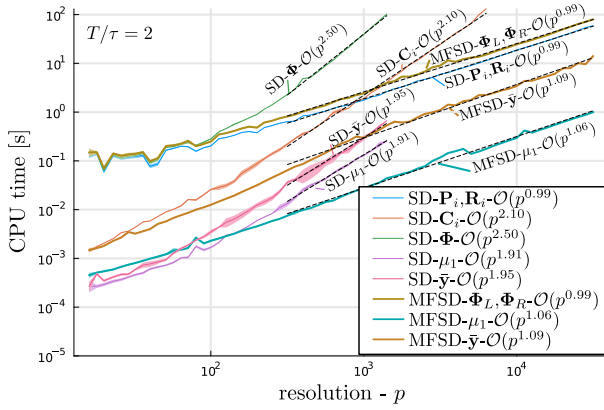


Figure 8. Numerical results for the beam model with Act-and-Wait delay feedback control loop and harmonic excitation. Computation time of the traditional Semi-Discretization (SD) Method (thin lines), and the Multiplication Free Semi-Discretization (MFSD) method (thick lines). The lines denote the mean values, and the coloured area denotes the corresponding ± 1 standard deviation. Thin dashed black lines show the fitted curve. The physical parameters used in the tests are: $E = 210$ GPa, $A = 100$ mm², $\rho = 7800$ kg/m³, $L = 100$ m, $\eta = 0.01$ and $K = 0.5$.

References

- Albertelli P, Musletti S, Leonesio M, Bianchi G and Monno M (2012) Spindle speed variation in turning: technological effectiveness and applicability to real industrial cases. *The International Journal of Advanced Manufacturing Technology* 62(1): 59–67.
- Ali M, Hou Z and Noori M (1998) Stability and performance of feedback control systems with time delays. *Computers and Structures* 66(2): 241–248. DOI:https://doi.org/10.1016/S0045-7949(97)00061-8.
- Arino O, Hbid ML and Dads EA (2007) Delay Differential Equations and Applications: Proceedings of the NATO Advanced Study Institute held in Marrakech, Morocco, 9-21 September 2002, volume 205. Springer Science & Business Media.
- Åström KJ and Murray R (2021) Feedback systems: an introduction for scientists and engineers. Princeton university press.
- Bachrathy D and Stepan G (2013) Improved prediction of stability lobes with extended multi frequency solution. *CIRP Annals* 62(1): 411–414.
- Bartfai A, Barrios A and Dombovari Z (2022) Stability analysis of a one degree-of-freedom robot model with sampled digital acceleration feedback controller in turning. p. V009T09A033.
- Bartfai A and Dombovari Z (2022) Hopf bifurcation calculation in neutral delay differential equations: Nonlinear robotic arms subject to delayed acceleration feedback control. *International Journal of Non-Linear Mechanics* 147: 104239.
- Bartfai A, Voros I and Takacs D (2024) Stability analysis of a digital hierarchical steering controller of autonomous vehicles with multiple time delays. *Journal of Vibration and Control* 30(1-2): 330–341.
- Bayly P, Halley J, Mann BP and Davies M (2003) Stability of interrupted cutting by temporal finite element analysis. *J. Manuf. Sci. Eng.* 125(2): 220–225.
- Bezanson J, Edelman A, Karpinski S and Shah VB (2017) Julia: A Fresh Approach to Numerical Computing. *SIAM Review* 59(1): 65–98. DOI:10.1137/141000671.
- Borgioli F, Hajdu D, Insperger T, Stepan G and Michiels W (2020) Pseudospectral method for assessing stability robustness for linear time-periodic delayed dynamical systems. *International Journal for Numerical Methods in Engineering* 121(16): 3505–3528.
- Breda D, Maset S and Vermiglio R (2005) Pseudospectral differencing methods for characteristic roots of delay differential equations. *SIAM Journal on Scientific Computing* 27(2): 482–495. DOI:10.1137/030601600.
- Butcher EA and Bobrenkov OA (2011) On the chebyshev spectral continuous time approximation for constant and periodic delay differential equations. *Communications in Nonlinear Science and Numerical Simulation* 16(3): 1541–1554. DOI:https://doi.org/10.1016/j.cnsns.2010.05.037.
- Ding Y, Zhu L, Zhang X and Ding H (2011) Milling stability analysis using the spectral method. *Science China Technological Sciences* 54: 3130–3136.
- Habib G, Rega G and Stepan G (2016) Delayed digital position control of a single-dof system and the nonlinear behavior of the act-and-wait controller. *Journal of Vibration and Control* 22(2): 481–495.
- Hamann D and Eberhard P (2018) Stability analysis of milling processes with varying workpiece dynamics. *Multibody System Dynamics* 42: 383–396.
- Henninger C and Eberhard P (2008) Improving the computational efficiency and accuracy of the semi-discretization method for periodic delay-differential equations. *European Journal of Mechanics - A/Solids* 27(6): 975–985. DOI:https://doi.org/10.1016/j.euromechsol.2008.01.006.
- Horvath HZ and Takacs D (2023) Balancing riderless electric scooters at zero speed in the presence of feedback delay. *Multibody System Dynamics*.
- Insperger T (2006) Act-and-wait concept for continuous-time control systems with feedback delay. *IEEE Transactions on Control Systems Technology* 14(5): 974–977.
- Insperger T (2010) Full-discretization and semi-discretization for milling stability prediction: some comments. *International Journal of Machine Tools and Manufacture* 50(7): 658–662.
- Insperger T and Stépán G (2002) Semi-discretization method for delayed systems. *International Journal for numerical methods in engineering* 55(5): 503–518.
- Insperger T and Stepan G (2004) Stability analysis of turning with periodic spindle speed modulation via semidiscretization. *Journal of vibration and control* 10(12): 1835–1855.
- Insperger T and Stépán G (2011) Semi-Discretization for Time-Delay Systems, *Applied Mathematical Sciences*, volume 178. New York, NY: Springer New York. ISBN 978-1-4614-0334-0. DOI:10.1007/978-1-4614-0335-7.
- Insperger T, Stépán G and Milton J (2022) Dynamics of human balancing. In: *Controlling Delayed Dynamics*:

- Advances in Theory, Methods and Applications. Springer, pp. 343–364.
- Insperger T, Stépán G and Turi J (2008) On the higher-order semi-discretizations for periodic delayed systems. *Journal of Sound and Vibration* 313(1-2): 334–341.
- Itovich GR, Gentile FS and Moiola JL (2024) Stability and bifurcations in time-delay systems: A novel and efficient blend of methods. *International Journal of Bifurcation and Chaos* 34(15): 2450195.
- Jiang¹ C, Gan¹ H, Vörös I and Takács D (2024) Safety filter for lane-keeping control. In: 16th International Symposium on Advanced Vehicle Control: Proceedings of AVEC'24–Society of Automotive Engineers of Japan. Springer Nature, p. 371.
- Kiss AK, Hajdu D, Bachrathy D, Stepan G and Dombovari Z (2022) In-process impulse response of milling to identify stability properties by signal processing. *Journal of Sound and Vibration* 527: 116849.
- Kovacic I, Rand R and Mohamed Sah S (2018) Mathieu's Equation and Its Generalizations: Overview of Stability Charts and Their Features. *Applied Mechanics Reviews* 70(2): 020802. DOI:10.1115/1.4039144.
- Kovacs BA and Insperger T (2018) Retarded, neutral and advanced differential equation models for balancing using an accelerometer. *International Journal of Dynamics and Control* 6: 694–706.
- Kumar R, Prasad PD, Mitra RK and Gayen D (2025) Delayed feedback control of pitch oscillations of an alp tower. *International Journal of Mechanical Sciences* 302: 110578.
- Le Gall F (2014) Powers of tensors and fast matrix multiplication. In: Proceedings of the 39th international symposium on symbolic and algebraic computation. pp. 296–303.
- Lehotzky D and Insperger T (2016) A pseudospectral tau approximation for time delay systems and its comparison with other weighted-residual-type methods. *International Journal for Numerical Methods in Engineering* 108(6): 588–613.
- Lehotzky D, Insperger T and Stepan G (2016) Extension of the spectral element method for stability analysis of time-periodic delay-differential equations with multiple and distributed delays. *Communications in Nonlinear Science and Numerical Simulation* 35: 177–189.
- Lehotzky D, Insperger T and Stepan G (2018) Numerical methods for the stability of time-periodic hybrid time-delay systems with applications. *Applied Mathematical Modelling* 57: 142–162. DOI:https://doi.org/10.1016/j.apm.2017.12.029.
- Li L, Gan J, Cui C, Ma H, Qu X, Wang Q and Ran B (2024) Potential field-based modeling and stability analysis of heterogeneous traffic flow. *Applied Mathematical Modelling* 125: 485–508. DOI:https://doi.org/10.1016/j.apm.2023.09.012.
- Lichtner M, Wolfrum M and Yanchuk S (2011) The spectrum of delay differential equations with large delay. *SIAM Journal on Mathematical Analysis* 43(2): 788–802. DOI: 10.1137/090766796.
- Liu C, Tang D, Chen X and Ding G (2024) An efficient and precise stability analysis method for milling process. *The International Journal of Advanced Manufacturing Technology* : 1–16.
- Martin A and Ruan S (2001) Predator-prey models with delay and prey harvesting. *Journal of Mathematical Biology* 43: 247–267.
- Martinovich K, Stepan G, Bachrathy D and Kiss KA (2025) Introducing state-dependent delay in the car-following model. *Nonlinear Dynamics* .
- Mi T, Takács D, Liu H and Orosz G (2024) Capturing the true bounding boxes: vehicle kinematic data extraction using unmanned aerial vehicles. *Journal of Intelligent Transportation Systems* : 1–13.
- Molnar TG and Orosz G (2024) Destroying phantom jams with connectivity and automation: Nonlinear dynamics and control of mixed traffic. *Transportation Science* 58(6): 1319–1334.
- Otto A and Radons G (2013) Application of spindle speed variation for chatter suppression in turning. *CIRP Journal of Manufacturing Science and Technology* 6(2): 102–109.
- Ozoeugu CG (2018) A general order full-discretization algorithm for chatter avoidance in milling. *Advances in Mechanical Engineering* 10(7): 1687814018773811.
- Ozturk E and Budak E (2008) Chatter stability of 5-axis milling using multi-frequency solution. In: Proceedings of the 3rd International Conference on High Performance Cutting. pp. 429–444.
- Pan D and Peng D (2024) Finite region stability and networked predictive control for 2-d nonlinear time-delayed systems with quantization, packet dropouts and random disturbances. *Systems & Control Letters* 188: 105801. DOI:https://doi.org/10.1016/j.sysconle.2024.105801. URL https://www.sciencedirect.com/science/article/pii/S0167691124000896.
- Reith MJ, Bachrathy D and Stepan G (2016) Improving the stability of multi-cutter turning with detuned dynamics. *Machining Science and Technology* 20(3): 440–459.
- Reith MJ, Bachrathy D and Stepan G (2017) Optimal detuning of a parallel turning system—theory and experiments. *Journal of Dynamic Systems, Measurement, and Control* 139(1): 014503.
- Ruan S (2009) On nonlinear dynamics of predator-prey models with discrete delay. *Mathematical Modelling of Natural Phenomena* 4(2): 140–188.
- Saad Y (2011) Numerical methods for large eigenvalue problems: revised edition. SIAM.
- Sanz-Calle M, Munoa J, Morelli L, Iglesias A, de Lacalle LNL and Dombovari Z (2024) On the effect of radial engagement on the milling stability of modes perpendicular to the feed direction. *CIRP Journal of Manufacturing Science and Technology* 49: 111–127.
- Smith H (2011) An introduction to delay differential equations with applications to the life sciences. Springer.
- Stepan G (2009) Delay effects in the human sensory system during balancing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 367(1891): 1195–1212.
- Strassen V (1969) Gaussian elimination is not optimal. *Numerische mathematik* 13(4): 354–356.

- Sun X and Li G (2017) Synchronization transitions induced by partial time delay in a excitatory–inhibitory coupled neuronal network. *Nonlinear Dynamics* 89(4): 2509–2520.
- Sykora HT and Bachrathy D (2020) Stochastic semidiscretization method: Second moment stability analysis of linear stochastic periodic dynamical systems with delays. *Applied Mathematical Modelling* 88: 933–950. DOI:<https://doi.org/10.1016/j.apm.2020.06.078>.
- Szekrényes A (2023) Stability of delaminated composite beams subjected to retarded periodic follower force. *Archive of Applied Mechanics* 93(11): 4197–4216.
- Szekrényes A (2025) Mechanics of delaminated composite beams subjected to retarded follower force with multiple time delay. *Acta Mechanica* 236(2): 655–672.
- Toth M, Bachrathy D and Stepan G (2017) Effect of wavy tool path on the stability properties of milling by the implicit subspace iteration method. *The International Journal of Advanced Manufacturing Technology* 91: 1781–1789.
- Vyasarayani C, Subhash S and Kalmár-Nagy T (2014) Spectral approximations for characteristic roots of delay differential equations. *International Journal of Dynamics and Control* 2: 126–132.
- Yang Y, Yuan JW, Tie D, Wan M and Zhang WH (2023) An efficient and accurate chatter prediction method of milling processes with a transition matrix reduction scheme. *Mechanical Systems and Signal Processing* 182: 109535.
- Zatarain M, Alvarez J, Bediaga I, Munoa J and Dombovari Z (2015) Implicit subspace iteration as an efficient method to compute milling stability lobe diagrams. *The International Journal of Advanced Manufacturing Technology* 77: 597–607.
- Zatarain M and Dombovari Z (2014) Stability analysis of milling with irregular pitch tools by the implicit subspace iteration method. *International Journal of Dynamics and Control* 2: 26–34.
- Zhang G and Xi G (2022) Vibration control of a time-delayed rotor-active magnetic bearing system by time-varying stiffness. *International Journal of Applied Mechanics* 14(04): 2250007.
- Zhang L and Stepan G (2016) Exact stability chart of an elastic beam subjected to delayed feedback. *Journal of Sound and Vibration* 367: 219–232. DOI:<https://doi.org/10.1016/j.jsv.2016.01.002>. URL <https://www.sciencedirect.com/science/article/pii/S0022460X16000304>.