BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

PHD THESIS

Brno, 2023 Ing. Petr Veigend

BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

HIGH ORDER NUMERICAL METHOD IN MODELLING AND CONTROL SYSTEMS

NUMERICKÁ METODA VYŠŠÍHO ŘÁDU V MODELOVÁNÍ A ŘÍZENÍ

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR **Ing. PETR VEIGEND**
AUTOR PRÁCE

SUPERVISOR **Ing. VÁCLAV ŠÁTEK, Ph.D.**
ŠKOLITEL

BRNO 2023

Abstract

Control systems are widely used as they enable precise management and regulation of complex processes across various industries. Ordinary differential equations are widely used in control theory because they provide a mathematical framework to describe the dynamic behaviour of control systems. They allow stability analysis, have good performance characteristics and can effectively regulate and optimise systems responses in real-time. The high-order numerical methods are not often used in the real-time context because of the large number of operations.

The thesis deals with the numerical solution of ordinary differential equations using a higher-order variable-step variable-order numerical method based on the Taylor series. The method is defined for linear and non-linear problems, and several optimisations to increase its performance are introduced. The positive properties of the method are thoroughly analysed and demonstrated on a set of real-world technical problems.

The results show that the Taylor-series-based method can be used in the area of control and regulation and outperforms the state-of-the-art methods in terms of speed and accuracy of the calculation.

Abstrakt

Systémy pro řízení a regulaci jsou používány téměř ve všech průmyslových oblastech. Pro jejich modelování se často používají diferenciální rovnice, které popisují dynamické chování těchto systémů a umožňují je detailněji analyzovat z hlediska přesnosti, stability, výkonu a reakcí těchto systémů v reálném čase. V této oblasti se běžně nepoužívají metody vyšších řádů, protože vykonávají velké množství operací.

Tato práce zkoumá numerické řešení obyčejných diferenciálních rovnic s použitím metody s proměnným řádem a proměnnou velikostí kroku, která je založena na Taylorově řadě. Metoda je navržena jak pro lineární, tak pro nelineární problémy a jsou implementovány její optimalizace pro snížení výpočetního času bez degradace jejích vlastností. Pozitivní vlastnosti metody jsou demonstrovány na sadě příkladů z technické praxe.

Výsledky práce ukazují, že metoda založena na Taylorově řadě může být použita v oblasti řízení a regulace a má lepší vlastnosti než běžně používané metody.

Keywords

ordinary differential equations, higher-order numerical methods, Taylor series, technical initial value problems, control, regulation, modelling, controllers, regulators

Klíčová slova

obyčejné diferenciální rovnice, numerické metody vyššího řádu, Taylorova řada, technické počáteční úlohy, řízení, modelování, regulátory

Reference

VEIGEND, Petr. *High order numerical method in modelling and control systems.* Brno, 2023. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Václav Šátek, Ph.D.

Rozšířený abstrakt

Tato práce se zabývá modelováním a simulací reálných problémů pomocí numerické integrační metody vyššího řádu, která je založena na Taylorově řadě. Tato metoda má velké množství pozitivních vlastností (např. automatické nastavení počtu členů Taylorovy řady, které se používají pro výpočet, vysokou numerickou stabilitu, vysokou přesnost a rychlost výpočtu), které lze využít při simulaci systémů, které jsou namodelovány pomocí obyčejných diferenciálních rovnic prvního řádu s počáteční podmínkou – technické počáteční problémy.

V rámci práce jsou nejprve představeny některé běžně používané explicitní numerické metody pro řešení obyčejných diferenciálních rovnic (např. metoda Eulerova, metody Runge-Kutta, apod.) a jejich vlastnosti. Metoda vyššího řádu založena na Taylorově řadě (se zkratkou MTSM – Moderní metoda Taylorovy řady) je definována jak pro lineární, tak pro nelineární problémy včetně detailního popisu možných optimalizací, hlavně pro velmi výpočetně náročný nelineární výpočet vyšších členů Taylorovy řady. Pozitivní vlastnosti metody jsou ověřeny na sadě příkladů z různých oblastní technické praxe (astronomie, elektrotechnika, fyzika, atp.). V drtivé většině případů metoda řeší definované problémy rychleji a se srovnatelnou přesností, než běžně používané numerické metody. Práce se zabývá aplikací metody v oblasti řízení a regulace, kde je opět na sadě příkladů demonstrováno chování metody při řešení problémů z této oblasti, včetně možných aplikací.

Práce ukazuje vhodnost použití MTSM v technické praxi a v oblasti řízení a regulace a ukazuje na další možné výzkumné směry (např. další optimalizace pro výpočet nelineárních problémů nebo aplikace metody ve skutečném systému).

High order numerical method in modelling and control systems

Declaration

I hereby declare that this thesis was prepared as an original work by the author under the supervision of Ing. Václav Šátek, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

. . . .
Petr Veigend
May 30, 2023

Acknowledgements

If I may, I would love to start with the dedication. When I first started writing, I had another one in mind, however, life intervened. I would like to dedicate this thesis to my grandfather Ladislav, who passed away in 2021 and my grandmother Jiřina, who passed away in 2022. Without them both, working and doing anything became that much harder. We were together a lot. My grandfather raised me and gave me values and outlook. My grandmother showed me how to be kind and decent human being. I miss them both very much. And I hope that if they are somehow watching and reading this, they are both proud of what I have accomplished. And because neither of them knew any English, allow me to very briefly switch to Czech:

> Babi, dědo, moc vám oběma děkuji za všechno, co jste pro mě kdy udělali. Bez vás by tato práce nemohla nikdy vzniknout. Hrozně nám s mamkou chybíte.

So, that is it for the dedication. Now, on with thanks. First, I would like to thank Jiří Kunovský. Even though he is not here with us any more, his presence is still felt. I miss him dearly as my friend and supervisor. I hope that he would be proud of this work. I would love to thank Vašek Šátek, my supervisor and friend that has helped guide me with the thesis. I hope that we are going to keep working together after the thesis is done.

I would like to thank my family for their support and love over the (many) years. I would not be able to finish this work without their help. I would love to thank Gabriela (*GN*), who has helped me with many, many things during the preparation of this thesis, including giving me the most valuable advice and keeping me motivated and smiling. I would like to thank my boss and dear friend Jarek, who has supported me and tried to find as much time as possible for me to actually work on the thesis in the hectic few years that we had on the faculty and in the Czech Republic. I would like to thank Kristýna, who, along with other study advisors, took so much work off my shoulders that it is honestly not even ok. And last but certainly not least, I have to thank Sylva and Svatka, my guardian angels and just super amazing people that just help me handle so many things on a daily basis with just a kind word or a funny message.

To name everyone that helped, supported or encouraged me on the way would probably take the rest of this thesis, so I'm going to stop here. I'm really sorry to everyone I did not mention by name (the amazing neighbours from the security group, friends from our department ...) and to all my friends from "the outside world". I have to make several exceptions. The first is probably my best friend Jirka. I can't wait to meet his new family. My friend Martin works so hard and is so incredibly unlucky, and I hope that his luck turns soon. I have to mention Bethaney and Hayley. Thank you both so much for being there for me. Their help and encouragement have helped me with this thesis immensely.

And lastly, I would like to thank all projects related to the thesis that I have participated in. Namely the internal BUT FIT projects FIT-S-14-2486[1], FIT-S-17-4014[2] and FIT-S-20-6427 [3] and international project Aktion-76p11[4].

[1]https://www.fit.vut.cz/research/project/750/.en

[2]https://www.fit.vut.cz/research/project/1132/.en

[3]https://www.fit.vut.cz/research/project/1421/.en

[4]https://www.fit.vut.cz/research/project/1103/.en

Contents

List of Figures

List of Tables

List of Abbreviations

h	Size of the integration step
t_{max}	Maximum time of computation
max_{ORD}	Maximum number of used Taylor series terms during one integration step
ODE	Ordinary Differential Equation
IVP	Initial Value Problem
MTSM	Modern Taylor Series Method
MDOR	Method of Derivative Order Reduction
MDORAV	Method of Derivative Order Reduction with Additional Variable
MSI	Method of Successive Integration

Chapter 1

Introduction

The presented thesis deals with quite an expansive problem – how to apply the high-order numerical integration method into the area of control systems (with a wide range of possible applications, including systems that have hard limits on time of calculation). These control systems working in real-time context are widely used, and if they are not working correctly, in the worst case scenario, lives might be lost. This places a high burden on the used control system. This thesis tries to answer the following question:

> *Is a high-order variable step, variable order numerical method suitable for use in the systems that operate in real-time?*

The posed question seems to have a uniform answer in the literature – (*no*), and that is going to be discussed in more detail. This answer is due to the fact that the number of operations is critical in the real-time context, and these methods do not seem to be well suited for the task.

The thesis deals with state-of-the-art numerical methods that can be used in the real-time context for comparison and shows how the presented high-order method resolves the shortcomings these methods might have. In the thesis, I will discuss how the proposed numerical method and its modifications and optimizations might be used to achieve better results. First generally and then in the context of a control system using a constructed simulation model on a set of experiments that are all cyber-physical and therefore have high usefulness in real-world applications.

1.1 Research objectives

The thesis has several research objectives:

- discuss the currently existing numerical integration methods, particularly in the context of control systems,
- analyse the properties of a high-order integration method based on the Taylor series and evaluate the applicability of this method on a set of technical problems,
- extend the capabilities of high-order Taylor series method to solve non-linear problems, propose and discuss possible optimizations,
- show the suitability of the high-order method to be used as a part of the control system using a set of examples (with strict considerations of time of calculation),

- show the potential for further research and additional improvements of the method for both linear and non-linear problems.

1.2 Structure of the thesis

The thesis is structured so that the general topics create a foundation for more specialized ones. In Chapter 2, the overview of the widely used numerical integration methods is presented with an emphasis on their properties, including stability and usability in real-world usage. Chapter 3 introduces the proposed numerical integration method with its positive properties highlighted compared to the state-of-the-art solvers introduced in the previous Chapter. The definition of the method for linear and non-linear problems is also included with the possible optimizations to the non-linear solver, which are non-trivial and can help to dramatically increase the performance of the calculation. Chapter 4 contains several linear and non-linear real-world example problems from several different areas (physics, electrical engineering, astronomy, etc.) that show how the method behaves when solving different types of problems. After establishing the properties and usability of the method when solving general problems, Chapter 5 overviews the basics of system control theory needed to understand the control systems experiments in the final Chapter. Linear and non-linear controllers are also introduced. Finally, Chapter 6 contains the experiments on a set of control problems that are solved using the proposed numerical method, showing that its properties make it suitable for working with control systems.

Chapter 2

Numerical solution of differential equations

This Chapter discusses the basics of the numerical solution of differential equations. Then, due to this thesis's topic, state-of-the-art methods that can be used in real-time applications are introduced. Analysis and discussion of commonly used numerical methods of several types follow with examples. The chapter is primarily compiled from [18], [32], [17], and other sources cited throughout.

In real-life situations, the differential equation that models the problem is too complicated to solve exactly, and one of two approaches is taken to approximate the solution. The first approach is to modify the problem by simplifying the differential equation to one that can be solved exactly and then use the solution of the simplified equation to approximate the solution to the original problem. The other approach uses methods for approximating the solution of the original problem. This is the approach that is most commonly taken because the approximation methods give more accurate results and realistic error information [17].

The numerical integration methods do not produce a continuous approximation of the solution of the initial-value problems. Instead, approximations are found at specific points, often spaced equally apart. These points form a solution mesh. The way of creating the mesh differs based on a selected method and other considerations.

First, let us briefly discuss how the numerical derivation works on a conceptual level before applying the principles to the numerical solution of the differential equations.

2.1 Numerical differentiation

When having a function defined by a set of values instead of its analytical solution, we need to be able to replace the function with a polynomial representation that would approximate it and create derivatives (or integrate, if need be) of the approximating function. The derivative of the function $y(x)$ can be expressed as

$$y'(x_0) = \lim_{h \to 0} \frac{y(x_0 + h) - y(x_0)}{h} \,. \tag{2.1}$$

The following example shows how to obtain the approximation of the derivative of the function. A function y at the point x_0 can be approximated using the Lagrange polynomial of the original function $y(x)$

$$y(x) = \frac{y(x_0)(x - x_0 - h)}{-h} + \frac{y(x_0 + h)(x - x_0)}{h} + \frac{(x - x_0)(x - x_1)}{2} f''(\xi(x))$$

for $x_0 \in (a, b)$ and $x_1 = x_0 + h$ for some $h \neq 0$ that is sufficiently small to ensure that $x_1 \in \langle a, b \rangle$ and $\xi(x)$ between x_0 and x_1. Differentiating the polynomial gives

$$y'(x) = \frac{y(x_0 + h) - y(x_0)}{h} + \frac{2(x - x_0) - h}{2} y''(\xi(x)) + \frac{(x - x_0)(x - x_0 - h)}{2} D_x(y''(\xi(x)))\,.$$

Deleting the terms involving $\xi(x)$ gives the approximation of the derivative. For $x = x_0$ we can therefore write

$$y'(x) \approx \frac{y(x_0 + h) - y(x_0)}{h}$$

or rather

$$y'(x) = \frac{y(x_0 + h) - y(x_0)}{h} - \frac{h}{2} y''(\xi) \tag{2.2}$$

for $x = x_0$. For small values of h, the quotient $\frac{y(x_0+h)-y(x_0)}{h}$ can be used to approximate the first derivative $y'(x)$. This formula is known as a *forward-difference formula* for $h > 0$ or *backward-difference formula* for $h < 0$ [71]. Figure 2.1 shows the geometric interpretation of the forward difference formula.

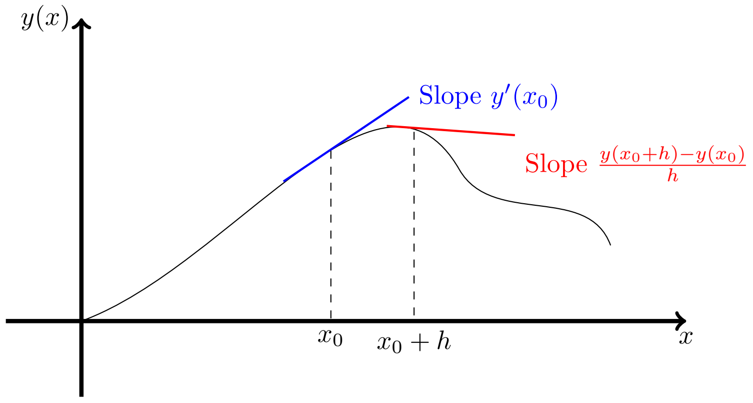


Figure 2.1: Geometric interpretation of the forward difference formula (based on [17]).

If we use more points, the general accuracy increases. However, the number of performed operations and the rise of round-off error makes higher orders than five generally not practical [17].

2.2 Ordinary differential equations

The first-order differential equation can be defined as [18]

$$y'(x) = f(x, y(x))\,, \tag{2.3}$$

where $y'(x) = \frac{\mathrm{d}y}{\mathrm{d}x}$. The differential equation alone does not generally define a unique solution, and a number of conditions have to be added to the problem – boundary or initial conditions (all components of y are specified at the single value of x). If the value $y(x_0) = y_0$ is given, then the equations

$$y'(x) = f(x, y(x)), \qquad y(x_0) = y_0\,, \tag{2.4}$$

are together known as a *initial value problem* (abbreviated as IVP). Note that if the IVP is multidimensional, initial values in all dimensions have to be defined. Therefore, we might write

$$\boldsymbol{y}'(x) = \boldsymbol{f}^T(x, \boldsymbol{y}(x)), \quad \boldsymbol{y}_0 = \left[y_0^0 y_0^1 y_0^2 \cdots y_0^N\right]^T. \tag{2.5}$$

For natural problems, the differential equations might have higher order. The higher-order differential equation is in the form

$$y^{(n)} = \phi\left(x, y, y', y'', \ldots, y^{(n-1)}\right), \tag{2.6}$$

with initial values given for $y(x_0)$, $y'(x_0)$, $y''(x_0)$, $\ldots$, $y^{(n-1)}(x_0)$. The methods that can transform such system into the system of the first-order ODEs are discussed in Section 2.6.

2.2.1 Existence and uniqueness of solutions

A fundamental question when creating a system model is whether a given differential equation with the associated initial condition can be reliably used to predict the system's future state. If the system is acceptable from this point of view, we call it *well-posed.* For the system to be well-posed, the following attributes have to be taken into account.

- Does solution exist?
- If solution exists, is it unique?
- How sensitive is the calculated solution to the small changes in the initial condition?

All criteria mentioned above can be satisfactorily answered by using the *Lipschitz condition.*

Definition 1 *The function $f : [a, b] \times \mathcal{R}^N \to \mathcal{R}^N$ is said to satisfy a Lipschitz condition in its second variable if there exists a constant L, known as Lipschitz constant, such that for any $x \in [a, b]$ and $\boldsymbol{Y}, \boldsymbol{Z} \in \mathcal{R}^N$, $||\boldsymbol{f}(x, \boldsymbol{Y}) - \boldsymbol{f}(x, \boldsymbol{Z})|| \leq L||\boldsymbol{Y} - \boldsymbol{Z}||$.*

Based on the Definition 1, the initial value problem (2.4) can be defined.

Theorem 1 *Consider the initial value problem*

$$\boldsymbol{y}'(x) = \boldsymbol{f}(x, \boldsymbol{y}(x)), \tag{2.7}$$

$$\boldsymbol{y}(a) = \boldsymbol{y}_0, \tag{2.8}$$

where $\boldsymbol{f} : [a, b] \times \mathcal{R}^N \to \mathcal{R}^N$ is continuous in its first variable and satisfies the Lipschitz condition in its second variable. Then unique solution to this problem exists.

The proof of Theorem 1 can be found in [18].

The third requirement for being well-posed is that the solution is not overly sensitive to the initial condition. This can be assessed for problems that satisfy the Lipschitz condition. Further analysis can be found in [18]. Of particular note here are stiff systems, that often have large Lipschitz constants and different approaches to their solution has to be used [83].

2.3 Numerical integration

In this Section, the concept of numerical integration is briefly introduced and several state-of-the-art methods that are commonly used are presented. The Section is mostly compiled from [18], [17] and [32].

The methods presented in this Section are used for comparison with the high-order method based on the Taylor series, which is going to be presented in detail in Chapter 3. The primary focus of the Section is going to be on performance, stability, and usefulness, mainly in real-time applications.

This thesis mainly considers the numerical solution of the ordinary differential equations (ODEs) with generally arbitrary order. The solution of these equations is performed in the time domain so that $x \to t$. The thesis discusses several approaches that can be used to transform the arbitrary order equation to the general form of first-order ODE

$$g(t, y(t), y'(t)) = 0 \tag{2.9}$$

and for the explicit methods (which are mainly considered in the rest of the thesis), the general formulation above can be rewritten as (2.7). The general solution defined by (2.7) contains the integration constant that can have an arbitrary value. To specify the solution of the problem, the initial condition (2.8) for the function in $t = t_0$ can be specified.

Before defining the methods commonly used for numerical integration, the general principle should be discussed first. When solving an IVP using a numerical method [28], we find the *approximate values* of the function $y(t)$ in the defined nodes of the solution mesh $\{t_0, t_1, t_2, \ldots, t_m\}$ in the interval $\langle a, b\rangle$. The nodes of the mesh on the interval are therefore defined as $a = t_0 < t_1 < t_2 < \cdots < t_m = b$. The step size of the mesh can be calculated as $h_{i+1} = t_{i+1} - t_i$, for $i = 0, 1, \ldots, n-1$. The step size can be the same in all areas of the mesh – *equidistant mesh*, or it can change during the calculation – *non-equidistant mesh.*

The basic principle of the numerical solution of differential equations is in Figure 2.2.

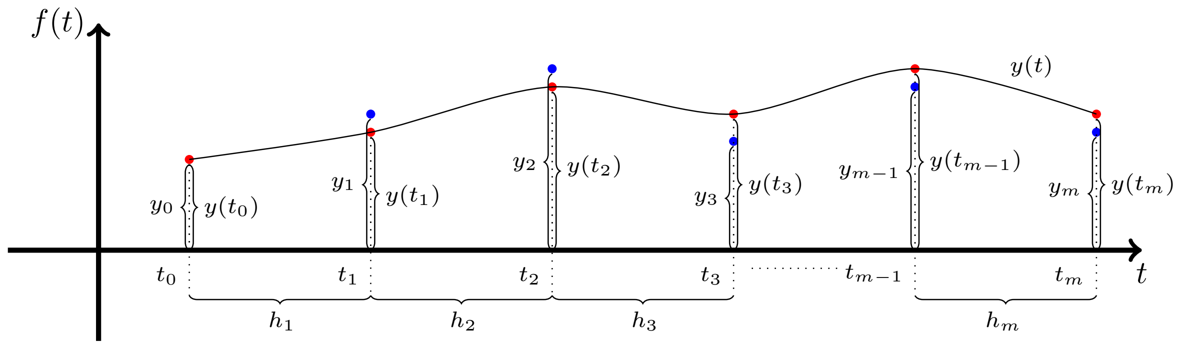


Figure 2.2: The basic principle of IVP solution. The problem is defined by $y' = f(t, y(t)), y(t_0) = y_0$. The curve $y(t)$ represents the accurate (analytical) solution. The blue points represent the solutions in the individual mesh points (based on [28]).

The values in the mesh nodes can be calculated using various numerical integration methods. These methods can be broadly categorized into two major groups:

- *single-step* methods use *the previous mesh point* to calculate the value in the next mesh point,

- *multi-step* methods use *multiple previous mesh points* to calculate the value in the next mesh point.

The basis for all single-step methods is the *Taylor series* [17], [83] and [32] which can be defined using the following Theorem.

Theorem 2 *Let $C^d(X)$ be the set of all functions that have the continuous derivative on X. Suppose $f \in C^d\,[a,b]$, that f^{d+1} exists on $[a,b]$ and $t_0 \in [a,b]$. For every $t \in [a,b]$ there exists a number $\xi(x)$ between t and t_0 with*

$$f(x) = P_N(x) + R_N(x)\,,$$

where

$$\begin{aligned} P_N(x) &= f(t_0) + f'(t)(t-t_0) + \frac{f''(t)}{2!}(t-t_0)^2 + \cdots + \frac{f^{(N)}(t_0)}{N!}(t-t_0)^N \\ &= \sum_{k=0}^{N} \frac{f^{(k)}(t_0)}{k!}(t-t_0)^k \end{aligned}$$

and

$$R_N(x) = \frac{f^{(N+1)}(\xi(x))}{(N+1)!}(t-t_0)^{N+1}\,.$$

The term $P_N(x)$ is called the *nth Taylor polynomial* for f in the neighbourhood of t_0 and $R_N(x)$ is called the *remainder term* (or *truncation error* – the error involved when using the truncated summation of an infinite series) associated with $P_N(x)$. The value of the function $\xi(x)$ cannot be determined explicitly, and Theorem 2 ensures that such function exists and that its value lies between t_0 and t. The infinite series obtained by taking a limit of $P_N(x)$ as $N \to \infty$ is called the *Taylor series* for f in the neighbourhood of t_0. Note that for $t_0 = 0$, the Taylor polynomial is often called the *Maclaurin series.*

All single-step methods are based on the Taylor series. It can provide the best approximation of the function when a sufficient number of the terms are calculated. The biggest problem while using the methods based on the Taylor series is the calculation of higher derivatives. This problem can be solved using the method presented in Chapter 3. Probably the most well-known, simplest and most straightforward single-step method, the *Euler method*, is will be discussed first.

2.3.1 Euler method

The Euler method, first published by Leonhard Euler between 1768 and 1770 and republished in his collected works, is based on a very simple principle that can be generalized to create more advanced methods. Suppose that a particle is moving in such a way that at time t_0, its position is equal to y_0, and at that time, the velocity is equal to v_0. The principle is that in a very short time, the velocity does not change significantly from v_0, and the change in position will be approximately equal to the change in time multiplied by y_0.

If the motion of the particle is driven by a differential equation, the value of v_0 will be known as the function of t_0 and y_0. Given t_0, the solution at t_1, assumed to be close to t_0 can be calculated as

$$y_1 = y_0 + (t_1 - t_0)v_0$$

which can be found from the known values of y_0, t_1 and t_0. The numerical solution of a differential equation is a sequence of approximations at times t_1, t_2, t_3, …, t_m that should lead to the acceptable approximation of the given function.

Of course, the interpretation is much broader than for a single particle. The dependent variable y might not have a meaning of distance and might not even be scalar. If $\boldsymbol{y}$ is a vector, then it can be interpreted as a collection of scalars y_1, y_2, …, y_n. The differential equation and the initial information together determine the values of $\boldsymbol{y}$ components as the time variable changes can be written in the form

$$\boldsymbol{y}'(t) = \boldsymbol{f}(t, \boldsymbol{y}(t)) \qquad \boldsymbol{y}(t_0) = \boldsymbol{y}_0 \,. \tag{2.10}$$

An important special case is that $\boldsymbol{f}$, or for the vector problems f_1, f_2, …, f_n does not depend on the time variable at all. In this case, the problem is *autonomous* and can be written in the form

$$\boldsymbol{y}'(t) = \boldsymbol{f}(\boldsymbol{y}(t)) \qquad \boldsymbol{y}(t_0) = \boldsymbol{y}_0 \,. \tag{2.11}$$

Using the terms of the Taylor series, the explicit method can be written as

$$\boldsymbol{y}_{i+1} = \boldsymbol{y}_i + h\boldsymbol{y}'_i \tag{2.12}$$

note that (2.12) contains the first term of the Taylor series. Therefore, the Euler method is so called *first-order* method.

In MATLAB, the method is not included in the set of standard solvers (due to its limited usability in solving real-world problems and its limited stability). However, understanding the basic principles is helpful because it demonstrates how numerical integration works on the most basic level. More advanced methods (i.e. the Runge-Kutta methods) that are going to be discussed next are more complicated.

2.3.2 Runge-Kutta methods

The methods of this type can generally be viewed as generalizations of the Euler method. The methods are attributed to Runge (1895), with Kutta (1901) and Heun providing further contributions. The methods came back into focus after the advent of modern digital computers, and new methods with higher orders are being developed and tested that fall into this broad category (including the methods included in the MATLAB software package).

The basic principle behind any Runge-Kutta method is simple: rather than calculate the function f just once in each time step, the methods might calculate the function multiple times per time step with different arguments. The number of function evaluations in the individual time steps can determine the order of the method. Further, the methods can be additionally augmented with additional calculations to improve the accuracy and other properties of the methods. For the method with the order p, at least s stages are necessary.

Second-order method

The second-order method (commonly abbreviated as RK2) evaluates the function f two times during each time step. One evaluation is performed at the beginning of an integration step, and the second evaluation can be calculated, for example, in the middle of an integration step (mid-point). This can, however be difficult, but the second value can be approximated using a similar formula as the Euler method. With two approximations completed, all the data are available to calculate the second-order function approximation using the trapezoidal rule formula. The second-order method that uses the mid-point can be formulated as:

$$\begin{aligned} k_1 &= y_{i-1} + hf(t_{i-1}, y_{i-1}) \\ y_i &= y_{i-1} + \frac{h}{2}(f(x_i, k_1) + f(t_{i-1}, y_{i-1}))\,. \end{aligned} \tag{2.13}$$

The construction of the second-order Runge-Kutta method can be generalized by using an arbitrary value for the distance into the current integration step (denoted as ϕ). Using the second-order quadrature formula

$$\int_0^1 \phi(t)\mathrm{d}t \approx (1 - \frac{1}{2\phi})\phi(0) + \frac{1}{2\phi}\phi(0).$$

As stated above, the distance ϕh can be arbitrary, most commonly $\phi = \frac{1}{2}$. The Runge-Kutta methods can also be described using the coefficient tableau with the following form.

$$\begin{array}{c|c} \boldsymbol{c} & \boldsymbol{A} \\ \hline & \boldsymbol{b}^T \end{array}$$

Vector $\boldsymbol{c}$ indicates the positions of the stage values within the step, matrix $\boldsymbol{A}$ indicates the dependence of the stages on the derivatives found at other stages, and $\boldsymbol{b}^T$ is a vector of weights, showing how the final result depends on the derivatives computed at the various stages. The tableaus of this configuration are commonly referred to as *Butcher* or *Runge-Kutta* tableaus [18]. As an example, consider the following tableau for the Euler method.

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$

For the second-order method we are currently discussing in this section, multiple configurations of the tableau can be created based on the value of ϕ. For the general value of ϕ, the tableau can be written as follows.

$$\begin{array}{c|ccc} 0 & & & \\ \phi & \phi & & \\ \hline & 1 & -\frac{1}{2\phi} & \frac{1}{2\phi} \end{array}$$

Table 2.1: Butcher tableau for the second-order Runge-Kutta method (general value of ϕ).

For $\phi = \frac{1}{2}$, the values are substituted, and the resulting tableau is therefore simpler.

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}$$

Table 2.2: Butcher tableau for the second order Runge-Kutta method ($\phi = \frac{1}{2}$).

For clarity, the equations for the mid-point ($\phi = \frac{1}{2}$) variant of the Runge-Kutta method with coefficients presented in Table 2.2 are

$$\begin{aligned} k_1 &= f(t_i, y_i) \\ k_2 &= f(t_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1)\,. \end{aligned} \tag{2.14}$$

The tableaus can be used to construct higher-order methods (for more examples, see [18]). Let us briefly look at the construction of the most common order of the Runge-Kutta method in technical practice – the fourth-order method. Note that the construction of the methods using polynomial conditions (as presented in [18]) is practical only up to the fourth order due to the massive increase in complexity of the conditions for the higher orders. Several polynomial conditions have to hold in order to construct the method [18]. These polynomial conditions can be solved, and the obtained coefficients can be used to derive several variants of the Runge-Kutta methods – the most common solution being shown in Table 2.3.

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

Table 2.3: Butcher tableau for commonly used fourth-order Runge-Kutta method.

Based on Table 2.3, the most common formulation of the fourth-order Runge-Kutta method can be written as

$$y_{i+1} = y_i + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\,. \tag{2.15}$$

where coefficients k_1,k_2,k_3 and k_4 are calculated using

$$\begin{aligned} k_1 &= hf(tx_i, y_i), \\ k_2 &= hf(t_i + \frac{1}{2}h, y_i + \frac{k_1}{2}), \\ k_3 &= hf(t_i + \frac{1}{2}h, y_i + \frac{k_2}{2}), \\ k_4 &= hf(t_i + h, y_i + k_3). \end{aligned} \tag{2.16}$$

Dormand-Prince method

Similarly to the Euler method, the basic fourth-order Runge-Kutta methods are not implemented in MATLAB as presented in Table 2.3. In MATLAB [69] Dormand–Prince method is used [25]. It uses six function evaluations and a simple error predictor to increase accuracy and determine the optimal step size. This variant of the Runge-Kutta method is very widely implemented in other software and suites for numerical integration. The first of these methods [18], RK5(4)7M the method has the Butcher tableau 2.4.

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$
	$-\frac{71}{57600}$	0	$\frac{71}{16695}$	$-\frac{71}{1920}$	$\frac{17253}{339200}$	$-\frac{22}{525}$	$\frac{1}{40}$

Table 2.4: Butcher tableau for the Dormand-Prince method.

First of the output approximations has order $N+1=5$ and propagates. This method has the so-called FSAL (first Same As Last) property. This means that vector $\boldsymbol{b}$, corresponding to the output approximation, has the last component equal to zero, and it is identical to the last row of the matrix $\boldsymbol{A}$. As a consequence, while the method has seven stages, it operates as if it only has six (the seventh stage derivative can be retained as the first derivative of the subsequent step). The method can be defined without this property or with different coefficients (that would lead to a larger region of stability). For more information, see [18].

Fehlberg method

The Fehlberg methods represents the early attempt at incorporating the error estimation into the Runge-Kutta method. When writing the tableaus, the following notation can be used

$$\begin{array}{c|c} \boldsymbol{c} & \boldsymbol{A} \\ \hline & \boldsymbol{b}^T \\ & \hat{\boldsymbol{b}}^T \\ \hline & \boldsymbol{d}^T \end{array}$$

where

$$\begin{array}{c|c} \boldsymbol{c} & \boldsymbol{A} \\ \hline & \boldsymbol{b}^T \end{array}$$

is the tableau for the Runge-Kutta method of order p and

$$\begin{array}{c|c} \boldsymbol{c} & \boldsymbol{A} \\ \hline & \hat{\boldsymbol{b}}^T \end{array}$$

is the tableau for the Runge-Kutta method of order p+1. The additional vector $\boldsymbol{d}^T = \hat{\boldsymbol{b}}^T - \boldsymbol{b}^T$ is used for error estimation. The fifth-order method, with sixth-order error estimation, can be expressed as

0								
$\frac{1}{6}$	$\frac{1}{6}$							
$\frac{4}{15}$	$\frac{4}{75}$	$\frac{15}{76}$						
$\frac{2}{3}$	$\frac{5}{6}$	$-\frac{8}{3}$	$\frac{5}{2}$					
$\frac{4}{5}$	$-\frac{8}{5}$	$\frac{144}{25}$	-4	$\frac{16}{25}$				
1	$\frac{362}{320}$	$-\frac{18}{5}$	$\frac{407}{128}$	$-\frac{11}{80}$	$-\frac{55}{128}$			
0	$-\frac{11}{640}$	0	$\frac{11}{256}$	$-\frac{11}{160}$	$\frac{11}{256}$	0		
1	$\frac{93}{640}$	$-\frac{18}{5}$	$\frac{803}{256}$	$-\frac{11}{160}$	$\frac{99}{256}$	0	1	
	$\frac{31}{384}$	0	$\frac{1125}{2816}$	$\frac{9}{32}$	$\frac{125}{768}$	$\frac{5}{66}$	0	0
	$\frac{7}{1408}$	0	$\frac{1125}{2816}$	$\frac{9}{32}$	$\frac{125}{768}$	0	$\frac{5}{66}$	$\frac{5}{66}$
	$-\frac{5}{66}$	0	0	0	0	$-\frac{5}{66}$	$\frac{5}{66}$	$\frac{5}{66}$

Table 2.5: Butcher tableau for the Fehlberg method.

Methods derived by Fehlberg can have difficulty with error estimation when the formulas characterized by the vectors $\boldsymbol{b}^T$ and $\hat{\boldsymbol{b}}^T$ are identical, which would mean that the error estimate is going to be too optimistic. Further information about the Fehlberg methods and their drawbacks can be found in [18].

Trapezoidal rule

The trapezoidal rule is another one of the very well-known methods. These methods can be generally written as

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

These methods are implicit and very efficient for solving stiff problems.

2.3.3 Multi-step methods

The multi-step methods are widely used in the area of real-time simulation [49], so this thesis is going to discuss the commonly used ones in greater detail. For the non-stiff linear problems, the *Adams* methods are the most important. These methods approximate the solution at t_n either by

$$y_{i+1} = y_i + h(\beta_1 f(t_i, y_i) + \beta_2 f(t_{i-1}, y_{i-1}) + \cdots + \beta_k f(t_{i-k}, y_{i-k})) \tag{2.17}$$

(Adams-Bashforth methods) or

$$y_{i+1} = y_i + h(\beta_0 f(t_i, y_i) + \beta_1 f(t_{i-1}, y_{i-1}) + \cdots + \beta_k f(t_{i-k}, y_{i-k})) \tag{2.18}$$

(Adams-Moulton methods) where the constants β_0, β_1, β_2, ..., β_k are chosen to give the highest possible order.

For Adams-Bashforth methods, assuming that no other errors were introduced when approximation at t_i is about to be calculated, the terms on the right-hand side can be replaced by the quantities that are approximated, that is, by y_{i-1}, y'_{i-1}, y'_{i-2}, ..., y'_{i-k} respectively. The amount by which the approximation written in this form differs from $y(t_i)$ is the error generated in the particular step. If this error can be estimated for a smooth problem as $\mathcal{O}(h^{N+1})$, then the method has the order N. For Adams-Moulton methods, the term with $f(t_i, y_i)$ complicates the calculation of the order. However, the resulting order is similar to the Adams-Bashforth methods. Generally, this class of methods are specialized in that the dependence of y_i on previously computed values ignores y_{i-1}, y_{i-2}, $\dots$, y_{i-k}.

Adams-Bashfort methods

The coefficients of the Adams-Bashfort method ($\beta_0 = 0$) for $k = 1 \dots 6$ steps are in Table 2.6.

k	β_1	β_2	β_3	β_4	β_5	β_6
1	1					
2	$\frac{3}{2}$	$-\frac{1}{2}$				
3	$\frac{23}{12}$	$-\frac{16}{12}$	$\frac{5}{12}$			
4	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{9}{24}$		
5	$\frac{1901}{720}$	$-\frac{2774}{720}$	$\frac{2616}{720}$	$-\frac{1274}{720}$	$\frac{251}{720}$	
6	$\frac{4277}{1440}$	$-\frac{7923}{1440}$	$\frac{9982}{1440}$	$-\frac{7298}{1440}$	$\frac{2877}{1440}$	$-\frac{475}{1440}$

Table 2.6: Coefficients for the Adams-Bashfort method [18].

For $k = 1$, the explicit Euler method (2.12) can be obtained. For higher values of k, the multistep methods can be obtained. For example, using $k = 2$

$$y_{i+1} = h\left(\frac{3}{2}f(t_i, y_i) - \frac{1}{2}f(t_{i-1}, y_{i-1})\right).$$

Adams-Moulton methods

The coefficients of the Adams-Moulton method ($\beta_0 = 1$) for $k = 0 \dots 5$ steps are in Table 2.7.

k	β_0	β_1	β_2	β_3	β_4	β_5
0	1					
1	$\frac{1}{2}$	$\frac{1}{2}$				
2	$\frac{5}{12}$	$\frac{8}{12}$	$-\frac{1}{12}$			
3	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$		
4	$\frac{251}{720}$	$\frac{646}{720}$	$-\frac{264}{720}$	$\frac{106}{720}$	$-\frac{19}{720}$	
5	$\frac{475}{1440}$	$\frac{1427}{1440}$	$-\frac{858}{1440}$	$\frac{482}{1440}$	$-\frac{173}{1440}$	$\frac{27}{1440}$

Table 2.7: Coefficients for the Adams-Moulton method [18].

By substituting the coefficients into (2.18), the implicit methods can be obtained (the implicit Euler method for $k = 1$, trapezoidal rule for $k = 2$). A multi-step method is obtained by using $k \geq 3$.

The biggest drawback of the multi-step methods is the need to generate the k starting points $y_0, y_1, \dots, y_{k-1}$. One of the approaches that can be used for k-step Adams-Bashfort method calculated again using Adams-Bashfort method, y_1 using the one-step Adams-Bashfort method, y_2 using two-step Adams-Bashfort method and y_{k-1} using y_{k-1}-step Adams-Bashfort method.

Predictor-corrector methods

The Adams-Bashfort and Adams-Moulton methods are rarely used on their own. They are more commonly used combined as so-called *predictor-corrector* methods. The predictor (preliminary calculation in y_i) is being performed by the Adams-Bashfort method. This preliminary predicted solution at the new step value is then used to evaluate an approximation to the derivative value at the new point.

This calculation scheme is often referred to by the abbreviation PECE (predict-evaluate-correct-evaluate). The steps on the scheme for the second-order predictor-corrector method can be written as follows:

- P 2nd order Adams-Bashfort method $y^*_{i+1} = y_i + \frac{1}{2}h\left(3f_i - f_{i-1}\right)$
- E $f^*_{i+1} = f(t_{i+1}, (y^*_{i+1})$
- C 2nd order Adams-Moulton method $y_{i+1} = y_i + \frac{1}{2}h\left(f^*_{i+1} + f_i\right)$
- E $f_{i+1} = f(t_{i+1}, (y_{i+1})$.

2.4 Implementation of integration methods in MATLAB

Some methods introduced in Section 2.3 are implemented in MATLAB [54]. The most widely used numerical solvers in MATLAB that are used in the thesis are summarized in Table 2.8.

Solver	Method
ode23	Bogacki-Shampine [69]
ode45	Dormand-Prince [25]
ode113	Adams-Bashfort method with predictor-corrector PECE scheme

Table 2.8: List of used state-of-the-art numerical integration methods.

In several experiments, Runge-Kutta second-order (2.14) and fourth-order (2.15) methods are used.

2.5 Errors of the numerical integration methods

When calculating the solution using the selected numerical integration method, it is imperative to have a good understanding of errors that are present. There are two types of errors that have to be taken into account:

- local truncation error and
- global truncation error.

Local truncation error at the specified step measures the amount by which the exact solution of the differential equation fails to satisfy the difference equation being used for the approximation at that step [17].

For the initial value problem

$$y' = f(t, y), \qquad a \le t \le b, \qquad y(a) = \alpha$$

the difference method can be defined as

$$\begin{aligned} w_0 &= \alpha \\ w_{i+1} &= w_i + h\phi(t_i, w_i) \end{aligned}$$

for each $i = 0, 1, \dots, n-1$. This difference method has *local truncation error*

$$\tau_{i+1} = \frac{y_{i+1} - (y_i + h\phi(t_i, y_i))}{h} = \frac{y_{i+1} - y_i}{h} - \phi(t_i, y_i)$$

for each $i = 0, 1, \dots, n-1$, where y_i denotes the solution in the time t_i and y_{i+1} denotes the solution in the time t_{i+1}.

For the Euler method, defined previously in this Chapter, the local truncation error at the ith step

$$\tau_{i+1} = \frac{y_{i+1} - y_i}{h} - f(t_i, y_i)$$

for each $i = 0, 1, \dots, N-1$. This is indeed a local error because it measures the accuracy at the specific step, assuming that the method was accurate at the previous step. For a method, it depends on

- differential equation,
- the selected step size and
- the particular step of the approximation.

The Euler method (2.12) has

$$\tau_{i+1}(h) = \frac{h}{2} y''(\xi_i)$$

for some ξ_i in (t_i, t_{i+1}). It can be proven [17] that the local truncation error of the Euler method is $O(h)$. If the method based on the Taylor series is used to approximate the solution of the function

$$y'(t) = f(t, y(t))\,, \qquad\qquad a \le t \le b \qquad\qquad y(a) = \alpha\,,$$

with step size h, $y \in C^{n+1}[a, b]$ the series can be rewritten

$$y_{i+1} - y_i - hf(t_i, y_i) - \frac{h^2}{2!} f'(t_i, y_i) - \dots - \frac{h^n}{n!} f^{(N-1)}(t_i, y_i) = \frac{h^{N+1}}{(N+1)!} f^{(N)}(\xi_i, y(\xi_i))\,,$$

for some ξ_i in (t_i, t_{i+1}). The local truncation error can therefore be written as

$$\tau_{i+1}(h) = \frac{y_{i+1} - y_i}{h} - P^{(N)}(t_i, y_i) = \frac{h^N}{N+1!} f^{(N)}(\xi_i, y(\xi_i))\,,$$

for each $i = 0, 1, \dots, N-1$. Since $y \in C^{N+1}[a, b]$, $y^{(N+1)}(t) = f^{(N)}(t, y(t))$ bounded on $[a, b]$ and $\tau_i(h) = \mathcal{O}(h^N)$, for each $i = 1, 2, \dots, N$. Global truncation error accumulates the *local truncation error* over the solution, assuming that the initial condition is given exactly.

2.6 Transformation of higher-order ODEs into the system of first-order ODEs

The higher-order ODEs can be transformed into the equivalent system of first-order ODEs using several known approaches. These approaches have their roots in analogue computations but are still used today [12]. More information is provided in [64] or in [42], used block algebra is defined in Appendix A.

2.6.1 Method of Derivative Order Reduction

The first discussed method (Method of Derivative Order Reduction – abbreviated as MDOR) is the simplest. It can be used with systems that have the input (coercive, forcing) function z with no derivatives. Consider the following equation:

$$y'''' + a_3 y''' + a_2 y'' + a_1 y' + a_0 y = b_0 z \tag{2.19}$$

with initial conditions $y(0) = y'(0) = y''(0) = y'''(0) = 0$. As stated before, to use the MDOR method, the forcing function does not contain a derivative. It can be rewritten using a differential Laplace operator s

$$\begin{aligned} y' &= sy \\ y'' &= s^2 y \\ y''' &= s^3 y \\ &\vdots \\ y^{(n)} &= s^n y \end{aligned}$$

to obtain

$$s^4 y + a_3 s^3 y + a_2 s^2 y + a_1 s y + a_0 y = b_0 z\,. \tag{2.20}$$

It is possible to rearrange (2.20) to as the system of first-order ODEs (2.21)). Elements $\frac{1}{s}$ denote numerical integrators

$$\begin{aligned} s^4 y &= b_0 z - a_3 s^3 y - a_2 s^2 y - a_1 s y - a_0 y \\ s^3 y &= \frac{1}{s} s^4 y && s^3 y(0) = 0 \\ s^2 y &= \frac{1}{s} s^3 y && s^2 y(0) = 0 \\ sy &= \frac{1}{s} s^2 y && sy(0) = 0 \\ y &= \frac{1}{s} sy && y(0) = 0\,. \end{aligned} \tag{2.21}$$

Figure 2.3 shows the corresponding block scheme.

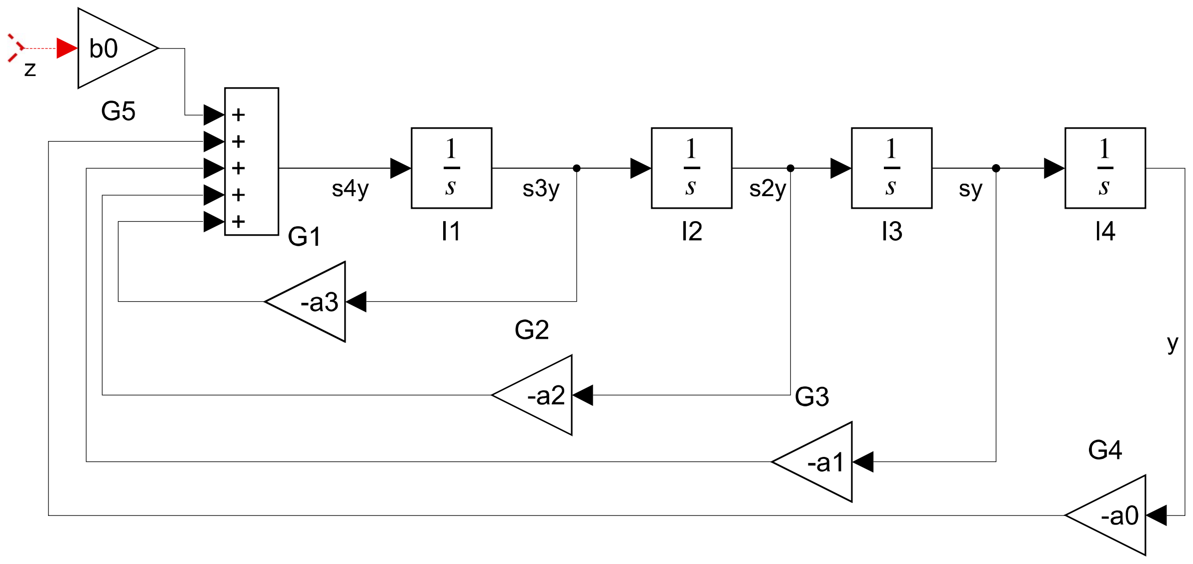


Figure 2.3: Block scheme for MDOR.

2.6.2 Method of Derivation Order Reduction with an Additional Variable

Another method that can be used, is the Method of Derivation Order Reduction with an Additional Variable, abbreviated as MDORAV. The differential equation (2.19) did not have the derivative of the forcing function z. If the differential equation contains the derivative of the forcing function, the MDOR cannot be used. For example, consider following higher order ordinary differential equation

$$y'''' + a_3 y''' + a_2 y'' + a_1 y' + a_0 y = b4 z'''' + b_3 z''' + b_2 z'' + b_1 z' + b_0 z\,, \tag{2.22}$$

with initial conditions $y(0) = y'(0) = y''(0) = y'''(0) = 0$ and $z(0) = 1, z'(0) = z''(0) = z'''(0) = z''''(0) = 0$. Using a similar transformation as with the (2.19), equation (2.22) can be simplified.

$$\begin{gathered} s^4 y + a_3 s^3 y + a_2 s^2 y + a_1 s y + a_0 y = b_4 s^4 z + b_3 s^3 z + b_2 s^2 z + b_1 s z + b_0 z \\ y(s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0) = z(b_4 s^4 + b_3 s^3 + b_2 s^2 + b_1 s + b_0)\,. \end{gathered} \tag{2.23}$$

The output of the system y can be expressed from (2.23) as

$$y = z \frac{b_4 s^4 + b_3 s^3 + b_2 s^2 + b_1 s + b_0}{s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0}\,, \tag{2.24}$$

with so-called *additional variable* v defined as

$$v = \frac{z}{s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0}\,. \tag{2.25}$$

The output of the system can therefore be written as

$$y = b_4 s^4 v + b_3 s^3 v + b_2 s^2 v + b_1 s v + b_0 v\,, \tag{2.26}$$

which means that to calculate the output of the system, the higher derivatives of the additional variable are required. Simplifying the (2.25), the equation for the additional variable can be obtained

$$s^4 v + a_3 s^3 v + a_2 s^2 v + a_1 s v + a_0 v = z\,.$$

This equation does not contain the derivative of the forcing function z, so the order can be reduced using the MDOR (defined in subsection 2.6.1).

$$\begin{aligned} s^4v &= z - a_3s^3v - a_2s^2v - a_1sv - a_0v \\ s^3v &= \frac{1}{s}s^4v & s^3v(0) &= 0 \\ s^2v &= \frac{1}{s}s^3v & s^2v(0) &= 0 \\ sv &= \frac{1}{s}s^2v & sv(0) &= 0 \\ v &= \frac{1}{s}sv & v(0) &= 0\,. \end{aligned} \tag{2.27}$$

The block scheme representing equations (2.27) and (2.26) is in Figure 2.4.

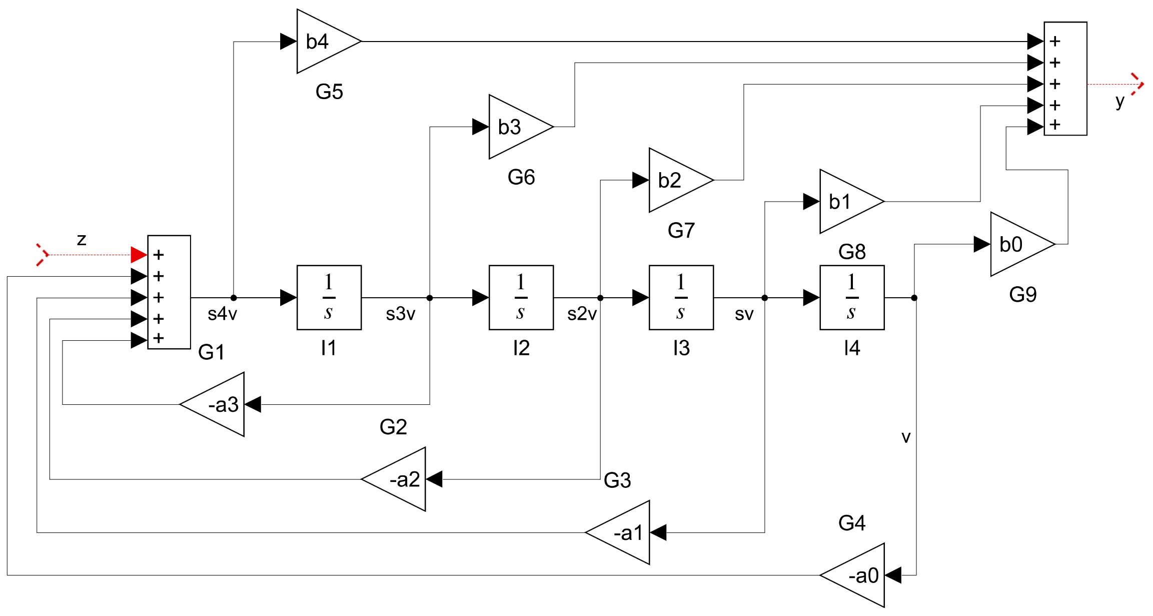


Figure 2.4: Block scheme for MDORAV.

2.6.3 Method of Successive Integration

The last method discussed in this section is the Method of Successive Integration, abbreviated as MSI. Once again, the coercive function z has a derivative. Consider the following differential equation

$$y'''' + a_3y''' + a_2y'' + a_1y' + a_0y = b_4z'''' + b_3z''' + b_2z'' + b_1z' + b_0z\,. \tag{2.28}$$

with initial conditions $y(0) = y'(0) = y''(0) = y'''(0) = 0$ and $z(0) = 1, z'(0) = z''(0) = z'''(0) = z''''(0) = 0$. Equation (2.28) can again be rewritten using the differential operator

$$s^4y + a_3s^3y + a_2s^2y + a_1sy + a_0y = b_4s^4z + b_3s^3z + b_2s^2z + b_1sz + b_0z\,. \tag{2.29}$$

The equation (2.29) can be rewritten so that the derivatives of the same order are grouped together

$$
\begin{aligned}
s^4 y &= b_4 s^4 z + s^3(b_3 z - a_3 y) + s^2(b_2 z - a_2 y) + s(b_1 z - a_1 y) + (b_0 z - a_0 y) \\
s^3 y &= b_4 s^3 z + s^2(b_3 z - a_3 y) + s(b_2 z - a_2 y) + (b_1 z - a_1 y) + v_1 \\
s^2 y &= b_4 s^2 z + s(b_3 z - a_3 y) + (b_2 z - a_2 y) + v_2 \\
sy &= b_4 s z + (b_3 z - a_3 y) + v_3 \\
y &= b_4 z + v_4\,.
\end{aligned}
\tag{2.30}
$$

Variables v_1, v_2, v_3 and v_4 can be calculated using the following system

$$
\begin{aligned}
v_1 &= \frac{1}{s}(b_0 z - a_0 y) & v_1(0) &= 0 \\
v_2 &= \frac{1}{s}(b_1 z - a_1 y + v_1) & v_2(0) &= 0 \\
v_3 &= \frac{1}{s}(b_2 z - a_2 y + v_2) & v_3(0) &= 0 \\
v_4 &= \frac{1}{s}(b_3 z - a_3 y + v_3) & v_4(0) &= 0\,.
\end{aligned}
\tag{2.31}
$$

If $a_i, b_i > 0$ then the block scheme in Figure 2.5 can be constructed.

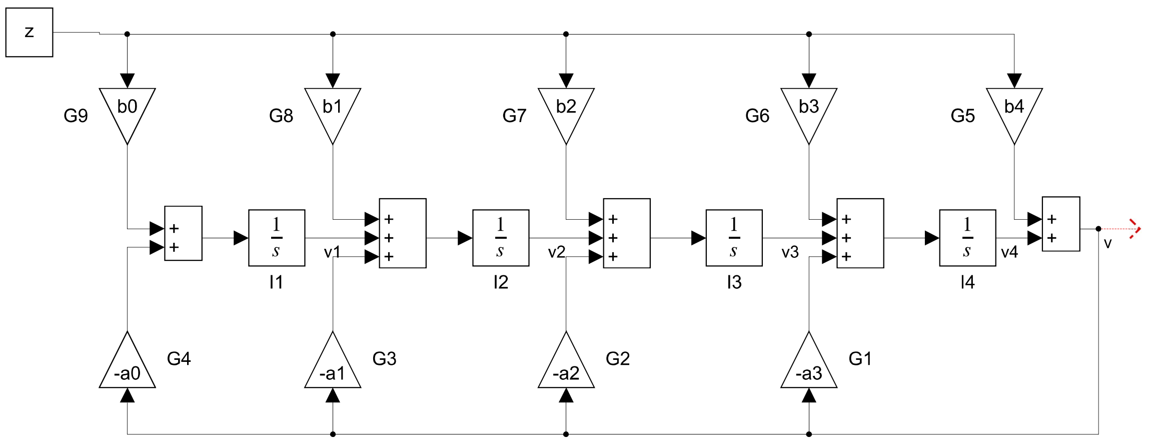


Figure 2.5: Block scheme for MSI.

2.6.4 Comparison of the methods

To compare MDORAV (Subsection 2.6.2) and MSI (Subsection 2.6.3), the following example problem can be used

$$
y'' + a_1 y' + a_0 y = b_2 z'' + b_1 z' + b_0 z \tag{2.32}
$$

with $a_1 = 2$, $a_0 = 3$, $b_2 = 8$, $b_1 = 13$, $b_0 = 1$ and $z = \sin(t)$. The error between two methods $(y_{MSI} - y_{MDORAV})$ when solving the equation is in Figure 2.6.

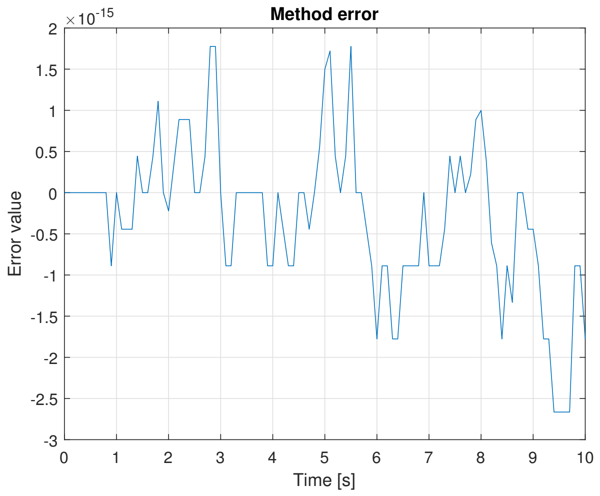


Figure 2.6: Comparison between MSI and MDORAV.

The error between the methods is very near to the numerical zero (1×10^{-15}), which means that the systems of equations generated by the methods are equivalent and can be used interchangeably.

2.7 Real-time considerations

Due to the nature of this thesis, the methods discussed above should not only provide results that are as accurate as possible but also provide those results before the hard limit of the real-time system is reached and performed calculations become useless. This might have disastrous consequences. The approaches and methods discussed in this Chapter should be viewed with an added focus on real-time applications that add additional constraints and requirements. One step of the real-time integration step usually consists of [19]:

- A/D conversion of the values obtained by the sensors. The values are obtained with a given accuracy,
- **numerical calculations** is the core problem discussed in this thesis,
- D/A conversion takes the calculated results and writes them back as analogue values,
- event handling and busy waiting waits for the end of the current integration step.

One integration step is visualized in Figure 2.7.

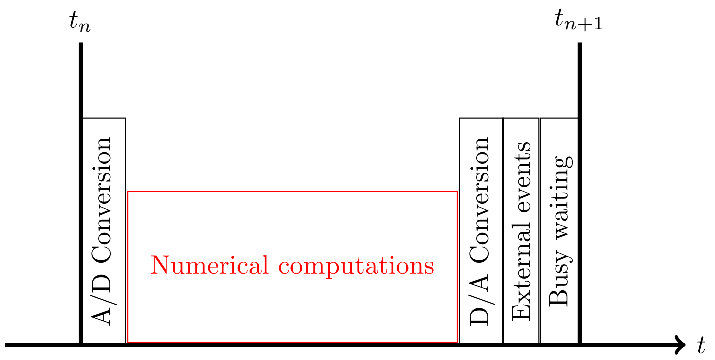


Figure 2.7: One integration step in the system working in real-time. Only the numerical computation part can be directly influenced by the method presented in this thesis.

As noted above, this thesis aims to determine if the variable-step, variable-order numerical integration method can be used in real-time context. It is, therefore necessary to establish properties of the method relevant to the context and compare them to the methods that are commonly used in real-time simulations. According to [19], the main classes of methods that might be useful in real-time context are:

- multi-step methods
  Due to the fact that these methods use higher-order polynomials, the usefulness in real-time real-world systems that do not produce smooth data is limited (the methods might give inaccurate results).

  As noted above, the methods have a lower number of evaluations, which is crucial in the real-time context.

- explicit single-step methods
  These methods again use few resources and can handle discontinuous inputs. They are not particularly well suited for stiff systems or systems that would require a larger step size than the stability criteria allows.

- implicit single-step methods
  These methods can again be used. However, they can be more computationally intensive (due to the need to solve a system of non-linear equations at each time step). This can be partially solved by limiting the number of iterations that is performed, however, that modifies the stability domain of the method.

- high-order methods
  According to the [19], due to the small sampling intervals that are used in real-time systems, it is rare to find a real-time system that uses a method with an order higher than two or three.

When considering a real-time system, the integration method races against the defined constraints of the system. If the method misses the integration step, it might lead to dire consequences. Therefore, used integration method has to end the calculation in the predetermined time as fast as possible. When we find that the method does not meet the required time constraints of the system, we can:

- increase the step size of the integration method (if it is possible), thereby decreasing the load and the time it takes to perform a computation of one step,
- optimize the model (to remove stiffness, for example) or
- improve the speed of the chosen algorithm.

The newly developed integration method (used in the thesis and was thoroughly tested on both linear and non-linear problems) cannot modify or optimize the state space model (even though some of the examples in this thesis were numerically optimized).

This thesis will show how are the state-of-the-art methods capable of handling real-time tasks and that the proposed method can be *more efficient* without a loss of precision. Further, with the proposed method using a higher order, the solution might be *faster*, and the method uses fewer operations than the state-of-the-art single-step and multi-step methods.

Chapter 3

High order Taylor series method

The high-order method based on the Taylor series is presented in this Chapter. This method is used for experiments and comparison with the state-of-the-art numerical integration methods presented in Chapter 2. The Chapter also shows positive properties of the method for possible applications in systems working in a real-time context, which is going to be demonstrated in the experiments in Chapter 4 and 6.

First, the method is defined for both linear and non-linear systems. Then, the positive properties (stability, accuracy, number of operations it performs in comparison to the state-of-the-art methods, etc.) are shown. The Chapter also shows the possible hardware implementation and its effectiveness.

Throughout this Chapter and the rest of the thesis, the method is going to be abbreviated as *MTSM* (Modern Taylor Series Method).

The best-known and the most accurate method of calculating a new value of the numerical solution of an ODE (as stated in the previous Chapter) is to construct the Taylor series in the form

$$y_{i+1} = y_i + hf(t_i, y_i) + \frac{h^2}{2!} f'(t_i, y_i) + \cdots + \frac{h^n}{n!} f^{[n-1]}(t_i, y_i)\,, \tag{3.1}$$

where h is the size of integration step, $y_i = y(t_i)$ is the previous value and $y_{i+1} = y(t_i + h)$ is the next value of the function $y(t)$ [32].

MTSM very effectively implements the variable-step-size, variable-order numerical solution of differential equations using the Taylor series. It is based on a recurrent calculation of the Taylor series terms for each integration step. Therefore, the complicated calculation of higher-order derivatives does not need to be performed. Rather the value of each Taylor series term can be numerically calculated [45]. Equation (3.1) can be rewritten in the form

$$y_{i+1} = DY(0)_i + DY(1)_i + DY(2)_i + \cdots + DY(N)_i\,, \tag{3.2}$$

where DY denotes the Taylor series terms. The function, which defines the number of used Taylor series terms during the current integration step y_{i+1} can be denoted as ORD ($ORD_{i+1} = N$).

The first implementation of MTSM is TKSL/386 (TKSL stands for Taylor-Kunovsky Simulation Language) [47]. Currently, MTSM has been implemented and tested in MATLAB [54], C++ (FOS [41], TKSL/C software [87]), Python and Julia (implemented by the author). Additionally, the method can be effectively implemented in hardware, which is going to be discussed in Section 3.7.

Multiple authors have presented several implementations of the Taylor series method in a variable-order, variable-step-size context. For example:

- TIDES software [66],
- TAYLOR [37] (includes a detailed description of a variable step size version),
- ATOMF [21],
- COSY INFINITY [52],
- DAETS [57].

The variable-step-size variable-order scheme is also described in [6], [7], [8] and [56], where simulations on a parallel computer are shown. The approach based on an approximate formulation of the Taylor methods can be found in [4]. Further research is being done, for example, in [3], which describes a generalized implementation of the Taylor series-based method with its order limited to three.

The solution of higher derivatives is different for linear and non-linear systems. Solution for linear systems is more straightforward and is going to be discussed first.

3.1 Modern Taylor Series Method for linear systems

For linear systems of ODEs, the equation (2.3) is in the form

$$\boldsymbol{y}' = \boldsymbol{A}\boldsymbol{y} + \boldsymbol{b}\,, \tag{3.3}$$

and the Taylor series (3.1) can be rewritten in matrix-vector notation as

$$\boldsymbol{y}_{i+1} = \boldsymbol{y}_i + h(\boldsymbol{A}\boldsymbol{y}_i + \boldsymbol{b}) + \frac{h^2}{2!}\boldsymbol{A}(\boldsymbol{A}\boldsymbol{y}_i + \boldsymbol{b}) + \cdots + \frac{h^N}{N!}\boldsymbol{A}^{(n-1)}(\boldsymbol{A}\boldsymbol{y}_i + \boldsymbol{b})\,, \tag{3.4}$$

where $\boldsymbol{A}$ is the constant Jacobian matrix and $\boldsymbol{b}$ is the constant right-hand side vector. The Taylor series terms DY in (3.2) can be computed recurrently using

$$\begin{aligned} \boldsymbol{DY}(0)_i &= \boldsymbol{y}_i, & \boldsymbol{DY}(1)_i &= h(\boldsymbol{A}\boldsymbol{y}_i + \boldsymbol{b}), \\ \boldsymbol{DY}(r)_i &= \frac{h}{r}\boldsymbol{A}\boldsymbol{DY}(r-1)_i, & r &= 2, \ldots, N. \end{aligned} \tag{3.5}$$

3.2 Modern Taylor Series Method for non-linear systems

The calculation using the method for non-linear systems is more complicated than for linear ones. This is due to the fact that the function being calculated is not multiplied by a constant but by other functions, and therefore, the chain rule has to be applied. This multiplication has to be performed more than once, and the number of function multiplication might be very large. This leads to a very high number of operations being performed in contrast to the linear solver, which was discussed previously. The basic principles of non-linear calculation using the method are therefore going to be discussed in great detail, including possible optimizations and drawbacks.

The solution of non-linear problems was first analysed in detail when the hardware implementation of the method was being considered and developed (several bachelor and diploma theses, for example [53] and [86]) and it is currently one of the main focuses of the FIT BUT HPC research group[1].

[1] https://www.fit.vut.cz/research/group/hpc/.en

3.2.1 Two-function multiplication and higher derivatives

Consider the following IVP

$$y' = qr \quad y(0) = y_0\,, \tag{3.6}$$

where q and r are arbitrary functions. To numerically solve this IVP using MTSM, the Taylor series in the form (3.2) for all functions (y, q and r) have to be constructed

$$\begin{aligned} y_{i+1} &= y_i + DY(1)_i + DY(2)_i + DY(3)_i + DY(4)_i + \cdots + DY(N)_i \\ q_{i+1} &= q_i + DQ(1)_i + DQ(2)_i + DQ(3)_i + DQ(4)_i + \cdots + DQ(N)_i \\ r_{i+1} &= r_i + DR(1)_i + DR(2)_i + DR(3)_i + DR(4)_i + \cdots + DR(N)_i, \end{aligned} \tag{3.7}$$

where function y represents the Taylor series for the final solution. The higher derivatives for the (3.6) can be constructed using the chain rule (3.8).

$$\begin{aligned} y' &= qr \\ y'' &= q'r + qr' \\ y''' &= q''r + q'r' + q'r' + qr'' = q''r + 2q'r' + qr'' \\ y^{[4]} &= q'''r + q''r' + 2q''r' + 2q'r'' + q'r'' + qr''' = q'''r + 3q''r' + 3q'r'' + qr''' \\ &\vdots \end{aligned} \tag{3.8}$$

Note that multiplicative constants next to derivatives create the Pascal triangle (shown here for $n = 3$).

$$\begin{array}{lccccccc} \text{n} = 0 & & & & 1 & & & \\ \text{n} = 1 & & & 1 & & 1 & & \\ \text{n} = 2 & & 1 & & 2 & & 1 & \\ \text{n} = 3 & 1 & & 3 & & 3 & & 1 \end{array}$$

The higher derivatives can therefore be calculated using the Binomial theorem, which can be generally written as

$$y^{[n+1]} = \sum_{n=0}^{N} \binom{N}{n} q^{[n-N]} r^{[n]}\,. \tag{3.9}$$

From (3.8) and (3.1), equations for the Taylor series terms can be expressed

$$\begin{aligned} \frac{DY(1)_i}{h} &= DQ(0)_i DR(0)_i \\ \frac{DY(2)_i}{\frac{h^2}{2!}} &= \frac{DQ(1)_i}{h} DR(0)_i + DQ(0)_i \frac{DR(1)_i}{h} \\ \frac{DY(3)_i}{\frac{h^3}{3!}} &= \frac{DQ(2)_i}{\frac{h^2}{2!}} DR(0)_i + 2\frac{DQ(1)_i}{h}\frac{DR(1)_i}{h} + DQ(0)_i \frac{DR(2)_i}{\frac{h^2}{2!}} \\ \frac{DY(4)_i}{\frac{h^4}{4!}} &= \frac{DQ(3)_i}{\frac{h^3}{3!}} DR(0)_i + 3\frac{DQ(2)_i}{\frac{h^2}{2!}}\frac{DR(1)_i}{h} + 3\frac{DQ(1)_i}{h}\frac{DR(2)_i}{\frac{h^2}{2!}} + DQ(0)_i \frac{DR(3)_i}{\frac{h^3}{3!}} \\ &\vdots \end{aligned} \tag{3.10}$$

Simplifying the (3.10), the equations for the individual Taylor series terms can finally be derived:

$$\begin{aligned}
DY(1)_i &= hDQ(0)_iDR(0)_i\\
DY(2)_i &= \frac{h}{2}(DQ(1)_iDR(0)_i + DQ(0)_iDR(1)_i)\\
DY(3)_i &= \frac{h}{3}(DQ(2)_iDR(0)_i + DQ(1)_iDR(1)_i + DQ(0)_iDR(2)_i)\\
DY(4)_i &= \frac{h}{4}(DQ(3)_iDR(0)_i + DQ(2)_iDR(1)_i + DQ(1)_iDR(2)_i + DQ(0)_iDR(3)_i)\\
&\vdots
\end{aligned} \tag{3.11}$$

Generally, the higher derivatives for two-function multiplication can be calculated using the following formula

$$DY(n)_i = \frac{h}{n}\sum_{a=1}^{n} DQ(n-a)_iDR(a-1)_i\,, \quad n = 1, \dots, N. \tag{3.12}$$

The number of element-by-element multiplications performed can be significant for higher orders of the Taylor series. The problem becomes more pronounced when more function multiplications are introduced (which is common in real-world systems).

3.2.2 Three-function multiplication and higher derivatives

For the three-function multiplication, the same principle can be applied as for the two-function multiplication. For three multiplications, the elementary IVP can be written as

$$y' = qrs \quad y(0) = y_0\,, \tag{3.13}$$

where q, r and s are arbitrary functions. To numerically solve the IVP using MTSM, the Taylor series for all functions (y, q, r and s) have to be constructed

$$\begin{aligned}
y_{i+1} &= y_i + DY(1)_i + DY(2)_i + DY(3)_i + DY(4)_i + \cdots + DY(N)_i\\
q_{i+1} &= q_i + DQ(1)_i + DQ(2)_i + DQ(3)_i + DQ(4)_i + \cdots + DQ(N)_i\\
r_{i+1} &= r_i + DR(1)_i + DR(2)_i + DR(3)_i + DR(4)_i + \cdots + DR(N)_i\\
s_{i+1} &= s_i + DS(1)_i + DS(2)_i + DS(3)_i + DS(4)_i + \cdots + DS(N)_i.
\end{aligned} \tag{3.14}$$

The higher derivatives for the (3.13) can be constructed:

$$\begin{aligned}
y' &= qrs\\
y'' &= q'rs + qr's + qrs'\\
y''' &= q''rs + q'r's + q'rs' + q'r's + qr''s + qr's' + q'rs' + qr's' + qrs''\\
&= q''rs + qr''s + qrs'' + 2q'r's + 2q'rs' + 2qr's'\\
y^{[4]} &= q'''rs + q''r's + q''rs' + q'r''s + qr'''s + qr''s' + q'rs'' + qr's'' + qrs''' +\\
&+ 2q''r's + 2q'r''s + 2q'r's' + 2q''rs' + 2q'r's' + 2q'rs'' + 2q'r's' + 2qr''s' + 2qr's''\\
&= q'''rs + qr'''s + qrs''' + 3q''r's + 3q''rs' + 3q'r''s + 3qr''s' + 3q'rs'' + 3qr's'' + 6q'r's'\\
&\vdots
\end{aligned} \tag{3.15}$$

From (3.15) and (3.1), equations for the Taylor series terms can be expressed

$$\begin{aligned}
\frac{DY(1)_i}{h} &= DQ(0)_i DR(0)_i DS(0)_i \\
\frac{DY(2)_i}{\frac{h^2}{2!}} &= \frac{DQ(1)_i}{h} DR(0)_i DS(0)_i + DQ(0)_i \frac{DR(1)_i}{h} DS(0)_i + DQ(0)_i DR(0)_i \frac{DS(1)_i}{h} \\
\frac{DY(3)_i}{\frac{h^3}{3!}} &= \frac{DQ(2)_i}{\frac{h^2}{2!}} DR(0)_i DS(0)_i + DQ(0)_i \frac{DR(2)_i}{\frac{h^2}{2!}} DS(0)_i + DQ(0)_i DR(0)_i \frac{DS(2)_i}{\frac{h^2}{2!}} + \\
&\quad 2\frac{DQ(1)_i}{h}\frac{DR(1)_i}{h} DS(0)_i + 2\frac{DQ(1)_i}{h} DR(0)_i \frac{DS(1)_i}{h} + 2DQ(0)_i \frac{DR(1)_i}{h}\frac{DS(1)_i}{h} \\
\frac{DY(4)_i}{\frac{h^4}{4!}} &= \frac{DQ(3)_i}{\frac{h^3}{3!}} DR(0)_i DS(0)_i + DQ(0)_i \frac{DR(3)_i}{\frac{h^3}{3!}} DS(0)_i + DQ(0)_i DR(0)_i \frac{DS(3)_i}{\frac{h^3}{3!}} + \\
&\quad + 3\frac{DQ(2)_i}{\frac{h^2}{2!}}\frac{DR(1)_i}{h} DS(0)_i + 3\frac{DQ(2)_i}{\frac{h^2}{2!}} DR(0)_i \frac{DS(1)_i}{h} + 3\frac{DQ(1)_i}{h}\frac{DR(2)_i}{\frac{h^2}{2!}} DS(0)_i + \\
&\quad + 3DQ(0)_i \frac{DR(2)_i}{\frac{h^2}{2!}}\frac{DS(1)_i}{h} + 3\frac{DQ(1)_i}{h} DR(0)_i \frac{DS(2)_i}{\frac{h^2}{2!}} + 3DQ(0)_i \frac{DR(1)_i}{h}\frac{DS(2)_i}{\frac{h^2}{2!}} + \\
&\quad + 6\frac{DQ(1)_i}{h}\frac{DR(1)_i}{h}\frac{DS(1)_i}{h} \\
&\vdots
\end{aligned} \tag{3.16}$$

Simplifying (3.16), the equations for the individual Taylor series terms can finally be derived:

$$\begin{aligned}
DY(1)_i &= hDQ(0)_i DR(0)_i DS(0)_i \\
DY(2)_i &= \frac{h}{2}(DQ(1)_i DR(0)_i DS(0)_i + DQ(0)_i DR(1)_i DS(0)_i + DQ(0)_i DR(0)_i DS(1)_i) \\
DY(3)_i &= \frac{h}{3}(DQ(2)_i DR(0)_i DS(0)_i + DQ(0)_i DR(2)_i DS(0)_i + DQ(0)_i DR(0)_i DS(2)_i + \\
&\quad + DQ(1)_i DR(1)_i DS(0)_i + DQ(1)_i DR(0)_i DS(1)_i + DQ(0)_i DR(1)_i DS(1)_i) \\
DY(4)_i &= \frac{h}{4}(DQ(3)_i DR(0)_i DS(0)_i + DQ(0)_i DR(3)_i DS(0)_i + DQ(0)_i DR(0)_i DS(3)_i + \\
&\quad + DQ(2)_i DR(1)_i DS(0)_i + DQ(2)_i DR(0)_i DS(1)_i + DQ(1)_i DR(2)_i DS(0)_i + \\
&\quad + DQ(0)_i DR(2)_i DS(1)_i + DQ(1)_i DR(0)_i DS(2)_i + DQ(0)_i DR(1)_i DS(2)_i + \\
&\quad + DQ(1)_i DR(1)_i DS(1)_i) \\
&\vdots
\end{aligned} \tag{3.17}$$

Generally, the higher derivatives for three-function multiplication can be calculated using the following formula

$$DY(n)_i = \frac{h}{n}\sum_{a=0}^{n-1} DQ(a)_i \sum_{b=1}^{n-a} DR(b-1)_i DS(n-a-b)_i\,, \quad n = 1, \ldots, N. \tag{3.18}$$

The derivation for more function multiplications is in Appendix C. Equations (3.11), (3.17), (C.5) and (C.10) show that the number of operations needed to calculate the Taylor series terms increases quite rapidly with more function multiplications and for higher orders.

3.2.3 Matrix-vector representation

When solving the non-linear systems of ODEs that contain one or more multiplications of functions, (2.3) can be rewritten as

$$\boldsymbol{y}' = \boldsymbol{A}\boldsymbol{y} + \boldsymbol{B}_1\boldsymbol{y}_{jk} + \boldsymbol{B}_2\boldsymbol{y}_{jkl} + \ldots + \boldsymbol{b}, \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0, \tag{3.19}$$

where $\boldsymbol{A} \in \boldsymbol{R}^{ne\times ne}$ is the constant matrix for the linear part of the system (see Section 3.1), the $\boldsymbol{B}_1 \in \boldsymbol{R}^{ne\times nm_{jk}}$, $\boldsymbol{B}_2 \in \boldsymbol{R}^{ne\times nm_{jkl}}$ are the constant matrices for non-linear part of the system. The vector $\boldsymbol{b} \in \boldsymbol{R}^{ne}$ is the right-hand side for the forces incoming to the system, $\boldsymbol{y}_0$ is a vector of the initial conditions and symbol ne stands for the number of equations of the system of ODEs. Symbols nm_{jk} and nm_{jkl} represent the number of two and three-function multiplications, respectively.

The unknown function $\boldsymbol{y}_{jk} \in \boldsymbol{R}^{nm_{jk}}$ represents the vector of multiplications $\boldsymbol{y}_j \odot \boldsymbol{y}_k$ and similarly $\boldsymbol{y}_{jkl} \in \boldsymbol{R}^{nm_{jkl}}$ represents the vector of multiplications $\boldsymbol{y}_{jj} \odot \boldsymbol{y}_{kk} \odot \boldsymbol{y}_{ll}$, where indices $j, k, jj, kk, ll \in (1, \ldots, ne)$ come from multiplications terms in (3.19). The operation $\odot$ stands for *element-by-element* multiplication, i.e. $\boldsymbol{y}_j \odot \boldsymbol{y}_k$ is a vector $(y_{j_1}y_{k_1}, y_{j_2}y_{k_2}, \ldots, y_{j_{nm_{jk}}}y_{k_{nm_{jk}}})^T$. For simplification, the matrices $\boldsymbol{A}, \boldsymbol{B}_1, \boldsymbol{B}_2, \ldots$ and the vector $\boldsymbol{b}$ are constant. The higher derivatives of the terms $\boldsymbol{B}_1\boldsymbol{y}_{jk}, \boldsymbol{B}_2\boldsymbol{y}_{jkl}$ in (3.19) can be included in a recurrent calculation of the Taylor series terms $\boldsymbol{DY}_{B1}$ and $\boldsymbol{DY}_{B2}$

$$\begin{aligned}
&\boldsymbol{DY}(1)_A = h(\boldsymbol{A}\boldsymbol{y}_i + \boldsymbol{b})\,, \boldsymbol{DY}(1)_{B1} = h(\boldsymbol{B}_1\boldsymbol{y}_{jk})\,, \boldsymbol{DY}(1)_{B2} = h(\boldsymbol{B}_2\boldsymbol{y}_{jkl})\,,\\
&\boldsymbol{DY}(r)_A = \frac{h}{r}\boldsymbol{A}\boldsymbol{DY}(r-1)\,,\\
&\boldsymbol{DY}(r)_{B1} = \frac{h}{r}\left(\boldsymbol{B}_1\sum_{a=1}^{r}\boldsymbol{DY}(a-1)_j \odot \boldsymbol{DY}(r-a)_k\right)\,,\\
&\boldsymbol{DY}(r)_{B2} = \frac{h}{r}\boldsymbol{B}_2\sum_{a=0}^{r-1}\boldsymbol{DY}(a)_{jj} \odot \left(\sum_{b=1}^{r-a}\boldsymbol{DY}(b-1)_{kk} \odot \boldsymbol{DY}(r-a-b)_{ll}\right)\,,
\end{aligned} \tag{3.20}$$

where $r = 2, \ldots, N$. Finally, the Taylor series terms are calculated as a sum of linear and non-linear terms

$$\boldsymbol{DY}(n) = \boldsymbol{DY}(n)_A + \boldsymbol{DY}(n)_{B1} + \boldsymbol{DY}(n)_{B2}\,, \qquad n = 1, \ldots, N\,, \tag{3.21}$$

where r and n are the current indexes of the Taylor series terms, a and b are the auxiliary indexes for the summation of two and three-term multiplications in non-linear part of the Taylor series, $\boldsymbol{DY}(r-1)_A$ is the linear term computed using recurrent calculation for linear systems (see Chapter 3.1). The next value of the function can be calculated using

$$\boldsymbol{y}_{i+1} = \boldsymbol{DY}(0)_i + \boldsymbol{DY}(1)_i + \boldsymbol{DY}(2)_i + \ldots + \boldsymbol{DY}(N)_i\,, \tag{3.22}$$

where $\boldsymbol{DY}(1)_i$ is the value of the function $\boldsymbol{y}_i$, $\boldsymbol{DY}(1)_i$, ..., $\boldsymbol{DY}(n)_i$ are the Taylor series terms calculated using (3.21). Multiplication terms of the the Taylor series for more multiplications $\boldsymbol{DY}_{B3}$, $\boldsymbol{DY}_{B4}$, ... can be calculated recurrently in a similar way.

3.3 Performance of the Modern Taylor Series Method and its optimizations

In this section, the performance of MTSM is going to be discussed in comparison with other state-of-the-art numerical methods that were discussed in Chapter 2. Several fundamental improvements to increase the performance of the method further are also presented.

3.3.1 Linear problems

For linear problems, the number of basic arithmetic operations (addition, subtraction, multiplication and division) can be calculated using (3.5). For the *first* term of the Taylor series ($n = 1$), the method performs:

- *one* matrix-vector multiplication ($\boldsymbol{A_y} = \boldsymbol{Ay}$),
- *one* matrix-vector addition ($\boldsymbol{A_y} + \boldsymbol{b}$), *one* matrix multiplication by constant ($\boldsymbol{DY_1} = h(\boldsymbol{A_y} + \boldsymbol{b})$),

and for higher terms of the Taylor series ($n = 2, \ldots, N$) it performs

- *one* matrix-vector multiplication ($\boldsymbol{Ay}_{n-1}$),
- *one* division by constant ($\boldsymbol{DY_n} = \frac{h}{n}(\boldsymbol{Ay}_{n-1})$.

Overall, the method (for the worst case $n = max_{ORD}$) performs

- n matrix vector multiplications and
- n matrix vector additions.

When using a method in a non-parallel fashion, the standard definition of the method outperforms the state-of-the-art numerical solvers by wide margins. It has many positive properties (more on that in Section 3.5). However, it does not work well when trying to use it in the multiple worker configuration, which is often required to solve complex problems. This is due to the fact, that every worker has to have all initial conditions (vector $\boldsymbol{y}$) for every Taylor series term. This bottlenecks the computation severely.

To alleviate this problem, the matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ can be pre-calculated for the selected step size and order so that the method becomes fixed-step fixed-order. When doing so, the resulting matrix can be easily column-wise decomposed and split between workers that then use only their respective slices of the initial condition vector. This approach is thoroughly discussed in [59], [60], [43] and other publications of our research group.

It is usable even when not performing the calculation in parallel. However, the pre-calculation of large matrices might be time-consuming. And the fact that step size and the maximum order of the method have to be set before the pre-calculation and cannot be changed afterwards causes several problems. The step size has to be small enough so that the halving of the step size that occurs when $ORD > N$ cannot occur, and the results become unusable. This can be mitigated by estimating the step size. However, the estimate is not necessarily optimal for the method and might lead to performance degradation. Therefore, this approach might be very risky in the real-time control context with strict time constraints.

3.3.2 Non-linear problems

The calculation for non-linear problems using MTSM can be challenging due to the fact that all possible combinations of derivatives of unknown functions have to be calculated for every Taylor series term (3.20). This is a serious problem because the method requires calculating many Taylor series terms to achieve the desired accuracy. The number of element-by-element multiplications that are required to calculate the Taylor series terms $\boldsymbol{DY}(n)$ is in Figure 3.1.

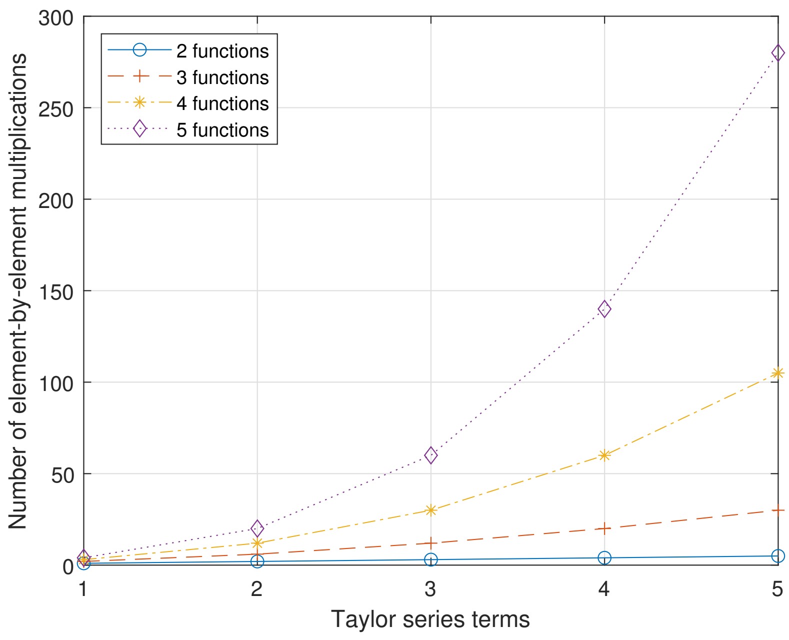


Figure 3.1: The number of element-by-element multiplication operations for two, three, four and five function multiplications.

Note that the number of operations increases exponentially for more function multiplications, so it is imperative to try and decrease the needed number of element-by-element multiplications as much as possible. To demonstrate the chosen approach, consider the following set of tables, which show the element-by-element multiplications that need to be performed during calculation of the Taylor series terms. Note that in the tables, multiplication is column-wise.

Table 3.1 shows the indexes of multiplications between the two terms of the Taylor series.

n	Indexes of Taylor series terms				
1	0				
	0				
2	1	0			
	0	1			
3	2	1	0		
	0	1	2		
4	3	2	1	0	
	0	1	2	3	
5	4	3	2	1	0
	0	1	2	3	4

Table 3.1: Two term multiplications.

Notice that there are no repeated columns or parts of columns in Table 3.1. This means that the number of multiplications cannot be decreased for term-by-term multiplication of two Taylor series terms. Term-by-term multiplications of three Taylor series terms can be described similarly.

n	Indexes of Taylor series terms																				
1	0																				
	0																				
	0																				
2	1	0	0																		
	0	1	0																		
	0	0	1																		
3	2	1	1	0	0	0															
	0	1	0	2	1	0															
	0	0	1	0	1	2															
4	3	2	2	1	1	1	0	0	0	0											
	0	1	0	2	1	0	3	2	1	0											
	0	0	1	0	1	2	0	1	2	3											
5	4	3	3	2	2	2	1	1	1	1	0	0	0	0	0						
	0	1	0	2	1	0	3	2	1	0	4	3	2	1	0						
	0	0	1	0	1	2	0	1	2	3	0	1	2	3	4						
6	5	4	4	3	3	3	2	2	2	2	1	1	1	1	1	0	0	0	0	0	0
	0	1	0	2	1	0	3	2	1	0	4	3	2	1	0	5	4	3	2	1	0
	0	0	1	0	1	2	0	1	2	3	0	1	2	3	4	0	1	2	3	4	5

Table 3.2: Three term multiplications.

Notice that indices in the coloured areas in Table 3.2 repeat. This means that the term-by-term multiplication of the coloured areas can be calculated only once during the computation, saved as a scalar and used in the multiplications instead. The indexes change only in the first (uncoloured) row. This approach leads to substantial savings of computational resources, as can be seen in Figure 3.2.

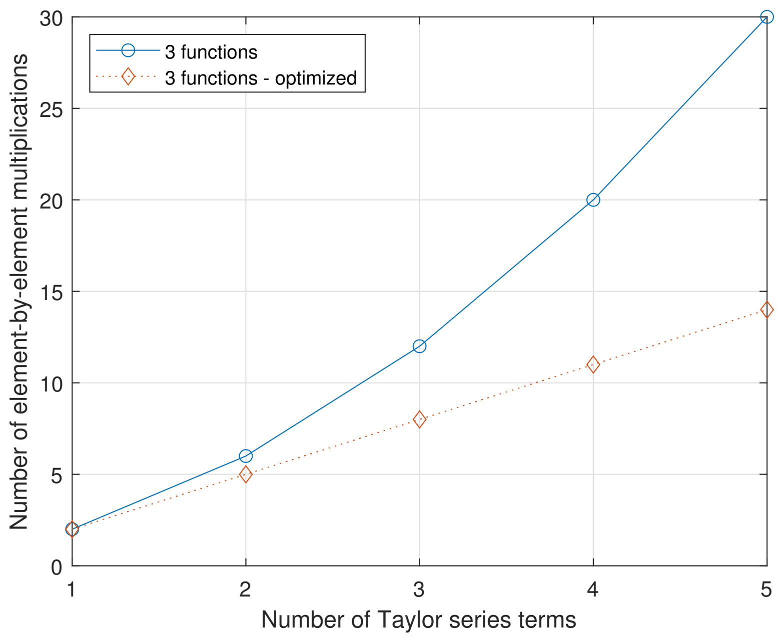


Figure 3.2: The number of element-by-element multiplication operations for three multiplications and the performed optimization.

The same approach works for more multiplications. Details are in Appendix D, including the description of the two different optimizations that were performed. The impact of the performed optimizations is very substantial, as shown in Figures 3.3 and 3.4.

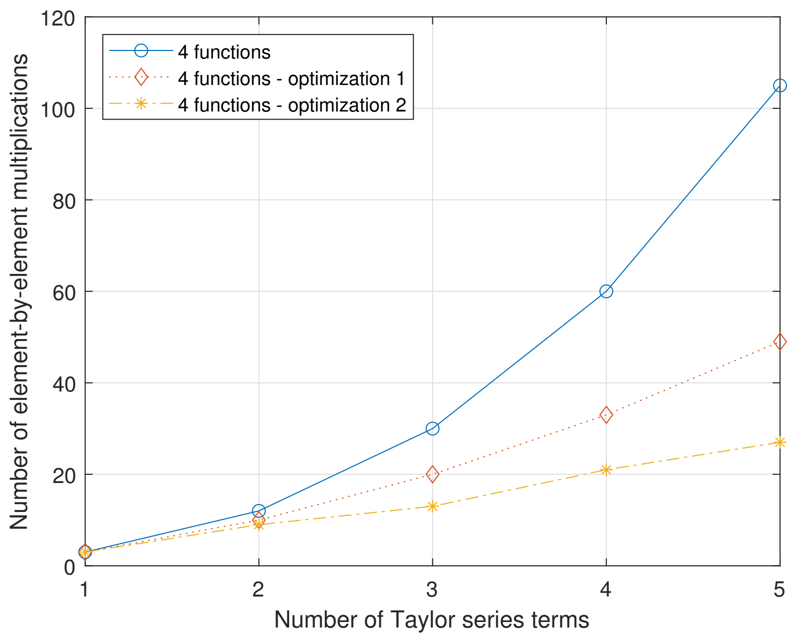


Figure 3.3: The number of element-by-element multiplication operations for four multiplications and the performed optimizations.

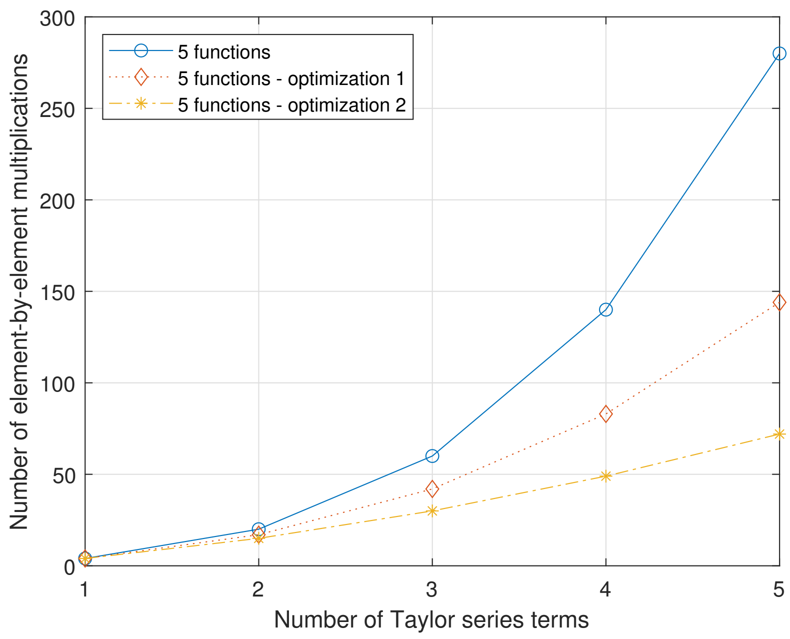


Figure 3.4: The number of element-by-element multiplication operations for five multiplications and the performed optimizations.

The results for more multiplications are similar. Other approaches to the optimization of this problem are going to be the aim of further research. The increase in speed is going to be shown in the non-linear examples in Chapters 4 and 6.

3.3.3 Step size control for non-linear systems

Additional approaches were tried in order to improve the performance of the non-linear MTSM solver. For some problems that are going to be discussed in the thesis, the value of the ORD function does not fluctuate during calculation but stays bound around a relatively low value (i.e. 10). Due to the fact that the method can automatically adjust the value of ORD in the current step based on the size of the step, the size can be dynamically increased to decrease the overall number of operations the method has to perform.

To work with this optimization, let us define h_{scale} as the scaling factor for the size of integration step h

$$h_{new} = h_{scale} \cdot h\,.$$

The new value size of integration step h_{new} is used until the calculation ends. The scaling factor is only applied when

$$\sum_{a=i-3}^{i} ORD(a) = min_{ORD} \cdot 3\,,$$

where min_{ORD} is the value of the ORD function that has to be kept for three integration steps. This approach is useful for problems where the previous approach cannot be used (i.e. for systems that only contain two function multiplications).

3.3.4 Further optimizations and improvements

The original version of the MTSM solver for non-linear systems $MTSM_{orig}$ has been improved by the optimizations from Subsection 3.3.2 and Subsection 3.3.3. Additional optimizations (mainly to increase the performance for two function multiplications) were also performed

- optimization of data structures and variable handling,
- removed unused operations for multiplications that are not performed,
- and other smaller optimizations and improvements.

The optimized solver for non-linear systems $MTSM_{opt}$ is generally faster than $MTSM_{orig}$ solver and state-of-the-art (see Chapter 4).

3.4 Automatic transformation

To use the method, the system of ODEs that describes the problem has to be transformed into a system of autonomous ODEs. This new system contains just elementary operations (addition, subtraction and multiplication) and allows for the recurrent calculation of the Taylor series terms. This Section presents widely used transformations and examples of their usage.

3.4.1 Elementary operations

First, let us briefly go over the transformation of basic arithmetic operations – addition, subtraction, multiplication and division.

Addition and subtraction

Addition and subtraction are rather simple. For the function

$$y = f(t) + g(t)$$

the derivative is simply a sum of derivatives of functions $f(t)$ and $g(t)$

$$y' = f(t)' + g(t)' \,. \tag{3.23}$$

When the functions f and g are not functions of time

$$y = f(a) + g(b) \,,$$

the resulting equation becomes

$$y' = f(a)'a' + g(b)'b' \,. \tag{3.24}$$

The operations are the same for subtraction, only with the minus sign instead of the plus sign.

Multiplication

Derivative of multiplication

$$y = f(t)g(t)$$

is

$$y' = f(t)'g(t) + f(t)g(t)' \,.$$

When the functions f and g are not functions of time

$$y = f(a)g(b) \,,$$

the resulting equation becomes

$$y' = f(a)'a'g(t) + f(a)g(b)'b' \,. \tag{3.25}$$

Division

Division is the most challenging operation from the arbitrary ones because it is the most expansive operation to perform on a modern CPU (it is up to several times slower than other operations, see[2]). It would therefore be beneficial to substitute it for a different operation that would be more effective. One of the possible approaches is to replace the operation division

$$y = \frac{1}{f(a)} = f(a)^{-1}$$

by multiplication. The derivative of y can therefore be written as

$$y' = -f(a)^{-2}f(a)'a' \,,$$

and because $f(a)^{-1} = y$, the equation can be simplified as

$$y' = -y^2 f(a)'a' \,.$$

3.4.2 Elementary functions

Now, let us discuss the transformations for several common elementary functions.

Exponential function

The exponential function in the form

$$y = e^t$$

simple to differentiate, therefore an auxiliary differential equation has a form

$$y' = e^t = y \,.$$

If exponential function has an arbitrary function ($f(a)$) in the argument

$$y = e^{f(a)}$$

the compound rule is applied

$$y' = e^{f(a)} f(a)' = y f(a)' \,.$$

[2]http://ithare.com/infographics-operation-costs-in-cpu-clock-cycles/

Sine and cosine functions

Sine and cosine functions are widely used and will be discussed next. For the sine function with time as the argument

$$y = \sin(t)$$

the derivative is well known

$$y' = \cos(t)\,.$$

The aim of the transformation has not been achieved yet, one new function was generated – the cosine function that has to also be transformed

$$z = \cos(t)\,.$$

This new function can again be derived

$$z' = -\sin(t)\,.$$

The resulting system

$$\begin{aligned} y' &= \cos(t) \\ z' &= -\sin(t) \end{aligned}$$

can be simplified using the original equations for sine and cosine

$$\begin{aligned} y' &= z & y(0) &= \sin(0) \\ z' &= -y & z(0) &= \cos(0)\,. \end{aligned}$$

Note that this system generates both sine and cosine functions. If a sine function has a function in the argument

$$y = \sin(f(a))$$

the steps to take are similar. The derivative of y can be calculated as

$$y' = \cos(f(a))f(a)'\,,$$

the addition equation for the cosine function

$$\begin{aligned} z &= \cos(f(a)) \\ z' &= -\sin(f(a))f'(a)\,, \end{aligned}$$

and the final system of two ODEs can be obtained

$$\begin{aligned} y' &= zf'(a) & y(0) &= \sin(f(a_0)) \\ z' &= -yf'(a) & z(0) &= \cos(f(a_0))\,, \end{aligned}$$

where a_0 is the value of the function $f(a)$ at time $t = 0$.

Nth root

The Nth root can also be represented using an ODE. For example, square root

$$y = \sqrt{f(t)}\,,$$

can be rewritten as

$$y = f(t)^{\frac{1}{2}}\,,$$

and an ordinary differential equation

$$y' = \frac{1}{2} f(t)^{-\frac{1}{2}} f'(t)\,.$$

3.4.3 Transformation example

As an example, consider the following ODE

$$y' = \sin(\sqrt{\cos(t)}) \qquad y(0) = y_0\,. \tag{3.26}$$

The transformation starts from the innermost function, so in this example, $\cos(t)$ function is substituted

$$y' = \sin\left(\sqrt{y_1}\right)$$

where y_1 is the new auxiliary ODE that can be calculated using the following system

$$\begin{aligned} y_1 &= \cos(t) & y_2 &= \sin(t) \\ y_1' &= -\sin(t) = -y_2 \quad y_1(0) = \cos(t_0) & y_2' &= \cos(t) = y_1 \quad y_2(0) = \sin(t_0)\,. \end{aligned}$$

Now the $\sqrt{y_1}$ function can be substituted by the auxiliary ODE

$$\begin{aligned} y_3 &= y_1^{\frac{1}{2}} \\ y_3' &= \frac{1}{2} y_1^{-\frac{1}{2}} y_1' = -\frac{1}{2} y_2 y_3^{-1} \end{aligned}$$

with division y_3^{-1} that can also be removed

$$\begin{aligned} y_4 &= y_3^{-1} \\ y_4' &= -y_3^{-2} y_3' = -y_4^2 y_3' = -y_4^2(-\frac{1}{2} y_2 y_3^{-1}) = -y_4^2(-\frac{1}{2} y_2 y_4) = \frac{1}{2} y_2 y_4 y_4 y_4 \quad y_4(0) = y_3(0)^{-1} \end{aligned}$$

so that the equation for y_3' has the final form

$$y_3' = -\frac{1}{2} y_2 y_4 \quad y_3(0) = y_1(0)^{\frac{1}{2}}\,.$$

The original ODE (3.26) now has the form

$$y' = \sin(y_3)$$

so that it can be directly replaced by the following system of ODEs

$$\begin{aligned} y_5 &= \sin(y_3) & y_6 &= \cos(y_3) \\ y_5' &= \cos(y_3) y_3' = y_6 y_3' & y_6' &= -\sin(y_3) y_3' = -y_5 y_3' \\ &= -\frac{1}{2} y_2 y_4 y_6 \quad y_5(0) = \sin(y_3(0)) & &= \frac{1}{2} y_2 y_4 y_5 \quad y_6(0) = \cos(y_3(0))\,. \end{aligned}$$

When all transformations are finished, the resulting system of autonomous ODEs contains seven equations

$$\begin{aligned} y' &= y_5 & y(0) &= y_0 \\ y_1' &= -y_2 & y_1(0) &= \cos(0) \\ y_2' &= y_1 & y_2(0) &= \sin(0) \\ y_3' &= -\frac{1}{2} y_2 y_4 & y_3(0) &= y_1(0)^{\frac{1}{2}} = \sqrt{\cos(0)} \\ y_4' &= \frac{1}{2} y_2 y_4 y_4 y_4 & y_4(0) &= y_3(0)^{-1} = \sqrt{\cos(0)}^{-1} = \frac{1}{\sqrt{\cos(0)}} \\ y_5' &= -\frac{1}{2} y_2 y_4 y_6 & y_5(0) &= \sin(y_3(0)) = \sin(\sqrt{\cos(0)}) \\ y_6' &= \frac{1}{2} y_2 y_4 y_5 & y_6(0) &= \cos(y_3(0)) = \cos(\sqrt{\cos(0)})\,. \end{aligned} \tag{3.27}$$

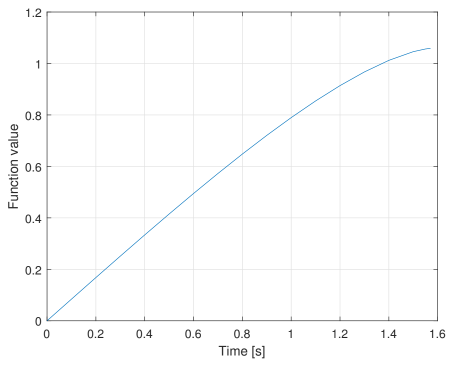
The resulting system (3.27) contains multiplications of functions and is non-linear. To check if the solution of (3.26) is equal to the solution of (3.27), both can be solved using the state-of-the-art ODE solver, the *ode45* for example.

The difference between the results of (3.26) and (3.27) $||y_{t_{MAX}}(3.26) - y_{t_{MAX}}(3.27)||$ is 3.1811×10^{-9}. Just for the sake of completeness, the system (3.27) can be solved using the MTSM. The matrix-vector notation for the MTSM solver is in (3.28).

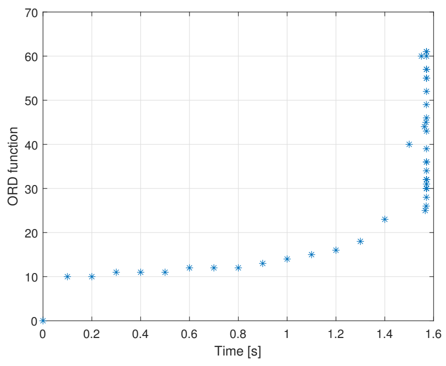
$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \boldsymbol{B}_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -0.5 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \boldsymbol{B}_2 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \quad \boldsymbol{B}_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.5 \\ 0 \\ 0 \end{pmatrix} \tag{3.28}$$

$$\boldsymbol{y}_{jk} = \begin{pmatrix} 5 & 3 \end{pmatrix} \quad \boldsymbol{y}_{jkl} = \begin{pmatrix} 5 & 3 & 7 \\ 5 & 3 & 6 \end{pmatrix} \quad \boldsymbol{y}_{jklm} = \begin{pmatrix} 5 & 5 & 5 & 3 \end{pmatrix}$$

The vector for the right-hand side $\boldsymbol{b}$ is zero. The solution of the ODE (3.26) and the *ORD* function are in Figure 3.5.



(a) Solution of the ODE (3.26).



(b) *ORD* function for (3.26).

Figure 3.5: Solution of the ODE (3.26) (Figure 3.5a) and the *ORD* function (Figure 3.5b) for $h = 0.1$ s. Note that the *ORD* function increases dramatically as the function approaches the discontinuity in the solution.

3.5 Positive properties

The presented method (MTSM) has several positive properties. This Section is going to discuss them in greater detail.

3.5.1 Automatic order setting

The first positive property is an *automatic integration order setting*. The method uses as many Taylor series terms as required by the defined accuracy of the calculation. As an example, consider the system of ODEs

$$\begin{aligned} y' &= z \quad & y(0) &= 0 \\ z' &= -y \quad & z(0) &= 5 \end{aligned} \tag{3.29}$$

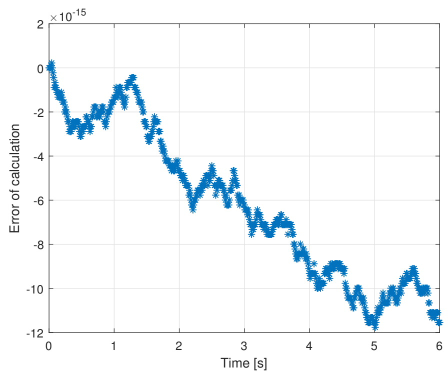
with the analytical solution

$$\begin{aligned} y &= 5\sin(t) \\ z &= 5\cos(t)\,. \end{aligned} \tag{3.30}$$

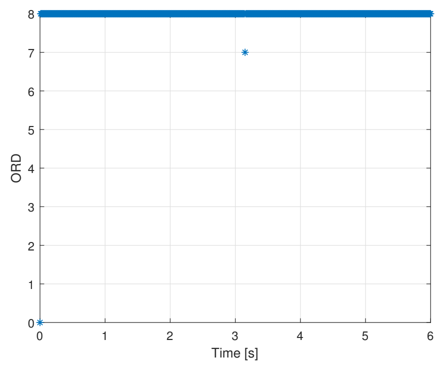
This system of ODEs is linear, (3.3) is used with the following values

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 0 \\ 5 \end{pmatrix}.$$

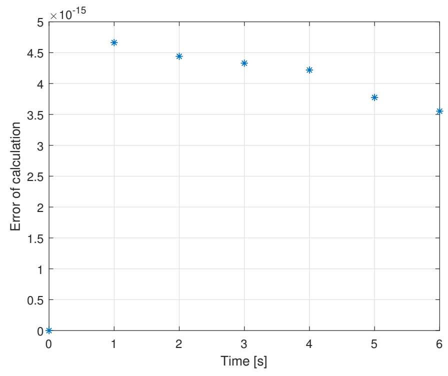
In the following experiment, the accuracy of all numerical solvers is set to $TOL = 1 \times 10^{-12}$. Further, the size of the integration step h can be changed arbitrarily without any meaningful loss of the precision of the calculation. If the step size is increased, it leads to a faster calculation of the problem. If it is decreased, the problem is calculated slower and with fewer number of Taylor series terms per step. This behaviour can be seen in Figure 3.6.



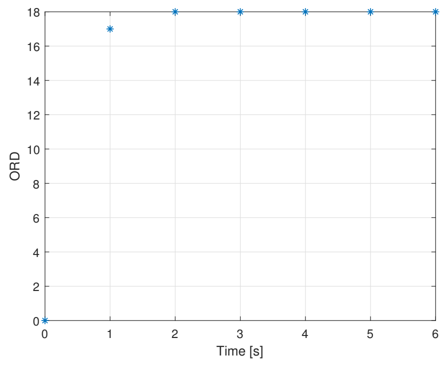
(a) Error function for $h = 0.01$.



(b) The ORD function for $h = 0.01$.



(c) Error function for $h = 1$.



(d) The ORD function for $h = 1$.

Figure 3.6: The comparison between errors of the calculation and values of the ORD function for different values of h.

The accuracy of the calculation is similar. The number of Taylor series terms rises to approximately 19 (up from 9 for $h = 1 \times 10^{-2}\,\mathrm{s}$). The behaviour of this system changes for different values of the parameter ω [rad·s^{-1}]. If (3.29) is modified to

$$\begin{aligned} y' &= \omega z & y(0) &= \sin(0) = 0 \\ z' &= -\omega y & z(0) &= \cos(0) = 1\,, \end{aligned} \tag{3.31}$$

with analytical solution

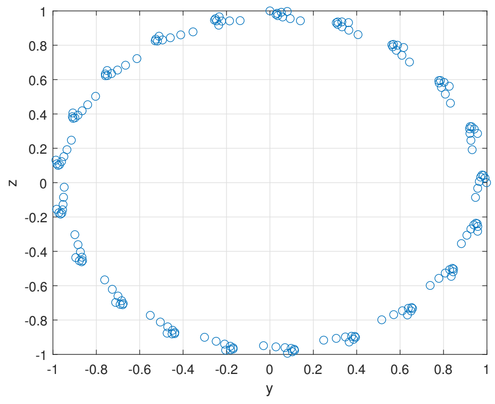
$$\begin{aligned} y &= \sin(\omega t) \\ z &= \cos(\omega t)\,. \end{aligned} \tag{3.32}$$

The system of ODEs is again linear, (3.3) is used with the following values

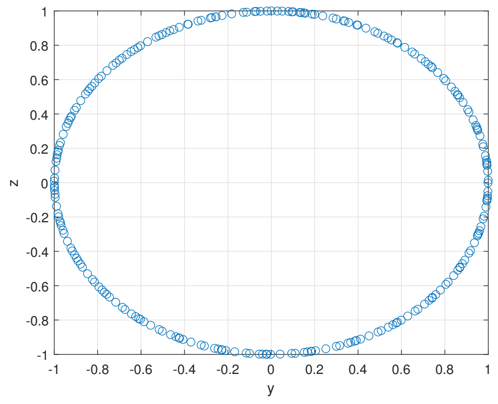
$$\boldsymbol{A} = \begin{pmatrix} 0 & \omega \\ -\omega & 0 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

The different values of ω can be used to compare the performance of MTSM and the state-of-the-art numerical solvers in MATLAB. Note that for all experiments $t_{MAX} = 50\,\mathrm{s}$.

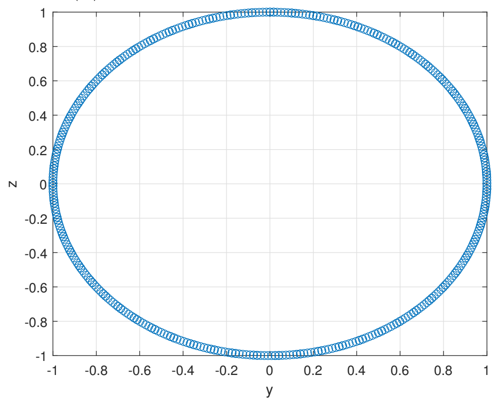
For the first set of experiments, consider $\omega = 1\,\mathrm{rad{\cdot}s^{-1}}$. Tolerances of the state-of-the-art methods are set so that the resulting accuracy matches the set accuracy of MTSM $MTSM_{TOL} = 1 \times 10^{-6}$, that uses step with the size $MTSM_h = 0.1\,\mathrm{s}$. The plot for this system is in Figure 3.7, the plot of the *ORD* function for MTSM solver is in Figure 3.8.



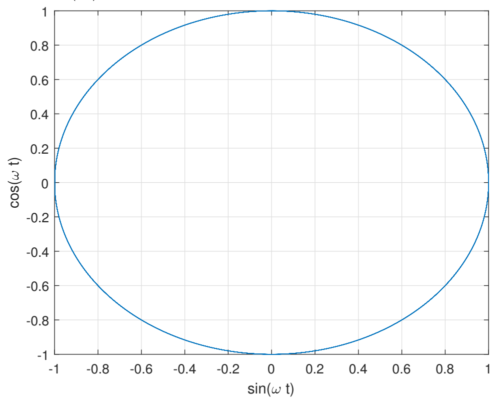
(a) Circle test for ode23 solver.



(b) Circle test for ode45 solver.



(c) Circle test for the MTSM.



(d) Analytical solution for the circle test.

Figure 3.7: Circle tests for commonly used ODE solvers with the provided analytical solution for $\omega = 1\,\mathrm{rad{\cdot}s^{-1}}$, $MTSM_{TOL} = 1 \times 10^{-6}$.

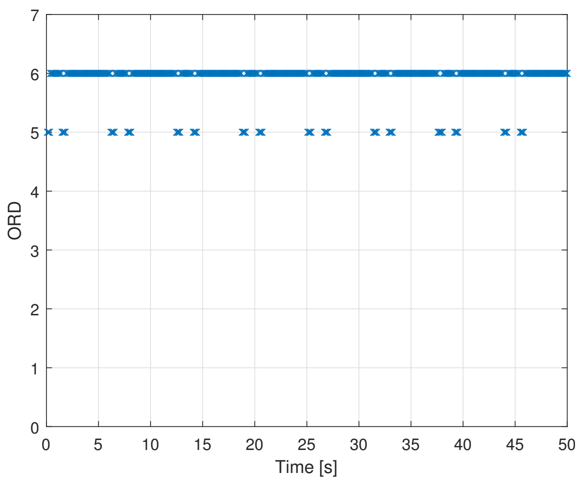


Figure 3.8: Plot of the ORD function for $\omega = 1\,\text{rad}\cdot\text{s}^{-1}$, $MTSM_{TOL} = 1 \times 10^{-6}$.

Note that values in the column Ratio in the following tables are calculated as ratios of computation times $ratio = \frac{ode}{MTSM}$, values in the column $||error||$ are norms of differences between analytical and numerical solutions ($||y_{solver} - y_{analytical}||$) for the entire solution. Several experiments for $\omega = 1\,\text{rad}\cdot\text{s}^{-1}$ were performed. When setting the tolerances of the numerical solvers to their default values $TOL = 1 \times 10^{-3}$, the results are summarized in Table 3.3.

Solver	Number of steps	$\|\|error\|\|$	Time of calculation [s]	Ratio
MTSM	500	7.587×10^{-6}	3.324×10^{-3}	–
ode23	245	3.657×10^{-2}	1.858×10^{-3}	0.56
ode45	277	7.311×10^{-3}	6.730×10^{-4}	0.2
ode113	134	1.110×10^{-2}	2.003×10^{-3}	0.6
ode15s	186	5.706×10^{-2}	1.167×10^{-2}	**3.5**

Table 3.3: Results for $\omega = 1\,\text{rad}\cdot\text{s}^{-1}$, $TOL = 1 \times 10^{-3}$, $MTSM_{TOL} = 1 \times 10^{-6}$.

Table 3.3 shows that the proposed method is the slowest of the selected numerical solvers by a wide margin (except for the ode15s solver). The accuracy of the proposed method is much higher than the requested accuracy (due to how many Taylor series terms smaller than the required accuracy are needed to be calculated). The next experiment will show the impact of equalizing the accuracy between the state-of-the-art solvers and the MTSM solver. This is achieved by setting the tolerances of all state-of-the-art ode solvers to 1×10^{-7}.

Solver	Number of steps	$\|\|error\|\|$	Time of calculation [s]	Ratio
MTSM	500	7.587×10^{-6}	3.328×10^{-3}	–
ode23	1660	8.043×10^{-5}	1.040×10^{-2}	**3.1**
ode45	937	9.032×10^{-6}	1.911×10^{-3}	0.57
ode113	235	1.550×10^{-5}	3.333×10^{-3}	**1**
ode15s	441	2.702×10^{-4}	2.523×10^{-2}	**7.6**

Table 3.4: Results for $\omega = 1\,\text{rad}\cdot\text{s}^{-1}$, accuracy of the ode solvers $TOL = 1 \times 10^{-7}$, $MTSM_{TOL} = 1 \times 10^{-6}$, $MTSM_h = 0.1\,\text{s}$.

Table 3.4 shows that the proposed method is still slower than the ode45 solver but about three times faster than the ode23 solver. MTSM can make up the difference, because the size of the integration step can be increased from its default value ($MTSM_h = 0.1\,\text{s}$) to a different value, for example, $MTSM_h = 10\,\text{s}$. This is not possible using state-of-the-art methods. To maintain the accuracy across all solvers, the accuracy of the MTSM solver was set to $MTSM_{TOL} = 1 \times 10^{-4}$. The performance and accuracy of the calculation with this setting are summarized in Table 3.5.

Solver	Number of steps	$\|\|error\|\|$	Time of calculation [s]	Ratio
MTSM	5	1.039×10^{-5}	1.802×10^{-4}	–
ode23	3575	8.044×10^{-6}	1.693×10^{-2}	**94**
ode45	1485	8.946×10^{-7}	2.393×10^{-3}	**13**
ode113	260	2.563×10^{-6}	3.282×10^{-3}	**18**
ode15s	645	3.851×10^{-5}	9.173×10^{-3}	**51**

Table 3.5: Results for $\omega = 1\,\text{rad}\cdot\text{s}^{-1}$, accuracy of the ode solvers $TOL = 1 \times 10^{-7}$, $MTSM_{TOL} = 1 \times 10^{-4}$, $MTSM_h = 10\,\text{s}$.

Table 3.5 shows that the increase in step size leads to an accurate solution faster than the state-of-the-art solvers. For this setting, the number of used Taylor series terms increases, which can be seen in Figure 3.9.

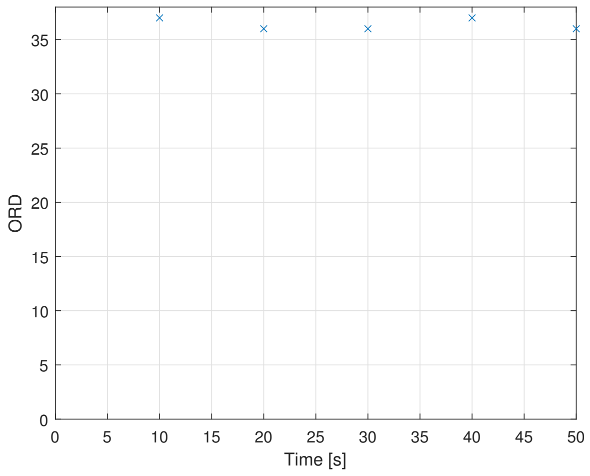
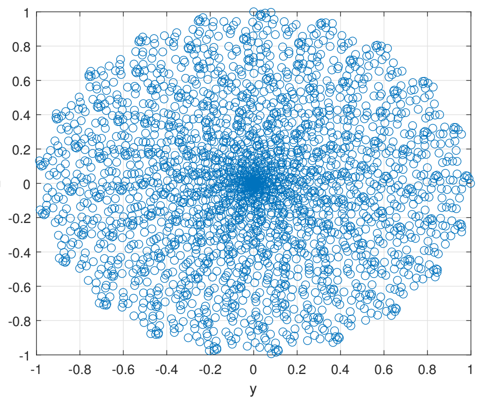
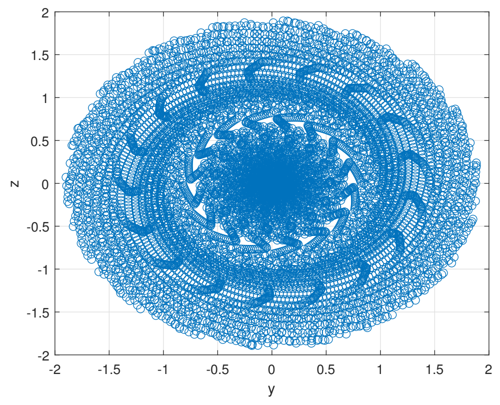


Figure 3.9: ORD function for $MTSM_h = 10\,\text{s}$.

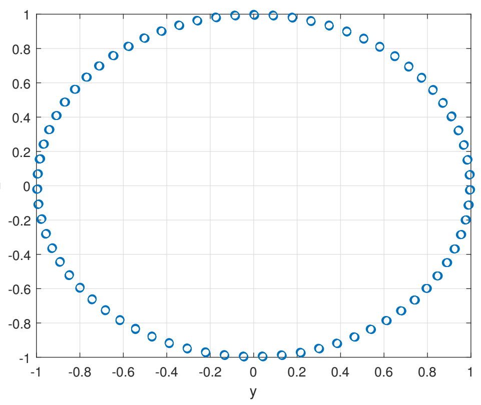
The next set of experiments is going to show the behaviour of the system for $\omega = 100\,\mathrm{rad{\cdot}s^{-1}}$. The first experiment for this value of ω is again with the default tolerances ($TOL = 1 \times 10^{-3}$, $MTSM_{TOL} = 1 \times 10^{-3}$) of all used numerical solvers, with $MTSM_h = 0.1\,\mathrm{s}$. The results are shown in Figure 3.10, the plot of the ORD function is in Figure 3.11.



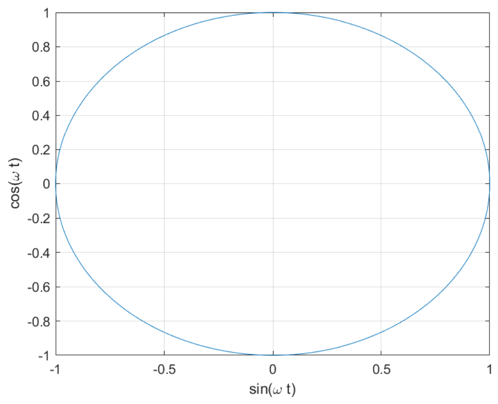
(a) Circle test for ode23 solver.



(b) Circle test for ode15s solver.



(c) Circle test for the Taylor method.



(d) Analytical solution for the circle test.

Figure 3.10: Circle tests for commonly used ODE solvers with the provided analytical solution for $\omega = 100\,\mathrm{rad{\cdot}s^{-1}}$, $TOL = 1 \times 10^{-3}$, $MTSM_{TOL} = 1 \times 10^{-3}$, $MTSM_h = 0.1\,\mathrm{s}$.

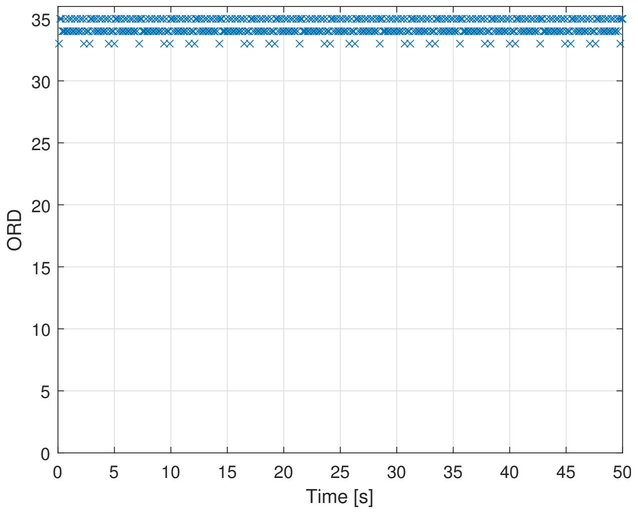


Figure 3.11: The plot of ORD function for $\omega = 100\,\text{rad·s}^{-1}$, $TOL = 1 \times 10^{-3}$, $MTSM_{TOL} = 1 \times 10^{-3}$, $MTSM_h = 0.1\,\text{s}$.

Table 3.6 shows that the accuracy of the state-of-the-art solvers is not acceptable (it is also visible in Figure 3.10).

Solver	Number of steps	$\|\|error\|\|$	Time of calculation [s]	Ratio
MTSM	500	1.218×10^{-2}	1.165×10^{-2}	–
ode23	23912	1.324	1.263×10^{-1}	**11**
ode45	25489	6.594×10^{-1}	3.680×10^{-2}	**3.2**
ode113	12635	5.190×10^{-1}	1.420×10^{-1}	**12**
ode15s	17532	6.429	2.731×10^{-1}	**23**

Table 3.6: Results for $\omega = 100\,\text{rad·s}^{-1}$, $TOL = 1 \times 10^{-3}$, $MTSM_{TOL} = 1 \times 10^{-3}$, $MTSM_h = 0.1\,\text{s}$.

To increase the accuracy to the level similar to the one achieved in the previous experiment, the tolerances of the state-of-the-art ODE solvers have to be increased to $TOL = 1 \times 10^{-10}$, $MTSM_{TOL} = 1 \times 10^{-7}$. The step size remains the same as in the previous experiment $MTSM_h = 0.1\,\text{s}$.

Solver	Number of steps	$\|\|error\|\|$	Time of calculation [s]	Ratio
MTSM	500	5.558×10^{-7}	1.382×10^{-2}	–
ode23	5882740	3.198×10^{-7}	2.776×10^{1}	**2000**
ode45	763977	3.625×10^{-8}	1.124	**81**
ode113	44501	3.044×10^{-8}	5.334×10^{-1}	**39**
ode15s	208852	9.677×10^{-6}	2.943	**210**

Table 3.7: Results for $\omega = 100\,\text{rad·s}^{-1}$, $TOL = 1 \times 10^{-10}$, $MTSM_{TOL} = 1 \times 10^{-7}$, $MTSM_h = 0.1\,\text{s}$.

The order used by MTSM solver is in Figure 3.12.

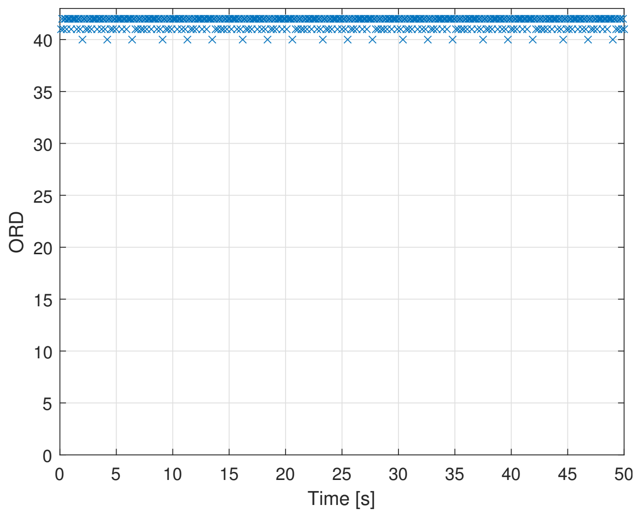


Figure 3.12: Plot of the ORD function for $h = 0.1\,\mathrm{s}$, $\omega = 100\,\mathrm{rad{\cdot}s^{-1}}$, $TOL = 1 \times 10^{-10}$, $MTSM_{TOL} = 1 \times 10^{-7}$.

Figure 3.12 shows that MTSM solver uses approximately 48 terms of the Taylor series for the computation, which approaches the upper limit for the double precision arithmetic (64 terms [45]). Due to this fact, the step size cannot be increased further without decreasing the required accuracy of the calculation. The method allows for computation with arbitrary accuracy and step size if variable-precision arithmetic is used.

3.5.2 Accuracy, error propagation

The accuracy of the proposed method does not depend on the step size. The following example shows this property quite well. It is an ordinary differential equation

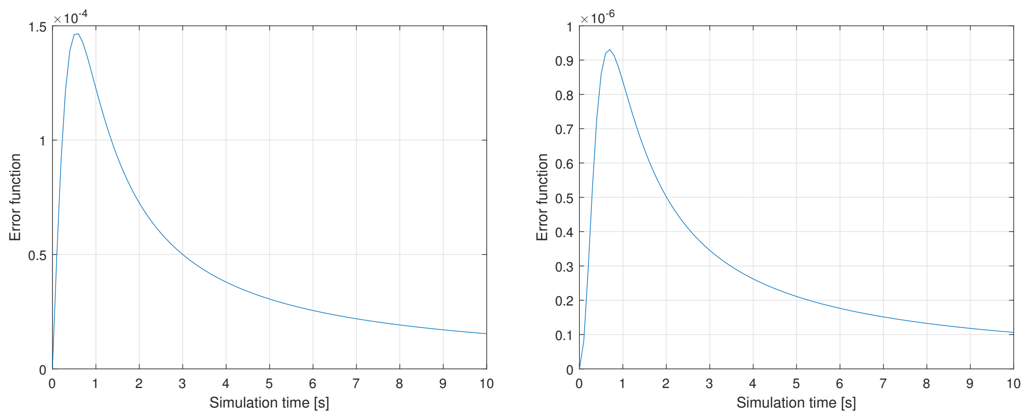
$$y' = \frac{y+t}{y-t}, \qquad y(0) = 1\,, \tag{3.33}$$

with the analytical solution

$$y(t) = t + \sqrt{1 + 2t^2}\,. \tag{3.34}$$

The differences between the numerical and the analytical solution using the (3.34) are in the following set of figures. First, the comparison between the second-order and fourth-order Runge-Kutta methods (with fixed step sizes[3]) and MTSM.

[3]https://www.mathworks.com/matlabcentral/answers/98293-is-there-a-fixed-step-ordinary-differential-equation-ode-solver-in-matlab-8-0-r2012b



(a) Absolute error for the RK2 method. (b) Absolute error for the RK4 method.

Figure 3.13: Absolute errors of the Runge-Kutta methods for the (3.33).

Note that MTSM cannot solve (3.33) directly. The transformations demonstrated in Section 3.4 have to be used. The resulting system of ODEs being solved for (3.33) becomes

$$
\begin{aligned}
y_1' &= y_7 & y_1(0) &= 1\\
y_2' &= y_7 + 1 & y_2(0) &= 1\\
y_3' &= -y_6 + y_5 & y_3(0) &= 1\\
y_4' &= -3y_5y_6 + 3y_5y_5 & y_4(0) &= 1\\
y_5' &= -2y_3y_6 + 2y_3y_5 & y_5(0) &= 1\\
y_6' &= -2y_3y_6 + y_4 + -3y_6y_8 + 3y_5y_8 & y_6(0) &= 1\\
y_7' &= y_3y_7 + y_3 - 2y_2y_6 + y_8 & y_7(0) &= 1\\
y_8' &= y_5y_7 + y_5 - 2y_6y_7 + 2y_5y_7 & y_8(0) &= 1
\end{aligned}
$$

with the following matrix-vector representation.

$$
\boldsymbol{A} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1\\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{pmatrix}
\quad
\boldsymbol{B}_1 = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
-3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & -2 & 2 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & -3 & 3 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & -2
\end{pmatrix}
$$

$$
\boldsymbol{y}_{jk} = \begin{pmatrix}
5 & 6\\
5 & 5\\
3 & 6\\
3 & 5\\
6 & 8\\
5 & 8\\
3 & 7\\
2 & 6\\
5 & 7\\
6 & 7
\end{pmatrix}
$$

For MTSM, the plot of the error function is in Figure 3.14 for $MTSM_{TOL} = 1 \times 10^{-9}$.

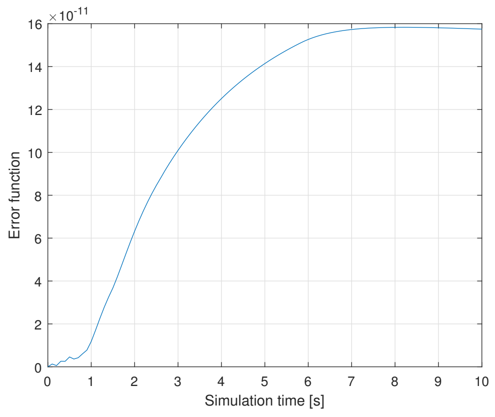


Figure 3.14: Absolute error for the MTSM.

Figure 3.14 shows that the method works very differently with the error and its accumulation during the calculation – it keeps the error bounded near the defined value.

3.5.3 Variable-step-size control

Another important positive property of the method that differentiates it from the commonly used state-of-the-art methods is its independence on the step size. To show how the method handles the variable-step-size, problem (3.33) is again used and is calculated using fixed step size. In the first example, the step size is set to $h = 0.1\,\text{s}$ and $t_{max} = 10\,\text{s}$, $MTSM_{TOL} = 1 \times 10^{-9}$. Figure 3.15 shows the solution for the commonly used solvers with fixed step size $h = 0.1\,\text{s}$. In the following tables, values in the column $||error||$ are calculated as $(||y_{solver} - y_{analytical}||)$.

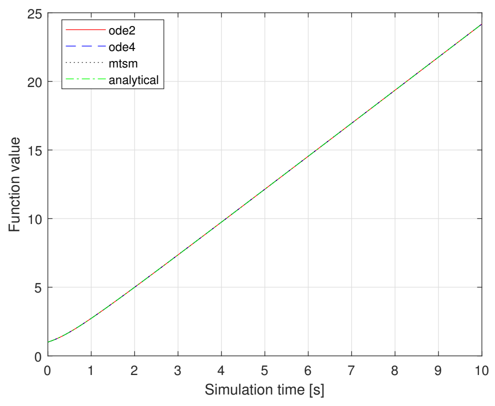


Figure 3.15: Solution for $h = 0.1\,\text{s}$ for state-of-the-art fixed step size solvers and MTSM.

The sizes of errors between the numerical solution and the analytical solution using (3.34) are in Table 3.8.

Solver	$\|\|error\|\|$
rk2	$2.679\,12 \times 10^{-2}$
rk4	$3.735\,96 \times 10^{-6}$
MTSM	$1.271\,79 \times 10^{-9}$

Table 3.8: Comparison of the errors between the analytical and numerical solution for the (3.33) with $h = 0.1$ s.

The second experiment shows the same problem using large step size for all solvers $h = 10$ s using the same tolerances as with the previous example.

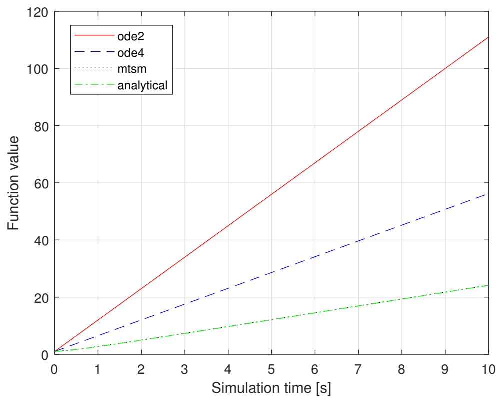


Figure 3.16: Solution using $h = 10$ s the accuracy of state-of-the-art fixed step size solvers and MTSM solver.

Figure 3.16 shows how the accuracy of the state-of-the-art methods depend on step size. The sizes of the error between the numerical solution and the analytical solution (3.34) are in Table 3.9. The mean value of the ORD function is 25.

Solver	$\|\|error\|\|$
rk2	86.8226
rk4	32.0678
MTSM	$5.480\,79 \times 10^{-8}$

Table 3.9: Comparison of the errors between the analytical and numerical solution for the (3.33) with $h = 10$ s.

Perhaps not surprisingly, the unmodified Runge-Kutta method cannot achieve the required accuracy of calculation independent of the step size. MTSM, on the other hand, calculates the accurate solution independent of step size, which makes it potentially interesting in some control applications where sampling times are not overly small.

3.5.4 Stability, convergence

First, let us very briefly talk about the definition of stability and some general remarks. This part of the Chapter is mainly based on [83] and [32]. For the stable numerical method, the value of the next integration step has to be smaller or equal to the current integration step

$$|y_{i+1}| \leq |y_i| \, . \tag{3.35}$$

The stability function $R(z)$ is defined as

$$R(z) = \frac{y_{i+1}}{y_i} \tag{3.36}$$

where $z = h\lambda$. The function is defined as stable in the absolute stability region $\mathcal{D}$, which is defined as

$$\mathcal{D} = \{z \in \mathcal{C} \qquad |R(z)| \leq 1\} \, . \tag{3.37}$$

The $\mathcal{D}(z)$ function is plotted in Figure 3.17.

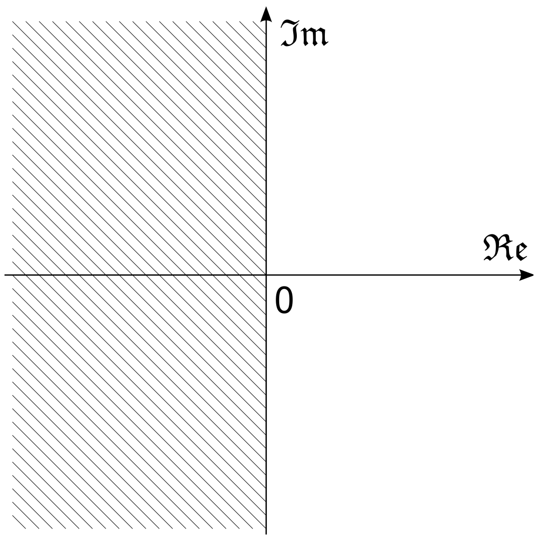


Figure 3.17: Function $\mathcal{D}(z)$.

If the following condition holds

$$\mathcal{D}(z) = \{z \in \mathcal{C}, Re(z) \leq 0\} \tag{3.38}$$

the numerical method is absolutely stable (A-stable). To determine the stability of the numerical methods, the *Dahlquist problem* can be used. It is defined as

$$y' = \lambda y, \qquad y(0) = 0, Re(\lambda) < 0 \, . \tag{3.39}$$

For example, for the Euler method, defined in Subsection 2.3.1 with (2.12), after substituting (2.12) into (3.39)

$$y_{i+1} = y_i + h\lambda y_i = (1 + h\lambda)y_i = (1 + h\lambda)^i y_0 \, , \tag{3.40}$$

where $y(0) = y_0 = 1$. For the stability condition (3.35), the following has to hold

$$|1 + h\lambda| \leq 1 \, . \tag{3.41}$$

The stability region (domain) for the Euler method is in Figure 3.18, the area inside the shape represents the stable region.

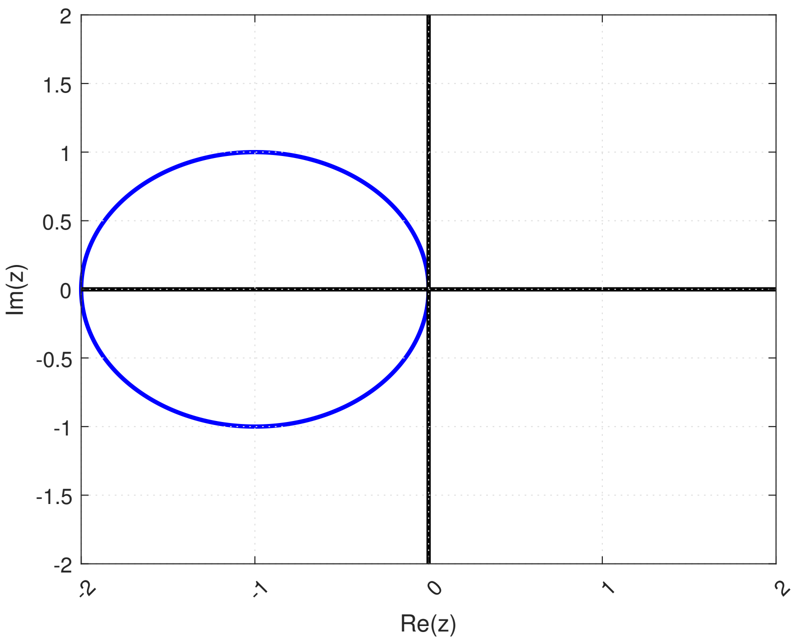


Figure 3.18: Region of stability $\mathcal{D}(z)$ of the Euler method $|1 + z| \leq 1$.

The calculation of the stability region can be applied for the Runge-Kutta and other explicit methods mentioned in Chapter 2. For example, for the second-order Runge-Kutta method, the region of stability expands slightly.

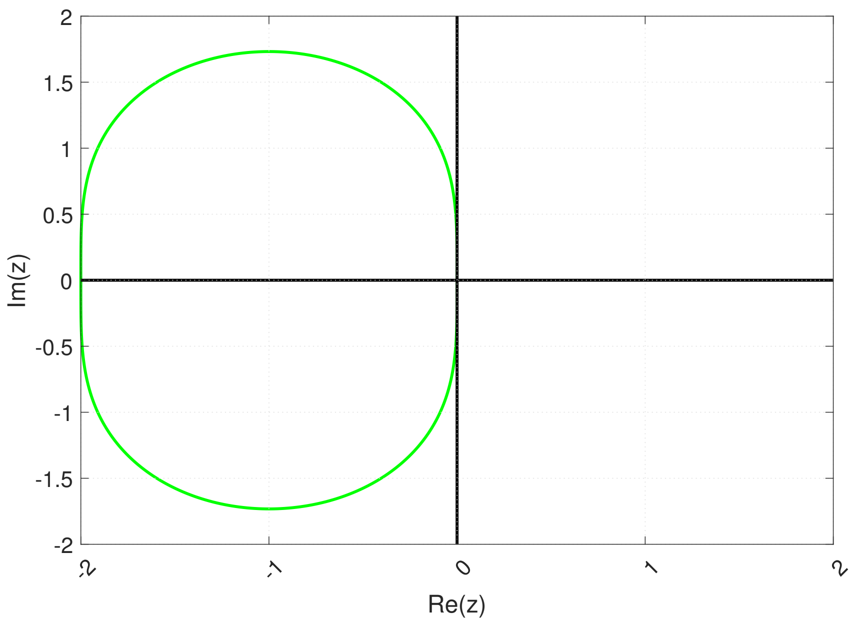


Figure 3.19: Region of stability $\mathcal{D}(z)$ of the Runge-Kutta 2nd order method.

It is obvious from Figures 3.18 and 3.19 that the region of stability for the simplest explicit methods is quite small. The comparison between stability regions of the Euler method and the Runge-Kutta methods is in Figure 3.20.

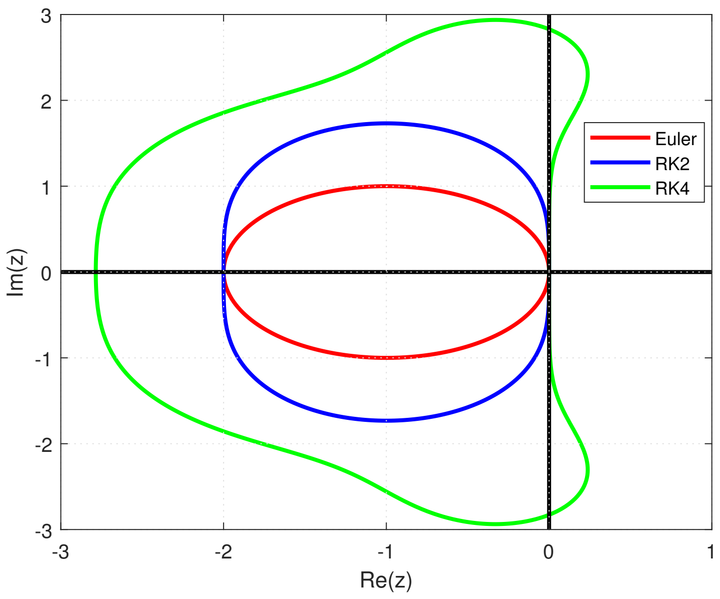


Figure 3.20: Region of stability $\mathcal{D}(z)$ for Runge-Kutta and Euler method.

More complicated methods have bigger regions of stability. For example the Felhberg method or Dormand–Prince method mentioned in Chapter 2 have their regions of stability displayed in Figure 3.21.

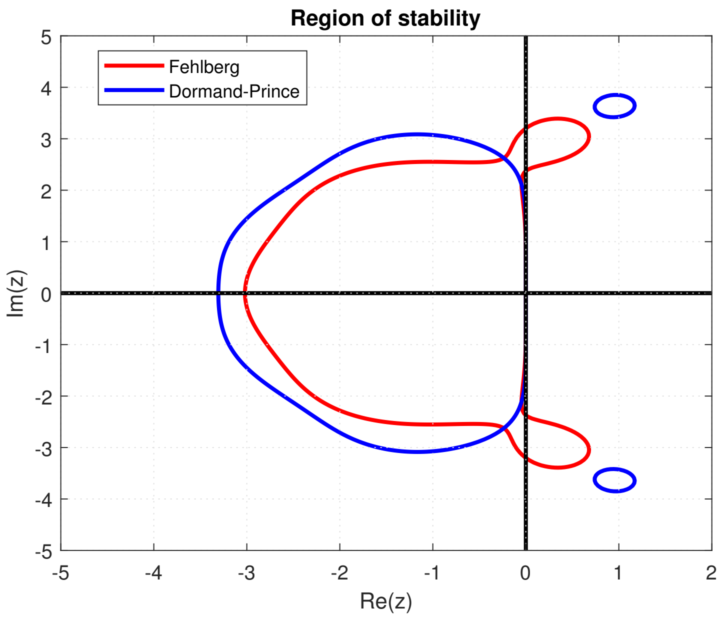


Figure 3.21: Region of stability $\mathcal{D}(z)$ for Runge-Kutta methods with the higher order.

For the variable-order method based on the Taylor series discussed in this Chapter, the stability characteristics are interesting. Using (3.1) and (3.2), we can again analyse the Dahlquist problem. First, the higher derivatives in the explicit Taylor scheme have to be substituted for

$$\begin{aligned} y_{i+1} &= y_i + h\lambda y_i + \frac{h^2}{2!}\lambda^2 y_i + \frac{h^3}{3!}\lambda^3 y_i + \cdots + \frac{h^n}{n!}\lambda^n y_i \\ &= \left(1 + h\lambda + \frac{h^2}{2!}\lambda^2 + \frac{h^3}{3!}\lambda^3 + \cdots + \frac{h^n}{n!}\lambda^n\right) y_i \end{aligned} \tag{3.42}$$

the stability function can be written using (3.42)

$$R(z) = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \cdots + \frac{z^n}{n!}, \tag{3.43}$$

where again $z = h\lambda$, $\lambda \in \mathcal{C}^-$. The stability region $\mathcal{D}$ can be plotted knowing that

$$|1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \cdots + \frac{z^n}{n!}| \leq 1\,.$$

Regions of stability for MTSM using orders 20, 40 and 64 is in Figure 3.22.

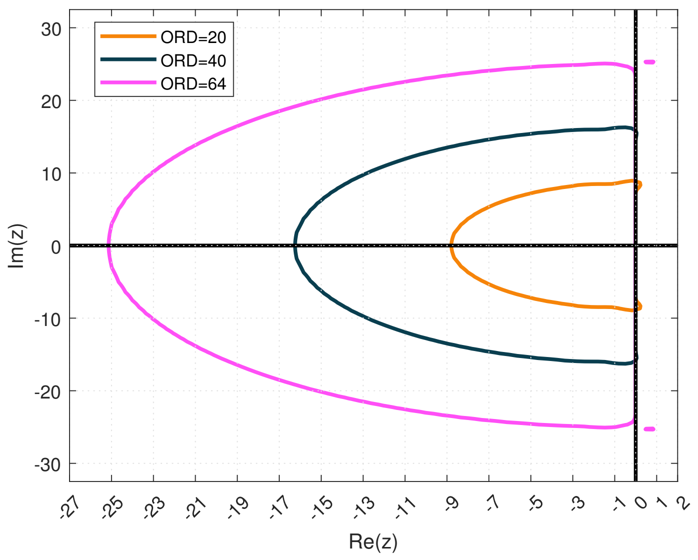


Figure 3.22: Region of stability $\mathcal{D}(z)$ for MTSM with orders 20, 40 and 64.

When comparing Figures 3.20 and 3.22, the region of stability the method can utilize is larger than state-of-the-art methods. Figure 3.22 shows that the area of absolute stability further increases with more terms of the Taylor series used. More terms of the Taylor series might be used when using variable-precision arithmetic.

3.6 Variable-precision arithmetic

The experiments in this thesis always consider the double precision arithmetic (64 bits). The MTSM can utilise the theoretically unlimited number of bits using the variable precision arithmetic schemes. The implementation of such schemes can, for example, be achieved by using the MPFR C library[4], and it is used in existing implementations of the method. The experiments were performed in [46]. To show that the method effectively uses the variable precision arithmetic where available, consider the following IVP:

$$y' = y \qquad y(0) = 1$$

with the analytical solution

$$y = e^t\,,$$

using $h = t_{max} = 1\,\mathrm{s}$ (the method performs one step), $TOL = 1 \times 10^{-15}$. First, consider the results for double precision arithmetic as summarized in Table 3.10. Values in column absolute error are calculated as $|y_1 - y(1)|$.

ORD	Reduced value y(1)	Absolute error
1	2.	$7.182\,818\,284\,590\,452\,35 \times 10^{-1}$
2	2.	$2.182\,818\,284\,590\,452\,35 \times 10^{-1}$
3	2.	$5.161\,516\,179\,237\,868\,3 \times 10^{-2}$
4	2.7	$9.948\,495\,125\,712\,053 \times 10^{-3}$
5	2.71	$1.615\,161\,792\,378\,750 \times 10^{-3}$
6	2.718	$2.262\,729\,034\,898\,66 \times 10^{-4}$
7	2.7182	$2.786\,020\,507\,716\,8 \times 10^{-5}$
8	2.7182	$3.058\,417\,775\,609 \times 10^{-6}$
9	2.718281	$3.028\,858\,531\,76 \times 10^{-7}$
10	2.7182818	$2.731\,266\,091\,1 \times 10^{-8}$
11	2.71828182	$2.260\,552\,523 \times 10^{-9}$
12	2.718281828	$1.728\,768\,24 \times 10^{-10}$
13	2.7182818284	$1.228\,639\,4 \times 10^{-11}$
14	2.71828182845	$8.156\,81 \times 10^{-13}$
15	2.71828182845	5.0959×10^{-14}
16	2.71828182845904	3.220×10^{-15}
17	2.71828182845904	4.44×10^{-16}
18	2.71828182845904	3.33×10^{-16}
19	2.71828182845904	3.33×10^{-16}
20	2.71828182845904	3.33×10^{-16}

Table 3.10: Calculation results using the double data type.

Note that the achieved accuracy is approximately 1×10^{-15}, which is about the same as the maximum achievable accuracy for the double data type. To achieve a better accuracy of calculation, variable precision arithmetic has to be used. The results using 128 bit arithmetic for $TOL = 1 \times 10^{-39}$ are in Table 3.11.

[4] https://www.mpfr.org/

ORD	Reduced value y(1)	Absolute error
1	2.	7.183×10^{-1}
2	2.	2.183×10^{-1}
3	2.7	5.162×10^{-2}
4	2.71	9.948×10^{-3}
5	2.718	1.615×10^{-3}
6	2.7182	2.663×10^{-4}
7	2.7182	2.786×10^{-5}
8	2.718281	3.059×10^{-6}
9	2.7182818	3.029×10^{-7}
10	2.71828182	2.731×10^{-8}
11	2.718281828	2.261×10^{-9}
12	2.7182818284	1.729×10^{-10}
13	2.71828182845	1.229×10^{-11}
14	2.71828182845	8.155×10^{-13}
15	2.71828182845904	5.077×10^{-14}
16	2.718281828459045	2.976×10^{-15}
17	2.7182818284590452	1.648×10^{-16}
18	2.71828182845904523	8.652×10^{-18}
19	2.7182818284590452353	4.315×10^{-19}
20	2.7182818284590452353	2.050×10^{-20}
21	2.7182818284590452353602	9.300×10^{-22}
22	2.71828182845904523536028	4.036×10^{-23}
23	2.7182818284590452353602874	1.679×10^{-24}
24	2.7182818284590452353602874	6.704×10^{-26}
25	2.718281828459045235360287471	2.575×10^{-27}
26	2.7182818284590452353602874713	9.523×10^{-29}
27	2.718281828459045235360287471352	3.397×10^{-30}
28	2.7182818284590452353602874713526	1.170×10^{-31}
29	2.718281828459045235360287471352662	3.896×10^{-33}
30	2.71828182845904523536028747135266249	1.255×10^{-34}
31	2.718281828459045235360287471352662497	3.910×10^{-36}
32	2.7182818284590452353602874713526624977	1.102×10^{-37}
33	2.7182818284590452353602874713526624977	4.408×10^{-39}
34	2.7182818284590452353602874713526624977	4.408×10^{-39}
35	2.7182818284590452353602874713526624977	4.408×10^{-39}

Table 3.11: Calculation results using the variable-precision arithmetic (128 bits).

Note that the number of valid digits increases significantly and the process can be repeated to gain additional accuracy by extending the arithmetic.

3.7 Hardware implementation

This Section covers the basic hardware implementation of the method and shows that the positive properties also translate to the low-level hardware implementation [42]. This Section is based on several theses [86], [53], [62] and the Ph.D. thesis [44].

The most important element of the computation system implemented in hardware is the numerical integrator consisting of computational blocks that perform mathematical and logical operations. As stated previously, the operations that have to be supported in hardware are:

- addition,
- subtraction,
- multiplication,
- division,
- multiplication with integration,
- division with integration.

Used integrators are connected using the interconnection network, which connects the outputs of integrators to different inputs depending on the computational scheme of the differential equation. The output of any integrator can be obtained through the multiplexer. The block scheme of the network is in Figure 3.23.

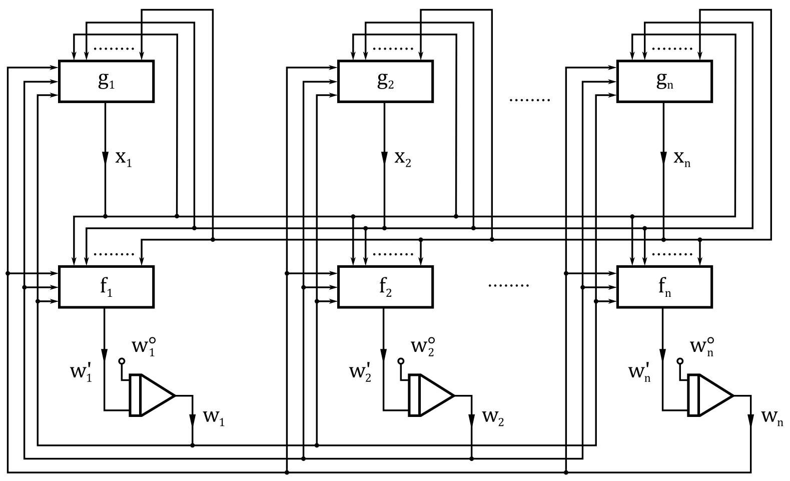


Figure 3.23: Block representation of the integrator network [67].

An example of the usage of the network for the system of ODEs

$$\begin{aligned} x' &= x & x(0) &= x_0 \\ y' &= y + x & y(0) &= y_0 \end{aligned} \tag{3.44}$$

is in Figure 3.24.

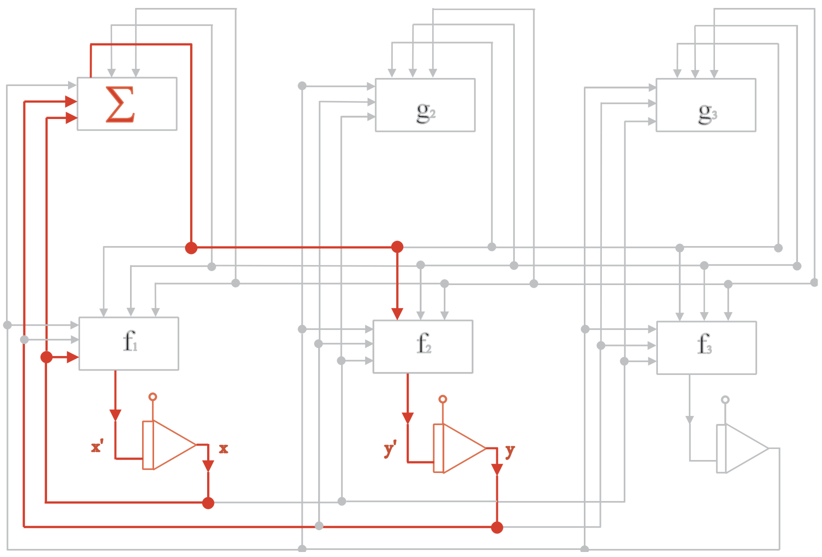


Figure 3.24: Block representation of the integrator network for (3.44) [67].

3.7.1 Operations

This Subsection presents the implementation of needed operations in hardware that use the following computational blocks:

- **RV** – result register,
- **MPX** – multiplexer,
- **SUM** – parallel adder,
- **ACC** – accumulator,
- **MUL** – multiplicator,
- **DIV** – simple divisor,
- **D** – resulting Taylor series terms,
- **A/C**, **B/D** – registers with constant coefficients and
- **CNST** – constant register.

Operation addition

The first operation that will be covered is addition. Derivative of addition $y = u + v$ is

$$y' = u' + v'$$

with initial conditions $y(0) = u_0 + v_0$, $u(0) = u_0$ and $v(0) = v_0$. The Taylor series can then be expressed as a sum of the individual terms

$$y_{i+1} = y_i + DY(1)_i + DY(2)_i + DY(3)_i + \cdots + DY(N)_i\,.$$

The first term of the Taylor series $DY(1)$ can be expressed as

$$\begin{aligned} DY(1) &= h(u' + v') \\ DU(1) &= hu' \\ DV(1) &= hv' \,. \end{aligned} \tag{3.45}$$

The derivatives of u and v can therefore be expressed as

$$\begin{aligned} u' &= \frac{DU(1)}{h} \\ v' &= \frac{DV(1)}{h} \,. \end{aligned} \tag{3.46}$$

The term $DY(1)$ can be calculated using the expressed derivatives

$$DY(1) = DU(1) + DV(1) \,.$$

The hardware representation of addition is in Figure 3.25.

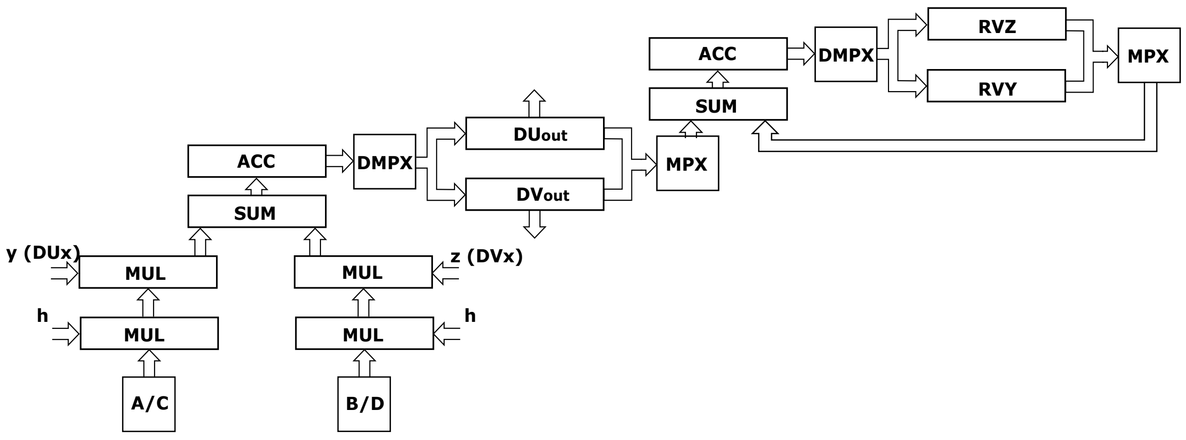


Figure 3.25: Hardware representation of addition [42].

Operation subtraction

Operation subtraction is fundamentally similar to addition, the plus sign is just replaced with the minus sign. The hardware representation and the first derivative is very similar. Just to see the increase in the complexity with higher terms of the Taylor series, the calculation for the second term of the series is

$$\begin{aligned} DY(2) &= \frac{h^2}{2!}(u'' - v'') \\ u'' &= \frac{2!DU(2)}{h^2} \\ v'' &= \frac{2!DV(2)}{h^2} \,. \end{aligned} \tag{3.47}$$

The final equation for the second term of the Taylor series for subtraction

$$DY(2) = DU(2) - DV(2) \,. \tag{3.48}$$

Operation multiplication

The first derivative of multiplication $y = uv$ is

$$y' = u'v + uv'$$

with initial conditions $y(0) = u_0 v_0$, $u(0) = u_0$ and $v(0) = v_0$. The first Taylor series term for multiplication can be expressed using the following equation

$$DY(1) = h(u'v + uv')\,.$$

By substituting the first terms ($DU(1)$ and $DV(1)$ from (3.46)), the following equation for the first term of the Taylor series $DY(1)$ can be obtained

$$DY(1) = h(\frac{DU(1)}{h}v + \frac{DV(1)}{h}u)$$

which can be further simplified as

$$DY(1) = DU(1)v + DV(1)u\,. \tag{3.49}$$

The hardware representation of multiplication is in Figure 3.26.

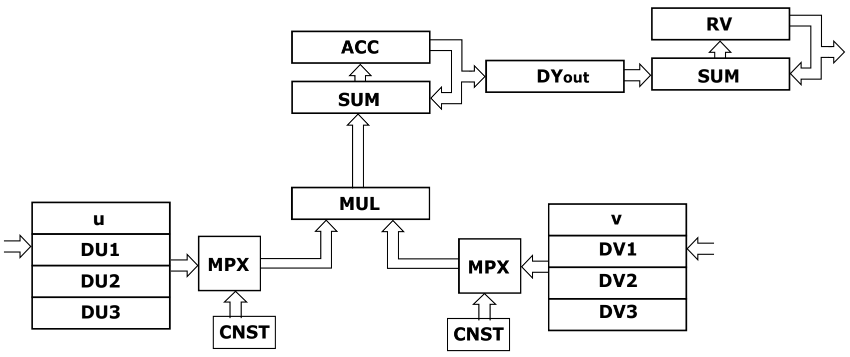


Figure 3.26: Hardware representation of operation multiplication [42].

Operation division

Due to the fact that division can be replaced by multiplication (as shown in Chapter 3.4), the following section just illustrates the basic principle of the possible hardware implementation. The first derivative for division $y = \frac{u}{v}$ is

$$y' = \frac{u'v - uv'}{v^2} \tag{3.50}$$

with initial conditions $y(0) = \frac{u_0}{v_0}$, $u(0) = u_0$, $v(0) = v_0$. Simplifying (3.50)

$$y' = \frac{1}{v}(u' - yv') \tag{3.51}$$

and by substituting the first terms of the Taylor series, we get the following equation

$$y' = h\frac{1}{v}\left(\frac{DU(1)}{h} - y\frac{DV(1)}{h}\right)\,. \tag{3.52}$$

Simplifying (3.52), the final equation for the first term of the Taylor series for division follows

$$y' = \frac{1}{v}(DU(1) - yDV(1))\,. \tag{3.53}$$

Multiplication with integration

The basic equation for integration of multiplication is $y' = uv$. By performing a derivation of the equation, the second derivative is obtained

$$y'' = u'v + uv'$$

with initial conditions $y(0) = y_0$, $u(0) = u_0$ and $v(0) = v_0$. The first and the second Taylor series terms can be expressed using the following equations

$$\begin{aligned} DY(1) &= h(u+v) \\ DY(2) &= \frac{h^2}{2!}\left(\frac{DU(1)}{h}v + \frac{DV(1)}{h}u\right). \end{aligned} \tag{3.54}$$

After simplifying, the equation for the second Taylor series term can be obtained

$$DY(2) = \frac{h}{2}(DU(1)v + DV(1)u). \tag{3.55}$$

Further terms can be obtained similarly. The hardware representation of multiplication with integration using MTSM is in Figure 3.27.

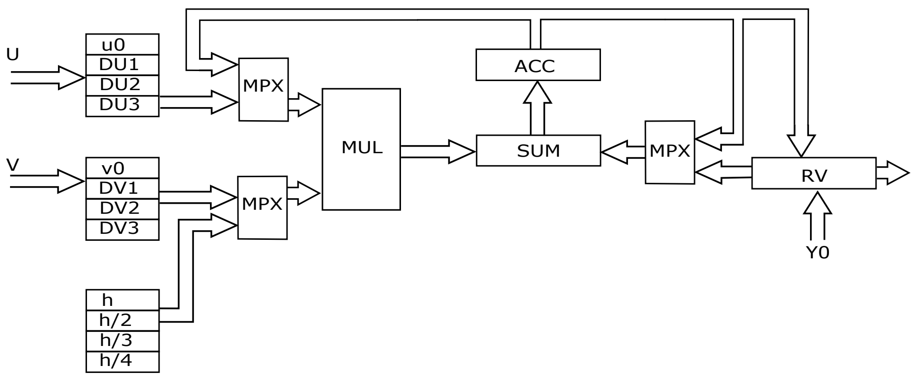


Figure 3.27: MTSM integrator for operation multiplication [42].

Division with integration

Equation for integration of division is $y' = \frac{u}{v}$, deriving it, the second derivative is obtained

$$y'' = \frac{u'v - uv'}{v^2}$$

with initial conditions $y(0) = y_0$, $u(0) = u_0$ and $v(0) = v_0$. Simplifying it, the equation for the second derivative is obtained

$$y'' = \frac{1}{v}(u' - y'v').$$

The first and second Taylor series terms for division with integration can be expressed using the following equations

$$\begin{aligned} DY(1) &= h\frac{u}{v} \\ DY(2) &= \frac{h^2}{2!}\frac{1}{v}\left(\frac{DU(1)}{h} - \frac{DY(1)}{h}\frac{DV(1)}{h}\right). \end{aligned} \tag{3.56}$$

The equation for the second Taylor series term can be further simplified

$$DY(2) = \frac{h}{2}\frac{1}{u}(DU(1) - DY(1)DV(1)) \,. \tag{3.57}$$

Additional terms can be obtained similarly. The hardware representation of the division with integration is in Figure 3.28, which shows a special purpose-built integrator that performs integration using MTSM.

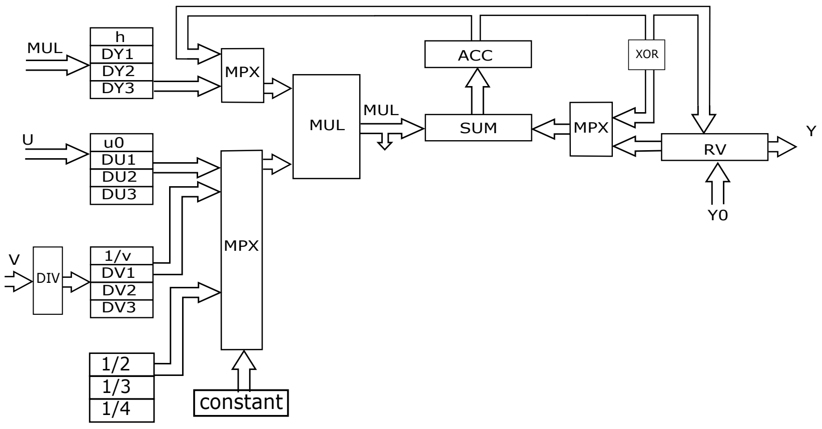


Figure 3.28: MTSM integrator for operation division [42].

3.7.2 Implementation using FPGA

Using the operations described in Subsection 3.7.1, MTSM can be implemented in hardware. The implementation in our research group and as a part of this thesis considered the basic FPGA[5] implementation. The main portion of the method was implemented using the VHDL[6]. The implementation is shown in Figure 3.29.

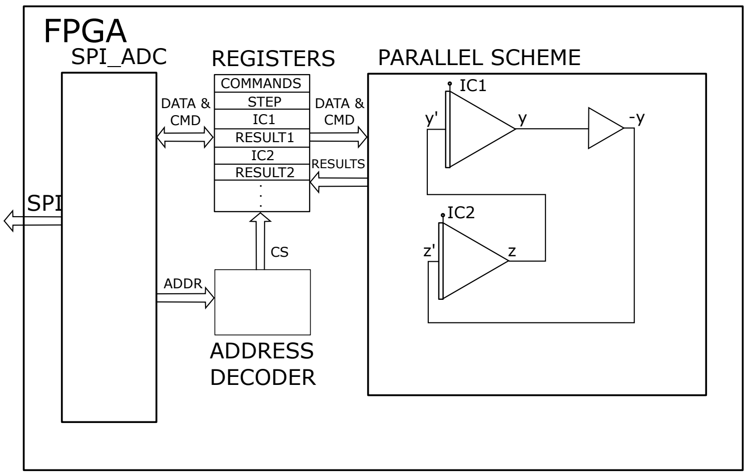


Figure 3.29: MTSM implementation on a FPGA.

The implementation is comprised of two parts:

[5]Field Programmable Gate Array
[6]VHSIC Hardware Description Language

- MCU, which handles parameter set-up and presentation of results,
- FPGA, which handles the calculation itself.

When the MCU is initialized, the initial conditions are loaded into the FPGA registers, followed by the step size. Then the integrator scheme is initialized, and initial conditions are loaded into the integrators, which wait for the `ENABLE` signal, which enables the computation. When the computation ends, the integrators write the results into the appropriate `RESULTx` register (`x` is the index of the result register). After the calculation is finished, the results are copied into MCU using pooling.

3.7.3 Effectiveness of MTSM in hardware

The analysis of performance in hardware was performed by the members of our research group and master students (see [62] or [44]) and published in [42]. To measure the effectiveness of the implementation in hardware, the following linear system of ODEs

$$\begin{aligned} x' &= a_1x + b_1y + c_1z \quad & x(0) = x_0 \\ y' &= a_2x + b_2y + c_2z \quad & y(0) = y_0 \\ z' &= a_3x + b_3y + c_3z \quad & z(0) = z_0\,. \end{aligned} \tag{3.58}$$

Using the mid-point formulation of the second order Runge-Kutta method (2.14), the first derivative can be calculated

$$\begin{aligned} x_1 &= x_0 + \frac{1}{2}(k_{1,x} + k_{2,x}) \\ y_1 &= y_0 + \frac{1}{2}(k_{1,y} + k_{2,y}) \\ z_1 &= z_0 + \frac{1}{2}(k_{1,z} + k_{2,z}) \end{aligned} \tag{3.59}$$

where coefficients k_1 can be expressed as

$$\begin{aligned} k_{1,x} &= h(a_1x_0 + b_1y_0 + c_1z_0) \\ k_{1,y} &= h(a_2x_0 + b_2y_0 + c_2z_0) \\ k_{1,z} &= h(a_3x_0 + b_3y_0 + c_3z_0) \end{aligned} \tag{3.60}$$

and coefficients k_2 can be expressed as

$$\begin{aligned} k_{2,x} &= h(a_1(x_0 + k_{1,x}) + b_1(y_0 + k_{1,y}) + c_1(y_0 + k_{1,y})) \\ k_{2,y} &= h(a_2(x_0 + k_{1,x}) + b_2(y_0 + k_{1,y}) + c_2(y_0 + k_{1,y})) \\ k_{2,z} &= h(a_3(x_0 + k_{1,x}) + b_3(y_0 + k_{1,y}) + c_3(y_0 + k_{1,y}))\,. \end{aligned} \tag{3.61}$$

The systems of equations above show that the mathematical operations in the equivalent equations are equal. These equations can be performed in parallel on computational units that support addition and multiplication. To compare the effectiveness of MTSM and the second order Runge-Kutta method, the parallel operations for the methods are compared in Table 3.12 [44].

	rk2			MTSM		
N.	CPU X	CPU Y	CPU Z	CPU X	CPU Y	CPU Z
1	$X1 = a_1x_0$	$Y1 = a_2x_0$	$Z1 = a_3x_0$	$X1 = a_1x_0$	$Y1 = a_2x_0$	$Z1 = a_3x_0$
2	$X2 = b_1y_0$	$Y2 = b_2y_0$	$Z2 = b_3y_0$	$X2 = b_1y_0$	$Y2 = b_2y_0$	$Z2 = b_3y_0$
3	$X3 = c_1z_0$	$Y3 = c_2z_0$	$Z3 = c_3z_0$	$X3 = c_1z_0$	$Y3 = c_2z_0$	$Z3 = c_3z_0$
4	$X4 = X1 + X2$	$Y4 = Y1 + Y2$	$Z4 = Z1 + Z2$	$X4 = X1 + X2$	$Y4 = Y1 + Y2$	$Z4 = Z1 + Z2$
5	$X5 = X4 + X3$	$Y5 = Y4 + Y3$	$Z5 = Z4 + Z3$	$X5 = X4 + X3$	$Y5 = Y4 + Y3$	$Z5 = Z4 + Z3$
6	$X6 = hX5$	$Y6 = hY5$	$Z6 = hZ5$	$X6 = hX5$	$Y6 = hY5$	$Z6 = hZ5$
7	$X7 = x_0 + X6$	$Y7 = x_0 + X6$	$Z7 = x_0 + X6$	$X7 = a_1X6$	$Y7 = a_2X6$	$Z7 = a_3X6$
8	$X8 = y_0 + Y6$	$Y8 = y_0 + Y6$	$Z8 = y_0 + Y6$	$X8 = b_1Z6$	$Y8 = b_2Y6$	$Z8 = b_3Z6$
9	$X9 = x_0 + Z6$	$Y9 = z0 + Z6$	$Z9 = z_0Z6$	$X9 = c_1Y6$	$Y9 = c2Y6$	$Z9 = c_3Z6$
10	$X10 = a_1X7$	$Y10 = a_2Y7$	$Z10 = a_3Z7$	$X10 = X7 + X8$	$Y10 = Y7 + Y8$	$Z10 = Z7 + Z8$
11	$X11 = b_1X8$	$Y11 = b_2Y8$	$Z11 = b_3Z8$	$X11 = X10 + X9$	$Y11 = Y10 + Y9$	$Z11 = Z10 + Z9$
12	$X12 = c_1X9$	$Y12 = c_2Y9$	$Z12 = c_3Z9$	$X12 = {}^h/_2X11$	$Y12 = {}^h/_2Y11$	$Z12 = {}^h/_2Z11$
13	$X13 = X10 + X11$	$Y13 = Y10 + Y11$	$Z13 = Z10 + Z11$	$X13 = X12 + X6$	$Y13 = Y12 + Y6$	$Z12 + Z6$
14	$X14 = X13 + X12$	$Y14 = Y13 + Y12$	$Z14 = Z13 + Z12$	$X14 = x_0 + X13$	$Y14 = y_0 + Y13$	$Z14 = z_0 + Z13$
15	$X15 = hX14$	$Y15 = hY14$	$Z15 = hZ14$			
16	$X16 = X6 + X15$	$Y16 = Y6 + Y15$	$Z16 = Z6 + Z15$			
17	$X17 = {}^1/_2X16$	$Y17 = {}^1/_2Y16$	$Z17 = {}^1/_2Z16$			
18	$X18 = x_0 + X17$	$Y18 = y_0 + Y17$	$Z18 = z_0 + Z17$			

Table 3.12: Comparison of the operations performed by the second-order Runge-Kutta method and MTSM in hardware [44].

The number of operations is lower for MTSM when calculating the same order (i.e. the same number of the Taylor series terms). The same pattern holds for the higher-order Runge-Kutta methods. For the fourth-order Runge-Kutta method, the CPU performs *46 operations* and *28 operations* for MTSM, which represents a decrease by approximately *60 percent* in the number of performed operations.

To demonstrate the hardware implementation of the method [53], ODE

$$y' = ye^{at}\,, \tag{3.62}$$

with the analytical solution

$$y = e^{\frac{e^{at}}{a}}e^{\frac{-1}{a}} \tag{3.63}$$

that can be transformed into a system of auxiliary ODEs

$$\begin{aligned} y' &= yz \quad y(0) = 1 \\ z' &= az \quad z(0) = 1 \; a < 0\,. \end{aligned} \tag{3.64}$$

For the following experiments, the maximum number of the Taylor series terms was set to 8. The column denoted Error represents the difference between the analytical and the numerical solution of the chosen method $|y_{method}(t_{max}) - y_{analytical}(t_{max})|$. Note that step sizes h for the state-of-the-art numerical methods in the following experiments are set so that the error would be the smallest possible. The first experiment uses $t_{max} = 1\,\mathrm{s}$, $MTSM_h = 1\,\mathrm{s}$.

Solver	h	Error	# of steps	# operations per step	# of operations
Euler	4.4×10^{-4}	2.554×10^{-5}	2273	12	27 276
rk2	3.750×10^{-2}	2.722×10^{-4}	27	25	675
rk4	3.5000×10^{-1}	4.560×10^{-5}	3	41	123
MTSM	1	5.961×10^{-4}	1	60	60

Table 3.13: The results of the experiment on hardware for $t_{max} = 1\,\mathrm{s}$, $MTSM_h = 1\,\mathrm{s}$.

The smallest error was achieved by the Euler method, however the method needed a huge amount of steps to achieve this accuracy. Therefore the time of calculation was very large. For $t_{max} = 1\,\mathrm{s}$, $MTSM_h = 0.5\,\mathrm{s}$, the results are in Table 3.14.

Solver	h	Error	# of steps	# operations per step	# of operations
rk2	2.34×10^{-3}	2.377×10^{-6}	427	25	10 675
rk4	8.580×10^{-2}	4.788×10^{-7}	3	164	492
MTSM	5.000×10^{-1}	1.735×10^{-6}	2	60	120

Table 3.14: The results of the experiment on hardware for $t_{max} = 1\,\mathrm{s}$, $MTSM_h = 0.5\,\mathrm{s}$.

Further information about the experiments can be found in [53]. The eighth-order MTSM performs the least amount of integration steps and is the most accurate. Due to the hardware limitations (i.e. the limited amount of components that can fit on the FPGA), the additional Taylor series terms could not be calculated. The hardware implementation and optimization of the algorithms is going to be a part of future research after completion of this thesis.

3.8 Concluding remarks

This Chapter described the method that is at the core of this thesis in detail for both linear and non-linear systems of ODEs. The positive properties of the method were also introduced. The analysis of the method was performed using the set of general real-world experiments that are detailed in Chapter 4.

Chapter 4

Practical examples with high-order Taylor series method

This Chapter contains several published examples that show the favourable properties of the method while solving real-world linear and non-linear problems. Note that in the tables, columns labelled as *Time of calculation* and *Ratio* are taken as a median value from 100 runs. Ratios of computation times $ratio = ode/MTSM \gg 1$ indicate significantly faster computation using MTSM. The experiments were performed using MATLAB 2021a.

4.1 Numerical solver benchmarks

As the first set of experiments, consider the benchmarks of numerical solvers published in [27]. Some problems from this set of benchmarks (for example, the Kepler problem) are going to be discussed in greater detail in Section 4.7. Some are just interesting to get an idea on how MTSM behaves while solving a wide range of real-world problems. This Section contains just numerical results and complete analysis of the most interesting benchmarks. Additional information is in Appendix E. The maximum time of calculation for all experiments is set to $t_{max} = 20\,\mathrm{s}$, tolerances for all used solvers are set to $TOL = 1 \times 10^{-12}$.

4.1.1 Problem A1

The definition of problem A1 is in Appendix E.1. The numerical results are in Table 4.1 and the plot of the ORD function is in Figure 4.1.

Solver	$\lvert error \rvert$	# of steps	Time of calculation [s]	Ratio
MTSM	–	200	$1.192\,15 \times 10^{-3}$	–
ode23	$1.232\,52 \times 10^{-13}$	22208	$1.365\,03 \times 10^{-1}$	**114.5**
ode45	$1.107\,93 \times 10^{-14}$	2389	$4.615\,45 \times 10^{-3}$	**3.87**
ode113	$2.055\,43 \times 10^{-14}$	161	1.985×10^{-3}	**1.67**

Table 4.1: Results for benchmark problem A1, $h = 0.1\,\mathrm{s}$.

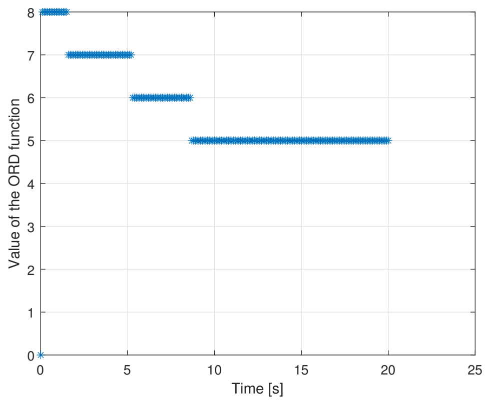


Figure 4.1: *ORD* function for benchmark problem A1, $h = 0.1$ s.

4.1.2 Problem A2

The definition of problem A2 is in Appendix E.2. The numerical results are in Table 4.2.

			MTSM orig		MTSM opt	
Solver	# of steps	Time of calculation [s]	$\lvert error\rvert$	Ratio	$\lvert error\rvert$	Ratio
MTSM orig	40	$1.511\,25 \times 10^{-3}$	–	–	–	–
MTSM opt	40	$1.217\,95 \times 10^{-3}$	–	–	–	–
ode23	28550	$1.780\,09 \times 10^{-1}$	$1.554\,24 \times 10^{-7}$	**117.79**	$1.554\,24 \times 10^{-7}$	**146.2**
ode45	2177	4.0415×10^{-3}	$1.554\,24 \times 10^{-7}$	**2.67**	$1.554\,24 \times 10^{-7}$	**3.32**
ode113	232	$3.214\,05 \times 10^{-3}$	$1.554\,24 \times 10^{-7}$	**2.13**	$1.554\,24 \times 10^{-7}$	**2.64**

Table 4.2: Results for benchmark problem A2, $h = 0.5$ s.

The optimization from Subsection 3.3.3 is used. The change in step size is visible in Figure 4.2.

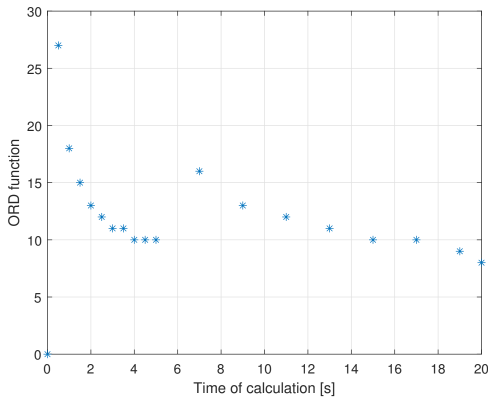


Figure 4.2: *ORD* function for benchmark problem A2, $h = 0.5$ s, $h_{scale} = 4$.

4.1.3 Problem A3

The definition of problem A3 is in Appendix E.3. The numerical results are in Table 4.3.

			MTSM orig		MTSM opt	
Solver	# of steps	Time of calculation [s]	$\|error\|$	Ratio	$\|error\|$	Ratio
MTSM orig	40	$1.957\,35 \times 10^{-3}$	–	–	–	–
MTSM opt	26	$1.253\,35 \times 10^{-3}$	–	–	–	–
ode23	153480	$9.626\,68 \times 10^{-1}$	$9.055\,58 \times 10^{-8}$	**491.82**	$8.311\,61 \times 10^{-9}$	**768.18**
ode45	9381	$1.694\,15 \times 10^{-2}$	$9.055\,93 \times 10^{-8}$	**8.66**	$8.315\,17 \times 10^{-9}$	**13.52**
ode113	570	7.4009×10^{-3}	$9.055\,96 \times 10^{-8}$	**3.78**	$8.315\,46 \times 10^{-9}$	**5.90**

Table 4.3: Results for benchmark problem A3, $h = 0.5\,$s.

The optimization from Subsection 3.3.3 is used. The change in step size is visible in Figure 4.3.

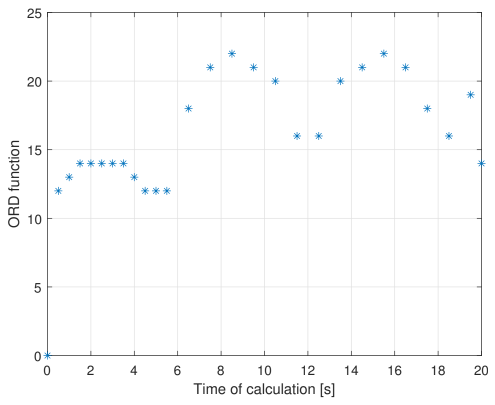


Figure 4.3: *ORD* function for benchmark problem A3, $h = 0.5\,$s, $h_{scale} = 2$.

4.1.4 Problem A4

The definition of the problem *A4* is in Appendix E.4. The numerical results are in Table 4.4, the *ORD* function for $h_{scale} = 4$ is in Figure 4.4.

			MTSM orig		MTSM opt	
Solver	# of steps	Time of calculation [s]	$\|error\|$	Ratio	$\|error\|$	Ratio
MTSM orig	40	1.3478×10^{-3}	–	–	–	–
MTSM opt	25	7.377×10^{-4}	–	–	–	–
ode23	37372	$2.306\,14 \times 10^{-1}$	1.7788×10^{-9}	**171.1**	$1.432\,26 \times 10^{-8}$	**312.61**
ode45	2921	$5.319\,95 \times 10^{-3}$	$1.427\,08 \times 10^{-9}$	**3.95**	$1.397\,08 \times 10^{-8}$	**7.21**
ode113	157	$2.263\,45 \times 10^{-3}$	$1.437\,35 \times 10^{-9}$	**1.68**	$1.398\,11 \times 10^{-8}$	**3.07**

Table 4.4: Results for benchmark problem A4, $h = 0.5\,$s, $h_{scale} = 2$.

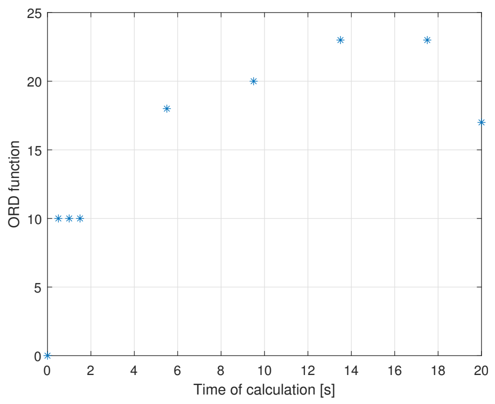


Figure 4.4: *ORD* function for benchmark problem A4, $h = 0.5\,\mathrm{s}$, $h_{scale} = 8$.

4.1.5 Problem A5

Problem A5 is interesting because operation division has to be removed, which leads to the application of automatic transformation (see Section 3.4). It is defined as

$$y' = \frac{y-t}{y+t'} \qquad y(0) = 4\,,$$

where operation division has to be replaced

$$\begin{aligned}
y_1' &= (y_1 - t)(y_1 + t')^{-1} && \\
y_2 &= y_1 - t && \\
y_2' &= y_1' - 1 && y_2(0) = y_1(0) - 0 = 4 \\
y_3 &= (y_1 + t')^{-1} && \\
y_3' &= -(y_1 + t')^{-2}(y_1' + t'') = -y_3^2(y_1' + 0) && y_3(0) = \frac{1}{y_1(0) + 0}\,.
\end{aligned}$$

After substituting and simplifying, the auxiliary system of ODEs representing the system can be written as

$$\begin{aligned}
y_1' &= y_2 y_3 && y_1(0) = 4 \\
y_2' &= y_2 y_3 - 1 && y_2(0) = y1(0) - 0 \\
y_3' &= -y_2 y_3 y_3 y_3 && y_3(0) = \frac{1}{y_1(0) + 0}
\end{aligned}$$

and it can be transformed into the matrix-vector representation (3.19).

$$\boldsymbol{B}_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \qquad \boldsymbol{y}_{jk} = \begin{pmatrix} 2 & 3 \end{pmatrix} \qquad \boldsymbol{B}_3 = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \qquad \boldsymbol{y}_{jklm} = \begin{pmatrix} 2 & 3 & 3 & 3 \end{pmatrix}$$

The maximum time of calculation is set to $t_{max} = 10\,\mathrm{s}$ for this problem. The numerical results are in Table 4.5, the *ORD* function for $h_{scale} = 4$ is in Figure 4.5.

			MTSM orig		MTSM opt	
Solver	# of steps	Time of calculation [s]	$\lvert error \rvert$	Ratio	$\lvert error \rvert$	Ratio
MTSM orig	20	$1.608\,65 \times 10^{-3}$	–	–	–	–
MTSM opt	8	$1.278\,45 \times 10^{-3}$	–	–	–	–
ode23	15460	$9.722\,53 \times 10^{-2}$	9.4959×10^{-9}	**60.44**	$4.989\,74 \times 10^{-8}$	**76.05**
ode45	893	1.8349×10^{-3}	$9.497\,71 \times 10^{-9}$	**1.14**	$4.989\,92 \times 10^{-8}$	**1.44**
ode113	115	$1.798\,85 \times 10^{-3}$	9.4934×10^{-9}	**1.12**	$4.989\,49 \times 10^{-8}$	**1.41**

Table 4.5: Results for benchmark problem A5, $h = 0.5\,\mathrm{s}$.

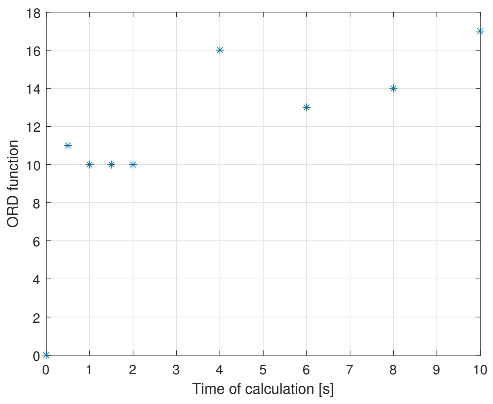


Figure 4.5: *ORD* function for benchmark problem A5, $h = 0.5\,\mathrm{s}$, $h_{scale} = 4$.

4.1.6 Problem B1

The definition of problem B1 is in Appendix E.5. The numerical results are in Table 4.6.

			MTSM orig		MTSM opt	
Solver	# of steps	Time of calculation [s]	$\lvert error \rvert$	Ratio	$\lvert error \rvert$	Ratio
MTSM orig	67	$3.707\,75 \times 10^{-3}$	–	–	–	–
MTSM opt	67	$2.753\,85 \times 10^{-3}$	–	–	–	–
ode23	237231	$1.492\,23$	$1.139\,78 \times 10^{-6}$	**402.46**	$1.139\,78 \times 10^{-6}$	**541.87**
ode45	14125	$2.525\,26 \times 10^{-2}$	$1.139\,77 \times 10^{-6}$	**6.81**	$1.139\,77 \times 10^{-6}$	**9.17**
ode113	917	$1.233\,35 \times 10^{-2}$	$1.139\,77 \times 10^{-6}$	**3.33**	$1.139\,77 \times 10^{-6}$	**4.48**

Table 4.6: Results for benchmark problem B1, $h = 0.3\,\mathrm{s}$.

No step-size scaling can be performed on double arithmetic. The better performance of the optimized solver is due to the additional improvements for two function multiplications. The *ORD* function is in Figure 4.6.

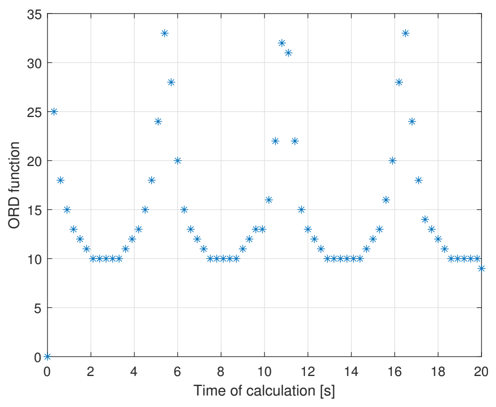


Figure 4.6: *ORD* function for benchmark problem B1, $h = 0.3\,\mathrm{s}$.

4.1.7 Problem B2

The definition of problem B2 is in Appendix E.6. The numerical results are in Table 4.7, the plot of the *ORD* function is in Figure 4.7.

Solver	$\lvert error \rvert$	# of steps	Time of calculation [s]	Ratio
MTSM	–	200	$1.654\,55 \times 10^{-3}$	–
ode23	$1.147\,97 \times 10^{-13}$	28461	$1.805\,74 \times 10^{-1}$	**109.14**
ode45	$2.442\,49 \times 10^{-15}$	3161	$5.979\,95 \times 10^{-3}$	**3.61**
ode113	$2.176\,04 \times 10^{-14}$	265	$3.619\,45 \times 10^{-3}$	**2.19**

Table 4.7: Results for benchmark problem B2, $h = 0.1\,\mathrm{s}$.

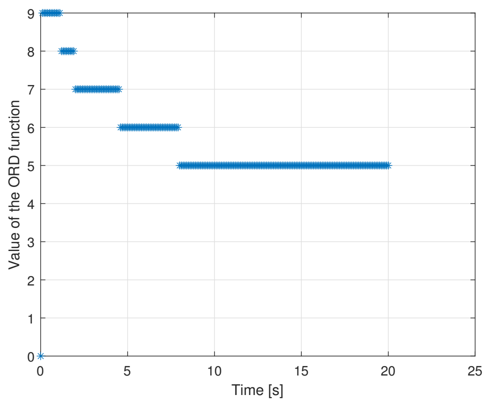


Figure 4.7: *ORD* function for benchmark problem B2, $h = 0.1\,\mathrm{s}$.

4.1.8 Problem B3

The definition of problem B3 is in Appendix E.7. The numerical results are in Table 4.8. The plot of the ORD function with step size scaling factor $h_{scale} = 4$ is in Figure 4.8.

			MTSM orig		MTSM opt	
Solver	# of steps	Time of calculation [s]	$\lvert error \rvert$	Ratio	$\lvert error \rvert$	Ratio
MTSM orig	40	1.2294×10^{-3}	–	–	–	–
MTSM opt	19	6.2215×10^{-4}	–	–	–	–
ode23	30217	1.89301×10^{-1}	9.67091×10^{-14}	**153.98**	2.90649×10^{-11}	**304.27**
ode45	2825	5.34695×10^{-3}	9.63948×10^{-14}	**4.35**	2.90646×10^{-11}	**8.59**
ode113	219	3.1773×10^{-3}	9.33876×10^{-14}	**2.58**	2.90616×10^{-11}	**5.11**

Table 4.8: Results for benchmark problem B3, $h = 0.5\,\mathrm{s}$.

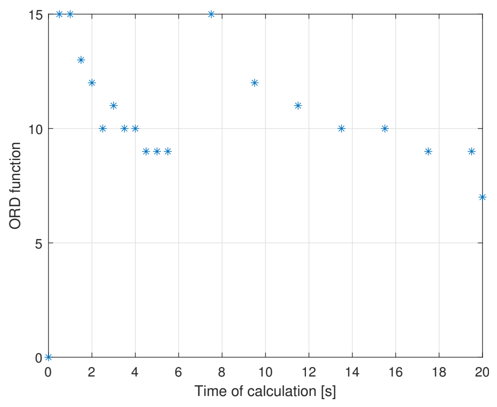


Figure 4.8: ORD function for benchmark problem B3, $h = 0.1\,\mathrm{s}$, $h_{scale} = 4$.

4.1.9 Problem B4

Problem B4 interesting because it contains several non-trivial mathematical operations that have to be replaced using the automatic transformations defined in Section 3.4. The problem is defined as

$$
\begin{aligned}
y_1' &= -y_2 - \frac{y_1 y_3}{\sqrt{y_1^2 + y_2^2}} & y_1(0) &= 3 \\
y_2' &= y_1 - \frac{y_2 y_3}{\sqrt{y_1^2 + y_2^2}} & y_2(0) &= 0 \\
y_3' &= \frac{y_1}{\sqrt{y_1^2 + y_2^2}} & y_3(0) &= 0\,,
\end{aligned}
$$

Several transformations have to be performed on the system for it to be solvable using MTSM. The first is the removal of division from the system using the set of auxiliary equations:

$$\begin{aligned}
y_4 &= (y_1^2 + y_2^2)^{\frac{1}{2}} \\
y_4' &= \frac{1}{2}(y_1^2 + y_2^2)^{-\frac{1}{2}}(2y_1y_1' + 2y_2y_2') && y_4(0) = y_1(0)^2 + y_2(0)^2 \\
y_5 &= \frac{1}{y_4} = y_4^{-1} \\
y_5' &= -y_4^{-2}y_4' = -y_5^2y_4' && y_5(0) = \frac{1}{y_4(0)}
\end{aligned}$$

$$\begin{aligned}
y_4' &= y_5(2y_1y_1' + 2y_2y_2') && y_4(0) = y_1(0)^2 + y_2(0)^2 \\
y_5' &= -y_5^2y_4' && y_5(0) = \frac{1}{y_5(0)}\,.
\end{aligned}$$

After performing the transformations, the original system can be rewritten as

$$\begin{aligned}
y_1' &= -y_2 - y_1y_3y_5 && y_1(0) = 3 \\
y_2' &= y_1 - y_2y_3y_5 && y_2(0) = 0 \\
y_3' &= y_1y_5 && y_3(0) = 0 \\
y_4' &= y_5(2y_1y_1' + 2y_2y_2') = 2y_1y_5y_1' + 2y_2y_5y_2' \\
&= 2y_1y_5\left(-y_2 - y_1y_3y_5\right) + 2y_2y_5\left(y_1 - y_2y_3y_5\right) \\
&= -2y_1y_2y_5 - 2y_1y_1y_3y_5y_5 + 2y_1y_2y_5 - 2y_2y_2y_3y_5y_5 = \\
&= -2y_1y_1y_3y_5y_5 - 2y_2y_2y_3y_5y_5 && y_4(0) = y_1(0)^2 + y_2(0)^2 \\
y_5' &= -y_5^2y_4' = -y_5^2\left(-2y_1y_1y_3y_5y_5 - 2y_2y_2y_3y_5y_5\right) \\
&= 2y_1y_1y_3y_5y_5y_5y_5 + 2y_2y_2y_3y_5y_5y_5y_5 && y_5(0) = \frac{1}{y_5(0)}\,.
\end{aligned} \tag{4.1}$$

The system is solvable using MTSM. The number of operations required (especially in equations y_4 and y_5) is too high (see Subsection 3.3.2). Therefore, the next set of optimizations aims to decrease the number of function multiplications. First, $y_5y_5y_5y_5$ has to be replaced using the following set of ODEs:

$$\begin{aligned}
y_6 &= y_5y_5y_5y_5 = y_5^4 \\
y_6' &= 4y_5^3y_5' && y_6(0) = y_5(0)^4
\end{aligned}$$

$$\begin{aligned}
y_7 &= y_5^3 \\
y_7' &= 3y_5^2y_5' && y_7(0) = y_5(0)^3
\end{aligned}$$

$$\begin{aligned}
y_8 &= y_5^2 \\
y_8' &= 2y_5y_5' && y_8(0) = y_5(0)^2\,.
\end{aligned}$$

The set has to be simplified, and derivatives on the right-hand side substituted:

$$\begin{aligned}
y_5' &= 2y_1y_1y_3y_5y_5y_5y_5 + 2y_2y_2y_3y_5y_5y_5y_5 = 2y_1y_1y_3y_6 + 2y_2y_2y_3y_6 \\
y_6' &= 4y_5^3y_5' = 4y_7(2y_1y_1y_3y_6 + 2y_2y_2y_3y_6) = 8y_1y_1y_3y_6y_7 + 8y_2y_2y_3y_6y_7 \\
y_7' &= 3y_5^2y_5' = 3y_8(2y_1y_1y_3y_6 + 2y_2y_2y_3y_6) = 6y_1y_1y_3y_6y_8 + 6y_2y_2y_3y_6y_8 \\
y_8' &= 2y_5y_5' = 2y_5(2y_1y_1y_3y_6 + 2y_2y_2y_3y_6) = 4y_1y_1y_3y_5y_6 + 4y_2y_2y_3y_5y_6
\end{aligned}$$

Using the new ODEs $y_6 - y_8$ the system (4.1) can be augmented

$$\begin{aligned}
y_1' &= -y_2 - y_1y_3y_5 & y_1(0) &= 3 \\
y_2' &= y_1 - y_2y_3y_5 & y_2(0) &= 0 \\
y_3' &= y_1y_5 & y_3(0) &= 0 \\
y_4' &= -2y_1y_1y_3y_8 - 2y_2y_2y_3y_8 & y_4(0) &= y_1(0)^2 + y_2(0)^2 \\
y_5' &= 2y_1y_1y_3y_6 + 2y_2y_2y_3y_6 & y_5(0) &= \frac{1}{y_4(0)} \\
y_6' &= 8y_1y_1y_3y_6y_7 + 8y_2y_2y_3y_6y_7 & y_6(0) &= y_5(0)^4 \\
y_7' &= 6y_1y_1y_3y_6y_8 + 6y_2y_2y_3y_6y_8 & y_7(0) &= y_5(0)^3 \\
y_8' &= 4y_1y_1y_3y_5y_6 + 4y_2y_2y_3y_5y_6 & y_8(0) &= y_5(0)^2 \,.
\end{aligned} \tag{4.2}$$

Further, the y_1y_1 and y_2y_2 can be simplified:

$$\begin{aligned}
y_9 &= y_1y_1 = y_1^2 \\
y_9' &= 2y_1^2y_1' = 2y_1y_1(-y_2 - y_1y_3y_5) = -2y_1y_1y_2 - 2y_1y_1y_1y_3y_5 = \\
&= -2y_9 - 2y_1y_3y_5y_9 \qquad\qquad y_9(0) = y_1(0)^2
\end{aligned}$$

$$\begin{aligned}
y_{10} &= y_2^2 \\
y_{10}' &= 2y_2y_2' = 2y_2(y_1 - y_2y_3y_5) = 2y_1y_2 - 2y_2y_2y_3y_5 = \\
&= 2y_1y_2 - 2y_3y_5y_{10} \qquad\qquad y_{10}(0) = y_2(0)^2
\end{aligned}$$

and the final system of auxiliary equations is

$$\begin{aligned}
y_1' &= -y_2 - y_1y_3y_5 & y_1(0) &= 3 \\
y_2' &= y_1 - y_2y_3y_5 & y_2(0) &= 0 \\
y_3' &= y_1y_5 & y_3(0) &= 0 \\
y_4' &= -2y_3y_8y_9 - 2y_3y_8y_{10} & y_4(0) &= y_1(0)^2 + y_2(0)^2 \\
y_5' &= 2y_3y_6y_9 + 2y_3y_6y_{10} & y_5(0) &= \frac{1}{y_4(0)} \\
y_6' &= 8y_3y_6y_7y_9 + 8y_3y_6y_7y_{10} & y_6(0) &= y_5(0)^4 \\
y_7' &= 6y_3y_6y_8y_9 + 6y_3y_6y_8y_{10} & y_7(0) &= y_5(0)^3 \\
y_8' &= 4y_3y_5y_6y_9 + 4y_3y_5y_6y_{10} & y_8(0) &= y_5(0)^2 \\
y_9' &= -2y_9 - 2y_1y_3y_5y_9 & y_9(0) &= y_1(0)^2 \\
y_{10}' &= 2y_1y_2 - 2y_3y_5y_{10} & y_{10}(0) &= y_2(0)^2 \,.
\end{aligned} \tag{4.3}$$

Further optimizations might be performed. For example, the three-term function multiplication $y_1y_3y_5$ can be replaced by the following system of ODEs:

$$\begin{aligned}
y_{11} &= y_1y_3y_5 \\
y_{11}' &= y_1'y_3y_5 + y_1y_3'y_5 + y_1y_3y_5' \\
&= y_3y_5(-y_2 - y_1y_3y_5) + y_1y_5y_1y_5 + y_1y_3(2y_3y_6y_9 + 2y_3y_6y_{10}) \\
&= -y_2y_3y_5 - y_3y_5y_{11} + y_8y_9 + 2y_1y_3y_3y_6y_9 + 2y_1y_3y_3y_6y_{10} \qquad y_{11}(0) = y_1(0)y_3(0)y_5(0) \\
y_{12} &= y_3^2 \\
y_{12}' &= 2y_3y_3' = 2y_3(y_1y_5) = 2y_1y_3y_5 = 2y_{11} \qquad\qquad y_{12}(0) = y_3(0)^2 \,.
\end{aligned}$$

Adding the two new equations into the system (4.3) and substituting:

$$
\begin{aligned}
y_1' &= -y_2 - y_{11} & y_1(0) &= 3 \\
y_2' &= y_1 - y_2y_3y_5 & y_2(0) &= 0 \\
y_3' &= y_1y_5 & y_3(0) &= 0 \\
y_4' &= -2y_3y_8y_9 - 2y_3y_8y_{10} & y_4(0) &= y_1(0)^2 + y_2(0)^2 \\
y_5' &= 2y_3y_6y_9 + 2y_3y_6y_{10} & y_5(0) &= \frac{1}{y_4(0))} \\
y_6' &= 8y_3y_6y_7y_9 + 8y_3y_6y_7y_{10} & y_6(0) &= y_5(0)^4 \\
y_7' &= 6y_3y_6y_8y_9 + 6y_3y_6y_8y_{10} & y_7(0) &= y_5(0)^3 \\
y_8' &= 4y_3y_5y_6y_9 + 4y_3y_5y_6y_{10} & y_8(0) &= y_5(0)^2 \\
y_9' &= -2y_9 - 2y_9y_{11} & y_9(0) &= y_1(0)^2 \\
y_{10}' &= 2y_1y_2 - 2y_3y_5y_{10} & y_{10}(0) &= y_2(0)^2 \\
y_{11}' &= -y_2y_3y_5 - y_3y_5y_{11} + y_8y_9 + 2y_1y_6y_9y_{12} + 2y_1y_6y_{10}y_{12} & y_{11}(0) &= y_1(0)y_3(0)y_5(0) \\
y_{12}' &= 2y_{11} & y_{12}(0) &= y_3(0)^2\,.
\end{aligned}
\tag{4.4}
$$

Note that further modifications are possible. The matrix-vector notation for (4.4) is in Appendix E.8. The numerical results are in Table 4.9. The plot of the *ORD* function with step size scaling factor set to $h_{scale} = 2.5$ is in Figure 4.9.

			MTSM orig		MTSM opt	
Solver	# of steps	Time of calculation [s]	$\lvert error\rvert$	Ratio	$\lvert error\rvert$	Ratio
MTSM orig	40	$1.020\,28 \times 10^{-2}$	–	–	–	–
MTSM opt	19	6.9581×10^{-3}	–	–	–	–
ode23	282708	1.835 37	$7.459\,31 \times 10^{-8}$	**179.89**	$8.398\,71 \times 10^{-8}$	**263.78**
ode45	11773	$2.454\,57 \times 10^{-2}$	$7.459\,42 \times 10^{-8}$	**2.41**	$8.398\,69 \times 10^{-8}$	**3.53**
ode113	468	$7.174\,75 \times 10^{-3}$	$7.459\,43 \times 10^{-8}$	0.7	$8.398\,69 \times 10^{-8}$	**1.03**

Table 4.9: Results for benchmark problem B4, $h = 0.5$ s.

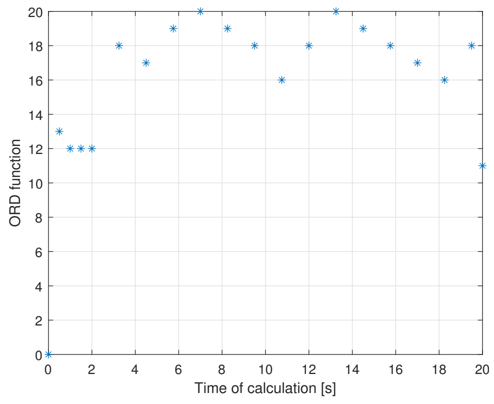


Figure 4.9: *ORD* function for benchmark problem B4, $h = 0.1$ s, $h_{scale} = 2.5$.

4.1.10 Problem B5

The definition of problem B5 is in Appendix E.9. The numerical results are in Table 4.10. The plot of the *ORD* function with step size scaling factor $h_{scale} = 2$ is in Figure 4.10.

			MTSM orig		MTSM opt	
Solver	# of steps	Time of calculation [s]	$\|error\|$	Ratio	$\|error\|$	Ratio
MTSM orig	40	2.3603×10^{-3}	–	–	–	–
MTSM opt	25	$1.802\,35 \times 10^{-3}$	–	–	–	–
ode23	126546	$7.859\,04 \times 10^{-1}$	$9.609\,63 \times 10^{-8}$	**332.97**	$9.609\,64 \times 10^{-8}$	**436.04**
ode45	8157	$1.475\,09 \times 10^{-2}$	$9.609\,52 \times 10^{-8}$	**6.25**	$9.609\,52 \times 10^{-8}$	**8.18**
ode113	577	$7.398\,95 \times 10^{-3}$	$9.609\,48 \times 10^{-8}$	**3.13**	$9.609\,48 \times 10^{-8}$	**4.11**

Table 4.10: Results for benchmark problem B5, $h = 0.5\,\mathrm{s}$.

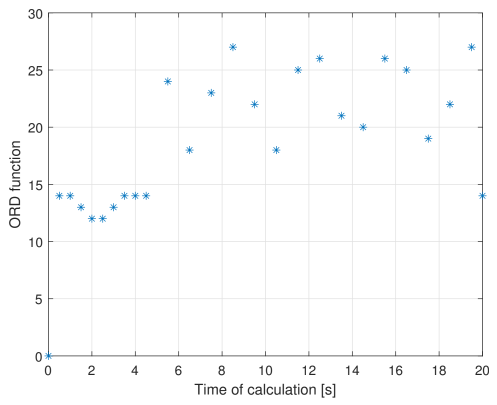


Figure 4.10: *ORD* function for benchmark problem B5, $h = 0.5\,\mathrm{s}$, $h_{scale} = 2$.

4.1.11 Problem C3

The definition of problem C3 is in Appendix E.10. The numerical results are in Table 4.11 and the plot of the *ORD* function is in Figure 4.11.

Solver	$\|error\|$	# of steps	Time of calculation [s]	Ratio
MTSM	–	20	3.02×10^{-4}	–
ode23	$3.503\,22 \times 10^{-10}$	27753	$1.576\,28 \times 10^{-1}$	**521.95**
ode45	$3.503\,72 \times 10^{-10}$	3085	5.532×10^{-3}	**18.32**
ode113	$3.503\,72 \times 10^{-10}$	279	$3.863\,65 \times 10^{-3}$	**12.79**

Table 4.11: Results for benchmark problem C3, $h = 1\,\mathrm{s}$.

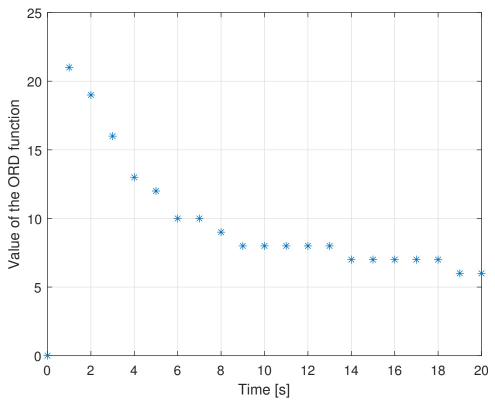


Figure 4.11: *ORD* function for benchmark problem C3, $h = 1\,\mathrm{s}$.

Problem E1

The definition of problem E1 is in Appendix E.11. The numerical results are in Table 4.12. The plot of the *ORD* function with step size scaling factor $h_{scale} = 3$ is in Figure 4.12.

			MTSM orig		MTSM opt	
Solver	# of steps	Time of calculation [s]	$\lvert error\rvert$	Ratio	$\lvert error\rvert$	Ratio
MTSM orig	29	$3.341\,85 \times 10^{-3}$	–	–	–	–
MTSM opt	15	1.9447×10^{-3}	–	–	–	–
ode23	93321	$5.825\,26 \times 10^{-1}$	$5.598\,66 \times 10^{-9}$	**174.31**	$6.750\,65 \times 10^{-9}$	**299.55**
ode45	7309	$1.357\,69 \times 10^{-2}$	$5.599\,61 \times 10^{-9}$	**4.06**	6.7516×10^{-9}	**6.98**
ode113	355	$4.756\,65 \times 10^{-3}$	$5.599\,72 \times 10^{-9}$	**1.42**	$6.751\,71 \times 10^{-9}$	**2.45**

Table 4.12: Results for benchmark problem E1, $h = 0.7\,\mathrm{s}$.

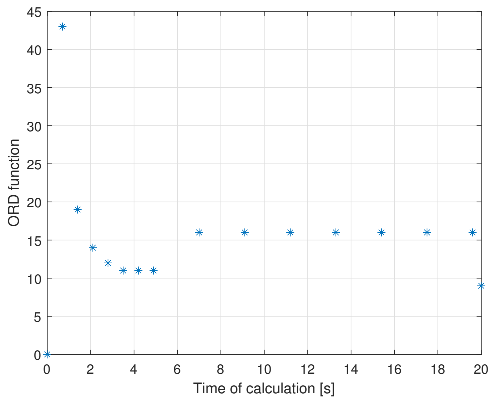


Figure 4.12: *ORD* function for benchmark problem E1, $h = 0.7\,\mathrm{s}$, $h_{scale} = 3$.

This problem is particularly suited for step size scaling because it requires a large number of Taylor series terms in during the first integration step, and the needed number drops sharply afterwards.

4.1.12 Problem E2

The definition of problem E2 is in Appendix E.12. The numerical results are in Table 4.13 and the plot of the *ORD* function is in Figure 4.13.

			MTSM orig		MTSM opt	
Solver	# of steps	Time of calculation [s]	$\|error\|$	Ratio	$\|error\|$	Ratio
MTSM orig	50	6.3573×10^{-3}	–	–	–	–
MTSM opt	50	5.1669×10^{-3}	–	–	–	–
ode23	288872	1.779 51	$3.798\,35 \times 10^{-8}$	**279.921**	$3.798\,35 \times 10^{-8}$	**344.41**
ode45	16801	$2.974\,05 \times 10^{-2}$	$3.798\,35 \times 10^{-8}$	**4.68**	$3.798\,35 \times 10^{-8}$	**5.76**
ode113	1063	$1.369\,03 \times 10^{-2}$	$3.798\,35 \times 10^{-8}$	**2.15**	$3.798\,35 \times 10^{-8}$	**2.65**

Table 4.13: Results for benchmark problem E2 , $h = 0.4$ s.

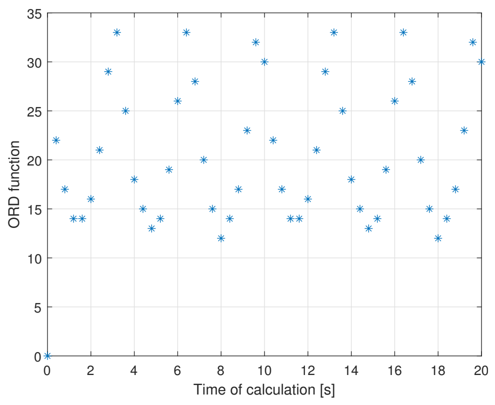


Figure 4.13: *ORD* function for benchmark problem E2, $h = 0.4$ s.

4.1.13 Problem E3

The definition of problem E3 is in Appendix E.13. The numerical results are in Table 4.14 and the plot of the *ORD* function is in Figure 4.14.

			MTSM orig		MTSM opt	
Solver	# of steps	Time of calculation [s]	$\|error\|$	Ratio	$\|error\|$	Ratio
MTSM orig	29	$3.775\,55 \times 10^{-3}$	–	–	–	–
MTSM opt	29	3.2215×10^{-3}	–	–	–	–
ode23	398190	2.477 89	$3.107\,29 \times 10^{-7}$	**656.3**	$3.107\,29 \times 10^{-7}$	**769.17**
ode45	26125	4.6268×10^{-2}	$3.107\,31 \times 10^{-7}$	**12.25**	$3.107\,31 \times 10^{-7}$	**14.36**
ode113	786	$9.832\,35 \times 10^{-3}$	$3.107\,31 \times 10^{-7}$	**2.60**	$3.107\,31 \times 10^{-7}$	**3.05**

Table 4.14: Results for benchmark problem E4, $h = 0.7$ s.

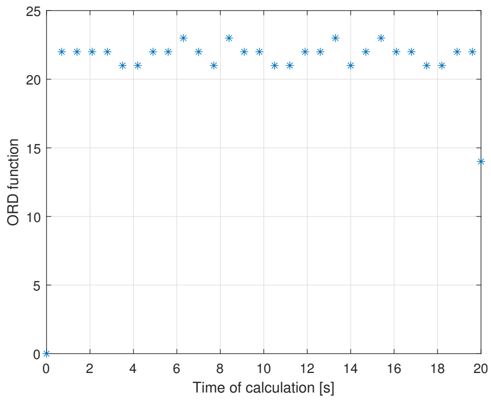


Figure 4.14: *ORD* function for benchmark problem E3, $h = 0.7\,\mathrm{s}$.

4.1.14 Problem E4

The definition of problem E4 is in Appendix E.14. The numerical results are in Table 4.15. The plot of the *ORD* function with step size scaling factor $h_{scale} = 4$ is in Figure 4.15.

			MTSM orig		MTSM opt	
Solver	# of steps	Time of calculation [s]	$\lvert error \rvert$	Ratio	$\lvert error \rvert$	Ratio
MTSM orig	40	$1.041\,05 \times 10^{-3}$	–	–	–	–
MTSM opt	13	3.757×10^{-4}	–	–	–	–
ode23	9038	$5.601\,32 \times 10^{-2}$	$5.200\,11 \times 10^{-10}$	**53.80**	$3.957\,12 \times 10^{-9}$	**149.09**
ode45	877	1.8017×10^{-3}	$5.212\,04 \times 10^{-10}$	**1.73**	$3.955\,93 \times 10^{-9}$	**4.8**
ode113	90	$1.474\,75 \times 10^{-3}$	$5.206\,93 \times 10^{-10}$	**1.42**	$3.956\,44 \times 10^{-9}$	**3.93**

Table 4.15: Results for benchmark problem E4, $h = 0.5\,\mathrm{s}$.

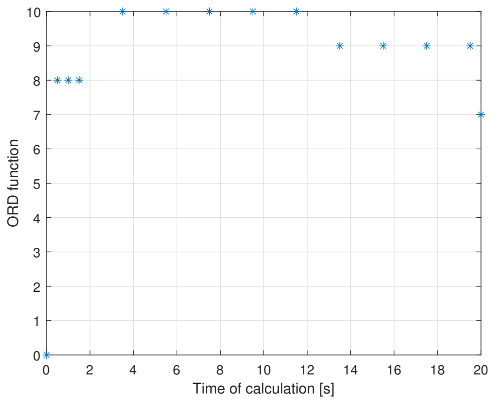


Figure 4.15: *ORD* function for benchmark problem E4, $h = 0.7\,\mathrm{s}$.

4.1.15 Problem E5

Problem E5 is interesting because it contains operation division and square root. These operations have to be removed using the automatic transformation defined in Section 3.4. The problem is defined as

$$\begin{aligned} y_1' &= y_2 & y_1(0) &= 0 \\ y_2' &= \frac{\sqrt{1+y_2y_2}}{25-t} & y_2(0) &= 0. \end{aligned} \tag{4.5}$$

The system (4.5) cannot be transformed into the matrix-vector notation as is, due to the fact that it contains operation division $\frac{1}{25-t}$ and the square root $\sqrt{1+y_2y_2}$. Both of these have to be removed by a system of auxiliary ODEs:

$$\begin{aligned} y_3 &= 1/(25-t) = (25-t)^{-1} & & \\ y_3' &= -(25-t)^{-2}(-1) = y_3y_3 & y_3(0) &= \frac{1}{25} \\ y_4 &= (1+y_2*y_2)^{\frac{1}{2}} & & \\ y_4' &= \frac{1}{2}(1+y_2y_2)^{\frac{1}{2}}(2y_2y_2') = y_4^{-1}y_2y_2' & y_4(0) &= \sqrt{1+y_2(0)^2} \\ y_5 &= \frac{1}{y_4} = y_4^{-1} & & \\ y_5' &= -y_4^{-2}y_4' = -y_5^2y_4' & y_5(0) &= \frac{1}{y_4(0)}. \end{aligned}$$

After substituting and simplifying, the system without operation division and square root is in (4.6)

$$\begin{aligned} y_1' &= y_2 & y_1(0) &= 0 \\ y_2' &= y_3y_4 & y_2(0) &= 0 \\ y_3' &= y_3y_3 & y_3(0) &= \frac{1}{25} \\ y_4' &= y_2y_3y_4y_5 & y_4(0) &= \sqrt{1+y_2(0)^2} \\ y_5' &= -y_2y_3y_4y_5y_5y_5 & y_5(0) &= \frac{1}{y_4(0)}. \end{aligned} \tag{4.6}$$

The system (4.6) is solvable using MTSM. However, the number of function multiplications it contains would make the solution very slow. Therefore, further optimizations are needed

$$\begin{aligned} y_6 &= y_5y_5y_5 = y_5^3 & & \\ y_6' &= 3y_5^2y_5 & y_6(0) &= y_5(0)^3 \\ & & & \\ y_7 &= y_5^2 & & \\ y_7' &= 2y_5y_5' & y_7(0) &= y_5(0)^2 \\ & & & \\ y_8 &= y_2y_3y_4 & & \\ y_8 &= y_2'y_3y_4 + y_2y_3'y_4 + y_2y_3y_4' & y_8(0) &= y_2(0)y_3(0)y_4(0) \end{aligned}$$

$$
\begin{aligned}
y_9 &= y_3y_4 \\
y_9' &= y_3'y_4 + y_3y_4' & y_9(0) &= y_3(0)y_4(0)
\end{aligned}
$$

$$
\begin{aligned}
y_{10} &= y_3y_3 \\
y_{10}' &= 2y_3y_3' & y_{10}(0) &= y_3(0)y_3(0)
\end{aligned}
$$

$$
\begin{aligned}
y_{11} &= y_2y_3 \\
y_{11}' &= y_2'y_3 + y_2y_3' & y_{11}(0) &= y_2(0)y_3(0)\,.
\end{aligned}
$$

After substituting further and simplifying the resulting equations, the system (4.7) can be written as:

$$
\begin{aligned}
y_1' &= y_2 & y_1(0) &= 0 \\
y_2' &= y_9 & y_2(0) &= 0 \\
y_3' &= y_{10} & y_3(0) &= \frac{1}{25} \\
y_4' &= y_5y_8 & y_4(0) &= \sqrt{1 + y_2(0)^2} \\
y_5' &= -y_6y_8 & y_5(0) &= \frac{1}{y_4(0)} \\
y_6' &= -3y_6y_7y_8 & y_6(0) &= y_5(0)^3 \\
y_7' &= -2y_5y_6y_8 & y_7(0) &= y_5(0)^2 \\
y_8' &= y_9y_9 + y_3y_8 + y_5y_8y_{11} & y_8(0) &= y_2(0)y_3(0)y_4(0) \\
y_9' &= y_4y_{10} + y_3y_5y_8 & y_9(0) &= y_3(0)y_4(0) \\
y_{10}' &= 2y_3y_{10} & y_{10}(0) &= y_3(0)y_3(0) \\
y_{11}' &= y_3y_9 + y_2y_{10} & y_{11}(0) &= y_2(0)y_3(0)\,.
\end{aligned}
\tag{4.7}
$$

The matrix-vector notation for (4.7) is in Appendix E.15. The numerical results are in Table 4.16. The plot of the *ORD* function with step size scaling factor $h_{scale} = 8$ is in Figure 4.16.

			MTSM orig		MTSM opt	
Solver	# of steps	Time of calculation [s]	$\lvert error \rvert$	Ratio	$\lvert error \rvert$	Ratio
MTSM orig	40	3.1188×10^{-3}	–	–	–	–
MTSM opt	19	1.7236×10^{-3}	–	–	–	–
ode23	31883	$2.569\,96 \times 10^{-1}$	$9.515\,46 \times 10^{-9}$	**82.40**	1.8429×10^{-7}	**149.1**
ode45	2345	$6.509\,45 \times 10^{-3}$	9.5217×10^{-9}	**2.09**	$1.842\,97 \times 10^{-7}$	**3.78**
ode113	130	$2.624\,45 \times 10^{-3}$	$9.511\,18 \times 10^{-9}$	0.84	$1.842\,86 \times 10^{-7}$	**1.52**

Table 4.16: Results for benchmark problem E5, $h = 0.5\,\mathrm{s}$.

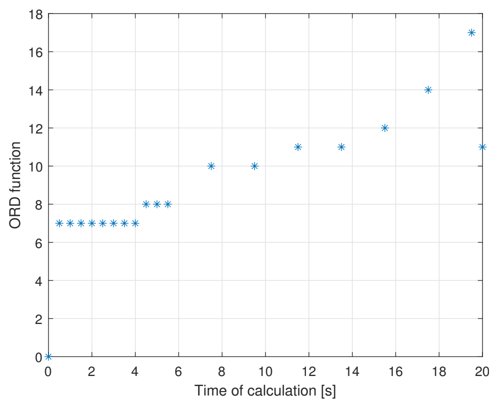


Figure 4.16: *ORD* function for benchmark problem E4, $h = 0.5\,\mathrm{s}$, $h_{scale} = 8$.

4.2 Padé approximation of transport delay

To approximate the delay in the system, the *Padé approximation* is widely used. This example shows how this approximation can be solved using the presented method using the approaches outlined above. The general form and derivation of the Padé approximation can be found in [14]. The modelling of time-delay using the Padé approximation is discussed in [33, 76].

The Padé approximation that replaces exponential function e^{-sT} can be used

$$e^{-sT} \approx \frac{E(-s)}{E(s)} = \frac{y}{z} = \frac{\sum_{k=0}^{n} \frac{(2n-k)!}{k!(n-k)!}(-sT)^k}{\sum_{k=0}^{n} \frac{(2n-k)!}{k!(n-k)!}(sT)^k},$$

where T is the required delay, s is the Laplace transform operator, y is the output – the delayed signal, z is the forcing function – the input signal and n is the required order of the approximation. As an example, consider the equations for $k = 5$ (the fifth-order approximation):

$$\begin{aligned}
k &= 0 \quad \frac{(2\cdot 5-0)!}{0!(5-0)!} = \frac{10!}{5!} = 30240 \\
k &= 1 \quad \frac{(2\cdot 5-1)!}{1!(5-1)!} = \frac{9!}{4!} = 15120 \\
k &= 2 \quad \frac{(2\cdot 5-2)!}{2!(5-2)!} = \frac{8!}{2!3!} = 3360 \\
k &= 3 \quad \frac{(2\cdot 5-3)!}{3!(5-3)!} = \frac{7!}{3!2!} = 420 \\
k &= 4 \quad \frac{(2\cdot 5-4)!}{4!(5-4)!} = \frac{6!}{4!} = 30 \\
k &= 5 \quad \frac{(2\cdot 5-5)!}{5!(5-5)!} = \frac{5!}{5!} = 1\,.
\end{aligned} \tag{4.8}$$

Using the coefficients calculated in (4.8), the polynomials for $E(-s)$ and $E(s)$ have the following form:

$$\begin{aligned}
E(s) &= 30240 + 15120Ts + 3360(Ts)^2 + 420(Ts)^3 + 30(Ts)^4 + 1(Ts)^5 \\
E(-s) &= 30240 - 15120Ts + 3360(Ts)^2 - 420(Ts)^3 + 30(Ts)^4 - 1(Ts)^5\,.
\end{aligned} \tag{4.9}$$

The polynomial can be rewritten as

$$\frac{y}{z} = \frac{E(-s)}{E(s)} = \frac{30240 - 15120Ts + 3360T^2s^2 - 420T^3s^3 + 30T^4s^4 - T^5s^5}{30240 + 15120Ts + 3360T^2s^2 + 420T^3s^3 + 30T^4s^4 + T^5s^5}\,. \tag{4.10}$$

The ODE (4.10) has to be transformed into the equivalent system of first-order ODEs using the previously introduced methods:

- Method of Derivative Order Reduction with Additional Variable – MDORAV (Subsection 2.6.2) or
- Method of Successive Integration – MSI (Subsection 2.6.3).

In Section 2.6, it was shown that both of the methods produce equivalent results. It might therefore seem arbitrary which method is used, but it might not be as simple, when considering the higher-order approximations. There might be other differences for the higher orders that are not immediately obvious. In the following examples, consider the forcing function $z = \sin(t)$.

4.2.1 Method of Derivative Order Reduction with Additional Variable

Simplifying (4.10), we get

$$y = z\frac{30240 - 15120Ts + 3360T^2s^2 - 420T^3s^3 + 30T^4s^4 - T^5s^5}{30240 + 15120Ts + 3360T^2s^2 + 420T^3s^3 + 30T^4s^4 + T^5s^5}\,, \tag{4.11}$$

so that the equation for the additional variable becomes

$$v = \frac{z}{30240 + 15120Ts + 3360T^2s^2 + 420T^3s^3 + 30T^4s^4 + T^5s^5}$$

and the equation for the output y using the additional variable v:

$$y = 30240v - 15120Tsv + 3360T^2s^2v - 420T^3s^3v + 30T^4s^4v - T^5s^5v\,. \tag{4.12}$$

The ODE for the output of the system contains the derivatives of the additional variable v. However, it does not contain a derivative of the forcing function. The Method of Derivative Order Reduction (Subsection 2.6.1) can therefore be used. For $z = \sin(t)$, the resulting system of linear ODEs can be written as

$$\begin{aligned}
s^5v &= \frac{z}{T^5} - \frac{30s^4v}{T} - \frac{420s^3v}{T^2} - \frac{3360s^2v}{T^3} - \frac{15120sv}{T^4} - \frac{30240v}{T^5}\\
s^4v &= \frac{1}{s}s^5v = \frac{1}{s}(\frac{z}{T^5} - \frac{30s^4v}{T} - \frac{420s^3v}{T^2} - \frac{3360s^2v}{T^3} - \frac{15120sv}{T^4} - \frac{30240v}{T^5})\\
s^3v &= \frac{1}{s}s^4v\\
s^2v &= \frac{1}{s}s^3v\\
sv &= \frac{1}{s}s^2v\\
v &= \frac{1}{s}sv
\end{aligned} \tag{4.13}$$

with all initial conditions equal to zero. The equation of the output is solved separately using addition with no additional integration required. The block scheme is in Figure 4.17.

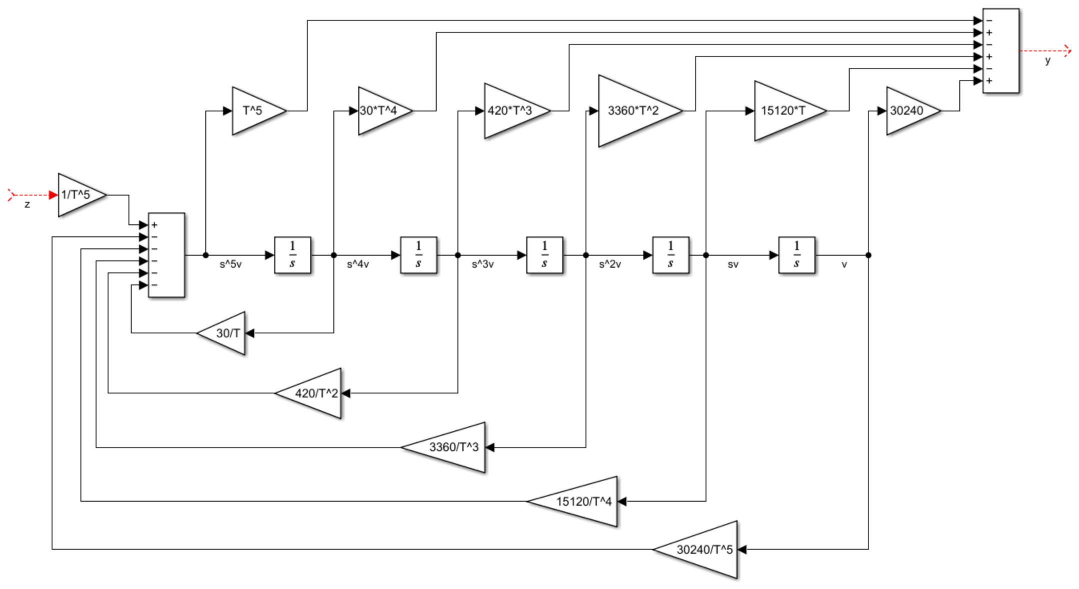


Figure 4.17: The block scheme for the delay modelled using the MDORAV method.

The forcing function z can either

- be generated using auxiliary ODEs in the form $\boldsymbol{y}' = \boldsymbol{A}\boldsymbol{y}+\boldsymbol{b}$, for example for $z = \sin(t)$

$$\boldsymbol{A} = \begin{pmatrix} -\frac{30}{T} & -\frac{420}{T^2} & -\frac{3360}{T^3} & -\frac{15120}{T^4} & -\frac{30240}{T^5} & \frac{1}{T^5} & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{pmatrix} \quad \boldsymbol{b} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \boldsymbol{y}_0 = \begin{pmatrix} s^4v(0) \\ s^3v(0) \\ s^2v(0) \\ sv(0) \\ v(0) \\ z(0) \\ a(0) \end{pmatrix} \qquad (4.14)$$

- be supplied to the system $\boldsymbol{y}' = \boldsymbol{A}\boldsymbol{y} + \boldsymbol{b}$ from the outside

$$\boldsymbol{A} = \begin{pmatrix} -\frac{30}{T} & -\frac{420}{T^2} & -\frac{3360}{T^3} & -\frac{15120}{T^4} & -\frac{30240}{T^5} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad \boldsymbol{b}(t) = \begin{pmatrix} \frac{1}{T^5}z(t) \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \boldsymbol{y}_0 = \begin{pmatrix} s^4v(0) \\ s^3v(0) \\ s^2v(0) \\ sv(0) \\ v(0) \end{pmatrix} . \qquad (4.15)$$

  Note that in this case, the higher derivative of the forcing function cannot be directly calculated.

4.2.2 Method of Successive Integration

First, the (4.10) has to be simplified

$$T^5s^5y + 30T^4s^4y + 420T^3s^3y + 3360T^2s^2y + 15120Tsy + 30240y =$$
$$= -T^5s^5z + 30T^4s^4z - 420T^3s^3z + 3360T^2s^2z - 15120Tsz + 30240z\,.$$

The highest derivative of the output function y has to be on the left side alone, so after further simplification

$$T^5s^5y = -T^5s^5z + 30T^4s^4z - 30T^4s^4y - 420T^3s^3z - 420T^3s^3y + 3360T^2s^2z - 3360T^2s^2y - \\ - 15120Tsz - 15120Tsy + 30240z - 30240y$$

$$s^5y = -s^5z + \frac{30}{T}s^4z - \frac{30}{T}s^4y - \frac{420}{T^2}s^3z - \frac{420}{T^2}s^3y + \frac{3360}{T^3}s^2z - \frac{3360}{T^3}s^2y - \\ - \frac{15120}{T^4}sz - \frac{15120}{T^4}sy + \frac{30240}{T^5}z - \frac{30240}{T^5}y$$

$$s^5y = -s^5z + \frac{30}{T}s^4(z-y) + \frac{420}{T^2}s^3(-z-y) + \frac{3360}{T^3}s^2(z-y) + \frac{15120}{T^4}s(-z-y) + \\ + \frac{30240}{T^5}(z-y)\,.$$

Now, the derivative can be decreased by multiplying the entire equation by $\frac{1}{s}$:

$$\begin{aligned}
s^4y &= -s^4z + \frac{30}{T}s^3(z-y) + \frac{420}{T^2}s^2(-z-y) + \frac{3360}{T^3}s(z-y) + \frac{15120}{T^4}(-z-y) + \frac{1}{s}\left(\frac{30240}{T^5}(z-y)\right) \\
s^3y &= -s^3z + \frac{30}{T}s^2(z-y) + \frac{420}{T^2}s(-z-y) + \frac{3360}{T^3}(z-y) + \frac{1}{s}\left(\frac{15120}{T^4}(-z-y) + v_1\right) \\
s^2y &= -s^2z + \frac{30}{T}s(z-y) + \frac{420}{T^2}(-z-y) + \frac{1}{s}\left(\frac{3360}{T^3}(z-y) + v_2\right) \\
sy &= -sz + \frac{30}{T}(z-y) + \frac{1}{s}\left(\frac{420}{T^2}(-z-y) + v_3\right) \\
y &= -z + \frac{1}{s}\left(\frac{30}{T}(z-y) + v_4\right) = -z + v_5\,.
\end{aligned} \tag{4.16}$$

After simplifying the system (4.16), the system of ODEs that are solved can be written as:

$$\begin{aligned}
y &= -z + v_5 \\
v_5' &= \frac{30}{T}(z-y) + v_4 \\
v_4' &= \frac{420}{T^2}(-z-y) + v_3 \\
v_3' &= \frac{3360}{T^3}(z-y) + v_2 \\
v_2' &= \frac{15120}{T^4}(-z-y) + v_1 \\
v_1' &= \frac{30240}{T^5}(z-y)\,.
\end{aligned} \tag{4.17}$$

The block scheme is in Figure 4.18.

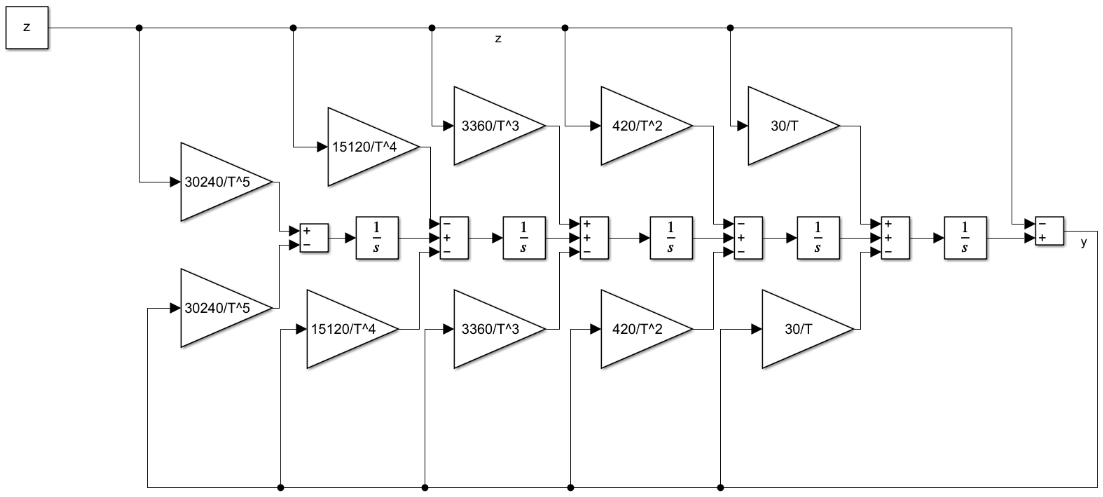


Figure 4.18: The block scheme for the delay modelled using the MSI method.

The system (4.17) directly represents the block scheme in Figure 4.18. It is not however directly usable for calculation using matrix-vector approach. The output equation (y) has to be substituted into the others. The resulting system therefore depends only on the input (forcing) function

$$
\begin{aligned}
v_5' &= \frac{30}{T}(z + z - v_5) + v_4 = \frac{30}{T}(2z - v_5) + v_4 \\
v_4' &= \frac{420}{T^2}(-z + z - v_5) + v_3 = \frac{420}{T^2}(-v_5) + v_3 \\
v_3' &= \frac{3360}{T^3}(z + z - v_5) + v_2 = \frac{3360}{T^3}(2z - v_5) + v_2 \\
v_2' &= \frac{15120}{T^4}(-z + z - v_5) + v_1 = \frac{15120}{T^4}(-v_5) + v_1 \\
v_1' &= \frac{30240}{T^5}(z + z - v_5) = \frac{30240}{T^5}(2z - v_5)\,.
\end{aligned}
\tag{4.18}
$$

The resulting system (4.18) is linear in the general form $\boldsymbol{y}' = \boldsymbol{A}\boldsymbol{y} + \boldsymbol{b}$. The forcing function z can either

- be generated using auxiliary ODEs, for example for $z = \sin(t)$

$$
\boldsymbol{A} = \begin{pmatrix}
-\frac{30}{T} & 1 & 0 & 0 & 0 & 2\frac{30}{T} & 0 \\
-\frac{420}{T^2} & 0 & 1 & 0 & 0 & 0 & 0 \\
-\frac{3360}{T^3} & 0 & 0 & 1 & 0 & 2\frac{3360}{T^3} & 0 \\
-\frac{15120}{T^4} & 0 & 0 & 0 & 1 & 0 & 0 \\
-\frac{30240}{T^5} & 0 & 0 & 0 & 0 & \frac{30240}{T^5} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & -1 & 0
\end{pmatrix}
\quad
\boldsymbol{b} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
\quad
\boldsymbol{y}_0 = \begin{pmatrix} v_5(0) \\ v_4(0) \\ v_3(0) \\ v_2(0) \\ v_1(0) \\ z(0) \\ a(0) \end{pmatrix}
\tag{4.19}
$$

- be supplied to the system from the outside

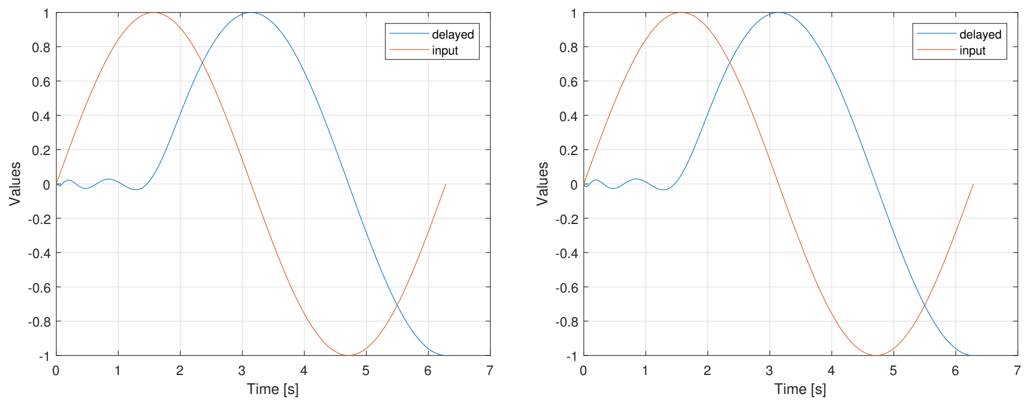
$$\boldsymbol{A} = \begin{pmatrix} -\frac{30}{T} & 1 & 0 & 0 & 0 \\ -\frac{420}{T^2} & 0 & 1 & 0 & 0 \\ -\frac{3360}{T^3} & 0 & 0 & 1 & 0 \\ -\frac{15120}{T^4} & 0 & 0 & 0 & 1 \\ -\frac{30240}{T^5} & 0 & 0 & 0 & 0 \end{pmatrix} \boldsymbol{b}(t) = \begin{pmatrix} 2\frac{30}{T}z(t) \\ 0 \\ 2\frac{3360}{T^3}z(t) \\ 0 \\ 2\frac{30240}{T^5}z(t) \end{pmatrix} \boldsymbol{y}_0 = \begin{pmatrix} v_5(0) \\ v_4(0) \\ v_3(0) \\ v_2(0) \\ v_1(0) \end{pmatrix}. \tag{4.20}$$

Note that in this case, the higher derivative of the forcing function cannot be directly calculated.

4.2.3 Numerical results

Numerically, the equations generated using both methods (MDORAV and MSI) produce equivalent results (see Section 2.6). In all of the following experiments $z = \sin(t)$ is always going to be used as the forcing function. The required delay is going to be set to $T = \frac{\pi}{2}$ s. Maximum time of calculation $t_{max} = 2\pi$ s, step size $h = \frac{t_{max}}{100}$ s and accuracy of the calculation $TOL = 1 \times 10^{-9}$. The values in the column labelled $|error|$ are calculated as $|y_{calculated}(T) - \sin(l - T)|$, where l is the last value of vector of integration times used by the numerical method. The order of approximation is set to five ($k = 5$). The results for additional orders of approximation are in Appendix B.

First, consider MDORAV with generated auxiliary equations (4.14). The plots of the input and the delayed function are in Figure 4.19, plot of the ORD function is in Figure 4.20.



(a) Delay calculated using MTSM solver.

(b) Delay calculated using ode45 solver.

Figure 4.19: Delay of $z = \sin(t)$ using the MDORAV method.

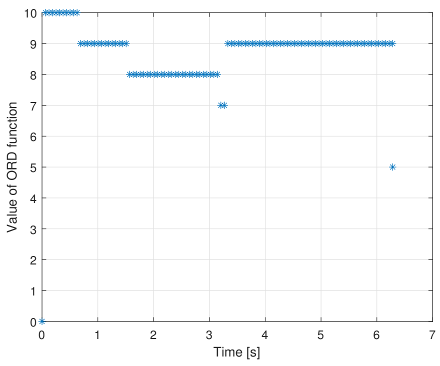


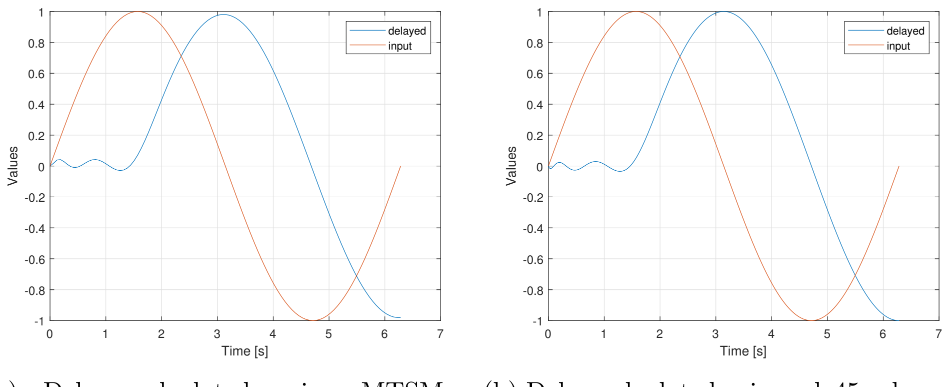
Figure 4.20: Plot of the ORD function for $z = \sin(t)$ using the MDORAV method.

The results for this experiment are in Table 4.17. The MTSM beats the state-of-the-art methods and performs the smallest number of steps.

Solver	Time of calculation [s]	Ratio	Number of steps	$\lvert error \rvert$
MTSM	1.3009×10^{-3}	–	100	$2.305\,66 \times 10^{-8}$
ode23	1.1001×10^{-2}	**8.46**	2089	$1.585\,61 \times 10^{-8}$
ode45	1.3756×10^{-3}	**1.06**	521	$2.236\,28 \times 10^{-8}$
ode113	2.2932×10^{-3}	**1.76**	115	$1.700\,88 \times 10^{-8}$

Table 4.17: The numerical results for the Padé approximation using MDORAV method for $z = \sin(t)$.

Next, the forcing function is going to be supplied from the outside of the system (therefore right-hand side vector changes in every integration step), as defined by (4.15). The plots of the input and the delayed function are in Figure 4.21, the plot of ORD function is in Figure 4.22.



(a) Delay calculated using MTSM solver. (b) Delay calculated using ode45 solver.

Figure 4.21: Delay of $z = \sin(t)$ being supplied from the outside of the system using the MDORAV method.

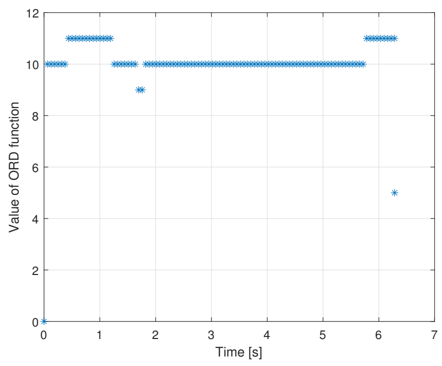


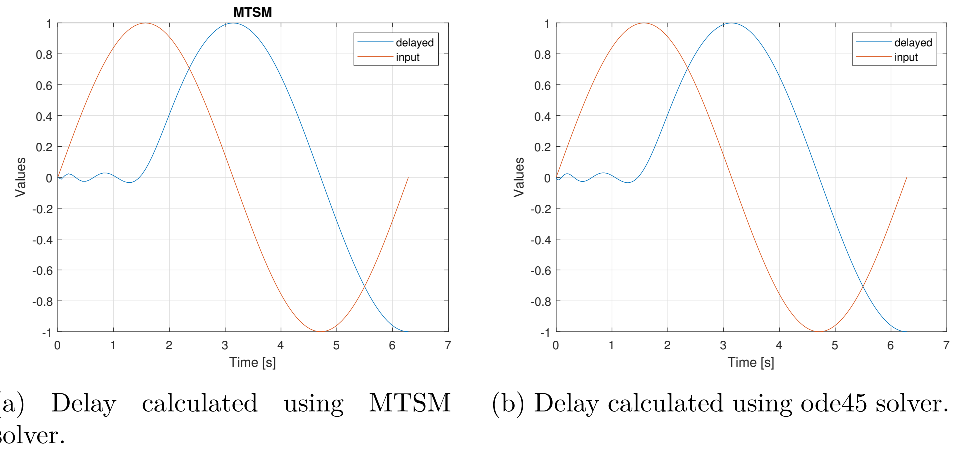
Figure 4.22: Plot of the ORD function for $z = \sin(t)$ being supplied from the outside of the system using the MDORAV method.

The results for this experiment are in Table 4.18. Because MTSM can only work with the supplied value of the forcing function, which is constant for every step, higher derivatives cannot be calculated directly. Therefore the required accuracy cannot be achieved using the outside forcing function. The method uses just one Taylor series term.

Solver	Time of calculation [s]	Ratio	Number of steps	$\|error\|$
MTSM	2.2486×10^{-3}	–	100	0.0196986
ode23	9.6258×10^{-3}	**4.29**	547	9.91653×10^{-7}
ode45	7.4261×10^{-3}	**3.3**	805	3.38862×10^{-8}
ode113	1.68324×10^{-2}	**7.49**	119	5.47037×10^{-7}

Table 4.18: The numerical results for the Padé approximation using MDORAV method, when z is being supplied from the outside of the system.

For the MSI, the expectation is that the results would be similar. First, for the system (4.19), plots of the input and the delayed function are in Figure 4.23, the plot of the ORD function is in Figure 4.24.



(a) Delay calculated using MTSM solver.

(b) Delay calculated using ode45 solver.

Figure 4.23: Delay of $z = \sin(t)$ using the MSI method.

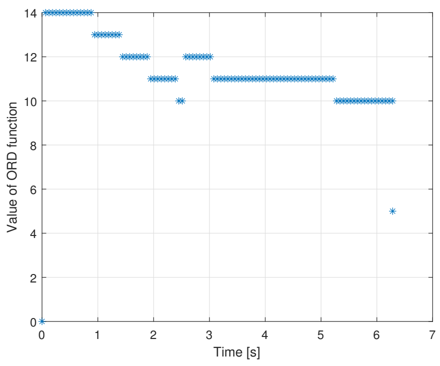


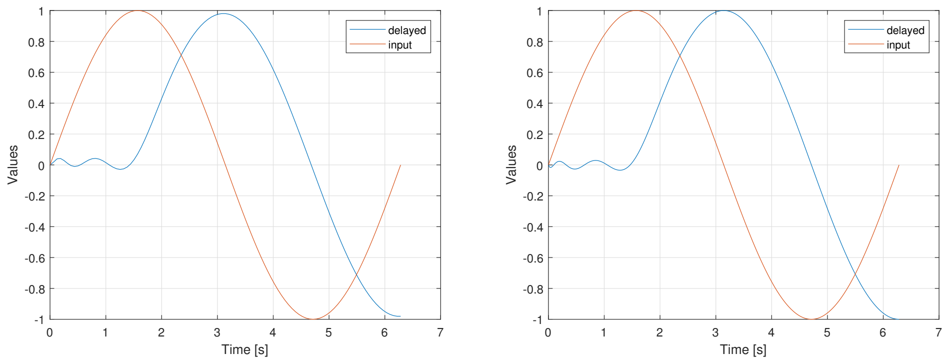
Figure 4.24: Plot of the *ORD* function for $z = \sin(t)$ using the MSI method.

The results for this experiment are in Table 4.19. The mean value of the *ORD* function is 11.49. The MTSM is again faster than the state-of-the-art methods by a wide margin.

Solver	Time of calculation [s]	Ratio	Number of steps	$\|error\|$
MTSM	1.5426×10^{-3}	–	100	$2.305\,67 \times 10^{-8}$
ode23	3.4744×10^{-2}	**22.52**	5249	2.1039×10^{-8}
ode45	3.5241×10^{-3}	**2.28**	1233	$2.289\,07 \times 10^{-8}$
ode113	3.0408×10^{-3}	**1.97**	202	$2.295\,63 \times 10^{-8}$

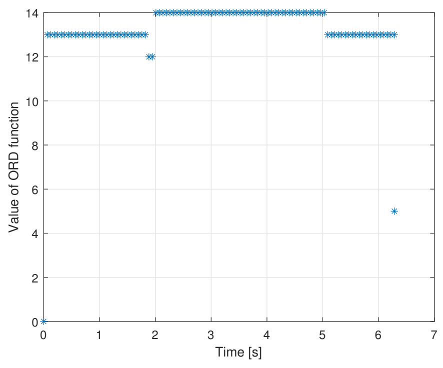
Table 4.19: The numerical results for the Padé approximation using MSI method.

The final experiment is with system (4.20). The plots of the input and the delayed function are in Figure 4.25 and the plot of the *ORD* function is in Figure 4.26.



(a) Delay calculated using mtsm solver. (b) Delay calculated using ode45 solver.

Figure 4.25: Delay of $z = \sin(t)$ using the MSI method using the Padé approximation when z is supplied from the outside of the system.



(a)

Figure 4.26: Plot of the ORD function for $z = \sin(t)$ using the MSI method using the Padé approximation when z is supplied from the outside of the system.

The results for this experiment are in Table 4.20. The mean value of the ORD function is between 13 and 14. The MTSM is again faster than the state-of-the-art methods by a wide margin. Because MTSM can only work with the supplied value of the forcing function, which is constant for every step, higher derivatives cannot be calculated, and the method just uses one Taylor series term, as with the MDORAV example.

Solver	Time of calculation [s]	Ratio	Number of steps	Error
MTSM	2.4124×10^{-3}	–	100	$1.969\,86 \times 10^{-2}$
ode23	$4.802\,74 \times 10^{-2}$	**19.91**	5325	$2.457\,36 \times 10^{-8}$
ode45	9.6089×10^{-3}	**3.98**	3005	$2.291\,49 \times 10^{-8}$
ode113	3.994×10^{-3}	**1.66**	203	$2.310\,85 \times 10^{-8}$

Table 4.20: The numerical results for the Padé approximation using MSI method, when z is being supplied from the outside of the system.

4.2.4 Concluding remarks

The results of experiments with Padé approximation are interesting and show the inherent limitations of the MTSM. When the system cannot be modelled entirely and without higher derivatives of the forcing function, the method cannot achieve the required precision. When applying the method in a real-world setting (in the context of this thesis, control of the systems in real-time), this shortcoming has to be taken into consideration, and it has to be worked around or addressed.

As for the experiments with the different transformation methods, MSI method produces much simpler block scheme representation of the system (for higher orders of the approximation, it becomes even more apparent). The number of integrators remains the same for both methods.

4.3 Movement of a charged particle

This experiment was published in [78]. Movement of a charged particle in an electromagnetic and electrostatic field is given by Lorenz (electromagnetic) force [65], [45]

$$\boldsymbol{f} = q(\boldsymbol{e} + \boldsymbol{v}\boldsymbol{b}), \tag{4.21}$$

where $\boldsymbol{e} = (e_x, e_y, e_z)^T$ is the electric field, $\boldsymbol{b} = (b_x, b_y, b_z)^T$ is the magnetic field and vector $\boldsymbol{v}$ is instantaneous velocity of the charged particle with charge q. The force vector $\boldsymbol{f}$ can be substituted as $m\boldsymbol{a}$,

$$m\boldsymbol{a} = q(\boldsymbol{e} + \boldsymbol{v}\boldsymbol{b})\,. \tag{4.22}$$

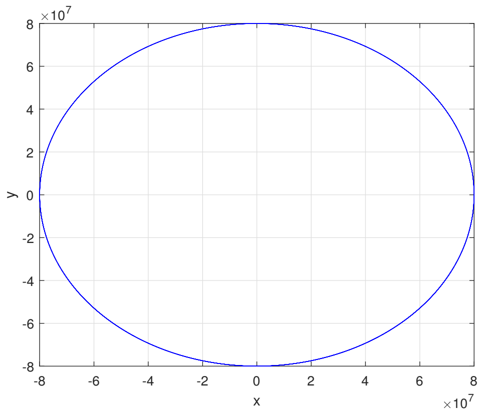
We can further suppose that acceleration x, y and z axis $\boldsymbol{a} = (x'', y'', z'')^T$ can be calculated as

$$\begin{aligned} x'' &= \frac{q}{m}\left(e_x + y'b_z - z'b_y\right)\,, \\ y'' &= \frac{q}{m}\left(e_y + z'b_x - x'b_z\right)\,, \\ z'' &= \frac{q}{m}\left(e_z + x'b_y - y'b_x\right)\,. \end{aligned} \tag{4.23}$$

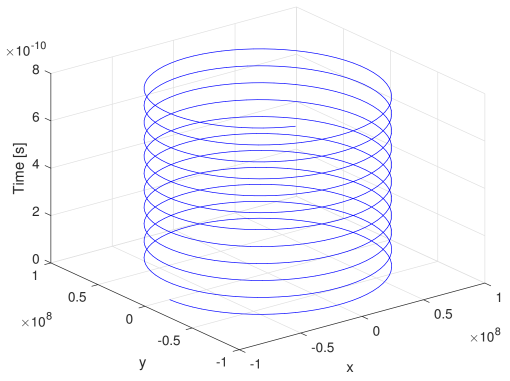
The speed of the particle can be represented by the vector $\boldsymbol{v} = (x', y', z')^T$. By substituting $\boldsymbol{y} = (x', y', z', x, y, z)^T$, the problem can be transformed into a system of linear ODEs in a matrix-vector representation (3.3)

$$\boldsymbol{A} = \begin{pmatrix} 0 & \frac{q}{m}b_z & -\frac{q}{m}b_y & 0 & 0 & 0 \\ -\frac{q}{m}b_z & 0 & \frac{q}{m}b_x & 0 & 0 & 0 \\ \frac{q}{m}b_y & -\frac{q}{m}b_x & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{4.24}$$

where initial conditions are $\boldsymbol{y}_0 = \left(-8 \cdot 10^7, 0, 0, 0, 0, 0\right)^T$. Constants are set to $m = 9.109\,383\,56 \times 10^{-31}\,\text{kg}$, $q = -1.6 \times 10^{-19}\,\text{C}$, $b_x = b_y = 0$, $b_z = 5 \times 10^{-1}\,\text{T}$ and $e_x = e_y = e_z = 0\,\text{T}$. Further, the right-hand side $\boldsymbol{b} = (\frac{q}{m}e_x, \frac{q}{m}e_y, \frac{q}{m}e_z, 0, 0, 0)^T = \boldsymbol{0}$. The behaviour of the electron is in Figure 4.27.



(a) Position of the charged particle.



(b) Trajectory of the charged particle.

Figure 4.27: Behaviour of the charged particle.

Results of simulations using IVP (4.24) for the given parameters of the electromagnetic and electrostatic fields and time of simulation $t_{max} = 1 \times 10^{-8}\,\text{s}$ are in Table 4.21. MTSM (with the step size $h = 1 \times 10^{-10}\,\text{s}$) calculates the problem faster than the state-of-the-art MATLAB ode solvers. The order of the method is $ORD = 51 \pm 1$.

Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	$4.772\,85 \times 10^{-3}$	–	100
ode23	193.097	**10826.2**	2235540
ode45	4.83689	**2712.2**	213925
ode113	7.8659×10^{-1}	**44.1**	9446

Table 4.21: Results of calculations for movement of a charged particle.

4.4 Telegraph line

Another linear problem, analysed very thoroughly in our research group, is the travel of a signal through a telegraph line. This problem was presented at several conferences (parallel solution in [60] for example) and in journal publications ([81], [58] for example).

Voltage and current change along the telegraph line continuously in time, and they can be expressed using equations

$$u = u(x,t), \tag{4.25}$$
$$i = i(x,t), \tag{4.26}$$

where x is a distance from the beginning of the line and t is the time. Voltage and current in the distance $x + \mathrm{d}x$ can be expressed as

$$u(x + \mathrm{d}x) = u(x,t) + \frac{\partial u}{\partial x}\mathrm{d}x, \tag{4.27}$$
$$i(x + \mathrm{d}x) = i(x,t) + \frac{\partial i}{\partial x}\mathrm{d}x\,. \tag{4.28}$$

Basic Line Equations (4.29), (4.30) describe the change of voltage and current on the line

$$-\frac{\partial u}{\partial x} = Ri + L\frac{\partial i}{\partial t}\,, \tag{4.29}$$
$$-\frac{\partial i}{\partial x} = Gu + C\frac{\partial u}{\partial t}\,, \tag{4.30}$$

where the following constants are the parameters of the line:

- R is the resistance of the wire,
- G is the conductance between wires,
- L is the inductance of the wire (e.g. due to the magnetic field around the wires) and
- C is the capacitance between two wires.

Using (4.29) and (4.30) it is possible to construct a model of the segment (Figure 4.28). The entire line is comprised of a series of infinite number of connected segments. The line chained using this model is *lossy*.

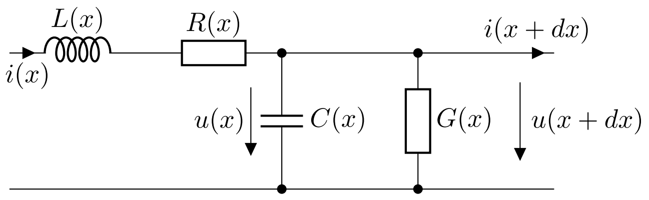


Figure 4.28: Modelling a segment of the telegraph line – complex model.

The model in Figure 4.28 can be simplified by removing the terms $R(x)$ and $G(x)$. The simplified model is in Figure 4.29. The line then becomes *lossless*.

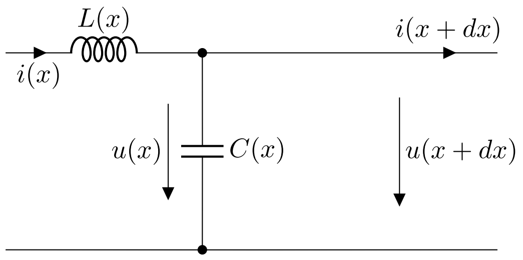


Figure 4.29: Modelling a segment of the wire – simplified model.

Based on the simplified model in Figure 4.29, partial differential equations for voltage and current can be derived

$$L \cdot C \frac{\partial^2 u(x,t)}{\partial t^2} - \frac{\partial^2 u(x,t)}{\partial x^2} = 0, \tag{4.31}$$

$$L \cdot C \frac{\partial^2 i(x,t)}{\partial t^2} - \frac{\partial^2 i(x,t)}{\partial x^2} = 0. \tag{4.32}$$

The equations (4.31) and (4.32) can be solved analytically using, for example, the method of Separation of Variables. However, the solution of large systems of PDEs is very complicated. It can, however, be solved numerically. By chaining the segments in Figure 4.29, the lossless model of the line can be constructed, with the number of segments denoted as S. The model is in Figure 4.30.

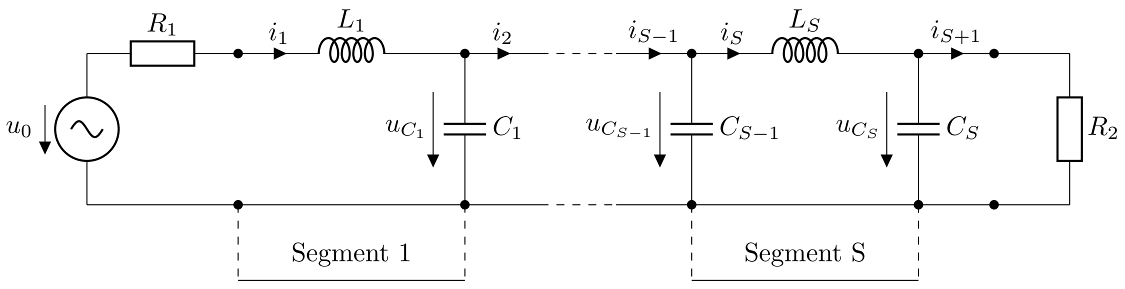


Figure 4.30: Model of the line – chain of S segments.

The model can be described by a system of first-order ODEs. For the first segment

$$\begin{aligned} u'_{C_1} &= \frac{1}{C_1}(i_1 - i_2) \\ i'_1 &= \frac{1}{L_1}(u_0 - u_{C_1} - R_1 \cdot i_1)\,, \end{aligned} \tag{4.33}$$

where u_0 is the input voltage of the system, u_{C_1} is the voltage on the first capacitor and i_1 is the current that flows through the first inductor. Resistor R_1 represents the input load of the transmission line. Equations for the following segments are very similar to one another. For the second segment

$$\begin{aligned} u'_{C_2} &= \frac{1}{C_2}(i_2 - i_3) \\ i'_2 &= \frac{1}{L_2}(u_{C_1} - u_{C_2})\,, \end{aligned} \tag{4.34}$$

for the next segments

$$\begin{aligned} u'_{C_k} &= \frac{1}{C_k}(i_k - i_{k+1}) \\ i'_k &= \frac{1}{L_k}(u_{C_{k-1}} - u_{C_k}), \end{aligned} \tag{4.35}$$

where $k \in \langle 3, S \rangle$. The last segment of the line ends with an output load, simulated by the resistor R_2

$$i_{S+1} = \frac{1}{R_2} u_{C_S}\,. \tag{4.36}$$

Note that all differential equations have initial conditions equal to zero. The input voltage u_0 can be either a constant (DC circuit) or harmonic (AC circuit) signal. For the DC circuit, the input voltage u_0 is hidden in constant right-hand side vector $\boldsymbol{b}$, see (4.38). For the AC circuit, the input voltage $u_0 = U_0 \sin(\omega t)$ can be calculated using auxiliary system or linear ODEs

$$\begin{aligned} u'_0 &= \omega x \qquad u_0(0) = 0 \\ x' &= -\omega u_0 \quad\; x(0) = U_0\,. \end{aligned} \tag{4.37}$$

The problem can be described using the matrix-vector representation

$$\boldsymbol{A} = \left(\begin{array}{c|c} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \hline \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{array}\right), \quad \boldsymbol{y} = \left(\begin{array}{c} u_{C_1} \\ \vdots \\ u_{C_S} \\ \hline i_1 \\ \vdots \\ i_S \end{array}\right), \quad \boldsymbol{b} = \left(\begin{array}{c} 0 \\ \vdots \\ 0 \\ \hline \frac{u_0}{L_1} \\ \vdots \\ 0 \end{array}\right), \tag{4.38}$$

where $\boldsymbol{A}_{11}$, $\boldsymbol{A}_{12}$, $\boldsymbol{A}_{21}$ and $\boldsymbol{A}_{22}$ are individual block matrices with size $S \times S$:

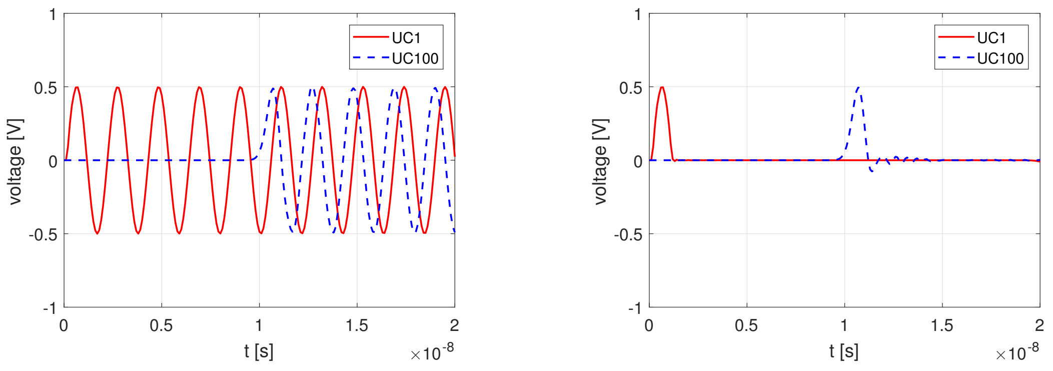
$$
\boldsymbol{A}_{11} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \frac{-1}{R_2 C_S} \end{pmatrix} \qquad \boldsymbol{A}_{12} = \begin{pmatrix} \frac{1}{C_1} & \frac{-1}{C_1} & 0 & \cdots & \cdots & 0 \\ 0 & \frac{1}{C_2} & \frac{-1}{C_2} & 0 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & \frac{1}{C_S} \end{pmatrix}
$$
$$
\boldsymbol{A}_{21} = \begin{pmatrix} \frac{-1}{L_1} & 0 & 0 & \cdots & \cdots & 0 \\ \frac{1}{L_2} & \frac{-1}{L_2} & 0 & 0 & \cdots & \vdots \\ 0 & \frac{1}{L_3} & \frac{-1}{L_3} & 0 & \cdots & \vdots \\ 0 & \cdots & \cdots & \cdots & \frac{1}{L_S} & \frac{-1}{L_S} \end{pmatrix} \qquad \boldsymbol{A}_{22} = \begin{pmatrix} \frac{-R_1}{L_1} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}. \tag{4.39}
$$

The behaviour of the transmission on the line is based on the values of the input (R_1) and the output (R_2) loads. If the condition

$$
R_1 = R_2 = \sqrt{L/C} \tag{4.40}
$$

holds, the line is adjusted, and a signal on the line is transmitted without change. For the simulation experiments, the line is adjusted for $R_1 = R_2 = 100\,\Omega$.

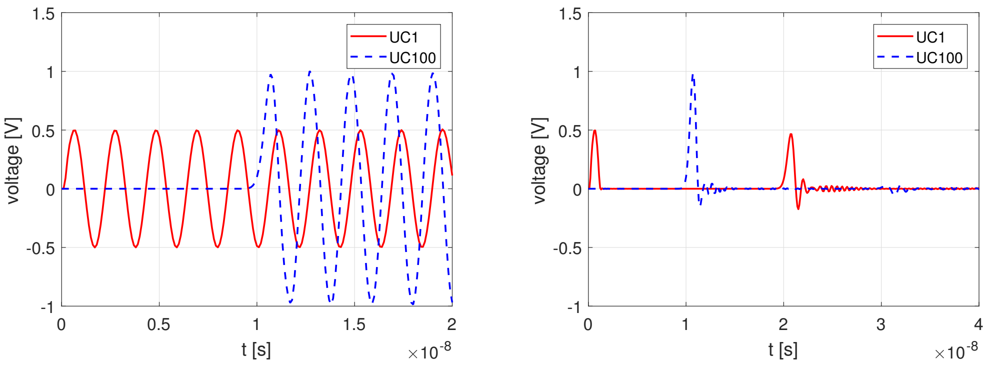
The propagation constant per unit length of one segment for the used model can be calculated as $t_{LC} = \sqrt{LC}$. The total delay of the input signal can be calculated as $t_{delay} = S \cdot t_{LC}$. The simulation time for all experiments was set $t_{max} = 2t_{delay}$, tolerances for all solvers were set to 10^{-7}. For $S = 100$ and used L and C, the output signal is delayed by $100 \cdot \sqrt{10^{-8} \cdot 10^{-12}} = 1 \times 10^{-8}$ s. The behaviour of the signal for the harmonic input and the impulse input on the line comprised of $S = 100$ segments is in Figure 4.31.



(a) Harmonic input. (b) Impulse input.

Figure 4.31: Adjusted line – output just delayed.

The next experiment was done for the not adjusted line. The cut line can be simulated by setting the parameters $R_1 = 100\,\Omega$, $R_2 = 1 \times 10^{12}\,\Omega$. The behaviour of such telegraph line with $S = 100$ segments is in Figure 4.32.



(a) Harmonic input. (b) Impulse input.

Figure 4.32: Not adjusted line – output delayed and amplified.

Notice that in Figure 4.32b, the input signal is amplified at the output and bounced back to the input. This can be useful to determine where the line is cut.

For the simulation experiments, the capacitances and inductances are set to the same values: $C_1 = C_2 = \cdots = C_S = 1\,\text{pF}$ and $L_1 = L_2 = \cdots = L_S = 10\,\text{nH}$ (homogeneous lossless telegraph line), MTSM uses the integration step size $h = 5 \times 10^{-10}\,\text{s}$. Table 4.22 shows the numerical results for the harmonic input.

	Adjusted line $R_1 = R_2 = 100\,\Omega$				Not adjusted line $R_1 = 100\,\Omega$, $R_2 = 1 \times 10^{12}\,\Omega$			
	ode23	ode45	ode113	MTSM	ode23	ode45	ode113	MTSM
S	*ratio*	*ratio*	*ratio*	[s]	*ratio*	*ratio*	*ratio*	[s]
200	**29.5**	**12.1**	**9.7**	3.5×10^{-2}	**28.6**	**10.8**	**9.4**	3.5×10^{-2}
600	**17.3**	**9.3**	**6.5**	2.11×10^{-1}	**18.2**	**9.4**	**6.5**	2.1×10^{-1}
1000	**10.3**	**5.1**	**2.8**	8.95×10^{-1}	**10.3**	**4.7**	**2.8**	8.92×10^{-1}

Table 4.22: MATLAB time of solutions for *harmonic input*: MTSM with fixed integration time step $h = 5 \times 10^{-10}\,\text{s}$.

Table 4.23 show the results for impulse input.

	Adjusted line $R_1 = R_2 = 100\,\Omega$				Not adjusted line $R_1 = 100\,\Omega$, $R_2 = 1 \times 10^{12}\,\Omega$			
	ode23	ode45	ode113	MTSM	ode23	ode45	ode113	MTSM
S	*ratio*	*ratio*	*ratio*	[s]	*ratio*	*ratio*	*ratio*	[s]
200	**28.1**	**11.6**	**9.3**	3.6×10^{-2}	**30.3**	**11.7**	**9.3**	3.6×10^{-2}
600	**16.5**	**9.3**	**6.4**	2.14×10^{-1}	**17.6**	**9.5**	**6.4**	2.13×10^{-1}
1000	**8.7**	**4.5**	**2.9**	9.07×10^{-1}	**9.6**	**4.7**	**3.0**	9.07×10^{-1}

Table 4.23: MATLAB time of solutions for *impulse input*: MTSM with fixed integration time step $h = 5 \times 10^{-10}\,\text{s}$.

Both experiments show that the method is noticeably faster than the state-of-the-art numerical methods. The method calculates the presented linear problems faster with similar or better accuracy, which is beneficial in real-time context as discussed in Chapter 2.

4.5 Coefficients of the Fourier series

The previous examples showed the behaviour of MTSM when solving real-world linear problems. The following two problems are going to show its behaviour while calculating the non-linear systems. The first non-linear example is the calculation of the coefficients of the Fourier series. This problem was published in [78].

A Fourier series is the limit of the sequence of trigonometric polynomials that have a cosine part and a sine part. They are mainly used in the study of phenomena with a periodic character. The advantage of the Fourier series is that the requirements for its convergence to the developed function are weaker than in the case of the Taylor series. For example, the existence of derivatives of all orders of a given function at a given point is not required. The calculation of coefficients can be (especially when using numerical methods) easier than for the Taylor series. Any periodic function $f(t)$ can be written as a sum of harmonics (4.41) called the Fourier series

$$f(t) = \frac{a_0}{2} + a_1 \cos(\omega t) + a_2 \cos(2\omega t) + \cdots + b_1 \sin(\omega t) + b_2 \sin(2\omega t) + \cdots . \tag{4.41}$$

The coefficients can be calculated using definite integrals

$$a_0 = \frac{2}{T} \int_0^T f(t)\,\mathrm{d}t, \tag{4.42}$$

$$a_k = \frac{2}{T} \int_0^T f(t) \cos(k\omega t)\,\mathrm{d}t, \qquad k = 1, 2, 3, \ldots, n\,, \tag{4.43}$$

$$b_k = \frac{2}{T} \int_0^T f(t) \sin(k\omega t)\,\mathrm{d}t, \qquad k = 1, 2, 3, \ldots, n\,, \tag{4.44}$$

where T is the period of the function $f(t)$. The definite integral

$$Y = \int_0^T f(x)\,\mathrm{d}x \tag{4.45}$$

can be transformed into the IVP

$$y' = f(x) \qquad y(0) = 0\,, \tag{4.46}$$

where $y(T) = Y$ and T denotes the maximum time of calculation $t \in (0, T)$. More details about transformation and numerical solution of definite integrals using MTSM can be found in [20]. For example, the analytical calculation of coefficient A_2 using the definite integral

$$A_2 = \frac{2}{T} \int_0^T f(t) \cos(2\omega t)\,\mathrm{d}t \tag{4.47}$$

can be transformed into the initial value problem

$$a_2' = \frac{2}{T} f(t) \cos(2\omega t), \quad a_2(0) = 0\,. \tag{4.48}$$

The solution of the IVP (4.48) at the maximum time T represents the calculated value of the definite integral (4.47) ($A_2 \approx a_2(T)$). As an example, consider a simple harmonic function

$$f(t) = \sin^2(\omega t) \tag{4.49}$$

with parameters $\omega = \frac{2\pi}{T}$ rad·s^{-1}, $T = 2$ s. The coefficients of the Fourier series for (4.49) that are given by

$$\sin^2(\omega t) = \frac{a_0}{2} + a_2 \cos(2\omega t) \tag{4.50}$$

can be calculated analytically as

$$\sin^2(\omega t) = \frac{1}{2} - \frac{1}{2}\cos(2\omega t) \tag{4.51}$$

where $A_0 = 1$ and $A_2 = -\frac{1}{2}$. The coefficients can be calculated numerically using the following system of ODEs

$$\begin{aligned} a_0' &= f(t) & a_0(0) &= 0 \\ a_2' &= \frac{2}{T} f(t)\cos(2\omega t) & a_2(0) &= 0\,. \end{aligned} \tag{4.52}$$

The solution of the IVP in $t \in (0, T)$, where $T = 2$ s is in Figure 4.33. The final values of functions $a_0(t)$ and $a_2(t)$ show the calculated values of Fourier coefficients $a_0(T) = 1$, $a_2(T) = -0.5$.

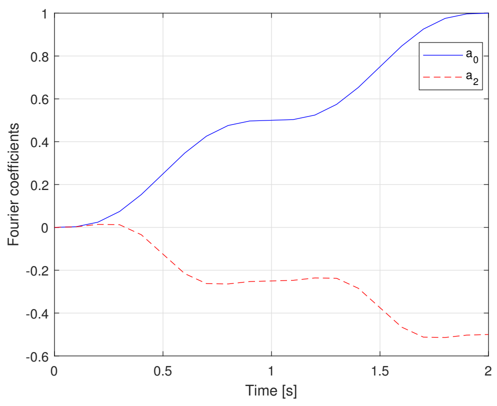


Figure 4.33: Fourier coefficients of $f(t) = \sin^2(\omega t)$.

The (4.52) can be transformed into different autonomous systems with auxiliary generating equations. The first substitution $\boldsymbol{y} = (a_0,\, a_2,\, \sin(\omega t),\, \cos(\omega t),\, \sin(2\omega t),\, \cos(2\omega t))^T$ leads to the following non-linear system of ODEs

$$\begin{aligned} y_1' &= y_3^2 & y_1(0) &= 0 \\ y_2' &= \frac{2}{T} y_3^2 y_6 & y_2(0) &= 0 \\ y_3' &= \omega y_4 & y_3(0) &= 0 \\ y_4' &= -\omega y_3 & y_4(0) &= 1 \\ y_5' &= 2\omega y_6 & y_5(0) &= 0 \\ y_6' &= -2\omega y_5 & y_6(0) &= 1\,. \end{aligned} \tag{4.53}$$

The more effective way of solving (4.52) is to transform it into an autonomous system with fewer multiplications or even better to the linear system of ODEs. The next substitution $\boldsymbol{y} = (a_0,\, a_2,\, \sin^2(\omega t),\, \sin(\omega t)\cos(\omega t),\, \cos^2(\omega t),\, \cos(2\omega t),\, \sin(2\omega t))^T$ leads to the following non-linear system of ODEs with at most two-term multiplication

$$
\begin{aligned}
y_1' &= y_3 & y_1(0) &= 0 \\
y_2' &= \frac{2}{T} y_3 y_6 & y_2(0) &= 0 \\
y_3' &= 2\omega y_4 & y_3(0) &= 0 \\
y_4' &= \omega(y_5 - y_3) & y_4(0) &= 0 \\
y_5' &= -2\omega y_4 & y_5(0) &= 1 \\
y_6' &= -2\omega y_7 & y_6(0) &= 1 \\
y_7' &= 2\omega y_6 & y_7(0) &= 0\,.
\end{aligned}
\tag{4.54}
$$

The last transformation of (4.52) using substitution $\boldsymbol{y} = (a_0, a_2, \sin^2(\omega t), \sin(\omega t)\cos(\omega t), \cos^2(\omega t), \sin^2(\omega t)\cos(2\omega t), \sin^2(\omega t)\sin(2\omega t), \sin(\omega t)\cos(\omega t)\cos(2\omega t), \sin(\omega t)\cos(\omega t)\sin(2\omega t), \cos^2(\omega t)\cos(2\omega t),\ \cos^2(\omega t)\sin(2\omega t))^T$ leads to the linear IVP

$$
\begin{aligned}
y_1' &= y_3 & y_1(0) &= 0 \\
y_2' &= \frac{2}{T} y_6 & y_2(0) &= 0 \\
y_3' &= 2\omega y_4 & y_3(0) &= 0 \\
y_4' &= \omega(y_5 - y_3) & y_4(0) &= 0 \\
y_5' &= -2\omega y_4 & y_5(0) &= 1 \\
y_6' &= 2\omega(y_8 - y_7) & y_6(0) &= 0 \\
y_7' &= 2\omega(y_9 + y_6) & y_7(0) &= 0 \\
y_8' &= \omega(y_{10} - y_6 - 2y_9) & y_8(0) &= 0 \\
y_9' &= \omega(y_{11} - y_7 + 2y_8) & y_9(0) &= 0 \\
y_{10}' &= -2\omega(y_8 + y_{11}) & y_{10}(0) &= 1 \\
y_{11}' &= -2\omega(y_9 - y_{10}) & y_{11}(0) &= 0\,.
\end{aligned}
\tag{4.55}
$$

Results of the calculation of the Fourier coefficients using IVPs (4.54) and (4.55) are in Table 4.24 and Table 4.25, respectively. The tolerance for all solvers was set to 1×10^{-10} with fixed integration step size for the MTSM solver set at $h = 0.4\,\mathrm{s}$ in all cases. The column labelled $||error||$ shows the norm of the error between analytical (A_0, A_2) and numerical solution ($a_0(T)$, $a_2(T)$).

Solver	Time of calculation [s]	Ratio	$\|\|error\|\|$
MTSM	$1.352\,45 \times 10^{-3}$	–	2.7×10^{-15}
ode23	$3.921\,18 \times 10^{-1}$	**289.9**	7.2×10^{-10}
ode45	$2.943\,24 \times 10^{-2}$	**21.8**	8.3×10^{-11}
ode113	$1.229\,99 \times 10^{-2}$	**9.1**	1.4×10^{-11}

Table 4.24: Results of calculations for Fourier coefficients, (4.54).

Solver	Time of calculation [s]	Ratio	$\|\|error\|\|$
MTSM	6.809×10^{-4}	–	1.1×10^{-11}
ode23	$5.337\,41 \times 10^{-1}$	**783.9**	2.9×10^{-13}
ode45	$4.568\,95 \times 10^{-2}$	**67.1**	1.8×10^{-12}
ode113	$1.593\,32 \times 10^{-2}$	**23.4**	1.6×10^{-10}

Table 4.25: Results of calculations for Fourier coefficients (4.55).

The results show that the MTSM is faster and more accurate than all tested state-of-the-art solvers. The average order used by the MTSM is $ORD = 25 \pm 1$.

4.6 Lorenz system

The next example shows another non-linear system – Lorenz system. It was presented in [58]. This system explains some of the unpredictable behaviour of the weather. The Lorenz model supposes that a planet's atmosphere consists of a two-dimensional fluid cell heated from below and cooled from above [35]. The fluid motion can be described by the three-dimensional system of ODEs (4.56)

$$\begin{aligned} x' &= \sigma(y - x) \qquad & x(0) &= 1 \\ y' &= \rho x - y - xz \qquad & y(0) &= 1 \\ z' &= xy - \beta z \qquad & z(0) &= 1\,, \end{aligned} \tag{4.56}$$

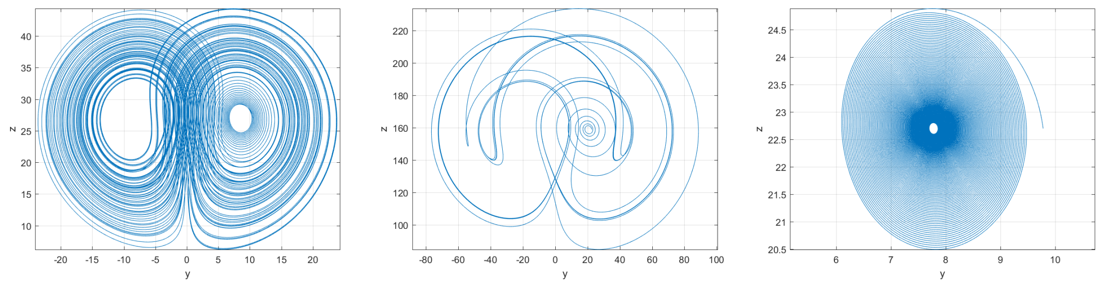
where σ is the Prandtl number, ρ is the Rayleigh number, and β is the parameter related to the physical size of the system. The behaviour of the system depends on the values of the parameters and initial conditions. Small changes in the initial conditions have a significant effect on the solution (and that is generally a reason why this model cannot be used to accurately predict the weather long-term). The (4.56) can be rewritten in the matrix-vector representation

$$\boldsymbol{y} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \boldsymbol{A} = \begin{pmatrix} -\sigma & \sigma & 0 \\ \rho & -1 & 0 \\ 0 & 0 & -\beta \end{pmatrix} \quad \boldsymbol{b} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \boldsymbol{B}_1 = \begin{pmatrix} 0 & 0 \\ -1 & 0 \\ 0 & 1 \end{pmatrix} \quad \boldsymbol{y}_{jj} = \begin{pmatrix} 1 & 3 \\ 1 & 2 \end{pmatrix} . \tag{4.57}$$

For the experiments, the parameters $\sigma = 10$ and $\beta = 8/3$ were fixed. Only the parameter ρ is changed to obtain different behaviour of the system (4.56). For $\rho = 28$, the value originally used by Lorenz [51]), the chaotic behaviour is observed. For large values of ρ, e.g. $\rho = 160$, the solution is periodic [40]. For $\rho = 23.7$, the solution is stable. Two equilibrium points can be calculated using (4.58). Initial conditions were then calculated by adding the constant vector $\boldsymbol{v} = (0, 2, 0)$ to the equilibrium point Q^+ [35], [34]

$$Q^{\pm} = (\pm\sqrt{\beta(\rho - 1)}, \pm\sqrt{\beta(\rho - 1)}, \rho - 1)\,. \tag{4.58}$$

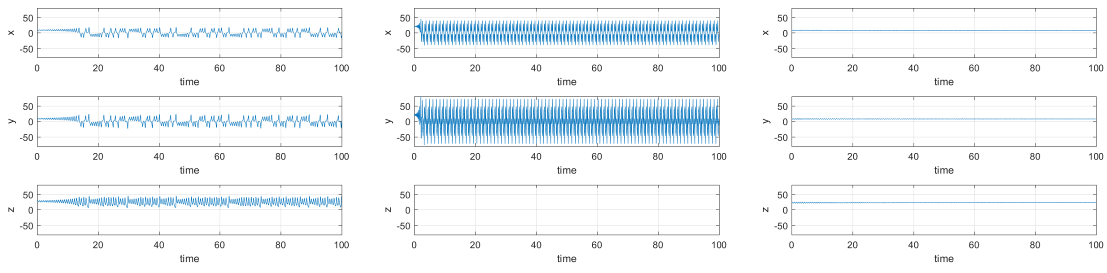
For the experiments with the Lorenz system, tolerances for all numerical solvers were set to 1×10^{-10}. The maximum simulation time was set to $t_{max} = 100\,\mathrm{s}$ for all experiments. Figure 4.34 shows the solution of the Lorenz system for different values of parameter ρ in the yz-plane.



(a) $\rho = 28$ (b) $\rho = 160$ (c) $\rho = 23.7$

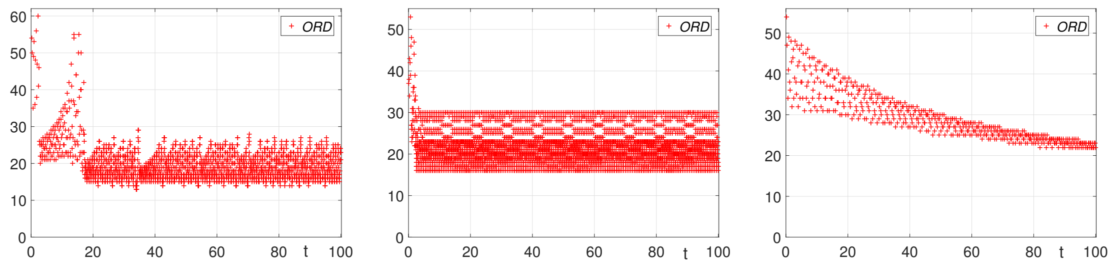
Figure 4.34: Behaviour of Lorenz system in yz-plane.

The solution in the time domain is in Figure 4.35 and the plot of the *ORD* function is in Figure 4.36.



(a) $\rho = 28$ (b) $\rho = 160$ (c) $\rho = 23.7$

Figure 4.35: Behaviour of Lorenz system in time domain.



(a) $\rho = 28$ (b) $\rho = 160$ (c) $\rho = 23.7$

Figure 4.36: Plots of the *ORD* function for different values of ρ.

Unlike linear problems, the values of ORD for non-linear problems are no longer nearly constant but can change quite rapidly during the calculation. Results of the simulation experiments are in Table 4.26.

	ode23		ode45		ode113		MTSM	
ρ	*ratio*	# steps	*ratio*	# steps	*ratio*	# steps	[s]	# steps
28	200.8	3993135	**7.3**	322496	**1.9**	19794	9.33×10^{-1}	2000
160	196.3	9907302	**6.9**	774340	**1.9**	48896	2.363	4000
23.7	15.7	1111529	**7.3**	147928	**4.6**	11029	5.38×10^{-1}	500

Table 4.26: Results of simulation experiments for Lorenz system.

4.7 Kepler problem

The final example in this Chapter of more general examples of the method is the Kepler problem. It was published in [78]. The Kepler problem describes the motion of a single planet around a heavy sun [18]. The problem is simplified so that the planet does not gravitationally influence the sun, and the sun is treated as stationary. The motion of the planet is also limited to the plane (it does not move up or down).

Let $y_1(t)$ and $y_2(t)$ denote rectangular coordinates centred at the sun, specifying at time t the position of the planet. Also let $y_3(t)$ and $y_4(t)$ denote a components of velocity in the y_1 and y_2 directions. If M denotes the mass of the sun, γ the gravitational constant, and m the mass of the planet, then the attractive force will have the magnitude

$$\frac{\gamma M m}{y_1^2 + y_2^2}\,.$$

The accelerations of the planet in y_1 and y_2 directions can be calculated as follows:

$$\frac{-\gamma M y_1}{(y_1^2 + y_2^2)^{\frac{3}{2}}}\,, \quad \frac{-\gamma M y_2}{(y_1^2 + y_2^2)^{\frac{3}{2}}}\,, \tag{4.59}$$

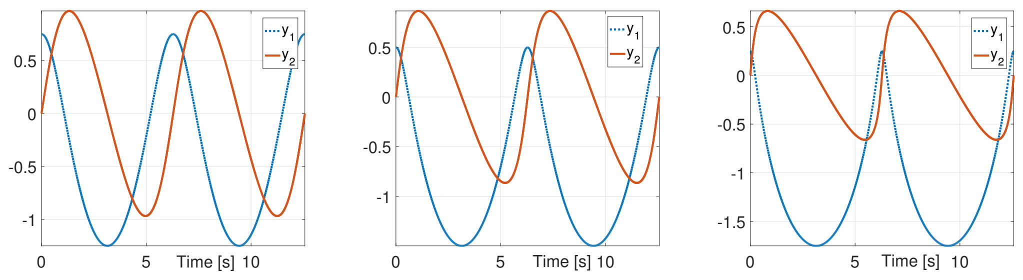
where the negative sign denotes the inward direction of the acceleration. By adjusting the scales of the variables, the factor γM can be removed. The motion can be represented by the following system of ODEs

$$\begin{aligned} y_1' &= y_3 & y_1(0) &= 1 - e \\ y_2' &= y_4 & y_2(0) &= 0 \\ y_3' &= \frac{-y_1}{r^3} & y_3(0) &= 0 \\ y_4' &= \frac{-y_2}{r^3} & y_4(0) &= \sqrt{\frac{1+e}{1-e}}\,, \end{aligned} \tag{4.60}$$

where e is an eccentricity of a rotating body and $r = \sqrt{y_1^2 + y_2^2}$. The analytical solution of (4.60) is defined by

$$(y_1 + e)^2 + \frac{y_2^2}{1 - e^2} - 1 = 0\,. \tag{4.61}$$

The change in time for the coordinates y_1 and y_2 for the different values of e is in Figure 4.37.



(a) $e = 0.25$ (b) $e = 0.5$ (c) $e = 0.75$

Figure 4.37: Kepler problem, solution.

To solve the system (4.60) using the MTSM with the recurrent calculation of the Taylor series' terms (3.20), the system of ODEs has to be transformed to the new autonomous equivalent system (3.19). The term $\frac{1}{r^3}$ has to be replaced by the equivalent generating ODEs that are added to (4.60). First, the term r^3 is substituted to give

$$\begin{aligned} y_5 &= \left(\sqrt{y_1^2 + y_2^2}\right)^3 = (y_1^2 + y_2^2)^{\frac{3}{2}} \\ y_5' &= \frac{3}{2}(y_1^2 + y_2^2)^{\frac{1}{2}}(2y_1y_1' + 2y_2y_2') = 3(y_1^2 + y_2^2)^{\frac{1}{2}}(y_1y_3 + y_2y_4)\,. \end{aligned}$$

Then the substitution of the term $(y_1^2 + y_2^2)^{\frac{1}{2}}$ is introduced to give

$$\begin{aligned} y_6 &= (y_1^2 + y_2^2)^{\frac{1}{2}} \\ y_6' &= \frac{1}{2}(y_1^2 + y_2^2)^{-\frac{1}{2}}(2y_1y_1' + 2y_2y_2') = (y_1^2 + y_2^2)^{-\frac{1}{2}}(y_1y_3 + y_2y_4) \\ &= \frac{(y_1y_3 + y_2y_4)}{(y_1^2 + y_2^2)^{\frac{1}{2}}} = \frac{(y_1y_3 + y_2y_4)}{y_6}\,. \end{aligned}$$

After the previous substitution, the terms that contain operation division (y_5^{-1} and y_6^{-1}) can be removed from the system using the following auxiliary ODEs

$$\begin{aligned} y_7 &= \frac{1}{y_5} = y_5^{-1} \\ y_7' &= -y_5^{-1}y_5' = -y_7^2y_5' = -y_7^2(3y_6(y_1y_3 + y_2y_4)) = -3y_6y_7^2(y_1y_3 + y_2y_4) \\ y_8 &= \frac{1}{y_6} = y_6^{-1} \\ y_8' &= -y_6^{-2}y_6' = -y_8^2y_6' = -y_8^2(y_8(y_1y_3 + y_2y_4)) = -y_8^3(y_1y_3 + y_2y_4)\,. \end{aligned}$$

The new autonomous system of ODEs with the substituted equations without operation division is obtained

$$\begin{aligned}
y_1' &= y_3 & y_1(0) &= 1-e \\
y_2' &= y_4 & y_2(0) &= 0 \\
y_3' &= -y_7 y_1 & y_3(0) &= 0 \\
y_4' &= -y_7 y_2 & y_4(0) &= \sqrt{\frac{1+e}{1-e}} \\
y_5' &= 3y_6(y_1y_3+y_2y_4) & y_5(0) &= (\sqrt{y_1(0)^2+y_2(0)^2})^3 \\
y_6' &= y_8(y_1y_3+y_2y_4) & y_6(0) &= \sqrt{y_1(0)^2+y_2(0)^2} \\
y_7' &= -3y_6y_7y_7(y_1y_3+y_2y_4) & y_7(0) &= \frac{1}{y_5(0)} \\
y_8' &= -y_8y_8y_8(y_1y_3+y_2y_4) & y_8(0) &= \frac{1}{y_6(0)}\,.
\end{aligned} \tag{4.62}$$

The new IVP (4.62) is equivalent to the original Kepler problem (4.60). Note that the new IVP (4.62) is an autonomous system of ODEs with operation multiplication in the form (3.19), and such system can be solved using the MTSM with recurrent calculation of the Taylor series terms using (3.20). The matrix-vector representation (3.19) of the system (4.62) is in Appendix F.1. The system of ODEs (4.62) can be further simplified by reducing the number of multiplications. By substituting $(y_1y_3 + y_2y_4)$ another auxiliary ODE can be obtained:

$$\begin{aligned}
y_9 &= y_1y_3 + y_2y_4 \\
y_9' &= y_1'y_3 + y_1y_3' + y_2'y_4 + y_2y_4' = y_3^2 + y_4^2 - y_1^2y_7 - y_2^2y_7\,.
\end{aligned}$$

Adding the new equation into (4.62) decreases the number of multiplications to four. The following system of ODEs is obtained:

$$\begin{aligned}
y_1' &= y_3 & y_1(0) &= 1-e \\
y_2' &= y_4 & y_2(0) &= 0 \\
y_3' &= -y_1 y_7 & y_3(0) &= 0 \\
y_4' &= -y_2 y_7 & y_4(0) &= \sqrt{\frac{1+e}{1-e}} \\
y_5' &= 3y_6y_9 & y_5(0) &= (\sqrt{y_1(0)^2+y_2(0)^2})^3 \\
y_6' &= y_8y_9 & y_6(0) &= \sqrt{y_1(0)^2+y_2(0)^2} \\
y_7' &= -3y_6y_7y_7y_9 & y_7(0) &= \frac{1}{y_5(0)} \\
y_8' &= -y_8y_8y_8y_9 & y_8(0) &= \frac{1}{y_6(0)} \\
y_9' &= y_3y_3 + y_4y_4 - y_7(y_1y_1+y_2y_2) & y_9(0) &= y_1(0)y_3(0) + y_2(0)y_4(0)\,.
\end{aligned} \tag{4.63}$$

The new IVP (4.63) is again equivalent to the original Kepler problem (4.60) and with (4.62). The matrix-vector representation of the ODEs system (4.63) using the notation of

(3.19) is in Appendix F.2. The number of multiplications can be decreased even further by simplifying the remaining ODEs with four multiplications. The multiplication $y_6 y_9$ can be substituted by an auxiliary ODE:

$$\begin{aligned} y_{10} &= y_6 y_9 \\ y_{10}' &= y_6' y_9 + y_6 y_9' = y_8 y_9 y_9 + y_6 y_3^2 + y_6 y_4^2 - y_6 y_7 (y_1^2 + y_2^2)\,. \end{aligned}$$

A new multiplication ($y_8 y_9$) in the new auxiliary ODE can also be substituted to give

$$\begin{aligned} y_{11} &= y_8 y_9 \\ y_{11}' &= y_8' y_9 + y_8 y_9' = -y_8^3 y_9^2 + y_8 (y_3^2 + y_4^2 - y_7 (y_1^2 + y_2^2)) \\ &= -y_8 y_{11}^2 + y_8 (y_3^2 + y_4^2 - y_7 (y_1^2 + y_2^2))\,. \end{aligned}$$

Finally, the bracket $(y_1^2 + y_2^2)$ can be substituted to give

$$\begin{aligned} y_{12} &= y_1^2 + y_2^2 \\ y_{12}' &= 2 y_1 y_1' + 2 y_2 y_2' = 2 y_1 y_3 + 2 y_2 y_4\,. \end{aligned}$$

Adding the three new auxiliary ODEs into (4.63) decreases the number of multiplications to three

$$\begin{aligned}
y_1' &= y_3 & y_1(0) &= 1 - e \\
y_2' &= y_4 & y_2(0) &= 0 \\
y_3' &= -y_1 y_7 & y_3(0) &= 0 \\
y_4' &= -y_2 y_7 & y_4(0) &= \sqrt{\frac{1+e}{1-e}} \\
y_5' &= 3 y_{10} & y_5(0) &= \left(\sqrt{y_1(0)^2 + y_2(0)^2}\right)^3 \\
y_6' &= y_{11} & y_6(0) &= \sqrt{y_1(0)^2 + y_2(0)^2} \\
y_7' &= -3 y_7 y_7 y_{10} & y_7(0) &= \frac{1}{y_5(0)} \\
y_8' &= -y_8 y_8 y_{11} & y_8(0) &= \frac{1}{y_6(0)} \\
y_9' &= y_3 y_3 + y_4 y_4 - y_7 y_{12} & y_9(0) &= y_1(0) y_3(0) + y_2(0) y_4(0) \\
y_{10}' &= y_9 y_{11} + y_3 y_3 y_6 + y_4 y_4 y_6 - y_6 y_7 y_{12} & y_{10}(0) &= y_6(0) y_9(0) \\
y_{11}' &= -y_8 y_{11} y_{11} + y_3 y_3 y_8 + y_4 y_4 y_8 - y_7 y_8 y_{12} & y_{11}(0) &= y_8(0) y_9(0) \\
y_{12}' &= 2 y_1 y_3 + 2 y_2 y_4 & y_{12}(0) &= y_1(0)^2 + y_2(0)^2\,.
\end{aligned} \tag{4.64}$$

The new IVP (4.64) is equivalent to all previous systems ((4.60), (4.62) and (4.63)). The matrix-vector representation of the ODEs system (4.64) in the (3.19) notation is in Appendix F.3.

The results for (4.62), (4.63), (4.64) using MTSM with fixed integration step size $h_{0.25} = \pi/25\,\mathrm{s}$, $h_{0.5} = \pi/50\,\mathrm{s}$, $h_{0.75} = \pi/100\,\mathrm{s}$ and $e = 0.25$, $e = 0.5$, $e = 0.75$ are in Tables 4.27, 4.28 and 4.29, respectively. The tolerances for all solvers were set to obtain the $||error|| = 10^{-8}$, where the $||error||$ is computed as the norm of (4.61). The maximum time of the simulation is set to 2 cycles, $t_{max} = 4\pi\,\mathrm{s}$. The obtained ratios indicate faster computation of the MTSM in most cases.

e	ode23 *ratio*	ode45 *ratio*	ode113 *ratio*	MTSM [s]
0.25	**146.4**	**4.7**	**1.3**	2.93×10^{-2}
0.5	**122.8**	**4.1**	**1.0**	5.36×10^{-2}
0.75	**90.4**	**3.0**	0.7	1.158×10^{-1}

Table 4.27: Results for Kepler problem (4.62).

e	ode23 *ratio*	ode45 *ratio*	ode113 *ratio*	MTSM [s]
0.25	**213.9**	**6.8**	**1.8**	2.05×10^{-2}
0.5	**174.4**	**5.7**	**1.5**	3.77×10^{-2}
0.75	**145.1**	**4.8**	**1.2**	7.3×10^{-2}

Table 4.28: Results for Kepler problem (4.63).

e	ode23 *ratio*	ode45 *ratio*	ode113 *ratio*	MTSM [s]
0.25	**312.2**	**12.1**	**2.5**	1.44×10^{-2}
0.5	**262.5**	**9.1**	**2.1**	2.63×10^{-2}
0.75	**221.3**	**7.4**	**1.7**	5.09×10^{-2}

Table 4.29: Results for Kepler problem (4.64).

The autonomous system with a low number of multiplications (4.64) is the fastest one, see Table 4.29. The greater number of multiplications slows down the calculation considerably. The ratios between the slowest system (4.62) and the systems of ODEs (4.63) and (4.64) for different values of e are in Table 4.30.

e	MTSM (4.62)/(4.63) *ratio*	MTSM (4.62)/(4.64) *ratio*
0.25	1.43	2.03
0.5	1.42	2.04
0.75	1.59	2.28

Table 4.30: Comparison between times of calculation of autonomous system (4.62) and systems (4.63) and (4.64).

Table 4.30 shows that the system with three-term multiplication (4.64) is approximately two times faster than the system with five-term multiplication (4.62). The difference is caused by the large number of multiplications of higher-order terms in the Taylor series. Optimizations from Subsection 3.3.2 are not used in the experiments, (3.20) is used directly. Table 4.31 shows the number of integration steps for different values of e.

	Number of steps			
e	ode23	ode45	ode113	MTSM
0.25	105180	8916	419	100
0.5	161323	12148	654	200
0.75	262631	19288	955	400

Table 4.31: Number of integration steps, Kepler problem (4.64).

The number of integration steps grows with increasing values of the constant e due to the Kepler problem becoming stiff. Explicit numerical methods have difficulties with such systems. The stiffness indicator σ for the time normalized Kepler problem with different values of constant e is in Figure 4.38 [73].

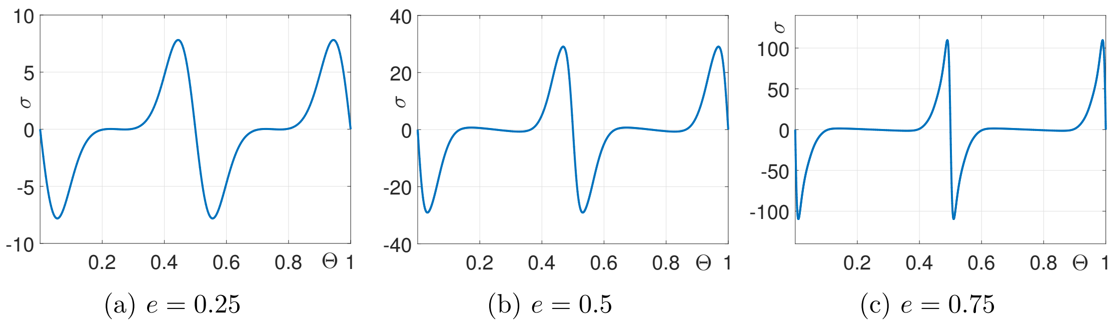

(a) $e = 0.25$ (b) $e = 0.5$ (c) $e = 0.75$

Figure 4.38: Stiffness indicator σ.

MTSM automatically detects the stiffness in the system and uses larger order (more terms of the Taylor series) in rapidly changing parts of solutions, see Figure 4.37 and Figure 4.39.

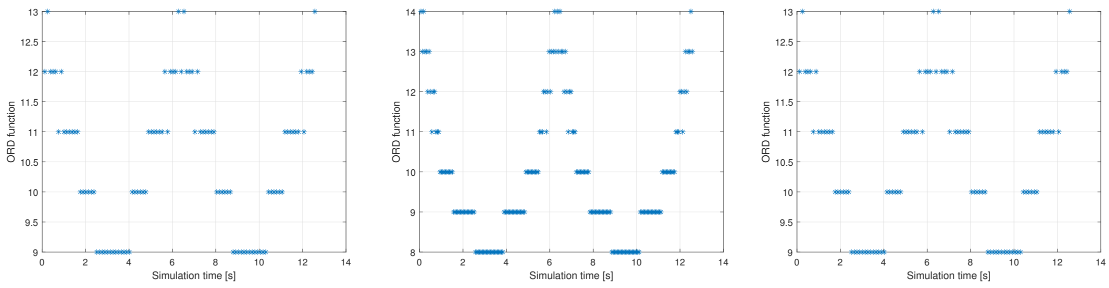
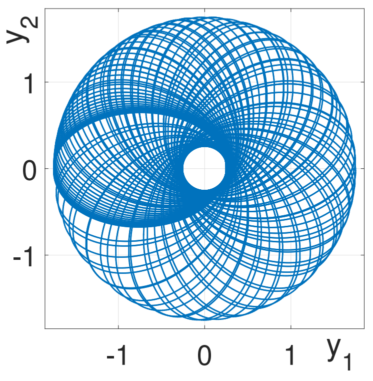
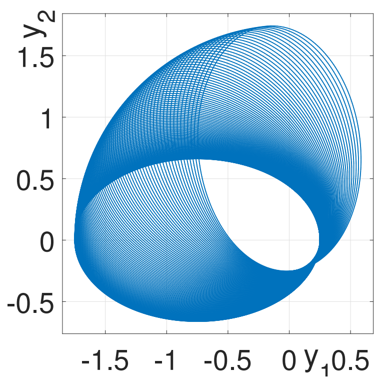

(a) $e = 0.25$ (b) $e = 0.5$ (c) $e = 0.75$

Figure 4.39: Plot of the ORD function.

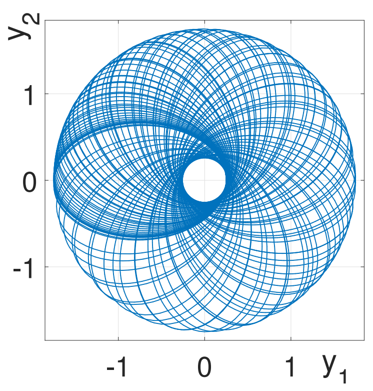
The last experiment shows that the solution of the Kepler problem becomes unstable using state-of-the-art solvers. The default tolerance $TOL = 1 \times 10^{-6}$ was set for all solvers. Results for $t_{max} = 200 \cdot \pi$ s and $e = 0.75$ are in Figure 4.40. Only MTSM provides a stable solution.



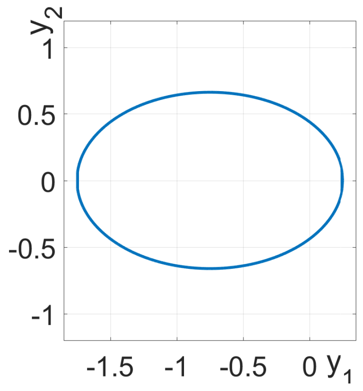
(a) ode23



(b) ode45



(c) ode113



(d) MTSM

Figure 4.40: Result of the circle test y_1y_2 plane.

The Kepler problem shows the possible problems and complexity when using the proposed method for solving non-linear systems. However, it is necessary because when modelling real-world problems, their models contain non-linearities. These might be removed by linearizing the model, but it might degrade the parameters or other properties.

4.8 Concluding remarks

This Chapter compares the behaviour of the method introduced in Chapter 3 with the state-of-the-art numerical integration solvers on a set of selected real-world problems. These problems serve as benchmarks to test the performance of the method. In additional experiments in Chapter 6, I will use the background and insight gained on these benchmark problems and I will apply the properties that seem to be favourable in the area of control system implementation with real-time considerations.

Chapter 5

Control theory

This Chapter briefly introduces important topics from control theory so that they can be used in the experiments. The content of this chapter has been mainly compiled from [24], [11], [23], [72], [16], [26], [84], and [31]. Other used sources are cited throughout where needed.

Control theory has helped shape the modern technological and industrial landscape [16]. Some examples include cruise control in cars, position control in construction equipment, packet routing on the Internet and PID temperature and pressure control in modern espresso machines. In the future, control systems are going to be increasingly applied to high-dimensional and strongly non-linear problems (for example, autonomous robots and self-driving cars).

5.1 Control systems

In this Section, the several types of control systems that are widely used are going to be discussed. There are several ways to categorize the control system and approaches to their design. One particularly useful categorization is by the way these control systems use energy and resources and how they can adapt to the change in the input and output of the controlled system [16].

The simplest form are the *passive* control systems (for example a speed limit sign or stop sign at the street intersection) that can change the traffic flow. The second very broad category is *active* control that does require input energy to operate.

These can again be split into several categories based on if they use sensors to augment or inform how the controller functions. For *open-loop* systems with no sensors, the pre-programmed control sequence is carried out. The *sensor based* control system can either combine open-loop control with feedback or *closed-loop* feedback control systems. These measure the system directly and shapes the control response to achieve the desired goal. These are going to be used in this thesis. The categories are visualized in Figure 5.1.

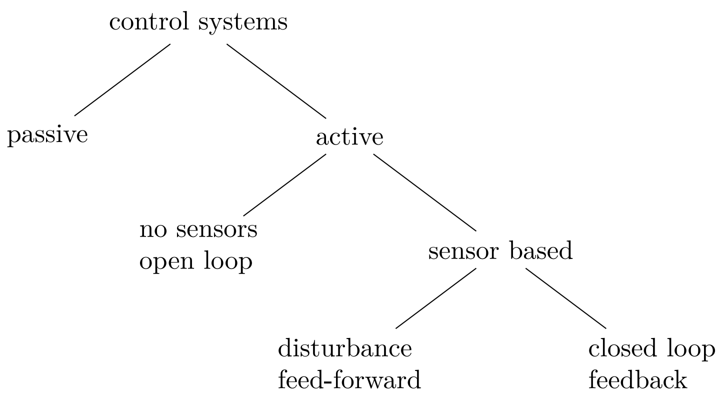


Figure 5.1: Types of the control systems (based on [16]).

The schematic of the open-loop control system is shown in Figure 5.2.

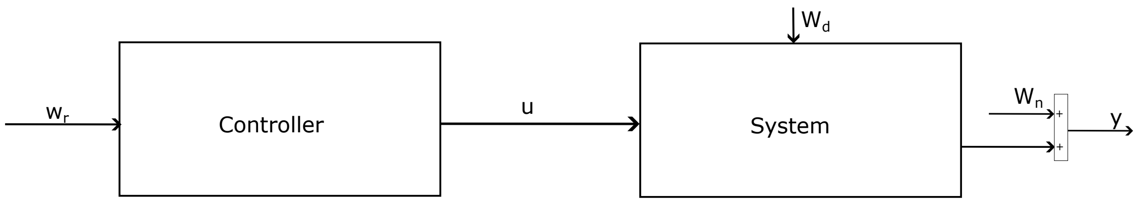


Figure 5.2: Schematic for the open-loop system (based on [16], [24]).

In Figure 5.2, w_r represents the reference signal. Given this signal, the controller, using a control law, constructs a control input u to drive a system. External disturbances w_d and sensor noise w_n are not accounted for and degrade performance. The open-loop control uses a model of the system to design an actuation signal u that produces the desired output. However, as stated before, it can only do so based on a pre-planned strategy and cannot account for external disturbances or any change in the system dynamics.

The general scheme of the feedback control is in Figure 5.3.

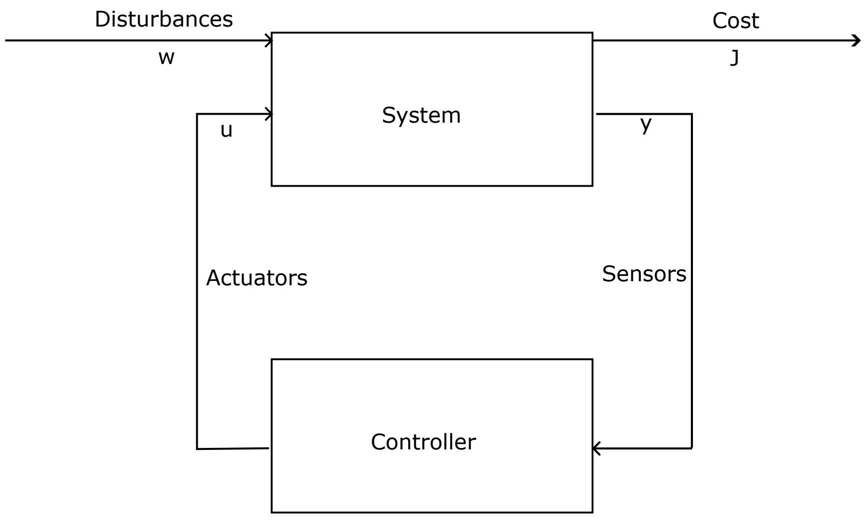


Figure 5.3: General scheme for feedback control system (based on [16]).

Sensor measurements $\boldsymbol{y}$ or a system are fed back into a controller that generates the actuation signal $\boldsymbol{u}$. This signal can manipulate dynamics and change the behaviour of the controlled system. It performs better than the open-loop control system and can better handle uncertainties and disturbances.

The vector of disturbances $\boldsymbol{w}$ can be decomposed as $\boldsymbol{w} = \left(\boldsymbol{w}_d^T \boldsymbol{w}_n^T \boldsymbol{w}_r^T\right)^T$ where

- $\boldsymbol{w}_d$ are the disturbances of the state of the system,
- $\boldsymbol{w}_n$ is the measurement noise,
- $\boldsymbol{w}_r$ is the reference that should be tracked in the closed-loop system.

The system can be generally described using the following ODE:

$$\boldsymbol{x}' = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w_d}) \tag{5.1}$$

and the output measured by the sensors described by

$$\boldsymbol{g} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w_n}) \,. \tag{5.2}$$

The control law is then generally described using a function $\boldsymbol{u}$

$$\boldsymbol{u} = \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{w_r}) \tag{5.3}$$

and it should (if the controller can do so) minimize the cost function J

$$J \triangleq J(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w_r}) \,. \tag{5.4}$$

Note that optimizing the cost of control is not a focus of this thesis. The schematic of the feedback control system is in Figure 5.4.

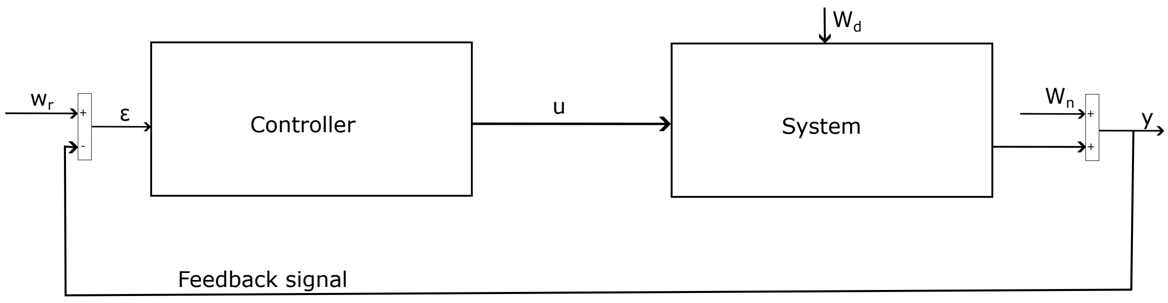


Figure 5.4: Closed-loop feedback control system (based on [16], [24]).

Figure 5.4 shows the sensor signal $\boldsymbol{y}$ fed back and subtracted from the reference signal $\boldsymbol{w_r}$. This provides information about how the system reacts to the control input (the actuation being performed). The controller uses the resulting error $\boldsymbol{\epsilon}$ to help determine the correct actuation signal $\boldsymbol{u}$ for the desired response. This kind of control system can effectively stabilize unstable dynamics while rejecting disturbances $\boldsymbol{w_d}$ and reducing noise $\boldsymbol{w}_n$.

5.2 Linear systems

According to [16], [24], and others, the more comprehensive theory of control systems and their representations has been developed for *linear systems.*

Due to the fact that real systems are not generally linear, the non-linear representation has to be *linearized* about a fixed point around which the chosen linear approximation is valid.

A system is defined as linear in terms of the input and response [24]. Generally, a necessary condition for any linear system can be determined based on excitation and response. So when a system receives an input $a_1(t)$, it responds with $y_1(t)$. And for an input $a_2(t)$, it responds with $y_2(t)$. For linear system , the excitation $a_1(t) + a_2(t)$ results in response $y_1(t) + y_2(t)$ – *principle of superposition.*

A linear system has to also be *homogeneous*, that is, when supplied with an input $a(t)$, the result is $y(t)$. Then when multiplying the input by a constant $a(t) \cdot \beta$, the response must be equal to the output being multiplied by the same constant $y(t) \cdot \beta$.

5.2.1 Laplace transformation

The *Laplace transformation* can be applied to linear systems. The method substitutes differential equations that are difficult to solve by algebraic equations [24]. The solution of the system is obtained by performing the following steps:

1. obtain linearized differential equations,
2. perform the Laplace transformation on the differential equations,
3. solve the resulting algebraic equations.

For function $f(t)$ to be transformable, it is sufficient for

$$\int_{0^-}^{\infty} |f(t)| e^{-\sigma_1 t} \, \mathrm{d}t < \infty,$$

for some real, positive σ_1. The 0^- indicates that the integral includes discontinuities, such as delta function at $t = 0$. If the magnitude of $f(t)$ is

$$|f(t)| < M e^{\alpha t}$$

for all positive t, the integral will converge for $\sigma_1 > \alpha$. Real physical signals always have Laplace transform. The *Laplace transformation* for a function of time $f(t)$ is

$$F(s) = \int_{0^-}^{\infty} f(t) e^{-st} \, \mathrm{d}t,$$

and the *inverse Laplace transform* is written as

$$f(t) = \frac{1}{2\pi j} \int_{\sigma - j\infty}^{\sigma + j\infty} F(s) e^{st} \, \mathrm{d}s \, .$$

The pairs of $f(t)$ and $F(s)$ are in [24] and there are mathematical models that represent linear systems using the transformed equations (transfer function modelling for example, see [24]). However, this thesis deals with the solution of ordinary differential equations, and the Laplace variable s can be considered to be a *differential operator* [24]

$$s \equiv \frac{d}{dt} \, . \tag{5.5}$$

5.2.2 State-space representation

The state-space representation of the system is very useful in the context of this thesis because it uses ordinary differential equations to represent the changes between individual states. Due to this fact, it is going to be discussed in greater detail.

The *state* of a system is a set of variables whose values, together with the input signals and the equations describing the dynamics, will provide the future state and the output of the system [24]. For the dynamic systems, the state of the system is described by a set of variables $\boldsymbol{x}(t) = (x_1(t), x_2(t), \dots, x_n(t))$. The state variables determine the future behaviour of the system when a present state of the system and its inputs are known.

The changes of the states can be described using *state differential equations*, which are the first-order ODEs for the state variables $(x_1, x_2, \dots, x_n)$ and the inputs $(u_1, u_2, \dots, u_n)$. These equations can be written in a form

$$\begin{aligned} x_1'(t) &= a_{11}x_1(t) + a_{12}x_2(t) + \dots + a_{1n}x_n(t) + e_{11}u_1(t) + \dots + e_{1m}u_m(t) \\ x_2'(t) &= a_{21}x_1(t) + a_{22}x_2(t) + \dots + a_{2n}x_n(t) + e_{21}u_1(t) + \dots + e_{2m}u_m(t) \\ &\vdots \\ x_n'(t) &= a_{n1}x_1(t) + a_{n2}x_2(t) + \dots + a_{nn}x_n(t) + e_{n1}u_1(t) + \dots + e_{nm}u_m(t)\,. \end{aligned} \tag{5.6}$$

This system of ODEs can be expressed in the matrix-vector notation

$$\begin{pmatrix} x_1'(t) \\ x_2'(t) \\ \vdots \\ x_n'(t) \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \dots & \dots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{pmatrix} + \begin{pmatrix} e_{11} & \dots & e_{1m} \\ \vdots & & \vdots \\ e_{n1} & \dots & e_{nm} \end{pmatrix} \begin{pmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{pmatrix} \tag{5.7}$$

where $\boldsymbol{x}(t)$ represents the state vector and $\boldsymbol{u}(t)$ represents the input function. The *state differential equation* or *state equation* can be therefore written as

$$\boldsymbol{x}'(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{E}\boldsymbol{u}(t)\,. \tag{5.8}$$

The outputs of the systems can be related to the inputs and the states by *output equation*, which can be written as

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t)\,. \tag{5.9}$$

The sensor noise and disturbances can also be modelled using the state-space model. The state equation and the output equation then become

$$\begin{aligned} \boldsymbol{x}'(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{E}\boldsymbol{u}(t) + \boldsymbol{G}\boldsymbol{w_d}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) + \boldsymbol{w_n}(t)\,, \end{aligned} \tag{5.10}$$

where $\boldsymbol{G}$ is the gain matrix for the disturbance $\boldsymbol{w_d}$ and $\boldsymbol{w_n}$ is the sensor noise that can affect the measurements. The schematic for this representation is in Figure 5.5.

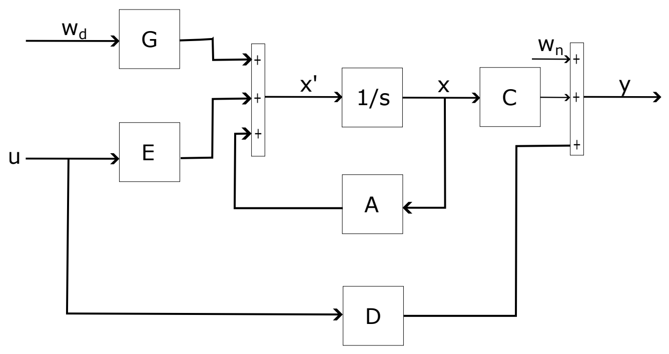


Figure 5.5: Schematic of the state-space representation with disturbances and sensor noise.

5.2.3 Controllability and observability

The concepts of *controllability* and *observability* of the system have to be introduced to understand if the stable eigenvalues of the system can even be obtained. Note that more information about linear algebra, eigenvalues and eigenvectors can be obtained from [50].

Controllability

The controllability of a system can be defined as [24]

Definition 2 *A system is completely controllable if there exists an unconstrained control $u(t)$ that can transfer any initial state $\boldsymbol{x}(t_0)$ to any other desired location $\boldsymbol{x}(t)$ in a finite time.*

To determine if the linear system described by (5.8) is controllable, the controllability matrix $\mathcal{C}$ has to be constructed

$$\mathcal{C} = \begin{pmatrix} \boldsymbol{B} & \boldsymbol{AE} & \boldsymbol{A}^2\boldsymbol{E} & \dots & \boldsymbol{A}^{n-1}\boldsymbol{E} \end{pmatrix} . \tag{5.11}$$

If the matrix $\mathcal{C}$ has n linearly independent columns (it spans $\mathcal{R}^n$), then (5.8) is controllable. The span of the columns of the controllability matrix $\mathcal{C}$ forms a *Krylov subspace*. This subspace determines which state vector directions in $\mathcal{R}^n$ may be manipulated with control. Therefore, if a system is controllable, it also implies that any state $\rho \in \mathcal{R}^n$ is reachable in finite time by some actuation signal $\boldsymbol{u}(t)$. The eigenvalues might be placed arbitrarily. However, some controllers have additional considerations.

Observability

The observability of a system can be defined as [24]

Definition 3 *The system is completely observable if and only if there exists a finite time T such that the initial state $\boldsymbol{x}(0)$ can be determined from the observation history $\boldsymbol{y}(t)$ given the control $u(t)$.*

Observability does not differ from controllability much but the interpretation is different. A system is observable if it is possible to estimate any state $\rho \in \mathcal{R}^n$ from a history of measurements of the system output $\boldsymbol{y}(t)$. The observability is determined by the row space of the *observability matrix* $\mathcal{O}$

$$\mathcal{O} = \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \boldsymbol{CA}^2 \\ \vdots \\ \boldsymbol{CA}^{n-1} \end{pmatrix} . \tag{5.12}$$

The observability comes into play when full-state measurements of the state vector $\boldsymbol{x}$ are not available, and it is necessary to estimate the vector from the measurements. This is only possible when a system is observable. The estimators are not used in this thesis, but observability is still an important property to be taken into account.

5.2.4 Linearization

This section is mainly derived from [24] and [15]. In real-world conditions, most systems are operated around a *operating point* (or points), for example, the range of temperatures, angles of attack and many others that depend on the plant or application being considered. This means that most of the very complicated equations (some examples are going to be shown in Chapter 6) that would be valid for all possible values are just valid for the smaller subset of these values around the operating point. Therefore, the question of linearity and the range of applicability must be considered for each system.

To linearize the non-linear system written as

$$\begin{aligned} x' &= f(\boldsymbol{x}, u) \\ y &= h(\boldsymbol{x}, u)\,, \end{aligned} \tag{5.13}$$

where f and h are non-linear functions, $\boldsymbol{x}$ is the state vector, u is a scalar control input and y is the scalar system output, the following procedure can be used. Given the non-linear system specified by these functions and an equilibrium point $\boldsymbol{x^*} = (x_1^* \dots x_n^*)^T$, obtained when $u = u^*$, a coordinate transformation can be defined to transform the state variables to the equilibrium point. Denote $\boldsymbol{\delta x} = \boldsymbol{x} - \boldsymbol{x^*}$

$$\delta \boldsymbol{x} = \begin{pmatrix} \delta x_1 \\ \vdots \\ \delta x_n \end{pmatrix} = \begin{pmatrix} x_1 - x^* \\ \vdots \\ x_n - x_n^* \end{pmatrix} , \tag{5.14}$$

$\delta u = u - u^*$ and $\delta y = y - h(\boldsymbol{x}^*, u^*)$. The new coordinates $\boldsymbol{\delta x}$, δu and δy represent the differences of $\boldsymbol{x}$, u and y from their equilibrium values. They serve as the new state variables $\boldsymbol{\delta x}$, new control input δu and the new output δy. The linearization of (5.13) at the equilibrium point $\boldsymbol{x}^*$ is given by

$$\begin{aligned} \boldsymbol{\delta x}' &= \boldsymbol{A\delta x} + \boldsymbol{E}\delta u \\ \boldsymbol{y} &= \boldsymbol{C\delta x} + \boldsymbol{D}\delta u\,, \end{aligned} \tag{5.15}$$

where

$$
\begin{aligned}
\boldsymbol{A} = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)_{x^*,u^*} &= \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x_1^*,\dots,x_n^*,u^*) & \dots & \frac{\partial f_1}{\partial x_n}(x_1^*,\dots,x_n^*,u^*) \\ \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x_1}(x_1^*,\dots,x_n^*,u^*) & \dots & \frac{\partial f_n}{\partial x_n}(x_1^*,\dots,x_n^*,u^*) \end{pmatrix}, \\
\boldsymbol{E} = \left(\frac{\partial \boldsymbol{f}}{\partial u}\right)_{x^*,u^*} &= \begin{pmatrix} \frac{\partial f_1}{\partial u}(x_1^*,\dots,x_n^*,u^*) \\ \vdots \\ \frac{\partial f_n}{\partial u}(x_1^*,\dots,x_n^*,u^*) \end{pmatrix}, \\
\boldsymbol{C} = \left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right)_{x^*,u^*} &= \left(\frac{\partial h}{\partial x_1}(x_1^*,\dots,x_n^*,u^*) \dots \frac{\partial h}{\partial x_n}(x_1^*,\dots,x_n^*,u^*)\right), \\
\boldsymbol{D} = \left(\frac{\partial \boldsymbol{h}}{\partial u}\right)_{x^*,u^*} &.
\end{aligned}
\tag{5.16}
$$

5.2.5 Equilibrium points of the system

The state $x^* \in \mathcal{R}^n$ is a equilibrium point for $x' = f(x)$ if

$$f(x^*) = 0 \tag{5.17}$$

for all $t \geq 0$. Note that if the system has an initial condition $x(0)$ equal to the equilibrium, it will stay in that equilibrium forever (the state of the system does not change).

5.3 Non-linear systems

Most control systems in real-world applications are linear (because many systems operate close to the set operating point and the behaviour around this point can be considered linear). In some circumstances (for example, when the model of the system is uncertain or when the non-linear parameters have to be taken into account [70]), to be able to develop a non-linear controller that would be effective and comparable in speed and simplicity of design with the linear controller. Due to the recent advances in micro-controller design and general power of computer system, non-linear control is becoming more common in interesting use cases. And because the method presented in this thesis shows good behaviour for non-linear problems, it would be a shame not to try to test at least some non-linear controllers and designs in the thesis to determine if the method can be used even in this context.

This section is mainly compiled from existing sources ([36], [70], [11]) that deal with non-linear control.

5.3.1 Common non-linear system behaviours

Several behaviours of non-linear systems are common and are going to be encountered in Chapter 6 if appropriate. These include:

- multiple equilibrium points,
- limit cycles (oscillation of fixed period and amplitude without external stimuli),
- bifurcations (values of system parameters at which the system properties change),
- chaos (the system output is extremely sensitive to the initial conditions).

5.3.2 Representation of the non-linear systems

The dynamic of the non-linear systems can be represented by the set of differential equations

$$\begin{aligned}\boldsymbol{x}'(t) &= f(\boldsymbol{x}(t), \boldsymbol{u}(t))\\ \boldsymbol{y}(t) &= g(\boldsymbol{x}(t), \boldsymbol{u}(t))\end{aligned}$$

where $\boldsymbol{x}$ is the state-vector. This representation is a generalized case of the linear system representation, defined in Subsection 5.2.2.

5.3.3 Lyapunov function, stability

We can use the Lyapunov function to prove the stability properties of a given function. The Lyapunov function is a generalized energy function that shows the stability around the equilibrium points. The method shows the stability properties without finding the trajectories of the system. The origin of the system is stable equilibrium if for each $\epsilon > 0$, there exists $\delta > 0$ such that $||x(t)|| < \delta \rightarrow ||x(t)|| < \epsilon \, \forall t \geq 0$. To define the stability of the function using the *Lyapunov stability theorem*, several other sets and function characteristics have to be defined.

The set $S_r(\mathbf{0})$ contains all points $\boldsymbol{x} \in \mathcal{R}^n$ which are strictly inside a ball of radius r around the origin

$$S_r(\mathbf{0}) = \boldsymbol{x} \in \mathcal{R}^n, ||\boldsymbol{x}|| < r\,.$$

In a 2D space, this can be represented as a circle with radius r, with the centre around the origin. A continuous function V is *positive definite* on $S_r(\mathbf{0})$ if

- $V(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in S_r(\mathbf{0}), \boldsymbol{x} \neq \mathbf{0}$ and
- $V(\mathbf{0}) = 0$.

A continuous function V is *negative definite* on $S_r(\mathbf{0})$ if

- $V(\boldsymbol{x}) < 0$ for all $\boldsymbol{x} \in S_r(\mathbf{0}), \boldsymbol{x} \neq \mathbf{0}$ and
- $V(\mathbf{0}) = 0$.

A continuous function V is *positive semidefinite* on $S_r(\mathbf{0})$ if

- $V(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x} \in S_r(\mathbf{0})$ and
- $V(\mathbf{0}) = 0$.

A continuous function V is *negative semidefinite* on $S_r(\mathbf{0})$ if

- $V(\boldsymbol{x}) \leq 0$ for all $\boldsymbol{x} \in S_r(\mathbf{0})$ and
- $V(\mathbf{0}) = 0$.

Lyapunov Stability Theorem states that the origin is a *stable equilibrium* of $\boldsymbol{x}' = f(\boldsymbol{x})$ if there exists $r > 0$ and a positive definite function $V(x)$ on $S_r(\mathbf{0})$ such that V' is negative semidefinite on $S_r(\mathbf{0})$. If there exists a locally positive definite function $V(x)$, such that V' is locally negative semidefinite, the origin is *stable.* If there exists a locally positive definite function $V(x)$, such that V' is locally negative definite, the origin is *asymptotically stable.*

5.3.4 LaSalle's Invariance Principle

LaSalle's Invariance Principle can be used instead of the Lyapunov stability theorem to determine the asymptotic stability of the system when V' is negative semidefinite. To define the principle, first consider the set ω is *positively invariant* with respect to $\boldsymbol{x}' = f(\boldsymbol{x})$ if $\boldsymbol{x}(0) \in \omega$. This implies that $\boldsymbol{x}(t) \in \omega \, \forall t \leq 0$. The LaSalle's Invariance Principle states that

- Let ω be a compact set that is *positively invariant* with respect to $\boldsymbol{x}' = f(\boldsymbol{x})$.
- Let the function V be a continuously differentiable function on ω such that $V'(\boldsymbol{x}) \leq 0$.
- Let $V \subset \omega$ be the set of all points in ω such that $V'(\boldsymbol{x}) = 0$.
- let M be the *largest positively invariant* set in E.

Then every solution starting in ω approaches M as $t \to \infty$.

5.4 Types of controllers

In this Section, several types of controllers are introduced that are going to be used in Chapter 6.

5.4.1 PID controllers

One of the most common types of controllers routinely applied in technical practice is the *PID* (proportional-integral-derivative) controller. The equation for this controller can generally be written as

$$u(t) = k_p \epsilon(t) + k_i \int_0^t \epsilon(\tau) \, \mathrm{d}\tau + k_d \epsilon(t)' \,, \tag{5.18}$$

where $\epsilon(t)$ is the error of the closed-loop feedback, k_p is the proportional gain of the controller, k_i is the integral gain of the controller and k_d is the derivative gain of the controller. The proportional part of the controller scales linearly with the error and the integral part accumulates the error signal over time. The derivative part of the controller scales with the derivative of the error. This type of controller tends towards minimizing the error ϵ and thus reduces overshoot.

Note that there are several variants of the PID controllers based on what parts of the controller are used. Most notable include:

- P (with only proportional term),
- PD (proportional-derivative terms),
- PI (proportional-integration terms) and
- PID.

The PID controller connected to the closed-loop feedback is in Figure 5.6.

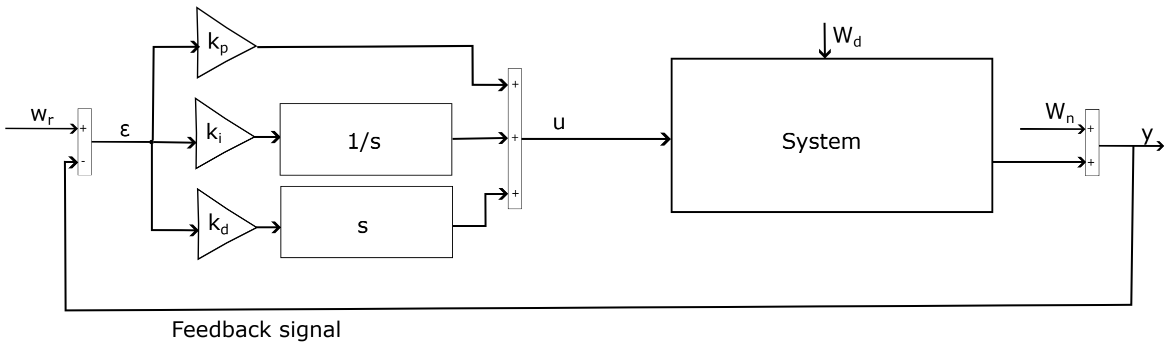


Figure 5.6: PID controller connected in the closed-loop feedback [24].

Transformation of a definite integral to the ordinary differential equation

Generally, a definite integral

$$F(T) = \int_0^T f(t)\,\mathrm{d}t^{1}\,,$$

can be transformed into an IVP [75]

$$F'(t) = f(t) \qquad F(0) = 0\,,$$

with the solution of the integral being obtained at $t = T$ s. The definite integral $F(T) = \int_0^T \epsilon(\tau)\,\mathrm{d}\tau$ from (5.18) can therefore be transformed as

$$F'(t) = \epsilon(\tau) \qquad F(0) = 0$$

with $t_{max} = t$ s.

5.4.2 Full-state closed-loop feedback controller

With full-state feedback available, the original state equation (5.8) (rewritten without time and disturbances for simplicity)

$$\boldsymbol{x}' = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{E}\boldsymbol{u}$$

the feedback signal $\boldsymbol{u}$ can be expressed as

$$\boldsymbol{u} = -\boldsymbol{k}\boldsymbol{x}\,, \tag{5.19}$$

where $\boldsymbol{k} = (k_1, k_2, \ldots, k_m)$, where m is the number of rows of $\boldsymbol{A}$. Therefore

$$\boldsymbol{x}' = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{E}(-\boldsymbol{k}\boldsymbol{x}) = (\boldsymbol{A} - \boldsymbol{E}\boldsymbol{k})\boldsymbol{x}\,.$$

The new $\boldsymbol{A}$ matrix of the system $\boldsymbol{A} - \boldsymbol{E}\boldsymbol{k}$ has stable eigenvalues (and the resulting system is therefore stable). To be able to construct the control law using this principle, the $\boldsymbol{A}, \boldsymbol{E}$ are controllable. The properties of the resulting controller can be tuned by the values of $\boldsymbol{k}$. Several approaches exist to determine the values of $\boldsymbol{k}$. The least robust one is to place the eigenvalues of $\boldsymbol{A}$ in the appropriate place in the left-hand part of the complex plane, so called *pole-placement method.*

To place the poles and therefore change the eigenvalues of the system, the *Ackerman* function can be used [24]. Note that for this formula to be used, the system has to be *completely controllable*

$$\operatorname{rank}(\boldsymbol{A}) = \operatorname{rank}(\mathcal{C})\,.$$

[1] $\int_a^b f(t)\,\mathrm{d}t = \int_{a+0}^{b+0} f(t-0)\,\mathrm{d}t$

Adding a reference value

In real-world applications, it is useful to specify the value that the system has to track – *reference value* $\boldsymbol{w}_r$. Note that when adding a reference value, the problems that are solved are often called *regulator* problems.

To add a reference value into (5.10), two approaches can be used. One (less robust) is to add the reference value directly to the feedback control loop

$$\boldsymbol{u} = -\boldsymbol{k}\boldsymbol{x} + \boldsymbol{w}_r \,,$$

so that the state equation can be written as

$$\boldsymbol{x}' = (\boldsymbol{A} - \boldsymbol{E}\boldsymbol{k})\boldsymbol{x} + \boldsymbol{E}\boldsymbol{w}_r \,. \tag{5.20}$$

The schematic describing (5.20) is in Figure 5.7.

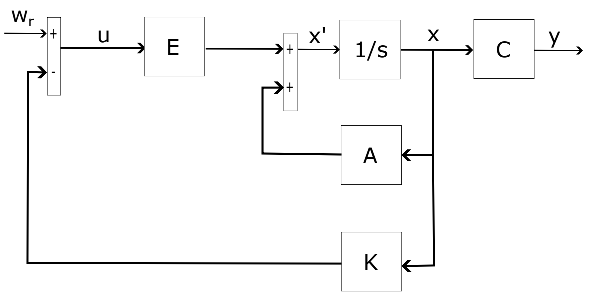


Figure 5.7: Closed-loop feedback with reference value.

The system described by (5.20) is going to have a big steady state error which is very difficult to control. To alleviate this, an additional gain $\boldsymbol{k}_r$ can be added. It can be calculated using the following process [24]. The output of the system $\boldsymbol{y} \to \boldsymbol{w}_r$, where $\boldsymbol{w}_r$ is the reference value. Therefore, the new state vector $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{u}}$ has to be selected, such that at $\boldsymbol{x} = \overline{\boldsymbol{x}}$, $\boldsymbol{u} = \overline{\boldsymbol{u}}$ and $\boldsymbol{x}' = \boldsymbol{0}$ and $\boldsymbol{y} = \boldsymbol{w}_r$. Substituting the values into (5.8)

$$\begin{aligned} \boldsymbol{x}' &= \boldsymbol{0} = \boldsymbol{A}\overline{\boldsymbol{x}} + \boldsymbol{E}\overline{\boldsymbol{u}} \\ \boldsymbol{y} &= \boldsymbol{w}_r = \boldsymbol{C}\overline{\boldsymbol{x}} \,. \end{aligned}$$

Writing these equations in state-space form for $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{u}}$

$$\begin{pmatrix} \overline{\boldsymbol{x}} \\ \overline{\boldsymbol{u}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{E} \\ \boldsymbol{C} & \boldsymbol{0} \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{1} \end{pmatrix} \boldsymbol{w}_r \,.$$

New gain $\boldsymbol{k}_r$ has to be calculated, so that $\boldsymbol{y} \to \boldsymbol{w}_r$ is a steady state. We can rewrite the feedback $\boldsymbol{u}$

$$\boldsymbol{u} = -\boldsymbol{k}\boldsymbol{x} + \boldsymbol{k}_r\boldsymbol{w}_r$$

and we can calculate the value of $\boldsymbol{k}_r = \overline{\boldsymbol{k}_r}$ from the steady state

$$\begin{aligned} &\boldsymbol{x}' = 0 = (\boldsymbol{A} - \boldsymbol{E}\boldsymbol{k})\overline{\boldsymbol{x}} + \boldsymbol{E}\boldsymbol{k}_r\boldsymbol{w}_r \\ &y = \boldsymbol{w}_r = C\overline{\boldsymbol{x}} = -\boldsymbol{C}(\boldsymbol{A} - \boldsymbol{E}\boldsymbol{k})^{-1}\boldsymbol{B}\boldsymbol{k}_r \qquad \to \qquad \overline{\boldsymbol{k}_r} = -\frac{1}{\boldsymbol{C}(\boldsymbol{A} - \boldsymbol{E}\boldsymbol{k})^{-1}\boldsymbol{E}} \,. \end{aligned}$$

Substituting $\boldsymbol{u}$ into back into state equation

$$\boldsymbol{x}' = (\boldsymbol{A} - \boldsymbol{E}\boldsymbol{k})\boldsymbol{x} + \boldsymbol{E}\boldsymbol{k}_r\boldsymbol{w}_r \,. \tag{5.21}$$

The schematic for (5.21) is in Figure 5.8.

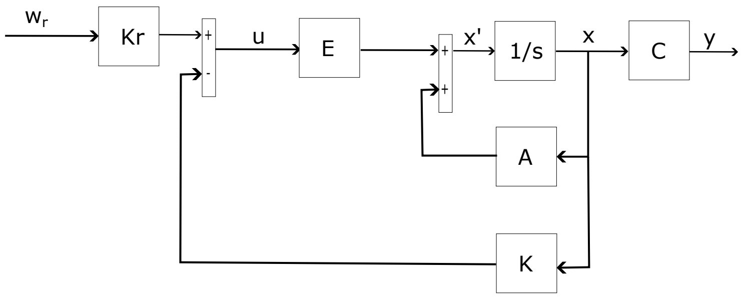


Figure 5.8: Closed-loop feedback with the reference value and compensation.

To improve the behaviour of the system even more during tracking, integral control can be added.

Adding an integral control

For more robust tracking, adding an *integral* might be useful. It can be achieved by adding a new state variable $\boldsymbol{z}$

$$\boldsymbol{z} = \boldsymbol{w}_r - \boldsymbol{y}\,,$$

where $\boldsymbol{w}_r$ is a reference value and $\boldsymbol{y}$ is the output of the system. The augmented state-space representation

$$\begin{pmatrix} \boldsymbol{x}' \\ \boldsymbol{z}' \end{pmatrix} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{0} \\ -\boldsymbol{C} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{pmatrix} + \begin{pmatrix} \boldsymbol{E} \\ \boldsymbol{0} \end{pmatrix} u + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{w}_r \end{pmatrix} .$$

A controller with an integral part again stabilizes the system by making $\boldsymbol{z}' \to \boldsymbol{0}$ and $\boldsymbol{x}' \to \boldsymbol{0}$. The control law u can be written as

$$u = \begin{pmatrix} \boldsymbol{k} & \boldsymbol{k}_i \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{pmatrix} + \boldsymbol{k}_r\boldsymbol{w}_r \,.$$

Substituting $\boldsymbol{u}$ into the augmented state-representation yields

$$\begin{pmatrix} \boldsymbol{x}' \\ \boldsymbol{z}' \end{pmatrix} = \begin{pmatrix} \boldsymbol{A} - \boldsymbol{E}\boldsymbol{k} & \boldsymbol{E}\boldsymbol{k}_i \\ -\boldsymbol{C} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{pmatrix} + \begin{pmatrix} \boldsymbol{E} \\ \boldsymbol{1} \end{pmatrix} \boldsymbol{w}_r \,. \tag{5.22}$$

The schematic of the state-space with integral control is in Figure 5.9.

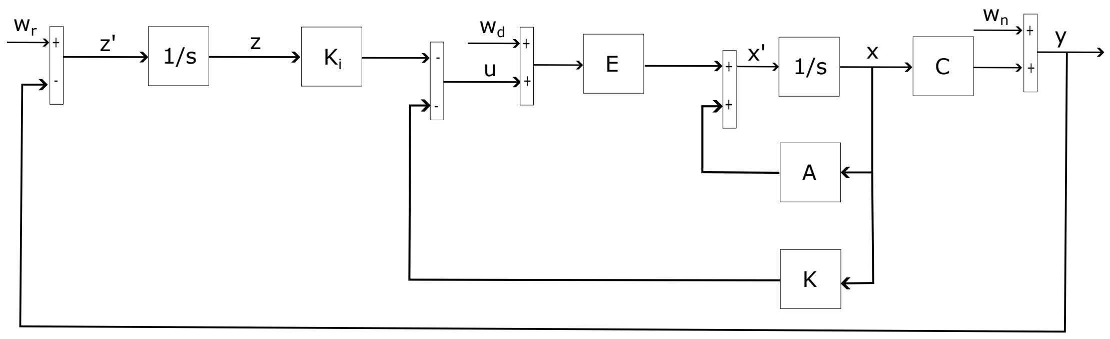


Figure 5.9: Closed-loop feedback with the integrator.

5.4.3 Linear-Quadratic-Regulator

Given the controllable system and either full-state measurements or an observable system with a full state-state estimate, there are many choices for the control law

$$u = -\boldsymbol{k}\boldsymbol{x}\,.$$

It is possible to make the eigenvalues of the closed-loop system $(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{k})$ arbitrarily stable, placing them as far as desired in the left-half of the complex plane (using the pole-placement method discussed previously). However, overly stable eigenvalues might be very expansive and might overwhelm the controller. The control system can also overreact to noise and disturbances.

Choosing the best gain vk to stabilize the system is the goal of *optimal control* [16]. The cost

$$J = \int_0^{\infty} \boldsymbol{x}^T(t)\boldsymbol{Q}\boldsymbol{x}(t) + \boldsymbol{R}u^2(t)\mathrm{d}t$$

balances the cost of effective regulation of the state with the cost of control. The matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ weight the cost of deviations of the state from zero and the actuation cost, respectively. This cost is minimized when

$$\boldsymbol{k} = \boldsymbol{R}^{-1}\boldsymbol{E}^T\boldsymbol{P}\,.$$

The matrix $\boldsymbol{P}$ can be determined by solving the algebraic *Riccati* equation

$$\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{A}\boldsymbol{P} - \boldsymbol{P}\boldsymbol{E}\boldsymbol{R}^{-1}\boldsymbol{b}^T\boldsymbol{P} + \boldsymbol{Q} = 0. \tag{5.23}$$

The schematic of the Linear Quadratic Regulator (LQR) connected to the system represented in the state-space representation is in Figure 5.10.

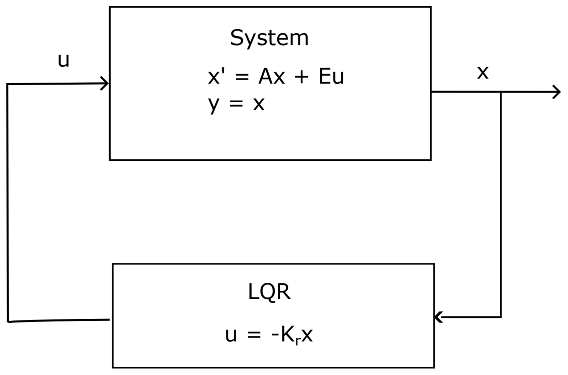


Figure 5.10: LQR controller connected in the closed-loop feedback [24].

5.4.4 Non-linear controllers

In this Subsection, several types of non-linear controllers will be introduced. This Subsection is compiled from [70] and [36].

Linearizing controllers

The central idea of this approach is to algebraically transform non-linear system dynamics into a (fully or partly) linear one so that the linear control techniques can be applied.

The idea of simplifying the dynamics of the system is not entirely unfamiliar. It is well known that the behaviour of the system depends considerably on a choice of reference frames or coordinate systems. The feedback linearization can be viewed as a *way to transform the original model of the system into the equivalent model of a simpler form.*

The feedback linearization can be simply applied to systems that can be described in so-called *controllability canonical form*:

$$\boldsymbol{x}' = f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u}\,,$$

where $\boldsymbol{u}$ is the control input, $\boldsymbol{x}$ is the state vector, $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are non-linear state functions. This form *does not have the derivative of the output.* To design the control input $\boldsymbol{u}$ for the single-input non-linear system of the form

$$\boldsymbol{x}' = f(\boldsymbol{x}, \boldsymbol{u})$$

the following two steps have to be performed. The state transformation $\boldsymbol{z} = \boldsymbol{z}(\boldsymbol{x})$ and the input transformation $\boldsymbol{u} = (\boldsymbol{x}, \boldsymbol{v})$ so that the non-linear system dynamics are transformed into an equivalent *linear time-invariant* dynamics in the state-space form (see Subsection 5.2.2):

$$\boldsymbol{z}' = \boldsymbol{A}\boldsymbol{z} + \boldsymbol{b}\boldsymbol{v}\,.$$

The second step is using standard techniques (pole placement for example). The schematic of the input-state linearization is in Figure 5.11.

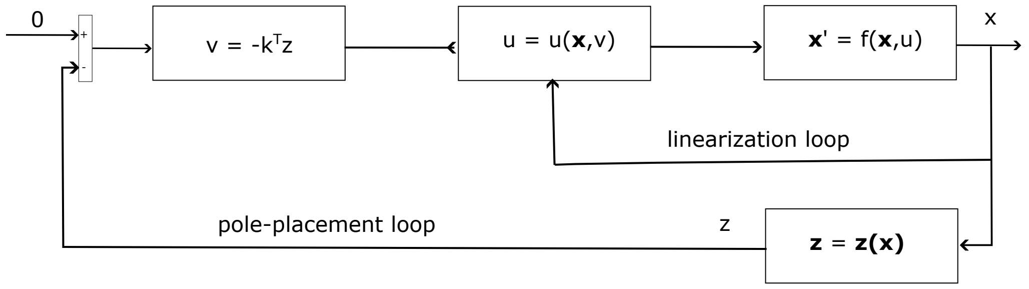


Figure 5.11: Input state linearization.

Detailed information about the construction of linearizing controllers can be found in [70].

Lyapunov based controller

The Lyapunov function (defined in Subsection 5.3.3) can be used as a control law (as a stabilizing controller) for the system. The biggest drawback of this approach is the great difficulty in obtaining the Lyapunov function for the system. The process of obtaining a candidate function and verifying its properties is explained in more detail in [70] and used practically in Chapter 6.

5.5 Control system implementation using MTSM

The main goal of this thesis (as stated previously) is to determine if the method introduced in Chapter 3 can be used in control systems with a particular focus on real-time systems (strict requirements for time of calculation, accuracy etc. [48]) in the simulation model of the real plant. The experiments in Chapter 4 indicate that the method can be used and behaves correctly is stable, and in most cases, faster and more accurate than commonly used methods in this context.

Because real hardware implementation of the method would require truly specialized hardware with extremely precise AD/DA conversion (for example, most common Arduino controllers have just 10-bit converters that would degrade the accuracy of the calculation to the point where the proposed method would lose usefulness) the goal of the thesis has shifted to approximate the real-system (so that both „real" and „simulated" plants are independently modelled in software) with many real-world issues (quantization, delay, . . .) being modelled as well. Because the thesis mainly considers how the method handles the simulation of the simulated plant, this seems to be an acceptable solution.

The method could be effectively used to model the behaviour of the real-system and use the obtained data to control the system (in the model-following control configuration), as shown in Figure 5.12.

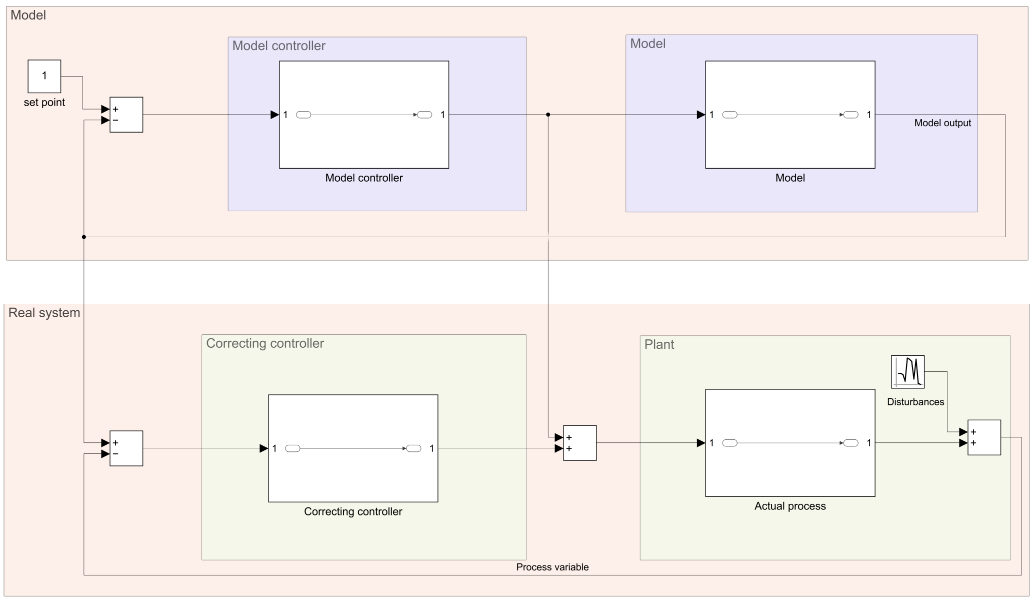


Figure 5.12: The general schematic of the system configuration used for testing.

Note that the experiments performed in this thesis consider that the method would operate in the *Model* part of the system, and that is simulated in the experiments performed in Chapter 6. The real system would, in operating conditions, be controlled by the simulated outputs of the method. The positive properties established in Chapter 3 can be used in most cases to increase the accuracy of the model, decrease the time of the calculation and make the control system more robust.

Chapter 6

Experiments

In this Chapter, the performed experiments using MTSM in the area of system control with a focus on real-time characteristics and behaviours are discussed in detail. Every experiment is first introduced and then the ODEs that describe the behaviour of the selected problem are defined. The solution obtained using the linear solver is compared to the solution of the original non-linearized system. The solution is then repeated using the state-of-the-art ODE solvers in MATLAB to show better performance and stability of the MTSM. Several examples are also fully non-linear using the basic approaches from non-linear control.

All experiments were performed in MATLAB 2021a on Ryzen 5 3600XT CPU with six cores equipped with 32 GB of RAM on Windows 11. Experiments were performed 100 times. Values in the column labelled *Times of calculation* or *Time* are taken as a *mean* of calculation times. Solver $MTSM_{opt}$ (see Subsection 3.3.2) is used for the solution of non-linear problems with the Modern Taylor Series Method.

6.1 Direct current motor

The DC (direct current) motor is a very interesting problem because it is widely used in real-world systems. Further, it combines mechanical and electrical parts, so the modelling and control of such a system can be challenging. The analysis of the control for the DC motor was first performed in [79], where the system was presented, and the first controller was implemented for it.

6.1.1 Mathematical description

The model is based on [10] and [74] and is in Figure 6.1.

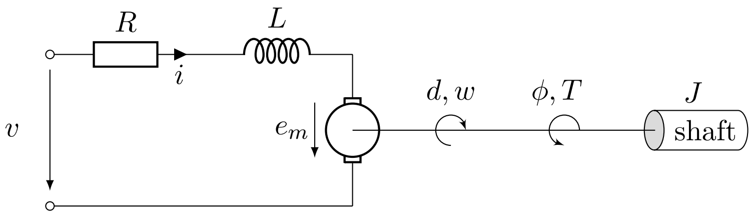


Figure 6.1: Scheme of the DC motor.

In Figure 6.1 v is the input voltage of the motor armature, R is the armature resistance, L is the armature inductance, e_m is the electromotive force, J is the motor inertia, ϕ is the angle of rotation of the shaft, T is the torque of the rotor, w is the disturbance torque and d is damping torque. For the mechanical part of the motor, the basic equation can be written as

$$J\phi'' = T - w - d\,. \tag{6.1}$$

In general, the torque T generated by a motor is proportional to the armature current i and the strength of the magnetic field. For the experiments in this thesis, the magnetic field is assumed to be *constant*. This means that the torque of the motor is proportional only to the armature current by a torque constant K_T

$$T = K_T i\,. \tag{6.2}$$

The damping torque d can be expressed as

$$d = b\phi'\,, \tag{6.3}$$

where b is the motor viscous friction constant [82]. Equations (6.2) and (6.3) can be substituted into (6.1)

$$J\phi'' = K_T i - w - b\phi'\,,$$

so the equation describing the mechanical part of the system can be written as

$$J\phi'' + b\phi' = K_T i - w\,. \tag{6.4}$$

The electrical part of the system (the armature) is described differently. From Figure 6.1, using Kirchhoff's circuit laws and Ohm's law, the following equation can be obtained

$$Li' + Ri = v - e_m\,. \tag{6.5}$$

The counter-electromotive force e_m (back-electromotive force, back EMF) is proportional to the angular velocity ϕ' of the shaft by a counter electromotive constant K_e

$$e_m = K_e\phi'\,. \tag{6.6}$$

Equation (6.6) can be substituted into (6.5)

$$Li' + Ri = v - K_e\phi'\,, \tag{6.7}$$

with the resulting ODE representing the behaviour of the armature. Equations (6.4) and (6.7) govern the dynamics of the DC motor, and for clarity, they can be written together:

$$\begin{aligned} J\phi'' + b\phi' &= K_T i - w \\ Li' + Ri &= v - K_e\phi'\,. \end{aligned} \tag{6.8}$$

Note that (6.8) is linear.

6.1.2 State-space representation

The state-space variables can be set as follows

$$
\begin{aligned}
x_1 &= \phi' \\
x_2 &= i
\end{aligned}
$$

the output of the system is set to x_1, the input voltage v is used as the control input $u = v$. Equation (6.8) can be rewritten

$$
\begin{aligned}
\phi'' &= x_1' = \frac{K_T}{J} i - \frac{1}{J} d - \frac{b}{J} \phi' = \frac{K_T}{J} x_2 - \frac{1}{J} d - \frac{b}{J} x_1 \qquad && x_1(0) = 0 \\
i' &= x_2' = \frac{1}{L} v - \frac{K_e}{L} x_1 - \frac{R}{L} x_2 && x_2(0) = 0
\end{aligned}
$$

with the following state-space matrix-vector representation (see Section 5.4.2)

$$
\boldsymbol{A} = \begin{pmatrix} -\frac{b}{J} & \frac{k_T}{J} \\ -\frac{K_e}{L} & -\frac{R}{L} \end{pmatrix}, \qquad \boldsymbol{b} = \boldsymbol{e} = \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix}, \qquad \boldsymbol{C} = \begin{pmatrix} 1 & 0 \end{pmatrix}, \qquad \boldsymbol{G} = \begin{pmatrix} -\frac{1}{J} \\ 0 \end{pmatrix}.
$$

For the definition of the matrices, see (5.10). For the numerical experiments, the following values are going to be used:

Variable	Meaning	Value
J	inertia of the motor	$0.01\,\mathrm{kg{\cdot}m^2}$
b	viscous friction constant of the motor	$0.1\,\mathrm{N{\cdot}m{\cdot}s}$
K_e	counter electromotive constant	$0.01\,\mathrm{V{\cdot}rad^{-1}{\cdot}s^{-1}}$
K_T	torque constant	$0.01\,\mathrm{N{\cdot}m{\cdot}A^{-1}}$
R	armature resistance	$1\,\Omega$
L	armature inductance	$0.5\,\mathrm{H}$

Table 6.1: Values for the following simulation experiments.

with the tolerances of all solvers set to be $TOL = 1 \times 10^{-9}$, maximum simulation time $t_{max} = 10\,\mathrm{s}$ and step-size $h = 0.1\,\mathrm{s}$. The behaviour of the system with constant input voltage $v = 1\,\mathrm{V}$ is in Figure 6.2, the plot of the ORD function is in Figure 6.3.

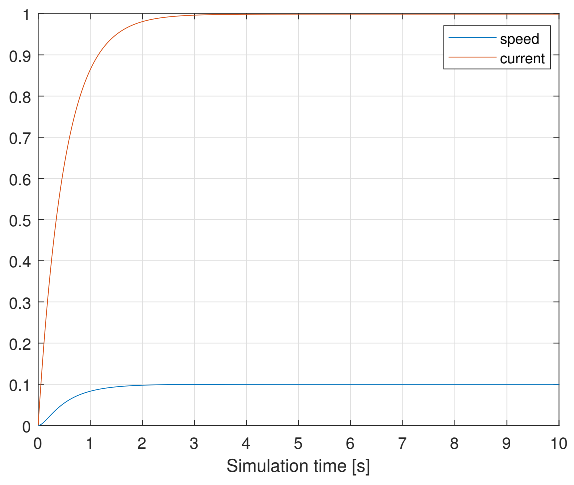


Figure 6.2: DC motor without control – reaction to the input voltage $v = 1\,\mathrm{V}$.

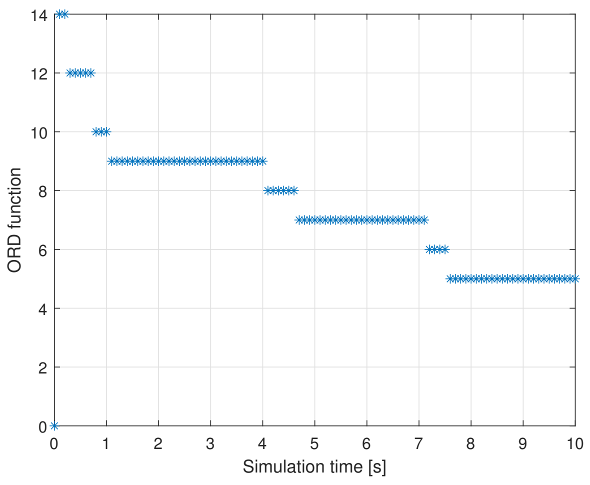


Figure 6.3: Plot of the *ORD* function for DC motor without control.

The results for the system without a controller are in Table 6.2.

Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	9.6335×10^{-4}	–	100
ode23	4.98945×10^{-2}	**51.79**	10486
ode45	2.8578×10^{-3}	**2.97**	1705
ode113	3.48815×10^{-3}	**3.62**	254

Table 6.2: Results of the simulation of the DC motor without a controller with a constant input voltage $v = 1\,\mathrm{V}$.

Table 6.2 shows that MTSM solves the problem faster than the selected state-of-the-art solvers. The next step is to add a controller into the system and determine how the method behaves.

6.1.3 Controllability, observability

To determine if the state-space controller can be implemented, the controllability of (6.9) has to be established. The constructed controllability matrix $\mathcal{C}$

$$\mathcal{C} = \begin{pmatrix} 0 & 2 \\ 2 & -4 \end{pmatrix}$$

has to have $\operatorname{rank}(\mathcal{C}) = 2$ equal to the number of rows of the matrix $\boldsymbol{A}$. This is true in this case, which means that the system is fully controllable. The observability of (6.9) can also be verified, note that the observer based controller is not going to be used. The observability matrix $\mathcal{O}$

$$\mathcal{O} = \begin{pmatrix} 1 & 0 \\ -10 & 1 \end{pmatrix}$$

has to have $\operatorname{rank}(\mathcal{O}) = 2$ equal to the number of rows of the matrix $\boldsymbol{A}$ for the system to be fully observable. Because the system is controllable, the controller can be constructed. The observer can potentially be added because the system is observable.

6.1.4 Experiments

The feedback controller is going to be constructed according to Subsection 5.4.2, with a reference speed that the motor has to maintain set to $\omega_r = 0.5\,\mathrm{rad{\cdot}s^{-1}}$. For the selected poles of the system

$$\begin{aligned} p_1 &= 5 - 1i \\ p_2 &= 5 + 1i \end{aligned} \tag{6.9}$$

with parameters $MTSM_h = 0.2\,\mathrm{s}$, $\boldsymbol{k} = (12.99, -1)$ and $k_r = 13$. The system stabilizes at the reference value at approximately $t \approx 1\,\mathrm{s}$, $t_{max} = 5\,\mathrm{s}$ which can be seen in Figure 6.4, the plot of the ORD function is in Figure 6.5.

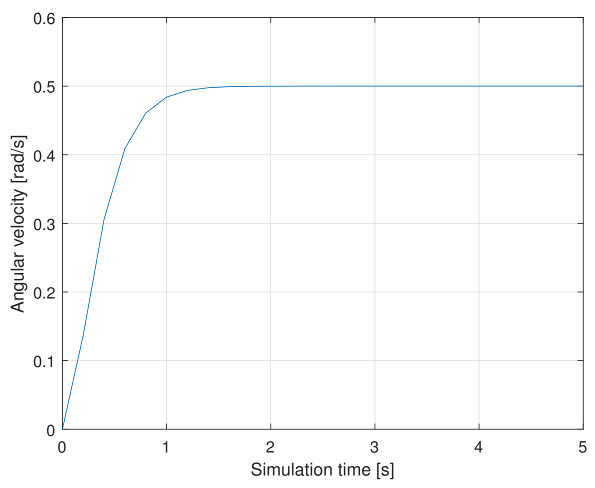


Figure 6.4: DC motor with a state-space feedback controller.

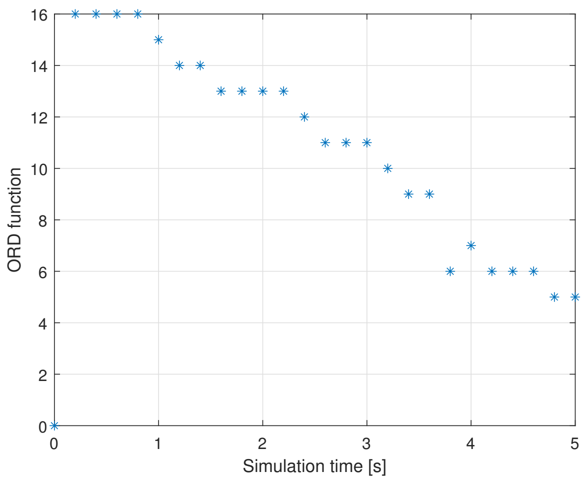


Figure 6.5: Plot of the *ORD* function for DC motor with state-space feedback controller.

The numerical results are in Table 6.3.

Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	3.101×10^{-4}	–	25
ode23	$5.925\,04 \times 10^{-2}$	**191.07**	11058
ode45	3.2057×10^{-3}	**10.34**	1717
ode113	2.6264×10^{-3}	**8.47**	183

Table 6.3: Results of the simulation of the DC motor with state-space full feedback controller.

Additionally, the integrator can be added to state feedback to handle a potential disturbance. The integral control can be added using $k_i = 15$. The disturbance is added at $t = 4.7\,\text{s}$, $t_{max} = 10\,\text{s}$. Using the representation defined in Section 5.4.2, the behaviour of the system is in Figure 6.6 with the plot of the *ORD* function in Figure 6.7.

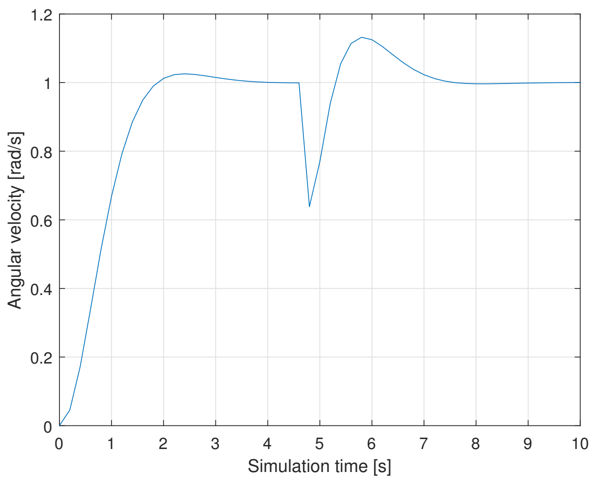


Figure 6.6: DC motor with state-space feedback controller with added integrator and disturbance.

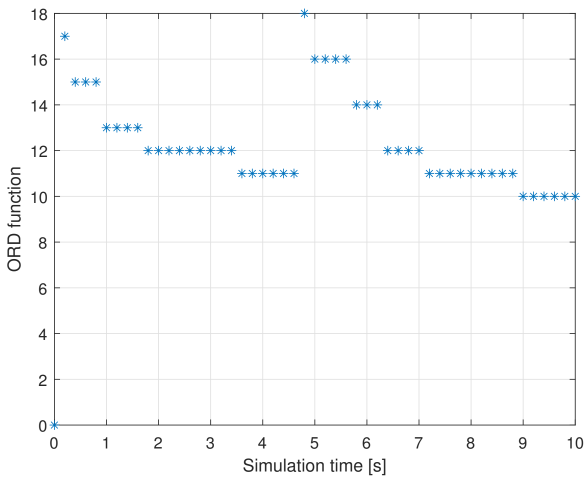


Figure 6.7: Plot of the *ORD* function for for DC motor with added integrator and disturbance.

The numerical results are in Table 6.4.

Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	6.5875×10^{-4}	–	50
ode23	8.1891×10^{-2}	**124.31**	16248
ode45	$3.972\,75 \times 10^{-3}$	**6.03**	2309
ode113	$3.473\,85 \times 10^{-3}$	**5.27**	248

Table 6.4: Results of the simulation of the DC motor with state-space full feedback controller with integrator and disturbance.

The presented controllers stabilize the system and track the reference value. However, the behaviour can be improved further by using a more optimal value of $\boldsymbol{k}$.

Linear Quadratic Regulator

In the second experiment with a DC motor, the LQR controller is going to be used to design the more optimal values of $\boldsymbol{k}$. The parameters were selected as

$$\boldsymbol{Q} = \begin{pmatrix} 15 & 0 \\ 0 & 2 \end{pmatrix}, \qquad R = 1\,, \qquad k_i = 15$$

with $\boldsymbol{k} = (0.1097, 0.7634)$. The behaviour of the system with the LQR controller is in Figure 6.8, and the plot of the ORD function is in Figure 6.9.

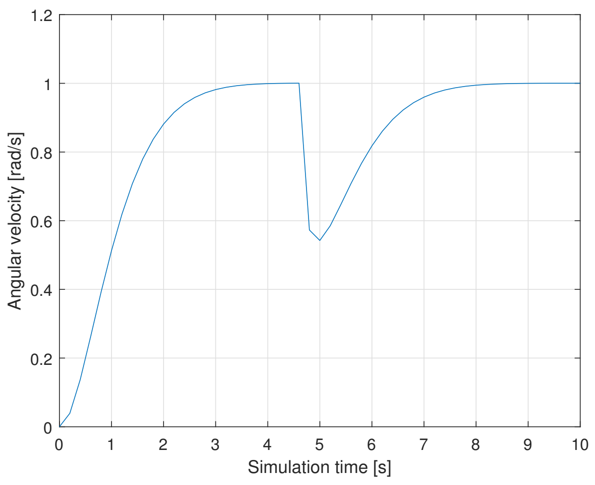


Figure 6.8: DC motor with LQR controller.

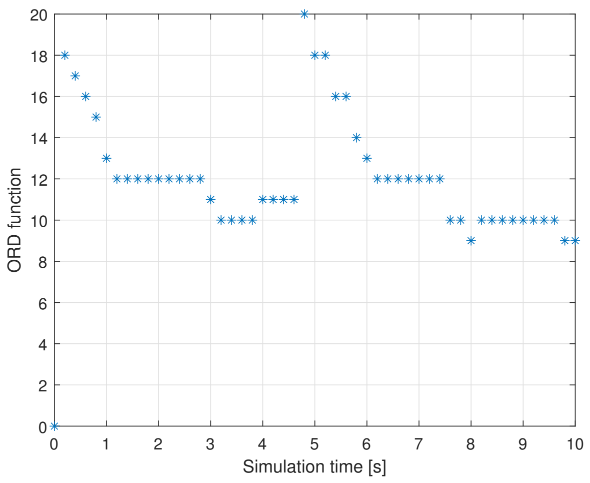


Figure 6.9: Plot of the ORD function for DC motor with LQR controller.

The numerical results are in Table 6.5.

Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	6.6215×10^{-4}	–	50
ode23	7.5141×10^{-2}	**113.48**	14517
ode45	$3.622\,35 \times 10^{-3}$	**5.47**	2101
ode113	3.9806×10^{-3}	**6.01**	284

Table 6.5: Results of the simulation of the DC motor with Linear Quadratic Regulator.

Behaviour for the different values of ω_r

The final experiment with the DC motor shows the behaviour of (6.9) for different values of the reference value ω_r, namely $\omega_r \in (1, 10, 100)$ rad·s^{-1}. The full-state feedback controller is going to be used. The results are summarized in Table 6.6.

	$\omega_r = 1\,\text{rad·s}^{-1}$			$\omega_r = 10\,\text{rad·s}^{-1}$			$\omega_r = 100\,\text{rad·s}^{-1}$		
Solver	Time [s]	Ratio	# steps	Time [s]	Ratio	# steps	Time [s]	Ratio	# steps
MTSM	3.5715×10^{-4}	–	25	3.751×10^{-4}	–	25	3.8625×10^{-4}	–	25
ode23	$1.058\,99 \times 10^{-1}$	**196.11**	12592	$9.371\,54 \times 10^{-2}$	**249.8**	16321	$1.598\,52 \times 10^{-1}$	**274.17**	18948
ode45	3.746×10^{-3}	**10.49**	1873	$4.038\,25 \times 10^{-3}$	**10.77**	2085	$4.353\,15 \times 10^{-3}$	**11.27**	2181
ode113	2.8527×10^{-3}	**7.99**	185	3.1127×10^{-3}	**8.3**	199	$3.117\,95 \times 10^{-3}$	**8.07**	201

Table 6.6: Time of the calculation for different values of ω_r.

The plot of the *ORD* function for the selected reference values are in Figure 6.10.

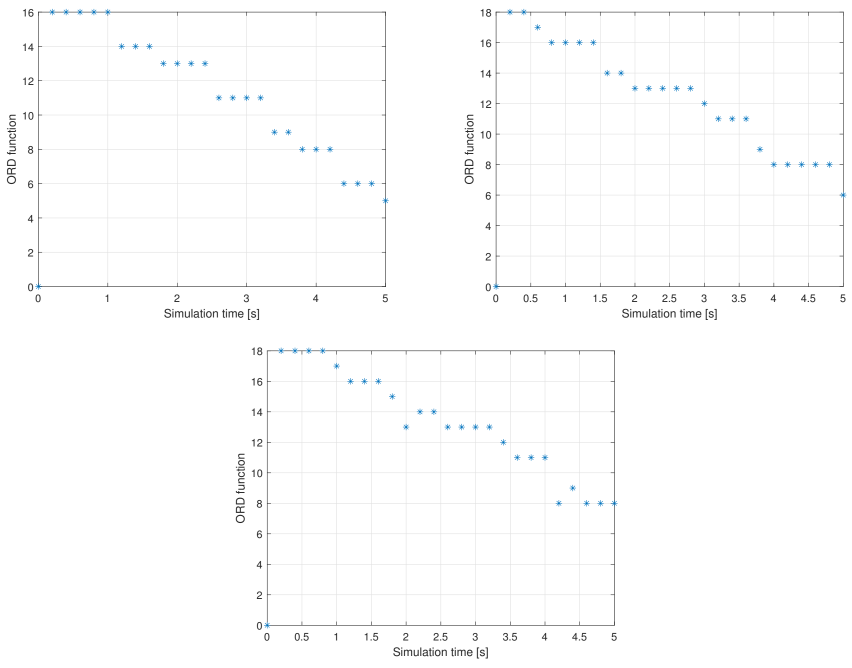


Figure 6.10: Plot of the *ORD* function for $\omega_r = 1\,\text{rad·s}^{-1}$ (left), $\omega_r = 10\,\text{rad·s}^{-1}$ (right), $\omega_r = 100\,\text{rad·s}^{-1}$ (bottom).

Figure 6.10 shows that the values of the ORD function are the highest at the beginning of the calculation and then the function stabilizes. The higher values of ω_r do not influence the ORD function much.

6.1.5 Concluding remarks

The analysis of the DC motor shows the potential of the method in control systems. The calculation of state equations using the method is several times faster than state-of-the-art methods with different controllers and reference values. This experiment worked with the *linear* variant of MTSM. The following experiments non-linear experiments are going to be more challenging.

6.2 Pendulum

The pendulum problem is one of the most interesting problems in physics. The pendulum has the same behaviour as an ideal oscillator (if damping is not present) and can exhibit chaotic behaviour. For the purposes of this thesis, the problem is non-linear, which is useful for testing the non-linear MTSM solver in the control system domain.

The control and analysis of the system with a simple pendulum was presented in [77]. This Section uses material from [5], [61] and [30].

6.2.1 Mathematical description

Let us briefly talk about the system and its physical parameters. The simple pendulum is an idealization of the real pendulum. It consists of a point mass m, attached to the infinitely light and rigid rod of length l, attached to the frictionless pivot point. The system is in Figure 6.11.

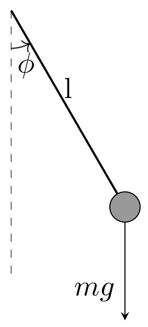


Figure 6.11: Stable simple pendulum without damping.

If the pendulum is displaced from its initial position, the idealized pendulum is going to oscillate forever with the constant amplitude (see Figure 6.12). Without damping, the system serves as an ideal oscillator.

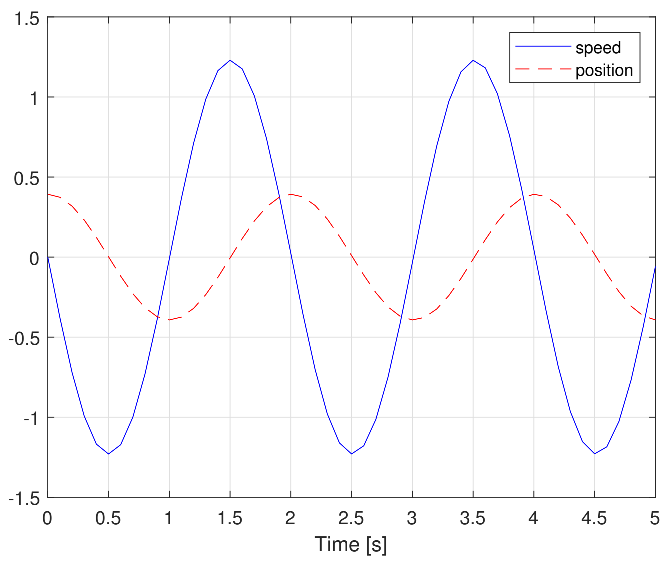


Figure 6.12: Speed and position of a pendulum without any damping (initial position $\frac{\pi}{8}$ rad).

To derive the equation of motion for the simple pendulum, we can use Newton's second law

$$F = ma\,,$$

where F is the force caused by the acceleration a on the mass m. The equation of motion for a simple pendulum without damping is then

$$ml\phi'' = -mg\sin\phi, \tag{6.10}$$

where ϕ is an angular displacement of the pendulum from the vertical position and g is the acceleration due to gravity. To make this problem more realistic, damping can be added to the pendulum. The updated force diagram is in Figure 6.13.

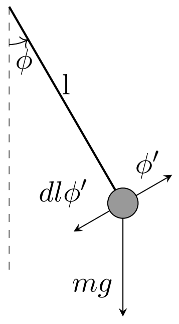


Figure 6.13: Stable simple pendulum with damping.

With the added damping, the pendulum loses speed over time (as seen in Figure 6.14).

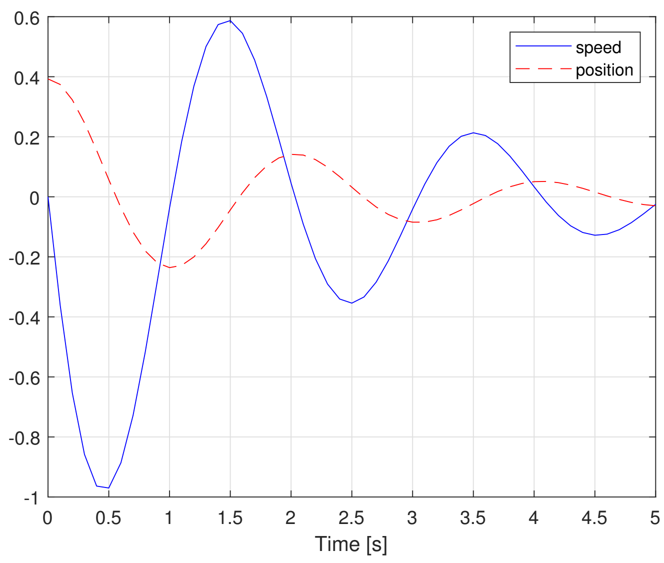


Figure 6.14: Speed and position of a pendulum with damping (initial position $\frac{\pi}{8}$).

By adding an additional damping term, the equation (6.10) becomes

$$ml\phi'' = -dl\phi' - mg\sin(\phi)\,, \tag{6.11}$$

where the term $dl\phi'$ represents damping. The term representing control for the pendulum u can be added

$$ml\phi'' = -dl\phi' - mg\sin(\phi) - u\,. \tag{6.12}$$

The equation (6.12) can be simplified

$$\phi'' = -\frac{d}{m}\phi' - \frac{g}{l}\sin(\phi) - \frac{u}{ml}\,. \tag{6.13}$$

The resulting equation represents the angular displacement of the pendulum with damping and possible control. Due to the fact that the numerical methods require first-order ODEs and (6.13) does not contain a derivative of a forcing function, the Method of Derivative Order Reduction (see Subsection 2.6.1) can be used. The resulting system

$$\begin{aligned}\omega' &= -\frac{d}{m}\omega - \frac{g}{l}\sin(\phi) - \frac{u}{ml}\\ \phi' &= \omega\,,\end{aligned} \tag{6.14}$$

where ω represents the angular speed of the pendulum and ϕ represents the position of the pendulum. To use MTSM with this system of ODEs, the term $\sin(\phi)$ has to be replaced with the generating equations (3.4)

$$\begin{aligned} q &= \sin(\phi) & r &= \cos(\phi)\\ q' &= \cos(\phi)\omega = r\omega & r' &= -\sin(\phi)\omega = -q\omega \end{aligned}$$

with initial conditions $q(0) = \sin(\phi_0)$ and $r(0) = \cos(\phi_0)$. The generated auxiliary ODEs can be added to the (6.14) so that the final representation of the problem becomes

$$\begin{aligned}
\omega' &= -\frac{d}{m}\omega - \frac{g}{l}q - \frac{1}{ml}u \quad & \omega(0) &= \omega_0 \\
\phi' &= \omega & \phi(0) &= \phi_0 \\
q' &= r\omega & q(0) &= \sin(\phi_0) \\
r' &= -q\omega & r(0) &= \cos(\phi_0)\,.
\end{aligned} \tag{6.15}$$

The parameters of the experiment are summarized in Table 6.7.

Variable	Meaning	Value
d	damping	1
m	mass of the point mass	1.5 kg
g	gravitational acceleration	$9.81\,\mathrm{m{\cdot}s^{-2}}$
l	length of the rod	1.5 m

Table 6.7: Values for the pendulum simulation experiments.

For $u = 0$ (system without control), the matrix-vector representation follows

$$\boldsymbol{y} = \begin{pmatrix} \omega \\ \phi \\ \sin(\phi) \\ \cos(\phi) \end{pmatrix} \quad \boldsymbol{A} = \begin{pmatrix} -\frac{d}{m} & 0 & -\frac{g}{l} & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \boldsymbol{B_1} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \boldsymbol{y}_{jk} = \begin{pmatrix} 4 & 1 \\ 3 & 1 \end{pmatrix}. \tag{6.16}$$

The system can be experimented with to determine if the non-linear implementation of the method handles this problem at all. For initial position $\phi_0 = \frac{\pi}{2}$ rad, initial speed $\omega_0 = 0\,\mathrm{rad{\cdot}s^{-1}}$, step size $MTSM_h = 0.1$ s, set tolerance $TOL = 1 \times 10^{-9}$ (with the state-of-the-art solvers achieving the similar precision) and $t_{max} = 5$ s, the numerical results (as a mean from 100 runs) are in Table 6.8.

Solver	Time of calculation [s]	Ratio
MTSM	$1.293\,32 \times 10^{-3}$	–
ode23	$1.952\,18 \times 10^{-2}$	**15.1**
ode45	$1.648\,12 \times 10^{-3}$	**1.27**
ode113	$2.852\,88 \times 10^{-3}$	**2.21**

Table 6.8: Results for the non-linear stable pendulum without control system.

Table 6.8 shows that the method performs the computation faster than the state-of-the-art methods. The error in the last step for the method was approximately 1×10^{-6}. The function describing the order of the method is in Figure 6.15. It shows that the function fluctuates (as is typical for non-linear problems).

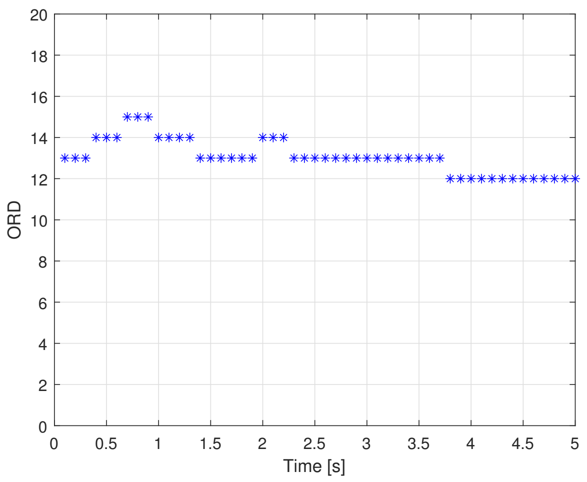


Figure 6.15: The *ORD* function for the non-linear pendulum.

Linearization

To linearize (6.14), the procedure established in Subsection 5.2.4 can be used. The equilibrium points can be calculated using (5.17) (for $\phi = 0\,\text{rad}$ or $\phi = \pi\,\text{rad}$)

$$\boldsymbol{x}^* = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \boldsymbol{x}^{**} = \begin{pmatrix} 0 \\ \pi \end{pmatrix}. \tag{6.17}$$

The partial derivatives of $\omega = x_1$ and $\phi = x_2$ from (6.14) can be calculated as follows:

$$\begin{aligned} \frac{\partial x_1}{\partial x_1} &= -\frac{d}{m} & \frac{\partial x_1}{\partial x_2} &= -\frac{g}{l}\cos(\phi) \\ \frac{\partial x_2}{\partial x_1} &= 1 & \frac{\partial x_2}{\partial x_2} &= 0 \\ \frac{\partial x_1}{\partial u_1} &= -\frac{u}{ml} & \frac{\partial x_2}{\partial u_2} &= 0\,. \end{aligned}$$

Linearizing around $\boldsymbol{x}^*$, the linearized system can be written as

$$\begin{aligned} \omega' &= -\frac{d}{m}\omega - \frac{g}{l}\phi \\ \phi' &= \omega \end{aligned} \tag{6.18}$$

and in the matrix-vector notation

$$\boldsymbol{x}' = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}u = \begin{pmatrix} -\frac{d}{m} & -\frac{g}{l} \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \omega \\ \phi \end{pmatrix} + \begin{pmatrix} -\frac{1}{ml} \\ 0 \end{pmatrix} u\,, \tag{6.19}$$

with initial conditions ω_0 as the initial velocity and ϕ_0 as the initial position of the pendulum being close to the $\boldsymbol{x}^*$. The behaviour of the system for initial speed $\omega_0 = 0\,\text{rad}\cdot\text{s}^{-1}$ and initial position $\phi_0 = \frac{\pi}{7}\,\text{rad}$, tolerances for all solvers were set to $TOL = 1 \times 10^{-9}$ and $t_{max} = 5\,\text{s}$. Note that for $\phi_0 = \frac{\pi}{2}\,\text{rad}$ the difference between linear and non-linear solution is approximately $3.362\,41 \times 10^{-1}$. The numerical results are in Table 6.9.

Solver	Time of calculation [s]	Ratio
MTSM	1.9693×10^{-3}	–
ode23	$2.588\,73 \times 10^{-2}$	**13.15**
ode45	7.0919×10^{-3}	**3.6**
ode113	5.9395×10^{-3}	**3.02**

Table 6.9: Results for the stable linear pendulum without a controller.

Table 6.9 shows that MTSM can solve the problem faster than the state-of-the-art methods selected for comparison. The plot of the ORD function is in Figure 6.16.

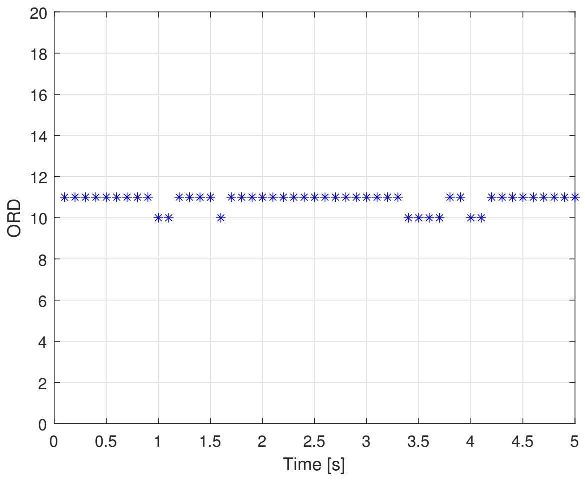


Figure 6.16: The ORD function for the linear pendulum.

The ORD function does not oscillate as much as with the non-linear solution (Figure 6.15) and is nearly constant during computation. Note that this problem is going to be relatively easy from a control standpoint because the pendulum is not upright.

6.2.2 Experiments with linear model

The experiments using the linear model obtained by linearizing the original non-linear equation are going to be discussed first.

PID controller

The PID controller can be used to control the linearized model of the pendulum (6.19). The model uses the structure for the PID controller established in Subsection 5.4.1. The matrix-vector representation is augmented

$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & -k_i \\ 1 & -k-k_d & -\frac{g}{l}-k_p \\ 0 & 1 & 0 \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} \omega_r k_i \\ \omega_r k_p \\ 0 \end{pmatrix}, \boldsymbol{y}_0 = \begin{pmatrix} 0 \\ 0 \\ \frac{\pi}{12} \end{pmatrix}.$$

The parameters of calculation are set to $t_{max} = 20\,\mathrm{s}$, $MTSM_h = 0.1\,\mathrm{s}$, and tolerances for all solvers $TOL = 1 \times 10^{-9}$. Parameters of the controller were set to $\omega_r = \frac{\pi}{8}$, $k_p = 1$, $k_i = 5$ and $k_d = 3$. The numerical results for this experiment are in Table 6.10.

Solver	Time of calculation [s]	Ratio
MTSM	$6.298\,16 \times 10^{-4}$	–
ode23	$1.437\,76 \times 10^{-2}$	**22.83**
ode45	$1.453\,79 \times 10^{-3}$	**2.31**
ode113	$2.245\,91 \times 10^{-3}$	**3.57**

Table 6.10: Results for the stable linear pendulum with PID controller.

The plot for the position of the pendulum is in Figure 6.17 with the plot of the *ORD* function in Figure 6.18.

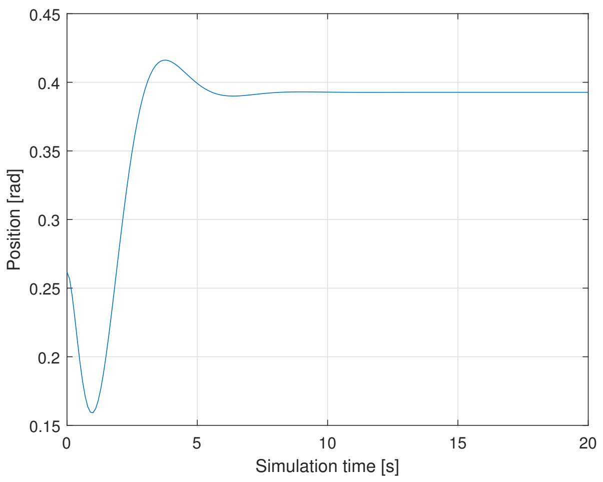


Figure 6.17: Plot of position ϕ for the pendulum with PID controller.

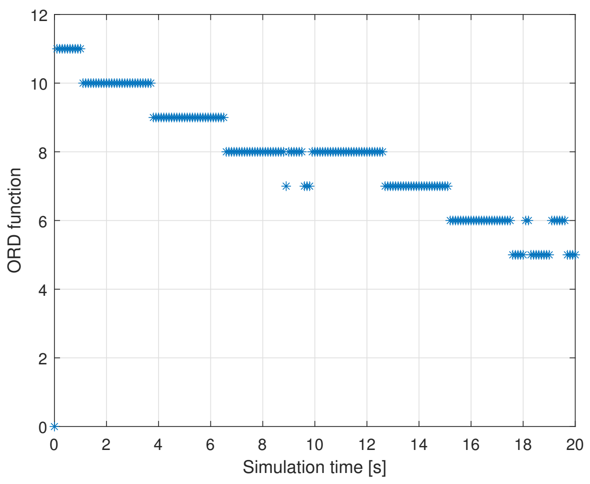


Figure 6.18: Plot of *ORD* function for the pendulum with PID controller.

Full-state feedback controller

The full-state feedback controller using state-space modelling can also be constructed using (6.14) for example. As stated above, for small angles, $\sin(\phi) \approx \phi$, so that

$$\begin{aligned} \omega' &= -\frac{d}{m}\omega - \frac{g}{l}\phi - \frac{1}{lm}u \\ \phi' &= \omega\,. \end{aligned} \tag{6.20}$$

The state variables

$$\begin{aligned} x_1' &= \omega' \\ x_2' &= \phi' \end{aligned}$$

represent the angular speed and angular position of the pendulum. Rewriting (6.20) using the state variables

$$\begin{aligned} x_1' &= -\frac{d}{m}x_1 - \frac{g}{l}x_2 - \frac{1}{lm}u \\ x_2' &= x_1\,. \end{aligned} \tag{6.21}$$

Using (6.21), the matrix-vector representation of the state equations can be expressed as

$$\boldsymbol{A} = \begin{pmatrix} -\frac{d}{m} & -\frac{g}{l} \\ 1 & 0 \end{pmatrix} \qquad \boldsymbol{x}_0 = \begin{pmatrix} x_1(0) = \omega_0 \\ x_2(0) = \phi_0 \end{pmatrix}, \qquad \boldsymbol{b} = \boldsymbol{e} = \begin{pmatrix} -\frac{1}{lm} \\ 0 \end{pmatrix}. \tag{6.22}$$

To determine if the state-space controller can be implemented, the controllability and observability of the (6.22) has to be established. The controllability matrix $\mathcal{C}$

$$\mathcal{C} = \begin{pmatrix} 0.4444 & -0.2963 \\ 0 & 0.4444 \end{pmatrix}$$

with $\text{rank}(\mathcal{C}) = 2$ equal to the number of rows of the matrix $\boldsymbol{A}$, which means that the system is controllable. The observer is not going to be used, however, to determine the potential viability, the observability matrix $\mathcal{O}$

$$\mathcal{O} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

with $\text{rank}(\mathcal{C}) = 2$ equal to the number of rows of the matrix $\boldsymbol{A}$, which means that the system is observable. Because the system is controllable, the feedback controller can be constructed, and the observer might also be constructed, due to the fact, that the system is observable. Using (5.20) for $\phi_r = 0.1\,\text{rad}$, $\boldsymbol{x}_0 = (2, 0)^T$ and $\boldsymbol{k} = (3, 4.7849)$ the following output is obtained.

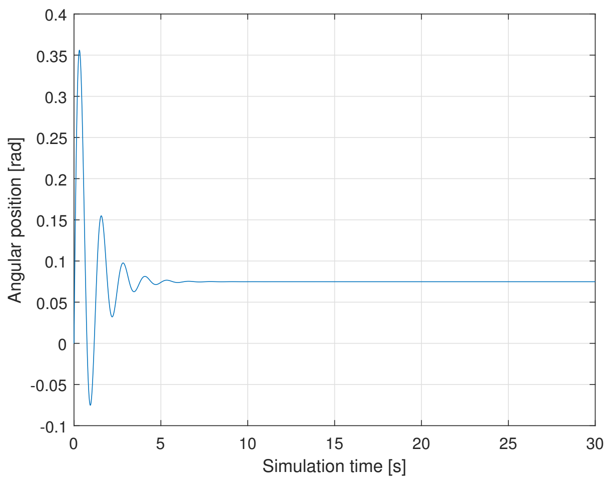


Figure 6.19: Steady state error.

The steady-state error is obvious, so (5.22) can be used to add an integrator. For $k_i = 7$ and $k_r = 29.06999$ the behaviour of the system is in Figure 6.20.

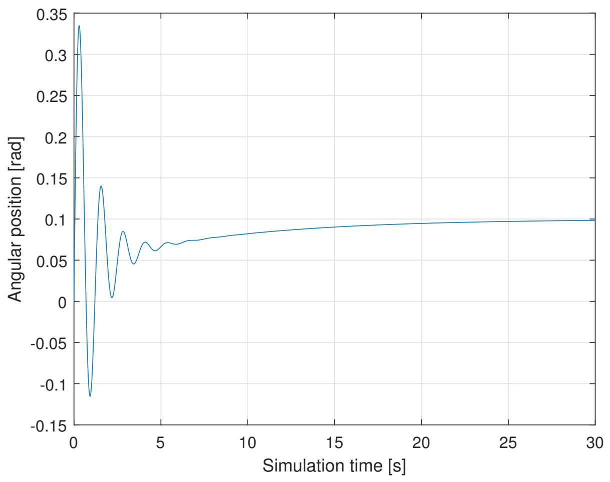


Figure 6.20: Plot of angular position ϕ of the pendulum for $\phi_r = 0.1\,\text{rad}$.

This controller eliminates the steady state error. The numerical results are in Table 6.11, and the plot of the *ORD* function is in Figure 6.21.

Solver	Time of calculation [s]	Ratio
MTSM	2.9739×10^{-3}	–
ode23	$1.471\,02 \times 10^{-1}$	**49.46**
ode45	$9.281\,45 \times 10^{-3}$	**3.12**
ode113	$8.631\,35 \times 10^{-3}$	**2.9**

Table 6.11: Results for the stable linear pendulum with the state-space controller.

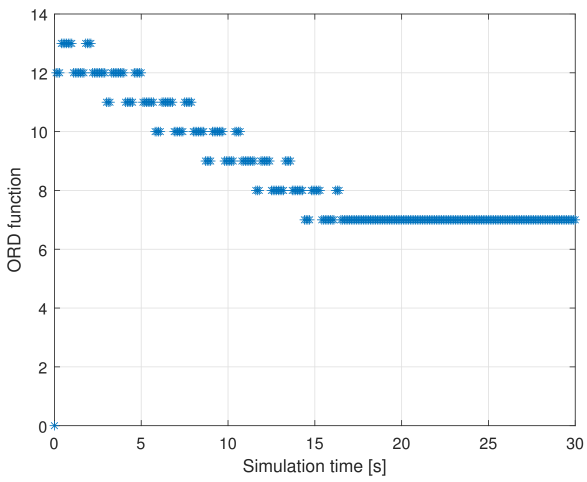


Figure 6.21: Plot of ORD function.

6.2.3 Experiments with non-linear model

Several experiments with non-linear model can also be performed.

Linearizing controller

When solving the pendulum using (6.16), the system can either be linearized before the calculation starts (so that (6.18) can be used) or a linearizing controller can be implemented that removes the non-linearity differently. The experiment with this type of controller was published in [80]. The problem is defined by augmenting the system of equations (6.15) by adding the forcing term u to the highest derivative. The modified system contains the added control term:

$$\begin{aligned} \omega' &= -\frac{d}{m}\omega - \frac{g}{l}a + u & \omega(0) &= \omega_0 \\ \phi' &= \omega & \phi(0) &= \phi_0 \\ a' &= b\omega & a(0) &= \sin(\phi_0) \\ b' &= -a\omega & b(0) &= \cos(\phi_0) \end{aligned} \tag{6.23}$$

where u is the added forcing term. When using a linearizing controller, the non-linear terms of the equation have to be removed. In this case, only non-linear term is $-\frac{g}{l}\sin(\phi)$. The equation of the controller can therefore be written as

$$u = \frac{g}{l}\sin(\phi) + v\,, \tag{6.24}$$

where v is the equation of the controller itself. For the experiment, PI controller was selected with the following equation

$$v = sp - y + 2(sp' - y') + sp''\,, \tag{6.25}$$

where sp is the chosen set-point and y is the selected output. For the experiment, $sp = \cos(t)$ (the pendulum should move according to the cosine function) and $y = \phi$ (the output of the controller is the position of the pendulum). Substituting the selected values into (6.25) gives the following equation

$$v = \cos(t) - \phi + 2(-\sin(t) - \omega) - \cos(t)\,, \tag{6.26}$$

and the full control law

$$u = \frac{g}{l}\sin(\phi) + \cos(t) - \phi + 2(-\sin(t) - \omega) - \cos(t)\,. \tag{6.27}$$

To use MTSM, the equations have to be transformed into a system of autonomous ODEs with just elementary mathematical operations. Equation (6.27) contains the $\sin(t)$ and $\cos(t)$ terms that have to be removed using another set of auxiliary ODEs

$$\begin{aligned}
c &= \sin(t) & & \\
c' &= \cos(t) = d & \qquad\qquad c(0) &= \sin(0) \\
d &= \cos(t) & & \\
d' &= -\sin(t) = -c & \qquad\qquad d(0) &= \cos(0)\,,
\end{aligned}$$

which is added to (6.23) so that the complete system becomes (6.28).

$$\begin{aligned}
\omega' &= -\frac{d}{m}\omega - \frac{g}{l}a + \frac{g}{l}a + d - \phi - 2c - 2\omega - d & \omega(0) &= \omega_0 \\
\phi' &= \omega & \phi(0) &= \phi_0 \\
a' &= b\omega & a(0) &= \sin(\phi_0) \\
b' &= -a\omega & b(0) &= \cos(\phi_0) \\
c' &= d & c(0) &= \sin(0) \\
d' &= -c & d(0) &= \cos(0)
\end{aligned} \tag{6.28}$$

The system (6.28) can be transformed into the matrix-vector notation:

$$\boldsymbol{y}_0 = \begin{pmatrix} \omega \\ \phi \\ \sin(\phi) \\ \cos(\phi) \\ \sin(t_0) \\ \cos(t_0) \end{pmatrix}, \quad \boldsymbol{A} = \begin{pmatrix} -\frac{d}{m} - 2 & 0 & 0 & -2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 \end{pmatrix}, \quad \boldsymbol{b} = \boldsymbol{e} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$
$$\boldsymbol{B}_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \boldsymbol{y}_{jk} = \begin{pmatrix} 4 & 1 \\ 3 & 1 \end{pmatrix}. \tag{6.29}$$

For $t_{max} = 20\,\mathrm{s}$, $MTSM_h = 0.1\,\mathrm{s}$, $d = 1$, $m = 1\,\mathrm{kg}$, $g = 10\,\mathrm{m{\cdot}s^{-2}}$, $L = 2\,\mathrm{m}$ with initial conditions $\omega_0 = 0\,\mathrm{rad{\cdot}s^{-1}}$ and $\phi_0 = \frac{\pi}{2}\,\mathrm{rad}$ the results are in the Table 6.12.

Solver	Time of calculation [s]	Ratio
MTSM	$1.321\,506 \times 10^{-2}$	–
ode23	$3.779\,570\,9 \times 10^{-1}$	**28.6**
ode45	$2.819\,609 \times 10^{-2}$	**2.1**
ode113	$1.349\,621 \times 10^{-2}$	**1.1**

Table 6.12: Results for the linearizing controller with $MTSM_h = 0.1\,\mathrm{s}$.

The behaviour of the pendulum is in Figure 6.22. The pendulum follows the function set as the set point (reference value).

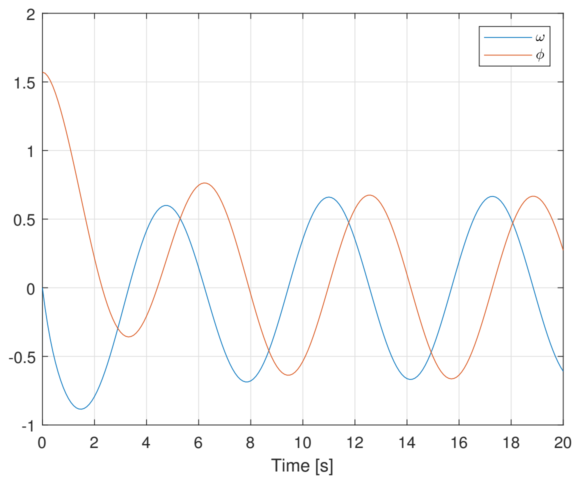


Figure 6.22: Plot of the speed (ω) and the position (ϕ) of the pendulum controlled by the linearizing controller.

The plot of the *ORD* function is in Figure 6.23. Due to the fact that the introduced control u removes non-linear terms during the calculation, the system exhibits linear behaviour.

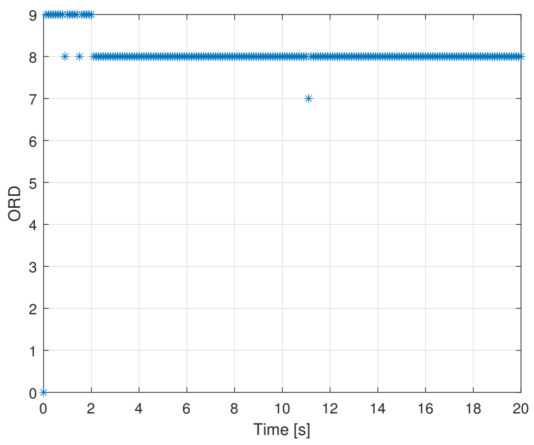


Figure 6.23: *ORD* function for the linearizing controller with $MTSM_h = 0.1\,\mathrm{s}$.

If the sampling rate was set higher (for example, the information about the current state of the pendulum would be obtained twice per second, i.e. $h = 0.5\,\mathrm{s}$), the results would be even more favourable, as shown in Table 6.13.

Solver	Time of calculation [s]	Ratio
MTSM	$5.536\,432 \times 10^{-3}$	–
ode23	$5.113\,741\,86 \times 10^{-1}$	**92.4**
ode45	$2.638\,108 \times 10^{-2}$	**4.8**
ode113	$9.467\,839 \times 10^{-3}$	**1.71**

Table 6.13: Results for the linearizing controller with $h = 0.5\,\mathrm{s}$.

The accuracy and the stability of MTSM are higher than state-of-the-art methods, which would mean more headroom during the integration step.

Lyapunov-based control system

As the final example, consider the pendulum system based on the Lyapunov function, described in Section 5.3. This example is going to show the usefulness of non-linear models. The controller is going to stabilize the pendulum with a range of initial conditions

$$\omega_0 \in (-10, 10)$$
$$\phi_0 \in (-10, 10)\,.$$

Without a controller, the pendulum stabilizes into its equilibrium as shown in Figure 6.24.

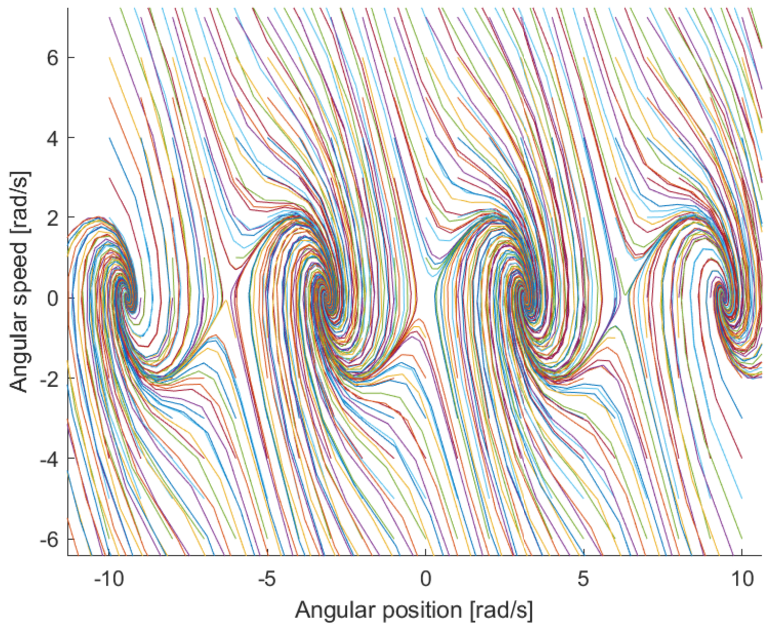


Figure 6.24: Pendulums with a range of initial speeds and positions without a controller.

The candidate function that can be used to stabilize the pendulum can be selected as, for example

$$u = \frac{-gx_2 - g\sin(x_2)}{l}\,.$$

The function is visualized in Figure 6.25. We can check if this function satisfies the conditions for the Lyapunov function. It has to be positive everywhere except the equilibrium point of the pendulum.

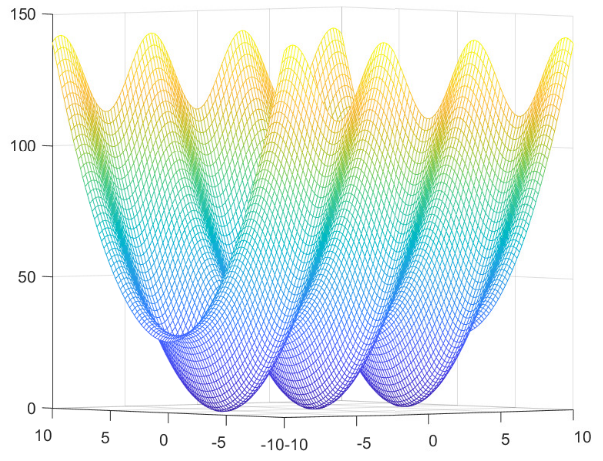


Figure 6.25: Plot of the candidate function.

The derivative of the function has to be negative everywhere for the system to be stable (see Figure 6.26).

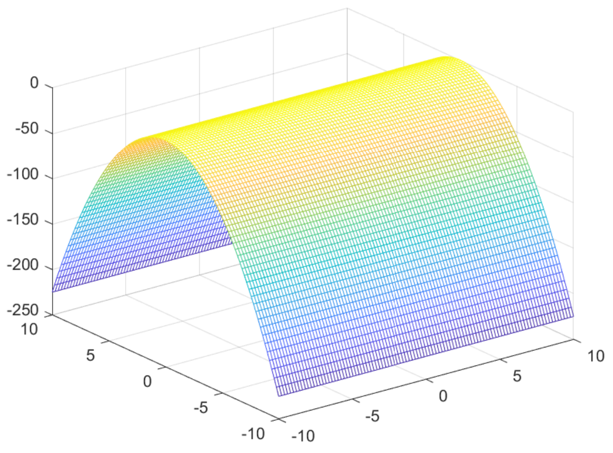


Figure 6.26: Plot of the derivative of the candidate function.

The Figures show that the selected function can be used as a Lyapunov function. Using (6.15), the value of input u can be substituted, and after simplifying the system, the augmented matrix-vector representation is obtained:

$$\boldsymbol{y} = \begin{pmatrix} \omega \\ \phi \\ \sin(\phi) \\ \cos(\phi) \end{pmatrix} \quad \boldsymbol{A} = \begin{pmatrix} -\frac{k}{m} & -\frac{g}{l} & -\frac{2g}{l} & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \boldsymbol{B_1} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \boldsymbol{y}_{jk} = \begin{pmatrix} 4 & 1 \\ 3 & 1 \end{pmatrix}. \tag{6.30}$$

For $t_{max} = 100\,\mathrm{s}$, $MTSM_h = 0.2\,\mathrm{s}$, tolerances for all used solvers $TOL = 1 \times 10^{-9}$ and $h_{scale} = 1$ the numerical results are in Table 6.14 and the plot of the ORD function is in Figure 6.27.

Solver	Time of calculation [s]	Number of steps	Ratio
MTSM	8.5387×10^{-3}	500	–
ode23	$3.795\,45 \times 10^{-1}$	81008	**44.45**
ode45	$1.334\,23 \times 10^{-2}$	9553	**1.56**
ode113	$1.352\,26 \times 10^{-2}$	1072	**1.58**

Table 6.14: Results for the Lyapunov based control system for $h = 0.2\,\mathrm{s}$, $h_{scale} = 1$.

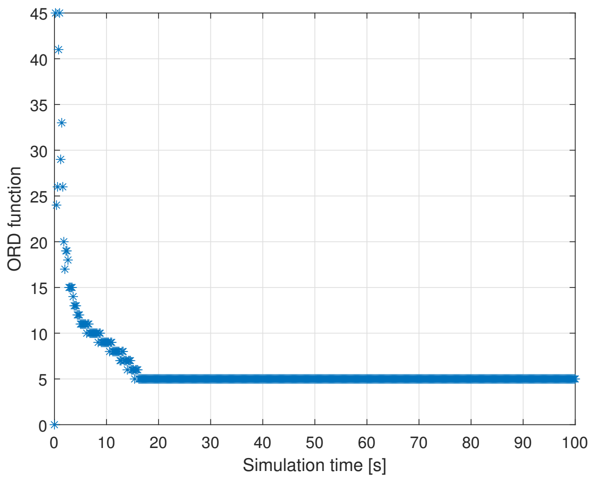


Figure 6.27: The plot of the ORD function for $h_{scale} = 1$.

When using additional optimizations implemented for non-linear MTSM, the performance can be improved substantially. Using $h_{scale} = 2$, the numerical results are in Table 6.15 and ORD function for $h_{scale} = 2$ is in Figure 6.28.

Solver	Time of calculation [s]	Number of steps	Ratio
MTSM	4.5361×10^{-3}	153	–
ode23	$3.816\,95 \times 10^{-1}$	81008	**84.15**
ode45	$1.456\,22 \times 10^{-2}$	9553	**3.21**
ode113	$1.432\,16 \times 10^{-2}$	1072	**3.16**

Table 6.15: Results for the Lyapunov based control system for $h = 0.2\,\mathrm{s}$, $h_{scale} = 2$.

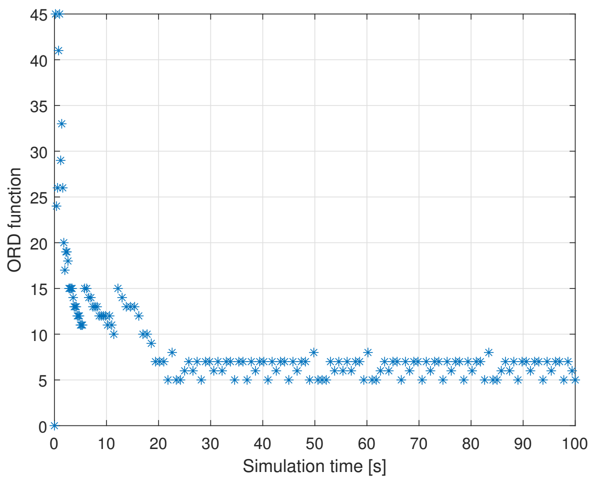


Figure 6.28: The plot of the ORD function for $h_{scale} = 2$.

The phase plot of the controlled system is in Figure 6.29, which shows that all pendulums stabilized in the original equilibrium point $(0, 0)$.

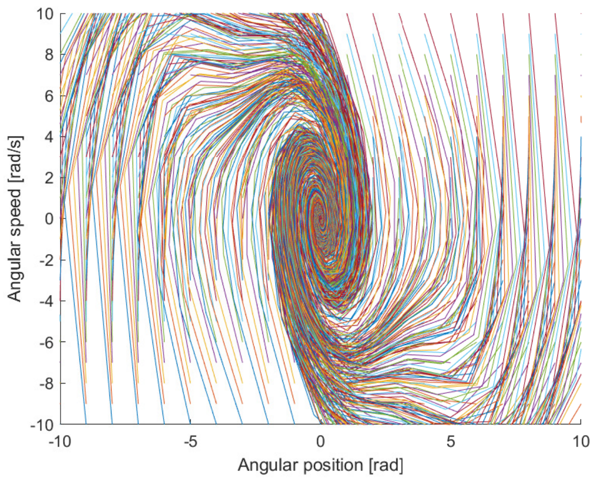


Figure 6.29: The phase plot of the experiment.

6.2.4 Concluding remarks

The pendulum problem showed the potential of MTSM with linear and non-linear control systems. The method performs better than state-of-the-art solvers in all tested examples and for all examined controllers by a wide margin. The implemented optimizations for the non-linear solver, detailed in Subsection 3.3.2, are beneficial.

6.3 Inverted pendulum on a cart

The experiments with just a simple pendulum may seem relatively simple. For a more demanding system, the inverted pendulum on a moving cart was chosen. It presents another set of challenges for designing a working controller, and the resulting system of ODEs is more complicated. The inverted pendulum on a cart problem is one of the most commonly analysed problems in modern control theory, and it represents a benchmark problem [13] due to its challenging nature. The problem is underactuated and has fewer control inputs then degrees of freedom [63], [24] and [29]. It is also highly non-linear, and the pendulum can behave chaotically.

6.3.1 Mathematical description

The problem can be visualized in Figure 6.30, where $x(t)$ represents the position of the cart and $\phi(t)$ is the angle referenced to the vertically upright position.

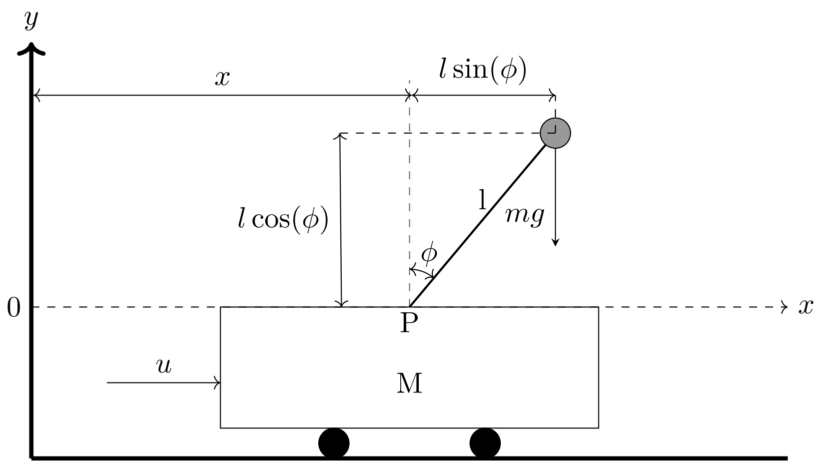


Figure 6.30: Inverted pendulum on the cart.

Assume that the rod of the pendulum and the hinge is massless and frictionless. The mass of the cart is denoted as M [kg], the mass of the point of the top of the pendulum as m [kg]. The external force u [N] is directed at the system in the x-axis direction.

The forces acting on the system have to be analysed. First, the forces acting on the cart and the pendulum in the x-axis direction. The sum of forces acting on the ball and the cart has to be equal to the external force u:

$$Mx'' + mx_G'' = u\,, \tag{6.31}$$

where x_G is the time-dependent centre of gravity given by the set of coordinates (x_G, y_G). According to Figure 6.30, the coordinates can be calculated as

$$\begin{aligned} x_G &= x + l\sin(\phi) \\ y_G &= l\cos(\phi) \end{aligned} \tag{6.32}$$

where l is the length of the rod of the pendulum. After substituting x_G into (6.31) (Appendix H.1) the final equation for the force in the x-axis direction can be written as

$$(M+m)\,x'' - ml\sin(\phi)\phi'^2 + ml\cos(\phi)\phi'' = u\,. \tag{6.33}$$

The torque balance of the system for the pendulum can be obtained using a similar principle. The torque created by acceleration force is balanced by the torque created by gravity. The resulting torque balance can therefore be written as

$$(F_x \cos(\phi))\, l - (F_y \sin(\phi))\, l = (mg \sin(\phi))\, l\,, \tag{6.34}$$

where F_x and F_y are the forces in the x and y direction respectively. After calculating the forces (Appendix H.2) the following system of non-linear ODEs is obtained:

$$(M+m)\,x'' - ml\sin(\phi)\phi'^2 + ml\cos(\phi)\phi'' = u \tag{6.35}$$

$$\cos(\phi)x'' + l\phi'' = g\sin(\phi)\,. \tag{6.36}$$

This system defines the pendulum on the cart system from Figure 6.30. The equations (6.35) and (6.36) have to be substituted into one another to obtain the final ODEs for the position of the cart and the angle of the pendulum.

For the *position of the cart* (Appendix H.3) the final ODE can be written as

$$x'' = \frac{u + ml\sin(\phi)\phi'^2 - mg\sin(\phi)\cos(\phi)}{M + m\sin^2(\phi)}\,, \tag{6.37}$$

and for the *angle of the pendulum* (Appendix H.4), the final ODE can be written as

$$\phi'' = \frac{-u\cos(\phi) + (Mg + mg)\sin(\phi) - ml\sin(\phi)\cos(\phi)\phi'^2}{Ml + ml\sin^2(\phi)}\,. \tag{6.38}$$

6.3.2 Used constants

For the following simulation experiments, the parameters of the systems were set according to Table 6.16.

Variable	Meaning	Value
m	mass of the point mass	1.5 kg
M	mass of the cart	10 kg
g	gravitational acceleration	$9.81\,\mathrm{m\cdot s^{-2}}$
l	length of the rod	1.5 m

Table 6.16: Values for the inverted pendulum on the cart simulation experiments.

6.3.3 Non-linear model control

Equations (6.38) and (6.37) cannot be solved as they are. They have to be transformed into a system of the first-order ODEs (see Section 3.4), and the states of the system have to be established for further analysis.

Transformation into state-space equations

For this example, the state variables are going to be denoted as y, because x was chosen to represent the position of the cart. The states of the system are going to represent:

- angular position of the pendulum ϕ [rad],
- angular speed of the pendulum ϕ' [rad·s^{-1}],
- position of the cart x [m],
- speed of the cart x' [m·s^{-1}].

The state variables for the states can be written as follows

$$\begin{aligned} y_1 &= \phi' & y_1' &= \phi'' \\ y_2 &= \phi & y_2' &= \phi' \\ y_3 &= x' & y_3' &= x'' \\ y_4 &= x & y_4' &= x' \end{aligned}$$

therefore equations (6.38) and (6.37) can be substituted into state definitions, so the state equations can be rewritten as

$$\begin{aligned} y_1' &= \frac{-u\cos(y_2) + (Mg + mg)\sin(y_2) - ml\sin(y_2)\cos(y_2)y_1y_1}{Ml + ml\sin^2(y_2)} & y_1(0) &= \phi_0' \\ y_2' &= y_1 & y_2(0) &= \phi_0 \\ y_3' &= \frac{u + ml\sin(y_2)y_1y_1 - mg\sin(y_2)\cos(y_2)}{M + m\sin^2(y_2)} & y_3(0) &= x_0' \\ y_4' &= y_3 & y_4(0) &= x_0\,. \end{aligned} \tag{6.39}$$

Transformation for MTSM

System (6.39) contains operation division and sine and cosine functions. Therefore it cannot be solved using MTSM. Transformations defined in Section 3.4 have to be performed to remove these operations from the system. Note that the additional equations are going to be represented as additional states of the system. First, the transformations for the sine and cosine functions can be performed

$$\begin{aligned} y_5 &= \sin(y_2) \\ y_5' &= \cos(y_2)y_2' = \cos(y_2)y_1 \\ y_6 &= \cos(y_2) \\ y_6' &= -\sin(y_2)y_2' = -\sin(y_2)y_1 \\ y_5' &= y_6y_1 & y_5(0) &= \sin(y_2(0)) = \sin(\phi_0) \\ y_6' &= -y_5y_1 & y_6(0) &= \cos(y_2(0)) = \cos(\phi_0) \end{aligned}$$

$$\begin{aligned} y_7 &= \sin^2(y_2) = y_5^2 \\ y_7' &= 2y_5y_5' = 2y_5y_6y_1 & y_7(0) &= y_5(0)^2\,. \end{aligned}$$

The denominator $\frac{1}{Ml+ml\sin^2(y_2)}$ can be replaced using the following system

$$\begin{aligned}
y_8 &= Ml + mly_7 \\
y_8' &= mly_7' = ml(2y_5y_6y_1) = 2mly_5y_6y_1 & y_8(0) &= Ml + mly_7(0) \\
\\
y_9 &= \frac{1}{y_8} = y_8^{-1} \\
y_9' &= -y_8^{-2}y_8' = -y_9^2y_8' = -y_9^2(2mly_5y_6y_1) \\
&= -2mly_9y_9y_5y_6y_1 & y_9(0) &= \frac{1}{y_8(0)}\,.
\end{aligned}$$

The denominator $\frac{1}{M+m\sin^2(y_2)}$ can be replaced using the following system

$$\begin{aligned}
y_{10} &= M + my_7 \\
y_{10}' &= my_7' = m(2y_5y_6y_1) = 2my_5y_6y_1 & y_{10}(0) &= M + my_7(0) \\
\\
y_{11} &= \frac{1}{y_{10}} = y_{10}^{-1} \\
y_{11}' &= -y_{10}{-2}y_{10}' = -y_{11}^2y_{10}' = -y_{11}^2(2my_5y_6y_1) \\
&= -2my_{11}y_{11}y_5y_6y_1 & y_{11}(0) &= \frac{1}{y_{10}(0)}\,.
\end{aligned}$$

Additionally, the $\sin(y_1)\cos(y_1)$ can be also replaced:

$$\begin{aligned}
y_{12} &= \sin(y_2)\cos(y_2) = y_5y_6 \\
y_{12}' &= y_5'y_6 + y_5y_6' = y_6y_1y_6 - y_5y_5y_1 & y_{12}(0) &= y_5(0)y_6(0)\,.
\end{aligned}$$

Substituting the obtained ODEs into (6.39) and simplifying

$$\begin{aligned}
y_1' &= -uy_6y_9 + (Mg+mg)y_5y_9 - mly_{12}y_1y_1y_9 & y_1(0) &= \phi_0' \\
y_2' &= y_1 & y_2(0) &= \phi_0 \\
y_3' &= uy_{11} + mly_5y_1y_1y_{11} - mgy_{12}y_{11} & y_3(0) &= x_0' \\
y_4' &= y_3 & y_4(0) &= x_0 \\
y_5' &= y_6y_1 & y_5(0) &= \sin(y_2(0)) = \sin(\phi_0) \\
y_6' &= -y_5y_1 & y_6(0) &= \cos(y_2(0)) = \cos(\phi_0) \\
y_7' &= 2y_{12}y_1 & y_7(0) &= y_5(0)^2 \\
y_8' &= 2mly_{12}y_1 & y_8(0) &= Ml + mly_7(0) \\
y_9' &= -2mly_9y_9y_{12}y_1 & y_9(0) &= \frac{1}{y_8(0)} \\
y_{10}' &= 2my_{12}y_1 & y_{10}(0) &= M + my_7(0) \\
y_{11}' &= -2my_{11}y_{11}y_{12}y_1 & y_{11}(0) &= \frac{1}{y_{10}(0)} \\
y_{12}' &= y_6y_1y_6 - y_5y_5y_1 & y_{12}(0) &= y_5(0)y_6(0)\,.
\end{aligned} \tag{6.40}$$

The matrix-vector representation of (6.40) is in Appendix G.1. For $u = 0$ and initial conditions $y_1(0) = 0\,\mathrm{rad{\cdot}s^{-1}}$, $y_2(0) = \pi + 0.1\,\mathrm{rad}$, $y_3(3) = 0\,\mathrm{m{\cdot}s^{-1}}$ and $y_4(0) = -1\,\mathrm{m}$, $MTSM_h = 0.5\,\mathrm{s}$ the results are visualized in Figure 6.31.

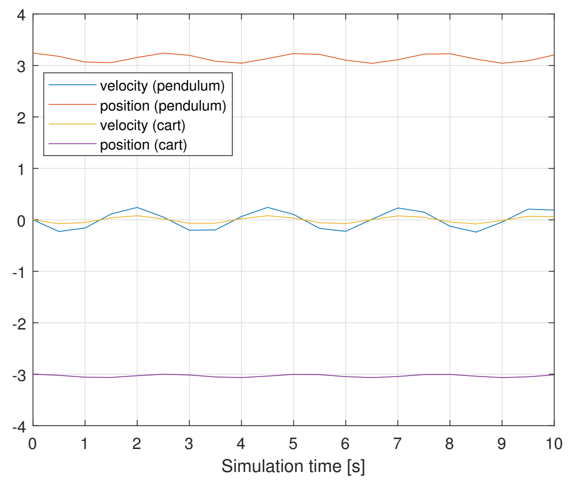


Figure 6.31: The behaviour of the pendulum on the cart without a controller.

The plot of the *ORD* function is in Figure 6.32 with the numerical results in Table 6.17.

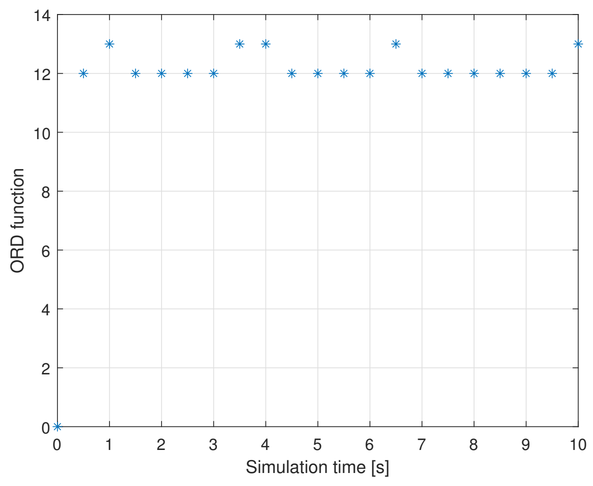


Figure 6.32: Plot of the *ORD* function with pendulum on the cart without a controller.

Solver	Time of calculation [s]	Number of steps	Ratio
MTSM	$4.623\,61 \times 10^{-3}$	20	–
ode23	$2.747\,23 \times 10^{-1}$	45288	**59.42**
ode45	1.0621×10^{-2}	5077	**2.3**
ode113	$5.384\,83 \times 10^{-3}$	387	**1.16**

Table 6.17: Numerical results for (6.40), $MTSM_h = 0.5\,\text{s}$.

6.3.4 Linear representation of the system

The linearization of the non-linear system (6.39) can again be performed. For the equilibrium point $\boldsymbol{y}*$

$$\boldsymbol{y}* = \begin{pmatrix} \phi & = \pi \\ x & = 0 \end{pmatrix}$$

the linearization can be calculated using the method outlined in Subsection 5.2.4:

$$\begin{aligned}
&\frac{\partial y_1}{\partial y_1} = 0 && \frac{\partial y_1}{\partial y_2} = -\frac{(M+m)g}{Ml} && \frac{\partial y_1}{\partial y_3} = 0 && \frac{\partial y_1}{\partial y_4} = 0 \\
&\frac{\partial y_2}{\partial y_1} = 1 && \frac{\partial y_2}{\partial y_2} = 0 && \frac{\partial y_2}{\partial y_3} = 0 && \frac{\partial y_2}{\partial y_4} = 0 \\
&\frac{\partial y_3}{\partial y_1} = 0 && \frac{\partial y_3}{\partial y_2} = \frac{mg}{M} && \frac{\partial y_3}{\partial y_3} = 0 && \frac{\partial y_3}{\partial y_4} = 0 \\
&\frac{\partial y_4}{\partial y_1} = 0 && \frac{\partial y_4}{\partial y_2} = 0 && \frac{\partial y_4}{\partial y_3} = 1 && \frac{\partial y_4}{\partial y_4} = 0 \\
&\frac{\partial y_1}{\partial u} = \frac{1}{Ml} && \frac{\partial y_2}{\partial u} = 0 && \frac{\partial y_3}{\partial u} = \frac{1}{M} && \frac{\partial y_4}{\partial u} = 0\,.
\end{aligned} \tag{6.41}$$

The performed linearization can be written in a matrix-vector notation

$$\boldsymbol{x}' = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}u = \begin{pmatrix} 0 & -\frac{(M+m)g}{Ml} & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & \frac{mg}{M} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \omega \\ \phi \\ v \\ x \end{pmatrix} + \begin{pmatrix} -\frac{1}{Ml} \\ 0 \\ \frac{1}{M} \\ 0 \end{pmatrix} u\,. \tag{6.42}$$

6.3.5 Non-linear model experiment

As a controller, the full state feedback controller is going to be implemented, as discussed in Subsection 5.4.2. For this experiment, the linear system (6.42) is going to be used to calculate $\boldsymbol{k}$ using the pole-placement method. Note that the eigenvalues of the (6.42) are $0 + 2.4261i$, $0 - 2.4261i$, 0, 0, therefore the system is unstable. To successfully apply the full state controller, the system (6.42) has to be controllable. The controllability matrix

$$\mathcal{C} = \begin{pmatrix} 0.1 & 0 & -0.5886 & 0 \\ 0 & 0.1000 & 0 & -0.5886 \\ 0.2 & 0 & 0.1962 & 0 \\ 0 & 0.2 & 0 & 0.1962 \end{pmatrix}$$

has the rank equal to the number of rows of $\boldsymbol{A}$, which means that the system is fully controllable and the controllers can be designed. For the new values of poles set by vector $\boldsymbol{p}$

$$\boldsymbol{p} = \begin{pmatrix} -1.8 \\ -1.7 \\ -1.6 \\ -1.5 \end{pmatrix}.$$

The vector $\boldsymbol{k}$ can be calculated by using the Ackerman function. The obtained value is

$$\boldsymbol{k} = \begin{pmatrix} 67.4439 & 118.8780 & 21.0374 & 8.6380 \end{pmatrix}.$$

The newly calculated vector $\boldsymbol{k}$ can be used to control the non-linear system (6.40) to achieve the reference value $\boldsymbol{w_r}$

$$\boldsymbol{w_r} = \begin{pmatrix} \phi_r' = 0 \\ \phi_r = \pi \\ v_r = 0 \\ x_r = 1 \end{pmatrix} .$$

The (5.20) can be used:

$$\begin{aligned} u &= -k_1(y_1 - r_1) - k_2(y_2 - r_2) - k_3(y_3 - r_3) - k_4(y_4 - r_4) + \boldsymbol{k w_r} = \\ &= -k_1 y_1 - k_2 y_2 - k_3 y_3 - k_4 y_4 + \boldsymbol{k w_r} . \end{aligned} \tag{6.43}$$

The control input u can be substituted into (6.40)

$$\begin{aligned} y_1' &= k_1 y_1 y_6 y_9 + k_2 y_2 y_6 y_9 + k_3 y_3 y_6 y_9 + k_4 y_4 y_6 y_9 - \boldsymbol{kr} y_6 y_9 + \\ &\quad + (Mg + mg) y_5 y_9 - mly_{12} y_1 y_1 y_9 && y_1(0) = \phi_0' \\ y_2' &= y_1 && y_2(0) = \phi_0 \\ y_3' &= -k_1 y_1 y_{11} - k_2 y_2 y_{11} - k_3 y_3 y_{11} - k_4 y_4 y_{11} + \boldsymbol{kr} y_{11} + \\ &\quad + mly_5 y_1 y_1 y_{11} - mgy_{12} y_{11} && y_3(0) = x_0' \\ y_4' &= y_3 && y_4(0) = x_0 \\ y_5' &= y_6 y_1 && y_5(0) = \sin(y_2(0)) \\ y_6' &= -y_5 y_1 && y_6(0) = \cos(y_2(0)) \\ y_7' &= 2 y_{12} y_1 && y_7(0) = y_5(0)^2 \\ y_8' &= 2mly_{12} y_1 && y_8(0) = Ml + mly_7(0) \\ y_9' &= -2mly_9 y_9 y_{12} y_1 && y_9(0) = \frac{1}{y_8(0)} \\ y_{10}' &= 2my_{12} y_1 && y_{10}(0) = M + my_7(0) \\ y_{11}' &= -2my_{11} y_{11} y_{12} y_1 && y_{11}(0) = \frac{1}{y_{10}(0)} \\ y_{12}' &= y_6 y_1 y_6 - y_5 y_5 y_1 && y_{12}(0) = y_5(0) y_6(0) . \end{aligned} \tag{6.44}$$

The matrix-vector representation of (6.44) is in Appendix G.2. The behaviour of the system using $t_{max} = 12\,\text{s}$, $MTSM_h = 0.25\,\text{s}$ with $TOL = 1 \times 10^{-9}$ for all solvers is in Figure 6.33.

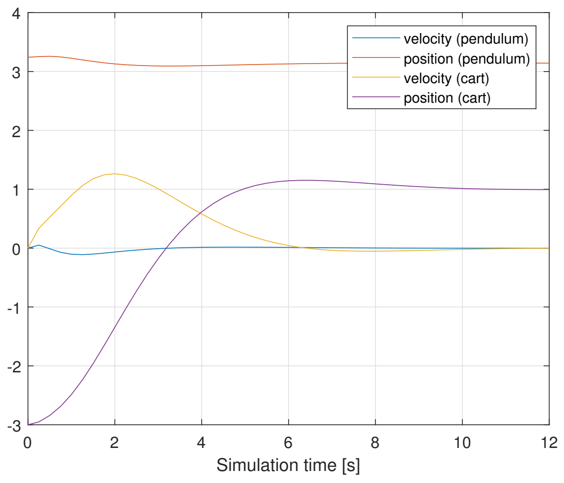


Figure 6.33: The behaviour of the pendulum on the cart with a full state feedback regulator.

The plot of the *ORD* function is in Figure 6.34, the numerical results are in Table 6.18.

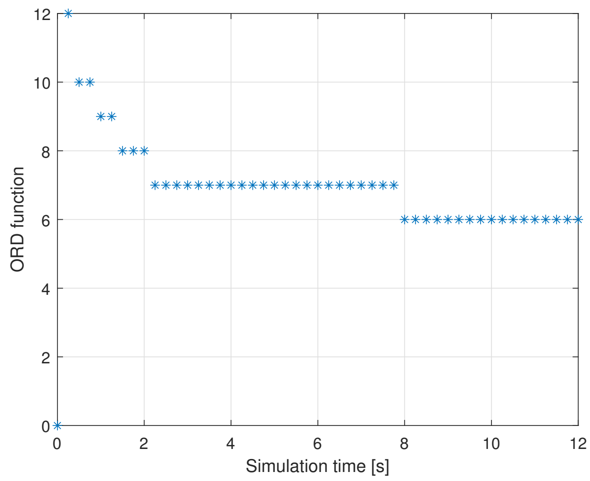


Figure 6.34: The plot of the *ORD* function.

Solver	Time of calculation [s]	Number of steps	Ratio
MTSM	$5.447\,98 \times 10^{-3}$	48	–
ode23	$1.539\,07 \times 10^{-1}$	23200	**28.25**
ode45	$6.351\,39 \times 10^{-3}$	2837	**1.17**
ode113	$3.681\,49 \times 10^{-3}$	239	0.68

Table 6.18: Results for the full state feedback controller $h = 0.25\,\mathrm{s}$.

The performance can be improved substantially by employing the step-size scaling defined in Subsection 3.3.3. When setting $h_{scale} = 4$, the results are in Table 6.19 with a plot of the *ORD* function in Figure 6.35.

Solver	Time of calculation [s]	Number of steps	Ratio
MTSM	$3.277\,06 \times 10^{-3}$	18	–
ode23	$1.612\,59 \times 10^{-1}$	23200	**49.21**
ode45	$6.351\,39 \times 10^{-3}$	2837	**2.24**
ode113	$3.906\,14 \times 10^{-3}$	239	**1.19**

Table 6.19: Results for the full state feedback controller $h = 0.25\,\mathrm{s}$.

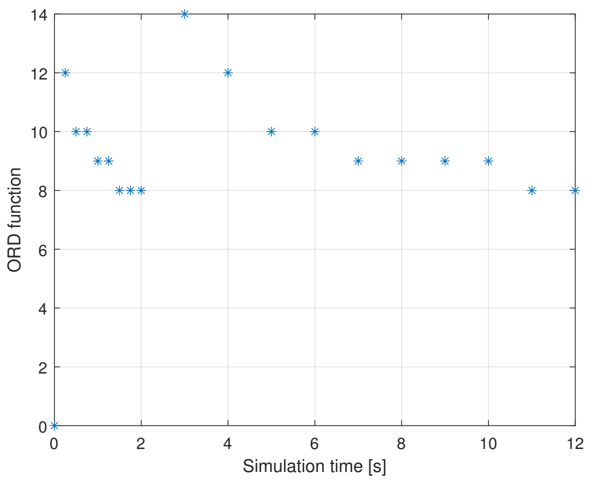


Figure 6.35: The plot of the ORD function with $h_{scale} = 4$.

6.3.6 Concluding remarks

This experiment shows the behaviour of MTSM when applied to a more complicated problem – the regulation of the inverted pendulum on a moving cart. The experiments show that the method performs the calculation faster than the state-of-the-art methods and that the performed optimizations have a significant positive impact on the performance of the method.

6.4 Magnetic levitation

In real-world industrial applications, magnetic levitation is used in many areas (for example, high-speed rail, vibration isolation systems, magnetic bearings, rocket guidance) [39]. It is heavily non-linear and therefore interesting in the context of this thesis as a control problem. The model of the system was derived, and experiments are mostly based on [2], [55], [1], [38], [11], [68], [22] and [39].

The magnetic levitation system is also interesting because, similarly to the DC motor discussed in Section 6.1, it combines mechanical and electromagnetic part that has to be modelled.

6.4.1 Mathematical description

The overall schematic of the magnetic levitation system is in Figure 6.36.

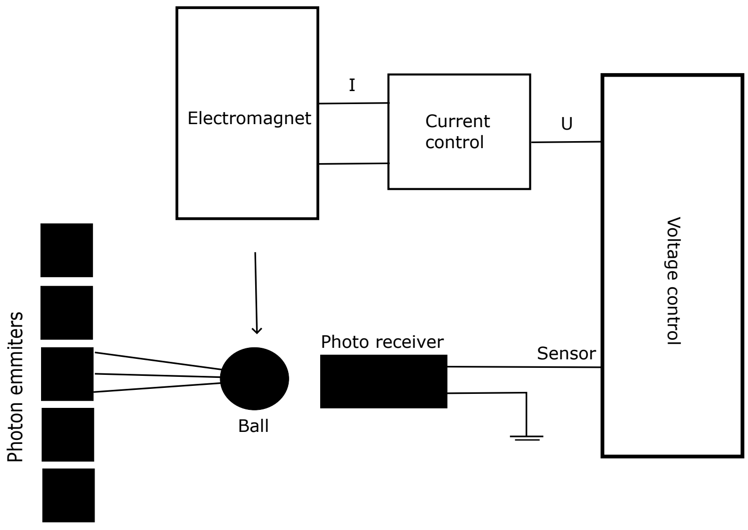


Figure 6.36: Schematic of the magnetic levitation system [1].

The electromagnet serves as an actuator while the sensor determines the position of the ferromagnetic ball. By regulating the electric current, the force generated by the electromagnet can be adjusted so that it is equal to the force of gravity that acts on the ball. The ball is going to levitate in the equilibrium state at the set distance from the electromagnet.

The electromagnetic part of the system can be modelled using the 2nd Kirchhoff´s law

$$u = U_R + U_L = iR + (L(x)i)' = iR + L(x)'i + L(x)i' \,, \tag{6.45}$$

where u is the applied voltage [V], i is the current in the coil of the electromagnet [A], R is the resistance of the coil [Ω] and $L(x)$ is a function describing the inductance of the electromagnet depending on the distance from the ball [H]. ODE for current can be obtained from (6.45) so that

$$\begin{aligned} u &= iR + L(x)'i + Li' \\ Li' &= u - iR - L(x)'i \\ i' &= \frac{u}{L} - \frac{R}{L}i - \frac{1}{L}L(x)'i \,. \end{aligned} \tag{6.46}$$

The function of inductance $L(x)$ can be approximated using several different approximations [38]. The most common one assumes that the inductance changes with the inverse of the position of the ball, that is

$$L(x) = L_\infty + \frac{L_0 x_0}{x}$$

where L_∞ represents the inductance of the coil when the ball is removed ($x = \infty$), and $L_\infty + L_0$ indicates the inductance when the ball is in contact with the electromagnet $x = 0$. The term $L_0 x_0$ represents a constant in the system that can be written as

$$c_1 = \frac{L_0 x_0}{2} \,.$$

The approximation for inductance $L(x)$ can therefore be written as

$$L(x) = L_\infty + \frac{2c_1}{x} \,. \tag{6.47}$$

Substituting (6.47) into (6.46) gives

$$i' = \frac{u}{L} - \frac{R}{L}i - \frac{1}{L}\left(L_\infty + \frac{2c_1}{x}\right)' i = \frac{u}{L} - \frac{R}{L}i - \frac{1}{L}\left(-\frac{2c_1}{x^2}x'\right) i = \frac{u}{L} - \frac{R}{L}i + \frac{2c_1}{L}\frac{i}{x^2}x', \tag{6.48}$$

with the initial condition $i(0) = i_0$. The mechanical part of the system can be modelled using the force diagram in Figure 6.37.

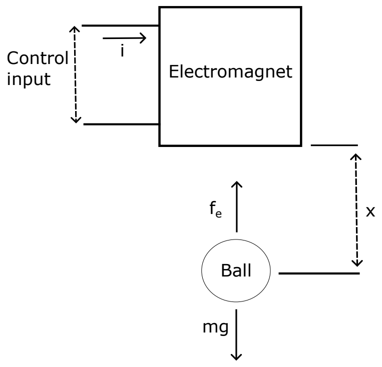


Figure 6.37: Force diagram of the mechanical part of the magnetic levitation system [38].

The dynamics of the levitated ball shown in Figure 6.37 can be written as [38]

$$mx'' = mg - f_e \tag{6.49}$$

where m denotes the mass of the ball [kg], x'' denotes the acceleration of the ball [m·s^{-2}] and g denotes the gravitational acceleration [m·s^{-2}]. The electromagnetic force f_e can be derived as follows [38, 11, 85]. Consider the energy stored in inductance W_e

$$W_e = \frac{1}{2}L(x)i^2 = \frac{1}{2}\left(L_\infty + \frac{2c_1}{x}\right)i^2 \tag{6.50}$$

where $L(x)$ again represents the inductance of the electromagnet [H] and i represents the current flowing through it [A]. The electromagnetic force is the partial derivative of W_e. With (6.47) substituted, the equation for the strength of the electromagnet can be written as

$$f_e = -\frac{\partial W_e}{\partial x} = -\frac{\partial\left[\frac{1}{2}\left(L_\infty + \frac{2c_1}{x}\right)i^2\right]}{\partial x} = \frac{i^2}{2}\frac{2c_1}{x^2} = c_1\frac{i^2}{x^2}. \tag{6.51}$$

Equation (6.51) can be substituted into (6.49) so that the final equation for the force in the system becomes

$$mx'' = mg - f_e = mg - c_1\frac{i^2}{x^2}. \tag{6.52}$$

Equation (6.52) is the second-order ODE, and it can be transformed into the system of first-order ODEs using MDOR (see Subsection 2.6.1):

$$\begin{aligned} mx'' &= mg - c_1\frac{i^2}{x^2} \\ x'' &= g - \frac{c_1}{m}\frac{i^2}{x^2} \\ v' &= g - \frac{c_1}{m}\frac{i^2}{x^2} \qquad && v(0) = v_0 \\ x' &= v && x(0) = x_0, \end{aligned} \tag{6.53}$$

where v is the velocity of the ball [m·s^{-1}] and x is the position of the ball [m]. To create a system representation, the state variable $\boldsymbol{y} = (v', x', i')^T$ is constructed, and a non-linear model of the magnetic levitation system can therefore be written in the notation for non-linear systems (Section 5.3):

$$f(y) = \begin{pmatrix} g - \frac{c_1}{m}\frac{y_3^2}{y_2^2} \\ y_1 \\ -\frac{R}{L}y_3 + \frac{2c_1}{L}\frac{y_3}{y_2^2}y_1 \end{pmatrix}, \; g(y) = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{L} \end{pmatrix}, \; y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \tag{6.54}$$

which means that the output of the system is the position of the ball.

For the experiments with the magnetic levitation system, the following values of constants are going to be used.

Parameter	Description	Value
m	mass of the ball	0.02 kg
g	gravitational acceleration	9.81 m·s^{-2}
R	resistance of the electromagnet	0.95 Ω
c_1	electromagnetic force constant	2.483 156 32 N·m^2·A^{-2}
L	inductance of the electromagnet	0.277 H

Table 6.20: Used constants [38].

The behaviour of the system without any control input is in Figure 6.38. The ball is not kept at the set distance from the electromagnet and keeps moving away due to gravitational acceleration.

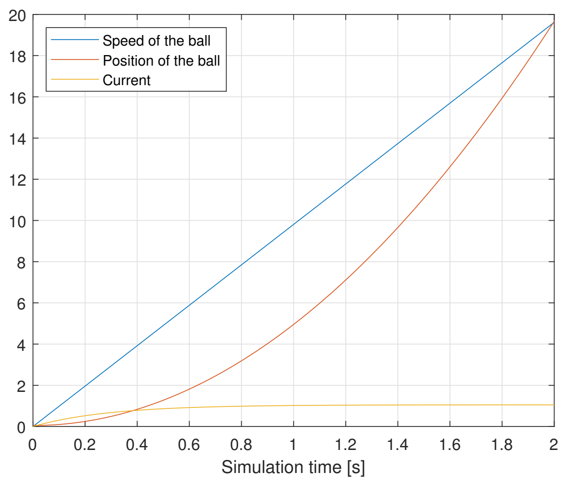


Figure 6.38: Behaviour of the magnetic levitation system without a controller.

To simulate the system using MTSM, the transformations have once again be performed.

6.4.2 Transformations for MTSM

To solve (6.54) using MTSM, the system has to be transformed into an appropriate form (Section 3.4). Only the functions $f(y)$ and $g(y)$ are transformed (not equations for output).

Therefore, the system (6.54) can be rewritten:

$$\begin{aligned} y_1' &= g - \frac{c_1}{m}\frac{y_3y_3}{y_2y_2} && y_1(0) = v(0) \\ y_2' &= y_1 && y_2(0) = x(0) \\ y_3' &= -\frac{R}{L}y_3 + \frac{2c_1}{L}\frac{y_1y_3}{y_2y_2} + \frac{1}{L}u && y_3(0) = i(0)\,. \end{aligned} \tag{6.55}$$

First, the division $(\frac{1}{y_2y_2})$ has to be removed. The auxiliary equation to remove it has the following form

$$\begin{aligned} y_4 &= \frac{1}{y_2} = y_2^{-1} \\ y_4' &= -y_2^{-2}y_2' = -y_1y_4y_4 && y_4(0) = \frac{1}{y_2(0)}\,, \end{aligned}$$

so that the new system with the equation substituted is without division can be written as

$$\begin{aligned} y_1' &= g - \frac{c_1}{m}(y_3y_3y_4y_4) && y_1(0) = v(0) \\ y_2' &= y_1 && y_2(0) = x(0) \\ y_3' &= -\frac{R}{L}y_3 + \frac{2c_1}{L}(y_1y_3y_4y_4) + \frac{1}{L}u && y_3(0) = i_0 \\ y_4' &= -y_1y_4y_4 && y_4(0) = \frac{1}{y_2(0)}\,. \end{aligned} \tag{6.56}$$

The system (6.56) has four-function multiplications that can be removed to optimize the calculation. First, the multiplication y_4y_4 can be replaced. The auxiliary equation

$$\begin{aligned} y_5 &= y_4y_4 = y_4^2 \\ y_5' &= 2y_4y_4' = 2y_4(-y_4y_4y_1) = -2y_4y_4y_4y_1 = -2y_1y_4y_5 && x_5(0) = y_4(0)y_4(0)\,, \end{aligned}$$

and the new system with the equation y_5 substituted

$$\begin{aligned} y_1' &= g - \frac{c_1}{m}(y_3y_3y_5) && y_1(0) = v(0) \\ y_2' &= y_1 && y_2(0) = x(0) \\ y_3' &= -\frac{R}{L}y_3 + \frac{2c_1}{L}(y_1y_3y_5) + \frac{1}{L}u && y_3(0) = i_0 \\ y_4' &= -y_1y_5 && y_4(0) = \frac{1}{y_2(0)} \\ y_5' &= -2y_1y_4y_5 && x_5(0) = y_4(0)y_4(0)\,. \end{aligned} \tag{6.57}$$

The same process can be repeated for the multiplication y_3y_3. The auxiliary equation

$$\begin{aligned} y_6 &= y_3y_3 = y_3^2 \\ x_6' &= 2y_3y_3' = 2y_3\left(-\frac{R}{L}y_3 + \frac{2k_1}{L}(y_1y_3y_5) + \frac{1}{L}u\right) = -2\frac{R}{L}y_3y_3 + 2\frac{2k_1}{L}(y_1y_3y_3y_5) + 2\frac{1}{L}uy_3 \\ &= \frac{-2R}{L}y_6 + \frac{4k_1}{L}(y_1y_6y_5) + \frac{2}{L}uy_3 \qquad y_6(0) = y_3(0)y_3(0)\,, \end{aligned}$$

and the substituted system can be written as

$$
\begin{aligned}
y_1' &= g - \frac{c_1}{m}\left(y_5 y_6\right) && y_1(0) = v(0) \\
y_2' &= y_1 && y_2(0) = x(0) \\
y_3' &= -\frac{R}{L} y_3 + \frac{2c_1}{L}\left(y_1 y_3 y_5\right) + \frac{1}{L} u && y_3(0) = i_0 \\
y_4' &= -y_1 y_5 && y_4(0) = \frac{1}{y_2(0)} \\
y_5' &= -2 y_1 y_4 y_5 && x_5(0) = y_4(0) y_4(0) \\
y_6' &= \frac{-2R}{L} y_6 + \frac{4c_1}{L}\left(y_1 y_5 y_6\right) + \frac{2}{L} u y_3 && y_6(0) = y_3(0) y_3(0)\,.
\end{aligned}
\tag{6.58}
$$

The final variant of the original system can be obtained by substituting the multiplication $y_1 y_5$, which is repeated often. The auxiliary equation

$$
\begin{aligned}
&y_7 = y_1 y_5 \\
&y_7' = y_1' y_5 + y_1 y_5' = \left[g - \frac{c_1}{m}\left(y_5 y_6\right)\right] y_5 + y_1\left(-2 y_1 y_4 y_5\right) = g y_5 - \frac{k_1}{m}\left(y_5 y_5 y_6\right) - 2 y_1 y_1 y_4 y_5 = \\
&\quad = g y_5 - \frac{k_1}{m}\left(y_5 y_5 y_6\right) - 2 y_1 y_4 y_7 \qquad y_7(0) = y_1(0) y_5(5)
\end{aligned}
$$

and the substituted system

$$
\begin{aligned}
y_1' &= g - \frac{c_1}{m}\left(y_5 y_6\right) && y_1(0) = v(0) \\
y_2' &= y_1 && y_2(0) = x(0) \\
y_3' &= -\frac{R}{L} y_3 + \frac{2c_1}{L}\left(y_3 y_7\right) + \frac{1}{L} u && y_3(0) = i_0 \\
y_4' &= -y_7 && y_4(0) = \frac{1}{y_2(0)} \\
y_5' &= -2 y_4 y_7 && x_5(0) = y_4(0) y_4(0) \\
y_6' &= \frac{-2R}{L} y_6 + \frac{4c_1}{L}\left(y_6 y_7\right) + \frac{2}{L} u y_3 && y_6(0) = y_3(0) y_3(0) \\
y_7' &= g y_5 - \frac{c_1}{m}\left(y_5 y_5 y_6\right) - 2 y_1 y_4 y_7 && y_7(0) = y_1(0) y_5(5)\,.
\end{aligned}
\tag{6.59}
$$

With $t_{max} = 2\,\mathrm{s}$ and $TOL = 1 \times 10^{-8}$, the behaviour of MTSM and state-of-the-art solvers for the generated auxiliary systems is depicted in the following set of tables. The numerical results using (6.56) are in Table 6.21, the plot of the ORD function is in Figure 6.39.

Solver	Time of calculation [s]	Number of steps	Ratio
MTSM	$5.435\,35 \times 10^{-3}$	26	–
ode23	$1.440\,22 \times 10^{-1}$	24025	**26.04**
ode45	$4.987\,55 \times 10^{-3}$	2533	0.9
ode113	3.4645×10^{-3}	235	0.62

Table 6.21: Numerical results for (6.56), $MTSM_h = 0.08\,\mathrm{s}$, $h_{scale} = 1$.

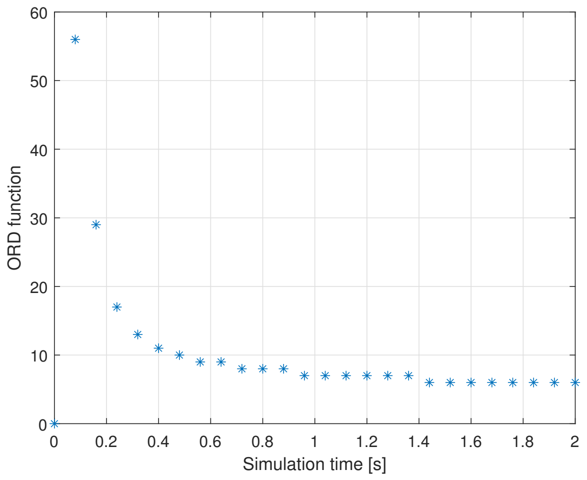


Figure 6.39: Plot of ORD function for for (6.56), $MTSM_h = 0.08\,\mathrm{s}$, $h_{scale} = 1$.

For $h_{scale} = 3$, the results for (6.56) are in Table 6.22 and the plot of the ORD function is in Figure 6.40.

Solver	Time of calculation [s]	Number of steps	Ratio
MTSM	3.2744×10^{-3}	17	–
ode23	1.46792×10^{-1}	24025	**44.83**
ode45	4.9825×10^{-3}	2533	**1.52**
ode113	3.63135×10^{-3}	235	**1.1**

Table 6.22: Numerical results for (6.56), $MTSM_h = 0.08\,\mathrm{s}$, $h_{scale} = 3$.

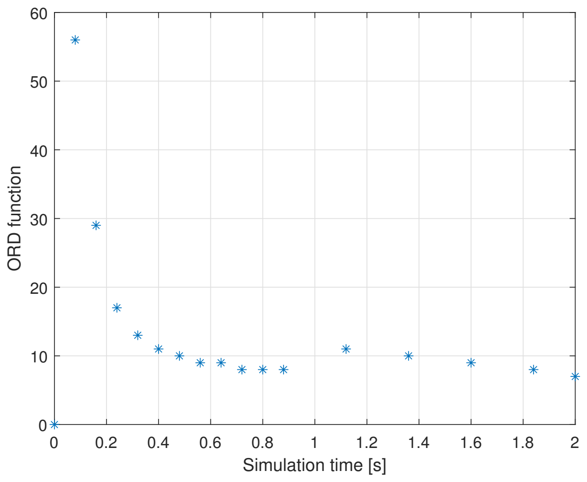


Figure 6.40: Plot of ORD function for for (6.56), $MTSM_h = 0.08\,\mathrm{s}$, $h_{scale} = 3$.

The step size scaling can be used for all auxiliary systems. The numerical results using (6.57) with $h_{scale} = 5$ are in Table 6.23, the plot of the ORD function is in Figure 6.41.

Solver	Time of calculation [s]	Number of steps	Ratio
MTSM	2.592×10^{-3}	24	–
ode23	$2.273\,49 \times 10^{-1}$	36248	**87.71**
ode45	$7.369\,55 \times 10^{-3}$	3653	**2.84**
ode113	4.619×10^{-3}	299	**1.78**

Table 6.23: Numerical results for (6.57), $MTSM_h = 0.065\,\text{s}$, $h_{scale} = 5$.

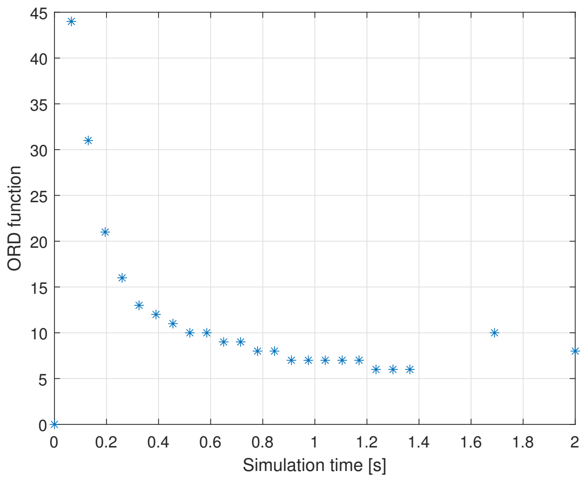


Figure 6.41: Plot of ORD function for for (6.57), $MTSM_h = 0.065\,\text{s}$, $h_{scale} = 5$.

The numerical results using (6.58) with $h_{scale} = 5$ are in Table 6.24, the plot of the ORD function is in Figure 6.42.

Solver	Time of calculation [s]	Number of steps	Ratio
MTSM	3.1588×10^{-3}	20	–
ode23	$3.939\,14 \times 10^{-1}$	47571	**124.7**
ode45	$1.288\,41 \times 10^{-2}$	4641	**4.08**
ode113	$5.792\,75 \times 10^{-3}$	343	**1.83**

Table 6.24: Numerical results for (6.58), $MTSM_h = 0.065\,\text{s}$, $h_{scale} = 5$.

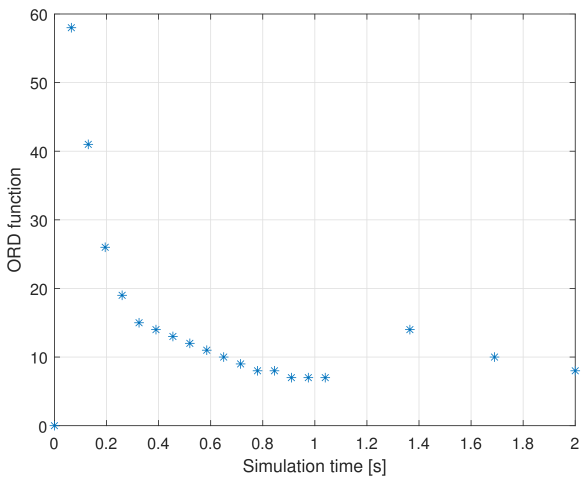


Figure 6.42: Plot of ORD function for (6.58), $MTSM_h = 0.065\,\mathrm{s}$, $h_{scale} = 5$.

The numerical results using (6.59) with $h_{scale} = 3$ are in Table 6.25, the plot of the ORD function is in Figure 6.43.

Solver	Time of calculation [s]	Number of steps	Ratio
MTSM	4.9234×10^{-3}	22	–
ode23	$4.851\,73 \times 10^{-1}$	53805	**98.54**
ode45	$1.387\,96 \times 10^{-2}$	4457	**2.82**
ode113	$6.153\,85 \times 10^{-3}$	354	**1.25**

Table 6.25: Numerical results for (6.59).

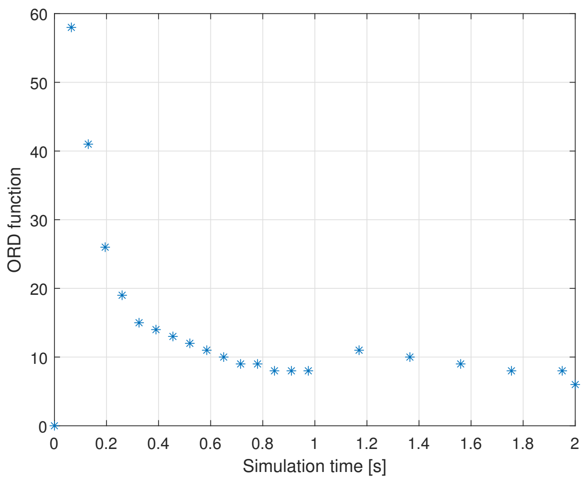


Figure 6.43: Plot of ORD function for for (6.59), $MTSM_h = 0.065\,\mathrm{s}$, $h_{scale} = 3$.

The results show that MTSM is measurably faster for all systems with auxiliary equations when optimizations from Subsection 3.3.2 are used. System (6.56) was chosen for the experiment with a controller.

6.4.3 Control system design

Without an appropriate controller, the ball would fall due to gravity and would not maintain its position (Figure 6.38 and Figure 6.44).

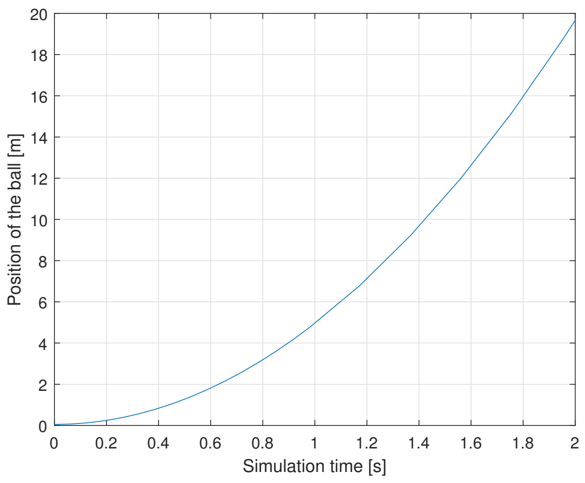


Figure 6.44: Plot the position of the ball.

The full-state feedback controller with an integrator is going to be implemented (Subsection 5.4.2). The reference vector $\boldsymbol{w}_r$ sets the control objective for the controller

$$\boldsymbol{w}_r = \begin{pmatrix} 0 \\ 0.009 \\ 0.8 \\ 0 \\ 0 \end{pmatrix},$$

where w_{r1} is the required final speed of the ball, w_{r2} is the required final position of the ball, w_{r3} is the required final current and w_{r5} is the required final value of the integral part of the controller. The integral part of the controller is represented using the following ODE

$$y_5' = y_2 - w_{r2} \qquad\qquad y_5(0) = 0\,. \tag{6.60}$$

The final system of ODEs with the controller added follows

$$\begin{aligned}
y_1' &= g - \frac{c_1}{m}\left(y_3 y_3 y_4 y_4\right) & y_1(0) &= v(0) \\
y_2' &= y_1 & y_2(0) &= x(0) \\
y_3' &= -\frac{R}{L} y_3 + \frac{2c_1}{L}\left(y_1 y_3 y_4 y_4\right) + \frac{1}{L} u & y_3(0) &= i_0 \\
y_4' &= -y_1 y_4 y_4 & y_4(0) &= \frac{1}{y_2(0)} \\
y_5' &= y_2 - w_{r2} & y_5(0) &= 0\,.
\end{aligned} \tag{6.61}$$

The full state feedback (see Subsection 5.4.2) is given by following equation

$$u = -\boldsymbol{k}(\boldsymbol{y} - \boldsymbol{w}_r)\,,$$

where $\boldsymbol{k} = (-57.7, -2762.6, 25.8, 0, -2000)$ [38]. Control law u can be rewritten part-wise

$$
\begin{aligned}
u &= k_1(w_{r1} - y_1) + k_2(w_{r2} - y_2) + k_3(w_{r3} - y_3) + k_4(w_{r4} - y_4) + k_5(w_{r5} - y_5) \\
u &= k_1 w_{r1} - k_1 y_1 + k_2 w_{r2} - k_2 y_2 + k_3 w_{r3} - k_3 y_3 + k_4 w_{r4} - k_4 y_4 + k_5 w_{r5} - k_5 y_5 \,.
\end{aligned}
$$

After substituting u into y_3, the final matrix-vector form for the (6.61) and (6.60) the matrix-vector representation that is going to be solved can be written as

$$
\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ -\frac{k_1}{L} & -\frac{k_2}{L} & -\frac{k_3}{L} - \frac{R}{L} & 0 & -\frac{k_5}{L} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \quad \boldsymbol{B}_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \\ 0 \end{pmatrix} \quad \boldsymbol{B}_3 = \begin{pmatrix} -\frac{c_1}{m} & 0 \\ 0 & 0 \\ 0 & \frac{2c_1}{L} \\ 0 & 0 \\ 0 & 0 \end{pmatrix}
$$
$$
\boldsymbol{b} = \boldsymbol{e} = \begin{pmatrix} g \\ 0 \\ \frac{w_r}{L} \\ 0 \\ -w_{r2} \end{pmatrix} \quad \boldsymbol{y}_{ijk} = \begin{pmatrix} 4 & 4 & 1 \end{pmatrix} \quad \boldsymbol{y}_{ijkl} = \begin{pmatrix} 3 & 3 & 4 & 4 \\ 3 & 1 & 4 & 4 \end{pmatrix} . \tag{6.62}
$$

The plot of the ORD function is in Figure 6.45.

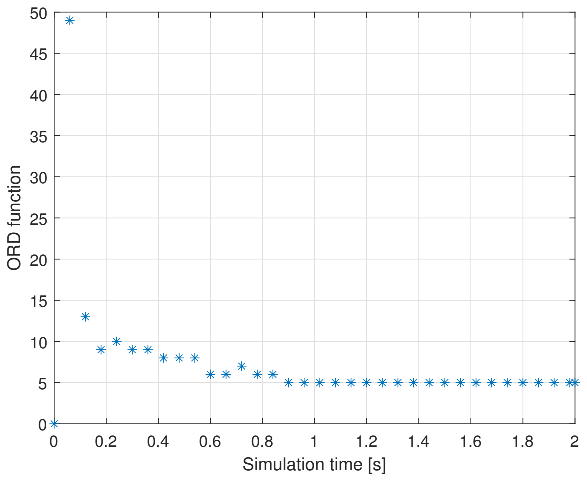


Figure 6.45: Plot of ORD function for for (6.62), $MTSM_h = 0.06$ s, $h_{scale} = 1$.

Note that Figure 6.45 shows that step-size scaling can again be used. For $h_{scale} = 2$, $MTSM_h = 0.06$ s the numerical results for (6.62) are in Table 6.26 and the plot of the ORD function is in Figure 6.46.

Solver	Time of calculation [s]	Number of steps	Ratio
MTSM	4.2185×10^{-3}	27	–
ode23	$9.358\,46 \times 10^{-2}$	11890	**22.18**
ode45	6.9611×10^{-3}	2533	**1.65**
ode113	5.2627×10^{-3}	308	**1.25**

Table 6.26: Numerical results for (6.62), $h_{scale} = 2$, $MTSM_h = 0.06$ s.

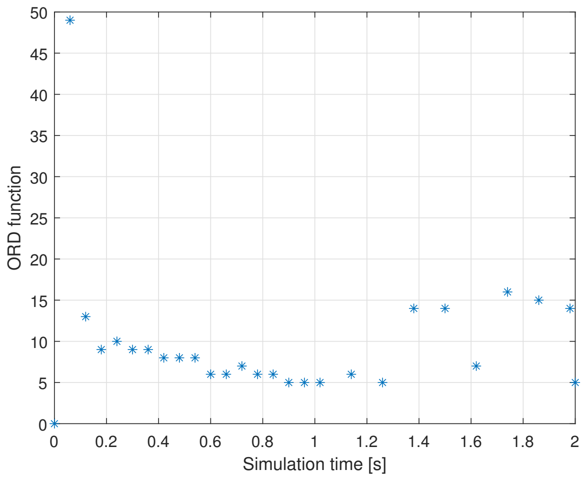


Figure 6.46: Plot of ORD function for for (6.62), $MTSM_h = 0.06\,\mathrm{s}$, $h_{scale} = 2$.

The implemented controller successfully stabilizes the ball at the required position, as shown in Figure 6.47.

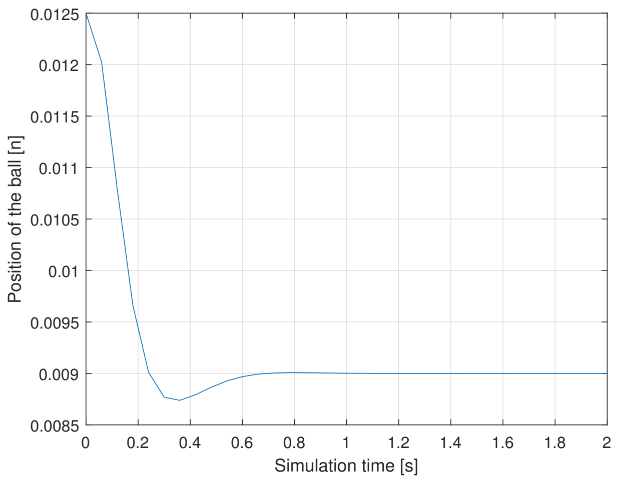


Figure 6.47: Position of the ball for (6.62), $MTSM_h = 0.06\,\mathrm{s}$, $h_{scale} = 2$.

Controller for (6.57)

The final experiment is going to show how the better performance of the auxiliary system makes the controller. The construction is going to be performed similarly to the previous example.

The system being solved can be represented in the matrix-vector notation as

$$\boldsymbol{A}=\begin{pmatrix}0&0&0&0&0&0\\1&0&0&0&0&0\\-\frac{k_1}{L}&-\frac{k_2}{L}&-\frac{k_3}{L}-\frac{R}{L}&0&-\frac{k_5}{L}&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&1&0&0&0&0\end{pmatrix}\quad \boldsymbol{B}_1=\begin{pmatrix}0\\0\\0\\-1\\0\\0\end{pmatrix}\quad \boldsymbol{B}_2=\begin{pmatrix}-\frac{c_1}{m}&0&0\\0&0&0\\0&\frac{2c_1}{L}&0\\0&0&0\\0&0&-2\\0&0&0\end{pmatrix}$$
$$\boldsymbol{b}=\boldsymbol{e}=\begin{pmatrix}g\\0\\\frac{\boldsymbol{k}\boldsymbol{w}_r}{L}\\0\\0\\-w_{r2}\end{pmatrix}\qquad \boldsymbol{y}_{ijk}=\begin{pmatrix}5&1\end{pmatrix}\quad \boldsymbol{y}_{ijk}=\begin{pmatrix}3&3&5\\3&1&5\\5&4&1\end{pmatrix}, \tag{6.63}$$

with $\boldsymbol{k}=(-57.7,-2762.6,25.8,0,0,-2000)$ and the reference vector $\boldsymbol{w}_r$

$$\boldsymbol{w}_r=\begin{pmatrix}0\\0.009\\0.8\\0\\0\\0\end{pmatrix}.$$

The numerical results for (6.63) are in Table 6.27 and the plot of the *ORD* function is in Figure 6.48.

Solver	Time of calculation [s]	Number of steps	Ratio
MTSM	$2.900\,05\times10^{-3}$	30	–
ode23	$1.186\,52\times10^{-1}$	13855	**40.91**
ode45	$9.092\,45\times10^{-3}$	2753	**3.13**
ode113	5.5814×10^{-3}	334	**1.92**

Table 6.27: Numerical results for (6.63), $h_{scale}=2$, $MTSM_h=0.06\,$s.

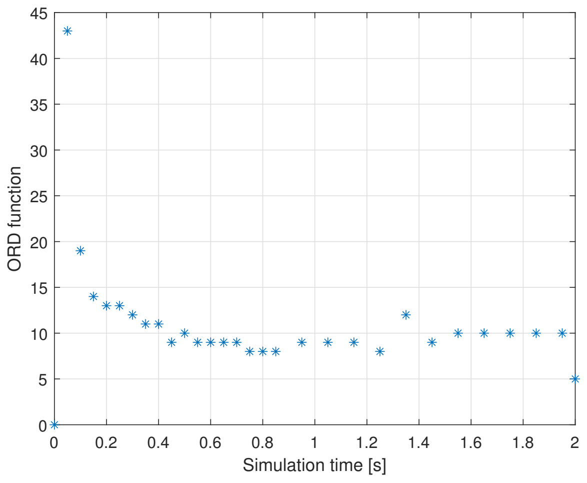


Figure 6.48: Plot of *ORD* function for for (6.63), $MTSM_h=0.06\,$s, $h_{scale}=2$.

The implemented controller again successfully stabilizes the ball at the required position as can be seen in Figure 6.49.

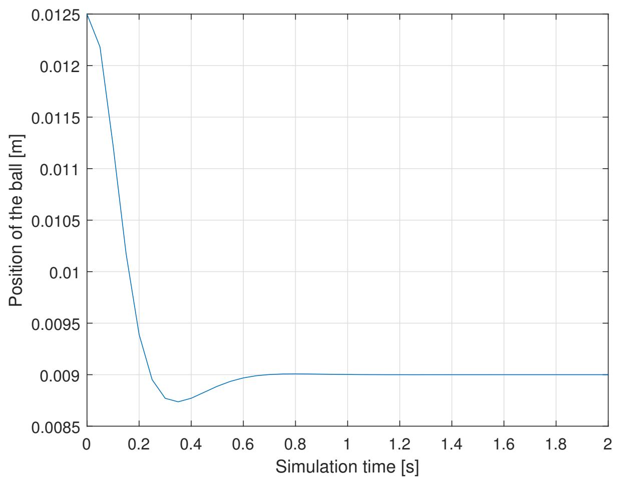


Figure 6.49: Position of the ball for (6.63), $MTSM_h = 0.06\,\mathrm{s}$, $h_{scale} = 2$.

6.4.4 Concluding remarks

The experiments with magnetic levitation again showed the usability of MTSM to solve non-linear technical problems and apply different control approaches to them. The method was again faster than the state-of-the-art in all experiments.

6.5 Conclusions

The experiments presented in this Chapter showed that the method is usable when designing the control systems of different types. MTSM is universally faster than the state-of-the-art numerical integration methods selected for comparison.

Chapter 7

Conclusion

This thesis presented the variable-order variable-step numerical integration method based on the Taylor series – MTSM. The method was first introduced in linear and newly developed non-linear form, and its positive properties were discussed. Then, it was used to solve a wide range of real-world technical problems and benchmarks to show that it can deal with them better than state-of-the-art methods. In the final Chapter of the thesis, a set of experiments with different types of controllers and systems was performed.

The research objectives stated in Section 1.1 were completed. Namely:

- *discuss the currently existing numerical integration methods, particularly in the context of control systems*
  Chapter 2 discussed various numerical integration methods, their advantages and disadvantages, and discussed what methods would be appropriate to use in a control system.

- *analyse the properties of a high-order integration method based on the Taylor series and evaluate the applicability of this method on a set of technical problems*
  The analysis was performed in Chapter 3, including the discussion on stability, error accumulation, automatic order settings and more. The evaluation of the method on a wide range of technical problems was performed in Chapter 4.

- *extend the capabilities of high-order Taylor series method to solve non-linear problems, propose and discuss possible optimizations*
  The newly developed non-linear version of MTSM was introduced in Chapter 3 and extensively tested in Chapter 4.

- *show the suitability of the high-order method to be used as a part of the control system using a set of examples (with strict considerations of time of calculation)*
  The experiments with several different control systems were performed in Chapter 6.

- *show the potential for further research and additional improvements of the method for both linear and non-linear problems*
  The potential for future research is established in Section 7.1. The non-linear solver, while improved considerably by introduced optimizations, could be improved further to extend its abilities. Additional support for variable precision arithmetic can also increase the method's potential due to the ability to use higher order.

7.1 Future work

As stated in the body of the thesis, the discussed topics cover a huge area of applied mathematics and control engineering. More work would have to be done to use the method in the real-world control system, including the possibility of designing custom-made specialised hardware for the method (the cooperation with the industrial partner did not work out for this thesis).

Some areas (like the calculation of non-linear systems) were not on the table, and the usefulness of having the plant model with non-linearities that could better handle boundary conditions seems obvious. The non-linear implementation of the method can still be considerably improved, even beyond what has already been done for this thesis and other projects. That will be the first topic of my research after the defence of this thesis (different implementation for operation division that would save one multiplication, for example).

Additionally, the thesis shows that the method is very useful in solving a wide range of technical problems. The application domain has thus far been experimental. The goal is to find a proper application for the method that would see it gain more recognition and adoption among the community.

Bibliography

[1] Ahmad, I. and Javaid, M. A. Nonlinear model & controller design for magnetic levitation system. *Recent advances in signal processing, robotics and automation.* 2010, p. 324–328.

[2] Ahmad, I., Shahzad, M. and Palensky, P. Optimal PID control of magnetic levitation system using genetic algorithm. In: IEEE. *2014 IEEE International Energy Conference (ENERGYCON).* 2014, p. 1429–1433.

[3] Amodio, P., Iavernaro, F., Mazzia, F., Mukhametzhanov, M. S. and Sergeyev, Y. D. A generalized Taylor method of order three for the solution of initial value problems in standard and infinity floating-point arithmetic. *Mathematics and Computers in Simulation.* Elsevier. 2017, vol. 141, p. 24–39.

[4] Baeza, A., Boscarino, S., Mulet, P., Russo, G. and Zorío, D. Approximate Taylor methods for ODEs. *Computers & Fluids.* Elsevier. 2017, vol. 159, p. 156–166. DOI: 10.1016/j.compfluid.2017.10.001. Available at: http://www.sciencedirect.com/science/article/pii/S0045793017303584.

[5] Baker, G. L. and Blackburn, J. A. *The pendulum: a case study in physics.* Oxford University Press, 2005.

[6] Barrio, R., Blesa, F. and Lara, M. VSVO formulation of the Taylor method for the numerical solution of ODEs. *Computers & mathematics with Applications.* Elsevier. 2005, vol. 50, 1-2, p. 93–111.

[7] Barrio, R. Performance of the Taylor series method for ODEs/DAEs. *Applied Mathematics and Computation.* Elsevier. 2005, vol. 163, no. 2, p. 525–545. ISSN 00963003.

[8] Barrio, R., Rodríguez, M., Abad, A. and Blesa, F. Breaking the limits: the Taylor series method. *Applied mathematics and computation.* Elsevier. 2011, vol. 217, no. 20, p. 7940–7954.

[9] Bartsch, H.-J. *Handbook of mathematical formulas.* Academic press, 2014. ISBN 1483235904.

[10] Basilio, J. and Moreira, M. State–Space Parameter Identification in a Second Control Laboratory. *IEEE Transactions on Education.* 2004, p. 204–210.

[11] Benomair, A. *Non-linear observer based control of magnetic levitation systems.* 2018. Dissertation. University of Sheffield.

[12] Bobek, M. *Analogové počítače.* Nakladatelství technické literatury ALFA, 1982. 380 p. ISBN 04-510-82.

[13] Boubaker, O. The inverted pendulum benchmark in nonlinear control theory: a survey. *International Journal of Advanced Robotic Systems.* SAGE Publications Sage UK: London, England. 2013, vol. 10, no. 5, p. 233.

[14] Brezinski, C. Extrapolation algorithms and Padé approximations: a historical survey. *Applied Numerical Mathematics.* 1996, vol. 20, no. 3, p. 299–318. DOI: https://doi.org/10.1016/0168-9274(95)00110-7. ISSN 0168-9274. Available at: https://www.sciencedirect.com/science/article/pii/0168927495001107.

[15] Broucke, M. E. *ECE311 - Dynamic Systems and Control Linearization of Nonlinear Systems.* [cit. 2022-03-18]. Available at: https://www.control.utoronto.ca/~broucke/ece311s/Handouts/linearization.pdf.

[16] Brunton, S. L. and Kutz, J. N. *Data-driven science and engineering: Machine learning, dynamical systems, and control.* Cambridge University Press, 2019. ISBN 1108422098.

[17] Burden, R. L. and Faires, J. D. *Numerical analysis.* Cengage Learning, 2011. ISBN 0-538-73351-9.

[18] Butcher, J. C. *Numerical methods for ordinary differential equations.* John Wiley & Sons, 2016. ISBN 9781119121503.

[19] Cellier, F. E. and Kofman, E. *Continuous system simulation.* Springer Science & Business Media, 2006. ISBN 0-387-26102-8.

[20] Chaloupka, J., Kocina, F., Veigend, P., Nečasová, G., Šátek, V. et al. Multiple Integral Computations. In: *14th International Conference of Numerical Analysis and Applied Mathematics.* American Institute of Physics, 2017, no. 1863. DOI: 10.1063/1.4992650. ISSN 0094-243X. Available at: https://www.fit.vut.cz/research/publication/11226.

[21] Chang, Y. F. and Corliss, G. ATOMF: solving ODEsand DAEs using Taylor series. *Computers Math. Applic.* Elsevier. 1994, vol. 28, p. 209–233.

[22] Cho, D., Kato, Y. and Spilman, D. Sliding mode and classical controllers in magnetic levitation systems. *IEEE Control Systems Magazine.* IEEE. 1993, vol. 13, no. 1, p. 42–48.

[23] De Jalon, J. G. and Bayo, E. *Kinematic and dynamic simulation of multibody systems: the real-time challenge.* Springer Science & Business Media, 2012. ISBN 978-1-4612-7601-2.

[24] Dorf, R. C. and Bishop, R. H. *Modern control systems.* Pearson, 2017. ISBN 1-292-15297-4.

[25] Dormand, J. R. and Prince, P. J. A family of embedded Runge-Kutta formulae. *Journal of computational and applied mathematics.* Elsevier. 1980, vol. 6, no. 1, p. 19–26.

[26] Duriez, T., Brunton, S. L. and Noack, B. R. *Machine Learning Control-Taming Nonlinear Dynamics and Turbulence*. Springer, 2017. ISBN 978-3319821405.

[27] Enright, W. H. and Pryce, J. D. Two FORTRAN packages for assessing initial value methods. *ACM Transactions on Mathematical Software (TOMS)*. ACM. 1987, vol. 13, no. 1, p. 1–27.

[28] Fajmon Břetislav, H. I. and Michal, N. *Matematika 3*. FEKT BUT, 2013.

[29] Fantoni, I., Lozano, R. and Lozano, R. *Non-linear control for underactuated mechanical systems*. Springer Science & Business Media, 2002. ISBN 1-85233-423-1.

[30] Forni, F. and Sepulchre, R. Differential analysis of nonlinear systems: Revisiting the pendulum example. In: IEEE. *53rd IEEE Conference on Decision and Control*. 2014, p. 3848–3859.

[31] Friedland, B. *Control system design: an introduction to state-space methods*. Cybernetics Technology, 1986. ISBN 978-0-486-44278-5.

[32] Hairer, E., Nørsett, S. P. and Wanner, G. *Solving ordinary differential equations. 1, Nonstiff problems*. 2nd ed. Springer-Verlag, 2008. ISBN 978-3-540-56670-0.

[33] Hanta, V. and Procházka, A. Rational approximation of time delay. *Institute of Chemical Technology in Prague. Department of computing and control engineering. Technická*. 2009, vol. 5, no. 166, p. 28.

[34] Hateley, J. *The Lorenz system* [http://web.math.ucsb.edu/~jhateley/paper/lorenz.pdf].

[35] Hirsch, M. W., Smale, S. and Devaney, R. L. *Differential equations, dynamical systems, and an introduction to chaos*. Academic press, 2004. ISBN 0-12-349703-5.

[36] Isidori, A. *Nonlinear control systems*. 3rd ed. Springer, 1995. ISBN 3-540-19916-0.

[37] Jorba, A. and Zou, M. A software package for the numerical integration of ODE by means of high-order Taylor methods. In: *Exp. Math.* 2005, vol. 14, p. 99–117.

[38] Jose, J. and Mija, S. An output feedback integral optimal sliding mode controller for magnetic levitation systems. In: IEEE. *2020 Fourth International Conference on Inventive Systems and Control (ICISC)*. 2020, p. 197–202.

[39] Khan, M. J., Junaid, M., Bilal, S., Siddiqi, S. J. and Khan, H. A. Modelling, simulation & control of non-linear magnetic levitation system. In: IEEE. *2018 IEEE 21st International Multi-Topic Conference (INMIC)*. 2018, p. 1–5.

[40] Knill, O. *The Lorenz System* [online]. Available at: http://www.math.harvard.edu/archive/118r_spring_05/handouts/lorentz.pdf.

[41] Kocina, F. *FOS*. Available at: https://www.fit.vut.cz/research/product/518/.en.

[42] Kocina, F., Kunovský, J., Nečasová, G., Šátek, V. and Veigend, P. Parallel solution of higher order differential equations. In: *Proceedings of the 2016 International Conference on High Performance Computing & Simulation (HPCS 2016)*. Institute of Electrical and Electronics Engineers, 2016, p. 302–309. DOI: 10.1109/HPCSim.2016.7568350. ISBN 978-1-5090-2088-1. Available at: https://www.fit.vut.cz/research/publication/11116.

[43] Kocina, F., Veigend, P., Nečasová, G. and Kunovský, J. Parallel Computations of Differential Equations. In: *Proceedings of the 10th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*. Ing. Vladislav Pokorný - Litera, 2015, p. 28–35. ISBN 978-80-214-5254-1. Available at: https://www.fit.vut.cz/research/publication/10925.

[44] Kraus, M. *Paralelní výpočetní architektury založené na numerické integraci*. Brno, CZ, 2013. Disertační práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.fit.vut.cz/study/phd-thesis/268/.

[45] Kunovský, J. *Modern Taylor Series Method*. 1994. Habilitation work. Faculty of Electrical Engineering and Computer Science, Brno University of Technology.

[46] Kunovský, J. Real-Time Applications of the Taylor Series. In: *Proceedings of ADIUS 2000, ADI*. 2000, p. 67–75. Available at: https://www.fit.vut.cz/research/publication/6086.

[47] Kunovský, J. *TKSL*. Available at: https://www.fit.vut.cz/research/product/51/.en.

[48] Laplante, P. A. et al. *Real-time systems design and analysis*. Institute of Electrical and Electronics Engineers, 2004. ISBN 0-7803-0400-0.

[49] Ledin, J. A. Hardware-in-the-loop simulation. *Embedded Systems Programming*. MILLER FREEMAN INC. 1999, vol. 12, p. 42–62.

[50] Liesen, J. and Mehrmann, V. *Linear algebra*. Springer, 2015. ISBN 978-3-319-24346-7.

[51] Lorenz, E. N. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*. 1963, vol. 20, no. 2, p. 130–141.

[52] Makino, K. and Berz, M. Cosy infinity version 8. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. Elsevier. 1999, vol. 427, 1-2, p. 338–343.

[53] Matečný, F. *Hardwarová realizace numerického integrátoru s metodou vyššího řádu*. Brno, CZ, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.fit.vut.cz/study/thesis/21205/.

[54] MathWorks. *MATLAB*. 2021. Available at: https://www.mathworks.com/products/matlab.html.

[55] Mfoumboulou, Y. and Tzoneva, R. Development of a nonlinear linearizing controller using Input-Output/Input-State linearization for implementation in Programmable Logic Controller (PLC). *International Journal of Engineering Research and Technology*. International Research Publication House. 2019.

[56] Mohazzabi, P. and Becker, J. L. Numerical Solution of Differential Equations by Direct Taylor Expansion. *Journal of Applied Mathematics and Physics*. Scientific Research Publishing. 2017, vol. 5, no. 3, p. 623–630. ISSN 2327-4352. Available at: http://www.scirp.org/journal/doi.aspx?DOI=10.4236/jamp.2017.53053.

[57] Nedialkov, N. S. and Pryce, J. Solving differential algebraic equations by Taylor series III. The DAETS code. *JNAIAM J. Numer. Anal. Ind. Appl. Math.* Elsevier. 2008, vol. 3, p. 61–80.

[58] Nečasová, G., Veigend, P. and Šátek, V. Modern Taylor series method in numerical integration: PART 2. In: *17th Czech-Polish Conference Modern Mathematical Methods in Engineering (3mi)*. VŠB - Technical University of Ostrava, 2018, p. 211–220. ISBN 978-80-248-4135-9. Available at: https://www.fit.vut.cz/research/publication/11639.

[59] Nečasová, G., Veigend, P. and Šátek, V. Parallel Solution of Partial Differential Equations Using Taylor Series Method. In: *18th International Conference of Numerical Analysis and Applied Mathematics*. American Institute of Physics, 2022, vol. 2425, no. 3, p. 1–4. DOI: 10.1063/5.0082209. ISSN 0094-243X. Available at: https://www.fit.vut.cz/research/publication/12296.

[60] Nečasová, G. and Šátek, V. Parallel Solution of Telegraph Line. In: *16th International Conference of Numerical Analysis and Applied Mathematics*. American Institute of Physics, 2019, vol. 2019, no. 7, p. 1–4. DOI: 10.1063/1.5114333. ISSN 0094-243X. Available at: https://www.fit.vut.cz/research/publication/11751.

[61] Olver, P. J. Nonlinear ordinary differential equations. *Introduction to Partial Differential Equations*. Citeseer. 2006, p. 1081–1142.

[62] Opálka, J. *Automatické řízení výpočtu ve specializovaném výpočetním systému*. Brno, CZ, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.fit.vut.cz/study/thesis/18806/.

[63] Prasad, L. B., Tyagi, B. and Gupta, H. O. Optimal control of nonlinear inverted pendulum system using PID controller and LQR: performance analysis without and with disturbance input. *International Journal of Automation and Computing*. Springer. 2014, vol. 11, no. 6, p. 661–670.

[64] Rábová, Z., Zendulka, J., Češka, M., Peringer, P. and Janoušek, V. *Modelování a simulace*. Brno University of Technology, 1992. ISBN 80-214-0480-9.

[65] Richard, P., Feynman, L., Robert, B. and Sands, M. *Feynman Lectures on Physics: The New Millennium Edition*. Basic Books, 2015.

[66] Rodríguez, M., Medina, A. A., Gil, R. B. and Blesa, F. TIDES: A free software based on the Taylor series method. *Monografías de la Real Academia de Ciencias Exactas, Físicas, Químicas y Naturales de Zaragoza*. Real Academia de Ciencias Exactas, Físicas, Químicas y Naturales de Zaragoza. 2011, no. 35, p. 83–95.

[67] Sekaninová, M. *Propojovací systém paralelních ALU pro numerickou integraci*. Brno, CZ, 2015. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.fit.vut.cz/study/thesis/17829/.

[68] Shameli, E., Khamesee, M. B. and Huissoon, J. P. Nonlinear controller design for a magnetic levitation device. *Microsystem Technologies*. Springer. 2007, vol. 13, no. 8, p. 831–835.

[69] Shampine, L. F. and Reichelt, M. W. The Matlab ODE Suite. *SIAM journal on scientific computing*. SIAM. 1997, vol. 18, no. 1, p. 1–22.

[70] Slotine, J.-J. E. et al. *Applied nonlinear control*. Prentice–Hall, 1991. ISBN 0-13-040890-5.

[71] Smith, G. D. and Smith, G. D. *Numerical solution of partial differential equations: finite difference methods*. Oxford university press, 1985. ISBN 0-19-859650-2.

[72] Stengel, R. F. *Optimal control and estimation*. Courier Corporation, 1994. ISBN 978-0-486-68200-6.

[73] Söderlind, G., Jay, L. and Calvo, M. Stiffness 1952–2012: Sixty years in search of a definition. *Bit Numer Math*. june 2015, vol. 55, p. 531–558. DOI: 10.1007/s10543-014-0503-3.

[74] University of Michigan. *DC Motor Speed: Simulink Modeling*. 2020. Available at: http://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SimulinkModeling.

[75] Veigend, P. *Semi-analytické výpočty určitých integrálů*. Brno, CZ, 2011. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.fit.vut.cz/study/thesis/12730/.

[76] Veigend, P., Nečasová, G., Kocina, F., Chaloupka, J., Šátek, V. et al. Real Time Simulation of Transport Delay. In: *14th International Conference of Numerical Analysis and Applied Mathematics*. American Institute of Physics, 2017, no. 1863, p. 1–4. DOI: 10.1063/1.4992523. ISSN 0094-243X. Available at: https://www.fit.vut.cz/research/publication/11227.

[77] Veigend, P., Nečasová, G. and Šátek, V. High Order Numerical Integration Method and its Applications – The First 36 Years of MTSM. In: *2019 IEEE 15th International Scientific Conference on Informatics*. 2019. Available at: https://www.fit.vut.cz/research/publication/12004.

[78] Veigend, P., Nečasová, G. and Šátek, V. Taylor Series Based Numerical Integration Method. *Open Computer Science*. 2021, vol. 11, no. 1, p. 60–69. DOI: 10.1515/comp-2020-0163. ISSN 2299-1093. Available at: https://www.fit.vut.cz/research/publication/12193.

[79] Veigend, P., Nečasová, G. and Šátek, V. System control using high order numerical method. In: *18th International Conference of Numerical Analysis and Applied Mathematics*. American Institute of Physics, 2022, vol. 2425, no. 3, p. 1–4. DOI: 10.1063/5.0082205. ISSN 0094-243X. Available at: https://www.fit.vut.cz/research/publication/12295.

[80] Veigend, P., Nečasová, G. and Šátek, V. System control using high order numerical method. In: *18th International Conference of Numerical Analysis and*

Applied Mathematics. American Institute of Physics, 2022, vol. 2425, no. 3, p. 1–4. DOI: 10.1063/5.0082205. ISSN 0094-243X. Available at: https://www.fit.vut.cz/research/publication/12295.

[81] Veigend, P., Šátek, V. and Nečasová, G. Model of the Telegraph line and its Numerical Solution. *Open Computer Science*. 2018, vol. 8, no. 1, p. 10–17. DOI: 10.1515/comp-2018-0002. ISSN 2299-1093. Available at: https://www.fit.vut.cz/research/publication/11666.

[82] Virgala, I., Frankovský, P. and Kenderová, M. Friction effect analysis of a DC motor. In: *American Journal of Mechanical Engineering*. 1st ed. 2013. DOI: 10.12691/ajme-1-1-1.

[83] Šátek, V. *Analýza stiff soustav diferenciálních rovnic*. Brno, CZ, 2012. Disertační práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.fit.vut.cz/study/phd-thesis/267/.

[84] Vítečková, M. and Víteček, A. *State space control*. VŠB Ostrava, 2016. ISBN 978-80-248-3979-0.

[85] Wong, T. Design of a magnetic levitation control system??? an undergraduate project. *IEEE Transactions on Education*. IEEE. 1986, no. 4, p. 196–200.

[86] Závada, V. *Simulátor procesoru s operací násobení*. Brno, CZ, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.fit.vut.cz/study/thesis/18534/.

[87] Šátek, V. *High Performance Computing Research Group*. Available at: https://www.fit.vut.cz/research/group/hpc/.en.

Appendices

List of Appendices

Appendix A

Block algebra

In this Appendix, the block algebra used in the thesis is presented. The figures use the notation used in SIMULINK[1].

A.1 Addition

To represent *addition*, the block in Figure A.1 is used.

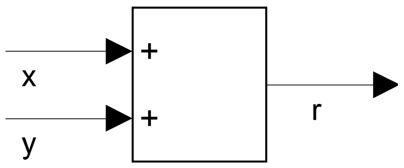


Figure A.1: Adder.

The block represents the operation

$$r = x + y\,.$$

A.2 Multiplication

When representing *multiplication*, generally, there are two cases:

- multiplication by a constant,
- multiplication of two or more functions.

These two cases are going to be discussed separately.

A.2.1 Multiplication by a constant

When multiplying a function value by a constant value, the block in Figure A.2 is used.

[1]https://www.mathworks.com/products/simulink.html

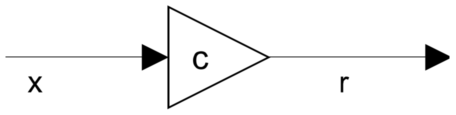


Figure A.2: Multiplication by a constant.

The block represents the operation

$$r = c \cdot x$$

and in the thesis, this is often simply abbreviated as

$$r = cx\,.$$

A.2.2 Multiplication of functions

When representing multiplications of two (or more) functions, the block in Figure A.3 is used.

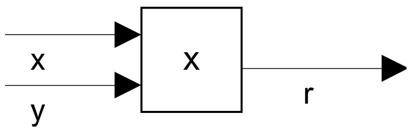


Figure A.3: Multiplication of two functions.

The block represents the operation

$$r = x \cdot y$$

where x and y can be arbitrary functions. In the thesis, this can again be abbreviated as

$$r = xy\,.$$

A.3 Integration

When representing integration, the block in Figure A.4 is used.

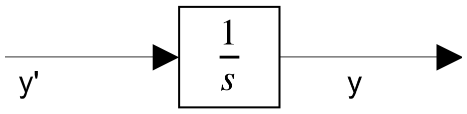


Figure A.4: Integration.

For the purposes of this thesis, the block represents the operation

$$y' = y \qquad y(0) = c$$

where c represents the initial condition of the integrator.

Appendix B

Padé approximations

In this Appendix, the complete results of the Padé approximation experiments for approximation orders $n = 1$, $n = 2$, $n = 7$, $n = 10$ and $n = 15$ are presented. More information about the approximations and the detailed analysis for $n = 5$ is presented in Section 4.2. All results are a median from 100 runs.

B.1 Approximation order 1

For $n = 1$, the error in the last step of the calculation is approximately 1×10^{-2} for all solvers, which does not favour the MTSM (mean order of the method is approximately 9) for all configurations.

Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	9.636×10^{-4}	–	101
ode23	5.5041×10^{-3}	**5.71**	2087
ode45	8.6605×10^{-4}	0.90	473
ode113	7.835×10^{-4}	0.81	71

Table B.1: Results using MDORAV with complete system for $n = 1$.

Solver	Time of calculation [s]	Ratio	Number of steps
mtsm	6.0455×10^{-4}	–	101
ode23	2.98785×10^{-3}	**5.94**	1375
ode45	6.1225×10^{-4}	**1.01**	397
ode113	5.84×10^{-4}	0.96	66

Table B.2: Results using MDORAV with constant input for $n = 1$.

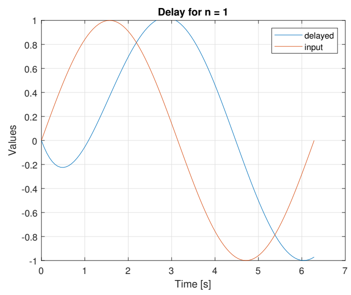
Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	9.7255×10^{-4}	–	101
ode23	$6.317\,95 \times 10^{-3}$	**6.49**	2345
ode45	8.7495×10^{-4}	0.90	485
ode113	8.3125×10^{-4}	0.85	75

Table B.3: Results using MSI for $n = 1$.

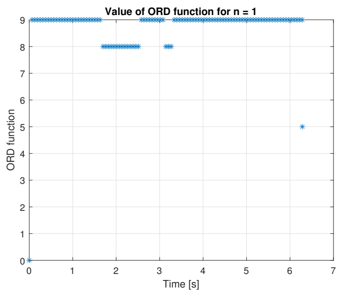
Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	5.722×10^{-4}	–	101
ode23	3.9807×10^{-3}	**6.96**	2080
ode45	7.136×10^{-4}	**1.25**	505
ode113	7.36×10^{-4}	**1.29**	74

Table B.4: Results using MSI with constant input for $n = 1$.

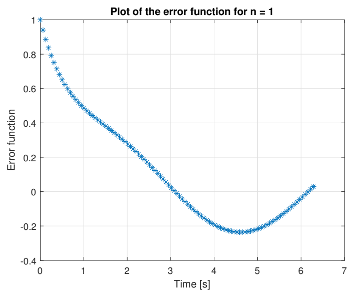
The results are in Figure B.1 for MDORAV and Figure B.2 for MSI.



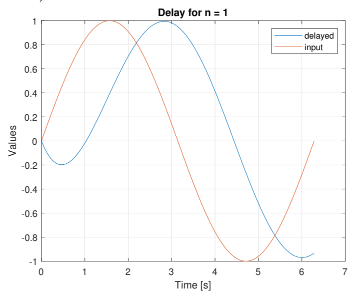
(a) MDORAV (complete system).



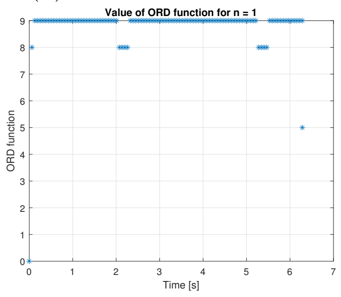
(b) ORD for MDORAV.



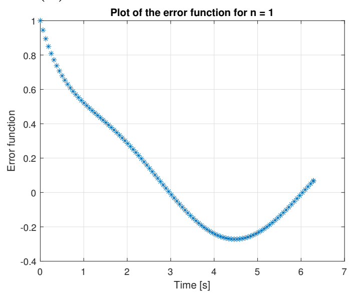
(c) Error for MDORAV.



(d) MDORAV (constant input).

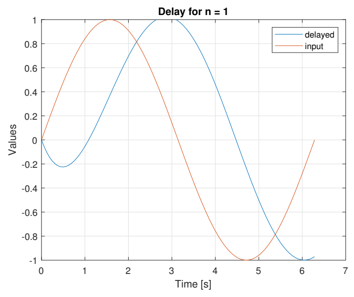


(e) ORD for MDORAV CI.

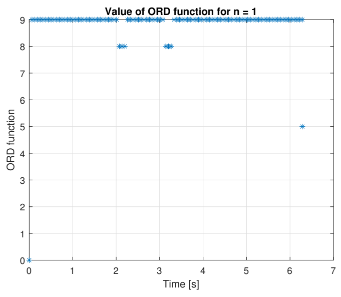


(f) Error for MDORAV CI.

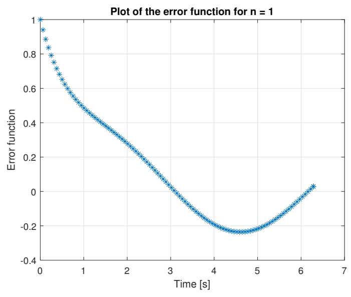
Figure B.1: MDORAV results for $n = 1$ with complete system (first row) and with constant input (second row).



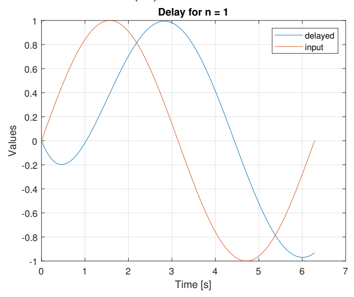
(a) MSI.



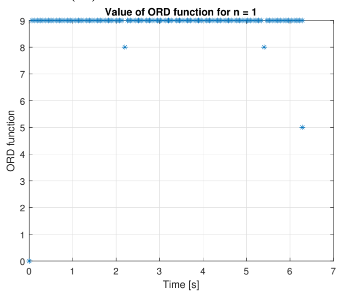
(b) ORD for MSI.



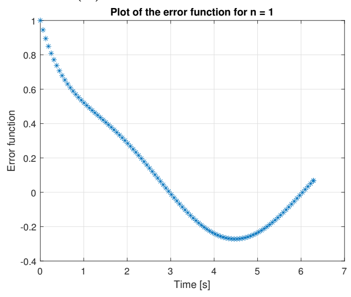
(c) Error for MSI.



(d) MSI (constant input).



(e) ORD for MSI CI.



(f) Error for MSI CI.

Figure B.2: MSI results for $n = 1$ with complete system (first row) and with constant input (second row).

B.2 Approximation order 2

For $n = 2$, the error in the last step of the calculation is approximately 1×10^{-5} for MDORAV (1×10^{-2} for constant input MTSM) and 0.2 for MSI for all state-of-the-art solvers. The mean order of the method is approximately 8.5 for all configurations.

Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	9.732×10^{-4}	–	101
ode23	$5.845\,45 \times 10^{-3}$	**6.01**	2092
ode45	9.2855×10^{-4}	0.95	493
ode113	8.6985×10^{-4}	0.89	78

Table B.5: Results using MDORAV with complete system for $n = 2$.

Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	1.0817×10^{-3}	–	101
ode23	$3.650\,15 \times 10^{-3}$	**3.37**	1020
ode45	1.2356×10^{-3}	**1.14**	557
ode113	$1.038\,25 \times 10^{-3}$	0.96	85

Table B.6: Results using MDORAV with constant input for $n = 2$.

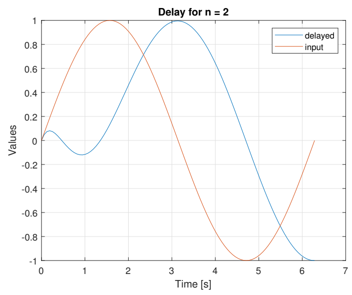
Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	1.3263×10^{-3}	–	101
ode23	9.6718×10^{-3}	**7.29**	2778
ode45	$1.610\,35 \times 10^{-3}$	**1.21**	613
ode113	1.3137×10^{-3}	0.99	89

Table B.7: Results using MSI for $n = 2$.

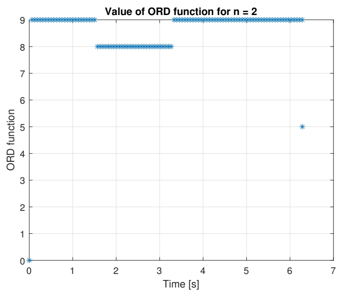
Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	1.0134×10^{-3}	–	101
ode23	9.3557×10^{-3}	**9.23**	3007
ode45	1.8704×10^{-3}	**1.84**	1033
ode113	7.36×10^{-4}	**1.08**	107

Table B.8: Results using MSI with constant input for $n = 2$.

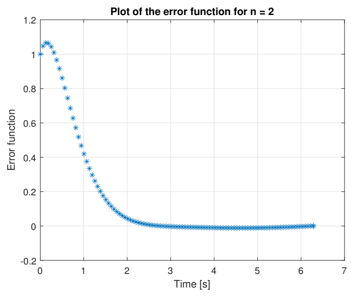
The results are in Figure B.3 for MDORAV and Figure B.4 for MSI.



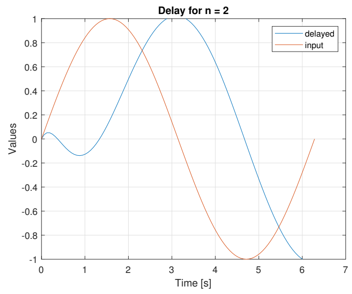
(a) MDORAV (complete system).



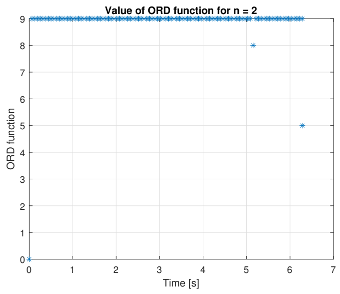
(b) ORD for MDORAV.



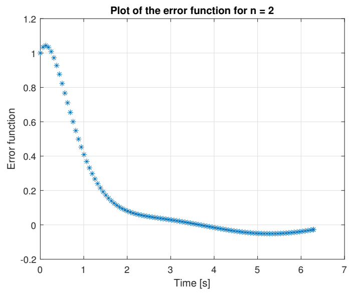
(c) Error for MDORAV.



(d) MDORAV (constant input).

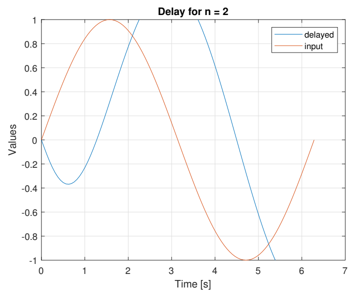


(e) ORD for MDORAV CI.

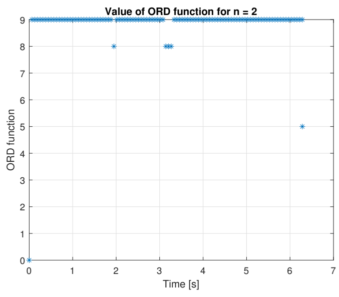


(f) Error for MDORAV CI.

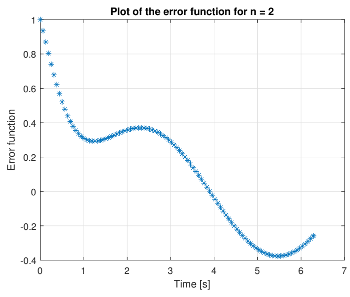
Figure B.3: MDORAV results for $n = 2$ with complete system (first row) and with constant input (second row).



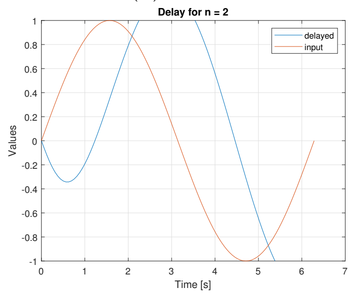
(a) MSI.



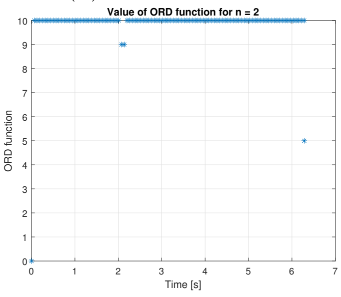
(b) ORD for MSI.



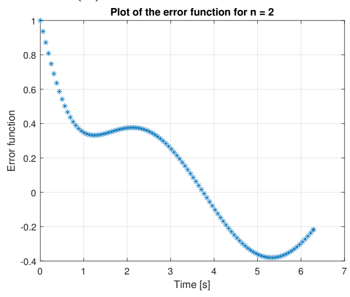
(c) Error for MSI.



(d) MSI (constant input).



(e) ORD for MSI CI.



(f) Error for MSI CI.

Figure B.4: MSI results for $n = 2$ with complete system (first row) and with constant input (second row).

B.3 Approximation order 7

For $n = 7$, the error in last step of the calculation is approximately 1×10^{-8} for MDORAV and MSI and 1×10^{-3} for configurations with constant inputs.

Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	9.7515×10^{-4}	–	101
ode23	5.9051×10^{-3}	**6.05**	2087
ode45	9.644×10^{-4}	0.99	521
ode113	1.3509×10^{-3}	**1.39**	143

Table B.9: Results using MDORAV with complete system for $n = 7$.

Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	1.0891×10^{-3}	–	101
ode23	$1.909\,85 \times 10^{-3}$	**1.75**	490
ode45	1.7602×10^{-3}	**1.61**	865
ode113	1.4258×10^{-3}	**1.31**	145

Table B.10: Results using MDORAV with constant input for $n = 7$.

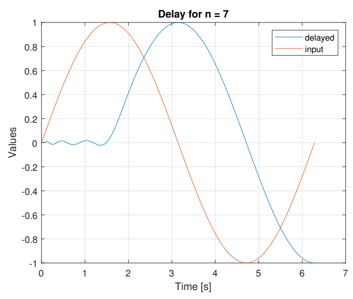
Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	1.2829×10^{-3}	–	101
ode23	$1.842\,25 \times 10^{-2}$	**14.36**	6616
ode45	2.5723×10^{-3}	**2.01**	1541
ode113	3.2247×10^{-3}	**2.51**	260

Table B.11: Results using MSI for $n = 7$.

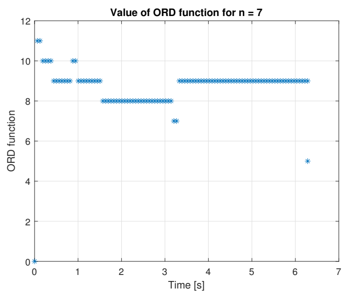
Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	1.5656×10^{-3}	–	101
ode23	2.9708×10^{-2}	**18.98**	6857
ode45	9.9732×10^{-3}	**6.38**	4389
ode113	2.814×10^{-3}	**1.80**	263

Table B.12: Results using MSI with constant input for $n = 7$.

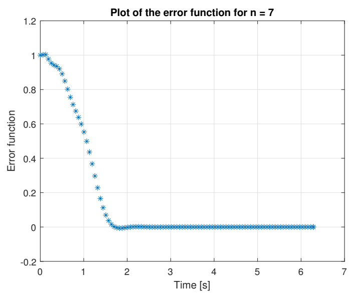
The results are in Figure B.5 for MDORAV and Figure B.6 for MSI.



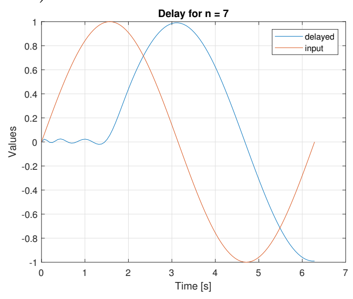
(a) MDORAV (complete system).



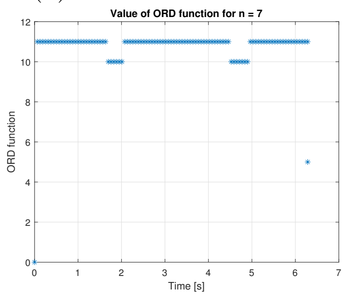
(b) ORD for MDORAV.



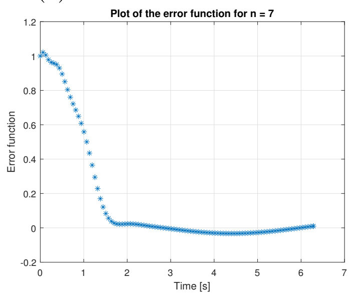
(c) Error for MDORAV.



(d) MDORAV (constant input).

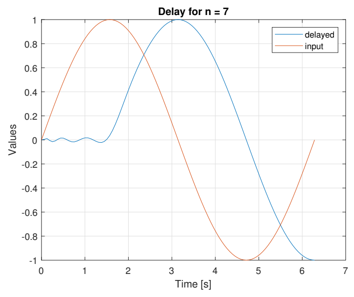


(e) ORD for MDORAV CI.

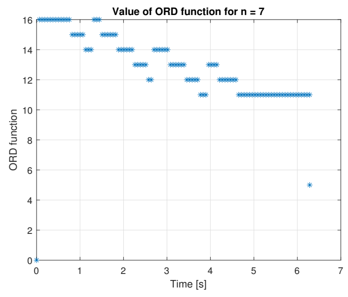


(f) Error for MDORAV CI.

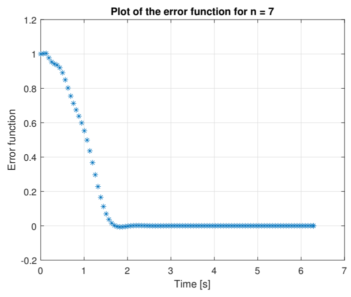
Figure B.5: MDORAV results for $n = 7$ with complete system (first row) and with constant input (second row).



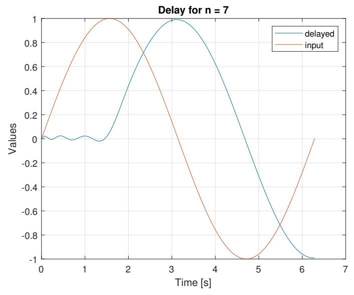
(a) MSI.



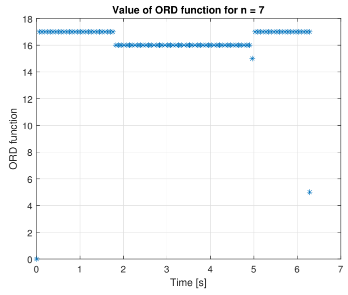
(b) ORD for MSI.



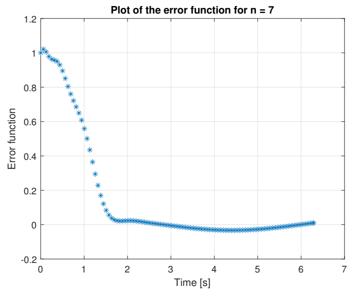
(c) Error for MSI.



(d) MSI (constant input).



(e) ORD for MSI CI.



(f) Error for MSI CI.

Figure B.6: MSI results for $n = 7$ with complete system (first row) and with constant input (second row).

B.4 Approximation order 10

For $n = 10$, the error in last step of the calculation is approximately 1×10^{-8} for complete systems. For constant inputs, the error for MDORAV is 1×10^{-3} and 1×10^{-2} for MSI.

Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	9.672×10^{-4}	–	101
ode23	6.0839×10^{-3}	**6.30**	2087
ode45	9.4695×10^{-4}	0.98	513
ode113	1.6513×10^{-3}	**1.71**	177

Table B.13: Results using MDORAV with complete system for $n = 10$.

Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	$1.173\,15 \times 10^{-3}$	–	101
ode23	1.6178×10^{-3}	**1.38**	406
ode45	1.8356×10^{-3}	**1.56**	877
ode113	1.7684×10^{-3}	**1.51**	188

Table B.14: Results using MDORAV with constant input for $n = 10$.

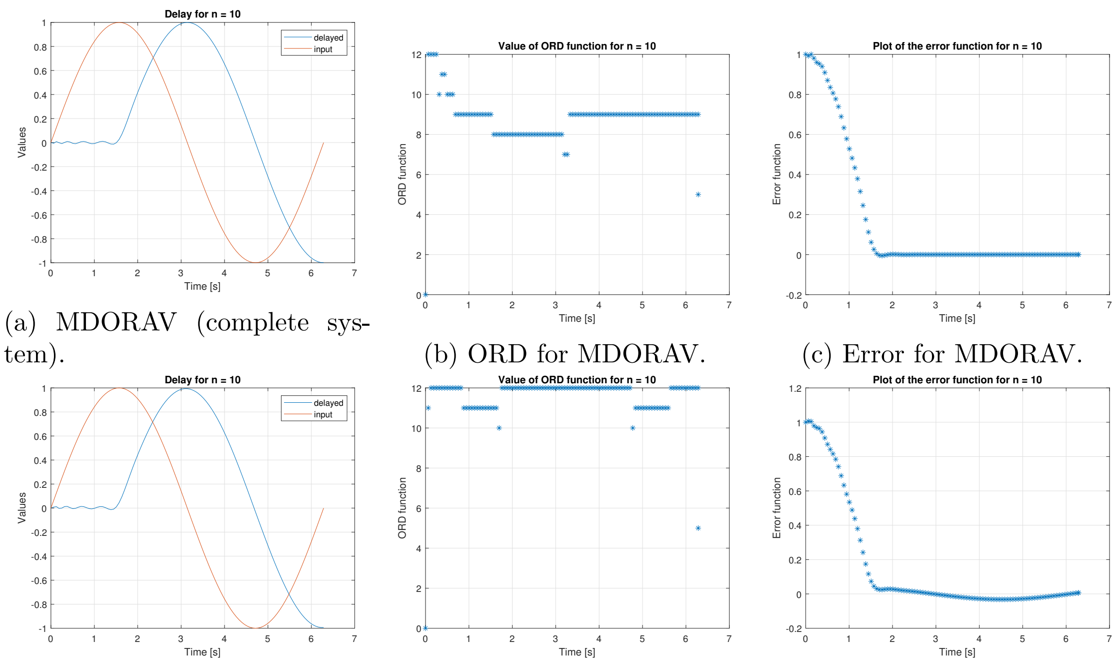
Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	$1.630\,95 \times 10^{-3}$	–	101
ode23	$3.511\,46 \times 10^{-2}$	**21.53**	11980
ode45	3.9805×10^{-3}	**2.44**	2325
ode113	4.7675×10^{-3}	**2.92**	364

Table B.15: Results using MSI for $n = 10$.

Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	2.236×10^{-3}	–	101
ode23	4.3402×10^{-2}	**19.41**	9217
ode45	$1.661\,58 \times 10^{-2}$	**7.43**	6529
ode113	4.022×10^{-3}	**1.80**	376

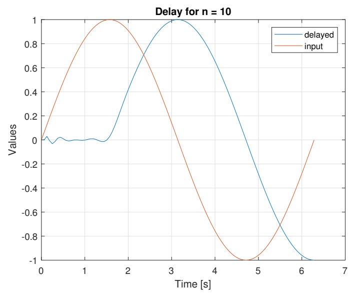
Table B.16: Results using MSI with constant input for $n = 10$.

The results are in Figure B.7 for MDORAV and Figure B.8 for MSI.

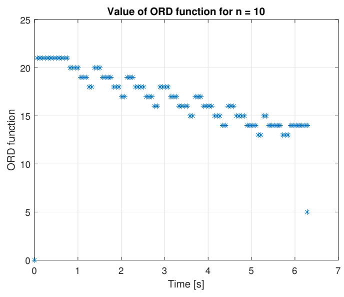


(a) MDORAV (complete system). (b) ORD for MDORAV. (c) Error for MDORAV. (d) MDORAV (constant input). (e) ORD for MDORAV CI. (f) Error for MDORAV CI.

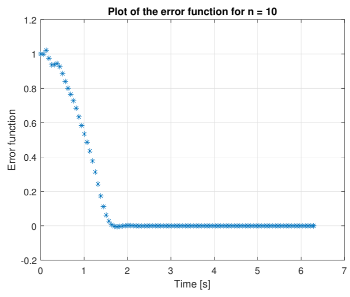
Figure B.7: MDORAV results for $n = 10$ with complete system (first row) and with constant input (second row).



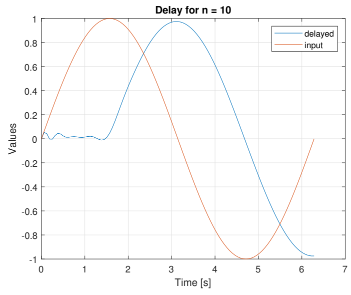
(a) MSI.



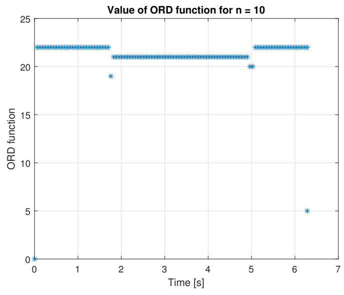
(b) ORD for MSI.



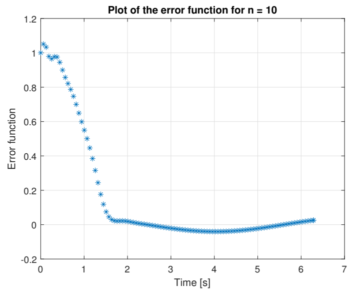
(c) Error for MSI.



(d) MSI (constant input).



(e) ORD for MSI CI.



(f) Error for MSI CI.

Figure B.8: MSI results for $n = 10$ with complete system (first row) and with constant input (second row).

B.5 Approximation order 15

For $n = 15$, the error in last step of the calculation is approximately 1×10^{-8} for complete systems. For constant inputs, the error for MDORAV is 1×10^{-3} and 1×10^{-2} for MSI.

Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	$1.119\,05 \times 10^{-3}$	–	101
ode23	$7.376\,45 \times 10^{-3}$	**6.59**	2087
ode45	$1.017\,05 \times 10^{-3}$	0.91	497
ode113	$2.383\,85 \times 10^{-3}$	**2.13**	214

Table B.17: Results using MDORAV with complete system for $n = 15$.

Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	1.4849×10^{-3}	–	101
ode23	1.4106×10^{-3}	0.95	293
ode45	1.901×10^{-3}	**1.28**	809
ode113	2.3736×10^{-3}	**1.60**	183

Table B.18: Results using MDORAV with constant input for $n = 15$.

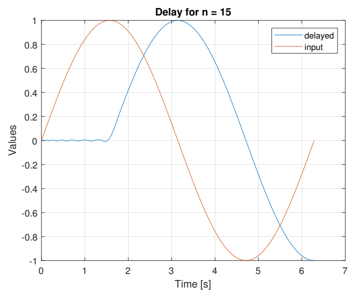
Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	$2.524\,35 \times 10^{-3}$	–	101
ode23	$4.647\,29 \times 10^{-2}$	**18.41**	13141
ode45	4.658×10^{-3}	**1.85**	2441
ode113	6.3988×10^{-3}	**2.53**	489

Table B.19: Results using MSI for $n = 15$.

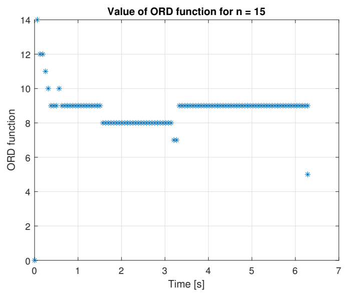
Solver	Time of calculation [s]	Ratio	Number of steps
MTSM	3.0888×10^{-3}	–	101
ode23	$9.652\,62 \times 10^{-2}$	**31.25**	14961
ode45	$3.481\,65 \times 10^{-2}$	**11.27**	10377
ode113	6.8476×10^{-3}	**2.22**	561

Table B.20: Results using MSI with constant input for $n = 15$.

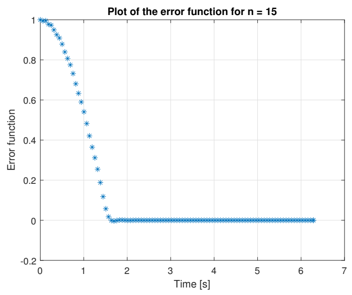
The results are in Figure B.9 for MDORAV and Figure B.10 for MSI.



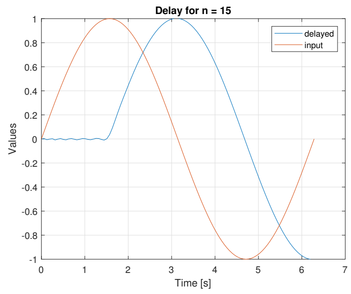
(a) MDORAV (complete system).



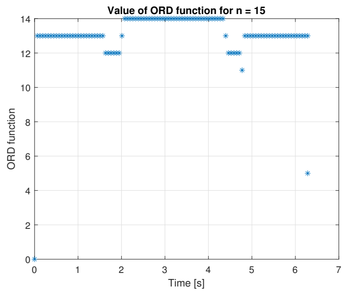
(b) ORD for MDORAV.



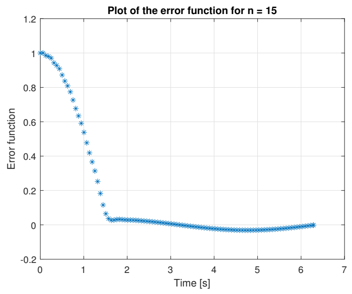
(c) Error for MDORAV.



(d) MDORAV (constant input).

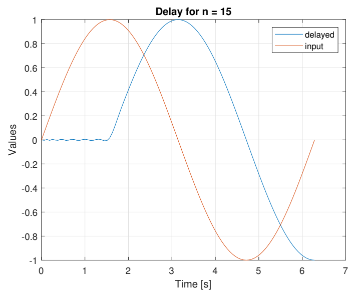


(e) ORD for MDORAV CI.

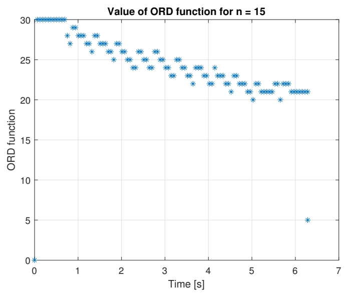


(f) Error for MDORAV CI.

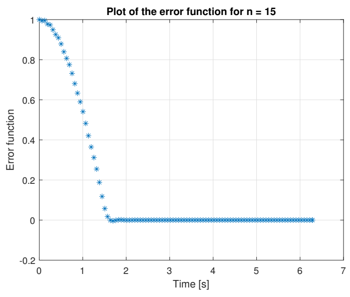
Figure B.9: MDORAV results for $n = 15$ with complete system (first row) and with constant input (second row).



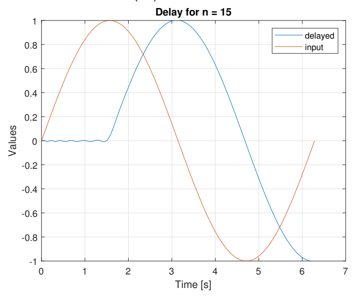
(a) MSI.



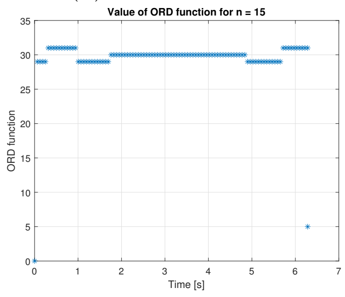
(b) ORD for MSI.



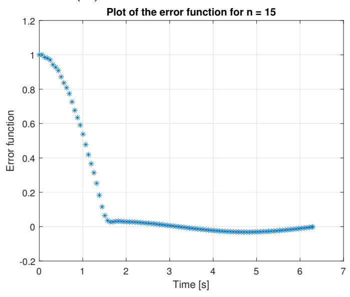
(c) Error for MSI.



(d) MSI (constant input).



(e) ORD for MSI CI.



(f) Error for MSI CI.

Figure B.10: MSI results for $n = 15$ with complete system (first row) and with constant input (second row).

B.6 Discussion, concluding remarks

The results of the experiments with the different approximation orders and the ODE system derivation methods show some interesting observations. First, it is not surprising that when using the constant input (which means that the delay component would receive just a constant input signal), the accuracy of the method is much lower. This behaviour is due to the fact that the constant vector $\boldsymbol{b}$ is used in just the first term of the Taylor series and nowhere else.

Another interesting aspect, and one worth pointing out, is how the accuracy while using MTSM is otherwise very high even though the method is always much faster than state-of-the-art (which is a pattern similar to other experiments presented in this thesis).

Furthermore, the better performance of the MSI derived systems has to be pointed out. These systems are simpler and contain fewer terms so they can be constructed in hardware using a smaller number of components.

Appendix C

Derivation of the four and five function multiplications

In this Appendix, the higher order derivatives for four and five-function multiplications are derived in a similar way as in the Subsection 3.3.2.

C.1 Four function multiplication and higher derivatives

For four-function multiplication, the elementary IVP can be written as

$$y' = qrsu \quad y(0) = y_0\,, \tag{C.1}$$

where q, r, s and u are arbitrary functions. To numerically solve the IVP using the MTSM, the Taylor series for all functions have to be constructed

$$\begin{aligned}
y_{i+1} &= y_i + DY(1)_i + DY(2)_i + DY(3)_i + DY(4)_i + \cdots + DY(N)_i \\
q_{i+1} &= q_i + DQ(1)_i + DQ(2)_i + DQ(3)_i + DQ(4)_i + \cdots + DQ(N)_i \\
r_{i+1} &= r_i + DR(1)_i + DR(2)_i + DR(3)_i + DR(4)_i + \cdots + DR(N)_i \\
s_{i+1} &= s_i + DS(1)_i + DS(2)_i + DS(3)_i + DS(4)_i + \cdots + DS(N)_i \\
u_{i+1} &= u_i + DU(1)_i + DU(2)_i + DU(3)_i + DU(4)_i + \cdots + DU(N)_i .
\end{aligned} \tag{C.2}$$

The higher derivatives for the (C.1) can be constructed:

$$
\begin{aligned}
y' &= qrsu \\
y'' &= q'rsu + qr'su + qrs'u + qrsu' \\
y''' &= q''rsu + q'r'su + q'rs'u + q'rsu' + q'r'su + qr''su + \\
&+ qr's'u + qr'su' + q'rs'u + qr's'u + qrs''u + qrs'u' + q'rsu' \\
&+ qr'su' + qrs'u' + qrsu'' = q''rsu + qr''su + qrs''u + qrsu'' + \\
&+ 2q'r'su + 2q'rs'u + 2qr's'u + 2qr'su' + 2qrs'u' + 2q'rsu' \\
y^{[4]} &= q'''rsu + q''r'su + q''rs'u + q''rsu' + \\
&+ q'r''su + qr'''su + qr''s'u + qr''su' + \\
&+ q'rs''u + qr's''u + qrs'''u + qrs''u' + \\
&+ q'rsu'' + qr'su'' + qrs'u'' + qrsu''' + \\
&+ 2q''r'su + 2q'r''su + 2q'r's'u + 2q'r'su' + \\
&+ 2q''rs'u + 2q'r's'u + 2q'rs''u + 2q'rs'u' + \\
&+ 2q'r's'u + 2qr''s'u + 2qr's''u + 2qr's'u' + \\
&+ 2q'r'su' + 2qr''su' + 2qr's'u' + 2qr'su'' + \\
&+ 2q'rs'u' + 2qr's'u' + 2qrs''u' + 2qrs'u'' + \\
&+ 2q''rsu' + 2q'r'su' + 2q'rs'u' + 2q'rsu'' = \\
&= q'''rsu + qr'''su + qrs'''u + qrsu''' + 3q''rs'u + \\
&+ 3q''rsu' + 3q'r''su + 3qr''s'u + 3qr''su' + 3q'rs''u + \\
&+ 3qr's''u + 3qrs''u' + 3q'rsu'' + 3qr'su'' + 3qrs'u'' + \\
&+ 3q''r'su + 6q'r's'u + 6q'r'su' + 6q'rs'u' + 6qr's'u' \\
&\vdots
\end{aligned}
\tag{C.3}
$$

From (C.3) and (3.1), equations for the Taylor series terms can be expressed

$$
\begin{aligned}
\frac{DY(1)_i}{h} &= DQ(0)_iDR(0)_iDS(0)_iDU(0)_i \\
\frac{DY(2)_i}{\frac{h^2}{2!}} &= \frac{DQ(1)_i}{h}DR(0)_iDS(0)_iDU(0)_i + DQ(0)_i\frac{DR(1)_i}{h}DS(0)_iDU(0)_i \\
&+ DQ(0)_iDR(0)_i\frac{DS(1)_i}{h}DU(0)_i + DQ(0)_iDR(0)_iDS(0)_i\frac{DU(1)_i}{h} \\
\frac{DY(3)_i}{\frac{h^3}{3!}} &= \frac{DQ(2)_i}{\frac{h^2}{2!}}DR(0)_iDS(0)_iDU(0)_i + DQ(0)_i\frac{DR(2)_i}{\frac{h^2}{2!}}DS(0)_iDU(0)_i+ \\
&+ DQ(0)_iDR(0)_i\frac{DS(2)_i}{\frac{h^2}{2!}}DU(0)_i + DQ(0)_iDR(0)_iDS(0)_i\frac{DU(2)_i}{\frac{h^2}{2!}} \\
&+ 2\frac{DQ(1)_i}{h}\frac{DR(1)_i}{h}DS(0)_iDU(0)_i + 2\frac{DQ(1)_i}{h}DR(0)_i\frac{DS(1)_i}{h}DU(0)_i+ \\
&+ 2DQ(0)_i\frac{DR(1)_i}{h}\frac{DS(1)_i}{h}DU(0)_i + 2DQ(0)_i\frac{DR(1)_i}{h}DS(0)_i\frac{DU(1)_i}{h}+ \\
&+ 2DQ(0)_iDR(0)_i\frac{DS(1)_i}{h}\frac{DU(1)_i}{h} + 2\frac{DQ(1)_i}{h}DR(0)_iDS(0)_i\frac{DU(1)_i}{h} \\
\frac{DY(4)_i}{\frac{h^4}{4!}} &= \frac{DQ(3)_i}{\frac{h^3}{3!}}DR(0)_iDS(0)_iDU(0)_i + DQ(0)_i\frac{DR(3)_i}{\frac{h^3}{3!}}DS(0)_iDU(0)_i+ \\
&+ DQ(0)_iDR(0)_i\frac{DS(3)_i}{\frac{h^3}{3!}}DU(0)_i + DQ(0)_iDR(0)_iDS(0)_i\frac{DU(3)_i}{\frac{h^3}{3!}} \\
&+ 3\frac{DQ(2)_i}{\frac{h^2}{2!}}DR(0)_i\frac{DS(1)_i}{h}DU(0)_i + 3\frac{DQ(2)_i}{\frac{h^2}{2!}}DR(0)_iDS(0)_i\frac{DU(1)_i}{h}+ \\
&+ 3\frac{DQ(1)_i}{h}\frac{DR(2)_i}{\frac{h^2}{2!}}DS(0)_iDU(0)_i + 3DQ(0)_i\frac{DR(2)_i}{\frac{h^2}{2!}}\frac{DS(1)_i}{h}DU(0)_i+ \\
&+ 3DQ(0)_i\frac{DR(2)_i}{\frac{h^2}{2!}}DS(0)_i\frac{DU(1)_i}{h} + 3\frac{DQ(1)_i}{h}DR(0)_i\frac{DS(2)_i}{\frac{h^2}{2!}}DU(0)_i+ \\
&+ 3DQ(0)_i\frac{DR(1)_i}{h}\frac{DS(2)_i}{\frac{h^2}{2!}}DU(0)_i + 3DQ(0)_iDR(0)_i\frac{DS(2)_i}{\frac{h^2}{2!}}\frac{DU(1)_i}{h}+ \\
&+ 3\frac{DQ(1)_i}{h}DR(0)_iDS(0)_i\frac{DU(2)_i}{\frac{h^2}{2!}} + 3DQ(0)_i\frac{DR(1)_i}{h}DS(0)_i\frac{DU(2)_i}{\frac{h^2}{2!}}+ \\
&+ 3DQ(0)_iDR(0)_i\frac{DS(1)_i}{h}\frac{DU(2)_i}{\frac{h^2}{2!}} + 3\frac{DQ(2)_i}{\frac{h^2}{2!}}\frac{DR(1)_i}{h}DS(0)_iDU(0)_i \\
&+ 6\frac{DQ(1)_i}{h}\frac{DR(1)_i}{h}\frac{DS(1)_i}{h}DU(0)_i + 6\frac{DQ(1)_i}{h}\frac{DR(1)_i}{h}DS(0)_i\frac{DU(1)_i}{h}+ \\
&+ 6\frac{DQ(1)_i}{h}DR(0)_i\frac{DS(1)_i}{h}\frac{DU(1)_i}{h} + 6DQ(0)_i\frac{DR(1)_i}{h}\frac{DS(1)_i}{h}\frac{DU(1)_i}{h} \\
&\vdots
\end{aligned}
\tag{C.4}
$$

Simplifying the (C.4), the equations for the individual Taylor series terms for four function multiplications can finally be derived:

$$\begin{aligned}
DY(1)_i &= hDQ(0)_i DR(0)_i DS(0)_i DU(0)_i \\
DY(2)_i &= \frac{h}{2}(DQ(1)_i DR(0)_i DS(0)_i DU(0)_i + DQ(0)_i DR(1)_i DS(0)_i DU(0)_i + \\
&\quad + DQ(0)_i DR(0)_i DS(1)_i DU(0)_i + DQ(0)_i DR(0)_i DS(0)_i DU(1)_i) \\
DY(3)_i &= \frac{h}{3}(DQ(2)_i DR(0)_i DS(0)_i DU(0)_i + DQ(0)_i DR(2)_i DS(0)_i DU(0)_i + \\
&\quad + DQ(0)_i DR(0)_i DS(2)_i DU(0)_i + DQ(0)_i DR(0)_i DS(0)_i DU(2)_i \\
&\quad + DQ(1)_i DR(1)_i DS(0)_i DU(0)_i + DQ(1)_i DR(0)_i DS(1)_i DU(0)_i + \\
&\quad + DQ(0)_i DR(1)_i DS(1)_i DU(0)_i + DQ(0)_i DR(1)_i DS(0)_i DU(1)_i + \\
&\quad + DQ(0)_i DR(0)_i DS(1)_i DU(1)_i + DQ(1)_i DR(0)_i DS(0)_i DU(1)_i) \\
DY(4)_i &= \frac{h}{4}(DQ(3)_i DR(0)_i DS(0)_i DU(0)_i + DQ(0)_i DR(3)_i DS(0)_i DU(0)_i + \\
&\quad + DQ(0)_i DR(0)_i DS(3)_i DU(0)_i + DQ(0)_i DR(0)_i DS(0)_i DU(3)_i + \\
&\quad + DQ(2)_i DR(0)_i DS(1)_i DU(0)_i + DQ(2)_i DR(0)_i DS(0)_i DU(1)_i + \\
&\quad + DQ(1)_i DR(2)_i DS(0)_i DU(0)_i + DQ(0)_i DR(2)_i DS(1)_i DU(0)_i + \\
&\quad + DQ(0)_i DR(2)_i DS(0)_i DU(1)_i + DQ(1)_i DR(0)_i DS(2)_i DU(0)_i + \\
&\quad + DQ(0)_i DR(1)_i DS(2)_i DU(0)_i + DQ(0)_i DR(0)_i DS(2)_i DU(1)_i + \\
&\quad + DQ(1)_i DR(0)_i DS(0)_i DU(2)_i + DQ(0)_i DR(1)_i DS(0)_i DU(2)_i + \\
&\quad + DQ(0)_i DR(0)_i DS(1)_i DU(2)_i + DQ(2)_i DR(1)_i DS(0)_i DU(0)_i + \\
&\quad + DQ(1)_i DR(1)_i DS(1)_i DU(0)_i + DQ(1)_i DR(1)_i DS(0)_i DU(1)_i + \\
&\quad + DQ(1)_i DR(0)_i DS(1)_i DU(1)_i + DQ(0)_i DR(1)_i DS(1)_i DU(1)_i) \\
&\ \vdots
\end{aligned} \tag{C.5}$$

Generally, the higher derivatives for four function multiplication can be calculated using the following formula

$$DY(n)_i = \frac{h}{n} \sum_{a=0}^{n-1} DQ(a)_i \sum_{b=0}^{n-a-1} DR(b)_i \sum_{c=1}^{n-a-b} DS(n-a-b-c)_i DU(c-1)_i \,, \tag{C.6}$$

where $n = 1, \ldots, N$.

C.2 Five function multiplication and higher derivatives

The elementary IVP for the five-function multiplication can be written as

$$y' = qrsuv \quad y(0) = y_0 \,, \tag{C.7}$$

where q, r, s, u and v are arbitrary functions. To numerically solve the IVP using MTSM, the Taylor series for all functions have to be constructed

$$\begin{aligned}
y_{i+1} &= y_i + DY(1)_i + DY(2)_i + DY(3)_i + DY(4)_i + \cdots + DY(N)_i \\
q_{i+1} &= q_i + DQ(1)_i + DQ(2)_i + DQ(3)_i + DQ(4)_i + \cdots + DQ(N)_i \\
r_{i+1} &= r_i + DR(1)_i + DR(2)_i + DR(3)_i + DR(4)_i + \cdots + DR(N)_i \\
s_{i+1} &= s_i + DS(1)_i + DS(2)_i + DS(3)_i + DS(4)_i + \cdots + DS(N)_i \\
u_{i+1} &= u_i + DU(1)_i + DU(2)_i + DU(3)_i + DU(4)_i + \cdots + DU(N)_i \\
v_{i+1} &= v_i + DV(1)_i + DV(2)_i + DV(3)_i + DV(4)_i + \cdots + DV(N)_i \,.
\end{aligned} \tag{C.8}$$

The higher derivatives for the (C.7) can be constructed:

$$
\begin{aligned}
y' &= qrsuv \\
y'' &= q'rsuv + qr'suv + qrs'uv + qrsu'v + qrsuv' \\
y''' &= q''rsuv + q'r'suv + q'rs'uv + q'rsu'v + q'rsuv'+ \\
&+ q'r'suv + qr''suv + qr's'uv + qr'su'v + qr'suv'+ \\
&+ q'rs'uv + qr's'uv + qrs''uv + qrs'u'v + qrs'uv'+ \\
&+ q'rsu'v + qr'su'v + qrs'u'v + qrsu''v + qrsu'v'+ \\
&+ q'rsuv' + qr'suv' + qrs'uv' + qrsu'v' + qrsuv'' = \\
&= q''rsuv + qr''suv + qrs''uv + qrsu''v + qrsuv''+ \\
&+ 2q'r'suv + 2q'rs'uv + 2q'rsu'v + 2q'rsuv' + 2qr's'uv+ \\
&+ 2qr'su'v + 2qr'suv' + 2qrs'u'v + 2qrsu'v' + 2qrs'uv' \\
y^{[4]} &= q'''rsuv + q''r'suv + q''rs'uv + q''rsu'v + q''rsuv' \\
&+ q'r''suv + qr'''suv + qr''s'uv + qr''su'v + qr''suv' \\
&+ q'rs''uv + qr's''uv + qrs'''uv + qrs''u'v + qrs''uv' \\
&+ q'rsu''v + qr'su''v + qrs'u''v + qrsu'''v + qrsu''v' \\
&+ q'rsuv'' + qr'suv'' + qrs'uv'' + qrsu'v'' + qrsuv''' \\
&+ 2q''r'suv + 2q'r''suv + 2q'r's'uv + 2q'r'su'v + 2q'r'suv'+ \\
&+ 2q''rs'uv + 2q'r's'uv + 2q'rs''uv + 2q'rs'u'v + 2q'rs'uv'+ \\
&+ 2q''rsu'v + 2q'r'su'v + 2q'rs'u'v + 2q'rsu''v + 2q'rsu'v'+ \\
&+ 2q''rsuv' + 2q'r'suv' + 2q'rs'uv' + 2q'rsu'v' + 2q'rsuv''+ \\
&+ 2q'r's'uv + 2qr''s'uv + 2qr's''uv + 2qr's'u'v + 2qr's'uv'+ \\
&+ 2q'r'su'v + 2qr''su'v + 2qr's'u'v + 2qr'su''v + 2qr'su'v'+ \\
&+ 2q'r'suv' + 2qr''suv' + 2qr's'uv' + 2qr'su'v' + 2qr'suv''+ \\
&+ 2q'rs'u'v + 2qr's'u'v + 2qrs''u'v + 2qrs'u''v + 2qrs'u'v'+ \\
&+ 2q'rsu'v' + 2qr'su'v' + 2qrs'u'v' + 2qrsu''v' + 2qrsu'v''+ \\
&+ 2q'rs'uv' + 2qr's'uv' + 2qrs''uv' + 2qrs'u'v' + 2qrs'uv'' = \\
&= q'''rsuv + qr'''suv + qrs'''uv + qrsu'''v + qrsuv'''+ \\
&+ 3q''r'suv + 3q''rs'uv + 3q''rsu'v + 3q''rsuv' + 3q'r''suv+ \\
&+ 3qr''s'uv + 3qr''su'v + 3qr''suv' + 3q'rs''uv + 3qr's''uv+ \\
&+ 3qrs''u'v + 3qrs''uv' + 3q'rsu''v + 3qr'su''v + 3qrs'u''v+ \\
&+ 3qrsu''v' + 3q'rsuv'' + 3qr'suv'' + 3qrs'uv'' + 3qrsu'v''+ \\
&+ 6q'r's'uv + 6q'r'su'v + 6q'r'suv' + 6q'rs'u'v + 6q'rs'uv'+ \\
&+ 6q'rsu'v' + 6qr's'u'v + 6qr's'uv' + 6qr'su'v' + 6qrs'u'v'+ \\
&\vdots
\end{aligned}
\tag{C.9}
$$

After expressing the Taylor series terms from (C.9) and (3.1) and simplifying the result, the equations for the Taylor series terms for five function multiplications have the following form:

$$
\begin{aligned}
DY(1)_i &= hDQ(0)_iDR(0)_iDS(0)_iDU(0)_iDV(0)_i\\
DY(2)_i &= \frac{h}{2}(DQ(1)_iDR(0)_iDS(0)_iDU(0)_iDV(0)_i + DQ(0)_iDR(1)_iDS(0)_iDU(0)_iDV(0)_i +\\
&\quad + DQ(0)_iDR(0)_iDS(1)_iDU(0)_iDV(0)_i + DQ(0)_iDR(0)_iDS(0)_iDU(1)_iDV(0)_i +\\
&\quad + DQ(0)_iDR(0)_iDS(0)_iDU(0)_iDV(1)_i)\\
DY(3)_i &= \frac{h}{3}(DQ(2)_iDR(0)_iDS(0)_iDU(0)_iDV(0)_i + DQ(0)_iDR(2)_iDS(0)_iDU(0)_iDV(0)_i +\\
&\quad + DQ(0)_iDR(0)_iDS(3)_iDU(0)_iDV(0)_i + DQ(0)_iDR(0)_iDS(0)_iDU(2)_iDV(0)_i +\\
&\quad + DQ(0)_iDR(0)_iDS(0)_iDU(0)_iDV(2)_i + DQ(1)_iDR(1)_iDS(0)_iDU(0)_iDV(0)_i +\\
&\quad + DQ(1)_iDR(0)_iDS(1)_iDU(0)_iDV(0)_i + DQ(1)_iDR(0)_iDS(0)_iDU(1)_iDV(0)_i +\\
&\quad + DQ(1)_iDR(0)_iDS(0)_iDU(0)_iDV(1)_i + DQ(0)_iDR(1)_iDS(1)_iDU(0)_iDV(0)_i +\\
&\quad + DQ(0)_iDR(1)_iDS(0)_iDU(1)_iDV(0)_i + DQ(0)_iDR(1)_iDS(0)_iDU(0)_iDV(1)_i +\\
&\quad + DQ(0)_iDR(0)_iDS(1)_iDU(1)_iDV(0)_i + DQ(0)_iDR(0)_iDS(0)_iDU(1)_iDV(1)_i +\\
&\quad + DQ(0)_iDR(0)_iDS(1)_iDU(0)_iDV(1)_i)\\
DY(4)_i &= \frac{h}{4}(DQ(3)_iDR(0)_iDS(0)_iDU(0)_iDV(0)_i + DQ(0)_iDR(3)_iDS(0)_iDU(0)_iDV(0)_i\\
&\quad DQ(0)_iDR(0)_iDS(3)_iDU(0)_iDV(0)_i + DQ(0)_iDR(0)_iDS(0)_iDU(3)_iDV(0)_i\\
&\quad DQ(2)_iDR(0)_iDS(0)_iDU(0)_iDV(3)_i + DQ(2)_iDR(1)_iDS(0)_iDU(0)_iDV(0)_i\\
&\quad DQ(2)_iDR(0)_iDS(1)_iDU(0)_iDV(0)_i + DQ(2)_iDR(0)_iDS(0)_iDU(1)_iDV(0)_i\\
&\quad DQ(2)_iDR(0)_iDS(0)_iDU(0)_iDV(1)_i + DQ(1)_iDR(2)_iDS(0)_iDU(0)_iDV(0)_i\\
&\quad DQ(0)_iDR(2)_iDS(1)_iDU(0)_iDV(0)_i + DQ(0)_iDR(2)_iDS(0)_iDU(1)_iDV(0)_i\\
&\quad DQ(0)_iDR(2)_iDS(0)_iDU(0)_iDV(1)_i + DQ(1)_iDR(0)_iDS(2)_iDU(0)_iDV(0)_i\\
&\quad DQ(0)_iDR(1)_iDS(2)_iDU(0)_iDV(0)_i + DQ(0)_iDR(0)_iDS(2)_iDU(1)_iDV(0)_i\\
&\quad DQ(0)_iDR(0)_iDS(2)_iDU(0)_iDV(1)_i + DQ(1)_iDR(0)_iDS(0)_iDU(2)_iDV(0)_i\\
&\quad DQ(0)_iDR(1)_iDS(0)_iDU(2)_iDV(0)_i + DQ(0)_iDR(0)_iDS(1)_iDU(2)_iDV(0)_i\\
&\quad DQ(0)_iDR(0)_iDS(0)_iDU(2)_iDV(1)_i + DQ(1)_iDR(0)_iDS(0)_iDU(0)_iDV(2)_i\\
&\quad DQ(0)_iDR(1)_iDS(0)_iDU(0)_iDV(2)_i + DQ(0)_iDR(0)_iDS(1)_iDU(0)_iDV(2)_i\\
&\quad DQ(0)_iDR(0)_iDS(0)_iDU(1)_iDV(2)_i + DQ(1)_iDR(1)_iDS(1)_iDU(0)_iDV(0)_i\\
&\quad DQ(1)_iDR(1)_iDS(0)_iDU(1)_iDV(0)_i + DQ(1)_iDR(1)_iDS(0)_iDU(0)_iDV(1)_i\\
&\quad DQ(1)_iDR(0)_iDS(1)_iDU(1)_iDV(0)_i + DQ(1)_iDR(0)_iDS(1)_iDU(0)_iDV(1)_i\\
&\quad DQ(1)_iDR(0)_iDS(0)_iDU(1)_iDV(1)_i + DQ(0)_iDR(1)_iDS(1)_iDU(1)_iDV(0)_i\\
&\quad DQ(0)_iDR(1)_iDS(1)_iDU(0)_iDV(1)_i + DQ(0)_iDR(1)_iDS(0)_iDU(1)_iDV(1)_i\\
&\quad DQ(0)_iDR(0)_iDS(1)_iDU(1)_iDV(1)_i)\\
&\quad \vdots
\end{aligned}
\tag{C.10}
$$

Generally, the higher derivatives for five-function multiplication can be calculated using the following formula

$$DY(n)_i = \frac{h}{n}\sum_{a=0}^{n-1} DQ(a)_i \sum_{b=0}^{n-a-1} DR(b)_i \sum_{c=0}^{n-a-b-1} DS(c)_i \sum_{d=1}^{n-a-b-c} DU(n-a-b-c-d)_i DV(d-1)_i\,, \tag{C.11}$$

where $n = 1, \ldots, N$. Note that more than five multiplications would be derived similarly.

Appendix D

Optimization of the non-linear Modern Taylor Series Method solver for more multiplications

This Appendix shows the optimizations of the MTSM for non-linear problems when using four and five-function multiplications. More information is in Subsection 3.3.2.

D.1 Four function multiplications

Four function multiplications can be seen Table D.1.

n	Indexes of Taylor series terms																			
1	0																			
	0																			
	0																			
	0																			
2	1	0	0	0																
	0	1	0	0																
	0	0	1	0																
	0	0	0	1																
3	2	1	1	1	0	0	0	0	0	0										
	0	1	0	0	0	0	1	2	1	0										
	0	0	1	0	0	1	0	0	1	2										
	0	0	0	1	2	1	1	0	0	0										
4	3	2	2	2	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	2	1	1	0	0	0	0	0	1	2	1	0	3	2	1	0
	0	0	1	0	0	1	0	2	1	0	0	1	0	0	1	2	0	1	2	3
	0	0	0	1	0	0	1	0	1	2	3	2	2	1	1	1	0	0	0	0

Table D.1: Four term multiplications.

It is possible to optimize the calculation even further. This optimization is visualised in Table D.2. Note that there are now two types of matrices that can be precalculated and then used during the calculation:

- results from the previous term (as seen in Tables 3.1, 3.2 and D.1) and
- partial results from the previous step. Notice the brighter areas in Table D.2. The results from multiplying the sub-matrices can again be used and they speed up the calculation of the current step by approximately 30 to 40 percent. Note that the number of calculations reduces more with the higher value of n.

n	Indexes of Taylor series terms																			
1	0																			
	0																			
	0																			
	0																			
2	1	0	0	0																
	0	0	0	1																
	0	0	1	0																
	0	1	0	0																
3	2	1	1	1	0	0	0	0	0	0										
	0	1	0	0	0	0	1	2	1	0										
	0	0	1	0	0	1	0	0	1	2										
	0	0	0	1	2	1	1	0	0	0										
4	3	2	2	2	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	2	1	1	0	0	0	0	0	1	2	1	0	3	2	1	0
	0	0	1	0	0	1	0	2	1	0	0	1	0	0	1	2	0	1	2	3
	0	0	0	1	0	0	1	0	1	2	3	2	2	1	1	1	0	0	0	0

Table D.2: Four term multiplications – partial results.

D.2 Five function multiplications

For five multiplications, the effectiveness is even more pronounced. The operations are again shown in the tabular form in Table D.3.

n	Indexes of Taylor series terms																																		
1	0																																		
	0																																		
	0																																		
	0																																		
	0																																		
2	1	0	0	0	0																														
	0	0	1	0	0																														
	0	0	0	1	0																														
	0	0	0	0	1																														
	0	1	0	0	0																														
3	2	1	1	1	1	0	0	0	0	0	0	0	0	0	0																				
	0	1	0	0	0	0	1	0	0	2	1	1	0	0	0																				
	0	0	1	0	0	0	0	1	0	0	1	0	2	1	0																				
	0	0	0	1	0	0	0	0	1	0	0	1	0	1	2																				
	0	0	0	0	1	2	1	1	1	0	0	0	0	0	0																				
4	3	2	2	2	2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	2	1	1	1	0	0	0	0	0	0	0	1	0	0	2	1	1	0	0	0	3	2	2	1	1	1	0	0	0	0
	0	0	1	0	0	0	1	0	0	2	1	1	0	0	0	0	0	1	0	0	1	0	2	1	0	0	1	0	2	1	0	3	2	1	0
	0	0	0	1	0	0	0	1	0	0	1	0	2	1	0	0	0	0	1	0	0	1	0	1	2	0	0	1	0	1	2	0	1	2	3
	0	0	0	0	1	0	0	0	1	0	0	1	0	1	2	3	2	2	2	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0

Table D.3: Five term multiplications.

It is again possible to optimize the calculation even further. This optimization is visualised in Table D.4. Note that there are now two types of matrices that can be precalculated and then used during the calculation:

- results from the previous term (as seen in Tables 3.1, 3.2, D.1 and D.3) and
- partial results from the previous step. Notice the brighter areas in Table D.4. The results from multiplying the sub-matrices can again be used, and they speed up the calculation of the current step by approximately 30 to 40 percent. Note that the number of calculations reduces more with the higher value of n.

n	Indexes of Taylor series terms																																		
1	0																																		
	0																																		
	0																																		
	0																																		
	0																																		
2	1	0	0	0	0																														
	0	0	1	0	0																														
	0	0	0	1	0																														
	0	0	0	0	1																														
	0	1	0	0	0																														
3	2	1	1	1	1	0	0	0	0	0	0	0	0	0	0																				
	0	1	0	0	0	0	1	0	0	2	1	1	0	0	0																				
	0	0	1	0	0	0	0	1	0	0	1	0	2	1	0																				
	0	0	0	1	0	0	0	0	1	0	0	1	0	1	2																				
	0	0	0	0	1	2	1	1	1	0	0	0	0	0	0																				
4	3	2	2	2	2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	2	1	1	1	0	0	0	0	0	0	0	1	0	0	2	1	1	0	0	0	3	2	2	1	1	1	0	0	0	0
	0	0	1	0	0	0	1	0	0	2	1	1	0	0	0	0	0	1	0	0	1	0	2	1	0	0	1	0	2	1	0	3	2	1	0
	0	0	0	1	0	0	0	1	0	0	1	0	2	1	0	0	0	0	1	0	0	1	0	1	2	0	0	1	0	1	2	0	1	2	3
	0	0	0	0	1	0	0	0	1	0	0	1	0	1	2	3	2	2	2	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0

Table D.4: Five term multiplications – partial results.

Appendix E

Complete results for numerical benchmarks

This Appendix discusses the solutions and derivations of the benchmarks from Section 4.1.

E.1 Problem A1

Problem A1 is defined as

$$y' = -y \qquad y_0 = (1)\,.$$

The problem is linear. To solve it using MTSM, the equation can be directly transformed into matrix-vector representation

$$\boldsymbol{A} = \begin{pmatrix} -1 \end{pmatrix} \qquad\qquad \boldsymbol{b} = \begin{pmatrix} 0 \end{pmatrix}\,.$$

E.2 Problem A2

Problem A2 is defined as

$$y' = \frac{-y^3}{2} = -0.5y^3 \qquad y(0) = 1\,.$$

To solve this non-linear system using MTSM, the y^3 term has to be replaced by a set of auxiliary ODEs

$$\begin{aligned}
&y_1' = -0.5y^3 && y(0) = 1\\
&y_2 = y_1^3 && \\
&y_2' = 3y_1^2y_1' = 3y_1^2(-0.5y^3) = -1.5y_1^2y_2 && y_2(0) = y_1(0)^3\\
&y_3 = y_1^2 && \\
&y_3' = 2y_1y_1' = 2y_1(-0.5y^3) = -y_1y_2 && y_3(0) = y_1(0)^2\,,
\end{aligned}$$

so the final system becomes

$$\begin{aligned}
&y_1' = -0.5y_2 && y_1(0) = 1\\
&y_2' = -1.5y_2y_3 && y_2(0) = 1\\
&y_3' = -y_1y_2 && y_3(0) = 1
\end{aligned}$$

which can be transformed into a matrix-vector representation (3.19)

$$\boldsymbol{A} = \begin{pmatrix} 0 & -1.5 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \boldsymbol{B}_1 = \begin{pmatrix} 0 & 0 \\ -1.5 & 0 \\ 0 & 1 \end{pmatrix} \qquad \boldsymbol{y}_{jk} = \begin{pmatrix} 3 & 2 \\ 1 & 2 \end{pmatrix} .$$

E.3 Problem A3

Problem A3 is defined as

$$y' = y\cos(t) \qquad y(0) = 1 ,$$

the auxiliary system of ODEs representing the system can be written as

$$\begin{aligned} y_1' &= y_1 y_2 & y_1(0) &= 1 \\ y_2' &= -y_3 & y_2(0) &= \cos(0) = 1 \\ y_3' &= y_2 & y_3(0) &= \sin(0) = 0 , \end{aligned}$$

and it can be transformed into the matrix-vector representation (3.19)

$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \qquad \boldsymbol{B}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \qquad \boldsymbol{y}_{jk} = \begin{pmatrix} 1 & 2 \end{pmatrix} .$$

E.4 Problem A4

Problem A4 is defined as

$$y' = \frac{y}{4} - \frac{y}{20} \qquad y(0) = 1 ,$$

the auxiliary system of ODEs representing the system can be written as

$$\begin{aligned} y_1' &= 0.25y_1 - (1/80)y_2 & y_1(0) &= 1 \\ y_2' &= 0.5y_2 - (1/40)y_1 y_2 & y_2(0) &= y_1(0)y_1(0) \end{aligned}$$

and it can be transformed into the matrix-vector representation (3.19)

$$\boldsymbol{A} = \begin{pmatrix} 0.25 & -\frac{1}{80} \\ 0 & 0.5 \end{pmatrix} \qquad \boldsymbol{B}_1 = \begin{pmatrix} 0 \\ -\frac{1}{40} \end{pmatrix} \qquad \boldsymbol{y}_{jk} = \begin{pmatrix} 1 & 2 \end{pmatrix} .$$

E.5 Problem B1

Problem B1 is defined as

$$\begin{aligned} y_1' &= 2y_1 - 2y_1 y_2 & y_1(0) &= 1 \\ y_2' &= -y_2 + y_1 y_2 & y_2(0) &= 3 \end{aligned}$$

with the auxiliary system not being necessary, the system can be transformed into the matrix-vector representation (3.19)

$$\boldsymbol{A} = \begin{pmatrix} 0.25 & -\frac{1}{80} \\ 0 & 0.5 \end{pmatrix} \quad \boldsymbol{B}_1 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \boldsymbol{y}_{jk} = \begin{pmatrix} 2 & 4 \\ 4 & 4 \end{pmatrix} \quad \boldsymbol{B}_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \end{pmatrix} \quad \boldsymbol{y}_{jklm} = \begin{pmatrix} 4 & 4 & 4 & 2 \end{pmatrix} .$$

E.6 Problem B2

Problem B2 is defined as

$$\begin{aligned} y_1' &= -y_1 + y_2 & y_1(0) &= 2 & \text{(E.1)}\\ y_2' &= -y_1 - 2y_2 + y_3 & y_2(0) &= 0 & \text{(E.2)}\\ y_3' &= y_2 - y_3 & y_3(0) &= 1\,. & \text{(E.3)} \end{aligned}$$

The problem is linear. To solve it using MTSM, the equation can be directly transformed into matrix-vector representation

$$\boldsymbol{A} = \begin{pmatrix} -1 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -1 \end{pmatrix} \qquad \boldsymbol{b} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}\,.$$

E.7 Problem B3

Problem B3 is defined as

$$\begin{aligned} y_1' &= -y_1 & y_1(0) &= 1\\ y_2' &= y_1 - y_2y_2 & y_2(0) &= 0\\ y_3' &= y_2y_2 & y_3(0) &= 0 \end{aligned}$$

with the auxiliary system not being necessary, the system can be directly transformed into the matrix-vector representation (3.19)

$$\boldsymbol{A} = \begin{pmatrix} -1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \boldsymbol{B}_1 = \begin{pmatrix} 0 & 0 \\ -1 & 0 \\ 1 & 0 \end{pmatrix} \qquad \boldsymbol{y}_{jk} = \begin{pmatrix} 2 & 2 \end{pmatrix}\,.$$

E.8 Problem B4

This section contains the matrix-vector representation for the benchmark problem B4, which is detailed in Section 4.1.9. The matrix-vector representation for the linear part of (4.4):

$$\boldsymbol{A} = \begin{pmatrix} 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \end{pmatrix} \qquad \boldsymbol{b} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

and for the non-linear part of (4.4):

$$\boldsymbol{B}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad \boldsymbol{y}_{jk} = \begin{pmatrix} 1 & 5 \\ 9 & 11 \\ 1 & 2 \\ 8 & 9 \end{pmatrix}$$

$$\boldsymbol{B}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \boldsymbol{y}_{jkl} = \begin{pmatrix} 2 & 3 & 5 \\ 3 & 8 & 9 \\ 3 & 8 & 10 \\ 3 & 6 & 9 \\ 3 & 6 & 10 \\ 3 & 5 & 10 \\ 2 & 3 & 5 \\ 3 & 5 & 11 \end{pmatrix}$$

$$\boldsymbol{B}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \boldsymbol{y}_{jklm} = \begin{pmatrix} 3 & 6 & 7 & 9 \\ 3 & 6 & 7 & 10 \\ 3 & 6 & 8 & 9 \\ 3 & 6 & 8 & 10 \\ 3 & 5 & 6 & 9 \\ 3 & 5 & 6 & 10 \\ 1 & 6 & 9 & 12 \\ 1 & 6 & 10 & 12 \end{pmatrix}.$$

E.9 Problem B5

Problem B5 is defined as

$$\begin{aligned} y_1' &= y_2 y_3 & \qquad y_1(0) &= 0, \\ y_2' &= -y_1 y_3 & \qquad y_2(0) &= 1, \\ y_3' &= -0.51 y_1 y_2 & \qquad y_3(0) &= 1. \end{aligned}$$

With the auxiliary system not being necessary, the matrix vector representation can be obtained directly

$$\boldsymbol{B}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -0.51 \end{pmatrix} \qquad \boldsymbol{y}_{jk} = \begin{pmatrix} 2 & 3 \\ 1 & 3 \\ 1 & 2 \end{pmatrix} .$$

E.10 Problem C3

Problem C3 is defined directly in the matrix-vector representation

$$\boldsymbol{A} = \begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix} \quad \boldsymbol{b} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \boldsymbol{y}_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} .$$

E.11 Problem E1

Problem E1 is defined as

$$\begin{aligned} y_1' &= y_2 & y_1(0) &= 0.6713967071418030 \\ y_2' &= -\frac{y_2}{t+1} - y_1 + 0.25\frac{y_1}{(t+1)^2} & y_2(0) &= 0.09540051444747446. \end{aligned}$$

The operation division has to be removed from the original system:

$$\begin{aligned} y_3 &= t+1 & & \\ y_3' &= 1 & y_3(0) &= t_0 + 1 = 1 \\ y_4 &= \frac{1}{t+1} = \frac{1}{y_3} = y_3^{-1} & & \\ y_4' &= -y_3^{-2} y_3' = -y_4^2 & y_4(0) &= \frac{1}{t_0+1} = 1 \\ y_5 &= y_4^2 & & \\ y_5' &= 2y_4 y_4' = -2y_4 y_4 y_4 = -2y_4 y_5 & y_5(0) &= \frac{1}{(t_0+1)^2} = 1 \end{aligned}$$

where t_0 is the initial time of calculation, which is zero in this case. Augmented system of ODEs becomes

$$\begin{aligned} y_1' &= y_2 & y_1(0) &= 0.6713967071418030 \\ y_2' &= -y_2 y_4 - y_1 - 0.5 y_1 y_4 y_5 & y_2(0) &= 0.09540051444747446 \\ y_3' &= 1 & y_3(0) &= 1 \\ y_4' &= -y_4 y_4 & y_4(0) &= 1 \\ y_5' &= -2 y_4 y_5 & y_5(0) &= 1. \end{aligned} \tag{E.4}$$

The system (E.4) can be rewritten into matrix-vector notation

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \boldsymbol{b} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\boldsymbol{B}_1 = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -2 \end{pmatrix} \qquad \boldsymbol{B}_2 = \begin{pmatrix} 0 \\ -0.5 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\boldsymbol{y}_{jk} = \begin{pmatrix} 2 & 4 \\ 4 & 4 \\ 4 & 5 \end{pmatrix} \qquad \boldsymbol{y}_{jkl} = \begin{pmatrix} 1 & 4 & 5 \end{pmatrix}.$$

E.12 Problem E2

Problem E2 is defined as:

$$\begin{aligned} y_1' &= y_2 & y_1(0) &= 2 \\ y_2' &= y_2 - y_1 y_1 y_2 - y_1 & y_2(0) &= 0. \end{aligned} \tag{E.5}$$

The system (E.5) can be transformed into the matrix-vector notation as is. However, it can be further optimized by removing the three term multiplication $y_1 y_1 y_2$:

$$\begin{aligned} y_3 &= y_1 y_1 = y_1^2 \\ y_3' &= 2 y_1 y_1' = 2 y_1 y_2 & y_3(0) &= y_1(0) y_1(0) = 4 \end{aligned}$$

and adding this equation into (E.5) and substituting

$$\begin{aligned} y_1' &= y_2 & y_1(0) &= 2 \\ y_2' &= y_2 - y_2 y_3 - y_1 & y_2(0) &= 0 \\ y_3' &= 2 y_1 y_2 & y_3(0) &= 4. \end{aligned} \tag{E.6}$$

The system (E.6) can be rewritten into a matrix-vector notation

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \boldsymbol{b} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\boldsymbol{B}_1 = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -2 \end{pmatrix} \qquad \boldsymbol{B}_2 = \begin{pmatrix} 0 \\ -0.5 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\boldsymbol{y}_{jk} = \begin{pmatrix} 2 & 4 \\ 4 & 4 \\ 4 & 5 \end{pmatrix} \qquad \boldsymbol{y}_{jkl} = \begin{pmatrix} 1 & 4 & 5 \end{pmatrix}.$$

E.13 Problem E3

Problem E3 is defined as

$$\begin{aligned} y_1' &= y_2 & y_1(0) &= 0 \\ y_2' &= \frac{y_1^3}{6} - y_1 + 2\sin(2.78535t) & y_2(0) &= 0. \end{aligned} \tag{E.7}$$

The system (E.7) cannot be transformed into the matrix-vector notation as is, because the term $\sin(2.78535t)$ has to be replaced by the system of auxiliary ODEs:

$$\begin{aligned} y_3 &= \sin(2.78535t) & & \\ y_3' &= 2.78535\cos(2.78535t) & y_3(0) &= \sin(2.78535t_0) = \sin(2.78535) \\ y_4 &= \cos(2.78535t) & & \\ y_4' &= -2.78535\sin(2.78535t) & y_4(0) &= \cos(2.78535t_0) = \cos(2.78535) \end{aligned}$$

where t_0 is the initial time of computation. Further the three function multiplication y_1^3 can also be removed

$$\begin{aligned} y_5 &= y_1^3 & & \\ y_5' &= 3y_1^2 y1' = 3y_1^2 y_2 & y_5(0) &= y_1(0)y_1(0)y_1(0) = 0 \\ y_6 &= y_1^2 & & \\ y_6' &= 2y_1 y_1' = 2y_1 y_2 & y_6(0) &= y_1(0)y_1(0) = 0. \end{aligned}$$

Adding the three ODEs above into the E.7 and substituting

$$\begin{aligned} y_1' &= y_2 & y_1(0) &= 0 \\ y_2' &= \frac{1}{6}y_5 - y_1 + 2y_3 & y_2(0) &= 0 \\ y_3' &= 2.78535y_4 & y_3(0) &= \sin(2.78535) \\ y_4' &= -2.78535y_3 & y_4(0) &= \cos(2.78535) \\ y_5' &= 3y_2y_6 & y_5(0) &= 0 \\ y_6' &= 2y_1y_2 & y_6(0) &= 0. \end{aligned} \tag{E.8}$$

The system (E.8) can be rewritten into a matrix-vector notation

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 2 & 0 & \frac{1}{6} & 0 \\ 0 & 0 & 0 & 2.78535 & 0 & 0 \\ 0 & 0 & 2.78535 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \boldsymbol{B}_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 3 & 0 \\ 0 & 2 \end{pmatrix} \quad \boldsymbol{B}_2 = \begin{pmatrix} 0 \\ -0.5 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\boldsymbol{y}_{jk} = \begin{pmatrix} 2 & 6 \\ 1 & 2 \end{pmatrix}.$$

E.14 Problem E4

Problem E4 is defined as

$$\begin{aligned} y_1' &= y_2 & y_1(0) &= 30 \\ y_2' &= 0.032 - 0.4 y_2 y_2 & y_2(0) &= 0. \end{aligned} \tag{E.9}$$

The system (E.9) can be transformed into the matrix-vector notation as is:

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \qquad \boldsymbol{b} = \begin{pmatrix} 0 \\ 0.032 \end{pmatrix} \qquad \boldsymbol{B}_1 = \begin{pmatrix} 0 & 0 \\ 0.4 & 0 \end{pmatrix} \qquad \boldsymbol{y}_{jk} = \begin{pmatrix} 2 & 2 \end{pmatrix}.$$

E.15 Problem E5

This section contains the matrix-vector representation for the benchmark problem E5, defined in Subsection 4.1.15. The matrix-vector representation for (4.7) follows:

$$\boldsymbol{A} = \begin{pmatrix} 0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \end{pmatrix} \qquad \boldsymbol{B}_1 = \begin{pmatrix} 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&-1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&2&0&0 \\ 0&0&0&0&0&0&1&1 \end{pmatrix}$$

$$\boldsymbol{B}_2 = \begin{pmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ -3&0&0&0 \\ 0&-2&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \end{pmatrix} \qquad \boldsymbol{y}_{jk} = \begin{pmatrix} 5&8 \\ 6&8 \\ 9&9 \\ 3&8 \\ 4&10 \\ 3&10 \\ 3&9 \\ 2&10 \end{pmatrix}$$

$$\boldsymbol{y}_{jkl} = \begin{pmatrix} 6&7&8 \\ 5&6&8 \\ 5&8&11 \\ 3&5&8 \end{pmatrix}.$$

Appendix F

Matrix-vector representation of Kepler problem

This Appendix contains the matrix-vector representations for Kepler problem, discussed in Section 4.7.

F.1 Matrix-vector representation for (4.62)

$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \boldsymbol{B}_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$
$$\boldsymbol{B}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 3 & 3 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad \boldsymbol{B}_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -3 & -3 & 0 & 0 \\ 0 & 0 & -1 & -1 \end{pmatrix}. \tag{F.1}$$

The vectors on the right hand side of (3.19) are defined as follows:

$$\boldsymbol{y_{jk}} = \begin{pmatrix} y_1 & y_7 \\ y_2 & y_7 \end{pmatrix} \quad \boldsymbol{y_{jkl}} = \begin{pmatrix} y_6 & y_1 & y_3 \\ y_6 & y_2 & y_4 \\ y_1 & y_3 & y_8 \\ y_2 & y_4 & y_8 \end{pmatrix} \boldsymbol{y_{jklmn}} = \begin{pmatrix} y_6 & y_1 & y_3 & y_7 & y_7 \\ y_6 & y_2 & y_4 & y_7 & y_7 \\ y_1 & y_3 & y_8 & y_8 & y_8 \\ y_2 & y_4 & y_8 & y_8 & y_8 \end{pmatrix}.$$

F.2 Matrix-vector representation for (4.63)

$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \boldsymbol{B}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \end{pmatrix}$$

$$\boldsymbol{B}_2 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -1 & -1 \end{pmatrix} \qquad \boldsymbol{B}_3 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -3 & 0 \\ 0 & -1 \\ 0 & 0 \end{pmatrix}. \tag{F.2}$$

The vectors on the right hand side of (3.19) are defined as follows:

$$\boldsymbol{y_{jk}} = \begin{pmatrix} y_1 & y_7 \\ y_2 & y_7 \\ y_6 & y_9 \\ y_8 & y_9 \\ y_3 & y_3 \\ y_4 & y_4 \\ y_1 & y_7 \\ y_2 & y_7 \end{pmatrix} \qquad \boldsymbol{y_{jkl}} = \begin{pmatrix} y_7 & y_1 & y_1 \\ y_7 & y_2 & y_2 \end{pmatrix} \qquad \boldsymbol{y_{jklm}} = \begin{pmatrix} y_6 & y_7 & y_7 & y_9 \\ y_8 & y_8 & y_8 & y_9 \end{pmatrix}.$$

F.3 Matrix-vector representation for (4.64)

$$\boldsymbol{A} = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\boldsymbol{B}_1 = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 2
\end{pmatrix}$$

$$\boldsymbol{B}_2 = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix} \tag{F.3}$$

The vectors on the right hand side of (3.19) are defined as follows:

$$\boldsymbol{y_{jk}} = \begin{pmatrix}
y_1 & y_7 \\
y_2 & y_7 \\
y_3 & y_3 \\
y_4 & y_4 \\
y_7 & y_{12} \\
y_7 & y_{11} \\
y_1 & y_3 \\
y_2 & y_4
\end{pmatrix}
\quad
\boldsymbol{y_{jkl}} = \begin{pmatrix}
y_7 & y_7 & y_{10} \\
y_8 & y_8 & y_{11} \\
y_6 & y_3 & y_3 \\
y_6 & y_4 & y_4 \\
y_6 & y_7 & y_{12} \\
y_8 & y_{11} & y_{11} \\
y_8 & y_3 & y_3 \\
y_8 & y_4 & y_4 \\
y_8 & y_7 & y_{12}
\end{pmatrix}.$$

Appendix G

Matrix-vector representation of the pendulum on the cart

This Appendix contains the matrix-vector representations of systems used in Chapter 6.3.

G.1 Representation for (6.40)

The matrix-vector representation for (6.40) can be written as

$$
\boldsymbol{A} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & u & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\quad
\boldsymbol{B}_1 = \begin{pmatrix}
-u & Mg+mg & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -mg & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 \\
0 & 0 & 0 & 0 & 0 & 2ml \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2m
\end{pmatrix}
$$

$$
\boldsymbol{B}_2 = \begin{pmatrix}
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
1 & -1
\end{pmatrix}
\quad
\boldsymbol{B}_3 = \begin{pmatrix}
-ml & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & ml & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & -2ml & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & -2m \\
0 & 0 & 0 & 0
\end{pmatrix},
$$

with $\boldsymbol{b} = \mathbf{0}^T$. The vectors on the right hand side of (3.19) are defined as follows:

$$\boldsymbol{y_{jk}} = \begin{pmatrix} y_6 & y_9 \\ y_5 & y_9 \\ y_{12} & y_{11} \\ y_6 & y_1 \\ y_5 & y_1 \\ y_{12} & y_1 \end{pmatrix} \quad \boldsymbol{y_{jkl}} = \begin{pmatrix} y_6 & y_1 & y_6 \\ y_5 & y_5 & y_1 \end{pmatrix}, \quad \boldsymbol{y_{jklm}} = \begin{pmatrix} y_{12} & y_1 & y_1 & y_9 \\ y_5 & y_1 & y_1 & y_{11} \\ y_9 & y_9 & y_{12} & y_1 \\ y_{11} & y_{11} & y_{12} & y_1 \end{pmatrix}.$$

G.2 Representation for (6.44)

The matrix-vector representation for (6.44) can be written as

$$\boldsymbol{A} = \begin{pmatrix} 0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&\boldsymbol{kr}&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \end{pmatrix} \quad \boldsymbol{B}_1 = \begin{pmatrix} \boldsymbol{kr} & Mg+mg & 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&-k_1&-k_2&-k_3&-k_4&-mg&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&-1&0 \\ 0&0&0&0&0&0&0&0&0&2 \\ 0&0&0&0&0&0&0&0&0&2ml \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&2m \end{pmatrix}$$

$$\boldsymbol{B}_2 = \begin{pmatrix} k_1&k_2&k_3&k_4&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&1&-1 \end{pmatrix} \quad \boldsymbol{B}_3 = \begin{pmatrix} -ml&0&0&0 \\ 0&0&0&0 \\ 0&ml&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&-2ml&0 \\ 0&0&0&0 \\ 0&0&0&-2m \\ 0&0&0&0 \end{pmatrix},$$

with $\boldsymbol{b} = \mathbf{0}^T$. The vectors on the right hand side of (3.19) are defined as follows:

$$\boldsymbol{y_{jk}} = \begin{pmatrix} y_6 & y_9 \\ y_5 & y_9 \\ y_1 & y_{11} \\ y_2 & y_{11} \\ y_3 & y_{11} \\ y_4 & y_{11} \\ y_{12} & y_{11} \\ y_6 & y_1 \\ y_5 & y_1 \\ y_{12} & y_1 \end{pmatrix} \quad \boldsymbol{y_{jkl}} = \begin{pmatrix} y_1 & y_6 & y_9 \\ y_2 & y_6 & y_9 \\ y_3 & y_6 & y_9 \\ y_4 & y_6 & y_9 \\ y_6 & y_1 & y_6 \\ y_5 & y_5 & y_1 \end{pmatrix}, \quad \boldsymbol{y_{jklm}} = \begin{pmatrix} y_{12} & y_1 & y_1 & y_9 \\ y_5 & y_1 & y_1 & y_{11} \\ y_9 & y_9 & y_{12} & y_1 \\ y_{11} & y_{11} & y_{12} & y_1 \end{pmatrix}.$$

Appendix H

Derivation of the equations for pendulum on the cart

This Appendix contains derivations for (6.37) and (6.38) in Section 6.3.

H.1 Coordinate substitution

The x_G coordinate can be substituted into (6.31) and the resulting equation simplified

$$\begin{aligned}
Mx'' + m\left(x + l\sin(\phi)\right)'' &= u \\
Mx'' + m\left(x' + l\cos(\phi)\phi'\right)' &= u \\
Mx'' + m\left(x'' - l\sin(\phi)\phi'\phi' + l\cos(\phi)\phi''\right) &= u \\
Mx'' + mx'' - ml\sin(\phi)\phi'^2 + ml\cos(\phi)\phi'' &= u
\end{aligned}$$

H.2 Forces on the pendulum

The forces F_x can be calculated as

$$\begin{aligned}
F_x &= mx''_G = m\left(x + l\sin(\phi)\right)'' = m\left(x' + l\cos(\phi)\phi'\right)' \\
&= m\left(x'' - l\sin(\phi)\phi'\phi' + l\cos(\phi)\phi''\right) = mx'' - ml\sin(\phi)\phi'^2 + ml\cos(\phi)\phi'' \\
F_y &= my''_G = m\left(l\cos(\phi)\right)'' = m\left(-l\sin(\phi)\phi'\right)' = \\
&= m\left(-l\cos(\phi)\phi'\phi' - l\sin(\phi)\phi''\right) = -ml\cos(\phi)\phi'^2 - ml\sin(\phi)\phi''\,.
\end{aligned} \tag{H.1}$$

Forces derived from (H.1) can be substituted into (6.34)

$$\begin{aligned}
&F_x\cos(\phi) - F_y\sin(\phi) = mg\sin(\phi) \\
&\left(mx'' - ml\sin(\phi)\phi'^2 + ml\cos(\phi)\phi''\right)\cos(\phi) - \left(-ml\cos(\phi)\phi'^2 - ml\sin(\phi)\phi''\right)\sin(\phi) = mg\sin(\phi) \\
&m\cos(\phi)x'' - ml\sin(\phi)\cos(\phi)\phi'^2 + ml\cos^2(\phi)\phi'' + ml\sin(\phi)\cos(\phi)\phi'^2 + ml\sin^2(\phi)\phi'' = mg\sin(\phi) \\
&m\cos(\phi)x'' + ml\cos^2(\phi)\phi'' + ml\sin^2(\phi)\phi'' = mg\sin(\phi) \\
&m\cos(\phi)x'' + ml\phi''\left(\cos^2(\phi) + \sin^2(\phi)\right) = mg\sin(\phi) \\
&m\cos(\phi)x'' + ml\phi'' = mg\sin(\phi) \qquad \to \qquad \cos(\phi)x'' + l\phi'' = g\sin(\phi)\,.
\end{aligned}$$

H.3 Position of the cart

Rearranging (6.36) to solve ϕ''

$$\phi'' = \frac{g\sin(\phi)}{l} - \frac{\cos(\phi)x''}{l}$$

and substituting the obtained equation into (6.35)

$$\begin{aligned}
&Mx'' + mx'' - ml\sin(\phi)\phi'^2 + ml\cos(\phi)\frac{g\sin(\phi)}{l} - ml\cos(\phi)\frac{\cos(\phi)x''}{l} = u\\
&Mx'' + mx'' - ml\sin(\phi)\phi'^2 + m\cos(\phi)g\sin(\phi) - m\cos(\phi)\cos(\phi)x'' = u\\
&Mx'' + mx'' - m\cos(\phi)\cos(\phi)x'' = u + ml\sin(\phi)\phi'^2 - mg\cos(\phi)\sin(\phi)\\
&x''\left(M + m - m\cos(\phi)\cos(\phi)\right) = u + ml\sin(\phi)\phi'^2 - mg\cos(\phi)\sin(\phi)\\
&x'' = \frac{u + ml\sin(\phi)\phi'^2 - mg\cos(\phi)\sin(\phi)}{M + m(1 - \cos^2(\phi))}
\end{aligned}$$

and using trigonometric identity $\sin^2(\phi) + \cos^2(\phi) = 1$ [9] the final second order ODE for the position of the cart (representing its acceleration in m·s^{-2}) can be written as

$$x'' = \frac{u + ml\sin(\phi)\phi'^2 - mg\sin(\phi)\cos(\phi)}{M + m\sin^2(\phi)}\,.$$

H.4 Angle of the pendulum

Rearranging (6.36) to solve x''

$$x'' = \frac{g\sin(\phi)}{\cos(\phi)} - \frac{l\phi''}{\cos(\phi)}$$

and substituting the obtained equation into (6.35)

$$\begin{aligned}
&(M + m)\,x'' - ml\sin(\phi)\phi'^2 + ml\cos(\phi)\phi'' = u\\
&Mx'' + mx'' - ml\sin(\phi)\phi'^2 + ml\cos(\phi)\phi'' = u\\
&M(\frac{g\sin(\phi)}{\cos(\phi)} - \frac{l\phi''}{\cos(\phi)}) + m(\frac{g\sin(\phi)}{\cos(\phi)} - \frac{l\phi''}{\cos(\phi)}) - ml\sin(\phi)\phi'^2 + ml\cos(\phi)\phi'' = u\\
&\frac{Mg\sin(\phi)}{\cos(\phi)} - \frac{Ml\phi''}{\cos(\phi)} + \frac{mg\sin(\phi)}{\cos(\phi)} - \frac{ml\phi''}{\cos(\phi)} - ml\sin(\phi)\phi'^2 + ml\cos(\phi)\phi'' = u\\
&Mg\sin(\phi) - Ml\phi'' + mg\sin(\phi) - ml\phi'' - ml\sin(\phi)\cos(\phi)\phi'^2 + ml\cos(\phi)^2\phi'' = u\cos(\phi)\\
&-Ml\phi'' - ml\phi'' + ml\cos(\phi)^2\phi'' = u\cos(\phi) - Mg\sin(\phi) - mg\sin(\phi) + ml\sin(\phi)\cos(\phi)\phi'^2\\
&Ml\phi'' + ml\phi'' - ml\cos(\phi)^2\phi'' = -u\cos(\phi) + Mg\sin(\phi) + mg\sin(\phi) - ml\sin(\phi)\cos(\phi)\phi'^2\\
&\phi''\left(Ml + ml(1 - \cos(\phi)^2)\right) = -u\cos(\phi) + Mg\sin(\phi) + mg\sin(\phi) - ml\sin(\phi)\cos(\phi)\phi'^2\\
&\phi'' = \frac{-u\cos(\phi) + Mg\sin(\phi) + mg\sin(\phi) - ml\sin(\phi)\cos(\phi)\phi'^2}{Ml + ml(1 - \cos(\phi)^2}
\end{aligned}$$

and using trigonometric identity $\sin^2(\phi) + \cos^2(\phi) = 1$ [9] the final second order ODE for the angle of the pendulum (representing its acceleration in rad·s^{-2}) can be written as

$$\phi'' = \frac{-u\cos(\phi) + (Mg + mg)\sin(\phi) - ml\sin(\phi)\cos(\phi)\phi'^2}{Ml + ml\sin^2(\phi)}\,.$$