BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

PHD THESIS

Brno, 2023 Ing. Gabriela Nečasová

BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

PARALLEL NUMERIC SOLUTION OF DIFFERENTIAL EQUATIONS

PARALELNÍ NUMERICKÉ ŘEŠENÍ DIFERENCIÁLNÍCH ROVNIC

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR — **Ing. GABRIELA NEČASOVÁ**
AUTOR PRÁCE

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

BRNO 2023

Abstract

Differential equations have been studied for over 300 years. Partial differential equations were first used by the Swiss mathematician and lawyer Nicolaus Bernoulli in the 18th century. Second-order partial differential equations are used to model a wide range of phenomena in science, engineering, and mathematics, such as the propagation of light and sound waves, the motion of fluids, and the diffusion of heat. The thesis deals with the parallel numerical solution of partial differential equations. Second-order partial differential equations are transformed into large systems of ordinary differential equations using the method of lines. The spatial derivatives in the partial differential equation are replaced by various types of finite differences. The resulting large systems of ordinary differential equations (initial value problem) are solved in parallel using Runge-Kutta methods and the newly proposed higher-order method based on Taylor series. The numerical experiments of the selected problems are calculated using a supercomputer with different numbers of compute nodes. The results show that the Taylor series-based numerical method significantly overperforms the state-of-the-art Runge-Kutta methods.

Abstrakt

Diferenciální rovnice se studují již vice než 300 let. Poprvé parciální diferenciální rovnice použil švýcarský matematik a právník Nicolaus Bernoulli v 18. století. Parciální diferenciální rovnice druhého řádu se používají k modelování široké škály jevů ve vědě, technice a matematice, například šíření světelných a zvukových vln, pohybu tekutin a šíření tepla. Práce se zabývá paralelním numerickým řešením parciálních diferenciálních rovnic. Parciální diferenciální rovnice druhého řádu jsou pomocí metody přímek převedeny na rozsáhlé soustavy obyčejných diferenciálních rovnic. Prostorové derivace v parciální diferenciální rovnici jsou nahrazeny různými typy konečných diferencí. Výsledné soustavy obyčejných diferenciálních rovnic (problémy počátečních hodnot) jsou řešeny paralelně pomocí Runge-Kutta metod a nově navržené metody vyššího řádu založené na Taylorově řadě. Numerické experimenty vybraných problémů jsou realizovány na superpočítači s různým počtem výpočetních uzlů. Výsledky ukazují, že metoda založená na Taylorově řadě výrazně překonává standardní Runge-Kutta metody.

Keywords

partial differential equations, ordinary differential equations, initial value problems, Taylor series, Runge-Kutta, method of lines, finite differences, parallel computations, supercomputers

Klíčová slova

parcilální diferenciální rovnice, obyčejné diferenciální rovnice, problémy počátečních hodnot, Taylorova řada, Runge-Kutta, metoda přímek, konečné diference, paralelní výpočty, superpočítače

Reference

NEČASOVÁ, Gabriela. *Parallel numeric solution of differential equations.* Brno, 2023. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Václav Šátek, Ph.D.

Rozšířený abstrakt

Parciální diferenciální rovnice jsou nezbytným nástrojem matematického modelování problémů reálného světa. Matematickým modelem je soustava rovnic anebo jiných matematických vztahů, které zachycují podstatné vlastnosti systémů nezbytné k jejich popisu či předpovědi jejich chování. Parciální diferenciální rovnice druhého řádu se vyskytují v mnoha matematických, fyzikálních i technických problémech. Umožňují modelování a analýzu jevů, jako například přenos tepla, šíření vln, dynamika tekutin, elektromagnetické pole, a jiné. Diferenciální rovnice za sebou mají přibližně 300letý vývoj. Poprvé parciální diferenciální rovnice použil švýcarský matematik a právník Nicolaus Bernoulli v 18. století. Rozkvět v této oblasti je spojen se slavnými jmény jako například Leonard Euler, Joseph-Louis Lagrange, Adrien-Marie Legendre, Pierre-Simon Laplace nebo Joseph Fourier.

Matematické modely sestávající z parciálních diferenciálních rovnic obvykle nelze vyřešit analyticky a jejich řešení je potřeba aproximovat numerickými metodami, například metodou přímek, metodou konečných diferencí, metodou konečných prvků nebo metodou konečných objemů.

V této práci byla využita semidiskretizační metoda přímek, která diskretizuje derivace pouze v prostorové doméně a danou parciální diferenciální rovnici tudíž převádí na soustavu obyčejných diferenciálních rovnic. Vzniklá soustava obyčejných diferenciálních rovnic je řešena pomocí Runge-Kutta metod a navržené metody vyššího řádu založené na Taylorově rozvoji. Tato metoda používá proměnný integrační krok a řád a je založena na rekurentním výpočtu členů Taylorovy řady v každém časovém intervalu. Důležitou vlastností metody je automatická volba řádu v závislosti na velikosti integračního kroku, což znamená, že je využito tolik členů Taylorovy řadu, kolik vyžaduje zadaná přesnost výpočtu.

Paralelní řešení diferenciálních rovnic metodami založenými na Taylorově řadě není příliš rozšířené. Jedním z důvodů je výpočetní závislost mezi členy Taylorovy řady. Několik publikací pojednává o paralelních implementacích založených na Taylorově řadě, avšak žádná z nich se nezabývá řešením rozsáhlých soustav obyčejných diferenciálních rovnic vznikajících z parciálních diferenciálních rovnic.

Cílem práce je ukázat, že rozsáhlé soustavy obyčejných diferenciálních rovnic vznikajících z parciálních diferenciálních rovnic pomocí metody přímek, lze vyřešit efektivněji pomocí paralelní metody založené na Taylorově rozvoji než běžnými numerickými metodami. V práci byly provedeny numerické experimenty na vybrané třídě úloh modelovanými parciálními diferenciálními rovnicemi druhého řádu. Konkrétně se jednalo o rovnici vedení tepla, vlnovou rovnici a telegrafní rovnici. V prostorové doméně byly využity tříbodové a pětibodové centrální konečné diference. Velikosti soustav obyčejných diferenciálních rovnic byly 128000, 256000, 512000, 1024000 a 2048000 rovnic. K řešení těchto soustav byly požity metody založené na Taylorově řadě a Runge-Kutta metody. MTSM metoda je založena na rekurentním výpočtu členů Taylorovy řady v každém integračním kroku. MTSM_PRECALC metoda přepočítává členy Taylorovy řady před začátkem výpočtu. Díky tomuto přístupu dochází k eliminaci závislostí mezi členy. Mezi zástupce Runge-Kutta metod byla zvolena Dormand-Prince metoda čtvrtého a pátého řádu (známá jako ode45 v MATLABu), zde označena jako TSRK5DP a Vernerova metoda sedmého a osmého řádu označena TSRK8VR. Finální numerické experimenty byly realizovány na superpočítači Barbora (IT4Innovations národní superpočítačové centrum), s využitím 32 výpočetních uzlů (36 procesů na 1 uzel), celkem tedy 1152 MPI procesů.

Bylo sledováno několik výkonnostních metrik jako jsou průměrný čas výpočtu, paralelní efektivita, zrychlení a zrychlení vůči TSRK5DP metodě. Mezi metriky zaměřené na cenu výpočtu patří zejména poměr zrychlení k ceně výpočtu.

Výsledky ukázaly, že MTSM_PRECALC metoda je vždy nejrychlejší ze všech metod. Přestože MTSM trpí výpočetními závislostmi mezi členy Taylorovy řady, může poskytnout výsledky srovnatelné nebo dokonce lepší než TSRK5DP nebo TSRK8VR, zejména pro vlnovou a telegrafní rovnici. Chování metod TSRK5DP a TSRK8VR je většinou velice podobné, v některých případech je TSRK5DP mírně rychlejší. Co se týče poměru zrychlení k ceně výpočtu, pro MTSM_PRECALC se vyplatí alokovat všech 32 výpočetních uzlů pro rovnici vedení tepla a telegrafní rovnici. Naopak pro vlnovou rovnici má smysl 32 uzlů alokovat pro úlohy větší nebo rovno 512000 rovnic. Pro MTSM se alokace 32 uzlů vyplatí pouze u telegrafní rovnice. Pro TSRK5DP pouze pro telegrafní rovnici s 2048000 rovnicemi a pro TSRK8VR pro rovnici vedení tepla diskretizovanou tříbodovými centrálními vzorci pro 1024000 rovnic a v případě diskretizace pětibodovými centrálními vzorci pro 512000 a 1024000 rovnic.

Mezi možná rozšíření této práce můžeme zmínit například analýzu ukončovací podmínky integračního kroku, redukci výpočetního času monitorováním hodnot členů Taylorovy řady, vylepšení volby integračního kroku, zaměření na vícedimenzionální parciální diferenciální rovnice, podporu rozšířené aritmetiky nebo paralelizaci nelineárních problémů.

Parallel numeric solution of differential equations

Declaration

Hereby, I declare that this dissertation thesis was prepared as an original author's work under the supervision of Ing. Václav Šátek, Ph.D. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Gabriela Nečasová
February 27, 2023

Acknowledgements

I would like to thank my supervisor Ing. Václav Šátek, Ph.D., for his very kind, valuable, and insightful advice, which helped me to create this thesis. I would also like to express my gratitude to doc. Ing. Jiří Kunovský, CSc. even he is not unfortunately with us anymore. His thoughts, his positive energy, and "fluidum" will be here forever. I am very grateful to doc. Ing. Jiří Jaroš, Ph.D., for his consultations and feedback.

Special thanks to the extraordinary people, my boyfriend Martin Sakin, my friend and colleague Petr Veigend, for their amazing support, advises and giving me the courage all the time. I want to thank to my friends Marta Jaroš, Ondra Olšák, Janka Puterová, and Dominika Regéciová for their unconditional mental support, enthusiasm, and positive vibes.

Many thanks belong to my family and friends who supported me throughout my studies. I want to thank my mother, Dana Nečasová, my father, Vladimír Nečas, and my sister Klára Nečasová for their unlimited support and patience with me.

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90140), 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 symbols and abbreviations

IVP	Initial Value Problem
MOL	Method of Lines
MTSM	Modern Taylor Series Method
MPI	Message Passing Interface
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
$AbsTol$	Absolute tolerance
$RelTol$	Relative tolerance
h	Step size in the time domain
Δx	Step size in the spatial domain
$maxORD$	Maximum order of Modern Taylor Series Method
t_{max}	Maximum simulation time
S	Number of segments in the spatial domain

Chapter 1

Introduction

Differential equations have been studied for over 300 years, dating back to the work of Isaac Newton (1643–1727) and Gottfried Wilhelm von Leibniz (1646–1716) in the 17th century. They developed calculus, which includes techniques for solving and manipulating differential equations. The development of calculus was a significant turning point in the history of mathematics and physics, as it allowed scientists and engineers to model and understand the behavior of physical systems.

In the 18th century, Leonhard Euler (1707–1783) and Joseph-Louis Lagrange (1736–1813) made significant contributions to the study of ordinary differential equations (ODEs), which describe the behavior of systems that change with time. They developed techniques for solving ODEs and applied them to problems in physics and engineering.

In the 19th century, the study of partial differential equations (PDEs) began to take shape. PDEs involve derivatives of multiple variables and describe more complex physical phenomena, such as the flow of heat and fluids. Mathematicians such as Augustin-Louis Cauchy (1789–1857), Jean-Baptiste Joseph Fourier (1768–1830), and Pierre-Simon Laplace (1749–1827) made valuable contributions to the study of PDEs, developing techniques for solving and analyzing these equations.

In the 20th century, the theory of differential equations advanced rapidly, with many new techniques being developed for solving ODEs and PDEs. Mathematicians such as Henri Poincaré (1854–1912), George David Birkhoff (1884–1944), Paul Duhamel (1838–1920), and David Hilbert (1862–1943), made important contributions to the field. Today, differential equations are a fundamental tool in many areas of science and engineering, including physics, engineering, finance, and biology. Many techniques and software are available to solve differential equations, and research in the field continues to uncover new applications and solutions.

This thesis deals with the parallel solution of second-order PDEs. Second-order PDEs are important because they are widely used to model many physical and mathematical phenomena. Some examples of second-order PDEs follow. The wave equation describes the behavior of mechanical waves, such as sound and water waves, and is used in acoustics and seismology fields. The heat equation describes how heat is diffused through a material. It plays a crucial role in many areas of physics and engineering, such as thermal analysis, heat transfer, and combustion. The Laplace equation describes the behavior of various phenomena, such as electrostatic potential, fluid flow, and temperature distribution. The Schrödinger equation is fundamental in quantum mechanics, and it is used to describe the time evolution of a quantum mechanical system.

Several techniques can be used to solve PDEs. For example, analytical methods find an exact solution to the PDE using mathematical techniques such as the separation of variables or eigenfunction expansion. These methods are not always possible to apply, especially for more complex PDEs. Numerical methods approximate the solution of PDE using a finite set of discrete values. Widely used numerical methods include, for example, the finite difference method, the finite element method, or the finite volume method. These methods are generally more versatile and applicable to a wider range of PDEs than analytical methods.

In this work, the second-order PDEs are solved using the method of lines (MOL). The MOL is a technique for solving PDEs by converting them into a system of ODEs by discretizing spatial variables. The first references to the MOL are from the turn of the 19th and 20th century, and since then, it has become a standard technique in the numerical solution of PDEs.

The resulting system of ODEs can then be solved using standard numerical methods. The Euler, Runge-Kutta, Adams-Bashforth, and Adams-Moulton methods are the most widely used. This thesis presents the numerical method based on the Taylor series that automatically changes the integration step and the order of a method during the calculation to achieve defined accuracy. It differs from the conventional methods by the possibility of calculating higher-order derivatives and their subsequent use in the calculation.

1.1 Motivation

The parallel implementation of the Taylor series method is rare. One of the reasons is that it typically requires a sequential process to compute the terms of the series. Parallelization of this method can be challenging due to the dependencies between the terms of the series. The parallel high-precision numerical solution of ODEs using high-order Taylor methods is discussed in [14]. The Arenstorf orbits and galactic dynamics problems were tested on a Sun UltraSPARC-II with four processors of 480 MHz using the Message Passing Interface (MPI) and the multiple-precision Fortran library mpf90. The authors noted that this approach is useful only for high-precision demands because there is a high number of inter-process communications. The specialized FPGA-based parallel system for numerical integration is discussed in [98] and [97].

The parallel implementation of the Taylor series method is described in the article [41] focusing on the OpenMP parallelization of the multiple precision Taylor series method using one computational node. The model problem is the chaotic dynamic system – the classical Lorenz system. The article also briefly introduces a CNS (Clean Numerical Simulation) originally published by Shijun Liao [109]. Article [81] introduces a hybrid MPI+OpenMP parallelization strategy for a multiple-precision Taylor series method with fixed step size and fixed order. The authors used up to six nodes (32 cores per node). Finally, the article [82] is based on the previous article [81] and modifies CNS with variable step size and fixed order modification of the method. The optimal integration steps are based on the approach published in [89].

None of those mentioned above articles deals with large-scale systems of ODEs arising from PDEs. Therefore, a parallel higher-order Taylor-series-based algorithm was proposed and implemented.

1.2 Research objectives

The following hypothesis was formulated:
Large-scale systems of ordinary differential equations arising from the second-order linear partial differential equations using the method of lines can be solved more efficiently by the parallel higher-order Taylor series-based numerical method than by conventional numerical methods.
The main objectives of the presented thesis can be summarized as follows:

1. To analyze and evaluate the suitability of the higher-order Taylor series-based method for the solution of ordinary and partial differential equations.
2. To propose, implement and deploy the Taylor series-based parallel method on an HPC cluster.
3. To experimentally evaluate the proposed method using the selected class of the second-order linear partial differential equations discretized in the spatial domain by finite difference formulas of different orders. Compare obtained results with the state-of-the-art numerical methods.
4. Discuss the achieved results and suggest possible extensions.

1.3 Thesis outline

The thesis is organized as follows. Chapter 1 introduces the research area, motivation, and research objectives. Chapter 2 provides an overview of numerical integration methods, including the Euler method, Runge-Kutta methods, Taylor series method, and multistep methods. Chapter 3 is focused on partial differential equations. Types of PDEs are discussed with the methods for solving PDEs. This chapter also provides an overview of the Taylor series-based finite difference formulas and analyzes their accuracy. The next part of this chapter introduces the MOL and its numerical stability. Finally, the transformation methods of higher-order ODEs into a system of first-order ODEs are presented. Chapter 4 introduces the Modern Taylor Series Method (MTSM) and provides the state-of-the-art of the Taylor series method. Recurrent calculation of Taylor series terms, automatic integration order setting, and automatic transformation are discussed. This chapter also presents a linear and nonlinear form of the MTSM, together with practical examples. Finally, a general parallelization method for the linear systems of ODEs is proposed. Chapter 5 is focused on the interconnection networks and topologies used in supercomputers and the challenges of current high-performance interconnection networks. It also summarizes parallel performance metrics and introduces the Berkeley Roofline model. Chapter 6 and Appendix C summarize achieved results. Finally, Chapter 7 includes concluding remarks and discusses possible future work.

Chapter 2

Numerical solution of differential equations

The exact solution of large ODE systems exists only for a few real-world problems. Therefore, the numerical solution is used to obtain approximate solutions. This chapter summarizes the basic knowledge of ODEs and the numerical methods of their solution. In general, it is not possible to select the best method, but it is possible to select the most appropriate methods for a particular class of problems with similar properties. For more details, refer to [30, 69, 74, 75].

An equation containing the derivatives of one or more unknown functions (or dependent variables), with respect to one or more independent variables, is said to be a *differential equation*. A common solution of the differential equation contains an integration constant that may have any value. To specify a unique solution, the value of the function must be defined at a certain point. This value is called an *initial condition*. The formal definition of the initial value problem follows.

Definition 2.0.1. Initial value problem. A first-order initial value problem (IVP) in ordinary differential equations consists of two parts:

1. An ordinary form of a *first-order differential equation* is formulated by

$$F(t, y(t), y'(t)) = 0\,. \tag{2.1}$$

   Equation (2.1) can be written in an explicit form

$$y' = f(t, y(t))\,, \tag{2.2}$$

   where f is continuous in some open set Ω in the (t, y) plane.

2. An initial condition that determines the value of $y(t)$ at a certain point $t = t_0$

$$y(t_0) = y_0\,, \tag{2.3}$$

   where $(t_0, y_0) \in \Omega$.

A *numerical solution* to the initial value problem means finding approximate values of the function $y(t)$ at a finite number of points $a = t_0 < t_1 < \ldots < t_k = b$. These points are called *interpolation points* or *net points*, and the set $I_h = \{t_0, t_1, \ldots, t_k\}$ is called a *net* or *grid*. The distance between two neighboring points $h_i = t_{i+1} - t_i$ is called a *net step length* at the point t_i, or *integration step*. If the step between individual net points is constant, the net is called *equidistant* and $h = \frac{b-a}{n}$.

Definition 2.0.2. Grid (net), integration step size. A grid (or net) is a decomposition I_h of the interval $I = \langle t_0, t_k \rangle$

$$I_h = \{t_0, t_1, \ldots, t_k\},$$

with the points $t_0 < t_1 < \ldots < t_k$ called *interpolation points* or *grid points* or *net points*. Differences between neighboring points $hi = t_{i+1} - t_i$, $i = 0, \ldots, k-1$ are called *integration step sizes*. If the step between individual net points is constant, the net is called *equidistant* and $h = \frac{b-a}{n}$.

Definition 2.0.3. An n^{th} **order differential equation.** The n^{th}-order differential equation is expressed in the explicit form

$$y^{(n)}(t) = F(t, y(t), y'(t), \ldots, y^{(n-1)}), \tag{2.4}$$

the initial conditions are

$$y(t_0) = y_{1,0}, \quad y'(t_0) = y_{2,0}, \ldots, y^{(n-1)}(t_0) = y_{n,0}. \tag{2.5}$$

The n^{th}-order differential equation (2.4) can be transformed into a system of first-order differential equations (2.6) with initial conditions (2.7). When $y_1 = y,\ y_2 = y', \ldots, y_n = y^{(n-1)}$, then $f_j = y_{j+1},\ j = 1, 2, \ldots, n-1$ and $f_n = F(t, y_1(t),\ y_2(t), \ldots,\ y_n(t))$.

Definition 2.0.4. System of the first-order differential equations. The system of first-order differential equations consists of n equations with n unknown functions $y_1(t), y_2(t), \ldots, y_n(t)$ expressed in the form

$$y'_j(t) = f_j(t, y_1(t), y_2(t), \ldots, y_n(t)), \quad j = 1, 2, \ldots, n, \tag{2.6}$$

with initial conditions

$$y_j(t_0) = y_{j,0}, \quad j = 1, 2, \ldots, n. \tag{2.7}$$

Let us assume that each initial value problem, the numeric solution that will be considered, has the only solution. It is known that this precondition of existence and the uniqueness of the solution will be met if the function $f(t, y(t))$ is continuous, bounded, and satisfies the Lipschitz condition.

Definition 2.0.5. Lipschitz condition. A function $f(t, y)$ satisfies a Lipschitz condition in the variable y on a set $D \subset \mathbb{R}^2$ if a constant $L > 0$ exists with

$$|f(t, y_1) - f(t, y_2)| \leq L\,|y_1 - y_2|\,,$$

whenever $(t, y_1), (t, y_2)$ are in D and L is the Lipschitz constant.

Analytical solution of large systems of ordinary differential equations is complex and, in many cases, impossible. Therefore, a numerical solution is even more used in practice. *Numerical method* for solving the initial value problem (2.0.1) is a formula for the gradual calculation of the approximations $y_1, y_2, \ldots, y_k$, $y_0 = const$. The methods for the numerical solution of differential equations can be divided according to the manner of calculation: into one-step and multistep, explicit and implicit methods.

Definition 2.0.6. One-step method. A one-step method for the calculation of an approximation y_{i+1} of the solution of the initial value problem (2.0.1) has the form

$$y_{i+1} = y_i + h_i \Phi(t_i, y_i, y_{i+1}, h_i; f),$$

where Φ is the so-called incremental function of four variables t_i, y_i, y_{i+1}, h_i and depends on the function $f(t, y)$.

Definition 2.0.7. Multistep method. Multistep methods (m-step methods) use the previously calculated values $y_i, y_{i-1}, \ldots, y_{i+1-k}$ to calculate the approximation y_{i+1}. The general formula is

$$y_{i+1} = h \sum_{j=0}^{m} b_j f(t_{i+1-k}, y_{i+1-k}) - \sum_{j=1}^{m} a_j y_{i+1-j} \,.$$

If the coefficient is $b_0 = 0$, the method is *explicit*. If $b_0 \neq 0$, the method is *implicit*, and we have to solve equation (2.8) to calculate the value of y_{i+1}:

$$y_{i+1} = \varphi(y_{i+1}), \quad \text{where} \quad \varphi(z) = hb_0 f(t_{i+1}, z) + \sum_{j=1}^{m} (hb_j f(t_{i+1-j}, y_{i+1-j}) - a_j y_{i+1-j}) \,. \tag{2.8}$$

Definition 2.0.8. Explicit and implicit methods. Let us $y(t_i)$ denote the solution of IVP (2.0.1) at the node t_i and y_i a numerical approximation of $y(t_i)$. A numerical method for the solution of IVP on a grid I_h is called *explicit*, if an approximation y_{i+1} in t_{i+1} can be calculated directly from already computed values y_j, $j \leq i$. Otherwise, the method is called *implicit*.

Definition 2.0.9. Linear differential equation of the order n. A linear differential equation of order n is an equation of the form

$$P_n(t)y^{(n)} + P_{n-1}(t)y^{(n-1)} + \ldots + P_1(t)y' + P_0(t)y = Q(t) \,,$$

where each P_k and Q is a function of the independent variable t, and $y^{(k)}$ denotes the k^{th} derivative of y with respect to t.

Definition 2.0.10. Homogeneous linear differential equation of order n. A homogeneous linear differential equation of order n is an equation of the form

$$P_n(t)y^{(n)} + P_{n-1}(t)y^{(n-1)} + \ldots + P_1(t)y' + P_0(t)y = 0 \,.$$

2.1 Adaptive-step-size numerical methods

To approximate continuous behavior with rapid changes, the *fixed-step-size* numerical methods must choose a small step size to maintain predefined accuracy and obtain satisfying results. The disadvantage is that the small step size results in a very high computational cost. In contrast, the *adaptive-step-size* numerical methods can use larger step sizes to obtain approximations more efficiently. The idea of adaptive step-size techniques is to adapt the trajectory of the approximation by estimating and controlling the error at each step. These error estimates are used to dynamically increase or decrease the step size [52, 71]. The main advantage of adaptive-steps-size numerical methods is the possibility of defining a user's *error tolerance* to balance desired precision and computational efficiency.

Let us assume that the IVP 2.0.1 is used to define the approximation errors. Consider that we have computed the values of the solution $y_1, y_2, \ldots, y_i$ at the points $t_1, t_2, \ldots, t_k, t_i = T$, respectively, to approximate $y(T)$ for $T > 0$. The exact values are denoted as $y(t_1)$, $y(t_2)$, $\ldots$, $y(t_i)$. For one-step methods, the computation of y_{i+1} is based on the value of y_i. For fixed-step-size methods, the step size (see definition 2.0.2) $h_i = t_{i+1} - t_i$ is equal for all h_i, $i = 0, \ldots, i-1$, for adaptive-step-size methods, the step size can differ.

Two sources of errors must be considered when using adaptive-step-size methods. Firstly, *round-off errors*, caused by the limited length of the word on the computer where the value of the number is stored.

Secondly, *approximation errors* caused by deviations between the approximations y_1, $y_2, \ldots, y_i$ and the exact values $y(t_1), y(t_2), \ldots, y(t_i)$. If we omit round-off errors, the *global error* is the total error in the computed solution at a certain point and shows how far the computed solution is from the original solution curve. The global error has two components, the global error at t_i and its propagation to t_{i+1} and the approximation error of the last step called *local error*. Let us denote the *local solution* u as $u'(t) = f(t, u(t))$ with the initial value $u(t_i) = y_i$.

Definition 2.1.1. Global error. The global error at t_{i+1} can be defined as

$$\begin{aligned}\epsilon^g_{i+1} &= y(t_{i+1}) - y_{i+1} \\ &= (y(t_{i+1}) - u(t_{i+1})) + (u(t_{i+1}) - y_{i+1})\,.\end{aligned}$$

Definition 2.1.2. Local error. The local error at t_{i+1} can be defined as

$$\epsilon^l_{i+1} = u(t_{i+1}) - y_{i+1}\,.$$

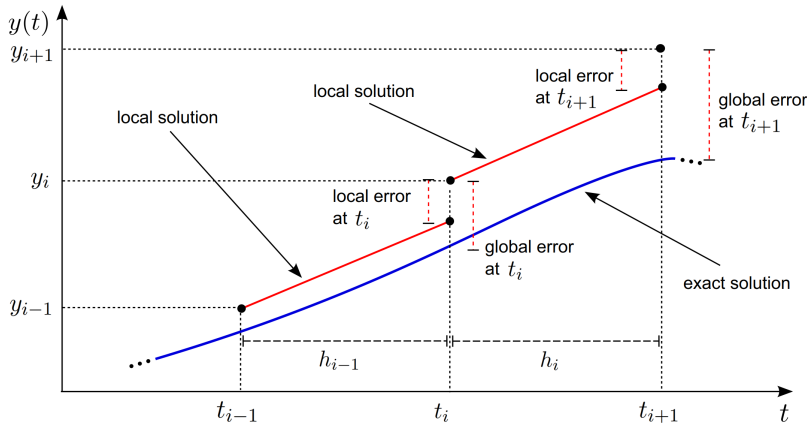


Figure 2.1: Local and global errors in a numerical approximation [52].

2.1.1 Adapting the step size

It is impossible to control the global error directly, but we can indirectly control the local error in each time step. The local in each time step is bound by the user-defined error tolerance τ. There are two common definitions *error per step*, where $|\epsilon^l_{i+1}| \leq \tau$ and *error per unit step*, where $|\epsilon^l_{i+1}| \leq \tau \cdot h_i$, for each $i \geq 0$.

The local error depends on the *order* of the numerical method, which determines how fast a sequence of approximations generated by a method converges toward the expected solution. The higher the order, the better the approximation [52].

Consistency is the study of local error. A numerical method is called *consistent* if the local error decays sufficiently fast as $h \longrightarrow 0$.

Definition 2.1.3. Consistency. The numerical method is said to be consistent with the differential equation it approximates if

$$\lim_{h \longrightarrow 0} |\epsilon^l_i(h)| = 0\,,$$

where ϵ_i^l is the difference between the numerical method and the differential equation at t_i.

A numerical method is called *convergent* if the absolute error between the numerical and analytical solution decreases with decreasing h.

Definition 2.1.4. Convergence. The numerical method is said to be convergent with respect to the differential equation if

$$\lim_{h \longrightarrow 0} |y_i - y(t_i)| = 0\,,$$

where $y(t_i)$ is the exact solution evaluated at t_i and y_i is the approximation at the same point.

Definition 2.1.5. Stability. The numerical method is stable if small changes in the initial condition produce correspondingly small changes in the resulting approximations.

Stability considerations are important because, in each step after the first step of a numerical technique, we start over again with a new initial-value problem, where the initial condition is the approximate solution value computed in the preceding step. This value will almost certainly vary at least slightly from the true value of the solution because of the presence of round-off errors. Another common source of error occurs in the initial condition itself. In physical applications, data are often obtained by imprecise measurements. For more details, see [57, 181].

Definition 2.1.6. Autonomous differential equation. A differential equation of the form (2.2) is called autonomous if the right-hand side f is not explicitly dependent on t, that is,

$$y' = F(y)\,.$$

Each differential equation can be transformed into an autonomous differential equation. This is called **autonomization**

$$Y = \begin{pmatrix} y \\ t \end{pmatrix}, \quad F(Y) = \begin{pmatrix} f(t,y) \\ 1 \end{pmatrix}, \quad Y' = F(Y)\,.$$

A method which provides the same solution for the autonomous differential equation as for the original IVP, is called **invariant under autonomization**.

2.2 Taylor series methods

First, the terms such as the Taylor polynomial and Taylor series will be introduced.

Definition 2.2.1. Taylor polynomial. Let the function f have all derivatives up to the n^{th} order at point x_0. Then, the n^{th} Taylor polynomial for f of the n^{th} degree with center x_0 can be defined as follows:

$$P_n(t) = f(x_0) + f'(x_0)(t-x_0) + \frac{f''(x_0)}{2!}(t-x_0)^2 + \cdots + \frac{f^{(n)}x_0}{n!}(t-x_0)^n = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(t-x_0)^k\,. \tag{2.9}$$

If we plug $t = t_i + h = t_{i+1}$ and $x_0 = t_i$ into (2.9), we obtain (explicit) *Taylor series*.

Definition 2.2.2. Taylor series. Taylor series can be defined as follows:

$$y_{i+1} = y_i + hy_i' + \frac{h^2}{2!}y_i'' + \cdots + \frac{h^n}{n!}y_i^{(n)}\,, \tag{2.10}$$

where $h \in \mathbb{R}$ is an integration step and $n \in \mathbb{N}$ is the order of the method.

Taylor polynomial (2.9) is the basis of one-step numerical methods and provides the most accurate approximation of the function. The problem is to obtain a higher derivative, but if Taylor series methods are effectively implemented, they can be used to solve general problems. *Local truncation error due to neglecting of higher terms of Taylor series*, is the error made by the numerical method in one step. For more information, see [29].

Definition 2.2.3. Local discretization error. General differential method

$$y_0 = y(t_0)\,, \tag{2.11}$$
$$y_{i+1} = y_i + h\varphi(t_i, y_i), \quad i = 0, 1, \ldots, k\,, \tag{2.12}$$

where $\varphi(t_i, y_i)$ is the function that depends on the particular differential method, the function has the local discretization error at each step:

$$\tau_{i+1}(h) = \frac{y_{i+1} - (y_i + h\varphi(t_i, y_i))}{h} = \frac{y_{i+1} - y_i}{h} - \varphi(t_i, y_i), \quad i = 0, 1, \ldots, k\,, \tag{2.13}$$

where $y_i = y(t_i)$ is the analytical solution.

Derivation of higher-order Taylor series methods
Consider the IVP

$$y'(t) = f(t, y), \quad a \le t \le b, \quad y(a) = \alpha\,. \tag{2.14}$$

Expand $y(t)$ in the n^{th} Taylor polynomial about t_i and evaluate at t_{i+1}:

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \cdots + \frac{h^n}{n!}y^{(n)}(t_i) + \frac{h^{n+1}}{(n+1)!}y^{(n+1)}(\xi_i)\,, \tag{2.15}$$

where $\xi_i \in (t_i, t_{i+1})$. By deriving the solution of $y(t)$ we get $y'(t) = f(t, y(t))$, $y''(t) = f'(t, y(t))$, $\ldots$, $y^{(n)}(t) = f^{(n-1)}(t, y(t))$. Substituting into equation (2.15), we get:

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \cdots + \frac{h^n}{n!}f^{(n-1)}(t_i, y(t_i)) + \frac{h^{n+1}}{(n+1)!}f^{(n)}(\xi_i, y(\xi_i))\,. \tag{2.16}$$

The n^{th} order Taylor series method is obtained by omitting the remainder term.

Definition 2.2.4. Taylor method of order n. Taylor series method of order n is defined as:

$$y_0 = \alpha\,, \tag{2.17}$$
$$y_{i+1} = y_i + hT^{(n)}(t_i, y_i), \quad i = 0, 1, \ldots, k\,, \tag{2.18}$$

where

$$T^{(n)}(t_i, y_i) = f(t_i, y_i) + \frac{h}{2}f'(t_i, y_i) + \cdots + \frac{h^{n-1}}{n!}f^{(n-1)}(t_i, y_i)\,. \tag{2.19}$$

Remark 2.2.1. The Euler method is the Taylor method of order one.

Lemma 2.2.1. Taylor's Theorem. Suppose $f \in C^n[a,b]$, that $f^{(n+1)}$ exists in $[a,b]$ and $x_0 \in [a,b]$. For every $x \in [a,b]$, there exists a number $\xi(x)$ between x and x_0 with

$$f(x) = P_n(x) + R_n(x)\,, \tag{2.20}$$

and

$$R_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)^{(n+1)}\,, \tag{2.21}$$

where $P_n(x)$ is the Taylor polynomial (2.9) for f about x_0 and $R_n(x)$ is called the remainder term (or truncation error) associated with $P_n(x)$.

Proof. See [87]. □

Lemma 2.2.2. If the Taylor series method of order n is used for the approximation of the IVP 2.0.1 with the integration step h and if $y \in C^{n+1}[a,b]$, then the local discretization error is $O(h^n)$.

Proof. Equation (2.9) can be rewritten in the form

$$y_{i+1} - y_i - hf(t_i,y_i) - \frac{h^2}{2}f'(t_i,y_i) - \cdots - \frac{h^n}{n!}f^{(n-1)}(t_i,y_i) = \frac{h^{n+1}}{(n+1)!}f^{(n)}(\xi_i, y(\xi_i))\,, \tag{2.22}$$

where $\xi \in (t_i, t_{i+1})$. Because $y \in C^{n+1}[a,b]$, we have $y^{(n+1)}(t) = f^{(n)}(t,y(t))$ in the interval $[a,b]$ and $\tau_{i+1} = O(h^n)$ for $i = 1, 2, \cdots, k+1$. Local discretization error is

$$\tau_{i+1}(h) = \frac{y_{i+1} - y_i}{h} - T^{(n)}(t_i,y_i) = \frac{h^n}{(n+1)!}f^{(n)}(\xi_i, y(\xi_i)), \quad i = 0,1,\ldots,k\,. \tag{2.23}$$

□

2.3 Euler method

Let us assume the initial value problem with an equidistant net $\{t_0, t_1, \ldots, t_k\}$ and step $h = t_{i+1} - t_i$, where $i = 0, \ldots, k-1$. In all net points hold that

$$y'(t_i) = f(t_i, y(t_i))\,. \tag{2.24}$$

The derivative on the left side of equation (2.24) can be expressed as

$$\frac{y(t_{i+1}) - y(t_i)}{h} \doteq f(t_i, y(t_i))\,. \tag{2.25}$$

By replacing $y(t_i)$ with the approximate value y_i, it is possible to express an approximate value $y(t_{i+1})$ and we obtain the *explicit Euler method*:

$$y_{i+1} = y_i + hf(t_i, y_i)\,. \tag{2.26}$$

The approximate solution at the next net point is calculated using the value of the previous. The *implicit Euler method* is defined as:

$$y_{i+1} = y_i + hf(t_{i+1}, y_{i+1})\,. \tag{2.27}$$

Because the unknown y_{i+1} is the argument of the function $f(t,y)$, it is not possible to obtain the explicit formula. Therefore, we have to solve the following equation:

$$g(z) := z - y_i - hf(t_{i+1}, z) = 0\,, \tag{2.28}$$

where the solution z is y_{i+1}. Equation (2.28) can be solved using iterative methods (for example, the Newton method).

2.4 Runge-Kutta methods

Runge-Kutta methods are an important group of one-step methods. The general form of the *implicit* Runge-Kutta method is

$$y_{i+1} = y_i + h \sum_{j=1}^{s} b_j k_j \,, \tag{2.29}$$

where coefficients k_j are defined as

$$k_j = f(t_i + hc_j, y_i + h \sum_{l=1}^{s} a_{jl} k_l), \quad j = 1, 2, \ldots, s \,, \tag{2.30}$$

and b_j, c_j, and a_{jl} are constants determined to reach the maximal order of the method. For *explicit* Runge-Kutta methods hold, $a_{jl} = 0$ for $l \geq j$, therefore, the coefficients k_j can be calculated as

$$k_1 = f(t_i, y_i) \,, \tag{2.31}$$

$$k_j = f(t_i + hc_j, y_i + h \sum_{l=1}^{j-1} a_{j,l} k_l), \quad j = 2, 3, \ldots, s \,. \tag{2.32}$$

Definition 2.4.1. Butcher tableau. The Runge-Kutta scheme can be written in the general form of a tableau, the so-called Butcher tableau [30]

$$\begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,s} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \quad \Longrightarrow \quad \begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array} \tag{2.33}$$

where $\mathbf{c}$ are the nodes, $\mathbf{A}$ is the matrix of the method, and $\mathbf{b}$ are weights.

Explicit Runge-Kutta methods are characterized by a strictly lower triangular matrix $\mathbf{A}$, i.e., $a_{ij} = 0$ if $j \geq i$. Moreover, the following condition holds

$$c_i = \sum_{j=1}^{s} a_{i,j}, \quad i = 1, 2, \ldots, s \,. \tag{2.34}$$

The maximal order $p(s)$ of the Runge-Kutta method with s stages depends on s as follows:

$$\begin{aligned} &p(s) = s, \quad \text{for } s = 1, 2, 3, 4, \\ &p(s) = s - 1, \quad \text{for } s = 5, 6, 7, \\ &p(s) = s - 2, \quad \text{for } s = 8, 9, \\ &p(s) \leq s - 2, \quad \text{for } s = 10, 11, \ldots \,. \end{aligned} \tag{2.35}$$

An overview of the well-known Runge-Kutta methods follows.

Runge-Kutta method $p = s = 1$

The Runge-Kutta method of the first order is the explicit Euler method. In this case, the Butcher tableau is of the form

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \qquad y_{i+1} = y_i + hf(t_i, y_i)\,. \tag{2.36}$$

Runge-Kutta method $p = s = 2$

There are three variants of the Runge-Kutta method of second order. The first is the *explicit midpoint method*

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array} \qquad \begin{aligned} y_{i+1} &= y_i + hk_2\,, \\ k_1 &= f(t_i, y_i)\,, \\ k_2 &= f(t_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}hk_1)\,. \end{aligned} \tag{2.37}$$

The second variant is known as the *modified Euler's method* or *Heun's method*

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \qquad \begin{aligned} y_{i+1} &= y_i + \tfrac{1}{2}h(k_1 + k_2)\,, \\ k_1 &= f(t_i, y_i)\,, \\ k_2 &= f(t_i + h, y_i + hk_1)\,. \end{aligned} \tag{2.38}$$

The third one is the *Ralston's method* with a minimum local error bound

$$\begin{array}{c|cc} 0 & & \\ \frac{2}{3} & \frac{2}{3} & \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array} \qquad \begin{aligned} y_{i+1} &= y_i + h\left(\tfrac{1}{4}k_1 + \tfrac{3}{4}k_2\right)\,, \\ k_1 &= f(t_i, y_i)\,, \\ k_2 &= f(t_i + \tfrac{2}{3}h, y_i + \tfrac{2}{3}hk_1)\,. \end{aligned} \tag{2.39}$$

Runge-Kutta method $p = s = 3$

The Butcher table for Runge-Kutta with three stages is

$$\begin{array}{c|ccc} 0 & & & \\ c_2 & c_2 & & \\ c_3 & c_3 - a_{3,2} & a_{3,2} & \\ \hline & b_1 & b_2 & b_3 \end{array} \tag{2.40}$$

Different variants of Runge-Kutta methods of the third order exist. One of the most popular is the *Ralston's third-order method*, used in the embedded Bogacki–Shampine method

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{3}{4} & 0 & \frac{3}{4} & \\ \hline & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} \end{array} \qquad \begin{aligned} y_{i+1} &= y_i + \tfrac{1}{9}h(2k_1 + 3k_2 + 4k_3)\,, \\ k_1 &= f(t_i, y_i)\,, \\ k_2 &= f(t_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}hk_1)\,, \\ k_3 &= f(t_i + \tfrac{3}{4}h, y_i + \tfrac{3}{4}hk_2)\,. \end{aligned} \tag{2.41}$$

Runge-Kutta method $p = s = 4$

The most widely used Runge-Kutta method of the fourth order is *classic Runge-Kutta method*

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \qquad \begin{aligned} y_{i+1} &= y_i + \tfrac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)\,, \\ k_1 &= f(t_i, y_i)\,, \\ k_2 &= f(t_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}hk_1)\,, \\ k_3 &= f(t_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}hk_2)\,, \\ k_4 &= f(t_i + h, y_i + hk_3)\,. \end{aligned} \tag{2.42}$$

2.4.1 Embedded methods

The idea of embedded methods is to combine two Runge-Kutta methods of different order but with the same stages. The first method is of order p, and the second is of order $p+1$. Both methods use the same coefficients a_{ij} and c_i, but the coefficients b_i differ. The coefficients can be written in the extended Butcher's tableau.

The local truncation error can be estimated as the difference between an approximation of order p and one of order $\hat{p}$ (usually $\hat{p} = p + 1$). This estimation can then be used for automatic step size selection. The Butcher tableau is extended as follows,

$$\begin{array}{c|ccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,s} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s} \\ \hline & b_1 & b_2 & \cdots & b_s \\ \hline & \hat{b}_1 & \hat{b}_2 & \cdots & \hat{b}_s \end{array} \quad \Longrightarrow \quad \begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \\ \hline & \hat{\mathbf{b}}^T \end{array} \tag{2.43}$$

where $\hat{\mathbf{b}}$ are the coefficients for the higher-order method. The common stages are

$$k_j = f(t_i + hc_j, y_i + h\sum_{l=1}^{s} a_{j,l}k_l), \quad j = 1, 2, \ldots, s\,, \tag{2.44}$$

the first method of order p

$$y_{i+1} = y_i + h\sum_{j=1}^{s} b_j k_j\,, \tag{2.45}$$

the second method of order $\hat{p} = p + 1$

$$\hat{y}_{i+1} = y_i + h\sum_{j=1}^{s} \hat{b}_j k_j\,, \tag{2.46}$$

the local error estimation, which is used for automatic step size selection, is

$$\varepsilon_i = |\hat{y}_{i+1} - y_{i+1}| = h\sum_{j=1}^{s}(\hat{b}_j - b_j)k_j\,, \quad j = 1, 2, \ldots, s\,. \tag{2.47}$$

The first method is *Runge-Kutta-Fehlberg* 5(4), a fifth-order scheme with a fourth-order embedded scheme with six stages. The method is defined by the following tableau [56]

$$
\begin{array}{c|cccccc}
0 & & & & & & \\
\frac{1}{4} & \frac{1}{4} & & & & & \\
\frac{3}{8} & \frac{3}{32} & \frac{9}{21} & & & & \\
\frac{12}{13} & \frac{1932}{2197} & -\frac{7200}{2197} & -\frac{7296}{2197} & & & \\
1 & \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & & \\
\frac{1}{2} & -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & \\
\hline
 & \frac{25}{216} & 0 & \frac{1408}{2565} & \frac{2197}{4104} & -\frac{1}{5} & 0 \\
\hline
 & \frac{16}{135} & 0 & \frac{6656}{12825} & \frac{28561}{5630} & -\frac{9}{50} & \frac{2}{55}
\end{array}
\tag{2.48}
$$

Dormand and Prince Runge-Kutta method of order 5(4), fifth-order scheme with a fourth-order embedded scheme [44] and seven stages follows

$$
\begin{array}{c|ccccccc}
0 & & & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & & \\
\frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & & & & \\
\frac{8}{9} & \frac{19372}{6561} & \frac{-25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & & & \\
1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & & \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & \\
\hline
 & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & -\frac{92097}{339200} & \frac{187}{2100} & \frac{1}{40} \\
\hline
 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0
\end{array}
\tag{2.49}
$$

The higher-order approximation $\hat{y}_{i+1}$ is usually used as an approximation of the solution. The value $\hat{y}_i$ is used in (2.45) and (2.46) instead of y_i. This approach is called *local extrapolation*.

Notice that the last row of $\mathbf{A}$ in Butcher's table 2.49 is the same as the $\hat{b}_i$ coefficients. This property is called *First Same As Last (FSAL)*. This means that the last stage k_7 used in the error estimation can be reused in the next time step as stage k_1. Therefore, the number of function evaluations of the Runge-Kutta-Fehlberg method is identical, although the Dormand and Prince method has one more stage.

Bogacki-Shampine Runge-Kutta method of order 3(2), third-order scheme with a second-order embedded scheme, and it has four stages [26]

$$
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{3}{4} & 0 & \frac{3}{4} & & \\
1 & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & \\
\hline
 & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & 0 \\
 & \frac{7}{24} & \frac{1}{4} & \frac{1}{3} & \frac{1}{8}
\end{array}
\qquad
\begin{aligned}
y_{i+1} &= y_i + \tfrac{2}{9}hk_1 + \tfrac{1}{3}hk_2 + \tfrac{4}{9}hk_3\,, \\
k_1 &= f(t_i, y_i)\,, \\
k_2 &= f(t_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}hk_1)\,, \\
k_3 &= f(t_i + \tfrac{3}{4}h, y_i + \tfrac{3}{4}hk_2)\,, \\
k_4 &= f(t_i + h, y_{i+1})\,, \\
z_{i+1} &= y_i + \tfrac{7}{24}hk_1 + \tfrac{1}{4}hk_2 + \tfrac{1}{3}kh_3 + \tfrac{1}{4}hk_4\,,
\end{aligned}
\tag{2.50}
$$

where z_{i+1} is a second-order approximation of the exact solution. The value of y_{i+1} is calculated according to Ralston's method (see Butcher's table 2.41), and it is a third-order approximation. The difference between values z_{i+1} and y_{i+1} can be used to adapt the step size. Thanks to the FSAL property, only three function evaluations have to be performed per step in this case.

Another Runge-Kutta method is *Bogacki-Shampine* Runge-Kutta method of order 5(4) [25], a fifth-order scheme with a fourth-order embedded scheme, and it has eight stages.

Other embedded Runge-Kutta methods are methods by J. Verner [167]. For example, *Verner* Runge-Kutta method of order 6(5), a sixth-order scheme with fifth-order embedded scheme and nine stages, *Verner* Runge-Kutta method of order 7(6), a seventh-order scheme with sixth-order embedded scheme and ten stages, *Verner* Runge-Kutta method of order 8(7), an eighth-order scheme with seventh-order embedded scheme and thirteen stages.

Many tools use embedded methods with automatic step size control to solve ODEs. For example, MATLAB software [158] provides two embedded Runge-Kutta methods, namely, ode23 is the implementation of the Bogacki-Shampine 3(2) method and ode45 is the implementation of the Dormand-Prince 5(4) method.

Portable, Extensible Toolkit for Scientific Computation (PETSc) library [2, 10, 11, 13] provides embedded methods similar to MATLAB, TSRK3BS implements Bogacki-Shampine 3(2) method, and TSRK5DP implements Dormand-Prince 5(4) method. Furthermore, the PETSc library offers implementations of the Bogacki-Shampine Runge-Kutta method of order 5(4) (TSRK5BS) [25]. Verner Runge-Kutta methods of orders 6(5), 7(6) and 8(7) are implemented as TSRK6VR, TSRK7VR, and TSRK8VR respectively.

Trapezoidal methods are one-step implicit Runge-Kutta methods of the second order defined as

$$y_{i+1} = y_i + \frac{1}{2}h[f(t_i, y_i) + f(t_{i+1}, y_{i+1})]\,. \tag{2.51}$$

In MATLAB, this method is the basis for the ode23t function. Another method is ode23tb, an implicit Runge-Kutta formula with a trapezoidal rule step as its first stage and a backward differentiation formula of order two as its second stage.

2.5 Multistep methods

As mentioned above, multi-step methods (m-step methods) use previously calculated values $y_i, y_{i-1}, \ldots, y_{i+1-m}$ to calculate the approximation y_{i+1}. The well-known families of linear multistep methods are the Adams-Bashforth methods, the Adams-Moulton methods, and the backward differentiation formulas (BDFs).

2.5.1 Adams–Bashforth methods

The general form of the Adams–Bashforth methods is

$$y_{i+1} = y_i + h\sum_{j=1}^{m} b_j f(t_{i+1-j}, y_{i+1-j})\,. \tag{2.52}$$

The coefficients for the number of steps $m = 1, \ldots, 6$ and the fixed step size h are in Table 2.1. For $m = 1$, we obtain the explicit Euler method (2.26), for $m = 2$, we obtain the multistep method:

$$y_{i+1} = y_i + h\frac{3}{2}f(t_i, y_i) - \frac{1}{2}f(t_{i-1}, y_{i-1})\,. \tag{2.53}$$

m	b_1	b_2	b_3	b_4	b_5	b_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 2.1: Coefficients of the Adams-Bashfort method.

2.5.2 Adams–Moulton methods

The general form of Adams–Moulton methods is

$$y_{i+1} = y_i + h \sum_{j=0}^{m} b_j f(t_{i+1-j}, y_{i+1-j}) \,. \tag{2.54}$$

Because the term $hb_0 f(t_{i+1}, y_{i+1})$ appears on the right-hand side of (2.54), these methods are called *implicit Adams-Moulton methods*. The coefficients for the number of steps $m = 1, \ldots, 6$ and the fixed step size h are in Table 2.2. For $m = 1$, we obtain the implicit Euler method (2.27), and for $m = 2$, we obtain the trapezoidal method. For $m \geq 3$, the multistep method is obtained.

m	b_0	b_1	b_2	b_3	b_4	b_5
1	1					
2	$\frac{1}{2}$	$\frac{1}{2}$				
3	$\frac{5}{12}$	$\frac{8}{12}$	$-\frac{1}{12}$			
4	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$		
5	$\frac{251}{720}$	$\frac{646}{720}$	$-\frac{264}{720}$	$\frac{106}{720}$	$-\frac{19}{720}$	
6	$\frac{475}{1440}$	$\frac{1427}{1440}$	$-\frac{798}{1440}$	$\frac{482}{1440}$	$-\frac{173}{1440}$	$\frac{27}{1440}$

Table 2.2: Coefficients of the Adams-Moulton method.

The problem of a multistep method with obtaining m starting values $y_0, y_1, \ldots, y_{m-1}$ can be solved using *self-starting* algorithms. For example, for m-step Adams-Bashfort methods, the values $y_1, y_2, \ldots, y_{m-1}$ are calculated using Adams-Bashfort methods with steps $1, 2, \ldots m-1$. The combination of Adams-Bashfort and Adams-Moulton methods is often known as predictor-corrector methods.

2.5.3 Predictor-corrector methods

The predictor-corrector method uses a combination of explicit and implicit methods. Typically, the explicit method is used for the predictor step, and the implicit method for the corrector step. This scheme is also called PECE (Predict-Evaluate-Correct-Evaluate) scheme. For example, the PECE scheme can be constructed from the explicit Adams-Bashfort and

implicit Adams-Moulton methods. The PECE scheme of the second order is defined as follows:

$$\begin{aligned}
\hat{y}_{i+1} &= y_i + \tfrac{1}{2}h(3f_i - f_{i-1}) && \text{(P) Adams-Bashfort } 2^{nd} \text{ order}\,,\\
\hat{f}_{i+1} &= f(t_{i+1}, \hat{y}_{i+1}) && \text{(E) Adams-Bashfort } 2^{nd} \text{ order}\,,\\
y_{i+1} &= y_i + \tfrac{1}{2}h(\hat{f}_{i+1} + f_i) && \text{(C) Adams-Moulton } 2^{nd} \text{ order}\,,\\
f_{i+1} &= f(t_{i+1}, y_{i+1}) && \text{(E) Adams-Moulton } 2^{nd} \text{ order}\,.
\end{aligned} \tag{2.55}$$

The PECE scheme (2.55) is implemented in MATLAB as the ode113 function. The algorithms based on predictor-corrector methods often use variable step and variable order (VSVO) during the computation.

2.5.4 Backward differentiation methods

The general formula for the m-step backward differentiation method is:

$$\begin{aligned}
\alpha_{m,0}y_{i+1} + \alpha_{m,1}y_i + \cdots + \alpha_{m,m}y_{i+1-m} &= hf(t_{i+1}, y_{i+1})\,,\\
\alpha_{m,j} &= hl'_j(t_{i+1})\,,\\
l_j(t) &= \prod_{n=1, n\neq j}^{k} \frac{t - t_{i+1-n}}{t_{i+1-j} - t_{i+1-n}}\,.
\end{aligned} \tag{2.56}$$

The coefficients for the number of steps $m = 1, \ldots, 6$ and the fixed step size h are in Table 2.3. The backward differentiation methods are m-step implicit methods. For $m = 1$, we obtain the implicit Euler method (2.27).

m	$\alpha_{k,0}$	$\alpha_{k,1}$	$\alpha_{k,2}$	$\alpha_{k,3}$	$\alpha_{k,4}$	$\alpha_{k,5}$	$\alpha_{k,6}$
1	1	-1					
2	$\frac{3}{2}$	-2	$\frac{1}{2}$				
3	$\frac{11}{6}$	-3	$\frac{3}{2}$	$-\frac{1}{3}$			
4	$\frac{25}{12}$	-4	3	$-\frac{4}{3}$	$\frac{1}{4}$		
5	$\frac{137}{60}$	-5	5	$-\frac{10}{3}$	$\frac{5}{4}$	$-\frac{1}{5}$	
6	$\frac{147}{60}$	-6	$\frac{15}{12}$	$-\frac{20}{3}$	$\frac{15}{4}$	$-\frac{6}{5}$	$\frac{1}{6}$

Table 2.3: Coefficients of backward differentiation methods.

Backward differentiation methods for $m = 1, \ldots, 5$ are implemented in MATLAB as a function ode15s. It is the VSVO method.

Chapter 3

Partial differential equations

Partial differential equations can be encountered when solving various technical and physical problems. Therefore, some partial differential equations are referred to as *equations of mathematical physics*. These equations describe some physical phenomena (with a certain extent and accuracy). Depending on the dimension of the space in which the investigated process takes place, the function $u(x,t), u(x,y,t), u(x,y,z,t)$, or $u(x_1,x_2,\ldots,x_N,t)$ in the case of higher dimensions, if appropriate, will be considered as unknown. Other significant areas of the application of partial differential equations include, for example, electrodynamics, dynamics of fluids, cosmology, unbalanced static mechanics, quantum mechanics, and others. Basic types of second-order PDEs include parabolic, hyperbolic, and elliptic PDE.

A solution to a partial differential equation is any function defined in the given domain, including its partial derivatives, up to the order of the equation and satisfying the given equation in the specified domain. Partial differential equations are discussed in publications [50, 55, 106].

Partial differential equations were discovered in the 18th century, as ordinary differential equations failed to describe the studied physical principles. Many famous mathematicians developed the subject of partial differential equations. Leonard Euler and Joseph-Louis Lagrange studied the waves on strings. Daniel Bernoulli, Leonard Euler, Adrien-Marie Legendre, and Pierre-Simon Laplace focused on potential theory and Joseph Fourier on series expansions for the heat equation. The discovery of fundamental partial differential equations for a given process has contributed significantly to advances in modern science. For example, James Clerk Maxwell's equations for electromagnetic theory provide a solution for problems in radio wave propagation and the diffraction of light, and Erwin Schrodinger's equation for quantum mechanical processes at the atomic level, which changed atomic physics and chemistry in the 20th century. Navier-Stokes equations describe the motion of viscous fluid substances and form a basis for widely disparate topics such as weather forecasting and the design of supersonic aircrafts. The study of partial differential equations is extensive and belongs to several areas of mathematics [7, 27, 49].

This chapter deals with partial differential equations, especially second-order PDEs. The method of lines is introduced to solve time-dependent PDEs. The spatial derivatives are replaced by finite differences, and a PDE is transformed into a system of ODEs [146].

3.1 Types of partial differential equations

Definition 3.1.1. Partial differential equations of n^{th} order. Partial differential equations of n^{th} order are equations of the following form:

$$F\left(x_1, x_2, \ldots, x_n, u, \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_n}, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \ldots, \frac{\partial^2 u}{\partial x_1 \partial x_n}, \frac{\partial^2 u}{\partial x_2^2}, \ldots, \frac{\partial^k u}{\partial x_n^k}\right) = 0\,, \quad (3.1)$$

where $u(x_1, x_2, \ldots, x_n)$ is a sought function of independent n variables. The order of a partial differential equation is given by the order of the highest derivative contained in such an equation.

Definition 3.1.2. Linear partial differential equations. If equation (3.1) is linear with respect to the sought function and its derivatives, it is called a linear partial differential equation.

Definition 3.1.3. Homogeneous linear partial differential equations. A linear partial differential equation is called *homogeneous* if it contains no dependent variable and its partial derivatives. Otherwise, the PDE is called *non-homogeneous*.

Definition 3.1.4. Second-order partial differential equations. The equation in the form

$$F\left(x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial x \partial y}, \frac{\partial^2 u}{\partial y^2}\right) = 0 \quad (3.2)$$

is called the *second-order partial differential equation* for the unknown function $u = u(x, y)$.

Definition 3.1.5. Solution of a partial differential equation. The solution to PDE (3.1) in domain $\Omega \subset \mathbb{R}^N$ is any function that possesses all partial derivatives of continuous type and which, after having been substituted together with the derivatives to (3.1), is satisfactory for all $(x_1, \ldots, x_n) \in \Omega$ in this equation.

Let us assume a common homogeneous linear PDE of second order in a plane (i.e., $u = u(x, y)$) where $A, \ldots, F$ are real numbers (these are constant coefficients)

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu = 0\,, \quad (3.3)$$

depending on the values of the coefficients $A, \ldots, F$, these equations are *elliptic*, *hyperbolic*, and *parabolic*:

- *elliptic* if $B^2 - 4AC < 0$,
- *parabolic* if $B^2 - 4AC = 0$,
- *hyperbolic* if $B^2 - 4AC > 0$.

The following text deals with partial equations of the second order (see Definition 3.1.4).

Parabolic PDE

The parabolic partial differential equation describes heat transfer. Let us assume an isolated rod with a length of L, and its cross section is neglected. The rod is placed along the axis x so that its left end merges with the beginning. The variable $u = u(x,t)$ describes the temperature of the rod at the point x and at time t. We can show that this function satisfies the partial differential equation (3.4)

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2} + f(x,t), \quad (x,t) \in (0,L) \times (0,T)\,, \tag{3.4}$$

where $a^2 = \frac{k}{\rho c}$ is the thermal diffusivity, k is the thermal conductivity coefficient, ρ is the specific mass and c is the specific heat. The function $f(x,t)$ characterizes the intensity of internal sources (if, for example, the rod is under voltage, it produces heat), and T is the time duration of the investigated process. There are infinitely many solutions to equation (3.4). If we want to unambiguously determine the rod temperature at any time and location, one initial condition and two boundary conditions must be added to the equation. The initial condition (3.5) describes the rod temperature at the beginning of the process

$$u(x,0) = g(x), \quad 0 < x < L\,. \tag{3.5}$$

The boundary conditions (3.6) and (3.7) characterize the situation at both ends of the rod throughout the process. They describe the situation where the left end of the rod is kept at temperature h_1 and the right end of the bar is kept at temperature h_2. The function $g(x)$ describes the temperature of the rod at the beginning of the process (3.5)

$$u(0,t) = h_1(t), \quad 0 < t < T, \tag{3.6}$$
$$u(L,t) = h_2(t), \quad 0 < t < T. \tag{3.7}$$

Hyperbolic PDE

The hyperbolic partial differential equation describes the propagation of waves. It is applied in various fields such as mechanics (description of strings or fluids), acoustics, optics, or electromagnetism. Let us assume a perfectly flexible string of length L. This string is anchored to the axis x and tensioned with force F. Equation (3.8) describes a vibrating string

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2} + f(x,t), \quad (x,t) \in (0,L) \times (0,T)\,, \tag{3.8}$$

where $a^2 = \frac{F}{\rho}$, and ρ is a specific mass of the string per unit length, F is the tension force, $u(x,t)$ is a vertical deflection from the balance position of the string at point x, and at time t and $f(x,t)$ expresses the other external load, if any (e.g., gravitation). Naturally, initial conditions (3.9) that describe an initial deflection and an initial string velocity must be considered in the equation (3.8)

$$u(x,0) = g_1(x), \quad \frac{\partial u}{\partial t}(x,0) = g_2(x), \quad 0 < x < L\,. \tag{3.9}$$

The boundary conditions that reflect the behavior of the string at its anchor points can also be described using equations (3.6) a (3.7). Depending on the dimension, the heat

conduction equations, and the wave equation can be expressed in the form (3.10)

$$\begin{aligned} \frac{\partial u}{\partial t} &= a^2 \Delta u + f, \\ \frac{\partial^2 u}{\partial t^2} &= a^2 \Delta u + f\,. \end{aligned} \tag{3.10}$$

The symbol Δ is called the *Laplace operator* defined below.

Definition 3.1.6. Hence, in the region of variables $x_1, \ldots, x_n$, in general, the Laplace operator has the following form

$$\Delta = \frac{\partial^2}{\partial x_1^2} + \cdots + \frac{\partial^2}{\partial x_n^2}\,. \tag{3.11}$$

Elliptic PDE

For time-stable models, the heat conduction equations and the wave equation formally have the same form:

$$-\Delta u = f\,, \tag{3.12}$$

where f is a function of variables $x_1, \ldots, x_n$. Equation (3.12) is called *Poisson equation*. In a two-dimensional space where $n = 2$, the equation can be expressed in the form (3.13)

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f(x, y)\,. \tag{3.13}$$

Laplace equation is a special case of Poisson equation (3.12)

$$-\Delta u = 0\,. \tag{3.14}$$

Equation (3.14) can be written in a two-dimensional space in the form (3.15)

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = 0\,. \tag{3.15}$$

There are many other interpretations of Laplace and Poisson equations. For example, Poisson equation describes the fluid flow potential through a layer with variable width, where u denotes the potential of the velocity vector, and f denotes a variable thickness of the fluid layer. If the layer has a constant width, the function f also equals zero, and the given problem can be described using Laplace equation.

Analytical solving of partial differential equations is difficult and, in many cases, practically impossible, particularly when solving boundary value problems in higher-order equations. In some special cases, the partial differential equations can be solved using the Fourier method, which assumes that such a solution can be obtained using separated variables. That is why this method is also called the method of separation of variables. The method can be applied when solving many linear partial differential equations with an initial condition, e.g., heat conduction, wave, and Laplace equations. The following sections focus on the numerical solution of partial differential equations.

3.2 Numerical solution of PDEs

Various methods exist in the literature for the solution of partial differential equations. The most well-known methods are the finite difference method (FDM), finite element method (FEM), finite volume method (FVM), method of lines (MOL), and boundary element method (BEM).

The basic idea of the *finite difference method* (*method of grids*) is to discretize the continuous domain into a discrete finite-difference grid. The individual partial derivatives in the partial differential equations are approximated by algebraic finite difference approximations, which are substituted into the partial differential equations to obtain an algebraic finite difference equation. The algebraic finite difference equations are solved for the dependent variable. For more details on the finite difference method, see [55, 106, 153].

The *finite volume method* represents and evaluates partial differential equations in the form of algebraic equations. Similarly to FDM, the values are calculated at discrete places on a meshed geometry. The finite volume method converts volume integrals in a partial differential equation containing divergence terms to surface integrals using the divergence theorem. The divergence terms are evaluated as fluxes at the surfaces of each finite volume. The method is used in many fluid dynamics or problems with body-fitted coordinate systems. For more information on the finite volume method, refer to [51, 107, 172].

The *finite element method* divides a given domain into the collection of subdomains, the so-called finite element mesh, and the solution in each element is constructed from the basis functions. The actual equations that are solved are typically obtained by restating the conservation equation in the weak form: the field variables are written in terms of the basis functions, the equation is multiplied by appropriate test functions, and then integrated over an element. For more information on the finite element method, see [83, 100, 138, 139, 180].

The *boundary element method* is an efficient alternative to FDM and FEM to solve PDEs. This method incorporates a mesh that is located only on the boundaries of the domain. The advantage of BEM over the FEM is that there is no need for discretizing the domain under consideration into the elements. This method only requires the boundary data as input, resulting in smaller system matrices. The disadvantage over the FEM is that the BEM matrices are fully populated with complex and frequency-dependent coefficients, which deteriorate the efficiency of the solution. Moreover, singularities may arise in the solution. For more details, see [47, 178].

The *method of lines* is a general procedure for solving time-dependent PDEs, and the method consists of two steps. The first step includes approximating the spatial derivatives using FDM, FVM, or FEM (or any other technique). In the second step, the time integration of the resulting semi-discrete ODEs (discrete in space, continuous in time) is performed [175]. The MOL is described in more detail in Section 3.4. The following section deals with the Taylor series-based finite difference approximations.

3.3 Taylor series based finite difference approximations

Finite difference formulas can approximate derivatives of any order at any point using a sufficient number of surrounding points. The derivative is calculated as the slope at a given point using the values and relative locations of the surrounding points. The locations of the sample points are called *finite difference stencil*. There are three types of finite differential formulas, and they can differ in the derivative approximated, stencil type, and size:

- forward difference formulas,
- backward difference formulas, and
- central difference formulas.

The finite differential formulas can be obtained from the Taylor series expansion of the derivatives around the node of interest. Let us assume that the derivative of the function f at the point x is defined as the limit of a difference quotient:

$$f'(x) = \lim_{\Delta x \longrightarrow 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} \,. \tag{3.16}$$

The finite difference equation $\frac{f(x+\Delta x)-f(x)}{\Delta x}$ is an approximation of the derivative $f'(x)$. The expression $f(x + \Delta x) - f(x)$ is the finite difference, and Δx denotes the spacing between the points. If $\Delta x > 0$, where Δx is a finite positive number, then

$$f'(x) = \frac{f(x + \Delta x) - f(x)}{\Delta x} + O(\Delta x) \tag{3.17}$$

is called *first-order forward finite difference formula* and corresponds to the finite difference stencil $\{0, 1\}$. Similarly, if $\Delta x < 0$, where $\Delta x > 0$, then:

$$f'(x) = \frac{f(x) - f(x - \Delta x)}{\Delta x} + O(\Delta x) \tag{3.18}$$

is called *first-order backward finite difference formula* and corresponds to the finite difference stencil $\{-1, 0\}$. By the combination of forward and backward finite difference formulas, the *second-order central difference formula* can be derived and corresponds to the finite difference stencil $\{-1, 0, 1\}$

$$f'(x) = \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} + O(\Delta x^2) \,. \tag{3.19}$$

The smaller spatial step Δx results in a better approximation. The error of the approximation depends on Δx. The finite difference formulas mentioned above and the corresponding truncation errors are derived in Section 3.3.1.

3.3.1 Derivation of truncation errors

Truncation errors rare the errors resulting from using an approximation instead of an exact mathematical procedure. It is known that the n-times continuously differentiable function $f(x)$ can be replaced by the Taylor polynomial $T_n(x)$ in the neighborhood $U(a)$ of point a, with a reminder $R_n(x)$ as follows:

$$f(x) = T_n(x) + R_n(x) \quad \forall x \in U(a) \,, \tag{3.20}$$

where $T_n(x)$ is the Taylor polynomial defined in (3.21):

$$T_n(x) = f(a)+f'(a)(x-a)+\frac{f''(a)}{2!}(x-a)^2+\cdots+\frac{f^{(n)}a}{n!}(x-a)^n = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k \,. \tag{3.21}$$

From the above relationship, the condition for the equality of the Taylor series of function $f(x)$ and function $f(x)$ is derived. Let the function $f(x)$ have derivatives of all orders in the interval I and let $a \in I$ be an internal point of the interval I. Then the following applies to this interval

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k \,, \tag{3.22}$$

if and only if

$$\lim_{n\to\infty} R_n(x) = 0 \quad \forall x \in I \,. \tag{3.23}$$

To verify the relationship (3.23), the *Lagrange form of remainder* $R_n(x)$ is used as follows:

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-a)^{n+1} \,. \tag{3.24}$$

The subscript n of R_n denotes that the remainder is for the nth-order approximation and ξ is an unspecified value of x, and it applies to $\xi \in (a; x)$. The Lagrange form of the remainder can be used for the analysis of the truncation errors. It is convenient to write the Taylor series in the following form by defining a step size $h = x - a$:

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}h^k + R_n \,, \tag{3.25}$$

where the remainder term R_n is

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1} \,. \tag{3.26}$$

An error term can also be expressed using the *Omikron (Big O)* notation. The higher the power of h in the error term, the smaller the error because $h \ll 1$. For such numbers, the following applies: $h > h^2 > h^3 > \ldots$. Equation (3.25) can be written in the form (3.27):

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}h^k + O(h^{n+1}) \,. \tag{3.27}$$

Similarly, truncation errors for any difference formulas can be derived. The following text describes the derivation of the backward, forward, and central finite difference formulas in detail.

Backward and forward finite difference formulas

First, the *first-order backward finite difference formula* is derived. The finite difference stencil $\{-1, 0\}$ corresponds to the points $x - \Delta x$ and x. The node of interest is point x. The following Taylor series (3.28) is obtained:

$$f(x-\Delta x) = f(x) - f'(x)\Delta x + \frac{f''(x)}{2!}\Delta x^2 - \frac{f'''(x)}{3!}\Delta x^3 + \cdots \,. \tag{3.28}$$

After moving the first term expression $f'(x)\Delta x$ to the left-hand side of the equation,

$$f'(x) = \frac{1}{\Delta x}\left[f(x) - f(x-\Delta x) + \frac{f''(x)}{2!}\Delta x^2 - \frac{f'''(x)}{3!}\Delta x^3 + \cdots\right] , \tag{3.29}$$

and

$$f'(x) = \frac{1}{\Delta x} f(x) - \frac{1}{\Delta x} f(x - \Delta x) + \frac{f''(x)}{2!}\Delta x - \frac{f'''(x)}{3!}\Delta x^2 + \cdots . \tag{3.30}$$

Neglecting the terms $\left[\frac{f''(x)}{2!}\Delta x - \frac{f'''(x)}{3!}\Delta x^2 + \cdots\right]$ results in a truncation error, caused by neglecting a certain number in terms of the Taylor series. To obtain an error term $O(\Delta x)$, the greatest term of the expression is taken, which is the term $\frac{f''(x_i)}{2!}\Delta x$. Note that the truncation error is proportional to Δx to the power of 1. Therefore, $\frac{f(x)-f(x-\Delta x)}{\Delta x}$ is called the *first-order backward finite difference* approximation of $f'(x)$. Equation (3.30) can be rewritten in the form with the remainder as stated in (3.26):

$$f'(x) = \frac{1}{\Delta x} f(x) - \frac{1}{\Delta x} f(x - \Delta x) + \frac{f''(\xi)}{2!}\Delta x\,, \quad \xi \in (x - \Delta x; x)\,. \tag{3.31}$$

The resulting relationship is expressed by (3.32)

$$f'(x) = \frac{f(x) - f(x - \Delta x)}{\Delta x} + O(\Delta x)\,. \tag{3.32}$$

Similarly, a *first-order forward finite difference formula* can be derived. In this case, the finite difference stencil $\{0, 1\}$ corresponds to the points $x, x+\Delta x$. The resulting relationship is expressed by (3.33)

$$f'(x) = \frac{f(x + \Delta x) - f(x)}{\Delta x} + O(\Delta x)\,. \tag{3.33}$$

Central difference formulas

Finally, a *central finite difference formula* is derived. The finite difference stencil is $\{-1, 0, 1\}$, corresponding to the points $x - \Delta x, x, x + \Delta x$. At first, Taylor series for the points $x - \Delta x$ (3.34) and $x + \Delta x$ (3.35) are constructed

$$f(x - \Delta x) = f(x) - f'(x)\Delta x + \frac{f''(x)}{2!}\Delta x^2 - \frac{f'''(x)}{3!}\Delta x^3 + \cdots \tag{3.34}$$

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + \frac{f''(x)}{2!}\Delta x^2 + \frac{f'''(x)}{3!}\Delta x^3 + \cdots . \tag{3.35}$$

Based on the above equations, the term $-f(x)$ is expressed as

$$-f(x) = -f(x_{i-1}) - f'(x)\Delta x + \frac{f''(x)}{2!}\Delta x^2 - \frac{f'''(x)}{3!}\Delta x^3 + \cdots \tag{3.36}$$

$$-f(x) = -f(x_{i+1}) + f'(x)\Delta x + \frac{f''(x)}{2!}\Delta x^2 + \frac{f'''(x)}{3!}\Delta x^3 + \cdots . \tag{3.37}$$

By subtracting (3.36) from (3.37), equation (3.38) is obtained

$$0 = -f(x + \Delta x) + f(x - \Delta x) + 2f'(x)\Delta x + 2\frac{f'''(x)}{3!}\Delta x^3 + \cdots . \tag{3.38}$$

After solving the equation for $-f(x)$, we obtain (3.39)

$$-f'(x) = \frac{1}{2\Delta x}\left[f(x - \Delta x) - f(x + \Delta x) + 2\frac{f'''(x)}{3!}\Delta x^3 + \cdots\right], \tag{3.39}$$

and

$$f'(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + \frac{f'''(x)}{3!}\Delta x^2 + \cdots . \tag{3.40}$$

Equation (3.40) can be rewritten in the form with the remainder according to (3.26):

$$f'(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + \frac{f'''(\xi)}{3!}\Delta x^2 . \tag{3.41}$$

The error term is $O(\Delta x^2)$ is derived from the remainder term $\frac{f'''(\xi)}{3!}\Delta x^2$ and the truncation error is proportional to Δx to the power of 2 and, therefore, (3.42) is called *second-order central finite difference formula*

$$f'(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + O(\Delta x^2) . \tag{3.42}$$

3.3.2 Higher-order finite difference formulas

In the following text, the fourth-order finite difference formulas are derived. For each point of interest, four Taylor series are constructed, and four neighboring points are used for the calculation. It is possible to obtain derivatives of the first to fourth order at each point. The matrix-vector notation to calculate finite difference coefficients is:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} . \tag{3.43}$$

The matrix $\mathbf{A}$ represents the matrix of coefficients obtained from the Taylor series. The vector $\mathbf{b}$ represents the values of differences between known samples. Let the samples be $f_{-l}, \ldots, f_{-1}, f_0, f_1, f_k$ at the points $-l\Delta x, \ldots, -\Delta x, 0, \ldots, k\Delta x$, and $n = l+k+1$ is the total number of stencil points. For given parameters (l, k), the matrix $\mathbf{A}^{-1}$ remains constant, and the vector $\mathbf{b}$ changes. Then, the following Taylor series polynomials can be used:

$$\forall i \ \in -l, -l+1, \ldots, -1, 0, 1, \ldots, k-1, k : f_i = f_0 + \sum_{m=1}^{n} \frac{i^m (\Delta x)^m}{m!} f^{(m)}(0). \tag{3.44}$$

Equation (3.44) can be written in matrix form:

$$\mathbf{b} = \begin{pmatrix} f_{-l} - f_0 \\ f_{-l+1} - f_0 \\ \cdots \\ f_{-1} - f_0 \\ f_1 - f_0 \\ \cdots \\ f_{k-1} - f_0 \\ f_k - f_0 \end{pmatrix} , \tag{3.45}$$

$$\mathbf{A} = \begin{pmatrix} (-l)^1 & (-l)^2 & \ldots & (-l)^n \\ (-l+1)^1 & (-l+1)^2 & \ldots & (-l+1)^n \\ \cdots & \cdots & \cdots & \cdots \\ (-1)^1 & (-1)^2 & \ldots & (-1)^n \\ (1)^1 & (1)^2 & \ldots & (1)^n \\ \cdots & \cdots & \cdots & \cdots \\ (k-1)^1 & (k-1)^2 & \ldots & (k-1)^n \\ (k)^1 & (k)^2 & \ldots & (k)^n \end{pmatrix} , \tag{3.46}$$

$$\mathbf{x} = \begin{pmatrix} f^{(1)}(0) \\ f^{(2)}(0) \\ \vdots \\ f^{(n)}(0) \end{pmatrix}. \tag{3.47}$$

Note that the finite difference coefficients in Tables 3.2, A.1, and A.2 correspond to the grid spacing $\Delta x = 1$. For other values of Δx, the coefficients must be divided by $(\Delta x)^m$, where m is the order of the derivative [60].

Central difference formulas

The *fourth-order central difference formula* corresponds to the finite difference stencil $\{-2, -1, 0, 1, 2\}$ and the points $x - 2\Delta x, x - \Delta x, x, x + \Delta x, x + 2\Delta x$. To approximate the derivatives at point x, four Taylor series have to be constructed. Figure 3.1 illustrates the five-point finite difference stencil. The point of interest x_i is marked in black, and the neighboring points are marked in red. For simplification, the points $x-2\Delta x, x-\Delta x, x, x+\Delta x, x+2\Delta x$ are denoted as $x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}$.

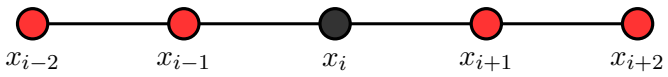


Figure 3.1: Five-point finite difference stencil for the fourth-order central finite difference formula.

Taylor series (3.48), (3.49), (3.50), and (3.51) for the neighborhood of point x_i follows:

$$\begin{aligned} f(x_{i-2}) &= f(x_i) + (-2\Delta x_i) f(x_i)' + \frac{(-2\Delta x_i)^2}{2!} f(x_i)'' + \frac{(-2\Delta x_i)^3}{3!} f(x_i)''' + \\ &+ \frac{(-2\Delta x_i)^4}{4!} f(x_i)^{(4)} + \frac{(-2\Delta x_i)^5}{5!} f^{(5)}(\xi_1) \end{aligned} \tag{3.48}$$

$$\begin{aligned} f(x_{i-1}) &= f(x_i) + (-\Delta x_i) f(x_i)' + \frac{(-\Delta x_i)^2}{2!} f(x_i)'' + \frac{(-\Delta x_i)^3}{3!} f(x_i)''' + \\ &+ \frac{(-\Delta x_i)^4}{4!} f(x_i)^{(4)} + \frac{(-\Delta x_i)^5}{5!} f^{(5)}(\xi_2) \end{aligned} \tag{3.49}$$

$$\begin{aligned} f(x_{i+1}) &= f(x_i) + \Delta x_i f(x_i)' + \frac{\Delta x_i^2}{2!} f(x_i)'' + \frac{\Delta x_i^3}{3!} f(x_i)''' + \\ &+ \frac{\Delta x_i^4}{4!} f(x_i)^{(4)} + \frac{(\Delta x_i)^5}{5!} f^{(5)}(\xi_3) \end{aligned} \tag{3.50}$$

$$\begin{aligned} f(x_{i+2}) &= f(x_i) + 2\Delta x_i f(x_i)' + \frac{(2\Delta x_i)^2}{2!} f(x_i)'' + \frac{(2\Delta x_i)^3}{3!} f(x_i)''' + \\ &+ \frac{(2\Delta x_i)^4}{4!} f(x_i)^{(4)} + \frac{(2\Delta x_i)^5}{5!} f^{(5)}(\xi_4)\,. \end{aligned} \tag{3.51}$$

The system of equations can be expressed in the matrix vector notation (3.52). Subsequently, we obtain the first four terms of the Taylor series denoted as $DX1, DX2, DX3, DX4$

$$\begin{pmatrix} DX1 \\ DX2 \\ DX3 \\ DX4 \end{pmatrix} = \begin{pmatrix} -2 & (-2)^2 & (-2)^3 & (-2)^4 \\ -1 & (-1)^2 & (-1)^3 & (-1)^4 \\ 1 & 1^2 & 1^3 & 1^4 \\ 2 & 2^2 & 2^3 & 2^4 \end{pmatrix}^{-1} \cdot \begin{pmatrix} x_{i-2} - x_i \\ x_{i-1} - x_i \\ x_{i+1} - x_i \\ x_{i+2} - x_i \end{pmatrix}, \tag{3.52}$$

therefore,

$$\begin{pmatrix} DX1 \\ DX2 \\ DX3 \\ DX4 \end{pmatrix} = \begin{pmatrix} \frac{1}{12} & -\frac{2}{3} & \frac{2}{3} & -\frac{1}{12} \\ -\frac{1}{24} & \frac{2}{3} & \frac{2}{3} & -\frac{1}{24} \\ -\frac{1}{12} & \frac{1}{6} & -\frac{1}{6} & \frac{1}{12} \\ \frac{1}{24} & -\frac{1}{6} & -\frac{1}{6} & \frac{1}{24} \end{pmatrix} \cdot \begin{pmatrix} x_{i-2} - x_i \\ x_{i-1} - x_i \\ x_{i+1} - x_i \\ x_{i+2} - x_i \end{pmatrix}. \tag{3.53}$$

To obtain the values of derivatives x_i', x_i'', x_i''' a $x_i^{(4)}$, the relationships (3.54), (3.55), (3.56), and (3.57) are used

$$x_i' = \frac{DX1}{h} \tag{3.54}$$

$$x_i'' = \frac{DX2}{\frac{h^2}{2!}} \tag{3.55}$$

$$x_i''' = \frac{DX3}{\frac{h^3}{3!}} \tag{3.56}$$

$$x_i^{(4)} = \frac{DX4}{\frac{h^4}{4!}}. \tag{3.57}$$

To approximate the first derivative $f'(x)$:

$$f'(x) \approx \frac{\frac{1}{12}x_{i-2} - \frac{2}{3}x_{i-1} + \frac{2}{3}x_{i+1} - \frac{1}{12}x_{i+2}}{\Delta x} + O(\Delta x^4). \tag{3.58}$$

To approximate the second derivative $f''(x)$:

$$f''(x) \approx \frac{-\frac{1}{12}x_{i-2} + \frac{4}{3}x_{i-1} - \frac{5}{2}x_i + \frac{4}{3}x_{i+1} - \frac{1}{12}x_{i+2}}{\Delta x^2} + O(\Delta x^4). \tag{3.59}$$

To approximate the third derivative $f'''(x)$:

$$f'''(x) \approx \frac{-\frac{1}{2}x_{i-2} + 1x_{i-1} - 1x_{i+1} - \frac{1}{2}x_{i+2}}{\Delta x^3} + O(\Delta x^2). \tag{3.60}$$

To approximate the fourth derivative $f^{(4)}(x)$:

$$f^{(4)}(x) \approx \frac{x_{i-2} - 4x_{i-1} + 6x_i - 4x_{i+1} - x_{i+2}}{\Delta x^4} + O(\Delta x^2). \tag{3.61}$$

For the given n-point stencil and type of difference formula, the inverse matrix in equation (3.52) is constant. The coefficients of the central difference formulas for five-point approximation are in Table 3.1.

The coefficients of the central difference formulas are shown in Table 3.2. The M ≥ 0 denotes the order of the derivative to approximate, and N is the number of grid points at x-coordinates, O denotes the order of accuracy, the node of interest $x = 0$. Note that the

coefficients are symmetric around the point $x = 0$. The coefficients have the opposite sign for odd derivatives, whereas the signs are the same for even derivatives. Recall that the coefficients in Tables 3.1 and 3.2 correspond to the grid spacing $\Delta x = 1$. For other values of Δx, the coefficients have to be divided by $(\Delta x)^m$, where m is the order of the derivative. An error term determines the order of accuracy.

M	**O**	**stencil x-coordinates**				
		−2	**−1**	**0**	**1**	**2**
1	4	$\frac{1}{12}$	$-\frac{2}{3}$	0	$\frac{2}{3}$	$-\frac{1}{12}$
2	4	$-\frac{1}{12}$	$\frac{4}{3}$	$-\frac{5}{2}$	$\frac{4}{3}$	$-\frac{1}{12}$
3	2	$-\frac{1}{2}$	1	0	−1	$\frac{1}{2}$
4	2	1	−4	6	−4	1

Table 3.1: Table of coefficients for central difference formulas.

M	**O**	**stencil x-coordinates**								
		−4	**−3**	**−2**	**−1**	**0**	**1**	**2**	**3**	**4**
1	**2**				$-\frac{1}{2}$	0	$\frac{1}{2}$			
	4			$\frac{1}{12}$	$-\frac{2}{3}$	0	$\frac{2}{3}$	$-\frac{1}{12}$		
	6		$-\frac{1}{60}$	$\frac{3}{20}$	$-\frac{3}{4}$	0	$\frac{3}{4}$	$-\frac{3}{20}$	$\frac{1}{60}$	
	8	$\frac{1}{280}$	$-\frac{4}{105}$	$\frac{1}{5}$	$-\frac{4}{5}$	0	$\frac{4}{5}$	$-\frac{1}{5}$	$\frac{4}{105}$	$-\frac{1}{280}$
2	**2**				1	−2	1			
	4			$-\frac{1}{12}$	$\frac{4}{3}$	$-\frac{5}{2}$	$\frac{4}{3}$	$-\frac{1}{12}$		
	6		$\frac{1}{90}$	$-\frac{3}{20}$	$\frac{3}{2}$	$-\frac{49}{18}$	$\frac{3}{2}$	$-\frac{3}{20}$	$\frac{1}{90}$	
	8	$-\frac{1}{560}$	$\frac{8}{315}$	$-\frac{1}{5}$	$\frac{8}{5}$	$-\frac{205}{72}$	$\frac{8}{5}$	$-\frac{1}{5}$	$\frac{8}{315}$	$-\frac{1}{560}$
3	**2**			$-\frac{1}{2}$	1	0	−1	$-\frac{1}{2}$		
	4		$\frac{1}{8}$	−1	$\frac{13}{8}$	0	$-\frac{13}{8}$	1	$-\frac{1}{8}$	
	6	$-\frac{7}{240}$	$\frac{3}{10}$	$-\frac{169}{120}$	$\frac{61}{30}$	0	$-\frac{61}{30}$	$\frac{169}{120}$	$-\frac{3}{10}$	$\frac{7}{240}$
4	**2**			1	−4	6	−4	1		
	4		$-\frac{1}{6}$	2	$-\frac{13}{2}$	$\frac{28}{3}$	$-\frac{13}{2}$	2	$-\frac{1}{6}$	
	6	$\frac{7}{240}$	$-\frac{2}{5}$	$\frac{169}{60}$	$-\frac{122}{15}$	$\frac{91}{8}$	$-\frac{122}{15}$	$\frac{169}{60}$	$-\frac{2}{5}$	$\frac{7}{240}$

Table 3.2: Coefficients for central finite difference formulas, $M = 4$, $N = 8$, $x_0 = 0$.

Forward difference formulas

The *fourth-order forward difference formula* corresponds to the finite difference stencil $\{0, 1, 2, 3, 4\}$ and the points $x, x + \Delta x, x + 2\Delta x, x + 3\Delta x, x + 4\Delta x$. To approximate the derivatives at point x, four Taylor series have to be constructed. Figure 3.2 illustrates the five-point finite difference stencil. The point of interest x_i is marked in black, and the

neighboring points are marked in red. For simplification, the points $x, x+\Delta x, x+2\Delta x, x+3\Delta x, x+4\Delta x$ are denoted as $x_i, x_{i+1}, x_{i+2}, x_{i+3}, x_{i+4}$.

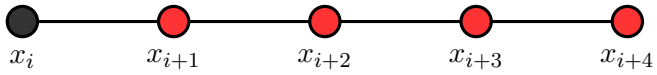


Figure 3.2: Five-point finite difference stencil for the fourth-order forward finite difference formula.

Taylor series (3.62), (3.63), (3.64), and (3.65) for the neighborhood of point x_i follows:

$$\begin{aligned} f(x_{i+1}) &= f(x_i) + \Delta x_i f(x_i)' + \frac{\Delta x_i^2}{2!} f(x_i)'' + \frac{\Delta x_i^3}{3!} f(x_i)''' + \\ &+ \frac{\Delta x_i^4}{4!} f(x_i)^{(4)} + \frac{(\Delta x_i)^5}{5!} f^{(5)}(\xi_1) \end{aligned} \tag{3.62}$$

$$\begin{aligned} f(x_{i+2}) &= f(x_i) + 2\Delta x_i f(x_i)' + \frac{(2\Delta x_i)^2}{2!} f(x_i)'' + \frac{(2\Delta x_i)^3}{3!} f(x_i)''' + \\ &+ \frac{(2\Delta x_i)^4}{4!} f(x_i)^{(4)} + \frac{(2\Delta x_i)^5}{5!} f^{(5)}(\xi_2) \end{aligned} \tag{3.63}$$

$$\begin{aligned} f(x_{i+3}) &= f(x_i) + 3\Delta x_i f(x_i)' + \frac{(3\Delta x_i)^2}{2!} f(x_i)'' + \frac{(3\Delta x_i)^3}{3!} f(x_i)''' + \\ &+ \frac{(3\Delta x_i)^4}{4!} f(x_i)^{(4)} + \frac{(3\Delta x_i)^5}{5!} f^{(5)}(\xi_3) \end{aligned} \tag{3.64}$$

$$\begin{aligned} f(x_{i+4}) &= f(x_i) + 4\Delta x_i f(x_i)' + \frac{(4\Delta x_i)^2}{2!} f(x_i)'' + \frac{(4\Delta x_i)^3}{3!} f(x_i)''' + \\ &+ \frac{(2\Delta x_i)^4}{4!} f(x_i)^{(4)} + \frac{(4\Delta x_i)^5}{5!} f^{(5)}(\xi_4)\,. \end{aligned} \tag{3.65}$$

Equation (3.66) represents the matrix-vector notation:

$$\begin{pmatrix} DX1 \\ DX2 \\ DX3 \\ DX4 \end{pmatrix} = \begin{pmatrix} 1 & 1^2 & 1^3 & 1^4 \\ 2 & 2^2 & 2^3 & 2^4 \\ 3 & 3^2 & 3^3 & 3^4 \\ 4 & 4^2 & 4^3 & 4^4 \end{pmatrix}^{-1} \cdot \begin{pmatrix} x_{i+1} - x_i \\ x_{i+2} - x_i \\ y_{k+3} - x_i \\ y_{k+4} - x_i \end{pmatrix}, \tag{3.66}$$

hence,

$$\begin{pmatrix} DX1 \\ DX2 \\ DX3 \\ DX4 \end{pmatrix} = \begin{pmatrix} 4 & -3 & \frac{4}{3} & -\frac{1}{4} \\ -\frac{13}{3} & \frac{19}{4} & -\frac{7}{3} & \frac{11}{24} \\ \frac{3}{2} & -2 & \frac{7}{6} & -\frac{1}{4} \\ -\frac{1}{6} & \frac{1}{4} & -\frac{1}{6} & \frac{1}{24} \end{pmatrix} \cdot \begin{pmatrix} x_{i+1} - x_i \\ x_{i+2} - x_i \\ x_{i+3} - x_i \\ x_{i+4} - x_i \end{pmatrix}. \tag{3.67}$$

To approximate the first derivative $f'(x)$:

$$f'(x) \approx \frac{-\frac{25}{12}x_i + 4x_{i+1} - 3x_{i+2} + \frac{4}{3}x_{i+3} - \frac{1}{4}x_{i+4}}{\Delta x} + O(\Delta x^4)\,. \tag{3.68}$$

To approximate the second derivative $f''(x)$:

$$f''(x) \approx \frac{\frac{35}{12}x_i - \frac{26}{3}x_{i+1} + \frac{19}{2}x_{i+2} - \frac{14}{3}x_{i+3} + \frac{11}{12}x_{i+4}}{\Delta x^2} + O(\Delta x^3)\,. \tag{3.69}$$

To approximate the third derivative $f'''(x)$:

$$f'''(x) \approx \frac{-\frac{5}{2}x_i + 9x_{i+1} - 12x_{i+2} + 7x_{i+3} - \frac{3}{2}x_{i+4}}{\Delta x^3} + O(\Delta x^2)\,. \tag{3.70}$$

To approximate the fourth derivative $f^{(4)}(x)$:

$$f^{(4)}(x) \approx \frac{1x_i - 4x_{i+1} + 6x_{i+2} - 4x_{i+3} + 1x_{i+4}}{\Delta x^4} + O(\Delta x)\,. \tag{3.71}$$

The coefficients of the forward difference formulas for five-point approximation are in Table 3.3.

M	**O**	**stencil x-coordinates**				
		0	**1**	**2**	**3**	**4**
1	4	$-\frac{25}{12}$	4	-3	$\frac{4}{3}$	$-\frac{1}{4}$
2	3	$\frac{35}{12}$	$-\frac{26}{3}$	$\frac{19}{2}$	$-\frac{14}{3}$	$\frac{11}{12}$
3	2	$-\frac{5}{2}$	9	-12	7	$-\frac{3}{2}$
4	1	1	-4	6	-4	1

Table 3.3: Table of coefficients for forward difference formulas.

The coefficients of the forward difference formulas for different stencils are in Table A.1 in Appendix A.

Backward difference formulas

The *fourth-order backward difference formula* corresponds to the finite difference stencil $\{-4, -3, -2, -1, 0\}$, and the points $x, x - \Delta x, x - 2\Delta x, x - 3\Delta x, x - 4\Delta x$. To approximate the derivatives at point x, four Taylor series have to be constructed. Figure 3.3 illustrates the five-point finite difference stencil. The point of interest x_i is marked in black, and the neighboring points are marked in red. For simplification, the points $x, x - \Delta x, x - 2\Delta x, x - 3\Delta x, x - 4\Delta x$ are denoted as $x_i, x_{i-1}, x_{i-2}, x_{i-3}, x_{i-4}$.

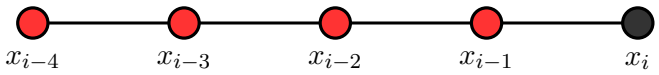


Figure 3.3: Five-point finite difference stencil for fourth-order backward finite difference formula

The resulting matrix-vector notation (3.72) to obtain the first four terms of the Taylor series follows:

$$\begin{pmatrix} DX1 \\ DX2 \\ DX3 \\ DX4 \end{pmatrix} = \begin{pmatrix} -4 & (-4)^2 & (-4)^3 & (-4)^4 \\ -3 & (-3)^2 & (-3)^3 & (-3)^4 \\ -2 & (-2)^2 & (-2)^3 & (-2)^4 \\ -1 & (-1)^2 & (-1)^3 & (-1)^4 \end{pmatrix}^{-1} \cdot \begin{pmatrix} x_{i-4} - x_i \\ x_{i-3} - x_i \\ x_{i-2} - x_i \\ x_{i-1} - x_i \end{pmatrix}, \tag{3.72}$$

therefore,

$$\begin{pmatrix} DX1 \\ DX2 \\ DX3 \\ DX4 \end{pmatrix} = \begin{pmatrix} \frac{1}{4} & -\frac{4}{3} & 3 & -4 \\ \frac{11}{24} & -\frac{7}{3} & \frac{19}{4} & -\frac{13}{3} \\ \frac{1}{4} & -\frac{7}{6} & 2 & -\frac{3}{2} \\ \frac{1}{24} & -\frac{1}{6} & \frac{1}{4} & -\frac{1}{6} \end{pmatrix} \cdot \begin{pmatrix} x_{i-4} - x_i \\ x_{i-3} - x_i \\ x_{i-2} - x_i \\ x_{i-1} - x_i \end{pmatrix} . \tag{3.73}$$

To approximate the first derivative $f'(x)$:

$$f'(x) \approx \frac{\frac{1}{4}x_{i-4} - \frac{4}{3}x_{i-3} + 3x_{i-2} - 4x_{i-1} + \frac{25}{12}x_i}{\Delta x} + O(\Delta x^4) . \tag{3.74}$$

To approximate the second derivative $f''(x)$:

$$f''(x) \approx \frac{\frac{11}{12}x_{i-4} - \frac{14}{3}x_{i-3} + \frac{19}{2}x_{i-2} - \frac{26}{3}x_{i-1} + \frac{35}{12}x_i}{\Delta x^2} + O(\Delta x^3) . \tag{3.75}$$

To approximate the third derivative $f'''(x)$:

$$f'''(x) \approx \frac{\frac{3}{2}x_{i-4} - 7x_{i-3} + 12x_{i-2} - 9x_{i-1} + \frac{5}{2}x_i}{\Delta x^3} + O(\Delta x^2) . \tag{3.76}$$

To approximate the fourth derivative $f^{(4)}(x)$:

$$f^{(4)}(x) \approx \frac{x_{i-4} - 4x_{i-3} + 6x_{i-2} - 4x_{i-1} + x_i}{\Delta x^4} + O(\Delta x) . \tag{3.77}$$

The coefficients of the backward difference formulas for five-point approximation are in Table 3.4.

M	**O**	**stencil x-coordinates**				
		−4	**−3**	**−2**	**−1**	**0**
1	4	$\frac{1}{4}$	$-\frac{4}{3}$	3	−4	$\frac{25}{12}$
2	3	$\frac{11}{12}$	$-\frac{14}{3}$	$\frac{19}{2}$	$-\frac{26}{3}$	$\frac{35}{12}$
3	2	$\frac{3}{2}$	−7	12	−9	$\frac{5}{2}$
4	1	1	−4	6	−4	1

Table 3.4: Table of coefficients for backward difference formulas.

The coefficients for the finite difference formulas are in Table A.2 in Appendix A and can be obtained from Table A.1 for forward finite difference formulas. The coefficients have the opposite sign for odd derivatives, whereas the signs are the same for even derivatives.

3.3.3 Parameters affecting the accuracy of finite difference formulas

There are several parameters affecting the accuracy. The first parameter is the *spatial step-size* (denoted as Δx), which depends on the number of grid points (denoted as S) and the length of the spatial domain (denoted as L). Therefore, the spatial step size is defined as $\Delta x = L/S$. If the smaller step is used, the resulting solution is more accurate. If the step is too small, the calculation time may increase considerably. On the other hand, when the selected step is too large, the solution may not be accurate.

The second parameter is the *order of the finite difference formulas* (denoted as O). The higher order is used, the more accurate solution is obtained. The selection of the difference formula depends on the order of derivatives that can be obtained. Consider a finite difference formula of O^{th}-order, then derivatives up to $O-1$ order can be obtained at each grid point. The selection of the order of the method is related to the selection of the step of the spatial variable.

There are also differences between *types of finite difference formulas*. If the forward or backward difference formulas are used, the same calculation error is obtained because both of these formulas are asymmetrical, which causes the error accumulation. On the other hand, the symmetrical difference formula uses the same number of points on both sides, and this formula has a smaller calculation error [pp8, pp9, pp18].

Let us demonstrate the behavior of finite difference on the one-dimensional wave equation. The wave equation (3.78) is an important hyperbolic partial differential equation, and it is widely used in many technical problems (vibration of the string, AC circuits, electromagnetism, etc.)

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial t^2}\,. \tag{3.78}$$

Let Ω be a domain of PDE $(x,t) = \langle 0, \pi\rangle \times \langle 0, t_{max}\rangle$, where t_{max} is the maximum simulation time, and homogeneous Dirichlet boundary conditions on $\partial\Omega$ are defined as follows:

$$u(0,t) = u(1,t) = 0\,. \tag{3.79}$$

Cauchy initial values follow:

$$u(x,0) = \sin(\pi x)\,, \quad \frac{\partial u(x,0)}{\partial t} = 0\,. \tag{3.80}$$

The wave equation describes a vibration of an ideal string of specified length. The string is modeled by a sine function. The string is fixed at both ends to the x-axis (see boundary values (3.79)). The string is released at time $t = 0$, and the initial velocity of the string is zero. The analytical solution is in the form:

$$y = \cos(\pi t)\sin(\pi x)\,. \tag{3.81}$$

First, we discuss the accuracy of the calculation in the space domain and then in the time domain. The absolute error between numerical and analytical solutions for the central difference formulas in the spatial domain is shown in Figure 3.4 (left). Note that the spatial step size is ($\Delta x = \pi/S$). The coefficients for the central difference formulas for parameters second-order derivative $M = 2$ and different orders of accuracy $O =$ are summarized in Table 3.2. The red function shows the absolute error for the finite difference formula of order $O = 4$ and $\Delta x \approx 0.1$ ($S = 32$ segments). The average absolute error is $5.7e^{-5}$. The blue function shows this error for $O = 8$ and $\Delta x \approx 0.01$ ($S = 315$ segments), where the average absolute error is $4.2e^{-17}$. Note that these functions mostly remain at the same level. The higher deviation between numerical and analytical solutions at the boundary of the spatial domain is caused by using asymmetrical difference formulas.

Figure 3.4 (right) shows the absolute error for the forward difference formulas. The settings are the same as the settings for the central difference formulas. In this case, the red function shows the absolute error of the finite difference formula for the parameters $O = 4$ and $\Delta x \approx 0.1$ ($S = 32$ segments), the average absolute error is $5.4e^{-4}$. The blue function shows the absolute error for $O = 8$ and $S = 315$ $\Delta x \approx 0.01$ ($S = 315$ segments),

the average absolute error is $3.9e^{-15}$. We can see that the absolute errors of the central difference formulas are lower than those of the forward difference formulas.

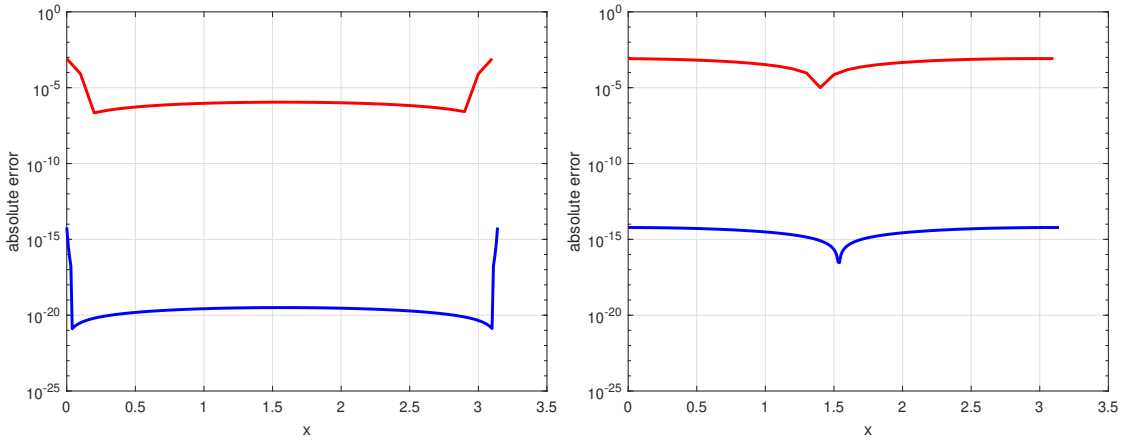


Figure 3.4: Central (left) and forward (right) difference formulas (spatial domain).

The absolute error in the time domain is shown in Figure 3.5 and is calculated as $error_{time} = |u_{middle} - \cos(\pi t)|$, where u_{middle} denotes the value of the function $u(L/2, t)$ at the middle point of the string. The upper red function shows the absolute error for $O = 2$ and $S = 10$ ($\Delta x \approx 0.3$). One option to reduce the absolute error is to use more segments S and, therefore, smaller spatial step-size Δx. The lower red function shows the absolute error for $O = 2$ and $S = 100$ ($\Delta x \approx 0.03$). The absolute error can also be reduced by using a higher-order difference formula. The upper blue function shows the error for $O = 4$ and $S = 12$ ($\Delta x \approx 0.26$), and the lower blue function shows the error for $0 = 4$ and $S = 100$.

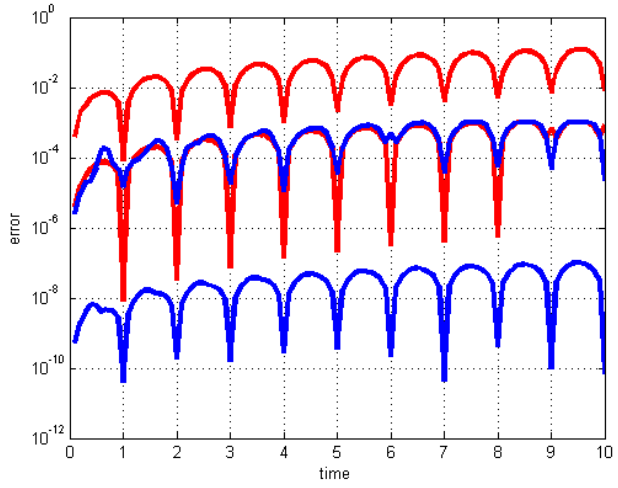


Figure 3.5: The absolute error between numerical and analytical solution for different (time domain).

Influence of arithmetic to finite difference formulas

The following experiments show the impact of arithmetic on the values of computed derivatives. The function $y = \sin(x)$ is used for these experiments [pp9, pp11]. The derivatives of the function are computed at the point $x = 0$, and the spatial step size is $\Delta x = 0.01$.

The experiments were performed using MATLAB software, which allows the use of variable precision arithmetic (function `vpa()`, for more details, see [159]). Equation (3.82) was used to convert significant digits to bits and vice versa. Let us denote *significant digits* as *SD* and *number of bits* as *BITS*

$$SD = \frac{BITS}{\log_2 10}, \qquad BITS = SD \cdot \log_2 10\,. \tag{3.82}$$

The experiments were performed using forward and central difference formulas of different orders. The arithmetic was set to 8 B, 16 B, 32 B, and 64 B. Notice that the arithmetic does not influence lower orders of finite difference formulas in contrast with higher orders. The following graphs show the absolute difference between numerical and analytical solutions (y-axis). The order of the derivative is shown on the x-axis. Only odd orders of derivatives of the function $y = \sin(x)$ are considered. Even derivatives equal zero and only reflect the variable precision arithmetic setting, so they are omitted. The analytical solution of the derivatives of the function $y = \sin(x)$, where $x = 0$, follows:

$$\begin{aligned} y' &= \cos(x) = 1 & y'' &= -\sin(x) &&= 0 \\ y''' &= -\cos(x) = -1 & y'''' &= \sin(x) &&= 0\,. \end{aligned} \tag{3.83}$$

The sequence repeats. Figures 3.6, 3.7 (left) always show the absolute error for forward difference formulas of the given order, Figures 3.6, 3.7 (right) for central difference formulas.

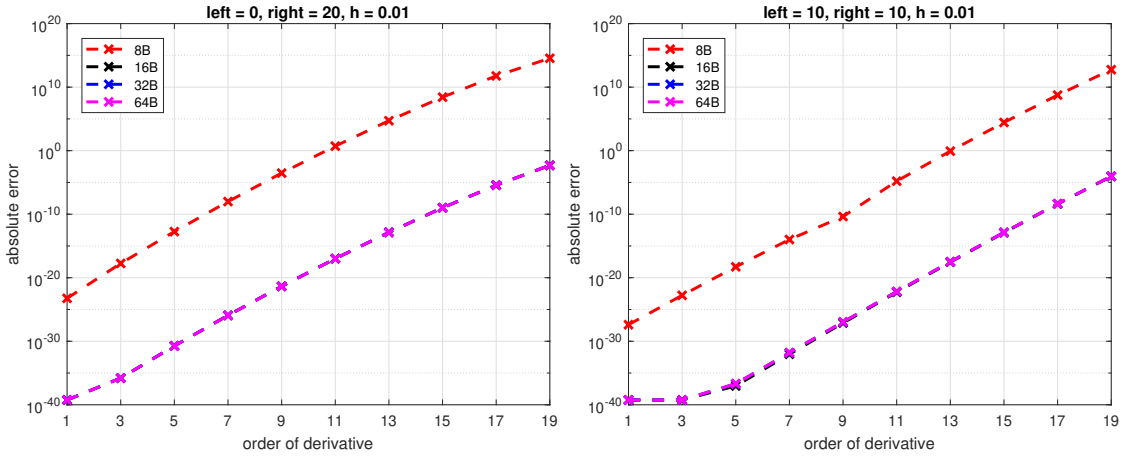


Figure 3.6: Forward (left) and central (right) difference formula, order $O = 20$.

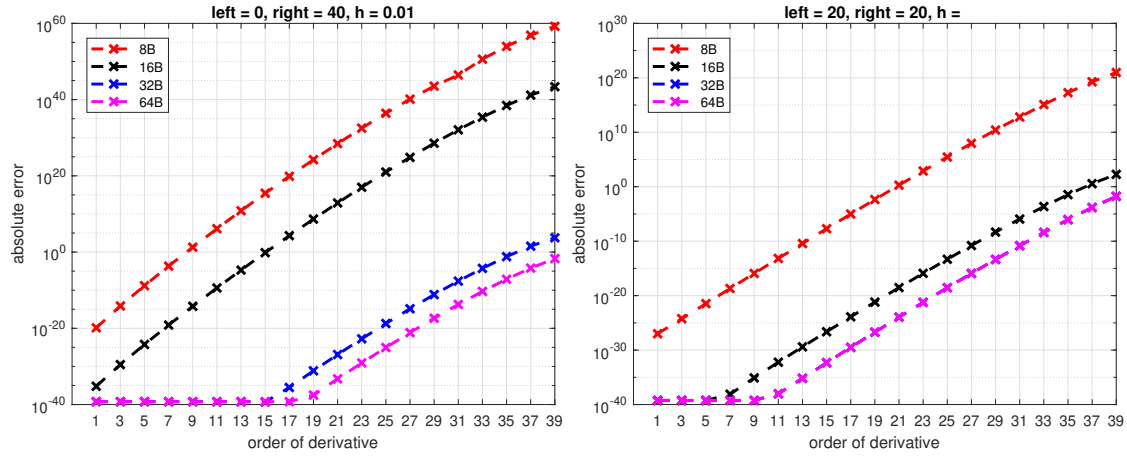


Figure 3.7: Forward (left) and central (right) difference formula, order $O = 40$.

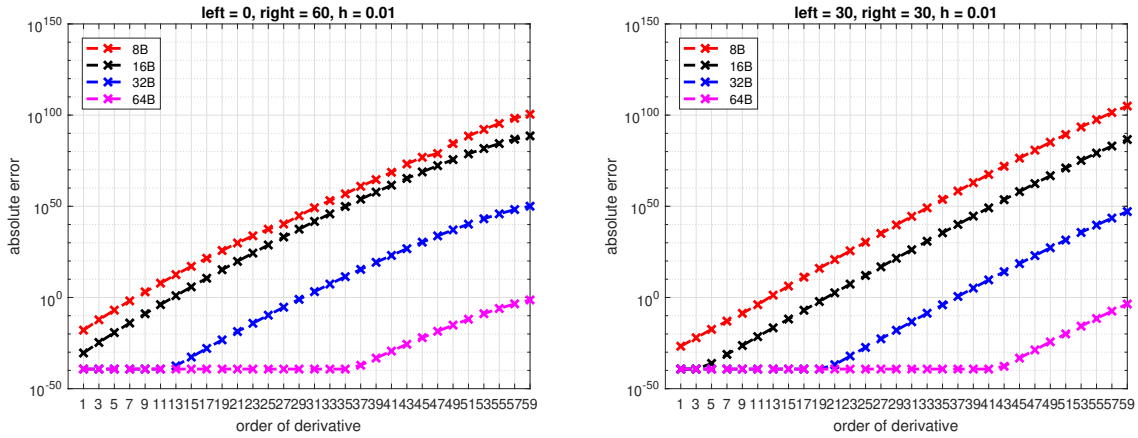


Figure 3.8: Forward (left) and central (right) difference formula, order $n = 60$.

Let l be a number of points to the left of the current point of the calculation, and r be a number of points to the right (see (3.44)). Figure 3.6 (left) shows the forward difference formula (parameters $l = 0$, $r = 20$), Figure 3.6 (right) shows the central difference formula (parameters $l = 10$, $r = 10$). The order of difference formulas is set to $O = 20$. Figure 3.7 (left) shows the forward difference formula (parameters $l = 0$, $r = 40$), Figure 3.6 (right) shows the central difference formula (parameters $l = 20$, $r = 20$). The order of the difference formulas is set to $O = 40$. Figure 3.8 (left) shows the forward difference formula (parameters $l = 0$, $r = 60$), Figure 3.8 (right) shows the central difference formula (parameters $l = 30$, $r = 30$). The order of difference formulas is set to $n = 60$.

Differences between various lengths of arithmetic occur when higher orders of differential formulas are used. The upper red function shows the absolute error for 8 B arithmetic, the black, blue, and magenta for 16 B, 32 B and 64 B arithmetic, respectively. Notice that derivatives computed using higher orders of finite difference formulas are more accurate than lower orders. There are also differences between types of finite difference formulas. The central difference formulas (right) are more accurate than the forward difference formulas (left) because the error accumulation is less significant than the forward difference formulas.

The selection of arithmetic has an essential influence on the accuracy of calculating derivatives' values. The following Tables 3.5, 3.6, 3.7, and 3.8 contain the values of deriva-

tives for function $y = \sin(x)$ at point $x = 0$, with a spatial step-size $\Delta x = 0.001$, and parameters $l = 10$, $r = 10$. Expected solutions of individual derivatives $y'(0) = 0$, $y''(0) = -1$, $y'''(0) = 0, \cdots$ are shown in the *Analytical* columns, numerical solutions in the *Numerical* columns, and the absolute error in the *Absolute error* columns.

Table 3.5 implies that an accumulated calculation error starts showing up on the 8 B arithmetic from the derivative of 1st order. On the other hand, in Tables 3.6, 3.7, and 3.8, an accumulated calculation error starts showing up on the 16 B, 32 B, 64 B arithmetic from the 3rd, 7th, and 17th-order derivative, respectively.

Derivation	**Numerical**	**Analytical**	**Absolute error**
1	**1.00e^{00}**	**1**	**4.77e^{-06}**
2	$4.68e^{-04}$	0	$4.68e^{-04}$
3	$-8.25e^{-01}$	-1	$1.75e^{-01}$
4	$-1.36e^{01}$	0	$1.36e^{01}$
5	$-8.25e^{03}$	1	$8.25e^{03}$
6	$5.55e^{05}$	0	$5.55e^{05}$
7	$3.41e^{08}$	-1	$3.41e^{08}$
8	$-2.08e^{10}$	0	$2.08e^{10}$
9	$-1.22e^{13}$	1	$1.22e^{13}$
10	$6.24e^{14}$	0	$6.24e^{14}$
11	$3.66e^{17}$	-1	$3.66e^{17}$
12	$-1.23e^{19}$	0	$1.23e^{19}$
13	$-8.97e^{21}$	1	$8.97e^{21}$
14	$2.48e^{22}$	0	$2.48e^{22}$
15	$1.67e^{26}$	-1	$1.67e^{26}$
16	$8.43e^{27}$	0	$8.43e^{27}$
17	$-2.10e^{30}$	1	$2.10e^{30}$
18	$-2.97e^{32}$	0	$2.97e^{32}$
19	$1.31e^{34}$	-1	$1.31e^{34}$
20	$3.96e^{36}$	0	$3.96e^{36}$

Table 3.5: Values of derivatives, 8 B arithmetic.

Derivation	Numerical	Analytical	Absolute error
1	$1.00e^{00}$	1	$3.64e^{-12}$
2	$0.00e^{00}$	0	$1.97e^{-10}$
3	$\mathbf{-1.00e^{00}}$	**−1**	$\mathbf{3.63e^{-07}}$
4	$-3.40e^{-05}$	0	$3.39e^{-05}$
5	$1.03e^{00}$	1	$2.63e^{-02}$
6	$3.50e^{00}$	0	$3.50e^{00}$
7	$-1.45e^{03}$	−1	$1.45e^{03}$
8	$-2.51e^{05}$	0	$2.51e^{05}$
9	$6.70e^{07}$	1	$6.70e^{07}$
10	$1.44e^{10}$	0	$1.44e^{10}$
11	$-2.62e^{12}$	−1	$2.62e^{12}$
12	$-6.72e^{14}$	0	$6.72e^{14}$
13	$8.48e^{16}$	1	$8.48e^{16}$
14	$2.54e^{19}$	0	$2.54e^{19}$
15	$-2.13e^{21}$	−1	$2.13e^{21}$
16	$-7.31e^{23}$	0	$7.31e^{23}$
17	$3.71e^{25}$	1	$3.71e^{25}$
18	$1.43e^{28}$	0	$1.43e^{28}$
19	$-3.34e^{29}$	−1	$3.34e^{29}$
20	$-1.44e^{32}$	0	$1.44e^{32}$

Table 3.6: Values of derivatives, 16 B arithmetic.

Derivation	Numerical	Analytical	Absolute error
1	$1.00e^{00}$	1	$0.00e^{00}$
2	$0.00e^{00}$	0	$2.73e^{-28}$
3	$-1.00e^{00}$	−1	$4.55e^{-19}$
4	$0.00e^{00}$	0	$1.70e^{-23}$
5	$1.00e^{00}$	1	$2.75e^{-14}$
6	$0.00e^{00}$	0	$8.01e^{-19}$
7	$\mathbf{-1.00e^{00}}$	**−1**	$\mathbf{1.39e^{-09}}$
8	$0.00e^{00}$	0	$3.22e^{-14}$
9	$1.00e^{00}$	1	$6.10e^{-05}$
10	$0.00e^{00}$	0	$1.10e^{-09}$
11	$-3.32e^{00}$	−1	$2.32e^{00}$
12	$-3.00e^{-05}$	0	$3.04e^{-05}$
13	$7.35e^{04}$	1	$7.35e^{04}$
14	$6.23e^{-01}$	0	$6.23e^{-01}$
15	$-1.82e^{09}$	−1	$1.82e^{09}$
16	$-7.40e^{03}$	0	$7.40e^{03}$
17	$3.12e^{13}$	1	$3.12e^{13}$
18	$-3.77e^{06}$	0	$3.77e^{06}$
19	$-2.77e^{17}$	−1	$2.77e^{17}$
. 20	$1.14e^{12}$	0	$1.14e^{12}$

Table 3.7: Values of derivatives, 32 B arithmetic.

Derivation	Numerical	Analytical	Absolute error
1	$1.00e^{00}$	1	$4.38e^{-47}$
2	$0.00e^{00}$	0	$1.85e^{-52}$
3	$-1.00e^{00}$	-1	$2.86e^{-42}$
4	$0.00e^{00}$	0	$2.06e^{-47}$
5	$1.00e^{00}$	1	$2.37e^{-37}$
6	$0.00e^{00}$	0	$2.28e^{-42}$
7	$-1.00e^{00}$	-1	$1.80e^{-32}$
8	$0.00e^{00}$	0	$1.60e^{-37}$
9	$1.00e^{00}$	1	$1.21e^{-27}$
10	$0.00e^{00}$	0	$8.92e^{-33}$
11	$-1.00e^{00}$	-1	$7.07e^{-23}$
12	$0.00e^{00}$	0	$4.09e^{-28}$
13	$1.00e^{00}$	1	$3.50e^{-18}$
14	$0.00e^{00}$	0	$1.52e^{-23}$
15	$-1.00e^{00}$	-1	$1.42e^{-13}$
16	$0.00e^{00}$	0	$4.32e^{-19}$
17	$\mathbf{1.00e^{00}}$	**1**	$\mathbf{4.39e^{-09}}$
18	$0.00e^{00}$	0	$8.38e^{-15}$
19	$-1.00e^{00}$	-1	$9.28e^{-05}$
20	$0.00e^{00}$	0	$8.32e^{-11}$

Table 3.8: Values of derivatives, 64 B arithmetic.

3.4 Method of lines

The problems described by PDEs differ in various aspects, for example, geometric classification (parabolic, hyperbolic, elliptic), linearity, types of coefficients, number of dependent variables (number of simultaneous PDEs), number of independent variables (number of dimensions), type of boundary conditions, and so on [78].

The method of lines is a general technique for solving PDEs, which discretizes one dimension and then integrates the semi-discrete problem as a system of ODEs. Spatial derivatives are often discretized, and the time variable remains continuous. The resulting system of ODEs can be solved using standard numerical methods for initial value ODEs to calculate the approximate numerical solution of the original PDE problem [70, 77, 113, 146, 145, 99]. One of the main advantages of MOL is that it can use existing and well-established numerical methods for a numerical solution to the PDE. In addition, the important advantages of MOL are the simplicity of the explicit methods. Higher-order approximations can be achieved in the discretization of spatial derivatives without significant increases in computational complexity [80]. This technique has broad applicability to physical and chemical systems modeled by PDEs [78, 146, 150].

Let us consider the linear advection equation

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = 0\,. \tag{3.84}$$

In physical applications, v denotes a linear or flow velocity. First, the spatial derivative $\frac{\partial u}{\partial x}$ is replaced by a *finite difference (FD)*

$$\frac{\partial u}{\partial x} \approx \frac{u_j - u_{j-1}}{\Delta x}\,. \tag{3.85}$$

where j is an index that indicates a position along a grid and Δx is the spatial step size along the grid. Assume that the grid has N points. The left end value of x has an index $x = 1$, and the right end value of x has an index $x = N$. The spatial domain of (3.84) is discretized using MOL, resulting in a system of N ODEs:

$$\frac{\partial u_i}{\partial t} = -v\frac{u_j - u_{j-1}}{\Delta x}, 1 \leq j \leq N. \tag{3.86}$$

The (3.84) is of first order in t and x. To complete the specification of the PDE problem (3.84), one initial condition (IC) and one boundary condition (BC) are required. The initial condition is

$$u(x, t = 0) = f(x). \tag{3.87}$$

The boundary condition is

$$u(x = 0, t) = g(x). \tag{3.88}$$

Since (3.86) consists of N ODEs, N initial conditions must be specified:

$$u(x_j, t = 0) = f(x_i), 1 \leq j \leq N. \tag{3.89}$$

The resulting MOL approximation of (3.84) consists of (3.86), (3.89), and (3.88). The solution of the system of ODEs is N functions $u_1(t), u_2(t), \ldots, u_{N-1}(t), u_N(t)$, an approximation to $u(x, t)$ at grid points $j = 1, 2, \ldots, N$. The principle of MOL is shown in Figure 3.9.

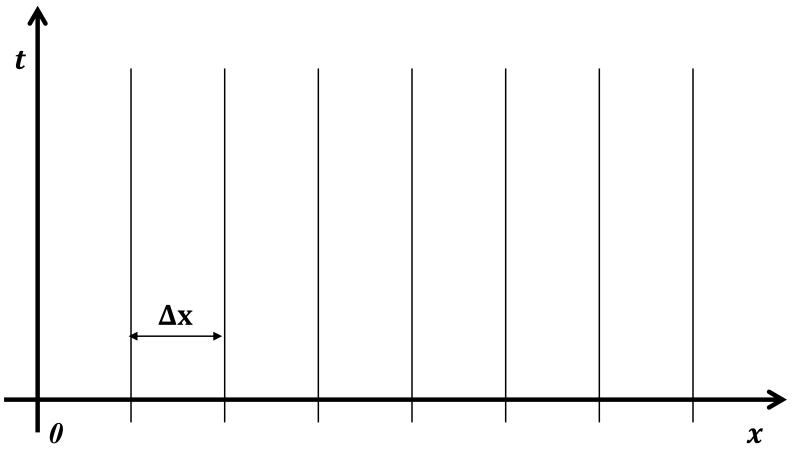


Figure 3.9: The principle of MOL.

The FD approximation can be written in the form

$$\frac{\partial u}{\partial x} \approx \frac{u_j - u_{j-1}}{\Delta x} + O(\Delta x), \tag{3.90}$$

where $O(\Delta x)$ is the *truncation error* of the approximation of (3.86), and is derived from a truncated Taylor series). Since Δx appears in the truncation error term to the first power, (3.90) is called *first order* FD. The name *finite difference* is derived as follows. Note that the numerator of (3.90) $u_j - u_{j-1}$ is a difference of two values of u. The denominator Δx is nonzero – finite. There are two main aspects to consider when using MOL:

- choice of the approximation of the spatial derivatives,

- choice of the numerical integration method for solving systems of ODEs.

The advantages of MOL are as follows:

- separating the problem of space and time allows establishing stability and convergence for a wide range of problems,
- numerical VSVO techniques can be directly applied to maintain the accuracy and stability of the solution.

A possible disadvantage is the more difficult optimization of MOL by decoupling the analysis of space and time discretizations [84].

3.5 von Neumann stability analysis

The prediction of numerical stability has great importance in practice. The *von Neumann stability analysis* [35] is a tool for validating a given numerical scheme based on Fourier analysis. There are several assumptions for the von Neumann stability analysis. First, it does not consider boundary effects, and second, it assumes that the finite difference coefficients vary slowly and can be considered constant in time and space. Therefore, the equations are linear. The von Neumann analysis defines an error in the solution and studies its amplification over time. If a solution is stable, any perturbation in the input remains bounded, whereas, for an unstable solution, the perturbations grow over time [150].

The solution can be written as the sum of eigenvalues in the following form:

$$u_j^n = \xi^n e^{ikxj}\,, \tag{3.91}$$

where k is the spatial wave number and $\xi(k)$ is a complex number. We can express the finite difference equation as:

$$u_j^{n+1} = T(\Delta t^p, \Delta x^q) u_j^n\,, \tag{3.92}$$

where $T(\Delta t^p, \Delta x^q)$ is the evolution operator of order p in time and q in space. From equations (3.91) and (3.92), we can see that the time evolution of a single eigenmode is a succession integer powers of the complex number ξ, which is called *amplification factor*. For the stability criterion, the modulus of the amplification factor is less than or equal to 1, therefore:

$$|\xi|^2 = \xi\xi^* \leq 1\,. \tag{3.93}$$

Hence, if $|\xi| > 1$ for any value of wavenumber k, then the scheme is unstable. Note that von Neumann stability is a necessary but not sufficient condition for stability. That means a numerical scheme that appears stable with respect to von Neumann stability analysis might still be unstable [140].

Let us consider the one-dimensional linear convection equation:

$$\frac{\partial u}{\partial t} = c\frac{\partial u}{\partial x}\,. \tag{3.94}$$

Consider the finite difference approximation of equation (3.94):

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = -v\frac{u_j^n - u_{j-1}^n}{\Delta x}\,, \tag{3.95}$$

for u_i^{n+1} we have:

$$u_j^{n+1} = u_j^n - \frac{v\Delta t}{\Delta x}(u_j^n - u_{j-1}^n)\,, j = 1, 2, \ldots, N\,, \tag{3.96}$$

where N is the number of ODEs, n is the index of the time variable t, and j is the index of the spatial variable x. Equation (3.96) is the basic ODE integration method called *forward Euler method*. This method is *explicit* because we can solve the solution at point t_{n+1} from the solution at point t_n. The numerical scheme (3.96) has a limitation. If the time step Δt is above a critical value, the calculation becomes unstable, which means that $\Delta u = u_j^{n+1} - u_j^n$ becomes larger and potentially unbounded as time moves forward. The numerical solution of (3.94) using (3.96) is stable if the expression $\frac{v\Delta t}{\Delta x}$ called *Courant-Friedrichs-Lewy (CFL) number* (also called *Courant number*) remains below the critical value [78].

Richard Courant, Kurt Friedrichs, and Hans Lewy described in their paper in 1928 [34] that a numerical scheme is stable only when its numerical domain of dependence at any point in space and time includes the mathematical/analytical domain of dependence. If a numerical scheme violates this condition, the scheme does not consider all the data necessary to advance the solution in time. On the other hand, if the numerical domain is much larger than the analytical domain, this causes extraneous data to be included in the solution [150]. The von Neumann analysis constrains the CFL number, which limits the mesh size and the timestep size, which are required for the stability of a numerical solution. The Courant number is dimensionless.

The CFL number in one-dimensional case for (3.94) is:

$$C = \frac{c\Delta t}{\Delta x}\,, \tag{3.97}$$

where c is the characteristic wave speed of the system, Δt is the time step of the numerical method, and Δx is the spatial step of the system.

The CFL condition defines a relation between two velocities: the *physical velocity* c in the medium where the waves are propagated, and the *grid velocity* $\frac{\Delta x}{\Delta t}$ relating the grid increment Δx divided to the time increment Δt. The CFL condition for explicit problems has the following form:

$$C = \frac{c\Delta t}{\Delta x} \leq C_{max}\,, \tag{3.98}$$

where the C_{max} value depends on the dimensionality of the problem and on the specific numerical method. The CFL condition is illustrated in Figure 3.10. The schematic diagram shows stable and unstable choices of time step Δt. The dashed lines limit the numerical domain of dependence of the solution at x_j^{n+1}, and the shaded area represents the physical domain of dependence. To achieve numerical stability, the numerical domain has to be larger than the physical domain.

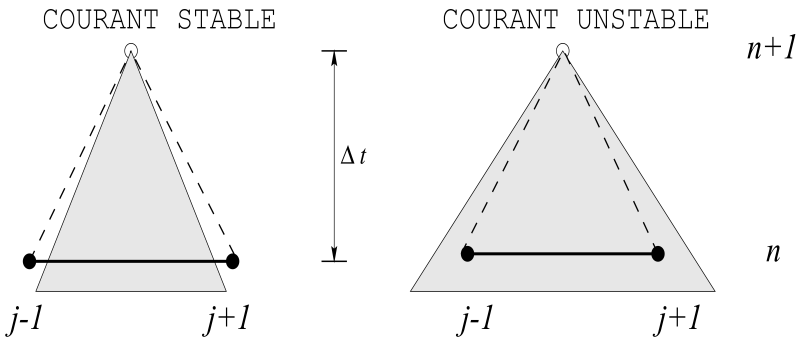


Figure 3.10: The illustration of the CFL condition [140].

To increase the accuracy of (3.96), a smaller Δx has to be chosen, which results in a larger number of grid points in the spatial domain x and, therefore, a larger number of ODEs. At the same time, a smaller value of Δt is required to keep the CFL number below the critical value. This results in a conflicting requirement: improving accuracy while maintaining numerical stability.

Although the Method of Lines does not need the explicit CFL condition, the CFL condition still applies to the time-stepping techniques. The Method of Lines transforms the given PDE into a semi-discrete system of ODEs without any time discretization. The CFL condition is important only if the system is discretized in time and space. But we can see that if we use, for example, the forward Euler method (3.96) to solve the ODE system, the standard CFL condition is required to guarantee uniform stability [80].

3.6 Numerical stability of method of lines

Let us consider the partial differential equation of the initial or boundary value type:

$$\frac{\partial u}{\partial t} = X(u)\,, \tag{3.99}$$

where $u = u(x,t)$ is a function of one or several spatial variables x and of time t. The $X(\cdot)$ is a linear partial differential operator with derivatives to one or more variables x. The number of initial and boundary conditions depends on the form of equation (3.99). Let $x_j = \{j \cdot \Delta x, j = 1, 2, \cdots, N\}$ be a grid of equidistant points along the x axis and let $u_j(t)$ denote the approximation of $u(x_j, t)$. The operator $X(\cdot)$ can be replaced by finite difference approximations in the following form:

$$[X(u)]_j = \sum_\beta A_\beta u_{j+\beta}\,, \tag{3.100}$$

where A_β are constant coefficients. Equation (3.99) is approximated along the lines $x = x_j$, which are parallel to the time axis using the system of ODEs:

$$\frac{\partial u_j}{\partial t} = \sum_\beta = A_\beta u_{j+\beta}\,, \tag{3.101}$$

The initial value problem (3.101) can then be solved by standard algorithms for ODEs. Similarly, discretization of time t in equal intervals yields in $t^n = n \cdot \Delta t, n = 0, 1, 2, \cdots, T$,

and u_j^n is the approximation of $u_j(t^n)$. Analysis of numerical stability in integration of PDEs (3.99) results in analysis of numerical stability in integration of ODEs (3.101), consisting of these aspects [169]:

- specific classes of PDEs – define specific properties of the system of ODEs (including the discretization in the spatial domain).
- specific algorithms for the integration in time – define conditions of numerical stability of these algorithms.

3.6.1 Notation

Let us define the space-shift operator E:

$$u_{j+1}^n = E \cdot u_n^j \tag{3.102}$$

and time-shift operator z:

$$u_j^{n+1} = z \cdot u_n^j \,. \tag{3.103}$$

The finite difference approximation (3.100) can be rewritten using the operator E as:

$$[X(u)]_j = A \cdot u_j, \quad A = \sum_\beta A_\beta E^\beta \,, \tag{3.104}$$

and (3.101) becomes:

$$\frac{\partial u_j}{\partial t} = A \cdot u_j \,. \tag{3.105}$$

The general form of common numerical integration algorithms for the system of ODEs can be expressed using the time-shift operator z:

$$u_j^{n+1} = M(\Delta t \cdot A, z) \cdot u_j^n \,, \tag{3.106}$$

where the operator $M(\Delta t \cdot A, z)$ contains integer powers of $\Delta t \cdot A$. For the numerical stability of (3.106), the solution must be of the form:

$$u_j^n = a^n e^{i\omega x_j} \,, \tag{3.107}$$

where $e^{i\omega x_j}$ is the eigenfunction of the operator A

$$\hat{A} \cdot e^{i\omega x_j} = \hat{A}(\omega) e^{i\omega x_j} \,, \tag{3.108}$$

and $\hat{A}(\omega)$ is a (complex) scalar function defined as:

$$\hat{A}(\omega) = \sum_\beta A_\beta e^{i\omega\beta\Delta x} \,, \tag{3.109}$$

the scalar function $\hat{A}(\omega)$ is a spectral representation of the operator A, therefore, it is called the *spectral function* of A.

Since the operator $M(\Delta t \cdot A, z)$ contains integer powers of $\Delta t \cdot A$, and $e^{i\omega x_j}$ is the eigenfunction of the operator A, then $e^{i\omega x_j}$ are also eigenfunctions of the operator $M(\Delta t \cdot A, z)$, hence:

$$M(\Delta t \cdot A, z) \cdot e^{i\omega x_j} = M(\Delta t \cdot \hat{A}(\omega), z) \cdot e^{i\omega x_j} \,, \tag{3.110}$$

where the space-shift operator E is eliminated in the operator $M(\Delta t \cdot \hat{A}(\omega), z)$. By substitution (3.107) into (3.103), and elimination of $e^{i\omega x_j}$, the relation between a^{n+1} and a^n is obtained:

$$\frac{a^{n+1}}{a^n} = z\,, \tag{3.111}$$

and z is a constant scalar number. By substituting (3.111) into (3.110), we can see that z is the solution of the characteristic equation:

$$z - M(\Delta t \cdot \hat{A}(\omega), z) = 0\,. \tag{3.112}$$

If there exist solutions z of (3.112) that $|z| > 1$, then there exists a corresponding sequence a^n that is unbounded for $n \longrightarrow \infty$, because $a^n = a^0 \cdot z^n$.

Therefore, the stability theorem is as follows: A necessary condition for the numerical stability of the method of lines (3.101) is that all z solutions of the characteristic equation (3.112) satisfy the condition:

$$\forall \omega : \; |z| \leq 1\,. \tag{3.113}$$

Note that the condition (3.113) is equivalent to the von Neumann condition for the finite difference method (see (3.93)). The eigenvalues z of (3.112) are equivalent to the amplification factors (see (3.92)).

3.6.2 Regions of stability

The characteristic equation (3.112) can be transformed from a region S in the $\Delta t \cdot \hat{A}(\omega)$ complex plane to the $|z| \leq 1$ in the complex plane. The region S is then called the *stability region* of the integration method, and S_B denotes the *stability boundary*. Let us denote S_R, the intersection of the stability boundary S_B with the real negative axis, and S_I, the intersection of the stability boundary S_B with the imaginary axis.

The function $\hat{A}(\omega)$ depends only on the operator A, which means that it depends only on the operator $X(\cdot)$ (3.99). In other words, it depend on the finite differences, which include the order of the finite difference formula (corresponds to the A_β coefficients), and on the spatial step Δx.

In summary, the numerical stability analysis depends on the spectral function $\hat{A}(\omega)$ of the analyzed equation and the stability region S specific to the integration-in-time method.

3.6.3 Stability regions of the selected methods

Figures 3.11 show the stability regions for the higher-order Taylor series method for orders 1–25. The higher-order Taylor series method is referred to as Modern Taylor Series Method (MTSM) and is described in detail in Chapter 4.

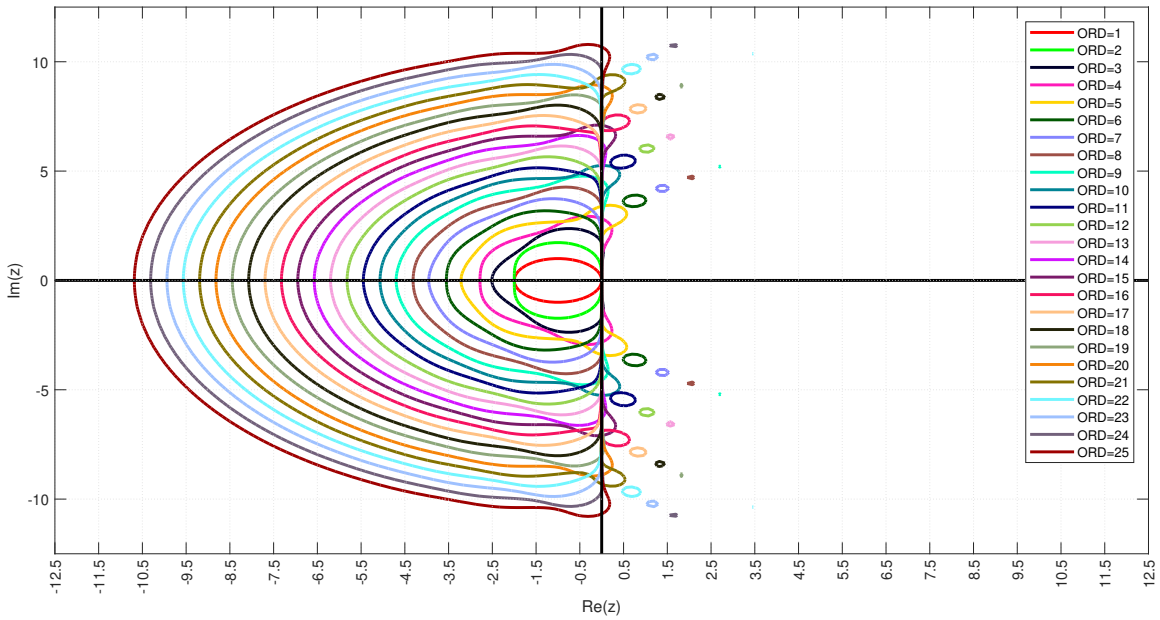


Figure 3.11: Stability regions of MTSM – orders 1–25.

The stability regions for the Dormand-Prince 5(4) method (denoted RK45) [44, 151] and the Verner Runge-Kutta method of order 8(7) (denoted RK8VR) [166] are shown in Figures 3.12 and 3.13, respectively.

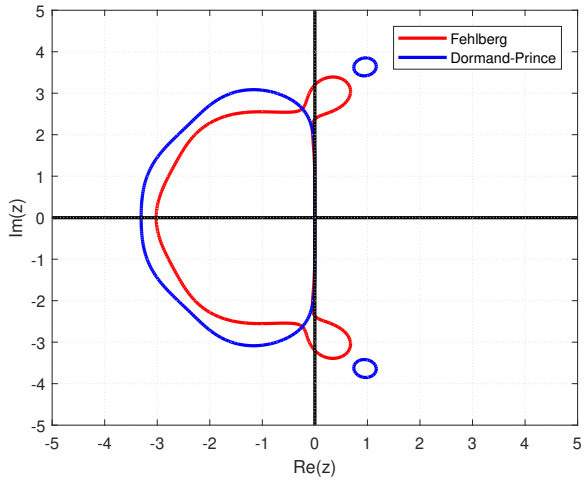


Figure 3.12: Stability regions of Dormand-Prince 5(4) and Fehlberg methods.

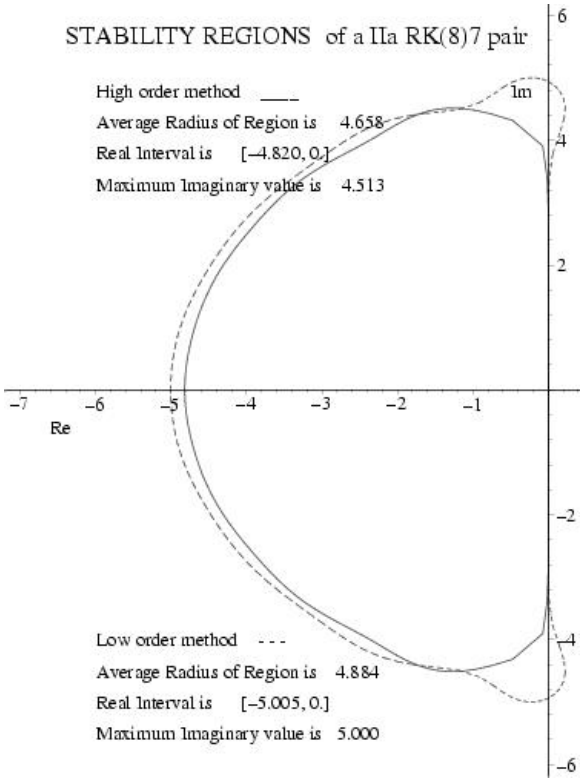


Figure 3.13: Stability regions of Verner Runge Kutta 8(7) method [167], [168].

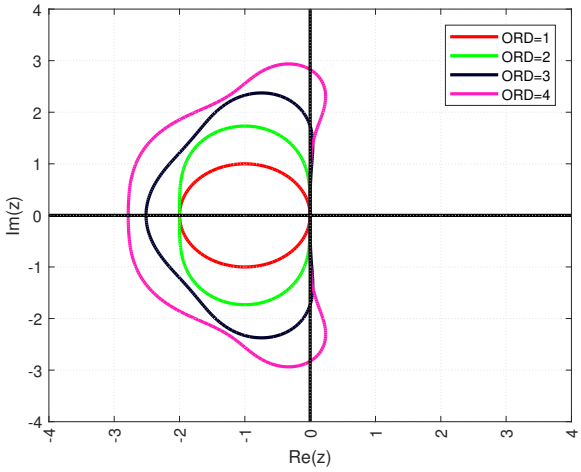


Figure 3.14: Stability regions for orders 1–4. The $ORD = 1$ corresponds to the explicit Euler method, $ORD = 4$ to the classical fourth-order Runge-Kutta method.

The S_R and S_I intersections of MTSM for orders 1–25 are summarized in Table 3.9. The intersections of MTSM for orders 26–64 are in Tables B.1 and B.2 in Appendix B.

Order	$\lvert S_R \rvert$	$\lvert S_I \rvert$
1	−2.00	±0.00
2	−2.00	±0.00
3	−2.51	±1.73
4	−2.79	±2.83
5	−3.22	±3.40
6	−3.55	±0.00
7	−3.95	±1.76
8	−4.31	±3.40
9	−4.70	±4.57
10	−5.07	±5.26
11	−5.45	±1.70
12	−5.82	±3.38
13	−6.20	±5.00
14	−6.57	±6.30
15	−6.95	±7.10
16	−7.32	±7.24
17	−7.70	±4.97
18	−8.07	±6.59
19	−8.45	±8.00
20	−8.82	±8.90
21	−9.20	±9.32
22	−9.57	±6.55
23	−9.94	±8.17
24	−10.32	±9.66
25	−10.69	±10.68

Table 3.9: S_R and S_I intersections of MTSM for orders 1–25.

The S_R and S_I intersections of the selected Runge-Kutta methods are shown in Table 3.10.

Method	$\lvert S_R \rvert$	$\lvert S_I \rvert$
RK45	−3.27	0
RK8VR	−4.82	±4.51

Table 3.10: S_R and S_I intersections of selected Runge-Kutta methods.

3.6.4 Stability analysis of the parabolic equation

First, let us examine the numerical stability of the one-dimensional heat equation:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \tag{3.114}$$

The (3.114) is discretized in space using second-order central differences:

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{u_{j-1} - 2u_j + u_{j+1}}{\Delta x^2} \right), \tag{3.115}$$

where the coefficients A_β from (3.101) are:

$$A_{-1} = A_1 = \frac{\alpha}{\Delta x^2}\,, \quad A_0 = \frac{-2\alpha}{\Delta x^2}\,. \tag{3.116}$$

The spectral function $\hat{A}(\omega)$ is:

$$\hat{A}(\omega) = \frac{\alpha}{\Delta x^2}\left(-2 + 1e^{i\omega\Delta x} + 1e^{i\omega\Delta x}\right)\,. \tag{3.117}$$

Because $2\cos(\omega) = e^{i\omega} + e^{-i\omega}$, we have:

$$\hat{A}(\omega) = \frac{\alpha}{\Delta x^2}\left(-2 + 2\cos(\omega\Delta x)\right) \tag{3.118}$$

and

$$\hat{A}(\omega) = \frac{-2\alpha}{\Delta x^2}\left(1 - \cos(\omega\Delta x)\right)\,. \tag{3.119}$$

The spectral function $\hat{A}(\omega)$ is real-negative for all values of ω. The stability condition is:

$$|\Delta t \cdot \hat{A}(\omega)|_{\max} = \frac{4\alpha\Delta t}{\Delta x^2} \leq |S_R|\,. \tag{3.120}$$

Note that most common integration methods M enclose at least a portion of the real-negative axis. The stability condition for these algorithms is that the portion of the real-negative axis contains (3.119) for all values of ω. From (3.120) we can express Δt to obtain the maximum integration step size for the given integration method:

$$\Delta t \leq \frac{|S_R|\Delta x^2}{4\alpha}\,. \tag{3.121}$$

Table 3.11 summarizes the stability conditions for different numerical methods.

Method	$\mathbf{\lvert S_R\rvert}$	$\mathbf{\lvert S_I\rvert}$	**Stability condition** $\frac{4\alpha\Delta t}{\Delta x^2} \leq \lvert S_R\rvert$
MTSM $(ORD = 25)$	-10.7	± 10.7	$\frac{\alpha\Delta t}{\Delta x^2} \leq 2.68$
RK45	-3.27	0	$\frac{\alpha\Delta t}{\Delta x^2} \leq 0.81$
RK8VR	-4.82	± 4.51	$\frac{\alpha\Delta t}{\Delta x^2} \leq 1.21$

Table 3.11: Conditions of numerical stability for the one-dimensional heat equation, discretized in space using second-order central differences and integrated in time by different methods.

3.6.5 Stability analysis of the hyperbolic equation

The one-dimensional hyperbolic equation can be written as follows:

$$\frac{\partial^2 u}{\partial t^2} = \alpha\frac{\partial^2 u}{\partial x^2}\,. \tag{3.122}$$

The spatial domain of (3.122) is discretized in space using second-order central differences:

$$\frac{\partial^2 u}{\partial t^2} = \alpha\left(\frac{u_{j-1} - 2u_j + u_{j+1}}{\Delta x^2}\right)\,, \tag{3.123}$$

the coefficients A_β from (3.101) are:

$$A_{-1} = A_1 = \frac{\alpha}{\Delta x^2}\,, \quad A_0 = \frac{-2\alpha}{\Delta x^2}\,. \tag{3.124}$$

The spectral function $\hat{A}(\omega)$ is:

$$\hat{A}(\omega) = \frac{\alpha}{\Delta x^2}\left(-2 + 1e^{i\omega\Delta x} + 1e^{i\omega\Delta x}\right)\,. \tag{3.125}$$

Because $2\cos(\omega) = e^{i\omega} + e^{-i\omega}$, we have:

$$\hat{A}(\omega) = \frac{\alpha}{\Delta x^2}\left(-2 + 2\cos(\omega\Delta x)\right) \tag{3.126}$$

and

$$\hat{A}(\omega) = \frac{-2\alpha}{\Delta x^2}\left(1 - \cos(\omega\Delta x)\right)\,. \tag{3.127}$$

Also, $1 - \cos(\omega) = 2\sin^2(\omega\Delta x/2)$, therefore:

$$\hat{A}(\omega) = \frac{-2\alpha}{\Delta x^2}\left(2\sin^2(\omega\Delta x/2)\right) \tag{3.128}$$

$$\hat{A}(\omega) = \frac{-4\alpha}{\Delta x^2}\left(\sin^2(\omega\Delta x/2)\right)\,. \tag{3.129}$$

Due to the second derivative in t, it is necessary to construct a first-order system:

$$\begin{pmatrix}\hat{u}(\omega,t)\\ \hat{v}(\omega,t)\end{pmatrix} = \frac{2\alpha\sin(\omega\Delta x/2)\Delta t}{\Delta x}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}u(\omega,t)\\ v(\omega,t)\end{pmatrix}\,, \tag{3.130}$$

where $\hat{u}(\omega,t)$ is the Fourier series of spatial discretization. The spectral function $\hat{A}(\omega)$ is pure-imaginary for all values of ω. Some numerical integration methods do not enclose any portion of the imaginary axis, which means that they are unstable, for example, the first-order Euler method or the second-order Runge-Kutta method; see Figure 3.14. The stability condition is:

$$|\Delta t \cdot \hat{A}(\omega)|_{\max} = \frac{2\alpha\Delta t}{\Delta x} \leq |S_I|\,. \tag{3.131}$$

From (3.131) we can express Δt to obtain the maximum integration step size for a given integration method:

$$\Delta t \leq \frac{|S_I|\Delta x}{2\alpha}\,. \tag{3.132}$$

Table 3.12 summarizes the stability conditions for different numerical methods.

Method	$\lvert\mathbf{S_R}\rvert$	$\lvert\mathbf{S_I}\rvert$	**Stability condition** $\frac{2\alpha\Delta t}{\Delta x} \leq \lvert S_I\rvert$
MTSM ($ORD = 25$)	-10.7	± 10.7	$\frac{\alpha\Delta t}{\Delta x^2} \leq 5.35$
RK45	-3.27	0	$\frac{\alpha\Delta t}{\Delta x^2} \leq 0$
RK8VR	-4.82	± 4.51	$\frac{\alpha\Delta t}{\Delta x^2} \leq 2.26$

Table 3.12: Conditions of numerical stability for the one-dimensional wave equation, discretized in space using second-order central differences and integrated in time by different methods.

3.7 Higher-order differential equations

The MTSM can only solve the first-order ODEs and systems of first-order ODEs. Any higher-order ODE can be transformed into the corresponding system of the first-order ODEs using the methods discussed in the following subsections. Note that the results of the methods are equivalent [pp4].

3.7.1 Method of derivation order reduction

On the right side, the coercive function z cannot have a derivative. Consider the following equation (3.133)

$$\begin{aligned} &y'''' + a_3 y''' + a_2 y'' + a_1 y' + a_0 y = b_0 z \\ &y(0) = y'(0) = y''(0) = y'''(0) = 0\,. \end{aligned} \tag{3.133}$$

We rewrite (3.133) using the Laplace transform to obtain (3.134)

$$s^4 y + a_3 s^3 y + a_2 s^2 y + a_1 s y + a_0 y = b_0 z\,. \tag{3.134}$$

It is possible to rewrite (3.134) to the system of ODEs (3.135). Elements $\frac{1}{s}$ denote numerical integrators

$$\begin{aligned} s^4 y &= b_0 z - a_3 s^3 y - a_2 s^2 y - a_1 s y - a_0 y \\ s^3 y &= \frac{1}{s} s^4 y & s^3 y(0) &= 0 \\ s^2 y &= \frac{1}{s} s^3 y & s^2 y(0) &= 0 \\ s y &= \frac{1}{s} s^2 y & s y(0) &= 0 \\ y &= \frac{1}{s} s y & y(0) &= 0\,. \end{aligned} \tag{3.135}$$

3.7.2 Method of derivation order reduction with an additional variable

The standard order reduction method cannot be used if the differential equation contains the derivative of the coercive function z. Let us consider the higher-order ordinary differential equation (3.136)

$$\begin{aligned} &y'''' + a_3 y''' + a_2 y'' + a_1 y' + a_0 y = b4 z'''' + b_3 z''' + b_2 z'' + b_1 z' + b_0 z \\ &y(0) = y'(0) = y''(0) = y'''(0) = 0 \\ &z(0) = 1, z'(0) = z''(0) = z'''(0) = z''''(0) = 0\,. \end{aligned} \tag{3.136}$$

Equation (3.136) can be simplified using the Laplace operator

$$\begin{aligned} s^4 y + a_3 s^3 y + a_2 s^2 y + a_1 s y + a_0 y &= b_4 s^4 z + b_3 s^3 z + b_2 s^2 z + b_1 s z + b_0 z \\ y(s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0) &= z(b_4 s^4 + b_3 s^3 + b_2 s^2 + b_1 s + b_0)\,. \end{aligned} \tag{3.137}$$

The equation for output y follows

$$y = \frac{b_4 s^4 + b_3 s^3 + b_2 s^2 + b_1 s + b_0}{s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0} z\,. \tag{3.138}$$

The additional variable v can be defined as

$$v = \frac{z}{s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0}\,. \tag{3.139}$$

Using the substitution of additional variable (3.139) to (3.138), we obtain

$$y = (b_4 s^4 + b_3 s^3 + b_2 s^2 + b_1 s + b_0)v\,. \tag{3.140}$$

The order of the additional variable can be reduced using the standard order reduction method (see Section 3.7.1)

$$\begin{aligned}
&s^4 v + a_3 s^3 v + a_2 s^2 v + a_1 s v + a_0 v = z \\
&s^4 v = z - a_3 s^3 v - a_2 s^2 v - a_1 s v - a_0 v \\
&s^3 v = \frac{1}{s} s^4 v && s^3 v(0) = 0 \\
&s^2 v = \frac{1}{s} s^3 v && s^2 v(0) = 0 \\
&s v = \frac{1}{s} s^2 v && s v(0) = 0 \\
&v = \frac{1}{s} s v && v(0) = 0\,.
\end{aligned} \tag{3.141}$$

3.7.3 Method of continuous integration

Again, the coercive function z has a derivative. Consider the following equation (3.142), initial conditions are $y(0) = 0, z(0) = 1$, all others are zero

$$y'''' + a_3 y''' + a_2 y'' + a_1 y' + a_0 y = b_4 z'''' + b_3 z''' + b_2 z'' + b_1 z' + b_0 z\,. \tag{3.142}$$

Equation (3.142) is rewritten using Laplace transform (3.143)

$$s^4 y + a_3 s^3 y + a_2 s^2 y + a_1 s y + a_0 y = b_4 s^4 z + b_3 s^3 z + b_2 s^2 z + b_1 s z + b_0 z\,. \tag{3.143}$$

It is possible to rewrite (3.143) in the following form. The derivatives of the same order are grouped in parentheses

$$\begin{aligned}
s^4 y &= b_4 s^4 z + s^3(b_3 z - a_3 y) + s^2(b_2 z - a_2 y) + s(b_1 z - a_1 y) + (b_0 z - a_0 y) \\
s^3 y &= b_4 s^3 z + s^2(b_3 z - a_3 y) + s(b_2 z - a_2 y) + (b_1 z - a_1 y) + v_1 \\
s^2 y &= b_4 s^2 z + s(b_3 z - a_3 y) + (b_2 z - a_2 y) + v_2 \\
s y &= b_4 s z + (b_3 z - a_3 y) + v_3 \\
y &= b_4 z + v_4\,.
\end{aligned} \tag{3.144}$$

The variables v_1, v_2, v_3 and v_4 can be calculated using the system (3.145)

$$\begin{aligned}
v_1 &= \frac{1}{s}(b_0 z - a_0 y) && v_1(0) = 0 \\
v_2 &= \frac{1}{s}(b_1 z - a_1 y + v_1) && v_2(0) = 0 \\
v_3 &= \frac{1}{s}(b_2 z - a_2 y + v_2) && v_3(0) = 0 \\
v_4 &= \frac{1}{s}(b_3 z - a_3 y + v_3) && v_4(0) = 0\,.
\end{aligned} \tag{3.145}$$

Chapter 4

Higher-order Taylor series method

This chapter introduces the developed method, the Modern Taylor Series Method (MTSM). It is based on previously published works, mainly [103] and [pp8, pp13, pp28, pp21, pp22] and other works cited throughout. The Modern Taylor Series Method is a variable step, variable order method for solving differential equations using the Taylor series. The experimental calculations verified that the speed of calculation, accuracy, and stability of the method could exceed commonly used algorithms for numerical solutions of differential equations. The speedup of the calculation is significant, especially when solving large systems of differential equations.

The Modern Taylor Series Method is based on the **recurrent calculation** of the Taylor series terms for each time interval. Therefore, the complicated calculation of higher-order derivatives (much criticized in the literature) need not be performed, but rather the value of each Taylor series term is numerically calculated [103].

An important part of the method is the **automatic integration order setting**, using as many Taylor series terms as defined accuracy requires. Usually, the computation uses different numbers of Taylor series terms for different steps of constant length. On the other hand, for a pre-set integration order, the integration step length may be selected. This fact positively affects the stability and speed of computation.

A necessary part of the Modern Taylor Series Method is **automatic transformation** of the original problem. The original system of differential equations is automatically transformed to a polynomial form, that is, to a form suitable for calculating the Taylor series terms using recurrent formulas [118]. The Modern Taylor Series Method also has some favorable properties for parallel processing. Many calculation operations are independent, and calculations can be performed independently using separate processors of parallel computing systems. After the automatic transformation of the task, only basic mathematical operations (addition, subtraction, multiplication, division) can be performed during the calculation of the transformed system. It is possible to design simple specialized elementary processors and obtain a parallel computing system with a relatively simple architecture [97, 98]. The characteristic property of the Modern Taylor Series Method is that using more Taylor series terms during the computation results in higher computation accuracy for a given step size. Of course, the increase in accuracy is limited. The saturated computation error for a given step size depends on the word length of the arithmetic unit. Sometimes, the saturation error can be reduced by decreasing the integration step size or extending the word length of the arithmetic unit [95].

4.1 State of the art

The Taylor Series Method is an integration method with a rich history. Among the first authors to outline the recurrent calculation of the Taylor series term in canonical form were Gibbons [65], and Moore [137]. Taylor coefficients of a function can be obtained by using rules for automatic differentiation of elementary functions. The automatic differentiation technique is described in [19, 20, 33, 68, 105, 136, 165]. The variable step size variable order scheme is described in [15, 18, 121]. The approach based on an approximate formulation of the Taylor methods can be found in [9]. Numerical calculations of higher-order derivatives up to the fourth order are shown in [119, 122]. The Taylor series method can successfully solve some problems that other schemes cannot solve [16, 141].

The first attempts to program the recurrent Taylor series were performed in FORTRAN [76]. For example, ATOMF [31] written in Fortran 77, and several C/C++ packages, TADIFF [22], COSY INFINITY [116], TIDES software [1, 17], and TAYLOR [89]. Other implementations based on the Taylor series include DAETS [126, 127, 128]. The first implementation of MTSM was TKSL/386 [102]. Currently, MTSM has been implemented and tested in MATLAB software[1], in C/C++ languages (FOS [94] and TKSL/C software[2]). Additionally, the method can be implemented effectively on the hardware [pp4].

The parallel high-precision numerical solution of ODEs using high-order Taylor methods is discussed in [14]. Selected problems are Arenstorf orbits and a galactic dynamics model. The numerical tests were performed on a Sun UltraSPARC-II with four processors of 480 MHz using the MPI and the multiple-precision Fortran library mpf90. The authors noted that the number of inter-process communications is high; therefore, this approach is useful only for high-precision demands.

The article [41] focuses on the OpenMP parallelization of multiple precision Taylor series method using one computational node. The model problem is the chaotic dynamic system – the classical Lorenz system. The application is written in C programming language with the GMP library (The GNU Multiple Precision Arithmetic Library[3]). The paper also briefly mentions the concept of CNS (Clean Numerical Simulation) originally published by Shijun Liao [109]. Chaotic dynamic systems have a sensitive dependence on initial conditions (SDIC); therefore, accurate long-term prediction of chaotic dynamic systems is almost impossible. The CNS provides reliable chaotic trajectories in a long enough interval of time. The paper presents a simulation for 1–28 cores on one CPU node, in the interval $[0, 5000]$, using the $N = 2000$ order of the Taylor series method and precision of 8000 bits (approximately 2412 decimal digits) and five integration steps were performed. The results show that the parallel efficiency for 28 cores is $\approx 75\%$. The experiment took approximately nine days and 14 hours.

A hybrid MPI+OpenMP parallelization strategy for the multiple-precision Taylor series method with fixed step size and fixed order is discussed in [81]. Again, the libraries for multiple precision arithmetics are used. Namely, GMP and MPIGMP[4] [96]. The hybrid strategy was used because OpenMP scalability is slightly better than MPI when using one computational node. The speedup depends on the order of the Taylor series method and the hardware properties of the HPC cluster. The authors claimed that this hybrid strategy could be applied to a large class of chaotic dynamical systems.

[1] https://www.mathworks.com/help/matlab
[2] https://www.fit.vutbr.cz/~satek/MTSM/tkslc.html
[3] http://gmplib.org
[4] https://na-inet.jp/na/bnc

The article [82] is based on the previous article [81] and introduces a modification of the CNS with variable step size and fixed order. The source code is similar to that in [81]. It only adds one OpenMP section to compute the optimal integration step. The optimal integration steps are based on the approach published in [89], where the last two terms of the Taylor series determine the optimal step size. The authors found out that the order of the Taylor series method is higher in comparison with the fixed-order approach. However, the larger integration step size reduces the number of integration steps. Also, the higher-order method increases parallel efficiency. The model problem is again a classical Lorenz system, and the simulation experiments are performed on 256 CPUs in the long time interval [0,11000]. Compared to the fixed step size approach, the variable step size strategy has 2.1 speedup. There are two reasons why the variable step size strategy has a higher speedup than the fixed step size strategy. The first is less overall work. Although the computational work per integration step increases by $\approx 80\%$, the average step size was ≈ 0.034 compared to the fixed step size strategy ≈ 0.01. Thus, the overall work decreases to $\approx 53\%$.

For the first experiment, the parallel efficiency increases from 55.5% to 63.6%, and for the second, from 56.2% to 64.3%. The reason is that using the higher-order Taylor series method increases the amount of parallel work. This ensures a lower impact on the serial amount of work and the parallel overhead.

4.2 Motivational example

Let us consider the following functions

$$\begin{aligned} u &= \sin(\omega t) \\ v &= \cos(\omega t)\,. \end{aligned} \tag{4.1}$$

The system (4.1) can be represented using the following system of ODEs

$$\begin{aligned} u' &= \omega v, \quad u(0) = 0 \\ v' &= -\omega u, \quad v(0) = 1\,. \end{aligned} \tag{4.2}$$

The matrix-vector representation of the problem (4.2) is $\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{b}\,, \quad \mathbf{y}(t_0) = \mathbf{y}_0$, where

$$\mathbf{A} = \begin{pmatrix} 0 & \omega \\ -\omega & 0 \end{pmatrix}, \quad \mathbf{y}_0 = (u, v)^T = (0, 1)^T, \quad \mathbf{b} = (0, 0)^T\,. \tag{4.3}$$

The behavior of the system of ODEs (4.2) strongly depends on the parameter ω. The results are calculated using the MTSM solver, implemented in MATLAB software using vectorization, and state-of-the-art non-stiff MATLAB solvers ode23, ode45, and ode113 [158].

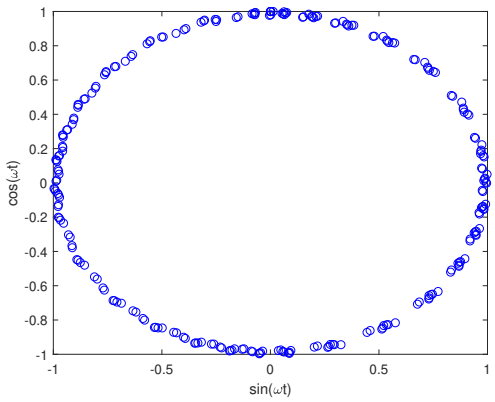
All experiments were performed using MATLAB software. For all experiments, the maximum simulation time is $t_{max} = 50$, and the maximal order of MTSM is $maxORD = 64$. Unless otherwise specified, the integration step of the MTSM is $h = 0.1$. The numerical results are compared with the analytic solution (4.1). The error (denoted as ||error||) is defined as

$$||\mathbf{error}||_\infty = \max(absolute\_error\_sin + absolute\_error\_cos)\,. \tag{4.4}$$

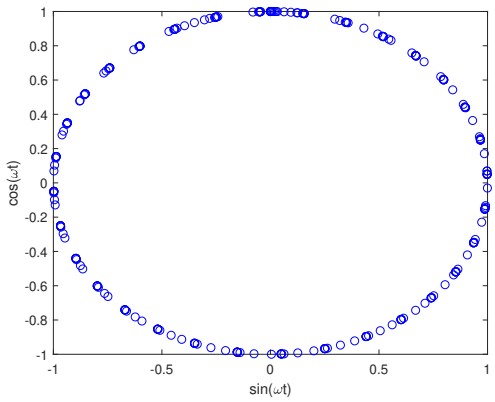
Ratios of computation times (denoted as ratio), $ratio = solver/MTSM \gg 1$ indicates significantly faster computation using the MTSM.

4.2.1 Experiment 1

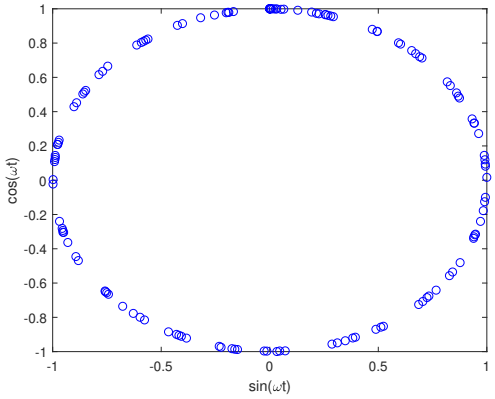
For the first experiment, $\omega = 1$, the tolerances for all ode solvers are $RelTol = 1e^{-3}$ and $AbsTol = 1e^{-6}$ (default MATLAB settings), and the tolerance for the MTSM solver is $tol = 1e^{-3}$. In the phase plane, the expected solution of (4.3) should form a circle. Figure 4.1 shows the results calculated by ode23, ode45, ode113, and MTSM, respectively. The average order of MTSM is 6.



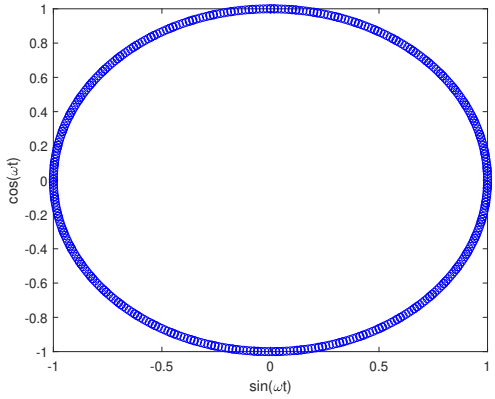
(a) Phase plane ode23



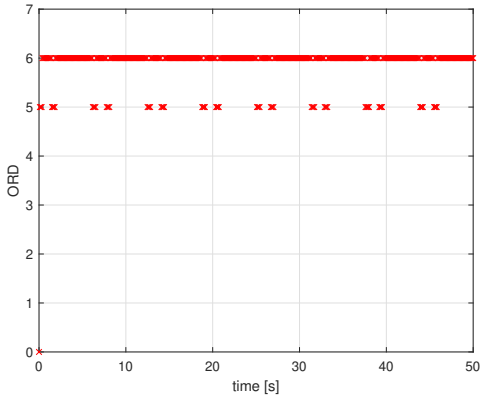
(b) Phase plane ode45



(c) Phase plane ode113



(d) Phase plane MTSM



(e) Order of the MTSM.

Figure 4.1: Results for $\omega = 1$, $tol = 1e^{-3}$, $RelTol = 1e^{-3}$, $AbsTol = 1e^{-6}$ (default settings).

The results in Table 4.1 show that the MTSM solver is the slowest of the selected ode solvers. The MTSM is approximately six times faster than the ode15s solver due to the low stiffness of the system. However, we can notice that due to the stopping rule, the accuracy of the MTSM solver is $||error|| = 7.59e^{-6}$, which is much higher than the required accuracy $tol = 1e^{-3}$.

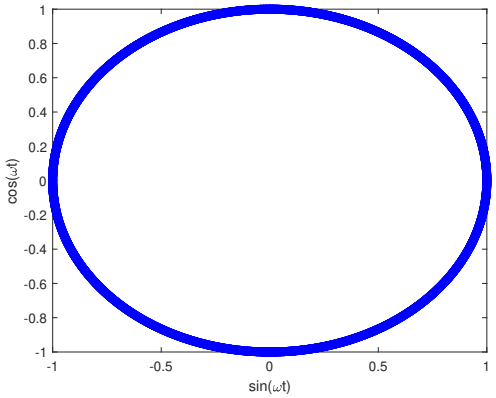
solver	**# steps**	**time [s]**	**||error||**	**ratio**
MTSM	500	$2.89e^{-02}$	$7.59e^{-06}$	–
ode23	244	$1.10e^{-02}$	$3.66e^{-02}$	0.38
ode45	276	5.98^{e-03}	$7.31e^{-03}$	0.21
ode113	133	$6.33e^{-03}$	$1.11e^{-02}$	0.22
ode15s	185	$1.72e^{-01}$	5.71^{e-02}	5.97

Table 4.1: Results for $\omega = 1$, $tol = 1e^{-3}$, $RelTol = 1e^{-3}$, $AbsTol = 1e^{-6}$ (default settings)

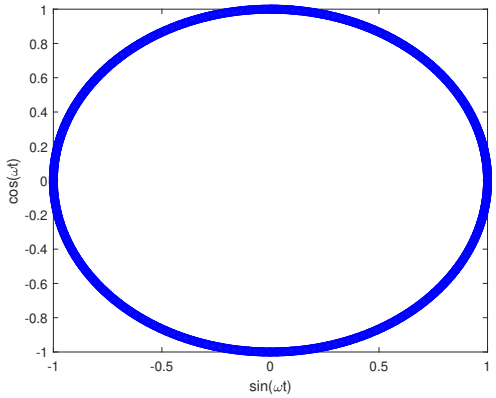
The following experiment sets the same level of accuracy for the ode solvers and MTSM solvers. This is achieved by setting the relative and absolute tolerances of the ode solvers to $RelTol = AbsTol = 1e^{-7}$.

4.2.2 Experiment 2

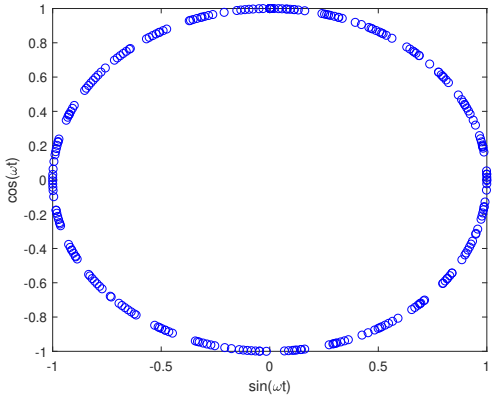
As mentioned above, consider $\omega = 1$ and tolerances of the ode solvers $RelTol = AbsTol = 1e^{-7}$, the tolerance of the MTSM solver remains the same as in the previous experiment; thus $tol = 1e^{-3}$. Numerical results are plotted in Figure 4.2 and summarized in Table 4.2.



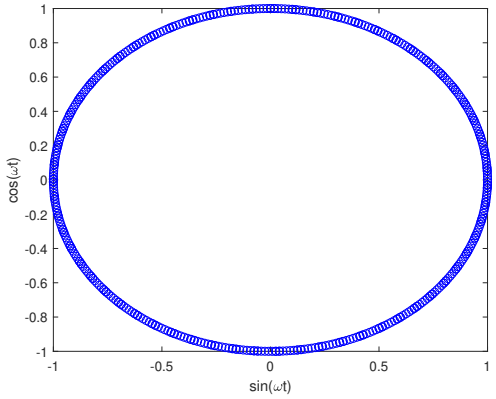
(a) Phase plane ode23



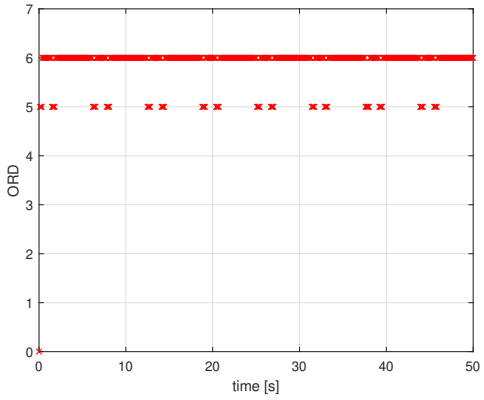
(b) Phase plane ode45



(c) Phase plane ode113



(d) Phase plane MTSM



(e) Order of the MTSM.

Figure 4.2: Results for $\omega = 1$, $tol = 1e^{-3}$, $RelTol = 1e^{-7}$, $AbsTol = 1e^{-7}$.

The results in Table 4.2 show that the MTSM method is comparable to ode113. The MTSM is almost two times faster than the ode45 and seven times faster than the ode23 solver.

solver	# steps	time [s]	||error||	ratio
MTSM	500	$3.04e^{-02}$	$7.59e^{-06}$	–
ode23	3574	$2.13e^{-01}$	$8.04e^{-06}$	7.01
ode45	1484	$5.34e^{-02}$	$8.95e^{-07}$	1.75
ode113	259	$3.20e^{-02}$	$2.56e^{-06}$	1.05
ode15s	644	$1.53e^{-01}$	$3.85e^{-05}$	5.04

Table 4.2: Results for $\omega = 1$, $tol = 1e^{-3}$, $RelTol = 1e^{-7}$, $AbsTol = 1e^{-7}$.

The MTSM solver can be even faster than the ode solvers by increasing the integration step to $h = 10$. In this case, the tolerance of the MTSM solver is increased to $tol = 1e^{-5}$ to maintain the required accuracy.

4.2.3 Experiment 3

For this experiment, the $\omega = 1$, tolerance of MTSM is $tol = 1e^{-5}$ and for ode solvers $RelTol = 1e^{-7}$, $AbsTol = 1e^{-7}$. The integration step of the MTSM is set to $h = 10$. The results in Table 4.3 show that MTSM is much faster than all selected state-of-the-art ode solvers. Because we increased the integration step h, the number of Taylor series terms also increased. The average order of the MTSM is 38, see Figure 4.3.

solver	# steps	time [s]	||error||	ratio
MTSM	5	$1.71e^{-03}$	$1.23e^{-06}$	–
ode23	3574	$1.57e^{-01}$	$8.04e^{-06}$	91.60
ode45	1484	$3.92e^{-02}$	$8.95e^{-07}$	22.90
ode113	259	$1.70e^{-02}$	$2.56e^{-06}$	9.92
ode15s	644	$9.23e^{-02}$	$3.85e^{-05}$	53.90

Table 4.3: Results for $\omega = 1$, $tol = 1e^{-5}$, $RelTol = 1e^{-7}$, $AbsTol = 1e^{-7}$, $h = 10$.

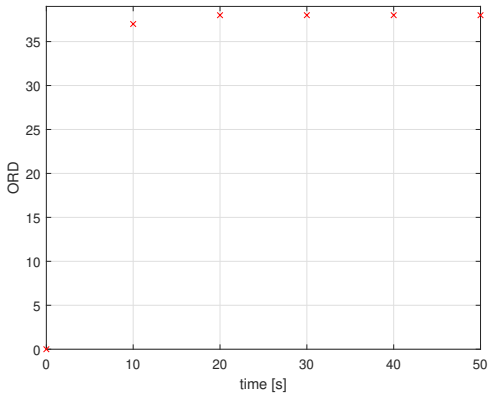
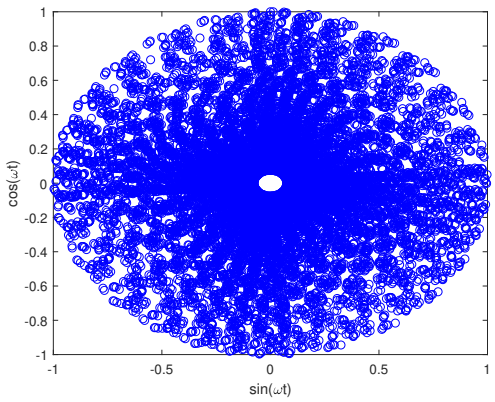


Figure 4.3: Order of the MTSM.

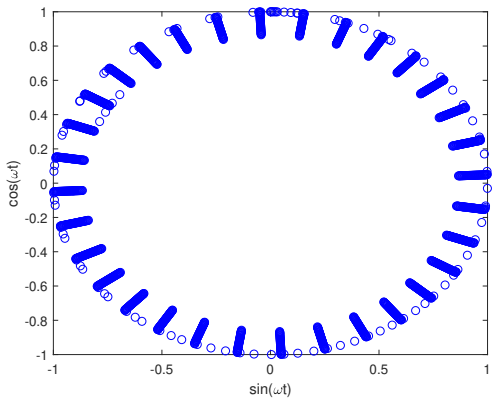
4.2.4 Experiment 4

This experiment shows the behavior of the system for $\omega = 100$. The higher value of ω increases the stiffness of the system. First, let us consider the default tolerance settings for ode solvers. Therefore, $RelTol = 1e^{-3}$, $AbsTol = 1e^{-6}$, the tolerance of the MTSM is $tol = 1e^{-3}$ and $h = 0.1$.

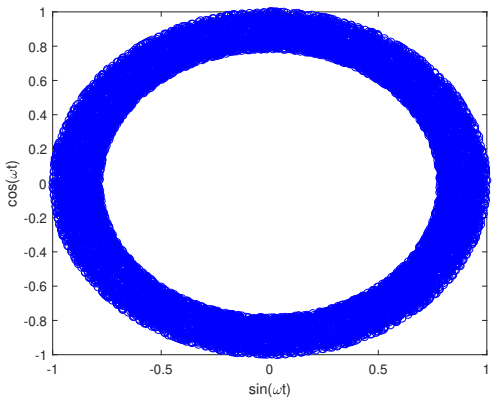
Only MTSM provides a stable numerical solution, but the numerical solution calculated by other ode solvers diverges. Figure 4.4 shows the results calculated by ode23, ode45, ode113, and MTSM, respectively. Note that the average order of MTSM is 33.



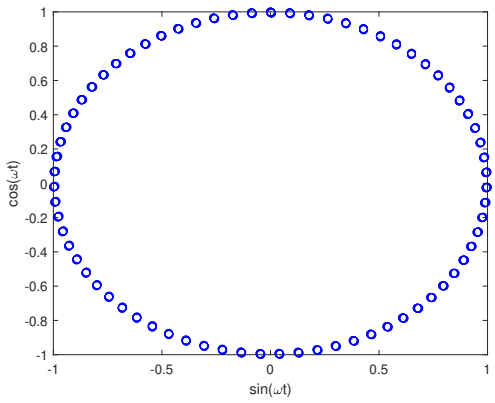
(a) Phase plane ode23



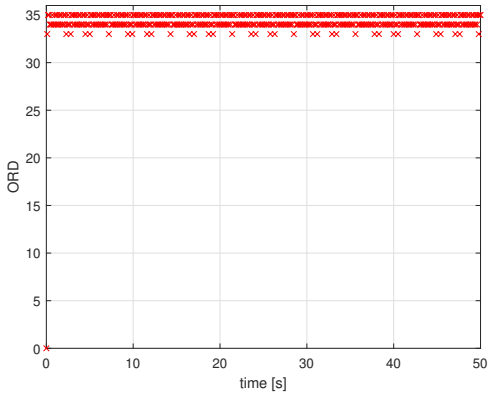
(b) Phase plane ode45



(c) Phase plane ode113



(d) Phase plane MTSM



(e) Order of the MTSM.

Figure 4.4: Results for $\omega = 100$, $tol = 1e^{-3}$, $RelTol = 1e^{-3}$, $AbsTol = 1e^{-6}$ (default settings).

Results of the calculation are in Table 4.4. Only the MTSM can maintain the predefined accuracy. The accuracy of the ode solvers is not acceptable.

solver	**# steps**	**time [s]**	**||error||**	**ratio**
MTSM	500	$9.60e^{-02}$	$1.22e^{-02}$	–
ode23	23915	1.19	1.32	12.40
ode45	25488	$6.67e^{-01}$	$6.59e^{-01}$	6.95
ode113	12634	$7.64e^{-01}$	$5.19e^{-01}$	7.95
ode15s	17531	2.84	6.43	29.60

Table 4.4: Results for $\omega = 100$, $tol = 1e^{-3}$, $RelTol = 1e^{-3}$, $AbsTol = 1e^{-6}$ (default settings).

4.2.5 Experiment 5

In this experiment, we increase the tolerance of all solvers to maintain accuracy at the same level as in Experiment 1. Therefore, $\omega = 100$, $h = 0.1$, the tolerance of the MTSM is increased to $tol = 1e^{-6}$, and the tolerances for the ode solvers are set to $RelTol = 1e^{-9}$, $AbsTol = 1e^{-9}$.

Figure 4.5 shows that the order of MTSM is 40, which is close to the upper limit for double arithmetic. Therefore, we cannot increase the integration step size dt; this would lead to a halving of the step size because the maximum order of MTSM $maxORD = 64$ will be reached. The MTSM can be calculated with an arbitrary tolerance and step size if variable precision arithmetic is used.

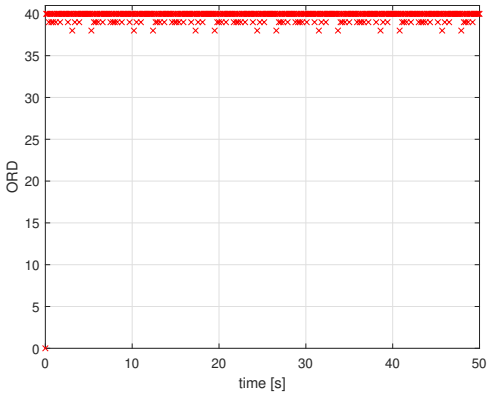


Figure 4.5: Order of MTSM

Results are summarized in Table 4.5. The MTSM is significantly faster than all state-of-the-art ode solvers.

solver	**# steps**	**time [s]**	**||error||**	**ratio**
MTSM	500	$1.91e^{-01}$	$6.48e^{-06}$	–
ode23	1658768	$7.99e^{01}$	$8.12e^{-06}$	417
ode45	371524	9.17	$8.94e^{-07}$	47.90
ode113	38623	1.90	$2.88e^{-07}$	9.94
ode15s	132897	$1.91e^{+01}$	$8.89e^{-05}$	99.60

Table 4.5: Results for $\omega = 100$, $tol = 1e^{-6}$, $RelTol = 1e^{-9}$, $AbsTol = 1e^{-9}$

4.3 Recurrent calculation of Taylor series terms

An ODE

$$y' = f(t, y)\,, \tag{4.5}$$

with initial condition

$$y(t_0) = y_0\,, \tag{4.6}$$

The best known and most accurate method for calculating a new value of the numerical solution of an ODE is to construct the Taylor series in the form

$$y_{i+1} = y_i + h \cdot f(t_i, y_i) + \frac{h^2}{2!} \cdot f'(t_i, y_i) + \cdots + \frac{h^n}{n!} \cdot f^{[n-1]}(t_i, y_i), \tag{4.7}$$

where h is the 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 $y(t)$ [74].

The MTSM effectively implements the variable step size, variable order numerical calculation of differential equations using the Taylor series [103]. It is based on a recurrent calculation of the Taylor series terms for each integration step. Therefore, it is not necessary to perform a complicated calculation of higher order derivatives, but instead the value of each Taylor series term can be numerically calculated [103]. Equation (4.7) can then be rewritten in the form

$$y_{i+1} = DY_{i0} + DY_{i1} + DY_{i2} + \cdots + DY_{in}\,, \tag{4.8}$$

where DY_i denotes the Taylor series terms in simulation time $t = i$. The MTSM transforms the input problem into a system of autonomous ODEs, which allows the recurrent calculation of terms of the Taylor series.

To demonstrate the recurrent calculation of Taylor series terms, let us assume the system of ODEs in the following form:

$$\begin{aligned} y' &= a_{11} \cdot y + a_{12} \cdot z + \cdots + a_{1m} \cdot w + b_1 \quad y(0) = y_0 \\ z' &= a_{21} \cdot y + a_{22} \cdot z + \cdots + a_{2m} \cdot w + b_2 \quad z(0) = z_0 \\ &\vdots \\ w' &= a_{m1} \cdot y + a_{m2} \cdot z + \cdots + a_{mm} \cdot w + b_m \quad w(0) = w_0 \,. \end{aligned} \tag{4.9}$$

The system of ODEs (4.9) consists of equations m with constant coefficients a_{kl}, where k denotes the index of the equation, and l is the index of the term, and $k, l \in \langle 1, m \rangle$. Each equation has one constant b_k and an initial condition $(y(0), z(0), \ldots, w(0))$. The system of ODEs (4.9) can be rewritten to (4.10). The first terms of the Taylor series are denoted as $DY_{i1}, DZ_{i1}, \ldots, DW_{i1}$:

$$\begin{aligned} DY_{i1} &= h \cdot y'_n = h \cdot (a_{11} \cdot y + a_{12} \cdot z + \cdots + a_{1m} \cdot w + b_1) \\ DZ_{i1} &= h \cdot z'_n = h \cdot (a_{21} \cdot y + a_{22} \cdot z + \cdots + a_{2m} \cdot w + b_2) \\ &\vdots \\ DW_{i1} &= h \cdot w'_n = h \cdot (a_{m1} \cdot y + a_{m2} \cdot z + \cdots + a_{mm} \cdot w + b_m) \,. \end{aligned} \tag{4.10}$$

The second terms of the Taylor series $DY_{i2}, DZ_{i2}, \ldots, DW_{i2}$ can be calculated as follows:

$$\begin{aligned} DY_{i2} &= \frac{h}{2} \cdot DY'_{i1} = \frac{h}{2} \cdot (a_{11} \cdot DY_{i1} + a_{12} \cdot DZ_{i1} + \cdots + a_{1m} \cdot DW_{i1} + b_1) \\ DZ_{i2} &= \frac{h}{2} \cdot DZ'_{i1} = \frac{h}{2} \cdot (a_{21} \cdot DY_{i1} + a_{22} \cdot DZ_{i1} + \cdots + a_{2m} \cdot DW_{i1} + b_2) \\ &\vdots \\ DW_{i2} &= \frac{h}{2} \cdot DW'_{i1} = \frac{h}{2} \cdot (a_{m1} \cdot DY_{i1} + a_{m2} \cdot DZ_{i1} + \cdots + a_{mm} \cdot DW_{i1} + b_m) \,. \end{aligned} \tag{4.11}$$

The higher-order Taylor series terms can be calculated analogously. If the stopping rule (4.13) is met, the result in the current integration step can be calculated as a sum of Taylor series terms and the initial condition of the integration step:

$$\begin{aligned} y_{i+1} &= y_i + DY_{i1} + DY_{i2} + \cdots + DY_{iORD_i} \\ z_{i+1} &= z_i + DZ_{i1} + DZ_{i2} + \cdots + DZ_{iORD_i} \\ &\vdots \\ w_{i+1} &= w_i + DW_{i1} + DW_{i2} + \cdots + DW_{iORD_i} \,. \end{aligned} \tag{4.12}$$

The results in the following integration steps can be calculated similarly. The results of (4.12) are used as an initial condition in the next time step. More information can be found in [103], [pp28].

4.4 Automatic integration order setting

An important part of the method is an automatic integration order setting, using as many Taylor series terms as the defined accuracy requires. Let us denote as ORD the function that changes during the computation and defines the number of Taylor series terms used in the current integration step ($ORD_{i+1} = n$). The MTSM allows computation with arbitrary accuracy and step size if variable-precision arithmetic and higher-order of method are used. The Taylor series terms can be used in error control [pp28]. The current integration step is completed when the following stopping rule is met:

$$||max(abs(\mathbf{DY}_j))|| \leq tol\,, \quad j = n - stopping, \cdots, n, \tag{4.13}$$

where $\mathbf{DY}_j$ is a vector of three successive terms of the Taylor series, tol is the error per step, n is the index of currently calculated Taylor series term, and $stopping$ denotes the number of successive terms of the Taylor series that have met the stopping rule (4.13). In this work, $stopping$ is set to three. For more information, see [103].

4.5 Automatic transformation

The general form of **linear non-homogeneous system of ODEs** is defined as

$$\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{f}(t)\,, \quad \mathbf{y}(t_0) = \mathbf{y}_0\,, \tag{4.14}$$

where $\mathbf{A}$ is the constant Jacobian matrix and $\mathbf{f}(t, y)$ is the right-hand side (the forcing function).

Automatic transformation is an important part of the MTSM. Basic differentiation rules can be used to derive the recurrent calculation of higher-order derivatives for basic arithmetic functions, for example, $\sin, \cos, \log, \exp$, and basic arithmetic operations. For the automatic recurrent calculation of Taylor series terms, several rules have to be followed:

- The functions $\mathbf{f_i}(t, y_1, y_2, \ldots, y_n)$ of the right sides of the system of n ordinary differential equations are composed of a finite number of elementary functions and a finite number of simple arithmetic operations.
- All intermediate calculations and all generating functions lead to auxiliary variables.
- Operands of intercomputing operations and generating functions use auxiliary variables, $t, y_1, y_2, \ldots, y_n$, and constants.

The function $\mathbf{f}(t, y)$ can be transformed into the system of generating ODEs. As a result, the non-homogeneous system of ODEs (4.14) is transformed into the homogeneous linear system of ODEs based on the definition (2.1.6):

$$\mathbf{z}' = \mathbf{F}(\mathbf{z})\,, \quad \mathbf{z}(t_0) = \mathbf{z}_0\,, \tag{4.15}$$

where the right-hand side $\mathbf{F}(\mathbf{z})$ is in a special form suitable for calculating higher-order derivatives. Equation (4.15) must meet the condition (4.16):

$$z_i(t) = y_i(t), \quad i = 1, \ldots, m\,, \tag{4.16}$$

where $\mathbf{y}(t) = (y_1(t), \ldots, y_m(t))$ is the original solution of system of ODEs (4.30). The automatic transformation process will be demonstrated using the system of two ODEs

$$\begin{aligned} y_1' &= a_{11} \cdot y_1 + a_{12} \cdot y_2 + f_1(t) \quad y_1(0) = y_{1,0} \\ y_2' &= a_{21} \cdot y_1 + a_{22} \cdot y_2 + f_2(t) \quad y_2(0) = y_{2,0}\,. \end{aligned} \tag{4.17}$$

Let us assume that $f_1(t)$ and $f_2(t)$ in the system of ODEs (4.17) are

$$f_1(t) = e^{-t} = y_3 \tag{4.18}$$
$$f_2(t) = \sin t = y_4\,. \tag{4.19}$$

By deriving a system of ODEs for (4.18) and (4.19), the system of generating ODEs can be obtained. For the equation (4.18):

$$y_3 = e^{-t} \quad y_3(0) = y_{3,0} = 1 \tag{4.20}$$
$$y_3' = -e^{-t} \tag{4.21}$$
$$y_3' = -y_3\,. \tag{4.22}$$

For equation (4.19):

$$y_4 = \sin t \quad y_4(0) = y_{4,0} = 0 \tag{4.23}$$
$$y_4' = \cos t\,. \tag{4.24}$$

After denoting

$$y_5 = \cos t \tag{4.25}$$
$$y_5' = -\sin t \tag{4.26}$$

the system of two ODEs is obtained

$$y_4' = y_5 \tag{4.27}$$
$$y_5' = -y_4\,. \tag{4.28}$$

As a result, the original system of ODEs (4.17) can be rewritten in the form (4.29)

$$\begin{aligned} y_1' &= a_{11} \cdot y_1 + a_{12} \cdot y_2 + y_3 & y_1(0) &= y_{1,0} \\ y_2' &= a_{21} \cdot y_1 + a_{22} \cdot y_2 + y_4 & y_2(0) &= y_{2,0} \\ y_3' &= -y_3 & y_3(0) &= y_{3,0} = 1 \\ y_4' &= y_5 & y_4(0) &= y_{4,0} = 0 \\ y_5' &= -y_4 & y_5(0) &= y_{5,0} = 1\,. \end{aligned} \tag{4.29}$$

Tables of the generating system of ODEs B.3 and B.4 are in Appendix B.

4.6 Linear MTSM

The general form of **linear homogeneous systems of ODEs** is

$$\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{b}\,, \quad \mathbf{y}(t_0) = \mathbf{y}_0\,, \tag{4.30}$$

the Taylor series (4.7) can be rewritten in matrix-vector notation as

$$\mathbf{y}_{i+1} = \mathbf{y}_i + h(\mathbf{A}\mathbf{y}_i + \mathbf{b}) + \frac{h^2}{2!}\mathbf{A}(\mathbf{A}\mathbf{y}_i + \mathbf{b}) + \cdots + \frac{h^n}{n!}\mathbf{A}^{(n-1)}(\mathbf{A}\mathbf{y}_i + \mathbf{b})\,, \tag{4.31}$$

where $\mathbf{A}$ is the constant Jacobian matrix of size $m \times m$, and $\mathbf{b}$ is the right-hand side constant vector of size $m \times 1$. Moreover, Taylor series terms $\mathbf{DY}$ (4.8) can be computed recurrently using

$$\begin{aligned} &\mathbf{DY}_0 = \mathbf{y}_i\,, \quad \mathbf{DY}_1 = h(\mathbf{A}\mathbf{y}_i + \mathbf{b}), \\ &\mathbf{DY}_j = \frac{h}{j}\mathbf{A}\mathbf{DY}_{j-1}\,, \quad j = 2, \ldots, n\,. \end{aligned} \tag{4.32}$$

4.7 Nonlinear MTSM

The effective solution of nonlinear systems of ODEs is described [pp21]. For such a system, the Taylor series-based numerical method can be implemented very effectively. Equation (4.5) for nonlinear systems of ODEs can be rewritten as

$$\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{B}_1\mathbf{y}_{jk} + \mathbf{B}_2\mathbf{y}_{jkl} + \ldots + \mathbf{b}\,, \quad \mathbf{y}(0) = \mathbf{y}_0\,, \tag{4.33}$$

where $\mathbf{A} \in \mathbb{R}^{ne \times ne}$ is the matrix for the linear part of the system and $\mathbf{B}_1 \in \mathbb{R}^{ne \times nm_{jk}}$, $\mathbf{B}_2 \in \mathbb{R}^{ne \times nm_{jkl}}$ are the matrices for the nonlinear part of the system, $\mathbf{b} \in \mathbb{R}^{ne}$ is the right-hand side for the forces incoming to the system, $\mathbf{y}_0$ is a vector of initial conditions, the symbol ne stands for the number of equations of the system of ODEs and the symbols nm_{jk}, nm_{jkl} mean the number of multiplications. The unknown function $\mathbf{y}_{jk} \in \mathbb{R}^{nm_{jk}}$ represents the vector of multiplications $\mathbf{y}_j .* \mathbf{y}_k$ and similarly $\mathbf{y}_{jkl} \in \mathbb{R}^{nm_{jkl}}$ represents the vector of multiplications $\mathbf{y}_{jj} .* \mathbf{y}_{kk} .* \mathbf{y}_{ll}$, where the indices $j, k, jj, kk, ll \in (1, \ldots, ne)$ come from the multiplications terms in the system (4.33). The operation '.*' stands for *element-by-element* multiplication, i.e. $\mathbf{y}_j .* \mathbf{y}_k$ 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 $\mathbf{A}, \mathbf{B}_1, \mathbf{B}_2$ and the vector $\mathbf{b}$ are constant. The higher derivatives of the terms $\mathbf{B}_1\mathbf{y}_{jk}, \mathbf{B}_2\mathbf{y}_{jkl}$ in (4.33) can be included in a recurrent calculation of the Taylor series terms $\mathbf{DY}_{B1}$ and $\mathbf{DY}_{B2}$

$$\begin{aligned} \mathbf{DY}_{B1_1} &= h(\mathbf{B}_1\mathbf{y}_{jk})\,, \mathbf{DY}_{B2_1} = h(\mathbf{B}_2\mathbf{y}_{jkl})\,, \\ \mathbf{DY}_{B1_r} &= \frac{h}{r}\left(\mathbf{B}_1 \sum_{p=1}^{r} \mathbf{DY}_{j,r-p} .* \mathbf{DY}_{k,p-1}\right)\,, \\ \mathbf{DY}_{B2_r} &= \frac{h}{r}\mathbf{B}_2 \sum_{q=0}^{r-1} \mathbf{DY}_{jj,q} .* \left(\sum_{p=1}^{r} \mathbf{DY}_{kk,r-p-q} .* \mathbf{DY}_{ll,p-1}\right)\,, \\ \mathbf{DY}_r &= \mathbf{DY}_{A_{r-1}} + \mathbf{DY}_{B1_{r-1}} + \mathbf{DY}_{B2_{r-1}}\,, r = 2, \ldots, n\,, \end{aligned} \tag{4.34}$$

where the linear term $\mathbf{DY}_{A_{r-1}}$ is calculated using (4.32). Higher-order multiplication terms in the Taylor series $\mathbf{DY}_{B3}, \mathbf{DY}_{B4}, \ldots$ can be calculated recurrently in a similar way.

4.8 Practical examples

This section introduces several linear and nonlinear problems solved using the MTSM, and state-of-the-art non-stiff solvers in MATLAB, specifically ode23, ode45, and ode113 [158]. The MTSM solver was implemented in MATLAB software using vectorization.

Linear problems include the movement of the charged particle in the electromagnetic and electrostatic field [pp21, pp22], and the linearized pendulum [pp21]. Nonlinear problems are Lorenz system [pp13], numerical calculation of Fourier coefficients [pp21, pp22], Kepler problem [pp22], nonlinear pendulum [pp21], Van der Pol oscillator and forced van der Pol oscillator [pp28].

4.9 General parallelization of the linear system of ODEs

The Modern Taylor Series Method is the variable step size, variable order method. For more information, see [pp14, pp16, pp17, pp24]. The best-known and most accurate method to calculate a new value of the numerical solution of the ODE

$$y' = f(t, y)\,, \quad y(t_0) = y_0 \tag{4.35}$$

is to construct the Taylor series in the form

$$y_{i+1} = y_i + hf(t_i, y_i) + \frac{h^2}{2!}f'(t_i, y_i) + \ldots + \frac{h^n}{n!}f^{[n-1]}(t_i, y_i)\,, \tag{4.36}$$

where h is the integration step, y_i and y_{i+1} are the next approximations of the value of the function $y(t)$, $y(t_i+h)$, respectively [74]. Let ORD denote the function that changes during the computation and define the number of Taylor series terms in the current integration step ($ORD_{i+1} = n$). For linear systems of ODEs $\mathbf{y}' = \mathbf{Ay} + \mathbf{b}$, where $\mathbf{A}$ is the constant Jacobian matrix, and $\mathbf{b}$ is the constant right-hand side, equation (4.36) can be rewritten in matrix-vector notation

$$\mathbf{y}_{i+1} \;=\; \mathbf{y}_i + h(\mathbf{Ay}_i + \mathbf{b}) + \tfrac{h^2}{2!}\mathbf{A}(\mathbf{Ay}_i + \mathbf{b}) + \cdots + \tfrac{h^n}{n!}\mathbf{A}^{n-1}(\mathbf{Ay}_i + \mathbf{b})\,. \tag{4.37}$$

Equation (4.37) can be rewritten in the form

$$\mathbf{y}_{i+1} = \mathbf{DY}_0 + \mathbf{DY}_1 + \mathbf{DY}_2 + \cdots + \mathbf{DY}_n\,, \tag{4.38}$$

where individual Taylor series terms can be computed recurrently [pp21]. The integration step is completed when the stopping rule is met (4.13). The resulting system can be effectively solved sequentially or in parallel. Equation (4.37) can be rewritten as follows:

$$\mathbf{y}_{i+1} = \mathbf{A}_y\mathbf{y}_i + \mathbf{A}_b\mathbf{b}\,, \tag{4.39}$$

where the matrices $\mathbf{A}_y$ and $\mathbf{A}_b$ are defined as:

$$\mathbf{A}_y = \sum_{k=0}^{n}\frac{h^k}{k!}\mathbf{A}^k\,, \quad \mathbf{A}_b = \sum_{k=1}^{n}\frac{h^k}{k!}\mathbf{A}^{k-1}\,. \tag{4.40}$$

When calculating the matrices $\mathbf{A}_y$ and $\mathbf{A}_b$, the fixed integration step size can be approximated using (4.41):

$$h < \sqrt[n]{\frac{tol \cdot n}{||\mathbf{A}^n||}}\,, \tag{4.41}$$

where $||\cdot||$ denotes the Euclidean norm.

Let $\mathbf{A}_{D_i}$ denote the submatrix of the matrix $\mathbf{A}$ decomposed by rows, where $i = \{1, 2, \cdots, p\}$ and p is the number of processes. Let $\mathbf{A}$ be a matrix of size $m \times m$. Each process owns m/p rows of the matrix $\mathbf{A}$. The matrices $\mathbf{A}_y$ and $\mathbf{A}_b$ are constant, and the matrices are precalculated only once at the beginning of the calculation:

$$\hat{\mathbf{A}}_{yD_i} = \sum_{k=0}^{n} \frac{h^k}{k!} \mathbf{A}_{D_i}^k \,, \quad \hat{\mathbf{A}}_{bD_i} = \sum_{k=1}^{n} \frac{h^k}{k!} \mathbf{A}_{D_i}^{k-1} \,, \quad i = 1, 2, \cdots, p \,. \tag{4.42}$$

Therefore, each process calculates the Taylor series terms recurrently:

$$\mathbf{A}_{D_1} = h\mathbf{I}_D \,, \quad \mathbf{A}_{D_j} = \frac{h}{j} \mathbf{A}_{D_{j-1}} \mathbf{A} \,, \quad j = 2 \ldots n \,, \quad \text{and} \quad \hat{\mathbf{A}}_D = \sum_{k=1}^{n} \mathbf{A}_{D_k} \,. \tag{4.43}$$

After the parallel precalculation, the matrix $\hat{\mathbf{A}}_{yG}$ is obtained by gathering individual matrices $\hat{\mathbf{A}}_{yD_i}$. Similarly, the matrix $\hat{\mathbf{A}}_{bG}$:

$$\hat{\mathbf{A}}_{yG} = \left(\hat{\mathbf{A}}_{yD_1}, \hat{\mathbf{A}}_{yD_2}, \cdots, \hat{\mathbf{A}}_{yD_p} \right)^T \,, \quad \hat{\mathbf{A}}_{bG} = \left(\hat{\mathbf{A}}_{bD_1}, \hat{\mathbf{A}}_{bD_2}, \cdots, \hat{\mathbf{A}}_{bD_p} \right)^T \,. \tag{4.44}$$

The final matrix $\hat{\mathbf{A}}$ and the vector $\hat{\mathbf{b}}$ are calculated afterwards, $\mathbf{I}_D$ is the identity matrix

$$\hat{\mathbf{A}} = \hat{\mathbf{A}}_{yG}\mathbf{A} + \mathbf{I} \,, \qquad \hat{\mathbf{b}} = \hat{\mathbf{A}}_{bG}\mathbf{b} \,. \tag{4.45}$$

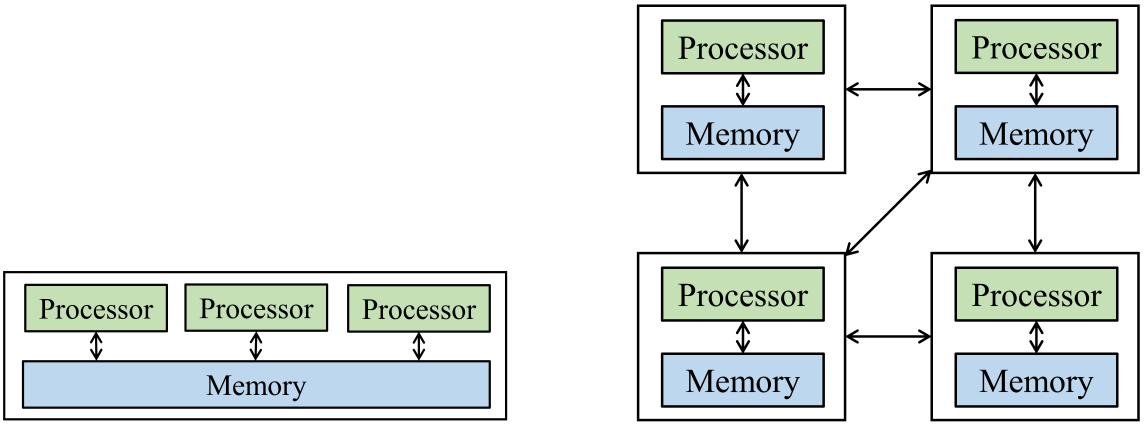
Using (4.45), we can rewrite (4.39) and solve it in parallel using the row-wise decomposition of matrix $\hat{\mathbf{A}}$

$$\mathbf{y}_{i+1} = \hat{\mathbf{A}}\mathbf{y}_i + \hat{\mathbf{b}} \,. \tag{4.46}$$

Chapter 5

Parallel and distributed computing

This chapter provides an overview of parallel and distributed computing. The key difference between parallel and distributed computing is that parallel computing executes multiple tasks using multiple processors simultaneously. In contrast, in distributed computing, multiple computers are interconnected via a network to communicate and collaborate to achieve a common goal (see Figure 5.1). For important information, refer to [5, 24, 46, 67, 73, 135, 148].



(a) Parallel computing

(b) Distributed computing

Figure 5.1: Differences between parallel and distributed computing.

Parallel computing (parallel processing) is the process of decomposing larger problems into smaller, independent, often similar parts that can be executed simultaneously by multiple processors communicating using shared memory. The main goal of parallel computing is to increase the available computation power for faster application processing and problem solving. We distinguish three main levels of parallelism (or types): bit, instruction, and task.

- *Bit-level parallelism* is based on increasing processor word size. Increasing the word size reduces the number of instructions that the processor must execute to perform an operation on variables whose sizes are greater than the length of the word. For example, the addition of two 16-bit integers on the 8-bit processor requires two instructions to complete a single operation. On the contrary, the same operation on the 16-bit processor requires only one instruction.

- *Instruction-level parallelism* means the simultaneous execution of multiple instructions in a program. For example, we can mention pipelining or connecting more functional units of the same type.
- *Task-level parallelism* means concurrent execution of the different tasks on multiple computing cores.

Distributed computing (distributed processing is the technique of connecting multiple computers over a network to a cluster (called a distributed system) to share data and coordinate processing power. Distributed computing offers advantages in scalability, performance, resilience, and cost-effectiveness.

The first parallel computers appeared in the late 1950s. In 1955, IBM introduced the IBM-704. The principal architect was Gene Amdahl, the first commercial machine with floating-point hardware, and it is capable of approximately five kFLOPS. Researchers and computer scientists have published papers on the need for parallel processing to improve computing speed and efficiency.

The 1960s and 1970s brought the first supercomputers, the first computers to use multiple processors. Supercomputers were initially presented in 1964 by Seymour Roger Cray – an American electrical engineer and supercomputer architect credited with creating the supercomputer industry. He designed the first commercially successful supercomputer – Control Data Corporation (CDC) 6600 supercomputer. The CDC 6600 had a single CPU that cost 8 million dollars, could handle 3 million FLOPS, and used vector processors.

In the mid-1980s, researchers at the California Institute of Technology started using massively parallel processors (MPPs) to create a supercomputer for scientific applications. MPP (massively parallel processing) [21, 132] is the coordinated processing of a program by multiple processors that work on different parts of the program. Each processor uses its operating system and memory. MPP processors typically communicate using a messaging interface.

Multi-core processors are the most widespread today. Parallel processing has become even more important because of the emphasis on energy efficiency. Increasing performance through parallel processing is much more energy efficient than increasing the clock frequencies of microprocessors. For more details, see [101, 161].

5.1 Motivating parallelism

The role of parallelism in the acceleration of computing speeds has been recognized for several decades. Its role in providing a multiplicity of data paths and increased access to storage elements has been significant in commercial applications. The scalable performance and lower cost of parallel platforms are reflected in the wide variety of applications. Some unmistakable trends in hardware design indicate that uniprocessor (or implicitly parallel) architectures may not be able to sustain the rate of realizable performance increments in the future because of a number of fundamental physical and computational limitations. This has led to the emergence of standardized parallel programming environments. Libraries and hardware have significantly reduced the time for (parallel) solutions.

5.1.1 Computational power argument

In 1965 Gordon Moore, co-founder and CEO of Intel, made the following observation regarding the transistor density [124]: *Moores's Law*: The number of transistors on a chip

doubles approximately every 1.5 years (later revised to every 2 years), though the cost of computers is halved. Figure 5.2 shows microchip transistor counts from 1970 to 2020. Another principle derived from Moore's Law is that the growth of microprocessors is exponential. Moore's law directly impacts the progress of computing power. Transistors in integrated circuits contain carbon and silicon molecules and can conduct electricity. The faster the circuit conducts electricity, the faster the computer operates.

The limits of Moore's law have been debated in the past few years. The key architectural issue is using transistors to achieve increasing computational rates. The logical solution is to rely on parallelism (both implicit and explicit). In a 2015 interview, Moore describes a couple of potential obstacles associated with miniaturization: the speed of light, the atomic nature of materials, and growing costs. Moore's law is becoming obsolete.

The first factor is the speed of light, which is finite and constant and represents a limitation on the number of computations that can be performed on a single transistor. Bits are modeled by electrons traveling through transistors. Both wires and transistors are characterized by capacitance and resistance. Capacitance C denotes the capacity to store electrons, and resistance R denotes how much the electrons resist the flow of the current. When miniaturizing chips, the resistance R increases, while the capacitance C decreases. Therefore, it is more difficult to perform the correct calculations. James R. Powell predicted that Moore's law would be obsolete in 2036 [133].

Regarding the atomic nature of materials, Robert Colwell, director of the Microsystems Technology Office at the Defense Advanced Research Projects Agency, said that in 2020, 7 nm could be the last process technology node, and even if the industry push to 5 nm sizes, there are not many advantages over 7 nm. Therefore, computers will reach their limits because transistors cannot operate within smaller circuits due to increasingly higher temperatures. Cooling the transistors will require more energy than the energy for the transistor itself.

A possible solution might be quantum computing. Quantum computers are based on qubits (quantum bits) and use superposition and entanglement techniques to overcome the miniaturization problems of classical computing approaches. The most discussed issue is the scalability of quantum computers up to millions of qubits. Another possibility is to use specialized architectures designed for particular algorithms. This area is growing rapidly due to increasing interest in machine learning. Many companies focused on artificial intelligence, such as Google (Tensor Processing Unit – TPUs), Graphcore, Horizon Robotics, etc. Another option is to use FPGA (Field Programmable Gate Array), which Intel and Microsoft use in data centers to accelerate binary search. Similarly to FPGAs, ASIC (Application Specific Integrated Circuit) is another option, recently used in cryptocurrency mining.

There are several other alternatives to classical computing, for example, spintronics. This area is still in the research phase and uses the spin of electrons. Optical computing uses light to perform calculations. Several experiments with non-silicon materials were recently made, including compounded semiconductors (a combination of two or more elements from the periodic table) or biological computing (which uses cells or DNA as the integrated circuit).

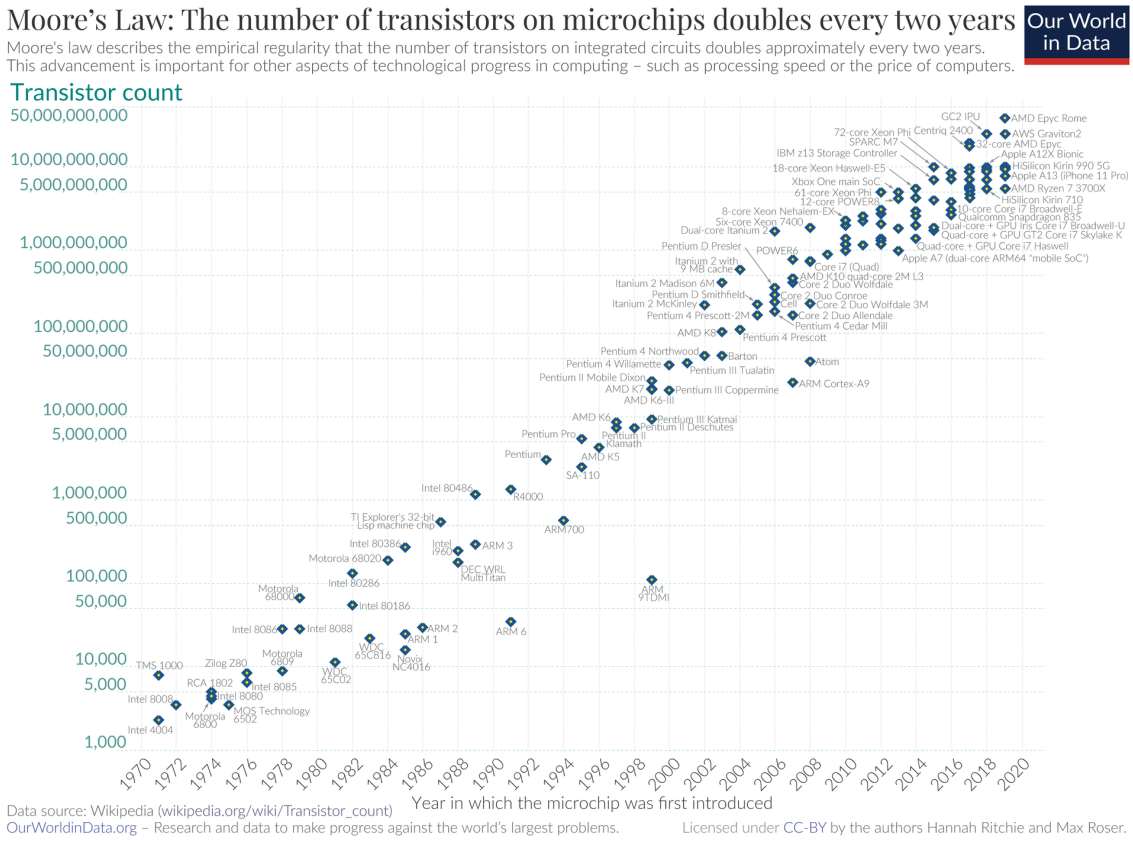


Figure 5.2: Moore's Law 1970-2020: transistor counts in microchips [142].

Figure 5.3[1] shows 50 years of microprocessor trends – transistor density (orange), performance (blue), frequency (green) and number of cores (black). The figure is based on known transistor counts published by Intel, AMD, and IBM's Power processors. It can be observed that the transistor count and power consumption increase, while the frequency and number of logical cores have decreased.

The growth of computing power over the first 25 years was based on the increase in single-thread exponential performance. Many simulation codes were implemented using MPI, released in 1994, allowing computations on clusters and supercomputers to be distributed. MPI became a standard for parallel computations. The codes scaled well, and so no major changes from developers or architects needed to be made.

Although Moore's law was still valid, the Dennard scaling (MOSFET scaling) [39] was not. In 1974, Robert H. Dennard introduced the idea that as the dimension of transistors decreases, so does power consumption. Smaller transistors can run faster, use less power, and cost less. But this scaling has a limit. In 2005, Dennard scaling ended around 2004 because it ignored the leakage current and threshold voltage, establishing a power baseline per transistor. The end of Dennard scaling resulted in a situation where Moore's law still holds, but the performances are not as significant as before. Therefore, the industry focused on new hardware architectures and code paradigms to keep trends on track. For more information, see [125, 143].

[1] https://github.com/karlrupp/microprocessor-trend-data

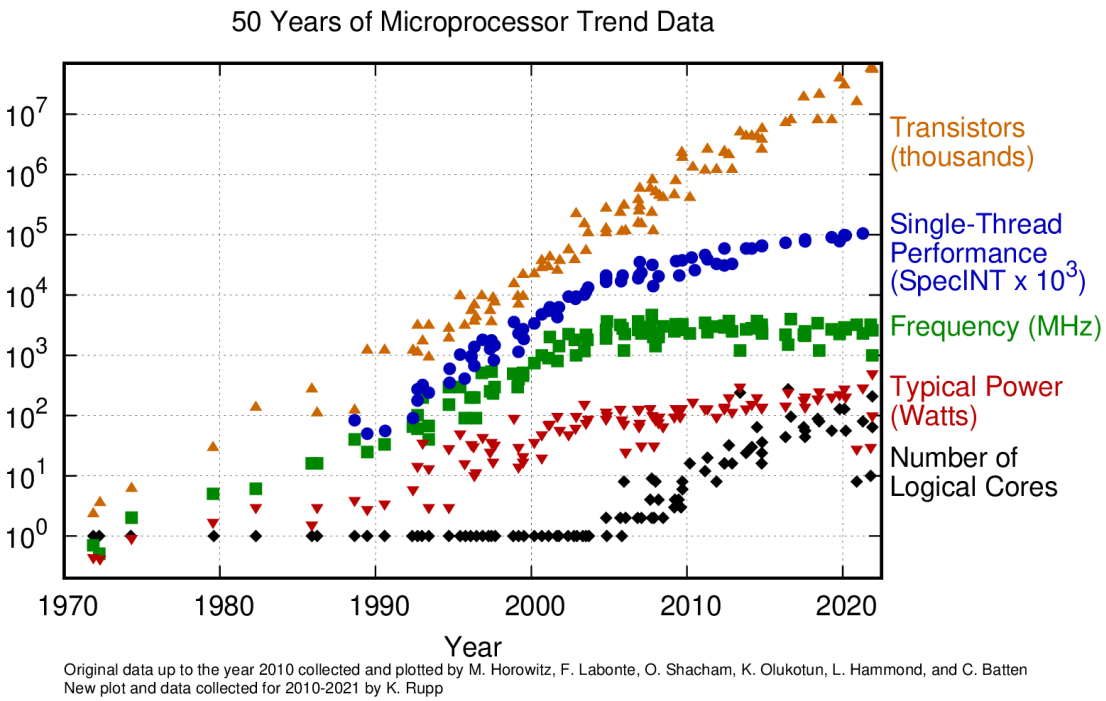


Figure 5.3: 50 years of Microprocessor Trend Data.

5.1.2 Memory/disk argument

The clock rates of high-end processors have increased to approximately 40% per year over the past decade, but the access times of DRAM have only improved to approximately 10% per year over this interval. The growing mismatch between the speed of the processor and memory and the effective net bandwidth between the DRAM and the processor causes significant performance bottlenecks. Parallel platforms provide the memory system with higher aggregate caches and higher aggregate bandwidth. The principles of parallel algorithms, such as data reference locality and bulk access, also apply to memory optimization. Some of the fastest-growing parallel computing applications utilize their ability to pump data to memory and disk faster than their raw computational speed.

5.1.3 Data communication argument

As the network evolves, the vision of the Internet as one large parallel/distributed computing environment has emerged. Many applications naturally use this computing paradigm. For example, SETI@Home (Search for Extra-Terrestrial Intelligence) was a project (1999–2020) of the Berkeley SETI Research Center to analyze radio signals, searching for signs of extraterrestrial intelligence[2]. Another project, Folding@Home[3] simulates how proteins from SARS-CoV-2 (the virus that causes COVID-19) work and how they interact with human cells. The project uses the idle processing resources of thousands of personal computers

[2] https://setiathome.berkeley.edu
[3] https://foldingathome.org

owned by volunteers who have installed the software on their systems. In many other applications, typically databases and data mining, there are constraints on the data location and resources across the Internet. Therefore, parallel interface techniques have to be used to perform analyses of this data.

5.2 Areas of parallel computing

Parallel computing greatly impacts many areas, such as computational simulations for scientific and engineering applications, commercial applications in data mining and transaction processing, etc.

- Design of airfoils (optimizing lift, drag, stability), internal combustion engines (optimizing charge distribution, burn), high-speed circuits (layouts for delays and capacitive and inductive effects) and structures (optimizing structural integrity, design parameters, cost, etc.).
- Design and simulation of micro- and nano-scale systems.
- Process optimization, operations research.

Scientific applications

- Functional and structural characterization of genes and proteins to understand and influence biological processes.
- Advances in computational physics and chemistry have explored new materials, an understanding of chemical pathways, and more efficient processes.
- Applications in astrophysics have explored the evolution of galaxies, thermonuclear processes, and the analysis of extremely large datasets from telescopes.
- Weather modeling, mineral prospecting, flood prediction, etc.
- Bioinformatics and astrophysics present some of the most challenging problems with respect to the analysis of huge datasets.

Commercial applications

- Emphasis on cost-effective servers capable of providing scalable performance.
- Some of the largest supercomputing networks power Wall Street.
- Data mining and analysis to optimize business and marketing decisions.

Applications in computer systems

- Network intrusion detection, cryptography, and multiparty computation require effective parallel and distributed algorithms.
- Embedded systems increasingly rely on distributed control algorithms.
- A modern automobile consists of tens of processors communicating to perform complex tasks to optimize handling and performance.

5.3 Types of parallel methods

According to Gear's classification [63], the parallel methods for IVPs can be divided into two main classes:

1. parallelism across the space (parallelism across the system),
2. parallelism across the time (parallelism across the method).

Parallelism across the space is effective only for systems with regular structure. Such problems arise from a direct discretization of a PDE via MOL. The original system is divided into several subsystems, processed concurrently by separated computing nodes with inter-process communications at the end of each integration step [154]. Parallel across space is suitable only for large-scale parallelism [63].

Parallelism across time is appropriate for small-scale parallelism and multi-core processors or computers with a few processors with fast inter-processor communication using shared memory. These methods are also known as *parallel-in-time methods*. They allow to search solutions for multiple steps simultaneously and can be classified into four groups:

1. methods based on multiple shooting,
2. methods based on domain decomposition and waveform relaxation,
3. space-time multigrid methods,
4. direct time parallel method.

For more information on parallel-in-time methods, see [48, 62].

For linear PDEs, the choice of method depends on the type of system of ODEs. Many solvers are designed for a particular type of system. The selection of *general-purpose* distributed-memory iterative-solution implementations is limited [91]. The known general-purpose packages are PETSc [10], hypre[4][5] (The Parallel High-Performance Preconditioners) [53, 54], and pARMS[6] (parallel Algebraic Recursive Multilevel Solvers) [108]. All mentioned libraries are based on the domain decomposition method [42, 152, 160] and MPI communication. The domain decomposition method is based on partitioning computational work among multiple processors by distributing the computational domain of a problem. Domain decomposition solves PDEs by iteratively solving subproblems corresponding to smaller subdomains [129]. The hypre and pARMS libraries both offer solvers for solving large, sparse linear systems of algebraic equations on massively parallel computers. Two main types of iterative methods for solving large linear systems of algebraic equations are Krylov subspace, and multigrid methods [108, 179]. The hypre library offers parallel multigrid solvers for structured and unstructured grid problems. On the other hand, the pARMS library offers parallel Krylov subspace solvers.

The PETSc library is the only one of the three mentioned libraries that offers solvers for large sparse systems of ODEs. PETSc library provides interfaces to various external packages. Therefore, it is possible to use solvers available in both pARMS and hypre libraries [10]. The list of external software supported by PETSc can be found in [12].

Since the work aimed to compare the MTSM method with commonly available methods for solving large systems of ODEs, the PETSc library was chosen for the numerical experiments.

[4] https://computing.llnl.gov/projects/hypre-scalable-linear-solvers-multigrid-methods
[5] https://hypre.readthedocs.io/en/latest
[6] https://www-users.cse.umn.edu/~saad/software/pARMS

5.4 Types of parallel architectures

There are two famous classifications of parallel computer architectures: Flynn's and Johnson's. Flynn's classification (see Figure 5.4) was developed in 1966 [59] and extended in 1972 [58] and distinguishes multiprocessor computer architectures according to how they can be classified along the two independent dimensions of Instruction Stream and Data Stream. A stream is a sequence of objects or actions. Each of these dimensions can have only one of two possible states: Single or Multiple. This results in four classes [129]:

1. SISD (single instruction, single data),
2. SIMD (single instruction, multiple data),
3. MISD (multiple instruction, single data),
4. MIMD (multiple instruction, multiple data).

These classes are described in Subsection 5.4.1.

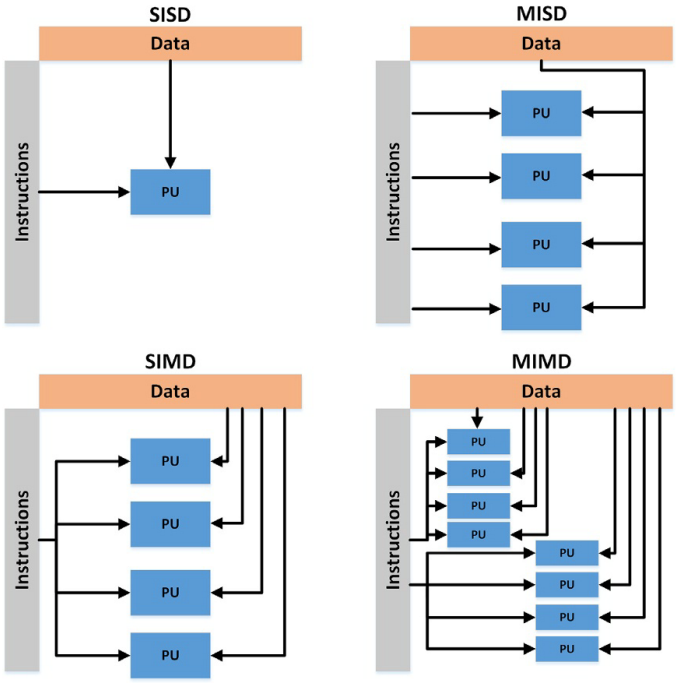


Figure 5.4: Flynn's classification of computer architectures [131].

In 1988, E. E. Johnson published an article *Completing an MIMD multiprocessor taxonomy* [88]. Johnson's classification is based on the memory structure (global or distributed) and the communication/synchronization mechanism (shared variable or message passing). This approach is much more practical than Flynn's classification because almost everything except the MIMD class of Flynn is not currently used. The resulting MIMD taxonomy contains four classes:

1. GMSV (Global Memory, Shared Variables) – the shared memory machines,
2. DMSV (Distributed Memory, Shared Variables) – the hybrid machines,

3. DMMP (Distributed Memory, Message Passing) – the message passing machines,
4. GMMP (Global Memory, Message Passing) machines.

This taxonomy omits the GMMP architectures, since they are not commonly used. As a result, computer architectures are divided into the following three classes:

1. UMA (Uniform Memory Access),
2. NUMA (Non-Uniform Memory Access),
3. NORMA (No Remote Memory Access).

These classes are described in subsection 5.4.2.

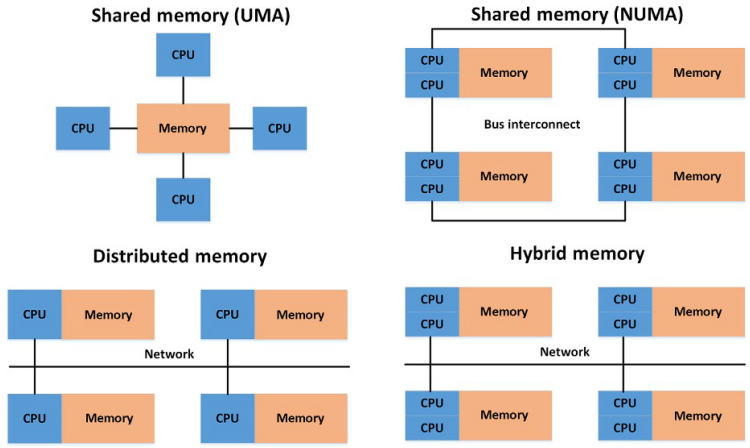


Figure 5.5: Johnson's classification of computer architectures [131].

5.4.1 Flynn's classification of parallel architectures

Flynn's classification defines four categories based on instruction and data: SISD, SIMD, MISD, and MIMD.
Single Instruction Single Data (SISD) – it is a traditional uniprocessor.

- Single Instruction: The CPU acted on only one instruction stream during any one clock cycle.
- Single Data: Only one data stream is being used as input during any one clock cycle.

The SISD class includes pipelined, superscalar, and VLIW (Very Long Instruction Word) processors. Pipelining is a process of arrangement of hardware elements of the CPU such that its overall performance is increased. Pipelined processors execute one or more instructions simultaneously.

The instruction stream consists of a sequence of instruction words. The instruction word represents the smallest executable packet visible to the programmer and executed by the processor. An SISD processor executes one or more operations per clock cycle from the instruction stream.

Scalar processors execute one single instruction per clock cycle. In contrast to a scalar processor, a superscalar processor can execute more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to different execution units on

the processor. Both scalar and superscalar processors execute one or more instructions per cycle, where each instruction contains a single operation. On the other hand, VLIW processors execute a single instruction word per cycle, where this instruction word contains multiple operations.

Single Instruction Multiple Data (SIMD) – it is a type of parallel computer.

- Single Instruction: All processing units execute the same instruction at any clock cycle.
- Multiple Data: Each processing unit can operate on a different data element.

The SIMD class of processors includes array processors and vector processors. They are suited for specialized problems characterized by high regularity, such as graphics or image processing. An array processor consists of interconnected processor elements; each element has its local memory. On the contrary, a vector processor consists of a single processor that references a single global memory space and has special functional units that operate specifically on vectors.

Multiple Instruction Single Data (MISD) – it is a type of parallel computer.

- Multiple Instruction: Each processing unit operates on the data independently via separate instruction streams.
- Single Data: A single data stream is fed into multiple processing units.

Data are streamed through the pipeline and forwards results from one function unit (stage) to the next. The MISD are not commonly used, but some MISD processors are commonly available, for example, GPGPUs (general purpose GPU) with an operational set at the individual stage in the pipeline. Another example is dataflow machines. The source program is converted to a data flow graph, where each node represents an operation. Data is streamed across the graph. Each path through the graph is an MISD implementation. Dataflow machines can be realized in FPGA implementations.

Multiple Instruction Multiple Data (MIMD) – it is a type of parallel computer.

- Multiple Instruction: Every processor may execute a different instruction stream.
- Multiple Data: Every processor may work with a different data stream.

In the MIMD class of processors, multiple processors of any type are interconnected. Most often, MIMD configurations are homogeneous. Heterogeneous MIMD configurations are usually used for special-purpose applications. The MIMD class includes multi-threaded and multi-core processors.

The multi-threaded processors consist of a single processor extended of multiple sets of program and data registers. Individual threads use each register set, and they are independent. If resources are available, then the threads continue execution.

The multi-core processors communicate results through the interconnection network and coordinate task control. The multi-core multiprocessor systems can be classified into two classes based on the nature of the memory address space.

- Systems with distributed memory: Each processor element has its own address space, and communication between processor elements is done through message passing.
- Systems with shared memory: The address space is shared, and communication is via the memory system.

There are two main problems when using shared-memory systems. The first problem is maintaining memory consistency. It is necessary to deal with the ordering effects on memory references within a processor element and also between different processor elements. This is solved using a combination of software and hardware techniques. The second problem is cache coherency, which means that it has to be ensured that all processor elements have the same value for a given memory location. This issue is solved using hardware techniques. Implementing a shared memory system is more difficult than implementing a distributed one because of memory consistency and cache coherency. On the other hand, distributed systems can be more difficult to program than shared memory systems. Recently, large-scale multiprocessors with distributed memory have been used.

5.4.2 Johnson's classification of parallel architectures

There are three main categories of parallel computer memory architectures, depending on the coupling of the processors and memory (see Figure 5.5): **Shared memory**: Multiple processors share a common memory and are fully connected to buses or switches. The CPUs have equal access to the physical memory and communicate using the shared memory model. If one processor changes a memory location, then this change is visible to all other processors. Shared memory parallel computers can be further classified into:

- Uniform Memory Access (UMA) – the access time to shared memory is the same for all processors. Nowadays, the most common form of UMA architecture is the Symmetric Multiprocessor (SMP) machine. The SMP includes two or more identical processors sharing a single main memory consisting of multiple identical processors with equal access and access time to the shared memory.

- Non-uniform Memory Access (NUMA) – many SMPs are linked, and one SMP can directly access the memory of another SMP. The access time to the memories of SMPs is not equal. In addition, the access to memory across the link is slower.

- No Remote Memory Access (NORMA) – a processor cannot access another processor's memory. Therefore, inter-processor communication is done by explicitly exchanging messages between processors. Each processor has its local memory.

Distributed memory: All processors have their local memory and can access it independently. All computers are connected through a network and can request data from other computers. The processors communicate with each other using message passing. The speed of data transmission is affected by the type of network used to connect computers.
Hybrid memory: A hybrid type of memory architecture combines shared and distributed memory architectures. All processors on a machine can share memory and request data from other computers. Modern supercomputers use hybrid memory architecture.

5.5 Supercomputers

Supercomputers are large systems specifically designed to solve complex scientific and industrial challenges. They have been used for highly compute-intensive tasks such as quantum physics, weather forecasting, climate research, oil and gas exploration, molecular modeling, and physical simulations.

Current supercomputers consist of many compute nodes interconnected through a high-speed network. Each compute node features one or more multicore or multithreaded processor chips, several memory modules, one or two network adapters, and possibly some local storage disks [66]. Supercomputer processing speed is measured in PFLOPS (petaflops), the quadrillion floating point operations per second.

The TOP500 project[7] ranks and details the 500 most powerful general-purpose supercomputers in the world based on the LINPACK Benchmark[8] introduced by Jack Dongarra [155]. The project was started in 1993 and published an updated list of supercomputers twice a year.

Figure 5.6 shows the development of the 500 fastest supercomputers in the world. It can be observed that in the last seventeen years, the performance of the computation has increased by a factor of one million. The frequency of processors increased until the 2000s. Since then, it has stagnated at approximately 2 GHz due to high power consumption and overheating at higher frequencies. Thus, parallelism is the only way to increase performance. Other challenges in HPC are the high power consumption of the overall system on the scale of megawatts and fault tolerance. The more components included, the greater the probability of failure.

As of June 2022, the Frontier system was the world's first exascale machine with an HPL[9] score of 1.102 Exaflop/s. The Frontier system at the US Department of Energy's (DOE's) Oak Ridge National Laboratory (ORNL) Oak Ridge National Laboratory (ORNL) in the US [32, 104] has 8 730 112 total cores, and Gigabit Ethernet for data transfer [156]. The IT4I Karolina supercomputer is also included in the TOP500 List. The Karolina CPU partition is at the 202nd position, and the GPU partition is at the 79th position. The Green500 project[10] evaluates the most energy-efficient supercomputers in the world. Karolina has reached the 14th place in the Green500 list (GPU partition) and the 65th place (CPU partition).

[7] https://www.top500.org
[8] https://www.top500.org/project/linpack
[9] A parallel implementation of the Linpack benchmark (HPL): http://www.netlib.org/benchmark/hpl/
[10] https://www.top500.org/lists/green500

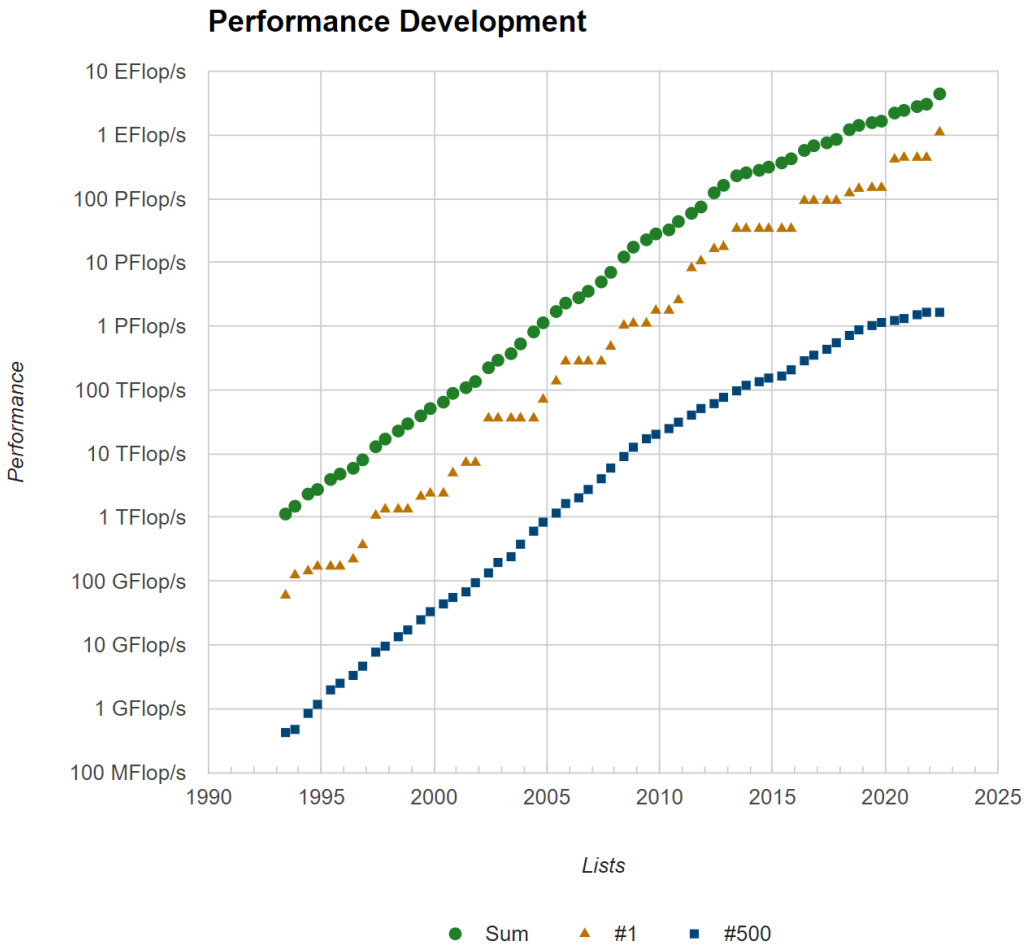


Figure 5.6: Performance development of the TOP500 supercomputers [156].

5.5.1 Interconnection networks and topologies

An important part of a high-performance computer system (HPC) is a high-performance interconnection network (ICN). The performance of ICN has a significant impact on the efficiency of parallel applications on HPCs. The performance of supercomputers increases, the number of computing cores increases, and the performance of computing nodes increases. The network scale of the HPC interconnection network also expands. For this reason, the overall performance of HPCs will be limited by the scalability of large-scale interconnection networks. As Moore's Law is slowing down, adopting new packaging design technologies to implement ICNs for high-performance computing is necessary. For more information, refer to articles [114, 123, 162], and books [37, 45].

5.5.2 Classification of high-performance interconnection networks

ICNs can be classified in different ways and according to different parameters. The main *performance properties* are bandwidth, latency, switch radix, and network topology. *Bandwidth* is the amount of data that can be transferred in a certain amount of time required for a packet to travel from a source node to a destination node. *Latency* measures the time

it takes for a packet to travel from a source node to a destination node. *Switch radix* is the number of switch ports through which it connects to other nodes.

ICNs can be classified into *static (direct)* and *dynamic (indirect)* networks. The network is *static* when a node is directly connected to its neighbors. For example, a fully interconnected network has direct links between any two nodes. Since the complexity of fully connected direct networks is quadratic ($O(N^2)$, where N is the number of nodes), it is not suitable for building large systems. Consequently, a node is directly connected to only a subset of other nodes. Communication with the remaining nodes is realized using routing messages through intermediate nodes (e.g., mesh, hypercube). To minimize the network diameter, high-radix routers are employed; from 8 ports, it has increased to 16, 24, 36, 48 ports, and even 64 ports. Note that the *diameter* of a network is the maximum length of any shortest path between an input and an output.

A *dynamic* network connects the nodes through one or more levels of switches. The crossbar enables a connection from any input to any output port if the output port is not already in use. A fully-connected network is ideal for keeping low-latency transfers, but unrealistic because of its complexity. Instead, smaller crossbars are used inside routers and switches. Switches are usually organized into stages using a regular connection pattern between stages (multistage ICNs).

The choice of ICN topology is influenced by the performance of the node and the interconnection technology [114]. The most frequently-used ICN topologies in the TOP500 list are direct k-ary n-cubes [36], fat tree, torus, mesh, and dragonfly.

Static topologies

The *k-ary n-cube* is an n-dimensional grid structure with k nodes accommodated in each dimension, where k denotes the radix, and n denotes the dimension. Each node can be identified by an n-digit radix k address [120]. Various types of k-ary n-cubes are depicted in Figure 5.7. Circles denote the communication nodes, and curves or lines denote the communication links.

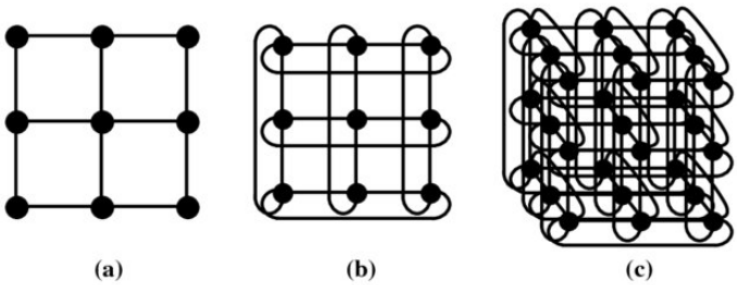


Figure 5.7: The various k-ary n-cube (a) a simple 3x3 regular mesh (b) a 3-ary 2-cube (2-D torus), and (c) a 3-ary 3-cube (3-D torus) [92].

Torus and meshes are n–dimensional grids with k nodes in each dimension and a total number of nodes $N = k^n$. If the border links in the grid are wrapped around each other, they form a torus [162]. Torus topologies directly interconnect a host with several neighbors in a k-dimensional lattice. The 3D torus topology is popular. One reason is that it has excellent packaging properties with uniformly short links for both the logical and the physical distances. The second reason is that many scientific problems are three-dimensional, and a large portion of the communication is between adjacent nodes. Finally, the node degree is

independent of the number of nodes. [162] The disadvantage of torus topologies is the low network throughput for adversary traffic patterns.

Hypercubes are another grid-based topologies. The number of nodes is fixed ($k = 2$), and the number of dimensions varies (see Figure 5.8). A 1D hypercube is a pair of two connected nodes, a 2D hypercube is a square with four nodes, a 3D hypercube is a cube, etc. A great advantage of the hypercube over the 3D torus was its lower hop count, consequently lower latency, and very good bandwidth scaling. [36, 162].

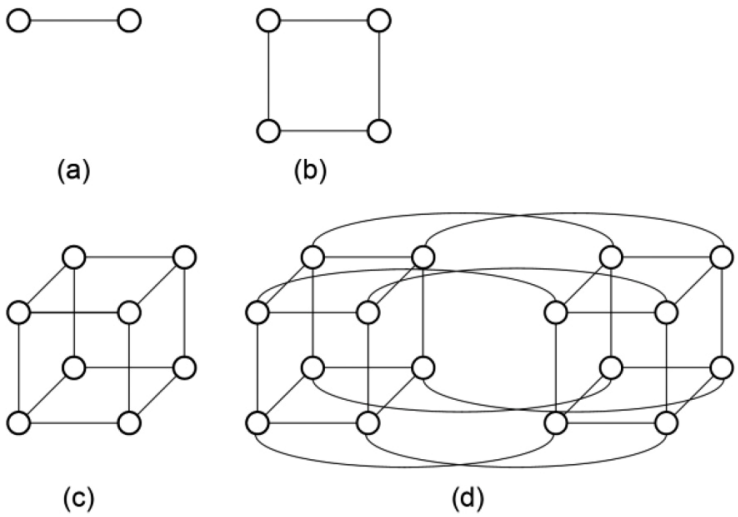


Figure 5.8: Hypercubes: (a) 2-ary 1-cube, (b) 2-ary 2-cube, (c) 2-ary 3-cube, and (d) 2-ary 4-cube [162].

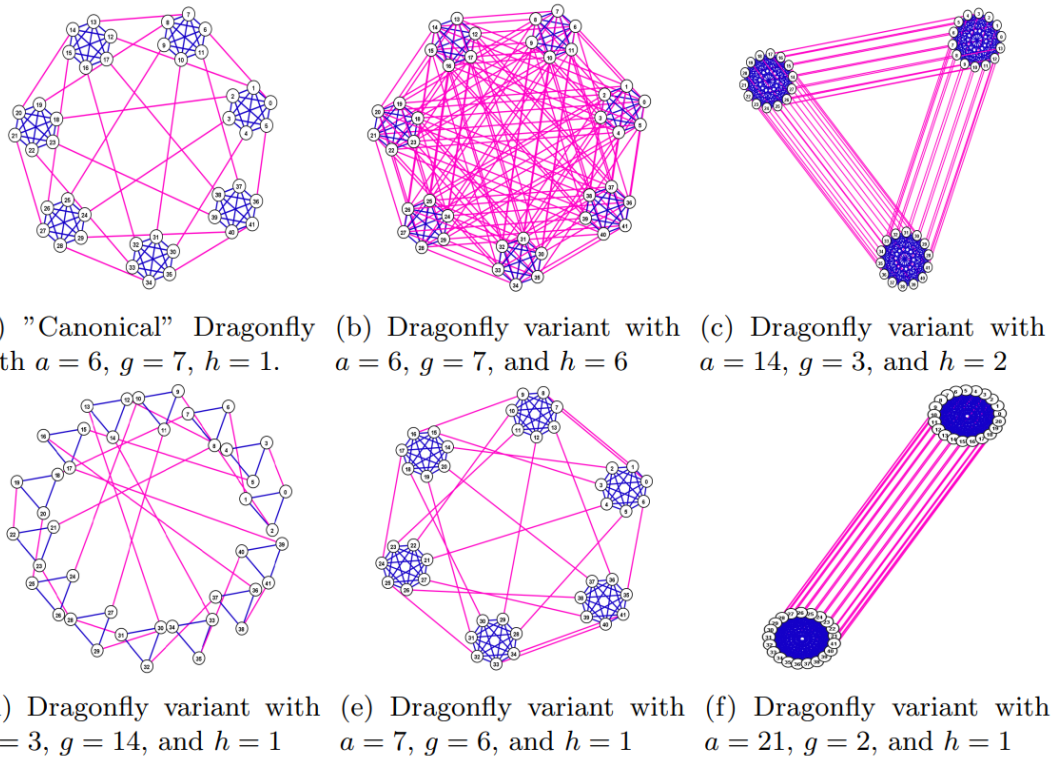
Dynamic topologies

Fat-Tree is one of the most widely used topologies. It provides low latency and maximizes data throughput for a variety of traffic patterns. In large-scale networks, this topology is expensive because it requires a large number of switches. The Fat-Tree topology consists of a tree of switches with the processing nodes as leaves. All components of the network are connected with links of the same speed. Communication progresses from the leaves to the root, and links closer to the root represent a bottleneck. To overcome this problem, faster links closer to the root are required, which also introduces a disadvantage. Another solution can be using multiple links between two levels of switches. The resulting tree has multiple roots, so faster root switches are not required. The level of the tree depends on the designer and a number of parameters. In practice, the Fat-Tree level is usually from two to four. [134, 162]

The *Dragonfly* topology was introduced by Kim John et al. [93]. Dragonfly provides good performance for various applications compared to the other topologies and reduces network costs by reducing the number of long links. [149] Every router contains a set of terminal connections leading to the end points and a set of topological connections leading to other routers. A collection of routers that belong to the same group is connected with intra-group connections, while router pairs belonging to different groups are connected with inter-group connections. In practice, routers and associated end-points that belong to a group are placed in a limited number of chassis or cabinets. This allows to implement

intra-group and terminal connections with short-distance, lower-cost electrical transmission links. [157]

The Dragonfly is comprised of g groups with a routers in each group, therefore a total number of routers is $S = ag$. Each router contains p terminal connections to endpoints. Considering Dragonflies with fully-meshed intra-group connectivity, each router has $a - 1$ intra-group connections to the other $a - 1$ routers in the group. Finally, each router has h inter-group connections to routers located in other groups [93].



(a) "Canonical" Dragonfly with $a = 6$, $g = 7$, $h = 1$.
(b) Dragonfly variant with $a = 6$, $g = 7$, and $h = 6$
(c) Dragonfly variant with $a = 14$, $g = 3$, and $h = 2$
(d) Dragonfly variant with $a = 3$, $g = 14$, and $h = 1$
(e) Dragonfly variant with $a = 7$, $g = 6$, and $h = 1$
(f) Dragonfly variant with $a = 21$, $g = 2$, and $h = 1$

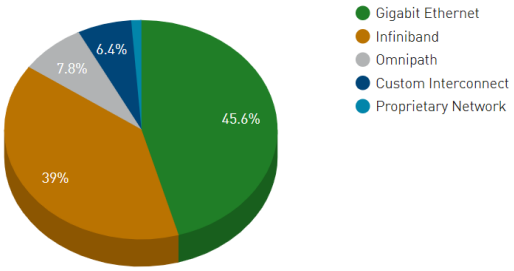
Figure 5.9: Examples 2-level Dragonfly variants with different parameters a, g, and h. The required number of routers $S = 42$. Purple links denote inter-group optical links, and blue links denote intra-group electrical links [157].

Typical ICNs in the TOP500 list

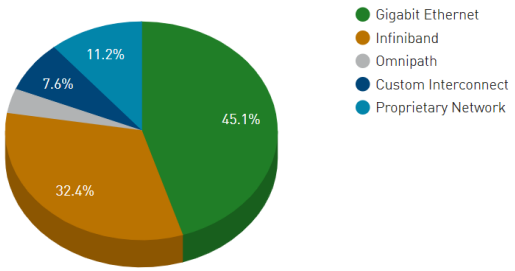
Typical ICNs in the Top500 are the InfiniBand interconnection [86], the Slingshot/Aries interconnection [38], Sunway [43], Tofu [3, 4], TH Express [110, 130, 170], BXI [40], and other custom or proprietary interconnections.

Figure 5.10a shows the interconnection distribution of the TOP500 systems from June 2022. It can be seen that Ethernet Interconnects take the largest share (45.6%), Infiniband (39%), Omnipath (7.8%), custom interconnect (6.4%) and other proprietary networks (1.2%). Figure 5.10b shows that Gigabit Ethernet contributes 45.1% performance share, Infiniband (32.4%), proprietary network (11.2%), custom interconnect (7.6%) and Omnipath (3.7%).

Ethernet Interconnects takes the largest share (50.8%) of the TOP500 systems, while it only contributes 19.6% performance share (i.e., Rmax) compared to IB interconnects that contribute 40% performance share with 31% system share.



(a) Interconnect Family (System Share)

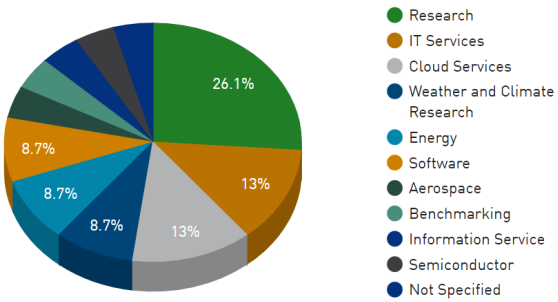


(b) Interconnect Family (Performance Share)

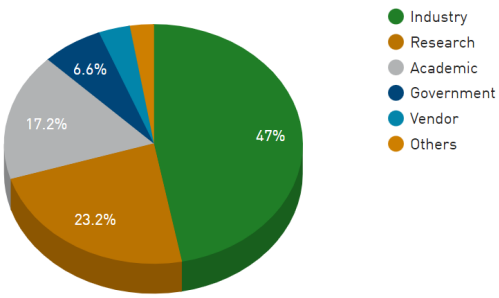
Figure 5.10: TOP500 list statistics: Interconnect family [156].

Figure 5.11a shows that the most widespread application area is Research with 26.1%. The second position in the ranking is shared by IT services and Cloud Services, both with 13 %. Next, Software, Energy, and Weather and Climate Research share the third position with 8.7%. The rest of the pie chart is divided into Aerospace, Benchmarking, Information Service, and Semiconductors (4.3%).

Figure 5.11b depicts the market segments. Almost half of the market segment is Industry, with 47%. Research, with 23.2% is in the second position, Academic segment in the third position 17.2%. The rest are Government 6.6%, Vendor 3.4%, and Others 2.6%.



(a) Application area (system share)

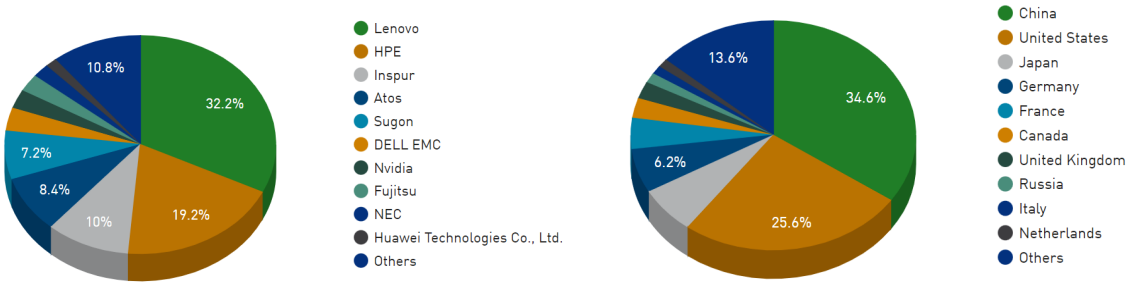


(b) Market segments (system share)

Figure 5.11: TOP500 list statistics: Application areas and market segments [156].

Figure 5.12a illustrates the ICN vendors, including Lenovo (China) 32.2%, Hewlett Packard Enterprise (HPE) in USA 19.2%, Inspur (China) 10%, Atos (France) 8.4%, Sugon (China) 7.2%, DELL EMC (USA) 3.4%, Nvidia (USA) 2.8%, Fujitsu (Japan) 2.6%, NEC (Japan) 2%, Huawei Technologies Co., Ldt. (China) 1.4%, and others.

The countries hosting the TOP500 supercomputers are shown in Figure 5.12b. China occupies the top spot on the list with 34.6%, followed by the United States (25.6%), Japan (6.6%), Germany (6.2%), France (3.8%), Canada (2.8%), the United Kingdom (2.4%), and others.



(a) Vendors (system share)

(b) Countries (system share)

Figure 5.12: TOP500 list statistics: Vendors and countries [156].

Figure 5.13 shows the evolution of the system-level interconnection technology used by supercomputers in the last year. It can be seen that Ethernet is still widely used. In the list from June 2022, the 228 (45.6%) supercomputers employ an Ethernet interconnection network. Many academic and industrial HPC systems cannot afford InfiniBand or are unwilling to give up Ethernet [114]. Compared to the list of June 2021, 247 (49.4%) supercomputers employed Ethernet. That means that Ethernet decreases by 7.7%. The portion of Infiniband increased; in the list from June 2022, the number of supercomputers interconnected with InfiniBand is 195 (39.0%), while in the list from June 2021, it is 168 (33.6%). Therefore, InfiniBand grew 16.07% year-over-year.

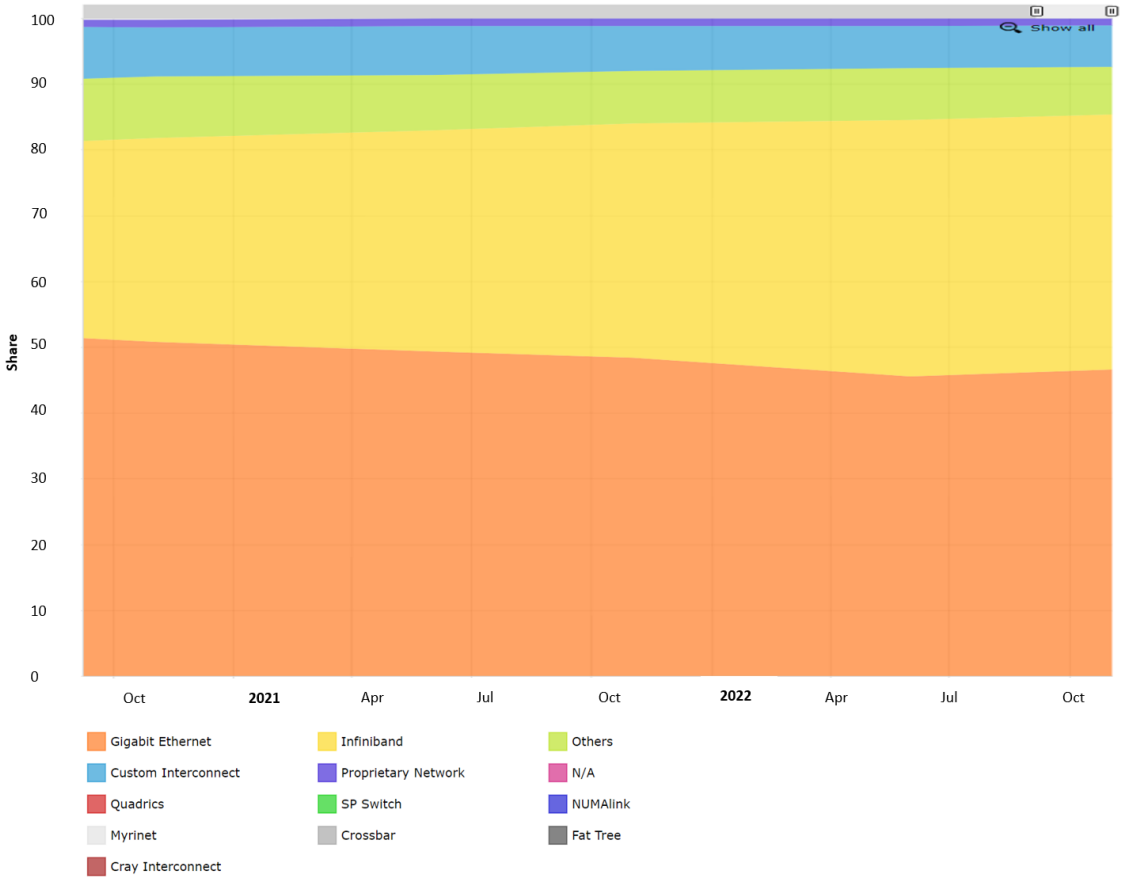


Figure 5.13: Distribution map of ICNs in the TOP500 list over last year [156].

Table 5.1 summarizes the selected properties on the Top10 list in June 2022, including manufacturer, country, market segment, Rmax (Maximal LINPACK performance achieved), interconnect family, and interconnect.

Rank	**System**	**Manufacturer**	**Country**	**Segment**	**Rmax** [TFLOP/s]	**Interconnect Family**	**Interconnect**	**Topology**	**Power** (MW)
1	Frontier	HPE	United States	Research	1 102 000	Gigabit Ethernet	Slingshot-11	Dragonfly	21.10
2	Fugaku	Fujitsu	Japan	Research	442 010	Proprietary Network	Tofu D	6D-Torus	29.90
3	LUMI	HPE	Finland	Research	151 900	Gigabit Ethernet	Slingshot-11	Star	2.94
4	Summit	IBM	United States	Research	148 600	Infiniband	EDR Infiniband	Fat-tree	10.10
5	Sierra	IBM/ NVIDIA/ Mellanox	United States	Research	94 640	Infiniband	EDR Infiniband	Fat-tree	7.44
6	Sunway TaihuLight	NRCPC	China	Research	93 014	Custom Interconnect	Sunway	Fat-tree	15.37
7	Perlmutter	HPE	United States	Research	70 870	Gigabit Ethernet	Slingshot-10	Dragonfly	2.59
8	Selene	Nvidia	United States	Vendor	63 460	Infiniband	HDR Infiniband	Fat-tree	2.65
9	Tianhe-2A	NUDT	China	Research	61 444.5	Custom Interconnect	TH Express-2	Fat-tree	18.48
10	Adastra	HPE	France	Academic	46 100	Gigabit Ethernet	Slingshot-11	–	0.92
79	Karolina GPU partition	HPE	Czechia	Academic	6 752	Infiniband	Infiniband HDR200	Fat-tree	0.31

Table 5.1: Top 10 supercomputers of the TOP500 list [156].

5.5.3 Challenges of current high-performance ICNs

The performance improvement of high-performance computers mainly depends on increasing the number of computing nodes and improving single-node computing performance. With increasing computing power of a single node, the communication bandwidth should also increase to maximize the computing performance of the node. The requirements for network bandwidth in the exascale HPC computer systems are high and represent a great technology challenge. Also, the diameter of the network is larger, resulting in increased node communication delay [111, 115]. The main challenges in the exascale era include power consumption, density, reliability, and cost, among other issues [114].

In 2008, the report [23] showed that the main four challenges of exascale supercomputers are power consumption, data movement, fault tolerance, and extreme parallelism. In 2021, a CTO for the Exascale Computing Project[11] and the Oak Ridge Leadership Computing Facility at ORNL, Al Geist, presented a seminar and explain how the Frontier[12] supercomputer overcomes the challenges mentioned above [64]. For more information, see [163].

Power consumption

Power consumption was the main challenge. The analysis presented in [23] shows that building a one exaflop system using current technologies results in a power consumption of more than 600 MW. The US Department of Energy's (DOE's) Oak Ridge National Laboratory (ORNL) developed a chip that reduces the power consumption to 20 MW per exaflop. This chip is installed in the Frontier supercomputer.

Speed and energy of data movement

The time and energy required to move a byte of data from memory to processors and from processors back to storage is orders of magnitude greater than the time and energy required to perform a floating-point operation on those data [163]. The rate of improvement in processor performance far exceeds the rate of improvement in DRAM memory speed. This situation is called the *memory wall problem*, first described by Wulf and McKee in 1995 [176]: „The memory wall is defined as a situation where the much faster improvement of processor speed as compared with dynamic random access memory (DRAM) speed will eventually result in processor speed improvements being masked by the relatively slow improvements to DRAM speed." Frontier reduces the memory problem by stacked, high-bandwidth memory soldered directly onto its GPUs, increasing the data moving by order of magnitude.

Fault tolerance

The computer fault rates increase because exascale systems are huge and complex. The US Department of Energy's (DOEs') invests in projects to develop failure-tolerant chips. Furthermore, the checkpoint times on Frontier on-node non-volatile memory have been reduced from minutes to seconds. Due to this solution, the checkpoint times are still much shorter than the mean time to failure.

[11] https://www.exascaleproject.org
[12] https://www.olcf.ornl.gov/frontier

Extreme parallelism

Exascale supercomputers must calculate = $1e^{18}$ (one quintillion) FLOPs. In other words, one exaflop requires 1 billion floating point units, each of which has to perform 1 billion calculations per second. Exascale applications may require the problem to be decomposed into billion parallel threads [163].

Frontier uses large multi-GPU nodes, each hiding between 1000–10 000-way concurrency inside their pipelines. Thanks to this solution, users do not have to think about parallelism so much. Instead, they have to choose the appropriate number of GPUs or the number of nodes on the computer. The system software of Frontier deals only with thousands of nodes and not a million. Frontier has 9472 computing nodes.

5.6 Parallel performance metrics and laws

There are many ways to measure the performance of a parallel algorithm running on a parallel processor. The most widely used measurements are the computation time, price/performance, speed-up, and efficiency [90, 144].

5.6.1 Execution time

The execution time of a serial program is the time elapsed between the beginning and the end of its execution on a sequential computer. The execution time of a parallel program is the time elapsed from the moment the first processor starts to the moment the last processor finishes execution.

5.6.2 Scalability

Scalability is defined as the measure of a parallel system's capacity to increase speedup in proportion to the number of processors. For hardware (clusters), this means that the capacity of the system can be proportionally increased by adding more hardware. For software, scalability can be defined as parallel efficiency, i.e., the ratio between the actual speedup and the ideal speedup obtained by a certain number of processors.

The main challenge of parallel computing is to decide how to divide a problem into individual parts that can be computed independently. To avoid high resource usage and high time consumption, large applications are not developed and tested using the full problem size and a number of processors. For this reason, it is desirable to decrease these factors and estimate the required resources for the full run more accurately using resource planning.

Scalability testing measures the performance of a system when the problem sizes and the number of processors vary. It tests the system's ability to meet growing needs. It does not test the general functionality of an application or its correctness. Systems can be divided into strong scaling and weak scaling systems. For more information, refer to [61, 73].

5.6.3 Strong scaling

In the case of strong scaling, the number of processors increases while the problem size remains constant. Strong scaling is defined as how the calculation time varies with the number of processors for a fixed total problem size, resulting in a reduced workload per processor. The workload per processor should be kept at a reasonable level to keep all processors occupied. The speedup typically decreases continuously. In the ideal case, the

problem scales linearly. That means the problem speeds up by a factor of N, where N denotes the number of parallel execution units.

Amdahl's law

At the AFIPS Spring Joint Computer Conference in 1967, the computer scientist Gene Amdahl pointed out that *the speedup of a program using multiple processors is limited by the time needed for the sequential fraction of the program* [6]. Amdahl's law can be formulated as follows:

$$S(p) = \frac{1}{f + \frac{(1-f)}{p}}, \tag{5.1}$$

where f is the serial fraction of the code, $(1 - f)$ is the parallel fraction of the code, and p denotes the number of processors. For a fixed problem size, the upper limit of speedup is determined by the serial fraction of the code:

$$\lim_{N \longrightarrow \infty} S(p) = \frac{1}{f}. \tag{5.2}$$

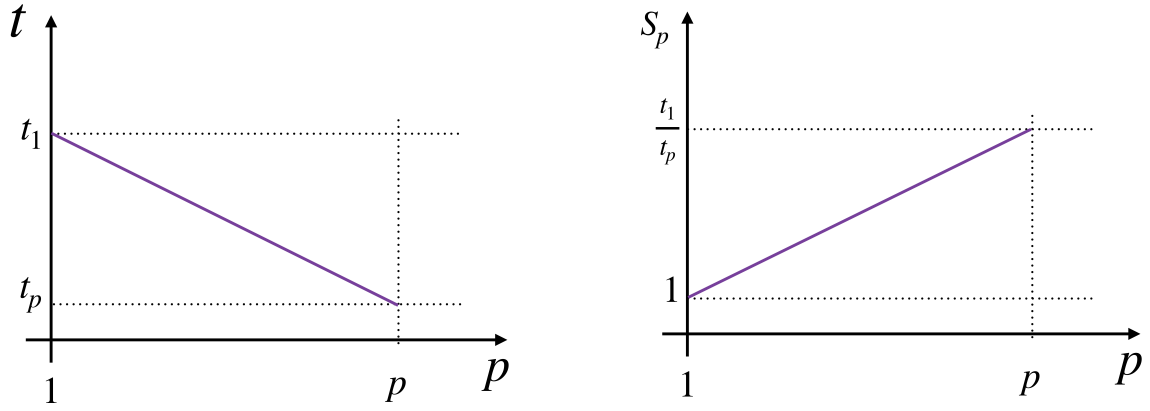
This analysis neglects other potential bottlenecks in computing systems, such as memory bandwidth and I/O bandwidth, and the performance cost associated with creating and managing threads.

Strong scaling speedup

Strong scaling speedup is defined as:

$$S = \frac{t_1}{t_p}, \tag{5.3}$$

where t_1 is the calculation time of a single processor and t_p is the calculation time for p processors. Strong scaling data representation is in Figures 5.14a and 5.14b.



(a) Strong scaling: execution time

(b) Strong scaling: strong speedup

Figure 5.14: Strong scaling: execution time and strong speedup [171].

Strong parallel efficiency

Parallel efficiency is defined as:

$$E = \frac{S}{p}\,, \tag{5.4}$$

where S is the strong scaling speedup and p is the number of processes.

5.6.4 Weak scaling

In weak scaling, both the number of processors and the problem size increase, resulting in a constant workload per processor. Amdahl's law defines the upper limit of speedup for fixed-size problems. This is a bottleneck for parallel computing. In practice, the sizes of problems scale with the number of available resources. If a problem requires a small amount of resources, it is not beneficial to use a large amount of resources for the computation. It is better to use a small amount of resources for small problems and a larger amount of resources for large problems.

Gustafson's law

In 1988 John L. Gustafson published the article *Reevaluation Amdahl's Law* [72] based on the approximations that the parallel part scales linearly with the amount of resources. The serial part does not increase with respect to the size of the problem.

Weak scaling speedup

Weak scaling speedup (scaled speedup) is defined as:

$$S(p) = f + p \cdot (1 - f)\,, \tag{5.5}$$

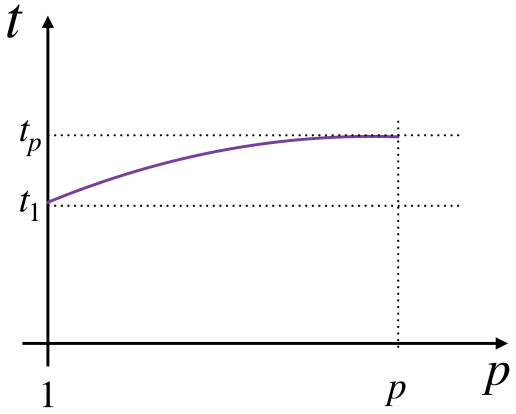
where p and f have the same meaning as in Amdahl's law. The scaled speedup increases linearly with respect to the number of processors with a slope smaller than one, and there is no upper limit for the scaled speedup. In contrast to Amdahl's law, where the problem size is fixed, the scaled speedup is calculated based on the amount of work done for a scaled problem size.

Weak parallel efficiency

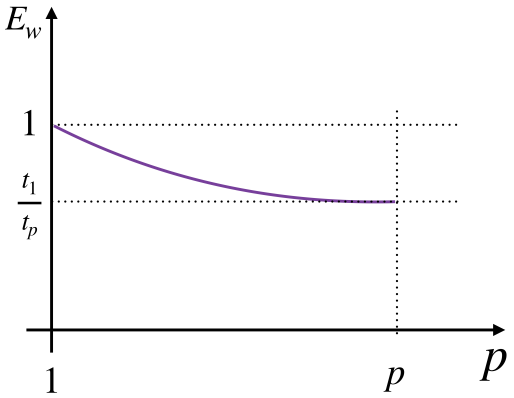
Weak scaling efficiency is commonly defined as speedup S_p (5.5) divided by the number of units of execution p

$$E_w = \frac{S(p)}{p} = \frac{t_1}{t_N}\,. \tag{5.6}$$

Weak scaling data representation is in Figures 5.15a and 5.15b.



(a) Weak scaling: execution time



(b) Weak scaling: weak efficiency

Figure 5.15: Strong scaling: execution time and strong speedup [171].

5.7 Berkeley Roofline model for multicore architectures

Multicore architectures can be complicated and very different. For example, some offer many simple processors, whereas others offer fewer complex processors, and some depend on multithreading. This diversity makes it difficult for programmers, compiler writers, and even architects [174]. Multicore architecture does not guarantee good scalability or performance, and it is crucial to understand the limits to both scalability and efficiency.

The roofline model is based on the simplified model of the CPU and hides most of the architecture-specific complexity. The roofline model was first proposed in 2009 by S. Williams [174], and it can be used not only on the most common architecture x86_64, but also on other architectures such as ARM, GPU accelerators, Intel Xeon Phi or FPGA. The roofline model analyzes bottlenecks during execution on a given hardware. The prerequisites for the roofline model are that the data transfer and the computation overlap perfectly, latency effects are ignored, and steady-state code execution (no wind-up/down effects). The hardware is viewed as two units. The execution unit which operates at the *peak performance* P_{peak} measured in [FLOPs/s], where $FLOPs$ is the number of floating-point operations, and the data unit, which can store or deliver data at a *maximum bandwidth* b_s measured in [B/s].

Let us denote the size of data read or written to memory as V and the number of floating point operations as F. The processor needs F/P_{peak}s to finish all calculations and it needs V/b_ss to perform memory transfer. The *arithmetic (computational) intensity (AI)* can be calculated as a ratio $I = F/V$ in [FLOPs/B], and it is the number of operations per one byte of memory transfer. Both numbers P_{peak} and $I \cdot b_s$ are upper limits of the expected computational performance, and it can be written as:

$$P = \min(P_{peak}, I \cdot b_s)\,. \tag{5.7}$$

From (5.7), we can see that the basic roofline model bounds performance as a function of machine peak performance, machine peak bandwidth, and arithmetic intensity [164].

The graphical representation of the roofline model is shown in Figure 5.16. The x-axis represents the arithmetic (operational) intensity [FLOPs/B], it is a linear function with a slope of b_s, and the y-axis shows the performance [FLOPs/s], it is a constant function. Both axes of the graph have a logarithmic scale. The intersection of the diagonal and

horizontal roof is called *ridge (knee) point*. The x-coordinate of the ridge point is the minimum arithmetic (operational) intensity required to achieve peak performance. The least arithmetic intensity needed by the application to reach the ridge point is called *machine balance* B_m and is defined as $B_m = b_s/P_{peak}$. The ridge point divides the area of the roofline model into two regions. If the arithmetic intensity of the application is smaller than the ridge point value, the application is referred to as *memory bound*. On the other hand, if the arithmetic of the application is higher than the ridge point, the application is referred to as *compute bound*. Figure 5.16 shows two kernels. The kernel with arithmetic intensity $AI = 1/2$ is memory bound, and the kernel with arithmetic intensity $AI = 2$ is compute bound.

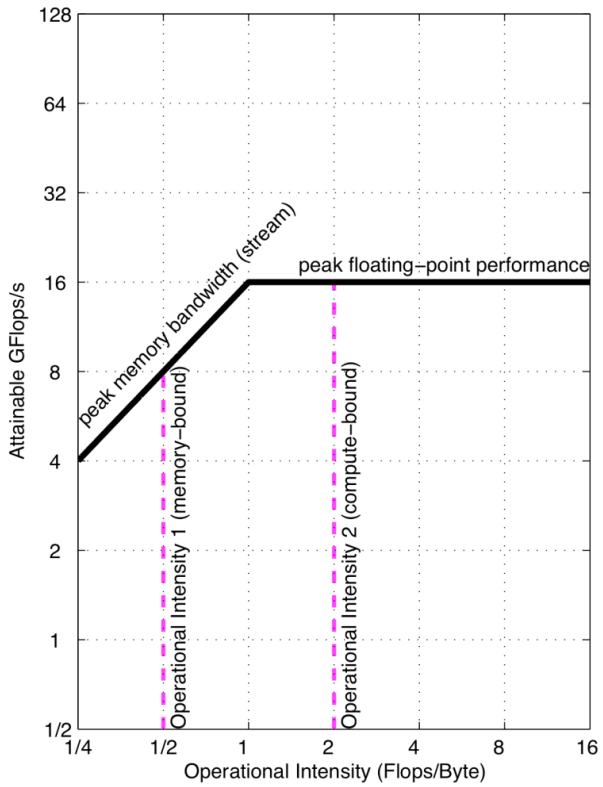


Figure 5.16: The roofline model for AMD Opteron X2 [174].

Since different architectures have different values for both the memory bandwidth and the peak performance, the application can be memory bound on one architecture and compute-bound on the other.

5.7.1 Roofline ceilings

The roofline ceilings are shown in Figure 5.17. The first subfigure 5.17 a) shows the *computational ceilings*. The ceiling labeled as *TLP only* reflects missing optimizations to increase ILP (Instruction Level Parallelism) or SIMD; the TLP stands for Thread Level Parallelism. The second ceiling is labeled *ILP or SIMD* and reflects when the floating-point operation mix is not balanced. The second subfigure 5.17 b) depicts the memory bandwidth ceilings without software prefetching, without affinity optimizations, and with only unit stride optimizations.

The last subfigure 5.17 c) combines two previously-mentioned subfigures into a single graph. The arithmetic intensity of the computational kernel determines the optimization region. Only computational optimizations are suggested if the kernel falls in the blue region (Kernel 2). The yellow region means that only memory optimizations are applicable. Finally, the green regions suggest both types of optimizations (Kernel 1). Two optimizations can be used to reduce computational bottlenecks:

1. Improve instruction level parallelism (ILP) and apply SIMD.
2. Balance floating-point operation mix.

Three optimizations can be used to reduce memory bottlenecks:

1. Restructure loops for unit stride access.
2. Ensure memory affinity.
3. Use software prefetching.

For more details on optimizations, see [173, 174, 177].

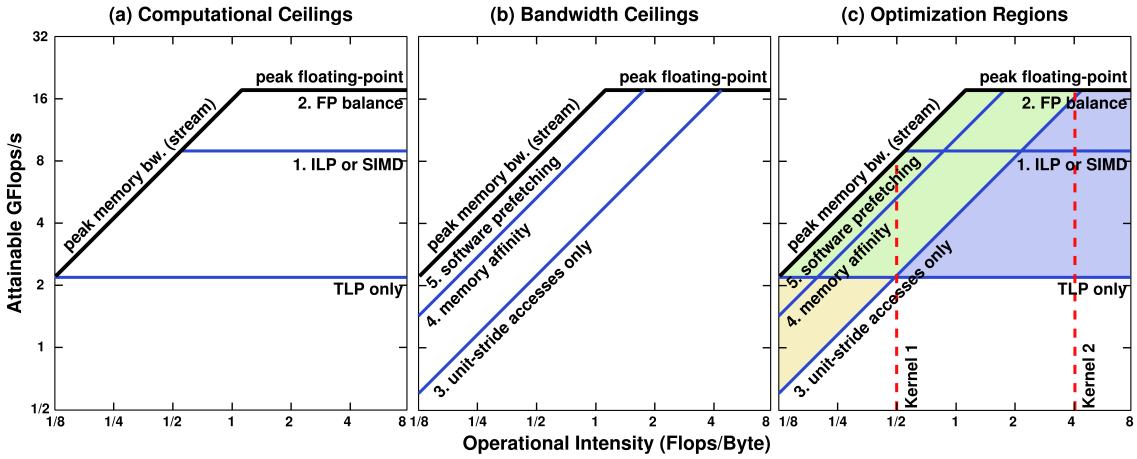


Figure 5.17: Roofline model example [174] with computational and bandwidth ceilings and its optimization regions.

Chapter 6

Results

This chapter focuses on the solution of the second-order PDEs, especially linear ones (elliptic, hyperbolic, and parabolic), which are very important in practical applications [50, 55]. The PDEs are solved using the MOL (see Section 3.4). The large systems of ODEs arising from the method of lines are solved in parallel using the higher-order MTSM and Runge-Kutta methods. To show the suitability of the MTSM to solve these kinds of problems and its advantages over the other commonly used methods, the set of experiments is performed, and the results are analyzed.

This chapter consists of several sections. Section 6.1 summarizes the technical specifications of the supercomputers used in numerical experiments. Tools for scientific computing are introduced in Section 6.2. Section 6.3 describes performance metrics to evaluate and characterize numerical results. Characteristics of the selected problems are summarized in Section 6.4. Sections 6.5 and 6.6 deal with the heat equation discretized in the spatial domain using a three-point and five-point difference formula, respectively. The wave equation discretized in the spatial domain by three-point and five-point difference formula is presented in Sections 6.7 and 6.8 respectively. Section 6.9 focuses on the numerical solution of the telegraph equation. Parallel performance analysis is presented in Section 6.10.

6.1 Technical specifications of supercomputers

Barbora supercomputer cluster[1], IT4Innovations National Supercomputing Center[2], Ostrava, Czech Republic, consists of 201 compute nodes, totaling 7232 compute cores with 44 544 GB RAM, giving over 848 TFLOP/s theoretical peak performance[3]. The compute nodes without accelerators have the following parameters: 192 nodes, 6912 cores in total, 2× Intel Cascade Lake 6240[4], 18-core, 2.60 GHz processors per node, 192 GB DDR4 2933 MT/s of physical memory per node (12× 16 GB), L1 Cache: 576 KiB, L2 Cache: 18 MiB, L3 Cache: 24.75 MiB, BullSequana X1120 blade servers, 2995.2 GFLOP/s per compute node, 1× 1 GB Ethernet, 1× HDR100 IB port, 3 computes nodes per X1120 blade server, PBS Professional scheduler, version 19.1.3[5], PETSc version 3.14.4, GCC version 10.2.0.

[1] https://docs.it4i.cz/barbora
[2] https://www.it4i.cz/en
[3] https://docs.it4i.cz/barbora/introduction
[4] https://docs.it4i.cz/barbora/compute-nodes
[5] https://www.altair.com/pbs-works-documentation

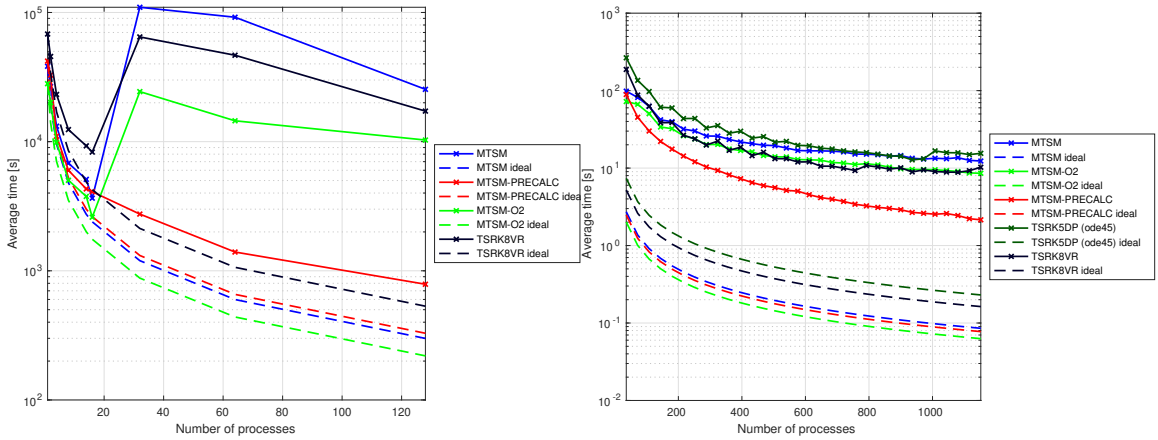
The first experiments were also performed on the ICS cluster[6], Institute of Computational Science[7], Università della Svizzera italiana, Lugano, Switzerland. The ICS cluster has the following parameters. CPU: 2×Intel Xeon E5-2650 v3 @ 2.3 GHz, 20 (2×10) cores, RAM: 64 GB DDR4 @ 2133 MHz, HDD: 1×1 TB SATA 6 Gb, Infiniband Adapter: Intel 40 Gbps QDR, MATLAB version R2020, PETSc version 3.13.5, GCC version 10.1.0, Slurm workload manager[8] version 20.11. Operating system: CentOS 8.2.2004.x86_64. The experiments were performed with different numbers of MPI processes. For 1–16 processes, one compute node was used; for 32 processes, two compute nodes; for 64 processes, four compute nodes; and for 128 processes, eight compute nodes. For more information, see [pp14] and [pp16]. Table 6.1 shows the parameters of the Barbora and ICS clusters.

Parameter	**Barbora**	**ICS**
Architecture	x86-64	x86-64
Operating system	Linux	Linux
Scheduler	PBS	Slurm
Total number of nodes	201	41
Number of nodes (without accelerators)	189	22
Processor cores	36	20
CPU	2× Intel Cascade Lake 6240, 18-core, 2.6 GHz processors per node	2× Intel Xeon E5-2650 v3 @ 2.3 GHz, 20 (2×10) cores
RAM	min. 6 GB	64 GB
HDD	192 GB DDR4 2933 MT/s of physical memory per node (12×16 GB)	$1 \times$ TB SATA 6 Gb
Compute network	InfiniBand HDR	InfiniBand 40 Gbps QDR

Table 6.1: Parameters of the Barbora and ICS clusters.

Figure 6.1 shows the average computation times for the telegraph equation (see Section 6.9) calculated on the ICS cluster (see Figure 6.1a) and on the Barbora cluster (see Figure 6.1b). As mentioned above, the experiments performed on the ICS cluster use 1–16 processes for one compute node, then 32 processes for two compute nodes, etc. The average computation time increases significantly between 16 (one node) and 32 processes (two nodes). This increase is probably caused by TCP communication among compute nodes. For this reason, the Barbora cluster was used for further experiments.

[6] https://intranet.ics.usi.ch/HPC
[7] https://www.ics.usi.ch
[8] https://slurm.schedmd.com



(a) Average time for $S = 256000$, ICS cluster (b) Average time for $S = 512000$, Barbora cluster

Figure 6.1: Average times for the telegraph equation problem.

6.2 Tools for scientific computing

MATLAB software was mainly used for benchmarking, data visualization, and initial experiments. The PETSc framework was used for parallel implementation.

6.2.1 MATLAB

MATLAB is the language of scientific computing. It contains a full suite of tools that can be used to solve various engineering and mathematical problems. The numerical experiments in this work use the MATLAB ODE suite.

The MATLAB ODE suite contains three explicit methods for nonstiff problems: ode23, ode45, and ode113 solvers. The ode23 solver is an implementation of an explicit Runge-Kutta (2,3) formula (Bogacki-Shampine), a third-order scheme with a second-order embedded scheme, four stages; it is a single-step solver (see (2.50)). The ode45 solver is based on an explicit Runge-Kutta (4,5) formula (Dormand-Prince), a fifth-order scheme with a fourth-order embedded scheme, seven stages; it is a single-step solver (see (2.49)). The ode113 solver is a variable-step variable-order (VSVO) Adams-Bashforth-Moulton PECE solver of orders 1 to 13. The highest order used appears to be 12. However, a formula of order 13 is used to form the error estimate, and the function performs local extrapolation to advance the integration at order 13 (see (2.55)).

The MATLAB ODE suite also contains implicit methods for stiff systems [8, 151]: ode15s and ode23s solvers. The ode23s solver is the second-order scheme with the third-order embedded scheme, and it is based on modified Rosenbrock methods of orders 3 and 2 with error control. It is a single-step solver. Solver ode15s is a multistep VSVO solver based on the numerical differentiation formulas (NDFs) of orders 1 to 5.

Vectorized MATLAB code of explicit MTSM with variable order and variable step size scheme for linear systems of ODEs has been implemented. The MTSM was compared with vectorized MATLAB explicit ode solvers. Numerical experiments are carefully described in [pp10, pp22, pp26, pp28].

6.2.2 PETSc

The Portable Extensible Toolkit for Scientific Computation (PETSc) is a software framework [2, 10, 11, 13] for the scalable numerical solution of complex problems in science and engineering. PETSc was designed primarily for typical computations connected with PDEs. PETSc is written in C but can also be employed, e.g., in C++, Fortran, and Python programs.

PETSc provides implementations of distributed sparse, dense, unassembled matrices, linear algebra, linear or non-linear system solvers, time integrators, mathematical optimization, discretization, and more. PETSc uses the message-passing model for parallel programming and employs MPI for all interprocessor communication. Figure 6.2 shows the components of the PETSc software package.

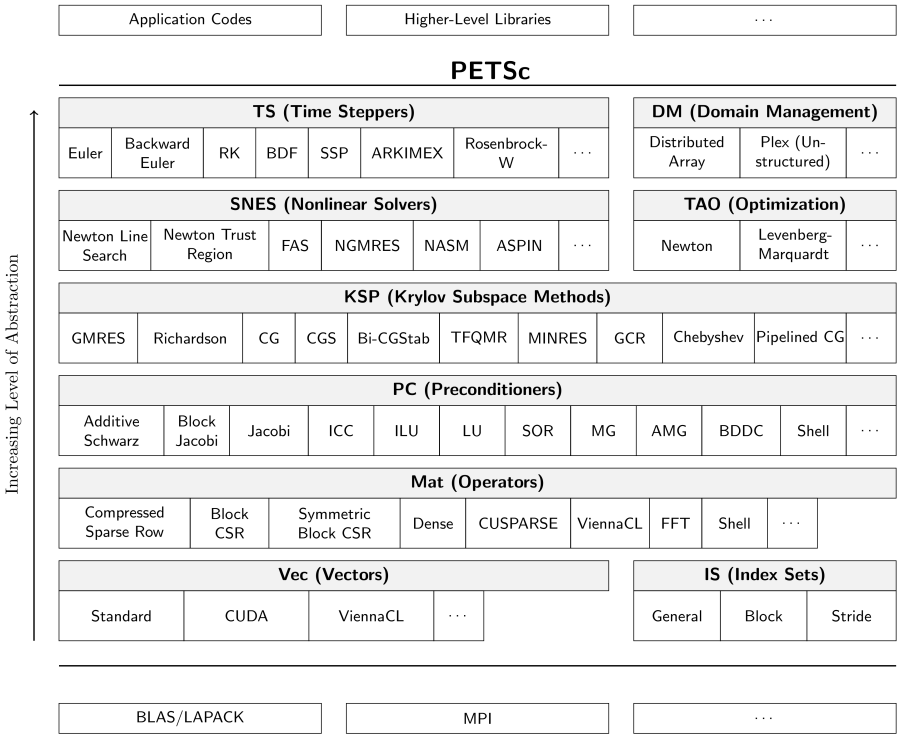


Figure 6.2: Components of the PETSc software package [10].

The parallel layout of the matrix is shown in Figure 6.3. Each process owns a certain number of matrix rows.

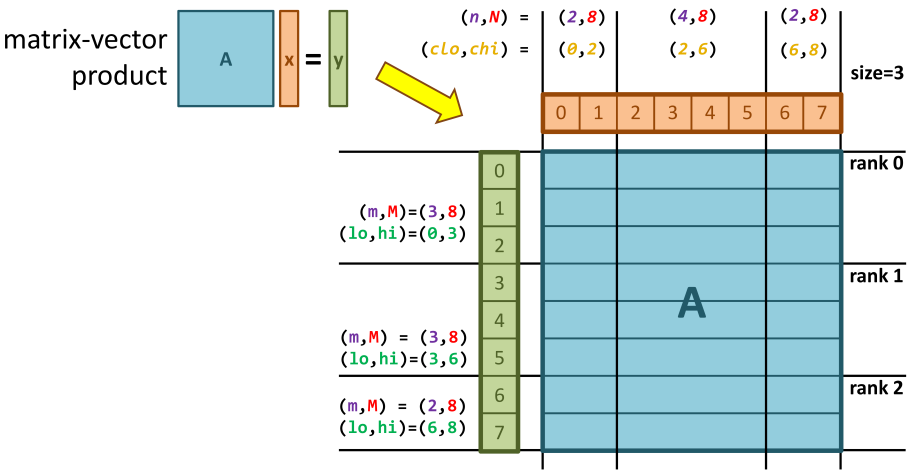


Figure 6.3: PETSc: parallel matrix layout [79].

The sparse parallel matrices are stored in compressed row storage (CRS), compressed sparse row (CSR), or Yale format as shown in Figure 6.4. The CRS format allows fast row access and matrix-vector multiplications and was published in 1967 [28]. This format represents the matrix $\mathbf{A}$ by three one-dimensional arrays: *row_ptr*, *col_idx*, and *val*. The *row_ptr* array contains the start and end pointers of the nonzeroes of the rows; the size of this array is $m + 1$, where m is the number of rows of the matrix. The *col_idx* array contains column indices of the nonzeros. The size of this array is nnz, where nnz denotes the number of nonzeros of the matrix. Finally, the V array of size nnz contains values of nonzeros.

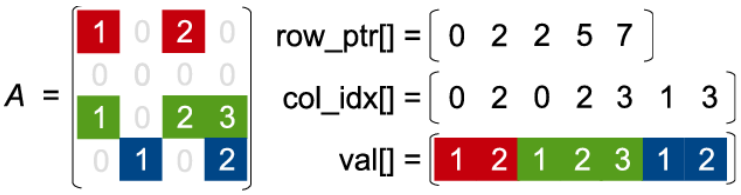


Figure 6.4: Illustration of the CSR format [112].

The library provides several methods with automatic step size control for solving ODEs. The error in each time step is calculated using the Runge-Kutta method and its embedded method. TSRK3BS implements the third-order Bogacki-Shampine 3(2) method with a second-order embedded method (known as ode23 in MATLAB, see (2.50)); the method has four stages with the First Same As Last (FSAL) property. The TSRK5DP implements the fifth-order Dormand-Prince 5(4) method with a fourth-order embedded method (known as ode45 in MATLAB, (2.49)); the method has seven stages with FSAL property. The TSRK5BS implements the fifth-order Bogacki-Shampine Runge-Kutta 5(4) method with a fourth-order embedded method, and it has eight stages and FSAL property. Robust Verner Runge-Kutta methods of orders 6(5), 7(6), and 8(7) are implemented as TSRK6VR, TSRK7VR and TSRK8VR, respectively. The TSRK6VR is the sixth-order robust Verner scheme with a fifth-order embedded method, nine stages, and the FSAL property. The TSRK7VR is the seventh-order robust Verner scheme with a sixth-order embedded method

and ten stages. Finally, the TSRK8VR is the eighth-order robust Verner scheme with a seventh-order embedded method with thirteen stages. For more information, see [10].

6.2.3 Intel Advisor Roofline model

Intel Advisor implements the cache-aware roofline model, which provides additional insight by addressing all memory/cache hierarchy levels. Sloped rooflines illustrate peak performance levels if all the data fit into the respective cache. Horizontal lines show the peak achievable performance levels if vectorization and other CPU resources are used effectively. Intel Advisor automatically runs quick benchmarks to measure the hardware limitations of a given machine.

Each dot in the roofline plot represents a loop or a function (see Figure 6.5). The size and color of the dots denote the relative execution time. The small green dots take relatively little time, while the red dots take the most time. According to Amdahl's law, loops that take the largest portion of the program's total run time lead to greater speedups than loops that take a smaller portion of the run time.

Note that in the classic roofline model, the arithmetic intensity of a kernel changes with the size of the problem or the optimization of the cache usage because the byte count was based only on DRAM traffic. On the contrary, in the cache-aware roofline model, the arithmetic intensity is fixed and tied to the algorithm. It only changes when the algorithm is changed. For more details, see Section 5.7.

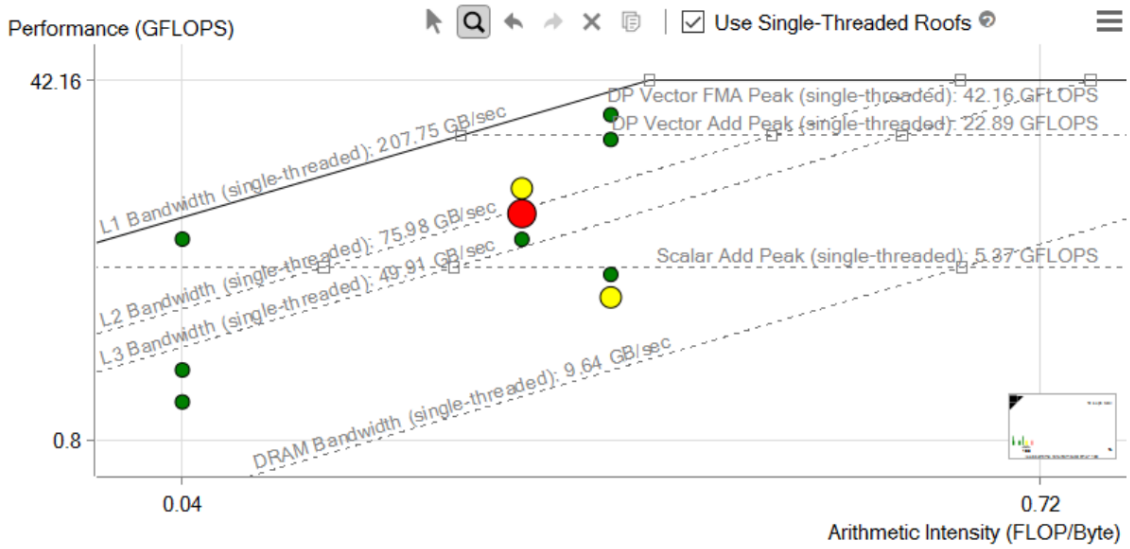


Figure 6.5: The example of Intel Advisor Roofline model [147].

6.3 Performance metrics

The performance metrics are used to evaluate, characterize, diagnose, and tune parallel performance [117]. Numerical results are presented using the following performance metrics: average time, speedup, speedup against the TSRK5DP solver, parallel efficiency, and parallel cost metrics.
The **average computation time** is defined as:

$$avgtime\,[\mathrm{s}] = \frac{\sum_{i=1}^{numRuns} t}{numRuns}\,, \tag{6.1}$$

where *numRuns* denotes the total number of runs.
The **speedup** of the solver is defined as:

$$speedup = \frac{t_1}{t_P}\,, \tag{6.2}$$

where t_1 denotes the serial computation time of one compute node, and t_P denotes the parallel computation time of P compute nodes. When $speedup > 1$, the speedup metric conveys performance improvement. On the contrary, when $speedup < 1$, the speedup metric conveys performance degradation. The ideal speedup (ideal scaling) is defined as $speedup = P$. The superlinear speedup is achieved when $speedup > P$.
The **speedup-against** ratio of the solver is defined as:

$$speedupAgainst = \frac{solverRef}{solver}\,, \tag{6.3}$$

where *solverRef* denotes the solver against which the speedup is calculated.
The $speedupAgainst \gg 1$ indicates a significantly faster computation using the given *solver*.
The **parallel efficiency** is defined as:

$$efficiency\,[\%] = \frac{speedup}{numProcs} \cdot 100 = \frac{\frac{t_S}{t_P}}{numProcs} \cdot 100\,, \tag{6.4}$$

where *numProcs* denotes the number of processes.
The **parallel cost** of an algorithm is defined as:

$$cost\,[node\text{-}hours] = avgtime \cdot numNodes\,, \tag{6.5}$$

where *avgtime* is the average time in hours and *numNodes* denotes the number of compute nodes.
The **parallel cost ratio** is defined as:

$$cost\ ratio = \frac{cost_P}{cost_1}\,, \tag{6.6}$$

where $cost_1$ is the parallel cost for one compute node and $cost_P$ is the parallel cost for more than one compute node. Parallel cost is calculated using (6.5).
The **parallel speedup-cost ratio** is defined as:

$$speedup\text{-}cost\ ratio = \frac{speedup}{cost\ ratio}\,, \tag{6.7}$$

where *speedup* is calculated by (6.2) and *cost ratio* by (6.6).

6.4 Characteristics of selected problems

The following text presents simulation results for different types of linear PDEs, namely, heat equation, wave equation, and telegraph equation. The PDEs are solved using the method of lines (see Section 3.4). The heat equation and the wave equation are discretized in the spatial domain using three- and five-point central difference formulas (see Table 3.2). The resulting large systems of ODEs are solved in parallel using the MTSM and Runge-Kutta methods.

The matrix-vector notation of the linear systems of ODEs is

$$\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{b}\,, \quad \mathbf{y}(t_0) = \mathbf{y}_0\,, \tag{6.8}$$

where $\mathbf{A}$ is the constant Jacobian matrix of size $m \times m$, and $\mathbf{b}$ is the right-hand-side vector of size $m \times 1$.

First, the experiments were performed in MATLAB for smaller problem sizes. The systems of ODEs were solved by the MTSM methods and standard methods such as ode45, ode23, and ode113.

6.4.1 Data sizes and solvers

Typical data sizes (denoted as S) are 128000, 256000, 512000, 1024000 and 2048000 ODEs. For each data size, four selected solvers compute the problem, namely, MTSM (classical implementation of MTSM), MTSM_PRECALC (parallel precalculation of MTSM), TSRK5DP (Dormand-Prince method 5(4)) [44] and TSRK8VR (Verner Runge-Kutta method of orders 8(7)) [166]. All Taylor series-based solvers were implemented using PETSc library routines. The MTSM implements the MTSM method (see Section 4.6). The MTSM_PREC implements the MTSM method with the precalculation of matrix $\mathbf{A}$ (see Section 4.9). The real data sizes for each problem differ. Table 6.2 shows the real data sizes for the matrix and vectors, and S denotes the number of segments in the spatial domain.

problem name	**matrix size** $m \times m$	**vector size** $m \times 1$
heat equation	$(S-1) \times (S-1)$	$(S-1) \times 1$
wave equation	$(2S-2) \times (2S-2)$	$(2S-2) \times 1$
telegraph equation	$(2S+2) \times (2S+2)$	$(2S+2) \times 1$

Table 6.2: Data sizes for different problems.

6.4.2 Cluster settings

The numerical experiments were performed on the Barbora supercomputer cluster, IT4Innovations National Supercomputing Center, Ostrava, Czech Republic using a various number of MPI processes, 36 processes per compute node. In total, 32 compute nodes were used; therefore, 1152 MPI processes were used. This research was supported by projects OPEN-22-47 and OPEN-25-51. All experiments were carried out in the production queue (qprod[9]) with the walltime set to 15 minutes. More details about the estimation of computation resources can be found in Appendix C.6. The execution priority of the job is determined by these job properties (in order of importance): queue priority, fair-share priority, and eligible time[10].

[9] https://docs.it4i.cz/general/resources-allocation-policy
[10] https://docs.it4i.cz/general/job-priority

6.4.3 General parameters

The general parameters for all simulation experiments are the following. The tolerances for all ode solvers are $RelTol = 1e^{-10}$ and $AbsTol = 1e^{-10}$, the tolerance for MTSM solvers is $tol = 1e^{-10}$. The maximum order of the MTSM is $maxORD = 64$. The maximum order for the MTSM_PRECALC method is $maxORD = 25$, which means that the matrix $\hat{\mathbf{A}}$ is precalculated to the order 25 (see (4.42)). Maximum simulation time $t_{max} = 10000 \cdot h$.

Note that for the heat equation (Sections 6.5 and 6.6) and the wave equation (Sections 6.7 and 6.8), the spatial step-size is always $\Delta x = L/S = 0.1$. Therefore, the size of the spatial domain L and the number of segments S are always set accordingly to meet this requirement. The coefficients of the central difference formulas are in Table 3.2. Problem-dependent parameters are further specified in the corresponding sections.

6.5 Heat equation – three-point central difference

The heat equation is the parabolic PDE that describes the distribution of heat in a given spatial domain x and time domain t. The one-dimensional heat equation is defined as

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} + f(x,t), \quad (x,t) \in (0,L) \times \langle 0, t_{max} \rangle \,, \tag{6.9}$$

where α is thermal diffusivity and L denotes the length of the rod. The function $f(x,t)$ characterizes the intensity of internal sources and t_{max} denotes the maximum simulation time. The variable $u = u(x,t)$ describes the temperature of the rod at the point x and at time t.
The initial conditions are

$$u(x,0) = 25, \quad 0 < x < L \,. \tag{6.10}$$

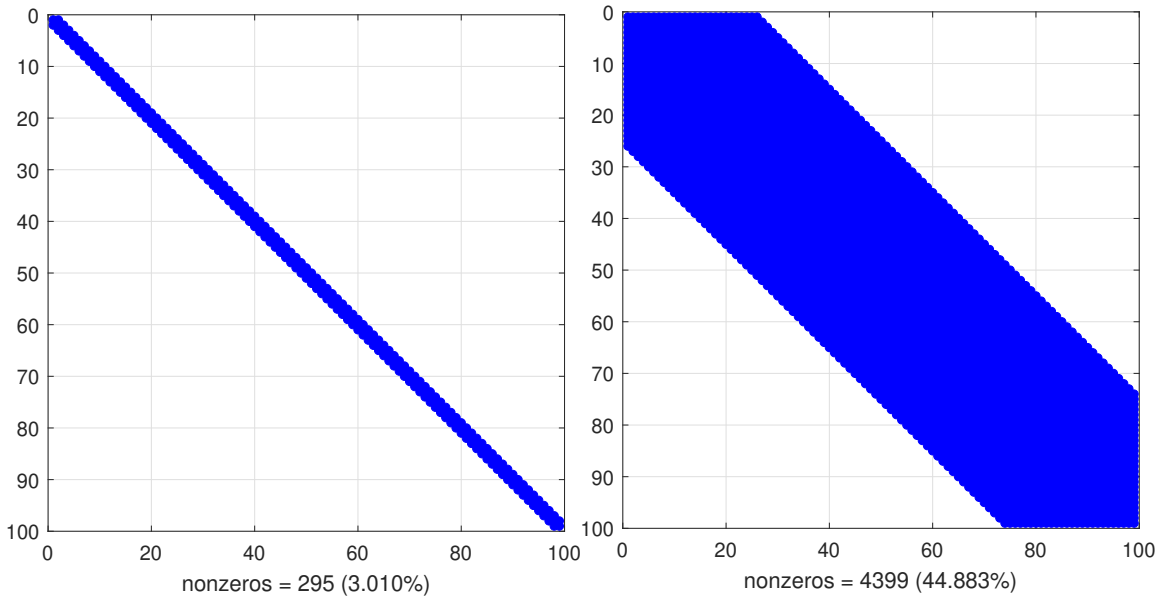
Dirichlet boundary conditions are

$$u(0,t) = 100, u(L,t) = 25, \quad 0 \leq t \leq t_{max} \,. \tag{6.11}$$

The resulting system of ODEs $\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{b}\,, \mathbf{y}(t_0) = \mathbf{y}_0$ arising from MOL is in the form:

$$\mathbf{A} = \begin{pmatrix} -2 & 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 1 & -2 & 1 & 0 & & & & \vdots \\ 0 & 1 & -2 & 1 & \ddots & & & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & & & \ddots & 1 & -2 & 1 & 0 \\ \vdots & & & & 0 & 1 & -2 & 1 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 & -2 \end{pmatrix}, \quad \mathbf{y} = u(x,0)\,, \quad \mathbf{b} = \begin{pmatrix} \alpha u(0,t) \\ \vdots \\ \alpha u(L,t) \end{pmatrix}, \tag{6.12}$$

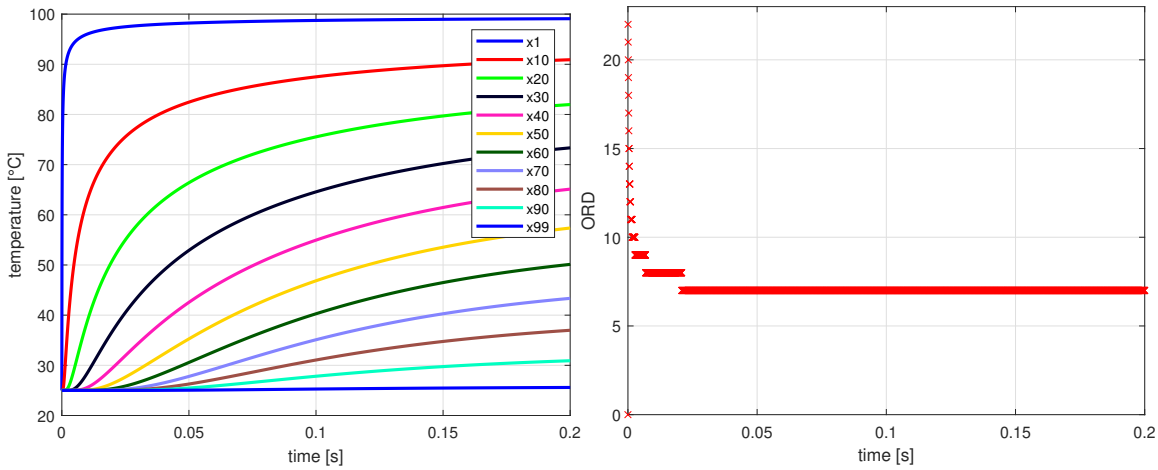
The sparsity patterns of the matrices $\mathbf{A}$ and $\hat{\mathbf{A}}$ precalculated using (4.46) for $S = 100$ segments are in Figure 6.6.



(a) Matrix $\mathbf{A}$ (b) Matrix $\hat{\mathbf{A}}$ precalculated using (4.46)

Figure 6.6: Sparsity patterns of input matrices, heat equation, three-point central difference formula, $S = 100$.

The numerical solution of the heat equation discretized in the spatial domain by the three-point central finite difference formula and the order of the MTSM for the segments $S = 100$ are shown in Figure 6.7.



(a) Numerical solution (temperature profile) (b) Order of the MTSM

Figure 6.7: Numerical solution and order of MTSM, heat equation, three-point central difference formula, $S = 100$, $h = 2e^{-4}$, $t_{max} = 1000 \cdot h$.

The parameters for the simulation experiments are as follows. The tolerances for all ode solvers are $RelTol = 1e^{-10}$ and $AbsTol = 1e^{-10}$, the tolerance for MTSM solvers is $tol = 1e^{-10}$, and the maximum simulation time $t_{max} = 10000 \cdot h$. The thermal diffusivity is $\alpha = 113\,\mathrm{mm/s}$.

6.5.1 Results overview – three-point central difference formula

Table 6.3 shows the number of integration steps and the average step sizes for each solver. The MTSM uses a step size approximately 2.7 times larger than the TSRK5DP solver and approximately 1.9 times larger than the TSRK8VR solver. The average order of the MTSM for the three-point central difference formulas is 20.6.

solver	# steps	average h
MTSM_PRECALC	10000	$2.00e^{-04}$
MTSM	10000	$2.00e^{-04}$
TSRK5DP	27377	$7.30e^{-05}$
TSRK8VR	18740	$1.07e^{-04}$

Table 6.3: Number of integration steps and average step sizes, heat equation, three-point central difference formula.

Tables 6.4 and 6.5 summarize the average parallel efficiency (6.4) and speedup against the TSRK5DP (6.3) for each problem size. These results are averages of values for 1–32 compute nodes. Table cells where the parallel efficiency is greater than or equal to 50% are marked in green. The cells in the table showing the speedup ratio with respect to the TSRK5DP solver are also marked in green. Notice that for the problem size $S = 2048000$, the TSRK8VR solver did not calculate the result because the maximum walltime was exceeded.

Table 6.4 shows the average parallel efficiency. The MTSM_PRECALC solver offers an efficiency greater than 50% for all problem sizes. The MTSM_PRECAL solver is always faster than the TSRK5DP solver, as shown in Table 6.5. Finally, the MTSM_PRECALC is the fastest of all solvers for all problem sizes.

solver	problem size				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	**91.93**	**198.46**	**188.85**	**189.95**	148.34
MTSM	17.49	23.85	34.12	49.22	80.43
TSRK5DP	34.52	50.48	89.63	155.11	**204.84**
TSRK8VR	40.02	60.50	113.46	171.13	–

Table 6.4: Average **efficiency** (6.4) comparison for 1–32 nodes, heat equation, three-point central difference formula.

solver	problem size				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	7.93	6.43	5.27	4.23	2.88
MTSM	0.29	0.35	0.44	0.57	0.72
TSRK8VR	0.75	0.74	0.69	0.68	–

Table 6.5: Average **speedup against TSRK5DP** (6.3) comparison for 1–32 nodes, heat equation, three-point central difference formula.

Table 6.6 indicates whether a solver calculates the result for 1–32 nodes for a given problem size. The letter "Y" means "Yes", and the letter "N" means "No". The notation "N (Xn)" implies that a given solver calculated the result for the maximum number of compute nodes denoted as "(Xn)". For the three-point central difference formula and $S = 2048000$, only the TSRK8VR solver did not compute the result for all 32 compute nodes and was able to use eight compute nodes at most.

solver	**problem size**				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	Y	Y	Y	Y	Y
MTSM	Y	Y	Y	Y	Y
TSRK5DP	Y	Y	Y	Y	Y
TSRK8VR	Y	Y	Y	Y	N (8 n)

Table 6.6: Yes/No table, heat equation, three-point central difference formula.

Table 6.7 shows, for each problem size, the number of compute nodes where the parallel efficiency is $E \geq 50\%$. Notice that for the MTSM_PRECALC solver $E \geq 50\%$ for 1–32 compute nodes and all problem sizes. For the MTSM, the number of nodes, where $E \geq 50\%$, gradually increases with each problem size up to 26 nodes. The TSRK5DP solver reaches efficiency $E \geq 50\%$ for 1–32 compute nodes for problem sizes $S \geq 512000$ equations. Finally, the TSRK8VR solver provides efficiency $E \geq 50\%$ for all 32 compute nodes when the problem sizes are $S = 512000$ and $S = 1024000$. For $S = 2048000$, it drops to eight nodes.

solver	**problem size**				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	32	32	32	32	32
MTSM	3	4	9	12	26
TSRK5DP	7	12	32	32	32
TSRK8VR	8	18	32	32	8

Table 6.7: Maximum number of nodes where efficiency $E \geq 50\%$, heat equation, three-point central difference formula.

Table 6.8 shows the maximum number of nodes to use for the calculation with respect to the speedup-cost ratio, i.e., where the speedup-cost tradeoff curve reaches a maximum.

solver	**problem size**				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	32	32	32	32	32
MTSM	1	4	4	4	16
TSRK5DP	4	4	8	32	32
TSRK8VR	8	8	16	32	8

Table 6.8: Parallel cost for all problem sizes and solvers, heat equation, three-point central difference formula.

The numerical results for the three-point central difference formula and different problem sizes follow. Detailed results, including results for $S = 256000$ and $S = 1024000$, can be found in Appendix C.1.

6.5.2 $S = 128000$, three-point central difference formula

Two quadruple graphs are presented for each problem size. Both visualize the metrics mentioned in Section 6.3. The first quadruple is depicted in Figure 6.8. The upper left subfigure shows the average computation time (6.1) where the dashed lines show the ideal average times for the given number of processes. The parallel efficiency (6.4) is shown in the upper right subfigure. The lower left subfigure depicts the speedup for each solver (6.2), and the dashed line shows the ideal speedup for the given number of processes. The last one shows the speedup ratio with respect to the TSRK5DP solver (ode45 in MATLAB) (6.3), the *speedup=TSRK5DP/solver* $\gg 1$ indicates significantly faster computation using the given *solver*.

The second quadruple (Figure 6.9) shows the parallel cost metrics for each solver. Each subfigure contains three curves. The curve labeled with the solver's name shows the parallel speedup of a given solver (6.2). The magenta curve represents the parallel cost ratio (6.6), and the cyan curve shows the parallel speedup-cost tradeoff (6.7).

We can see that the ideal number of nodes for the MTSM_PRECALC solver is 32 because the speedup (red curve) increases proportionally with the parallel cost ratio (magenta curve). Therefore, the speedup-cost ratio (cyan curve) also increases. For the MTSM solver, it is not beneficial to use more than 1 compute node. Note that for the number of nodes greater than one, the speedup is almost constant while the cost ratio grows almost linearly. In the case of the TSRK5DP solver, it is ideal to use four compute nodes, and for TSRK8VR, eight nodes.

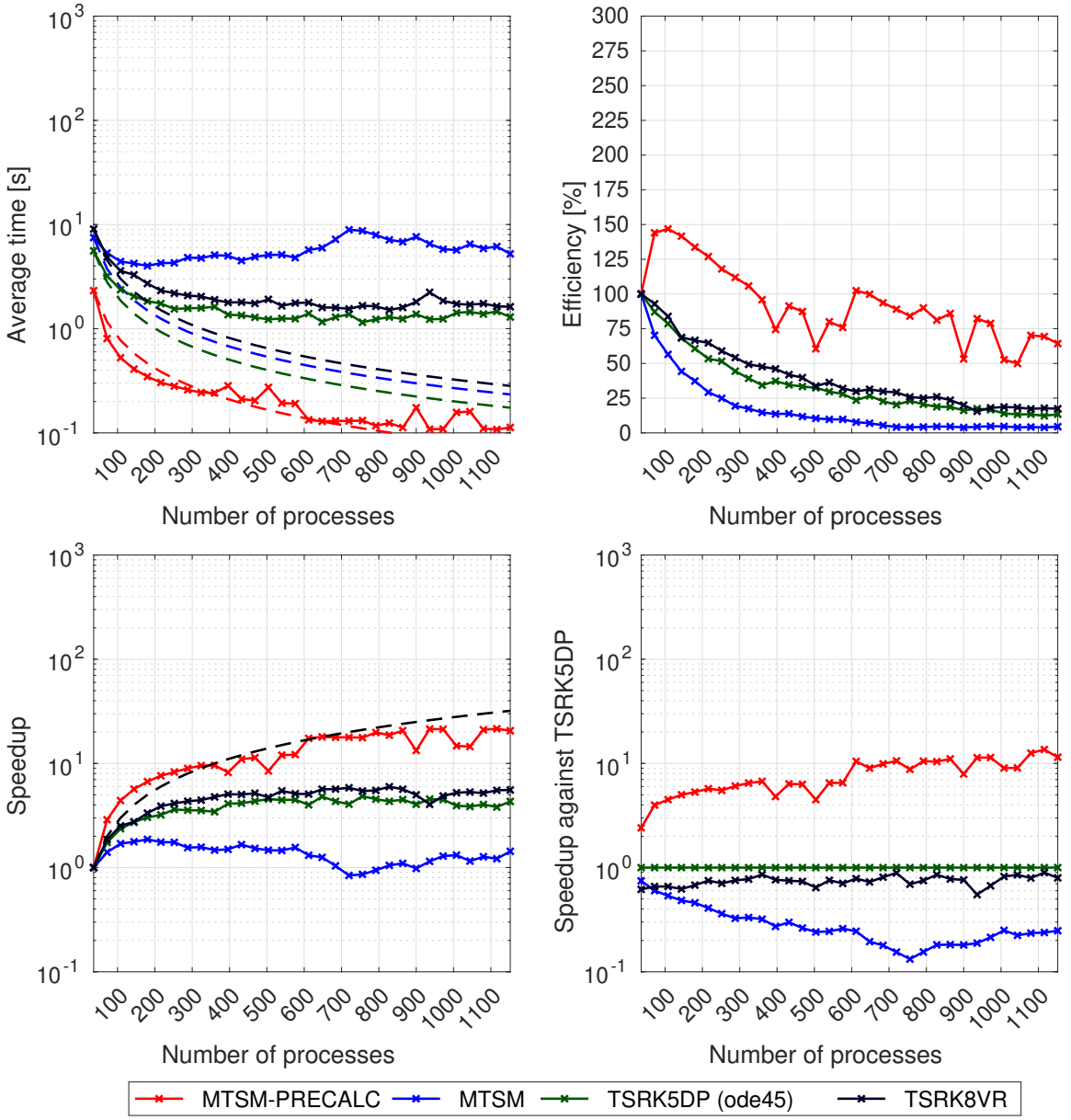


Figure 6.8: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 128000$, heat equation, three-point central difference formula.

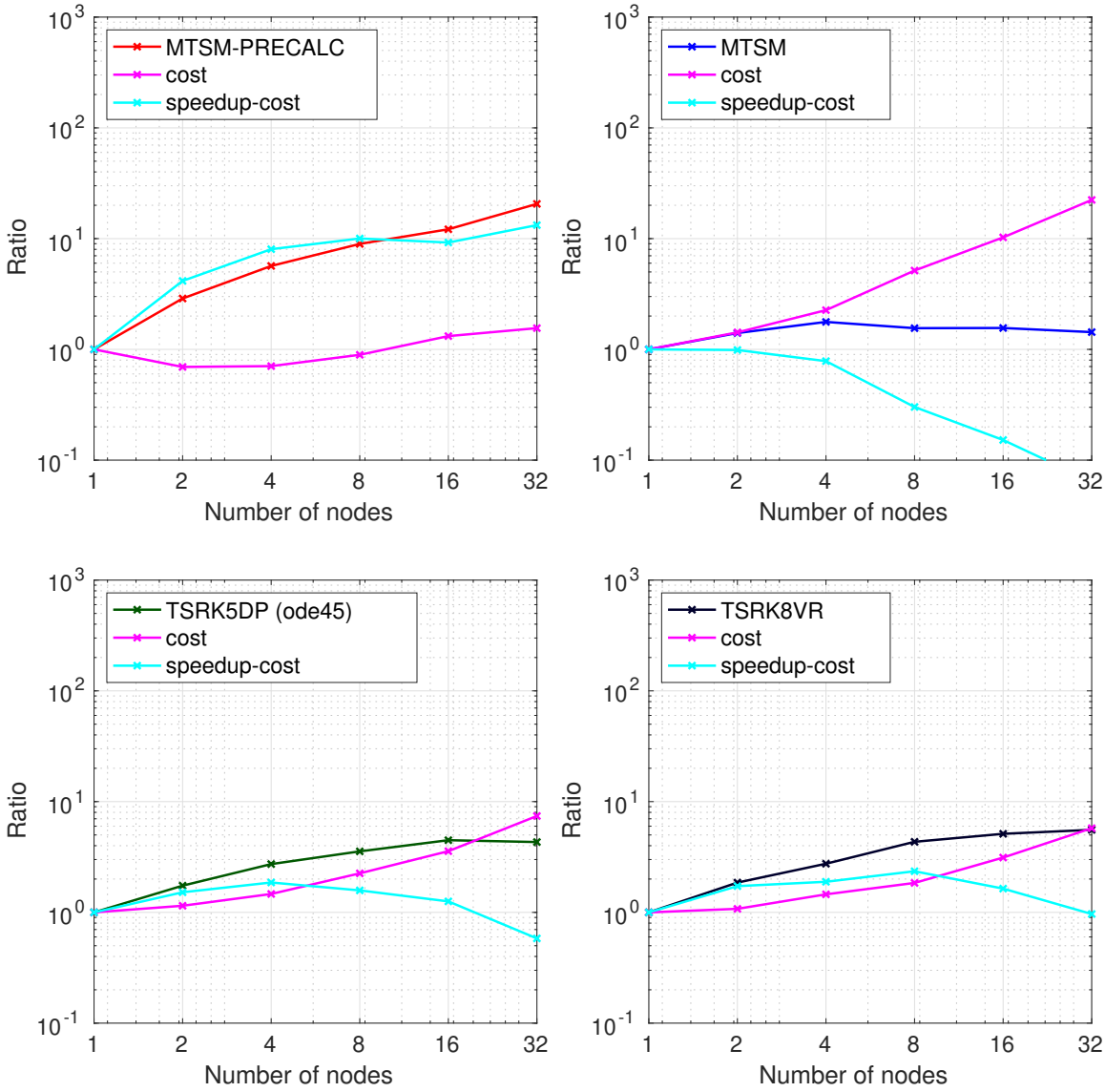


Figure 6.9: Parallel cost ratio and speedup-cost ratio, $S = 128000$, heat equation, three-point central difference formula.

6.5.3 $S = 512000$, three-point central difference formula

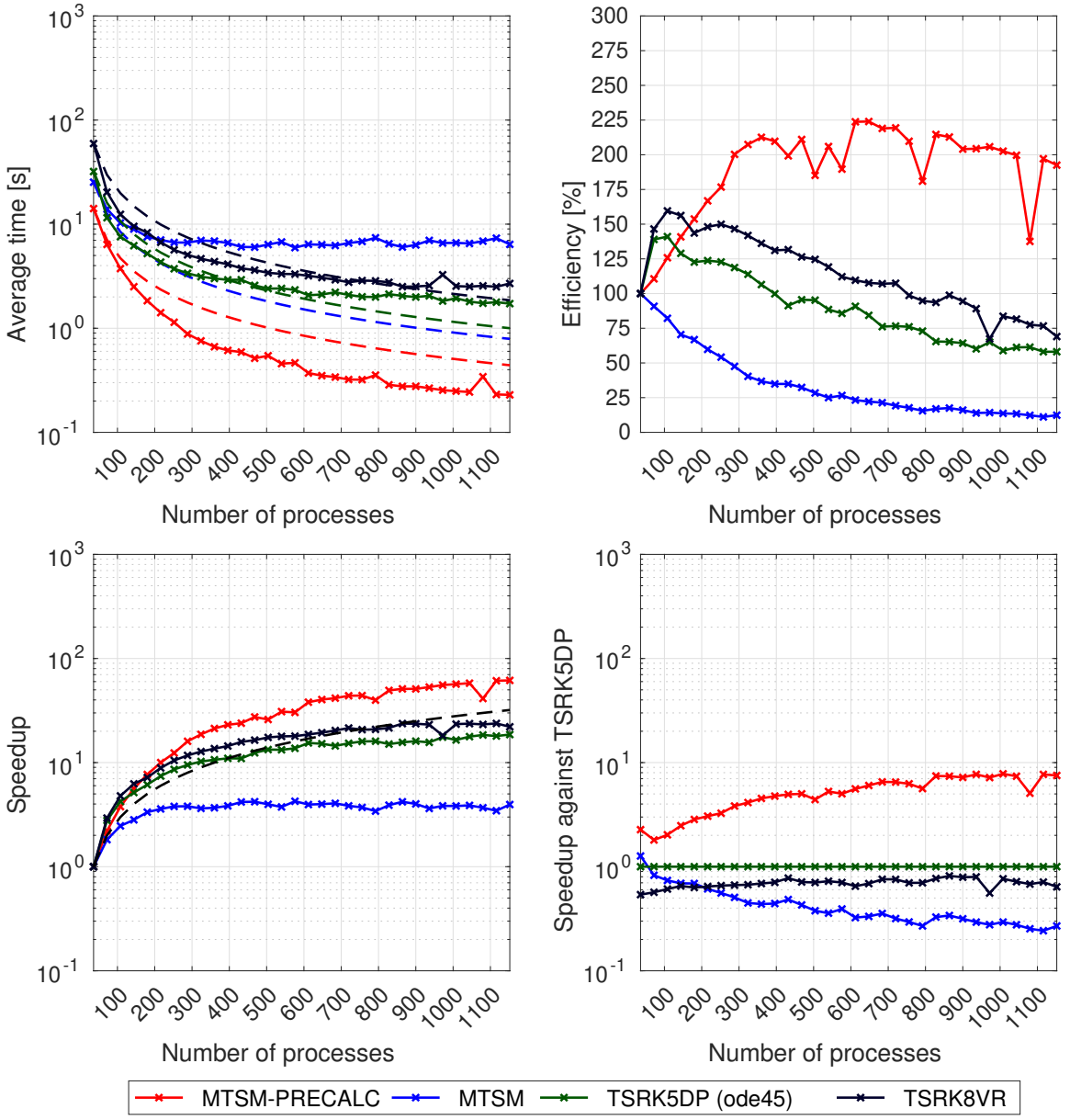


Figure 6.10: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 512000$, heat equation, three-point central difference formula.

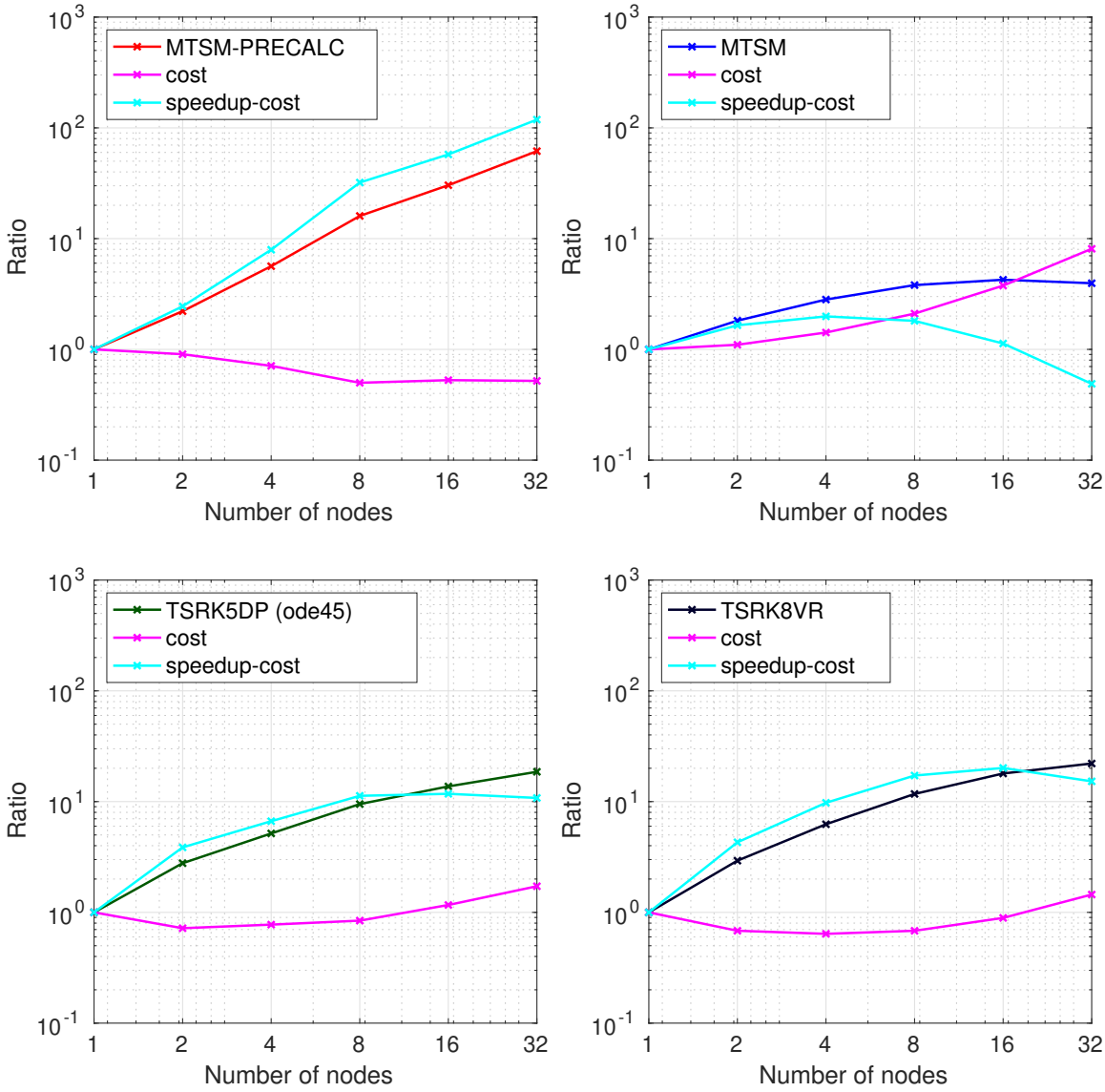


Figure 6.11: Parallel cost ratio and speedup-cost ratio, $S = 512000$, heat equation, three-point central difference formula.

6.5.4 $S = 2048000$, three-point central difference formula

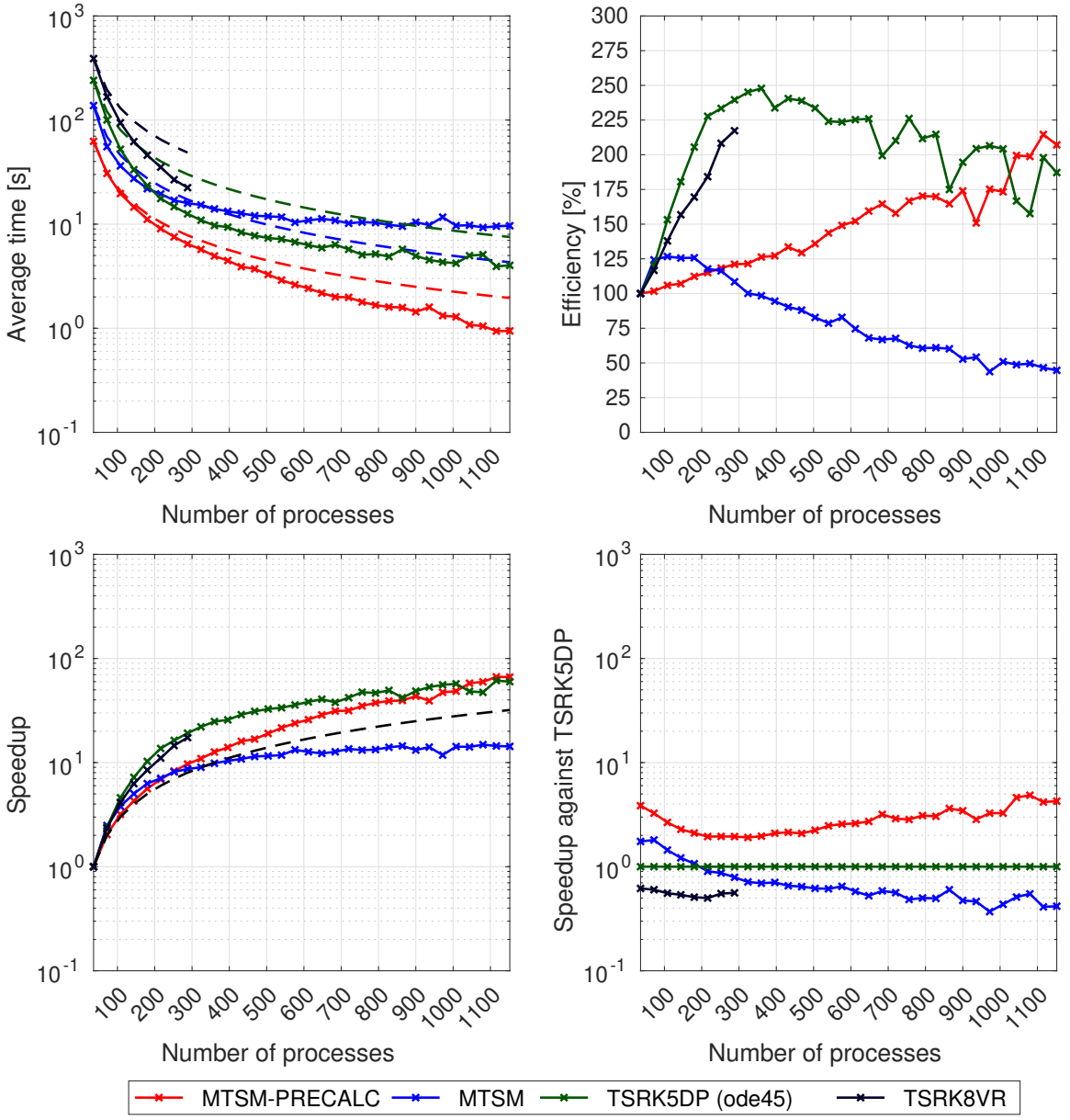


Figure 6.12: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 2048000$, heat equation, three-point central difference formula.

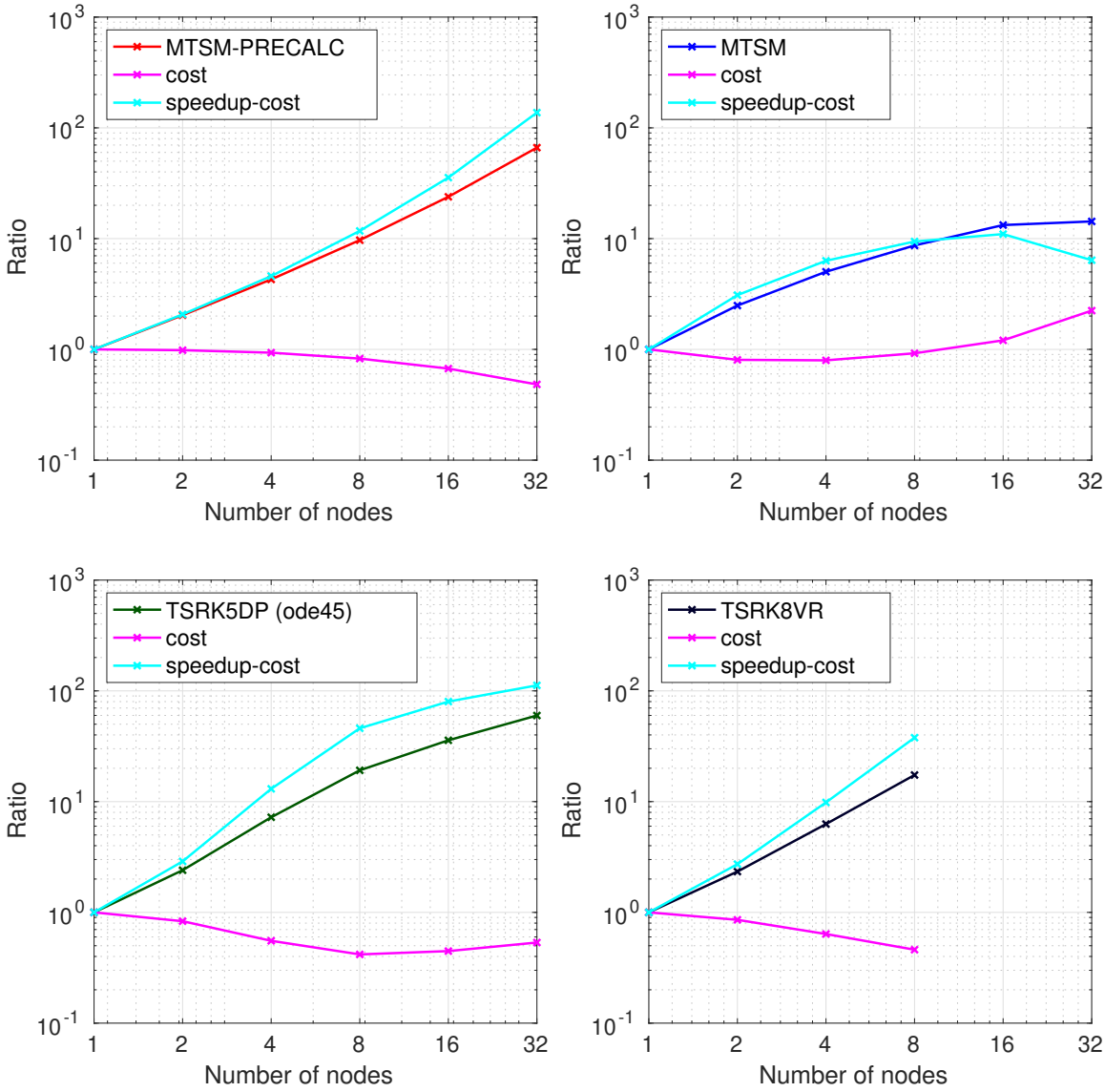
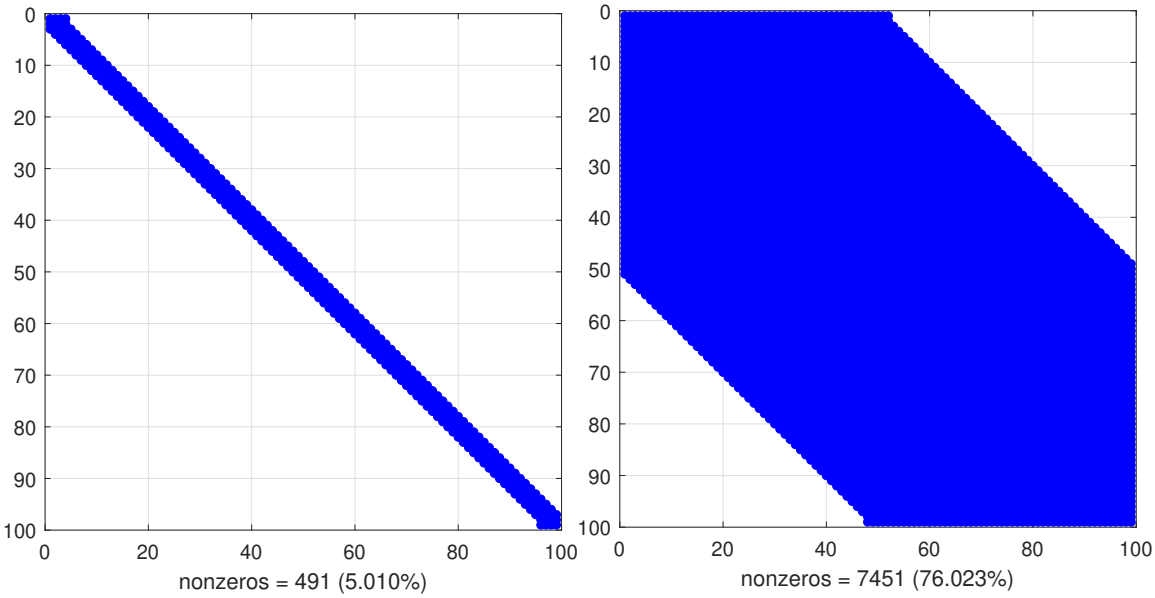


Figure 6.13: Parallel cost ratio and speedup-cost ratio, $S = 2048000$, heat equation, three-point central difference formula.

6.6 Heat equation – five-point central difference formula

The definition of the problem and the simulation parameters are the same as for the heat equation discretized with the three-point central difference formulas (see Section 6.5). The sparsity patterns of the matrices $\mathbf{A}$ and $\hat{\mathbf{A}}$ precalculated using (4.46) for $S = 100$ segments are shown in Figure 6.14.



(a) Matrix $\mathbf{A}$ (b) Matrix $\hat{\mathbf{A}}$ precalculated using (4.46)

Figure 6.14: Sparsity patterns of input matrices, heat equation, five-point central difference formula, $S = 100$.

6.6.1 Results overview – five-point central difference formula

Table 6.9 shows the number of integration steps and average step sizes for each solver. The step size of the MTSM solvers is approximately 2.1 times larger than that of the TSRK5DP solver and approximately 1.4 times larger than that of the TSRK8VR solver. The average order of the MTSM for the five-point central difference formulas is 20.6.

solver	**# steps**	**average h**
MTSM_PRECALC	10000	$1.50e^{-04}$
MTSM	10000	$1.50e^{-04}$
TSRK5DP	20542	$7.30e^{-05}$
TSRK8VR	14059	$1.07e^{-04}$

Table 6.9: Number of integration steps and average step sizes, heat equation, five-point central difference formula.

Tables 6.10 and 6.11 summarize the average efficiency and speedup against the TSRK5DP for each problem size.

solver	problem size				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	**94.95**	**152.95**	**184.74**	**191.74**	120.50
MTSM	17.65	24.18	36.37	49.93	110.33
TSRK5DP	34.28	50.35	106.18	162.89	**224.19**
TSRK8VR	41.36	61.69	129.36	176.25	–

Table 6.10: Average **efficiency** (6.4) comparison for 1–32 nodes, heat equation, five-point central difference formula.

solver	problem size				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	6.05	5.17	2.60	3.08	1.28
MTSM	0.31	0.39	0.45	0.60	0.75
TSRK8VR	0.79	0.73	0.70	0.66	–

Table 6.11: Average **speedup against TSRK5DP** (6.3) comparison for 1–32 nodes, heat equation, five-point central difference formula.

Table 6.12 shows that for the five-point central difference formula, only the TSRK8VR solver did not compute the result for all 32 compute nodes, and it was able to use six compute nodes at most.

solver	problem size				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	Y	Y	Y	Y	Y
MTSM	Y	Y	Y	Y	Y
TSRK5DP	Y	Y	Y	Y	Y
TSRK8VR	Y	Y	Y	Y	N (6 n)

Table 6.12: Yes/No table, heat equation, five-point central difference formula.

Table 6.13 shows, for each problem size (denoted as S), the number of compute nodes where the parallel efficiency $E \geq 50\%$. The parallel efficiency of the MTSM_PRECALC solver is always greater than 50% for 1–32 compute nodes for all problem sizes. The efficiency of the MTSM solver increases with each problem size, and for $S = 2048000$ it reaches the parallel efficiency $E \geq 50\%$ for all 32 compute nodes. The TSRK5DP solver offers efficiency $E \geq 50\%$ for all 32 compute nodes for problem sizes $S \geq 512000$ equations. Finally, the TSRK8VR solver offers efficiency $E \geq 50\%$ for all 32 compute nodes when the problem sizes are $S = 512000$ and $S = 1024000$. For $S = 2048000$, it drops to 6 nodes.

solver	problem size				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	32	32	32	32	32
MTSM	3	4	8	13	32
TSRK5DP	7	15	32	32	32
TSRK8VR	8	19	32	32	6

Table 6.13: Maximum number of nodes where efficiency $E \geq 50\%$, heat equation, five-point central difference formula.

Table 6.14 shows the maximum number of nodes to use for the calculation with respect to the speedup-cost ratio, i.e., where the speedup-cost tradeoff curve reaches a maximum.

solver	problem size				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	32	32	32	32	32
MTSM	2	4	4	16	8
TSRK5DP	8	8	32	32	32
TSRK8VR	8	16	32	32	4

Table 6.14: Parallel cost for all problem sizes and solvers, heat equation, five-point central difference formula.

The numerical results for the five-point central difference formula and different number of ODEs follow. Detailed results, including results for $S = 256000$ and $S = 1024000$, can be found in Appendix C.2.

6.6.2 $S = 128000$, five-point central difference formula

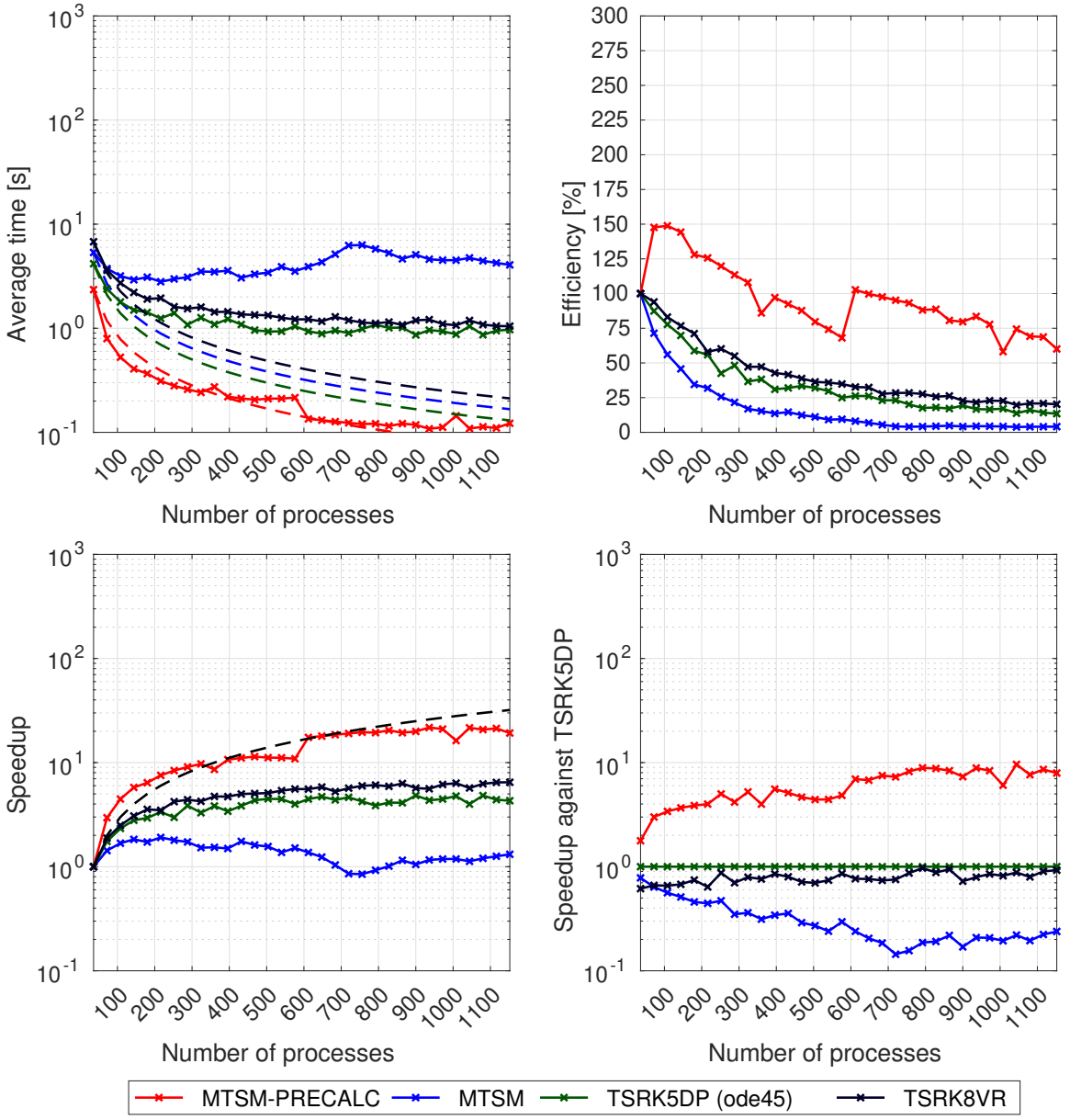


Figure 6.15: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 128000$, heat equation, five-point central difference formula.

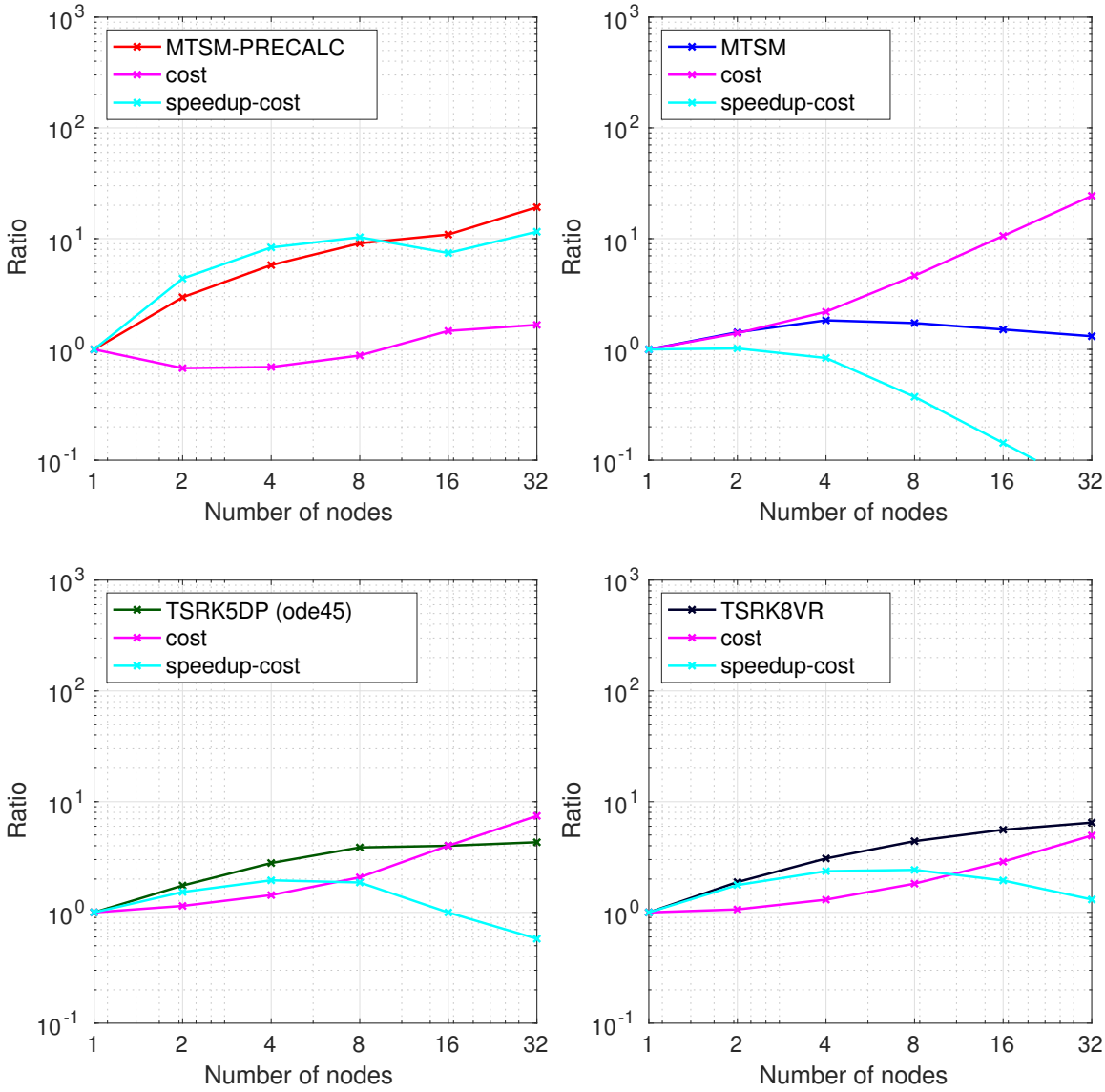


Figure 6.16: Parallel cost ratio and speedup-cost ratio, $S = 128000$, heat equation, five-point central difference formula.

6.6.3 $S = 512000$, five-point central difference formula

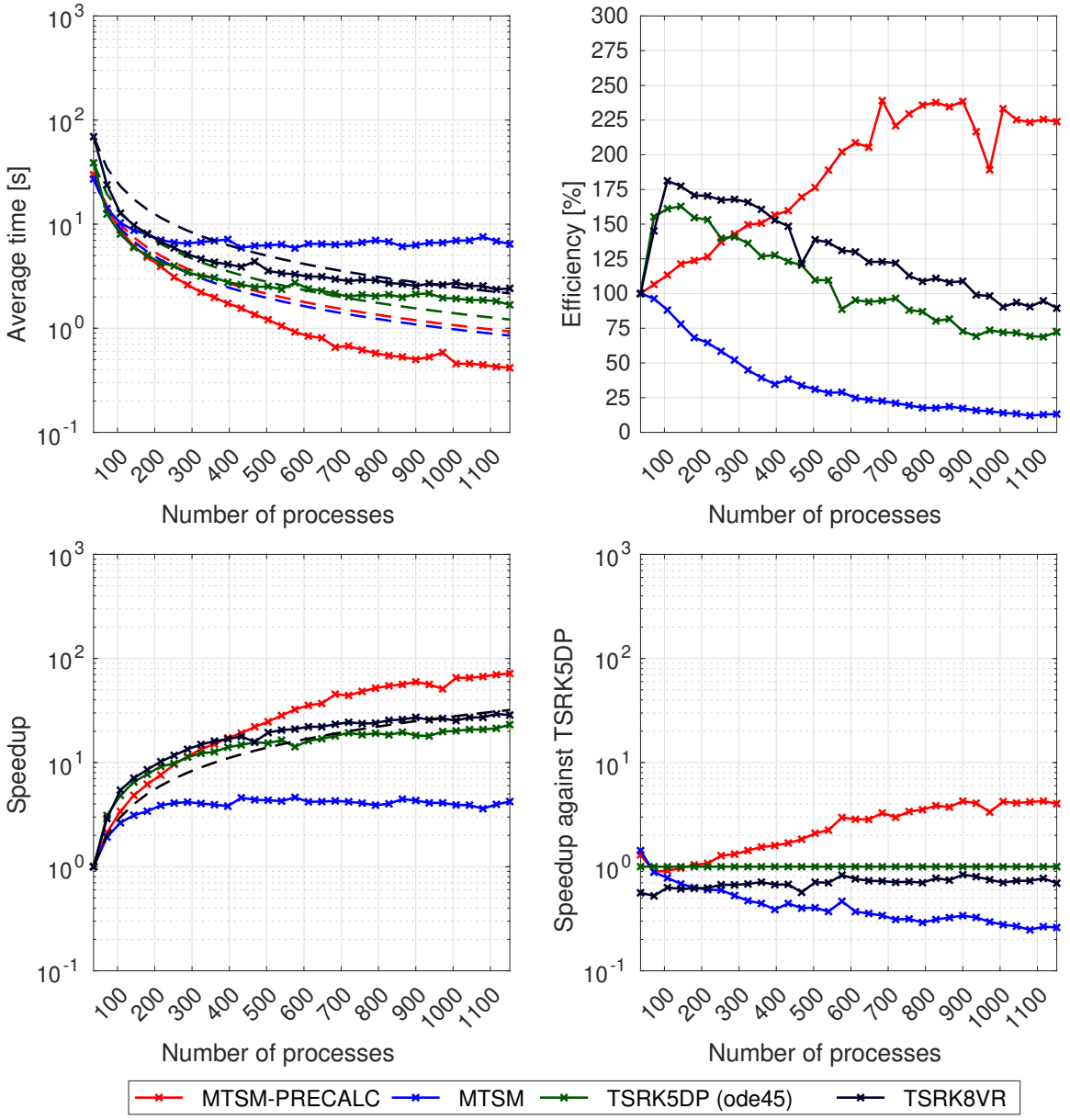


Figure 6.17: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 512000$, heat equation, five-point central difference formula.

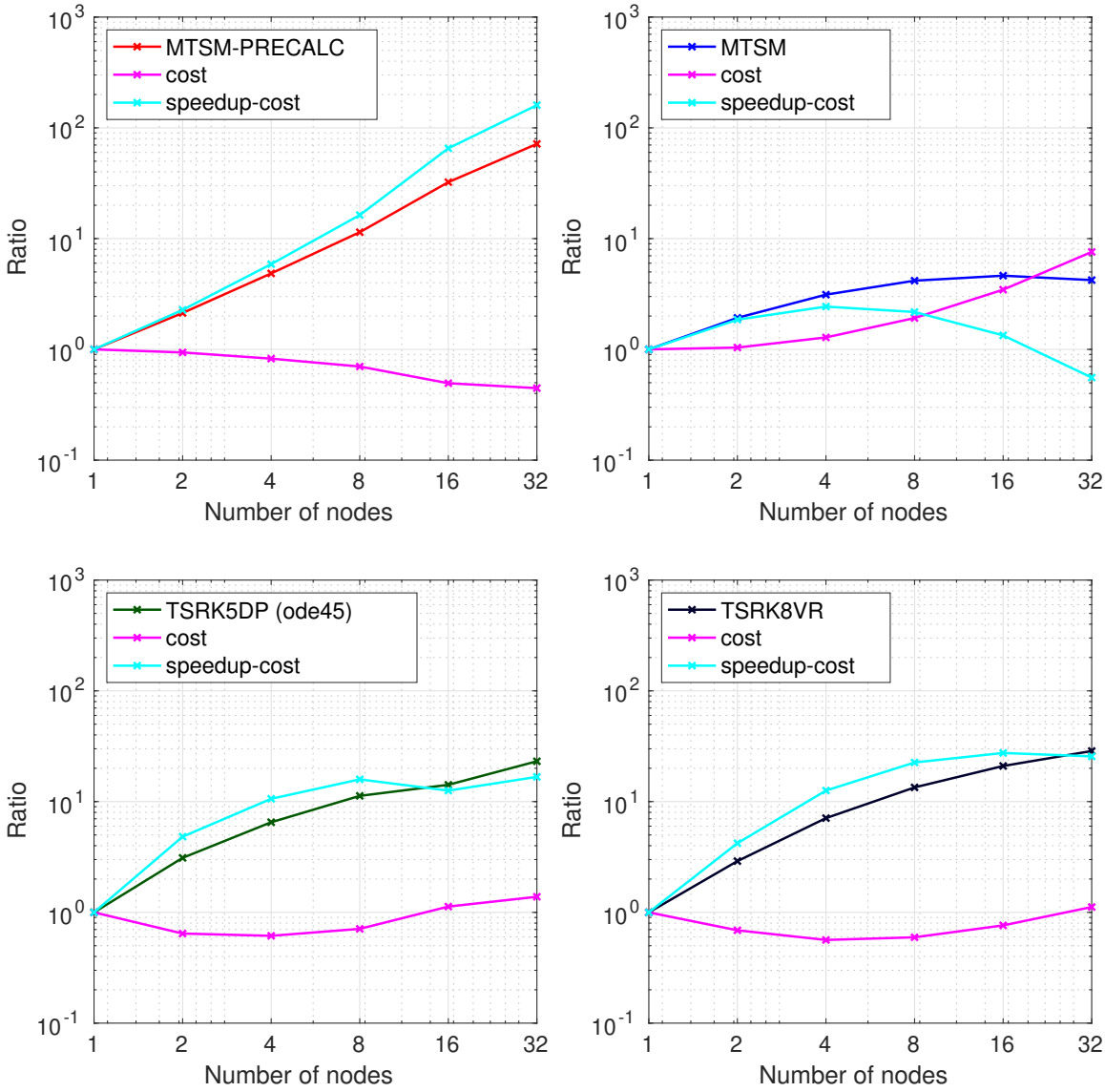


Figure 6.18: Parallel cost ratio and speedup-cost ratio, $S = 512000$, heat equation, five-point central difference formula.

6.6.4 $S = 2048000$, five-point central difference formula

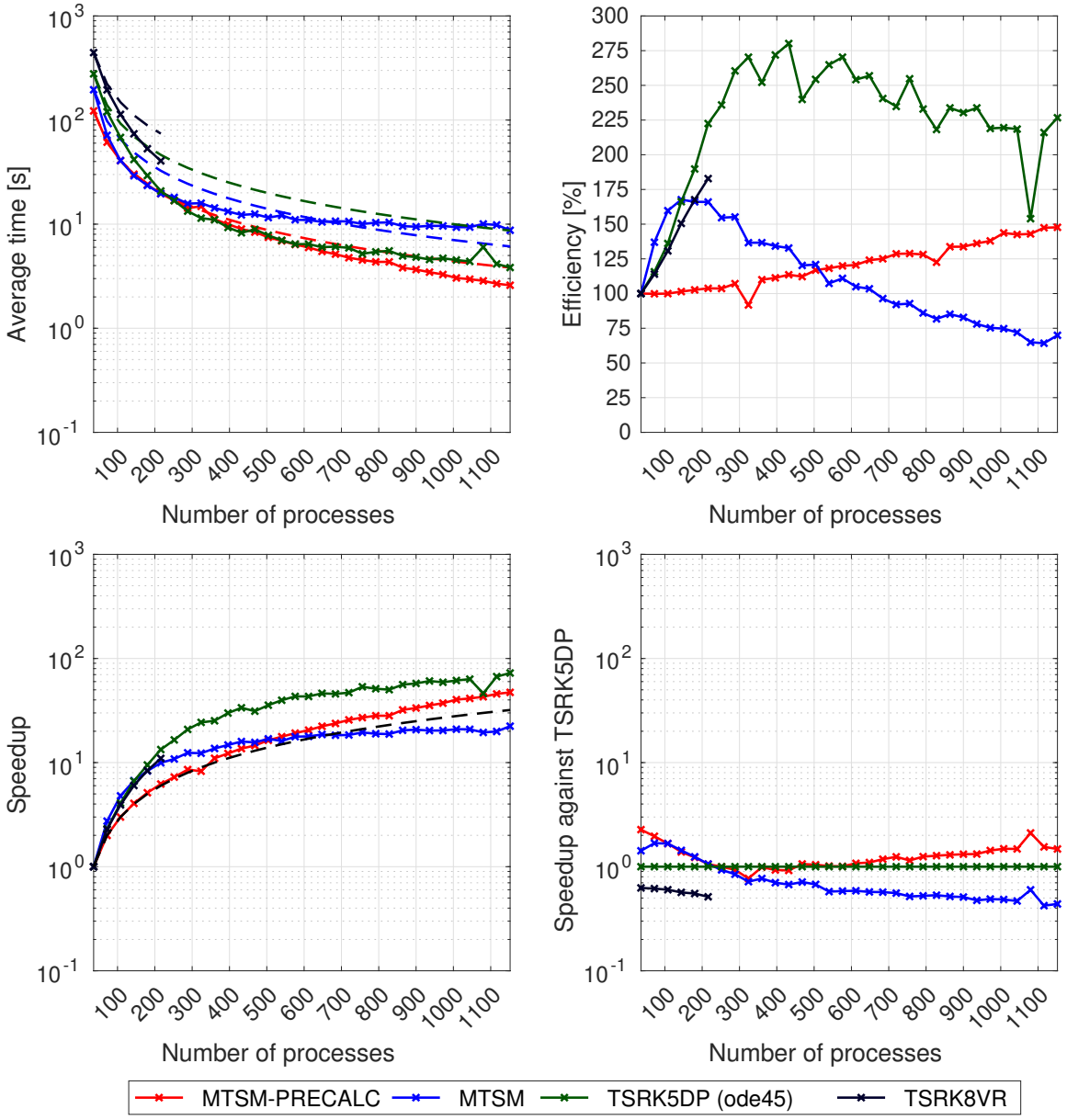


Figure 6.19: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 2048000$, heat equation, five-point central difference formula.

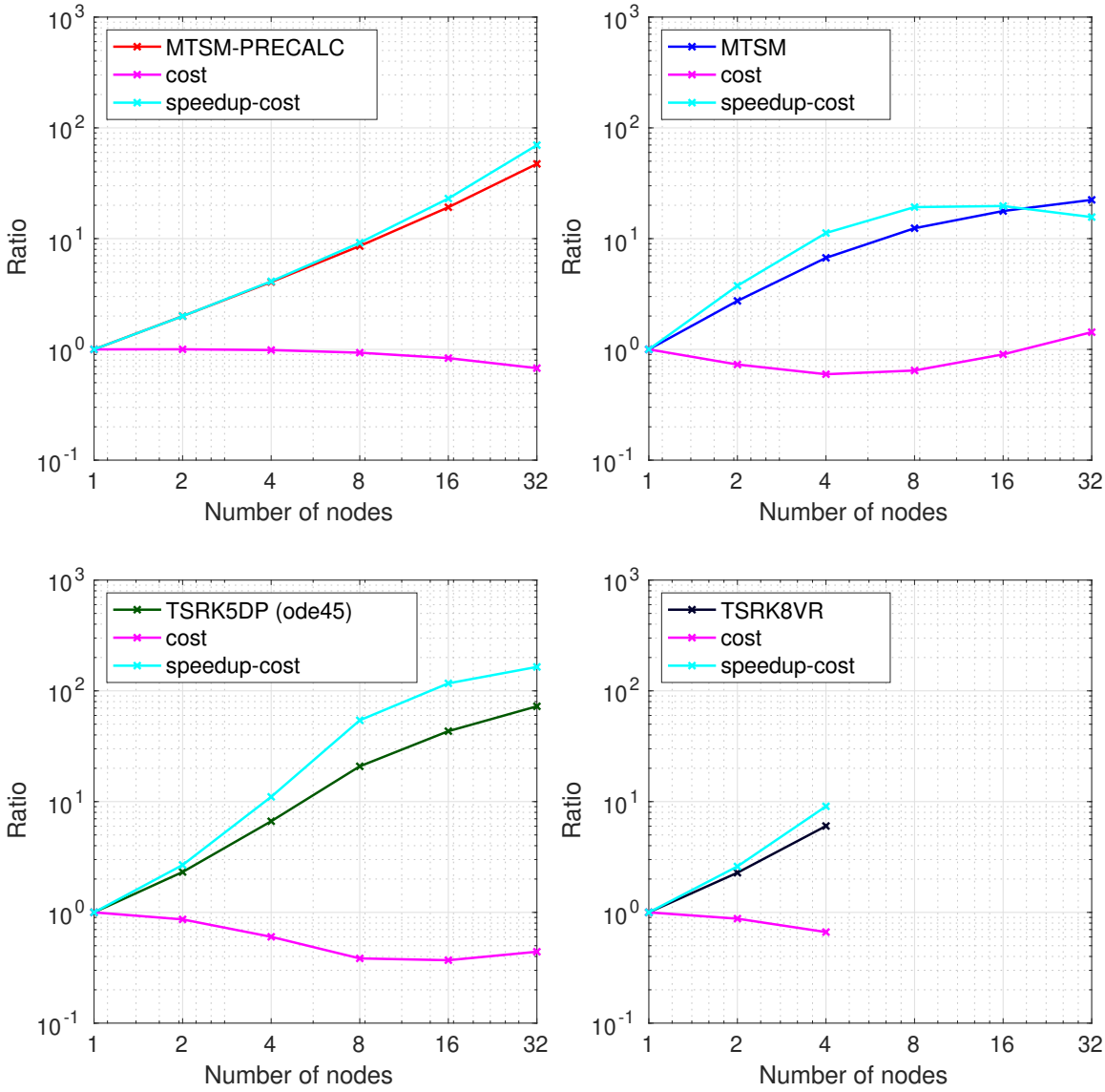


Figure 6.20: Parallel cost ratio and speedup-cost ratio, $S = 2048000$, heat equation, five-point central difference formula.

6.7 Wave equation – three-point central difference formula

The wave equation is the hyperbolic PDE [153] of the second order (6.13).

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial^2 u(x,t)}{\partial t^2} \quad (x,t) \in (0,L) \times \langle 0, t_{max} \rangle \,. \tag{6.13}$$

The homogeneous Dirichlet boundary conditions are defined as follows:

$$u(0,t) = u(L,t) = 0, \quad 0 \leq t \leq t_{max} \,. \tag{6.14}$$

where L is the length of the string and t_{max} is the maximum simulation time. The Cauchy initial values follow:

$$u(x,0) = \sin(\pi x) \,, \tag{6.15}$$

$$\frac{\partial u(x,0)}{\partial t} = 0 \,, \quad 0 < x < L \,. \tag{6.16}$$

The wave equation describes the oscillations of an ideal string of a specified length. Both ends of the string are fixed in time (see the boundary conditions (6.14)). The initial velocity of the string is zero (see equation (6.16)). The initial position of the string is modelled as a sine function (see equation (6.15)). The analytical solution of the wave equation follows:

$$u \quad = \cos(\pi t) \sin(\pi x). \tag{6.17}$$

Partial derivatives with respect to time and space are

$$\frac{\partial u}{\partial t} = -\pi \sin(\pi t) \sin(\pi x), \quad \frac{\partial^2 u}{\partial t^2} = -\pi^2 \cos(\pi t) \sin(\pi x) \,, \tag{6.18}$$

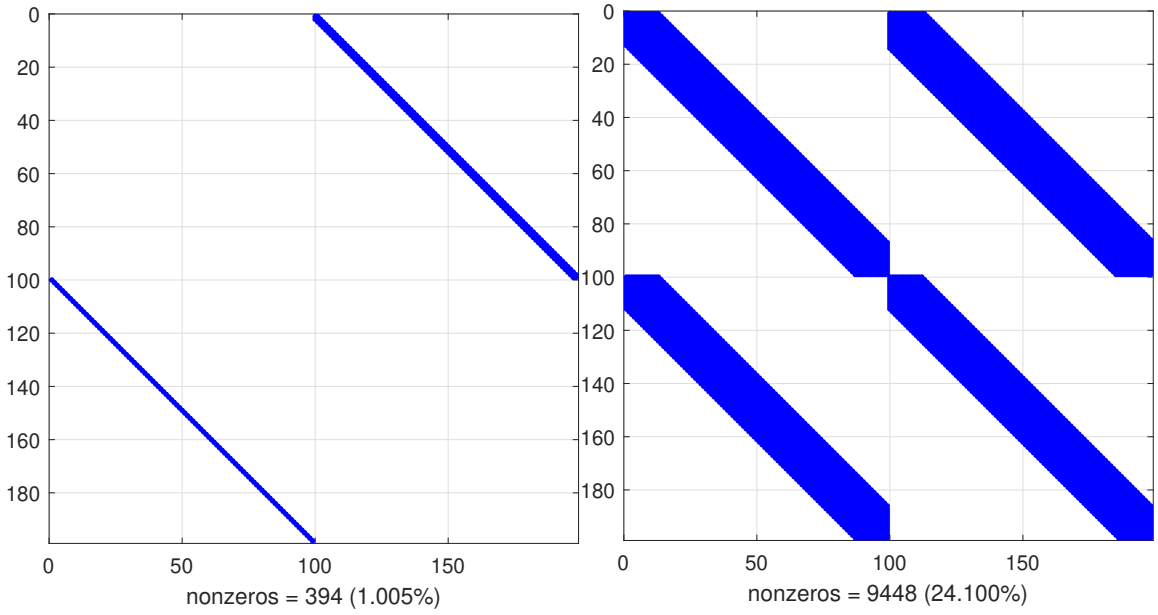
and

$$\frac{\partial u}{\partial x} = \pi \cos(\pi x) \cos(\pi t) \,, \quad \frac{\partial^2 u}{\partial x^2} = -\pi^2 \sin(\pi x) \cos(\pi t) \,, \tag{6.19}$$

respectively. The resulting system of ODEs $\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{b} \,, \mathbf{y}(t_0) = \mathbf{y}_0$ arising from MOL is in the form:

$$\mathbf{A} = \left(\begin{array}{c|c} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \hline \mathbf{A}_{21} & \mathbf{A}_{22} \end{array}\right) \,, \quad \mathbf{y}_0 = \begin{pmatrix} \sin(\pi x) \\ \mathbf{0} \end{pmatrix} \,, \quad \mathbf{b} = \mathbf{0} \,. \tag{6.20}$$

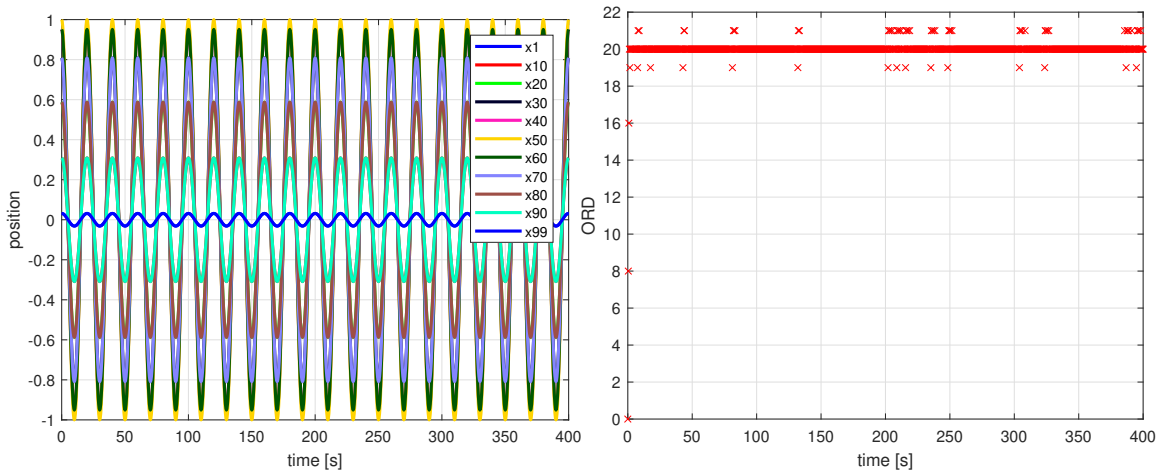
where $\mathbf{A}_{12}$ is the discretiazion matrix, $\mathbf{A}_{21}$ is the identity matrix, and matrices $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ are zero matrices. The sparsity patterns of the matrices $\mathbf{A}$ and $\hat{\mathbf{A}}$ precalculated using (4.46) for $S = 100$ segments are in Figure 6.21.



(a) Matrix $\mathbf{A}$ (b) Matrix $\hat{\mathbf{A}}$ precalculated using (4.46)

Figure 6.21: Sparsity patterns of input matrices, wave equation, three-point central difference formula, $S = 100$.

The numerical solution of the wave equation discretized in the spatial domain by the three-point central finite difference formula and the order of the MTSM for $S = 100$ segments are shown in Figure 6.22.



(a) Numerical solution (b) Order of the MTSM

Figure 6.22: Numerical solution and order of MTSM, wave equation, three-point finite difference approximation, $S = 100$, $h = 4e^{-1}$, $t_{max} = 1000 \cdot h$.

6.7.1 Results overview – three-point central difference formula

The number of integration steps and the average step sizes for each solver is shown in Table 6.15. The MTSM uses a step size approximately 7.8 times larger than the TSRK5DP

solver and approximately 3.2 times larger than the TSRK8VR solver. The average order of the MTSM for the three-point central difference formulas is 19.

solver	# steps	average h
MTSM_PRECALC	10000	$4.00e^{-01}$
MTSM	10000	$4.00e^{-01}$
TSRK5DP	78238	$5.11e^{-02}$
TSRK8VR	32000	$1.26e^{-01}$

Table 6.15: Number of integration steps and average step sizes, wave equation, three-point central difference formula

Tables 6.16 and 6.17 summarize the average efficiency and speedup against the TSRK5DP for each problem size. These results are averages of values for 1–32 compute nodes. Table cells where the parallel efficiency $E \geq 50\%$ are marked in green. The cells of the table showing the speedup ratio with respect to the TSRK5DP solver are also marked in green. The average parallel efficiency is shown in Table 6.16; the MTSM_PRECALC solver offers the parallel efficiency $E \geq 50\%$ for all problem sizes. The MTSM_PRECAL and MTSM solvers are always faster than the TSRK5DP solver, as shown in Table 6.17. Finally, the MTSM_PRECALC is the fastest of all solvers for all problem sizes.

solver	problem size				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	**51.10**	**77.41**	**127.64**	**136.43**	**119.31**
MTSM	15.30	19.77	25.62	33.67	–
TSRK5DP	25.91	29.86	–	–	–
TSRK8VR	29.76	40.24	–	–	–

Table 6.16: Average **efficiency** (6.4) comparison for 1–32 nodes, wave equation, three-point central difference formula.

solver	problem size				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	16.04	20.90	–	–	–
MTSM	1.04	1.63	–	–	–
TSRK8VR	0.97	1.23	–	–	–

Table 6.17: Average **speedup against TSRK5DP** (6.3) comparison for 1–32 nodes, wave equation, three-point central difference formula.

Table 6.18 indicates whether the solver calculates the result for a given problem size. For $S \geq 512000$, the TSRK5DP and TSRK8VR solvers did not provide results because the maximum walltime was exceeded. The MTSM solver did not calculate the results for $S = 2048000$ segments. Finally, the MTSM_PRECALC computes the result for all 32 nodes.

solver	problem size				
	128000	256000	512000	1024000	2048000
MTSM_PRECALC	Y	Y	Y	Y	Y
MTSM	Y	Y	Y	Y	N (6 n)
TSRK5DP	Y	Y	N (27 n)	N (4 n)	–
TSRK8VR	Y	Y	N (21 n)	N (2 n)	–

Table 6.18: Yes/No table, wave equation, three-point central difference formula.

Table 6.19 shows, for each problem size, the number of compute nodes where parallel efficiency $E \geq 50\%$. Notice that the parallel efficiency of the MTSM_PRECALC solver $E \geq 50\%$ for 1–32 compute nodes for problem sizes greater than or equal to $S = 512000$. The efficiency of the MTSM solver stagnates and increases only for $S = 2048000$, but the solver cannot reach parallel efficiency for all 32 compute nodes. For the TSRK5DP and TSRK8VR solvers, the efficiency increases up to $S = 512000$, for $S = 1024000$, it drops to four and two nodes, respectively.

solver	problem size				
	128000	256000	512000	1024000	2048000
MTSM_PRECALC	10	26	32	32	32
MTSM	1	1	1	1	6
TSRK5DP	1	2	16	4	–
TSRK8VR	1	4	21	2	–

Table 6.19: Maximum number of nodes where efficiency $E \geq 50\%$, wave equation, three-point central difference formula.

Table 6.20 shows the maximum number of nodes to use for the calculation with respect to the speedup-cost ratio, i.e., where the speedup-cost tradeoff curve reaches a maximum.

solver	problem size				
	128000	256000	512000	1024000	2048000
MTSM_PRECALC	4	8	32	32	32
MTSM	1	1	16	16	4
TSRK5DP	8	8	16	4	–
TSRK8VR	8	16	16	2	–

Table 6.20: Parallel cost for all problem sizes and solvers, wave equation, three-point central difference formula.

The numerical results for the three-point central difference formula and different problem sizes follow. Detailed results, including results for $S = 128000$ and $S = 512000$, can be found in Appendix C.3.

6.7.2 $S = 64000$, three-point central difference formula

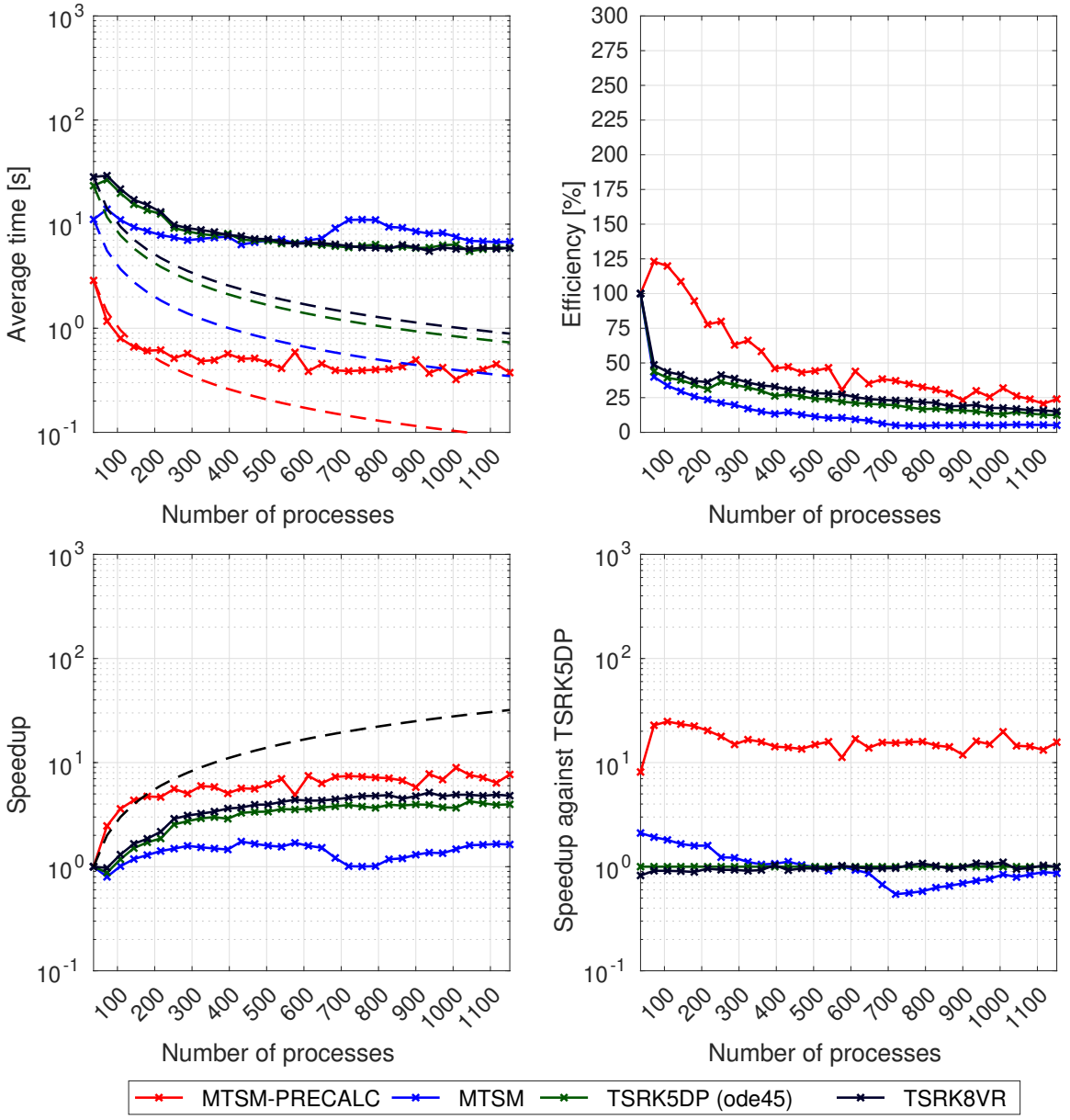


Figure 6.23: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 64000$, wave equation, three-point central difference formula.

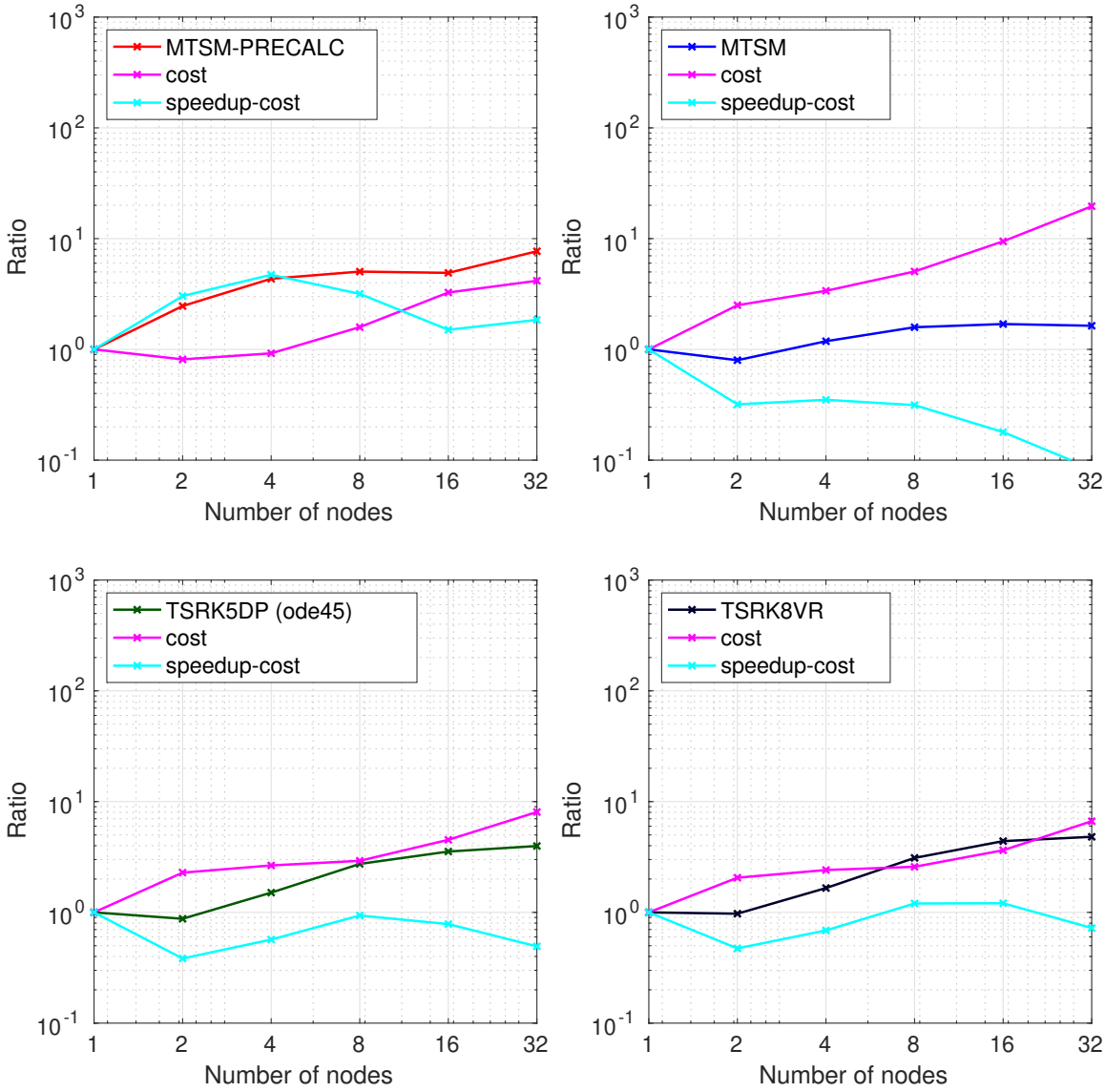


Figure 6.24: Parallel cost ratio and speedup-cost ratio, $S = 64000$, wave equation, three-point central difference formula.

6.7.3 $S = 256000$, three-point central difference formula

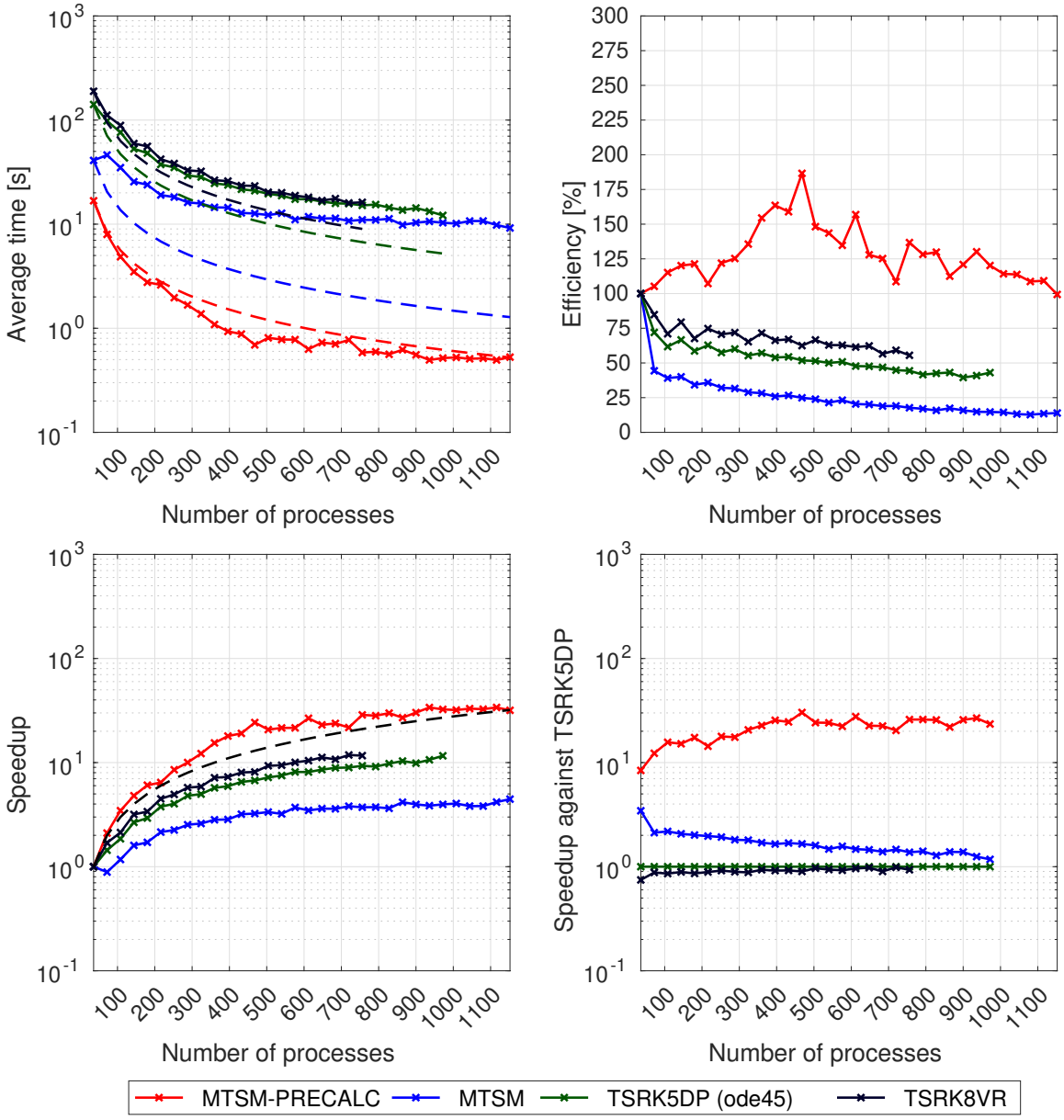


Figure 6.25: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 256000$, wave equation, three-point central difference formula.

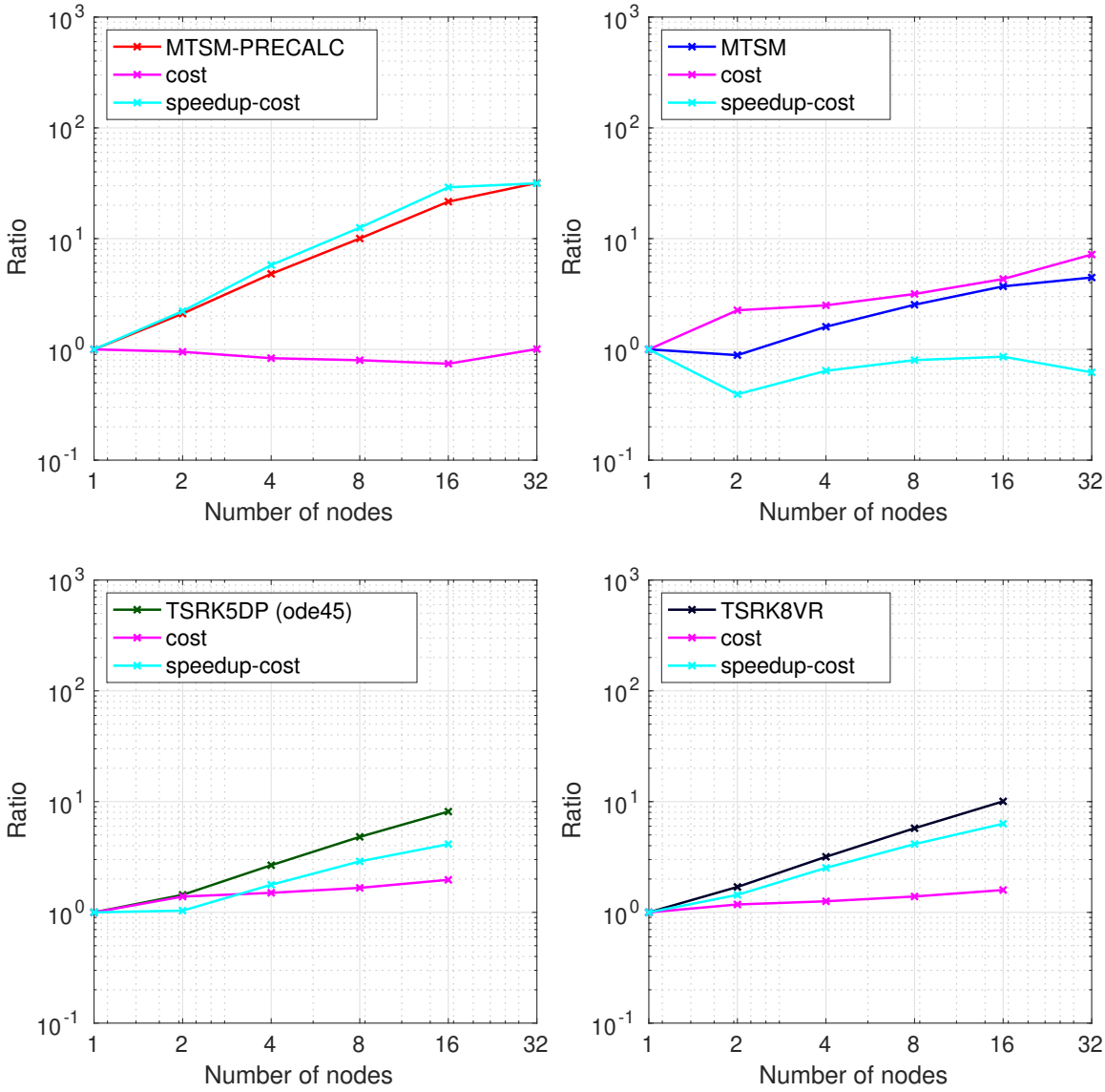


Figure 6.26: Parallel cost ratio and speedup-cost ratio, $S = 256000$, wave equation, three-point central difference formula.

6.7.4 $S = 1024000$, three-point central difference formula

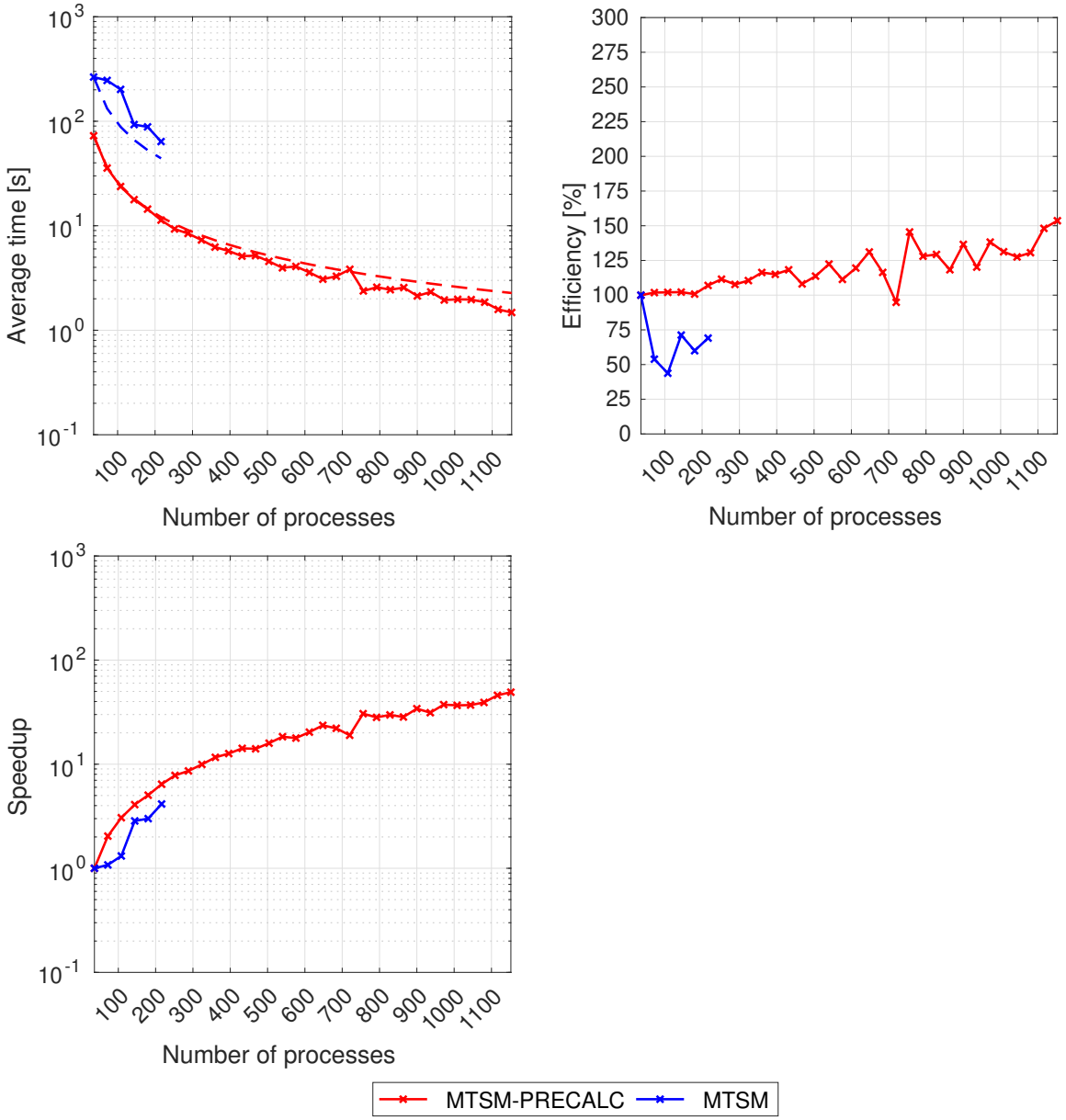


Figure 6.27: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 1024000$, wave equation, three-point central difference formula.

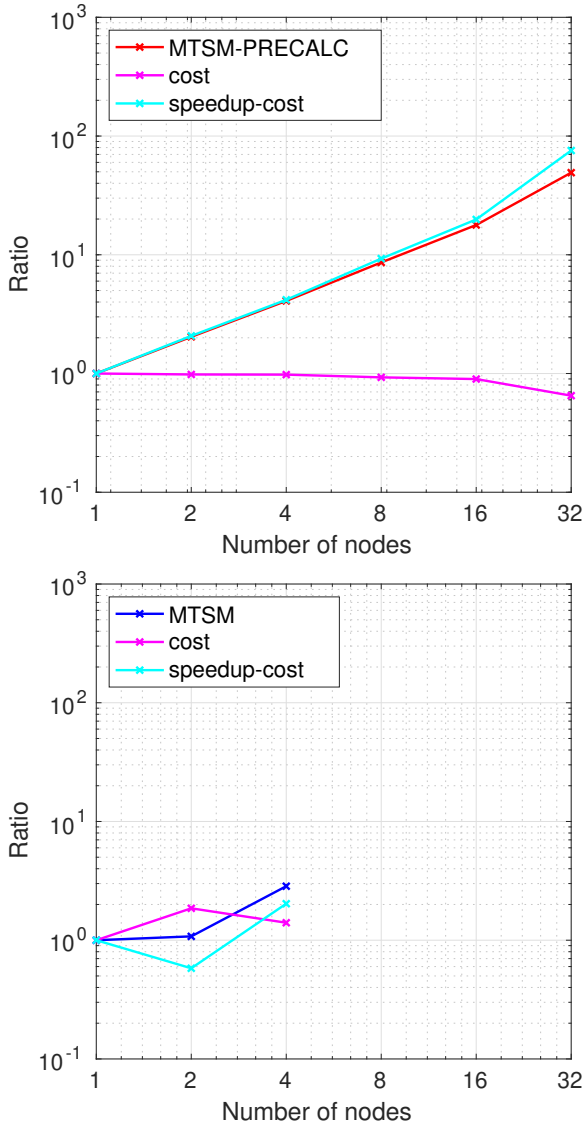
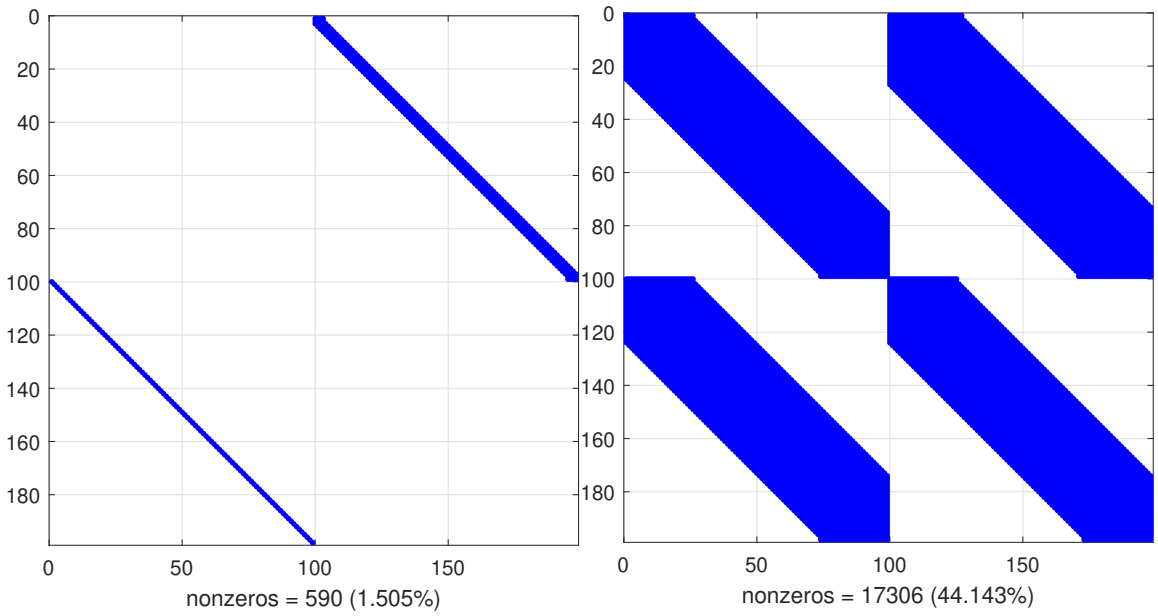


Figure 6.28: Parallel cost ratio and speedup-cost ratio, $S = 1024000$, wave equation, three-point central difference formula.

6.8 Wave equation – five-point central difference formula

The definition of the problem and simulation parameters are the same as for wave equation discretized with the three-point central difference formulas (see Section 6.7). The sparsity patterns of the matrices $\mathbf{A}$ and $\hat{\mathbf{A}}$ precalculated using (4.46) for $S = 100$ segments are in Figure 6.29.



(a) Matrix $\mathbf{A}$ (b) Matrix $\hat{\mathbf{A}}$ precalculated using (4.46)

Figure 6.29: Sparsity patterns of input matrices, wave equation, five-point central difference formula, $S = 100$.

6.8.1 Results overview – five-point central difference formula

Table 6.21 shows the number of integration steps and the average step sizes for each solver. Taylor series-based solvers use a step size approximately 8.4 times larger than the TSRK5DP solver and approximately 3.4 times larger than the TSRK8VR solver. The average order of the MTSM for the five-point central difference formulas is 19.5.

solver	# steps	average h
MTSM_PRECALC	10000	$3.70e^{-01}$
MTSM	10000	$3.70e^{-01}$
TSRK5DP	26030	$4.42e^{-02}$
TSRK8VR	10775	$1.10e^{-01}$

Table 6.21: Number of integration steps and average step sizes, wave equation, five-point central difference formula.

Tables 6.22 and 6.23 summarize the average efficiency, and speedup against the TSRK5DP for each problem size.

solver	problem size				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	**96.82**	**137.98**	**144.77**	**130.29**	**132.39**
MTSM	14.94	19.97	23.10	–	–
TSRK5DP	25.91	35.20	–	–	–
TSRK8VR	29.36	46.10	–	–	–

Table 6.22: Average **efficiency** (6.4) comparison for 1–32 nodes, wave equation, five-point central difference formula.

solver	problem size				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	15.87	18.54	–	–	–
MTSM	1.10	1.49	–	–	–
TSRK8VR	0.96	1.09	–	–	–

Table 6.23: Average **speedup against TSRK5DP** (6.3) comparison for 1–32 nodes, wave equation, five-point central difference formula.

Table 6.24 indicates whether or not the solver calculates the result for a given problem size. For the five-point central difference formula, the TSRK5DP and TSRK8VR solvers did not compute the result for all 32 compute nodes for the problem size $S = 2048000$. The MTSM_PRECALC calculates results for all 1–32 nodes for all problem sizes.

solver	problem size				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	Y	Y	Y	Y	Y
MTSM	Y	Y	Y	N (20 n)	N (5 n)
TSRK5DP	Y	Y	N (19 n)	N (3 n)	–
TSRK8VR	Y	Y	N (15 n)	N (2 n)	–

Table 6.24: Yes/No table, wave equation, five-point central difference formula.

Table 6.25 shows, for each problem size, the number of compute nodes where parallel efficiency $E \geq 50\%$. Notice that the parallel efficiency of the MTSM_PRECALC solver is greater than 50% for 1–32 compute nodes for problem sizes greater than or equal to $S = 256000$. The efficiency of the MTSM solver stagnates for the first three problem sizes and increases for $S = 1024000$ and $S = 2048000$, the solver can reach parallel efficiency for all 32 compute nodes for $S = 2048000$. For the TSRK5DP and TSRK8VR solvers, the efficiency increases up to $S = 1024000$, for $S = 2048000$, it drops to one and three nodes, respectively.

solver	problem size				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	26	32	32	32	32
MTSM	1	1	1	8	5
TSRK5DP	1	4	18	3	–
TSRK8VR	1	8	15	2	–

Table 6.25: Maximum number of nodes where efficiency $E \geq 50\%$, wave equation, five-point central difference formula.

Table 6.26 shows the maximum number of nodes to use for the calculation with respect to the speedup-cost ratio, i.e., where the speedup-cost tradeoff curve reaches a maximum.

solver	problem size				
	128000	**256000**	**512000**	**1024000**	**2048000**
MTSM_PRECALC	16	16	32	32	32
MTSM	1	1	16	16	4
TSRK5DP	8	16	16	2	–
TSRK8VR	8	16	8	2	–

Table 6.26: Parallel cost for all problem sizes and solvers, wave equation, five-point central difference formula.

The numerical results for the five-point central difference formula and different number of ODEs follows. Detailed results, including results for $S = 128000$ and $S = 512000$, can be found in Appendix C.4.

6.8.2 $S = 64000$, five-point central difference formula

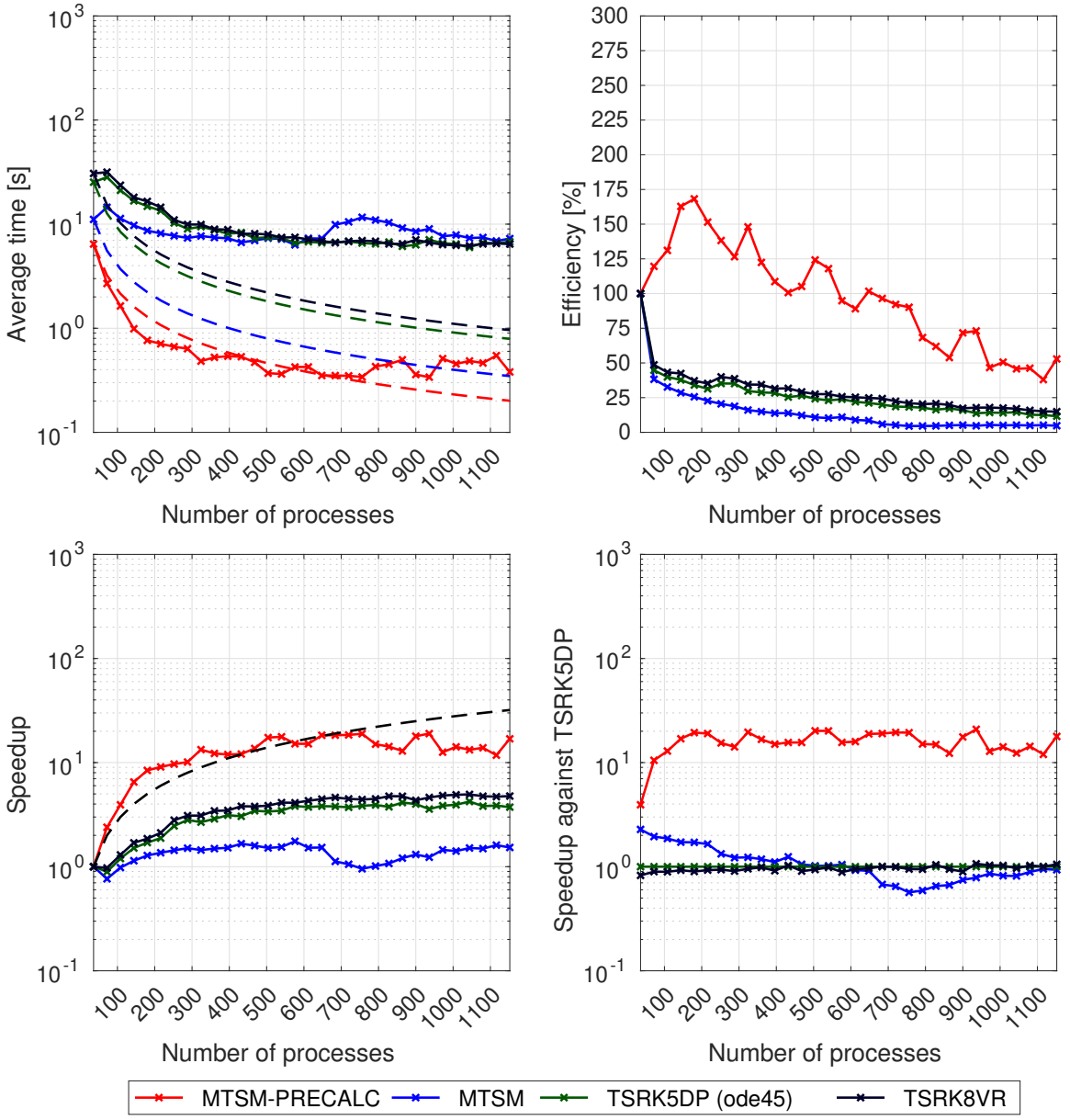


Figure 6.30: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 64000$, wave equation, five-point central difference formula.

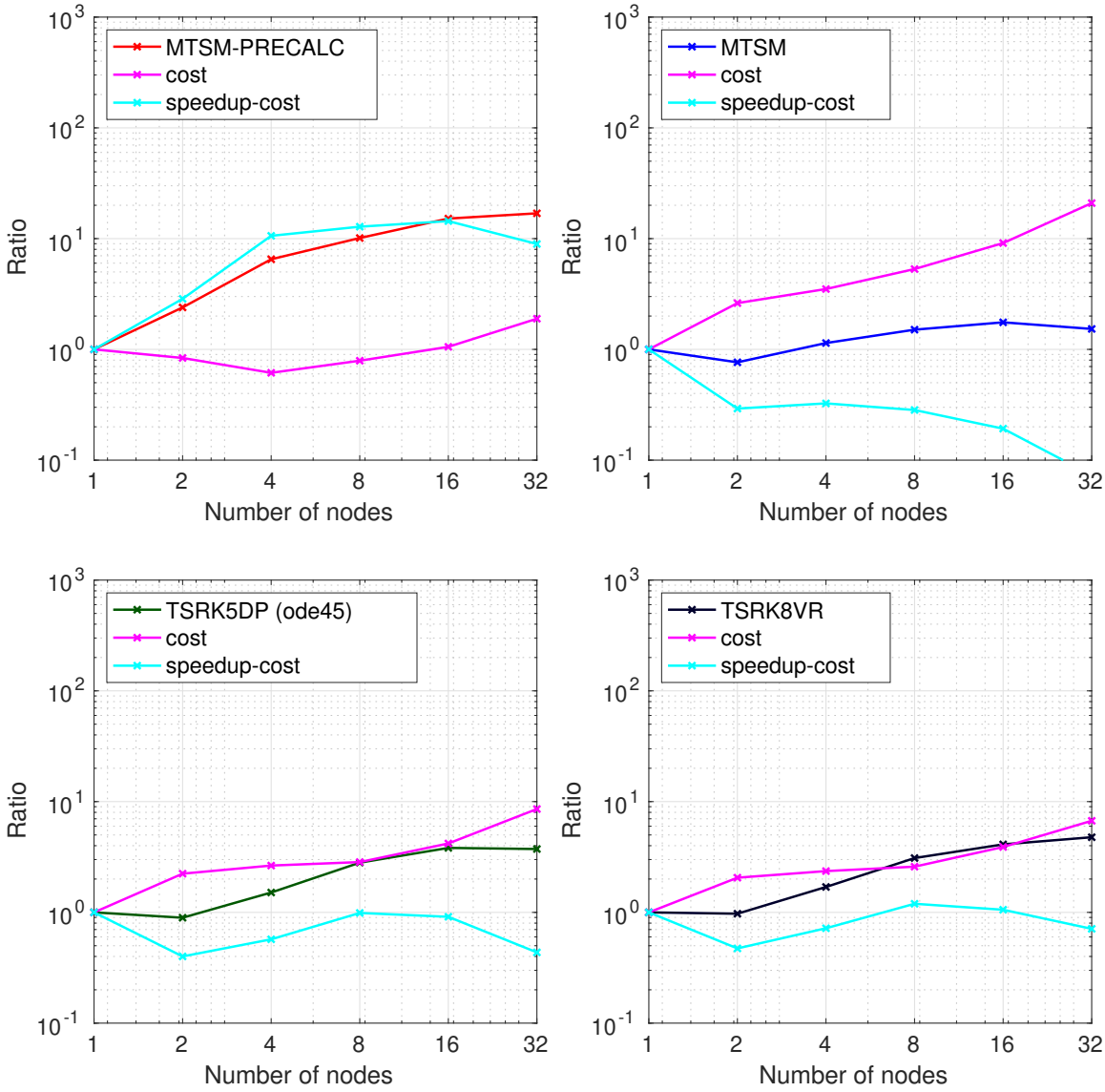


Figure 6.31: Parallel cost ratio and speedup-cost ratio, $S = 64000$, wave equation, five-point central difference formula.

6.8.3 $S = 256000$, five-point central difference formula

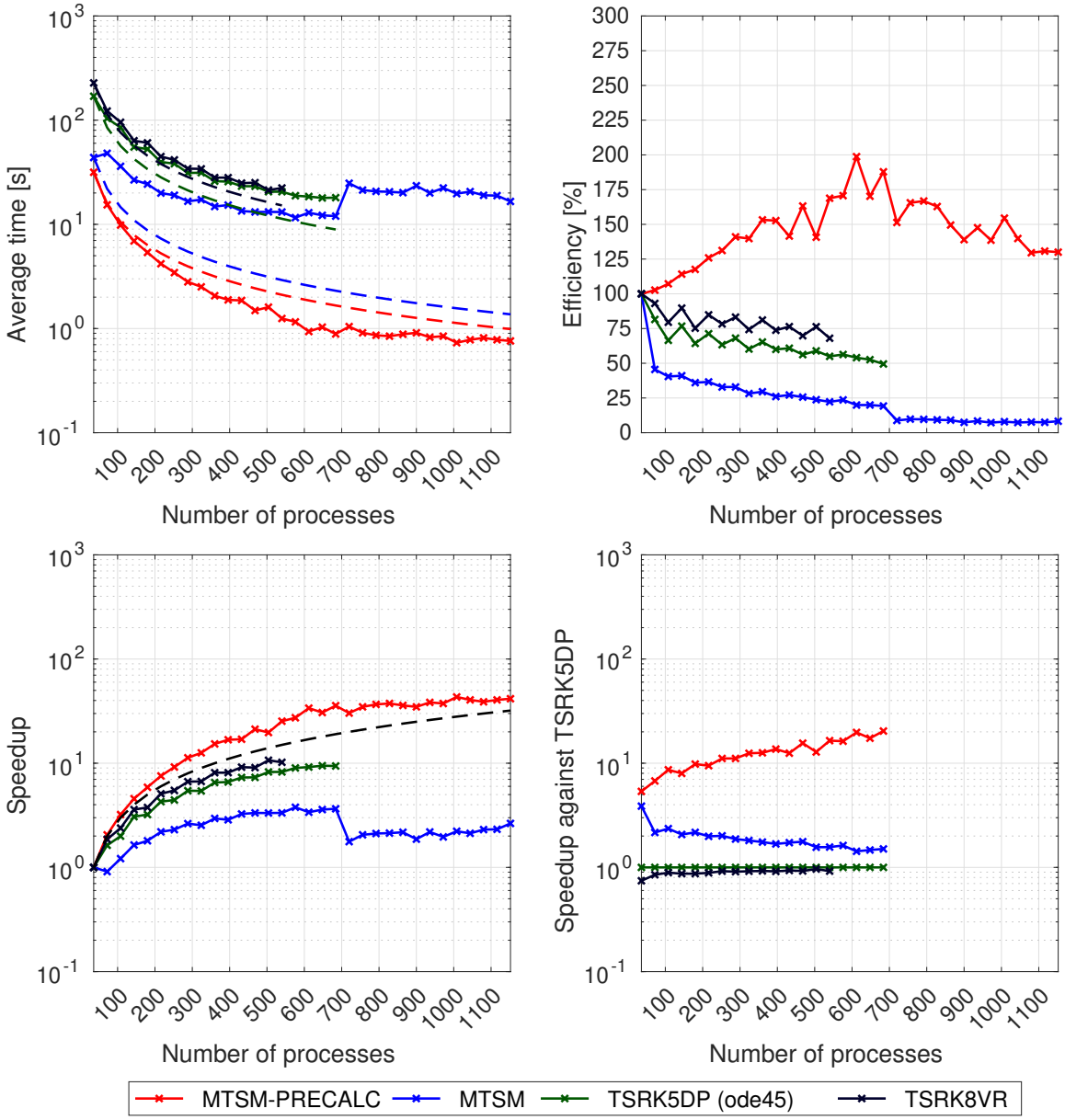


Figure 6.32: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 256000$, wave equation, five-point central difference formula.

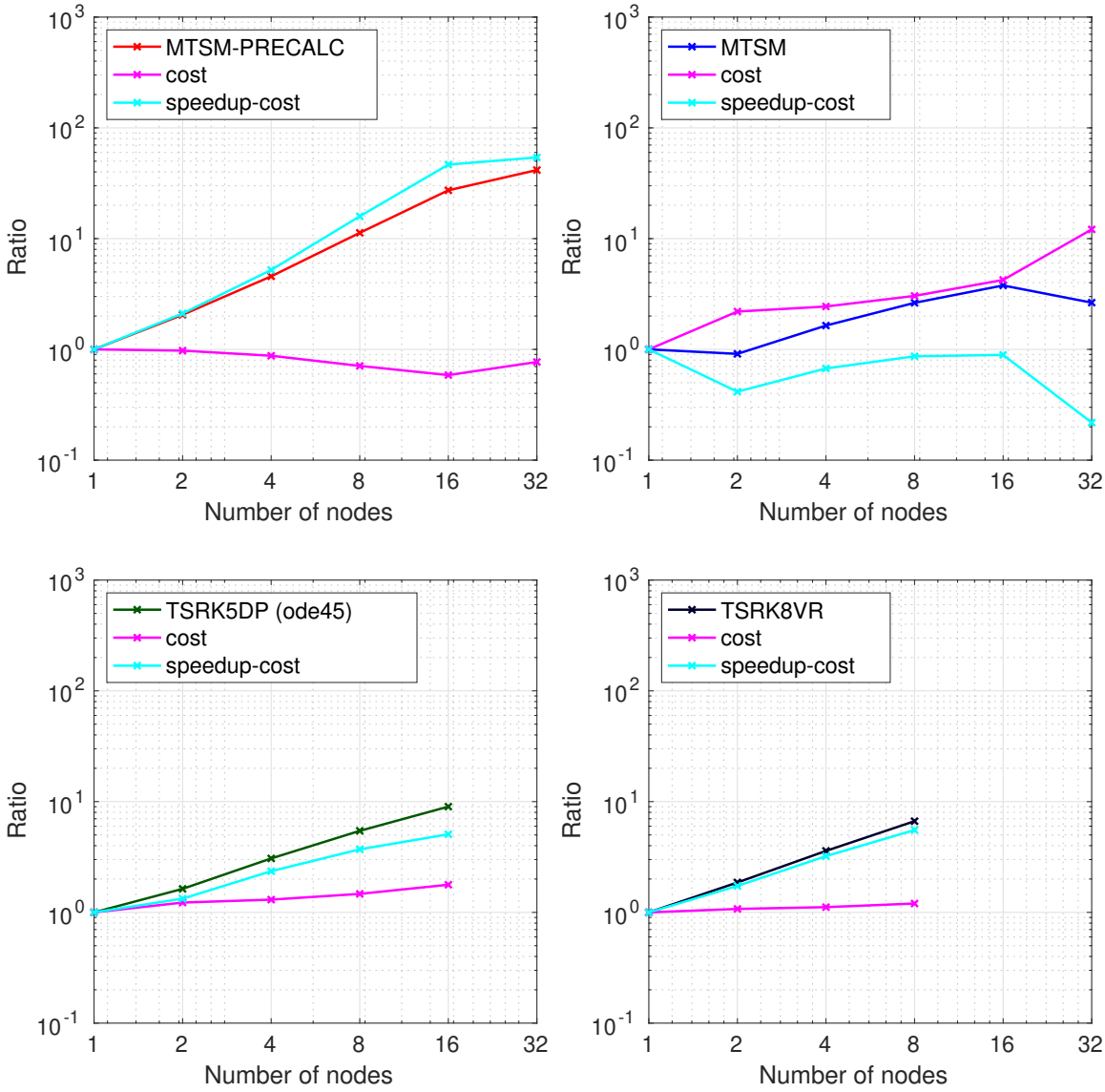


Figure 6.33: Parallel cost ratio and speedup-cost ratio, $S = 256000$, wave equation, five-point central difference formula.

6.8.4 $S = 1024000$, five-point central difference formula

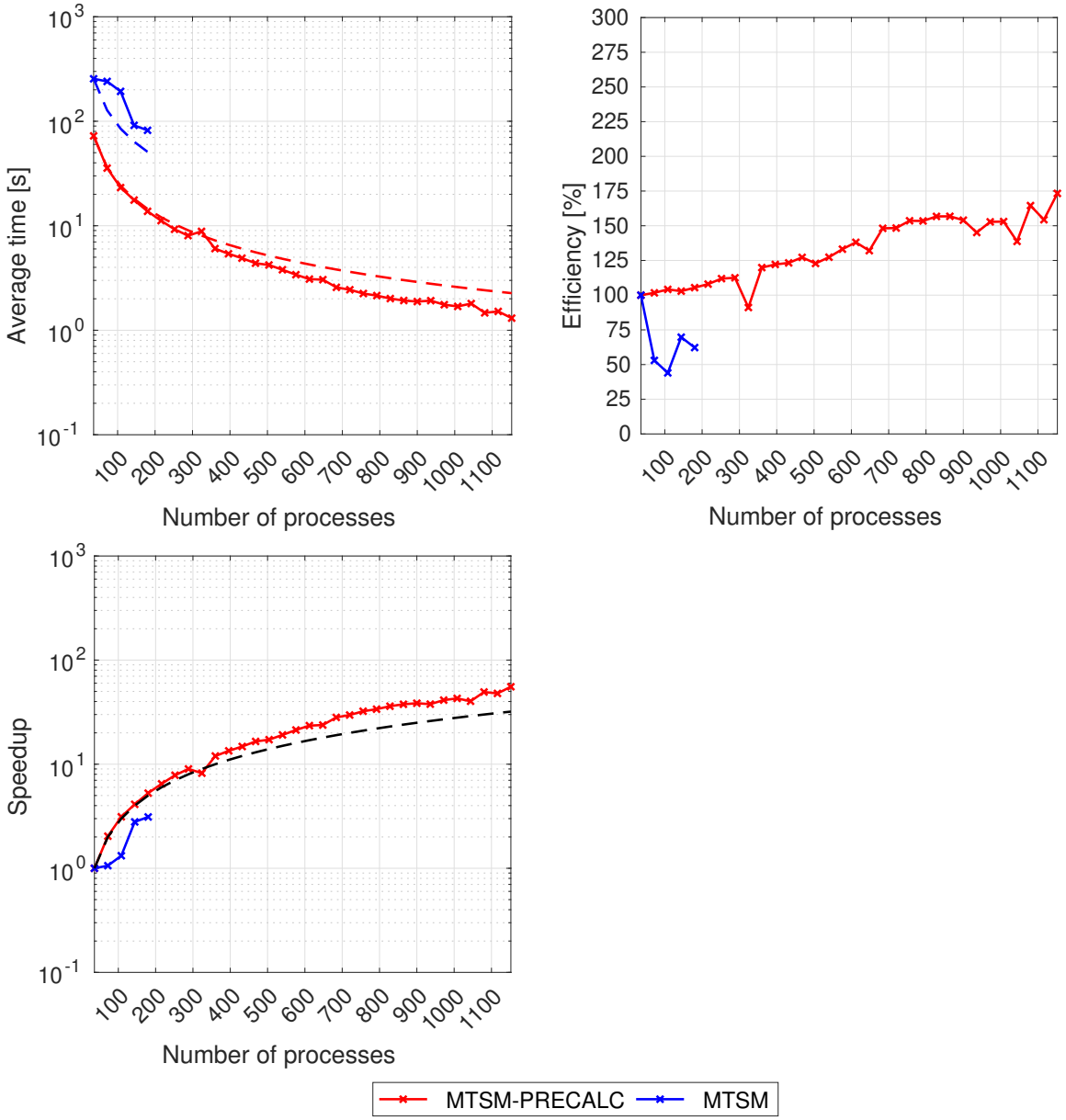


Figure 6.34: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 1024000$, wave equation, five-point central difference formula.

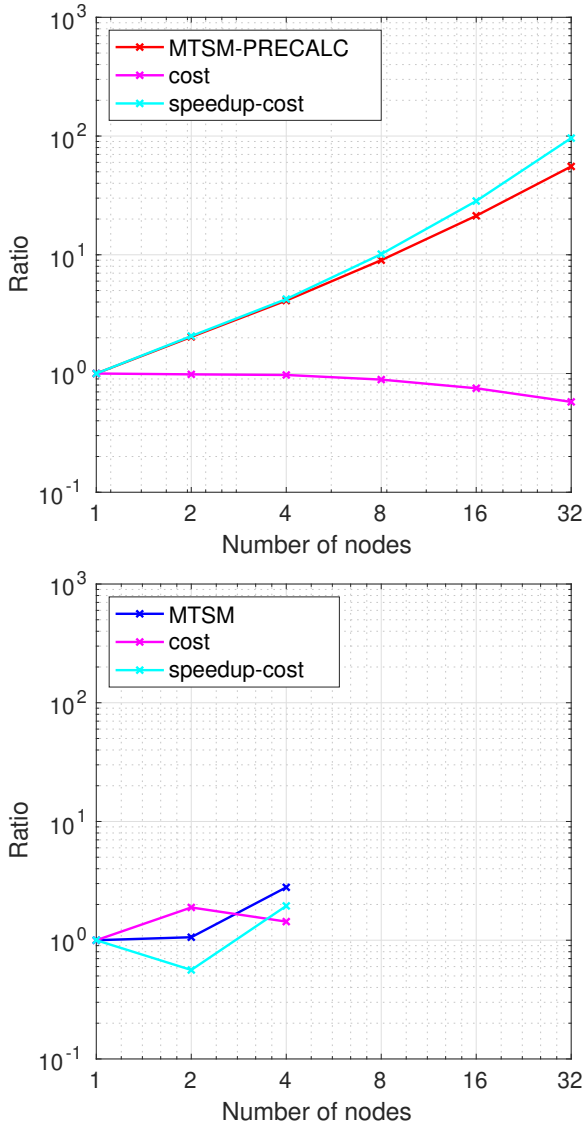


Figure 6.35: Parallel cost ratio and speedup-cost ratio, $S = 1024000$, wave equation, five-point central difference formula.

6.9 Telegraph equation

The telegraph equation is a second-order PDE that describes a telegraph line – a long wire that serves as a transmission medium for a signal. The PDE can describe the behavior of the signal. However, this description does not contain any specific information about the conditions of the wire, which is complicated. The numerical model of the telegraph line presented in this section consists only of ODEs, it is relatively simple and easy to configure, and the results match the real output precisely [pp10, pp26, pp28].

Equations $u = u(x,t)$ and $i = i(x,t)$ express the voltage and current changes along the telegraph line, where x is the distance from the beginning of the line and t is the time. The voltage and current in the distance $x + \mathrm{d}x$ can be expressed using the Taylor series with the second and higher derivatives omitted.

$$u(x+\mathrm{d}x) = u(x,t) + \frac{\partial u}{\partial x}\mathrm{d}x\,, \tag{6.21}$$

$$i(x+\mathrm{d}x) = i(x,t) + \frac{\partial i}{\partial x}\mathrm{d}x\,. \tag{6.22}$$

Basic Line Equations (6.23), (6.24) describe the change of voltage and current on the line

$$-\frac{\partial u}{\partial x} = Ri + L\frac{\partial i}{\partial t}\,, \tag{6.23}$$

$$-\frac{\partial i}{\partial x} = Gu + C\frac{\partial u}{\partial t}\,, \tag{6.24}$$

where constants R, G, L, C are parameters of the line: R [Ω/m] the resistance of the wire, G [S/m] the conductance between wires, L [H/m] inductance of the wire (e.g., due to the magnetic field around the wires) and C [F/m] capacitance between two wires, respectively.

6.9.1 Lossy telegraph line

The lossy telegraph line can be described by partial differential equations for voltage (6.25) and current (6.26):

$$L\cdot C\frac{\partial^2 u(x,t)}{\partial t^2} + (L\cdot G + C\cdot R)\frac{\partial u(x,t)}{\partial t} + R\cdot G\cdot u(x,t) - \frac{\partial^2 u(x,t)}{\partial x^2} = 0\,, \tag{6.25}$$

$$L\cdot C\frac{\partial^2 i(x,t)}{\partial t^2} + (L\cdot G + C\cdot R)\frac{\partial i(x,t)}{\partial t} + R\cdot G\cdot i(x,t) - \frac{\partial^2 i(x,t)}{\partial x^2} = 0\,. \tag{6.26}$$

Using (6.23) and (6.24) it is possible to construct a model of the segment (see Figure 6.36).

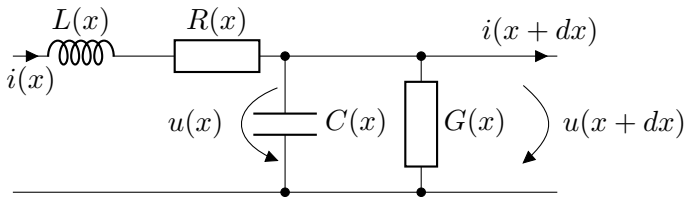


Figure 6.36: Lossy model of the telegraph equation – a segment of the wire.

6.9.2 Lossy telegraph equation model

The model of the lossy telegraph line (Figure 6.37) consists of an infinite number of connected segments (Figure 6.36). Let us denote the number of segments of the telegraph line as S.

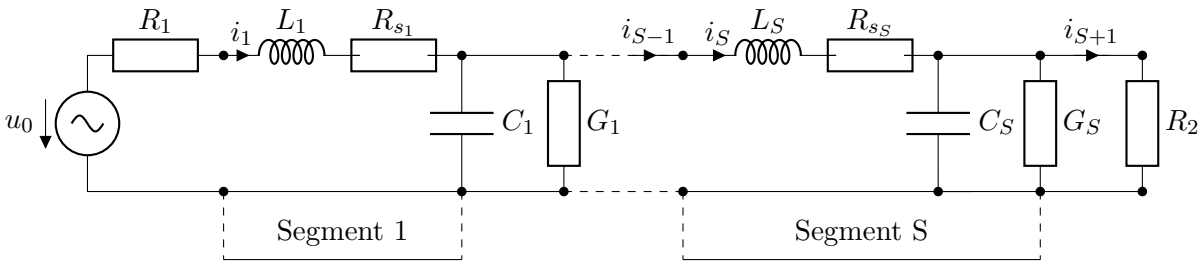


Figure 6.37: Lossy model of the telegraph equation – series of S segments.

The equations describing the model are below. For the first segment,

$$\begin{aligned} u'_{C_1} &= \frac{1}{C_1}(i_1 - G_1 \cdot u_{C_1} - i_2), \\ i'_1 &= \frac{1}{L_1}(u_0 - u_{C_1} - R_1 \cdot i_1 - R_{s_1} \cdot i_1)\,, \end{aligned} \tag{6.27}$$

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 flowing through the first inductor. The resistor R_1 represents an input load of the transmission line. The resistors $R_{s1}, \ldots, R_{sS}$ denote the resistances of the wire, and $G_1, \ldots, G_S$ denote the conductances of the wire. For the next segments

$$\begin{aligned} u'_{C_k} &= \frac{1}{C_k}(i_k - G_k \cdot u_{C_k} - i_{k+1}), \\ i'_k &= \frac{1}{L_k}(u_{C_{k-1}} - u_{C_k} - R_{s_S} \cdot i_k)\,, \end{aligned} \tag{6.28}$$

where $k \in \langle 2, S \rangle$. The last segment of the line ends with an output load, simulated by the resistor R_2

$$i_{S+1} = \frac{1}{R_2} u_{C_S}\,. \tag{6.29}$$

Note that all differential equations have initial conditions equal to zero.

6.9.3 Lossless telegraph line

The model in Figure 6.36 can be simplified by omitting the terms $R(x) = G(x) = 0$. The simplified model is in Figure 6.38. The line then becomes lossless.

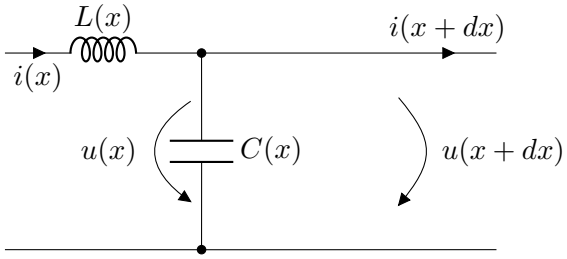


Figure 6.38: Lossy model of the line – a segment of the wire.

Based on the simplified model, partial differential equations (6.30) and (6.31) for voltage and current can be derived using (6.25) and (6.26):

$$L \cdot C \frac{\partial^2 u(x,t)}{\partial t^2} - \frac{\partial^2 u(x,t)}{\partial x^2} = 0\,, \tag{6.30}$$

$$L \cdot C \frac{\partial^2 i(x,t)}{\partial t^2} - \frac{\partial^2 i(x,t)}{\partial x^2} = 0\,. \tag{6.31}$$

6.9.4 Lossless telegraph equation model

The model of the lossless telegraph line (Figure 6.39) consists of an infinite number of connected segments (Figure 6.38). The number of segments of the telegraph line is denoted as S.

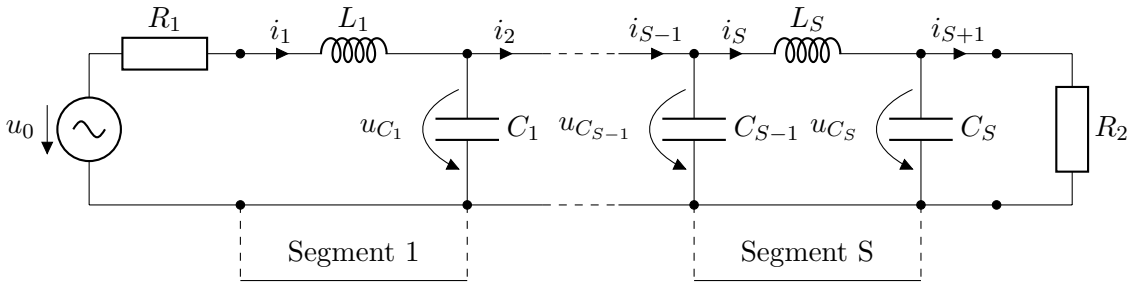


Figure 6.39: Lossless model of the line – series of S segments.

The model can be described using the equations below. For the first segment,

$$\begin{aligned} u'_{C_1} &= \frac{1}{C_1}(i_1 - i_2)\,, \\ i'_1 &= \frac{1}{L_1}(u_0 - u_{C_1} - R_1 \cdot i_1)\,, \end{aligned} \tag{6.32}$$

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. The resistor R_1 represents the input load of the transmission line. Equations for the following segments are very similar to each other. For the next segments

$$\begin{aligned} u'_{C_k} &= \frac{1}{C_k}(i_k - i_{k+1})\,, \\ i'_k &= \frac{1}{L_k}(u_{C_{k-1}} - u_{C_k})\,, \end{aligned} \tag{6.33}$$

where $k \in \langle 2, S \rangle$. The last segment of the line ends with an output load, simulated by the resistor R_2

$$i_{S+1} = \frac{1}{R_2} u_{C_S}\,. \tag{6.34}$$

Note that all differential equations have initial conditions equal to zero.

The model can be represented as the linear system of ODEs in the matrix-vector notation,

$$\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{b}, \quad \mathbf{y}(0) = \mathbf{y}_0\,, \tag{6.35}$$

where $\mathbf{A}$ is the sparse matrix, $\mathbf{y}$ is the vector of voltages and currents, and $\mathbf{b}$ is the right-hand-side vector. The block structure of the matrix $\mathbf{A}$, vectors $\mathbf{y}$ and $\mathbf{b}$ is

$$\mathbf{A} = \left(\begin{array}{c|c} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \hline \mathbf{A}_{21} & \mathbf{A}_{22} \end{array}\right), \quad \mathbf{y} = \begin{pmatrix} u_{C_1} \\ \vdots \\ u_{C_S} \\ i_1 \\ \vdots \\ i_S \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \frac{u_0}{L_1} \\ \vdots \\ 0 \end{pmatrix}, \tag{6.36}$$

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

$$\mathbf{A}_{11} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \frac{-1}{R_2 C_S} \end{pmatrix}$$

$$\mathbf{A}_{12} = \begin{pmatrix} \frac{1}{C_1} & \frac{-1}{C_1} & 0 & \cdots & \cdots & 0 \\ 0 & \frac{1}{C_2} & \frac{-1}{C_2} & 0 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & \frac{1}{C_S} \end{pmatrix}$$

$$\mathbf{A}_{21} = \begin{pmatrix} \frac{-1}{L_1} & 0 & 0 & \cdots & \cdots & 0 \\ \frac{1}{L_2} & \frac{-1}{L_2} & 0 & 0 & \cdots & \vdots \\ 0 & \frac{1}{L_3} & \frac{-1}{L_3} & 0 & \cdots & \vdots \\ 0 & \cdots & \cdots & \cdots & \frac{1}{L_S} & \frac{-1}{L_S} \end{pmatrix}$$

$$\mathbf{A}_{22} = \begin{pmatrix} \frac{-R_1}{L_1} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}.$$

The input voltage u_0 can generally be a constant (DC) or harmonic (AC) signal. In the case of a DC circuit, the input voltage u_0 is hidden in constant right-hand side $\mathbf{b}$ see (6.36). In the case of an AC circuit, the input voltage $u_0 = U_0 \sin(\omega t)$ can be computed using an auxiliary system of coupled linear ODEs

$$\begin{aligned} u_0' &= \omega x\,, & u_0(0) &= 0 \\ x' &= -\omega u_0\,, & x(0) &= U_0\,. \end{aligned} \tag{6.37}$$

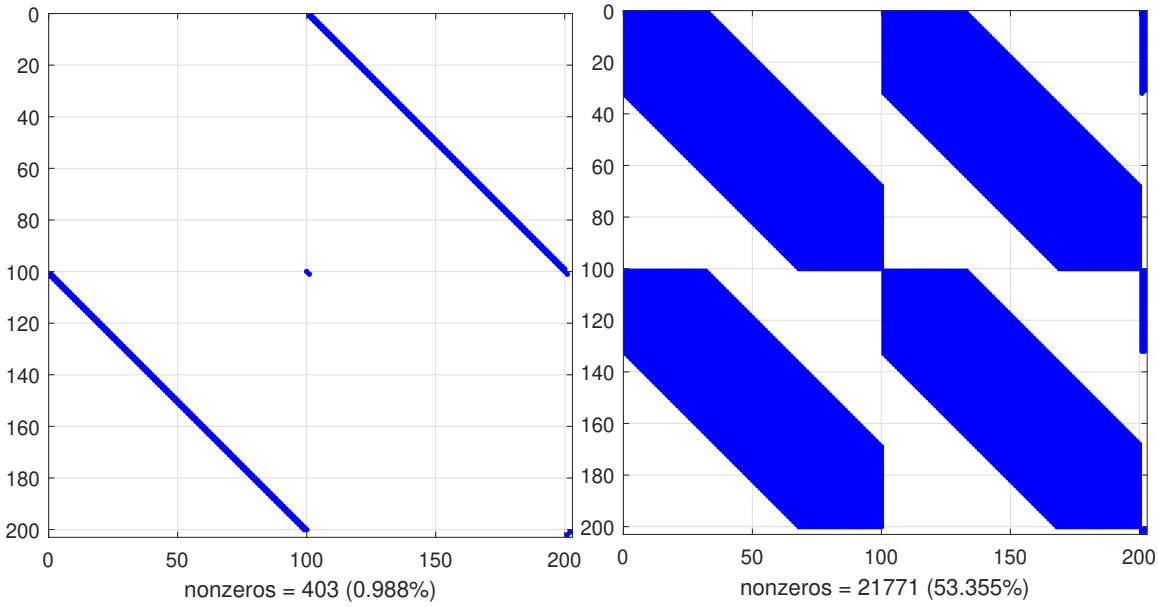
For simulation experiments, the parameters are the same: $C_1 = C_2 = \cdots = C_S = 1\,\mathrm{pF}$, $L_1 = L_2 = \cdots = L_S = 10\,\mathrm{nH}$ (homogeneous lossy telegraph line). The behavior of the transmission on the line is based on the values of the input (R_1) and the output (R_2) loads. If the condition

$$R_1 = R_2 = \sqrt{\frac{L}{C}} \tag{6.38}$$

is satisfied, the line is adjusted. For 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 is known $t_{LC} = \sqrt{LC}$. Then the total delay of the input signal can be computed as $t_{delay} = S\,t_{LC}$. The simulation time for all experiments was set $t_{max} = 2\,t_{delay}$.

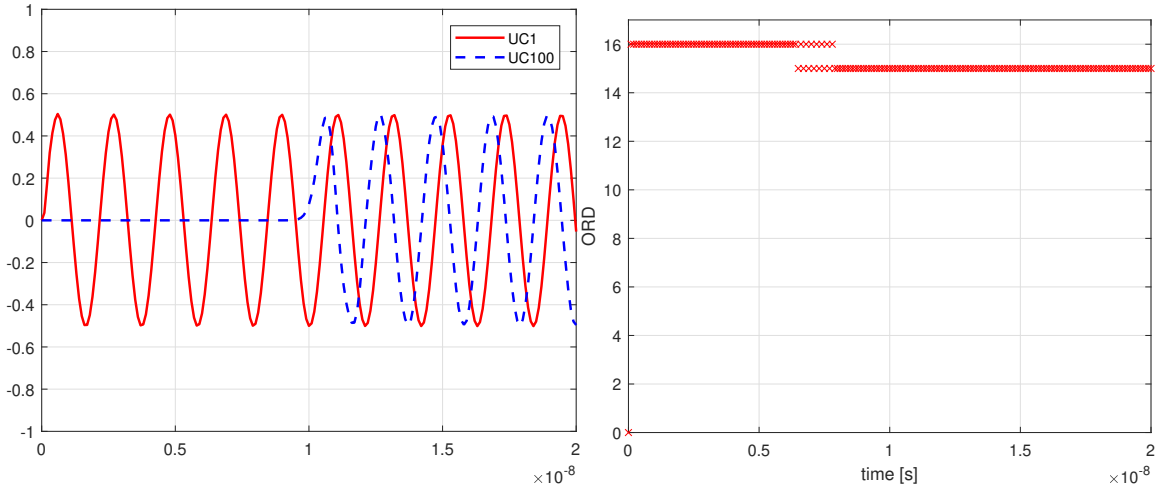
Tolerances for all ode solvers are $RelTol = 1e^{-10}$ and $AbsTol = 1e^{-10}$, the tolerance for MTSM solvers is $tol = 1e^{-10}$, integration step $h = \sqrt{LC}$, $maxORD = 64$.

Figure 6.40 shows the sparsity pattern of the matrix $\mathbf{A}$ and the precalculated matrix $\hat{\mathbf{A}}$. The behavior of the signal for the harmonic input on the line consisting of 100 segments ($S = 100$) and the order of the MTSM are depicted in Figure 6.41.



(a) Matrix $\mathbf{A}$ with auxiliary system of ODE (b) Matrix $\hat{\mathbf{A}}$ precalculated using (4.46)

Figure 6.40: Sparsity patterns of input matrices, telegraph equation, $S = 100$.



(a) Adjusted line – output delayed (b) Order of the MTSM

Figure 6.41: Numerical solution and order of MTSM, telegraph equation, $S = 100$, $h = 1e^{-10}$, $t_{max} = 2e^{-8}$.

6.9.5 Results overview

Table 6.27 shows the number of integration steps and average step sizes for each solver. The MTSM uses a step size approximately 5 times larger than the TSRK5DP solver and approximately 1.5 times larger than the TSRK8VR solver. The average order of the MTSM is 17.

solver	**# steps**	**average h**
MTSM_PRECALC	10000	$1.00e^{-10}$
MTSM	10000	$1.00e^{-10}$
TSRK5DP	49720	$2.01e^{-11}$
TSRK8VR	15141	$6.60e^{-11}$

Table 6.27: Number of integration steps and average step sizes, telegraph equation.

Tables 6.28 and 6.29 summarize the average efficiency and speedup against the TSRK5DP for each problem size. The average parallel efficiency is shown in Table 6.28. All solvers, except the MTSM (for $S = 1024000$), offer an efficiency greater than 50% for all problem sizes. All solvers are always faster than the TSRK5DP solver, as shown in Table 6.29. Finally, the MTSM_PRECALC is the fastest of all solvers for all problem sizes.

solver	**problem size**	
	1024000	**2048000**
MTSM_PRECALC	**115.21**	62.15
MTSM	38.59	60.43
TSRK5DP	81.06	**109.50**
TSRK8VR	91.98	101.77

Table 6.28: Average **efficiency** (6.4) comparison for 1–32 nodes, telegraph equation.

solver	**problem size**	
	1024000	**2048000**
MTSM_PRECALC	4.41	3.57
MTSM	1.26	1.30
TSRK8VR	1.61	1.32

Table 6.29: Average **speedup against TSRK5DP** (6.3) comparison for 1–32 nodes, telegraph equation.

Table 6.30 indicates whether or not the solver calculates the result for a given problem size. All solvers computed results for all 32 compute nodes for all problem sizes.

solver	problem size	
	1024000	2048000
MTSM_PRECALC	Y	Y
MTSM	Y	Y
TSRK5DP	Y	Y
TSRK8VR	Y	Y

Table 6.30: Yes/No table, telegraph equation.

Table 6.31 shows, for each problem size, the number of compute nodes where an efficiency $E \geq 50\%$ was achieved. Notice that the parallel efficiency of the MTSM_PRECALC, TSRK5DP and TSRK8VR solvers is greater than 50% for 1–32 compute nodes for all selected problem sizes. The efficiency of the MTSM solver increases, but cannot reach parallel efficiency $E \geq 50\%$ for all 32 compute nodes.

solver	problem size	
	1024000	2048000
MTSM_PRECALC	32	32
MTSM	6	25
TSRK5DP	32	32
TSRK8VR	32	32

Table 6.31: Maximum number of nodes where efficiency $E \geq 50\%$, telegraph equation.

Table 6.32 shows the maximum number of nodes to use for the calculation with respect to the speedup-cost ratio, i.e., where the speedup-cost tradeoff curve reaches a maximum.

solver	problem size	
	1024000	2048000
MTSM_PRECALC	32	32
MTSM	32	32
TSRK5DP	16	32
TSRK8VR	16	16

Table 6.32: Parallel cost for all problem sizes and solvers, telegraph equation.

The numerical results for different numbers of ODEs follow. Detailed results can be found in Appendix C.5.

6.9.6 $S = 512000$

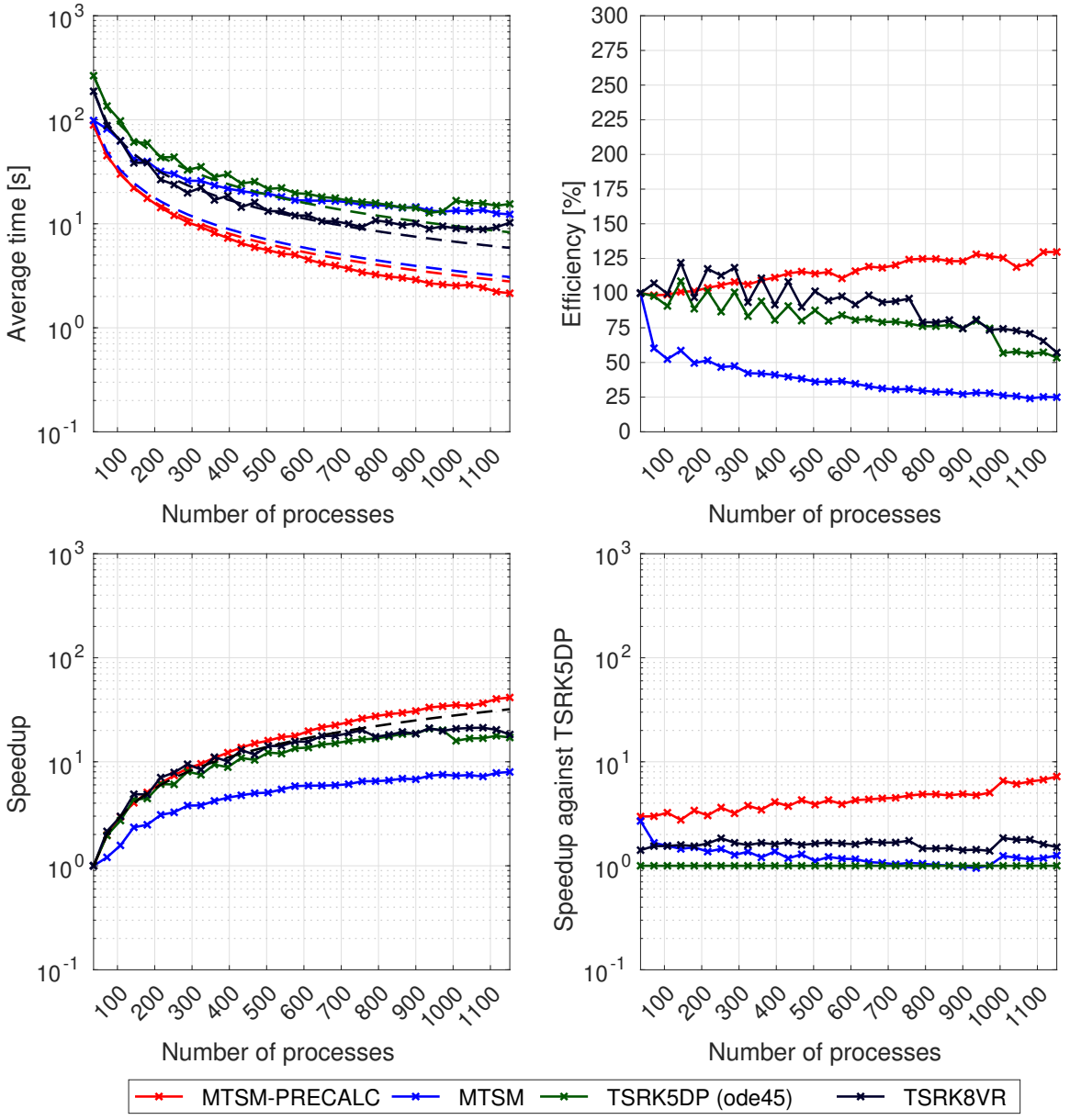


Figure 6.42: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 512000$, telegraph equation.

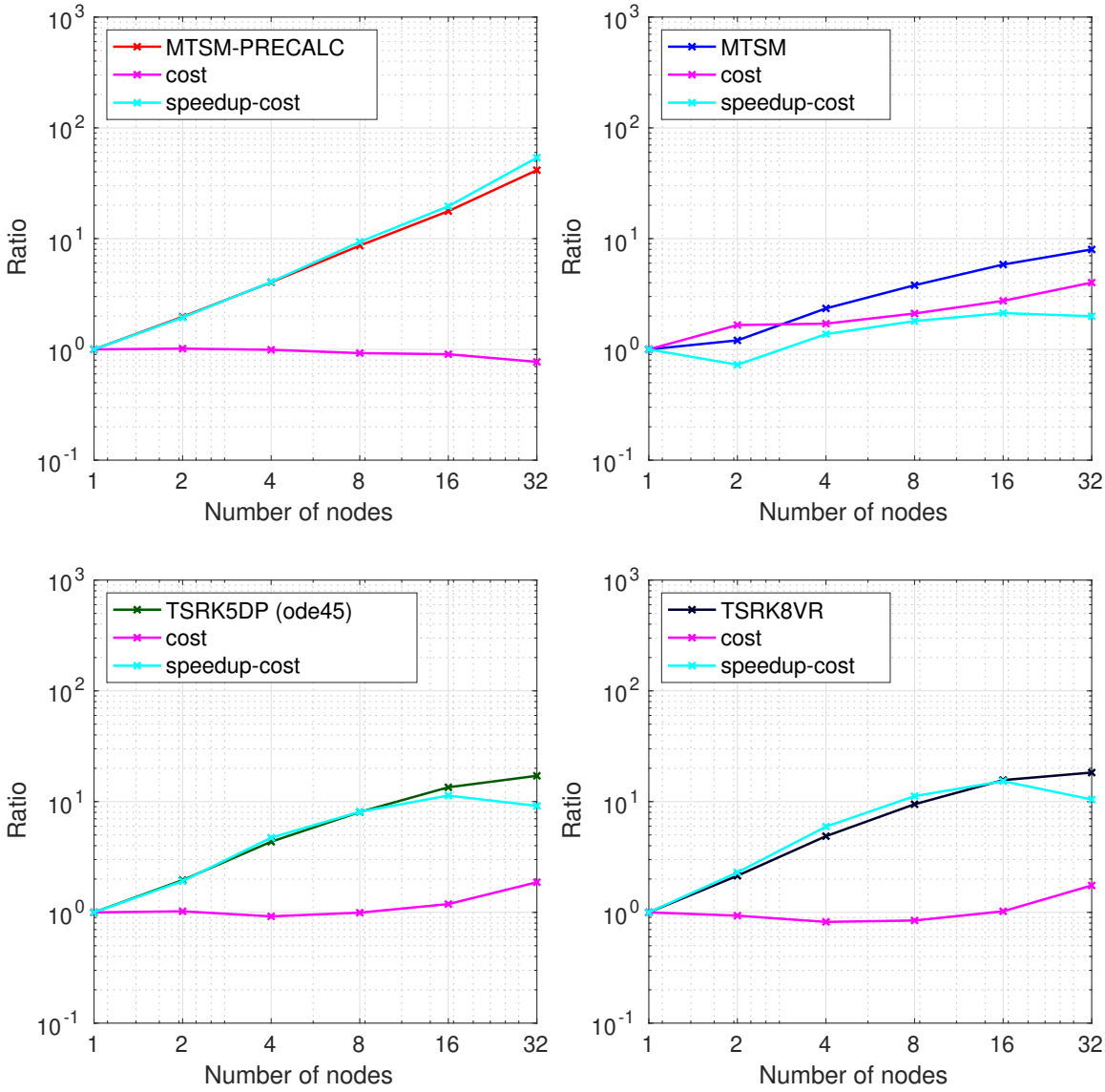


Figure 6.43: Parallel cost ratio and speedup-cost ratio, $S = 512000$, telegraph equation.

6.9.7 $S = 1024000$

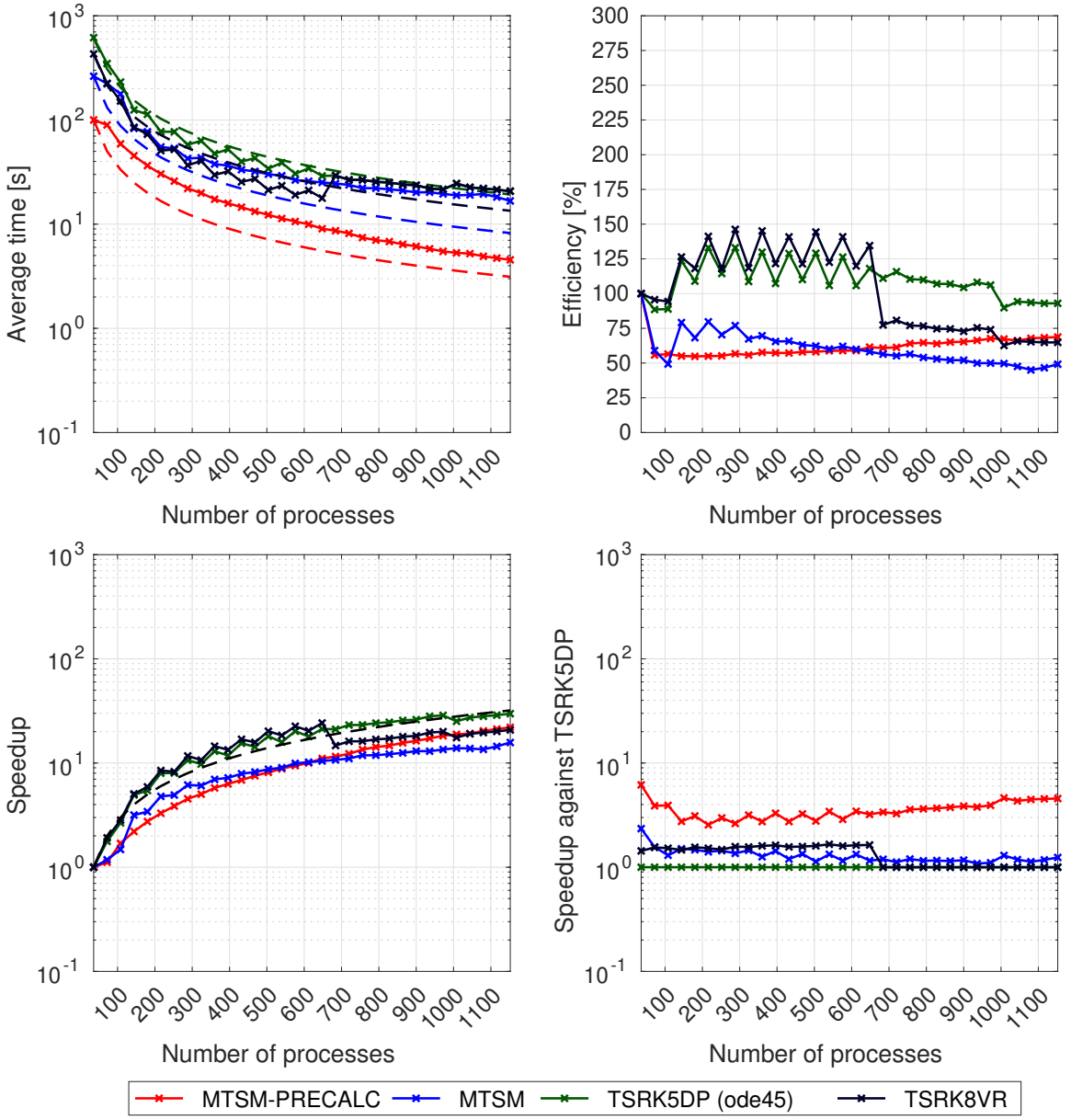


Figure 6.44: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 1024000$, telegraph equation.

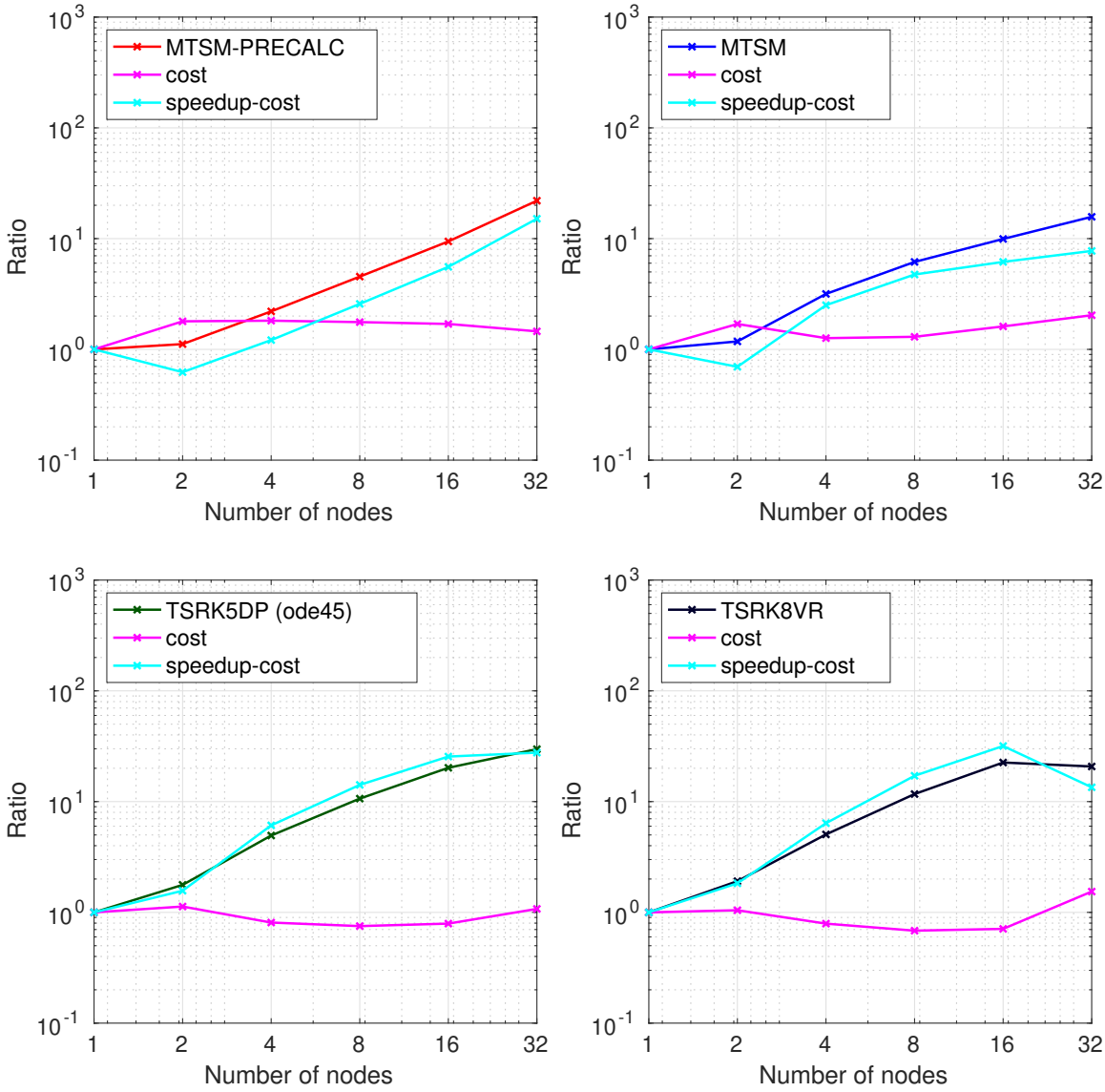


Figure 6.45: Parallel cost ratio and speedup-cost ratio, $S = 1024000$, telegraph equation.

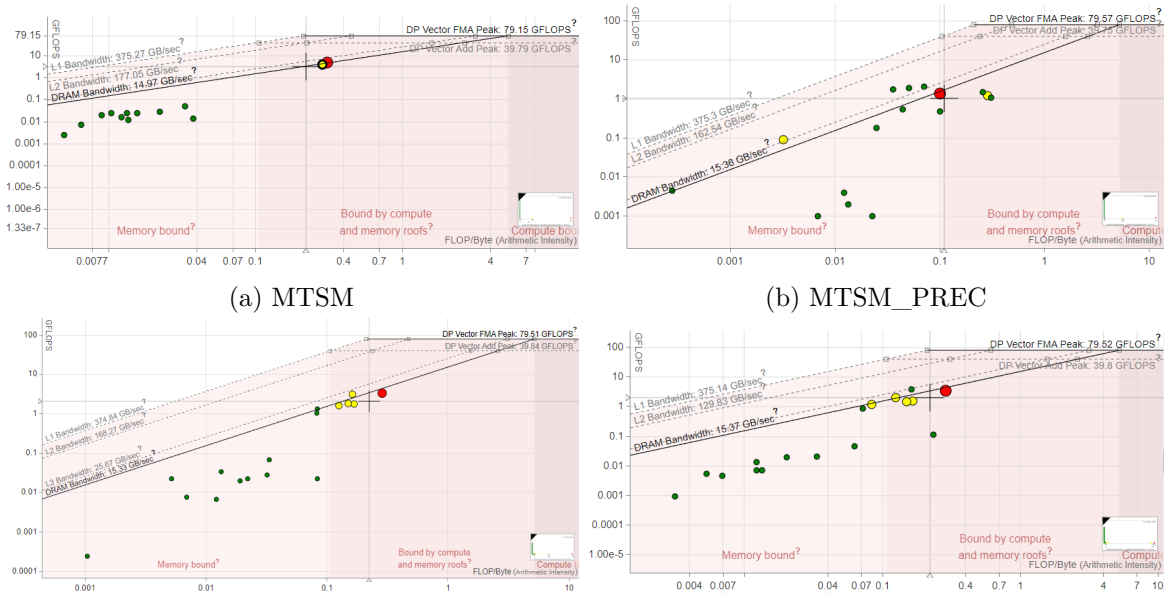
6.10 Parallel performance analysis

The roofline model is based on the simplified model of the CPU and hides most of the architecture-specific complexity and provides insight into architectural bottlenecks and potential application optimizations. The roofline model was first proposed by S. Williams [174] and then expanded to provide additional insight by addressing all levels of memory/-cache hierarchy [85]. The model shows the performance of the executed code against its arithmetic (operational) intensity $AI = F/V$ [FLOP/B], where F is the number of floating point operations, and V is the size of data read or written to memory.

The hardware is viewed as two units. The execution unit that operates at *peak performance* P_{peak} measured in [FLOPS], which represents the number of floating-point operations per second, and the data unit, which can store or deliver data at *maximum bandwidth* b_s measured in [B/s]. Peak performance and maximum bandwidth are two performance bottlenecks.

In the graphical representation of the roofline model, the x-axis represents the arithmetic (operational) intensity [FLOP/B], and it is a linear function with a slope of b_s. The y-axis shows the performance [FLOPS]. Both axes have a logarithmic scale.

Fig. 6.46 shows the roofline models of each solver for the heat equation described by $S = 1024000$ ODEs (C.1.4). Roofline models were obtained using Intel Advisor[11]. In all cases, the code is memory-bound and compute-bound. It means kernels in that area are fundamentally compute-bound but have upper memory roofs. The size and color of the dots in the roofline graph indicate how much of the total program time a loop or function takes. Small, green dots take up relatively little time, while large, red dots take up the most time. The red dots in the roofline plots for the MTSM_PREC, TSRK5DP, and TSRK8VR solvers correspond to the `MatMultAdd` kernel, and the red dot for MTSM corresponds to the `MatMult` kernel. The average computation time is shown in Table 6.33; the MTSM_PREC solver is the fastest of all solvers.



(a) MTSM (b) MTSM_PREC

(c) TSRK5DP (d) TSRK8VR

Figure 6.46: Roofline models.

Table 6.34 summarizes the results of the performance analysis obtained from the roofline model for each solver. Measurements were made on one compute node with 36 MPI processes. Each roofline model consists of different types and numbers of computational kernels. In this article, we focus on matrix-vector and matrix-matrix operations with performance $P \geq 0.1\,\text{GFLOPS}$. The column *performance* denotes the percentage of the total number of floating point operations per second with respect to the DP Vector FMA (Fused Multiply-Add) Peak. The FMA instruction can execute two operations – multiplication, and addition in one cycle. The average DP Vector FMA Peak is 79.44 GFLOPS.

The column *bandwidth* represents the percentage of DRAM memory bandwidth on a given machine. The average DRAM bandwidth is 15.23 [GB/s]. Notice that the bandwidth of the `MatMultAdd` kernels for TSRK5DP, and TSRK8VR solvers is 76.43% and 80.89% of DRAM and corresponds to the evaluation of $\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{b}$. The bandwidth of the

[11] https://www.intel.com/content/www/us/en/developer/tools/oneapi/advisor.html

`MatMatMult` kernel for MTSM_PREC solver is 96.6% and represents the precalculation of the matrix $\hat{\mathbf{A}}$ (4.45). The bandwidth of the `MatMult` kernel for the MTSM solver is 101.44% and represents the calculation of Taylor series terms (4.32). The size of the sparse matrix $\mathbf{A}$ is 40 MB, and the size of each vector $\mathbf{b}$ and $\mathbf{y}_0$ is 8 MB. The matrix $\mathbf{A}$ does not fit in the cache. Therefore, the kernels are bounded by DRAM bandwidth.

The AI stands for Arithmetic Intensity [FLOP/B]. The typical arithmetic intensity for stencils arising from PDEs is very low, typically 0.1–1.0 FLOP/B. The number of nonzeroes of the sparse matrix $\mathbf{A}$ is 0.0003%, and for the sparse matrix $\hat{\mathbf{A}}$ is 0.005%. The application is compute bound when $AI = 5.21$ FLOP/B. The AI of compute kernels listed in Table 6.34 is an order of magnitude lower than 5.21 FLOP/B.

Table 6.35 shows the total computation time, time spent running application code (columns denoted as CPU) and time spent in MPI calls (columns denoted as MPI). The results were measured using the Allinea Performance Reports[12] on eight nodes, eight MPI processes per node, therefore 64 processes in total. Note that the amount of time spent in CPU and MPI varies between solvers. Taylor-series-based solvers MTSM and MTSM_PREC spend an average of 72% on CPU and 28% on MPI. In contrast, Runge-Kutta solvers TSRK5DP and TSRK8VR spend, on average, 82% on CPU and 18% on MPI. The MTSM_PREC solver spends more time on MPI communication than Runge-Kutta solvers but is still the fastest of all solvers.

solver	**# processes (avgtime) [s]**								
	36	**144**	**288**	**432**	**576**	**720**	**864**	**1008**	**1152**
MTSM_PREC	30.80	6.83	2.95	1.51	1.02	0.63	0.52	0.45	0.43
MTSM	63.15	17.59	11.21	9.09	8.74	9.51	8.34	8.74	8.76
TSRK5DP	102.29	12.01	6.20	4.46	3.70	4.15	3.05	2.81	2.43
TSRK8VR	166.29	21.08	10.05	6.81	5.48	4.80	4.07	3.88	3.65

Table 6.33: Average time, 36 processes per node, 32 nodes, 1152 processes in total.

Solver	**Name**	**Kernel**	**Performance** [% of FMA peak]	**Bandwidth** [% of DRAM]	**AI** [FLOP/B]
MTSM	MatMult	$\mathbf{y} = \mathbf{A} \cdot x$	6.03	101.44	0.31
	`MatMultAdd`	$\mathbf{y} = x_2 + \mathbf{A} \cdot x_1$	4.83	87.66	0.29
MTSM_PREC	`MatMultAdd` (calc)	$\mathbf{y} = x_2 + \mathbf{A} \cdot x_1$	5.75	35.18	0.19
	`MatAXPY` (precalc)	$\mathbf{A} = c \cdot \mathbf{B} + \mathbf{A}$	2.40	3.03	0.05
	`MatMatMult` (precalc)	$\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$	5.72	96.6	0.05
	`MatMult` (precalc)	$\mathbf{y} = \mathbf{A} \cdot x$	1.39	23.31	0.31
TSRK5DP	`MatMultAdd`	$\mathbf{y} = x_2 + \mathbf{A} \cdot x_1$	4.21	76.43	0.29
TSRK8VR	`MatMultAdd`	$\mathbf{y} = x_2 + \mathbf{A} \cdot x_1$	4.45	80.89	0.29

Table 6.34: Roofline model data, 36 processes per node.

Table 6.36 shows the ratios with respect to the MTSM_PRECALC solver. MTSM and TSRK8VR use approximately four times more computation time than the MTSM_PRECALC method. The TSRK5DP method requires approximately three times more computation time than MTSM_PRECALC.

[12] https://docs.it4i.cz/software/debuggers/allinea-performance-reports

Solver	**Time** [s]	**CPU** [s]	**CPU** [%]	**MPI** [s]	**MPI** [%]
MTSM	46	31.28	68.00	14.67	31.90
MTSM_PREC	12	9.16	76.30	2.82	23.50
TSRK5DP	32	25.86	80.80	6.14	19.20
TSRK8VR	48	40.22	83.80	7.73	16.10

Table 6.35: Performance reports, eight nodes, eight processes per one node, 64 processes in total.

Solver	**Time** (Ratio)	**CPU** [s] (Ratio)	**CPU** [%] (Ratio)	**MPI** [s] (Ratio)	**MPI** [%] (Ratio)
MTSM	3.83	3.41	0.89	5.20	1.36
TSRK5DP	2.66	2.82	1.06	2.18	0.82
TSRK8VR	4.00	4.39	1.10	2.74	0.69

Table 6.36: Performance reports, eight nodes, eight processes per one node, 64 processes in total, $Ratio = solver/MTSM\_PRECALC$.

Chapter 7

Conclusion

This thesis dealt with the parallel numeric solution of partial differential equations. All the objectives of the work were fulfilled.

1. To analyze and evaluate the suitability of the higher-order Taylor series-based method for the solution of ordinary and partial differential equations.

   - The higher-order Taylor series-based method (MTSM) was introduced and described in Chapter 4 The stability boundaries of the MTSM and other state-of-the-art methods are analyzed in Chapter 3 and Appendix B.
   - The suitability of the MTSM for solving ordinary differential equations is clearly demonstrated in research articles, for example [pp2, pp13, pp21, pp22, pp28]. All numerical experiments are solved using the MTSM and the state-of-the-art MATLAB solvers, namely, ode23 (Bogacki-Shampine 3(2) method), ode45 (Dormand-Prince 5(4)), and ode113 (VSVO Adams-Bashforth-Moulton PECE solver of orders 1 to 13).
     - Publication [pp22] deals with linear and nonlinear MTSM and presents a set of practical examples such as the movement of the charged particle, the calculation of Fourier coefficients, and the Kepler problem. This publication extends [pp21].
     - Publication [pp28] focuses on linear and nonlinear (quadratic) ODEs solved by MTSM and carefully analyzes the van der Pol oscillator.
     - Publication [pp13] presents the solution to linear and nonlinear problems. The telegraph line represents a linear problem, whereas the Lorenz system is the nonlinear one.
     - Publication [pp2] analyzes the hardware representation for the solution of ODEs and compares the number of operations of Runge-Kutta and MTSM methods.
   - The suitability of the MTSM for solving partial differential equations is demonstrated in the following research articles:
     - Publications [pp2, pp8, pp18] show the experiments with wave equation implemented in MATLAB and focus on different types of finite difference formulas derived from the Taylor series, namely, backward, forward, and central. The spatial step size and the order of the finite difference formula influence the accuracy of the calculation. The publications show several

experiments with different types and orders of finite difference formulas together with errors in spatial and time domains.
  - Publications [pp11, pp18] demonstrate MATLAB experiments with wave equation discretized using seven-point difference formula in the spatial domain and different precision settings.
  - Publication [pp9] discusses the influence of arithmetic (8 B, 16 B, 32 B, and 64 B) on the accuracy of forward and central finite difference formulas.
  - Publications [pp3, pp4] introduce the hardware representation of the wave equation.
  - Publications [pp13, pp26] deal with the different sizes of the telegraph equation and compare the simulation results of MTSM with fixed integration step or fixed number of steps with the state-of-the-art Runge-Kutta solvers, namely, ode23, ode45, and ode113.
  - Publications [pp10, pp26] compare simulation results not only with the state-of-the-art Runge Kutta solvers, but also with the second-order implicit integration methods in LTSpice such as Gear, the trapezoidal method, and modified trapezoidal methods.
  - Publication [pp8] shows the first experiments with a telegraph equation with 100 segments using TKSL.

2. To propose, implement and deploy the Taylor series-based parallel method on an HPC cluster.

   - The parallel implementation approach of the MTSM is described in Section 4.9 and publications [pp14, pp16, pp17, pp24]. The matrix $\hat{\mathbf{A}}$ is precalculated only once before the calculation starts. This approach considerably eliminates the dependency between the terms of the Taylor series.
   - The MTSM is a VSVO method and allows the use of a larger integration step for the calculation than the state-of-the-art methods.

3. To experimentally evaluate the proposed method using the selected class of the second-order linear partial differential equations discretized in the spatial domain by finite difference formulas of different orders. Compare obtained results with the state-of-the-art numerical methods.

   - During my internship at Università della Svizzera italiana in Switzerland, the first experiments on the ICS cluster were carried out. When computing on multiple compute nodes, there was a significant increase in computation time compared to computing on a single node. The increase was probably due to the TCP communication between the nodes. Therefore, the Barbora supercomputer cluster was used for further experiments. The results are published in papers [pp14] and [pp16].
   - The parallel higher-order Taylor series-based method was implemented using the PETSc library and tested on the Barbora supercomputer cluster (IT4Innovations National Supercomputing Center, Ostrava, Czech Republic) using 32 compute nodes (36 MPI processes per node) as described in Chapter 6.
   - Numerical experiments were performed on a selected class of the second-order linear PDE. Note that the heat and wave equations were discretized in the spatial

domain by finite difference formulas of different orders. The analyzed problems are the following:

  - *Heat-3* – heat equation discretized with the three-point central difference formula (Sections 6.5 and C.1),
  - *Heat-5* – heat equation discretized with the five-point central difference formula (Sections 6.6 and C.2),
  - *Wave-3* – wave equation discretized with the three-point central difference formula (Sections 6.7 and C.3),
  - *Wave-5* – wave equation discretized with the three-point central difference formula (Sections 6.8 and C.4),
  - *Telegraph* – telegraph equation that represents a telegraph line (Sections 6.9 and C.5).

- Several numerical experiments were performed with different input data sizes for each selected problem type. Typical data sizes were 128000, 256000, 512000, 1024000 and 2048000 ODEs.
- For each size of the input data, four solvers compute the given problem type. Namely, MTSM (classical implementation of MTSM), MTSM_PRECALC (MTSM with precalculation of Tayor series terms), TSRK5DP (Dormand-Prince 5(4) method, ode45 in MATLAB) and TSRK8VR (Verner Runge-Kutta method of orders 8(7).
- For each numerical experiment, *parallel performance metrics* and *parallel-cost metrics* were evaluated (see Section 5.6). Performance metrics are *average computation time*, *parallel efficiency*, *parallel speedup*, and *speedup against the state-of-the-art Dormand-Prince 5(4) Runge-Kutta method (TSRK5DP solver)*. The parallel cost metrics are *parallel cost*, *parallel cost ratio*, and *parallel speedup-cost ratio*.
- The performance analysis results obtained from the roofline model for each solver are summarized in Section 6.10.
- Publication [pp24] focuses on the heat equation discretized in the spatial domain with the three-point central difference formula. The problem size is 1024000 ODEs. Parallel performance metrics are analyzed, together with roofline models for all selected solvers and performance reports.
- Publication [pp17] shows the experiments with a heat equation with 256000 ODEs and analyzes its parallel performance metrics. The numerical experiments are performed on the Barbora supercomputer cluster, IT4Innovations National Supercomputing Center, using 32 compute nodes (36 MPI processes per node). The numerical results are compared with the TSRK5DP and TSRK8VR Runge-Kutta solvers.

  The results can be summarized with respect to several aspects:

- Given the average computation time:
  - The MTSM_PRECALC is the fastest solver for all selected problem types.
  - Although MTSM suffers from computational dependencies between the terms of the Taylor series, it can provide results comparable or even better than TSRK5DP or TSRKVR, especially for *Wave-3*, *Wave-5*, and *Telegraph*.

  - The behavior of the TSRK5DP and TSRK8VR solvers is comparable, although, in some cases, the TSRK8VR solver is slightly slower than TSRK5DP.
- Given the ability to provide results for all 1–32 compute nodes:
  - The MTSM_PRECALC solver always computed results for all 32 compute nodes for all selected problem types.
  - The MTSM solver always provided results for *Heat-3*, *Heat-5*, and *Telegraph*. For the *Wave-3*, it did not calculate results for all 32 nodes for $S = 2048000$. For *Wave-5*, for $S \geq 1024000$.
  - The TSRK5DP and TSRK8VR solvers always provided results for all 32 compute nodes for*Heat-3*, *Heat-5*, and *Telegraph*. In the case of *Wave-3* and *Wave-5*, they did not calculate the results for all 32 nodes for problem sizes $S \geq 512000$.
- Given the maximum number of compute nodes where parallel efficiency $E \geq 50\%$ for all 32 compute nodes:
  - The MTSM_PRECALC solver provides $E \geq 50\%$ for *Heat-3* and *Heat-5*. For *Wave-3*, $E \geq 50\%$ is for problem sizes $S \geq 512000$, for *Wave-5*, for $S \geq 256000$.
  - The MTSM solver provides $E \geq 50\%$ for *Heat-5* for $S = 2048000$.
  - The TSRK5DP solver provides $E \geq 50\%$ for *Heat-3* and *Heat-5* for $S \geq 512000$ and for *Telegraph*.
  - The TSRK5DP solver provides $E \geq 50\%$ for *Heat-3* and *Heat-5* for $S = 512000$ and $S = 1024000$, and for *Telegraph*.
- Given the speedup-cost curve reaches 32 compute nodes (i.e., it is worth allocating 32 compute nodes):
  - The MTSM_PRECALC solver: for *Heat-3*, *Heat-5*, and *Telegraph* for all problem sizes. For *Wave-3* and *Wave-5* for problem sizes $S \geq 512000$.
  - The MTSM solver: for *Telegraph* for all problem sizes.
  - The TSRK5DP solver: for *Telegraph*, $S = 2048000$.
  - The TSRK8VR solver: for *Heat-3*, where $S = 1024000$, and for *Heat-5*, where $S = 512000$ and $S = 1024000$.

4. Discuss the achieved results and suggest possible extensions.
   - The results are discussed and carefully described in Chapter 6 and Appendix C, and summarized in this chapter.
   - Future work and possible extensions are outlined in Section 7.1.

7.1 Future work

There are several directions on how to develop further the research results, mainly related to the MTSM for solving systems of ODEs:

- *Stopping-rule analysis.* In the current implementation, the calculation of the current integration step is terminated if the norm of the absolute value of the last three consecutive Taylor series terms is less than or equal to the required accuracy ε (4.13). An interesting experiment could be to use more or fewer Taylor series terms in the termination condition of the integration step and observe how the order and integration step of the method change.

- *Reducing computation time by monitoring Taylor series terms.* In some problems, the last Taylor series terms can be omitted. If the absolute value of the currently calculated term of the Taylor series is less than the required accuracy, then the value of this term is considered to be zero. In the next integration step, this term is not included in the calculation. The calculation of the integration step ends if all terms of the Taylor series have zero value. This approach promises to reduce the order of the method.

- *Improving the integration stepsize control.* In the current implementation, the step is halved during the calculation if the maximum order of the method is exceeded. The three following approaches to finer control of the integration step size can include:

  1. *Define minORD and maxORD thresholds.* If the order of the method in the currently calculated integration step is less than the required lower threshold $minORD$, then the integration step is increased by a factor of f. Analogously, if the order of the method is greater than the upper threshold $maxORD$, the integration step is decreased by a factor of f.
  2. *Maintaining the halving-stepsize history.* If the integration step size in the current time step is not halved, this information is stored. If an integration step is not halved during several consecutive integration steps, the integration step can be increased by the selected factor (e.g., doubled).
  3. *Combination of approaches mentioned above.* That means defining the $minORD$ and $maxORD$ thresholds and maintaining the history of halving the integration step.

- *Support variable-precision arithmetic.* The variable-precision arithmetic can be beneficial in problems requiring long-term integration or extremely accurate numerical simulations.

- *Two- and three-dimensional PDEs.* The two and three-dimensional PDEs can be analyzed. Two- and three-dimensional PDEs can model systems with more complex spatial relationships and interactions than one-dimensional PDEs.

- *Parallelization of nonlinear problems.* Detailed analysis of computational dependencies will be required.

Bibliography

[1] Abad, A., Barrio, R., Blesa, F. and Rodríguez, M. Algorithm 924: TIDES, a Taylor series integrator for differential equations. *ACM Transactions on Mathematical Software (TOMS)*. Association for Computing Machinery. November 2012, vol. 39, no. 1, p. 1–28. DOI: 10.1145/2382585.2382590. ISSN 0098-3500.

[2] Abhyankar, S., Brown, J., Constantinescu, E. M., Ghosh, D., Smith, B. F. et al. *PETSc/TS: A Modern Scalable ODE/DAE Solver Library*. arXiv, June 2018. DOI: 10.48550/arXiv.1806.01437.

[3] Ajima, Y., Inoue, T., Hiramoto, S., Uno, S., Sumimoto, S. et al. Tofu Interconnect 2: System-on-Chip Integration of High-Performance Interconnect. In: *Supercomputing*. Cham: Springer International Publishing, 2014, p. 498–507. ISBN 9783319075181.

[4] Ajima, Y., Kawashima, T., Okamoto, T., Shida, N., Hirai, K. et al. The Tofu Interconnect D. In: IEEE. *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. 2018, p. 646–654. DOI: 10.1109/CLUSTER.2018.00090. ISBN 9781538683200.

[5] Almasi, G. S. and Gottlieb, A. *Highly parallel computing*. Benjamin-Cummings Publishing Co., Inc., 1994. ISBN 9780805304435.

[6] Amdahl, G. M. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In: *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*. New York, NY, USA: Association for Computing Machinery, 1967, p. 483—-485. DOI: 10.1145/1465482.1465560. ISBN 9781450378956.

[7] Archibald, T., Fraser, C. and Grattan Guinness, I. The History of Differential Equations, 1670–1950. *Oberwolfach Reports*. European Mathematical Society - EMS - Publishing House GmbH. September 2005, vol. 1, no. 4, p. 2729–2794. DOI: 10.4171/owr/2004/51. ISSN 1660-8941.

[8] Ashino, R., Nagase, M. and Vaillancourt, R. Behind and beyond the MATLAB ODE suite. *Computers & Mathematics with Applications*. Elsevier. September 2000, vol. 40, no. 4, p. 491–512. DOI: 10.1016/S0898-1221(00)00175-9. ISSN 0898-1221.

[9] 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. ISSN 0045-7930.

[10] Balay, S., Abhyankar, S., Adams, M. F., Benson, S., Brown, J. et al. *PETSc/-TAO Users Manual*. ANL-21/39 - Revision 3.18. Argonne National Laboratory, 2022.

[11] Balay, S., Abhyankar, S., Adams, M. F., Benson, S., Brown, J. et al. *PETSc Web page*. 2022. Available at: https://petsc.org/.

[12] Balay, S., Abhyankar, S., Adams, M. F., Benson, S., Brown, J. et al. *Supported External Software*. 2022. Available at: https://petsc.org/release/install/external_software/.

[13] Balay, S., Gropp, W. D., McInnes, L. C. and Smith, B. F. Efficient Management of Parallelism in Object-Oriented Numerical Software Libraries. In: Arge, E., Bruaset, A. M. and Langtangen, H. P., ed. Boston, MA: Birkhäuser Boston, 1997, p. 163–202. ISBN 9781461219866.

[14] Barrio, R., Blesa, F. and Lara, M. High-precision numerical solution of ODE with high-order Taylor methods in parallel. *Monografías de la Real Academia de Ciencias de Zaragoza*. January 2003, vol. 22, p. 67–74. ISSN 1132-6360.

[15] Barrio, R., Blesa, F. and Lara, M. VSVO formulation of the taylor method for the numerical solution of ODEs. *Computers & Mathematics with Applications*. 2005, vol. 50, no. 1, p. 93–111. DOI: 10.1016/j.camwa.2005.02.010. ISSN 0898-1221.

[16] Barrio, R., Rodríguez, M., Abad, A. and Blesa, F. Breaking the limits: The Taylor series method. *Applied Mathematics and Computation*. Elsevier. 2011, vol. 217, p. 7940–7954. DOI: 10.1016/j.amc.2011.02.080. ISSN 0096-3003.

[17] Barrio, R., Rodríguez, M., Abad, A. and Blesa, F. TIDES: A free software based on the Taylor series method. *Monografías de la Real Academia de Ciencias de Zaragoza*. Elsevier. 2011, vol. 35, p. 83–95. ISSN 1132-6360.

[18] Barrio, R. Performance of the Taylor series method for ODEs/DAEs. *Applied Mathematics and Computation*. 2005, vol. 163, no. 2, p. 525–545. DOI: 10.1016/j.amc.2004.02.015. ISSN 0096-3003.

[19] Barrio, R. Sensitivity analysis of ODEs/DAEs using the Taylor series method. *SIAM Journal on Scientific Computing*. SIAM. 2006, vol. 27, no. 6, p. 1929–1947. DOI: 10.1137/030601892. ISSN 1095-7197.

[20] Barton, D., Willers, I. and Zahar, R. The automatic solution of systems of ordinary differential equations by the method of Taylor series. *The Computer Journal*. Oxford University Press. January 1971, vol. 14, no. 3, p. 243–248. DOI: 10.1093/comjnl/14.3.243. ISSN 0010-4620.

[21] Batcher, K. Design of a Massively Parallel Processor. *IEEE Transactions on Computers*. Los Alamitos, CA, USA: IEEE Computer Society. 1980, vol. 29, no. 09, p. 836–840. DOI: 10.1109/TC.1980.1675684. ISSN 1557-9956.

[22] Bendtsen, C. and Stauning, O. *TADIFF, a flexible C++ package for automatic differentiation*. Lyngby, Denmark, 1997.

[23] Bergman, K., Borkar, S., Campbell, D., Carlson, W., Dally, W. et al. Exascale computing study: Technology challenges in achieving exascale systems. *Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Technical Representative*. 2008, vol. 15, p. 181.

[24] Bertsekas, D. and Tsitsiklis, J. *Parallel and distributed computation: numerical methods*. Athena Scientific, 2015. ISBN 9781886529151.

[25] Bogacki, P. and Shampine, L. F. An Efficient Runge-Kutta (4,5) Pair. *Computers & Mathematics with Applications*. 1996, vol. 32, no. 6, p. 15–28. DOI: 10.1016/0898-1221(96)00141-1. ISSN 0898-1221.

[26] Bogacki, P. and Shampine, L. F. A 3 (2) pair of Runge-Kutta formulas. *Applied Mathematics Letters*. 1989, vol. 2, no. 4, p. 321–325. DOI: 10.1016/0893-9659(89)90079-7. ISSN 0893-9659.

[27] Brezis, H. and Browder, F. Partial Differential Equations in the 20th Century. *Advances in Mathematics*. 1998, vol. 135, no. 1, p. 76–144. DOI: 10.1006/aima.1997.1713. ISSN 0001-8708.

[28] Buluç, A., Fineman, J. T., Frigo, M., Gilbert, J. R. and Leiserson, C. E. Parallel Sparse Matrix-Vector and Matrix-Transpose-Vector Multiplication Using Compressed Sparse Blocks. In: *Proceedings of the Twenty-First Annual Symposium on Parallelism in Algorithms and Architectures*. New York, NY, USA: Association for Computing Machinery, 2009, p. 233–244. SPAA '09. DOI: 10.1145/1583991.1584053. ISBN 9781605586069.

[29] Burden, R. L. and Faires, J. D. *Numerical Analysis*. 8th ed. Brooks-Cole Publishing, 2004. ISBN 0-534-39200-8.

[30] Butcher, J. C. *Numerical Methods for Ordinary Differential Equations*. Second Edition, John Wiley & Sons Ltd., 2008. ISBN 9780-470-72335-7.

[31] Chang, Y. and Corliss, G. ATOMFT: solving ODEs and DAEs using Taylor series. *Computers & Mathematics with Applications*. 1994, vol. 28, no. 10, p. 209–233. DOI: 10.1016/0898-1221(94)00193-6. ISSN 0898-1221.

[32] Choi, C. Q. *The Beating Heart of the World's First Exascale Supercomputer*, 24. June 2022. Available at: https://spectrum.ieee.org/frontier-exascale-supercomputer.

[33] Corliss, G. and Chang, Y. Solving ordinary differential equations using Taylor series. *ACM Transactions on Mathematical Software (TOMS)*. ACM New York, NY, USA. 1982, vol. 8, no. 2, p. 114–144. DOI: 10.1145/355993.355995. ISSN 1557-7295.

[34] Courant, R., Friedrichs, K. and Lewy, H. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische annalen*. Springer. 1928, vol. 100, no. 1, p. 32–74. DOI: 10.1007/BF01448839.

[35] Crank, J. and Nicolson, P. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society*. Cambridge University Press. 1947, vol. 43, no. 1, p. 50–67. DOI: 10.1017/S0305004100023197. ISSN 1469-8064.

[36] Dally, W. J. Performance analysis of k-ary n-cube interconnection networks. *IEEE transactions on Computers*. IEEE. 1990, vol. 39, no. 6, p. 775–785. DOI: 10.1109/12.53599.

[37] Dally, W. J. and Towles, B. P. *Principles and practices of interconnection networks*. 1st ed. Morgan Kaufmann, 2004. ISBN 9780122007514.

[38] De Sensi, D., Di Girolamo, S., McMahon, K. H., Roweth, D. and Hoefler, T. An In-Depth Analysis of the Slingshot Interconnect. In: *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. 2020, p. 1–14. DOI: 10.1109/SC41405.2020.00039. ISSN 2167-4337.

[39] Dennard, R., Gaensslen, F., Yu, H.-N., Rideout, V., Bassous, E. et al. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*. 1974, vol. 9, no. 5, p. 256–268. DOI: 10.1109/JSSC.1974.1050511. ISSN 0018-9200.

[40] Derradji, S., Palfer Sollier, T., Panziera, J.-P., Poudes, A. and Atos, F. W. The BXI interconnect architecture. In: IEEE. *2015 IEEE 23rd Annual Symposium on High-Performance Interconnects*. 2015, p. 18–25. DOI: 10.1109/HOTI.2015.15. ISBN 9781467391603.

[41] Dimova, S., Hristov, I., Hristova, R., Puzynin, I., Puzynina, T. et al. *OpenMP parallelization of multiple precision Taylor series method*. arXiv, 2019. DOI: 10.48550/ARXIV.1908.09301.

[42] Dolean, V., Jolivet, P. and Nataf, F. *An introduction to domain decomposition methods: algorithms, theory, and parallel implementation*. Society for Industrial and Applied Mathematics, 2016. ISBN 9781611974058.

[43] Dongarra, J. Sunway TaihuLight supercomputer makes its appearance. *National Science Review*. september 2016, vol. 3, p. 265–266. DOI: 10.1093/nsr/nww044. ISSN 2095-5138.

[44] Dormand, J. and Prince, P. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*. 1980, vol. 6, no. 1, p. 19–26. DOI: 10.1016/0771-050X(80)90013-3. ISSN 0377-0427.

[45] Duato, J., Yalamanchili, S. and Ni, L. *Interconnection networks*. Morgan Kaufmann, 2011. ISBN 9780123991805.

[46] Duncan, R. A survey of parallel computer architectures. *Computer*. 1990, vol. 23, no. 2, p. 5–16. DOI: 10.1109/2.44900. ISSN 1558-0814.

[47] Egab, L. Chapter 10 - Computer-aided engineering findings on the physics of tire/road noise. In: Wang, X., ed. *Automotive Tire Noise and Vibrations*. Butterworth-Heinemann, 2020, p. 217–243. DOI: https://doi.org/10.1016/B978-0-12-818409-7.00010-6. ISBN 978-0-12-818409-7.

[48] Emmett, M. and Minion, M. Toward an efficient parallel in time method for partial differential equations. *Communications in Applied Mathematics and Computational Science*. MSP. 2012, vol. 7, no. 1, p. 105–132. DOI: 10.2140/camcos.2012.7.105. ISSN 2157-5452.

[49] Evans, G., Blackledge, J. and Yardley, P. *Analytic methods for partial differential equations*. Springer Science & Business Media, 2012. ISBN 9783540761242.

[50] Evans, L. C. *Partial differential equations*. Providence, RI: American Mathematical Society, 2022. ISBN 9781470469429.

[51] Eymard, R., Gallouët, T. and Herbin, R. Finite volume methods. In: *Solution of Equation in Rn (Part 3), Techniques of Scientific Computing (Part 3)*. Elsevier, 2000, vol. 7, p. 713–1018. Handbook of Numerical Analysis. DOI: 10.1016/S1570-8659(00)07005-8. ISSN 1570-8659.

[52] Fadlisyah, M., Ölveczky, P. C. and Ábrahám, E. Adaptive-Step-Size Numerical Methods in Rewriting-Logic-Based Formal Analysis of Interacting Hybrid Systems. *Electronic Notes in Theoretical Computer Science*. 2011, vol. 274, p. 17–32. DOI: https://doi.org/10.1016/j.entcs.2011.07.004. ISSN 1571-0661. 4th International Workshop on Harnessing Theories for Tool Support in Software (TTSS).

[53] Falgout, R. D., Jones, J. E. and Yang, U. M. The Design and Implementation of hypre, a Library of Parallel High Performance Preconditioners. In: *Numerical Solution of Partial Differential Equations on Parallel Computers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, p. 267–294. DOI: 10.1007/3-540-31619-1_8. ISBN 978-3-540-31619-0.

[54] Falgout, R. D. and Yang, U. M. Hypre: A Library of High Performance Preconditioners. In: Sloot, P. M. A., Hoekstra, A. G., Tan, C. J. K. and Dongarra, J. J., ed. *Computational Science — ICCS 2002*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, p. 632–641. DOI: 10.1007/3-540-47789-6_66. ISBN 978-3-540-47789-1.

[55] Farlow, S. J. *Partial differential equations for scientists and engineers*. Courier Corporation, 2012. ISBN 860-1234581253.

[56] Fehlberg, E. Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems. *Computing*. 1970, vol. 6. DOI: doi:10.1007/BF02241732. ISSN 1436-5057.

[57] Ferziger, J. H., Perić, M. and Street, R. L. *Computational methods for fluid dynamics*. 3rd ed. Springer, 2002. ISBN 9783-642-56026-2.

[58] Flynn, M. J. Some Computer Organizations and Their Effectiveness. *IEEE Transactions on Computers*. 1972, C-21, no. 9, p. 948–960. DOI: 10.1109/TC.1972.5009071. ISSN 1557-9956.

[59] Flynn, M. Very high-speed computing systems. *Proceedings of the IEEE*. 1966, vol. 54, no. 12, p. 1901–1909. DOI: 10.1109/PROC.1966.5273. ISSN 1558-2256.

[60] Fornberg, B. Fast generation of weights in finite difference formulas. In: *Recent Developments in Numerical Methods and Software for ODEs/DAEs/PDEs*. World Scientific, 1992, p. 97–123. DOI: 10.1142/9789814335867_0006. ISBN 9789814506397.

[61] Foster, I. *Designing and building parallel programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN 9780201575941.

[62] Gander, M. J. 50 years of time parallel time integration. In: *Multiple shooting and time domain decomposition methods*. Springer International Publishing, 2015, p. 69–113. DOI: 10.1007/978-3-319-23321-5_3. ISBN 978-3-319-23321-5.

[63] Gear, C. W. Parallel methods for ordinary differential equations. *Calcolo*. Springer. 1988, vol. 25, 1-2, p. 1–20. DOI: 10.1007/BF02575744. ISSN 1126-5434.

[64] Geist, A. *A Decade of Design To Reach Exascale for ModSim*. 2022. ModSim 2022: Workshop on Modeling & Simulation of Systems and Applications. Available at: https://www.bnl.gov/modsim2022/files/talks/al-geist.pdf.

[65] Gibbons, A. A Program for the Automatic Integration of Differential Equations using the Method of Taylor Series. *The Computer Journal*. Oxford University Press. 1960, vol. 3, no. 2, p. 108–111. DOI: 10.1093/comjnl/3.2.108. ISSN 0010-4620.

[66] Gioiosa, R. Chapter 5 - Resilience for extreme scale computing. In: Vega, A., Bose, P. and Buyuktosunoglu, A., ed. *Rugged Embedded Systems*. Boston: Morgan Kaufmann, 2017, p. 123–148. DOI: https://doi.org/10.1016/B978-0-12-802459-1.00005-1. ISBN 9780-12-802459-1.

[67] Grama, A., Kumar, V., Gupta, A. and Karypis, G. *Introduction to parallel computing*. 2nd ed. Addison-Wesley, 2003. ISBN 9780201648652.

[68] Griewank, A. and Walther, A. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Society for Industrial and Applied Mathematics, 2008. ISBN 9780898716597.

[69] Griffiths, D. F. and Higham, D. J. *Numerical Methods for Ordinary Differential Equations: Initial Value Problems*. Springer Science & Business Media, 2010. ISBN 9780857291479.

[70] Griffiths, G. and Schiesser, W. E. *Traveling wave analysis of partial differential equations: numerical and analytical methods with MATLAB and Maple*. Academic Press, 2011. ISBN 9780123846525.

[71] Gupta, G. K., Sacks Davis, R. and Tescher, P. E. A Review of Recent Developments in Solving ODEs. *ACM Comput. Surv.* New York, NY, USA: Association for Computing Machinery. March 1985, vol. 17, no. 1, p. 5–47. DOI: 10.1145/4078.4079. ISSN 0360-0300.

[72] Gustafson, J. L. Reevaluating Amdahl's law. *Communications of the ACM*. New York, USA: Association for Computing Machinery. 1988, vol. 31, no. 5, p. 532–533. DOI: 10.1145/42411.42415. ISSN 0001-0782.

[73] Hager, G. and Wellein, G. *Introduction to High Performance Computing for Scientists and Engineers*. 1st ed. CRC Press, 2010. ISBN 9781439811924.

[74] Hairer, E., Nørsett, S. P. and Wanner, G. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer-Verlag, 1993. ISBN 9783540566700.

[75] Hairer, E. and Wanner, G. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer-Verlag, 1996. ISBN 9783642052217.

[76] Halin, H. The applicability of Taylor series methods in simulation. In: *SummerSim '15: Proceedings of the Conference on Summer Computer Simulation*. 1983, p. 1032–1076. DOI: 10.5555/2874916. ISBN 9781510810594.

[77] Hall, G. and Watt, J. M. *Modern numerical methods for ordinary differential equations*. Oxford University Press, 1976. ISBN 9780198533481.

[78] Hamdi, S., Schiesser, W. E. and Griffiths, G. W. Method of lines. *Scholarpedia*. 2007, vol. 2, no. 7, p. 1–11. DOI: 10.4249/scholarpedia.2859.

[79] Hapla, V. *Scalable Parallel Computations with PETSc @ CSC*. May 2019. Helsinki, Finland. Available at: https://events.prace-ri.eu/event/871/attachments/888/1299/petsc_04_matrices.pdf.

[80] Hon, Y., Schaback, R. and Zhong, M. The meshless Kernel-based method of lines for parabolic equations. *Computers & Mathematics with Applications*. 2014, vol. 68, 12, Part A, p. 2057–2067. DOI: 10.1016/j.camwa.2014.09.015. ISSN 0898-1221.

[81] Hristov, I., Hristova, R., Dimova, S., Armyanov, P., Shegunov, N. et al. Parallelizing Multiple Precision Taylor Series Method for Integrating the Lorenz System. In: Georgiev, I., Kostadinov, H. and Lilkova, E., ed. *Advanced Computing in Industrial Mathematics*. Springer International Publishing, 2023, p. 56–66. DOI: 10.1007/978-3-031-20951-2_6. ISBN 978-3-031-20951-2.

[82] Hristov, I., Hristova, R., Dimova, S., Armyanov, P., Shegunov, N. et al. On the efficient parallel computing of long term reliable trajectories for the Lorenz system. In:. 2021, vol. 2933, p. 78–88. DOI: 10.18287/1613-0073-2019. ISSN 1613-0073.

[83] Hutton, D. V. *Fundamentals of Finite Element Analysis*. 1st ed. McGraw-Hill Science, 2004. ISBN 9780072395365.

[84] Hyman, J. *The method of lines solution of partial differential equations*. 1976. ISBN 9781330357095.

[85] Ilic, A., Pratas, F. and Sousa, L. Cache-aware Roofline model: Upgrading the loft. *IEEE Computer Architecture Letters*. 2014, vol. 13, no. 1, p. 21–24. DOI: 10.1109/L-CA.2013.6. ISSN 1556-6064.

[86] InfiniBand Trade Association. *InfiniBand^{TM} Architecture Specification*. Available at: https://www.infinibandta.org/ibta-specification/.

[87] Jeffreys, H., Jeffreys, B. and Swirles, B. *Methods of mathematical physics*. 3rd ed. Cambridge University Press, 2000. ISBN 9780521664028.

[88] Johnson, E. E. Completing an MIMD Multiprocessor Taxonomy. *SIGARCH Comput. Archit. News*. New York, NY, USA: Association for Computing Machinery. 1988, vol. 16, no. 3, p. 44—47. DOI: 10.1145/48675.48682. ISSN 0163-5964.

[89] Jorba, A. and Zou, M. A Software Package for the Numerical Integration of ODEs by Means of High-Order Taylor Methods. *Experimental Mathematics*. A K Peters, Ltd. 2005, vol. 14, no. 1, p. 99–117. DOI: em/1120145574. ISSN 1944-950X.

[90] Karp, A. H. and Flatt, H. P. Measuring Parallel Processor Performance. *Commun. ACM*. New York, NY, USA: Association for Computing Machinery. May 1990, vol. 33, no. 5, p. 539—543. DOI: 10.1145/78607.78614. ISSN 0001-0782.

[91] Kendall, R., Sosonkina, M., Gropp, W., Numrich, R. and Sterling, T. Parallel Programming Models Applicable to Cluster Computing and Beyond. In:. January 2006, vol. 51, p. 3–54. DOI: 10.1007/3-540-31619-1_1. ISBN 3-540-29076-1.

[92] Kianrad, A., Khonsari, A., Yazdani, N. and Dadlani, A. Performance Analysis of Optical Packet-Switched Meshes: Metrics and Modeling. *The CSI Journal on Computer Science and Engineering*. January 2007. ISSN 2676-542X.

[93] Kim, J., Dally, W. J., Scott, S. and Abts, D. Technology-Driven, Highly-Scalable Dragonfly Topology. In: *2008 International Symposium on Computer Architecture*. 2008, p. 77–88. DOI: 10.1109/ISCA.2008.19. ISBN 9780769531748.

[94] Kocina, F. *FOS: Fast ODE Solver*. 2017. Faculty of Information Technology, Brno University of Technology. Available at: https://www.fit.vut.cz/research/product/518/.en.

[95] Kopriva, J., Kraus, M., Kunovsky, J. and Satek, V. Semi-Analytical Computations Based on TKSL. In: *2008 Second UKSIM European Symposium on Computer Modeling and Simulation*. 2008, p. 159–164. DOI: 10.1109/EMS.2008.39. ISBN 9780769533254.

[96] Kouya, T. A Brief Introduction to MPIGMP & MPIBNCpack. April 2008. Available at: https://na-inet.jp/na/bnc/brief_intro_mpibncpack.pdf.

[97] Kraus, M., Kunovský, J. and Šátek, V. Taylor Series Numerical Integrator. In: *2008 Second UKSIM European Symposium on Computer Modeling and Simulation*. 2008, p. 177–180. DOI: 10.1109/EMS.2008.40. ISBN 9780769533254.

[98] Kraus, M., Vlastimil, K., Kunovský, J. and Šátek, V. Parallel Computations Based on Analogue Principles. In: *2009 11th International Conference on Computer Modelling and Simulation*. 2009, p. 111–116. DOI: 10.1109/UKSIM.2009.15. ISBN 9780769535937.

[99] Kreiss, H.-O. and Scherer, G. Method of lines for hyperbolic differential equations. *SIAM Journal on Numerical Analysis*. SIAM. 1992, vol. 29, no. 3, p. 640–646. ISSN 0036-1429.

[100] Kumar, M. and Kumar, P. A finite element approach for finding positive solutions of semilinear elliptic Dirichlet problems. *Numerical Methods for Partial Differential Equations*. 2009, vol. 25, no. 5, p. 1119–1128. DOI: https://doi.org/10.1002/num.20390. ISSN 0749159X.

[101] Kumar, V. *Parallel Algorithm and Computation*. Khanna Publishing, 2013. ISBN 9789381068861.

[102] Kunovský, J. *TKSL: Taylor-Kunovský Simulation Language*. 2005. Taylor Series based simulation language for continuous systems simulations. Faculty of Information Technology, Brno University of Technology. Available at: https://www.fit.vut.cz/research/product/51/.en.

[103] Kunovský, J. *Modern Taylor Series Method*. 1994. Habilitation. Faculty of Electrical Engineering and Computer Science, Brno University of Technology.

[104] Laboratory, O. R. N. *Frontier supercomputer debuts as world's fastest, breaking exascale barrier*, 30. May 2022. Available at: https://www.ornl.gov/news/frontier-supercomputer-debuts-worlds-fastest-breaking-exascale-barrier/.

[105] Lara, M., Elipe, A. and Palacios, M. Automatic programming of recurrent power series. *Mathematics and Computers in Simulation*. 1999, vol. 49, no. 4, p. 351–362. DOI: https://doi.org/10.1016/S0378-4754(99)00087-7. ISSN 0378-4754.

[106] LeVeque, R. J. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. Society for Industrial and Applied MAthematics, 2007. ISBN 9780898716290.

[107] LeVeque, R. J. et al. *Finite volume methods for hyperbolic problems*. Cambridge University Press, 2002. ISBN 9780511791253.

[108] Li, Z., Saad, Y. and Sosonkina, M. PARMS: a parallel version of the algebraic recursive multilevel solver. *Numerical Linear Algebra with Applications*. 2003, vol. 10, 5-6, p. 485–509. DOI: https://doi.org/10.1002/nla.325. ISSN 1099-1506.

[109] Liao, S. On the reliability of computed chaotic solutions of non-linear differential equations. *Tellus A: Dynamic Meteorology and Oceanography*. Taylor & Francis. 2008, vol. 61, no. 4, p. 550–564.

[110] Liao, X.-K., Pang, Z.-B., Wang, K.-F., Lu, Y.-T., Xie, M. et al. High performance interconnect network for Tianhe system. *Journal of Computer Science and Technology*. Springer. 2015, vol. 30, no. 2, p. 259–272. DOI: 10.1111/j.1600-0870.2009.00402.x. ISSN 1600-0870.

[111] Liao, X., Lu, K., Yang, C., Li, J. wen, Yuan, Y. et al. Moving from exascale to zettascale computing: challenges and techniques. *Frontiers of Information Technology & Electronic Engineering*. 2018, vol. 19, no. 10, p. 1236–1244. DOI: 10.1631/FITEE.1800494. ISSN 2095-9184.

[112] Liu, W. and Vinter, B. CSR5: An Efficient Storage Format for Cross-Platform Sparse Matrix-Vector Multiplication. In: *Proceedings of the 29th ACM on International Conference on Supercomputing*. New York, NY, USA: Association for Computing Machinery, 2015, p. 339—-350. ICS '15. DOI: 10.1145/2751205.2751209. ISBN 9781450335591.

[113] Loeb, A. M. and Schiesser, W. E. Stiffness and Accuracy in the Method of Lines Integration of Partial Differential Equations. In: *Stiff Differential Systems*. Boston, MA: Springer US, 1974, p. 229–243. DOI: 10.1007/978-1-4684-2100-2_18. ISBN 978-1-4684-2100-2.

[114] Lu, P.-J., Lai, M.-C. and Chang, J.-S. A Survey of High-Performance Interconnection Networks in High-Performance Computer Systems. *Electronics*. Multidisciplinary Digital Publishing Institute. 2022, vol. 11, no. 9. DOI: 10.3390/electronics11091369. ISSN 2079-9292.

[115] Lu, Y. Paving the way for China exascale computing. *CCF Transactions on High Performance Computing*. august 2019, vol. 1, p. 63–72. DOI: 10.1007/s42514-019-00010-y. ISSN 2524-4922.

[116] Makino, K. and Berz, M. COSY INFINITY Version 9. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. 2006, vol. 558, no. 1, p. 346–350. DOI: 10.1016/j.nima.2005.11.109.

ISSN 0168-9002. Proceedings of the 8th International Computational Accelerator Physics Conference.

[117] Malony, A. D. Metrics. In: Padua, D., ed. *Encyclopedia of Parallel Computing*. Boston, MA: Springer US, 2011, p. 1124–1130. DOI: 10.1007/978-0-387-09766-4_69. ISBN 9780-387-09766-4.

[118] Mikulášek, K. *Polynomial Transformations of Systems of Differential Equations and Their Applications*. Brno, Czech Republic, 2000. Dissertation. Faculty of Electrical Engineering and Computer Science, Brno University of Technology.

[119] Miletics, E. and Molnárka, G. Taylor Series Method with Numerical Derivatives for numerical solution of ODE initial values problems. *Hungarian Electronic Journal of Sciences, Section Applied and Numerical Mathematics*. Citeseer. 2003, p. 1–16. DOI: 10.3233/JCM-2004-41-213.

[120] Min, G. *Performance modelling and analysis of multicomputer interconnection networks*. Glasgow, United Kingdom, 2003. Dissertation. University of Glasgow, Faculty of Information and Mathematical Science.

[121] Mohazzabi, P. and L. Becker, J. Numerical Solution of Differential Equations by Direct Taylor Expansion. *Journal of Applied Mathematics and Physics*. 2017, vol. 05, no. 03, p. 623–630. DOI: 10.4236/jamp.2017.53053. ISSN 2327-4352.

[122] Molnárka, G. and Miletics, E. Implicit Extension of Taylor Series Method with Numerical Derivatives. In: Wiley Online Library. *PAMM: Proceedings in Applied Mathematics and Mechanics*. 2003, vol. 3, no. 1, p. 569–570. DOI: 10.1002/pamm.200310552. ISSN 617-7061.

[123] Mondigo, A., Ueno, T., Sano, K. and Takizawa, H. Comparison of Direct and Indirect Networks for High-Performance FPGA Clusters. In: *Applied Reconfigurable Computing. Architectures, Tools, and Applications: 16th International Symposium, ARC 2020, Toledo, Spain, April 1–3, 2020, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2020, p. 314—329. DOI: 10.1007/9783030445348_24. ISBN 978-3-030-44533-1.

[124] Moore, G. E. Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter*. 2006, vol. 11, no. 3, p. 33–35. DOI: 10.1109/N-SSC.2006.4785860. ISSN 1098-4232.

[125] Muralidhar, R., Borovica Gajic, R. and Buyya, R. Energy efficient computing systems: Architectures, abstractions and modeling to techniques and standards. *ACM Computing Surveys (CSUR)*. ACM New York, NY. 2022, vol. 54, 11s, p. 1–37. DOI: 10.1145/3511094. ISSN 0360-0300.

[126] Nedialkov, N. S. and Pryce, J. Solving Differential-Algebraic Equations by Taylor Series (III): the DAETS Code. *Journal of Numerical Analysis, Industrial and Applied Mathematics*. Elsevier. 2008, vol. 3, p. 61–80. ISSN 1790-8140.

[127] Nedialkov, N. S. and Pryce, J. D. Solving differential-algebraic equations by Taylor series (I): Computing Taylor coefficients. *BIT Numerical Mathematics*. Springer. 2005, vol. 45, no. 3, p. 561–591. DOI: 10.1007/s10543-005-0019-y. ISSN 1572-9125.

[128] Nedialkov, N. S. and Pryce, J. D. Solving differential-algebraic equations by Taylor series (II): Computing the System Jacobian. *BIT Numerical Mathematics*. Springer. 2006, vol. 47, no. 1, p. 121–135. DOI: 10.1007/s10543-006-0106-8. ISSN 1572-9125.

[129] Padua, D. *Encyclopedia of parallel computing*. 1st ed. Springer Science & Business Media, 2011. ISBN 9780387097664.

[130] Pang, Z., Xie, M., Zhang, J., Zheng, Y., Wang, G. et al. The TH Express high performance interconnect networks. *Frontiers of Computer Science*. Springer. 2014, vol. 8, no. 3, p. 357–366. DOI: 0.1007/s11704-014-3500-9. ISSN 2095-2236.

[131] Ploskas, N. and Samaras, N. *GPU programming in MATLAB*. Morgan Kaufmann, 2016. ISBN 9780128051320.

[132] Potter, J. L. *The massively parallel processor*. The MIT Press, 1985. ISBN 9780262661799.

[133] Powell, J. The Quantum Limit to Moore's Law. *Proceedings of the IEEE*. september 2008, vol. 96, p. 1247–1248. DOI: 10.1109/JPROC.2008.925411. ISSN 1558-2256.

[134] Qiao, P., Wang, X., Yang, X., Fan, Y. and Lan, Z. Preliminary Interference Study About Job Placement and Routing Algorithms in the Fat-Tree Topology for HPC Applications. In: *2017 IEEE International Conference on Cluster Computing (CLUSTER)*. 2017, p. 641–642. DOI: 10.1109/CLUSTER.2017.90. ISSN 1552-5244.

[135] Quinn, M. J. *Parallel computing theory and practice*. McGraw-Hill College, 1994. ISBN 9780070512948.

[136] Rall, L. B. *Automatic Differentiation: Techniques and Applications*. Springer, 1981. Lecture Notes in Computer Science. ISBN 9783540108610.

[137] R.A.Moore. *Interval Analysis*. New York, USA: Prentice-Hall, 1966. ISSN 2577-9435.

[138] Rao, S. S. *The finite element method in engineering*. Butterworth-Heinemann, 2018. ISBN 9781856176613.

[139] Reddy, J. N. *An introduction to the finite element method*. 3rd ed. McGraw-Hill Education, 2005. ISBN 9780072466850.

[140] Rezzolla, L. *Numerical Methods for the Solution of Hyperbolic Partial Differential Equations Lecture Notes*. June 2005.

[141] Rodríguez, M. and Barrio, R. Reducing rounding errors and achieving Brouwer's law with Taylor Series Method. *Applied Numerical Mathematics*. Elsevier. 2012, vol. 62, p. 1014–1024. DOI: 10.1016/j.apnum.2012.03.008. ISSN 0168-9274.

[142] Roser, M., Ritchie, H. and Mathieu, E. Technological Change. *Our World in Data*. 2013. Available at: https://ourworldindata.org/technological-change.

[143] Rusanovsky, M., Harel, R., Alon, L.-o., Mosseri, I., Levin, H. et al. *BACKUS: Comprehensive High-Performance Research Software Engineering Approach for Simulations in Supercomputing Systems*. arXiv, 2019. DOI: 10.48550/arXiv.1910.06415.

[144] Sahni, S. and Thanvantri, V. Parallel computing: Performance metrics and models. *IEEE Parallel and Distributed Technology*. Citeseer. 1996, vol. 4, no. 1, p. 43–56. ISSN 1063-6552.

[145] Schiesser, W. E. *The numerical method of lines: integration of Partial Differential Equations*. Elsevier, 2012. ISBN 9780126241303.

[146] Schiesser, W. E. and Griffiths, G. W. *A compendium of partial differential equation models: method of lines analysis with Matlab*. Cambridge University Press, 2009. ISBN 9780511576270.

[147] Schinsel, A. M. *What is a Roofline Model?*, 3. February 2017. Available at: https://www.intel.com/content/www/us/en/developer/articles/guide/intel-advisor-roofline.html.

[148] Scott, L. R., Clark, T. and Bagheri, B. *Scientific parallel computing*. Princeton University Press, 2021. ISBN 9780691119359.

[149] Shainer, G. *Super-Connecting the Supercomputers – Innovations Through Network Topologies*, 15. July 2019. Available at: https://www.hpcwire.com/2019/07/15/super-connecting-the-supercomputers-innovations-through-network-topologies/.

[150] Shakeri, F. and Dehghan, M. The method of lines for solution of the one-dimensional wave equation subject to an integral conservation condition. *Computers & Mathematics with Applications*. Elsevier. 2008, vol. 56, no. 9, p. 2175–2188. DOI: https://doi.org/10.1016/j.camwa.2008.03.055. ISSN 0898-1221.

[151] 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. DOI: 10.1137/S1064827594276424. ISSN 1095-7197.

[152] Smith, B. F. Domain Decomposition Methods for Partial Differential Equations. In: *Parallel Numerical Algorithms*. Springer Netherlands, 1997, p. 225–243. DOI: 10.1007/978-94-011-5412-3_8. ISBN 978-94-011-5412-3.

[153] Smith, G. D. *Numerical Solution of Partial Differential Equations: Finite Difference Methods*. Oxford university press, 1985. ISBN 9780-19859-650-9.

[154] Solodushkin, S. I. and Iumanova, I. F. Parallel numerical methods for ordinary differential equations: a survey. In: CEUR-WS. *CEUR Workshop Proceedings*. 2016, vol. 1729, p. 1–10. ISSN 1613-0073.

[155] Strohmaier, E., Dongarra, J., Simon, H. and Meuer, M. *TOP500: The List: The Linpack Benchmark*. Available at: https://www.top500.org/project/linpack/.

[156] Strohmaier, E., Dongarra, J., Simon, H. and Meuer, M. *TOP500: The List: June 2022*. 2022. Available at: https://www.top500.org/lists/top500/2022/06/.

[157] Teh, M. Y., Wilke, J. J., Bergman, K. and Rumley, S. Design Space Exploration of the Dragonfly Topology. In: Kunkel, J. M., Yokota, R., Taufer, M. and Shalf, J., ed. *High Performance Computing*. Cham: Springer International Publishing, 2017, p. 57–74. ISBN 978-3-319-67630-2.

[158] The MathWorks Inc. *MATLAB: Choose an ODE Solver*. [cit. 2021-02-04]. Natick Massachusetts United States. Available at: https://www.mathworks.com/help/matlab/math/choose-an-ode-solver.html.

[159] The MathWorks Inc. *MATLAB: Variable-precision arithmetic*. [cit. 2021-03-06]. Natick Massachusetts United States. Available at: https://www.mathworks.com/help/symbolic/vpa.html.

[160] Toselli, A. and Widlund, O. *Domain decomposition methods-algorithms and theory*. Springer Science & Business Media, 2004. ISBN 9783540266624.

[161] Trew, A. and Wilson, G. *Past, present, parallel: a survey of available parallel computer systems*. Springer Science & Business Media, 2012. ISBN 9781447118428.

[162] Trobec, R., Vasiljević, R., Tomašević, M., Milutinović, V., Beivide, R. et al. Interconnection networks in petascale computer systems: A survey. *ACM Computing Surveys (CSUR)*. ACM New York, NY, USA. 2016, vol. 49, no. 3, p. 1–24. DOI: 10.1145/2983387. ISSN 0360-0300.

[163] Turczyn, C. *Exascale Computing'S Four Biggest Challenges and How They Were Overcome*, 18. October 2021. Available at: https://www.olcf.ornl.gov/2021/10/18/exascale-computings-four-biggest-challenges-and-how-they-were-overcome.

[164] U.S. Department of Energy National Laboratory Operated by the University of California. *Performance and Algorithms Research*. 2022. Available at: https://crd.lbl.gov/divisions/amcr/computer-science-amcr/par/research/roofline/introduction/.

[165] Verma, A. An introduction to automatic differentiation. *Current Science*. JSTOR. 2000, p. 804–807. ISSN 2795-8639.

[166] Verner, J. Numerically optimal Runge-Kutta pairs with interpolants. *Numerical Algorithms*. march 2010, vol. 53, p. 383–396. DOI: 10.1007/s11075-009-9290-3. ISSN 1017-1398.

[167] Verner, J. H. Explicit Runge–Kutta Methods with Estimates of the Local Truncation Error. *SIAM Journal on Numerical Analysis*. SIAM. 1978, vol. 15, no. 4, p. 772–790. DOI: 10.1137/0715051. ISSN 1095-7170.

[168] Verner, J. *Jim Verner's Refuge for Runge-Kutta Pairs*. Available at: https://www.sfu.ca/~jverner/.

[169] Vichnevetsky, R. *Numerical stability of methods of lines for partial differential equations*. Rutgers University, 1971. DOI: 10.7282/t3-85rg-4q53.

[170] Wang, R., Lu, K., Chen, J., Zhang, W., Li, J. et al. Brief introduction of TianHe exascale prototype system. *Tsinghua Science and Technology*. TUP. 2020, vol. 26, no. 3, p. 361–369. DOI: 10.26599/TST.2020.9010009. ISSN 1007-0214.

[171] Wermelinger, F. *High Performance Computing for Science and Engineering I: Strong and Weak Scaling*. Computational Science & Engineering Laboratory, 2020. Available at: https://www.cse-lab.ethz.ch/wp-content/uploads/2018/11/amdahl_gustafson.pdf.

[172] Wesseling, P. *Principles of computational fluid dynamics*. Springer Science & Business Media, 2009. ISBN 9783642051463.

[173] Williams, S., Datta, K., Carter, J., Oliker, L., Shalf, J. et al. PERI - Auto-tuning memory-Intensive kernels for multicore - art. no. 012038. *Journal of Physics: Conference Series*. august 2008, vol. 125. DOI: 10.1088/1742-6596/125/1/012038. ISSN 1742-6596.

[174] Williams, S., Waterman, A. and Patterson, D. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*. ACM New York, NY, USA. 2009, vol. 52, no. 4, p. 65–76. DOI: 10.1145/1498765.1498785. ISSN 1557-7317.

[175] Wouwer, A. V., Saucez, P. and Vilas, C. *Simulation of ODE/PDE Models with MATLAB®, OCTAVE and SCILAB: Scientific and Engineering Applications*. Springer, 2014. ISBN 9783319067902.

[176] Wulf, W. and McKee, S. Hitting the Memory Wall: Implications of the Obvious. *Computer Architecture News*. january 1996, vol. 23, p. 20–24. DOI: 10.1145/216585.216588. ISSN 0163-5964.

[177] Yang, C. *Introduction to the Roofline Model*. June 2019. ISC High Performance 2019, Frankfurt, Germany. Available at: https://www.nersc.gov/assets/Uploads/Tutorial-ISC2019-Intro-v2.pdf.

[178] Zeneli, M., Nikolopoulos, A., Karellas, S. and Nikolopoulos, N. Chapter 7 - Numerical methods for solid-liquid phase-change problems. In: Datas, A., ed. *Ultra-High Temperature Thermal Energy Storage, Transfer and Conversion*. Woodhead Publishing, 2021, p. 165–199. Woodhead Publishing Series in Energy. DOI: https://doi.org/10.1016/B978-0-12-819955-8.00007-7. ISBN 978-0-12-819955-8.

[179] Zeng, C. V. P. W. S. Krylov subspace and multigrid methods applied to the incompressible Navier-Stokes equations. In: *Seventh Copper Mountain Conference on Multigrid Methods*. 1995. ISSN 0191-7811.

[180] Zienkiewicz, O. C. and Morice, P. *The finite element method in engineering science*. McGraw-Hill London, 1971. ISBN 978-0070941380.

[181] Zill, D. G. *A first course in differential equations with modeling applications*. 10th ed. Cengage Learning, 2012. ISBN 9781111827052.

List of publications

[pp1] 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, p. 1–4. DOI: 10.1063/1.4992650. ISSN 0094-243X. Available at: https://www.fit.vut.cz/research/publication/11226.

[pp2] Chaloupka, J., Nečasová, G., Veigend, P., Kunovský, J. and Šátek, V. Modern Taylor series method in numerical integration: PART 1. In: *16th Czech-Polish Conference Modern Mathematical Methods in Engineering (3mi)*. 2017, vol. 6, no. 4, p. 263–273. ISBN 978-83-65265-14-2. Available at: https://www.fit.vut.cz/research/publication/11354.

[pp3] Kocina, F., Kunovský, J., Marek, M., Nečasová, G., Schirrer, A. et al. New Trends in Taylor Series Based Computations. In: *12th International Conference of Numerical Analysis and Applied Mathematics*. American Institute of Physics, 2015, vol. 2015, no. 1648, p. 1–4. DOI: 10.1063/1.4913136. ISBN 978-0-7354-1287-3. Available at: https://www.fit.vut.cz/research/publication/10496.

[pp4] 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.

[pp5] Kocina, F., Nečasová, G., Veigend, P., Chaloupka, J., Šátek, V. et al. Modelling VLSI Circuits Using Taylor Series. In: *14th International Conference of Numerical Analysis and Applied Mathematics*. American Institute of Physics, 2017, no. 1863, p. 1–4. DOI: 10.1063/1.4992513. ISSN 0094-243X. Available at: https://www.fit.vut.cz/research/publication/11228.

[pp6] 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.

[pp7] Kocina, F., Šátek, V., Veigend, P., Nečasová, G., Valenta, V. et al. New Trends in Taylor Series Based Applications. In: *13rd International Conference of Numerical Analysis and Applied Mathematics*. American Institute of Physics,

2015, vol. 2016, no. 1648, p. 1–4. DOI: 10.1063/1.4952173. ISBN 978-0-7354-1287-3. Available at: https://www.fit.vut.cz/research/publication/10856.

[pp8] Kunovský, J., Šátek, V., Nečasová, G., Veigend, P. and Kocina, F. The Positive Properties of Taylor Series Method. In: *Proceedings of the 13th International Conference Informatics' 2015*. Institute of Electrical and Electronics Engineers, 2015, p. 156–160. DOI: 10.1109/Informatics.2015.7377825. ISBN 978-1-4673-9867-1. Available at: https://www.fit.vut.cz/research/publication/10923.

[pp9] Nečasová, G., Kocina, F., Veigend, P., Chaloupka, J., Šátek, V. et al. Solving Wave Equation Using Finite Differences and Taylor Series. In: *14th International Conference of Numerical Analysis and Applied Mathematics*. American Institute of Physics, 2017, no. 1863, p. 1–4. DOI: 10.1063/1.4992649. ISSN 0094-243X. Available at: https://www.fit.vut.cz/research/publication/11229.

[pp10] Nečasová, G., Kocina, F., Veigend, P., Šátek, V. and Kunovský, J. Model of the Telegraph Line. In: *Informatics 2017 - 14th International Scientific Conference on Informatics*. Institute of Electrical and Electronics Engineers, 2017, p. 271–275. DOI: 10.1109/INFORMATICS.2017.8327259. ISBN 978-1-5386-0888-3. Available at: https://www.fit.vut.cz/research/publication/11305.

[pp11] Nečasová, G., Kunovský, J. and Šátek, V. Numerical Solution of Wave Equation Using Higher Order Methods. In: *15th International Conference of Numerical Analysis and Applied Mathematics*. American Institute of Physics, 2017, p. 1–4. DOI: 10.1063/1.5043964. ISBN 978-0-7354-1690-1. Available at: https://www.fit.vut.cz/research/publication/11417.

[pp12] Nečasová, G., Kunovský, J., Šátek, V., Chaloupka, J. and Veigend, P. Taylor Series Based Differential Formulas. In: *MATHMOD VIENNA 2015 - 8th Vienna Conference on Mathematical Modelling*. ARGE Simulation News, 2015, p. 705–706. ARGESIM REPORT No. 44. DOI: 10.1016/j.ifacol.2015.05.209. ISBN 978-3-901608-46-9. Available at: https://www.fit.vut.cz/research/publication/10746.

[pp13] 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.

[pp14] 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.

[pp15] 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.

[pp16] Nečasová, G. and Šátek, V. Taylor series based parallel numerical solution of partial differential equations. In:. 2021. In press. Available at: https://www.fit.vut.cz/research/publication/12510.

[pp17] Nečasová, G. and Šátek, V. Parallel solution of parabolic partial differential equation using higher-order method. In:. 2022. In press. Available at: https://www.fit.vut.cz/research/publication/12780.

[pp18] Valenta, V., Nečasová, G., Kunovský, J., Šátek, V. and Kocina, F. Adaptive Solution of the Wave Equation. In: *Proceedings of the 5th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*. SciTePress - Science and Technology Publications, 2015, p. 154–162. ISBN 978-989-758-120-5. Available at: https://www.fit.vut.cz/research/publication/10822.

[pp19] Veigend, P., Kunovský, J., Kocina, F., Nečasová, G., Šátek, V. et al. Electronic Representation of Wave Equation. In: *13rd International Conference of Numerical Analysis and Applied Mathematics*. American Institute of Physics, 2015, no. 1738, p. 1–4. DOI: 10.1063/1.4952174. ISBN 978-0-7354-1392-4. Available at: https://www.fit.vut.cz/research/publication/10857.

[pp20] 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.

[pp21] 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*. Institute of Electrical and Electronics Engineers, 2019, p. 25–30. DOI: 10.1109/Informatics47936.2019.9119258. ISBN 978-1-7281-3179-5. Available at: https://www.fit.vut.cz/research/publication/12004.

[pp22] 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.

[pp23] 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.

[pp24] Veigend, P., Nečasová, G. and Šátek, V. Taylor Series Method in Numerical Integration: Linear and Nonlinear problems. In:. 2022. In press. Available at: https://www.fit.vut.cz/research/publication/12754.

[pp25] Veigend, P., Raffai, P., Nečasová, G., Šátek, V. and Kunovský, J. Gas flow through the piston ring pack. In: *15rd International Conference of Numerical*

Analysis and Applied Mathematics. American Institute of Physics, 2018, p. 1–4. DOI: 10.1063/1.5043967. ISBN 978-0-7354-1690-1. Available at: https://www.fit.vut.cz/research/publication/11084.

[pp26] 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.

[pp27] Šátek, V., Veigend, P. and Nečasová, G. Taylor Series Based Solution of Nonlinear-quadratic ODE Systems. In: *MATHMOD VIENNA 2018 - 9th Vienna International Conference on Mathematical Modelling*. ARGE Simulation News, 2018, p. 99–100. DOI: 10.11128/arep.55. ISBN 978-3-901608-91-9. Available at: https://www.fit.vut.cz/research/publication/11544.

[pp28] Šátek, V., Veigend, P. and Nečasová, G. Taylor Series Based Integration in Electric Circuits Simulations. *Advances in Electrical and Electronic Engineering*. 2019, vol. 17, no. 3, p. 352–359. DOI: 10.15598/aeee.v17i3.3369. ISSN 1804-3119. Available at: https://www.fit.vut.cz/research/publication/11732.

Appendices

Appendix A

Finite difference coefficients

The finite difference coefficients for the forward and backward finite difference formulas are in Tables A.1 and A.2, respectively.

M	O	stencil x-coordinates								
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	**1**	-1	1							
	2	$-\frac{3}{2}$	2	$-\frac{1}{2}$						
	3	$-\frac{11}{6}$	3	$-\frac{3}{2}$	$\frac{1}{3}$					
	4	$-\frac{25}{12}$	4	-3	$\frac{4}{3}$	$-\frac{1}{4}$				
	5	$-\frac{137}{60}$	5	-5	$\frac{10}{3}$	$-\frac{5}{4}$	$\frac{1}{5}$			
	6	$-\frac{49}{20}$	6	$-\frac{15}{2}$	$\frac{20}{3}$	$-\frac{15}{4}$	$\frac{6}{5}$	$-\frac{1}{6}$		
	7	$-\frac{363}{140}$	7	$-\frac{21}{2}$	$\frac{35}{3}$	$-\frac{35}{4}$	$\frac{21}{5}$	$-\frac{7}{6}$	$\frac{1}{7}$	
	8	$-\frac{761}{280}$	8	-14	$\frac{56}{3}$	$-\frac{35}{2}$	$\frac{56}{5}$	$-\frac{14}{3}$	$\frac{8}{7}$	$-\frac{1}{8}$
2	**1**	1	-2	1						
	2	2	-5	4	-1					
	3	$\frac{35}{12}$	$-\frac{26}{3}$	$\frac{19}{2}$	$-\frac{14}{3}$	$\frac{11}{12}$				
	4	$\frac{15}{4}$	$-\frac{77}{6}$	$\frac{107}{6}$	-13	$\frac{61}{12}$	$-\frac{5}{6}$			
	5	$\frac{203}{45}$	$-\frac{87}{5}$	$\frac{117}{4}$	$-\frac{254}{9}$	$\frac{33}{2}$	$-\frac{27}{5}$	$\frac{137}{180}$		
	6	$\frac{469}{90}$	$-\frac{223}{10}$	$\frac{879}{20}$	$-\frac{949}{18}$	41	$-\frac{201}{10}$	$\frac{1019}{180}$	$-\frac{7}{10}$	
	7	$\frac{1916}{327}$	$-\frac{962}{35}$	$\frac{621}{10}$	$-\frac{4006}{45}$	$\frac{691}{8}$	$-\frac{282}{5}$	$\frac{2143}{90}$	$-\frac{206}{35}$	$\frac{363}{560}$
3	**1**	-1	3	-3	1					
	2	$-\frac{5}{2}$	9	-12	7	$-\frac{3}{2}$				
	3	$-\frac{17}{4}$	$\frac{71}{4}$	$-\frac{59}{2}$	$\frac{49}{2}$	$-\frac{41}{4}$	$\frac{7}{4}$			
	4	$-\frac{49}{8}$	29	$-\frac{461}{8}$	62	$-\frac{307}{8}$	13	$-\frac{15}{8}$		
	5	$-\frac{967}{120}$	$\frac{638}{15}$	$-\frac{3929}{40}$	$\frac{389}{3}$	$-\frac{2545}{24}$	$\frac{268}{5}$	$-\frac{1849}{120}$	$\frac{29}{15}$	
	6	$-\frac{801}{80}$	$\frac{349}{6}$	$-\frac{18353}{120}$	$\frac{2391}{10}$	$-\frac{1457}{6}$	$\frac{4891}{30}$	$-\frac{561}{8}$	$\frac{527}{30}$	$-\frac{469}{240}$
4	**1**	1	-4	6	-4	1				
	2	3	-14	26	-24	11	-2			
	3	$\frac{35}{6}$	-31	$\frac{137}{2}$	$-\frac{242}{3}$	$\frac{107}{2}$	-19	$\frac{17}{6}$		
	4	$\frac{28}{3}$	$-\frac{111}{2}$	142	$-\frac{1219}{6}$	176	$-\frac{185}{2}$	$\frac{82}{3}$	$-\frac{7}{2}$	
	5	$\frac{1069}{80}$	$-\frac{1316}{15}$	$\frac{15289}{60}$	$-\frac{2144}{5}$	$\frac{10993}{24}$	$-\frac{4772}{15}$	$\frac{2803}{20}$	$-\frac{536}{15}$	$\frac{967}{240}$

Table A.1: Coefficients for forward finite difference formulas, $M = 4$, $N = 8$, $x_0 = 0$.

M	O	stencil x-coordinates								
		-8	-7	-6	-5	-4	-3	-2	-1	0
1	1								-1	1
	2							$\frac{1}{2}$	-2	$\frac{3}{2}$
	3						$-\frac{1}{3}$	$\frac{3}{2}$	-3	$\frac{11}{6}$
	4					$\frac{1}{4}$	$-\frac{4}{3}$	3	-4	$\frac{25}{12}$
	5				$-\frac{1}{5}$	$\frac{5}{4}$	$-\frac{10}{3}$	5	-5	$\frac{137}{60}$
	6			$\frac{1}{6}$	$-\frac{6}{5}$	$\frac{15}{4}$	$-\frac{20}{3}$	$\frac{15}{2}$	-6	$\frac{49}{20}$
	7		$-\frac{1}{7}$	$\frac{7}{6}$	$-\frac{21}{5}$	$\frac{35}{4}$	$-\frac{35}{3}$	$\frac{21}{2}$	-7	$\frac{363}{140}$
	8	$\frac{1}{8}$	$-\frac{8}{7}$	$\frac{14}{3}$	$-\frac{56}{5}$	$\frac{35}{2}$	$-\frac{56}{3}$	14	-8	$\frac{761}{280}$
2	1							1	-2	1
	2						-1	4	-5	2
	3					$\frac{11}{12}$	$-\frac{14}{3}$	$\frac{19}{2}$	$-\frac{26}{3}$	$\frac{35}{12}$
	4				$-\frac{5}{6}$	$\frac{61}{12}$	-13	$\frac{107}{6}$	$-\frac{77}{6}$	$\frac{15}{4}$
	5			$\frac{137}{180}$	$-\frac{27}{5}$	$\frac{33}{2}$	$-\frac{254}{9}$	$\frac{117}{4}$	$-\frac{87}{5}$	$\frac{203}{45}$
	6		$-\frac{7}{10}$	$\frac{1019}{180}$	$-\frac{201}{10}$	41	$-\frac{949}{18}$	$\frac{879}{20}$	$-\frac{223}{10}$	$\frac{469}{90}$
	7	$\frac{363}{560}$	$-\frac{206}{35}$	$\frac{2143}{90}$	$-\frac{282}{5}$	$\frac{691}{8}$	$-\frac{4006}{45}$	$\frac{621}{10}$	$-\frac{962}{35}$	$\frac{1916}{327}$
3	1						1	-3	3	-1
	2					$\frac{3}{2}$	-7	12	-9	$\frac{5}{2}$
	3				$-\frac{7}{4}$	$\frac{41}{4}$	$-\frac{49}{2}$	$\frac{59}{2}$	$-\frac{71}{4}$	$\frac{17}{4}$
	4			$\frac{15}{8}$	-13	$\frac{307}{8}$	-62	$\frac{461}{8}$	-29	$\frac{49}{8}$
	5		$-\frac{29}{15}$	$\frac{1849}{120}$	$-\frac{268}{5}$	$\frac{2545}{24}$	$-\frac{389}{3}$	$\frac{3929}{40}$	$-\frac{638}{15}$	$\frac{967}{120}$
	6	$\frac{469}{240}$	$-\frac{527}{30}$	$\frac{561}{8}$	$-\frac{4891}{30}$	$\frac{1457}{6}$	$-\frac{2391}{10}$	$\frac{18353}{120}$	$-\frac{349}{6}$	$\frac{801}{80}$
4	1					1	-4	6	-4	1
	2				-2	11	-24	26	-14	3
	3			$\frac{17}{6}$	-19	$\frac{107}{2}$	$-\frac{242}{3}$	$\frac{137}{2}$	-31	$\frac{35}{6}$
	4		$-\frac{7}{2}$	$\frac{82}{3}$	$-\frac{185}{2}$	176	$-\frac{1219}{6}$	142	$-\frac{111}{2}$	$\frac{28}{3}$
	5	$\frac{967}{240}$	$-\frac{536}{15}$	$\frac{2803}{20}$	$-\frac{4772}{15}$	$\frac{10993}{24}$	$-\frac{2144}{5}$	$\frac{15289}{60}$	$-\frac{1316}{15}$	$\frac{1069}{80}$

Table A.2: Coefficients for backward finite difference formulas, $M = 4$, $N = 8$, $x_0 = 0$.

Appendix B

Higher-order Taylor series method

Tables B.1 and B.2 show the intersections of MTSM for orders 26–45 and 46–64, respectively.

Order	$\lvert\mathbf{S_R}\rvert$	$\lvert\mathbf{S_I}\rvert$
26	−11.06	±11.23
27	−11.43	±8.13
28	−11.81	±9.74
29	−12.18	±11.28
30	−12.55	±12.43
31	−12.92	±13.09
32	−13.30	±9.70
33	−13.67	±11.31
34	−14.04	±12.88
35	−14.41	±14.16
36	−14.78	±14.92
37	−15.16	±15.09
38	−15.53	±12.88
39	−15.90	±14.46
40	−16.27	±15.85
41	−16.63	±16.72
42	−17.05	±17.13
43	−17.39	±14.45
44	−17.73	±16.04
45	−18.22	±17.51

Table B.1: S_R and S_I intersections of MTSM for orders 26–45.

Order	$\lvert\mathbf{S_R}\rvert$	$\lvert\mathbf{S_I}\rvert$
46	−18.37	±18.50
47	−18.00	±19.03
48	−19.05	±16.02
49	−18.48	±17.61
50	−21.21	±19.13
51	−21.77	±20.26
52	−22.57	±20.89
53	−23.01	±17.59
54	−18.60	±19.18
55	−24.51	±20.73
56	−21.82	±21.98
57	−25.54	±22.72
58	−17.77	±19.16
59	−26.89	±20.75
60	−27.12	±22.32
61	−27.18	±23.68
62	−28.73	±24.52
63	−28.63	±24.89
64	−30.28	±22.32

Table B.2: S_R and S_I intersections of MTSM for orders 46–64.

Tables B.3 and B.4 of generating system of ODEs follow.

	Generated function	**Generating ODEs**	**Initial conditions**
1.	$y_1 = t$	$y_1' = 1$	$y_1(0) = 0$
2.	$y_1 = t + a$	$y_1' = 1$	$y_1(0) = a$
3.	$y_1 = t^2$	$y_1' = y_2$ $y_2' = 2$	$y_1(0) = 0$ $y_2(0) = 0$
	$y_1 = y^2$	$y_1' = y_2 y'$ $y_2' = 2y'$	$y_1(0) = y_0^2$ $y_2(0) = 2y_0$
4.	$y_1 = t^3$	$y_1' = y_2$ $y_2' = y_3$ $y3' = 6$	$y_1(0) = 0$ $y_2(0) = 0$ $y_3(0) = 0$
	$y_1 = y^3$	$y_1' = y_2 y'$ $y_2' = y_3 y'$ $y_3' = 6y'$	$y_1(0) = y_0^3$ $y_2(0) = 3y_0^2$ $y_3(0) = 6y_0$
5.	$y_1 = t^n$ $n \in \mathbb{N}$	$y_1' = y_2$ $y_2' = y_3$ $\vdots$ $y_n' = n!$	$y_1(0) = 0$ $y_2(0) = 0$ $\vdots$ $y_n(0) = 0$
	$y_1 = y^n$	$y_1' = y_2 y'$ $y_2' = y_3 y'$ $\vdots$ $y_n' = n! y'$	$y_1(0) = y_0^n$ $y_2(0) = n y_0^{n-1}$ $\vdots$ $y_n(0) = n! y_0$
6.	$y_1 = e^{at}$	$y_1' = a y_1$	$y_1(0) = 1$
	$y_1 = e^{ay}$	$y_1' = a y_1 y'$	$y_1(0) = e^{a y_0}$
7.	$y_1 = \sin t$	$y_1' = y_2$ $y_2' = -y_1$	$y_1(0) = 0$ $y_2(0) = 1$
	$y_1 = \sin y$	$y_1' = y_2 y'$ $y_2' = -y_1 y'$	$y_1(0) = \sin y_0$ $y_2(0) = \cos y_0$
8.	$y_1 = \cos t$	$y_1' = y_2$ $y_2' = -y_1$	$y_1(0) = 1$ $y_2(0) = 0$
	$y_1 = \cos y$	$y_1' = y_2 y'$ $y_2' = -y_1 y'$	$y_1(0) = \cos y_0$ $y_2(0) = -\sin y_0$
9.	$y_1 = \tan t$	$y_1' = y' + y_1 y_1$	$y_1(0) = 0$
	$y_1 = \tan y$	$y_1' = y' + y_1 y_1 y'$	$y_1(0) = \tan y_0$
10.	$y_1 = \cot(t + a)$	$y_1' = -(1 + y_1 y_1)$	$y_1(0) = \cot a$
	$y_1 = \cot(y + a)$	$y_1' = -(y' + y_1 y_1 y')$	$y_1(0) = \cot(y_0 + a)$

Table B.3: Tables of generating system of ODEs.

	Generated function	**Generating ODEs**	**Initial conditions**
11.	$y_1 = \sqrt{t+a}$	$y_1' = \frac{1}{2}\frac{1}{y_1}$	$y_1(0) = \sqrt{a}$
	$y_1 = \sqrt{y+a}$	$y_1' = \frac{1}{2}\frac{1}{y_1}y'$	$y_1(0) = \sqrt{y_0+a}$
12.	$y_1 = \sqrt[3]{t+a}$	$y_1' = \frac{1}{3}\frac{1}{y_2}$ $y_2' = y_3 y_1'$ $y_3' = 2y_1'$	$y_1(0) = \sqrt[3]{a}$ $y_2(0) = (y_0+a)^{\frac{2}{3}}$ $y_3(0) = 2\sqrt[3]{y_0+a}$
	$y_1 = \sqrt[3]{y+a}$	$y_1' = \frac{1}{3}\frac{1}{y_2}y'$ $y_2' = y_3 y_1'$ $y_3' = 2y_1'$	$y_1(0) = \sqrt[3]{y_0+a}$ $y_2(0) = (y_0+a)^{\frac{2}{3}}$ $y_3(0) = 2\sqrt[3]{y_0+a}$
13.	$y_1 = \sqrt[n]{t+a}$	$y_1' = \frac{1}{n}\frac{1}{y_2}$ $y_2' = y_3 y_1'$ $y_3' = y_4 y_1'$ $\vdots$ $y_n' = (n-1)!y_1'$	$y_1(0) = \sqrt[n]{a}$ $y_2(0) = a^{\frac{n-1}{n}}$ $y_3(0) = (n-1)a^{\frac{n-2}{n}}$ $\vdots$ $y_n(0) = (n-1)!a^{\frac{1}{n}}$
	$y_1 = \sqrt[n]{y+a}$	$y_1' = \frac{1}{n}\frac{1}{y_2}y'$ $y_2' = y_3 y_1'$ $y_3' = y_4 y_1'$ $\vdots$ $y_n' = (n-1)!y_1'$	$y_1(0) = \sqrt[n]{y_0+a}$ $y_2(0) = (y_0+a)^{\frac{n-1}{n}}$ $y_3(0) = (n-1)(y_0+a)^{\frac{n-1}{n}}$ $\vdots$ $y_n(0) = (n-1)!(y_0+a)^{\frac{1}{n}}$
14.	$y_1 = (t+a)^2$ $n \in \mathbb{R}$	$y_1' = ny_2y_1$ $y_2' = -y_2y_2$	$y_1(0) = a^n$ $y_2(0) = \frac{1}{a}$
	$y_1 = (y+a)^2$	$y_1' = ny_2y_1y'$ $y_2' = -y_2y_2y'$	$y_1(0) = (y_0+a)^n$ $y_2(0) = \frac{1}{y_0+a}$
15.	$y_1 = \ln(t+a)$	$y_1' = y_2$ $y_2' = -y_2y_2$	$y_1(0) = \ln a$ $y_2(0) = \frac{1}{a}$
	$y_1 = \ln(y+a)$	$y_1' = \frac{y'}{y}$	$y_1(0) = \ln(y_0+a)$

Table B.4: Continued: Tables of generating system of ODEs.

Appendix C

Results

This chapter contains tables for each numerical experiment, specifically tables for the average time, parallel efficiency, speedup, speedup against the TSRK5DP solver, and for information about the input data.

Table cells where the parallel efficiency is greater than or equal to 50% are marked in bold. Also, the cells of the table showing the speedup ratio with respect to the TSRK5DP solver where $speedup = TSRK5DP/solver \gg 1$ indicates significantly faster computation using the given *solver*, are marked in bold.

C.1 Heat equation – three-point central difference formula

Subsections C.1.1–C.1.5 provide numerical results for $S = 128000$, $S = 256000$, $S = 512000$, $S = 1024000$, and $S = 2048000$, respectively.

C.1.1 $S = 128000$, three-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	2.32	0.81	0.53	0.41	0.35	0.30	0.28	0.26
MTSM	7.49	5.33	4.42	4.23	4.01	4.26	4.29	4.82
TSRK5DP	5.59	3.21	2.37	2.05	1.84	1.74	1.55	1.57
TSRK8VR	9.04	4.87	3.59	3.29	2.71	2.33	2.19	2.09
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	0.24	0.24	0.28	0.21	0.20	0.27	0.19	0.19
MTSM	4.76	5.09	5.00	4.51	4.90	5.10	5.13	4.80
TSRK5DP	1.58	1.63	1.36	1.35	1.29	1.23	1.25	1.25
TSRK8VR	2.04	1.90	1.78	1.80	1.75	1.91	1.66	1.77
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	0.13	0.13	0.13	0.13	0.13	0.12	0.12	0.11
MTSM	5.70	5.98	7.21	8.91	8.70	7.94	7.13	6.82
TSRK5DP	1.40	1.16	1.29	1.38	1.15	1.23	1.29	1.24
TSRK8VR	1.78	1.60	1.60	1.55	1.66	1.64	1.51	1.60
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	0.17	0.11	0.11	0.16	0.16	0.11	0.11	0.11
MTSM	7.63	6.53	5.80	5.69	6.48	5.89	6.16	5.23
TSRK5DP	1.38	1.23	1.24	1.42	1.45	1.38	1.47	1.29
TSRK8VR	1.81	2.24	1.86	1.73	1.70	1.74	1.64	1.63

Table C.1: Average time, $S = 128000$, heat equation, three-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**144.06**	**146.86**	**141.65**	**133.78**	**126.94**	**118.04**	**111.95**
MTSM	**100.00**	**70.25**	**56.53**	44.25	37.33	29.28	24.96	19.42
TSRK5DP	**100.00**	**87.14**	**78.65**	**68.21**	**60.67**	**53.40**	**51.59**	44.43
TSRK8VR	**100.00**	**92.89**	**83.86**	**68.73**	**66.64**	**64.78**	**58.97**	**54.21**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**105.77**	**95.91**	**74.47**	**91.39**	**87.37**	**60.55**	**80.00**	**75.90**
MTSM	17.48	14.72	13.61	13.85	11.76	10.49	9.72	9.75
TSRK5DP	39.28	34.34	37.28	34.59	33.38	32.47	29.71	28.03
TSRK8VR	49.33	47.67	46.09	41.91	39.79	33.78	36.35	32.00
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**102.39**	**99.81**	**93.64**	**89.07**	**84.08**	**89.98**	**81.21**	**85.98**
MTSM	7.72	6.96	5.47	4.20	4.10	4.29	4.57	4.58
TSRK5DP	23.53	26.67	22.77	20.29	23.09	20.61	18.79	18.74
TSRK8VR	29.84	31.41	29.83	29.16	25.88	25.04	25.98	23.56
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**53.22**	**82.25**	**78.84**	**52.67**	49.98	**70.18**	**69.39**	**64.31**
MTSM	3.93	4.41	4.78	4.70	3.98	4.24	3.92	4.47
TSRK5DP	16.19	17.46	16.65	14.04	13.28	13.45	12.29	13.49
TSRK8VR	19.98	15.56	18.04	18.67	18.33	17.29	17.81	17.38

Table C.2: Efficiency, $S = 128000$, heat equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.88	4.41	5.67	6.69	7.62	8.26	8.96
MTSM	1.00	1.41	1.70	1.77	1.87	1.76	1.75	1.55
TSRK5DP	1.00	1.74	2.36	2.73	3.03	3.20	3.61	3.55
TSRK8VR	1.00	1.86	2.52	2.75	3.33	3.89	4.13	4.34
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	9.52	9.59	8.19	10.97	11.36	8.48	12.00	12.14
MTSM	1.57	1.47	1.50	1.66	1.53	1.47	1.46	1.56
TSRK5DP	3.54	3.43	4.10	4.15	4.34	4.55	4.46	4.48
TSRK8VR	4.44	4.77	5.07	5.03	5.17	4.73	5.45	5.12
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	17.41	17.97	17.79	17.81	17.66	19.80	18.68	20.64
MTSM	1.31	1.25	1.04	0.84	0.86	0.94	1.05	1.10
TSRK5DP	4.00	4.80	4.33	4.06	4.85	4.54	4.32	4.50
TSRK8VR	5.07	5.65	5.67	5.83	5.43	5.51	5.98	5.65
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	13.30	21.39	21.29	14.75	14.49	21.05	21.51	20.58
MTSM	0.98	1.15	1.29	1.32	1.16	1.27	1.22	1.43
TSRK5DP	4.05	4.54	4.50	3.93	3.85	4.03	3.81	4.32
TSRK8VR	4.99	4.04	4.87	5.23	5.31	5.19	5.52	5.56

Table C.3: Speedup, $S = 128000$, heat equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**2.41**	**3.98**	**4.50**	**5.00**	**5.31**	**5.72**	**5.51**	**6.07**
MTSM	0.75	0.60	0.54	0.48	0.46	0.41	0.36	0.33
TSRK8VR	0.62	0.66	0.66	0.62	0.68	0.75	0.71	0.75
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**6.48**	**6.72**	**4.81**	**6.36**	**6.30**	**4.49**	**6.48**	**6.52**
MTSM	0.33	0.32	0.27	0.30	0.26	0.24	0.24	0.26
TSRK8VR	0.78	0.86	0.76	0.75	0.74	0.64	0.76	0.71
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**10.48**	**9.01**	**9.90**	**10.57**	**8.77**	**10.51**	**10.41**	**11.05**
MTSM	0.24	0.19	0.18	0.15	0.13	0.16	0.18	0.18
TSRK8VR	0.78	0.73	0.81	0.89	0.69	0.75	0.85	0.78
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**7.91**	**11.34**	**11.40**	**9.03**	**9.06**	**12.56**	**13.59**	**11.48**
MTSM	0.18	0.19	0.21	0.25	0.22	0.24	0.24	0.25
TSRK8VR	0.76	0.55	0.67	0.82	0.85	0.79	0.89	0.80

Table C.4: Speedup against TSRK5DP, $S = 128000$, heat equation, three-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A_{PRECALC}}$ nnz	$\mathbf{A_{PRECALC}}$ nnz [%]
5.12	1.02	1.02	383995	$2.34e^{-03}$	6527299	$3.98e^{-02}$

Table C.5: Characteristics of input data, $S = 128000$, heat equation, three-point central difference formula.

C.1.2 $S = 256000$, three-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	8.27	2.42	1.45	0.91	0.66	0.54	0.51	0.43
MTSM	13.17	8.46	6.45	5.76	5.36	5.05	5.29	5.18
TSRK5DP	11.63	5.83	4.13	3.40	2.85	2.53	2.30	2.31
TSRK8VR	19.50	9.61	6.54	5.01	4.17	3.71	3.28	3.07
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	0.39	0.36	0.34	0.35	0.31	0.29	0.28	0.27
MTSM	5.79	5.55	5.62	5.09	5.38	5.95	5.73	5.43
TSRK5DP	2.09	1.92	1.87	1.83	1.84	1.80	1.61	1.55
TSRK8VR	2.92	2.80	2.57	2.48	2.41	2.42	2.36	2.14
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	0.21	0.25	0.19	0.25	0.18	0.23	0.18	0.21
MTSM	5.78	5.87	5.92	5.85	6.70	6.43	6.18	5.49
TSRK5DP	1.72	1.62	1.60	1.55	1.52	1.52	1.67	1.46
TSRK8VR	2.06	2.06	2.15	2.13	1.99	1.93	2.01	1.97
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	0.16	0.16	0.19	0.15	0.15	0.17	0.15	0.15
MTSM	5.91	7.11	6.35	6.47	6.24	6.35	8.90	6.49
TSRK5DP	1.42	1.44	1.38	1.35	1.47	1.45	1.40	1.49
TSRK8VR	2.04	1.86	1.85	1.74	1.73	1.89	1.70	1.66

Table C.6: Average time, $S = 256000$, heat equation, three-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**170.99**	**190.49**	**227.43**	**252.08**	**254.99**	**230.85**	**241.36**
MTSM	**100.00**	**77.81**	**68.12**	**57.22**	49.15	43.48	35.56	31.80
TSRK5DP	**100.00**	**99.79**	**93.93**	**85.54**	**81.49**	**76.66**	**72.25**	**62.81**
TSRK8VR	**100.00**	**101.41**	**99.33**	**97.36**	**93.50**	**87.50**	**85.05**	**79.29**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**238.06**	**228.55**	**220.09**	**195.49**	**205.44**	**203.59**	**199.15**	**191.99**
MTSM	25.26	23.74	21.30	21.56	18.85	15.81	15.33	15.17
TSRK5DP	**61.76**	**60.48**	**56.40**	**52.91**	48.64	46.15	48.27	47.03
TSRK8VR	**74.21**	**69.64**	**69.01**	**65.64**	**62.17**	**57.50**	**55.05**	**56.82**
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**230.36**	**185.14**	**224.17**	**164.56**	**214.90**	**160.58**	**205.24**	**161.05**
MTSM	13.40	12.47	11.72	11.26	9.36	9.31	9.26	10.00
TSRK5DP	39.84	39.93	38.19	37.60	36.45	34.87	30.31	33.19
TSRK8VR	**55.60**	**52.50**	47.76	45.78	46.71	46.04	42.24	41.28
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**201.64**	**199.06**	**157.33**	**191.19**	**186.90**	**163.83**	**177.14**	**176.99**
MTSM	8.92	7.12	7.68	7.27	7.28	6.92	4.77	6.34
TSRK5DP	32.67	30.95	31.31	30.79	27.19	26.66	26.77	24.44
TSRK8VR	38.30	40.36	39.09	39.97	38.94	34.36	36.99	36.70

Table C.7: Efficiency, $S = 256000$, heat equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	3.42	5.71	9.10	12.60	15.30	16.16	19.31
MTSM	1.00	1.56	2.04	2.29	2.46	2.61	2.49	2.54
TSRK5DP	1.00	2.00	2.82	3.42	4.07	4.60	5.06	5.02
TSRK8VR	1.00	2.03	2.98	3.89	4.67	5.25	5.95	6.34
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	21.43	22.86	24.21	23.46	26.71	28.50	29.87	30.72
MTSM	2.27	2.37	2.34	2.59	2.45	2.21	2.30	2.43
TSRK5DP	5.56	6.05	6.20	6.35	6.32	6.46	7.24	7.52
TSRK8VR	6.68	6.96	7.59	7.88	8.08	8.05	8.26	9.09
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	39.16	33.32	42.59	32.91	45.13	35.33	47.21	38.65
MTSM	2.28	2.24	2.23	2.25	1.97	2.05	2.13	2.40
TSRK5DP	6.77	7.19	7.26	7.52	7.65	7.67	6.97	7.96
TSRK8VR	9.45	9.45	9.07	9.16	9.81	10.13	9.72	9.91
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	50.41	51.76	42.48	53.53	54.20	49.15	54.91	56.64
MTSM	2.23	1.85	2.07	2.03	2.11	2.08	1.48	2.03
TSRK5DP	8.17	8.05	8.45	8.62	7.89	8.00	8.30	7.82
TSRK8VR	9.58	10.49	10.55	11.19	11.29	10.31	11.47	11.74

Table C.8: Speedup, $S = 256000$, heat equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**1.41**	**2.41**	**2.85**	**3.74**	**4.35**	**4.68**	**4.49**	**5.40**
MTSM	0.88	0.69	0.64	0.59	0.53	0.50	0.43	0.45
TSRK8VR	0.60	0.61	0.63	0.68	0.68	0.68	0.70	0.75
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**5.42**	**5.31**	**5.49**	**5.19**	**5.94**	**6.20**	**5.80**	**5.74**
MTSM	0.36	0.35	0.33	0.36	0.34	0.30	0.28	0.28
TSRK8VR	0.72	0.69	0.73	0.74	0.76	0.74	0.68	0.72
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**8.13**	**6.52**	**8.25**	**6.15**	**8.29**	**6.47**	**9.52**	**6.82**
MTSM	0.30	0.28	0.27	0.26	0.23	0.24	0.27	0.27
TSRK8VR	0.83	0.78	0.75	0.73	0.76	0.79	0.83	0.74
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**8.68**	**9.04**	**7.06**	**8.73**	**9.66**	**8.64**	**9.30**	**10.18**
MTSM	0.24	0.20	0.22	0.21	0.24	0.23	0.16	0.23
TSRK8VR	0.70	0.78	0.74	0.77	0.85	0.77	0.82	0.90

Table C.9: Speedup against TSRK5DP, $S = 256000$, heat equation, three-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz [%]
10.24	2.05	2.05	767995	$1.17e^{-03}$	13055299	$1.99e^{-02}$

Table C.10: Characteristics of input data, $S = 256000$, heat equation, three-point central difference formula.

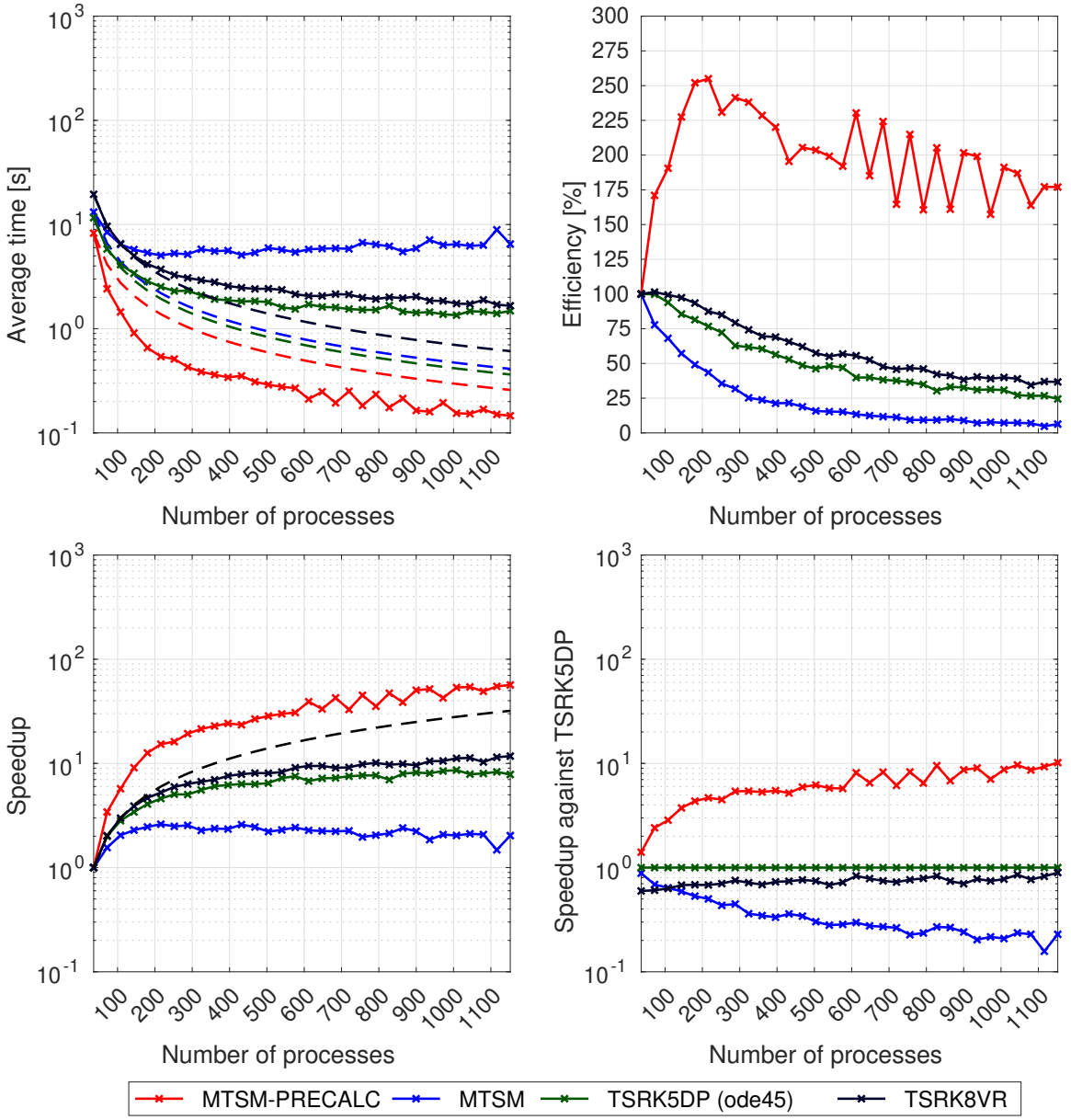


Figure C.1: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 256000$, heat equation, three-point central difference formula.

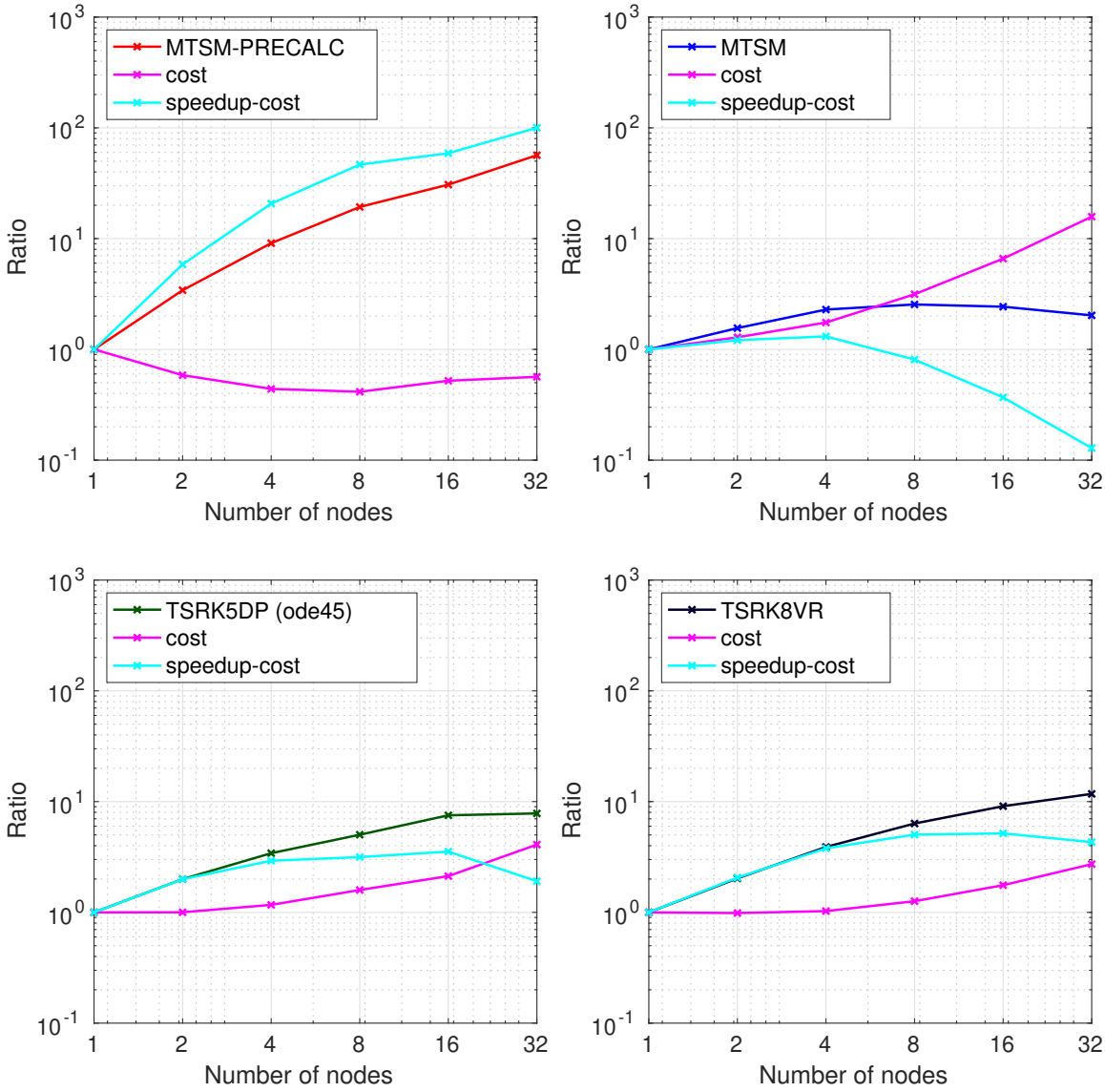


Figure C.2: Parallel cost ratio and speedup-cost ratio, $S = 256000$, heat equation, three-point central difference formula.

C.1.3 $S = 512000$, three-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	14.12	6.38	3.74	2.51	1.84	1.41	1.14	0.88
MTSM	25.30	13.93	10.26	8.98	7.57	7.05	6.67	6.65
TSRK5DP	32.04	11.53	7.57	6.21	5.22	4.32	3.73	3.37
TSRK8VR	59.47	20.29	12.42	9.52	8.28	6.69	5.67	5.07
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	0.76	0.66	0.61	0.59	0.51	0.54	0.46	0.47
MTSM	6.97	6.88	6.59	6.05	6.01	6.37	6.75	5.95
TSRK5DP	3.13	3.01	2.92	2.92	2.58	2.40	2.41	2.34
TSRK8VR	4.66	4.37	4.12	3.76	3.62	3.41	3.33	3.31
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	0.37	0.35	0.34	0.32	0.32	0.35	0.29	0.28
MTSM	6.40	6.34	6.23	6.59	6.81	7.40	6.49	6.02
TSRK5DP	2.08	2.11	2.21	2.09	2.01	2.00	2.13	2.05
TSRK8VR	3.19	3.07	2.92	2.77	2.87	2.85	2.76	2.51
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	0.28	0.27	0.25	0.25	0.24	0.34	0.23	0.23
MTSM	6.31	6.98	6.59	6.61	6.52	6.87	7.34	6.40
TSRK5DP	1.99	2.05	1.82	1.94	1.80	1.74	1.78	1.72
TSRK8VR	2.52	2.57	3.27	2.54	2.51	2.56	2.50	2.69

Table C.11: Average time, $S = 512000$, heat equation, three-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**110.59**	**125.79**	**140.84**	**153.73**	**166.81**	**176.70**	**200.27**
MTSM	**100.00**	**90.83**	**82.19**	**70.45**	**66.81**	**59.79**	**54.19**	47.56
TSRK5DP	**100.00**	**138.98**	**141.02**	**128.99**	**122.67**	**123.72**	**122.78**	**118.75**
TSRK8VR	**100.00**	**146.52**	**159.56**	**156.22**	**143.62**	**148.07**	**149.94**	**146.60**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**207.40**	**212.57**	**209.68**	**199.22**	**211.02**	**185.17**	**205.93**	**189.67**
MTSM	40.33	36.80	34.88	34.87	32.37	28.39	24.99	26.58
TSRK5DP	**113.90**	**106.48**	**99.83**	**91.28**	**95.54**	**95.25**	**88.51**	**85.74**
TSRK8VR	**141.85**	**136.19**	**131.10**	**131.64**	**126.42**	**124.62**	**119.12**	**112.17**
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**223.75**	**223.94**	**218.97**	**219.38**	**209.74**	**180.97**	**214.62**	**212.78**
MTSM	23.26	22.17	21.37	19.20	17.68	15.54	16.95	17.50
TSRK5DP	**90.80**	**84.31**	**76.17**	**76.59**	**76.04**	**72.93**	**65.47**	**65.24**
TSRK8VR	**109.60**	**107.68**	**107.08**	**107.40**	**98.65**	**94.79**	**93.61**	**98.78**
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**204.06**	**204.38**	**205.76**	**202.58**	**199.64**	**137.57**	**197.13**	**192.54**
MTSM	16.04	13.94	14.23	13.66	13.37	12.28	11.12	12.36
TSRK5DP	**64.26**	**60.16**	**65.02**	**58.91**	**61.24**	**61.40**	**58.06**	**58.08**
TSRK8VR	**94.47**	**89.13**	**67.38**	**83.65**	**81.61**	**77.55**	**76.68**	**69.02**

Table C.12: Efficiency, $S = 512000$, heat equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.21	3.77	5.63	7.69	10.01	12.37	16.02
MTSM	1.00	1.82	2.47	2.82	3.34	3.59	3.79	3.80
TSRK5DP	1.00	2.78	4.23	5.16	6.13	7.42	8.59	9.50
TSRK8VR	1.00	2.93	4.79	6.25	7.18	8.88	10.50	11.73
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	18.67	21.26	23.07	23.91	27.43	25.92	30.89	30.35
MTSM	3.63	3.68	3.84	4.18	4.21	3.97	3.75	4.25
TSRK5DP	10.25	10.65	10.98	10.95	12.42	13.33	13.28	13.72
TSRK8VR	12.77	13.62	14.42	15.80	16.43	17.45	17.87	17.95
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	38.04	40.31	41.60	43.88	44.05	39.81	49.36	51.07
MTSM	3.95	3.99	4.06	3.84	3.71	3.42	3.90	4.20
TSRK5DP	15.44	15.18	14.47	15.32	15.97	16.05	15.06	15.66
TSRK8VR	18.63	19.38	20.35	21.48	20.72	20.85	21.53	23.71
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	51.02	53.14	55.55	56.72	57.90	41.27	61.11	61.61
MTSM	4.01	3.62	3.84	3.83	3.88	3.68	3.45	3.96
TSRK5DP	16.07	15.64	17.56	16.49	17.76	18.42	18.00	18.59
TSRK8VR	23.62	23.17	18.19	23.42	23.67	23.27	23.77	22.09

Table C.13: Speedup, $S = 512000$, heat equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**2.27**	**1.81**	**2.02**	**2.48**	**2.84**	**3.06**	**3.26**	**3.83**
MTSM	**1.27**	0.83	0.74	0.69	0.69	0.61	0.56	0.51
TSRK8VR	0.54	0.57	0.61	0.65	0.63	0.64	0.66	0.67
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**4.13**	**4.53**	**4.76**	**4.95**	**5.01**	**4.41**	**5.28**	**5.02**
MTSM	0.45	0.44	0.44	0.48	0.43	0.38	0.36	0.39
TSRK8VR	0.67	0.69	0.71	0.78	0.71	0.70	0.73	0.70
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**5.59**	**6.03**	**6.52**	**6.50**	**6.26**	**5.63**	**7.44**	**7.40**
MTSM	0.32	0.33	0.36	0.32	0.29	0.27	0.33	0.34
TSRK8VR	0.65	0.69	0.76	0.76	0.70	0.70	0.77	0.82
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**7.20**	**7.71**	**7.18**	**7.80**	**7.40**	**5.08**	**7.70**	**7.52**
MTSM	0.32	0.29	0.28	0.29	0.28	0.25	0.24	0.27
TSRK8VR	0.79	0.80	0.56	0.76	0.72	0.68	0.71	0.64

Table C.14: Speedup against TSRK5DP, $S = 512000$, heat equation, three-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz [%]
20.48	4.10	4.10	1535995	$5.86e^{-04}$	26111299	$9.96e^{-03}$

Table C.15: Characteristics of input data, $S = 512000$, heat equation, three-point central difference formula.

C.1.4 $S = 1024000$, three-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	30.51	14.49	9.01	6.35	4.77	3.78	3.10	2.51
MTSM	53.73	26.45	18.15	14.27	12.37	11.10	10.35	9.52
TSRK5DP	97.70	33.48	17.22	12.40	9.36	7.77	6.98	6.15
TSRK8VR	163.04	62.33	34.23	20.95	15.59	13.48	11.11	9.91
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	2.13	1.86	1.74	1.49	1.38	1.15	1.02	0.94
MTSM	9.69	10.11	9.24	8.05	8.29	8.68	8.11	7.57
TSRK5DP	6.11	5.29	4.84	4.39	4.25	3.98	5.65	3.67
TSRK8VR	9.79	9.22	7.69	6.76	6.30	6.02	6.17	5.47
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	0.75	0.72	0.68	0.80	0.61	0.62	0.59	0.52
MTSM	7.92	7.87	8.39	7.95	8.00	7.85	7.82	7.47
TSRK5DP	3.61	3.29	3.28	3.13	3.14	3.70	2.97	3.07
TSRK8VR	5.32	4.90	7.24	4.66	4.49	4.48	4.21	3.93
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	0.49	0.47	0.46	0.57	0.44	0.43	0.47	0.44
MTSM	7.90	7.93	7.69	8.18	8.04	7.66	7.84	7.29
TSRK5DP	2.82	2.86	2.82	2.73	3.39	3.46	2.66	2.34
TSRK8VR	4.06	4.04	4.55	3.53	3.70	3.56	3.54	3.37

Table C.16: Average time, $S = 1024000$, heat equation, three-point central difference formula

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**105.32**	**112.87**	**120.16**	**127.88**	**134.46**	**140.79**	**151.70**
MTSM	**100.00**	**101.55**	**98.66**	**94.09**	**86.88**	**80.69**	**74.19**	**70.56**
TSRK5DP	**100.00**	**145.91**	**189.13**	**197.04**	**208.65**	**209.65**	**199.86**	**198.63**
TSRK8VR	**100.00**	**130.78**	**158.76**	**194.52**	**209.16**	**201.52**	**209.71**	**205.57**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**159.27**	**163.68**	**158.99**	**170.76**	**170.60**	**189.40**	**198.99**	**202.25**
MTSM	**61.62**	**53.16**	**52.89**	**55.64**	49.85	44.23	44.19	44.34
TSRK5DP	**177.61**	**184.79**	**183.69**	**185.38**	**176.82**	**175.14**	**115.22**	**166.60**
TSRK8VR	**185.06**	**176.92**	**192.62**	**201.05**	**198.92**	**193.43**	**176.24**	**186.28**
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**238.39**	**235.29**	**235.51**	**191.76**	**238.54**	**225.05**	**223.24**	**244.83**
MTSM	39.92	37.93	33.71	33.81	31.99	31.11	29.88	29.97
TSRK5DP	**159.28**	**164.96**	**156.73**	**156.16**	**148.07**	**119.98**	**142.96**	**132.70**
TSRK8VR	**180.16**	**184.97**	**118.44**	**174.91**	**172.83**	**165.55**	**168.26**	**173.08**
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**247.77**	**247.32**	**246.18**	**190.11**	**240.78**	**236.56**	**210.94**	**219.01**
MTSM	27.22	26.07	25.89	23.47	23.03	23.38	22.10	23.03
TSRK5DP	**138.58**	**131.36**	**128.46**	**127.74**	**99.49**	**94.10**	**118.37**	**130.39**
TSRK8VR	**160.45**	**155.24**	**132.60**	**164.91**	**151.92**	**152.76**	**148.54**	**150.97**

Table C.17: Efficiency, $S = 1024000$, heat equation, three-point central difference formula

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.11	3.39	4.81	6.39	8.07	9.86	12.14
MTSM	1.00	2.03	2.96	3.76	4.34	4.84	5.19	5.64
TSRK5DP	1.00	2.92	5.67	7.88	10.43	12.58	13.99	15.89
TSRK8VR	1.00	2.62	4.76	7.78	10.46	12.09	14.68	16.45
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	14.33	16.37	17.49	20.49	22.18	26.52	29.85	32.36
MTSM	5.55	5.32	5.82	6.68	6.48	6.19	6.63	7.09
TSRK5DP	15.98	18.48	20.21	22.25	22.99	24.52	17.28	26.66
TSRK8VR	16.66	17.69	21.19	24.13	25.86	27.08	26.44	29.81
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	40.53	42.35	44.75	38.35	50.09	49.51	51.34	58.76
MTSM	6.79	6.83	6.40	6.76	6.72	6.84	6.87	7.19
TSRK5DP	27.08	29.69	29.78	31.23	31.09	26.40	32.88	31.85
TSRK8VR	30.63	33.29	22.50	34.98	36.29	36.42	38.70	41.54
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	61.94	64.30	66.47	53.23	69.83	70.97	65.39	70.08
MTSM	6.80	6.78	6.99	6.57	6.68	7.01	6.85	7.37
TSRK5DP	34.64	34.15	34.69	35.77	28.85	28.23	36.70	41.73
TSRK8VR	40.11	40.36	35.80	46.18	44.06	45.83	46.05	48.31

Table C.18: Speedup, $S = 1024000$, heat equation, three-point central difference formula

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**3.20**	**2.31**	**1.91**	**1.95**	**1.96**	**2.05**	**2.26**	**2.45**
MTSM	**1.82**	**1.27**	0.95	0.87	0.76	0.70	0.68	0.65
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	0.60	0.54	0.50	0.59	0.60	0.58	0.63	0.62
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**2.87**	**2.84**	**2.77**	**2.95**	**3.09**	**3.46**	**5.53**	**3.89**
MTSM	0.63	0.52	0.52	0.55	0.51	0.46	0.70	0.48
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	0.62	0.57	0.63	0.65	0.67	0.66	0.92	0.67
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**4.79**	**4.57**	**4.81**	**3.93**	**5.16**	**6.01**	**5.00**	**5.91**
MTSM	0.46	0.42	0.39	0.39	0.39	0.47	0.38	0.41
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	0.68	0.67	0.45	0.67	0.70	0.83	0.71	0.78
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**5.72**	**6.03**	**6.14**	**4.77**	**7.75**	**8.05**	**5.71**	**5.38**
MTSM	0.36	0.36	0.37	0.33	0.42	0.45	0.34	0.32
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	0.69	0.71	0.62	0.77	0.92	0.97	0.75	0.69

Table C.19: Speedup against TSRK5DP, $S = 1024000$, heat equation, three-point central difference formula

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz [%]
40.96	8.19	8.19	3071995	$2.93e^{-04}$	52223299	$4.98e^{-03}$

Table C.20: Characteristics of input data, $S = 1024000$, heat equation, three-point central difference formula

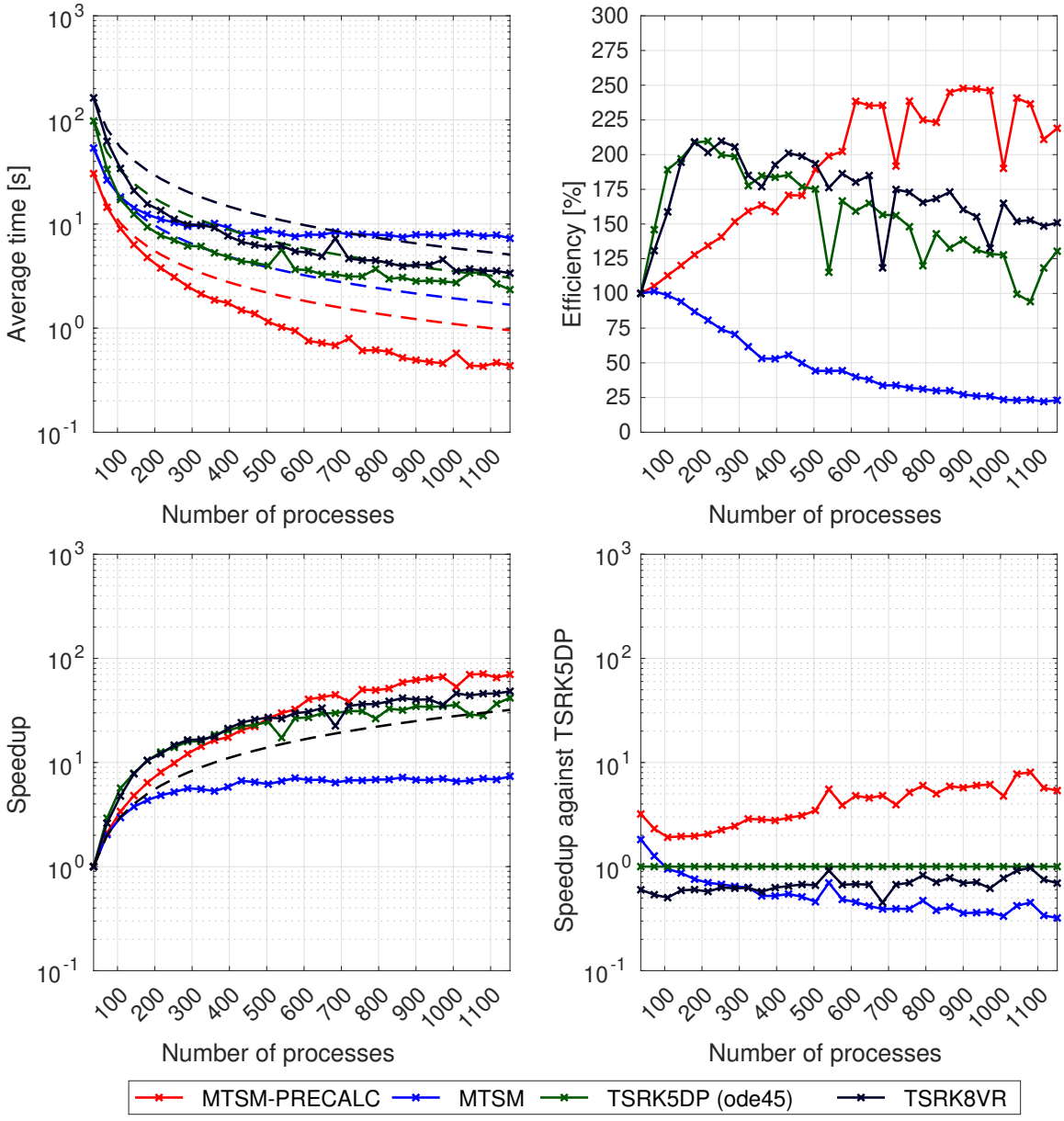


Figure C.3: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 1024000$, heat equation, three-point central difference formula.

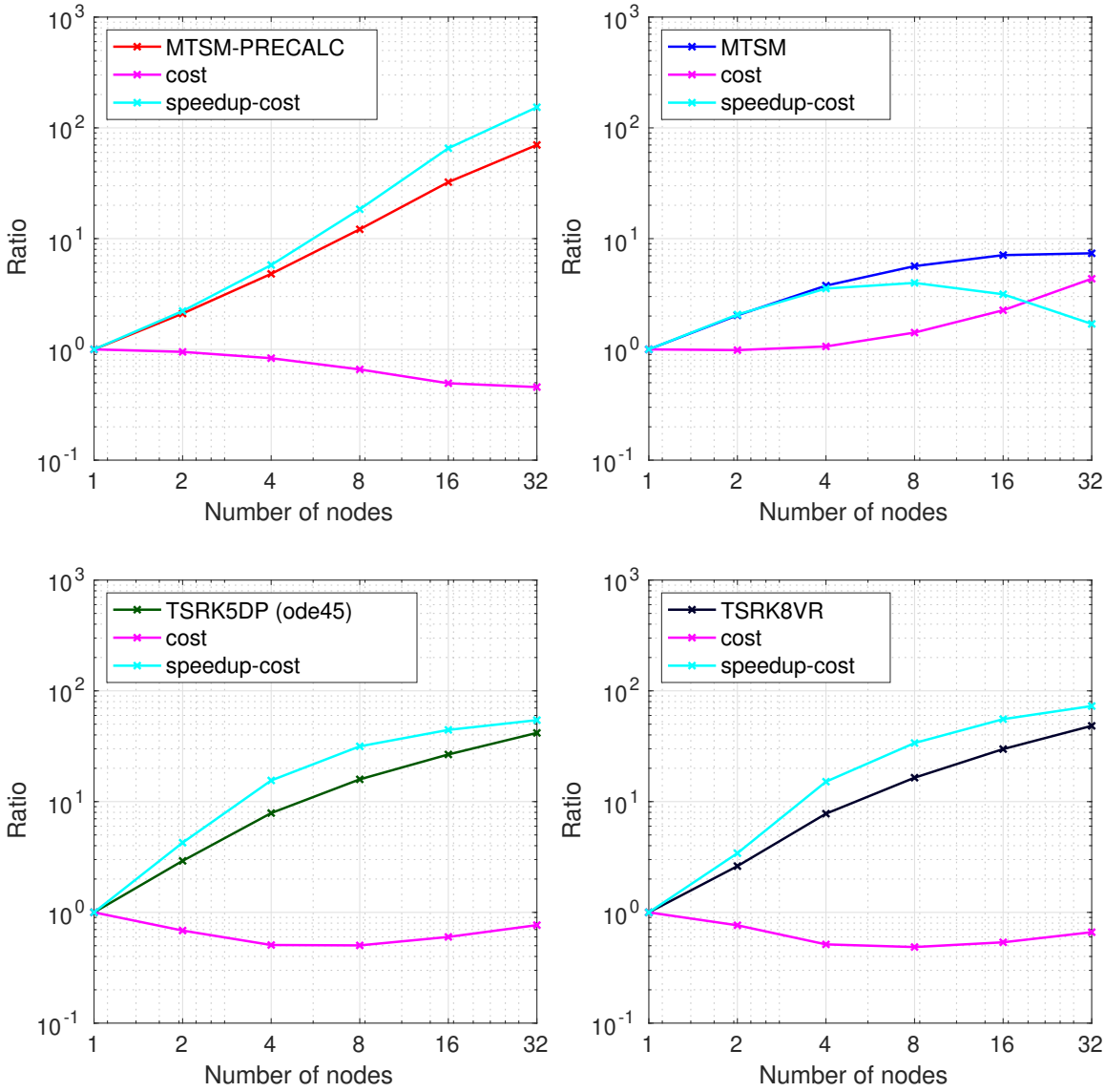


Figure C.4: Parallel cost ratio and speedup-cost ratio, $S = 1024000$, heat equation, three-point central difference formula.

C.1.5 $S = 2048000$, three-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	62.53	30.74	19.68	14.60	11.13	9.05	7.55	6.45
MTSM	137.94	55.48	36.29	27.46	21.93	19.52	16.94	15.89
TSRK5DP	240.86	100.26	52.41	33.35	23.43	17.63	14.74	12.57
TSRK8VR	389.80	166.96	94.19	62.15	46.00	35.27	26.74	22.42
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	5.72	4.95	4.47	3.90	3.72	3.29	2.90	2.62
MTSM	15.30	14.01	13.27	12.73	12.05	11.90	11.69	10.40
TSRK5DP	10.92	9.72	9.37	8.35	7.76	7.36	7.17	6.73
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	2.42	2.18	2.00	1.98	1.79	1.67	1.60	1.58
MTSM	10.87	11.26	10.86	10.19	10.47	10.34	9.83	9.55
TSRK5DP	6.29	5.92	6.36	5.73	5.07	5.17	4.88	5.74
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	1.44	1.59	1.32	1.29	1.08	1.05	0.94	0.94
MTSM	10.45	9.79	11.68	9.68	9.76	9.28	9.55	9.65
TSRK5DP	4.95	4.53	4.32	4.21	4.98	5.09	3.93	4.02
TSRK8VR	-	-	-	-	-	-	-	-

Table C.21: Average time, $S = 2048000$, heat equation, three-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**101.71**	**105.92**	**107.09**	**112.36**	**115.11**	**118.30**	**121.11**
MTSM	**100.00**	**124.32**	**126.70**	**125.59**	**125.79**	**117.78**	**116.35**	**108.48**
TSRK5DP	**100.00**	**120.12**	**153.20**	**180.57**	**205.57**	**227.73**	**233.45**	**239.59**
TSRK8VR	**100.00**	**116.73**	**137.94**	**156.80**	**169.47**	**184.18**	**208.26**	**217.31**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**121.46**	**126.33**	**127.20**	**133.56**	**129.38**	**135.91**	**143.64**	**149.06**
MTSM	**100.15**	**98.44**	**94.51**	**90.31**	**88.06**	**82.81**	**78.65**	**82.87**
TSRK5DP	**245.11**	**247.82**	**233.79**	**240.52**	**238.88**	**233.64**	**224.07**	**223.62**
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**152.18**	**159.45**	**164.56**	**157.86**	**166.68**	**170.28**	**169.85**	**164.67**
MTSM	**74.62**	**68.05**	**66.83**	**67.69**	**62.74**	**60.63**	**60.99**	**60.20**
TSRK5DP	**225.28**	**225.91**	**199.41**	**210.18**	**226.15**	**211.66**	**214.76**	**174.93**
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**173.96**	**150.88**	**175.20**	**173.31**	**199.50**	**198.77**	**214.62**	**207.10**
MTSM	**52.78**	**54.18**	43.74	**50.90**	48.74	49.53	46.59	44.68
TSRK5DP	**194.56**	**204.38**	**206.49**	**204.23**	**166.65**	**157.61**	**197.93**	**187.16**
TSRK8VR	-	-	-	-	-	-	-	-

Table C.22: Efficiency, $S = 2048000$, heat equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.03	3.18	4.28	5.62	6.91	8.28	9.69
MTSM	1.00	2.49	3.80	5.02	6.29	7.07	8.14	8.68
TSRK5DP	1.00	2.40	4.60	7.22	10.28	13.66	16.34	19.17
TSRK8VR	1.00	2.33	4.14	6.27	8.47	11.05	14.58	17.38
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	10.93	12.63	13.99	16.03	16.82	19.03	21.55	23.85
MTSM	9.01	9.84	10.40	10.84	11.45	11.59	11.80	13.26
TSRK5DP	22.06	24.78	25.72	28.86	31.05	32.71	33.61	35.78
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	25.87	28.70	31.27	31.57	35.00	37.46	39.07	39.52
MTSM	12.69	12.25	12.70	13.54	13.18	13.34	14.03	14.45
TSRK5DP	38.30	40.66	37.89	42.04	47.49	46.57	49.39	41.98
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	43.49	39.23	47.30	48.53	57.85	59.63	66.53	66.27
MTSM	13.19	14.09	11.81	14.25	14.14	14.86	14.44	14.30
TSRK5DP	48.64	53.14	55.75	57.18	48.33	47.28	61.36	59.89
TSRK8VR	-	-	-	-	-	-	-	-

Table C.23: Speedup, $S = 2048000$, heat equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**3.85**	**3.26**	**2.66**	**2.28**	**2.11**	**1.95**	**1.95**	**1.95**
MTSM	**1.75**	**1.81**	**1.44**	**1.21**	**1.07**	0.90	0.87	0.79
TSRK8VR	0.62	0.60	0.56	0.54	0.51	0.50	0.55	0.56
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**1.91**	**1.96**	**2.10**	**2.14**	**2.09**	**2.24**	**2.47**	**2.57**
MTSM	0.71	0.69	0.71	0.66	0.64	0.62	0.61	0.65
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**2.60**	**2.72**	**3.18**	**2.89**	**2.84**	**3.10**	**3.05**	**3.63**
MTSM	0.58	0.53	0.59	0.56	0.48	0.50	0.50	0.60
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**3.44**	**2.84**	**3.27**	**3.27**	**4.61**	**4.86**	**4.18**	**4.26**
MTSM	0.47	0.46	0.37	0.44	0.51	0.55	0.41	0.42
TSRK8VR	-	-	-	-	-	-	-	-

Table C.24: Speedup against TSRK5DP, $S = 2048000$, heat equation, three-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A_{PRECALC}}$ nnz	$\mathbf{A_{PRECALC}}$ nnz [%]
81.92	16.38	16.38	6143995	$1.46e^{-04}$	104447299	$2.49e^{-03}$

Table C.25: Characteristics of input data, $S = 2048000$, heat equation, three-point central difference formula

C.2 Heat equation – five-point central difference formula

Subsections C.2.1–C.2.5 show numerical results for $S = 128000$, $S = 256000$, $S = 512000$, $S = 1024000$, and $S = 2048000$, respectively.

C.2.1 $S = 128000$, five-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	2.35	0.80	0.53	0.41	0.37	0.31	0.28	0.26
MTSM	5.34	3.74	3.18	2.92	3.10	2.80	2.98	3.09
TSRK5DP	4.17	2.39	1.79	1.49	1.42	1.24	1.40	1.08
TSRK8VR	6.78	3.61	2.72	2.21	1.91	1.95	1.61	1.54
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	0.24	0.27	0.22	0.21	0.21	0.21	0.21	0.22
MTSM	3.51	3.48	3.57	3.05	3.31	3.42	3.90	3.53
TSRK5DP	1.27	1.09	1.22	1.09	0.96	0.93	0.93	1.04
TSRK8VR	1.60	1.43	1.44	1.36	1.34	1.33	1.26	1.22
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	0.13	0.13	0.13	0.12	0.12	0.12	0.12	0.12
MTSM	3.90	4.32	5.15	6.24	6.31	5.78	5.28	4.64
TSRK5DP	0.93	0.89	0.95	0.90	0.99	1.08	1.01	1.01
TSRK8VR	1.22	1.16	1.29	1.19	1.13	1.11	1.15	1.08
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	0.12	0.11	0.11	0.14	0.11	0.11	0.11	0.12
MTSM	5.09	4.60	4.51	4.50	4.75	4.44	4.25	4.06
TSRK5DP	0.86	0.96	0.93	0.87	1.05	0.87	0.95	0.97
TSRK8VR	1.19	1.21	1.10	1.07	1.19	1.08	1.05	1.05

Table C.26: Average time, $S = 128000$, heat equation, five-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**147.68**	**148.80**	**144.28**	**128.08**	**125.71**	**119.78**	**113.49**
MTSM	**100.00**	**71.42**	**56.06**	45.73	34.52	31.84	25.66	21.60
TSRK5DP	**100.00**	**87.41**	**77.78**	**69.79**	**58.77**	**55.90**	42.53	48.25
TSRK8VR	**100.00**	**93.97**	**83.07**	**76.75**	**71.12**	**58.02**	**60.21**	**55.00**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**107.95**	**85.99**	**97.11**	**92.40**	**87.75**	**79.69**	**74.20**	**68.09**
MTSM	16.93	15.35	13.59	14.60	12.43	11.15	9.14	9.45
TSRK5DP	36.63	38.26	30.97	32.00	33.44	32.05	29.75	24.97
TSRK8VR	47.19	47.29	42.83	41.58	38.82	36.37	35.86	34.85
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**102.70**	**99.72**	**97.59**	**95.18**	**93.19**	**88.15**	**88.70**	**80.65**
MTSM	8.06	6.87	5.46	4.28	4.03	4.20	4.40	4.80
TSRK5DP	26.25	26.15	23.08	23.18	20.15	17.57	17.95	17.13
TSRK8VR	32.69	32.35	27.71	28.39	28.45	27.65	25.71	26.23
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**79.68**	**83.51**	**77.78**	**58.14**	**74.41**	**69.18**	**68.74**	**60.07**
MTSM	4.20	4.47	4.39	4.24	3.88	4.01	4.06	4.11
TSRK5DP	19.33	16.72	16.53	17.03	13.75	16.07	14.20	13.44
TSRK8VR	22.77	21.55	22.84	22.64	19.66	20.84	20.80	20.22

Table C.27: Efficiency, $S = 128000$, heat equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.95	4.46	5.77	6.40	7.54	8.38	9.08
MTSM	1.00	1.43	1.68	1.83	1.73	1.91	1.80	1.73
TSRK5DP	1.00	1.75	2.33	2.79	2.94	3.35	2.98	3.86
TSRK8VR	1.00	1.88	2.49	3.07	3.56	3.48	4.21	4.40
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	9.72	8.60	10.68	11.09	11.41	11.16	11.13	10.89
MTSM	1.52	1.53	1.50	1.75	1.62	1.56	1.37	1.51
TSRK5DP	3.30	3.83	3.41	3.84	4.35	4.49	4.46	4.00
TSRK8VR	4.25	4.73	4.71	4.99	5.05	5.09	5.38	5.58
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	17.46	17.95	18.54	19.04	19.57	19.39	20.40	19.36
MTSM	1.37	1.24	1.04	0.86	0.85	0.92	1.01	1.15
TSRK5DP	4.46	4.71	4.39	4.64	4.23	3.87	4.13	4.11
TSRK8VR	5.56	5.82	5.27	5.68	5.97	6.08	5.91	6.30
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	19.92	21.71	21.00	16.28	21.58	20.75	21.31	19.22
MTSM	1.05	1.16	1.19	1.19	1.13	1.20	1.26	1.31
TSRK5DP	4.83	4.35	4.46	4.77	3.99	4.82	4.40	4.30
TSRK8VR	5.69	5.60	6.17	6.34	5.70	6.25	6.45	6.47

Table C.28: Speedup, $S = 128000$, heat equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**1.78**	**3.00**	**3.40**	**3.67**	**3.87**	**3.99**	**5.00**	**4.18**
MTSM	0.78	0.64	0.56	0.51	0.46	0.44	0.47	0.35
TSRK8VR	0.62	0.66	0.66	0.68	0.74	0.64	0.87	0.70
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**5.23**	**3.99**	**5.57**	**5.13**	**4.66**	**4.41**	**4.43**	**4.84**
MTSM	0.36	0.31	0.34	0.36	0.29	0.27	0.24	0.30
TSRK8VR	0.79	0.76	0.85	0.80	0.71	0.70	0.74	0.86
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**6.95**	**6.77**	**7.51**	**7.29**	**8.21**	**8.91**	**8.77**	**8.36**
MTSM	0.24	0.21	0.18	0.14	0.16	0.19	0.19	0.22
TSRK8VR	0.77	0.76	0.74	0.75	0.87	0.97	0.88	0.94
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**7.32**	**8.87**	**8.36**	**6.06**	**9.61**	**7.64**	**8.59**	**7.93**
MTSM	0.17	0.21	0.21	0.19	0.22	0.19	0.22	0.24
TSRK8VR	0.72	0.79	0.85	0.82	0.88	0.80	0.90	0.93

Table C.29: Speedup against TSRK5DP, $S = 128000$, heat equation, five-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz [%]
5.12	1.02	1.02	383995	$2.34e^{-03}$	6527299	$3.98e^{-02}$

Table C.30: Characteristics of input data, $S = 128000$, heat equation, five-point central difference formula.

C.2.2 $S = 256000$, five-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	6.22	2.34	1.38	0.83	0.65	0.55	0.47	0.42
MTSM	9.23	5.86	4.81	3.96	3.92	3.60	3.51	3.48
TSRK5DP	8.83	4.46	3.25	2.56	2.08	1.87	1.70	1.55
TSRK8VR	15.27	7.25	4.68	3.69	3.12	2.84	2.54	2.28
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	0.39	0.38	0.35	0.44	0.31	0.41	0.35	0.32
MTSM	3.85	3.79	3.76	3.48	3.94	4.09	3.88	3.44
TSRK5DP	1.50	1.58	1.39	1.30	1.61	1.39	1.16	1.29
TSRK8VR	2.12	2.18	1.95	1.79	1.90	1.74	1.63	1.60
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	0.21	0.20	0.19	0.19	0.18	0.18	0.17	0.17
MTSM	3.97	4.02	4.06	4.11	3.94	4.30	4.23	3.79
TSRK5DP	1.13	1.24	1.15	1.17	1.12	1.23	1.19	1.03
TSRK8VR	1.54	1.65	1.50	1.54	2.16	1.56	1.45	1.55
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	0.16	0.16	0.16	0.16	0.15	0.15	0.15	0.14
MTSM	4.35	4.29	4.48	4.20	4.67	4.29	4.41	4.29
TSRK5DP	1.17	1.42	1.32	1.43	1.16	1.04	1.05	1.09
TSRK8VR	1.41	2.04	1.42	1.40	1.41	1.42	1.94	1.44

Table C.31: Average time, $S = 256000$, heat equation, five-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**132.90**	**150.90**	**186.73**	**192.52**	**189.79**	**190.89**	**185.08**
MTSM	**100.00**	**78.83**	**64.03**	**58.21**	47.13	42.79	37.59	33.15
TSRK5DP	**100.00**	**99.00**	**90.66**	**86.19**	**84.98**	**78.65**	**74.12**	**71.02**
TSRK8VR	**100.00**	**105.30**	**108.81**	**103.44**	**97.95**	**89.70**	**85.97**	**83.82**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**177.53**	**165.66**	**163.11**	**118.32**	**156.89**	**108.44**	**117.06**	**123.02**
MTSM	26.63	24.34	22.32	22.09	18.02	16.13	15.84	16.77
TSRK5DP	**65.59**	**55.86**	**57.81**	**56.40**	42.26	45.41	**50.72**	42.94
TSRK8VR	**79.88**	**70.18**	**71.03**	**70.99**	**61.95**	**62.87**	**62.44**	**59.64**
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**171.13**	**171.41**	**168.48**	**162.72**	**161.08**	**157.65**	**156.37**	**150.97**
MTSM	13.66	12.74	11.96	11.23	11.17	9.76	9.50	10.15
TSRK5DP	46.03	39.42	40.24	37.83	37.39	32.73	32.27	35.65
TSRK8VR	**58.53**	**51.49**	**53.47**	49.43	33.64	44.57	45.74	41.08
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**152.15**	**148.23**	**145.81**	**142.58**	**139.30**	**137.81**	**135.73**	**134.17**
MTSM	8.50	8.28	7.63	7.85	6.82	7.18	6.76	6.73
TSRK5DP	30.16	23.90	24.70	22.06	26.17	28.40	27.24	25.40
TSRK8VR	43.18	28.80	39.70	39.01	37.27	35.83	25.45	33.06

Table C.32: Efficiency, $S = 256000$, heat equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.66	4.53	7.47	9.63	11.39	13.36	14.81
MTSM	1.00	1.58	1.92	2.33	2.36	2.57	2.63	2.65
TSRK5DP	1.00	1.98	2.72	3.45	4.25	4.72	5.19	5.68
TSRK8VR	1.00	2.11	3.26	4.14	4.90	5.38	6.02	6.71
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	15.98	16.57	17.94	14.20	20.40	15.18	17.56	19.68
MTSM	2.40	2.43	2.45	2.65	2.34	2.26	2.38	2.68
TSRK5DP	5.90	5.59	6.36	6.77	5.49	6.36	7.61	6.87
TSRK8VR	7.19	7.02	7.81	8.52	8.05	8.80	9.37	9.54
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	29.09	30.85	32.01	32.54	33.83	34.68	35.96	36.23
MTSM	2.32	2.29	2.27	2.25	2.35	2.15	2.18	2.44
TSRK5DP	7.83	7.10	7.65	7.57	7.85	7.20	7.42	8.56
TSRK8VR	9.95	9.27	10.16	9.89	7.06	9.80	10.52	9.86
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	38.04	38.54	39.37	39.92	40.40	41.34	42.07	42.93
MTSM	2.12	2.15	2.06	2.20	1.98	2.15	2.10	2.15
TSRK5DP	7.54	6.21	6.67	6.18	7.59	8.52	8.45	8.13
TSRK8VR	10.79	7.49	10.72	10.92	10.81	10.75	7.89	10.58

Table C.33: Speedup, $S = 256000$, heat equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**1.42**	**1.90**	**2.36**	**3.07**	**3.21**	**3.42**	**3.65**	**3.70**
MTSM	0.96	0.76	0.68	0.65	0.53	0.52	0.49	0.45
TSRK8VR	0.58	0.61	0.69	0.69	0.67	0.66	0.67	0.68
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**3.84**	**4.21**	**4.00**	**2.98**	**5.27**	**3.39**	**3.27**	**4.06**
MTSM	0.39	0.42	0.37	0.37	0.41	0.34	0.30	0.37
TSRK8VR	0.70	0.73	0.71	0.73	0.85	0.80	0.71	0.80
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**5.27**	**6.17**	**5.94**	**6.10**	**6.11**	**6.83**	**6.87**	**6.01**
MTSM	0.28	0.31	0.28	0.28	0.29	0.29	0.28	0.27
TSRK8VR	0.73	0.76	0.77	0.76	0.52	0.79	0.82	0.67
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**7.15**	**8.80**	**8.37**	**9.17**	**7.55**	**6.88**	**7.07**	**7.49**
MTSM	0.27	0.33	0.30	0.34	0.25	0.24	0.24	0.25
TSRK8VR	0.83	0.70	0.93	**1.02**	0.82	0.73	0.54	0.75

Table C.34: Speedup against TSRK5DP, $S = 256000$, heat equation, five-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz [%]
10.24	2.05	2.05	767995	$1.17e^{-03}$	13055299	$1.99e^{-02}$

Table C.35: Characteristics of input data, $S = 256000$, heat equation, five-point central difference formula.

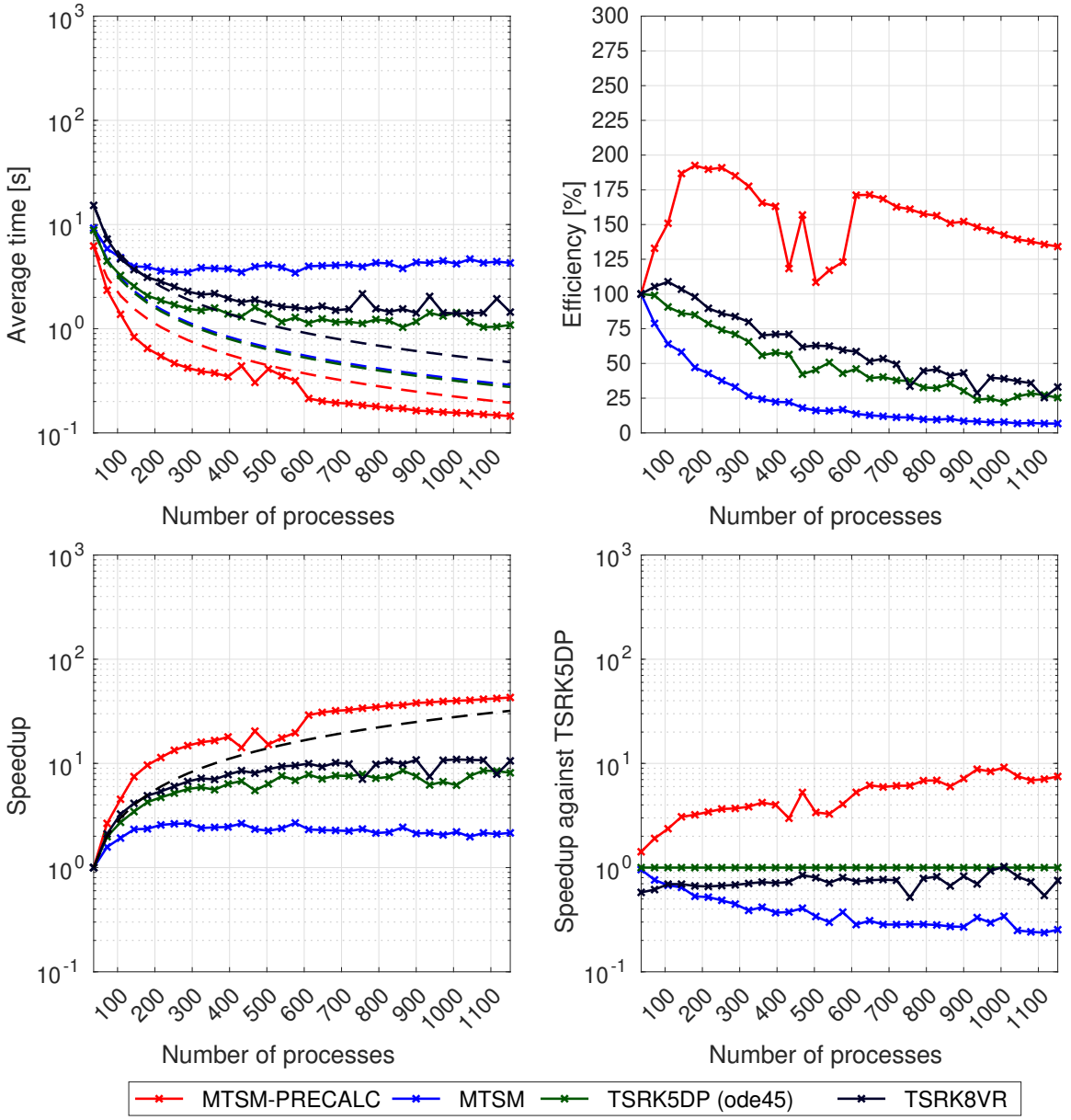


Figure C.5: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 256000$, heat equation, five-point central difference formula.

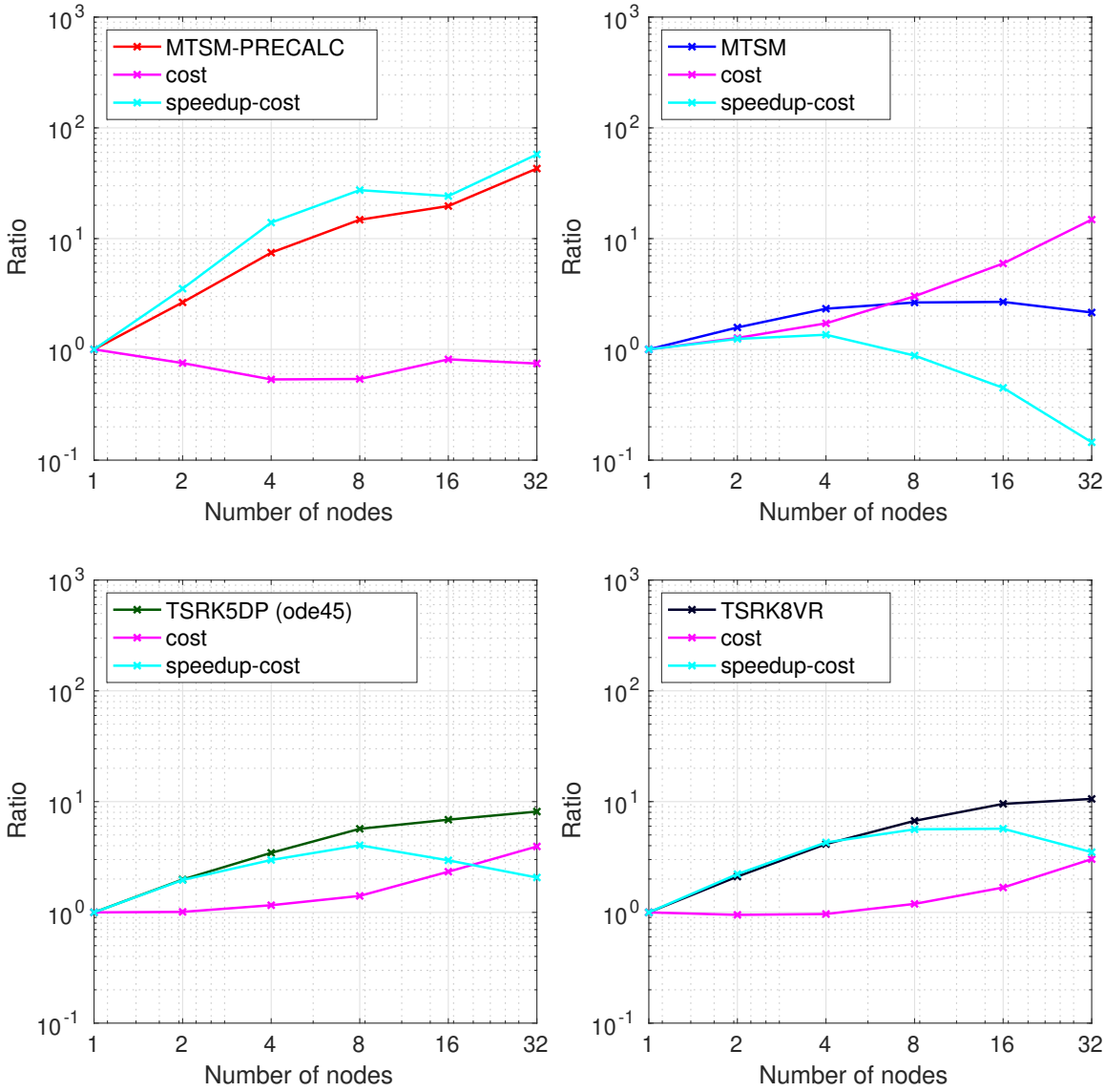


Figure C.6: Parallel cost ratio and speedup-cost ratio, $S = 256000$, heat equation, five-point central difference formula.

C.2.3 $S = 521000$, five-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	29.84	14.00	8.78	6.15	4.82	3.93	3.11	2.61
MTSM	27.21	14.13	10.29	8.72	7.98	7.02	6.65	6.52
TSRK5DP	38.83	12.50	8.03	5.96	5.02	4.22	3.97	3.45
TSRK8VR	69.24	23.85	12.74	9.76	8.11	6.77	5.91	5.15
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	2.22	1.98	1.73	1.56	1.35	1.21	1.05	0.92
MTSM	6.73	6.91	7.14	5.92	6.20	6.26	6.38	5.88
TSRK5DP	3.17	3.06	2.76	2.63	2.48	2.53	2.36	2.74
TSRK8VR	4.64	4.31	4.12	3.89	4.37	3.56	3.37	3.30
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	0.84	0.81	0.66	0.68	0.62	0.58	0.55	0.53
MTSM	6.48	6.45	6.35	6.47	6.67	6.98	6.77	6.11
TSRK5DP	2.40	2.29	2.15	2.01	2.10	2.03	2.10	1.98
TSRK8VR	3.13	3.13	2.96	2.84	2.92	2.89	2.71	2.67
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	0.50	0.53	0.58	0.46	0.46	0.45	0.43	0.42
MTSM	6.29	6.63	6.63	6.95	6.97	7.55	6.85	6.45
TSRK5DP	2.13	2.16	1.96	1.93	1.87	1.87	1.82	1.68
TSRK8VR	2.54	2.69	2.61	2.74	2.55	2.55	2.36	2.42

Table C.36: Average time, $S = 512000$, heat equation, five-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**106.58**	**113.31**	**121.28**	**123.79**	**126.49**	**137.20**	**142.80**
MTSM	**100.00**	**96.31**	**88.16**	**78.01**	**68.24**	**64.57**	**58.42**	**52.13**
TSRK5DP	**100.00**	**155.31**	**161.14**	**162.80**	**154.66**	**153.18**	**139.88**	**140.88**
TSRK8VR	**100.00**	**145.14**	**181.18**	**177.38**	**170.81**	**170.35**	**167.37**	**167.91**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**149.58**	**150.71**	**156.61**	**159.75**	**169.56**	**176.41**	**188.92**	**202.11**
MTSM	44.94	39.37	34.65	38.28	33.77	31.05	28.44	28.92
TSRK5DP	**136.23**	**126.80**	**127.70**	**123.18**	**120.43**	**109.77**	**109.53**	**88.64**
TSRK8VR	**165.78**	**160.67**	**152.92**	**148.47**	**121.75**	**138.80**	**136.79**	**131.08**
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**208.65**	**205.37**	**238.98**	**220.90**	**229.47**	**235.73**	**237.61**	**234.61**
MTSM	24.71	23.43	22.55	21.04	19.44	17.71	17.49	18.57
TSRK5DP	**95.29**	**94.03**	**94.88**	**96.56**	**88.09**	**86.96**	**80.27**	**81.67**
TSRK8VR	**129.97**	**122.74**	**122.93**	**121.92**	**112.81**	**108.74**	**111.00**	**107.98**
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**238.41**	**216.59**	**189.13**	**233.18**	**225.28**	**223.42**	**225.53**	**223.85**
MTSM	17.29	15.79	15.20	13.99	13.45	12.02	12.81	13.19
TSRK5DP	**72.90**	**69.21**	**73.56**	**72.01**	**71.67**	**69.34**	**68.81**	**72.29**
TSRK8VR	**108.93**	**99.03**	**98.23**	**90.37**	**93.61**	**90.50**	**94.73**	**89.47**

Table C.37: Efficiency, $S = 512000$, heat equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.13	3.40	4.85	6.19	7.59	9.60	11.42
MTSM	1.00	1.93	2.64	3.12	3.41	3.87	4.09	4.17
TSRK5DP	1.00	3.11	4.83	6.51	7.73	9.19	9.79	11.27
TSRK8VR	1.00	2.90	5.44	7.10	8.54	10.22	11.72	13.43
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	13.46	15.07	17.23	19.17	22.04	24.70	28.34	32.34
MTSM	4.04	3.94	3.81	4.59	4.39	4.35	4.27	4.63
TSRK5DP	12.26	12.68	14.05	14.78	15.66	15.37	16.43	14.18
TSRK8VR	14.92	16.07	16.82	17.82	15.83	19.43	20.52	20.97
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	35.47	36.97	45.41	44.18	48.19	51.86	54.65	56.31
MTSM	4.20	4.22	4.28	4.21	4.08	3.90	4.02	4.46
TSRK5DP	16.20	16.93	18.03	19.31	18.50	19.13	18.46	19.60
TSRK8VR	22.10	22.09	23.36	24.38	23.69	23.92	25.53	25.92
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	59.60	56.31	51.07	65.29	65.33	67.03	69.91	71.63
MTSM	4.32	4.11	4.10	3.92	3.90	3.61	3.97	4.22
TSRK5DP	18.22	17.99	19.86	20.16	20.78	20.80	21.33	23.13
TSRK8VR	27.23	25.75	26.52	25.30	27.15	27.15	29.37	28.63

Table C.38: Speedup, $S = 512000$, heat equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**1.30**	0.89	0.92	0.97	**1.04**	**1.07**	**1.28**	**1.32**
MTSM	**1.43**	0.88	0.78	0.68	0.63	0.60	0.60	0.53
TSRK8VR	0.56	0.52	0.63	0.61	0.62	0.62	0.67	0.67
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**1.43**	**1.55**	**1.60**	**1.69**	**1.83**	**2.09**	**2.24**	**2.97**
MTSM	0.47	0.44	0.39	0.44	0.40	0.40	0.37	0.47
TSRK8VR	0.68	0.71	0.67	0.68	0.57	0.71	0.70	0.83
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**2.85**	**2.84**	**3.28**	**2.98**	**3.39**	**3.53**	**3.85**	**3.74**
MTSM	0.37	0.36	0.34	0.31	0.31	0.29	0.31	0.32
TSRK8VR	0.76	0.73	0.73	0.71	0.72	0.70	0.78	0.74
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**4.26**	**4.07**	**3.35**	**4.21**	**4.09**	**4.19**	**4.26**	**4.03**
MTSM	0.34	0.33	0.29	0.28	0.27	0.25	0.27	0.26
TSRK8VR	0.84	0.80	0.75	0.70	0.73	0.73	0.77	0.69

Table C.39: Speedup against TSRK5DP, $S = 512000$, heat equation, five-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz [%]
20.48	4.10	4.10	2559991	$9.77e^{-04}$	51709351	$1.97e^{-02}$

Table C.40: Characteristics of input data, $S = 512000$, heat equation, five-point central difference formula.

C.2.4 $S = 1024000$, five-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	30.66	14.27	9.09	6.51	4.93	3.84	3.11	2.58
MTSM	38.17	18.71	12.97	10.11	8.65	7.66	7.18	7.17
TSRK5DP	75.62	25.76	13.17	9.14	7.10	5.92	5.14	4.59
TSRK8VR	125.42	48.18	26.51	16.40	11.99	9.79	8.28	7.71
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	2.23	1.99	1.78	1.61	1.40	1.20	1.02	0.91
MTSM	7.02	6.43	6.21	5.79	5.74	5.78	5.77	5.16
TSRK5DP	4.16	3.98	3.62	3.30	3.18	2.91	3.63	2.77
TSRK8VR	7.05	6.33	5.48	5.11	5.24	4.69	4.40	4.17
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	0.76	0.79	0.66	0.62	0.59	0.60	0.59	0.52
MTSM	5.41	5.67	5.91	5.58	5.43	5.32	5.63	5.10
TSRK5DP	3.24	2.64	2.67	2.29	2.23	2.32	2.38	2.18
TSRK8VR	3.80	3.95	3.42	3.50	3.46	3.19	3.08	3.16
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	0.50	0.49	0.48	0.48	0.49	0.42	0.45	0.40
MTSM	5.36	5.54	5.39	6.31	5.35	5.26	5.84	5.03
TSRK5DP	2.13	1.97	2.14	2.05	2.00	2.00	1.79	1.86
TSRK8VR	3.04	2.87	3.13	3.05	2.79	2.82	2.77	2.76

Table C.41: Average time, $S = 1024000$, heat equation, five-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**107.41**	**112.42**	**117.73**	**124.27**	**132.97**	**140.74**	**148.70**
MTSM	**100.00**	**101.99**	**98.11**	**94.38**	**88.24**	**83.03**	**75.96**	**66.58**
TSRK5DP	**100.00**	**146.75**	**191.38**	**206.84**	**213.00**	**212.83**	**210.09**	**205.97**
TSRK8VR	**100.00**	**130.16**	**157.73**	**191.16**	**209.29**	**213.62**	**216.28**	**203.39**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**152.77**	**154.07**	**156.98**	**158.65**	**168.63**	**183.22**	**200.10**	**210.09**
MTSM	**60.44**	**59.33**	**55.91**	**54.95**	**51.16**	47.18	44.08	46.20
TSRK5DP	**201.79**	**190.09**	**189.75**	**190.78**	**183.04**	**185.39**	**138.82**	**170.39**
TSRK8VR	**197.60**	**198.24**	**208.07**	**204.39**	**184.10**	**191.21**	**190.22**	**187.85**
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**237.59**	**214.63**	**243.38**	**246.88**	**246.53**	**230.44**	**225.87**	**247.71**
MTSM	41.53	37.42	34.00	34.21	33.50	32.62	29.46	31.20
TSRK5DP	**137.08**	**159.24**	**149.09**	**165.12**	**161.23**	**148.31**	**138.10**	**144.57**
TSRK8VR	**194.16**	**176.48**	**193.12**	**179.08**	**172.72**	**178.76**	**176.88**	**165.36**
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**245.14**	**241.71**	**236.83**	**227.67**	**216.31**	**245.12**	**222.12**	**239.04**
MTSM	28.49	26.49	26.22	21.59	24.59	24.18	21.09	23.72
TSRK5DP	**142.03**	**147.78**	**131.15**	**131.71**	**130.49**	**126.00**	**136.46**	**127.21**
TSRK8VR	**165.29**	**167.90**	**148.32**	**147.05**	**155.27**	**148.32**	**146.06**	**141.98**

Table C.42: Efficiency, $S = 1024000$, heat equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.15	3.37	4.71	6.21	7.98	9.85	11.90
MTSM	1.00	2.04	2.94	3.78	4.41	4.98	5.32	5.33
TSRK5DP	1.00	2.93	5.74	8.27	10.65	12.77	14.71	16.48
TSRK8VR	1.00	2.60	4.73	7.65	10.46	12.82	15.14	16.27
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	13.75	15.41	17.27	19.04	21.92	25.65	30.02	33.61
MTSM	5.44	5.93	6.15	6.59	6.65	6.60	6.61	7.39
TSRK5DP	18.16	19.01	20.87	22.89	23.80	25.95	20.82	27.26
TSRK8VR	17.78	19.82	22.89	24.53	23.93	26.77	28.53	30.06
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	40.39	38.63	46.24	49.38	51.77	50.70	51.95	59.45
MTSM	7.06	6.74	6.46	6.84	7.03	7.18	6.77	7.49
TSRK5DP	23.30	28.66	28.33	33.02	33.86	32.63	31.76	34.70
TSRK8VR	33.01	31.77	36.69	35.82	36.27	39.33	40.68	39.69
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	61.29	62.85	63.94	63.75	62.73	73.54	68.86	76.49
MTSM	7.12	6.89	7.08	6.04	7.13	7.25	6.54	7.59
TSRK5DP	35.51	38.42	35.41	36.88	37.84	37.80	42.30	40.71
TSRK8VR	41.32	43.65	40.05	41.17	45.03	44.50	45.28	45.43

Table C.43: Speedup, $S = 1024000$, heat equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**2.47**	**1.81**	**1.45**	**1.40**	**1.44**	**1.54**	**1.65**	**1.78**
MTSM	**1.98**	**1.38**	**1.02**	0.90	0.82	0.77	0.72	0.64
TSRK8VR	0.60	0.53	0.50	0.56	0.59	0.61	0.62	0.60
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**1.87**	**2.00**	**2.04**	**2.05**	**2.27**	**2.44**	**3.56**	**3.04**
MTSM	0.59	0.62	0.58	0.57	0.55	0.50	0.63	0.54
TSRK8VR	0.59	0.63	0.66	0.65	0.61	0.62	0.83	0.66
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**4.28**	**3.32**	**4.03**	**3.69**	**3.77**	**3.83**	**4.03**	**4.23**
MTSM	0.60	0.47	0.45	0.41	0.41	0.44	0.42	0.43
TSRK8VR	0.85	0.67	0.78	0.65	0.65	0.73	0.77	0.69
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**4.26**	**4.03**	**4.45**	**4.26**	**4.09**	**4.80**	**4.02**	**4.63**
MTSM	0.40	0.36	0.40	0.32	0.37	0.38	0.31	0.37
TSRK8VR	0.70	0.68	0.68	0.67	0.72	0.71	0.65	0.67

Table C.44: Speedup against TSRK5DP, $S = 1024000$, heat equation, five-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A_{PRECALC}}$ nnz	$\mathbf{A_{PRECALC}}$ nnz [%]
40.96	8.19	8.19	3071995	$2.93e^{-04}$	52223299	$4.98e^{-03}$

Table C.45: Characteristics of input data, $S = 1024000$, heat equation, five-point central difference formula.

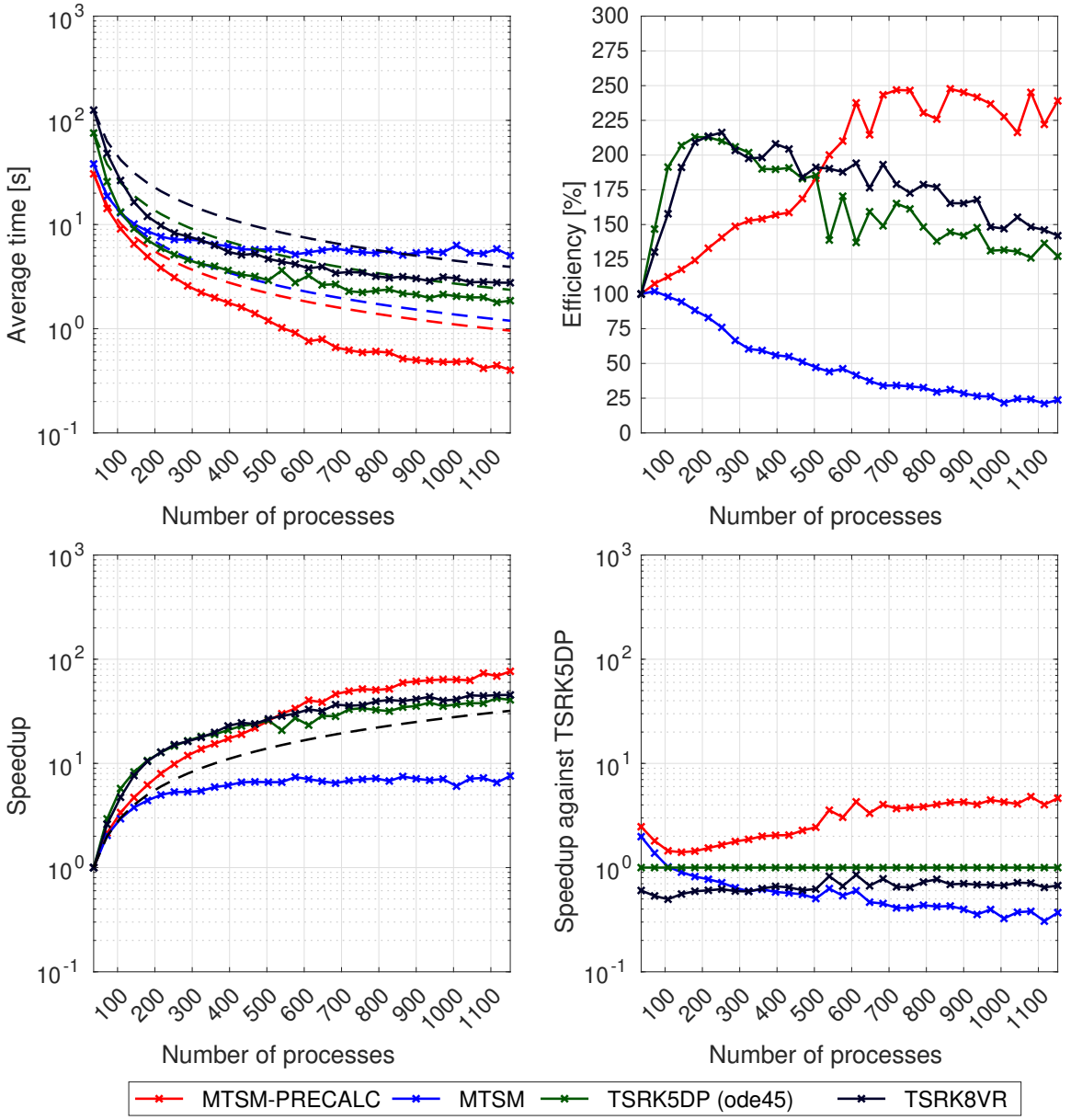


Figure C.7: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 1024000$, heat equation, five-point central difference formula.

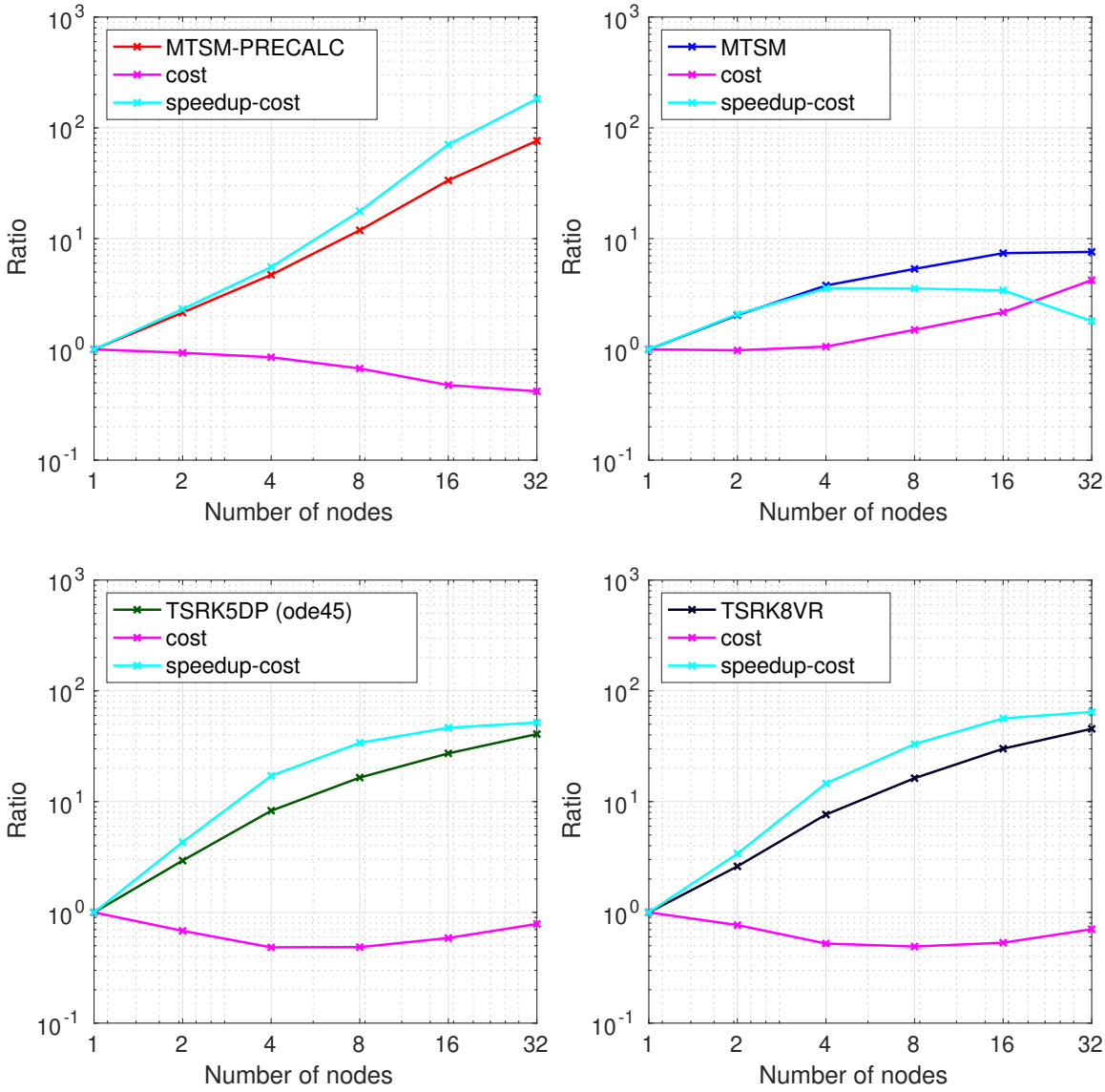


Figure C.8: Parallel cost ratio and speedup-cost ratio, $S = 1024000$, heat equation, five-point central difference formula.

C.2.5 $S = 2048000$, five-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	122.41	61.31	40.82	30.17	23.85	19.66	16.88	14.28
MTSM	195.56	71.39	40.81	29.17	23.55	19.64	18.06	15.76
TSRK5DP	277.92	120.10	68.08	41.81	29.29	20.82	16.83	13.34
TSRK8VR	444.61	195.01	113.36	73.83	53.05	40.53	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	14.82	11.13	10.00	8.98	8.39	7.48	6.90	6.38
MTSM	15.90	14.31	13.25	12.27	12.51	11.56	12.15	11.02
TSRK5DP	11.42	11.02	9.29	8.27	8.91	7.81	7.00	6.43
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	5.97	5.48	5.15	4.76	4.53	4.34	4.34	3.81
MTSM	10.96	10.50	10.68	10.61	10.04	10.33	10.40	9.57
TSRK5DP	6.43	6.01	6.08	5.92	5.19	5.42	5.54	4.95
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	3.66	3.46	3.29	3.04	2.96	2.85	2.68	2.59
MTSM	9.45	9.63	9.61	9.35	9.37	10.04	9.82	8.74
TSRK5DP	4.83	4.57	4.70	4.52	4.39	6.02	4.15	3.83
TSRK8VR	-	-	-	-	-	-	-	-

Table C.46: Average time, $S = 2048000$, heat equation, five-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**99.83**	**99.95**	**101.43**	**102.64**	**103.76**	**103.62**	**107.13**
MTSM	**100.00**	**136.97**	**159.72**	**167.63**	**166.11**	**165.97**	**154.65**	**155.16**
TSRK5DP	**100.00**	**115.70**	**136.08**	**166.18**	**189.77**	**222.49**	**235.90**	**260.42**
TSRK8VR	**100.00**	**113.99**	**130.74**	**150.56**	**167.61**	**182.83**	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**91.77**	**109.97**	**111.32**	**113.55**	**112.22**	**116.89**	**118.23**	**119.94**
MTSM	**136.63**	**136.67**	**134.21**	**132.81**	**120.26**	**120.88**	**107.34**	**110.94**
TSRK5DP	**270.30**	**252.21**	**271.89**	**280.15**	**239.87**	**254.30**	**264.82**	**270.31**
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**120.60**	**124.08**	**125.06**	**128.61**	**128.81**	**128.16**	**122.56**	**133.73**
MTSM	**104.95**	**103.44**	**96.38**	**92.13**	**92.76**	**86.02**	**81.73**	**85.11**
TSRK5DP	**254.16**	**256.91**	**240.53**	**234.79**	**254.76**	**232.95**	**218.19**	**233.76**
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**133.74**	**136.12**	**137.94**	**143.68**	**142.54**	**142.98**	**147.42**	**147.72**
MTSM	**82.82**	**78.09**	**75.33**	**74.71**	**71.98**	**64.93**	**64.22**	**69.93**
TSRK5DP	**230.38**	**233.77**	**218.95**	**219.52**	**218.50**	**153.94**	**216.04**	**226.68**
TSRK8VR	-	-	-	-	-	-	-	-

Table C.47: Efficiency, $S = 2048000$, heat equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.00	3.00	4.06	5.13	6.23	7.25	8.57
MTSM	1.00	2.74	4.79	6.71	8.31	9.96	10.83	12.41
TSRK5DP	1.00	2.31	4.08	6.65	9.49	13.35	16.51	20.83
TSRK8VR	1.00	2.28	3.92	6.02	8.38	10.97	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	8.26	11.00	12.24	13.63	14.59	16.37	17.73	19.19
MTSM	12.30	13.67	14.76	15.94	15.63	16.92	16.10	17.75
TSRK5DP	24.33	25.22	29.91	33.62	31.18	35.60	39.72	43.25
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	20.50	22.33	23.76	25.72	27.05	28.19	28.19	32.10
MTSM	17.84	18.62	18.31	18.43	19.48	18.92	18.80	20.43
TSRK5DP	43.21	46.24	45.70	46.96	53.50	51.25	50.18	56.10
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	33.44	35.39	37.24	40.23	41.34	42.89	45.70	47.27
MTSM	20.70	20.30	20.34	20.92	20.88	19.48	19.91	22.38
TSRK5DP	57.59	60.78	59.12	61.47	63.37	46.18	66.97	72.54
TSRK8VR	-	-	-	-	-	-	-	-

Table C.48: Speedup, $S = 2048000$, heat equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**2.27**	**1.96**	**1.67**	**1.39**	**1.23**	**1.06**	1.00	0.93
MTSM	**1.42**	**1.68**	**1.67**	**1.43**	**1.24**	**1.06**	0.93	0.85
TSRK8VR	0.63	0.62	0.60	0.57	0.55	0.51	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	0.77	0.99	0.93	0.92	**1.06**	**1.04**	**1.01**	**1.01**
MTSM	0.72	0.77	0.70	0.67	0.71	0.68	0.58	0.58
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**1.08**	**1.10**	**1.18**	**1.24**	**1.15**	**1.25**	**1.28**	**1.30**
MTSM	0.59	0.57	0.57	0.56	0.52	0.52	0.53	0.52
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**1.32**	**1.32**	**1.43**	**1.49**	**1.48**	**2.11**	**1.55**	**1.48**
MTSM	0.51	0.47	0.49	0.48	0.47	0.60	0.42	0.44
TSRK8VR	-	-	-	-	-	-	-	-

Table C.49: Speedup against TSRK5DP, $S = 2048000$, heat equation, five-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz [%]
81.92	16.38	16.38	10239991	$2.44e^{-04}$	206845351	$4.93e^{-03}$

Table C.50: Characteristics of input data, $S = 2048000$, heat equation, five-point central difference formula.

C.3 Wave equation – three-point central difference formula

Subsections C.3.1–C.3.5 present numerical results for $S = 64000$, $S = 128000$, $S = 256000$, $S = 512000$, and $S = 1024000$, respectively.

C.3.1 $S = 64000$, three-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	2.88	1.17	0.80	0.66	0.61	0.62	0.51	0.57
MTSM	11.10	13.90	10.98	9.37	8.58	7.85	7.45	7.01
TSRK5DP	23.38	26.70	19.84	15.51	13.63	12.53	9.18	8.54
TSRK8VR	28.41	29.21	21.74	17.14	15.33	13.09	9.84	9.16
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	0.48	0.49	0.57	0.51	0.51	0.46	0.41	0.59
MTSM	7.23	7.43	7.60	6.36	6.70	6.97	7.14	6.56
TSRK5DP	8.03	7.80	8.10	7.11	6.95	6.92	6.54	6.60
TSRK8VR	8.79	8.40	7.84	7.69	7.21	7.19	6.78	6.46
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	0.39	0.46	0.39	0.39	0.39	0.40	0.41	0.43
MTSM	7.00	7.30	9.15	10.98	11.06	10.97	9.42	9.24
TSRK5DP	6.51	6.31	6.16	5.97	6.19	6.37	5.93	6.05
TSRK8VR	6.60	6.57	6.39	6.18	5.96	5.93	5.83	6.32
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	0.50	0.37	0.42	0.32	0.38	0.40	0.45	0.37
MTSM	8.52	8.13	8.24	7.54	6.91	6.84	6.72	6.79
TSRK5DP	5.90	5.95	6.28	6.36	5.49	5.75	5.96	5.88
TSRK8VR	5.96	5.51	5.98	5.78	5.79	5.92	5.78	5.91

Table C.51: Average time, $S = 64000$, wave equation, three-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**123.11**	**119.85**	**108.60**	**94.60**	**77.76**	**79.92**	**62.96**
MTSM	**100.00**	39.93	33.70	29.61	25.86	23.56	21.30	19.81
TSRK5DP	**100.00**	43.78	39.27	37.69	34.30	31.10	36.39	34.23
TSRK8VR	**100.00**	48.63	43.57	41.45	37.08	36.17	41.26	38.77
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**66.25**	**58.28**	46.01	47.16	43.07	44.24	46.56	30.66
MTSM	17.06	14.95	13.28	14.54	12.75	11.37	10.36	10.58
TSRK5DP	32.37	29.96	26.23	27.39	25.87	24.14	23.84	22.13
TSRK8VR	35.92	33.82	32.95	30.78	30.30	28.24	27.93	27.49
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	43.90	35.10	38.41	37.15	34.88	32.68	30.76	28.05
MTSM	9.33	8.45	6.39	5.06	4.78	4.60	5.12	5.00
TSRK5DP	21.14	20.58	19.99	19.56	17.99	16.69	17.13	16.10
TSRK8VR	25.34	24.02	23.40	23.00	22.69	21.78	21.20	18.74
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	23.20	29.91	25.45	31.92	26.22	23.89	20.57	24.02
MTSM	5.21	5.25	4.99	5.26	5.54	5.41	5.33	5.11
TSRK5DP	15.86	15.12	13.79	13.12	14.69	13.55	12.65	12.42
TSRK8VR	19.07	19.83	17.59	17.57	16.91	16.00	15.85	15.03

Table C.52: Efficiency, $S = 64000$, wave equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.46	3.60	4.34	4.73	4.67	5.59	5.04
MTSM	1.00	0.80	1.01	1.18	1.29	1.41	1.49	1.58
TSRK5DP	1.00	0.88	1.18	1.51	1.71	1.87	2.55	2.74
TSRK8VR	1.00	0.97	1.31	1.66	1.85	2.17	2.89	3.10
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	5.96	5.83	5.06	5.66	5.60	6.19	6.98	4.91
MTSM	1.54	1.49	1.46	1.74	1.66	1.59	1.55	1.69
TSRK5DP	2.91	3.00	2.89	3.29	3.36	3.38	3.58	3.54
TSRK8VR	3.23	3.38	3.62	3.69	3.94	3.95	4.19	4.40
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	7.46	6.32	7.30	7.43	7.32	7.19	7.07	6.73
MTSM	1.59	1.52	1.21	1.01	1.00	1.01	1.18	1.20
TSRK5DP	3.59	3.70	3.80	3.91	3.78	3.67	3.94	3.86
TSRK8VR	4.31	4.32	4.45	4.60	4.76	4.79	4.88	4.50
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	5.80	7.78	6.87	8.94	7.61	7.17	6.38	7.69
MTSM	1.30	1.37	1.35	1.47	1.61	1.62	1.65	1.63
TSRK5DP	3.96	3.93	3.72	3.67	4.26	4.07	3.92	3.97
TSRK8VR	4.77	5.16	4.75	4.92	4.90	4.80	4.91	4.81

Table C.53: Speedup, $S = 64000$, wave equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**8.12**	**22.83**	**24.78**	**23.40**	**22.40**	**20.30**	**17.83**	**14.94**
MTSM	**2.11**	**1.92**	**1.81**	**1.65**	**1.59**	**1.59**	**1.23**	**1.22**
TSRK8VR	0.82	0.91	0.91	0.90	0.89	0.96	0.93	0.93
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**16.62**	**15.80**	**14.24**	**13.98**	**13.52**	**14.88**	**15.86**	**11.25**
MTSM	**1.11**	**1.05**	**1.07**	**1.12**	**1.04**	0.99	0.92	**1.01**
TSRK8VR	0.91	0.93	**1.03**	0.92	0.96	0.96	0.96	**1.02**
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**16.86**	**13.85**	**15.60**	**15.42**	**15.74**	**15.90**	**14.58**	**14.15**
MTSM	0.93	0.86	0.67	0.54	0.56	0.58	0.63	0.65
TSRK8VR	0.99	0.96	0.96	0.97	**1.04**	**1.07**	**1.02**	0.96
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**11.88**	**16.06**	**14.99**	**19.75**	**14.50**	**14.31**	**13.21**	**15.71**
MTSM	0.69	0.73	0.76	0.84	0.79	0.84	0.89	0.87
TSRK8VR	0.99	**1.08**	**1.05**	**1.10**	0.95	0.97	**1.03**	1.00

Table C.54: Speedup against TSRK5DP, $S = 64000$, wave equation, three-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A_{PRECALC}}$ nnz	$\mathbf{A_{PRECALC}}$ nnz [%]
3.58	1.02	1.02	255994	$1.56e^{-03}$	6527248	$3.98e^{-02}$

Table C.55: Characteristics of input data, $S = 64000$, wave equation, three-point central difference formula.

C.3.2 $S = 128000$, three-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	7.03	3.62	2.15	1.58	0.98	0.94	0.77	0.73
MTSM	20.35	24.96	19.44	14.73	13.23	11.22	11.05	10.11
TSRK5DP	51.75	48.24	38.42	27.64	24.56	20.56	18.54	16.24
TSRK8VR	64.26	54.38	44.48	30.44	27.10	22.49	19.93	17.36
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	0.72	0.68	0.64	0.59	0.63	0.77	0.77	0.75
MTSM	12.84	9.55	9.79	8.77	9.19	9.03	8.28	7.88
TSRK5DP	15.76	14.56	14.14	12.83	23.57	21.01	18.89	19.29
TSRK8VR	17.44	15.33	14.76	13.37	13.62	10.71	9.91	9.46
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	0.60	0.58	0.58	0.56	0.59	0.53	0.57	0.54
MTSM	8.36	8.29	8.14	8.37	8.25	8.56	8.76	7.39
TSRK5DP	17.46	17.84	18.01	16.43	14.35	13.96	8.93	8.62
TSRK8VR	9.83	9.16	9.23	8.90	9.33	8.66	8.75	8.13
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	0.54	0.53	0.62	0.57	0.51	0.49	0.49	0.55
MTSM	7.82	8.25	8.33	7.71	8.04	8.63	8.77	8.70
TSRK5DP	8.76	8.00	8.02	7.93	6.99	7.31	7.23	7.58
TSRK8VR	8.41	8.47	7.67	8.31	7.48	7.21	7.51	7.67

Table C.56: Average time, $S = 128000$, wave equation, three-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**97.26**	**109.14**	**110.96**	**144.05**	**124.96**	**131.00**	**119.78**
MTSM	**100.00**	40.77	34.89	34.53	30.76	30.22	26.32	25.16
TSRK5DP	**100.00**	**53.64**	44.90	46.80	42.14	41.95	39.87	39.83
TSRK8VR	**100.00**	**59.08**	48.15	**52.77**	47.43	47.62	46.06	46.27
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**108.52**	**103.66**	**100.30**	**99.44**	**85.58**	**65.28**	**60.99**	**58.48**
MTSM	17.61	21.31	18.90	19.33	17.02	16.10	16.38	16.13
TSRK5DP	36.49	35.55	33.27	33.62	16.89	17.60	18.26	16.77
TSRK8VR	40.93	41.91	39.58	40.04	36.28	42.83	43.23	42.45
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**68.69**	**66.88**	**63.76**	**62.59**	**56.47**	**60.36**	**53.76**	**53.99**
MTSM	14.31	13.63	13.16	12.16	11.74	10.80	10.10	11.47
TSRK5DP	17.44	16.12	15.12	15.75	17.17	16.85	25.20	25.02
TSRK8VR	38.45	38.95	36.65	36.12	32.80	33.71	31.93	32.95
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**51.85**	**51.00**	41.91	44.33	47.77	47.98	46.07	40.20
MTSM	10.41	9.49	9.05	9.42	8.73	7.86	7.49	7.31
TSRK5DP	23.63	24.88	23.90	23.30	25.55	23.61	23.08	21.33
TSRK8VR	30.56	29.18	31.03	27.61	29.61	29.70	27.60	26.17

Table C.57: Efficiency, $S = 128000$, wave equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	1.95	3.27	4.44	7.20	7.50	9.17	9.58
MTSM	1.00	0.82	1.05	1.38	1.54	1.81	1.84	2.01
TSRK5DP	1.00	1.07	1.35	1.87	2.11	2.52	2.79	3.19
TSRK8VR	1.00	1.18	1.44	2.11	2.37	2.86	3.22	3.70
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	9.77	10.37	11.03	11.93	11.12	9.14	9.15	9.36
MTSM	1.59	2.13	2.08	2.32	2.21	2.25	2.46	2.58
TSRK5DP	3.28	3.56	3.66	4.03	2.20	2.46	2.74	2.68
TSRK8VR	3.68	4.19	4.35	4.80	4.72	6.00	6.48	6.79
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	11.68	12.04	12.11	12.52	11.86	13.28	12.36	12.96
MTSM	2.43	2.45	2.50	2.43	2.47	2.38	2.32	2.75
TSRK5DP	2.96	2.90	2.87	3.15	3.61	3.71	5.80	6.00
TSRK8VR	6.54	7.01	6.96	7.22	6.89	7.42	7.34	7.91
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	12.96	13.26	11.31	12.41	13.85	14.39	14.28	12.86
MTSM	2.60	2.47	2.44	2.64	2.53	2.36	2.32	2.34
TSRK5DP	5.91	6.47	6.45	6.52	7.41	7.08	7.15	6.83
TSRK8VR	7.64	7.59	8.38	7.73	8.59	8.91	8.55	8.37

Table C.58: Speedup, $S = 128000$, wave equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**7.36**	**13.34**	**17.88**	**17.44**	**25.15**	**21.92**	**24.17**	**22.13**
MTSM	**2.54**	**1.93**	**1.98**	**1.88**	**1.86**	**1.83**	**1.68**	**1.61**
TSRK8VR	0.81	0.89	0.86	0.91	0.91	0.91	0.93	0.94
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**21.88**	**21.45**	**22.18**	**21.76**	**37.27**	**27.29**	**24.57**	**25.66**
MTSM	**1.23**	**1.52**	**1.44**	**1.46**	**2.56**	**2.33**	**2.28**	**2.45**
TSRK8VR	0.90	0.95	0.96	0.96	**1.73**	**1.96**	**1.91**	**2.04**
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**28.99**	**30.53**	**31.02**	**29.23**	**24.20**	**26.36**	**15.69**	**15.88**
MTSM	**2.09**	**2.15**	**2.21**	**1.96**	**1.74**	**1.63**	**1.02**	**1.17**
TSRK8VR	**1.78**	**1.95**	**1.95**	**1.85**	**1.54**	**1.61**	**1.02**	**1.06**
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**16.14**	**15.08**	**12.90**	**14.00**	**13.76**	**14.95**	**14.69**	**13.86**
MTSM	**1.12**	0.97	0.96	**1.03**	0.87	0.85	0.82	0.87
TSRK8VR	**1.04**	0.94	**1.05**	0.95	0.93	**1.01**	0.96	0.99

Table C.59: Speedup against TSRK5DP, $S = 128000$, wave equation, three-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz [%]
7.17	2.05	2.05	511994	$7.81e^{-04}$	13055248	$1.99e^{-02}$

Table C.60: Characteristics of input data, $S = 128000$, wave equation, three-point central difference formula.

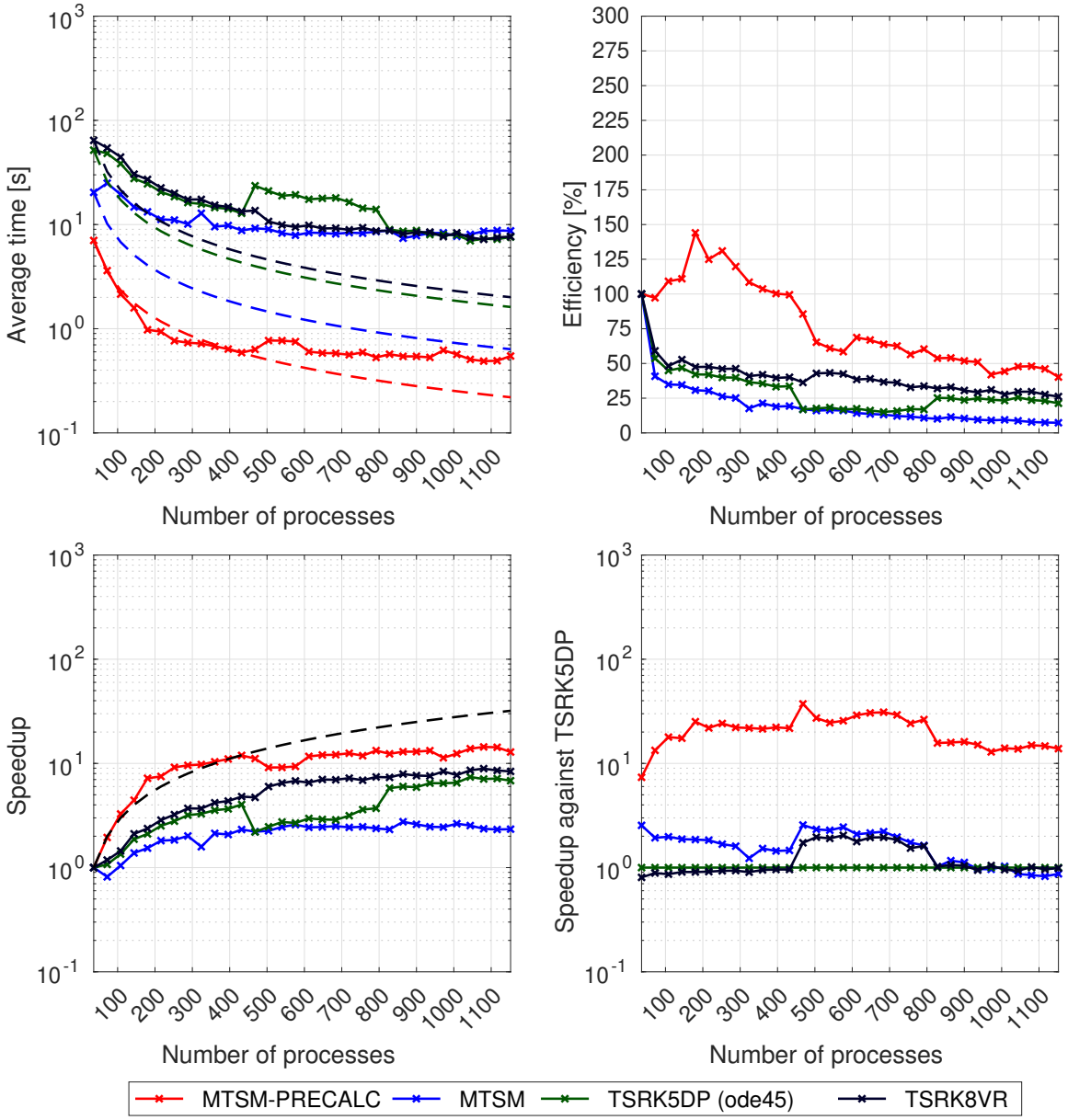


Figure C.9: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 128000$, wave equation, three-point central difference formula.

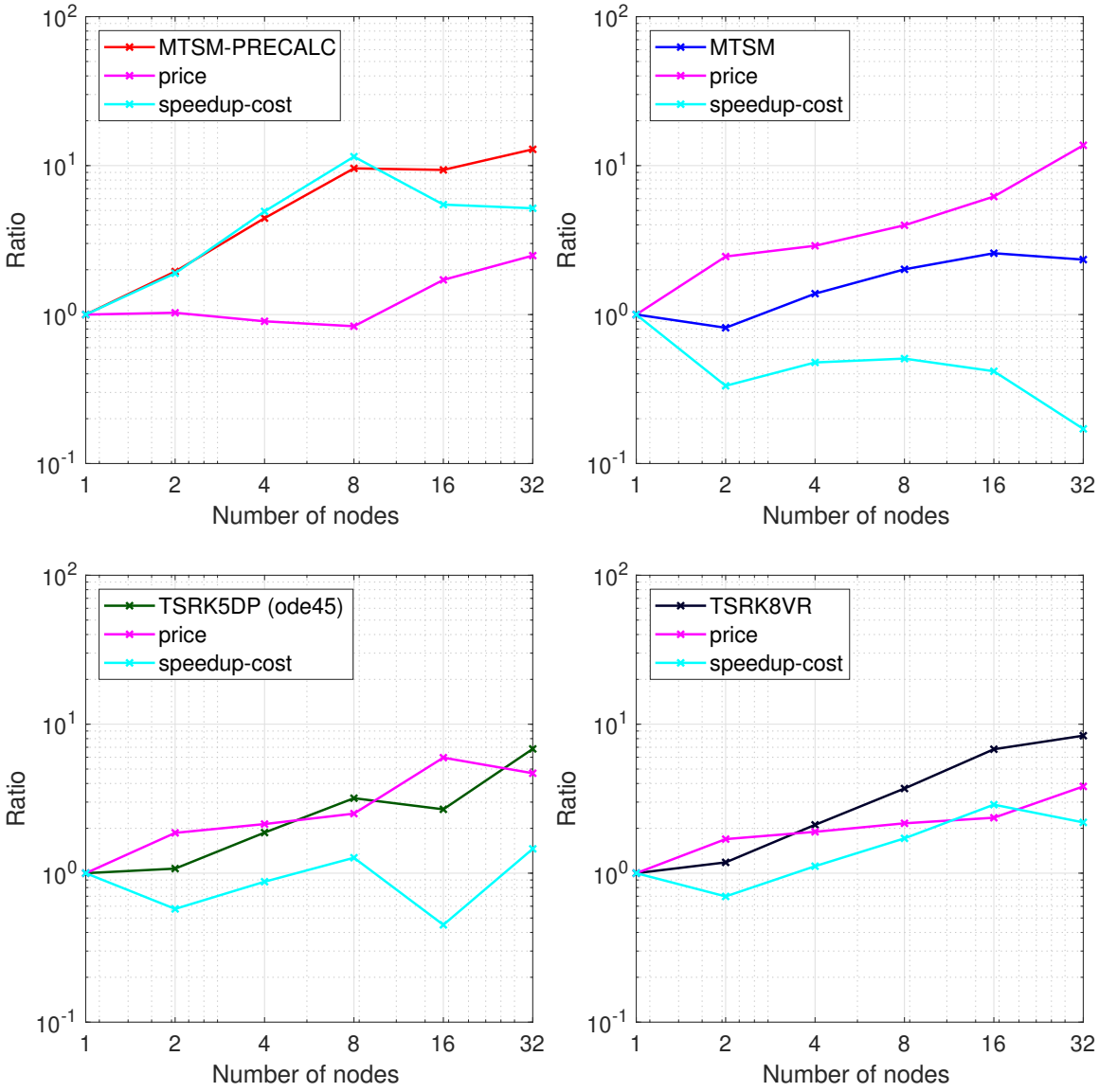


Figure C.10: Parallel cost ratio and speedup-cost ratio, $S = 128000$, wave equation, three-point central difference formula.

C.3.3 $S = 256000$, three-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	16.78	7.98	4.86	3.49	2.77	2.61	1.97	1.67
MTSM	40.96	46.17	34.92	25.58	23.90	19.04	18.20	16.19
TSRK5DP	141.11	98.01	76.16	52.95	48.14	37.45	35.07	29.36
TSRK8VR	189.06	111.45	88.76	59.53	55.89	42.13	38.24	32.87
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	1.37	1.09	0.93	0.88	0.69	0.81	0.78	0.78
MTSM	15.76	14.51	14.41	12.82	12.65	12.23	12.73	11.05
TSRK5DP	28.35	24.67	23.79	21.63	20.96	19.59	18.79	17.36
TSRK8VR	32.18	26.42	25.94	23.51	23.28	20.27	20.04	18.81
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	0.63	0.73	0.71	0.77	0.58	0.59	0.56	0.62
MTSM	11.80	11.33	11.39	10.75	11.01	10.97	11.26	9.83
TSRK5DP	17.39	16.48	15.85	15.74	15.15	15.43	14.43	13.64
TSRK8VR	18.11	16.86	17.60	15.98	16.21	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	0.56	0.50	0.52	0.52	0.51	0.51	0.50	0.53
MTSM	10.33	10.62	10.32	10.13	10.72	10.72	9.81	9.19
TSRK5DP	14.29	13.27	12.14	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.61: Average time, $S = 256000$, wave equation, three-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**105.20**	**115.14**	**120.15**	**121.33**	**107.22**	**121.93**	**125.26**
MTSM	**100.00**	44.35	39.09	40.02	34.27	35.85	32.14	31.62
TSRK5DP	**100.00**	**71.99**	**61.76**	**66.62**	**58.63**	**62.80**	**57.48**	**60.07**
TSRK8VR	**100.00**	**84.82**	**71.00**	**79.40**	**67.65**	**74.79**	**70.63**	**71.89**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**135.64**	**154.47**	**163.56**	**158.94**	**186.60**	**148.17**	**143.57**	**134.77**
MTSM	28.87	28.23	25.83	26.62	24.90	23.91	21.44	23.17
TSRK5DP	**55.30**	**57.20**	**53.93**	**54.35**	**51.79**	**51.46**	**50.06**	**50.80**
TSRK8VR	**65.28**	**71.55**	**66.27**	**67.02**	**62.47**	**66.61**	**62.89**	**62.82**
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**156.90**	**127.95**	**125.21**	**108.61**	**136.75**	**128.22**	**129.84**	**112.51**
MTSM	20.42	20.08	18.93	19.05	17.72	16.97	15.81	17.35
TSRK5DP	47.73	47.57	46.85	44.82	44.36	41.58	42.51	43.11
TSRK8VR	**61.43**	**62.29**	**56.53**	**59.16**	**55.53**	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**120.93**	**130.12**	**120.22**	**114.21**	**113.73**	**108.67**	**109.24**	**99.35**
MTSM	15.86	14.83	14.69	14.44	13.17	12.73	13.47	13.92
TSRK5DP	39.50	40.89	43.05	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.62: Efficiency, $S = 256000$, wave equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.10	3.45	4.81	6.07	6.43	8.54	10.02
MTSM	1.00	0.89	1.17	1.60	1.71	2.15	2.25	2.53
TSRK5DP	1.00	1.44	1.85	2.66	2.93	3.77	4.02	4.81
TSRK8VR	1.00	1.70	2.13	3.18	3.38	4.49	4.94	5.75
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	12.21	15.45	17.99	19.07	24.26	20.74	21.54	21.56
MTSM	2.60	2.82	2.84	3.19	3.24	3.35	3.22	3.71
TSRK5DP	4.98	5.72	5.93	6.52	6.73	7.20	7.51	8.13
TSRK8VR	5.88	7.15	7.29	8.04	8.12	9.33	9.43	10.05
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	26.67	23.03	23.79	21.72	28.72	28.21	29.86	27.00
MTSM	3.47	3.61	3.60	3.81	3.72	3.73	3.64	4.16
TSRK5DP	8.11	8.56	8.90	8.96	9.32	9.15	9.78	10.35
TSRK8VR	10.44	11.21	10.74	11.83	11.66	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	30.23	33.83	32.46	31.98	32.98	32.60	33.86	31.79
MTSM	3.96	3.86	3.97	4.04	3.82	3.82	4.18	4.46
TSRK5DP	9.87	10.63	11.62	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.63: Speedup, $S = 256000$, wave equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**8.41**	**12.29**	**15.68**	**15.16**	**17.40**	**14.36**	**17.83**	**17.53**
MTSM	**3.45**	**2.12**	**2.18**	**2.07**	**2.01**	**1.97**	**1.93**	**1.81**
TSRK8VR	0.75	0.88	0.86	0.89	0.86	0.89	0.92	0.89
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**20.62**	**22.71**	**25.50**	**24.59**	**30.29**	**24.21**	**24.12**	**22.31**
MTSM	**1.80**	**1.70**	**1.65**	**1.69**	**1.66**	**1.60**	**1.48**	**1.57**
TSRK8VR	0.88	0.93	0.92	0.92	0.90	0.97	0.94	0.92
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**27.64**	**22.61**	**22.47**	**20.38**	**25.92**	**25.93**	**25.68**	**21.94**
MTSM	**1.47**	**1.45**	**1.39**	**1.46**	**1.38**	**1.41**	**1.28**	**1.39**
TSRK8VR	0.96	0.98	0.90	0.99	0.93	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**25.74**	**26.76**	**23.48**	-	-	-	-	-
MTSM	**1.38**	**1.25**	**1.18**	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.64: Speedup against TSRK5DP, $S = 256000$, wave equation, three-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A_{PRECALC}}$ nnz	$\mathbf{A_{PRECALC}}$ nnz [%]
14.34	4.10	4.10	1023994	$3.91e^{-04}$	26111248	$9.96e^{-03}$

Table C.65: Characteristics of input data, $S = 256000$, wave equation, three-point central difference formula.

C.3.4 $S = 512000$, three-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	35.36	17.76	11.32	8.05	6.35	4.93	5.17	4.38
MTSM	90.54	90.76	68.21	47.24	42.99	33.92	33.15	27.70
TSRK5DP	411.04	217.75	160.53	101.99	-	-	-	-
TSRK8VR	539.66	260.22	-	-	-	-	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	3.06	2.83	2.66	2.80	2.41	1.94	1.66	1.88
MTSM	26.90	23.46	23.45	23.41	21.09	18.95	19.72	17.42
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	1.52	1.15	0.99	1.15	1.03	0.89	0.96	0.90
MTSM	17.83	17.02	17.09	18.40	16.30	15.54	15.73	15.23
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	0.85	0.78	0.98	0.76	0.79	0.75	0.76	0.90
MTSM	14.92	14.72	14.45	13.59	14.44	14.45	13.39	13.19
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.66: Average time, $S = 512000$, wave equation, three-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**99.57**	**104.11**	**109.81**	**111.35**	**119.43**	**97.62**	**101.02**
MTSM	**100.00**	49.88	44.25	47.92	42.12	44.49	39.02	40.86
TSRK5DP	**100.00**	**94.38**	**85.35**	**100.76**	-	-	-	-
TSRK8VR	**100.00**	**103.69**	-	-	-	-	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**128.46**	**125.00**	**120.86**	**105.24**	**112.74**	**129.87**	**141.73**	**117.67**
MTSM	37.40	38.60	35.10	32.24	33.03	34.13	30.61	32.48
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**136.72**	**170.38**	**187.76**	**154.18**	**163.93**	**179.62**	**160.80**	**164.38**
MTSM	29.86	29.55	27.89	24.60	26.45	26.49	25.02	24.78
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**165.63**	**173.36**	**134.18**	**165.53**	**154.93**	**156.71**	**150.91**	**122.29**
MTSM	24.27	23.66	23.20	23.80	21.62	20.88	21.82	21.46
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.67: Efficiency, $S = 512000$, wave equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	1.99	3.12	4.39	5.57	7.17	6.83	8.08
MTSM	1.00	1.00	1.33	1.92	2.11	2.67	2.73	3.27
TSRK5DP	1.00	1.89	2.56	4.03	-	-	-	-
TSRK8VR	1.00	2.07	-	-	-	-	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	11.56	12.50	13.29	12.63	14.66	18.18	21.26	18.83
MTSM	3.37	3.86	3.86	3.87	4.29	4.78	4.59	5.20
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	23.24	30.67	35.67	30.84	34.43	39.52	36.98	39.45
MTSM	5.08	5.32	5.30	4.92	5.55	5.83	5.75	5.95
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	41.41	45.07	36.23	46.35	44.93	47.01	46.78	39.13
MTSM	6.07	6.15	6.27	6.67	6.27	6.26	6.76	6.87
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.68: Speedup, $S = 512000$, wave equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**11.63**	**12.26**	**14.18**	**12.67**	-	-	-	-
MTSM	**4.54**	**2.40**	**2.35**	**2.16**	-	-	-	-
TSRK8VR	0.76	0.84	-	-	-	-	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	-	-	-	-	-	-	-	-
MTSM	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	-	-	-	-	-	-	-	-
MTSM	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	-	-	-	-	-	-	-	-
MTSM	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.69: Speedup against TSRK5DP, $S = 512000$, wave equation, three-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A_{PRECALC}}$ nnz	$\mathbf{A_{PRECALC}}$ nnz [%]
28.67	8.19	8.19	2047994	$1.95e^{-04}$	52223248	$4.98e^{-03}$

Table C.70: Characteristics of input data, $S = 512000$, wave equation, three-point central difference formula.

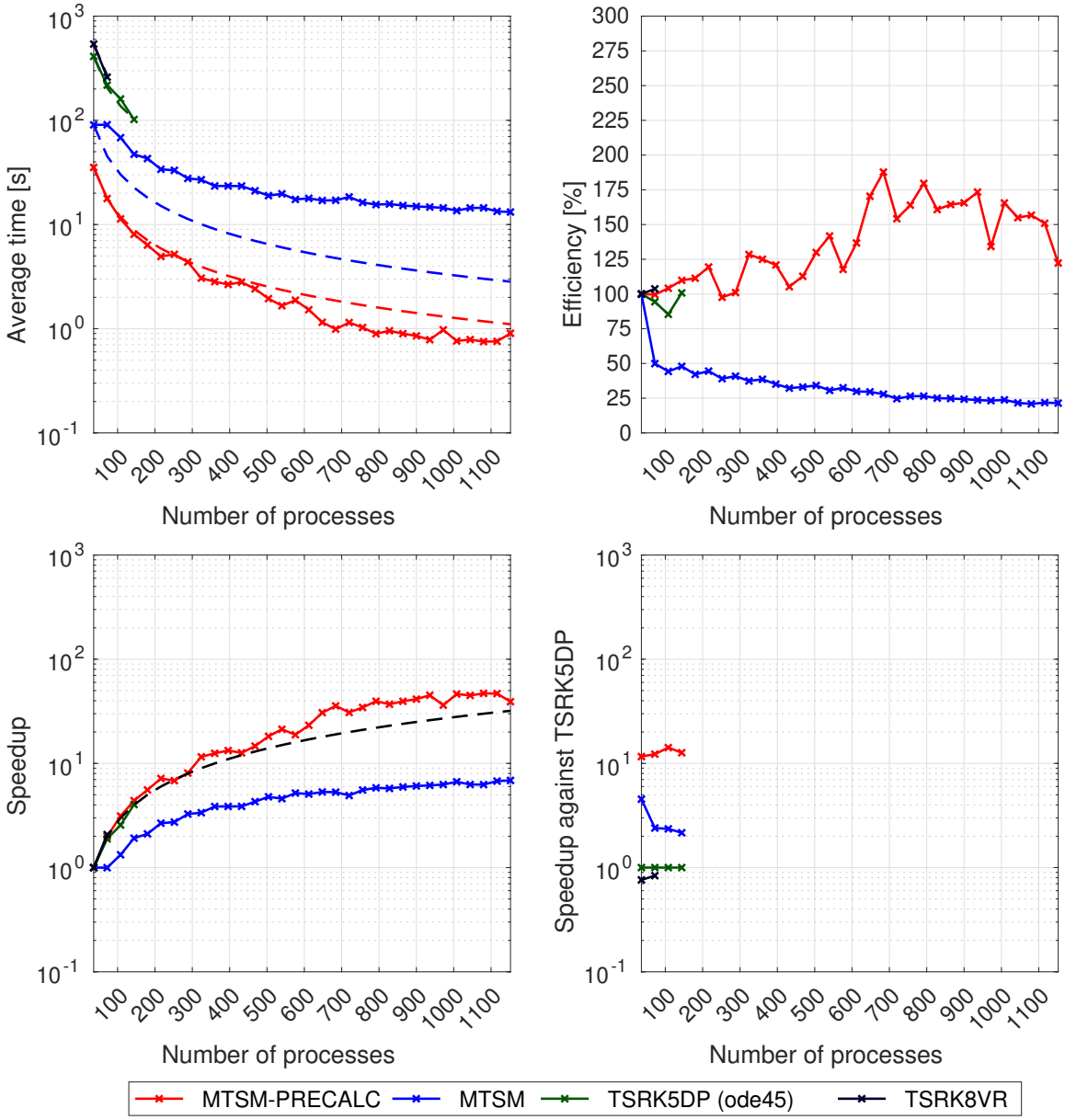


Figure C.11: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 512000$, wave equation, three-point central difference formula.

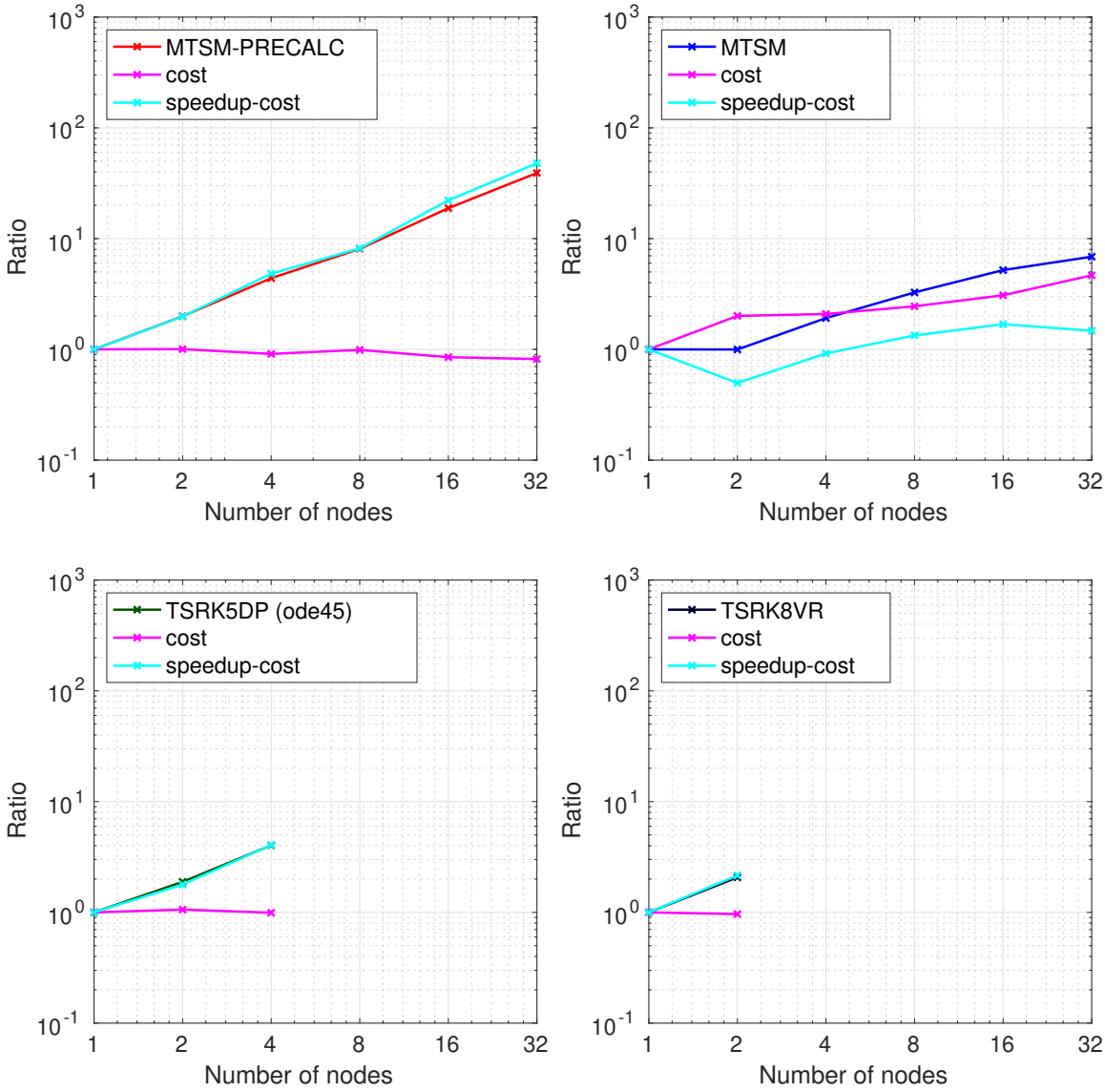


Figure C.12: Parallel cost ratio and speedup-cost ratio, $S = 512000$, wave equation, three-point central difference formula.

C.3.5 $S = 1024000$, three-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	72.68	35.68	23.75	17.78	14.43	11.32	9.31	8.43
MTSM	264.94	245.92	201.78	92.97	88.35	63.93	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	7.31	6.25	5.75	5.12	5.17	4.57	3.96	4.08
MTSM	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	3.58	3.08	3.29	3.83	2.38	2.58	2.44	2.56
MTSM	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	2.13	2.33	1.95	1.98	1.97	1.86	1.58	1.48
MTSM	-	-	-	-	-	-	-	-

Table C.71: Average time, $S = 1024000$, wave equation, three-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**101.85**	**102.03**	**102.19**	**100.76**	**107.01**	**111.52**	**107.73**
MTSM	**100.00**	**53.87**	43.77	**71.25**	**59.98**	**69.07**	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**110.50**	**116.32**	**114.99**	**118.27**	**108.09**	**113.69**	**122.39**	**111.35**
MTSM	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**119.52**	**131.10**	**116.39**	**94.84**	**145.52**	**128.04**	**129.34**	**118.30**
MTSM	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**136.58**	**120.19**	**138.23**	**131.27**	**127.53**	**130.59**	**148.10**	**153.63**
MTSM	-	-	-	-	-	-	-	-

Table C.72: Efficiency, $S = 1024000$, wave equation, three-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.04	3.06	4.09	5.04	6.42	7.81	8.62
MTSM	1.00	1.08	1.31	2.85	3.00	4.14	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	9.95	11.63	12.65	14.19	14.05	15.92	18.36	17.82
MTSM	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	20.32	23.60	22.11	18.97	30.56	28.17	29.75	28.39
MTSM	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	34.14	31.25	37.32	36.76	36.98	39.18	45.91	49.16
MTSM	-	-	-	-	-	-	-	-

Table C.73: Speedup, $S = 1024000$, wave equation, three-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz [%]
57.34	16.38	16.38	4095994	$9.80e^{-05}$	104447248	$2.49e^{-03}$

Table C.74: Characteristics of input data, $S = 1024000$, wave equation, three-point central difference formula.

C.4 Wave equation – five-point central difference formula

Subsections C.4.1–C.4.5 show numerical results for $S = 64000$, $S = 128000$, $S = 256000$, $S = 512000$, and $S = 1024000$, respectively.

C.4.1 $S = 64000$, five-point central difference formula

solver	**# processes (avgtime) [s]**							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	6.44	2.69	1.64	0.99	0.77	0.71	0.67	0.64
MTSM	11.11	14.53	11.33	9.74	8.67	8.16	7.73	7.38
TSRK5DP	25.31	28.30	21.13	16.75	14.86	13.46	10.27	9.00
TSRK8VR	30.71	31.58	23.66	18.11	16.53	14.53	10.97	9.93
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	0.48	0.53	0.54	0.53	0.47	0.37	0.36	0.42
MTSM	7.69	7.44	7.31	6.67	6.97	7.33	7.19	6.33
TSRK5DP	9.46	8.80	8.10	8.31	7.35	7.47	7.33	6.63
TSRK8VR	9.91	8.94	8.85	8.07	8.10	7.96	7.44	7.47
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	0.43	0.35	0.35	0.35	0.34	0.43	0.45	0.50
MTSM	7.30	7.27	9.88	10.52	11.63	10.96	10.35	9.19
TSRK5DP	6.76	6.64	6.67	6.81	6.59	6.47	6.73	6.12
TSRK8VR	7.14	6.88	6.64	6.85	6.95	6.85	6.46	6.47
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	0.36	0.34	0.51	0.45	0.49	0.46	0.55	0.38
MTSM	8.48	9.01	7.66	7.87	7.35	7.45	6.90	7.26
TSRK5DP	6.33	7.08	6.57	6.44	5.99	6.64	6.55	6.78
TSRK8VR	7.07	6.64	6.37	6.26	6.21	6.46	6.53	6.44

Table C.75: Average time, $S = 64000$, wave equation, five-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**119.59**	**131.07**	**162.78**	**168.22**	**151.47**	**138.22**	**126.55**
MTSM	**100.00**	38.22	32.67	28.52	25.61	22.70	20.52	18.83
TSRK5DP	**100.00**	44.73	39.93	37.79	34.08	31.34	35.21	35.14
TSRK8VR	**100.00**	48.62	43.26	42.39	37.17	35.23	39.98	38.66
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**148.04**	**122.42**	**108.63**	**100.71**	**105.14**	**124.20**	**117.97**	**94.83**
MTSM	16.06	14.92	13.81	13.87	12.25	10.82	10.30	10.97
TSRK5DP	29.75	28.78	28.42	25.40	26.50	24.19	23.02	23.87
TSRK8VR	34.43	34.36	31.54	31.72	29.16	27.56	27.50	25.69
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**89.05**	**101.66**	**96.49**	**92.16**	**90.13**	**68.29**	**61.93**	**53.87**
MTSM	8.95	8.49	5.91	5.28	4.55	4.61	4.67	5.04
TSRK5DP	22.01	21.19	19.96	18.59	18.29	17.79	16.35	17.22
TSRK8VR	25.30	24.79	24.33	22.41	21.05	20.38	20.67	19.78
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**71.70**	**73.06**	46.62	**50.56**	45.76	46.25	38.00	**52.87**
MTSM	5.24	4.74	5.37	5.04	5.21	4.97	5.20	4.78
TSRK5DP	15.99	13.75	14.28	14.04	14.58	12.71	12.47	11.67
TSRK8VR	17.37	17.79	17.87	17.51	17.06	15.85	15.17	14.90

Table C.76: Efficiency, $S = 64000$, wave equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.39	3.93	6.51	8.41	9.09	9.68	10.12
MTSM	1.00	0.76	0.98	1.14	1.28	1.36	1.44	1.51
TSRK5DP	1.00	0.89	1.20	1.51	1.70	1.88	2.46	2.81
TSRK8VR	1.00	0.97	1.30	1.70	1.86	2.11	2.80	3.09
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	13.32	12.24	11.95	12.09	13.67	17.39	17.70	15.17
MTSM	1.45	1.49	1.52	1.66	1.59	1.51	1.55	1.75
TSRK5DP	2.68	2.88	3.13	3.05	3.45	3.39	3.45	3.82
TSRK8VR	3.10	3.44	3.47	3.81	3.79	3.86	4.13	4.11
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	15.14	18.30	18.33	18.43	18.93	15.02	14.24	12.93
MTSM	1.52	1.53	1.12	1.06	0.95	1.01	1.07	1.21
TSRK5DP	3.74	3.81	3.79	3.72	3.84	3.91	3.76	4.13
TSRK8VR	4.30	4.46	4.62	4.48	4.42	4.48	4.75	4.75
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	17.93	18.99	12.59	14.16	13.27	13.87	11.78	16.92
MTSM	1.31	1.23	1.45	1.41	1.51	1.49	1.61	1.53
TSRK5DP	4.00	3.58	3.86	3.93	4.23	3.81	3.87	3.73
TSRK8VR	4.34	4.62	4.82	4.90	4.95	4.75	4.70	4.77

Table C.77: Speedup, $S = 64000$, wave equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**3.93**	**10.51**	**12.91**	**16.94**	**19.41**	**19.01**	**15.44**	**14.16**
MTSM	**2.28**	**1.95**	**1.87**	**1.72**	**1.71**	**1.65**	**1.33**	**1.22**
TSRK8VR	0.82	0.90	0.89	0.92	0.90	0.93	0.94	0.91
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**19.57**	**16.73**	**15.03**	**15.59**	**15.60**	**20.19**	**20.15**	**15.62**
MTSM	**1.23**	**1.18**	**1.11**	**1.24**	**1.05**	**1.02**	**1.02**	**1.05**
TSRK8VR	0.95	0.98	0.91	**1.03**	0.91	0.94	0.98	0.89
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**15.91**	**18.86**	**19.01**	**19.49**	**19.38**	**15.10**	**14.90**	**12.30**
MTSM	0.93	0.91	0.68	0.65	0.57	0.59	0.65	0.67
TSRK8VR	0.95	0.96	**1.00**	0.99	0.95	0.94	**1.04**	0.95
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**17.64**	**20.89**	**12.84**	**14.16**	**12.34**	**14.31**	**11.98**	**17.82**
MTSM	0.75	0.79	0.86	0.82	0.81	0.89	0.95	0.93
TSRK8VR	0.90	**1.07**	**1.03**	**1.03**	0.96	**1.03**	**1.00**	**1.05**

Table C.78: Speedup against TSRK5DP, $S = 64000$, wave equation, five-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz [%]
5.12	1.02	1.02	383990	$2.34e^{-03}$	12797306	$7.81e^{-02}$

Table C.79: Characteristics of input data, $S = 64000$, wave equation, five-point central difference formula.

C.4.2 $S = 128000$, five-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	14.70	6.91	4.14	2.69	2.02	1.77	1.37	1.06
MTSM	21.37	25.69	20.59	15.10	14.11	11.66	11.53	10.67
TSRK5DP	59.67	52.21	40.67	29.81	25.68	21.40	20.01	17.30
TSRK8VR	80.54	58.69	48.22	33.63	28.69	24.27	20.71	18.65
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	1.04	0.81	0.83	0.73	0.56	0.61	0.67	0.61
MTSM	10.91	9.74	10.13	9.06	9.77	9.29	9.08	7.86
TSRK5DP	16.88	15.18	14.78	13.75	13.73	10.37	10.06	10.03
TSRK8VR	18.39	16.39	16.47	14.80	14.46	11.61	10.73	10.75
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	0.55	0.51	0.50	0.48	0.47	0.46	0.49	0.43
MTSM	8.89	8.97	8.70	8.73	8.46	8.96	8.53	8.48
TSRK5DP	10.20	9.28	9.42	9.37	9.39	8.94	9.19	8.08
TSRK8VR	10.58	10.23	9.70	9.63	9.50	9.70	9.50	9.06
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	0.43	0.51	0.60	0.51	0.58	0.49	0.55	0.47
MTSM	7.86	8.67	8.70	8.60	8.25	9.18	9.54	8.79
TSRK5DP	8.48	9.01	17.19	14.76	15.22	14.34	16.30	13.50
TSRK8VR	8.51	9.04	8.55	8.84	7.66	8.60	8.52	8.53

Table C.80: Average time, $S = 128000$, wave equation, five-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**106.31**	**118.46**	**136.57**	**145.22**	**138.20**	**153.68**	**174.13**
MTSM	**100.00**	41.58	34.60	35.37	30.29	30.54	26.48	25.04
TSRK5DP	**100.00**	**57.15**	48.91	**50.04**	46.48	46.47	42.61	43.12
TSRK8VR	**100.00**	**68.61**	**55.68**	**59.87**	**56.14**	**55.31**	**55.56**	**53.99**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**157.37**	**182.13**	**161.79**	**168.79**	**201.55**	**173.46**	**146.00**	**149.80**
MTSM	21.77	21.94	19.18	19.66	16.83	16.43	15.69	16.99
TSRK5DP	39.29	39.31	36.70	36.15	33.43	41.10	39.54	37.17
TSRK8VR	48.66	49.14	44.45	45.33	42.85	49.53	**50.03**	46.82
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**157.82**	**160.37**	**155.80**	**152.10**	**147.61**	**145.12**	**130.50**	**142.40**
MTSM	14.14	13.23	12.93	12.23	12.03	10.85	10.89	10.50
TSRK5DP	34.41	35.74	33.32	31.84	30.26	30.32	28.22	30.79
TSRK8VR	44.79	43.75	43.69	41.80	40.38	37.76	36.86	37.05
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**136.13**	**111.45**	**90.57**	**102.77**	**87.27**	**99.76**	**85.53**	**96.74**
MTSM	10.88	9.48	9.10	8.87	8.94	7.76	7.22	7.59
TSRK5DP	28.15	25.47	12.86	14.44	13.52	13.87	11.81	13.81
TSRK8VR	37.85	34.28	34.87	32.54	36.27	31.20	30.50	29.50

Table C.81: Efficiency, $S = 128000$, wave equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.13	3.55	5.46	7.26	8.29	10.76	13.93
MTSM	1.00	0.83	1.04	1.41	1.51	1.83	1.85	2.00
TSRK5DP	1.00	1.14	1.47	2.00	2.32	2.79	2.98	3.45
TSRK8VR	1.00	1.37	1.67	2.39	2.81	3.32	3.89	4.32
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	14.16	18.21	17.80	20.25	26.20	24.28	21.90	23.97
MTSM	1.96	2.19	2.11	2.36	2.19	2.30	2.35	2.72
TSRK5DP	3.54	3.93	4.04	4.34	4.35	5.75	5.93	5.95
TSRK8VR	4.38	4.91	4.89	5.44	5.57	6.93	7.50	7.49
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	26.83	28.87	29.60	30.42	31.00	31.93	30.02	34.18
MTSM	2.40	2.38	2.46	2.45	2.53	2.39	2.50	2.52
TSRK5DP	5.85	6.43	6.33	6.37	6.35	6.67	6.49	7.39
TSRK8VR	7.62	7.87	8.30	8.36	8.48	8.31	8.48	8.89
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	34.03	28.98	24.45	28.78	25.31	29.93	26.51	30.96
MTSM	2.72	2.47	2.46	2.48	2.59	2.33	2.24	2.43
TSRK5DP	7.04	6.62	3.47	4.04	3.92	4.16	3.66	4.42
TSRK8VR	9.46	8.91	9.42	9.11	10.52	9.36	9.45	9.44

Table C.82: Speedup, $S = 128000$, wave equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**4.06**	**7.55**	**9.83**	**11.08**	**12.68**	**12.07**	**14.64**	**16.39**
MTSM	**2.79**	**2.03**	**1.98**	**1.97**	**1.82**	**1.84**	**1.74**	**1.62**
TSRK8VR	0.74	0.89	0.84	0.89	0.89	0.88	0.97	0.93
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**16.26**	**18.81**	**17.90**	**18.95**	**24.47**	**17.13**	**14.99**	**16.36**
MTSM	**1.55**	**1.56**	**1.46**	**1.52**	**1.41**	**1.12**	**1.11**	**1.28**
TSRK8VR	0.92	0.93	0.90	0.93	0.95	0.89	0.94	0.93
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**18.62**	**18.21**	**18.98**	**19.39**	**19.80**	**19.43**	**18.77**	**18.78**
MTSM	**1.15**	**1.03**	**1.08**	**1.07**	**1.11**	1.00	**1.08**	0.95
TSRK8VR	0.96	0.91	0.97	0.97	0.99	0.92	0.97	0.89
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**19.63**	**17.76**	**28.60**	**28.90**	**26.20**	**29.20**	**29.40**	**28.43**
MTSM	**1.08**	**1.04**	**1.98**	**1.72**	**1.85**	**1.56**	**1.71**	**1.54**
TSRK8VR	1.00	1.00	**2.01**	**1.67**	**1.99**	**1.67**	**1.91**	**1.58**

Table C.83: Speedup against TSRK5DP, $S = 128000$, wave equation, five-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A_{PRECALC}}$ nnz	$\mathbf{A_{PRECALC}}$ nnz [%]
10.24	2.05	2.05	767990	$1.17e^{-03}$	25597306	$3.91e^{-02}$

Table C.84: Characteristics of input data, $S = 128000$, wave equation, five-point central difference formula.

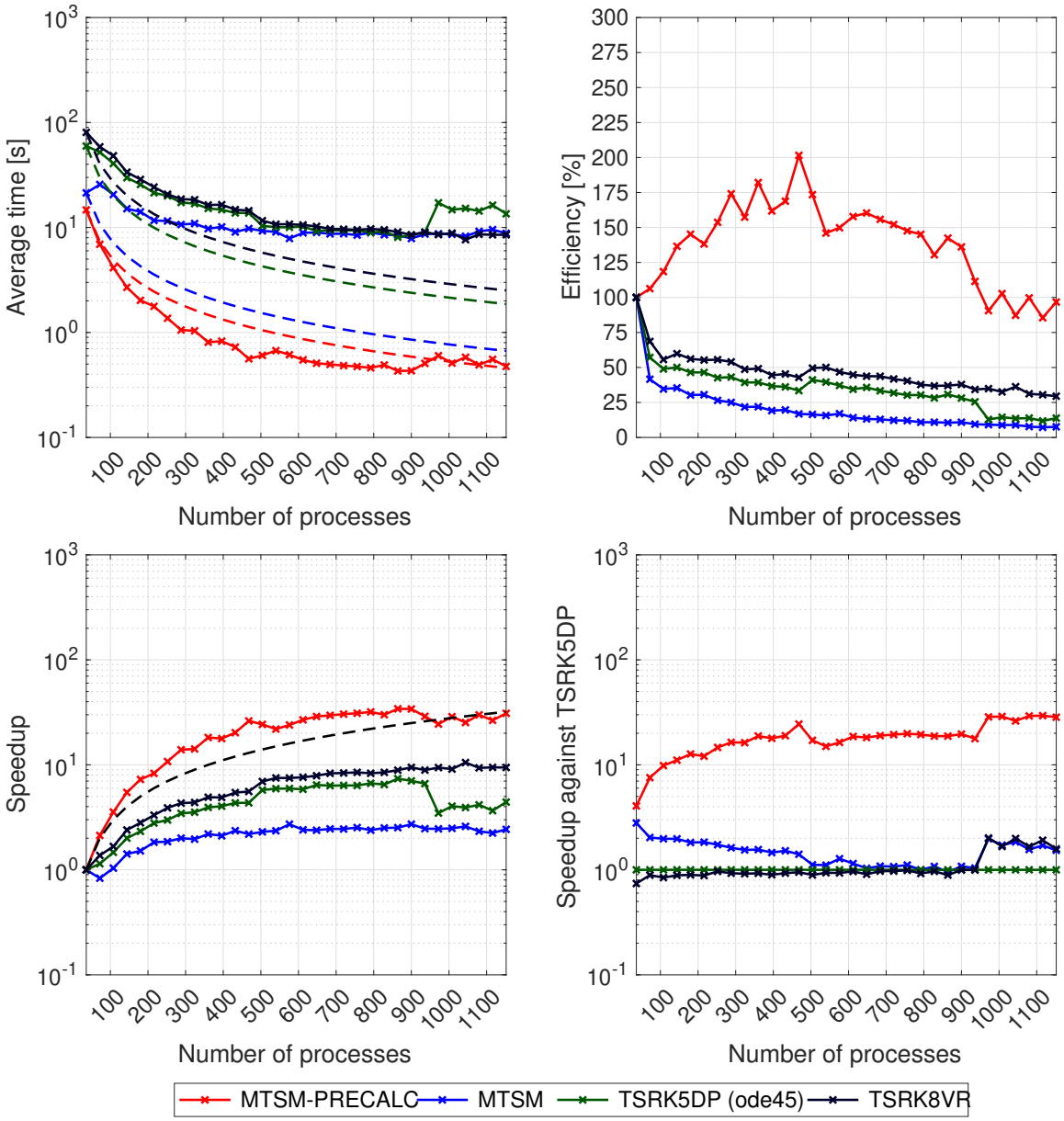


Figure C.13: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 128000$, wave equation, five-point central difference formula.

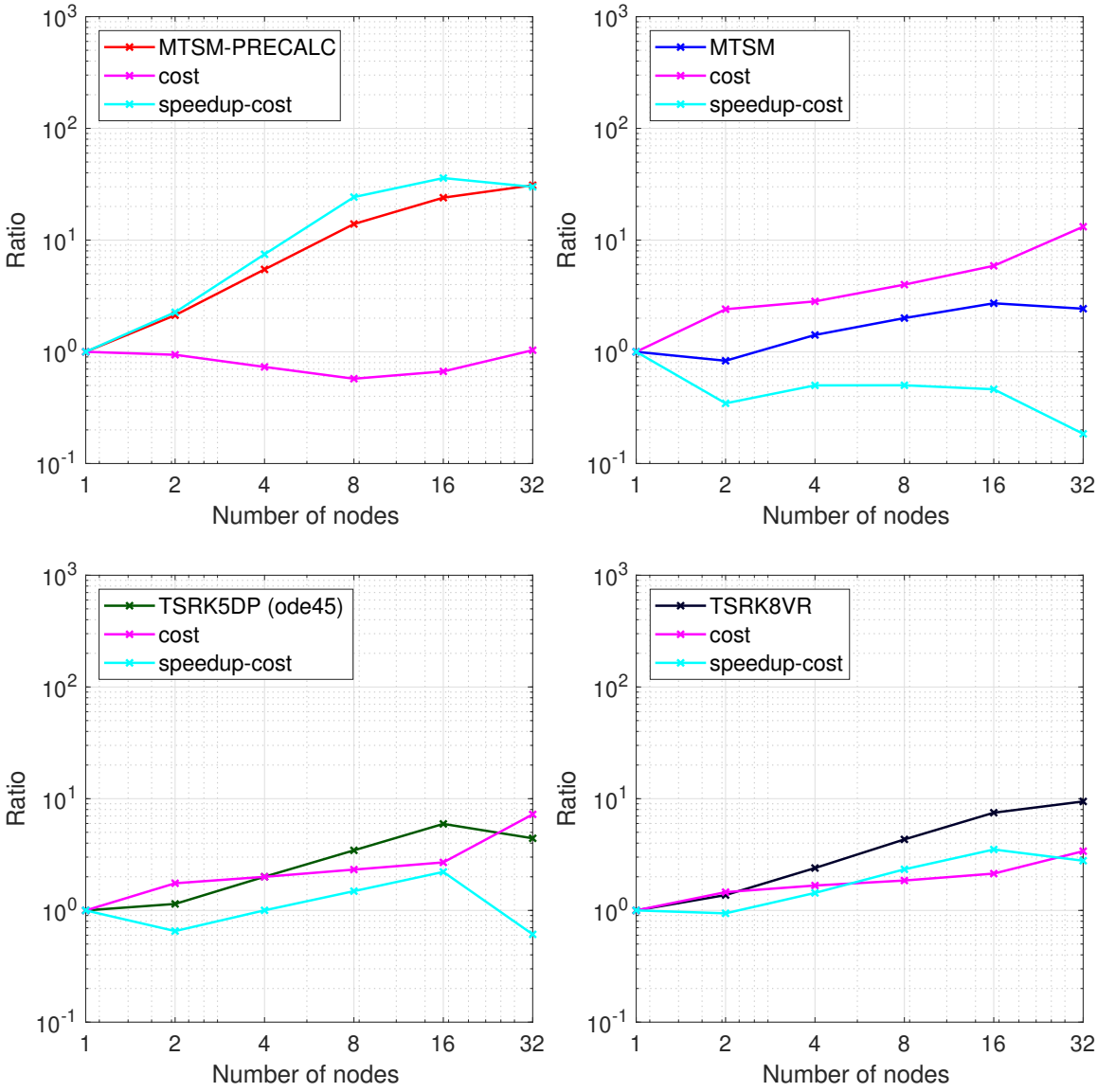


Figure C.14: Parallel cost ratio and speedup-cost ratio, $S = 128000$, wave equation, five-point central difference formula.

C.4.3 $S = 256000$, five-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	31.61	15.40	9.83	6.92	5.38	4.19	3.44	2.80
MTSM	43.80	48.09	36.07	26.70	24.32	19.96	19.02	16.64
TSRK5DP	169.60	104.01	85.03	55.26	52.74	39.65	38.25	31.14
TSRK8VR	227.85	122.39	95.51	63.50	60.57	44.75	41.58	34.23
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	2.51	2.06	1.88	1.86	1.49	1.60	1.25	1.16
MTSM	17.27	14.84	15.30	13.45	13.14	13.17	13.13	11.60
TSRK5DP	31.30	25.97	25.71	23.24	23.20	20.57	20.57	18.83
TSRK8VR	34.09	28.09	28.08	24.86	25.11	21.33	22.34	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	0.94	1.03	0.89	1.04	0.91	0.86	0.84	0.88
MTSM	12.94	12.20	11.99	24.80	21.32	20.71	20.53	20.15
TSRK5DP	18.47	17.92	18.04	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	0.91	0.82	0.84	0.73	0.78	0.81	0.78	0.76
MTSM	23.48	20.01	22.33	19.71	20.62	19.00	18.87	16.58
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.85: Average time, $S = 256000$, wave equation, five-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**102.65**	**107.19**	**114.15**	**117.57**	**125.87**	**131.17**	**140.98**
MTSM	**100.00**	45.55	40.48	41.01	36.02	36.58	32.89	32.90
TSRK5DP	**100.00**	**81.53**	**66.49**	**76.73**	**64.31**	**71.28**	**63.35**	**68.08**
TSRK8VR	**100.00**	**93.08**	**79.52**	**89.70**	**75.24**	**84.86**	**78.28**	**83.21**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**139.76**	**153.23**	**152.68**	**141.56**	**163.20**	**140.73**	**168.81**	**170.66**
MTSM	28.19	29.52	26.02	27.15	25.64	23.76	22.24	23.61
TSRK5DP	**60.21**	**65.30**	**59.98**	**60.82**	**56.22**	**58.89**	**54.96**	**56.29**
TSRK8VR	**74.27**	**81.13**	**73.78**	**76.38**	**69.80**	**76.29**	**68.01**	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**198.72**	**170.23**	**187.82**	**151.35**	**165.54**	**166.78**	**162.83**	**149.49**
MTSM	19.91	19.94	19.22	8.83	9.78	9.61	9.28	9.06
TSRK5DP	**54.01**	**52.58**	49.49	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**138.98**	**147.58**	**138.59**	**154.45**	**139.80**	**129.55**	**130.72**	**129.98**
MTSM	7.46	8.42	7.26	7.94	7.33	7.69	7.49	8.26
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.86: Efficiency, $S = 256000$, wave equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.05	3.22	4.57	5.88	7.55	9.18	11.28
MTSM	1.00	0.91	1.21	1.64	1.80	2.19	2.30	2.63
TSRK5DP	1.00	1.63	1.99	3.07	3.22	4.28	4.43	5.45
TSRK8VR	1.00	1.86	2.39	3.59	3.76	5.09	5.48	6.66
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	12.58	15.32	16.79	16.99	21.22	19.70	25.32	27.31
MTSM	2.54	2.95	2.86	3.26	3.33	3.33	3.34	3.78
TSRK5DP	5.42	6.53	6.60	7.30	7.31	8.25	8.24	9.01
TSRK8VR	6.68	8.11	8.12	9.17	9.07	10.68	10.20	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	33.78	30.64	35.69	30.27	34.76	36.69	37.45	35.88
MTSM	3.38	3.59	3.65	1.77	2.05	2.11	2.13	2.17
TSRK5DP	9.18	9.46	9.40	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	34.75	38.37	37.42	43.25	40.54	38.87	40.52	41.59
MTSM	1.87	2.19	1.96	2.22	2.12	2.31	2.32	2.64
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.87: Speedup, $S = 256000$, wave equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**5.36**	**6.75**	**8.65**	**7.98**	**9.81**	**9.47**	**11.11**	**11.11**
MTSM	**3.87**	**2.16**	**2.36**	**2.07**	**2.17**	**1.99**	**2.01**	**1.87**
TSRK8VR	0.74	0.85	0.89	0.87	0.87	0.89	0.92	0.91
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**12.45**	**12.59**	**13.66**	**12.49**	**15.57**	**12.82**	**16.48**	**16.26**
MTSM	**1.81**	**1.75**	**1.68**	**1.73**	**1.77**	**1.56**	**1.57**	**1.62**
TSRK8VR	0.92	0.92	0.92	0.93	0.92	0.96	0.92	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**19.74**	**17.37**	**20.36**	-	-	-	-	-
MTSM	**1.43**	**1.47**	**1.50**	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	-	-	-	-	-	-	-	-
MTSM	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.88: Speedup against TSRK5DP, $S = 256000$, wave equation, five-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A_{PRECALC}}$ nnz	$\mathbf{A_{PRECALC}}$ nnz [%]
20.48	4.10	4.10	1535990	$5.86e^{-04}$	51197306	$1.95e^{-02}$

Table C.89: Characteristics of input data, $S = 256000$, wave equation, five-point central difference formula.

C.4.4 $S = 512000$, five-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	64.69	32.69	21.49	15.47	12.13	9.91	8.31	7.18
MTSM	114.01	95.65	69.87	49.58	43.82	35.40	34.23	28.34
TSRK5DP	506.32	243.82	170.10	-	-	-	-	-
TSRK8VR	621.15	291.84	-	-	-	-	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	6.32	5.52	5.08	4.42	3.95	3.63	3.26	3.09
MTSM	28.87	24.53	25.38	21.69	21.59	20.18	19.82	18.71
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	2.69	2.53	2.59	2.37	2.31	2.22	1.95	1.85
MTSM	20.44	35.05	36.67	30.84	-	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	1.73	1.67	1.60	1.52	1.48	1.50	1.20	1.22
MTSM	-	-	-	-	-	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.90: Average time, $S = 512000$, wave equation, five-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**98.94**	**100.34**	**104.57**	**106.64**	**108.82**	**111.16**	**112.56**
MTSM	**100.00**	**59.60**	**54.39**	**57.49**	**52.04**	**53.68**	47.58	**50.29**
TSRK5DP	**100.00**	**103.83**	**99.22**	-	-	-	-	-
TSRK8VR	**100.00**	**106.42**	-	-	-	-	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**113.79**	**117.28**	**115.79**	**121.85**	**125.99**	**127.37**	**132.20**	**130.89**
MTSM	43.87	46.49	40.84	43.80	40.63	40.37	38.36	38.08
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**141.48**	**142.12**	**131.42**	**136.75**	**133.51**	**132.16**	**144.16**	**145.57**
MTSM	32.82	18.07	16.37	18.49	-	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**149.64**	**149.07**	**149.81**	**151.57**	**150.76**	**143.78**	**173.81**	**165.41**
MTSM	-	-	-	-	-	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.91: Efficiency, $S = 512000$, wave equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	1.98	3.01	4.18	5.33	6.53	7.78	9.00
MTSM	1.00	1.19	1.63	2.30	2.60	3.22	3.33	4.02
TSRK5DP	1.00	2.08	2.98	-	-	-	-	-
TSRK8VR	1.00	2.13	-	-	-	-	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	10.24	11.73	12.74	14.62	16.38	17.83	19.83	20.94
MTSM	3.95	4.65	4.49	5.26	5.28	5.65	5.75	6.09
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	24.05	25.58	24.97	27.35	28.04	29.07	33.16	34.94
MTSM	5.58	3.25	3.11	3.70	-	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	37.41	38.76	40.45	42.44	43.72	43.13	53.88	52.93
MTSM	-	-	-	-	-	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.92: Speedup, $S = 512000$, wave equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**7.83**	**7.46**	**7.92**	-	-	-	-	-
MTSM	**4.44**	**2.55**	**2.43**	-	-	-	-	-
TSRK8VR	0.82	0.84	-	-	-	-	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	-	-	-	-	-	-	-	-
MTSM	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	-	-	-	-	-	-	-	-
MTSM	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	-	-	-	-	-	-	-	-
MTSM	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.93: Speedup against TSRK5DP, $S = 512000$, wave equation, five-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A_{PRECALC}}$ nnz	$\mathbf{A_{PRECALC}}$ nnz [%]
40.96	8.19	8.19	3071990	$2.93e^{-04}$	102397306	$9.76e^{-03}$

Table C.94: Characteristics of input data, $S = 512000$, wave equation, five-point central difference formula.

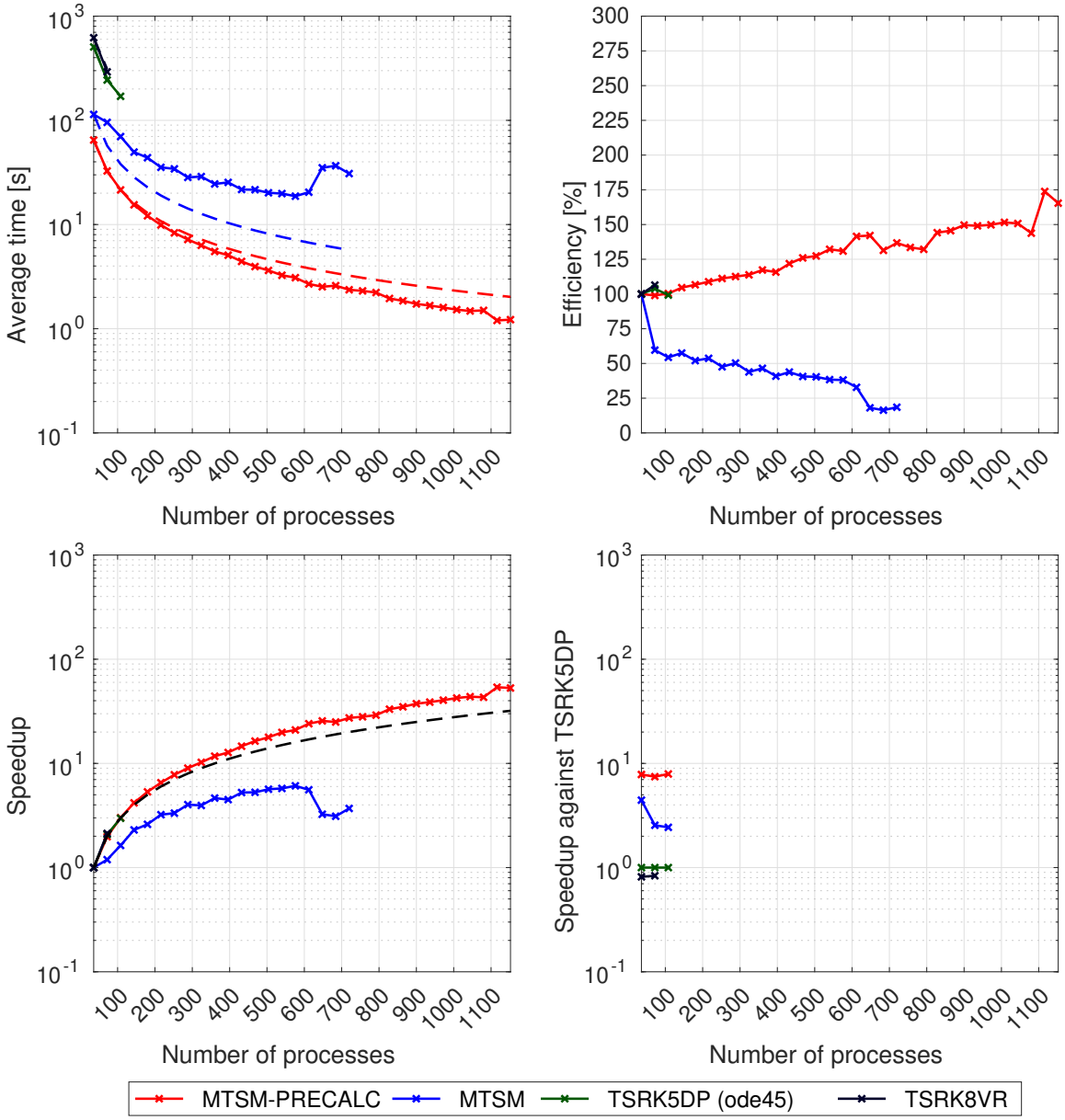


Figure C.15: Average time, parallel efficiency, parallel speedup, speedup against the TSRK5DP solver, $S = 512000$, wave equation, five-point central difference formula.

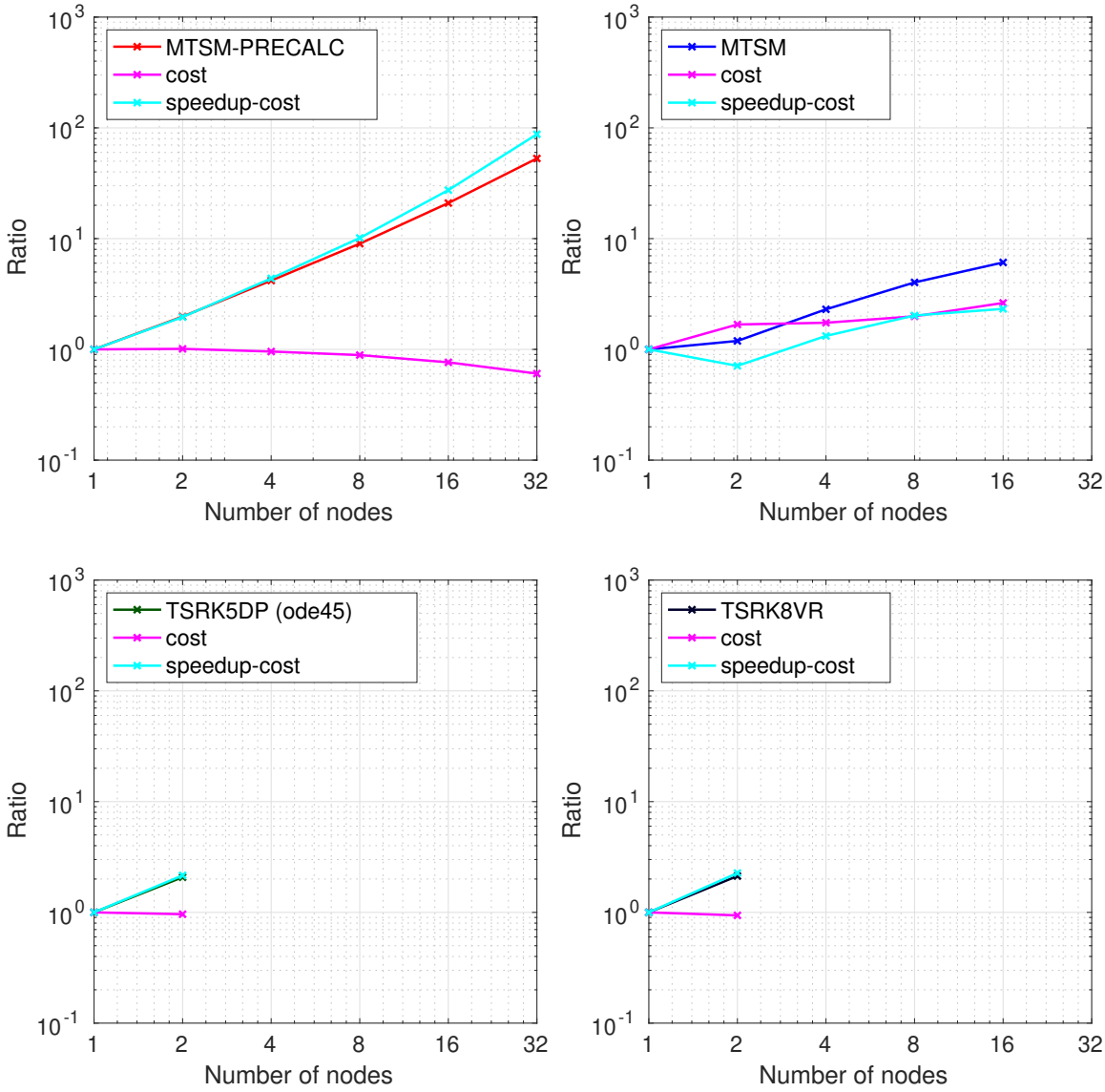


Figure C.16: Parallel cost ratio and speedup-cost ratio, $S = 512000$, wave equation, five-point central difference formula.

C.4.5 $S = 1024000$, five-point central difference formula

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	72.44	35.63	23.19	17.60	13.74	11.18	9.25	8.05
MTSM	254.89	240.55	193.09	91.40	81.92	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	8.84	6.04	5.39	4.90	4.38	4.21	3.79	3.40
MTSM	-	-	-	-	-	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	3.09	3.05	2.57	2.44	2.25	2.15	2.01	1.93
MTSM	-	-	-	-	-	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	1.88	1.92	1.76	1.69	1.80	1.47	1.51	1.31
MTSM	-	-	-	-	-	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.95: Average time, $S = 1024000$, wave equation, five-point central difference formula.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**101.66**	**104.14**	**102.89**	**105.41**	**107.97**	**111.91**	**112.52**
MTSM	**100.00**	**52.98**	44.00	**69.72**	**62.23**	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**91.09**	**119.85**	**122.13**	**123.23**	**127.23**	**122.83**	**127.35**	**133.13**
MTSM	-	-	-	-	-	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**138.07**	**132.05**	**148.16**	**148.35**	**153.60**	**153.45**	**156.73**	**156.79**
MTSM	-	-	-	-	-	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**154.02**	**145.13**	**152.81**	**153.03**	**138.73**	**164.53**	**154.34**	**173.30**
MTSM	-	-	-	-	-	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.96: Efficiency, $S = 1024000$, wave equation, five-point central difference formula.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	2.03	3.12	4.12	5.27	6.48	7.83	9.00
MTSM	1.00	1.06	1.32	2.79	3.11	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	8.20	11.99	13.43	14.79	16.54	17.20	19.10	21.30
MTSM	-	-	-	-	-	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	23.47	23.77	28.15	29.67	32.26	33.76	36.05	37.63
MTSM	-	-	-	-	-	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	38.51	37.73	41.26	42.85	40.23	49.36	47.85	55.45
MTSM	-	-	-	-	-	-	-	-
TSRK5DP	-	-	-	-	-	-	-	-
TSRK8VR	-	-	-	-	-	-	-	-

Table C.97: Speedup, $S = 1024000$, wave equation, five-point central difference formula.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz	$\mathbf{A}_{\mathbf{PRECALC}}$ nnz [%]
57.34	16.38	16.38	4095994	$9.80e^{-05}$	104447248	$2.49e^{-03}$

Table C.98: Characteristics of input data, $S = 1024000$, wave equation, five-point central difference formula.

C.5 Telegraph equation

Subsections C.5.1 and C.5.2 present numerical results for $S = 512000$, and $S = 1024000$, respectively.

C.5.1 $S = 512000$

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM	98.42	81.63	62.67	42.00	39.69	31.87	30.12	25.94
MTSM_PRECALC	89.21	45.29	30.10	22.11	17.59	14.32	12.06	10.32
TSRK5DP	265.34	135.59	97.49	61.05	59.81	43.61	43.76	32.97
TSRK8VR	187.69	87.64	62.89	38.49	38.73	26.61	23.80	19.82
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM	25.87	23.43	21.76	20.67	19.75	19.49	18.16	16.87
MTSM_PRECALC	9.33	8.17	7.29	6.51	5.94	5.59	5.16	5.04
TSRK5DP	35.36	28.20	29.92	24.38	25.47	21.64	22.12	19.70
TSRK8VR	22.33	16.95	18.63	14.45	16.04	13.23	13.21	12.00
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM	16.70	16.68	16.55	16.16	15.16	15.12	14.87	14.29
MTSM_PRECALC	4.53	4.16	3.97	3.71	3.42	3.25	3.11	3.02
TSRK5DP	19.37	18.10	17.66	16.70	16.19	15.84	15.15	14.36
TSRK8VR	12.03	10.59	10.58	9.97	9.30	10.79	10.35	9.70
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM	14.51	13.41	13.08	13.40	13.20	13.63	12.60	12.33
MTSM_PRECALC	2.90	2.68	2.61	2.54	2.59	2.44	2.22	2.15
TSRK5DP	14.23	12.76	13.18	16.68	15.81	15.75	14.93	15.51
TSRK8VR	10.08	8.92	9.47	9.03	8.88	8.83	9.25	10.26

Table C.99: Average time, $S = 512000$, telegraph equation.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM	**100.00**	**60.28**	**52.35**	**58.59**	49.59	**51.47**	46.68	47.43
MTSM_PRECALC	**100.00**	**98.49**	**98.79**	**100.87**	**101.43**	**103.83**	**105.67**	**108.05**
TSRK5DP	**100.00**	**97.85**	**90.72**	**108.66**	**88.73**	**101.41**	**86.62**	**100.60**
TSRK8VR	**100.00**	**107.08**	**99.48**	**121.91**	**96.92**	**117.54**	**112.67**	**118.38**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM	42.27	42.01	41.12	39.68	38.33	36.07	36.13	36.46
MTSM_PRECALC	**106.24**	**109.19**	**111.25**	**114.20**	**115.53**	**113.99**	**115.26**	**110.63**
TSRK5DP	**83.38**	**94.09**	**80.62**	**90.70**	**80.14**	**87.58**	**79.97**	**84.18**
TSRK8VR	**93.41**	**110.76**	**91.60**	**108.22**	**90.01**	**101.36**	**94.69**	**97.79**
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM	34.67	32.78	31.30	30.45	30.91	29.59	28.78	28.70
MTSM_PRECALC	**115.84**	**119.14**	**118.27**	**120.23**	**124.21**	**124.77**	**124.72**	**123.08**
TSRK5DP	**80.58**	**81.44**	**79.08**	**79.44**	**78.04**	**76.14**	**76.15**	**76.99**
TSRK8VR	**91.76**	**98.45**	**93.35**	**94.11**	**96.09**	**79.09**	**78.87**	**80.59**
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM	27.13	28.23	27.87	26.23	25.71	24.07	25.20	24.94
MTSM_PRECALC	**123.05**	**128.03**	**126.59**	**125.44**	**118.77**	**121.87**	**129.63**	**129.67**
TSRK5DP	**74.59**	**79.98**	**74.56**	**56.81**	**57.87**	**56.16**	**57.33**	**53.46**
TSRK8VR	**74.47**	**80.92**	**73.38**	**74.25**	**72.88**	**70.86**	**65.45**	**57.16**

Table C.100: Efficiency, $S = 512000$, telegraph equation.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM	1.00	1.21	1.57	2.34	2.48	3.09	3.27	3.79
MTSM_PRECALC	1.00	1.97	2.96	4.03	5.07	6.23	7.40	8.64
TSRK5DP	1.00	1.96	2.72	4.35	4.44	6.08	6.06	8.05
TSRK8VR	1.00	2.14	2.98	4.88	4.85	7.05	7.89	9.47
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM	3.80	4.20	4.52	4.76	4.98	5.05	5.42	5.83
MTSM_PRECALC	9.56	10.92	12.24	13.70	15.02	15.96	17.29	17.70
TSRK5DP	7.50	9.41	8.87	10.88	10.42	12.26	12.00	13.47
TSRK8VR	8.41	11.08	10.08	12.99	11.70	14.19	14.20	15.65
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM	5.89	5.90	5.95	6.09	6.49	6.51	6.62	6.89
MTSM_PRECALC	19.69	21.44	22.47	24.05	26.08	27.45	28.68	29.54
TSRK5DP	13.70	14.66	15.02	15.89	16.39	16.75	17.51	18.48
TSRK8VR	15.60	17.72	17.74	18.82	20.18	17.40	18.14	19.34
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM	6.78	7.34	7.52	7.34	7.46	7.22	7.81	7.98
MTSM_PRECALC	30.76	33.29	34.18	35.12	34.44	36.56	40.18	41.49
TSRK5DP	18.65	20.79	20.13	15.91	16.78	16.85	17.77	17.11
TSRK8VR	18.62	21.04	19.81	20.79	21.14	21.26	20.29	18.29

Table C.101: Speedup, $S = 512000$, telegraph equation.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM	**2.70**	**1.66**	**1.56**	**1.45**	**1.51**	**1.37**	**1.45**	**1.27**
MTSM_PRECALC	**2.97**	**2.99**	**3.24**	**2.76**	**3.40**	**3.05**	**3.63**	**3.19**
TSRK8VR	**1.41**	**1.55**	**1.55**	**1.59**	**1.54**	**1.64**	**1.84**	**1.66**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM	**1.37**	**1.20**	**1.38**	**1.18**	**1.29**	**1.11**	**1.22**	**1.17**
MTSM_PRECALC	**3.79**	**3.45**	**4.10**	**3.75**	**4.29**	**3.87**	**4.29**	**3.91**
TSRK8VR	**1.58**	**1.66**	**1.61**	**1.69**	**1.59**	**1.64**	**1.67**	**1.64**
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM	**1.16**	**1.09**	**1.07**	**1.03**	**1.07**	**1.05**	**1.02**	**1.00**
MTSM_PRECALC	**4.28**	**4.35**	**4.45**	**4.50**	**4.73**	**4.87**	**4.87**	**4.75**
TSRK8VR	**1.61**	**1.71**	**1.67**	**1.67**	**1.74**	**1.47**	**1.46**	**1.48**
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM	0.98	0.95	**1.01**	**1.24**	**1.20**	**1.16**	**1.18**	**1.26**
MTSM_PRECALC	**4.91**	**4.76**	**5.05**	**6.57**	**6.10**	**6.45**	**6.73**	**7.21**
TSRK8VR	**1.41**	**1.43**	**1.39**	**1.85**	**1.78**	**1.78**	**1.61**	**1.51**

Table C.102: Speedup against TSRK5DP, $S = 512000$, telegraph equation.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	**A**$_{\text{PRECALC}}$ nnz	**A**$_{\text{PRECALC}}$ nnz [%]
28.67	8.19	8.19	2048003	$1.95e^{-04}$	132091971	$1.26e^{-02}$

Table C.103: Characteristics of input data, $S = 512000$, telegraph equation.

C.5.2 $S = 1024000$

solver	# processes (avgtime) [s]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	100.00	89.60	59.03	45.40	36.47	30.32	25.88	22.03
MTSM	262.97	222.88	177.82	83.04	77.04	54.98	53.36	42.69
TSRK5DP	616.89	348.31	231.40	124.85	113.21	77.27	76.99	57.96
TSRK8VR	430.33	224.96	152.10	85.13	72.87	50.79	52.13	36.78
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	19.91	17.35	15.86	14.56	13.26	12.29	11.33	10.60
MTSM	43.39	37.74	36.49	33.32	32.09	30.16	29.18	26.48
TSRK5DP	63.06	47.51	52.26	39.88	43.09	34.13	38.85	30.53
TSRK8VR	40.35	29.65	32.17	25.46	27.27	21.30	23.41	19.09
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	9.97	9.04	8.65	8.15	7.43	7.03	6.80	6.40
MTSM	25.82	25.12	24.54	23.81	22.18	22.13	21.59	21.03
TSRK5DP	34.30	29.03	29.25	26.64	26.59	25.51	25.05	24.04
TSRK8VR	21.11	17.77	29.25	26.64	26.59	25.51	25.05	24.04
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	6.12	5.80	5.48	5.30	5.21	4.92	4.72	4.55
MTSM	20.20	20.25	19.50	18.92	19.07	19.46	18.20	16.71
TSRK5DP	23.63	21.94	21.52	24.52	22.57	21.97	21.40	20.73
TSRK8VR	23.63	21.94	21.52	24.52	22.57	21.97	21.40	20.73

Table C.104: Average time, $S = 1024000$, telegraph equation.

solver	# processes (efficiency) [%]							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**100.00**	**55.81**	**56.47**	**55.07**	**54.84**	**54.97**	**55.19**	**56.73**
MTSM	**100.00**	**58.99**	49.30	**79.17**	**68.27**	**79.72**	**70.41**	**76.99**
TSRK5DP	**100.00**	**88.56**	**88.86**	**123.53**	**108.98**	**133.06**	**114.47**	**133.04**
TSRK8VR	**100.00**	**95.65**	**94.31**	**126.37**	**118.12**	**141.21**	**117.93**	**146.25**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**55.79**	**57.65**	**57.31**	**57.24**	**58.02**	**58.11**	**58.82**	**58.94**
MTSM	**67.35**	**69.68**	**65.52**	**65.77**	**63.05**	**62.27**	**60.08**	**62.06**
TSRK5DP	**108.69**	**129.84**	**107.32**	**128.90**	**110.13**	**129.10**	**105.86**	**126.27**
TSRK8VR	**118.51**	**145.13**	**121.62**	**140.84**	**121.38**	**144.33**	**122.53**	**140.90**
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**59.00**	**61.45**	**60.81**	**61.32**	**64.08**	**64.70**	**63.90**	**65.07**
MTSM	**59.91**	**58.16**	**56.41**	**55.21**	**56.45**	**54.01**	**52.95**	**52.11**
TSRK5DP	**105.79**	**118.05**	**111.02**	**115.77**	**110.47**	**109.90**	**107.07**	**106.93**
TSRK8VR	**119.89**	**134.57**	**77.44**	**80.76**	**77.06**	**76.67**	**74.69**	**74.59**
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**65.32**	**66.28**	**67.63**	**67.32**	**66.14**	**67.81**	**68.29**	**68.73**
MTSM	**52.08**	49.94	49.95	49.65	47.55	45.04	46.61	49.18
TSRK5DP	**104.41**	**108.15**	**106.16**	**89.84**	**94.25**	**93.60**	**92.98**	**93.00**
TSRK8VR	**72.84**	**75.44**	**74.06**	**62.67**	**65.75**	**65.30**	**64.86**	**64.88**

Table C.105: Efficiency, $S = 1024000$, telegraph equation.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	1.00	1.12	1.69	2.20	2.74	3.30	3.86	4.54
MTSM	1.00	1.18	1.48	3.17	3.41	4.78	4.93	6.16
TSRK5DP	1.00	1.77	2.67	4.94	5.45	7.98	8.01	10.64
TSRK8VR	1.00	1.91	2.83	5.05	5.91	8.47	8.26	11.70
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	5.02	5.77	6.30	6.87	7.54	8.14	8.82	9.43
MTSM	6.06	6.97	7.21	7.89	8.20	8.72	9.01	9.93
TSRK5DP	9.78	12.98	11.81	15.47	14.32	18.07	15.88	20.20
TSRK8VR	10.67	14.51	13.38	16.90	15.78	20.21	18.38	22.54
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	10.03	11.06	11.55	12.26	13.46	14.23	14.70	15.62
MTSM	10.18	10.47	10.72	11.04	11.85	11.88	12.18	12.51
TSRK5DP	17.98	21.25	21.09	23.15	23.20	24.18	24.62	25.66
TSRK8VR	20.38	24.22	14.71	16.15	16.18	16.87	17.18	17.90
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	16.33	17.23	18.26	18.85	19.18	20.34	21.17	21.99
MTSM	13.02	12.98	13.49	13.90	13.79	13.51	14.45	15.74
TSRK5DP	26.10	28.12	28.66	25.16	27.33	28.08	28.82	29.76
TSRK8VR	18.21	19.61	20.00	17.55	19.07	19.59	20.11	20.76

Table C.106: Speedup, $S = 1024000$, telegraph equation.

solver	# processes (ratio)							
	36	**72**	**108**	**144**	**180**	**216**	**252**	**288**
MTSM_PRECALC	**6.17**	**3.89**	**3.92**	**2.75**	**3.10**	**2.55**	**2.97**	**2.63**
MTSM	**2.35**	**1.56**	**1.30**	**1.50**	**1.47**	**1.41**	**1.44**	**1.36**
TSRK8VR	**1.43**	**1.55**	**1.52**	**1.47**	**1.55**	**1.52**	**1.48**	**1.58**
	324	**360**	**396**	**432**	**468**	**504**	**540**	**576**
MTSM_PRECALC	**3.17**	**2.74**	**3.29**	**2.74**	**3.25**	**2.78**	**3.43**	**2.88**
MTSM	**1.45**	**1.26**	**1.43**	**1.20**	**1.34**	**1.13**	**1.33**	**1.15**
TSRK8VR	**1.56**	**1.60**	**1.62**	**1.57**	**1.58**	**1.60**	**1.66**	**1.60**
	612	**648**	**684**	**720**	**756**	**792**	**828**	**864**
MTSM_PRECALC	**3.44**	**3.21**	**3.38**	**3.27**	**3.58**	**3.63**	**3.68**	**3.75**
MTSM	**1.33**	**1.16**	**1.19**	**1.12**	**1.20**	**1.15**	**1.16**	**1.14**
TSRK8VR	**1.62**	**1.63**	1.00	1.00	1.00	1.00	1.00	1.00
	900	**936**	**972**	**1008**	**1044**	**1080**	**1116**	**1152**
MTSM_PRECALC	**3.86**	**3.78**	**3.93**	**4.62**	**4.33**	**4.47**	**4.53**	**4.56**
MTSM	**1.17**	**1.08**	**1.10**	**1.30**	**1.18**	**1.13**	**1.18**	**1.24**
TSRK8VR	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table C.107: Speedup against TSRK5DP, $S = 1024000$, telegraph equation.

A [MB]	**ic** [MB]	**b** [MB]	**A** nnz	**A** nnz [%]	**A**$_{\text{PRECALC}}$ nnz	**A**$_{\text{PRECALC}}$ nnz [%]
57.34	16.38	16.38	4096003	$9.80e^{-05}$	264187971	$6.30e^{-03}$

Table C.108: Characteristics of input data, $S = 1024000$, telegraph equation.

C.6 Open Access Grant Competitions of IT4Innovations

This section provides detailed information on the Open Access Grant Competitions of IT4Innovations[1] (see Subsections C.6.1 and C.6.2).

C.6.1 24th Open Access Grant Competition OPEN-22-47

The large systems of ODEs (matrix-vector representation) will be tested from the size approximately from 100 thousands up to 2 million. The linear partial differential equations (PDEs) – telegraph equation, wave equation 1D, heat equation, and Heat equation 2D will be transformed into the system of ODEs. The MPI code will run on 1–32 nodes, most often 36 processes per one node. Typical data sizes are 128 000, 256 000, 512 000, 1 024 000, 2 048 000 ODEs. For each data size, five selected solvers will compute the problem, namely, are MTSM (classical implementation of MTSM), MTSM_O2 (MTSM with modified stopping rule), MTSM_PRECALC (parallel precalculation of MTSM), TSRK5DP (Dormand-Prince 5(4) method), and TSRK8VR (Verner Runge-Kutta methods of orders 8(7)). Each test is expected to take about 15 minutes of the wall-clock time. 50 000 core hours are estimated to be required.

- Nodes: 32
- Processes per node: 36
- Total number of processes: $32 \cdot 36 = 1152$
- Wall-Clock Core-Hours (WCH): 0.25 hours (15 min)
- Normalized Core-Hours (NCH), CPU, Barbora: $F = 1.4$
- Estimated number of experiments: 20 (4problems $\cdot$ 5data sizes)
- Number of solvers: 5
- Number of wall-clock hours: $32 \cdot 36 \cdot 0.25 \cdot 1.4 \cdot 20 \cdot 5 = 40320$
- Number of node hours: $40320/1.4/36 = 800$
- Number of estimated core hours: 50 000
- Number of node hours: $50000/1.4/36 \approx 992 \doteq 1000$

[1] https://www.it4i.cz/en/for-users/open-access-competition

C.6.2 25th Open Access Grant Competition OPEN-25-51

The large systems of ODEs (matrix-vector representation) will be tested from approximately 100 thousand up to 2 million equations. The linear partial differential equations (PDEs) – for example, telegraph equation, wave equation 1D/2D, heat equation 1D/2D, will be transformed into the system of ODEs. The MPI code will run on 1–32 nodes, most often 36 processes per one node. Typical data sizes are 128 000, 256 000, 512 000, 1 024 000, 2 048 000 ODEs. For each data size,five selected solvers will compute the problem, namely, the MTSM (classical implementation of MTSM), MTSM_O2 (MTSM with modified stopping rule), MTSM_PRECALC (parallel precalculation of MTSM), TSRK5DP (Dormand-Prince 5(4) method), and TSRK8VR (Verner Runge-Kutta method of orders 8(7)). Each test is expected to take about 15 minutes of the wall-clock time. 1400 node hours are estimated to be required.

- Nodes: 32
- Wall-Clock Core-Hours (WCH): 0.25 hours (15 min)
- Estimated number of experiments: 35 (7problems $\cdot$ 5data sizes)
- Number of solvers: 5
- Number of node hours: $32 \cdot 0.25 \cdot 35 \cdot 5 = 1400$