Adaptive time step control for global-in-time Galerkin-Petrov discretizations of evolution equations in the context of incompressible flows

---

Dissertation

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

Der Fakultät für Mathematik der

Technischen Universität Dortmund

vorgelegt von

Lydia Carmen Wambach

im Dezember 2024

Dissertation

Adaptive time step control for global-in-time Galerkin-Petrov discretizations of evolution equations in the context of incompressible flows

Fakultät für Mathematik
Technische Universität Dortmund

Erstgutachter: Prof. Dr. Stefan Turek
Zweitgutachter: apl. Prof. Dr. Friedhelm Schieweck

Tag der mündlichen Prüfung: 10. 04. 2025

Abstract

The goal of adaptive time step control is to efficiently control the accuracy of a simulation. In terms of accuracy, a higher order time-stepping scheme is advantageous, although it results in a higher computational cost. In particular, High Performance Computing (HPC) focuses on efficient performance by using an ever-increasing number of cores and thus applying numerical schemes in parallel. Therefore, global-in-time adaptive time step control based on a higher order time discretization scheme is of interest, especially in the area of Computational Fluid Dynamics (CFD).

In this work, we use the higher order continuous Galerkin-Petrov method as a time discretization scheme in a global-in-time adaptive time step control. The length of the time steps is controlled by so-called controllers based on error estimators. These errors are estimated by two approximations of the solution of different order. Here, we use a linear post-processing step from the continuous Galerkin-Petrov method with low computational cost to obtain a higher order solution. Although this is common in the literature on no-pressure problem types, we present a velocity and a new pressure error estimator for the Navier-Stokes equations. These error estimators approximate the analytic errors suitable, as we show for the heat equation and for incompressible flows as in the Navier-Stokes equations. Especially, for the flow around a cylinder benchmark for Newtonian and non-Newtonian fluids, we present error estimators for the lift and drag coefficients. All numerical tests in this thesis have been implemented in the FEAT3 software.

We also introduce a new global-in-time adaptive strategy. This provides a time-parallel approach. In order to deepen this topic, we briefly introduce two existing global-in-time approaches and discuss a possible realization for the $cGP(2)$ adaptive time-stepping, but no parallel numerical investigation is done.

The new global-in-time adaptive strategy defines a new time grid in each adaptive iteration and computes the associated solution over the entire time interval. The classic step size controllers described in the literature are adapted to take into account error estimates for pressure, lift, and drag coefficients. Investigations and comparisons follow in the numerical studies for the different model problems, where advisable controllers are highlighted.

Acknowledgements

First of all, I would like to express my sincere gratitude to Prof. Dr. Stefan Turek, for giving me the opportunity to pursue my Ph.D. under his supervision, and I am truly grateful for his guidance. His scientific insights and ideas have been essential to this work and have always motivated me. I am deeply grateful for his valuable support both in my academic journey and in balancing personal and professional commitments, as well as for his trust and positive attitude that he has given me throughout these years.

I am grateful to apl. Prof. Dr. Friedhelm Schieweck from the University of Magdeburg for the opportunity to build on his work, and I appreciate his support and willingness to review my thesis.

I am so thankful to Xenia, not only for being my discussion partner but also for offering invaluable emotional support over the years. Thank you, for always making a place for me, both in your time and in your home.

I would like to express my sincere thanks to all members of the Institute of Applied Mathematics and Numerics, LS III, TU Dortmund, and I would like to highlight some individuals: Abida Begum, Dr. Arooj Fatima, Dr. Christoph Lohmann, Falko Ruppenthal, Jonas Dünnebacke, Mirco Arndt, Peter Zajac, and Dr. Shafqat Hussain. Your assistance in implementing and developing FEAT3, helping with other requests, and creating a welcoming atmosphere in the office has been great. I am especially grateful to some of you for your exceptional support in reviewing my thesis.

I am incredibly fortunate to have a family and friends who have supported me with their love and encouragement over the years. To my brother: Your brotherly care and support have meant the world to me. To my mother: Your love, devotion, and inspiration have shaped me. I thank you for everything. To Carsten: Thank you for sharing half of my life's journey with me and I am exceptional grateful for your love.

Dortmund, December 2024

Lydia Carmen Wambach

Contents

1 Introduction

1.1. Motivation and overview

Higher order variational time discretization

For solving time-dependent problems such as evolution equations, a higher order time discretization is appropriate for various reasons, such as achieving a sufficiently smooth solution. A well-known class of time discretization schemes is of the variational type, including in particular the higher order Galerkin methods, see for example the monograph [58]. A famous representative of this type is the continuous Galerkin-Petrov method. This approach has often been studied for various problems such as the heat equation [32, 34]. In addition, superconvergence in the endpoints of the time interval and optimal error estimates were proved [8]. Furthermore, different types of incompressible flow problems were studied [29–31, 33].

The continuous Galerkin-Petrov method, denoted $cGP(k)$-method, and the discontinuous Galerkin method, denoted $dG(k-1)$-method, use a discontinuous test space consisting of discontinuous polynomial functions of degree $k-1$ to construct a time-stepping scheme. In contrast to the $dG(k-1)$-method, where the ansatz space is discontinuous and consists of discontinuous polynomial functions of degree $k-1$, the $cGP(k)$-method uses a higher ansatz space consisting of continuous polynomial functions of degree k. The number of unknowns of the $cGP(k)$-method is the same as in the $dG(k-1)$-method. Since the $cGP(k)$-method uses a higher ansatz space, it is obvious that the accuracy of the $cGP(k)$-method is one order higher than that of the $dG(k-1)$-method. For all these reasons, we consider in this thesis the continuous Galerkin-Petrov method, in particular the $cGP(2)$-method, as well as the C^1-continuous Galerkin-Petrov method, abbreviated as cGP-$C1(3)$-method.

The C^1-continuous Galerkin-Petrov method, the cGP-$C1(k+1)$-method, for $k > 1$, was introduced by Matthies and Schieweck [46] for a generalized class of non-linear ordinary differential equations (ODEs). Here, the discrete solution and its first derivative are continuous, hence the name. This method can be achieved by performing a linear

post-processing step on the continuous discrete Galerkin-Petrov solution to increase the global smoothness of the discrete solution from C^0, where only the discrete solution is continuous, to C^1-continuity. Furthermore, the *cGP-C*$1(k+1)$-method can be understood as a *cGP*$(k+1)$-method with a reduced numerical time integration formula, which was shown by Hussain et al. [34] for the heat equation. The linear post-processing step has already been investigated for several problems such as the wave equations [7], the time-dependent convection-diffusion-reaction equations [3], the transient Stokes problems [1] and for evolutionary Navier-Stokes equations [4].

Since this approach can be interpreted as a technique using polynomials of one degree higher, it is traceable that the *cGP-C*$1(k+1)$-method is one order more accurate in the L^2-norm than the *cGP*(k)-method. Furthermore, both methods agree at the endpoint of the time intervals. Thus, the two methods are of equal order at the endpoints and both are A-stable. For example, we consider the *cGP*(k)- and the *cGP-C*$1(k+1)$-methods for $k=2$, which is the smallest k we can use to construct the *cGP-C*$1(k+1)$-method. The *cGP*(2)-method is of order three in the L^2-norm and superconvergent with order four at the discrete time points. The *cGP-C*$1(3)$-method is of order four at the discrete time points and in the L^2-norm. In addition, it was shown numerically that for the transient Stokes problem the post-processed solution is of order $2k$ at the discrete time points for velocity and pressure [1, 5]. Consequently, we can use the *cGP-C*$1(k+1)$-method and the *cGP*(k)-method to construct an error estimator in time for adaptive time step control.

Adaptive time step control

Adaptive time step control aims at efficiently solving time-dependent evolution problems with controlled accuracy. The main reason for adaptive time-stepping is to reduce the high computational cost and to solve problems more efficiently. This goes along with achieving computational stability and reliability. Especially in physical simulations that involve solving differential equations, the computational effort is high, because more accurate or more physically relevant solutions are needed. Certainly, the accuracy and thus the arising errors from a discretization method depend on the employed step sizes and the problem properties. If the dynamic of the problem is more oscillatory or not very stable, a smaller step size has to be used. Thus, by automatically adapting the time step size, it is possible to achieve a given error tolerance in terms of accuracy of the solution with a minimum number of time steps. Choosing an appropriate time step size to achieve a given accuracy is one of the two components of adaptive time step control. The second is the estimation of the error resulting from the step size used. Since the exact error is often unknown, so-called error estimators are used by comparing different solutions.

Therefore, an adaptive time step control first requires an automatic technique to define variable time step sizes. These algorithms are called controllers for short. They depend on the order of accuracy of the method and can determine the new time step size such that the estimated error is less than a given error tolerance. Thus, controllers usually involve a history of estimated errors and time step sizes. An overview of different types of controllers was given by Söderlind [57]. In particular, the elementary local error controller *(EL)*, is shown, and further developed controllers such as the predictive controller (*PC11*) and the proportional-integral-derivative controller (*PID*) are also studied. They are based on a feedback control perspective and have been applied to adaptively control the step

sizes of partial differential equations (PDEs) [6, 36], especially for the incompressible Navier-Stokes equations.

Obviously, adaptive approaches also find application in spatial discretizations, especially in the context of finite element methods (FEM). There are several space-time FEMs where the mesh size and the time step size are adapted simultaneously [53]. Very popular in recent years have also been the goal-oriented adaptive schemes with time and/or space variation. They are based on a variational formulation in space and time, see, e.g., [10].

Common approaches for estimating an error are to compare the numerically solutions for two different time step sizes or two different time-stepping methods of different or same order. They are based on a posteriori error estimation, which can be found in the textbook literature, see, e.g., [13]. This is often done for first or second order methods.

Another approach is to use embedded schemes that use a post-processed solution that is one order higher than the solution obtained by the original method to estimate the error of the lower order original method. This post-processing step has the advantage of low computational cost and is therefore very efficient, but has to be of higher order. However, this does not seem to be common in CFD, but this application area has various recommendations of adaptive time step control. Embedded schemes have been studied for ODEs [39] and applied to incompressible Navier-Stokes equations [36]. In an adaptive approach, the higher order of the methods can be fully exploited. Higher order variational embedded approaches are the $cGP(k)$ and $cGP\text{-}C1(k+1)$-methods described above. Thus, an error estimator for the $cGP(k)$-method can be obtained using the post-processed $cGP\text{-}C1(k+1)$-method. This has already been studied by Ahmed and John [2] for the $cGP(k)$- and $dG(k-1)$-methods. They showed that the $cGP(2)$-method is the best method in terms of efficiency and accuracy compared to $cGP(3)$, $dG(1)$ and $dG(2)$, as well as an adaptive Crank-Nicolson scheme [35]. As we will see in this work, an error for the pressure in incompressible flow problems cannot be estimated like the velocity error. Thus, we use an approach similar to the work of Hussain et al. [29] to obtain a new higher order pressure solution.

Certainly, the $cGP(2)$-method, or in general higher order variational time discretization schemes, have a higher computational cost, since the resulting systems are larger and more complex to solve. Therefore, we study an adaptive time step control designed for a parallel-in-time use for incompressible flow problems. In classical sequential adaptive time-stepping, one time step is processed after the other. Thus, this procedure is obviously not suitable for a global-in-time approach. Accordingly, we will construct an adaptive time step control based on adaptive iterations, where each adaptive iteration has to solve a global-in-time problem. Thus, in contrast to classical sequential adaptive approaches, we construct a completely new time grid in each adaptive iteration.

Global-in-time

In the field HPC, attention is focused on the growing number of cores. To make efficient use of these cores, the programs to be executed, and thus the numerical methods, must be applied in parallel. This leads to the redevelopment and modification of sequential numerical algorithms for solving differential equations. The problem of parallelization of ODEs and time-dependent PDEs and their numerical solution arises from the time dependence.

Starting from an initial time, the solution at the next time is computed. Therefore, the solution at the previous time point must be given as a requirement for the calculation of the solution at the next time point. Accordingly, initial value problems are approximated sequentially in time by numerical methods, while parallelization in space is already widely used. The decomposition of the space generates independent subproblems at each time step, which are suitable for parallelization. However, if we have a small number of spatial grid points, the parallelism in each time step is obviously limited. Furthermore, if the subproblems per processor are too small, the computation time will be dominated by the communication time between the processors rather than by the actual computational cost. If, in addition, we have a large number of time steps due to a long time interval or a requirement of small time steps, as in an adaptive time step control, where the solution must achieve a certain accuracy, a parallel or simultaneous in time approach is needed.

The problem of solving multiple time steps at once was recognized early on, so that many approaches can be found in the literature. Often they can be identified under the keyword *parallel-in-time*, since there is often no distinction between *parallel-in-time* and *time-simultaneous*. One can divide these methods into different types, which is shown in the work of Gander [22], where a detailed overview of already existing parallel-in-time algorithms is given.

A very famous iterative method is the *parareal algorithm* of Lions et al. [41], where the time interval is decomposed, while the *domain decomposition methods in space and time* describe another group of parallel-in-time algorithms. They decompose the problem into individual spatial and temporal subproblems that can be computed in parallel. This is followed by an iterative coupling of the approximations. A special modification is the *Waveform Relaxation* algorithm, which was originally developed for the simulation of integrated circuits. The *Multigrid Waveform Relaxation*, first published by Lubich and Ostermann [44], combines this approach with the class of parallel-in-time or rather time-simultaneous multigrid methods, *Multigrid Methods in Space-Time*. These algorithms are not naturally parallel, but the execution of the individual components can be performed in parallel at space-time intervals, since they perform simultaneously on the entire space-time domain. Here, a global discrete system is solved with multigrid methods. There are a lot of further developments, However, Hackbusch [26] was the first to publish a *multigrid only in time* method in 1984. Among the best known extensions of *Multigrid in Space-Time* are the PFASST algorithm [48] and the MGRIT algorithm [21].

Another promising *global-in-time multigrid* approach for parabolic equations was presented by Gander and Neumüller [23], which is based on a higher order discontinuous Galerkin time discretization and a finite element discretization in space, where a block Jacobi smoother is the key component. Recently, a similar *space-time multigrid* method for higher order continuous and discontinuous variational time discretizations with spatial finite element discretizations of the heat and wave equations was introduced by Margenberg and Munch [45]. It differs from [23] in that it uses a different smoother and focuses more on practical aspects of the space-time multigrid. Dünnebacke et al. published another *parallel-in-space* and *simultaneous-in-time* multigrid method for parabolic evolution equations [18]. Although this method was developed independently, it can be seen as a special case of the *multigrid waveform relaxation* method. In addition, Dünnebacke et al. extend this method to non-linear PDEs [19]. Furthermore, Lohmann and Turek recently presented a *global-in-time pressure Schur complement solver* for incompressible

flows [43] and a further development with an augmented Lagrangian acceleration [42] based on [17].

1.2. Outline

In Chapter 2 we start with an introduction to the evolutionary model problems, such as the heat equation and the non-stationary incompressible Navier-Stokes equations. We also consider fundamentals such as function spaces and the Gauss-Lobatto quadrature formula. Since the focus of this work is on the time discretization, we briefly consider at this point the spatial discretization by finite elements.

Next, the higher order time discretization is introduced in the Chapter 3. First, we present the *cGP*(k)- and *cGP-C*1$(k+1)$-methods in general for non-linear ODEs and delve into the post-processing step. In particular, we consider the resulting *cGP*(2) block system and its post-processing procedure. This is followed by extensions to the heat equation and the Navier-Stokes equations. For non-linear problems, a linearization technique such as fixed-point iteration is described.

In Chapter 4 we introduce adaptive time step control. Firstly, we study the error estimators, in particular the *cGP-C*1-velocity error estimator and a new pressure error estimator. Then, we design our new *global adaptive time step control*, where in each adaptive iteration a new time grid is constructed to obtain an adaptive global-in-time setting. On this basis, we study special control strategies such as interpolating time step sizes and error estimators to have a corresponding interpolated value for each time point in the new time grid. These are used in dedicated controllers to determine a new time step size in a new time grid. Possible restrictions in our control strategies are also discussed.

Chapter 5 provides more information about a global-in-time approach and how our higher order adaptive time-stepping technique fits into it. Therefore, we briefly discuss two global-in-time approaches, one developed for parabolic problems [19] and the other for incompressible flow problems [42].

Finally, our approaches are examined in the Chapter 6, where our numerical results for the heat equation and the Navier-Stokes equations are presented. To study incompressible flows, we consider the flow around a cylinder benchmark and the Carreau-Yasuda model.

This thesis is summarized with a conclusion and further outlooks in Chapter 7.

2

Notations and fundamentals

In this chapter, we introduce the model problems and the function spaces to discretize them in space and time. Since this work mainly focuses on the time discretization, we briefly discuss the space discretization. In this context, we apply the finite element method in space. As another fundamental concept, we consider the Gauss-Lobatto quadrature formula.

2.1. Evolution equations

Evolution equations describe in general the development of a system depending on a continuous time variable t for example given by an ODE of the general form

$$\partial_t u = f(u),$$

where $\partial_t u$ denotes the time derivative of the solution $u(t)$ and f is a given vector field in the state of the system at time t. Primarily a Hilbert space setting is applied in order to study differential equations such that a general unified approach can solve ODEs and PDEs. In this work, we will study the continuous Galerkin-Petrov method for ODEs and further extend it to PDEs like the heat equation and the Navier-Stokes equations. To transform a PDE into an ODE context, one possibility is to semi-discretize the equations in space by means of the finite element method. Thus, a large system of ODEs is created for the time-dependent nodal vector of the finite element solution, where afterwards a time discretization is performed. We refer to this discretization strategy in the next chapter, where the non-linear ODE for a semi-discretization with the finite element method in space will be defined, which was presented in [46]. Such discretization strategies can be found in the literature, see, e.g., [20, 28, 54]. At this point, we introduce the evolution equations, which we will explicitly study in this work such as the heat equation and the non-stationary incompressible Navier-Stokes equations.

2.1.1. The heat equation

A fundamental equation of mathematical physics, especially in thermodynamics, is the heat equation, which describes the heat transfer in a body over a period of time. For a physical domain $\Omega \subset \mathbb{R}^2$, we consider the solution $u : \Omega \times [0,T] \to \mathbb{R}$, which describes the temperature in the point $x \in \Omega$ at time $t \in I = [0,T]$ with a positive final time T. We seek the solution $u(x,t)$ such that

$$\begin{aligned} \partial_t u - \Delta u &= f && \text{in } \Omega \times (0,T), \\ u &= 0 && \text{on } \partial\Omega \times I, \\ u(x,0) &= u_0(x) && \text{for } x \in \Omega, \end{aligned} \tag{2.1.1}$$

where $f : \Omega \times (0,T) \to \mathbb{R}$ is a given source term. Furthermore, the initial temperature field at time $t = 0$ is $u_0 : \Omega \to \mathbb{R}$. For the sake of simplicity, we assume homogeneous Dirichlet boundary conditions at the boundary $\partial\Omega$ of a polygonal domain Ω.

2.1.2. The non-stationary incompressible Navier-Stokes equations

In order to model incompressible flow problems, we consider the non-stationary incompressible Navier-Stokes equations. For a domain $\Omega \subset \mathbb{R}^2$, the unknown velocity field $\mathbf{u}(t) : \Omega \to \mathbb{R}^2$ and the pressure field $p(t) : \Omega \to \mathbb{R}$ solve

$$\begin{aligned} \partial_t \mathbf{u} - \nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p &= \mathbf{f} && \text{in } \Omega \times (0,T), \\ \nabla\cdot\mathbf{u} &= 0 && \text{in } \Omega \times [0,T], \\ \mathbf{u} &= g && \text{on } \delta\Omega \times [0,T], \\ \mathbf{u}(x,0) &= \mathbf{u}_0(x) && \text{in } \Omega \text{ for } t = 0, \end{aligned} \tag{2.1.2}$$

where ν denotes the viscosity, f the body force and $\mathbf{u}_0$ the initial field at time $t = 0$. We assume homogeneous Dirichlet boundary conditions at the boundary $\partial\Omega$ of a polygonal domain Ω for simplicity, that is $g = 0$. In this case, the pressure field $p(t)$ at each time t in the subspace $L^2_0(\Omega) \subset L^2(\Omega)$ of functions with zero integral mean value.

2.1.3. The flow around a cylinder benchmark

Since the global-in-time adaptive $cGP(k)$-time step control, introduced in this work, is also studied for incompressible flow problems, it is indispensable not to consider the DFG flow around a cylinder benchmark [52] as a representative of real flow phenomena. Thus, we consider the Navier-Stokes equations (2.1.2) with source term $f = 0$, a fixed time interval with final time $T = 8$ and a viscosity parameter $\nu = 10^{-3}$. The initial condition at time $t = 0$ is defined as $\mathbf{u}_0 = \mathbf{0}$. The domain is illustrated in the Figure 2.1, which is a pipe without a circular cylinder $\Omega = [0,2.2] \times [0,041] \backslash B_r(0.2,0.2)$ with $r = 0.05$. The inflow data on the left edge $\Gamma_{in} = 0 \times [0,0.41]$ is described by the parabolic boundary profile

$$\mathbf{u}(x,y;t) = U(t)\frac{4y(0.41-y)}{0.41^2}\begin{pmatrix}1\\0\end{pmatrix}, \;\; U(t) = U_0 \sin(\pi t/8), \;\; U_0 = 1.5, \tag{2.1.3}$$

with a maximum velocity magnitude $U(t)$ at time $t = 4$, such that $U(4) = U_0 = 1.5$. On the right edge $\Gamma_{out} = 2.2 \times [0,0.41]$ the outflow is defined by do-nothing boundary conditions

$$\nu\delta_n u - pn = 0,$$

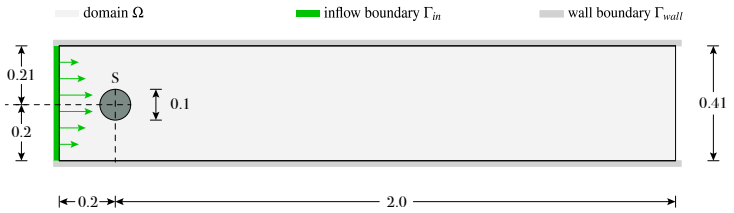


Figure 2.1: Domain of the flow around a cylinder benchmark.

where n indicates the outer normal vector. The boundary of the cylinder $S = \delta B_r(0.2, 0.2)$ as well as the lower and upper walls Γ_{wall} are defined as no-slip boundary conditions

$$\mathbf{u}_{|\Gamma_{wall}} = \mathbf{u}_{|S} = 0.$$

Furthermore, the maximum velocity magnitude leads to a mean velocity of

$$U_{mean} = \frac{2}{3} \cdot 1.5 = 1. \tag{2.1.4}$$

The characteristic length of the flow is given by the diameter of the circular obstacle $L = 2 \cdot 0.05 = 1$. This leads to a Reynolds number of $Re = \frac{U_{mean} L}{\nu} = 100$. This number indicates whether the flow is laminar. The most characteristic quantities of interest are the drag and lift coefficients at the obstacle. These forces are defined by

$$c_l(t) := -\frac{2}{U_{mean}^2 L} \int_S (\nu \frac{\partial u_{t_S}(t)}{\partial \mathbf{n}} n_x + p(t) n_y) dS,$$
$$c_d(t) := \frac{2}{U_{mean}^2 L} \int_S (\nu \frac{\partial u_{t_S}(t)}{\partial \mathbf{n}} n_y + p(t) n_x) dS,$$

where $\mathbf{n} = (n_x, n_y)^T$ is the unit normal vector on S pointing into Ω. Moreover, $u_{t_S} := \mathbf{u} \cdot \mathbf{t}_S$ indicates the tangential velocity along the circle with $\mathbf{t}_S = (n_y, -n_x)^T$ is the unit tangential vector. Another parameter of interest is the difference of the pressure between the front and the rear of the cylinder

$$\delta p(t) := p(t; 0.15, 0.2) - p(t; 0.25, 0.2).$$

Furthermore, we consider a shear-thinning Carreau fluid flow over the geometry of the flow around a cylinder benchmark, Figure 2.1, where the constant viscosity parameter ν is replaced by a shear-depended one.

2.1.4. The Carreau-Yasuda model

These fluids often appear in the industry such as thermoplastic fluids or biological fluids and are commonly modeled by the Carreau-Yasuda model [59, 60]. Thus, here a non-Newtonian viscosity parameter is applied, which is based on the deformation tensor such that the first equation of (2.1.2) is replaced by

$$\partial_t \mathbf{u} - 2\nabla \cdot (\nu(\dot{\gamma}) D(\mathbf{u})) + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = f \qquad \text{in } \Omega \times (0, T], \tag{2.1.5}$$

with source term $f = 0$, for a final time $T = 8$ and with the effective shear rate of the strain rate tensor

$$D(\mathbf{u}) := \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^\top).$$

Furthermore, the non-Newtonian viscosity parameter is given by

$$\nu(\dot{\gamma}) := \nu_\infty + (\nu_0 - \nu_\infty)(1 + (\lambda\dot{\gamma})^a)^{(n-1)/a},$$

where $\dot{\gamma} = \sqrt{2}||D(\mathbf{u})||_F$ and $||.||_F$ denotes the Frobenius norm. The other coefficients are non-negative material coefficients. As in the global-in-time approach of Lohmann and Turek [42, 43], we choose $a = 2$, which describes the Carreau model [15] and we set $n = 0.31$, $\nu_\infty = 0$, $\nu_0 = 10^{-2}$, $\lambda = 1$. For this problem, the maximum velocity in (2.1.3) is $U_0 = 0.3$ in the parabolic boundary inflow.

2.2. Function spaces

In order to discretize the above evolution equations in time and space, we need some definitions. The notation in the following two sections are based on the definitions provided in [31, 46].

Function spaces for the heat equation

Let $V = H_0^1(\Omega)$ be a Hilbert space and $C(I,V)$ the space of the continuous functions $u : I \to V$ equipped with the associated norm

$$\|u\|_{C(I,V)} := \sup_{t \in I} \|u(t)\|_V.$$

We consider a decomposition of the time interval I into N subintervals I_n, which are defined as

$$I_n := (t_{n-1}, t_n], \quad n = 1, \ldots, N \quad \text{with} \quad 0 := t_0 < t_1 < \ldots < t_{N-1} < t_N =: T, \tag{2.2.1}$$

and

$$\tau := \max_{1 \le n \le N} \tau_n, \tag{2.2.2}$$

with the local time step size

$$\tau_n := t_n - t_{n-1}. \tag{2.2.3}$$

We seek the approximation of the solution with respect to time $u_\tau : I \to V$ in the continuous ansatz space, which consists of piecewise polynomials of order k by

$$X_\tau^k := \{u \in C(I,V) : u|_{I_n} \in \mathbb{P}_k(I_n, V) \quad \forall\, n = 1, \ldots, N\} \tag{2.2.4}$$

with

$$\mathbb{P}_k(I_n, V) := \left\{ u : I_n \to V : u(t) = \sum_{j=0}^{k} U^j t^j,\ \forall\, t \in I_n,\ U^j \in V,\ \forall\, j \right\}.$$

Furthermore, the C^1-continuous time-discrete space is defined by

$$X_\tau^{k,C1} := \{u \in C^1(I,V) : u|_{I_n} \in \mathbb{P}_k(I_n, V) \quad \forall\, n = 1, \ldots, N\}. \tag{2.2.5}$$

We define the space of discontinuous functions with the associated norm by

$$L^2(I,V) := \left\{ u : I \to V : \|u\|_{L^2(I,V)} < \infty \right\}, \qquad \|u\|_{L^2(I,V)} := \left(\int_I \|u(t)\|_V^2 dt \right)^{\frac{1}{2}}.$$

This is needed in order to construct the discontinuous test space Y_τ^{k-1}, which consists of piecewise polynomials of order $k-1$

$$Y_\tau^{k-1} := \{ v \in L^2(I,V) : v|_{I_n} \in \mathbb{P}_{k-1}(I_n,V) \quad \forall\, n = 1,\dots,N \}. \tag{2.2.6}$$

The functions in this test space are permitted to be discontinuous at the end points of the subintervals t_n, $n = 1,\dots,N-1$. Therefore, we define for an approximation of the solution with respect to time $u_\tau : I \to V$ the left-sided value u_n^-, the right-sided value u_n^+, and the jump $[u_\tau]_n$ by

$$u_n^- := \lim_{t \to t_n - 0} u_\tau(t), \qquad u_n^+ := \lim_{t \to t_n + 0} u_\tau(t), \qquad [u_\tau]_n := u_n^+ - u_n^-.$$

If we refer in this work to the value $u_\tau(t_n)$ for $n \geq 1$, it is defined by the left-sided value $u_\tau(t_n) := u_n^-$.

Inner product and bilinear forms for the heat equation

The inner product in $L^2(\Omega)$ is denoted by $(\cdot,\cdot)_\Omega$ and $a(\cdot,\cdot)$ indicates the following bilinear form:

$$(u,v)_\Omega := \int_\Omega u(x)v(x)dx, \qquad a(u,v) := (\nabla u, \nabla v)_\Omega \qquad \forall\, u,v \in V$$

Function spaces for the Navier-Stokes equations

When considering the Navier-Stokes equations, the solution now describes a velocity field $\mathbf{u}(t) := (u(t), v(t))$, which is an element of

$$\mathbf{V} := V^2 := (H_0^1(\Omega))^2. \tag{2.2.7}$$

To define the required spaces, we seek now the time discrete solution $\mathbf{u}_\tau(t) : I \to \mathbf{V}$ such that the above spaces are defined by $\mathbf{X}_\tau^k$ (2.2.4), $\mathbf{X}_\tau^{k,C1}$ (2.2.5), and $\mathbf{Y}_\tau^{k-1}$ (2.2.6) accordingly. Additionally, we look for the pressure $p(t)$ in the scalar valued space

$$Q := L_0^2(\Omega) \tag{2.2.8}$$

and define the analogous time-continuous pressure ansatz space $\tilde{X}_\tau^k$ and time-discontinuous pressure test space $\tilde{Y}_\tau^{k-1}$.

Inner product and bilinear forms for the Navier-Stokes equations

Now, $(\cdot,\cdot)_\Omega$ denotes the usual inner product in $(L^2(\Omega))^2$, the bilinear forms $a(\cdot,\cdot)$ and $b(\cdot,\cdot)$ on $\mathbf{V} \times \mathbf{V}$ and $\mathbf{V} \times Q$ are defined by

$$a(\mathbf{u},\mathbf{v}) := \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v}\, dx \quad \mathbf{u},\mathbf{v} \in \mathbf{V}, \qquad b(\mathbf{v},p) := -\int_\Omega \nabla \cdot \mathbf{v}\, p\, dx, \tag{2.2.9}$$

and the trilinear form $c(\cdot,\cdot,\cdot)$ on $\mathbf{V}\times\mathbf{V}\times\mathbf{V}$ is given by $c(\mathbf{w},\mathbf{u},\mathbf{v}) := \sum_{i=1}^{2} c_s(\mathbf{w},u_i,v_i)$ with $\mathbf{u}=(u_1,u_2)$ and $\mathbf{v}=(v_1,v_2)$, where

$$c_s(\mathbf{w},u_i,v_i) := \int_\Omega (\mathbf{w}\cdot\nabla u_i)v_i\,dx \quad \forall \mathbf{w}\in\mathbf{V},\ \ u_i,v_i\in V. \tag{2.2.10}$$

2.3. Space discretization

For solving parabolic evolution problems, we have to discretize them in space and time. As the title of this thesis suggests, the focus is on the time discretization by the continuous Galerkin-Petrov method, which is applied to a space semi-discretization using finite elements. For more detailed information on the finite element method, see,[11, 16, 56].

In our numerical studies, we use biquadratic finite elements for the solution of the heat equation and the velocity field of the Navier-Stokes equations while the pressure field is approximated by discontinuous linear finite elements on a quadrilateral mesh Ω. For the Navier-Stokes equations, the so-called Q_2/P_1^{disc} Stokes element is a popular choice [24], since this pair is a well-known combination and fulfills the Ladyzhenskaya-Babuška-Brezzi (LBB) condition, see, [12, 25, 47]. The corresponding finite element spaces are denoted by $\mathbf{V}_h\subset\mathbf{V}$ and $Q_h\subset Q$, which were defined in (2.2.7) and (2.2.8). The resulting error in the L^2-norm is of order $O(h^3)$ for the velocity and of order $O(h^2)$ for the pressure. It is sufficient to consider directly the space discretization for the Navier-Stokes equations, because the biquadratic finite elements are also applied to the heat equation.

By taking the finite element subspaces $\mathbf{V}_h\subset\mathbf{V}$ and $Q_h\subset Q$ into account, we get the finite element semi-discretization of the Navier-Stokes equations

$$\begin{aligned}(\mathbf{u}_{h,t},\mathbf{v}_h)_\Omega + a(\mathbf{u}_h,\mathbf{v}_h) + c_s(\mathbf{u}_h,\mathbf{u}_h,\mathbf{v}_h) + b(\mathbf{v}_h,p_h) &= (f_h,\mathbf{v}_h)_\Omega,\\ b(\mathbf{u}_h,q_h) &= 0,\\ \mathbf{u}(0) &= \mathbf{u}_0 \qquad\qquad \forall(\mathbf{v}_h,q_h)\in\mathbf{V}_h\times Q_h.\end{aligned}$$

Now, we represent $(\mathbf{u}_h(t),p_h(t))$ by a finite element function. Let $\phi_\mu\in V_h, \mu=1,\dots,m_h$ indicate the scalar finite element basis functions of V_h. Then the semi-discrete finite element solution $\mathbf{u}_h(t)=(u_h(t),v_h(t))\in\mathbf{V}_h$ is represented by

$$u_h(t) := \sum_{\mu=1}^{m_h}(u_\mu(t))\phi_\mu \qquad \text{and} \qquad v_h(t) := \sum_{\mu=1}^{m_h}(v_\mu(t))\phi_\mu, \tag{2.3.1}$$

with the nodal vectors $(u_\mu(t))\in\mathbb{R}^{m_h}$ and $(v_\mu(t))\in\mathbb{R}^{m_h}$. The same can be stated with respect to the pressure. The finite element basis functions of Q_h are indicated by $\psi_\mu\in Q_h$, $\mu=1,\dots,n_h$. Then, we denote the nodal vector of the semi-discrete finite element solution $p_h(t)\in Q_h$ by $(p_\mu(t))\in\mathbb{R}^{n_h}$ and obtain

$$p_h(t) := \sum_{\mu=1}^{n_h}(p_\mu(t))\psi_\mu.$$

Next, we define some well known matrices, which result from applying the finite element method in space and use the above defined bilinear forms (2.2.9) and (2.2.10). The mass

matrix $M \in \mathbb{R}^{m_h \times m_h}$, the discrete Laplacian matrix $L \in \mathbb{R}^{m_h \times m_h}$ and the gradient matrices $B_i \in \mathbb{R}^{m_h \times n_h}$, $i = 1,2$, are given by

$$M_{\nu,\mu} := (\phi_\mu, \phi_\nu)_\Omega, \quad L_{\nu,\mu} := a(\phi_\mu, \phi_\nu), \quad (B_i)_{\nu,\mu} := b(\phi_\nu \mathbf{e}^i, \psi_\mu). \tag{2.3.2}$$

Furthermore, we introduce for each time step the two right-hand-side vectors $\underline{F}, \underline{G} \in \mathbb{R}^{m_h}$ with the components

$$(\underline{F})_\nu := (f(t), \phi_\nu \mathbf{e}^1)_\Omega, \quad (\underline{G})_\nu := (f(t), \phi_\nu \mathbf{e}^2)_\Omega. \tag{2.3.3}$$

Finally, we define the matrix $N(\underline{\mathbf{w}})_{\nu,\mu} \in \mathbb{R}^{m_h \times m_h}$ for a given discrete velocity field $\mathbf{w}_h \in \mathbf{V}_h$ with the nodal vector $\underline{\mathbf{w}} \in \mathbb{R}^{2m_h}$ as

$$N(\underline{\mathbf{w}})_{\nu,\mu} := c_s(\mathbf{w}_h, \phi_\mu, \phi_\nu). \tag{2.3.4}$$

2.4. Gauss-Lobatto quadrature formula

In order to numerically approximate the occurring integrals, we will apply the Gauss-Lobatto quadrature formula for instance in the continuous Galerkin-Petrov method. We consider the reference interval $\hat{I} = [-1,1]$ with $k+1$ points $x_0, ..., x_k$, where $x_0 = -1$ and $x_k = 1$ are the start and end points of the interval and $x_1, \dots x_{k-1}$ are the zero points of the first derivative of the Legendre polynomial P_k. The weights are given by

$$\omega_i := \frac{2}{k(k+1)P_k^2(x_i)}, \qquad i = 0, \dots, k. \tag{2.4.1}$$

Hence, the quadrature formula results in

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=0}^{k} \omega_i f(x_i). \tag{2.4.2}$$

Polynomials up to degree $2k-1$ are integrated exactly. In this work we mainly use the 3-point and 5-point Gauss-Lobatto formulas, which are given by the nodal points and weights in the following Table 2.1.

$k+1$	nodal points x_i	weights ω_i
3	0	$\frac{4}{3}$
	± 1	$\frac{1}{3}$
5	0	$\frac{32}{45}$
	$\pm\sqrt{\frac{3}{7}}$	$\frac{49}{90}$
	± 1	$\frac{1}{10}$

Table 2.1: 3-point and 5-point Gauss-Lobatto quadrature formula.

Furthermore, we show the use of the quadrature formula concretely on the time subinterval I_n (2.2.1) for a to integrated function $g(t)$. By applying an affine transformation

$T_n : \hat{I} \to I_n$ to transform all I_n integrals into integrals over the reference interval, we define

$$t = T_n(\hat{t}) := \frac{t_{n-1}+t_n}{2} + \frac{t_n - t_{n-1}}{2}\hat{t} \quad \in I_n \qquad \forall\, \hat{t} \in \hat{I},\ n = 1,\dots,N, \tag{2.4.3}$$

$$\hat{t} := T_n^{-1}(t) = \frac{2}{t_n - t_{n-1}}\left(t - \frac{t_n + t_{n-1}}{2}\right) \quad \in \hat{I}. \tag{2.4.4}$$

Inserting this in

$$\int_{I_n} g(t)dt = \int_{t_{n-1}}^{t_n} g(t)\ dt = \frac{t_n - t_{n-1}}{2} \int_{-1}^{1} g(T_n(\hat{t}))d\hat{t}. \tag{2.4.5}$$

Using the quadrature formula (2.4.2) with the reference points $\hat{t}$ and the weights $\hat{\omega}$ (2.4.1), we obtain

$$\frac{t_n - t_{n-1}}{2} \int_{-1}^{1} g(T_n(\hat{t}))d\hat{t} \approx \frac{\tau_n}{2} \sum_{i=0}^{k} \hat{w}_i\ g(T_n(\hat{t}_i)).$$

Now, we define the i-th reference point $\hat{t}_i$ on the subinterval I_n, by

$$t_{n,i}^{GL(k+1)} := T_n(\hat{t}_i). \tag{2.4.6}$$

In addition, we can define the quadrature points $\hat{t}_i$ on the subinterval I_n also for the $(k+1)$-point Gauss-Legendre formula, which is denoted by $t_{n,i}^{G(k+1)}$.

3

Higher order time discretization

In this chapter, we consider two continuous Galerkin-Petrov methods, known as the $cGP(k)$-method and the $cGP\text{-}C1(k+1)$-method for a non-linear ODE. Next, we discuss the post-processing step separately, since we do not need to compute the $cGP\text{-}C1(k+1)$-method to obtain the discrete C^1-continuous solution in time. Afterwards, we consider extensions necessary for the application of the continuous Galerkin-Petrov methods to the heat equation and the incompressible Navier-Stokes equations. In particular, we present the algebraic block system of the $cGP(2)$-method and the associated post-processing step to obtain the $cGP\text{-}C1(3)$ solution for the different problem types. Finally, we consider a linearization technique based on the fixed-point iteration to handle the non-linearity.

3.1. Continuous Galerkin-Petrov methods

In order to apply the continuous Galerkin-Petrov time step method to different evolution equations, we generalize the continuous Galerkin-Petrov method by considering the approach outlined in [46]. This serves as the main reference for this chapter, and we adopt the notation introduced there. The following non-linear ODE as a model problem, where $u : [0,T] \to V$ solves

$$\begin{aligned} M d_t u(t) &= F(t,u(t)) \qquad \forall\, t \in (0,T), \\ u(0) &= u_0, \end{aligned} \tag{3.1.1}$$

where $u(t) = (u_j(t)) \in \mathbb{R}^{m_h}$ denotes the nodal vector of the semi-discrete finite element solution $u_h(t) \in V_h$ to a parabolic partial differential equation such as (2.3.1). M is the time-independent mass matrix resulting from the space discretization. Additionally, the function $F : [0,T] \times V \to V$ is assumed to be sufficiently smooth and can be non-linear. The initial value at time $t = 0$ is denoted by $u_0 \in V$.

We choose the following explanation especially for this type of problem, because we can transform a partial differential equation by semi-discretization in space into a system of ordinary differential equations. Furthermore the extension of the algebraic block system is straightforward for the Navier-Stokes equations considered below.

3.1.1. $cGP(k)$-method

To introduce the continuous Galerkin-Petrov method in a general way, we discretize the non-linear ODE (3.1.1) in time as in [46]. To obtain a time marching process, we decompose the time interval I into N subintervals I_n as in (2.2.1). The time discretization parameter is given by τ_n (2.2.2) with the local time step size (2.2.3). The solution $u : I \to V$ is approximated by a time discrete function $u_\tau : I \to V$, which is a piecewise polynomial of some order k from the time discrete solution space X_τ^k (2.2.4). The discrete test space is defined by Y_τ^{k-1} (2.2.6), which consists of piecewise polynomials of order $k-1$ that are globally discontinuous at the endpoints of the time intervals.

To get a time discrete problem, we use a variational approach and multiply (3.1.1) with the test function $v_\tau \in Y_\tau^{k-1}$. Furthermore, we integrate over I and obtain the time-discrete global problem, where we seek $u_\tau \in X_\tau^k$ such that

$$\int_I \langle M d_t u_\tau(t), v_\tau(t)\rangle \, dt = \int_I \langle F(t, u_\tau(t)), v_\tau(t)\rangle \, dt \qquad \forall\, v_\tau \in Y_\tau^{k-1}, \tag{3.1.2}$$

where $u_\tau(0) = u_0$. Also we will denote problem (3.1.2) by the "exact $cGP(k)$-method", because the integral in time on the right-hand-side is evaluated exactly.

To solve the problem (3.1.2) as a time marching process, the discontinuous test space is exploited, which leads to local subproblems that can be solved successively. We choose test functions $v_\tau(t) \in Y_\tau^{k-1}$ of the form $v_\tau(t) = v\psi(t)$ with an arbitrary time-independent $v \in V$ and a scalar function $\psi : I \to \mathbb{R}$, which is zero at $I \setminus I_n$. By applying the test functions to I_n in (3.1.2), we obtain the "I_n-problem of the exact $cGP(k)$-method" such that $u_\tau(0) = u_0$.

$$\int_{I_n} \langle M d_t u_\tau(t), v\rangle \psi(t) \, dt = \int_{I_n} \langle F(t, u_\tau(t)), v\rangle \psi(t) \, dt \qquad \forall\, v \in V,\ \forall \psi \in \mathbb{P}_{k-1}(I_n, V). \tag{3.1.3}$$

To numerically approximate the integrals, we choose the $(k+1)$-point Gauss-Lobatto formula as in Section 2.4 with the reference points $t_{n,j}^{GL(k+1)}$ with $j = 0, \ldots, k$ (2.4.6) on the subinterval I_n (2.4.5) and with their weights $\hat{\omega}_j$ (2.4.1). This was also proposed in [34, 51]. Let $t_{n,j} = t_{n,j}^{GL(k+1)}$, $j = 0, \ldots, k$ be the $(k+1)$-points of the $(k+1)$-point Gauss-Lobatto formula used throughout this chapter. This leads to $t_{n,0} = t_{n-1}$ and $t_{n,k} = t_n$. Especially, this quadrature formula enables a payoff in the post-processing and thus in the cGP-$C1(k+1)$-method. Since it is exact for polynomials of degree less than or equal to $2k-1$, the left-side is integrated exactly. Then, we numerically integrate (3.1.3) by the $(k+1)$-point Gauss-Lobatto quadrature formula and get the "I_n-problem of the numerically integrated $cGP(k)$-method", where we want to find $u_\tau|_{I_n}$, such that

$$\begin{aligned} u_\tau(t_{n,0}) &= u_\tau(t_{n-1}) = u_{n-1}, \\ \sum_{j=0}^{k} \hat{\omega}_j M d_t u_\tau(t_{n,j}) \psi(t_{n,j}) &= \sum_{j=0}^{k} \hat{\omega}_j F(t_{n,j}, u_\tau(t_{n,j})) \psi(t_{n,j}) \ \ \forall\, \psi \in \mathbb{P}_{k-1}(I_n, V), \end{aligned} \tag{3.1.4}$$

with the initial condition at each subinterval u_{n-1}. We will need this problem representation later. First, we identify $u_\tau(t_{n,j})$ by continuing with the Lagrange basis functions with

respect to the $k+1$ nodes $t_{n,j} \in I_n$ to define $u_\tau(t)$ at I_n

$$\phi_{n,j}(t) := \prod_{\substack{l=0 \\ l \neq j}}^{k} \frac{t - t_l}{t_j - t_l} \quad \in \mathbb{P}_k(I_n),\ j = 0, \ldots, k,$$

which satisfy

$$\phi_{n,j}(t_{n,i}) = \delta_{i,j}, \quad i, j = 0, \ldots, k,$$

the Kronecker symbol $\delta_{i,j}$. We define $u_\tau(t)|_{I_n}$ by a polynomial ansatz

$$u_\tau(t) := \sum_{j=0}^{k} U_n^j \phi_{n,j}(t) \qquad \forall\, t \in I_n, \tag{3.1.5}$$

with coefficients $U_n^j \in \mathbb{R}^{m_h}$. We can write $U_n^j = u_\tau|_{I_n}(t_{n,j})\ \forall j$.

Using the Gauss-Lobatto points, we obtain $t_{n,0} = t_{n-1}$ and the initial condition is given by

$$U_n^0 = u_\tau|_{I_{n-1}}(t_{n-1}) \ \text{ if } n \geq 2 \qquad \text{or} \qquad U_n^0 = u_0 \ \text{ if } n = 1.$$

Since the Gauss-Lobatto quadrature formula in Section 2.4 is defined on a reference interval $\hat{I}$, we transform all I_n integrals into integrals over the reference interval (2.4.5). The basis functions of the ansatz space $\phi_{n,j}(t)$ are defined on the reference interval $\hat{I}$ with respect to the Gauss-Lobatto points $\hat{t}_i \in \hat{I}$ with $\hat{\phi}_j \in \mathbb{P}_k(\hat{I})$ by

$$\begin{aligned} \hat{\phi}_j(\hat{t}_i) &= \delta_{i,j}, && i, j = 0, \ldots, k, \\ \phi_{n,j}(t) &:= \hat{\phi}_j(\hat{t}) = \hat{\phi}_j(T_n^{-1}(t)), \end{aligned}$$

with the affine transformation $T_n^{-1}(t)$ (2.4.4). Since the discrete test space Y_τ^{k-1} consists of piecewise polynomials of order $k-1$, the functions $\psi_{n,i}$ are also piecewise polynomials of order less than or equal to $k-1$ on I_n. We define the basis functions of the test space by

$$\psi_{n,i}(t) := \hat{\psi}_i(T_n^{-1}(t)) = \hat{\psi}_i(\hat{t}) \quad \in \mathbb{P}_{k-1}(\hat{I}) \quad \forall\, t \in I_n,\ i = 1, \ldots, k.$$

We transform the derivative $\partial_t \phi_{n,j}(t)$ on $\hat{I}_n$ by the chain rule, which leads to

$$\partial_t \phi_{n,j}(t) = \partial_{\hat{t}} \hat{\phi}_j(\hat{t}) \cdot \partial_t \underbrace{\left(\frac{2}{\tau_n} \left(t - \frac{t_n + t_{n-1}}{2} \right) \right)}_{= T_n^{-1}(t)} = \partial_{\hat{t}} \hat{\phi}_j(\hat{t}) \frac{2}{\tau_n}.$$

Next, we apply the representation (3.1.5) to (3.1.3), transforming all integrals to the reference interval $\hat{I}$

$$\begin{aligned} \int_{I_n} \sum_{j=0}^{k} \langle M U_n^j, v \rangle \phi_j'(t) \psi(t)\, dt \quad &= \int_{I_n} \sum_{j=0}^{k} \langle F(t, U_n^j), v \rangle \phi_j(t) \psi(t)\, dt \\ \Leftrightarrow \int_{\hat{I}} \sum_{j=0}^{k} \langle M U_n^j, v \rangle \frac{2}{\tau_n} \hat{\phi}_j'(\hat{t}) \hat{\psi}(\hat{t})\, d\hat{t} &= \int_{\hat{I}} \sum_{j=0}^{k} \langle F(T_n(\hat{t}), U_n^j), v \rangle \hat{\phi}_j(\hat{t}) \hat{\psi}(\hat{t})\, d\hat{t} \end{aligned}$$

for each test basis function $\psi \in \mathbb{P}_{k-1}(I_n)$ and all $v \in V$. Then, we approximate the integrals numerically using the $(k+1)$ Gauss-Lobatto formula and obtain:

$$\sum_{\mu=0}^{k}\sum_{j=0}^{k}\langle MU_n^j, v\rangle \frac{2}{\tau_n}\hat{\phi}_j'(\hat{t}_\mu)\hat{\psi}(\hat{t}_\mu)\hat{\omega}_\mu = \sum_{\mu=0}^{k}\sum_{j=0}^{k}\langle F(T_n(\hat{t}_\mu), U_n^j), v\rangle \hat{\phi}_j(\hat{t}_\mu)\hat{\psi}(\hat{t}_\mu)\hat{\omega}_\mu. \tag{3.1.6}$$

Here, the test functions $\hat{\psi}_i \in \mathbb{P}_{k-1}(\hat{I})$ are chosen as in [34, 46] as

$$\hat{\psi}_i(\hat{t}_\mu) = (\hat{\omega}_\mu)^{-1}\delta_{i,\mu} \qquad \forall\, i,\mu \in \{1,\dots,k\}. \tag{3.1.7}$$

Inserting (3.1.7) in (3.1.6), we achieve

$$\sum_{j=0}^{k} MU_n^j \underbrace{\sum_{\mu=0}^{k}\hat{\phi}_j'(\hat{t}_\mu)\hat{\psi}_i(\hat{t}_\mu)\hat{\omega}_\mu}_{=:\alpha_{i,j}} = \frac{\tau_n}{2}\sum_{j=0}^{k} F(T_n(\hat{t}_j), U_n^j) \underbrace{\sum_{\mu=0}^{k}\hat{\phi}_j(\hat{t}_\mu)\hat{\psi}_i(\hat{t}_\mu)\hat{\omega}_\mu}_{=:\beta_{i,j}} \quad \forall i = 1,\dots,k, \tag{3.1.8}$$

with coefficients $\alpha_{i,j}$ and $\beta_{i,j}$. This results in the "special form of the I_n-problem of the numerically integrated $cGP(k)$-method" as the following block system.

For a given initial value $U_n^0 \in \mathbb{R}^{m_h}$, compute the "coefficients" $U_n^1,\dots,U_n^k \in \mathbb{R}^{m_h}$ by solving the coupled system:

$$\sum_{j=0}^{k}\alpha_{i,j}\, MU_n^j = \frac{\tau_n}{2}\sum_{j=0}^{k}\beta_{i,j}\, F(t_{n,j}, U_n^j) \quad \forall i = 1,\dots,k,$$

where the initial value is defined by $U_n^0 = u_\tau(t_{n-1})$ for $n > 1$ and $U_1^0 = u_0$.

$cGP(1)$-method

Here, we use for $k = 1$, the 2-point Gauss-Lobatto formula, which coincides with the trapezoidal rule, with the points $t_{n,0} = t_{n-1}$ and $t_{n,1} = t_n$ using the weights $\hat{\omega}_0 = \hat{\omega}_1 = 1$. Then, with the choice of the test function (3.1.7), we obtain that $\hat{\psi}_1(\hat{t}) \equiv 1$

$$\alpha_{1,0} = -1,\ \ \alpha_{1,1} = 1,\ \ \beta_{1,0} = \beta_{1,1} = 1$$

and the I_n-problem reads:
For the given initial value $U_n^0 \in \mathbb{R}^{m_h}$, determine the "coefficient" $U_n^1 \in \mathbb{R}^{m_h}$ by solving the following block system on I_n

$$MU_n^1 = \frac{\tau_n}{2}\left\{F(t_n, U_n^1) + F(t_{n-1}, U_n^0)\right\} + MU_n^0.$$

$cGP(2)$-method

Here, we set $k = 2$ and use the 3-point Gauss-Lobatto formula with the points $t_{n,0} = t_{n-1}$, $t_{n,1} = (t_{n-1} + t_n)/2$ and $t_{n,2} = t_n$ with the weights $\hat{\omega}_0 = \hat{\omega}_2 = \frac{1}{3}$ and $\hat{\omega}_1 = \frac{4}{3}$. Then, with our choice of the test function (3.1.7), we obtain

$$\hat{\psi}_1(\hat{t}) = \frac{3}{4}(1-\hat{t}), \qquad \hat{\psi}_2(\hat{t}) = 3\hat{t}$$

and

$$(\alpha)_{i,j} = \begin{pmatrix} -\frac{5}{4} & 1 & \frac{1}{4} \\ 2 & -4 & 2 \end{pmatrix}, \qquad (\beta)_{i,j} = \begin{pmatrix} \frac{1}{2} & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}.$$

The I_n-problem reads:
For the given initial value $U_n^0 \in \mathbb{R}^{m_h}$, determine the two "coefficients" $U_n^1, U_n^2 \in \mathbb{R}^{m_h}$ by solving the following block system on I_n

$$\begin{aligned} MU_n^1 + \frac{1}{4}MU_n^2 &= \frac{5}{4}MU_n^0 + \frac{\tau_n}{2}\left\{ F(t_{n,1}, U_n^1) + \frac{1}{2}F(t_{n,0}, U_n^0) \right\}, \\ -4MU_n^1 + 2MU_n^2 &= -2MU_n^0 + \frac{\tau_n}{2}\left\{ F(t_{n,2}, U_n^2) - F(t_{n,0}, U_n^0) \right\}. \end{aligned} \tag{3.1.9}$$

3.1.2. *cGP-C*1$(k+1)$-method

The *cGP-C*1$(k+1)$-method is derived from the *cGP*(k)-method, where we take advantage of the C^1-continuous time-discrete solution u_τ. Matthies and Schieweck were the first to show this in their work [46] for a generalized class of non-linear ODEs. Thus, the level of the global smoothness of the discrete solution is increased from C^0- to C^1-continuity. Hence, the *cGP-C*1$(k+1)$-method can also be interpreted as the *cGP*$(k+1)$-method with a reduced numerical time integration, which was proved by Matthies and Schieweck [46]. Hence, the *cGP-C*1$(k+1)$-method is one order more accurate in the time interval than the *cGP*(k)-method, but without increasing the number of unknowns and thus the computational complexity. In particular, Hussain et al. showed that the *cGP-C*1(3)-method for the heat equation can be understood as a method with a reduced numerical time integration, if the 4-point Gauss-Lobatto formula is replaced by the 3-point Gauss-Lobatto formula [34].

We let u_τ be a polynomial of degree $k+1$ for $k \geq 2$. Therefore, we need $k+2$ conditions on each subinterval. Consider the first interval $I_1 = (0, t_1)$ as an example. First, we know the initial value $u_\tau(0) = u_0$ of the differential equation and second, the time derivative of u_τ at $t = 0$:

$$M d_t u_\tau(t_0) = F(t_0, u_0).$$

Here, we only have to solve a system with the mass matrix to obtain the time derivative of the solution $d_t u_\tau(t_0)$. Another quasi-explicit condition, which couples $u_\tau(t_1)$ to $d_t u_\tau(t_1)$, is given by

$$M d_t u_\tau(t_1) = F(t_1, u_1).$$

Accordingly, $k-1$ unknowns remain implicitly coupled. The "global *cGP-C*1$(k+1)$-method" for $k \geq 2$, where we seek $u_\tau \in X_\tau^{k+1,C1}$, is given by

$$\int_I \langle M d_t u_\tau(t), v_\tau(t) \rangle \, dt = \int_I \langle F(t, u_\tau(t)), v_\tau(t) \rangle \, dt \qquad \forall\, v_\tau \in Y_\tau^{k-2}, \tag{3.1.10}$$

with $u_\tau(0) = u_0$. Analogous to the *cGP*(k)-method, we can solve (3.1.10) by means of a time-marching process due to the discontinuity of the test function v_τ and write the

local time discrete problem as the "I_n-problem of the exactly integrated $cGP\text{-}C1(k+1)$-method", where we seek $u_\tau|_{I_n} \in \mathbb{P}_{k+1}$:

$$\begin{aligned} u_\tau(t_{n-1}) &= u_{n-1}, \\ Md_t u_\tau(t_{n-1}) &= F(t_{n-1}, u_\tau(t_{n-1})), \\ Md_t u_\tau(t_n) &= F(t_n, u_\tau(t_n)), \\ \int_{I_n} Md_t u_\tau(t)\psi(t)\,dt &= \int_{I_n} F(t, u_\tau(t))\psi(t)\,dt \quad \forall\, \psi \in \mathbb{P}_{k-2}(I_n). \end{aligned} \tag{3.1.11}$$

By applying the $k+1$-point Gauss-Lobatto formula, we obtain the "I_n-problem of the numerically integrated $cGP\text{-}C1(k+1)$-method", where we seek $u_\tau|_{I_n} \in \mathbb{P}_{k+1}$ such that

$$u_\tau(t_{n-1}) = u_{n-1}, \tag{3.1.12}$$

$$Md_t u_\tau(t_{n-1}) = F(t_{n-1}, u_\tau(t_{n-1})), \tag{3.1.13}$$

$$Md_t u_\tau(t_n) = F(t_n, u_\tau(t_n)), \tag{3.1.14}$$

$$\sum_{j=0}^{k} \hat{\omega}_j Md_t u_\tau(t_{n,j})\psi(t_{n,j}) = \sum_{j=0}^{k} \hat{\omega}_j F(t_{n,j}, u_\tau(t_{n,j}))\psi(t_{n,j}) \quad \forall\, \psi \in \mathbb{P}_{k-2}. \tag{3.1.15}$$

Because of the condition $u_\tau|_{I_{n+1}}(t_n) = u_\tau|_{I_n}(t_n) =: u_\tau(t_n)$, we obtain from (3.1.11) that the derivative of the left-sided and right-sided values u_n^- and u_n^+ coincide such that

$$d_t^- u_\tau(t_n) = d_t u_\tau|_{I_n}(t_n) = F(t_n, u_\tau(t_n)) = d_t u_\tau|_{I_{n+1}}(t_n) = d_t^+ u_\tau(t_n),$$

and the solution u_τ is C^1-continuous.

We do not need to build the system of the $cGP\text{-}C1(k+1)$-method as described above. There is a connection between the $cGP\text{-}C1(k+1)$-method and the $cGP(k)$-method, where we can compute the $cGP\text{-}C1(k+1)$ solution from the $cGP(k)$ solution by a simple post-processing step.

3.2. Post-processing step

In this section, we confirm that we can solve the discrete $cGP\text{-}C1(k+1)$-problem by performing a post-processing step based on the $cGP(k)$ solution as it was proved in [46]. We then consider this associated theorem and the proof provided in [46]. Furthermore, we derive a post-processing step that requires only the solution of a linear system with the mass matrix. This problem is still linear even for highly non-linear equations such as the Navier-Stokes equations. Therefore, the total computational cost is almost the same as considering only the $cGP(k)$-method.

Theorem 3.1. *Let u_τ and $\tilde{u}_\tau$ represent the solutions of the numerically integrated variants of the discrete $cGP(k)$- and $cGP\text{-}C1(k+1)$-methods respectively. Assume that the time step sizes τ_n are sufficiently small such that the I_n-problems of both methods have a unique solution for all $n = 1, \ldots, N$. Then, for all time intervals $I_n = (t_{n-1}, t_n]$, we find*

$$\tilde{u}_\tau(t) = u_\tau(t) + a_n \zeta_n(t) \qquad \forall t \in I_n, \tag{3.2.1}$$

where the vector $a_n \in \mathbb{R}^{m_h}$ can be computed by

$$a_n := M^{-1}\{F(t_n, u_\tau(t_n)) - Md_t u_\tau(t_n)\} \tag{3.2.2}$$

and

$$\zeta_n(t) := \frac{\tau_n}{2}\hat{\zeta}(\hat{t}) \quad with \quad \hat{t} := T_n^{-1}(t),$$

where $T_n : (-1,1] \to I_n$ denotes the affine reference mapping and $\hat{\zeta} \in P_{k+1}(\hat{I})$ is the polynomial defined by the conditions $\hat{\zeta}'(1) = 1$ and $\hat{\zeta}(\hat{t}_j) = 0$ for all $j = 0,\dots,k$, where $\hat{t}_j$ denotes the the integration points of the $(k+1)$-point Gauss-Lobatto formula [46].

Proof. We define $z(t) := u_\tau(t) + a_n\zeta_n(t)$ and show that this $z(t)$ satisfies all conditions, which are given by (3.1.12) -(3.1.15) of the solution to the discrete *cGP-C*$1(k+1)$ problem. Since this solution is unique, the proof is complete. The polynomial ζ is constructed with the $t_{n,j}, j = 0,\dots,k$ quadrature points of the $(k+1)$-Gauss-Lobatto formula with $\zeta_n(t_{n,j}) = 0$, such that

$$z(t_{n,j}) = u_\tau(t_{n,j}), \quad j = 0,\dots,k.$$

Then, for $u_\tau(t_{n-1}) = u_{n-1}$ and $t_{n,0} = t_{n-1}$ we have

$$z(t_{n-1}) = z(t_{n,0}) = u_\tau(t_{n,0}) = u_\tau(t_{n-1}),$$

which shows the first condition (3.1.12).

To prove the fourth condition (3.1.15), we need to show that the following holds:

$$\sum_{j=0}^{k} \hat{\omega}_j M d_t z(t_{n,j})\psi(t_{n,j}) = \sum_{j=0}^{k} \hat{\omega}_j F(t_{n,j}, z(t_{n,j}))\psi(t_{n,j})$$

Therefore, we start with the left side of the above equation and insert the definition of $z(t)$

$$\begin{aligned}
&\sum_{j=0}^{k} \hat{\omega}_j M d_t z(t_{n,j})\psi(t_{n,j}) \\
&= \sum_{j=0}^{k} \hat{\omega}_j M d_t u_\tau(t_{n,j})\psi(t_{n,j}) + \sum_{j=0}^{k} \hat{\omega}_j M \underbrace{d_t a_n \zeta_n(t_{n,j})}_{a_n\zeta_n'(t_{n,j})}\psi(t_{n,j}) \\
&= \sum_{j=0}^{k} \hat{\omega}_j F(t_{n,j}, u_\tau(t_{n,j}))\psi(t_{n,j}) + \sum_{j=0}^{k} \hat{\omega}_j M a_n \zeta_n'(t_{n,j})\psi(t_{n,j}) \\
&= \sum_{j=0}^{k} \hat{\omega}_j F(t_{n,j}, u_\tau(t_{n,j}))\psi(t_{n,j}) + M a_n \frac{2}{\tau_n}\int_{I_n} \zeta_n'(t)\psi(t)\,dt
\end{aligned}$$

and have used (3.1.4) such that u_τ solves the "I_n-problem of the numerically integrated *cGP*(k)-method". Then, due to the fact, that the $(k+1)$-point Gauss-Lobatto formula is exact for polynomials of degree less than or equal to $2k-1$, we can use the exact integral form. Furthermore, we integrate by parts and show that this term vanishes by

using $\zeta(t_{n,j}) = 0$ and then again using the $(k+1)$-point Gauss-Lobatto formula with $t_{n,j}$ as the quadrature points:

$$\begin{aligned}\int_{I_n} \zeta_n'(t)\psi(t)\,dt &= -\int_{I_n} \zeta_n(t)\psi'(t)\,dt + \underbrace{\zeta_n(t_n)}_{=0}\psi(t_n) - \underbrace{\zeta_n(t_{n-1})}_{=0}\psi(t_{n-1}) \\ &= -\frac{\tau_n}{2}\sum_{j=0}^{k}\hat{\omega}_j \underbrace{\zeta_n(t_{n,j})}_{=0}\psi'(t_{n,j}) = 0.\end{aligned}$$

To sum up and insert $z(t_{n,j}) = u_\tau(t_{n,j})$, we obtain

$$\sum_{j=0}^{k}\hat{\omega}_j M d_t z(t_{n,j})\psi(t_{n,j}) = \sum_{j=0}^{k}\hat{\omega}_j F(t_{n,j},u_\tau(t_{n,j}))\psi(t_{n,j}) \quad = \sum_{j=0}^{k}\hat{\omega}_j F(t_{n,j},z(t_{n,j}))\psi(t_{n,j}),$$

which shows that z fulfills the fourth condition (3.1.15).

The third condition (3.1.14) is given by inserting the definition of $z(t)$ using $\zeta_n'(t_n) = 1$ and the formula for computing a_n (3.2.2)

$$\begin{aligned}M d_t z(t_n) &= M d_t u_\tau(t_n) + M d_t a_n \zeta_n(t_n) = M d_t u_\tau(t_n) + M a_n \\ &= M d_t u_\tau(t_n) + F(t_n,u_\tau(t_n)) - M d_t u_\tau(t_n) = F(t_n,u_\tau(t_n)) = F(t_n,z(t_n)).\end{aligned}$$

Now, we have to show the second condition (3.1.13), such that the following is satisfied:

$$M d_t z(t_{n-1}) = F(t_{n-1},z(t_{n-1})).$$

Therefore, we aim to prove that z satisfies the differential equation in t_{n-1} by choosing the polynomial $\sigma_n \in \mathbb{P}_{k-1}$ as a test function in the discrete $cGP(k)$-method, which vanishes in all inner Gauss-Lobatto points $t_{n,j} = 0, j = 1,\dots,k-1$, and performing $\sigma_n(t_n) = 1$. This is an admissible test function in the I_n-problem of the $cGP(k)$-method. We obtain by inserting σ_n as a test function in (3.1.4):

$$\begin{aligned}&\sum_{j=0}^{k}\hat{\omega}_j M d_t u_\tau(t_{n,j})\sigma(t_{n,j}) = \sum_{j=0}^{k}\hat{\omega}_j F(t_{n,j},u_\tau(t_{n,j}))\sigma(t_{n,j}) \\ \Leftrightarrow\ &\hat{\omega}_0 M d_t u_\tau(t_{n,0})\sigma_n(t_{n,0}) + \hat{\omega}_k M d_t u_\tau(t_{n,k})\sigma_n(t_{n,k}) \\ &= \hat{\omega}_0 F(t_{n,0},u_\tau(t_{n,0}))\sigma(t_{n,0}) + \hat{\omega}_k F(t_{n,k},u_\tau(t_{n,k}))\sigma(t_{n,k}).\end{aligned}$$

Next, we exploit $\hat{\omega}_0 = \hat{\omega}_k$ and further $\sigma_n(t_n) = 1$:

$$\begin{aligned}&M d_t u_\tau(t_{n,0})\sigma_n(t_{n,0}) + M d_t u_\tau(t_{n,k}) = F(t_{n,0},u_\tau(t_{n,0}))\sigma(t_{n,0}) + F(t_{n,k},u_\tau(t_{n,k})) \\ \Leftrightarrow\ &\sigma_n(t_{n,0})\left\{M d_t u_\tau(t_{n,0}) - F(t_{n,0},u_\tau(t_{n,0}))\right\} = \underbrace{F(t_{n,k},u_\tau(t_{n,k})) - M d_t u_\tau(t_{n,k})}_{M a_n}.\end{aligned}$$

To derive the second equivalence formulation, we used that the weights of the Gauss-Lobatto formula satisfy $\hat{\omega}_0 = \hat{\omega}_k \neq 0$ and $\sigma_n(t_n) = 1$. Then, we can simplify the equation by dividing by $\hat{\omega}_0$. Afterwards we transform it to obtain the third equation.

Now, we can start with the definition of z and replace the term Ma_n with the above equation

$$\begin{aligned} Md_t z(t_{n-1}) &= Md_t u_\tau(t_{n-1}) + Ma_n \zeta_n'(t_{n-1}) \\ &= Md_t u_\tau(t_{n-1}) + \zeta_n'(t_{n-1})\sigma_n(t_{n-1}) \left\{ Md_t u_\tau(t_{n-1}) - F(t_{n-1}, u_\tau(t_{n-1})) \right\}, \end{aligned}$$

where $\zeta_n'(t_{n-1})\sigma_n(t_{n-1}) = -1$, which we show by integration by parts, $\sigma_n(t_{n-1}) = 0$, $\zeta_n(t_n) = 0$ and using the quadrature formula

$$\begin{aligned} 0 &= \int_{I_n} \zeta_n(t)\sigma_n'(t)\,dt = -\int_{I_n} \zeta_n'(t)\sigma_n(t)\,dt + \zeta_n(t_n)\sigma_n(t_n) - \zeta_n(t_{n-1})\sigma_n(t_{n-1}) \\ &= -\frac{\tau_n}{2}\hat{\omega}_0 \zeta_n'(t_{n-1})\sigma_n(t_{n-1}) - \frac{\tau_n}{2}\hat{\omega}_k = -\frac{\tau_n}{2}\hat{\omega}_0 \left(\zeta_n'(t_{n-1})\sigma_n(t_{n-1}) + 1 \right). \end{aligned}$$

We get

$$0 = \zeta_n'(t_{n-1})\sigma_n(t_{n-1}) + 1 \quad \Leftrightarrow \quad \zeta_n'(t_{n-1})\sigma_n(t_{n-1}) = -1$$

and by inserting this we obtain

$$Md_t z(t_{n-1}) = Md_t u_\tau(t_{n-1}) - Md_t u_\tau(t_{n-1}) + F(t_{n-1}, u_\tau(t_{n-1})) = F(t_{n-1}, z(t_{n-1})),$$

which shows the second condition (3.1.13) and we have shown that $z(t)$ satisfies all conditions (3.1.12) -(3.1.15) for the solution of the I_n-problem of the $cGP\text{-}C1(k+1)$-method. Since the solution is unique, it follows that $z = \tilde{u}_\tau$ and the proof is complete. □

3.3. Application to the heat equation

In this section, we briefly consider the application of the $cGP(k)$-method and its post-processing step to the heat equation (2.1.1), which can be done in a straightforward way. After performing the discretization in space as in Chapter 2.3, we obtain a large system of ODEs as the considered model problem (3.1.1). For the heat equation, this system has the form

$$\begin{aligned} Md_t u(t) &= \underbrace{\underline{F}(t) - Lu(t)}_{F(t,u(t))} \qquad \forall t \in (0,T), \\ u(0) &= u_0, \end{aligned}$$

where M and L denote the mass and the Laplacian matrix as defined in (2.3.2). Furthermore, the vector of the right-hand-side is denoted by $\underline{F}$ as in equation (2.3.3).

The discretization in time is now straightforward and we receive the "discrete I_n-problem of the $cGP(k)$-method" by the following coupled block system at each time interval I_n for a given $U_n^0 \in \mathbb{R}^{m_h}$, find $U_n^1, \ldots, U_n^k \in \mathbb{R}^{m_h}$ such that

$$\sum_{j=0}^{k} \alpha_{i,j} M U_n^j + \frac{\tau_n}{2}\beta_{i,j} L U_n^j = \sum_{j=0}^{k} \frac{\tau_n}{2}\beta_{i,j}\underline{F}_n^j, \qquad \forall\, i = 1, \ldots, k,$$

where $\alpha_{i,j}, \beta_{i,j}$ are defined in (3.1.8). The vector U_n^0 is defined as the initial condition u_0 if $n = 1$ and as the discrete solution of the previous time interval U_{n-1}^2 if $n \geq 2$.

3.3.1. $cGP(2)$-method

We use the 3-point Gauss-Lobatto formula with the points $t_{n,0} = t_{n-1}$, $t_{n,1} = (t_{n-1} + t_n)/2$ and $t_{n,2} = t_n$ while the weights are given by $\hat{\omega}_0 = \hat{\omega}_2 = \frac{1}{3}$ and $\hat{\omega}_1 = \frac{4}{3}$. We obtain

$$(\alpha_{i,j}) = \begin{pmatrix} -\frac{5}{4} & 1 & \frac{1}{4} \\ 2 & -4 & 2 \end{pmatrix}, \qquad (\beta_{i,j}) = \begin{pmatrix} \frac{1}{2} & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

and the I_n-problem reads:

For the given initial value $U_n^0 \in \mathbb{R}^{m_h}$, determine the two "coefficients" $U_n^1, U_n^2 \in \mathbb{R}^{m_h}$ by solving the following block system on I_n:

$$\begin{pmatrix} M + \frac{\tau_n}{2}L & \frac{1}{4}M \\ -4M & 2M + \frac{\tau_n}{2}L \end{pmatrix} \begin{pmatrix} U_n^1 \\ U_n^2 \end{pmatrix} = \begin{pmatrix} R_n^2 \\ R_n^2 \end{pmatrix}, \tag{3.3.1}$$

where

$$R_n^1 = \frac{5}{4}MU_n^0 - \frac{\tau_n}{4}LU_n^0 + \frac{\tau_n}{2}\left\{ \underline{F}_n^1 + \frac{1}{2}\underline{F}_n^0 \right\},$$

$$R_n^2 = -2MU_n^0 + \frac{\tau_n}{2}LU_n^0 + \frac{\tau_n}{2}\left\{ \underline{F}_n^2 - \underline{F}_n^0 \right\}.$$

3.3.2. cGP-$C1(3)$-method

Here we show the post-processing step that is performed to obtain the discrete solution of the cGP-$C1(3)$-method by using the initial vector U_n^0 and the solution U_n^1, U_n^2, which were computed by the $cGP(2)$-method at each time interval (3.3.1), as explained before. Therefore, in the equation for the post-processing step in (3.2.2), we replace $F(t,u(t))$ by the corresponding functions for the heat equation $\underline{F}(t) - Lu(t)$. Then, we have to calculate the vector $a_n \in \mathbb{R}^{m_h}$ at each time interval I_n by solving the following system with the mass matrix:

$$Ma_n = \underline{F}_n^2 - LU_n^2 - M\sum_{j=0}^{2} U_n^j \phi'_{n,j}(t_n), \tag{3.3.2}$$

where $U_n^0 = U_{n-1}^2$. Afterwards, we obtain the nodal vector of the discrete solution $\tilde{u}_\tau(t)$ of the cGP-$C1(3)$-method at some point in time $t \in I_n$ by the formula

$$\tilde{u}_\tau(t) = u_\tau(t) + a_n \zeta_n(t) \qquad \forall t \in I_n,$$

where $T_n : (-1,1] \to I_n$ denotes the affine reference mapping and $\hat{\zeta} \in P_3(\hat{I})$ is the polynomial defined by the conditions $\hat{\zeta}'(1) = 1$ and $\hat{\zeta}(\hat{t}_j) = 0$ for all $j = 0, \ldots, 2$ where $\hat{t}_j$ denotes the integration points of the 3-point Gauss-Lobatto formula [46].

3.4. Extension to the Navier-Stokes equations

In this section, we consider the $cGP(k)$-method and its post-processing step for the non-stationary incompressible Navier-Stokes equations (2.1.2). By applying a space discretization as in Section 2.3, we obtain a matrix-vector representation of the semi-discrete

problem with matrices (2.3.2), (2.3.4) and (2.3.3). This non-linear system of differential algebraic equations for the nodal vectors $(\mathbf{u}(t), p(t)) \in \mathbb{R}^{2m_h} \times \mathbb{R}^{n_h}$ of the semi-discrete finite element solution $(\mathbf{u}_h(t), p_h(t)) \in \mathbf{V}_h \times Q_h$ can be written as

$$\begin{aligned} \mathbf{M}d_t\mathbf{u}(t) + \mathbf{B}p(t) &= \underbrace{\underline{\mathbf{F}}(t) - \mathbf{L}\mathbf{u}(t) - \mathbf{N}(\mathbf{u}(t))\mathbf{u}(t)}_{\mathbf{F}(t,\mathbf{u}(t))} \qquad \forall t \in (0,T), \\ \mathbf{u}(0) &= \mathbf{u}_0, \\ \mathbf{B}^T\mathbf{u}(t) &= 0. \end{aligned} \tag{3.4.1}$$

Then, we apply the higher order time discretization as in the previous section. At this point we define the polynomial ansatz to represent the velocity $\mathbf{u}_\tau \in \mathbf{X}_\tau^k$ and the pressure $p_\tau \in \tilde{X}_\tau^k$, which are given by

$$\mathbf{u}_\tau(t) := \sum_{j=0}^{k} \mathbf{U}_n^j \phi_{n,j}(t), \qquad p_\tau(t) := \sum_{j=0}^{k} P_n^j \phi_{n,j}(t), \qquad t \in I_n. \tag{3.4.2}$$

For more details, the interested reader is referred to [29]. The discrete "I_n-problem of the $cGP(k)$-method" is given by the following coupled block system at each time interval I_n with $\mathbf{U}_n^0 \in \mathbb{R}^{2m_h}$, where we seek $\mathbf{U}_n^1, \ldots, \mathbf{U}_n^k \in \mathbb{R}^{2m_h}$ and $P_n^1, \ldots, P_n^k \in \mathbb{R}^{n_h}$, $j = 1, \ldots, k$ such that for all $i = 1, \ldots, k$, it satisfies

$$\begin{aligned} \sum_{j=0}^{k} \alpha_{i,j}\mathbf{M}\mathbf{U}_n^j + \frac{\tau_n}{2}\beta_{i,j}\left\{\mathbf{L}\mathbf{U}_n^j + \mathbf{N}(\mathbf{U}_n^j)\mathbf{U}_n^j + \mathbf{B}P_n^j\right\} &= \sum_{j=0}^{k} \frac{\tau_n}{2}\beta_{i,j}\underline{\mathbf{F}}_n^j, \\ \mathbf{B}^T\mathbf{U}_n^i &= 0, \end{aligned}$$

with

$$\mathbf{U}_n^0 := \mathbf{U}_{n-1}^2 \ \text{ if } n > 1 \quad \mathbf{U}_1^0 := \mathbf{u}_0, \tag{3.4.3}$$

with the coefficients α and β defined in (3.1.8) and the block matrices

$$\mathbf{M} = \begin{bmatrix} M & 0 \\ 0 & M \end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix} L & 0 \\ 0 & L \end{bmatrix}, \quad \mathbf{N}(\mathbf{w}) = \begin{bmatrix} N(\mathbf{w}) & 0 \\ 0 & N(\mathbf{w}) \end{bmatrix},$$
$$\mathbf{B} = \begin{bmatrix} B_u \\ B_v \end{bmatrix}, \qquad \underline{\mathbf{F}}_n^i = \begin{bmatrix} \underline{F}_n^i \\ \underline{G}_n^i \end{bmatrix}.$$

3.4.1. Post-processing step

The post-processing step for the Navier-Stokes equations is not as simple as for the heat equation, since we have to include the pressure variable. This idea has also been generalized in [1, 4]. Note that this is still linear for highly non-linear equations such as the Navier-Stokes equations.

Here, we consider the "I_n-problem of the numerically integrated $cGP\text{-}C1(k+1)$-

method", where we seek $\mathbf{u}_\tau|_{I_n}$ and $p_\tau|_{I_n}$ such that

$$\mathbf{u}_\tau(t_{n-1}) = \mathbf{u}_{n-1}, \tag{3.4.4}$$

$$\mathbf{M}d_t\mathbf{u}_\tau(t_{n-1}) + \mathbf{B}p_\tau(t_{n-1}) = \mathbf{F}(t_{n-1}, \mathbf{u}_\tau(t_{n-1})), \tag{3.4.5}$$

$$\mathbf{M}d_t\mathbf{u}_\tau(t_n) + \mathbf{B}p_\tau(t_n) = \mathbf{F}(t_n, \mathbf{u}_\tau(t_n)), \tag{3.4.6}$$

$$\sum_{j=0}^{k} \hat{\omega}_j \left\{\mathbf{M}d_t\mathbf{u}_\tau(t_{n,j}) + \mathbf{B}p_\tau(t_{n,j})\right\} \psi(t_{n,j}) = \sum_{j=0}^{k} \hat{\omega}_j \mathbf{F}(t_{n,j}, \mathbf{u}_\tau(t_{n,j}))\psi(t_{n,j}) \;\forall\, \psi \in \mathbb{P}_{k-2}(I_n), \tag{3.4.7}$$

$$\mathbf{B}^T\mathbf{u}_\tau(t_{n,i}) = 0 \qquad \text{for } i = 1, \ldots, k. \tag{3.4.8}$$

The following theorem was also employed in [4].

Theorem 3.2. *Let $(\boldsymbol{u}_\tau, p_\tau) \in \boldsymbol{X}_\tau^k \times \tilde{X}_\tau^k$ and $(\tilde{\boldsymbol{u}}_\tau, \tilde{p}_\tau) \in \boldsymbol{X}_\tau^{k+1,C1} \times \tilde{X}_\tau^k$ represent the solutions of the numerically integrated variants of the discrete cGP(k)- and cGP-C1($k+1$)-methods respectively. Assume that the time step sizes τ_n are sufficiently small so that the I_n-problems of both methods have a unique velocity solution for all $n = 1, \ldots, N$. Then, for all time intervals $I_n = (t_{n-1}, t_n]$, it is satisfied*

$$\tilde{\boldsymbol{u}}_\tau(t) = \boldsymbol{u}_\tau(t) + \boldsymbol{a}_n\zeta_n(t) \quad \textit{and} \quad \tilde{p}_\tau(t) = p_\tau(t) + b_n\zeta_n'(t) \qquad \forall t \in I_n. \tag{3.4.9}$$

The vectors $\boldsymbol{a}_n := (a_n^u, a_n^v) \in \mathbb{R}^{2m_h}$ and $b_n \in \mathbb{R}^{n_h}$ can be computed by solving the linear saddle point problem

$$\begin{aligned} \boldsymbol{M}\boldsymbol{a}_n + \boldsymbol{B}b_n &= \underbrace{\underline{\mathbf{F}}_n - \{\boldsymbol{L} + \boldsymbol{N}(\boldsymbol{u}_\tau(t_n))\}\boldsymbol{u}_\tau(t_n)}_{\boldsymbol{F}(t_n, \boldsymbol{u}_\tau(t_n))} - \boldsymbol{B}p_\tau(t_n) - \boldsymbol{M}\chi_n, \\ \boldsymbol{B}^T\boldsymbol{a}_n &= 0, \end{aligned}$$

with $\chi_n := (\chi_n^u, \chi_n^v) \in \mathbb{R}^{2m_h}$ denotes the nodal representation of the components of $\boldsymbol{u}_\tau'(t_n)$, which is computed by $\sum_{j=0}^{k} \boldsymbol{U}_n^j \phi_{n,j}'(t_n)$. The block matrices are defined in (2.3.2) *and* (2.3.4). *The polynomial $\zeta_n(t)$ is given by*

$$\zeta_n(t) := \frac{\tau_n}{2}\hat{\zeta}(t) \quad \textit{with} \quad \hat{t} := T_n^{-1}(t),$$

with $T_n : (-1, 1] \to I_n$ denotes the affine reference mapping and $\hat{\zeta} \in P_{k+1}(\hat{I})$ is the polynomial defined by the conditions $\hat{\zeta}'(1) = 1$ and $\hat{\zeta}(\hat{t}_j) = 0$ for all $j = 0, \ldots, k$, where $\hat{t}_j$ denotes the integration points of the $(k+1)$-point Gauss-Lobatto formula.

Proof. This proof is considered as an extension to the proof of [46], that we showed for Theorem 3.1. Similar to the differential algebraic (3.4.1), where we added the pressure on the left side of the equation and summarized the Laplacian term and the non-linear term on the right side, we are now add the pressure parts in this proof.

We show that the *cGP-C*1 velocity solution satisfies all conditions (3.4.4) -(3.4.8). This is sufficient to prove the theorem due to the uniqueness of the velocity solution. We define the *cGP-C*1 solutions by

$$\mathbf{z}(t) := \mathbf{u}_\tau(t) + \mathbf{a}_n\zeta_n(t), \qquad \tilde{z}(t) := p_\tau(t) + b_n\zeta_n'(t).$$

The first condition (3.4.4) coincides with (3.1.12), which we have already shown in the proof of Theorem 3.1. Accordingly, it is satisfied

$$\mathbf{z}(t_{n-1}) = \mathbf{z}(t_{n,0}) = \mathbf{u}_\tau(t_{n,0}) = \mathbf{u}_\tau(t_{n-1}).$$

To prove the fourth condition (3.4.7), we start with the left side of the above equation, insert the definition of $\mathbf{z}(t)$ and $\tilde{z}$, and use the fact that $(\mathbf{u}_\tau, p_\tau)$ solves the "I_n-problem of the numerically integrated $cGP(k)$-method"

$$\begin{aligned}
&\sum_{j=0}^{k} \hat{\omega}_j \mathbf{M} d_t \mathbf{z}(t_{n,j})\psi(t_{n,j}) + \hat{\omega}_j \mathbf{B}\tilde{z}(t_{n,j})\psi(t_{n,j}) \\
&= \sum_{j=0}^{k} \hat{\omega}_j \mathbf{F}(t_{n,j}, \mathbf{u}_\tau(t_{n,j}))\psi(t_{n,j}) + \sum_{j=0}^{k} \hat{\omega}_j \mathbf{M}\mathbf{a}_n \zeta_n'(t_{n,j})\psi(t_{n,j}) + \hat{\omega}_j \mathbf{B} b_n \zeta_n'(t_{n,j})\psi(t_{n,j}) \\
&= \sum_{j=0}^{k} \hat{\omega}_j \mathbf{F}(t_{n,j}, \mathbf{u}_\tau(t_{n,j}))\psi(t_{n,j}) + \{\mathbf{M}\mathbf{a}_n + \mathbf{B}b_n\} \underbrace{\frac{2}{\tau} \int_{I_n} \zeta_n'(t)\psi(t)\,dt}_{=0} \\
&= \sum_{j=0}^{k} \hat{\omega}_j \mathbf{F}(t_{n,j}, \mathbf{z}(t_{n,j}))\psi(t_{n,j}).
\end{aligned}$$

Since the $(k+1)$-point Gauss-Lobatto formula is exact for polynomials of degree less than or equal to $2k-1$, we can replace the quadrature formula by the exact integrals. Additionally, we see by integration by parts that this term vanishes by using $\zeta(t_{n,j}) = 0$ and afterwards integrating again with the $(k+1)$-point Gauss-Lobatto formula with the quadrature points $t_{n,j}$. We insert again the definition of $\mathbf{z}$, which shows that $\mathbf{z}(t)$ satisfies the fourth condition (3.4.7).

The third condition (3.4.6) is given by inserting the definition of $\mathbf{z}(t)$ and $\tilde{z}$ using $\zeta_n'(t_n) = 1$ and the formula for computing $\mathbf{a}_n, b_n$:

$$\begin{aligned}
\mathbf{M} d_t \mathbf{z}(t_n) + \mathbf{B}\tilde{z}(t_n) &= \mathbf{M} d_t \mathbf{u}_\tau(t_n) + \mathbf{B} p_\tau(t_n) + (\mathbf{M}\mathbf{a}_n + \mathbf{B}b_n)\zeta_n'(t_n) \\
&= \mathbf{M} d_t \mathbf{u}_\tau(t_n) + \mathbf{B} p_\tau(t_n) + \mathbf{F}(t_n, \mathbf{u}_\tau(t_n)) - \mathbf{M} d_t \mathbf{u}_\tau(t_n) - \mathbf{B} p_\tau(t_n) \\
&= \mathbf{F}(t_n, \mathbf{u}_\tau(t_n)) = \mathbf{F}(t_n, \mathbf{z}(t_n)).
\end{aligned}$$

For the second condition (3.4.5), we have to show that $\mathbf{z}$ and $\tilde{z}$ satisfy the differential equation in t_{n-1} by choosing the polynomial $\sigma_n \in \mathbb{P}_{k-1}$ as the test function in the discrete $cGP(k)$-method, which vanishes in all inner Gauss-Lobatto points $t_{n,j} = 0, j = 1, \ldots, k-1$, and performs $\sigma_n(t_n) = 1$. This is an admissible test function in the I_n-problem of the $cGP(k)$-method. By inserting σ_n as the test function and transforming it as in the proof

for the non-linear ODE, we obtain

$$\begin{aligned}
&\sum_{j=0}^{k} \hat{\omega}_j \mathbf{M} d_t \mathbf{u}_\tau(t_{n,j})\sigma(t_{n,j}) + \hat{\omega}_j \mathbf{B} p_\tau(t_{n,j})\sigma(t_{n,j}) = \sum_{j=0}^{k} \hat{\omega}_j \mathbf{F}(t_{n,j}, \mathbf{u}_\tau(t_{n,j}))\sigma(t_{n,j}) \\
\Leftrightarrow \quad & \left\{\mathbf{M} d_t \mathbf{u}_\tau(t_{n,0}) + \mathbf{B} p_\tau(t_{n,0})\right\} \sigma_n(t_{n,0}) + \left\{\mathbf{M} d_t \mathbf{u}_\tau(t_{n,k}) + \mathbf{B} p_\tau(t_{n,k})\right\} \\
& = \mathbf{F}(t_{n,0}, \mathbf{u}_\tau(t_{n,0}))\sigma(t_{n,0}) + \mathbf{F}(t_{n,k}, \mathbf{u}_\tau(t_{n,k})) \\
\Leftrightarrow \quad & \sigma_n(t_{n,0}) \left\{\mathbf{M} d_t \mathbf{u}_\tau(t_{n,0}) + \mathbf{B} p_\tau(t_{n,0}) - \mathbf{F}(t_{n,0}, \mathbf{u}_\tau(t_{n,0}))\right\} \\
& = \underbrace{\mathbf{F}(t_{n,k}, \mathbf{u}_\tau(t_{n,k})) - \mathbf{M} d_t \mathbf{u}_\tau(t_{n,k}) - \mathbf{B} p_\tau(t_{n,k})}_{\mathbf{M}\mathbf{a}_n + \mathbf{B} b_n}.
\end{aligned}$$

Now, we can start with the definitions of $\mathbf{z}, \tilde{z}$ and replace the term $\mathbf{M}\mathbf{a}_n + \mathbf{B}b_n$ by the above equation and further $\zeta_n'(t_{n-1})\sigma_n(t_{n-1}) = -1$ which we showed in the proof of Theorem 3.1 using integration by parts. Substituting this, we obtain

$$\begin{aligned}
& \mathbf{M} d_t \mathbf{z}(t_{n-1}) + \mathbf{B}\tilde{z}(t_{n-1}) \\
&= \mathbf{M} d_t u_\tau(t_{n-1}) + \mathbf{M}\mathbf{a}_n \zeta_n'(t_{n-1}) + \mathbf{B} p_\tau(t_{n-1}) + \mathbf{B} b_n \zeta_n'(t_{n-1}) \\
&= \mathbf{M} d_t u_\tau(t_{n-1}) + \mathbf{B} p_\tau(t_{n-1}) \\
&\quad + \underbrace{\zeta_n'(t_{n-1})\sigma_n(t_{n-1})}_{=-1} \left\{\mathbf{M} d_t \mathbf{u}_\tau(t_{n-1}) + \mathbf{B} p_\tau(t_{n-1}) - \mathbf{F}(t_{n,0}, \mathbf{u}_\tau(t_{n,0}))\right\} \\
&= \mathbf{F}(t_{n-1}, \mathbf{u}_\tau(t_{n-1})) = \mathbf{F}(t_{n-1}, \mathbf{z}(t_{n-1})),
\end{aligned}$$

which shows the second condition (3.4.5). The last condition (3.4.8) guarantees that the discrete velocity solution is discretely divergence-free and satisfies

$$\mathbf{B}^T \mathbf{z}(t_{n,i}) = \mathbf{B}^T \mathbf{u}_\tau(t_{n,i}) = 0, \qquad \text{for } i = 1, \ldots, k,$$

due to the equivalence of the $cGP(k)$ and $cGP\text{-}C1(k+1)$ velocity solutions. We showed that $(\mathbf{z}, \tilde{z})$ satisfies all conditions of (3.4.4) -(3.4.8) for the solution of the I_n-problem of the $cGP\text{-}C1(k+1)$-method. Since the velocity solution is unique, it follows that $\mathbf{z} = \tilde{\mathbf{u}}_\tau$ and $\tilde{z} = \tilde{p}_\tau$ and the proof is complete. □

3.4.2. $cGP(2)$-method

We use the 3-point Gauss-Lobatto formula with the points $t_{n,0} = t_{n-1}$, $t_{n,1} = (t_{n-1} + t_n)/2$, $t_{n,2} = t_n$ and weights $\hat{\omega}_0 = \hat{\omega}_2 = \frac{1}{3}$, $\hat{\omega}_1 = \frac{4}{3}$. We obtain

$$(\alpha_{i,j}) = \begin{pmatrix} -\frac{5}{4} & 1 & \frac{1}{4} \\ 2 & -4 & 2 \end{pmatrix}, \qquad (\beta_{i,j}) = \begin{pmatrix} \frac{1}{2} & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}.$$

The initial velocity solution for the next time interval is given by $\mathbf{U}_{n+1}^0 := \mathbf{U}_n^2$ due to equation (3.4.2) and since the velocity solution is continuous. Furthermore, the initial pressure solution is needed, because P_n^0 will appear on the right-hand-side. We choose the initial pressure for each time step in the $cGP(2)$ approach as the post-processed pressure solution, since it is not equal to the $cGP(2)$ pressure solution in t_n, as in [1, 4], by

$$P_{n+1}^0 := \tilde{p}_\tau(t_n) \;\text{ if }\; n \geq 1 \qquad \text{and} \qquad P_{n+1}^0 := p_\tau(t_n) \;\text{ if }\; n = 0. \tag{3.4.10}$$

There are several ways to obtain an initial value for the pressure at t_0. We used another time discretization scheme that does not depend on the initial pressure and approximates a solution $p_\tau(t_0)$ by using a very small time step size. Furthermore, after the first time step is solved by the $cGP(2)$-method, we can apply the post-processing step to correct the $cGP(2)$ solution. The pressure solution in t_n and so the initial pressure solution is of a higher order once this has been done due to the C^0-continuity of the pressure solution, which will be proved in a forthcoming paper by Friedhelm Schieweck, Markus Bause and Gunar Matthies.

Then, for given initial values $\boldsymbol{U}_n^0 = (U_n^0, V_n^0)$ and P_n^0, we solve the following system to find $U_n^1, U_n^2, V_n^1, V_n^2$ and P_n^1, P_n^2 with

$$u = \begin{bmatrix} u^1 \\ u^2 \end{bmatrix} = \begin{bmatrix} U_n^1 \\ U_n^2 \end{bmatrix}, \quad v = \begin{bmatrix} v^1 \\ v^2 \end{bmatrix} = \begin{bmatrix} V_n^1 \\ V_n^2 \end{bmatrix}, \quad p = \begin{bmatrix} p^1 \\ p^2 \end{bmatrix} = \begin{bmatrix} \tau P_n^1 \\ \tau P_n^2 \end{bmatrix}, \tag{3.4.11}$$

such that

$$\begin{bmatrix} A(u,v) & 0 & B_u \\ 0 & A(u,v) & B_v \\ B_u^T & B_v^T & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ p \end{bmatrix} = \begin{bmatrix} R_u \\ R_v \\ 0 \end{bmatrix}, \tag{3.4.12}$$

where the block matrices are given

$$\begin{aligned} B_u &= \begin{bmatrix} B_1 & 0 \\ 0 & B_1 \end{bmatrix}, \quad B_v = \begin{bmatrix} B_2 & 0 \\ 0 & B_2 \end{bmatrix}, \\ A(u,v) &= \begin{bmatrix} M + \frac{\tau_n}{2} L + \frac{\tau_n}{2} N(u^1, v^1) & \frac{1}{4} M \\ -4M & 2M + \frac{\tau_n}{2} L + \frac{\tau_n}{2} N(u^2, v^2) \end{bmatrix}, \\ R_u &= \begin{bmatrix} \frac{\tau_n}{2}(\underline{F}_n^1 + \frac{1}{2}\underline{F}_n^0) + \frac{5}{4} M U_n^0 - \frac{\tau_n}{4}(L + N(\boldsymbol{U}_n^0)) U_n^0 - \frac{\tau_n}{4} B_1 P_n^0 \\ \frac{\tau_n}{2}(\underline{F}_n^2 - \underline{F}_n^0) - 2M U_n^0 + \frac{\tau_n}{2}(L + N(\boldsymbol{U}_n^0)) U_n^0 + \frac{\tau_n}{2} B_1 P_n^0 \end{bmatrix}, \end{aligned} \tag{3.4.13}$$

and the right side of the block system reads

$$R_v = \begin{bmatrix} \frac{\tau_n}{2}(\underline{G}_n^1 + \frac{1}{2}\underline{G}_n^0) + \frac{5}{4} M V_n^0 - \frac{\tau_n}{4}(L + N(\boldsymbol{U}_n^0)) V_n^0 - \frac{\tau_n}{4} B_2 P_n^0 \\ \frac{\tau_n}{2}(\underline{G}_n^2 - \underline{G}_n^0) - 2M V_n^0 + \frac{\tau_n}{2}(L + N(\boldsymbol{U}_n^0)) V_n^0 + \frac{\tau_n}{2} B_2 P_n^0 \end{bmatrix}.$$

To confirm our choice of the initial condition for the pressure variable, we will investigate it in the numerical results presented in Chapter 6. A superconvergence behavior of the more accurate pressure $cGP\text{-}C1(k+1)$-method in the Gauss-Lobatto points will also be shown. This is further proved in a forthcoming paper by Friedhelm Schieweck, Markus Bause and Gunar Matthies.

The lack of an initial condition for the pressure variable is often the main reason for choosing another quadrature formula, such as the Gauss-Legendre formula in the $cGP(k)$-method. Inserting another quadrature formula leads to different coefficients $\alpha_{i,j}$ and $\beta_{i,j}$, such that the initial pressure does not appear on the right-hand-side. Thus, no initial pressure is needed for the solution of the coupled block system of the $cGP(k)$-method. However, we do not easily compute the solutions of velocity and pressure at the final time t_n of the time interval I_n when solving the coupled block system. We have to perform an additional extrapolation step of the discrete time points to get the solution at the end point of the time interval. In view of a global-in-time implementation, it is beneficial to disclaim the extrapolation step and, thus not to use the Gauss-Legendre formula but to stick to the Gauss-Lobatto formula. Especially by using the higher order $cGP\text{-}C1(3)$-method, the initial pressure for the next time interval is not a problem anymore.

3.4.3. *cGP-C*1(3)-method

Here, we show the post-processing step that is performed to obtain the discrete solutions $\tilde{\mathbf{u}}_\tau, \tilde{p}_\tau$ of the *cGP-C*1(3)-method by using the initial vectors $\mathbf{U}_n^0, P_n^0$ and the solution $\mathbf{U}_n^1, \mathbf{U}_n^2, P_n^1, P_n^2$ of the *cGP*(2)-method at each time interval, like explained before. From the spatial semi-discretization it is obvious that we now have to compute the vectors $\mathbf{a}_n := (a_n^u, a_n^v) \in \mathbb{R}^{2m_h}$ and $b_n \in \mathbb{R}^{n_h}$ at each time interval I_n, which solve the following system:

$$\begin{aligned}\begin{pmatrix} M & 0 & B_1 \\ 0 & M & B_2 \\ B_1^T & B_2^T & 0 \end{pmatrix} \begin{pmatrix} a_n^u \\ a_n^v \\ b_n \end{pmatrix} = \begin{pmatrix} \underline{F}_{n,2} \\ \underline{G}_{n,2} \\ 0 \end{pmatrix} - \begin{pmatrix} L+N(U_n^2) & 0 & B_1 \\ 0 & L+N(V_n^2) & B_2 \\ B_1^T & B_2^T & 0 \end{pmatrix} \begin{pmatrix} U_n^2 \\ V_n^2 \\ P_n^2 \end{pmatrix} \\ - \begin{pmatrix} M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \chi_n^u \\ \chi_n^v \\ 0 \end{pmatrix},\end{aligned} \tag{3.4.14}$$

where $\chi_n^u, \chi_n^v \in \mathbb{R}^{m_h}$ are given by

$$(\chi_n^u, \chi_n^v)^T = \sum_{j=0}^{2} \mathbf{U}_n^j \phi'_{n,j}(t_n),$$

where U_n^0 is defined in (3.4.3).

Note that this is only a linear saddle point system, even for a highly non-linear equation, which has to be solved in order to apply the post-processing step. We then obtain the discrete solution $\tilde{\mathbf{u}}_\tau(t), \tilde{p}_\tau(t)$ at some time point $t \in I_n$ by the formula

$$\tilde{\mathbf{u}}_\tau(t) = \mathbf{u}_\tau(t) + \mathbf{a}_n \zeta_n(t) \quad \text{and} \quad \tilde{p}_\tau(t) = p_\tau(t) + b_n \zeta'_n(t), \qquad t \in I_n, \tag{3.4.15}$$

where the polynomial ζ is defined in Theorem 3.2.

3.4.4. Linearization by fixed-point iteration

To handle the non-linear convective part of the Navier-Stokes equations, i.e. $(\mathbf{u} \cdot \nabla)\mathbf{u}$, we first linearize the problem to solve it. In the case of the *cGP*(2)-method, we solve the saddle point system (3.4.12)

$$\begin{bmatrix} A(u,v) & 0 & B_u \\ 0 & A(u,v) & B_v \\ B_u^T & B_v^T & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ p \end{bmatrix} = \begin{bmatrix} R_u \\ R_v \\ 0 \end{bmatrix},$$

for each time interval. Here, we will solve this by the fixed-point equation, although the Newton's method would be possible. In [31], this was also done for the *cGP*(2)-method, and the linearization by fixed-point iteration is considered as well as the Newton's method. Our presentation follows that of the previous reference. We perform an outer iteration, which is generally non-linear and in each outer iteration step, and we solve a coupled linear system with iterates $(u_\iota, v_\iota, p_\iota)$. Therefore, we set the initial guess (u_0, v_0, p_0) of the non-linear iteration to the solution of the previous time interval at time t_{n-1} for $n > 1$ or as the space-discrete initial solution for $n = 1$, which is defined by

$$u_0 = \begin{bmatrix} u_0^1 \\ u_0^2 \end{bmatrix} = \begin{bmatrix} U_n^0 \\ U_n^0 \end{bmatrix}, \; v_0 = \begin{bmatrix} v_0^1 \\ v_0^2 \end{bmatrix} = \begin{bmatrix} V_n^0 \\ V_n^0 \end{bmatrix}, \; p_0 = \begin{bmatrix} p_0^1 \\ p_0^2 \end{bmatrix} = \begin{bmatrix} \tau P_n^0 \\ \tau P_n^0 \end{bmatrix}.$$

Starting from the previous iterate $(u_\iota, v_\iota, p_\iota)$, we compute the defect vector

$$\begin{bmatrix} d_u \\ d_v \\ d_p \end{bmatrix} = \begin{bmatrix} R_u \\ R_v \\ 0 \end{bmatrix} - \begin{bmatrix} A(u_\iota, v_\iota) & 0 & B_u \\ 0 & A(u_\iota, v_\iota) & B_v \\ B_u^T & B_v^T & 0 \end{bmatrix} \begin{bmatrix} u_\iota \\ v_\iota \\ p_\iota \end{bmatrix}.$$

Then, we solve a linear auxiliary subproblem with the defect vector as the right-hand-side

$$\begin{bmatrix} A(u_\iota, v_\iota) & 0 & B_u \\ 0 & A(u_\iota, v_\iota) & B_v \\ B_u^T & B_v^T & 0 \end{bmatrix} \begin{bmatrix} \Delta u_\iota \\ \Delta v_\iota \\ \Delta p_\iota \end{bmatrix} = \begin{bmatrix} d_u \\ d_v \\ d_p \end{bmatrix}.$$

Afterwards, we update the iterate to get the next approximation $(u_{\iota+1}, v_{\iota+1}, p_{\iota+1})$

$$\begin{bmatrix} u_{\iota+1} \\ v_{\iota+1} \\ p_{\iota+1} \end{bmatrix} = \begin{bmatrix} u_\iota \\ v_\iota \\ p_\iota \end{bmatrix} + \begin{bmatrix} \Delta u_\iota \\ \Delta v_\iota \\ \Delta p_\iota \end{bmatrix}.$$

The non-linear iteration procedure is terminated when a stopping criterion is satisfied. In our numerical results, the non-linear iteration stops when the L^2-norm of the defect is less than 10^{-12}. Then, the last iterate $(u_{\iota+1}, v_{\iota+1}, p_{\iota+1})$ is accepted as a sufficiently accurate approximation to the exact solution.

4

Adaptive time step control

In this thesis, we use the $cGP(2)$-method and its post-processing step to estimate an error per time interval of the $cGP(2)$-method. The $cGP(2)/cGP\text{-}C1(3)$-error estimator has already been studied for convection-diffusion-reaction equations by Ahmed and John [2] for the $cGP(k)$- and $dG(k-1)$-methods. They illustrated that the time step sizes correctly represent the dynamics of the solution. Furthermore, the $cGP(2)$-method is the best method in terms of efficiency and accuracy compared to $cGP(3)$-, $dG(1)$-, $dG(2)$-methods and the adaptive Crank-Nicolson scheme [35]. The $cGP(2)/cGP\text{-}C1(3)$-error estimator leads to a smaller number of time steps for the same accuracy. Ahmed and John have the impression that error estimators based on two solutions of different order are more reliable than error estimators based on two solutions of the same order. These results underline the advantages of the $cGP(2)/cGP\text{-}C1(3)$-error estimator and motivate to focus only on this method.

First, we consider error estimators for the heat equation and then for incompressible flow problems. In particular, we extend this approach for the pressure variable. Since we use the post-processed pressure solution as the initial condition for the pressure at each time step (3.4.10), we cannot determine the pressure error estimator like the velocity error estimator. Therefore, we use an approach similar to that in [29] to obtain a higher order pressure solution.

On the other hand, we will study and later numerically investigate new global adaptive strategies that support a global-in-time solution technique. To construct such a global strategy, we will perform adaptive iterations in which all time steps are considered. Therefore, we assemble a completely new time grid in each adaptive iteration in the so-called control strategy. In particular, this control strategy involves an interpolation of time step sizes and error estimates. This has the advantage that we know all error estimates at all times, so we can use information about future times instead of only considering the past time steps. A further step in the control strategy, which we discuss in the last section of this chapter, is to use a controller to determine a new time step size based on the interpolated time step sizes and error estimates. The well-known controllers are typically used in a sequential time step control. In principle, they do not need to be changed, but our global strategy offers further possibilities in their application, so that they can additionally be adapted to a global strategy.

4.1. Error estimator

The construction of an accurate error estimator is the most important part for an adaptive time step control. Here, we take advantage of the higher order solution obtained by the post-processing step to estimate an error of the $cGP(2)$-method. This approximation of an error estimator is not restricted to the $cGP(2)$-method such that other choices of k are possible.

4.1.1. Error estimator for evolution problems

The $cGP(2)$-method applied to the heat equation is superconvergent with order four in the discrete time points, measured in the L^∞-norm and with order three in the entire time interval, measured in the L^2-norm, while the cGP-$C1(3)$-method is of order four in the entire time interval, see, e.g., [46]. Thus, we obtain an accurate error estimator of the $cGP(2)$-method in the L^2-error norm. The error estimator is based on the L^2-error of the numerical $cGP(2)$-solution $u_{h,\tau}(t)$ and its post-processed solution $\tilde{u}_{h,\tau}(t)$ as the estimator of the analytical error of the $cGP(2)$-method. It reads

$$\eta_n = \frac{1}{\sqrt{\tau_n}} ||u_{h,\tau} - \tilde{u}_{h,\tau}||_{L^2(I_n,L^2(\Omega))}, \quad t \in I_n, \tag{4.1.1}$$

for the time interval I_n with $n = 1,2,\ldots,N$.

The discrete solution $u_{h,\tau}(t)$ at the time point t is determined via the polynomial ansatz (3.1.5) and the C^1-solution $\tilde{u}_{h,\tau}(t)$ via the post-processing step (3.2.1).

4.1.2. Error estimators for incompressible flow problems

The $cGP(2)$-method applied to the Navier-Stokes equations retains the same convergence properties for velocity as for the heat equation, see, e.g., [1, 5]. Therefore, we can also calculate an accurate estimate of the error for the $cGP(2)$-method in the L^2-error norm. We can further identify error estimators for pressure and other values of interest such as drag and lift coefficients. In the case of velocity $\mathbf{u}$, the error estimator is obtained as above, but here it is based on the $L^2(I_n,\mathbf{V}_h)$-norm and to distinguish the error estimators for velocity and pressure, a superscript "u" is added:

$$\eta_n^u = \frac{1}{\sqrt{\tau_n}} ||\mathbf{u}_{h,\tau} - \tilde{\mathbf{u}}_{h,\tau}||_{L^2(I_n,(L^2(\Omega))^2)}, \quad t \in I_n, \tag{4.1.2}$$

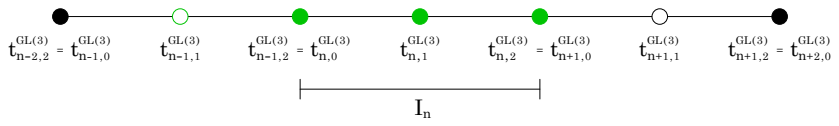
for the time interval I_n with $|I_n| = \tau_n$, $n = 1,2,\ldots,N$.

Since we have already used the higher order post-processed pressure solution as the initial condition for the pressure in the $cGP(k)$-method for each time interval, see (3.4.10), an error estimator of the $cGP(k)$ pressure solution is defined in a different way. This higher order post-processed pressure is of fourth order in the Gauss-Lobatto points, as will be shown in a forthcoming paper by Schieweck et al. mentioned above, and of third order in the L^2-norm as we will see both in the numerical studies in Section 6. Hence, we want to approximate a higher order pressure solution that is similar, w.r.t. the order of convergence, to the velocity solution, also of fourth order over the entire time interval.

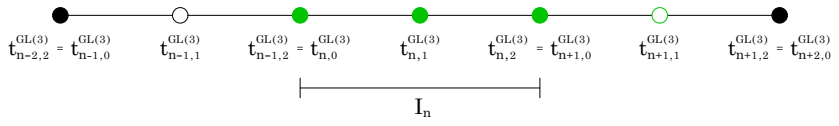
Hussain et al. presented a post-processing step for high order pressure approximations in the discrete time points t_n [29]. In contrast to our work, where we use the Gauss-Lobatto formula, they set up the $cGP(k)$-method with the Gauss-Legendre formula, where they needed an extrapolation step to obtain the solution at the discrete time points t_n. For example, the $cGP(2)$-method is based on the 2-point Gauss-Legendre formula with the intermediate points $t_{n,1}^{G(2)}$ and $t_{n,2}^{G(2)}$. Therefore, the required extrapolation uses these two intermediate time points and the point $t_{n,0} = t_{n-1}$ to extrapolate a solution for the discrete time point t_n. This requires the initial pressure, which is unknown from the classical $cGP(k)$-method. They then use two neighboring subintervals to have four intermediate time points and construct a cubic Lagrangian polynomial. Accordingly, they obtain the solution at the discrete time point t_n.

Due to the fact that we use the Gauss-Lobatto formula in the $cGP(2)$-method, we have the solutions at the three Gauss-Lobatto points $t_{n,j}^{GL(3)}$, $j = 0,1,2$, which are of order four. Like in the above mentioned paper [29], we can use the cubic Lagrange interpolation to obtain a higher order pressure solution to estimate the error. We will see in the numerical studies in Section 6, that this approach also leads to a higher pressure solution with an order of convergence of four at each time.

We apply the cubic Lagrange interpolation, shown in Figure 4.1, by using the three known Gauss-Lobatto points $t_{n,j}^{GL(3)}$, $j = 0,1,2$ on I_n, and a fourth point, which is chosen to be the time point $t_{n-1,1}^{GL(3)}$ or $t_{n+1,1}^{GL(3)}$, from one of the neighboring time intervals, as interpolation points.



(a) cLI,1



(b) cLI,2

Figure 4.1: Pressure error estimator: time points for cubic Lagrangian polynomial on I_n.

The choice of which time point to use, can depend on whether $t_{n-1,1}^{GL(3)}$ or $t_{n+1,1}^{GL(3)}$ is closer to an evaluated time point t. We denote by the suffix "cLI,1" the use of $t_{n-1,1}^{GL(3)}$ and by "cLI,2" the use of $t_{n+1,1}^{GL(3)}$ as fourth point, while the other three points are the Gauss-Lobatto points.

Thus, we can consider two polynomials, $j = 1,2$, to obtain the higher order post-

processed pressure solution $\tilde{p}_{h,\tau}^{cLI,j} \in \mathbb{P}_3$ by cubic Lagrangian interpolation

$$\tilde{p}_{h,\tau}^{cLI,j}(t) = \sum_{z=0}^{3} \tilde{p}_{h,\tau}(t_{n,z}^{cLI,j}) \phi_{n,z}^{cLI,j}(t) \;\; \forall\, t \;\in I_n, \tag{4.1.3}$$

where

$$\begin{aligned} \{t_{n,z}^{cLI,1}, z = 0,\ldots,3\} &:= \{t_{n-1,1}^{GL(3)}, t_{n,0}^{GL(3)}, t_{n,1}^{GL(3)}, t_{n,2}^{GL(3)}\}, \\ \{t_{n,z}^{cLI,2}, z = 0,\ldots,3\} &:= \{t_{n,0}^{GL(3)}, t_{n,1}^{GL(3)}, t_{n,2}^{GL(3)}, t_{n+1,1}^{GL(3)}\}, \end{aligned} \tag{4.1.4}$$

with the associated Lagrangian basis functions $\phi_{n,z}^{cLI,j} \in \mathbb{P}_3$. Then, we define $I_n = I_{n,1} \cup I_{n,2}$ with

$$I_{n,1} = (t_{n-1}, t_n^*], \qquad I_{n,2} = (t_n^*, t_n],$$

such that

$$\tilde{p}_{h,\tau}^{cLI}(t) := \begin{cases} \tilde{p}_{h,\tau}^{cLI,1}(t), & t \in I_{n,1}, \\ \tilde{p}_{h,\tau}^{cLI,2}(t), & t \in I_{n,2}. \end{cases} \tag{4.1.5}$$

It can be shown that there exists at least one t_n^* such that the following condition is satisfied:

$$I_{n,1} = \{t \in I_n \mid \; |t - t_{n-1,1}^{GL(3)}| \leq |t - t_{n+1,1}^{GL(3)}\}.$$

Since, $t_{n-1,1}^{GL(3)}$ in "cLI,1" does not exist for $n = 1$, we always use in the first time step "cLI,2". Of course, $t_{n+1,1}^{GL(3)}$ in "cLI,2" does not exist for $n = N$, so we always use "cLI,1" in the last time step. Moreover, we can also use only one set of interpolating nodes, $t_{n,z}^{cLI,1}$ or $t_{n,z}^{cLI,2}$ during the entire time interval.

To summarize the definition, the higher order post-processed pressure solution based on the cubic Lagrangian interpolation is denoted by $\tilde{p}_{h,\tau}^{cLI}$. However, we can set up a pressure error estimator in the L^2-norm based on the post-processed pressure solution $\tilde{p}_{h,\tau}$ of the *cGP-C*1(3)-method and the post-processed pressure solution based on the cubic Lagrangian interpolation $\tilde{p}_{h,\tau}^{cLI}$ by

$$\eta_n^p = \frac{1}{\sqrt{\tau_n}} ||\tilde{p}_{h,\tau} - \tilde{p}_{h,\tau}^{cLI}||_{L^2(I_n,L^2(\Omega))}, \quad t \in I_n, \tag{4.1.6}$$

with $|I_n| = \tau_n$, $n = 1,2,\ldots,N$.

We also introduce a pointwise pressure error estimator, which is calculated by the difference of the absolute pointwise error of the numerical solution $\tilde{p}_{h,\tau}(t)$ and the cubic Lagrangian interpolated solution $\tilde{p}_{h,\tau}^{cLI}(t)$

$$\eta_n^p(t) = ||\tilde{p}_{h,\tau}(t) - \tilde{p}_{h,\tau}^{cLI}(t)||_{L^2(\Omega)}, \quad t \in I_n \backslash \{t_{n,0}^{GL(3)}, t_{n,1}^{GL(3)}, t_{n,2}^{GL(3)}\}, \tag{4.1.7}$$

where t is any time point on the time interval I_n, except of the three Gauss-Lobatto points on this time interval, because of $\eta_n^p(t_{n,j}^{GL(3)}) = 0,\ j = 0,\ldots,2$.

Since the lift and drag coefficients are usually evaluated at a single time point, a pointwise error estimator is reasonable. As we will see in the numerical results in Chapter 6, the pressure error estimator in the integral-based L^2-norm, and the pointwise pressure

estimator have no significant differences and approximate the exact pressure error very well.

Additionally, we can also estimate a pointwise error of the lift values c_{lift}, calculated by $u_{h,\tau}$ and $\tilde{p}_{h,\tau}$. Since we need a higher order velocity and pressure solution, we adapt the error estimation of the pressure also for the lift and drag coefficients, so that the following $\eta_n^{c_l}$ is given by

$$\eta_n^{c_l}(t) = |c_{lift}(t) - c_{lift}^{cLI}(t)|, \quad t \in I_n \backslash \{t_{n,0}^{GL(3)}, t_{n,1}^{GL(3)}, t_{n,2}^{GL(3)}\}, \tag{4.1.8}$$

by using the lift coefficients c_{lift}^{cLI} with the *cGP-C*1 velocity solution $\tilde{u}_{h,\tau}$ and the "cLI" pressure solution $\tilde{p}_{h,\tau}^{cLI}$. Similarly, we can estimate the error for the drag coefficient $\eta_n^{c_d}$.

Especially for solving flow problems, it may be more efficient in the adaptive approach to use relative error estimators rather than absolute error estimators. However, we divide the errors by the higher order solution or in the case of an error estimator for the lift or drag coefficient, computed by the lift or drag values, which was calculated by the higher order solutions for that time interval. The relative error estimators are given by

$$\eta_n^{u,rel} = \frac{1}{\sqrt{\tau_n}} || \frac{\mathbf{u}_{h,\tau} - \tilde{\mathbf{u}}_{h,\tau}}{\tilde{\mathbf{u}}_{h,\tau}} ||_{L^2(I_n,(L^2(\Omega))^2)}, \tag{4.1.9}$$

$$\eta_n^{p,rel} = \frac{1}{\sqrt{\tau_n}} || \frac{p_{h,\tau} - \tilde{p}_{h,\tau}^{cLI}}{\tilde{p}_{h,\tau}^{cLI}} ||_{L^2(I_n,L^2(\Omega))}, \tag{4.1.10}$$

$$\eta_n^{c_l,rel}(t) = | \frac{c_{lift}(t) - c_{lift}^{cLI}(t)}{c_{lift}^{cLI}(t)} |, \qquad t \in I_n \backslash \{t_{n,0}^{GL(3)}, t_{n,1}^{GL(3)}, t_{n,2}^{GL(3)}\}. \tag{4.1.11}$$

Furthermore, we can use a mixed of relative and absolute error estimator, e.g. for the velocity error estimator. We check if the velocity solution is less than 0.001 and, if so, use an absolute one otherwise a relative one.

$$\eta_n^{u,relabs} = \begin{cases} \eta_n^u & ||\mathbf{u}_{h,\tau}(t_n)||_{L^2(\Omega)} < 0.001, \\ \eta_n^{u,rel} & else. \end{cases} \tag{4.1.12}$$

After estimating certain errors of interest, we now turn to the adaptive strategy, where the error estimators are used in order to modify the time step sizes.

4.2. Adaptive time step control in a global-in-time solution

In the classical sequential approach, shown in Figure 4.2, we solve the resulting system sequentially for one time step after another. More precisely, one solves the resulting system for one time step, then approximates an error estimator and accepts or rejects the used time step size depending on the value of the error estimator and the given error tolerance. If the time step size is rejected, a new one is determined by using so-called controllers. Then, the system is solved again by using the new time step size. This procedure is repeated until the error estimator is below a certain tolerance and the final time is reached. Therefore, a classical adaptive time step control does not work for global-in-time approaches, where all time steps are solved simultaneously.

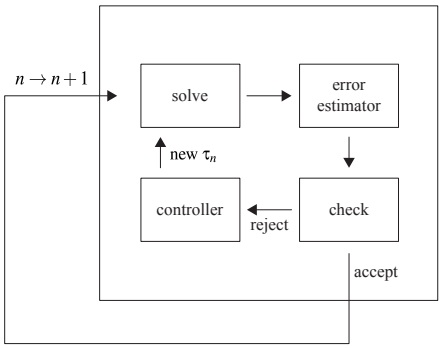


Figure 4.2: Sequential adaptive strategy.

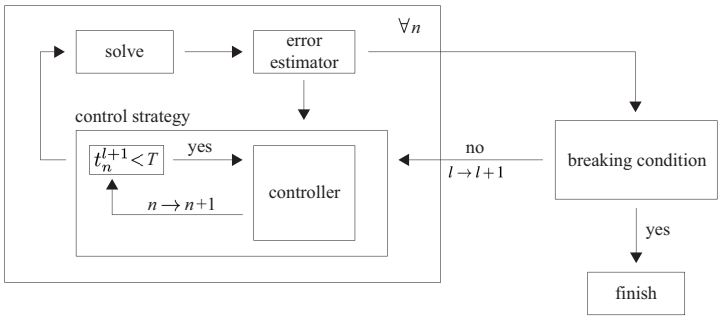


Figure 4.3: Global adaptive strategy for a global-in-time approach.

Thus, we solve one global system for all time steps, which is described in more detail in Chapter 5, where we will further discuss how to handle the initial condition for the pressure in such a global approach. However, we take it for granted here that our adaptive strategy treats all time steps. In the global adaptive approach, which is illustrated in Figure 4.3, we determine the solutions of the $cGP(2)$-method for all time steps. Next, we perform the post-processing step to obtain the $cGP\text{-}C1(3)$ solutions. Then we evaluate the error estimates for each time step. Finally, we obtain error estimates for all time steps and check whether the stopping criterion is satisfied. This is done by analyzing whether the error estimates are smaller than the given tolerance. If this is not the case, we start a next adaptive iteration $l+1$, where we first generate a new time grid. Therefore, we introduce an index for the adaptive level l in order to indicate the time grid at level l. This is constructed sequentially by applying the control strategy.

4.3. Control strategy

In the adaptive strategy, we perform a control strategy to obtain the new time grid determined by new time step sizes. To adopt a new time step size, we have to control the value of the error estimator so that it is smaller than a given tolerance. This is done by so-called controllers, which will be discussed in the next Section 4.4. But first, we consider the adaptive control strategy, which starts after the error estimators for each time step have been determined. Here, it is of no consequence, which specific error estimator we use whether it is pointwise or in the L^2-norm, or whether it is related to the the heat equation

or the Navier-Stokes equations, since all error estimators are scalar values. Suppose we are in the l-th adaptive iteration. The maximum number of adaptive iterations is denoted by L, so $l = 1,\dots,L$. We define the grid at level l by

$$G^l := \{t_n^l \in I \mid n = 0,\dots N^l,\ 0 = t_0^l < t_1^l < \cdots < t_{N^l}^l = T\},$$

and the corresponding time step sizes by

$$\tau_n^l := t_n^l - t_{n-1}^l, \qquad \forall n = 1,\dots,N^l. \tag{4.3.1}$$

Furthermore, we take advantage of the global adaptive approach by knowing all used time step sizes and, more importantly, all resulted error estimates at each discrete time point. Therefore, we can interpolate the time step sizes and error estimates with interpolation points t_n^l, $n = 0,\dots,N^l$. There are several choices of interpolation methods, but considering the next subsection, we define two possible function spaces here by

$$\begin{aligned} Z_0(G^l) &:= \{w \in L^2(I;\mathbb{R}) \mid w\,|_{I_n} \in \mathbb{P}_0(I_n,\mathbb{R})\ \forall I_n = (t_{n-1}^l, t_n^l], n = 1,\dots,N^l\},\\ \tilde{Z}_k(G^l) &:= \{w \in C^0(I;\mathbb{R}) \mid w\,|_{I_n} \in \mathbb{P}_k(I_n,\mathbb{R})\ \forall I_n = (t_{n-1}^l, t_n^l], n = 1,\dots,N^l\}. \end{aligned}$$

Moreover, we define a time step function $\bar{\tau}$ associated with the time grid G^l, e.g. for the case of piecewise constant interpolation $\bar{\tau} \in Z_0(G^l)$ or for piecewise linear interpolation $\bar{\tau} \in \tilde{Z}_1(G^l)$ such that

$$\bar{\tau}(t_{n-1}^l) := \tau_n^l = t_n^l - t_{n-1}^l \ \ \forall n = 1,\dots,N^l, \quad \text{and} \quad \bar{\tau}(T) := \bar{\tau}(t_{N^l-1}^l).$$

Analogous to the error estimator for a set of $\{\eta_n^l \mid n = 0,\dots,N^l\}$, we can define a function $\bar{\eta} \in Z_0(G^l)$ or $\bar{\eta} \in \tilde{Z}_1(G^l)$ such that

$$\bar{\eta}(t_{n-1}^l) := \eta_n^l \quad \forall n = 1,\dots,N, \qquad \text{and} \qquad \bar{\eta}(T) := \bar{\eta}(t_{N^l-1}^l).$$

Due to the interpolation, we obtain error estimates and time step sizes for each required time point, which are approximated by values of multiple discrete time points, depending on the chosen interpolation. However, at least two time points are involved, so at least one time point is in the future. Thus, in this global strategy, knowledge of the future is incorporated into the calculation of the new time step size.

Furthermore, in our global approach, the entire time grid is reconstructed sequentially, so that we need new time step sizes τ_n^{l+1} to determine a specific new time grid point $t_n^{l+1} = t_{n-1}^{l+1} + \tau_n^{l+1}$. These are obtained by using a controller

$$\tau_n^{l+1} = contr(t_{n-1}^{l+1}, \bar{\tau}, \bar{\eta}),$$

where t_{n-1}^{l+1} is the last given time point on G^{l+1} and functions $\bar{\tau}, \bar{\eta} : I \to \mathbb{R}$.

Finally, the new time grid G^{l+1} is constructed from the old time grid G^l and the interpolation functions $\bar{\tau}, \bar{\eta}$ with $t_0^{l+1} = 0$ by the recursion $t_{n-1}^{l+1} \to t_n^{l+1}$:

$$\begin{cases} \tau_n^{l+1} = contr(t_{n-1}^{l+1}, \bar{\tau}, \bar{\eta}) \\ t_n^{l+1} = t_{n-1}^{l+1} + \tau_n^{l+1}. \end{cases}$$

If the last new time point is greater than the final time, it is set as the final time, so that $t^{l+1}_{N^{l+1}} = T$ and the step size is calculated accordingly by $\tau(t^{l+1}_{N^{l+1}}) = t^{l+1}_{N^{l+1}} - t^{l+1}_{N^{l+1}-1}$.

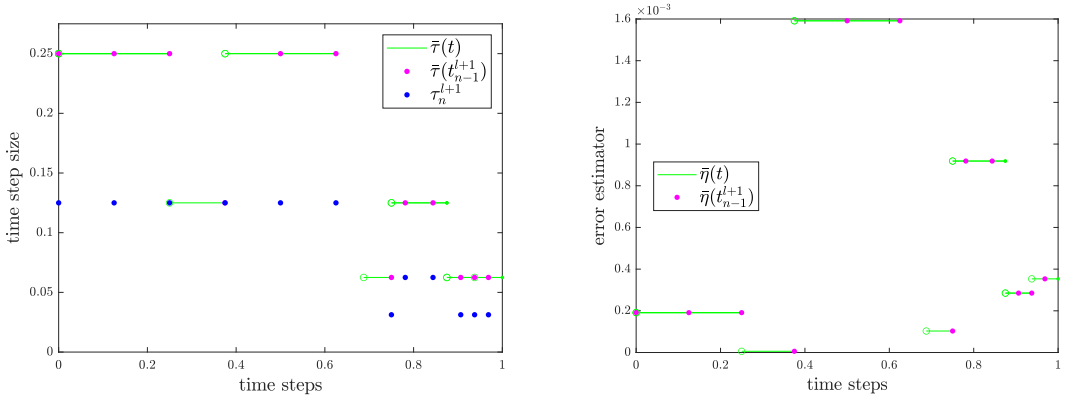
Before discussing the controllers, we first describe the determination of the time step sizes and the error estimator used in a controller to compute a new time step size. To introduce the topic, we start with an obvious *strategy using piecewise constant interpolation* and further investigate the *strategy using linear interpolation*. Other types of interpolation are also possible.

4.3.1. Strategy using piecewise constant interpolation

The first interpolation strategy is called *strategy using piecewise constant interpolation*, denoted by *CI*, and can be incorporated into the control strategy described above to simplify the interpolation of time step sizes and error estimators. To illustrate how the step sizes evolve within the control strategy, consider Figure 4.4. The x-axis describes a time grid and the y-axis represent the size of the time step sizes in Figure 4.4a. The green line indicates the interpolation function of the time step size over a time grid $G^l = [0,1]$ in the adaptive iteration l. The length of a green line is therefore identical with the time step size. Thus, the discrete time points t^l_n of the old time grid G^l are the beginning and endpoints of the green line. These are represented by a green circle and a green point, respectively. The pink dots show the interpolated old time step sizes, which are used in the controller to determine new time step sizes for the new time grid points on G^{l+1}, shown as blue dots. For example, the controller used here halves the interpolated time step size if the interpolated error estimator is greater than the tolerance, which is here set to 10^{-4}. Since a piecewise constant interpolation is used, it can be observed that the interpolated time step size for the new time point is consistent with the step size on the previous time grid G^l. The controller is then executed with the interpolated values, resulting in the new step size of the new time grid, as shown by the blue dot. The error estimators are shown in Figure 4.4b, which also shows the interpolation function and the interpolated values used, but now for the error estimators. For example, looking at the first time interval, we see that the error estimator is about 0.2 and thus bigger than the tolerance. Therefore, the first time step size is halved, when looking at Figure 4.4a again.

4.3.2. Strategy using linear interpolation

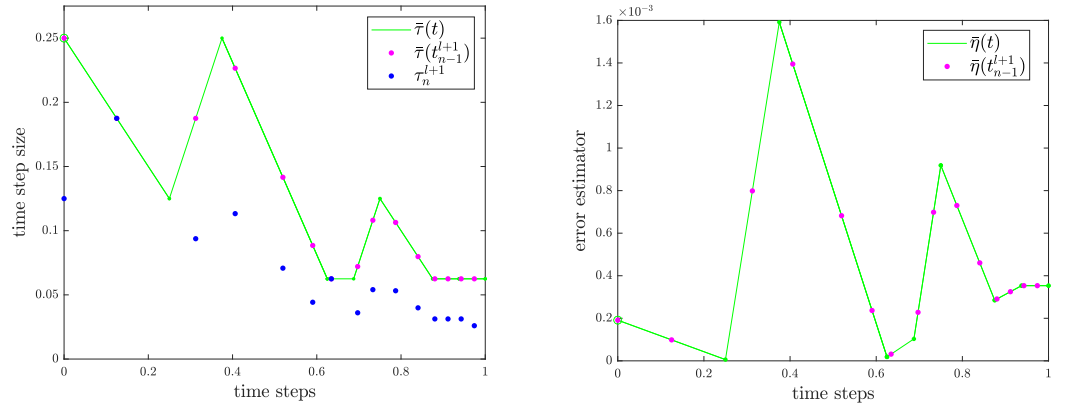
The second concept takes more advantage of the global adaptive approach, since the error estimators are known for all time grid points. This concept is called *strategy with linear interpolation*, denoted by *LI*, and is illustrated in Figure 4.5. The old and new time step sizes for the adaptive iteration and their evolution are shown in Figure 4.5a, while the corresponding error estimators are shown in Figure 4.5b. The same legend and variables as in Figure 4.4 are displayed in Figure 4.5a. Furthermore, the same old time step sizes $\tau(t^l_n)$ and old error estimators $\eta(t^l_n)$ were used, here in Figure 4.5b, as well as the same controller. However, linear interpolation is applied so that the new time step sizes do not coincide with the old ones at intermediate time points. The time step sizes are modified with respect to the future time step. First, we see a difference between the two interpolation strategies, Figures 4.4 and 4.5. The constant interpolation results in new time step sizes that are more stair-like than the linear interpolation. To illustrate, if we consider the



(a) Time step sizes. (b) Error estimators.

Figure 4.4: Strategy with piecewise constant interpolation.

two time step sizes of the time points $t_1^l = 0.25$ and $t_2^l = 0.375$, which are shown by green dots in Figure 4.5a, the second one is larger. Consequently, the interpolated time step size at time $t = 0.31$, which is illustrated with a pink dot, is larger than τ_1^l and smaller than τ_2^l at $t_1^l = 0.25$. Thus an interpolated error estimator, illustrated in Figure 4.5b, is built into the controllers, as shown by the pink dot. Since the error estimator at time $t = 0.31$ is about 0.0008, which is the second highest estimated error, the new time step size has been halved, which is illustrated with a blue dot in Figure 4.5a.



(a) Time step sizes. (b) Error estimators.

Figure 4.5: Strategy with linear interpolation.

Another interpolation, which is studied in the numerical tests, is the piecewise cubic Hermite interpolation with the function space $\tilde{Z}_3$, which is denoted by *CHI*.

In the sequential adaptive strategy, the primary objective is to perform a minimum number of time steps in order to reduce the computational effort while maintaining a certain level of solution accuracy. However, in our global approach, the minimum number of time steps in one adaptive iteration can be inefficient, if it results in a large number

of adaptive iterations. On the other hand, a small number of adaptive iterations can be inefficient if each iteration has a large number of time steps. Therefore, our controllers must combine and satisfy both design criteria.

4.4. Controller

So far, we have outlined the process for determining the time step sizes and error estimates employed in a controller. We will now consider time step controllers. In general, the controller adjusts the time step size so that the associated error estimator is less than a given tolerance. Therefore, the old time step size and the resulting error estimator for that time step size are considered. Basically, the controller does not change comparable to a sequentially adaptive approach, but as described in the section above, the possibilities of associated error estimators and step sizes change. The first and simplest controller we show doubles and halves the time step size, when the error estimator is smaller or greater than a given tolerance, denoted by *tol*. This controller does not take advantage of the characteristics of the time step scheme used. Therefore, we look at other controllers that use the knowledge of the convergence behavior of the methods. If we consider two time step sizes τ_1 and τ_2 with an associated error or error estimator $e(\tau_1)$ and $e(\tau_2)$, then we obtain the experimental order of convergence (EOC), which is given by

$$eoc := \frac{\log(e(\tau_1)/e(\tau_2))}{\log(\tau_1/\tau_2)}. \tag{4.4.1}$$

Since the experimental order of convergence explains how the error changes due to changes in the time step size, we can use this relationship to determine how the time step size should be changed to obtain an error estimate that is within tolerance.

An overview of controller types is given in [57]. We will test some of them in our adaptive strategy for the global-in-time approach to compare them with each other. In particular, we will look at the elementary local error controller (*EL*) and more advanced controllers such as the predictive controller (*PC11*) and the proportional-integral-derivative controller (*PID*). They are based on a feedback control point of view and have been applied to adaptive step size control of PDEs [6, 36], especially for the incompressible Navier-Stokes equations. The first controller depends only on the last time step, while the *PC11* controller includes the last two time steps. The *PID* controller, as the name suggests, has three control terms, which are the proportional, integral, and derivative terms. Therefore, the new step size depends on the last three time steps. Since the *PC11* and the *PID* controllers are two well-known controllers and give good results in [2], we also adjust them into the setting of the global approach.

4.4.1. Controllers for evolution problems

Now we define controllers that are executed in the control strategy and are based on interpolated values of time step sizes $\bar{\tau}(t_{n-1}^{l+1})$ and error estimators $\bar{\eta}(t_{n-1}^{l+1})$. They lead to a new time step size τ_n^{l+1}, where the new time grid point is computed by $t_n^{l+1} = t_{n-1}^{l+1} + \tau_n^{l+1}$, see (4.3.1).

The first and simplest control is to double or halve the time step sizes, denoted by *DH*.

Controller 4.1 (*DH*). *The optimal time step size* τ_n^{l+1} *is determined by*

$$\begin{cases} \frac{\bar{\tau}(t_{n-1}^{l+1})}{2}, & \bar{\eta}(t_{n-1}^{l+1}) > K_1 \cdot tol \\ \bar{\tau}(t_{n-1}^{l+1}) \cdot 2, & \bar{\eta}(t_{n-2}^{l+1}), \bar{\eta}(t_{n-1}^{l+1}) \leq K_2 \cdot tol \\ \bar{\tau}(t_{n-1}^{l+1}), & else. \end{cases}$$

Here we set $K_1 = 0.75$ and $K_2 = 0.01$, which will be used throughout the work, if nothing else is written. In this definition, the time step size is coarsened, if the error estimate is less than 0.01 multiplied by the tolerance, and reduced, if it is greater than 0.75 multiplied by the tolerance. These limits were chosen to prevent the resulting error estimates from alternating too frequently between coarsening and refining. Unlike sequential adaptive time-stepping techniques, a new time grid is constructed in each adaptive iteration. Therefore, changing the step size can affect the errors of later time grid points in the next adaptive iteration. If we set these limits too tightly, time step sizes that were not changed in the current adaptive iteration could be changed in the next adaptive iteration. In addition to the goal of minimizing the number of adaptive iterations, it is preferable to avoid frequent changes. These limits are applied consistently throughout the work of each controller, unless otherwise specified.

The next three controllers [57] depend on the order of convergence of the time integrator and have been modified to fit the global-in-time approach. Therefore, we determine how the step size should be changed so that the error estimator coincides with the given tolerance. If we consider the calculation of the EOC (4.4.1), where the new time step size and error estimator are τ_2 and $e(\tau_2)$ and the old time step size and error estimator are τ_1 and $e(\tau_1)$, then we can replace $e(\tau_2)$ by the tolerance and the EOC, denoted as *eoc*, by a parameter, which refers to the expected order of convergence ρ. By rearranging the equation after the new step size τ_2, we obtain the following equation of the controller to determine the new time step size. In addition, to prevent the next time step from being too large, a security parameter $\theta < 1$ is included.

Controller 4.2 (*EL*). *In the elementary local error controller [57], the optimal time step size* τ_n^{l+1} *is determined by*

$$\begin{cases} \bar{\tau}(t_{n-1}^{l+1})\theta(\frac{tol}{\bar{\eta}(t_{n-1}^{l+1})})^{\frac{1}{\rho}}, & \bar{\eta}(t_{n-1}^{l+1}) > K_1 \cdot tol \ \ or \ \ \bar{\eta}(t_{n-1}^{l+1}) \leq K_2 \cdot tol \\ \bar{\tau}(t_{n-1}^{l+1}), & else, \end{cases}$$

where θ *is a security parameter and* ρ *refers to the order of convergence.*

We set ρ as the order of convergence of the time-stepping scheme for a local error per unit step, see [57]. We use error estimates per unit step because we normalize the error with respect to the time step size. In the case of the $cGP(2)$-method, measured in the L^2-norm, the expected order of convergence is three such that $\rho = 3$.

Controller 4.3 (*PC11*). *In the PC11 controller [57], the optimal time step size* τ_n^{l+1} *is determined by*

$$\begin{cases} \frac{\bar{\tau}(t_{n-1}^{l+1})^2}{\bar{\tau}(t_{n-2}^{l+1})}\theta\left(\frac{tol \cdot \bar{\eta}(t_{n-2}^{l+1})}{\bar{\eta}(t_{n-1}^{l+1})^2}\right)^{\frac{1}{\rho}}, & \bar{\eta}(t_{n-1}^{l+1}) > K_1 \cdot tol \ \ or \ \ \bar{\eta}(t_{n-1}^{l+1}) \leq K_2 \cdot tol \\ \bar{\tau}(t_{n-1}^{l+1}), & else, \end{cases}$$

where θ *is a security parameter and* ρ *refers to the order of convergence.*

Since the *PC11* controller and the *PID* controller refer to more than one time point, they cannot be applied to the first or to first and second time interval. In these cases, the *EL* controller is used.

Controller 4.4 (*PID*). *In the PID controller [57], the optimal time step size* τ_n^{l+1} *is determined by*

$$\begin{cases} \bar{\tau}(t_{n-1}^{l+1})\theta\left(\frac{tol}{\bar{\eta}(t_{n-1}^{l+1})}\right)^{\frac{0.525}{\rho}}\left(\frac{\bar{\eta}(t_{n-2}^{l+1})}{\bar{\eta}(t_{n-1}^{l+1})}\right)^{\frac{0.225}{\rho}}\left(\frac{(\bar{\eta}(t_{n-2}^{l+1}))^2\bar{\eta}(t_{n-1}^{l+1})}{\bar{\eta}(t_{n-3}^{l+1})}\right)^{\frac{0.03}{\rho}}, & (\bar{\eta}(t_{n-1}^{l+1}) > K_1 \cdot tol \\ & \quad or\ \bar{\eta}(t_{n-1}^{l+1}) \leq K_2 \cdot tol) \\ \bar{\tau}(t_{n-1}^{l+1}), & else, \end{cases}$$

where θ *is a security parameter and* ρ *refers to the order of convergence.*

4.4.2. Controllers for incompressible flow problems

In the case of two primal variables, we can define our controllers presented above for both variables, here velocity and pressure, and further for the lift and/or drag coefficient. We use the suffix *-u-p* to denote the inclusion of velocity and pressure error estimates in a controller. We also add the suffix *-c_l* and/or *-c_d* to the lift and drag coefficient controllers. Of course, we treat the pressure estimator in the same way as the velocity error estimator in the control strategy. Therefore, we denote the interpolated velocity error estimator by $\bar{\eta}^u(t_{n-1}^{l+1})$, the interpolated pressure error estimator by $\bar{\eta}^p(t_{n-1}^{l+1})$ and the error estimator of the lift coefficient by $\bar{\eta}^{c_l}(t_{n-1}^{l+1})$ at the new time t_{n-1}^{l+1} to compute the next time t_n^{l+1}. Since we will see in the numerical studies in Section 6, that the elementary local error controller turns out to be the most suitable one for a global adaptive approach, we modify for example the *EL* controller to include the the pressure variable.

Controller 4.5 (*EL-u-p*). *Here, we use hierarchical case conditions to determine the optimal time step size* τ_n^{l+1} *by*

$$\begin{cases} \bar{\tau}(t_{n-1}^{l+1})\theta\left(\frac{tol}{\bar{\eta}^u(t_{n-1}^{l+1})}\right)^{\frac{1}{\rho}}, & \bar{\eta}^u(t_{n-1}^{l+1}) > K_1 \cdot tol\ \ or\ \ \bar{\eta}^u(t_{n-1}^{l+1}), \bar{\eta}^p(t_{n-1}^{l+1}) \leq K_2 \cdot tol \\ \bar{\tau}(t_{n-1}^{l+1})\theta\left(\frac{tol}{\bar{\eta}^p(t_{n-1}^{l+1})}\right)^{\frac{1}{\rho}}, & \bar{\eta}^p(t_{n-1}^{l+1}) > K_1 \cdot tol\ \ and\ \ \bar{\eta}^u(t_{n-1}^{l+1}) < K_1 \cdot tol \\ \bar{\tau}(t_{n-1}^{l+1}), & else, \end{cases}$$

where θ *is a security parameter and* ρ *refers to the order of convergence.*

In the case of the $cGP(2)$-method and cGP-$C1(3)$-method, measured in the L^2-norm, $\rho = 3$ for velocity and pressure, since both converges with order three. However, the previous controller includes the velocity and pressure error estimators. First, the velocity error estimator is considered. If it satisfies the reduction condition, then the time step size is modified based on the velocity error estimator. Otherwise, the time step size would be accepted with respect to the velocity error. Then, the pressure error estimator is considered and the time step size is modified accordingly. The time step size is only coarsened depending on the velocity if the velocity and pressure error estimators satisfy the coarsening criterion. This is hereafter referred to *hierarchical case conditions* in the following.

In addition, we can also construct a velocity and pressure adaptive controller that selects the most constrained error estimate for the velocity and pressure error estimates per time step, denoted by the suffix $-M$, since we are considering the maximum value.

Controller 4.6 (*EL-u-p-M*)**.** *Here, we use maximum case conditions to determine the optimal time step size* τ_n^{l+1} *by*

$$\begin{cases} \bar{\tau}(t_{n-1}^{l+1})\theta\left(\frac{tol}{\bar{\eta}^u(t_{n-1}^{l+1})}\right)^{\frac{1}{\rho}}, & \left(\bar{\eta}^u(t_{n-1}^{l+1}) > K_1 \cdot tol \text{ and } \bar{\eta}^p(t_{n-1}^{l+1}) \leq \bar{\eta}^u(t_{n-1}^{l+1})\right) \\ & or\ \ \bar{\eta}^u(t_{n-1}^{l+1}), \bar{\eta}^p(t_{n-1}^{l+1}) \leq K_2 \cdot tol \\ \bar{\tau}(t_{n-1}^{l+1})\theta\left(\frac{tol}{\bar{\eta}^p(t_{n-1}^{l+1})}\right)^{\frac{1}{\rho}}, & \bar{\eta}^p(t_{n-1}^{l+1}) > K_1 \cdot tol \text{ and } \bar{\eta}^u(t_{n-1}^{l+1}) < \bar{\eta}^p(t_{n-1}^{l+1}) \\ \bar{\tau}(t_{n-1}^{l+1}), & else, \end{cases}$$

where θ *is a security parameter and* ρ *refers to the order of convergence.*

Additionally, we present a controller that includes only the error estimator for the lift coefficient. Here, the criteria for coarsening and refinement are defined as in the case of the *EL* controller from the previous section.

Controller 4.7 (*EL-l*)**.** *The optimal time step size* τ_n^{l+1} *is determined by*

$$\begin{cases} \bar{\tau}(t_{n-1}^{l+1})\theta\left(\frac{tol}{\bar{\eta}^{c_l}(t_{n-1}^{l+1})}\right)^{\frac{1}{\rho}}, & \bar{\eta}^{c_l}(t_{n-1}^{l+1}) > K_1 \cdot tol \ \ or \ \ \bar{\eta}^{c_l}(t_{n-1}^{l+1}), \leq K_2 \cdot tol \\ \bar{\tau}(t_{n-1}^{l+1}), & else, \end{cases}$$

where θ *is a security parameter and* ρ *refers to the order of convergence.*

The lift-only controller introduced previously has the same case conditions as the *EL* Controller 4.2, so obviously we can develop an only-velocity and an only-pressure controller of the same type. Now we introduce a controller that includes a combination of velocity, pressure, and error estimates for the lift coefficient.

Controller 4.8 (*EL-u-p-l*)**.** *The optimal time step size* τ_n^{l+1} *is determined by*

$$\begin{cases} \bar{\tau}(t_{n-1}^{l+1})\theta\left(\frac{tol}{\bar{\eta}^u(t_{n-1}^{l+1})}\right)^{\frac{1}{\rho}}, & \bar{\eta}^u(t_{n-1}^{l+1}) > K_1 \cdot tol \ \ or \ \ \bar{\eta}^u(t_{n-1}^{l+1}), \bar{\eta}^p(t_{n-1}^{l+1}), \bar{\eta}^{c_l}(t_{n-1}^{l+1}) \leq K_2 \cdot tol \\ \bar{\tau}(t_{n-1}^{l+1})\theta\left(\frac{tol}{\bar{\eta}^p(t_{n-1}^{l+1})}\right)^{\frac{1}{\rho}}, & \bar{\eta}^p(t_{n-1}^{l+1}) > K_1 \cdot tol \ \ and \ \ \bar{\eta}^u(t_{n-1}^{l+1}) < K_1 \cdot tol \\ \bar{\tau}(t_{n-1}^{l+1})\theta\left(\frac{tol}{\bar{\eta}^{c_l}(t_{n-1}^{l+1})}\right)^{\frac{1}{\rho}}, & \bar{\eta}^{c_l}(t_{n-1}^{l+1}) > K_1 \cdot tol \ \ and \ \ \bar{\eta}^u(t_{n-1}^{l+1}), \bar{\eta}^p(t_{n-1}^{l+1}) < K_1 \cdot tol \\ \bar{\tau}(t_{n-1}^{l+1}), & else, \end{cases}$$

where θ *is a security parameter and* ρ *refers to the order of convergence.*

Here, we used *hierarchical case conditions*, where we first examine the velocity variable, then the pressure variable, and finally the lift coefficient. The time step size will only

be coarsened depending on the velocity if all three error estimators satisfy the coarsening condition.

Of course we can also set up a *EL-u-p-l-M* controller with *maximum case conditions*, where the case condition depends on the maximum of the three error estimators per time step, which have no hierarchical order. Instead of a lift controller, these controllers can easily be set up for the drag coefficient in a straightforward manner.

In the numerical studies in Chapter 6, we will investigate the previously introduced controllers for different interpolation strategies, which are added to the name of the particular controller, for example, the *EL-u-p-l-LI* controller uses the *EL* controller for velocity, pressure, and lift coefficient with linear interpolation of time step sizes and error estimators with *hierarchical case conditions*.

4.4.3. Further control restrictions

There are, of course, many ways to add further restrictions. We will discuss some of these.

Restriction on time step sizes

The most classical restriction, see [27], is to prevent the step sizes from shrinking or growing too much, so the already computed time step size is limited by

$$\tau_n^{l+1} = \min\{s_{max}\bar{\tau}(t_{n-1}^{l+1}), \max\{s_{min}\bar{\tau}(t_{n-1}^{l+1}), \tau_n^{l+1}\}\}.$$

In this work, we set $s_{max} = 2$ and $s_{min} = 0.05$ and it is used in all variants of the controller.

Restriction on case conditions

Since we build new time grids in each adaptive iteration, the time grids over successive adaptive iterations coincide from the start time until the time we change the time step size for the first time. Figure 4.6 illustrates the time grids G^l and G^{l+1}. The time points match until the step size is reduced from 0.125 to 0.115 and we insert the new time point at 0.49. Therefore, we do not need to solve the problem up to this new time grid point, because we already solved the same problem at the same time grid points in the last adaptive iteration. Therefore, the new time grid can be solved starting from a later time grid point and all in all we compute the solution in fewer time intervals.

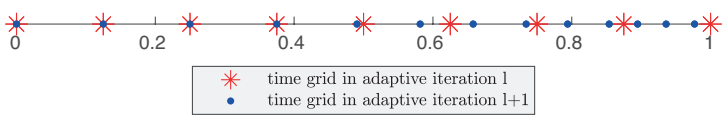


Figure 4.6: Time grids per adaptive iterations.

The above mentioned criteria of cases with $K_1 = 0.01$ with $0.01 \cdot tol$ for coarsening and $K_1 = 0.75$ with $0.75 \cdot tol$ for reducing a time step size are set up to ensure less changes in the time step size over the adaptive iterations, but they do not specifically ensure less

time step modifications during an adaptive iteration. To pursue this goal, we can ignore the coarsening case for time step sizes and reduce time step sizes only if the associated error estimator is larger than the given tolerance. These case conditions can be assumed until a time step size is reduced, otherwise the previously described criteria are assumed. To illustrate, consider the *EL-LI* controller with these new conditions.

Controller 4.9 (*EL-LI* controller with limiting of case conditions)**.** *The optimal time step size* τ_n^{l+1} *is determined by*

$$\begin{cases} \bar{\tau}(t_{n-1}^{l+1})\theta(\frac{tol}{\bar{\eta}(t_{n-1}^{l+1})})^{\frac{1}{\rho}}, & t_n^{l+1} = t_n^l, \ \textit{and} \ \bar{\eta}(t_{n-1}^{l+1}) > tol \\ \bar{\tau}(t_{n-1}^{l+1})\theta(\frac{tol}{\bar{\eta}(t_{n-1}^{l+1})})^{\frac{1}{\rho}}, & t_n^{l+1} \neq t_n^l \ \textit{and} \ (\bar{\eta}(t_{n-1}^{l+1}) > K_1 \cdot tol \ \textit{or} \ \bar{\eta}(t_{n-1}^{l+1}) < 0.01 \cdot tol) \\ \bar{\tau}(t_{n-1}^{l+1}), & \textit{else}, \end{cases}$$

where θ *is a security parameter and* ρ *refers to the order of convergence.*

Consideration of the experimental order of convergence

The final controller provided in this thesis incorporates the experimental order of convergence to ensure that an error estimator, and thus a controller, behaves as it should. To calculate an EOC, we have to consider two different time grids. Therefore, the first two adaptive iterations are performed with two equidistant time step sizes. In general, we assume that we are in the $(l+1)$-th adaptive iteration and want to reconstruct the new time grid G^{l+1}. Then, after interpolation of the time step sizes and error estimators, we compute the EOC of the error estimator. Therefore, we consider the last two time grids G^{l-1} and G^l. For each time grid point t_m^{l-1}, $m = 1, \ldots, \mid G^{l-1} \mid$, we first have to find the nearest time points of t_n^l, $n = 1, \ldots, \mid G^l \mid$ by

$$n^* = \arg\min_n \mid t_m^{l-1} - t_n^l \mid, \qquad \text{for } m = 1, \ldots, \mid G^{l-1} \mid .$$

Then we can compute the EOC values eoc_m^{l-1}, see (4.4.1), to each time point t_m^{l-1}, $m = 1, \ldots, \mid G^{l-1} \mid$ by the time step sizes $\tau_{n^*}^l$ and τ_m^{l-1} and the associated error estimators for each time point t_m^{l-1}, $m = 1, \ldots, \mid G^{l-1} \mid$.

Before determining new time step sizes to construct G^{l+1} based on G^l, we need to find the corresponding EOC value eoc_n^l on the time grid G^l by searching again the nearest time points for each time step. Then we can check if the EOC value is between a certain range $[\rho - 1, \rho + 1]$. If it is, the error estimator converges as it should, such that we can continue with a controller and change the time step size accordingly. If not, the time step size is halved. Then the next new time grid point is checked.

Controller 4.10 (Controller with consideration of EOC)**.** *The optimal time step size* τ_n^{l+1} *is determined by*

$$\begin{cases} \bar{\tau}(t_{n-1}^{l+1})/2, & eoc_n^l \leq \rho - 1 \ \textit{and} \ eoc_n^l \geq \rho + 1 \\ contr(t_{n-1}^{l+1}, \bar{\tau}, \bar{\eta}) & \textit{else}. \end{cases}$$

In the next chapter, we will look at global-in-time approaches.

5

Global-in-time approach

In this chapter, we present how the adaptive time step control, including error estimation based on continuous Galerkin-Petrov methods and the post-processing step, could be applied in a global-in-time approach. Detailed numerical studies on a precise global-in-time solver are beyond the scope of this thesis. We start with a review of a *time-simultaneous multigrid* solver by Dünnebacke et al. for parabolic evolution equations [18] and for non-stationary non-linear PDEs [19]. Based on this, we will shortly discuss a global-in-time approach for the $cGP(2)$-method and parallelization possibilities in the global adaptive approach. This *time-simultaneous multigrid solver* achieves very good results for diffusion-dominated test cases and provides an appropriate approach in the methodology. However, it is not practical for incompressible flow problems. Thus, it can be linked to a *global-in-time velocity solver* as described in [42], which is an extension of a *global-in-time Newton-Multigrid-pressure Schur complement solver* as proposed in [17, 43]. Note that other global-in-time approaches published by Gander and Neumüller [23] or Margenberg and Munch [45] are explicitly not excluded to obtain a global-in-time approach. However, for the sake of brevity, these techniques are not considered in this work.

5.1. A time-simultaneous multigrid method

We now briefly introduce the *time-simultaneous multigrid method* of Dünnebacke et al., [18, 19], which is very close to the *multigrid waveform relaxation* first published in [44]. For a more detailed description of the *time-simultaneous multigrid method* we refer to [18, 19]. This approach assembles a global system consisting of all spatial grid points for all time steps, which is reordered to solve multiple time steps simultaneously for each spatial grid point. This is done by solving many time steps at once by blocking them together. Thus, the global system can be interpreted as a space-only problem. As a result, fewer but larger systems are solved. This leads to an improvement of the scaling behavior of the parallelism in space by reducing the latency costs.

First, discretization in space is done by finite differences or finite elements, resulting in a semi-discrete system for a semi-discrete solution vector containing all discrete spatial

unknowns. Furthermore, a time discretization is performed by a linear one-step or multi-step method. A requirement for the time discretization scheme is that a spatial geometric multigrid method can efficiently solve the associated system in each time step. If we use the $cGP(2)$-method as the time discretization scheme, the block system to be solved in each time step is as in (3.1.9) or especially for the heat equation (3.3.1) and the Navier-Stokes equations (3.4.12). Therefore, this can be written into a global system that must be solved to find the solution vectors, which consist of all discrete spatial unknowns, at each time step. This space-time grid is a Cartesian product of a space and time grid, which has a "space-major" ordering. Since the idea of [18, 19] is to solve many time steps simultaneously, they reorder the space-time grid in a "time-major" ordering. This means that the space and time grid now arranges all time steps belonging to one spatial grid point, then the next spatial grid point with all time steps, and so on. We now interpret all time steps corresponding to a spatial unknown as a vector-valued unknown. The global system and the reordering will be shown in detail later in the case of the $cGP(2)$-method. Since the new system was created by combining and rearranging the equations, the entire system, including the system matrix, the solution vector, and the right-hand-side, naturally has the same solution as a sequential time-stepping scheme.

Next, a geometric multigrid method is applied to this reordered global system by blocking together the time steps corresponding to a spatial unknown. Furthermore, it is interpreted as a single block matrix entry. Then, Dünnebacke et al. use a damped Block-Jacobi method as a smoother in the multigrid method, where the blocks of the Block-Jacobi method refer to the blocks consisting of all time steps corresponding to a spatial grid point. Each block can be solved in parallel in the Block-Jacobi iteration. Furthermore, the block diagonal of the reordered matrix can also be used as a preconditioner in BiCGSTAB (or GMRES) iterations as a smoother, especially for non-linear problems as in [19].

After smoothing, a key component of any multigrid algorithm, the coarse grid correction, is performed. The spatial grid is coarsened in each multigrid level, while the temporal grid remains the same. Thus, the grid transfer in space can be done independently at each time step using the Kronecker product with the usual grid transfer operators and an identity matrix that has the size of the number of all time steps. With this block smoothing operator and adapted transfer operators, the usual multigrid can now be used to solve the global-in-space-time system simultaneously in time and in space, since the smoothing step and the coarse grid correction can be performed in space in parallel over all time steps.

Another aspect to choose is the number of time steps to block. If the number of time steps N is too large, it is not advantageous or practical to perform all time steps in one system. By simply treating K time steps per spatial unknown simultaneously, the next block of K time steps is computed in a sequential manner, using the previous solution as the initial value for the next run. The number of iterations is given by $\frac{N}{K}$. For example, if we have a total of 1000 time steps, we could choose $K = N = 1000$ and run it once, or we could choose $K = 100$ and run it 10 times.

To solve non-linear problems, Dünnebacke et al. presented in [18, 19] an outer non-linear iteration, which is done by fixed-point or Newton iterations, the latter having better convergence results if appropriate initial values are available. The time discretization method has no major impact on the non-linear solver. In both cases, we first linearize

the PDE over the global space-time domain and then solve it simultaneously in time and space using the multigrid method as described above.

5.1.1. Results and implications for the adaptive cGP-time step control

Now, we present an overview of the main results of this *time-simultaneous multigrid method* from [18, 19] and explain why it is suitable for the time step control.

The authors explained that this time-blocked smoother and the space-only restriction operator provide good convergence rates for different time discretizations with linear multistep methods and any time step size. Therefore, it should be applicable to the $cGP(k)$-time-stepping scheme and for different lengths of step sizes that occur in the adaptive approach. Furthermore, the numerical examples showed that the convergence rates are robust to the chosen block size and thus to the number of simultaneous time steps, the grid size, and the time step size.

When the number of cores is small, the computational cost of the *time-simultaneous multigrid method* is slightly higher than that of the sequential time-stepping method, which only exploits parallelization in space. However, at a certain number of cores, sequential time-stepping does not achieve any improvement when more processors are used due to the computational overhead. The *time-simultaneous multigrid method*, on the other hand, increases the scalability of spatial parallelization, so that the computational time becomes lower as the number of cores increases, and especially lower than that of the sequential time-stepping scheme. It performs better than the sequential method for a large number of time steps, which is usually the case for problems that are interesting for adaptive step size selection.

Regarding the block sizes, i.e. the number of simultaneously blocked time steps with the same total number of time steps, the algorithm scales even better with larger block sizes for the investigated test problems on parabolic evolution equations and non-linear PDEs. Even for small block sizes with the same total number of time steps, the time to solution is shorter than for sequential time steps, when more CPUs are used. Again, this shows the advantage of the *time-simultaneous multigrid* method. Therefore, we can suggest as a constraint for the adaptive approach, that the total number of time steps over all adaptive iterations should be at least equal to the number of time steps we would need in a non-adaptive time step scheme with an equidistant minor time step size that is comparable in accuracy.

In addition, Dünnebacke et al. showed that the average number of multigrid iterations is independent of the block size and is the same as for a sequential time step, if the ratio between the time step size and the spatial grid size $\frac{\tau}{h^2}$ is greater than one, i.e. for very small time step sizes. However, for a ratio smaller than one, the multigrid iteration also needs fewer iterations for smaller block sizes than for larger sizes. Thus, the first adaptive iterations should need fewer multigrid iterations than the last ones, since we want to start the adaptive error control with a moderate number of time steps.

Furthermore, for solving non-linear PDEs, the initial guess for the Newton's method might be a problem only in the first iteration. However, we could perform the first adaptive iteration without parallelization with a very small number of time steps. Due to the post-processing step, we obtain good initial values for the Newton iteration in the further adaptive iterations. The initial guess for the further adaptive iterations is no longer a

problem for us, since we can obtain it from the previous adaptive iteration.

5.1.2. Approach with the cGP adaptive error control

After discussing why the time-simultaneous solution strategies can be used in combination with an adaptive time-stepping scheme, we will look at the global space-time $cGP(2)$ system and discuss a parallel approach. We consider the individual components of the adaptive time step control using the continuous Galerkin-Petrov method. We examine the individual components for parallelization options, which are the $cGP(2)$-time-stepping method, the post-processing step, the error estimation, and the adaptive strategy for reconstructing a new time grid.

$cGP(2)$-time-stepping

The block system of the $cGP(2)$-method for a non-linear ODE is shown in (3.1.9). Depending on the type of problem to be solved, the block diagonal of the block matrix changes. For example, considering the heat equation, the scaled Laplacian matrix $L \in \mathbb{R}^{m_h \times m_h}$ is added to the mass matrix $M \in \mathbb{R}^{m_h \times m_h}$ on the blocks of the diagonal, as in (3.3.1), which was given by

$$\underbrace{\begin{pmatrix} M + \frac{\tau_n}{2}L & \frac{1}{4}M \\ -4M & 2M + \frac{\tau_n}{2}L \end{pmatrix}}_{A_n} \underbrace{\begin{pmatrix} U_n^1 \\ U_n^2 \end{pmatrix}}_{U_n} = \begin{pmatrix} R_n^1 \\ R_n^2 \end{pmatrix},$$

where the right hand side vectors read

$$\begin{aligned} R_n^1 &= \underbrace{\frac{\tau_n}{2}(\underline{F}_n^1 + \frac{1}{2}\underline{F}_n^0)}_{b_n^1} - \underbrace{(-\frac{5}{4}M + \frac{\tau_n}{4}L)}_{C_n^1} U_n^0, \\ R_n^2 &= \underbrace{\frac{\tau_n}{2}(\underline{F}_n^2 - \underline{F}_n^0)}_{b_n^2} - \underbrace{(2M - \frac{\tau_n}{2}L)}_{C_n^2} U_n^0 \end{aligned}$$

and $b_n := (b_n^1, b_n^2)^T$. If we write the initial solution at each time step, i.e. for each time interval I_n, with $n = 1, \ldots, N$ in another matrix and add this to the global space-time system, we define the following 2×2 block matrix

$$C_n := \begin{pmatrix} Z & C_n^1 \\ Z & C_n^2 \end{pmatrix} = \begin{pmatrix} Z & -\frac{5}{4}M + \frac{\tau_n}{4}L \\ Z & 2M - \frac{\tau_n}{2}L \end{pmatrix},$$

where Z is a $m_h \times m_h$ zero matrix. Since the solution at the discrete time step t_n is given by $u_\tau(t_n) = U_n^2$, we can write the global space-time system over the entire time interval

$[0,T]$ as

$$\underbrace{\begin{pmatrix} A_1 & & & & \\ C_2 & A_2 & & & \\ & C_3 & A_3 & & \\ & & \ddots & \ddots & \\ & & & C_N & A_N \end{pmatrix}}_{Y} \underbrace{\begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_N \end{pmatrix}}_{U} = \underbrace{\begin{pmatrix} R_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{pmatrix}}_{b}.$$

To solve this system in a time-simultaneous manner, the solution vector $U \in \mathbb{R}^{m_h N \times 1}$ over all time steps and over all spatial unknowns is given by

$$U_n^i = (U_{n,1}^i, U_{n,2}^i, \ldots, U_{n,m_h}^i)^T,$$

is reordered from a "space-major" ordering

$$\begin{aligned} U = (&U_{1,1}^1, U_{1,2}^1, \ldots, U_{1,m_h}^1, U_{1,1}^2, U_{1,2}^2, \ldots, U_{1,m_h}^2, U_{2,1}^1, U_{2,2}^1, \ldots, \\ &U_{2,m_h}^1, U_{2,1}^2, U_{2,2}^2, \ldots, U_{2,m_h}^2, \ldots, U_{N,m_h}^1, U_{N,1}^2, U_{N,2}^2, \ldots, U_{N,m_h}^2)^T, \quad \in \mathbb{R}^{m_h N \times 1} \end{aligned}$$

with $U_{n,l}^i = u(t_{n,i}, x_l)$ for $l = 1, \ldots, m_h$ to a "time-major" ordering

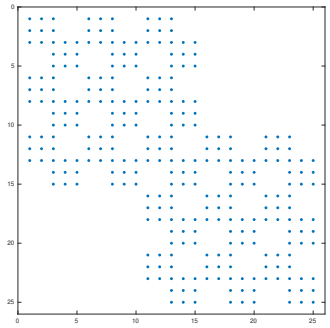
$$\begin{aligned} \tilde{U} = (&U_{1,1}^1, U_{1,1}^2, U_{2,1}^1, U_{2,1}^2 \ldots, U_{N,1}^1, U_{N,1}^2, U_{1,2}^1, U_{1,2}^2, U_{2,2}^1, U_{2,2}^2, \ldots, \\ &U_{N,m_h-1}^2, U_{1,m_h}^1, U_{1,m_h}^2, \ldots, U_{N,m_h}^1, U_{N,m_h}^2)^T \quad \in \mathbb{R}^{m_h N \times 1}. \end{aligned}$$

Therefore, all time steps $t_{n,i}$, $n = 1, \ldots, N$, $i = 1, 2$ corresponding to a spatial grid point are ordered and we can block them as described in [18, 19].

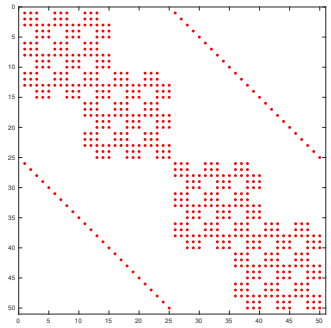
Figure 5.1 shows the sparsity pattern of the discrete Laplacian matrix, the block system of the $cGP(2)$-method at each time step, and the "space-major", as well as the "time-major" global space-time system matrix in the case of the heat equation. For a better illustration, the space discretization was done by 2D biquadratic finite elements with a total of 25 degrees of freedom. Furthermore, the time interval is divided into 5 subintervals. The mass matrix is lumped so that it is only a diagonal matrix. This showed no difference in our studies, but of course in general this means sacrificing some accuracy [56]. In Figures 5.1a and 5.1b, the sparsity pattern of the Laplacian matrix and the 2×2 block system matrix of the $cGP(2)$-method at each time step are displayed. On the main block diagonal, the structure of the Laplacian matrix is highlighted, while the secondary off-diagonals correspond to the mass matrix. This red 2×2 block system of the $cGP(2)$-method is also shown in red in the "space-major" global space-time system for all time steps in Figure 5.1c. The corresponding dependence on the initial solution is shown in blue. The reordered system matrix shown in Figure 5.1d has the same macroscopic structure as the discrete Laplacian matrix shown in Figure 5.1a. Even if the same structure is not sufficient for a good convergence behavior of multigrid methods, the system should at least be solvable by a multigrid method in space.

Post-processing step and error estimations

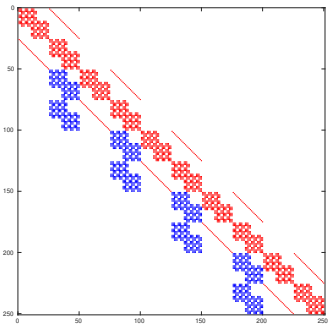
After solving the global space-time block system of the $cGP(2)$-method in a space-parallel and time-simultaneous manner, we solve the post-processing system (3.2.2), which



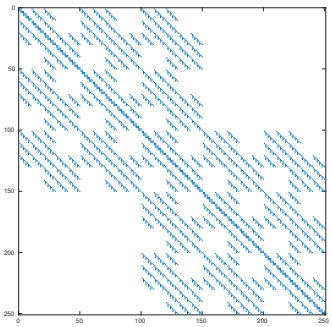
(a) Laplacian matrix.



(b) $cGP(2)$-block system.



(c) "Space-major" global space-time system.



(d) "Time-major" global space-time system.

Figure 5.1: Structure of matrices with Q_2 finite elements with 25 total degrees of freedom. The global systems consist of 5 time intervals.

is a system consisting only of the mass matrix per time step. Since these systems are completely independent, we can perform this in parallel. Furthermore, the post-processed system does not have to be solved in every time step, because the post-processed solutions satisfy the continuity properties at t_n. However, due to inaccuracies, it is more feasible to do it in every time step, since the error estimator and adaptive strategy are based on it. Moreover, as mentioned above, the computational cost is negligible, as will be shown briefly in the numerical studies in Chapter 6.

Overall, we obtain the $cGP\text{-}C1(3)$ solutions independently by the post-processing step, we can easily perform the error estimation in parallel per time step.

Adaptive strategy for reconstructing a new time grid

Finally, the adaptive strategy, including the reconstruction of a new time grid, is considered. Since the new time grid is constructed sequentially, it can be parallelized by dividing the old time grid into subgrids. Then, the new time grid can be constructed sequentially by applying the adaptive strategy to each subinterval independently and in parallel. However, the error estimator for a time step is a scalar value, the reconstruction of the new time grid is not expensive.

5.2. Global-in-time Pressure Schur complement solvers

Starting with the *time-simultaneous multigrid method* presented above, we now explore the topic further for incompressible flow problems. Therefore, we briefly summarize the *augmented Lagrangian acceleration of a global-in-time pressure Schur complement solver* by Lohmann and Turek [42] based on [17]. They introduced a *global-in-time Newton-Multigrid-pressure Schur complement solver* in [43]. They studied these approaches for incompressible flow problems, which fits into the context of the global adaptive approach. We refer to the above works for detailed information.

For the incompressible Navier-Stokes equations, they linearize the non-linear system using Picard iteration or Newton's method. Then they discretize the linear system first in space and then in time. In addition, the global system is set up by considering all time steps and then blocking subproblems as discussed in the previous section. The result is a blocked saddle-point problem. To solve this blocked system more efficiently, they did a pressure Schur complement iteration to have a global linear system for the pressure, which is the only unknown due to the fact that the velocity unknowns are eliminated. This global-in-time iteration solves this system for all time steps simultaneously by decomposing it into independent subproblems. Thus, each global system can be interpreted as a space-only problem. Of course, they take advantage of common iterative solution techniques such as efficient parallel preconditioners. Since they showed that the convergence behavior for the basic Schur complement iteration is quite slow, they applied an augmented Lagrangian methodology, where simple algebraic manipulations are used to stabilize. This can easily be adopted for a global-in-time approach.

In each pressure Schur complement iteration, an auxiliary velocity field must be computed. For this purpose, a global-in-time velocity solver has to be applied. Therefore, a highly specialized multigrid solver [44] was combined with a *multigrid waveform relaxation* approach [55]. Thus, in each step of the *pressure Schur complement solver*, a global-in-time velocity auxiliary problem, which is a linear system of equations, has to be solved. This system contains velocity solutions for all time steps and all spatial grid points. This can be interpreted as a space-only problem, which can be solved by a space-only multigrid method. If a Jacobi smoother is chosen in the multigrid method, this coincides with the *time-simultaneous and space-parallel multigrid method* of Dünnebacke [18]. Furthermore, another specialized multigrid solver was established, which is necessarily due to the augmented Lagrangian acceleration [42].

5.2.1. Results and implications for the adaptive *cGP*-time step control

They studied several test cases for small to moderate Reynolds numbers, such as the Carreau-Yasuda model. They showed that a fast convergence rate can be successfully achieved. Therefore, this presented solver can be implemented to solve the global adaptive time step control effectively. For convection dominated problems, such as the well-known flow around a cylinder benchmark for a Reynolds number of 100, see Section 2.1.3 and [52], the same performance cannot be achieved. While the augmented Lagrangian stabilization improves the convergence behavior, the total number of multigrid iterations for the global-in-time velocity problem increases for convection-dominated regions and for a higher number of blocked time step sizes. In general, this is of course an efficiency

issue, but on the other hand it could be an argument for adaptive time step control. With an adaptive global-in-time approach, the number of blocked time steps decreases, since the main goal is to use as few time steps as necessary to build a time grid. However, it is not verified whether fewer time steps with larger time step sizes lead to an improvement in efficiency. It could also be that the physical time horizon is crucial. Lohmann and Turek point to future studies on other possible efficient solvers in combination with an *augmented Lagrangian accelerator of pressure Schur complement specialized multigrid solver*.

5.2.2. Approach with the cGP adaptive error control

In the case of the Navier-Stokes equations, the 6×6 block system of the $cGP(2)$-method must be solved in each adaptive step. Since this is a saddle-point problem, which includes the initial pressure on the right-hand-side, we need a higher order pressure approximation as the initial pressure solution at each time step to solve the $cGP(2)$ block system (3.4.13). We can either use the higher order post-processed pressure solution for each current time step, defined in (3.4.15), from the last adaptive iteration, or we solve once the post-processing step and receive due to the continuity property of the pressure $\tilde{p}_\tau$ for each current time step the initial higher order pressure solution, defined in 3.4.10, like the velocity solution. Another possibility is to set up a global space-time system that includes the $cGP(2)$-system, the post-processing, and the resulting higher order initial pressure value at each time step. This global system is at least larger and especially worse structured. The first two options result in an appropriate global system that should be solved efficiently such that these are preferable. In addition, the solution of the post-processing system is independent and of very low computational cost, which we will show briefly in the next chapter.

In summary, we have shown in this section that the time-stepping control including error estimation based on continuous Galerkin-Petrov methods and its post-processing step can be performed simultaneously or in parallel due to the global-in-time approach.

6

Numerical studies

In this chapter, we perform several numerical tests to analyze the $cGP(2)$- and $cGP\text{-}C1(3)$-time-stepping scheme and the error estimators with application to a global adaptive time-stepping approach. We study the heat equation, the Navier-Stokes equations, including a flow around cylinder benchmark for Newtonian and non-Newtonian fluids in the two-dimensional domain. As a spatial discretization, we use the previously introduced spatial finite element discretization Q_2 and in the case of the Navier-Stokes equations, we use the finite element pair (Q_2, P_1^{disc}), see Section 2.3.

Here, we focus only on the time discretization error. Therefore, in all numerical examples, we have chosen a spatial grid size that is sufficiently small that the spatial discretization error is smaller than the time discretization error. To analyze the accuracy of the time discretization methods, we measure the discretization error at the discrete time points t_n in the discrete L^∞-norm of a function $v : I \to L^2(\Omega)$, which is defined as

$$||v||_\infty := \max_{1\leq n\leq N} ||v^-(t_n)||_{L^2(\Omega)}, \quad v^-(t_n) := \lim_{t\to t_n-0} v(t), \quad \forall v : I \to L^2(\Omega).$$

We introduce a shorter notation for the L^2-norm by

$$||\cdot||_2 = ||\cdot||_{L^2(I,L^2(\Omega))}.$$

To approximate the L^2-norm, we use the 5-point Gauss-Lobatto formula with the reference points $t_{n,i}^{GL(5)}$, $i = 0,\ldots,4$

$$||v||_{L^2(I_n,L^2(\Omega))} = \left(\int_{I_n} ||v(t)||^2_{L^2(\Omega)} dt\right)^{\frac{1}{2}} \approx \left(\frac{\tau_n}{2}\sum_{i=0}^{4} \hat{\omega}_i ||(v(t_{n,i}^{GL(5)})||^2_{L^2(\Omega)}\right)^{\frac{1}{2}}.$$

Since the L^2-norm is approximated by the 5-point Gauss-Lobatto formula with $t_{n,i}^{GL(5)}$, $i = 0,\ldots,4$, we define the function $v_i : I \to L^2(\Omega)$ to the i-th Gauss-Lobatto point by

$$v_i := v(t_{n,i}^{GL(5)}),$$

such that we can evaluate the discrete L^∞-norm for the i-th quadrature points $t_{n,i}^{GL(5)}$. We denote the discretization error of the $cGP(2)$-method by $e := u - u_{\tau,h}$ and the discretization error of the $cGP\text{-}C1(3)$-method by $\tilde{e} := u - \tilde{u}_{\tau,h}$ for the exact solution $u(t)$. Analogous to the error estimator (4.1.1), we define the errors e_n and $\tilde{e}_n$ by the L^2-norm on I_n by

$$e_n = \frac{1}{\sqrt{\tau_n}} ||u - u_{h,\tau}||_{L^2(I_n,L^2(\Omega))}, \qquad \tilde{e}_n = \frac{1}{\sqrt{\tau_n}} ||u - \tilde{u}_{h,\tau}||_{L^2(I_n,L^2(\Omega))}.$$

In case of the Navier-Stokes equations, we use the associated vector-valued norms on $(L^2(\Omega))^2$. All numerical tests in this work have been implemented in the FEAT3 software.

6.1. The heat equation

We start our numerical studies by considering the heat equation (2.1.1) on the domain $\Omega = (0,1)^2$. We choose sequences of structured meshes, which are set up by a uniform refinement. To study the accuracy of the time discretization schemes, as well as the resulting error estimator in an adaptive time step control, we consider two examples with different given exact solutions $u(x,y,t)$ for the heat equation.

6.1.1. Exponential-in-time problem

First, we consider the problem (2.1.1) on the time interval $I = [0,1]$ for the following analytic solution

$$u(x,y,t) = x(1-x)y(1-y)e^t,$$

with the associated data f, u_0 and homogeneous Dirichlet boundary conditions.

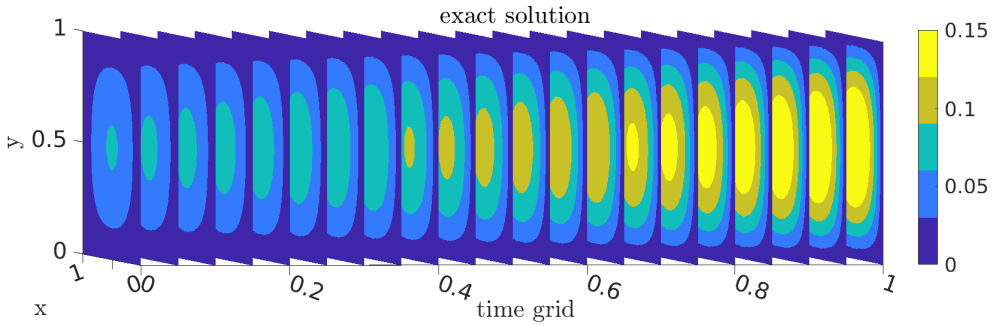


Figure 6.1: Illustration of $u(x,y,t)$ on the time grid $[0,1]$ and on $\Omega = (0,1)^2$.

In Figure 6.1 we illustrate the exact solution. The time grid is shown on the x-axes, while the spatial grid $\Omega = (0,1)^2$ is shown on the y- and z-axes of this plot. The solution values lie between $[0, 0.15]$, as we can see in the legend on the right. At the beginning of the time interval the solution is smaller and gets larger over time until the final time is reached. The solution does not vary much, so this is a good first test problem.

For all tests concerning this problem, the spatial mesh is fixed with a step size of $h = 2^{-8}$ and the equidistant time step size is $\tau = 1/N$.

	$cGP(2)$				$cGP\text{-}C1(3)$			
$\frac{1}{\tau}$	$\|\|e\|\|_\infty$	EOC	$\|\|e\|\|_2$	EOC	$\|\|\tilde{e}\|\|_\infty$	EOC	$\|\|\tilde{e}\|\|_2$	EOC
4	5.34e-07		5.36e-06		5.34e-07		2.56e-07	
8	3.60e-08	3.89	6.70e-07	3.00	3.60e-08	3.89	1.49e-08	4.11
16	2.29e-09	3.97	8.37e-08	3.00	2.29e-09	3.97	9.02e-10	4.04
32	1.44e-10	3.99	1.05e-08	3.00	1.44e-10	3.99	5.59e-11	4.01
64	9.01e-12	4.00	1.31e-09	3.00	9.01e-12	4.00	3.48e-12	4.00
128	5.64e-13	4.00	1.63e-10	3.00	5.64e-13	4.00	2.18e-13	4.00
256	3.86e-14	3.87	2.04e-11	3.00	3.86e-14	3.87	2.21e-14	3.30

Table 6.1: Discrete L^∞-errors and L^2-errors.

In Table 6.1 the errors for the $cGP(2)$-method and the $cGP\text{-}C1(3)$-method are measured in the discrete L^∞-norm as well as in the L^2-norm. Moreover, the corresponding experimental orders of convergence (4.4.1) are shown. The discrete L^∞-errors are the same for both methods, which is obvious since they have the same solution at discrete time points t_n. Here, both methods converge with order four, while the experimental order of convergence of the L^2-error of the $cGP(2)$-method is three over the whole time interval and of the $cGP\text{-}C1(3)$-method is four over the whole time interval.

	$cGP(2)$					
$\frac{1}{\tau}$	$\|\|e_1^{GL(5)}\|\|_\infty$	EOC	$\|\|e_2^{GL(5)}\|\|_\infty$	EOC	$\|\|e_3^{GL(5)}\|\|_\infty$	EOC
4	9.54e-06		2.65e-07		1.00e-05	
8	1.28e-06	2.90	1.89e-08	3.81	1.31e-06	2.93
16	1.66e-07	2.95	1.24e-09	3.93	1.68e-07	2.97
32	2.11e-08	2.97	7.94e-11	3.97	2.13e-08	2.98
64	2.67e-09	2.99	5.01e-12	3.99	2.68e-09	2.99
128	3.35e-10	2.99	3.14e-13	4.00	3.36e-10	3.00
256	4.20e-11	3.00	1.97e-14	3.99	4.20e-11	3.00
	$cGP\text{-}C1(3)$					
	$\|\|\tilde{e}_1^{GL(5)}\|\|_\infty$	EOC	$\|\|\tilde{e}_2^{GL(5)}\|\|_\infty$	EOC	$\|\|\tilde{e}_3^{GL(5)}\|\|_\infty$	EOC
4	7.68e-08		2.65e-07		5.19e-07	
8	7.61e-09	3.34	1.89e-08	3.81	2.65e-08	4.29
16	7.85e-10	3.28	1.24e-09	3.93	1.41e-09	4.23
32	5.97e-11	3.72	7.94e-11	3.97	7.97e-11	4.15
64	4.07e-12	3.87	5.01e-12	3.99	4.70e-12	4.08
128	2.66e-13	3.94	3.14e-13	4.00	2.86e-13	4.04
256	4.86e-14	2.45	1.97e-14	3.99	4.88e-14	2.55

Table 6.2: Discrete L^∞-errors at time points $t_{n,i}^{GL(5)}$, $i = 1,\dots,3$.

The quadrature points in the $cGP(2)$-method are chosen as 3-point Gauss-Lobatto points. Therefore, the solutions of the $cGP(2)$-method and the $cGP\text{-}C1(3)$-method coincide at these points. Thus, we look at two more intermediate points of the 5-point Gauss-Lobatto formula, denoted by $t_{n,1}^{GL(5)}$ and $t_{n,3}^{GL(5)}$. The other three points correspond

to the 3-points of the Gauss-Lobatto formula. Table 6.2 shows the errors in the discrete L^∞-norm. The experimental order of convergence is three for the intermediate points $t_{n,1}^{GL(5)}$ and $t_{n,3}^{GL(5)}$ and at the point $t_{n,2}^{GL(5)}$ it is four in the case of the $cGP(2)$-method. The cGP-$C1(3)$-method converges with order four for every time point.

6.1.2. Sinus-problem

Since we want to study an adaptive time step control, we now consider a more oscillatory example on the time interval $I = [0,T]$ with $T = 10$, which has the following analytical solution:

$$u(x,y,t) = x(1-x)y(1-y)\sin(\pi(9-(t-3)^2)\sin(\pi t/6)),$$

with the associated data f, u_0 and homogeneous Dirichlet boundary conditions.

The Figure 6.2 shows the exact solution $u(x,y,t)$ at different times, where the time interval is divided into subintervals. In contrast to the previous test problem, the solution is oscillating.

For all tests concerning this problem, the spatial mesh is fixed with a step size of $h = 2^{-6}$ and the equidistant time step size is $\tau = T/N$ with $T = 10$.

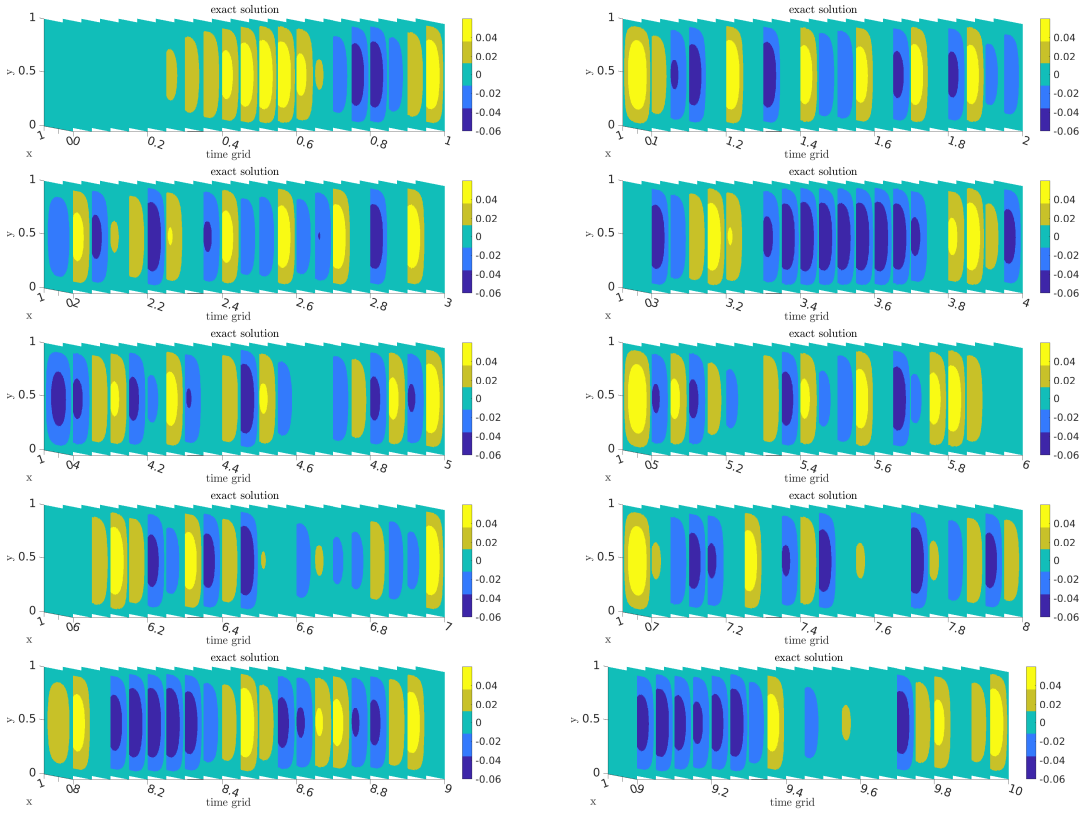


Figure 6.2: Solution $u(x,y,t)$ on the time grid $[0,10]$ and on $\Omega = (0,1)^2$.

	$cGP(2)$				$cGP\text{-}C1(3)$			
$\frac{1}{\tau}$	$\|\|e\|\|_\infty$	EOC	$\|\|e\|\|_2$	EOC	$\|\|\tilde{e}\|\|_\infty$	EOC	$\|\|\tilde{e}\|\|_2$	EOC
64	2.17e-01	1.34	1.25e-01	1.59	2.17e-01	1.34	1.24e-01	1.60
128	2.31e-03	6.55	6.08e-03	4.36	2.31e-03	6.55	4.34e-03	4.84
256	1.06e-04	4.44	7.25e-04	3.07	1.06e-04	4.44	2.80e-04	3.95
512	6.24e-06	4.09	9.07e-05	3.00	6.24e-06	4.09	1.75e-05	4.00
1024	3.84e-07	4.02	1.13e-05	3.00	3.84e-07	4.02	1.09e-06	4.00
2048	2.39e-08	4.01	1.42e-06	3.00	2.39e-08	4.01	6.83e-08	4.00
4096	1.49e-09	4.00	1.77e-07	3.00	1.49e-09	4.00	4.27e-09	4.00
8192	9.32e-11	4.00	2.22e-08	3.00	9.32e-11	4.00	2.67e-10	4.00

Table 6.3: Discrete L^∞-errors and L^2-errors.

	$cGP(2)$					
$\frac{1}{\tau}$	$\|\|e_1^{GL(5)}\|\|_\infty$	EOC	$\|\|e_2^{GL(5)}\|\|_\infty$	EOC	$\|\|e_3^{GL(5)}\|\|_\infty$	EOC
64	2.21e-01	1.35	1.69e-01	1.70	1.80e-01	1.75
128	7.65e-03	4.85	9.64e-03	4.13	7.20e-03	4.64
256	1.14e-03	2.74	6.10e-04	3.98	1.17e-03	2.63
512	1.53e-04	2.90	3.85e-05	3.99	1.55e-04	2.91
1024	1.95e-05	2.97	2.41e-06	4.00	1.97e-05	2.98
2048	2.46e-06	2.99	1.51e-07	4.00	2.47e-06	3.00
4096	3.08e-07	3.00	9.44e-09	4.00	3.08e-07	3.00
8192	3.85e-08	3.00	5.90e-10	4.00	3.85e-08	3.00
	$cGP\text{-}C1(3)$					
	$\|\|\tilde{e}_1^{GL(5)}\|\|_\infty$	EOC	$\|\|\tilde{e}_2^{GL(5)}\|\|_\infty$	EOC	$\|\|\tilde{e}_3^{GL(5)}\|\|_\infty$	EOC
64	2.01e-01	1.48	1.69e-01	1.70	2.01e-01	1.59
128	3.76e-03	5.74	9.64e-03	4.13	3.04e-03	6.05
256	1.72e-04	4.45	6.10e-04	3.98	1.40e-04	4.44
512	9.48e-06	4.18	3.85e-05	3.99	8.36e-06	4.07
1024	5.62e-07	4.08	2.41e-06	4.00	5.27e-07	3.99
2048	3.44e-08	4.03	1.51e-07	4.00	3.32e-08	3.99
4096	2.13e-09	4.01	9.44e-09	4.00	2.09e-09	3.99
8192	1.32e-10	4.01	5.90e-10	4.00	1.31e-10	3.99

Table 6.4: Discrete L^∞-errors at time points $t_{n,i}^{GL(5)}$, $i = 1,\dots,3$.

Accuracy of the $cGP(2)$- and $cGP\text{-}C1(3)$-methods

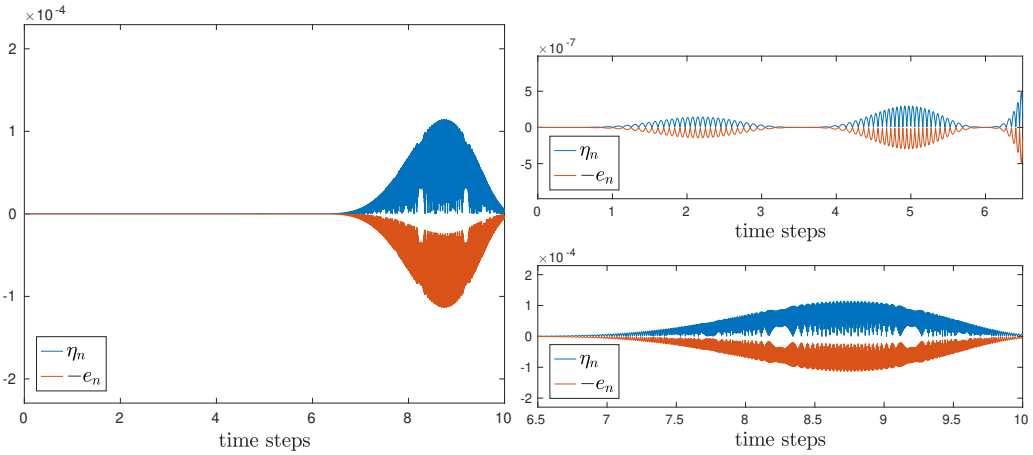
Table 6.3 shows the error in the discrete L^∞-norm, as well as in the L^2-norm of the $cGP(2)$- and $cGP\text{-}C1(3)$-methods. The experimental order of convergence is as expected. Only for larger time step sizes, we see that in the L^2-norm the error of $cGP\text{-}C1(3)$-method is not significantly smaller than the error of the $cGP(2)$-method, since the post-processing step requires an accurate $cGP(2)$ solution.

The discrete L^∞-error for this oscillating test problem is displayed in Table 6.4. The experimental order of convergence is three at the intermediate times $t_{n,1}^{GL(5)}$ and $t_{n,3}^{GL(5)}$

for the $cGP(2)$-method. At time $t_{n,2}^{GL(5)}$, the experimental order of convergence is four. Furthermore, the $cGP\text{-}C1(3)$-method converges with order four for every point. Thus, the discrete L^∞-errors for the $cGP\text{-}C1(3)$-method are smaller than for the $cGP(2)$-method. We have validated that the $cGP\text{-}C1(3)$-method can be used as a higher order approximation for the $cGP(2)$-method and thus for our error estimator.

In Figure 6.3 the error estimates (4.1.1) are plotted in blue and the negatively scaled exact errors are displayed in red with a minus sign in the legend. This is done, for a better comparison with the error estimators. Furthermore an equidistant time step size $\tau^1 = 1/512$ is used. With regard to subsequent adaptive studies, it should be noted at this point that τ^1 is always an equidistant time step size in the first adaptive iteration.

In Figure 6.3a, the entire time interval is considered. The error estimates approximate the errors quite well. We can also see that the error estimates show quite well the oscillating behavior of the solution at the end of the time interval. The time interval is divided into two figures to better illustrate the smaller errors at the beginning of the time interval in Figure 6.3b. The upper figure shows that the error estimates and the exact errors are less than $5 \cdot 10^{-7}$, while the lower figure illustrates errors that are less than $2 \cdot 10^{-4}$. For each time point in the time interval, the error estimates seem to approximate the exact errors well. Therefore, in the next section, we consider the global adaptive time step control strategies based on the error estimator.



(a) $I = [0, 10]$. (b) $I = [0, 6.5)$ and $I = [6.5, 10]$.

Figure 6.3: Error estimates versus exact errors for $\tau^1 = 1/512$.

The first studies of the global adaptive approach including the error estimator (4.1.1) are carried out. We investigate several adaptive strategies in combination with different controllers. We then look at the effects of the initial time step size and different tolerances. Finally, the adaptive time step control is compared to an equidistant time step run.

Comparison of adaptive time step control strategies

Figure 6.4 shows the error estimates and the exact errors for different adaptive strategies such as the *DH* controller, the *EL* controller, the *PC11* controller and the *PID* controller, see, Controllers 4.1-4.4, by using the *piecewise constant interpolation strategy*, the *linear interpolation strategy* and the *piecewise cubic Hermite interpolation strategy*, which can

be developed like the *linear interpolation strategy*. In the *DH* controller, Controller 4.1, the limits of refining and coarsening were set exceptionally to $K_1 = 1$ and $K_2 = 8/90$. For all other controllers in this thesis we used the values $K_1 = 0.75$ and $K_2 = 0.01$. The initial equidistant time step size used in the first iteration is 0.25 and the tolerance was set to 10^{-4}. The number of time steps in the final L-th adaptive iteration, that satisfy the stopping criterion, is denoted by N^L.

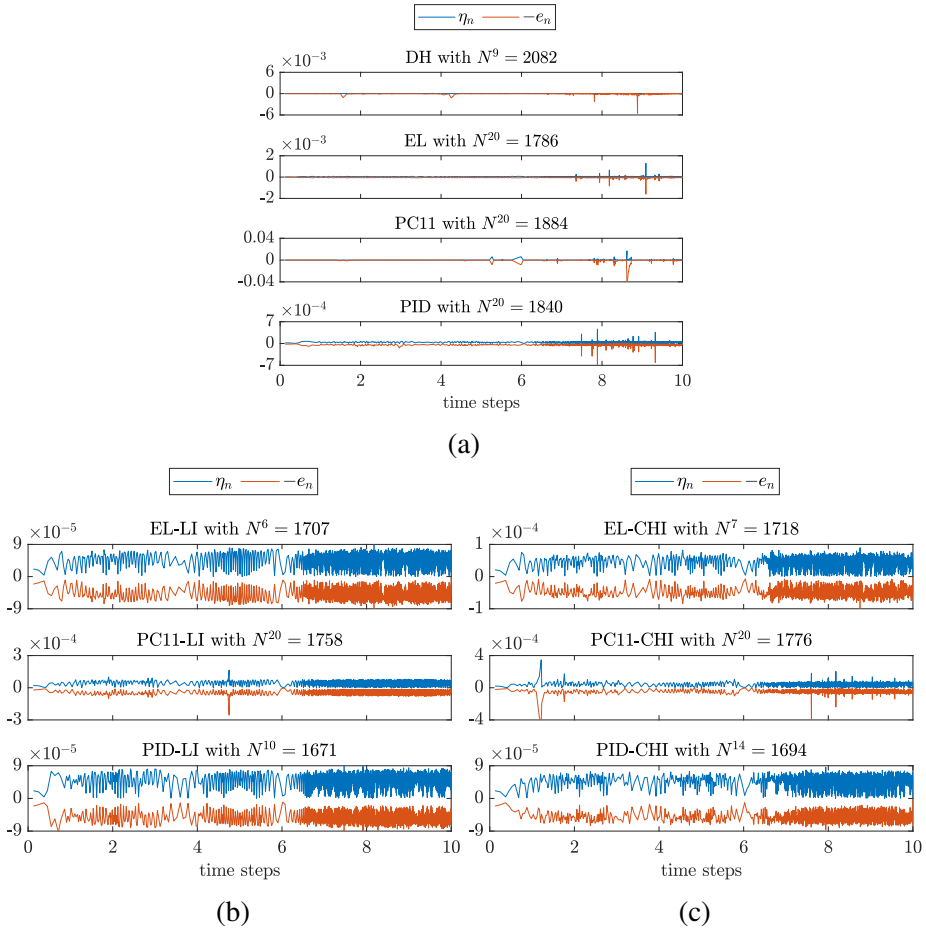


Figure 6.4: Comparison of error estimators versus exact errors for different adaptive strategies with a tolerance of 10^{-4} and $\tau^1 = 0.25$.

Figure 6.4a shows the error estimates for the adaptive strategies with *piecewise constant interpolation* and the exact errors. In none of these strategies, the error estimates and the exact errors are smaller than the given tolerance. Three strategies do not satisfy the stopping criterion and reach the maximum number of adaptive iterations, which was set to 20. The doubling and halving strategy *DH* stops after nine iterations, but we see that the exact errors are larger than the error estimates at some points.

Figures 6.4b and 6.4c illustrate the error estimates and the exact errors for the adaptive strategies with linear *-LI* and *piecewise cubic Hermite interpolation -CHI*. We see immediately, that the interpolation has a positive effect on the number of iterations and the number of time steps. An exception is the *PC11* controller, which does not satisfy the stopping criterion after 20 iterations in both cases of interpolation. The *EL* controller satisfies the stopping criterion after 6 and 7 adaptive iterations and the *PID* controller at

iteration 10 and 14. The error estimators approximate the exact errors very well. Due to the global-in-time approach, the small difference in the number of time steps is not so important compared to the number of adaptive iterations used. Therefore, we can assume that other controller variants do not necessarily need to be tested.

	DH	EL	PC11	PID	EL-LI	PC11-LI	PID-LI	EL-CHI	PC11-CHI	PID-CHI
$TOL = 1e-3$										
iterations	7	20	20	20	10	15	13	11	20	16
$\max_n \eta_n$	9e-04	4e-03	1e-01	7e-03	9e-04	9e-04	9e-04	1e-03	4e-01	1e-03
$\max_n e_n$	6e-01	6e-01	4e-01	3e-01	9e-04	1e-03	9e-04	1e-03	5e-01	9e-04
$TOL = 1e-4$										
iterations	9	20	20	20	6	20	10	7	20	14
$\max_n \eta_n$	7e-05	1e-03	2e-02	5e-04	9e-05	2e-04	9e-05	9e-05	3e-04	9e-05
$\max_n e_n$	6e-03	2e-03	4e-02	7e-04	9e-05	3e-04	9e-05	1e-04	4e-04	9e-05
$TOL = 1e-5$										
iterations	10	20	20	20	4	20	11	6	20	13
$\max_n \eta_n$	7e-06	1e-04	1e-01	9e-05	1e-05	2e-05	9e-06	9e-06	5e-03	9e-06
$\max_n e_n$	3e-04	2e-04	1e-01	1e-04	1e-05	3e-05	9e-06	1e-05	3e-02	1e-05

Table 6.5: Comparison of adaptive iterations between the use of an equidistant time step size and our adaptive error control strategy *EL-LI*.

Furthermore, we verify our observations for other tolerances in Table 6.5. As in the previous figure, we see that for the *piecewise constant interpolation strategy* in column two to five, no error is less than the required tolerance. For all other strategies that use an interpolation technique, except of the *PC11*, the maximum error or error estimator achieves the required tolerance. In addition, the *EL-LI* controller needs the least number of iterations for all tolerances. Therefore, the elementary local error controller with interpolation can be considered as the best version. Comparing the two interpolation strategies, we see that for the linear case, fewer time steps are needed. However, this may depend on the underlying problem to be solved. In the end, the elementary local error controller *EL-LI*, Controller 4.2, with *linear interpolation* is the best choice for this test problem.

To explain why the *EL-LI* controller is better, we can consider the reasons for modifying the time step size for three best controller types. In the Figure 6.5, the three modification cases “unmodified”, “reduced” and “coarsened” represent whether the interpolated time step size at this time point was unmodified, shown in green, reduced in red or coarsened in blue compared to the step size used in the previous iteration.

Thus, after solving the first iteration, the new time point for the second iteration is built from the cases of iteration one. At the beginning, i.e. during the first and second iteration, we see that the *EL* controller reduces the time step sizes more than the other controllers, since we see more red points. For a higher number of adaptive iterations, the number of time steps to be reduced, especially at the end of the time interval, is higher for the *PID-LI* controller than for the *EL-LI* and *EL-CHI* controllers. Furthermore, there is no coarsening

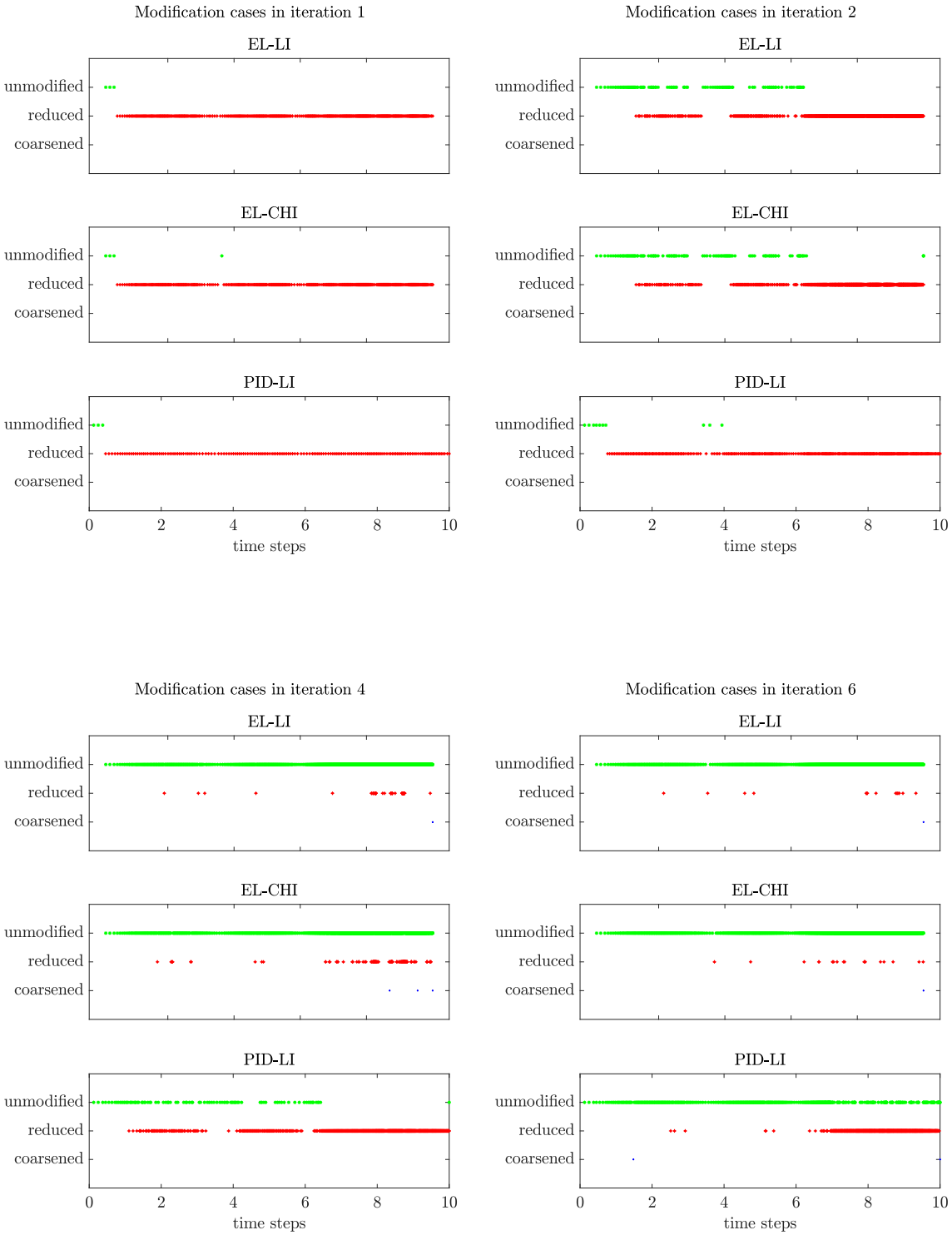


Figure 6.5: Reasons for modification of adaptive strategies with a tolerance of 10^{-4} and $\tau^1 = 0.25$.

of the time steps, since the initial equidistant time step size of 0.25 is already large enough. To see that our best strategy *EL-LI* performs coarsening, we need to start with a smaller initial time step size, which we will investigate in the next section. Therefore, from now on we will only use the strategy with the *EL-LI* controller, Controller 4.2.

Initial time step size

Using different equidistant initial time step sizes in the first iteration leads to different reasons for modifying the time step size, as shown in Figure 6.6. We see for 20, 40 and 80 initial time steps, that there is also no coarsening of time step sizes. For 320 and 640 initial time steps, the time step sizes are coarsened mainly at the beginning and in the middle of the time interval. However, all three initial step sizes lead to satisfying the stopping criterion after at least seven adaptive iterations, which is written in the legend of Figure 6.7, where the corresponding time step sizes after L adaptive iterations by using different equidistant initial time step sizes is illustrated.

Additionally, we can see in Figure 6.7 that the smaller the initial time step size is, the more time steps are needed in the final adaptive iteration, whereby they have almost the same number of time steps. Furthermore, we observe that the length of the time steps now covers the dynamics of the solution well, such that larger time step sizes are used at the beginning of the time interval and smaller time step sizes are used at the end of the time

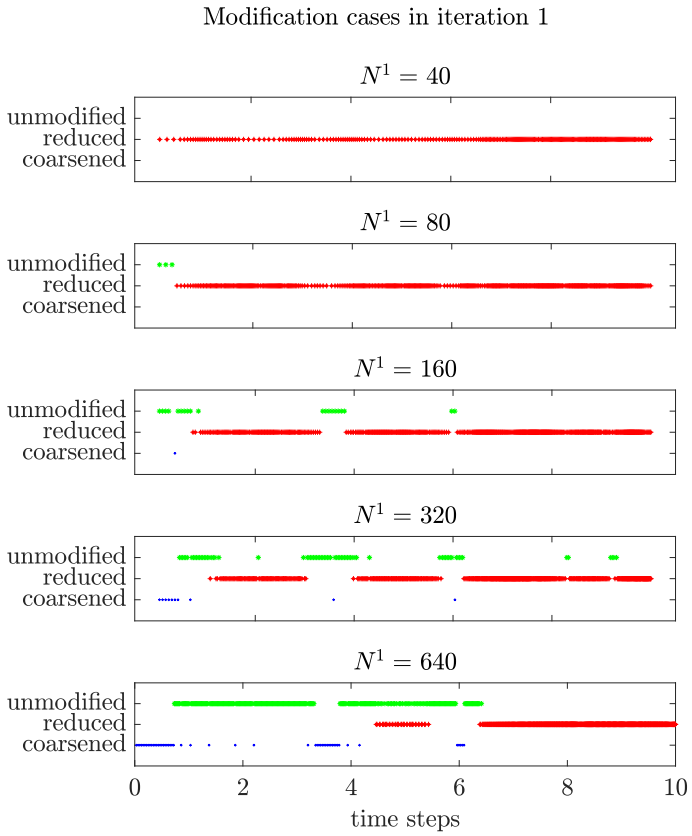


Figure 6.6: Reasons for modification for different initial time step sizes after the first iteration for the *EL-LI* controller with a tolerance of 10^{-4}.

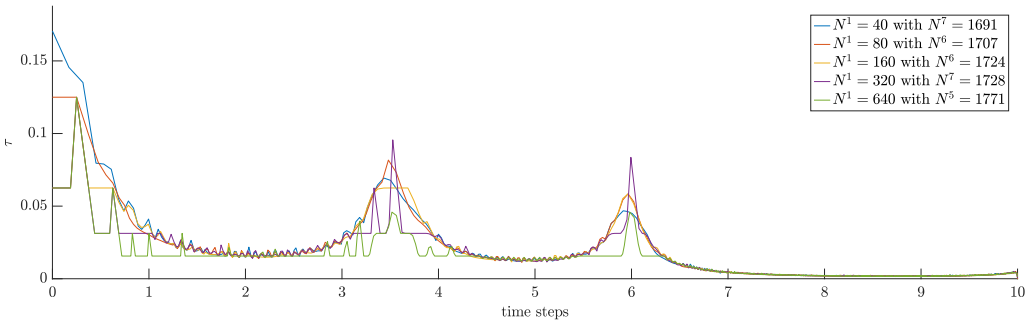
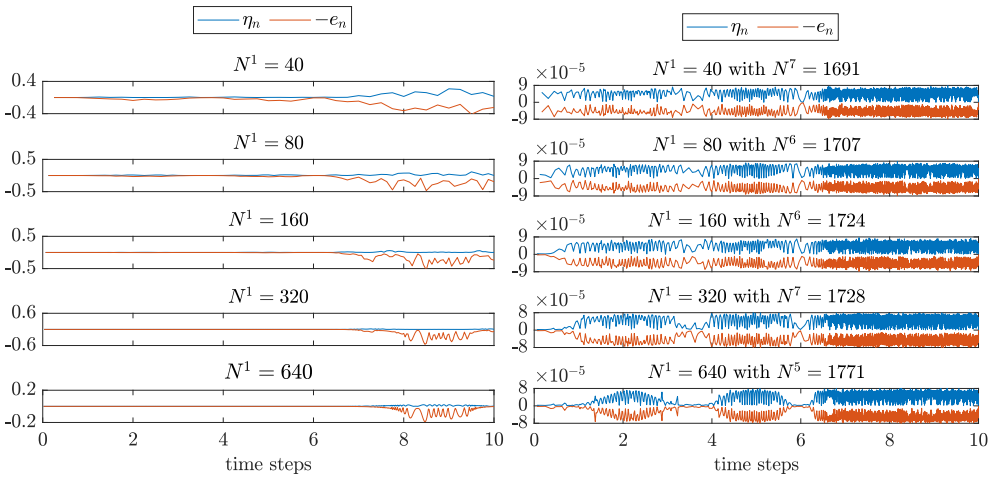


Figure 6.7: Time step sizes for different initial time step sizes for the *EL-LI* controller with a tolerance of 10^{-4}.

interval. The smallest number of adaptive iterations is $L = 5$ for 640 initial time steps, but this also has the highest number of time steps. Furthermore, the time step sizes increase and decrease more often than for the other initial time steps. Before making a statement about the best choice of initial time step size, we first look at the resulting error estimates and exact errors, as well as the computational time.



(a) Adaptive iteration 1.

(b) Adaptive iteration L.

Figure 6.8: Error estimates versus exact errors for different initial time step sizes for the *EL-LI* controller with a tolerance of 10^{-4}.

The resulting error estimates and exact errors are shown for different initial time step sizes in iteration one in Figure 6.8a and in the iteration L in Figure 6.8b, where the stopping criterion is fulfilled. In the first iteration, the error estimates do not approximate the exact errors well. Nevertheless, the adaptive strategy leads to a good performance, see Figure 6.8a, since the error estimates and the exact errors are smaller than the required tolerance for each initial time step size. In addition, all initial time steps lead to good adaptive error control and all error estimates approximate the exact errors sufficiently accurately. By using larger initial time step sizes, the errors are much more equilibrated than for smaller initial time step sizes. Furthermore, a smaller initial time step size does not necessarily lead to a smaller number of adaptive iterations.

Finally, in Table 6.6, we consider the total number of time steps, defined as $\bar{N}$, over all adaptive iterations and their required computing time for different initial time step sizes. The computing time was measured for the entire adaptive approach, including the solution of the $cGP(2)$ system and the post-processing step, as well as the adaptive strategy to construct a new time grid and thus over all adaptive iterations. A simulation with $N^1 = 80$ leads to the smallest number of time steps overall and is also the fastest. This is followed by $N^1 = 160$ and $N^1 = 640$. For this test problem, the best number of initial time steps is 80.

N^1	$\bar{N}$	CPU time
40	8,218	1,618
80	6,988	1,369
160	7,308	1,434
320	9,342	1,826
640	7,391	1,588

Table 6.6: Comparison of computing time (in seconds) by using different initial time step sizes.

In general we have to take care that the initial time step size is not too large, since the resulting error estimates cannot approximate the exact errors. Checking the experimental order of convergence can be a good criterion for investigating whether an initial time step size is suitable or not. This will be studied in the case of the flow around a cylinder benchmark. In addition, using a small initial time step size will result in a higher number of time steps, and further, the time step sizes will increase and decrease more. Obviously, the choice of the initial time step size also depends on the required tolerance.

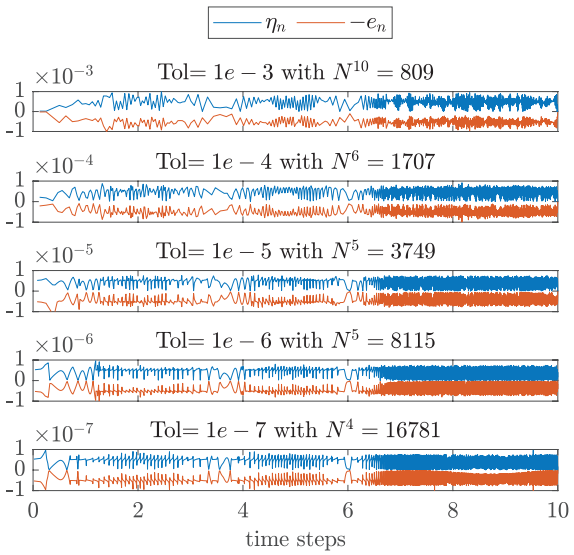
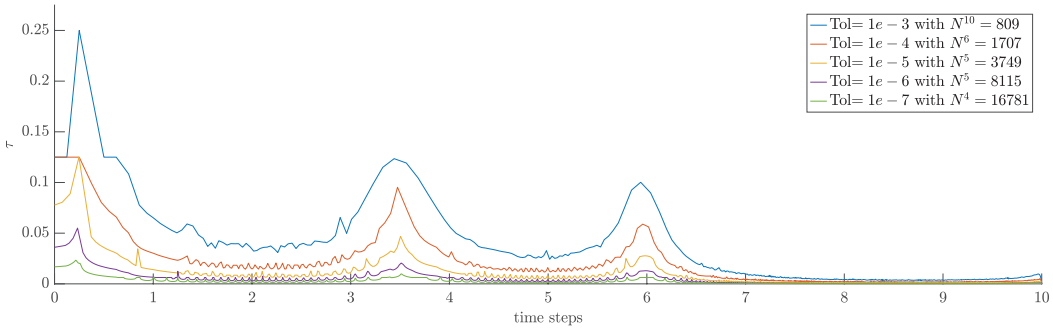


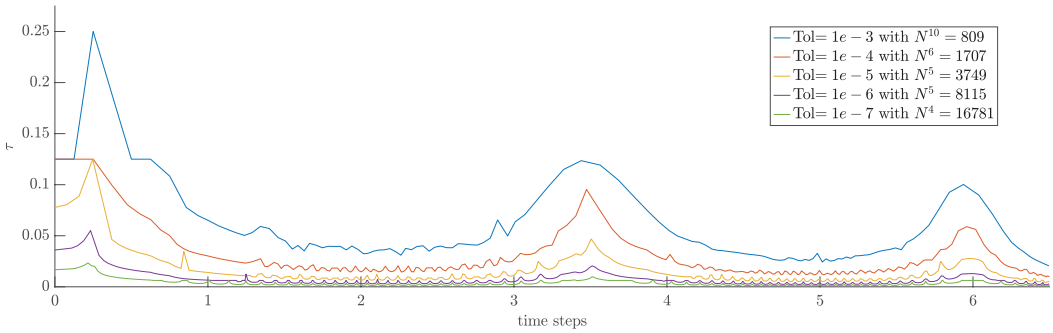
Figure 6.9: Error estimates versus exact errors for different tolerances for the *EL-LI* controller with $N^1 = 80$.

Comparison of different tolerances

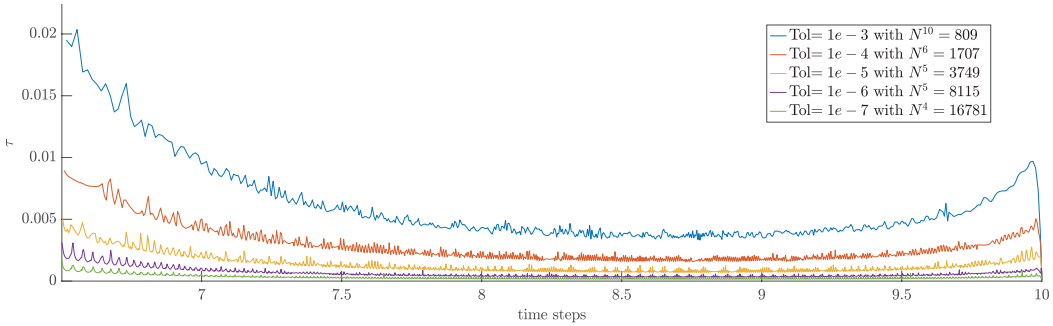
First, we compare in Figure 6.9 the error estimates with the exact errors in the final adaptive iteration, where the stopping criterion is reached, by using the *EL-LI* controller for different error tolerances. For all five tolerances, the error estimates approximate the exact errors very well and fulfill the stopping criteria within 10 iterations. Obviously, the smaller the tolerance, the more time steps and also the fewer adaptive iterations are needed. This is due to not varying too much between decreasing and increasing the time step sizes.



(a) $I = [0, 10]$.



(b) $I^{(b)} = [0, 6.5)$.



(c) $I^{(c)} = [6.5, 10]$.

Figure 6.10: Time steps for different tolerances in the adaptive strategy with *EL-LI* controller and $N^1 = 80$.

Furthermore, Figure 6.10 shows the constructed time grids at the final L-th iteration. In Figure 6.10a the whole time interval is displayed. The time interval from $[0, 6.5]$ is shown in Figure 6.10b and from $[6.5, 10]$ in Figure 6.10c. Obviously, the smaller the tolerance, the smaller the time step sizes. Due to the test problem, the time step sizes are smallest in the time interval $[6.5, 10]$. Comparing the different tolerances, the time step sizes in $[6.5, 10]$ are almost all the same size or at least closer to the same size than at the beginning of the time interval, where we see larger differences. Also, there is much more variation in time step sizes at the beginning of the time interval than at the end. In the following section, we compare the global adaptive time step control with an equidistant time step simulation. Based on the previous results, we can assume that in the time interval $[6.5, 10]$, our adaptive approach leads to nearly equidistant step sizes, and that we have a disparity especially in the beginning and in the middle of the time interval.

Comparison of simulations with equidistant time step sizes versus adaptive approaches

Here, we show the advantage of the adaptive error control strategy *EL-LI* compared to a simulation with an equidistant time step size by considering the number of time steps. To find an equidistant time step size that leads to comparable results, we identify the smallest time step size so that the error estimators or errors are of the same size. The smallest time step size for each tolerance is in the time interval $[6.5, 10]$. We choose the equidistant time step size by the maximal exact error for each time step, such that these are comparable with the given tolerance for the adaptive approach, which are shown in the following Table 6.7.

time steps	$\max e_n$
2500	9.37e-04
5500	9.27e-05
12000	9.02e-06
25000	9.99e-07
55000	9.39e-08

Table 6.7: Maximum of L^2-error for different equidistant time steps.

Finally, Table 6.8 shows a comparison of the number of time steps between the use of an equidistant time step size and our adaptive error control strategy *EL-LI*. Furthermore, the ratio of used adaptive steps to used equidistant steps is given. This is shown for the whole time interval $[0, 10]$, for $[0, 6.5)$ and for $[6.5, 10]$. In the adaptive strategy, the *EL-LI controller with limiting case condition*, Controller 4.9 is used. This means that, if the time grid is not changed until the n-th time subinterval I_n, we do not have to solve the previous $n-1$ time intervals, since we have already solved them in the previous adaptive iteration.

First, for lower tolerances, the total number of time steps in $[0, 10]$ used by the adaptive approach is smaller than that used by the equidistant one. For higher tolerances, it is the other way around. This is due to the number of adaptive iterations, which becomes smaller for lower tolerances. On the interval $[0, 6.5)$ the advantage of the adaptive approach is

immense, which can be seen by looking at the ratio of the two approaches, where up to 83.66% of time steps can be eliminated here. On the other hand, on the last time interval, the disadvantage is illustrated. The value of the ratio also highlights this. The result on $[6.5, 10]$ is quite obvious, since if an equidistant time grid is suitable for the underlying problem, as we saw in Figure 6.10, an adaptive approach cannot result in the same or fewer time steps than an equidistant time grid.

	equidistant time steps			adaptive time steps			ratio of adaptive to equidistant steps		
tol \ I	$[0,10]$	$[0,6.5)$	$[6.5,10]$	$[0,10]$	$[0,6.5)$	$[6.5,10]$	$[0,10]$	$[0,6.5)$	$[6.5,10]$
10^{-3}	2500	1625	875	4149	979	3170	165.96%	60.25%	362.29%
10^{-4}	5500	3575	1925	5805	1083	4722	105.55%	30.29%	245.3%
10^{-5}	12000	7800	4200	11171	1593	9578	93.09%	20.42%	228.05%
10^{-6}	25000	16250	8750	24643	4127	20516	98.57%	25.4%	234.47%
10^{-7}	55000	35750	19250	30483	5841	24642	55.42%	16.34%	128.01%

Table 6.8: Comparison of total number of time steps between simulations with equidistant time step sizes and our adaptive error control strategy *EL-LI with limiting case condition*.

Summary of the numerical studies on the heat equation

First, we verified the accuracy of the $cGP(2)$- and $cGP\text{-}C1(3)$-methods by observing that the $cGP(2)$-method achieved an experimental order of convergence of three in the integral-based L^2-error norm and order four in the discrete L^∞-norm. The $cGP\text{-}C1(3)$-method achieves an order of four in both error norms. Furthermore, we confirmed the accuracy of the error estimator.

The first studies of the global adaptive approach have been carried out. We investigated several adaptive strategies in combination with different controllers, where the elementary local error controller, used in the *linear interpolation strategy*, *EL-LI*, showed the best results in terms of accuracy and number of time steps. The *piecewise cubic Hermite interpolation strategy* gives comparable results, but not quite as good. We also showed that more advanced controllers such as *PC11* and *PID* lead to worse performances. Thus, we will not be testing any new controllers from the literature anymore. The adaptive strategy leads to a good performance, although in the first iteration the error estimates do not approximate the exact errors very well for larger time step sizes. Also, a smaller initial time step size does not necessarily result in a smaller number of adaptive iterations. The choice of initial time step sizes depends on the considered problem and on the tolerance. On the one hand, a small initial time step size will lead to more time steps with bigger changes. On the other hand, the time step size should not be too large, as this results in inaccurate error estimates. Furthermore, the adaptive time step control showed an advantage in computational complexity, measured in terms of the number of time steps, when compared to an equidistant time step run.

6.2. The incompressible Navier-Stokes equations

Having verified the error estimator and the global adaptive strategy for the heat equation in the previous section, we now study its application to the Navier-Stokes equations and in particular to incompressible flows. For the space discretization, we use the finite element pair Q_2/P_1^{disc}, as introduced in the Chapter (2.3).

6.2.1. The Navier-Stokes problems

We consider the Navier-Stokes equations (2.1.2) on the spatial domain $\Omega := (0,1)^2$ and on the time interval $I = [0,1]$. We set the viscosity parameter to $\nu = 1$. The velocity field $\mathbf{u} = (u,v)$ is given by

$$u(x,y,t) := x^2(1-x)^2[2y(1-y)^2 - 2y^2(1-y)]\sin(10\,\pi\,t),$$
$$v(x,y,t) := -[2x(1-x)^2 - 2x^2(1-x)]y^2(1-y)^2]\sin(10\,\pi\,t)$$

and the pressure distribution reads

$$p(x,y,t) := -(x^3 + y^3 - 0.5)(1.5 + 0.5\sin(10\,\pi\,t)).$$

The initial data is set as $\mathbf{u}_0(x,y) = \mathbf{u}(x,y,0)$.

This test problem is well suited to investigate the experimental order of convergence and to validate our constructed error estimators, especially for the pressure component. All results presented in this section are obtained with a small spatial step size of $h = 2^{-8}$.

First, the analytical errors and the experimental order of convergence for velocity and pressure are considered. The choice of the post-processed pressure solution as the initial pressure condition (3.4.10) at each time step in the $cGP(2)$-method is investigated. Then, we have a look at the computational time of the post-processing step compared to the cost of solving the $cGP(2)$ block system. Furthermore, we study the pressure error estimator in the L^2-norm (4.1.6) and the pointwise pressure error estimator (4.1.7) along with the choice of the interpolation polynomials $cLI,1$ and $cLI,2$ of the cubic Lagrangian interpolation (4.1.4). In addition, the velocity (4.1.2) and the new pressure error estimates (4.1.6) are validated. Finally, we investigate the adaptive strategies in terms of the only-velocity and the only-pressure controller, where both are set up like the Controller 4.2. Furthermore, we study a velocity and pressure controller with the *hierarchical case conditions*, Controller 4.5, and the *maximum case conditions*, Controller 4.6.

Errors and experimental orders of convergence

We start by considering Table 6.9, which shows the velocity errors in the $||\cdot||_\infty$-norm and in the $||\cdot||_2$-norm. As expected, the velocity errors in the discrete L^∞-norm demonstrates an experimental order of convergence of four for both methods, since the velocity solutions at the discrete time points coincide. In the integral-based $||\cdot||_2$-norm, the experimental order of convergence is three for the $cGP(2)$ velocity solutions and four for the post-processed $cGP\text{-}C1(3)$ velocity solutions.

Table 6.10 confirms these results by showing the discrete L^∞-norm for the intermediate Gauss-Lobatto points $t_{n,i}^{GL(5)}$. Furthermore, we note that the errors of the post-processed

	$cGP(2)$				$cGP\text{-}C1(3)$			
$\frac{1}{\tau}$	$\|\|e^u\|\|_\infty$	EOC	$\|\|e^u\|\|_2$	EOC	$\|\|\tilde{e}^u\|\|_\infty$	EOC	$\|\|\tilde{e}^u\|\|_2$	EOC
10	9.83e-04		5.95e-04		9.83e-04		6.91e-04	
20	5.67e-05	4.12	1.10e-04	2.43	5.67e-05	4.12	3.87e-05	4.16
40	4.36e-06	3.70	1.48e-05	2.90	4.36e-06	3.70	2.38e-06	4.02
80	2.83e-07	3.95	1.90e-06	2.96	2.83e-07	3.95	1.46e-07	4.02
160	1.76e-08	4.00	2.39e-07	2.99	1.76e-08	4.00	9.22e-09	3.99

Table 6.9: Velocity discrete L^∞-errors and L^2-errors.

	$cGP(2)$					
$\frac{1}{\tau}$	$\|\|e_1^{u,GL(5)}\|\|_\infty$	EOC	$\|\|e_2^{u,GL(5)}\|\|_\infty$	EOC	$\|\|e_3^{u,GL(5)}\|\|_\infty$	EOC
10	4.91e-04		5.27e-04		7.25e-04	
20	1.52e-04	1.69	4.87e-05	3.44	1.81e-04	2.00
40	2.64e-05	2.53	4.01e-06	3.60	2.82e-05	2.68
80	3.57e-06	2.88	2.51e-07	4.00	3.69e-06	2.94
160	4.56e-07	2.97	1.60e-08	3.97	4.63e-07	2.99
	$cGP\text{-}C1(3)$					
	$\|\|\tilde{e}_1^{u,GL(5)}\|\|_\infty$	EOC	$\|\|\tilde{e}_2^{u,GL(5)}\|\|_\infty$	EOC	$\|\|\tilde{e}_3^{u,GL(5)}\|\|_\infty$	EOC
10	6.38e-04		5.27e-04		9.96e-04	
20	4.69e-05	3.77	4.87e-05	3.44	4.02e-05	4.63
40	2.51e-06	4.23	4.01e-06	3.60	2.66e-06	3.92
80	1.52e-07	4.04	2.51e-07	4.00	1.54e-07	4.11
160	9.80e-09	3.96	1.60e-08	3.97	9.79e-09	3.98

Table 6.10: Velocity discrete L^∞-errors on time points $t_{n,i}^{GL(5)}$, $i = 1,\ldots,3$.

velocity solutions at the intermediate time points are of the similar order of magnitude as at the discrete time point $t_{n,2}^{GL(5)}$, which is also the discrete time point $t_{n,1}^{GL(3)}$ and such converges with order four.

In the case of the pressure component, we first reinforce the choice of the post-processed pressure solution as the initial pressure solution in each time subinterval (3.4.10). Therefore, in Table 6.11a, the errors for the $cGP(2)$-time-stepping scheme, where the initial pressure is set to the $cGP(2)$ pressure solution $P_{n+1}^0 = p_{h,\tau}(t_n)$ are presented and in Table 6.11b the initial pressure is set to the post-processed $cGP\text{-}C1(3)$ pressure solution $P_{n+1}^0 = \tilde{p}_{h,\tau}(t_n)$. We denote by "$cGP\text{-}C1(3)$-cLI" the approach, where the higher order pressure solution is applied as an initial solution in each time step in the $cGP(2)$-method and the post-processed solution at some time is approximated by the cubic Lagrangian interpolation 4.1.5. In Figure 6.11a, the discrete L^∞-error does not become smaller than $9e-6$ and the experimental order of convergence does not fit due to no correction of the pressure. Perhaps, this is because of the dominance of the space error, since the space error in the L^2-norm is of order h^3 for the velocity and of order h^2 for the pressure. Also, this test problem has already been investigated by Hussain et al. for the discretization scheme $cGP(2)\text{-}Q_2/P_1^{disc}$ [31]. They showed that for a fixed mesh size and sufficiently small time step, the space error becomes dominant for the velocity variable. Therefore, it

	$cGP(2)$				$cGP\text{-}C1(3)-cLI$			
$\frac{1}{\tau}$	$\|\|e^p\|\|_\infty$	EOC	$\|\|e^p\|\|_2$	EOC	$\|\|\tilde{e}^{p,cLI}\|\|_\infty$	EOC	$\|\|\tilde{e}^{p,cLI}\|\|_2$	EOC
10	1.85e-03		7.82e-03		1.85e-03		9.83e-03	
20	1.09e-04	4.09	2.97e-03	1.40	1.09e-04	4.09	7.76e-04	3.66
40	1.27e-05	3.10	3.83e-04	2.96	1.27e-05	3.10	5.17e-05	3.91
80	9.36e-06	0.44	4.88e-05	2.97	9.36e-06	0.44	6.38e-06	3.02
160	9.34e-06	0.00	7.41e-06	2.72	9.34e-06	0.00	5.52e-06	0.21

(a) Pressure discrete L^∞-errors and L^2-errors with an initial pressure for the subintervals of $P^0_{n+1} = P^2_n$.

	$cGP\text{-}C1(3)$				$cGP\text{-}C1(3)-cLI$			
$\frac{1}{\tau}$	$\|\|\tilde{e}^p\|\|_\infty$	EOC	$\|\|\tilde{e}^p\|\|_2$	EOC	$\|\|\tilde{e}^{p,cLI}\|\|_\infty$	EOC	$\|\|\tilde{e}^{p,cLI}\|\|_2$	EOC
10	1.85e-03		7.82e-03		1.85e-03		9.83e-03	
20	1.09e-04	4.09	2.97e-03	1.40	1.09e-04	4.09	7.76e-04	3.66
40	8.65e-06	3.65	3.83e-04	2.96	8.65e-06	3.65	5.15e-05	3.91
80	5.84e-07	3.89	4.87e-05	2.98	5.84e-07	3.89	3.26e-06	3.98
160	8.09e-08	2.85	6.13e-06	2.99	8.09e-08	2.85	2.11e-07	3.95

(b) Pressure discrete L^∞-errors and L^2-errors with an initial pressure for the subintervals of $P^0_{n+1} = \tilde{p}_{h,\tau}(t_n)$.

Table 6.11: Errors and experimental order of convergence depending on the choice of the initial pressure in each time step.

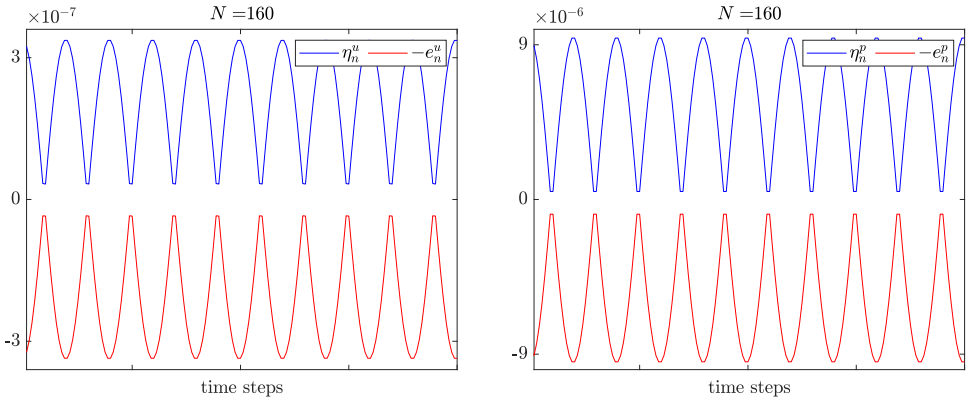
stands to reason that the same will happen for the pressure. Furthermore, we can observe the equivalent for the post-processed L^2-error. Only the results for the L^2-error for the $cGP(2)$-method are as expected. In Table 6.11b, the errors in the discrete L^∞-norm show an experimental order of convergence of four for both methods, since the pressure solution on the discrete time points are equal and are about $8e-8$. Thus confirms the choice of the post-processed $cGP\text{-}C1(3)$ pressure solution as initial condition for the next time interval. Furthermore, we see, that the new cubic Lagrangian interpolation leads to an experimental order of convergence of four in the L^2-norm, which is perfectly suitable as a pressure error estimator for the pressure error of the higher order $cGP\text{-}C1(3)$-method.

Table 6.12 confirms these new results by showing the discrete L^∞-norm over the intermediate Gauss-Lobatto points $t^{GL(5)}_{n,i}$. The new cubic Lagrangian interpolation of the $cGP\text{-}C1(3)$ solution leads to solutions that are convergent with order four in the whole time interval.

Having verified that we are obtaining solutions of different accuracy, we now have a first look at the velocity and pressure error estimates and the exact errors in Figure 6.11. Both error estimators accurately approximate the exact errors. Furthermore, the velocity error is smaller than the pressure error by a factor of 10, which can be explained by the spatial discretization.

	$cGP\text{-}C1(3)$					
$\frac{1}{\tau}$	$\|\|\tilde{e}_1^{p,GL(5)}\|\|_\infty$	EOC	$\|\|\tilde{e}_2^{p,GL(5)}\|\|_\infty$	EOC	$\|\|\tilde{e}_3^{p,GL(5)}\|\|_\infty$	EOC
10	1.12e-02		9.23e-04		1.11e-02	
20	4.63e-03	1.27	8.82e-05	3.39	4.63e-03	1.26
40	7.10e-04	2.71	7.57e-06	3.54	7.10e-04	2.71
80	9.31e-05	2.93	4.87e-07	3.96	9.31e-05	2.93
160	1.18e-05	2.98	7.39e-08	2.72	1.18e-05	2.98
	$cGP\text{-}C1(3) - cLI$					
	$\|\|\tilde{e}_1^{p,GL(5),cLI}\|\|_\infty$	EOC	$\|\|\tilde{e}_2^{p,GL(5),cLI}\|\|_\infty$	EOC	$\|\|\tilde{e}_3^{p,GL(5),cLI}\|\|_\infty$	EOC
10	1.41e-02		9.23e-04		1.40e-02	
20	1.42e-03	3.31	8.82e-05	3.39	1.42e-03	3.30
40	9.71e-05	3.87	7.57e-06	3.54	9.71e-05	3.87
80	6.21e-06	3.97	4.87e-07	3.96	6.21e-06	3.97
160	3.95e-07	3.97	7.39e-08	2.72	3.95e-07	3.97

Table 6.12: Pressure discrete L^∞-errors on time points $t_{n,i}^{GL(5)}$, $i = 1,\dots,3$.



(a) Velocity. (b) Pressure.

Figure 6.11: Error estimates versus exact errors for $\tau^1 = 1/160$.

Computational effort of solving the post-processing step

At this point, we would like to practically substantiate a statement made in Chapter 5 concerning the computational complexity of solving the resulting post-processed system. Since the post-processed solutions satisfy the continuity properties at each discrete point of a subinterval, it is not necessary to solve them at each time step. However, due to the importance of the post-processing step in the adaptive approach and also due to inaccuracies, it is worthwhile to do it in every time step. Especially since the computational time needed to construct and solve the post-processed system is very small. This can be seen in Table 6.13. Here, a direct solver was used to solve the block systems. Other non-parallel solvers were also tested, such as the Vanka preconditioner in an FGMRES solver, but there was no significant speedup.

$\frac{1}{\tau}$	total	$cGP(2)$	post-processing	$cGP(2)$ in %	post-processing in %
10	6,904	6,726	178	97%	3%
20	24,998	24,702	296	99%	1%
40	14,957	14,473	484	97%	3%
80	27,530	26,683	846	97%	3%
160	43,849	42,792	1,057	98%	2%

Table 6.13: Computational time in seconds of the $cGP(2)$-method and of the post-processing step.

In Table 6.13 the runtimes are given in seconds and measure the duration of all time steps. We illustrate this for a different number of time steps. We verify that the post-processing step takes only about 1% to 3% of the total run time. Compared to the runtime of the $cGP(2)$-method, this is of little importance. The explanation for this is of course the size of the underlying block system, but most of all the linearity. While the post-processing step is always a linear problem, the $cGP(2)$ system here is nonlinear and has to be linearized by e.g. fixed-point iteration. In summary, we will solve the post-processing step in every time step.

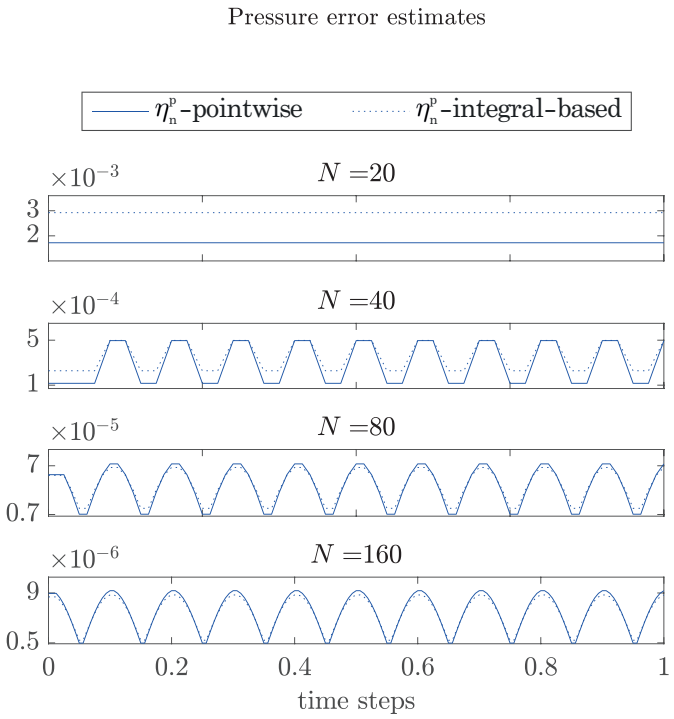
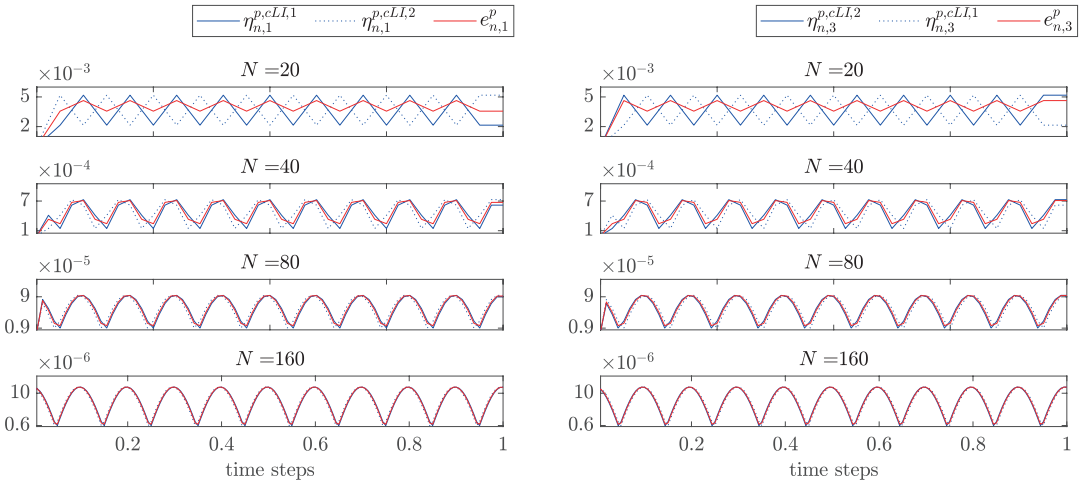


Figure 6.12: L^2-pressure error estimates and pointwise pressure error estimates for different equidistant time step sizes.

Pressure error estimator

Here, the pressure error estimator is examined in more detail. The integral-based L^2-pressure error estimators (4.1.6), and the pointwise pressure error estimators (4.1.3), for different equidistant time step sizes are shown in Figure 6.12. There is no significant difference between the two variants of the pressure error estimator, which confirms the selection of a pointwise error estimator in the following flow around a cylinder benchmark, but the integral based L^2 pressure error estimator, would also be possible.

Another aspect that we addressed in Section 4.1.2 is the choice of interpolation polynomials and nodes in the neighboring subintervals. Since we need four nodes to perform a cubic interpolation and, due to the $cGP(2)$-time-stepping method, we have only three discrete time points in the interval I_n of order four that refer to the points $[t_{n,0}^{GL(5)}, t_{n,2}^{GL(5)}, t_{n,4}^{GL(5)}]$, we need a fourth node from a neighboring subinterval. We defined the interpolation nodes depending on the evaluated time point t as in equation (4.1.4) for the two interpolation polynomial $\tilde{p}_{h,\tau}^{cLI,j}$, $j = 1,2$ in equation (4.1.5). Since we here use an equidistant time step size, we can consider the intermediate points $t_{n,1}^{GL(5)} \in (t_{n,0}^{GL(5)}, t_{n,2}^{GL(5)})$ that are associated to the polynomial "*cLI,1*" and $t_{n,3}^{GL(5)} \in (t_{n,2}^{GL(5)}, t_{n,4}^{GL(5)})$ that are associated to the polynomial "*cLI,2*" as examples of time points that are in the time interval. However, these time points can be interpolated by either choice of neighboring subintervals.



(a) $t_{n,1}^{GL(5)}$. (b) $t_{n,3}^{GL(5)}$.

Figure 6.13: Pressure error estimators based on "cLI,1" and "cLI,2" versus exact pressure errors different equidistant time steps sizes.

We show in Figure 6.13 the difference between using both interpolation polynomials once for the time $t_{n,1}^{GL(5)}$ in Figure 6.13a and $t_{n,3}^{GL(5)}$ in 6.13b. Here $\eta_{n,i}^{p,cLI,1,GL(5)}$ denotes the pointwise error due to cubic Lagrangian interpolation "*cLI,1*" and the error estimator evaluated at time $t_{n,i}^{GL(5)}$. The smooth blue line refers to the application with the closest neighboring subinterval for the time points, so for $t_{n,1}^{GL(5)}$, it would be "*cLI,1*" and for $t_{n,3}^{GL(5)}$, it would be "*cLI,2*". The dotted blue line corresponds to the other interpolation

polynomial. The red line represents the analytical pressure error. For both figures, we see that for a smaller number of time steps, the choice of the closest subinterval for that time point is more consistent with the exact error. Regarding the adaptive strategy, where we start with a coarse time grid, it is favorable to choose the fourth interpolation node depending on the evaluated time point. Since the dotted blue line is shifted to the left in Figure 6.13a and to the right in Figure 6.13b. However, as the number of time steps increases, both pointwise error estimators approximate the exact errors well.

In short, we have confirmed our choice of the cubic Lagrangian interpolation polynomial to obtain the fourth order pressure solution. Furthermore, we have shown that both, the pointwise pressure error estimator and the integral-based L^2-norm pressure error estimator, are appropriate. From now on, we use the pointwise pressure error estimator and the choice of the fourth interpolation node in the cubic Lagrangian interpolation depends on the evaluated time point, which is set to the second point of the Gauss-Legendre formula $t_{n,2}^{G(4)}$.

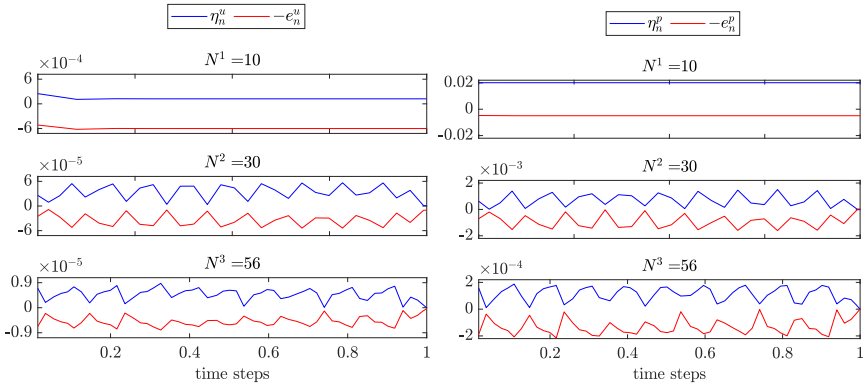
Adaptive approach

In the following, we examine some control strategies based on the velocity and pressure error estimator. First, we provide a reminder of how we denote the strategy. The first component is the basic strategy, here the elementary local error control *EL*. Next, we specify the parameter by which we want to control the time step size and thus which error estimator to use. The abbreviation *-u* designates the velocity, while in case of *-p* or *-u-p*, we control the error estimates of the pressure or the velocity and pressure respectively. Further, we can choose between *hierarchical case conditions*, the Controller 4.5, where we first check the velocity error estimate and then the pressure error estimate, and the variant with the maximal value of velocity and pressure estimates so with *maximum case conditions*, denoted by the suffix *-M*, as the Controller 4.6. Without this suffix, we use the *hierarchical case conditions*. Finally, we add the interpolation strategy, which is denoted by *-LI* for *linear interpolation*. Since the *EI-LI* controller was identified as the best controller in the heat equation studies, we will focus our further studies on this controller.

Figure 6.14 shows the error estimates and the exact errors of the *EL-u-LI* controller in Figure 6.14a for the velocity and in Figure 6.14b for the pressure. Since the adaptive strategy only checks the velocity error estimates, only these estimates fulfill the tolerance. The pressure error estimator is larger. However, both error estimates approximate the analytical errors very well and lead to a more equilibrated errors.

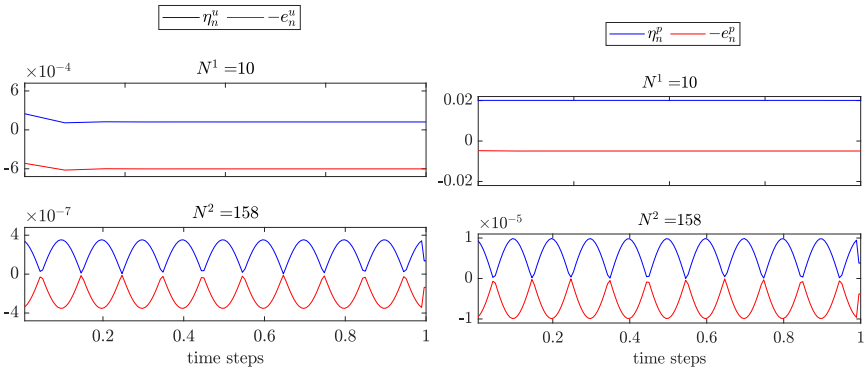
After looking at the velocity controller, we now consider the only-pressure controller in Figure 6.15. In contrast to the above Figure 6.14, here the errors are not so much equilibrated, but the error estimators lead to sufficiently approximations. Now all pressure error estimators are smaller than the given tolerance. In addition, the velocity error estimators are also smaller, but this is obvious since for this test problem the velocity errors are smaller by a factor of 10 compared to the pressure errors. Reducing the time step size will of course reduce the velocity errors.

The velocity and pressure error estimators are considered in the case of the *EL-u-p-LI* controller with *hierarchical case conditions* in Figures 6.16 and 6.17. The error estimates against the exact errors are shown in Figure 6.16. The errors are quite well equilibrated.



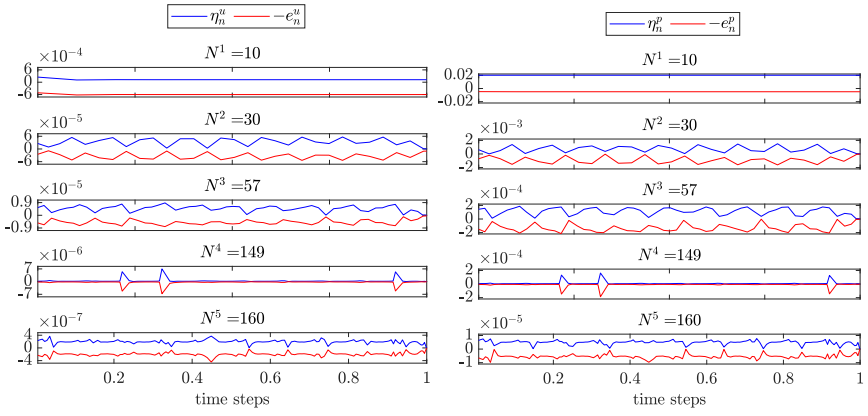
(a) Velocity. (b) Pressure.

Figure 6.14: Error estimates versus exact errors for the controller *EL-u-LI* with a tolerance of 10^{-5} and $N^1 = 10$.



(a) Velocity. (b) Pressure.

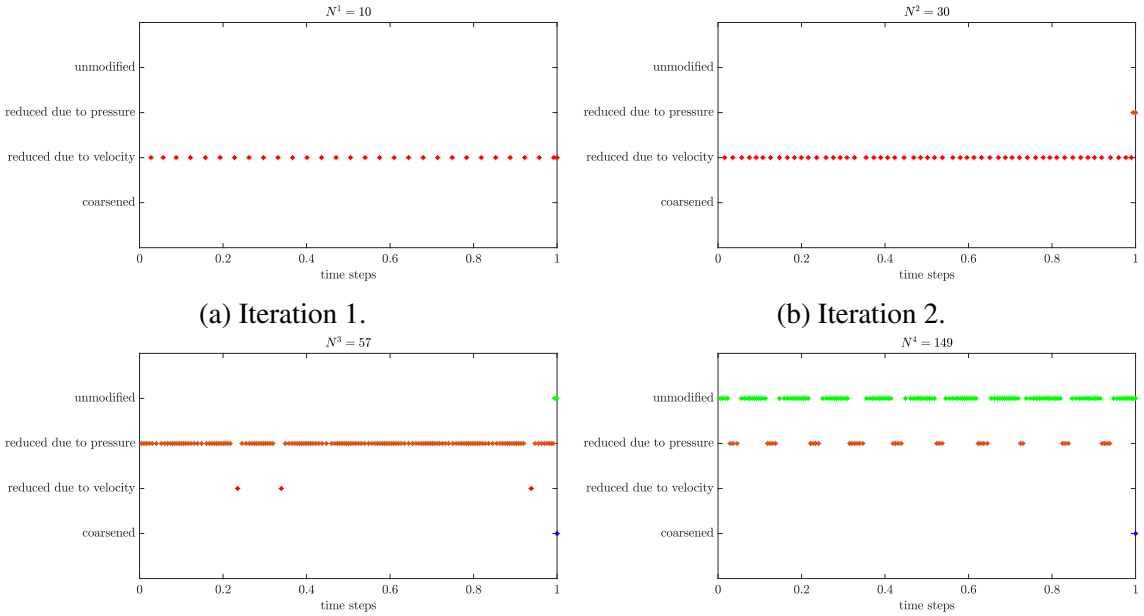
Figure 6.15: Error estimates versus exact errors for the controller *EL-p-LI* with a tolerance of 10^{-5} and $N^1 = 10$.



(a) Velocity. (b) Pressure.

Figure 6.16: Error estimates versus exact errors for the controller *EL-u-p-LI* with a tolerance of 10^{-5} and $N^1 = 10$.

However, we still need five adaptive iterations to satisfy the stopping criterion. To explain, why we need more adaptive iterations than with the previous controller, we have to look at the reasons for the modification in Figure 6.17. This controller reduces the time step size after two adaptive iterations only based on velocity. After the third iteration, we start to reduce the time step size adapted from the pressure error estimator. Finally, for the last adaptive iteration, most of the time step sizes have been accepted and about a third or a quarter of the time steps is reduced due to the pressure error estimator.



(a) Iteration 1.

(b) Iteration 2.

(c) Iteration 3.

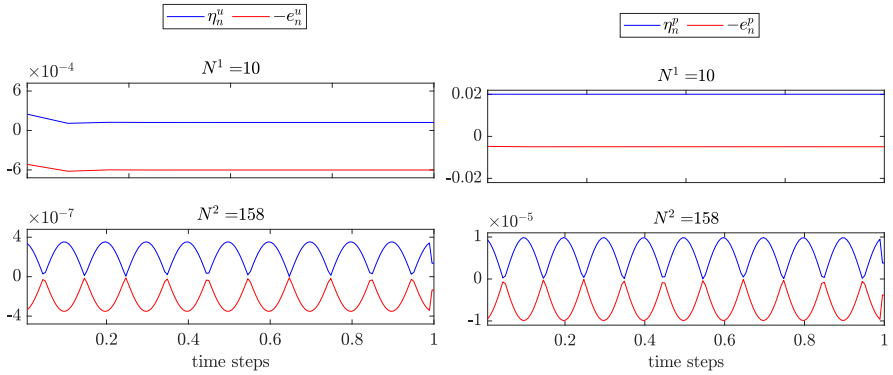
(d) Iteration 4.

Figure 6.17: Reasons for modification of controller *EL-u-p-LI* with a tolerance of 10^{-5}.

Figures 6.18 and 6.19 show the error estimates and the exact errors, as well as the reasons for modification by using the *EL-u-p-LI-M* controller. This approach requires only two iterations and the same number of time steps, as the *EL-p-LI* controller. Considering the reasons for the modification, it is obvious why these two controllers behave the same, since it only reduces the time step size based on the pressure error estimator. In such test problems, where the velocity error has the same behavior as the pressure and is much smaller, there is no need to modify the time step size depending on the velocity. However, as we will see for other test problems, an adaptive time step controller based on both velocity and pressure is advisable.

Summary of the numerical studies on the Navier-Stokes equations

The choice of the post-processed pressure solution as the initial pressure condition at each time step in the $cGP(2)$-method was confirmed. Along with verifying the experimental order of convergence of three in the L^2-error norm for the post-processed pressure solution as well as order four in the discrete L^∞-norm. Most interesting was the experimental order of convergence of four for the new higher order cubic interpolated post-processed pressure solution. We also illustrated that the computational cost of the post-processing



(a) Velocity. (b) Pressure.

Figure 6.18: Error estimates versus exact errors for the controller *EL-u-p-LI-M* with a tolerance of 10^{-5} and $N^1 = 10$.

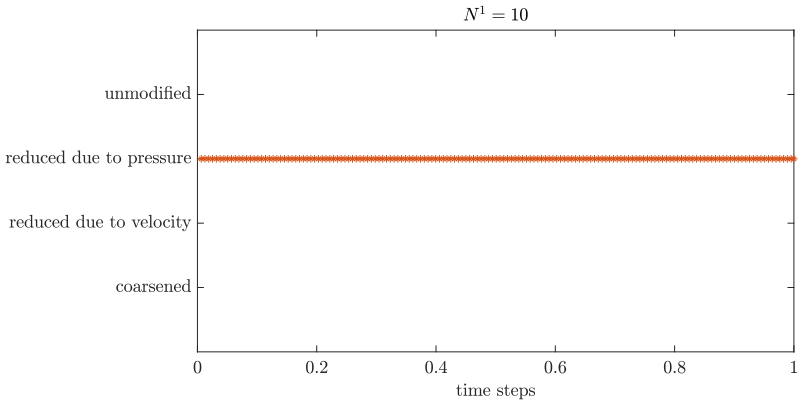


Figure 6.19: Reasons for modification of controller *EL-u-p-LI-M* with a tolerance of 10^{-5}.

step is negligible compared to the cost of solving the $cGP(2)$ block system. Furthermore, we showed that the pressure error estimator in the L^2-norm and the pointwise pressure error estimator do not differ for small time step sizes. The same holds for the choice of the interpolation polynomials "*cLI,1*" and "*cLI,2*" of the cubic Lagrangian interpolation. In addition, the velocity and the new pressure error estimates were validated and approximated the analytical pressure error very well. Finally, we studied the adaptive strategies in terms of the velocity-only, the pressure-only and the velocity and pressure controller. The last one for the controller with the *hierarchical case conditions* and the *maximum case conditions*. Since the pressure values were lower than the velocity values, the *maximum case conditions* was the better choice.

6.2.2. The flow around a cylinder benchmark

In this section, we will investigate the global adaptive time-stepping scheme for the special flow configuration given by the flow around a cylinder benchmark [52]. In Section 2.1.3 this benchmark was described. The coarse mesh under investigation is displayed in Figure 6.20. This will be uniformly refined until a refinement level of four is reached. In order to discretize the problem in space, we also apply the inf-sup stable Q_2/P_1^{disc} element pair, which results in 8320 elements and 92352 spatial degrees of freedom on the mesh level four. Since we want to study several incompressible flows, we will investigate the flow around a cylinder benchmark for two Reynolds numbers, which are $Re = 100$ and $Re = 250$.

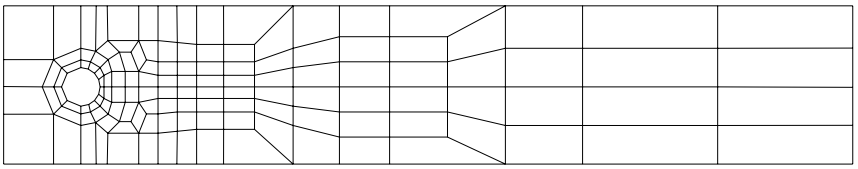


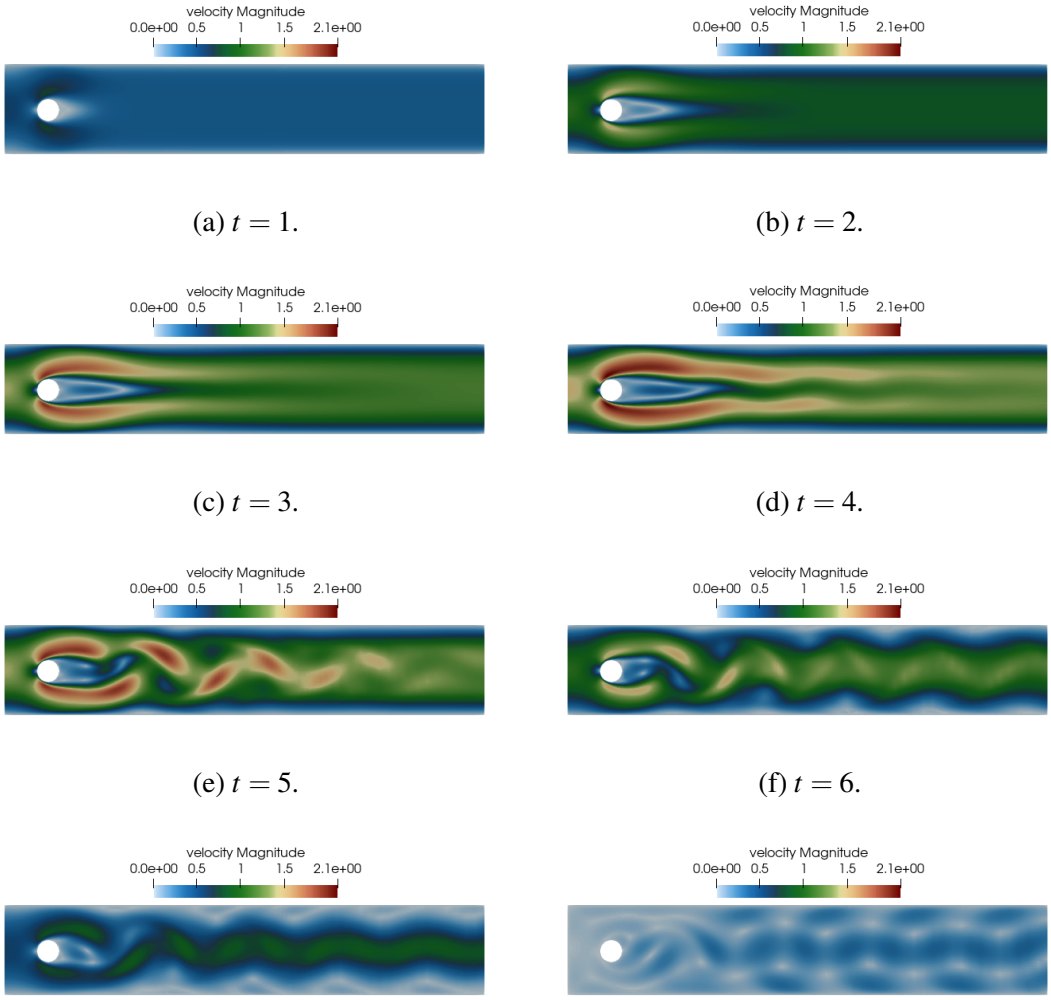
Figure 6.20: Coarse mesh (level one) for flow around a cylinder.

First, we look at the most characteristic quantities of interest, such as the drag and lift coefficients, which are verified by reference values [38]. Then, we validate the velocity and pressure error estimator. In addition, different types of *EL* controllers, see Controllers 4.5, 4.7 - 4.8, are studied in terms of accuracy, measured in the distance error of the lift coefficient to reference values of the lift coefficient. Furthermore, we investigate global-in-time adaptive time-stepping based on different combinations of variables such as absolute, relative or relative and absolute velocity, pressure and lift values, see (4.1.2), (4.1.7) - (4.1.12). This is also studied for different tolerances and a comparison between simulations with equidistant time step sizes. Additionally, we now denote the error estimator for the lift coefficient by η^l, analogously for the drag coefficient η^d, instead of η^{c_l} and η^{c_d}. Studying a Reynolds number of 250, we study the controllers with considering the experimental order of convergence of the velocity error estimates, Controller 4.10.

Reynolds number $Re = 100$

We start our studies with a Reynolds number of 100, as described in Chapter 2.1.3. The resulting velocity magnitudes at different times are shown in Figure 6.21. At the beginning, for $t = 1$, the velocity magnitude is close to zero, explicitly in the surface of the cylinder. With more time, the flow is more adapted to the attendance of the cylinder. At $t = 4$ the maximum velocity is reached and the vortex shedding is detected. After that, we see a turbulent flow pattern that becomes less and less until the final time.

As the most characteristic quantities of interest, we illustrate the drag and lift coefficients as well as the pressure difference for equidistant time step sizes in Figure 6.22. The drag coefficient and pressure difference are shown over the entire time interval in Figures 6.22a,6.22c and on the interval $[5.04, 5.8]$ in Figures 6.22b,6.22d. The lift coefficients over the entire time interval and on the subinterval are illustrated in Figure 6.22e and Figure 6.22f.



(a) $t = 1$. (b) $t = 2$.

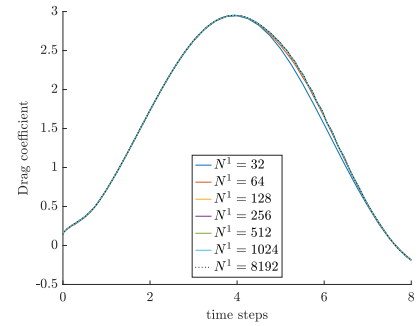
(c) $t = 3$. (d) $t = 4$.

(e) $t = 5$. (f) $t = 6$.

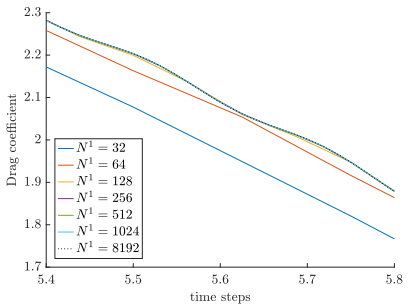
(g) $t = 7$. (h) $t = 8$.

Figure 6.21: Velocity magnitudes on mesh level four and $\tau = 2^{-10}$ for different times.

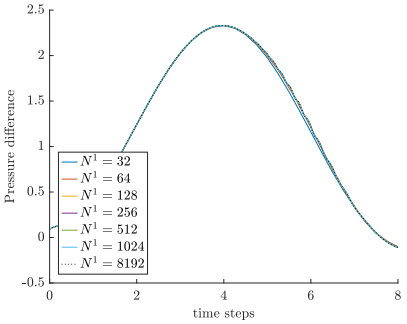
After the maximum of the drag coefficient is reached, there are tiny oscillations in the drag values, as shown in the second Figure 6.22b. The pressure difference has almost the same behavior as the drag coefficient. Even for very large time steps, the drag coefficient does not change significantly. The lift coefficient shows much more oscillation than the drag coefficient. Especially, the red line, which is related to the number of time steps with $N^1 = 64$ is slightly shifted, such that the evolution of the coefficient is not wrong, but not exact. In addition, in the subinterval of $[5.67, 5.715]$, where the maximum of the lift values is located, we notice that only small time step sizes, which belong to the number of time steps for example of $N^1 = 1024$ or $N^1 = 512$, are very close to the values of $N^1 = 8192$. This phenomenon is a typical consequence of vortex shedding behind the cylinder, which is characteristic of this flow problem. The timing of the vortices is closely linked to the lift coefficient, resulting in a phase error that explains why the lift coefficients are shifted. Furthermore, up to the time $t = 3.5$, the lift values are close to zero, which motivates the use of relative error estimators.



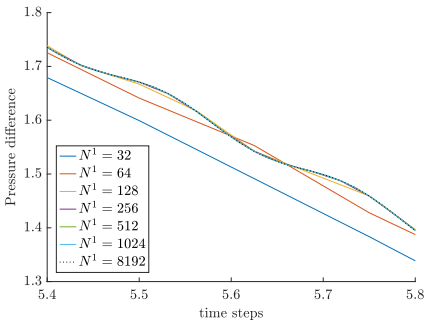
(a) Drag coefficient on $[0, 8]$.



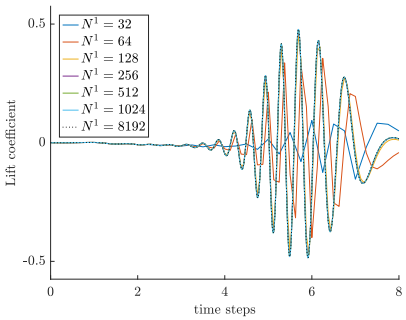
(b) Drag coefficient on $[5.4, 5.8]$.



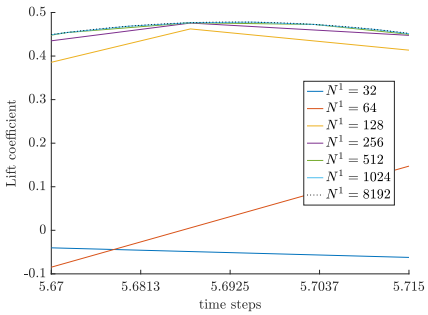
(c) Pressure difference on $[0, 8]$.



(d) Pressure difference on $[5.4, 5.8]$.



(e) Lift coefficient on $[0, 8]$.



(f) Lift coefficient on $[5.67, 5.715]$.

Figure 6.22: Drag and lift coefficients for equidistant time step sizes.

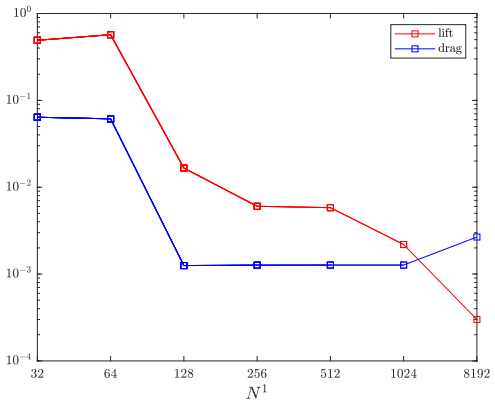


Figure 6.23: Distance errors of drag and lift coefficients for equidistant time step sizes.

We verify our results by comparing the maximum drag and lift values and the associated time points to reference values

$$(c_{d,max}^{ref}, t_{d,max}^{ref}) = (2.950921575, 3.93625),$$
$$(c_{l,max}^{ref}, t_{l,max}^{ref}) = (0.47795, 5.693125),$$

given in [38]. We compute a distance error, see [36], using the formula

$$err_{drag} = \sqrt{(t_{d,max}^{ref} - t_{d,max})^2 + (c_{d,max}^{ref} - c_{d,max})^2},$$
$$err_{lift} = \sqrt{(t_{l,max}^{ref} - t_{l,max})^2 + (c_{l,max}^{ref} - c_{l,max})^2}. \tag{6.2.1}$$

In Figure 6.23 we show the error to the reference values for different time step sizes. Both errors become smaller as the time step size is reduced. Although the drag coefficient is taken into account, the error does not change for the time step size. In addition, the error of the lift coefficients is larger than that of the drag coefficients. Combined with the oscillations seen above for the lift coefficient, it is reasonable to focus more on the lift coefficient than on the drag coefficient in this test case.

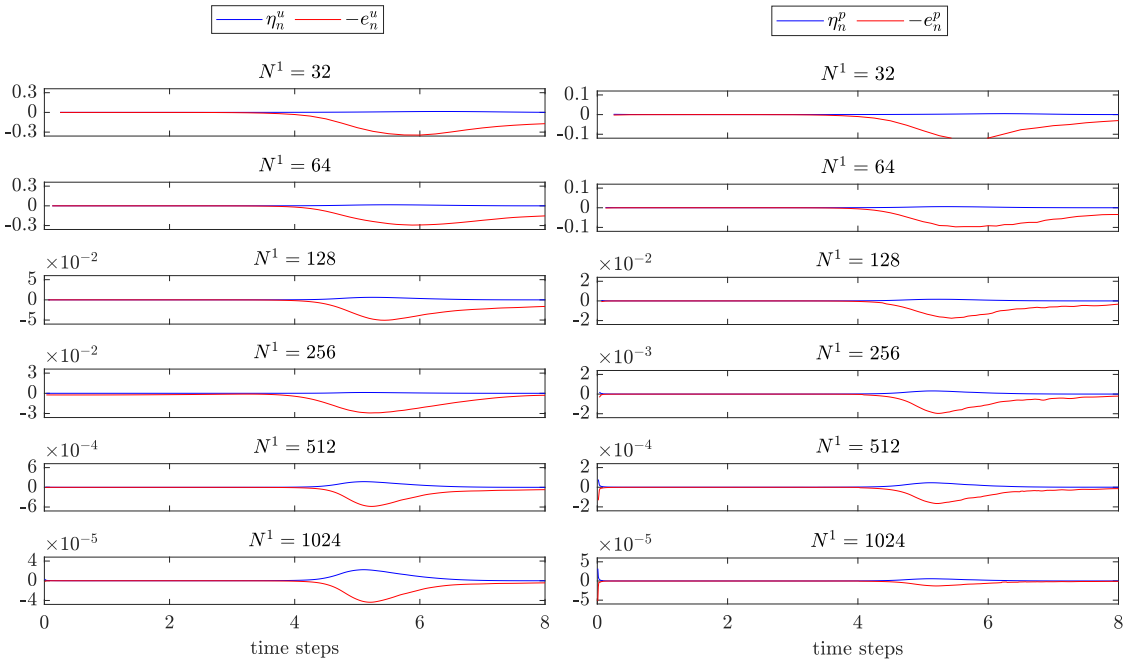
To get a first impression of how the different error estimates behave, we plot them in Figure 6.24 for equidistant time step sizes. We approximate the reference solution, which is the solution using an equidistant time step size of 2^{-10}. Therefore, e_n^u is calculated from the $cGP(2)$ solution and the reference solution. The same is valid for the other variables.

The errors and error estimates for the velocity variable are shown in Figure 6.24a, for the pressure variable in Figure 6.24b, for the lift coefficient in Figure 6.24c and for the drag coefficient in Figure 6.24d. First, the errors are almost the same for all variables. For the pressure variable, the errors are slightly smaller than for the velocity. Similarly, we see that for the errors of the lift and drag coefficients. Overall, the error estimates are smaller than the reference errors. For the smallest time step size, which corresponds to 1024 time steps, the error estimates approximate the errors sufficiently. An error or error estimate of 10^{-5} in the middle to the end of the time interval implies much smaller errors at the beginning of this time interval. For these reasons and also because of the previous results showing, that even a large initial time step size can be sufficient for the adaptive strategy, we will now investigate the global strategy mainly for tolerances of $10^{-3}, 10^{-4}, 10^{-5}$ and use an initial time step size of 2^{-3}.

Adaptive strategy

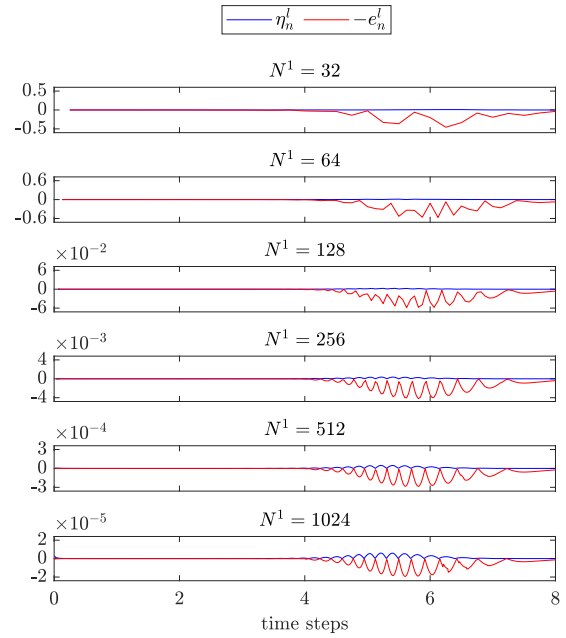
In this section, we compare controllers based on the velocity and pressure variables *u-p*, on the lift coefficients *l*, or a combination of these *u-p-l*. Furthermore, we distinguish between error estimators that use an absolute *-abs*, a relative *-rel*, or a mixture of both error *-relabs*, see, (4.1.2), (4.1.7) - (4.1.12). The latter uses absolute error estimates if the absolute value of the solution is less than 0.001, otherwise a relative one. If just one suffix is given, it means that all variables use the same type of error estimator. In addition, when considering controllers that rely on multiple variables, we can choose between *hierarchical case conditions* or the *maximum case condition*,*-M*.

We use the integral-based velocity error estimator (4.1.2), the pointwise pressure error estimator (4.1.7) and the pointwise lift error estimator (4.1.8). Both pointwise error estimators are evaluated at the second point of the Gauss-Legendre formula $t_{n,2}^{G(4)}$.

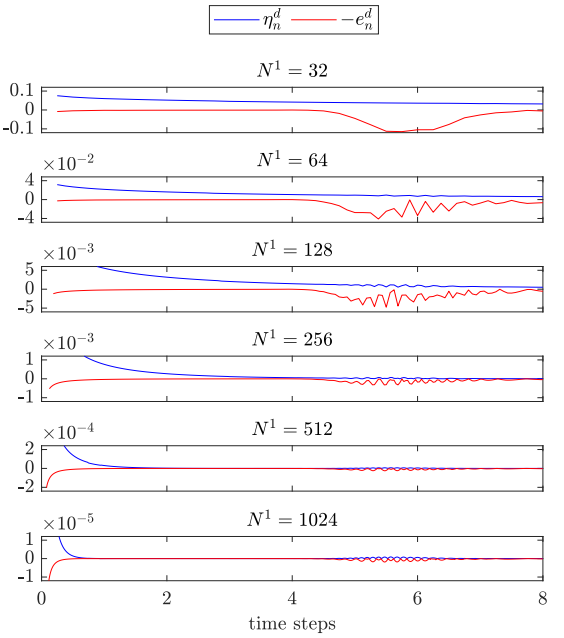


(a) Velocity.

(b) Pressure.



(c) Lift coefficient.



(d) Drag coefficient.

Figure 6.24: Error estimates versus the reference errors for different equidistant time step sizes.

Since we observed that for equidistant time step sizes all error estimates are nearly equal in magnitude, the *hierarchical case conditions* are the closer choice. Another argument is that since the lift and drag error estimates are computed from the velocity and pressure solution, it makes sense to check the velocity error estimate first, then the pres-

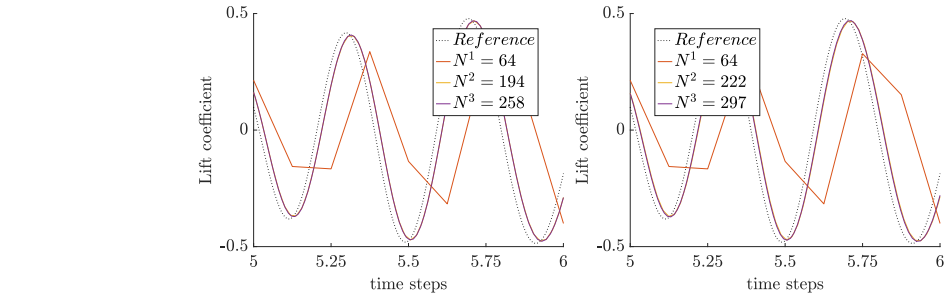
sure, and finally the lift estimates.

Nevertheless, we show in Table 6.14 a comparison between all the studied error estimators for a different choice of controllers. We investigate the total number of adaptive iterations L, the number of time steps in the final iteration N^L, where the stopping criterion is reached, and the total number of time steps $\bar{N}$ during all adaptive iterations. Although the number of time steps is not the most important criterion for selecting a controller due to the global approach, it is still an important quantity. Especially, for this test problem, the number of time steps of the *maximal case condition controller* is about twice as large as for of the *hierarchical case condition* strategy. Furthermore, the *cubic Hermite interpolation strategy* leads to similar results as the *EL-LI* controller for some combinations of error estimators. In other cases, however, it requires much larger time steps. Therefore, we will only study the *elementary local error controller with linear interpolation with hierarchical case conditions*, *EL-LI* in this section, so we skip the abbreviation *EL-LI*. In addition, almost every controller needs about three adaptive iterations, which is a considerable amount. By the way, this table gives a good overview of the controllers to be studied.

	EL-LI-M			EL-LI			EL-CHI		
error estimator	L	N^L	$\bar{N}$	L	N^L	$\bar{N}$	L	N^L	$\bar{N}$
u-p-abs	3	573	1055	3	258	516	3	258	516
u-p-rel	3	669	1237	3	297	583	3	293	576
l-abs	4	390	1167	4	173	496	4	172	494
l-rel	2	944	1008	2	533	597	3	756	1352
l-relabs	4	1035	2868	3	489	886	3	510	918
u-p-l-abs	3	577	1060	3	258	516	3	240	464
u-p-l-rel	3	1400	2419	3	495	813	4	661	1410
u-p-abs-l-rel	3	1385	2394	3	589	953	4	741	1689
u-p-rel-l-abs	3	668	1236	3	298	584	3	294	577
u-p-l-relabs	3	1093	1935	3	374	684	3	377	689
u-p-abs-l-relabs	3	1068	1879	3	419	734	3	422	735

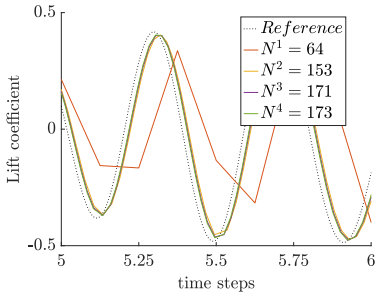
Table 6.14: Comparison between controllers with *hierarchical case conditions* and *maximum case conditions* with *linear interpolation* and *piecewise cubic Hermite interpolation* with $\tau^1 = 2^{-3}$ and $tol = 10^{-4}$.

The lift coefficients for the different controllers and for all required adaptive iterations are presented in Figure 6.25. The approximated lift value for the equidistant time step size of 2^{-10} is referred to as "Reference". All controllers result in a lift coefficient that follows the same evolution as the reference lift value. However, some controllers result in a lift coefficient that is shifted. We have already seen this for the equidistant case. The controllers based on the smallest amount of time steps are shifted. There are controllers, mostly including a relative or relative and absolute error estimates of the lift coefficients, which refine the time step size such that there is hardly any shift visible. Although, as we will see next, no error estimator accurately detects this shift, but several perform better.

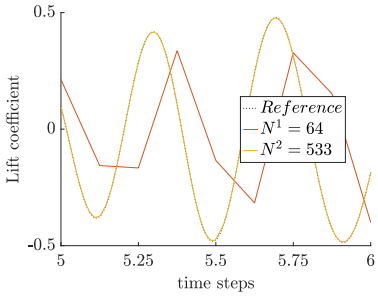


(a) u-p-abs.

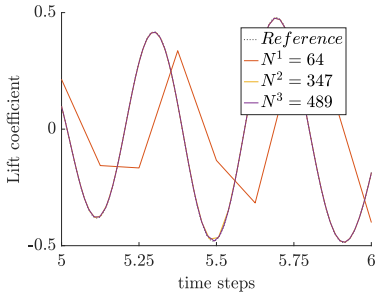
(b) u-p-rel.



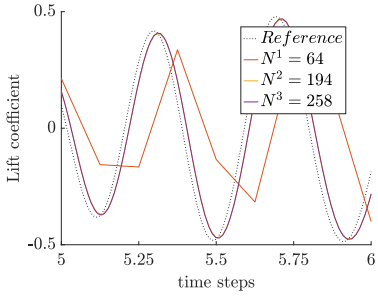
(c) l-abs.



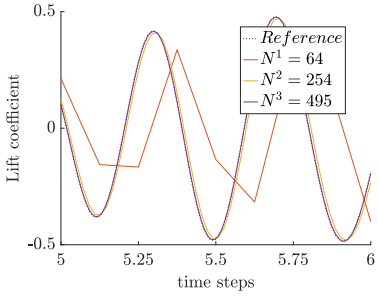
(d) l-rel.



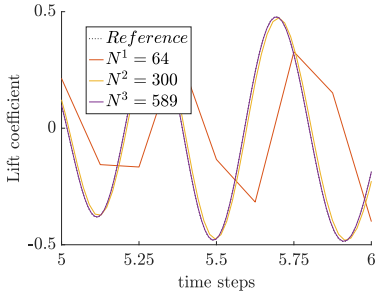
(e) l-relabs.



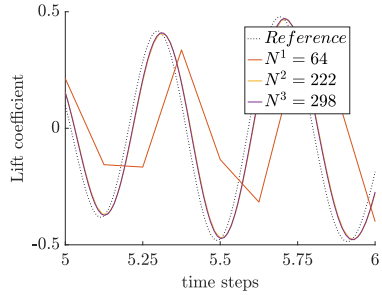
(f) u-p-l-abs.



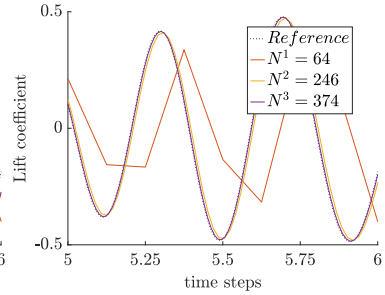
(g) u-p-l-rel.



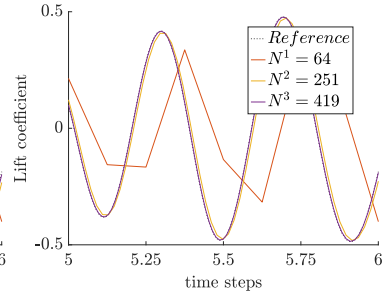
(h) u-p-abs-l-rel.



(i) u-p-rel-l-abs.



(j) u-p-l-relabs.



(k) u-p-abs-l-relabs.

Figure 6.25: Lift coefficients in case of several *EL-LI* controllers with $\tau^1 = 2^{-3}$ and $tol = 10^{-4}$.

Now, we summarize the errors computed by the distance formula using the reference values from (6.2.1) for several controllers and three tolerances in Figure 6.26. For all controllers, the distance errors are not below the specified tolerance. However, this is not a problem, as the distance errors are measured differently compared to the error estimators, and the tolerance applies to the error estimators. Many controllers reduce the error when considering a smaller tolerance. However, a few controllers achieve a small error for any given tolerance, namely the *u-p-abs-l-relabs* and the *u-p-l-relabs*, as well as the *u-p-l-relabs*, *u-p-abs-l-rel* and l-rel controllers.

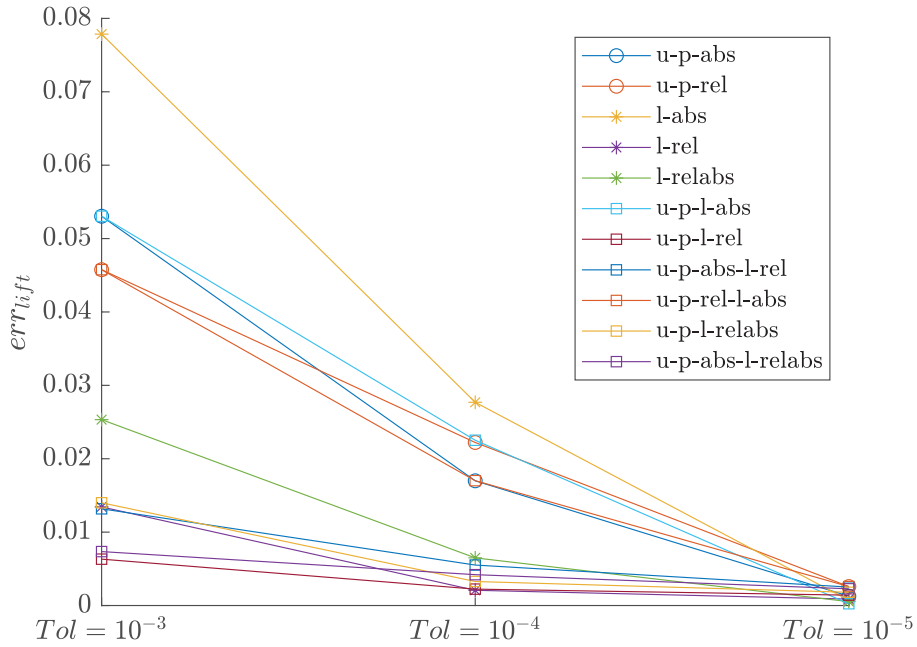
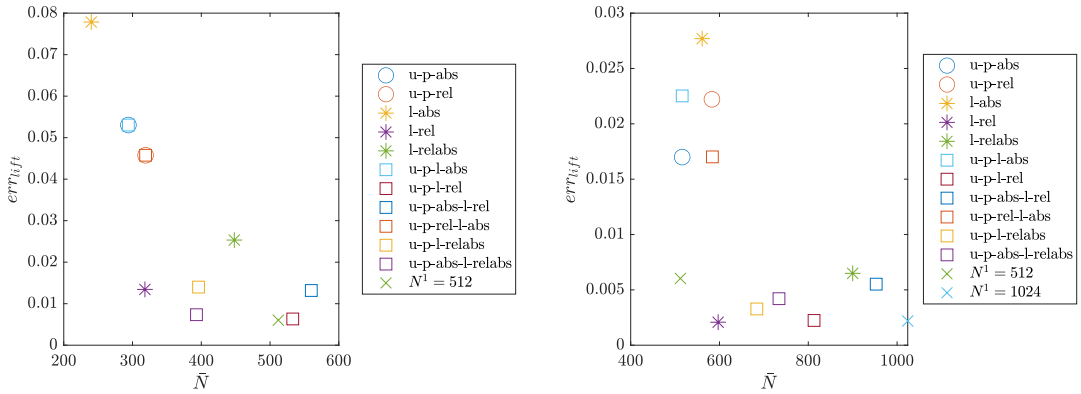
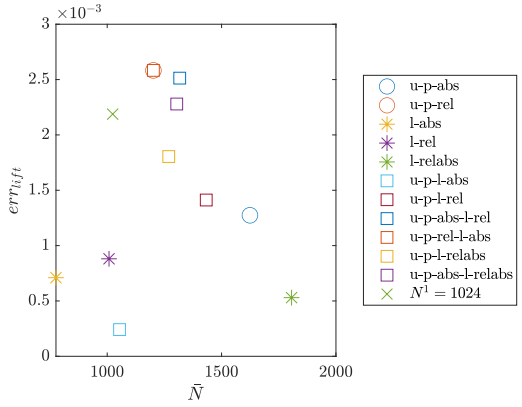


Figure 6.26: Distance errors of lift coefficients for several *EL-LI* controllers for different tolerances with $\tau^1 = 2^{-3}$.

To relate the previous observation to the use of an equidistant time step size, consider Figure 6.27, where the distance errors are plotted against the total number of time steps required. Several controllers are examined, and the distance errors are also shown for different equidistant time step sizes for three tolerances. Controllers whose distance error is in the upper half of the figure are less accurate than those in the lower half. The number of time steps is also taken into account, so that the left part of the figure shows a low number and the right part a high number. Accordingly, the controllers in the lower left part of the figure show the best performance. Considering a tolerance of 10^{-3} and 10^{-4} in Figures 6.27a and 6.27b, we see, that the same controllers behave effectively. Three controllers show better performance than the equidistant simulation, which are the *u-p-abs-l-relabs*, the *u-p-l-relabs* and *l-rel*. For a tolerance of 10^{-5}, all controllers are reliable, but most of them require more time steps over all adaptive iterations than the equidistant simulation.



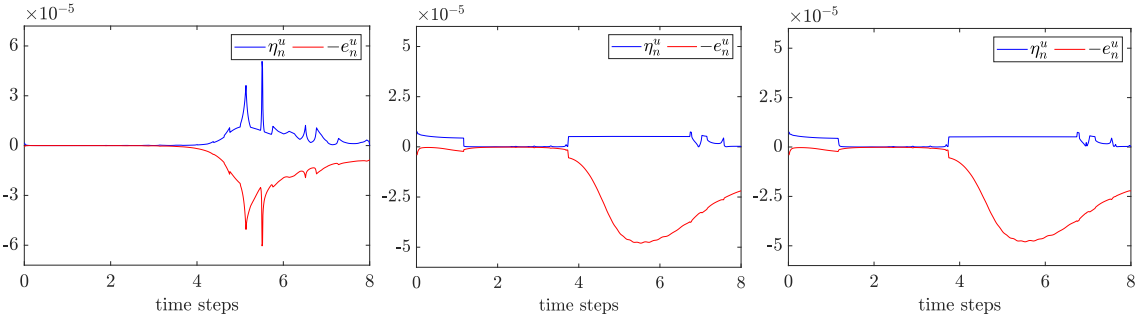
(a) $tol = 10^{-3}$. (b) $tol = 10^{-4}$.



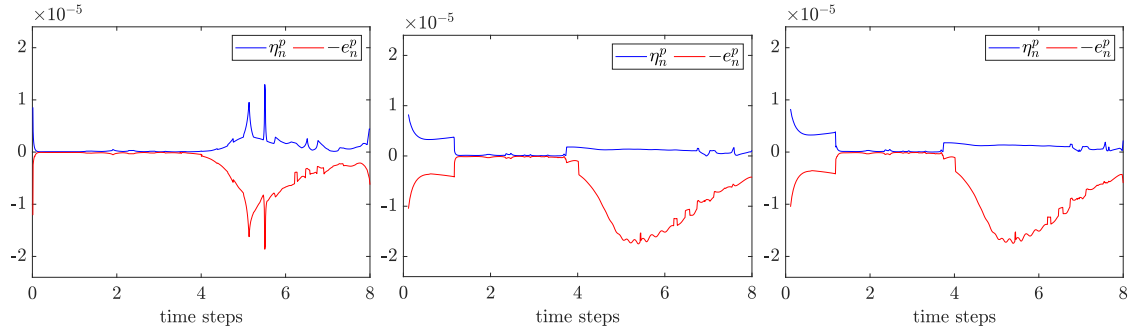
(c) $tol = 10^{-5}$.

Figure 6.27: Distance errors of lift coefficients for several *EL-LI* controllers by using different tolerances with $\tau^1 = 2^{-3}$ and equidistant time step sizes performances.

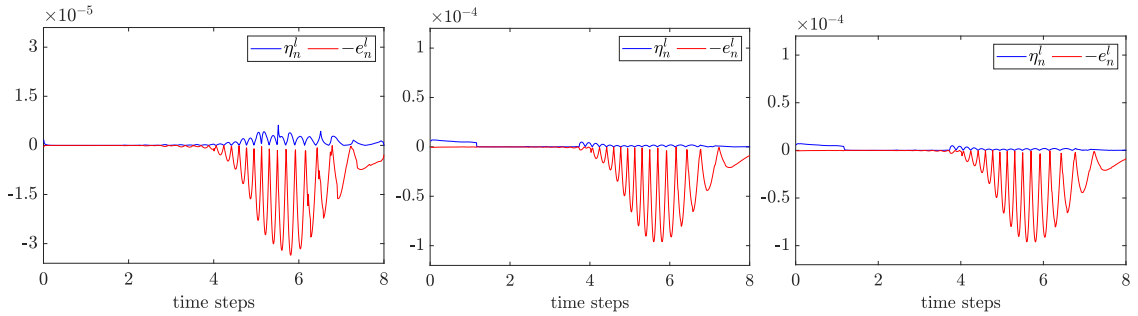
To show the accuracy of the error estimates, we consider Figure 6.28. The first row shows the velocity error estimates and the $cGP(2)$ velocity errors, the second row the pressure error estimates and the $cGP\text{-}C1(3)$ pressure errors and the third the error estimates for the lift coefficient and the lift coefficients errors. First, the velocity error estimates are smaller than the tolerance, when the controller includes the velocity variable, as in Figures 6.28b and 6.28c. Although in both figures the error estimates do not approximate the reference error as well as in Figure 6.28a. We see an analogous behavior for the pressure variable. Overall, however, the velocity and pressure error estimators are satisfactory. The error estimates of the lift coefficients do not approximate the reference error very well, but they are not completely inaccurate. However, they are significantly smaller than the reference errors, which confirms our previous results. Nevertheless, questions arise as to whether these approximations would improve if a smaller tolerance would be tested, and whether the same would apply to the drag coefficient, which will be studied next.



(a) l-rel. (b) u-p-abs-l-rel. (c) u-p-abs-l-relabs.

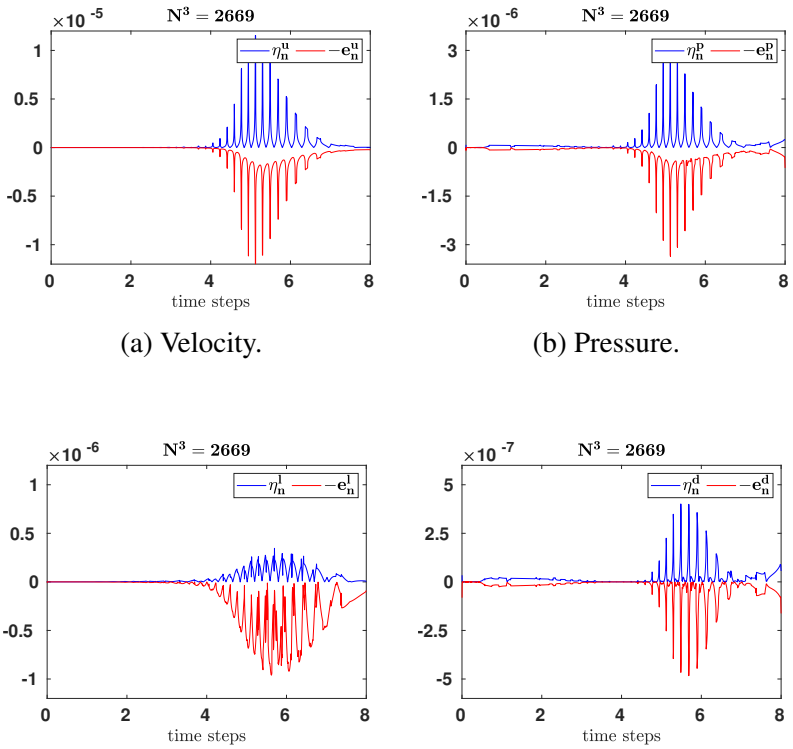


(d) l-rel. (e) u-p-abs-l-rel. (f) u-p-abs-l-relabs.



(g) l-rel. (h) u-p-abs-l-rel. (i) u-p-abs-l-relabs.

Figure 6.28: Error estimators verses reference errors for different controllers with $\tau^1 = 2^{-3}$ and $tol = 10^{-5}$.



(a) Velocity. (b) Pressure.

(c) Lift coefficient. (d) Drag coefficient.

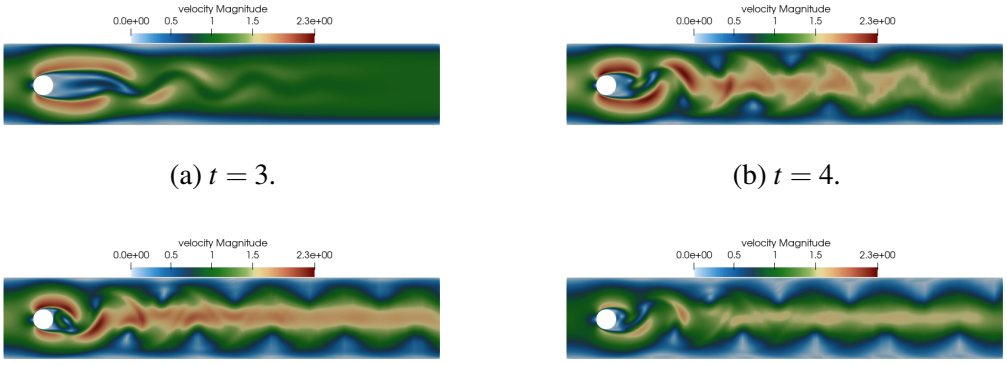
Figure 6.29: Error estimators verses reference errors for the *EL-LI-l-rel* controller with $\tau^1 = 2^{-3}$ and $tol = 10^{-6}$.

To answer the above questions, we choose the *EL-LI-l-rel* controller as an example controller. In Figure 6.29 a simulation with a tolerance of 10^{-6} is shown and further the error estimators and reference errors of the drag coefficient are added in Figure 6.29d. We see in Figures 6.29a,6.29b and 6.29d, that the error estimates of the velocity, pressure and drag coefficients approximate their reference error well. With smaller tolerances and thus a correspondingly higher number of time steps, the error estimates of the lift coefficients reflect the reference errors more accurately, but they have not reached the necessary level of quality. Since the error estimator of the drag coefficient is a good approximation, we can assume that in principle the error estimation of those coefficients is not the reason. Rather it has to do with effects on the lift coefficient such as the vortex shedding. The timing of the vortex is strongly connected to the lift coefficient, resulting in a phase error that explains why the lift coefficients are shifted. Furthermore, the *cGP-C*1 solution is unable to reflect this phase error, meaning that the error estimator cannot capture it.

Overall, it is essential to include the error estimates of the lift coefficients in a relative or relative and absolute manner in the controllers, since these controllers show the best performance.

Reynolds number $Re = 250$

Now a Reynolds number of $Re = 250$ is examined on mesh level four. Figure 6.30 shows the velocity magnitude at times three, four, five and six. Compared to the velocity magnitude we saw in Figure 6.21 for a Reynolds number of 100, the velocity shows a wider velocity distribution. The flow is getting more turbulent, as expected, since the higher Reynolds number can be associated with more intense turbulence, so the velocity fluctuates more.

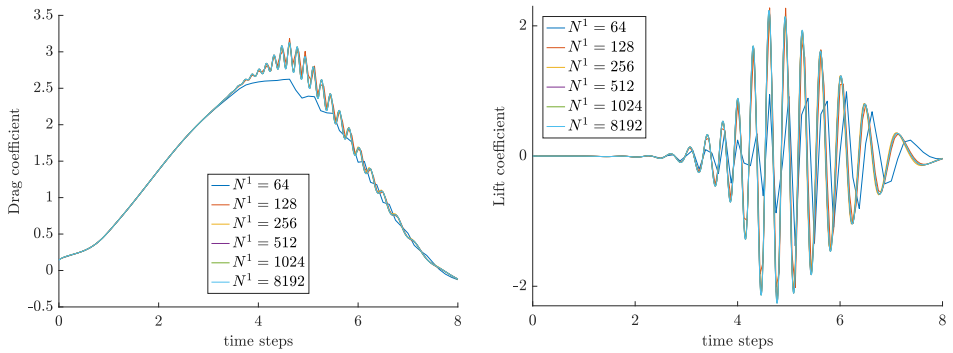


(a) $t = 3$. (b) $t = 4$.

(c) $t = 5$. (d) $t = 6$.

Figure 6.30: Velocity magnitudes on mesh level four and $\tau = 2^{-10}$ for different times, $Re = 250$.

The accuracy in this and the following sections is quantified by reference values obtained from our simulations with a very small time step size on the same spatial mesh. Here we used a time step size of $1/2^{-10}$.



(a) Drag coefficient. (b) Lift coefficient.

Figure 6.31: Drag and lift coefficients for equidistant time step sizes, $Re = 250$.

The drag and lift coefficients are shown in Figure 6.31. Now, the drag value also oscillates a little, starting just before the maximum value is reached and continuing to the final time. Also, the lift coefficient is shifted a bit to the beginning and is higher than in the $Re = 100$ test case.

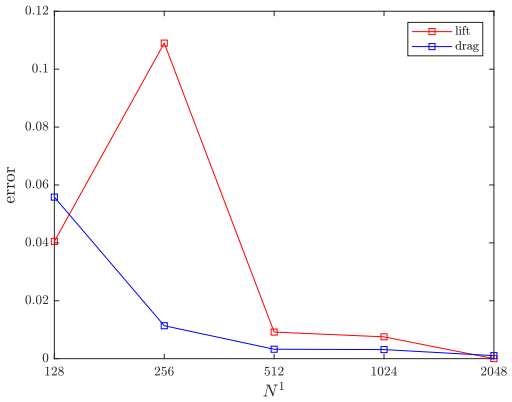


Figure 6.32: Distance errors of drag lift coefficients for equidistant time step sizes, $Re = 250$.

The distance errors of the drag and lift coefficients for equidistant time steps is also examined in Figure 6.32. Both errors seem to stagnate at about 0.005 for small time step sizes.

Consideration of the experimental order of convergence

In addition, this section investigates whether the order of convergence of the velocity error estimator, as described in Controller 4.10, can be used to ensure that the estimator and, consequently, the controller behave as intended. This is because the post-processing step depends on the $cGP(2)$ solution, which must be sufficiently accurate. Otherwise the error estimator cannot approximate the analytic or referenced error well.

Figure 6.33 shows a pointwise experimental order of convergence for different equidistant time step sizes. Obviously, the smaller the time step size, the more constant the EOC. By using smaller time step sizes, the EOC of the velocity variable is about three, while the EOC of the pressure variable is about two until the time $t = 2$ is reached, and then it is about three. Perhaps, this is due to instabilities or not considering the exact initial pressure, but an approximation. In particular, we start our adaptive iteration, which will be shown in the next figures, by using the initial number of time steps of $N^1 = 64$ and $N^1 = 128$. Therefore, the experimental order of convergence examined at each time step is shown in Figure 6.33a. Both error estimates lead to an experimental order of convergence that is not constant and varies between one and five.

To apply the Controller 4.10, we perform first two adaptive iterations with two equidistant time steps. Then, in the third adaptive iteration, we compute the order of convergence on each time grid point on the first time interval G^1 by considering the absolute velocity and pressure error estimates of the first and second adaptive iterations. Now, for each time step, we first check whether the experimental order of convergence of velocity and pressure is between $[2,4]$, since the experimental order of convergence of the velocity error estimator and the pressure error estimator should be three. Also, if the time step size is not changed for a time step, we cannot get a significant EOC value. If either of these is the case, we continue with the controller and change the time step size accordingly. If not, the time step size is halved. Then the next new time grid point is checked. The stopping

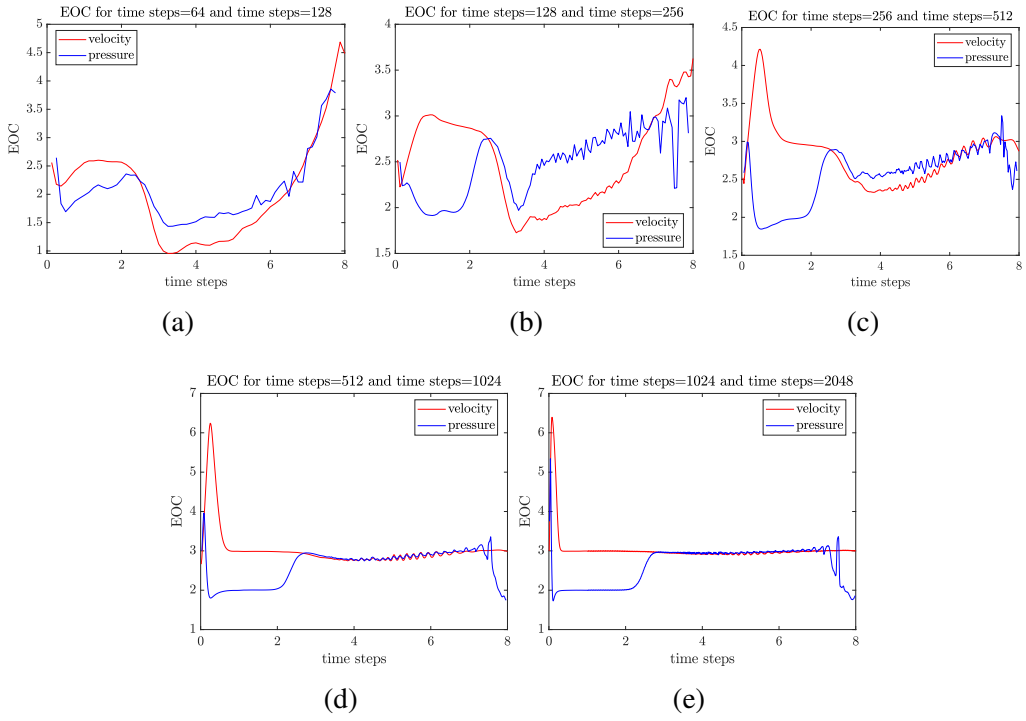


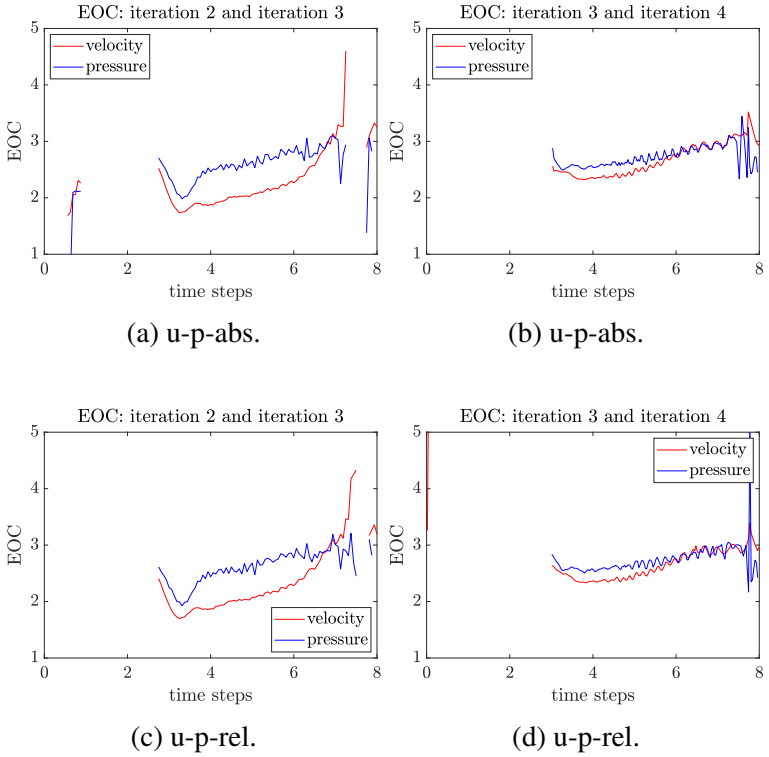
Figure 6.33: Experimental order of convergence of equidistant time step sizes, $Re = 250$.

criterion for the adaptive iteration, which requires that all error estimates be less than the specified tolerance, has been modified to include the additional condition that all velocity EOC values must be in the range of two to four.

Furthermore, we use the integral-based velocity error estimator (4.1.2), the pointwise pressure error estimator (4.1.7) and the pointwise error estimator for the lift coefficient (4.1.8). Both pointwise error estimators are evaluated at the second point of the Gauss-Legendre formula $t_{n,2}^{G(4)}$. We also use the *EL* controller, Controller 4.2 in the EOC controller, Controller 4.10. The tolerance in the stopping criterion is set to 10^{-3} and the maximum number of iterations is set to five.

Consideration of the experimental order of convergence in the adaptive approach

In this paragraph, we will study the global adaptive approach by considering the experimental order of convergence within the control strategy. In the next three Figures 6.34 6.35 and 6.36 the experimental order of convergence for each time step by applying different controllers is shown. Each figure is assigned to a controller category. The first category includes the velocity and pressure error estimators. The second category includes only-lift error estimates, while the third category is a combination of velocity, pressure, and lift variables. The achieved experimental order of convergence of one controller is shown in a row, with multiple adaptive iterations per column. If we do not obtain significant EOC values by not changing the time step size in the last two iterations, see (4.4.1), we do not include them in the plot.



(a) u-p-abs. (b) u-p-abs.

(c) u-p-rel. (d) u-p-rel.

Figure 6.34: Experimental order of convergence of *EL-LI* controllers of first category using a tolerance of 10^{-3} and $\tau^1 = 0.125$, $Re = 250$.

Considering the first two controllers in Figures 6.34a-6.34d, which depend only on velocity and pressure, we see that one adaptive iteration brings the EOC values closer to three, which leads to the assumption of a good error estimation. Hence, the stopping criterion is reached after four adaptive iterations.

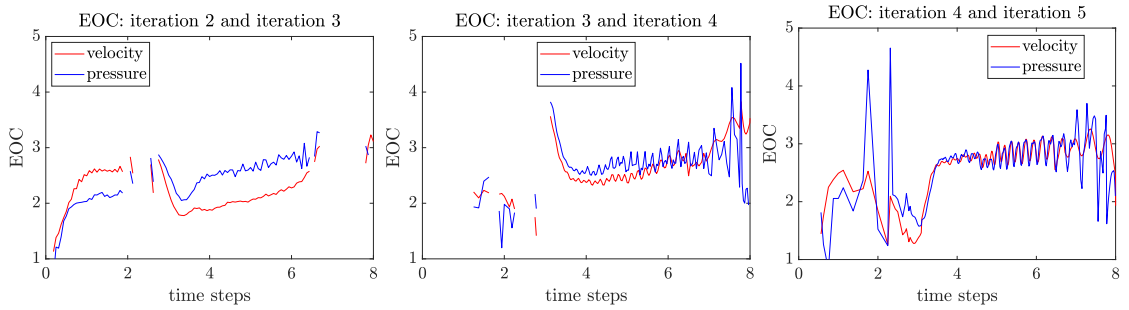
Figure 6.35 illustrates the three lift-only controllers. All three controllers show more oscillatory behavior than in the previous category. However, the absolute lift controller and the mixed relative and absolute controller fail to converge within the specified five iterations, which we can see in row one and three. The relative lift controller shows that the EOC is more consistent at a value of three, as observed in the first two iterations. Taking the first two categories into account, it is reasonable to assume that combining a relative lift error estimator with a velocity and pressure error estimator is a sensible choice, as will be shown next.

The third category of controllers is dependent on velocity, pressure and lift variables. The results of this are illustrated in Figure 6.36. The experimental order of convergence in the final iteration is observed to vary very close to three. Around $t = 2$, and before, particularly the pressure shows some outbreaks especially for the two controllers using both relative and absolute values, which further fail to satisfy the stopping criterion.

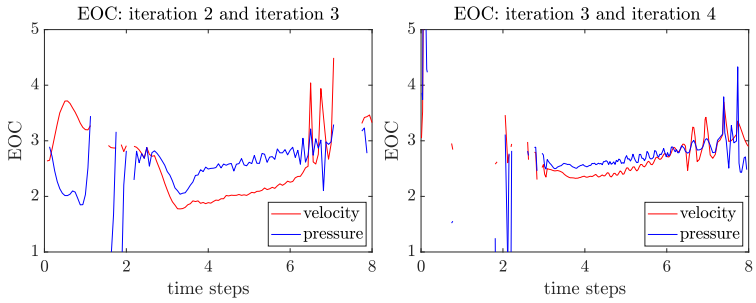
Finally, the controllers in the adaptive strategy are compared without and with consideration of the order of convergence as observed by the distance error measurement. Figure 6.37a illustrates the distance errors over the total number of time steps required without considering the order of convergence in the controller. Figure 6.37b, on the other hand, shows the same scenario with the order of convergence considered, Controller 4.10.

In contrast, considering the order of convergence results in significantly higher accuracy for all controller types. This is also the case for those controllers, which show high distance errors in the case without checking the experimental order of convergence. It is evident that the total number of time steps involved in the entire adaptive iteration process also increases.

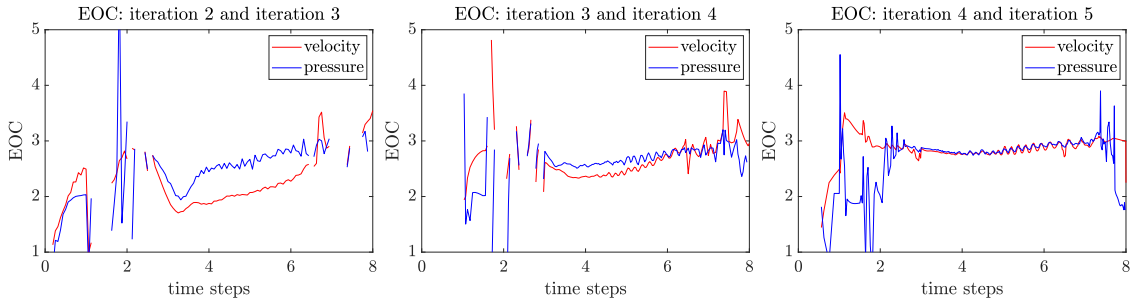
In conclusion, in the absence of information on the underlying test problem, it is strongly recommended to include the order of convergence. Moreover, should this imply an increased number of total time steps, the primary objective is to guarantee the accuracy of the quantities of interest.



(a) l-abs. (b) l-abs. (c) l-abs.

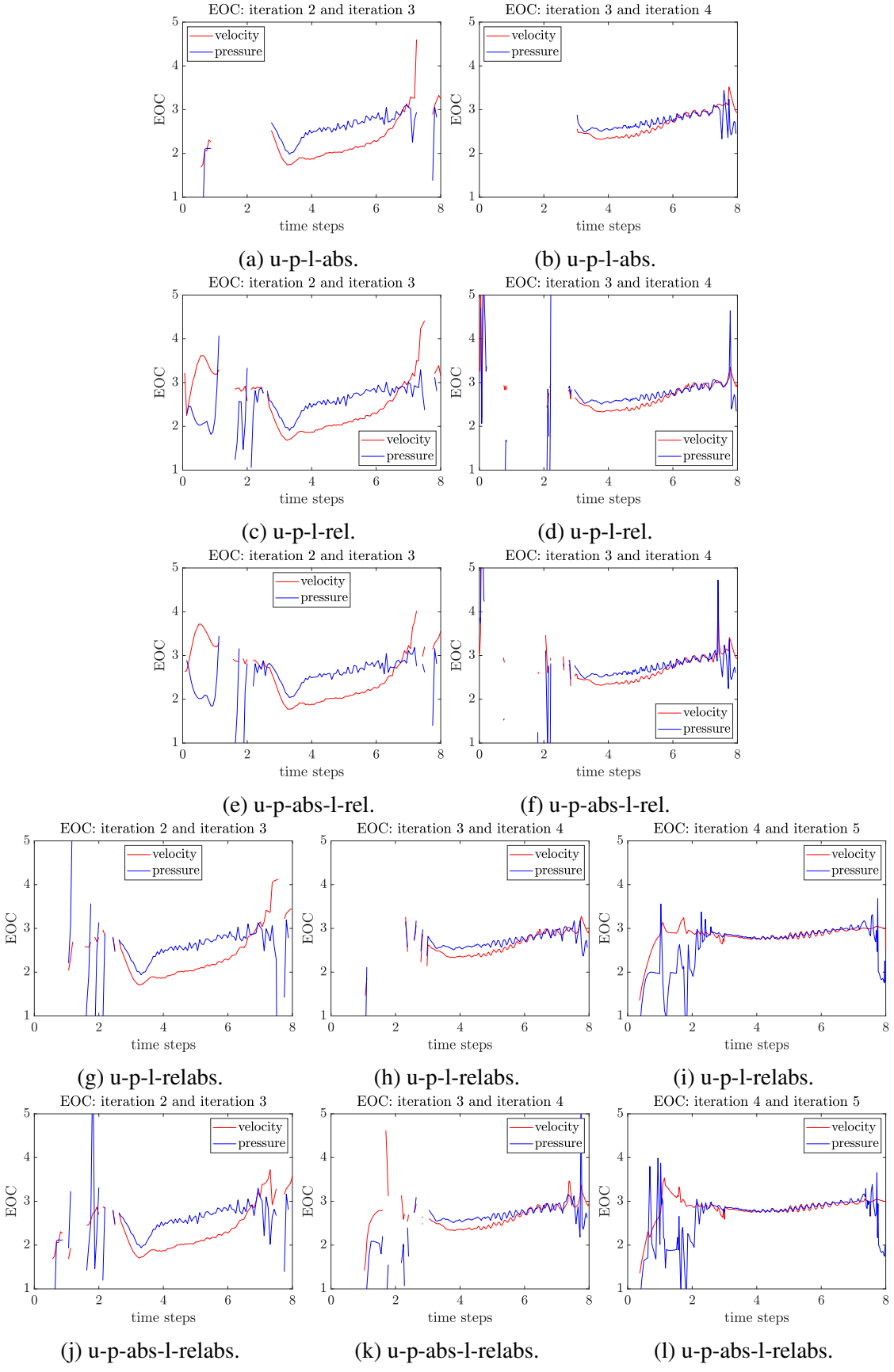


(d) l-rel. (e) l-rel.



(f) l-relabs. (g) l-relabs. (h) l-relabs.

Figure 6.35: Experimental order of convergence of *EL-LI* controllers of second category using a tolerance of 10^{-3} and $\tau^1 = 0.125$, $Re = 250$.



(a) u-p-l-abs. (b) u-p-l-abs.

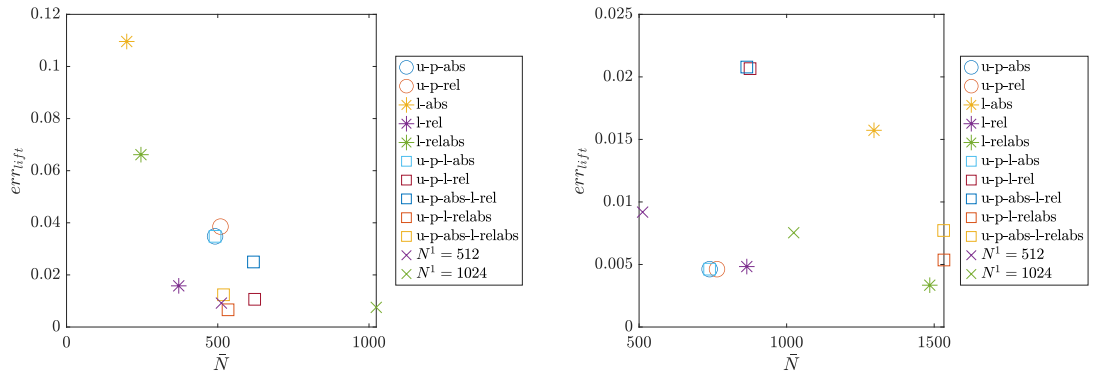
(c) u-p-l-rel. (d) u-p-l-rel.

(e) u-p-abs-l-rel. (f) u-p-abs-l-rel.

(g) u-p-l-relabs. (h) u-p-l-relabs. (i) u-p-l-relabs.

(j) u-p-abs-l-relabs. (k) u-p-abs-l-relabs. (l) u-p-abs-l-relabs.

Figure 6.36: Experimental order of convergence of *EL-LI* controllers of third category using a tolerance of 10^{-3} and $\tau^1 = 0.125$, $Re = 250$.



(a) Without consideration of EOC. (b) With consideration of EOC.

Figure 6.37: Distance errors of lift coefficients over the total number of time steps required without and with considering the order of convergence for several *EL-LI* controllers using a tolerance of 10^{-3} and $\tau^1 = 0.125$, $Re = 250$.

Summary of the numerical studies on the flow around a cylinder benchmark

The incompressible flow studies were performed on the flow around a cylinder benchmark with Reynolds numbers of 100 and 250. First, we looked at the most characteristic quantities of interest, such as the drag and lift coefficients, where we saw that the lift coefficient was more problematic than the drag coefficient due to oscillations. Therefore, we decided to include the error estimator for the lift coefficient in the adaptive strategy. But first we confirmed that the estimated velocity and pressure errors were good approximations of the corresponding reference errors. The lift error estimates did indeed exhibit some of the behavior of the reference errors, but not in terms of magnitude. Therefore, we studied a smaller tolerance and also the drag error estimator, which reasonably approximated the reference error. We assumed that the shift in the lift coefficients arises from vortex shedding, which is typical of this flow problem. Therefore, the resulting phase error could not be approximated using our error estimation method. In addition, several types of the *EL* controllers were tested in terms of the accuracy measured in the distance errors of the lift coefficient from reference lift values (6.2.1) for global-in-time adaptive time-stepping based on different variables such as velocity, pressure and lift. Especially for smaller tolerances, all resulting errors were small. For larger tolerances, the best controllers include velocity and pressure error estimates and relative or relative and absolute error estimates for the lift coefficient such as, the *u-p-abs-l-relabs* and the *u-p-l-relabs*, as well as the *u-p-abs-l-rel* and *l-rel* controllers. Furthermore, for these controllers, the distance error did not change much by varying the tolerance. They also outperformed equidistant time step simulations in terms of accuracy and number of time steps.

Using the flow around a cylinder benchmark with a Reynolds number of 250, we showed that taking into account the experimental order of convergence of the velocity and pressure error estimates, we can achieve high accuracy with respect to the distance error for lift coefficients for all controllers. All in all, the global adaptive strategy with the presented controllers handled the incompressible flows quite well, especially the new pressure error estimator was convincing.

6.2.3. The Carreau-Yasuda model

In this section, we study the Carreau-Yasuda model (2.1.5) with the same parameters as described in Section 2.1.4, where we set $\lambda = 1$. We also study a delayed feedback of the fluid to changes in the shear rate by considering $\lambda = 10$. We perform our investigations at mesh level four, with the same initial mesh as shown in Figure 6.20 in Section 6.2.2. We also use the Q_2/P_1^{disc} element pair. We chose this model problem, because it was studied in [42, 43] to investigate the global-in-time solver.

(a) $\lambda = 1$. (b) $\lambda = 10$.

Figure 6.38: Velocity magnitudes on mesh level 4 and $\tau = 2^{-10}$ for $t = 4$.

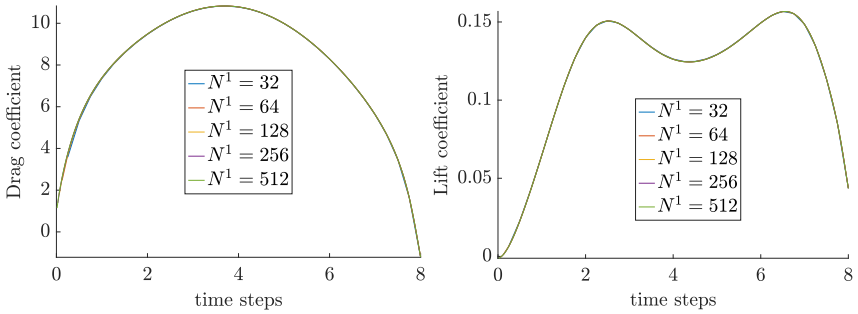
We now concentrate on the study of $\lambda = 1$ and then $\lambda = 10$, where the velocity magnitudes at time $t = 4$ are illustrated in Figure 6.38. The velocity magnitudes are a little higher for $\lambda = 1$ than for $\lambda = 10$. Especially we can see the delayed feedback of the fluid for $\lambda = 10$ compared to $\lambda = 1$.

In the further studies, we will first investigate the quantities of interest for equidistant time step sizes, and then we illustrate whether our controllers in the global adaptive approach, now including the error estimator for the drag coefficient, perform as well or better than simulations with equidistant time step sizes. We study a quantitative evaluation of several controllers including different combinations of velocity, pressure, lift and drag, see Controller 4.5, 4.7 - 4.8, as well as for relative, absolute and relative and absolute error estimators, see, (4.1.2), (4.1.7)-(4.1.12). Finally, we focus on the case with $\lambda = 10$ and show the distance errors of the lift coefficient for the best controllers so far and compare them to runs with equidistant time step sizes.

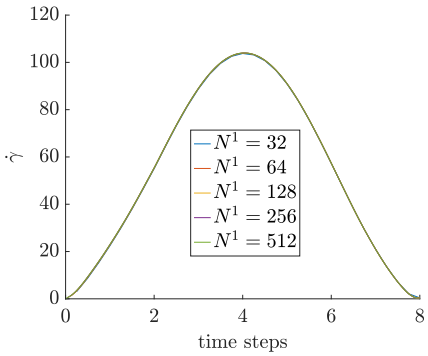
Studies on $\lambda = 1$

The first studies, performed with $\lambda = 1$, show the drag and lift coefficients, as well as the value of $\dot{\gamma}$, in Figure 6.39. In this case, we observe considerable drag values and relatively modest lift values. Even for larger time step sizes, we see no difference for different equidistant time step sizes. In addition, the value of $\dot{\gamma}$ does not appear to vary for larger time step sizes. Since no value oscillates extremely, we will also look at the drag error estimator in this section. But first, we will illustrate the distance errors for equidistant time step sizes.

Figure 6.40 illustrates the distance errors of the drag and lift coefficients for equidistant time step sizes. The drag and lift coefficient errors are quite similar. Moreover, both stagnate for smaller time step sizes. Perhaps, the error is no longer dominated by the temporal discretization, but by the spatial discretization.



(a) Drag coefficient. (b) Lift coefficient.



(c) Values of $\dot{\gamma}$.

Figure 6.39: Drag and lift coefficients as well as pressure difference for equidistant time step sizes, $\lambda = 1$.

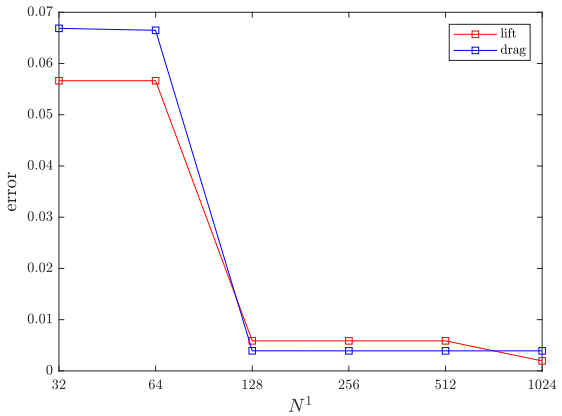


Figure 6.40: Distance errors of drag and lift coefficients for equidistant time step sizes, $\lambda = 1$.

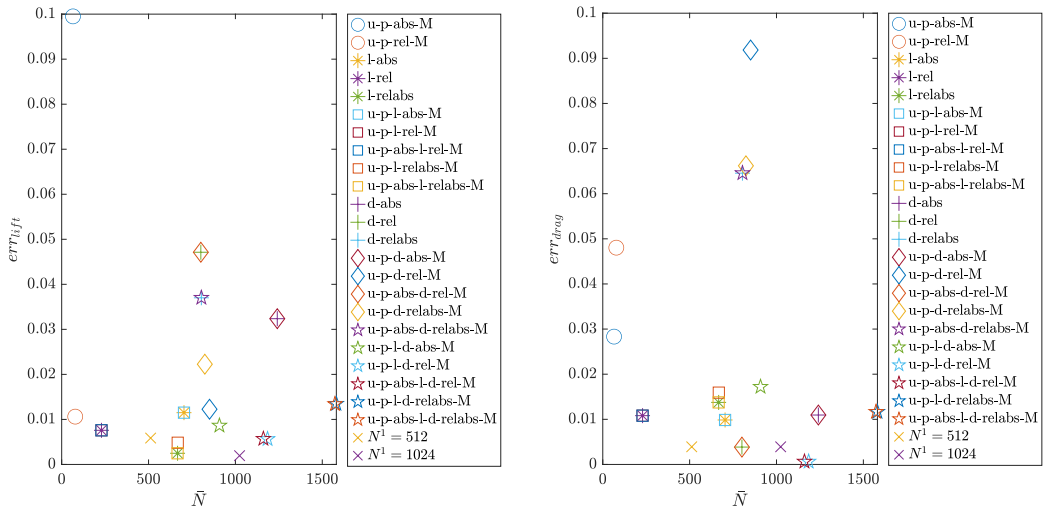
Quantitative evaluation of controllers in the adaptive approach

In this section, we will further consider the error estimator for the drag coefficients, denoted by the suffix *-d*. The error estimator for the drag coefficient is analogous to the estimator for the lift coefficient (4.1.8). Both and additionally the pressure estimator are chosen to be pointwise error estimators and are evaluated at the second point of the Gauss-Legendre formula $t_{n,2}^{G(4)}$. Furthermore, we use the integral-based velocity error es-

timator (4.1.2). In addition, we use the *EL* controller, see Controller 4.2, the tolerance in the stopping criterion is set to 10^{-3} and the initial time step size is set to 0.25.

Considering the error estimators for the drag coefficient leads to the construction of several other controllers based on the previously presented variable controllers, but using the error estimator for the lift coefficient. Obviously, considering the absolute, relative, or absolute and relative drag-only error estimators are the first new controllers. In addition, we combine it with the velocity and pressure error estimators. Finally, we perform a combination of all four error estimators.

Here we consider the *maximum case conditions*, where we select the larger error estimator of velocity and pressure per time step, as in the Controller 4.6. Therefore, the suffix *-M* is added. If the controller is based on only one variable, we omit the suffix.



(a) Distance errors of lift coefficients.

(b) Distance errors of drag coefficients.

Figure 6.41: Distance errors of drag and lift coefficients for different *EL-LI-M* controllers using a tolerance of 10^{-3} and $\tau^1 = 0.25$, $\lambda = 1$.

The distance errors for a multiple number of controllers and their required total number of time steps is given in Figure 6.41, where Figure 6.41a shows the error of the lift coefficient and Figure 6.41b shows the error of the drag coefficient. In both figures two distance errors of runs with an equidistant time step size are plotted as yellow and purple crosses. Multiple controllers give comparable and also better results to the equidistant simulations. To be able to make more precise statements about the controllers, we provide a quantitative evaluation, similar to [36], in the following tables.

We consider score classes for the lift coefficient distance error, the drag coefficient distance error, and the total number of time steps required during all adaptive iterations. These score classes range from very good to very weak on a scale of $\{-2,-1,0,1,2\}$. The specific distribution of the scores is shown in the Table 6.15. In addition, these scores are multiplied by weighting factors, since different objectives will score a controller differently. For example, focusing on the lift coefficient leads to different favorable controllers than focusing on the drag coefficient. Also, in our global-in-time adaptive approach, the number of time steps is not as important as the accuracy. Therefore, it is weighted less.

We could also add the pressure difference, but all the controllers showed the same small magnitude of pressure differences. So this was needless.

Scores	err_{lift}	err_{drag}	$\bar{N}$
2	$5e-3$	$1e-3$	2· best
1	$1e-2$	$5e-3$	4· best
0	$5e-2$	$1e-2$	8· best
-1	$1e-1$	$5e-2$	16· best
-2	$> 5e-1$	$> 1e-1$	>16· best

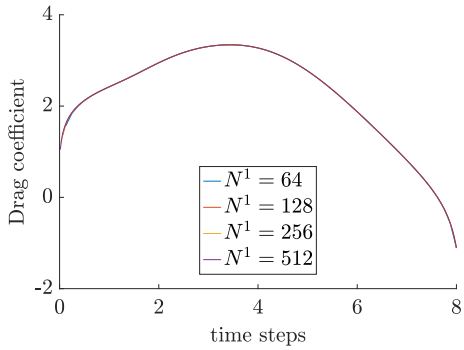
Table 6.15: Score classes used in the quantitative evaluation of controllers. To obtain the scores, the distance errors should be less than or equal to the specified boundary.

	err_{lift}	err_{drag}	$\bar{N}$	total points
weights / controller	2	2	1	
u-p-abs-M	-2	-2	2	-2
u-p-rel-M	0	-2	2	0
l-abs	0	0	0	0
l-rel	2	-4	1	-1
l-relabs	4	-4	0	0
u-p-l-abs-M	0	0	-1	-1
u-p-l-rel-M	2	-4	1	-1
u-p-abs-l-rel-M	2	-4	0	-2
u-p-l-relabs-M	4	-4	0	0
u-p-abs-l-relabs-M	4	-4	0	0
d-abs	0	-2	-2	-4
d-rel	0	2	-1	1
d-relabs	0	-4	-1	-5
u-p-d-abs-M	0	-2	-2	-4
u-p-d-rel-M	0	-4	-1	-5
u-p-abs-d-rel-M	0	2	-1	1
u-p-d-relabs-M	0	-4	-1	-5
u-p-abs-d-relabs-M	0	-4	-1	-5
u-p-l-d-abs-M	2	-4	-1	-3
u-p-l-d-rel-M	2	4	-2	4
u-p-abs-l-d-rel-M	2	4	-2	4
u-p-l-d-relabs-M	0	-2	-2	-4
u-p-abs-l-d-relabs-M	0	-2	-2	-4

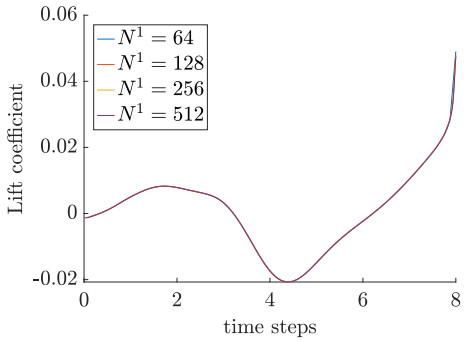
Table 6.16: Quantitative evaluation of the *EL-LI-M* controllers using a tolerance of 10^{-3} and $\tau^1 = 0.25$.

In Table 6.16, we consider weighting factors of the form $\{2,2,1\}$, where the accuracy of the lift and drag coefficients are weighted with two and the number of time steps with

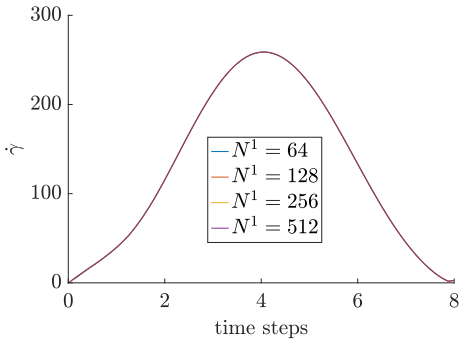
a factor of one. We determine that the lift coefficient is to be given the same importance as the drag coefficient. The second column shows the scores of the controllers concerning the lift coefficients distance errors and the third column shows the scores of the controllers concerning the drag coefficients distance errors. The controllers that involve a modification of the time step size depending on the relative and absolute estimates for the lift coefficient and the controllers depending on the velocity, pressure and lift variables are shown in the second column to exhibit optimal performance. Considering the results for the distance errors for the drag coefficients as well as for the total results, it can be seen that the best performance is achieved by the controller that depends on all four variables. The simulations that include all relative estimator and additional variables such as the absolute value of velocity and pressure as well as relative lift and drag show optimal performance. Including the velocity and pressure error estimators is always preferable, regardless of whether greater interest lies in the lift, drag or both coefficients. If the lift coefficient is deemed the most significant factor, then the lift error estimator should be incorporated, in a manner analogous to the drag coefficient. In conclusion, the controllers behave according to the specifications for their intended functionality.



(a) Drag coefficient.



(b) Lift coefficient.

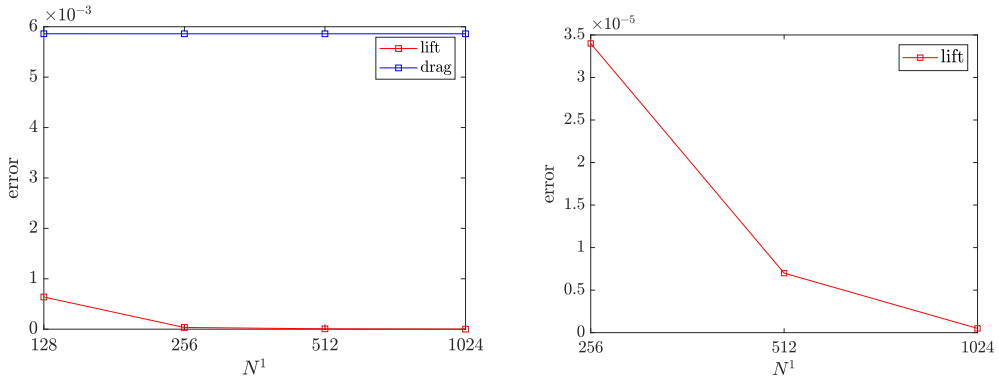


(c) $\dot{\gamma}$.

Figure 6.42: Drag and lift coefficients as well as $\dot{\gamma}$ values for equidistant time step sizes, $\lambda = 10$.

Studies on $\lambda = 10$

We start with the quantities of interest on the Carreau-Yasuda model with $\lambda = 10$ in Figure 6.42. A quick examination of all variables shows that there are no significant differences between larger and smaller time step sizes. No oscillations of the values are observed. Moreover, the drag and lift coefficients are reduced, while the values of $\dot{\gamma}$ are increased compared to the values observed in the test problem $\lambda = 1$, see Figure 6.39.



(a) Distance errors of drag and lift coefficients. (b) Distance errors of lift coefficients.

Figure 6.43: Distance errors of drag and lift coefficients for equidistant time step sizes, $\lambda = 10$.

Figure 6.43 shows the distance errors of the drag and lift coefficients. The errors of the drag coefficients stagnate, while those of the lift coefficients show a decreasing trend. Therefore, we look at the distance errors of the lift coefficients in more detail in Figure 6.43b. The errors continue to decrease and are very small, since for small time steps they have a size of 10^{-6}. Thus, we will finally study the distance errors for the lift coefficients for the error estimators based on the variables velocity, pressure and lift value.

Adaptive studies

The error estimator for the drag coefficient, the estimator for the lift coefficient (4.1.8) and the pressure estimator are chosen to be a pointwise error estimator and are evaluated at the second point of the Gauss-Legendre formula $t_{n,2}^{G(4)}$. Furthermore, we use the integral-based velocity error estimator (4.1.2). In addition, we use the *EL-M* controller 4.6 with the *maximum case conditions*, setting the tolerance in the stopping criterion to 10^{-3} and the initial time step size is 0.125.

The distance errors of the lift coefficients for their total number of time steps during all adaptive iterations from the use of different controllers are shown in Figure 6.44. Here, three controllers in the global adaptive strategy outperform the equidistant time step simulations, namely the *u-p-l-rel-M*, the *u-p-abs-l-rel-M*, and the *u-p-l-relabs-M* controllers. They result in a smaller number of time steps by providing significant accuracy, thus ensuring the main objective of the global adaptive strategy.

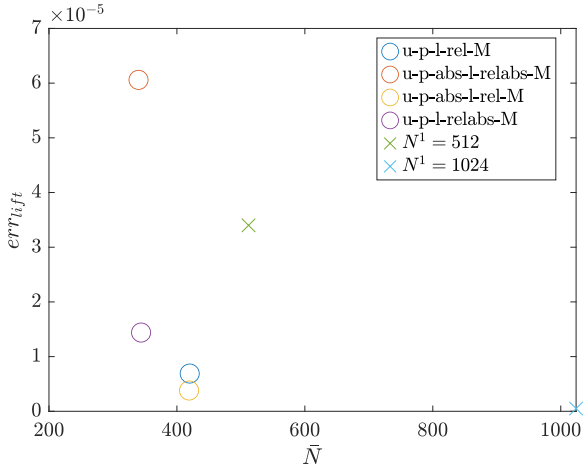


Figure 6.44: Distance errors of lift coefficients for different controller using a tolerance of 10^{-3} and $\tau^1 = 0.125$, $\lambda = 10$.

Summary of the numerical studies on the Carreau-Yasuda model

Here, we focused on the research of $\lambda = 1$ and $\lambda = 10$. We confirmed that our controllers in the global adaptive approach, now including the error estimator for the drag coefficient, also outperform simulations with equidistant time step sizes. We performed a quantitative evaluation of several controllers for $\lambda = 1$, including several variants of velocity, pressure, lift and drag as well as for relative, absolute and relative and absolute error estimators. To sum up, the best choice of controller naturally depends on the quantities of interest. The inclusion of the velocity and pressure error estimator is always preferable, regardless of the greater interest in lift, drag, or both coefficients. But of course, if the lift coefficient is the most interesting quantity, the lift error estimator should be included, analogously to the drag coefficient. Finally, we performed the global adaptive approach with $\lambda = 10$. Here we were able to approximate the most accurate distance errors of the lift coefficient compared to all other tested incompressible flows. As a result, multiple velocity, pressure and lift controllers also outperformed comparable equidistant time step simulations, namely the *u-p-l-rel-M*, the *u-p-abs-l-rel-M*, and the *u-p-l-relabs-M* controllers.

7

Conclusions

7.1. Summary

In this work, we studied an adaptive time step control for a global-in-time Galerkin-Petrov time discretization of evolution equations. Adaptive time-stepping is a useful tool for improving stability, controlling error, and increasing accuracy. When combined with higher order methods, such as the higher order Galerkin-Petrov method, we improve the accuracy and ensure that the solution characteristics are adapted in a more sophisticated way, which is particularly important for incompressible flows. Although adaptive approaches lead to optimized computational complexity, Galerkin-Petrov methods are computationally expensive. To improve the efficiency, it is recommended to apply a global-in-time approach, especially in the field of Computational Fluid Dynamics.

The mathematical framework was introduced in Chapter 2. We considered several evolution equations and defined the function spaces needed in the Galerkin-Petrov time discretization. We also presented the spatial discretization used in this work. We applied biquadratic finite elements to solve the heat equation and the velocity field of the Navier-Stokes equations, while the pressure field was approximated by discontinuous linear finite elements on a quadrilateral mesh. Furthermore, the Gauss-Lobatto quadrature formula is provided, which is needed to numerically compute the resulting integrals in the ansatz space of a variational formulation of the time-stepping scheme.

The continuous Galerkin-Petrov methods as well as the C^1-continuous version were studied in Chapter 3 for a nonlinear ODE. It was shown that the *cGP-C*$1(k+1)$-method is derived from the *cGP*(k)-method by a post-processing step. In particular, the always linear post-processing step for the Navier-Stokes equations was revealed, which includes not only a more accurate velocity solution, but also a more accurate pressure solution. The resulting block systems for each time step to be solved by the *cGP*(2)-method and the post-processing step were provided once for the heat equation and once for the Navier-Stokes equations. Since the Gauss-Lobatto formula was applied, the unknown initial pressure condition appears on the right side of the block systems. Here, we chose this unknown to be the known post-processed pressure solution. To handle the nonlinear convective term,

we explained a linearization by fixed-point iteration.

In Chapter 4, we illustrated the adaptive time step control. The required error estimators resulting from the $cGP(2)$ solution and the higher order post-processed solution were presented. In particular, a new pressure error estimator was established. It is constructed by a cubic Lagrangian interpolation of the post-processed pressure solution. In addition, a new error estimation for the drag and lift coefficients was established. In contrast to classical adaptive time step control, where one time step after another is evaluated successively, we demonstrated a global strategy. By performing adaptive iteration, where the time grid is completely reconstructed in each iteration, we provided a new adaptive control strategy that is suitable in a global-in-time approach. This goes hand in hand with other extensions, such as an interpolation strategy to obtain error estimates and time step sizes for each time in the time grid. Furthermore, several controllers were provided that fit into this global adaptive context.

Two existing global-in-time approaches were demonstrated in Chapter 5. Furthermore, we explored this topic in more depth by looking at how the new adaptive $cGP(2)$ time step control fits in. Due to the post-processing step, the initial values for velocity and pressure at each time step are obtained such that a global-in-time approach can be used effectively.

The numerical studies on the heat equation, on the Navier-Stokes equations and on incompressible flows were presented. We chose a small spatial mesh size for all tests in order to focus on the time error. Starting with the heat equation in Section 6.1, we verified the accuracy of the $cGP(2)$ and $cGP\text{-}C1(3)$-methods by noting that the $cGP(2)$-method achieved an experimental order of convergence of three in the integral-based L^2-error norm and order four in the discrete L^∞-norm. The $cGP\text{-}C1(3)$-method achieves an order of four in both error norms. This confirms the accuracy of the error estimator. Moreover, we studied several adaptive strategies in combination with different controllers, where the *elementary local error controller* applied in the *linear interpolation strategy* showed the best results in terms of accuracy and number of time steps. Furthermore, the adaptive time step control was compared to a run with equidistant time step sizes and showed an advantage in terms of computational complexity, also measured in terms of the number of time steps.

The choice of the post-processed pressure solution as the initial pressure condition at each time step in the $cGP(2)$-method was confirmed by considering the Navier-Stokes equations in Section 6.2.1. Along with verifying the experimental order of convergence of three in the L^2-error norm for the pressure variable in the $cGP\text{-}C1(3)$-method as well as order four in the discrete L^∞-norm. Most interesting was the experimental order of convergence of four for the new higher order cubic interpolated post-processed pressure. In addition, the new pressure error estimator was validated and approximated the analytical pressure error very well.

The incompressible flow studies were performed on the flow around a cylinder benchmark with Reynolds numbers of 100 and 250 in Section 6.2.2. Starting with the a Reynolds number of 100, we verified that the velocity error estimates and the pressure error estimates sufficiently approximated the associated reference errors. The lift and drag error estimators did indeed show some the behavior of the reference errors, but not in magnitude. In addition, several controllers were tested for global-in-time adaptive time-stepping

based on different variables such as velocity, pressure, and lift. In particular, for smaller tolerances, all resulting distance errors were small. For larger tolerances, the best controllers include velocity and pressure error estimates and relative or relative and absolute error estimates for the lift coefficient. Furthermore, for these controllers, the distance error did not change much by varying the tolerance. They also showed better performance in terms of accuracy and number of time steps than simulations with equidistant time steps. Using the flow around a cylinder benchmark with a Reynolds number of 250, we have shown that, considering the experimental order of convergence of the velocity error estimates, we can achieve high accuracy in terms of distance errors for the lift coefficients for all controllers.

Another incompressible flow was studied by the Carreau-Yasuda model with $\lambda = 1$ and $\lambda = 10$ in Section 6.2.3. We confirmed that our controllers in the global adaptive approach, now including the error estimator for the drag coefficient, also outperform simulations with equidistant time step sizes for non-Newtonian flows. In addition, we performed a quantitative evaluation of several controllers and summarized that the best choice of controller naturally depends on the quantities of interest. The inclusion of the velocity and pressure error estimator is always preferable, regardless of the higher interest in lift, drag, or both coefficients. Finally, we performed the global adaptive approach with $\lambda = 10$. Here we were able to approximate the most accurate distance errors of the lift coefficient compared to all other incompressible flow test cases. As a result, multiple velocity, pressure, lift and drag controllers also outperformed comparable equidistant time step simulations.

7.2. Outlook

This thesis studied a global adaptive time step control based on a higher order time discretization scheme. While we focused on the main goals of adaptive time step control by investigating the computational effort, measured in the required number of time steps, and verifying the accuracy of error estimators, adaptive strategies involving multiple controllers, there are still open research areas.

The first step in the future is the implementation of a global-in-time solver for our higher order global adaptive strategy. With this approach, we are able to solve the previous studies more efficiently and with higher accuracy. In particular, incompressible flows would benefit from considering smaller tolerances in the adaptive approach and applying a significantly smaller spatial mesh, which is only possible with a parallel simulation due to the high computational cost of the $cGP(2)$-method. Along with further research on the impact on global-in-time solvers. Since the application of adaptive time step controls has the further advantage of balancing stability, it is not out of the question that the solvers could benefit from this, especially in unstable regions.

Further studies on the error estimator for the lift coefficients are recommended. We saw in the numerical studies that the approximation of the errors of the lift coefficients by the associated error estimators could be improved. While the inclusion of an error estimator for the lift coefficients has made significant progress, further research could lead to notable advances in the controllers.

Bibliography

[1] N. Ahmed, S. Becher, G. Matthies. Higher-order discontinuous Galerkin time stepping and local projection stabilization techniques for the transient Stokes problem. *Comput. Methods Appl. Mech. Eng. 313(1)*, 28–52, 2017

[2] N. Ahmed, V. John. Adaptive time step control for higher order variational time discretizations applied to convection–diffusion–reaction equations. *Comput. Methods Appl. Mech. Engrg.*, 285:83–101, 2015.

[3] N. Ahmed, G. Matthies. Higher order continuous Galerkin-Petrov time stepping schemes for transient convection-diffusion-reaction equations. *ESAIM Math. Model. Numer. Anal. 49(5)*, 1429–1450 , 2015

[4] N. Ahmed, G. Matthies. Numerical Studies of Higher Order Variational Time Stepping Schemes for Evolutionary Navier-Stokes Equations. *Boundary and Interior Layers,Computational and Asymptotic Methods BAIL*, 2016.

[5] N. Ahmed, G. Matthies. Numerical studies of variational-type time-discretization techniques for transient Oseen problem. *In: Algoritmy 2012. 19th Conference on Scientific Computing*,Vysoké Tatry, Podbanské, Slovakia, September 9–14, 2012. Proceedings of Contributed Papers and Posters. Slovak University of Technology, Faculty of Civil Engineering, Department of Mathematics and Descriptive Geometry, Bratislava, pp. 404–415, 2012

[6] V. AMP, C. GF, C. ALGA. Control strategies for timestep selection in finite element simulation of incompressible flows and coupled reaction-convection-diffusion processes. *International Journal for Numerical Methods in Fluids*,47(3): 201-231,514–524, 2005

[7] M. Anselmann, M. Bause, S. Becher, G. Matthies. Galerkin-collocation approximation in time for the wave equation and its post-processing. *ESAIM Math. Model. Numer. Anal., 54*, pp. 2099–2123, 2020

[8] A. K. Aziz and P. B. Monk Continuous finite elements in space and time for the heat equation *Mathematics of Computation Vol. 52*, 255-274 , 1989

[9] S. Bailoor, J. H. Seo, R. Mittal. Vortex shedding from a circular cylinder in shear-thinning Carreau fluids. *Physics of Fluids*, doi:10.1063/1.5086032, 2019

[10] M. Besier, R. Rannacher. Goal-oriented spacetime adaptivity in the finite element Galerkin method for the computation of nonstationary incompressible flow. *Numer. Methods Fluids 70 (9)*, 1139–1166. 2012

[11] S.C. Brenner, L.R. Scott, The Mathematical Theory of Finite Element Methods. *Texts in Applied Math- ematics*, vol. 15, 3rd edn. Springer, New York, 2008

[12] D. Boffi and L. Gastaldi. On the quadrilateral Q_2 -P_1 element for the Stokes problem. *International Journal for Numerical Methods in Fluids*, 39:1001–1011, 2002.

[13] JC. Butcher. Numerical Methods for Ordinary Differential Equations. *2nd edn., John Wiley Sons, Ltd*, doi:10.1002/9780470753767. URL http://dx.doi.org/10.1002/9780470753767, 2008

[14] W. Cao, L. Jia, and Z. Zhang. A C^1 Petrov-Galerkin method and Gauss collocation method for 1D general elliptic problems and superconvergence. *DCDS-B 26*, pp. 81-105, 2021

[15] P.J. Carreau. Rheological equations from molecular network theories. *Trans. Soc. Rheol. 16(1)*, 99–127, 1972

[16] C. Cuvelier, A. Segal, A. Steenhoven. Finite element methods and Navier-Stokes equations. *Mathematics and its applications*, 1986

[17] F. Danieli, B. S. Southworth, and A. J. Wathen. Space-Time Block Preconditioning for Incompressible Flow. *In: SIAM Journal on Scientific Computing 44.1*,A337–A363, 2022

[18] J. Dünnebacke, S. Turek, P. Zajac, A. Sokolov. A time-simultaneous multigrid method for parabolic evolution equations. *Technical report. TU Dortmund: Fakultät für Mathematik.*, DOI:10.17877/ DE290R-20382. Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 619, 2019

[19] J. Dünnebacke, S. Turek, C. Lohmann, A. Sokolov, P. Zajac. Increased space-parallelism via time-simultaneous Newton-multigrid methods for nonstationary nonlinear PDE problems *In: The International Journal of High Perfor- mance Computing Applications (2021)*, url: https://doi.org/10.1177/10943420211001940., 2021

[20] L.C. Evans Partial Differential Equations. *vol. 19. Graduate Studies in Mathematics* , American Mathematical Society, Providence, RI, 1998

[21] R. D. Falgout, S. Friedhoff, TZ. V. Kolev, S. P. MacLachlan, J. B. Schroder. PARALLEL TIME INTEGRATION WITH MULTIGRID. *SIAM Journal on Scientific Computing*, 36, No.6, pp. C635–C661, 2014

[22] M. J. Gander. 50 years of Time Parallel Time Integration. *Multiple Shooting and Time Domain Decomposition Methods*, pp.69–113, Springer, 2015

[23] MJ. Gander, M. Neumüller. Analysis of a new space- time parallel multigrid algorithm for parabolic problems. *SIAM Journal of Scientific Computing*, 38(4): A2173–A2208, 2016

[24] V. Girault, P.-A. Raviart. Finite Element Methods for Navier–Stokes equations. *Springer-Verlag*, Berlin–Heidelberg–New York, 1986

[25] .M. Gresho, R.L. Sani. Incompressible Flow and the Finite Element Method. *Wiley, Chichester,*, Berlin–Heidelberg–New York, 2000

[26] W. Hackbusch. Parabolic multi-grid methods. *In Computing Methods in Applied Sciences and Engineering*, VI, R. Glowinski and J.-L. Lions, eds., North-Holland, 1984, pp. 189–197, 1984

[27] E. Hairer and G. Wanner. Solving ordinary differential equations. *II, volume 14 of Springer Series in Computational Mathematics.Springer-Verlag, Berlin*, Stiff and differential-algebraic problems, Second revised edition, paperback, 2010

[28] J.K. Hunter. Nonlinear Evolution Equations. *Department of Mathematics. University of California Davis*, https://www.math.ucdavis.edu/~hunter/notes/nonlinev.pdf, 1996

[29] S. Hussain, F. Schieweck, S. Turek. A note on accurate and efficient higher order Galerkin time stepping schemes for the nonstationary Stokes equations. *Open Numer. Methods J. 4*, 35–45, 2012

[30] S. Hussain, F. Schieweck, S. Turek. An efficient and stable finite element solver of higher order in space and time for nonstationary incompressible flow. *Int. J. Numer. Methods Fluids 73(11)*, 927–952, 2013

[31] S. Hussain, F. Schieweck, S. Turek. Efficient Newton-multigrid solution techniques for higher order space-time Galerkin discretizations of incompressible flow. *Appl. Numer. Math. 83*, 51–71, 2014

[32] S. Hussain, F. Schieweck, S. Turek. Higher order Galerkin time discretizations and fast multigrid solvers for the heat equation. *Journal of Numerical Mathematics*, 19(1):41–61, 2011

[33] S. Hussain, F. Schieweck, S. Turek. Higher order Galerkin time discretization for nonsta- tionary incompressible flow. *Numerical Mathematics and Advanced Applications 2011*, pp. 509–517. Springer, Heidelberg, 2013

[34] S. Hussain, F. Schieweck, S. Turek, P. Zajac. On a Galerkin discretization of 4th order in space and time applied to the heat equation. *International Journal of Numerical Analysis and Modeling*, Series B, Volume 4, Number 4, 353–371 , 2012

[35] V. John, G. Matthies, J. Rang. A comparison of time-discretization/linearization approaches for the incompressible Navier–Stokes equations. *Comput. Methods Appl. Mech. Engrg., 195 (44–47)*, pp. 5995-6010, 2006

[36] V. John, J. Rang. Adaptive time step control for the incompressible Navier–Stokes equations. *Comput. Methods Appl. Mech. Engrg. 199*,(9–12),514–524, 2010

[37] V. John, G. Matthies. Higher Order Finite Element Discretizations in a Benchmark Problem for Incompressible Flows. *vol. 37. International Journal for Numerical Methods in Fluids*, doi:10.1002/fld.195, 2001

[38] V. John. Reference value for drag and lift of a two-dimensional time dependent flow around cylinder. *International Journal for Numerical Methods in Fluids, Vol. 44*,p. 777 - 788, doi:10.1002/fld.679, 2004

[39] J. Lang. Adaptive Multilevel Solution of Nonlinear Parabolic PDE Systems: Theory, Algorithm and Applications. *in: Lecture Notes in Computational Science and Engineering*,vol. 16, Springer-Verlag, Berlin, 2001

[40] E. Lelarasmee, A. E. Ruehli and A. L. Sangiovanni-Vincentelli The Waveform Relaxation Method for Time-Domain Analysis of Large Scale Integrated Circuits. *IEEE Trans. on CAD of IC and Syst., 1*, pp. 131–145 , 1982

[41] J.-L. Lions, Y. Maday, G. Turinici. A "Parareal" in time discretization of PDE's. *M C. R. Math. Acad. Sci. Paris*, 332, no. 7, pp. 661-668, 2001

[42] C. Lohmann, S. Turek. Augmented Lagrangian Acceleration of Global-in-Time Pressure Schur Complement Solvers for Incompressible Oseen Equations. *J. Math. Fluid Mech.*, 26, 27, https://doi.org/10.1007/s00021-024-00862-7, 2024

[43] C. Lohmann, S. Turek. On the Design of Global-in-Time Newton-Multigrid-Pressure Schur Complement Solvers for Incompressible Flow Problems. *J. Math. Fluid Mech. 25*, 64, https://doi.org/10.1007/s00021-023-00807-6, 2024

[44] C. Lubich, A. Ostermann. Multi-grid dynamic iteration for parabolic equations. *BIT Numerical Mathematics 27(2):*, 216–234, 1987

[45] N. Margenberg, P. Munch. A Space-Time Multigrid Method for Space-Time Finite Element Discretizations of Parabolic and Hyperbolic PDEs. *https://arxiv.org/abs/2408.04372*, 2024

[46] G. Matthies, F. Schieweck. Higher order variational time discretizations for nonlinear systems of ordinary differential equations. *Preprint 23/2011*, Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, 2011

[47] G. Matthies and L. Tobiska. The inf-sup condition for the mapped Q_k P_{k1}^{disc} element in arbitrary space dimensions. *Computing*, 69:119–139, DOI: 10.1007/s00607-002-1451-3. URL http://dl.acm.org/citation.cfm?id=635904.635906, 2002

[48] M. L. Minion. hybrid parareal spectral deferred corrections method. *Communications in Applied Mathematics and Computational Science*,5 (2010), pp. 265–301, 2012

[49] B.E. Rapp. Microfluidics: Modeling, Mechanics and Mathematics. *Vol. 1, Elsevier,* Cambridge, 2016

[50] J. Rauch. Partial Differential Equations. *Springer-Verlag*,New York, 1991

[51] F. Schieweck. A-stable discontinuous Galerkin-Petrov time discretization of higher order. *J. Numer. Math.*,,18(1):25 – 57, 2010

[52] M. Schäfer, S. Turek. Benchmark Computations of Laminar Flow Around a Cylinder. *E. Hirschel (Ed.), Flow Simulation with High-Performance Computers II, Vol. 52 of Notes on Numerical Fluid Mechanics*, Vieweg, pp. 547–566, 1996

[53] D. Schötzau, C. Schwab. Time discretization of parabolic problems by the hp-version of the discontinuous Galerkin finite element method. *SIAM J. Numer.Anal. 38 (3)*,837–875. 2000

[54] C. Seifert, S. Trostorff, M. Waurick. Evolutionary equations—Picard's theorem for partial differential equations, and applications. *volume 287 of Operator The- ory: Advances and Applications. Birkh¨auser/Springer*, Cham,doi:10.1007/ 978-3-030-89397-2., 2022

[55] Y. -h. Shih, G. Stadler, and F. Wechsung. Robust Multigrid Techniques for Augmented Lagrangian Preconditioning of Incompressible Stokes Equations with Extreme Viscosity Variations. *In: SIAM Journal on Scientific Computing 45.3*, S27–S53, 2023

[56] G. Strang, G. J. Fix. Analysis of the Finite Element Method. *Prentice Hall*, 1973

[57] G. Söderlind. Automatic control and adaptive time-stepping. *Numerical methods for ordinary differential equations*,Numer. Algorithms 31(1-4), 281–310 (2002) https://doi.org/10.1023/A:1021160023092, 2001

[58] V. Thomée. Galerkin finite element methods for parabolic problems. *volume 25 of Springer Series in Computational Mathematics*,Springer-Verlag, Berlin, second edition, 2006

[59] K. Yasuda. Investigation of the analogies between viscometric and linear viscoelastic properties of polystyrene fluids. *PhD Thesis*, Massachusetts Institute of Technology, 1979

[60] K. Yasuda, R.C. Armstrong, R.E. Cohen. Shear flow properties of concentrated solutions of linear and star branched polystyrenes. *Rheol. Acta 20*, 163–178 , 1981