# Numerical analysis of new class of higher order Galerkin time discretization schemes for nonstationary incompressible flow problems

---

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften

Der Fakultät für Mathematik der
Technischen Universität Dortmund
vorgelegt von

Shafqat Hussain

Numerical analysis of new class of higher order Galerkin time discretization schemes for nonstationary incompressible flow problems

Shafqat Hussain

Dissertation eingereicht am: 02. 02. 2012
Tag der mündlichen Prüfung: 22. 03. 2012

Mitglieder der Prüfungskommission:

Prof. Dr. Stefan Turek (1. Gutachter, Betreuer)
Prof. Dr. Friedhelm Schieweck (2. Gutachter)
Prof. Dr. Christian Meyer
Prof. Dr. Rudolf Scharlau
Dr. Matthias Möller

*To my lovely daughter*
*Zaina Shafqat*

# Acknowledgements

With the name of Allah, the Most Gracious, Most Compassionate and Ever Merciful, Who gave me the power to do the right, the wisdom to think, observe, judge and analyze. It is He, Who has blessed me in every way and made me capable of accomplishing this thesis successfully.

First and foremost, I would like to express my heartiest gratitude to Prof. Dr. Stefan Turek, my supervisor, for providing me the opportunity of my Ph.D under his supervision. I deeply appreciate for his guidance, encouragement, advice, ideas, and invaluable support, throughout the duration of this thesis. His keen scientific insights were indispensable for this thesis and will always be a source of inspiration for my future career.

Second, I am extremely grateful to Prof. Dr. Friedhelm Schieweck from the university of Magdeburg, for providing me the opportunity to develop and analyze the underlying methods that finally lead to this thesis. He provided me greatest knowledge about theory and implementation of higher order time discretizations. He patiently introduced me the basic knowledge of the topic and always helped me whenever I needed. I am also thankful to him for accepting to review my thesis.

Third, I am very much thankful to Dr. Michael Köster who always helped me during my study at TU Dortmund. His remarks and suggestions to my thesis are gratefully acknowledged.

My sincere thanks goes to all my colleagues at Institut für Angewandte Mathematik und Numerik LS III, TU Dortmund, for providing me the friendly environment during these last years . In particular, I would express my appreciation to S. Buijssen and C. Becker for administrative support. I would like to thank Dr. D. Kuzmin and Dr. M. Möller for offering series of lectures on computational fluid dynamics (CFD) related topics and their willingness for discussions. My special thanks goes to Dr. A. Ouazzi and R. Mahmood for their fruitful discussions from time to time.

I gratefully acknowledge the funding source that made my Ph.D work possible. I was funded by the Higher Education Commission (HEC) of Pakistan.

Finally, I am forever grateful to my family: especially my parents and my wife for their patience and love. Their continuous support and unceasing prayers enabled me to reach the present position in my life. Without their love, time and guidance, I would not be able to make all this possible.

Dortmund, February 02, 2012

Shafqat Hussain

# Contents

# 1

# Introduction

## 1.1. Introduction and Motivation

Many of the real world phenomenons are modeled with time dependent partial differential equations. In seeking the numerical solution of these time dependent problems, the spatial and temporal domains are typically treated separately. A discretization in either of the variables is known as semi-discretization. Discretizing the space variable is known as *spatial discretization* while the discretization in time is usually referred to as *temporal discretization*. Although such problems contain the time dependent terms which are just the derivatives with respect to independent variable (time), one needs to do a special treatment of these terms if we look from the physical point of view.

In order to find the numerical solution of such problems by using the finite element method (FEM), the first step is usually to discretize the spatial variable which leads to a large system of ordinary differential equation (ODEs) for the time dependent nodal vector of the FEM solution. This approach is referred to as method of lines (MOL). In this approach, the space variable is discretized in a first step, while the time variable is still a continuous independent variable. The second step is to solve this system of ODEs by employing a suitable implicit or explicit time discretization. For a very small mesh size in space, the behavior of this ODE-system becomes more and more stiff and creates some numerical complexities. Therefore, explicit time discretizations are not a good choice for such types of problems and implicit methods are required. Usually, the options are restricted to 2nd order backward difference formula (BDF) methods. If we want to have higher order A-stable time discretizations, typical choices are the implicit Runge-Kutta method or the *discontinuous* Galerkin (dG) method with higher order polynomials.

As time dependent simulations are much more time consuming than stationary simulations, the question is to find a time stepping scheme which allows very large time step sizes to gain highly accurate results at the minimum numerical cost. Moreover, higher order methods are often essential in order to achieve accurate results on computationally feasible grids. However, constructing higher order numerical methods maintaining stability and physical constraints becomes increasingly difficult or challenging. To this end, many state of the art numerical integration schemes are available in the literature. A well-known approach to solve time dependent problems is the Galerkin method, see for instance the monograph [55]. In order to obtain a time marching process for this discretization, at least the test space needs to be discontinuous in time. In the *discontinuous* Galerkin method where solution and test space are the same, also the discrete solution space consists of discontinuous piecewise polynomials in time. Therefore, some jump terms appear in this discretization which however can be avoided if a continuous discrete solution space is combined with a discontinuous test space. Then, the method is called a *Galerkin-Petrov method*. In the present thesis, we investigate a new class of time discretizations of 'variational type methods', called *continuous* Galerkin-Petrov (cGP) scheme which is advantageous over the standard time

discretization methods and gains highly accurate results at reasonable numerical cost. Discontinuous Galerkin (dG) time discretizations are also studied and compared with respect to accuracy and numerical efficiency.

The term 'variational type method' refers to a derivation technique based on the finite element method. It has a couple of advantages [40] in comparison to 'traditional methods' like, for example:

- ⊕ A uniform variational approach in space and time is often advantages for the analysis of the fully discrete problem and for the construction of space-time adaptive methods.
- ⊕ It is very natural to construct higher order methods which provide reasonable numerical costs in practice.
- ⊕ Finite element stability concepts of the Galerkin-Petrov or *discontinuous* Galerkin methods can be used to obtain at least A-stable methods.
- ⊕ Fully adaptive finite element techniques can be utilized to change the polynomial degree as well as the length of the time intervals in order to increase accuracy and decrease numerical costs.

The numerical methods which we investigate here are based on Rothe's method. We utilize both *continuous* Galerkin-Petrov (cGP) and *discontinuous* Galerkin (dG) methods to discretize the heat equation in time. Discontinuous Galerkin methods are a class of finite element methods which combine the interesting features of the finite element and the finite volume method to achieve high order accuracy and have recently become popular among computational scientists and engineers. Here, we compare this approach, which we call *continuous Galerkin-Petrov discretization* (cGP($k$)-method), with the well-known *discontinuous Galerkin time discretization* (dG($k$)-method) [55]. For the cGP($k$)-method, the discrete solution space consists of continuous piecewise polynomial functions in time of degree $k \geq 1$ and the discrete test space of discontinuous polynomial functions of degree $k-1$. In the dG($k$)-method, both the solution and test space are constructed by means of discontinuous polynomial functions of degree $k$. With respect to the computational costs, which mainly depend on the size of the resulting block system that has to be solved for each time interval, the cGP($k$)-method is comparable to the dG($k$-1)-method. However, concerning the discretization error in time, the accuracy of the cGP($k$)-method is one order higher than that of the dG($k$-1)-method. Furthermore, all cGP($k$)-methods are A-stable, while all dG($k$)-methods are even "strongly A-stable" (or L-stable), i.e., the dG-methods have better damping properties with respect to high frequency error components.

As a first step, a space-time finite element method is proposed for the solution of the two dimensional heat equation. In particular, we present a numerical study of the higher order time discretizations cGP(1), cGP(2) and dG(1), taking into account aspects of numerical accuracy and efficiency of the corresponding solution methods for the resulting (coupled) linear systems. First of all, the cGP(1)-method is very close to the well-known Crank-Nicolson scheme: Both methods differ only in the choice of the unknown that is solved for on each time interval and in the way the numerical integration of the right hand side is done. The cGP(1)-method is accurate of order 2 in the whole time interval similar to the Crank-Nicolson scheme. However, for the cGP(2)-method as well as in the dG(1)-method, we have two unknowns on each time interval which have to be computed by solving a $2 \times 2$ block system. The cGP(2)-method is accurate of order 3 in the whole time interval and superconvergent of order 4 in the discrete time points [2]. The dG(1)-method is of order 2 in the whole time interval and superconvergent of order 3 in the discrete time points. For all presented time stepping schemes, we apply the standard Galerkin finite element method (FEM) with biquadratic quadrilateral elements for the spatial discretization.

As a next step, we extend our work for the nonstationary Stokes equations in two dimensions. In order to achieve higher order accuracy for the pressure also in the discrete points in time too, which is required to compute the hydrodynamic forces in CFD problems such as drag, lift. etc, we make use of the Lagrangian interpolation polynomials in the cGP-method. In contrast to the dG(1)-method, we cannot obtain the pressure in cGP-methods at the discrete time points by using the same extrapolation as for velocity since this would involve the initial pressure which we do not have. The same technique is then also applied for the dG(1)-method which gives better results than the associated extrapolation. By means of numerical experiments we compare the different time discretizations w.r.t. accuracy and computational costs and we show that the convergence behavior of the corresponding multigrid method is almost independent of the mesh size and time step, leading to an efficient solution process.

Finally, we extend our numerical study for the nonstationary Navier-Stokes equation which was our actual aim. We perform nonstationary simulations to analyze the temporal accuracy and efficiency of the presented time discretization schemes for the incompressible Navier-Stokes equations. All the computational aspects are discussed regarding the Navier-Stokes equations. As a first test problem, we consider the *flow around cylinder* which exactly corresponds to the classical flow around cylinder benchmark [61]. Here, we will concentrate only on the nonstationary behavior of the flow pattern with periodic oscillations and examine the ability of different time discretization schemes to capture the dynamics of the flow. As a second test case, we consider the nonstationary flow for a higher Reynolds number through a venturi pipe which has many real life and industrial applications, for instance, this venturi pipe can be used as a small device in sailing boats. The test configuration for the flow through a venturi pipe which is considered here is slightly changed from the framework which was already used in [57, 58]. The objective of this simulation is to analyze the instantaneous and mean flux through this device.

The spatial discretization in the case of the Stokes or Navier-Stokes equations is carried out by using biquadratic finite elements for velocity and discontinuous linear elements for pressure. We discuss implementational aspects as well as methods for solving the resulting block systems with monolithic multigrid solvers based on a local pressure Schur complement smoothers. The associated system of nonlinear equations which employs saddle point character, is treated by using the nonlinear fixed point and Newton method.

## 1.2. Thesis Contributions

Looking in the literature, it is found that the approach of *continuous* Galerkin has been used by Aziz and Monk in 1989 for the linear heat equation [2]. They have proved optimal error estimates as well as superconvergence results in the end points of the time intervals. However, their approach is different to our approach and uses Gauß-Legendre integration. In [49], the cGP($k$)-method was analyzed by F. Schieweck for the linear parabolic equation (but not under this name) in the abstract Hilbert-space settings as well as for a nonlinear ODE-system. There, the *continuous* Galerkin-Petrov method (cGP($k$)-method) has been called *discontinuous* Galerkin-Petrov (dGP($k$)-method). The construction and analysis of the *discontinuous* Galerkin (dG($k$)-method) time discretization is well-known, see for instance, in [55] and [20].

The main contribution of this thesis is to investigate and analyze a new class of variational type time discretization scheme for the nonstationary simulation of CFD problems. This scheme is known as *continuous* Galerkin-Petrov method (cGP($k$)-method). This class of cGP($k$)-methods was originally developed in [49] and analyzed for the ODE-system under the name '*discontinuous* Galerkin-Petrov method (dGP($k$)-method)'. We start with a numerical analysis of the cGP($k$)-method for the heat equation in two dimensions. Another well known class of *discontinuous* Galerkin (dG($k$)-method) time discretization is also studied for the heat equation. We perform nu-

merical tests to confirm the theoretical order of convergence. All the presented time discretizations are compared w.r.t. numerical efficiency. These time discretization schemes are extended for the nonstationary Stokes and Navier-Stokes equations in two dimensions. A special postprocessing technique is introduced in order to get higher order pressure in the discrete time points. Analytical test cases are considered to confirm the theoretical orders of convergence for all presrented time discretizations for the nonstationary Stokes equations. Finally, several nonstationary simulations of benchmarking character are performed to demonstrate the temporal accuracy and efficiency of all the presented time discretizations.

All the computations during the preparation of this thesis are performed by using the finite element solver package FEATFLOW (Finite element analysis tool for flow problems). FEATFLOW is a general purpose open source FEM software package particularly for the simulations of computational fluid dynamics (CFD) problems. A comprehensive introduction can be found at `http://www.featflow.de`. The computational CPU-times are measured on an AMD Opteron 250 at 2.4GHz.

## 1.3. Publications

During the preparation of this thesis, the author has published a number of technical reports at Institut für Angewandte Mathematik, Fakultät für Mathematik, TU Dortmund [27, 29, 31, 33] [1].

### Journal Articles

At the time of writing, two journal articles have been published in the *Journal of Numerical Mathematics* [28] and in *The Open Numerical Methods Journal* [30]. One article has been accepted for publication in *ENUMATH 2011 Proceedings Volume* [32]. Several more articles were submitted for publications by the author of this thesis.

[1] `http://www.mathematik.tu-dortmund.de/lsiii/cms/de/schriften/ergebnisberichte.html`

## 1.4. Thesis Outline

This thesis is organized such that each chapter has its own introduction which summarises its contents. Nevertheless, we provide the brief overview of each chapter and sections in a more detailed outline.

**Chapter 2** provides an overview about various well known time discretization schemes and discusses two important approaches for the numerical solution of partial differential equations. Different aspects for explicit and implicit methods are also discussed. Section 2.2 explains the stability of the time discretizations schemes which are discussed in this chapter.

**Chapter 3** explains the fundamentals of the finite element method which are used for the spatial discretization used throughout this thesis. Section 3.2 describes the details of the *method of weighted residuals (MWR)* and in Section 3.3 the construction of the underlying finite element spaces is discussed. Section 3.4 contains the descriptions of the biquadratic quadrilateral ($Q_2$) element and the discontinuous linear ($P_1$) element. In Section 3.5, we discuss the properties of well-known LBB-stable finite element pair $Q_2/P_1^{disc}$. At the end of this chapter, we give a brief introduction for edge oriented jump stabilization in Section 3.6.

**Chapter 4** is devoted to the nonlinear and linear solvers. Section 4.1 introduces iterative solvers for sparse linear systems arising from the discretizations of PDEs. We discuss the Bi-Conjugate Gradient Stabilized method (BiCGStab) and the Generalized Minimal Residual Method (GMRES) in Section 4.1.2 as representation of Krylov space methods. Section 4.3 discusses the geometrical multigrid method together with its ingredients.

**Chapter 5** introduces new time discretizations schemes of variational type, namely the continuous Galerkin-Petrov methods cGP($k$) and the discontinuous Galerkin methods dG($k$) method for heat equation in two dimensions. Sections 5.1 and 5.2 explain the theoretical aspects in detail for the cGP($k$) and dG($k$) methods, respectively. Section 5.3 describes the spacial discretization by using the finite element method and the corresponding block-systems are derived for all schemes. Section 5.4 gives the details how the associated linear systems are solved. Finally, we perform several numerical tests to demonstrate the temporal accuracy and efficiency of all the presented schemes in Section 5.5.

**Chapter 6** presents a variant of the continuous Galerkin-Petrov methods cGP($k$) which is based on the Gauß-Lobatto points. Apart from the theoretical details in this chapter we also perform some numerical tests to demonstrate and compare the temporal accuracy of Gauß-Lobatto based cGP($k$)-methods in Section 6.3.

**Chapter 7** discusses some theoretical properties regarding the cGP($k$)-methods. We prove the A-stability and optimal error estimates for the cGP($k$)-methods in Section 7.1 and 7.2, respectively. The stability of the dG($k$) is also discussed in Section 7.3.

**Chapter 8** extends the continuous Galerkin-Petrov methods cGP($k$) and the discontinuous Galerkin methods dG($k$) method for nonstationary Stokes equations. Section 8.1 describes the theoretical aspects in detail for the cGP($k$) and dG($k$) methods, respectively, applied to the nonstationary Stokes equations. In Section 8.2, we explain the spacial discretization which is carried out by using the finite element method and present the corresponding block-systems for all schemes. Section 8.4 gives the details how the associated linear systems are solved. By means of numerical tests, we analyze the temporal accuracy and efficiency of all the presented schemes in Section 8.5.

**Chapter 9** is devoted to the extension of presented time discretizations for the nonstationary Navier-Stokes equations in two dimensions. In Section 9.1, we provide theoretical details for the applications of the cGP($k$) and dG($k$) methods, resp., to the Navier-Stokes equations. Section 9.2 deals with the spacial discretization by using the finite element method, and the corresponding block-systems are presented for all schemes. Section 9.3 describes the solutions techniques for the solution of discretized nonlinear systems which are solved by using the fixed-point and Newton method.

**Chapter 10** presents nonstationary simulations for two benchmarking configurations to analyze the temporal accuracy and efficiency of the presented time discretization schemes. As a first test problem, we consider a classical flow around cylinder benchmark [61] in Section 10.1. As a second test case, we consider the nonstationary flow for a higher Reynolds number through a venturi pipe in Section 10.2. In Section 10.3, all presented solvers are analyzed with respect to their numerical costs for two prototypical flow configurations.

Chapter 11 gives a conclusion and further outlooks. The work closes with some appendix chapters. In Appendix A, we prove that the cGP(1)-method and the well-known Crank-Nicolson are identical. By means of numerical tests, we confirm the theoretical results. In Appendix B.1, we present and shortly analyze the dG-C0($k$)-method proposed in [40] for the heat equation. We also perform preliminary numerical tests to demonstrate the accuracy of the dG-C0(2)-method.

# 2

# Basics of the time stepping schemes

## 2.1. Introduction

In many practical situations, the processes under consideration are nonstationary and consequently the phenomenons are modeled as time dependent partial differential equations (PDEs). The solution of such problems requires the simulation of time dependent PDEs. In this chapter, we discuss the basic aspects regarding the numerical solution of such PDEs. These PDEs are commonly solved by treating the space and time variable separately. Discretizing only one of the variables is referred to as the semi-discretization. Spatial discretization is usually carried out by using the finite difference, finite element and finite volume method. After discretizing the space variable, a system of ordinary differential equations (w.r.t. time) is obtained which can be solved by using a state of the art numerical integration scheme. Our focus will be on the temporal discretization throughout this thesis.

Semi discretization in time corresponds to the solution at discrete points in time. Depending on the ordering of the semi-discretization, we distinguish the following two approaches:

- Method of lines
- Rothe's method

### 2.1.1. Method of lines

Finding the solution of the time dependent PDEs where the spatial variable is discretized first and then the time variable is known as the method of lines. After applying the semi-discretization in space, a system of ordinary differential equations (ODEs) is obtained. For first order PDEs, solving this system of ODEs gives the discrete solution along the trajectories or lines in time.

### 2.1.2. Rothe's method

The solution approach which first conducts the semi-discretization in time is referred to as Rothe's method, also known as the method of discretization in time. The basic idea in Rothe's method is to consider a PDE as ODE in the function space. Semi-discretization in time is applied at first which leads to a set of time independent or stationary PDEs. The solution of these PDEs yields the solution at the discrete points in time.

In order to discuss the various time stepping schemes, we consider the following time dependent problem

$$\frac{\partial u}{\partial t} = \mathcal{L}(t,u), \qquad u(0) = u_0, \quad \text{in} \quad [0,T]\times\Omega, \tag{2.1}$$

with $\Omega \subset \mathbb{R}^d$ a domain, $\mathcal{L}$ is the linear differential operator and $u_0$ is an initial condition. All data is assumed to be smooth enough. To start with the time discretization, a time interval $[0,T]$ is divided into $N$ subintervals $[t_{n-1},t_n]$, $n = 1,\ldots,N$ such that $0 = t_0$, $t_{n-1} < t_n$ and $t_N = T$. The length of each subinterval is denoted by

$$\tau_n := t_n - t_{n-1}, \qquad n = 1,\ldots,N.$$

The solution at time discrete point $t_n$ can only depend on the previous points $t_{n-1},t_{n-2},\ldots$ Thus the time discretization always corresponds to the extrapolation. The solution at the initial time $t_0$ is always known which is the initial condition. If only one previous time level is involved to find the solution at time $t_n$, the associated method is known as one-step method. On the other hand, if the solution at point $t_n$ depends on more than one previous level, the method is called multi-step method. The most commonly used time stepping schemes are the one-step methods which are discussed here. The time marching methods are generally classified [12] according to the choice of points at which right hand side is computed. Typically the following two types are formulated:

**Definition 2.1 (Explicit methods)** *If the numerical method computes the solution at the new time level from the known solutions at previous time levels, then the method is said to be* explicit method *[12]. Explicit methods are easy to implement and require low computational cost per time step. Their major disadvantage is the limitation on the time step size, specially for the differential equations with fast decaying solution (stiff problems) due to Courant-Friedrichs-Levy condition (CFL) condition [25].*

**Definition 2.2 (Implicit methods)** *If the numerical method includes the information from the previous time levels as well as the information from the current time level to find the solution at the new time level, then the method is said to be* implicit method *[12]. Usually, much bigger time steps are allowed, and some methods are even unconditionally stable, although, this is achieved at the increased computational cost per time step.*

Next, we discuss the most commonly used time stepping schemes to find the numerical solution of PDEs.

#### 2.1.3. Explicit Euler (EE) method

The simplest example of a one-step method to integrate an ODE is the Explicit Euler method which is based on the approximation of the time derivative by using forward differences,

$$\frac{\partial u}{\partial t}(t_n) \approx \frac{u_{n+1} - u_n}{\tau_n} = \mathcal{L}(t_n, u_n).$$

This method is first order accurate in time. In the past, explicit methods have commonly been used for the simulation of nonstationary incompressible flows but due to the limitation on the time step size for the stability reasons, implicit methods nowadays become much more popular than the explicit methods. One common example is Explicit Euler method which is only stable for small time step sizes due to the CFL-condition [25].

#### 2.1.4. Implicit Euler (IE) method

In this method, the time derivative at time $t_{n+1}$ is approximated by using the backward difference formula

$$\frac{\partial u}{\partial t}(t_{n+1}) \approx \frac{u_{n+1} - u_n}{\tau_n} = \mathcal{L}(t_{n+1}, u_{n+1}).$$

Here, the method differs from the Explicit Euler method only in the computation of the right hand side which is now computed on the new time level and consequently, one has to solve the system of equations to obtain the solution at time $t_n$. Similar to the Explicit Euler method, the Implicit Euler is also 1st order accurate in time but every step is more expensive. The method fulfills the so called strong A-stability condition (which will be discussed later) and therefore, there is no limitation on the time step size.

### 2.1.5. Crank-Nicolson (CN) method

An important implicit one-step scheme which corresponds to the central difference scheme in time is the Crank-Nicolson method where the time derivative at time $t_{n+1/2} = \frac{t_{n+1}+t_n}{2}$ is approximated by

$$\frac{\partial u}{\partial t}(t_{n+1/2}) \approx \frac{u_{n+1}-u_n}{\tau_n} = \frac{1}{2}[\mathcal{L}(t_{n+1},u_{n+1}) + \mathcal{L}(t_n,u_n)].$$

The Crank-Nicolson method is based on the average of the explicit and implicit Euler method. This method is 2nd order accurate in time and satisfies the so called A-stability condition, so there is no limitation in time step size. The computational cost of the CN is slightly higher than the Implicit Euler method but on the other hand, it is more accurate in time as compared to the Implicit Euler method. Although the Crank-Nicolson method is an implicit method, sometimes it may suffer from (non physical) oscillations because of its weaker damping properties (no strong A-stability).

Note that the Explicit/Implicit Euler and Crank-Nicolson methods can be summarized as a single equation by introducing a control parameter $\theta$ $(0 \leq \theta \leq 1)$ as follows:

$$\frac{u_{n+1}-u_n}{\tau_n} = [\theta\mathcal{L}(t_{n+1},u_{n+1}) + (1-\theta)\mathcal{L}(t_n,u_n], \tag{2.2}$$

where $\tau_n = t_{n+1} - t_n$ is the length of the time step and $\theta$ is the implicitness parameter which controls the scheme as follows:

| | | | |
|---|---|---|---|
| $\theta = 0,$ | explicit scheme, | order $\mathcal{O}(\tau)$, | Forward Euler scheme, |
| $\theta = \frac{1}{2},$ | implicit scheme, | order $\mathcal{O}(\tau^2)$, | Crank-Nicolson scheme, |
| $\theta = 1,$ | implicit scheme, | order $\mathcal{O}(\tau)$, | Backward Euler scheme. |

### 2.1.6. Fractional-step-$\theta$ (FS-$\theta$) method

Another method which combines the advantage of implicit Euler (strong A-stability) and Crank-Nicolson (2nd order accurate) is the *fractional-step*-$\theta$ scheme, originally proposed by Glowinski [22], features the same computational cost as the Crank-Nicolson scheme. It uses three different values of the time step $\tau$ and for the parameter $\theta$. For a realistic comparison, we choose a macro time step $K = 3\tau$ consisting of three time steps of size $\tau$. Then for the implicit Euler or Crank-Nicolson method we perform three time steps with the same values for $\theta$ and $\tau$, while in case of the FS-$\theta$-method different values are used in each micro time step.

For the FS-$\theta$-method, we proceed as follows: Given the parameters $\theta \in (0,1), \theta' = 1-2\theta$, and $\alpha \in [0,1]$, subdivide the time interval $(t_n, t_{n+1})$ into three subsets and update the solution as follows:

1. $u_{n+\theta} = u_n + [\alpha\mathcal{L}(t_{n+\theta},u_{n+\theta}) + (1-\alpha)\mathcal{L}(t_n,u_n)]\theta\tau$
2. $u_{n+1-\theta} = u_{n+\theta} + [(1-\alpha)\mathcal{L}(t_{n+1-\theta},u_{n+1-\theta}) + \alpha\mathcal{L}(t_{n+\theta},u_{n+\theta})]\theta'\tau$

3. $u_{n+1} = u_{n+1-\theta} + [\alpha\mathcal{L}(t_{n+1}, u_{n+1}) + (1-\alpha)\mathcal{L}(t_{n+1-\theta}, u_{n+1-\theta})]\theta\tau$

This method is second-order accurate in the special case $\theta = 1 - \frac{\sqrt{2}}{2}$, and the coefficient matrices are the same for all substeps if $\alpha = \frac{1-2\theta}{1-\theta}$. This method is strongly A-stable which is advantageous in the case of rough initial or boundary data due to a strong damping of (non physical) oscillations. Moreover, this method is little dissipative which is important in the computation of non-physical temporal oscillations which may occur in the simulation of incompressible flows. A rigorous theoretical analysis of the FS-θ-method can be found in [9, 36, 46].

## 2.2. Stability concepts for time discretization schemes

In general, the stability of numerical methods is studied by applying these methods to the following scalar test problem

$$\frac{du}{dt} = \lambda u, \qquad \lambda \in \mathbb{C},\ t \in [0,\infty) \tag{2.3}$$

subject to $u(0) = 1$. This simple problem has the exact solution

$$u(t) = e^{\lambda t}.$$

This implies that the exact solution approaches zero for $t \to \infty$ if $Re(\lambda) < 0$. If the numerical method also posses the same property, then the numerical method is said to be A-stable. Before giving the precise definition of the A-stability, we first define the *stability region*. The numerical method applied to the test problem (2.3) can be written as

$$u^{n+1} = \phi(h\lambda)u^n,$$

for a so called stability function $\phi : \mathbb{C} \to \mathbb{C}$. The absolute stability region (or simply the stability region) is defined to be the set

$$\{z \in \mathbb{C} || \phi(z)| < 1\}.$$

**Definition 2.3 (A-stability)** *A numerical method is said to be A-stable [14, 16] if the region of absolute stability includes the set*

$$\{z \in \mathbb{C} | Re(z) < 0\},$$

*that is, the left half plane.*

Next, we discuss the stability of various time stepping schemes presented in this chapter.

### 2.2.1. Stability of explicit/implicit Euler and Crank-Nicolson method

The explicit Euler method for the test problem (2.3) can be written as

$$u^{n+1} = \phi(z)u^n$$

for $\phi(z) = 1 + z$. The stability region for this method is

$$\{z \in \mathbb{C} || 1 + z| < 1\},$$

which is the unit circle centered at -1 on the real axis (see [10]). The stability region for the explicit Euler and Crank-Nicolson are shown in Figure 2.1. The stability region for the implicit Euler is the whole complex plane except the unit circle centered at 1 on the real axis (see [10]).

Next, we define the concept of L-stability (or strongly A-stable). A-stability only insures the boundedness of the solution but it does not give the perfect stability. Sometime, it may happen that the rapidly decaying components are damped very slowly and consequently, non-physical oscillations are produced which are damped only for very small time steps. This leads to the concept of L-stability.

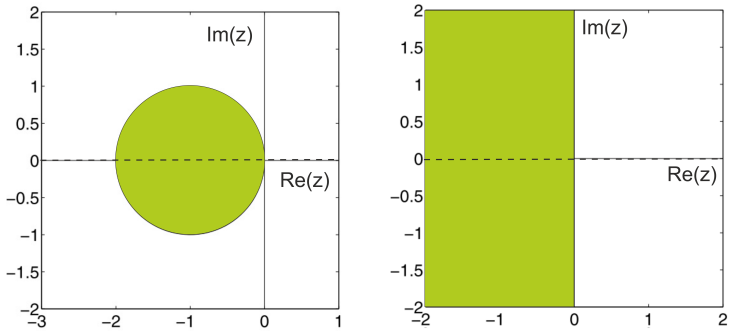



**Figure 2.1**: Stability region for the Explicit Euler method (left) and Crank-Nicolson method (right).

**Definition 2.4 (L-stability)** *A method is said to be L-stable [18] (or strongly A-stable) if it is A-stable and* $|\phi(z)| \to 0$ *as* $|z| \to \infty$, $z \in \mathbb{C}$.

**Remark 2.1** *Among the one-step methods discussed before, the following is true [26]:*

- ⊕ *The Explicit Euler is not A-stable, nor L-stable.*
- ⊕ *The Crank-Nicolson is only A-stable.*
- ⊕ *The Implicit Euler is A-stable as well as L-stable.*

# 3

# Fundamentals of the finite element method

## 3.1. Introduction

In this chapter, we explain the fundamentals of the finite element method (FEM) which is used for the spatial discretization throughout this numerical study. The finite element method was basically developed to solve the equation of elasticity and structural mechanics but nowadays it has become a powerful tool for the numerical treatment of partial differential equations (PDEs) in engineering, solid mechanics and fluid mechanics due to its flexibility, robustness and accuracy. Moreover, this method is also perfectly suitable as an adaptive method because it allows to make local refinement of the solution.

The idea of FEM is to reformulate the original PDE into a more suitable form known as the variational or weak form. This variational form can be achieved by multiplying the strong form of the equation by an arbitrary function called the test function and integrate over the domain. Once the variational form is obtained, there are two approaches, the Ritz method and the Galerkin method, which is based on the weighted residual method. Both these approaches finally lead to the same numerical results.

## 3.2. Weighted residual formulation

As an example, we consider the linear model problem

$$\begin{aligned} \mathcal{L}u &= f \quad \text{in} \quad \Omega \\ u &= 0 \quad \text{on} \quad \partial\Omega, \end{aligned} \tag{3.1}$$

where $\mathcal{L}$ is the 2nd order elliptic partial differential operator with homogeneous Dirichlet conditions at the boundary $\partial\Omega$ of a polygonal domain $\Omega \subset \mathbb{R}^d$, $d = 1,2,3$. A function $u : \Omega \to \mathbb{R}$ is a solution of the problem (3.1) if it is smooth enough and fulfills (3.1). Moreover, if the function $u$ is the exact solution, then the residual $\mathcal{R}(u) = \mathcal{L}[u] - f = 0$, but for an approximate solution $\bar{u} \approx u$, the residual $\mathcal{R}(\bar{u}) = \mathcal{L}[\bar{u}] - f \neq 0$ since the approximate solution does not satisfy the given differential equation rigorously.

The goal is to find the function $u_h$ which makes the residual zero weighted by appropriate functions (see [7, 13, 34, 42, 51]). To apply the finite element method, we first need to transform the equation (3.1) to the integral form, known as the *weak* (*or variational*) form. To this end, we multiply the strong form (3.1) by a suitable *test* (*or weight*) function $v$ and integrate over the domain $\Omega$, we have

$$\int_\Omega (\mathcal{L}[u] - f) v \, d\Omega = 0. \tag{3.2}$$

So the minimum requirement for the test function $v$ is thus the integral must exists. With this

notion, we have

$$\int_{\Omega} (\mathcal{L}[u] - f) v \, d\Omega = 0, \qquad \forall\, v \in V, \tag{3.3}$$

where $V$ is the test space. At this stage, it is common practice to reduce the regularity of the solution by applying the integration by parts to get rid of the highest order derivatives. By making use of Green's formula for partial integration and substitution of boundary condition leads to the following form

$$a(u,v) = \int_{\Omega} f v \, d\Omega, \qquad \forall\, v \in V, \tag{3.4}$$

where

$$a(u,v) = \int_{\Omega} \sum_{i,j=0}^{2} \frac{\partial u}{\partial x_i} \frac{\partial v}{\partial x_j} \, dx$$

is the bilinear form. We suppose that the function $u$ is approximated by the trial function $u_h \in U_h$ of the form

$$u_h(x) \approx \sum_{j=1}^{N} c_j \phi_j \tag{3.5}$$

from the finite dimensional trial space $U_h$, where $\phi_j$ are the basis basis functions (or interpolation functions) and $c_i$ are the unknown coefficients to be determined. In order to compute the approximate solution of equation (3.1), we need to replace the infinite dimensional space $V$ by a finite dimensional space $V_h \subset V$. The computational domain is subdivided into a finite number of non-overlapping pieces (subdomains) $\Omega_k$ called elements. That is,

$$\cup_{k=1}^{N} \Omega_k = \Omega.$$

In 1D these elements are called intervals, in 2D usually triangles or quadrilaterals and in 3D, tetrahedral and hexahedral elements are very popular choices [11, 13]. Since the test space $V_h$ is finite dimensional, the test function $v_h \in V_h$ can be written as

$$v_h(x) = \sum_{i=1}^{N} d_i \psi_i, \tag{3.6}$$

where $\psi_i$ are the basis functions from the test space and $d_i$ are the unknown coefficients. Substituting the equation (3.5) and (3.6) into equation (3.4), we have

$$a(u_h, v_h) = \int_{\Omega} f v_h \, dx, \qquad \forall\, v_h \in V_h. \tag{3.7}$$

The basis functions $\phi_i$ and $\psi_i$ are predefined and the goal is to compute the unknown $c_i$. Afterwards, the equation (3.4) implies that the expression (3.7) holds for all possible choices of the coefficients $d_i$, $i = 1,2,\ldots,N$. Thus, the unknown coefficients $c_i$ for the approximate solution $u_h$ can be determined by solving the following system of equations

$$\sum_{j=1}^{N} c_j \, a(\phi_i, \psi_j) = \int_{\Omega} f \psi_j \, dx, \qquad \forall\, i = 1,2,\ldots,N. \tag{3.8}$$

There are various methods of weighted residuals depending on the choice of $\psi_i$, for instance, the collocation method, the sub-domain method and the least squares method. One of the obvious choice to select the trial space and test space to be the same, i.e., $\phi_i = \psi_i$ which leads to the well known standard Galerkin's (or Bubnov-Galerkin) method that is used in this thesis. Use of functions $\phi_i \neq \psi_i$ yields the so-called Petrov-Galerkin approximation. We refer to the literature [7, 11, 13, 17, 34, 35, 42–44, 51] for the comprehensive introduction of FEM.

### 3.3. Construction of FEM basis functions

Now we have done a great deal of work but we are not much close to find the solution of the given differential equation as we know nothing about the FEM basis functions. The finite element method offers a general and systematic techniques for the construction of the basis functions in case of the so-called Galerkin method. Here the basis functions can be chosen as the piecewise polynomial of any order over the subdomains $\Omega_k$.

Let us now construct a finite dimensional subspace $V_h$ of $V$. To this end, we assume for simplicity that the boundary $\partial\Omega$ is a polygonal curve, thus the computational domain $\Omega$ is a polygonal domain. Afterwards, the computational domain $\Omega$ is decomposed into a finite number of non-overlapping triangles or quadrilaterals, called elements, i.e.,

$$\Omega = \cup_{k=1}^{N} \Omega_k, \qquad \Omega_k \cap \Omega_l = \emptyset \quad k \neq l,$$

also known as the triangulation of $\Omega$. Such partitions of domain are usually designed by hand or by using the automatic mesh generation tool. For the time dependent nodal coefficients $u_j(t) = u_h(\mathbf{x}_j, t)$ at any time $t$, the approximated solution in each element $\Omega_k$ can be interpolated by using the local *basis functions* (*or shape functions*) $\phi_j^{(k)}$ such that [45, 52]

$$u_h(\mathbf{x}, t) = \sum_{j=1}^{m} u_j(t)\phi_j^{(k)}(\mathbf{x}) \qquad \forall\, \mathbf{x} \in \Omega_k,$$

where $m$ is the number of local degrees of freedom for a single element $\Omega_k$. Summing up these local basis functions over all elements, we obtain an approximation for $u_h$ over the whole domain

$$u_h(\mathbf{x}, t) = \sum_{j=1}^{M} u_j(t)\phi_j(\mathbf{x}) \qquad \forall\, \mathbf{x} \in \Omega,$$

where $M$ denotes the total number of degrees of freedom. Usually, the resulting system matrix is dense and ill-conditioned except for if strongly orthogonal polynomials are used. To circumvents these difficulties, the basis functions for Lagrange finite elements are defined locally which has the following properties [45, 52]:

- Interpolation property: The basis function $\phi_j$ is one at the node $j$ and vanishes at all other nodes

$$\phi_j(\mathbf{x}_i) = \delta_{ij} = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{if } j \neq i \end{cases} \tag{3.9}$$

$$\implies u_h(\mathbf{x}_j, t) = \sum_{j=1}^{M} u_j(t)\phi_j(\mathbf{x}) = u_j(t),$$

  where $\delta_{ij}$ is the Kronecker delta.

- Constant sum property: The sum of the local basis functions $\phi_j^{(k)}$ in each element $k$ should be identically to one, i.e.,

$$\sum_{j=1}^{M} \phi_j(\mathbf{x}) = 1 \qquad \forall\, \mathbf{x} \in \Omega_k. \tag{3.10}$$

- Conservation property: The derivative sum of all the basis functions at any given location in each element vanishes which is the consequence of the constant sum property

$$\sum_{j=1}^{M} \nabla\phi_j(\mathbf{x}) = 0 \qquad \forall\, \mathbf{x} \in \Omega_k. \tag{3.11}$$

Note that vanishing the basis function $\phi_j^{(k)}$ outside the element $k$ leads to the sparsity of the resulting system matrix which is desirable for the computational efficiency. An easy and systematic way for generating the shape functions of any order can be achieved by using the Lagrange polynomials.

### 3.4. Quadrilateral elements

The choice of an element always depends on the problem being solved and the accuracy required. In general, the quadrilateral elements are usually favorable above triangular elements except where the region cannot be approximated easily by quadrilaterals [11], for instance, when the irregular or complex geometries having curved boundaries have to be approximated. In comparison to triangular elements, only one half of the number of quadrilateral elements are required and consequently, the overall computational time for the construction of matrices and vectors reduces. Here, we explain the biquadratic quadrilateral $(Q_2)$ elements which has been used in this thesis.

**Biquadratic elements ($Q_2$):**

The domain $\Omega \subset \mathbb{R}^2$ is discretized into a number of quadrilateral cells $\Omega_k$, $k \in \mathbb{N}$. Moreover, we also assume that each $\Omega_k$ is convex and no angle of $\Omega_k$ is too close to $0^\circ$ [11]. On each quadrilateral, the biquadratic element $Q_2$ is defined by introducing the four additional mid-side node points, together with a ninth node at the centre as shown in Figure 3.1 (see [19]). To construct the local shape functions on an arbitrary physical element $\Omega_k$, we make use of a reference coordinate system. A local coordinate system $(\xi,\eta)$ introduced. Let $\hat{\Omega}_k = [-1,1]^2$ be the reference element located at the center of this coordinate system $(\xi,\eta)$. A one-to-one bilinear mapping between the physical and the reference element is referred to $F_k : \hat{\Omega}_k \to \Omega_k$. Once the basis functions have been defined on the reference element in terms of reference coordinates, the inverse mapping $F_k^{-1} : \Omega_k \to \hat{\Omega}_k$ can be employed to get back to the physical space. Figure 3.1 illustrates the biquadratic element in both the physical and reference coordinate system. In this case, there are nine shape functions, four associated with the vertices, four with the edge mid-points and one internal (or bubble) function. The nodal shape functions in the reference coordinate system are defined locally such that the basis function $\hat{\phi}_j$ is one at the node $j$ and vanishes at all other nodes

$$\hat{\phi}_j(\hat{\mathbf{x}}_i) = \delta_{ij} = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{if } j \neq i \end{cases} \tag{3.12}$$

where $\hat{\mathbf{x}}_i \in \hat{\Omega}_k$ represent the $i$th node in $\hat{\Omega}_k$, i.e., vertices, edge mid-points and the centroid. In general, these nine shape functions are biquadratic polynomials, a linear combination of the following monomials

$$\left\{1, \xi, \eta, \xi\eta, \xi^2, \eta^2, \xi^2\eta, \xi\eta^2, \xi^2\eta^2\right\},$$

where $-1 \leq \xi, \eta \leq 1$. Then, the space $Q_2(\Omega_k)$ on the physical element is defined as follows

$$Q_2(\Omega_k) = \left\{q \circ F_k^{-1} : q \in \text{span}\{1, x, y, xy, x^2, y^2, x^2y, xy^2, x^2y^2\}\right\} \tag{3.13}$$

We refer the interested reader [11, 43] for more details.

**Discontinuous $P_1$ element ($P_1^{disc}$):**

On each quadrilateral $\Omega_k$ of $\Omega$, the discontinuous $P_1$ finite element space consists of piecewise linear polynomials which are discontinuous across inter-element boundaries (zero outside of the element) [19]. Three shape function are defined locally at the center of each element as shown in Figure 3.2. These shape functions correspond to the function value and both of its partial

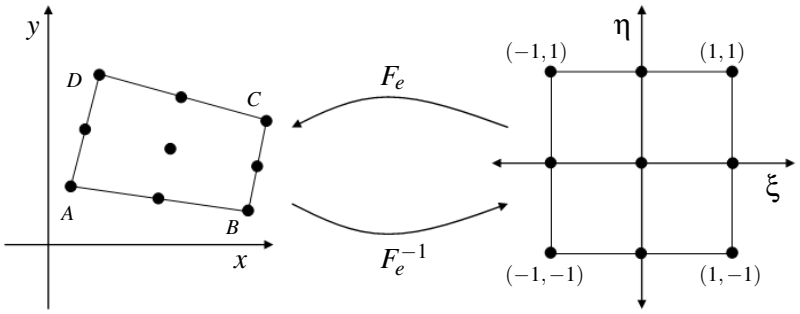



**Figure 3.1**: Mapping between the biquadratic physical and reference element.

derivatives. There are two possible choices for the $P_1^{disc}$ finite element: *the unmapped approach* (where the finite element space is spanned by the constant 1 and the global coordinates $x$ and $y$) or *the mapped approach* (where the finite element space is defined locally by means of the constant 1 and the local coordinates $\xi$ and $\eta$). In case of the mapped approach, the basis functions are defined by using the bilinear mapping between the physical $\Omega_k$ and the reference element $\hat{\Omega}_k = [-1,1]^2$. The nodal shape functions on the reference element are defined as follows

$$\hat{\phi}_1(\xi,\eta) = 1$$
$$\hat{\phi}_2(\xi,\eta) = \xi$$
$$\hat{\phi}_3(\xi,\eta) = \eta.$$

The space $P_1(\Omega_k)$ on the physical element is defined as

$$P_1(\Omega_k) := \left\{ q \circ F_k^{-1} : q \in \text{span}\{1,x,y\} \right\}. \tag{3.14}$$

On the other hand, in case of the unmapped approach, a local coordinate system can be considered (see [1, 58, 60]) which is obtained by joining the midpoints of the opposite sides of $\Omega_k$. This leads to the concept of nonparametric finite element. Then, we set on each element

$$P_1(\Omega_k) := \text{span}\{1,\xi,\eta\}.$$

We remark that this choice is known to provide optimal error estimates on general meshes (see [5]).

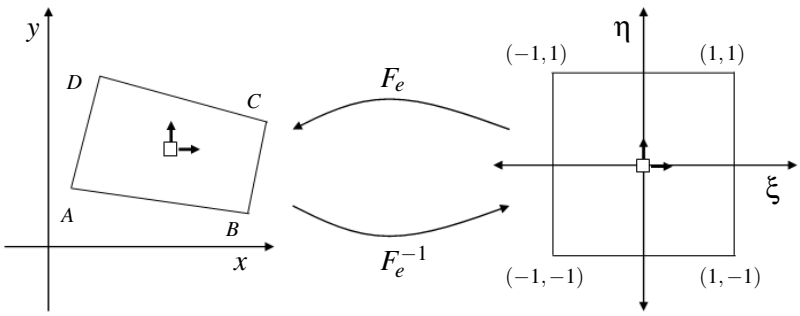



**Figure 3.2**: Mapping between the discontinuous $P_1$ physical and reference element.

### 3.5. The conforming Stokes element $Q_2/P_1^{disc}$

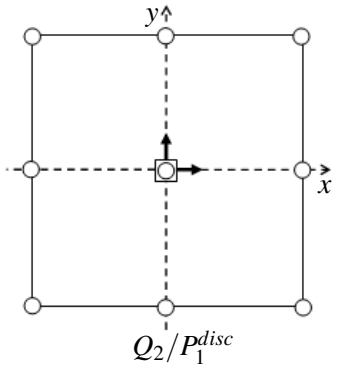
After discretizing the continuous (Navier-) Stokes problem by a standard Galerkin method in Chapter 8 and 9, we approximate the domain $\Omega \subset \mathbb{R}^2$ by a collection of quadrilaterals $\Omega_k$, $k \in \mathbb{N}$. On each quadrilateral $\Omega_k$, we define the finite element space $V_h$ for the velocity and the finite element space $Q_h$ for the pressure as follows

$$V_h := \left\{ v_h \in [H_0^1(\Omega)]^2, \quad v_{h|\Omega_k} \in [Q_2(\Omega_k)]^2 \quad \forall\, \Omega_k \subset \Omega, \quad v_h = 0 \quad \text{on} \quad \partial\Omega \right\}, \tag{3.15}$$

$$Q_h := \left\{ q_h \in L_2(\Omega), \quad q_{h|\Omega_k} \in P_1(\Omega_k) \quad \forall\, \Omega_k \subset \Omega \right\}, \tag{3.16}$$

where $Q_2(\Omega_k)$ and $P_1(\Omega_k)$ are the biquadratic and linear spaces, resp., on the quadrilateral element $\Omega_k$. Let $\hat{\Omega}_k = [-1,1]^2$ be the reference square and we define a bilinear mapping $F_k : \hat{\Omega}_k \to \Omega_k$ from the reference square $\hat{\Omega}_k$ to an arbitrary quadrilateral $\Omega_k$. The space $Q_2(\Omega_k)$ is defined by (3.13) with 9 local degrees of freedom located at the vertices, mid-points of the edges and in the center of the quadrilateral as show in Figure 4.2. Similarly, the space $P_1(\Omega_k)$ on the physical element is defined by (3.14) with 3 local degrees of freedom, the function value and both of its partial derivatives located in the center of the quadrilateral (see Figure 4.2).

In two dimensions this choice yields 18 velocity degrees of freedom and three pressure degrees of freedom in case of the cGP(1) or the Crank-Nicolson method. The chosen finite element pair $Q_2/P_1^{disc}$ is one of the most popular Stokes element which satisfies the discrete Ladyzenskaya-Babuška-Brezzi (LBB) stability condition (see [5, 8, 21, 41, 53]), i.e., inf-sup condition.




$$\begin{cases} \circ & u_h, v_h & \rightarrow \text{velocity at } \hat{\mathbf{x}}_i \\ \square & p_h, \frac{\partial p_h}{\partial x}, \frac{\partial p_h}{\partial y} & \rightarrow \text{pressure at } \hat{\mathbf{x}}_i \end{cases}$$

**Figure 3.3**: Location of the degrees of freedom for the $Q_2/P_1^{disc}$ on the reference element.

#### Approximation order

The bilinear transformation $F_k$ applied to a linear function on the reference element does not contain the full bilinear basis and consequently, the method can be only first order on general meshes (see [1, 5])

$$\|p - p_h\| = \mathcal{O}(h),$$

where $h = \max(h_k)$ is the diameter of the mesh cell $\Omega_k$. To remedy this situation, a local coordinate system can be considered (see [1, 58, 60]) which is obtained by joining the midpoints of the opposite sides of $\Omega_k$. Then, we set on each element

$$P_1(\Omega_k) := \text{span}\{1, \xi, \eta\}.$$

The inf-sup condition is also satisfied in this case and the second order approximation is recovered for the pressure (see [5, 21])

$$\|p - p_h\| = \mathcal{O}(h^2).$$

For smooth solutions, the approximation errors for the velocity and pressure are $\mathcal{O}(h^3)$ and $\mathcal{O}(h^2)$, respectively,(see [5]).

## 3.6. Edge oriented jump stabilization

Several methods are known to stabilize the standard Galerkin discretization and to reduce the oscillatory behavior of the solution. For instance, Streamline-Upwind Petrov-Galerkin (SUPG) and Streamline Diffusion (SD) method. Here we utilize a method originally proposed by Douglas and Dupont, known as edge-oriented stabilization. The main idea in this technique is to augment the standard Galerkin discretization by an interior penalty term involving a jump in the gradient over the element boundaries. To be specific, we use the following term which is also used by A. Ouazzi and S. Turek [59],

$$\langle Ju_h, v_h \rangle = \sum_{\text{edge E}} \max(\gamma^* \nu h_E, \gamma h_E^2) \int_E [\nabla u_h][\nabla v_h] d\sigma, \tag{3.17}$$

where $\nu, h_E$ denote the viscosity and length of element edge, resp. The parameters $\gamma$ and $\gamma^*$ have no significant influence on the accuracy of the results and the solution is stable and accurate for large range of parameters. In our case, these parameters are set to 0.1 and 0.0, respectively. To integrate the proposed jump term (3.17) into the resulting system matrix, a loop over all the edges is performed. In order to deal this term with standard Galerkin method appropriately, an extension of the matrix stencil of the system matrix is required to put the extra nonzero entries. This extension of matrix stencils leads to some additional memory requirement. For more details, see [59].

# 4

# Solution of nonlinear and linear systems

This chapter deals with the solution methods for the discrete systems obtained from the finite element discretization of the partial differential equations. From the discretization of incompressible Navier-Stokes equation in Chapter 9, an algebraic system of nonlinear equations is obtained. This system has to be solved in every time step. An efficient way to solve the nonlinear system is the Newton method which is well known for its quadratic convergence if the initial solution is chosen close enough to the exact solution and the problem is smooth enough. Therefore, once the Newton method converges, it requires only a very few iterations. However, the construction of the Newton matrix in every nonlinear iteration is sometimes very expensive due to the approximation of the Fréchet-derivative of the nonlinear operator with respect to the last iteration. Another choice for solving the system of nonlinear algebraic equations is to use the fixed point iteration method where the approximation of the Fréchet-derivative may be as chosen the associated nonlinear operator itself or even the linear part of the nonlinear operator. The resulting method is less efficient than the Newton method (i.e., with linear or superlinear convergence only). In our numerical study, we apply the fixed-point and Newton method.

The resulting linear systems, for instance, from the discretization of heat equation, Stokes equations or the linearized Navier-Stokes equations are solved by using Krylov space or multigrid methods. The coefficient matrices of these linear systems arising from finite element discretization are very large and sparse. A sparse matrix is a matrix which has only a small proportion of nonzero entries. Mainly, the methods for solving the linear systems can be divided into two classes, direct methods (i.e., those which execute in a predetermined number of operations) and iterative methods (i.e., those which attempt to converge to the desired solution in an unknown number of iterations).

The most commonly used direct methods are the Gaussian Elimination and LU decomposition. The complexity of these methods is $\mathcal{O}(n^3)$ [54, 56], where $n$ is the number of degrees of freedom associated with matrix $A$. Another direct method is the Cholesky factorization for symmetric positive definite matrices (SPD) which also has the same complexity as Gaussian Elimination. Direct methods are often used for small dense problems and become prohibitively expensive for large problems due to their asymptotically cubic runtime. Although the associated coefficient matrices are sparse but unfortunately their structure is such that after a few steps of Gaussian elimination (approximately $\sqrt{n}$), the computational effort grows significantly because most of the zero elements are replaced by nonzero ones (see [6] for more details). Moreover, one major disadvantage of these methods is a need to form an explicit matrix which in practice requires a lot of storage.

On the other hand, the solution of large sparse problems which are typically encountered in the discretization of PDEs, can be solved in a more efficient way by using the iterative methods. For the formal description of iterative methods, we refer the reader to [4].

In the following, we explain in more detail the linear solver.

## 4.1. Basic iterative solvers for sparse linear systems

The iterative methods for the solution of linear systems refer to a wide class of methods which attempt to find the solution of a problem by using successive approximations to obtain more accurate solution starting from an initial guess. There are two main classes of iterative methods: stationary and nonstationary iterative methods [4].

### 4.1.1. Stationary iterative solvers

The basic iterative methods used for solving large linear systems $Ax = b$, where $A$ is a given matrix and $b$ is a given vector, are the stationary iterative solvers. Stationary iterative methods are those which can be expressed in the simple form.

$$x^{(l)} = Bx^{(l-1)} + c$$

(where neither the matrix $B$ nor the vector $c$ depend upon the iteration count $l$) [4]. These iterative methods are based on relaxation of the coordinates. Starting with an initial guess, these methods modify the components of the approximation, one or a few at a time and in a certain order, until the convergence is reached. Nowadays these methods are rarely used as stand-alone iterative solvers due to the poor convergence. Nevertheless, they have not lost their importance. Indeed they are often used in conjunction with modern efficient iterative methods, for example in the Krylov space method for preconditioning, and in the multigrid method for smoothing. The four main stationary methods are the following:

- Jacobi method (JAC),
- Gauss-Seidel method (GS),
- Successive overrelaxation method (SOR),
- Symmetric successive overrelaxation method (SSOR).

All these methods can be formulated as a *defect-correction* as follows

$$x^{(l)} = x^{(l-1)} + \omega P^{-1}(b - Ax^{(l-1)}),$$

where $P$ is matrix for which we classify for $A = L + D + R$:

| | |
|---|---|
| JAC: | $P = D$ |
| GS: | $P = D + L$ |
| SOR: | $P = D + \omega L, \qquad 0 < \omega < 2$ |
| SSOR : | $P = (D + \omega L)D^{-1}(D + \omega U).$ |

Here, $L, D, R$ denote the lower triangular, diagonal and upper triangular part of the matrix $A$.

### 4.1.2. Nonstationary iterative solvers

Nonstationary iterative methods are those in which the computations involve information that changes at each iteration $k$, for instance, the Krylov space methods, i.e., methods that seek to generate better approximations from the Krylov subspace. These methods find an approximation $x_k$ to the linear system $Ax = b$ over the $m$-th Krylov subspace $K_m(A, r_0) + x_0$, where $x_0$ is the initial solution and $r_0$ represents the initial residual vector, given by $r_0 = b - Ax_0$. A Krylov subspace of

dimension $m \in \mathbb{N}$ generated by a matrix $A \in \mathbb{R}^{n\times n}$ and a vector $r \in \mathbb{R}^n$ is the subspace spanned by the vectors of the Krylov sequence:

$$K_m(A,r) = \text{span}\left\{r, Ar, A^2r, \cdots, A^{m-1}r\right\}.$$

The dimension of this subspace increases by one for every step of the approximation process. These methods are extensively used nowadays for the solution of linear systems arising from the discretization of partial differential equations (PDEs). Commonly known Krylov space methods are

- Conjugate Gradient (CG),
- Generalized Minimal Residual (GMRES),
- Biconjugate Gradient (BiCG),
- Quasi-Minimal Residual (QMR),
- Conjugate Gradient Squared (CGS),
- Biconjugate Gradient Stabilized (BiCGStab),
- Chebyshev Iteration.

A comprehensive study regarding the Krylov space methods can be found in [4]. Among the Krylov space method, we apply the Bi-Conjugate Gradient Stabilized method (BiCGStab) and the Generalized Minimal Residual Method (GMRES) in our numerical study.

#### Bi-Conjugate Gradient Stabilised Method (BiCGStab)

The Bi-Conjugate Gradient Stabilised method was originally developed by van der Vorst [62] in 1992 from the CGS, BiCG and GMRES methods, for the solution of non-symmetric linear system of equations while avoiding the often highly irregular convergence behavior of CGS and BiCG, and the large storage requirements of GMRES. This algorithm requires six auxiliary vectors, and performs two preconditioning steps in each iteration. Furthermore, the two matrix-vector product are also perform in each BiCGStab step. Its convergence behavior is much smoother and in most of the cases, it converges considerably faster than CGS. Like other iterative methods, BiCGStab method is usually combined with a preconditioning to speed up its convergence. We apply the standard algorithm for the preconditioned BiCGStab method in our numerical study (see [62]).

#### Generalized Minimal Residual Method (GMRES)

The Generalized Minimal Residual Method (GMRES) was proposed by Saad and Schultz [48] in 1986 to find the solution of non-symmetric linear systems. This method is an extension of the minimal residual method (MINRES) which is only applicable for the solution of symmetric systems. In the GMRES method, a sequence of orthogonal vectors is generated and all the previously computed vectors in the orthogonal sequence are required to calculate the next iteration. Consequently, the required storage for these vectors often exceeds the available memory limit before reaching the stopping criterion which is the drawback of this method. To remedy this difficulty, a restarted version of this method, usually refereed by GMRES($m$) is commonly used where the algorithm is restarted in every $m$ steps for fixed integer $m$. The GMRES method has an advantage over BiCGStab method due to its convergence behavior since it is more stable in practice for ill-conditioned systems. Moreover, only one matrix-vector multiplication is performed in the GMRES while the BiCGStab requires two matrix-vector products. Our implementation for the preconditioned GMRES($m$) method corresponds to the algorithm in [48].

#### Krylov-space smoothers

Beside using the preconditioned Krylov space methods as stand-alone solver, these methods can also be applied as smoother in the multigrid method to accelerate the convergence and robustness (see [19]). Since the most important part of the multigrid method is the smoothing, the efficiency and robustness depend in many cases on the smoothing algorithms. In contrast to the classical smoothers such as Jacobi, Gauß-Seidel or SSOR whose damping properties for different error modes is fixed [19], the damping properties of GMRES depend on the initial residual and the GMRES itself is constructed to minimize the residual. Therefore, it damps those error modes which leads to largest residual norm reduction. For instance, the discretization of incompressible Navier-Stokes equations yields an indefinite system of equations and standard smoothers become unstable for indefinite problems [19]. To remedy this, one of the choice is to use Krylov space methods as an inner or outer iteration for the multigrid methods. In the present numerical study, we employ the preconditioned GMRES method as smoother in the multigrid solver.

### 4.2. Preconditioning

Preconditioning is a technique by which the condition number of the system matrix is improved to speed up the convergence rate of the iterative methods. It is an important ingredient behind the success of iterative methods [47]. In the subject of numerical analysis, a preconditioner $P$ for a matrix $A$ can be chosen in a way such that $P^{-1}A$ has a smaller condition number than $A$. Rather than solving the original system, we solve the system $Ax = b$ indirectly by solving

$$P^{-1}Ax = P^{-1}b.$$

One may apply the preconditioner $P$ either from the left or right. For a symmetric positive definite matrix $A$, it is also suggested that the preconditioner $P$ should be symmetric positive definite. If we choose

$$P = A,$$

in this case, the conditioner number becomes one and the solution of the system is reached only in one iteration. However, in practice it is not much useful as calculating the inverse of the matrix $A$ is not so easy. Thus in practice, parts of the matrix $A$ are taken. The most common used preconditioners [4, 47] are

| | |
|---|---|
| JAC: | $P = D$ |
| GS: | $P = D + L$ |
| SOR: | $P = D + \omega L, \qquad 0 < \omega < 2$ |
| SSOR : | $P = (D + \omega L)D^{-1}(D + \omega U).$ |

### 4.3. Multigrid solvers for linear equations

Another efficient alternative for the solution of large and sparse linear systems is to use the multigrid or multilevel methods (see for instance [3, 23, 38, 47, 64]). These methods were mainly developed for linear systems arising from the discretization of elliptic PDEs and later on also extended to handle other types of PDEs, including nonlinear ones. In the subject of numerical analysis, these methods form a group of algorithms for the solution of differential equations using a hierarchy of discretizations. Multigrid solvers are regarded as the most efficient solvers for solving the large sparse linear systems, in particular for those arising from the discretization of PDEs where the conditions number of the system matrix deteriorates with increasing the problem size.

In contrast to the other iterative solvers (like for instance BiCGStab or GMRES), these solvers converge independent of the mesh size and require only a linear amount of operations w.r.t. the number of unknowns. The efficiency and the robustness of these solvers crucially depend on the smoothing operator. Multigrid methods can be classified into two categories:

- Algebraic Multigrid Methods (AMG)
- Geometric Multigrid Methods (GMG)

The primary difference between the algebraic and geometric multigrid algorithms is that the AMG methods require only a single mesh information and the system matrices of the 'coarse grids' are constructed by using algebraic operations [47]. On the other hand, the GMG generates the coarser levels by using the hierarchy of mesh refinements. In recent years, the multigrid methods have become quite popular for the solution of discretized linear systems due to their convergence behavior. Throughout this thesis, we will consider only the geometric multigrid methods (see [23, 38, 39, 58] for more details).

The resulting linear systems on each time interval $[t_{n-1}, t_n]$ in Chapter 8 or 9, which are $6 \times 6$ block systems in the case of the cGP(2) and dG(1) approach and $3 \times 3$ block systems for the cGP(1)-method, are treated by using a geometrical multigrid solver with a smoother based on an element loop where, for each element, simultaneously all unknowns are updated that belong to this element. This type of smoother is also called "local pressure Schur complement smoother" in [58] or "Vanka-type smoother" in [65] (which can be additionally applied as preconditioner in a GMRES method to make this method more robust).

#### Idea of multigrid method

The fundamental concept in multigrid techniques is to exploit different mesh discretization of the underlying problem for capturing errors and to obtain the optimal convergence rates from the relaxation techniques on different levels. The basic iterative methods such as Jacobi, Gauß-Seidel, successive over-relaxation (SOR), symmetric successive over-relaxation (SSOR) are referred to as relaxation methods. These method have the the ability to reduce oscillatory modes or high-frequency errors rapidly but the smooth modes or low-frequency errors are damped very slowly. However, the low frequency modes when mapped to the coarser meshes becomes the high-frequency mode and therefore they are annihilated on these meshes efficiently. This process can be successively repeated on a hierarchy of meshes in order to eliminate all the components of the error. Thus the mutigrid is characterized by a defect correction method acting on a hierarchy of mesh levels. We formulate the standard multigrid algorithm by considering the following (discrete) linear system of equations

$$Au = f \tag{4.1}$$

with $A \in \mathbb{R}^{N \times N}$ a system matrix and $u, f \in \mathbb{R}^N$ the solution and the right hand side vectors, where $N = N(k)$ denotes the number of degrees of freedom, independently form the previous notation we denote in the sequel on the mesh level $k$, $k = 1, \ldots, L \in \mathbb{N}$. Let $I_{k-1}^{k} : \mathbb{R}^{N(k-1)} \mapsto \mathbb{R}^{N(k)}$ denotes the prolongation operator and $I_{k}^{k-1} : \mathbb{R}^{N(k)} \mapsto \mathbb{R}^{N(k-1)}$ the corresponding restriction.

### 4.3.1. Geometric Multigrid Methods (GMG)

We assume the existence of a hierarchy of levels $k$, $k = 1, \ldots, L$ created by using standard refinement scheme [58] of a coarse mesh and the associated finite element spaces $V_1 \subset V_2 \subset \ldots V_L = V_h$ based upon meshes $T_1, T_2, \ldots, T_L$. On each of these levels $k$, we have to assemble the discrete problem matrix $A_k$ and corresponding right hand side $f_k$. The RHS $f_N = f$ is specified on the finest

level $N$ only, while all other $f_k$ are generated during the multigrid run. Then, a standard $L$-level geometric multigrid algorithm MG$(L,\ldots)$ for the solution of (4.1) is described in Algorithm 4.1.

**Algorithm 4.1** Geometric multigrid algorithm.

**The $k$-level iteration MG$(k,A_k,u_k^0,f_k,\nu_1,\nu_2,\mu)$:**
The $k$-level iteration with initial $u_k^0$ yield an approximation to $u_k$, the solution of the problem

$$A_k u_k = f_k.$$

One step can be described in the following way:
For $k=1$, MG$(1,A_1,u_1^0,f_1,\nu_1,\nu_2,\mu)$ is the exact solution

$$MG(1,A_1,u_1^0,f_1,\nu_1,\nu_2,\mu) = A_1^{-1} f_1$$

For $k>1$, there are four steps:

1. Pre-smoothing steps: Compute the approximation $u_k^{\nu_1}$ by applying $\nu_1$ smoothing steps (iterations of a relaxation scheme) to $u_k^0$ with a basic iteration.

2. Correction step: Calculate the restricted residual (with the restriction operator $I_k^{k-1}$)

   $$f_{k-1} = I_k^{k-1}(f_k - A_k u_k^{\nu_1})$$

   and let $u_{k-1}^i (1 \le i \le \mu,\ \mu \ge 1)$ be defined recursively by

   $$u_{k-1}^i = MG(k-1,A_{k-1},u_{k-1}^{i-1},f_{k-1},\nu_1,\nu_2,\mu),\ 1 \le i \le \mu,\ u_{k-1}^0 = 0.$$

   Interpolate the error correction onto the fine grid and obtain $u_k^{\nu_1+1}$ (with prolongation operator $I_{k-1}^k$) via

   $$u_k^{\nu_1+1} = u_k^{\nu_1} + I_{k-1}^k u_{k-1}^{\mu}.$$

3. Post-smoothing steps: Analogously to step 1), apply $\nu_2$-smoothing steps to $u_k^{\nu_1+1}$ and obtain $u_k^{\nu_1+\nu_2+1}$.

Each iteration in MG$(L,\ldots)$ is called one cycle. Sufficiently many cycles on level $L$ are required to obtain a good approximate solution of problem (4.1). The whole process continues until the prescribed convergence criterion is reached. The manners of visiting the mesh levels are characterized by the cycle index $\mu$, which indicates the number of multigrid steps performed on a coarser level. The case $\mu=1$, is called V-cycle which is the simplest multigrid cycle, and $\mu=2$ leads to the so-called W-cycle. Another interesting case lies inbetween, so called F-cycle which we use in our application. The number of pre-smoothing steps $\nu_1 \in \mathbb{N}$ and post-smoothing steps $\nu_2 \in \mathbb{N}$, resp., is typically small, for instance between 1 and 4. Moreover, the coarsest grid should be coarse enough in order to be efficient.

The key ingredients in the multigrid methods are the smoothing, restriction and prolongation operators, which are sketched in the following.

#### 4.3.2. Local pressure Schur complement (LPSC) type Smoothers

The efficiency and robustness of the multigrid method is essentially influenced by the smoothing operators. In our numerical study, we employ a geometrical multigrid solver with a local pressure

Schur complement type smoother which was originally proposed by Vanka [63] for the solution of Navier-Stokes equations discretized with the finite difference method. These smoothers were originally developed to solve saddle point problems having a zero block appearing on the main diagonal of the system matrix, where standard smoothers such as Jacobi or Gauß-Seidel fails a usual situation in the field of computational fluid dynamics (CFD). The local pressure Schur complement type smoothers can be considered as block Gauß-Seidel methods, where in each smoothing step a local system of equations has to be solved exactly on fixed patches. The associated degrees of freedom are updated successively in a Gauß-Seidel manner. That is, the information which has been updated in previous element influences immediately all (velocity and pressure) degrees of freedom which are connected to current mesh cell. In every smoothing step, a loop over all mesh elements is performed (see [15, 65] for more details). Because the local pressure Schur complement type smoothers solve the problem element-wise, they are also called local smoothers ("local Multilevel Pressure Schur Complement (MPSC)" in [58]).

The idea of local MPSC smoother is to apply a defect correction of the type

$$x_{j+1} = x_j + \omega C^{-1}(b - Ax_j), \qquad \omega > 0$$

over all elements, where $C$ is an appropriate preconditioner. To explain the MPSC smoother in a more formal way, we consider the mesh $\Omega_H$ of the finite element space at given level $H$ with mesh size $h$ and $K \in \Omega_H$ be the element of $\Omega_H$. An outer loop over all elements $K \in \Omega_H$ is performed. A global defect is set up, with all components of the defect vector which do not belong the current element $K$ forced to zero. Consequently, the global defect can be reduced to local defect and global preconditioner to a local preconditioner for the all elements $K$.

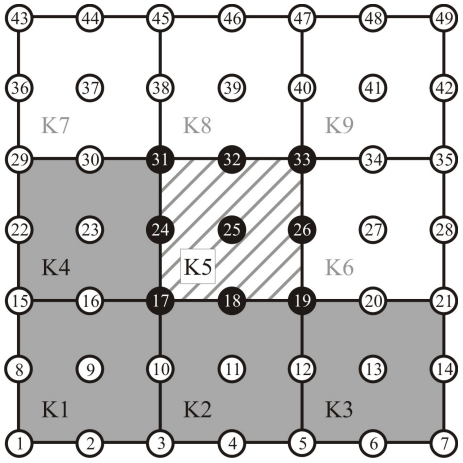



**Figure 4.1**: Local smoothing process for PSCS type smoother on element $K_5$.

To demonstrate the local smoothing process, we consider the problem discretized with $Q_2$ finite elements (see [37] for $Q_1$ discretization) with 9 mesh cells as shown in Figure 4.1. Then, the corresponding discrete system reads

$$Ax = b, \tag{4.2}$$

where $A \in \mathbb{R}^{49\times 49}$ is a system matrix and $x, b \in \mathbb{R}^{49}$ the solution and right hand side vectors, respectively. Let $I(K)$ denotes a list of all degrees of freedom located on an element $K$ and $A_{I(K)}$ be the matrix which corresponds to the rows and columns associated by the index set $I(K)$. Similarly, $x_{I(K)}, b_{I(K)}$ and $r_{I(K)} = (b - Ax)_{I(K)}$ represents the subvectors of $x, b$ and $r$ restricted to element $K$. The corresponding preconditioner $C_K^{-1}$ can be obtained by applying LU-decomposition (e.g., with LAPACK package) to the matrix $A_{I(K)}$.

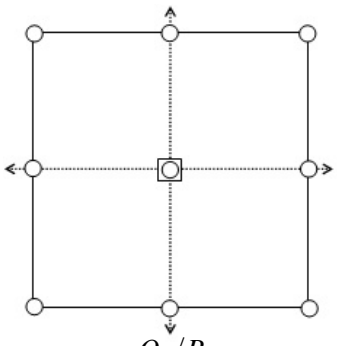
Let us assume there are 9 (local) degrees of freedom per element for simple presentation and the elements $K_1, K_2, K_3, K_4$ are already processed. Then, the degrees of freedom on the current element $K_5$ are updated as follows

$$\begin{pmatrix} x_{17} \\ x_{18} \\ x_{19} \\ x_{24} \\ x_{25} \\ x_{26} \\ x_{31} \\ x_{32} \\ x_{33} \end{pmatrix} \leftarrow \begin{pmatrix} x_{17} \\ x_{18} \\ x_{19} \\ x_{24} \\ x_{25} \\ x_{26} \\ x_{31} \\ x_{32} \\ x_{33} \end{pmatrix} + \omega C_5^{-1} \left( \begin{pmatrix} b_{17} \\ b_{18} \\ b_{19} \\ b_{24} \\ b_{25} \\ b_{26} \\ b_{31} \\ b_{32} \\ b_{33} \end{pmatrix} - \begin{pmatrix} a_{17,1} & \dots & a_{17,49} \\ a_{18,1} & \dots & a_{18,49} \\ a_{19,1} & \dots & a_{19,49} \\ a_{24,1} & \dots & a_{24,49} \\ a_{25,1} & \dots & a_{25,49} \\ a_{26,1} & \dots & a_{26,49} \\ a_{31,1} & \dots & a_{31,49} \\ a_{32,1} & \dots & a_{32,49} \\ a_{33,1} & \dots & a_{33,49} \end{pmatrix} \begin{pmatrix} x_1 \\ \\ . \\ \\ . \\ \\ . \\ \\ x_{49} \end{pmatrix} \right)$$

where

$$C_5 = \begin{pmatrix} a_{17,17} & a_{17,18} & a_{17,19} & a_{17,24} & a_{17,25} & a_{17,26} & a_{17,31} & a_{17,32} & a_{17,33} \\ a_{18,17} & a_{18,18} & a_{18,19} & a_{18,24} & a_{18,25} & a_{18,26} & a_{18,31} & a_{18,32} & a_{18,33} \\ a_{19,17} & a_{19,18} & a_{19,19} & a_{19,24} & a_{19,25} & a_{19,26} & a_{19,31} & a_{19,32} & a_{19,33} \\ a_{24,17} & a_{24,18} & a_{24,19} & a_{24,24} & a_{24,25} & a_{24,26} & a_{24,31} & a_{24,32} & a_{24,33} \\ a_{25,17} & a_{25,18} & a_{25,19} & a_{25,24} & a_{25,25} & a_{25,26} & a_{25,31} & a_{25,32} & a_{25,33} \\ a_{26,17} & a_{26,18} & a_{26,19} & a_{26,24} & a_{26,25} & a_{26,26} & a_{26,31} & a_{26,32} & a_{26,33} \\ a_{31,17} & a_{31,18} & a_{31,19} & a_{31,24} & a_{31,25} & a_{31,26} & a_{31,31} & a_{31,32} & a_{31,33} \\ a_{32,17} & a_{32,18} & a_{32,19} & a_{32,24} & a_{32,25} & a_{32,26} & a_{32,31} & a_{32,32} & a_{32,33} \\ a_{33,17} & a_{33,18} & a_{33,19} & a_{33,24} & a_{33,25} & a_{33,26} & a_{33,31} & a_{33,32} & a_{33,33} \end{pmatrix}$$

and $\omega > 0$ is a relaxation parameter. The entry $a_{i,j}$ represents the $i$-th row and $j$-th column of the matrix $A$. Continuing this process on the next element $K_6$, the unknowns connected to node $19, 20, 21, 26, 33$ will be updated again while the unknown on the node $27, 28, 34, 35$ will computed the first time and so on. Here, the size of the preconditioner $C$ depends on the problem. For the conforming LBB-stable finite element pair $Q_2/P_1^{disc}$, there are 18 velocity degrees of freedom and three pressure degrees of freedom if the cGP(1) or Crank-Nicolson method is applied for the time discretization. In the case of cGP(2) or dG(1)-method, the size of the block system doubles and the total velocity and pressure degree of freedom become 36 and six, respectively. Thus in a loop over all elements, linear system of size $21 \times 21$ and $42 \times 42$ have to be solved in the cGP(1) and cGP(2) case, respectively. Figure 4.2 describes the location of the local degrees of freedom for $Q_2/P_1^{disc}$ for the presented time discretization schemes.



$Q_2/P_1$

$$\text{cGP(1)} = \begin{cases} \circ & \underline{U}_n^1, \underline{V}_n^1 \\ \square & \underline{P}_n^1, \frac{\partial \underline{P}_n^1}{\partial x}, \frac{\partial \underline{P}_n^1}{\partial y} \end{cases}$$

$$\text{cGP(2)} = \begin{cases} \circ & \underline{U}_n^1, \underline{V}_n^1, \underline{U}_n^2, \underline{V}_n^2 \\ \square & \underline{P}_n^1, \frac{\partial \underline{P}_n^1}{\partial x}, \frac{\partial \underline{P}_n^1}{\partial y}, \underline{P}_n^2, \frac{\partial \underline{P}_n^2}{\partial x}, \frac{\partial \underline{P}_n^2}{\partial y} \end{cases}$$

**Figure 4.2**: Location of the degrees of freedom for the $Q_2/P_1$ element.

### 4.3.3. Restriction and prolongation

Multigrid methods require the transfer of information between the different mesh levels. To this end, we need to define some inter-grid transfer operators (*restriction and prolongation*). It is sufficient to consider these grid transfer operators only for two mesh levels $\Omega_k$ (fine) and $\Omega_{k-1}$ (coarse). The transfer of the residual from a finer grid to a coarser grid is referred to as restriction. The most common notation for the restriction operator is

$$I_k^{k-1} : \mathbb{R}^{N(k)} \mapsto \mathbb{R}^{N(k-1)}.$$

Prolongation is defined as the transfer of computed correction on the coarser grids to the finer grids. That is a function

$$I_{k-1}^{k} : \mathbb{R}^{N(k-1)} \mapsto \mathbb{R}^{N(k)}.$$

We use the canonical grid transfer routines based on FEM space which treat all solution components separately. In the case of conforming $Q_2$ finite elements, the prolongation operator is constructed by using a biquadratic interpolation. Let $\{v_i\}$ be the nodes of the coarse grid $k-1$ and $\{w_i\}$ the nodes of the fine grid $k$ as shown in Figure 4.3(left) (see [23, 38, 39] for the details). Then, the weights of the nodes $w_1, w_2, \ldots, w_9$ are computed as follows (see also Figure 4.3(left))

$$w_1 := v_1, \qquad w_2 := \frac{1}{8}(3v_1 + 6v_2 - v_3),$$
$$w_9 := \frac{1}{64}(9v_1 + 18v_2 - 3v_3 - 6v4 + v5 - 6v_6 - 3v_7 + 18v_8 + 36v_9).$$

The weights for remaining nodes $w_i$ can be computed similarly. The restriction is then set up as the adjoint of the prolongation operator, i.e., the matrices associated to $I_{k-1}^{k}$ and $I_k^{k-1}$ are exactly transposed to each other. Next, we describe the construction of the prolongation operator for the

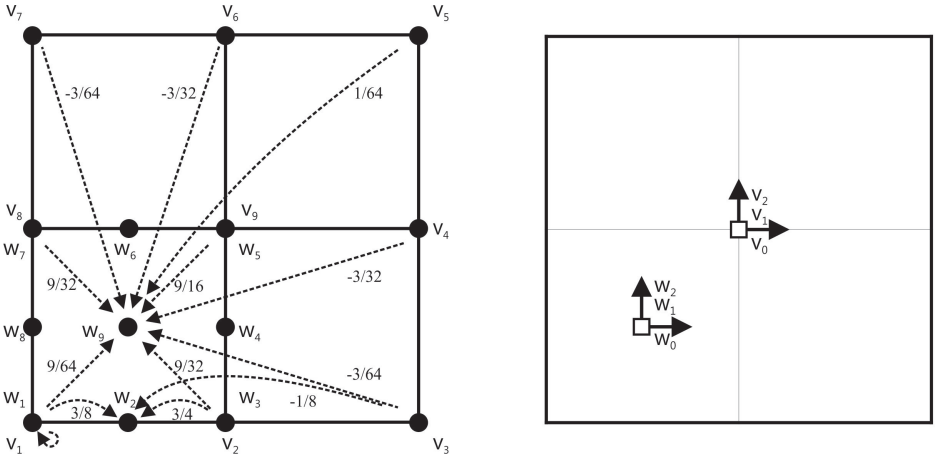



**Figure 4.3**: Prolongation in $Q_2$ with biquadratic interpolation (left) and $P_1$ with linear interpolation (right).

discontinuous $P_1$ finite element which can be done with an associated linear interpolation. To this end, we suppose that $v_0, v_1, v_2$ denote the nodal value and its derivatives w.r.t. $x$ and $y$, respectively, on the coarse grid $k-1$ and $w_0, w_1, w_2$ be the corresponding values on the fine grid $k$ as shown in Figure 4.3 (right). Then, the weights of the nodes $w_i$ are computed as follows

$$w_0 := v_0 - \frac{1}{2}v_1 - \frac{1}{2}v_2, \qquad w_1 := v_1 \qquad w_2 := v_2.$$

Here, the values of $w_1, w_2$ will remain the same $v_1, v_2$, respectively, since the derivative of linear function is always constant. Again, the restriction is set up as adjoint of the prolongation operator.

# 5

# Galerkin time discretizations for the heat equation using Gauß-points

In this chapter, we start by introducing the continuous Galerkin-Petrov and discontinuous Galerkin time discretizations applied to the heat equation as a prototypical example for scalar parabolic partial differential equations. For the space discretization, we use biquadratic quadrilateral finite elements on general two-dimensional meshes. We discuss implementational aspects of the time discretization as well as efficient methods for solving the resulting block systems. By means of numerical experiments we compare the different time discretizations with respect to accuracy and computational costs.

To solve the associated linear block systems, we apply a preconditioned BiCGStab solver as standard Krylov space method and a geometrical (block) multigrid solver. Furthermore, we also compare a preconditioned BiCGStab solver with an adapted geometrical multigrid solver. Only the convergence of the multigrid method is almost independent of the mesh size and the time step leading to an efficient solution process.

## 5.1. The cGP-method for the heat equation

As a model problem we consider the heat equation: *Find* $u : \Omega \times [0,T] \to \mathbb{R}$ *such that*

$$\begin{aligned} \frac{\partial u}{\partial t} - \triangle u &= f && \text{in} \quad \Omega \times (0,T), \\ u &= 0 && \text{on} \quad \partial\Omega \times [0,T], \\ u(x,0) &= u_0(x) && \text{for} \quad x \in \Omega, \end{aligned} \tag{5.1}$$

where $u(x,t)$ denotes the temperature in the point $x \in \Omega$ at time $t \in [0,T]$, $f : \Omega \times (0,T) \to \mathbb{R}$ a given source term and $u_0 : \Omega \to \mathbb{R}$ the initial temperature field at time $t = 0$. For simplicity, we assume homogeneous Dirichlet conditions at the boundary $\partial\Omega$ of a polygonal domain $\Omega \subset \mathbb{R}^2$.

We start with the time discretization of problem (5.1) which is of variational type. In the following, let $I = [0,T]$ be the time interval with some positive final time $T$. For a function $u : \Omega \times I \to \mathbb{R}$ and a fixed $t \in I$ we will denote by $u(t) := u(\cdot,t)$ the associated space function at time $t$ which is an element of a suitable function space $V$. In case of the heat equation, this space is the Sobolev space $V = H_0^1(\Omega)$. In order to characterize functions $t \mapsto u(t)$ we define the space $C(I,V)$ as the space of continuous functions $u : I \to V$ equipped with the norm

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

and the space $L^2(I,V)$ containing discontinuous functions as

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

In the time discretization, we decompose the time interval $I$ into $N$ subintervals $I_n := [t_{n-1}, t_n]$, where $n = 1, \ldots, N$ and $0 = t_0 < t_1 < \cdots < t_{N-1} < t_N = T$. The symbol $\tau$ will denote the *time discretization parameter* and will also be used as the maximum time step size $\tau := \max_{1 \le n \le N} \tau_n$, where $\tau_n := t_n - t_{n-1}$. Then, we approximate the solution $u : I \to V$ by means of a function $u_\tau : I \to V$ which is piecewise polynomial of some order $k$ with respect to time, i.e., we are looking for $u_\tau$ in the discrete time space

$$X_\tau^k := \{u \in C(I,V) : \, u\big|_{I_n} \in \mathbb{P}_k(I_n,V) \quad \forall\, n = 1, \ldots, N\}, \tag{5.2}$$

where

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

We introduce the discrete time test space

$$Y_\tau^k := \{v \in L^2(I,V) : \, v\big|_{I_n} \in \mathbb{P}_{k-1}(I_n,V) \quad \forall\, n = 1, \ldots, N\} \tag{5.3}$$

consisting of piecewise polynomials of order $k-1$ which are globally discontinuous at the end points of the time intervals. Now, we multiply the first equation in (5.1) with a test function $v_\tau \in Y_\tau^k$, integrate over $\Omega \times I$, use Fubini's Theorem and partial space integration of the Laplacian term and obtain the following *time discrete problem*: *Find* $u_\tau \in X_\tau^k$ *such that* $u_\tau(0) = u_0$ *and*

$$\int_0^T \Big\{ (d_t u_\tau(t), v_\tau(t))_\Omega + a(u_\tau(t), v_\tau(t)) \Big\} dt = \int_0^T (f(t), v_\tau(t))_\Omega \, dt \qquad \forall\, v_\tau \in Y_\tau^k, \tag{5.4}$$

where $(\cdot,\cdot)_\Omega$ denotes the usual inner product in $L^2(\Omega)$ and $a(\cdot,\cdot)$ the bilinear form on $V \times V$ defined as

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

We will call this discretization the *exact continuous Galerkin-Petrov method* of *order k* or briefly the "*exact* cGP($k$)*-method*". The name Galerkin-Petrov is due to the fact that the test space $Y_\tau^k$ is different from the ansatz space $X_\tau^k$. With "exact" we indicate that the time integral at the right hand side in (5.4) is evaluated exactly.

Since the discrete test space $Y_\tau^k$ is discontinuous, problem (5.4) can be solved in a time marching process where successively local problems on the time intervals are solved. Therefore, we choose test functions $v_\tau(t) = v\psi_{n,i}(t)$ with an arbitrary time independent $v \in V$ and a scalar function $\psi_{n,i} : I \to \mathbb{R}$ which is zero on $I \setminus I_n$ and a polynomial of order less or equal $k-1$ on $I_n$. Then, we obtain from (5.4) the "$I_n$-problem": *Find* $u_\tau|_{I_n} \in \mathbb{P}_k(I_n,V)$ *such that*

$$\int_{I_n} \Big\{ (d_t u_\tau(t), v)_\Omega + a(u_\tau(t), v) \Big\} \psi_{n,i}(t) \, dt = \int_{I_n} (f(t), v)_\Omega \, \psi_{n,i}(t) \, dt \qquad \forall\, v \in V \tag{5.5}$$

for $i = 1, \ldots, k$, with the "initial condition" $u_\tau|_{I_n}(t_{n-1}) = u_\tau|_{I_{n-1}}(t_{n-1})$ for $n \ge 2$ or $u_\tau|_{I_n}(t_{n-1}) = u_0$ for $n = 1$.

To determine $u_\tau|_{I_n}$ we represent it by a polynomial ansatz

$$u_\tau(t) := \sum_{j=0}^{k} U_n^j \phi_{n,j}(t) \qquad \forall\, t \in I_n, \tag{5.6}$$

where the "coefficients" $U_n^j$ are elements of the Hilbert space $V$ and the real functions $\phi_{n,j} \in \mathbb{P}_k(I_n)$ are the Lagrange basis functions with respect to $k+1$ suitable nodal points $t_{n,j} \in I_n$ satisfying the conditions

$$\phi_{n,j}(t_{n,i}) = \delta_{i,j}, \qquad i, j = 0, \ldots, k \tag{5.7}$$

with the Kronecker symbol $\delta_{i,j}$. In [49], the $t_{n,j}$ have been chosen as the quadrature points of the $(k+1)$-point Gauß-Lobatto formula on $I_n$. Here, we take another choice: For an easy treatment of the initial condition for (5.5), we set $t_{n,0} = t_{n-1}$. Then, the initial condition is equivalent to the condition

$$U_n^0 = u_\tau|_{I_{n-1}}(t_{n-1}) \quad \text{if} \quad n \geq 2 \qquad \text{or} \qquad U_n^0 = u_0 \quad \text{if} \quad n = 1. \tag{5.8}$$

The other points $t_{n,1}, \ldots, t_{n,k}$ are chosen as the quadrature points of the $k$-point Gauß formula on $I_n$. This formula is exact if the function to be integrated is a polynomial of degree less or equal $2k-1$. From the representation (5.6) we get

$$\int_{I_n} (d_t u_\tau(t), v)_\Omega \, \psi_{n,i}(t) dt = \sum_{j=0}^{k} \left(U_n^j, v\right)_\Omega \int_{I_n} \phi'_{n,j}(t) \psi_{n,i}(t) \, dt \qquad \forall\, v \in V. \tag{5.9}$$

We define the basis functions $\phi_{n,j} \in \mathbb{P}_k(I_n)$ of (5.6) via the affine reference transformation $T_n : \hat{I} \to I_n$ where $\hat{I} := [-1,1]$ and

$$t = T_n(\hat{t}) := \frac{t_{n-1} + t_n}{2} + \frac{\tau_n}{2}\hat{t} \in I_n \qquad \forall\, \hat{t} \in \hat{I},\ n = 1, \ldots, N. \tag{5.10}$$

Let $\hat{\phi}_j \in \mathbb{P}_k(\hat{I})$, $j = 0, \ldots, k$, denote the basis functions satisfying the conditions

$$\hat{\phi}_j(\hat{t}_i) = \delta_{i,j}, \qquad i, j = 0, \ldots, k, \tag{5.11}$$

where $\hat{t}_0 = -1$ and $\hat{t}_i$, $i = 1, \ldots, k$, are the standard *Gauß* quadrature points for the reference interval $\hat{I}$. Then, we define the basis functions on the original time interval $I_n$ by

$$\phi_{n,j}(t) := \hat{\phi}_j(\hat{t}) \qquad \text{with} \qquad \hat{t} := T_n^{-1}(t) = \frac{2}{\tau_n}\left(t - \frac{t_n - t_{n-1}}{2}\right) \in \hat{I}. \tag{5.12}$$

Similarly, we define the test basis functions $\psi_{n,i}$ by suitable reference basis functions $\hat{\psi}_i \in \mathbb{P}_{k-1}(\hat{I})$, i.e.,

$$\psi_{n,i}(t) := \hat{\psi}_i(T_n^{-1}(t)) \qquad \forall\, t \in I_n,\ i = 1, \ldots, k. \tag{5.13}$$

For practical computations, we have to approximate the right hand side in the exact cGP($k$)-method (5.5) by some numerical integration. To this end, we replace the function $f(t)$ by the time-polynomial $\pi_k f \in \mathbb{P}_k(I_n, L^2(\Omega))$ defined as the Lagrange interpolate

$$\pi_k f(t) := \sum_{j=0}^{k} f(t_{n,j}) \phi_{n,j}(t) \qquad \forall\, t \in I_n.$$

Now, we transform all integrals in (5.5) to the reference interval $\hat{I}$ and obtain the following system of equations for the "coefficients" $U_n^j \in V$ in the ansatz (5.6)

$$\sum_{j=0}^{k} \left\{ \alpha_{i,j} \left(U_n^j, v\right)_\Omega + \frac{\tau_n}{2} \beta_{i,j} a(U_n^j, v) \right\} = \frac{\tau_n}{2} \sum_{j=0}^{k} \beta_{i,j} \left(f(t_{n,j}), v\right)_\Omega \qquad \forall\, v \in V \tag{5.14}$$

where $i = 1, \ldots, k$,

$$\alpha_{i,j} := \int_{\hat{I}} \hat{\phi}'_j(\hat{t}) \hat{\psi}_i(\hat{t}) \, d\hat{t}, \qquad \beta_{i,j} := \int_{\hat{I}} \hat{\phi}_j(\hat{t}) \hat{\psi}_i(\hat{t}) \, d\hat{t} \tag{5.15}$$

and the "coefficient" $U_n^0 \in V$ is known. Due to the polynomial degree of $\hat{\phi}_j$ and $\hat{\psi}_i$ we can compute the integrals for $\alpha_{i,j}$ and $\beta_{i,j}$ exactly by means of the $k$-point Gauß formula with weights $\hat{w}_\mu$ and points $\hat{t}_\mu$, $\mu = 1, \ldots, k$. If we choose the test functions $\hat{\psi}_i \in \mathbb{P}_{k-1}(\hat{I})$ in (5.15) such that

$$\hat{\psi}_i(\hat{t}_\mu) = (\hat{w}_i)^{-1} \delta_{i,\mu} \qquad \forall\, i, \mu \in \{1, \ldots, k\},$$

we get from (5.15) that

$$\alpha_{i,j} = \hat{\phi}_j'(\hat{t}_i), \qquad \beta_{i,j} = \delta_{i,j}, \qquad 1 \le i \le k,\ 0 \le j \le k.$$

Then, the system (5.14) is equivalent to the following coupled system of equations for the $k$ unknown "coefficients" $U_n^j \in V$, $j = 1,\dots,k$,

$$\sum_{j=0}^{k} \alpha_{i,j} \left(U_n^j, v\right)_\Omega + \frac{\tau_n}{2} a(U_n^i, v) = \frac{\tau_n}{2} \left(f(t_{n,i}), v\right)_\Omega \qquad \forall\, v \in V, \quad i = 1,\dots,k, \tag{5.16}$$

where $U_n^0 = u_\tau(t_{n-1})$ for $n > 1$ and $U_1^0 = u_0$. In the following, we specify the cGP($k$)-method for the cases $k = 1$ and $k = 2$.

### 5.1.1. cGP(1)-method

We use the one-point Gauß quadrature formula with the point $\hat{t}_1 = 0$ and $t_{n,1} = t_{n-1} + \frac{\tau_n}{2}$. Then, we get $\alpha_{1,0} = -1$ and $\alpha_{1,1} = 1$. Thus, equation (5.16) leads to the following equation for the one "unknown" $U_n^1 = u_\tau(t_{n-1} + \frac{\tau_n}{2}) \in V$

$$\left(U_n^1, v\right)_\Omega + \frac{\tau_n}{2} a(U_n^1, v) = \frac{\tau_n}{2} \left(f(t_{n,1}), v\right)_\Omega + \left(U_n^0, v\right)_\Omega \qquad \forall\, v \in V. \tag{5.17}$$

Once we have determined the solution $U_n^1$, we get the solution at discrete time $t_n$ by means of linear extrapolation

$$u_\tau(t_n) = 2U_n^1 - U_n^0, \tag{5.18}$$

where $U_n^0$ is the initial value at the time interval $[t_{n-1}, t_n]$ coming from the previous time interval or the initial value $u_0$. If we would replace $f(t_{n,1})$ by the mean value $(f(t_{n-1}) + f(t_n))/2$, which means that we replace the one-point Gauß quadrature of the right hand side by the Trapezoidal rule, the resulting cGP(1)-method would be equivalent to the well-known *Crank-Nicolson scheme*.

### 5.1.2. cGP(2)-method

We use the 2-point Gauß quadrature formula with the points $\hat{t}_1 = -\frac{1}{\sqrt{3}}$ and $\hat{t}_2 = \frac{1}{\sqrt{3}}$. Then, we obtain the coefficients

$$(\alpha_{i,j}) = \begin{pmatrix} -\sqrt{3} & \frac{3}{2} & \frac{2\sqrt{3}-3}{2} \\ \sqrt{3} & \frac{-2\sqrt{3}-3}{2} & \frac{3}{2} \end{pmatrix} \qquad i = 1,2,\ j = 0,1,2.$$

On the time interval $I_n = [t_{n-1}, t_n]$ we have to solve for the two "unknowns" $U_n^j = u_\tau(t_{n,j})$ with $t_{n,j} := T_n(\hat{t}_j)$ for $j = 1,2$. The coupled system for $U_n^1, U_n^2 \in V$ reads

$$\begin{aligned} \left\{\alpha_{1,1} \left(U_n^1, v\right)_\Omega + \tfrac{\tau_n}{2} a(U_n^1, v)\right\} + \alpha_{1,2} \left(U_n^2, v\right)_\Omega &= \tfrac{\tau_n}{2} \left(f(t_{n,1}), v\right)_\Omega - \alpha_{1,0} \left(U_n^0, v\right)_\Omega, \\ \alpha_{2,1} \left(U_n^1, v\right)_\Omega + \left\{\alpha_{2,2} \left(U_n^2, v\right)_\Omega + \tfrac{\tau_n}{2} a(U_n^2, v)\right\} &= \tfrac{\tau_n}{2} \left(f(t_{n,2}), v\right)_\Omega - \alpha_{2,0} \left(U_n^0, v\right)_\Omega, \end{aligned} \tag{5.19}$$

which has to be satisfied for all $v \in V$. Once we have determined the solution $(U_n^1, U_n^2)$, we get the solution at discrete time $t_n$ by means of quadratic extrapolation

$$u_\tau(t_n) = U_n^0 + \sqrt{3}(U_n^2 - U_n^1), \tag{5.20}$$

where $U_n^0$ is the initial value at the time interval $I_n$ coming from the previous time interval or the initial value $u_0$.

## 5.2. Discontinuous Galerkin methods

In this section we describe the details of the discontinuous Galerkin method dG($k$-1) which is with respect to the size of the coupled system comparable to the cGP($k$)-method. Here the discrete solution space is the same as the test space $Y_\tau^k$ of the cGP($k$)-method defined in (B.2). Therefore, the discrete solution $u_\tau$ is on each time interval $I_n$ a polynomial in time of degree $k-1$ with a representation

$$u_\tau(t) := \sum_{j=1}^{k} U_n^j \phi_{n,j}(t) \qquad \forall\, t \in I_n, \tag{5.21}$$

where the "coefficients" $U_n^j$ are elements of the Hilbert space $V$ and the real functions $\phi_{n,j} \in \mathbb{P}_{k-1}(I_n)$ are the Lagrange basis functions with respect to $k$ suitable nodal points $t_{n,j} \in I_n$ satisfying the conditions $\phi_{n,j}(t_{n,i}) = \delta_{i,j}$ for all $i,j = 1,\dots,k$. Since $u_\tau$ is discontinuous at $t_n$, we define the following left and right sided values $u_n^-$ and $u_n^+$ and the jump $[u_\tau]_n$ as:

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

Then, the discontinuous Galerkin method dG($k$-1) reads: *Find $u_\tau \in Y_\tau^k$ such that $u_\tau(0) = u_0$ and for all $v_\tau \in Y_\tau^k$ it holds*

$$\sum_n \int_{I_n} \Big\{ (d_t u_\tau(t), v_\tau(t))_\Omega + a(u_\tau(t), v_\tau(t)) \Big\} dt + \sum_n \big( [u_\tau]_{n-1}, v_{n-1}^+ \big)_\Omega = \int_0^T (f(t), v_\tau(t))_\Omega \, dt.$$

To decouple this formulation we choose test functions $v_\tau(t) = v\psi_{n,i}(t)$ with an arbitrary $t$-independent $v \in V$ and a scalar piecewise polynomial function $\psi_{n,i}$ which is zero on $I \setminus I_n$ and a basis function of the polynomial space $\mathbb{P}_{k-1}$ on $I_n$. Then, using the fact that $U_n^0 := u_{n-1}^-$ is known from the previous time interval or the initial value $u_0$, the solution of the dG($k$-1)-method can be determined by successively solving a local problem on each time interval $I_n$. This "$I_n$-problem" reads: *Find $u_\tau \in \mathbb{P}_{k-1}(I_n, V)$ such that for all $v \in V$ and all $i = 1,\dots,k$ holds*

$$\begin{aligned} &\int_{I_n} \Big\{ (d_t u_\tau(t), v)_\Omega + a(u_\tau(t), v) \Big\} \psi_{n,i}(t)\, dt + \big( u_{n-1}^+, v \big)_\Omega \psi_{n,i}(t_{n-1}) \\ &\qquad = \big( U_n^0, v \big)_\Omega \psi_{n,i}(t_{n-1}) + \int_{I_n} (f(t), v)_\Omega \, \psi_{n,i}(t)\, dt. \end{aligned} \tag{5.22}$$

As in the cGP-method we define the basis functions $\phi_{n,j} \in \mathbb{P}_{k-1}(I_n)$ in (5.21) by means of reference basis functions $\hat\phi_j \in \mathbb{P}_{k-1}(\hat I)$ via the transformation $T_n : \hat I \to I_n$ given in (5.10). The $\hat\phi_j$, $j = 1,\dots,k$, are chosen as the Lagrange basis functions associated with the integration points $\hat t_\mu$ of the $k$-point Gauß quadrature rule on $\hat I = [-1,1]$, i.e., they satisfy the conditions $\hat\phi_j(\hat t_\mu) = \delta_{j,\mu}$ for all $j,\mu = 1,\dots,k$. Similarly, we define the test basis functions $\psi_{n,i}$ by means of reference basis functions $\hat\psi_i \in \mathbb{P}_{k-1}(\hat I)$. We use the choice $\hat\psi_i := (\hat w_i)^{-1}\hat\phi_i$ where the $\hat w_i$ denote the weights of the the $k$-point Gauß formula on $\hat I$.

Now, if we insert the representation (5.21) of $u_\tau$ into the $I_n$-problem (5.22) and transform the integrals to the reference interval $\hat I$, we obtain the following system of equations for the "coefficients" $U_n^j \in V$, $j = 1,\dots,k$

$$\sum_{j=1}^{k} \Big\{ \alpha_{i,j} \big( U_n^j, v \big)_\Omega + \frac{\tau_n}{2} \beta_{i,j} a(U_n^j, v) + c_j d_i \big( U_n^j, v \big)_\Omega \Big\} = d_i \big( U_n^0, v \big)_\Omega + \frac{\tau_n}{2} \int_{\hat I} \big( \hat f, v \big)_\Omega \hat\psi_i d\hat t,$$

where $i = 1,\dots,k$ and

$$\alpha_{i,j} := \int_{\hat I} \hat\phi_j'(\hat t)\hat\psi_i(\hat t)\, d\hat t, \qquad \beta_{i,j} := \int_{\hat I} \hat\phi_j(\hat t)\hat\psi_i(\hat t)\, d\hat t, \qquad c_j := \hat\phi_j(-1), \qquad d_i := \hat\psi_i(-1).$$

We approximate the integral on the right hand side by the $k$-point Gauß formula and get due to the special choice of $\hat{\psi}_i$

$$\int_{\hat{I}} \left(\hat{f}(\hat{t}),v\right)_\Omega \hat{\psi}_i(\hat{t})\,d\hat{t} \;\approx\; \sum_{\mu=1}^{k} \hat{w}_\mu \left(f(t_{n,\mu}),v\right)_\Omega \hat{\psi}_i(\hat{t}_\mu) = \left(f(t_{n,i}),v\right)_\Omega, \tag{5.23}$$

where the $t_{n,\mu} := T_n(\hat{t}_\mu) \in I_n$ denote the mapped Gauß points. Since the $k$-point Gauß quadrature formula is exact for the integrals defining $\alpha_{i,j}$ and $\beta_{i,j}$, we conclude from the Kronecker delta properties of $\hat{\phi}_j$ and $\hat{\psi}_i$ that

$$\alpha_{i,j} = \hat{\phi}_j'(\hat{t}_i), \qquad \beta_{i,j} = \delta_{i,j}, \qquad c_j = \hat{\phi}_j(-1), \qquad d_i = \frac{1}{\hat{w}_i} c_i \tag{5.24}$$

for $i,j = 1,\ldots,k$.

Summarizing all pieces, we get the following version of the dG($k$-1)-method with numerically integrated right hand side on the time interval $I_n = [t_{n-1},t_n]$. For a given value $U_n^0 := u_{n-1}^-$ known from the previous time interval or the initial value $u_0$, find $k$ unknowns $U_n^1,\ldots,U_n^k \in V$ such that for all $v \in V$ and all $i = 1,\ldots,k$, holds

$$\sum_{j=1}^{k} \left(\alpha_{i,j} + c_j d_i\right) \left(U_n^j,v\right)_\Omega + \frac{\tau_n}{2} a(U_n^i,v) = d_i \left(U_n^0,v\right)_\Omega + \frac{\tau_n}{2} \left(f(t_{n,i}),v\right)_\Omega. \tag{5.25}$$

Once we have solved this system, we can compute by means of the ansatz (5.21) the left side value $u_n^-$ of $u_\tau$ at time $t_n$. Then, we enter the next time interval $I_{n+1} = [t_n,t_{n+1}]$ and set the initial value to $U_{n+1}^0 := u_n^-$.

Finally, we will specify the case $k = 2$ leading to the dG(1)-method which we will use in our numerical experiments. Here, the discrete solution $u_\tau$ is piecewise linear, i.e., we have to compute two "coefficients" on each time interval to define the linear approximation. The associated 2-point Gauß quadrature formula has the weights $\hat{w}_1 = \hat{w}_2 = 1$ and integration points $\hat{t}_1 = -\frac{1}{\sqrt{3}}$, $\hat{t}_2 = \frac{1}{\sqrt{3}}$. With the abbreviation $\gamma_{i,j} := \alpha_{i,j} + c_j d_i$ we obtain for the coefficients in (5.25) the values

$$(\alpha_{i,j}) = \begin{pmatrix} \frac{-\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ \frac{-\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \end{pmatrix}, \quad (c_i) = (d_i) = \begin{pmatrix} \frac{\sqrt{3}+1}{2} \\ \frac{-\sqrt{3}+1}{2} \end{pmatrix}, \quad (\gamma_{i,j}) = \begin{pmatrix} 1 & \frac{\sqrt{3}-1}{2} \\ \frac{-\sqrt{3}-1}{2} & 1 \end{pmatrix}.$$

Thus, in the **dG(1)-method**, one has to determine on the time interval $I_n$ the two "unknowns" $U_n^1, U_n^2 \in V$ as the solution of the following coupled system

$$\begin{aligned} \left\{\gamma_{1,1}\left(U_n^1,v\right)_\Omega + \tfrac{\tau_n}{2} a(U_n^1,v)\right\} + \gamma_{1,2}\left(U_n^2,v\right)_\Omega &= d_1\left(U_n^0,v\right)_\Omega + \tfrac{\tau_n}{2}\left(f(t_{n,1}),v\right)_\Omega, \\ \gamma_{2,1}\left(U_n^1,v\right)_\Omega + \left\{\gamma_{2,2}\left(U_n^2,v\right)_\Omega + \tfrac{\tau_n}{2} a(U_n^2,v)\right\} &= d_2\left(U_n^0,v\right)_\Omega + \tfrac{\tau_n}{2}\left(f(t_{n,2}),v\right)_\Omega, \end{aligned} \tag{5.26}$$

which has to be satisfied for all $v \in V$. Once we have solved the above system, we get the solution at the time $t_n$ by means of the following linear extrapolation with the "values" $U_n^j = u_\tau(t_{n,j})$, $j = 1,2$

$$u_\tau(t_n) = \frac{\sqrt{3}+1}{2} U_n^2 - \frac{\sqrt{3}-1}{2} U_n^1. \tag{5.27}$$

### 5.3. Space Discretization by FEM

After discretizing equation (5.1) in time, we now apply the finite element method to discretize each of the "$I_n$-problem" in space. To this end, let $V_h \subset V$ denote a finite element space. In the numerical

experiments, $V_h$ will be defined by biquadratic finite elements on a quadrilateral mesh $T_h$ for the computational domain $\Omega$. Each "$I_n$-problem" for the cGP($k$)-method or the dG($k$-1)-method has the structure: *For given $U_n^0 \in V$, find $U_n^1, \ldots, U_n^k \in V$ such that*

$$\sum_{j=1}^{k} \gamma_{i,j} \left(U_n^j, v\right)_\Omega + \frac{\tau_n}{2} a(U_n^i, v) = d_i \left(U_n^0, v\right)_\Omega + \frac{\tau_n}{2} \left(f(t_{n,i}), v\right)_\Omega \qquad \forall\, v \in V, \tag{5.28}$$

for all $i = 1, \ldots, k$, where $\gamma_{i,j}$ and $d_i$ are given constants. In the space discretization, each $U_n^j \in V$ is approximated by a finite element function $U_{n,h}^j \in V_h$ and the fully discrete "$I_n$-problem" reads: *For given $U_{n,h}^0 \in V_h$, find $U_{n,h}^1, \ldots, U_{n,h}^k \in V_h$ such that*

$$\sum_{j=1}^{k} \gamma_{i,j} \left(U_{n,h}^j, v_h\right)_\Omega + \frac{\tau_n}{2} a(U_{n,h}^i, v_h) = d_i \left(U_{n,h}^0, v_h\right)_\Omega + \frac{\tau_n}{2} \left(f(t_{n,i}), v_h\right)_\Omega \qquad \forall\, v_h \in V_h, \tag{5.29}$$

for all $i = 1, \ldots, k$. Once we have solved this system, we can compute for each time $t \in I_n$ a finite element approximation $u_{\tau,h}(t) \in V_h$ of the time discrete solution $u_\tau(t) \in V$. To this end, we replace the "constants" $U_n^j \in V$ in the ansatz of $u_\tau(t)$ by the space discrete "constants" $U_{n,h}^j \in V_h$.

In the following, we will write the problem (5.29) as a linear algebraic block system. Let $b_\mu \in V_h$, $\mu = 1, \ldots, m_h$, denote the finite element basis functions and $\underline{U}_n^j \in \mathbb{R}^{m_h}$ the nodal vector of $U_{n,h}^j \in V_h$ such that

$$U_{n,h}^j(x) = \sum_{\mu=1}^{m_h} (\underline{U}_n^j)_\mu b_\mu(x) \qquad \forall\, x \in \Omega.$$

Furthermore, let us introduce 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 vector $F_n^i \in \mathbb{R}^{m_h}$ as

$$M_{\nu,\mu} := (b_\mu, b_\nu)_\Omega, \qquad L_{\nu,\mu} := a(b_\mu, b_\nu), \qquad (F_n^i)_\nu := \left(f(t_{n,i}), b_\nu\right)_\Omega. \tag{5.30}$$

Then the fully discrete "$I_n$-problem" is equivalent to the following linear $k \times k$ block system: *For given $\underline{U}_n^0 \in \mathbb{R}^{m_h}$, find $\underline{U}_n^1, \ldots, \underline{U}_n^k \in \mathbb{R}^{m_h}$ such that*

$$\sum_{j=1}^{k} \gamma_{i,j} M \underline{U}_n^j + \frac{\tau_n}{2} L \underline{U}_n^i = d_i M \underline{U}_n^0 + \frac{\tau_n}{2} F_n^i, \qquad \forall\, i = 1, \ldots, k. \tag{5.31}$$

The vector $\underline{U}_n^0$ is defined as the finite element nodal vector of the fully discrete solution $u_{\tau,h}(t_{n-1})$ computed from the previous time interval $[t_{n-2}, t_{n-1}]$ if $n \geq 2$ or from a finite element interpolation of the initial data $u_0$ if $n = 1$.

In the following, we will present the resulting block systems for the cGP(1), cGP(2) and dG(1) method which are used in our numerical experiments.

### 5.3.1. cGP(1)-method

The problem on time interval $I_n$ reads: *For given $\underline{U}_n^0 \in \mathbb{R}^{m_h}$, find $\underline{U}_n^1 \in \mathbb{R}^{m_h}$ such that*

$$\left(M + \frac{\tau_n}{2} L\right) \underline{U}_n^1 = \frac{\tau_n}{2} F_n^1 + M \underline{U}_n^0. \tag{5.32}$$

Once we have determined the solution $\underline{U}_n^1$, we compute the nodal vector $\underline{U}_{n+1}^0$ of the fully discrete solution $u_{\tau,h}$ at the time $t_n$ by using the following linear extrapolation

$$u_{\tau,h}(t_n) \sim \underline{U}_{n+1}^0 = 2\underline{U}_n^1 - \underline{U}_n^0.$$

### 5.3.2. cGP(2)-method

The $2 \times 2$ block system on time interval $I_n$ reads: *For given* $\underline{U}_n^0 \in \mathbb{R}^{m_h}$, *find* $\underline{U}_n^1, \underline{U}_n^2 \in \mathbb{R}^{m_h}$ *such that*

$$\begin{pmatrix} 3M + \tau_n L & \left(2\sqrt{3}-3\right) M \\ \left(-2\sqrt{3}-3\right) M & 3M + \tau_n L \end{pmatrix} \begin{pmatrix} \underline{U}_n^1 \\ \underline{U}_n^2 \end{pmatrix} = \begin{pmatrix} R_n^1 \\ R_n^2 \end{pmatrix} \tag{5.33}$$

where

$$\begin{aligned} R_n^1 &= \tau_n F_n^1 + 2\sqrt{3} M \underline{U}_n^0 \\ R_n^2 &= \tau_n F_n^2 - 2\sqrt{3} M \underline{U}_n^0. \end{aligned}$$

Once we have determined the solution $(\underline{U}_n^1, \underline{U}_n^2)$, we compute the nodal vector $\underline{U}_{n+1}^0$ of the fully discrete solution $u_{\tau,h}$ at the time $t_n$ by using the following quadratic extrapolation

$$u_{\tau,h}(t_n) \sim \underline{U}_{n+1}^0 = \underline{U}_n^0 + \sqrt{3}(\underline{U}_n^2 - \underline{U}_n^1).$$

### 5.3.3. dG(1)-method

The $2 \times 2$ block system on time interval $I_n$ reads: *For given* $\underline{U}_n^0 \in \mathbb{R}^{m_h}$, *find* $\underline{U}_n^1, \underline{U}_n^2 \in \mathbb{R}^{m_h}$ *such that*

$$\begin{pmatrix} 2M + \tau_n L & \left(\sqrt{3}-1\right) M \\ \left(-\sqrt{3}-1\right) M & 2M + \tau_n L \end{pmatrix} \begin{pmatrix} \underline{U}_n^1 \\ \underline{U}_n^2 \end{pmatrix} = \begin{pmatrix} R_n^1 \\ R_n^2 \end{pmatrix} \tag{5.34}$$

where

$$\begin{aligned} R_n^1 &= \tau_n F_n^1 + \left(\sqrt{3}+1\right) M \underline{U}_n^0 \\ R_n^2 &= \tau_n F_n^2 + \left(-\sqrt{3}+1\right) M \underline{U}_n^0. \end{aligned}$$

Once we have determined the solution $(\underline{U}_n^1, \underline{U}_n^2)$, we compute the nodal vector $\underline{U}_{n+1}^0$ of the left side limit of the fully discrete solution $u_{\tau,h}$ at the time $t_n$ by using the following linear extrapolation

$$u_{\tau,h}^-(t_n) \sim \underline{U}_{n+1}^0 = \frac{\sqrt{3}+1}{2} U_n^2 - \frac{\sqrt{3}-1}{2} U_n^1.$$

## 5.4. Solution of the linear systems

The resulting linear systems in each time interval $[t_{n-1}, t_n]$, which are 2x2 block matrices in the case of the cGP(2) and dG(1) approach, are treated either by preconditioned Krylov-space or geometrical multigrid solvers (explained in Chapter 4). Among the Krylov-space methods we employ the classical BiCGStab solver (which is chosen in view of future nonsymmetric matrices due to convection-diffusion or even the Navier-Stokes equations) with the SSOR method (see Chapter 4) as preconditioner. However, such 'single grid' methods cannot treat the associated systems in an optimally efficient manner since the corresponding condition numbers depend also on the mesh size so that it is expected to become very expensive for higher mesh levels.

To overcome this difficulty, we utilize a geometrical multigrid solver with corresponding (block) smoothers and grid transfer operators. Multigrid methods are regarded as the most efficient iterative methods for the solution of large linear systems arising from the discretization of partial differential equations, particularly of elliptic type. In this chapter, the multigrid solver uses the same SSOR method as smoother. However, the standard block variants of Jacobi, SOR and ILU methods can be easily applied with the multigrid and BiCGStab solver. Moreover, we use the canonical grid transfer routines regarding the chosen FEM space which treat both solution components separately in the case of the cGP(2) and dG(1) approaches (see [39] for the details). Finally, the coarse grid problem is solved by a direct solver.

## 5.5. Numerical results

In this section, we perform several numerical tests in order to compare the accuracy of our time discretization schemes. As the first test example we consider problem (5.1) with the domain $\Omega := (0,1)^2$ and the right hand side

$$f(x,y,t) = x(1-x)y(1-y)e^t + 2[y(1-y)+x(1-x)]e^t,$$

which has been derived from the prescribed exact solution

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

The initial data is $u_0(x,y) = u(x,y,0)$.

This and all other examples have the property that there is no spatial error which can be seen as follows. For each fixed time $t$, the exact solution $u(t)$ is an element of the finite element space $V_h$. Therefore, the standard semi-discrete solution with respect to space $u_h(t) \in V_h$ is equal to $u(t)$. If we now apply the time discretization to $u_h(t) = u(t)$ we get $u_{h,\tau}(t) = u_\tau(t)$. Since space and time discretizations are of Galerkin type we obtain $u_{\tau,h}(t) = u_{h,\tau}(t) = u_\tau(t)$. Therefore, in all of our examples, the full discretization error $u(t) - u_{\tau,h}(t)$ is equal to the time discretization error $u(t) - u_\tau(t)$, i.e. we will concentrate only on the time discretization error and exclude interactions with the spatial error.

We apply the time discretization schemes cGP(1), cGP(2) and dG(1) with an equidistant time step size $\tau = T/N$. To measure the error, the following discrete time $L^\infty$-norm of a function $v : I \to L^2(\Omega)$ is used

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

The behavior of the standard $L^2$-norm $\|\cdot\|_2 := \|\cdot\|_{L^2(I,L^2(\Omega))}$ and the discrete $L^\infty$-norm of the time discretization error $u(t) - u_\tau(t)$ over the time interval $I = [0,1]$ can be seen in Table 5.1 and 5.2, respectively. The estimated value of the experimental order of convergence (EOC) is also calculated and compared with the theoretical order of convergence.

| | cGP(1) | | cGP(2) | | dG(1) | |
|---|---|---|---|---|---|---|
| $1/\tau$ | $\|u-u_\tau\|_2$ | EOC | $\|u-u_\tau\|_2$ | EOC | $\|u-u_\tau\|_2$ | EOC |
| 10 | 5.65E-05 | | 3.04E-07 | | 3.08E-05 | |
| 20 | 1.41E-05 | 2.00 | 3.64E-08 | 3.06 | 8.28E-06 | 1.90 |
| 40 | 3.53E-06 | 2.00 | 4.50E-09 | 3.02 | 2.16E-06 | 1.94 |
| 80 | 8.83E-07 | 2.00 | 5.60E-10 | 3.00 | 5.53E-07 | 1.97 |
| 160 | 2.21E-07 | 2.00 | 7.00E-11 | 3.00 | 1.40E-07 | 1.98 |
| 320 | 5.52E-08 | 2.00 | 8.75E-12 | 3.00 | 3.52E-08 | 1.99 |
| 640 | 1.38E-08 | 2.00 | 1.09E-12 | 3.00 | 8.82E-09 | 2.00 |
| 1280 | 3.45E-09 | 2.00 | | | 2.21E-09 | 2.00 |
| 2560 | 8.63E-10 | 2.00 | | | 5.53E-10 | 2.00 |
| 5120 | 2.16E-10 | 2.00 | | | 1.38E-10 | 2.00 |
| 10240 | 5.39E-11 | 2.00 | | | 3.46E-11 | 2.00 |
| 20480 | 1.35E-11 | 2.00 | | | 8.64E-12 | 2.00 |
| 40960 | 3.37E-12 | 2.00 | | | 2.16E-12 | 2.00 |
| 81920 | 8.37E-13 | 2.01 | | | 5.42E-13 | 2.00 |

**Table 5.1**: Error norms $\|u-u_\tau\|_2$ for the first test case.

| | cGP(1) | | cGP(2) | | dG(1) | |
|---|---|---|---|---|---|---|
| $1/\tau$ | $\|u-u_\tau\|_\infty$ | EOC | $\|u-u_\tau\|_\infty$ | EOC | $\|u-u_\tau\|_\infty$ | EOC |
| 10 | 3.63E-06 | | 4.14E-07 | | 1.80E-05 | |
| 20 | 9.09E-07 | 2.00 | 2.65E-08 | 3.97 | 2.59E-06 | 2.80 |
| 40 | 2.27E-07 | 2.00 | 1.67E-09 | 3.99 | 3.51E-07 | 2.88 |
| 80 | 5.68E-08 | 2.00 | 1.05E-10 | 3.99 | 4.59E-08 | 2.93 |
| 160 | 1.42E-08 | 2.00 | 6.57E-12 | 4.00 | 5.90E-09 | 2.96 |
| 320 | 3.55E-09 | 2.00 | 4.01E-13 | 4.03 | 7.49E-10 | 2.98 |
| 640 | 8.88E-10 | 2.00 | | | 9.45E-11 | 2.99 |
| 1280 | 2.22E-10 | 2.00 | | | 1.19E-11 | 2.99 |
| 2560 | 5.55E-11 | 2.00 | | | 1.33E-12 | 3.16 |
| 5120 | 1.39E-11 | 2.00 | | | | |
| 10240 | 3.46E-12 | 2.00 | | | | |
| 20480 | 8.62E-13 | 2.00 | | | | |

**Table 5.2**: Error norms $\|u-u_\tau\|_\infty$ for the first test case with known analytical solution.

We see that the cGP(2)-method is of order 3 in the $L^2$-norm and superconvergent of order 4 at the discrete points $t_n$, while the dG(1)-method is of order 2 in the $L^2$-norm and superconvergent of order 3 at the end points of the time intervals as expected from the theory. The cGP(1)-method is of order 2 everywhere which is the same behavior as that of the well-known Crank-Nicolson scheme.

Next, we perform numerical tests to analyze the corresponding behavior of the multigrid solver for the different time discretization schemes. As explained before, the solver uses the Symmetric Successive Overrelaxation (SSOR) method in the smoothing process and applies one pre- and post-smoothing step. We present the average number of multigrid and preconditioned BiCGStab iterations per time step for solving the corresponding systems in Table 5.3 and 5.4. For a better comparison, the BiCGStab solver also utilizes SSOR as a preconditioner. 'Lev' denotes the refinement level of the space mesh. From Table 5.3, we see that the multigrid solver requires almost the

| Lev | $\tau = 1/20$ | $\tau = 1/80$ | $\tau = 1/320$ | $\tau = 1/1280$ |
|---|---|---|---|---|
| 6 | 13-13-13 | 13-13-13 | 12-12-12 | 10-10-10 |
| 7 | 13-13-13 | 13-13-13 | 13-13-13 | 12-12-12 |
| 8 | 14-13-14 | 13-13-13 | 13-13-13 | 13-13-13 |

**Table 5.3**: Averaged multigrid iterations per time step for cGP(1) - cGP(2) - dG(1).

| Lev | $\tau = 1/20$ | $\tau = 1/80$ | $\tau = 1/320$ | $\tau = 1/1280$ |
|---|---|---|---|---|
| 3 | 7-10-10 | 5-13-12 | 4-17-14 | 5-15-16 |
| 4 | 14-18-19 | 10-14-16 | 5-15-15 | 4-15-16 |
| 5 | 29-35-36 | 22-25-29 | 12-16-19 | 5-15-16 |
| 6 | 61-70-76 | 43-48-53 | 23-24-32 | 12-16-19 |
| 7 | 108-138-158 | 81-89-101 | 46-50-60 | 26-27-33 |
| 8 | 214-284-327 | 160-168-201 | 88-100-119 | 52-50-63 |

**Table 5.4**: Averaged BiCGStab iterations per time step for cGP(1) - cGP(2) - dG(1).

same number of iterations for the different presented time discretization schemes. Moreover, the number of multigrid iterations remains fairly constant if we increase the refinement level of the

space mesh. There is also no noticeable increase in the number of iterations if we decrease the time step by a factor of 2. This means that the behavior of the multigrid solver is almost independent of the space mesh size and the time step. On the other hand, Table 5.4 shows that the averaged number of iterations per time step increases almost by a factor of 2 if we increase the space mesh level in the BiCGStab solver.

Next, in order to measure and compare the efficiency of the two iterative solvers for the different time schemes, we present in Tables 5.5-5.7 the averaged CPU-time required for one solver iteration on a given space mesh level. In Table 5.5, the CPU-times in seconds are related to space

| $1/\tau$ | Multigrid Solver | | | BiCGStab Solver | | |
|---|---|---|---|---|---|---|
| | **cGP(1)** Sec | **cGP(2)** Sec | **dG(1)** Sec | **cGP(1)** Sec | **cGP(2)** Sec | **dG(1)** Sec |
| 20 | 0.014 | 0.024 | 0.030 | 0.003 | 0.008 | 0.007 |
| 80 | 0.014 | 0.032 | 0.029 | 0.003 | 0.008 | 0.012 |
| 320 | 0.012 | 0.032 | 0.030 | 0.005 | 0.011 | 0.013 |
| 1280 | 0.014 | 0.026 | 0.036 | 0.007 | 0.012 | 0.018 |

**Table 5.5**: CPU-time per solver iteration for space mesh level=6.

| $1/\tau$ | Multigrid Solver | | | BiCGStab Solver | | |
|---|---|---|---|---|---|---|
| | **cGP(1)** Sec | **cGP(2)** Sec | **dG(1)** Sec | **cGP(1)** Sec | **cGP(2)** Sec | **dG(1)** Sec |
| 20 | 0.046 | 0.101 | 0.100 | 0.013 | 0.039 | 0.033 |
| 80 | 0.048 | 0.105 | 0.105 | 0.013 | 0.034 | 0.038 |
| 320 | 0.044 | 0.132 | 0.133 | 0.018 | 0.044 | 0.038 |
| 1280 | 0.048 | 0.145 | 0.105 | 0.022 | 0.053 | 0.048 |

**Table 5.6**: CPU-time per solver iteration for space mesh level=7.

| $1/\tau$ | Multigrid Solver | | | BiCGStab Solver | | |
|---|---|---|---|---|---|---|
| | **cGP(1)** Sec | **cGP(2)** Sec | **dG(1)** Sec | **cGP(1)** Sec | **cGP(2)** Sec | **dG(1)** Sec |
| 20 | 0.192 | 0.475 | 0.575 | 0.064 | 0.199 | 0.186 |
| 80 | 0.191 | 0.568 | 0.487 | 0.065 | 0.177 | 0.189 |
| 320 | 0.224 | 0.538 | 0.532 | 0.073 | 0.175 | 0.204 |
| 1280 | 0.214 | 0.701 | 0.702 | 0.086 | 0.191 | 0.175 |

**Table 5.7**: CPU-time per solver iteration for space mesh level=8.

mesh level 6. We observe that the CPU-time in case of cGP(2) or dG(1) is almost 2-3 times the CPU-time of cGP(1) for both, the multigrid and BiCGStab solver. We also note that the CPU-time grows approximately by a factor of 4-5 if we increase the space mesh level to level 7 and 8 in Table 5.6 and 5.7. These factors are nearly optimal since the number of space unknowns is increased by a factor of 4 if the level is increased by one.

Next we compare the time discretization schemes with respect to accuracy and numerical costs. Here, the multigrid solver uses four SSOR iterations in the pre- and post-smoothing step. Table 5.8 shows, for different sizes of the time step $\tau$ and different time discretization schemes, the global $L^2$-norm error and the total CPU-time required for the computations in all time intervals. The space discretization was done on mesh level 6. One can see that, in order to achieve the

| | cGP(1) | | cGP(2) | | dG(1) | |
|---|---|---|---|---|---|---|
| $1/\tau$ | $\|u-u_\tau\|_2$ | CPU | $\|u-u_\tau\|_2$ | CPU | $\|u-u_\tau\|_2$ | CPU |
| 10 | 5.65E-05 | 2 | 3.04E-07 | 4 | 3.08E-05 | 4 |
| 20 | 1.41E-05 | 3 | 3.64E-08 | 8 | 8.28E-06 | 8 |
| 40 | 3.53E-06 | 6 | 4.50E-09 | 16 | 2.16E-06 | 17 |
| 80 | 8.83E-07 | 11 | 5.60E-10 | 28 | 5.53E-07 | 29 |
| 160 | 2.21E-07 | 23 | 7.00E-11 | 61 | 1.40E-07 | 59 |
| 320 | 5.52E-08 | 47 | 8.75E-12 | 115 | 3.52E-08 | 117 |
| 640 | 1.38E-08 | 87 | 1.09E-12 | 199 | 8.82E-09 | 203 |
| 1280 | 3.45E-09 | 171 | | | 2.21E-09 | 399 |
| 2560 | 8.63E-10 | 298 | | | 5.53E-10 | 839 |
| 5120 | 2.16E-10 | 517 | | | 1.38E-10 | 1483 |
| 10240 | 5.39E-11 | 825 | | | 3.46E-11 | 3160 |
| 20480 | 1.35E-11 | 1677 | | | 8.64E-12 | 6662 |
| 40960 | 3.37E-12 | 4131 | | | 2.16E-12 | 13519 |
| 81920 | 8.37E-13 | 10568 | | | 5.42E-13 | 27513 |

**Table 5.8**: Error norms $\|u-u_\tau\|_2$ and total CPU-times to achieve the accuracy of $10^{-12}$.

accuracy of $10^{-12}$, we need the very small time step $\tau = 1/81920$ for the cGP(1) and dG(1) scheme while this accuracy can be already achieved with $\tau = 1/640$ in the cGP(2) scheme. To compare the numerical costs per time step let us note that the number of multigrid iterations to solve one linear block system is approximately the same (about 5) for the three time discretization schemes. However, the cost of one multigrid iteration in the cGP(2) or dG(1) method is about 2-3 times higher than in cGP(1). Nevertheless, for a desired accuracy of $10^{-12}$, the cGP(2) scheme is about 50 times faster than cGP(1) due to the much larger time step size required for cGP(2).

In Table 5.9, we show the analogous comparison between the three time discretizations with respect to the numerical costs and the accuracy measured in the discrete $L^\infty$-norm. Due to its superconvergence of order 3 in the discrete time points, the dG(1)-method is now faster than cGP(1) which is only of order 2. The most efficient scheme is again cGP(2).

| | cGP(1) | | cGP(2) | | dG(1) | |
|---|---|---|---|---|---|---|
| $1/\tau$ | $\|u-u_\tau\|_2$ | CPU | $\|u-u_\tau\|_2$ | CPU | $\|u-u_\tau\|_2$ | CPU |
| 10 | 3.63E-06 | 2 | 4.14E-07 | 4 | 1.80E-05 | 4 |
| 20 | 9.09E-07 | 3 | 2.65E-08 | 8 | 2.59E-06 | 8 |
| 40 | 2.27E-07 | 6 | 1.67E-09 | 16 | 3.51E-07 | 17 |
| 80 | 5.68E-08 | 11 | 1.05E-10 | 28 | 4.59E-08 | 29 |
| 160 | 1.42E-08 | 23 | 6.57E-12 | 61 | 5.90E-09 | 59 |
| 320 | 3.55E-09 | 47 | 4.01E-13 | 115 | 7.49E-10 | 117 |
| 640 | 8.88E-10 | 87 | | | 9.45E-11 | 203 |
| 1280 | 2.22E-10 | 171 | | | 1.19E-11 | 399 |
| 2560 | 5.55E-11 | 298 | | | 1.33E-12 | 839 |
| 5120 | 1.39E-11 | 517 | | | | |
| 10240 | 3.46E-12 | 825 | | | | |
| 20480 | 8.62E-13 | 1677 | | | | |

**Table 5.9**: Error norms $\|u-u_\tau\|_\infty$ and total CPU-time to achieve the accuracy of $10^{-12}$.

To show that the proposed time discretization schemes can also efficiently handle the case when the solution approaches a steady state, we provide numerical tests with very large time

steps. We consider problem (5.1) with $\Omega = (0,1)^2$ and the prescribed (time-independent, steady state) solution

$$u(x,y,t) := x(1-x)y(1-y).$$

For this function $u$, we compute the corresponding right hand side $f$. As initial data we take $u_0 = 0$. Table 5.10 and 5.11 indicate for the multigrid and the BiCGStab method, respectively, the number of solver iterations required for one time step. From Table 5.10, one can see that there is

| Lev | $\tau = 10^{-6}$ | $\tau = 10^{-3}$ | $\tau = 1$ | $\tau = 10^{3}$ | $\tau = 10^{6}$ |
|---|---|---|---|---|---|
| 6 | 3-5-6 | 4-4-5 | 5-6-6 | 5-6-6 | 5-6-6 |
| 7 | 3-5-6 | 5-5-6 | 6-6-6 | 5-6-6 | 5-6-6 |
| 8 | 2-4-5 | 5-6-6 | 5-6-6 | 6-6-6 | 6-6-6 |

**Table 5.10**: Averaged multigrid iterations per time step for cGP(1) - cGP(2) - dG(1).

| Lev | $\tau = 10^{-6}$ | $\tau = 10^{-3}$ | $\tau = 1$ | $\tau = 10^{3}$ | $\tau = 10^{6}$ |
|---|---|---|---|---|---|
| 6 | 8-17-19 | 14-16-18 | 78-78-82 | 69-83-82 | 72-86-86 |
| 7 | 7-17-17 | 31-26-35 | 145-173-172 | 131-147-164 | 149-170-157 |
| 8 | 6-15-16 | 60-51-67 | 280-362-297 | 280-290-290 | 288-322-293 |

**Table 5.11**: Averaged BiCGStab iterations per time step for cGP(1) - cGP(2) - dG(1).

no big difference in the number of solver iterations for time step size $\tau = 10^{-6}$ up to $\tau = 10^{6}$. This means that the behavior of the multigrid convergence is pretty robust with respect to very small and very large time steps. However, from Table 5.11, we observe that, in the case of time step sizes $\tau \geq 10^{-3}$, the BiCGStab solver is suitable only for lower space mesh levels since the number of iterations nearly doubles if we increase the space mesh level by one. Furthermore, the number of BiCGStab iterations grows on each space mesh level if we increase the size of the time step in the range between $\tau = 10^{-6}$ and $\tau = 1$.

At the end, we want to show the behavior of the presented time discretizations for another prototypical problem which is more oscillating in time. For this, in problem (5.1) with $\Omega = (0,1)^2$, we prescribe the exact solution

$$u(x,y,t) := x(1-x)y(1-y)\sin(10\pi t)$$

and compute the corresponding right hand side $f$ and the initial data $u_0(x,y) = u(x,y,0)$. Table 5.12 shows the behavior of the discrete $L^\infty$-norm of the error and the EOC over the time interval $[0,1]$. One can see that the presented time discretizations confirm their theoretical order of convergence also for such an oscillatory solution. In order to compare the three discretization schemes, one can state that the cGP(2) gains the same accuracy at an eight times larger time step size than dG(1) while for cGP(1), we actually need to bisect the time step size ten times more. Consequently, the cGP(2)-method is almost thousand times and dG(1) almost hundred times faster than cGP(1) for such an oscillatory exact solution. This effect has been much more visible when we considered the more complex solution with a higher number of oscillations in the sine function, i.e., if we have prescribed the solution

$$u(x,y,t) := x(1-x)y(1-y)\sin(100\pi t)$$

in our model problem. Here, the discrepancy has been even more obvious between the high order schemes dG(1) and cGP(2) on the one side and the second order method cGP(1), which is close to the standard Crank-Nicolson scheme and representative for many other 2nd order schemes, on the other side. These tests demonstrate the superior quality of the high order approaches, together with corresponding fast solvers, for complex dynamical problems.

| | cGP(1) | | cGP(2) | | dG(1) | |
|---|---|---|---|---|---|---|
| $1/\tau$ | $\|u-u_\tau\|_\infty$ | EOC | $\|u-u_\tau\|_\infty$ | EOC | $\|u-u_\tau\|_\infty$ | EOC |
| 20 | 5.85E-03 | | 2.03E-04 | | 4.19E-04 | |
| 40 | 1.50E-03 | 1.96 | 1.31E-05 | 3.95 | 7.75E-05 | 2.44 |
| 80 | 3.72E-04 | 2.01 | 8.34E-07 | 3.97 | 1.06E-05 | 2.87 |
| 160 | 9.43E-05 | 1.98 | 5.29E-08 | 3.98 | 1.40E-06 | 2.92 |
| 320 | 2.35E-05 | 2.00 | 3.32E-09 | 3.99 | 1.80E-07 | 2.96 |
| 640 | 5.88E-06 | 2.00 | 2.08E-10 | 4.00 | 2.29E-08 | 2.97 |
| 1280 | 1.47E-06 | 2.00 | 1.30E-11 | 4.00 | 2.90E-09 | 2.98 |
| 2560 | 3.68E-07 | 2.00 | 8.13E-13 | 4.00 | 3.65E-10 | 2.99 |
| 5120 | 9.19E-08 | 2.00 | | | 4.57E-11 | 2.99 |
| 10240 | 2.30E-08 | 2.00 | | | 5.73E-12 | 3.00 |
| 20480 | 5.75E-09 | 2.00 | | | 7.14E-13 | 3.00 |
| 40960 | 1.44E-09 | 2.00 | | | | |
| 81920 | 3.59E-10 | 2.00 | | | | |
| 163840 | 8.97E-11 | 2.00 | | | | |
| 327680 | 2.24E-11 | 2.00 | | | | |
| 655360 | 5.61E-12 | 2.00 | | | | |
| 1310720 | 1.40E-12 | 2.00 | | | | |
| 2621440 | 3.50E-13 | 2.00 | | | | |

**Table 5.12**: Error norms $\|u-u_\tau\|_\infty$ for the "sin-test" case.

## 5.6. Summary

In this chapter, we have presented continuous Galerkin-Petrov and discontinuous Galerkin time discretization schemes for the two dimensional heat equation. In particular, we have analyzed the cGP(1), cGP(2) and dG(1)-method.

### Accuracy

From our numerical results, we have observed that the estimated experimental orders of convergence confirm the theoretical orders. Furthermore, the tests show that the cGP(2)-scheme provides significantly more accurate numerical solutions than the other presented schemes cGP(1) and dG(1) which means that quite large time step sizes can be used without losing accuracy.

### Efficiency

We have discussed implementation aspects of the presented time discretizations as well as efficient methods for solving the resulting block systems. Here, we have compared a preconditioned BiCGStab solver as a Krylov space method with an adapted geometrical multigrid solver. It has been analyzed that the multigrid solver requires almost the same number of iterations for all the presented time discretization schemes. Moreover, the number of multigrid iterations remains fairly constant by increasing the refinement level of the space mesh. There is also no noticeable increase in the number of iterations if we decrease the time step size. This means that the behavior of the multigrid solver is almost independent of the space mesh and the time step size. On the other hand, the averaged number of iterations per time step increases almost by a factor of 2 if we increase the space mesh level in the BiCGStab solver. Thus, the multigrid solver is much more efficient than the preconditioned BiCGStab solver since it shows a robust convergence behavior which is nearly independent of the space mesh and the time step size.

# 6

# Galerkin time discretizations for the heat equation using Gauß-Lobatto points

In this chapter, we shortly analyze a variant of the continuous Galerkin-Petrov (cGP) method based on the Gauß-Lobatto quadrature formula [49] for the heat equation. This implementation of cGP($k$)-method only differs from the previous version of cGP($k$)-method in Chapter 5 in the choice of Gauß quadrature formula. The Gauß-Lobatto based cGP($k$)-method gives the same accuracy as the standard cGP($k$)-method but avoiding any extrapolation for getting the solution at the time discrete points. By means of numerical experiments we analyze and compare the accuracy of the Gauß-Lobatto based cGP($k$)-method.

## 6.1. The cGP-method for the heat equation

For the time discretization of the heat equation (5.1), we decompose the time interval $I = [0,T]$ into subintervals $I_n := [t_{n-1}, t_n]$, $n = 1, \ldots, N$. Applying the *exact* cGP($k$)-method to heat equation (5.1) in Chapter 5 we obtain a time marching process with the following "$I_n$-problem":

*Find* $u_\tau|_{I_n} \in \mathbb{P}_k(I_n, V)$ *such that*

$$\int_{I_n} \Big\{ (d_t u_\tau(t), v)_\Omega + a(u_\tau(t), v) \Big\} \psi_{n,i}(t)\, dt = \int_{I_n} (f(t), v)_\Omega\, \psi_{n,i}(t)\, dt \qquad \forall\, v \in V \tag{6.1}$$

for $i = 1, \ldots, k$, with the "initial condition" $u_\tau|_{I_n}(t_{n-1}) = u_\tau|_{I_{n-1}}(t_{n-1})$ for $n \geq 2$ or $u_\tau|_{I_1}(t_0) = u_0$ which ensures the continuity of the time discrete solution $u_\tau : I \to V$. The functions $\psi_{n,i}$ denote real-valued basis functions of the polynomial space $\mathbb{P}_{k-1}(I_n)$ and the notation $u_\tau|_{I_n} \in \mathbb{P}_k(I_n, V)$ means that there exist $V$-valued coefficients $U_n^j \in V$ such that $u_\tau|_{I_n}$ has the representation

$$u_\tau|_{I_n}(t) := \sum_{j=0}^{k} U_n^j \phi_{n,j}(t) \qquad \forall\, t \in I_n, \tag{6.2}$$

where the real functions $\phi_{n,j} \in \mathbb{P}_k(I_n)$ are the Lagrange basis functions with respect to $k+1$ suitable nodal points $t_{n,j} \in I_n$ satisfying the usual conditions (with $\delta_{i,j}$ denoting the Kronecker symbol)

$$\phi_{n,j}(t_{n,i}) = \delta_{i,j}, \quad i,j = 0, \ldots, k, \qquad \text{such that} \qquad U_n^j = u_\tau|_{I_n}(t_{n,j}) \quad \forall\, j. \tag{6.3}$$

The new implementation of cGP($k$)-method here only differs from the standard version of cGP($k$)-method in Chapter 5 in the choice of Gauß quadrature formula. Here, we choose the nodal points $t_{n,j}$ as the quadrature points of the $(k+1)$-point Gauß-Lobatto formula on $I_n$ which is the same choice used in [49] while in Chapter 5 these nodal points $t_{n,j}$ have been chosen as $t_{n,0} = t_{n-1}$ and the other points $t_{n,1}, \ldots, t_{n,k}$ are chosen as the quadrature points of the $k$-point Gauß formula

on $I_n$. For the Gauß-Lobatto points, it holds $t_{n,0} = t_{n-1}$ which implies that the initial condition is equivalent to the condition

$$U_n^0 = u_\tau|_{I_{n-1}}(t_{n-1}) \quad \text{if} \quad n \geq 2 \qquad \text{or} \qquad U_n^0 = u_0 \quad \text{if} \quad n = 1. \tag{6.4}$$

This formula is exact if the function to be integrated is a polynomial of degree less or equal $2k-1$. From the representation (6.2) we get

$$\int_{I_n} (d_t u_\tau(t), v)_\Omega \psi_{n,i}(t) dt = \sum_{j=0}^{k} \left(U_n^j, v\right)_\Omega \int_{I_n} \phi'_{n,j}(t)\psi_{n,i}(t)\, dt \qquad \forall\, v \in V. \tag{6.5}$$

In order to derive a numerical scheme from the exact cGP($k$)-method (6.1) we transform all $I_n$-integrals into integrals over the reference interval $\hat{I} := [-1,1]$ by means of the affine transformation as in Chapter 5. That is, we define the basis functions $\phi_{n,j} \in \mathbb{P}_k(I_n)$ of (6.2) via the affine reference transformation $T_n : \hat{I} \to I_n$ where $\hat{I} := [-1,1]$ and

$$t = T_n(\hat{t}) := \frac{t_{n-1}+t_n}{2} + \frac{\tau_n}{2}\hat{t} \in I_n \qquad \forall\, \hat{t} \in \hat{I},\ n = 1,\ldots,N. \tag{6.6}$$

Let $\hat{\phi}_j \in \mathbb{P}_k(\hat{I})$, $j = 0,\ldots,k$, denote the basis functions satisfying the conditions

$$\hat{\phi}_j(\hat{t}_i) = \delta_{i,j}, \qquad i,j = 0,\ldots,k, \tag{6.7}$$

where $\hat{t}_i$, $i = 0,\ldots,k$, are the Gauß-Lobatto quadrature points for the reference interval $\hat{I}$. Then, we define the basis functions on the original time interval $I_n$ by

$$\phi_{n,j}(t) := \hat{\phi}_j(\hat{t}) \qquad \text{with} \qquad \hat{t} := T_n^{-1}(t) = \frac{2}{\tau_n}\left(t - \frac{t_n - t_{n-1}}{2}\right) \in \hat{I}. \tag{6.8}$$

Similarly, we define the test basis functions $\psi_{n,i}$ by suitable reference basis functions $\hat{\psi}_i \in \mathbb{P}_{k-1}(\hat{I})$, i.e.,

$$\psi_{n,i}(t) := \hat{\psi}_i(T_n^{-1}(t)) \qquad \forall\, t \in I_n,\ i = 1,\ldots,k. \tag{6.9}$$

Again, we have to approximate the right hand side in the exact cGP($k$)-method (6.1) by some numerical integration. This leads us to the *numerically integrated cGP(k)-method*. To this end, we replace the function $f(t)$ by the time-polynomial $\pi_k f \in \mathbb{P}_k(I_n, L^2(\Omega))$ defined as the Lagrange interpolate

$$\pi_k f(t) := \sum_{j=0}^{k} f(t_{n,j})\phi_{n,j}(t) \qquad \forall\, t \in I_n.$$

Now, we transform all integrals in (6.1) to the reference interval $\hat{I}$ and obtain the following system of equations for the "coefficients" $U_n^j \in V$ in the ansatz (6.2)

$$\sum_{j=0}^{k} \left\{ \alpha_{i,j} \left(U_n^j, v\right)_\Omega + \frac{\tau_n}{2}\beta_{i,j} a(U_n^j, v) \right\} = \frac{\tau_n}{2} \sum_{j=0}^{k} \beta_{i,j} \left(f(t_{n,j}), v\right)_\Omega \qquad \forall\, v \in V \tag{6.10}$$

where $i = 1,\ldots,k$,

$$\alpha_{i,j} := \int_{\hat{I}} \hat{\phi}'_j(\hat{t})\hat{\psi}_i(\hat{t})\, d\hat{t}, \qquad \beta_{i,j} := \int_{\hat{I}} \hat{\phi}_j(\hat{t})\hat{\psi}_i(\hat{t})\, d\hat{t} \tag{6.11}$$

and the "coefficient" $U_n^0 \in V$ is known. Due to the polynomial degree of $\hat{\phi}_j$ and $\hat{\psi}_i$ we can compute the integrals for $\alpha_{i,j}$ and $\beta_{i,j}$ exactly by means of the $(k+1)$-point Gauß-Lobatto formula with weights $\hat{w}_\mu$ and points $\hat{t}_\mu$, $\mu = 0,\ldots,k$. We choose the test functions $\hat{\psi}_i \in \mathbb{P}_{k-1}(\hat{I})$ in (6.11) such that

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

Then, the system (6.10) is equivalent to the following coupled system of equations for the $k$ unknown "coefficients" $U_n^j \in V$, $j = 1,\dots,k$,

$$\sum_{j=0}^{k} \left\{ \alpha_{i,j} \left(U_n^j, v\right)_\Omega + \frac{\tau_n}{2} \beta_{i,j} a(U_n^i, v) \right\} = \frac{\tau_n}{2} \sum_{j=0}^{k} \beta_{i,j} \left(f(t_{n,i}), v\right)_\Omega \qquad \forall\, v \in V, \quad i = 1,\dots,k, \tag{6.12}$$

where $U_n^0 = u_\tau(t_{n-1})$ for $n > 1$ and $U_1^0 = u_0$. If we apply for the quadrature formula as the $(k+1)$-point Gauß-Lobatto formula in the cGP($k$)-method we will call the corresponding method shortly the cGP($k$)-GL($k+1$)-method.

In the following, we specify the cGP($k$)-GL($k+1$)-method, for the cases $k = 1$ and $k = 2$.

### 6.1.1. cGP(1)-GL(2)-method

We use the 2-point Gauß-Lobatto formula (trapezoidal rule) with $\hat{w}_0 = \hat{w}_1 = 1$ and $\hat{t}_0 = -1$, $\hat{t}_1 = 1$. Then, we obtain

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

Using the notation $U_n^0 := u_\tau(t_{n-1})$ we obtain the following equation for the "unknown" $U^n \in V$ :

$$\left(U_n^1, v\right)_\Omega - \frac{\tau_n}{2} a(U_n^1, v) = \frac{\tau_n}{2} \left\{ \left(f(t_{n,1}), v\right)_\Omega + \left(f(t_{n,0}), v\right)_\Omega \right\} + \left(U_n^0, v\right)_\Omega - \frac{\tau_n}{2} a(U_n^0, v) \tag{6.13}$$

for all $v \in V$ which is the well-known *Crank-Nicolson method*. Here, $U_n^0$ is the initial value at the time interval $[t_{n-1}, t_n]$ coming from the previous time interval or the initial value $u_0$. This scheme is the well-known *Crank-Nicolson method*.

### 6.1.2. cGP(2)-GL(3)-method

We use the 3-point Gauß-Lobatto formula with $\hat{w}_0 = \hat{w}_2 = 1/3$, $\hat{w}_1 = 4/3$ and $\hat{t}_0 = -1$, $\hat{t}_1 = 0$, $\hat{t}_2 = 1$, which is the Simpsons's rule. Then, we get

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

On the time interval $I_n = [t_{n-1}, t_n]$ we have to solve for the two "unknowns" $U_n^j = u_\tau(t_{n,j})$ with $t_{n,j} := T_n(\hat{t}_j)$ for $j = 1, 2$. The coupled system for $U_n^1, U_n^2 \in V$ reads

$$\begin{array}{rcl} \left\{ \alpha_{1,1} \left(U_n^1, v\right)_\Omega + \frac{\tau_n}{2} \beta_{1,1} a(U_n^1, v) \right\} + \alpha_{1,2} \left(U_n^2, v\right)_\Omega & = & \ell_1(v), \\ \alpha_{2,1} \left(U_n^1, v\right)_\Omega + \left\{ \alpha_{2,2} \left(U_n^2, v\right)_\Omega + \frac{\tau_n}{2} \beta_{2,2} a(U_n^2, v) \right\} & = & \ell_2(v), \end{array} \tag{6.14}$$

which has to be satisfied for all $v \in V$ with

$$\ell_i(v) = \frac{\tau_n}{2} \left\{ \beta_{i,2} \left(f(t_{n,i}), v\right)_\Omega + \beta_{i,1} \left(f(t_{n,i}), v\right)_\Omega + \beta_{1,0} \left(f(t_{n,0}), v\right)_\Omega \right\} - \alpha_{i,0} \left(U_n^0, v\right)_\Omega - \frac{\tau_n}{2} \beta_{i,0} a(U_n^0, v)$$

for $i = 1, 2$. Here, $U_n^0$ is the initial value at the time interval $I_n$ coming from the previous time interval or the initial value $u_0$.

## 6.2. Space Discretization by FEM

After discretizing equation (5.1) in time, we now apply the finite element method to discretize each of the "$I_n$-problem" in space in a similar way as in Chapter 5. In the following, we will present the fully discrete block systems for the cGP(1)-GL(2) and cGP(2)-GL(3) method which are used in our numerical experiments.

#### 6.2.1. cGP(1)-GL(2)-method

The problem on time interval $I_n$ reads: *For given* $\underline{U}_n^0 \in \mathbb{R}^{m_h}$, *find* $\underline{U}_n^1 \in \mathbb{R}^{m_h}$ *such that*

$$\left(M + \frac{\tau_n}{2}L\right)\underline{U}_n^1 = \frac{\tau_n}{2}(F_n^1 + F_n^1) + (M - \frac{\tau_n}{2}L)\underline{U}_n^0. \tag{6.15}$$

#### 6.2.2. cGP(2)-GL(3)-method

The $2 \times 2$ block system on time interval $I_n$ reads: *For given* $\underline{U}_n^0 \in \mathbb{R}^{m_h}$, *find* $\underline{U}_n^1, \underline{U}_n^2 \in \mathbb{R}^{m_h}$ *such that*

$$\begin{pmatrix} (2M + \tau_n L) & \frac{1}{2}M \\ -8M & (4M + \tau_n L) \end{pmatrix} \begin{pmatrix} \underline{U}_n^1 \\ \underline{U}_n^2 \end{pmatrix} = \begin{pmatrix} R_n^1 \\ R_n^2 \end{pmatrix} \tag{6.16}$$

where

$$\begin{aligned} R_n^1 &= \tau_n(F_n^1 + \tfrac{1}{2}F_n^0) + (\tfrac{5}{2}M - \tfrac{\tau_n}{2}L)\underline{U}_n^0 \\ R_n^2 &= \tau_n(F_n^2 - F_n^0) + (\tau_n L - 4M)\underline{U}_n^0. \end{aligned}$$

### 6.3. Numerical results

In this section, we perform some numerical tests in order to analyze and compare the accuracy of the cGP(1)-GL(2) and cGP(2)-GL(3)-method. As a test example we consider problem (5.1) with the domain $\Omega := (0,1)^2$ and the right hand side

$$f(x,y,t) = x(1-x)y(1-y)e^t + 2[y(1-y) + x(1-x)]e^t,$$

which has been derived from the prescribed exact solution

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

The initial data is $u_0(x,y) = u(x,y,0)$.

We apply all the time discretization schemes cGP(1), cGP(2), cGP(1)-GL(2) and cGP(2)-GL(3) with an equidistant time step size $\tau = T/N$. Here, the cGP(1)-GL(2) and cGP(2)-GL(3) denote the Gauß-Lobatto based cGP(1) and cGP(2)-method. To measure the error, the following discrete time $L^\infty$-norm of a function $v : I \to L^2(\Omega)$ is used, similar to Chapter 5,

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

The behavior of the standard $L^2$-norm $\|\cdot\|_2 := \|\cdot\|_{L^2(I,L^2(\Omega))}$ and the discrete $L^\infty$-norm of the time discretization error $u(t) - u_\tau(t)$ over the time interval $I = [0,1]$ can be seen in Table 6.1–6.4, respectively. The estimated value of the experimental order of convergence (EOC) is also calculated and compared with the theoretical order of convergence.

| | cGP(1) | | cGP(2) | |
|---|---|---|---|---|
| $1/\tau$ | $\|u-u_\tau\|_2$ | EOC | $\|u-u_\tau\|_2$ | |
| 10 | 5.65E-05 | | 3.04E-07 | |
| 20 | 1.41E-05 | 2.00 | 3.64E-08 | 3.06 |
| 40 | 3.53E-06 | 2.00 | 4.50E-09 | 3.02 |
| 80 | 8.83E-07 | 2.00 | 5.60E-10 | 3.00 |
| 160 | 2.21E-07 | 2.00 | 7.00E-11 | 3.00 |
| 320 | 5.52E-08 | 2.00 | 8.75E-12 | 3.00 |
| 640 | 1.38E-08 | 2.00 | 1.09E-12 | 3.00 |
| 1280 | 3.45E-09 | 2.00 | | |
| 2560 | 8.63E-10 | 2.00 | | |
| 5120 | 2.16E-10 | 2.00 | | |
| 10240 | 5.39E-11 | 2.00 | | |
| 20480 | 1.35E-11 | 2.00 | | |
| 40960 | 3.37E-12 | 2.00 | | |
| 81920 | 8.37E-13 | 2.01 | | |

**Table 6.1**: Error norms $\|u-u_\tau\|_2$ for the test case using the cGP(1), cGP(2)-method.

| | cGP(1)-GL(2) | | cGP(2)-GL(3) | |
|---|---|---|---|---|
| $1/\tau$ | $\|u-u_\tau\|_2$ | EOC | $\|u-u_\tau\|_2$ | EOC |
| 10 | 5.65E-05 | | 4.05E-07 | |
| 20 | 1.41E-05 | 2.00 | 5.07E-08 | 3.00 |
| 40 | 3.53E-06 | 2.00 | 6.33E-09 | 3.00 |
| 80 | 8.83E-07 | 2.00 | 7.92E-10 | 3.00 |
| 160 | 2.21E-07 | 2.00 | 9.90E-11 | 3.00 |
| 320 | 5.52E-08 | 2.00 | 1.24E-11 | 3.00 |
| 640 | 1.38E-08 | 2.00 | 1.55E-12 | 3.00 |
| 1280 | 3.45E-09 | 2.00 | 1.93E-13 | 3.00 |
| 2560 | 8.63E-10 | 2.00 | | |
| 5120 | 2.16E-10 | 2.00 | | |
| 10240 | 5.39E-11 | 2.00 | | |
| 20480 | 1.35E-11 | 2.00 | | |
| 40960 | 3.37E-12 | 2.00 | | |
| 81920 | 8.37E-13 | 2.01 | | |

**Table 6.2**: Error norms $\|u-u_\tau\|_2$ for the test case using the cGP(1)-GL(2), cGP(2)-GL(3) method.

| | cGP(1) | | cGP(2) | |
|---|---|---|---|---|
| $1/\tau$ | $\|u-u_\tau\|_\infty$ | EOC | $\|u-u_\tau\|_\infty$ | EOC |
| 10 | 3.63E-06 | | 4.14E-07 | |
| 20 | 9.09E-07 | 2.00 | 2.65E-08 | 3.97 |
| 40 | 2.27E-07 | 2.00 | 1.67E-09 | 3.99 |
| 80 | 5.68E-08 | 2.00 | 1.05E-10 | 3.99 |
| 160 | 1.42E-08 | 2.00 | 6.57E-12 | 4.00 |
| 320 | 3.55E-09 | 2.00 | 4.01E-13 | 4.03 |
| 640 | 8.88E-10 | 2.00 | | |
| 1280 | 2.22E-10 | 2.00 | | |
| 2560 | 5.55E-11 | 2.00 | | |
| 5120 | 1.39E-11 | 2.00 | | |
| 10240 | 3.46E-12 | 2.00 | | |
| 20480 | 8.62E-13 | 2.00 | | |

**Table 6.3**: Error norms $\|u-u_\tau\|_\infty$ for the test case using the cGP(1), cGP(2)-method.

| | cGP(1)-GL(2) | | cGP(2)-GL(3) | |
|---|---|---|---|---|
| $1/\tau$ | $\|u-u_\tau\|_\infty$ | EOC | $\|u-u_\tau\|_\infty$ | EOC |
| 10 | 3.63E-06 | | 1.49E-08 | |
| 20 | 9.09E-07 | 2.00 | 9.41E-10 | 3.98 |
| 40 | 2.27E-07 | 2.00 | 5.90E-11 | 4.00 |
| 80 | 5.68E-08 | 2.00 | 3.69E-12 | 4.00 |
| 160 | 1.42E-08 | 2.00 | 2.31E-13 | 4.00 |
| 320 | 3.55E-09 | 2.00 | 9.72E-15 | 4.57 |
| 640 | 8.88E-10 | 2.00 | 8.85E-16 | 3.46 |
| 1280 | 2.22E-10 | 2.00 | | |
| 2560 | 5.55E-11 | 2.00 | | |
| 5120 | 1.39E-11 | 2.00 | | |
| 10240 | 3.46E-12 | 2.00 | | |
| 20480 | 8.62E-13 | 2.01 | | |

**Table 6.4**: Error norms $\|u-u_\tau\|_\infty$ for the test case using the cGP(1)-GL(2), cGP(2)-GL(3) method.

We see from Table 6.1 to 6.4 that the cGP(2)-GL(3) and the standard cGP(2)-method have the accuracy of order 3 in the $L^2$-norm and the super-convergent results of order 4 at the discrete time points $t_n$. The cGP(1)-GL(2) method is of order 2 everywhere in the time interval which is the same behavior as that of the standard cGP(1)-method. Thus, both the Gauß-Lobatto based version and the standard cGP($k$)-method have the same accuracy everywhere in the time interval. An advantage of the Gauß-Lobatto based cGP($k$)-method is that it does not require any extrapolation to get the solution at the discrete time points $t_n$.

## 6.4. Summary

In this chapter, we have investigated a variant of the cGP($k$)-method which is based on the Gauß-Lobatto points. From the numerical experiments, we have analyzed that the Gauß-Lobatto based cGP($k$)-method has the same order of convergence as that of the standard cGP($k$)-method. Moreover, it is advantageous as compared to the cGP($k$)-method for the heat equation. However, for saddle point problems like the Stokes or Navier-Stokes equations, the Gauß-Lobatto based variant

is not applicable; the explicit time discretization of the pressure is numerically instable. To overcome this difficulty, we have the motivation to reconsider the cGP($k$)-methods using the standard Gauß points analyzed in this thesis.

# 7

# Analysis of the continuous Galerkin-Petrov methods

This chapter is concerned with the analysis of newly introduced class of time discretization schemes, namely the cGP($k$)-method. In this analysis, we discuss two important aspects regarding the time discretization, i.e., stability and optimal error estimates. We prove by some energy arguments the A-stability of cGP($k$)-method and an optimal error estimates of order $k+1$ in the standard $L^2$-norm.

## 7.1. Stability of the cGP($k$)-method

In order to prove the A-stability of the cGP($k$)-method, we consider the following model problem which is sufficiently smooth in time and space:
*Find a function* $u : [0,T] \to \mathbb{C}$ *such that*

$$\begin{aligned} d_t u(t) &= \lambda u(t) \quad \forall\, t \in [0,T] \\ u(0) &= u_0, \end{aligned} \tag{7.1}$$

*for a given* $\lambda \in \mathbb{C}$ *and an initial value* $u_0 \in \mathbb{C}$. To this end, it sufficient to prove that for the simple situation of just one time step, i.e., for $t_1 = T = \tau$, it holds

$$|u_\tau(\tau)| < |u_0| \quad \tau > 0, \quad \forall\, \lambda \in \{z \in \mathbb{C} : Re(z) < 0\}, \quad \forall\, u_0 \in \mathbb{C} \setminus \{0\}. \tag{7.2}$$

**Theorem 1** *The exact cGP(k)-method as well as the cGP(k)-method are A-stable in the sense that* (7.2) *holds if the methods are applied to model problem* (7.1).

**Proof.** (From [49]) We prove this theorem by considering the problem (7.1) in an equivalent way as an ODE-system in $\mathbb{R}^2$ such that the application of cGP($k$)-method to problem (7.1) is equivalent to the application of cGP($k$)-method to new model problem. To this end, let $\lambda \in \mathbb{C}$ be given complex number such that $Re(\lambda) < 0$ and define a vector valued function $\vec{u} : [0,T] \to \mathbb{C}$ associated to a given complex function $u : [0,T] \to \mathbb{C}$ such that

$$\vec{u} := \begin{pmatrix} Re(u(t)) \\ Im(u(t)) \end{pmatrix} \qquad \forall\, t \in [0,T].$$

Then, the model problem (7.1) is equivalent to the following problem:
*Find a function* $\vec{u} : [0,T] \to \mathbb{C}$ *such that*

$$\begin{aligned} d_t \vec{u}(t) &= A\vec{u}(t) \quad \forall\, t \in [0,T] \\ \vec{u}(0) &= \vec{u}_0, \end{aligned} \tag{7.3}$$

*where A is* $2\times 2$ *matrix given by*

$$A := \begin{pmatrix} Re(\lambda) & -Im(\lambda) \\ Im(\lambda) & Re(\lambda) \end{pmatrix}$$

*and the initial value* $\vec{u}_0 := (Re(u_0), Im(u_0))^T$. The matrix $A$ is regular and negative definite since $Re(\lambda) < 0$ and

$$\xi^T A\xi = \xi^T A^T \xi = Re(\lambda)\|\xi\|^2 \quad \forall\, \xi \in \mathbb{R}^2, \tag{7.4}$$

where $\|.\|$ denotes the Euclidean norm in $\mathbb{R}^2$. Now, the problem (7.1) and (7.3) are equivalent, so the application of the cGP($k$)-method to problem (7.1) is equivalent to the application of the cGP($k$)-method to problem (7.3) with the Hilbert spaces $V = H = \mathbb{R}^2$ equipped with the Euclidean norm $\|.\|$. The corresponding dual spaces $H' = V'$ are identified with $\mathbb{R}^2$ in the sense that $\langle .,. \rangle_{V',V} = \langle .,. \rangle_{H',H} = (.,.)$ is the usual Euclidean scalar product. The Riesz operator $M : H \to H'$ corresponds to the identity matrix and the associated function $F : [0,T]\times\mathbb{R}^2 \to \mathbb{R}^2$ is defined as $F(t,\vec{u}) := A\vec{u}$. Then, for $t_1 = T = \tau$, the exact cGP($k$)-method reads:
*Find* $\vec{u}_\tau \in X_\tau^k$ *such that* $\vec{u}(0) = \vec{u}_0$ *and*

$$\int_0^T (d_t\vec{u}_\tau(t), \vec{v}_\tau(t))dt = \int_0^T (A\vec{u}_\tau(t), \vec{v}_\tau(t))dt \qquad \forall\, \vec{v} \in Y_\tau^k. \tag{7.5}$$

We apply the $k$-point Gauß formula for the integration which is exact if the function to be integrated is a polynomial of degree less or equal $2k-1$. This means that the cGP($k$)-method with numerical integration is equivalent to the exact cGP($k$)-method such that equation (7.5) is also satisfied. The discrete spaces $X_\tau^k$ and $Y_\tau^k$ are chosen such that for each $\vec{u}_\tau \in X_\tau^k$ the function $\vec{v}_\tau : [0,T] \to \mathbb{R}^2$ defined by $\vec{v}_\tau(t) := (A^{-1})^T d_t\vec{u}_\tau(t)$ is contained in the test space $Y_\tau^k$. Using this test function $\vec{v}_\tau$ in (7.5), leads to

$$\int_0^T (d_t\vec{u}_\tau(t), (A^{-1})^T d_t\vec{u}_\tau(t))dt = \int_0^T (\vec{u}_\tau(t), d_t\vec{u}_\tau(t))dt = \frac{1}{2}\int_0^T \frac{d}{dt}\|\vec{u}_\tau(t)\|^2 dt. \tag{7.6}$$

Since the matrix $A$ is regular and negative definite, we have

$$\|\vec{u}_\tau(t)\|^2 - \|\vec{u}_0\|^2 = 2Re(\lambda)\int_0^T \|A^{-1}d_t\vec{u}_\tau(t)\|^2 dt. \tag{7.7}$$

If the integral on the right hand in (7.7) is zero, then we get, $\|d_t\vec{u}_\tau(t)\| = 0$ for all $t \in [0,\tau]$ since A is regular which means $\vec{u}(0) = \vec{u}_0$ for all $t \in [0,\tau]$. Also from (7.5), if we take $\vec{v}_\tau = \vec{u}_0 \in Y_\tau^k$, the time-constant test function, we obtain $\vec{u}_0^T A\vec{u}_0 = Re(\lambda)\|\vec{u}_0\| = 0$, which is a contradiction to the assumption that $\vec{u}_0 \in \mathbb{C}\setminus\{0\}$, i.e. $\|\vec{u}_0\| \neq 0$. Thus, the right hand side in (7.7) must be negative and we get that $\|\vec{u}_\tau(\tau)\| < \|\vec{u}_0\|$. This completes the proof. □

## 7.2. Optimal error estimate of the cGP($k$)-method

We consider the heat equation: *Find* $u : \Omega\times[0,T] \to \mathbb{R}$ *such that*

$$\begin{array}{rcll} \frac{\partial u}{\partial t} - \triangle u &=& f & in \quad \Omega\times(0,T), \\ u &=& 0 & on \quad \partial\Omega\times[0,T], \\ u(x,0) &=& u_0(x) & for \quad x\in\Omega, \end{array} \tag{7.8}$$

*where* $u(x,t)$ *denotes the temperature in the point* $x \in \Omega$ *at time* $t \in [0,T]$, $f : \Omega\times(0,T) \to \mathbb{R}$ *a given source term and* $u_0 : \Omega \to \mathbb{R}$ *the initial temperature field at time* $t = 0$. Instead of solving

equation (7.8) for the discrete solution $u_\tau \in u_0 + X^k_{\tau,0}$, we consider only the homogeneous part $u^0_\tau := u_\tau - u_0 \in X^k_{\tau,0}$. Then, the application of *exact continuous Galerkin-Petrov method* of *order k* or briefly the "*exact* cGP(*k*)*-method*" to heat equation (7.8) in Chapter 5, for $u^0_\tau$ reads as follows *Find* $u^0_\tau \in X^k_{\tau,0}$ *such that* $u_\tau(0) = u_0$ *and*

$$\int_0^T \Big\{ \big(d_t u^0_\tau(t), v_\tau(t)\big)_\Omega + a(u^0_\tau(t), v_\tau(t)) \Big\} dt = \int_0^T (f(t), v_\tau(t))_\Omega \, dt \qquad \forall\, v_\tau \in Y^k_\tau, \tag{7.9}$$

*where* $(\cdot,\cdot)_\Omega$ *denotes the usual inner product in* $L^2(\Omega)$ *and* $a(\cdot,\cdot)$ *the bilinear form on* $V \times V$ *defined as*

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

In order to prove an optimal error estimate for the discrete solution of the heat equation (7.8), we assume the following approximation property: There exist an interpolation operator $\prod_\tau : X_0 \to X^k_{\tau,0}$ such that for all sufficiently smooth $u \in X_0$ and all time intervals $I_n$, $n = 1, \ldots, N$, it holds

$$\|u - \textstyle\prod_\tau u\|_{L^2(I_n,V)} \le C\tau_n^{k+1} \|d_t^{k+1} u\|_{L^2(I_n,V)}, \tag{7.10}$$

$$\|d_t(u - \textstyle\prod_\tau u)\|_{L^2(I_n,V)} \le C\tau_n^{k} \|d_t^{k+1} u\|_{L^2(I_n,V)}. \tag{7.11}$$

**Theorem 2** *Let* $u^0 : [0,T] \to V$ *be the solution of the heat equation* (7.8) *which is sufficiently smooth with respect to time such that the approximation properties* (7.10) *and* (7.11) *are fulfilled for* $u = u^0$*. Then, the solution* $u^0_\tau \in X^k_{\tau,0}$ *of the corresponding discrete problem of the exact cGP(k)-method satisfies the following error estimate*

$$\|u^0 - u^0_\tau\|_X \le C\tau \Big\{ \sum_{n=1}^N \tau_n^{2k} \|d_t^{k+1} u^0\|^2_{L^2(I_n,V)} \Big\}^{1/2} \quad \le C\tau^k \|d_t^{k+1} u^0\|_{L^2(I,V)}, \tag{7.12}$$

*where the constants C are independent of* $\tau_n$, $\tau$, $T$ *and* $u^0$.

**Proof.** (see [49] for proof). □

**Lemma 3** *Let* $I_n = (t_{n-1}, t_n)$ *be a time interval and* $u \in H^1(I_n, V)$*. Then, it holds the inequality*

$$\|u\|^2_{L^2(I_n,V)} \le 2\tau_n \|u(t_{t-1})\|^2_V + \tau_n^2 \|u\|^2_{L^2(I_n,V)}. \tag{7.13}$$

**Proof.** (From [49]) Since $u \in H^1(I_n, V)$, this implies that $u \in C([t_{n-1}, t_n], V)$ and the values $u(t) \in V$ are well-defined for all $t \in [t_{n-1}, t_n]$. Then, the following holds

$$u(t) = u(t_{n-1}) + \int_{t_{n-1}}^t d_t u(s) ds \quad \text{V}.$$

This implies

$$\begin{aligned} \|u(t)\|_V &\le \|u(t_{n-1})\|_V + \textstyle\int_{t_{n-1}}^t \|d_t u(s)\|_V ds \\ &\le \|u(t_{n-1})\|_V + \Big(\int_{t_{n-1}}^t 1 ds\Big)^{1/2} \Big(\int_{t_{n-1}}^t \|d_t u(s)\|^2_V ds\Big)^{1/2} \quad \forall\, t \in [t_{n-1}, t_n]. \end{aligned}$$

Therefore, we have

$$\|u(t)\|^2_V \le 2\|u(t_{n-1})\|^2_V + 2(t - t_{n-1})\|d_t u\|^2_V$$

Integrating over the interval $I_n$, we get the inequality (7.13), i.e.,

$$\|u\|^2_{L^2(I_n,V)} \leq 2\tau_n \|u(t_{t-1})\|^2_V + \tau_n^2 \|u\|^2_{L^2(I_n,V)}.$$

This completes the proof. □

We will also make use of the duality argument to prove an optimal error estimate. Therefore, we need the following definitions and regularity assumption. Let $A' : V \to V'$ be the dual operator such that

$$\langle A'u, v\rangle := a'(u,v) := a(u,v) \quad \forall\, u,v \in V,$$

where the bilinear form $a' : V \times V \to \mathbb{R}$ satisfying the following assumptions

$$a'(v,v) \geq k_1 \|v\|^2_V \qquad \forall\, v \in V, \tag{7.14}$$

$$|a'(u,v)| \leq k_2 \|u\|_V \|v\|_V \qquad \forall\, u,v \in V, \tag{7.15}$$

with constants $k_1, k_2 > 0$. Now, we consider the problem

$$\int_0^T \langle d_t w(t), v(t)\rangle\, dt + \int_0^T \big\langle A'w(t), v(t)\big\rangle\, dt = \int_0^T (r(t), v(t))_H dt \quad \forall v \in Y. \tag{7.16}$$

Then, the problem (7.16) has a unique solution $w \in X_0$ for the right hand side $r \in L^2(I,H)$. The regularity assumption for this problem is that the solution $w \in X_0$ satisfies $w \in H^1(I,V)$ for every right hand side $r \in L^2(I,H)$ and also a priori estimate

$$\|d_t w\|_{L^2(I,V)} \leq k_3 \|r\|_{L^2(I,H)} \tag{7.17}$$

for some constant $k_3$ independent of $r$ and $T$.

**Theorem 4** *Let $u^0 : [0,T] \to V$ be the solution of the heat equation 7.8 which is sufficiently smooth with respect to time such that the approximation properties* (7.10) *and* (7.11) *are fulfilled for $u = u^0$. Assume that the regularity assumption* (7.17) *for the dual problem is satisfied. Then, the solution $u^0_\tau \in X^k_{\tau,0}$ of the corresponding discrete problem of the exact cGP(k)-method satisfies the following error estimate*

$$\|u^0 - u^0_\tau\|_{L^2(I,V)} \leq C\tau \Big\{ \sum_{n=1}^N \tau_n^{2k} \|d_t^{k+1} u^0\|^2_{L^2(I_n,V)} \Big\}^{1/2} \leq C\tau^{k+1} \|d_t^{k+1} u^0\|_{L^2(I,V)}, \tag{7.18}$$

*where the constants $C$ are independent of $\tau_n$, $\tau$, $T$ and $u^0$.*

**Proof.** (From [49]) Let us define the continuous dual problem for the error $e := u^0 - u^0_\tau \in L^2(I,V)$: *Find a function $z : [0,T] \to V$ such that $z(T) = 0$ and*

$$d_t z(t) + A'z(t) = e(t) \quad \forall\, t \in (0,T). \tag{7.19}$$

For the function $w(t) := z(T-t)$, we get the problem:
*Find a function $w : [0,T] \to V$ such that $w(0) = 0$ and*

$$d_t w(t) + A'w(t) = e(t) \quad \forall\, t \in (0,T). \tag{7.20}$$

The weak formulation of this problem is just problem (7.16) with $r = e$. Using the regularity assumption (7.17) to transform the problem back to $z$, we have

$$\|d_t z\|_{L^2(I,V)} \leq C_a \|e\|_{L^2(I,V)} \tag{7.21}$$

and the equation

$$-\int_0^T \langle d_t z(t), v(t)\rangle\, dt + \int_0^T \langle A' z(t), v(t)\rangle\, dt = \int_0^T (e(t), v(t))_H dt \quad \forall v \in Y. \tag{7.22}$$

Thus, in particular, we have $z \in C(\bar{I}, V)$ since $z \in H^1(I,V)$. Let $\prod_0 : H^1(I,V) \to Y_\tau^k$ be an interpolation operator which is defined on each time interval $I_n$ as

$$\prod_0 z(t) := z(t_{n-1}) \quad \forall\, t \in I_n.$$

Using the inequality (7.13) for $u = z - \prod_0 z$ we have

$$\|z - \prod_0 z\|^2_{L^2(I_n,V)} \leq \tau_n^2 . \|d_t z\|^2_{L^2(I_n,V)}$$

We choose $v = e$ in equation (7.22) such that $e(0) = 0$. Using Galerkin orthogonality of the cGP($k$)-method, we have

$$\begin{aligned}
\|e\|_{L^2(I,V)} &= \int_0^T \langle d_t e(t), z(t)\rangle\, dt + \int_0^T a(e(t), z(t) dt \\
&= \int_0^T \langle d_t e(t), z(t) - \prod_0 z(t)\rangle\, dt + \int_0^T a(e(t), z(t) - \prod_0 z(t) dt \\
&\leq \|d_t e\|_{L^2(I,V')} \|z(t) - \prod_0 z(t)\|_{L^2(I,V)} + C_2 \|e\|_{L^2(I,V)} \|z(t) - \prod_0 z(t)\|_{L^2(I,V)} \\
&\leq \Big\{ \|d_t e\|_{L^2(I,V')} + C_2 \|e\|_{L^2(I,V)} \Big\} \tau_n \|d_t z\|_{L^2(I_n,V)} \\
&\leq C \|e\|_X C_a \tau_n \|e\|_{L^2(I,V)}
\end{aligned}$$

By applying Theorem 2, we obtain the following error estimate

$$\|u^0 - u_\tau^0\|_{L^2(I,V)} \leq C\tau \Big\{ \sum_{n=1}^N \tau_n^{2k} \|d_t^{k+1} u^0\|^2_{L^2(I_n,V)} \Big\}^{1/2} \quad \leq C\tau^{k+1} \|d_t^{k+1} u^0\|_{L^2(I,V)},$$

which completes the proof. □

## 7.3. Stability of the dG($k$)-method

Next, we analyze the stability of the discontinuous Gherkin (dG($k$)-method). To this end, we gain consider problem (7.1) *Find a function* $u : [0,T] \to \mathbb{C}$ *such that*

$$\begin{aligned} d_t u(t) &= \lambda u(t) \quad \forall\, t \in [0,T] \\ u(0) &= u_0, \end{aligned} \tag{7.23}$$

for a given $\lambda \in \mathbb{C}$ and an initial value $u_0 \in \mathbb{C}$. As for the cGP($k$)-method, we consider the equivalent form of problem 7.23 in order to show the A-stability of the dG($k$)-method. This equivalent problem is as follows:
*Find a function* $\vec{u} : [0,T] \to \mathbb{C}$ *such that*

$$\begin{aligned} d_t \vec{u}(t) &= A\vec{u}(t) \quad \forall\, t \in [0,T] \\ \vec{u}(0) &= \vec{u}_0, \end{aligned} \tag{7.24}$$

where $A$ is $2 \times 2$ matrix given by

$$A := \begin{pmatrix} Re(\lambda) & -Im(\lambda) \\ Im(\lambda) & Re(\lambda) \end{pmatrix}$$

and the initial value $\vec{u}_0 := (Re(u_0), Im(u_0))^T$. The matrix $A$ is a negative definite matrix, i.e., there exist a real number $\alpha > 0$

$$\langle Av, v\rangle \leq -\alpha\|v\|^2 \qquad \forall\, v \in \mathbb{R}^2. \tag{7.25}$$

In the following theorem we prove the A-stability of the dG($k$)-method.

**Theorem 5** *Let $u_0 \in \mathbb{R}^2$ be the initial value of the model problem* (7.24) *and $u_1 := u_\tau|_{I_1}(t_1) = u_1^-$ with $t_1 := \tau$ and $I_1 := (0, t_1]$ is the left sided value at time $t = t_1$ of the discrete solution $u_\tau$ of the dG(k)-method applied to problem* (7.24)*. Then, under the assumption* (7.30) *with the parameter $\alpha > 0$ for the matrix A, the following estimate holds*

$$\|u_1\| \leq \frac{1}{\sqrt{1 + \hat{w}_{k+1}\alpha\tau}}\|u_0\| < \|u_0\| \qquad \forall \tau > 0, \tag{7.26}$$

*where $\hat{w}_k > 0$ denotes the weight of the k-point Gauss formula on the unit interval $[-1, 1]$ associated to $\hat{t}_k$. Thus, the dG(k)-method is A-stable.*

**Proof.** (From [40]) The application of the discontinuous Galerkin (dG($k$)-method) to problem (7.24) reads the following:
*Find $u_\tau \in Y_\tau^k$ such that $u_\tau(0) = u_0$ and for all $v_\tau \in Y_\tau^k$ it holds*

$$\sum_{n=1}^{N} \int_{I_n} \langle d_t u_\tau(t), v_\tau(t)\rangle\, dt + \sum_{n=1}^{N} \left\langle [u_\tau]_{n-1}, v_{n-1}^+ \right\rangle = \int_0^T \langle Au_\tau(t), v_\tau(t)\rangle\, dt.$$

Here, the exact dG($k$)-method is equivalent to the numerically integrated dG($k$)-method since the $k$-point Gauss formula is exact for the polynomials of degree less or equal to $2k-1$. Now, we choose $v_\tau$ such that $v_\tau = u_\tau$ in the interval $I_1$ and $v_\tau = 0$ outside the time interval $I_1$. By using the assumption (7.30), we obtain

$$\int_{I_1} \langle d_t u_\tau(t), u_\tau(t)\rangle\, dt + \left\langle [u_\tau]_0, v_0^+ \right\rangle \leq -\alpha \int_{I_1} \|u_\tau\|^2 dt \leq -\alpha \frac{\tau_n}{2} \hat{w}_k \|u_1\|^2. \tag{7.27}$$

This implies

$$\frac{1}{2}\|u_1\|^2 - \frac{1}{2}\|u_0^+\|^2 + \left\langle u_0^+ - u_0, u_0^+ \right\rangle \leq -\alpha \frac{\tau_n}{2} \hat{w}_k \|u_1\|^2.$$

Rearranging the terms in the above equation yields

$$(1 + \hat{w}_k \alpha\tau)\|u_1\|^2 \leq -\|u_0^+\|^2 + 2\left\langle u_0, u_0^+ \right\rangle \leq \|u_0^+\|^2.$$

This completes the proof. □

**Theorem 6** *Let $u_0 \in \mathbb{C}$ be the initial value of the model problem* (7.23) *for some complex number $\lambda \in \mathbb{C}$. Furthermore, $u_1(\lambda, \tau) := u_\tau|_{I_1}(t_1) = u_1^-$ with $t_1 := \tau$ and $I_1 := (0, t_1]$ is the left sided value at time $t = t_1$ of the discrete solution $u_\tau$ of the dG(k)-method applied to problem* (7.23)*. Then, it holds*

$$\lim_{Re(\lambda)\tau \to -\infty} \frac{|u_1(\lambda, \tau)|}{|u_0|}, \tag{7.28}$$

*i.e., the dG(k)-method is L-stable.*

**Proof.** (From [40]) We know that the problem 7.23 can be considered with an equivalent problem as follows:
*Find a function $\vec{u} : [0, T] \to \mathbb{C}$ such that*

$$\begin{aligned} d_t \vec{u}(t) &= A\vec{u}(t) \quad \forall\, t \in [0, T] \\ \vec{u}(0) &= \vec{u}_0, \end{aligned} \tag{7.29}$$

where $A$ is $2 \times 2$ matrix given by

$$A := \begin{pmatrix} Re(\lambda) & -Im(\lambda) \\ Im(\lambda) & Re(\lambda) \end{pmatrix}$$

and the initial value $\vec{u}_0 := (Re(u_0), Im(u_0))^T$. The matrix $A$ is a negative definite matrix, i.e., there exist a real number $\alpha > 0$

$$\langle Av, v \rangle \leq -\alpha \|v\|^2 \qquad \forall\, v \in \mathbb{R}^2. \tag{7.30}$$

By using Theorem 5, we have

$$0 \leq \frac{|u_1(\lambda, \tau)|}{|u_0|} = \frac{\|\vec{u}_\tau(\tau)\|}{\|\vec{u}_0\|} \leq \frac{1}{\sqrt{1 + \hat{w}_k \alpha \tau}},$$

which completes the proof. $\square$

# 8

# Galerkin time discretizations for the Stokes equations

In this chapter, we extend our work for the heat equation in Chapter 5 for the nonstationary Stokes equations. We implement and compare numerically *continuous* Galerkin-Petrov (cGP) and *discontinuous* Galerkin (dG) time discretizations for the nonstationary Stokes equations in two dimensions. For the space discretization, we use the LBB-stable finite element pair $Q_2/P_1^{disc}$ and we discuss implementation aspects as well as methods for solving the resulting block systems which are treated by using monolithic multigrid solvers with LPSC type smoothers. By means of numerical experiments we compare the different time discretizations w.r.t. accuracy and computational costs and we show that the convergence behavior of the multigrid method is almost independent of mesh size and time step leading to an efficient solution process.

## 8.1. The cGP- and dG-methods for the Stokes equations

We consider the nonstationary Stokes equations, i.e. we want to find a velocity $\mathbf{u} : \Omega \times [0,T] \to \mathbb{R}^2$ and a pressure $p : \Omega \times [0,T] \to \mathbb{R}$ *such that*

$$\begin{aligned} \partial_t \mathbf{u} - \nu \triangle \mathbf{u} + \nabla p = f, \qquad & \operatorname{div} \mathbf{u} = 0 \quad \text{in} \quad \Omega \times (0,T), \\ \mathbf{u} = 0 \quad \text{on} \quad \partial\Omega \times [0,T], \qquad & \mathbf{u}(x,0) = \mathbf{u}_0(x) \quad \text{in} \quad \Omega \quad \text{for} \quad t = 0, \end{aligned} \tag{8.1}$$

where $\nu$ denotes the viscosity, $f : \Omega \times (0,T) \to \mathbb{R}^2$ is the body force and $\mathbf{u}_0 : \Omega \to \mathbb{R}^2$ the initial velocity field at time $t = 0$. For simplicity, we assume homogeneous Dirichlet conditions at the boundary $\partial\Omega$ of a polygonal domain $\Omega \subset \mathbb{R}^2$. To make this problem well-posed, one needs to impose an additional condition on $p$, i.e., $\int_\Omega p d\Omega = 0$.

We start with the time discretization of problem (8.1) which is of variational type. In the following, let $I = [0,T]$ denote the time interval with some positive final time $T$. For a function $\mathbf{u} : \Omega \times I \to \mathbb{R}^2$ and a fixed $t \in I$, we will denote by $\mathbf{u}(t) := \mathbf{u}(\cdot,t)$ the associated velocity function at time $t$ which is an element of a suitable function space $\mathbf{V}$. In case of the Stokes equations, this space is the Sobolev space $\mathbf{V} = (H_0^1(\Omega))^2$. Similarly, we denote by $p(t) := p(\cdot,t)$ the associated pressure function at time $t$ which is an element of the function space $Q = L_0^2(\Omega)$ where

$$L_0^2(\Omega) = \Big\{ q \in L^2(\Omega) : \int_\Omega q\, dx = 0 \Big\}.$$

In the time discretization, we decompose the time interval $I$ into $N$ subintervals $I_n := [t_{n-1}, t_n)$, where $n = 1, \ldots, N$ and $0 = t_0 < t_1 < \cdots < t_{N-1} < t_N = T$. The symbol $\tau$ will denote the *time discretization parameter* and will also be used as the maximum time step size $\tau := \max_{1 \le n \le N} \tau_n$, where $\tau_n := t_n - t_{n-1}$.

Then, for the cGP($k$)-method, we approximate the solution $\mathbf{u} : I \to \mathbf{V}$ by means of a function $u_\tau : I \to \mathbf{V}$ which is piecewise polynomial of order $k$ with respect to time, i.e., we are looking for

$u_\tau$ in the discrete time space

$$\mathbf{X}_\tau^k := \{\mathbf{u} \in C(I,\mathbf{V}) : \mathbf{u}\big|_{I_n} \in \mathbb{P}_k(I_n,\mathbf{V}) \quad \forall\, n = 1,\dots,N\}, \tag{8.2}$$

where

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

We introduce the time discrete test space

$$\mathbf{Y}_\tau^{k-1} := \{\mathbf{v} \in L^2(I,\mathbf{V}) : \mathbf{v}\big|_{I_n} \in \mathbb{P}_{k-1}(I_n,\mathbf{V}) \quad \forall\, n = 1,\dots,N\} \tag{8.3}$$

consisting of piecewise polynomials of order $k-1$ which are globally discontinuous at the end points of the time intervals. Similarly, we will use for the time discrete pressure $p_\tau$ an analogous ansatz space $X_\tau^k$, where the vector valued space $\mathbf{V}$ is replaced by the scalar valued space $Q$, and an analogous discontinuous test space $Y_\tau^{k-1}$.

Now, in order to derive the time discretization, we multiply the momentum equation in (8.1) with some suitable $I_n$-supported test functions $v_\tau \in \mathbf{Y}_\tau^{k-1}$, integrate over $\Omega \times I_n$, use Fubini's Theorem and partial space integration of the terms $\triangle\mathbf{u}$ and $\nabla p$ and apply the $k$-point Gaussian quadrature rule for the evaluation of the time integrals. To determine $u_\tau|_{I_n}$ and $p_\tau|_{I_n}$ we represent them by the polynomial ansatz

$$u_\tau(t) := \sum_{j=0}^{k} \mathbf{U}_n^j \phi_{n,j}(t), \qquad p_\tau(t) := \sum_{j=0}^{k} P_n^j \phi_{n,j}(t), \tag{8.4}$$

where the "coefficients" $(\mathbf{U}_n^j, P_n^j)$ are elements of the Hilbert space $\mathbf{V} \times Q$ and the real functions $\phi_{n,j} \in \mathbb{P}_k(I_n)$ are the Lagrange basis functions with respect to $k+1$ suitable nodal points $t_{n,j} \in I_n$ satisfying the conditions

$$\phi_{n,j}(t_{n,i}) = \delta_{i,j}, \qquad i,j = 0,\dots,k \tag{8.5}$$

with the Kronecker symbol $\delta_{i,j}$. For an easy treatment of the initial condition, we set $t_{n,0} = t_{n-1}$. Then, the initial condition is equivalent to the condition

$$\mathbf{U}_n^0 = u_\tau|_{I_{n-1}}(t_{n-1}) \quad \text{if} \quad n \geq 2 \qquad \text{or} \qquad \mathbf{U}_n^0 = \mathbf{u}_0 \quad \text{if} \quad n = 1. \tag{8.6}$$

The other points $t_{n,1},\dots,t_{n,k}$ are chosen as the quadrature points of the $k$-point Gaussian formula on $I_n$. This formula is exact if the function to be integrated is a polynomial of degree less or equal to $2k-1$. We define the basis functions $\phi_{n,j} \in \mathbb{P}_k(I_n)$ of (8.4) via the affine reference transformation $T_n : \hat{I} \to I_n$ where $\hat{I} := [-1,1]$ and

$$t = T_n(\hat{t}) := \frac{t_{n-1}+t_n}{2} + \frac{\tau_n}{2}\hat{t} \in I_n \qquad \forall\, \hat{t} \in \hat{I},\ n = 1,\dots,N. \tag{8.7}$$

Let $\hat{\phi}_j \in \mathbb{P}_k(\hat{I})$, $j = 0,\dots,k$, denote the basis functions satisfying the conditions

$$\hat{\phi}_j(\hat{t}_i) = \delta_{i,j}, \qquad i,j = 0,\dots,k, \tag{8.8}$$

where $\hat{t}_0 = -1$ and $\hat{t}_i$, $i = 1,\dots,k$, are the standard *Gaussian quadrature points* for the reference interval $\hat{I}$. Then, we define the basis functions on the original time interval $I_n$ by

$$\phi_{n,j}(t) := \hat{\phi}_j(\hat{t}) \qquad \text{with} \qquad \hat{t} := T_n^{-1}(t) = \frac{2}{\tau_n}\left(t - \frac{t_n - t_{n-1}}{2}\right) \in \hat{I}. \tag{8.9}$$

At the end, we obtain the following *time discrete $I_n$-problem of the cGP(k)-method* [28, 49]:

Find on interval $I_n = [t_{n-1}, t_n)$ the $k$ unknown pairs of "coefficients" $(\mathbf{U}_n^j, P_n^j) \in \mathbf{V} \times Q$, $j = 1, \ldots, k$, such that for all $i = 1, \ldots, k$, it holds

$$\begin{aligned} \sum_{j=0}^{k} \alpha_{i,j} \left(\mathbf{U}_n^j, \mathbf{v}\right)_\Omega + \tfrac{\tau_n}{2} a(\mathbf{U}_n^i, \mathbf{v}) + \tfrac{\tau_n}{2} b(\mathbf{v}, P_n^i) &= \tfrac{\tau_n}{2} \left(f(t_{n,i}), \mathbf{v}\right)_\Omega \qquad \forall\, \mathbf{v} \in \mathbf{V}, \\ b(\mathbf{U}_n^i, q) &= 0 \qquad \forall\, q \in Q, \end{aligned} \tag{8.10}$$

where $\tau_n$ denotes the length of the time interval $I_n$, $\mathbf{U}_n^0 := u_\tau(t_{n-1})$ for $n > 1$, $\mathbf{U}_1^0 := \mathbf{u}_0$ and $(\cdot,\cdot)_\Omega$ the usual inner product in $L^2(\Omega)$. The bilinear forms $a(\cdot,\cdot)$ and $b(\cdot,\cdot)$ on $\mathbf{V} \times \mathbf{V}$ and $\mathbf{V} \times Q$, respectively, are defined as

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

A typical property of this cGP($k$)-variant is that the initial pressure $P_n^0$ of the ansatz (8.4) does not occur in this formulation. This will be the reason for some problems to achieve superconvergence for the pressure approximation at the discrete time levels $t_n$.

In the following subsections, we specify the constants $\alpha_{i,j}$ of the cGP($k$)-method for the cases $k = 1$ and $k = 2$ and we describe explicitly the well-known dG(1) approach.

### 8.1.1. cGP(1)-method

We use the one-point Gaussian quadrature formula with the point $\hat{t}_1 = 0$ and $t_{n,1} = t_{n-1} + \frac{\tau_n}{2}$. Then, we get $\alpha_{1,0} = -1$ and $\alpha_{1,1} = 1$. Thus, equation (8.10) leads to the following equation for the "one" unknown $\mathbf{U}_n^1 = u_\tau(t_{n-1} + \frac{\tau_n}{2}) \in \mathbf{V}$ and $P_n^1 = p_\tau(t_{n-1} + \frac{\tau_n}{2}) \in Q$

$$\begin{aligned} \left(\mathbf{U}_n^1, \mathbf{v}\right)_\Omega + \tfrac{\tau_n}{2} a(\mathbf{U}_n^1, \mathbf{v}) + \tfrac{\tau_n}{2} b(\mathbf{v}, P_n^1) &= \tfrac{\tau_n}{2} \left(f(t_{n,1}), \mathbf{v}\right)_\Omega + \left(\mathbf{U}_n^0, \mathbf{v}\right)_\Omega \qquad \forall\, \mathbf{v} \in \mathbf{V} \\ b(\mathbf{U}_n^1, q) &= 0 \qquad \forall\, q \in Q. \end{aligned} \tag{8.11}$$

Once we have determined the solution $\mathbf{U}_n^1$ at the midpoint $t_{n,1}$ of the time interval $I_n$, we get the solution at the next discrete time point $t_n$ simply by polynomial interpolation with the ansatz (8.4), i.e.,

$$u_\tau(t_n) = 2\mathbf{U}_n^1 - \mathbf{U}_n^0, \tag{8.12}$$

where $\mathbf{U}_n^0$ is the initial value at the time interval $[t_{n-1}, t_n)$ coming from the previous time interval $I_{n-1}$ or the initial value $\mathbf{u}_0$.

If we would replace $f(t_{n,1})$ by the mean value $(f(t_{n-1}) + f(t_n))/2$, which means that we replace the one-point Gaussian quadrature of the right hand side by the Trapezoidal rule, the resulting cGP(1)-method is equivalent to the well-known *Crank-Nicolson scheme*.

### 8.1.2. cGP(2)-method

Here, we use the 2-point Gaussian quadrature formula with the points $\hat{t}_1 = -\frac{1}{\sqrt{3}}$ and $\hat{t}_2 = \frac{1}{\sqrt{3}}$. Then, we obtain the coefficients

$$(\alpha_{i,j}) = \begin{pmatrix} -\sqrt{3} & \frac{3}{2} & \frac{2\sqrt{3}-3}{2} \\ \sqrt{3} & \frac{-2\sqrt{3}-3}{2} & \frac{3}{2} \end{pmatrix} \qquad i = 1,2,\ j = 0,1,2.$$

On the time interval $I_n$, we have to solve for the two "unknowns"

$$(\mathbf{U}_n^j, P_n^j) = \left(u_\tau(t_{n,j}), p_\tau(t_{n,j})\right) \in \mathbf{V} \times Q \qquad \text{with} \qquad t_{n,j} := T_n(\hat{t}_j) \quad \text{for} \quad j = 1,2.$$

The corresponding coupled system reads:

$$\begin{aligned}
\alpha_{1,1}\left(\mathbf{U}_n^1,\mathbf{v}\right)_\Omega + \tfrac{\tau_n}{2}a(\mathbf{U}_n^1,\mathbf{v}) + \alpha_{1,2}\left(\mathbf{U}_n^2,\mathbf{v}\right)_\Omega + \tfrac{\tau_n}{2}b(\mathbf{v},P_n^1) &= \tfrac{\tau_n}{2}\left(f(t_{n,1}),\mathbf{v}\right)_\Omega - \alpha_{1,0}\left(\mathbf{U}_n^0,\mathbf{v}\right)_\Omega \\
\alpha_{2,1}\left(\mathbf{U}_n^1,\mathbf{v}\right)_\Omega + \alpha_{2,2}\left(\mathbf{U}_n^2,\mathbf{v}\right)_\Omega + \tfrac{\tau_n}{2}a(\mathbf{U}_n^2,\mathbf{v}) + \tfrac{\tau_n}{2}b(\mathbf{v},P_n^2) &= \tfrac{\tau_n}{2}\left(f(t_{n,2}),\mathbf{v}\right)_\Omega - \alpha_{2,0}\left(\mathbf{U}_n^0,\mathbf{v}\right)_\Omega \\
b(\mathbf{U}_n^1,q) &= 0 \\
b(\mathbf{U}_n^2,q) &= 0,
\end{aligned} \tag{8.13}$$

which has to be satisfied for all $\mathbf{v}\in\mathbf{V}$ and $q\in Q$. Once we have determined the solutions $(\mathbf{U}_n^j,P_n^j)$ at the Gaussian points in the interior of the interval $I_n$, we get the solution at the right boundary $t_n$ of $I_n$ again by means of polynomial interpolation from the ansatz (8.4), i.e.,

$$u_\tau(t_n) = \mathbf{U}_n^0 + \sqrt{3}(\mathbf{U}_n^2 - \mathbf{U}_n^1), \tag{8.14}$$

where $\mathbf{U}_n^0$ is the initial value at the time interval $I_n$.

### 8.1.3. dG(1)-method

In the dG(1)-method, velocity and pressure are approximated by a discontinuous piecewise linear ansatz space, i.e. $(u_\tau,p_\tau)\in\mathbf{Y}_\tau^1\times Y_\tau^1$. On time interval $I_n$ we use the polynomial representation

$$u_\tau(t) := \sum_{j=1}^{2}\mathbf{U}_n^j\phi_{n,j}(t), \qquad p_\tau(t) := \sum_{j=1}^{2}P_n^j\phi_{n,j}(t), \tag{8.15}$$

with the two "coefficients" $(\mathbf{U}_n^j,P_n^j)\in\mathbf{V}\times Q$, $j=1,2$, which are the values of $u_\tau$ and $p_\tau$, respectively, at the points $t_{n,j}\in I_n$ of the 2 point Gaussian formula. The real functions $\phi_{n,j}\in\mathbb{P}_1(I_n)$ are the linear Lagrange basis functions with respect to these two Gaussian points.

In order to present the method, we use the following constants for $i,j\in\{1,2\}$

$$(\gamma_{i,j}) = \begin{pmatrix} 1 & \frac{\sqrt{3}-1}{2} \\ \frac{-\sqrt{3}-1}{2} & 1 \end{pmatrix}, \quad (d_i) = \begin{pmatrix} \frac{\sqrt{3}+1}{2} \\ \frac{-\sqrt{3}+1}{2} \end{pmatrix}.$$

Then, on the time interval $I_n$, one has to determine the two "unknowns" $(\mathbf{U}_n^j,P_n^j)\in\mathbf{V}\times Q$ as the solution of the following coupled system:

$$\begin{aligned}
\gamma_{1,1}\left(\mathbf{U}_n^1,\mathbf{v}\right)_\Omega + \tfrac{\tau_n}{2}a(\mathbf{U}_n^1,\mathbf{v}) + \tfrac{\tau_n}{2}b(\mathbf{v},P_n^1) + \gamma_{1,2}\left(\mathbf{U}_n^2,\mathbf{v}\right)_\Omega &= d_1\left(\mathbf{U}_n^0,\mathbf{v}\right)_\Omega + \tfrac{\tau_n}{2}\left(f(t_{n,1}),\mathbf{v}\right)_\Omega, \\
\gamma_{2,1}\left(\mathbf{U}_n^1,\mathbf{v}\right)_\Omega + \gamma_{2,2}\left(\mathbf{U}_n^2,\mathbf{v}\right)_\Omega + \tfrac{\tau_n}{2}a(\mathbf{U}_n^2,\mathbf{v}) + \tfrac{\tau_n}{2}b(\mathbf{v},P_n^2) &= d_2\left(\mathbf{U}_n^0,\mathbf{v}\right)_\Omega + \tfrac{\tau_n}{2}\left(f(t_{n,2}),\mathbf{v}\right)_\Omega, \\
b(\mathbf{U}_n^1,q) &= 0 \\
b(\mathbf{U}_n^2,q) &= 0
\end{aligned} \tag{8.16}$$

which has to be satisfied for all $\mathbf{v}\in\mathbf{V}$ and $q\in Q$. Once we have solved the above system, we obtain $u_\tau$ and $p_\tau$ at the time $t_n$ by means of the following linear interpolation

$$u_\tau(t_n) = \frac{\sqrt{3}+1}{2}\mathbf{U}_n^2 - \frac{\sqrt{3}-1}{2}\mathbf{U}_n^1 \qquad \text{and} \qquad p_\tau(t_n) = \frac{\sqrt{3}+1}{2}P_n^2 - \frac{\sqrt{3}-1}{2}P_n^1. \tag{8.17}$$

## 8.2. Space Discretization by FEM

Next, in each time step, we apply a standard Galerkin finite element discretization with the so-called $Q_2/P_1^{disc}$ Stokes element, i.e., with biquadratic finite elements for the velocity and discontinuous piecewise linear elements for the pressure. This LBB-stable element pair leads to an

$L_2$-approximation order of $O(h^3)$ for the velocity and $O(h^2)$ for the pressure where $h$ denotes the mesh size of the space grid.

After discretizing the Stokes equations (8.1) in time, we now discretize the resulting "$I_n$-problems" in space by using the finite element method [11, 17, 35, 55]. In our numerical experiments, the finite element spaces $\mathbf{V}_h \subset \mathbf{V}$ and $Q_h \subset Q$ are defined by biquadratic and discontinuous linear finite elements, respectively, on a quadrilateral mesh $T_h$ covering the computational domain $\Omega$. Each "$I_n$-problem" for the cGP($k$) or the dG($k-1$)-approach has the structure:

*For given $\mathbf{U}_n^0 \in \mathbf{V}$, find $\mathbf{U}_n^1, \ldots, \mathbf{U}_n^k \in \mathbf{V}$ and $P_n^1, \ldots, P_n^k$ such that*

$$\begin{aligned} \sum_{j=1}^{k} \alpha_{i,j} \left(\mathbf{U}_n^j, \mathbf{v}\right)_\Omega + \frac{\tau_n}{2} a(\mathbf{U}_n^i, \mathbf{v}) + \frac{\tau_n}{2} b(\mathbf{v}, P_n^i) &= \ell_i(\mathbf{v}), \\ b(\mathbf{U}_n^i, q) &= 0, \end{aligned} \tag{8.18}$$

*which has to be satisfied for all $\mathbf{v} \in \mathbf{V}$ and $q \in Q$ with*

$$\ell_i(\mathbf{v}) = \frac{\tau_n}{2} \left(f(t_{n,i}), \mathbf{v}\right)_\Omega + d_i \left(\mathbf{U}_n^0, \mathbf{v}\right)_\Omega \qquad \forall i = 1, \ldots, k, \tag{8.19}$$

*where $\alpha_{i,j}$ and $d_i$ are the corresponding constants.* For the space discretization, each $\mathbf{U}_n^j \in \mathbf{V}$ and $P_n^j \in Q$ are approximated by a finite element function $\mathbf{U}_{n,h}^j \in \mathbf{V}_h$ and $P_{n,h}^j \in Q_h$, resp., and the fully discrete "$I_n$-problem" reads:

*For given $\mathbf{U}_{n,h}^0 \in \mathbf{V}_h$, find $\mathbf{U}_{n,h}^1, \ldots, \mathbf{U}_{n,h}^k \in \mathbf{V}_h$ and $P_{n,h}^1, \ldots, P_{n,h}^k \in Q_h$ such that for all $\mathbf{v}_h \in \mathbf{V}_h$, $q_h \in Q_h$*

$$\begin{aligned} \sum_{j=1}^{k} \alpha_{i,j} \left(\mathbf{U}_{n,h}^j, \mathbf{v}_h\right)_\Omega + \frac{\tau_n}{2} a(\mathbf{U}_{n,h}^i, \mathbf{v}_h) + \frac{\tau_n}{2} b(\mathbf{v}_h, P_{n,h}^i) &= \ell_i(\mathbf{v}), \\ b(\mathbf{U}_{n,h}^i, q_h) &= 0, \end{aligned} \tag{8.20}$$

*for all $i = 1, \ldots, k$.* Once we have solved this system, we have computed for each time $t \in I_n$ a finite element approximation $u_{\tau,h}(t) \in \mathbf{V}_h$ of the time discrete solution $u_\tau(t) \in \mathbf{V}$. To this end, we replace the continuous functions $\mathbf{U}_n^j \in \mathbf{V}$ in the ansatz of $u_\tau(t)$ by the discrete functions $\mathbf{U}_{n,h}^j \in \mathbf{V}_h$.

In the following, we will write the problem (8.20) as a linear algebraic block system. Let $\phi_\mu \in \mathbf{V}_h$, $\mu = 1, \ldots, m_h$, denote the finite element basis functions and $\underline{\mathbf{U}}_n^j \in \mathbb{R}^{2m_h}$ the nodal vector of $\mathbf{U}_{n,h}^j = (U_{n,h}^j, V_{n,h}^j) \in \mathbf{V}_h$ such that

$$U_{n,h}^j(x) = \sum_{\mu=1}^{m_h} (\underline{U}_n^j)_\mu \phi_\mu(x), \qquad V_{n,h}^j(x) = \sum_{\mu=1}^{m_h} (\underline{V}_n^j)_\mu \phi_\mu(x) \qquad \forall\, x \in \Omega.$$

Similarly for the pressure, let $\psi_\mu \in Q_h$, $\mu = 1, \ldots, n_h$, denote the finite element basis functions and $\underline{P}_n^j \in \mathbb{R}^{n_h}$ the nodal vector of $P_{n,h}^j \in Q_h$ such that

$$P_{n,h}^j(x) = \sum_{\mu=1}^{n_h} (\underline{P}_n^j)_\mu \psi_\mu(x) \qquad \forall\, x \in \Omega.$$

Furthermore, let us introduce the mass matrix $M$, the discrete Laplacian matrix $L$ and the vector $F_n^i$ with the following components

$$M_{\nu,\mu} := (\phi_\mu, \phi_\nu)_\Omega, \qquad L_{\nu,\mu} := a(\phi_\mu, \phi_\nu), \qquad B_{\nu,\mu} := b(\phi_\mu, \psi_\nu), \qquad (F_n^i)_\nu := \left(f(t_{n,i}), \phi_\nu\right)_\Omega. \tag{8.21}$$

Then the fully discrete "$I_n$-problem" is equivalent to the following (nonlinear) $k \times k$ block system:
*For given $\underline{\mathbf{U}}_n^0$, find $\underline{\mathbf{U}}_n^1, \ldots, \underline{\mathbf{U}}_n^k$ and $\underline{P}_n^1, \ldots, \underline{P}_n^k$ such that*

$$\begin{aligned} \sum_{j=1}^{k} \alpha_{i,j} \mathbf{M} \underline{\mathbf{U}}_n^j + \frac{\tau_n}{2} \mathbf{L} \underline{\mathbf{U}}_n^i + \frac{\tau_n}{2} \mathbf{B} \underline{P}_n^i &= d_i \mathbf{M} \underline{\mathbf{U}}_n^0 + \tfrac{\tau_n}{2} \mathbf{F}_n^i, \qquad \forall\, i = 1, \ldots, k \\ \mathbf{B}^T \underline{\mathbf{U}}_n^i &= 0, \end{aligned} \tag{8.22}$$

where

$$\mathbf{M} = \begin{bmatrix} M & 0 \\ 0 & M \end{bmatrix}, \qquad \mathbf{L} = \begin{bmatrix} L & 0 \\ 0 & L \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$$

and the right hand side vector $\mathbf{F}_n^i$ is given by

$$\mathbf{F}_n^i = \begin{bmatrix} F_n^i \\ G_n^i \end{bmatrix}.$$

The vector $\underline{\mathbf{U}}_n^0$ is defined as the finite element nodal vector of the fully discrete solution $u_{\tau,h}(t_{n-1})$ computed from the previous time interval $[t_{n-2}, t_{n-1}]$ if $n \geq 2$ or from a finite element interpolation of the initial data $\mathbf{u}_0$ if $n = 1$.

In the following, we will present the resulting block systems for the cGP(1)-, cGP(2)- and dG(1)-method which are used in our numerical experiments.

### 8.2.1. cGP(1)-method

The corresponding $3 \times 3$ block system on each time interval $I_n$ reads: *For given initial velocity coefficient vectors $\underline{\mathbf{U}}_n^0 = (\underline{U}_n^0, \underline{V}_n^0)$, find $\underline{U}_n^1, \underline{V}_n^1$ and a scaled pressure vector $\underline{\tilde{P}}_n^1$ such that*

$$\begin{aligned} \left(M + \tfrac{\tau_n}{2} A\right) \underline{U}_n^1 + B_1 \underline{\tilde{P}}_n^1 &= \tfrac{\tau_n}{2} F_n^1 + M \underline{U}_n^0 \\ \left(M + \tfrac{\tau_n}{2} A\right) \underline{V}_n^1 + B_2 \underline{\tilde{P}}_n^1 &= \tfrac{\tau_n}{2} G_n^1 + M \underline{V}_n^0 \\ B_1^T \underline{U}_n^1 + B_2^T \underline{V}_n^1 &= 0 \end{aligned}$$

where $\underline{\tilde{P}}_n^1 := \frac{\tau_n}{2} \underline{P}_n^1$, and $M, A$ and $B$ denote the mass, Laplacian and gradient matrices, respectively. Once we have determined the solution $\underline{U}_n^1$, $\underline{V}_n^1$ we compute the nodal vector $\underline{U}_{n+1}^0$, $\underline{V}_{n+1}^0$ of the discrete solution $u_{\tau,h}$ at the time $t_n$ by using the following linear extrapolation

$$u_{\tau,h}(t_n) \sim \underline{U}_{n+1}^0 = 2\underline{U}_n^1 - \underline{U}_n^0, \qquad v_{\tau,h}(t_n) \sim \underline{V}_{n+1}^0 = 2\underline{V}_n^1 - \underline{V}_n^0.$$

### 8.2.2. cGP(2)-method

The $6 \times 6$ block system on each time interval $I_n$ reads: *For given initial velocity vectors $\underline{\mathbf{U}}_n^0 = (\underline{U}_n^0, \underline{V}_n^0)$, find $\underline{U}_n^1, \underline{U}_n^2, \underline{V}_n^1, \underline{V}_n^2$ and scaled pressure vectors $\underline{\tilde{P}}_n^1, \underline{\tilde{P}}_n^2$ such that*

$$\begin{pmatrix} 3M + \tau_n A & \left(2\sqrt{3} - 3\right) M & 0 & 0 & B_1 & 0 \\ \left(-2\sqrt{3} - 3\right) M & 3M + \tau_n A & 0 & 0 & 0 & B_1 \\ 0 & 0 & 3M + \tau_n A & \left(2\sqrt{3} - 3\right) M & B_2 & 0 \\ 0 & 0 & \left(-2\sqrt{3} - 3\right) M & 3M + \tau_n A & 0 & B_2 \\ B_1^T & 0 & B_2^T & 0 & 0 & 0 \\ 0 & B_1^T & 0 & B_2^T & 0 & 0 \end{pmatrix} \begin{pmatrix} \underline{U}_n^1 \\ \underline{U}_n^2 \\ \underline{V}_n^1 \\ \underline{V}_n^2 \\ \underline{\tilde{P}}_n^1 \\ \underline{\tilde{P}}_n^2 \end{pmatrix} = \begin{pmatrix} R_n^1 \\ R_n^2 \\ R_n^3 \\ R_n^4 \\ 0 \\ 0 \end{pmatrix}$$

where $\underline{\tilde{P}}_n^i := \tau_n \underline{P}_n^i$ and

$$\begin{aligned} R_n^1 &= \tau_n F_n^1 + 2\sqrt{3} M \underline{U}_n^0, \qquad R_n^2 = \tau_n F_n^2 - 2\sqrt{3} M \underline{U}_n^0, \\ R_n^3 &= \tau_n G_n^1 + 2\sqrt{3} M \underline{V}_n^0, \qquad R_n^4 = \tau_n G_n^2 - 2\sqrt{3} M \underline{V}_n^0. \end{aligned}$$

Here, we compute the nodal vector $\underline{U}^0_{n+1}$ and $\underline{V}^0_{n+1}$ of the fully discrete solution $u_{\tau,h}$ at the time $t_n$ by using the following quadratic extrapolation

$$u_{\tau,h}(t_n) \sim \underline{U}^0_{n+1} = \underline{U}^0_n + \sqrt{3}(\underline{U}^2_n - \underline{U}^1_n), \qquad v_{\tau,h}(t_n) \sim \underline{V}^0_{n+1} = \underline{V}^0_n + \sqrt{3}(\underline{V}^2_n - \underline{V}^1_n).$$

### 8.2.3. dG(1)-method

The analogues $6 \times 6$ block system on the time interval $I_n$ reads: *For given initial velocity vector* $\underline{\mathbf{U}}^0_n = (\underline{U}^0_n, \underline{V}^0_n)$, *find* $\underline{U}^1_n, \underline{U}^2_n, \underline{V}^1_n, \underline{V}^2_n$ *and scaled pressure coefficient vectors* $\underline{\tilde{P}}^1_n, \underline{\tilde{P}}^2_n$ *such that*

$$\begin{pmatrix} 2M+\tau_n A & \left(\sqrt{3}-1\right)M & 0 & 0 & B_1 & 0 \\ \left(-\sqrt{3}-1\right)M & 2M+\tau_n A & 0 & 0 & 0 & B_1 \\ 0 & 0 & 2M+\tau_n A & \left(\sqrt{3}-1\right)M & B_2 & 0 \\ 0 & 0 & \left(-\sqrt{3}-1\right)M & 2M+\tau_n A & 0 & B_2 \\ B_1^T & 0 & B_2^T & 0 & 0 & 0 \\ 0 & B_1^T & 0 & B_2^T & 0 & 0 \end{pmatrix} \begin{pmatrix} \underline{U}^1_n \\ \underline{U}^2_n \\ \underline{V}^1_n \\ \underline{V}^2_n \\ \underline{\tilde{P}}^1_n \\ \underline{\tilde{P}}^2_n \end{pmatrix} = \begin{pmatrix} R^1_n \\ R^2_n \\ R^3_n \\ R^4_n \\ 0 \\ 0 \end{pmatrix}$$

where $\underline{\tilde{P}}^i_n := \tau_n \underline{P}^i_n$ and

$$\begin{aligned} R^1_n &= \tau_n F^1_n + \left(\sqrt{3}+1\right) M \underline{U}^0_n, \quad R^2_n = \tau_n F^2_n + \left(-\sqrt{3}+1\right) M \underline{U}^0_n, \\ R^3_n &= \tau_n G^1_n + \left(\sqrt{3}+1\right) M \underline{V}^0_n, \quad R^4_n = \tau_n G^2_n + \left(-\sqrt{3}+1\right) M \underline{V}^0_n. \end{aligned}$$

In this case, we compute the nodal vector $\underline{U}^0_{n+1}$,$\underline{V}^0_{n+1}$ and $\underline{P}^0_{n+1}$ of the left side limit of the fully discrete solution $u_{\tau,h}$ at the time $t_n$ by using the following linear extrapolation

$$u^-_{\tau,h}(t_n) \sim \underline{U}^0_{n+1} = \frac{\sqrt{3}+1}{2} U^2_n - \frac{\sqrt{3}-1}{2} U^1_n, \qquad v^-_{\tau,h}(t_n) \sim \underline{V}^0_{n+1} = \frac{\sqrt{3}+1}{2} V^2_n - \frac{\sqrt{3}-1}{2} V^1_n.$$

One can obtain the pressure at the discrete time points $t_n$ by using the same extrapolation

$$p^-_{\tau,h}(t_n) \sim \underline{P}^0_{n+1} = \frac{\sqrt{3}+1}{2} P^2_n - \frac{\sqrt{3}-1}{2} P^1_n.$$

## 8.3. Postprocessing for high order pressure

In many flow problems, often the hydrodynamic forces such as drag, lift etc, have to be calculated. These forces consist of functionals for velocity and pressure at the same discrete time points. Now, since we have superconvergence results for the velocity only at the discrete time points $t_n$, it is desirable to get a high order pressure at the same points. In contrast to the dG(1)-method, we cannot obtain the pressure in cGP-methods at the discrete time points by using the same extrapolation as for velocity since this would involve the initial pressure which we do not have. In this section, we explain how to get higher order accuracy for the pressure in the cGP-methods at the discrete time points $t_n$ from the obtained pressure at the intermediate $k$ Gaussian points in the subinterval $[t_{n-1}, t_n]$. The same technique is then also applied for the dG(1)-method which gives better results than the associated extrapolation. To do this, we use the Lagrangian interpolation polynomials to get the solution at time $t_n$ which we explain in the following for the cGP(1), cGP(2) and dG(1)-method, respectively.

### 8.3.1. cGP(1)-method

We consider the left and right subinterval at time $t_n$. Let $t_0$ and $t_1$ be the intermediate Gaussian points in these subintervals where the solution is already known. We can construct the corresponding linear Lagrangian interpolation polynomial $L(t)$ such that $L(t_i) = p_\tau(t_i)$ for $i = 0, 1$. Once this linear polynomial $L(t)$ is obtained we can get the solution at the discrete time point $t_n$. In this way, one extra time step would be required to compute the solution at the end point $t = T$ of the simulation. The corresponding interpolation is visualized in Figure 8.1.

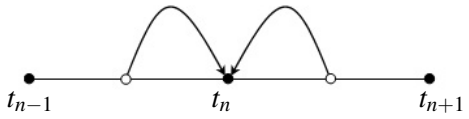



**Figure 8.1**: Lagrange interpolation for pressure at the discrete time point $t_n$.

### 8.3.2. cGP(2) and dG(1)-method

In case of cGP(2) or dG(1)-method, we have two Gaussian points in each subinterval $I_n$. Let $t_0, t_1, t_2$ and $t_3$ be the four points in the neighboring subintervals at $t_n$. Now we construct the cubic Lagrangian polynomial passing through these four points. Once we have determined this polynomial $L(t)$ we obtain the solution at the next discrete time point $t_n$ (see Figure 8.2). As in case of cGP(1), one more time step is needed to find the solution at the end point.

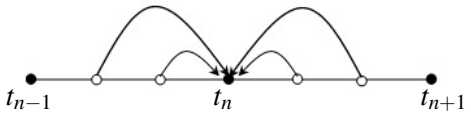



**Figure 8.2**: Lagrange interpolation for pressure at the discrete time point $t_n$.

## 8.4. Solution of the linear systems

The resulting linear systems in each time interval $[t_{n-1}, t_n]$, which are $6 \times 6$ block systems in the case of the cGP(2) and dG(1) approach and $3 \times 3$ block systems for the cGP(1)-method, are treated by using a geometrical multigrid solver with a LPSC type smoother (described in Chapter 4). The multigrid solver uses the standard refinement scheme (see [58] for the grid hierarchies) and the canonical grid transfer operators described in Chapter 4 regarding the chosen FEM space which treat both solution components separately in the case of the cGP(2) and dG(1) approaches (see [39] for the details). Finally, the coarse grid problem is solved by the Gaussian elimination method.

## 8.5. Numerical results

In this section, we perform several numerical tests in order to compare the accuracy of the proposed time discretization schemes. As a test problem we consider the Stokes problem (8.1) with the domain $\Omega := (0,1)^2$ and $\nu = 1$. The prescribed velocity field $\mathbf{u} = (u_1, u_2)$ is

$$\begin{aligned} u_1(x,y,t) &:= x^2(1-x)^2\left[2y(1-y)^2 - 2y^2(1-y)\right]\sin(10\pi t), \\ u_2(x,y,t) &:= -\left[2x(1-x)^2 - 2x^2(1-x)\right]y^2(1-y)^2\sin(10\pi t), \end{aligned}$$

and the pressure distribution $p(x,y,t) := -(x^3+y^3-0.5)(1.5+0.5\sin(10\pi t))$. The initial data is $\mathbf{u}_0(x,y) = \mathbf{u}(x,y,0)$.

We apply the time discretization schemes cGP(1), cGP(2) and dG(1) with an equidistant time step size $\tau = T/N$. To measure the error (in time), the following discrete $L^\infty$-norm of a function $v : I \to L^2(\Omega)$ is used

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

The behavior of the standard $L^2$-norm $\|\cdot\|_2 := \|\cdot\|_{L^2(I,L^2(\Omega))}$ and the discrete $L^\infty$-norm of the time discretization error $u(t) - u_{h,\tau}(t)$ for the velocity over the time interval $I = [0,1]$ can be seen in Table 8.1 and 8.2, respectively. The estimated value of the experimental order of convergence (EOC) is also calculated and compared with the theoretical order of convergence for both velocity and pressure. All our numerical tests, where we compared the accuracy of our time discretization schemes are related to space mesh level 7 with equidistant $h = 2^{-6}$.

| | cGP(1) | | cGP(2) | | dG(1) | |
|---|---|---|---|---|---|---|
| $1/\tau$ | $\|u-u_{h,\tau}\|_2$ | EOC | $\|u-u_{h,\tau}\|_2$ | EOC | $\|u-u_{h,\tau}\|_2$ | EOC |
| 10 | 5.56E-03 | | 4.40E-04 | | 2.91E-03 | |
| 20 | 1.53E-03 | 1.86 | 1.11E-04 | 1.99 | 6.51E-04 | 2.16 |
| 40 | 3.93E-04 | 1.97 | 1.33E-05 | 3.06 | 1.82E-04 | 1.84 |
| 80 | 9.87E-05 | 1.99 | 1.62E-06 | 3.04 | 4.83E-05 | 1.91 |
| 160 | 2.47E-05 | 2.00 | 2.03E-07 | 3.00 | 1.25E-05 | 1.95 |
| 320 | 6.18E-06 | 2.00 | | | 3.17E-06 | 1.98 |
| 640 | 1.55E-06 | 2.00 | | | 8.00E-07 | 1.99 |
| 1280 | 3.88E-07 | 2.00 | | | 2.03E-07 | 1.98 |
| 2560 | 1.01E-07 | 1.94 | | | | |

**Table 8.1**: Error norms $\|u-u_{h,\tau}\|_2$ for velocity.

| | cGP(1) | | cGP(2) | | dG(1) | |
|---|---|---|---|---|---|---|
| $1/\tau$ | $\|u-u_{h,\tau}\|_\infty$ | EOC | $\|u-u_{h,\tau}\|_\infty$ | EOC | $\|u-u_{h,\tau}\|_\infty$ | EOC |
| 10 | 2.38E-15 | | 6.74E-04 | | 2.18E-03 | |
| 20 | 8.17E-04 | -38.32 | 1.38E-04 | 2.29 | 3.73E-04 | 2.55 |
| 40 | 2.10E-04 | 1.96 | 1.03E-05 | 3.75 | 5.98E-05 | 2.64 |
| 80 | 5.13E-05 | 2.03 | 6.88E-07 | 3.90 | 8.86E-06 | 2.75 |
| 160 | 1.28E-05 | 2.00 | 4.75E-08 | 3.86 | 1.19E-06 | 2.90 |
| 320 | 3.20E-06 | 2.00 | | | 1.60E-07 | 2.90 |
| 640 | 8.01E-07 | 2.00 | | | | |
| 1280 | 2.01E-07 | 1.99 | | | | |
| 2560 | 5.72E-08 | 1.82 | | | | |

**Table 8.2**: Error norms $\|u-u_{h,\tau}\|_\infty$ for velocity.

We see that the cGP(2)-method is of order 3 in the $L^2$-norm and superconvergent of order 4 at the discrete time points $t_n$, while the dG(1)-method is of order 2 in the $L^2$-norm and superconvergent of order 3 at the end points of the time intervals as expected from the theory. The cGP(1)-method is of order 2 everywhere which is the same behavior as that of the well-known Crank-Nicolson scheme.

Now we show the accuracy of our time discretization schemes for the pressure. Here, we also illustrate the behavior of the $L^2$-norm $\|\cdot\|_2 := \|\cdot\|_{L^2(I,L^2(\Omega))}$ and the discrete $L^\infty$-norm of the error in the pressure, respectively. From Table 8.3, we observe that the experimental orders of convergence (EOC) coincide with the theoretical orders of convergence for corresponding time

| | cGP(1) | | cGP(2) | | dG(1) | |
|---|---|---|---|---|---|---|
| $1/\tau$ | $\|p-p_{h,\tau}\|_2$ | EOC | $\|p-p_{h,\tau}\|_2$ | EOC | $\|p-p_{h,\tau}\|_2$ | EOC |
| 10 | 2.10E-01 | | 7.89E-04 | | 2.60E-03 | |
| 20 | 4.25E-02 | 2.30 | 3.03E-04 | 1.38 | 8.31E-04 | 1.64 |
| 40 | 1.08E-02 | 1.97 | 4.18E-05 | 2.86 | 2.53E-04 | 1.72 |
| 80 | 2.73E-03 | 1.99 | 5.31E-06 | 2.98 | 7.13E-05 | 1.82 |
| 160 | 6.83E-04 | 2.00 | 6.87E-07 | 2.95 | 1.88E-05 | 1.93 |
| 320 | 1.71E-04 | 2.00 | 1.91E-07 | 1.85 | 4.86E-06 | 1.95 |
| 640 | 4.39E-05 | 1.96 | | | 1.25E-06 | 1.96 |
| 1280 | 1.10E-05 | 2.00 | | | 3.57E-07 | 1.81 |
| 2560 | 2.75E-06 | 2.00 | | | 1.37E-07 | 1.38 |
| 5120 | 6.86E-07 | 2.00 | | | | |
| 10240 | 1.74E-07 | 1.98 | | | | |

**Table 8.3**: Error norms $\|p-p_{h,\tau}\|_2$ at the Gaussian points for pressure.

discretization schemes. Next, we want to analyze the behavior of $L^\infty$-error for the pressure. As we have already discussed, one can achieve the high order pressure at the discrete time points $t_n$ by using the Lagrangian interpolation polynomials symmetric at $t_n$. The behavior of the discrete $L^\infty$-norm of the error for pressure can be seen in Table 8.4.

| | cGP(1) | | cGP(2) | | dG(1) | |
|---|---|---|---|---|---|---|
| $1/\tau$ | $\|p-p_{h,\tau}\|_\infty$ | EOC | $\|p-p_{h,\tau}\|_\infty$ | EOC | $\|p-p_{h,\tau}\|_\infty$ | EOC |
| 10 | 9.97E-06 | | 7.55E-04 | | 2.25E-03 | |
| 20 | 1.00E-01 | -13.30 | 1.35E-03 | -0.84 | 1.36E-03 | 0.73 |
| 40 | 2.94E-02 | 1.77 | 8.86E-05 | 3.93 | 1.14E-04 | 3.57 |
| 80 | 7.63E-03 | 1.94 | 5.60E-06 | 3.98 | 1.91E-05 | 2.58 |
| 160 | 1.93E-03 | 1.99 | 4.13E-07 | 3.76 | 2.43E-06 | 2.98 |
| 320 | 4.83E-04 | 2.00 | | | 3.84E-07 | 2.66 |
| 640 | 1.21E-04 | 2.00 | | | | |
| 1280 | 3.02E-05 | 2.00 | | | | |
| 2560 | 7.55E-06 | 2.00 | | | | |
| 5120 | 1.90E-06 | 1.99 | | | | |
| 10240 | 4.74E-07 | 2.00 | | | | |

**Table 8.4**: Error norms $\|p-p_{h,\tau}\|_\infty$ for the pressure using Lagrange interpolation.

We observe that the cGP(2)-method has superconvergent results of order 4 for pressure at the discrete time points $t_n$, while both the cGP(1) and dG(1)-method are of order 2 and 3, respectively, at the end points of the time intervals as expected.

Since the error norms we compared so far contain both the spatial and time error, after a certain stage the space error becomes dominant. To see the accuracy for the time error more clearly, we now compute the norm $\|u_h-u_{h,\tau}\|_\infty \sim \|u_{h,\tau^*}-u_{h,\tau}\|_\infty$ by considering the reference time step size $\tau^* = 1/2560$ for velocity and pressure for the cGP(2) and dG(1)-method.

One can see form Table 8.5 that the experimental orders of convergence for the cGP(2) and dG(1)-methods are much more visible in the absence of spatial discretization errors.

Next, we perform numerical tests to analyze the corresponding behavior of the multigrid solver for the different time discretization schemes. As explained before, the solver uses a cell oriented LPSC type smoother and applies four pre- and post-smoothing steps. We present the averaged number of multigrid iterations per time step for solving the corresponding systems in Table 8.6. 'Lev' denotes the refinement level of the space mesh.

From Table 8.6, we see that the multigrid solver requires almost the same number of iterations for the different presented time discretization schemes. Moreover, the number of multigrid

| | cGP(2) | | | | dG(1) | | | |
|---|---|---|---|---|---|---|---|---|
| $1/\tau$ | $\|u_h - u_{h,\tau}\|_\infty$ | EOC | $\|p_h - p_{h,\tau}\|_\infty$ | EOC | $\|u_h - u_{h,\tau}\|_\infty$ | EOC | $\|p_h - p_{h,\tau}\|_\infty$ | EOC |
| 10 | 6.74E-04 | | 7.55E-04 | | 2.18E-03 | | 2.25E-03 | |
| 20 | 1.38E-04 | 2.29 | 1.35E-03 | -0.84 | 3.73E-04 | 2.55 | 1.36E-03 | 0.73 |
| 40 | 1.03E-05 | 3.75 | 8.86E-05 | 3.93 | 5.98E-05 | 2.64 | 1.14E-04 | 3.57 |
| 80 | 6.88E-07 | 3.90 | 5.60E-06 | 3.98 | 8.86E-06 | 2.75 | 1.50E-05 | 2.93 |
| 160 | 4.39E-08 | 3.97 | 3.51E-07 | 4.00 | 1.19E-06 | 2.90 | 2.42E-06 | 2.63 |
| 320 | 2.75E-09 | 4.00 | 2.19E-08 | 4.00 | 1.55E-07 | 2.94 | 3.46E-07 | 2.80 |
| 640 | 1.71E-10 | 4.01 | 1.37E-09 | 4.01 | 1.96E-08 | 2.98 | 4.57E-08 | 2.92 |

**Table 8.5**: Temporal errors for velocity and pressure.

| Lev | $\tau = 1/20$ | $\tau = 1/80$ | $\tau = 1/320$ | $\tau = 1/1280$ |
|---|---|---|---|---|
| 3 | 6-7-7 | 8-9-8 | 9-10-10 | 10-11-10 |
| 4 | 9-8-9 | 8-8-8 | 8-10-9 | 10-11-7 |
| 5 | 9-9-9 | 8-8-8 | 8-9-8 | 9-10-9 |
| 6 | 10-10-9 | 10-10-8 | 8-8-8 | 7-8-8 |
| 7 | 10-10-9 | 10-10-10 | 9-9-10 | 8-8-8 |

**Table 8.6**: Averaged multigrid iterations per time step for cGP(1) - cGP(2) - dG(1).

iterations remains fairly constant if we increase the refinement level of the space mesh. There is also no noticeable increase in the number of iterations if we decrease the time step (due to the non-diagonal mass matrix of $Q_2$). This means that the behavior of the multigrid solver is almost independent of the spatial mesh size and the time step.

Next, in order to measure and compare the efficiency of the multigrid solver for our time discretizations, we present in Table 8.7 the averaged CPU-time required for one solver iteration on a given space mesh level.

| | Lev=5 | | | Lev=6 | | | Lev=7 | | |
|---|---|---|---|---|---|---|---|---|---|
| $1/\tau$ | cGP(1) | cGP(2) | dG(1) | cGP(1) | cGP(2) | dG(1) | cGP(1) | cGP(2) | dG(1) |
| 10 | 0.10 | 0.33 | 0.35 | 0.43 | 1.40 | 1.39 | 1.86 | 5.98 | 5.86 |
| 20 | 0.10 | 0.33 | 0.35 | 0.43 | 1.41 | 1.40 | 1.83 | 5.88 | 5.84 |
| 40 | 0.11 | 0.33 | 0.33 | 0.47 | 1.40 | 1.43 | 2.03 | 6.02 | 6.14 |
| 80 | 0.10 | 0.34 | 0.35 | 0.44 | 1.41 | 1.40 | 2.03 | 6.12 | 6.15 |
| 160 | 0.10 | 0.33 | 0.36 | 0.43 | 1.44 | 1.41 | 1.95 | 6.19 | 6.11 |
| 320 | 0.10 | 0.33 | 0.36 | 0.53 | 1.45 | 1.55 | 1.94 | 6.23 | 5.88 |
| 640 | 0.11 | 0.34 | 0.34 | 0.53 | 1.40 | 1.40 | 2.11 | 6.22 | 6.09 |
| 1280 | 0.10 | 0.36 | 0.36 | 0.46 | 1.48 | 1.40 | 1.91 | 6.23 | 6.27 |

**Table 8.7**: CPU-time per solver iteration for space mesh level=5,6,7, respectively.

In Table 8.7, we observe that the CPU-time in case of cGP(2) or dG(1) is almost 3 times the CPU-time of cGP(1) for the multigrid solver. We also note that the CPU-time grows approximately by a factor of 4 as expected if we increase the space mesh level. These factors are nearly optimal since the number of space unknowns is increased by a factor of 4 if the level is increased by one.

Finally, we compare the time discretization schemes with respect to accuracy and numerical costs. Here, the multigrid solver uses four LPSC type iterations in the pre- and post-smoothing step. The space discretization was done on mesh level 7. Table 8.8 shows, for different sizes of the time step $\tau$ and different time discretization schemes, the discrete $L^\infty$-norm and the total CPU-time for the computation in all time intervals. Due to its superconvergence of order 3 in the discrete time points, the dG(1)-method is faster than cGP(1) which is only of order 2. One can see that, in order to achieve the accuracy of $10^{-7}$, we need the very small time step $\tau = 1/2560$ for the cGP(1)

| | cGP(1) | | cGP(2) | | dG(1) | |
|---|---|---|---|---|---|---|
| $1/\tau$ | $\|u-u_{h,\tau}\|_\infty$ | CPU | $\|u-u_{h,\tau}\|_\infty$ | CPU | $\|u-u_{h,\tau}\|_\infty$ | CPU |
| 10 | 2.38E-15 | 40 | 6.74E-04 | 183 | 2.18E-03 | 189 |
| 20 | 8.17E-04 | 80 | 1.38E-04 | 364 | 3.73E-04 | 365 |
| 40 | 2.10E-04 | 160 | 1.03E-05 | 713 | 5.98E-05 | 764 |
| 80 | 5.13E-05 | 342 | 6.88E-07 | 1401 | 8.86E-06 | 1445 |
| 160 | 1.28E-05 | 645 | 4.75E-08 | 2866 | 1.19E-06 | 2884 |
| 320 | 3.20E-06 | 1337 | | | 1.60E-07 | 6145 |
| 640 | 8.01E-07 | 2833 | | | | |
| 1280 | 2.01E-07 | 5564 | | | | |
| 2560 | 5.72E-08 | 13441 | | | | |

**Table 8.8**: Error norms $\|u-u_{h,\tau}\|_\infty$ and total CPU-time to achieve the accuracy of $10^{-7}$ for velocity field.

while this accuracy can be already achieved with $\tau = 1/160$ and $\tau = 1/320$ in cGP(2) and dG(1)-schemes, respectively. To compare the numerical costs per time step let us note that the number of multigrid iterations to solve one linear block system is approximately the same (about 8) for the three time discretization schemes. However, the costs of one multigrid iteration in the cGP(2) or dG(1) method is almost 3 times higher than in cGP(1). Nevertheless, for a desired accuracy of $10^{-7}$, the cGP(2) scheme is about 5 times faster than cGP(1) due to the much larger time step size required for cGP(2).

Next, we also compare our presented time discretization schemes with respect to accuracy and numerical costs for the pressure. To this end, we will only compare the accuracy measured in the discrete $L^\infty$-norm. Here, the pressure is obtained at the discrete time points $t_n$ by using the Lagrangian interpolation procedure. From Table 8.9, it can be seen that to achieve the accuracy

| | cGP(1) | | cGP(2) | | dG(1) | |
|---|---|---|---|---|---|---|
| $1/\tau$ | $\|p-p_{h,\tau}\|_\infty$ | CPU | $\|p-p_{h,\tau}\|_\infty$ | CPU | $\|p-p_{h,\tau}\|_\infty$ | CPU |
| 10 | 9.97E-06 | 40 | 7.55E-04 | 183 | 2.25E-03 | 189 |
| 20 | 1.00E-01 | 80 | 1.35E-03 | 364 | 1.36E-03 | 365 |
| 40 | 2.94E-02 | 160 | 8.86E-05 | 713 | 1.14E-04 | 764 |
| 80 | 7.63E-03 | 342 | 5.60E-06 | 1401 | 1.91E-05 | 1445 |
| 160 | 1.93E-03 | 645 | 4.13E-07 | 2866 | 2.43E-06 | 2884 |
| 320 | 4.83E-04 | 1337 | | | 3.84E-07 | 6145 |
| 640 | 1.21E-04 | 2833 | | | | |
| 1280 | 3.02E-05 | 5564 | | | | |
| 2560 | 7.55E-06 | 13441 | | | | |
| 5120 | 1.90E-06 | 27427 | | | | |
| 10240 | 4.74E-07 | 53692 | | | | |

**Table 8.9**: Error norms $\|p-p_{h,\tau}\|_\infty$ and total CPU-time to achieve the accuracy of $10^{-7}$ for the pressure using interpolation.

of $10^{-7}$, the cGP(1) and dG(1)-methods need very small time step sizes, i.e., $\tau = 1/10240$ and $\tau = 1/320$, while this accuracy has been already achieved with $\tau = 1/160$ for the cGP(2) scheme. Hence, the cGP(2)-method always gives the accurate results for velocity and pressure in a much more efficient way.

At the end, to show that the proposed time discretization schemes can also efficiently handle the case when the solution approaches a steady state, we provide numerical tests with very large time steps. We consider problem (8.1) for $\Omega = (0,1)^2$ and the prescribed (time-independent)

velocity

$$\begin{aligned} u_1(x,y,t) &:= x^2(1-x)^2\left[2y(1-y)^2-2y^2(1-y)\right], \\ u_2(x,y,t) &:= -\left[2x(1-x)^2-2x^2(1-x)\right]y^2(1-y)^2, \end{aligned}$$

and the pressure distribution $p(x,y,t) := -(x^3+y^3-0.5)$. For these analytical solutions for $\mathbf{u}$ and $p$, we compute the corresponding right hand sides. As initial data we take $\mathbf{u_0} = 0$. Table 8.10

| Lev | $\tau = 10^{-6}$ | $\tau = 10^{-3}$ | $\tau = 1$ | $\tau = 10^{3}$ | $\tau = 10^{6}$ |
|---|---|---|---|---|---|
| 3 | 5-5-5 | 5-5-5 | 4-4-4 | 4-4-4 | 4-4-4 |
| 4 | 7-7-7 | 5-6-6 | 6-6-6 | 6-6-6 | 6-6-6 |
| 5 | 8-8-8 | 6-7-7 | 6-7-7 | 7-7-7 | 7-7-7 |
| 6 | 8-8-8 | 6-7-7 | 9-9-9 | 9-9-9 | 9-9-9 |
| 7 | 8-8-8 | 7-7-7 | 9-9-9 | 9-9-9 | 9-9-9 |
| 8 | 8-8-8 | 8-8-8 | 9-9-9 | 9-9-9 | 9-9-9 |

**Table 8.10**: Averaged multigrid iterations per time step for cGP(1) - cGP(2) - dG(1).

indicates for the multigrid method the number of solver iterations required for one time step which shows that there is no big difference in the number of solver iterations for time step size $\tau = 10^{-6}$ up to $\tau = 10^{6}$. This means that the behavior of the multigrid convergence is pretty robust with respect to very small as well as very large time steps.

## 8.6. Summary

We have extended our work from heat equation in Chapter 5 and described in detail the application of the *continuous Galerkin-Petrov* and *discontinuous Galerkin* time discretization schemes to the nonstationary Stokes equations.

### Accuracy

From the numerical studies in this chapter, we observed that the estimated experimental orders of convergence confirm the expected theoretical orders of all the presented time discretization schemes. Furthermore, the numerical tests have shown that the cGP(2)-scheme provides significantly more accurate numerical solutions for both velocity and pressure than the other presented schemes cGP(1) and dG(1) which means that quite large time step sizes are allowed to gain highly accurate results.

Since we obtain superconvergence results for the velocity only at the discrete time points $t_n$, it is also desirable to get a high order pressure at the same points. In order to get a higher order pressure, we have performed a special post processing which gives the same order of approximation both for, the velocity and pressure in the discrete time point $t_n$.

### Efficiency

In order to measure and compare the efficiency of the multigrid solver, we have performed some numerical tests for the different time discretization schemes. To this end, we presented the averaged number of multigrid iterations per time step for solving the corresponding linear systems. We have analyzed that the multigrid solver requires almost the same number of iterations for the different presented time discretization schemes. Moreover, the number of multigrid iterations remains almost constant if we increase the refinement level of the space mesh. There is also no noticeable increase in the number of iterations by decreasing the time step size. This means that the behavior of the multigrid solver is almost independent of the spatial mesh size and the time step. Furthermore, the CPU-time in case of the cGP(2) or dG(1) is almost 3 times the CPU-time

of cGP(1) for the multigrid solver and the CPU-time grows approximately by a factor of 4 as expected if we increase the space mesh level. Thus, the multigrid-solver turned out to be of optimal computational complexity.

# 9

# Galerkin time discretizations for the Navier-Stokes equations

In this chapter, we extend our work for the heat equation in Chapter 5 and for the Stokes equations in Chapter 8 to the nonstationary Navier-Stokes equations in two dimensions. We present fully implicit *continuous* Galerkin-Petrov (cGP) and *discontinuous* Galerkin (dG) time stepping schemes for incompressible Navier-Stokes equations which are, in contrast to standard approaches like for instance the Crank-Nicolson scheme, of higher order in time. In particular, we implement and analyze numerically the higher order dG(1) and cGP(2)-methods which are super-convergent of 3rd, resp., 4th order in time, while for the space discretization, the well-known LBB-stable finite element pair $Q_2/P_1^{disc}$ (explained in Chapter 3) is used. The resulting discretized systems of nonlinear equations are also presented for the cGP(1), cGP(2) and dG(1) method which will be used in our numerical experiments. The solution techniques for the discretized nonlinear systems are also explained in this chapter.

## 9.1. The cGP- and dG-methods for the Navier-Stokes equation

For a domain $\Omega \subset \mathbb{R}^d$, we consider the nonstationary incompressible Navier-Stokes equations, i.e., we want to find for each time $t \in [0,T]$ a velocity field $\mathbf{u}(t) : \Omega \to \mathbb{R}^d$ and a pressure field $p(t) : \Omega \to \mathbb{R}$ *such that*

$$\begin{aligned} \partial_t \mathbf{u} - \nu \triangle \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p &= f \quad \text{in} \quad \Omega \times (0,T], \\ \operatorname{div} \mathbf{u} &= 0 \quad \text{in} \quad \Omega \times (0,T], \\ \mathbf{u} &= g \quad \text{on} \quad \partial\Omega \times (0,T], \\ \mathbf{u}(x,0) &= \mathbf{u}_0(x) \quad \text{in} \quad \Omega \quad \text{for} \quad t = 0, \end{aligned} \tag{9.1}$$

where $\nu$ denotes the viscosity, $f$ the body force and $\mathbf{u}_0$ the initial velocity field at time $t = 0$. For simplicity, we restrict to the case $d = 2$ and we assume homogeneous Dirichlet conditions at the boundary $\partial\Omega$ of a polygonal domain $\Omega$ (for other choices see [24]). To make this problem well-posed in the case of pure Dirichlet boundary conditions, we have to look for the field $p(t)$ at each time $t$ in the subspace $L_0^2(\Omega) \subset L^2(\Omega)$ of functions with zero integral mean value.

For the time discretization, we decompose the time interval $I = (0,T]$ into $N$ disjoint subintervals $I_n := (t_{n-1}, t_n]$, where $n = 1, \ldots, N$ and $0 = t_0 < t_1 < \cdots < t_{N-1} < t_N = T$. Thus, the value of the time-discrete approximation $\mathbf{u}_\tau$ at time $t_n$ is always defined as the $I_n$-value (i.e., the left-sided value in case of discontinuous approximation) $\mathbf{u}_\tau(t_n) := \mathbf{u}^- := \mathbf{u}_\tau|_{I_n}(t_n)$. The symbol $\tau$ denotes the *time discretization parameter* and is also used as the maximum time step size $\tau := \max_{1 \le n \le N} \tau_n$, where $\tau_n := t_n - t_{n-1}$. Then, for the subsequent continuous and discontinuous Galerkin time stepping schemes, we approximate the solution $\mathbf{u}$ by means of a function $\mathbf{u}_\tau$ which is piecewise polynomial with respect to time. In case of the cGP($k$)-method, we are looking for $\mathbf{u}_\tau$ in the discrete

time-continuous space (with $\mathbf{V} = (H_0^1(\Omega))^2$)

$$\mathbf{X}_\tau^k := \{\mathbf{u} \in C(I,\mathbf{V}) : \mathbf{u}\big|_{I_n} \in \mathbb{P}_k(I_n,\mathbf{V}) \quad \forall\, n = 1,\ldots,N\}, \tag{9.2}$$

where

$$\mathbb{P}_k(I_n,\mathbf{V}) := \Big\{\mathbf{u} : I_n \to \mathbf{V} : \mathbf{u}(t) = \sum_{j=0}^{k} \mathbf{U}^j t^j,\ \forall t \in I_n,\ \mathbf{U}^j \in \mathbf{V},\ \forall\, j\Big\}. \tag{9.3}$$

Moreover, we introduce the discrete time-discontinuous test space

$$\mathbf{Y}_\tau^{k-1} := \{\mathbf{v} \in L^2(I,\mathbf{V}) : \mathbf{v}\big|_{I_n} \in \mathbb{P}_{k-1}(I_n,\mathbf{V}) \quad \forall\, n = 1,\ldots,N\} \tag{9.4}$$

consisting of piecewise polynomials of order $k-1$ which are (globally) discontinuous at the end points of the time intervals. Similarly, we will use for the time-discrete pressure $p_\tau$ an analogous ansatz space $\tilde{X}_\tau^k$, where the vector valued space $\mathbf{V}$ is replaced by the scalar valued space $Q = L_0^2(\Omega)$, and an analogous discontinuous test space $\tilde{Y}_\tau^{k-1}$.

In case of the dG($k-1$)-method, we are looking for $\mathbf{u}_\tau$ in the time-discontinuous discrete space $\mathbf{Y}_\tau^{k-1}$. Next, we describe separately the cGP($k$) and dG($k-1$)-method.

In order to derive the time discretization, we multiply the equations in (9.1) with some suitable $I_n$-supported test functions and integrate over $\Omega \times I_n$. To determine $\mathbf{u}_\tau|_{I_n}$ and $p_\tau|_{I_n}$ we represent them by the polynomial ansatz

$$\mathbf{u}_\tau|_{I_n}(t) := \sum_{j=0}^{k} \mathbf{U}_n^j \phi_{n,j}(t), \qquad p_\tau|_{I_n}(t) := \sum_{j=0}^{k} P_n^j \phi_{n,j}(t), \tag{9.5}$$

where the "coefficients" $(\mathbf{U}_n^j, P_n^j)$ are elements of the function spaces $\mathbf{V} \times Q$ and the polynomial functions $\phi_{n,j} \in \mathbb{P}_k(I_n)$ are the Lagrange basis functions with respect to the $k+1$ nodal points $t_{n,j} \in I_n$ satisfying the conditions

$$\phi_{n,j}(t_{n,i}) = \delta_{i,j}, \qquad i,j = 0,\ldots,k \tag{9.6}$$

with the Kronecker symbol $\delta_{i,j}$. For an easy treatment of the initial condition, we set $t_{n,0} = t_{n-1}$. Then, the initial condition is equivalent to the condition

$$\mathbf{U}_n^0 = \mathbf{u}_\tau|_{I_{n-1}}(t_{n-1}) \quad \text{if} \quad n \geq 2 \qquad \text{or} \qquad \mathbf{U}_n^0 = \mathbf{u}_0 \quad \text{if} \quad n = 1. \tag{9.7}$$

The other points $t_{n,1},\ldots,t_{n,k}$ are chosen as the quadrature points of the $k$-point Gaussian formula on $I_n$ which is exact if the function to be integrated is a polynomial of degree less or equal to $2k-1$. We define the basis functions $\phi_{n,j} \in \mathbb{P}_k(I_n)$ of (9.5) via affine reference transformations (see Chapter 5 and 8 for more details). Now, we can describe the *time discrete $I_n$-problem of the cGP(k)-method* (see Chapter 5 and 8,[49]):

Find on the interval $I_n = (t_{n-1}, t_n]$ the $k$ unknown pairs of "coefficients" $(\mathbf{U}_n^j, P_n^j) \in \mathbf{V} \times Q$, $j = 1,\ldots,k$, such that for all $i = 1,\ldots,k$, it holds for all $(\mathbf{v},q) \in \mathbf{V} \times Q$

$$\begin{aligned} \sum_{j=0}^{k} \alpha_{i,j} \left(\mathbf{U}_n^j, \mathbf{v}\right)_\Omega + \frac{\tau_n}{2} a(\mathbf{U}_n^i, \mathbf{v}) + \frac{\tau_n}{2} n(\mathbf{U}_n^i, \mathbf{U}_n^i, \mathbf{v}) + \frac{\tau_n}{2} b(\mathbf{v}, P_n^i) &= \tfrac{\tau_n}{2} \left(f(t_{n,i}), \mathbf{v}\right)_\Omega \\ b(\mathbf{U}_n^i, q) &= 0 \end{aligned} \tag{9.8}$$

with $\mathbf{U}_n^0 := \mathbf{u}_\tau(t_{n-1})$ for $n > 1$, $\mathbf{U}_1^0 := \mathbf{u}_0$ and $(\cdot,\cdot)_\Omega$ denotes the usual inner product in $(L^2(\Omega))^d$. The bilinear forms $a(\cdot,\cdot)$ and $b(\cdot,\cdot)$ on $\mathbf{V} \times \mathbf{V}$ and $\mathbf{V} \times Q$, respectively, are defined as

$$a(\mathbf{u},\mathbf{v}) := \int_\Omega \nabla \mathbf{u} \cdot \nabla \mathbf{v}\, dx \quad \forall\, \mathbf{u},\mathbf{v} \in \mathbf{V}, \qquad b(\mathbf{v},p) := -\int_\Omega \nabla \cdot \mathbf{v}\, p\, dx, \tag{9.9}$$

and the trilinear form $n(\cdot,\cdot,\cdot)$ on $\mathbf{V}\times\mathbf{V}\times\mathbf{V}$ is given as $n(\mathbf{w},\mathbf{u},\mathbf{v}) := \sum_{i=1}^{d} n_i(\mathbf{w},u_i,v_i)$ where

$$n_i(\mathbf{w},u_i,v_i) := \int_\Omega (\mathbf{w}\cdot\nabla u_i)v_i\,dx \quad \forall\,\mathbf{w}\in\mathbf{V}, \quad u_i,v_i\in H_0^1(\Omega). \tag{9.10}$$

A typical property of this cGP($k$)-variant is that the initial pressure $P_n^0$ of the ansatz (9.5) does not occur in this formulation. In order to achieve superconvergence for the pressure approximation at the discrete time levels $t_n$ special interpolation techniques using two neighboured time intervals can be applied (see Chapter 8).

In the following subsections, we specify the constants $\alpha_{i,j}$ of the cGP($k$)-method for the cases $k=1$ and $k=2$, and for comparison we describe explicitly the well-known dG(1) approach (see Chapter 8 for more details).

### 9.1.1. cGP(1)-method.

We use the one-point Gaussian quadrature formula with $\hat{t}_1 = 0$ and $t_{n,1} = t_{n-1} + \frac{\tau_n}{2}$. Then, we get $\alpha_{1,0} = -1$ and $\alpha_{1,1} = 1$ (see Chapter 5 and 8). Thus, problem (9.8) leads to the following problem for the "one" pair of unknowns $\mathbf{U}_n^1 = \mathbf{u}_\tau(t_{n-1}+\frac{\tau_n}{2})$ and $P_n^1 = p_\tau(t_{n-1}+\frac{\tau_n}{2})$: Find $(\mathbf{U}_n^1,P_n^1)\in\mathbf{V}\times Q$ such that for all $(\mathbf{v},q)\in\mathbf{V}\times Q$ it holds

$$\begin{aligned} \big(\mathbf{U}_n^1,\mathbf{v}\big)_\Omega + \tfrac{\tau_n}{2}a(\mathbf{U}_n^1,\mathbf{v}) + \tfrac{\tau_n}{2}n(\mathbf{U}_n^1,\mathbf{U}_n^1,\mathbf{v}) + \tfrac{\tau_n}{2}b(\mathbf{v},P_n^1) &= \tfrac{\tau_n}{2}\,(f(t_{n,1}),\mathbf{v})_\Omega + \big(\mathbf{U}_n^0,\mathbf{v}\big)_\Omega \\ b(\mathbf{U}_n^1,q) &= 0. \end{aligned} \tag{9.11}$$

Once we have determined the solution $\mathbf{U}_n^1$ at the midpoint $t_{n,1}$ of the time interval $I_n$, we get the solution at the next discrete time point $t_n$ simply by linear extrapolation based on the ansatz (9.5), i.e.,

$$\mathbf{u}_\tau(t_n) = 2\mathbf{U}_n^1 - \mathbf{U}_n^0, \tag{9.12}$$

where $\mathbf{U}_n^0$ is the initial value at the time interval $(t_{n-1},t_n]$ coming from the previous time interval $I_{n-1}$ or the initial value $\mathbf{u}_0$.

If we would replace $f(t_{n,1})$ by the mean value $(f(t_{n-1})+f(t_n))/2$, which means that we replace the one-point Gaussian quadrature of the right hand side by the trapezoidal rule, the resulting cGP(1)-method is equivalent to the well-known *Crank-Nicolson scheme*. The cGP(1)-method is accurate of order 2 in the whole time interval as it is known for the Crank-Nicolson scheme. Concerning the pressure approximation, one observes that the second order accuracy holds only in the midpoints of the time intervals. By means of linear interpolation between the midpoints of two neighbouring time intervals we get second order accuracy also at the discrete time levels $t_n$.

### 9.1.2. cGP(2)-method.

Here, we use the 2-point Gaussian quadrature formula with the points $\hat{t}_1 = -\frac{1}{\sqrt{3}}$ and $\hat{t}_2 = \frac{1}{\sqrt{3}}$. Then, we obtain the coefficients

$$(\alpha_{i,j}) = \begin{pmatrix} -\sqrt{3} & \frac{3}{2} & \frac{2\sqrt{3}-3}{2} \\ \sqrt{3} & \frac{-2\sqrt{3}-3}{2} & \frac{3}{2} \end{pmatrix} \qquad i=1,2,\ j=0,1,2. \tag{9.13}$$

Consequently, on the time interval $I_n$, we have to solve for the two "unknowns"

$$(\mathbf{U}_n^j,P_n^j) = \big(\mathbf{u}_\tau(t_{n,j}),p_\tau(t_{n,j})\big)\in\mathbf{V}\times Q \qquad \text{with} \qquad t_{n,j} := (t_{n-1}+t_n+\tau_n\hat{t}_j)/2 \quad \text{for} \quad j=1,2. \tag{9.14}$$

The corresponding coupled system reads

$$\begin{aligned} \alpha_{1,1}\left(\mathbf{U}_n^1,\mathbf{v}\right)_\Omega + \tfrac{\tau_n}{2}a(\mathbf{U}_n^1,\mathbf{v}) + \tfrac{\tau_n}{2}n(\mathbf{U}_n^1,\mathbf{U}_n^1,\mathbf{v}) + \alpha_{1,2}\left(\mathbf{U}_n^2,\mathbf{v}\right)_\Omega + \tfrac{\tau_n}{2}b(\mathbf{v},P_n^1) &= \ell_1(\mathbf{v}) \\ \alpha_{2,1}\left(\mathbf{U}_n^1,\mathbf{v}\right)_\Omega + \alpha_{2,2}\left(\mathbf{U}_n^2,\mathbf{v}\right)_\Omega + \tfrac{\tau_n}{2}a(\mathbf{U}_n^2,\mathbf{v}) + \tfrac{\tau_n}{2}n(\mathbf{U}_n^2,\mathbf{U}_n^2,\mathbf{v}) + \tfrac{\tau_n}{2}b(\mathbf{v},P_n^2) &= \ell_2(\mathbf{v}) \\ b(\mathbf{U}_n^1,q) &= 0 \\ b(\mathbf{U}_n^2,q) &= 0, \end{aligned} \tag{9.15}$$

which has to be satisfied for all $(\mathbf{v},q)\in\mathbf{V}\times Q$ with

$$\ell_i(\mathbf{v}) = \frac{\tau_n}{2}\left(f(t_{n,i}),\mathbf{v}\right)_\Omega - \alpha_{i,0}\left(\mathbf{U}_n^0,\mathbf{v}\right)_\Omega \qquad i=1,2. \tag{9.16}$$

Once we have determined the solutions $\mathbf{U}_n^1,\mathbf{U}_n^2$ at the Gaussian points in the interior of the interval $I_n$, we get the solution at the right boundary $t_n$ of $I_n$ by means of quadratic extrapolation from the ansatz (9.5), i.e.,

$$\mathbf{u}_\tau(t_n) = \mathbf{U}_n^0 + \sqrt{3}(\mathbf{U}_n^2 - \mathbf{U}_n^1), \tag{9.17}$$

where $\mathbf{U}_n^0$ is the initial value at the time interval $I_n$. The cGP(2)-method is accurate of order 3 in the whole time interval and superconvergent of order 4 in the discrete time points (see Chapter 5 and 8).

### 9.1.3. dG(1)-method

Here, the time-discrete velocity and pressure solution is determined in the solution space $(\mathbf{u}_\tau,p_\tau)\in \mathbf{Y}_\tau^{k-1}\times\tilde{Y}_\tau^{k-1}$, where $k\geq 1$. The ansatz for $(\mathbf{u}_\tau,p_\tau)$ on interval $I_n$ is then analog to (9.5) with the difference that the sum starts with $j=1$ and the scalar basis functions $\phi_{n,j}$ are polynomials of order $k-1$. In this thesis, we will concentrate only on the case $k=2$, i.e. on the well-known dG(1)-method. We can derive the following constants for $i,j\in\{1,2\}$ (see again Chapter 5 and 8)

$$(\alpha_{i,j}) = \begin{pmatrix} 1 & \frac{\sqrt{3}-1}{2} \\ \frac{-\sqrt{3}-1}{2} & 1 \end{pmatrix}, \quad (d_i) = \begin{pmatrix} \frac{\sqrt{3}+1}{2} \\ \frac{-\sqrt{3}+1}{2} \end{pmatrix}. \tag{9.18}$$

Then, on the time interval $I_n$, one has to determine the two "unknowns" $(\mathbf{U}_n^j,P_n^j)\in\mathbf{V}\times Q$ as the solution of the following coupled system

$$\begin{aligned} \alpha_{1,1}\left(\mathbf{U}_n^1,\mathbf{v}\right)_\Omega + \tfrac{\tau_n}{2}a(\mathbf{U}_n^1,\mathbf{v}) + \tfrac{\tau_n}{2}n(\mathbf{U}_n^1,\mathbf{U}_n^1,\mathbf{v}) + \tfrac{\tau_n}{2}b(\mathbf{v},P_n^1) + \alpha_{1,2}\left(\mathbf{U}_n^2,\mathbf{v}\right)_\Omega &= \ell_1(\mathbf{v}) \\ \alpha_{2,1}\left(\mathbf{U}_n^1,\mathbf{v}\right)_\Omega + \alpha_{2,2}\left(\mathbf{U}_n^2,\mathbf{v}\right)_\Omega + \tfrac{\tau_n}{2}a(\mathbf{U}_n^2,\mathbf{v}) + \tfrac{\tau_n}{2}n(\mathbf{U}_n^2,\mathbf{U}_n^2,\mathbf{v}) + \tfrac{\tau_n}{2}b(\mathbf{v},P_n^2) &= \ell_2(\mathbf{v}) \\ b(\mathbf{U}_n^1,q) &= 0 \\ b(\mathbf{U}_n^2,q) &= 0 \end{aligned} \tag{9.19}$$

which has to be satisfied for all $(\mathbf{v},q)\in\mathbf{V}\times Q$ with $\ell_i(\cdot)$ defined by

$$\ell_i(\mathbf{v}) = \frac{\tau_n}{2}\left(f(t_{n,i}),\mathbf{v}\right)_\Omega + d_i\left(\mathbf{U}_n^0,\mathbf{v}\right)_\Omega \qquad i=1,2. \tag{9.20}$$

Once we have solved the above system, we obtain $\mathbf{u}_\tau$ and $p_\tau$ at the time $t_n$ by means of the following linear extrapolation

$$\mathbf{u}_\tau(t_n) = \frac{\sqrt{3}+1}{2}\mathbf{U}_n^2 - \frac{\sqrt{3}-1}{2}\mathbf{U}_n^1 \qquad \text{and} \qquad p_\tau(t_n) = \frac{\sqrt{3}+1}{2}P_n^2 - \frac{\sqrt{3}-1}{2}P_n^1. \tag{9.21}$$

The dG(1)-method is of order 2 in the whole time interval and superconvergent of order 3 in the discrete time points (see Chapter 5 and 8).

## 9.2. Space Discretization by FEM

After discretizing the Navier-Stokes equations (9.1) in time, we now discretize the resulting "$I_n$-problems" in space by using the finite element method [11, 17, 35, 55]. In our numerical experiments, the finite element spaces $\mathbf{V}_h \subset \mathbf{V}$ and $Q_h \subset Q$ are defined by biquadratic and discontinuous linear finite elements, respectively, on a quadrilateral mesh $T_h$ covering the computational domain $\Omega$. Each "$I_n$-problem" for the cGP($k$) or the dG($k-1$)-approach has the structure:

*For given $\mathbf{U}_n^0 \in \mathbf{V}$, find $(\mathbf{U}_n^j, P_n^j) \in \mathbf{V} \times Q$, $j = 1, \ldots, k$, such that*

$$\begin{aligned} \sum_{j=1}^{k} \alpha_{i,j} \left(\mathbf{U}_n^j, \mathbf{v}\right)_\Omega + \frac{\tau_n}{2} a(\mathbf{U}_n^i, \mathbf{v}) + \frac{\tau_n}{2} n(\mathbf{U}_n^i, \mathbf{U}_n^i, \mathbf{v}) + \frac{\tau_n}{2} b(\mathbf{v}, P_n^i) &= \ell_i(\mathbf{v}) \\ b(\mathbf{U}_n^i, q) &= 0, \end{aligned} \tag{9.22}$$

*which has to be satisfied for all $i = 1, \ldots, k$ and all $(\mathbf{v}, q) \in \mathbf{V} \times Q$ with*

$$\ell_i(\mathbf{v}) := \frac{\tau_n}{2} \left(f(t_{n,i}), \mathbf{v}\right)_\Omega + d_i \left(\mathbf{U}_n^0, \mathbf{v}\right)_\Omega \tag{9.23}$$

*where $\alpha_{i,j}$ and $d_i$ are the corresponding constants described above.*

For the space discretization, all $(\mathbf{U}_n^j, P_n^j) \in \mathbf{V} \times Q$ are approximated by finite element functions $(\mathbf{U}_{n,h}^j, P_{n,h}^j) \in \mathbf{V}_h \times Q_h$, respectively, and the fully discrete "$I_n$-problem" reads:

*For given $\mathbf{U}_{n,h}^0 \in \mathbf{V}_h$, find $(\mathbf{U}_{n,h}^j, P_{n,h}^j) \in \mathbf{V}_h \times Q_h$, $j = 1, \ldots, k$, such that it holds*

$$\begin{aligned} \sum_{j=1}^{k} \alpha_{i,j} \left(\mathbf{U}_{n,h}^j, \mathbf{v}_h\right)_\Omega + \frac{\tau_n}{2} a(\mathbf{U}_{n,h}^i, \mathbf{v}_h) + \frac{\tau_n}{2} n(\mathbf{U}_{n,h}^i, \mathbf{U}_{n,h}^i, \mathbf{v}_h) + \frac{\tau_n}{2} b(\mathbf{v}_h, P_{n,h}^i) &= \ell_i(\mathbf{v}_h) \\ b(\mathbf{U}_{n,h}^i, q_h) &= 0 \end{aligned} \tag{9.24}$$

*for all $(\mathbf{v}_h, q_h) \in \mathbf{V}_h \times Q_h$ and all $i = 1, \ldots, k$.*

Once we have solved this system, we have computed for each time $t \in I_n$ a finite element approximation $u_{\tau,h}(t) \in \mathbf{V}_h$ of the time discrete solution $\mathbf{u}_\tau(t) \in \mathbf{V}$ which is defined by an analogous ansatz to (9.5) where the $\mathbf{U}_n^j \in \mathbf{V}$ are replaced by the discrete functions $\mathbf{U}_{n,h}^j \in \mathbf{V}_h$.

In the following, we will write problem (9.24) as a nonlinear algebraic block system. Let $S_h \subset H_0^1(\Omega)$ denote the scalar finite element space for the velocity components $U_n^j, V_n^j \in S_h$ of $\mathbf{U}_{n,h}^j = (U_{n,h}^j, V_{n,h}^j) \in \mathbf{V}_h = S_h^2$ and let $\phi_\mu \in S_h$, $\mu = 1, \ldots, m_h$, denote the scalar finite element basis functions of $S_h$. Then, we define the nodal vector $\underline{\mathbf{U}}_n^j = (\underline{U}_n^j, \underline{V}_n^j) \in \mathbb{R}^{2m_h}$ of $\mathbf{U}_{n,h}^j = (U_{n,h}^j, V_{n,h}^j) \in \mathbf{V}_h$ such that

$$U_{n,h}^j(x) = \sum_{\mu=1}^{m_h} (\underline{U}_n^j)_\mu \phi_\mu(x), \qquad V_{n,h}^j(x) = \sum_{\mu=1}^{m_h} (\underline{V}_n^j)_\mu \phi_\mu(x) \qquad \forall\, x \in \Omega. \tag{9.25}$$

Similarly for the pressure, let $\psi_\mu \in Q_h$, $\mu = 1, \ldots, n_h$, denote the finite element basis functions and $\underline{P}_n^j \in \mathbb{R}^{n_h}$ the nodal vector of $P_{n,h}^j \in Q_h$ such that

$$P_{n,h}^j(x) = \sum_{\mu=1}^{n_h} (\underline{P}_n^j)_\mu \psi_\mu(x) \qquad \forall\, x \in \Omega. \tag{9.26}$$

Furthermore, we introduce the mass matrix $M \in \mathbb{R}^{m_h \times m_h}$, the discrete Laplacian matrix $L \in \mathbb{R}^{m_h \times m_h}$, the gradient matrices $B_i \in \mathbb{R}^{n_h \times m_h}$, $i = 1, 2$, as

$$M_{\nu,\mu} := (\phi_\mu, \phi_\nu)_\Omega, \quad L_{\nu,\mu} := a(\phi_\mu, \phi_\nu), \quad (B_i)_{\nu,\mu} := b(\phi_\mu \mathbf{e}^i, \psi_\nu), \tag{9.27}$$

and the right hand side vectors $F_n^i, G_n^i \in \mathbb{R}^{m_h}$, $i = 1,\dots,k$, with the components

$$(F_n^i)_\nu := \left(f(t_{n,i}), \phi_\nu \mathbf{e}^1\right)_\Omega, \quad (G_n^i)_\nu := \left(f(t_{n,i}), \phi_\nu \mathbf{e}^2\right)_\Omega. \tag{9.28}$$

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

$$N(\underline{\mathbf{w}})_{\nu,\mu} := n(\mathbf{w}_h, \phi_\mu, \phi_\nu). \tag{9.29}$$

Using the block-matrices and block-vectors

$$\mathbf{M} = \begin{bmatrix} M & 0 \\ 0 & M \end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix} L & 0 \\ 0 & L \end{bmatrix}, \quad \mathbf{N}(\underline{\mathbf{w}}) = \begin{bmatrix} N(\underline{\mathbf{w}}) & 0 \\ 0 & N(\underline{\mathbf{w}}) \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \quad \mathbf{F}_n^i = \begin{bmatrix} F_n^i \\ G_n^i \end{bmatrix}, \tag{9.30}$$

the fully discrete "$I_n$-problem" is equivalent to the following nonlinear $k \times k$ block system:

*For given $\underline{\mathbf{U}}_n^0 \in \mathbb{R}^{2m_h}$, find $\underline{\mathbf{U}}_n^j \in \mathbb{R}^{2m_h}$ and $\underline{P}_n^j \in \mathbb{R}^{n_h}$, $j = 1,\dots,k$, such that for all $i = 1,\dots,k$, it holds*

$$\begin{aligned} \sum_{j=1}^{k} \alpha_{i,j} \mathbf{M} \underline{\mathbf{U}}_n^j + \frac{\tau_n}{2} \mathbf{L} \underline{\mathbf{U}}_n^i + \frac{\tau_n}{2} \mathbf{N}(\underline{\mathbf{U}}_n^i) \underline{\mathbf{U}}_n^i + \frac{\tau_n}{2} \mathbf{B} \underline{P}_n^i &= d_i \mathbf{M} \underline{\mathbf{U}}_n^0 + \tfrac{\tau_n}{2} \mathbf{F}_n^i, \\ \mathbf{B}^T \underline{\mathbf{U}}_n^i &= 0. \end{aligned} \tag{9.31}$$

The vector $\underline{\mathbf{U}}_n^0$ is defined as the finite element nodal vector of the fully discrete solution $\mathbf{u}_{\tau,h}(t_{n-1})$ computed from the previous time interval $[t_{n-2}, t_{n-1}]$ if $n \geq 2$ or from a finite element interpolation of the initial data $\mathbf{u}_0$ if $n = 1$. In the case of higher Reynolds numbers, we apply additionally an edge oriented FEM stabilization (EOFEM) [59] for the convective term. This means that we replace the trilinear form $n(\mathbf{w},\cdot,\cdot)$ by a modified form $n_h(\mathbf{w},\cdot,\cdot)$ such that, in (9.31), differences will appear only in the nonlinear matrix $\mathbf{N}(\underline{\mathbf{w}})$.

In the following, we will present the resulting block systems for the cGP(1), cGP(2) and dG(1) method which will be used in our numerical experiments.

### 9.2.1. cGP(1)-method

The $3 \times 3$ block system on the time interval $I_n$ reads:

*For given initial velocity $\underline{\mathbf{U}}_n^0 = (\underline{U}_n^0, \underline{V}_n^0)$, find $\underline{\mathbf{U}}_n^1 = (\underline{U}_n^1, \underline{V}_n^1)$ and a pressure $\underline{P}_n^1$ such that*

$$\begin{bmatrix} A(u,v) & 0 & B_u \\ 0 & A(u,v) & B_v \\ B_u^T & B_v^T & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ p \end{bmatrix} = \begin{bmatrix} R_u \\ R_v \\ 0 \end{bmatrix} \tag{9.32}$$

where

$$A(u,v) = M + \frac{\tau_n}{2} L + \frac{\tau_n}{2} N(u,v), \qquad B_u = B_1, \qquad B_v = B_2, \tag{9.33}$$

with the abbreviations

$$u = \underline{U}_n^1, \quad v = \underline{V}_n^1, \quad p = \frac{\tau_n}{2} \underline{P}_n^1 \tag{9.34}$$

and the convection matrix $N(u,v)$ denoting the matrix $N(\underline{\mathbf{w}})$ defined in (9.29) with the nodal vector $\underline{\mathbf{w}} := (u,v) \in \mathbb{R}^{2m_h}$. The right hand side vectors $R_u$ and $R_v$ are given by

$$R_u = \tfrac{\tau_n}{2} F_n^1 + M \underline{U}_n^0, \qquad R_v = \tfrac{\tau_n}{2} G_n^1 + M \underline{V}_n^0. \tag{9.35}$$

Once we have determined the solution $\underline{\mathbf{U}}_n^1 = (\underline{U}_n^1, \underline{V}_n^1)$ we compute the nodal vector $\underline{\mathbf{U}}_{n+1}^0 = (\underline{U}_{n+1}^0, \underline{V}_{n+1}^0)$ of the fully discrete solution $u_{\tau,h}$ at the time $t_n$ by using the following linear extrapolation

$$u_{\tau,h}(t_n) \sim \underline{U}_{n+1}^0 = 2\underline{U}_n^1 - \underline{U}_n^0, \qquad v_{\tau,h}(t_n) \sim \underline{V}_{n+1}^0 = 2\underline{V}_n^1 - \underline{V}_n^0. \tag{9.36}$$

### 9.2.2. cGP(2)-method

The $6 \times 6$ block system on the time interval $I_n$ reads:

*For given initial velocity* $\underline{\mathbf{U}}_n^0 = (\underline{U}_n^0, \underline{V}_n^0)$, *find* $\underline{U}_n^1, \underline{U}_n^2, \underline{V}_n^1, \underline{V}_n^2$ *and* $\underline{P}_n^1, \underline{P}_n^2$ *such that*

$$\begin{bmatrix} A(u,v) & 0 & B_u \\ 0 & A(u,v) & B_v \\ B_u^T & B_v^T & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ p \end{bmatrix} = \begin{bmatrix} R_u \\ R_v \\ 0 \end{bmatrix} \tag{9.37}$$

where

$$A(u,v) = \begin{bmatrix} 3M + \tau_n L + \tau_n N(u^1, v^1) & \left(2\sqrt{3} - 3\right) M \\ \left(-2\sqrt{3} - 3\right) M & 3M + \tau_n L + \tau_n N(u^2, v^2) \end{bmatrix}, \tag{9.38}$$

$$B_u = \begin{bmatrix} B_1 & 0 \\ 0 & B_1 \end{bmatrix}, \qquad B_v = \begin{bmatrix} B_2 & 0 \\ 0 & B_2 \end{bmatrix} \tag{9.39}$$

with the abbreviations

$$u = \begin{bmatrix} u^1 \\ u^2 \end{bmatrix} = \begin{bmatrix} \underline{U}_n^1 \\ \underline{U}_n^2 \end{bmatrix}, \quad v = \begin{bmatrix} v^1 \\ v^2 \end{bmatrix} = \begin{bmatrix} \underline{V}_n^1 \\ \underline{V}_n^2 \end{bmatrix}, \quad p = \begin{bmatrix} p^1 \\ p^2 \end{bmatrix} = \begin{bmatrix} \tau_n \underline{P}_n^1 \\ \tau_n \underline{P}_n^2 \end{bmatrix} \tag{9.40}$$

The right hand side vectors $R_u$ and $R_v$ are given by

$$R_u = \begin{bmatrix} R_u^1 \\ R_u^2 \end{bmatrix} = \begin{bmatrix} \tau_n F_n^1 + 2\sqrt{3} M \underline{U}_n^0 \\ \tau_n F_n^2 - 2\sqrt{3} M \underline{U}_n^0 \end{bmatrix}, \qquad R_v = \begin{bmatrix} R_v^1 \\ R_v^2 \end{bmatrix} = \begin{bmatrix} \tau_n G_n^1 + 2\sqrt{3} M \underline{V}_n^0 \\ \tau_n G_n^2 - 2\sqrt{3} M \underline{V}_n^0 \end{bmatrix}. \tag{9.41}$$

The nodal vectors $\underline{U}_n^i$ and $\underline{V}_n^i$, $i = 1, 2$, are associated with the finite element approximations $u_{\tau,h}(t_{n,i})$ and $v_{\tau,h}(t_{n,i})$, respectively, where $t_{n,i}$ denotes the $i$-th integration point of the 2-point Gaussian quadrature rule on the time interval $I_n$. Once they have been computed, we get the nodal vector $\underline{\mathbf{U}}_{n+1}^0 = (\underline{U}_{n+1}^0, \underline{V}_{n+1}^0)$ of the fully discrete solution $u_{\tau,h}$ at the time $t_n$ by using the following quadratic extrapolation

$$u_{\tau,h}(t_n) \sim \underline{U}_{n+1}^0 = \underline{U}_n^0 + \sqrt{3}(\underline{U}_n^2 - \underline{U}_n^1), \qquad v_{\tau,h}(t_n) \sim \underline{V}_{n+1}^0 = \underline{V}_n^0 + \sqrt{3}(\underline{V}_n^2 - \underline{V}_n^1). \tag{9.42}$$

### 9.2.3. dG(1)-method

The analogous $6 \times 6$ block system on the time interval $I_n$ reads:

*For given initial velocity* $\underline{\mathbf{U}}_n^0 = (\underline{U}_n^0, \underline{V}_n^0)$, *find* $\underline{U}_n^1, \underline{U}_n^2, \underline{V}_n^1, \underline{V}_n^2$ *and* $\underline{P}_n^1, \underline{P}_n^2$ *such that*

$$\begin{bmatrix} A(u,v) & 0 & B_u \\ 0 & A(u,v) & B_v \\ B_u^T & B_v^T & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ p \end{bmatrix} = \begin{bmatrix} R_u \\ R_v \\ 0 \end{bmatrix} \tag{9.43}$$

where

$$A(u,v) = \begin{bmatrix} 2M + \tau_n L + \tau_n N(u^1, v^1) & \left(\sqrt{3} - 1\right) M \\ \left(-\sqrt{3} - 1\right) M & 2M + \tau_n L + \tau_n N(u^2, v^2) \end{bmatrix} \tag{9.44}$$

and $B_u$, $B_v$, $u$, $v$, $p$ are defined as in (9.39) and (9.40). The right hand side vectors $R_u$ and $R_v$ are given by

$$R_u = \begin{bmatrix} R_u^1 \\ R_u^2 \end{bmatrix} = \begin{bmatrix} \tau_n F_n^1 + \left(\sqrt{3} + 1\right) M \underline{U}_n^0 \\ \tau_n F_n^2 + \left(-\sqrt{3} + 1\right) M \underline{U}_n^0 \end{bmatrix}, \quad R_v = \begin{bmatrix} R_v^1 \\ R_v^2 \end{bmatrix} = \begin{bmatrix} \tau_n G_n^1 + \left(\sqrt{3} + 1\right) M \underline{V}_n^0 \\ \tau_n G_n^2 + \left(-\sqrt{3} + 1\right) M \underline{V}_n^0 \end{bmatrix}. \tag{9.45}$$

Again the nodal vectors $\underline{U}_n^i$ and $\underline{V}_n^i$, $i = 1,2$, are associated with the finite element approximations $u_{\tau,h}(t_{n,i})$ and $v_{\tau,h}(t_{n,i})$, respectively, where $t_{n,i}$ denotes the $i$-th integration point of the 2-point Gaussian quadrature rule on the time interval $I_n$. Once they have been computed, we get the nodal vector $\underline{\mathbf{U}}_{n+1}^0 = (\underline{U}_{n+1}^0, \underline{V}_{n+1}^0)$ and $\underline{P}_{n+1}^0$ of the left side limit of the fully discrete solution $u_{\tau,h}$ at the time $t_n$ by using the following linear extrapolation

$$\begin{aligned} u_{\tau,h}^-(t_n) \sim \underline{U}_{n+1}^0 &= \tfrac{\sqrt{3}+1}{2} U_n^2 - \tfrac{\sqrt{3}-1}{2} U_n^1, \\ v_{\tau,h}^-(t_n) \sim \underline{V}_{n+1}^0 &= \tfrac{\sqrt{3}+1}{2} V_n^2 - \tfrac{\sqrt{3}-1}{2} V_n^1, \\ p_{\tau,h}^-(t_n) \sim \underline{P}_{n+1}^0 &= \tfrac{\sqrt{3}+1}{2} P_n^2 - \tfrac{\sqrt{3}-1}{2} P_n^1. \end{aligned} \tag{9.46}$$

## 9.3. Nonlinear Solver

For all introduced time-space discretization schemes described before, a nonlinear system of algebraic equations of the following type has to be solved for each time interval:

$$\begin{bmatrix} A(u,v) & 0 & B_u \\ 0 & A(u,v) & B_v \\ B_u^T & B_v^T & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ p \end{bmatrix} = \begin{bmatrix} R_u \\ R_v \\ 0 \end{bmatrix} \tag{9.47}$$

The nonlinear system (9.47) can be characterized as a saddle point problem. This system will be solved either by means of a standard fixed point iteration or by the Newton method. We will denote this solution approach as the outer nonlinear iteration. In each outer iteration step, a coupled linear system has to be solved. The linear subproblems are solved by using a coupled geometrical multigrid solver with a smoother based on blocking of all cell unknowns and canonical FEM grid transfer operators. In addition, the coarse grid problem is solved by using the Gaussian elimination or preconditioned GMRES method.

### 9.3.1. General nonlinear outer iteration

On an actual time interval $I_n = (t_{n-1}, t_n]$, we define the start iterate $(u_0, v_0, p_0)$ of the nonlinear iteration by means of the known solution at time $t_{n-1}$ from the previous time interval for $n > 1$ or from the space-discrete initial solution for $n = 1$. In the case of cGP(2) or dG(1), we define also the start iterates of the second components $u^2$, $v^2$ and $p^2$ in (9.40) as the solution at time $t_{n-1}$. For a given old iterate $(u_\ell, v_\ell, p_\ell)$, we perform the following three steps to compute the new iterate $(u_{\ell+1}, v_{\ell+1}, p_{\ell+1})$:

1. Compute a defect vector containing the nonlinear residual for $(u_\ell, v_\ell, p_\ell)$

$$\begin{bmatrix} d_u \\ d_v \\ d_p \end{bmatrix} = \begin{bmatrix} R_u \\ R_v \\ 0 \end{bmatrix} - \begin{bmatrix} A(u_\ell, v_\ell) & 0 & B_u \\ 0 & A(u_\ell, v_\ell) & B_v \\ B_u^T & B_v^T & 0 \end{bmatrix} \begin{bmatrix} u_\ell \\ v_\ell \\ p_\ell \end{bmatrix}. \tag{9.48}$$

2. Solve an auxiliary (linear) subproblem with the defect vector as right hand side

$$\begin{bmatrix} A_{uu}(u_\ell, v_\ell) & A_{uv}(u_\ell, v_\ell) & B_u \\ A_{vu}(u_\ell, v_\ell) & A_{vv}(u_\ell, v_\ell) & B_v \\ B_u^T & B_v^T & 0 \end{bmatrix} \begin{bmatrix} \Delta u_\ell \\ \Delta v_\ell \\ \Delta p_\ell \end{bmatrix} = \begin{bmatrix} d_u \\ d_v \\ d_p \end{bmatrix}, \tag{9.49}$$

where $A_{uu}, A_{uv}, A_{vu}, A_{vv}$ are chosen due to the fixed-point iteration or the Newton method.

3. Update the iterate to obtain $(u_{\ell+1}, v_{\ell+1}, p_{\ell+1})$

$$\begin{bmatrix} u_{\ell+1} \\ v_{\ell+1} \\ p_{\ell+1} \end{bmatrix} = \begin{bmatrix} u_\ell \\ v_\ell \\ p_\ell \end{bmatrix} + \omega_\ell \begin{bmatrix} \Delta u_\ell \\ \Delta v_\ell \\ \Delta p_\ell \end{bmatrix}, \tag{9.50}$$

where $\omega_\ell$ is a damping parameter which is in most of the cases set to 1 (see [58] for adaptive strategies). The last iterate $(u_{\ell+1}, v_{\ell+1}, p_{\ell+1})$ for the nonlinear iteration, depending on the stopping criterion, is accepted for the solution. In our numerical simulations, the nonlinear iteration is stopped if the $L^2$-norm of the nonlinear residual drops down below $10^{-10}$.

### 9.3.2. Fixed-point iteration

For the fixed-point iteration, the matrices $A_{uu}, A_{uv}, A_{vu}, A_{vv}$ in (9.49) are given by

$$A_{uu}(u_\ell, v_\ell) := A_{vv}(u_\ell, v_\ell) := A(u_\ell, v_\ell), \tag{9.51}$$
$$A_{uv}(u_\ell, v_\ell) := A_{vu}(u_\ell, v_\ell) := 0. \tag{9.52}$$

### 9.3.3. Newton method

Applying the standard Newton method yields the following block matrices

$$A_{uu}(u_\ell, v_\ell) := A(u_\ell, v_\ell) + S_{uu}^{(k)}(u_\ell, v_\ell), \tag{9.53}$$
$$A_{vv}(u_\ell, v_\ell) := A(u_\ell, v_\ell) + S_{vv}^{(k)}(u_\ell, v_\ell), \tag{9.54}$$
$$A_{uv}(u_\ell, v_\ell) := S_{uv}^{(k)}(u_\ell, v_\ell), \tag{9.55}$$
$$A_{vu}(u_\ell, v_\ell) := S_{vu}^{(k)}(u_\ell, v_\ell), \tag{9.56}$$

where $k \in \{1,2\}$ denotes the order of the underlying time discretization cGP($k$) or dG($k-1$), respectively, and $S_{\alpha\beta}^{(k)}$ correspond to the additional terms from the linearization of the nonlinear convection term $(\mathbf{u}_\ell \cdot \nabla \mathbf{u}_\ell)$. In the following, we present explicitly for each time discretization the blocks of the full Newton matrix $\mathcal{N}$ in (9.49), i.e.,

$$\mathcal{N} := \begin{bmatrix} A(u_\ell, v_\ell) + S_{uu}^{(k)}(u_\ell, v_\ell) & S_{uv}^{(k)}(u_\ell, v_\ell) & B_u \\ S_{vu}^{(k)}(u_\ell, v_\ell) & A(u_\ell, v_\ell) + S_{vv}^{(k)}(u_\ell, v_\ell) & B_v \\ B_u^T & B_v^T & 0 \end{bmatrix}. \tag{9.57}$$

In the case of the cGP(1)-method, we have

$$A(u,v) = M + \frac{\tau_n}{2} L + \frac{\tau_n}{2} N(u,v), \qquad B_u = B_1, \qquad B_v = B_2, \tag{9.58}$$

and

$$\left(S_{uu}^{(1)}(\tilde{u},\tilde{v})\right)_{\nu,\mu} = \frac{\tau_n}{2}\left((\partial_x \tilde{u}_h)\phi_\mu, \phi_\nu\right)_\Omega, \qquad \left(S_{uv}^{(1)}(\tilde{u},\tilde{v})\right)_{\nu,\mu} = \frac{\tau_n}{2}\left((\partial_y \tilde{u}_h)\phi_\mu, \phi_\nu\right)_\Omega, \tag{9.59}$$
$$\left(S_{vu}^{(1)}(\tilde{u},\tilde{v})\right)_{\nu,\mu} = \frac{\tau_n}{2}\left((\partial_x \tilde{v}_h)\phi_\mu, \phi_\nu\right)_\Omega, \qquad \left(S_{vv}^{(1)}(\tilde{u},\tilde{v})\right)_{\nu,\mu} = \frac{\tau_n}{2}\left((\partial_y \tilde{v}_h)\phi_\mu, \phi_\nu\right)_\Omega, \tag{9.60}$$

where, $\tilde{u}_h \in S_h$ and $\tilde{v}_h \in S_h$ denote the finite element functions associated with the nodal vectors $\tilde{u}$ and $\tilde{v}$, respectively. For the cGP(2)-method, we have

$$A(u,v) = \begin{bmatrix} 3M + \tau_n L + \tau_n N(u^1, v^1) & \left(2\sqrt{3} - 3\right) M \\ \left(-2\sqrt{3} - 3\right) M & 3M + \tau_n L + \tau_n N(u^2, v^2) \end{bmatrix}, \tag{9.61}$$

$$B_u = \begin{bmatrix} B_1 & 0 \\ 0 & B_1 \end{bmatrix}, \qquad B_v = \begin{bmatrix} B_2 & 0 \\ 0 & B_2 \end{bmatrix}, \tag{9.62}$$

$$S^{(2)}_{\alpha\beta}(\tilde{u},\tilde{v}) = \begin{bmatrix} 2S^{(1)}_{\alpha\beta}(\tilde{u},\tilde{v}) & 0 \\ 0 & 2S^{(1)}_{\alpha\beta}(\tilde{u},\tilde{v}) \end{bmatrix} \qquad \forall\, \alpha,\beta \in \{u,v\} \tag{9.63}$$

with $S^{(1)}_{\alpha\beta}(\tilde{u},\tilde{v})$ defined in (9.59) and (9.60). In the case of the dG(1)-method, the matrix block $A(u,v)$ is

$$A(u,v) = \begin{bmatrix} 2M + \tau_n L + \tau_n N(u^1,v^1) & \left(\sqrt{3}-1\right) M \\ \left(-\sqrt{3}-1\right) M & 2M + \tau_n L + \tau_n N(u^2,v^2) \end{bmatrix} \tag{9.64}$$

and the other matrix blocks of $\mathcal{N}$ in (9.57) are the same as for the cGP(2)-method.

**Remark 9.1** *If the edge oriented jump stabilization (see [59]) is used, then the corresponding velocity matrix block $A(u,v)$ in* (9.47) *is updated as follows*

$$\tilde{A}(u_\ell, v_\ell) = A(u_\ell, v_\ell) + J, \tag{9.65}$$

*where the matrix J corresponds to additional "jump terms" in the weak formulation. This matrix J has the following form for the cGP(1) and cGP(2)-method, respectively,*

$$\textit{cGP(1):} \qquad J = (J_{\nu,\mu}), \qquad J_{\nu,\mu} = \sum_E max\{\gamma^* \nu h_E, \gamma h_E^2\} \int_E [\nabla \phi_\mu]_E [\nabla \phi_\nu]_E \, d\sigma, \tag{9.66}$$

$$\textit{cGP(2):} \qquad J = \begin{bmatrix} J_{\nu,\mu} & 0 \\ 0 & J_{\nu,\mu} \end{bmatrix}, \tag{9.67}$$

*where the sum in* (9.66) *is taken over all inner edges E of the mesh. $\nu, h_E$ denote the viscosity and the length of edge E. The parameters $\gamma$ and $\gamma^*$ have no significant influence on the accuracy of the results and the solution is stable and accurate for a large range of parameters. In our case, these parameters are set to 0.1 and 0.0, respectively. The jump $[.]_E$ is defined as $[\psi]_E = \psi^+ - \psi^-$, where $\psi^+, \psi^-$ indicate the values of the discontinuous function $\psi$ coming from the elements $K^+, K^-$ sharing the interior edge E.*

# 10

# Numerical Results

Analytical test cases to analyze the order of convergence have been considered in Chapter 8. In this chapter, we perform nonstationary simulations for more complex flow configurations to demonstrate the temporal accuracy and efficiency of the presented higher order time discretization schemes for prototypical test cases of benchmarking character. To this end, we continue the work which had been started by one of the authors in [57], where different time stepping schemes for two class of problems, namely *flow around cylinder* and *flow through a Venturi pipe*, were analyzed. First, we consider the *flow around cylinder* which has been described in [61]. Here, we will concentrate only on the nonstationary behavior of the flow patterns with periodic oscillations and examine the ability of the different time discretization schemes (and the chosen FEM discretization) to capture the dynamics of the flow.

As a second test case, we consider the nonstationary behavior of a higher Reynolds number flow through a Venturi pipe which has many real life and industrial applications, for instance, this venturi pipe can be used as a small device in sailing boats. If the inflow speed from the inlet is sufficiently high, then due to the Bernoulli principle, the narrow section in the middle of the pipe produces a low pressure which creates a flux through the upper part of the small pipe. The objective of this simulation is to control the instantaneous and mean flux through this device.

## 10.1. Nonstationary flow around cylinder benchmark

The flow configuration related to the 'flow around cylinder' configuration [61], which is considered here, is as follows:

- (Laminar) nonstationary Navier-Stokes equations at $Re = 100$
- Parabolic inflow with $U_{\max} = 1.5$
- Time domain: $[t_0, t_0 + T] = [0, 10]$, where $t_0$ corresponds to the fully developed solution for each mesh level
- Space-discretisation: $Q_2/P_1^{\text{disc}}$
- Time-discretisation: Crank-Nicolson (CN), cGP(1), cGP(2), dG(1)
- Stabilisation: none

We consider the nonstationary Navier-Stokes equations in a bounded domain $\Omega \subset \mathbb{R}^2$ with piecewise smooth boundary $\Gamma$. Here, the domain $\Omega$ consists of a channel of height $H = 0.41$ and length $L = 2.2$ having a circular cylinder located at $(0.2, 0.2)$ with diameter $D = 0.1$, placed at right angle to the direction of the fluid (see Figure 10.1). The value of the kinematic viscosity is set to $\nu = 10^{-3}$ and the density of the fluid $\rho = 1$.

The upper part ($y = 0.41$) and the lower part ($y = 0$) of the boundary $\Gamma$ and the boundary of the obstacle itself are subjected to the 'no-slip' boundary conditions ($u = v = 0$). The most left part of the boundary is set to produce an inflow of parabolic profile having the maximum velocity in x-direction $U_{max} = 1.5$. The velocity at the inlet is prescribed as

$$U(0,y) = 4U_{max}y(H-y)/H^2, \qquad V(0,y) = 0.$$

The most right part of the boundary $\Gamma$ acts as the outlet and natural boundary conditions are prescribed here. The *Reynolds number Re* determining the flow properties is defined as

$$Re = \frac{U_{mean}D}{\nu},$$

where $U_{mean}$ is calculated as

$$U_{mean} = \frac{2}{3}U(0,\frac{H}{2}).$$

The maximum velocity $U_{max} = 1.5$ yields $Re = 100$ which leads to periodically oscillatory time-dependent vortex shedding behind the cylinder. For this range of *Reynolds numbers* together with the $Q_2/P_1^{\text{disc}}$ discretization, our results have shown that there is no need for stabilization.

**Quantities of physical interest:**

The examined accuracy of the benchmark crucially depends on the following quantities

$$F_D = \int_S (\rho\nu\frac{\partial u_t}{\partial n}n_y - pn_x)dS, \quad \text{and particularly} \quad F_L = -\int_S (\rho\nu\frac{\partial u_t}{\partial n}n_x + pn_y)dS$$

representing the total forces in the horizontal and vertical directions, respectively. The resulting drag and lift coefficients are

$$C_D = \frac{2F_D}{\rho U_{mean}^2 D}, \qquad C_L = \frac{2F_L}{\rho U_{mean}^2 D}.$$

Furthermore, the pressure drop between two points on the cylinder which is defined as

$$\Delta p = p_A - p_B,$$

where $A(0.15,0.2)$ and $B(0.25,0.2)$ are points on the boundary of the cylinder, is also of interest. Beside these quantities, we also compare the accuracy of the different time discretization schemes by computing the $v$ velocity at the point $P(0.4,0.2)$ near the obstacle.

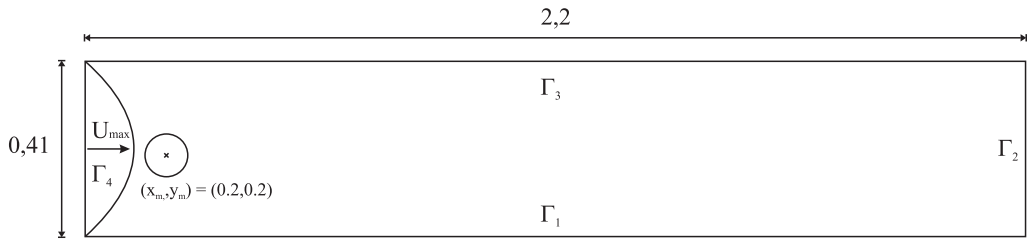



**Figure 10.1**: Geometry for the *flow around cylinder* configuration in 2D.

Figure 10.2 shows the initial coarse mesh (level 1), which will be uniformly refined, and Table 10.1 presents for different space mesh levels the number '#EL' of elements and the total number '#DOF' of degrees of freedom. The finite element discretization is carried out by using the described biquadratic $Q_2$-element for velocity and discontinuous $P_1$-element for the pressure.

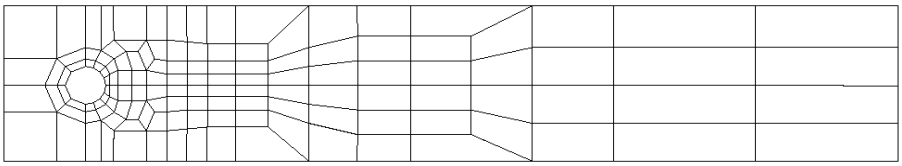


**Figure 10.2**: Coarse mesh designed for the *flow around cylinder* configuration in 2D.

| Lev. | #EL | #DOF(total) |
|---|---|---|
| 2 | 520 | 5 928 |
| 3 | 2 080 | 23 296 |
| 4 | 8 320 | 92 352 |

**Table 10.1**: Size of the different systems in space.

In order to compare the accuracy of the different time discretizations, the flow is started from a fully developed solution, that means the simulation on the same mesh with small time step had been performed until a fully periodical flow behavior had been reached, at time $t_0$. Then, after restarted with start solution $u(t_0)$, the simulation is performed again until T=10 using different time discretization methods for various uniform time step sizes $\tau_n := \tau$. After T=10, all the introduced quantities are plotted and analyzed in detail. To this end, in addition to the global picture of all quantities from time $t_0$ to $t_0 + 10$, these quantities are also zoomed in the last unit from T=9 to T=10. Since the results obtained from the Crank-Nicolson and cGP(1)-method are almost identical as expected, we show the results for the Crank-Nicolson only together with the cGP(2) and dG(1)-method. All the time discretization schemes are started from the same initial (fully developed) velocity field and pressure on each mesh level, and the simulations are performed with the same time step sizes. First, we show the results for mesh level 2. Next, we demonstrate that the results are already grid independent by showing them for different (space) levels for the cGP(2)-method, so that higher levels are not required.

For space mesh level 3 and 4, almost the same results have been obtained. Here, we plot only the lift coefficient and $v$-velocity for space level 3 and 4, respectively. Since the difference of these quantities between the space levels is very small, we show only the zoomed picture in the last time unit form T=9 to T=10.

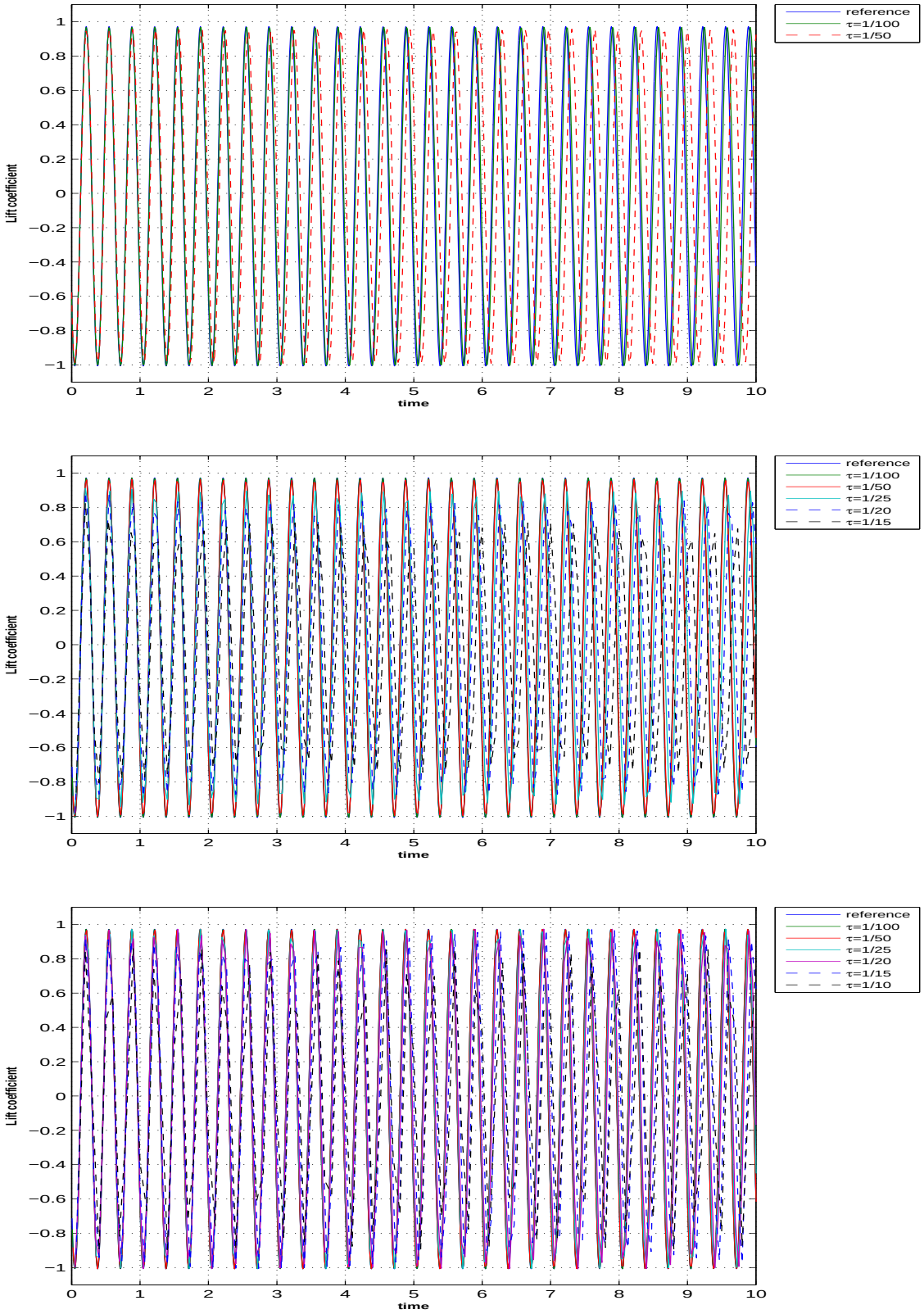



**Figure 10.3**: Lift coefficient for different $\tau$, using CN (top), dG(1) (middle) and cGP(2) (bottom) method for space level 2.

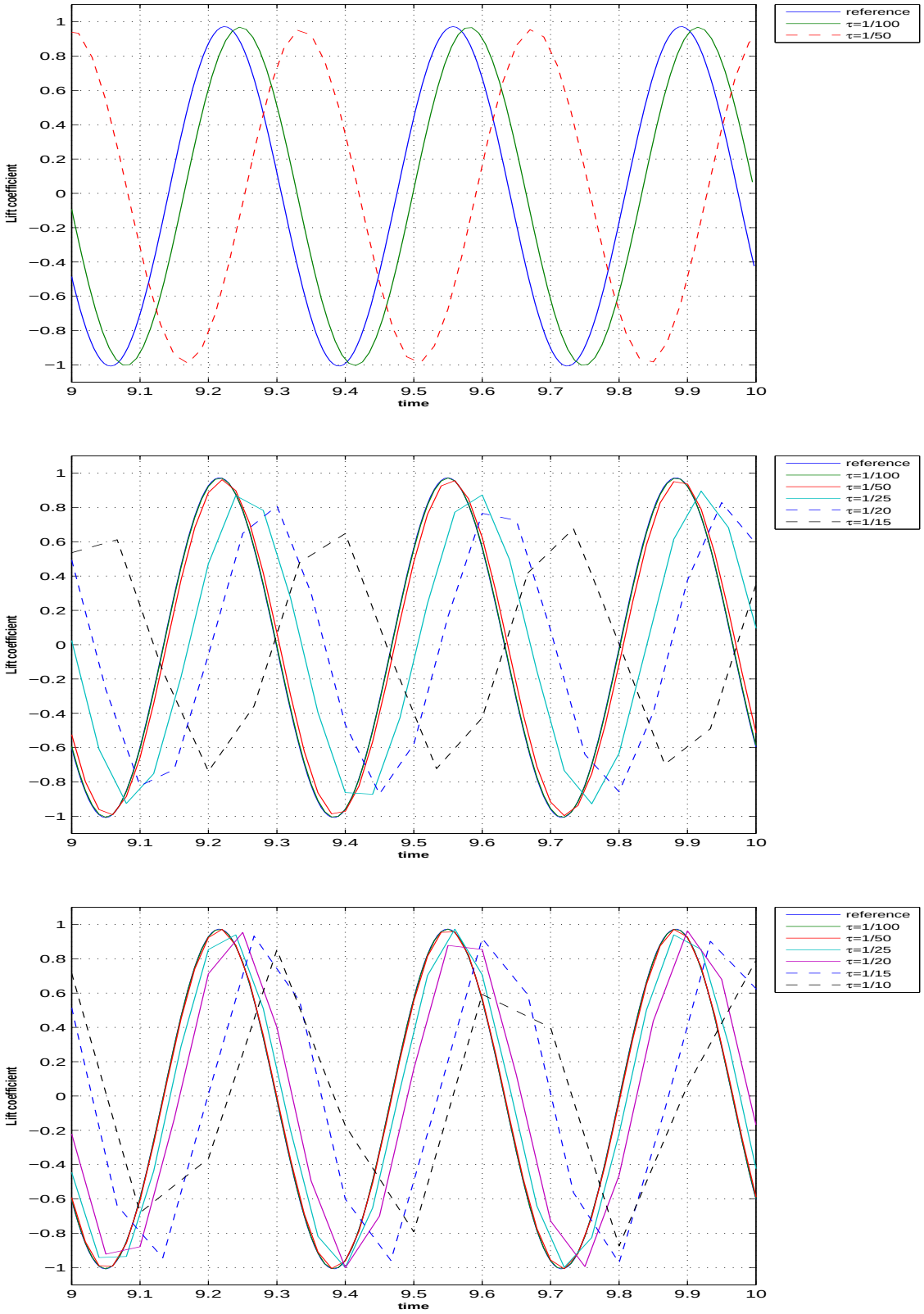



**Figure 10.4**: Lift coefficient for different $\tau$, using CN (top), dG(1) (middle) and cGP(2) (bottom) method for space level 2.

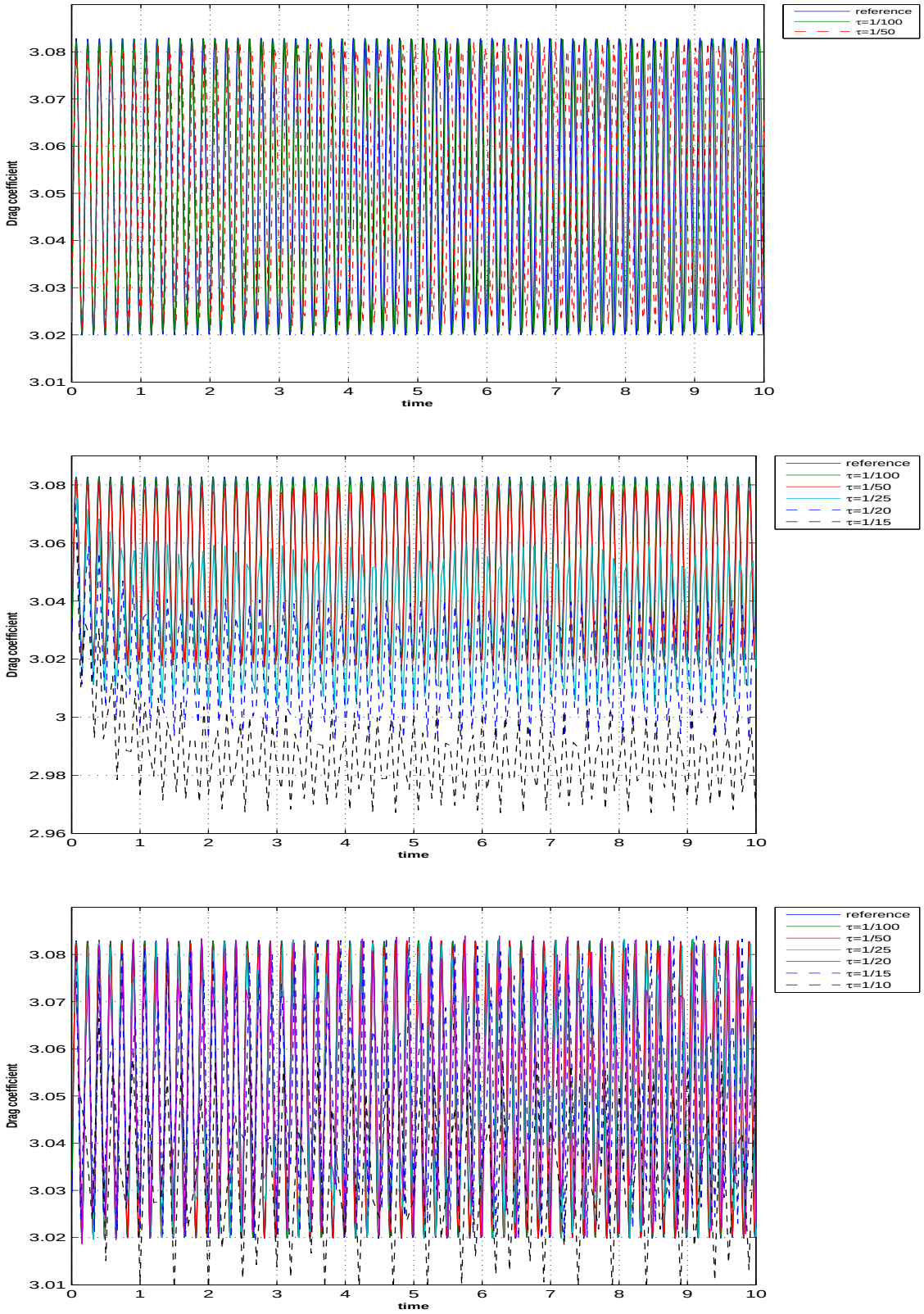



**Figure 10.5**: Drag coefficient for different $\tau$, using CN (top), dG(1) (middle) and cGP(2) (bottom) method for space level 2.

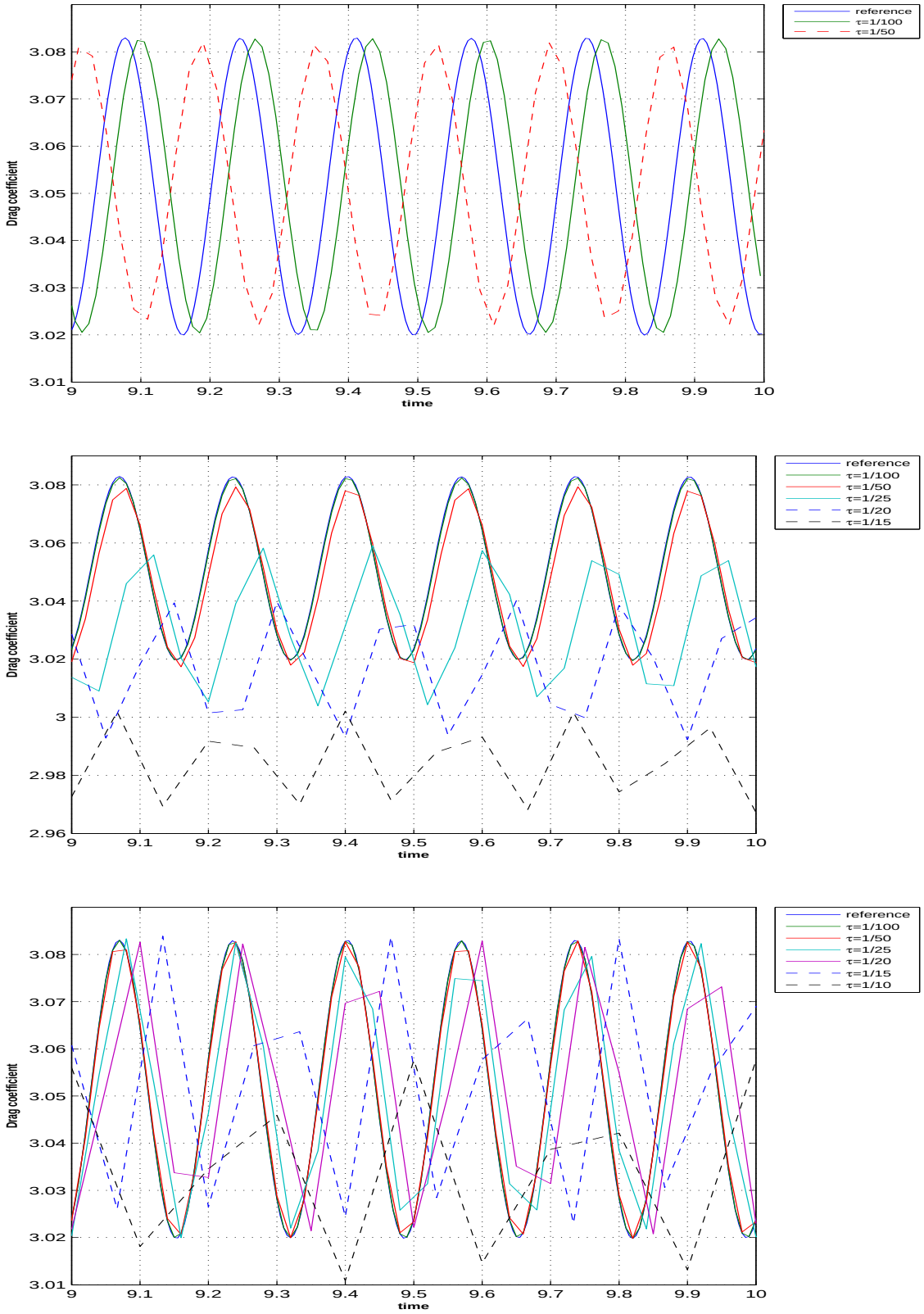



**Figure 10.6**: Drag coefficient for different $\tau$, using CN (top), dG(1) (middle) and cGP(2) (bottom) method for space level 2.

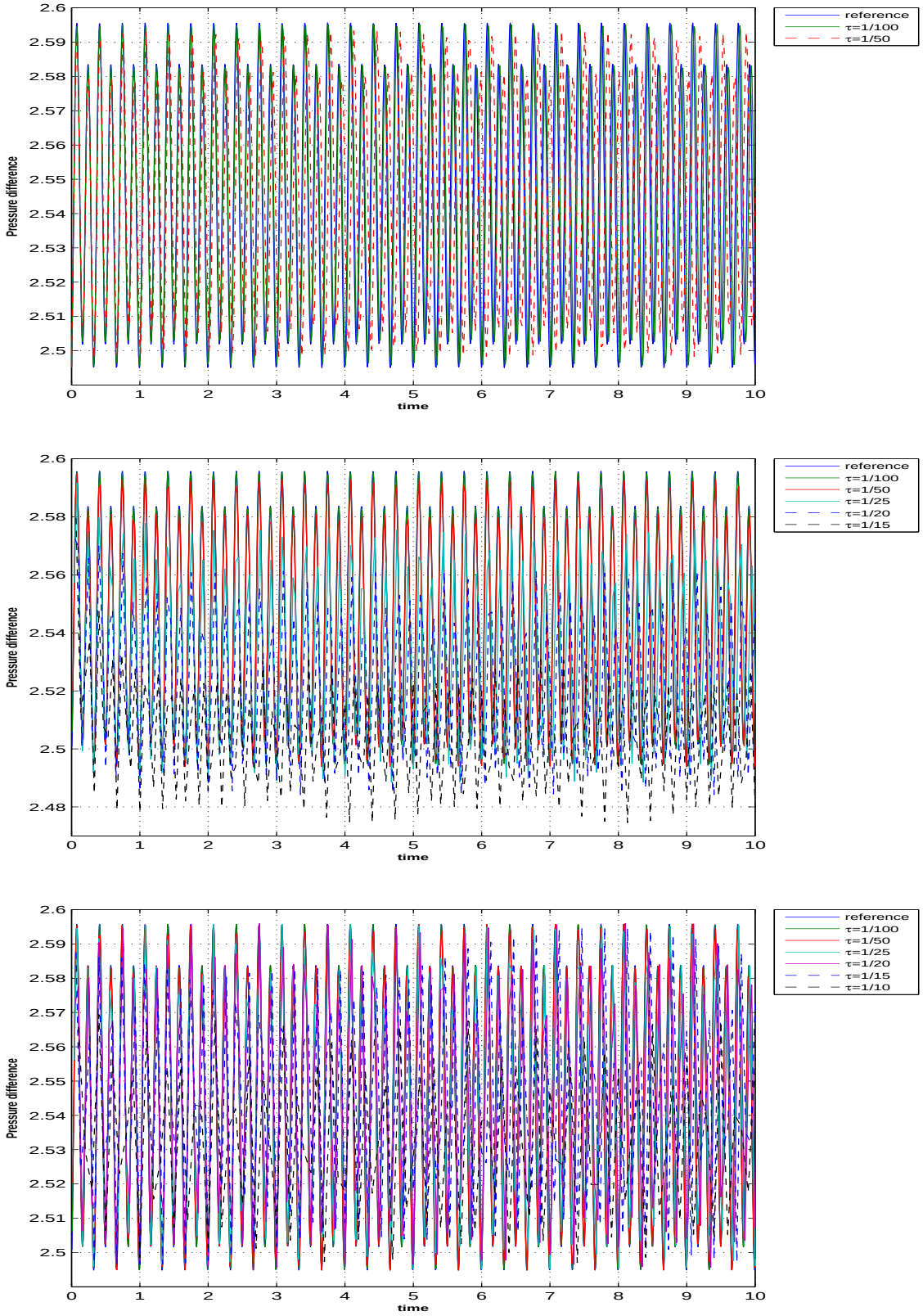



**Figure 10.7**: Pressure difference $\Delta p = p_A(0.15, 0.2) - p_B(0.25, 0.2)$ for different $\tau$, using CN (top), dG(1) (middle) and cGP(2) (bottom) method for space level 2.

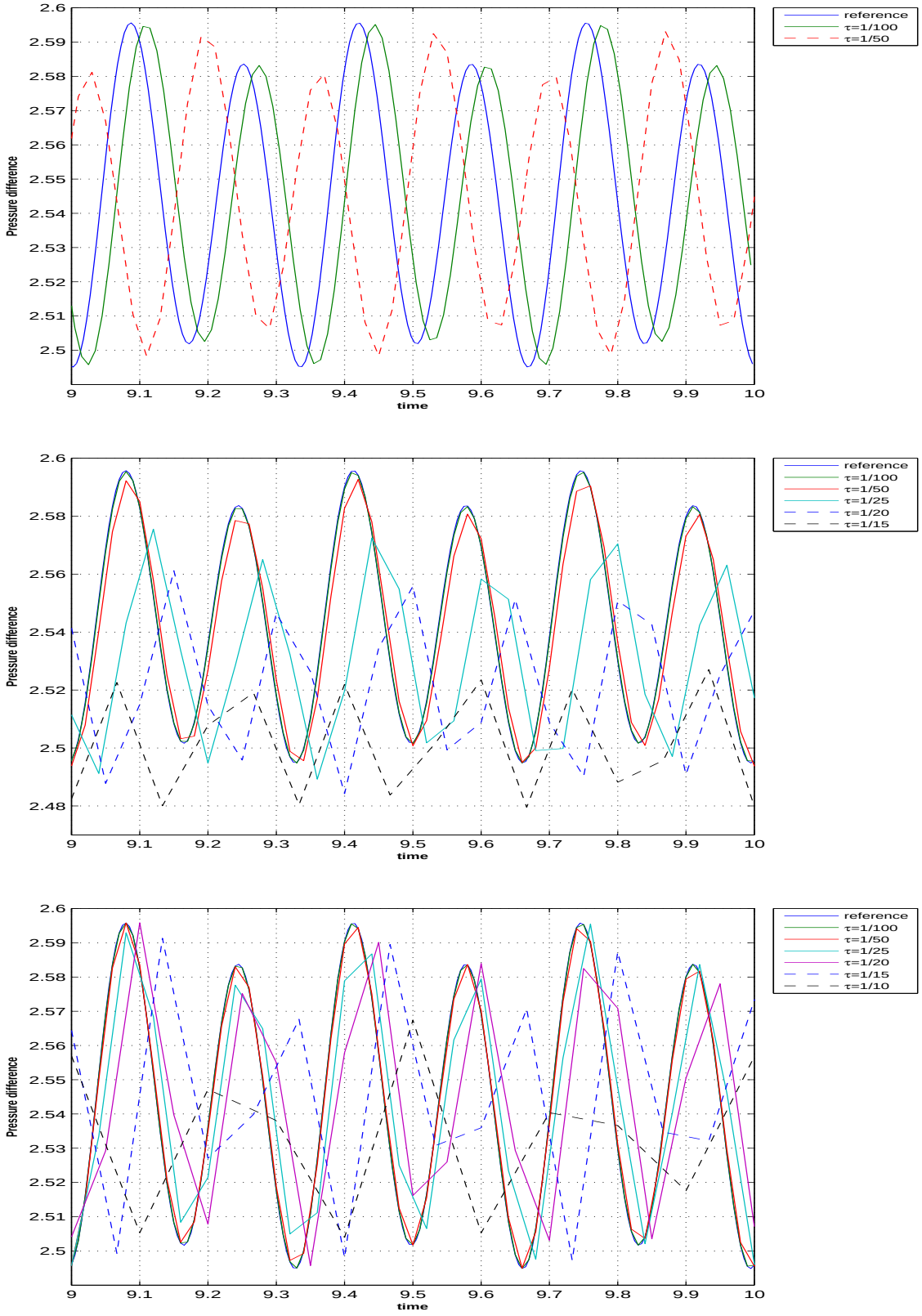



**Figure 10.8**: Pressure difference $\Delta p = p_A(0.15, 0.2) - p_B(0.25, 0.2)$ for different $\tau$, using CN (top), dG(1) (middle) and cGP(2) (bottom) method for space level 2.

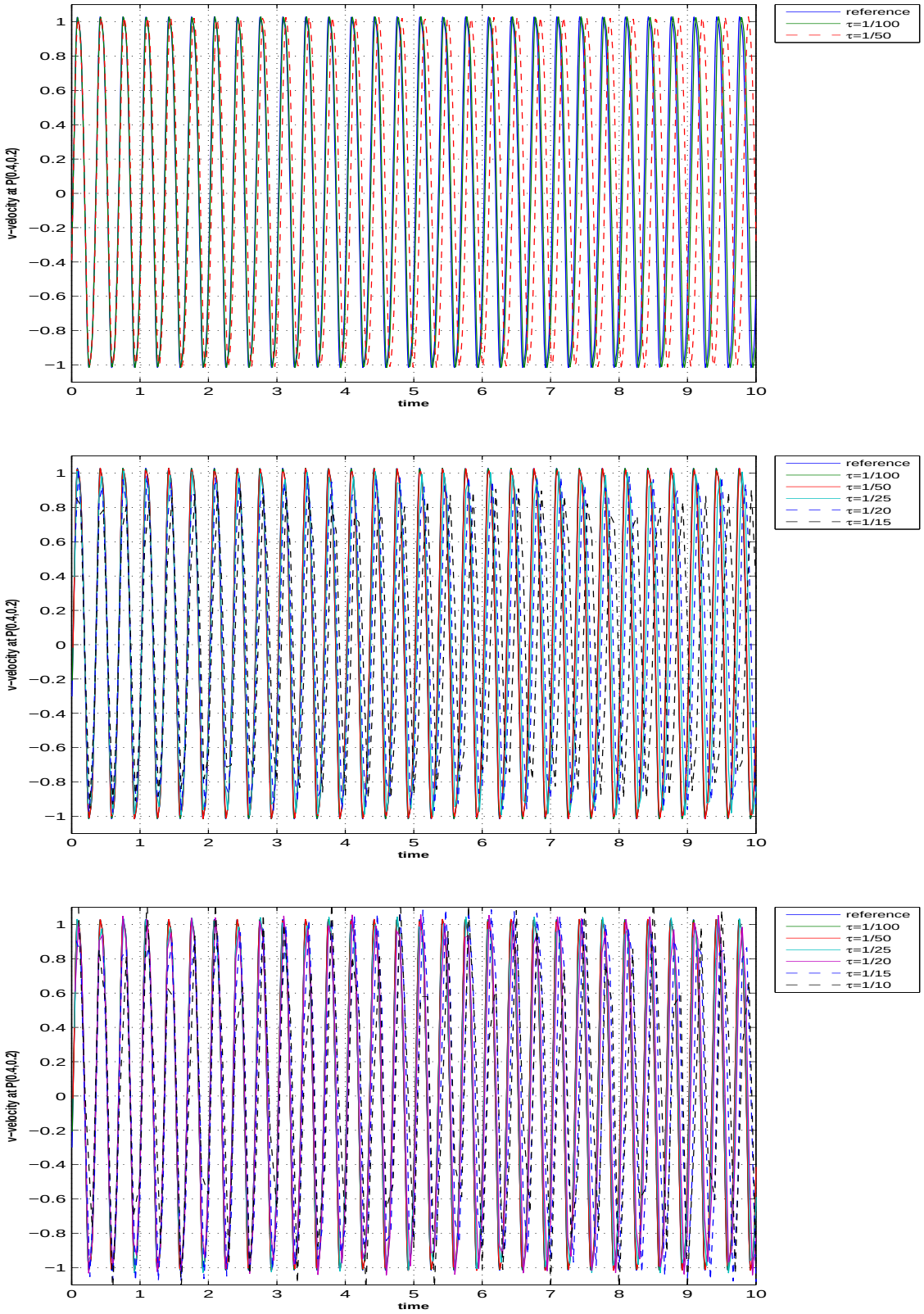



**Figure 10.9**: $v$-velocity at point $P(0.4, 0.2)$ for different $\tau$, using CN (top), dG(1) (middle) and cGP(2) (bottom) method for space level 2.

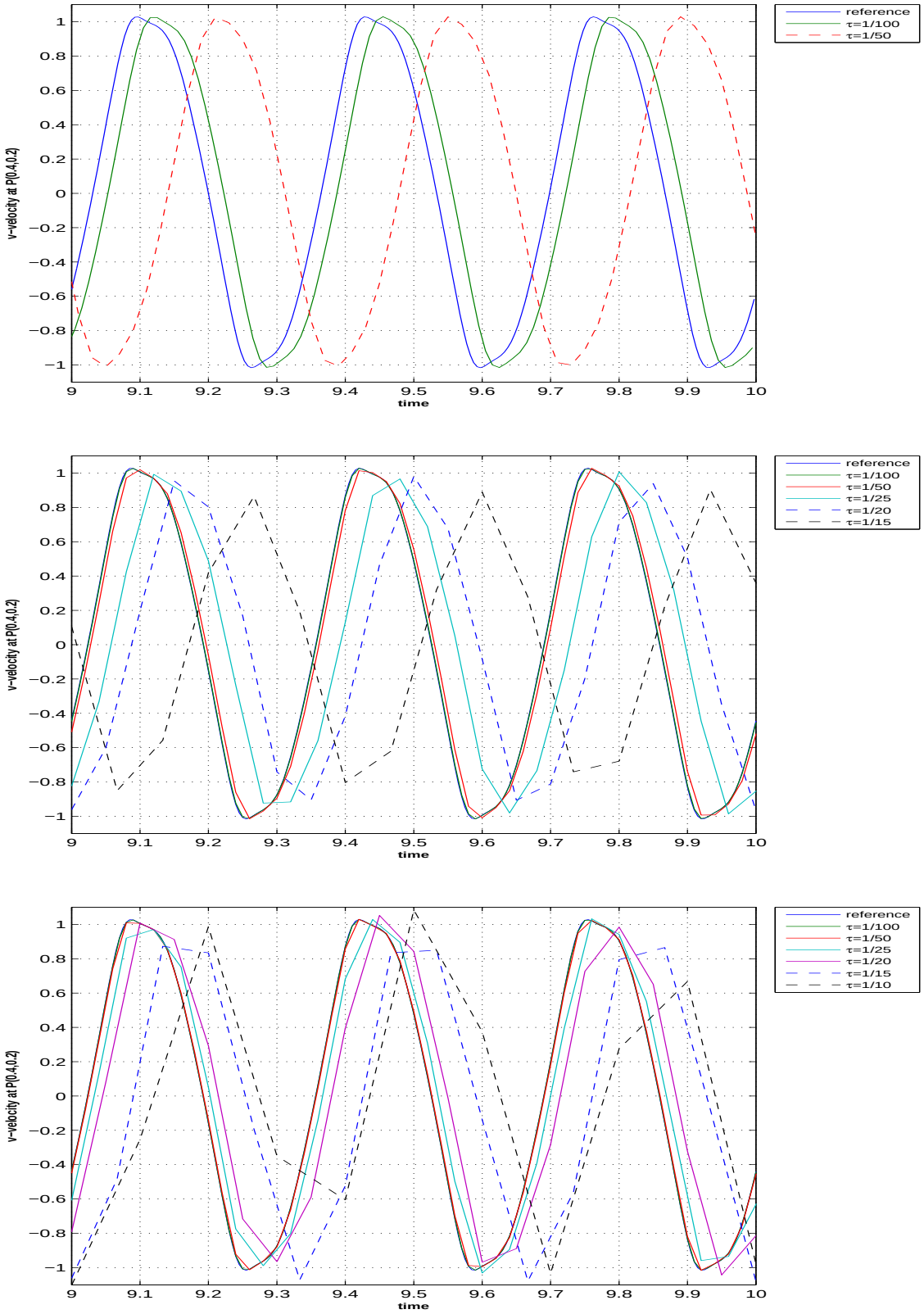



**Figure 10.10**: $v$-velocity at point $P(0.4, 0.2)$ for different $\tau$, using CN (top), dG(1) (middle) and cGP(2) (bottom) method for space level 2.

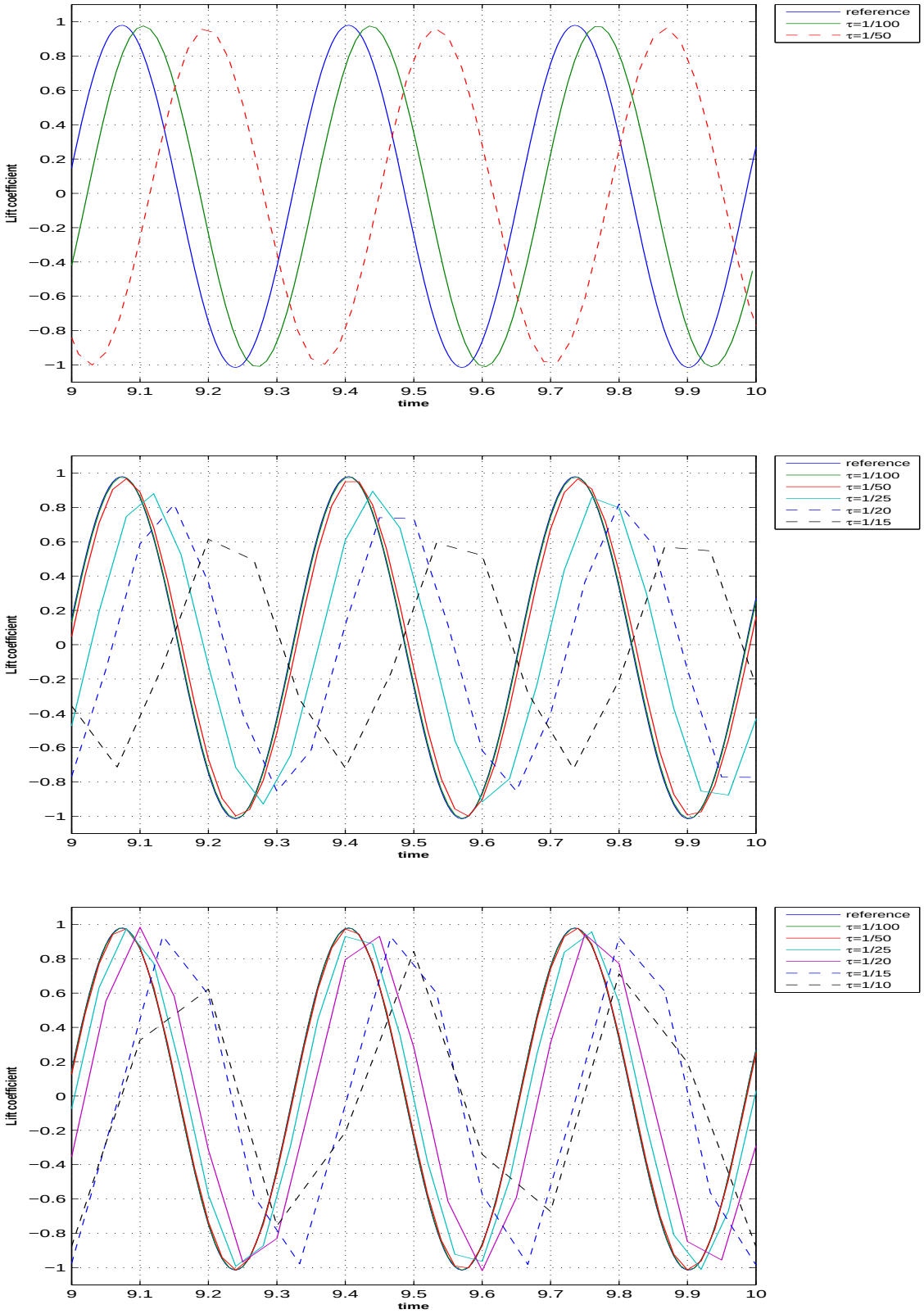



**Figure 10.11**: Lift coefficient for different $\tau$, using CN (top), dG(1) (middle) and cGP(2) (bottom) method for space level 3.

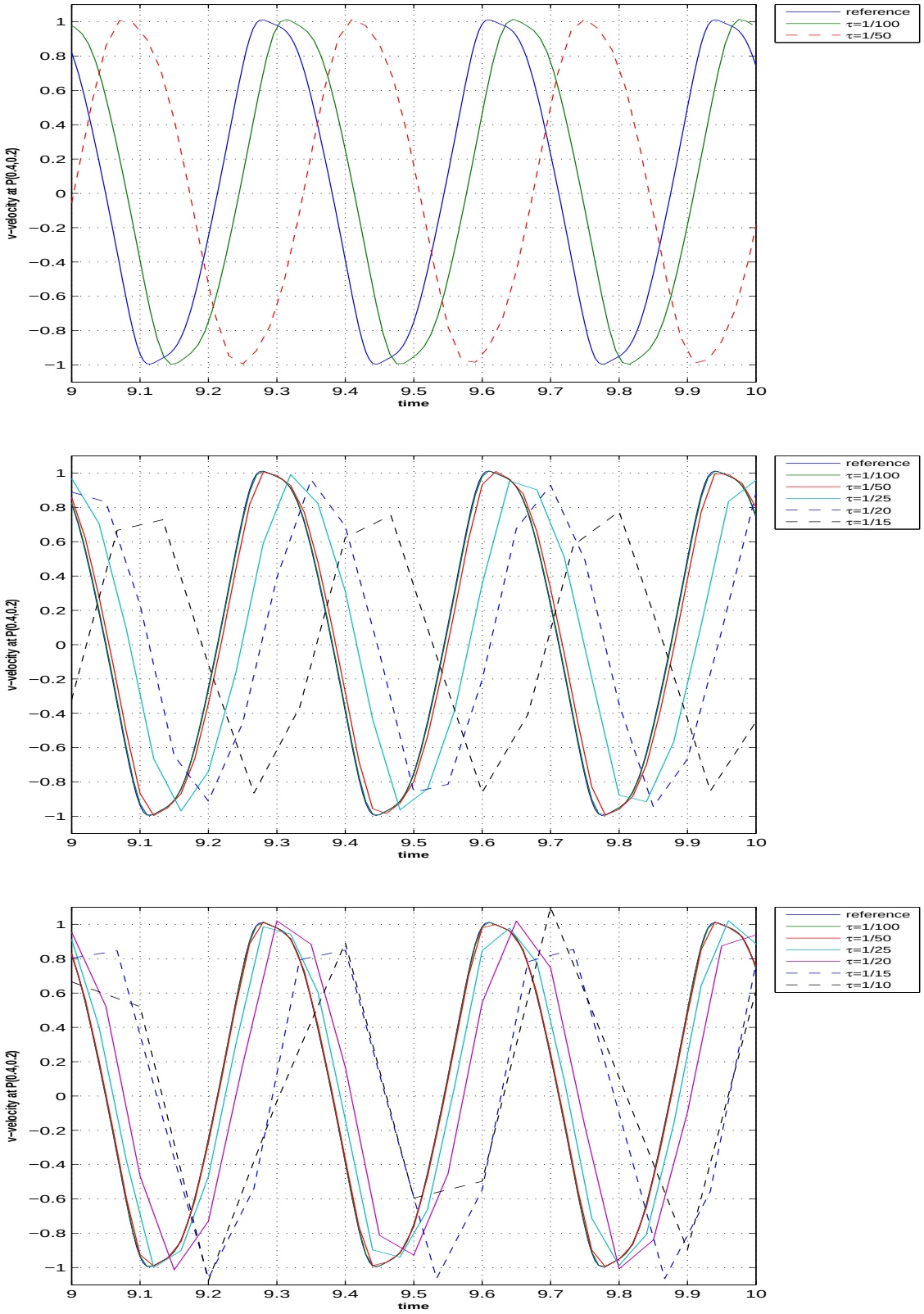



**Figure 10.12**: $v$-velocity at point $P(0.4, 0.2)$ for different $\tau$, using CN (top), dG(1) (middle) and cGP(2) (bottom) method for space level 3.

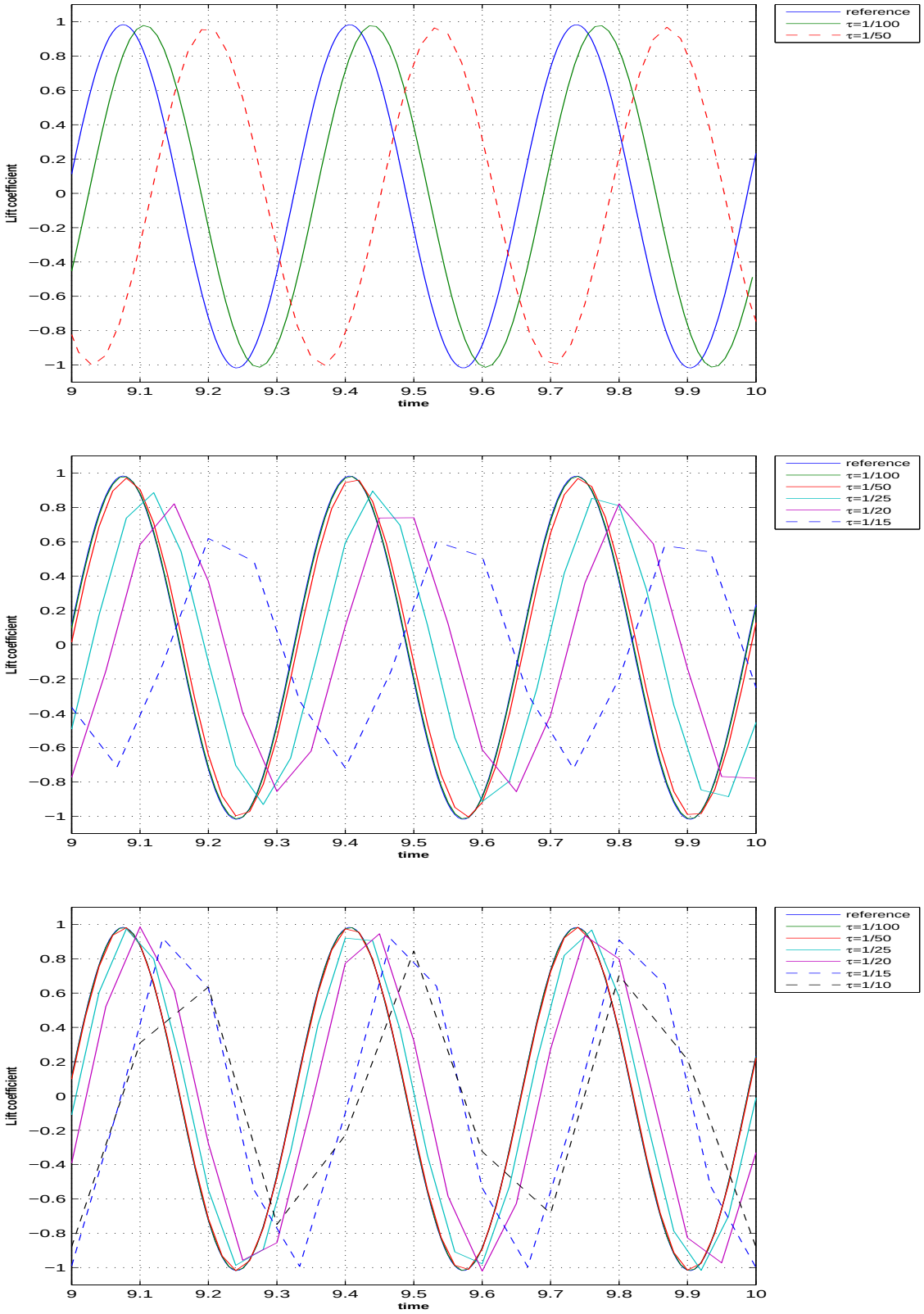



**Figure 10.13**: Lift coefficient for different $\tau$, using CN (top), dG(1) (middle) and cGP(2) (bottom) method for space level 4.

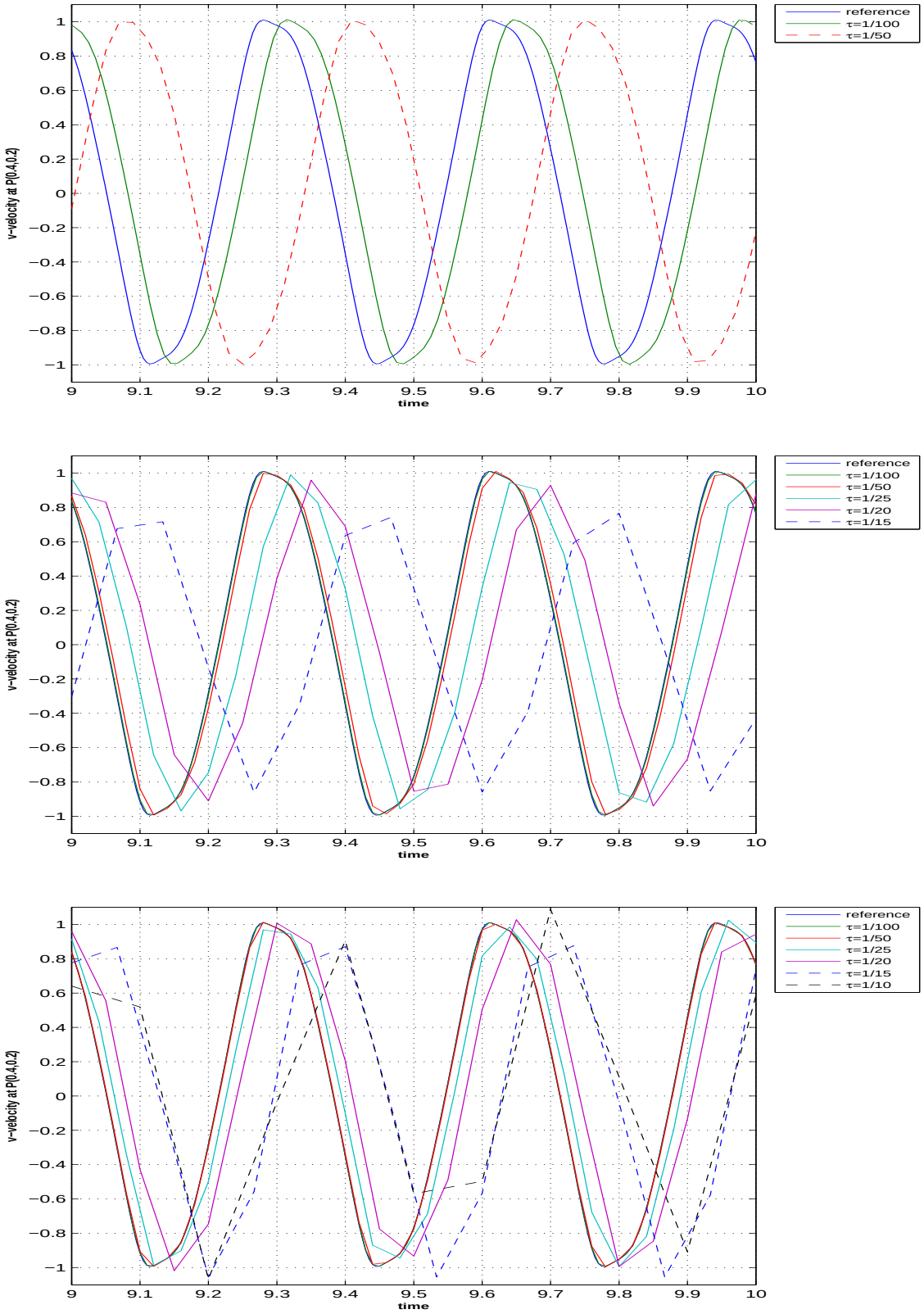



**Figure 10.14**: $v$-velocity at point $P(0.4, 0.2)$ for different $\tau$, using CN (top), dG(1) (middle) and cGP(2) (bottom) method for space level 4.

Next, we perform a more careful analysis on the basis of the plots already (partially) shown in Figure 10.3 to 10.14. To this end, we will mainly concentrate on the values of the lift coefficient $(C_L)$ because this quantity has the larger amplitude in comparison with others. Table 10.2 and 10.3 demonstrate the 'deviation in percentage of the curves per cycle' for the corresponding time step sizes from the reference values, i.e., $\frac{\Delta x}{30\times 0.33} \times 100\%$, where $\Delta x$ is the total deviation after T=10 (with length of period $\approx 0.33$, number of cycles until T=10 $\approx 30$). The reference values are taken from the higher order cGP(2) scheme with very small time step $\tau = 1/200$ (keep in mind that all tests started from the same start solution and that we perform approximately 30 oscillations until T=10). The different time discretization schemes are then compared in the sense that allows large time steps to gain the desired accuracy.

| $\tau$ | **CN** | **cGP(1)** | **cGP(2)** | **dG(1)** |
|---|---|---|---|---|
| 1/100 | 0.32% | 0.32% | 0.00% | 0.01% |
| 1/50 | 1.27% | 1.29% | 0.01% | 0.05% |
| 1/25 | | | 0.11% | 0.39% |
| 1/20 | | | 0.25% | 0.75% |
| 1/15 | | | 0.72% | 1.73% |
| 1/10 | | | 0.93% | |

**Table 10.2**: Change of the lift coefficient in percentage for space Level=2.

| $\tau$ | **CN** | **cGP(1)** | **cGP(2)** | **dG(1)** |
|---|---|---|---|---|
| 1/100 | 0.33% | 0.33% | 0.00% | 0.01% |
| 1/50 | 1.29% | 1.30% | 0.01% | 0.05% |
| 1/25 | | | 0.10% | 0.38% |
| 1/20 | | | 0.20% | 0.71% |
| 1/15 | | | 0.72% | 1.61% |
| 1/10 | | | 0.82% | |

**Table 10.3**: Change of the $v$-velocity at point P(0.4, 0.2) in percentage for space Level=2.

Table 10.4 and 10.5 depict the same percentage change for mesh level 3.

| $\tau$ | **CN** | **cGP(1)** | **cGP(2)** | **dG(1)** |
|---|---|---|---|---|
| 1/100 | 0.34% | 0.34% | 0.00% | 0.01% |
| 1/50 | 1.35% | 1.35% | 0.01% | 0.06% |
| 1/25 | | | 0.13% | 0.40% |
| 1/20 | | | 0.29% | 0.74% |
| 1/15 | | | 0.84% | 1.64% |
| 1/10 | | | 0.96% | |

**Table 10.4**: Change of the lift coefficient in percentage for space Level=3.

| τ | CN | cGP(1) | cGP(2) | dG(1) |
|---|---|---|---|---|
| 1/100 | 0.35% | 0.34% | 0.00% | 0.01% |
| 1/50 | 1.34% | 1.34% | 0.01% | 0.06% |
| 1/25 | | | 0.13% | 0.39% |
| 1/20 | | | 0.30% | 0.70% |
| 1/15 | | | 0.85% | 1.56% |
| 1/10 | | | 0.85% | |

**Table 10.5**: Change of the $v$-velocity at point P(0.4, 0.2) in percentage for space Level=3.

Table 10.6 and 10.7 depicts the same change for mesh level 4.

| τ | CN | cGP(1) | cGP(2) | dG(1) |
|---|---|---|---|---|
| 1/100 | 0.34% | 0.32% | 0.00% | 0.01% |
| 1/50 | 1.35% | 1.33% | 0.01% | 0.05% |
| 1/25 | | | 0.12% | 0.39% |
| 1/20 | | | 0.29% | 0.71% |
| 1/15 | | | 0.85% | 1.61% |
| 1/10 | | | 0.98% | |

**Table 10.6**: Change of the lift coefficient in percentage for space Level=4.

| τ | CN | cGP(1) | cGP(2) | dG(1) |
|---|---|---|---|---|
| 1/100 | 0.34% | 0.32% | 0.00% | 0.01% |
| 1/50 | 1.33% | 1.32% | 0.01% | 0.05% |
| 1/25 | | | 0.12% | 0.37% |
| 1/20 | | | 0.29% | 0.68% |
| 1/15 | | | 0.85% | 1.51% |
| 1/10 | | | 0.85% | |

**Table 10.7**: Change of the $v$-velocity at point P(0.4, 0.2) in percentage for space Level=4.

In the next Table 10.8, we conclude the maximum allowed time step sizes to gain comparable results with an error of less than 1% per period at a given space level for the corresponding time discretization schemes.

| Lev | cGP(1) | cGP(2) | dG(1) |
|---|---|---|---|
| 2 | 1/100 | 1/10 | 1/20 |
| 3 | 1/100 | 1/10 | 1/20 |
| 4 | 1/100 | 1/10 | 1/20 |
| factor | 10 | 1 | 2 |

**Table 10.8**: Maximum allowed timestep sizes to obtain errors of less than 1% per period at given space level for cGP(1) vs. cGP(2) vs. dG(1).

Summarizing the results from Table 10.8, we have seen that the corresponding time step sizes

to gain the accuracy with an error of less than 1% per period for the cGP(2) are two times larger than the dG(1)-method and ten times larger than the cGP(1)-method.

Next, we want to show all the presented time discretization schemes with corresponding time step sizes for which the lift coefficient $(C_L)$ values are almost identical $(\leq 0.5\%)$ after T=10, resp., approx 30 periods. Figure 10.15 depicts the corresponding lift coefficients, for 10 time units. We also zoom these quantities in the last time unit to see the (very small) differences more clearly in Figure 10.16.

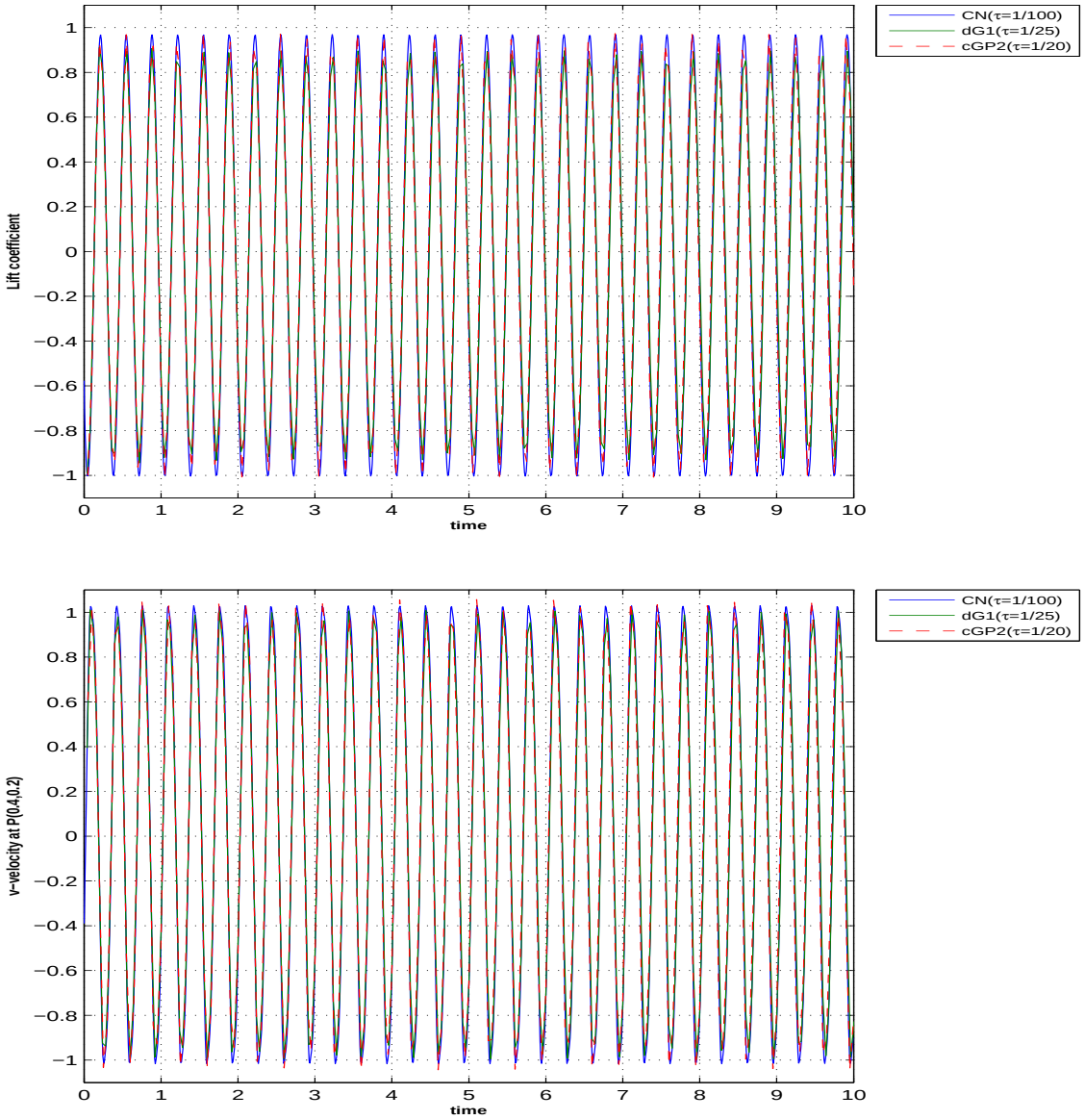



**Figure 10.15**: Lift coefficient/$v$-velocity at point $P(0.4, 0.2)$ for CN vs. dG(1) vs. cGP(2) at space level 2.

Finally, we show how the solution components look like for these time step sizes corresponding to Figure 10.15 and 10.16.

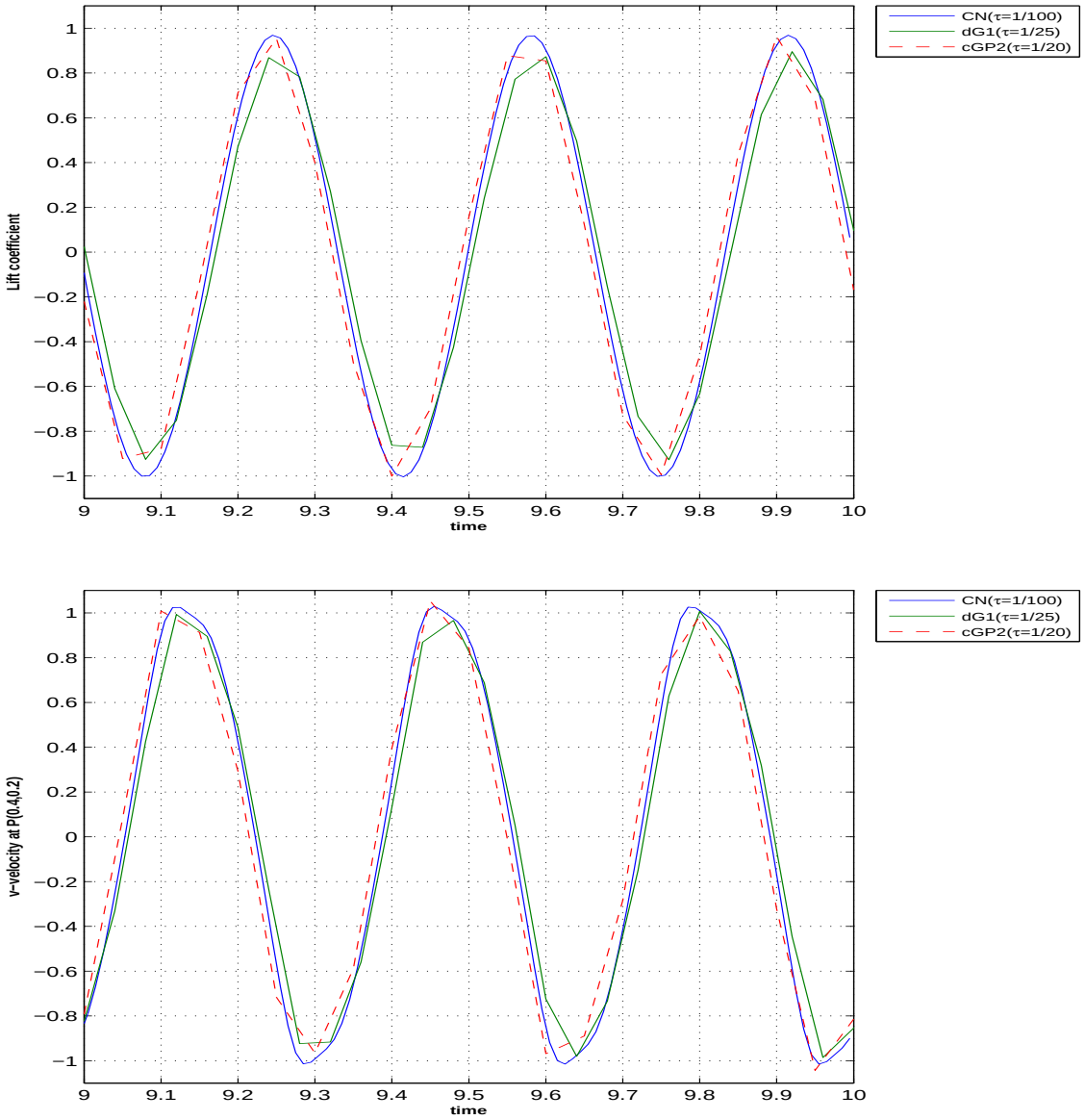



**Figure 10.16**: Lift coefficient/$v$-velocity at point $P(0.4, 0.2)$ for CN vs. dG(1) vs. cGP(2) at space level 2.

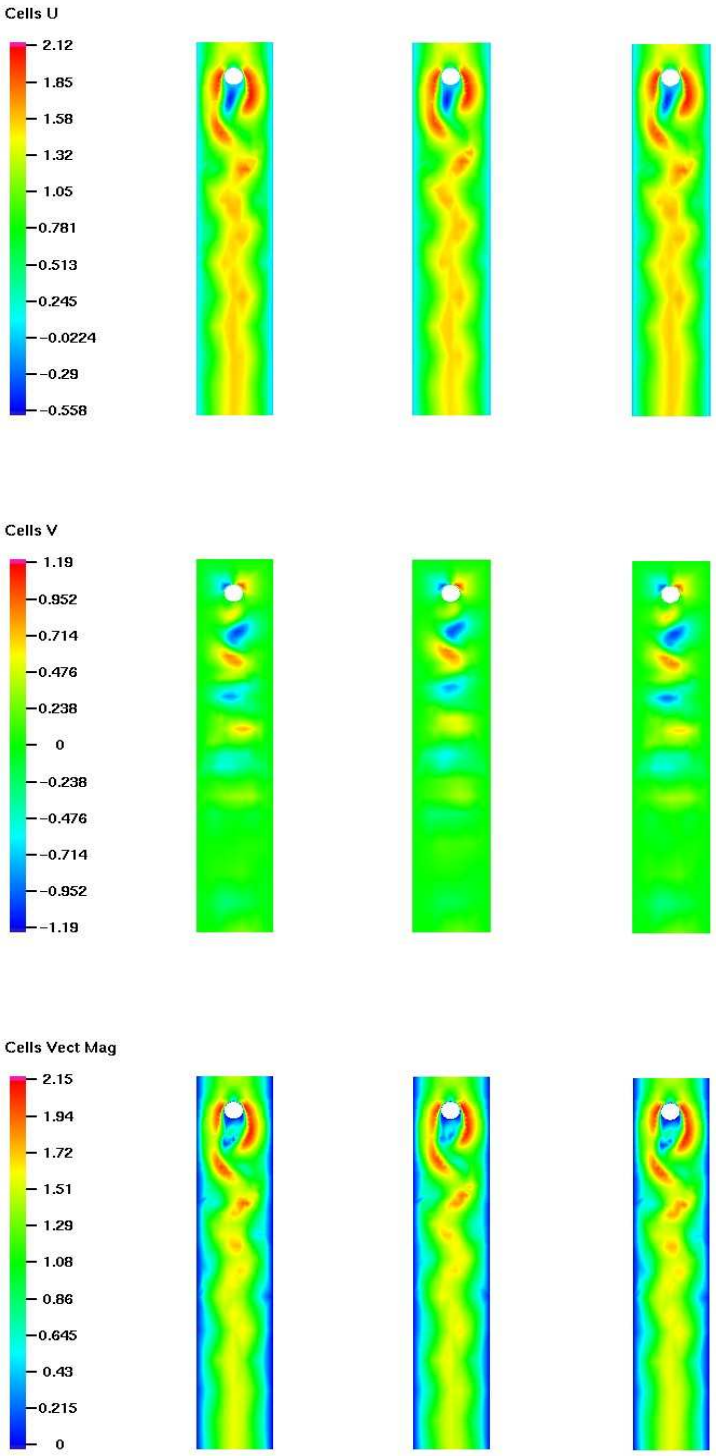



**Figure 10.17**: Visualization of the velocity $u, v$ and magnitude for CN (left), dG(1) (center) and cGP(2) (right) method with timesteps $\tau = 1/100, 1/25, 1/20$, respectively.

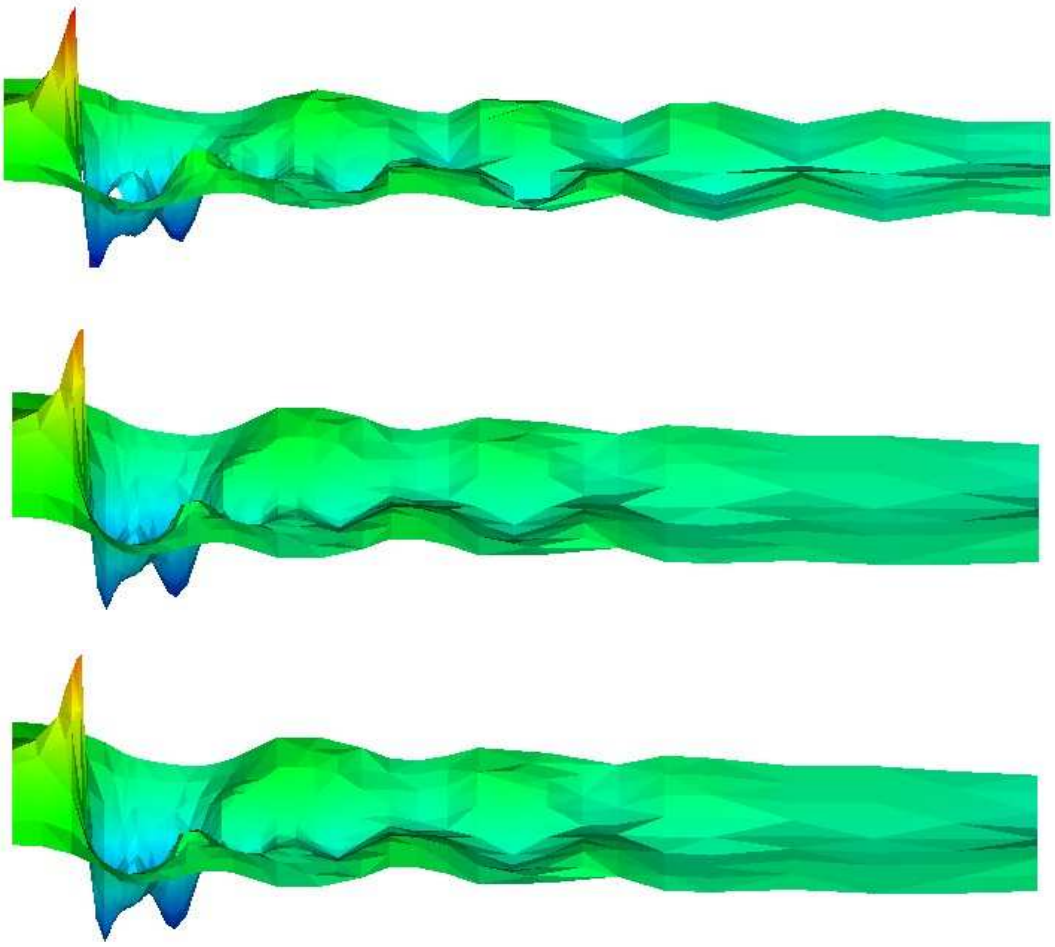


**Figure 10.18**: Visualization of pressure in flow around cylinder for CN (top), dG(1) (center) and cGP(2)-method (bottom) with timesteps $\tau = 1/100, 1/25, 1/20$, respectively.

## 10.2. Nonstationary flow through a Venturi pipe

The test configuration for the flow through a Venturi pipe which is considered here is slightly changed from the framework which has been already used in [57, 58]. Figure 10.19 shows the geometry and the coarse mesh (level 1) used for this simulation. The coarse mesh is recursively refined by joining opposite midpoints. The total length of the Venturi pipe is $L = 42$, the height of the Venturi pipe at the in/outlet is $H = 5$, the height in the most narrowing part is $H_i = 1$ and the width of the small upper channel is $W_i = 0.8$. The upper, lower walls of the pipe and the sides of the small upper channel are subjected to the no slip boundary conditions. At the inlet (left part of the boundary), an inflow of constant velocity $U = 1$ is prescribed while natural boundary conditions are prescribed at the outlet (right part of the boundary) and at the small upper in/outlet. The value of the kinematic viscosity is set to $\nu = 10^{-2}$ and the density of the fluid $\rho = 1$.

The *Reynolds Re number* determining the flow properties may be defined as

$$Re = \frac{\overline{U} H_i}{\nu},$$

where $\overline{U}$ is the maximum velocity through the narrow section in the pipe and $L$ is the height of this section. The resulting maximum velocity $\overline{U} \approx 7.0$ yields $Re \approx 700$. The resulting *Reynolds number* produces complex flow patterns which are oscillating in space and time. As we explained before, the aim of the simulation is to control the flux through the upper channel. Beside this interesting flow quantity we also compute the $v$-velocity at the point $P(16.0, 5.4)$ (top of the small channel) and $P(27.05, 2.5)$ (right of the pipe) and the pressure at the point $P(16.0, 2.5)$ in the middle of the pipe to compare the accuracy of all the presented time discretization schemes. Figure 10.20 gives an overview of the size of the problem on different space mesh levels where the finite element discretization is carried out by using the biquadratic $Q_2$-element for the velocity and discontinuous $P_1$-element for the pressure. Here, we employ the edge oriented jump FEM stabilization approach (see [59]) with stabilization parameter $\gamma = 0.1$. In order to compare the accuracy of different time

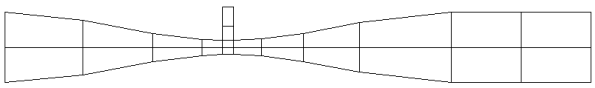


**Figure 10.19**: Coarse mesh for the venturi pipe flow

| Lev. | #EL | #DOF |
|---|---|---|
| 3 | 384 | 4 466 |
| 4 | 1 536 | 17 378 |
| 5 | 6 144 | 68 546 |
| 6 | 24 576 | 272 258 |

**Figure 10.20**: Size of the different systems in space.

discretizations, the flow is started on each mesh level from the corresponding Stokes solution at time $t = 0$, and the simulation is performed until T=30 using different time discretization methods for different time step sizes $\tau$. At T=30, all the quantities of interest are plotted and analyzed in detail. Since the results obtained from Crank-Nicolson and cGP(1)-method are almost identical as expected, therefore we show again the results for cGP(1) only together with the cGP(2) and dG(1)-method.

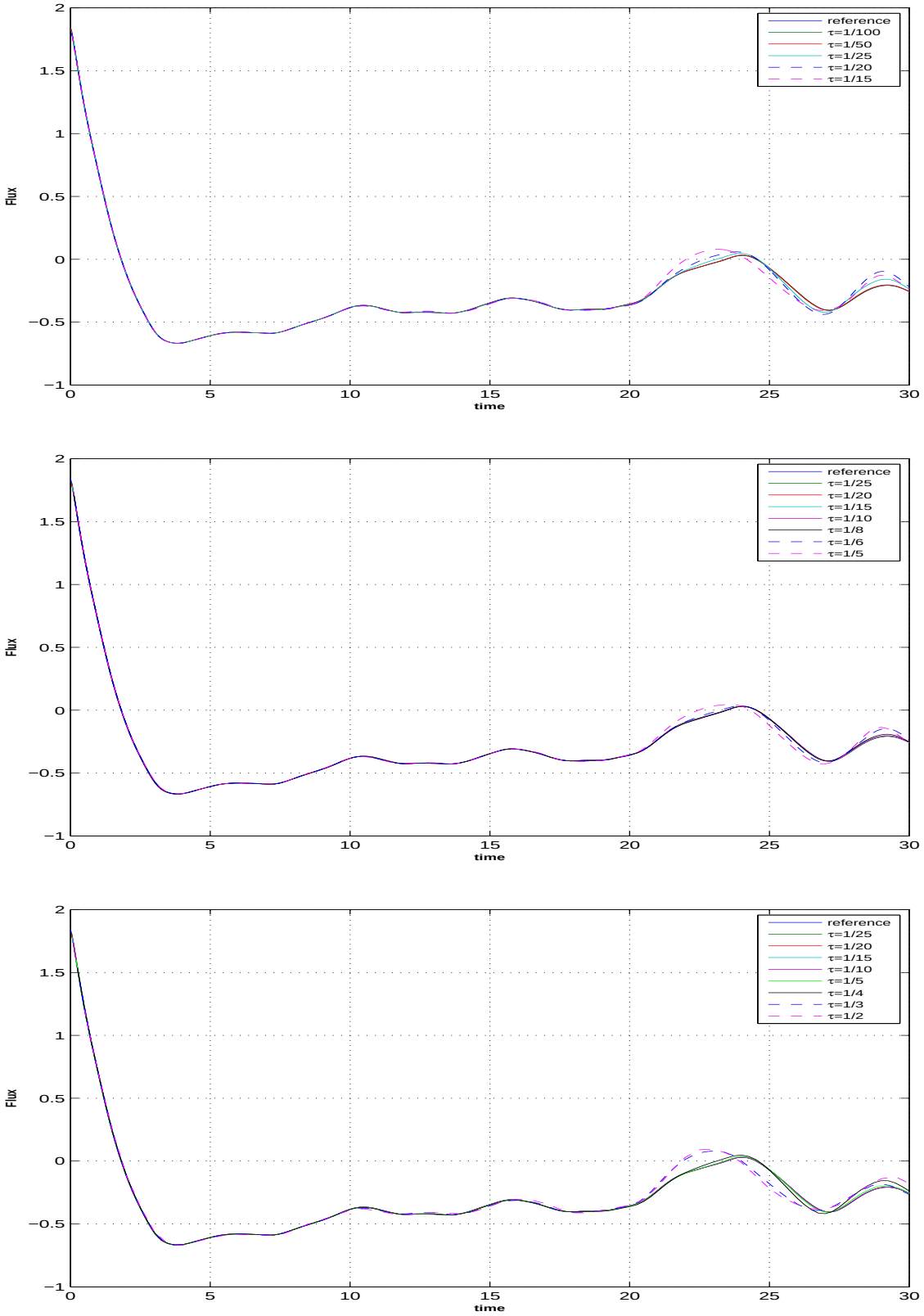



**Figure 10.21**: Flux through the upper inlet/outlet, using the cGP(1) (top), dG(1) (middle) and cGP(2) (bottom) method for space level 3.

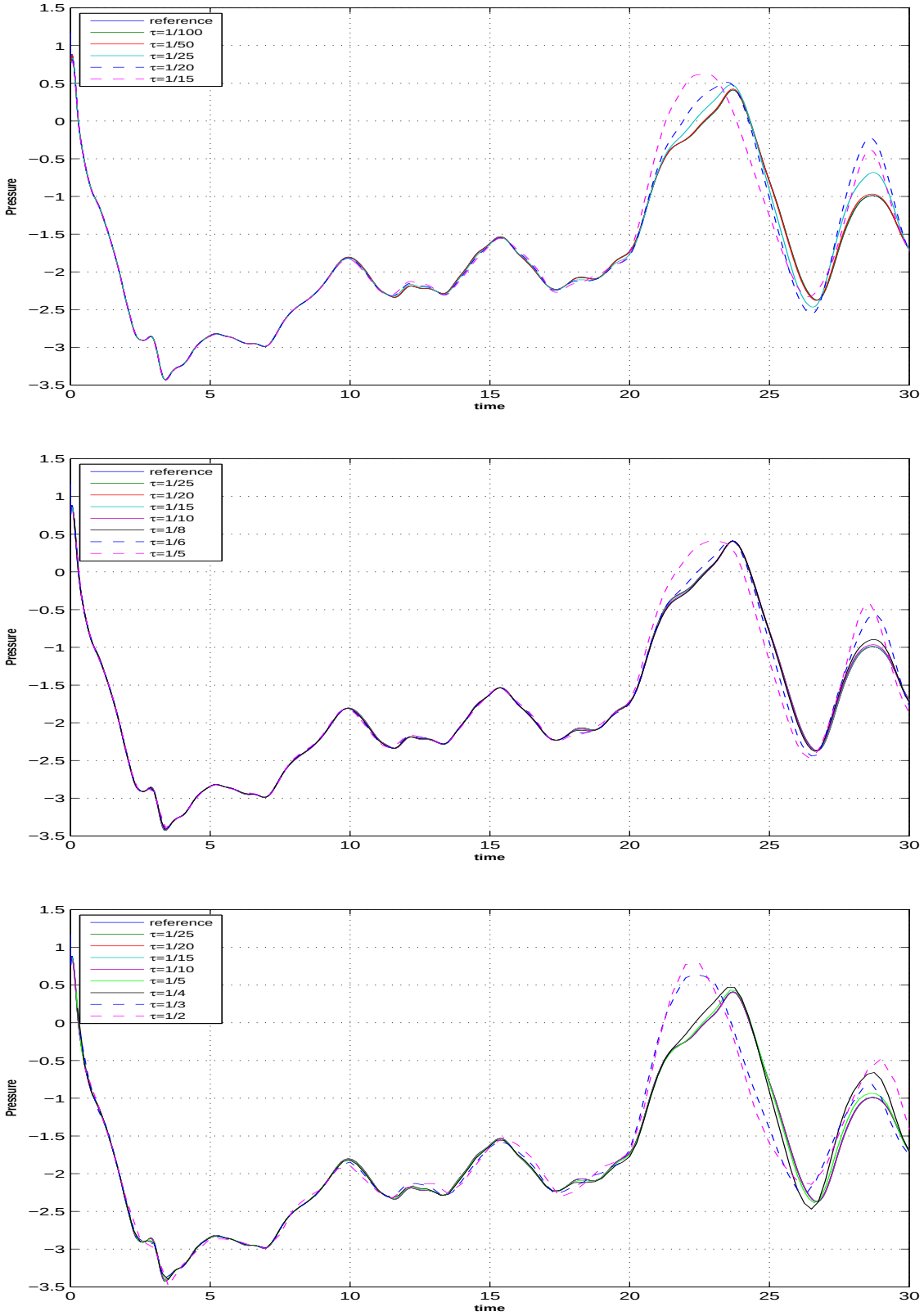



**Figure 10.22**: Pressure at point $P(16.0, 2.5)$, using the cGP(1) (top), dG(1) (middle) and cGP(2) (bottom) method for space level 3.

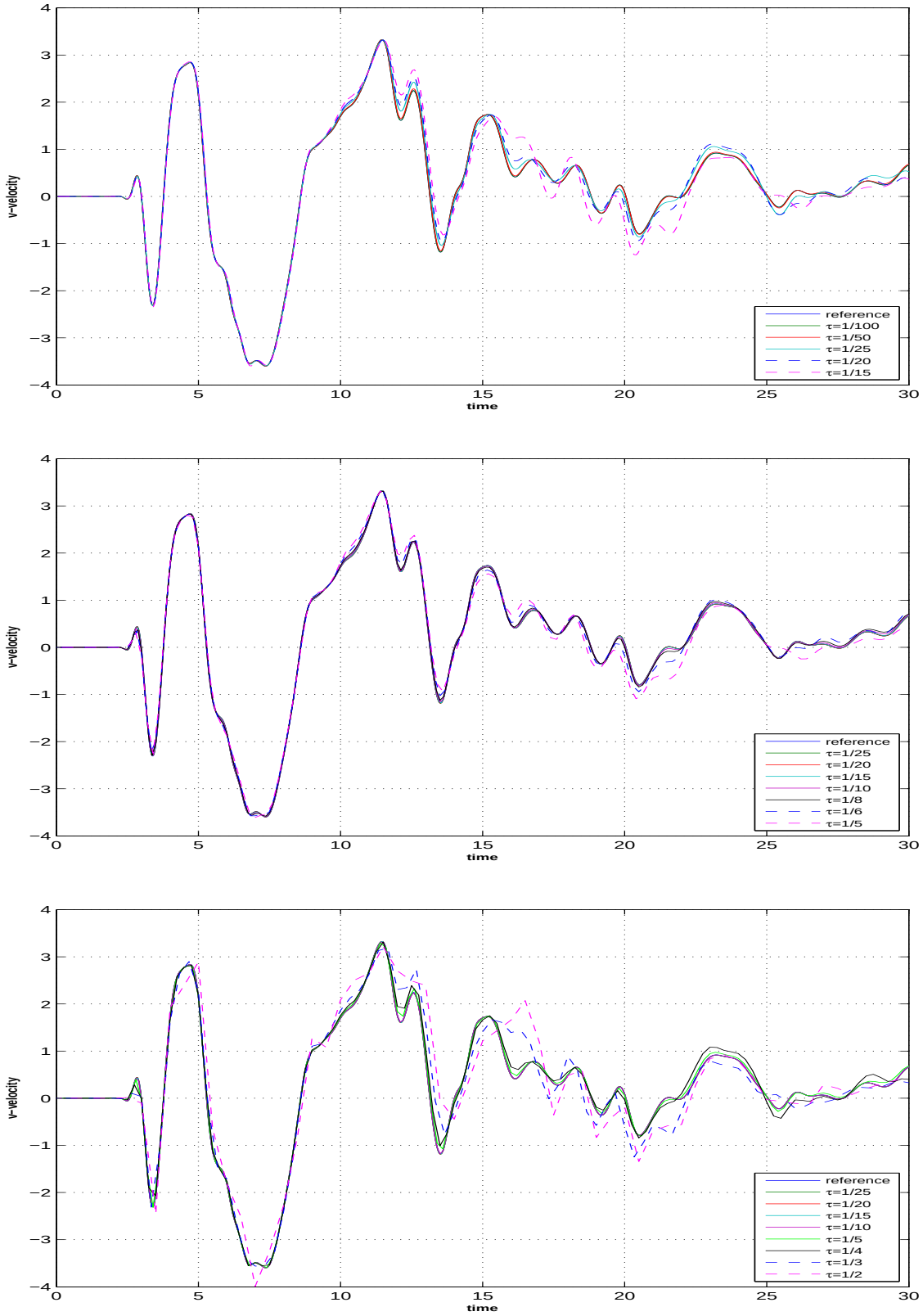



**Figure 10.23**: $v$-velocity at point $P(27.05, 2.5)$, using the cGP(1) (top), dG(1) (middle) and cGP(2) (bottom) method for space level 3.

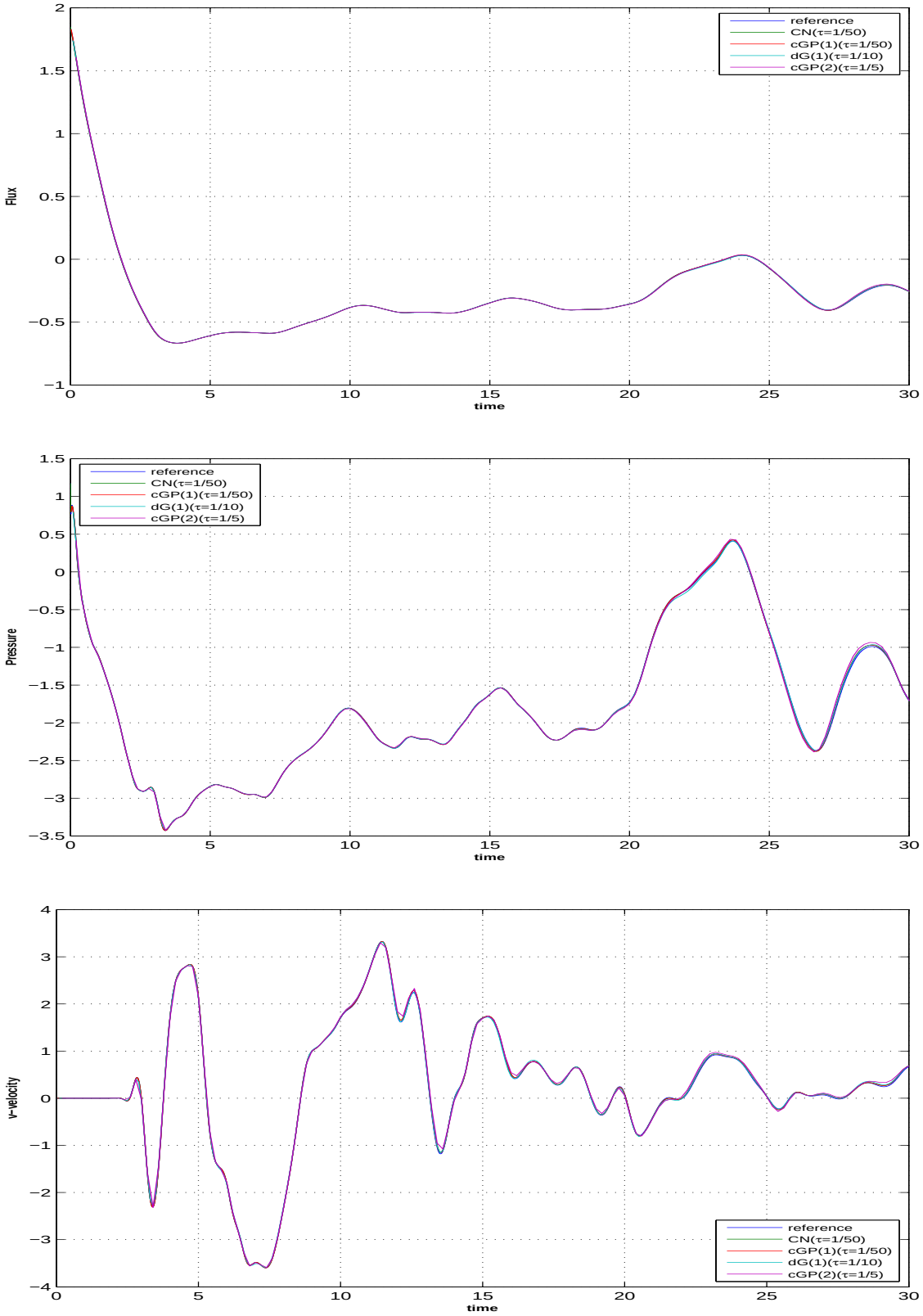



**Figure 10.24**: Flux (top)/pressure (middle)/$v$-velocity (bottom) at point $P(27.05, 2.5)$, using the CN, cGP(1), dG(1) and cGP(2)-method with max. allowed time steps for space level 3.

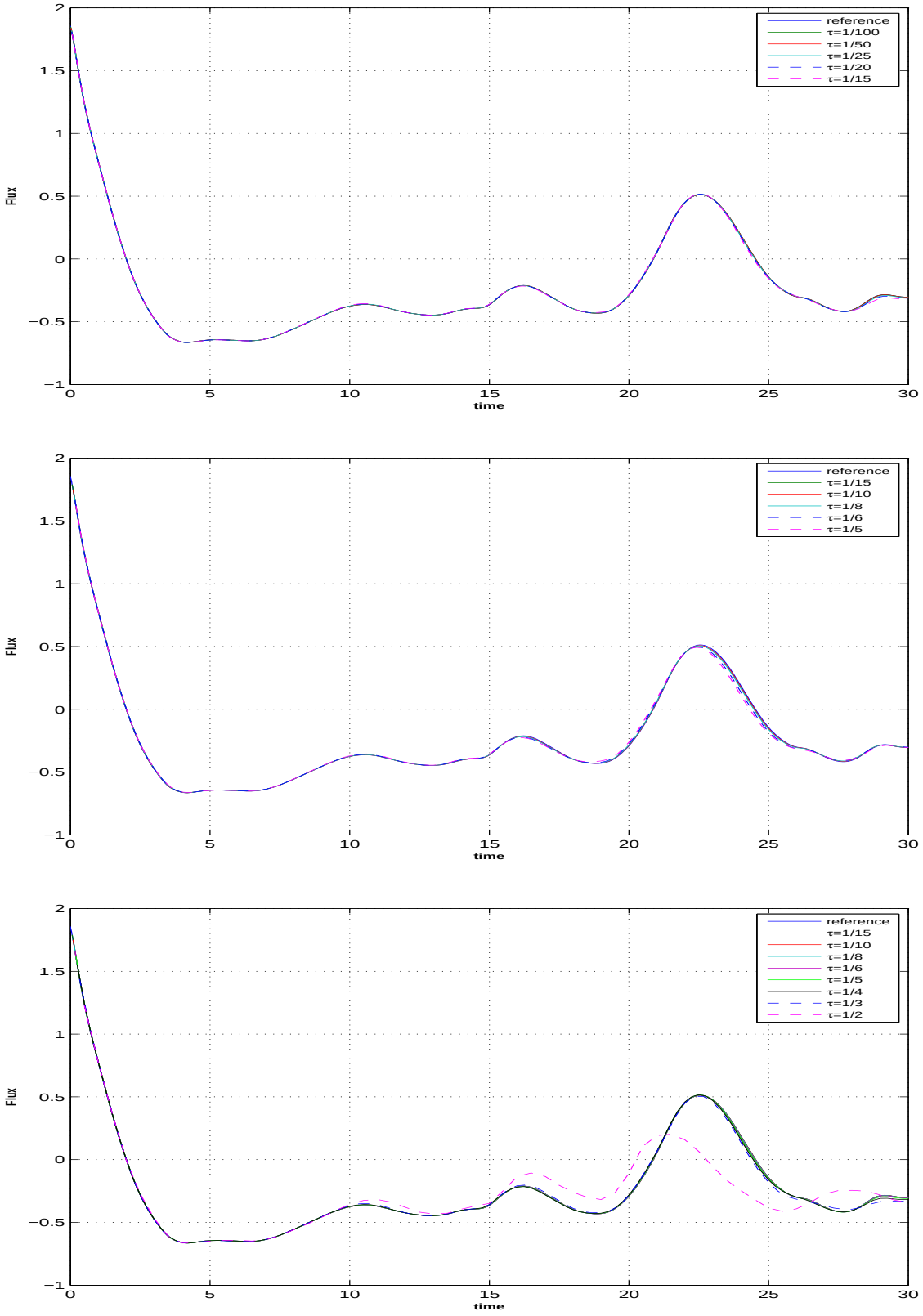



**Figure 10.25**: Flux through the upper inlet/outlet, using the cGP(1) (top), dG(1) (middle) and cGP(2) (bottom) method for space level 4.

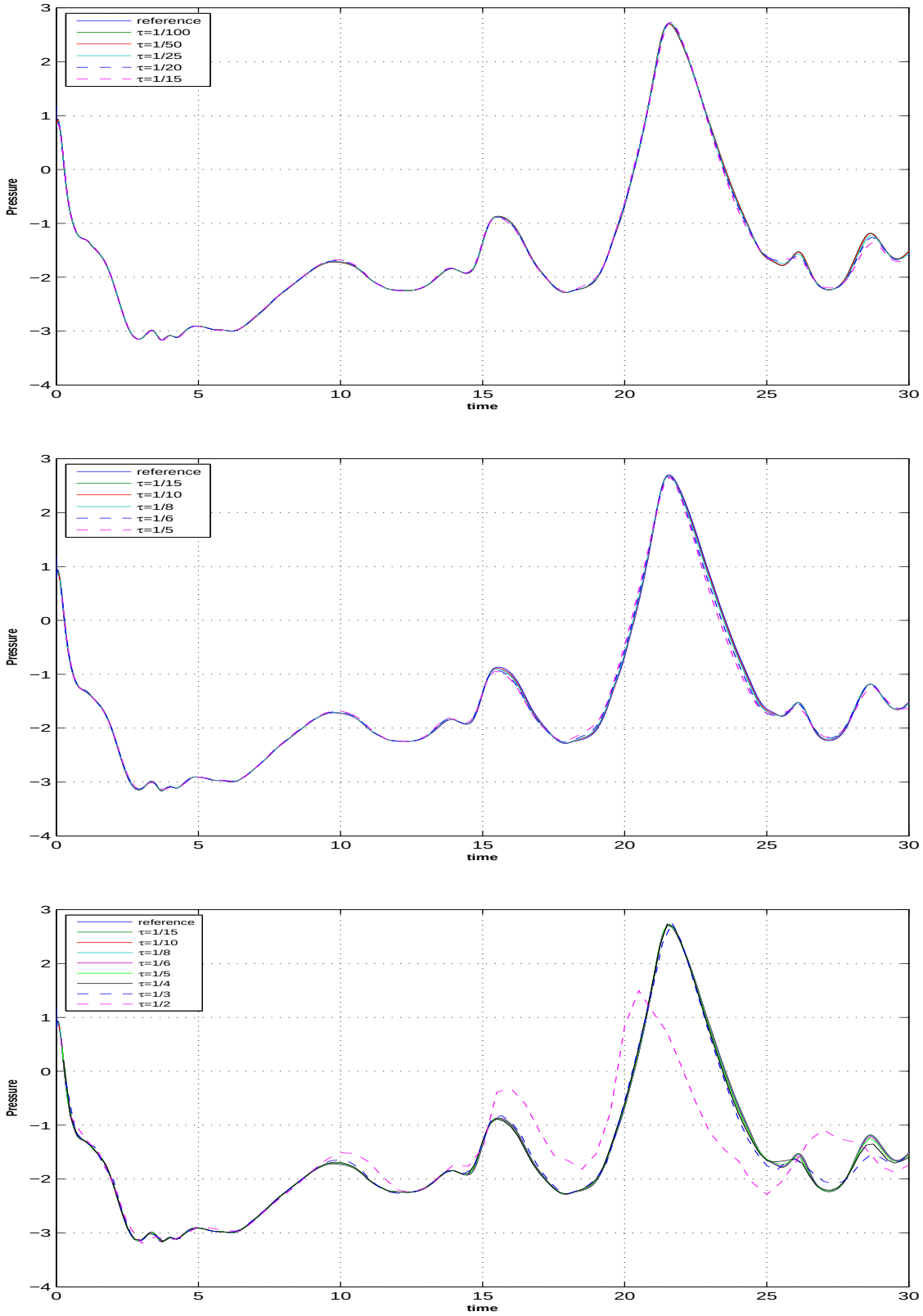



**Figure 10.26**: Pressure at point $P(16.0, 2.5)$, using the cGP(1) (top), dG(1) (middle) and cGP(2) (bottom) method for space level 4.

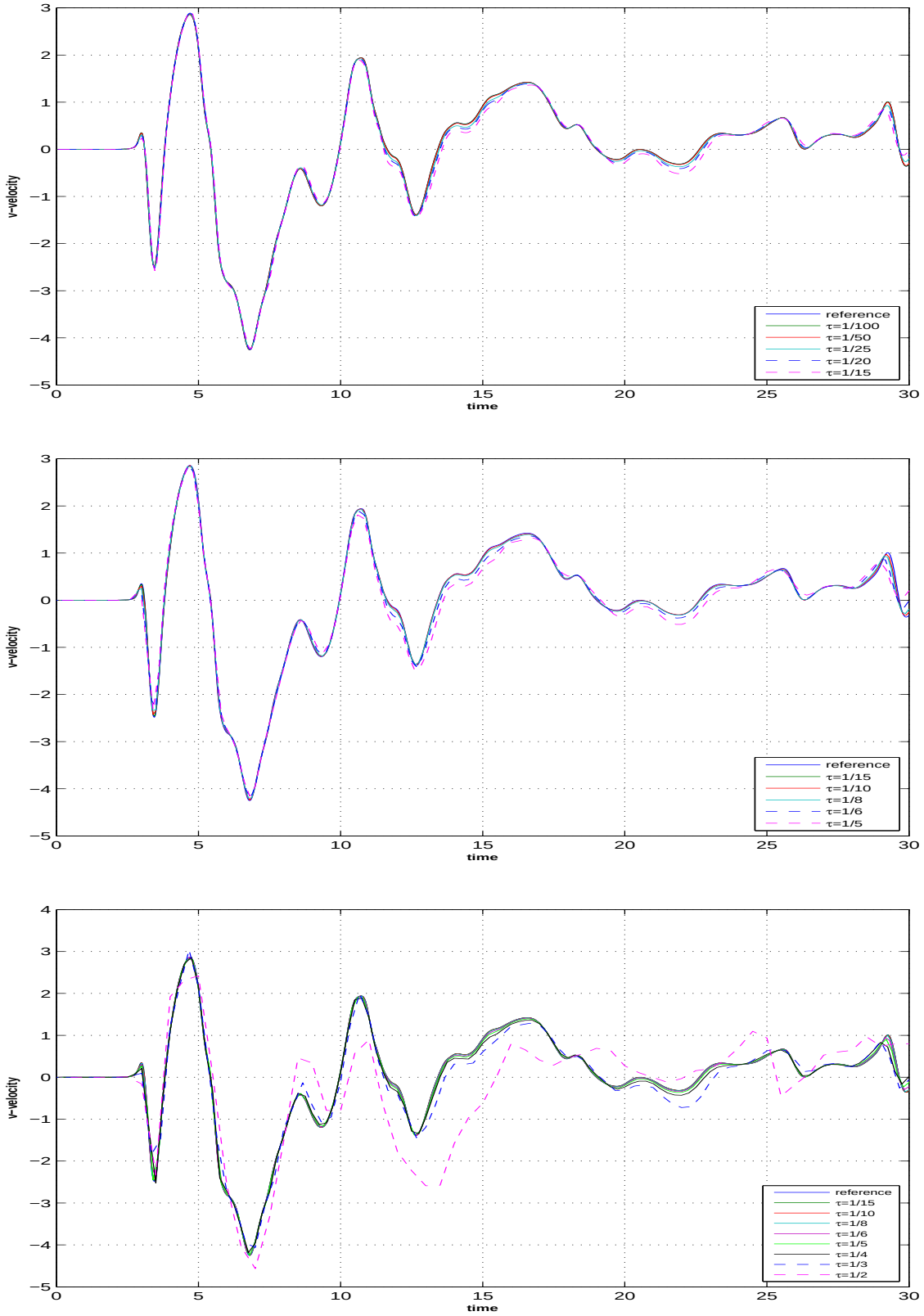



**Figure 10.27**: $v$-velocity at point $P(27.05, 2.5)$, using the cGP(1) (top), dG(1) (middle) and cGP(2) (bottom) method for space level 4.

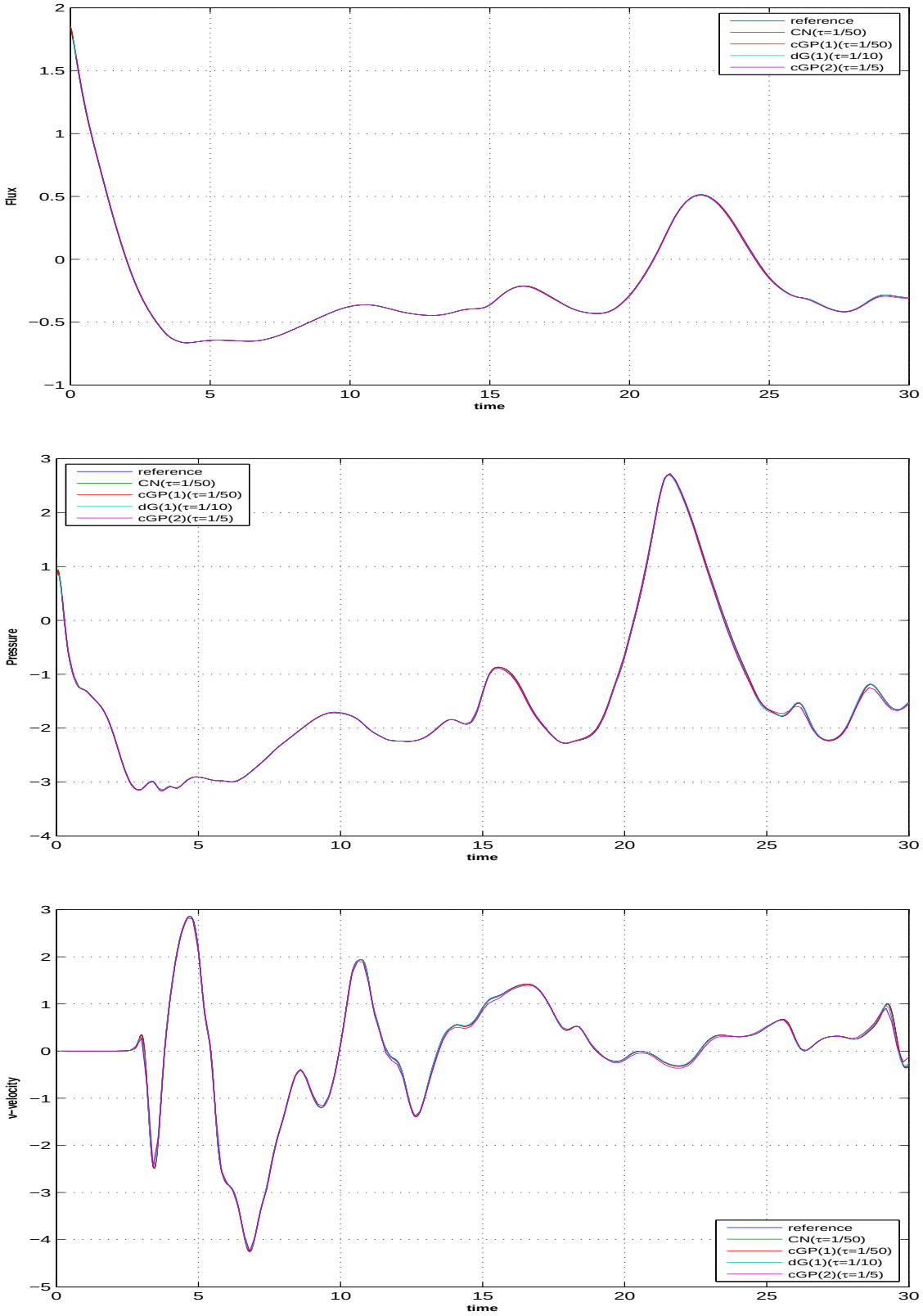



**Figure 10.28**: Flux (top)/pressure (middle)/$v$-velocity (bottom) at point $P(27.05, 2.5)$, using the CN, cGP(1), dG(1) and cGP(2)-method with max. allowed time steps for space level 4.

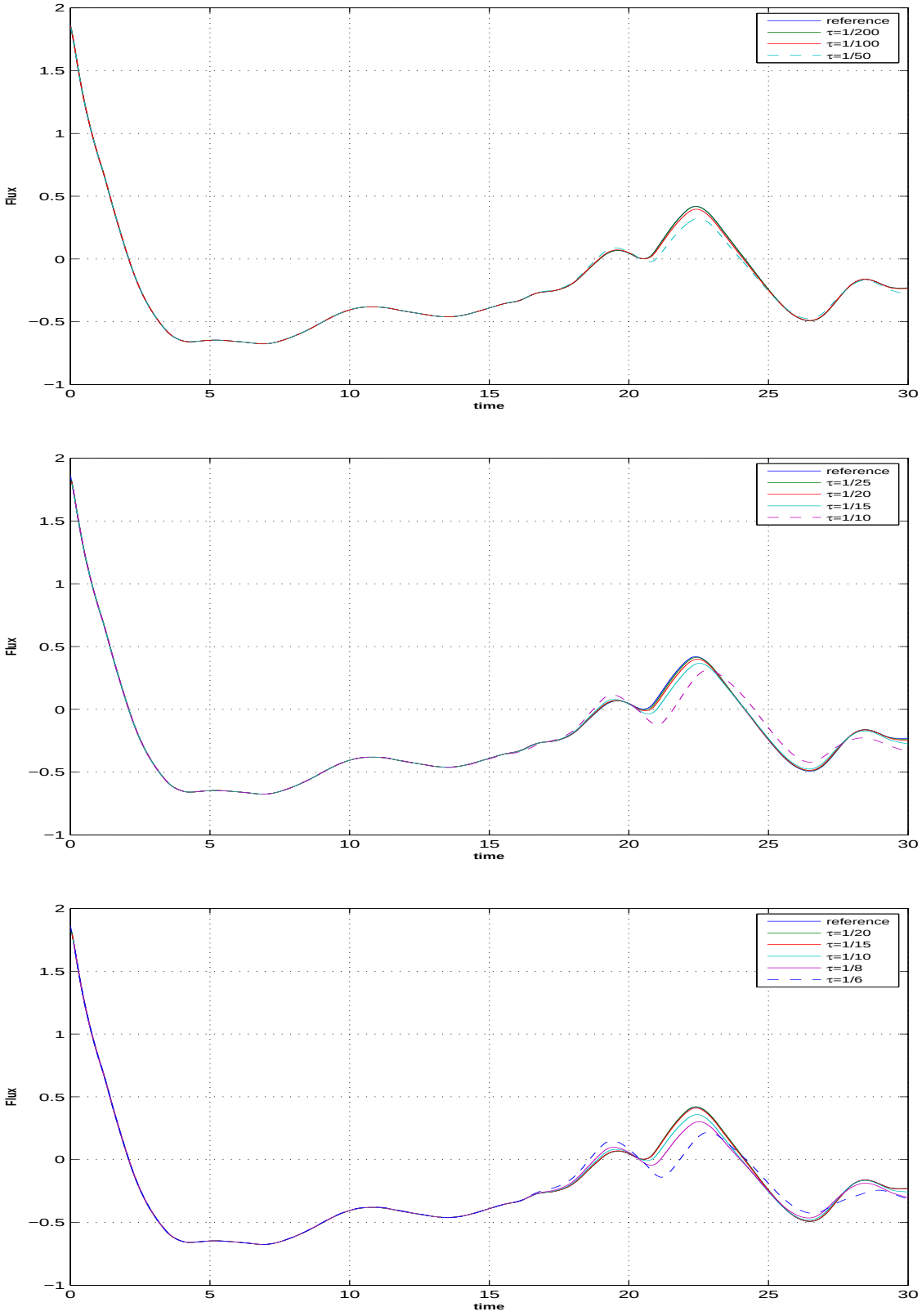



**Figure 10.29**: Flux through the upper inlet/outlet, using the cGP(1) (top), dG(1) (middle) and cGP(2) (bottom) method for space level 5.

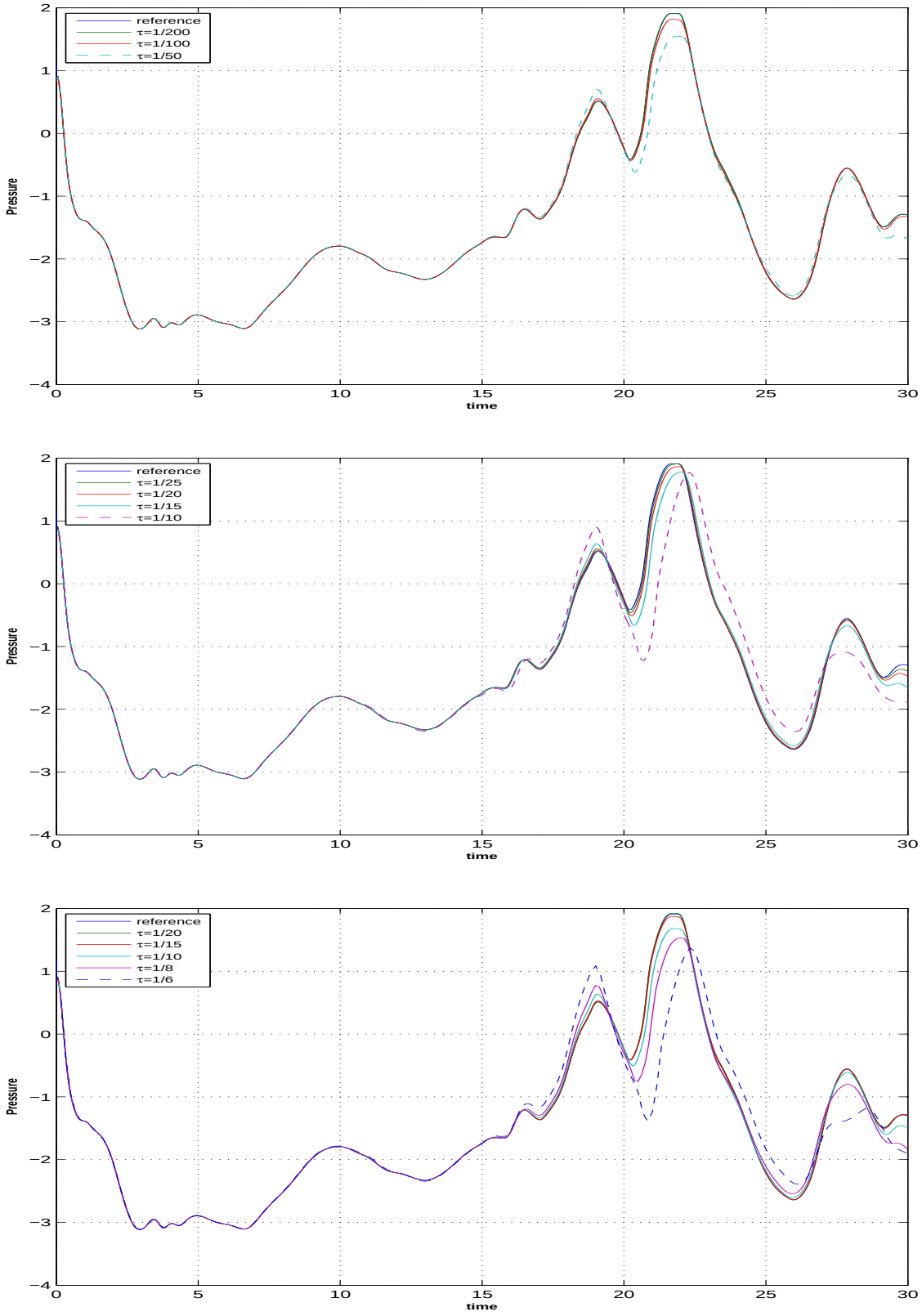



**Figure 10.30**: Pressure at point $P(16.0, 2.5)$, using the cGP(1) (top), dG(1) (middle) and cGP(2) (bottom) method for space level 5.

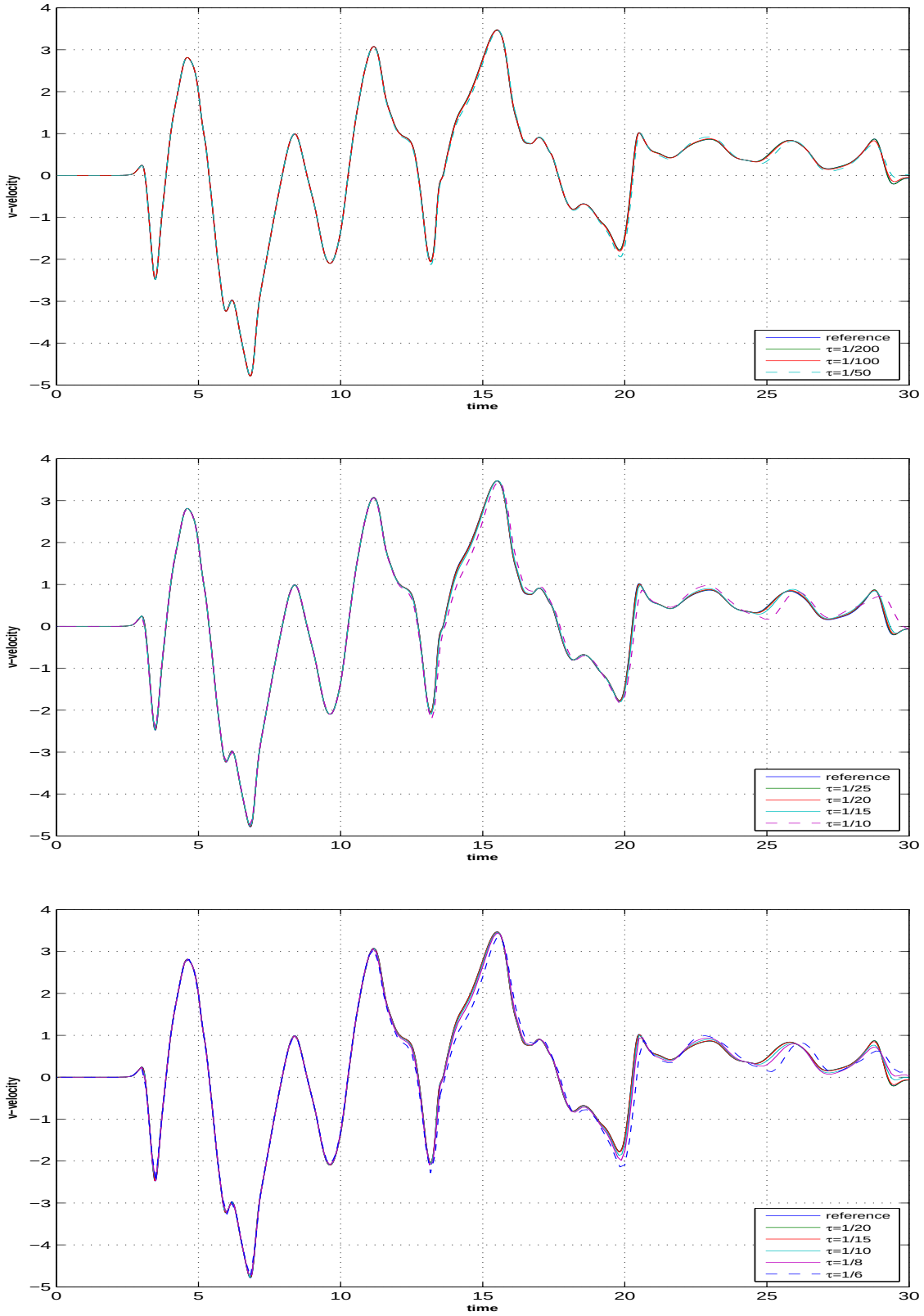



**Figure 10.31**: $v$-velocity at point $P(27.05, 2.5)$, using the cGP(1) (top), dG(1) (middle) and cGP(2) (bottom) method for space level 5.

We observe from Table 10.21 to 10.31 that the cGP(2)-method captures the dynamics of the flow at quite large time step sizes as expected. The results here on different mesh levels look somewhat more different due to the higher Reynolds number. As in the case of the cylinder before, we demonstrate in the same way the maximum allowed time step sizes which lead to very similar results in the 'picture norm'. Table 10.9 shows these time step sizes for different space mesh levels.

| **Lev** | **cGP(1)** | **cGP(2)** | **dG(1)** |
|---|---|---|---|
| 3 | 1/50 | 1/5 | 1/10 |
| 4 | 1/50 | 1/5 | 1/10 |
| 5 | 1/100 | 1/10 | 1/20 |
| factor | 10 | 1 | 2 |

**Table 10.9**: Maximum allowed timestep sizes which lead (almost) to same results.

Accordingly, we show the results associated to the time steps in Table 10.32. Finally, we demonstrate how the solution patterns develop in the last 13 time units from T=18 to T=30. Figure 10.33,10.34,10.35 and 10.36 illustrate the velocity $u, v$, magnitude and the pressure, respectively.

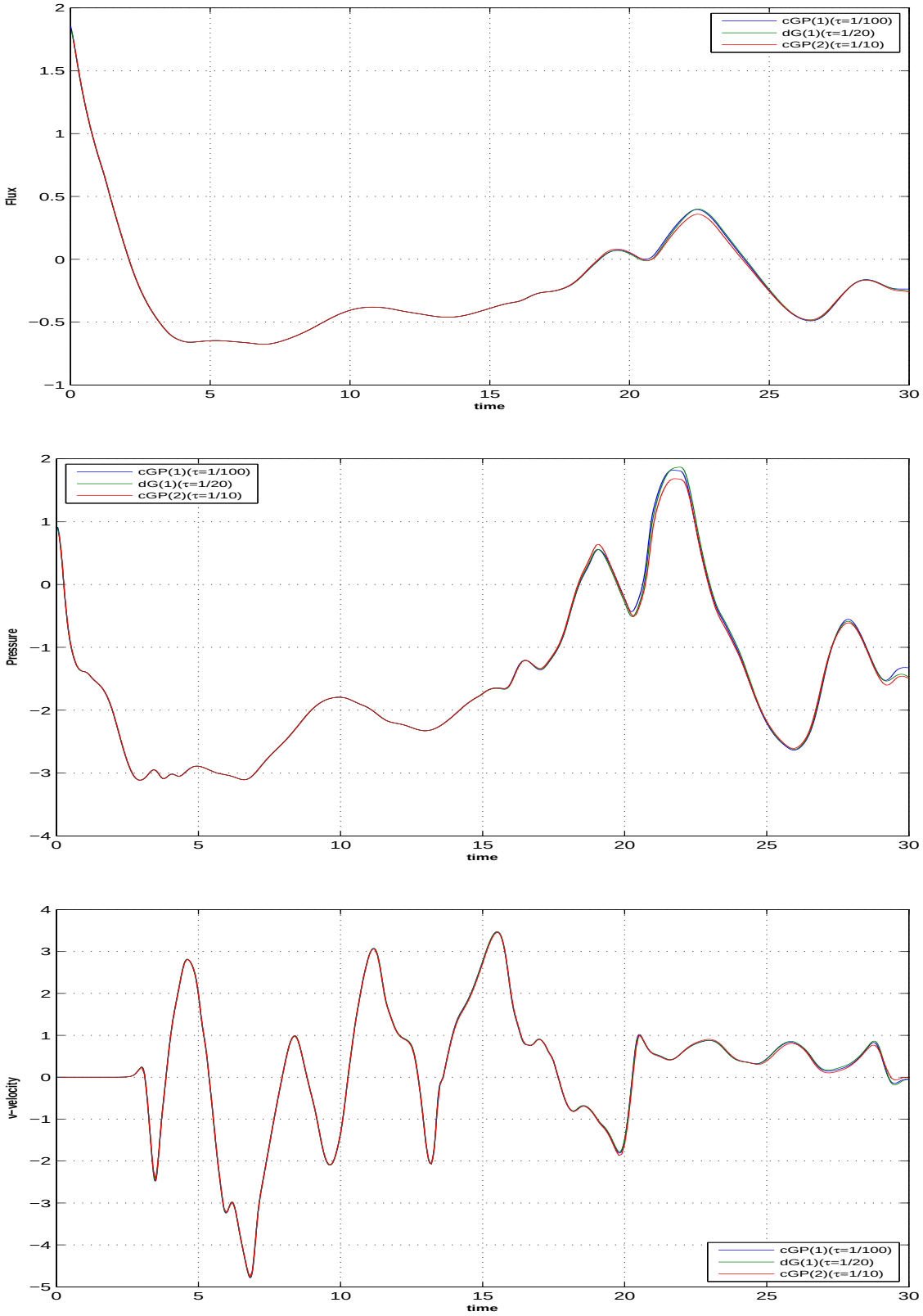



**Figure 10.32**: Flux (top)/pressure (middle)/$v$-velocity (bottom) at point $P(27.05, 2.5)$ , using the cGP(1) vs. dG(1) vs. cGP(2)-method with max. allowed time steps for space level 5.

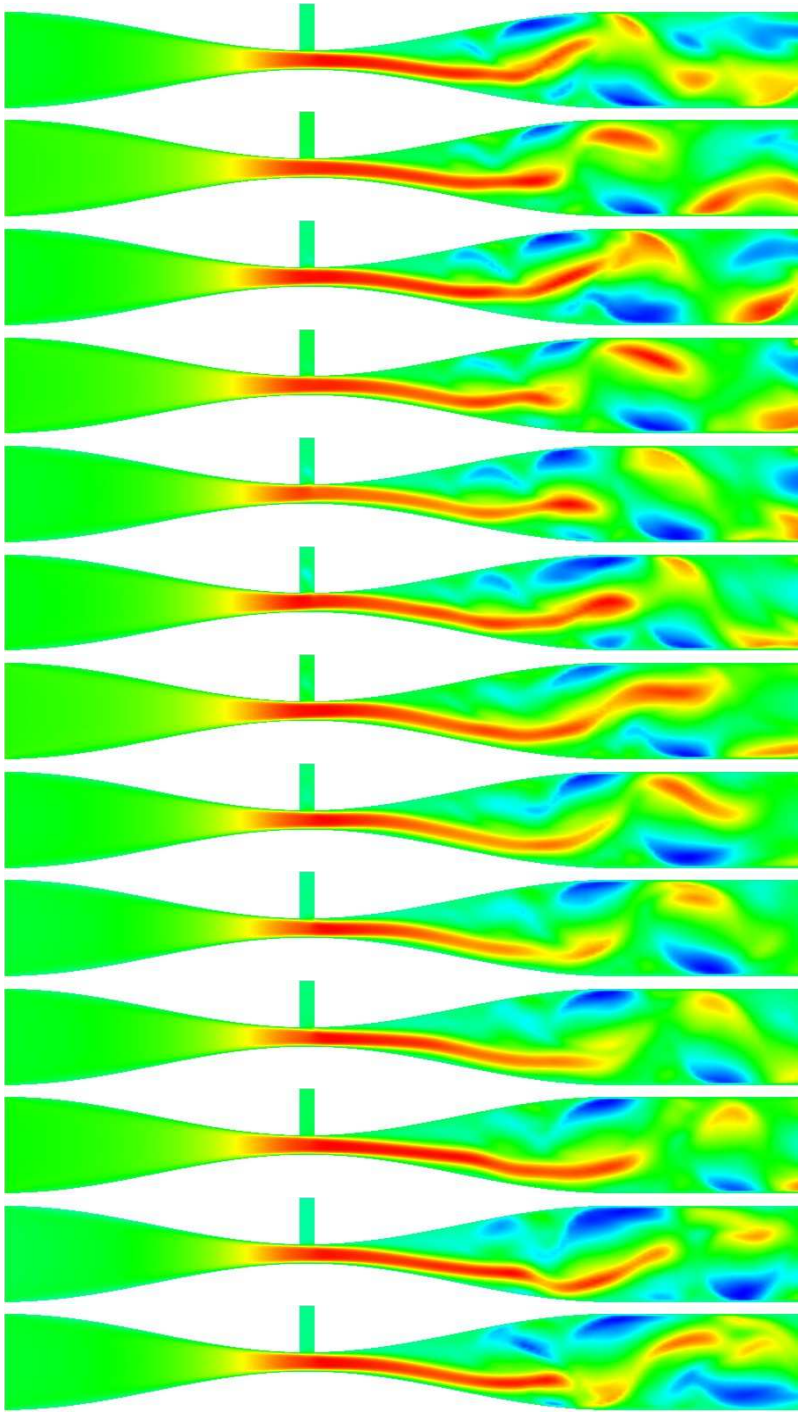


**Figure 10.33**: Visualization of the $u$ velocity in the Venturi pipe at space level 5 for 13 subsequent time units.

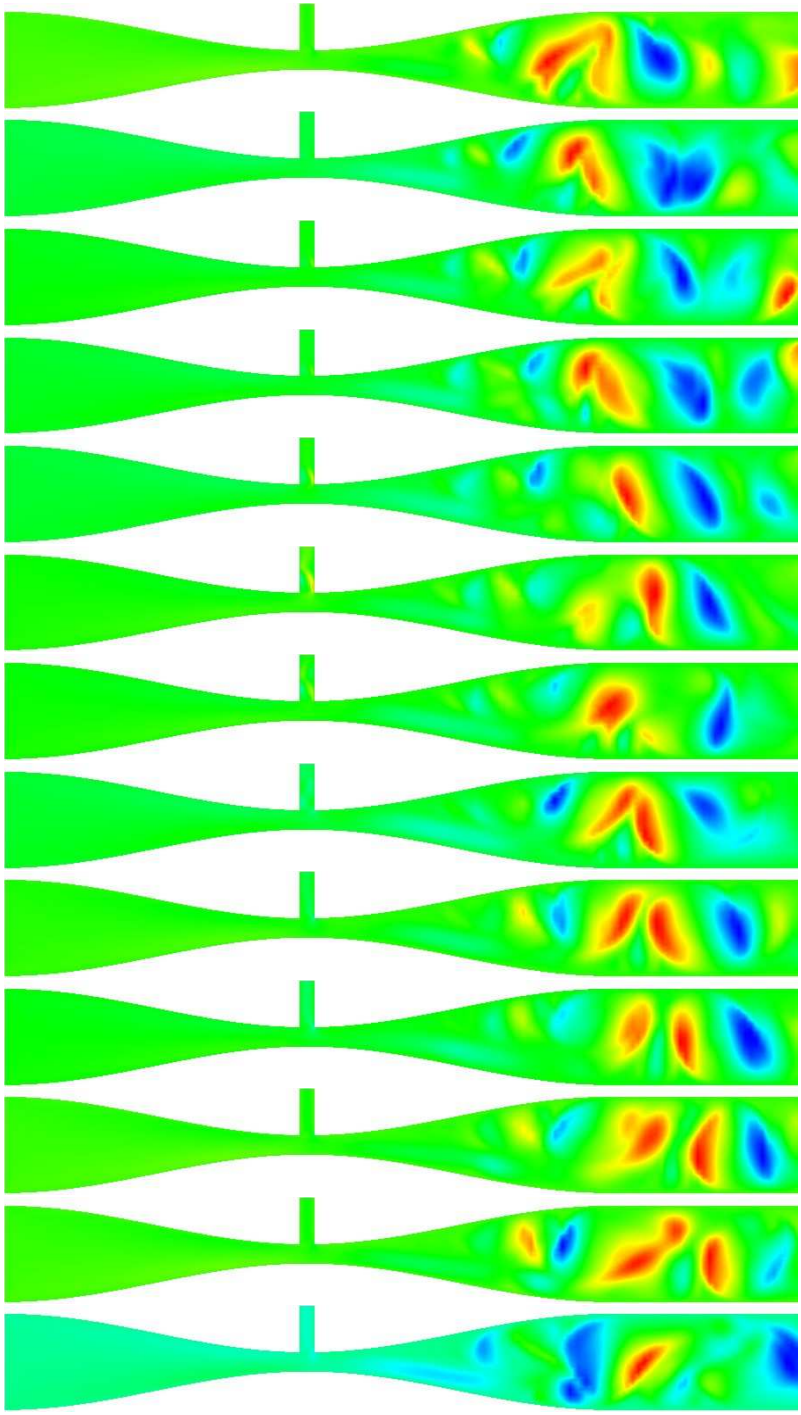


**Figure 10.34**: Visualization of the $v$ velocity in the Venturi pipe at space level 5 for 13 subsequent time units.

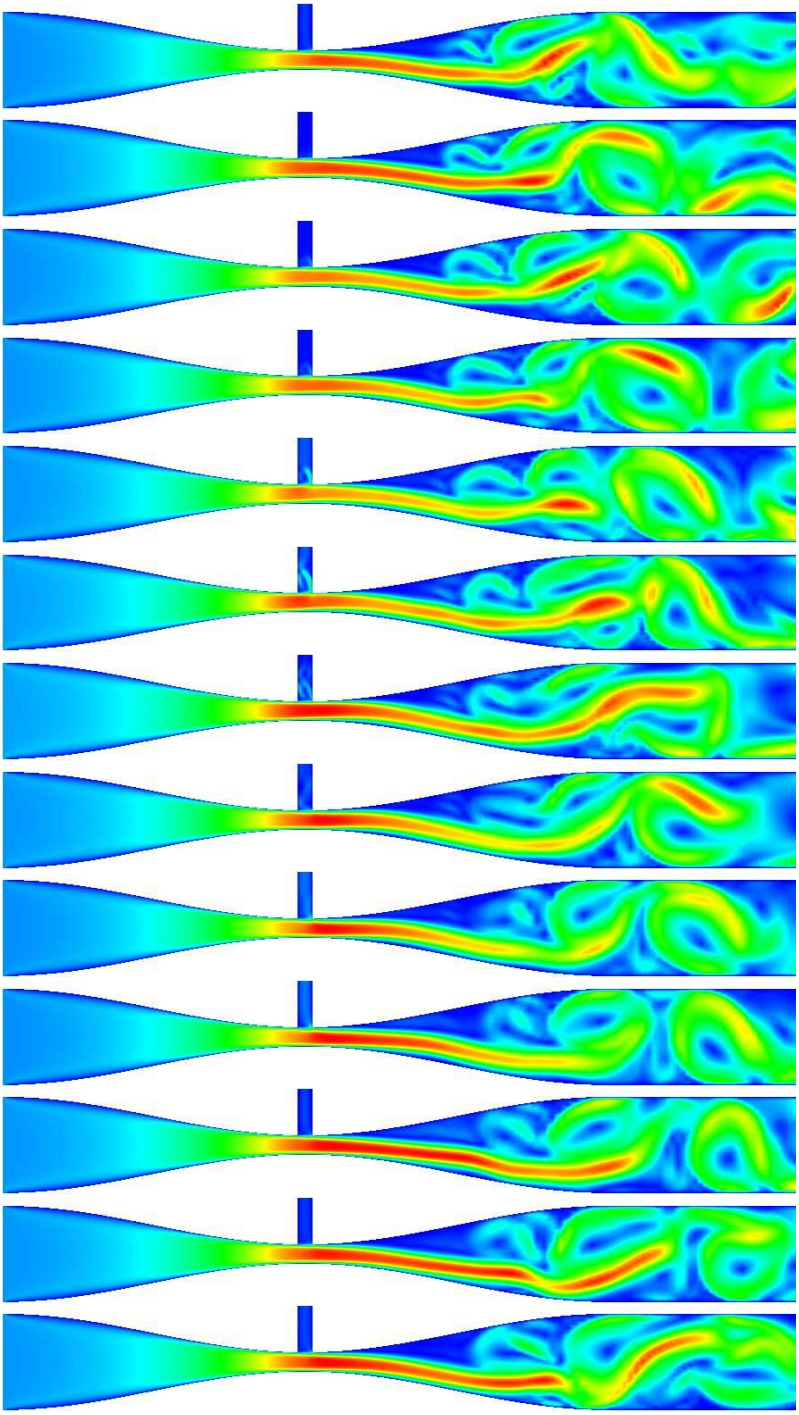


**Figure 10.35**: Visualization of the velocity magnitude in the Venturi pipe at space level 5 for 13 subsequent time units.

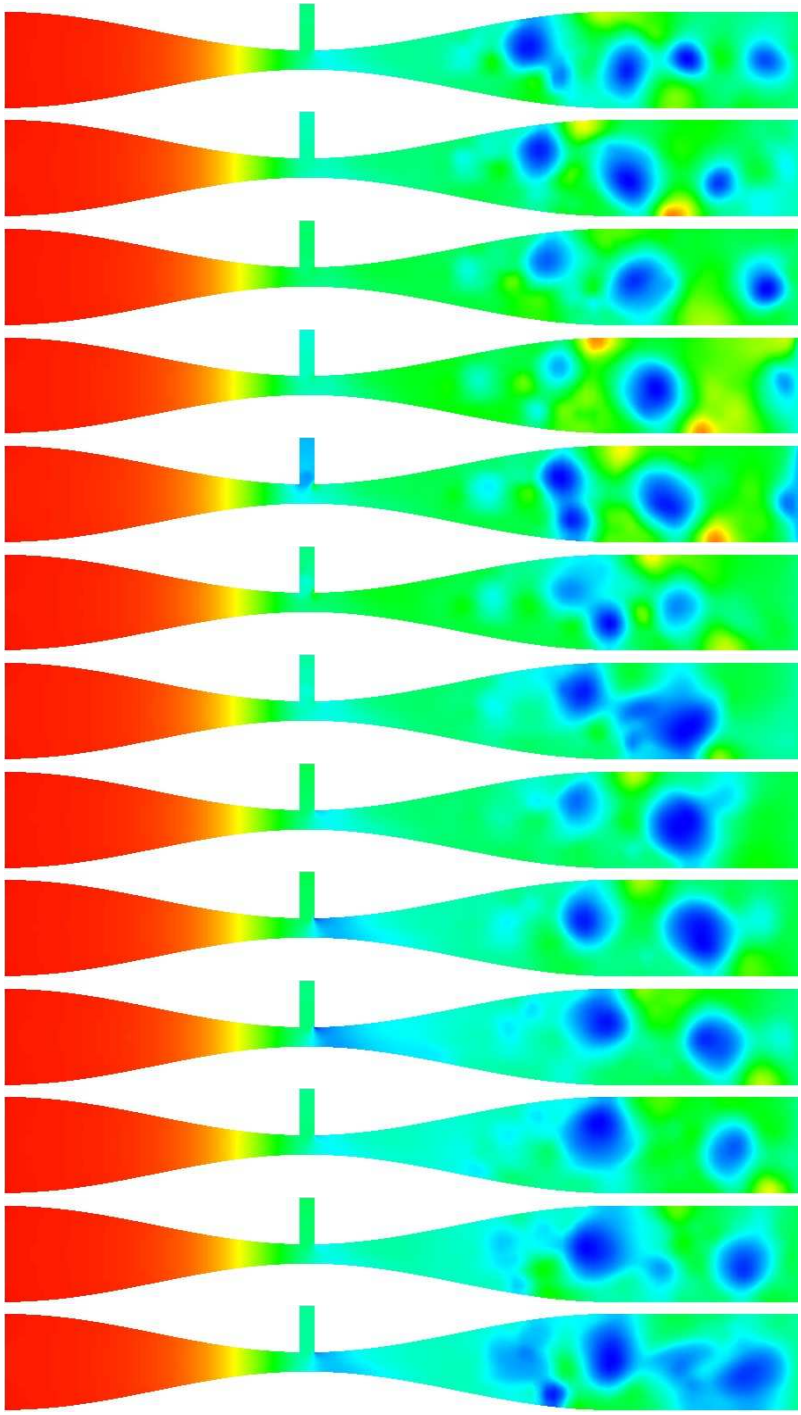


**Figure 10.36**: Visualization of the pressure in the Venturi pipe at space level 5 for 13 subsequent time units.

## 10.3. Solver analysis

In Section 10.1 and 10.2, we performed nonstationary simulations for two special flow configurations, namely, *flow around cylinder* and *flow through a Venturi pipe*, to demonstrate the temporal accuracy and efficiency of the presented higher order time discretization schemes for the incompressible Navier-Stokes equations. Thus, we have continued the work started in [57], where different time stepping schemes were analyzed for these classes of problems. The test problem *flow around cylinder* corresponds to the classical benchmark in [61]. Here, the main difficulty is to compute the right nonstationary behavior of the flow pattern with periodic oscillations and examine the ability of different time discretization schemes to capture the dynamics of the flow. As a second test case, we consider the nonstationary flow for a high Reynolds number through a Venturi pipe which has many real life and industrial applications. In this section, we mainly analyze the behavior of the nonlinear and linear solvers for all the presented time discretizations for the presented problems.

### 10.3.1. Nonstationary flow around cylinder benchmark

In order to measure and compare the efficiency of the nonlinear solvers for the presented time discretization schemes, we show the averaged number '#NL' of nonlinear iterations per time step for the fixed-point and Newton method for different (space) mesh levels. We stop the nonlinear iteration if the $L^2$-norm of the nonlinear residual drops down below $10^{-10}$.

Table 10.10 shows that, for the fixed-point iteration as well as for the Newton method, almost the same number of nonlinear iterations are required for the different time discretization schemes. Moreover, as expected, the number of iterations decreases if we reduce the time step size. Furthermore, the Newton method converges 2-3 time faster as compared to the fixed-point method due to its quadratic convergence. Table 10.11 and 10.12 demonstrate the same behavior for the space level 4 and 5, respectively.

Summarizing, the number of nonlinear steps associated with the maximum allowed time step sizes to gain the accuracy with an error of less than 1% per time period for the cGP(2) and dG(1)-method (see Table 10.8) require only approx. 3 and 2 times more than the cGP(1)-method, respectively. Moreover, this factor is eventually improved by using the Newton method.

Next, we perform numerical tests to analyze the corresponding behavior of the multigrid solver for the solution of linear subproblems w.r.t. the different time discretization schemes. To this end, we present the averaged number '#MG' of multigrid iterations per (nonlinear) step. Here, the multigrid solver uses a preconditioned GMRES method (preconditioned with a cell oriented LPSC type scheme) as smoother and applies four pre- and post-smoothing steps. The multigrid solver stops if the $L^2$-norm of the relative residual is smaller than $10^{-6}$ or the absolute residual drops down below $10^{-15}$.

From the Table 10.13 to 10.15, we see that the multigrid solver requires almost the same number of iterations for the different presented time discretization schemes. Moreover, the number of multigrid iterations remains fairly constant if we increase the refinement level of the space mesh. There is also no noticeable increase in the number of iterations if we decrease the time step which is due to the incompressibility constraint. This means that the behavior of the multigrid solver is very robust and almost independent of the mesh size, the time step size and the used time discretization method.

| τ | Fixed-point | | | Newton | | |
|---|---|---|---|---|---|---|
| | **cGP(1)** #NL | **cGP(2)** #NL | **dG(1)** #NL | **cGP(1)** #NL | **cGP(2)** #NL | **dG(1)** #NL |
| 1/10 | 11.82 | 11.18 | 13.18 | 4.18 | 5.00 | 4.73 |
| 1/13 | 9.86 | 9.43 | 11.00 | 4.00 | 4.50 | 4.07 |
| 1/15 | 9.00 | 8.62 | 9.88 | 3.94 | 4.06 | 4.06 |
| 1/20 | 8.00 | 7.14 | 8.24 | 3.81 | 4.00 | 4.00 |
| 1/25 | 7.00 | 7.00 | 7.04 | 3.04 | 4.00 | 3.69 |
| 1/50 | 5.02 | 5.02 | 5.02 | 3.00 | 3.02 | 3.02 |
| 1/100 | 4.01 | 4.01 | 4.01 | 2.01 | 2.01 | 2.01 |
| 1/200 | 3.00 | 3.00 | 3.00 | 2.00 | 2.00 | 2.00 |

**Table 10.10**: Averaged number of nonlinear iterations per time step for cGP(1) - cGP(2) - dG(1) at space level=3.

| τ | Fixed-point | | | Newton | | |
|---|---|---|---|---|---|---|
| | **cGP(1)** #NL | **cGP(2)** #NL | **dG(1)** #NL | **cGP(1)** #NL | **cGP(2)** #NL | **dG(1)** #NL |
| 1/10 | 10.09 | 10.18 | 11.64 | 4.00 | 4.91 | 4.09 |
| 1/13 | 8.57 | 8.43 | 9.64 | 3.93 | 4.07 | 4.07 |
| 1/15 | 8.00 | 7.94 | 8.69 | 3.94 | 4.00 | 4.06 |
| 1/20 | 7.00 | 7.00 | 7.05 | 3.05 | 4.00 | 4.00 |
| 1/25 | 6.04 | 6.00 | 6.04 | 3.04 | 3.04 | 3.04 |
| 1/50 | 4.02 | 4.24 | 5.00 | 3.00 | 3.00 | 3.00 |
| 1/100 | 3.01 | 3.98 | 4.00 | 2.01 | 2.01 | 2.01 |
| 1/200 | 3.00 | 3.00 | 3.00 | 2.00 | 2.00 | 2.00 |

**Table 10.11**: Averaged number of nonlinear iterations per time step for cGP(1) - cGP(2) - dG(1) at space level=4.

| τ | Fixed-point | | | Newton | | |
|---|---|---|---|---|---|---|
| | **cGP(1)** #NL | **cGP(2)** #NL | **dG(1)** #NL | **cGP(1)** #NL | **cGP(2)** #NL | **dG(1)** #NL |
| 1/10 | 9.00 | 9.00 | 10.27 | 4.00 | 4.55 | 4.09 |
| 1/13 | 7.57 | 7.50 | 8.43 | 3.86 | 4.00 | 4.07 |
| 1/15 | 7.00 | 7.00 | 7.69 | 3.56 | 4.00 | 4.00 |
| 1/20 | 6.00 | 6.00 | 6.05 | 3.05 | 3.90 | 3.38 |
| 1/25 | 5.04 | 5.04 | 6.00 | 3.04 | 3.04 | 3.04 |
| 1/50 | 4.02 | 4.02 | 4.02 | 2.02 | 3.00 | 3.00 |
| 1/100 | 3.01 | 3.01 | 3.01 | 2.01 | 2.01 | 2.01 |
| 1/200 | 2.00 | 3.00 | 3.00 | 2.00 | 2.00 | 2.00 |

**Table 10.12**: Averaged number of nonlinear iterations per time step for cGP(1) - cGP(2) - dG(1) at space level=5.

| τ | Fixed-point-multigrid | | | Newton-multigrid | | |
|---|---|---|---|---|---|---|
| | **cGP(1)** #MG | **cGP(2)** #MG | **dG(1)** #MG | **cGP(1)** #MG | **cGP(2)** #MG | **dG(1)** #MG |
| 1/10 | 11.92 | 10.91 | 11.75 | 10.80 | 10.60 | 10.80 |
| 1/13 | 11.60 | 10.70 | 11.45 | 11.25 | 10.60 | 10.80 |
| 1/15 | 11.67 | 10.89 | 11.40 | 11.25 | 10.20 | 10.80 |
| 1/20 | 11.62 | 11.00 | 11.11 | 11.25 | 10.75 | 11.25 |
| 1/25 | 11.71 | 11.43 | 11.38 | 11.25 | 11.00 | 11.00 |
| 1/50 | 12.17 | 11.83 | 12.00 | 11.67 | 11.00 | 11.25 |
| 1/100 | 12.40 | 11.60 | 12.00 | 11.67 | 11.00 | 11.67 |
| 1/200 | 12.25 | 12.00 | 12.00 | 11.67 | 11.00 | 11.33 |

**Table 10.13**: Averaged number of multigrid iterations per nonlinear step for cGP(1) - cGP(2) - dG(1) at space level=3.

| τ | Fixed-point-multigrid | | | Newton-multigrid | | |
|---|---|---|---|---|---|---|
| | **cGP(1)** #MG | **cGP(2)** #MG | **dG(1)** #MG | **cGP(1)** #MG | **cGP(2)** #MG | **dG(1)** #MG |
| 1/10 | 10.80 | 10.20 | 10.64 | 10.50 | 9.60 | 10.00 |
| 1/13 | 10.78 | 10.22 | 10.50 | 10.50 | 9.60 | 9.80 |
| 1/15 | 10.75 | 10.50 | 10.67 | 10.25 | 10.00 | 9.80 |
| 1/20 | 10.71 | 10.71 | 10.75 | 10.00 | 10.25 | 10.25 |
| 1/25 | 10.57 | 11.00 | 11.00 | 10.00 | 10.25 | 10.25 |
| 1/50 | 11.00 | 11.60 | 11.20 | 10.33 | 10.67 | 10.67 |
| 1/100 | 12.00 | 12.00 | 12.00 | 11.33 | 11.33 | 11.33 |
| 1/200 | 12.50 | 12.50 | 12.50 | 11.67 | 12.00 | 12.00 |

**Table 10.14**: Averaged number of multigrid iterations per nonlinear step for cGP(1) - cGP(2) - dG(1) at space level=4.

| τ | Fixed-point-multigrid | | | Newton-multigrid | | |
|---|---|---|---|---|---|---|
| | **cGP(1)** #MG | **cGP(2)** #MG | **dG(1)** #MG | **cGP(1)** #MG | **cGP(2)** #MG | **dG(1)** #MG |
| 1/10 | 9.56 | 9.11 | 9.36 | 9.79 | 9.00 | 9.20 |
| 1/13 | 9.88 | 9.12 | 9.44 | 9.50 | 9.25 | 9.00 |
| 1/15 | 10.14 | 9.43 | 9.75 | 9.50 | 9.25 | 9.50 |
| 1/20 | 10.00 | 9.67 | 9.86 | 9.50 | 9.25 | 9.50 |
| 1/25 | 10.00 | 9.50 | 10.00 | 9.25 | 9.25 | 9.50 |
| 1/50 | 10.00 | 10.00 | 10.00 | 9.67 | 9.67 | 9.67 |
| 1/100 | 9.75 | 11.00 | 10.50 | 9.67 | 10.33 | 10.00 |
| 1/200 | 11.00 | 12.00 | 11.67 | 9.50 | 11.33 | 11.00 |

**Table 10.15**: Averaged number of multigrid iterations per nonlinear step for cGP(1) - cGP(2) - dG(1) at space level=5.

### 10.3.2. Nonstationary flow through a Venturi pipe

Now we analyze the solver for the Venturi pipe flow similar to the example "flow around cylinder". Again, we show the averaged number of nonlinear iterations per time step for the fixed-point and Newton method for different space mesh levels. We apply the same stopping criterion as in the previous example for the nonlinear and linear solver, respectively.

From Table 10.16 to 10.18, we see again that, for the fixed-point iteration as well as for the Newton method, almost the same number of iterations is required for every time discretization scheme. Moreover, for a fixed space mesh level, the number of nonlinear iterations decreases if the time step size is reduced, as expected. Concerning the number of nonlinear iterations, it is noticed that the Newton method is more efficient than the fixed point iteration which shows almost the same behavior as in case of the flow around cylinder example. These results indicate that the nonlinear solver is also robust with respect to the underlying flow configuration.

| $\tau$ | Fixed-point | | | Newton | | |
|---|---|---|---|---|---|---|
| | **cGP(1)** #NL | **cGP(2)** #NL | **dG(1)** #NL | **cGP(1)** #NL | **cGP(2)** #NL | **dG(1)** #NL |
| 1/10 | 10.36 | 9.64 | 10.91 | 3.09 | 3.45 | 3.36 |
| 1/13 | 9.07 | 8.64 | 9.64 | 3.07 | 3.07 | 3.07 |
| 1/15 | 8.56 | 8.50 | 9.19 | 3.06 | 3.06 | 3.06 |
| 1/20 | 7.67 | 7.62 | 8.14 | 3.05 | 3.05 | 3.05 |
| 1/25 | 7.04 | 7.00 | 7.58 | 2.77 | 3.04 | 3.04 |
| 1/50 | 5.80 | 5.75 | 5.90 | 2.08 | 2.55 | 2.53 |
| 1/100 | 4.72 | 4.77 | 4.88 | 2.01 | 2.01 | 2.01 |
| 1/200 | 4.00 | 4.00 | 4.00 | 2.00 | 2.00 | 2.00 |

**Table 10.16**: Averaged number of nonlinear iterations per time step for cGP(1) - cGP(2) - dG(1) at space level=3.

| $\tau$ | Fixed-point | | | Newton | | |
|---|---|---|---|---|---|---|
| | **cGP(1)** #NL | **cGP(2)** #NL | **dG(1)** #NL | **cGP(1)** #NL | **cGP(2)** #NL | **dG(1)** #NL |
| 1/10 | 10.27 | 9.91 | 11.09 | 3.27 | 3.73 | 3.73 |
| 1/13 | 9.07 | 9.00 | 9.79 | 3.07 | 3.36 | 3.29 |
| 1/15 | 8.56 | 8.75 | 9.25 | 3.06 | 3.06 | 3.06 |
| 1/20 | 7.76 | 7.95 | 8.52 | 3.05 | 3.05 | 3.05 |
| 1/25 | 7.12 | 7.31 | 7.73 | 3.04 | 3.04 | 3.04 |
| 1/50 | 5.63 | 5.57 | 5.80 | 2.12 | 2.49 | 2.47 |
| 1/100 | 4.33 | 4.38 | 4.50 | 2.01 | 2.01 | 2.01 |
| 1/200 | 3.60 | 3.76 | 3.81 | 2.00 | 2.00 | 2.00 |

**Table 10.17**: Averaged number of nonlinear iterations per time step for cGP(1) - cGP(2) - dG(1) at space level=4.

| τ | Fixed-point | | | Newton | | |
|---|---|---|---|---|---|---|
| | **cGP(1)** #NL | **cGP(2)** #NL | **dG(1)** #NL | **cGP(1)** #NL | **cGP(2)** #NL | **dG(1)** #NL |
| 1/10 | 9.64 | 9.73 | 10.64 | 3.27 | 3.73 | 3.73 |
| 1/13 | 8.71 | 9.00 | 9.64 | 3.07 | 3.29 | 3.21 |
| 1/15 | 8.31 | 8.62 | 9.00 | 3.06 | 3.06 | 3.06 |
| 1/20 | 7.52 | 7.76 | 8.24 | 3.05 | 3.05 | 3.05 |
| 1/25 | 6.96 | 7.23 | 7.58 | 3.04 | 3.04 | 3.04 |
| 1/50 | 5.39 | 5.41 | 5.55 | 2.06 | 2.25 | 2.24 |
| 1/100 | 4.01 | 4.01 | 4.24 | 2.01 | 2.01 | 2.01 |
| 1/200 | 3.00 | 3.16 | 3.27 | 2.00 | 2.00 | 2.00 |

**Table 10.18**: Averaged number of nonlinear iterations per time step for cGP(1) - cGP(2) - dG(1) at space level=5.

Next, we analyze the behavior of the multigrid solver for the solution of linear subproblems. Here, the multigrid solver is using the same settings as in the example with the flow around a cylinder. In Table 10.19 to 10.21, we present the averaged number of multigrid iterations per nonlinear step for solving the corresponding linear block systems for the space mesh levels 3-5.

We observe again that the multigrid solver requires almost the same number of iterations for the presented time discretization schemes. Moreover, the number of multigrid iterations remains almost constant for increasing space mesh level. There is also no noticeable increase in the number of iterations if we decrease the time step. Comparing with the results in the flow around cylinder example we observe a similar solver behavior as for the Venturi pipe flow. This indicates that our multigrid solver is also robust with respect to the underlying flow configurations.

| τ | Fixed-point-multigrid | | | Newton-multigrid | | |
|---|---|---|---|---|---|---|
| | **cGP(1)** #MG | **cGP(2)** #MG | **dG(1)** #MG | **cGP(1)** #MG | **cGP(2)** #MG | **dG(1)** #MG |
| 1/10 | 9.45 | 10.80 | 10.45 | 10.00 | 11.50 | 11.25 |
| 1/13 | 9.80 | 10.78 | 10.80 | 10.50 | 11.25 | 11.25 |
| 1/15 | 9.89 | 10.33 | 10.70 | 10.25 | 11.00 | 11.00 |
| 1/20 | 9.89 | 10.38 | 10.56 | 10.25 | 11.00 | 11.00 |
| 1/25 | 10.38 | 10.00 | 10.38 | 10.75 | 11.00 | 11.00 |
| 1/50 | 9.83 | 10.33 | 10.50 | 10.67 | 10.50 | 10.25 |
| 1/100 | 10.20 | 10.20 | 10.60 | 10.67 | 10.33 | 10.67 |
| 1/200 | 10.40 | 10.00 | 10.40 | 10.67 | 10.33 | 10.67 |

**Table 10.19**: Averaged number of multigrid iterations per nonlinear step for cGP(1) - cGP(2) - dG(1) at space level=3.

| τ | Fixed-point-multigrid | | | Newton-multigrid | | |
|---|---|---|---|---|---|---|
| | **cGP(1)** #MG | **cGP(2)** #MG | **dG(1)** #MG | **cGP(1)** #MG | **cGP(2)** #MG | **dG(1)** #MG |
| 1/10 | 9.50 | 10.00 | 9.91 | 10.50 | 10.40 | 10.00 |
| 1/13 | 10.11 | 10.00 | 10.20 | 10.25 | 11.25 | 11.00 |
| 1/15 | 10.11 | 10.33 | 10.30 | 10.00 | 11.00 | 11.25 |
| 1/20 | 10.25 | 10.38 | 10.56 | 10.50 | 10.75 | 11.50 |
| 1/25 | 10.12 | 10.43 | 10.50 | 10.75 | 11.00 | 11.25 |
| 1/50 | 10.83 | 10.17 | 10.50 | 11.00 | 10.25 | 10.50 |
| 1/100 | 11.00 | 11.20 | 10.80 | 11.00 | 11.00 | 11.33 |
| 1/200 | 10.80 | 10.40 | 10.20 | 11.00 | 10.67 | 11.00 |

**Table 10.20**: Averaged number of multigrid iterations per nonlinear step for cGP(1) - cGP(2) - dG(1) at space level=4.

| τ | Fixed-point-multigrid | | | Newton-multigrid | | |
|---|---|---|---|---|---|---|
| | **cGP(1)** #MG | **cGP(2)** #MG | **dG(1)** #MG | **cGP(1)** #MG | **cGP(2)** #MG | **dG(1)** #MG |
| 1/10 | 10.67 | 10.20 | 11.20 | 12.00 | 10.60 | 11.80 |
| 1/13 | 10.44 | 10.33 | 10.80 | 11.25 | 11.50 | 12.25 |
| 1/15 | 10.38 | 10.62 | 11.11 | 11.00 | 11.75 | 12.00 |
| 1/20 | 10.00 | 11.29 | 11.38 | 10.75 | 11.50 | 12.00 |
| 1/25 | 10.43 | 11.14 | 12.00 | 10.75 | 12.00 | 12.00 |
| 1/50 | 11.00 | 11.17 | 12.17 | 11.33 | 11.00 | 12.00 |
| 1/100 | 11.80 | 11.00 | 11.20 | 11.00 | 10.67 | 11.33 |
| 1/200 | 12.25 | 11.25 | 11.50 | 12.00 | 11.00 | 11.00 |

**Table 10.21**: Averaged number of multigrid iterations per nonlinear step for cGP(1) - cGP(2) - dG(1) at space level=5.

## 10.4. Summary

In this chapter, we have analyzed the *continuous Galerkin-Petrov* and *discontinuous Galerkin* time discretization schemes for the nonstationary Navier-Stokes equations.

### Accuracy in practice

In order to validate the applicability and the comparison of various temporal discretization schemes, we have considered a couple of practical problems. The first test problem considered in this chapter exactly corresponds to the classical flow around cylinder benchmark [61]. In this simulation, we have concentrated only on the nonstationary behavior of the flow pattern with periodic oscillations and examined the ability of different time discretization schemes to capture the dynamics of the flow. The quantities of physical interest are the lift, drag coefficient and pressure drop. Beside these quantities, we have also demonstrated the temporal accuracy of the presented time discretization schemes by computing the $v$-velocity near the obstacle in the direction of the fluid. A quantitative analysis has been performed for the lift coefficient $C_L$ and $v$-velocity near the obstacle. The numerical tests have shown that the cGP(2)-method gains the same accuracy at a time step size which is 10 times larger than the associated time steps for cGP(1) or CN while the dG(1)-method achieves this accuracy for which the associated time step is 5 time larger than that of cGP(1) or CN. The same numerical tests have been performed for different (space) mesh levels

and almost the same results are obtained for all levels.

As a second test problem, we have chosen the nonstationary flow for a high Reynolds number through a venturi pipe. The objective of this simulation is to control the instantaneous and mean flux through this device. The numerical results have demonstrated that the time step size for cGP(2), for which almost the same results are obtained as for cGP(1) or CN, can again be chosen 10 times larger than the associated time steps for cGP(1) or CN. For the dG(1) this factor is 5 as compared to cGP(1) or CN. This factor becomes even more clear for higher space mesh levels. Consequently, we can say that the cGP(2)-method is 10 times faster than the cGP(1) or CN and dG(1) is 5 times faster than cGP(1) or CN (at least for our test problems) at comparable numerical costs.

#### Efficiency

Finally, we have analyzed the behavior of the nonlinear and linear solvers for all the presented time discretization schemes. This analysis is performed for both test problems. The numerical results have shown that almost the same number of iterations is required for every time discretization scheme for both the fixed-point and Newton method. Moreover, the number of nonlinear iterations decreases by reducing the time step size as expected. Furthermore, the well-known Newton method converges 2-3 time faster as compared to the fixed-point method due to its quadratic convergence. Next, we also performed numerical tests to analyze the corresponding behavior of the multigrid solver for the solution of linear subproblems for the different time discretization schemes. Here, the multigrid solver is almost independent of the spatial mesh size, the time step, the underlying flow configuration and the discretization scheme used.

# 11

# Conclusions and Outlook

## 11.1. Conclusions

The investigation of the higher order accurate time discretization schemes is an interesting topic since the development of such discretizations is advantageous specially for the computation of long time simulations in the field of computational fluid dynamics (CFD). They become particularly essential in the computation of long term simulations. In this thesis, high order accurate time discretization schemes have been investigated for the nonstationary simulation of partial differential equations. In practice, first and second order time discretization schemes have been predominantly used due to their simple form or due to stability reasons of high order time discretizations. The most commonly used time stepping schemes are the implicit Euler, the Crank-Nicolson and the discontinuos Galerkin (dG($k$)) method as a representative of arbitrary order scheme. In the present thesis, a new class of stable time discretization schemes, namely the continuous Galerkin-Petrov (cGP($k$))) method has been developed and analyzed. For the cGP($k$)-methods, the discrete solution space consists of the continuous piecewise polynomial functions of degree $k$ and the discrete test space of discontinuous polynomials of degree $k-1$ while in the case of dG($k$)-methods, both the discrete solution and test spaces consist of discontinuous polynomials of degree $k$.

### Stability

The stability of the time discretization schemes is an important aspect in theory as well as in practice. In the present thesis, we have discussed the concept of 'A-stability' and 'L-stability'. The well-known dG($k$)-methods are known to be strongly A-stable (or L-stable). In comparison to dG($k$)-methods, the cGP($k$)-methods are A-stable. We have proved by some energy arguments the A-stability of cGP($k$)-method and an optimal error estimates of order $k+1$ in standard $L^2$-norm.

### Accuracy

The presented class of time discretization schemes are accurate of arbitrary order. In general, the cGP($k$)-methods have the accuracy of order $k+1$ in the standard $L^2$-norm. On the other hand, the dG($k$)-methods have the same accuracy of order $k+1$ but at the higher numerical costs than the cGP($k$)-methods.

In this thesis, we have presented a numerical study of the higher order time discretizations cGP(1), cGP(2) and dG(1), taking into account aspects of numerical accuracy and efficiency of the corresponding solution methods for the resulting (coupled) linear systems. In our numerical results, we have observed that the estimated experimental orders of convergence confirm the theoretical orders for the heat and the nonstationary Stokes equation in two dimensions. First of all, the cGP(1)-method is very close to the well-known Crank-Nicolson scheme: Both methods differ only in the choice of the unknown that is solved for on each time interval and in the way how the

numerical integration of the right hand side is done. The cGP(1)-method is accurate of order 2 in the whole time interval as it is known for the Crank-Nicolson scheme. However, for the cGP(2)-method as well as in dG(1)- method, we have two unknowns on each time interval which have to be computed by solving a $2 \times 2$ block system. The cGP(2)-method is accurate of order 3 in the whole time interval and shows even superconvergence of order 4 in the discrete time points. In contrast, the dG(1)-method is of order 2 in the whole time interval and superconvergent of order 3 in the discrete time points at the same numerical costs. Furthermore, the tests show that the cGP(2)-scheme provides significantly more accurate numerical solutions than the other presented schemes cGP(1) and dG(1) which means that quite large time step sizes are allowed to gain highly accurate results.

#### Different variants of the cGP($k$)-methods

In [49], the cGP($k$)-methods using the Gauß-Lobatto points were analyzed for the ODE system in $\mathbb{R}^2$. We have performed some numerical tests using this variant of the cGP($k$)-method for $k = 1,2$ for the heat equation. The Gauß-Lobatto based cGP(1)-method is of order 2 in the whole time interval which is the same behavior as the cGP(1)-method analyzed in this thesis. The Gauß-Lobatto based cGP(2)-method is of order 4 in the whole time interval while the standard variant of the cGP(2) is accurate of order 3 in the whole time interval and superconvergent of order 4 in the discrete time points. Furthermore, an advantage of the Gauß-Lobatto based cGP($k$)-method is that it does not require any extrapolation to get the solution at the discrete time points $t_n$ while in case of our presented cGP($k$)-method, the solution at the discrete time points $t_n$ always obtained by using the extrapolation from the known solution at the Gauß points.

However, all the advantages only hold for the heat equation. For saddle point problems like the Stokes or Navier-Stokes equations, the Gauß-Lobatto based variant is not applicable; the explicit time discretization of the pressure is numerically instable. To overcome this difficulty, we have the motivation to reconsider the cGP($k$)-methods using the standard Gauß points.

#### Postprocessing for high order pressure

Regarding the computation of the pressure, since we have superconvergence results for the velocity only at the discrete time points $t_n$, it is desirable to get a high order pressure at the same points too. In the cGP($k$)-methods, we apply extrapolation as a special reconstruction technique to obtain the velocity at the discrete time points $t_n$ from the obtained velocity at the intermediate $k$ Gaussian points in the subinterval $[t_{n-1}, t_n]$. This leads to superconvergence results in these time points. Unfortunately, the same extrapolation technique is not applicable for the pressure, so the order of approximation is lower with this technique. However, using a special Lagrangian interpolation technique which also includes the pressure at the next time step allows to recover the higher order approximation in discrete time points $t_n$. Thatway, order of approximation is the same for velocity and pressure in these points. This technique has proven to be also advantageous in practice for being used with dG($k$)-method. This technique applied for the dG(1)-method gives better results than the associated extrapolation.

#### Realization of efficient solver techniques

From the discretization of incompressible Navier-Stokes equation, the nonlinear coupled systems of multiple DOF's in time which can be characterized as saddle point problems, are obtained. This system has to be solved in every time step. An efficient way to solve the nonlinear system is the Newton method which is well known for its quadratic convergence if the initial solution is chosen close enough to the exact solution and the problem is smooth enough. Another method to solve

the system of nonlinear equations is the fixed-point method which is less efficient than the Newton method (i.e., with linear or superlinear convergence only).

The resulting linear systems, for instance, from the discretization of heat equation, Stokes equation or the linearized Navier-Stokes equations are solved by using preconditioned Krylov space or multigrid methods. The corresponding linear systems in each time interval $[t_{n-1}, t_n]$ in our application, which are $6 \times 6$ fully coupled block systems in the case of the cGP(2) and dG(1) approach and $3 \times 3$ block systems for the cGP(1)-method. In case of the heat equation, standard iterative methods such as Jacobi, Gauss-Seidel, SOR or ILU can be applied as a preconditioner with Krylov space and smoothers with geometrical multigrid solver. On the other hand, in case of the Stokes or Navier-Stokes equations, such standard iterative methods cannot be applied. Some of the options in this case are to use Krylov space methods or the local pressure Schur complement (LPSC) type scheme as an inner or outer iteration for the multigrid methods. In this numerical study, the corresponding linear systems in case of the Navier-Stokes equations are solved by using a coupled multigrid solver with preconditioned GMRES method (preconditioned with a local pressure Schur complement (LPSC) type scheme) as smoother. The numerical experiments confirm that multigrid methods can be regarded as the most efficient iterative solvers since their rate of convergence is almost independent of the problem size which is characterized here by the mesh size of the space grid and the size of the time step.

#### Numerical costs

With respect to the computational costs, which mainly depend on the size of the resulting block system that has to be solved for each time interval, the cGP($k$)-method is comparable to the dG($k-1$)-method. However, concerning the discretization error in time, the accuracy of the cGP($k$)-method is one order higher than that of the dG($k-1$)-method.

In our numerical study, all the presented time discretization schemes are analyzed numerically w.r.t. computational costs for the heat and Stokes equations. From the numerical tests, we have seen that the multigrid solver requires almost the same number of iterations for the different presented time discretization schemes. Moreover, the number of multigrid iterations remains fairly constant if we increase the refinement level of the space mesh. There is also no noticeable increase in the number of iterations if we decrease the time step. This means that the behavior of the multigrid solver is almost independent of the spatial mesh and the time step size. Furthermore, we have observed that the CPU-time in case of cGP(2) or dG(1) is almost 2-3 times the CPU-time of cGP(1) for the multigrid solver. We also note that the CPU-time grows approximately by a factor of 4 as expected if we increase the space mesh level. These factors are nearly optimal since the number of space unknowns is increased by a factor of 4 if the level is increased by one.

#### In practice

Finally, we have analyzed how the cGP($k$) and dG($k$)-methods behave in practice since it is always required to simulate the time dependent CFD problems in an efficient way. In order to validate the applicability of the presented time discretization schemes, we have considered a couple of practical examples in Chapter 9.

The first test problem considered in this thesis exactly corresponds to the classical flow around cylinder benchmark [61]. In this simulation, we have concentrated only on the nonstationary behavior of the flow pattern with periodic oscillations and examine the ability of different time discretization schemes to capture the dynamics of the flow. The quantities of physical interest are the lift and drag coefficient, the pressure drop between the two points on the obstacle and the $v$-velocity near the obstacle in the direction of the fluid. The temporal discretizations were then compared by means of plotting these physical quantities for various time step sizes and measuring

the deviation over time in comparison to reference curves. The numerical tests have shown that the cGP(2)-method gains the same accuracy at a time step size which is 10 times larger then the associated time steps for cGP(1) or CN while the dG(1)-method achieves this accuracy for which the associated time step is 5 time larger than that of cGP(1) or CN. The same numerical tests have been performed for different (space) mesh levels and almost the same results are obtained for all levels.

As a second test problem, we chose the nonstationary flow for a high Reynolds number through a venturi pipe. If the inflow speed from the inlet is sufficiently enough, then due to the Bernoulli principle, the narrow section in the middle of the pipe produces a low pressure which creates a flux through the upper part of the small pipe. The objective of this simulation is to control the instantaneous and mean flux through this device. In order to compare the presented temporal discretizations the flow is restarted from the same start solution (Stokes solution) and the simulations are performed for 30 seconds with various time step sizes. The numerical results have demonstrated that the time step size for cGP(2) for which almost the same results are obtained as for cGP(1) or CN, is again 10 times larger than the associated time steps for cGP(1) or CN while for the dG(1) this factor is 5 as compared to cGP(1) or CN. This factor becomes even more clear for higher space mesh levels. Consequently, we can say that the cGP(2)-method is 10 times faster than the cGP(1) or CN and dG(1) is 5 times faster than cGP(1) or CN at comparable numerical costs.

At the end, we have analyzed the behavior of the nonlinear and linear solvers for all the presented time discretization schemes. This analysis is performed for both test problems. In order to measure and compare the efficiency of the nonlinear solver for the presented time discretization schemes, we showed the averaged number of nonlinear iterations per time step for the fixed-point and Newton method for different (space) mesh levels. The numerical results have shown that almost the same number of iterations is required for every time discretization scheme for both fixed-point and Newton method. Moreover, the number of nonlinear iterations decreases by reducing the time step size as expected. Furthermore, the well-known Newton method converges 2-3 time faster as compared to the fixed-point method due to its quadratic convergence. Next, we also performed numerical tests to analyze the corresponding behavior of the multigrid solver for the solution of linear subproblems for the different time discretization schemes. Here, the multigrid solver is almost independent of the spatial mesh and the time step size.

## 11.2. Outlook

The work done in the presented thesis can be extended in various directions to improve the efficiency and stability of the FE solver for the nonstationary simulation of complex time dependent flow problems. For instance, the following improvements can be made:

- The nonlinear solver can be improved by choosing an optimal damping in each nonlinear step (see [58]).
- The efficiency of the linear multigrid solver can be increased if more powerful smoothing is used (see [50]).
- The same implementation and analysis of cGP($k$)-method can be extended by increasing the degree of time polynomials, i.e., cubic polynomials can be used to 3 (for which we obtained the cGP(3)-method) to increase temporal accuracy.
- One time step corresponds to multiple fully coupled Navier-Stokes equations in time; system solved with a monolithic Newton multigrid solver which treats all solution components in one time step simultaneously. Analysis of the LPSC type smoother for $k > 2$.

- Adaptivity in time can be used to improve the time accuracy.

#### Future developments and alternative approaches

The presented time discetization cGP($k$)-method is good in stability and accuracy of arbitrary order. However, a lot of development is currently going on in the construction of time discretization schemes. On particular example is dG_C0($k+1$)-method, a new time discretization scheme of variational type is proposed in [40], which is derived from the dG($k$)-method. The idea in this method is to increase the global smoothness of the discrete solution space space by one level higher than the original dG($k$)-method, i.e., the smoothness of the discrete solution is increased form $C^{-1}$- to $C^0$-continuity. Consequently, the continuity constraints of the discrete solution at the end-points of the time intervals allow us to increase the polynomial order to $k+1$ without increasing the total number of unknowns in the discrete system being solved in each time step. Therefore, by increasing the polynomial order, the accuracy of the new method is one order higher with the same computational cost. Moreover, the dG-C0($k+1$)-method generates the same solution values at the end-points of the time intervals as the dG($k$)-method. Thus, we obtain the strong A-stability of the dG-C0($k+1$)-method from the strong A-stability of the original dG($k$)-method. We have presented and shortly analyzed the dG-C0($k$)-method for the heat equation in Appendix B.1. In particular, we have also performed preliminary numerical tests to demonstrate the accuracy of the dG-C0(2)-method. The experimental order of convergence confirms the theoretical order of convergence. In our future work, we also plan to extend the dG-C0(2)-method, implemented for the heat equation in Appendix B.1, for the nonstationary Navier-Stokes equations to simulate complex time dependent flow problems together with highly sophisticated Newton-multigrid techniques for the corresponding saddle point problems. The results are already promising and a future analysis will reveal advantages and disadvantages in comparison to the cGP($k$)-method introduced in this thesis.

# A

## The cGP(1)-method and Crank-Nicolson scheme

In this appendix we show that the cGP(1)-method and the well-known Crank-Nicolson scheme are identical for the heat equation.

**Proof.** On each time interval $[t_n, t_{n+1}]$ the cGP(1)-method applied to the heat equation is given as

$$\frac{u^{n+\frac{1}{2}} - u^n}{dt} = f^{n+\frac{1}{2}} + \triangle u^{n+\frac{1}{2}} \tag{A.1}$$

where $dt = \frac{\tau_n}{2}$, and the corresponding linear extrapolation to find the solution at time $t_{n+1}$ is

$$u^{n+1} = 2u^{n+\frac{1}{2}} - u^n. \tag{A.2}$$

Applying a finite element discretization, we have

$$(M + \frac{\tau_n}{2}L)\underline{U}^{n+\frac{1}{2}} = dtF^{n+\frac{1}{2}} + M\underline{U}^n \tag{A.3}$$

where $L = -\triangle$ denotes the Laplacian matrix and $M$ is the mass matrix. Combining equation (A.2) and (A.3), we obtain

$$(M + \frac{\tau_n}{2}L)\underline{U}^{n+1} = \frac{\tau_n}{2}(F^{n+1} + F^n) + M\underline{U}^n - \frac{\tau_n}{2}L\underline{U}^n \tag{A.4}$$

which is the well-known Crank-Nicolson scheme. □
Next, we perform some numerical experiments which confirm the above theoretical results.

**Example A.1** *As a test example we consider the following two dimensional heat equation*

$$\begin{array}{rcll} d_t u(x,t) - \Delta u(x,t) &=& f & in \quad \Omega, \quad \forall\, t \in [0,1], \\ u &=& 0 & at \quad \partial\Omega \end{array} \tag{A.5}$$

*where*

$$f = x(1-x)y(1-y)e^t + 2[y(1-y) + x(1-x)]e^t$$

*is a given function and* $\Omega \subset \mathbb{R}^2$ *is the unit square and* $u(x,t) = x(1-x)y(1-y)e^t$.

We apply the cGP(1) and Crank-Nicolson method with an equidistant time step size $\tau$. The error over the time interval $[0,1]$ for the analytical solution can be seen in the following table.

| | cGP(1) | | CN | |
|---|---|---|---|---|
| $1/\tau$ | $\|u-u_\tau\|_2$ | EOC | $\|u-u_\tau\|_2$ | EOC |
| 10 | 6.27E-05 | | 6.27E-05 | |
| 20 | 1.57E-05 | 2.00 | 1.57E-05 | 2.00 |
| 40 | 3.92E-06 | 2.00 | 3.92E-06 | 2.00 |
| 80 | 9.80E-07 | 2.00 | 9.80E-07 | 2.00 |
| 160 | 2.45E-07 | 2.00 | 2.45E-07 | 2.00 |
| 320 | 6.13E-08 | 2.00 | 6.13E-08 | 2.00 |
| 640 | 1.53E-08 | 2.00 | 1.53E-08 | 2.00 |
| 1280 | 3.83E-09 | 2.00 | 3.83E-09 | 2.00 |

**Table A.1**: Error norms $\|u-u_\tau\|_2$ for the test case with known analytical solution.

From the above Table A.1, the results demonstrate that the cGP(1) and Crank-Nicolson schemes are identical. Hence, the experimental results confirm the theoretical results.

# B

# The dG-C0($k$)-method

In this appendix, a new variational time discretization proposed in [40], is considered for the heat equation which is a variant of the dG($k$)-method, called the dG-C0($k$+1)-method. This was numerically analyzed for the Burgers equation by [40]. The dG-C0($k$+1)-method is derived from the dG($k$)-method by increasing the level of the global smoothness of the discrete solution from $C^{-1}$- to $C^0$-continuity. In this way, due to the continuity constraint on the time derivative of the solution at the endpoints of the time intervals, they can increase order of the time polynomials to k+1 without increasing the total number of unknowns in the system. Thus, we obtain a new method called dG-C0($k$ + 1) which has the same computational costs as for the dG($k$)-method but with a one order higher accuracy due to the approximation by polynomials with a one higher degree. Similar to the original dG($k$)-method, the dG-C0($k$)-method is also strongly A-stable (or L-stable). Moreover, the new higher order method and the original method coincide at the endpoints of the time intervals.

### 2.1. The dG-C0($k$)-method for the heat equation

To apply the dG-C0($k$)-method for the heat equation (5.1), the idea is to make the discrete solution one level smoother in time such that in the time marching process more information of the solution from the previous time interval can be reused in the new time interval. This allows to reduce the computational costs. That means in our case, we have to change from a discontinuous solution in the dG($k$)-method to a continuous solution in the dG-C0($k$)-method. The result is a method that combines variational equations with left-sided collocation conditions at the rightmost Gauß-Radau points on the time intervals.

We start by decomposing the time interval $I = [0,T]$ into $N$ subintervals $I_n := [t_{n-1}, t_n]$, where $n = 1,\dots,N$ and $0 = t_0 < t_1 < \cdots < t_{N-1} < t_N = T$. The symbol $\tau$ will denote the *time discretization parameter* and will also be used as the maximum time step size $\tau := \max_{1\le n\le N} \tau_n$, where $\tau_n := t_n - t_{n-1}$. Then, we approximate the solution $u : I \to V$ by means of a function $u_\tau : I \to V$ which is piecewise polynomial of some order $k$ with respect to time, i.e., we are looking for $u_\tau$ in the discrete time space

$$\mathbb{P}_k^c(\mathcal{M}_\tau) := \{u \in C(I,V) : \, u\big|_{I_n} \in \mathbb{P}_k(I_n,V) \quad \forall\, I_n \in \mathcal{M}_\tau\}, \tag{B.1}$$

where

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

and $(\mathcal{M}_\tau) = \{I_1,\dots,I_n\}$ is the *time mesh*. We introduce the discrete time test space

$$\mathbb{P}_k^{dc}(\mathcal{M}_\tau) := \{v \in L^2(I,V) : \, v\big|_{I_n} \in \mathbb{P}_k(I_n,V) \quad \forall\, I_n \in \mathcal{M}_\tau\} \tag{B.2}$$

consisting of piecewise polynomials of order $k-1$ which are globally discontinuous at the end points of the time intervals. Now, we multiply the first equation in (5.1) with a test function $v_\tau \in Y_\tau^k$, integrate over $\Omega \times I$, use Fubini's Theorem and partial space integration of the Laplacian term and obtain the following the "**global dG-C0($k$)-method**":

*Find $u_\tau \in \mathbb{P}_k^c(\mathcal{M}_\tau)$ such that $u_\tau(0) = u_0$,*

$$\int_0^T \Big\{ (d_t u_\tau(t), v_\tau(t))_\Omega + a(u_\tau(t), v_\tau(t)) \Big\} dt = \int_0^T (f(t), v_\tau(t))_\Omega \, dt \qquad \forall\, v_\tau \in \mathbb{P}_{k-2}^{dc}(\mathcal{M}_\tau) \tag{B.3}$$

and

$$d_t u_n^- - \triangle u_n^- = f(t_n) \qquad \forall\, n = 1, \ldots, N. \tag{B.4}$$

where

$$d_t u_n^- := \lim_{t \to t_n - 0} d_t u_\tau(t)$$

is the left-sided derivative at $t_n$. The discontinuity of the test space allows to calculate the solution of the dG-C0($k$)-method in a time-marching process. Again, we choose test functions $v_\tau(t) = v\psi_{n,i}(t)$ with an arbitrary time independent $v \in V$ and a scalar function $\psi_{n,i} : I \to \mathbb{R}$ which is zero on $I \setminus I_n$ and a polynomial of order less or equal $k-2$ on $I_n$. Then, we obtain the following "**$I_n$-problem of the exact dG-C0($k$)-method** ":

*Find $u_\tau|_{I_n} \in \mathbb{P}_k(I_n, V)$ such that*

$$\begin{aligned} u_\tau(t_{n-1}) &= u_{n-1}^-, \\ d_t u_n^- - \triangle u_n^- &= f(t_n), \\ \int_{I_n} \Big\{ (d_t u_\tau(t), v)_\Omega + a(u_\tau(t), v) \Big\} \psi_{n,i}(t)\, dt &= \int_{I_n} (f(t), v)_\Omega \, \psi_{n,i}(t)\, dt \qquad \forall\, \psi \in \mathbb{P}_{k-2}(I_n). \end{aligned} \tag{B.5}$$

If we apply the right-sided k-point Gauß-Radau quadrature formula, we get the following "**$I_n$-problem of the numerically integrated dG-C0($k$)-method**":

*Find $u_\tau|_{I_n} \in \mathbb{P}_k(I_n, V)$ such that*

$$\begin{aligned} u_\tau(t_{n-1}) &= u_{n-1}^-, \\ d_t u_n^- - \triangle u_n^- &= f(t_n), \\ \sum_{j=1}^k \hat{w}_j \Big\{ (d_t u_\tau(t_{n,j}), v)_\Omega + a(u_\tau(t), v) \Big\} \psi_{n,i}(t_{n,j})\, dt &= \sum_{j=1}^k \hat{w}_j \, (f(t_{n,j}), v)_\Omega \, \psi_{n,i}(t_{n,j})\, dt \quad \forall\, v \in V. \end{aligned} \tag{B.6}$$

To determine $u_\tau|_{I_n}$ we represent it by a polynomial ansatz

$$u_\tau(t) := \sum_{j=0}^k U_n^j \phi_{n,j}(t) \qquad \forall\, t \in I_n, \tag{B.7}$$

where the "coefficients" $U_n^j$ are elements of the Hilbert space $V$ and the real functions $\phi_{n,j} \in \mathbb{P}_k(I_n)$ are the Lagrange basis functions with respect to $k+1$ suitable nodal points. In the following, we specify the dG-C0($k$)-method for the cases $k = 2$ which leads to the **dG-C0(2)-method**.

### 2.1.1. dG-C0(2)-method

We apply the *2-point Gauß-Radau formula* with the points $\hat{t}_1 = -1/3$, $\hat{t}_2 = 1$, i.e.,

$$t_{n,1} = \frac{(t_{n-1}+t_n)}{2} - \frac{\tau_n}{6}, \quad t_{n,2} = t_n$$

and the reference weights $\hat{w}_1 = \frac{3}{2}, \hat{w}_2 = \frac{1}{2}$. Then, the $2\times 2$ block system on time interval $I_n$ reads: *For given* $\underline{U}_n^0 \in \mathbb{R}^{m_h}$, *find* $\underline{U}_n^1, \underline{U}_n^2 \in \mathbb{R}^{m_h}$ *such that*

$$\frac{\tau_n}{2} L\underline{U}_n^1 + M\underline{U}_n^2 = \frac{\tau_n}{2} F_n^2, \tag{B.8}$$

$$\sum_{j=0}^{2} \left\{ \sum_{k=1}^{2} \hat{w}_k \hat{\phi}_j'(\hat{t}_k) \right\} MU_n^j + \frac{\tau_n}{2}\hat{w}_1 \sum_{j=0}^{2} \hat{\phi}_j(\hat{t}_1) LU_n^j + \frac{\tau_n}{2}\hat{w}_2 LU_n^1 = \frac{\tau_n}{2}\left(\hat{w}_1 F_n^1 + \hat{w}_2 F_n^2\right), \tag{B.9}$$

where $M$, $L$ denote the mass and Laplacian matrix, respectively. $\underline{U}_n^1$ and $\underline{U}_n^2$ are the value and the derivative of the solution vector at the time discrete point $t_n$. $F_n^1$, $F_n^2$ are RHS vectors at point $t_{n,1}$ and $t_n$. Computing the constants $\beta_j := \hat{\phi}_j(\hat{t}_1)$, we obtain

$$\beta_0 := \frac{4}{9}, \quad \beta_1 := \frac{5}{9}, \quad \beta_2 := -\frac{4}{9}.$$

Thus, from (B.9) we get

$$M\left(U_n^1 - U_n^0\right) + \frac{3\tau_n}{4}\left(\frac{4}{9}LU_n^0 + \frac{5}{9}LU_n^1 - \frac{4}{9}LU_n^2\right) + \frac{\tau_n}{4}LU_n^1 = \frac{\tau_n}{4}\left(3F_n^1 + F_n^2\right).$$

The final block-system reads:

$$\begin{pmatrix} \frac{\tau_n}{2}L & M \\ M + \frac{2\tau_n}{3}L & -\frac{\tau_n}{3}L \end{pmatrix} \begin{pmatrix} \underline{U}_n^1 \\ \underline{U}_n^2 \end{pmatrix} = \begin{pmatrix} R_n^1 \\ R_n^2 \end{pmatrix} \tag{B.10}$$

where

$$\begin{aligned} R_n^1 &= \tfrac{\tau_n}{2} F_n^2 \\ R_n^2 &= \tfrac{\tau_n}{4}\left(3F_n^1 + F_n^2\right) + \left(M - \tfrac{\tau_n}{3}L\right)\underline{U}_n^0. \end{aligned}$$

## 2.2. Numerical results

In this section, we perform some numerical tests to show the accuracy of the dG-C0(2)-method. We consider the same problem (5.1) as a test example with the domain $\Omega := (0,1)^2$ and the right hand side

$$f(x,y,t) = x(1-x)y(1-y)e^t + 2[y(1-y) + x(1-x)]e^t,$$

which has been derived from the prescribed exact solution

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

The initial data is $u_0(x,y) = u(x,y,0)$. The behavior of the standard $L^2$-norm $\|\cdot\|_2 := \|\cdot\|_{L^2(I,L^2(\Omega))}$ and the discrete $L^\infty$-norm of the time discretization error $u(t) - u_\tau(t)$ over the time interval $I = [0,1]$ can be seen in Table B.1. The estimated value of the experimental order of convergence (EOC) is also calculated and compared with the theoretical order of convergence. From Table B.1, we see that the $L^2$-norm and the discrete $L^\infty$-norm of the error behave like order of 3. This means the experimental order of convergence confirms the theoretical results.

| | dG-C0(2) | | | |
|---|---|---|---|---|
| $1/\tau$ | $\|u-u_\tau\|_2$ | EOC | $\|u-u_\tau\|_\infty$ | EOC |
| 10 | 4.71E-07 | | 1.21E-06 | |
| 20 | 6.14E-08 | 2.94 | 1.74E-07 | 2.80 |
| 40 | 7.90E-09 | 2.96 | 2.34E-08 | 2.89 |
| 80 | 1.01E-09 | 2.98 | 3.04E-09 | 2.94 |
| 160 | 1.27E-10 | 2.99 | 3.87E-10 | 2.97 |
| 320 | 1.59E-11 | 2.99 | 4.89E-11 | 2.99 |
| 640 | 2.00E-12 | 3.00 | 6.14E-12 | 2.99 |
| 1280 | 2.49E-13 | 3.00 | 7.67E-13 | 3.00 |
| 2560 | 3.10E-14 | 3.01 | 9.54E-14 | 3.01 |
| 5120 | 3.61E-15 | 3.10 | 7.96E-15 | 3.58 |

**Table B.1**: $L^2$- and $L^\infty$-error norms for the test case.

# Bibliography


[1] D. N. Arnold, D. Boffi, and R. S. Falk. Approximation by quadrilateral finite elements. *Mathematics of Computation*, 71:909–922, 2002.

[2] A. K. Aziz and P. Monk. Continuous finite elements in space and time for the heat equation. *Math. Comp.*, 52(186):255–274, 1989. ISSN 0025-5718.

[3] E. R. Bank and T. Dupont. An optimal order process for solving finite element equations. *Mathematics of Computation*, 36(153):35 – 51, 1981.

[4] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[5] 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.

[6] D. Braess. *Finite elements: theory, fast solvers, and applications in solid mechanics*. Cambridge ; New York : Cambridge University Press, 3rd edition, 2007.

[7] S. C. Brenner and R. Scott. *The Mathematical Theory of Finite Element Methods*. Springer-Verlag, 1994.

[8] F. Brezzi and M. Fortin. *Mixed and Hybrid Finite Element Methods*, volume 15 of *Springer Series in Computational Mathematics*. Springer-Verlag, 1991.

[9] M. O. Bristeau, R. Glowinski, and Periaux J. Numerical methods for the Navier-Stokes equations. applications to the simulation of compressible and incompressible viscous flow. *Computer Physics Reports*, 6:73–187, 1987.

[10] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, Ltd, 2008. URL `http://doi.wiley.com/10.1002/0470868279`.

[11] C. Cuvelier, A. Segal, A. A. Steenhoven. *Finite element methods and Navier-Stokes equations*. D. Reidel Publishing Company, 1986. ISBN ISBN 90-277-2148-3.

[12] T. Cebeci, J. P. Shao, F. Kafyeke, and E. Laurendeau. *Computational Fluid Dynamics for Engineers*. Springer, 2005. ISBN 978-3-540-24451-6.

[13] P. Ciarlet and P. Lions. *Handbook of Numerical Analysis II. Part 1: Finite Element Methods*. North-Holland, 1991.

[14] G. G. Dahlquist. A special stability problem for linear multistep methods. *BIT Numerical Mathematics*, 3:27–43, 1963. ISSN 0006-3835. URL http://dx.doi.org/10.1007/BF01963532. 10.1007/BF01963532.

[15] H. Damanik, J. Hron, A. Ouazzi, and S. Turek. Monolithic Newton-multigrid solution techniques for incompressible nonlinear flow models. Technical report, Fakultät für Mathematik, TU Dortmund, 2011. Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 426, submitted to IJNMF.

[16] J. G. de Jalón and E. Bayo. *Kinematic and Dynamic Simulation of Multibody Systems. The Real-Time Challenge.* Mechanical engineering series. Springer-Verlag, New-York, 1994. ISBN 0-387-94096-0.

[17] J. Donea and A. Huerta. *Finite element methods for flow problems*. JohnWiley and Sons, New York, 2003.

[18] B. L. Ehle. *On Padé approximations to the exponential function and A-stable methods for the numerical solution of initial value problems*. University of Waterloo, 1969. URL http://books.google.de/books?id=BanINQAACAAJ.

[19] H. C. Elman, O. G. Ernst, and D. P. O'Leary. A multigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations. *SIAM J. Sci. Comput.*, 23:1291–1315, April 2001. ISSN 1064-8275. DOI: 10.1137/S1064827501357190. URL http://dl.acm.org/citation.cfm?id=587152.587192.

[20] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. *Computational differential equations*. Cambridge University Press, Cambridge, 1996.

[21] V. Girault and P. A. Raviart. *Finite Element Methods for Navier-Stokes equations*. Springer, Berlin-Heidelberg, 1986.

[22] R. Glowinski. Viscous flow simulation by finite element methods and related numerical techniques. In E. M. Murman & S. S. Abarbanel, editor, *Progress and supercomputing in computational fluid dynamics; Proceedings of U.S.-Israel Workshop, Jerusalem, Israel, December 1984 (A86-46787 22-34). Boston, MA, Birkhaeuser*, pages 173–210, 1985.

[23] W. Hackbusch. *Multi-Grid Methods and Applications.* Springer-Verlag, Berlin, 1985.

[24] J. Heywood, R. Rannacher, and S. Turek. Artificial boundaries and flux and pressure conditions for the incompressible Navier–Stokes equations. *International Journal for Numerical Methods in Fluids*, 22:325–352, 1996.

[25] C. Hirsch. *Numerical computation of internal & external flows: fundamentals of numerical discretization*. John Wiley & Sons, Inc., New York, NY, USA, 1988. ISBN 0-471-91762-1.

[26] M. H. Holmes. *Introduction to Numerical Methods in Differential Equations (Texts in Applied Mathematics)*. Springer, Ist edition, 2006. ISBN 0387308911.

[27] S. Hussain, F. Schieweck, and S. Turek. Higher order Galerkin time discretizations and fast multigrid solvers for the heat equation. Technical report, Fakultät für Mathematik, TU Dortmund, November 2010. Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 416.

[28] S. Hussain, F. Schieweck, and S. Turek. Higher order Galerkin time discretizations and fast multigrid solvers for the heat equation. *Journal of Numerical Mathematics*, 19(1):41–61, 2011.

[29] S. Hussain, F. Schieweck, and S. Turek. A note on accurate and efficient higher order Galerkin time stepping schemes for the nonstationary Stokes equations. Technical report, Fakultät für Mathematik, TU Dortmund, April 2011. Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 424.

[30] S. Hussain, F. Schieweck, and S. Turek. A note on accurate and efficient higher order Galerkin time stepping schemes for the nonstationary Stokes equations. *The Open Numerical Methods Journal*, 4:35–45, 2012.

[31] S. Hussain, F. Schieweck, and S. Turek. Higher order Galerkin time discretization for nonstationary incompressible flow. Technical report, Fakultät für Mathematik, TU Dortmund, January 2012. Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 444.

[32] S. Hussain, F. Schieweck, and S. Turek. Higher order Galerkin time discretization for nonstationary incompressible flow. In *Proceedings Enumath 2012 Leicester*. Wiley-VCH, 2012. accepted.

[33] S. Hussain, F. Schieweck, and S. Turek. An efficient and stable finite element solver of higher order in space and time for nonstationary incompressible flow. Technical report, Fakultät für Mathematik, TU Dortmund, 2012. Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 450.

[34] D. V. Hutton. *Fundamentals of Finite Element Analysis*. McGraw-Hill, 2004. ISBN 0072395362.

[35] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press New York, 1988. ISBN ISBN 0-521-34758-0.

[36] P. Klouck and F. S. Rys. Stability of the fractional step–θ–scheme for the nonstationary Navier–Stokes equations. *SIAM J. on Numerical Analysis*, 31(5):1312–1335, 1994.

[37] M. Köster. *A Hierarchical Flow Solver for Optimisation with PDE Constraints*. PhD thesis, TU Dortmund, Lehrstuhl III für Angewandte Mathematik und Numerik, 2011. Slightly corrected version with an additional appendix concerning prolongation/restriction.

[38] M. Köster. Robuste Mehrgitter-Krylowraum-Techniken für FEM-Verfahren. Master's thesis, Institut für Angewandte Mathematik und Numerik, Technische Universität Dortmund, 2004.

[39] M. Köster and S. Turek. The influence of higher order FEM discretisations on multigrid convergence. *Computational Methods in Applied Mathematics*, 6(2):221–232, 2006.

[40] G. Matthies and F. Schieweck. Higher order variational time discretizations for nonlinear systems of ordinary differential equations. Preprint 23/2011, Otto-von-Guericke Universität Magdeburg, Fakultät für Mathematik, 2011.

[41] G. Matthies and L. Tobiska. The inf-sup condition for the mapped $Q_k - P_{k-1}^{disc}$ element in arbitrary space dimensions. *Computing*, 69:119–139, October 2002. ISSN 0010-485X. DOI: 10.1007/s00607-002-1451-3. URL `http://dl.acm.org/citation.cfm?id=635904.635906`.

[42] S. Moaveni. *Finite Element Analysis Theory and Application with ANSYS (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2007. ISBN 0131890808.

[43] M. Möller. Mathematical and Practical Aspects of Finite Elements. Lecture Notes: Summer term, Institut für Angewandte Mathematik und Numerik, Technische Universität Dortmund, 2010.

[44] M. Möller. Finite Element Methods for Flow Problems. Lecture Notes: Winter term, Institut für Angewandte Mathematik und Numerik, Technische Universität Dortmund, 2010/11.

[45] M. Möller. *Adaptive high–resolution finite element schemes*. PhD thesis, Technische Universität Dortmund, December 2008.

[46] S. Müller, A. Prohl, R. Rannacher, and S. Turek. Implicit time–discretization of the nonstationary incompressible Navier–Stokes equations. In W. Hackbusch and G. Wittum, editors, *Proc. 10th GAMM-Seminar*. Vieweg, 1994.

[47] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003. ISBN 0898715342.

[48] Y. Saad and M. H. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3): 856–869, 1986. URL http://link.aip.org/link/SJOCE3/v7/i3/p856/s1&Agg=doi.

[49] F. Schieweck. A-stable discontinuous Galerkin-Petrov time discretization of higher order. *J. Numer. Math.*, 18(1):25 – 57, 2010.

[50] R. Schmachtel. *Robuste lineare und nichtlineare Lösungsverfahren für die inkompressiblen Navier–Stokes-Gleichungen*. PhD thesis, Universität Dortmund, Fakultät für Mathematik, Lehrstuhl 3 für Angewandte Mathematik und Numerik, June 2003.

[51] L. J. Segerlind. *Applied finite element analysis*. John Wiley & Sons, 1984.

[52] R. A. Shapiro. *Adaptive finite element solution algorithm for the Euler equations*, volume 32 of Notes on Numerical Fluid Mechanics. Vieweg, Braunschweig-Wiesbaden, 1991.

[53] R. Stenberg. Analysis of mixed finite element methods for the Stokes problem: A unified approach. *Mathematics of Computation*, 42(165):9–23, 1984.

[54] G. Strang. *Linear Algebra and Its Applications*. Brooks Cole, 3 edition, February 1988. ISBN 0155510053. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0155510053.

[55] V. Thomée. *Galerkin finite element methods for parabolic problems*, volume 25 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 2006. ISBN 978-3-540-33121-6; 3-540-33121-2.

[56] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, 1997. ISBN 0898713617. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0898713617.

[57] S. Turek. A comparative study of some time-stepping techniques for the incompressible Navier-Stokes equations: from fully implicit nonlinear schemes to semi-implicit projection methods. *Int. J. Numer. Meth. Fluids*, 22:987 – 1011, 1996.

[58] S. Turek. *Efficient solvers for incompressible flow problems. An algorithmic and computational approach*, volume 6 of *Lecture Notes in Computational Science and Engineering*. Springer, 1999.

[59] S. Turek and A. Ouazzi. Unified edge–oriented stabilization of nonconforming FEM for incompressible flow problems: Numerical investigations. *Journal of Numerical Mathematics*, 15(4):299–322, 2007.

[60] S. Turek and R. Rannacher. A simple nonconforming quadrilateral Stokes element. *Numerical Methods for Partial Differential Equations*, 8(2):97–111, 1992.

[61] S. Turek and M. Schäfer. Benchmark computations of laminar flow around cylinder. In E. Hirschel, editor, *Flow Simulation with High-Performance Computers II*, volume 52 of *Notes on Numerical Fluid Mechanics*, pages 547–566. Vieweg, January 1996.

[62] H. A. van der Vorst. BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644, March 1992. ISSN 0196-5204. DOI: http://dx.doi.org/10.1137/0913035. URL `http://dx.doi.org/10.1137/0913035`.

[63] S. P. Vanka. Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *Journal of Computational Physics*, 65(1):138 – 158, 1986.

[64] P. Wesseling. *An Introduction to Multigrid Methods*. John Wiley & Sons, Ist edition, 1992.

[65] H. Wobker and S. Turek. Numerical studies of Vanka-type smoothers in Computational Solid Mechanics. *Advances in Applied Mathematics and Mechanics*, 1(1):29–55, 2009.