

Time step size adaptation for unsteady aerodynamic flow simulations: Application to gust encounter simulations

Christian Lenz

Deutsches Zentrum für Luft- und Raumfahrt
Institut für Aerodynamik und
Strömungstechnik
Abteilung C²A²S²E, Braunschweig



DLR

Deutsches Zentrum
für Luft- und Raumfahrt



Herausgeber

Deutsches Zentrum
für Luft- und Raumfahrt e. V.
Bibliotheks- und
Informationswesen
D-51170 Köln
Porz-Wahnheide
Linder Höhe
D-51147 Köln

Telefon

(0 22 03) 6 01 - 44 44

Telefax

(0 22 03) 6 01 - 47 47

Als Manuskript gedruckt.
Abdruck oder sonstige Verwendung
nur nach Absprache mit dem DLR gestattet.

ISSN 1434-8454

RANS equations, time step adaptation, unsteady flow simulations, gust encounter

(Published in English)

Christian Lenz

Institute of Aerodynamics and Flow Technology of the DLR, Braunschweig

**Time step size adaptation for unsteady aerodynamic flow simulations:
Application to gust encounter simulations**

Doctoral thesis Technische Universität Braunschweig

*DLR-Forschungsbericht 2018-25, 2018, 154 pages, 36 figures., 15 tables., 66 references,
34.00 €*

This thesis investigates suitable time integration, error estimation and step size control algorithms for the temporal discretization and approximate solution of the unsteady Euler, Navier-Stokes, and Reynolds-Averaged-Navier-Stokes (RANS) equations. The focus lies on applications with an induced unsteadiness such as gust encounter simulations.

Embedded Runge-Kutta methods are used to get an estimate of the temporal discretization error. Based on this error, different adaptation algorithms are used to adjust the time step size. The effect of the time step size adaptation is demonstrated for the gust encounter simulations resulting in small time step sizes during the actual encounter with larger time steps before and afterwards. A new time step size limiter is introduced which uses the information from a rejected step to avoid subsequent rejections.

Diagonally implicit Runge-Kutta methods with constant and adaptive time step sizes are investigated and compared to backward difference formulae. To solve the linear systems several iterative solution methods are compared together with a discussion on different stage value predictors which yield suitable initial values for the iterative methods.

RANS Gleichungen, Zeitschrittadaption, instationäre Strömungssimulation, Böeninteraktion

Christian Lenz

Institut für Aerodynamik und Strömungstechnik des DLR, Braunschweig

Zeitschrittadaption für instationäre Strömungssimulationen mit Anwendung auf Böensimulationen

Dissertation Technische Universität Braunschweig

DLR-Forschungsbericht 2018-25, 2018, 154 Seiten, 36 Bilder, 15 Tabellen, 66 Literaturstellen, 34,00 € zzgl. MwSt.

Diese Arbeit untersucht Zeitintegrationsverfahren zusammen mit Zeitschrittkontrollmethoden beruhend auf Fehlerschätzern zur Lösung der Euler, Navier-Stokes und Reynolds-gemittelten Navier-Stokes Gleichungen im Bereich der instationären Strömungssimulationen. Schwerpunkt ist die Anwendung auf Simulationen mit Böeninteraktion.

Eingebettete Runge-Kutta Verfahren dienen zur Schätzung des Fehlers, der genutzt wird, um verschiedene Zeitschrittadaptionen zu implementieren und zu testen. Der Effekt der Zeitschrittadaption wird unter anderem am Beispiel einer Böeninteraktion demonstriert. Ein neu entwickelter Zeitschrittweitenbegrenzer wird vorgestellt, der die Anzahl verworfener Zeitschritte signifikant reduziert.

Darüber hinaus werden diagonal implizite Runge-Kutta Verfahren sowohl mit konstanten als auch adaptiven Zeitschritten untersucht und mit Backward-Difference Verfahren verglichen. Zum Lösen der linearen Gleichungssysteme werden verschiedene Lösungsverfahren untersucht und verglichen. Außerdem werden geeignete Verfahren für die Bestimmung der Initialwerte zum Lösen der Gleichungssysteme diskutiert.

Forschungsbericht 2018-25

Time step size adaptation for unsteady aerodynamic flow simulations: Application to gust encounter simulations

Christian Lenz

Deutsches Zentrum für Luft- und Raumfahrt
Institut für Aerodynamik und
Strömungstechnik
Braunschweig

154 Seiten
36 Bilder
15 Tabellen
66 Literaturstellen



DLR

Deutsches Zentrum
für Luft- und Raumfahrt

**Time step size adaptation for unsteady aerodynamic flow simulations:
Application to gust encounter simulations**

Von der
Carl-Friedrich-Gauß-Fakultät
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades eines
Doktoringenieurs (Dr.-Ing.)

genehmigte Dissertation

von
Christian Lenz
geboren am 13.03.1985
in Limburg a.d. Lahn

Eingereicht am:	24.11.2017
Disputation am:	11.07.2018
1. Referentin/Referent:	Prof. Dr. Hermann G. Matthies
2. Referentin/Referent:	Prof. Dr. Andreas Meister

2018

Abstract

This thesis presents an investigation of suitable time integration, error estimation and step size control algorithms for the temporal discretization and approximate solution of the unsteady Euler, Navier-Stokes, and Reynolds-Averaged-Navier-Stokes (RANS) equations in the framework of compressible computational fluid dynamics (CFD) for aerospace applications. The focus lies on applications with an induced unsteadiness, in particular a turbulent pitching oscillation as well as inviscid, laminar, and turbulent gust encounter simulations.

Embedded Runge-Kutta methods are used to get an estimate of the temporal discretization error. Based on this error, different adaptation algorithms are used to adjust the time step size during the simulation. The effect of the time step size adaptation is demonstrated for the gust encounter simulations resulting in small time step sizes during the actual encounter with larger time steps before and afterwards. To reduce the number of step rejections for the gust encounter simulations, a new time step size limiter is introduced which uses the information from a rejected step to avoid subsequent rejections. Furthermore, the effect of different adaptation controllers and error norms are investigated.

Especially for turbulent flow, the systems of ordinary differential equations (ODEs), arising from the finite volume discretization in space, are stiff and implicit A-stable methods should be used in order to allow for reasonably large time step sizes. In this thesis, different diagonally implicit Runge-Kutta (DIRK) methods with constant and adaptive time step sizes are investigated and compared to backward difference formulae (BDFs). The usage of implicit methods requires the solution of several systems of nonlinear equations. Depending on the application, these systems can get very large in the field of CFD. Therefore, suitable iterative solution methods have to be used. Some available methods have been adapted for the DIRK methods used in this thesis. A comparison of these methods together with a discussion on different stage value predictors which yield suitable initial values for the iterative methods is also included.

Kurzfassung

In dieser Arbeit werden adäquate Zeitintegrationsmethoden, Fehlerschätzer und Zeitschrittweitenadaptionalgorithmen für die zeitliche Diskretisierung und numerische Lösung von instationären Euler, Navier Stokes und Reynolds gemittelten Navier-Stokes Gleichungen im Bereich der aerodynamischen Strömungssimulation untersucht. Der Schwerpunkt der Anwendungen liegt dabei auf Simulationen mit von außen angeregten Veränderungen, insbesondere die Simulation einer oszillierenden Nickbewegung, sowie der Simulation von reibungsfreien, laminaren und turbulenten Böeninteraktionen.

Eingebettete Runge-Kutta Methoden werden zur Schätzung des Zeitdiskretisationsfehlers genutzt. Mit verschiedenen Adaptionalgorithmen wird die Zeitschrittweite basierend auf diesem Fehlerschätzer während der Simulation angepasst. Anhand von Böensimulationen wird der Effekt der Zeitschrittadaptionalgorithmen demonstriert. Sie führen zu kleinen Zeitschrittweiten während dem Auftreffen der Böe und größeren Zeitschrittweiten davor und danach. In dieser Arbeit wird ein neuer Zeitschrittweitenbegrenzer eingeführt, der die Informationen eines verworfenen Zeitschrittes nutzt um folgende Zeitschrittweiten zu begrenzen und die Anzahl der verworfenen Schritte zu reduzieren. Außerdem wird der Einfluss verschiedener Adaptionalgorithmen und Fehlnormen untersucht.

Besonders für turbulente Strömungen entstehen bei der Finite-Volumen Diskretisierung der Gleichungen steife Systeme von gewöhnlichen Differentialgleichungen. Daher müssen implizite Methoden zur Lösung der Differentialgleichungen genutzt werden um Zeitschrittweiten angemessener Größe zu erlauben. In der vorliegenden Arbeit werden verschiedene diagonalimplizite Runge-Kutta Methoden sowohl mit konstanten, wie auch mit variablen Zeitschrittweiten genutzt und mit BDFs verglichen.

Durch die Anwendung impliziter Methoden zur Lösung der Systeme gewöhnlicher Differentialgleichungen entstehen mehrere nichtlineare Gleichungssysteme, die je nach Anwendungsfall sehr groß werden können. Zur approximativen Lösung dieser Systeme müssen daher passende iterative Methoden genutzt werden. Im Zuge dieser Arbeit wurden die im genutzten Strömungslöser vorhandenen Algorithmen zum Lösen der aus den

diagonalimpliziten Runge-Kutta Verfahren entstehenden Gleichungssysteme angepasst. Außerdem ist ein Vergleich dieser Methoden sowie eine Erörterung verschiedener Methoden zur Anfangswertberechnung beinhaltet.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Previous Work	2
1.2.1	Embedded Runge-Kutta methods for error estimation	3
1.2.2	Step size control algorithm	4
1.2.3	Step size control for flow simulations	4
1.3	Contributions	6
1.4	Thesis Overview	7
2	Governing equations	11
2.1	Euler equations	11
2.2	Reynolds-Averaged-Navier-Stokes equations	13
2.3	Spatial discretization and TAU code	16
2.4	Moving grid and basis transformation	20
2.5	Disturbance velocity approach for gust encounter simulations	21
2.6	Boundary conditions	24
2.7	Summary of discrete equation	28
3	Numerical methods for ordinary differential equations	29
3.1	Introduction to Runge-Kutta methods	30
3.2	Stiff equations and stability of Runge-Kutta methods	32
3.2.1	Linear stability	33
3.2.2	Nonlinear stability	34
3.2.3	Order reduction and stage order	36
3.3	Multistep methods	37
3.4	Comparison of Multistep and Runge-Kutta methods	39
3.5	Implemented methods	40
3.5.1	Embedded explicit Runge-Kutta methods	40

3.5.2	Embedded diagonally implicit Runge-Kutta methods	42
3.5.3	Backward Difference Formulae	44
4	Solution of nonlinear algebraic systems	47
4.1	Solution methods	47
4.2	Stage value predictor	51
4.3	Truncation criteria	52
5	Time step size adaptation algorithms	55
5.1	Time step estimator	55
5.1.1	Elementary (integral) control	58
5.1.2	Advanced controllers	60
5.1.3	New time step size limiter	65
5.1.4	Summary of adaptation algorithms	69
5.2	Error estimator	69
5.3	Error norm or functional	72
6	Numerical test cases	75
6.1	Ehrenfried vortex	75
6.2	Kármán vortex street	76
6.3	NACA0012 laminar steady	78
6.4	NACA0012 turbulent pitching oscillation	78
6.5	Gust encounter	81
6.5.1	NACA0012 inviscid	81
6.5.2	SDM fighter configuration inviscid	81
6.5.3	Transport aircraft half model turbulent	84
7	Numerical results	87
7.1	Temporal order of convergence of implemented methods	88
7.2	Grid refinement study for explicit and implicit methods	91
7.2.1	Application of explicit methods	92
7.2.2	Application of implicit methods	93
7.3	Adaptation algorithm	95
7.3.1	Verification	95
7.3.2	Comparison of algorithms	95
7.3.3	Error functional	106

7.4	Solution of nonlinear systems	110
7.4.1	Solution methods	110
7.4.2	Stage value predictors	112
7.5	Investigation of DIRK method properties	114
7.6	Comparison of BDFs and (adaptive) DIRK methods	120
8	Conclusion	127
	Acronyms	131
	Bibliography	139

CHAPTER 1

Introduction

1.1 Motivation

Nowadays, the numerical simulation of flow around an aircraft using compressible computational fluid dynamics (CFD) is an essential part of aircraft development and testing. In CFD the solution of the Euler, Navier-Stokes, or Reynolds-Averaged-Navier-Stokes (RANS) equations, is approximated. These equations can be represented by a set of partial differential equations (PDEs) or integral equations. In general, one can distinguish between unsteady and steady problems, referring to flows which do and do not change over time, respectively. Due to the high computational costs of numerical flow simulations, the focus of CFD in aerospace applications was first set on steady configurations. With the tremendous increase in computational power in the last years, the computation of unsteady flows is getting more practical. However, the simulation of complex configurations is still very time consuming. Therefore, the improvement of existing algorithms in terms of efficiency and accuracy is a topic of ongoing research.

To solve the PDEs numerically, a standard procedure is to employ the method of lines approach, first discretizing the PDEs in space leaving a set of ordinary differential equations (ODEs) in time to be solved afterwards. For unsteady computations, the discretization in time to approximatively solve these ODEs is of particular interest. In CFD codes it is common to use a uniform discretization in time, i. e. a constant time step size, often in combination with a backward difference formula (BDF) as ODE solver. On the other hand, there is a well established mathematical theory available regarding the solution of ODEs [16, 36, 37] which recommends the usage of an adaptive step size selection, i. e. an automatism to calculate appropriate (time) step sizes during the simulation. Typically, this is done by a local time step size adaptation algorithm, which uses an estimate of the local temporal error to select the next step size. More recently, a time

step size adaptation based on the global error, taking error propagation into account, has been developed [31, 54]. However, with this approach the error estimation is done after the simulation. This requires the user to rerun the simulation with the adapted step sizes. In this thesis the focus is on step size adaptation using a local error estimate.

The advantages of using such a technique as compared to using the current standard constant time step size are as follows:

1. The user does not have to specify a time step size. Especially for complex simulations where the choice of an appropriate time step size is not obvious, this could lead to less “trial and error”.
2. An error estimate is available, giving the user an indication on the accuracy of the simulation in terms of temporal discretization.
3. An adaptive time step size selection allows for larger time step sizes in parts of the simulation, while ensuring a sufficiently fine temporal discretization when necessary. Depending on the configuration, this could lead to a speed up of the simulation.

To apply the time step size adaptation, diagonally implicit Runge-Kutta (DIRK) methods are used as ODE solvers instead of BDFs. These methods are better suited and more efficient for time adaptive flow computations (cf. Sections 1.2.3 and 3.4). For a time adaptive simulation, it is necessary to

1. compute the flow solution of the next time step
2. estimate the local temporal discretization error
3. apply a suitable adaptation algorithm to compute the new time step size.

1.2 Previous Work

In this section some previous work done in the research field of this thesis is stated. First developments on error estimation using embedded Runge-Kutta methods for ordinary differential equations are reviewed. Second, previous work on the control algorithm is summarized, and finally some research results for the application to computational fluid dynamics are discussed.

1.2.1 Embedded Runge-Kutta methods for error estimation

According to Butcher [15], the research on error estimation and step size control for ordinary differential equations using Runge-Kutta methods started in 1957. Merson [55] proposed a first embedded explicit Runge-Kutta method with 5-stages of order 4 with an embedded method of order 3 for error estimation. Afterwards, further research in the field of explicit embedded Runge-Kutta methods was done by Fehlberg, who developed methods up to order 7 [29]. A famous method named after him has 6 stages and is of order 5 with a fourth order method embedded. Further major contributions in this field are due to Dormand and Prince [26]. They concentrated on finding methods with an extended region of absolute stability and proposed 7-stage methods of order 5 with a fourth order method embedded. The method has effectively only 6 function evaluations, since the last evaluation can be reused as the first evaluation in the next step. This popular feature is known as the *first same as last* property or *Fehlberg-trick* named after E. Fehlberg who used this property in the context of Runge-Kutta-Nyström methods [30].

When using explicit methods for stiff differential equations stability rather than accuracy limits the time step size. In this context, implicit Runge-Kutta methods were proposed by Kuntzmann [47] and by Butcher [12]. For a system of ODEs with d equations an s -stage implicit Runge-Kutta method requires to solve a (potentially nonlinear) system of equations of size ds in each step. This can be very costly. In order to reduce the computational cost, DIRK methods have been introduced in the 1970's, see [1, 57]. They require to solve s (nonlinear) systems of equations with only d equations each.

In 1974 Prothero and Robinson [58] pointed out, that for certain stiff equations many Runge-Kutta methods suffer from an order reduction phenomenon and introduced a concept of stiff order of a method. Frank, Schneid, and Ueberhuber [32] extended this idea to the B-convergence theory. Burrage, Hundsdorfer, and Verwer [11] then showed results on the connection between B-convergence and the minimal order of all stages of a Runge-Kutta method (the stage order). This shows that for stiff equations not only the classical order of a Runge-Kutta method is important, but also its stage order. However, they also note that the stage order of a DIRK method can not exceed one.

In order to increase the stage order, while almost retaining the computational efficiency of a DIRK method, Burrage [10] and Butcher [14] introduced singly implicit Runge-Kutta (SIRK) methods with an embedded method for error estimation. These are methods with a Runge-Kutta matrix with only one eigenvalue. Through a transformation introduced by Butcher [13] these methods can be computed efficiently.

An other idea to increase the stage order without the need of a transformation is

to use an explicit first stage singly diagonal implicit Runge-Kutta (ESDIRK) method. Such singly diagonal implicit Runge-Kutta (SDIRK) methods with an additional explicit stage were already used by Cooper and Sayfy [22], but a more sophisticated introduction reusing the last residual for the next step was given by Butcher and Chen [17]. Later, Kværnø [49] constructed stiffly accurate embedded ESDIRK methods for error estimation.

1.2.2 Step size control algorithm

In this subsection a short overview of the work done on step size control algorithms is given, mainly according to the review of G. Söderlind [60]. In the early years of step size control a common and quite successful algorithm called *elementary local error control* or *integral (I) control* was used (c.f. [33, p.156]). The algorithm is based on the assumptions that the step size is within the asymptotic region of the error estimator and has a slow variation. Then, the local optimal next step size is computed. However, these assumptions are in general not fulfilled in practice and oscillations of the step size may occur (c.f. [35]). These oscillations can especially be observed for step sizes at the stability limit of the error estimator [60]. Furthermore, they may even affect the numerical solution of the differential equations (c.f. [34]). Due to this fact, Hall [38, 39] examined the stability of a numerical method in combination with its control algorithm. Following, Higham and Hall [42] investigated Runge-Kutta methods with an improved stability together with an integral controller.

Instead of adapting the numerical method to the controller, Gustafsson, Lundh and Söderlind investigated controllers matching the numerical method. In [35] they found that using a proportional-integral (PI) controller with proper parameters results in a more robust performance and less oscillations of the step size. Söderlind [61] further investigated in this topic and suggested various step size control algorithms based on the theory of digital filters. An implementation of these techniques in the ODE code *DASSL* [9], which shows improved stability, can be found in [62].

1.2.3 Step size control for flow simulations

In the field of computational fluid dynamics the research on temporal discretization techniques and time step size control using embedded methods is a topic of current research. In 2003, Carpenter et al. [21] compared BDFs with ESDIRK methods without step size control for fluid dynamic simulations. Later on, they added step size control to the methods [20]. As test cases they studied a laminar flow around a cylinder computed

with the code TLNS2D and a turbulent flow around both a cylinder and a NACA0012 profile using the codes TLNS2D and FUN2D. They conclude that for the laminar test case the higher order ESDIRK (order 3, 4 and 5) schemes outperform the BDF2 method in terms of efficiency. They also state that the BDF3 method can compete with those schemes, however it shows an irregular behavior possibly due to lack of A-stability. For the turbulent test cases the BDF schemes perform slightly better than the ESDIRK schemes. Especially at low precisions the BDF2 scheme has the best work-precision ratio. The authors state that the decreased performance of the ESDIRK schemes is partly due to the bad convergence of the algebraic solver in the FUN2D code. For the BDF schemes significantly less inner iterations are needed.

A further article on the temporal integration techniques in CFD is due to Ashcroft et al. [3]. They compared various ESDIRK methods with BDFs in the field of turbomachinery and conclude that the ESDIRK methods are robust and deliver good convergence rates.

In the field of incompressible Navier-Stokes equations research on time step size control was done by John and Rang [45]. They compare Θ -schemes with ESDIRK and Rosenbrock-Wanner schemes for a 2D flow around a cylinder. The authors state that the Rosenbrock-Wanner schemes outperform both other schemes in terms of efficiency. In [7], Blom et al. come to a similar conclusion in the field of compressible Navier-Stokes equations, stating that the investigated Rosenbrock-Wanner schemes perform better than the ESDIRK schemes in the case of a laminar flow around a cylinder. In contrast to that, the investigations by Birken et al. in [5] and [4] show a better performance of the ESDIRK schemes. In the latter one, instabilities of the Rosenbrock-Wanner scheme in case of higher tolerances were observed in case of a laminar flow around a cross section of a wind turbine possibly due to an insufficient approximation of the Jacobian or some nonlinear stability problem. The ESDIRK schemes didn't show such problems. Apparently, the performance of the Rosenbrock-Wanner and ESDIRK schemes depends significantly on the test cases. Furthermore, the solution of the nonlinear and linear systems in the case of ESDIRK and Rosenbrock-Wanner methods, respectively, the choice of the step size adaptation strategy, as well as initial value and termination criteria choices for the solution of the nonlinear equations are important factors for a performance comparison (cf. [20, Chapter 5]).

For CFD problems, results from the literature show that so far no superior method was found in general and that results are very much case dependent. This justifies further research.

1.3 Contributions

This thesis presents an investigation of suitable time integration, error estimation and step size control algorithms for the temporal discretization and approximate solution of the Euler, Navier-Stokes, and RANS equations in the framework of compressible CFD for aerospace applications. The contributions are detailed in the following.

For the aerodynamic fluid flow applications, the turbulent adaptive unstructured (TAU) code, developed at the “Deutsches Zentrum für Luft und Raumfahrt e.V.”, has been used as flow solver. In order to investigate in adaptive time step size algorithms, the TAU code was extended. A general framework allowing the use of embedded explicit and diagonally implicit Runge-Kutta methods has been implemented and the existing nonlinear system solvers have been adapted for the use on the implicit Runge-Kutta stages. Furthermore, several time step size adaptation algorithms, error norms and stage value predictors have been implemented and the code has been structured to work with adaptive time steps and step rejections.

The TAU code has been validated and used for inviscid, laminar and turbulent flow simulations. In contrast to the existing literature, which mainly considers inherently unsteady flow applications [4, 7, 20], applications with an induced unsteadiness like a forced pitching oscillation or a gust encounter are considered as numerical examples. The gust encounter simulations range from a 2-D inviscid subsonic NACA0012 airfoil up to a 3-D turbulent transonic half model configuration of a transport aircraft. Especially in the gust encounter applications, significant changes in the time step size during the simulation are expected depending on the position of the gust. Therefore, the choice of the adaptation algorithm is of increasing interest and an investigation of different algorithms is presented. For gust encounter applications, unnecessary repeated step size rejections have been observed which lead to the development of a new time step size limiter based on information of previously rejected steps.

Especially for turbulent flows, the resulting nonlinear systems of equations are hard to solve. The choice of appropriate initial values and termination criteria can have a significant impact on the required computational expenses. Therefore, the influences of different stage value predictors and termination criteria are investigated. Obviously, the solution methods to solve the systems of equations are of great importance. While the development of new methods in this field is not in the scope of this thesis, investigations regarding the existing methods in TAU have been done.

Regarding the solution methods for ODEs, the main focus in this thesis is on DIRK methods. Several methods with different properties are evaluated. This includes stability

properties, convergence order investigations, recommendations on the c coefficients of a Runge-Kutta method, and investigations on the influence of the Runge-Kutta coefficients on the solution of the resulting nonlinear algebraic systems. A comparison to explicit Runge-Kutta (ERK) methods and existing constant time stepping BDFs is also included. Other methods to solve systems of ODEs, such as the popular Rosenbrock-Wanner schemes (cf. Section 1.2.3), have not been implemented. An evaluation of these methods, especially regarding their ability to cope with the nonlinearities in turbulent transonic flow simulations — see the problems for high tolerances in [4, Section 7.2] — is an interesting topic of further research.

1.4 Thesis Overview

The thesis is structured as follows: First, the governing Euler, Navier-Stokes, and RANS equations are introduced along with the utilized spatial discretization in Chapter 2. This includes a description of the treatment of moving grids as well as the simulation of gusts. The spatial discretization leads to a system of ODEs which is solved numerically by the adaptation algorithm as sketched in Figure 1.1. Starting with an initial solution and time step size, an embedded Runge-Kutta method is used to compute a new solution for the next time step as well as a corresponding error estimate. Based on this error estimate, the adaptation algorithm then computes a new time step size. Furthermore, it is decided whether the step is accepted or has to be rejected and recomputed. Only if it is accepted, the solution vector is updated. Otherwise, the current step has to be recomputed with the new time step size. Algorithmic details on this solution process are described in chapters 3 to 5 as follows:

Chapter 3 introduces (embedded) Runge-Kutta methods used in the time adaptive solution process to solve the system of ODEs numerically. They also provide a second solution for error estimation which can be computed with almost no numerical overhead. A special focus lies on methods suitable to solve stiff equations and the corresponding stability notions. Furthermore, multistep methods, especially BDFs are considered as an alternative to Runge-Kutta methods. Both options are compared and the specific methods used in this thesis are stated.

In the case of DIRK methods, it is necessary to solve a nonlinear system of algebraic equations in each stage of the method, where the diagonal of the Butcher scheme is $\neq 0$. Therefore, Chapter 4 introduces iterative solution methods used in this thesis. Furthermore, stage value predictors and truncation criteria are introduced, which specify the initial value and the stopping criteria for the iterative solution method.

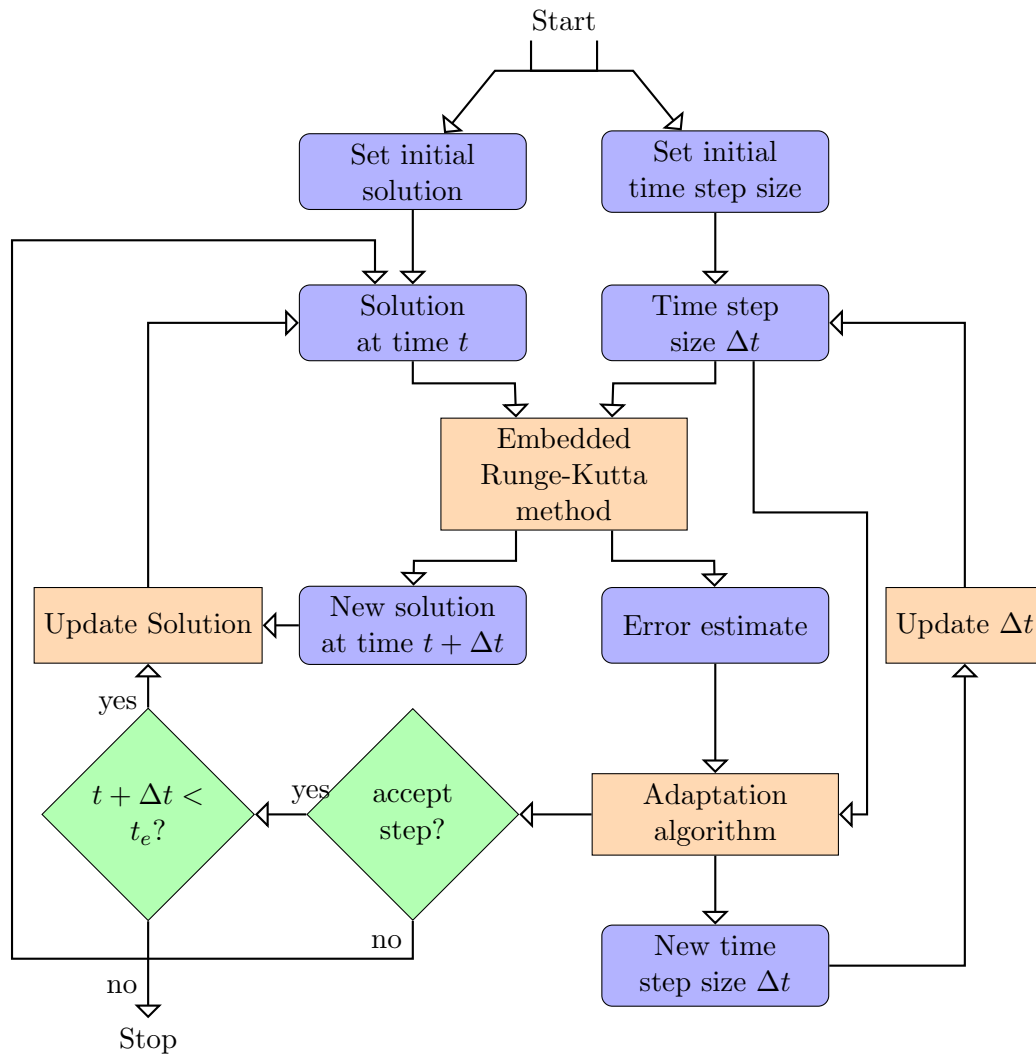


Figure 1.1: Overview of the time step size adaptation algorithm used in this thesis.

The focus in Chapter 5 is on the theory of adaptive time step size selection, including several step size controllers like the PI controller, predictive (P) controller, and digital filter (DF). Furthermore, the new developed time step size limiter is derived in Section 5.1.3. Afterwards, the error estimation is explained including the utilized error norm or functional, which maps the approximated error in the flow solution to a scalar value needed by the adaptation algorithm.

In Chapter 6, several aerodynamic fluid flow test cases are introduced. The cases range from a convection of a vortex over an inherently unsteady vortex shedding to test cases with an induced unsteadiness, in particular a pitching oscillation of an airfoil and several gust encounter cases. Inviscid, laminar and turbulent cases are considered ranging from small two dimensional grids up to the large grid of a three dimensional transport aircraft half model.

Afterwards, the test cases are investigated in Chapter 7. The order of convergence of the methods implemented in the TAU code are verified. Afterwards, the effect of grid refinement for explicit and implicit methods is shown. As a main part of the investigations, the different adaptation algorithms, controllers and error functionals are compared in Section 7.3 including a demonstration of the effect of the new limiter. Section 7.4 presents a comparison of the different solution methods and stage value predictors introduced in Chapter 4. Finally, a discussion of different properties of Runge-Kutta methods as well as a comparison of constant and adaptive DIRK methods with BDFs for the problems at hand is given.

The thesis is closed by a discussion of the results in Chapter 8

CHAPTER 2

Governing equations

As mentioned before, the main area of application governed in this work is in the field of computational fluid dynamics (CFD). The equations to be solved in the numerical simulations are either the Euler equations, the Navier-Stokes equations, or the Reynolds-Averaged-Navier-Stokes (RANS) equations. To this end, the CFD solver turbulent adaptive unstructured (TAU) is used, a code developed at the Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR). In the following, the governing equations as well as the discretization used in the TAU-code are stated, following the notation in the publications by Langer [50, 52, 53]. For a detailed derivation of the equations see e.g. the book of J. Blazek [6]. Since most of the test cases presented in this thesis either use the Euler or the RANS equations, these equations are introduced in detail in Sections 2.1 and 2.2, respectively. However, there is also one laminar test case, approximately solving the Navier-Stokes equations. Therefore, Remark 2.4 explains how to revert the RANS equations to the Navier-Stokes equations.

2.1 Euler equations

To describe the flow of an inviscid fluid in three dimensional space, consider a domain $\Omega \subset \mathbb{R}^3$ representing a control volume in the flow field for a given coordinate system.

Definition 2.1. *An open subset $\Omega \subset \mathbb{R}^n$ is called connected, if it cannot be decomposed in two open nonempty disjoint subsets, i. e. there do not exist $\Omega_1, \Omega_2 \subset \Omega$ open with*

$$\Omega_1 \neq \emptyset, \quad \Omega_2 \neq \emptyset, \quad \Omega_1 \cup \Omega_2 = \Omega, \quad \Omega_1 \cap \Omega_2 = \emptyset \quad (2.1)$$

Definition 2.2. *A subset $\Omega \subset \mathbb{R}^n$ is called domain, if it is open and connected.*

Here, the equations are considered in a cartesian inertial coordinate system with Ω

fixed relative to that system. Changes for moving control volumes are addressed in Section 2.4. For $\mathbf{x} \in \Omega$ and time $t \in [0, \infty)$ the density of the fluid is denoted by $\rho(\mathbf{x}, t)$, its velocity by

$$\mathbf{u}(\mathbf{x}, t) = (u_1(\mathbf{x}, t), u_2(\mathbf{x}, t), u_3(\mathbf{x}, t))^T, \quad (2.2)$$

and its total energy per unit mass by $E(\mathbf{x}, t)$. The dynamics of the inviscid fluid in conservative variables

$$\bar{\mathbf{W}} := (\rho, \rho u_1, \rho u_2, \rho u_3, \rho E)^T \quad (2.3)$$

are described by the Euler equations

$$\frac{d}{dt} \int_{\Omega} \bar{\mathbf{W}} d\mathbf{x} + \int_{\partial\Omega} \bar{\mathcal{F}}_c \mathbf{n} ds = 0. \quad (2.4)$$

Here, $\partial\Omega$ represents the boundary of Ω , $\mathbf{n} = (n_1, n_2, n_3)^T$ the unit outer normal vector on the boundary and

$$\bar{\mathcal{F}}_c = \begin{pmatrix} \rho \mathbf{u}^T \\ \rho u_1 \mathbf{u}^T + p(1, 0, 0) \\ \rho u_2 \mathbf{u}^T + p(0, 1, 0) \\ \rho u_3 \mathbf{u}^T + p(0, 0, 1) \\ \rho H \mathbf{u}^T \end{pmatrix} \quad (2.5)$$

the convective flux tensor. Hence,

$$\bar{\mathcal{F}}_c \mathbf{n} = \begin{pmatrix} \rho V \\ \rho u_1 V + p n_1 \\ \rho u_2 V + p n_2 \\ \rho u_3 V + p n_3 \\ \rho H V \end{pmatrix} \quad (2.6)$$

with the normal velocity $V = \mathbf{u}^T \mathbf{n}$. Furthermore,

$$H = E + \frac{p}{\rho} \quad (2.7)$$

is the total enthalpy and

$$p = (\gamma - 1)\rho \left(E - \frac{\|\mathbf{u}\|_2^2}{2} \right) \quad (2.8)$$

the pressure for a perfect gas. The constant γ is the gas dependent ratio of specific heat. In this thesis, a value of $\gamma = 1.4$ is used, which is the approximate value for air.

2.2 Reynolds-Averaged-Navier-Stokes equations

In this section, the flow of a *viscous* fluid in three dimensional space is considered. The dynamics of such a fluid can be described by the Navier-Stokes equations. However, especially for turbulent flow, a very fine spatial discretization is in general necessary to capture all relevant physical phenomena occurring in a viscous flow. That is why a modified set of equations, the RANS equations are considered for turbulent flows.

The idea to derive the RANS equations from the Navier-Stokes equations is to decompose the flow variables in a mean part and a fluctuating part. The flow equations are then solved for the mean part, such that small scale phenomena are only described by average values. The fluctuations can be omitted if they appear linearly, since the average of the fluctuations is, by definition, zero. However, additional terms are introduced. For example, the square of the fluctuations is not equal to zero in average. Hence, there are additional variables and equations, called turbulence equations, which are necessary to close the system of equations.

To find appropriate equations is the topic of turbulence modeling and will not be further discussed here. In the following, the RANS equations with a one equation turbulence model, the modified Spalart-Allmaras model [2], which is used for the turbulent test cases in this thesis, are stated. The resulting set of equations is a generalization of the Euler equations including an additional viscous flux, source terms and one turbulence equation.

Remark 2.3 (Averaging procedure). *Two different averaging procedures are used for the definition of the mean part of a flow variable, namely Reynolds averaging and Favre averaging (also called mass averaging). In both cases the flow variable is assumed to be a (discrete) random variable and the mean value is defined as the expectation of the random variable. The Reynolds averaging for a flow variable W_i is denoted by $\langle W_i \rangle$ and given by*

$$\langle W_i \rangle(\mathbf{x}) := \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{j=1}^N (W_i)_j(\mathbf{x}).$$

The Favre average of W_i denoted by $\langle W_i \rangle_\rho$ is given by

$$\langle W_i \rangle_\rho := \frac{\langle \rho W_i \rangle}{\langle \rho \rangle}.$$

For the RANS equations stated in the following, the Reynolds averaging is used for the density and pressure, while Favre averaging is used for the velocity, Energy and

temperature. For the sake of readability, the average operator symbols are dropped for the rest of the thesis. When dealing with the RANS equations, the variables should always be interpreted as averaged values. More details on the averaging procedure can be found e. g. in [6, Chapter 7] or in [59, Section 2.2]

If not stated otherwise, the notation from Section 2.1 is adopted. There is one additional variable from the chosen turbulence model and the vector of conservative variables reads

$$\mathbf{W} := (\rho, \rho u_1, \rho u_2, \rho u_3, \rho E, \rho \tilde{\nu})^T. \quad (2.9)$$

Here, $\rho \tilde{\nu}$ is an auxiliary variable. The RANS equations with modified Spalart-Allmaras turbulence model then read

$$\frac{d}{dt} \int_{\Omega} \mathbf{W} d\mathbf{x} + \int_{\partial\Omega} (\mathcal{F}_c \mathbf{n} - \mathcal{F}_v \mathbf{n}) ds = \int_{\Omega} \mathbf{Q} d\mathbf{x}. \quad (2.10)$$

with convective flux tensor $\mathcal{F}_c$, viscous flux tensor $\mathcal{F}_v$, and source terms $\mathbf{Q}$ given through

$$\mathcal{F}_c \mathbf{n} = \begin{pmatrix} \rho V \\ \rho u_1 V + p n_1 \\ \rho u_2 V + p n_2 \\ \rho u_3 V + p n_3 \\ \rho H V \\ \rho \tilde{\nu} V \end{pmatrix}, \quad \mathcal{F}_v \mathbf{n} = \begin{pmatrix} 0 \\ n_1 \tau_{11} + n_2 \tau_{12} + n_3 \tau_{13} \\ n_1 \tau_{21} + n_2 \tau_{22} + n_3 \tau_{23} \\ n_1 \tau_{31} + n_2 \tau_{32} + n_3 \tau_{33} \\ n_1 \theta_1 + n_2 \theta_2 + n_3 \theta_3 \\ n_1 \tau_1^{\text{Tu}} + n_2 \tau_2^{\text{Tu}} + n_3 \tau_3^{\text{Tu}} \end{pmatrix}, \quad (2.11)$$

and

$$\mathbf{Q} = (0, 0, 0, 0, 0, Q^{\text{Tu}})^T. \quad (2.12)$$

Note that all but the turbulent sources Q^{Tu} are zero since any outer forces like e. g. gravitation as well as chemical reactions and absorption or emission of radiation are omitted. The viscous terms are given by

$$\begin{aligned} \tau_{ii} &:= 2\mu_{\text{eff}} \frac{\partial u_i}{\partial x_i} + \lambda_{\text{eff}} \text{div} \mathbf{u}, \quad \lambda_{\text{eff}} = -\frac{2}{3}\mu_{\text{eff}} \quad i = 1, 2, 3, \\ \tau_{ij} &:= \mu_{\text{eff}} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \quad i, j = 1, 2, 3, \quad i \neq j, \\ \tau_i^{\text{Tu}} &:= \begin{cases} \frac{\mu_l + \rho \tilde{\nu}}{\sigma} \frac{\partial \tilde{\nu}}{\partial x_i}, & \tilde{\nu} \geq 0 \\ \frac{\mu_l + f_n \rho \tilde{\nu}}{\sigma} \frac{\partial \tilde{\nu}}{\partial x_i}, & \tilde{\nu} < 0 \end{cases}, \quad f_n := \frac{c_{n_1} + \chi^3}{c_{n_1} - \chi^3}, \quad c_{n_1} = 16, \quad i = 1, 2, 3 \end{aligned} \quad (2.13)$$

and

$$\theta_i := \left(\sum_{j=1}^3 u_j \tau_{ij} \right) + q_i, \quad i = 1, 2, 3, \quad (2.14)$$

with constant $\sigma = \frac{2}{3}$ and the heat flux vector $\mathbf{q} := (q_1, q_2, q_3)^T$ given by Fourier's law

$$\mathbf{q}^T := \kappa_{\text{eff}} \text{grad} T, \quad (2.15)$$

where $T = \frac{p}{\rho R}$ is the temperature and R the universal gas constant. The effective viscosity μ_{eff} and the effective conductivity κ_{eff} are given by

$$\mu_{\text{eff}} := \mu_l + \mu_t, \quad \kappa_{\text{eff}} := \kappa_l + \kappa_t. \quad (2.16)$$

The molecular viscosity μ_l is modeled by Sutherland's law,

$$\mu_l := \mu_{l,\infty} \left(\frac{T}{T_\infty} \right)^{\frac{3}{2}} \frac{T_\infty + \bar{T}}{T + \bar{T}}, \quad (2.17)$$

with constants chosen as $\bar{T} = 110.4\text{K}$, $T_\infty = 273\text{K}$ and $\mu_{l,\infty} = 1.716 \cdot 10^{-5}\text{Pa}\cdot\text{s}$. The thermal conductivity κ_l can be defined by

$$\kappa_l := \frac{c_p \mu_l}{\text{Pr}_l}, \quad c_p := R \frac{\gamma}{\gamma - 1}, \quad (2.18)$$

with a Prandtl number $\text{Pr}_l = 0.72$. The turbulent conductivity κ_t and the eddy viscosity μ_t are given by

$$\kappa_t := \frac{c_p \mu_t}{\text{Pr}_t}, \quad \mu_t := \rho \tilde{\nu} f_{v_1}, \quad f_{v_1} := \frac{\chi^3}{\chi^3 + c_{v_1}^3}, \quad \chi := \frac{\tilde{\nu}}{\nu_l}, \quad \nu_l := \frac{\mu_l}{\rho}, \quad (2.19)$$

with turbulent Prandtl number $\text{Pr}_t := 0.92$ and constant $c_{v_1} := 7.1$.

Finally, the turbulent source term Q^{Tu} is given as the difference of the turbulent

production Pr and destruction De added to the diffusion Di:

$$Q^{\text{Tu}} := \rho(\text{Pr} - \text{De}) + \text{Di}, \quad (2.20)$$

$$\text{Pr} := \begin{cases} c_{b_1}(1 - f_{t_2})\tilde{S}\tilde{\nu} & , \tilde{\nu} \geq 0 \\ c_{b_1}(1 - c_{t_3})S\tilde{\nu} & , \tilde{\nu} < 0 \end{cases}, \quad f_{t_2} := c_{t_3} \exp(-c_{t_4}\chi^2) \quad (2.21)$$

$$\text{De} := \begin{cases} \left(c_{w_1}f_w - \frac{c_{b_1}}{\kappa^2}f_{t_2}\right)\left(\frac{\tilde{\nu}}{d}\right)^2 & , \tilde{\nu} \geq 0 \\ -c_{w_1}\left(\frac{\tilde{\nu}}{d}\right)^2 & , \tilde{\nu} < 0 \end{cases}, \quad f_w := g\left(\frac{1 + c_{w_3}^6}{(g)^6 + c_{w_3}^6}\right)^{1/6} \quad (2.22)$$

$$\text{Di} := \frac{c_{b_2}}{\sigma}\rho \sum_{j=1}^3 \left(\frac{\partial \tilde{\nu}}{\partial x_j}\right)^2 - \frac{\nu_l + \tilde{\nu}}{\sigma} \sum_{j=1}^3 \frac{\partial \rho}{\partial x_j} \frac{\partial \tilde{\nu}}{\partial x_j}, \quad (2.23)$$

where d denotes the wall distance and the other terms are given by

$$S := \|\text{curl}(\mathbf{u})\|, \quad \bar{S} := \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v_2}, \quad \tilde{S} := \begin{cases} S + \bar{S} & , \bar{S} \geq -c_{v_2}S \\ S + \frac{S(c_{v_2}^2 S + c_{v_3}\bar{S})}{(c_{v_3} - 2c_{v_2})S - \bar{S}} & , \bar{S} < -c_{v_2}S \end{cases}, \quad (2.24)$$

$$f_{v_2} := 1 - \frac{\chi}{1 + \chi f_{v_1}}, \quad g := r + c_{w_2}(r^6 - r), \quad r := \min\left(\frac{\tilde{\nu}}{\kappa^2 d^2 \tilde{S}}, 10\right). \quad (2.25)$$

Further, the following constants are used:

$$c_{b_1} := 0.1355, \quad c_{b_2} := 0.622, \quad \kappa := 0.41, \quad (2.26)$$

$$c_{w_1} := \frac{c_{b_1}}{\kappa} + \frac{1 + c_{b_2}}{\sigma}, \quad c_{w_2} := 0.3, \quad c_{w_3} := 2, \quad (2.27)$$

$$c_{t_3} := 1.2, \quad c_{t_4} := 0.5, \quad c_{v_2} := 0.7, \quad c_{v_3} := 0.9. \quad (2.28)$$

Remark 2.4 (Navier-Stokes equations). *To get the Navier-Stokes equations, modify the RANS equations with Spalart-Allmaras turbulence model by deleting the last equation in (2.10) and setting $\mu_t = 0$ and $\kappa_t = 0$. Note however that the meaning of the variables differs from the RANS equations, where the variables can be interpreted as the mean (or expected value) of stochastic variables (ensemble averaging) (cf. Remark 2.3).*

2.3 Spatial discretization and TAU code

For unsteady simulations TAU uses the method of lines, first discretizing in space leaving a system of ordinary differential equations to be solved afterwards. The spatial discretization is based on an unstructured finite volume node-centered (cell-vertex) scheme

(cf. [6, Section 4.2.3]) for which a so called *dualgrid* is created in a preprocessing step.

In the following a short overview over the discrete approximations in space used in this thesis is presented. Denote the set of dualgrid cells by $G := \{1, \dots, N\}$, the domain of a cell $i \in G$ by Ω_i , and its boundary by $\partial\Omega_i$. The unknowns in conservative form are given on the vertices of the primary grid, each corresponding to a cell in the dualgrid, by $\mathbf{W}_1, \dots, \mathbf{W}_N$. The set of neighbors of a cell $i \in G$ is denoted by $N(i)$ and the face between points $i \in G$ and $j \in N(i)$ by ij . The spatially discretized RANS equations are the result of the approximations of the integrals as described in the following subsections. In this section, only the discretization for cells in the interior of the computational domain is described. For cells at the boundary refer to Section 2.6.

Inviscid flux

For the inner volume Ω_i of a cell $i \in G$, the convective flux is approximated by a central scheme with matrix-valued artificial dissipation as

$$\int_{\partial\Omega_i} \mathcal{F}_c \mathbf{n} ds \approx \left(\int_{\partial\Omega_i} \mathcal{F}_c \mathbf{n} ds \right)_{\text{disc}} \quad (2.29)$$

$$\left(\int_{\partial\Omega_i} \mathcal{F}_c \mathbf{n} ds \right)_{\text{disc}} = \sum_{j \in N(i)} \frac{1}{2} ((\mathcal{F}_c \tilde{\mathbf{n}}_{ij})(\mathbf{W}_i) + (\mathcal{F}_c \tilde{\mathbf{n}}_{ij})(\mathbf{W}_j)) - \mathbf{D}_{ij}(\mathbf{W}) \quad (2.30)$$

with $\mathbf{n}_{ij}$ the unit normal vector for face ij pointing to face j and

$$\tilde{\mathbf{n}}_{ij} := \text{svol}(\Omega_{ij}) \mathbf{n}_{ij}, \quad \text{svol}(\Omega_{ij}) := \int_{\partial\Omega_i \cap \partial\Omega_j} 1 \, ds. \quad (2.31)$$

Furthermore, the artificial dissipation is given by

$$\mathbf{D}_{ij}(\mathbf{W}) := |\mathcal{A}_{ij}| \left[\frac{1}{2} \Psi_{ij}(\mathbf{W}_j - \mathbf{W}_i) - \xi \eta(N(i), N(j)) s_{ij}(\mathbf{W}) (1 - \Psi_{ij})(\mathbf{L}_j(\mathbf{W}) - \mathbf{L}_i(\mathbf{W})) \right]. \quad (2.32)$$

Here, the undivided Laplacian operator

$$\mathbf{L}_i(\mathbf{W}) := \sum_{j \in N(i)} (\mathbf{W}_j - \mathbf{W}_i), \quad (2.33)$$

the pressure switch

$$\Psi_{ij} := \min(\epsilon_\Psi \max(\Psi_i, \Psi_j), 1), \quad \Psi_i := \left| \frac{\sum_{j \in N(i)} (p_j - p_i)}{\sum_{j \in N(i)} (p_j + p_i)} \right|, \quad (2.34)$$

the user defined constant $\epsilon_\Psi \in [1, 8]$, and the derivative of the convective flux

$$\mathcal{A}_{ij} := \left. \frac{\partial(\mathcal{F}_c \tilde{\mathbf{n}}_{ij})}{\partial \mathbf{W}} \right|_{\mathbf{W}_{\text{Roe}}} \quad (2.35)$$

with Roe-averaged variables are used. Taking the absolute value of the matrix $|\mathcal{A}_{ij}|$ is done with respect to an operator functional calculus, i. e. if $\mathcal{A}_{ij} = \sum_{l=1}^6 \lambda_l \mathbf{g}_l \mathbf{q}_l$ with eigenvalues λ_l , right eigenvectors $\mathbf{g}_l$ and left eigenvectors $\mathbf{q}_l$, then $|\mathcal{A}_{ij}| = \sum_{l=1}^6 |\lambda_l| \mathbf{g}_l \mathbf{q}_l$ (cf. [51, (14)]). The construction and eigenvalue decomposition of $\mathcal{A}_{ij}$ is described in detail in [50, 51]. Furthermore, the cell stretching coefficient $s_{ij}(\mathbf{W})$ is computed by

$$s_{ij}(\mathbf{W}) := 1 + 2 \frac{\sqrt{z_i} \sqrt{z_j}}{\sqrt{z_i} + \sqrt{z_j} + \epsilon}, \quad z_i := \frac{\max(\frac{1}{2}\Lambda_i - \Lambda_{ij}, 0)}{\Lambda_{ij}}, \quad \epsilon = 1 \cdot 10^{-16}, \quad (2.36)$$

where

$$\Lambda_i := \sum_{j \in N(i)} \Lambda_{ij}, \quad \Lambda_{ij} := |V_{ij}| + \text{svol}(\Omega_{ij}) a_{ij}, \quad (2.37)$$

with speed of sound and normal velocity

$$a_i := \sqrt{\gamma \frac{p_i}{\rho_i}} \quad a_{ij} := \frac{1}{2}(a_i + a_j), \quad V_{ij} := \frac{1}{2}(\mathbf{u}_i + \mathbf{u}_j) \tilde{\mathbf{n}}_{ij}. \quad (2.38)$$

The coefficient η given by

$$\eta(N(i), N(j)) := \frac{9(\zeta(N(i)) + \zeta(N(j)))}{\zeta(N(i))\zeta(N(j))}, \quad \zeta(N(i)) := (1 + \#N(i))\#N(i) \quad (2.39)$$

is a neighbor scaling introduced to deal with unstructured meshes and the factor ξ is a global constant weighting factor. A typical choice is $\xi \in [\frac{1}{128}, \frac{1}{32}]$. For an analysis of the coefficients see [52].

Viscous flux

The viscous flux is approximated by computing the flux of the average variable on a cell boundary

$$\int_{\partial\Omega_i} \mathcal{F}_v \mathbf{n} ds \approx \left(\int_{\partial\Omega_i} \mathcal{F}_v \mathbf{n} ds \right)_{\text{disc}} \quad (2.40)$$

$$\left(\int_{\partial\Omega_i} \mathcal{F}_v \mathbf{n} ds \right)_{\text{disc}} = \sum_{j \in \mathcal{N}(i)} (\mathcal{F}_v \tilde{\mathbf{n}}_{ij}) (\mathbf{W}_{ij}, (\text{grad } \mathbf{W}_{ij})^{\text{corr}}), \quad (2.41)$$

with

$$\mathbf{W}_{ij} = \frac{1}{2}(\mathbf{W}_i + \mathbf{W}_j) \quad (2.42)$$

and the derivatives on the face calculated by a Green-Gauss method in the following way.

$$\frac{\partial \mathbf{W}_i}{\partial x_k} \approx \left(\frac{\partial \mathbf{W}_i}{\partial x_k} \right)^{\text{GG}} := \frac{1}{\text{vol}(\Omega_i)} \sum_{j \in \mathcal{N}(i)} \frac{(\tilde{n}_{ij})_k}{2} (\mathbf{W}_i + \mathbf{W}_j), \quad k = 1, 2, 3 \quad (2.43)$$

$$(\text{grad } \mathbf{W}_i)^{\text{GG}} := \left(\left(\frac{\partial \mathbf{W}_i}{\partial x_1} \right)^{\text{GG}}, \left(\frac{\partial \mathbf{W}_i}{\partial x_2} \right)^{\text{GG}}, \left(\frac{\partial \mathbf{W}_i}{\partial x_3} \right)^{\text{GG}} \right) \quad (2.44)$$

where $(\tilde{n}_{ij})_k$ is the k th component of $\tilde{\mathbf{n}}_{ij}$ and $\text{vol}(\Omega_i)$ the volume of the cell. The value of the gradient on the face is then approximated by averaging and correction,

$$(\text{grad } \mathbf{W}_{ij})^{\text{GG}} := \frac{1}{2} ((\text{grad } \mathbf{W}_i)^{\text{GG}} + (\text{grad } \mathbf{W}_j)^{\text{GG}}), \quad (2.45)$$

$$(\text{grad } \mathbf{W}_{ij})^{\text{corr}} := (\text{grad } \mathbf{W}_{ij})^{\text{GG}} - ((\text{grad } \mathbf{W}_{ij})^{\text{GG}} (\mathbf{x}_j - \mathbf{x}_i) - (\mathbf{W}_j - \mathbf{W}_i)) \frac{(\mathbf{x}_j - \mathbf{x}_i)^T}{\|\mathbf{x}_j - \mathbf{x}_i\|_2^2}. \quad (2.46)$$

Source terms

For the source terms the variables are taken as constant on a cell, i. e.

$$\int_{\Omega_i} \mathbf{Q} d\mathbf{x} \approx \left(\int_{\Omega_i} \mathbf{Q} d\mathbf{x} \right)_{\text{disc}} \quad (2.47)$$

$$\left(\int_{\Omega_i} \mathbf{Q} d\mathbf{x} \right)_{\text{disc}} = \text{vol}(\Omega_i) \mathbf{Q}(\mathbf{W}_i, (\text{grad } \mathbf{W}_i)^{\text{GG}}). \quad (2.48)$$

Similarly, it is

$$\int_{\Omega_i} \mathbf{W} d\mathbf{x} \approx \text{vol}(\Omega_i) \mathbf{W}_i. \quad (2.49)$$

2.4 Moving grid and basis transformation

Grid velocities

In the previous sections the Euler and RANS equations are described in a cartesian inertial coordinate system for a fixed Ω , i.e. the control volumes (or grid cells) are stationary with respect to the chosen inertial coordinate system. However, in unsteady simulations, the grid might move (rigid body motion), or deform (mesh deformation). As a result of the Reynolds transport theorem, the grid velocities $\mathbf{u}_g$ have to be substracted from the normal velocity in the convective part, yielding an additional convective flux term $-V_g \mathbf{W}$, $V_g = \mathbf{u}_g^T \mathbf{n}$. This can be incorporated by taking $V = (\mathbf{u} - \mathbf{u}_g)^T \mathbf{n}$ in (2.6) and (2.11) and adding $V_g p$ in the energy flux. Hence, the convective flux in (2.11) reads

$$\mathcal{F}_c \mathbf{n} = \begin{pmatrix} \rho V \\ \rho u_1 V + p n_1 \\ \rho u_2 V + p n_2 \\ \rho u_3 V + p n_3 \\ \rho H V + V_g p \\ \rho \tilde{v} V \end{pmatrix}, \quad V = (\mathbf{u} - \mathbf{u}_g)^T \mathbf{n} \quad (2.50)$$

Basis transformation

In the simulations of this thesis, the grid is attached to the body (wing, airplane,...). Hence, the grid rotates when the body rotates as for example in a simulation of a pitching oscillation. In that case, we want to have the vector valued variables represented in the orthonormal basis $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ rotating with the body. Denoting the representation of a vector $\mathbf{v}$ in the rotating body fixed basis by $\mathbf{v}'$, gives

$$\mathbf{v} = \sum_{i=1}^3 v'_i \mathbf{b}_i = \mathcal{B} \mathbf{v}' \quad \text{with} \quad \mathcal{B} = (\mathbf{b}_1 \mathbf{b}_2 \mathbf{b}_3) \in \mathbb{R}^{3 \times 3}. \quad (2.51)$$

Let $\boldsymbol{\omega}$ denote the angular velocity of the rotating basis, i.e.

$$\frac{d}{dt} \mathbf{b}_i = \boldsymbol{\omega} \times \mathbf{b}_i. \quad (2.52)$$

This yields

$$\frac{d}{dt} \int_{\Omega} \rho \mathbf{u} d\mathbf{x} = \frac{d}{dt} \int_{\Omega} \rho \mathcal{B} \mathbf{u}' d\mathbf{x} \quad (2.53)$$

$$= \frac{d}{dt} \left(\mathcal{B} \int_{\Omega} \rho \mathbf{u}' d\mathbf{x} \right) \quad (2.54)$$

$$= \mathcal{B} \frac{d}{dt} \int_{\Omega} \rho \mathbf{u}' d\mathbf{x} + \frac{d}{dt} \mathcal{B} \int_{\Omega} \rho \mathbf{u}' d\mathbf{x} \quad (2.55)$$

$$= \mathcal{B} \frac{d}{dt} \int_{\Omega} \rho \mathbf{u}' d\mathbf{x} + \int_{\Omega} \rho \boldsymbol{\omega} \times (\mathcal{B} \mathbf{u}') d\mathbf{x} \quad (2.56)$$

$$= \mathcal{B} \frac{d}{dt} \int_{\Omega} \rho \mathbf{u}' d\mathbf{x} + \int_{\Omega} \rho \boldsymbol{\omega} \times \mathbf{u} d\mathbf{x} \quad (2.57)$$

$$= \mathcal{B} \frac{d}{dt} \int_{\Omega} \rho \mathbf{u}' d\mathbf{x} + \mathcal{B} \int_{\Omega} \rho \boldsymbol{\omega}' \times \mathbf{u}' d\mathbf{x}, \quad (2.58)$$

and hence, the momentum equation

$$\frac{d}{dt} \int_{\Omega} \rho \mathbf{u} d\mathbf{x} + \int_{\delta\Omega} (\rho \mathbf{u} (\mathbf{u} - \mathbf{u}_g)^T \mathbf{n} + p \mathbf{n}) d\mathbf{s} + \int_{\delta\Omega} \mathcal{T} \mathbf{n} d\mathbf{s} = 0, \quad (2.59)$$

with viscous stress tensor $\mathcal{T} = (\tau_{ij})_{i,j=1,2,3}$ represented in the rotating basis reads

$$\frac{d}{dt} \int_{\Omega} \rho \mathbf{u}' d\mathbf{x} + \int_{\delta\Omega} (\rho \mathbf{u}' (\mathbf{u}' - \mathbf{u}'_g)^T \mathbf{n}' + p \mathbf{n}') d\mathbf{s} + \int_{\delta\Omega} \mathcal{T} \mathbf{n}' d\mathbf{s} + \int_{\Omega} \rho \boldsymbol{\omega}' \times \mathbf{u}' d\mathbf{x} = 0. \quad (2.60)$$

In the remainder of the thesis, the prime for values in the rotating basis is omitted and the equations are given with respect to the rotating basis if not stated otherwise. Hence, the source terms (2.12) read

$$\mathbf{Q} = (0, (\rho \boldsymbol{\omega} \times \mathbf{u})_1, (\rho \boldsymbol{\omega} \times \mathbf{u})_2, (\rho \boldsymbol{\omega} \times \mathbf{u})_3, 0, Q^{\text{Tu}})^T. \quad (2.61)$$

in the rotating basis.

2.5 Disturbance velocity approach for gust encounter simulations

The main focus of the numerical results in this thesis is on gust encounter simulations. For these simulations, there are two different approaches available in TAU [40]. The gust can be modeled either by an unsteady boundary condition feeding the gust into the flow field, or by a *disturbance velocity approach*, modeling the gust by adding the negative

gust speed to the local grid velocities.

$$\mathbf{u}_g(\mathbf{x}, t) = -\mathbf{u}_{gust}(\mathbf{x}, t) \quad (2.62)$$

The first approach is more accurate, since it captures the mutual interaction of gust and aircraft. The second approach only captures the influence of the gust on the aircraft. The influence of the aircraft on the gust is not modeled. On the other hand, the disturbance velocity approach is much more efficient, since standard meshes can be used, whereas the more accurate approach needs a high resolution especially for the gust in order to avoid too much numerical losses when the gust is transported through the flow field. A detailed comparison of the two approaches can be found in [40]. In this thesis, only the disturbance velocity approach is used to model gusts. While this approach can be used to model any kind of gust, the simulations in this thesis are restricted to $1 - \cos$ -shaped upwards gusts. Such a gust is specified by an Amplitude A , a length l , a starting point x_s , and a gust translation speed u_t , typically equal to the negative speed of the aircraft. The gust is then modeled by adding local grid velocities

$$\mathbf{u}_g(\mathbf{x}, t) = \begin{cases} \mathcal{B}(0, 0, -A/2 (1 - \cos(\frac{2\pi}{l}(x_1 - (x_s + tu_t))))))^T & \text{if } x_1 - (x_s + tu_t) \in [0, l], \\ (0, 0, 0)^T & \text{otherwise} \end{cases} \quad (2.63)$$

to the normal velocities as described in Section 2.4. Here, the matrix $\mathcal{B}$ represents the coordinate transformation from the considered inertial coordinate system to the body fixed system in which the variables are represented.

Analysis of disturbance velocity approach

As mentioned above, the disturbance velocity approach is only an approximation for the influence of a gust on the aircraft. To get a better view on the method, a short analysis of the method is done. The basic idea of the disturbance velocity approach is, to add the local gust velocity to the contravariant velocity V in order to capture the convective influence of the gust. This is numerically equivalent to adding the negative gust velocity to the local grid velocity $\mathbf{u}_g$ (see Section 2.4). However, note that there is no real local grid movement or deformation. The grid stays the same.

This approach has a view limitations. As mentioned above, one limitation is, that exterior influences on the gust (e.g. by an airplane) are not modeled. The local speed of the gust is predefined as described above. Furthermore, the additional velocities due to the gust are not explicitly integrated in the flow field, i.e. the momentum variables

$\rho \mathbf{u}$ have wrong values. Even though the influence on the convective part of the flux is addressed (by the change of the contravariant velocity), the influence of the gust on the viscous fluxes is neglected (changes in $\frac{\partial u_i}{\partial x_j}$). Additionally, this approach is also limited in the range of applications, as for example the expansion of a gust entering the flow field can not be modeled.

Despite the limitations, the approach leads to results with good accuracy regarding the lift coefficient for applications as covered in this thesis (see [40] for a numerical evaluation).

Geometric conservation law

The geometric conservation law results from the requirement, that the numerical scheme preserves the state of a uniform flow also for deforming grids. In integral form it reads

$$\frac{\partial}{\partial t} \int_{\Omega} d\mathbf{x} = \int_{\partial\Omega} V_g ds, \quad V_g = \mathbf{u}_g \mathbf{n}. \quad (2.64)$$

Since the divergence of the local grid velocities in (2.63) is 0, application of the integral theorem of Gauss results in

$$\int_{\partial\Omega} V_g ds = 0, \quad V_g = \mathbf{u}_g \mathbf{n} \quad (2.65)$$

for the local grid velocities resulting from a disturbance velocity approach to model the gust. Hence, the geometric conservation law (2.64) is not changed. In particular, for non-deformed meshes as considered in the applications in this thesis, it is

$$0 = \frac{\partial}{\partial t} \int_{\Omega} d\mathbf{x} = \int_{\partial\Omega} V_g ds. \quad (2.66)$$

The discretization should also respect this law. However, when computing the gust velocity on a face of the dualgrid as mean of the gust velocities in the neighboring vertices of the primary grid, the geometric conservation law reads

$$\left(\frac{\partial}{\partial t} \text{vol}(\Omega_i) \right)_{\text{disc}} = \left(\int_{\partial\Omega} V_g ds \right)_{\text{disc}} = \sum_{j \in \mathcal{N}(i)} \frac{1}{2} (\mathbf{u}_{g_i} + \mathbf{u}_{g_j}) \tilde{\mathbf{n}}_{ij}, \quad (2.67)$$

which is not necessarily zero for $\mathbf{u}_g$ as in (2.63). This means, that adding a gust with the disturbance velocity approach as in (2.63) with gust translation speed $u_t = 0$ to a uniform flow, does not lead to a flow which is constant in time (as it should). To remedy this flaw in the discretization, an additional term $\left(\frac{\partial}{\partial t} \text{vol}(\Omega_i) \right)_{\text{disc}} \mathbf{W}_i$ with $\left(\frac{\partial}{\partial t} \text{vol}(\Omega_i) \right)_{\text{disc}}$

as in (2.67) is added to the residual in cell i (cf. Section 2.7).

2.6 Boundary conditions

In the numerical test cases presented in this thesis different boundary conditions are used. In detail, an inviscid wall, a viscous wall, a farfield, a symmetry, and a periodic boundary condition are used. In the following, these boundary treatments are shortly described. In general there are two different kinds of boundary conditions in TAU: *weak* and *strong* boundary conditions. Boundary conditions based on a flux formulation, also called *weak boundary conditions* add an additional flux to the convective and viscous fluxes. Denoting the boundary of the computational domain by ∂D , (2.30) then reads

$$\left(\int_{\partial\Omega_i} \mathcal{F}_c \mathbf{n} ds \right)_{\text{disc}} = \sum_{j \in N(i)} \frac{1}{2} ((\mathcal{F}_c \tilde{\mathbf{n}}_{ij})(\mathbf{W}_i) + (\mathcal{F}_c \tilde{\mathbf{n}}_{ij})(\mathbf{W}_j)) - \mathbf{D}_{ij}(\mathbf{W}) + \sum_{b \in N_{\text{bdry}}(i)} \left(\int_{\partial\Omega_i \cap \partial D_b} \mathcal{F}_c \mathbf{n} ds \right)_{\text{disc}}, \quad (2.68)$$

and (2.41) is transformed to

$$\left(\int_{\partial\Omega_i} \mathcal{F}_v \mathbf{n} ds \right)_{\text{disc}} = \sum_{j \in N(i)} (\mathcal{F}_v \tilde{\mathbf{n}}_{ij})(\mathbf{W}_{ij}, (\text{grad } \mathbf{W}_{ij})^{\text{corr}}) + \sum_{b \in N_{\text{bdry}}(i)} \left(\int_{\partial\Omega_i \cap \partial D_b} \mathcal{F}_v \mathbf{n} ds \right)_{\text{disc}}. \quad (2.69)$$

So called *strong boundary conditions* on the other hand directly change specific flow variables and do not add an additional flux to (2.68) and (2.69). They can be described by operators

$$\mathbf{P}_{\text{bdry_var}} = \mathbf{P}_{\text{bdry_var}, \partial D_1} \circ \dots \circ \mathbf{P}_{\text{bdry_var}, \partial D_{N_{\text{bdry}}}}, \quad (2.70)$$

and

$$\mathbf{P}_{\text{bdry_res}} = \mathbf{P}_{\text{bdry_res}, \partial D_1} \circ \dots \circ \mathbf{P}_{\text{bdry_res}, \partial D_{N_{\text{bdry}}}} \quad (2.71)$$

which are applied to the conservative variables before the residual evaluation and the residual, respectively (cf. (2.93)). Here $\circ$ denotes the composition of the operators.

Inviscid wall

The inviscid wall boundary condition is used for the boundaries between flow field and structure for inviscid (Euler) simulations. Since the wall is assumed to be impermeable, and there is no friction, the boundary treatment is described by the condition

$$(\mathbf{u} - \mathbf{u}_g)\mathbf{n} = 0, \quad (2.72)$$

i. e. the normal velocity V on the wall is zero. Since a node-centered scheme is used, the flow variables are located on the vertices of the primary grid. This means that in case of a boundary cell the values are located directly on the boundary. The convective flux for the boundary face bi corresponding to the inviscid wall boundary part b and cell i can then be computed by

$$\left(\int_{\partial\Omega_i \cap \partial D_b} \mathcal{F}_c \mathbf{n} ds \right)_{\text{disc}} = \begin{pmatrix} 0 \\ p_i \tilde{n}_{bi,1} \\ p_i \tilde{n}_{bi,2} \\ p_i \tilde{n}_{bi,3} \\ p_i \mathbf{u}_g \tilde{\mathbf{n}}_{bi} \end{pmatrix}, \quad (2.73)$$

with face normal

$$(\tilde{n}_{bi,1}, \tilde{n}_{bi,2}, \tilde{n}_{bi,3})^T = \tilde{\mathbf{n}}_{bi} = \text{svol}(\Omega_{bi}) \mathbf{n}_{bi} \quad (2.74)$$

and pressure p_i .

Viscous wall

For the viscous wall boundary condition, it is assumed that the wall is adiabatic, i. e. there is no heat transfer across the wall, and the velocity of the fluid directly at the surface equals that of the surface, i. e. equal to the grid velocity $\mathbf{u}_g$. This yields the following conditions for the flow directly at the surface:

$$\mathbf{u} = \mathbf{u}_g, \quad \frac{\partial T}{\partial \mathbf{n}} = 0, \quad \tilde{\nu} = 0. \quad (2.75)$$

The current practice in TAU is to use a strong formulation for the viscous wall boundary condition, meaning that boundary values are set to a certain value. For the viscous wall boundary condition, this is done by setting $\mathbf{W}_i = \left(\rho_i, \rho_i u_{g,i,1}, \rho_i u_{g,i,2}, \rho_i u_{g,i,3}, (\rho E)_i, 0 \right)^T$ for all cells i adjacent to a face belonging to a viscous boundary part before computing the residual. After the computation of the residual, it (the residual) is additionally set

to zero in the momentum and turbulent equations. This yields the operators

$$\mathbf{P}_{\text{bdry_var}, \partial D_b} : \mathbb{R}^{Nd} \rightarrow \mathbb{R}^{Nd} \quad (2.76)$$

$$(\mathbf{W}_i)_{i \in 1, \dots, N} \rightarrow \begin{cases} (\mathbf{W}_i), & \mathbf{x}_i \notin \partial D_b \\ \left(\rho_i, \rho_i u_{g_i,1}, \rho_i u_{g_i,2}, \rho_i u_{g_i,3}, (\rho E)_i, 0 \right)^T, & \mathbf{x}_i \in \partial D_b \end{cases} \quad (2.77)$$

and

$$\mathbf{P}_{\text{bdry_res}, \partial D_b} : \mathbb{R}^{Nd} \rightarrow \mathbb{R}^{Nd} \quad (2.78)$$

$$(\mathbf{R}_i)_{i \in 1, \dots, N} \rightarrow \begin{cases} (\mathbf{R}_i), & \mathbf{x}_i \notin \partial D_b \\ ((\mathbf{R}_i)_1, 0, 0, 0, (\mathbf{R}_i)_5, 0)^T, & \mathbf{x}_i \in \partial D_b \end{cases} \quad (2.79)$$

with $d = 6$ for the RANS equations. The respective operators with $d = 5$ for the Euler and Navier-Stokes equations are the same operators restricted to the first 5 components.

Farfield boundary

For the farfield boundary condition, it is necessary to define a farfield state

$$\mathbf{W}_\infty = (\rho_\infty, (\rho \mathbf{u}_1)_\infty, (\rho \mathbf{u}_2)_\infty, (\rho \mathbf{u}_3)_\infty, (\rho E)_\infty, (\rho \tilde{\nu})_\infty)^T, \quad (2.80)$$

typically identical to the free stream conditions. The boundary contribution in (2.68) is then computed by

$$\left(\int_{\partial \Omega_i \cap \partial D_b} \mathcal{F}_c \mathbf{n} ds \right)_{\text{disc}} = (\mathcal{F}_c \tilde{\mathbf{n}}_{bi})(\mathbf{W}_i) + \sum_{l=1}^6 \min(0, \lambda_l) \mathbf{g}_l \mathbf{q}_l (\mathbf{W}_\infty - \mathbf{W}_i), \quad (2.81)$$

for components $1, \dots, 5$ and

$$\left(\int_{\partial \Omega_i \cap \partial D_b} \mathcal{F}_c \mathbf{n} ds \right)_{\text{disc}, 6} = \max(0, \lambda_6) (\rho \tilde{\nu})_i + \min(0, \lambda_6) (\rho \tilde{\nu})_\infty. \quad (2.82)$$

Here, λ_l , $\mathbf{g}_l$, and $\mathbf{q}_l$ $l = 1, \dots, 6$ are the eigenvalues, left eigenvectors, and right eigenvectors, respectively, of the convective flux Jacobian evaluated at the Roe-average of $\mathbf{W}_\infty$ and $\mathbf{W}_i$. The viscous boundary flux contribution in (2.69) is set to zero

$$\left(\int_{\partial \Omega_i \cap \partial D_b} \mathcal{F}_v \mathbf{n} ds \right)_{\text{disc}} = 0. \quad (2.83)$$

Symmetry boundary

The symmetry boundary condition is implemented in a strong formulation in TAU. It is implemented as a projection of the momentum on the plane tangential to the boundary part. For every cell $i \in G$ adjacent to the symmetry boundary part b the velocity is set to

$$\mathbf{u}_i^{\text{sym}} = \mathbf{u}_i - \mathbf{n}_{bi}(\mathbf{n}_{bi}\mathbf{u}_i) \quad (2.84)$$

before the computation of the residual. Afterwards, the residual is corrected by an equivalent projection of the residual of the momentum equations. Hence, the symmetry boundary condition is given by operators

$$\mathbf{P}_{\text{bdry_var}, \partial D_b} : \mathbb{R}^{Nd} \rightarrow \mathbb{R}^{Nd} \quad (2.85)$$

$$(\mathbf{W}_i)_{i \in 1, \dots, N} \rightarrow \begin{cases} (\mathbf{W}_i), & \mathbf{x}_i \notin \partial D_b \\ \left(\rho_i, \rho_i u_{i,1}^{\text{sym}}, \rho_i u_{i,2}^{\text{sym}}, \rho_i u_{i,3}^{\text{sym}}, (\rho E)_i, (\rho \tilde{v})_i \right)^T, & \mathbf{x}_i \in \partial D_b \end{cases} \quad (2.86)$$

and

$$\mathbf{P}_{\text{bdry_res}, \partial D_b} : \mathbb{R}^{Nd} \rightarrow \mathbb{R}^{Nd} \quad (2.87)$$

$$(\mathbf{R}_i)_{i \in 1, \dots, N} \rightarrow \begin{cases} (\mathbf{R}_i), & \mathbf{x}_i \notin \partial D_b \\ (\mathbf{R}_i)^{\text{sym}}, & \mathbf{x}_i \in \partial D_b \end{cases}, \quad (2.88)$$

with

$$R_{i,j}^{\text{sym}} = \begin{cases} R_{i,j}, & j \in \{1, 5, 6\} \\ R_{i,j} - n_{bi,j-1} \left(\sum_{j=2}^4 n_{bi,j-1} R_{i,j} \right), & j \in \{2, 3, 4\} \end{cases}. \quad (2.89)$$

Again, the operators with $d = 6$ are given for the RANS equations and the respective operators with $d = 5$ for the Euler and Navier-Stokes equations are the same operators restricted to the first 5 components.

Periodic boundary

For the periodic boundary condition it is necessary to have at least two periodic boundary parts of the same size linked together, such that each cell at one boundary part has a corresponding cell at the other part. Then so called *dummy cells* are introduced at the boundary with flow values equal to those of the corresponding cells of the other boundary part. Using those dummy cells, the flux across the boundary is then computed as usual.

2.7 Summary of discrete equation

Denote the unknowns for the whole grid by the vector

$$\mathbf{W} := (\mathbf{W}_i)_{i=1,\dots,N} \in \mathbb{R}^{Nd}, \quad (2.90)$$

where d is the number of equations in the system of integral equations. The distinction to the vector of conservative variables (2.9) in the non-discretized form is clear from the context. Furthermore, define the volume matrix

$$\mathcal{V} := \text{diag}(\text{vol}(\Omega_i)) \otimes \mathcal{I}_d \in \mathbb{R}^{Nd \times Nd}, \quad (2.91)$$

which is constant in time for the applications considered in this thesis. The residual is given by the Nd dimensional vector $\mathbf{R}(t, \mathbf{W})$ consisting of the discrete form of $\int_{\partial\Omega} (\mathcal{F}_c \mathbf{n} - \mathcal{F}_v \mathbf{n}) ds - \int_{\Omega} \mathbf{Q} d\mathbf{x}$. Note that the residual can be explicitly time dependent in unsteady simulations, e.g. due to time dependent boundary conditions or grid velocities. Then, the RANS equations (2.10) discretized in space can be written as

$$\mathcal{V} \frac{d}{dt} \mathbf{W}(t) = -\mathbf{R}(t, \mathbf{W}(t)), \quad (2.92)$$

with

$$\mathbf{R}(t, \mathbf{W}(t)) = \mathbf{P}_{\text{bdry_res}} \mathbf{R}^{\text{flux}}(t, \mathbf{P}_{\text{bdry_var}} \mathbf{W}(t)) \quad (2.93)$$

and

$$\mathbf{R}_i^{\text{flux}} = \left(\int_{\partial\Omega_i} \mathcal{F}_c \mathbf{n} ds \right)_{\text{disc}} - \left(\int_{\partial\Omega_i} \mathcal{F}_v \mathbf{n} ds \right)_{\text{disc}} - \left(\int_{\Omega_i} \mathbf{Q} d\mathbf{x} \right)_{\text{disc}} + \left(\frac{\partial}{\partial t} \text{vol}(\Omega_i) \right)_{\text{disc}} \mathbf{W}_i. \quad (2.94)$$

Here, the summands of (2.94) are as in (2.68), (2.69), (2.48), and (2.67) and the operators for the strong boundary conditions in (2.93) are as in (2.70) and (2.71). Note that in case of the RANS equations the convective flux including moving grids is given in (2.50), the viscous flux is given in (2.11), and the source terms are given in (2.61).

In this thesis, the explicit time dependence of the residual is sometimes not stated and e.g. $\mathbf{R}(\mathbf{W}^j)$ is written instead of $\mathbf{R}(t_j, \mathbf{W}^j)$. The corresponding time for the evaluation is clear from the context.

CHAPTER 3

Numerical methods for ordinary differential equations

As described in the previous chapter, discretizing the governing equations in space results in a system of ordinary differential equations (2.92). Together with the starting solution $\mathbf{W}^0$ this yields the initial value problem (IVP)

$$\frac{d}{dt}\mathbf{W}(t) = -\mathcal{V}^{-1}\mathbf{R}(t, \mathbf{W}(t)), \quad t \in I = [t_s, t_e] \subset \mathbb{R}, \quad t_s < t_e \quad (3.1)$$

$$\mathbf{W}(t_s) = \mathbf{W}^0 \quad (3.2)$$

In this thesis, it is assumed that the IVP has a unique solution $\mathbf{W} : I \rightarrow \mathbb{R}^{N_d}$, though existence and uniqueness of a solution has not been proven in general for the RANS equations. For an introduction to existence and uniqueness theory as well as a general introduction to ordinary differential equations there are several textbooks available [16, 25, 36]. To get an approximate solution of the IVP (3.1), a discretization

$$D = \{t_0, t_1, \dots, t_n\} \subset [t_s, t_e], \quad (3.3)$$

satisfying

$$t_i < t_j, \quad i < j, \quad t_0 = t_s, \quad t_n = t_e \quad (3.4)$$

is chosen. The corresponding time steps are denoted by

$$\Delta t_j = t_j - t_{j-1}, \quad j = 1, \dots, n \quad (3.5)$$

and an approximate solution by $\mathbf{W}^D$. The short notation

$$\mathbf{W}^j := \mathbf{W}^D(t_j) \quad (3.6)$$

is used during the thesis. Furthermore, denote the exact evolution of (3.1) by Φ . It is defined by

$$\Phi^{t,\tilde{t}}\mathbf{W}(\tilde{t}) := \mathbf{W}(t). \quad (3.7)$$

The discrete numerical evolution for some specified numerical method to solve (3.1) is identified by Ψ with

$$\Psi^{t_j, t_i} \mathbf{W}^i := \mathbf{W}^j. \quad (3.8)$$

3.1 Introduction to Runge-Kutta methods

A Runge-Kutta method is given by two vectors $\mathbf{b}$ and $\mathbf{c}$ of length s as well as an $s \times s$ matrix $\mathcal{A}$. Here, s is the number of stages of the method. For the IVP (3.1) it is then defined by

$$\mathbf{z}_m = \mathbf{W}^{j-1} + \Delta t_j \sum_{n=1}^s a_{mn} \mathbf{k}_n \quad m = 1, \dots, s \quad (3.9)$$

$$\mathbf{k}_m = -\mathcal{V}^{-1} \mathbf{R}(t_{j-1} + c_m \Delta t_j, \mathbf{z}_m) \quad m = 1, \dots, s \quad (3.10)$$

$$\mathbf{W}^j = \mathbf{W}^{j-1} + \Delta t_j \sum_{n=1}^s b_n \mathbf{k}_n \quad (3.11)$$

Different notations found in the literature can be derived by inserting either $\mathbf{z}_m$ or $\mathbf{k}_m$ in the remaining two equations. Usually the coefficients of such a Runge-Kutta method are represented in a *Butcher tableau*

$$\begin{array}{c|c} \mathbf{c} & \mathcal{A} \\ \hline & \mathbf{b}^T \end{array}.$$

In this thesis, only Runge-Kutta methods which are invariant with respect to autonomization, i. e. which fulfill

$$\sum_{n=1}^s a_{mn} = c_m, \quad m = 1, \dots, s \quad (3.12)$$

are considered.

Classical order conditions

The classical order of accuracy of a Runge-Kutta method depends on the coefficients $\mathbf{b}$, $\mathbf{c}$ and $\mathcal{A}$. In the following, the classical order conditions for up to order $p = 4$ are stated. They can be derived by comparing the Taylor-expansions of the exact evolution and the

discrete evolution of a general IVP. For a detailed derivation of the conditions as well as a more general introduction to Runge-Kutta methods see [16, 36] or similar textbooks. A Runge-Kutta method with $\sum_{n=1}^s a_{mn} = c_m$ has order

- $p = 1$, if

$$\sum_m b_m = 1, \quad (3.13)$$

- $p = 2$, if additionally

$$\sum_m b_m c_m = \frac{1}{2}, \quad (3.14)$$

- $p = 3$, if additionally

$$\sum_m b_m c_m^2 = \frac{1}{3} \quad (3.15)$$

$$\sum_{m,n} b_m a_{mn} c_n = \frac{1}{6} \quad (3.16)$$

- $p = 4$, if additionally

$$\sum_m b_m c_m^3 = \frac{1}{4} \quad (3.17)$$

$$\sum_{m,n} b_m c_m a_{mn} c_n = \frac{1}{8} \quad (3.18)$$

$$\sum_{m,n} b_m a_{mn} c_n^2 = \frac{1}{12} \quad (3.19)$$

$$\sum_{m,n,k} b_m a_{mn} a_{nk} c_k = \frac{1}{24}. \quad (3.20)$$

Explicit, implicit and Diagonally implicit Runge-Kutta methods

Definition 3.1 (explicit and implicit methods). *A Runge-Kutta method is called explicit if $a_{mn} = 0$, $m \leq n$. Otherwise it is called implicit.*

Definition 3.2 (DIRK, SDIRK, ESDIRK). *A Runge-Kutta method is called diagonally implicit, if it is implicit and $a_{mn} = 0$, $m > n$. It is called singly diagonally implicit, if additionally $a_{mm} = a_{nn}$, $m, n \in \{1, \dots, s\}$ and explicit first stage singly diagonally implicit, if it is diagonally implicit, $a_{mm} = a_{nn}$, $m, n \in \{2, \dots, s\}$, and $a_{11} = 0$.*

For the IVP (3.1) with Nd equations, an implicit Runge-Kutta method with s stages requires the solution of a system of sNd algebraic equation. For a diagonally implicit

Runge-Kutta (DIRK) method, the system of equations can be decoupled. Hence, only the solution of s systems of size Nd is required. Explicit Runge-Kutta method can be evaluated directly without solving a system of algebraic equations. However, they have limited stability properties (cf. Section 3.2).

Embedded Runge-Kutta methods

The idea of embedded Runge-Kutta methods is to use the computed stages $\mathbf{k}_m$ for a second solution with different order (typically order p and $p + 1$) by combining them with different weights $\hat{b}_n$. In the case of an embedded DIRK method as used in this thesis, two solution $\mathbf{W}^D$ and $\hat{\mathbf{W}}^D$ are given by

$$\mathbf{z}_m = \mathbf{W}^{j-1} + \Delta t_j \sum_{n=1}^m a_{mn} \mathbf{k}_n \quad \forall m \in \{1, \dots, s\} \quad (3.21)$$

$$\mathbf{k}_m = -\mathcal{V}^{-1} \mathbf{R}(t_{j-1} + c_m \Delta t_j, \mathbf{z}_m) \quad \forall m \in \{1, \dots, s\} \quad (3.22)$$

$$\mathbf{W}^j = \mathbf{W}^{j-1} + \Delta t_j \sum_{n=1}^s b_n \mathbf{k}_n \quad (3.23)$$

$$\hat{\mathbf{W}}^j = \mathbf{W}^{j-1} + \Delta t_j \sum_{n=1}^s \hat{b}_n \mathbf{k}_n \quad (3.24)$$

It's coefficients are usually written in the form

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

Embedded Runge-Kutta methods can be used for computational inexpensive error estimation by using the difference of the two solutions (cf. Section 5.2).

3.2 Stiff equations and stability of Runge-Kutta methods

When using Runge-Kutta methods for the solution of differential equations, it is at first glance much cheaper to use explicit methods. They do not require the solution of a system of algebraic equations. However, especially for so called *stiff* equations (cf. Remark 3.16), they often have stability issues and the simulation crashes if the time step size is too large. In this chapter the theory on linear and nonlinear stability of Runge-Kutta methods and the phenomenon of order reduction for stiff differential equations is introduced.

3.2.1 Linear stability

The linear stability analysis is based on Dahlquist's test equation

$$\dot{y}(t) = \lambda y(t), \quad \lambda \in \mathbb{C}, \quad y(0) = y_0 \quad (3.25)$$

with exact solution

$$y(t) = y(0) \exp(\lambda t). \quad (3.26)$$

The exact solution is multiplied by $\exp(\lambda \Delta t)$ in a single step of length Δt . In this time interval, the approximate numerical solution is multiplied by a *stability function* $R(z)$ depending on the particular Runge-Kutta method.

Definition 3.3 (stability function). *Given a Runge-Kutta method, let $y^D(\lambda, t + \Delta t)$ be the corresponding approximate solution of (3.25) by doing one Runge-Kutta step with step size Δt starting at some value $y^D(\lambda, t)$. Then, the stability function of this method is defined by*

$$R(z) := \frac{y^D(\lambda, t + \Delta t)}{y^D(\lambda, t)} \quad (3.27)$$

with $z = \lambda \Delta t$.

Theorem 3.4. *The stability function of a Runge-Kutta method is well defined and given by*

$$R(z) = 1 + z \mathbf{b}^T (\mathcal{I} - z \mathcal{A})^{-1} \mathbf{1}. \quad (3.28)$$

Here, $\mathcal{I}$ is the identity matrix, $\mathbf{b}$ and $\mathcal{A}$ are Runge-Kutta coefficients, and $\mathbf{1}$ is the 1-vector.

Proof. see [36, Proposition 3.1]. □

Definition 3.5 (stability region). *The stability region of a Runge-Kutta method is defined as the set*

$$\{z \in \mathbb{C} : |R(z)| \leq 1\}. \quad (3.29)$$

The stability region contains all points z in the complex plane for which the approximate solution corresponding to the Runge-Kutta method remains bounded. As mentioned before, the exact solution of (3.25) is multiplied by $\exp(z)$ in a single step of length Δt . Hence, the exact solution remains bounded if z is in the left complex half plane, i. e. if $\operatorname{Re}(\lambda) \leq 0$. Ideally, the numerical method should inherit this behavior.

Definition 3.6 (A- and L-stable). *A Runge-Kutta method is called A-stable, if the stability region contains the left complex half plain, i. e. if*

$$\operatorname{Re}(z) \leq 0 \Rightarrow |R(z)| \leq 1. \quad (3.30)$$

If additionally

$$R(z) \rightarrow 0 \quad \text{as } z \rightarrow \infty, \quad (3.31)$$

the method is called L-stable. A-stable (and L-stable) methods are of particular interest for the solution of stiff differential equations, as they allow for large timesteps without getting stability issues.

Definition 3.7 ($A(\alpha)$ -, $L(\alpha)$ -stable). *Let α denote an angle satisfying $\alpha \in (0, \frac{\pi}{2})$ and let $S(\alpha)$ denote the set of points $x + iy$ in the complex plane such that $x \leq 0$ and $-\tan(\alpha)|x| \leq y \leq \tan(\alpha)|x|$. A Runge-Kutta method with stability function $R(z)$ is $A(\alpha)$ -stable if $|R(z)| \leq 1$ for all $z \in S(\alpha)$. If additionally*

$$R(z) \rightarrow 0 \quad \text{as } z \rightarrow \infty, \quad (3.32)$$

the method is called $L(\alpha)$ -stable.

Remark 3.8. *An explicit Runge-Kutta method cannot be $A(\alpha)$ -stable and thus in particular not A-stable.*

3.2.2 Nonlinear stability

Up to now, only linear stability has been discussed. However, the ordinary differential equations (ODEs) resulting from a spatial finite volume discretization of the RANS equations are in general nonlinear. Therefore, it is not clear whether A- or L-stability is sufficient for the problems at hand and an overview on the stability theory on a general possibly nonlinear problem (3.1) is given in the following. Numerical results for the problems considered in this thesis regarding linear and nonlinear stability are described in Section 7.5.

Definition 3.9 (dissipative). *The right hand side $-\nu^{-1}\mathbf{R}(t, \mathbf{W})$ of (3.1) is called dissipative if*

$$(\nu^{-1}\mathbf{R}(t, \tilde{\mathbf{W}}) - \nu^{-1}\mathbf{R}(t, \mathbf{W}))^T (\mathbf{W} - \tilde{\mathbf{W}}) \leq 0 \quad \mathbf{W}, \tilde{\mathbf{W}} \in \mathbb{R}^{N_d}, t \in I. \quad (3.33)$$

Definition 3.10 (contractive). Let $\mathbf{W}$ and $\tilde{\mathbf{W}}$ be two solutions of (3.1). The ordinary differential equation is called contractive if $\|\mathbf{W}(t) - \tilde{\mathbf{W}}(t)\| \leq \|\mathbf{W}(s) - \tilde{\mathbf{W}}(s)\|$ for all $t \geq s$.

Remark 3.11. Every ordinary differential equation (3.1) with dissipative right hand side is contractive with respect to the 2-norm, since

$$\frac{d}{dt} \|\mathbf{W}(t) - \tilde{\mathbf{W}}(t)\|_2^2 = 2(\mathcal{V}^{-1} \mathbf{R}(t, \tilde{\mathbf{W}}(t)) - \mathcal{V}^{-1} \mathbf{R}(t, \mathbf{W}(t)))^T (\mathbf{W}(t) - \tilde{\mathbf{W}}(t)) \leq 0. \quad (3.34)$$

Now a notion of nonlinear stability of a Runge-Kutta method is introduced requiring that this contractivity is reflected in the method.

Definition 3.12 (B-stability). Given an ordinary differential equation with dissipative right hand side $-\mathcal{V}^{-1} \mathbf{R}(t, \mathbf{W})$. A Runge-Kutta method is called B-stable (or contractive) if every pair of solutions $\mathbf{W}^D$ and $\tilde{\mathbf{W}}^D$ satisfy

$$\|\mathbf{W}^{j+1} - \tilde{\mathbf{W}}^{j+1}\| \leq \|\mathbf{W}^j - \tilde{\mathbf{W}}^j\| \quad \text{for all } j \quad (3.35)$$

Remark 3.13. A B-stable method is A-stable, since (3.25) with $\lambda < 0$ is dissipative and using the contractivity (B-stability) of the method yields $|R(\lambda \Delta t)| < 1$.

To formalize the conditions on a Runge-Kutta method to be B-stable the notion of algebraic stability was introduced, which is basically equivalent to B-stability (see [44] for details).

Definition 3.14. An s -stage Runge-Kutta method $\mathcal{A}, \mathbf{b}, \mathbf{c}$ is called algebraically stable if

$$b_m > 0 \quad m = 1, 2, \dots, s \quad (3.36)$$

and if the matrix $\mathcal{M}$ given by

$$\mathcal{M} = \text{diag}(\mathbf{b})\mathcal{A} + \mathcal{A}^T \text{diag}(\mathbf{b}) - \mathbf{b}\mathbf{b}^T, \quad (3.37)$$

is positive semi-definite.

Example 3.15. The SDIRK3(2) method 3.28 described in section 3.5 with

$$\mathbf{b}^T = \left(\frac{26}{61}, \frac{324}{671}, \frac{1}{11} \right) \quad (3.38)$$

and

$$\mathcal{A} = \begin{pmatrix} \frac{5}{6} & 0 & 0 \\ -\frac{61}{108} & \frac{5}{6} & 0 \\ -\frac{23}{183} & -\frac{33}{61} & \frac{5}{6} \end{pmatrix} \quad (3.39)$$

results in

$$\mathcal{M} = \begin{pmatrix} \frac{5902}{11163} & -\frac{19587}{40931} & -\frac{101}{2013} \\ -\frac{19587}{40931} & \frac{257364}{450241} & -\frac{687}{7381} \\ -\frac{101}{2013} & -\frac{687}{7381} & \frac{52}{363} \end{pmatrix}. \quad (3.40)$$

Since $\mathcal{M}$ has eigenvalues $\lambda_1 \approx 0.2132$, $\lambda_2 \approx 1.0304$ and $\lambda_3 = 0$, it is positive semidefinite. Furthermore, it is $b_m > 0$, which means the method is algebraically stable and thus B -stable (as was already shown by Nørsett and Thomsen [56]). The embedded method with

$$\hat{\mathbf{b}} = \left(\frac{25}{61}, \frac{36}{61}, 0 \right) \quad (3.41)$$

however is not algebraically stable, since

$$\hat{\mathcal{M}} = \begin{pmatrix} \frac{5750}{11163} & -\frac{6421}{11163} & 0 \\ \frac{6421}{11163} & \frac{2364}{3721} & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (3.42)$$

has a negative eigenvalue $\hat{\lambda}_1 \approx -0.0031$.

3.2.3 Order reduction and stage order

The phenomenon of order reduction was described first in a paper by Prothero and Robinson [58]. They investigated the model problem

$$\dot{y}(t) = \lambda y(t) + \dot{g}(t) - \lambda g(t), \quad t \geq 0, \quad y(0) = y_0, \quad \lambda \ll 0 \quad (3.43)$$

and found that for large $|\lambda|$ and moderately small step sizes, such that $|\lambda \Delta t|$ is large, some Runge-Kutta methods show an order of convergence which is smaller than their classical order.

Remark 3.16 (Stiffness). *The phenomenon of order reduction only occurs for stiff differential equations. For an introduction to stiffness see [24, Chapter 1]. As a short description they state that:*

“The essence of stiffness is that the solution to be computed is slowly varying but that perturbations exist which are rapidly damped.”

An other indication for stiffness especially for systems of differential equations is given when the Jacobian of the right hand side of (3.1) has eigenvalues of strongly varying magnitude.

The reduced order of a Runge-Kutta method is related to the so called *stage order*, which is given by the minimum order of the individual stage solutions of a Runge-Kutta method and can be defined in the following way.

Definition 3.17 (stage order). *A Runge-Kutta method has stage order $\tilde{p}$ if*

$$\sum_{n=1}^s a_{mn} c_n^{k-1} = \frac{c_m^k}{k} \quad k = 1, \dots, \tilde{p}, \quad m = 1, \dots, s \quad (3.44)$$

It can be shown that the reduced order of convergence is at least equal to the minimum of the order and the stage order when certain stability criteria are met. Details can be found in e. g. [24, Chapter 7].

A further important property of Runge-Kutta methods is defined as follows.

Definition 3.18 (stiffly accurate). *An s -stage Runge-Kutta method with*

$$b_n = a_{sn} \quad \forall n \in \{1, \dots, s\} \quad (3.45)$$

is called stiffly accurate.

Stiffly accurate methods have the advantage that certain parts of the discretization error cancel each other in the case of stiff problems. Therefore they typically have a smaller error compared to similar methods of the same order without this property. Details can be found in e. g. [16, Section 362] and [49]. Furthermore, since for stiffly accurate methods the solution of the last stage is equal to the solution of the Runge-Kutta method, the evaluation of (3.11) can be omitted.

3.3 Multistep methods

Alongside the described Runge-Kutta methods, linear multistep methods are well known in the field of numerical solution algorithms to ordinary differential equations. The basic idea of linear multistep methods is to use the solution values and derivatives at several previous steps to increase the convergence order of the method. In general, a linear multistep method for the computation of $\mathbf{W}^j$ for the IVP (3.1) using values and

derivatives of k previously computed values $\mathbf{W}^{j-1}, \mathbf{W}^{j-2}, \dots, \mathbf{W}^{j-k}$ can be defined by

$$\mathbf{W}^j = \sum_{i=1}^k \alpha_i \mathbf{W}^{j-i} - \Delta t_j \mathcal{V}^{-1} \sum_{i=0}^k \beta_i \mathbf{R}(t_{j-i}, \mathbf{W}^{j-i}) \quad (3.46)$$

with the coefficients $\alpha_1, \alpha_2, \dots, \alpha_k$ and $\beta_0, \beta_1, \dots, \beta_k$ describing the specific method. Such a linear multistep method can be described by polynomials

$$\alpha(z) = 1 - \alpha_1 z - \alpha_2 z^2 - \dots - \alpha_k z^k, \quad (3.47)$$

$$\beta(z) = \beta_0 + \beta_1 z + \beta_2 z^2 + \dots + \beta_k z^k, \quad (3.48)$$

and denoted by $[\alpha, \beta]$.

Definition 3.19 (explicit and implicit methods). *A linear multistep method is called explicit if $\beta_0 = 0$, otherwise it is called implicit.*

Similar to implicit Runge-Kutta methods, a system of (possibly nonlinear) algebraic equations has to be solved in every time step when using implicit linear multistep methods. This is not necessary for explicit methods.

Definition 3.20 (stability). *A linear multistep method $[\alpha, \beta]$ is stable if the difference equation*

$$y_n = \sum_{i=1}^k \alpha_i y_{n-i} \quad (3.49)$$

has only bounded solutions.

Theorem 3.21. *A linear multistep method $[\alpha, \beta]$ has order p (or higher) if and only if*

$$\alpha(\exp(z)) + z\beta(\exp(z)) = \mathcal{O}(z^{p+1}), \quad (3.50)$$

or equivalently

$$\frac{z}{\log(1+z)} \frac{\alpha(1+z)}{z} + \beta(1+z) = \mathcal{O}(z^p). \quad (3.51)$$

Proof. See [16, Theorem 410B, Theorem 410D]. $\square$

The following important results regarding the order and stability of linear multistep methods are known as the *Dahlquist barriers*.

Theorem 3.22 (First Dahlquist barrier). *Let $[\alpha, \beta]$ denote a stable linear multistep method with order p . Then*

$$p \leq \begin{cases} k+1, & k \text{ odd}, \\ k+2, & k \text{ even}. \end{cases} \quad (3.52)$$

Proof. See [16, Theorem 441C] □

Theorem 3.23 (Second Dahlquist barrier). *An A-stable linear multistep method cannot have order greater than 2.*

Proof. See [16, Theorem 443A] □

The most well known multistep methods for non-stiff problems are the explicit *Adams-Bashforth* methods and the implicit *Adams-Moulton* methods. Both families of methods are characterized by a fixed $\alpha(z) = 1 - z$ and a variable $\beta(z)$ to match the desired order. In the Adams-Bashforth case $\beta_0 = 0$ is required additionally to get explicit methods.

For stiff problems, backward difference formulae (BDFs) are some of the most used methods, since they have better stability properties than Adams-Bashforth and Adams-Moulton methods. In this thesis, the notation $\text{BDF}k$ is used for the order k member of this family. This family is given by the choice $\beta = \beta_0$ with the alpha polynomial chosen such that the method has the desired order. The coefficients for BDF methods of order $p = k \in \{1, 2, 3\}$ are given in Table 3.1.

k	α_1	α_2	α_3	β_0
1	1			1
2	$\frac{4}{3}$	$-\frac{1}{3}$		$\frac{2}{3}$
3	$\frac{18}{11}$	$-\frac{9}{11}$	$\frac{2}{11}$	$\frac{6}{11}$

Table 3.1: Coefficients of BDF methods up to order 3

BDF methods are A-stable for $k \leq 2$, $A(\alpha)$ -stable (cf. Definition 3.7) for $3 \leq k \leq 6$ with $\alpha = 1.5015 = 86.03^\circ$ for $k = 3$, $\alpha = 1.2802 = 73.35^\circ$ for $k = 4$, $\alpha = 0.9048 = 51.84^\circ$ for $k = 5$, $\alpha = 0.3114 = 17.84^\circ$ for $k = 6$ and unstable for $k > 6$.

3.4 Comparison of Multistep and Runge-Kutta methods

In this section a theoretical comparison of BDF and DIRK methods is made. The focus is on their use for time step adaptation for stiff systems of differential equations as they occur after spatial discretization of integral equations like the RANS equations.

An obvious advantage of BDFs is the fact that only one nonlinear system of equations of dimension equal to the number of ODEs has to be solved in contrast to s such systems in the case of a DIRK method ($s-1$ for explicit first stage singly diagonal implicit Runge-Kutta (ESDIRK) methods), where s is the number of stages. DIRK methods on

the other hand can be designed with better stability properties. In particular, BDFs are only A-stable for orders 1 and 2, whereas higher order DIRK methods can be designed to be A-stable (e.g. the third and fourth order DIRK methods described in Section 3.5). Furthermore, DIRK methods can be designed to have smaller error coefficients than BDFs, allowing for larger time step sizes for a given accuracy. Additionally, BDFs (except BDF1) have the disadvantage of requiring some special treatment at the beginning of the iteration, where the solutions of previous time steps are not known. Hence, the solver needs to switch to a specific start up method, e.g. a lower order BDF or a Runge-Kutta method. More details can be found for instance in [16, Section 401].

When using variable step sizes, like in the case of step size adaptation, BDFs (except BDF1) as described in Section 3.3 are no longer of the desired order. It is possible to avoid this by adapting the coefficients to the time step sizes, which is called variable coefficient BDF. Further possibilities are interpolatory BDFs and fixed leading coefficient BDFs. In the case of a large step size increase, additional stability issues arise when using these variations of the BDFs [18, 19].

Whether BDFs or DIRK methods should be preferred in the end depends on the application at hand. For CFD, BDFs with constant time steps are widely used. However, additional effort is necessary for time step size adaptation using BDFs, whereas this is more straight forward for DIRK methods. Furthermore, the required special treatment in the startup phase might be problematic, as using lower order BDFs can reduce the order of the method (cf. Section 7.1).

A numerical comparison of BDFs and DIRK methods for aerodynamic flow problems is done in Section 7.6. Time step size adaptation is only considered for the Runge-Kutta methods.

3.5 Implemented methods

In the following, the different Runge-Kutta and BDF methods used in this thesis are described. The name of the method is given by the type (BDF, explicit Runge-Kutta (ERK), singly diagonal implicit Runge-Kutta (SDIRK), or ESDIRK), the order of the propagated method and, in case of the Runge-Kutta methods, the order of the embedded method for error estimation in parenthesis.

3.5.1 Embedded explicit Runge-Kutta methods

All implemented embedded ERK methods have the first same as last property, meaning that the last stage residual of the previous time step is equal to the first stage residual

in the current time step, effectively reducing the stage number by one. In detail the methods are (the last line representing the embedded method):

Method 3.24 (ERK3(2)). *A four stage ERK method of order 3 with a 2nd order method embedded for error estimation due to Bogacki and Shampine [8] given by the tableau*

0	0			
$\frac{1}{2}$	$\frac{1}{2}$	0		
$\frac{3}{4}$	0	$\frac{3}{4}$	0	
1	$\frac{2}{9}$	$\frac{3}{9}$	$\frac{4}{9}$	0
	$\frac{2}{9}$	$\frac{3}{9}$	$\frac{4}{9}$	0
	$\frac{7}{24}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{8}$

Method 3.25 (ERK4(3)). *A five stage ERK method of order 4 with a 3rd order method embedded for error estimation. The basic method is the classical Runge-Kutta method by Kutta [48] with an embedded method given by the criteria of equation (4.9) in [36]. The method is given by the tableau*

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

Method 3.26 (ERK5(4)). *A 7 stage ERK method of order 5 with a 4th order method embedded due to Dormand and Prince [26] specifically tailored to have a small principal truncation term in the fifth order and an extended region of stability. It is given by*

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

3.5.2 Embedded diagonally implicit Runge-Kutta methods

The following DIRK methods have been implemented:

Method 3.27 (SDIRK2(1)). A 2 stage SDIRK method of order 2 with a 1st order method embedded for error estimation due to Ellsiepen [28]. The second order method is L -stable and stiffly accurate. The first order method is A -stable. The method has stage order 1. The method is given by

γ	γ
1	$1 - \gamma \quad \gamma$
	$1 - \gamma \quad \gamma$
	$1 - \alpha \quad \alpha$

with $\gamma = 1 - \frac{\sqrt{2}}{2}$, and $\alpha = 2 - \frac{5}{4}\sqrt{2}$.

Method 3.28 (SDIRK3(2)). A 3 stage SDIRK method of order 3 with a 2nd order method embedded for error estimation due to Nørsett and Thomsen [56]. The third order method is B -stable but not L -stable and has stage order 1. The embedded second order method is A -stable. It is given by the tableau

$\frac{5}{6}$	$\frac{5}{6}$		
$\frac{29}{108}$	$-\frac{61}{108}$	$\frac{5}{6}$	
$\frac{1}{6}$	$-\frac{23}{183}$	$-\frac{33}{61}$	$\frac{5}{6}$
	$\frac{26}{61}$	$\frac{324}{671}$	$\frac{1}{11}$
	$\frac{25}{61}$	$\frac{36}{61}$	

Method 3.29 (SDIRK4(3)). A 5 stage SDIRK method of order 4 with a 3rd order method embedded due to Hairer and Wanner [37]. The propagated method is L-stable, the error estimator however is not A-stable. Both methods have stage order 1. The method is given by

$\frac{1}{4}$	$\frac{1}{4}$				
$\frac{3}{4}$	$\frac{1}{2}$	$\frac{1}{4}$			
$\frac{11}{20}$	$\frac{17}{50}$	$-\frac{1}{25}$	$\frac{1}{4}$		
$\frac{1}{2}$	$\frac{371}{1360}$	$-\frac{137}{2720}$	$\frac{15}{544}$	$\frac{1}{4}$	
1	$\frac{25}{24}$	$-\frac{49}{48}$	$\frac{125}{16}$	$-\frac{85}{12}$	$\frac{1}{4}$
	$\frac{25}{24}$	$-\frac{49}{48}$	$\frac{125}{16}$	$-\frac{85}{12}$	$\frac{1}{4}$
	$\frac{59}{48}$	$-\frac{17}{96}$	$\frac{225}{32}$	$-\frac{85}{12}$	0

Method 3.30 (ESDIRK2(1)). A 2 stage ESDIRK method of order 2 with a 1st order method embedded for error estimation due to Crank and Nicolson [23]. The second order method is A-stable with stage order 2 and stiffly accurate. The embedded method however is not A-stable. The method is given by

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

Method 3.31 (ESDIRK3(2)). A 4 stage ESDIRK method of order 3 with a 2nd order method embedded for error estimation due to Kværnø [49]. The method has stage order 2. The third order method is L-stable, the second order method A-stable and both methods are stiffly accurate (meaning stiffly accurate for the reduced method without the last stage in case of the embedded method). The method is given by

0	0			
2γ	γ	γ		
1	0.4905633884	0.0735700901	γ	
1	0.3088099700	1.4905633882	-1.2352398797	γ
	0.3088099700	1.4905633882	-1.2352398797	γ
	0.4905633884	0.0735700901	γ	0

with $\gamma = 0.4358665215$.

Method 3.32 (ESDIRK3(4)). A 5 stage ESDIRK method of order 4 with a 3rd order method embedded. Due to its L-stability, the third order method is used to propagate the solution and the fourth order method for error estimation. The method is also due to Kværnø [49]. It has stage order 2. The third order method is L-stable, the fourth order method is A-stable and both methods are stiffly accurate (meaning stiffly accurate for the reduced method without the last stage in case of the propagated method). It is given by

0	0					
2γ	γ	γ				
0.4682387448	0.1407377747	-0.1083655514	γ			
1	0.1023994006	-0.3768784523	0.8386125302	γ		
1	0.1570248979	0.1173304413	0.6166780304	-0.3268998911	γ	
	0.1023994006	-0.3768784523	0.8386125302	γ		
	0.1570248979	0.1173304413	0.6166780304	-0.3268998911	γ	

with $\gamma = 0.4358665215$ as before.

Method	Stages	Order	Stage order	B-stable	L-stable	A-stable	Stiffly acc.
ERK3(2)	4	3(2)	1	-(-)	-(-)	-(-)	-(-)
ERK4(3)	5	4(3)	1	-(-)	-(-)	-(-)	-(-)
ERK5(4)	7	5(4)	1	-(-)	-(-)	-(-)	-(-)
SDIRK2(1)	2	2(1)	1	-(-)	x(-)	x(x)	x(-)
SDIRK3(2)	3	3(2)	1	x(-)	-(-)	x(x)	-(-)
SDIRK4(3)	5	4(3)	1	-(-)	x(-)	x(-)	x(-)
ESDIRK2(1)	2	2(1)	2	-(-)	-(-)	x(-)	x(-)
ESDIRK3(2)	4	3(2)	2	-(-)	x(-)	x(x)	x(x)
ESDIRK3(4)	5	3(4)	2	-(-)	x(-)	x(x)	x(x)

Table 3.2: Properties of investigated Runge-Kutta methods, method for error estimation in parenthesis

3.5.3 Backward Difference Formulae

The BDFs used in this thesis were already implemented in the TAU code. Available methods are of order 1, 2 and 3 as described in Section 3.3. The corresponding coefficients are given in Table 3.1. Only constant time stepping is available. The start up of BDF2 and BDF3 is done by using BDF1 for the first step and BDF2 for the second time step.

However, this start up procedure reduces the convergence order of BDF3 to 2. Hence, a modified start up strategy using the Runge-Kutta Method 3.31 for the first two steps of the BDF3 computations is also considered for investigations in Section 7.1.

CHAPTER 4

Solution of nonlinear algebraic systems

4.1 Solution methods

When using implicit Runge-Kutta methods, it is in general necessary to solve (3.9) for all $m = 1, \dots, s$ simultaneously, resulting in a system of equations of size sNd in each time step. Here, s is the number of Runge-Kutta stages, N is the number of grid points in the spatial discretization, and d is the number of equations (6 for the RANS equations used here). In case of DIRK methods, the system of equations decouples into s systems of size Nd given in (3.21). Those systems can be solved numerically by an iterative solution procedure yielding iterates $\mathbf{z}_m^{(0)}, \dots, \mathbf{z}_m^{(n)}$ and the approximate solution $\mathbf{z}_m \approx \mathbf{z}_m^{(n)}$. Different choices for the initial value $\mathbf{z}_m^{(0)}$, as well as criteria to stop the iterative solution process are treated in Section 4.2 and Section 4.3, respectively. A well known iterative solution procedure for nonlinear systems of equations is the *Newton algorithm*, which calculates the iterates by solving the linear system of equations

$$\left(\frac{\mathcal{V}}{a_{mm}\Delta t_j} + \tilde{\mathcal{J}}\right)\Delta \mathbf{z}_m^{(l+1)} = -\tilde{\mathbf{R}}(\mathbf{z}_m^{(l)}), \quad (4.1)$$

$$\mathbf{z}_m^{(l+1)} = \mathbf{z}_m^{(l)} + \Delta \mathbf{z}_m^{(l+1)}, \quad (4.2)$$

with unsteady residual

$$\tilde{\mathbf{R}}(\mathbf{z}_m^{(l)}) := \frac{\mathcal{V}}{a_{mm}\Delta t_j}(\mathbf{z}_m^{(l)} - \tilde{\mathbf{W}}) + \mathbf{R}(\mathbf{z}_m^{(l)}), \quad (4.3)$$

$$\tilde{\mathbf{W}} := \mathbf{W}^{j-1} - \Delta t_j \mathcal{V}^{-1} \sum_{n=1}^{m-1} a_{mn} \mathbf{R}(\mathbf{z}_n) \quad (4.4)$$

and the Jacobian

$$\tilde{\mathcal{J}} = \left. \frac{\partial \mathbf{R}}{\partial \mathbf{W}} \right|_{\mathbf{z}_m^{(l)}}. \quad (4.5)$$

The TAU code uses a generalization of the Newton method to solve the nonlinear systems of equations (3.21). The method used in TAU is introduced in the following, basically following [53], where the corresponding solver for a steady state simulation is described. Note that the algorithms presented here were already available in the TAU code and have been adapted to work with DIRK methods in the context of this thesis. The solution algorithm can in general be formulated as

$$\begin{aligned} \tilde{\mathbf{z}}^{(0)} &:= \mathbf{z}_m^{(l)} \\ \tilde{\mathbf{z}}^{(1)} &:= \tilde{\mathbf{z}}^{(0)} - \alpha_{21} \mathcal{P}_1(\tilde{\mathbf{z}}^{(0)})^{-1} \tilde{\mathbf{R}}(\tilde{\mathbf{z}}^{(0)}) \\ &\vdots \\ \tilde{\mathbf{z}}^{(r)} &:= \tilde{\mathbf{z}}^{(0)} - \alpha_{r+1,r} \mathcal{P}_r(\tilde{\mathbf{z}}^{(r-1)})^{-1} \tilde{\mathbf{R}}(\tilde{\mathbf{z}}^{(r-1)}) \\ \mathbf{z}_m^{(l+1)} &:= \tilde{\mathbf{z}}^{(r)}, \end{aligned} \quad (4.6)$$

with

$$\mathcal{P}_k(\tilde{\mathbf{z}}^{(k-1)}) := \mathcal{V} \left(\mathcal{D}^{(l)-1} + \frac{\mathcal{I}}{a_{mm} \Delta t_j} \right) + \mathcal{J}. \quad (4.7)$$

Here,

$$\mathcal{J} \approx \left. \frac{\partial \mathbf{R}}{\partial \mathbf{W}} \right|_{\tilde{\mathbf{z}}^{(k-1)}} \quad (4.8)$$

is an approximation to the Jacobian of the residual $\mathbf{R}$ and

$$\mathcal{D}^{(l)} = \text{diag}(\Delta \tau_i^{(l)}) \otimes \mathcal{I}_d \quad (4.9)$$

(with Kronecker product $\otimes$) is the diagonal matrix consisting of the local pseudo time steps

$$\begin{aligned} \Delta \tau_i^{(l)} &:= \text{CFL} \cdot \text{vol}(\Omega_i) \left(\sum_{j \in \mathcal{N}(i)} \frac{1}{2} \Lambda_{ij} \right. \\ &\quad \left. + \frac{C_\nu (\mu_{\text{eff}})_{ij} \text{vol}(\Omega_{ij})}{\|\mathbf{x}_i - \mathbf{x}_j\|_{2\rho_i}} \left(\max \left(\frac{4}{3}, \frac{(\kappa_{\text{eff}})_{ij}(\gamma - 1)}{(\mu_{\text{eff}})_{ij}} \right) \right) \right)^{-1} \bigg|_{\mathbf{z}_m^{(l)}}, \quad C_\nu := 8 \end{aligned} \quad (4.10)$$

with maximal convective eigenvalue Λ_{ij} as in (2.37). As shown in [52], the solution method can be derived by a diagonally implicit Runge-Kutta method and can be in-

interpreted as a Rosenbrock method due to the linearization. Note that choosing $r = 1$, $\alpha_{21} = 1$, $\text{CFL} = \infty$, i.e. omitting the term $\mathcal{D}^{(l+1)-1}$ in (4.7), and taking the exact Jacobian $\mathcal{J}$, yields the Newton algorithm. Additional stages $r > 1$ can be introduced to stabilize the solver by choosing suitable $0 < \alpha_{21} < \dots < \alpha_{r+1,r} = 1$. Furthermore, the parameter CFL can be tuned and adapted during the simulation. The idea of this Levenberg-Marquardt type algorithm is to stabilize the solution process by choosing smaller CFL numbers if necessary and getting the fast convergence of Newton's method as CFL approaches infinity.

Note that $\mathcal{P}_k(\tilde{\mathbf{z}}^{(k-1)})^{-1}$ in (4.6) is not computed explicitly, but the corresponding linear system of equations is solved approximatively. Different solution methods are available in TAU differing in the approximation of $\mathcal{J}$ and the choice of the linear system solver. There are two choices regarding the approximation of the Jacobian $\mathcal{J}$ in TAU. One is based on the 1st order approximation of the residual, where the dissipation

$$\mathbf{D}_{ij}(\mathbf{W}) := \frac{1}{2}|\mathcal{A}_{ij}|(\mathbf{W}_j - \mathbf{W}_i) \quad (4.11)$$

is used in (2.30) for the inviscid terms and a thin shear layer approximation (cf. [6, Section 2.4.3]) for the viscous terms. Furthermore, $|\mathcal{A}_{ij}|$, μ_{eff} , and κ_{eff} are assumed to be constant. Details of the approximation can be found in [50, 51]. The second approach is a further simplification based on taking the spectral radius of the terms of the first order Jacobian (cf. [53]). In the following, we denote $\mathcal{P}_k$ by $\mathcal{P}_k^{\text{1st}}$ using the first approximation for $\mathcal{J}$ and by $\mathcal{P}_k^{\text{1st,scalar}}$ using the second approach.

In the latter case, the resulting linear system is solved approximatively using one symmetric Gauss-Seidel sweep. By splitting $\mathcal{P}_k^{\text{1st,scalar}}$ in a lower triangular-, diagonal- and upper triangular- matrix

$$\mathcal{P}_k^{\text{1st,scalar}} = (\mathcal{L} + \mathcal{D} + \mathcal{U}), \quad (4.12)$$

the Gauss-Seidel sweep is coded using

$$(\mathcal{D} + \mathcal{L})\mathcal{D}^{-1}(\mathcal{D} + \mathcal{U})\Delta\tilde{\mathbf{z}}^{(k)} = -\alpha_{k+1,k} \left(\frac{\mathcal{V}}{\Delta t_j a_{mm}} (\tilde{\mathbf{z}}^{(k-1)} - \tilde{\mathbf{W}}) + \mathbf{R}(\tilde{\mathbf{z}}^{(k-1)}) \right). \quad (4.13)$$

This scheme using one stage in (4.6) is the current standard in TAU and is known as the lower-upper symmetric Gauss-Seidel (LU-SGS) method introduced by Yoon and Jameson [65].

In case of the first approach, a linear system with system matrix $\mathcal{P}_k^{\text{1st}}$, denoted shortly by $\mathcal{P}$ in the following, has to be solved approximatively. To do this, several matrix

splitting methods like (Line) (Symmetric) Block Gauss-Seidel or a (Line) Block-Jacobi method with a prescribed number of iterations can be used. In this thesis, we use 5 sweeps of the Line Symmetric Block Gauss-Seidel (LSBGS) method to solve the system if not stated otherwise. For this method, an Algorithm (described in [51]) determines lines in the computational grid. This yields a family of sets G_i , $i = 1, \dots, n$ with

$$G_i \subset G, \quad G_i \cap G_j = \emptyset, \quad i \neq j, \quad \cup_{i=1}^n G_i = G \quad (4.14)$$

representing the lines in the grid. Then for each line, a block tridiagonal matrix $\mathcal{M}_{G_i}$ is formed out of $\mathcal{P}$ by only keeping the diagonal blocks corresponding to cells $j \in G_i$ as well as up to two off-diagonal blocks corresponding to adjacent cells lying on the line i and deleting all lines and rows corresponding to cells not on the line. The LSBGS method then consists of a forward sweep of solving the systems

$$\mathcal{M}_{G_i} \Delta \tilde{\mathbf{z}}_{G_i}^{(k,q+1)} = \mathbf{b}_{G_i}^{(k)} - \sum_{s \in G_j, j < i} \mathcal{P}_{G_i,s} \Delta \tilde{\mathbf{z}}_s^{(k,q+1)} - \sum_{s \in G_j, j > i} \mathcal{P}_{G_i,s} \Delta \tilde{\mathbf{z}}_s^{(k,q)} \quad (4.15)$$

by a Thomas algorithm, and a corresponding backward sweep of solving

$$\mathcal{M}_{G_i} \Delta \tilde{\mathbf{z}}_{G_i}^{(k,q+1)} = \mathbf{b}_{G_i}^{(k)} - \sum_{s \in G_j, j < i} \mathcal{P}_{G_i,s} \Delta \tilde{\mathbf{z}}_s^{(k,q)} - \sum_{s \in G_j, j > i} \mathcal{P}_{G_i,s} \Delta \tilde{\mathbf{z}}_s^{(k,q+1)}, \quad (4.16)$$

with

$$\mathbf{b}^{(k)} = -\alpha_{k+1,k} \left(\frac{\mathcal{V}}{\Delta t_j a_{mm}} (\tilde{\mathbf{z}}^{(k-1)} - \tilde{\mathbf{W}}) + \mathbf{R}(\tilde{\mathbf{z}}^{(k-1)}) \right). \quad (4.17)$$

The algorithms (4.6) above are used in general as a smoother within a nonlinear agglomeration multigrid algorithm (full approximation scheme). Details on the multigrid algorithm can be found in [53]. Furthermore, it is possible to run the solver in parallel using domain decomposition. In case of the LSBGS method, the parallelization is done in the following way:

1. Separate lines across domain boundaries such that no line crosses the boundary and the Thomas algorithm runs within a domain.
2. Communicate the required data, that is the $\tilde{\mathbf{z}}^{(k,r)}$ of points next to the domain boundary, to the respective neighbor domain after each sweep. No further data is required since $\mathcal{P}_k^{1st}$ has entries for a point and it's direct neighbors only in each row.

In this thesis, when we speak of using either the LU-SGS or the LSBGS method, this

refers to using the respective solution method presented above. Note that the LSBGS method is a recent and ongoing development in TAU. In the unsteady turbulent transport aircraft gust encounter test case (cf. Section 6.5.3) presented in this thesis, we had convergence issues when using this method. On the other hand, improved convergence speed was usually observed in the other presented test cases. Therefore, if not stated otherwise, the LU-SGS method is used for the transport aircraft gust encounter test case and LSBGS is used for other test cases in this thesis. In case of the turbulent simulations using LU-SGS, multigrid was found to be beneficial and is used in this thesis. When using LSBGS for the other investigated cases however, no advantage in using multigrid was found. Furthermore, in these cases it was beneficial to use one stage in (4.6), very high CFL numbers and to keep the matrix $\mathcal{P}_k^{1st}$ constant within the solution procedure of each nonlinear system. The resulting solution method is similar to a simplified Newton algorithm with an approximate Jacobian and an approximate solution of the linear systems. Note that these observations regarding the LSBGS method are based only on the unsteady simulations presented here. Further details can be found in Section 7.4.1.

4.2 Stage value predictor

The choice of a stage value predictor giving the initial solution in the iterative solver for the nonlinear system in (3.21) can have a significant influence on the computational cost and stability of the solution process. In this thesis, three different choices are investigated.

1. $\mathbf{z}_m^{(0)} = \mathbf{W}^{j-1}$, $m = 1, \dots, s$,
2. $\mathbf{z}_m^{(0)} = \mathbf{z}_n$, $m = 1, \dots, s$, $n < m$,
3. $\mathbf{z}_m^{(0)} = \mathbf{z}_n + \Delta t^j (c_m - c_n) \mathbf{k}_n$, $m = 1, \dots, s$, $n < m$.

Here, the notation $\mathbf{z}_0 := \mathbf{W}^{j-1}$, $k_0 := 0$, and $c_0 := 0$ is used. A good choice of n in the last two options depends on the specific Runge-Kutta method. Typical choices include $n = m - 1$ or choosing $n < m$ such that the difference of Runge-Kutta coefficients $|c_m - c_n|$ is minimal. For the ESDIRK3(2) method (Method 3.31) and the SDIRK2(1) method (Method 3.27), which are used for the investigations in Section 7.4.2, both choices coincide. Depending on the specific situation and problem at hand, any of these choices could be superior. However, in most cases we would assume choice one to be inferior to the other choices, since the latter take information from already computed stages into account. Typically, using an extrapolation as in the third choice is advantageous, especially for smooth solutions. However, in the case of a moving shock front,

an extrapolation could lead to significant overshoots as sketched in Figure 4.1 for linear extrapolation. If not stated otherwise, Choice 2 is used in this thesis. Investigations regarding the choice of stage value predictor are described in Section 7.4.2.

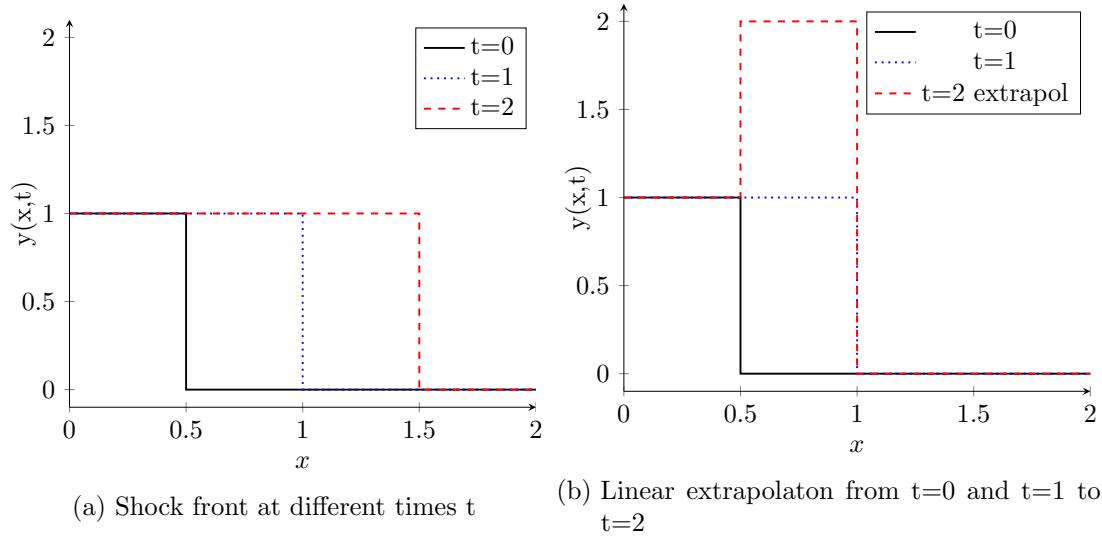


Figure 4.1: Extrapolation in case of a moving shock, a linear extrapolation of the solution at $t=2$ from solutions at $t=0$ and $t=1$ results in the solution in Figure 4.1b instead of the $t=2$ curve in Figure 4.1a

4.3 Truncation criteria

When using iterative solution methods for systems of nonlinear equations, as described in Section 4.1, a criterion is needed to signify the truncation of the iterative solution process. Ideally, the criterion should ensure that the solution of the system of nonlinear equations is accurate enough, such that the error has no significant influence on the simulation results. Furthermore, it should also not be too strict in order to avoid excessive computational costs. In TAU, one can choose between a residual based criterion and a criterion based on the difference of subsequent solution approximations. The former is based on a reduction of the unsteady residual (4.3). The criterion is based on the 2-norm of the unsteady density residual weighted with the inverse cell volume and the number of grid cells, i. e. the seminorm

$$\|\tilde{\mathbf{R}}(\mathbf{z}_m^{(l)})\|_\rho := \sqrt{\frac{1}{N} \sum_{i=1}^N \left(\frac{\tilde{\mathbf{R}}(\mathbf{z}_m^{(l)})_{di+1}}{\text{vol}(\Omega_i)} \right)^2} \quad (4.18)$$

is used. The resulting value is then normalized either by the first unsteady residual of the solution process $\|\tilde{\mathbf{R}}(\mathbf{z}_m^{(0)})\|_\rho$ or the steady freestream residual $\|\mathbf{R}(\mathbf{W}_{\text{freestream}})\|_\rho$. If for a user defined value θ

$$\frac{\|\tilde{\mathbf{R}}(\mathbf{z}_m^{(l)})\|_\rho}{\|\tilde{\mathbf{R}}(\mathbf{z}_m^{(0)})\|_\rho} < \theta \quad (4.19)$$

in the first case, or

$$\frac{\|\tilde{\mathbf{R}}(\mathbf{z}_m^{(l)})\|_\rho}{\|\mathbf{R}(\mathbf{W}_{\text{freestream}})\|_\rho} < \theta \quad (4.20)$$

in the second case, the solution process for the system of nonlinear equations is truncated.

The second criterion choice is based on the difference of some functional $f : \mathbb{R}^{Nd} \rightarrow \mathbb{R}$ of the current and previous solution approximations $f(\mathbf{z}_m^{(l)}) - f(\mathbf{z}_m^{(l-k)})$, $k = 1, \dots, n$. In detail, it is required that

$$|f(\mathbf{z}_m^{(l)}) - f(\mathbf{z}_m^{(l-k)})| < \Theta, \quad k = 1, \dots, n \quad (4.21)$$

for absolute convergence or

$$\frac{|f(\mathbf{z}_m^{(l)}) - f(\mathbf{z}_m^{(l-k)})|}{|f(\mathbf{z}_m^{(l)})|} < \Theta, \quad k = 1, \dots, n \quad (4.22)$$

for relative convergence, with user defined n , Θ and f . There are many possibilities implemented for the choice of f . Typical choices are the lift-, drag-, and pitching moment coefficient.

In TAU, it is still up to the user to specify an appropriate truncation criterion. An automatic choice linked to the tolerance of the time step size adaptation would be desirable in the future. However, finding an appropriate criterion is not a trivial task, because the error of the approximate solution has to be estimated. There are suggestions, for instance using the error estimate given by Banach's fixed point theorem [25, Section 6.2.2], but it requires linear convergence of the solver, which can not be guaranteed. There are also estimates of the error based on the residual, as e. g. described in [4, Section 5.4]. However, the estimate depends on the condition number of the Jacobian, which is typically large for complex turbulent flow simulations.

In summary, there are three convergence criteria described above, namely the

1. residual based criterion normed by the first unsteady residual of the solution process
2. residual based criterion normed by the steady freestream residual

3. criterion based on the difference of subsequent iterates.

The third choice has the disadvantage that it declares convergence if the solver stagnates at some point. The first choice on the other hand has the disadvantage that it reduces the residual by a given amount independent of the initial value of the solution process. Hence, unless the solver is superlinear convergent, a good stage value predictor might not result in a lower computational cost, but a more stringent convergence criterion. Therefore, the second criterion is chosen for the numerical tests in this thesis.

CHAPTER 5

Time step size adaptation algorithms

In the next section, the general theory on step size control as well as several adaptation algorithms and controllers are introduced. Afterwards, the idea of local error estimators is sketched. As in Chapter 3, the theory is again stated for the initial value problem (3.1).

5.1 Time step estimator

The aim of step size control is to estimate a time step at each time t_j such that it is as large as possible while satisfying some given error bound. The error bound could be given either in terms of the local or the global discretization error which are defined as follows (with exact evolution Ψ and numerical evolution Φ as in (3.7) and (3.8)).

Definition 5.1 (global discretization error). *The global discretization error $\epsilon_j^{\text{global}}$ of a numerical method for the IVP (3.1) with discretization Δ (given by the evolution Ψ) at time t_j is given by*

$$\epsilon_j^{\text{global}} := \Psi^{t_j, t_0} \mathbf{W}^0 - \Phi^{t_j, t_0} \mathbf{W}^0, \quad (5.1)$$

i. e.

$$\epsilon_j^{\text{global}} = \mathbf{W}^j - \mathbf{W}(t^j) \quad (5.2)$$

Definition 5.2 (local discretization error). *The local discretization error $\epsilon_j^{\text{local}}$ of a numerical method for the IVP (3.1) with discretization Δ at time t_j is given by*

$$\epsilon_j^{\text{local}} := \Psi^{t_j, t_{j-1}} \mathbf{W}^{j-1} - \Phi^{t_j, t_{j-1}} \mathbf{W}^{j-1}. \quad (5.3)$$

The global and local errors are connected by

$$\epsilon_j^{\text{global}} = \Psi^{t_j, t_{j-1}} \mathbf{W}^{j-1} - \Phi^{t_j, t_{j-1}} \mathbf{W}(t_{j-1}) \quad (5.4)$$

$$= (\Psi^{t_j, t_{j-1}} \mathbf{W}^{j-1} - \Phi^{t_j, t_{j-1}} \mathbf{W}^{j-1}) + (\Phi^{t_j, t_{j-1}} \mathbf{W}^{j-1} - \Phi^{t_j, t_{j-1}} \mathbf{W}(t_{j-1})) \quad (5.5)$$

$$= \epsilon_j^{\text{local}} + (\Phi^{t_j, t_{j-1}} \mathbf{W}^{j-1} - \Phi^{t_j, t_{j-1}} \mathbf{W}(t_{j-1})). \quad (5.6)$$

The second part on the right hand side of the equation (5.6) can be interpreted as the propagation of the error $\epsilon_{j-1}^{\text{global}}$. It is determined by the previously accumulated errors and the stability of the IVP itself. Since the IVP is solved sequentially, sitting at time t_{j-1} the propagated error part of $\epsilon_j^{\text{global}}$ can not be influenced unless the simulation is recomputed.

Nevertheless, the error propagation has an influence on the optimal step size. If the problem is unstable, the local errors at the beginning of the time interval I have the largest impact on the global error at the end. On the other hand, the local errors in the later steps have a larger influence if the propagated error is damped. However, the information about the error propagation properties of the IVP is not always directly available. Here, in absence of further information, it is assumed that the error is neither increased nor decreased by the propagation. In this case, the global error is just the sum of the local errors. Furthermore, assume that the local error for a numerical method of order p in step j in some norm $\|\cdot\|$ is given by $c_j \Delta t_j^{p+1}$ for some constant $c_j > 0$. Thus, higher order terms are neglected, which is asymptotically correct in the limit $\Delta t_j \rightarrow 0$.

Given these assumptions, what is the optimal discretization of the interval $I = [t_s, t_e]$ such that the global error is minimized while the number of time steps $(n + 1)$ remains constant?

One way to answer this question is to solve the optimization problem

$$\min_{\Delta t_j \geq 0} \quad \|\epsilon_n^{\text{global}}(\Delta t_1, \dots, \Delta t_n)\| = \sum_{j=1}^n c_j \Delta t_j^{p+1} \quad (5.7)$$

$$\text{s. t.} \quad \sum_{j=1}^n \Delta t_j = t_e - t_s$$

with derivatives

$$\frac{\partial \|\epsilon_n^{\text{global}}\|}{\partial \Delta t_j} = c_j(p+1) \Delta t_j^p \geq 0, \quad j = 1, \dots, n. \quad (5.8)$$

Lemma 5.3. *The unique optimal solution of (5.7) is achieved if and only if the Δt_j are*

such that

$$\frac{\partial \|\epsilon_n^{\text{global}}\|}{\partial \Delta t_i} = \frac{\partial \|\epsilon_n^{\text{global}}\|}{\partial \Delta t_j} \quad \text{for all } i, j = 1, \dots, n. \quad (5.9)$$

Proof. The $\Delta t_j, j = 1, \dots, n$ are restricted to the compact interval $[0, t_e - t_s]$ and (5.7) is continuous. Hence, a minimum exists. Furthermore, the derivatives (5.8) are continuous, monotonically increasing in Δt_j and 0 for $\Delta t_j = 0$. Thus, there exists exactly one tuple $(\Delta t_1, \dots, \Delta t_n)$ satisfying the constraints of (5.7) as well as equation (5.9). Now, we only have to show that (5.9) is necessary for optimality in the interior and that optimality is not achieved on the boundaries. Therefore, assume $\frac{\partial \|\epsilon_n^{\text{global}}\|}{\partial \Delta t_i} > \frac{\partial \|\epsilon_n^{\text{global}}\|}{\partial \Delta t_j}$. Then there exists $\epsilon > 0$ such that

$$\|\epsilon_n^{\text{global}}(\Delta t_1, \dots, \Delta t_i, \dots, \Delta t_j, \dots, \Delta t_N)\| > \|\epsilon_n^{\text{global}}(\Delta t_1, \dots, \Delta t_i - \epsilon, \dots, \Delta t_j + \epsilon, \dots, \Delta t_N)\| \quad (5.10)$$

and optimality is not achieved. For tuples on the boundary, there is always at least one $j \in \{1, \dots, n\}$ such that $\Delta t_j = 0$ and thus the corresponding partial derivative is also zero. With a similar argument as above, conclude that optimality is not achieved on the boundary. $\square$

Denote this unique optimal solution by $\Delta t_j^*, j = 1, \dots, n$ and define

$$\text{tol} := \left. \frac{\partial \|\epsilon_n^{\text{global}}\|}{\partial \Delta t_j} \right|_{\Delta t_j^*} \frac{1}{p+1}. \quad (5.11)$$

Using (5.8), this yields the corresponding local error

$$\|\epsilon_j^{\text{local}*}\| = c_j(\Delta t_j^*)^{p+1} \stackrel{(5.8)}{=} \text{tol} \Delta t_j^* \quad (5.12)$$

and the global error is

$$\|\epsilon_N^{\text{global}*}\| = \text{tol}(t_e - t_s). \quad (5.13)$$

Hence, tol is the error per unit time step.

For the step size control using (error per unit time step), demand that the local error is restricted by

$$\|\epsilon_j^{\text{local}}\| \leq \text{tol} \Delta t_j. \quad (5.14)$$

Here, the error bound tol is an input and the number of time steps n necessary to achieve this error bound will be determined automatically. However, the exact value of the local error can usually not be computed, since this would require the knowledge of the exact evolution. Instead, some error estimation $\|[\epsilon_j^{\text{local}}]\|$ to approximate the local error is used

and

$$\|[\epsilon_j^{\text{local}}]\| \leq \text{tol} \Delta t_j \quad (5.15)$$

is required. After each time step, check for this condition and accept the result if it is satisfied. If the local error estimate is too large, the time step size has to be adjusted and the step recomputed.

Alternatively, choose to restrict the (error per time step) rather than the error per unit time step by demanding

$$\|[\epsilon_j^{\text{local}}]\| \leq \text{tol}. \quad (5.16)$$

While the error per unit time is best applied to problems where the propagated error neither increases nor decreases, the error per time step is of good use for stable problems where the error from previous time steps is damped. In the following subsections, methods to control the time step based on this criterion are discussed.

5.1.1 Elementary (integral) control

In this subsection the derivation of a widely used elementary method to control the step size is shown.

First, note that for a method of order p the local error asymptotically behaves like

$$\|[\epsilon_j^{\text{local}}]\| \approx \|\epsilon_j^{\text{local}}\| = c_j \Delta t_j^{p+1} + \mathcal{O}(\Delta t_j^{p+2}) \approx c_j \Delta t_j^{p+1} \quad (5.17)$$

as $\Delta t_j \rightarrow 0$ if $c_j \neq 0$. Here, c_j is a usually unknown error constant. The underlying assumptions for the elementary controller are as follows:

- 1. Asymptotics:** For a method of order p assume that (5.17) is a reasonable approximation of the local error, i.e. that the time step is small enough for this asymptotic approximation and $c_j \neq 0$.
- 2. Slow variation:** Assume that the error constant c_j varies slowly.

The second assumption means that the optimal time step size in the next step will be similar to the current one. Thus, the current estimated error can be used to compute the next time step size.

So how is the new time step size determined? For error per unit time step, the new time step Δt should fulfill

$$\Delta t \text{ tol} \approx \|[\epsilon_j^{\text{local}}]\| \approx c_j \Delta t^{p+1}. \quad (5.18)$$

Dividing (5.18) by (5.17) gives

$$\frac{\Delta t \text{ tol}}{\|[\epsilon_j^{\text{local}}]\|} \approx \frac{\Delta t^{p+1}}{\Delta t_j^{p+1}} \quad (5.19)$$

and thus

$$\Delta t \approx \sqrt[p]{\frac{\Delta t_j \text{ tol}}{\|[\epsilon_j^{\text{local}}]\|}} \Delta t_j. \quad (5.20)$$

To avoid frequent rejections of a step, a user defined certainty factor $\eta < 1$ is introduced and the new time step size is computed by

$$\Delta t = \sqrt[p]{\frac{\Delta t_j \eta \text{ tol}}{\|[\epsilon_j^{\text{local}}]\|}} \Delta t_j. \quad (5.21)$$

Similarly, choosing error per time step yields

$$\Delta t = \sqrt[p+1]{\frac{\eta \text{ tol}}{\|[\epsilon_j^{\text{local}}]\|}} \Delta t_j \quad (5.22)$$

and in general it is

$$\Delta t = \left(\frac{\eta}{r_j} \right)^{\frac{1}{k}} \Delta t_j \quad (5.23)$$

with $r_j = \frac{\|[\epsilon_j^{\text{local}}]\|}{\text{tol}}$, $k = p + 1$ for error per time step and $r_j = \frac{\|[\epsilon_j^{\text{local}}]\|}{\text{tol} \Delta t_j}$, $k = p$ for error per unit time step control. The error estimate r_j might also be computed slightly differently by use of a combination of absolute and relative error tolerances as described in Section 5.3.

Remark 5.4 (Safety factor). *Instead of multiplying the step size by the safety factor directly, the tolerance is reduced to $\eta \text{ tol}$ to compute the new time step size. This means that the controller aims for an error of $\eta \text{ tol}$, $0 < \eta < 1$ and a step is rejected if the error estimate exceeds the user defined tolerance tol , or possibly $\phi \text{ tol}$ ($\phi \Delta t_j \text{ tol}$ for error per unit time step control) with a rejection factor $\phi \geq 1$. In comparison, multiplying the step size by a safety factor $\tilde{\eta}$ directly would be equivalent to multiplying the tolerance by $\eta = \tilde{\eta}^k$ in case of the elementary controller. For other controllers, this relation can differ.*

Sometimes, the error estimate might get very small or even zero. For this case a factor $q_{\max} > 1$ determining the maximal increase of the time step size is introduced. Similarly, $q_{\min} < 1$ describes the maximal decrease of the time step size. The general algorithm to

solve (3.1) with elementary step size control is given by Algorithm 1 using Algorithm 2 in line 22 to compute the time step size with (5.23) as controller in line 2 of Algorithm 2. In order to use this algorithm, a suitable error estimator as well as an error norm or functional (line 18 of Algorithm 1) has to be determined. Those topics will be discussed in more detail in Sections 5.2 and 5.3, respectively.

5.1.2 Advanced controllers

The elementary control algorithm from Section 5.1.1 works well if the assumptions are fulfilled. However, this is often not true in practice. Especially for stiff initial value problems (cf. 3.16) the time step size is often outside the asymptotic region. Furthermore, the assumption of slow variation of the error constant often does not hold either.

In the case of a stiff initial value problem solved by an explicit time integration method Gustafsson [34, 35] found that the elementary control leads to a time step size strongly oscillating around the stability limit of the method with a lot of step rejections. Based on this observation he investigated in control theoretic approaches to improve the adaptation algorithm. He found that a so called proportional-integral (PI) stepsize control algorithm resulted in smoother step size selections for explicit Runge-Kutta methods. Later on, Söderlind [60, 61] continued his work and introduced *digital filters* for step size control also for implicit methods.

PI(D)-controller

Following [25, Chapter 5.2], a proportional-integral-differential (PID) controller can be used to control a quantity $\xi \in \mathbb{R}$, aiming at finding $\xi = \xi^*$, such that

$$F(\xi^*) = 0 \in \mathbb{R} \quad (5.24)$$

for some function $F : \mathbb{R} \rightarrow \mathbb{R}$. In linear control theory, a linear model F_0 of F is taken, i. e.

$$F(\xi) \approx F_0(\xi) = \alpha(\xi - \xi_0), \quad (5.25)$$

and it is assumed that the output of F can be written as the output of F_0 with a perturbed input

$$F(\xi) = F_0(\xi + \delta). \quad (5.26)$$

A PID controller then gives iterates

$$\xi_j := \xi_0 - \delta_j \quad (5.27)$$

Algorithm 1 Basic adaptation algorithm to solve (3.1, 3.2)

Input: Discrete evolution Ψ of order p , error estimator with norm or functional $\|\cdot\|$, error bound $\text{tol} > 0$, start time $t_0 = t_s$, end time t_e , first step size $0 < \Delta t_1 \leq t_e - t_s$, maximal increase and decrease factors $q_{\max} > 1$, $q_{\min} < 1$, certainty factor $\eta \in (0, 1)$, rejection factor $\phi \geq 1$, error per time step indicator bool_{EPS} , initial solution $\mathbf{W}^0$

```

1: if  $\text{bool}_{\text{EPS}} = 1$  then {error per time step}
2:    $k = p + 1$ ;
3: else {error per unit time step}
4:    $k = p$ ;
5: end if
6:  $j = 1$ ;
7:  $\mathbf{D} = \{t_0\}$ ;
8:  $\mathbf{W}^{\mathbf{D}}(t_0) = \mathbf{W}^0$ ;
9: while  $t < t_e$  do
10:  if  $\text{bool}_{\text{EPS}} = 1$  then
11:     $\text{tol}_{\text{final}} = \text{tol}$ 
12:  else
13:     $\text{tol}_{\text{final}} = \Delta t_j \text{tol}$ 
14:  end if
15:   $t = t_{j-1} + \Delta t_j$ ;
16:   $\mathbf{W} = \Psi^{t, t_{j-1}} \mathbf{W}^{\mathbf{D}}(t_{j-1})$ ;
17:   $\hat{\mathbf{W}} = \hat{\Psi}^{t, t_{j-1}} \mathbf{W}^{\mathbf{D}}(t_{j-1})$ ;
18:   $r_j = \text{compute\_error\_estimate}(\mathbf{W}, \hat{\mathbf{W}}, \text{tol}_{\text{final}})$ ;
19:  if  $r_j = 0$  then
20:     $\Delta t = q_{\max} \Delta t_j$ ;
21:  else
22:     $\Delta t = \text{compute\_new\_timestep\_size}(\eta, r_j, \Delta t_j, j, \dots)$ ;
23:     $\Delta t = \min(q_{\max} \Delta t_j, \Delta t)$ ;
24:     $\Delta t = \max(q_{\min} \Delta t_j, \Delta t)$ ;
25:  end if
26:  if  $r_j > \phi$  then {recompute step}
27:     $\Delta t_j = \min(\Delta t, t_e - t_j)$ ;
28:  else {accept step}
29:     $t_j = t$ ;
30:     $\mathbf{D} = \mathbf{D} \cup \{t_j\}$ ;
31:     $\mathbf{W}^{\mathbf{D}}(t_j) = \mathbf{y}$ ;
32:     $\Delta t_{j+1} = \min(\Delta t, t_e - t_j)$ ;
33:     $j = j + 1$ ;
34:  end if
35: end while

```

Algorithm 2 Standard method for computation of time step size

```

1: if  $j > 1$  then
2:    $\Delta t = \text{controller}(\eta, r_j, \Delta t_j, \dots)$ ;
3: else
4:    $\Delta t = \text{elementary\_control}(\eta, r_j, \Delta t_j, k)$ ;
5: end if
6: return  $\Delta t$ ;

```

by use of the iteration

$$\delta_j = \beta_P F(\xi_{j-1}) + \beta_I \sum_{k=0}^{j-1} F(\xi_k) + \beta_D \nabla F(\xi_{j-1}), \quad (5.28)$$

with $\nabla F(\xi_{j-1}) = F(\xi_{j-1}) - F(\xi_{j-2})$. The coefficients β_P , β_I , and β_D of the proportional, integral, and differential terms, respectively, have to be chosen appropriately. By taking the differences of two iterates

$$\xi_{j+1} - \xi_j = -(\delta_{j+1} - \delta_j) \quad (5.29)$$

$$= -\beta_P (F(\xi_j) - F(\xi_{j-1})) - \beta_I F(\xi_j) - \beta_D (\nabla F(\xi_j) - \nabla F(\xi_{j-1})), \quad (5.30)$$

the controller can be written in the form

$$\xi_{j+1} = \xi_j - \beta_I F(\xi_j) - \beta_P \nabla F(\xi_j) - \beta_D \nabla^2 F(\xi_j), \quad (5.31)$$

with $\nabla^2 F(\xi_j) = F(\xi_j) - 2F(\xi_{j-1}) + F(\xi_{j-2})$.

To use this controller for step size adaptation, use the model (5.17) as in the elementary controller. Again, the aim is to get the local error close to ηtol in case of error per time step control and $\Delta t \eta \text{tol}$ in case of error per unit time step control. Correspondingly, there is a step size Δt such that

$$\eta \text{tol} = c_j \Delta t^{p+1} \quad \text{or} \quad \Delta t \eta \text{tol} = c_j \Delta t^{p+1}. \quad (5.32)$$

Dividing (5.17) by (5.32) results in

$$\frac{r_j}{\eta} \approx \frac{c_j \Delta t_j^k}{c_j \Delta t^k} = \frac{\Delta t_j^k}{\Delta t^k}, \quad (5.33)$$

with r_j and k as in (5.23). A linear model can then be obtained by taking the logarithm,

which yields

$$\log\left(\frac{r_j}{\eta}\right) \approx k \log\left(\frac{\Delta t_j}{\Delta t}\right). \quad (5.34)$$

In order to apply the PID-controller, define $\xi_j := \log\left(\frac{\Delta t_j}{\Delta t}\right)$, resulting in

$$\log\left(\frac{r_j}{\eta}\right) \approx k\xi_j = F(\xi_j). \quad (5.35)$$

Now, the PID controller (5.31) can be used, yielding

$$\log\left(\frac{\Delta t_{j+1}}{\Delta t}\right) = \log\left(\frac{\Delta t_j}{\Delta t}\right) - \beta_I \log\left(\frac{r_j}{\eta}\right) - \beta_P \nabla \log\left(\frac{r_j}{\eta}\right) - \beta_D \nabla^2 \log\left(\frac{r_j}{\eta}\right), \quad (5.36)$$

and hence,

$$\Delta t_{j+1} = \left(\frac{\eta}{r_j}\right)^{\beta_I} \left(\frac{\eta}{r_j} \frac{r_{j-1}}{\eta}\right)^{\beta_P} \left(\frac{\eta}{r_j} \frac{2r_{j-1}}{\eta} \frac{\eta}{r_{j-2}}\right)^{\beta_D} \Delta t_j. \quad (5.37)$$

The PI controller, i. e. PID controller with $\beta_D = 0$, then reads

$$\Delta t = \left(\frac{\eta}{r_j}\right)^{\beta_I} \left(\frac{r_{j-1}}{r_j}\right)^{\beta_P} \Delta t_j \quad (5.38)$$

and $\Delta t_{j+1} = \Delta t$ is set in case of an accepted time step. Note that the elementary controller can be reconstructed by choosing $\beta_I = \frac{1}{k}$ and $\beta_P = 0$. It is therefore also called integral (I) control. For the PI controller, appropriate values for the *integral gain* β_I and the *proportional gain* β_P have to be chosen. In [34] values of $\beta_I = \frac{0.3}{k}$ and $\beta_P = \frac{0.4}{k}$ were suggested and tested successfully for explicit Runge-Kutta methods. These values are also chosen for the PI controller in this thesis. In [35], where the PI control for step size adaptation was introduced, the author also states that using PID control instead of PI control results in no significant differences. In the case of a rejected step as well as the step after, a special treatment was suggested in [34] in order to reduce the number of rejected steps. In case of a rejected step, the algorithm uses the elementary controller instead of the PI controller and in the step afterwards, the step size resulting from the PI controller gets multiplied by the ratio of the last step size to the step size before the rejection. The latter is done in anticipation of a necessary further step size reduction. In detail, the whole algorithm to compute the time step size (to be used in line 22 of Algorithm 1) is given by using Algorithm 3 with PI control (5.38) as controller in line 3.

Algorithm 3 Time step size computation using a special treatment after a rejection, due to [34]

```

1: if  $r_j < \phi$  then {accept step}
2:   if  $j > 1$  then
3:      $\Delta t = \text{controller}(\eta, r_j, \Delta t_j, \dots);$ 
4:   else
5:      $\Delta t = \text{elementary\_control}(\eta, r_j, \Delta t_j, k);$ 
6:   end if
7:   if previous step rejected then
8:      $\Delta t = \Delta t \frac{\Delta t_j}{h};$ 
9:   end if
10: else {reject step}
11:   if previous step not rejected then
12:      $h = \Delta t_j;$ 
13:   end if
14:    $\Delta t = \text{elementary\_control}(\eta, r_j, \Delta t_j, k);$ 
15: end if
16: return  $\Delta t$ 

```

Predictive controller

The predictive (P) controller, as described in [37, Section IV.8] and also used in [45], can be derived from the assumption that the change in the error constant c_j in the error model (5.17) is approximately constant, i. e.

$$\frac{c_{j+1}}{c_j} \approx \frac{c_j}{c_{j-1}} \quad (5.39)$$

inserting

$$c_j \approx \frac{\|[\epsilon_j^{\text{local}}]\|}{\Delta t_j^{p+1}} = \frac{r_j \text{tol}}{\Delta t_j^k} \quad (5.40)$$

yields

$$c_{j+1} \approx \frac{c_j}{c_{j-1}} c_j \approx \frac{r_j}{r_{j-1}} \left(\frac{\Delta t_{j-1}}{\Delta t_j} \right)^k \frac{r_j \text{tol}}{\Delta t_j^k}. \quad (5.41)$$

Aiming again at

$$\eta \text{tol} \approx c_{j+1} \Delta t_{j+1}^k, \quad \text{i. e.} \quad \Delta t_{j+1} \approx \left(\frac{\eta \text{tol}}{c_{j+1}} \right)^{\frac{1}{k}} \quad (5.42)$$

and inserting (5.41) gives the P controller

$$\Delta t = \left(\frac{\eta}{r_j} \right)^{\frac{1}{k}} \left(\frac{r_{j-1}}{r_j} \right)^{\frac{1}{k}} \frac{\Delta t_j}{\Delta t_{j-1}} \Delta t_j. \quad (5.43)$$

The controller is equal to the elementary controller multiplied by two additional fractions predicting the change in the error coefficients c_j . In this context, it is clear that the elementary control should be used in case of a step rejection (as done in Algorithms 3), since the step j is recomputed and no prediction for c_{j+1} is necessary. Note that the P controller is equivalent to the PI controller with $\beta_I = \beta_P = \frac{1}{k}$ multiplied by the fraction $\frac{\Delta t_{j-1}}{\Delta t_j}$.

Digital filter

Digital filters for adaptive step size control as introduced by Söderlind [60, 61] represent a whole family of step size adaptation algorithms. Based on the chosen parameters, algorithms with different properties such as higher order adaptivity or low pass filtering, reducing higher frequency components in the time step size, can be created. Among the many methods, the digital filter (DF) H211b (cf. Söderlind [61, Section 5.2]) given by

$$\Delta t = \left(\frac{\eta}{r_j} \right)^{\frac{1}{bk}} \left(\frac{\eta}{r_{j-1}} \right)^{\frac{1}{bk}} \left(\frac{\Delta t_j}{\Delta t_{j-1}} \right)^{\frac{-1}{b}} \Delta t_j \quad (5.44)$$

with the parameter $b = 4$, as recommended, has been chosen for testing in this thesis. This controller was also used in [7], where Rosenbrock time integration schemes are compared to ESDIRK schemes for a laminar flow around a circular cylinder. The ratio of error and tolerance r_j and the order k are again given as in (5.23) depending on the choice of error per time step or error per unit time step control. This controller shows rather smooth step sizes in the numerical test of [61].

5.1.3 New time step size limiter

The main applications of interest in this thesis are gust encounter simulations. In case of such an application, a rather large time step size can be taken before the actual gust encounter. Once the gust hits the structure, the adaptation algorithm significantly reduces the time step size. Due to this sudden, non-smooth decrease of the required time step size, several repeated step rejections can be observed when using the adaptation algorithms introduced in this chapter (cf. Section 7.3.2). While this is not a problem in principle, as the adaptation algorithms still work as desired, it is a problem of efficiency,

since every rejected step leads to a recomputation of this step. Therefore, it is desirable, to reduce the amount of step size rejections. In this section, a new limiter is introduced addressing this problem. The idea is to use the information from the first rejected step to avoid the subsequent rejections.

Start again with (5.17) as model for the local discretization error. Similar to the remainder of Taylor's theorem, this can be written in integral form as

$$\|[\epsilon_j^{\text{local}}]\| \approx c_j \Delta t_j^{p+1} = \int_0^{\Delta t_j} c_j(p+1)(\Delta t_j - x)^p dx. \quad (5.45)$$

Now suppose there is a rejected time step with step size Δt_{rej} and a corresponding error estimate

$$c_{\text{rej}} \Delta t_{\text{rej}}^{p+1} = \int_0^{\Delta t_{\text{rej}}} c_{\text{rej}}(p+1)(\Delta t_{\text{rej}} - x)^p dx \approx \|[\epsilon_{\text{rej}}^{\text{local}}]\| = g_{\text{rej}} \text{tol}, \quad (5.46)$$

with

$$g_{\text{rej}} := \frac{\|[\epsilon_{\text{rej}}^{\text{local}}]\|}{\text{tol}}. \quad (5.47)$$

Furthermore, suppose that after reducing the step size, a step of size $\Delta t_{\text{acc}} < \Delta t_{\text{rej}}$ was accepted with error estimate

$$c_{\text{acc}} \Delta t_{\text{acc}}^{p+1} = \int_0^{\Delta t_{\text{acc}}} c_{\text{acc}}(p+1)(\Delta t_{\text{acc}} - x)^p dx \approx \|[\epsilon_{\text{acc}}^{\text{local}}]\| = g_{\text{acc}} \text{tol}, \quad (5.48)$$

$$g_{\text{acc}} := \frac{\|[\epsilon_{\text{acc}}^{\text{local}}]\|}{\text{tol}}. \quad (5.49)$$

Then, the error constant for the accepted step is approximated by

$$c_{\text{acc}} \approx \frac{g_{\text{acc}} \text{tol}}{\Delta t_{\text{acc}}^{p+1}}. \quad (5.50)$$

With this information about the error constant, split the integral term of (5.46) in

$$\begin{aligned} \int_0^{\Delta t_{\text{rej}}} c_{\text{rej}}(p+1)(\Delta t_{\text{rej}} - x)^p dx &= \int_0^{\Delta t_{\text{acc}}} c_{\text{acc}}(p+1)(\Delta t_{\text{rej}} - x)^p dx \\ &\quad + \int_{\Delta t_{\text{acc}}}^{\Delta t_{\text{rej}}} c_{\text{rem}}(p+1)(\Delta t_{\text{rej}} - x)^p dx, \end{aligned} \quad (5.51)$$

with the error constant c_{rem} for the remaining part of the rejected step. The first integral

of the sum is then approximated by using (5.50)

$$\int_0^{\Delta t_{\text{acc}}} c_{\text{acc}}(p+1)(\Delta t_{\text{rej}} - x)^p dx = c_{\text{acc}}(\Delta t_{\text{rej}}^{p+1} - (\Delta t_{\text{rej}} - \Delta t_{\text{acc}})^{p+1}) \quad (5.52)$$

$$\approx \frac{g_{\text{acc}} \text{tol}}{\Delta t_{\text{acc}}^{p+1}} (\Delta t_{\text{rej}}^{p+1} - (\Delta t_{\text{rej}} - \Delta t_{\text{acc}})^{p+1}), \quad (5.53)$$

resulting in

$$\int_{\Delta t_{\text{acc}}}^{\Delta t_{\text{rej}}} c_{\text{rem}}(p+1)(\Delta t_{\text{rej}} - x)^p dx \approx g_{\text{rej}} \text{tol} - \frac{g_{\text{acc}} \text{tol}}{\Delta t_{\text{acc}}^{p+1}} (\Delta t_{\text{rej}}^{p+1} - (\Delta t_{\text{rej}} - \Delta t_{\text{acc}})^{p+1}). \quad (5.54)$$

$$= (g_{\text{rej}} - g_{\text{sf}}) \text{tol}, \quad (5.55)$$

with

$$g_{\text{sf}} = \frac{g_{\text{acc}}}{\Delta t_{\text{acc}}^{p+1}} (\Delta t_{\text{rej}}^{p+1} - (\Delta t_{\text{rej}} - \Delta t_{\text{acc}})^{p+1}). \quad (5.56)$$

On the other hand, it is

$$\int_{\Delta t_{\text{acc}}}^{\Delta t_{\text{rej}}} c_{\text{rem}}(p+1)(\Delta t_{\text{rej}} - x)^p dx = c_{\text{rem}}(\Delta t_{\text{rej}} - \Delta t_{\text{acc}})^{p+1}, \quad (5.57)$$

and hence, the error constant c_{rem} is approximated by

$$c_{\text{rem}} \approx (g_{\text{rej}} - g_{\text{sf}}) \frac{\text{tol}}{(\Delta t_{\text{rej}} - \Delta t_{\text{acc}})^{p+1}}. \quad (5.58)$$

Requiring

$$c_{\text{rem}} \Delta t_{\text{rem}}^{p+1} \stackrel{!}{\leq} \text{tol} \quad \text{or} \quad c_{\text{rem}} \Delta t_{\text{rem}}^{p+1} \stackrel{!}{\leq} \Delta t_{\text{rem}} \text{tol}, \quad (5.59)$$

for error per time step or error per unit time step control, respectively, a time step size limit

$$\Delta t_{\text{rem}} = \left(\frac{(\Delta t_{\text{rej}} - \Delta t_{\text{acc}})^{p+1}}{g_{\text{rej}} - g_{\text{sf}}} \right)^{\frac{1}{k}} \quad (5.60)$$

can be computed using the information from the rejected step and the step after. Similarly, if n steps have been accepted after the rejected step and

$$\Delta t_{\text{rej}} > \sum_{i=1}^n \Delta t_{\text{acc},i}, \quad (5.61)$$

equation (5.51) can be written as

$$\begin{aligned} \int_0^{\Delta t_{\text{rej}}} c_{\text{rej}}(p+1)(\Delta t_{\text{rej}} - x)^p dx &= \int_0^{\Delta t_{\text{acc},1}} c_{\text{acc},1}(p+1)(\Delta t_{\text{rej}} - x)^p dx \\ &+ \dots + \int_{\sum_{i=1}^{n-1} \Delta t_{\text{acc},i}}^{\sum_{i=1}^n \Delta t_{\text{acc},i}} c_{\text{acc},n}(p+1)(\Delta t_{\text{rej}} - x)^p dx \\ &+ \int_{\sum_{i=1}^n \Delta t_{\text{acc},i}}^{\Delta t_{\text{rej}}} c_{\text{rem}}(p+1)(\Delta t_{\text{rej}} - x)^p dx. \end{aligned} \quad (5.62)$$

Equation (5.56) then reads

$$\begin{aligned} g_{\text{sf}} &= \frac{g_{\text{acc},1}}{\Delta t_{\text{acc},1}^{p+1}} \left(\Delta t_{\text{rej}}^{p+1} - (\Delta t_{\text{rej}} - \Delta t_{\text{acc}})^{p+1} \right) + \dots \\ &+ \frac{g_{\text{acc},n}}{\Delta t_{\text{acc},n}^{p+1}} \left(\left(\Delta t_{\text{rej}} - \sum_{i=1}^{n-1} \Delta t_{\text{acc},i} \right)^{p+1} - \left(\Delta t_{\text{rej}} - \sum_{i=1}^n \Delta t_{\text{acc},i} \right)^{p+1} \right) \end{aligned} \quad (5.63)$$

and the time step size Δt_{rem} is given by

$$\Delta t_{\text{rem}} = \left(\frac{(\Delta t_{\text{rej}} - \sum_{i=1}^n \Delta t_{\text{acc},i})^{p+1}}{g_{\text{rej}} - g_{\text{sf}}} \right)^{\frac{1}{k}}. \quad (5.64)$$

The new limiter is given by using the minimum of the time step size calculated by the adaptation algorithm and Δt_{rem} as new time step size. Note that the limiter only has information till the end of the rejected step. Hence, if

$$\Delta t_{\text{rej}} < \sum_{i=1}^n \Delta t_{\text{acc},i} + \Delta t_{\text{rem}}, \quad (5.65)$$

the information is not available on the whole step Δt_{rem} . Therefore, we choose to apply the limiter only if information on more than half the next step is available, i. e. if

$$\Delta t_{\text{rej}} > \sum_{i=1}^n \Delta t_{\text{acc},i} + \frac{\Delta t_{\text{rem}}}{2}. \quad (5.66)$$

Since the model (5.17) is only an approximation, this limiter may lead to a stagnation, if $\Delta t_{\text{rej}} \approx \sum_{i=1}^n \Delta t_{\text{acc},i}$ but $g_{\text{rej}} \gg g_{\text{sf}}$. To avoid that, the limiter is relaxed when $\sum_{i=1}^n \Delta t_{\text{acc},i} > \sigma \Delta t_{\text{rej}}$. In detail, this is done by computing a new $g_{\text{rej}} = \gamma g_{\text{rej}}$. If not stated otherwise, it is $\sigma = 0.8$ and $\gamma = \frac{2}{3}$ in this thesis. Furthermore, the limiter is only applied if $g_{\text{rej}} > g_{\text{sf}}$. The algorithm to compute the time step size using this limiter is

described in Algorithm 4. Note that Algorithm 4 additionally switches to the elementary control when a step is rejected.

5.1.4 Summary of adaptation algorithms

To give an overview over the possible adaptation algorithms described in the previous sections, all possibilities for the computation of the time step size are listed below. Four different controllers have been introduced, namely the standard I controller (5.23), the PI controller (5.38) with $\beta_I = \frac{0.3}{k}$ and $\beta_P = \frac{0.4}{k}$, the P controller (5.43), and the DF (5.44) with $b = 4$. They can be combined with either Algorithm 2 (i.e. use the specified controller except for a rejected step, where the standard controller is used), Algorithm 3 (i.e. additionally anticipate a further necessary step size reduction by reducing the step size in the step after a rejection), or Algorithm 4 (i.e. use the introduced step size limiter for the steps after a step rejection) to compute the time step size in line 22 of Algorithm 1. Furthermore, either error per time step or error per unit time step can be chosen. This yields 24 different possibilities for the step size adaptation, which are compared in Section 7.3.

5.2 Error estimator

As mentioned in Section 5.1, the exact local discretization error can usually not be computed, since the exact evolution of the IVP is unknown. Hence, an estimate of the error is taken, which is usually done by taking the difference of two numerical solutions. Let Ψ and $\hat{\Psi}$ be two discrete evolutions with local discretization errors

$$\epsilon_j^{\text{local}} = \Psi^{t_j, t_{j-1}} \mathbf{W}^{j-1} - \Phi^{t_j, t_{j-1}} \mathbf{W}^{j-1}, \quad (5.67)$$

$$\hat{\epsilon}_j^{\text{local}} = \hat{\Psi}^{t_j, t_{j-1}} \mathbf{W}^{j-1} - \Phi^{t_j, t_{j-1}} \mathbf{W}^{j-1}. \quad (5.68)$$

Assume that $\hat{\Psi}$ is more accurate than Ψ and thus

$$\theta := \frac{\|\hat{\epsilon}_j^{\text{local}}\|}{\|\epsilon_j^{\text{local}}\|} < 1, \quad (5.69)$$

yields an error estimator

$$[\epsilon_j^{\text{local}}] := \Psi^{t_j, t_{j-1}} \mathbf{W}^{j-1} - \hat{\Psi}^{t_j, t_{j-1}} \mathbf{W}^{j-1} = \epsilon_j^{\text{local}} - \hat{\epsilon}_j^{\text{local}} \quad (5.70)$$

Algorithm 4 Time step size computation using the new limiter

Note: To avoid the application of the limiter before the first step rejection, $\Delta t_{\text{rej}} = 0$ and $\Delta t_{\text{sf}} \geq 0$ should be initialized in the first call.

```

1: if  $r_j > \phi$  then {recompute step}
2:    $\Delta t_{\text{rej}} = \Delta t_j$ ;
3:    $g_{\text{sf}} = 0$ ;
4:    $\Delta t_{\text{sf}} = 0$ ;
5:   if  $\text{bool}_{\text{EPS}} = 1$  then {error per time step}
6:      $g_{\text{rej}} = r_j$ ;
7:   else {error per unit time step}
8:      $g_{\text{rej}} = \Delta t_j r_j$ ;
9:   end if
10:   $\Delta t = \text{elementary\_control}(\eta, r_j, k, \Delta t_j)$ ;
11: else {accept step}
12:   if  $j > 1$  then
13:      $\Delta t = \text{controller}(\eta, r_j, \Delta t_j, \dots)$ ;
14:   else
15:      $\Delta t = \text{elementary\_control}(\eta, r_j, \Delta t_j, k)$ ;
16:   end if
17:   if  $\Delta t_{\text{sf}} + \Delta t_j < \Delta t_{\text{rej}}$  then {compute limiter}
18:     if  $\Delta t_{\text{sf}} + \Delta t_j > \sigma \Delta t_{\text{rej}}$  then
19:        $g_{\text{rej}} = \gamma g_{\text{rej}}$ 
20:     end if
21:     if  $\text{bool}_{\text{EPS}} = 1$  then
22:        $g_{\text{acc}} = r_j$ ;
23:     else
24:        $g_{\text{acc}} = \Delta t_j r_j$ ;
25:     end if
26:      $g_{\text{sf}} = g_{\text{sf}} + \frac{g_{\text{acc}}}{\Delta t_j^{p+1}} \left( (\Delta t_{\text{rej}} - \Delta t_{\text{sf}})^{p+1} - (\Delta t_{\text{rej}} - (\Delta t_{\text{sf}} + \Delta t_j))^{p+1} \right)$ ;
27:      $\Delta t_{\text{sf}} = \Delta t_{\text{sf}} + \Delta t_j$ ;
28:     if  $g_{\text{rej}} > g_{\text{sf}}$  then
29:        $\Delta t_{\text{rem}} = \left( \frac{(\Delta t_{\text{rej}} - \Delta t_{\text{sf}})^{p+1}}{g_{\text{rej}} - g_{\text{sf}}} \right)^{\frac{1}{k}}$ ;
30:       if  $\Delta t_{\text{rej}} > \Delta t_{\text{sf}} + \frac{\Delta t_{\text{rem}}}{2}$  then {apply limiter}
31:          $\Delta t = \min(\Delta t, \Delta t_{\text{rem}})$ ;
32:       end if
33:     end if
34:   else
35:      $\Delta t_{\text{sf}} = \Delta t_{\text{sf}} + \Delta t_j$ ;
36:   end if
37: end if
38: return  $\Delta t$ ;

```

with

$$\|[\epsilon_j^{\text{local}}] - \epsilon_j^{\text{local}}\| = \|\hat{\epsilon}_j^{\text{local}}\| = \theta \|\epsilon_j^{\text{local}}\|. \quad (5.71)$$

This then gives

$$\|[\epsilon_j^{\text{local}}]\| - \|\epsilon_j^{\text{local}}\| \leq \theta \|\epsilon_j^{\text{local}}\|, \quad (5.72)$$

$$-\|[\epsilon_j^{\text{local}}]\| + \|\epsilon_j^{\text{local}}\| \leq \theta \|\epsilon_j^{\text{local}}\|, \quad (5.73)$$

and thus

$$(1 - \theta) \|\epsilon_j^{\text{local}}\| \leq \|[\epsilon_j^{\text{local}}]\| \leq (1 + \theta) \|\epsilon_j^{\text{local}}\|. \quad (5.74)$$

Hence, the error estimator is reasonably accurate as long as $\theta \ll 1$ holds. In the case of an embedded DIRK method to solve (3.1), the error estimator (5.70) reads

$$[\epsilon_j^{\text{local}}] = \mathbf{W}^j - \hat{\mathbf{W}}^j. \quad (5.75)$$

In this thesis error estimators where $\hat{\Psi}$ has a higher order of accuracy than Ψ are used, which typically results in $\theta \ll 1$ for small enough time step sizes. Suppose Ψ is of order p and $\hat{\Psi}$ is of order $> p$. Then, $\epsilon_j^{\text{local}}$ is given by (5.17) and $\hat{\epsilon}_j^{\text{local}} = \mathcal{O}(\Delta t_j^{p+2})$. In this case, it is $\theta \rightarrow 0$ as $\Delta t_j \rightarrow 0$ if $c_j \neq 0$. This yields not only

$$\|[\epsilon_j^{\text{local}}]\| \xrightarrow{\Delta t_j \rightarrow 0} \|\epsilon_j^{\text{local}}\| \quad (5.76)$$

but also

$$\frac{\|[\epsilon_j^{\text{local}}]\|}{\|\epsilon_j^{\text{local}}\|} \xrightarrow{\Delta t_j \rightarrow 0} 1 \quad (5.77)$$

meaning that the error estimator is asymptotically exact as long as $c_j \neq 0$.

Up to now, only the error estimate for the less accurate evolution Ψ has been discussed. However, it might be preferable to take the more accurate evolution $\hat{\Psi}$ as result. The error of $\hat{\Psi}$ is bounded by

$$\|\hat{\epsilon}_j^{\text{local}}\| = \theta \|\epsilon_j^{\text{local}}\| \leq \frac{\theta}{1 - \theta} \|[\epsilon_j^{\text{local}}]\| \quad (5.78)$$

and for $\theta \leq \frac{1}{2}$ it is

$$\|\hat{\epsilon}_j^{\text{local}}\| \leq \|[\epsilon_j^{\text{local}}]\|. \quad (5.79)$$

Thus, even so the discretization is chosen based on the error of Ψ , it is also an appropriate upper bound for $\hat{\Psi}$ if $\theta \leq \frac{1}{2}$.

A very important aspect in the choice of a suitable error estimator is not only the ac-

curacy of the estimation, but also the required additional computational cost. The naive way of computing the required two different numerical solutions completely independent might be a very inefficient method. A much more efficient way is to use embedded Runge-Kutta methods as described in Chapter 3.1

5.3 Error norm or functional

As described in Section 5.2, the error estimator for the step size adaptation is based on the difference of two flow solutions. In order to get a positive scalar value, as required by the adaptation algorithm, some norm or functional has to be chosen. Some standard choices are the discrete L_2 -norm, the maximum (L_∞ -) norm, or the 2-norm (Euclidean norm). However, it does not have to be a norm. In fact, the absolute of any functional could be chosen in theory. Of course, this functional needs to be a suitable choice for the problem at hand. In CFD the interest often lies more in the integral forces like lift or drag. Therefore, it seems appropriate to choose an error functional in terms of these quantities. In this thesis, the absolute value of the difference in the lift coefficient is used as functional for the error estimate, but other aerodynamic coefficients could be chosen just as well. Note that the 2-norm is very mesh sensitive, finer meshes leading to higher error values. Therefore, the 2-norm is scaled by the number of grid points N here to avoid this effect. The different error norms or functionals investigated in this thesis are detailed below. Any of these functionals could be used to compute the error estimate in line 18 of Algorithm 1. The error is divided by the tolerance

$$\text{tol}_{\text{final}} = \text{tol} \quad \text{or} \quad \text{tol}_{\text{final}} = \Delta t_j \text{tol} \quad (5.80)$$

for error per time step or error per unit time step control respectively. A mixed absolute and relative tolerance with fixed scaling (cf. [63, Section 4.3]) is used. On one hand, this avoids the problem of dividing by zero or very small values, as it occurs when using sheer relative tolerance. On the other hand, it avoids the dominating effects of variables that have a large scale over others, which can be observed when using the absolute tolerance. Given two solution vectors $\mathbf{W}$ and $\hat{\mathbf{W}}$ in conservative variables resulting from the Runge-Kutta method and its embedded method, the scaled difference

$$\delta = (\mathbf{W} - \hat{\mathbf{W}}) ./ ((\mathbf{W} + \mathbf{1}) \text{tol}_{\text{final}}) \in \mathbb{R}^{Nd} \quad (5.81)$$

is determined by pointwise division of the error by the sum of absolute and relative tolerance. The error estimate can then be computed using one of the following norms,

where $\mathcal{V}$ is the Volume matrix as defined in (2.91).

Discrete L_2 -norm

$$\|\boldsymbol{\delta}\|_{L_2} = \sqrt{\boldsymbol{\delta}^T \mathcal{V} \boldsymbol{\delta}} \quad (5.82)$$

Scaled 2-norm

$$\|\boldsymbol{\delta}\|_2 = \frac{1}{N} \sqrt{\boldsymbol{\delta}^T \boldsymbol{\delta}} \quad (5.83)$$

L_∞ -norm

$$\|\boldsymbol{\delta}\|_\infty = \max_{i=1,\dots,N_d} (\delta_i) \quad (5.84)$$

We then have $r_j = \|\boldsymbol{\delta}\|$, with $\|\cdot\|$ one of the norms above as error estimate in Algorithm 1.

Alternatively, the error can be computed by the absolute value of the difference of the lift coefficients

$$\epsilon^{\text{local}} = |\text{c}_{\text{lift}}(\mathbf{W}) - \text{c}_{\text{lift}}(\hat{\mathbf{W}})| \quad (5.85)$$

giving

$$r_j = \frac{\epsilon^{\text{local}}}{\text{tol}_{\text{final}}} = \frac{|\text{c}_{\text{lift}}(\mathbf{W}) - \text{c}_{\text{lift}}(\hat{\mathbf{W}})|}{\text{tol}_{\text{final}}}. \quad (5.86)$$

To compute the lift coefficient, first sum up all (inviscid) pressure forces and viscous forces of the boundary faces $bi \in \partial D_W$ which are part of a (inviscid or viscous) wall boundary. This reads

$$\mathbf{S}(\mathbf{W}) = \sum_{bi \in \partial D_W} p_i \tilde{\mathbf{n}}_{bi} + \mathcal{T}_i \tilde{\mathbf{n}}_{bi} \in \mathbb{R}^3. \quad (5.87)$$

where p_i and $\mathcal{T}_i$ are the pressure and the viscous stress tensor evaluated at the (boundary) point i and $\tilde{\mathbf{n}}_{bi}$ is the normal vector of the boundary face with length equal to the face area as defined in (2.31). The pressure forces $\mathbf{S}(\mathbf{W})$ consist of the drag $\mathbf{S}_D(\mathbf{W})$ and the Lift $\mathbf{S}_L(\mathbf{W})$:

$$\mathbf{S}(\mathbf{W}) = \mathbf{S}_D(\mathbf{W}) + \mathbf{S}_L(\mathbf{W}) \quad (5.88)$$

The drag is the component of the force parallel to the flow direction and the lift is the component of the force that is perpendicular to the flow direction. The lift coefficient is then computed as

$$\text{c}_{\text{lift}}(\mathbf{W}) = \frac{2\sqrt{\mathbf{S}_L(\mathbf{W})^T \mathbf{S}_L(\mathbf{W})}}{u_{\text{ref}}^2 \rho_{\text{ref}} A_{\text{ref}}}, \quad (5.89)$$

with reference velocity u_{ref} , reference density ρ_{ref} , and reference area A_{ref} .

CHAPTER 6

Numerical test cases

In this chapter, the numerical test cases simulated for this thesis are introduced. the corresponding results can be found in Chapter 7. Note that the TAU code used in this thesis always runs in 3-D. Hence, the 2-D test cases have an additional third dimension consisting of one layer of elements. The grids then have twice the number of grid points of the corresponding 2-D grids. The number of grid points in the primary grid is equal to the number of elements in the dual grid which multiplied by the number of equations gives the degrees of freedom of the test case in the TAU code. In the following, inviscid, laminar, and turbulent test cases are presented, which correspond to solving the Euler, Navier-Stokes, and RANS equations as described in Chapter 2. For all test cases only the aerodynamics and no fluid structure interaction is considered.

6.1 Ehrenfried vortex

To verify the order of the implemented Runge-Kutta methods for fluid dynamic applications a convection of an inviscid vortex is simulated. In particular, a vortex proposed by Ehrenfried [27] which is continuous in all conservative variables is chosen. The vortex has a density distribution prescribed as

$$\frac{\rho(r)}{\rho_\infty} = \begin{cases} 1 - \frac{\Delta\rho_v}{2\rho_\infty} \left[1 + \sin \left(\pi \left(1 - \frac{r}{R_0} \right)^3 - \frac{\pi}{2} \right) \right] & \text{for } 0 \leq r \leq R_0 \\ 1 & \text{for } R_0 < r \end{cases}, \quad (6.1)$$

where $\Delta\rho_v$ is a given density reduction in the center of the vortex compared to the density ρ_∞ outside the finite vortex, R_0 is the radius of the vortex and r the distance to the center of the vortex. As described in [46], the pressure distribution can be derived

from

$$\frac{p}{p_\infty} = \left(\frac{\rho}{\rho_\infty} \right)^\gamma, \quad (6.2)$$

which is valid assuming isentropic flow of an ideal gas. The momentum equation for radial flow

$$\frac{dp}{dr} = \frac{\rho v_\theta^2}{r} \quad (6.3)$$

then yields the circumferential velocity component. The vortex is symmetric with a zero radial velocity $v_r = 0$, such that it does not change over time when chosen as initial condition for the Euler equations (2.4). To get the convection of the vortex, a constant velocity of Mach 0.5 from left to right is added to the initial flow field. The simulation is done on a rectangular structured mesh with $2 \times 85 \times 101$ grid points, periodic boundary conditions left and right as well as farfield boundary conditions on the top and bottom. The usage of periodic boundary conditions in both directions is currently not possible in TAU. The grid and initial density distribution of this test case are shown in Figure 6.1. The extension in z-direction is large enough to avoid influences from the farfield boundary.

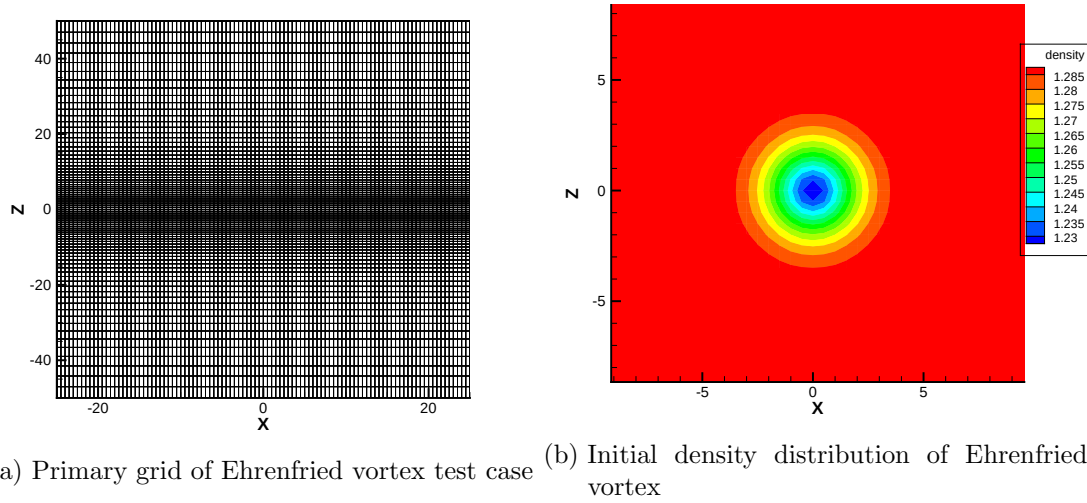
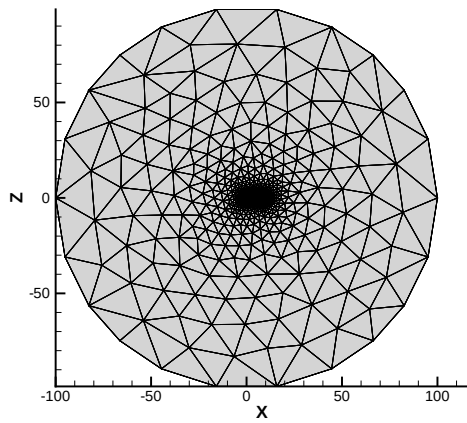


Figure 6.1: Ehrenfried vortex test case: Grid and initialization

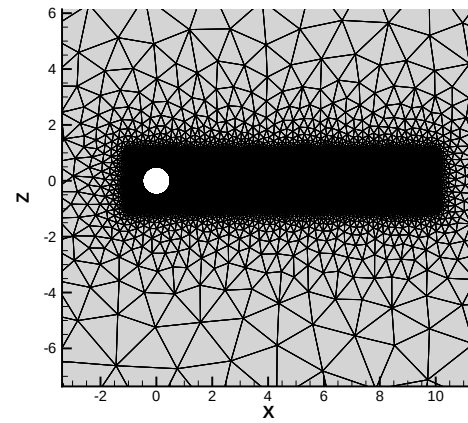
6.2 Kármán vortex street

As an inherently unsteady viscous test case, a laminar 2-D flow around a circular cylinder resulting in a Kármán vortex street behind the cylinder is chosen. The simulation is

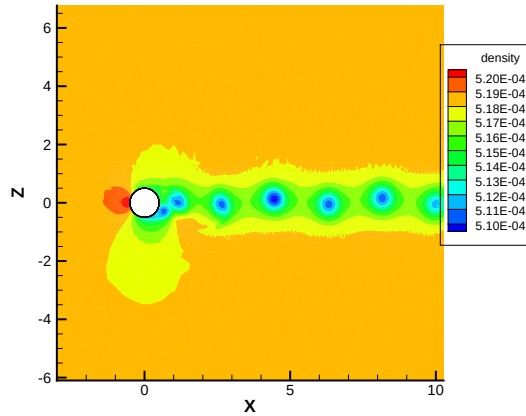
performed at a Mach number of 0.1 and a Reynolds number of 1000. An unstructured mesh with 26,090 grid points as shown in Figure 6.2a and 6.2b is used. The mesh is refined around the cylinder as well as in its wake. Figure 6.2c shows a snapshot of the density distribution showing the vortex shedding behind the cylinder. This test case is used to investigate the order of the implemented methods in Section 7.1



(a) Whole primary grid



(b) Part of primary grid



(c) Density distribution snapshot

Figure 6.2: Primary grid and density distribution snapshot of Kármán vortex street test case

6.3 NACA0012 laminar steady

A laminar flow over a NACA0012 airfoil at Mach 0.5, a Reynolds number of $5 \cdot 10^3$, and an angle of attack of 1° is considered. The test case is equal to the Case 2 in [64] with a coarser grid of size $2 \times 257 \times 129$ points. This steady test case is used in Section 7.3.1 to investigate the behavior of a time adaptive simulation for a steady state solution. The test case presented here is especially interesting, since as noted in [64], there have been publications that suggest that the test case has an inherently unsteady solution. Figure 6.3 shows the grid as well as the final density and Mach number distributions. The added streamlines clearly show the trailing edge separation occurring for this test case.

6.4 NACA0012 turbulent pitching oscillation

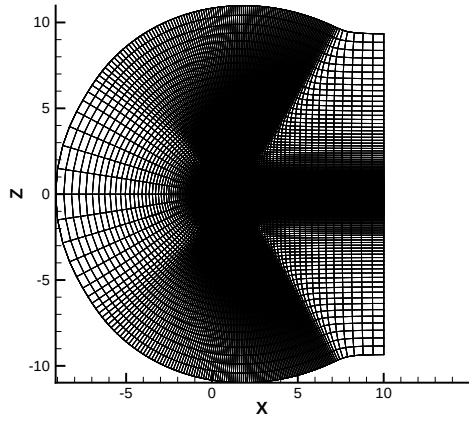
The turbulent flow around the 2-D NACA0012 airfoil subject to a pitching oscillation is considered. The forced oscillation around the $\frac{1}{4}$ -chord point of the airfoil can be described by the angle of attack given by

$$\alpha(t) = \alpha_0 + \sum_{i=1}^{N_F} \alpha_i \sin(ikt). \quad (6.4)$$

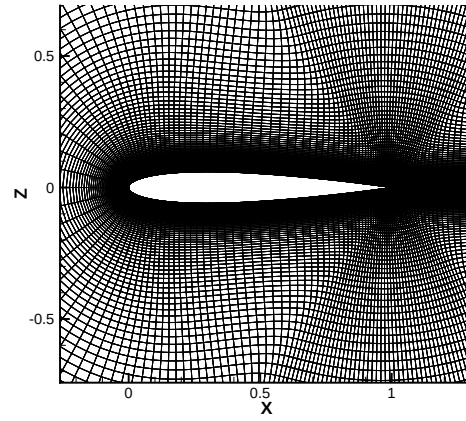
Here, a reduced frequency $k = 0.1628$, $N_F = 7$, and the coefficients $\alpha_0 = 10^\circ$, $\alpha_1 = 2.51^\circ$, $\alpha_7 = 1^\circ$, and $\alpha_i = 0^\circ$, $i \in \{2, \dots, 6\}$ are used. The simulation is performed on a structured grid of size $2 \times 449 \times 129$ taken from the NASA Turbulence Modeling Resource website [43]. For the grid refinement study in Section 7.2, the next coarser and finer grid of the FAMILY II grids with $2 \times 225 \times 65$ and $2 \times 897 \times 257$ grid points, respectively, are also considered. As on the website, the free-stream Mach number 0.15 and Reynolds number $6 \cdot 10^6$ are chosen.

As starting point for the simulation results shown in this thesis, the flow solution after one initial period is taken. This means, a steady computation at an angle of attack of 10° is done first followed by an unsteady computation of one period of the lower frequency oscillation. Using the result as starting value, the simulation of another period is then investigated. There is one exception, namely in Chapter 7.5, where appropriate DIRK methods to compute the initial transient starting from the steady solution are discussed.

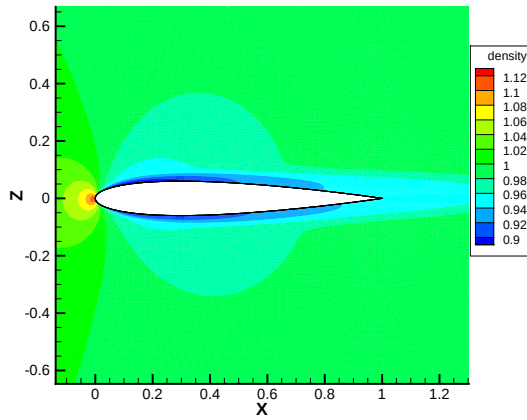
Figure 6.4 shows the medium grid, the initial density distribution, and the evolution of the lift coefficient for this test case.



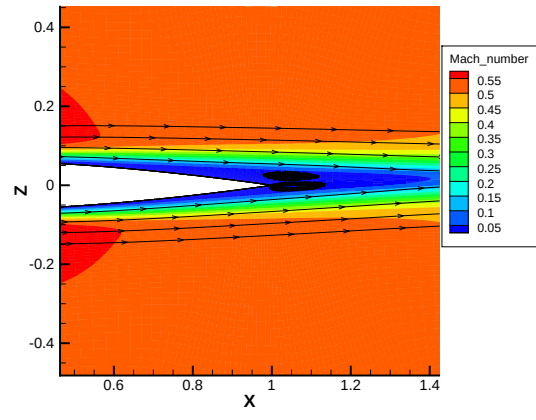
(a) Whole primary grid



(b) Part of primary grid

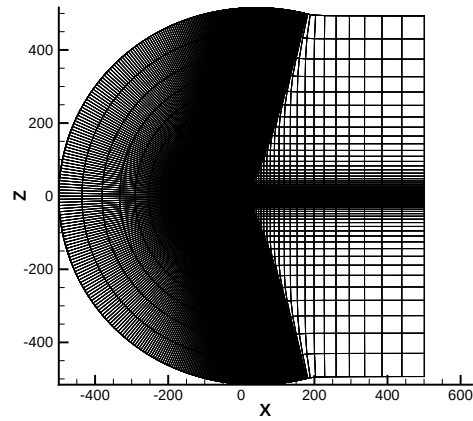


(c) Final density distribution

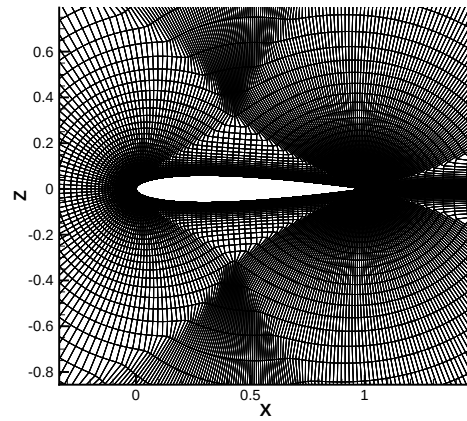


(d) Mach number distribution and streamlines

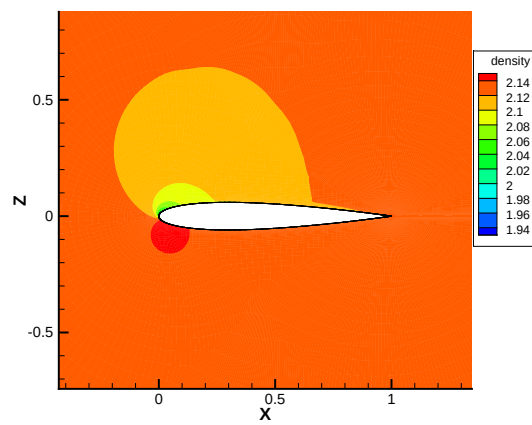
Figure 6.3: Primary grid, initial density distribution, Mach number, and streamtraces for the laminar NACA0012 test case



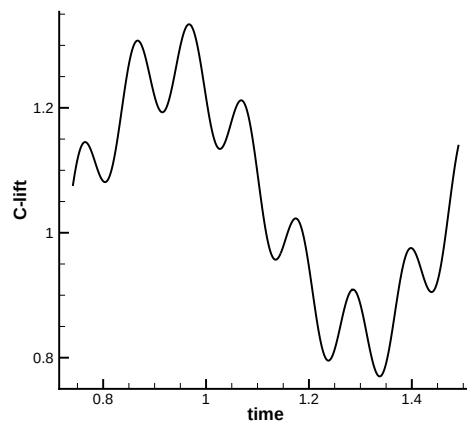
(a) Whole primary grid



(b) Part of primary grid



(c) Initial density distribution



(d) Evolution of lift coefficient

Figure 6.4: Primary grid, initial density distribution, and evolution of lift coefficient for the NACA0012 pitching oscillation test case

6.5 Gust encounter

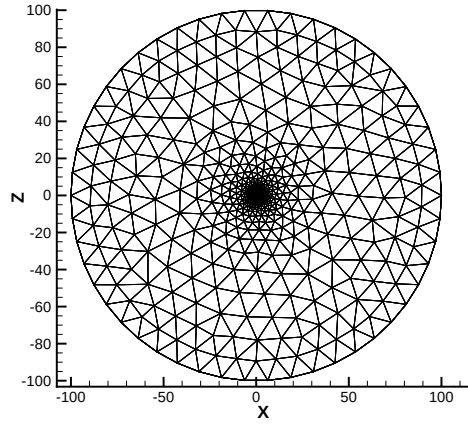
To demonstrate the behavior of the implemented time step size adaptation algorithm for fluid dynamic applications, gust encounters of different airplane configurations are simulated. For these test cases the time step size is expected to vary a lot depending on whether the gust currently interacts with the plane or not. Hence, an advantage in computational time when using an adaptive time step size compared to a constant one can be expected. To reduce the computational cost of the simulations, the disturbance velocity approach as described in Section 2.5 and in [40] is used to model the gust.

6.5.1 NACA0012 inviscid

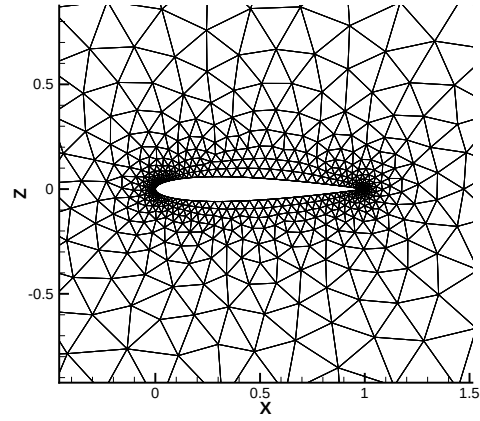
In this test case, the inviscid flow around a 2-D NACA0012 airfoil encountering a 1-cos shaped upwards gust is simulated. The gust is of length 2m and has an amplitude of $10 \frac{\text{m}}{\text{s}}$. It starts 4m in front of the airfoil which moves at Mach 0.2880277995. This equals to a velocity of $v = 100 \frac{\text{m}}{\text{s}}$ for a temperature of $T = 300\text{K}$, a specific gas constant of $R = 287 \frac{\text{m}^2}{\text{s}^2\text{K}}$, and a ration of specific heats of $\gamma = 1.4$. Hence, the gust reaches the beginning of the airfoil at $t = 0.04\text{s}$ and, since the airfoil has a length of 1m, leaves it at $t = 0.07\text{s}$. The considered angle of attack is 0° and the end time of the simulation is 0.15s. The unstructured grid with 2280 grid points is shown in Figure 6.5 together with the initial density distribution (computed with a steady simulation) and the evolution of the lift coefficient.

6.5.2 SDM fighter configuration inviscid

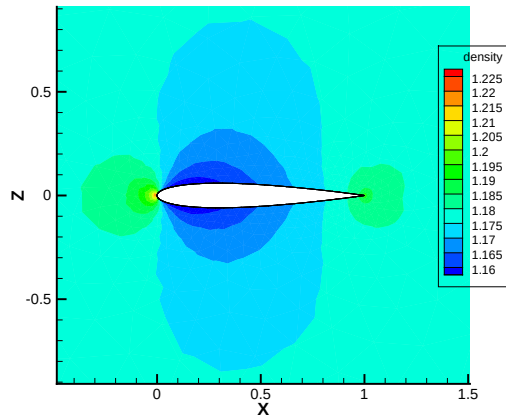
An encounter of the 3-D standard dynamic model (SDM) generic fighter configuration [66] with a vertical 1-cos gust is considered. The gust has a length of 30m, an amplitude of $10 \frac{\text{m}}{\text{s}}$ and starts 10.17m in front of the airplane. The SDM model has a length of about 15m. The inviscid simulation on an unstructured grid with 59,542 points is done at Mach 0.5 with an angle of attack of 2° . With a reference density of $1.2925 \frac{\text{kg}}{\text{m}^3}$, reference pressure of $101325 \frac{\text{N}}{\text{m}^2}$, and a ratio of specific heats of 1.4, this relates to a velocity of approximatively $165.6 \frac{\text{m}}{\text{s}}$. The simulation starts from a converged steady state solution and ends at $t_e = 0.5\text{s}$. Figure 6.6 shows parts of the grid and initial density distribution as well as the evolution of the lift coefficient. Note that the increase of the lift coefficient starting at about $t = 0.11$ relates to the gust hitting the airfoil. Figure 6.6 shows the grid, initial density distribution and the evolution of the lift coefficient for this test case.



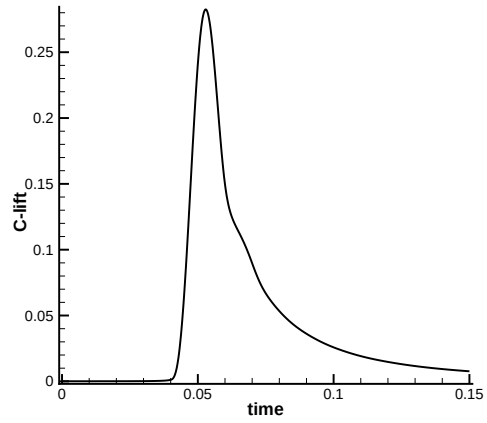
(a) Whole primary grid



(b) Part of primary grid

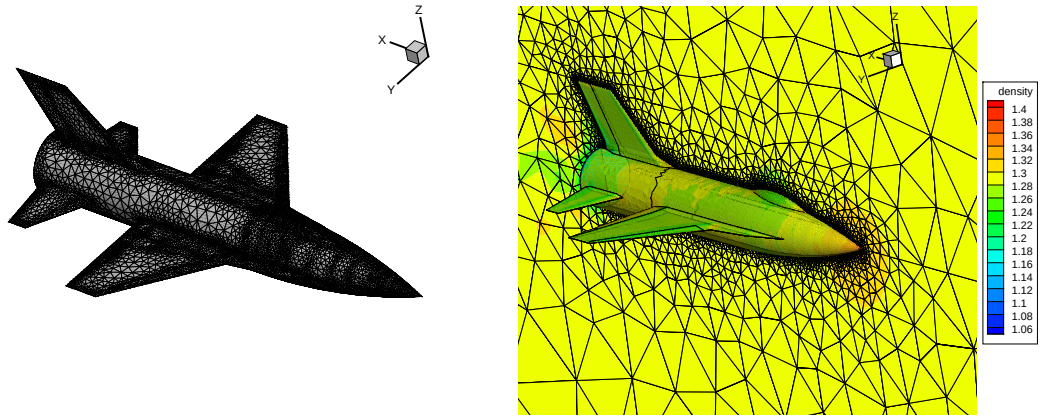


(c) Initial density distribution



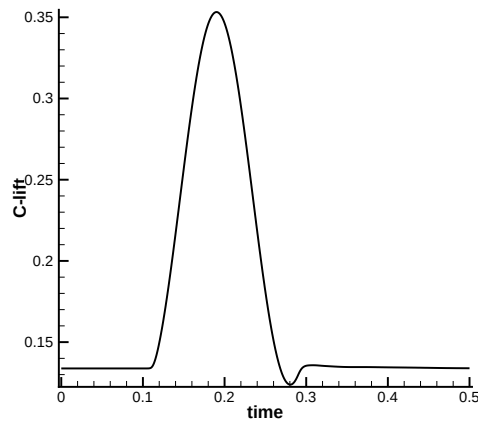
(d) Evolution of lift coefficient

Figure 6.5: Primary grid, initial density distribution, and evolution of lift coefficient for the NACA0012 gust encounter test case



(a) Primary grid on the surface

(b) Initial density distribution on surface and y=0 plane and grid on y=0 plane

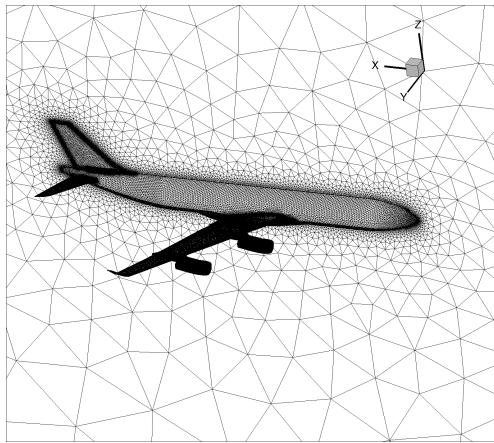


(c) Evolution of lift coefficient

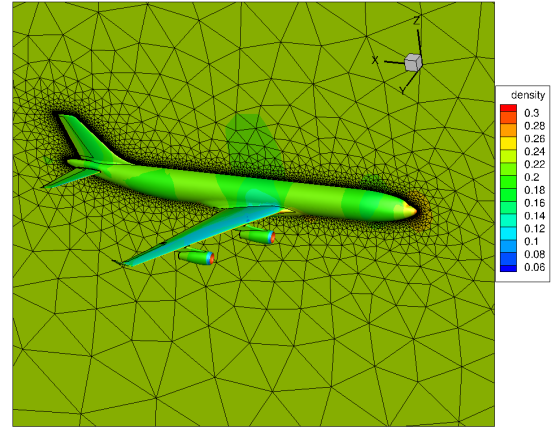
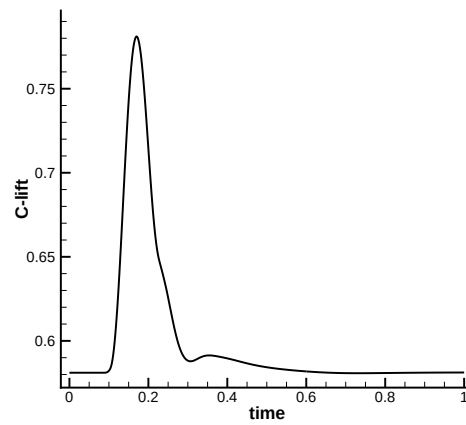
Figure 6.6: Primary grid, initial density distribution, and evolution of lift coefficient for the SDM gust encounter test case

6.5.3 Transport aircraft half model turbulent

The most complex case presented in this thesis is a gust encounter of a transport aircraft half model. The vertical $1 - \cos$ gust considered has a length of 30m, an amplitude of $10 \frac{m}{s}$ and starts 20m in front of the airplane. The turbulent transonic simulation on an unstructured grid with 6,380,749 points (as used in [41]) is computed at Mach 0.8208 and a Reynolds number of $26.3 \cdot 10^6$ with an angle of attack of 2.5793° . With a reference density of $0.2283 \frac{kg}{m^3}$, reference pressure of $17899 \frac{N}{m^2}$, and a ratio of specific heats of 1.4, this relates to a velocity of approximately $271.9 \frac{m}{s}$. The simulation starts from a steady state solution which is converged to a density residual of $1 \cdot 10^{-8}$ compared to the freestream residual. The total physical simulation time is $t_e = 1.0s$. Figure 6.7 shows the grid, initial density distribution and the evolution of the lift coefficient for this test case.



(a) Primary grid on the surface

(b) Initial density distribution on surface and $y=0$ plane and grid on $y=0$ plane

(c) Evolution of lift coefficient

Figure 6.7: Primary grid, initial density distribution, and evolution of lift coefficient for the transport aircraft gust encounter test case

CHAPTER 7

Numerical results

In this section, the methods and algorithms introduced in Chapters 3, 4, and 5 are applied to the numerical test cases introduced in Chapter 6 and investigated. We start by investigating the temporal order of convergence of the various Runge-Kutta and BDF methods described in Section 3.5. In Section 7.1 this is done for an inviscid as well as a viscous test case, namely the Ehrenfried vortex test case as described in Section 6.1 and the Kármán vortex street as described in Section 6.2.

Afterwards, the influence of grid refinement on explicit and implicit (A-stable) Runge-Kutta methods is investigated in Section 7.2. To show the limitations of explicit methods for stiff differential equations, the turbulent NACA0012 pitching oscillation test case (cf. Section 6.4) is used here with a sequence of three grids.

In Section 7.3, the different time step size adaptation algorithms introduced in Chapter 5 are investigated. As a first test, it is verified, that the adaptation algorithm chooses arbitrarily large time step sizes for a steady state simulation once the steady state is reached. For this, the laminar NACA0012 test case described in Section 6.3 is used. This case is chosen here for testing since according to [64] there have been publications suggesting an inherently unsteady behavior for the case. By applying the unsteady solution algorithm here, the results in [64] suggesting a steady state solution are confirmed. In Section 7.3.2 the different controllers and adaptation algorithms, including the new limiter (cf. Section 5.1.3) are compared. The inherently unsteady test cases, i.e. the Ehrenfried vortex and the Kármán vortex street are not very interesting in terms of time step size adaptation, since the chosen step sizes do not vary much. Therefore, the focus in this section is on the gust encounter simulations. Due to the many available settings of the adaptation algorithms, the inviscid NACA0012 gust encounter test case (Section 6.5.1) is mainly used here. It has a reasonably small computational cost and still shows the main effects regarding the step size adaptation for gust encounter sim-

ulations. A few further tests are done for the gust encounter of the standard dynamic model fighter configuration as described in Section 6.5.2 to check the previous results for a different test case. As a comparison, the NACA0012 pitching oscillation test case (Section 6.4) is also investigated, showing that the choice of the adaptation algorithm has almost no influence for such a test case since the time step size is almost constant. In the final part of this section, various norms and functionals for error estimation are compared. Here, the inviscid NACA0012 gust encounter (Section 6.5.1) and the turbulent NACA0012 pitching oscillation (Section 6.4) test cases are considered again.

In Section 7.4, methods for the solution of the systems of nonlinear equations as described in Chapter 4 are investigated. The systems of equations resulting from turbulent test cases are typically much harder to solve than those from laminar and inviscid cases. Therefore the focus in this section is on the turbulent cases, namely the NACA0012 pitching oscillation and the 3D transonic transport aircraft gust encounter case as described in Sections 6.4 and 6.5.3, respectively. For these cases, the choice of the solution method (cf. Section 4.1) as well as the different stage value predictors as described in Section 4.2 are investigated.

Afterwards, some properties of the different DIRK methods considered in this thesis are investigated in Section 7.5 and special problems are described that might occur for aerodynamic simulations depending on the choice of the method.

Finally, a comparison in terms of efficiency of different DIRK and BDF methods is described in Section 7.6. For the DIRK methods, constant and adaptive time steps are used, while the BDFs only use constant time steps. Here, the NACA0012 gust encounter test case (Section 6.5.1) as well as both turbulent test cases, the NACA0012 pitching oscillation (Section 6.4) and the transport aircraft gust encounter (Section 6.5.3), are investigated.

7.1 Temporal order of convergence of implemented methods

To analyse the temporal order of convergence of the various implemented Runge-Kutta methods the convection of the Ehrenfried vortex (see Section 6.1) and, as a viscous test case, the vortex shedding behind a circular cylinder (Kármán vortex street) as described in Section 6.2 have been simulated.

Ehrenfried vortex

For the convergence analysis, the convection of the Ehrenfried vortex has been simulated until time $t = 0.02$ and the difference of the final flow solution with different constant

time step sizes to a reference solution with a small time step size of $1 \cdot 10^{-6}$ has been computed. The scalar value of the global error then has been determined by taking the 2-norm (5.83) over the density error values normalized by the 2-norm of the density values of the reference solution.

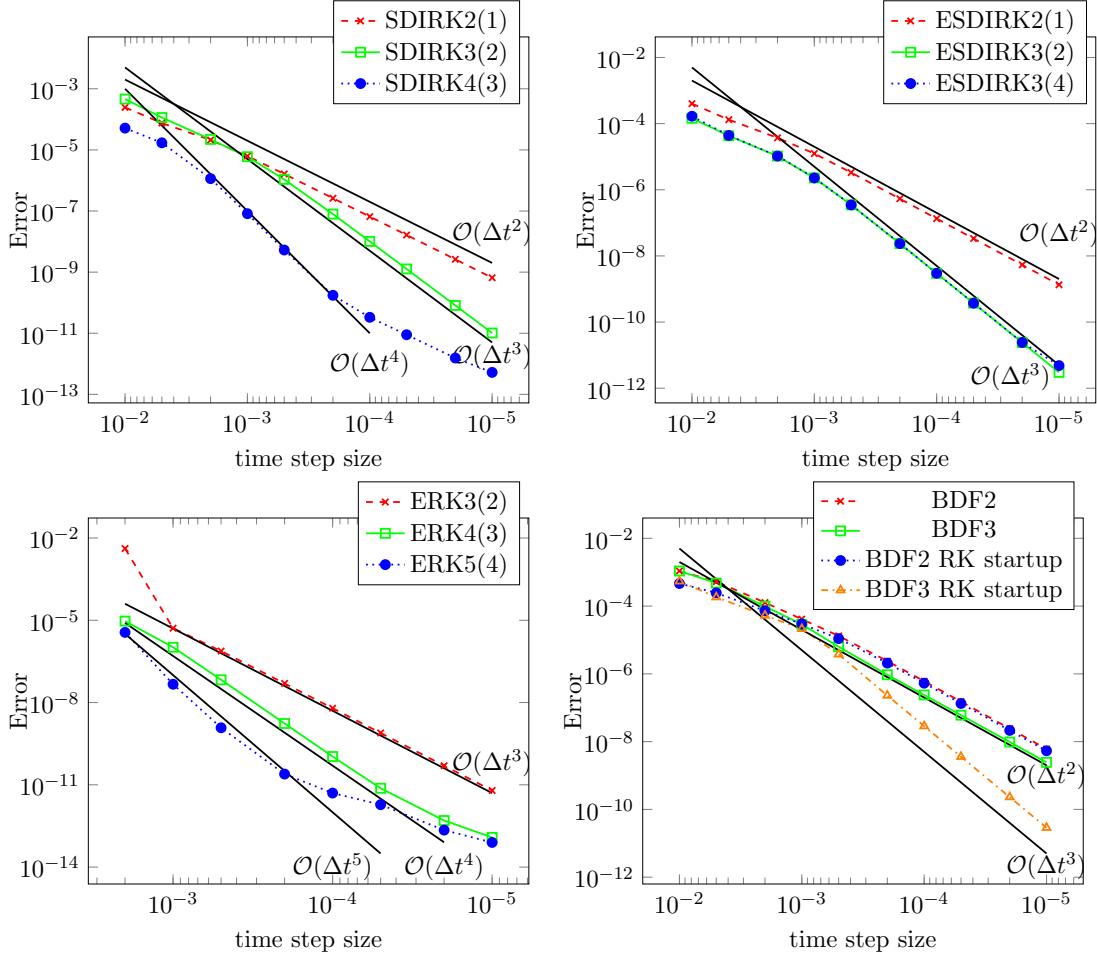


Figure 7.1: Convergence study of the implemented Runge-Kutta methods and BDFs for the Ehrenfried test case

Figure 7.1 shows the convergence behavior of the implemented explicit and implicit methods, respectively. The theoretical order of convergence for methods up to order 3 can be observed. For the higher order methods a reduction of convergence order for very small time step sizes occurs. The specific reasons for this order reduction are not clear. The known order reduction for Runge-Kutta methods mentioned in Section 3.2.3 can be observed for large time step sizes and should not be the reason for the reduced order at small step sizes. Furthermore, the order reduction also does not seem to be due to the

boundaries, since the error is located near the vortex and far away from the boundaries, as shown in Figure 7.2. Potential influences of accumulated rounding errors have not been investigated in detail and might be responsible for the reduced order for small time step sizes.

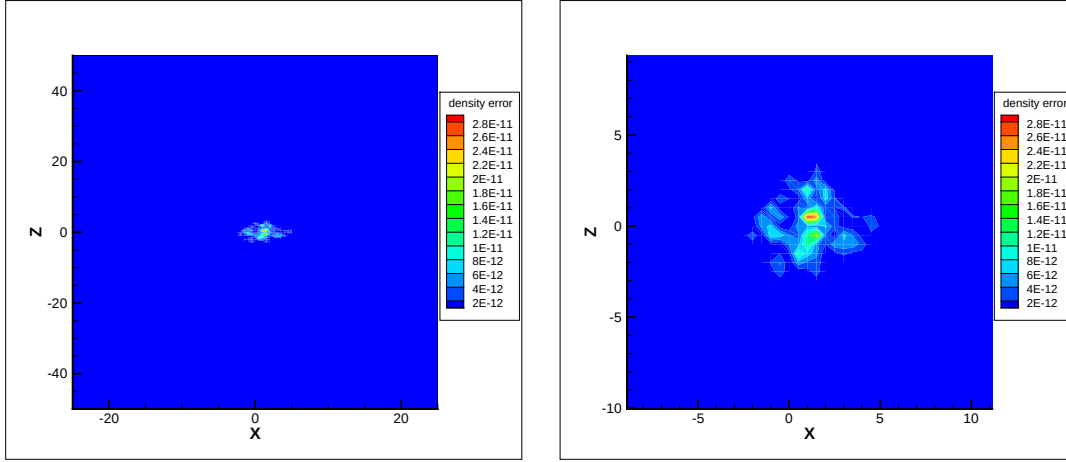


Figure 7.2: Temporal density error distribution for the Ehrenfried vortex test case using SDIRK4(3) with a step size of $1 \cdot 10^{-5}$, whole flow field and zoom at the vortex.

Note that in case of the BDFs, the convergence study has been done twice using either a lower order BDF or a Runge-Kutta method as startup for the first step(s) of the simulation. The strategy of using BDF1 for the first step and BDF2 for the second step, which is the current standard in TAU, results in a 2nd order convergence only for BDF3. This is due to the first order consistency error in the first step. Taking a Runge-Kutta method of at least order 2 as startup in the first step leads to the theoretical order 3 of BDF3 and the reduction in order is removed.

Kármán vortex street

As a viscous test, the temporal order of convergence for a laminar 2-D flow around a circular cylinder resulting in a Kármán vortex street behind the cylinder as described in Section 6.2 has been investigated. Due to the stiffness of the problem with small cells near the cylinder only implicit methods are considered. The explicit methods either diverge or, in the case of very small time step sizes, show errors in the range of the rounding errors. As in the case of the Ehrenfried vortex above, the simulation time is again 0.02s. The computation of the error is also done in the same way. The initial solution is computed by doing a few steps of a steady simulation first, followed by an

unsteady simulation of 1s to get the initial vortex street as shown in Section 6.2. Similar to previous results from the Ehrenfried vortex the expected order of convergence for implicit methods up to order three can be observed. The fourth order method again does not show its theoretical convergence order for small time steps (cf. Figure 7.3). Once again, using a first order BDF as startup for BDF3 results in an order reduction.

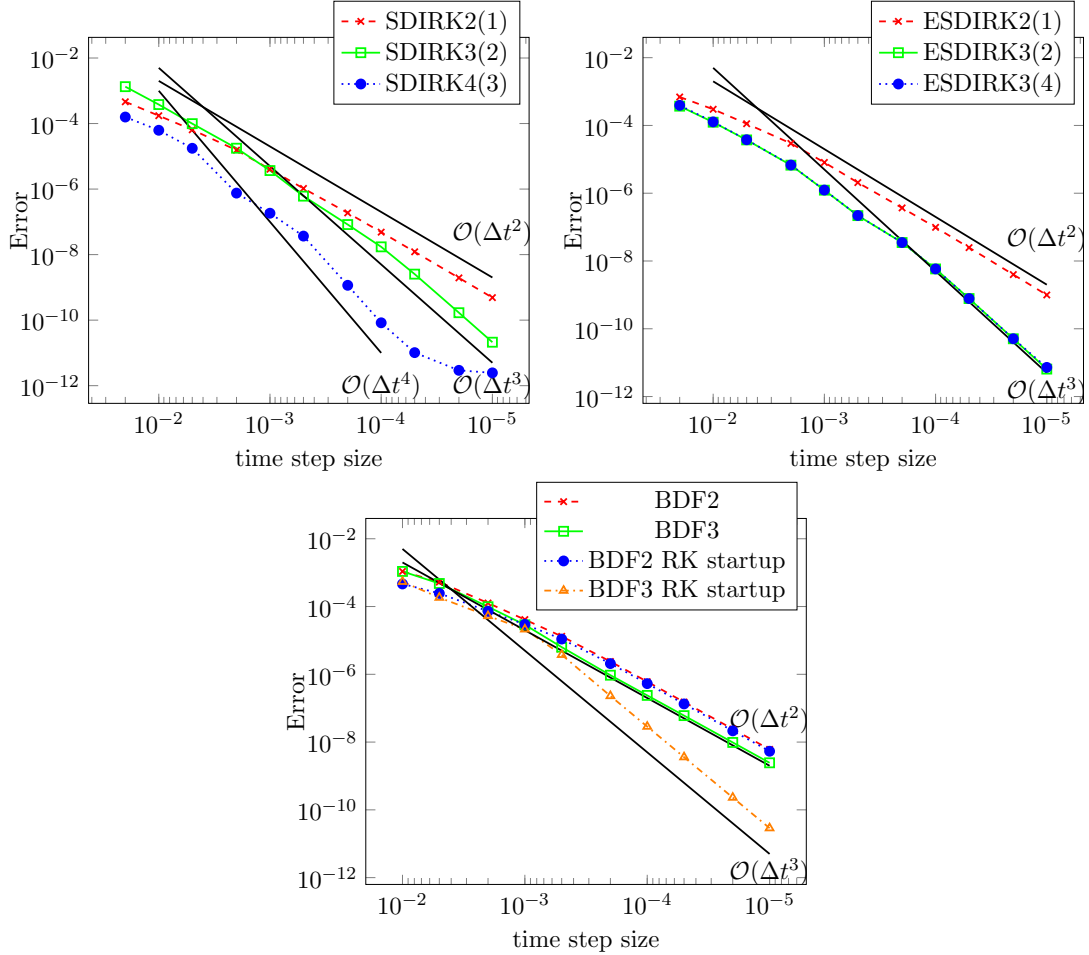


Figure 7.3: Convergence study of the implemented Runge-Kutta methods and BDFs for the Kármán vortex street test case

7.2 Grid refinement study for explicit and implicit methods

In this section, we investigate the influence of grid refinement on the time step size adaptation using explicit and implicit Runge-Kutta methods and demonstrate the limitations of explicit methods when used to solve stiff systems of ODEs emerging in the simulation

of turbulent flows. The turbulent test case for the following investigations is the pitching oscillation of the NACA0012 wing profile as described in Section 6.4.

7.2.1 Application of explicit methods

It is well known in literature (see e.g. [37]), that ERK methods have a limited region of stability and in the case of a stiff ODE, the time step size may be severely limited by stability. When using such a method to solve the system of ODEs emerging in the solution process of partial differential equations (PDEs) using a method of lines approach, the maximal stable time step size depends on the spatial resolution, finer grids requiring smaller time step sizes. A too large time step size will first and foremost be noticed in small cells of the grid. In order to reflect such instabilities in small cells in the error estimate of the numerical solution, we recommend to use either the L_∞ norm (5.84) or the 2-norm (5.83) for simulations using ERK methods. The L_2 norm (5.82) is problematic due to the volume weighting, leading to a smaller influence of the small cells. Similarly, such instabilities might also not show off directly when using the lift coefficient as error functional. In this section, the simulations are done using the 2-norm for error estimation. Figures 7.4a, 7.4b, and 7.4c show the time step size evolution for the coarse, medium, and fine grid, using a tolerance of either $\text{tol} = \text{tol}_{\text{final}} = 1 \cdot 10^{-3}$ or $\text{tol} = \text{tol}_{\text{final}} = 1 \cdot 10^{-5}$ (with error per time step control) in Algorithm 1 (and hence in (5.81)).

At first, we note that when using explicit methods, the time step sizes are way too small to simulate a whole period of the pitching oscillation (period length of $T = 0.741$) with a reasonable effort. Two further observations can be made:

1. The time step size is smaller for finer grids.
2. The time step size is almost independent of the chosen tolerance (as long as the tolerance is not too small) except at the beginning of the simulation.

Both observation suggest that the time step size in this case is restricted by the stability of the method. The first one can immediately be explained by the aforementioned dependence of the maximal stable time step size on the grid resolution. To explain the second observation, note that in the case of an unstable time step size, the errors typically increase with every iteration. Furthermore, the time step sizes of the simulations are very small, leading to very small errors when the time step size is in the stable region. Hence, the step size eventually levels out at the stability limit after an initial transient phase in which the desired simulation error is accumulated.

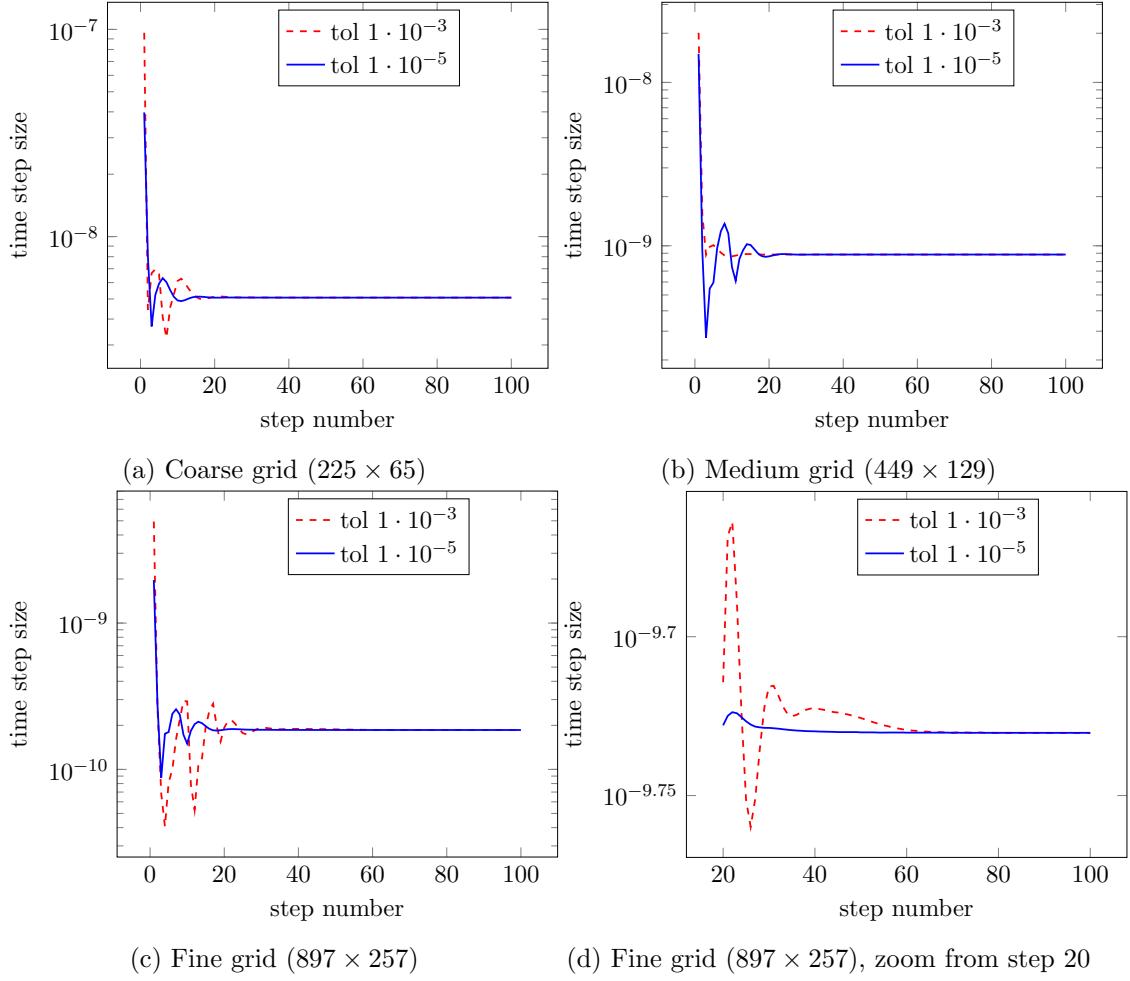


Figure 7.4: Time step size evolution for a NACA0012 pitching oscillation for different grids and tolerances using the ERK5(4) Method 3.26

7.2.2 Application of implicit methods

To investigate the influence of mesh refinement for implicit methods, we use the ESDIRK3(2) Method 3.31 to simulate the aforementioned pitching oscillation of the NACA0012 wing profile. Here, we use a tolerance of $1 \cdot 10^{-3}$ with the lift coefficient as error functional (5.85). Figures 7.5a and 7.5b show the evolution of the lift coefficient and the time step size for the coarse, medium and fine mesh.

As can be seen, the chosen time step size is almost independent of the grid resolution in case of the implicit method. The evolution of the lift coefficient is also similar for all meshes, though the coarse mesh shows slightly lower values. Much larger time steps are chosen when using the implicit ESDIRK3(2) method compared to the explicit method

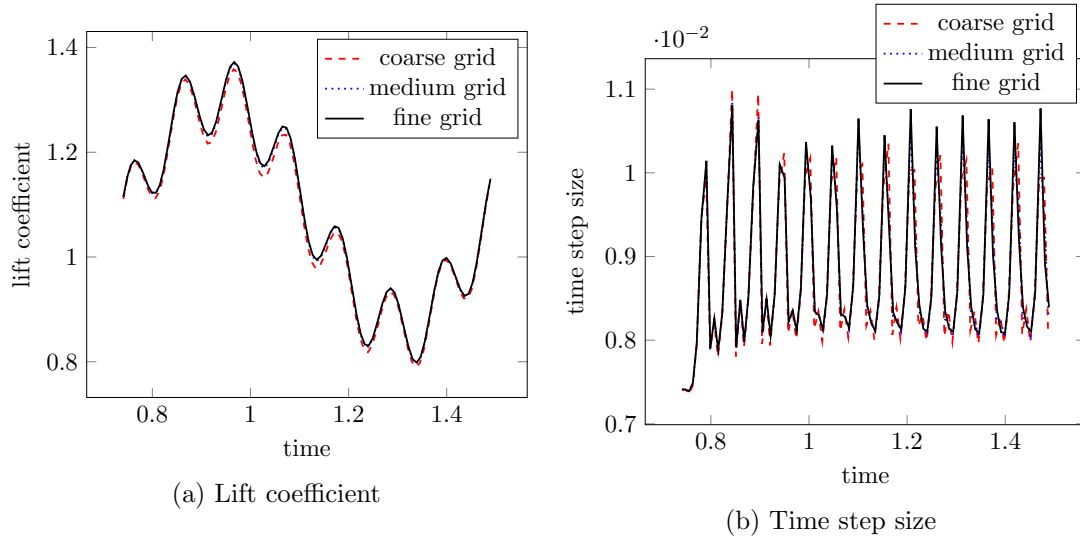


Figure 7.5: Evolution of lift coefficient and time step size for a NACA0012 pitching oscillation for different grids using the ESDIRK3(2) Method 3.31

discussed before (depending on the mesh $1 \cdot 10^6$ to almost $1 \cdot 10^8$ times larger). Here, the step size is limited by accuracy rather than stability. Hence, a difference in the chosen time step sizes can be seen when changing the tolerance, as shown in Figure 7.6.

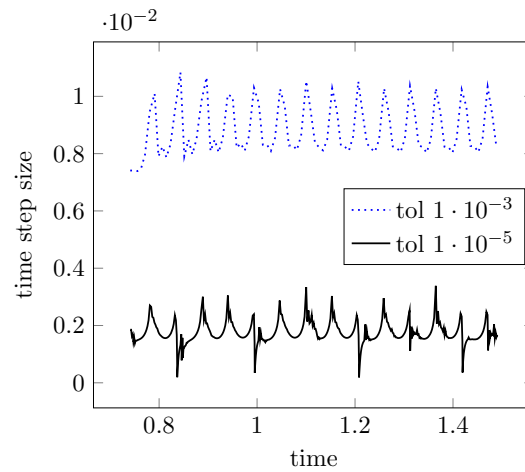


Figure 7.6: Evolution of time step size for a NACA0012 pitching oscillation for different tolerances using the ESDIRK3(2) Method 3.31

7.3 Adaptation algorithm

7.3.1 Verification

To verify the implemented adaptation algorithm, it is used to solve a steady simulation. Once the solution reaches a steady state it should no longer change over time and the temporal discretization error approaches zero. Hence, we expect the step size adaptation algorithm to choose very large time step sizes in this case. To verify this behavior is the purpose of the test case presented here. The laminar NACA0012 test case described in Section 6.3 is chosen for this purpose. The evolution of the time step size and the lift coefficient can be seen in Figure 7.7. A comparison of the coefficient to a steady state computation using the LSBGS method is also given. As expected, the time step sizes eventually get arbitrarily large, and the aerodynamic coefficients match those of the steady computation. Starting the steady simulation with the results of the unsteady solver results in an initial residual of $2 \cdot 10^{-15}$ compared to freestream. Hence, not only the aerodynamic coefficients, but also the whole flow field matches the results of the steady simulation. Note that the growth of the time step size in the end is equal to the maximum allowed growth rate, which is 2 in this case.

In case of the time adaptive computation using SDIRK2(1), two systems of nonlinear equations have to be solved in each time step. As truncation criterion, the residual has to be reduced below $1 \cdot 10^{-12}$ compared to the freestream residual (cf. Section 4.3) in each time step. For the standard steady state computation, a residual reduction below $1 \cdot 10^{-14}$ is achieved. Figure 7.8 shows the residual evaluation for both simulations. As can be seen in the left plot, the convergence for the time adaptive simulation first slows down as the time step sizes increase. However, the initial residual is reduced in each iteration, eventually reaching the truncation criterion. At this point, only a minimum (user defined) number of inner iterations is computed in each time step.

7.3.2 Comparison of algorithms

To compare the different adaptation algorithms introduced in Section 5.1, we focus mainly on the NACA0012 gust encounter test case introduced in Section 6.5.1, since it has a rather low computation cost and the main effects of the different adaptation algorithms can be shown. However, since the results might depend on the special test case, investigations using the SDM configuration as described in Section 6.5.2 as well as the viscous NACA0012 pitching oscillation (Section 6.4) are also shown.

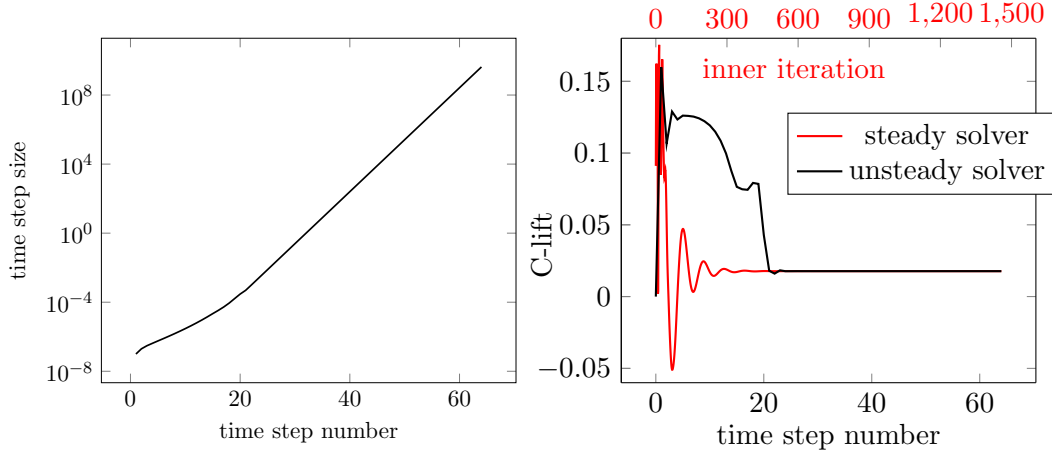


Figure 7.7: Evolution of time step size and lift coefficient for a time adaptive computation of a steady laminar NACA0012 test case compared to a standard steady state simulation

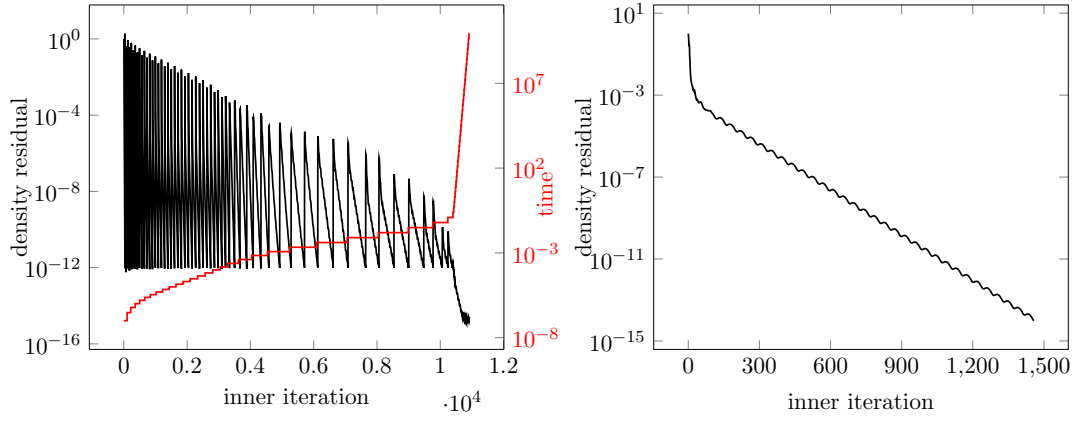


Figure 7.8: Residual plot for a time adaptive computation of a steady laminar NACA0012 test case (left) and a standard steady state simulation (right).

NACA0012 gust encounter

All 24 different possibilities as described in Section 5.1.4 have been applied to the test case using the ESDIRK3(2) Method 3.31 with a tolerance of $\text{tol} = 5 \cdot 10^{-5}$, the scaled 2-norm (5.83) as error norm, and an initial time step size of $0.015s$. The inner iterations stop when the residual is down to $1 \cdot 10^{-11}$ compared to the freestream residual (cf. Section 4.3). Tables 7.1 and 7.2 show the number of accepted and rejected time steps, the approximative error, and the relative CPU time. The temporal error is computed as the error in the lift coefficient in an approximated L_2 -norm over the whole simulation time of $0.15s$ compared to a reference solution with a much smaller time step size.

Only the adaptation algorithm and controller has been changed, all other settings are equivalent for the simulations.

Algorithm	time steps	rejections	Error C-lift	rel. CPU time
I Alg 2	42	6	$4.0 \cdot 10^{-5}$	1.22
I Alg 3	41	2	$7.1 \cdot 10^{-5}$	1.07
I Alg 4	42	2	$4.2 \cdot 10^{-5}$	1.06
PI Alg 2	44	3	$5.2 \cdot 10^{-5}$	1.13
PI Alg 3	47	2	$4.0 \cdot 10^{-5}$	1.14
PI Alg 4	45	2	$4.3 \cdot 10^{-5}$	1.11
P Alg 2	40	2	$5.6 \cdot 10^{-5}$	1.03
P Alg 3	41	2	$4.3 \cdot 10^{-5}$	1.04
P Alg 4	40	1	$4.7 \cdot 10^{-5}$	1.00
DF Alg 2	∞	∞	—	∞
DF Alg 3	44	3	$5.3 \cdot 10^{-5}$	1.14
DF Alg 4	44	2	$4.6 \cdot 10^{-5}$	1.10

Table 7.1: Comparison of different adaptation algorithms for a NACA0012 gust encounter with *error per time step* control and the scaled 2-norm as error functional using an initial step size of 0.015s. Compared are the standard integral (I) controller, the Proportional Integral (PI) controller, the Predictive (P) controller and the Digital Filter (DF). The CPU time is computed relative to the CPU time of P Alg 4.

For error per time step control, the simulations using the new limiter need the lowest CPU time and the fewest step rejections (sometimes on par with Algorithm 3) for all controllers. Comparing the different controllers, the P controller shows the best results for this test case with only 1 or 2 step rejections, the smallest CPU time, and errors on par with those of the other controllers. Furthermore, it is noticeable, that the DF needs some kind of special treatment after a rejected step. In fact, if the last error estimate before a rejection is very small, it can happen that the step size is never reduced enough to get an accepted step. This happens in this simulation when using the DF in combination with Algorithm 2. Using Algorithm 3 or 4 solves this problem, since these algorithms switch to I control after a rejection. For error per unit time step control, similar observations can be made, though there is one case where Algorithm 2 has a similar computational effort as Algorithm 4. Even though using Algorithm 3 sometimes results in a slightly smaller error, the combination of the P controller with the new limiter (Algorithm 4) is

Algorithm	time steps	rejections	Error C-lift	rel. CPU time
I Alg 2	58	6	$2.8 \cdot 10^{-5}$	1.45
I Alg 3	58	4	$2.7 \cdot 10^{-5}$	1.38
I Alg 4	57	2	$3.8 \cdot 10^{-5}$	1.28
PI Alg 2	61	2	$4.8 \cdot 10^{-5}$	1.36
PI Alg 3	62	2	$2.1 \cdot 10^{-5}$	1.37
PI Alg 4	62	2	$3.7 \cdot 10^{-5}$	1.36
P Alg 2	55	3	$5.0 \cdot 10^{-5}$	1.25
P Alg 3	55	2	$3.0 \cdot 10^{-5}$	1.23
P Alg 4	54	2	$4.3 \cdot 10^{-5}$	1.20
DF Alg 2	∞	∞	—	∞
DF Alg 3	61	3	$5.6 \cdot 10^{-5}$	1.40
DF Alg 4	61	3	$3.8 \cdot 10^{-5}$	1.37

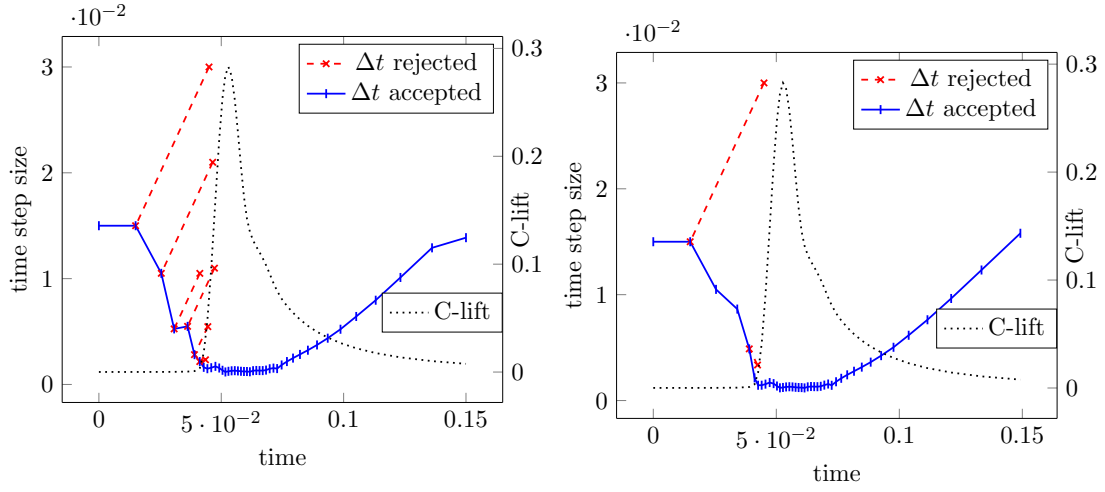
Table 7.2: Comparison of different adaptation algorithms for a NACA0012 gust encounter with *error per unit time step* control and the scaled 2-norm as error functional using an initial step size of 0.015s. The CPU time is computed relative to the CPU time of P Alg 4 in Table 7.1.

arguably the best choice.

For all controllers, it can be noted that using error per time step instead of error per unit time step control yields better results. Note that for error per unit time step control the nondimensionalized time has been used, i.e. it is required that $\|[\epsilon_j^{\text{local}}]\| < \frac{\Delta t_j}{\Delta t_{\text{ref}}}$ tol, with $\Delta t_{\text{ref}} \approx 0.0034s$.

The effect of the limiter (Algorithm 4) can be seen in Figures 7.9 and 7.10. The simulation starts with a step size of $\Delta t_1 = 0.015s$. Since the gust did not yet reach the wing, the error is very small and the step size is increased by $q_{\text{max}} = 2$ to $\Delta t = 0.03s$. At this point ($t = 0.045s$), the gust hits the wing and the step has to be rejected. The new step size is then determined to be $\Delta t_2 = 0.0105s$. Since the gust does not hit the wing at $t = 0.0255s$, we get a very small error again, leading to an increased time step size and a subsequent rejection when using Algorithm 2 without limiter. As shown in Figure 7.9a, this type of repeated rejection can occur several times before the gust is finally reached with an accepted step. If the limiter is used, the information of the first rejected time step is used and the step size is reduced monotonically till the beginning of the gust. When using Algorithm 3, as shown in Figure 7.10a, Δt_3 is smaller than in the case of Algorithm 2 due to the special treatment after a rejected step. However,

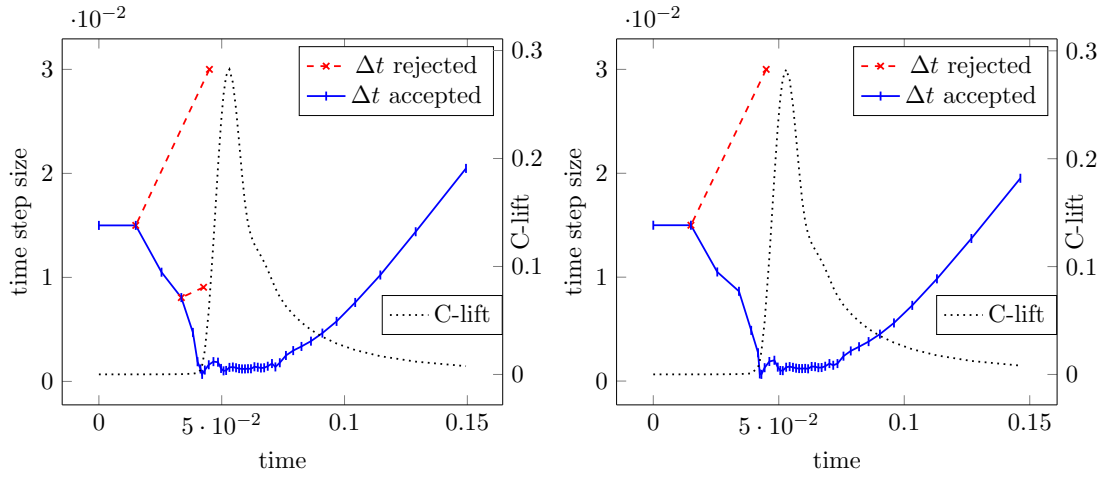
since the gust is not yet reached, the step size is increased afterwards, again leading to a rejection.



(a) Time step size without limiter (Algorithm 2)

(b) Time step size with limiter (Algorithm 4)

Figure 7.9: Effect of new limiter with integral control for a gust encounter of a NACA0012 wing profile using the 2-norm as error functional



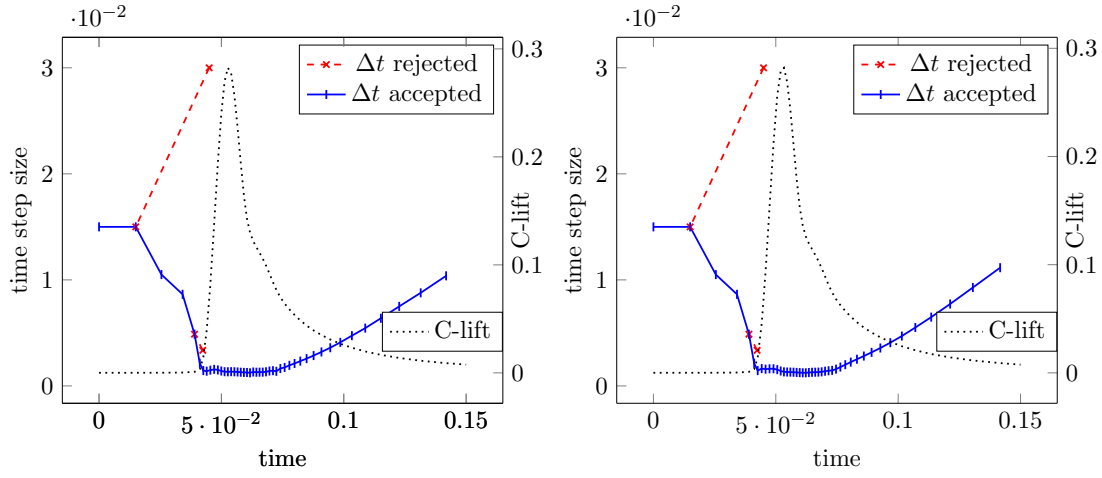
(a) Time step size using Algorithm 3

(b) Time step size using Algorithm 4

Figure 7.10: Effect of new limiter with predictive control for a gust encounter of a NACA0012 wing profile using the 2-norm as error functional

After demonstrating the effect of the limiter, we investigate the differences of the controllers. Comparing Figure 7.11 with Figures 7.10b and 7.9b, we notice that the P controller needs less time steps mainly due to the faster increase of time step sizes after

the gust encounter. Especially the PI controller and the DF are too conservative in the step size choice in this region.



(a) Time step size using the PI controller with Algorithm 4 (b) Time step size using the DF with Algorithm 4

Figure 7.11: Evolution of time step size for a gust encounter of a NACA0012 wing profile using different controllers and the 2-norm as error functional

Next, we simulate the same test case with the lift coefficient as error functional and a tolerance of $\text{tol} = 1 \cdot 10^{-3}$. This time, we only show results for error per time step control, since it is clearly superior to error per unit time step control for the considered test case. As can be seen in Table 7.3, using the new limiter (Algorithm 4) again shows the most efficient results. However, this time the PI controller and the I controller are superior to the P controller. Figure 7.12 shows the time step size evolution and rejections for this test case using different controllers with Algorithm 4. As can be seen, the curves are less smooth than in the case of the 2-norm as error functional (cf. Section 7.3.3). This leads to more rejections for all controllers except for the PI controller.

By comparing the results using either the 2norm or the lift coefficient as error functional, it can be noticed, that the best controller choice depends on the specific parameter settings of the test case. Further tests show that the choice of the initial step size is also an important parameter influencing the results for the different adaptation algorithms and controllers. In order to get a more meaningful result about the choice of the adaptation algorithm, several tests with initial step sizes 0.001, 0.002, ..., 0.02 have been done. Tables 7.4 and 7.5 show the average results using the 2-norm and the lift coefficient as error functional, respectively.

Regarding the adaptation algorithm, it is again evident that the Algorithm 4 using the

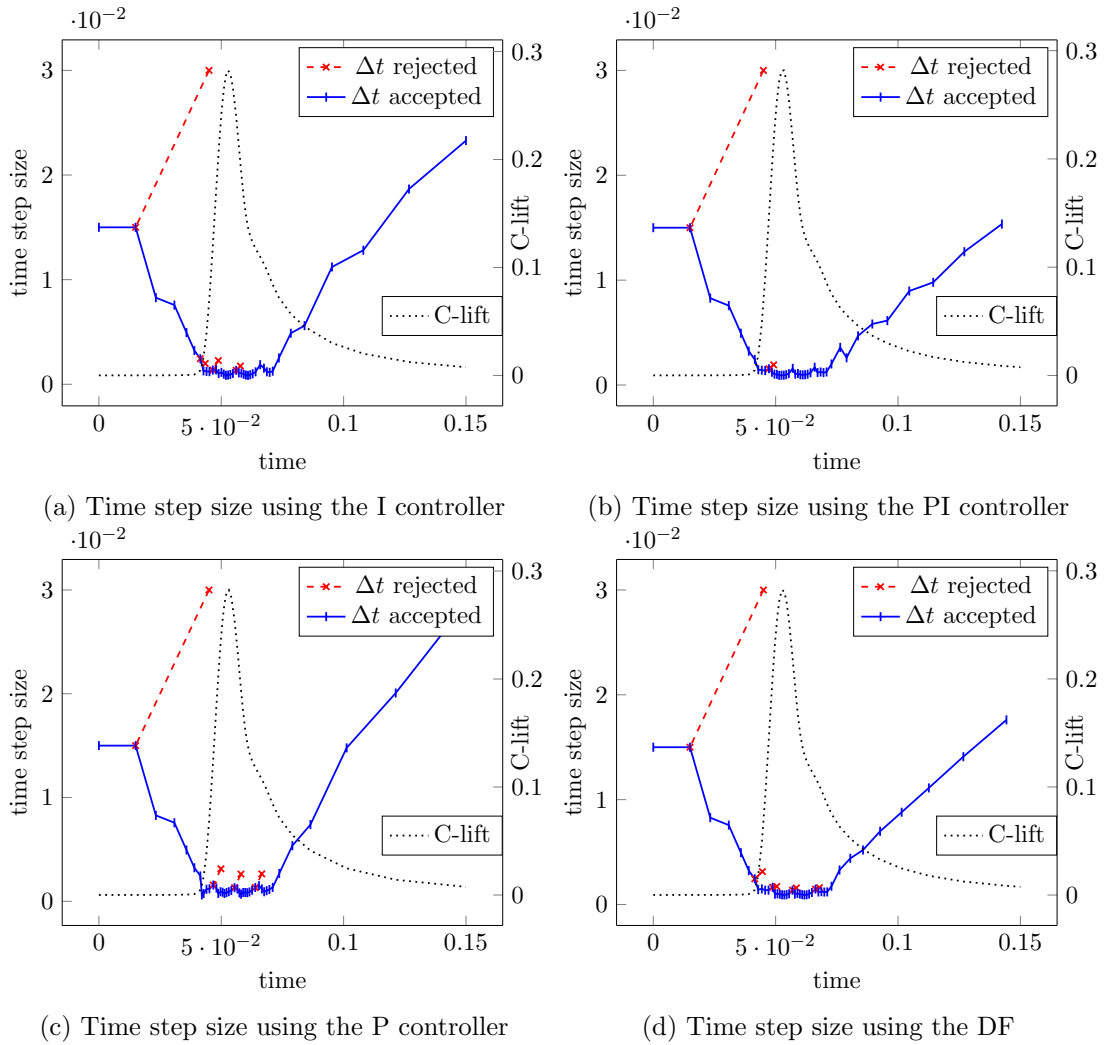


Figure 7.12: Evolution of time step size for a gust encounter of a NACA0012 wing profile using different controllers in combination with the limiter (Algorithm 4) and the lift coefficient as error functional using an initial step size of 0.015s

Algorithm	time steps	rejections	Error C-lift	rel. CPU time
I Alg 2	37	6	$1.1 \cdot 10^{-4}$	1.10
I Alg 3	41	7	$3.0 \cdot 10^{-5}$	1.19
I Alg 4	39	4	$3.1 \cdot 10^{-5}$	1.05
PI Alg 2	46	6	$2.6 \cdot 10^{-5}$	1.24
PI Alg 3	44	3	$2.6 \cdot 10^{-5}$	1.12
PI Alg 4	42	2	$3.0 \cdot 10^{-5}$	1.06
P Alg 2	42	6	$5.2 \cdot 10^{-5}$	1.16
P Alg 3	43	6	$3.6 \cdot 10^{-5}$	1.19
P Alg 4	41	4	$3.6 \cdot 10^{-5}$	1.07
DF Alg 2	∞	∞	—	∞
DF Alg 3	47	7	$2.8 \cdot 10^{-5}$	1.28
DF Alg 4	42	5	$2.9 \cdot 10^{-5}$	1.13

Table 7.3: Comparison of different adaptation algorithms for a NACA0012 gust encounter with error per time step control and the lift coefficient as error functional using an initial step size of $0.015s$. The CPU time is computed relative to the CPU time of P Alg 4 in Table 7.1.

new limiter is the best choice. For the controller, the P controller is the recommended choice in the case of the 2-norm as error functional. When using the lift coefficient as error functional instead, the I controller in combination with Algorithm 4 shows slightly better results regarding CPU time and error. Furthermore, the PI controller, is a good option as well. Though having a slightly higher computational cost, it shows the fewest step rejections and the smallest error.

SDM gust encounter

In case of the SDM gust encounter test case (cf Section 6.5.2) the comparison of the adaptation algorithms has been done using an initial time step size of $0.01s$, the scaled 2-norm as error norm and tolerances of $\text{tol} = 1 \cdot 10^{-5}$ and $\text{tol} = 5 \cdot 10^{-5}$ using error per time step control. The results using the ESDIRK3(2) Method 3.31 are shown in Tables 7.6 and 7.7. The test case shows very similar results to the NACA0012 gust encounter. Once again, using the limiter (Algorithm 4) yields the most efficient results. Regarding the controllers, the PI controller needs the fewest rejections but also the most time steps resulting in a slightly larger computational time than the simulations using the P or I

Algorithm	time steps	rejections	Error C-lift	rel. CPU time
I Alg 2	41.8	4.3	$4.7 \cdot 10^{-5}$	1.15
I Alg 3	42.2	2.45	$4.9 \cdot 10^{-5}$	1.09
I Alg 4	41.95	1.8	$4.7 \cdot 10^{-5}$	1.06
PI Alg 2	44.25	3.35	$5.7 \cdot 10^{-5}$	1.13
PI Alg 3	46.55	1.75	$4.6 \cdot 10^{-5}$	1.11
PI Alg 4	45.7	1.25	$4.9 \cdot 10^{-5}$	1.08
P Alg 2	40.5	1.95	$5.0 \cdot 10^{-5}$	1.04
P Alg 3	41.15	1.75	$4.6 \cdot 10^{-5}$	1.03
P Alg 4	40.4	1.05	$4.7 \cdot 10^{-5}$	1.00
DF Alg 2	∞	∞	—	∞
DF Alg 3	45.1	2.6	$4.9 \cdot 10^{-5}$	1.13
DF Alg 4	44.45	2.0	$5.2 \cdot 10^{-5}$	1.10

Table 7.4: Comparison of different adaptation algorithms for a NACA0012 gust encounter with error per time step control and the scaled 2-norm as error functional averaged over multiple simulations using different initial step sizes.

Algorithm	time steps	rejections	Error C-lift	rel. CPU time
I Alg 2	38.25	7.10	$1.1 \cdot 10^{-4}$	1.15
I Alg 3	40.75	5.75	$4.3 \cdot 10^{-5}$	1.14
I Alg 4	39.85	4.75	$4.2 \cdot 10^{-5}$	1.09
PI Alg 2	43.80	4.45	$4.4 \cdot 10^{-5}$	1.16
PI Alg 3	46.20	2.65	$3.3 \cdot 10^{-5}$	1.14
PI Alg 4	45.15	3.05	$3.2 \cdot 10^{-5}$	1.13
P Alg 2	41.95	5.95	$5.7 \cdot 10^{-5}$	1.15
P Alg 3	42.50	5.15	$5.0 \cdot 10^{-5}$	1.13
P Alg 4	41.55	4.45	$4.6 \cdot 10^{-5}$	1.09
DF Alg 2	∞	∞	—	∞
DF Alg 3	42.35	5.10	$7.0 \cdot 10^{-4}$	1.14
DF Alg 4	40.90	4.65	$7.0 \cdot 10^{-4}$	1.10

Table 7.5: Comparison of different adaptation algorithms for a NACA0012 gust encounter with error per time step control and the lift coefficient as error functional averaged over multiple simulations using different initial step sizes. The CPU time is computed relative to the CPU time of P Alg 4 in Table 7.4.

controller with the limiter.

Algorithm	time steps	rejections	Error C-lift	rel. CPU time
I Alg 2	70	5	$5.8 \cdot 10^{-6}$	1.06
I Alg 3	71	4	$6.2 \cdot 10^{-6}$	1.05
I Alg 4	70	2	$5.9 \cdot 10^{-6}$	1.02
PI Alg 2	72	2	$5.7 \cdot 10^{-6}$	1.02
PI Alg 3	76	2	$5.7 \cdot 10^{-6}$	1.04
PI Alg 4	73	2	$5.6 \cdot 10^{-6}$	1.03
P Alg 2	72	5	$5.2 \cdot 10^{-6}$	1.05
P Alg 3	70	3	$5.3 \cdot 10^{-6}$	1.02
P Alg 4	69	2	$5.4 \cdot 10^{-6}$	1.00
DF Alg 2	∞	∞	—	∞
DF Alg 3	75	5	$5.7 \cdot 10^{-6}$	1.08
DF Alg 4	72	3	$5.7 \cdot 10^{-6}$	1.04

Table 7.6: Comparison of different adaptation algorithms for a gust encounter of the SDM configuration using error per time step control, the 2-norm as error norm and a tolerance of $1 \cdot 10^{-5}$.

Turbulent NACA0012 pitching oscillation

Regarding the turbulent NACA0012 pitching oscillation test case, Table 7.8 shows that the results are quite different in this case. In this simulation, one period was computed using the ESDIKR3(2) method with an initial step size of $3.705 \cdot 10^{-3}$, which corresponds to 200 steps per period. Choosing a tolerance of $\text{tol} = 1 \cdot 10^{-5}$ with the 2-norm as error functional, not a single step rejection was necessary using the PI controller, I controller, or the DF. Since Algorithms 2, 3, and 4 only differ after a rejection, the results for these controllers are independent of the specific choice of algorithm. The P controller on the other hand needs a few rejections. For this test case, the determined time step size varies little and the limiter in Algorithm 4 has almost no effect.

As shown in Figure 7.13, the different controllers show a qualitatively similar behavior. The main difference is the amplitude of the oscillations, with the PI controller having the smallest amplitude, followed by the DF and the I controller. The P controller has the largest amplitude, eventually leading to some rejections. Note that the oscillations are due to the higher frequency pitching oscillation of the problem at hand and not an

Algorithm	time steps	rejections	Error C-lift	rel. CPU time
I Alg 2	40	3	$2.4 \cdot 10^{-5}$	1.03
I Alg 3	41	3	$2.1 \cdot 10^{-5}$	1.04
I Alg 4	41	2	$2.4 \cdot 10^{-5}$	1.00
PI Alg 2	44	2	$4.3 \cdot 10^{-5}$	1.04
PI Alg 3	48	2	$2.3 \cdot 10^{-5}$	1.08
PI Alg 4	45	1	$2.3 \cdot 10^{-5}$	1.03
P Alg 2	43	5	$1.6 \cdot 10^{-5}$	1.08
P Alg 3	40	3	$2.3 \cdot 10^{-5}$	1.02
P Alg 4	41	3	$2.3 \cdot 10^{-5}$	1.01
DF Alg 2	∞	∞	—	∞
DF Alg 3	44	3	$2.1 \cdot 10^{-5}$	1.06
DF Alg 4	42	3	$2.6 \cdot 10^{-5}$	1.03

Table 7.7: Comparison of different adaptation algorithms for a gust encounter of the SDM configuration using error per time step control, the 2-norm as error norm and a tolerance of $5 \cdot 10^{-5}$.

Algorithm	time steps	rejections	Error C-lift	rel. CPU time
I control Alg 2, 3, or 4	161	0	$5.9 \cdot 10^{-5}$	1.00
PI control Alg 2, 3, or 4	161	0	$3.8 \cdot 10^{-5}$	1.01
P Alg 2	162	3	$6.4 \cdot 10^{-5}$	1.03
P Alg 3	162	2	$7.4 \cdot 10^{-5}$	1.02
P Alg 4	162	3	$6.4 \cdot 10^{-5}$	1.03
DF Alg 2, 3, or 4	160	0	$4.4 \cdot 10^{-5}$	1.00

Table 7.8: Comparison of different adaptation algorithms for the NACA0012 pitching oscillation test case. The CPU time is computed relative to the CPU time of I Alg 4 in Table 7.8.

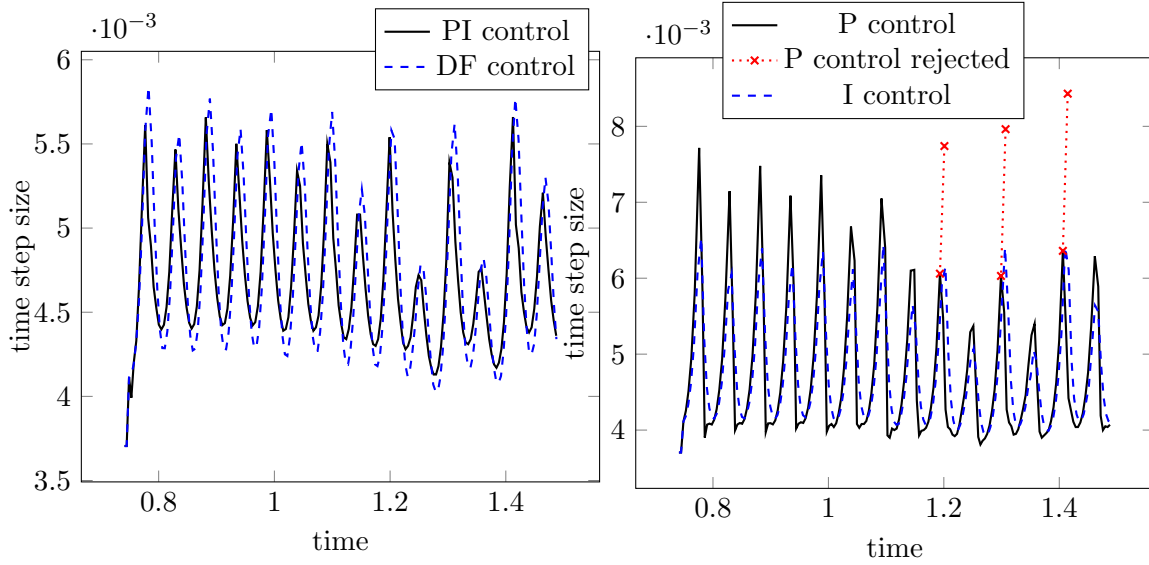


Figure 7.13: Time step size evolution for different controllers for the NACA0012 pitching oscillation using Algorithm 4

effect of the step size controllers as sometimes encountered when using explicit methods for stiff ODEs (cf. [35]).

Due to the results shown in this section, it is recommended to use the new limiter (Algorithm 4) as adaptation algorithm. This is especially recommended for the gust encounter test cases, where the changes in the time step size are large. However, since the adaptation algorithms only differ after a rejected time step, the overall difference in efficiency is typically in the range of a few percentages. Regarding the controller, there is no obvious best choice. For the gust encounter simulations the P controller was the most efficient. However it had the worst results in the pitching oscillation test case. The PI controller usually had the fewest rejections, but also needed the most accepted time steps. The I controller showed good results in all simulations, especially in combination with the limiter. The DF often lagged behind in case of the gust encounter simulations but performed very good in case of the pitching oscillation test case. When using the DF, it is important to use a special treatment after a rejected step as done in Algorithms 3 and 4, since the algorithm might stagnate otherwise.

7.3.3 Error functional

In this section, the influence of the choice of error norm or functional (cf. Section 5.3) on the choice of the time step sizes is investigated. In detail, the use of either the

lift coefficient, the L_2 -norm, the 2-norm, or the L_∞ -norm is compared. Using the lift coefficient obviously doesn't work for all problems. For example for the Ehrenfried Vortex test case, as described in Section 6.1, no lift coefficient is available. Furthermore, note that even if a structure is available and the lift coefficient can be computed, it can happen that the error estimate is zero even though there is an error in the individual variables. This can happen either due to cancellation of errors or if the errors in the solution are in parts that are not used for the computation of the lift coefficient. However, this could also be a desired property in case we are specifically interested in the lift coefficient and not so much in the variables of the whole flow field. The choice of the 2, L_2 , or L_∞ norm could then result in unnecessarily small time steps. This can be observed for example in the case of a gust encounter of the NACA0012 airfoil (cf. Section 6.5.1). As can be seen in Table 7.9 and Figure 7.14, using the lift coefficient as error functional allows for larger time steps compared to the simulations using the L_2 , L_∞ or 2-norm for a similar accuracy in the lift coefficient. However, the step size choice is also less smooth in case of the lift coefficient as error functional. This leads to an increased number of step rejections. The tolerance of the computations was chosen such that the resulting error in the lift coefficient is similar for all computations. All other parameters are the same as those used in the NACA0012 gust encounter simulation of Section 7.3.2 with an initial time step size of 0.015 and the P controller with the limiter as adaptation algorithm.

Error functional	time steps	rejections	Error C-lift	rel. CPU time
lift coefficient	37	5	$5.1 \cdot 10^{-5}$	1.04
2-norm	40	1	$4.7 \cdot 10^{-5}$	1.00
L_∞ -norm	46	2	$4.8 \cdot 10^{-5}$	1.12
L_2 -norm	52	1	$4.7 \cdot 10^{-5}$	1.17

Table 7.9: Comparison of different error functionals for the NACA0012 gust encounter simulation. The CPU time is computed relative to the CPU time when using the 2-norm.

The dependence of the error in the lift coefficient (computed as approximation to the L_2 -norm over the whole simulation) as well as a work over error plot are shown in Figures 7.15 and 7.16 using the P and PI controller, respectively, with limiter. It can be seen, that using the 2-norm or the lift coefficient as error functional is the most efficient. This can be explained by the slow increase of the time step size after the gust encounter for the L_2 and L_∞ norm as shown in Figure 7.14. Even though the choice of the 2-norm

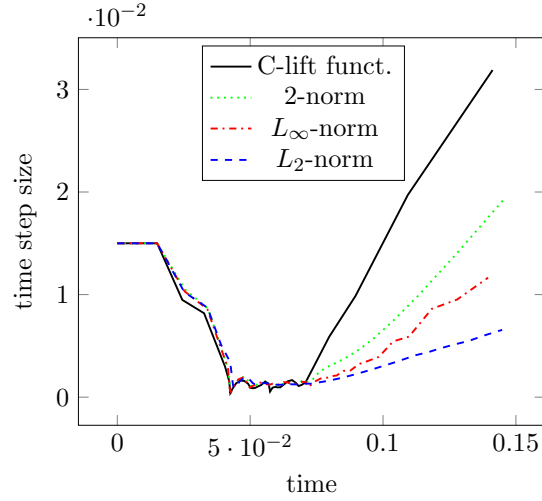


Figure 7.14: Time step size evolution for different error functionals for the NACA0012 gust encounter simulation for similar levels of accuracy (cf. Table 7.9) using the predictive controller with limiter.

yields smaller time step sizes compared to the lift coefficient, it can compete due to the lower amount of step rejections.

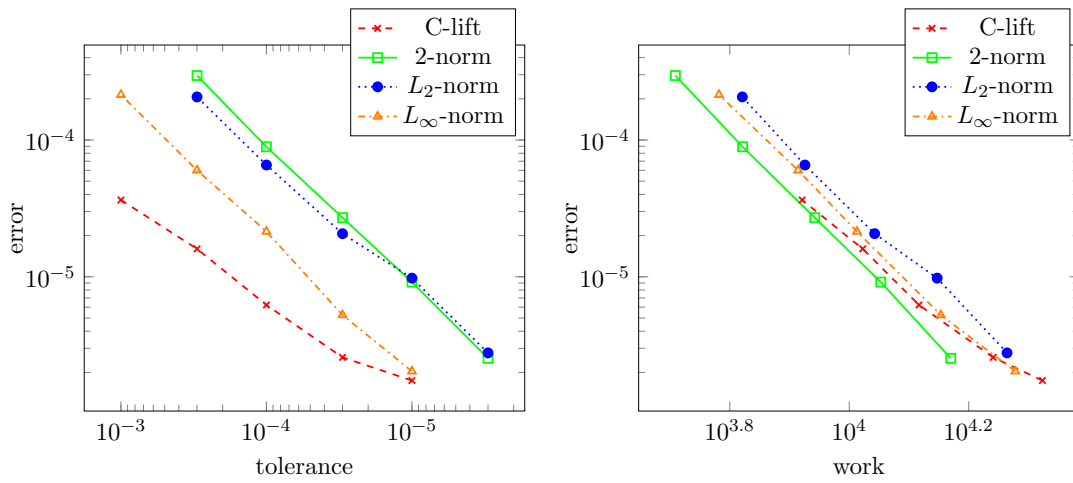


Figure 7.15: Error-tolerance and error-work plots for the NACA0012 gust encounter simulation using different norms or functionals for error estimation. The adaptation was done using the predictive controller (5.43) with limiter (Algorithm 4).

As a second test case, the viscous NACA0012 pitching oscillation (cf. Section 6.4) is investigated. The simulation was done using the PI controller with limiter (Algorithm

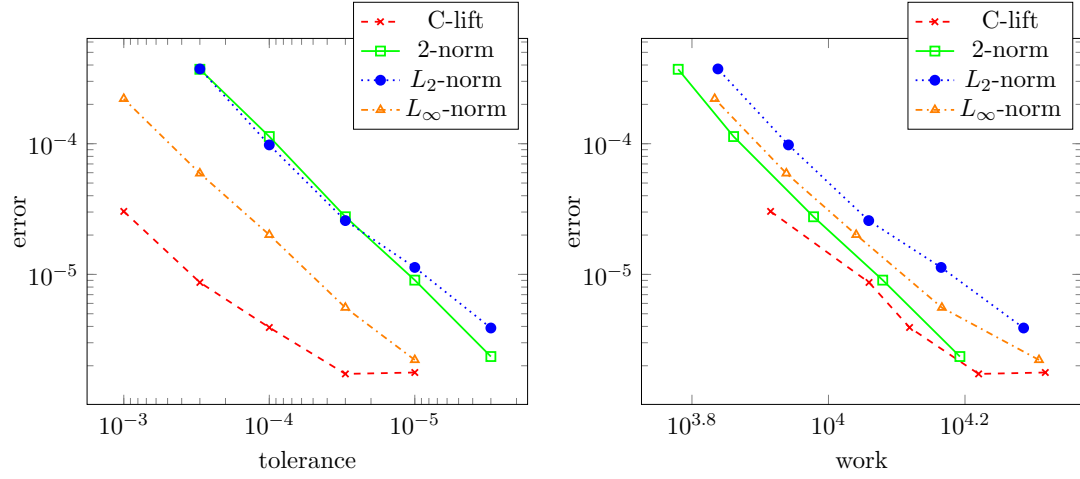


Figure 7.16: Error-tolerance and error-work plots for the NACA0012 gust encounter simulation using different norms or functionals for error estimation. The adaptation was done using the PI controller (5.38) with limiter (Algorithm 4).

4). As can be seen in Figure 7.17, all error functionals show a similar efficiency, with the L_∞ norm being slightly behind and the L_2 norm slightly ahead.

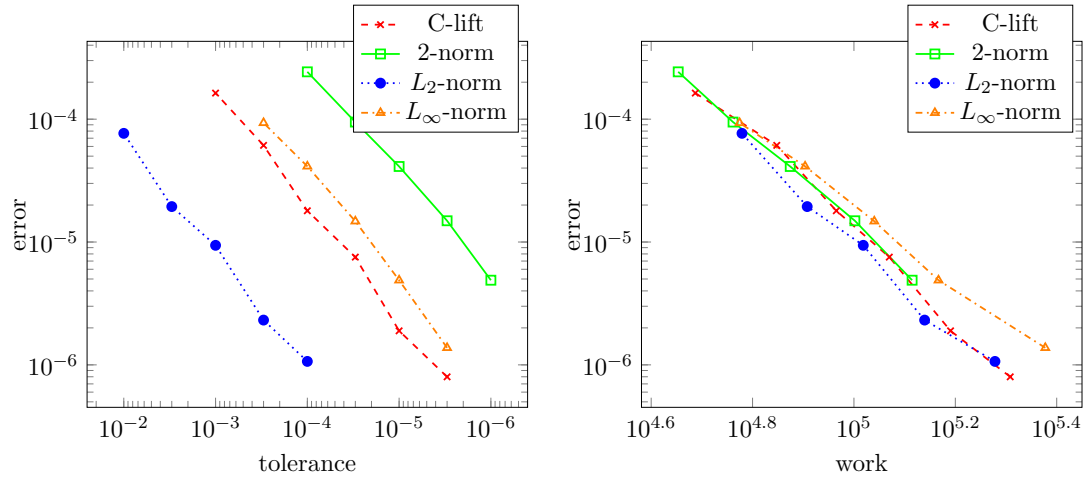


Figure 7.17: Error-tolerance and error-work plots for the turbulent NACA0012 pitching oscillation simulation using different norms or functionals for error estimation. The adaptation was done using the PI controller (5.38) with limiter (Algorithm 4).

While there is no clear best error functional for every test case, the use of the (scaled) 2-norm (5.83) is suggested. The L_2 and L_∞ norms are in case of the gust encounter simulation significantly less efficient and the use of the lift coefficient typically results

in more step size rejections. Furthermore, the simulations using the 2-norm show a nice linear dependency of the error on the tolerance. This is not always the case for the lift coefficient as can be seen in Figure 7.16.

7.4 Solution of nonlinear systems

The solution of the nonlinear systems of equations has a significant impact on the stability and computational cost of aerodynamic fluid flow simulations. In the following subsections, the influence of the choice of the solution method, initial value and truncation criterion on fluid flow simulations presented in this thesis are investigated.

7.4.1 Solution methods

The efficient solution of the nonlinear systems of equations arising in aerodynamic CFD simulations is a research topic on its own and it is out of the scope of this thesis to analyse it in detail. Nevertheless, some results on the efficiency and robustness of the solution algorithms available in the TAU code (cf. Section 4.1) are presented in the following. The investigations are focused on the turbulent test cases presented in this thesis, since they are typically much harder to solve than laminar or even inviscid cases.

NACA0012 pitching oscillation

To test the different solver settings, five time steps of size $3.70543 \cdot 10^{-3}$ s, which corresponds to 200 time steps per period, are computed with a BDF1 scheme. The nonlinear systems of equations are converged to a residual of $1 \cdot 10^{-9}$ compared to the freestream residual. Six different solver settings as shown in Table 7.10 are considered. The LU-SGS and LSBGS solution methods are as described in section 4.1. For the multigrid, either a v-cycle with 3 or 6 levels or a 1 level single grid is chosen. The option to use several stages for the Runge-Kutta method described in 4.1 is only available for LSBGS. The CFL number is constant in case of the LU-SGS method and increases during the solution in case of the LSBGS method. For the LSBGS method, there is also the option to freeze the approximate Jacobian $\mathcal{P}_k^{1st}$ for several steps, which has been utilized in Case 3. As can be seen in Table 7.10, using multigrid accounts for a speed up of approximately factor 10 in combination with the LU-SGS method for this test case. When using the LSBGS method on the other hand, no speed up is observed. Furthermore, it is advantageous to use 1 Runge-Kutta stage and freeze the approximate Jacobian, resulting in a simplified Newton type iteration. Freezing the Jacobian for 20 steps results in a speed up

of factor 2 and the usage of a 3 stage Runge-Kutta method in the solution algorithm for the nonlinear system increases the computational cost by a factor of 2.7. Furthermore, it can be observed that the LSBGS algorithm is more efficient than LU-SGS here. Note that in addition to the efficiency aspect, the LU-SGS algorithm was not able to get the residual below $1 \cdot 10^{-10}$ while the LSBGS was able to reduce it below $1 \cdot 10^{-11}$ for all considered settings. Figure 7.18 shows the residual evaluation for the first physical time step. It can be seen that the LU-SGS method has an especially slow convergence in the end, where the residual is already low. Even though the LSBGS method with multigrid needs the fewest inner iterations, it is not the most efficient method for this test case, since one iteration equals to a whole multigrid cycle, which has a significantly higher computational effort than a single grid iteration.

Case	Method	Multigrid	RK coeff	CFL	recompute $\mathcal{P}_k^{\text{1st}}$	CPU time
1	LUSGS	-	1.0	1000	every step	60.0
2	LUSGS	6v	1.0	1000	every step	6.1
3	LSBGS	-	1.0	1000...10000	every 20 steps	1.0
4	LSBGS	-	1.0	1000...10000	every step	2.0
5	LSBGS	-	0.15, 0.4, 1.0	1000...10000	every step	5.4
6	LSBGS	3v	1.0	1000...10000	every step	2.4

Table 7.10: Investigated solution algorithm settings and CPU time for the NACA0012 pitching oscillation test case

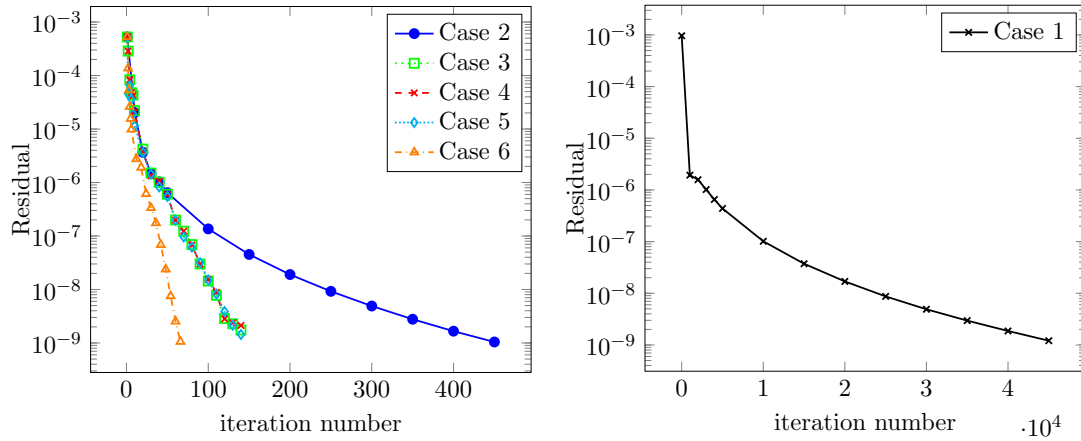


Figure 7.18: Convergence plot for first time step of NACA0012 pitching oscillation with BDF1 using different solver settings as described in Table 7.10

The results presented here are dependent on the specific implementation of the methods and the test case at hand. In particular it should be noted that significantly different results are expected for steady simulations. In this case, the systems of equations are typically harder to solve and a good initial solution from the last time step is not at hand.

Transport aircraft gust encounter

Severe convergence problems have been encountered in case of the transport aircraft gust encounter test case (cf. Section 6.5.3). Here, a residual reduction below $1 \cdot 10^{-8}$ compared to freestream has been used as convergence criterion. Further convergence below $1 \cdot 10^{-9}$ could not be achieved with any tested method for reasonable time step sizes. Furthermore, large step sizes sometimes resulted in a not converged solution. The maximal converging step size is dependent on the location of the gust and is not known exactly. For this test case it was observed that using the LU-SGS method resulted in less convergence problems than using LSBGS. The latter resulted in many rejected steps due to not converging solver iterations and is thus not considered any further. Regarding the multigrid, note that a 3v cycle is used for this test case. Using 6v as for the NACA0012 pitching oscillation resulted in further convergence problems. Due to the size of this test case it was computed in parallel on 96 processes. The exact reasons for the convergence problems have not been investigated in detail. This is out of the scope of this thesis.

7.4.2 Stage value predictors

In this section, different choices of stage value predictors, i. e. different starting values for the nonlinear systems to be solved in the individual stages of a DIRK method, are investigated. Recall from Section 4.2 that three different choices of stage value predictors were introduced taking either

1. the solution of the last time step
2. the solution of a previous stage (if available)
3. an extrapolation from the solution of a previous stage

as initial value for the solution of the nonlinear system. Once again the focus is on the turbulent test cases since they are more interesting regarding the solution of the algebraic equations. To compare the different stage value predictors described in section 4.2, both turbulent test cases are computed with constant time steps and the required computational time is compared.

In case of the NACA0012 pitching oscillation, two simulations with 100 and 200 steps per period were performed using the ESDIRK3(2) method and the solver settings of Case 3 in Table 7.10. The relative CPU times are given in Table 7.11.

Stage value predictor	rel. CPU time 100 spp	rel. CPU time 200 spp
Choice 1	1.33	2.23
Choice 2	1.0	1.61
Choice 3	-	1.15

Table 7.11: relative computational time for different stage value predictors for the NACA0012 pitching oscillation test case for 100 and 200 steps per period (spp)

As can be seen, Choice 2 is significantly faster than Choice 1. Using the extrapolation (Choice 3), leads to a further speed up but is also less reliable, as it lead to a crash in the 54th physical time step when using 100 steps per period. Before the crash it was also faster than both other methods, taking about 0.77 times the computational time as Choice 2. As can be seen in Figure 7.19, the crash occurred at a local minimum of

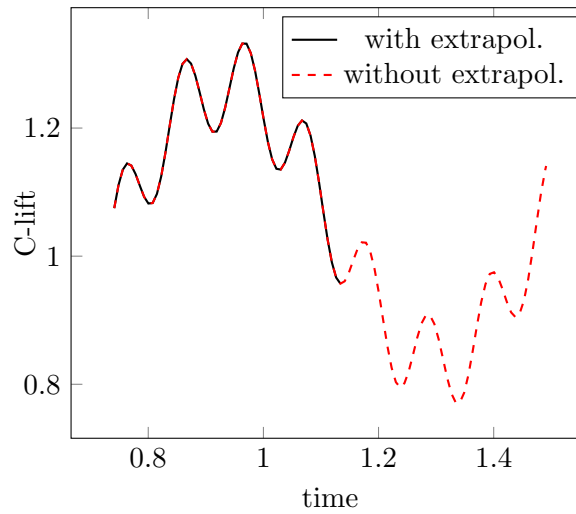


Figure 7.19: C-lift for NACA0012 pitching oscillation with 100 steps per period. Position of crash using extrapolation

the lift coefficient. For such a sinusoidal pitching oscillation, extrapolation is especially susceptible to problems near the local extrema. For this test case, either Choice 2 or Choice 3 could be the most efficient. Therefore, these cases are considered for an other test case in the following.

The NACA0012 pitching oscillation test case presented above is subsonic. However, as described in Section 4.2, the stage value predictor using the extrapolation might be problematic in case of shocks. Therefore, an evaluation of the influence of the stage value predictors for the transport aircraft gust encounter test case, which is transonic, was performed. A constant time step size $\Delta t = 0.005s$ has been chosen in combination with the SDIRK2(1) method. When using the extrapolation (Choice 3 in Section 4.2), the simulation crashes in the 32nd physical time step using LU-SGS with a 3v multigrid and $CFL = 100$. When using Choice 2 as stage value predictor on the other hand, the residual was reduced below $1 \cdot 10^{-8}$ in every time step without convergence problems. For the first 31 physical time steps the computational time for both choices was similar. Switching to a time step size of $\Delta t = 0.0025s$ leads to convergence in every step when using extrapolation. However, the computational effort is larger compared to the usage of Choice 2 (cf. Table 7.12).

Stage value predictor	rel. CPU time $\Delta t = 0.005s$	rel. CPU time $\Delta t = 0.0025s$
Choice 2	1.0	1.48
Choice 3	-	2.17

Table 7.12: relative computational time for different stage value predictors for the transport aircraft gust encounter test case for $\Delta t = 0.005s$ and $\Delta t = 0.0025s$

In summary, it can be concluded that using Choice 2 as stage value predictor is typically more efficient than Choice 1. Furthermore, using the extrapolation (Choice 3) can be advantageous or disadvantageous, depending on the test case. However, it may result in convergence problems for large time steps. Therefore, Choice 2 is recommended as standard procedure.

7.5 Investigation of DIRK method properties

In this chapter, different DIRK methods are compared and recommendations regarding favorable properties for aerodynamic fluid flow applications as presented in this thesis are made.

Nonlinear stability

First, the stability of the implemented methods is considered. Note that the system of ODEs to be solved after spatial discretization of the governing equations is nonlinear and especially in the turbulent case also stiff. Due to the stiffness, the use of implicit methods

with good stability properties is preferred (cf. Section 7.2). However, the question arises, whether nonlinear (B-)stability (Definition 3.12) is required or whether linear stability (A-stability) is sufficient. As described in Section 3.5, most implicit methods considered in this thesis are only A-stable (Definition 3.6) and not B-stable, with the exception of the SDIRK3(2) Method 3.28. However, this linear stability property seems to be sufficient for the applications considered in this thesis, since no stability issues of the A-stable methods have been observed in any of the simulations.

Influence of c coefficients of the Runge-Kutta method

In fact, the B-stable SDIRK3(2) method had problems when applied to a gust encounter test case with adaptive time stepping. Figure 7.20 shows the evolution of the lift coef-

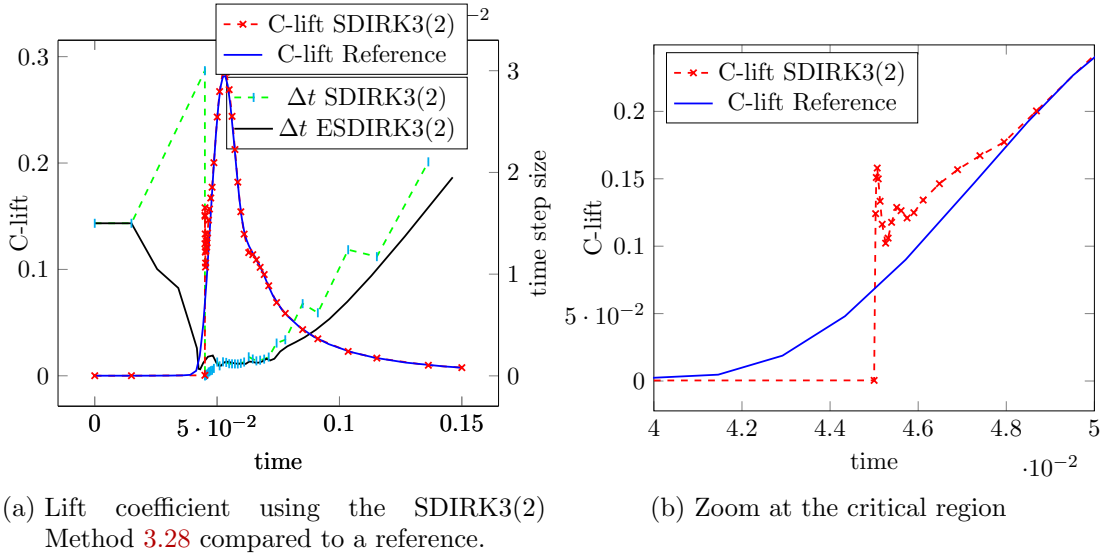


Figure 7.20: Problem when using SDIRK3(2) Method 3.28 with time step adaptation for the simulation of a gust encounter of the NACA0012 airfoil profile. Major issues occur at the beginning of the gust encounter.

ficient for the simulation of the NACA0012 gust encounter simulation performed with this method. As can be seen, major issues arise at the beginning of the gust encounter. These problems are not due to a lack of stability of the method itself, but due to a too large time step at the beginning of the gust. The adaptation algorithm fails to reject the second time step, since, despite the large actual error, the error estimate is smaller than the tolerance. This is due to the fact that for the SDIRK3(2) method (Method 3.28), the coefficients c_i of the Butcher tableau are all smaller than 1. Hence, in the second time step of the simulation, the residual is only evaluated at time instances before the actual

gust encounter, leading to a correspondingly small error estimate. On the contrary, if the method would have $c_i > 1$ for some $i \in \{1, \dots, s\}$, it might happen that a step is rejected unnecessarily. While this would not lead to such errors as in Figure 7.20, the use of DIRK methods with $0 \leq c_i \leq 1$ and $c_j = 1$ for at least one j is still recommended.

Order of the method

Regarding the order of the methods, it is clear, that lower order methods tend to be more efficient for very large tolerances (or large time step sizes) and higher order methods eventually overtake as the tolerance (or time step size) is reduced. Since the desired convergence order for methods of order 4 or greater could not be shown for small time steps (cf. Section 7.1), the main focus in this thesis is on methods of order 2 and 3. Furthermore, it has been observed, that the choice of larger time step sizes typically results in a slower convergence of the nonlinear system solver. This is due to two reasons. First, the initial solution (stage value prediction) is typically worse if the time step is larger. Second, the diagonally dominance of the matrix (4.7) is increased for smaller time steps, which typically favours Gauss-Seidel like solution algorithms. For complex configurations like the transport aircraft half model (cf. Section 6.5.3) the solver sometimes did not converge at all for large time step sizes resulting in a rejection of the step and a reduction of the step size. In such a case, the advantage of higher order methods to be able to take larger time step sizes is gone. It can be concluded, that the best choice of the order depends on the test case, the accuracy needs, and the nonlinear solution algorithm.

L-stability

For a simulation of a pitching airfoil like the NACA0012 test case described in 6.4, the typical procedure is to start the pitching oscillation from a steady state solution. In this case there is a discontinuity in the acceleration of the airfoil at the beginning of the unsteady simulation. This discontinuity leads to very small time steps in the initial transient phase when adaptive time stepping is used as shown in Figure 7.21d. Often only the final periodic motion of the pitching oscillation is of interest, in which case it is more efficient to use constant rather than adaptive time stepping in the initial transient phase.

Such simulations with constant time steps with different DIRK methods are shown in Figure 7.21. It can be seen, that the L-stability of the method is of special importance in this case. The not L-stable SDIRK3(2) and ESDIRK2(1) methods show strong oscil-

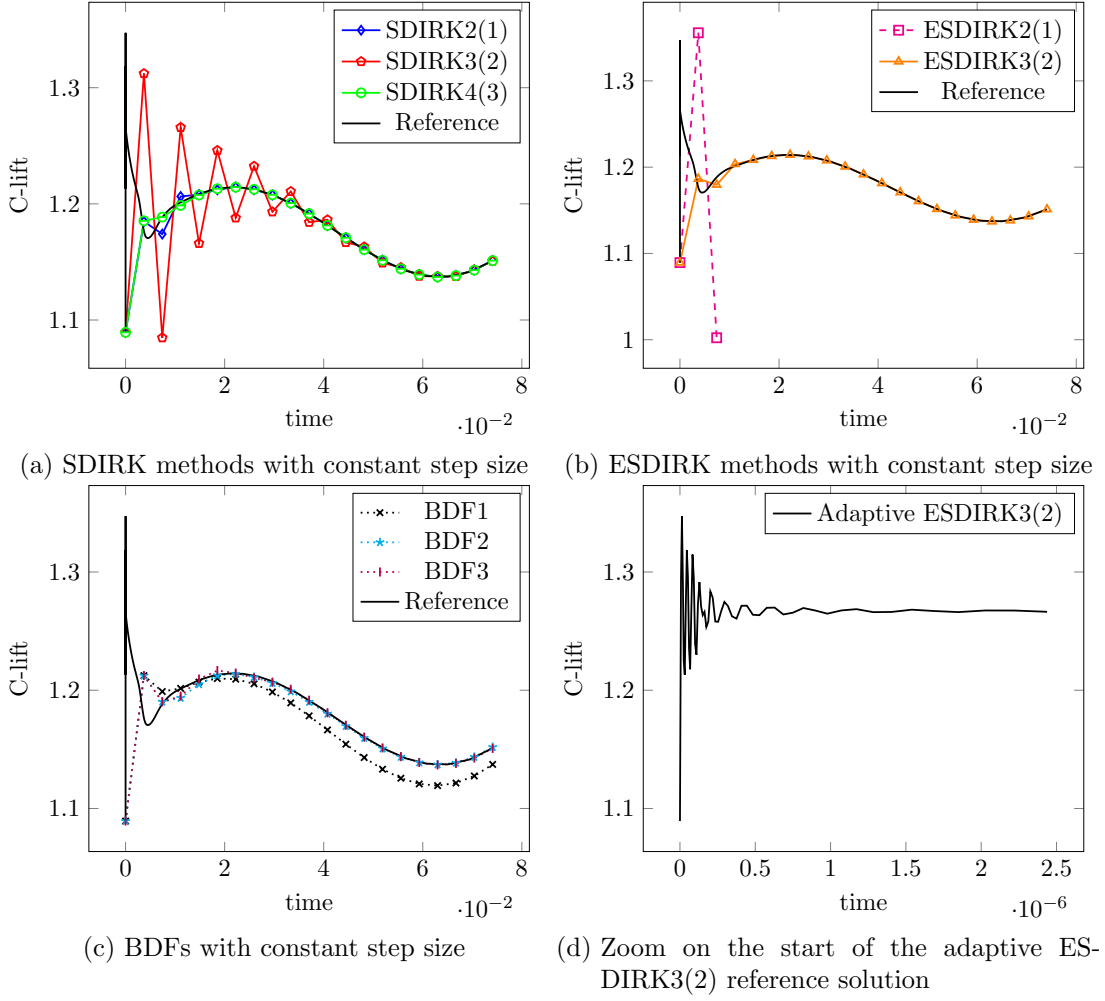


Figure 7.21: NACA0012 pitching oscillation startup from steady solution

lations. The simulation with the ESDIRK2(1) method with stability function $R(z) = 1$ for $z \rightarrow \infty$ even crashes after 2 steps. For this kind of application, L-stable methods or at least methods with $R(z) < 1$ for $z \rightarrow \infty$ are recommended. The simulations using BDFs shown in Figure 7.21c show a nice behavior, since the methods are L-stable or $L(\alpha)$ -stable in case of BDF3. Note that BDF1 was used as startup for BDF2 and BDF3 reducing the theoretical order of BDF3 to 2 (cf. Section 7.1).

Influence of method coefficients on solution of nonlinear algebraic systems

The coefficients of a DIRK method can have a significant impact on the difficulty to solve the resulting nonlinear systems of algebraic equations. Smaller (positive) elements

on the diagonal of the coefficient matrix $\mathcal{A}$ result in larger diagonal elements of the preconditioner (4.7) increasing the diagonal dominance. Depending on the solution method, this might be advantageous. Furthermore, there is an influence of the previous stages in the computation of $\tilde{\mathbf{W}}$ in (4.4). This can be seen in the first time step of the NACA0012 pitching oscillation when starting from the steady state solution. Figure 7.22 shows convergence histories of several DIRK methods compared to BDF1. The slow

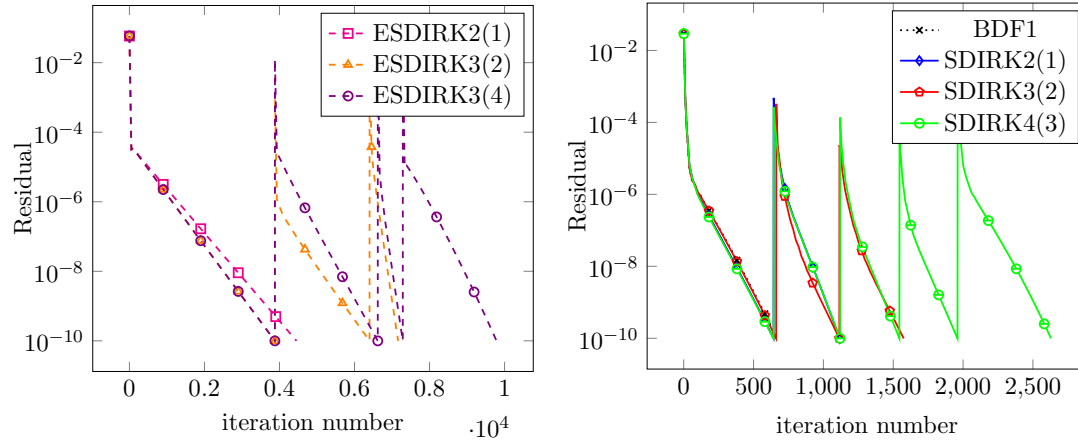


Figure 7.22: Convergence plot for first time step of NACA0012 pitching oscillation with different DIRK methods, CFL 100...1000, starting from the steady state solution. The plots show the residuals for the stage solutions of the DIRK methods. Hence, the convergence for several systems of nonlinear equations is shown depending on the specific method.

convergence is especially evident in the case of the ESDIRK methods, due to the explicit first stage. For subsequent time steps, the convergence is faster for the ESDIRK3(2) and ESDIRK3(4) schemes (Figure 7.23) since the strongly varying part of the solution is already damped a lot by one step of the L-stable methods. For the ESDIRK2(1) method the problems continue due to the missing L-stability, eventually leading to a crash. The convergence speed can be increased by taking larger CFL numbers. Figure 7.24 shows the convergence for 10 times larger CFL numbers. As can be seen, only the BDF1, SDIRK3(2), and SDIRK4(3) methods converge with these settings. For SDIRK2(1) convergence can be achieved in the first stage, but the simulation crashes in the second stage. All ESDIRK methods do not converge in the second stage (first implicit stage). Note that these convergence problems are not encountered when starting the test case after one initial period with the same settings.

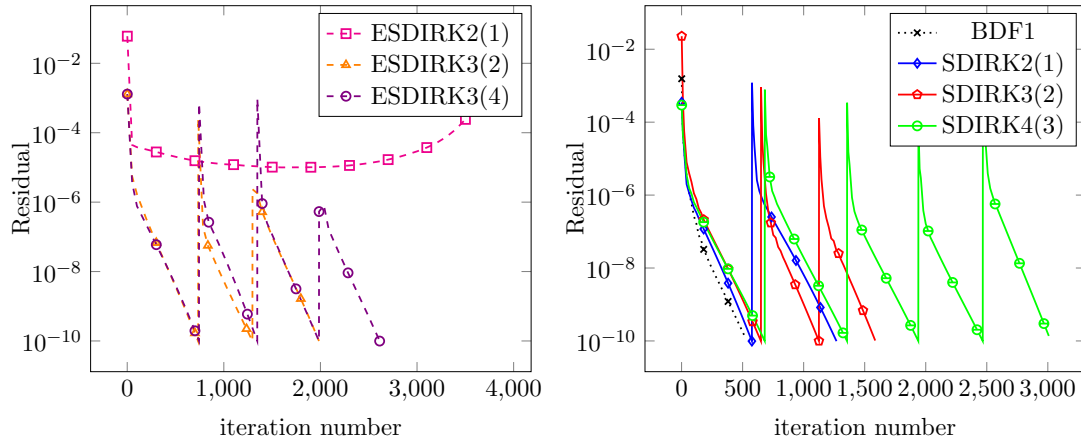


Figure 7.23: Convergence plot for second time step of NACA0012 pitching oscillation with different DIRK methods, CFL 100...1000, starting from the steady state solution. The plots show the residuals for the stage solutions of the DIRK methods. Hence, the convergence for several systems of nonlinear equations is shown depending on the specific method.

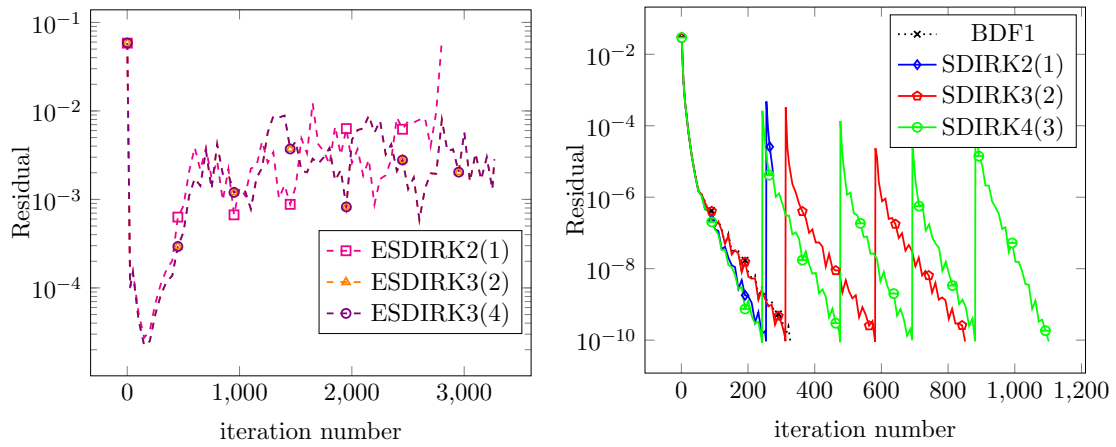


Figure 7.24: Convergence plot for first time step of NACA0012 pitching oscillation with different DIRK methods, CFL 1000...10000, starting from the steady state solution. The plots show the residuals for the stage solutions of the DIRK methods. Hence, the convergence for several systems of nonlinear equations is shown depending on the specific method.

7.6 Comparison of BDFs and (adaptive) DIRK methods

In this Section, the DIRK methods and BDFs used in this thesis are compared regarding their efficiency. The DIRK methods are considered using constant and adaptive time steps.

NACA0012 gust encounter

To compare the efficiency for the NACA0012 gust encounter test case, several simulations with different time step sizes or tolerances in case of adaptive simulations have been computed. The temporal accuracy is plotted over the required computational work in Figure 7.25. As can be seen, the SDIRK3(2) method, which is the only DIRK method which is not stiffly accurate, is the worst method in terms of efficiency (disregarding BDF1). The large error for the adaptive simulation with SDIRK3(2) is due to the problems shown in Figure 7.20 and explained in Chapter 7.5. Regarding the efficiency of the other methods, ESDIRK3(2), ESDIRK2(1), and SDIRK2(1) are significantly more efficient than SDIRK4(3) and ESDIRK3(4), especially in the case of adaptive time stepping (Figure 7.25a). For constant time steps, BDF3 is the most efficient except for low accuracy simulations where ESDIRK2(1) is slightly ahead (Figure 7.25c). BDF2 on the other hand lags slightly behind compared to ESDIRK3(2), ESDIRK2(1), and SDIRK2(1) with constant time steps. Using adaptive time stepping leads to a significant speed up for this test case (Figure 7.25d), such that the adaptive ESDIRK3(2), ESDIRK2(1), and SDIRK2(1) methods are more efficient than BDF3. However, it should be noted that the difference in efficiency when comparing constant to adaptive time steps depends on the physical simulation time for this test case. A longer physical simulation time favors the adaptive time stepping due to the large time steps at the end of the simulation.

Furthermore, it is noticeable, that the SDIRK4(3) method shows an irregular behavior, where the error increases with decreasing step sizes at some point. Detailed investigations on this topic have not been done here. The choice of the error measure might be a reason, since the lift coefficient is not a norm and cancellation of errors can happen. However, using other measures did not entirely solve the convergence problems for this method (though the problems were not as prominent).

NACA0012 pitching oscillation

For the NACA0012 pitching oscillation, it can be observed (Figure 7.26) that the SDIRK3(2) method, which is the only method that is not stiffly accurate, is much less efficient than

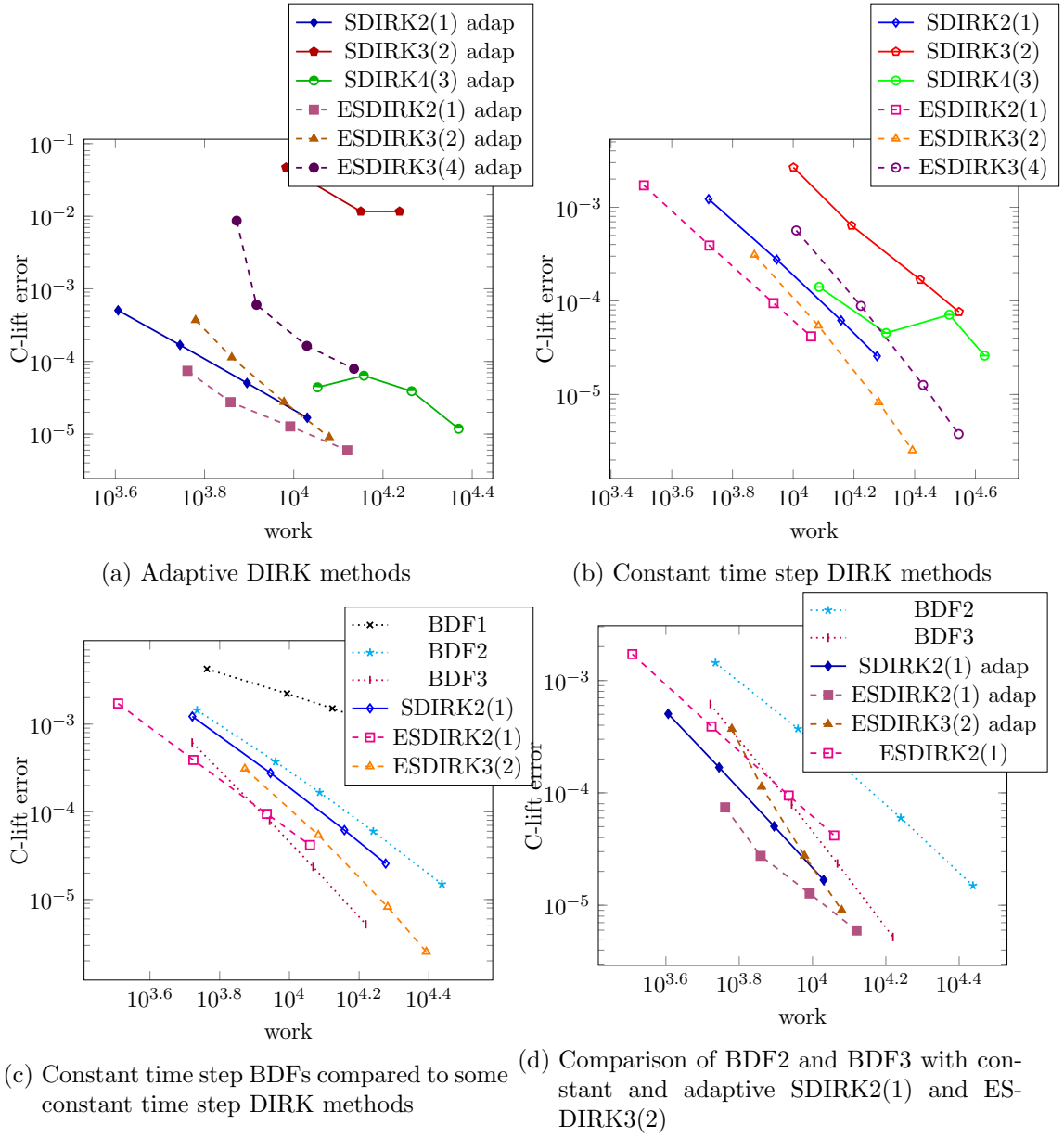


Figure 7.25: Work-C-lift-error plots for the inviscid NACA0012 gust encounter simulation using different DIRK methods. The adaptation was done using the PI controller (5.38) with limiter (Algorithm 4). The work is given as the total number of algebraic solver iterations, which is approximately proportional to the computational time for the applications presented here.

all other DIRK methods for both constant and adaptive time stepping. The second order methods SDIRK2(1) and ESDIRK2(1) show almost the same efficiency in the case of adaptive time stepping with ESDIRK2(1) being slightly ahead for constant time steps. Regarding the higher order methods, the ESDIRK3(2) method is slightly more efficient than ESDIRK3(4) and SDIRK4(3). Furthermore, adaptive time stepping does not increase the efficiency for this test case, since the variations in the time step size are minor (cf. Figure 7.13 showing the evolution of the time step size for this test case). The BDFs are less efficient than the best DIRK methods of the same order since significantly more time steps are required for the same accuracy. Especially in the case of the error in C-lift, the SDIRK4(3) method has a irregular behavior, even leading to an increased error for a smaller time step at one point in Figure 7.26b. Since this behavior can not be observed in case of the error in C-drag, it probably is due to some error cancellation in the computation of the lift coefficient. However further investigations on this topic have not been done here.

Transport aircraft gust encounter

Only few different settings have been used to simulate the transport aircraft gust encounter due to the high computational cost. Table 7.13 shows a comparison of the computational efficiency of the SDIRK2(1) method compared to BDF2. As can be seen, the SDIRK2(1) method is more efficient than BDF2 to reach the same error in the lift coefficient (computed in an approximate L_2 -norm over the entire simulation). This is due to the fact that less time steps are required for the SDIRK2(1) method to get the same accuracy. When using time step size adaptation, the required number of time steps for the same accuracy is further reduced (approximately halved). However, the computational effort is almost the same as in the case of constant time stepping. This is due to two reasons:

1. For larger time steps the solver for the nonlinear algebraic systems needs more iterations.
2. Some time steps (4 in this case) have to be recomputed since they have been rejected. Here, this accounted for about 8% of the computational time.

Regarding the first point, note that this difference in the computational effort to solve the nonlinear systems might be reduced by future improvements in the nonlinear solver efficiency. This would favour adaptive time stepping. Note that convergence problems have been encountered in the simulations using time adaptive SDIRK2(1) method and

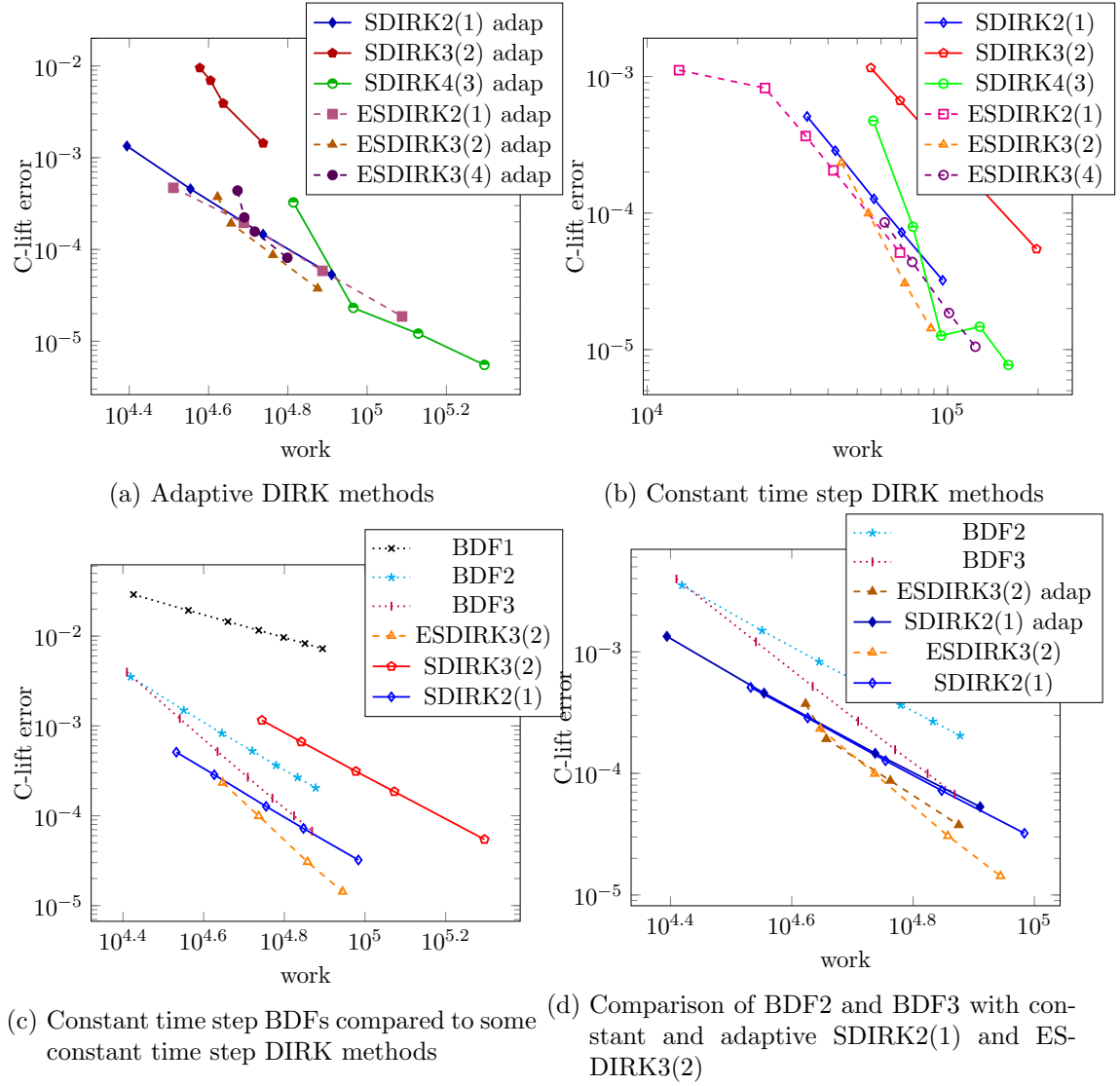


Figure 7.26: Work-C-lift-error plots for the turbulent NACA0012 pitching oscillation simulation using different DIRK methods. The adaptation was done using the PI controller (5.38) with limiter (Algorithm 4). The work is given as the total number of algebraic solver iterations, which is approximately proportional to the computational time for the applications presented here.

BDF2 with 200 steps, where 1 and 4 iterative solution processes failed to converge in 2000 iterations. In the former case, the step was rejected and recomputed with a smaller time step. In the latter case, the result after 2000 iterations was accepted. In all these cases, the residual was stuck at about $4 \cdot 10^{-5}$. On the other hand, the iterative solution process typically took less than 500 iterations when convergence to $1 \cdot 10^{-8}$ was achieved.

method	time steps (+rejections)	Error C-lift	rel. CPU time
SDIRK2(1) adap	83 (+4)	$5.5 \cdot 10^{-5}$	1.0
SDIRK2(1) const	166	$6.7 \cdot 10^{-5}$	1.0
SDIRK2(1) const	200	$4.6 \cdot 10^{-5}$	1.09
BDF2	200	$3.6 \cdot 10^{-4}$	0.86
BDF2	500	$5.6 \cdot 10^{-5}$	1.28

Table 7.13: CPU times with adaptive SDIRK and constant time step BDF method for the transport aircraft gust encounter

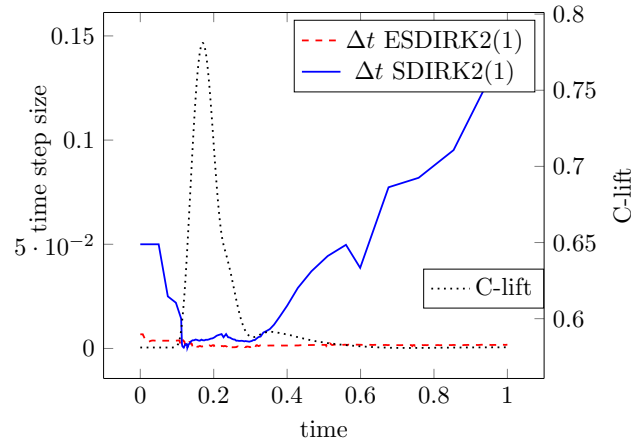


Figure 7.27: Evolution of time step size and lift coefficient for the transport aircraft gust encounter test case using ESDIRK2(1) and SDIRK2(1)

It can be concluded that for the simulations presented here, the ESDIRK3(2), ESDIRK2(1), and SDIRK2(1) methods are typically more efficient than the other investigated DIRK methods. Even though the ESDIRK2(1) method is slightly ahead of the SDIRK2(1) method, the latter is recommended as standard, since it is L-stable (see the investigations on L-stability in Section 7.5). Furthermore, the ESDIRK2(1) method with adaptive time stepping has problems in case of the transport aircraft gust encounter test case. The method fails to increase the time step after the gust encounter (Figure 7.27). This might be due to the missing L-stability in combination with the not A-stable em-

bedded method. Using constant time steps, the ESDIRK2(1) method works fine for this test case. Regarding the investigated multistep methods it can be concluded that the BDF3 method is competitive and the BDF2 lags slightly behind. Note again that the BDF3 method has the disadvantage of not being A-stable. Even though this was not a problem for the investigated cases, this might get problematic for other applications or settings.

CHAPTER 8

Conclusion

In this thesis, the numerical solution of unsteady aerodynamic flow problems using a time step size adaptation algorithm has been discussed. The focus was on applications with an induced unsteadiness, especially the pitching oscillation of an airfoil as well as several gust encounter applications. Since explicit methods require very small time steps for turbulent test cases, the main focus was on DIRK methods suitable for the solution of stiff problems. Methods with different properties have been discussed and applied to aerodynamic test cases leading to recommendations for specific applications.

Furthermore, a new time step size limiter has been derived and tested for gust encounter applications. In the presented test cases, the limiter was able to prevent repeated time step rejections, reducing the total number of rejected steps and thus lowering the computational effort for the gust encounter simulations.

In addition to the new time step size limiter, several existing step size control algorithms, namely the integral controller, proportional integral controller, predictive controller, and a digital filter have been investigated. It can be concluded that the optimal controller choice depends on the specific application. While the predictive controller showed the best results in terms of computational efficiency for the inviscid gust encounter simulations, it was unsuitable for the considered pitching oscillation. However, the differences in computational time were minor and sometimes small changes in other parameters, for example the initial time step size, changed the outcome of the investigation.

Moreover, several different error norms or functionals, mapping the estimated error in the flow solution to a scalar value required by the step size control algorithm, have been investigated. The L_2 -norm, L_∞ -norm, scaled 2-norm and the error in the lift coefficient have been considered. For gust encounter simulations it was observed, that using the L_2 -norm or L_∞ -norm results in a too slow increase of the time step size after the gust

passed the structure. Using the lift coefficient as error functional resulted in the fastest increase of the time step size. This results in a reduced number of necessary time steps but also in the most rejected time steps. Therefore, the use of the 2-norm, which has a reasonably fast increase of the time step size paired with a smooth step size evolution is recommended as standard procedure. Though different applications might favor other choices.

Regarding the solution of the arising systems of algebraic equations, different solution methods available in TAU have been investigated. It has been observed, that using the LSBGS method in a simplified newton type iteration without multigrid was the fastest method for the turbulent NACA0012 pitching oscillation test case. This method is recommended as standard procedure. Only in cases where convergence problems are encountered, like for the transport aircraft gust encounter, the LU-SGS method might be preferred. However, it should be noted, that the reasons for the convergence issues have not been investigated and might well be due to the specific implementation in TAU rather than the method itself.

Additionally, several stage value predictors have been discussed. The numerical test showed that extrapolation has the potential to speed up the solution process. Though, it also resulted in diverging solution processes in some cases. Taking the solution of a previous stage (if available) for initialization has shown to be more reliable and is therefore recommended. It is also typically faster than using the solution of the last time step for the investigated methods.

Finally, the newly implemented DIRK methods have been compared to the BDFs which were already available in TAU. It was observed, that the error of the DIRK methods was typically smaller than that of the BDFs for the same time step size. Therefore, larger time steps could be taken, which compensated for the requirement of solving several systems of algebraic equations in each time step in case of most DIRK methods. For the presented test cases, some DIRK methods were more efficient than the BDFs. Furthermore, the influence of time step size adaptation regarding the computational effort of the simulation was investigated. As expected, the results were heavily dependent on the specific test case. For the pitching oscillation the time step sizes did not change much and using adaptive time stepping did not increase the efficiency. The gust encounter simulations however favored time step size adaptation, since significantly smaller time steps were taken during the gust encounter than before or after. However, it was shown, that even though the required number of time steps for adaptive time stepping was much lower than for constant time stepping (e.g. approximately halved for the transport aircraft gust encounter), the gain in computational time was much smaller

since the solution algorithms for the nonlinear systems of equations require significantly more computational work for larger time step sizes. Changes of the solution algorithm or of the stage value predictor resulted had a larger impact on the computational efficiency. Nevertheless, using adaptive time stepping is recommended for gust encounter simulations, as besides the advantage in computational efficiency it might also lead to less trial and error in finding an appropriate time step size. Due to reasons of efficiency and reliability, the SDIRK2(1) and ESDIRK3(2) method are recommended as standard for the applications presented here.

Further research is necessary to get a better truncation criterion for the solution of the nonlinear algebraic systems. Ideally, it should be coupled directly to the tolerance of the time step size adaptation. Investigating the potential of Rosenbrock-Wanner schemes for complex aerodynamic applications could be a topic of further research. While promising results have been shown using these methods for several applications like laminar flow simulations, it is to be investigated whether they are suitable for complex turbulent flow simulations.

Acronyms

BDF	backward difference formula
CFD	computational fluid dynamics
CFL	Courant-Friedrichs-Lewy
DF	digital filter
DIRK	diagonally implicit Runge-Kutta
DLR	Deutsches Zentrum für Luft- und Raumfahrt e.V.
ERK	explicit Runge-Kutta
ESDIRK	explicit first stage singly diagonal implicit Runge-Kutta
I	integral
IVP	initial value problem
LSBGS	Line Symmetric Block Gauss-Seidel
LU-SGS	lower-upper symmetric Gauss-Seidel
ODE	ordinary differential equation
P	predictive
PDE	partial differential equation
PI	proportional-integral
PID	proportional-integral-differential
RANS	Reynolds-Averaged-Navier-Stokes

SDIRK	singly diagonal implicit Runge-Kutta
SDM	standard dynamic model
SIRK	singly implicit Runge-Kutta
TAU	turbulent adaptive unstructured

List of Figures

1.1	Overview of the time step size adaptation algorithm	8
4.1	Extrapolation in case of a moving shock, a linear extrapolation of the solution at $t=2$ from solutions at $t=0$ and $t=1$ results in the solution in Figure 4.1b instead of the $t=2$ curve in Figure 4.1a	52
6.1	Ehrenfried vortex test case: Grid and initialization	76
6.2	Primary grid and density distribution snapshot of Kármán vortex street test case	77
6.3	Primary grid, initial density distribution, Mach number, and streamtraces for the laminar NACA0012 test case	79
6.4	Primary grid, initial density distribution, and evolution of lift coefficient for the NACA0012 pitching oscillation test case	80
6.5	Primary grid, initial density distribution, and evolution of lift coefficient for the NACA0012 gust encounter test case	82
6.6	Primary grid, initial density distribution, and evolution of lift coefficient for the SDM gust encounter test case	83
6.7	Primary grid, initial density distribution, and evolution of lift coefficient for the transport aircraft gust encounter test case	85
7.1	Convergence study of the implemented Runge-Kutta methods and BDFs for the Ehrenfried test case	89
7.2	Temporal density error distribution for the Ehrenfried vortex test case using SDIRK4(3) with a step size of $1 \cdot 10^{-5}$, whole flow field and zoom at the vortex.	90
7.3	Convergence study of the implemented Runge-Kutta methods and BDFs for the Kármán vortex street test case	91

7.4	Time step size evolution for a NACA0012 pitching oscillation for different grids and tolerances using the ERK5(4) Method 3.26	93
7.5	Evolution of lift coefficient and time step size for a NACA0012 pitching oscillation for different grids using the ESDIRK3(2) Method 3.31	94
7.6	Evolution of time step size for a NACA0012 pitching oscillation for different tolerances using the ESDIRK3(2) Method 3.31	94
7.7	Evolution of time step size and lift coefficient for a time adaptive computation of a steady laminar NACA0012 test case compared to a standard steady state simulation	96
7.8	Residual plot for a time adaptive computation of a steady laminar NACA0012 test case (left) and a standard steady state simulation (right).	96
7.9	Effect of new limiter with integral control for a gust encounter of a NACA0012 wing profile using the 2-norm as error functional	99
7.10	Effect of new limiter with predictive control for a gust encounter of a NACA0012 wing profile using the 2-norm as error functional	99
7.11	Evolution of time step size for a gust encounter of a NACA0012 wing profile using different controllers and the 2-norm as error functional . . .	100
7.12	Evolution of time step size for a gust encounter of a NACA0012 wing profile using different controllers in combination with the limiter (Algorithm 4) and the lift coefficient as error functional using an initial step size of 0.015s	101
7.13	Time step size evolution for different controllers for the NACA0012 pitching oscillation using Algorithm 4	106
7.14	Time step size evolution for different error functionals for the NACA0012 gust encounter simulation for similar levels of accuracy (cf. Table 7.9) using the predictive controller with limiter.	108
7.15	Error-tolerance and error-work plots for the NACA0012 gust encounter simulation using different norms or functionals for error estimation. The adaptation was done using the predictive controller (5.43) with limiter (Algorithm 4).	108
7.16	Error-tolerance and error-work plots for the NACA0012 gust encounter simulation using different norms or functionals for error estimation. The adaptation was done using the PI controller (5.38) with limiter (Algorithm 4).	109

7.17	Error-tolerance and error-work plots for the turbulent NACA0012 pitching oscillation simulation using different norms or functionals for error estimation. The adaptation was done using the PI controller (5.38) with limiter (Algorithm 4).	109
7.18	Convergence plot for first time step of NACA0012 pitching oscillation with BDF1 using different solver settings as described in Table 7.10	111
7.19	C-lift for NACA0012 pitching oscillation with 100 steps per period. Position of crash using extrapolation	113
7.20	Problem when using SDIRK3(2) Method 3.28 with time step adaptation for the simulation of a gust encounter of the NACA0012 airfoil profile. Mayor issues occur at the beginning of the gust encounter.	115
7.21	NACA0012 pitching oscillation startup from steady solution	117
7.22	Convergence plot for first time step of NACA0012 pitching oscillation with different DIRK methods, CFL 100 . . . 1000, starting from the steady state solution. The plots show the residuals for the stage solutions of the DIRK methods. Hence, the convergence for several systems of nonlinear equations is shown depending on the specific method.	118
7.23	Convergence plot for second time step of NACA0012 pitching oscillation with different DIRK methods, CFL 100 . . . 1000, starting from the steady state solution. The plots show the residuals for the stage solutions of the DIRK methods. Hence, the convergence for several systems of nonlinear equations is shown depending on the specific method.	119
7.24	Convergence plot for first time step of NACA0012 pitching oscillation with different DIRK methods, CFL 1000 . . . 10000, starting from the steady state solution. The plots show the residuals for the stage solutions of the DIRK methods. Hence, the convergence for several systems of nonlinear equations is shown depending on the specific method.	119
7.25	Work-C-lift-error plots for the inviscid NACA0012 gust encounter simulation using different DIRK methods. The adaptation was done using the PI controller (5.38) with limiter (Algorithm 4). The work is given as the total number of algebraic solver iterations, which is approximately proportional to the computational time for the applications presented here.	121

-
- 7.26 Work-C-lift-error plots for the turbulent NACA0012 pitching oscillation simulation using different DIRK methods. The adaptation was done using the PI controller (5.38) with limiter (Algorithm 4). The work is given as the total number of algebraic solver iterations, which is approximately proportional to the computational time for the applications presented here. 123
- 7.27 Evolution of time step size and lift coefficient for the transport aircraft gust encounter test case using ESDIRK2(1) and SDIRK2(1) 124

List of Tables

3.1	Coefficients of BDF methods up to order 3	39
3.2	Properties of investigated Runge-Kutta methods, method for error estimation in parenthesis	44
7.1	Comparison of different adaptation algorithms for a NACA0012 gust encounter with <i>error per time step</i> control and the scaled 2-norm as error functional using an initial step size of 0.015s. Compared are the standard integral (I) controller, the Proportional Integral (PI) controller, the Predictive (P) controller and the Digital Filter (DF). The CPU time is computed relative to the CPU time of P Alg 4.	97
7.2	Comparison of different adaptation algorithms for a NACA0012 gust encounter with <i>error per unit time step</i> control and the scaled 2-norm as error functional using an initial step size of 0.015s. The CPU time is computed relative to the CPU time of P Alg 4 in Table 7.1.	98
7.3	Comparison of different adaptation algorithms for a NACA0012 gust encounter with error per time step control and the lift coefficient as error functional using an initial step size of 0.015s. The CPU time is computed relative to the CPU time of P Alg 4 in Table 7.1.	102
7.4	Comparison of different adaptation algorithms for a NACA0012 gust encounter with error per time step control and the scaled 2-norm as error functional averaged over multiple simulations using different initial step sizes.	103
7.5	Comparison of different adaptation algorithms for a NACA0012 gust encounter with error per time step control and the lift coefficient as error functional averaged over multiple simulations using different initial step sizes. The CPU time is computed relative to the CPU time of P Alg 4 in Table 7.4.	103

7.6	Comparison of different adaptation algorithms for a gust encounter of the SDM configuration using error per time step control, the 2-norm as error norm and a tolerance of $1 \cdot 10^{-5}$	104
7.7	Comparison of different adaptation algorithms for a gust encounter of the SDM configuration using error per time step control, the 2-norm as error norm and a tolerance of $5 \cdot 10^{-5}$	105
7.8	Comparison of different adaptation algorithms for the NACA0012 pitching oscillation test case. The CPU time is computed relative to the CPU time of I Alg 4 in Table 7.8.	105
7.9	Comparison of different error functionals for the NACA0012 gust encounter simulation. The CPU time is computed relative to the CPU time when using the 2-norm.	107
7.10	Investigated solution algorithm settings and CPU time for the NACA0012 pitching oscillation test case	111
7.11	relative computational time for different stage value predictors for the NACA0012 pitching oscillation test case for 100 and 200 steps per period (spp)	113
7.12	relative computational time for different stage value predictors for the transport aircraft gust encounter test case for $\Delta t = 0.005s$ and $\Delta t = 0.0025s$	114
7.13	CPU times with adaptive SDIRK and constant time step BDF method for the transport aircraft gust encounter	124

Bibliography

- [1] R. Alexander. Diagonally implicit Runge-Kutta methods for stiff ODEs. *SIAM Journal of Numerical Analysis*, 14:1006–1021, 1977.
- [2] S. Allmaras, F. Johnson, and P. Spalart. Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. In *ICCFD7-1902, Seventh International Conference on Computational Fluid Dynamics, Big Island, Hawaii*, 2012.
- [3] G. Ashcroft, K. Heitkamp, and E. Kgeler. High-order accurate implicit Runge-Kutta schemes for the simulation of unsteady flow phenomena in turbomachinery. 2010.
- [4] P. Birken. Numerical methods for the unsteady compressible navier-stokes equations. Habilitation thesis, University of Kassel, 2012.
- [5] P. Birken, G. Gassner, M. Haas, and C. Munz. Efficient time integration for Discontinuous Galerkin method for the unsteady 3D Navier-Stokes equations. In *European Congress on Computational Methods and Applied Sciences and Engineering (EC-COMAS 2012), number Eccomas*, 2012.
- [6] J. Blazek. *Computational Fluid Dynamics: Principles and Applications*. Elsevier, 2001.
- [7] D. Blom, H. Bijl, P. Birken, A. Meister, and A. van Zuijlen. Rosenbrock time integration for unsteady flow simulations. In *Coupled Problems 2013: 5th International Conference on Computational Methods for Coupled Problems in Science and Engineering*. CIMNE, june 2013.
- [8] P. Bogacki and L. Shampine. A 3(2) pair of Runge-Kutta formulas. *Applied Mathematics Letters*, 2(4):321–325, 1989.

- [9] K. Brenan, S. Campbell, and L. Petzold. *Numerical solution of initial-value problems in differential-algebraic equations*. Society for Industrial and Applied Mathematics, 1996.
- [10] K. Burrage. A special family of Runge-Kutta methods for solving stiff differential equations. *BIT Numerical Mathematics*, 18(1):22–41, 1978.
- [11] K. Burrage, W. Hundsdorfer, and J. Verwer. A study of B-convergence of Runge-Kutta methods. *Computing*, 36(1):17–34, 1986.
- [12] J. Butcher. Implicit Runge-Kutta processes. *Mathematics of computation*, 18(85):50–64, 1964.
- [13] J. Butcher. On the implementation of implicit Runge-Kutta methods. *BIT Numerical Mathematics*, 16(3):237–240, 1976.
- [14] J. Butcher. A transformed implicit Runge-Kutta method. *Journal of the ACM*, 26(4):731–738, 1979.
- [15] J. Butcher. A history of Runge-Kutta methods. *Applied Numerical Mathematics*, 20(3):247–260, 1996.
- [16] J. Butcher. *Numerical methods for ordinary differential equations*. Wiley Online Library, 2008.
- [17] J. Butcher and D. Chen. A new type of singly-implicit Runge-Kutta method. *Applied Numerical Mathematics*, 34(2):179–188, 2000.
- [18] M. Calvo, T. Grande, and R. Grigorieff. On the zero stability of the variable order variable stepsize BDF-formulas. *Numerische Mathematik*, 57(1):39–50, 1990.
- [19] M. Calvo, J. Montijano, and L. Rández. A_0 -stability of variable stepsize BDF methods. *Journal of computational and applied mathematics*, 45(1):29–39, 1993.
- [20] M. Carpenter, C. Kennedy, H. Bijl, S. Viken, and V. Vatsa. Fourth-order Runge-Kutta schemes for fluid mechanics applications. *Journal of Scientific Computing*, 25(1):157–194, 2005.
- [21] M. Carpenter, S. Viken, and E. Nielsen. The efficiency of high order temporal schemes. *AIAA Paper*, 86, 2003.
- [22] G. Cooper and A. Sayfy. Semiexplicit A-stable Runge-Kutta methods. *Mathematics of computation*, 33(146):541–556, 1979.

- [23] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 43, pages 50–67, 1947.
- [24] K. Dekker and J. Verwer. *Stability of Runge-Kutta methods for stiff nonlinear differential equations*. North-Holland Amsterdam, 1984.
- [25] P. Deuffhard and F. Bornemann. *Numerische Mathematik II*. de Gruyter, 1994.
- [26] J. Dormand and P. Prince. A family of embedded Runge-Kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.
- [27] K. Ehrenfried. Ein Finite-Volumen-Verfahren zur Berechnung von instationären, transsonischen Strömungen mit Wirbeln. Technical report, Institut für Strömungsmechanik Göttingen, Deutsche Forschungsanstalt für Luft- und Raumfahrt e. V., 1995. Forschungsbericht 94-33.
- [28] P. Ellsiepen. *Zeit- und ortsadaptive Verfahren angewandt auf Mehrphasenprobleme poröser Medien*. PhD thesis, University of Stuttgart, 1999.
- [29] E. Fehlberg. Klassische Runge-Kutta-Formeln fünfter und siebenter Ordnung mit Schrittweiten-Kontrolle. *Computing*, 4:93–106, 1969.
- [30] E. Fehlberg. Klassische Runge-Kutta-Nyström-Formeln mit Schrittweiten-Kontrolle für Differentialgleichungen. *Computing*, 10(4):305–315, 1972.
- [31] B. Flynt and D. Mavriplis. Discrete adjoint based adaptive error control in unsteady flow problems. *AIAA Paper*, 2012.
- [32] R. Frank, J. Schneid, and C. W. Ueberhuber. The concept of B-convergence. *SIAM Journal on Numerical Analysis*, 18:753–780, 1981.
- [33] C. Gear. *Numerical initial value problems in ordinary differential equations*. Prentice Hall PTR, 1971.
- [34] K. Gustafsson. Control theoretic techniques for stepsize selection in explicit Runge-Kutta methods. *ACM Transactions on Mathematical Software*, 17(4):533–554, 1991.
- [35] K. Gustafsson, M. Lundh, and G. Söderlind. A PI stepsize control for the numerical solution of ordinary differential equations. *BIT Numerical Mathematics*, 28(2):270–287, 1988.

- [36] E. Hairer, S. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems*. Springer, 1993.
- [37] E. Hairer and G. Wanner. *Solving ordinary differential equations II. Stiff and Differential-Algebraic Problems*. Springer, 1991.
- [38] G. Hall. Equilibrium states of Runge-Kutta schemes. *ACM Transactions on Mathematical Software*, 11(3):289–301, 1985.
- [39] G. Hall. Equilibrium states of Runge-Kutta schemes: part II. *ACM Transactions on Mathematical Software*, 12(3):183–192, 1986.
- [40] R. Heinrich and L. Reimer. Comparison of different approaches for gust modeling in the CFD code TAU. In *In proceedings of International Forum on Aeroelasticity & Structural Dynamics*, 2013.
- [41] R. Heinrich, L. Reimer, and A. Michler. Multidisciplinary simulation of maneuvering aircraft interacting with atmospheric effects using the DLR TAU code. In *RTO AVT-189 Specialists Meeting on Assessment of Stability and Control Prediction Methods for Air and Sea Vehicles, Portsmouth West, UK*, 2011.
- [42] D. Higham and G. Hall. Embedded Runge-Kutta formulae with stable equilibrium states. *Journal of computational and applied mathematics*, 29(1):25–33, 1990.
- [43] <http://turbmodels.larc.nasa.gov>. Turbulence Modeling Resource Website, 2013.
- [44] W. Hundsdorfer and M. Spijker. A note on B-stability of Runge-Kutta methods. *Numerische Mathematik*, 36(3):319–331, 1980.
- [45] V. John and J. Rang. Adaptive time step control for the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 199(9-12):514–524, 2010.
- [46] P. Kelleners. Accurate Simulation of Flows with Convected Vortices: Cartesian TAU. In *Computational Flight Testing*, volume 123 of *Notes on numerical fluid mechanics and multidisciplinary*, pages 17–25. Springer, 2013.
- [47] J. Kuntzmann. Neuere Entwicklungen der Methoden von Runge und Kutta. *Zeitschrift für Angewandte Mathematik und Mechanik*, 41:T28–T31, 1961.
- [48] W. Kutta. Beitrag zur näherungsweise Integration totaler Differentialgleichungen. *Zeitschr. für Math. u. Phys.*, 46:435–453, 1901.

- [49] A. Kværnø. Singly diagonally implicit Runge-Kutta methods with an explicit first stage. *BIT Numerical Mathematics*, 44(3):489–502, 2004.
- [50] S. Langer. Investigation and application of point implicit Runge-Kutta methods to inviscid flow problems. *International Journal for Numerical Methods in Fluids*, 69(2):332–352, 2012.
- [51] S. Langer. Application of a line implicit method to fully coupled system of equations for turbulent flow problems. *International Journal of Computational Fluid Dynamics*, 27(3):131–150, 2013.
- [52] S. Langer. Agglomeration multigrid methods with implicit Runge-Kutta smoothers applied to aerodynamic simulations on unstructured grids. *Journal of Computational Physics*, 277:72–100, 2014.
- [53] S. Langer, A. Schwöppe, and N. Kroll. The DLR Flow Solver TAU - Status and Recent Algorithmic Developments. *AIAA SciTech*, 2014.
- [54] Y. Luo and K. Fidkowski. Output-based space-time mesh adaptation for unsteady aerodynamics. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 491, 2011.
- [55] R. Merson. An operational method for the study of integration processes. *Proceedings Symposium on Data Processing*, 1957.
- [56] S. Nørsett and P. Thomsen. Embedded SDIRK-method of basic order three. *BIT Numerical Mathematics*, 24(4):634–646, 1984.
- [57] S. P. Nørsett. *Semi-explicit Runge-Kutta methods*. Dept. of Math, Univ. of Trondheim, 1974.
- [58] A. Prothero and A. Robinson. On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Mathematics of computation*, 28(125):145–162, 1974.
- [59] S. Schoenawa. *Higher-order discontinuous Galerkin discretizations of two-equation models of turbulence*. PhD thesis, 2015.
- [60] G. Söderlind. Automatic control and adaptive time-stepping. *Numerical Algorithms*, 31(1):281–310, 2002.

- [61] G. Söderlind. Digital filters in adaptive time-stepping. *ACM Transactions on Mathematical Software*, 29(1):1–26, 2003.
- [62] G. Söderlind and L. Wang. Adaptive time-stepping and computational stability. *Journal of computational and applied mathematics*, 185(2):225–243, 2006.
- [63] G. Söderlind and L. Wang. Evaluating numerical ODE/DAE methods, algorithms and software. *Journal of computational and applied mathematics*, 185(2):244 – 260, 2006.
- [64] R. Swanson and S. Langer. Steady-State Laminar Flow Solutions for NACA0012 Airfoil. *Computers and Fluids*, page to appear, 2015.
- [65] S. Yoon and A. Jameson. Lower-upper symmetric-Gauss-Seidel method for the Euler and Navier-Stokes equations. *AIAA Journal*, 26(9):1025–1026, 1988.
- [66] S. Zan et al. Wing and Fin Buffet on the Standard Dynamic Model. Technical Report RTO-TR-26, 2000. pp 361-381.

ISSN 1434-8454 ISRN

DLR-FB--2018-25