Geometric Integration on Lie Groups and its Applications in Lattice QCD

Dissertation

zur Erlangung
des akademischen Grades
eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

der
Fakultät 4 – Mathematik und Naturwissenschaften
der
Bergischen Universität Wuppertal (BUW)
vorgelegt von

Michèle Wandelt

geboren am 05.02.1978 in Oberhausen

betreut von: Prof. Dr. Michael Günther (BUW)
Prof. Dr. Francesco Knechtli (BUW)

Wuppertal, 7. Februar 2020

The PhD thesis can be quoted as follows:

urn:nbn:de:hbz:468-20210322-110757-5
[http://nbn-resolving.de/urn/resolver.pl?urn=urn%3Anbn%3Ade%3Ahbz%3A468-20210322-110757-5]

DOI: 10.25926/c8nn-p108
[https://doi.org/10.25926/c8nn-p108]

Contents

Contents I

List of Figures 1

1 Introduction 3

I Foundations 9

2 Lattice QCD and Differential Equations 11
- 2.1 Gauge Fields in Lattice QCD 12
  - 2.1.1 Lattice Gauge Fields 12
  - 2.1.2 Plaquettes, Staples and Gauge Invariance 13
- 2.2 Hamiltonian Equations of Motion 15
  - 2.2.1 The Hamiltonian 15
  - 2.2.2 Hamiltonian Equations of Motion. 17
  - 2.2.3 Derivation of the Hamiltonian Equations of Motion 18
  - 2.2.4 Structure of the Hamiltonian Equations of Motion. 23
- 2.3 Hybrid Monte Carlo Method 25
- 2.4 The Wilson Flow 29
- 2.5 Summary 32

3 Numerical Integration of Differential Equations on Lie Groups 35
- 3.1 Foundations 36
  - 3.1.1 Differential Equations on Lie Groups 36
  - 3.1.2 Numerical Integration via Local Parameterization 37
  - 3.1.3 Runge-Kutta Methods in the Abelian Case 40
- 3.2 Runge-Kutta Methods for Lie Groups 43
  - 3.2.1 Lie-Euler 44
  - 3.2.2 Crouch-Grossmann Runge-Kutta 45
  - 3.2.3 Munthe-Kaas Runge-Kutta 47
- 3.3 Summary 52

II Developments in Lie Group Methods 55

4 Runge-Kutta Methods for Lie Groups 57
- 4.1 Step Size Control 57
  - 4.1.1 Step Size Prediction for the Abelian Case 58

4.1.2 Step Size Control for Munthe-Kaas Runge-Kutta Schemes . 60
4.2 The Cayley Transform 64
4.2.1 The Cayley Transform as Local Parameterization 65
4.2.2 Numerical Integration with the Cayley Mapping 66
4.3 Summary 68

5 Geometric Numerical Integration 71
5.1 Geometric Integration 72
5.2 Störmer-Verlet or Leapfrog Method 75
5.2.1 The Abelian Case 75
5.2.2 Leapfrog for Lie Groups 77
5.2.3 Leapfrog using the Cayley Mapping 79
5.3 Symmetric Partitioned Runge-Kutta Methods 80
5.3.1 The Abelian Case 81
5.3.2 The Non-Abelian Case 82
5.4 Time-reversible Projection Schemes 89
5.4.1 Projection Schemes for the Abelian Case 90
5.4.2 Projection Schemes for the Non-Abelian Case 94
5.5 Summary 96

6 Exponential Smoothing Splines 99
6.1 Minimization Problem 99
6.2 Relation to Other Splines 100
6.3 The Exponential Smoothing Spline and its Coefficients 101
6.4 Lagrange Parameter 103
6.5 Example 104
6.6 Summary 105

III Simulation 107

7 HMC in Lattice QCD 109
7.1 HMC Simulations on Lattice Gauge Fields 109
7.1.1 HMC 109
7.1.2 Tests for Geometric Integration 113
7.2 Geometric Integration Methods 117
7.2.1 Leapfrog Method 117
7.2.2 The Cayley-Leapfrog Method 118
7.2.3 The Symmetric Partitioned Runge-Kutta Method 121
7.2.4 The Time-Reversible Projection Method 126
7.3 Summary 128

8 The Wilson Flow and Finite Temperature QCD 131
8.1 The Model 131
8.2 Step Size Control for Munthe-Kaas Runge-Kutta Methods 133
8.3 Energy Difference Method for the Critical Temperature 136
8.4 Summary 145

9 Conclusion and Outlook 147
9.1 Conclusion . . . 147
9.2 Outlook . . . 151

Acknowledgements 159

10 Appendix 161
.1 Similarity Transformations . . . 161
.2 Example: Deriving Hamiltonian Equations of Motion for SU(2) . . . 161
.3 Exponential Smoothing Spline . . . 165

List of Figures

2.1 Links in forward and backward direction 12
2.2 Plaquette 13
2.3 Lattice gauge field in 2 dimensions. 15

6.1 Example of an exponential smoothing spline. 105

7.1 Leapfrog: convergence order 119
7.2 Geometric Properties of the Leapfrog schemes. 120
7.3 Computational cost of the Leapfrog schemes 120
7.4 Time-reversibility of the symmetric partitioned MK-RK scheme . . 125
7.5 Convergence order of the SPRK scheme. 125
7.6 Computational cost of the symmetric partitioned Munthe-Kaas Runge-Kutta scheme. 126
7.7 Difference of 2 successive Hamiltonians. 128
7.8 Volume-preservation of the symmetric projection method. 129

8.1 Wilson energy computed with a Munthe-Kaas Runge-Kutta method 136
8.2 Lattice average P_L of the Polyakov loop along the Wilson flow . . . 139
8.3 Energy difference ΔE 140
8.4 Pairs $(\beta, \Delta E(\beta)$ 142
8.5 Critical values of β 142
8.6 Critical value for β 143
8.7 Integrated auto-correlation times 144

Chapter 1

Introduction

Some of the most fascinating questions in our time are:

- What is the world made of?
- What are the smallest naturally occurring particles and what sticks them together?

Quantum Chromo Dynamics (QCD) tries to find answers to these questions [25]. It is the theory of the strong interaction also called color force which is one of the elementary forces of the standard model in particle physics. The color force explains the binding between the quarks which are the smallest known particles in the subatomic range: three quarks form a proton or a neutron from which the nucleus is build. The quarks interact with help of gluons which are the carriers of the color force.

Some results of QCD cannot be described from a pertubative point of view, i.e. not from a derivation of a series in the coupling constant. For that reason, the theory of QCD is simulated via a discretization of space-time on a lattice. This scientific discipline is called Lattice Quantum Chromo Dynamics (Lattice QCD). Here, a four-dimensional hypercubic lattice is used on which the quarks and gluons are arranged: the quark fields are situated on the lattice points and the gluon fields on the links in between. The goal of the simulations is the preferably accurate computation of one or more expectation values, i.e. the statistical errors should be as small as possible. This succeeds via running a Monte Carlo simulation to generate ensembles of field configurations which implies high computational costs. Mostly, the results are obtained with the famous and widely used Hybrid Monte Carlo (HMC) method which is very costly. [21, 16]

The HMC method combines a Metropolis Monte Carlo method with a Molecular Dynamics (MD) step such that a Markov chain of configurations is produced which is the basis for the computation of the expectation values. In this context, Hamiltonian equations of motion have to be solved with a geometric numerical integration scheme, i.e., the numerical method has to be time-reversible and volume-preserving. The specialties in Lattice QCD are that the gluons are described as elements of the special unitary Lie group $SU(3, \mathbb{C})$. The quarks are described by complex vectors with a spin and color index. The color index of the quarks transforms in the fundamental representation of $SU(3)$. So, the Hamiltonian equations of motions arise in the form of differential equations on Lie groups which have to be solved during the MD step of the HMC method. In this step, geometric integration has to be used to ensure that the equilibrium distribution of the Markov chain can be reached. The geometric numerical integration of the Hamiltonian equations of motion causes the main computational cost during the simulation. This is due to the quarks since the inverse of the discretized Dirac operator appears in the action and enters the force in the MD steps. Thus, the development of new less expensive geometric integration schemes would be very helpful. [21, 16]

In Lattice QCD simulations, a second differential equation, the Wilson flow, occurs. The Wilson flow is a smoothing procedure for the gauge field and can be used to construct renormalized quantities. The Wilson flow is a differential equation on a Lie group and underlies no constraints. Thus, it can be computed using any numerical integration method on Lie groups.

The focus of this work is the development of better, i.e. less time-consuming algorithms for the numerical integration of differential equations on Lie groups. We concentrate on two main topics:

- the geometric numerical integration of the Hamiltonian equations of motion
- and the numerical integration of the Wilson flow.

Besides, we develop a method for the detection of the phase transition in finite temperature QCD. For this purpose, we refine the exponential smoothing spline.

Structure. This thesis is split into three parts covering 7 chapters as follows:

I: Foundations (chapter 1 and 2):

The first part provides the foundations of Lattice QCD and numerical integration on Lie groups in chapters 2 and 3.

In the first chapter, a brief introduction of Lattice QCD and its connection to the differential equations on Lie groups is given. It starts with a short explanation of lattice gauge fields and its most frequently used terms plaquettes and staples in section 2.1. Then, the Hamiltonian equations of motion are considered in section 2.2. Here, the Hamiltonian and its equations of motion are mentioned. Furthermore, we concentrate on the derivation and the structure of these equations. Afterwards, we turn to the computation of expectation values via the Hybrid Monte Carlo method in section 2.3 and exhibit the importance of using a time-reversible and volume-preserving numerical integration scheme for the Hamiltonian equations of motion. Moreover, the Wilson Flow is discussed in 2.4. Finally, there is a summary 2.5 of this chapter with focus on the occurring differential equations on Lie groups and an outlook to the next chapters. For the whole chapter, it is assumed that the reader has no prior knowledge of Lattice QCD.

Chapter 3 is split into two parts 3.1 and 3.2. Section 3.1 deals with the foundations of numerical integration of differential equations on Lie groups. It starts with some theory about differential equations on Lie groups and the possibility to reach a numerical solution via a local parameterization. Then, Runge-Kutta methods are introduced for the general Abelian case $\mathbb{R}^n$. The second section 3.2 focuses on Runge-Kutta methods for Lie groups. Here, some well-known Runge-Kutta methods for Lie groups (Lie-Euler, Crouch-Grossmann and Munthe-Kaas) are introduced.

II: Developments in Lie group methods (chapters 3,4 and 5):

The advancement in Lie group methods and exponential smoothing splines performed in the framework of this thesis are described in chapters 4-6 in part II.

In chapter 4, Runge-Kutta methods for Lie group problems $\dot{Y} = AY$ are developed based on the schemes described in section 3.2. First, a step size prediction for Munthe-Kaas Runge-Kutta schemes is discussed in section 4.1 based on the Abelian case. Then, the focus is put on the opportunity to use the Cayley mapping as function for the local parameterization on the Lie group in section 4.2. Here, Munthe-Kaas Runge-Kutta schemes are combined with the Cayley transform.

Chapter 5 deals with geometric integration methods on Lie groups. Here, the Hamiltonian system $\dot{Y} = AY$, $\dot{A} = F(Y)$ with a coupled Lie group / Lie algebra problem is solved by means of some geometric integration methods. After an introductory section 5.1 concerning geometric integration in general, the Leapfrog method is discussed in section 5.2. It includes the Abelian case and the non-Abelian case using the exponential function and also the Cayley mapping as two different local parameterizations. Afterwards, symmetric partitioned Runge-Kutta schemes are developed in section 5.3. Starting from the Abelian case, symmetric partitioned Munthe-Kaas Runge-Kutta schemes are discussed for three different variants. For one of these modifications, order conditions are computed. The next section 5.4 deals with time-reversible projection methods in section. Based on the Abelian case, time-reversible and projection schemes are developed and investigated for volume-preservation. Then, they are adapted to the non-Abelian case. At the end, the developed schemes are summarized with prospect on its possible usage for Lattice QCD in section 5.5.

Chapter 6 explains the idea of exponential smoothing splines. These splines joining the concepts of smoothing splines and exponential splines are needed for the determination of the critical temperature in Lattice QCD via the Wilson flow. They can be used for any data with uncertainties which have to be approximated avoiding oscillations not given in the data.

III: Simulation (chapter 6 and 7):

Finally, part III contains simulations of lattice gauge fields via a Hybrid Monthe Carlo simulation as well as of the Wilson flow in $SU(3,\mathbb{C})$ Yang-Mills theory. Here, we concentrate on the numerical results of the aforementioned developed methods. Furthermore, we investigate a phase transition in finite temperature QCD.

In chapter 7, the geometric integration methods (Cayley-Leapfrog, symmetric partitioned Munthe-Kaas Runge-Kutta schemes and the time-reversible projection method) developed during this thesis are applied on a HMC simulation producing an ensemble of lattice gauge fields. Paragraph 7.1 starts from a description of the model – a pure gauge field in Lattice QCD – and explains all steps of the implementation of a Hybrid Monte Carlo method for this model. Then, the focus is put on the geometric integration inside the Molecular Dynamics step in general. Here, the investigations of the different properties of the numerical scheme (i.e., convergence order, time-reversibility, volume-preservation, computational cost, ...) are described.

Next, the numerical results of the different methods developed in chapter 5 are presented in section 7.2. It starts with the standard Störmer-Verlet or Leapfrog method for Lie groups. This scheme and its geometric properties are well-known, in order that it is used as reference. Then, the Cayley-Leapfrog method derived in section 4.2 is investigated. In the following, the focus is put on symmetric partitioned Munthe-Kaas Runge-Kutta methods in part followed by a part on time-reversible projection methods. Also here, the chapter closes with a summary in section 7.3.

The numerical simulation of the Wilson flow is described in chapter 8. It opens with a short description of the model, a four-dimensional lattice gauge field in $SU(3)$-Yang-Mills theory, in section 8.1. Then, the step size prediction is investigated for a single configuration in paragraph 8.2 followed by section 8.3 on the determination of the critical temperature using the Wilson flow. Here, a new method for the detection of the critical temperature in finite temperature simulations is described using the difference of the temporal and spatial Wilson energy computed at a certain time point of the Wilson flow. This detection strongly depends on the usage of the exponential smoothing spline described in chapter 6.

The thesis closes with a conclusion chapter 9. Here, the thesis is summarized and it is stated which methods are suitable to be used in Lattice QCD. Also, the next possible steps are described.

Results. Parts of this thesis are already published in [60], [62], [63], [64] and [66].

The outcome of part II and III can be condensed as the development of

- embedded Munthe-Kaas Runge-Kutta schemes (section 4.1),
  applied on the Wilson flow (section 8.2), see [63],
- Munthe-Kaas Runge-Kutta schemes using the Cayley transform as local parameterization (section 4.2), see [65],
- the Störmer-Verlet scheme in Munthe-Kaas formulation with Cayley transform as local parameterization (section 5.2),
  applied on the Hamiltonian equations of motion (section 7.2.2), see [65],
- symmetric partitioned Munthe-Kaas Runge-Kutta schemes (section 5.3),
  applied on the Hamiltonian equations of motion (section 7.2.3), see [60, 62, 64],

- time-reversible projection schemes (section 5.4),
  applied on the Hamiltonian equations of motion (section 7.2.4)
- exponential smoothing splines (chapter 6),
  used for the energy difference scheme
- the energy difference scheme (section 8.3), see [66].

Part I

Foundations

Chapter 2

2 Lattice QCD and Differential Equations

Lattice QCD is the only known non-perturbative way to validate that the theory of QCD is correct. It aims at computing expectation values of some operators, for example, the mass of elementary particles. Afterwards, these values are used for comparison with experimental data in order that the theory of QCD can be validated. In Lattice QCD, space-time is discretized as a four-dimensional equidistant Euclidean lattice with lattice spacing a in order that the elementary particles, i.e. quarks and gluons, investigated in QCD are situated on the lattice. The sites of the lattice can be seen as quarks and the interconnecting lines between the quarks as gluons carrying the color force.

The calculation of an expectation value of an operator comprises a large ensemble of lattice configurations gained via the Hybrid Monte Carlo (HMC) method. In this method, differential equations have to be solved for each lattice point of a lattice configuration. Since the lattice spacing a should be quite small, lattices comprise many grid points. A typical size of a lattice used in present-day research is of size $T \times L^3 = 128 \times 48^3$ with time direction T and spatial directions L as described by Bruno et al in table 1 in [10]. This means, many differential equations have to be solved to get some expectation values. So, it is important to analyze and improve the numerical integration methods used for the differential equations appearing in Lattice QCD.

This thesis deals with the numerical integration schemes that are used to solve the differential equations occurring in Lattice QCD or, more specifically, in the SU(N) Yang-Mills theory. So, we work in the most simple frame of Lattice QCD called pure gauge theory. Here, we concentrate on the gluons, the quarks are left out. There are several introductory textbooks on SU(3) Yang-Mills theory and its simulation, for example, the books of Gattringer and Lang [21], de Grand and del Tar [16], Knechtli et al [34], Montvay and Münster [42] or Rothe [55]. In this chapter, the most important concepts described in these books are summarized as they are needed for the understanding of the simulations done in this work. First, lattice gauge fields and all expressions needed for the computation of expectation values are introduced in section 2.1. Then, there is a section 2.2 about the Hamiltonian equations of motion. These are the differential equations needed for the Hybrid Monte Carlo simulation which is explained in part 2.3. Next, the Wilson flow, a second differential equation used in the context of Lattice QCD, is introduced in section 2.4. Finally, the important aspects of this chapter are summarized in paragraph 2.5 and it is stated in which context they will occur later on.

2.1 Gauge Fields in Lattice QCD

This section opens with an introduction of lattice gauge fields in order that the model and the simulations described in the simulation part can be understood. Firstly, a field of link matrices and the structure of the lattice is explained. This is important for the insight in the implementation of the code in chapters 7 and 8. Furthermore, the terms plaquette, staples and gauge invariance are explained since they are used for the Hamiltonian equations of motion and the Wilson flow mentioned in the next paragraphs 2.2 and 2.4 as well as for the simulations.

2.1.1 Lattice Gauge Fields

In the context of Lattice QCD, a lattice gauge field is a field of "gluons" which carry the color force. The gluons are put on a d-dimensional equidistant grid with lattice spacing a – the lattice.

In general, a d-dimensional lattice has size $n_p := T \times L^{d-1}$ with time extension T and space extension L^{d-1}. Each of the directions of the lattice is labeled with the Lorentz index, which is an index of the space-time. The Lorentz index is denoted with a Greek letter μ or ν taking the values from 0 to $d-1$ labeling a specific direction. Usually, the time direction has the value 0 in lattice computations. Furthermore, the unit vector is marked with a hat in order that the unit vector in direction μ is called $\hat{\mu}$. A special point of this grid can be labeled with the space-time argument x. Related to this point, the next point in direction μ is at site $x + a\hat{\mu}$ which means that you start from site x and go a distance of length a in direction μ. [21]

The "gluons" are the interconnecting lines between the grid points in order that they are called links or, due to their shape, link matrices in this context. Mathematically, they are elements of the special unitary Lie group SU(3,C) which means they are unitary with determinant one. Link matrices are directional in order that a link from site x in direction μ is called forward link $U_{x,\mu}$. It connects the adjacent sites x and $x + a\hat{\mu}$. A link from site $x + a\hat{\mu}$ to x in the opposite direction is the inverse $U_{x,\mu}^{-1}$ of $U_{x,\mu}$. It is marked with $U_{x,\mu}^{\dagger}$ because of the unitarity $U_{x,\mu}^{-1} = U_{x,\mu}^{\dagger}$. This means, links marked with a dagger always point in a negative direction. Links with forward and backward direction are depicted in figure 2.1. [21]

x
$x + a\hat{\mu}$
$U_{x,\mu}$
x
$x + a\hat{\mu}$
$U_{x,\mu}^{\dagger} = U_{x+a\hat{\mu},-\mu}$

Figure 2.1: *Links in forward and backward direction* – Left: Link $U_{x,\mu}$ from site x in direction μ. Right: Link $U_{x,\mu}^{\dagger}$ in the reverse direction (from site $x + a\mu$ in direction $-\mu$).

A d-dimensional lattice contains a lattice gauge field of size $n_l := d \cdot T \cdot L^{d-1}$ whose elements are link matrices. A configuration $[U]$ is the set of all link variables arranged on the lattice, i.e. values and positions of the link variables are fixed. An ensemble $\{[U]\}$ of configurations $[U]$ comprises all available (i.e. already computed) configurations $[U]_1, [U]_2, \dots, [U]_n$.

2.1.2 Plaquettes, Staples and Gauge Invariance

On a lattice, there are some frequently used terms called plaquette, staples and gauge invariance. They occur in the context of the formulae used for the computations of the Hamiltonian equations of motion, the Wilson flow and the observables of interest during the Monte Carlo simulations. All these terms are defined, for example, in [16], [21], [34], [42] or [55].

Plaquette. The plaquette is an important term in Lattice QCD. It is the smallest closed loop of links in the (μ,ν) plane starting at site x in direction μ and ending at x as shown in figure 2.2. Formally, it is the product of contiguous link matrices in order that

$$\mathcal{P}_{x,\mu\nu}([U]) := U_{x,\mu} U_{x+a\hat{\mu},\nu} U^{\dagger}_{x+a\hat{\nu},\mu} U^{\dagger}_{x,\nu} \,. \tag{2.1}$$

For each space-time argument x and each plane (μ,ν) a plaquette $\mathcal{P}_{x,\mu\nu}$ can be computed. On the other hand, the plaquette can also be computed in the plane (ν,μ) in order that its orientation points in the other direction

$$\mathcal{P}_{x,\nu\mu}([U]) := U_{x,\nu} U_{x+a\hat{\nu},\mu} U^{\dagger}_{x+a\hat{\mu},\nu} U^{\dagger}_{x,\mu} \,. \tag{2.2}$$

It always holds $\mathcal{P}_{x,\nu\mu}([U]) = \mathcal{P}_{x,\mu\nu}([U])^{\dagger}$ which can be proven by a simple calculation. By convention, the plaquette usually is computed in anti-clockwise direction. In general, there are $2(d-1)$ plaquettes for each grid point on a d-dimensional lattice. This implies that a 2-dimensional lattice contains the planes $(0,1)$ and $(1,0)$. On a four-dimensional lattice, there are six planes nominated $(0,1)$, $(0,2)$, $(0,3)$, $(1,2)$, $(1,3)$ and $(2,3)$ and each link $U_{x,\mu}$ is contained in three of them.

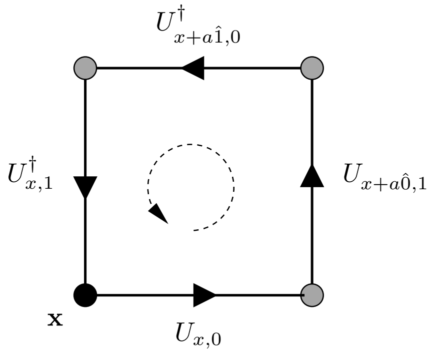

Figure 2.2: *Plaquette* – The plaquette is the shortest closed loop on the lattice.

Staples. Moreover, each link $U_{x,\mu}$ is clutched with the sum of staples $\mathcal{S}(U_{x,\mu})$ which embraces the link:

$$\mathcal{S}(U_{x,\mu}) = \sum_{\nu \neq \mu} U_{x+a\hat{\mu},\nu}\, U^\dagger_{x+a\hat{\nu},\mu}\, U^\dagger_{x,\nu} + U^\dagger_{x+a(\hat{\mu}-\hat{\nu}),\nu}\, U^\dagger_{x-a\hat{\nu},\mu}\, U_{x-a\hat{\nu},\nu} \tag{2.3}$$

This term comes into play in the Hamiltonian equations of motion as well as in the computation of the Wilson flow, so it is an important term in this work. The staples is the sum of the $2(d-1)$ products of every three link matrices starting at site $x + a\hat{\mu}$ and ending at site x. So, the staples comprises $2 \cdot 3 \cdot (d-1)$ link matrices in d dimensions.

Gauge invariance. It is essential that the computed physical quantities are gauge invariant. A quantity is gauge invariant if a change of all elements of the configuration $[U]$ according to a gauge transformation

$$U_{x,\mu} \to \tilde{U}_{x,\mu} = \Phi_x U_{x,\mu} \Phi^\dagger_{x+a\hat{\mu}} \tag{2.4}$$

with randomly chosen special unitary matrices Φ_x does not change the value of the quantity. The trace of the plaquette is the most simple gauge invariant quantity in Lattice QCD. Thus, it is frequently used in lattice computations, for example, for the computation of the Wilson gauge action. The gauge invariance of the trace of the plaquette can be easily seen due to the invariance of Φ_x, i.e. $\Phi_x \cdot \Phi^\dagger_x = I$. Let $[U]$ be a gauge field and $[\tilde{U}]$ a gauge transformed field according to equation (2.4). Then, the trace of the plaquette of the configuration $[U]$ has the same value as the one of $[\tilde{U}]$ since

$$\begin{aligned}
\mathcal{P}_{x,\mu\nu}([\tilde{U}]) &\overset{(2.1)}{=} \tilde{U}_{x,\mu}\tilde{U}_{x+a\hat{\mu},\nu}(\tilde{U}_{x+a\hat{\nu},\mu})^\dagger(\tilde{U}_{x,\nu})^\dagger \\
&\overset{(2.4)}{=} \Phi_x U_{x,\mu}\Phi^\dagger_{x+a\hat{\mu}} \;\cdot\; \Phi_{x+a\hat{\mu}} U_{x+a\hat{\mu},\nu}\Phi^\dagger_{x+a(\hat{\mu}+\nu)} \\
&\qquad \cdot (\Phi_{x+a\hat{\nu}} U_{x+a\hat{\nu},\mu}\Phi^\dagger_{x+a(\hat{\mu}+\nu)})^\dagger \;\cdot\; (\Phi_x U_{x,\nu}\Phi^\dagger_{x+a\hat{\nu}})^\dagger \\
&= \Phi_x U_{x,\mu}\, U_{x+a\hat{\mu},\nu}\Phi^\dagger_{x+a(\hat{\mu}+\nu)} \;\cdot\; \Phi_{x+a(\hat{\mu}+\nu)} U^\dagger_{x+a\hat{\nu},\mu}\Phi^\dagger_{x+a\hat{\nu}} \;\cdot\; \Phi_{x+a\hat{\nu}} U_{x,\nu}\Phi^\dagger_x \\
&= \Phi_x U_{x,\mu}\, U_{x+a\hat{\mu},\nu}\, U^\dagger_{x+a\hat{\nu},\mu}\, U_{x,\nu}\Phi^\dagger_x \; .
\end{aligned} \tag{2.5}$$

Due to the similarity-invariance of the trace (see equation (.2) in the appendix), it holds

$$\begin{aligned}
Tr(\mathcal{P}_{x,\mu\nu}([\tilde{U}]) &= Tr\Big(\Phi_x U_{x,\mu}\, U_{x+a\hat{\mu},\nu}\, U^\dagger_{x+a\hat{\nu},\mu}\, U_{x,\nu}\Phi^\dagger_x\Big) \\
&= Tr\Big(U_{x,\mu}\, U_{x+a\hat{\mu},\nu}\, U^\dagger_{x+a\hat{\nu},\mu}\, U_{x,\nu}\Phi^\dagger_x\, \Phi_x\Big) \\
&= Tr\Big(U_{x,\mu}\, U_{x+a\hat{\mu},\nu}\, U^\dagger_{x+a\hat{\nu},\mu}\, U_{x,\nu}\Big) = Tr(\mathcal{P}_{x,\mu\nu}([U]))
\end{aligned} \tag{2.6}$$

in order that the trace of a plaquette is gauge invariant. In Lattice QCD, expectation values are computed via Monte Carlo simulations on gauge invariant configurations. Thus, it is very important that each simulation code is checked for gauge invariance.

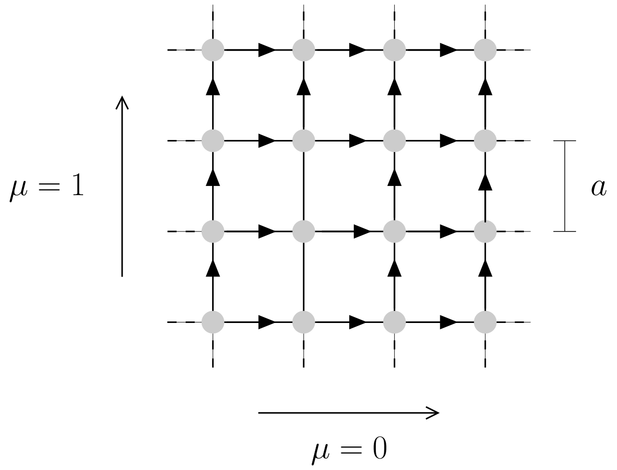

Figure 2.3: Lattice gauge field in 2 dimensions.

2.2 Hamiltonian Equations of Motion

Simulations in Lattice QCD aim at computing expectation values of some operators using the HMC algorithm developed by Duane et al in [17]. The heart of the simulations is the numerical integration in the Molecular Dynamics step. Here, Hamiltonian equations of motion have to be solved in a Lie group context using a geometric numerical integration method. The Hamiltonian and its equations of motion are stated in sections 2.2.1 and 2.2.2. There are different ways to set up the Hamiltonian equations of motion for Lattice QCD - either via considerations based on rotations and the Hamiltonian as constant in time or the more formal derivative of the Hamiltonian with respect to the Lie group and Lie algebra elements. These approaches are considered in paragraph 2.2.3. Finally, we focus on the structure of the differential equations of motion in section 2.2.4.

2.2.1 The Hamiltonian

The Hamiltonian equations of motion play an important role in HMC computations. They are solved in a fictitious computing time and produce the field configurations tending to the equilibrium distribution. The equations of motion depend on the Hamiltonian

$$\mathcal{H}([U],[P]) = E_K([P]) + S_G([U]) \tag{2.7}$$

which is composed as sum of the kinetic energy an $E_K([P])$ and the Wilson gauge action $S_G([U])$.

Wilson action. Here, the Wilson action S_G depends on all plaquettes of one configuration $[U]$, more precisely on the sum of the traces of all plaquettes in clockwise and anti-clockwise direction:

$$S_G([U]) = \sum_x \sum_{\mu \neq \nu} \beta \left(1 - \frac{1}{2N} \mathrm{Tr}\Big(\mathcal{P}_{x,\mu\nu}\left([U]\right)\Big)\right) . \tag{2.8}$$

N is the dimension of the matrices of the Lie group $SU(N,C)$ and β a coupling constant. This formula can be transformed to

$$\begin{aligned} S_G([U]) &= \sum_x \sum_{\mu<\nu} \beta \left(1 - \frac{1}{2N} \mathrm{Tr}\Big(\mathcal{P}_{x,\mu\nu}\left([U]\right) + \mathcal{P}_{x,\nu\mu}\left([U]\right)\Big)\right) \\ &= \sum_x \sum_{\mu<\nu} \beta \left(1 - \frac{1}{2N} \mathrm{Tr}\Big(\mathcal{P}_{x,\mu\nu}\left([U]\right) + \mathcal{P}^\dagger_{x,\mu\nu}\left([U]\right)\Big)\right) \\ &= \sum_x \sum_{\mu<\nu} \beta \left(1 - \frac{1}{N} Re\Big(\mathrm{Tr}\Big(\mathcal{P}_{x,\mu\nu}\left([U]\right)\Big)\Big)\right) \end{aligned} \tag{2.9}$$

in order that it considers the real part of the trace of all anti-clockwise orientated plaquettes. The last transformation holds due to the similarity transformation $\mathrm{Tr}(U + U^\dagger) = 2Re\mathrm{Tr}(U)$ (see equation (.3) in the appendix). The formula for the Wilson gauge action can be used for the computation of the Wilson flow and occurs as one part of the Hamiltonian. The Wilson action is gauge invariant since it is a sum of already gauge invariant plaquettes.

Kinetic energy. The momenta $P_{x,\mu}$, $x = 0, \ldots, n_p - 1$, $\mu = 0, \ldots, d-1$, are complex, traceless and hermitian matrices of size $N \times N$. They build the kinetic energy

$$E_K([P]) = \frac{1}{2} \sum_{x,\mu} \mathrm{Tr}\left(P^2_{x,\mu}\right) . \tag{2.10}$$

Hamiltonian. Both, the Wilson gauge action (2.9) and the kinetic energy (2.10) build the Hamiltonian

$$\mathcal{H}([U],[P]) = \frac{1}{2} \sum_{x,\mu} \mathrm{Tr}\left(P^2_{x,\mu}\right) + \sum_x \sum_{\mu<\nu} \beta \left(1 - \frac{1}{N} Re\left(\mathrm{Tr}\left(\mathcal{P}_{x,\mu\nu}[U]\right)\right)\right) \tag{2.11}$$

of the lattice gauge field in order that it depends on the lattice gauge field $[U]$ and its associated field of (fictitious) momenta $[P]$. Its value never changes. The momenta $[P]$ are associated to the links $[U]$ in order that every link $U_{x,\mu}$ corresponds to its momentum $P_{x,\mu}$. The links represent the gluons but the momenta are fictitious in order that the Hamiltonian is also a fictitious value.

2.2.2 Hamiltonian Equations of Motion.

The formulae for the Hamiltonian equations of motion

$$\frac{\partial \mathcal{H}}{\partial P_{x,\mu}} = \dot{U}_{x,\mu} \quad \text{and} \quad \frac{\partial \mathcal{H}}{\partial U_{x,\mu}} = -\dot{P}_{x,\mu}\,, \tag{2.12}$$

can be derived in several ways: first of all, they can be computed in a direct approach in order that the derivatives of the Hamiltonian $\mathcal{H}([U],[P])$ with respect to $P_{x,\mu}$ and $U_{x,\mu}$ are simply computed in a Lie group, respective Lie algebra context. Here, one has to know the procedure of computing derivatives of an operator with respect to Lie group and Lie algebra valued elements which depends on the structure of the Lie group and the Lie algebra. On the other hand, $\dot{U}_{x,\mu}$ and $\dot{P}_{x,\mu}$ can be derived in a more application oriented model via infinitesimal rotations for $\dot{U}_{x,\mu}$ and the energy conservation of the Hamiltonian for $\dot{P}_{x,\mu}$.

Finally, the Hamiltonian equations of motion are stated in the following definition.

Definition 2.1 (Hamiltonian Equations of Motion)**.** Let a d-dimensional lattice gauge field be given. Then, the Hamiltonian equations of motion read

$$\frac{\partial \mathcal{H}}{\partial P_{x,\mu}} = \dot{U}_{x,\mu} = iP_{x,\mu}U_{x,\mu}\,, \tag{2.13a}$$

$$\frac{\partial \mathcal{H}}{\partial U_{x,\mu}} = -\dot{P}_{x,\mu} = -i\frac{\beta}{N}\Big\{U_{x,\mu}\mathcal{S}(U_{x,\mu})\Big\}_{TA} \tag{2.13b}$$

for $x = 0,\ldots,T\cdot L^{d-1}-1$ and $\mu = 0,\ldots,d-1$. Here, β is the (prescribed) coupling constant and N the dimension of the matrix which is usually set to $N=3$ or $N=2$. The function $\mathcal{S}(U_{x,\mu})$ is called staples of $U_{x,\mu}$ as mentioned in equation (2.3) in section 2.1. Here, the traceless and anti-hermitian operator $\{\cdot\}_{TA}$ is given as

$$\Big\{M_{x,\mu}\Big\}_{TA} = \frac{1}{2}\big(M_{x,\mu} - M^{\dagger}_{x,\mu}\big) - \frac{1}{2N}\mathrm{Tr}(M_{x,\mu} - M^{\dagger}_{x,\mu})\cdot I_N \tag{2.14}$$

with $M_{x,\mu} := U_{x,\mu}\mathcal{S}(U_{x,\mu})$. The first part $M_{x,\mu} - M^{\dagger}_{x,\mu}$ of (2.14) provides the skew-symmetry of the result and the second part ensures that it traceless. using $M_{x,\mu}$ defined as $M_{x,\mu} := U_{x,\mu}\mathcal{S}_{x,\mu} - (U_{x,\mu}\mathcal{S}_{x,\mu})^{\dagger}$.

Numerical Solution. The Hamiltonian equations of motion have to be solved with a geometric integration scheme. This means, the numerical result of equation (2.13a) must be in the Lie group $SU(N)$. Furthermore, the numerical scheme has to be time-reversible and the volume of the phase space has to be preserved. Theoretically, the volume-preservation could be overcome by a multiplication of the inverse of the Jacobian of the system. However, in practice, this requires a small phase space with the result that this approach is not feasible for lattice computations.

The demanded performance of the properties time-reversibility and volume-preservation can be motivated in two different ways: first of all, volume-preservation and time-reversibility is preserved by the flow of a Hamiltonian system (see paragraph 5.1). Hence, a numerical integration scheme should do it as well. Second, it is absolutely necessary for the Hybrid Monte Carlo step that the Hamiltonian equations of motion are solved using a time-reversible and volume-preserving scheme, in order that the Markov chain tends to the correct equilibrium distribution $\exp(-S_G([U]))$.

In the following, it is shown how the Hamiltonian equations of motion can be deduced using different approaches.

2.2.3 Derivation of the Hamiltonian Equations of Motion

The formula for the Hamiltonian equations of motion of lattice gauge fields can be established via observations of the underlying special unitary Lie group and its associated Lie algebra.

In the context of lattice gauge theory, the Lie group

$$SU(N,\mathbb{C}) = \{U \in \mathbb{C}^{N\times N} \mid U^\dagger = U^{-1} \quad \text{and} \quad \det(U) = 1\} \tag{2.15}$$

and its associated Lie algebra

$$\mathfrak{su}(N,\mathbb{C}) = \{A \in \mathbb{C}^{N\times N} \mid A = -A^\dagger \quad \text{and} \quad \mathrm{Tr}(A) = 0\} \tag{2.16}$$

are considered. Its elements are matrices of size $N \times N$. The Lie group $SU(N,\mathbb{C})$ consists of unitary matrices with determinant one and can be identified with the $(N+1)$-dimensional unit sphere

$$\mathbb{S}_N = \{x \in \mathbb{R}^{N+1} \quad \text{with} \quad ||x||_2 = 1\}. \tag{2.17}$$

This means, every link matrix $U_{x,\mu}$ can be imagined to be situated on the unit sphere $\mathbb{S}_N$. Moreover, each element A of the Lie algebra $\mathfrak{su}(N,\mathbb{C})$ can be written as weighted sum of its N^2-1 generators T^a,

$$A = \sum_{a=1}^{N^2-1} \lambda_a T^a \quad \text{with} \quad \lambda_a \in \mathbb{R}, T^a \in \mathbb{C}^{n\times n}. \tag{2.18}$$

The generators T^a are traceless and anti-hermitian. Now, this knowledge can be used to set up the Hamiltonian equations of motion.

Infinitesimal Rotations. The differential equation (2.13a),

$$\dot{U}_{x,\mu} = iP_{x,\mu}U_{x,\mu}\,,$$

can be derived covering infinitesimal rotations. This approach is described in various text books about particle physics like, for example, in [67] or [14] and can be explained as follows.

It is possible to define small rotations

$$\mathcal{R}_N : SU(N,\mathbb{C}) \to SU(N,\mathbb{C}), \qquad U_{x,\mu} \to U_{x,\mu} = \mathcal{R}_N U_{x,\mu} \tag{2.19}$$

on the unit sphere $\mathbb{S}_N$ which transform one element $U_{x,\mu}$ of the Lie group to a slightly changed one

$$\widetilde{U}_{x,\mu} = U_{x,\mu} + \varepsilon \dot{U}_{x,\mu} + \mathcal{O}(\varepsilon^2)\,. \tag{2.20}$$

On the other hand, this rotation $\mathcal{R}_N$ can also be seen as infinitesimal transformation build from the identity and an infinitesimal transformation δT_N with very small δT_N, i.e.,

$$\mathcal{R}_N = I_N + \delta T_N\,. \tag{2.21}$$

All in all, this leads to a formula for changing the elements on the sphere:

$$\widetilde{U}_{x,\mu} = \mathcal{R}_N U_{x,\mu} = (I_N + \delta T_N)U_{x,\mu} = U_{x,\mu} + \delta T_N U_{x,\mu}\,. \tag{2.22}$$

A combination of equations (2.20) and (2.22) leads to

$$\dot{U}_{x,\mu} = \tfrac{1}{\varepsilon}\delta T_n U_{x,\mu} = \widetilde{\delta T}_N \cdot U_{x,\mu}\,. \tag{2.23}$$

It can easily be shown that $\widetilde{\delta T}_N$ has to be traceless and anti-hermitian, i.e., $\widetilde{\delta T}_N =: iP_{x,\mu}$ is in the Lie group $\mathfrak{su}(N,\mathbb{C})$. Thus, we derived equation (2.13a).

Conserved quantity Hamiltonian. For a Hamiltonian function, there is an easier way depending on the fact that the Hamiltonian is a conserved quantity, i.e., $d\mathcal{H}/dt = 0$. Gottlieb et al describe it in [22] and tel Tar and de Grand recapitulate it in chapter 7.2.3 in [16].

The time derivative of the Hamiltonian reads

$$\begin{aligned}\frac{\mathrm{d}\,\mathcal{H}([U],[P])}{\mathrm{d}\,t} &= \frac{\mathrm{d}\,E_K([P])}{\mathrm{d}\,t} + \frac{\mathrm{d}\,S_G([U])}{\mathrm{d}\,t}\\ &= \sum_{x,\mu}\mathrm{Tr}\Big\{P_{x,\mu}\Big(\dot{P}_{x,\mu} - i\,\underbrace{\frac{\beta}{N}\big(U_{x,\mu}\mathcal{S}(U_{x,\mu}) - \mathcal{S}(U_{x,\mu})^\dagger U^\dagger_{x,\mu}\big)}_{=:M_{x,\mu}}\Big)\Big\}\end{aligned} \tag{2.24}$$

with time derivatives

$$\frac{\mathrm{d}\,E_K([P])}{\mathrm{d}\,t} = \sum_{x,\mu}\mathrm{Tr}\big\{\dot{P}_{x,\mu}P_{x,\mu}\big\} \tag{2.25}$$

and

$$\begin{aligned}\frac{\mathrm{d}\,S_G([U])}{\mathrm{d}\,t} &= -i\frac{\beta}{N}\sum_{x,\mu}\mathrm{Tr}\Big\{P_{x,\mu}\Big(U_{x,\mu}\mathcal{S}(U_{x,\mu}) - \mathcal{S}(U_{x,\mu})^\dagger U^\dagger_{x,\mu}\Big)\Big\} \\ &= -i\frac{\beta}{N}\sum_{x,\mu}\mathrm{Tr}\Big\{P_{x,\mu}\Big(U_{x,\mu}\mathcal{S}(U_{x,\mu}) - \mathcal{S}(U_{x,\mu})^\dagger U^\dagger_{x,\mu}\Big)\Big\} \qquad (2.26)\end{aligned}$$

for the kinetic energy (2.10) and the Wilson gauge action (2.8). Based on equation (2.24), we derive a formula for $\dot{P}_{x,\mu}$ if the sufficient condition

$$\mathrm{Tr}\Big\{P_{x,\mu}\cdot\Big(\dot{P}_{x,\mu} - iM_{x,\mu}\Big)\Big\} = 0 \quad \forall\ x,\mu \qquad (2.27)$$

is fulfilled. Since $P_{x,\mu}$ is traceless and hermitian, the second factor $\dot{P}_{x,\mu} - iM_{x,\mu}$ of equation (2.27) must be a multiple of the identity matrix, i. e.

$$\dot{P}_{x,\mu} - iM_{x,\mu} = c\cdot I_N\,. \qquad (2.28)$$

Now, we just have to compute the unknown variable c. Since a Lie algebra is a linear space, the derivative $\dot{P}_{x,\mu}$ of the traceless element $P_{x,\mu}$ has also be traceless:

$$0 = \mathrm{Tr}\{\dot{P}_{x,\mu}\} = \mathrm{Tr}\{c\cdot I_N + iM_{x,\mu}\} = c\cdot N + i\cdot\mathrm{Tr}\{M_{x,\mu}\} \qquad (2.29)$$

and it follows

$$c = -\frac{i}{N}\mathrm{Tr}\{M_{x,\mu}\}\,. \qquad (2.30)$$

Finally, we get

$$\dot{P}_{x,\mu} = iM_{x,\mu} - \frac{i}{N}\mathrm{Tr}\{M_{x,\mu}\}\cdot I_N = i\frac{\beta}{N}\Big\{U_{x,\mu}\mathcal{S}(U_{x,\mu})\Big\}_{TA} \qquad (2.31)$$

which is equal to equation (2.13b).

Lie group derivative. The derivative $\partial\mathcal{H}([U],[P])/U_{x,\mu}$ is computed as follows as described in, for example, [37], [38] or [43]. Let $U_{x,\mu}$ be an element of a matrix Lie group $SU(N,\mathbb{C})$ and f be a function inside the group in order that

$$f:\big(SU(N,\mathbb{C})\big)^{n_l}\to\mathbb{C}, \qquad\qquad [U]\mapsto f([U]). \qquad (2.32)$$

The Lie group derivative $\partial_{x,\mu}f([U])$ is the derivative of the function $f([U])$ with respect to the Lie group element $U_{x,\mu}$. It is defined in [37] as

$$\partial_{x,\mu}f([U]) := \frac{\partial}{\partial U_{x,\mu}}f([U]) = T^a\partial^a_{x,\mu}f([U]), \qquad (2.33a)$$

$$\text{with}\quad \partial^a_{x,\mu}f([U]) = \frac{d}{ds}f\big([\exp(sX)U]\big)|_{s=0}\,, \qquad (2.33b)$$

$$\text{and} \quad X(y,\nu) = \begin{cases} T^a & \text{if } (y,\nu) = (x,\mu) \\ 0 & \text{otherwise} \end{cases} \tag{2.33c}$$

and needs some explanation. First of all, the summation convention of Einstein is used in (2.33a) in order that is a summation over a. Since each Lie algebra $\mathfrak{su}(N,\mathbb{C})$ is composed as weighted sum of N^2-1 generators T^a, the sum goes from $a=1$ to $a=N^2-1$. Thus, equation (2.33a) can be denoted as

$$\frac{\partial}{\partial U_{x,\mu}} f(U) = \partial_{x,\mu} f(U) = T^a \partial^a_{x,\mu} f(U) = \sum_{a=1}^{N^2-1} T^a \partial^a_{x,\mu} f(U)\,. \tag{2.34}$$

The derivatives $\partial^a_{x,\mu} f(U)$ given in (2.33b) are total derivatives with respect to time evaluated at time point $s=0$. Here, the combination (y,ν) denotes all links with arbitrary indices. This implies that all links called $U_{y,\nu}$ are replaced with $\exp(sX)U_{y,\nu}$. Afterwards, its derivative is computed with respect to s followed by an evaluation at $s=0$, i.e., the expression $\exp(sX)$ is replaced with X. Finally, X is set to T^a if the index (y,ν) of the link is equal to x,μ, otherwise X is set to 0. This means, all parts of the function $f(U)$ depending on the links $U_{x,\mu}$ contribute to the Lie group derivative $\partial^a_{x,\mu} f(U)$, the others are set to 0. Both, the Hamiltonian equation of motion $\partial\mathcal{H}([U],[P])\partial U_{x,\mu}$ and the Wilson flow depend on a Lie group derivative of the Wilson gauge action $S_G([U])$ with respect to a link variable $U_{x,\mu}$ and can be computed this way.

For the computation of the Lie group derivative of the Wilson gauge action $S_G([U])$ using notation (2.8), i.e.,

$$S_G([U]) = \sum_x \sum_{\mu\neq\nu} \beta \left(1 - \frac{1}{2N} \mathrm{Tr}\Big(\mathcal{P}_{x,\mu\nu}([U])\Big)\right)\,,$$

we consider its shape. $S_G([U])$ can be split into a part

$$S_G([U])^{(1)} = -\frac{\beta}{2N} Tr\Big(U_{x,\mu}\mathcal{S}(U_{x,\mu}) + \big(U_{x,\mu}\mathcal{S}(U_{x,\mu})\big)^\dagger\Big) \tag{2.35}$$

and a part not containing the link $U_{x,\mu}$. It follows

$$\begin{aligned} \partial^a_{x,\mu} S_G([U]) &= \partial^a_{x,\mu} S^{(1)}_G([U]) \\ &= -\frac{\beta}{N} \cdot \frac{d}{ds} \mathrm{Tr}\Big\{\exp^{sX} U_{x,\mu}\mathcal{S}(U_{x,\mu}) + \big(\exp^{sX} U_{x,\mu}\mathcal{S}(U_{x,\mu})\big)^\dagger\Big\}\Big|_{s=0} \end{aligned} \tag{2.36}$$

since $\mathcal{S}(U_{x,\mu})$ consists of all links of the plaquette excluding $U_{x,\mu}$. Moreover,

$$\partial^a_{x,\mu} S_G([U]) = -\frac{\beta}{N} \mathrm{Tr}\Big\{X U_{x,\mu}\mathcal{S}(U_{x,\mu}) + \big(X U_{x,\mu}\mathcal{S}(U_{x,\mu})\big)^\dagger\Big\} \tag{2.37}$$

$$= -\frac{\beta}{N} \mathrm{Tr}\Big\{T^a\Big(U_{x,\mu}\mathcal{S}(U_{x,\mu}) - \mathcal{S}(U_{x,\mu})^\dagger U^\dagger_{x,\mu}\Big)\Big\} \tag{2.38}$$

$$=: -\frac{\beta}{N} \mathrm{Tr}\Big\{T^a M_{x,\mu}\Big\}\,. \tag{2.39}$$

Thus, it holds

$$\partial_{x,\mu} S_G([U]) = \sum_a T^a \partial^a_{x,\mu} S_G([U]) = -\frac{\beta}{N} \sum_a T^a \mathrm{Tr}\Big\{T^a M_{x,\mu}\Big\} \tag{2.40}$$

with $M_{x,\mu} := U_{x,\mu}\mathcal{S}(U_{x,\mu}) - \mathcal{S}(U_{x,\mu})^\dagger U^\dagger_{x,\mu}$. Using the identity

$$-\frac{\beta}{N} \sum_a T^a \mathrm{Tr}\Big\{T^a M_{x,\mu}\Big\} = \{M_{x,\mu}\}_{TA} \tag{2.41}$$

(see equation (2.58) of [34]), we get

$$\partial_{x,\mu} S_G([U]) = -\frac{\beta}{N} \sum_a T^a \mathrm{Tr}\Big\{T^a M_{x,\mu}\Big\} = -\frac{\beta}{N}\{M_{x,\mu}\}_{TA} = i\dot{P}_{x,\mu}\,. \tag{2.42}$$

Lie algebra derivative. Also the Lie algebra derivative $\partial\mathcal{H}([U],[P])/\partial P_{x,\mu}$ can be computed in a similar way. Here, even a bit more theory has to be used since the Lie algebra derivative depends on fundamental 2-forms. This approach is described by Kennedy in [33] in detail and recapitulated in [34]. Shortly, it can be described in the following way:

We can denote the derivatives

$$\dot{U}_{x,\mu} = \frac{\partial\mathcal{H}([U,P])}{\partial P_{x,\mu}} \quad \text{and} \quad \dot{P}_{x,\mu} = \frac{\partial\mathcal{H}([U,P])}{\partial U_{x,\mu}} \tag{2.43}$$

with the Poisson bracket

$$\{A,B\} = \frac{\partial A}{\partial P_{x,\mu}}\frac{\partial B}{\partial U_{x,\mu}} - \frac{\partial A}{\partial U_{x,\mu}}\frac{\partial B}{\partial P_{x,\mu}} \tag{2.44}$$

for a fundamental zero-form A. This leads to

$$\dot{U}_{x,\mu} = \{\mathcal{H}, U_{x,\mu}\} = \frac{\partial\mathcal{H}}{\partial P_{x,\mu}}\frac{\partial U_{x,\mu}}{\partial U_{x,\mu}} - \frac{\partial\mathcal{H}}{\partial U_{x,\mu}}\frac{\partial U_{x,\mu}}{\partial P_{x,\mu}} \tag{2.45}$$

and

$$\dot{P}_{x,\mu} = \{\mathcal{H}, P_{x,\mu}\} = \frac{\partial\mathcal{H}}{\partial P_{x,\mu}}\frac{\partial P_{x,\mu}}{\partial U_{x,\mu}} - \frac{\partial\mathcal{H}}{\partial U_{x,\mu}}\frac{\partial P_{x,\mu}}{\partial P_{x,\mu}}\,. \tag{2.46}$$

At the same time, it holds

$$\{\mathcal{H}, A\} = \widehat{\mathcal{H}}A \tag{2.47}$$

with

$$\widehat{\mathcal{H}} = \frac{\partial\mathcal{H}}{\partial P_{x,\mu}}\frac{\partial}{\partial U_{x,\mu}} - \frac{\partial\mathcal{H}}{\partial U_{x,\mu}}\frac{\partial}{\partial P_{x,\mu}} \tag{2.48a}$$

$$= \frac{\partial\mathcal{H}}{\partial p_a}e_a - e_a(\mathcal{H})\frac{\partial}{\partial p_a}\,. \tag{2.48b}$$

This formula includes some transformations. First, the traceless and hermitian matrix $P_{x,\mu}$ is denoted as sum of its generators T_a, i.e.,

$$P_{x,\mu} = -i \sum_a p_a T_a =: -i p_a T_a \,. \tag{2.49}$$

Then, e_a is a linear differential operator acting on $U_{x,\mu}$ similar to $\partial^a_{x,\mu}$ of equation (2.33a). It holds $-T_a U_{x,\mu} \equiv e_a(U_{x,\mu})$, i.e., the generators T_a are defined by the linear differential operator. Furthermore, Einstein's summation convention is used – equation (2.48b) includes a summation over a.

It follows

$$\dot{U}_{x,\mu} = \{\mathcal{H}, U_{x,\mu}\} = \widehat{\mathcal{H}} U_{x,\mu} = \frac{\partial \mathcal{H}}{\partial p_a} e_a(U_{x,\mu}) - e_a(\mathcal{H}) \frac{\partial U_{x,\mu}}{\partial p_a} \tag{2.50}$$

$$\text{and} \quad \dot{P}_{x,\mu} = \{\mathcal{H}, P_{x,\mu}\} = \widehat{\mathcal{H}} P_{x,\mu} = \frac{\partial \mathcal{H}}{\partial p_a} e_a(P_{x,\mu}) - e_a(\mathcal{H}) \frac{\partial P_{x,\mu}}{\partial p_a} \,. \tag{2.51}$$

We know the partial derivatives of $\mathcal{H}$, $U_{x,\mu}$ and $P_{x,\mu}$ with respect to p_a, i.e.,

$$\frac{\partial \mathcal{H}}{p_a} = -i p_a, \qquad \frac{\partial U_{x,\mu}}{p_a} = 0, \qquad \frac{\partial P_{x,\mu}}{p_a} = -i T_a \,, \tag{2.52}$$

and the acting of the linear differential operator e_a, i.e.,

$$e_a(U_{x,\mu}) = -T_a U_{x,\mu}, \qquad e_a(P_{x,\mu}) = 0, \qquad e_a(\mathcal{H}) = e_a(S_G) \,. \tag{2.53}$$

Thus,

$$\dot{U}_{x,\mu} = -i p_a(-i T_a U_{x,\mu}) - e_a(S_G) \cdot 0 = i P_{x,\mu} U_{x,\mu} \tag{2.54}$$

$$\text{and} \quad \dot{P}_{x,\mu} = \frac{\partial \mathcal{H}}{\partial p_a} \cdot 0 - e_a(S_G) T_a = -e_a(S_G) T_a \,. \tag{2.55}$$

Equation (2.54) coincides with equation (2.13a) and (2.55) with (2.13b).

2.2.4 Structure of the Hamiltonian Equations of Motion.

Concerning the development of numerical integration methods, it is convenient to change the notation of the Hamiltonian equations of motion from the indices (x, μ) towards a single index j. Moreover, we have to care about the structure of the single differential equations.

Change of notation. So far, the link matrix $U_{x,\mu}$ is an element of the Lie group $SU(N, \mathbb{C})$ and the momentum $P_{x,\mu}$ is traceless and hermitian. For differential equations on Lie groups and its underlying theory, it is convenient to work in the appropriate Lie algebra. In case of $SU(N, \mathbb{C})$, the appropriate special unitary Lie algebra $\mathfrak{su}(N, \mathbb{C})$ consists of traceless and anti-hermitian matrices. So, a multiplication with

$i = \sqrt{-1}$ leads to a traceless and anti-hermitian element $iP_{x,\mu}$ which is an element of $\mathfrak{su}(N, \mathbb{C})$. Furthermore, a single index j is more convenient for the development of numerical integration methods than a combination of x and μ. Thus, the links $U_{x,\mu}$ can be identified with Lie group elements Y_j, the scaled momenta $iP_{x\mu}$ with Lie algebra elements A_j and the staples $\mathcal{S}(U_{x,\mu})$ with square matrices S_j. It holds

$$U_{x,\mu} \leftrightarrow Y_j, \qquad iP_{x\mu} \leftrightarrow A_j, \qquad S(U_{x,\mu}) \leftrightarrow \mathcal{S}_j\,. \tag{2.56}$$

The exchange of the indices $(x, \mu) \leftrightarrow j$ can be done using the lexicographic index

$$j := n_p \cdot \mu + x \quad \text{with} \quad n_p = T \cdot L^{d-1} \tag{2.57a}$$

in both directions. The pair (x, μ) can be gained via

$$x = j \mod n_p \quad \text{followed by} \quad \mu = (j - x)/n_p\,. \tag{2.57b}$$

The substitution has to be done for all $n_l := d \cdot T \cdot L^{d-1}$ positions of the link matrices. Thus, the equations of motion (2.13) can be formulated in a more convenient way as

$$\dot{Y}_j(t) = A_j(t) \cdot Y_j(t), \tag{2.58a}$$

$$\dot{A}_j(t) = -\frac{\beta}{N}\Big\{Y_j(t)\mathcal{S}_j(t)\Big\}_{TA} \tag{2.58b}$$

for $j = 0, \ldots, n_l - 1$ with staples S_j and traceless anti-hermitian operator $\{\cdot\}_{TA}$ defined in (2.14). Here, the indices j of formula (2.58) and (x, μ) of equation (2.13) can just be transformed according to the transformations (2.57).

Structure of the Hamiltonian equations of motion. Equation (2.58a) contains the elements A_j and Y_j at position j, i.e., it is a scalar equation. On the other hand, equation (2.58b) depends on the link Y_j and its staples $\mathcal{S}_j(t)$. Thus, $6(d-1)$ further links – belonging to the staples – are involved in this equation. This set of links is denoted as $[Y]_s$ as denoted in the previous paragraph 2.1.2. In the following, the notation of equation (2.58b) is changed into

$$\dot{A}_j(t) = -\frac{\beta}{N}\Big\{Y_j(t)\mathcal{S}_j(t)\Big\}_{TA} =: F\Big([Y_j(t)]_s\Big) \tag{2.59}$$

with

$$[Y_j]_s := \{Y_j, Y_{\sigma_1,j}, \ldots, Y_{\sigma_s,j}\}\,. \tag{2.60}$$

being the set of the link matrix Y_j itself and its $s := 6(d-1)$ elements $Y_{\sigma_1,j}, \ldots, Y_{\sigma_s,j}$ of the staples $S(Y_j)$. The expression $\sigma_{k,j}$, $k = 1, \ldots, s$, denotes computation of the k-th element of the staples of Y_j according to equation (2.3). So, the evaluation of the functions F in (2.59) depends on more than one single element $Y_j(t)$ expressed by $[Y_j(t)]_s$. For the development of integration methods on Lie groups, we neglect this fact and treat equation (2.59) as a scalar one. It is just important for the (parallel version of) the implementation.

2.3 Hybrid Monte Carlo Method

In SU(3) Yang-Mills theory, expectation values are computed using a sophisticated Monte Carlo method: the Hybrid Monte Carlo (HMC) method developed in 1980 by Duane et al [17]. It combines a Metropolis Monte Carlo method with a Molecular Dynamics step. Here, a Hamiltonian (which is a conserved quantity) is introduced for the Molecular Dynamics step. It is based on the gauge field $[U]$ and its associated field $[P]$ of momenta and leads to Hamiltonian equations of motion which have to be solved with a geometric numeric integration scheme. Hamiltonians are constant in time with the result that the difference of two successive Hamiltonians - before and after the numerical integration of one trajectory - just depends on the numerical errors. It is intended that the numerical integration leads to a high transition probability in the Monte Carlo method.

This section starts with a rough overview on expectation values and its computation. Then, the Metropolis Monte Carlo methods are introduced with focus on Markov chains and its stability criterion since this is the reason for the necessity of an integration scheme which is volume-preserving and time-reversible. Finally, the Hybrid Monte Carlo method is presented and explained.

Expectation values. Lattice gauge theory aims at computing expectation values of some operators of the gauge field $[U]$. Starting from field configurations $[U]$, the operator $\mathcal{A}([U])$ leads to the expectation value $\langle\mathcal{A}\rangle$ computed by the path integral

$$\langle\mathcal{A}\rangle = \frac{1}{Z}\int [dU]\; p([U])\; \mathcal{A}([U]) \tag{2.61}$$

with probability $p([U])$, partition function $Z = \int [dU] p([U])$ and Haar measure $[dU]$. Since the integral is of high dimension, the direct evaluation of the integral is not possible. Theoretically, the expectation value $\langle\mathcal{A}\rangle$ could be computed as a weighted sum over all possible configurations $\{[U]\}$ with probability

$$p([U]) = \exp\bigl(-S_G([U])\bigr) \tag{2.62}$$

that the configuration $[U]$ will occur:

$$\langle\mathcal{A}\rangle = \sum_{\{[U]\}} \mathcal{A}\bigl([U]\bigr)\cdot p\bigl([U]\bigr)\;. \tag{2.63}$$

Unfortunately, the number of all possible configurations is too large to be used in practice. So, just an ensemble of field configurations $\{[U]\}$ can be selected for the computation of $\langle\mathcal{A}\rangle$. This is realized in a Monte Carlo simulation in order that the selection is done randomly. However, this simple sampling leads to troubles because the probability for the existence of most configurations is very small since the lattice is very large: the statistical errors would be very large and thus the expectation value quite inaccurate. Of course, this can be improved using more configurations. The usage of an importance sampling technique is an alternative: configurations occurring with a higher probability will be preferably selected and the expectation value is computed as

$$\langle \bar{\mathcal{A}} \rangle = \frac{1}{N_{sample}} \sum_i \mathcal{A}([U]_i) \ . \tag{2.64}$$

For the realization of the importance sampling, it is necessary that any random initial configuration leads to configurations occurring with high probability. This means, the equilibrium distribution concerning the probability distribution should be met. This can be realized via Monte Carlo methods based on Markov processes.

Monte Carlo Methods. The Metropolis Monte Carlo (MMC) method by Rosenbluth et. al [40] is based on a Markov process and generates configurations which tend to the equilibrium distribution. The Markov chain can be infinitely large and each configuration can occur several times followed by arbitrary possible configurations. For this purpose, the target distribution has to be known with the result that the probability $p([U])$ can be computed for any samples $[U]$. The idea of the MMC method is stated in the following algorithm:

Algorithm 2.2 (Metropolis Monte Carlo Method)**.**

1. Start with a random field $[U]_1$. Set the index i to 1.
2. Generate a random field $[U]_j$.
3. Accept the new field $[U]_j$ with transition probability

$$T_{ij} = \min\big(1,\, p([U]_j)/p([U]_i)\big). \tag{2.65}$$

This means,

$$[U]_{i+1} = \begin{cases} [U]_j & \text{in case of acceptance,} \\ [U]_i & \text{in case of a rejection} \end{cases} \tag{2.66}$$

of the random field $[U]_i$. Set $i := i+1$ and proceed with step 2.

In algorithm 2.2, a Markov chain of configurations

$$[U]_1 \to [U]_2 \to [U]_3 \to \ldots \to \tag{2.67}$$

is produced as proposed by Metropolis et al in [40]. Starting from configuration $[U]_i$, the proposal of a random field $[U]_j$ is alternated with an acceptance step according to the transition probability (2.65). $[U]_j$ is just a suggestion for the next link of the Markov chain. Afterwards, the more probable configuration $[U_i]$ or $[U_j]$ is added in the acceptance step which works as follows: if $[U]_j$ occurs with higher probability than $[U]_i$, i.e., $p([U]_j) > p([U]_i)$, $[U]_{i+1}$ is set to $[U]_j$. Otherwise, a uniformly distributed random number $r \in [0,1]$ is drawn and compared to $p([U]_j)/p([U]_i)$. If

r is smaller than or equal to

$$\frac{p([U]_j)}{p([U]_i} = \exp(-\Delta S_G), \quad \Delta S_G = S_G([U]_j - S_G([U]_i)\,, \tag{2.68}$$

the new configuration $[U]_j$ also is accepted. Otherwise, the new configuration is set to $[U]_{i+1} := [U]_i$ in order that the old configuration is more probable than the new one and occurs several (at least two) times in the Markov chain.

Markov Chains. Reaching the equilibrium distribution is based on the consideration that the the Markov process can be described by a fixed point equation

$$\pi = \pi T \quad \text{with} \quad \pi \in \mathbb{R}^n \quad \text{and} \quad T \in \mathbb{R}^{n\times n} \tag{2.69}$$

using the following considerations according to, for example, [34] or [42]: let there be n possible configurations $[U]_1, \ldots, [U]_n$ which occur with the probabilities $p_i := p([U]_i)$, $i = 1, \ldots, n$. The probability to reach configuration $[U]_j$ from $[U]_i$ is given by the transition probability

$$T\Big([U]_i \to [U]_j\Big) =: T_{ij} \tag{2.70}$$

such that all probable transition probabilities T_{ij} can be collected in the transition matrix $T = \big(T_{ij}\big)$ of size $n \times n$. It holds

$$\sum_j T_{ij} = 1 \tag{2.71}$$

and each configuration can be reached from any other one, i.e., $T_{ij} \geq 0$.

Furthermore, $\pi^{(k)} \in \mathbb{R}^n$ contains the probabilities of all configurations occuring in step k of the Markov chain. Then, the Markov process can be described by the equation

$$\pi^{(k+1)} = \pi^{(k)}T = \pi^{(0)}T^k. \tag{2.72}$$

This implies

$$\lim_{k\to\infty} \pi^{(0)}T^k = \pi \tag{2.73}$$

with equilibrium distribution π and (2.69) immediately follows. The fixed point π is unique if it holds

$$\pi_j = \sum_i \pi_i T_{ij} \quad \forall \; j\,. \tag{2.74}$$

This is the case if the sufficient but not necessary detailed balance condition

$$\pi_j T_{ji} = \pi_i T_{ij} \quad \forall \; i,j \tag{2.75}$$

is fulfilled which can easily be shown: a summation over i and the usage of equation

(2.71) leads to equation (2.74), i.e.,

$$\sum_i \pi_i T_{ij} = \sum_i \pi_j T_{ji} = \pi_j \sum_i T_{ji} = \pi_j \quad \forall\, i, j\,. \tag{2.76}$$

Thus, the detailed balance condition (2.75) is essential to reach the equilibrium distribution of the Markov chain.

HMC Algorithm and Molecular Dynamics Step. The Hybrid Monte Carlo (HMC) method is a type of Markov Chain Monte Carlo (MCMC) method published by Duane et al. [17] in 1987.

The idea is to combine a Molecular Dynamics step with an acceptance step and use the Hamiltonian $\mathcal{H}$ (which is a constant in time) for the probability distribution $p = \exp(-\mathcal{H})$. Since the Hamiltonian is a constant in time, this leads to a high acceptance rate depending on the errors of the numerical integration method used in the Molecular Dynamics step. For the Molecular Dynamics step, a fictitious time and a grid of fictitious momenta $[P]$ are introduced such that the Hamiltonian equations of motion

$$\frac{\partial \mathcal{H}}{\partial P_{x,\mu}} = \dot{U}_{x,\mu} \quad \text{and} \quad \frac{\partial \mathcal{H}}{\partial U_{x,\mu}} = -\dot{P}_{x,\mu} \tag{2.77}$$

can be solved numerically for the whole grid, i.e., for $x = 0, \ldots, n_p - 1$ and $\mu = 0, 1, 2, 3$. Here, it holds that each momentum $P_{x,\mu}$ is related to its link $U_{x,\mu}$. Due to the fact that the Hamiltonian is conserved in time, the Hamiltonians of two successive configurations will be the same up to the numerical errors of the integration method.

The HMC algorithm works as follows:

1. Start with an initial configuration $([U]_j, [P]_j)$ at time $t = 0$. Thereby, the configuration of momenta has to be updated in each step and consists of Gaussian distributed random numbers.
2. Reach the new configuration $([U]_k, [P]_k)$ by applying a Molecular Dynamics step, i.e., integrate the system $([U]_j, [P]_j)$ numerically.
3. Accept the new configuration with transition probability

   $$T_{jk} := \min(1, \tfrac{p_k}{p_j}) = \min\Big(1, \exp(-\Delta\mathcal{H})\Big)\ , \ \Delta\mathcal{H} := \mathcal{H}_k - \mathcal{H}_j .$$

   Otherwise keep $[U]_j$ and $[P]_j$.
4. Proceed with step 1.

As the transition probability depends only on the difference of the Hamiltonians $\Delta\mathcal{H}$, the acceptance rate is linked to the convergence order of the numerical integration scheme used in step 2 of the Molecular Dynamics step. If the new configuration $[U]_k$ is accepted, it is added to the ensemble of link configurations. Otherwise, the old configuration $[U]_j$ is added (again) to the ensemble since this state is more probable than the new one.

It is essential for the Hybrid Monte Carlo method that the Markov process converges to the fixed point of the equilibrium distribution of the field configurations $[U]$. To ensure this, the numerical integration scheme still has to fulfill the detailed balance condition (2.75) concerning the action $S([U])$. Therefore, the numerical integration schemes used in the Molecular Dynamics Step have to be time-reversible and should be volume-preserving. The time-reversibility is mandatory, while a missing volume-preservation can be compensated theoretically by a proper scaling with the determinant of the Jacobian of the whole system $\partial([U]_j, [P]_j)/\partial([U]_i, [P]_i)$. This is discussed in [17]. The convergence order p of the numerical integration scheme is just of interest to achieve a high acceptance rate because $|\Delta\mathcal{H}|$ is proportional to the error of the integration scheme.

2.4 The Wilson Flow

The Wilson flow [38, 47, 36] is a gradient flow in lattice gauge theory. Starting from an initial gauge field $[V]_0$ of size $n_l = dn_p$ with $n_p = T \cdot L^{d-1}$, the Wilson flow

$$[V]_0 \to [V]_1 \to [V]_2 \to [V]_3 \to \dots$$

is the solution of an ordinary differential equation

$$\dot{V}_{x,\mu}(t) = Z_{x,\mu}(t) \cdot V_{x,\mu}(t) \tag{2.78a}$$

$$\text{with} \quad Z_{x,\mu}(t) = -\frac{\partial S_G([V(t)])}{\partial V_{x,\mu}(t)} \tag{2.78b}$$

for $x = 0, \dots, n_p - 1$, $n_p = TL^{d-1}$, and $\mu = 0, \dots, d-1$. The indices (x, μ) of the links $V_{x,\mu}$ depict that the link is situated between the grid points x and $x + a\hat{\mu}$ with direction μ. Equation (2.78b) is computed via the principle of computing the derivative of a function with respect to a Lie group element. Since the Wilson gauge action is a part of the Hamiltonian, it is no surprise that the shape of $Z = -\partial S_G / V_{x,\mu}$ is similar to the shape of the Hamiltonian equation $\dot{P}_{x,\mu} = -\partial\mathcal{H}/\partial U_{x,\mu}$. Finally, the formula for $Z : SU(N, \mathbb{C}) \to \mathfrak{su}(N, \mathbb{C})$ is

$$Z_{x,\mu}(t) = -\frac{\partial S_G([V(t)])}{\partial V_{x,\mu}(t)} = -\frac{\beta}{N}\big\{V_{x,\mu}\mathcal{S}(V_{x,\mu})\big\}_{TA} \tag{2.79}$$

with staples $\mathcal{S}(V_{x,\mu})$ defined in equation (2.3) and the traceless anti-hermitian operator $\{\cdot\}_{TA}$ stated in formula (2.14).

Notation and structure. Using the lexicographic index $j := \mu \cdot n_p + x$ described in the previous section 2.2, we transform equation (2.78) and (2.79) to

$$\dot{V}_j(t) = Z_j(t) \cdot V_j(t) \tag{2.80a}$$

$$\text{with} \quad Z_j(t) = -\frac{\beta}{N}\{V_j \mathcal{S}(V_j)\}_{TA} =: F([V_j]_s) \tag{2.80b}$$

for $j = 0, \ldots, n_l - 1$. The values $V_j(t)$ are elements of the special unitary Lie group $SU(N, \mathbb{C})$ and denote the link matrices. Z_j is a Lie-algebra valued function from the Lie group to the Lie algebra. It is already explained in the previous section 2.2 that the computation of Z_j depends on the link V_j itself and its $s := 6(d-1)$ elements of its staples $\mathcal{S}(V_j)$ leading to an expression $Z_j(t) = F([V_j]_s)$ according to the right-hand side of equation (2.59).

So, equation (2.80) is a differential equation on a Lie group. In contrast to equation (2.58a) of the Hamiltonian equation of motions, it is no scalar equation but depends on the set $[V_j]_s$. For the development of integration methods, we neglect this fact and consider the first equation of (2.80) as a scalar differential equation on a Lie group and the second one as an evaluation of an expression in the Lie algebra.

The differential equation is solved as initial value problem with initial values $[V]_0 := [U]$ arising from a configuration of link variables gained, for example, via the HMC or the heat-bath algorithm. So, the shape of the Wilson flow (2.80) is very similar to the one of the Hamiltonian equations of motion. Concerning numerical integration schemes on Lie groups, the Wilson flow (2.80) could be solved with any numerical method for Lie groups like the ones described in section 3.2 or chapter 4. No geometric structure – except for the Lie group structure – needs to be obtained.

Observables. The Wilson flow (2.78) can be integrated numerically with initial values $[V(t_0)]$ taken, for example, from HMC or heat bath configurations computed for a particular value of the coupling constant β. The numerical integration is performed up to a certain flow time and gauge invariant observables of interest are measured during this computation. These observables are, for example, the energy $a^4 E(t)$ given in formula (2.1) in [37] and its related observable

$$W(t) = t\,\frac{d}{dt}\Big\{t^2 \langle E(t)\rangle\Big\}$$

as proposed in [7].

In this work, two different gauge-invariant energies are considered, i.e. the Wilson energy $E_\square$ and the more symmetric field strength energy E_{FT}. Thereby, the Wilson energy is summed over all oriented plaquettes in anti-clockwise direction and is defined as

$$a^4 E_\square(t) = 2 \sum_{x,\mu<\nu} \mathrm{ReTr}\Big\{1 - \mathcal{P}_{x,\mu\nu}\big([V(t)]\big)\Big\}. \tag{2.81}$$

The field strength energy $a^4 E_{FT}(t)$ depends on the symmetric field strength tensor $G_{\mu\nu}$ as described in [37].

Field Strength Tensor. The field-strength tensor $G_{\mu,\nu}$ is used in the computation of the expectation values of the Wilson energy gained from the Wilson flow. It is the sum of all smallest loops starting and ending at site x. It reads

$$a^4 E_{FT}(t) = \frac{1}{4} G^A_{\mu\nu} G^A_{\mu\nu} \tag{2.82}$$

which can be computed as

$$a^4 E_{FT}(t) = -\frac{1}{2} \sum_{x,\mu\neq\nu} Tr\big(G_{\mu\nu}(x) G_{\mu\nu}(x)\big), \qquad G_{\mu\nu}(x) = \{Q_{\mu\nu}(x)\}_{TA}. \tag{2.83}$$

Here, $Q_{\mu\nu}(x)$ is the average of the four plaquettes in the (μ, ν) plane which have the same orientation and start and end at x as shown in figure 1 of [37]:

$$\begin{aligned} Q_{\mu,\nu}(x) = \frac{1}{4}\Big\{ & V_{x,\mu} V_{x+a\hat{\mu},\nu} V^{-1}_{x+a\hat{\nu},\mu} V^{-1}_{x,\nu} \\ & + V_{x,\nu} V^{-1}_{x+a\hat{\nu}-\hat{\mu},\mu} V^{-1}_{x-a\hat{\mu},\nu} V_{x-a\hat{\mu},\mu} \\ & + V^{-1}_{x-a\hat{\mu},\mu} V^{-1}_{x-a(\hat{\mu}-\hat{\nu}),\nu} V_{x-a\hat{\mu}-\hat{\nu},\mu} V_{x-a\hat{\nu},\nu} \\ & + V^{-1}_{x-a\hat{\nu},\nu} V_{x-a\hat{\nu},\mu} V_{x+a(\hat{\mu}-\hat{\nu}),\nu} V^{-1}_{x,\mu} \Big\} \,. \end{aligned} \tag{2.84}$$

It is important that the field strength tensor $G_{\mu,\nu}$ does not simply contain the plaquettes as defined in equation (2.1) but the product of contiguous link matrices starting and ending at site x.

The different energies take approximately the same values. The computation of the plaquette energy is easier whereas the field strength energy is more symmetric and thus more accurate.

Critical Temperature. On the lattice, there is a critical temperature such that the system behaves totally different below and above this value. We find out [66] that the critical temperature can be detected using the Wilson flow. In doing so, the Wilson energy $a^4 E_\square$ or $a^4 E_{FT}$ is split into a temporal and spatial part E_{st} and E_{ss} which consist of plaquettes with space-time indices (i.e. E_{st}) and space-space indices (i.e. E_{ss}). For a Wilson Flow, both energy parts E_{ss} and E_{st} behave similar for temperatures below the critical one and totally different for temperatures above the critical one. Thus, we developed the energy difference method based on the difference in the spatial and temporal part of the energies of the Wilson flow. The method and its results are described in [66] and with more details also in chapter 8.

2.5 Summary

In Lattice QCD, expectation values are computed using the Hybrid Monte Carlo (HMC) algorithm which is a combination of a Metropolis and Molecular Dynamics step. The HMC aims at reaching the equilibrium distribution, i.e. the fixed point of the Markov chain. In this context, the Hamiltonian equations of motion have to be solved numerically in the Molecular Dynamics step. They have the shape of a coupled Lie group / Lie algebra differential equation

$$\dot{Y}(t) = A(t) \cdot Y(t), \tag{2.85a}$$

$$\dot{A}(t) = F\big([Y(t)]_s\big)\,. \tag{2.85b}$$

Here, $Y(t)$ is an element of the special unitary Lie group $\mathrm{SU}(N, \mathbb{C})$, $A(t)$ an element of the associated special unitary Lie algebra $\mathfrak{su}(N, \mathbb{C})$,

$$[Y(t)]_s \in \mathrm{SU}(N, \mathbb{C})^{1+6(d-1)} \tag{2.86}$$

a set of $1 + 6(d-1)$ elements of $\mathrm{SU}(N, \mathbb{C})$ and the function

$$F : \mathrm{SU}(N, \mathbb{C})^{1+6(d-1)} \to \mathfrak{su}(N, \mathbb{C}) \tag{2.87}$$

maps an element from a Lie group and its staples to its associated Lie algebra. This means, equation (2.85a) is a scalar differential equation on a Lie group – in section 3.1.1, it will be shown that it has to be solved with a Lie group method.

On the other hand, equation (2.85b) can be solved with any general numerical integration method because the second equation is a differential equation in a Lie algebra which is a linear space. So, any numerical method known for the Abelian case can be employed. The only difficulty is that equation (2.85b) depends on $1 + 6(d-1)$ elements which causes no problems for a single equation. For the set of n_l coupled equations of motion, we have to care about the coupling. For the method's development, we neglect this fact and concentrate on single differential equations.

It is very important that the numerical integration scheme applied on (2.85) fulfills the geometric properties time-reversibility and volume-preservation. Otherwise, the Markov chain would not reach the correct equilibrium distribution such that the desired expectation value would not be computed correctly. So, we concentate on the development of geometric numerical integration schemes in chapter 5.

Another differential equation in the context of Lattice QCD is the Wilson flow (2.80) having the shape

$$\dot{V}(t) = Z(t) \cdot V(t), \tag{2.88a}$$

$$Z(t) = F\big([V(t)]_s\big) \tag{2.88b}$$

similar to equations (2.85). Also here, $V(t)$ is an element of the Lie group $\mathrm{SU}(N, \mathbb{C})$,

$Z(t)$ an element of its associated Lie algebra $\mathfrak{su}(N, \mathbb{C})$ and F is given by the mapping (2.87). This implies that $\dot{A}(t)$ and $Z(t)$ can be computed the same way. Thus, equation 2.88 is not a single but a coupled differntial equation. If we also neglect the coupling in this part and consider $Z(t)$ as already computed Lie-algebra valued variable, equation (2.88a) and (2.85a) have the same shape.

So, both the Hamiltonian equations of motion and the Wilson flow have in common that they are solved numerically with a Lie group method. Additionally, the Hamiltonian equations of motion need a geometric numerical integration method including time-reversibility and volume-preservation.

Outlook on the next chapters. Numerical methods for differential equations on Lie groups are considered in the next chapters 3, 4 and 5. Starting with some theory about differential equations on Lie groups and its numerical integration schemes described in chapter 3, some new methods are described in chapter 4: a step size control for Munthe-Kaas Runge-Kutta schemes and a local parameterization usingthe Cayley transform. Furthermore,some new geometric schemes are developed in chapter 5: the Cayley-Leapfrog scheme, symmetric partitioned Munthe-Kaas Runge-Kutta schemes and time-reversible projection schemes. Finally, these new methods are simulated for lattice gauge fields in chapter 7 and 8.

3

Chapter 3

Numerical Integration of Differential Equations on Lie Groups

This chapter deals with the numerical solution of differential equation in matrix Lie groups. Let G be a matrix Lie group and $\mathfrak{g}$ its associated matrix Lie algebra. Now, we assume that $Y(t)$ is an element of G and $A(Y(t))$ an element of $\mathfrak{g}$ which depends somehow on $Y(t)$. We consider the single scalar differential equation

$$\dot{Y}(t) = A(Y(t)) \cdot Y(t) \tag{3.1}$$

which is already known from the previous chapter for the special case $G = \mathrm{SU}(N, \mathbb{C})$. This equation occurs in the set of Hamiltonian equations of motion and in a slightly modified way also in the set of differential equations describing the Wilson flow.

We will see that equation (3.1) is a differential equation on a Lie group, i.e., its result has to be in the Lie group, again. So, the differential equation has to be solved with a numerical integration scheme for Lie groups based on a local parameterization – which are also investigated in this chapter. The Lie-Euler method, Crouch-Grossmann Runge-Kutta schemes and Munthe-Kaas Runge-Kutta methods are mentioned and shortly described as examples of numerical integration schemes on Lie groups.

This chapter is organized in two parts. First, there is a section 3.1 concerning the foundations of numerical integration on Lie groups. It covers differential equations on Lie groups in subsection 3.1.1 as well as the local parameterization used for the numerical integration on Lie groups in subsection 3.1.2. Furthermore, Runge-Kutta methods for the Abelian case and its convergence orders are mentioned in subsection 3.1.3. In this part, examples for Runge-Kutta schemes of convergence order one, two and three are given. Then, the focus is put on well-known Runge-Kutta schemes for Lie groups in a 2nd section 3.2. Here, the Lie-Euler method, Crouch-Grossmann Runge-Kutta schemes and Munthe-Kaas Runge-Kutta methods are mentioned and shortly described in the subsections 3.2.1, 3.2.2 and 3.2.3. All paragraphs close with examples of convergence order one, two and three for the Abelian case which are applied on these Lie group methods. Finally, the advantages and disadvantages of Crouch-Grossmann and Munthe-Kaas schemes are discussed in paragraph 3.3. In addition, the content is summarized with focus on the development of Runge-Kutta methods for Lie group problems.

3.1 Foundations

As Lie groups and Lie algebras are connected via differentiable manifolds and its tangent space, this section starts with a recapitulation of differential equations on manifolds. Then, the focus is put on local parameterization which are the theoretical background of numerical solutions of differential equations on Lie groups. Both sections are mainly taken from the book of Hairer et al [29]. Finally, Runge-Kutta methods for the Abelian case are reminded in section 3.1.3 following the book of Hairer et al [30].

First, the connection of Lie groups and Lie algebras via differential manifolds is described in subsection 3.1.1 leading to differential equations on manifolds. Then, the focus is put on the numerical solution of differential equations on manifolds in paragraph 3.1.2. Afterwards, some foundations of numerical integration with Runge-Kutta schemes are recapitulated in paragraph 3.1.3.

3.1.1 Differential Equations on Lie Groups

In this section, some theory about differential equations on Lie groups is introduced. More precisely, the definition of Lie groups, Lie algebras and tangent spaces of manifolds can be found here. Furthermore, it is shown that the solution of a differential equation on a Lie group has to be in the Lie group as well following the line of Hairer et al [29]. The differential equation $\dot{Y} = A(Y)Y$ given in equation (3.1) serves as starting point for the discussion about differential equations on matrix Lie groups. Here, Y is in the Lie group G and $A(Y)$ in its associated Lie algebra $\mathfrak{g}$ which are defined as follows:

Definition 3.1 (Lie Group, [29]). A Lie group G is a group which is also a differentiable manifold. Its product is a differentiable mapping $G \times G \mapsto G$.

Definition 3.2 (Lie Algebra). Let F be a field and $\mathfrak{g}$ be a vector space over F. We say that $\mathfrak{g}$ is a Lie algebra if there exists a map

$$\mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}, \quad (x, y) \mapsto [x, y] \tag{3.2}$$

such that

- $[x, y]$ is bilinear over F.
- $[x, x] = 0$ for all $x \in \mathfrak{g}$.
- the Jacobi identity

  $$[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0 \tag{3.3}$$

  holds for all $x, y, z \in \mathfrak{g}$.

Lie groups G and Lie algebras $\mathfrak{g}$ are connected via differentiable manifolds and its tangent spaces. For a deeper understanding on differential equations on manifolds, some theory about differential manifolds, its tangent spaces and their connection is needed:

Definition 3.3 (Differentiable Manifold, [29])**.** Let $g : U \mapsto \mathbb{R}^m$ be a differentiable function with U being a neighborhood of the point $a \in \mathbb{R}^n$. Assume that $g(a) = 0$ and $g'(a)$ has full rank m. Then,

$$\mathcal{M} = \{y \in U\,;\ \ g(y) = 0\} \tag{3.4}$$

is a differentiable manifold.

Thus, the manifold contains a curve (or a surface) γ and the tangent space is a tangent (or plane) described by $\dot{\gamma}(0)$ passing through the contact point $a = \gamma(0) \in \mathcal{M}$:

Definition 3.4 (Tangent Space of a Manifold, [29])**.** ,

$$T_a\mathcal{M} = \{v \in \mathbb{R}^n : \exists\, \gamma : (-\varepsilon, \varepsilon) \mapsto \mathbb{R}^n \ \text{ with } \ \gamma(t) \in \mathcal{M}\ \forall\, t,\ \gamma(0) = a, \dot{\gamma}(0) = v\}\,.$$

It is known that the Lie algebra $\mathfrak{g}$ can be identified with the tangent space $T_I G$ at the identity of the differentiable manifold G. In general, the tangent space through the contact point $Y \in G$ is called $T_Y G$. For a Lie-algebra element $A \in \mathfrak{g}$, the problem $\dot{Y} = A \cdot Y$ with $Y \in G$ is a differential equation on the manifold G.

Theorem 3.5 (Differential Equation on a Manifold, [29])**.** Let $\mathcal{M}$ be a manifold of $\mathbb{R}^n$. The problem $\dot{Y} = AY$ is a differential equation on the manifold $\mathcal{M}$ if and only if $AY \in T_Y\mathcal{M}\ \forall\, Y \in \mathcal{M}$.

Proof. Let A be an element of the Lie algebra $\mathfrak{g}$. Due to $\mathfrak{g} = T_I G$, there exists a path $\gamma \in G$ with $\gamma(0) = I$ and $\dot{\gamma}(0) = A$. A multiplication with Y leads to the element $AY \in T_Y\mathcal{M}$ as follows: For the element AY there exists a different path $\widetilde{\gamma} := \gamma \cdot Y$ with $\widetilde{\gamma}(0) = Y$ and $\dot{\widetilde{\gamma}} = AY$. By the definition of the tangent space, AY is an element of the tangent space $T_Y\mathcal{M}$. This leads to the following theorem: □

Theorem 3.6 (Differential Equations on a Matrix Lie Group, [29])**.** Let G be a matrix Lie group and $\mathfrak{g}$ its Lie algebra. If $A(Y) \in \mathfrak{g}$ for all $Y \in G$ and if $y_n \in G$, then the solution of $\dot{Y} = A(Y)Y$ satisfies $Y(t) \in G$ for all t.

Proof. As is the proof of 3.5, the element $A(Y)Y$ is an element of the tangent space $T_Y G$ Thus, the result of $\dot{Y} = A(Y)Y$ has to be in the Lie group G. □

3.1.2 Numerical Integration via Local Parameterization

Differential equations on manifolds can be solved using a local parameterization of the element of the manifold, i.e.

$$Y(t) = \Psi(\Omega(t))\,. \tag{3.5}$$

Differentiation with respect to time leads to

$$\frac{\mathrm{d}}{\mathrm{d}\,t}Y(t) = \frac{\mathrm{d}}{\mathrm{d}\,t}\Psi(\Omega(t)) = \frac{\mathrm{d}}{\mathrm{d}\,\Omega}\Psi(\Omega(t))\,\dot{\Omega}(t)\,. \tag{3.6}$$

Combining equation (3.6) with the initial differential equation (3.1) and the introduced local parameterization (3.5) leads to

$$\frac{\mathrm{d}}{\mathrm{d}\,\Omega}\Psi(\Omega(t))\,\dot{\Omega}(t) = A \cdot \Psi(\Omega(t))\,. \tag{3.7}$$

Here, the difficulty is that the inverse of the expression $\frac{\mathrm{d}}{\mathrm{d}\,\Omega}\Psi(\Omega(t))$ has to be found to set up the differential equation for $\dot{\Omega}(t)$. This can be done via the definition of

$$\left(\frac{\mathrm{d}}{\mathrm{d}\,\Omega}\Psi(\Omega)\right)H := \Big(\mathrm{d}\,\Psi_\Omega(H)\Big) \cdot \Psi(\Omega) \tag{3.8}$$

which is a theoretical construct with a special shape. Here, the expression $(\frac{\mathrm{d}}{\mathrm{d}\,\Omega}\Psi(\Omega))H$ on the left-hand side means that the so-far unknown $\frac{\mathrm{d}}{\mathrm{d}\,\Omega}\Psi(\Omega)$ is applied on an element H of the tangent space of the manifold. The shape of the right-hand side is similar to the right-hand side of equation (3.7). It is defined as the also unknown expression $\mathrm{d}\,\Psi_\Omega(H)$ times the local parameterization $\Psi(\Omega)$.

Inserting (3.8) on the left-hand side of (3.7) leads to

$$\Big(\mathrm{d}\,\Psi_\Omega(\dot{\Omega})\Big) \cdot \Psi(\Omega) = A \cdot \Psi(\Omega) \tag{3.9}$$

such that

$$A = \mathrm{d}\,\Psi_\Omega(\dot{\Omega}) \tag{3.10}$$

holds since the term $\Psi(\Omega)$ introduced in equation (3.8) can be neglected on both sides. Applying the inverse of $\mathrm{d}\,\Psi_\Omega$ leads to

$$\dot{\Omega} = \Big(\mathrm{d}\,\Psi_\Omega^{-1}\Big)A. \tag{3.11}$$

Hence, we obtain a differential equation for $\dot{\Omega}$ such that an expression for the new unknown $\mathrm{d}\,\Psi_\Omega^{-1}(A)$ has to be found. For initial-value problems, the parameterization has to be chosen such that the initial values agree with those used in the parameterization, i.e. $\Psi(\Omega_n) = Y_n$. This reasoning is based on the book of Hairer et al [29] and leads to the local coordinate approach.

Assuming that this happened, the numerical integration on manifolds, i.e. the mapping from $Y_n \mapsto Y_{n+1}$, works as described in section IV of [29]:

- define a differential equation $\dot{\Omega} = f(\Omega)$
- solve $\Omega_{n+1} = \Phi(\Omega_n)$ with numerical scheme Φ and initial value Ω_n

- define the numerical solution in the Lie group by $Y_{n+1} = \Psi(\Omega_{n+1})$

The important things are that the local parameterization maps an element from the tangent space of the manifold to the manifold and that the initial values of the differential equation are mapped to the ones of the origin differential equation. So, the differential equation is solved in the linear space of the tangent space and then mapped back on the manifold via the parameterization . For initial value problems $\dot{Y} = AY$ with given $Y(t_0) = Y_n$, the mapping Ψ and the initial value $\Omega(t_0) = \Omega_n$ have to be chosen in a consistent way such that $\Psi(\Omega_n) = Y_n$ holds.

For Lie groups, the local parameterization is chosen as mapping $\Psi : \mathfrak{g} \mapsto G$ from the Lie algebra $\mathfrak{g}$ to the Lie group G. So, the differential equation is solved in the linear space of the Lie algebra and its result is mapped to the Lie group. The mapping can be, for example, the exponential map $\Psi(\Omega) = \exp(\Omega)$ or the Cayley transform $\Psi(\Omega) = (I - \Omega)^{-1}(I + \Omega)$ for quadratic Lie groups as explained in [29]. Usually, the exponential map is used, therefore it is mentioned in the next paragraph.

Exponential Function as Local Parameterization. Exponential functions are widely used in the context of physics as well as for Lie group methods. The exponential function has the property that it maps an element from a Lie algebra to a Lie group for all Lie groups. This is expressed in the next lemma:

Lemma 3.7 (Exponential Map, [29])**.** Consider a matrix Lie group G and its Lie algebra $\mathfrak{g}$. The matrix exponential is a map

$$\exp : \mathfrak{g} \mapsto G, A \mapsto \exp(A) = \sum_{k \geq 0} \frac{1}{k!} A^k \tag{3.12}$$

i.e., for $A \in \mathfrak{g}$ we have $\exp(A) \in G$. Moreover, exp is a local diffeomorphism in a neighborhood of $A = 0$.

With focus on usage of the exponential function as local parameterization, also the inverse of the derivative of the exponential map $\mathrm{d}\exp_\Omega^{-1}(h)$ is of interest. This expression is given as

$$\mathrm{d}\exp_\Omega^{-1}(H) = \sum_{k=0}^{\infty} \frac{B_k}{k!} \operatorname{ad}_\Omega^k(H) \tag{3.13}$$

with Bernoulli numbers B_k and adjoint operator $\operatorname{ad}_\Omega(A)$ stated in the following definitions 3.8 and 3.9.

Definition 3.8 (Bernoulli Numbers, [29], [5])**.** The Bernoulli numbers B_k are defined by

$$\sum_{k \geq 0} \frac{B_k}{k!} x^k = \frac{x}{\exp(x) - 1} \, . \tag{3.14}$$

The first few Bernoulli numbers are $B_0 = 1, B_1 = -\frac{1}{2}, B_2 = \frac{1}{6}, B_3 = 0$.

Definition 3.9 (Adjoint Operator, [29], [61])**.** The adjoint operator $\mathrm{ad}_\Omega(A)$ is a linear operator

$$\mathrm{ad} : \mathfrak{g} \mapsto \mathfrak{g}, \quad A \mapsto \mathrm{ad}_\Omega(A) = [\Omega, A] \tag{3.15}$$

for a fixed Ω and uses matrix commutators $[\Omega, A] = \Omega A - A\Omega$. The adjoint operator can be used iteratively, such that ad_Ω^k denotes the k-th iterated application of the linear operator ad_Ω. It holds

$$\mathrm{ad}_\Omega^{k+1} = \left[\Omega, \mathrm{ad}_\Omega^k\right]. \tag{3.16}$$

By convention, $\mathrm{ad}_\Omega^0(A)$ is set to A.

The dexpinv equation. The inverse of the derivative of the exponential function plays a special role in the development of numerical integration schemes for differential equations on Lie groups. According to Iserles [32], we label equation (3.13) as the *dexpinv* equation. In general, it is an infinite sum which can not be written in a closed form. Nevertheless, there are Lie groups in which the *dexpinv* equation can be denoted in a closed form, for example, for $SO(3)$ and $SU(2)$. For the development of the Lie group methods, we assume that no closed form of the *dexpinv* equation is known. Thus, the discussion of the model error in Munthe-Kaas Runge-Kutta methods is based on equation (3.13).

3.1.3 Runge-Kutta Methods in the Abelian Case

Runge-Kutta methods are constructed by Runge, Heun and Kutta around the year 1900 as described by Hairer et all in section II.1 and II.2 in [30]. In general, Runge-Kutta methods are defined for the Abelian case $\mathbb{R}^n$. They represent a general form for one-step schemes.

Definition 3.10 (Runge-Kutta Method, [30])**.** Let a differential equation

$$\dot{y} = f(y) \quad \text{with initial value} \quad y(t_0) = y_n \tag{3.17}$$

be given. The method

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i \qquad \text{with} \qquad k_i = f(y_n + h\sum_{j=1}^{s} a_{ij}k_j), \quad i = 1, \dots, s, \tag{3.18}$$

is called Runge-Kutta method with s stages. Here, the stage number s and the coefficients a_{ij}, b_i for $i, j = 1, \dots, s$ are given. The Runge-Kutta scheme is an explicit method if it holds $a_{ij} = 0$ for $i \leq j$, otherwise it is an implicit method.

Runge-Kutta methods are characterized by its coefficients. According to Butcher [11], they can be described in a convenient way using Butcher tableaus:

Definition 3.11 (Butcher Tableau, [11], [30]). The coefficients a_{ij} and b_i, $i, j = 1, \ldots, s$ of the Runge-Kutta method described above can be denoted in the following Butcher tableau

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \ldots & a_{1s} \\ c_2 & a_{21} & a_{22} & \ldots & a_{1s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \ldots & a_{ss} \\ \hline & b_1 & b_2 & \ldots & b_s \end{array}$$

in a compact way. The elements c_i are defined as $c_i = \sum_j a_{ij}$ for $i = 1, \ldots, s$.

Convergence Order. In this paragraph, the definitions for the terms consistency and convergence order are stated since they are used in the rest of this thesis. The convergence order depends on the local discretization error, which is defined as the difference of the numerical one-step scheme $\Phi_h(y_n)$ and the true solution $y(t)$:

Definition 3.12 (Local discretization error, [30]). Let $y(x)$ be the exact solution of the differential equation (3.17) and $y_{n+1} = \Phi_h(y_n)$ be the numerical approximation after one step with step size h. The local discretization error is defined as

$$\tau(h) := y_{n+1} - y(t_0 + h)\,. \tag{3.19}$$

In practice, $\tau(h)$ is computed as the difference between the numerical approximation and the Taylor expansion of the exact solution. The consistency is based on the local discretization error as follows:

Definition 3.13 (Consistency, [30]). A method Φ_h is consistent if the local discretization error (3.19) uniformly tends to zero:

$$||\tau(h)|| \leq \epsilon(h), \qquad\qquad \lim_{h \to 0} \epsilon(h) = 0\,. \tag{3.20}$$

The method is consistent of order p if

$$||\tau(h)|| = \mathcal{O}(h^{p+1})\,. \tag{3.21}$$

The convergence for a consistent one-step scheme itself depends on the global discretization error:

Definition 3.14 (Global discretization error and convergence, [30]). Let $y(T)$ be the exact solution of the differential equation (3.17) at the point $T > t_0$, and y_{n+M} be the numerical approximation after M steps with step size $h_i = t_{i+1} - t_i$, $i = 0, \ldots, M - 1$.

Then, the global error

$$E_M = y_{n+M} - y(T)$$

is defined as the difference of the numerical approximation y_{n+M} obtained after M steps and the true solution $y(T)$.

For $M \to \infty$, we assume that $h_i \to 0$. The method is convergent if it holds

$$\lim_{M \to \infty} E_M = 0.$$

Finally, one is interested in the convergence order of the numerical one step scheme:

Theorem 3.15 (Convergence Order)**.** Let a differential equation (3.17) be given. Assume that $f(y)$ is continuous in y and satisfies the Lipschitz condition with respect to y. Furthermore, let Φ_h a consistent one-step method of consistency order p.

Then, the method is of convergence order p if

$$||E_M|| = \mathcal{O}(h^p)$$

for $h = \max\{h_0, \dots, h_{M-1}\}$, $h_i = t_{i+1} - t_i$ and $i = 0, \dots, M-1$.

Thus, for one-step schemes, convergence follows from consistency. Moreover, the convergence order can be obtained by investigating the consistency order. At the end, order conditions can be computed by a comparison of the numerical approximation with the Taylor series of the exact solution. For Runge-Kutta methods, the convergence order depends on the coefficients of the scheme. Moreover, there are always the same order conditions depending on the coefficients of the scheme.

Definition 3.16 (Order Conditions for Runge-Kutta Methods, [30])**.** A Runge-Kutta method has convergence order p if the coefficients b_i, a_{ij} and $c_i = \sum_j a_{ij}$ fulfill the following order conditions up to order p:

$$p = 1: \qquad \sum_i b_i = 1, \tag{3.22a}$$

$$p = 2: \qquad \sum_i b_i c_i = \frac{1}{2}, \tag{3.22b}$$

$$p = 3: \qquad \sum_i b_i c_i^2 = \frac{1}{3}, \qquad \sum_{i,j} b_i a_{ij} c_j = \frac{1}{6}. \tag{3.22c}$$

In practice just these order conditions are checked instead of performing Taylor expansions.

Examples. In the following, some simple (explicit) examples are given for Runge-Kutta methods of convergence order $p = 1$, $p = 2$ and $p = 3$: the explicit Euler scheme and the methods of Heun with 2 and 3 stages as mentioned in [30]. Its

	s	p	b_i	a_{ij}
Explicit Euler	1	1	$b_1 = 1$	
Heun	2	2	$b_1 = b_2 = \frac{1}{2}$	$a_{21} = 1$
Heun	3	3	$b_1 = \frac{1}{4}, b_2 = 0, b_3 = \frac{3}{4}$	$a_{21} = \frac{1}{3}$, $a_{32} = \frac{2}{3}$

Table 3.1: *Some simple Runge-Kutta methods of convergence order 1,2,3.* – s is the stage number, p the convergence order, b_i and a_{ij} the coefficients

coefficients are stated in table 3.1. In section 3.2, these examples are used to illustrate the Crouch-Grossmann and Munthe-Kaas scheme.

p=1. The explicit Euler method is the most simple Runge-Kutta scheme with convergence order $p = 1$. It has $s = 1$ stages and just the non-zero coefficient $b_1 = 1$. It is given by

$$y_{n+1} = y_n + hk_1 \quad \text{with} \quad k_1 = f(y_n). \tag{3.23}$$

p=2. The method of Heun with $s = 2$ stages and non-zero coefficients $b_1 = b_2 = \frac{1}{2}, a_{21} = 1$ has convergence order $p = 2$. It reads for the general case

$$y_{n+1} = y_n + \tfrac{h}{2}(k_1 + k_2) \quad \text{with} \quad k_1 = f(y_n) \quad \text{and} \quad k_2 = f(y_n + hk_1). \tag{3.24}$$

p=3. The method of Heun with $s = 3$ stages and coefficients $b_1 = \frac{1}{4}, b_3 = \frac{3}{4}$, $a_{21} = \frac{1}{3}$, $a_{32} = \frac{2}{3}$ has convergence order $p = 3$. It reads

$$\begin{aligned} y_{n+1} &= y_n + \tfrac{h}{4}(k_1 + 3k_3) \qquad (3.25)\\ \text{with} \quad k_1 = f(y_n), &\quad k_2 = f(y_n + \tfrac{h}{3}k_1) \quad \text{and} \quad k_3 = f(y_n + \tfrac{2}{3}hk_2). \end{aligned}$$

The convergence orders can simply be checked using via definition 3.16.

3.2 Runge-Kutta Methods for Lie Groups

In this section, some well-known numerical integration methods for the Lie group initial value problem

$$\dot{Y}(t) = A(Y(t)) \cdot Y(t) \quad \text{with initial values} \quad Y(t_0) = Y_n \tag{3.26}$$

are described. Here, Y is in the Lie group G, A in the Lie algebra $\mathfrak{g}$ and $\dot{Y} = AY$ in $T_Y G$ which is the tangent space of the Lie group at point Y. We know from theorem 3.6 that the result $Y(t)$ has to be in the Lie group G. So, the numerical integration scheme has to ensure this.

We start with the Lie-Euler method as introductory example in paragraph 3.2.1. Then, Crouch-Grossmann (CGRK) and Munthe-Kaas Runge-Kutta (MKRK) schemes are mentioned in 3.2.2 and 3.2.3. Both methods are illustrated using the simple examples of convergence order one, two and three given in table 3.1.

3.2.1 Lie-Euler

This paragraph starts with a demonstration that standard numerical integration methods have to be adapted to the Lie group structure. Let the initial value $Y_n \in G$ be given. For example, the standard Euler method

$$Y_{n+1} = Y_n + h\dot{Y}_n \tag{3.27}$$

cannot be used for matrix Lie groups since the Lie group is just closed under multiplication. This means that, in general, the result of (3.27) would not be in the Lie group any more. Nevertheless, it gives a first hint how to solve differential equations on Lie groups. Considering the Lie group initial value problem (3.26), the standard Euler method can be rewritten as

$$Y_{n+1} = Y_n + h\dot{Y}_n = Y_n + hA(Y_n)Y_n = \big(I + hA(Y_n)\big)\,Y_n. \tag{3.28}$$

Comparing the standard Euler method (3.28) to the truncated series of the exponential map (3.12)

$$\exp\big(hA(Y_n)\big) = \sum_{k\geq 0} \frac{1}{k!}\big(hA(Y_n)\big)^k = \big(I + hA(Y_n) + \frac{h^2}{2}A(Y_n)^2\big) + \mathcal{O}(h^3), \tag{3.29}$$

leads to the result that they coincide up to and including terms of order h^2. This gives rise to the Lie-Euler method stated in definition 3.17:

Definition 3.17 (Lie Euler method)**.** Let the initial value problem (3.26)

$$\dot{Y}(t) = A(Y(t)) \cdot Y(t) \quad \text{with initial values} \quad Y(t_0) = Y_n$$

with $Y, Y_n \in G$, $A(Y) \in \mathfrak{g}$ be given. Then, the method

$$Y_{n+1} = \exp\big(hA(Y_n)\big)\,Y_n. \tag{3.30}$$

is called Lie-Euler method.

It is the most simple numerical integration method for Lie groups and has convergence order one which easily can be seen by a comparison of the Taylor expansions of the exact solution $Y(t_0 + h)$ with the numerical approximation Y_{n+1}. The Taylor expansion of the exact solution leads to

$$Y(t_0+h) = Y_n + hA(Y_n)Y_n + \frac{h^2}{2}\ddot{Y}(t_0) + \mathcal{O}(h^3). \tag{3.31}$$

Here, the differential equation (3.26) with initial values $Y(t_0) = Y_n$ are used. The numerical approximation Y_{n+1} uses a part of the exponential function (3.29):

$$\begin{aligned} Y_{n+1} &= \exp\big(hA(Y_n)\big)Y_n \\ &= \big(I + hA(Y_n) + \frac{h^2}{2}(A(Y_n))^2\big)Y_n + \mathcal{O}(h^3) \\ &= Y_n + hA(Y_n)Y_n + \frac{h^2}{2}(A(Y_n))^2Y_n + \mathcal{O}(h^3). \end{aligned} \tag{3.32}$$

So, equations (3.31) and (3.32) coincide up to and including terms of order h. For order 2, the terms $\ddot{Y}(t_0)$ will not coincide with $(A(Y_n))^2Y_n$ in general. Thus, the error of the numerical scheme is of order $p = 2$ and the Lie Euler scheme is a numerical approximation of order $p = 1$ which is also described by Celledoni [12] in a slightly different way.

The Lie-Euler method can be generalized to Runge-Kutta methods of Crouch-Grossmann type.

3.2.2 Crouch-Grossmann Runge-Kutta

Crouch-Grossmann Runge-Kutta methods have been developed in 1993 by Crouch and Grossmann [15]. They are based on the idea to take explicit Runge-Kutta methods and replace the update step with a Lie-Euler step.

Definition 3.18 (Crouch-Grossman method, [29])**.** Let b_i and a_{ij} with $i, j = 1, \dots, s$ be real numbers. An explicit s-stage Crouch-Grossman method is given by

$$Y_{n+1} = \exp(hb_sK_s) \ \cdot \ldots \cdot \ \exp(hb_1K_1)Y_n, \tag{3.33a}$$

with increments $K_i = A(Y_i)$ for $i = 1, \dots, s$ and internal stages

$$Y_i = \exp\big(ha_{i,i-1}K_{i-1}\big) \ \cdot \ldots \cdot \ \exp\big(ha_{i1}K_1\big)\ Y_n. \tag{3.33b}$$

The result Y_{n+1} is per construction in the Lie group. A big advantage of the Crouch-Grossmann method is that it is an explicit method. So, its computation is cheap and it is suitable also for the solutions of differential equations in the field of Lattice QCD. Indeed, methods of Crouch-Grossmann type are used for the computation of the Wilson flow, for example, Lüscher used them in [37]. A disadvantage of this method is that the development of higher-order methods involve additional order conditions compared to the ones of a general Runge-Kutta scheme given in definition 3.16. For example, convergence order 3 is achieved with one additional convergence order

$$\sum_i b_i c_i \Big(\frac{1}{2} b_i + \sum_{j=i+1}^{s} b_j \Big) = \frac{1}{6} \tag{3.34}$$

compared to classical Runge-Kutta methods. For the construction of a method of order 4, 13 conditions including the 8 conditions for classical Runge-Kutta schemes occur. This means, higher-order Crouch-Grossmann schemes can not use already known sets of coefficients for higher-order Runge-Kutta schemes. Nevertheless, it is possible to find coefficients for a higher-order scheme of Crouch-Grossmann type. Owren and Marthinsen list the order conditions and construct a 4th order method with $s = 5$ stages in [49].

Examples. The following examples illustrate Crouch-Grossmann schemes for convergence order one, two and three. More precisely, the explicit Euler scheme and the methods of Heun with 2 and 3 stages mentioned in table 3.1 are used here.

p=1. The one-stage Crouch-Grossmann Runge-Kutta scheme using the coefficients $b_1 = 1$, and $a_{11} = 0$ of the explicit Euler method is stated as

$$Y_{n+1} = \exp(hA(Y_n))Y_n. \tag{3.35}$$

So, the Crouch-Grossmann scheme for order $p = 1$ coincides with the Lie-Euler method (3.30).

p=2. For convergence order $p = 2$, the method of Heun with $s = 2$ stages (and non-zero coefficients $b_1 = b_2 = \frac{1}{2}$, $a_{21} = 1$) can be adapted to a Crouch-Grossmann scheme as

$$Y_{n+1} = \exp(\tfrac{h}{2} K_2) \ \cdot \ \exp(\tfrac{h}{2} K_1) \cdot Y_n, \tag{3.36a}$$

with increments $K_1 = A(Y_1)$ and $K_2 = A(Y_2)$ and internal stages

$$Y_1 = Y_n, \quad \text{and} \quad Y_2 = \exp\big(hK_1\big)\, Y_n. \tag{3.36b}$$

Since there occur no extra order conditions for $p = 1$ and $p = 2$, all classical Runge-Kutta schemes of convergence order one and two can just be adapted to CGRK schemes. This means, the explicit Euler CGRK scheme and the CGRK method of Heun with 2 stages have the same order conditions as in the classical RK scheme.

p=3. The method of Heun with 3 stages reads

$$\begin{aligned} Y_{n+1} &= \exp\Big(\tfrac{3}{4} h \cdot K_3\Big) \cdot \exp\Big(0 \cdot h \cdot K_2\Big) \cdot \exp\Big(\tfrac{h}{4} K_1\Big) \cdot Y_n \\ &= \exp\Big(\tfrac{3}{4} K_3\Big) \cdot \exp\Big(\tfrac{h}{4} K_1\Big) \cdot Y_n \end{aligned} \tag{3.37a}$$

with increments $K_1 = A(Y_1)$, $K_2 = A(Y_2)$ and $K_3 = A(Y_3)$ and internal stages

$$Y_1 = Y_n, \qquad Y_2 = \exp\Big(\tfrac{h}{3}A(Y_1)\Big)Y_n, \qquad Y_3 = \exp\Big(\tfrac{2}{3}hA(Y_2)\Big)Y_n. \tag{3.37b}$$

Here, the non-zero coefficients for the $s = 3$ stages are $b_1 = \frac{1}{4}, b_3 = \frac{3}{4}$, $a_{21} = \frac{1}{3}$, $a_{32} = \frac{2}{3}$. Due to $b_2 = 0$, the stage Y_2 does not occur in equation (3.37a) but it is needed for the computation of stage Y_3.

For convergence order 3, it has to be checked that the coefficients of the classical RK method also fulfill the additional order condition (3.34) for Crouch-Grossmann schemes. Obviously, the additional order condition (3.34)

$$b_1c_1 \cdot (\tfrac{1}{2}b_1 + b_2 + b_3) + b_2c_2 \cdot (\tfrac{1}{2}b_2 + b_3) + b_3c_3 \cdot (\tfrac{1}{2}b_3) = \frac{3}{8} \neq \frac{1}{6} \tag{3.38}$$

is not fulfilled such that the method has just convergence order $p = 2$. Nevertheless, it is possible to develop a CGRK method of convergence order 3 with $s = 3$ stages but care has to be taken for the method's coefficients.

3.2.3 Munthe-Kaas Runge-Kutta

Munthe-Kaas developed in [45] and [46] Runge-Kutta schemes for Lie groups that need no additional order conditions. The Munthe-Kaas Runge-Kutta method combines general Runge-Kutta methods with the theorem of Magnus:

Theorem 3.19 (Magnus, [39], [29])**.** Let G be a Lie group and $\mathfrak{g}$ its associated Lie algebra. The solution of the differential equation

$$\dot{Y}(t) = A(t)Y(t) \tag{3.39}$$

with $A(t) \in \mathfrak{g}$ and $Y(t) \in G$ can be written as

$$Y(t) = \exp(\Omega(t))Y_n \tag{3.40a}$$

with initial value $Y_n \in G$ and $\Omega(t)$ defined by the derivative of the inverse exponential map

$$\dot{\Omega}(t) = \mathrm{d}\exp_{\Omega}^{-1}(A(t)), \quad \Omega(t_n) = 0. \tag{3.40b}$$

As long as $||\Omega(t)|| < \pi$ the convergence of the $d\exp_{\Omega}^{-1}$ expansion

$$\mathrm{d}\exp_{\Omega}^{-1}(A) = \sum_{k\geq 0} \frac{B_k}{k!}\,\mathrm{ad}_{\Omega}^{k}(A) \tag{3.40c}$$

with Bernoulli numbers B_k and adjoint operator $\mathrm{ad}_{\Omega}(A) = [\Omega, A]$ is assured.

Proof. The proof of this theorem follows the idea of numerical integration via local parameterization described in section 3.1.2 and is taken from section IV.7 of [28].

We compute the derivative of equation (3.40a),

$$\dot{Y}(t) = \Big(\frac{d}{d\Omega}\exp(\Omega(t))\Big)\dot{\Omega}(t)Y_n \overset{(3.8)}{=} \mathrm{d}\exp_{\Omega}(\dot{\Omega}(t))\exp(\Omega(t))Y_n\,, \tag{3.41}$$

and compare it with with equation (3.39). So, we yield

$$A = \mathrm{d}\exp_{\Omega}(\dot{\Omega}(t))\,. \tag{3.42}$$

Afterwards, we apply the inverse operator $(\mathrm{d}\exp_{\Omega})^{-1}$ and reach equation (3.40b). The initial value Ω_n takes the value $\Omega_n = 0$ to be consistent with the initial value Y_n. □

The Magnus approach is formulated for general Lie group problems $Y(t) = A(t)Y(t)$ with $A \in \mathfrak{g}$ and $Y \in G$. Here, the Lie algebra element A may depend on the Lie group element Y or not. However, the term $A(Y(t))$ in (3.26) indicates that the Lie algebra element A has a mathematical relation to the Lie group element Y. In the context of the shape of differential equations on Lie groups, $A(t)$, $A(Y(t))$ or even $A_Y(t)$ can be used as synonyms for each other:

$$A(t) \equiv A(Y(t)) \equiv A_Y(t) \tag{3.43}$$

just the affiliation to the Lie group is of importance.
The definitions of the Bernoulli numbers B_k and the adjoint operator $\mathrm{ad}_{\Omega}(A)$ are already mentioned in definition 3.8 and 3.9 of section 3.1.1.

The theorem of Magnus contains the differential equation

$$\dot{\Omega}(t) = \mathrm{d}\exp_{\Omega}^{-1}(A(t)) = \sum_{k\geq 0}\frac{B_k}{k!}\,\mathrm{ad}_{\Omega}^{k}(A(t)), \quad \Omega(t_0) = 0 \tag{3.44}$$

whose solution $\Omega(t)$ is used as local parameterization $Y(t) = \exp(\Omega(t))\,Y_n$. Both, $\Omega(t)$ and $\dot{\Omega}(t)$ are situated in the Lie algebra $\mathfrak{g}$ which is also a linear vector space. Thus, the differential equation (3.44) can be solved in the Lie algebra using any general Runge-Kutta scheme. For the numerical simulation, the infinite series of the inverse of the derivative of the exponential map (called *dexpinv* equation) has to be truncated in a suitable manner such that the convergence order of the Runge-Kutta scheme will be preserved:

$$\dot{\Omega}(t) = \mathrm{d}\exp_{\Omega}^{-1}(A(t)) \approx \sum_{k\geq 0}^{q}\frac{B_k}{k!}\,\mathrm{ad}_{\Omega}^{k}(A(t)) =: f_q(\Omega, A)\;. \tag{3.45}$$

The truncation index q induces a model error because $f_q(\Omega, A)$ is just an approximation of the differential equation $\dot{\Omega}(t) = \mathrm{d}\exp_{\Omega}^{-1}(A(t))$. This model error should be smaller or equal than the convergence order of the numerical method used for the detection of the solution Ω. The model error is investigated in detail, for example, by Striebel et al in [60].

Munthe-Kaas developed an algorithm for the numerical computation of the result of the Lie group differential equation $\dot{Y} = AY$ in [46]. The algorithm is stated next followed by theorem 3.21 about the order of the truncation index. It can also be found in [29].

Algorithm 3.20. (Munthe-Kaas Runge-Kutta Method, [29])
Consider the problem (3.1) with $A(Y) \in \mathfrak{g}$ for $Y \in G$. Assume that the initial value $Y(t_0) := Y_n$ lies in the Lie group G. Then, the step $Y_n \mapsto Y_{n+1}$ is defined as follows:

1. Consider the differential equation

$$\mathrm{d}\exp_{\Omega}^{-1}(A) \approx \dot{\Omega} = \sum_{k \geq 0}^{q} \frac{B_k}{k!} \operatorname{ad}_{\Omega}^{k}(A) =: f_q(\Omega, A) \tag{3.46}$$

   with initial value $\Omega(t_0) = \Omega_n = 0$.

2. Apply a Runge-Kutta method (explicit or implicit) with initial values $\Omega_n = 0$ and Y_n to get an approximation $\Omega_{n+1} \approx \Omega(t_0 + h)$.

3. Define the numerical solution by

$$Y_{n+1} = \exp(\Omega_{n+1})\, Y_n. \tag{3.47}$$

Note that the differential equation $\dot{\Omega}$ stated in equation (3.46) is slightly modified compared to equation (3.45). It includes the model error.

Theorem 3.21 (Truncation Index, [29])**.** *The method of algorithm 3.22 is of order p if the truncation index in (3.46) satisfies $q \geq p-2$ and the underlying Runge–Kutta method is also of (classical) order p.*

The proof of this theorem can be found in [29].

The truncation index q has to be adapted to the desired convergence order p of the scheme: it has to satisfy $q \geq p - 2$. Furthermore, the smallest possible value for q is zero. Thus, the differential equations for convergence order $p = 1$ and $p = 2$ are the same:

$$\dot{\Omega}(t) = f_0(\Omega, A) = A \tag{3.48}$$

for Lie-algebra valued terms Ω and A.

Algorithm 3.22. (Munthe-Kaas Runge-Kutta Method of Convergence Order p)
Consider the problem (3.1) with $A(Y) \in \mathfrak{g}$ for $Y \in G$. Assume that the initial value $Y(t_0) := Y_n$ lies in the Lie group G. Then, the step $Y_n \mapsto Y_{n+1}$ is defined as follows:

1. Consider the differential equation

$$\mathrm{d}\exp_{\Omega}^{-1}(A) \approx \dot{\Omega} = \sum_{k=0}^{p-2} \frac{B_k}{k!} \operatorname{ad}_{\Omega}^{k}(A) =: f_{p-2}(\Omega, A) \tag{3.49}$$

   with initial value $\Omega(t_0) = \Omega_n = 0$.

2. Compute the approximation $\Omega_{n+1} \approx \Omega(t_0 + h)$ by a Runge-Kutta scheme of convergence order p:

$$\Omega_{n+1} = \Omega_n + h \sum_{i=1}^{s} b_i K_i, \tag{3.50a}$$

$$\text{with increments} \quad K_i = f_{p-2}\Big(\bar{\Omega}_i, A(\bar{Y}_i)\Big) \tag{3.50b}$$

$$\text{and internal stages} \quad \bar{\Omega}_i = \Omega_n + h \sum_{j=1}^{s} a_{ij} K_j, \qquad \bar{Y}_i = \exp(\bar{\Omega}_i) Y_n \tag{3.50c}$$

   for $i = 1, \ldots, s$.

3. Define the numerical solution of convergence order p by

$$Y_{n+1} = \exp(\Omega_{n+1})\, Y_n$$

   as described in equation (3.47).

Examples. The influence of the truncation index, respective the convergence order, is shown in the following examples of convergence order one, two and three. Again, the coefficients of the explicit Euler scheme and the methods of Heun of order $p = 2$ and order $p = 3$ of table 3.1 are used for the approximation of the initial value problem (3.26). Starting from the equations stated in formulae (3.50), the increments K_i, $i = 1, \ldots, s$ have to be chosen properly – according to a combination of formula (3.46) and theorem 3.21 – to reach the desired convergence order p. Moreover, the initial value Ω_n is always set to zero.

p=1. For convergence order one, the Munthe-Kaas Runge-Kutta scheme using the coefficients of the explicit Euler method ($s = 1, b_1 = 1$) reads

$$\Omega_{n+1} = hK_1 = A(Y_n) \tag{3.51}$$

and leads to $Y_{n+1} = \exp(hA(Y_n))\, Y_n$. Thus, these coefficients lead to the Lie-Euler method (3.30).

p=2. The method of Heun of order $p = 2$ ($s = 2$, $b_1 = b_2 = \frac{1}{2}$, $a_{21} = 1$) uses the increments

$$K_i = f_0\Big(\bar{\Omega}_i, A(\bar{Y}_i)\Big) = A(\bar{Y}_i). \tag{3.52}$$

Finally, Ω_{n+1} is computed as

$$\Omega_{n+1} = \frac{1}{2}\big(K_1 + K_2\big) \tag{3.53a}$$

$$\text{with increments} \quad K_1 = A(Y_n) \quad \text{and} \quad K_2 = A\big(\exp\big(hA(Y_n)\big)Y_n\big) \tag{3.53b}$$

$$\text{and internal stages} \quad \bar{\Omega}_i = h\sum_{j=1}^{s} a_{ij}K_j, \quad \text{and} \quad \bar{Y}_i = \exp(\bar{\Omega}_i)Y_n \tag{3.53c}$$

p=3. For methods with convergence order $p = 3$, the differential equation

$$\dot{\Omega}(t) = f_1(\Omega, A) = A - \frac{1}{2}[\Omega, A] \tag{3.54}$$

has to be used. For example, the method of Heun with $s = 3$ stages and non-zero coefficients $b_1 = \frac{1}{4}, b_3 = \frac{3}{4}$, $a_{21} = \frac{1}{3}$, $a_{32} = \frac{2}{3}$ reads

$$\Omega_{n+1} = \frac{h}{4}(K_1 + 3K_3), \tag{3.55a}$$

with increments K_i, $i = 1, 2, 3$ defined by

$$K_i = f_1(\bar{\Omega}_i, A(\bar{Y}_i)) = A(\bar{Y}_i) - \frac{1}{2}[\bar{\Omega}_i, A(\bar{Y}_i)]. \tag{3.55b}$$

The internal stages are given by

$$\bar{\Omega}_1 = \Omega_n = 0, \qquad \bar{\Omega}_2 = \frac{h}{3}K_1, \qquad \bar{\Omega}_3 = \frac{2h}{3}K_2, \tag{3.55c}$$

$$\bar{Y}_1 = Y_n, \qquad \bar{Y}_2 = \exp(\bar{\Omega}_2)Y_n \qquad \bar{Y}_3 = \exp(\bar{\Omega}_3)Y_n. \tag{3.55d}$$

Finally, Ω_{n+1} is mapped via equation (3.47) to Y_{n+1}. It straightforwardly can be shown that Y_{n+1} takes the same convergence order than Ω_{n+1}.

The big advantage of the Munthe-Kaas Runge-Kutta method is that there are no additional order conditions comparing this method with general Runge-Kutta schemes. This means, any already known set of coefficients for Runge-Kutta schemes can be simply used. On the other hand, Munthe-Kaas truncates the infinite series given by Magnus in (3.46). This means, the function f_q has to be adapted to the convergence order and includes more and more Lie brackets with ascending order. In general, a model error is introduced at this place because not the exact derivative of the inverse of the exponential function is approximated by the Runge-Kutta method but an approximation of it as mentioned in [60]. As discussed in paragraph 3.1.2, there are Lie groups so as to the *expinv* equation can be expressed in a closed form. For our considerations, we regard the general case and do not take into account the closed forms.

3.3 Summary

This chapter is split into two parts and can be summarized as follows: First, the theory of differential equations needed for the development of its numerical solutions is provided. Starting from a differential equation (3.1), i.e.,

$$\dot{Y}(t) = A(Y(t)) \cdot Y(t)$$

on Lie groups with $Y(t)$ being in a Lie group G and $A(Y(t))$ in an associated Lie algebra $\mathfrak{g}$, the result has to be in a Lie group G. Numerical integration schemes on Lie groups ensure the affiliation to a Lie group via the usage of a local parameterization (3.5)

$$Y(t) = \Psi(\Omega(t)).$$

Here, the expression (3.11)

$$\dot{\Omega} = \left(d\Psi_{\Omega}^{-1}\right) A$$

has to be identified such that Ω_{n+1} can be computed by a numerical method $\Omega_{n+1} = \Phi(\Omega_n)$ and afterwards be mapped to the Lie group via $Y_{n+1} = \Psi(\Omega_{n+1})$.

For Lie groups, the local parameterization should be a mapping from the associated Lie algebra $\mathfrak{g}$ to the Lie group G:

$$\Psi : \mathfrak{g} \to G, \quad \Omega \mapsto Y(t) = \Psi(\Omega(t)) \tag{3.56}$$

such that (3.11) is a differential equation in the linear space of the Lie algebra $\mathfrak{g}$. Usually, the exponential function is taken as local parameterization

In the second part, three kinds of Runge-Kutta schemes for differential equations on Lie groups are introduced: the Lie Euler scheme, Crouch-Grossmann and Munthe-Kaas Runge-Kutta methods which are all based on the exponential function as local parameterization. The Lie-Euler is an extension of the abelian Euler scheme attained via a comparison of the Euler scheme with the first two summands in the series of the exponential function. It is the most simple Runge-Kutta scheme for Lie groups. Moreover, it coincides with Crouch-Grossmann and Munthe-Kaas schemes of convergence order one.

Crouch-Grossmann Runge-Kutta schemes are generalizations of the Lie-Euler scheme. The intermediate steps Y_i are computed as exponential functions of the increments K_i for $i = 1, \ldots, s$. This method suffers from the disadvantage that higher-order schemes have more and more order conditions compared to general Runge-Kutta schemes for the Abelian case.

Finally, Munthe-Kaas Runge-Kutta schemes are based on the theorem of Magnus which uses the exponential function as local parameterization. The advantage of Munthe-Kaas Runge-Kutta schemes is that any already known set of coefficients of a certain convergence order can be used. The disadvantage is that, in general, there occur more and more powers of the adjoint operator with ascending convergence order. This is due to the truncation of the inverse of the derivative of the exponential map called *dexpinv* equation. This disadvantage disappears if the exponential mapping is exchanged to a mapping without truncation. As alternative, the usage of the Cayley transform is possible (but just for quadratic Lie groups). This approach is described later (in section 4.2).

Part II

Developments in Lie Group Methods

Chapter 4

4

Runge-Kutta Methods for Lie Groups

Based on the well-known Runge-Kutta methods for Lie groups described in the previous chapter, we focus on the development of new Munthe-Kaas Runge-Kutta Methods here: a Munthe-Kaas Runge-Kutta scheme with step size control and a Munthe-Kaas Runge-Kutta scheme using the Cayley transform as alternative local parameterization.

We concentrate on the initial value problem (3.26) as before, namely

$$\dot{Y}(t) = A(t) \cdot Y(t) \quad \text{with initial values} \quad Y(t_0) = Y_n, \tag{4.1}$$

with Lie group element $Y(t)$ and Lie algebra element $A(t)$.

We open this chapter with a step size prediction for Munthe-Kaas Runge-Kutta schemes evolved in section 4.1. Starting with a usual step size prediction in the Abelian case $\mathbb{R}^n$ using embedded schemes in subsection 4.1.1, we develop embedded Munthe-Kaas Runge-Kutta schemes for the non-Abelian case in subsection 4.1.2. In a second section 4.2, we focus on another approach: the substitution of the exponential function in the local parameterization on the Lie group. Here, the Cayley transform is used as local parameterization in 4.2.1 and discussed in the context of the Lie-Euler method and the Munthe-Kaas Runge-Kutta method in paragraph 4.2.2.

4.1 Step Size Control

The Runge-Kutta methods for differential equations on Lie groups mentioned in section 3.2 are also suitable to be used with a step size prediction. One example is a step size prediction for Runge-Kutta methods of Crouch-Grossmann type established in [20] by Fritzsch and Ramos. At the same time, we designed a step size prediction for Munthe-Kaas Runge-Kutta schemes [63] based on a step size control in the Abelian case. This development is explained in detail in this paragraph. Then, the step size prediction is adapted to Lie group problems of the type $\dot{Y}(t) = A(Y(t))Y(t)$ and its solution via Munthe-Kaas Runge-Kutta schemes, also with focus on embedded schemes. In both cases, an example with Bogacki-Shampine coefficients [6] is outlined which is adapted to the Wilson flow (2.78).

4.1.1 Step Size Prediction for the Abelian Case

Let the differential equation

$$\dot{y}(t) = f(y(t)) \quad \text{with} \quad y(t), f(y(t)) \in \mathbb{R}^n \tag{4.2}$$

be given. Considering numerical integration schemes, there exists an optimal step size at each time point of the numerical integration. Here, optimal means that the step size is as large as possible to perform as few steps as possible and save computing time and at the same time small enough to meet a prescribed error tolerance. Thus, the errors of a method are controlled in a suitable manner. This is described by Hairer et al in section II.4 of [30] and recapitulated in this section.

Let a Runge-Kutta method for y_{n+1} with order p and a Runge-Kutta method $\hat{y}_{n+1}$ of order $\hat{p} = p+1$ be given. The difference $y_{n+1} - \hat{y}_{n+1}$ can be used as error estimation in the following sense as

$$\text{err} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\Big(\frac{||\hat{y}_{n+1,j} - y_{n+1,j}||}{ATOL + RTOL \cdot ||\hat{y}_{n+1,j}||}\Big)^2} \tag{4.3}$$

for $y_{n+1}, \hat{y}_{n+1} \in \mathbb{R}^n$. Here, $y_{n+1,j}$ and $\hat{y}_{n+1,j}$ describe the j-th component of the vectors y_{n+1} and $\hat{y}_{n+1}$. The absolute and relative error tolerances are mentioned as $ATOL$ and $RTOL$. Starting from an initial step size h, the optimal step size h_{opt} for a method with convergence order p is computed as

$$h_{\text{opt}} = h \cdot \sqrt[p+1]{\frac{1}{\text{err}}}. \tag{4.4}$$

In practice, h_{opt} is multiplied with a safety factor ρ which is a little bit smaller than 1 such that h_{opt} is given as

$$h_{\text{opt}} = h \cdot \sqrt[p+1]{\frac{1}{\text{err}}} \cdot \rho, \qquad \rho < 1. \tag{4.5}$$

This means, the new optimal step size is a bit smaller than permitted for safety reasons. A step size control works in the following way:

Algorithm 4.1 (Step Size Control, [30])**.** Given is an initial value problem $\dot{y} = f(y)$ with initial value $y(t_0) = y_n$. A step size control is performed as follows:

1. Start from an initial step size h at time t_0.
2. Compute two numerical solutions y_{n+1} and $\hat{y}_{n+1}$ with convergence order p and $\hat{p} = p + 1$.
3. Measure the error with an error measure like the error given in (4.3).
4. Compute the step size for the next step h_{opt} with (4.5).

   a) If err ≤ 1, then take y_{n+1} as new value at time $t_{i+1} = t_i + h$.
   b) If err > 1, the step size was too large: compute y_{n+1} at time $t_{i+1} = t_i + h_{\text{opt}}$.

5. Set $h = h_{\text{opt}}$ and proceed with step 2.

In step 4a, it is also possible to use the more accurate value $\hat{y}_{n+1}$.

Embedded Methods. The step size prediction even can be improved using the combination of two Runge-Kutta methods employing the same function evaluations. This kind of step size control is known as embedded method. Embedded methods have the advantage that the increments k_i just have to be computed once for both methods. Often, this idea is combined with the first same as last (FSAL) trick: the last increment of the method with more stages is set to the numerical result of the method with less stages. Note that there is no general rule for the size of the stage numbers s of the lower-order and $\hat{s}$ of the higher-order method. Depending on the coefficients, it may happen that s is smaller or larger than or even equal to $\hat{s}$. In this paragraph, we just assume that $\hat{p} = p + 1$ holds.

Definition 4.2 (Embedded Runge-Kutta Methods, [30])**.** Let the two Runge-Kutta methods for y_{n+1} and $\hat{y}_{n+1}$ with order p and $\hat{p} = p + 1$ be given using the same function values:

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i, \quad \hat{y}_{n+1} = y_n + h\sum_{i=1}^{\hat{s}} \hat{b}_i k_i, \quad k_i = f(y_n + h\sum_{j=1}^{\hat{s}} a_{ij} k_j) \tag{4.6}$$

Then, the difference $y_{n+1} - \hat{y}_{n+1}$ can be used as cheap way of an error estimation.

Next, there is an example of an embedded method using the Bogacki-Shampine coefficients suggested by Bogacki and Shampine in [6]. It has convergence order (2)3 such that $\hat{y}_{n+1}$ is the higher-order approximation with convergence order $\hat{p} = 3$ and y_{n+1} the lower-order approximation with $p = 2$. The characteristic of this scheme is that the higher-order approximation $\hat{y}_{n+1}$ determines the numerical result. The coefficients are mentioned in table 4.1.

Examples 4.3 (Embedded Runge-Kutta with Bogacki-Shampine Coefficients, [6])**.**

$$\hat{y}_{n+1} = y_n + h\sum_{i=1}^{\hat{s}=3} \hat{b}_i k_i = y_n + \frac{2}{9}hk_1 + \frac{1}{3}hk_2 + \frac{4}{9}hk_3, \tag{4.7a}$$

$$y_{n+1} = y_n + h\sum_{i=1}^{s=4} b_i k_i = y_n + \frac{7}{24}hk_1 + \frac{1}{4}hk_2 + \frac{1}{3}hk_3 + \frac{1}{8}hk_4 \tag{4.7b}$$

$$k_1 = f(y_n), \quad k_2 = f(y_n + \frac{h}{2}k_1), \quad k_3 = f(y_n + \frac{3}{4}hk_2), \quad k_4 = y_{n+1} \tag{4.7c}$$

Table 4.1: Bogacki-Shampine coefficients

0					
1/2	1/2				
3/4	0	3/4			
1	2/9	1/3	4/9		
	2/9	1/ 3	4/9	0	← $\hat{b}$
	7/24	1/4	1/3	1/8	← b

The example combines the definition of embedded Runge-Kutta methods 4.2 with the Bogacki-Shampine coefficients stated in table 4.1. Here, the last increment k_4 of y_{n+1} is set as $k_4 := \hat{y}_{n+1}$. Hence, $\hat{y}_{n+1}$ is computed with $\hat{s} = 3$ stages and y_{n+1} is calculated from previously computed increments k_1, k_2, k_3 and the numerical approximation of higher order $\hat{y}_{n+1}$. The main benefit is that an error control without additional function evaluations is possible. All embedded Runge-Kutta schemes for the Abelian case can be easily adapted to Runge-Kutta schemes for Lie groups as described in the next part.

4.1.2 Step Size Control for Munthe-Kaas Runge-Kutta Schemes

Step size predictions for Runge-Kutta methods of Munthe-Kaas type work similar as general step size predictions for Runge-Kutta methods. Starting from the initial value problem (4.1)

$$\dot{Y}(t) = A(t) \cdot Y(t) \quad \text{with initial values} \quad Y(t_0) = Y_n,$$

the numerical approximations Y_{n+1} and $\hat{Y}_{n+1}$ with convergence order p and $\hat{p} = p+1$ are needed for the error estimation based on the difference of Y_{n+1} and $\hat{Y}_{n+1}$ such that the optimal step size can be determined. Since Munthe-Kaas Runge-Kutta schemes are solved in the Lie algebra and then mapped via the exponential function via

$$Y_{n+1} = \exp(\Omega_{n+1})Y_n \quad \text{and} \quad \hat{Y}_{n+1} = \exp(\widehat{\Omega}_{n+1})Y_n, \tag{4.8}$$

in the Lie group, the error of the method already occurs in the numerical approximations Ω_{n+1} and $\widehat{\Omega}_{n+1}$, i.e., the error estimation is based on the Lie algebra elements Ω_{n+1} and $\widehat{\Omega}_{n+1}$. This is advantageous because (due to the linear structure of the Lie algebra) a comparison of its elements is simpler than for Lie group elements.

Nevertheless, the formula for the error measure (4.3) has to be adapted to matrices instead of vectors:

$$\text{err} = \frac{||\widehat{\Omega}_{n+1} - \Omega_{n+1}||}{ATOL + RTOL \cdot ||\widehat{\Omega}_{n+1}||} \,. \tag{4.9}$$

Since the numerical approximations Ω_{n+1} and $\widehat{\Omega}_{n+1}$ are elements of the matrix Lie algebra $\mathfrak{g}$, the norms $||\widehat{\Omega}_{n+1} - \Omega_{n+1}||$ and $||\widehat{\Omega}_{n+1}||$ have to be chosen as matrix norms like the Frobenius norm, the spectral norm or the row sum norm. ATOL and RTOL are prescribed error tolerances as usual.

The numerical approximations Ω_{n+1} and $\widehat{\Omega}_{n+1}$ of convergence order p and $\hat{p} = p+1$ are computed via Munthe-Kaas Runge-Kutta schemes according to algorithm 3.22. So, the differential equations

$$\dot{\Omega} = \sum_{k=0}^{p-2} \frac{B_k}{k!} \operatorname{ad}_{\Omega}^{k}(A) =: f_{p-2}(\Omega, A) \tag{4.10a}$$

$$\text{and} \quad \dot{\widehat{\Omega}} = \sum_{k=0}^{\hat{p}-2} \frac{B_k}{k!} = \operatorname{ad}_{\Omega}^{k}(A) = \sum_{k=0}^{p-1} \frac{B_k}{k!} \operatorname{ad}_{\Omega}^{k}(A) =: f_{p-1}(\Omega, A) \tag{4.10b}$$

are considered and it holds

$$\Omega_{n+1} = \Omega_n + h \sum_{i=1}^{s} b_i K_i \quad \text{with} \quad K_i = f_{p-2}\Big(\bar{\Omega}_i, A(\bar{Y}_i)\Big) \tag{4.10c}$$

$$\text{and} \quad \widehat{\Omega}_{n+1} = \Omega_n + h \sum_{i=1}^{\hat{s}} \hat{b}_i \widehat{K}_i \quad \text{with} \quad \widehat{K}_i = f_{p-1}\Big(\hat{\bar{\Omega}}_i, A(\hat{\bar{Y}}_i)\Big) \tag{4.10d}$$

with $\Omega_n = 0$ and internal stages

$$\bar{\Omega}_i = \Omega_n + h \sum_{j=1}^{s} a_{ij} K_j, \qquad \bar{Y}_i = \exp(\bar{\Omega}_i) Y_n \quad \text{for} \quad i = 1, \ldots, s, \tag{4.10e}$$

$$\hat{\bar{\Omega}}_i = \Omega_n + h \sum_{j=1}^{\hat{s}} \hat{a}_{ij} \widehat{K}_j, \qquad \hat{\bar{Y}}_i = \exp(\hat{\bar{\Omega}}_i) Y_n \quad \text{for} \quad i = 1, \ldots, \hat{s}. \tag{4.10f}$$

Afterwards, the results can be mapped from the Lie algebra $\mathfrak{g}$ to the Lie group G via equation (4.8).

So, the step size control for Lie group problems using the Munthe-Kaas Runge-Kutta scheme is similar to the one for differential equations in the Abelian case. Just the error measure has to be adapted to matrix Lie algebras and an additional mapping to the Lie group is necessary. The whole procedure is explained in the next algorithm.

Algorithm 4.4 (Step Size Control using Munthe-Kaas Runge-Kutta methods)**.** Given is an initial value problem (4.1)

$$\dot{Y}(t) = A(t) \cdot Y(t) \quad \text{with initial values} \quad Y(t_0) = Y_n.$$

A step size control is performed as follows:

1. Start from an initial step size h at time t_0.

2. Identify Y and Ω due to

$$Y(t) = \exp(\Omega(t)) \quad \text{and} \quad \dot{\Omega}(t) = \operatorname{d}\exp_{\Omega}^{-1}(A(t)) \tag{4.11}$$

3. Compute two numerical solutions Ω_{n+1} and $\widehat{\Omega}_{n+1}$ with convergence order p and $\hat{p} = p + 1$ according to equations (4.10).
4. Measure the error with an error measure like (4.9).
5. Compute the step size for the next step h_{opt} with (4.5).
    a) If err ≤ 1,
        - compute $Y_{n+1} = \exp(\Omega_{n+1})Y_n$
        - and take Y_{n+1} as new value at time $t_{i+1} = t_i + h$

    b) If err > 1, the step size was too large.
6. Set $h = h_{\text{opt}}$ and proceed with step 3.

Embedded Munthe-Kaas Runge Kutta Methods. As mentioned before, embedded Runge-Kutta schemes are computed using the same set of coefficients for the different numerical approximations of convergence order p and $p + 1$. This means, the increments k_i, $i = 1, \ldots, s$ are the same for both approximations and just computed once. For Munthe-Kaas Runge-Kutta schemes, there are different increments K_i and $\widehat{K}_i$, $i = 1, \ldots, s$ described by equations (4.10c) and (4.10d):

$$\begin{aligned} & K_i = f_{p-2}\big(\bar{\Omega}_i, A(\bar{Y}_i)\big) \\ \text{and} \quad & \widehat{K}_i = f_{p-1}\big(\bar{\Omega}_i, A(\bar{Y}_i)\big) = f_{p-2}\big(\bar{\Omega}_i, A(\bar{Y}_i)\big) + \frac{B_{p-1}}{(p-1)!} \operatorname{ad}_{\Omega}^{p-1}(A) \end{aligned}$$

The difference depends on the truncation index of the Munthe-Kaas scheme according to the desired convergence order. The truncation index induces a model error and has to be at least $p-2$ for a desired convergence order p and $\hat{p}-2$ for the desired convergence order $\hat{p}$. Thus, it is also possible to use the more accurate model with truncation index $\hat{p} - 2 = p - 1$ for both schemes, i.e. set

$$K_i = f_{p-1}\big(\bar{\Omega}_i, A(\bar{Y}_i)\big) = \sum_{k \geq 0}^{p-1} \frac{B_k}{k!} \operatorname{ad}_{\Omega}^{k}(A). \tag{4.12}$$

Based on these considerations, the embedded Munthe-Kaas Runge-Kutta method can be defined as follows:

Definition 4.5 (Embedded Munthe-Kaas Runge Kutta Methods)**.** The embedded

Munthe-Kaas Runge Kutta scheme reads

$$\Omega_{n+1} = h\sum_{i=1}^{s} b_i K_i \quad \text{and} \quad \widehat{\Omega}_{n+1} = h\sum_{i=1}^{\hat{s}} \hat{b}_i K_i, \tag{4.13a}$$

and uses the increments

$$K_i = f_{p-1}\Big(\bar{\Omega}_i, A(\bar{Y}_i)\Big) \quad \text{with} \quad \bar{\Omega}_i = h\sum_{j=1}^{s} a_{ij} K_j \quad \text{and} \quad \bar{Y}_i = \exp(\bar{\Omega}_i) Y_n \tag{4.13b}$$

for all stages $i = 1, \dots, \max(s, \hat{s})$.

It is advantageous to use this embedded Munthe-Kaas Runge-Kutta scheme at step 3 of the step size prediction explained in algorithm 4.4. Next, we state an example of an embedded Munthe-Kaas Runge-Kutta scheme with Bogacki-Shampine coefficients (see table 4.1). Here, Ω_{n+1} is computed with $s = 4$ stages and convergence order $p = 2$ and $\widehat{\Omega}_{n+1}$ with $\hat{s} = 3$ stages and convergence order $\hat{p} = 3 = p + 1$.

Examples 4.6 (Embedded MK-RK Method with Bogacki-Shampine Coefficients)**.** Applying the Bogacki-Shampine coefficients of table 4.1 on the embedded Munthe-Kaas Runge-Kutta scheme (4.13) yields

$$\widehat{\Omega}_{n+1} = h\Big(\frac{1}{2}K_1 + \frac{1}{3}K_2 + \frac{4}{9}K_3\Big) \tag{4.14a}$$

$$\text{and} \quad \Omega_{n+1} = h\Big(\frac{7}{24}K_1 + \frac{1}{4}K_2 + \frac{1}{3}K_3 + \frac{1}{8}K_4\Big) \tag{4.14b}$$

According to theorem 3.21 and formula (3.46), the increments K_i, $i = 1, \dots, 4$, are computed as

$$K_i = f_1\Big(\bar{\Omega}_i, A(\bar{Y}_i)\Big) = A(\bar{Y}_i) - \frac{1}{2}\Big[\bar{\Omega}_i, A(\bar{Y}_i)\Big] \tag{4.14c}$$

for a numerical approximation of order $\hat{p} = 3$. Here, the internal stages

$$\bar{\Omega}_1 = \Omega_n = 0, \qquad \bar{\Omega}_2 = \frac{h}{2}K_1, \qquad \bar{\Omega}_3 = \frac{3}{4}hK_2, \qquad \bar{\Omega}_4 = \Omega_{n+1} \tag{4.14d}$$

are used and $\bar{Y}_i$ is set to $\bar{Y}_i = \exp(\bar{\Omega}_i)$ for $i = 1, \dots, 4$.

Adaption to Lattice QCD. Our aim is the development of numerical schemes for differential equations on Lie groups for Lattice QCD. Indeed, the embedded Munthe-Kaas Runge-Kutta scheme can be used for the computation of the Wilson flow (2.80), i.e.,

$$\dot{V}_j(t) = Z_j(t) \cdot V_j(t), \tag{4.15a}$$

$$Z_j = F([V_j]), \tag{4.15b}$$

$j = 0, \dots, n_l - 1$, in Lattice QCD. The Wilson flow is a coupled differential equation but we consider just the differential equation (4.15a) and suppose that the algebraic equation (4.15b) is already computed. So, each of the n_l equations of formula (4.15a) has the shape of the single scalar equation (4.1) – $Z_j(t)$ is situated in the Lie algebra $\mathfrak{su}(N, \mathbb{C})$ and $V_j(t)$ in the Lie group $\mathrm{SU}(N, \mathbb{C})$.

Compared to algorithm 4.4, the only difference is that the differential equation has to be solved for n_l instead of just one lattice points. So, the numerical approximations

$$\{\Omega_{n+1,0}, \dots, \Omega_{n+1,n_l-1}\} \quad \text{and} \quad \{\widehat{\Omega}_{n+1,0}. \dots, \widehat{\Omega}_{n+1,n_l-1}\}$$

have to be computed. According to this, the error measure must be computed from all n_l single errors, for example with the Euclidean norm

$$\mathrm{err} = \sqrt{\frac{1}{n_l} \sum_{j=0}^{n_l-1} \Big(\frac{||\widehat{\Omega}_{n+1,j} - \Omega_{n+1,j}||}{ATOL + RTOL \cdot ||\widehat{\Omega}_{n+1,j}||} \Big)^2} \,. \tag{4.16}$$

In chapter 8.2, an embedded Munthe-Kaas Runge-Kutta scheme for the Wilson flow is described and simulated for a single configuration of a lattice gauge field.

4.2 The Cayley Transform

Almost all methods for solving the differential equations $\dot{Y} = AY$ on Lie groups are based on the usage of a local parameterization (3.5), i.e.,

$$\Psi : \mathfrak{g} \to G, \quad Y(t) = \Psi(\Omega(t)). \tag{4.17}$$

of the Lie group. Here, Ψ must be a mapping from the Lie algebra to its associated Lie group. Furthermore, there is a new unknown $\Omega(t)$ which is the solution of a differential equation in the Lie algebra implicitly given by the local parameterization:

$$\mathfrak{g} \to \mathfrak{g} : \quad \dot{\Omega}(t) = \mathrm{d}\,\Psi^{-1}_{\Omega(t)}(A(t)) \tag{4.18}$$

Usually, the exponential function is taken as local parameterization. The theorem of Magnus (theorem 3.19) states that Lie group differential equation of the shape $Y(t) = A(t)Y(t)$ can be solved by using the mapping $Y(t) = \exp(\Omega(t))Y_n$ with new unknown Ω being the result of $\dot{\Omega} = \mathrm{d}\exp_{\Omega}^{-1}$. Unfortunately, $\mathrm{d}\exp_{\Omega}^{-1}$ is an infinite series based on powers of the adjoint operator which has some disadvantages. First, the infinite series has to be truncated somewhere such that a model error is introduced. Furthermore, it is difficult to develop higher-order methods since the powers of the adjoint operator have to be considered in the computation of the order conditions leading to unpleasant expressions.

4.2.1 The Cayley Transform as Local Parameterization

The local parameterization can be considered in a more abstract way as already mentioned in paragraph 3.1.1:

Algorithm 4.7 (Numerical solution of a Lie group IVP using a local parameterization)**.** Given is a differential equation on a Lie group (4.1)

$$\dot{Y}(t) = A(t) \cdot Y(t) \quad \text{with initial values} \quad Y(t_0) = Y_n,$$

i.e. $Y(t)$ and Y_n are elements of the Lie group G and $A(t)$ an element of the associated Lie algebra $\mathfrak{g}$. Then, a numerical solution can be obtained performing the following steps

1. Identify a mapping

   $$\Psi : \mathfrak{g} \to G, \quad Y(t) = \Psi\big(\Omega(t)\big) \cdot Y(t_0) \tag{4.19}$$

   from the Lie algebra to the corresponding Lie group with unknown $\Omega(t) \in \mathfrak{g}$.

2. The function $\Omega(t)$ is the result of the initial value problem

   $$\dot{\Omega} = \mathrm{d}\,\Psi_{\Omega}^{-1}\big(A(t)\big). \tag{4.20}$$

   with initial value $\Omega_n = 0 \in \mathfrak{g}$.

3. Compute the numerical approximation $\Omega_{n+1} \approx \Omega(t_0 + h)$ with any numerical method in the Lie algebra $\mathfrak{g}$.

4. Finally, the approximation Ω_{n+1} is mapped to the Lie group:

   $$Y_{n+1} = \Psi\big(\Omega_{n+1}\big) \cdot Y_n. \tag{4.21}$$

Most frequently, the exponential function is chosen as mapping from the Lie algebra to the Lie group. Since the only requirements on the mapping is that it maps from the Lie algebra to the Lie group and that the derivative of its inverse function exists, also other mappings can be chosen instead of the exponential function.

One example for a different local parameterization Ψ is the Cayley transform which just exists for quadratic Lie groups

$$G = \{Y | Y^H P Y = P\} \tag{4.22}$$

with given constant matrix P. The corresponding Lie algebra reads

$$\mathfrak{g} = \{\Omega | P\Omega + \Omega^H P = 0\}. \tag{4.23}$$

The Lie group $SU(N)$ used in Lattice QCD is an example of a quadratic Lie group with identity matrix P. The Cayley transform is mentioned in the next lemma:

Lemma 4.8 (Cayley transform, [29])**.** Consider a quadratic matrix Lie group G and its Lie algebra $\mathfrak{g}$. The Cayley transform is defined as

$$\mathfrak{g} \mapsto G : \quad \Omega \mapsto \mathrm{cay}(\Omega) = \big(I - \Omega(t)\big)^{-1}\big(I + \Omega(t)\big). \tag{4.24}$$

i.e., for $A \in \mathfrak{g}$ we have $\mathrm{cay}(A) \in G$. Moreover, cay is a local diffeomorphism in a neighborhood of $A = 0$.

It is obvious that the Cayley transform is just applicable if the matrix $I - \Omega$ has full rank such that is invertible. Indeed, this is the case for special unitary Lie groups $SU(N, \mathbb{C})$ because they are quadratic Lie groups with constant matrix $P = I$. The appropriate Lie algebra $\mathfrak{su}(N, \mathbb{C})$ consists of traceless and anti-hermitian matrices such that its eigenvalues are pure imaginary and the matrix $(I - \Omega)$ is invertible.

For the solution of $\dot{\Omega}$ (needed in the second step of algorithm 4.7), the inverse of the derivative of the Cayley mapping is given in the following definition.

Definition 4.9 (Inverse of the derivative of the Cayley mapping, [29])**.** The differential equation $\dot{\Omega}(t) = \mathrm{d}\,\mathrm{cay}_{\Omega}^{-1}$ can be computed via

$$\mathfrak{g} \mapsto \mathfrak{g} : \qquad \mathrm{d}\,\mathrm{cay}_{\Omega}^{-1}\big(A(t)\big) = \frac{1}{2}\big(I - \Omega(t)\big)A(t)\big(I + \Omega(t)\big). \tag{4.25}$$

It is a differential equation in the Lie algebra $\mathfrak{g}$.

The Cayley transform has two big advantages compared to the exponential function: it has a closed form, i.e. it introduces no model error. Furthermore, the computational effort is lower than for the *dexpinv* equation. This argument holds especially for higher order methods since no powers of the adjoint operator are needed. Furthermore, there are analytical formulae for the inverse of the matrices for $N = 2$ and $N = 3$.

All already investigated methods in this thesis can be adapted to a local parameterization using the Cayley transform provided that the underlying Lie group is a quadratic one.

4.2.2 Numerical Integration with the Cayley Mapping

The different local parameterizations influence the shape of numerical integration schemes on Lie groups for the initial value problem (4.1). This is shown for the example of the Lie-Euler method. Following algorithm 4.7, $Y(t)$ is described by the parameterization $Y(t) = \Psi(\Omega(t))Y_n$ where $\Omega(t)$ is given by the differential equation $\dot{\Omega}(t) = \mathrm{d}\,\Psi_{\Omega}^{-1}(A(t))$ in the Lie algebra.

For the exponential function, it holds $\dot{\Omega}(t) = \mathrm{d}\exp_{\Omega}^{-1}(A(t))$ which is an infinite series as mentioned in equation (3.13). For convergence order $p = 1$, it is sufficient to use

$$\dot{\Omega}(t) = \mathrm{d}\exp_{\Omega}^{-1}(A(t)) \approx A \tag{4.26}$$

such that Ω_{n+1} is computed as $\Omega_{n+1} = hA(Y_n)$.

Thus, the standard Lie-Euler method using the exponential function is given by

$$Y_{n+1} = \exp\bigl(hA(Y_n)\bigr)Y_n \tag{4.27}$$

as already stated in definition 3.17. The Lie-Euler method with Cayley transform is formulated as

Definition 4.10 (Lie Euler method with Cayley transform)**.** Let the initial value problem (4.1)

$$\dot{Y}(t) = A(t) \cdot Y(t) \quad \text{with initial values} \quad Y(t_0) = Y_n$$

with $Y, Y_n \in G$, $A(Y) \in \mathfrak{g}$ be given. Then, the method

$$Y_{n+1} = \mathrm{cay}\bigl(\tfrac{h}{2}A(Y_n)\bigr)Y_n = \bigl(I - \tfrac{h}{2}A(Y_n)\bigr)^{-1}\bigl(I + \tfrac{h}{2}A(Y_n)\bigr)\,Y_n \tag{4.28}$$

is called Cayley-Lie-Euler method.

Comparing equations (4.27) and (4.28), the exponential function is evaluated at points $hA(Y_n)$ and the Cayley transform at $\frac{h}{2}A(Y_n)$. At a first glance, this is a bit confusing but due to the different shapes of $\dot{\Omega}$. For the exponential function, $\dot{\Omega}$ is given as $A(Y_n)$. For the Cayley transform, $\dot{\Omega}(t)$ is computed via equation (4.25). With initial value $\Omega_n = 0$, Ω_{n+1} is computed as

$$\Omega_{n+1} = \Omega_n + h\frac{1}{2}(I - \Omega_n)A_n(I + \Omega_n) = hA/2. \tag{4.29}$$

The Lie-Euler method is a special case of Munthe-Kaas Runge-Kutta methods of convergence order $p = 1$ for the different local parameterizations exp and cay. In general, Munthe-Kaas Runge-Kutta schemes with Cayley transform used as local parameterization are formulated as follows.

Algorithm 4.11 (Munthe-Kaas Runge-Kutta Method using the Cayley Mapping)**.** Given is a differential equation

$$\dot{Y} = A(t) \cdot Y(t), \quad Y(t) \in G, A(t) \in \mathfrak{g}\ . \tag{4.30}$$

This differential equation is solved numerically following these steps:

1. Identify a mapping

$$\begin{aligned} \Psi :\ \mathfrak{g} \to G, \qquad Y(t) &= \mathrm{cay}(\Omega(t)) \cdot Y(t_0) \\ &= \bigl(I - \Omega(t)\bigr)^{-1}\bigl(I + \Omega(t)\bigr) \cdot Y(t_0) \end{aligned} \tag{4.31}$$

   from the Lie algebra to the corresponding Lie group with unknown $\Omega(t) \in \mathfrak{g}$ and initial value $\Omega(t_0) = \Omega_n = 0$.

2. $\Omega(t)$ is defined by the derivative of the inverse of the mapping Ψ:

$$\dot{\Omega} = \mathrm{d}\,\mathrm{cay}_{\Omega}^{-1}\big(A(t)\big) = \frac{1}{2}\big(I - \Omega(t)\big)A(t)\big(I + \Omega(t)\big)\,. \tag{4.32}$$

3. Solve $\dot{\Omega} = \mathrm{d}\,\mathrm{cay}_{\Omega}^{-1}\big(A(t)\big)$ numerically, for example, with a Runge-Kutta method.

4. Map the solution Ω_{n+1} to the Lie group:

$$Y_{n+1} = \big(I - \Omega(t)\big)^{-1}\big(I + \Omega(t)\big) \cdot Y_0\,. \tag{4.33}$$

Since there is no model error introduced in this case, the convergence order of the numerical method is met in any case.

Crouch-Grossmann schemes can also be used either with the exponential function or the Cayley transform. This is described, for example, for partitioned Runge-Kutta methods in a Lie group setting by Engo [18] but not the subject of this thesis.

The Cayley transform as alternative local parameterization is used for the Leapfrog scheme in paragraph 5.2.3. At the end, the Cayley transform is utilized in a simulation of the Hamiltonian equations of motion by means of the Leapfrog scheme in part 7.2.2.

4.3 Summary

In this chapter, Munthe-Kaas Runge-Kutta methods for differential equations on Lie groups have been developed further.

Section 4.1 combines a step size prediction with Munthe-Kaas Runge-Kutta schemes. Thus, a step size control using Munthe-Kaas Runge-Kutta schemes is achieved with focus on cheap embedded schemes via algorithm 4.4 using equations (4.13). This method can be applied, for example, on the Wilson flow. The step size control using embedded Munthe-Kaas Runge-Kutta schemes with Bogacki-Shampine coefficients for the Wilson flow and its numerical results can be found in section 8.2.

In section 4.2, the focus is put on the local parameterization of the Lie group elements used for the numerical integration of differential equations on Lie groups. It is suggested to use the Cayley transform instead of the exponential function. The advantages of the Cayley transform are on the one side low computational cost for matrix Lie groups of small size (N=2 or N=3) and on the other side the fact that the inverse of its derivative can be computed as a closed form. Compared to the exponential function, there is no additional model error introduced. Thus, the differential equation in the Lia algebra, i.e., $\dot{\Omega} = \mathrm{d}\,\Psi^{-1}(A)$, remains the same for all desired convergence orders. The Cayley transform has the small disadvantage that it has to be ensured that the Lie group is a quadratic Lie group and that the inverse of $I - \Omega(t)$ indeed exists, i.e., that the matrix $I - \Omega(t)$ has full rank. These points have to be investigated before the utilization of the Cayley transform. For applications in Lattice QCD, the Cayley transform can be used without restriction because special unitary Lie groups are quadratic and all eigenvalues of $I - \Omega(t)$ are non-zero. In this topic, the exponential function could be replaced by the Cayley transform in any numerical integration method for differential equations on Lie groups.

We test the Cayley transform as local parameterization in the context of Lattice QCD: the Hamiltonian equations of motion occurring in the Hybrid Monte Carlo method are simulated with the Leapfrog method using the Cayley transform. This geometric method is described in chapter 5.2 in subsection 5.2.3 and its application on a lattice gauge field is shown in chapter 7 in paragraph 7.2.2. In a further step, both schemes could be combined and also applied on the Wilson flow.

Chapter 5

5 Geometric Numerical Integration

For Hamiltonian systems, a system of coupled differential equations

$$\dot{Y}(t) = A(Y(t)) \cdot Y(t), \qquad \dot{A}(t) = F(Y(t)) \tag{5.1}$$

on a Lie group and its corresponding Lie algebra may arise. Here, Y is in the Lie group, A in the Lie algebra and F is a Lie-algebra valued function which maps an element from the Lie group to the Lie algebra. The first equation is a differential equation on a Lie group as explained in chapter 3. It has the same shape and properties as equation (3.1) and arises from the derivative of the Hamiltonian $\mathcal{H}(Y, A)$ with respect to the Lie algebra element A. The second equation $\dot{A}(t) = F(Y(t))$ is a differential equation in the linear space of the Lie algebra such that it can be solved with any general numerical integration method. It is closely related to the derivative of the Hamiltonian $\mathcal{H}(Y, A)$ with respect to the Lie group element Y. Compared to the Hamiltonian equations of motion for Lattice QCD computations depending on $F([Y]_s)$, i.e., the function F depends on several Lie group elements, we neglect this fact here. For the method's development in this chapter, we assume that we have two coupled scalar differential equations.

Hamiltonian systems have to be solved by geometric numerical integration schemes for some reasons. First, the flow of the system preserves the geometric properties such that the numerical integration scheme should do it as well. Moreover, geometric integration is quite important for the development of schemes that occur in the context of the Molecular Dynamics step of the HMC. Here, the numerical integration scheme must be time-reversible and volume-preserving to compute the expectation values in a correct way. So, the concept of geometric integration is introduced in section 5.1 taken from [29]. It can also be found, for example, in the textbook of Highham and Griffiths [24] or the recent overview article of Bou-Rabee and Sanz-Serna [8].

The methods described in the previous chapter 4 can be used for the numerical time integration of differential equations on Lie groups. In general, these methods are not time-reversible and not volume-preserving such that numerical integration methods with these properties have to be investigated in more detail. In this chapter, some new geometric integration methods are developed with focus on its application in Lattice QCD: the Cayley-Leapfrog method, symmetric partitioned Runge-Kutta schemes and symmetric and symplectic projection schemes. All schemes are finally applied on the Hamiltonian equations of motion for a lattice gauge field such that the numerical results are shown and discussed in chapter 7.

This chapter is organized as follows: it starts with a short introduction on geometric integration in section 5.1 where the terms symmetry, time-reversibility, volume-preservation and symplecticity are explained.
Then, a section 5.2 about the Leapfrog scheme follows. It opens with the common Leapfrog or Störmer-Verlet scheme for the Abelian case in paragraph 5.2.1 and turns to the Leapfrog scheme for Lie group / Lie algebra problems in paragraph 5.2.2. Finally, the Cayley transform (described in section 4.2) is put in the context of the Leapfrog method in paragraph 5.2.3.

In a next subsection 5.3, symmetric partitioned Runge-Kutta methods are explained. First, partitioned Runge-Kutta schemes for the Abelian case are mentioned and then adapted to partitioned Munthe-Kaas Runge-Kutta methods. Afterwards, the symmetry conditions for partitioned Munthe-Kaas Runge-Kutta methods are investigated such that the schemes finally are adapted to be symmetric using three different ways. Based on one of the symmetric schemes, order conditions for convergence order 3 are developed and one example for a set of coefficients is derived.

The last section 5.4 deals with symmetric and symplectic projection schemes. Based on symmetric projection methods symmetric and volume-preserving projection methods for Lie group / Lie algebra problems are evolved. Here, it is shown that the dependence on the projection parameter is essential for the method.

5.1 Geometric Integration

In many applications, the Hamiltonian system

$$\dot{y} = J^{-1}\nabla\mathcal{H}(y) \quad \text{with} \quad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}, \tag{5.2}$$

$y \in \mathbb{R}^{2n}$, $\mathcal{H} : \mathbb{R}^{2n} \to \mathbb{R}$ and $J \in \mathbb{R}^{2n\times 2n}$ with blocks of identity matrices I of size $n\times n$ has to be solved numerically. Here, some geometric numeric integration schemes have to be used to preserve the geometric properties of the system. The term geometric integration has been introduced by Sanz-Serna [56] and comprises the terms symmetry, symplecticity, time-reversibility and volume-preservation (sometimes also called area-preservation) depending on the context. Most of the time, the terms symmetry and symplecticity are used in mathematical literature and time-reversibility and volume-preservation in applications meaning similar but at the end different things. The solution of the system (5.2) can be described by its flow $\varphi_t(y_n)$ mentioned in the next definition.

Definition 5.1 (Flow of a system, [29])**.** The flow $\varphi_t(y_n)$ describes the solution of the differential equation

$$\dot{y} = f(y), \quad y(t_0) = y_n \quad \text{with} \quad y, y_n \in \mathbb{R}^n, \quad f : \mathbb{R}^n \to \mathbb{R}$$

described in (3.17) with respect to its initial values as

$$\varphi_t(y_n) := y(t, t_0, f). \tag{5.3}$$

As mentioned in the introduction of this chapter, the flow φ_t preserves the geometric properties symplecticity, symmetry and time-reversibility for Hamiltonian systems (5.2) which are introduced in definitions 5.2 to 5.6.

Definition 5.2 (Symplecticity of a flow, [29]). A flow φ_t is symplectic if it holds

$$\left(\frac{\partial \varphi_t}{\partial y_n}\right)^T J \left(\frac{\partial \varphi_t}{\partial y_n}\right) = J \quad \text{with} \quad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}. \tag{5.4}$$

Symplecticity implies that the volume of the phase space stays constant which leads to the definition for volume-preservation.

Definition 5.3 (Volume-Preservation of a flow). The flow φ_t preserves the volume of the phase space if

$$\left|\det\left(\frac{\partial \varphi_t}{\partial y_n}\right)\right| = 1. \tag{5.5}$$

Volume-preservation and symplecticity are closely related to each other. Equation (5.4) is equivalent to the fact that the determinant of the Jacobian $\frac{\partial \varphi_t}{\partial y_n}$ takes the value one, i.e.

$$\det\left(\frac{\partial \varphi_t}{\partial y_n}\right) = 1. \tag{5.6}$$

This leads directly to the volume-preservation (5.5) whereas volume-preservation does not imply symplecticity because the determinant of the Jacobian may take the value -1.

The terms symmetry and time-reversibility are also closely linked together via the term ρ-reversibility.

Definition 5.4 (ρ-reversible differential equation, [29]). The exact flow φ_t of a ρ-reversible differential equation satisfies

$$\rho \circ \varphi_t = \varphi_{-t} \circ \rho = \varphi_{-t}^{-1} \circ \rho \tag{5.7}$$

Definition 5.5 (Symmetry of a flow, [29]). A flow φ_t is called symmetric if it satisfies

$$\varphi_t \circ \varphi_{-t} = id \quad \text{or equivalently} \quad \varphi_t = \varphi_{-t}^{-1}. \tag{5.8}$$

Here φ_{-t} means that the sign of the time of the flow is reversed in order that it is computed backward in time. The expression φ^{-1} implies that the flow itself is reversed. If both expressions coincide, i.e. $\varphi_{-t} = \varphi^{-1}$, the flow is symmetric. If the flow is in addition ρ-reversible

Definition 5.6 (Time-reversibility of a flow, [29])**.** A flow φ_t is time-reversible if it holds

$$\rho \circ \varphi_t \circ \rho \circ \varphi_t = id \quad \text{with} \quad \rho = \begin{pmatrix} -I & 0 \\ 0 & I \end{pmatrix}, \tag{5.9}$$

provided that φ_t is symmetric, i.e. $\varphi_t = \varphi_{-t}^{-1}$.

The flow of a system (3.17) should be approximated by a numerical integration method

$$\Phi_h : \mathbb{R}^n \to \mathbb{R}^n : \quad y_n \mapsto y_{n+1} = \Phi_h(y_n, h, f) \tag{5.10}$$

which preserves the geometric properties symplecticity, symmetry and time-reversibility. The definition of these terms for numerical methods Φ_h are quite similar to the definitions of the terms for the flow:

Definition 5.7 (Symplecticity, [29])**.** The numerical one-step scheme Φ_h is symplectic if it holds

$$\left(\frac{\partial \Phi_h}{\partial y_n}\right)^T J \left(\frac{\partial \Phi_h}{\partial y_n}\right) = J \quad \text{with} \quad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \tag{5.11}$$

Definition 5.8 (Volume-Preservation, [29])**.** The numerical one-step scheme Φ_h is volume-preserving if

$$\left| \det\left(\frac{\partial \Phi_h}{\partial y_n}\right) \right| = 1.$$

Volume-preservation is a direct consequence of symplecticity following from definition 5.7.

Definition 5.9 (Symmetry, [29])**.** A numerical one-step method Φ_h is called symmetric if it satisfies

$$\Phi_h \circ \Phi_{-h} = id \quad \text{or equivalently} \quad \Phi_h = \Phi_{-h}^{-1}. \tag{5.12}$$

Definition 5.10 (Adjoint, [29])**.** The function

$$\Phi_{-h}^{-1} =: \Phi_h^* \tag{5.13}$$

is called adjoint of the method Φ_h. The adjoint can be computed using the method Φ_h itself by exchanging the positive step size h against the negative step size $-h$ and the initial values y_n versus the results $y_{n+1} = \Phi(y_n)$.

Symmetry means that the numerical scheme Φ_h coincides with its adjoint Φ_h^*. It can be checked by setting $y_{n+1} = \Phi_h(y_n)$, changing $h \leftrightarrow -h$ and $y_{n+1} \leftrightarrow y_n$ and solving the equation for y_{n+1}. If it holds $y_{n+1} = \Phi_h(y_n)$ again, the scheme is symmetric.

Definition 5.11 (Time-reversibility, [29])**.** A numerical one-step method Φ_h is time-reversible if it holds

$$\rho \circ \Phi_h \circ \rho \circ \Phi_h = id \quad \text{with} \quad \rho = \begin{pmatrix} -I & 0 \\ 0 & I \end{pmatrix}, \tag{5.14}$$

provided that Φ_h is symmetric, i.e. $\Phi_h = \Phi_{-h}^{-1}$.

The condition for time-reversibility is equivalent to the condition

$$\rho \circ \Phi_h = (\rho \circ \Phi_h)^{-1} = \Phi_h^{-1} \circ \rho = \Phi_{-h} \circ \rho \tag{5.15}$$

These terms will be used for the development of geometric integration methods on Lie groups in the next paragraphs of this chapter and also for the simulations in chapter 7.

5.2 Störmer-Verlet or Leapfrog Method

The most popular geometric numerical integration method is the Leapfrog or Störmer-Verlet method. For more information, Hairer et al provide an overview which can be found in [27]. The Leapfrog scheme for the Abelian case and the formulation of the Hamiltonian system is taken from [29].

5.2.1 The Abelian Case

The Hamiltonian system (5.2) can be stated as

$$\begin{pmatrix} \dot{p} \\ \dot{q} \end{pmatrix} = J^{-1} \begin{pmatrix} \mathcal{H}_p \\ \mathcal{H}_q \end{pmatrix} \quad \text{with} \quad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \quad \text{and} \quad p, q, \mathcal{H}_p, \mathcal{H}_q \in \mathbb{R}^n. \tag{5.16}$$

Here, y is replaced by $(p, q)^\top$. This formulation is equivalent to the formulation of the Hamiltonian equations of motion which read

$$\dot{p} = -\mathcal{H}_q(p, q) \quad \text{and} \quad \dot{q} = \mathcal{H}_p(p, q). \tag{5.17}$$

Following the line of [28], the Leapfrog or Störmer-Verlet scheme for the partitioned system (5.17) is defined as follows:

Definition 5.12 (Leapfrog or Störmer-Verlet, [29])**.** Let the initial value problem (5.17) be given with initial values $p_n, q_n \in \mathbb{R}^n$. The differential equations can be integrated numerically as

$$\begin{aligned} p_{n+\frac{1}{2}} &= p_n - \tfrac{h}{2}\mathcal{H}_q(p_n, q_n), \\ q_{n+1} &= q_n + \tfrac{h}{2}\Big(\mathcal{H}_p(p_{n+\frac{1}{2}}, q_n) + \mathcal{H}_p(p_{n+\frac{1}{2}}, q_{n+1})\Big), \\ p_{n+1} &= p_{n+\frac{1}{2}} - \tfrac{h}{2}\mathcal{H}_q(p_{n+\frac{1}{2}}, q_{n+1}) \end{aligned} \tag{5.18}$$

which is called Leapfrog or Störmer-Verlet method.

This Störmer-Verlet method is composed of two symplectic Euler steps

$$p_{n+1} = p_n - h\mathcal{H}_q(p_{n+1}, q_n), \qquad q_{n+1} = q_n + h\mathcal{H}_p(p_{n+1}, q_n) \tag{5.19a}$$

or

$$p_{n+1} = p_n - h\mathcal{H}_q(p_n, q_{n+1}), \qquad q_{n+1} = q_n + h\mathcal{H}_p(p_n, q_{n+1}) \tag{5.19b}$$

of convergence order 1. More precise, the symplectic Euler scheme (5.19a) is applied on the initial values (p_0, q_0) with half step size to reach $(p_{n+\frac{1}{2}}, q_{n+\frac{1}{2}})$ and afterwards the symplectic Euler scheme (5.19b) leads to (p_1, q_1):

$$p_{n+\frac{1}{2}} = p_n - \frac{h}{2}\mathcal{H}_q(p_{n+\frac{1}{2}}, q_n), \tag{5.20a}$$

$$q_{n+\frac{1}{2}} = q_n + \frac{h}{2}\mathcal{H}_p(p_{n+\frac{1}{2}}, q_n) \tag{5.20b}$$

$$q_{n+1} = q_{n+\frac{1}{2}} + \frac{h}{2}\mathcal{H}_p(p_{n+\frac{1}{2}}, q_{n+1}) \tag{5.20c}$$

$$p_{n+1} = p_{n+\frac{1}{2}} - \frac{h}{2}\mathcal{H}_q(p_{n+\frac{1}{2}}, q_{n+1}), \tag{5.20d}$$

Finally, the combination of step (5.20b) and (5.20c) leads to the Störmer-Verlet scheme (5.18). Due to the symmetry in the construction, the leading error terms of the symplectic Euler schemes vanish in order that the Leapfrog scheme has convergence order $p = 2$.

For separable Hamiltonians $H(p, q) = T(p) + U(q)$, the Störmer-Verlet scheme simplifies to

$$\begin{aligned} p_{n+\frac{1}{2}} &= p_n - \tfrac{h}{2}\mathcal{H}_q(q_n), \\ q_{n+1} &= q_n + h\mathcal{H}_p(p_{n+\frac{1}{2}}), \\ p_{n+1} &= p_{n+\frac{1}{2}} - \tfrac{h}{2}\mathcal{H}_q(q_{n+1}) \end{aligned} \tag{5.21}$$

and the scheme turns to be explicit, i.e. it is composed of three single explicit Euler steps with step size h or $\frac{h}{2}$.

Furthermore, it is a geometric integration scheme in the sense that it is symplectic and thus volume-preserving, symmetric and time-reversible. All these properties are well-known and can be shown in a straightforward way. It is shown in [29] that the composition of one or more symplectic, volume-preserving, symmetric or time-reversible schemes is again symplectic, volume-preserving, symmetric or time-reversible.

The symplecticity can be shown by computing the determinant of the Jacobians

$$\left(\frac{\partial(p_{n+\frac{1}{2}}, q_n)}{\partial(p_n, q_n)}\right) = \begin{pmatrix} 1 & -\frac{h}{2}\mathcal{H}_{qq}(q_n) \\ 0 & 1 \end{pmatrix}, \tag{5.22}$$

$$\left(\frac{\partial(p_{n+\frac{1}{2}}, q_{n+1})}{\partial(p_{n+\frac{1}{2}}, q_n)}\right) = \begin{pmatrix} 1 & 0 \\ h\mathcal{H}_{pp}(p_{n+\frac{1}{2}}) & 1 \end{pmatrix}, \tag{5.23}$$

$$\left(\frac{\partial(p_{n+1}, q_{n+1})}{\partial(p_{n+\frac{1}{2}}, q_{n+1})}\right) = \begin{pmatrix} 1 & -\frac{h}{2}\mathcal{H}_{qq}(q_{n+1}) \\ 0 & 1 \end{pmatrix} \tag{5.24}$$

of the three single steps which all take the values 1 in order that the single steps are symplectic. Thus, also the combination of the steps is symplectic and volume preservation follows immediately.

If the part $T(p)$ of the Hamiltonian is symmetric, i.e. it holds $T(p) = -T(-p)$, the Leapfrog scheme is also time-reversible. Symmetry can be easily shown by exchanging $(h, p_n, q_n) \leftrightarrow (-h, p_{n+1}, q_{n+1})$. Based on that, a comparison of $\rho \circ \Psi_h(p_0, q_0)$ and $\Psi_h(-p_0, q_0)$ shows that both terms coincide with the result that the Leapfrog scheme is time-reversible.

It has to be mentioned that the scheme (5.18) is known as *pqp*-Leapfrog scheme but there is also the possibility to use a *qpq*-Leapfrog scheme

$$\begin{aligned} q_{n+\frac{1}{2}} &= p_n - \frac{h}{2}\mathcal{H}_p(p_n, q_n), \\ p_{n+1} &= p_n + h\mathcal{H}_q(p_n, q_{n+\frac{1}{2}}), \\ q_{n+1} &= q_{n+\frac{1}{2}} - \frac{h}{2}\mathcal{H}_p(p_{n+1}, q_{n+\frac{1}{2}}) \end{aligned} \tag{5.25}$$

Both schemes (5.18) and (5.25) lead to different results of convergence order 2. The Leapfrog scheme can also be rewritten as partitioned Runge-Kutta scheme as given in definition 5.16 with two stages and coefficients $a_{21} = b_1 = b_2 = \frac{1}{2}, \hat{a}_{21} = 1, \hat{b}_2 = 1$. All other coefficients of the scheme are zero.

5.2.2 Leapfrog for Lie Groups

The system of differential equations on Lie groups and Lie algebras given in equation (5.1) in the beginning of this chapter is achieved via the Hamiltonian system

$$\dot{Y} = \mathcal{H}_A = A \cdot Y \qquad\qquad \dot{A} = -\mathcal{H}_Y = F(Y). \tag{5.26}$$

Here, Y is in the Lie group G, A in the Lie algebra $\mathfrak{g}$ and $F : G \mapsto \mathfrak{g}$ a function that maps from the Lie group G to the Lie algebra $\mathfrak{g}$. The proper initial values (Y_n, A_n) have to be a pair of Lie group / Lie algebra elements such that $(Y_n, A_n) \in G \times \mathfrak{g}$.

All in all, the initial value problem is given in the following definition.

Definition 5.13 (Lie group / Lie algebra initial value problem)**.** Let G be a matrix

Lie group and $\mathfrak{g}$ its associated matrix Lie algebra. In addition, let $Y(t) \in G$, $A(Y(t)) \in \mathfrak{g}$ and $F : G \mapsto \mathfrak{g}$ be a Lie algebra-valued function.

Then, the differential equations

$$\dot{Y}(t) = A(Y(t)) \cdot Y(t) \qquad \text{with initial values } A_n \in \mathfrak{g}, Y_n \in G, \tag{5.27a}$$

$$\dot{A}(t) = F(Y(t)) \qquad \text{with initial values } Y_n \in G \tag{5.27b}$$

are defined as Lie group / Lie algebra initial value problem. Finally, the numerical approximations Y_{n+1} are settled in the Lie group G and A_{n+1} in the Lie algebra $\mathfrak{g}$.

An application of the Leapfrog scheme (5.18) for the Abelian case would lead to

$$A_{n+\frac{1}{2}} = A_n - \tfrac{h}{2}\mathcal{H}_Y(A_n, Y_n), \tag{5.28a}$$

$$Y_{n+1} = Y_n + \tfrac{h}{2}\Big(\mathcal{H}_A(A_{n+\frac{1}{2}}, Y_n) + \mathcal{H}_A(A_{n+\frac{1}{2}}, Y_{n+1})\Big), \tag{5.28b}$$

$$A_{n+1} = A_{n+\frac{1}{2}} - \tfrac{h}{2}\mathcal{H}_Y(A_{n+\frac{1}{2}}, Y_{n+1}). \tag{5.28c}$$

The computation of A is performed in the Lie algebra such that it can be treated as an equation in the Abelian case. Unfortunately, the update of Y reads

$$Y_{n+1} = Y_n + \frac{h}{2}\Big(A_{n+\frac{1}{2}} \cdot Y_n + A_{n+\frac{1}{2}} \cdot Y_{n+1}\Big) \tag{5.29}$$

in order that the result would not be in the Lie group any more because the matrix Lie group is just closed under multiplication.

This problem is already considered in paragraphs 3.1.2 and the Lie group step in the Leapfrog scheme has to be adapted according to one of the methods described in section 3.2. Based on the local parameterization $Y(t) = \Psi(\Omega(t))$ (see equation (3.5)), the differential equation (5.27a) can be expressed as differential equation $\dot{\Omega} = \big(\mathrm{d}\,\Psi_{\Omega}^{-1}\big)A$ (see (3.11)) in the Lie algrebra whose result is mapped in the Lie group via the local parameterization. The most common way is the choice of $\Psi :=$ exp with the result that $\dot{\Omega}$ can be expressed as $\dot{\Omega} = A$ for schemes of convergence order two. This is a kind of Lie-Euler step described in equation (3.30). So, step (5.28b) changes from (5.29) to

$$\Omega_{n+1} = \Omega_n + hA_{n+\frac{1}{2}}, \tag{5.30}$$

followed by the mapping $Y_{n+1} = \exp(\Omega_{n+1})Y_n$. In particular, $\dot{\Omega}$ depends just on a single variable in order that equation (5.30) turns to be explicit. Moreover, the initial value Ω_n takes the value 0.

All in all, the Lie group initial value problem (5.27) can be solved with the Leapfrog method defined below.

Definition 5.14 (Leapfrog for Lie group / Lie algebra problems)**.** Let the Lie group / Lie algebra initial value problem (5.27) be given. Then, the Leapfrog or

Störmer-Verlet scheme is defined as

$$\begin{aligned} A_{n+\frac{1}{2}} &= A_n + \tfrac{h}{2} F(Y_n), \\ Y_{n+1} &= \exp(hA_{n+\frac{1}{2}}) Y_n, \\ A_{n+1} &= A_{n+\frac{1}{2}} + \tfrac{h}{2} F(Y_{n+1}). \end{aligned} \tag{5.31a}$$

Here, the aforementioned equations (5.1) are alternately solved starting and ending with one half-step.

Another possibility is

$$\begin{aligned} Y_{n+\frac{1}{2}} &= \exp(\tfrac{h}{2} A_n)\, Y_n, \\ A_{n+1} &= A_n + hF(Y_{n+\frac{1}{2}}), \\ Y_{n+1} &= \exp(\tfrac{h}{2} A_{n+1}) Y_{n+\frac{h}{2}}. \end{aligned} \tag{5.31b}$$

Both schemes are explicit and deliver different numerical solutions of convergence order $p = 2$. As shown in paragraph 5.2.1 the Leapfrog method is volume-preserving since it is composed of three volume-preserving steps. The time-reversibility follows for separable Hamiltonians $\mathcal{H}(Y, A) = T(A) + U(Y)$ with symmetric function $T(A)$. In the following, we concentrate on the shape of (5.31a) for our discussion.

The Leapfrog scheme is quite important because it is used as basis for the development of other higher order methods. In Lattice QCD, for example, splitting methods, Omelyan methods or force-gradient methods are widely known as higher-order methods based on the Leapfrog scheme.

The common Leapfrog method (5.31) for Hamiltonian systems formulated as Lie group / Lie algebra problems stated in definition 5.13 uses the exponential function as local parameterization of the Lie group element $Y(t)$. As mentioned in section 4.2, also other local parameterization are possible.

5.2.3 Leapfrog using the Cayley Mapping

Usually, the exponential function is used in the Leapfrog method for the mapping between Lie algebra and Lie group:

$$\mathfrak{g} \mapsto G: \quad A_{n+\frac{1}{2}} \mapsto Y_{n+1} = \exp(hA_{n+\frac{1}{2}})\, Y_n. \tag{5.32}$$

As mentioned in section 4.2, another possibility for quadratic Lie groups as, for example, $SU(N, \mathbb{C})$ is the usage of the Cayley mapping (4.24)

$$\mathfrak{g} \mapsto G: \quad \Omega \mapsto \operatorname{cay}(\Omega) = \big(I - \Omega(t)\big)^{-1} \big(I + \Omega(t)\big).$$

such that the mapping $\mathfrak{g} \mapsto G$ reads

$$\begin{aligned} A_{n+\frac{1}{2}} \mapsto Y_{n+1} &= \operatorname{cay}(h/2\ A_{n+\frac{1}{2}})\ Y_n \\ &= \big(I - h/2\ A_{n+\frac{1}{2}}\big)^{-1}\big(I + h/2\ A_{n+\frac{1}{2}}\big)\ Y_n \end{aligned} \tag{5.33}$$

This is an alternative to the exponential function which introduces no model error. Since the exchange of the mapping from the Lie algebra to the Lie group is no special property of geometric integration schemes, the details can be found in section 4.2. Here, just its geometric properties are discussed.

Taking the Cayley transform as local parameterization, the Leapfrog method (5.31) for coupled Lie group / Lie algebra problems will turn into the Cayley-Leapfrog method:

Definition 5.15 (Cayley-Leapfrog for coupled Lie group / Lie algebra problems)**.** Let the coupled Lie group / Lie algebra initial value problem mentioned in definition 5.13 be given. Then, the Cayley-Leapfrog or Cayley-Störmer-Verlet scheme is defined as

$$\begin{aligned} A_{n+\frac{1}{2}} &= A_n + \frac{h}{2}F(Y_n), \\ Y_{n+1} &= \operatorname{cay}(0.5hA_{n+\frac{1}{2}})Y_n, \\ A_{n+1} &= A_{n+\frac{1}{2}} + \frac{h}{2}F(Y_{n+1}). \end{aligned} \tag{5.34}$$

Here, the Lie-Euler step used in (5.31) is exchanged against a Cayley-Lie-Euler step. Here, the aforementioned equations (5.1) are alternately solved starting and ending with one half-step.

It is of interest if the Cayley-Leapfrog scheme is indeed a geometric integration scheme. So, symmetry, time-reversibility and volume-preservation have to be investigated. For Lattice QCD, these properties are proven for the Lie group $SU(3,\mathbb{C})$ in a recent Bachelor thesis [44]. The time-reversibility holds for all quadratic Lie groups and the volume-preservation for all Lie groups with determinant one. In chapter 7.2.2, the Cayley-Leapfrog method is used in a HMC simulation of a lattice gauge field, leading to promising results for the usage of this method.

5.3 Symmetric Partitioned Runge-Kutta Methods

Based on partitioned Runge-Kutta methods for the Abelian case in $\mathbb{R}^n$, partitioned Runge-Kutta methods for Lie group problems (5.1),

$$\dot{Y}(t) = A(t)Y(t), \qquad \dot{A}(t) = F(Y(t)),$$

can be developed and afterward adapted for being symmetric partitioned Runge-Kutta schemes for geometric Lie group problems.

This section starts with partitioned Runge-Kutta schemes for $\mathbb{R}^n$ for the differential equations

$$\dot{y} = f(y, z), \qquad z' = g(y, z) \tag{5.35}$$

and an investigation of its symmetry. Then, the partitioned Runge-Kutta scheme is adjusted to Lie group problems (5.1) similar to [64]. This means, a Runge-Kutta scheme for Lie groups has to be used for the first equation whereas the 2nd equation goes along with the special case $z' = g(y)$. Using Munthe-Kaas Runge-Kutta schemes, the partitioned Runge-Kutta scheme takes place in the Lie algebra and is combined with a mapping from the Lie algebra to the Lie group whenever an evaluation in the Lie group is needed. Afterwards, the symmetry of partitioned Munthe-Kaas Runge-Kutta methods is discussed and the scheme is adapted to be symmetric based on three different changes. Finally, a set of coefficients for one version of a symmetric partitioned Munthe-Kaas Runge-Kutta method is derived.

5.3.1 The Abelian Case

We start with a short introduction of partitioned Runge-Kutta methods following the ideas of Hairer et al stated in [29].

Definition 5.16 (Partitioned Runge-Kutta Method [29])**.** Let the differential equations $\dot{y} = f(y, z)$ and $\dot{z} = g(y, z)$ with initial values $(y(t_0), z(t_0)) = (y_n, z_n)$ be given. Furthermore, let s be an integer and a_{ij}, b_i be real coefficients for $i, j = 1, \dots, s$.

The method

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i, \qquad k_i = f(y_n + h\sum_{j=1}^{s} a_{ij}k_j, z_n + h\sum_{j=1}^{s} \hat{a}_{ij}l_j), \tag{5.36a}$$

$$z_{n+1} = z_n + h\sum_{i=1}^{s} \hat{b}_i l_i, \qquad l_i = g(y_n + h\sum_{j=1}^{s} a_{ij}k_j, z_n + h\sum_{j=1}^{s} \hat{a}_{ij}l_j) \tag{5.36b}$$

is called partitioned Runge-Kutta method with s stages. It is an explicit method if it holds $a_{ij} = \hat{a}_{ij} = 0$ for all $i \leq j$, otherwise it is an implicit method.

In the Abelian case, the partitioned Runge-Kutta method Φ_h is symmetric if the coefficients of its adjoint method $\Phi_h^* = \Phi_{-h}^{-1}$ (see definition 5.10) in fulfill the well-known symmetry conditions

$$b_i = b_{s+1-i}, \qquad a_{ij} + a_{s+1-i,s+1-j} = b_j \tag{5.37a}$$

$$\hat{b}_i = \hat{b}_{s+1-i}, \qquad \hat{a}_{ij} + \hat{a}_{s+1-i,s+1-j} = \hat{b}_j \tag{5.37b}$$

for $i, j = 1, \ldots, s$, see [29]. These symmetry conditions can be achieved exchanging (y_{n+1}, z_{n+1}, h) against $(y_n, z_n, -h)$ in the method Φ_h itself. Rearranging for (y_{n+1}, z_{n+1}) leads to the adjoint method Φ_h^*.

We are interested in equations of type (5.1) which is related to the special case

$$\dot{y} = f(y, z), \qquad\qquad z' = g(y) \tag{5.38}$$

with partitioned Runge-Kutta method

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i, \qquad k_i = f(y_n + h\sum_{j=1}^{s} a_{ij}k_j, z_n + h\sum_{j=1}^{s} \hat{a}_{ij}l_j), \tag{5.39a}$$

$$z_{n+1} = z_n + h\sum_{i=1}^{s} \hat{b}_i l_i, \qquad l_i = g(y_n + h\sum_{j=1}^{s} a_{ij}k_j) \tag{5.39b}$$

Here, the symmetry conditions (5.37) also have to be fulfilled. Now, we adapt the scheme (5.39) to a Lie group / Lie Algebra problem (5.1).

5.3.2 The Non-Abelian Case

Concerning the system of differential equations $\dot{Y} = AY, \dot{A} = F(Y)$, the partitioned Runge-Kutta method has to be mixed with a Lie group Runge-Kutta method. Here, it is possible to use either Crouch-Grossmann or Munthe-Kaas schemes. Moreover, the local parameterization can be chosen in an arbitrary way. As an example, Engø describes partitioned Runge-Kutta methods based on Crouch-Grossmann methods in [18] which use either the exponential function or the Cayley transform as local parameterization. In this thesis, just partitioned Munthe-Kaas Runge-Kutta methods based on the exponential function are used.

In general, Munthe-Kaas schemes are solved in the linear space of the Lie algebra and use a projection to the Lie group whenever needed. Additionally, the function $f_q = f_{p-2}$ has to be adapted according to the desired convergence order p of the scheme. For example, for $p = 2$ it is sufficient to use $f_0(\Omega, A) = A$ such that a partitioned Munthe-Kaas Runge-Kutta scheme can coincide with the Leapfrog method of definition 5.14. Next, the Munthe-Kaas Runge-Kutta method mentioned in algorithm 3.22 in chapter 3 is adapted for partitioned Runge-Kutta methods:

Algorithm 5.17. (Partitioned Munthe-Kaas Runge-Kutta Method)
Let G be a matrix Lie group and $\mathfrak{g}$ its associated Lie algebra and consider the Lie group / Lie algebra problem (5.27)

$$\begin{aligned} \dot{Y}(t) &= A(Y(t)) \cdot Y(t) && \text{with initial values } A_n \in \mathfrak{g}, Y_n \in G, \\ \dot{A}(t) &= F(Y(t)) && \text{with initial values } Y_n \in G. \end{aligned}$$

Then, the step $(Y_n, A_n) \mapsto (Y_{n+1}, A_{n+1})$ is defined as follows:

1. Consider the differential equations

$$(\mathrm{d}\exp_\Omega^{-1}(A) \approx)\ \ \dot{\Omega} = \sum_{k\leq 0}^{q} \frac{B_k}{k!}\,\mathrm{ad}_\Omega^k(A) =: f_q(\Omega, A), \qquad \dot{A} = F(Y)$$

   with $\Omega(t_0) = \Omega_n$ and initial values Y_n, A_n and $\Omega_n = 0$.

2. Apply a partitioned Runge-Kutta method (explicit or implicit)

$$\Omega_{n+1} = \Omega_n + h\sum_{i=1}^{s} b_i K_i, \qquad A_{n+1} = A_n + h\sum_{i=1}^{s} \hat{b}_i L_i, \tag{5.40a}$$

   with increments

$$K_i = f_q\Big(\bar{\Omega}_i, A(\bar{Y}_i)\Big), \qquad L_i = F\Big(\exp(\bar{\Omega}_i)Y_n\Big) \tag{5.40b}$$

   and internal stages

$$\bar{\Omega}_i = \Omega_n + h\sum_{j=1}^{s} a_{ij}K_j, \qquad \bar{Y}_i = \exp(A_n + h\sum_{j=1}^{s} \hat{a}_{ij}L_j)\ Y_n. \tag{5.40c}$$

   The initial values (Y_n, A_n) and $\Omega_n = 0$ and lead to the approximations

$$\Omega_{n+1} \approx \Omega(t_0 + h) \quad \text{and} \quad A_{n+1} \approx A(t_0 + h). \tag{5.41}$$

3. Define the numerical solution by

$$(Y_{n+1}, A_{n+1}) \quad \text{with} \quad Y_{n+1} = \exp(\Omega_{n+1})\ Y_n. \tag{5.42}$$

Here, the steps 1 and 3 are performed due to the local parameterization of the Lie group which is also used in the computation of the internal stages $\bar{Y}_i$ and the increments L_i. The partitioned Runge Kutta method itself is solved in the linear space of the Lie algebra $\mathfrak{g}$ for (Ω_{n+1}, A_{n+1}) in step 2. Here, the specialty is that the initial value Ω_n is always set to zero for consistency of the initial values (Ω_n, Y_n) with the transformation $Y(t) = \exp(\Omega(t))Y_n$. Furthermore, the increments L_i are computed through a mapping F from the Lie group G to the Lie algebra $\mathfrak{g}$. Here, the Lie group also has to be obtained via an application of the exponential function on the term $\Omega_n + h\sum_{j=1}^{s} a_{ij}K_j$.

Comparing the partitioned Munthe-Kaas Runge-Kutta scheme given in algorithm 5.17 with the partitioned Runge-Kutta scheme stated in equations (5.39) there are a few differences:

1. there is an additional mapping $Y_{n+1} = \exp(\Omega_{n+1})Y_n$,

2. the internal stages $\bar{\Omega}_i$ occur in the Lie algebra (through K_i) as well as in the evaluation of the exponential function inside L_i,

3. the initial value Ω_n is always 0.

So, it has to be checked if the symmetry conditions (5.37) are still valid.

Symmetry. The symmetry is investigated using the short-hand notation

$$(Y_{n+1}, A_{n+1}) = \Phi_h(Y_n, A_n) \tag{5.43}$$

with Φ_h being the partitioned Munthe-Kaas Runge-Kutta method. Φ_h would be symmetric, if its adjoint $\Phi_h^*(Y_n, A_n)$ equals

$$\Phi_h^*(Y_n, A_n) = \Phi_{-h}^{-1}(Y_{n+1}, A_{n+1}), \tag{5.44}$$

i.e. the values $(Y_n, A_n, Y_{n+1}, A_{n+1}, h)$ have to be exchanged according to

$$(Y_n, A_n, h) \leftrightarrow (Y_{n+1}, A_{n+1}, -h) \quad \text{and} \quad \Omega_{n+1} \leftrightarrow -\Omega_{n+1}.$$

The replacement of $\Omega_{n+1} \leftrightarrow -\Omega_{n+1}$ is implied by equation (3.47) for consistency reasons: it has to hold

$$Y_{n+1} = \exp(\Omega_{n+1})Y_n \quad \Leftrightarrow \quad Y_n = \exp(-\Omega_{n+1})Y_{n+1}.$$

Comparing the coefficients of $\Phi(Y_n, A_n)$ with them of $\Phi^*(Y_n, A_n)$ of equation (5.44) leads to the symmetry conditions for general partitioned Runge-Kutta schemes for Lie group / Lie algebra problems of type (5.27)

$$b_i = b_{s+1-i} \qquad \text{(due to } \Omega^*_{n+1}) \tag{5.45a}$$

$$\hat{b}_i = \hat{b}_{s+1-i} \qquad \text{(due to } A^*_{n+1}) \tag{5.45b}$$

$$\hat{a}_{ij} = \hat{b}_{s+1-j} - \hat{a}_{s+1-i,s+1-k} \qquad \text{(due to } \bar{Y}_i \text{ in } K_i^*) \tag{5.45c}$$

$$a_{ij} = -a_{s+1-i,s+1-j} \qquad \text{(due to } \bar{\Omega}_i \text{ in } K_i^*) \tag{5.45d}$$

$$a_{ij} = b_{s+1-j} - a_{s+1-i,s+1-k} \qquad \text{(due to } \bar{\Omega}_i \text{ in } L_i^*) \tag{5.45e}$$

$$a_{ij} = d_i \cdot b_j \qquad \text{(due to } L_i^*) \tag{5.45f}$$

for $i, j = 1, \ldots, s$ and constant d_i. Here, the conditions for Abelian partitioned Runge-Kutta schemes stated in (5.37) can be found in the first four equations plus two additional ones (5.45e) and (5.45f) concerning the coefficients a_{ij}. The three conditions (5.45d-5.45f) for a_{ij} obviously lead to contradictions which are discussed in the following.

Equation (5.45e) arises from the internal stages $\bar{\Omega}_i$ of the increments K_i inside the Lie algebra. Its shape finally depends on the initial values $\Omega_n = 0$. The other equations (5.45d) and (5.45f) are related to each other. They occur because the adjoint L_i^* of $L_i = F\big(\exp(\bar{\Omega}_i)Y_n\big)$ mentioned in equation (5.40b) reads

$$\begin{aligned} L_i^* = F\Big(\exp(\bar{\Omega}_i^*)Y_{n+1}\Big) &= F\Big(\exp(\bar{\Omega}_i^*)\exp(\Omega_{n+1})Y_n\Big) \\ &= F\Big(\exp(\bar{\Omega}_i^* + \Omega_{n+1})Y_n\Big) \end{aligned} \tag{5.46}$$

if the matrices $\bar{\Omega}_i^*$ and Ω_{n+1} commute. Here, Y_n is exchanged towards $Y_{n+1} = \exp(\Omega_{n+1})Y_n$. One simple case of commuting matrices is one matrix being a multiple of the other, i.e. the coefficients a_{ij} must be a multiple of the coefficients b_j for $j = 1, \dots, s$. So, condition (5.45f) is a prerequisite for condition (5.45d).

Theoretically, all three conditions can be simultaneously fulfilled if all coefficients b_j and $a_{i,j}$, $j = 1, \dots, s$, are zero. Since this choice destroys the numerical scheme (the numerical approximations stays constant and not even convergence order one would be fulfilled), this is not valid.

The conflict between the conditions (5.45e) and (5.45d) can be solved

- if one of the conditions disappears
- or if the increments K_i and L_i are evaluated at different points, i.e. a new set of coefficients (c_{ij}), $i, j = 1, \dots, s$ is introduced.

Of course, also a combination of both solution leads to the desired result as described in [64]. So, there are several ways of fixing the contradiction in the symmetry conditions which are described next.

Symmetric Partitioned Munthe-Kaas Runge-Kutta Methods. Based on algorithm 5.17, the partitioned Munthe-Kaas Runge-Kutta method can be turned into a symmetric partitioned Runge-Kutta scheme.

Concerning the special case of a small convergence order p, i.e. $p < 3$, the increments

$$K_i = f_0(\Omega_i, A(\bar{Y}_i)) = A(\bar{Y}_i)$$

do not depend on the internal stages $\bar{\Omega}_i$ for all $i = 1, \dots, s$. So, algorithm 5.17 can be applied without changes because condition (5.45e) vanishes and (5.45d) and (5.45f) can be simultaneously solved. One example is the Leapfrog method for Lie groups. Here, the partitioned Munthe-Kaas Runge-Kutta methods with coefficients

$$a_{21} = \frac{1}{2},\ b_1 = \frac{1}{2},\ b_2 = \frac{1}{2}, \qquad\qquad \hat{a}_{21} = 1,\ \hat{b}_1 = 1 \tag{5.47}$$

leads to the Leapfrog method introduced in definition 5.14.

For a general convergence order $p > 2$, just the increments L_i, $i = 1, \dots, s$, have to be adapted in the sense that the symmetry conditions (5.45d) - (5.45f) do not occur altogether for an arbitrary convergence order. This can be achieved by one of the following changes of the increments of algorithm 5.17:

1.
$$L_i = F\Big(\exp\big(\bar{\Omega}_i\big)\exp\big(\tfrac{1}{2}\Omega_{n+1}\big)Y_n\Big), \tag{5.48}$$

2.
$$L_i = F\Big(\exp(\bar{\omega}_i)Y_n\Big) \quad \text{with} \quad \bar{\omega}_i = \Omega_n + h\sum_{j=1}^{s} c_{ij}K_j, \tag{5.49}$$

3. or
$$L_i = F\Big(\exp\big(\bar{\omega}_i\big)\exp\big(\tfrac{1}{2}\Omega_{n+1}\big)Y_n\Big) \quad \text{with} \quad \bar{\omega}_i = \Omega_n + h\sum_{j=1}^{s} c_{ij}K_j. \tag{5.50}$$

Case 1. In the first case, the focus can be put on avoiding the multiplication of exponential functions inside the adjoint method. Here, an additional term X can be introduced in the increments L_i such that L_i can be rewritten as

$$L_i = F\Big(\exp(\bar{\Omega}_i)\cdot X \cdot Y_n\Big). \tag{5.51}$$

Now, the adjoint of L_i reads

$$L_i^* = F\Big(\exp\big(\bar{\Omega}_i^*\big)X^*Y_{n+1}^*\Big) = F\Big(\exp\big(\bar{\Omega}_i^*\big)\cdot X^* \cdot \exp\big(\Omega_{n+1}^*\big)Y_n^*\Big). \tag{5.52}$$

The first and the last term in L_i and L_i^* are similar. So we put our emphasis on an expression for X such that X in L_i has the same structure as $X^*\cdot\exp(\Omega_{n+1})$ in the adjoint L_i^*. This would have the advantage that just the coefficients of $\bar{\Omega}_i$ and $\bar{\Omega}_i^*$ have to be compared since the rest is of equal form. Indeed, according to (5.51) the choice

$$X = \exp\big(\tfrac{1}{2}\Omega_{n+1}\big) \tag{5.53}$$

leads to

$$L_i = F\Big(\exp\big(\bar{\Omega}_i\big)\exp\big(\tfrac{1}{2}\Omega_{n+1}\big)Y_n\Big). \tag{5.48}$$

The adjoint L_i^* is composed of

$$\bar{\Omega}_i^*, \quad \Omega_{n+1}^* = -\Omega_{n+1} \quad \text{and} \quad Y_n^* = Y_{n+1} = \exp(\Omega_{n+1})Y_n. \tag{5.54}$$

Thus, it holds

$$\begin{aligned} L_i^* &= F\Big(\exp\big(\bar{\Omega}_i^*\big)\cdot\exp\big(-\tfrac{1}{2}\Omega_{n+1}\big)\cdot\exp\big(\Omega_{n+1}\big)Y_n\Big) \\ &= F\Big(\exp\big(\bar{\Omega}_i^*\big)\cdot\exp\big(\tfrac{1}{2}\Omega_{n+1}\big)Y_n\Big) \end{aligned} \tag{5.55}$$

such that the increment L_i and its adjoint have the same structure. Thus, after a change of the increments L_i, $i = 1, \dots, s$, according to equation (5.48) just the symmetry conditions (5.45a)-(5.45e) have to be fulfilled and (5.45d) and (5.45f) vanish.

Unfortunately, the numerical solution Ω_{n+1} has to be used in the term $\exp(\Omega_{n+1})$ has to be introduced inside the function F defining the increments $L_i, i = 1, \dots, s$ for A_{n+1}. For this, the symmetric partitioned Munthe-Kaas Runge-Kutta method turns to be fully implicit and the effort for its computation is increased dramatically compared to the not symmetric partitioned Munthe-Kaas Runge-Kutta scheme.

Case 2. Second, a new set of coefficients can be introduced such that $\bar{\Omega}_i$ is replaced by $\bar{\omega}_i$ either in the internal stages of K_i or L_i for $i = 1, \dots, s$. For example, the increments L_i, $i = 1, \dots, s$ from equation (5.40b) can be exchanged against

$$L_i = F\Big(\exp(\bar{\omega}_i)Y_n\Big) \quad \text{with} \quad \bar{\omega}_i = \Omega_n + h\sum_{j=1}^{s} c_{ij}K_j$$

mentioned in equation (5.49). In this case, equations (5.45d) and (5.45f) are replaced through

$$c_{ij} = b_{s+1-j} - c_{s+1-i,s+1-k} \qquad \text{(due to } \bar{\Omega}_i \text{ in } L_i^*) \qquad (5.56)$$

$$c_{ij} = d_i \cdot b_j \qquad \text{(due to shape of } L_i^*) \qquad (5.57)$$

for $i, j = 1, \dots, s$ and constant d_i.

Case 3. A combination of both changes (5.48) and (5.49) leads to the change (5.50) and to the symmetry conditions

$$b_i = b_{s+1-i}, \qquad \hat{b}_i = \hat{b}_{s+1-i}, \qquad (5.58a)$$

$$a_{ij} = -a_{s+1-i,s+1-j} \qquad \hat{b}_j = \hat{a}_{ij} + \hat{a}_{s+1-i,s+1-j}, \qquad (5.58b)$$

$$c_{ij} = -c_{s+1-i,s+1-j} \qquad (5.58c)$$

for $i, j = 1, \dots, s$.

All in all, we have seen that there are three ways of deriving symmetric partitioned Munthe-Kaas Runge-Kutta methods for the Lie group / Lie algebra problem (5.27) for a general convergence order p. The increments L_i can be computed according to equation (5.48), (5.49) or (5.50). The next step in the development of the new scheme would be a derivation of a set of coefficients. Here, it is essential that these coefficients meet the order conditions for the desired convergence order.

Compared to the partitioned Runge-Kutta methods for the Abelian case, the increments K_i and L_i, $i = 1, \dots, s$ change. L_i changes due to the choice (5.48),(5.49) or (5.50) and $K_i = f_q$ due to the desired convergence order which depend to its truncation index $q = p - 2$. So, some powers of the adjoint operator are included in K_i. This leads to additional order conditions. The order conditions have to be computed by a comparison of the Taylor expansion of the exact solution $(Y(t_0+h), A(t_0+h))$ and the numerical approximation (Y_{n+1}, A_{n+1}). Fortunately, this can be done via a comparison of the numerical approximation (Ω_{n+1}, A_{n+1}) with the Taylor expansion of the exact solution $(\Omega(t_0+h), A(t_0+h))$ as described by Striebel in [60]. If it holds

$$||\Omega_{n+1} - \Omega(t_0+h)|| = \mathcal{O}(h^{p+1}) \quad \text{and} \quad ||A_{n+1} - A(t_0+h)|| = \mathcal{O}(h^{p+1}), \tag{5.59}$$

then also the approximation (Y_{n+1}, A_{n+1}) is of convergence order p.

Example. We consider a symmetric partitioned Runge-Kutta scheme (5.40) with increments L_i computed according to formula (5.50). For this example, we compute the order conditions

$$p=1: \qquad \sum_i b_i = 1, \qquad \sum_i \hat{b}_i = 1 \tag{5.60a}$$

$$p=2: \qquad \sum_i b_i\hat{a}_i = \frac{1}{2}, \qquad \sum_{i,j} \hat{b}_i\Big(c_i + \frac{b_j}{2}\Big) = \frac{1}{2} \tag{5.60b}$$

$$p=3: \qquad \sum_{i,j} b_i(a_i\hat{a}_i - a_{ij}\hat{a}_j) = \frac{1}{6}, \qquad \sum_{i,j} \hat{b}_i\Big(c_i + \frac{b_j}{2}\Big)^2 = \frac{1}{3} \tag{5.60c}$$

$$\sum_i b_i\hat{a}_{ij}\Big(c_j + \frac{1}{2}\sum_k b_k\Big) = \frac{1}{6}, \qquad \sum_{i,j} \hat{b}_i\Big(c_{ij} + \frac{1}{2}b_j\Big)\hat{a}_j = \frac{1}{6} \tag{5.60d}$$

as described before. Then, for convergence order $p=3$ computed with $s=3$ stages, one set of coefficients

$$a_{21} = c_{11} = -\frac{\sqrt{3}}{6}, \qquad a_{23} = c_{33} = \frac{\sqrt{3}}{6}, b_2 = 1, \tag{5.61a}$$

$$\hat{a}_{11} = \frac{3+\sqrt{3}}{6}, \qquad \hat{a}_{21} = \frac{3+\sqrt{3}}{12}, \hat{b}_1 = \hat{b}_3 = \frac{1}{2}, \tag{5.61b}$$

$$\hat{a}_{23} = \frac{3-\sqrt{3}}{12}, \qquad \hat{a}_{31} = \frac{1}{2}, \tag{5.61c}$$

$$\hat{a}_{33} = -\frac{\sqrt{3}}{6} \tag{5.61d}$$

is derived based on the order conditions (5.60) and the symmetry conditions (5.58). Finally, these coefficients are used for the numerical simulation of a lattice gauge field which is described in paragraph 7 and also in [64]. It is remarkable that the convergence order is even. In fact, a convergence order of size $p=4$ is achieved although just a scheme of convergence order $p=3$ is developed. This depends on the symmetry of the scheme as described in [60], [64] and [62].

Probably, a usage of simply (5.48) or (5.49) would lead to less complicated order conditions such that a development of the schemes should be considered in future work. Here, the increments L_i computed according to equation(5.49) seem promising because the scheme will not become fully implicit. Another idea can be realized using [70].

Concerning Lattice QCD, the numerical integration method has to be time-reversible and volume-preserving. Time-reversibility could be easily computed via the condition (5.15). In this case, this is not done but this property is checked via the simulation of a lattice gauge field in chapter 7. The scheme is not constructed for being volume-preserving. Theoretically, the volume-preservation can be ensured by a division of the value of the determinant of the Jacobian $\partial(Y_{n+1}, A_{n+1})/\partial(Y_n, A_n)$ if the size of the Jacobian is small enough. In the case of Lattice QCD, the size of the determinant is quite large such that this is no option.

All in all, this geometric scheme is developed in [64] with focus on its application on Lattice QCD, i. e. for the usage inside the Molecular Dynamics step inside the Hybrid Monte Carlo algorithm. Here, the crucial point is that the energy difference of the Hamiltonian before and after the numerical integration scheme is small to achieve a high acceptance rate. This could also be reached with a projection method focusing on a constant Hamiltonian as mentioned in the next section.

5.4 Time-reversible Projection Schemes

This section is concerned with time-reversible and volume-preserving projection methods for the solution of

$$\dot{Y}(t) = A(t)Y(t), \qquad \dot{A}(t) = F(Y(t))$$

as given in equation (5.1). The idea is based on symmetric projection schemes introduced by Hairer for ODE systems on a manifold in the Abelian case $\mathbb{R}^n$. These methods combine a symmetric scheme with a projection on the manifold, arising in an overall symmetric and time-reversible scheme which preserves the constraint defined by the manifold. We have generalized this scheme to projection schemes, which join a symmetric, time-reversible and symplectic scheme (Leapfrog, for example) with a projection on the manifold described by the Hamiltonian, resulting in a scheme with the aforementioned properties which preserves the Hamiltonian exactly. In a further step, we adapted the method to the non-Abelian case of matrix Lie groups for equations of type (5.1) having in mind to reduce, for example, the computational costs of simulations performed in Lattice QCD.

We start from symmetric projection schemes [26] and develop time-reversible and volume-preserving projection schemes for the Abelian case. These results are generalized and applied to the non-Abelian case of Lattice QCD where coupled differential equations on Lie groups and Lie algebras arise. Here, the volume-preservation depends on the properties of the projection parameter μ – the volume is preserved if μ is constant, i.e., if it depends not on the initial values. Nevertheless, the method is applied on a gauge field in $SU(3, \mathbb{C})$ Yang-Mills theory which is described in section 7.2.4. Here, it is discussed if the numerical result implies volume-preservation or not. This section is developed in close collaboration with Prof. Dr. M. Günther and Prof. Dr. M. Striebel.

Projection Schemes

Our aim is the development of projection schemes for the Molecular Dynamics Step of the Hybrid Monte Carlo method. For this purpose, the projection scheme has to fulfill several properties:

- first of all, it has to be time-reversible to reach the correct fixed point of the Markov chain;
- then, it should be volume-preserving; otherwise the Jacobian of a system of huge dimension has to be computed;
- additionally, the Hamiltonian should remain constant.

If all these demands are met, one gets an overall time-reversible, volume-preserving projection scheme that preserves the Hamiltonian.

This scheme would be very advantageous for its utilization in the Molecular Dynamics step of the Hybrid Monte Carlo algorithm because the acceptance step can be dropped such that each proposed new configuration is used. Usually, it is of interest to achieve a high acceptance rate via a small difference in the Hamiltonian ΔH before and after the Molecular Dynamics step. Since $|\Delta H|$ is proportional to the error of the integration scheme, the convergence order p of a numerical integration scheme used in the HMC should be high. Using the projection method, there is no need for higher order methods or small step sizes, one large integration step is sufficient provided that the projection method is time-reversible and volume-preserving.

In a first step, we will construct such schemes for the Abelian case.

5.4.1 Projection Schemes for the Abelian Case

Symmetric projection schemes have been introduced by Hairer [26] to solve numerically the initial value problem

$$\dot{y} = f(y), \quad y(t_0) = y_n \tag{3.17}$$

in the Abelian case $y \in \mathbb{R}^{2n}$ subject to an invariant manifold defined by a constraint $g(y) = 0$. These schemes combine one step of a symmetric integration method Φ_h with a symmetric forward and backward projection to obtain a symmetrical method and at the same time to preserve the constraint: first of all, the initial value y_n is perturbed orthogonal to the manifold:

$$P_f^{\mu}: \quad \tilde{y}_n = y_n + G^{\top}(y_n)\mu \tag{5.62a}$$

with $G^\top(y)$ denoting the Jacobian of $g(y)$. After that, one symmetric numerical integration step is performed:

$$\Phi_h : \quad \tilde{y}_{n+1} = \Phi_h(\tilde{y}_n)\,. \tag{5.62b}$$

Finally, the result is projected back to the manifold:

$$P_b^\mu : \quad y_{n+1} = \tilde{y}_{n+1} + G^\top(y_{n+1})\mu\,. \tag{5.62c}$$

Here, the explicit perturbation P_f^μ and implicit projection step P_b^μ depend on a parameter μ that is implicitly defined by the constraint $g(y) = 0$. This method is not symplectic and not volume-preserving.

Our aim is to adapt the symmetric projection method to a simple time-reversible and volume-preserving projection scheme in the case of general Hamiltonian equations of motion given by (5.2) as

$$\dot{y} = f(y) = J^{-1}\nabla\mathcal{H}(y) \qquad \text{with} \qquad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix},$$

initial values $y(t_0) = y_n$ and the energy-preserving constraint

$$g(y) = \mathcal{H}(y) - \mathcal{H}(y_n) = 0\,. \tag{5.63}$$

Note that we regard the projection parameter μ as unknown constant, thus yielding a parametrized family of numerical approximations y_h^μ for step size h. At the end, we are interested in the scheme $y_h^{\mu_{opt}}$ defined by

$$g(y_h^{\mu_{opt}}) = \mathcal{H}(y_h^{\mu_{opt}}) - \mathcal{H}(y_n) = 0\,. \tag{5.64}$$

The aforementioned adaption can be realized by three steps:

1. replace the symmetric integration method with a symmetric and at the same time time-reversible and volume-preserving one;
2. next, ensure that the overall scheme is volume-preserving;
3. and after all, make the overall scheme time-reversible.

The first step is very similar to the symmetric projection and can be realized, for example, by use of the Leapfrog integration scheme or the implicit midpoint rule.The new scheme

$$\Psi_h^\mu := P_b^\mu \circ \Phi_h \circ P_f^\mu\,. \tag{5.65}$$

is still symmetric, but in general neither volume-preserving nor time-reversible.

Now, we investigate the volume-preservation. We see that the projection strongly depends on its projection parameter μ as stated below and, hence, volume-preservation cannot be obtained. However, if we assume that the influence of the Jacobian of the projection parameter is small, i.e. the projection parameter μ takes a constant value, volume-preservation would be possible. Based on these considerations, the overall scheme can be selected to be time-reversible.

Volume-Preservation

The volume is preserved, if

$$|\det\left(D\Psi_h^\mu(y_n)\right)| = 1 \quad \text{with} \quad D\Psi_h^\mu(y) := \frac{\partial \Psi_h^\mu(y)}{\partial y} \tag{5.66}$$

holds, i.e., it has to be checked whether the modulus of the determinant of the Jacobian is equal 1. Since the result $y_{n+1} = \Psi_h^\mu(y_n)$ is computed in three steps, a short computation shows that the method is in general not volume-preserving because the determinant of the Jacobian reads

$$|\det D\Psi_h^\mu(y_n)| = \left|\det\left(DP_b^\mu(y)\Big|_{y=(\Phi_h\circ P_f^\mu)(y_n)} \cdot D\Phi_h(y)\Big|_{y=P_f^\mu(y_n)} \cdot DP_f^\mu(y_n)\right)\right| \tag{5.67}$$

$$= \left|\det\left(DP_b^\mu(y)\Big|_{y=(\Phi_h\circ P_f^\mu)(y_n)} \cdot DP_f^\mu(y_n)\right)\right| \tag{5.68}$$

$$\text{with} \quad \det P_f^\mu(y_n) = \det\Big(I + D\,G^\top(y_n)\mu + G^\top(y_n)\frac{\partial\mu(y_n)}{y_n}\Big), \tag{5.69}$$

$$\text{and} \quad \det DP_b^\mu(y)\Big|_{y=(\Phi_h\circ P_f^\mu)(y_n)} = \det\Big(I - D\,G^\top(y_{n+1})\mu\Big)^{-1} \tag{5.70}$$

and is usually not equal to 1. Nevertheless, there is a simple way to ensure the volume-preservation requiring three changes: first of all, replace the matrix $D\,G^\top(y)$ by a constant matrix $C^\top$ such that the projection matrix $G^\top(y)$ is replaced by $C^\top\cdot y$. In doing so, the modulus of the determinant of the Jacobian reads

$$|\det D\Psi_h^\mu(y_n)| = \left|\det(I - C^\top\mu)^{-1}\cdot\det\Big(I + C^\top\mu + G^\top(y_n)\tfrac{\partial\mu(y_n)}{y_n}\Big)\right|. \tag{5.71}$$

Second, there has to be an additional sign flip in either the forward or the backward projection to achieve this. A sign flip in the backward projection gives the overall system

$$\begin{aligned}
P_f^\mu:&\qquad \tilde{y}_n = y_n + C^\top y_n\mu\\
\Phi_h:&\qquad \tilde{y}_{n+1} = \Phi_h(\tilde{y}_n)\\
P_b^\mu:&\qquad y_{n+1} = \tilde{y}_{n+1} - C^\top y_{n+1}\mu\,, \qquad\qquad \mu:\mathcal{H}\ \text{constant}\,.
\end{aligned} \tag{5.72}$$

Using the short-hand notation Ψ_h^μ defined in (5.65), we get

$$|\det D\Psi_h^\mu(y_n)| = \left|\det(I + C^\top\mu)^{-1}\cdot\det\Big(I + C^\top\mu + G^\top(y_n)\tfrac{\partial\mu(y_n)}{y_n}\Big)\right| \;=\; 1\,. \tag{5.73}$$

If the term $\frac{\partial\mu(y_n)}{y_n}$ would be zero, i.e. we assume the projection parameter $\mu(y_n)$ being constant, the method given in (5.72) is volume-preserving.

The volume-preservation depends on the fact that the projection parameter μ is independent of the initial values. It has to be investigated if this case occurs or if $\mu(y_n)$ has to be a function depending on the Hamiltonian and the initial value y_n.

So, we assume that the projection parameter does not depend on the initial value but can be chosen as constant value μ_{opt} such that equation (5.64) is fulfilled. For a working scheme, this would have to be proven depending on the model and its Hamiltonian $\mathcal{H}$. Anyway, the argumentation is continued leading to - at a first glance - promising results for Lattice QCD in paragraph 7.2.4. So, the volume-preserving and time-reversible projection method is an example that illustrates the necessity of critical examinations in the development of new methods.

Time-Reversibility

So far, we have a symmetric and symplectic projection scheme given in (5.72). It is essential, that the scheme is also time-reversible, i.e., the condition

$$\rho \circ \Psi_h^\mu = (\rho \circ \Psi_h^\mu)^{-1} \qquad \text{with} \qquad \rho = \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix} \tag{5.74}$$

consisting of blocks of size $n \times n$ has to be fulfilled. The overall system reads

$$\Psi_h^\mu(y) = \left(I + C^\top \mu\right)^{-1} \cdot \Phi_h\Big(\left(I + C^\top \mu\right)(y)\Big) \tag{5.75}$$

and with the time-reversibility of the inner one-step scheme Φ_h it can be shown that it is sufficient to choose a matrix C with block-diagonal structure according to the dimensions of q and p of $y = (q, p)^\top$.

Proof: for

$$\Psi_h^\mu = B_\mu^{-1} \cdot \Phi_h \cdot B_\mu \qquad \text{with} \qquad B_\mu := I + C^\top \mu \,.$$

we have two verify the condition for time-reversibility

$$\rho\Psi_h^\mu = (\rho\Psi_h^\mu)^{-1} \quad \Leftrightarrow \quad \rho \cdot \Psi_h^\mu \cdot \rho \cdot \Psi_h^\mu = id,$$

or equivalently,

$$\rho \cdot B_\mu^{-1} \cdot \Phi_h \cdot B_\mu \cdot \rho \cdot B_\mu^{-1} \cdot \Phi_h \cdot B_\mu \overset{!}{=} id \,.$$

Using the time-reversibility of Φ_h and the fact that ρ is idempotent, we can replace one of the one-step schemes Φ_h with $\rho \cdot \Phi_h^{-1} \cdot \rho$ and get

$$id \overset{!}{=} \rho \cdot B_\mu^{-1} \cdot \Phi_h \cdot \left(B_\mu \cdot \rho \cdot B_\mu^{-1} \cdot \rho\right) \cdot \Phi_h^{-1} \cdot \rho \cdot B_\mu \,.$$

If we assume $B_\mu \cdot \rho \cdot B_\mu^{-1} \cdot \rho = id$ we are done. A sufficient but not necessary condition would be that the matrix B_μ, respective C, is block-diagonal. □

Note that the time-reversibility does not depend on the fact that the projection parameter μ is constant. So, the projection scheme (5.72) is always time-reversible.

5.4.2 Projection Schemes for the Non-Abelian Case

In the previous section, we developed the time-reversible and volume-preserving projection scheme $\Psi_h^\mu := P_b^\mu \circ \Phi_h \circ P_f^\mu$ described in (5.72) with block-diagonal matrix C for the Hamiltonian equations of motion

$$\dot{y} = f(y) = J^{-1}\nabla\mathcal{H}(y) \quad \text{with} \quad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \text{ and } y(t_0) = y_n$$

in the Abelian case. Since $y \in \mathbb{R}^{2n}$, it can be split as $y = (p,q)^\top$ in two parts $p, q \in \mathbb{R}^n$. Since we are interested in a scheme to solve the Hamiltonian equations of motion in the non-Abelian case of Lattice QCD, we adapt the projection scheme to the Lie group / Lie algebra problem

$$\dot{Y}(t) = A(Y(t)) \cdot Y(t), \qquad\qquad \dot{A}(t) = F(Y(t))$$

mentioned in (5.1). These equations also can be formulated similar to the Abelian one:

$$\dot{y} = f(y) = J^{-1}\nabla\mathcal{H}(y), \qquad y = (Y, A)^\top, \qquad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \tag{5.76}$$

with $Y \in G$, $A \in \mathfrak{g}$ and initial values $(Y_n, A_n) \in (G \times \mathfrak{g})$. Then, the projection scheme reads in a first step

$$\Psi_h^\mu(y) = \big(I + C^\top \mu\big)^{-1} \cdot \Phi_h\Big(\big(I + C^\top \mu\big)(y)\Big) \tag{5.77}$$

and has to be adapted to the Lie group / Lie algebra structure. Here, it is convenient to split the geometric integration method Φ_h in parts Φ_h^Y and Φ_h^A related to the parts in the Lie group and Lie algebra. This can also be done for the projection scheme such that Ψ_h^μ reads

$$\Psi_{h,\mu}^Y(Y, A) = \big(I + C_Y^\top \mu\big)^{-1} \cdot \Phi_h^Y\Big(\big(I + C_Y^\top \mu\big)Y, \big(I + C_A^\top \mu\big)A\Big) \tag{5.78a}$$

$$\Psi_{h,\mu}^A(Y, A) = \big(I + C_A^\top \mu\big)^{-1} \cdot \Phi_h^A\Big(\big(I + C_Y^\top \mu\big)Y, \big(I + C_A^\top \mu\big)A\Big) \tag{5.78b}$$

with different constant projection matrices C_Y and C_A.

Due to the special structure of the system (5.76), the integration method Φ_h has to be a time-reversible and volume-preserving integration method for coupled Lie group / Lie algebra problems (5.1). Here, the Leapfrog scheme described in section

5.2 could be used.

Choice of the projection matrices. The projection steps can be adapted to the structure of the Lie group. Applying the time-reversible and volume-preserving scheme (5.75) on the aforementioned differential equations on Lie groups / Lie algebras (5.76), there is a problem in the perturbation step:

$$Y \in G \quad \rightarrow \quad Y + \mu C_Y \notin G \,. \tag{5.79}$$

As the Lie group G is not closed with respect to addition, the results of the overall projection scheme will not be elements of the Lie group. A simple way out the choice $C_Y = 0$, i.e., the Lie group elements are not perturbed. A second approach could be the perturbation step

$$Y \in G \quad \rightarrow \quad Y \cdot \mu C_Y \in G \quad \text{with} \quad C_Y \in G \,. \tag{5.80}$$

The variables $A \in \mathfrak{g}$ are elements of a linear space. Here, we have no problems with the projection step. It reads

$$A \rightarrow \Big(I + \mu C_A\Big) \cdot A \,. \tag{5.81}$$

We make a very simple choice for the scaling of the Lie group and Lie algebra elements. For the Lie group element, we take formula (5.79) with $C_Y = 0$. This means, the element Y is never projected, it stays constant.

In addition, we set $C_A = I$ for a simple scaling of the Lie algebra element A. Hence, the variable A is just scaled by a factor $1 + \mu$ in the perturbation step.

According to equation (5.78) and with $C_Y = 0$ and $C_A = I$, $\Psi_h^\mu = \big(\Psi_{h,\mu}^Y, \Psi_{h,\mu}^A\big)$ reads

$$\Psi_h^\mu(Y_n, A_n) = \begin{pmatrix} \Psi_{h,\mu}^Y(Y_n, A_n) \\ \Psi_{h,\mu}^A(Y_n, A_n) \end{pmatrix} = \begin{pmatrix} Y_{n+1} \\ A_{n+1} \end{pmatrix} = \begin{pmatrix} \Phi_h^Y\Big(Y_n, (1+\mu)A_n\Big) \\ \frac{1}{1+\mu}\Phi_h^A\Big(Y_n, (1+\mu)A_n\Big) \end{pmatrix} . \tag{5.82}$$

with

$$(I + C_Y^\top \mu)^{-1} = I \quad \text{and} \quad (I + C_A^\top \mu)^{-1} = \frac{1}{1+\mu} I.$$

So, the result of the intermediate one-step integration scheme $\Phi_h = (\Phi_h^Y, \Phi_h^A)^\top$ is rescaled by a factor $1/(1+\mu)$ for Φ_h^A.

Applying the aforementioned projection, the Lie group/Lie algebra structure is preserved – Φ_h^Y is still in the Lie group, and Φ_h^A in the corresponding Lie algebra. Furthermore, the scheme is still symmetric, time-reversible and volume-preserving. A big advantage of the projection scheme (5.82) is that there occurs no additional cost, except for the solution of the scalar equation

$$\tilde{g}(\mu) := g(y_h^{\mu_{opt}}) = \mathcal{H}(y_h^{\mu_{opt}}) - \mathcal{H}(y_n) = 0 \tag{5.83}$$

for the determination of μ^{opt}.

It has to be mentioned that there occur two open questions:

1. Takes the projection parameter μ a constant value? This is the prerequisite for the volume-dependence.
2. Is the perturbation $Y \rightarrow Y + \mu \cdot 0$ an appropriate choice? Or would it be better to use the perturbation $Y \rightarrow \mu Y \cdot C_Y$ with $C_Y \in G$?

We test the projection method (5.82) for a 2-dimensional lattice gauge field in $SU(3)$. The results can be found in section 7.2.4.

5.5 Summary

In this chapter, different geometric integration methods for the initial value Lie group / Lie algebra problem

$$\begin{aligned} \dot{Y}(t) &= A(Y(t)) \cdot Y(t), && \text{with initial values } A_n \in \mathfrak{g}, Y_n \in G \\ \dot{A}(t) &= F(Y(t)) && \text{with initial values } Y_n \in G \end{aligned}$$

stated in (5.27) are developed. These new schemes are based on common Leapfrog, partitioned Runge-Kutta and symmetric projection schemes which are adapted according to the Lie group / Lie algebra structure of the problem and due to volume-preservation, symmetry and time-reversibility.

Based on the results of section 5.2, the usual Leapfrog method with exponential function can be replaced by a Leapfrog method with a different local parameterization: the Cayley-Leapfrog method. Here, the local parameterization $\Psi : \mathfrak{g} \rightarrow G$, $\Psi(\Omega) = \exp(\Omega)$ is replaced with $\Psi(\Omega) = \mathrm{cay}(\Omega)$. This approach is very promising for quadratic Lie groups because the necessity of a truncation is omitted and thus no additional model error is introduced.

In paragraph 5.3, symmetric partitioned Munthe-Kaas Runge-Kutta methods are developed. First, it is investigated that adaption of symmetric partitioned Runge-Kutta methods for the Abelian case towards Munthe-Kaas schemes for the non-Abelian case do not lead to convergence orders higher than two. So, the increments K_i or L_i, $i = 1, \dots, s$ of partitioned Munthe-Kaas Runge-Kutta methods must be adapted. We suggest three possible options. Case 1 leads to a fully implicit partitioned MK-RK scheme whereas case 2 just introduces a new set of coefficients such that the scheme could be explicit. Case 3 combines both changes. All schemes lead to different symmetry conditions stated in paragraph 5.3.2. Concerning conver-

gence, the new schemes require require a development of additional order conditions depending on the adaption and the used truncation of $\mathrm{d}\exp_{\Omega}^{-1}$. For case 3, we compute the order conditions up to $p = 3$ and derive one set of coefficients which is applied on a lattice gauge field in $SU(3,\mathbb{C})$ Yang-Mills theory as described in [64] and section 7.2.3.
In a next step, one of the other suggested adaption should be worked out. Case 2 seems promising for the determination of an explicit symmetric partitioned Runge-Kutta method. Furthermore, the development of symmetric partitioned Munthe-Kaas Runge-Kutta methods using the Cayley transform makes sense. This class of schemes would have the advantage that the order conditions are based on increments having the same shape for all desired convergence orders.

In the last paragraph 5.4, a volume-preserving and time-reversible projection method is suggested, first for the Abelian case and then for the non-Abelian one. At a first glance, this projection seems promising but the volume-preservation suffers from the dependence of the projection parameter upon the initial values. Here, it has to be investigated if there are models with constraints $g(\mu)$ which are independent of the input data, i.e. if a constant μ leads to a constraint with desired accuracy. Turning to the non-Abelian case, there is a projection developed for a coupled Lie group / Lie algebra problem which is at least time-reversible (and in case of constant μ also volume-preserving). This scheme is applied on the Hamiltonian equations of motion for a lattice gauge field in section 7.2.4.

Chapter 6

6 Exponential Smoothing Splines

Sometimes, data with uncertainties have to be approximated avoiding undesired oscillations. Usually, the well known smoothing splines [52] are taken for the approximation of data with uncertainties where some oscillations may occur. On the other hand, the also well known exponential splines [57], [53] interpolate data avoiding oscillations not given in the data but without regarding the errors of the data.

The class of exponential smoothing splines combine both methods. They are developed in the master thesis of Werneburg [68] for the approximation of price-load curves in financial mathematics and also explained in [66] with slightly changed formula for the determination of the Lagrange parameter. In this chapter, the details of the construction of the exponential smoothing spline used in [66] are explained. Furthermore, the computation of the tension parameters is mentioned explicitly. In the context of this thesis, exponential smoothing splines are needed for the determination of the phase transition in lattice QCD using the Wilson flow. They are applied in the energy difference method [66] described in section 8.3.

This section opens with the idea of exponential smoothing splines denoted as minimization problem. Then, the formula (6.8) for the spline is stated in section 6.3 accompanied by the linear equations which lead to the coefficients of the spline. Paragraph 6.4 is dedicate to the computation of the Lagrange parameter of the minimization problem. Then, two examples are given in section 6.5, one of artificial data and one for the detection of the critical temperature in finite temperature $SU(3,\mathbb{C})$ Yang-Mills theory. Finally, this chapter closes with a summary in part 6.6.

6.1 Minimization Problem

Let there be n data points (x_i, y_i), with uncertainties w_i of y_i and $i = 1, \ldots, n$. The data should be approximated within the region of their errors and oscillations should be avoided at the same time. The exponential smoothing spline is developed for this purpose. It can be found by minimizing the energy function

$$\int_{x_0}^{x_n} \left[f''(x)^2 + \Lambda(x)^2 f'(x)^2 \right] dx \tag{6.1a}$$

among all $f \in C^2(x_0, x_n)$ using the constraints

$$\sum_{i=0}^{n} \left(\frac{f(x_i) - y_i}{w_i} \right)^2 \leq S \quad \text{and} \quad \Lambda(x) = \lambda_i > 0 \tag{6.1b}$$

with $x \in [x_i, x_{i+1})$ for $i = 0, \ldots, n-1$. Here, the piecewise constant tension functions $\Lambda(x)$ avoid undesired oscillations, the weights w_i are correlated to the errors of y_i and the smoothing parameter S defines how much the approximated values $f(x_i)$ may differ from y_i.

The minimization problem (6.1) is solved by building a functional J combining the minimization problem and the constraint. Afterwards, it is minimized with respect to its parameters. For this purpose, a slack variable z is introduced which turns equation (6.1b) in an equation:

$$\sum_{i=0}^{n} \left(\frac{f(x_i) - y_i}{w_i} \right)^2 = S - z^2. \tag{6.2}$$

Then, a functional $J(\epsilon, p, z)$ is composed of the minimization problem (6.1a) and the constraint (6.2) using a Lagrange parameter p. In addition, the calculus of variation is used to replace $f(x)$ by

$$f(x) = s(x) + \epsilon h(x) \quad \text{with} \quad 0 \leq \epsilon << 1 \qquad \text{and} \quad h(x_0) = 0 = h(x_n). \tag{6.3}$$

All in all, the functional reads

$$\begin{aligned} J(\epsilon, p, z) := & \int_{x_0}^{x_n} \left[\Big(s''(x) + \epsilon h''(x) \Big)^2 + \Lambda(x)^2 \Big(s'(x) + \epsilon h'(x) \Big)^2 \right] dx \\ & + p \cdot \left\{ \sum_{i=0}^{n} \Big(\frac{s(x_i) + \epsilon h(x_i) - y_i}{w_i} \Big)^2 - S + z^2 \right\} \end{aligned} \tag{6.4}$$

which has to be minimized with respect to its three parameters ϵ, p, z and evaluated at $\epsilon = 0$. A minimization of $\partial J(\epsilon, p, z)/\partial \epsilon$ leads to the shape of the exponential smoothing spline.

6.2 Relation to Other Splines

The exponential smoothing spline is a generalization of already known splines. Concerning the minimization problem (6.1) with

- $\Lambda(x) = 0, S \neq 0$, oscillations are allowed leading to the smoothing splines

$$s(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{6.5}$$

for $x \in [x_i, x_{i+1})$, $i = 0, \dots, n-1$ developed in [52],

- $S = 0, \Lambda(x) \neq 0$, the data are met exactly, but the oscillations are suppressed by the tension functions $\Lambda(x)$ as described, e.g, in [13], [51], [57], [59], [53] or in [54]. This leads to the exponential splines

$$\begin{aligned} s(x) = y_{i+1}t + y_i(1-t) + \frac{d_{i+1}}{\lambda_i^2}\left(\frac{\sinh(\mu_i t)}{\sinh(\mu_i)} - t\right) \\ + \frac{d_i}{\lambda_i^2}\left(\frac{\sinh(\mu_i(1-t))}{\sinh(\mu_i)} - 1 + t\right) \end{aligned} \qquad (6.6)$$

for $x \in [x_i, x_{i+1})$, $i = 0, \dots, n-1$ and

$$t = \frac{x - x_i}{x_{i+1} - x_i}, \quad \mu_i := \lambda_i \cdot h_i, \quad h_i := x_{i+1} - x_i, \qquad (6.7)$$

- $\Lambda(x) = 0$ and $S = 0$, the famous cubic splines are met.

All splines are constructed as the function $f(x)$ among all functions $f(x) \in C^2[x_0, x_n]$ in order that the integrals are minimized while fulfilling the constraints.

6.3 The Exponential Smoothing Spline and its Coefficients

The minimization problem (6.1) is solved by the exponential smoothing spline

$$s(x) = s_{i+1}t + s_i(1-t) + \frac{d_{i+1}}{\lambda_i^2}\left(\frac{\sinh(\mu_i t)}{\sinh(\mu_i)} - t\right) + \frac{d_i}{\lambda_i^2}\left(\frac{\sinh(\mu_i(1-t))}{\sinh(\mu_i)} - 1 + t\right). \qquad (6.8)$$

It has the same shape as the exponential spline. The only difference to the parameterization of the exponential spline is that the data y_i are replaced by the unknown $s_i := s(x_i)$ since it holds $s(x_i) \neq y_i$ in general. Concerning formula (6.8), the values λ_i, μ_i and t are given by equations (6.7) for each interval $[x_i, x_{i+1})$ for $i = 0, \dots, n-1$. The tension parameters λ_i are also pre-determined for $i = 0, \dots, n-1$. According to [53], the tension parameters λ_i can be chosen as uniformly distributed random values in the interval $[4h_i, 15h_i]$. This is just one - the most simplest - of several possibilities to determine the tension parameters, see [53].

The parameters s_i, s_{i+1}, d_i, d_{i+1}, $i = 0, \dots, n-1$, are the unknowns, i.e $4n$ equations are needed for its calculation. They can be determined using the smoothing conditions

$$\begin{aligned} s(x_i^-) - s(x_i^+) &= 0, & i &= 1, \ldots, n-1, \qquad (6.9a)\\ s'(x_i^-) - s'(x_i^+) &= 0, & i &= 1, \ldots, n-1, \qquad (6.9b)\\ s''(x_i^-) - s''(x_i^+) &= 0, & i &= 0, \ldots, n, \qquad (6.9c) \end{aligned}$$

and the jump equation

$$\Big(s'''(x_i^-) - \Lambda^2(x_i^-)s'(x_i^-)\Big) - \Big(s'''(x_i^+) - \Lambda^2(x_i^+)s'(x_i^+)\Big) = 2p\frac{s(x_i) - y_i}{w_i^2} \qquad (6.9d)$$

for $i = 0, \ldots, n$. These equations imply that the splines of the subintervals $[x_{i-1}, x_i)$ and $[x_i, x_{i+1})$ are turned smoothly into each other. In equation (6.9), the limits

$$s^{(k)}(x_i^\pm) = \lim_{h \to 0, h>0}(x_i \pm h), \quad \text{for} \quad k = 0,1,2,3, \quad i = 0, \ldots, n \qquad (6.10)$$

and natural boundary conditions $s''(x_0) = s''(x_n) = 0$ are used for the k-th derivative of the function s.

The smoothing condition (6.9b) and the jump equation (6.9d) lead to two linear equations determining the unknowns $s = (s_0, \ldots, s_n)^\top$ and $d = (d_1, \ldots, d_{n-1})^\top$. The values $d_0 = d_n = 0$ are already given through the natural boundary conditions $s''(x_0) = s''(x_n) = 0$.

The first linear equation

$$Qs = Td \qquad (6.11)$$

depends on the smoothing condition (6.9b) and contains sparse matrices Q and T. The matrix Q is a tridiagonal matrix with $n-1$ rows and $n+1$ colums with entries

$$Q_{i,i} = -\frac{1}{h_{i-1}}, \qquad Q_{i,i+1} = \frac{1}{h_{i-1}} + \frac{1}{h_i}, \qquad Q_{i,i+2} = -\frac{1}{h_i}. \qquad (6.12)$$

T a symmetric tridiagonal matrix of order $n-1$ with entries

$$\begin{aligned} T_{i,i} &= t_{i-1} + t_i, & t_i &:= -\frac{\cosh(\mu_i)}{\lambda_i \sinh(\mu_i)} + \frac{1}{\lambda_i^2 h_i} \qquad (6.13a)\\ T_{k,k+1} = T_{k+1,k} &= \tilde{t}_k, & \tilde{t}_k &:= \frac{1}{\lambda_k \sinh(\mu_k)} - \frac{1}{\lambda_k^2 h_k}. \qquad (6.13b) \end{aligned}$$

The indices i and k run from $i = 1, \ldots, n-1$ and $k = 1, \ldots, n-2$.

The second linear equation

$$Us - Q^T d = pD^{-2}(s - y) \qquad (6.14)$$

with diagonal matrix $D := \mathrm{diag}(w_0, \ldots, w_n)^T$ and symmetric tridiagonal matrix U

of size $(n+1) \times (n+1)$ arises from the jump equation (6.9d). U is given by

$$U_{1,1} := -\frac{\lambda_0^2}{h_0}, \qquad U_{n+1,n+1} := -\frac{\lambda_n^2}{h_n}, \tag{6.15a}$$

$$U_{i,i} := -\left(\frac{\lambda_{i-1}^2}{h_{i-1}} + \frac{\lambda_i^2}{h_i}\right) \qquad \text{for} \quad i = 2, \dots, n, \tag{6.15b}$$

$$\text{and} \quad U_{j,j+1} = U_{j+1,j} := \frac{\lambda_j^2}{h_j} \qquad \text{for} \quad j = 1, \dots, n. \tag{6.15c}$$

The details of the derivation of these two linear equations (6.11) and (6.14) can be found in the appendix, see section .3.

6.4 Lagrange Parameter

The Lagrange parameter can be evolved by the minimization of the functional $J(\epsilon, p, z)$ given in equation (6.4) with respect to p and z:

$$\frac{\partial J(\epsilon,p,z)}{\partial p}|_{\epsilon=0} = \sum_{i=0}^{n}\Big(\frac{s(x_i)-y_i}{w_i}\Big)^2 - S + z^2 \qquad \frac{\partial J(\epsilon,p,z)}{\partial z}|_{\epsilon=0} = 2pz \tag{6.16}$$

The minimization leads to

$$S - z^2 = \sum_{i=0}^{n}\left(\frac{s(x_i)-y_i}{w_i}\right)^2 = ||D^{-1}(s-y)||_2^2 =: F(p)^2 \ .$$

Using

$$s = -p(U - Q^T T^{-1} Q - pD^{-2})^{-1} D^{-2} y$$

computed by a combination of the aforementioned linear equations (6.11), the Lagrange parameter can be computed by $F(p)^2 = S - z^2$ with

$$F(p) = \|D^{-1}(s-y)\|_2 = ||pD^{-1}\big((U - Q^T T^{-1} Q - pD^{-2})^{-1} D^{-2} y - y\big)||_2.$$

The derivative of the functional $J(\epsilon, p, z)$ with respect to z leads to $z = 0$ in order that $F(p)^2 - S = 0$ has to be solved.

Finally, the Lagrange parameter p is still unknown and can be computed by $F(p)^2 = S$ with

$$F(p) = ||pD^{-1}(U - Q^T T^{-1} Q - pD^{-2})^{-1} D^{-2} y - pD^{-2} y||_2 \ . \tag{6.17}$$

This equation has to be solved for the unknown p, for example, with interval nesting or a Newton iteration. Thereby, it must be taken into account that the starting value $p^{(0)} = 0$ should be avoided since it would lead to vectors $s = 0, d = 0$ and therefore to a spline $s(x) = 0$.

6.5 Example

The exponential smoothing spline can be used to approximate error-prone data in many fields of applications. In this section, we show the advantage of the exponential smoothing spline, respective smoothing spline under tension, for an academic example with random data and weights.

In this example, we created random data y_i with random weights w_i, $i = 1, \dots, 14$, both uniformly distributed. The values are given in table 6.1 and shown in figure 6.1. Additionally, the exponential smoothing spline is drawn and compared to the smoothing spline and the exponential spline. We see that the smoothing spline approximates the data and the exponential spline interpolates them. The advantage of the exponential smoothing spline is the approximation the data and at the same time the suppression of oscillations not necessarily given in the data.

x	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7
y	0.26	0.27	0.46	0.47	0.52	0.64	0.59	0.55	0.61	0.42	0.45	0.39	0.44	0.62
w	0.06	0.12	0.06	0.07	0.01	0.10	0.07	0.07	0.04	0.04	0.05	0.08	0.11	0.06

Table 6.1: Values of figure 6.1.

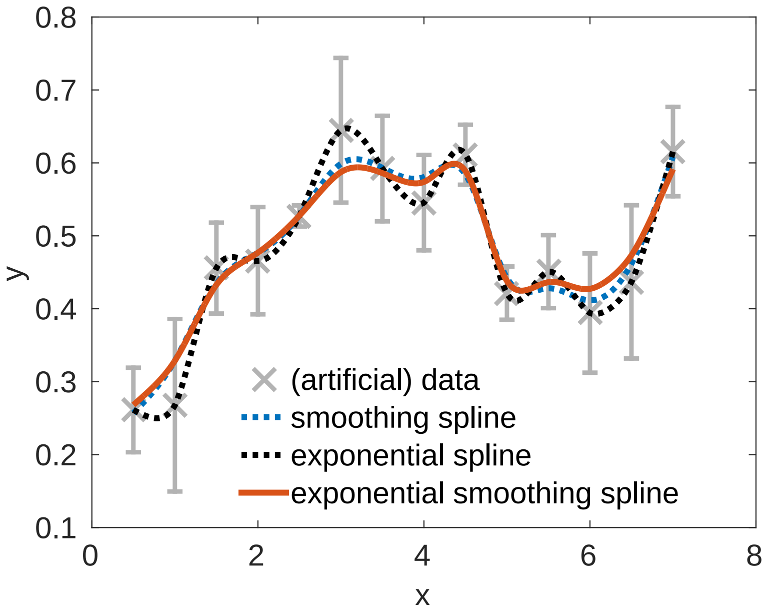

Figure 6.1: *Example of an exponential smoothing spline.* – Academic example with random data and random weights.

6.6 Summary

In this chapter, the exponential smoothing spline is introduced following the line of section 2.3 in [68]. It has the advantage that data with statistical or measurement errors can be approximated without oscillations not being part of the data. The content of section 6.1, 6.3 and 6.4 is mainly taken from [68]. Here, the computation of the Lagrange parameter is changed. Furthermore, the computation of the tension parameters according to Rentrop [53] is explained in detail. However, the investigation of the best approximation described in [68] is left out here. The exponential smoothing spline is computed for an academic example with random data. This serves as clarification of the shapes of the exponential smoothing spline and the related exponential and smoothing spline.

In chapter 8, the exponential smoothing spline is used in the context of finite temperature QCD. This second example is taken from [66]. Here, a method for the detection of the finite temperature phase transition in SU(N) Yang-Mills theory is developed based on the usage of this spline.

Part III

Simulation

Chapter 7

7 HMC in Lattice QCD

One of the main contents of this thesis is the development of geometric numerical integration methods for Lie groups and its application in Lattice QCD. In this chapter, the Hamiltonian equations of motion occurring in the Hybrid Monte Carlo simulation are implemented using the different geometric integration schemes explained in chapter 5: the Cayley-Leapfrog method, the symmetric partitioned Munthe-Kaas Runge-Kutta method and the projection scheme. For this purpose, lattice gauge fields in $SU(N)$ Yang-Mills theory are simulated.

This chapter is split into two parts. We start with the description of Hybrid Monte Carlo simulations on lattice gauge fields in section 7.1 followed by a section 7.2 on the simulations of lattice gauge field using the different geometric numerical integration methods described in chapter 5.

7.1 HMC Simulations on Lattice Gauge Fields

We simulate the Wilson action S_G of lattice gauge fields in $SU(N)$ Yang-Mills theory (which is described in paragraph 2.1) in the context of a HMC simulation. Thereby, the model consists of a field of links $[U]$ and a related field of fictitious momenta $[P]$. The links $U_{x\mu}$ belong to a special unitary Lie group $SU(N, \mathbb{C})$ with $N = 2$ or $N = 3$ depending on the model. The momenta $P_{x\mu}$ are traceless and hermitian and have the same size $N \times N$ as the link matrices. The lattice has two dimensions in order that its size is $T \times L$ with time extension T and space extension L.This means, we simulate $n_l = 2 \cdot T \cdot L$ link matrices situated at the interconnecting lines between two grid points. At each position of the grid, there exist 2 links and its corresponding momenta. The details of the HMC simulation are obtained via personal communication with Prof. Dr. F. Knechtli unless noted otherwise.

7.1.1 HMC

The Wilson gauge action is simulated by means of the Hybrid Monte Carlo method which works as follows:

1. Start with a gauge field of links $[U]_i$.

2. Draw a field of Gaussian distributed momenta $[P]_i$.

3. Compute the Hamiltonian $\mathcal{H}([U]_i, [P]_i)$.
4. Perform a trajectory of Molecular Dynamics Steps using a geometric integration scheme Φ:
$$([U]_i, [P]_i) \to ([U]_j, [P]_j) = \Phi([U]_i, [P]_i) \tag{7.1}$$
5. Compute the Hamiltonian $\mathcal{H}([U]_j, [P]_j)$.
6. Perform an acceptance step. Add the new field $[U]_{i+1}$ to the ensemble of field configurations.
7. Proceed with step 2.

For the execution of the single steps of the simulation, some further information is needed.

Representation of the Matrices for Links and Momenta. First of all, the gauge field of links should be uniformly distributed and the momenta are drawn from Gaussian distributed random numbers. In this thesis, we work with special unitary Lie groups of dimension $N = 2$ and $N = 3$. Special unitary complex Lie group and Lie algebra matrices of dimension $N = 2$ can be represented by the Pauli matrices

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{7.2}$$

These matrices are traceless and hermitian with the result that it holds $\sigma_j = \sigma_j^\dagger$ for $j = 1, 2, 3$ as described, for example in [34].

Lemma 7.1. Every matrix $U \in SU(2, \mathbb{C})$ can be represented by

$$U = \sum_{j=1}^{3} x_j i \sigma_j + x_4 \cdot I_2 = \begin{pmatrix} x_4 + ix_3, & x_2 + ix_1 \\ -x_2 + ix_1, & x_4 - ix_3 \end{pmatrix} \tag{7.3}$$

with complex i and a vector $x \in \mathbb{R}^4$ and $||x||_2 = 1$. $I_2 \in \mathbb{R}^{2\times 2}$ is the identity matrix and σ_1, σ_2 and σ_3 are the Pauli matrices.

Lemma 7.2. The matrices $A \in \mathfrak{su}(2, \mathbb{C})$ are traceless and anti-hermitian, in order that they can be build from a linear combination of the Pauli matrices σ_1, σ_2, σ_3 as

$$A = i \sum_{j=1}^{3} y_j \sigma_j \tag{7.4}$$

with $y_j \in \mathbb{R}, j = 1, 2, 3$ and complex i. [34]

The conjugated momenta of the links have to be Gaussian distributed with the result that the fixed point of the Markov chain gained via the HMC method can be reached. Thus, the momenta are chosen as

$$P = \frac{1}{\sqrt{2}} \sum_{j=1}^{3} y_j \sigma_j = \frac{1}{\sqrt{2}} \begin{pmatrix} y_3 & y_1 - iy_2 \\ y_1 + iy_2 & -y_3 \end{pmatrix}. \tag{7.5}$$

The elements U of the Lie group $SU(3, \mathbb{C})$ can be build from elements of its associated Lie algebra $\mathfrak{su}(3, \mathbb{C})$ via the exponential map. The generators of the Lie algebra $\mathfrak{su}(3, \mathbb{C})$ are the eight Gell-Mann matrices λ_j, $j = 1, \ldots, 8$ in order that it holds

$$A = i \sum_{j=1}^{8} y_j \lambda_j \in \mathfrak{su}(3, \mathbb{C}) \tag{7.6}$$

with $y_j \in \mathbb{R}$, $j = 1, \ldots, 8$ as described, for example, in [21].

2-Dimensional Gauge Field of $SU(N, \mathbb{C})$ and its Hamiltonian. For the test of geometric numerical integration schemes for the Molecular Dynamics step, 2-dimensional lattice gauge fields with links being elements of the special unitary Lie group $SU(2, \mathbb{C})$ or $SU(3, \mathbb{C})$ and periodic boundary conditions are used. In 2 dimensions, the Lorentz indices μ and ν take the values $\mu = 0$ and $\nu = 1$. The Hamiltonian (7.22) can be adapted to

$$\mathcal{H}([U], [P]) = \frac{1}{2} \sum_{x,\mu} \mathrm{Tr}\left(P_{x,\mu}^2\right) + \sum_{x} \sum_{\mu<\nu} \beta \left(1 - \frac{1}{N} Re\left(Tr(\mathcal{P}_{x,\mu\nu})\right)\right) \tag{7.7}$$

and N is set to $N = 2$ or $N = 3$. This means, the kinetic energy is computed from all momenta and the Wilson gauge action from all anti-clockwise oriented plaquettes. The sum $\sum_{\mu<\nu}$ could be left out because there exists just the single $(\mu = 0, \nu = 1)$-plane as shown in figure 2.3.

Molecular Dynamics Step. The heart of the HMC algorithm is the geometric integration of the Hamiltonian equations of motion and the computation of the Hamiltonians for the acceptance step. The geometric numerical integration schemes developed during this thesis are intended to be used in Lattice QCD. Thus, they are applied on the Hamiltonian equations of motion (2.13) which read

$$\frac{\partial \mathcal{H}}{\partial P_{x,\mu}} = \dot{U}_{x,\mu} = iP_{x,\mu} U_{x,\mu}, \qquad \frac{\partial \mathcal{H}}{\partial U_{x,\mu}} = -\dot{P}_{x,\mu} = -i\frac{\beta}{N} \Big\{ U_{x,\mu} \mathcal{S}(U_{x,\mu}) \Big\}_{TA}.$$

with traceless anti-hermitian operator

$$\Big\{ M_{x,\mu} \Big\}_{TA} = \Big(M_{x,\mu} - M_{x,\mu}^{\dagger} \Big) - \frac{1}{N} \mathrm{Tr}\Big(M_{x,\mu} - M_{x,\mu}^{\dagger} \Big) \cdot I_2 \tag{7.8}$$

and staples

$$\mathcal{S}(U_{x,\mu}) = U_{x+a\hat{\mu},\nu} U^{\dagger}_{x+a\hat{\nu},\mu} U^{\dagger}_{x,\nu} + U^{\dagger}_{x+a(\hat{\mu}-\hat{\nu}),\nu} U^{\dagger}_{x-a\hat{\nu},\mu} U_{x-a\hat{\nu},\nu}. \tag{7.9}$$

N is the dimension of the matrix (usually $N = 3$ or even $N = 2$) and β the coupling constant. For the simulations in this chapter, β is set to $\beta = 2.0$. The identification of the correct links inside the staples is realized via a lexicographical index inside the code.

Acceptance Step. The new configuration $[U]_j$ is accepted with transition probability $\min(1, \exp(-\Delta\mathcal{H}))$ and $\Delta\mathcal{H} = \mathcal{H}([U]_j, [P]_j) - \mathcal{H}([U]_i, [P]_i)$. This means, it is accepted if the new Hamiltonian is smaller than the old one. Otherwise, a uniformly distributed random number $r \in [0, 1]$ is drawn. If r is smaller than $\exp(-\Delta\mathcal{H})$, the new configuration is accepted, otherwise the old one. In any case, the more probable and thus accepted configuration is added to the Markov chain of HMC configurations. It is the origin for the next step with a new Gaussian distributed field of momenta.

Thermalization. The HMC starts from a random configuration which might not be close to the target distribution. So, it may take a while to get in the region of the target distribution. This time is called thermalization phase. Usually, the thermalization phase is left out for the computation of the expectation values and thus, the expectation values are computed from configurations after the thermalization.

Statistical Errors. Mean values $\langle\mathcal{A}\rangle$ of Monte Carlo simulations go along with statistical errors including auto-correlation effects. Statistical errors are computed as

$$\sigma^2 = \frac{var(A)}{N/2\tau_{int,A}} \tag{7.10}$$

with variance $var(A)$ and integrated auto-correlation time

$$\tau_{int,A} = \frac{1}{2} + \sum_{t=1}^{\infty} \frac{\Gamma(t)}{\Gamma(0)}. \tag{7.11}$$

The auto-correlation $\Gamma(t)$ is a measure for the independence of subsequent configurations inside a Monte Carlo simulation. For a time series of N measurements A_i from a Markov process, the auto-correlation is defined as

$$\Gamma(t) = \Gamma_{ij} = \Big\langle \Big(A_i - \langle A_i\rangle\Big)\Big(A_j - \langle A_j\rangle\Big)\Big\rangle \tag{7.12}$$

with $t = |i - j|$. So, Γ_{ij} does not depend on the indices i and j itself but on its distance t. Furthermore, $\langle A_i\rangle = \langle A_j\rangle =: a$ Thus, $\Gamma(t)$ can be expressed as

$$\Gamma(t) = \Big\langle \Big(A_i - a\Big)\Big(A_{i+t} - a\Big)\Big\rangle \tag{7.13}$$

The auto-correlation is defined, for example, in [21], [34] or [69]. In this thesis, the statistical errors including auto-correlation effects are computed using the method of Wolff explained in [69].

Validation. The model should be validated. For this purpose, the model is tested for gauge invariance and the value of its plaquette is cross-checked with other existing programs. Starting from a field of links settled at the interconnecting lines of a grid, the plaquette value can be computed via

$$\mathcal{P}([U]) = \sum_{x} \sum_{\mu<\nu} Re\Big(Tr\big(\mathcal{P}_{x,\mu\nu}([U])\big)\Big) \tag{7.14}$$

with

$$\mathcal{P}_{x,\mu\nu}([U]) := U_{x,\mu} U_{x+a\hat{\mu},\nu} U^{\dagger}_{x+a\hat{\nu},\mu} U^{\dagger}_{x,\nu} \ .$$

given in equation (2.1).

Another validation is the check of gauge invariance which can be done as proposed in paragraph 2.1.2. Here, random special unitary matrices $V_x \in SU(N,\mathbb{C})$ are put at each grid point with the result that a gauge transformation

$$U_{x,\mu} \to \tilde{U}_{x,\mu} = V_x U_{x,\mu} V^{\dagger}_{x+a\hat{\mu}}.$$

given in equation (2.4) changes each link $U_{x,\mu}$ to $\tilde{U}_{x,\mu}$. Finally, the plaquette values $\mathcal{P}([\tilde{U}])$ and $\mathcal{P}([U])$ defined in equation (7.14) must have exactly the same values (up to rounding errors in the region of machine precision).

7.1.2 Tests for Geometric Integration

Inside the Hybrid Monte Carlo simulation, the Hamiltonian equation of motion are solved with a geometric integration scheme in the Molecular Dynamics part.

The quality of the numerical integration scheme is tested by means of the convergence order, its computational cost and a check of its geometric properties. Furthermore, the model itself is validated due to gauge invariance and the mean plaquette value as described above..

Time-Reversibility. For the geometric integration scheme used inside the HMC method, it is absolutely necessary that the scheme Φ_h is time-reversible, i.e.

$$\rho \circ \Phi_h \circ \rho \circ \Phi_h = id \quad \text{with} \quad \rho = \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix} \tag{7.15}$$

(provided that Φ_h is symmetric, i.e. $\Phi_h = \Phi_{-h}^{-1}$) stated in equation (5.14) in definition 5.11.

This is checked numerically as follows starting from an initial configuration $([U]_i, [P]_i)$:

1. Compute $\Phi_h([U]_i, [P]_i) = ([U]_{i+1}, [P]_{i+1})$,
2. then put a minus in front of $[P]_{i+1}$,
3. compute $\Phi_h([U]_{i+1}, -[P]_{i+1}) =: ([U]_j, -[P]_j)$
4. and flip the sign of the momenta $[P]_j$ again.

Finally, compare $([U]_j, [P]_j)$ with $([U]_i, [P]_i)$. The fields should be the same up to machine precision.

Volume-Preservation. The numerical method is volume-preserving if the absolute value of the determinant of its Jacobian is equal to one:

$$\left|\frac{\partial\Phi([U]_i, [P]_i)}{\partial([U]_i, [P]_i)}\right| = 1. \tag{7.16}$$

Theoretically, this can be overcome by a multiplication of the inverse of the Jacobian. Due to the fact that the system is very large, this theoretical possibility is not feasible in practice.

Since there is no general analytical formula for the computation of the Jacobian of the geometric integration of a lattice gauge field, the Jacobian is computed from difference formulae. These formulae depend on the structure of $([U]_i, [P]_i)$, respective on its implementation inside the code. Thus, the numerical investigation of the volume-preservation will be discussed separately for each model in the following paragraphs.

Convergence Order. The convergence order can be checked via the difference of the Hamiltonian of the field configurations before and after the Molecular Dynamics step. More precisely, the mean value of the absolute difference of the old Hamiltonian and the new one after the geometric integration and is computed by

$$\Delta\mathcal{H} := \frac{1}{n}\sum_{i=1}^{n}\left|\mathcal{H}([U]_i, [P]_i) - \mathcal{H}\Big(\phi([U]_i, [P]_i)\Big)\right| . \tag{7.17}$$

$\Delta\mathcal{H}$ is the difference between two Hamiltonians computed before and after a whole trajectory of numerical integration steps. The length of this trajectory is τ and its value is set to $\tau = 1$ unless it is mentioned that τ has a different value. For a numerical integration scheme of convergence order p, a local error of size $\mathcal{O}(h^{p+1})$ is expected, with the result that $\Delta\mathcal{H}$ would have the value $\Delta\mathcal{H} = c \times h^{p+1}$ with constant c. Due to the fact that the determination of $\Delta\mathcal{H}$, requires the computation of a whole trajectory of length τ, $n = \tau/h$ steps are computed in order that the energy difference $\Delta\mathcal{H}$ has the magnitude

$$\Delta\mathcal{H} = c \cdot h^{p+1} \cdot n = c \cdot \tau \cdot h^{p} \tag{7.18}$$

is expected as the result of the simulations. Finally, the mean value $\langle\Delta\mathcal{H}\rangle$ is taken for the investigation. For this purpose, the statistical errors including autocorrelation effects are also considered.

The convergence check starts from a certain configuration after the thermalization. From this configuration, all values of $\Delta\mathcal{H}$ are collected, regardless if the new configuration is accepted or not.

Computational Cost. For the computational cost, the computational time is measured. Since the numerical integration is performed on lattices of different size, the computational time is scaled for comparison reasons. It is the time per trajectory of length $\tau = 1$ per link, i.e. the time is divided by $d \cdot T \cdot L^{d-1}$ with $d = 2$. The computational cost is related to the step size of the applied integration scheme or the even more relevant difference in the Hamiltonian. It strongly depends on the implementation of the different schemes which is described in the next paragraph.

Implementation. The Runge-Kutta schemes are implemented in Matlab for 2-dimensional lattice gauge fields in $SU(2,\mathbb{C})$. According to equations (7.3) and (7.5), the matrices of the Lie group and the Lie algebra have the shape

$$Y = \sum_{j=1}^{3} u_j \sigma_j + u_4 I_2 \quad \text{and} \quad A = i \sum_{j=1}^{3} a_j \sigma_j \tag{7.19}$$

with a_1, a_2, a_3, u_1, u_2, u_3 and $u_4 \in \mathbb{R}$, Pauli matrices σ_1, σ_2 and σ_3 and identity matrix $I_2 \in \mathbb{R}^{2\times 2}$. The values u_j, $j = 1, 2, 3, 4$ are drawn as uniformly distributed random numbers with $u_j \in [-1, 1]$. Afterwards, the vector u is normalized by a division of $||u||_2$ [34]. The random numbers a_j $(j = 1, 2, 3)$ are Gaussian distributed (with mean 0 and variance 1).

The matrices Y and A are represented by its sets of coefficients (u_1, u_2, u_3, u_4) and (a_1, a_2, a_3) in order that the code is quite efficient. Just the exponential function and the Cayley transform require the computation of matrices. So, the implementation the exponential function and the Cayley transform are of importance. In this work, the matrix exponential $\exp(A)$ is computed via the matrix decomposition method described by Moler and Van Loan in [41] based on a similarity transformation

$$A = UDU^{-1} \qquad \text{followed by} \qquad \exp(A) = U\exp(D)U^{-1}. \tag{7.20}$$

Concerning the Cayley map $\mathrm{cay}(A) = (I - A)^{-1}(I + A)$, the matrix

$$B = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad \text{leads to its inverse} \quad B^{-1} = \frac{1}{\det(B)} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}. \tag{7.21}$$

Afterwards, the matrices are stored again as set of coefficients.

Notation. For the geometric integration schemes, the equations of motion (2.13) are denoted as

$$\dot{Y}_j(t) = A_j(t)Y_j(t), \qquad \dot{A}_j(t) = F([Y_j(t)]_s) \quad \text{for} \quad j = 0, \ldots, n_l - 1 \tag{7.22}$$

similar to equation (2.58) in section 2.2. For simplicity,

$$[Y] := \{Y_0, \ldots, Y_{n_l-1}\} \quad \text{and} \quad [A] := \{A_0, \ldots, A_{n_l-1}\} \tag{7.23}$$

describe the set of all Lie group, respective Lie algebra elements of one configuration.The initial values are denoted as

$$[Y_n] := \{Y_{n,0}, \ldots, Y_{n,n_l-1}\} \quad \text{and} \quad [A_n] := \{A_{n,0}, \ldots, A_{n,n_l-1}\}\ . \tag{7.24}$$

Thus, one step of a numerical integration scheme can be denoted as

$$\Big([Y_n], [A_n]\Big) \to \Phi_h\Big([Y_n], [A_n]\Big) = \Big([Y_{n+1}], [A_{n+1}]\Big). \tag{7.25}$$

Multiplying a field with a constant $a \in \mathbb{R}$ means that all components are multiplied with this value in order that

$$a[A] = [aA]. \tag{7.26}$$

Similarly, the sum of two fields means that the components are summed component by component:

$$[A] + [B] = [A + B]. \tag{7.27}$$

Inside the code, the Hamiltonian equations of motion are solved according to equation (2.13). Thus, the one-to-one correspondence (2.56), i.e.,

$$U_{x,\mu} \leftrightarrow Y_j \quad \text{and} \quad iP_{x,\mu} \leftrightarrow A_j\ ,$$

depending on the lexicographic index (2.57a), i.e.,

$$j := n_p \cdot \mu + x \quad \text{with} \quad n_p = T \cdot L^{d-1}\ ,$$

is used for a change between the different notations.

7.2 Geometric Integration Methods

In this section, the Hamiltonian equations of motion are solved with different numerical integration schemes for a 2-dimensional gauge field of $SU(2,\mathbb{C})$ or $SU(3,\mathbb{C})$ matrices. More precisely, the geometric numerical integration methods developed in chapter 5 are adapted for a lattice gauge field. Thus, the methods are described for a grid of coupled differential equations on Lie groups / Lie algebras

$$\dot{Y}_j(t) = A_j(t)Y_j(t), \qquad \dot{A}_j(t) = F([Y_j(t)]_s) \quad \text{for} \quad j = 0,\dots,n_l - 1$$

as described in equation (7.22). Here, we consider

- the Leapfrog method based on the exponential function as well as
- the Leapfrog method using the the Cayley mapping,
- the symmetric partitioned Runge-Kutta method and
- the time-reversible projection method.

In general, the Leapfrog method uses the exponential function as local parameterization. This method is known as time-reversible and volume-preserving in order that its numerical solution is taken as reference for the other methods. Next, all four methods for coupled differential equations on Lie groups / Lie algebras are adjusted to the Hamiltonian equations of motion in Lattice gauge theory. Furthermore, the three kinds of Runge-Kutta methods are implemented for 2-dimensional lattice gauge fields in $SU(2,\mathbb{C})$ Yang-Mills theory and the projection method in $SU(3,\mathbb{C})$ Yang-Mills theory. For a first insight in the geometric properties of the schemes, the size of the matrices does not matter. The convergence order, the computational cost as well as the geometric properties time-reversibility and volume-preservation are checked as described in paragraph 7.1.2 and shown in figures 7.1 to 7.8. Here, a lattice of size 8 × 8 is used in order that the number of link matrices reads $n_l = 2 \cdot 8 \cdot 8 = 128$.

7.2.1 Leapfrog Method

The Leapfrog or Störmer-Verlet Lie group method is known as geometric integration scheme on Lie groups with convergence order $p = 2$. The Leapfrog scheme (5.31a) adapted for the Hamiltonian equations of motion (7.22) of a lattice gauge field is given by

$$\begin{aligned}
A_{n+\frac{1}{2},j} &= A_{n,j} + \frac{h}{2}F\big([Y_{n,j}]_s\big), && j = 0,\dots,n_l - 1,\\
Y_{n+1,j} &= \exp\big(hA_{n+\frac{1}{2},j}\big)\, Y_{n,j}, && j = 0,\dots,n_l - 1, \qquad (7.28)\\
A_{n+1,j} &= A_{n+\frac{1}{2},j} + \frac{h}{2}F\big([Y_{n+1,j}]_s\big), && j = 0,\dots,n_l - 1
\end{aligned}$$

with initial values $Y_{n,j} \in SU(2,\mathbb{C})$ and $A_{n,j} \in \mathfrak{su}(2,\mathbb{C})$ for $j = 0,\dots,n_l - 1$. The first index n, $n+\frac{1}{2}$ or $n+1$ denotes the initial values, intermediate steps and the final results of the numerical scheme. The index j refers to the different elements of the field. The Leapfrog scheme is used for a validation of the Hybrid Monte Carlo program through the mean plaquette value by a cross-check with other programs. The computed plaquette values are in good agreement with other programs – the mean plaquette value for $SU(2,\mathbb{C})$ gauge fields should take the value $\langle \operatorname{Tr} \mathcal{P}([U])\rangle = 0.8669(2)$ (personal communication with F. Knechtli). The results of the Leapfrog method concerning convergence, time-reversibility and volume-preservation are shown in figures 7.1 to 7.2. Afterwards, the other developed schemes are compared to these values.

7.2.2 The Cayley-Leapfrog Method

The Cayley-Leapfrog scheme is an adjustment of the Leapfrog scheme 7.28 for Lie group / Lie algebra problems. Here, just the local parameterization is exchanged. The Cayley-Leapfrog scheme for the differential equations (7.22) reads

$$\begin{aligned}
A_{n+\frac{1}{2},j} &= A_{n,j} + \frac{h}{2} F\big([Y_{n,j}]_s\big), && j = 0,\dots,n_l-1,\\
Y_{n+1,j} &= \operatorname{cay}\big(\tfrac{h}{2} A_{n+\frac{1}{2},j}\big)\, Y_{n,j}, && j = 0,\dots,n_l-1,\\
A_{n+1,j} &= A_{n+\frac{1}{2},j} + \frac{h}{2} F\big([Y_{n+1,j}]_s\big), && j = 0,\dots,n_l-1
\end{aligned} \tag{7.29}$$

with initial values $Y_{j,n} \in SU(2,\mathbb{C})$ and $A_{j,n} \in \mathfrak{su}(2,\mathbb{C})$ for $j = 0,\dots,n_l - 1$. This scheme is based on the considerations of section 4.2 and definition 5.15 of subsection 5.2.3.

Convergence. The convergence order of the Cayley-Leapfrog method is shown in figure 7.1. Here, the Cayley-Leapfrog method given in (7.29) is performed where the mean value of the absolute difference of the Hamiltonian before and after one computed trajectory is plotted versus the step size h of the single geometric integration schemes. The Cayley-Leapfrog scheme has convergence order $p = 2$ with the result that an overall error of order $\mathcal{O}(h^2)$ is expected after one trajectory. Indeed, the numerical result of the Cayley-Leapfrog scheme for Lie group/Lie algebra problems has convergence order $p = 2$ as well as the shown numerical result of the (standard) Leapfrog method.

Geometric Properties. The geometric properties which have to be fulfilled are the time-reversibility stated in equation (7.15) and the volume-preservation stated in equation (7.16). On the left-hand side of figure 7.2, the maximum point wise difference of the initial values $[Y_n, A_n]$ and $\rho \circ \Phi \circ \rho \circ \Phi([Y_n, A_n])$ with $\rho([Y,A]) = [Y,-A]$ are shown. It can be seen that these values are in a region close to machine precision for both methods. So, both methods are numerically time-reversible. The

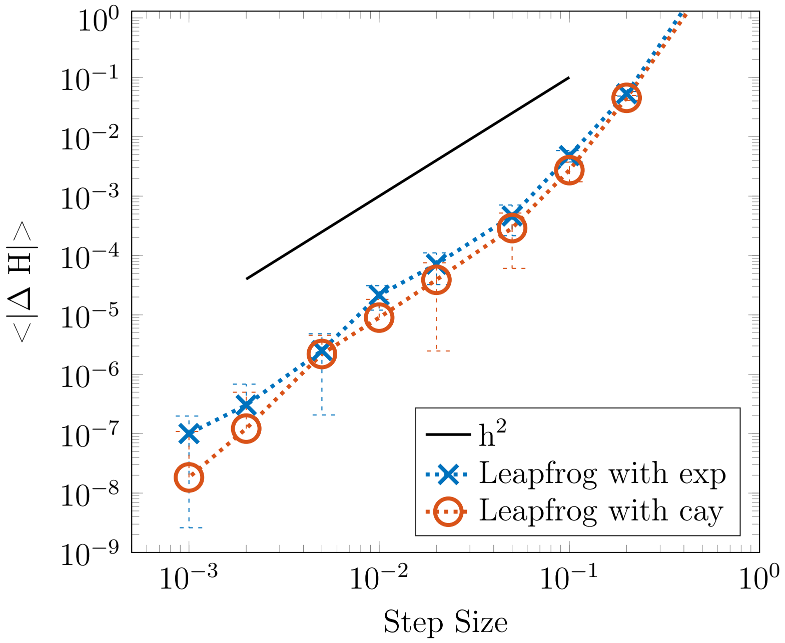

Figure 7.1: *Leapfrog: convergence order* – for the exponential map and the Cayley transform. The Cayley-Leapfrog scheme (red circles) and the standard Leapfrog scheme (blue crosses) both have convergence order $p = 2$.

right-hand side of figure 7.2 shows the value of the determinant of the Jacobian $J = \partial[Y_{n+1}, A_{n+1}]/\partial[Y_n, A_n]$, computed via numerical differentiation. More precise, the coefficients $u_1, u_2, u_3, u_4, a_1, a_2, a_3$ from equation (7.19) for each of the n_l links are collected in a large vector $v \in \mathbb{R}^{7n_l}$. Then, the Jacobian is obtained via the one-sided forward difference operator

$$\frac{f(v + \varepsilon e_i) - f(v)}{\varepsilon} \tag{7.30}$$

with unit vector e_i and $\varepsilon = 10^{-6}$. This means, we choose an initial lattice gauge field and compute one trajectory. Then, we collect the coefficients to obtain $f(v)$. Afterwards, each value of the vector v is disturbed ... each value

This value is close to 1 in order that both methods are volume-preserving.

Computational Cost. The computational cost is illustrated in figure 7.3. Shown is the CPU time for one Molecular Dynamics step ($\tau = 1$, τ/h steps) inside the HMC integration versus the step size h and the difference in the Hamiltonian. Both, the Cayley transform and the exponential function are implemented as described above. In doing so, the Cayley transform is ≈ 4.5 times faster than the exponential function.

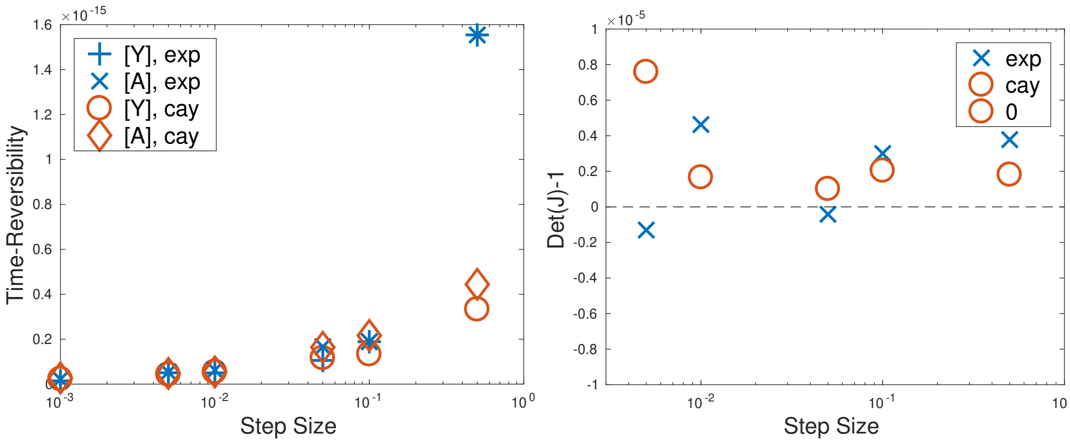

Figure 7.2: *Geometric Properties of the Leapfrog schemes.* – Shown is the time-reversibility (left) and the volume-preservation (right) of the standard Leapfrog scheme (blue crosses and plus signs) and the Cayley-Leapfrog scheme (red circles and diamonds).

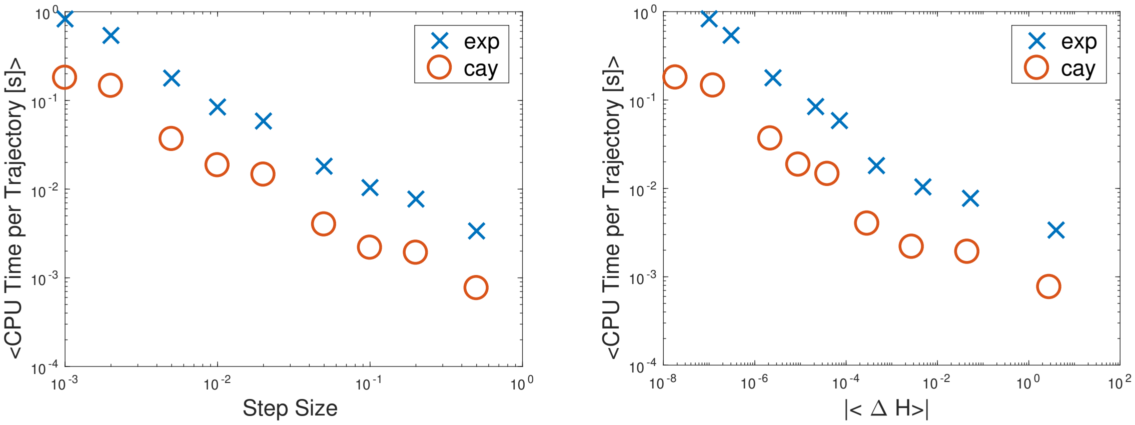

Figure 7.3: *Computational cost of the Leapfrog schemes* – Left: CPU time versus the step size, right: CPU time versus the mean difference in the Hamiltonians before and after one Molecular Dynamics step. The Cayley-Leapfrog scheme (red circles) is less time-consuming than the standard Leapfrog scheme (blue crosses).

Conclusion. It is widely known, that the Leapfrog method is suitable for the numerical solution of the Hamiltonian equations of motion (7.22). Many numerical schemes are based on this method. The Cayley-Leapfrog method exhibits the same numerical results with less computing time. With the aforementioned results, it is shown that the Cayley-Leapfrog method works better than the standard Leapfrog method for lattice gauge fields. Different parameterizations of the Lie group elements are not restricted to the Leapfrog or Störmer-Verlet method. Thus, the Cayley transform can be used in any Lie group method as, for example, the Munthe-Kaas or Crouch-Grossmann Runge-Kutta methods. The only requirements are that the Lie group is a quadratic Lie group.

7.2.3 The Symmetric Partitioned Runge-Kutta Method

The symmetric partitioned Munthe-Kaas method explained in chapter 5.3 can also be adapted to the system of differential equations (7.22) representing the Hamiltonian equations of motion for lattice gauge fields. For each coupled system of Lie group / Lie algebra differential equations, the symmetric partitioned Munthe-Kaas Runge-Kutta method stated in algorithm 5.17 with increments L_i, $i = 1, \dots, s$ formulated according to equation (5.50) leads to a time-reversible result. Starting from a gauge field of initial values $([Y_n], [A_n])$, the outcome $([Y_{n+1}], [A_{n+1}])$ is obtained by the following algorithm.

Algorithm 7.3. (Symmetric partitioned MKRK method for Lattice Gauge Fields)

1. Start with initial values $[Y_n]$,$[A_n]$ and $[\Omega(t_0)] = [\Omega_n] = \{0, \dots, 0\}$.
2. Consider the set of coupled differential equations

$$\begin{aligned}(d\exp_{\Omega_l}^{-1}(A_l) \approx) \quad \dot{\Omega}_l &= \sum_{k\leq 0}^{q} \frac{B_k}{k!} \operatorname{ad}_{\Omega_l}^k(A_l) =: f_q(\Omega_l, A_l), \\ \dot{A}_l &= F([Y_l]_s)\end{aligned}$$

   for $l = 0, \dots, n_l - 1$ in the Lie algebra.

3. Apply a partitioned Runge-Kutta method (explicit or implicit)

$$\Omega_{n+1,l} = h \sum_{i=1}^{s} b_i K_{i,l}, \tag{7.31a}$$

$$A_{n+1,l} = A_{n,l} + h \sum_{i=1}^{s} \hat{b}_i L_{i,l}, \tag{7.31b}$$

with increments

$$K_{i,\,l} = f_q\Big(\bar{\Omega}_{i,\,l}, A(\bar{Y}_{i,\,l})\Big), \tag{7.31c}$$

$$L_{i,\,l} = F\Big(\exp(\bar{\omega}_{i,\,l})\ \exp\big(\tfrac{1}{2}\Omega_{n+1,\,l}\big)\ Y_{n,\,l}\Big) \tag{7.31d}$$

and internal stages

$$\bar{\Omega}_{i,\,l} = h\sum_{j=1}^{s} a_{ij}K_{j,\,l}, \tag{7.31e}$$

$$\bar{Y}_{i,\,l} = \exp\Big(A_{n,\,l} + h\sum_{j=1}^{s} \hat{a}_{ij}L_{j,\,l}\Big), \tag{7.31f}$$

$$\bar{\omega}_{i,\,l} = h\sum_{j=1}^{s} c_{ij}K_{j,\,l}, \tag{7.31g}$$

for $l = 0, \ldots, n_l - 1$. This leads to the approximations

$$\Omega_{n+1,\,l} \approx \Omega_k(t_0 + h) \quad \text{and} \quad A_{n+1,\,l} \approx A_k(t_0 + h). \tag{7.32}$$

4. Define the numerical solution by $\big([Y_{n+1}], [A_{n+1}]\big)$ with

$$Y_{n+1,\,l} = \exp\big(\Omega_{n+1,\,l}\big)\ Y_{n,\,l} \qquad \text{for} \qquad l = 0, \ldots, n_l - 1. \tag{7.33}$$

Due to the shape of the increments $L_{i,l}$ stated in equation (7.31d), the symmetric partitioned Munthe-Kaas Runge-Kutta scheme is an implicit scheme. Thus, all $n_l \cdot s$ increments $K_{i,l}$ and $L_{i,l}$ (for $i = 1, \ldots, s$ and $l = 0, \ldots, n_l - 1$) are approximated by a nonlinear equation. We use a fixed point iteration for the determination of all values which works as follows:

1. We start with index $j = 0$ and equations (7.31c) and (7.31d) and set

$$K^0_{i,l} \quad \text{and} \quad L^0_{i,l} \quad \text{to zero for} \quad i = 1, \ldots, s,\ l = 0, \ldots, n_l - 1.$$

2. In the next step, compute

$$\bar{\Omega}_{i,\,l}, \bar{Y}_{i,\,l} \quad \text{and} \quad \bar{\omega}_{i,\,l} \quad \text{via equations} \quad (7.31\text{e}), (7.31\text{f}) \quad \text{and} \quad (7.31\text{g})$$

followed by

$$K^{j+1}_{i,l} \quad \text{and} \quad L^{j+1}_{i,l} \quad \text{for} \quad i = 1, \ldots, s,\ l = 0, \ldots, n_l - 1.$$

3. Compute

$$\delta = \max\left(\frac{K^{j+1}_{i,l} - K^{j}_{i,l}}{K^{j+1}_{i,l}}, \frac{L^{j+1}_{i,l} - L^{j}_{i,l}}{L^{j+1}_{i,l}}\right) \tag{7.34}$$

for $i = 1, \dots, s,\ l = 0, \dots, n_l - 1$.

4. If the value of equation (7.34) is small enough, take the values. Otherwise, increment j and proceed with step 2.

In our example, we compute a 2-dimensional lattice gauge field with periodic boundary conditions.Furthermore, we take a scheme with $s = 3$ stages using the coefficients derived in [64] and stated in equation (5.61). Algorithm 7.3 is developed in order to have convergence order $p = 3$. Due to the symmetry of the method, the convergence order is even with the result that a higher convergence order $p = 4$ is achieved as described in [60].

During the simulations, a lattice of size 8×8 is simulated. In this case, 5000 trajectories are computed with a thermalization phase of 500 additional trajectories. For the fixed point iteration, a relative error of $\delta = 10^{-8}$ is chosen as stopping criterion which is achieved after a few steps in case of our small investigated lattice. The convergence, time-reversibility and the computational cost are explored as described before.

The time-reversibility is checked for different relative errors in the fixed point iteration in figure 7.4. This time, we investigate a lattice of size 32×32 because the computation is quite fast. As expected, we see that the error of the time-reversibility goes along with the relative error of the fixed point iteration. Thus, we choose a relative error of $\delta = 10^{-8}$ (indicated with red plus signs) as stopping criterion of the fixed point iteration.

The volume-preservation is just roughly checked since we did not create a volume-preserving scheme. We investigated that the symmetric partitioned Runge-Kutta method is not volume-preserving (see figure 1 of [60]).

In figure 7.5, we compare the convergence orders of the standard Leapfrog scheme (blue crosses), the partitioned Munthe-Kaas Runge-Kutta scheme of convergence order 2 (yellow circles) and the symmetric partitioned Munthe-Kaas Runge-Kutta scheme of convergence order 4 (red plus signs). Here, the Leapfrog scheme serves as a reference. It can be seen that the convergence results of the Munthe-Kaas Runge-Kutta scheme of order 2 coincides with the Leapfrog scheme. Furthermore, it is shown that the symmetric partitioned Munthe-Kaas Runge-Kutta scheme of convergence order 4 indeed takes convergence order 4.

We are interested especially in a comparison of the computational cost of the Leapfrog scheme and the symmetric partitioned Munthe-Kaas Runge-Kutta scheme of convergence order 4. Thus, we investigate the CPU time per trajectory (divided by the number of links) in dependence on the mean value of the absolute differences in the Hamiltonian in figure 7.6. Again, the Leapfrog scheme is visualized via blue crosses and the SPRK4 scheme via red plus signs. We see that the computational cost of the SPRK4 scheme is lower than the one of the Leapfrog scheme for values of $\langle|\Delta H|\rangle$ lower than 10^{-3}. This means, the symmetric partitioned Munthe-Kaas scheme of convergence order 4 is cheaper than the Leapfrog scheme for mean differences in the Hamiltonian lower than approximately $10 - 3$ in case of a 8×8 lattice gauge field of $SU(2)$ matrices. This is a promising result and it indicates the potential of Munthe-Kaas schemes.

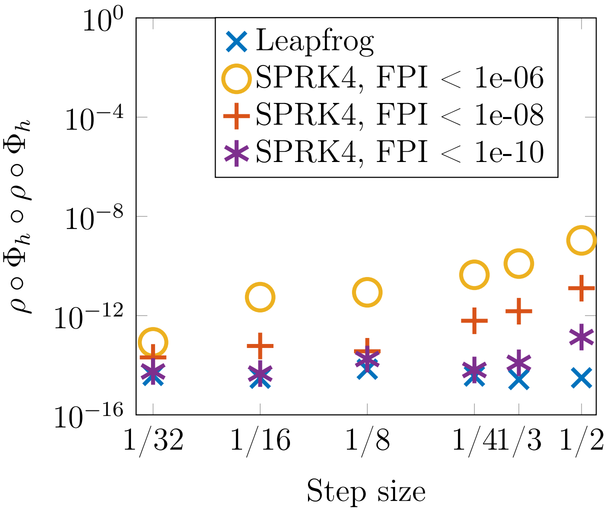

Figure 7.4: *Time-reversibility of the symmetric partitioned MK-RK scheme* – . Shown is the time-reversibility (of one configuration) of the standard Leapfrog scheme (blue crosses) and the SPMKRK scheme with relative errors smaller than 10^{-6} (yellow circles), 10^{-8} (red plus signs circles) and 10^{-10} (violet asterisks) for a lattice of size 32×32.

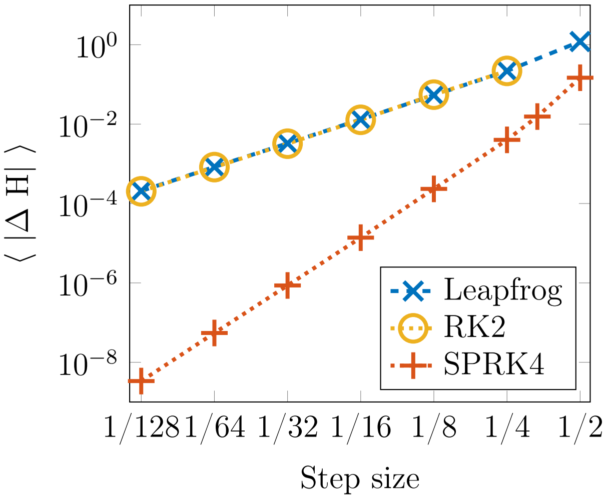

Figure 7.5: *Convergence order of the SPRK scheme.* – Shown are the mean value of the absolute difference of 2 Hamiltonians before and after a MD step with trajectory length $\tau = 1$ versus the step size h. The symmetric partitioned MKRK scheme (red plus signs diamonds) exhibits convergence order $p = 4$. The Leapfrog scheme (blue crosses) and the partitioned MKRK scheme of convergence order 2 (yellow circles) show convergence order $p = 2$.

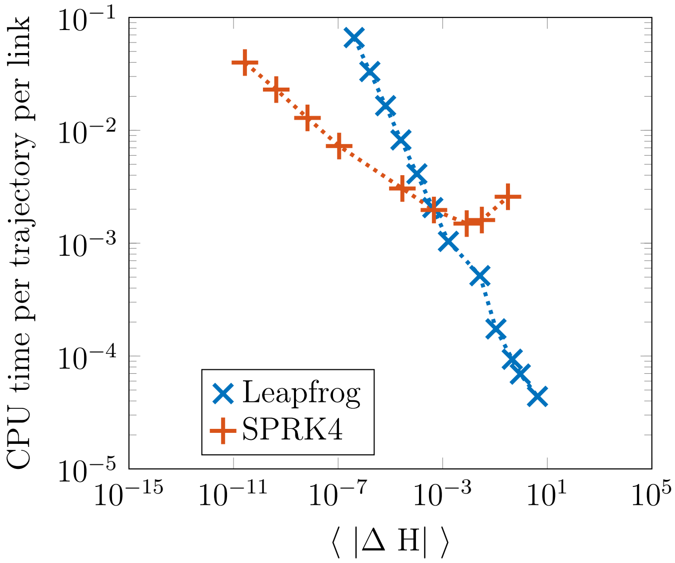

Figure 7.6: *Computational cost of the symmetric partitioned Munthe-Kaas Runge-Kutta scheme.* – Shown is the CPU time of the symmetric partitioned Munthe-Kaas Runge-Kutta scheme (blue) and the Leapfrog scheme (red) versus the mean value of the absolute difference of 2 Hamiltonians before and after a MD step with trajectory length $\tau = 1$.

7.2.4 The Time-Reversible Projection Method

The projection method stated in formula (5.78) is applied on the Hamiltonian equations of motion used in the HMC

$$\dot{Y}_j(t) = A_j(t)Y_j(t), \qquad \dot{A}_j(t) = F([Y_j(t)]_s) \quad \text{for} \quad j = 1, \ldots, n_l$$

which are stated in formula (7.22). For this problem, equation (5.82) has to be adapted to a whole configuration of Lie group / Lie algebra elements:

$$Y_{n+1,j} = \Phi_h^Y\big(Y_{n,j}, (1+\mu)A_{n,j}\big), \tag{7.35a}$$

$$A_{n+1,j} = \frac{1}{1+\mu}\Phi_h^A\big(Y_{n,j}, (1+\mu)A_{n,j}\big) \tag{7.35b}$$

for $j = 0, \ldots, n_l - 1$. The numerical method Φ_h can be chosen arbitrarily, it just has to be a method for a field of coupled Lie group / Lie algebra problems which is volume-preserving and time-reversible. For example, the Leapfrog methods (7.28) or (7.29) can be chosen in order that Φ_h^Y denotes the influence of the method on the Lie group elements $[Y]$ and Φ_h^A the change of the Lie algebra elements $[A]$. The parameter μ is a constant chosen in such a way that the constraint

$$\tilde{g}(\mu) := \mathcal{H}\Big([Y_{n+1}], [A_{n+1}]\Big) - \mathcal{H}\Big([Y_n], [A_n]\Big) = 0 \tag{7.36}$$

described in equation (5.83) is small enough, i.e. close to zero. Here, the Hamiltonian is computed according to formula (7.7).

Due to its numerical implementation, the projection scheme implies a very small change in the Hamiltonian. So, using projection schemes as geometric integration scheme inside the Hybrid Monte Carlo method, the proposed configuration will be accepted in almost all cases regardless of the step size. This means, the geometric properties time-reversibility and volume-preservation have to be investigated as well as the step-size dependence of the defect in the Hamiltonian. In section 5.4.2, we have shown that the projection scheme (5.78) is time-reversible. So, the adapted version (7.35) for a lattice gauge field is also time-reversible. The volume-preservation is called into question. So, we study the volume-preservation for the new projection scheme through a comparison with the pure Leapfrog scheme without any projection – which is known to be volume-preserving.

The projection scheme (7.35) is implemented in MATLAB using the Leapfrog scheme (7.28) for the inner one-step scheme Φ_h. Here, a code written by Prof. Dr. M. Striebel is used which determines the parameter μ via the scalar equation (7.36) using the bisection method. The numerical tests are performed for an $SU(3,\mathbb{C})$ lattice gauge field on a small lattice of size 8×8 with periodic boundary conditions. In doing so, we start from already thermalized configurations and compute the mean value of 100 different configurations including statistical errors – which are in most cases so small that they cannot be seen in the figures.

Numerical Results. The accuracy of the projection method just depends on the stopping criterion of the bisection method used for the constraint $\tilde{g}(\mu)$. This means, the difference in the Hamiltonian can be chosen with the result that it could always be smaller than another geometric integration method. In figure, 7.7 we compare the projection method using the Leapfrog scheme with a single Leapfrog scheme. The difference ΔH in the Hamiltonians is computed after a whole trajectory in order that the error of the Leapfrog method is of order h^2. Thus, the stopping criterion $g(\mu)$ of the projection method can be adapted in such a way that the difference in the Hamiltonian is smaller than for the Leapfrog method.

Volume-Preservation. For the volume-preservation, we compute the absolute value of the determinant of the Jacobian

$$D\Psi_h^\mu = \frac{\partial \Psi_h^\mu\big([Y_n],[A_n]\big)}{\partial\big([Y_n],[A_n]\big)}$$

of the system $\Psi_h^\mu\big([Y_n],[A_n]\big)$. Here, the Jacobian is obtained via a one-sided numerical differentiation with fixed perturbation $\varepsilon = 10^{-6}$. In contrast to the Abelian case in $\mathbb{R}^n$, the computation of the Jacobian of a Lie-group/Lie-algebra is rather involved and follows the calculus derived formula (2.3) in [38].

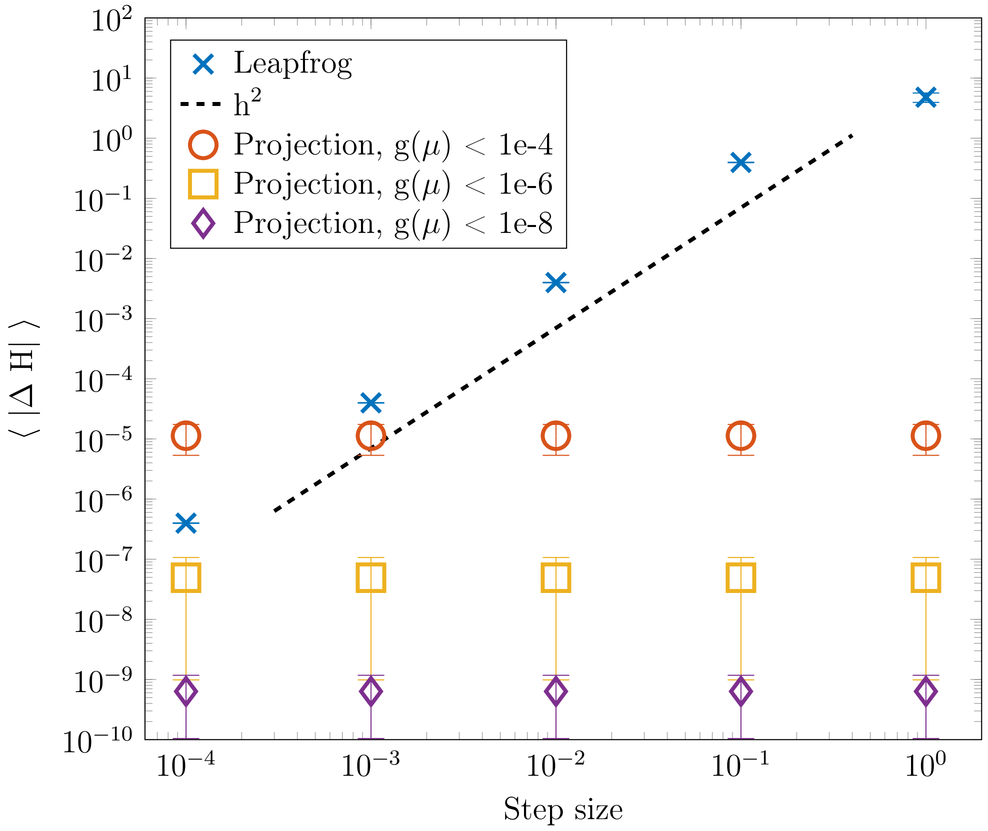

Figure 7.7: *Difference of 2 successive Hamiltonians.* – Absolute value of the difference in the Hamiltonian of 2 successive configurations $[Y_n, A_n]$ and $[Y_{n+1}, A_{n+1}]$ for Leapfrog method (blue circle) and projection method with constraint $g(\mu)$ smaller than 10^{-4} (black diamond), 10^{-6} (black square) and 10^{-8} (red diamond). Here, the mean value of 100 different configurations including statistical errors is given.

Figure 7.8 compares the Leapfrog method (which is known as volume-preserving)) and the projection method with constraint $g(\mu) < 10^{-8}$ according to volume-preservation. As discussed in section 5.4, the projection method is just volume-preserving if the projection parameter is constant, i.e. does not depend on the initial values. In our case of an $SU(3, \mathbb{C})$ lattice gauge field, the results of both schemes coincide. So, the numerical test results suggest the assumption that the projection scheme indeed is volume preserving, i.e., the projection parameter μ is constant or nearly constant. The errors are due to the discretization errors introduced by numerical differentiation.

7.3 Summary

In this chapter, the geometric Lie group methods developed in chapter 5 are implemented and tested in the context of lattice gauge fields in $SU(N)$ Yang-Mills theory.

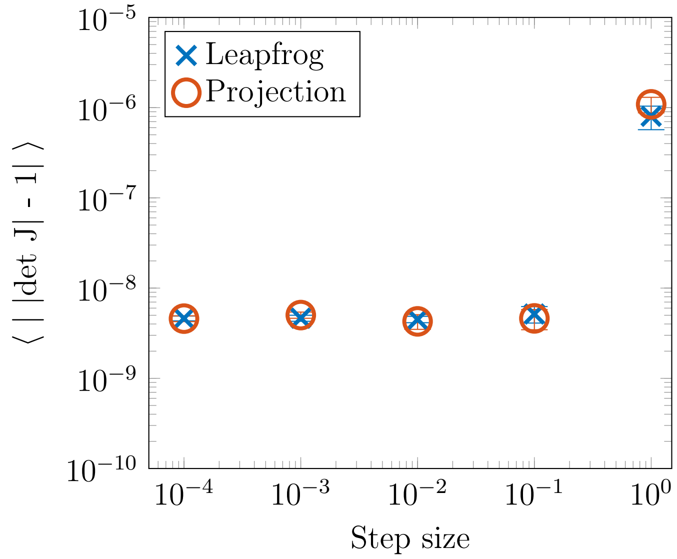

Figure 7.8: *Volume-preservation of the symmetric projection method.* – Volume-preservation: mean value of the absolute value of the determinant of the Jacobian of the system for Leapfrog (blue circle) and projection method with constraint $g(\mu)$ smaller than 10^{-8} (red diamond).

More precisely, lattice gauge fields $[U]$ with Wilson action $S_G([U])$ are simulated with a Hybrid Monte Carlo method with the result that the target distribution $\exp(-S_G)$ is met.

The focus of this thesis is the development of geometric integration methods for Lie group problems, particularly for the Hamiltonian equations of motion occurring in the Molecular Dynamics step of the HMC. The Cayley-Leapfrog method, the symmetric partitioned Munthe-Kaas Runge-Kutta method and the projection method designed in chapter 5 are tested in this chapter and compared with the standard Leapfrog method. The computational costs of all three schemes are lower than the one of the standard Leapfrog scheme. Furthermore, the methods are time-reversible.

The Cayley-Leapfrog scheme is also volume-preserving. It is the most promising among the analyzed schemes. The only disadvantage is that it is just suitable for quadratic Lie groups but the Lie group $\mathrm{SU}(N, \mathbb{C})$ used in Lattice QCD is quadratic. So, the Cayley transform is suitable for Lattice QCD computations and could also be used in any other numerical scheme apart from the standard Leapfrog scheme, for example, in higher-order Munthe-Kaas Runge-Kutta schemes.

The symmetric partitioned Munthe-Kaas Runge-Kutta scheme is time-reversible. The volume-preservation is not investigated here since the scheme suffers from its implicitness. The system is highly implicit due to the term $\exp(\frac{1}{2}\Omega_{n+1})$ in the increments L_i, l. So, the scheme needs a lot of computing time in order that a nonlinear system has to be solved in each step of the integration. The scheme is just cheaper in case of very accurate solutions. Thus, the scheme might be applicable for Lattice QCD but there are better ones. For example, the symmetric partitioned Munthe-Kaas Runge-Kutta scheme can be designed using the increments L_i, l, $i = 1, \ldots, s$, $l = 0, \ldots, n_l - 1$ stated in equation (5.49). Here, the appropriate order conditions have to be computed. Then, new coefficients can be found to set up a different symmetric partitioned Runge-Kutta scheme.

The projection scheme is designed to be time-reversible and volume-preserving as stated in chapter 5.4. However, the volume-preservation depends on a constant projection parameter μ. Although the numerical results indicate that violations of the volume-preservation are small, this is not yet proven. So, thetime-reversible projection scheme cannot be applied in Lattice QCD computations. Nevertheless, further investigations may show that it could be suitable for differential equations on Lie groups / Lie algebras.

8

Chapter 8

The Wilson Flow and Finite Temperature QCD

This chapter deals with simulations of the Wilson flow. Starting from initial values gained from Markov chain Monte Carlo simulations in Lattice QCD, the Wilson flow can be computed in addition. Since it is a differential equation on a Lie group, it can be computed with any of the Lie group methods described in paragraph 3.2 or developed in chapter 4, i.e. Crouch-Grossmann or Munthe-Kaas Runge-Kutta schemes with or without step size prediction and with exponential function or Cayley transform as local parameterization.

The first section introduces the model, a 4-dimensional lattice gauge field in $SU(3,\mathbb{C})$ Yang-Mills theory. Then, the step size prediction for Munthe-Kaas Runge-Kutta schemes developed in section 4.1 is applied on the Wilson flow in section 8.2. It uses standard Munthe-Kaas schemes based on the exponential function.Furthermore, we describe in paragraph 8.3 that the Wilson flow can be used for the detection of the critical temperature in Lattice QCD. This technique uses the difference between the temporal and spatial Wilson energy and approximates the critical temperature using the exponential smoothing spline described in chapter 6. These surveys are published in [63] and [66].

8.1 The Model

For the simulation of the Wilson flow, the software DDHMC provided by M. Lüscher is used. It simulates a 4-dimensional lattice with (time extension T and spacial volume L^3 and) periodic boundary conditions and $SU(3,\mathbb{C})$ gauge field. Here, Markov chains of gauge fields are computed following a probability distribution proportional to the Boltzmann distribution of the Wilson action. The Markov chains are computed for special values of β and taken as initial values for the Wilson flow.

Starting with initial values $[V_0] := [V(t_n)]$ given by the Hybrid Monte Carlo method, the Wilson flow is computed for a whole configuration of link matrices. At the same time, some observables are measured in order that their expectation values can be determined at the end.

Notation. The lattice consists of $n_l := 4 \cdot T \cdot L^3$ link matrices. In this chapter, we use two different notations for the Wilson flow:

$$\dot{V}_j(t) = Z_j(t) \cdot V_j(t), \qquad Z_j = F([V_j]_s) \quad \text{for} \quad j = 0, \ldots, n_l - 1 \tag{8.1}$$

as mentioned in equation (2.80) for the description of the algorithm and

$$\dot{V}_{x,\mu}(t) = Z_{x,\mu}(t) \cdot V_{x,\mu}(t), \qquad Z_{x,\mu}(t) = -\{V_{x,\mu}\mathcal{S}(V_{x,\mu})\}_{TA} \tag{8.2}$$

with $x = 0, \ldots T \cdot L^3 - 1$ and $\mu = 0, 1, 2, 3$ for the description of the formulae used for the computation inside the code. (2.78) As in chapter 7, there is a one-to-one correspondence between both notations. It holds

$$V_{x,\mu} \leftrightarrow V_j \quad \text{and} \quad Z_{x,\mu} \leftrightarrow Z_j = F([V_j]_s) \tag{8.3}$$

related through the lexicographic index $j := n_p \cdot \mu + x$ with $n_p = T \cdot L^{d-1}$ stated in equation (2.57a). The indices x and μ can also be computed from j via equation (2.57b). A configuration $[V_n] := [V(t_n)]$ consists of n_l elements. According to equation (7.23), $[V_n]$ can be denoted as

$$[V_n] = \{V_{n,0}, \ldots, V_{n,n_l-1}\} \, . \tag{8.4}$$

The fields $[\Omega_{n+1}]$ and $[\hat{\Omega}_{n+1}]$ appearing in the integration can be written down in the same way as

$$[\Omega_n] = \{\Omega_{n,0}, \ldots, \Omega_{n,n_l-1}\} \quad \text{and} \quad [\hat{\Omega}_n] = \{\hat{\Omega}_{n,0}, \ldots, \hat{\Omega}_{n,n_l-1}\} \, . \tag{8.5}$$

Wilson Energy. The Wilson energy a^4E is of special interest. It can be gained via the Wilson flow, taking the configurations of an HMC simulation as initial values. For the simulations, the Wilson energies

$$a^4E_{\square}(t) = 2 \sum_{x,\mu<\nu} ReTr\{1 - \mathcal{P}_{x,\mu\nu}([U](t))\}. \tag{2.81}$$

and

$$a^4E_{FT}(t) = \frac{1}{4} G^A_{\mu\nu} G^A_{\mu\nu} \tag{2.82}$$

stated in paragraph 2.4 are studied. The simple Wilson energy $a^4E_{\square}(t)$ is used for the investigation of the step size prediction and the more precise field strength energy a^4E_{FT} for the detection of the critical temperature.

The step size prediction is based on the units $[a^4E]$ and $[t/a^2]$. For the detection of the critical temperature, a scaling towards the physical units $[t/r_0^2]$ and $[t^2E]$ has to be done.

Scale setting. During the simulations, the Wilson energy a^4E is computed along a simulation time in units $[t/a^2]$. For physically interpretable simulations, both the energy unit and the simulation time have to be re-scaled. The simulation time goes from t/a^2 to t/r_0^2 and the energy unit is transformed from a^4E to t^2E. This means, the energy of the Wilson flow is multiplied with its squared simulation time t/a^2 in order that the units are transformed according to $a^4E \cdot (t/a^2)^2 = t^2E$. The scaling of the simulation time is a bit more complicated. The value a/r_0 can be computed for each β using equation (8.13). Thus, the value of β leads to a constant factor $z = ln(a/r_0)$ so as to a/r_0 can be computed as $\exp(z)$.

Using this model, the step size prediction of the Munthe-Kaas Runge-Kutta method and the detection of a phase transition by means of the Wilson flow are studied.

8.2 Step Size Control for Munthe-Kaas Runge-Kutta Methods

We consider the step size control for Munthe-Kaas Runge-Kutta schemes described in paragraph 4.1 for the Wilson flow (2.78), respective

$$\dot{V}_j(t) = Z_j(t) \ \cdot \ V_j(t), \qquad Z_j = F([V_j]_s) \quad \text{for} \quad j = 0, \ldots, n_l - 1 \tag{2.80}$$

with $V_j \in SU(3, \mathbb{C})$, $Z_j \in \mathfrak{su}(N, \mathbb{C})$ and a function $F : SU(3, \mathbb{C}) \to \mathfrak{su}(N, \mathbb{C})$. In this survey, the Wilson flow is computed for a single configuration with initial values chosen from a Markov chain Monte Carlo simulation. The configuration consists of a lattice gauge field of n_l link variables. For the exploration of the step size prediction, the observable of interest is the Wilson energy computed in units a^4E for the simulation time t/a^2. That implies that the step size prediction is applied on unscaled data which makes sense because it is done deep inside the simulation code.

Starting from initial values $[V(t_0)] = [V_n]$, i. e. from a given configuration, the new configuration $[V_{n+1}]$ is reached using a step size control as described in the next algorithm.

Algorithm 8.1 (Embedded Runge-Kutta Schemes for the Wilson Flow)**.**

1. Start from the initial values $[V(t_0)] = [V_n] = \{V_{n,0}, \ldots, V_{n,n_l-1}\}$ in the Lie group and choose an initial step size h.

2. Identify $V_j(t)$ and $\Omega_j(t)$ as

$$V_j(t) = \exp(\Omega_j(t)) \ V_{n,j}(t) \quad \text{and} \quad \dot{\Omega}_j(t) = d\exp_{\Omega_j}^{-1}(Z_j(t)). \tag{8.6}$$

   for $j = 0, \ldots, n_l - 1$.

3. Compute numerical approximations $[\Omega_{n+1}]$ and $[\hat{\Omega}_{n+1}]$ in the Lie algebra with convergence order p and $\hat{p} = p + 1$. Use the embedded scheme (8.8) here.

4. Determine the error *err* according to

$$\text{err} = \sqrt{\frac{1}{n_l}\sum_{j=0}^{n_l-1}\Big(\frac{||\hat{\Omega}_{n+1,j} - \Omega_{n+1,j}||}{ATOL + RTOL \cdot ||\hat{\Omega}_{n+1,j}||}\Big)^2} \tag{4.16}$$

   in an appropriate matrix norm.

5. Compute the optimal step size h_{opt} according to

$$h_{\text{opt}} = h \cdot \sqrt[p+1]{\frac{1}{\text{err}}} \cdot \rho, \qquad \rho < 1\ . \tag{8.7}$$

   as already defined in equation (4.5).

   - If $err \leq 1$, accept the new step size.
     - Compute $[V_{n+1}]$ from $[\Omega_{n+1}]$ or $[\hat{\Omega}_{n+1}]$.
     - Take $[V_{n+1}]$ as new value at time $t_{i+1} = t_i + h$.
   - If $err > 1$, the step size was too large. The step has to be repeated with the smaller step size h_{opt} computed in (8.7).

6. Set $h := h_{\text{opt}}$ and proceed with step 3.

In step 3, the following embedded Munthe-Kaas Runge-Kutta scheme is used:

Definition 8.2 (Embedded MKRK scheme for the Wilson Flow)**.** Let the Wilson flow (2.80) be given and use the parameterization $V_j = \exp(\Omega_j)V_n$. The unknowns $[\Omega_{n+1}]$ and $[\hat{\Omega}_{n+1}]$ are computed from

$$\Omega_{n+1,k} = h\sum_{i=1}^{s} b_i K_{k,i} \quad \text{and} \quad \hat{\Omega}_{n+1,k} = h\sum_{i=1}^{s} \hat{b}_i K_{k,i} \tag{8.8a}$$

with increments

$$K_{k,i} = f_{\hat{p}-2}(Y_{k,i}, Z_{k,i}) = \sum_{k=0}^{\hat{p}-2} \frac{B_k}{k!} ad^k_{Y_{k,i}}(Z_{k,i}) \tag{8.8b}$$

and internal stages

$$Y_{k,i} = h\sum_{j=1}^{\max(s,\hat{s})} a_{ij}K_{k,j}, \qquad V_{k,i} = \exp(Y_{k,i}) \cdot V_{k,n}, \qquad Z_{k,i} = F([V_{k,i}]_s) \tag{8.8c}$$

for all n_l points of the configuration. This means, the running index $k = 0, \ldots, n_l-1$ labels the elements of the lattice and i the stage numbers of the method.

Concerning step 5, the solutions $[V_{n+1}]$ of convergence order p and $[\hat{V}_{n+1}]$ of convergence order $\hat{p}$ are reached for all lattice points via

$$V_{n+1,k} = \exp(\Omega_{n+1,k})V_{n,k} \quad \text{and} \quad \hat{V}_{n+1,k} = \exp(\hat{\Omega}_{n+1,k})V_{n,k} \tag{8.9}$$

for $k = 0, \ldots, n_l - 1$. It is sufficient just to compute $[V_{n+1}]$ or $\hat{V}_{n+1}$ in order that the result has convergence oder p or $\hat{p}$.

Examples 8.3 (Embedded MK-RK scheme for the Wilson flow with BS coefficients)**.** The combination of embedded MKRK scheme for the Wilson Flow described in definition 8.2 with Bogacki-Shampine coefficients 4.1 reads

$$\hat{\Omega}_{n+1,k} = h\left(\frac{1}{2}K_{k,1} + \frac{1}{3}K_{k,2} + \frac{4}{9}K_{k,3}\right) \tag{8.10a}$$

$$\text{and} \quad \Omega_{n+1,k} = h\left(\frac{7}{24}K_{k,1} + \frac{1}{4}K_{k,2} + \frac{1}{3}K_{k,3} + \frac{1}{8}K_{k,4}\right) \tag{8.10b}$$

with joint increments

$$K_{k,i} = f_{\hat{p}-2}(Y_{k,i}, Z_{k,i}) = Z_{k,i} - \frac{1}{2}[Y_{k,i}, Z_{k,i}] \tag{8.10c}$$

and coefficients $k = 0, \ldots, n_l - 1$ and $i = 1, \ldots, 4$. The internal stages $Y_{k,i}$ and $Z_{k,i}$ read

$$Y_{k,1} = 0, \qquad Y_{k,2} = \frac{h}{2}K_{k,1}, \qquad Y_{k,3} = \frac{3}{4}hK_{k,2}, \qquad Y_{k,n} = \Omega_{n+1,k} \tag{8.10d}$$

and

$$Z_{k,i} = F([V_{k,i}]_s) \quad \text{with} \quad V_{k,i} = \exp(Y_{k,i}) \cdot V_{k,n}. \tag{8.10e}$$

We implement algorithm 8.1 using the embedded MK-RK scheme of definition 8.2 with Bogacki-Shampine coefficients, i.e. we use the equations (8.10) in step 3. Moreover, the maximum norm is used in equation (4.16). The parameters for the step size control are set to ATOL=$1e-3$, RTOL=0, $\rho = 0.8$. Additionally, the step size should not change too much which is controlled by

$$h_{\text{opt}} = \min(\alpha \cdot h, \max(\beta \cdot h, h_{\text{opt}}))$$

following the suggestion in [30] with $\alpha = 0.5$ and $\beta = 2$. This step size control leads to schemes of convergence order (2)3.

We compare the Wilson energy (2.81) computed (via the Wilson flow) with a Runge-Kutta method of order 2 with one computed with the aforementioned step size prediction for a lattice of size 8^4 and $\beta = 6.0$. In Lattice QCD, a common step size for Runge-Kutta methods of order $p = 2$ is $h = 0.01$. So, we compute the Wilson flow up to a simulation time $15[\frac{t}{a^2}]$ which takes 1500 steps. On the other hand, we used the step size prediction specified before. This takes just 99 steps with the result that it indicates that a step size prediction is very useful. The result is shown in figure 8.1.

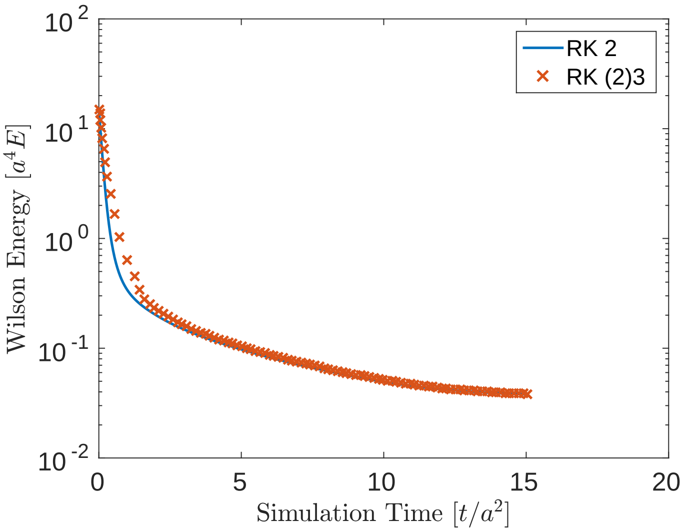

Figure 8.1: *Wilson energy computed with a Munthe-Kaas Runge-Kutta method* – of convergence order 2 (blue) and with step size control (red).

Discussion. In most cases, the Wilson flow is computed via a Runge-Kutta method with fixed step size. Here, the expectation values of the observables of interest are computed from many configurations including their statistical errors. The step size prediction has two advantages: first, the computational effort is reduced since the step size is adapted to the dynamics of the system. Then, the overall error can be controlled leading to smaller statistical errors. The example is computed for a single configuration of a small lattice of size 8^4, i.e. for $4 \cdot 8^4$ link variables. So far, there is no parallel version of the step size prediction implemented. So, the next natural step would be the development of a parallel version of this routine (the rest of the DDHMC code is parallelized) in order that more configurations and larger lattices can be computed leading to expectation values of observables of physical interest. Here, the Wilson flows of different configurations are probably computed at different time points with the result that the observables of interest finally have to be interpolated to a prescribed time grid. Furthermore, the computation of the error $||\Omega_{n+1} - \hat{\Omega}_{n+1}||$ should be explored and improved with focus on the used norm. Perhaps the most interesting thing is the control of the statistical errors. They can be reduced in a suitable way depending on the numerical error which is limited through the control parameters. This has to be analyzed in a next step. Also the control parameters $ATOL, RTOL$, and ρ can be approved.

8.3 Energy Difference Method for the Critical Temperature

In lattice computations, there is a critical temperature in the sense that the system behaves totally different below and above this temperature. Or, in other words:

"At a critical temperature Tc ≈ 200 MeV, QCD predicts a transition between the familiar confined hadron physics and a deconfined phase of quark gluon plasma (QGP)." [50]

The Wilson flow can be used for the detection of this critical temperature instead of using the Polyakov loop susceptibility. In this section, there is a new method for the detection of the critical temperature developed: the energy difference method which is based the Wilson flow. More precisely, the difference in the spatial and temporal Wilson energy is used for the detection of the phase transition.

This section is organized as follows: it starts with a description of the relation of lattice parameters and the temperature. Then, the computation of the temperature via the Polyakov loop is explained followed by the detection of the temperature via the new energy difference method. In both cases, the exponential smoothing spline developed in chapter 6 is used. Finally, the results of a Monte Carlo simulation are shown including a description of all steps necessary to reach the result.

Temperature. In Lattice QCD, simulations are performed on a 4-dimensional lattice with time extension T, space extension L and coupling constant β which sometimes is also called inverse temperature. The finite temperature $\mathcal{T}$ of a system can be determined if the space extension L is much larger than the time extension, i.e. $L \geq 4 \cdot T$ as described in [9]. Then, it holds

$$\mathcal{T}(\beta) = \frac{1}{T \cdot a(\beta)} \tag{8.11}$$

with lattice spacing $a(\beta)$ which is not directly known. Nevertheless, the last equation can be transformed to an equation in physical units:

$$\mathcal{T}(\beta)[\text{MeV}] = \frac{r_0/a(\beta)}{T \cdot r_0} \cdot \hbar c \tag{8.12}$$

According to Sommer [58], the values $\hbar c = 197.3$ MeV fm and $r_0 = 0.49$ fm can be used here. The remaining unknown r_0/a is the scale and can be computed via

$$\ln(a/r_0) = -1.6804 - 1.7331(\beta - 6) + 0.7849(\beta - 6)^2 - 0.4428(\beta - 6)^3 \tag{8.13}$$

for $5.7 \leq \beta \leq 6.92$ as explained in [48].

All in all, it is obvious that a fixed value for β is related to a fixed temperature of the system. Thus, the critical temperature $\mathcal{T}_c$ can be attained via a detection of the value for the critical β_c through the formula

$$\mathcal{T}_c(\beta_c)[\text{MeV}] = \frac{r_0/a(\beta_c)}{T \cdot 0.49} \cdot 197.3 \text{ MeV}. \tag{8.14}$$

Usually, the critical Temperature $\mathcal{T}_c$ is investigated using the susceptibility of the Polyakov loop which is explained next following the explanation of Lo in [35].

Polyakov Loop. The Polyakov loop is the product of link matrices $U_{x,\mu}$ in time dimension. Here, $\mu = 0$ is fixed and the link matrix can be described as

$$U_{x,0} = U_0(\vec{x}, n_t) \tag{8.15}$$

with time index n_t and index vector $\vec{x}$ describing the three space dimensions. So, for each spatial lattice point in L^3 there exists a Polyakov loop. The lattice average P_L of the Polyakov loop is defined as

$$P_L = \frac{1}{L^3} \sum_{\vec{x}} \frac{1}{3}\, Tr \prod_{n_t=1}^{T} U(\vec{x}, n_t). \tag{8.16}$$

Since the matrices $U(\vec{x}, n_t)$ are complex, the lattice average of the Polyakov loop is also a complex number. Thus, its susceptibility χ_P is computed as variance of the absolute value of P_L:

$$\chi_P := L^3 \cdot \Big(\langle |P_L|^2 \rangle - \langle |P_L| \rangle^2 \Big). \tag{8.17}$$

This example motivates the determination of the critical temperature via the Polyakov loop susceptibility according to Boyd et al [9] which is described here:

Algorithm 8.4. (Critical temperature via the Polyakov loop susceptibility)

1. Fix a lattice size $T \cdot L^3$
2. Let simulations run with various values of β around β_c.
   Compute the Polyakov loop susceptibility $\chi_P(\beta)$.
3. Take pairs of values $(\beta, \chi_p(\beta))$ and fit a curve through these points.
4. The curve $\beta, \chi_P(\beta)$ contains a peak around β_c. So, determine $\beta_c(\chi_P, T, L)$ as the value of β at which the curve has its maximum, e. g., through a spline.
5. Perform steps 2-4 for a new lattice with different spatial extension L.

Finally, the value $\beta_c(\chi_P, T, \infty)$ for an infinite volume is desired. It can be reached via an extrapolation in $1/L^3$ towards 0.

This way of detecting the critical temperature is very expensive if not just a pure gauge field is considered but in addition also a fermionic field. This does not depend on the computation itself but on the large auto-correlation effects as described in [69]. We develop an alternative way for the determination of the finite temperature phase transition: the energy difference method.

In combination with the Wilson flow, the Polyakov loop indicates a phase transition through the visualization of the lattice average P_L in the complex plane as shown in figure 8.2. The single Polyakov loops start at the center $(0, 0)$ and enlarge its distances from this point through the Wilson flow. For values of β which are smaller than the critical value β_c, the elements of P_L build a small circle. On the other hand (if $\beta > \beta_c$), the elements of P_L form a star with up to 3 arms, with progression of the Wilson flow, the elements are more far away from the center than in the other case.

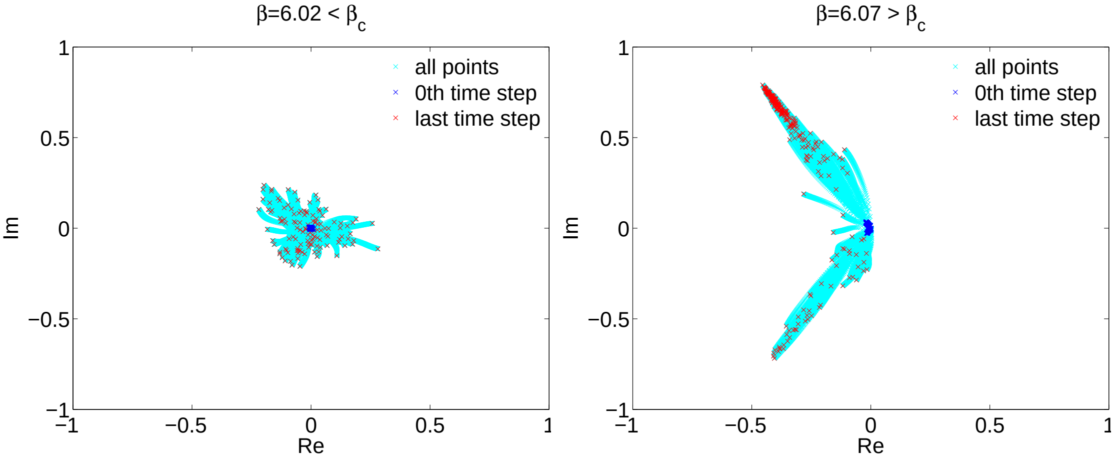

Figure 8.2: *Lattice average P_L of the Polyakov loop along the Wilson flow* – shown are the single Polyakov loops.

Energy Difference Method. The energy difference method is based on the fact that the Wilson energy (2.81) can be split in a spatial and a temporal part in order that the temporal part of the energy $a^4E_{st}(t)$ contains all plaquettes in the space-time planes and the spatial part $a^4E_{ss}(t)$ all plaquettes in the space-space planes. Both sums contain $3 \cdot T \cdot L^3$ plaquettes in order that it can be expected that the difference of the temporal and spatial energy is approximately zero. Indeed, this is the case for some values of β, more precisely, for values smaller than the critical value β_c. For values of β larger than the critical value β_c, the energy difference

$$\Delta E(t) := a^4E_{ss}(t) - a^4E_{st}(t) \tag{8.18}$$

is non-zero. The energy can be computed via the Wilson energy (2.81) or the field strength energy (2.82). In the first case, the energy is split in a temporal and spatial part of the energy as

$$a^4E_{\square,st}(t) = 2\sum_{x,\mu<\nu,\mu=0} ReTr\{1 - \mathcal{P}_{x,0\nu}([U(t)])\} \tag{8.19a}$$

$$\text{and} \quad a^4E_{\square,ss}(t) = 2\sum_{x,\mu<\nu,\mu\neq0} ReTr\{1 - \mathcal{P}_{x,\mu\nu}([U(t)])\} \tag{8.19b}$$

depending on the plaquettes $\mathcal{P}_{x,\mu\nu}([U(t)]$ in order that the energy difference reads

$$\Delta E_\square(t) := a^4E_{\square,ss}(t) - a^4E_{\square,st}(t). \tag{8.20}$$

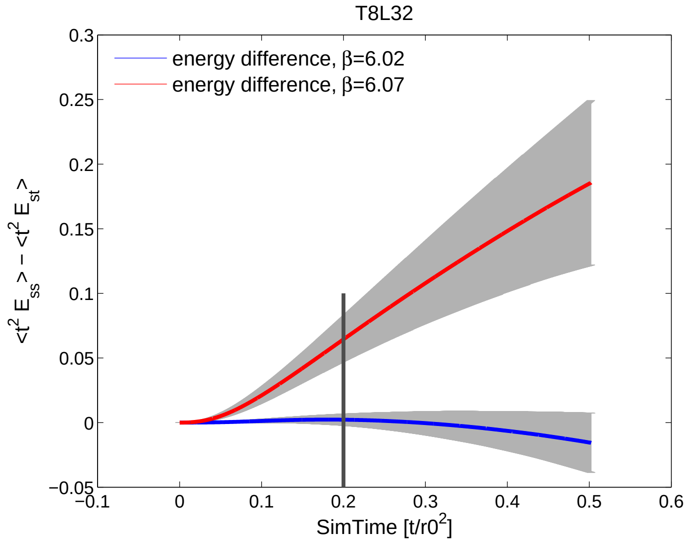

Figure 8.3: *Energy difference* $\Delta E -$ for a lattice of size 8×32^3.It holds $\Delta E \approx 0$ for $\beta = 6.02 < \beta_c$ and $\Delta E \neq 0$ for $\beta = 6.07 > \beta_c$ for $t \geq t_*$, e.g. $t_* = 0.2$

The field strength energy is partitioned the same way as

$$\Delta E_{FT}(t) := a^4 E_{FT,ss}(t) - a^4 E_{FT,st}(t) \tag{8.21}$$

with

$$a^4 E_{FT,ss}(t) := \frac{1}{4} G^A_{\mu\nu} G^A_{\mu\nu}, \qquad a^4 E_{FT,st}(t) := \frac{1}{4} G^A_{0\nu} G^A_{0\nu} \tag{8.22}$$

for $\mu, \nu = 1, 2, 3$, $\mu < \nu$ and field-strength tensor $G_{\mu\nu}$. Equations (8.22) are computed as described in equation (2.83) with a partitioning into a space-time and space-space part similar to formulae (8.19).

The energy difference is depicted in figure 8.3. Here, the Wilson flow is computed for $\beta = 6.02$ and $\beta = 6.07$ up to a flow time of $0.5[t/r_0^2]$ on a lattice of size 8×32^3. Then, the energy difference $\Delta E(t, \beta)$ is shown. We see that the energy difference is compatible with zero in the case of the small value of β and grows for the large value of β. This already holds for small simulation times.

So, the critical temperature $T_c(\beta_c)$ for the phase transition can be determined by finding the value of β at which the energies differ from each other. This survey is performed at a specific chosen flow time t_* which is chosen as large as needed to overcome the cut-off effects in the beginning of a Wilson flow and as small as possible to save computational time. The unit of the time is chosen as $\sqrt{t}/r_0$ according to the scale r_0 defined in [58]. This works as described in the following algorithm.

Algorithm 8.5. Energy Difference Method

1. Fix a lattice size $T \cdot L^3$ with $L \geq 4 \cdot T$.

2. Let simulations run with various values of β around β_c.

Compute the energy difference $\Delta E(\beta)$ at a certain time point t_* of the Wilson flow.

3. Take pairs of values $(\beta, \Delta E(\beta, t_*))$ and fit a curve through these points.

   Take into account that the values ΔE are expectation values gained via Monte Carlo simulations. So, they come along with a statistical error $\delta\Delta E(\beta, t_*)$ including auto-correlation effects.

4. The slope of the curve $(\beta, \Delta E(\beta, t_*))$ has its steepest slope at β_c. So, determine $\beta_c(\Delta E, T, L)$ as the value of β at which the slope of the curve has its turning point, e. g., through a spline.

5. Perform steps 2-4 for a new lattice with different spatial extension (which means the value of T is kept and the value of L changes).

The single steps need some more explanation. First of all, Boyd et al [9] mention that the spatial extension L must be larger than four times the time extension T. For our studies, we fix $T = 8$ and compute $\beta_{c,\Delta E}(T, L)$ for $L = \{32, 40, 48\}$.

Concerning step 3, the time point t_* for the evaluation of the energy difference has to be fixed. It should be as small as possible to save computing time and at the same time large enough in order that the shape of the curve $(\beta, \Delta E)$ allows the determination of the critical value β_c. Thus, we consider the curves $(\beta, \Delta E)$ at different time points. In figure 8.4, the development of $\langle a^4 \Delta E \rangle$ is shown in dependence of β for $t/r_0^2 = \{0, 0.01, 0.005, 0.1, 0.15, 0.2\}$. Here, the function approximating the values $(\beta, \Delta E)$ includes a steep ascent for values of $t_* \geq 0.01$ in units t/r_0^2. According to this, we choose $t_* = 0.0225[t/r_0^2]$, respective $t_* = 0.15[\sqrt{t}/r_0]$ for our considerations.

The curve $(\beta, \Delta E, \delta\Delta E)$ has to be fitted by a function which approximates the data as good as possible and at the same time avoids oscillations not included in the data. For this purpose, the exponential smoothing spline (6.8) developed in chapter 6 is used. It solves the minimization problem (6.1) with $y \equiv \Delta E(\beta, t_*)$, $w \equiv \delta\Delta E(\beta, t_*)$ and $S = 2$ as shown in figure 8.5.

Point 4 of algorithm 8.5 leads to the values of the critical coupling $\beta_c(\Delta E, T, L)$. Here we compute the maximum slope of the spline, i.e., the critical value $\beta_c(\Delta E, T, L)$ takes the value of β at which the second order derivative of the spline is zero which is also shown in figure 8.5.

Finally, we are interested in the value β_c for an infinite large lattice. For this purpose, a number of values $\beta_c(\Delta E, T, L)$ with different values of L are needed for an extrapolation towards $L = \infty$. Here, we repeat steps 2-4 of algorithm 8.5. For the extrapolation, also the errors $\delta\beta_c(\Delta E, T, L)$ have to be taken into account. They are computed using the Gaussian error propagation law as described in [66].

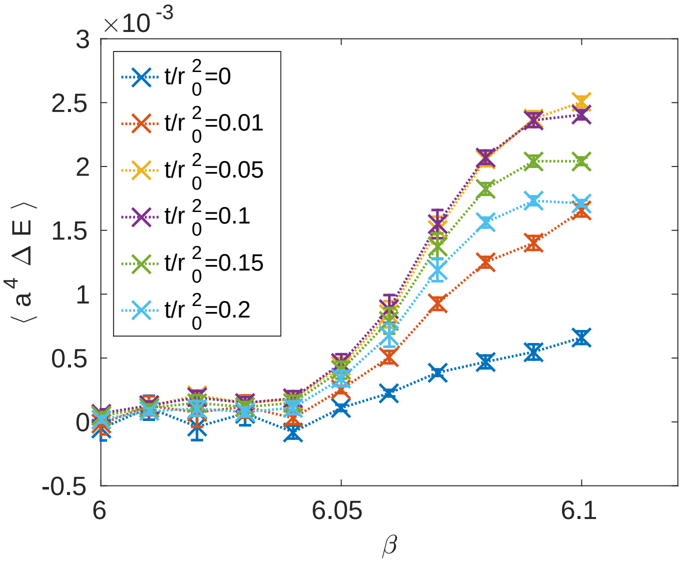

Figure 8.4: *Pairs* $(\beta, \Delta E(\beta)$ – including errors for a lattice of size 8×32^3 at certain time points.

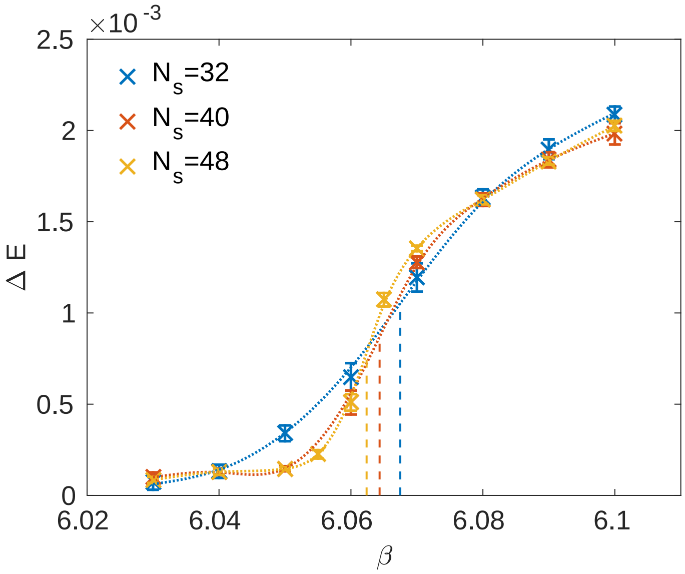

Figure 8.5: *Critical values of* β – for different lattices of size 8×32,8×40, 8×48. The data are approximated with an exponential smoothing spline and the values of $\beta_c(L, t_*)$ are detected as turning point of the slope.

Simulation. In this part, the details of the simulation are described. We simulate a lattice gauge field in $SU(3, \mathbb{C})$ Yang-Mills theory. The simulations are performed on a 4-dimensional lattice using a code which is based on the DDHMC code of Prof. Dr. M. Lüscher. We added the computation of the lattice average of the Polyakov loop, the Wilson energy and the field strength energy and the splitting in the spatial and temporal part of the energies. For the processing of the data using Matlab, an input routine of Dr. B. Leder is extended. So, we run Monte Carlo

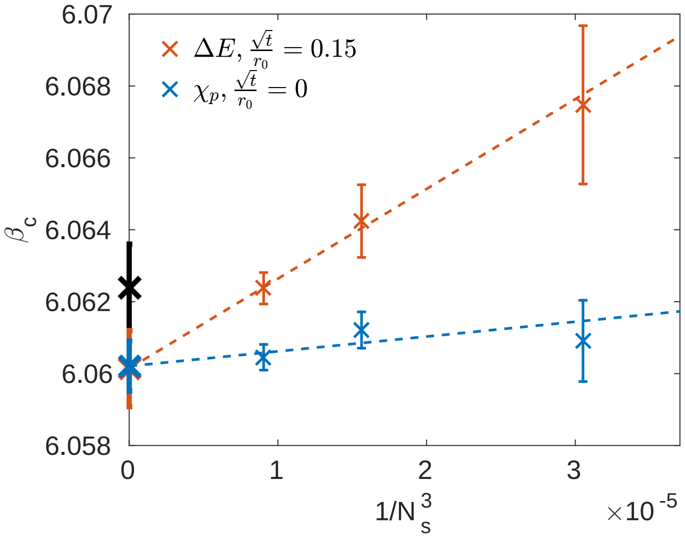

Figure 8.6: *Critical value for* β – comparison between the values β_c gained from [4] (black), and our simulations with the Polyakov loop susceptibility (blue) and the energy difference method (red).

simulations on the three lattices $8 \cdot 32^3$, $8 \cdot 40^3$ and $8 \cdot 48^3$ for different values of β to get initial values for the Wilson flow. Then, the Wilson flow is computed up to $t_* = 0.15\sqrt{t}/r_0$. At this point, the difference in the spatial and temporal part of the field strength energy is measured in order that pairs of data $(\beta_i, \Delta E_i, \delta\Delta E_i)$, $i = 1, \dots, n$ for each lattice occur. These data are fitted through an exponential smoothing spline explained in chapter 6. At the end $\beta_c(\Delta E, T, L)$ is computed as turning point of the spline, i. e. the point with maximum slope as shown in figure 8.5. Finally, the critical value $\beta_c(\Delta E, T, \infty)$ is of interest. So, the data $\beta_c(\Delta E, T, L)$ for $L \in \{32, 40, 48\}$ are extrapolated towards infinity leading to the final value. This is visualized in figure 8.6. In this picture, the critical value of β is detected via the energy difference method (red) and the Polyakov loop susceptibility (blue). Furthermore, the reference value (black) taken from [4] is shown. It can be seen that all three values are in good agreement.

Computational Effort. Monte Carlo simulations aim at producing expectation values from independent configurations. On the other hand, the Markov chain of configurations is correlated, quantified by the auto-correlation τ_{int}. According to equation (7.10), two configurations are considered to be independent if 2 τ_{int} HMC steps are computed in between. In doing so, the number of independent configurations is the number of all configurations divided by $2 \cdot \tau_{int}$. For the computation of the phase transition, we would like to have 100 independent configurations for each observable. We start with computing several Monte Carlo simulations for different values of β and different lattice sizes. In doing so, the Wilson flow is computed after 128 HMC steps of trajectory length 1. The number of initial configurations of the Wilson flow for the different combinations is given in table 8.1. Then, we compute the integrated auto-correlation times according to Wolff [69] using the program of

	$N_s = 32$	$N_s = 40$	$N_s = 48$
$\beta = 6.03$	220	281	401
$\beta = 6.04$	560	693	509
$\beta = 6.05$	2000	1597	2493
$\beta = 6.055$	-	-	1981
$\beta = 6.06$	2000	4097	9297
$\beta = 6.065$	-	-	3785
$\beta = 6.07$	2000	4861	3405
$\beta = 6.08$	397	437	548
$\beta = 6.09$	297	232	32
$\beta = 6.10$	397	132	312

Table 8.1: Number of computed configurations.

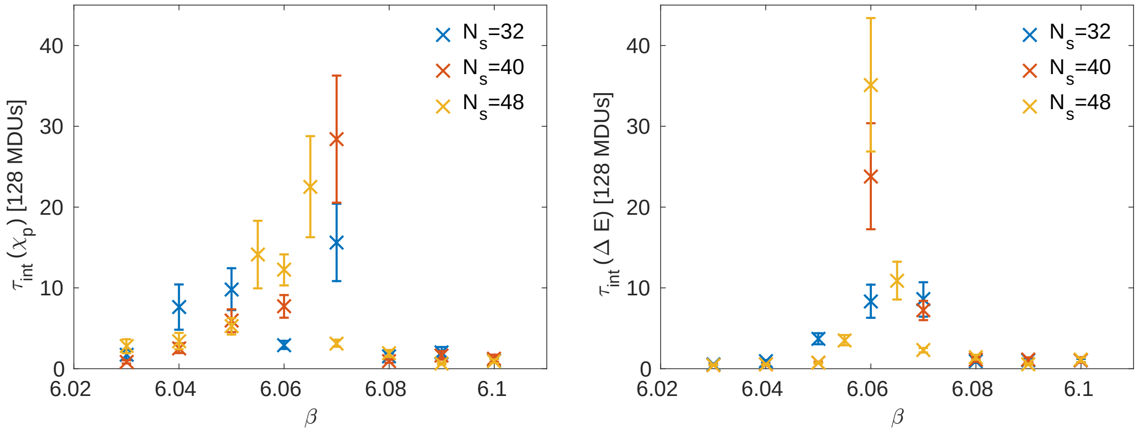

Figure 8.7: *Integrated auto-correlation times* – of the Polyakov loop susceptibility and the energy difference method.

Wolff for the Polyakov loop susceptibility and the energy difference method. These auto-correlation times are depicted in figure 8.7. We see that the value for $\tau_{int}(\chi_p)$ is – except for the value $\beta = 6.06$ – always larger than the value for $\tau_{int}(\Delta E)$. Dividing the values of table 8.1 by the values $2\tau_{int}(\chi_p)$ and $\tau_{int}(\Delta E)$ leads to the number of independent configurations mentioned in table 8.2. Thus, the energy difference method needs less configurations than the determination of the phase transition via the Polyakov loop susceptibility.

	$N_s = 32$		$N_s = 40$		$N_s = 48$	
	χ_p	ΔE	χ_p	ΔE	χ_p	ΔE
$\beta = 6.03$	62(20)	208(39)	166(38)	287(34)	72(22)	426(43)
$\beta = 6.04$	36(13)	312(56)	135(32)	683(73)	74(21)	508(45)
$\beta = 6.05$	101(26)	269(49)	134(31)	1034(112)	238(46)	1784(139)
$\beta = 6.055$	-	-	-	-	70(20)	280(50)
$\beta = 6.06$	339(57)	119(29)	265(48)	86(23)	380(59)	132(31)
$\beta = 6.065$	-	-	-	-	84(23)	173(37)
$\beta = 6.07$	63(19)	116(29)	85(23)	337(55)	540(74)	746(89)
$\beta = 6.08$	134(33)	215(46)	236(48)	198(44)	145(34)	193(40)
$\beta = 6.09$	72(23)	145(35)	70(22)	105(29)	246(47)	279(44)
$\beta = 6.10$	195(45)	198(42)	51(18)	61(19)	168(40)	143(34)

Table 8.2: Number of independent configurations.

8.4 Summary

In this chapter, the focus is put on the Wilson flow which is a differential equation on the Lie group $SU(3,\mathbb{C})$. Thus, the Wilson flow can be computed with any numerical integration method for Lie groups as, for example, the Munthe-Kaas Runge-Kutta scheme with step size prediction or Munthe-Kaas schemes using the Cayley transform as local parameterization or a combination.

In section 8.2, we applied a Munthe-Kaas Runge-Kutta scheme with step size prediction on one configuration of a four-dimensional lattice gauge field. This leads to promising results in order that this investigation should be extended towards larger lattices, more realistic simulations including many configurations and the control of its statistical errors. In section 8.3, we developed a new method for the detection of the critical temperature in finite temperature computations. Compared to the standard way using the Polyakov loop susceptibility, the Wilson flow has to be computed in addition, but just towards the time point t_* which can be chosen as quite small value, for example, as $t_* = 0.15[\sqrt{t}/r_0]$. In return, the auto-correlation errors are much smaller which saves computing time at the end. This would be advantageous for simulations including fermions. Indeed, the energy difference method is used from other groups as one possibility for the detection of the finite temperature phase transition [19, 2, 3, 1, 31]. In [2], it is indeed applied on fermions.

9

Chapter 9

Conclusion and Outlook

9.1 Conclusion

This work includes two different topics: simulations of gauge fields in Lattice QCD and numerical solutions of differential equations on Lie groups.

The connection between these topics are the Hamiltonian equations of motion which have to be solved in the Molecular Dynamics step of Hybrid Monte Carlo simulations in Lattice QCD. The Hamiltonian equations of motion consist of many pairwise coupled differential equations of shape

$$\dot{Y}(t) = A(t) \cdot Y(t), \qquad \dot{A}(t) = F([Y(t)]_s) \tag{2.85}$$

on a Lie group and its associated Lie algebra as mentioned in chapter 2. Here, $Y(t)$ is a matrix Lie group element, $A(t)$ an element of its associated matrix Lie algebra and the function F a mapping from the Lie group to the Lie algebra. The underlying theory for differential equations on Lie groups is stated in chapter 3. Thereby, the applied numerical integration scheme for the Hamiltonian equations is required to be time-reversible and volume-preserving. So, geometric numerical integration schemes for coupled Lie group / Lie algebra problems are of interest. Here, the focus is put on the one hand on the geometric properties time-reversibility and volume-preservation and on the other hand on the numerical solution of differential equations of Lie groups which may be used in the context of Lattice QCD. The differential equation in the Lie algebra needs no special treatment and can be solved with any known numerical integration scheme for the Abelian case. So, based on differential equations on Lie groups and its numerical solution described in chapter 3, different new concepts for the numerical solution of differential equations on Lie groups are evolved in chapter 4:

- a step size prediction for Munthe-Kaas Runge-Kutta schemes and
- the Cayley transform as local parameterization on the Lie group.

These concepts can be applied, for example, on the Wilson flow which is a differential equation on a Lie group of shape

$$\dot{V}(t) = Z(t) \cdot V(t) \quad \text{with} \quad Z(t) = F([V(t)]_s) \tag{2.88}$$

with matrix Lie group element $V(t)$ and matrix Lie algebra element $Z(t)$. Partly based on that, geometric numerical integration schemes for differential equations of type (2.85) are developed in chapter 5:

- the Cayley-Leapfrog method,
- symmetric partitioned Munthe-Kaas Runge-Kutta methods,
- a time-reversible projection method.

Finally, the different numerical integration schemes are tested in the context of gauge fields in $SU(N)$ Yang-Mills theory, either in a Hybrid Monte Carlo simulation or just using the Wilson flow in the context of a lattice gauge field.

In the following, the different numerical methods are subsumed, embedded and evaluated in the context of simulations in lattice gauge theory.

Step size prediction. A step size control for Munthe-Kaas Runge-Kutta schemes is designed in chapter 4.1. Here, a step size control for embedded Runge-Kutta schemes for the Abelian case is combined with Munthe-Kaas Runge-Kutta methods. In this case, it is convenient to compute all increments K_i, $i = 1, \dots, \max\{s, \hat{s}\}$ according to a truncation index $q = \max\{p, \hat{p}\} - 2$. Furthermore, the error measure has to be adapted for matrix Lie algebra elements. Finally, an example with Bogacki-Shampine coefficients (2)3 is described for a single differential equation on a Lie group. This example is adapted to the Wilson flow of a single configuration and finally simulated in chapter 8.2 for a small ($T = 8$, $L = 8$) 4-dimensional lattice using a program based on the DDHMC code of Lüscher. Here, just the feasibility of a step size prediction for the Wilson flow is tested. Due to the shape of the Wilson flow - a steep descend followed by slowly varying values, the numerical results look promising with the result that this kind of step size control would always be advantageous for the Wilson flow. The step size prediction is just tested for single configurations. This survey can be found in a compact form in [63].

Since the Wilson flow is used for the computation of expectation values of some observables as, for example, the energy, ensembles of configurations should be considered. For the evaluation of the expectation values at certain time points, the observables finally have to be interpolated on a time grid which can be done, for example, using linear interpolation. The computation of expectation values could be the next point in further work. Furthermore, the error norm and the parameters of the step size control could be improved. Based on that, the effect of the step size control on auto-correlation effects can be investigated. In the best case, a mechanism for the control of auto-correlation errors could be developed in this way.

Cayley transform. The second new concept is the usage of the Cayley transform as possibility for a parameterization of the Lie group that can be used in any numerical method for Lie groups. The goal of this development is the application of the Cayley transform in Lattice QCD where almost all numerical integration schemes are based on the Leapfrog or Störmer-Verlet scheme. So, the usage of the

Cayley transform is explained in the context of the Lie-Euler scheme in section 4.2 which is re-used in the Leapfrog scheme in paragraph 5.2. In paragraph 7.2.2, the Cayley-Leapfrog method is adapted for the Hamiltonian equations of motion occurring in the HMC simulation of a 2-dimensional $SU(2,\mathbb{C})$ gauge field in Lattice QCD. The numerical results are obtained with a Matlab-program on a small lattice of size 8×8. They show that the method is time-reversible and volume-preserving. Furthermore, the desired convergence order $p = 2$ is achieved. A comparison with the standard Leapfrog scheme using the exponential function shows that the Cayley-Leapfrog scheme is $\approx 4 - 5$ times faster than the standard scheme.

The results look very promising such that the Cayley transform should be investigated in more detail. In a recent bachelor thesis [44], Muniz proved the time-reversibility and volume-preservation for the Cayley-Leapfrog scheme for the Lie group $SU(3,\mathbb{C})$. Furthermore, the aforementioned results are published in [65]. In a next step, the Cayley-Leapfrog method could be tested on a 4-dimensional lattice in $SU(3,\mathbb{C})$ Yang-Mills theory, for example, with the aforementioned DDHMC-code. Concerning the exponential function and the Cayley transform, I would expect that the Cayley transform can be favored in all simulations of quadratic matrix Lie groups, at least for special unitary Lie groups. Nevertheless, it should be investigated if there may occur unpleasant surprises for the Cayley transform and also if the auto-correlation is affected.

Symmetric partitioned Munthe-Kaas Runge-Kutta methods. Symmetric partitioned Munthe-Kaas Runge-Kutta methods are evolved in section 5.3. Here, partitioned Runge-Kutta methods for the Abelian case are adapted to the Lie group / Lie algebra structure of differential equations of type (2.85) leading to algorithm 5.17. For the special case of convergence order $p = 2$, partitioned Munthe-Kaas Runge-Kutta methods work and coincide with the Leapfrog method. This happens due to the truncation inside the Munthe-Kaas Runge-Kutta scheme. For a general case $p > 2$, it can be shown that the symmetry conditions for the non-Abelian case do not coincide with these of the Abelian case. More precisely, there are six instead of four symmetry conditions and three of them lead to a contradiction or to coefficients such that the whole method does not work (because the initial values would be fixed). These contradictions can be overcome changing the increments $L_i, i = 1, \dots, s$ in algorithm 5.17 according to the proposals (5.48),(5.49) or (5.50). This change involves a variation of the symmetry conditions such that the dissent is resolved. For a derivation of appropriate coefficients, some further work has to be included. The order conditions have to be derived depending on the shape of the symmetric partitioned Munthe-Kaas Runge-Kutta method and the desired convergence order. This leads to the order conditions similar but not equal to them of the partitioned Runge-Kutta method for the Abelian case.

One set of coefficients is derived for the choice (5.50) of increments L_i, $i = 1, \dots, s$ and convergence order $p = 3$, i.e. $K_i = f_q$ with $q = 1$. Here, the scheme gets fully implicit due to the shape of the increments L_i. Nevertheless, this method is implemented for a 2-dimensional gauge field in $SU(2,\mathbb{C})$. This model is described in section 7.1.2 and the results of the simulation are discussed in section 7.2.3. It can be seen that the method is time-reversible and has convergence order $p = 4$. This depends on the symmetry of the scheme. The volume-preservation is not investigated here, but since it is not involved in the development of the method, it is assumed that the scheme is not volume-preserving. Due to the implicitness of the scheme, a fixed point iteration is included in the implementation. In spite of that, the computational cost for small step sizes is smaller than for the Leapfrog method.

The next natural step would be the investigation of symmetric partitioned Munthe-Kaas Runge-Kutta methods with increments L_i chosen according to equation (5.49) so as to the increments K_i and L_i depend on different coefficients but L_i is not fully implicit by construction. If this is successful, i.e. if there are symmetry and order conditions without contradiction, a set of coefficients could be computed. Then, in a next step, volume-preservation could be included.

Time-reversible projection methods. The time-reversible projection method is designed to be a volume-preserving and time-reversible projection method. It is developed for the Abelian and adapted for the non-Abelian case of coupled Lie group / Lie algebra problems. Unfortunately, this method is just volume-preserving if the projection parameter μ derived by the constraint does not depend on the initial values. This has to be checked for each model separately. Nevertheless, the method is applied on a lattice gauge field of $SU(3, \mathbb{C})$ matrices and leads to promising results. It seems that the projection parameter μ is independent on the input data in that case. In a next step, this should be investigated in more detail.

Exponential Smoothing Splines. Data with uncertainties need to be approximated without insertion of oscillations not given in the data. The exponential smoothing spline achieves this smooth approximation. It is stated in chapter 6 including all formulae for its computation. In this work, it is required for the computation of the critical temperature using the energy difference method in section 8.3

Critical temperature and the energy difference method. During the Work with the Wilson flow, we detected that the spatial and temporal Wilson energy behave sometimes similar and sometimes totally different depending on the value of β. More exact, there is a critical temperature $\mathcal{T}_c$ of the system related to a critical value of β called β_c with the result that the absolute value of the energy difference ΔE is approximately zero for values $\beta < \beta_c$ and larger than zero for values $\beta > \beta_c$. So, running simulations for a lattice of size $T \times L^3$, an exponential smoothing spline can be put through the data $(\beta, \Delta E(\beta), \delta\Delta E(\beta))$ with errors $\delta\Delta E(\beta)$. The turning point of this spline delivers the critical value $\beta_c(L)$. Fixing T and varying L leads to more critical values for different lattice sizes. Finally, an extrapolation in $1/L^3$ leads to the desired value for $\beta_c(\infty)$ for infinite lattice size. We run various simulations in order that β_c can be determined. The energy difference method is very promising because its auto-correlation effects are smaller than the ones of the Polyakov loop susceptibility which is the standard method. So, less simulations have to be run than using the standard approach but with an additional computation of the Wilson flow. However, the disadvantage of computing the Wilson flow is negligible because just a few steps of the Wilson flow up to the simulation time $t_* = 0.0225[t^2/r_0^2]$ are necessary. In next step, the energy difference method can be applied on a lattice gauge field including fermions. This utilization is even more promising because the auto-correlation errors are even larger in HMC simulations using fermions.

9.2 Outlook

Geometric Integration. The focus of this thesis is the development of geometric numerical integration methods for coupled Lie group / Lie algebra schemes that can be employed inside the Hybrid Monte Carlo algorithm of Lattice QCD computations. Here, the usage of the Cayley transform in any geometric integration scheme is very promising for Lattice computations and should be pursued. The next natural step would be the replacement of the exponential function by the Cayley transform in some HMC simulations and the investigation of the consumed computation time and the expectation values obtained in that way. The Cayley transform can also be used for the development of symmetric partitioned Munthe-Kaas Runge-Kutta schemes in order that the increments K_i, $i = 1, \dots, s$ do not change due to a desired convergence order. Independently, the order conditions for symmetric partitioned Munthe-Kaas Runge-Kutta schemes using the two other proposed increments L_i, $i = 1, \dots, s$ should be developed and compared to its symmetry conditions. Provided that there occur no contradictions, a set of coefficients can be derived and tested for a lattice gauge field. In the best case, an explicit (or closely related) symmetric partitioned Runge-Kutta scheme can be obtained. In addition, the volume-preservation could be included in a further step. The time-reversible projection scheme is volume-preserving if the projection parameter is independent on the initial values. Since a lattice gauge field contains many links and momenta which build the Hamiltonian contained in the constraint, this will likely be true and has to be investigated in more detail. If the projection parameter is constant for a chosen step size, this would be very advantageous for the HMC since the acceptance step coincides with the constraint of the projection method. So, the acceptance step can be dropped and any proposed new configuration will be accepted regardless of the step size of the numerical integration method.

The Wilson Flow. The Wilson flow is the second big topic for the outlook. First of all, it is used for the detection of the critical temperature of finite temperature simulations. Up to now, just lattice gauge fields without fermions have been considered. In a next step, the fermions should be included in the simulations with the result that the quality of the energy difference method can be investigated in that case. Then, the step size prediction can be investigated in more detail. So far, we have seen that a step size prediction for the Wilson flow is possible and makes sense.

Here, the open question arises how the step size prediction can control the statistical errors including its auto-correlation behavior. So, the absolute and relative tolerances as well as the chosen matrix norm for the error control can be considered more specifically.

Finally, all observed topics can be related to each other and used jointly. For example, the open question arises if a step size control for geometric integration schemes for coupled differential equations on Lie groups / Lie algebras is possible. Furthermore, the developed schemes can be applied on different models occurring in different topics, for example, in rigid body simulations or stochastic differential equations on matrix Lie groups.

Bibliography

[1] V. Ayyar, T. DeGrand, D. C. Hackett, W. I. Jay, and E. T. Neil. Chiral transition of su(4) gauge theory with fermions in multiple representations. *EPJ Web Conf.*, 175, 2018.

[2] V. Ayyar, T. DeGrand, D. C. Hackett, W. I. Jay, E. T. Neil, Y. Shamir, and B. Svetitsky. Finite-temperature phase structure of SU(4) gauge theory with multiple fermion representations. *Phys.Rev.D*, 97(11), 2018.

[3] V. Ayyar, D. Hackett, W. Jay, and E. Neil. Confinement study of an SU(4) gauge theory with fermions in multiple representations. *EPJ Web Conf.*, 175, 2018.

[4] B. Beinlich, F. Karsch, E. Laermann, and A. Peikert. String tension and thermodynamics with tree level and tadpole improved actions. *Eur. Phys. J.*, C(6), 1999.

[5] J. Bernoulli. Ars conjectandi. *Werke*, 3(pp. 107–286), 1975.

[6] P. Bogacki and L. Shampine. A 3(2) pair of Runge - Kutta formulas. *Applied Mathematics Letters*, 2(4):321 – 325, 1989.

[7] S. Borsanyi, S. Durr, Z. Fodor, C. Hoelbling, S. D. Katz, S. Krieg, T. Kurth, L. Lellouch, T. Lippert, and K. K. S. C. McNeile. High-precision scale setting in lattice QCD. 2012. arXiv:1203.4469 [hep-lat].

[8] N. Bou-Rabee and J. M. Sanz-Serna. Geometric integrators and the Hamiltonian Monte carlo method. *Acta Numerica*, 27:113–206, 2018.

[9] G. Boyd, J. Engels, F. Karsch, E. Laermann, C. Legeland, M. Lutgemeier, and B. Petersson. Thermodynamics of SU(3) lattice gauge theory. *Nucl. Phys.*, B469:419–444, 1996.

[10] M. Bruno, T. Korzec, and S. Schaefer. Setting the scale for the CLS $2+1$ flavor ensembles. *Phys. Rev.*, D95(7):074504, 2017.

[11] J. Butcher. On runge-kutta processes of high order. *J.Austral. Math. Soc.*, IV, Part 2:179–194, 1964.

[12] E. Celledoni. MA8404-Notes on Lie group methods and exponential integrators. *Lecture Notes, Department of Mathematical Sciences NTNU*, 2006.

[13] A. Cline. Scalar and planar-valued curve fitting in one and two-dimensional spaces using splines under tension. *Comm. ACM*, 17:218–223, 1974.

[14] G. Costa and G. Fogli. *Symmetries and Group Theory in Particle Physics: An Introduction to Space-Time and Internal Symmetries*, volume 823 of *Lecture Notes in Physics*. Springer, 2012.

[15] P. E. Crouch and R. Grossmann. Numerical Integration of Ordinary Differential Equations on Manifolds. *J. Nonlinear Sci.*, 3:1–33, 1993.

[16] T. DeGrand and C. DeTar. *Lattice Methods for Quantum Chromodynamics*. World Scientific Publishing Co Pte., 2006.

[17] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195:216–222, 1987.

[18] K. Engø. Partitioned Runge-Kutta Methods in a Lie-Group Setting. *BIT Numerical Mathematics*, 43:21–39, 2003.

[19] A. Florio, O. Kaczmarek, and L. Mazur. Open-Boundary Conditions in the Deconfined Phase. *Eur.Phys.J.C*, 79(12), 2019.

[20] P. Fritzsch and A. Ramos. The gradient flow coupling in the Schrödinger Functional. *JHEP*, 2013.

[21] C. Gattringer and C. B. Lang. Quantum chromodynamics on the lattice. *Lect. Notttliebes Phys.*, 788:1–343, 2010.

[22] S. A. Gottlieb, W. Liu, D. Toussaint, R. L. Renken, and R. L. Sugar. Hybrid Molecular Dynamics Algorithms for the Numerical Simulation of Quantum Chromodynamics. *Phys. Rev.*, D35:2531–2542, 1987.

[23] S. A. Gottlieb, W. Liu, D. Toussaint, R. L. Renken, and R. L. Sugar. Hybrid molecular dynamics algorithms for the numerical simulation of quantum chromodynamics. *Phys. Rev.*, D35:2531–2542, 1987.

[24] D. Griffiths and D. Higham. *Numerical Methods for Ordinary Differential Equations: Initial Value Problems*. Springer Undergraduate Mathematics Series. Springer, 2010.

[25] D. J. Griffiths. *Introduction to elementary particles*. TextBook Physics. Wiley, New York, NY, 1987.

[26] E. Hairer. Symmetric projection methods for differential equations on manifolds. *BIT Numerical Mathematics*, 40:726–734, 2000. 10.1023/A:1022344502818.

[27] E. Hairer, C. Lubich, and G. Wanner. Geometric numerical integration illustrated by the Störmer–Verlet method. *Acta Numerica*, 31:399–450, 2003.

[28] E. Hairer, C. Lubich, and G. Wanner. Geometric numerical integration illustrated by the störmer/verlet method. *Acta Numerica*, 12:399–450, 2003.

[29] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration – Structure-Preserving Algorithms for Ordinary Differential Equations*, volume 31 of *Springer series in computational mathematics.* Springer Heidelberg, 2 edition, 2006.

[30] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I – Nonstiff Problems.* Springer, second revised edition, 2000.

[31] K. Hieda, H. Makino, and H. Suzuki. Proof of the renormalizability of the gradient flow. *Nucl.Phys.B*, 918:23–51, 2017.

[32] A. Iserles, H. Z. Munthe-Kaas, S. P. Nørsett, and A. Zanna. Lie-group methods. *Acta Numerica*, 9:215–365, 2000.

[33] A. D. Kennedy, P. J. Silva, and M. A. Clark. Shadow Hamiltonians, Poisson Brackets, and Gauge Theories. *Phys. Rev.*, D87(3):034511, 2013.

[34] F. Knechtli, M. Günther, and M. Peardon. *Lattice Quantum Chromodynamics: Practical Essentials.* SpringerBriefs in Physics. Springer, 2017.

[35] P. M. Lo. Polyakov loop susceptibilities in pure gauge system. *J. Phys.: Conf. Ser.*, 503(012034), 2014.

[36] R. Lohmeyer and H. Neuberger. Continuous smearing of 'Wilson Loops'. *PoS LATTICE2011*, 2011.

[37] M. Lüscher. Properties and uses of the Wilson flow in lattice QCD. *JHEP*, 08:071, 2010. [Erratum: JHEP03,092(2014)].

[38] M. Lüscher. Trivializing maps, the Wilson flow and the HMC algorithm. *Commun. Math. Phys.*, 293:899–919, 2010.

[39] W. Magnus. On the exponential solution of differential equations for a linear operator. *Communications on Pure and Applied Mathematics*, 7(4):649–673, 1954.

[40] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[41] C. Moler and C. V. Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later, 2003.

[42] I. Montvay and G. Münster. *Quantum fields on a lattice.* Cambridge monographs on mathematical physics. Cambridge Univ. Press, Cambridge, 1994.

[43] C. Morningstar and M. Peardon. Analytic Smearing of SU(3) Link Variables

in Lattice QCD. *Phys.Rev. D69 (2004) 054501*, 2003.

[44] M. Muniz. Geometrische Integration unter Verwendung der Cayley-Transformation. Bachelor Thesis, Bergische Universitaet Wuppertal, Gaussstrasse 20, D-42119 Wuppertal, March 2016.

[45] H. Munthe-Kaas. Runge-Kutta methods on Lie groups. *BIT*, 38(1):92–111, 1998.

[46] H. Munthe-Kaas. High order Runge-Kutta methods on manifolds. *APPL. NUMER. MATH*, 29:115–127, 1999.

[47] R. Narayanan and H. Neuberger. Infinite N phase transitions in continuum Wilson loop operators. *JHEP*, 03:064, 2006.

[48] S. Necco and R. Sommer. The Nf=0 heavy quark potential from short to intermediate distances. *Nucl. Phys.*, 622(B):328–346, 2002.

[49] B. Owren and A. Marthinsen. Runge–Kutta methods adapted to manifolds and based on rigid frames. *BIT*, 39(1):116–142, 1999.

[50] O. Philipsen. Lattice QCD at finite temperature and density. *Eur. Phys. J. ST*, 152:29–60, 2007.

[51] S. Pruess. Properties of splines in tension. *J. Approximation Theory*, 17:86–96, 1976.

[52] C. Reinsch. Smoothing by spline functions. *Numer. Math*, 10:177–183, 1967.

[53] P. Rentrop. An algorithm for the computation of the exponential spline. *Numer. Math.*, 35:81–93, 1980.

[54] P. Rentrop and U. Wever. Computational strategies for the tension parameters of the exponential spline. *In: Bulirsch R., Miele A., Stoer J., Well K.H. (eds), Lecture Notes in Control and Information Sciences*, 95, 1987.

[55] H. J. Rothe. *Lattice Gauge Theories: An Introduction*, volume 43 of *Lecture Notes in Physics*. World Scientific, 1992.

[56] Sanz-Serna. Geometric integration. *Inst. Math. Appl. Conf. Ser. New Ser., Oxford Univ. Press, New York*, 63:121–143, 1997.

[57] D. G. Schweikert. An interpolation curve using a spline in tension. *Journal of Mathematics and Physics*, 45:312–317, 1966.

[58] R. Sommer. A New way to set the energy scale in lattice gauge theories and its applications to the static force and α_s in SU(2) Yang-Mills theory. *Nucl. Phys.*, 411(B):839, 1994.

[59] H. Spaeth. Exponential spline interpolation. *Computing*, 4:225–233, 1969.

[60] M. Striebel, M. Günther, F. Knechtli, and M. Wandelt. Accuracy of Symmetric Partitioned Runge-Kutta Methods for Differential Equations on Lie-Groups. *PoS*, LATTICE2011:049, 2011.

[61] S. Varadarajan. *Lie Groups, Lie Algebras and Their Representations*, volume [III.4], [IV.6], [IV.8]. Prentice-Hall, Englewood Cliffs, New Jersey, 1974.

[62] M. Wandelt. Implicit partitioned Runge-Kutta integrators for simulations of gauge theories. Master's thesis, Bergische Universitaet Wuppertal, Gaussstrasse 20, D-42119 Wuppertal, April 2010.

[63] M. Wandelt and M. Günther. Efficient Numerical Simulation of the Wilson Flow in Lattice QCD. *Special Issue ECMI2014 of the Springer Journal of Mathematics in Industry*, 2016.

[64] M. Wandelt, M. Günther, F. Knechtli, and M. Striebel. Symmetric partitioned Runge-Kutta methods for differential equations on Lie groups. *Applied Numerical Mathematics*, 62:1740–1748, 2012.

[65] M. Wandelt, M. Günther, and M. Muniz. Geometric integration on Lie groups using the Cayley transform with focus on Lattice QCD. *Journal of Computational and Applied Mathematics*, (112495), 2019.

[66] M. Wandelt, F. Knechtli, and M. Günther. The Wilson flow and the finite temperature phase transition. *Journal of High Energy Physics*, 2016(10):61, Oct 2016.

[67] S. Weinberg. *Lectures on Quantum Mechanics.* Cambridge University Press, 2 edition, 2015.

[68] P. Werneburg. Determination of the Price-Load Curve Using Smoothing Splines Under Tension. Master's thesis, Bergische Universitaet Wuppertal, Gaussstrasse 20, D-42119 Wuppertal, October 2008.

[69] U. Wolff [ALPHA Collaboration]. Monte Carlo errors with less errors. *Comput. Phys. Commun.*, 2004.

[70] A. Zanna, K. Engø, and H. Munthe-Kaas. Adjoint and selfadjoint Lie-group methods. *BIT*, 41(2):395–421, 2001.

Acknowledgements

I would like to thank many people supporting this thesis.

First, I thank my supervisors Prof. Dr. Michael Günther and Prof. Dr. Francesco Knechtli for their support during writing this thesis. In the course of my Master's studies in CSIS they raised my interest in numerical integration and in Lattice QCD leading to a Master thesis in this field followed by this thesis.

I also thank Prof. Dr. Michael Striebel for the cooperation concerning symmetric partitioned Runge-Kutta methods, the time-reversible projection methods as well as sharing his code on projection methods. Moreover, I thank Dr. Björn Leder and Dr. Jacob Finkenrath for their support in the topic of the Wilson flow and its implementation. They guided me through the (DDHMC) code and helped me realizing my own parts of the code on the cluster. Furthermore, I thank Dr. Roman Höllwieser for reading my thesis and his valuable comments. Additionally, I thank Prof. Dr. Martin Lüscher for the permission of using his DDHMC code and Prof. Dr. Ulli Wolff for the provison of a routine computing the autocorrelation error.

Next, I thank all (present and former) colleagues of the working group "Applied Mathematics / Numerical Analysis" for the warm and friendly atmosphere in the group. Last but not least I thank my family for their support.

Finally, I thank the DFG. My position was partly funded by the SFB/TRR 55 "Hadron Physics from Lattice QCD".

10 Chapter 10 Appendix

.1 Similarity Transformations

In this part, we list the similarity transformations for traces.

For square matrices U, V, W it holds

$$\begin{aligned} Tr\big(U+V\big) = & \qquad\qquad Tr(U) + Tr(V) & (.1) \\ Tr\big(UV\big) = Tr\big(VU\big) & & (.2) \end{aligned}$$

and for unitary matrices U

$$\mathrm{Re}\big(Tr\big(U\big)\big) = \frac{1}{2}Tr\big(U + U^\dagger\big) \tag{.3}$$

Proof. The first two equations are obvious, and equation (.3) follows from

$$\begin{aligned} Tr\big(U + U^\dagger\big) &= Tr\big(\mathrm{Re}(U) + \mathrm{Im}(U) + \mathrm{Re}(U) - \mathrm{Im}(U)\big) \\ &= Tr\big(2 \cdot \mathrm{Re}(U)\big) = 2 \cdot Tr\big(\mathrm{Re}(U)\big) \end{aligned}$$

□

.2 Example: Deriving Hamiltonian Equations of Motion for SU(2)

This section expands section 2.2.3. We compute the Hamiltonian equations of motion for the momenta $P_{x,\mu}$ in the Lie group $SU(2)$. The aim of this computation is curiosity:

- How is the theoretical formula $\partial_{x,\mu} f(U)$ related to the formulas used for computation of, for example, $\dot{P}_{x,\mu}$ or $Z(V_{x,\mu})$?
- Is it really quite easy to compute the derivatives of functions on Lie groups with respect to Lie group elements?

The Derivatives of the Wilson Gauge Action. We compute the derivatives $\partial_{x,\mu}$ of $f(U) = -S_G(U)$ for the Lie group $SU(2)$ according to formula (2.33). Here, we use the generators $T^1 = \frac{i}{2}\sigma_1$, $T^2 = \frac{i}{2}\sigma_2$, $T^3 = \frac{i}{2}\sigma_3$ with Pauli matrixes

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} , \qquad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} , \qquad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} .$$

The generators T^a of the Lie algebra are chosen in such a way that it holds $\mathrm{Tr}(T^a T^b) = -\frac{1}{2}\delta$ (see (A.2) in [38]). We start with $U_{x,\mu}$ and $\mathcal{S}(U_{x,\mu})$. These matrices have the initial values

$$U_{x,\mu} = \begin{pmatrix} u_4 + iu_3 & u_2 + iu_1 \\ -u_2 + iu_1 & u_4 - iu_3 \end{pmatrix} , \qquad \mathcal{S}(U_{x,\mu}) = \begin{pmatrix} v_4 + iv_3 & v_2 + iv_1 \\ -v_2 + iv_1 & v_4 - iv_3 \end{pmatrix}$$

according to lemma 7.1. (For the Lie group $SU(2, \mathbb{C})$, it can be easily shown, that the staples $\mathcal{S}(U_{x,\mu})$ is in $SU(2, \mathbb{C})$.)

It follows $W_{x,\mu} = U_{x,\mu} \cdot \mathcal{S}(U_{x,\mu})$ with

$$W_{x,\mu} = \begin{pmatrix} w_4 + iw_3 & w_2 + iw_1 \\ -w_2 + iw_1 & w_4 - iw_3 \end{pmatrix}$$

and

$$\begin{aligned} w_4 &= u_4v_4 - u_1v_1 - u_2v_2 - u_3v_3 & \qquad w_3 &= u_3v_4 + u_4v_3 + u_2v_1 - u_1v_2 \\ w_2 &= u_4v_2 - u_3v_1 + u_2v_4 + u_1v_3 & \qquad w_1 &= u_3v_2 + u_4v_1 - u_2v_3 + u_1v_4 \end{aligned}$$

in order that $M_{x,\mu} := U_{x,\mu}\mathcal{S}(U_{x,\mu}) - \mathcal{S}(U_{x,\mu})^\dagger U_{x,\mu}^\dagger = W_{x,\mu} - W_{x,\mu}^\dagger$ looks like

$$\begin{aligned} M_{x,\mu} &= \begin{pmatrix} w_4 + iw_3 & w_2 + iw_1 \\ -w_2 + iw_1 & w_4 - iw_3 \end{pmatrix} - \begin{pmatrix} w_4 - iw_3 & -w_2 - iw_1 \\ w_2 - iw_1 & w_4 + iw_3 \end{pmatrix} \\ &= \begin{pmatrix} w_4 + iw_3 - (w_4 - iw_3) & w_2 + iw_1 - (-w_2 - iw_1) \\ -w_2 + iw_1 - (w_2 - iw_1) & w_4 - iw_3 - (\ w_4 + iw_3) \end{pmatrix} \\ &= \begin{pmatrix} 2iw_3 & 2w_2 + 2iw_1 \\ -2w_2 + 2iw_1 & -2iw_3 \end{pmatrix} . \end{aligned}$$

Using equation (2.39), we have

$$\partial^a_{x,\mu} S(U) = -\mathrm{Tr}\big\{T^a M_{x,\mu}\big\} = -\mathrm{Tr}\left\{T^a \cdot 2 \begin{pmatrix} iw_3 & w_2 + iw_1 \\ -w_2 + iw_1 & -iw_3 \end{pmatrix}\right\}$$

with

$$
\begin{aligned}
\partial^1_{x,\mu} S(U) &= -\mathrm{Tr}\big\{T^1 M_{x,\mu}\big\} \\
&= -\mathrm{Tr}\left\{ \frac{i}{2} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot 2 \begin{pmatrix} iw_3 & w_2 + iw_1 \\ -w_2 + iw_1 & -iw_3 \end{pmatrix} \right\} \\
&= -i \cdot \mathrm{Tr}\left\{ \begin{pmatrix} w_2 + iw_1 & -iw_3 \\ iw_3 & -w_2 + iw_1 \end{pmatrix} \right\} \\
&= -i \cdot \big(w_2 + iw_1 + (-w_2) + iw_1 \big) \\
&= 2w_1 \, , \\
\partial^2_{x,\mu} S(U) &= -\mathrm{Tr}\big\{T^2 M_{x,\mu}\big\} \\
&= -\mathrm{Tr}\left\{ \frac{i}{2} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 2iw_3 & 2w_2 + 2iw_1 \\ -2w_2 + 2iw_1 & -2iw_3 \end{pmatrix} \right\} \\
&= -i \cdot \mathrm{Tr}\left\{ \begin{pmatrix} w_1 + iw_2 & -w_3 \\ -w_3 & -w_1 + iw_2 \end{pmatrix} \right\} \\
&= -i \cdot \big(w_1 + iw_2 + (-w_1) + iw_2 \big) \\
&= 2w_2 \, ,
\end{aligned}
$$

$$
\begin{aligned}
\partial^3_{x,\mu} S(U) &= -\mathrm{Tr}\big\{T^3 M_{x,\mu}\big\} \\
&= -\mathrm{Tr}\left\{ \frac{i}{2} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \cdot 2 \begin{pmatrix} iw_3 & w_2 + iw_1 \\ -w_2 + iw_1 & -iw_3 \end{pmatrix} \right\} \\
&= -i \cdot \mathrm{Tr}\left\{ \begin{pmatrix} iw_3 & w_2 + iw_1 \\ w_2 - w_1 & iw_3 \end{pmatrix} \right\} \\
&= -i \cdot \big(iw_3 + iw_3 \big) \\
&= 2w_3 \, .
\end{aligned}
$$

For simplicity, the factor $\frac{\beta}{N}$ is left out in the whole computation. At the end, we receive the result:

$$
\begin{aligned}
\frac{\partial}{\partial U_{x,\mu}} f(U) = \partial_{x,\mu} f(U) &= T^a \partial^a_{x,\mu} f(U) = -T^a \partial^a_{x,\mu} S(U) \\
&= -T^1 \cdot 2w_1 - T^2 \cdot 2w_2 - T^3 \cdot 2w_3 \\
&= -\frac{i}{2} \left\{ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot 2w_1 + \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \cdot 2w_2 + \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \cdot 2w_3 \right\} \\
&= -i \cdot \begin{pmatrix} w_3 & w_1 - iw_2 \\ w_1 + iw_2 & -w_3 \end{pmatrix} \\
&= - \begin{pmatrix} iw_3 & w_2 + iw_1 \\ -w_2 + iw_1 & -iw_3 \end{pmatrix} \\
&= -\frac{1}{2} M_{x,\mu}
\end{aligned}
\tag{.4}
$$

which coincides with the traceless and anti-hermitian operator $\{M_{x,\mu}\}_{TA}$ because the trace of $M_{x,\mu}$ is zero (in the Lie group $SU(2,\mathbb{C})$).

Derivation via Hamiltonian equations of motion. The following derivative is given in [62], the derivation is described in [23] and chapter 7.2.3 in [16]:

$$\begin{aligned} i\dot{P}_{x.\mu} &= \frac{i^2}{2}\Big(U_{x,\mu}\mathcal{S}(U_{x,\mu}) - \mathcal{S}(U_{x,\mu})^\dagger U_{x,\mu}^\dagger\Big) - \frac{i^2}{2N}\mathrm{Tr}\Big(U_{x,\mu}\mathcal{S}(U_{x,\mu}) - \mathcal{S}(U_{x,\mu})^\dagger U_{x,\mu}^\dagger\Big)\cdot I_N \\ &= \frac{i^2}{2}\Big(M_{x,\mu}\Big) - \frac{i^2}{2N}\mathrm{Tr}\Big(M_{x,\mu}\Big)\cdot I_N \\ &= -\frac{1}{2}M_{x,\mu} \end{aligned} \tag{.5}$$

Note: the result for $SU(2)$ matrices coincides with the theoretical result for $SU(2)$ matrices.

Formulae in Literature [38] Additionally, the formula (5.3) combined with (5.4) in [38] give the same result:

$$\begin{aligned} Z\big(V(t)\big) &= -\frac{\partial}{\partial\mathcal{S}(U_{x,\mu})(t)} S\big(\{V(t)\}\big) \\ &= -\sum_{\mu\neq\nu}\mathcal{P}\Big\{\mathcal{S}(U_{x,\mu})(t)\; V_{x+\hat{\mu},\nu}(t)\; V^{-1}_{x+\hat{\nu},\mu}(t)\; V^{-1}_{x,\nu}(t) \\ &\qquad\qquad + \;\mathcal{S}(U_{x,\mu})(t)\; V^{-1}_{x+\hat{\nu}-\hat{\mu},\mu}(t)\; V^{-1}_{x-\hat{\nu},\mu}(t)\; V_{x-\hat{\nu},\nu}(t)\Big\} \end{aligned}$$

given in [38]. Here, $\hat{\mu}$ and $\hat{\nu}$ denote the unit vectors in direction μ and ν and

$$\mathcal{P}\{M\} = \frac{1}{2}(M - M^\dagger) - \frac{1}{6}tr(M - M^\dagger)$$

projects any matrix M of size 3×3 to the Lie algebra $\mathfrak{su}(3,\mathbb{C})$.

Notation adapted to the previous one:

- $V(t)$ means U
- $V_{x+\hat{\mu},\nu}(t)\; V^{-1}_{x+\hat{\nu},\mu}(t)\; V^{-1}_{x,\nu}(t) + V^{-1}_{x+\hat{\nu}-\hat{\mu},\mu}(t)\; V^{-1}_{x-\hat{\nu},\mu}(t)\; V_{x-\hat{\nu},\nu}(t)$ is the part of the staple $\mathcal{S}(U_{x,\mu})$ in direction (μ,ν), the whole staple is the sum of the parts over all possible directions (μ,ν) and (ν,μ)
- $\sum_{\mu\neq\nu}\mathcal{P}\{U_{x,\mu}\mathcal{S}(U_{x,\mu})\} = \frac{1}{2}M_{x,\mu}$

It follows

$$\begin{aligned}
Z\big(U(t)\big) &= -\frac{\partial}{\partial U_{x,\mu}(t)} S\big(\{U(t)\}\big) \\
&= -\sum_{\mu\neq\nu} \mathcal{P}\Big\{ U_{x,\mu}\, U_{x+\hat\mu,\nu}\, U^{-1}_{x+\hat\nu,\mu}\, U^{-1}_{x,\nu} + \, U_{x,\mu}\, U^{-1}_{x+\hat\nu-\hat\mu,\mu}\, U^{-1}_{x-\hat\nu,\mu}\, U_{x-\hat\nu,\nu} \Big\} \\
&= -\sum_{\mu\neq\nu} \mathcal{P}\Big\{ U_{x,\mu} \mathcal{S}(U_{x,\mu}) \Big\} \\
&= -\frac{1}{2} M_{x,\mu} \qquad (.6)
\end{aligned}$$

Comparison Finally, the result $\partial/\partial U_{x,\mu} f(U)$ given in (.4) coincides with

- e.g. Z(U) given as formula (5.3) in [38] (see equation (.6))
- or equation (1.8) in [62], derived like equation (7.4.5) of [16] (see equation (.5) here).

Thus, the computation of a function with respect to a Lie group element is straightforward.

.3 Exponential Smoothing Spline

In this section, the coefficients of the exponential smoothing spline

$$s(x) = s_{i+1} t + s_i (1-t) + \frac{d_{i+1}}{\lambda_i^2} \left(\frac{\sinh\big(\mu_i t\big)}{\sinh(\mu_i)} - t \right) + \frac{d_i}{\lambda_i^2} \left(\frac{\sinh(\mu_i (1-t))}{\sinh(\mu_i)} - 1 + t \right) \qquad (6.8)$$

are determined. They are computed as solution (s, d) of the linear equations

$$Qs = Td \qquad (6.11)$$

$$\text{and} \quad Us - Q^T d = pD^{-2}(s-y) \qquad (6.14)$$

which follow from the smoothing condition

$$s'(x_i^-) - s'(x_i^+) = 0, \qquad i = 1, \ldots, n-1, \qquad (6.9b)$$

and the jump equation

$$\Big(s'''(x_i^-) - \Lambda^2(x_i^-) s'(x_i^-) \Big) - \Big(s'''(x_i^+) - \Lambda^2(x_i^+) s'(x_i^+) \Big) = 2p \frac{s(x_i) - y_i}{w_i^2} \qquad (6.9d)$$

for $i = 0, \ldots, n$.

The linear equations (6.11) and (6.14) are computed as follows.

Preliminary Considerations. First, we compute the derivatives of the exponential smoothing spline (6.8). Here, we use $t' = \frac{1}{h_i}$ and $(\mu_i t)' = \lambda_i$ according to the notations

$$t := \frac{x - x_i}{x_{i+1} - x_i}, \quad \mu_i := \lambda_i \cdot h_i, \quad h_i := x_{i+1} - x_i \tag{6.7}$$

and get

$$s'(x) = \frac{s_{i+1} - s_i}{h_i} + \frac{d_{i+1}}{\lambda_i}\left(\frac{\cosh(\mu_i t)}{\sinh(\mu_i)} - \frac{1}{\mu_i}\right) + \frac{d_i}{\lambda_i}\left(-\frac{\cosh(\mu_i(1-t))}{\sinh(\mu_i)} + \frac{1}{\mu_i}\right), \tag{.7}$$

$$s''(x) = d_{i+1}\frac{\sinh(\mu_i t)}{\sinh(\mu_i)} + d_i\frac{\sinh(\mu_i(1-t))}{\sinh(\mu_i)}, \tag{.8}$$

$$s'''(x) = d_{i+1}\lambda_i\frac{\cosh(\mu_i t)}{\sinh(\mu_i)} - d_i\lambda_i\frac{\cosh(\mu_i(1-t))}{\sinh(\mu_i)}. \tag{.9}$$

For x_i^-, i.e., the right-hand side of the left interval, it holds $t = 1$. On the other hand, the left-hand side of the right interval is called x_i^+ and implies $t = 0$. Together with $\sinh(0) = 0$ and $\cosh(0) = 1$, we get

$$s'(x_i^+) = \frac{s_{i+1} - s_i}{h_i} + \frac{d_{i+1}}{\lambda_i}\left(\frac{1}{\sinh(\mu_i)} - \frac{1}{\mu_i}\right) + \frac{d_i}{\lambda_i}\left(-\frac{\cosh(\mu_i)}{\sinh(\mu_i)} + \frac{1}{\mu_i}\right), \tag{.10}$$

$$s''(x_i^+) = d_i\,, \tag{.11}$$

$$s'''(x_i^+) = d_{i+1}\lambda_i\frac{1}{\sinh(\mu_i)} - d_i\lambda_i\frac{\cosh(\mu_i)}{\sinh(\mu_i)} \tag{.12}$$

and

$$\begin{aligned} s'(x_i^-) = \frac{s_i - s_{i-1}}{h_{i-1}} &+ \frac{d_i}{\lambda_{i-1}}\left(\frac{\cosh(\mu_{i-1})}{\sinh(\mu_{i-1})} - \frac{1}{\mu_{i-1}}\right) \\ &+ \frac{d_{i-1}}{\lambda_{i-1}}\left(\frac{-1}{\sinh(\mu_{i-1})} + \frac{1}{\mu_{i-1}}\right), \end{aligned} \tag{.13}$$

$$s''(x_i^-) = d_i\,, \tag{.14}$$

$$s'''(x_i^-) = d_i\lambda_{i-1}\frac{\cosh(\mu_{i-1})}{\sinh(\mu_{i-1})} - d_{i-1}\lambda_{i-1}\frac{1}{\sinh(\mu_i)}. \tag{.15}$$

First Linear Equation. Now, we set

$$t_i := -\frac{1}{\lambda_i}\frac{\cosh(\mu_i)}{\sinh(\mu_i)} + \frac{1}{\lambda_i\mu_i} \quad \text{and} \quad \bar{t}_i := \frac{1}{\lambda_i}\frac{1}{\sinh(\mu_i)} - \frac{1}{\lambda_i\mu_i} \tag{.16}$$

in order that equations (.13) and (.10) simplify to

$$s'(x_i^+) = \frac{s_{i+1} - s_i}{h_i} + d_{i+1}\bar{t}_i + d_i t_i \tag{.17}$$

$$\text{and} \quad s'(x_i^-) = \frac{s_i - s_{i-1}}{h_{i-1}} - d_i t_{i-1} - d_{i-1}\bar{t}_{i-1}\,. \tag{.18}$$

Then, the smoothing condition (6.9b) leads to

$$\frac{s_i - s_{i-1}}{h_{i-1}} - \frac{s_{i+1} - s_i}{h_i} = d_i t_{i-1} + d_{i-1}\bar{t}_{i-1} + d_{i+1}\bar{t}_i + d_i t_i \quad (.19)$$

$$\Leftrightarrow -\frac{1}{h_{i-1}} s_{i-1} + \Big(\frac{1}{h_{i-1}} + \frac{1}{h_i}\Big) s_i - \frac{1}{h_i} s_{i+1} = d_{i-1}\bar{t}_{i-1} + d_i(t_{i-1} + t_i) + d_{i+1}\bar{t}_i \quad (.20)$$

for $i = 0, \dots, n-1$. Here, the left-hand side can be denoted as Qs with tridiagonal matrix Q of size $(n-1) \times (n+1)$ with non-zero entries

$$Q_{i,i} = -\frac{1}{h_{i-1}}, \qquad Q_{i,i+1} = \frac{1}{h_{i-1}} + \frac{1}{h_i}, \qquad Q_{i,i+2} = -\frac{1}{h_i} \quad (6.12)$$

for $i = 0, \dots, n-1$. Besides, the right-hand side of equation (.20) can be described as Td with symmetric tridiagonal matrix T of order $n-1$ and entries

$$T_{i,i} = t_{i-1} + t_i, \qquad T_{k,k+1} = T_{k+1,k} = \bar{t}_k. \quad (6.13)$$

The indices i and k run from $i = 1, \dots, n-1$ and $k = 1, \dots, n-2$. Finally, we yield the linear equation $Qs = Td$ given by equation (6.11).

Second Linear Equation. The jump equation (6.9d), i.e.,

$$2p\frac{s(x_i) - y_i}{w_i^2} = \Big(s'''(x_i^-) - \Lambda^2(x_i^-)s'(x_i^-)\Big) - \Big(s'''(x_i^+) - \Lambda^2(x_i^+)s'(x_i^+)\Big)$$

includes the functions $s'''(x_i^-)$, $s'''(x_i^+)$, $\Lambda^2(x_i^-)s'(x_i^-)$ and $\Lambda^2(x_i^+)s'(x_i^+)$. With equations (.17) and (.18) and

$$\Lambda(x_i^-) = \lambda_{i-1} \quad \text{and} \quad \Lambda(x_i^+) = \lambda_i\,, \quad (.21)$$

it reads

$$\Lambda^2(x_i^-)s'(x_i^-) = \lambda_{i-1}^2\bigg(\frac{s_i - s_{i-1}}{h_{i-1}} - d_i t_{i-1} - d_{i-1}\bar{t}_{i-1}\bigg) \quad (.22)$$

$$\text{and} \quad \Lambda^2(x_i^+)s'(x_i^+) = \lambda_i^2\bigg(\frac{s_{i+1} - s_i}{h_i} + d_{i+1}\bar{t}_i + d_i t_i\bigg). \quad (.23)$$

Using the abbreviations t_i and $\bar{t}_i$ of equation (.16), $s'''(x_i^-)$ and $s'''(x_i^+)$ can be denoted as

$$s'''(x_i^-) = d_i\Big(-\lambda_{i-1}^2 t_{i-1} + \frac{1}{h_{i-1}}\Big) - d_{i-1}\Big(\lambda_{i-1}\bar{t}_{i-1} + \frac{1}{h_{i-1}}\Big) \quad (.24)$$

$$\text{and} \quad s'''(x_i^+) = d_{i+1}\Big(\lambda_i^2\bar{t}_i + \frac{1}{h_i}\Big) - d_i\Big(\lambda_i^2 t_i - \frac{1}{h_i}\Big)\,. \quad (.25)$$

So, the differences $\Big(s'''(x_i^-)-\Lambda^2(x_i^-)s'(x_i^-)\Big)$ and $\Big(s'''(x_i^+)-\Lambda^2(x_i^+)s'(x_i^+)\Big)$ simplify to

$$\begin{aligned}\Big(s'''(x_i^-)-\Lambda^2(x_i^-)s'(x_i^-)\Big) &= d_i\Big(-\lambda_{i-1}^2 t_{i-1}+\frac{1}{h_{i-1}}\Big)-d_{i-1}\Big(\lambda_{i-1}^2\bar{t}_{i-1}+\frac{1}{h_{i-1}}\Big)\\ &\quad-\Big(\lambda_{i-1}^2\Big(\frac{s_i-s_{i-1}}{h_{i-1}}-d_i t_{i-1}-d_{i-1}\bar{t}_{i-1}\Big)\Big)\\ &=\frac{1}{h_{i-1}}d_i-\frac{1}{h_{i-1}}d_{i-1}-\frac{\lambda_{i-1}^2}{h_{i-1}}s_i+\frac{\lambda_{i-1}^2}{h_{i-1}}s_{i-1} \qquad (.26)\end{aligned}$$

and

$$\begin{aligned}\Big(s'''(x_i^+)-\Lambda^2(x_i^+)s'(x_i^+)\Big) &= d_{i+1}\Big(\lambda_i^2\bar{t}_i+\frac{1}{h_i}\Big)-d_i\Big(\lambda_i^2 t_i-\frac{1}{h_i}\Big)\\ &\quad-\Big(\lambda_i^2\Big(\frac{s_{i+1}-s_i}{h_i}+d_{i+1}\bar{t}_i+d_i t_i\Big)\Big)\\ &=\frac{1}{h_i}d_{i+1}+\frac{1}{h_i}d_i-\frac{\lambda_i^2}{h_i}s_{i+1}+\frac{\lambda_i^2}{h_i}s_i\,. \qquad (.27)\end{aligned}$$

At the end, the equation

$$\begin{aligned}2p\frac{s(x_i)-y_i}{w_i^2} &= \Big(s'''(x_i^-)-\Lambda^2(x_i^-)s'(x_i^-)\Big)-\Big(s'''(x_i^+)-\Lambda^2(x_i^+)s'(x_i^+)\Big)\\ &=\frac{1}{h_{i-1}}d_i-\frac{1}{h_{i-1}}d_{i-1}-\frac{\lambda_{i-1}^2}{h_{i-1}}s_i+\frac{\lambda_{i-1}^2}{h_{i-1}}s_{i-1}-\Big(\frac{1}{h_i}d_{i+1}+\frac{1}{h_i}d_i-\frac{\lambda_i^2}{h_i}s_{i+1}+\frac{\lambda_i^2}{h_i}s_i\Big)\\ &=-\frac{1}{h_{i-1}}d_{i-1}+\Big(\frac{1}{h_{i-1}}-\frac{1}{h_i}\Big)d_i-\frac{1}{h_i}d_{i+1}+\frac{\lambda_{i-1}^2}{h_{i-1}}s_{i-1}-\Big(\frac{\lambda_{i-1}^2}{h_{i-1}}+\frac{\lambda_i^2}{h_i}\Big)s_i+\frac{\lambda_i^2}{h_i}s_{i+1}\end{aligned} \qquad (.28)$$

for $i=0,\dots,n$ is reached. Also here, the predefinitions $s'''(x_0^-)=\Lambda(x_0^-)=0$ and $s'''(x_n^+)=\Lambda(x_n^+)=0$ have to be used. Finally, we define

$$u_i := \frac{\lambda_i^2}{h_i} \quad \text{and} \quad v_j := u_j+u_{j+1}\,. \qquad (.29)$$

Using

$$U := \begin{pmatrix} -u_1, & u_1, & & & 0\\ u_1, & -v_1, & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & -v_{n-2}, & u_{n-1}\\ 0 & & & u_{n-1}, & -u_{n-1}\end{pmatrix}, \qquad (.30)$$

equation (.28) can be rewritten in matrix notation as

$$Us-Q^Td=pD^{-2}(s-y)$$

which coincides with equation (6.14).