Time-adaptive Semi-Lagrangian Approximation to Hamilton-Jacobi-Bellman Equation

Ilja Kalmykov

Geboren am 20. Juli 1986 in Prochladny (RU)

1st September, 2016

Bachelorarbeit Mathematik

Betreuer: Prof. Dr. Jochen Garcke

Zweitgutachter: Prof. Dr. Ira Neitzel

INSTITUT FÜR NUMERISCHE SIMULATION

MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT DER

RHEINISCHEN FRIEDRICH-WILHELMS-UNIVERSITÄT BONN

моим бабушкам и дедушкам

Contents

1 Introduction 1

2 Preliminaries 4
2.1 Method of characteristics 4
2.2 Hamilton-Jacobi equation 5
2.2.1 Calculus of variations 5
2.2.2 Hamilton-Jacobi equation 6
2.3 Optimal control problem 7
2.3.1 Controlled system 7
2.3.2 Cost functional 8

3 Semi-Lagrangian schemes 10
3.1 SL schemes for the optimal control problems 10
3.2 Examples of SL schemes 11

4 Sparse grids 13

5 Error estimation and step size control for Semi-Lagrangian schemes 17
5.1 Approximation error of SL schemes 17
5.1.1 Local and global error of numerical integration methods for ODEs 17
5.1.2 Error of quadrature 18
5.1.3 Properties of collocation methods 18
5.1.4 Local error of SL scheme 19
5.2 Step size control 20
5.2.1 Estimation of error for step size control 21

6 High order time methods for Semi-Lagrangian schemes 22
6.1 Structure preserving numerical integration for the SL schemes 22
6.2 Complexity of SL schemes with high order time discretization 22
6.2.1 Explicit time integration 23
6.2.2 Implicit time integration 25
6.2.3 Reduction of complexity 26
6.3 SL schemes with implicit midpoint rule 28
6.3.1 Composition methods 28
6.3.2 Quadrature with implicit midpoint rule 31
6.3.3 High order SL schemes with implicit midpoint rule 32
6.3.4 Optimization of the control and time inversion 33

7 Results of numerical experiments **34**
7.1 Control of the harmonic oscillator 34
7.1.1 Experiment parameters 34
7.1.2 Parameters of test environment 34
7.1.3 First experiments and modifications of test environment 36
7.1.4 Adaptive time steps 44
7.2 Control of the semi-discrete wave equation 48

8 Software **50**
8.1 SG++ 50
8.2 Automatization of numerical tests 50
8.3 OpenMP 50

1 Introduction

This work investigates the application of Semi-Lagrangian (SL) approximation schemes to the optimal control problems with emphasis on the time discretization aspects.

The basic idea of the SL schemes dates back to the work of Courant, Isaacson and Rees in [CIR52] and has a strong connection to the method of characteristics. For a given spatial node, the solution in each iteration is computed by searching for the foot of the characteristic curve, which passes through this node. The new value is then obtained by evaluating the solution from last time step at the calculated foot point.

Algorithms, which are based on the SL scheme, require an interaction of multiple approximation techniques. The most important building blocks are spatial reconstruction, numerical integration and quadrature.

The numerical experiments in this work are based on optimal control problems. Examples are minimization of fuel consumption for aeroplanes or maximization of utility of the spendings for a household. Another non-trivial example is the free kick in a football game. An optimal control problem is aimed at finding control law for a given system, which is sufficient with respect to some cost function.

The mathematical description of many optimal control problems can be formulated in the framework of the Hamilton-Jacobi-Bellman (HJB) equation. This approach yields a sufficient condition for the optimality of the control. The minimal costs can be characterized as the unique viscosity solution of the HJB equation (see [FF11]).

In order to overcome the complexity of the spatial discretization for high-dimensional control problems, the usage of an appropriate reconstruction technique is important. This work is based on the publication [GK15], where SL schemes with sparse grids were investigated. The sparse grids interpolation allows to reduce significantly the rise of complexity due to the curse of dimension ([BG04, Gar13]).

The numerical experiments for the control of a harmonic oscillator respectively semi-discrete wave equation in 4 dimension were adapted from [GK15] and analysed with respect to the time discretization.

The ability of SL schemes to advance in time with larger steps (compared to e.g. finite differences discretization, see [Bon04, FF11]) motivates the investigation of numerical integrators with high order and step size control. [FF94] gives examples of SL approximations based on Runge-Kutta methods up to order 4. The disadvantage of these schemes is the high complexity of the optimization problem, which is exponential in the number of stages.

To overcome this problem, SL schemes based on the diagonal implicit symplectic Runge-Kutta (DISRK) methods were proposed and investigated in this work. This approach allows to utilize the Belmann's Dynamic Programming Principle within the single time step and leads to the complexity of the optimization for the control, which is linear in the number of stages. The drawback is that the value function has to be tracked within the Rungke-Kutta step. The reconstruction is required for each single stage.

The time-adaptive SL schemes in this work are based on the step size control methods for ordinary differential equations. The investigation is motivated by [FF06]. Effects of the adaptive time steps on the global approximation error and efficiency of the controller have been analysed.

The structure of this theses is as follows. Chapter 2 recalls some concepts, which are important for the understanding of the SL methods and the HJB equation. In addition, the basic principles of the optimal control are briefly described. In the chapter 3, a detailed presentation of the SL schemes with some examples is given. Chapter 4 provides a short introduction to the sparse grids. Chapter 5 address the ideas of time step size control for the numerical integration and discusses its application to the SL schemes. In chapter 6 SL schemes based on DISRK methods are derived. Finally, chapter 7 presents the results of numerical experiments. Some issues regarding implementation and software being used are described in chapter 8.

Acknowledgements I would like to thank my supervisor for giving me a chance to investigate such a challenging and interesting topic. In addition, my special thanks go to my family and 超级诗鹿 for their constant support, understanding and motivation.

Zusammenfassung Diese Arbeit untersucht die Auswirkung von der Zeitdiskretisierung auf den globalen Fehler der Semi-Lagrange (SL) Approximation für die Lösung von der Hamilton-Jacobi-Bellman Gleichung. Die Struktur der auf den Runge-Kutta Methoden basierenden SL Verfahren wurde untersucht und neue Approximationschema mit verbesserter Ordnung bzgl. der Zeitschrittweite und geringeren Komplexität vorgeschlagen. Darüberhinaus wurden Methoden mit einer adaptiven Zeitschrittweiten implementiert und getestet.

2 Preliminaries

The idea of the Semi-Lagrangian scheme is derived from the method of characteristics for the solution of PDEs. Hence, some elementary aspects of this approach are recalled in this chapter. In addition, an introduction to the Hamilton-Jacobi equation is provided. This equation can be considered as the basis for the optimal control problem and the Hamilton-Jacobi-Bellman equation.

2.1 Method of characteristics

This section gives a brief introduction to the method of characteristics. For a more detailed analysis see e.g. [Eva98].

Consider a first order non-linear PDE with a boundary condition:

$$\begin{aligned} F(Du, u, x) &= 0 \qquad (2.1)\\ u &= g \text{ on } \Gamma, \end{aligned}$$

$x \in \Omega \subset \mathbb{R}^n$, Ω open, $u : \overline{\Omega} \to \mathbb{R}$ and $F : \mathbb{R}^n \times \mathbb{R} \times \overline{\Omega} \to \mathbb{R}$, $\Gamma \subset \partial\Omega$, $g : \Gamma \to \mathbb{R}$. F, g are assumed to be smooth functions, i.e. $F, g \in C^\infty$.

The basic idea of the method of characteristics is to find the value $u(x)$ of the solution to eq. (2.1) by tracking a curve $\gamma \subset \Omega$ from x back to some point $x_0 \in \Gamma$. In this context, γ is called the characteristic and x_0 the foot point of the characteristic. With this approach, the problem eq. (2.1) transforms to an ODE w.r.t. $\gamma(s)$ for some parameter $s \in I \subset \mathbb{R}$ as a new variable. The hope is to achieve an easier representation for the solution $u(x)$. The challenge is however to calculate $\gamma(s)$.

To choose $\gamma(s)$ so, that we are able to transform eq. (2.1) to an ODE let us first define some auxiliary variables

$$\begin{aligned} z(s) &\overset{\text{def}}{=} u(\gamma(s))\\ \mathbf{p}(s) &\overset{\text{def}}{=} Du(\gamma(s)), \end{aligned}$$

Now, for $\dot{z}$ and $\dot{p}$ we can write

$$\dot{z} = \sum_{j=1}^{n} u_{\gamma_j}\dot{\gamma}_j = \sum_{j=1}^{n} p_j\dot{\gamma}_j \qquad (2.2)$$

$$\dot{p}_i = \sum_{j=1}^{n} u_{\gamma_i\gamma_j}\dot{\gamma}_j\ , \forall i = 1, \dots, n \qquad (2.3)$$

We are looking for an ODE for $\gamma(s)$, which can be expressed in terms of u and Du, respectively z and p. With eq. (2.2), we already have on ODE characterising $z(s)$ as a function of p and γ. But it is not the case for $p(s)$, as eq. (2.3) depends on second derivative of u. Now, the idea is to use eq. (2.1) with $\gamma(s)$ as an independent variable to obtain new expression for $\dot{p}_i$. The derivation of eq. (2.1) w.r.t. γ_i yields:

$$\sum_{j=1}^{n} \frac{\partial F}{\partial \mathbf{p}_j}(\mathbf{p}, z, \gamma) u_{\gamma_j \gamma_i} + \frac{\partial F}{\partial \gamma_i}(\mathbf{p}, z, \gamma) + \frac{\partial F}{\partial z}(\mathbf{p}, z, \gamma) u_{\gamma_i} = 0$$

and

$$\sum_{j=1}^{n} \frac{\partial F}{\partial \mathbf{p}_j}(\mathbf{p}, z, \gamma) u_{\gamma_j \gamma_i} = -\frac{\partial F}{\partial \gamma_i}(\mathbf{p}, z, \gamma) - \frac{\partial F}{\partial z}(\mathbf{p}, z, \gamma) p_i$$

Following the eq. (2.3) we can choose

$$\dot{\gamma}_j = \frac{\partial F}{\partial \mathbf{p}_j}(\mathbf{p}, z, \gamma)$$

and obtain the following system of ODEs in vector notation(see [Eva98])

$$\begin{aligned}
\dot{p}(s) &= -D_x F(\mathbf{p}(s), z(s), \gamma(s)) - D_z F(\mathbf{p}(s), z(s), \gamma(s)) \cdot \mathbf{p}(s) \\
\dot{z}(s) &= D_p F(\mathbf{p}(s), z(s), \gamma(s)) \cdot p(s) \\
\dot{\gamma}(s) &= D_p F(\mathbf{p}(s), z(s), \gamma(s)) \qquad (2.4)
\end{aligned}$$

Equations (2.4) are called the characteristic system of the PDE (2.1). So, the solution of eq. (2.1) along the curve $\gamma(s)$ defined by the system (2.4) can be is defined by an ODE.

2.2 Hamilton-Jacobi equation

This sections introduces the Hamilton-Jacobi equation, some basic theoretical results regarding this equation as well as details, which are important for the numerical analysis.

2.2.1 Calculus of variations

Consider a function $\mathbf{w}(s) \in \mathcal{A}$ with $\mathcal{A}$ given by

$$\mathcal{A} = \left\{ \mathbf{w}(\cdot) \in C^2([0, t]; \mathbb{R}^n) | \mathbf{w}(0) = y, \mathbf{w}(t) = x \right\}$$

and an action functional

$$I[\mathbf{w}(\cdot)] = \int_0^t L(\dot{\mathbf{w}}(s), \mathbf{w}(s))\, ds$$

with $L : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$

$$L = L(q, x) = L(q_1, \ldots, q_n, x_1, \ldots, x_n) \quad (q, x \in \mathbb{R}^n)$$

L is assumed to be smooth function and usually referred to as Lagrangian.

In the calculus of the variations (see e.g. [Eva98]) we are looking for a solution to

$$I[\mathbf{x}(\cdot)] = \min_{\mathbf{w}(\cdot)\in\mathcal{A}} I[\mathbf{w}(\cdot)] \tag{2.5}$$

Assuming the solution for the eq. (2.5) exists, it satisfies the Euler-Lagrange equations

$$-\frac{d}{ds}(D_q L(\dot{\mathbf{x}}(s), \mathbf{x}(s))) + D_x L(\dot{\mathbf{x}}(s), \mathbf{x}(s)) = 0 \quad (0 \leqslant s \leqslant t) \tag{2.6}$$

Equation (2.6) is a system of n second-order ODEs. We can convert it into a first order system of $2 \cdot n$ equations. The resulting system is known as Hamilton's equations. To obtain this result we have to introduce new variables to substitute $\dot{\mathbf{x}}(s)$ and $\mathbf{x}(s))$. The following presentation is very similar to the method of characteristics. First set

$$\mathbf{p}(s) \overset{\text{def}}{=} D_q L(\dot{\mathbf{x}}(s), \mathbf{x}(s)) \quad (0 \leqslant s \leqslant t)$$

$\mathbf{p}(\cdot)$ is called the generalized momentum.

Now, the Hamiltonian H associated to the Lagrangian L is defined by

$$H(\mathbf{p}, \mathbf{x}) \overset{\text{def}}{=} \mathbf{p} \cdot \mathbf{q}(p, x) - L(\mathbf{q}(p, x), \mathbf{x}) \quad (\mathbf{p}, \mathbf{x} \in \mathbb{R}^n) \tag{2.7}$$

where $\mathbf{q}(p, x)$ is a given by the equation

$$\mathbf{p}(s) = D_q L(\mathbf{q}(p, x), x)$$

Please note that in this context $\mathbf{q}$ essentially represents $\dot{\mathbf{x}}(s)$.

Now, for $0 \leqslant s \leqslant t$, the functions $\mathbf{x}(s)$ and $\mathbf{p}(s)$ can be defined as solution of the Hamilton's equations

$$\begin{aligned} \dot{\mathbf{x}}(s) &= D_p H(\mathbf{p}(s), \mathbf{x}(s)) \\ \dot{\mathbf{p}}(s) &= -D_x H(\mathbf{p}(s), \mathbf{x}(s)) \end{aligned} \tag{2.8}$$

which are a system of $2 \cdot n$ first-order ODEs. For the equivalence to the Euler-Lagrange equations see [Eva98].

2.2.2 Hamilton-Jacobi equation

The Hamilton-Jacobi PDE is given by

$$u_t + H(Du, x) = 0 \tag{2.9}$$

with H defined by eq. (2.7). Following the notation from the section 2.1 with $\tilde{x} = (x, t)$, $\tilde{p} = (p, u_t)$ and $z = u$ we obtain

$$\tilde{p}_{n+1} + H(p, x) = F(\tilde{p}, z, \tilde{x}) = 0$$

and

$$\begin{aligned}\dot{z}(s) &= D_{\tilde{p}}F(\tilde{p}, z, \tilde{x}) = D_p H(\mathbf{p}(s), x(s)) \cdot \mathbf{p}(s) + \tilde{p}_{n+1}\\ &= D_p H(\mathbf{p}(s), x(s)) \cdot \mathbf{p}(s) - H(\mathbf{p}(s), x(s))\end{aligned}$$

for $z(s)$. The ODEs for $\mathbf{p}(s)$ and $\mathbf{x}(s)$ are given by

$$\left.\begin{aligned}\dot{\mathbf{p}}(s) &= -D_x H(\mathbf{p}(s), \mathbf{x}(s))\\ \dot{\mathbf{x}}(s) &= D_p H(\mathbf{p}(s), \mathbf{x}(s))\end{aligned}\right\}\text{Hamilton's equations}$$

Note that the characteristics of the Hamilton-Jacobi equation for $\mathbf{p}(s)$ and $\mathbf{x}(s)$ are the ODEs (2.8) derived from the calculus of variations problem (2.5). This indicates a strong connection between the Hamilton-Jacobi equation and the minimization of the action functional. An analogue relation can be obtained for the Hamilton-Jacobi-Bellman equation and the optimal control problem.

The equation for $z(s)$ can be computed straightforward from $\mathbf{p}(s)$ and $\mathbf{x}(s)$

$$\begin{aligned}\dot{z}(s) &= D_p H(\mathbf{p}(s), x(s)) \cdot \mathbf{p}(s) - H(\mathbf{p}(s), x(s))\\ &= \mathbf{p}(s)\left(\mathbf{q}(p, x) + \mathbf{p}(s) \cdot D_p \mathbf{q}(p, x) - \underbrace{D_q L(\mathbf{q}(p, x), \mathbf{x})}_{\mathbf{p}(s)} \cdot D_p \mathbf{q}(p, x)\right) - H(\mathbf{p}(s), x(s))\\ &= L(\mathbf{q}(p, x), \mathbf{x})\end{aligned}$$

Therefore we obtain

$$\min_{\mathbf{w}(\cdot)\in\mathcal{A}} I[\mathbf{w}(\cdot)] = z(t)$$

Consequently, the Hamilton-Jacobi equation provides a characterization of the minimization problem (2.5) in terms of a partial differential equation. See also [Eva98] for more details.

2.3 Optimal control problem

This section provides the necessary background for the treatment of the optimal control problems. The representation follows [FF11].

2.3.1 Controlled system

We consider a controlled dynamical system given by

$$\begin{cases}\dot{y}(s) = f(y(s), s, \alpha(s))\\ y(t_0) = x_0\end{cases} \tag{2.10}$$

with $x_0, y(s) \in \mathbb{R}^d$, and

$$\alpha : [t_0, T] \to A \subseteq \mathbb{R}^m, \qquad T \in \mathbb{R} \cup \{+\infty\},\ t_0 < T$$

For a measurable α the existence of the solution for eq. (2.10) can be derived from the Carathéodory theorem. Therefore, for every fixed control in the set of admissible controls

$$\alpha \in \mathcal{A} \overset{\text{def}}{=} \{\alpha : [t_0, T] \to A, \text{ measurable}\}$$

the existence of unique trajectory is guaranteed. In general, there is a set of solutions depending on α. For the sake of simplicity, let us consider autonomous dynamic in eq. (2.10) with $t_0 = 0$ and write x for x_0, respectively $y_x(s, \alpha)$ for the family of solutions starting from x.

2.3.2 Cost functional

Similar to the approach of the calculus of variations (see section 2.2.1) to select an optimal solution to eq. (2.10) we must provide some functional to compare the trajectories $y_x(s, \alpha)$. In the control theory this quantity is called cost functional and denoted by $J : \mathcal{A} \to \mathbb{R}$. The definition of J depends on the underlying control task. For an infinite horizon problem we consider

$$J_x(\alpha) = \int_0^\infty g(y_x(s, \alpha), \alpha(s))\, e^{-\lambda s}\, ds \tag{2.11}$$

with a function g for the running cost and λ for the discount factor. The existence of eq. (2.11) is guaranteed for a bounded g. A similar formulation for the finite horizon problem is

$$J_{x,t}(\alpha) = \int_t^T g(y_{x,t}(s, \alpha), \alpha(s))\, e^{-\lambda s}\, ds + e^{-\lambda(T-t)} \psi(y_{x,t}(T, \alpha)) \tag{2.12}$$

with a function ψ, which represents the costs of the final state. The goal of the optimal control problem is to determine the $\alpha^\star$ which minimizes the cost functional in eq. (2.11) or eq. (2.12). It is reasonable to distinguish between open-loop and feedback system. The open-loop architecture assumes the optimal control $\alpha^\star$ to be a function of t only. The main tool for this kind of problems is the Pontryagin Maximum Principle. However this approach has some drawbacks. E.g. it cannot take into account measure errors for the state of the system. Also different initial state requires new computation of the control.

On the contrary, the feedback approach assumes the control to be a function of the state. This allows to characterize the optimal feedback for each state and hence is more robust. However, we have to take the complete state space into account, which yields to higher computational costs. In the following we consider the optimal control problem in the feedback form.

To characterize the solution to the optimal control problem the the value function is introduced

$$v(x) \overset{\text{def}}{=} \inf_{\alpha \in \mathcal{A}} J_x(\alpha) \tag{2.13}$$

We can interpret the value function as the optimal cost for a system starting from the position x. An important property of eq. (2.13) is the Bellman's dynamic programming principle (DPP)

Proposition 2.1 Assume the solution to eq. (2.10) exists and is unique, then for all $x \in \mathbb{R}^d$ and $\tau > 0$

$$v(x) = \inf_{\alpha \in \mathcal{A}} \left\{ \int_0^\tau g(y_x(s,\alpha), \alpha(s))\, e^{-\lambda s}\, ds + e^{-\lambda\tau} v(y_x(\tau,\alpha)) \right\} \tag{2.14}$$

Proof See e. g.[FF11]. ☐

Please note that the DPP is essential for construction of Semi-Lagrangian schemes for the optimal control problem (see chapter 3).

From eq. (2.14) we can derive a characterization of the value function in terms of a non-linear PDE. Assume $(y^\star, \alpha^\star)$ to be an optimal pair. If we replace the $\inf$ by $\min$, the eq. (2.14) yields

$$v(x) - e^{-\lambda\tau} v(y_x^\star(\tau,\alpha^\star)) = \int_0^\tau g(y_x(s,\alpha^\star), \alpha^\star(s))\, e^{-\lambda s}\, ds$$

and by adding and subtracting $e^{\lambda\tau} v(x)$, dividing by τ and passing to the limit for $\tau \to 0^+$ we get

$$\lim_{\tau\to 0^+} \left[\underbrace{e^{-\lambda\tau} \frac{v(x) - v(y^\star(\tau))}{\tau}}_{-Dv(x)\cdot f(x,\alpha^\star(0))} + \underbrace{\frac{(1-e^{-\lambda\tau})v(x)}{\tau}}_{\lambda v(x)} = \underbrace{\frac{1}{\tau}\int_0^\tau g(y_x(s,\alpha^\star), \alpha^\star(s))\, e^{-\lambda s}\, ds}_{g(x,\alpha^\star(0))} \right]$$

Thus, with $\alpha(0) = a$, we can conclude

$$\lambda v(x) - Dv(x) \cdot f(x, a^\star) - g(x, a^\star) = 0$$

In this derivation we can replace $\alpha^\star$ by any other control $\alpha \in \mathcal{A}$ and obtain the Hamilton-Jacobi-Bellman (HJB) equation

$$\lambda v(x) + \sup_{a \in A} \{-Dv(x) \cdot f(x,a) - g(x,a)\} = 0 \tag{2.15}$$

as a counterpart to the Hamilton-Jacobi equation in the calculus of variations.

3 Semi-Lagrangian schemes

The idea of the Semi-Lagrangian schemes was first proposed for the advection equation in [CIR52]. A detailed derivation and survey can be found in [FF11]. This chapter gives a brief introduction to the topic.

The SL discretization is performed on the representation formula for the solution of a PDE, rather than on the equation itself. In the case of the HJB equation we can use the eq. (2.14) representation for the value function.

The method of characteristics can be considered as a motivation behind the SL schemes. As shown in section 2.2.2, for the first order HJ equation we can use characteristics to derive a representation formula for the solution in the integral form. In contrast to the PDE, which provides a characterization of the solution for infinitesimal small time steps, this formulation has allows to use larger time steps in the numerical calculation.

3.1 SL schemes for the optimal control problems

To construct a Semi-Lagrangian approximation we need to solve two problems: find a representation for the boundary data and solve the ODE for the characteristics. In the case of the HJB equation this means to approximate the ODE eq. (2.10). We can replace a trajectory from the set $y_x(s,\alpha)$ by a one step approximation. Therefore consider a discrete function, which is defined on the time grid $\{t_0,\dots,t_N\}$. We write y^{n+1} for $y^{t_{n+1}}$, respectively Δt_i for $t_{i+1}-t_i$. Thus, the approximated solution of the eq. (2.10) is given by

$$\begin{cases} y^{n+1} = y^n + \Delta t_i \Phi(y^n, A^n, \Delta t_i) \\ \quad y^0 = x \end{cases} \tag{3.1}$$

In the section 5.1.1 A^n denotes the set of discrete control vectors $A^n = \left\{a_0^n,\dots,a_{s-1}^n\right\}$ for an approximation with s stages. Note that the controls are a part of the solution.

The next part of the time discretization is the approximation of the integral in the eq. (2.14). Therefor we can use a quadrature formula with q nodes. Consider an index set $I = \{0,\dots,q-1\}$. The approximation $G^\Delta(x,\Delta t)$ (we omit the $e^{-\lambda s}$) is given by

$$\int_t^{t+\Delta t} g(y_x(s,\alpha),\alpha(s))\,ds \approx G^\Delta(x,\Delta t, A^n) \overset{\text{def}}{=} \Delta t \sum_{i\in I} \omega_i g(y_x^{\tau_i}, a_i^n) \tag{3.2}$$

τ_i and ω_i are the nodes and weights of a quadrature formula

$$0 \leqslant \tau \leqslant 1, \quad \omega_i \geqslant 0, \quad \sum_{i\in I} \omega_i = 1$$

The second problem is the approximation of the $v(y_x(t_{n+1}, \alpha))$. We consider a finite spatial grid composed of the points $\{x_1, \dots, x_n\}$. $I[V^{n+1}](y_x(t_{n+1}, \alpha))$ denotes an interpolation of $v(y_x(t_{n+1}, \alpha))$. We use a finite set of function values $V_i^{n+1} = v(x_i)$ for the construction. Thus the spatial approximation is given by

$$v(y_x(t_{n+1}, \alpha)) \approx I[V^{n+1}](y_x(t_{n+1}, \alpha)) \overset{\text{def}}{=} I[V^{n+1}](x, \alpha)$$

Now the complete Semi-Lagrangian scheme can be composed to

$$\begin{cases} v_j^n = \min\limits_{A^n} \left\{ G^\Delta(x_j, \Delta t_n, A^n) + I[V^{n+1}](x_j, A^n) \right\} \\ v_j^N = u(x_j) \end{cases} \tag{3.3}$$

3.2 Examples of SL schemes

[FF94] gives some examples for SL schemes of higher order in time. The simplest possible case for the approximation in time is the combination of the Euler method for the ODE and rectangular rule for the quadrature. We get $I = \{0\}$, $\tau_0 = 0$ and $\omega_0 = 1$

$$v_j^n = \min_{a_0^n} \left[\Delta t_n g(x_j, a_0^n) + I[V^{n+1}](x_j + \Delta t f(t_n, x_j, a_0^n)) \right] \tag{3.4}$$

In this section $I[V](x_j + \Delta t \Phi(t_n, x_j, A^n))$ states for $I[V]((x_j, A^n))$ to bring out the numerical integration scheme.

A more sophisticated approach is to use the Heun formula for the solution of the ODE and the trapezoid rule for the integration. In this case we have $I = \{0, 1\}$, $\omega_0 = \omega_1 = 1/2$ and $\tau_0 = 0$, $\tau_1 = 1$. The complete scheme is of the form

$$\begin{aligned} v_j^n = \min_{a_0^n, a_1^n} \Bigg[& \frac{\Delta t_n}{2} g(x_j, a_0^n) + \frac{\Delta t_n}{2} g(x_j + \Delta t \Phi(t_n, x_j, a_0^n, a_1^n, \Delta t), a_1^n) \\ & + I[V^{n+1}](x_j + \Delta t \Phi(t_n, x_j, a_0^n, a_1^n, \Delta t)) \Bigg] \end{aligned} \tag{3.5}$$

with

$$\Phi(t_n, x_j, a_0^n, a_1^n, \Delta t) = \frac{1}{2} f(t_n, x_j, a_0^n) + \frac{1}{2} f(t_n, x_j + \Delta t f(t_n, x_j, a_0^n), a_1^n)$$

Another example is the fourth order Runge-Kutta scheme. The parameters are $I = \{0, 1, 2, 3\}$, $\omega_0 = \omega_2 = 1/6$, $\omega_1 = \omega_3 = 1/3$, $\tau_0 = 0$, $\tau_1 = \tau_2 = 1/2$, $\tau_3 = 1$. The resulting scheme is

$$\begin{aligned} y_x^{\tau_0} &= x \\ y_x^{\tau_1} &= x + \Delta t \frac{k_0}{2} \\ y_x^{\tau_2} &= x + \Delta t \frac{k_1}{2} \\ y_x^{\tau_3} &= x + \Delta t k_2 \end{aligned}$$

with

$$
\begin{aligned}
k_0 &= f(t_n + \tau_0, x, a_0^n) \\
k_1 &= f(t_n + \tau_1, x + \Delta t \frac{k_0}{2}, a_1^n) \\
k_2 &= f(t_n + \tau_2, x + \Delta t \frac{k_1}{2}, a_2^n) \\
k_3 &= f(t_n + \tau_3, x + \Delta t k_2, a_3^n) \\
\Phi(t_n, x, a_0, a_1, a_2, a_3, \Delta t) &= \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3)
\end{aligned}
$$

$$
\begin{aligned}
v_j^n = \min_{a_0^n, a_1^n, a_2^n, a_3^n} \Bigg[& \frac{\Delta t_n}{6} \Big(g(y_{x_j}^{\tau_0}, a_0^n) + 2g(y_{x_j}^{\tau_1}, a_1^n) + 2g(y_{x_j}^{\tau_2}, a_2^n) + g(y_{x_j}^{\tau_3}, a_3^n) \Big) \\
& + I[V^{n+1}](x_j + \Delta t \Phi(t_n, x, a_0^n, a_1^n, a_2^n, a_3^n, \Delta t) \Bigg]
\end{aligned}
\tag{3.6}
$$

The approach corresponds to the Butcher tableau

$$
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
$$

As we can see in eq. (3.5) and eq. (3.6), the main drawback of the higher order schemes is the increasing complexity of the minimization problem. In fact, for a simple discretization of the control space with d_c samples and minimization through comparison we get $\mathcal{O}(d^2)$ evaluations of the trajectory for the eq. (3.5). The complexity for the RK4 discretization is $\mathcal{O}(d^4)$ eq. (3.6). In general for an one-step method with s stages the complexity for the minimization is $\mathcal{O}(d^s)$. An approach to overcome this disadvantage is presented in chapter 6.

4 Sparse grids

The derivation of SL schemes in the previous chapter (chapter 3) requires spatial approximation of the solution at the time points t_k. In particular, for the discretization of the Bellman's DPP eq. (2.14) an interpolation of the value function must be used to obtain the remaining running costs for a given trajectory.

In general every spatial approximation can be applied to the eq. (3.3) (e.g. Lagrangian, ENO, WENO or FE interpolation, see [FF11]). The interpolation of the solution is also referred to as reconstruction.

In this work, the spatial approximation is implemented with sparse grids. A detailed presentation and introduction to this reconstruction technique can be found in [BG04] and [Gar13] respectively. This chapter recalls main ideas of the sparse grids. The representation follows [GK15].

In the following, the presentation is for a space domain of the form $Q = [0,1]^d$, $d \in \mathbb{N}$. For a multi-index

$$\underline{l} = (l_1, \dots, l_d) \in \mathbb{N}^d$$

consider a mesh $Q_{\underline{l}}$ and a set of mesh parameters

$$h_{\underline{l}} = (h_{l_1}, \dots, h_{l_d}) = (2^{-l_1}, \dots, 2^{-l_d})$$

which represents the spatial resolution in every dimension for a given $\underline{l}$. Now, the points of the mesh $Q_{\underline{l}}$ can be denoted by

$$x_{\underline{l},\underline{j}} = (x_{l_1,j_1}, \dots, x_{l_d,j_d}), \quad x_{l_t,j_t} = j_t \cdot h_{l_t},\ t = 1, \dots, d.$$

$\underline{l}$ is also referred to as level and defines the resolution of the discretization, $\underline{j}$ defines the position of the point $x_{\underline{l},\underline{j}}$.

The space of all d-dimensional piecewise d-linear hat functions is

$$V_{\underline{l}} \overset{\text{def}}{=} \text{span}\left\{ \varphi_{\underline{l},\underline{j}} \,|\, j_t = 0, \dots, 2^{l_t},\ t = 1, \dots, d \right\}$$

For $V_{\underline{l}}$, the hierarchical difference space is given by

$$W_{\underline{l}} \overset{\text{def}}{=} V_{\underline{l}} \setminus \bigoplus_{t=1}^{d} V_{\underline{l}-\underline{e_t}}$$

(cf. fig. 4.1) with t-th unit vector $\underline{e_t}$. Thus, $V_{\underline{l}}$ can be represented as

$$V_{\underline{l}} = \bigoplus_{\underline{k} \leqslant \underline{l}} W_{\underline{k}}$$

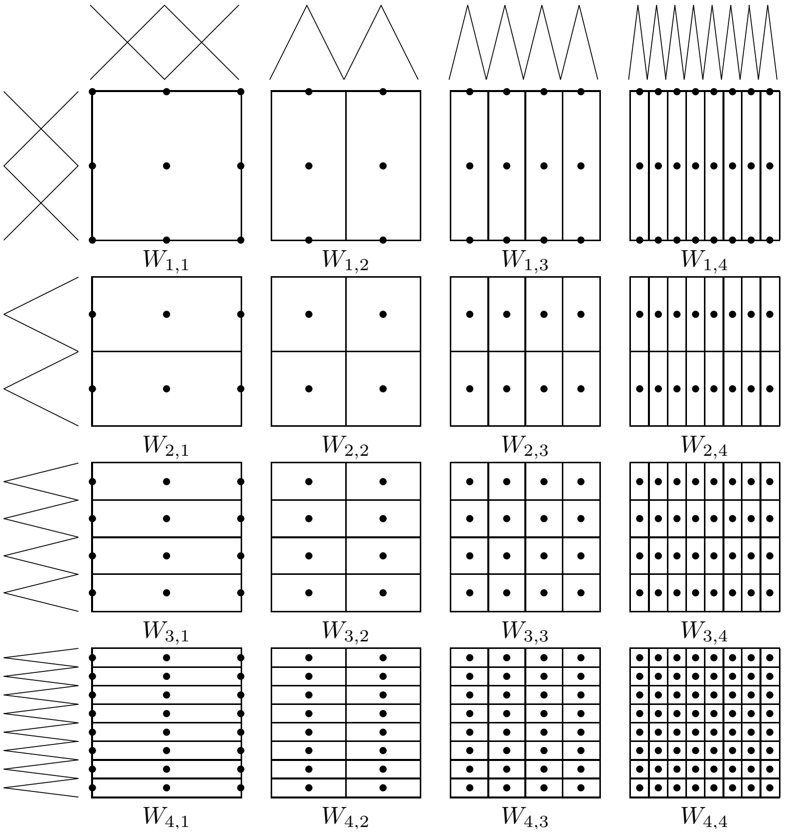


Figure 4.1: Hierarchical spaces for $V_{(4,4)}$

For $\underline{n} = (n, \dots, n) \in \mathbf{N}^d$, every $f \in V_{\underline{n}}$ can be characterized as

$$f(\underline{x}) = \sum_{|l|_\infty \leqslant n} \sum_{\underline{j} \in \mathcal{B}_{\underline{l}}} \alpha_{\underline{l},\underline{j}} \varphi_{\underline{l},\underline{j}}(\underline{x}) \tag{4.1}$$

where $\underline{x} \in Q$ and

$$\mathcal{B}_{\underline{l}} \overset{\text{def}}{=} \left\{ \underline{j} \in \mathbb{N} \,\middle|\, \begin{array}{lll} j_t = 0, \dots, 2^{l_t} - 1,\ j_t \text{ odd}, & t = 1, \dots, d, & \text{if } l_t > 1 \\ j_t = 0, 1, 2, & t = 1, \dots, d, & \text{if } l_t = 1 \end{array} \right\}$$

The idea of a sparse mesh is to take out those basis function $\varphi_{\underline{l},\underline{j}}$, which have a small contribution to the representation of the interpolated function f. For this purpose, the H^2_{mix} norm and semi-norm are considered

$$\|f\|^2_{H^2_{mix}(Q)} = \sum_{0 \leqslant \underline{k} \leqslant 2} \left| \frac{\partial^{|\underline{k}|_1} f}{\partial x_1^{k_1}, \dots, x_d^{k_d}} \right|_2^2 \quad \text{and} \quad |f|_{H^2_{mix}(Q)} = \left\| \frac{\partial^{2d} f}{\partial x_1^2, \dots, x_d^2} \right\|_2$$

The function space

$$H^2_{mix}(Q) = \left\{ f \in H \,|\, \|f\|_{H^2_{mix}(Q)} \leqslant C \text{ for } C > 0 \right\}$$

has the property, that for $f \in H^2_{mix}(Q)$ it holds

$$\|f_{\underline{l}}\|_2 \leqslant C(d) \cdot 2^{-2|\underline{l}|_1} |f|_{H^2_{mix}(Q)}$$

with constant $C(d) > 0$, which depends on the dimension d, and

$$f_{\underline{l}} \overset{\text{def}}{=} \sum_{\underline{j} \in \mathcal{B}_{\underline{l}}} \alpha_{\underline{l},\underline{j}} \varphi_{\underline{l},\underline{j}}(\underline{x}) \in W_{\underline{l}} \tag{4.2}$$

The estimation in eq. (4.2) motivates the replacement of $|\underline{l}|_\infty \leqslant n$ in eq. (4.1) by

$$|\underline{l}|_1 \leqslant n + d - 1 \tag{4.3}$$

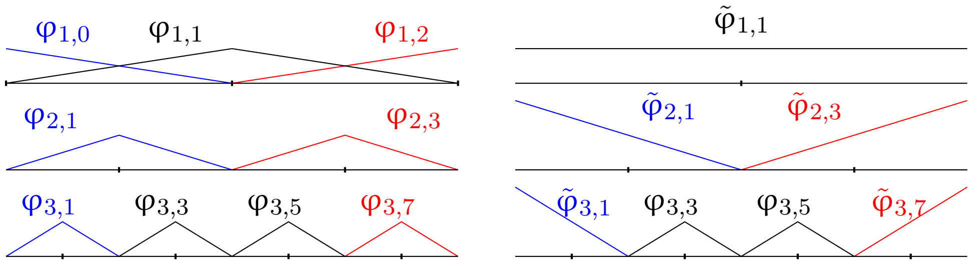
The resulting sparse grid space has the dimension $\dim V^s_{\underline{n}} = \mathcal{O}(2^n \cdot n^{d-1})$ in comparison to $\dim V_{\underline{n}} = \mathcal{O}(2^{nd})$ for regular grids. This fact leads to a significant reduction of the computational complexity. The error estimate for a function $f \in H^2_{mix}(Q)$ is

$$\|f - f^s_{\underline{n}}\|_2 = \mathcal{O}(h^2_n \cdot \log(h^{-1}_n)^{d-1})$$

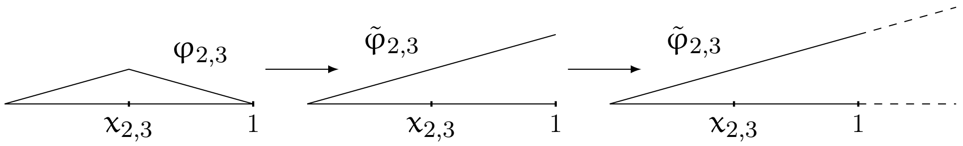
compared to

$$\|f - f_{\underline{n}}\|_2 = \mathcal{O}(h^2_n)$$

for a regular grid.



(a) normal basis (b) fold-out basis



(c) last basis before the boundary is folded up and extrapolated linearly

Figure 4.2: Normal and fold out basis functions.

For increasing level and dimension, the rule in eq. (4.3) leads to a high number of points on the boundary of the domain in comparison to the inner area. A different approach is to use the so-called fold out ansatz function. In this case, a sparse grid consists only of inner nodes and the reconstruction is extrapolated to and over the boundary, cf. fig. 4.2(c). An example of a sparse grid with fold out basis functions is given in fig. 4.3.

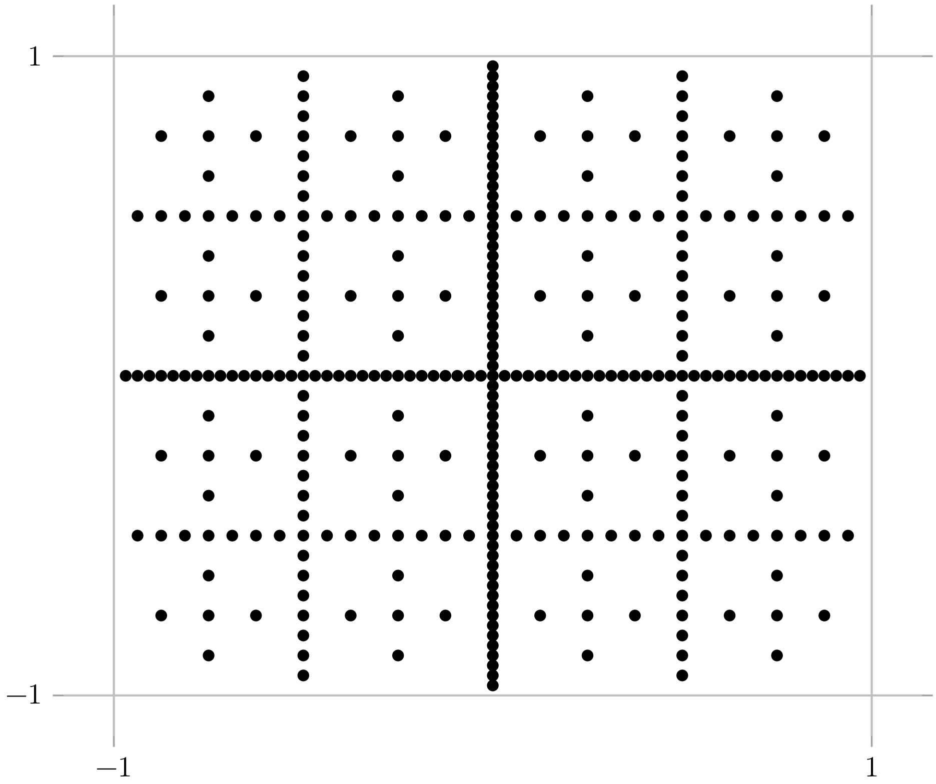


Figure 4.3: Sparse grid for level 6 on the domain $[-1, 1]^2$ with fold out ansatz functions.

5 Error estimation and step size control for Semi-Lagrangian schemes

The goal of this theses and the carried out numerical experiments in chapter 7 is to investigate the error of the SL scheme w.r.t. Δt. This analysis is the basis for the control of the time step. This chapter first gives an overview of basic aspects of the error estimation for numerical schemes. The second part deals with the control of the step size.

5.1 Approximation error of SL schemes

The common approach for the calculation of optimal Δt for the ODEs uses the estimation of the local error. However, the SL scheme consists of multiple approximation steps. Besides the numerical solution of the ODE eq. (2.10), there are integration and interpolation. This section describes the error estimates for the SL approximation of the HJB equation.

5.1.1 Local and global error of numerical integration methods for ODEs

Recall the definition of the consistency error (see e.g. [FF11, DB02, Sha94])

$$e_{loc}(t) = \frac{1}{\Delta t}\big[U(t+\Delta t) - S(\Delta; t, U(t))\big] \tag{5.1}$$

where U is the exact solution and S is the numeric approximation. A Runge-Kutta method has order p if $e_{loc}(t) = \mathcal{O}(\Delta t^p)$. The consistency error is a measure for the deviation of S from the exact solution after an iteration with Δt. However, from the practical viewpoint, it is important to know the global discretization error

$$e^m = U(t_m) - V^m \tag{5.2}$$

where V^m, the solution after m iterations, is given by

$$\begin{cases} V^m = S(\Delta; t_{m-1}, V^{m-1}) \\ V^0 = U_0 \end{cases}$$

It is a well known result for ODEs (see e.g. [DB02, Sha94], that for a Lipschitz-continuous increment function Φ the consistency error of order p is sufficient for the convergence of order p. Please note that this result holds especially for the case of a linear ODE with constant coefficients, which are investigated in chapter 7.

In the following, the convergence order is referred to as the order of Runge-Kutta scheme.

5.1.2 Error of quadrature

The error estimation for the quadrature formula eq. (3.2) is different from the one for ODEs. There is no dependence on the state variable and no propagation of the local errors. The only independent quantity is the integration variable. Thus, the global error is the sum of local errors.

Recall, that the quadrature formula G^Δ has accuracy level k if all polynomials $p \in \Pi_k$ (all polynomials of order at most k) are integrated exactly (see [SK09]). For a sufficiently differentiable function, the error order of G^Δ is $\mathcal{O}(\Delta t^{k+1})$ (or $\mathcal{O}(\Delta t^{k+2})$ in a case of non-negative weights ω_i). In general a quadrature formula with s nodes can achieve the accuracy level of up to $2s-1$ and thus provide an error order $\mathcal{O}(\Delta t^{2s+1})$. Those are the Gaussian quadrature methods. In the following, the order q of G^Δ is referred to as $q =$ error order $- 1$. So for the Gauss method we obtain order $2s$, where s is again the number of nodes.

Please note that there is a strong connection between Runge-Kutta and quadrature methods. We can apply any Runge-Kutta scheme to the initial value problem

$$\dot{x}(t) = f(t), \quad x(0) = x_0$$

obtaining a quadrature G^Δ with

$$\omega_i = b_i,\ \tau_i = t_0 + c_i\tau,\ g(y_x(\tau_i)) = f(t_0 + c_i\tau),\ i = 1, \dots, q$$

In this case the quadrature formula has at least order of the Runge-Kutta method, however not vice versa (see [Gar15]).

5.1.3 Properties of collocation methods

A special class of implicit Runge-Kutta methods are the collocation methods (every collocation method is equivalent to an implicit Runge-Kutta sheme). It can be shown, that by setting the collocation points to the nodes, which are defined by some quadrature formula, one obtains a superconvergence: the corresponding Runge-Kutta method has the same order as the underlying quadrature formula (see [HLW06, DB02]). Therefore, the collocation methods based on the Gauss quadrature are of order $2s$, where s is the number of nodes. In particular the implicit midpoint rule is of order 2.

Another important property of the collocation methods is the implicit definition of a collocation polynomial. In general, the approximation of the ODE by a Runge-Kutta method provides a solution y_{t_k} at discrete time points t_k. However, the calculation of the quadrature G^Δ in eq. (3.2) requires the evaluation of the trajectory at the time points τ_i, i.e. $y_x^{\tau_i} = y_x(\tau_i)$. These points can be obtained at the boundary of the time interval t_k and $t_k + \Delta t$. But, in general, the quadrature formula also uses interior points. Thus, to preserve the order of the error estimates for the G^Δ, we have to provide an approximation of $y_x(\tau)$ of appropriate order (see [FF94]). This consideration is important for the construction of high order SL schemes as in chapter 6 and can formulate in

Problem 5.1 To a given quadrature rule with order q and a numerical integration method for the ODE section 5.1.1 find an approximation of the solution $y_x(\tau_i)$ at the intermidiate quadrature nodes τ_i, which is at least of the same order $r \geqslant q$.

For the collocation polynomial $u(t)$, it can be shown (see [HLW06])

$$|u(t) - y(t)| \leqslant C \cdot h^{s+1} \quad \text{for } t \in [t_k, t_k + \Delta t]$$

where s is the number of stages of the collocation method. Although, the collocation polynomial is not explicitly constructed, the evaluation is easy obtained for the implicit midpoint rule (see section 6.3). This property will be used to construct SL scheme of high order for the solution of the HJB equation.

5.1.4 Local error of SL scheme

With observations on the error estimates for the quadrature and numerical integration, we can now consider the local error of the SL scheme.

Let $I_r[V^{n+1}]$ be an interpolation operator of order r (i.e. the error is given by $\mathcal{O}(\Delta x^{r+1})$ for the spatial discretization with $\Delta x = \sup_i \|x_{i+1} - x_i\|$). To compute the overall error for the SL scheme consider eq. (2.14) for the exact optimal control $\alpha^\star$. Further assume, that the $\inf$ is a $\min$ and A^n is an approximation to $a^\star$ of sufficient order. For the local error we obtain

$$\begin{aligned}
v(x_j, t_n) - v_j(\Delta t, \widehat{V}^{n+1}) &= \min_{\alpha \in \mathcal{A}} \left\{ \int_{t_n}^{t_n+\Delta t} g(y_{x_j}(s,\alpha),\alpha)\, ds + v(y_{x_j}(t_n+\Delta t, \alpha)) \right\} \\
&\quad - \min_{A^n} \left\{ G^\Delta(x_j, \Delta t, A^n) + I_r[\widehat{V}^{n+1}](x_j, A^n) \right\} \\
&= \int_{t_n}^{t_n+\Delta t} g(y_{x_j}(s,\alpha^\star),\alpha^\star)\, ds + v(y_{x_j}(t_n+\Delta t, \alpha^\star)) \\
&\quad - G^\Delta(x_j, \Delta t, A^n) + I_r[\widehat{V}^{n+1}](x_j, A^n) \\
&= \mathcal{O}(\Delta t^{q+1}) + \underbrace{v(y_x(\Delta t, \alpha)) - v(y_x^{t+\Delta t})}_{\mathcal{O}(\Delta t^{q+1})} + \underbrace{v(y_x^{t+\Delta t}) - I_r[\widehat{V}^{n+1}](x_j, A^n)}_{\mathcal{O}(\Delta x^{r+1})} \\
&= \mathcal{O}(\Delta t^{q+1} + \Delta x^{r+1})
\end{aligned} \tag{5.3}$$

where $\widehat{V}^{n+1}$ is the exact value function for t_{n+1} and q is the order of quadrature formula, respectively the Runge-Kutta method.

Please note that the assumption on the control approximation haven't been validated in this work. However, the numerical experiments shows comparable behaviour for different optimization techniques (i.e. gradient and compare optimization). This result is also observed in [GK15].

Up to now we have provided an upper bound on the error for I[V]. Though, the interpolation error also depends on Δt. Due to consistency of the Runge-Kutta method, the trajectory of the controlled system $y_{x_j}(\Delta t, \alpha)$ converges to the point x_j for $\Delta t \to 0$. At the same time, x_j is a grid point for the interpolation I[V]. Therefore, the error estimate can be improved to

$$|v(x) - I_r[V](x)| \leqslant C\Delta x^{r+1} \min_{m \in M} \frac{x - x_m}{\Delta x}$$

where $v \in C^{r+1}$ and M is the set of spatial nodes (see [FF11]).

In the SL scheme, the grid point x_m is approached on the trajectory $y_{x_m}(\Delta t)$. Hence, for a bounded f in eq. (2.10), we can set

$$L_f = \max_{x \in \Omega, \alpha \in \Omega_c} |f(x, \alpha)|$$

where Ω and Ω_c denote state space and control space respectively. Next, we can obtain the following estimation

$$|y_{x_m}(\Delta t) - y_{x_m}(0)| \leqslant L_f \Delta t$$

The estimation for the interpolation error is now

$$|v(x) - I_r[V](x)| \leqslant C \Delta x^{r+1} \frac{L_f \Delta t}{\Delta x} = \widehat{C} \Delta x^r \Delta t$$

This bound is still global and the factor $(x - x_m)/\Delta x$ only takes into account the reduction of the error when approaching a node. For the complete SL scheme the eq. (5.3) is now of the form

$$v(x_j, t) - v_j(\Delta t, \widehat{V}^{n+1}) = \mathcal{O}(\Delta t^{q+1} + \Delta x^r \Delta t) \tag{5.4}$$

5.2 Step size control

There are many references for the control of the step size for ODEs. In this section the representation is based on [DB08, Gar15]. On the one hand, the goal of the step size adaptation is to set Δt so, that

$$\|\varepsilon_{n+1}\| \leqslant \mathrm{TOL}$$

is guaranteed. Hereby ε_{n+1} denotes the estimation of the local consistency error and TOL is the accepted tolerance. On the other hand, too little time steps result in high computational effort. Hence, the objective is to have the optimal error $\hat{\varepsilon}_{n+1}$

$$\|\hat{\varepsilon}_{n+1}\| \approx \mathrm{TOL}$$

as close to the TOL as possible. From the analysis of the local error in section 5.1.1, we know, that for a method of order p it holds

$$\|\varepsilon_{n+1}\| \approx C(t_n) \Delta t^{p+1}$$

and can now provide an expression for an optimal step size $\widehat{\Delta t_k}$

$$\widehat{\Delta t_n} = \sqrt[p+1]{\frac{\rho \mathrm{TOL}}{\|\varepsilon_{n+1}\|}} \Delta t_n$$

with a safety coefficient ρ. The optimal $\widehat{\Delta t}$ can be used as a suggestion for the next time step.

The control approach also usually incorporates some limits on the step size Δt_{max} and on the increase of the Δt. Thus, the complete formulation for the new step size is

$$\Delta t_{n+1} = \min \left\{ q\Delta t_n,\ \Delta t_{max},\ \sqrt[p+1]{\frac{\rho TOL}{\|\varepsilon_{n+1}\|}}\Delta t_n \right\} \tag{5.5}$$

Please note that the underlying model for this type of controller differs from the one, derived in eq. (5.4) for the SL scheme. However, we can use eq. (5.5) at least for the small values of the spatial discretization error Δx.

5.2.1 Estimation of error for step size control

There exist different approaches to provide the estimation of the local error ε for eq. (5.5). The approach of the Richardson interpolation is to repeat the computation of the solution with a time step $\Delta t/2$. To illustrate this method, we consider an approximation scheme $y(h)$, which depends on a parameter h and has the error formula

$$y - y(h) = C_{k_0}h^{k_0} + C_{k_1}h^{k_1} + \dots$$

where y denotes the exact solution. Using different step sizes h and h/m we get

$$\begin{aligned} y &= y(h) + C_{k_0}h^{k_0} + C_{k_1}h^{k_1} + \dots \\ y &= y\left(\frac{h}{m}\right) + C_{k_0}\left(\frac{h}{m}\right)^{k_0} + C_{k_1}\left(\frac{h}{m}\right)^{k_1} + \dots \end{aligned}$$

and after multiplying the second equation with m^{k_0} and building the difference

$$\begin{aligned} \left(m^{k_0} - 1\right) y &= m^{k_0} y\left(\frac{h}{m}\right) - y(h) + m^{k_0} C_{k_1}\left(\frac{h}{m}\right)^{k_1} - C_{k_1}h^{k_1} + \dots \\ &= m^{k_0} y\left(\frac{h}{m}\right) - y(h) + \mathcal{O}(h^{k_1}) \end{aligned}$$

These equations can be solved for y and we obtain an approximation of order k_1 in h

$$y = \underbrace{\frac{m^{k_0} y\,(h/m) - y(h)}{m^{k_0} - 1}}_{z(h)} + \mathcal{O}(h^{k_1})$$

We can use the approximation z as the exact value and compute the error estimation to

$$\varepsilon = z(h) - y(h)$$

6 High order time methods for Semi-Lagrangian schemes

Some properties of the SL schemes regarding the time integration and the usage for the HJB equation are discussed in this section. Based on these considerations, a SL scheme, which uses the composition of implicit midpoint rules for the solution of the ODE and the quadrature, is proposed.

6.1 Structure preserving numerical integration for the SL schemes

One of the reasons for the development of SL schemes is the possibility to use large time step sizes in the numerical integration (see [Bon04]). To be more precise, we can consider the the linear advection equation. For the finite differences discretization, the concept of numerical domain of dependence imposes the CFL condition. In contrast, the SL discretization of the same equation calculates the foot of the characteristic (additional cost) for each grid node x_j and thus follows the numerical domain of dependence (see [FF11]).

This consideration motivates the usage of Runge-Kutta schemes which are appropriate for the long time integration. These schemes usually take into account additional properties of the underlying dynamical system, i.e. symmetry of symplecticity (see [HLW06]).

The numerical experiments that are carried out in this theses are based on results in [GK15], especially on the test case with controlled harmonic oscillator, which is in fact symplectic (also the controlled system fulfils the symplecticity condition as in [HLW06]). This suggests the usage of structure preserving Runge-Kutta schemes.

6.2 Complexity of SL schemes with high order time discretization

The usage of high order Runge-Kutta methods and the compare approach for the optimization in the SL scheme is restricted by the exponentially rising complexity (see section 3.2). E.g., for the SL scheme with a fourth order Runge-Kutta method (eq. (3.6)), we have to calculate $\mathcal{O}(d_c^4)$ trajectories in each time step. This section provides some suggestions to reduce the complexity.

The presented approach follows the idea of the Bellman's DPP eq. (2.14). By considering the optimization problem backward in time, it is possible to significantly reduce the amount of path evaluations for the controlled system. Similar approach can be used within a single time step of the numerical integration. However, this requires the separation of the impact of the control on the stages of the RK method. This seems not to be impossible, as we can easy verify, that for a explicit, respectively diagonally implicit RK scheme, the last element a^n_s of the discrete control vector A^n affects only the last stage k_s. In fact, a condition on the coefficients of a RK method can be formulated, so that the optimization of the control can be applied separately to each stage. A class of RK methods, which fulfils this requirements are the diagonaly implicit symplectic Runge-Kutta (DISRK) schemes. These methods were developed for the long time integration of Hamiltonian systems (see [HLW06]). The result is especially interesting in the context of strong connections between the HJB and the HJ equations (cf. chapter 2).

6.2.1 Explicit time integration

Assume we have an explicit Runge-Kutta method with s stages given by the tableau

$$\begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b} \end{array} = \begin{array}{c|cccc} 0 & 0 & 0 & \dots & 0 \\ c_2 & a_{21} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ c_s & a_{s1} & \dots & a_{ss-1} & 0 \\ \hline & b_1 & b_3 & \dots & b_s \end{array}$$

To prevent ambiguous notation between the coefficients a_{ij} of the RK scheme and the control a^n_j, which is applied in the stage j, let us write A^n_j for the a^n_j. Following the notation in chapter 3 we obtain $A^n = \{A^n_0, \dots, A^n_{s-1}\}$. Then, the approximation of the system ODE eq. (2.10) is given by

$$\begin{cases} y^{n+1}_x = x + \tau \sum_{j=1}^{s} b_j k_j \\ k_j = f\left(t_n + c_j, x + \tau \sum_{i=1}^{j-1} a_{ji} k_i, A^n_j\right), \qquad j = 1, \dots, s \end{cases} \tag{6.1}$$

where τ denotes the step size. Applying eq. (6.1) to the SL scheme eq. (3.3) and splitting off the terms with k_s we get ($\tau = \Delta t$)

$$v^n_j = \min_{A^n} \left[G^\Delta(x_j, \tau, A^n) + I[V^{n+1}](x_j, A^n)\right] \tag{6.2}$$

$$= \min_{\{A^n_1,\dots,A^n_s\}} \left[\tau \sum_{i=1}^{s-1} b_i g(y^{\tau_i}_x, A^n_i) + I[V^{n+1}]\left(x_j + \tau \sum_{i=1}^{s-1} b_i k_i + \tau b_s k_s\right)\right] \tag{6.3}$$

$$= \min_{\{A^n_1,\dots,A^n_s\}} \left[\tau \sum_{i=1}^{s-1} b_i g(y^{\tau_i}_x, A^n_i)\right. \tag{6.4}$$

$$\left. + I[V^{n+1}]\left(x_j + \tau \sum_{i=1}^{s-1} b_i k_i + \tau b_s f\left(t_n + c_s, x_j + \tau \sum_{i=1}^{s-1} a_{si} k_i, A^n_{s-1}\right)\right)\right] \tag{6.5}$$

Next, to simplify the notation, consider auxiliary variables

$$\gamma_{s-1} \overset{\text{def}}{=} x_j + \tau \sum_{j=1}^{s-1} b_j k_j$$

$$\kappa_{s-1} \overset{\text{def}}{=} x_j + \tau \sum_{i=1}^{s-1} a_{s-1i} k_i \tag{6.6}$$

Further, we assume, there exists an expression for $y_x^{\tau_i}$ of the form

$$y_x^{\tau_i} = p_x(\kappa_{i-1})$$

An example for such a function p_x is given in section 6.3.2. With these preliminary remarks we can write eq. (6.5) as

$$\begin{aligned} v_j^n = \min_{\{A_1^n,\dots,A_s^n\}} \Bigg[& \tau \sum_{i=1}^{s-1} b_i g(p_x(\kappa_{i-1}), A_i^n) + \tau b_s g\left(p_x(\kappa_{i-1}), A_s^n\right) \\ & + I[V^{n+1}]\left(\gamma_{s-1} + b_s f\left(\kappa_{s-1}, A_s^n\right)\right) \Bigg] \end{aligned} \tag{6.7}$$

Note, that in the eq. (6.7) A_s^n influences only k_s. Therefore, analogue to the DDP (2.14), we can write

$$\begin{aligned} v_j^n = \min_{\{A_1^n,\dots,A_{s-1}^n\}} \Bigg[& \tau \sum_{i=1}^{s-1} b_i g(p_x(\kappa_{i-1}), A_i^n) \\ & + \min_{A_s^n} \left[\tau b_s g\left(p_x(\kappa_{i-1}), A_s^n\right) + I[V^{n+1}]\left(\gamma_{s-1} + \tau b_s f\left(\kappa_{s-1}, A_s^n\right)\right) \right] \Bigg] \end{aligned} \tag{6.8}$$

and minimize over A_s^n independent of $A_1^n, \dots, A_{s-1}^n$.

The evaluation of the last stage in eq. (6.8) depends only on γ_{s-1}, κ_{s-1} and A_s^n. To simplify the notation we can define a function

$$\vartheta_s^n(\gamma_{s-1}, \kappa_{s-1}) \overset{\text{def}}{=} \min_{A_s^n} \left[\tau b_s g\left(p_x(\kappa_{s-1}), A_s^n\right) + I[V^{n+1}]\left(\gamma_{s-1} + \tau b_s f\left(\kappa_{s-1}, A_s^n\right)\right) \right]$$

ϑ_s^n is the part of the minimization problem, which depends on A_s^n only. Now, a new representation for the value function is

$$v_j^n = \min_{\{A_1^n,\dots,A_{s-1}^n\}} \left[\tau \sum_{i=1}^{s-1} b_i g(p_x(\kappa_{i-1}), A_i^n) + \vartheta_s^n(\gamma_{s-1}, \kappa_{s-1}) \right]$$

Similar to the approach of the SL scheme, we can approximate ϑ_s^n with an interpolation over the $(\gamma_{s-1}, \kappa_{s-1})$. The resulting scheme is of the form

$$v_j^n = \min_{\{A_1^n,\dots,A_{s-1}^n\}} \left[\tau \sum_{i=1}^{s-1} b_i g(p_x(\kappa_{i-1}), A_i^n) + I[\theta_s^n](\gamma_{s-1}, \kappa_{s-1}) \right]$$

with the vector of node values

$$(\theta_s^n)_{ij} = \vartheta_s^n((\gamma_{s-1})_i, (\kappa_{s-1})_j) \tag{6.9}$$

for some spatial grid $\left\{(\gamma_{s-1})_i, (\kappa_{s-1})_j\right\}_{i,j}$.

Up to now, we have an expression, which separates the impact of the last control sample A_s^n on the trajectory. The complexity of the minimization problem in eq. (6.9) w.r.t control is $\mathcal{O}(d_c^{s-1} + d_c)$. A disadvantage of this form is, that we have to calculate the discretization over the space $(\gamma_{s-1}, \kappa_{s-1})$ and the complexity to obtain or evaluate $I[\theta_s^n]$ would be square of the complexity for the discretization of the state space.

Another simplification can be realized if a dependency between κ_{s-1} and γ_{s-1} is assumed. In particular, we are interested in a representation of the form

$$\gamma_{s-1} = r(\kappa_{s-1})$$

for some function r. Given such relationship, the resulting SL scheme is

$$v_j^n = \min_{\{A_1^n,\dots,A_{s-1}^n\}} \left[\tau \sum_{i=1}^{s-1} b_i g(p_x(\kappa_{i-1}), A_i^n) + I[\theta_s^n](\kappa_{s-1})\right]$$

Now, we can apply the same derivation to the stages $s-1, \dots, 1$ of the RK scheme. In each step a stage value function must be constructed. We can conclude

$$v_j^n = \min_{\{A_1^n\}} \left[\tau b_1 g(p_x(\kappa_0), A_1^n) + I[\theta_2^n](\kappa_1)\right] \tag{6.10}$$

where for $l = 2, \dots, s$

$$\begin{aligned} I[\theta_l^n]((\kappa_{l-1})_j) &= \vartheta_l^n((\kappa_{l-1})_j) \\ \vartheta_l^n((\kappa_{l-1})_j) &\overset{\text{def}}{=} \min_{A_l^n} \left[\tau b_l g\left(p_x(\kappa_{l-1}), A_l^n\right) + I[\theta_l^n]\left(\kappa_{l-1}\right)\right] \end{aligned}$$

Please note that this derivation assumes, that κ_i and γ_i are provided for all $i = 1, \dots, s$. A possible way to calculate this quantities will be illustrated in section 6.2.3.

6.2.2 Implicit time integration

Next, consider an implicit Runge-Kutta method with s stages given by the tableau

$$\begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b} \end{array} = \begin{array}{c|ccccc} c_1 & a_{11} & a_{12} & \dots & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & \dots & b_s \end{array}$$

Analogue to eq. (6.1) we obtain

$$\begin{cases} y_x^{n+1} = x + \tau \sum_{j=1}^{s} b_j k_j \\ k_j = f\left(t_n + c_j, x + \tau \sum_{i=1}^{s} a_{ji} k_i, A_j^n\right), \qquad j = 1, \dots, s \end{cases} \tag{6.11}$$

In general, the control A_j^n for some $j = 1, \dots, s$ has impact on all the stages due to the implicit character of the scheme. E.g. modification of A_s^n influences k_s and therefore all k_j, which are functions of k_s.

However, it is possible to apply the derivation of eq. (6.10) to the diagonally implicit methods. These are the implicit Runge-Kutta schemes with $a_{ij} \neq 0$ only for $j \leqslant i$. In this case the, calculation of the stages k_j is implicit

$$k_i = f\left(x_n + \tau \sum_{j=1}^{i} a_{ij} k_j, A_i^n\right) = f\left(\kappa_{i-1}, A_i^n\right) \tag{6.12}$$

Specially, we have to solve a non-linear equation. Please note that in eq. (6.12) we assume A_i^n to be constant w.r.t. k_i. This is e.g. true for the comparison approach in the minimization of the cost functional.

6.2.3 Reduction of complexity

Now let us examine the calculation of κ_i and γ_i that were defined in eq. (6.6). The SL scheme is applied backward in time to the HJB equation. This means that the construction of the stage value function $\vartheta_l^n(\gamma_{l-1}, \kappa_{l-1})$ implicitly defines the values of κ_{l-1} and γ_{l-1} for each $l = 2, \dots, s$. I.e., in the next step $l - 1$, the $\vartheta_{l-1}^n(\gamma_{l-2}, \kappa_{l-2})$ can be only calculated if we provide γ_{l-1} and κ_{l-1}. These can be plugged into the interpolation operator $I[\theta_l^n]$ to approximate the value of $\vartheta_l^n(\gamma_{l-1}, \kappa_{l-1})$.

However, the values of the stages $k_{l-2}, \dots, k_1$ are not known in the step $l-1$. I.e. we do not know the evolution of the exact trajectory up to some time point, we just assume, the trajectory ends in some x_j on the spatial grid. Then the remaining running costs are calculated, starting from this point x_j. This is the idea of the Bellman's dynamical programming principle (eq. (2.14)). For the sake of simplicity, in the following it is assumed, that $\gamma_{s-1} = r(\kappa_{s-1})$. Therefore we can write $\vartheta_s^n(\gamma_s, \kappa_s) = \vartheta_l^n(\kappa_{l-1})$. This simplification can be verified in following (see e.g. (6.13)).

The consideration up to now implies, that if we want to utilize the Bellman's DPP, then it is necessary to calculate κ_{l-1} from κ_{l-2} and k_{l-1}. A trivial approach is e.g.

$$\kappa_l = x_n$$

for all $l = 1, \dots, s - 1$. In this case, the resulting Runge-Kutta scheme would be in the form

$$\begin{array}{c|ccccc}
c_1 & a_{11} & 0 & \dots & \dots & 0 \\
c_2 & 0 & a_{22} & \dots & \dots & 0 \\
\vdots & 0 & 0 & a_{33} & \ddots & \vdots \\
\vdots & \vdots & & \ddots & \ddots & \vdots \\
c_s & 0 & \dots & \dots & 0 & a_{ss} \\
\hline
 & b_1 & b_2 & \dots & \dots & b_s
\end{array}$$

Please note that already for κ_i, which is defined by

$$\kappa_i = x + \tau a_{ii-1} k_{i-1}$$

it is not possible to calculate κ_i given κ_{i-1} and k_{i-1}. I.e. we know

$$\kappa_{i-1} = x + \tau a_{i-1i-2} k_{i-2}$$

and k_{i-1}. However, to calculate κ_i we have to provide x explicitly (except for the last stage). There is only one equation (κ_{i-1}) and two unknowns (k_{i-2} and x).

Further, to achieve the form $\vartheta_s^n(\gamma_s, \kappa_s) = \vartheta_l^n(\kappa_{l-1})$, we can set $b_1, \dots, b_{s-1} = 0$. In this case the resulting expression for γ_l are $\gamma_{s-1} = \kappa_{s-1}$ and $\gamma_l = 0$ for $l = 1, \dots, s-2$. The Butcher tableau is now in the form

$$\begin{array}{c|ccccc}
c_1 & a_{11} & 0 & \dots & \dots & 0 \\
c_2 & 0 & a_{22} & \dots & \dots & 0 \\
\vdots & 0 & 0 & a_{33} & \ddots & \vdots \\
\vdots & \vdots & & \ddots & \ddots & \vdots \\
c_s & 0 & \dots & \dots & 0 & a_{ss} \\
\hline
 & 0 & \dots & \dots & 0 & b_s
\end{array}$$

However, this Runge-Kutta method is not of great interest It is essentially equivalent to a scheme with tableau

$$\begin{array}{c|c}
c_s & a_s s \\
\hline
 & b_s
\end{array}$$

The requirement to utilize the DPP for the numerical integration can be more precise formulated as

Problem 6.1 Find a Runge-Kutta scheme such that for $i = 2, \dots, s$ k_i can be expressed in terms of κ_{i-1} and the stage k_{i-1}.

Another solution to problem 6.1 is given by the tableau

$$\begin{array}{c|ccccc}
c_1 & \frac{b_1}{2} & 0 & \dots & \dots & 0 \\
c_2 & b_1 & \frac{b_2}{2} & 0 & \dots & 0 \\
\vdots & b_1 & b_2 & \frac{b_3}{2} & 0 & \vdots \\
\vdots & \vdots & & & \ddots & 0 \\
c_s & b_1 & b_2 & b_3 & \dots & \frac{b_s}{2} \\
\hline
 & b1 & b_2 & b_3 & \dots & b_s
\end{array} \tag{6.13}$$

We can verify, that (6.13) provides a solution to the problem 6.1. For a given

$$\kappa_i = x_n + \tau \sum_{j=1}^{i-1} b_j k_j$$

and k_i, the expression for k_{i+1} is

$$\kappa_{i+1} = x_n + \tau \sum_{j=1}^{i-1} b_j k_j + \tau b_i k_i = \kappa_i + \tau b_i k_i$$

$$k_{i+1} = f\left(\kappa_{i+1} + \tau \frac{b_{i+1}}{2} k_{i+1}\right)$$

At the first sight, the condition (6.13) may seem rather strict. However, there are actually Runge-Kutta methods which satisfy the tableau (6.13). Those are exactly the diagonally implicit symplectic Runge-Kutta (DISRK) methods. With this result, we can construct a SL scheme, which has $\mathcal{O}(d_c \cdot s)$ complexity for the minimization problem.

6.3 SL schemes with implicit midpoint rule

This section is based on the results from section 6.2.3. The derivation of a SL scheme with the DISRK integration, especially the implicit midpoint rule, is illustrated in the following section.

6.3.1 Composition methods

An important property of DISRK is that these methods are equivalent to a composition of implicit midpoint schemes.

Theorem 6.1 If Φ_τ is a map of diagonally implicit Runge-Kutta scheme satisfying the symplecticity condition

$$b_i a_{ij} + b_j a_{ji} = b_i b_j \quad \text{for all } i, j = 1, \dots, s \tag{6.14}$$

and $b_i \neq 0$, then Φ_τ is equivalent to the composition

$$\Phi_\tau = \Phi^M_{b_s\tau} \circ \ldots \circ \Phi^M_{b_2\tau} \circ \Phi^M_{b_1\tau}$$

where Φ^M_τ denotes the implicit midpoint rule.

Proof See [HLW06]. □

The condition in eq. (6.14) for a Runge-Kutta method to be symplectic is derived from the requirement to preserve quadratic first integrals (see [HLW06]).

The equivalence of a composition of the implicit midpoint schemes to some DISRK allows to construct methods of high order. Additional theorem states the condition on the time steps $b_i\tau$

Theorem 6.2 Let Φ_τ be a one-step method of order p and

$$\Psi = \Phi_{\gamma_s\tau} \circ \ldots \circ \Phi_{\gamma_2\tau} \circ \Phi_{\gamma_1\tau}$$

If $\gamma_1, \ldots, \gamma_s$ fulfil

$$\begin{aligned} &\gamma_1 + \ldots + \gamma_s = 1 \\ &\gamma_1^{p+1} + \ldots + \gamma_s^{p+1} = 0 \end{aligned} \tag{6.15}$$

then the composition method is at least of order $p+1$.

Proof See [HLW06]. □

With theorem 6.2, the construction of higher order methods can be achieved by solving eq. (6.15). Different schemes of this kind are quoted in [HLW06] and references therein. First recall that the implicit midpoint rule has the following Butcher's tableau

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array} \tag{6.16}$$

The simplest DISRK scheme, which is obtained through the composition has order 4 and is of the form

$$\gamma_1 = \gamma_3 = \frac{1}{2 - 2^{1/(p+1)}}, \quad \gamma_2 = -\frac{2}{2 - 2^{1/(p+1)}} \tag{6.17}$$

Corresponding Butcher's tableau is (see [FQ10])

$$\begin{array}{c|ccc} \frac{1}{2}a & \frac{1}{2}a & & \\ \frac{1}{2} & a & \frac{1}{2} - a & \\ 1 - \frac{1}{2}a & a & 1 - 2a & \frac{1}{2}a \\ \hline & a & 1 - 2a & a \end{array} \tag{6.18}$$

where $a = \frac{1}{2 - 2^{1/(p+1)}} \overset{(p=2)}{=} 1.35120719195966$.

In the following, this method is referred to as DISRK3. Please note that it is not necessary to calculate the Butcher's tableau of the composition DISRK explicit. The construction of the scheme allows to apply the implicit midpoint methods Φ_{γ_i} with time steps $\gamma_i \tau$ given by eq. (6.15) in a loop. Thus, the method in tableau 6.18 is equivalent to

$$\Phi_\tau^{\mathrm{DISRK3}} = \Phi_{\gamma_3 \tau}^{M} \circ \Phi_{\gamma_2 \tau}^{M} \circ \Phi_{\gamma_1 \tau}^{M}$$

with γ_i from eq. (6.17).

Other examples for solutions to eq. (6.15) are

$$\begin{aligned} \gamma_1 = \gamma_7 &= 0.78451361047755726381949763 \\ \gamma_2 = \gamma_6 &= 0.23557321335935813368479318 \\ \gamma_3 = \gamma_5 &= -1.17767998417887100694641568 \\ \gamma_4 &= 1.31518632068391121888424973 \end{aligned} \tag{6.19}$$

and

$$\begin{aligned} \gamma_1 = \gamma_9 &= 0.39216144400731413927925056 \\ \gamma_2 = \gamma_8 &= 0.33259913678925943859974864 \\ \gamma_3 = \gamma_7 &= -0.70624617255763935980996482 \\ \gamma_4 = \gamma_6 &= 0.08221359629355080023149045 \\ \gamma_5 &= 0.79854399093482996339895035 \end{aligned} \tag{6.20}$$

which are both methods of order $q = 6$. However, the method 6.20 is constructed by increasing the minimal number of stages and minimizing $\max_i |\gamma_i|$. This derivation yields smaller error coefficients (see [HLW06] and results of numerical experiments in chapter 7). In the following, these two methods are referred to as DISRK7 (for 6.19) and DISRK9 (for 6.20).

Please note that there are also other Runge-Kutta methods satisfying tableau 6.13. Those can be for example derived by solving the order equation of the Runge-Kutta methods subject to the special form of the tableau 6.13. An example can be found in [FQ10]

$$\begin{aligned} \gamma_1 &= -2.70309412 \\ \gamma_2 &= -0.53652708 \\ \gamma_3 &= 2.37893931 \\ \gamma_4 &= 1.8606818856 \end{aligned}$$

Other reference is [ZM15], where a DISRK method of order 5 has been constructed.

Essentially, the construction of a method with a composition according to theorem 6.2 provides a solution to the non-linear Runge-Kutta order equations. However, multiple solutions can exist for a non-linear equation system. But note that the theorem 6.1 holds for every DISRK method. This fact will be used in section 6.3.3 to derive the an algorithm for the SL scheme.

6.3.2 Quadrature with implicit midpoint rule

It has been shown, that we can obtain diagonally implicit symplectic Runge-Kutta methods of high order by composition and these methods fulfil problem 6.1. However the implicit midpoint scheme can also provide a solution to problem 5.1, if we also use it for the quadrature G^{Δ} in eq. (3.2). Recall that G^{Δ} is defined by

$$G^{\Delta}(x, \Delta t, A^n) = \Delta t g(y_x(\Delta t/2), a^n)$$

Now the collocation polynomial $p(\tau)$ of the implicit midpoint scheme for an autonomous system is defined with the conditions

$$\begin{aligned} p(0) &= x \\ \dot{p}(\Delta t/2) &= f(p(\Delta t/2)) \end{aligned}$$

and is therefore a polynomial of order 1

$$p(\tau) = x + \alpha\tau = x + f(p(\Delta t/2))\tau$$

$p(\tau)$ provides approximation of order $r = 2$ to the solution $y_x(t)$. To calculate G^{Δ} we have to evaluate this polynomial at $\Delta t/2$

$$p(\Delta t/2) = x + \frac{\Delta t}{2} f(p(\Delta t/2)) \tag{6.21}$$

Now, for the implicit midpoint rule, it holds

$$y_x^{\Delta t} = x + \Delta t k \Leftrightarrow k = \frac{y_x^{\Delta t} - x}{\Delta t}$$

and

$$k = f(x + (\Delta t/2)k)$$

Thus $x + (\Delta t/2)k$ is a solution to eq. (6.21)

$$x + \frac{\Delta t}{2} f(x + (\Delta t/2)k) = x + \frac{\Delta t}{2} f(x + (\Delta t/2)k)$$

and the evaluation of the collocation polynomial $p(\Delta t/2)$. This means, the quadrature formula is now of the form

$$G^{\Delta}(x, \Delta t, A^n) = \Delta t g(x + (\Delta t/2)k, a^n) \tag{6.22}$$

with the error order $q = 2$ (see section 5.1.3).

6.3.3 High order SL schemes with implicit midpoint rule

Next, based on section 6.3.1 and section 6.3.2, a SL scheme for the HJB equation is proposed. To simplify the notation consider a state vector

$$Y_x(t) \overset{\text{def}}{=} \begin{pmatrix} y_x(t) \\ \eta_x(t) \end{pmatrix} = \begin{pmatrix} y_x(t) \\ \int_0^t g(y_x(t))\,dt \end{pmatrix}$$

and the corresponding differential equation

$$\dot{Y}_x(t) = \begin{pmatrix} Ay_x \\ g(y_x(t)) \end{pmatrix} \overset{\text{def}}{=} F(t, Y) \tag{6.23}$$

Implicit midpoint rule applied to eq. (6.23) yields

$$Y_x^{\Delta t} = Y_x + \Delta t F\left(\frac{\Delta t}{2}, \frac{Y_x^{\Delta} - Y_x}{2}\right) \overset{\text{eq. (6.22)}}{=} Y_x + \Delta t \begin{pmatrix} A(y_x + (\Delta t/2)k) \\ g(y_x + (\Delta t/2)k) \end{pmatrix} \tag{6.24}$$

Altogether, application of the midpoint rule for the numerical integration and quadrature in eq. (3.3) can be considered as a joint implicit midpoint scheme for the extended state vector Y_x. Therefore, it is possible to use the composition from section 6.3.1 for the construction of higher order DISRK methods. Algorithm 1 illustrates the basic idea of an SL scheme with a DISRK method. For the sake of simplicity let us assume the optimal control $\alpha^\star$ to be known. X denotes the vector of grid points from the spatial discretization, i.e. for $X = \{x_1, \dots, x_M\}$

$$Y_X = \left(Y_{x_1} Y_{x_2} \dots Y_{x_M}\right)$$

with

$$Y_x = Y_x(0) = \begin{pmatrix} x \\ 0 \end{pmatrix}, \qquad k = \frac{y_x^{\Delta t} - x}{\Delta t}$$

Algorithm 1: SL scheme for the HJB equation with DISRK

```
Data: $\gamma_1, \dots, \gamma_s$ - composition coefficients, $T$ - end time, $\Delta t$ - time step
      $v_T$ - value function at T, $\alpha^\star$ - optimal control
      I - interpolation operator
Result: Approximation of the value function for $t = 0$
$t = T$
$V_{old} = v_T$
while $t > 0.0$ do
    for $j = 1, \dots, s$ do
        $Y_X^{\Delta t} = \Phi^M_{\gamma_j \Delta t}(Y_X, \alpha^\star(t))$
        $V = (Y_X)_2 + V_{old}((Y_X)_1)$        /* $(Y_X)_2 = G^{\Delta}(X, \Delta t)$, $(Y_X)_1 = y_X^{\Delta}$ */
        $V_{old} = V$
    end
    $t = t - \Delta t$
end
return V
```

The important part of the Algorithm 1 is the inner for-loop (lines 4 to 8), where the composition is calculated for given $\gamma_1, \dots, \gamma_s$. Please also note, that the value function V is updated after each step with $\gamma_i \Delta t$.

6.3.4 Optimization of the control and time inversion

Another important aspect of the SL scheme for the HJB equation is the approximation of the optimal control $\alpha^\star$. One approach is to use the gradient of the value function. Recall, that for a differentiable functions v the steepest slope is given by the ∇v. Therefore we can set $\alpha^\star = h(\nabla v)$, where h is the feedback operator (see [GK15]). In the numerical calculation, ∇v must be approximated by finite differences (e.g. central differences).

A more general approach is to generate a discretization of the control space and minimize by comparison.

Please note that complexity of both algorithms can be reduced for the SL schemes based on DISRK methods. It is obvious for the comparison minimization. For the gradient approach, the dimension of the stage value function ϑ_s^n from section 6.2.1 can be reduced to the dimension of the state space.

The numerical experiments in this theses use both methods (see chapter 7).

In the construction of higher order DISRK methods the coefficients γ_i also become negative. This is important from the viewpoint of the DPP eq. (2.14). Essentially this means, that the evaluation of the system eq. (2.10) is considered in the reverse time direction. Note that for a given dynamic function $f(y(t), \alpha(t))$, i.e. known $\alpha(s)$, the evaluation for negative time steps is straightforward. This was the assumption to apply the algorithm 1.

In the case of the SL scheme, α is a part of solution. Therefore, for each evaluation of the inner loop in the algorithm 1, we have to provide a control α_j^n, which is consistent to the previous iterations. Essentially this means, that the controlled system can be evaluated in both time direction. Note that this is important for the solution to achieve a higher order in time.

The rule for the computation of such a control sequence could not be derived in this work. Especially it is not clear how to calculate α_j^n with the comparison approach. Therefore the gradient minimization is used for the numerical experiments. This means, that in each iteration j the gradient of the stage value function ϑ_j^n is computed and evaluated by the feedback operator. The results of the numerical experiments for such a control sequence show a behaviour, which is consistent to the time order of the DISRK scheme. However, the theoretical basis has to be investigated in more details.

7 Results of numerical experiments

7.1 Control of the harmonic oscillator

This section presents the results of the approximation of the optimal control for a harmonic oscillator.

7.1.1 Experiment parameters

The system dynamics is defined by the following equation

$$\dot{y} = Ay + Bu, \quad A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{7.1}$$

The cost function is of the form

$$J_x(u(t)) = \frac{1}{2} y_x^T Q y_x + \frac{\alpha}{2} u^T R u, \quad Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} 1 \end{bmatrix} \tag{7.2}$$

In this special case there exists a solution for the infinite time horizon problem (see e.g. [FF11]). Figure 7.1 shows the exact solution of the value function.

The carried out numerical experiments approximate the value function for a finite time. In this case two reference solutions are considered. The fist one is computed with a finite differences approximation. The second one by solving the Riccati differential equation numerically (see section 7.1.3).

7.1.2 Parameters of test environment

The setting for experiment follows [GK15]. This means, the parameters are $\alpha = 0.1$ and discretization of the control space for the comparison approach is $d_c = 40$. The boundary condition defined by setting the second derivative of the value function to zero. This corresponds to a linear extrapolation of the values on innver nodes over the boundary. The experiments estimate the global error of the SL scheme to the reference solution.

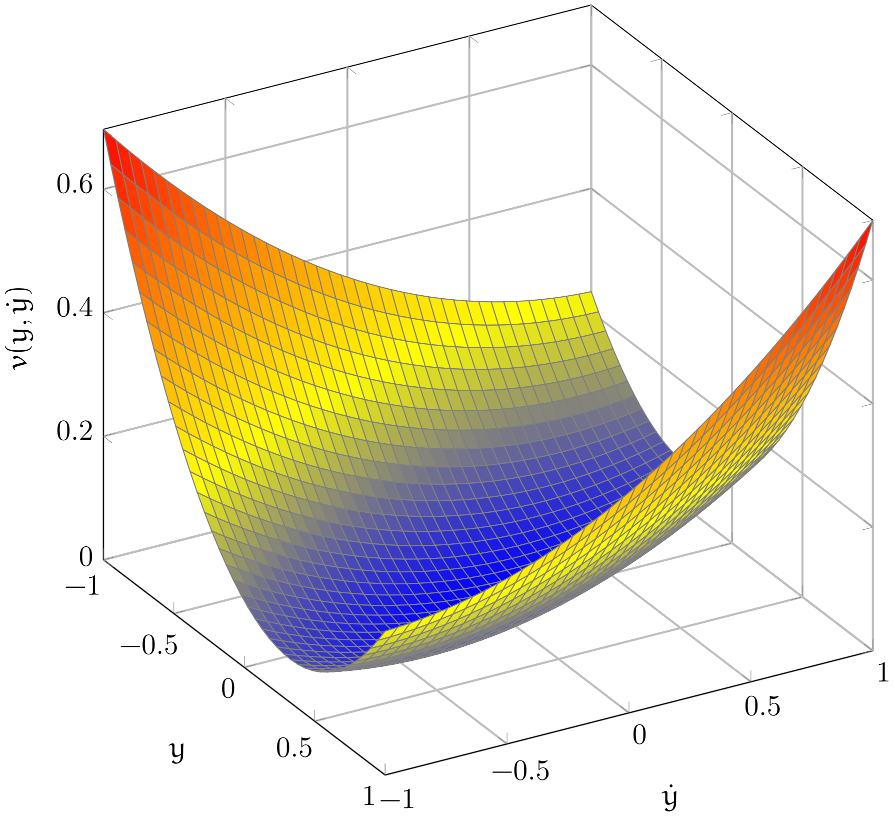


Figure 7.1: Value function for the infinite horizon harmonic oscillator control problem.

7.1.3 First experiments and modifications of test environment

The first numerical experiments were performed for the integration with Euler method and rectangular rule for the quadrature. The main goal of these tests was the verification of the test environment. The corresponding SL scheme is presented in Equation (3.4). Following [GK15], the integration time is set to $T = 1.0$. All test in this section use the compare optimization.

As we can see in fig. 7.2, the behaviour of the discretization error is in good agreement with the properties of the Euler method, which has the order 1. However, the error depends also on the spatial resolution.

Next test was performed for the SL scheme, which is based on the Heun method with the trapezoidal rule, cf. eq. (3.5). The results are presented in fig. 7.3. Again, the order of convergence is approximately 2.

The second experiment with the Heun integration scheme was performed with 20 CPUs in parallel. However, the overall computation time was approximately 1 week, compared to 1 day for the Euler scheme. for the control space discretization of $d_c = 40$. This illustrates the rising complexity of the optimization for the SL schemes with high order time methods.

The last test in this section was for the SL scheme with implicit Euler method for the numerical integration and quadrature, cf. fig. 7.4. The purpose was to verify the solution of the implicit equations for the stages of the Runge-Kutta scheme. Corresponding to the order of the underlying RK method, we can observe the convergence of order 1 w.r.t. the time step.

Subsequently, the first tests with SL schemes, which are based on the DISRK methods were performed. The results for the DISRK1 method are presented in fig. 7.5. However, especially for the higher order methods (DISRK7 and DISRK7) with high level spatial discretization, it was notices, that the error doest not decrease below the level of approximately 10^{-4}. This behaviour can also be observed in [GK15]. Therefore, a new reference solution was constructed to verify the solution which was computed by the FD scheme. For this purpose the differential Riccati equation was solved in time with an explicit Runge-Kutta method ($3/8$ rule) to obtain the value function for a given time. The convergence results for the new solution are presented in fig. 7.6. As we can see, the behaviour changes significantly for high spatial resolutions.

Another modification was the extension of the I/O-operations for the solver and postprocessor, as well as the scripts of the test environment (see chapter 8). For example, the generation of the compare points in the original environment was performed with the precision of 2 digits after the decimal point.

In the following, the modified test environment was used for the numerical experiments. To reduce the computation time, the optimization was performed with the gradient approach and the end time for the integration was set to $T = 0.1$.

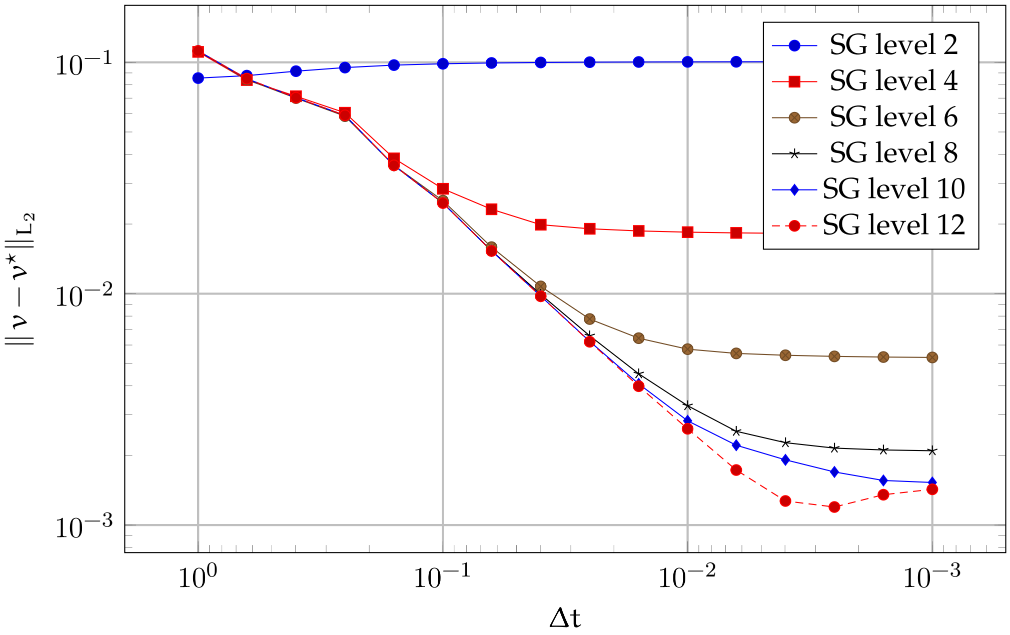


Figure 7.2: Error to the FD reference solution for explicit Euler and rectangular rule. $T = 1.0$.

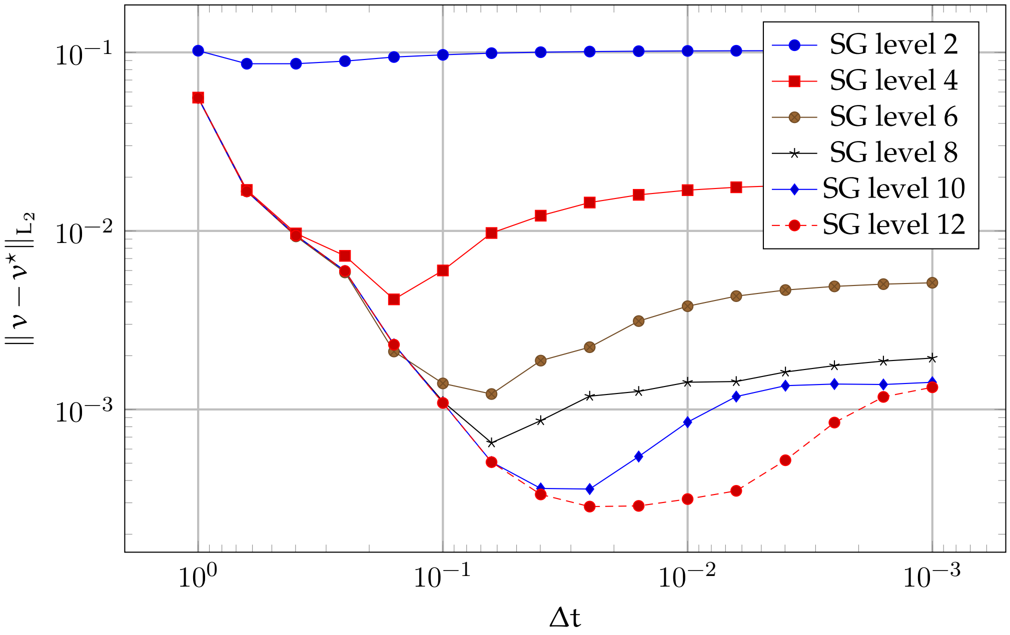


Figure 7.3: Error to the FD reference solution for Heun and trapezoid rule. $T = 1.0$.

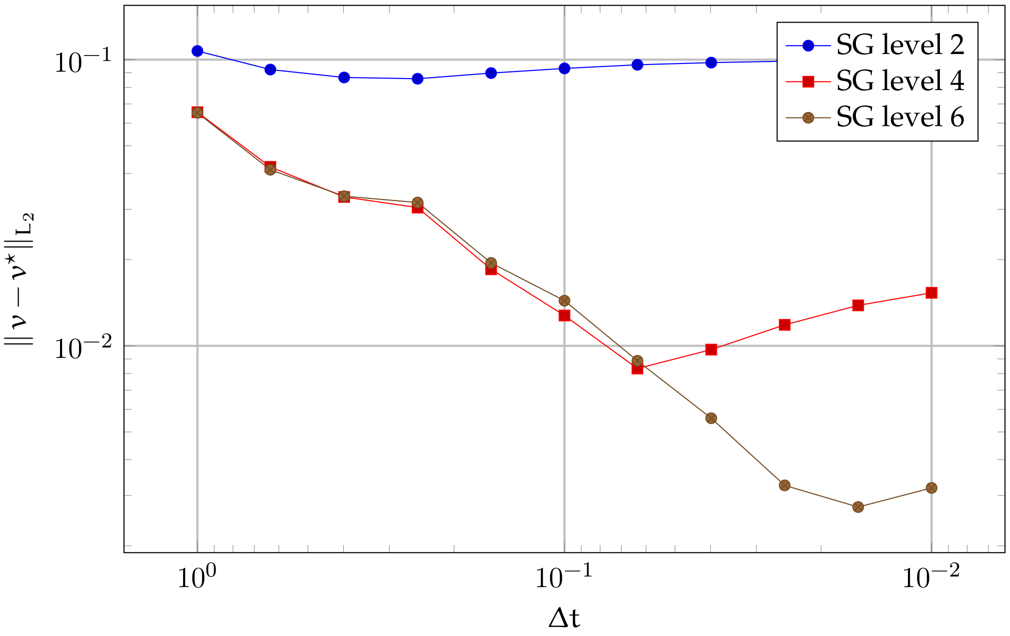


Figure 7.4: Error to the FD reference solution for implicit Euler. $T = 1.0$.

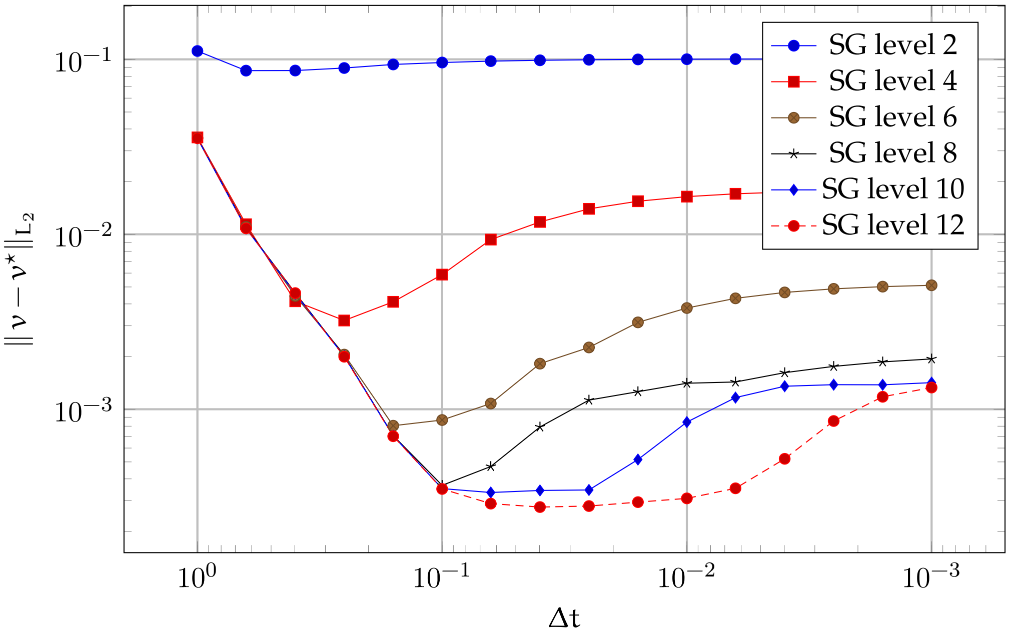


Figure 7.5: Error to the FD reference solution for DISRK1 and $T = 1.0$.

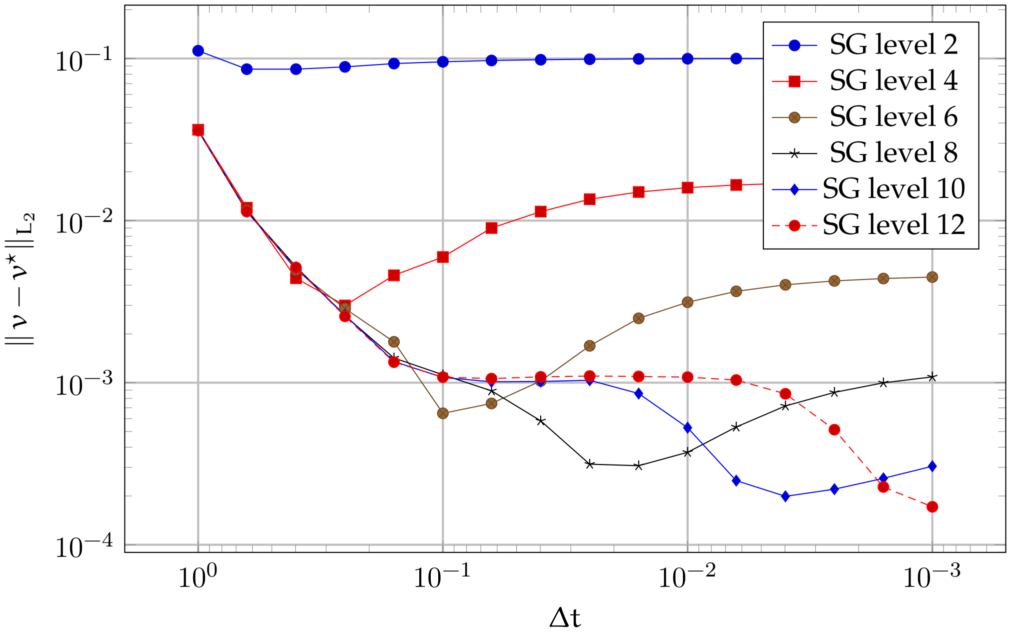


Figure 7.6: Global error to the matrix reference solution for DISRK1 for $T = 1.0$.

DISRK1

The results of the computation with DISRK1 method are presented in fig. 7.7. The global error for the large values of Δt (from 10^{-1} to 10^{-2}) shows the convergence of approximately order 2. This result corresponds to the order of the DISRK1 scheme. However, for small values of Δt the error an interesting pattern. A possible explanation is the interplay between spatial reconstruction and time integration, due to larger number of interpolation operations.

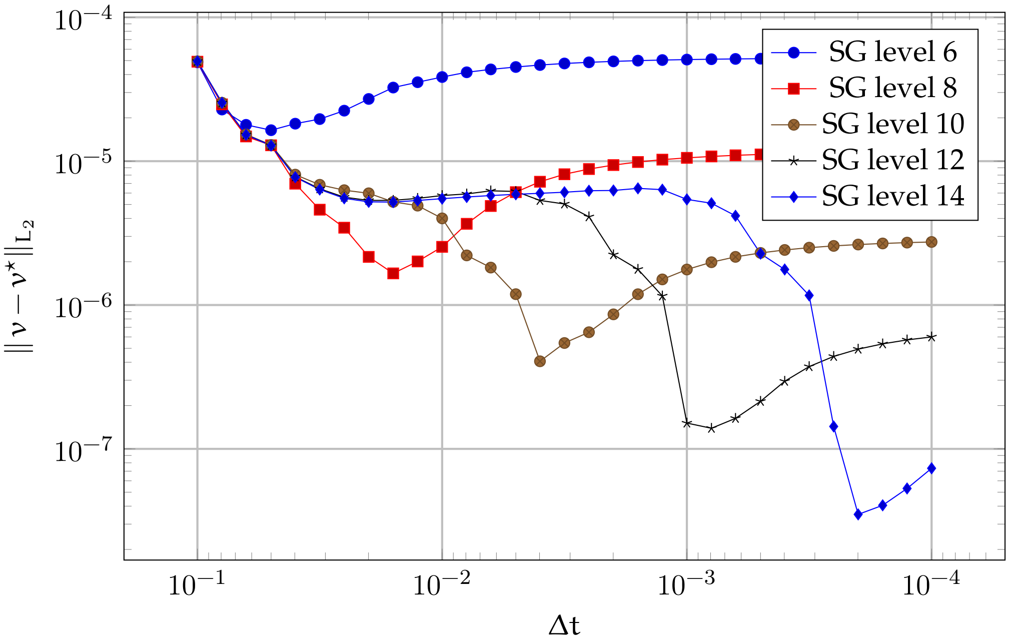


Figure 7.7: Error for DISRK1 and $T = 0.1$.

DISRK3

The results for the DISRK3 scheme have some similarity to the DISRK1 method, cf. fig. 7.8. Because of the time reasons it was not possible to obtain the complete the experiments for the SG level 14. Again, the convergence of the error for large values of Δt is approximately of order 4. Another interesting aspect is the local minimum by $10^{-1.5}$.

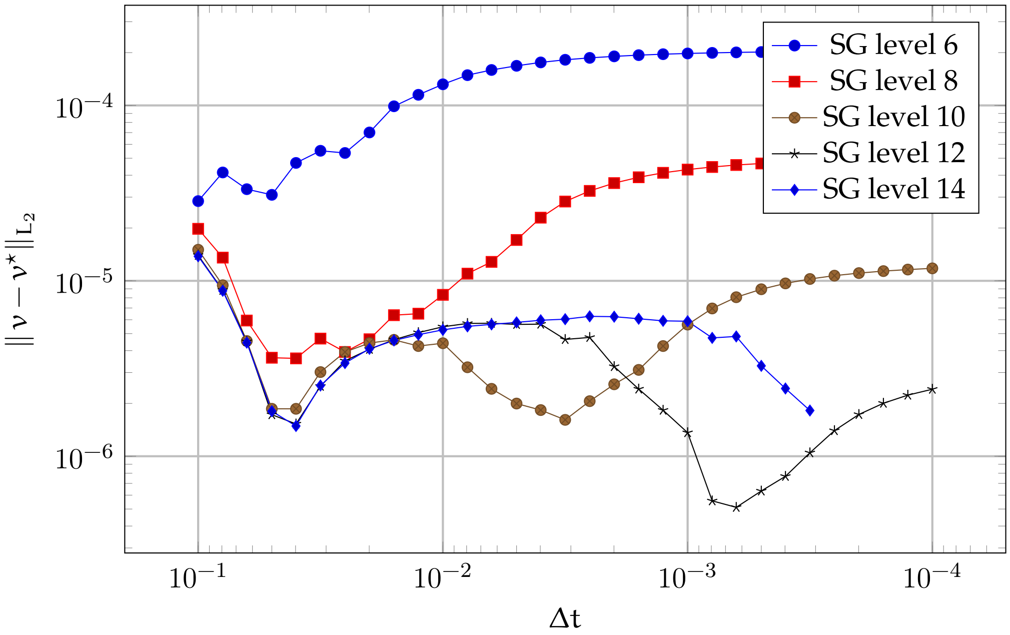


Figure 7.8: Error for DISRK3 and $T = 0.1$.

DISRK7

The results of the experiments for the DISRK7 are presented in fig. 7.9. Again, because of time reasons it was not possible to obtain all data. For the small time steps, the behaviour of the global error is similar to DISRK1 and DISRK3. Here, the convergence seems to be dominated by the error from the spatial discretization. The expected convergence order of p cannot be observed, due to small values of error for the $\Delta t = 10^{-1}$. This is also a main difference to the DISRK1 and DISRK3. Approximately the same error is obtained by the DISRK3 for the $\Delta t \approx 10^{-1.5}$. For the DISRK3 this value of Δ corresponds to 30 interpolations. On the contrary, for DISRK7, we achieve this value already for 7 spatial reconstructions (i.e. 1 time step).

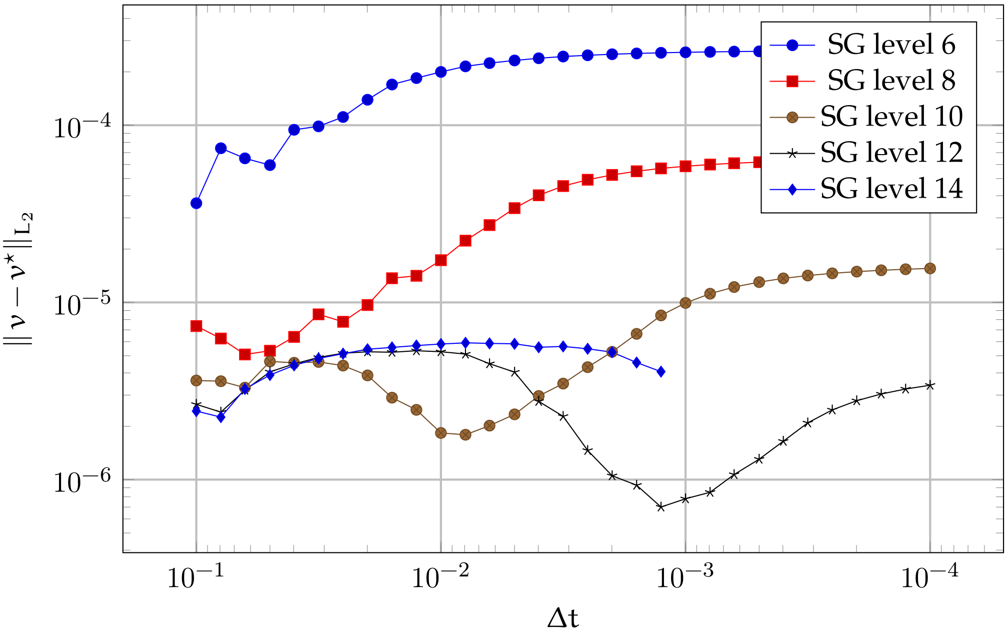


Figure 7.9: Error for DISRK7 and $T = 0.1$.

DISRK9

Figure 7.10 presents the results for the experiments with DISRK9 time discretization. The behaviour is similar to the DISRK7, however, the second local minimum for the error is obtained at slightly larger time steps (i.e., for the SG level 12, $10^{-2.8}$ for DISRK9 vs. approximately $10^{-3.0}$ for DISRK7).

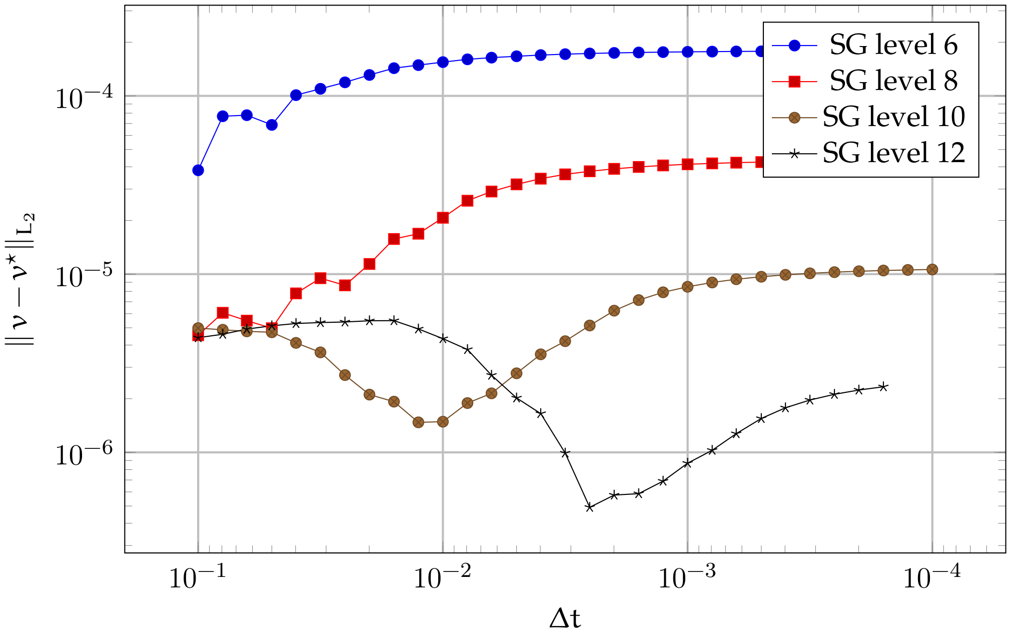


Figure 7.10: Error for DISRK9 and $T = 0.1$.

7.1.4 Adaptive time steps

Due to the time reasons and modifications of the test environment, it was not possible to verify the time-adaptive integration for the SL scheme. The obtained results are presented in this section.

The parameters for the controller eq. (5.5) are $\rho = 0.8$ and $q = 1.25$. The initial time step is set to $\Delta t = 10^{-3}$.

Essentially the same behaviour is observed for different methods cf. figure 7.11 to 7.13. This can be explained by the infeasibility of the harmonic oscillator example, at least for the considered tolerances TOL for the local error. However, we can see the adaptation of the step sizes w.r.t the TOL.

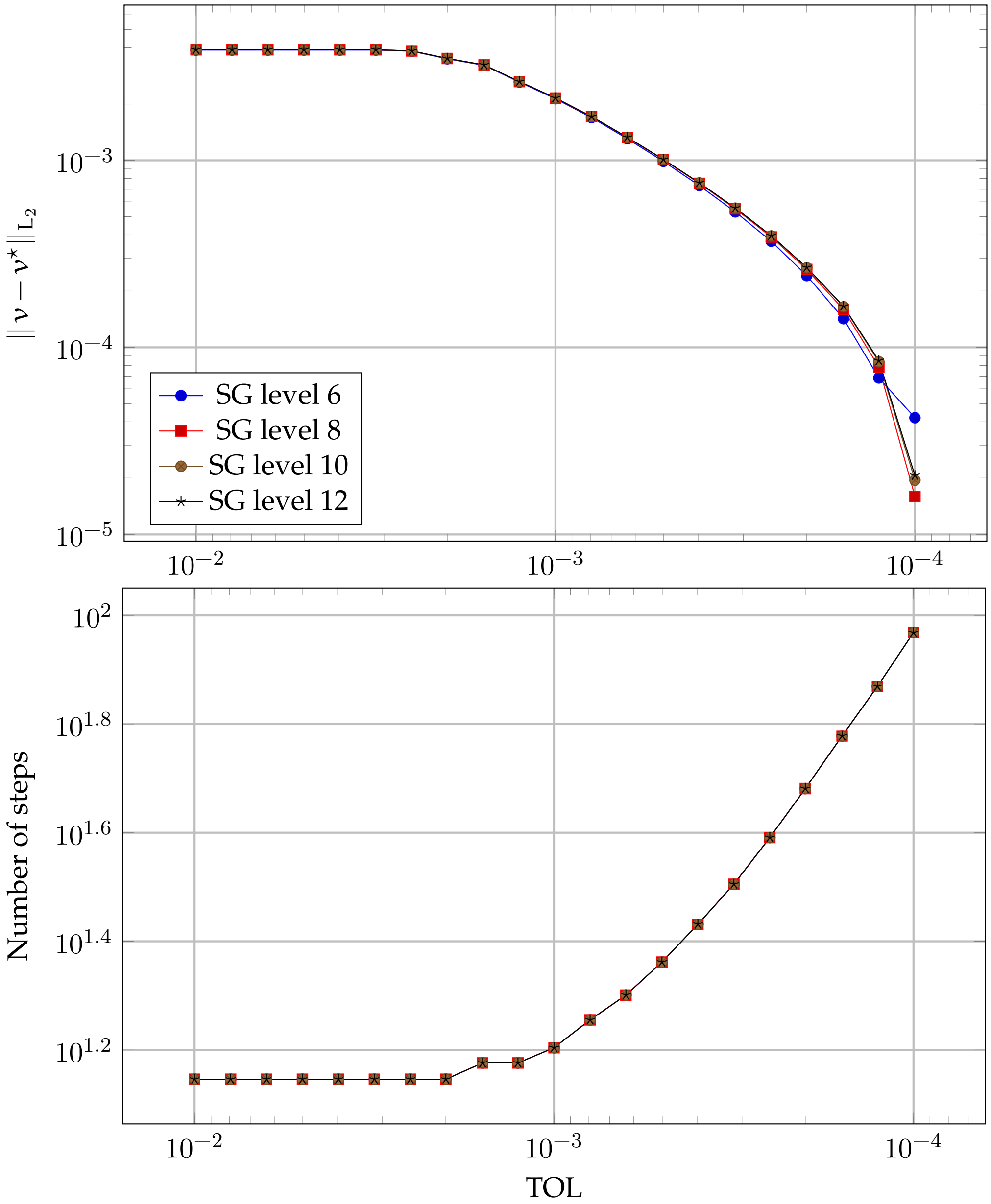


Figure 7.11: Error and number of steps for DISRK1 and $T = 0.1$.

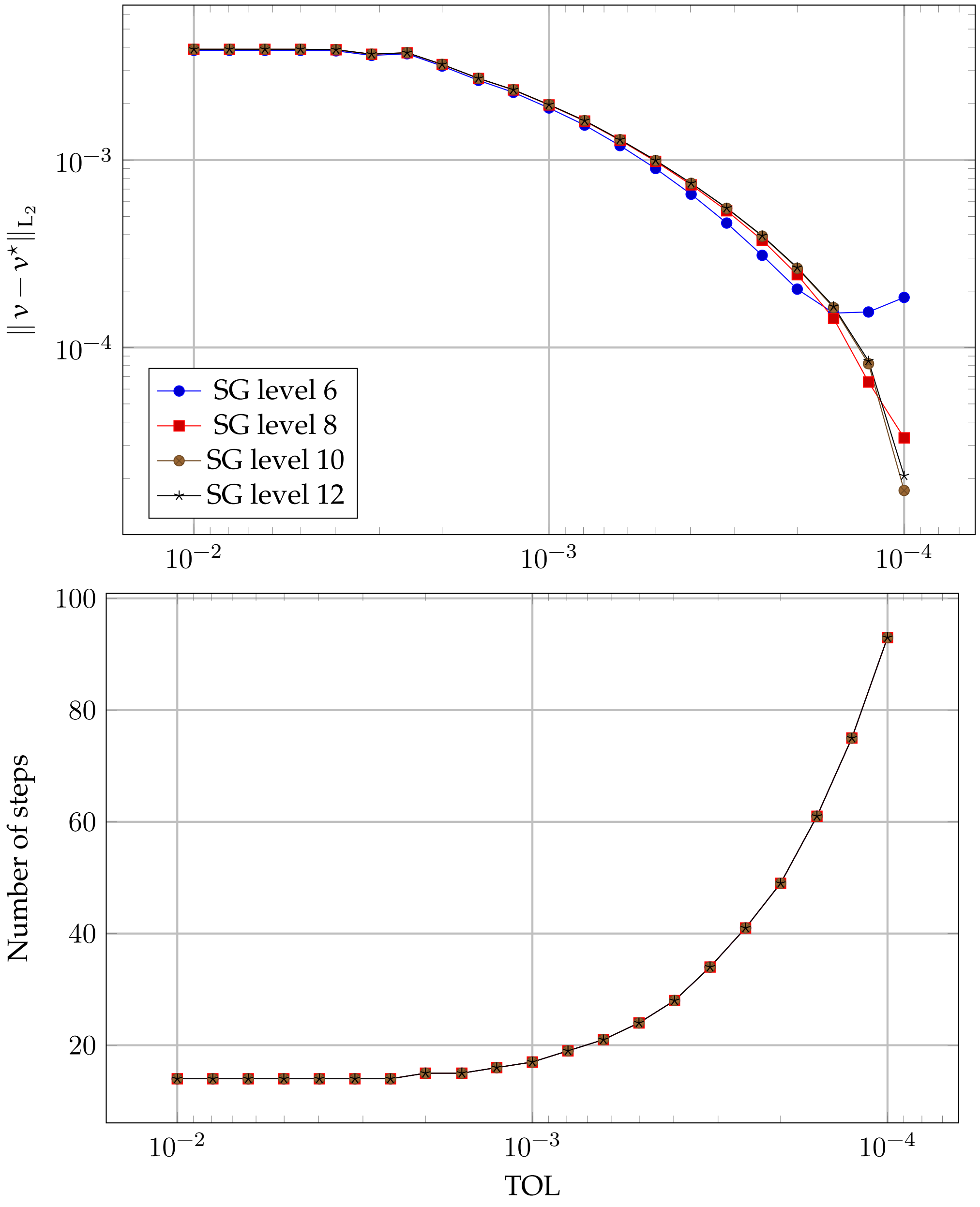


Figure 7.12: Error and number of steps for DISRK3 and $T = 0.1$.

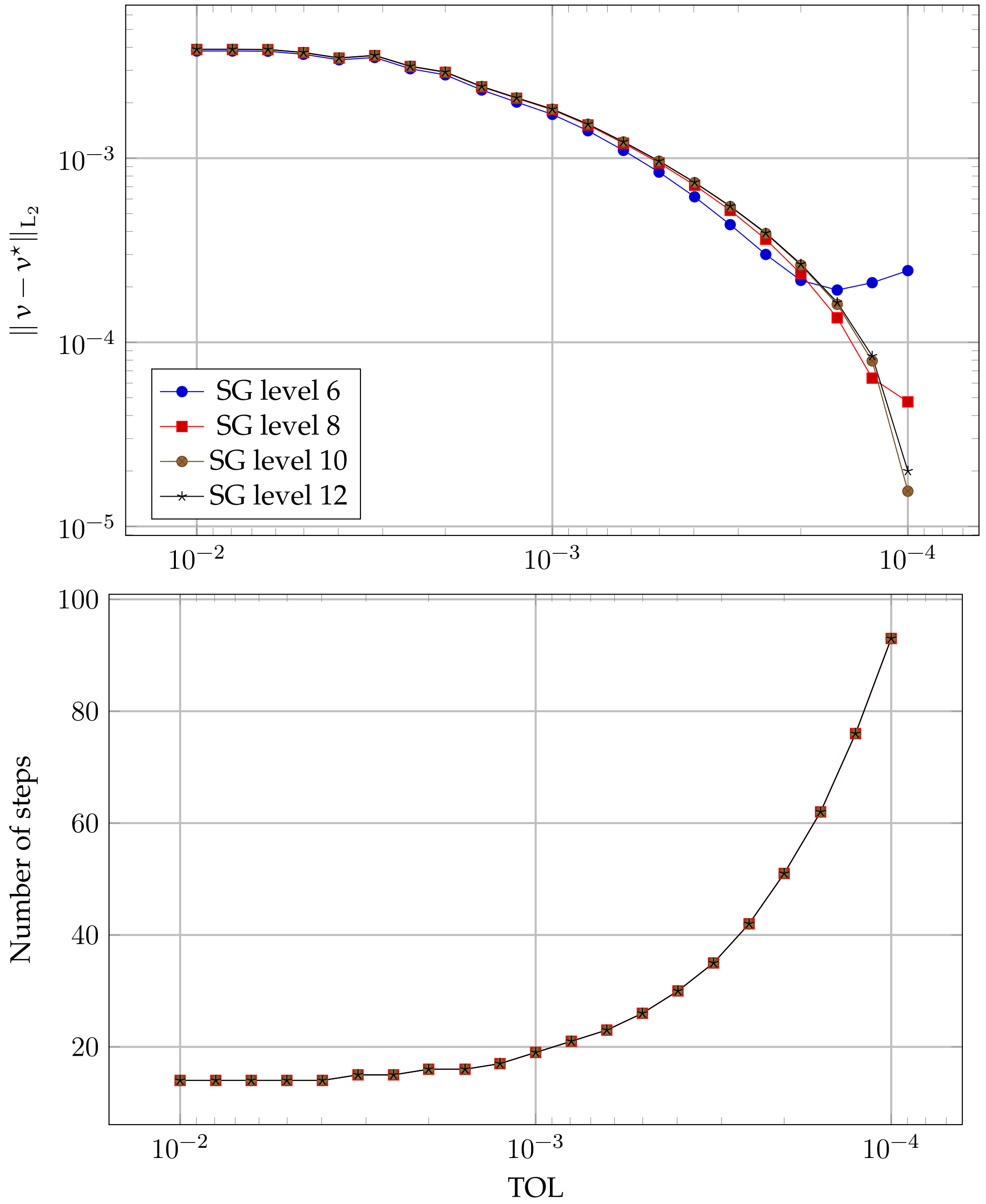


Figure 7.13: Error and number of steps for DISRK7 and $T = 0.1$.

7.2 Control of the semi-discrete wave equation

The second considered example is the control of the semi-discrete wave equation. The setting again follows [GK15]. The system dynamics is defined by

$$\dot{y} = Ay + Bu, \quad A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -0.49 & 0 & 0 & 0 \\ 0 & -1.97 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{7.3}$$

The cost function is

$$J_x(u(t)) = \frac{1}{2} y_x^T Q y_x + \frac{\alpha}{2} u^T R u, \quad Q = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{7.4}$$

The setting of the experiment follows [GK15]. The value of α is $\alpha = 0.1$. The time horizon is $T = 0.1$. The reference solution is constructed by solving the Riccati differential equation. The comparison to the reference solution was performed for an equidistant mesh with 10 points in each dimension, i.e. $10^{(}4)$ points.

The calculation of the solution for the 4 dimensional example is much more time consuming. For this reason, it was only possible to obtain the computations for the SG levels 6 and 8. The fig. 7.14 presents the results for the DISRK1. Again, we can observe approximately 2nd order convergence for large values of Δt. The results for DISRK3, cf. fig. 7.15 also shows the expected convergence order for the larger time steps. The behaviour for the small time steps cannot be explained without further investigation of the interplay between spatial and time discretization.

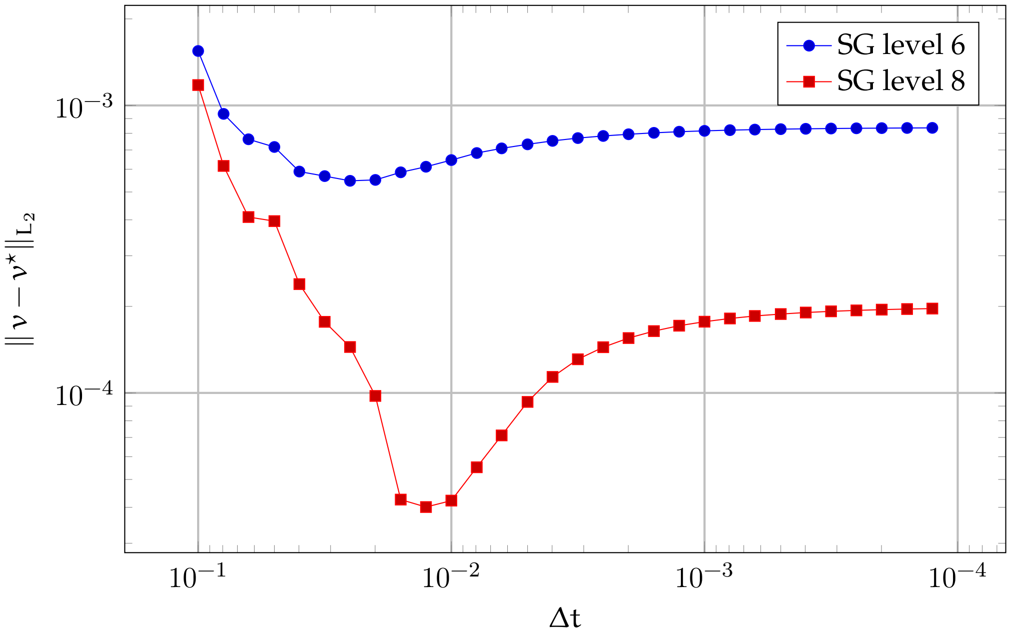


Figure 7.14: Error for DISRK1 and $T = 0.1$. 4D example.

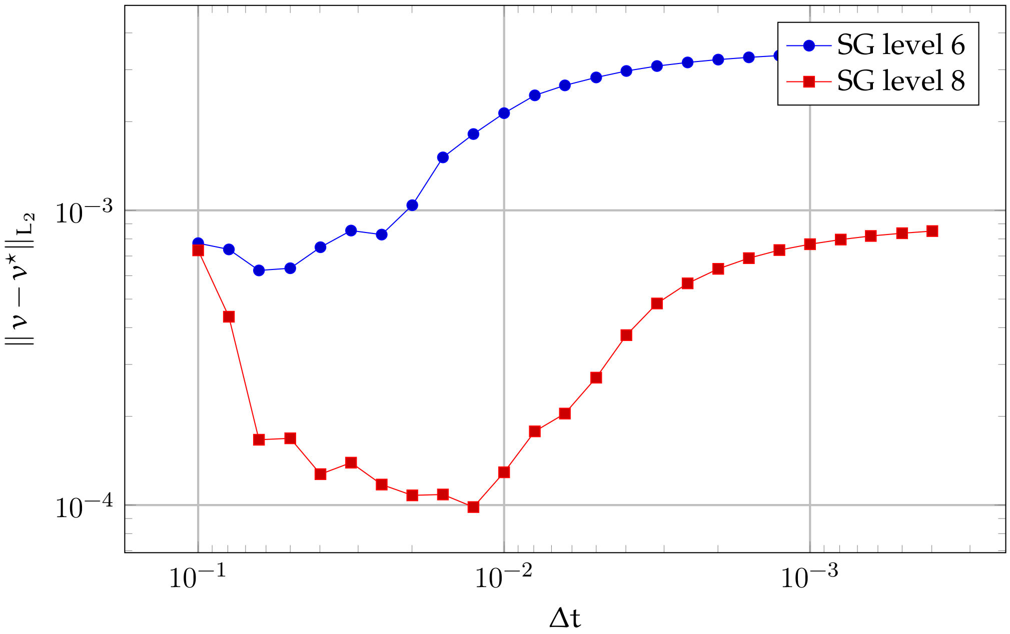


Figure 7.15: Error for DISRK3 and $T = 0.1$. 4D example.

8 Software

This chapter describes important aspects of the implementation for the SL scheme in this theses. The numerical experiments, which are presented in the chapter 7, are carried out with a C++-solver and post-processor. The testing environment is developed in PYTHON. Implementations are based on the RICAM-solver and PYTHON-scripts, which were provided by the Institute for Numerical Simulation at the University of Bonn and are used for numerical experiments in [GK15].

8.1 SG++

SG++ is an implementation of the sparse grids in C++, which is developed by the SG++ project (see [Pfl10]). The goal of the project is to provide algorithms for spatially adaptive sparse grids, which can be used straightforward for scientific computations. This work is based on the version 1.1.0 of SG++ (the version 2.0.0 was published on 20th May, 2016).

Some adaptations to SG++ were made in this theses. The most important is the extension for the data types DATAVECTOR and DATAMATRIX to provide additional linear algebra operations as well as iterator support. In addition, an extrapolation for the fold out grid class was implemented.

8.2 Automatization of numerical tests

The test environment in this work is based on PYTHON and especially the package SCITOOLS.MULTIPLELOOP. This extension allows to define parameter ranges and increments for the numerical experiments and then provides a discretization for the test space. A loop over all configuration of the parameters can be used to start specific experiments. A reference for the SCITOOLS.MULTIPLELOOP as well as PYTHON is [Lan04].

8.3 OpenMP

The optimization of the control for the HJB equation is rather time-consuming. However, the comparison algorithm is well suited for parallelization. OPENMP was used for this purpose. The numerical experiments were performed with up to 25 CPUs in parallel.

An important aspect of the implementation is the usage of the REDUCTION clause (see [Ope]). This method defines the synchronisation rule for the threads after a parallel code section.

From the version 4.0, the REDUCTION clause can be defined by the user. This is essential to work with SG++ data types like DATAVECTOR or DATAMATRIX (see section 8.1), which are not C inherent.

Bibliography

[BG04] BUNGARTZ, H.-J. ; GRIEBEL, M.: *Sparse grids*. Cambridge University Press, 2004 `http://wissrech.ins.uni-bonn.de/research/pub/griebel/sparsegrids.pdf`

[Bon04] BONAVENTURA, L.: *An introduction to semi-Lagrangian methods for geophysical scale flows*. ERCOFTAC Leonhard Euler Lectures SAM-ETH Zurich, 2004

[CIR52] COURANT, R. ; ISAACSON, E. ; REES, M.: On the solution of nonlinear hyperbolic differential equations by finite differences. In: *Comm. Pure Appl. Math.* 5 (1952)

[DB02] DEUFLHARD, P. ; BORNEMANN, F.: *Scientific Computing with Ordinary Differential Equations.* Springer, 2002

[DB08] DEUFLHARD, Peter ; BORNEMANN, Folkmar: *Numerische Mathematik 2. Gewöhnliche Differentialgleichungen.* de Gruyter, 2008

[Eva98] EVANS, L. C.: *Partial Differential Equations*. American Mathematical Society, 1998

[FF94] FALCONE, M. ; FERRETTI, R.: Discrete-time high-order schemes for viscosity solutions of Hamilton-Jacobi equations. In: *Numer. Math.* 67 (1994)

[FF06] FALCONE, M. ; FERRETTI, R.: A time adaptive Semi-Lagrangian approximation to mean curvature motion. In: *Numerical Mathematics Advanced Applications- ENUMATH2005* 67 (2006)

[FF11] FALCONE, M. ; FERRETTI, R.: *Semi-Lagrandgian Approximation Schemes for Linear and Hamilton-Jacobi Equations.* 2011

[FQ10] FENG, K. ; QIN, M.: *Symplectic Geometric Algorithms for Hamiltonian Systems.* Springer, 2010

[Gar13] GARCKE, J.: Sparse Grids in a Nutshell. Version: 2013. `http://dx.doi.org/10.1007/978-3-642-31703-3_3`. In: GARCKE, J. (Hrsg.) ; GRIEBEL, M. (Hrsg.): *Sparse grids and applications* Bd. 88. Springer, 2013. – DOI 10.1007/978–3–642–31703–3_3, S. 57–80. – extended version with python code `http://garcke.ins.uni-bonn.de/research/pub/sparse_grids_nutshell_code.pdf`

[Gar15] GARCKE, J.: *Lecture Notes for the Introduction to Numerics.* University of Bonn, 2015

[GK15] GARCKE, J. ; KRÖNER, A.: Suboptimal feedback control of PDEs by solving HJB equations on adaptive sparse grids. (2015)

[HLW06] HAIRER, E. ; LUBICH, C. ; WANNER, G.: *Geometric Numerical Integration. Structure-Preserving Algorithms for Ordinary Differential Equations.* Springer, 2006

[Lan04] LANGTANGEN, Hans P.: *Python Scripting for Computational Science.* Springer, 2004

[Ope] *OpenMP Application Program Interface.* : *OpenMP Application Program Interface*

[Pfl10] PFLÜGER, Dirk: *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. München : Verlag Dr. Hut, 2010 `http://www5.in.tum.de/pub/pflueger10spatially.pdf`. – ISBN 9783868535556

[Sha94] SHAMPINE, L. F.: *Numerical solution of ordinary differential equations.* Chapman & Hall, 1994

[SK09] SCHWARZ, H. R. ; KÖCKLER, N.: *Numerische Mathematik*. Vieweg+Teubner | GWV Fachverlage GmbH, Wiesbaden, 2009

[ZM15] Z.KALOGIRATOU ; MONOVASILIS, Th.: Diagonally Implicit Symplectic Runge-Kutta Methods with Special Properties. In: *Applied Mathematics & Information Sciences* 9, No. 1L, 11-17 (2015) (2015)