EFFICIENT HIGHER ORDER TIME DISCRETIZATION SCHEMES FOR HAMILTON-JACOBI-BELLMAN EQUATIONS BASED ON DIAGONALLY IMPLICIT SYMPLECTIC RUNGE-KUTTA METHODS

JOCHEN GARCKE AND ILJA KALMYKOV

REVISED VERSION 13.12.2017

ABSTRACT. We consider a semi-Lagrangian approach for the computation of the value function of a Hamilton-Jacobi-Bellman equation. This problem arises when one solves optimal feedback control problems for evaluationary partial differential equations. A time discretization with Runge-Kutta methods leads in general to a complexity of the optimization problem for the control which is exponential in the number of stages of the time scheme. Motivated by this, we introduce a time discretization based on Runge-Kutta composition methods, which achieves higher order approximation with respect to time, but where the overall optimization costs increase only linearly with respect to the number of stages of the employed Runge-Kutta method. In numerical tests we can empirically confirm an approximately linear complexity with respect to the number of stages. The presented algorithm is in particular of interest for those optimal control problems which do involve a costly minimization over the control set.

1. INTRODUCTION

In this work we consider the numerical solution of deterministic optimal control problems, with a focus on the discretization in time. In particular, we are interested in the numerical approximation of the value function stemming from the Dynamic Programming Principle, which connects to the characterization of the solution in terms of the Hamilton-Jacobi-Bellman (HJB) equation, which describes the closed-loop (or feedback) optimal control of evolutionary partial differential equations (PDEs).

We consider for example finite time horizon problems, for some time $T < \infty$, such as

$$v(x) \stackrel{\text{def}}{=} \inf_{\alpha \in \mathcal{A}} J_x(\alpha) = \inf_{\alpha \in \mathcal{A}} \int_0^T g(y_x(s,\alpha), \alpha(s))\, \mathrm{e}^{-\lambda s}\, ds, \tag{1}$$

with a function g as the running cost and $\lambda > 0$ the so-called discount factor. Here, $J_x(\alpha)$ is the cost functional and v the value function, which one can interpret as the optimal cost for a system trajectory starting from position x. The goal of an optimal control problem is to determine a control policy $\alpha^\star$ which minimizes the cost functional (1).

The underlying to be controlled dynamical system is given by

$$\begin{cases} \dot{y}(s) = f(s, y(s), \alpha(s)) \\ y(t_0) = x_0, \end{cases} \tag{2}$$

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

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

1991 *Mathematics Subject Classification.* 49J15, 49J20, 49L20, 65L06.

Key words and phrases. Optimal feedback control, Hamilton-Jacobi-Bellman equation, Dynamic Programming Principle, Runge-Kutta composition methods, diagonally implicit symplectic Runge-Kutta methods, sparse grids.

Given the set of admissible controls

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

we will consider the solution of the ODE (2) for a fixed α. In the following we call the set of all possible values of trajectories of (2) the state space. Note that the existence of the solution of the initial value problem (2) for a measurable α can be derived from the Carathéodory theorem, in particular we assume f is continuous and Lipschitz continuous with respect to y.

Now, to derive a numerical scheme for the solution of (1), one has to take several discretization steps to obtain an approximation of the value function. First is the spatial discretization, where one has to consider the complete domain in which the state dynamics are observed. Typically, this means that the computational effort rises exponentially with the dimension of the state space. One way to address this issue is a spatial discretization based on (adaptive) sparse grids as e.g. in [BGGK13, GK17].

In addition to the spatial discretization we have to introduce a discretization in time. Note here, that an important property of (1) is Bellman's Dynamic Programming Principle (DPP), which states that assuming the solution to equation (2) exists and is unique, then for all $x \in \mathbb{R}^d$ and $\tau > 0$ it holds

$$v(x) = \inf_{\alpha \in \mathcal{A}} \left\{ \int_0^\tau g(y_x(s, \alpha), \alpha(s)) \, \mathrm{e}^{-\lambda s} \, ds + \mathrm{e}^{-\lambda \tau} \, v(y_x(\tau, \alpha)) \right\}. \tag{3}$$

Based on employing the DPP piecewise, one can construct semi-Lagrangian schemes for the optimal control problem, as presented in the next section. To achieve that numerically, one has to consider the discretization of the state dynamics (2), which relates also to the numerical quadrature for the integral in (3). In this work, we are analysing higher order time discretizations for the state dynamics, in particular Runge-Kutta methods, and corresponding numerical quadrature approaches.

The determination of the optimal control α for a certain time interval now is the last ingredient of the numerical algorithm. The needed optimization here interacts with the time discretization. For simplicity, we consider the computation of the optimal control for a certain time by comparison over a discrete sample of the control set. If we now apply a Runge-Kutta method for the numerical integration, a representation of the control has to be provided for each stage. For the simple discretization of the control set with an arbitrary sampling this will lead to exponential complexity with respect to the number of sampling points when using standard Runge-Kutta schemes [FF94].

In this work, we introduce an approach which allows to overcome this drawback, i.e. we can achieve approximately linear complexity with respect to the costs of one minimization problem, e.g. the number of sampling points, even when using a Runge-Kutta scheme with several stages. For the approximation of the value function we use a semi-Lagrangian method [BGGK13, FF14, GK17]. The contribution of our work is based on an idea similar to Bellman's Dynamic Programming Principle, but applied to the numerical integration and quadrature parts within the semi-Lagrangian method. Therefore we use the DPP twice, as usual for the semi-Lagrangian scheme, but also for each iteration step of it, where we now separate the control optimization for the stages of the employed Runge-Kutta method. For this purpose we consider a special class of schemes based on Runge-Kutta composition methods for the time discretization, the so-called Diagonally Implicit Symplectic Runge-Kutta (DISRK) methods. These are compositions of the Implicit Midpoint rule and allow an overall scheme of linear complexity with respect to the number of control sampling points for one optimization problem.

Note that in the following we assume an autonomous dynamic in equation (2), i.e. for a non-autonomous system (2) we then consider the independent variable s as part of an extended state vector.

Furthermore, in the numerical examples we also consider an infinite horizon problem, i.e. we have the modified cost functional

$$J_x(\alpha) = \int_0^\infty g(y_x(s,\alpha),\alpha(s))\,\mathrm{e}^{-\lambda s}\,ds. \tag{4}$$

The finiteness of the integral (4) is guaranteed for a bounded g. In the following we will additionally assume that g is Lipschitz continuous.

The structure of the article is as follows. In Section 2 we describe the approximation of solutions to first order PDEs with semi-Lagrangian schemes. Section 3 provides an overview for the interpolation techniques with sparse grids, that will be used later for numerical tests. In Section 4 we discuss time discretization approaches for the semi-Lagrangian schemes. Section 5 describes the results of numerical tests.

2. Semi-Lagrangian scheme for the Hamilton-Jacobi-Bellman equation

The underlying idea of semi-Lagrangian schemes was first proposed for the advection equation in [CIR52]. A detailed derivation and survey can be found in [FF14]. This section gives a brief introduction to the topic.

2.1. Semi-Lagrangian schemes for optimal control problems.

The idea of a semi-Lagrangian scheme is the discretization of the integral representation of the value function (3). The basic building blocks are the approximation of the ODE (2) for ${y_x}^1$, the approximation of the integral for the running costs[1], and the spatial reconstruction of the value function at time τ. An additional numerical approximation has to be provided for the optimization over the control set $\mathcal{A}$.

In the first step we replace a trajectory of $y_x(s,\alpha)$ by a numerical solution of the ODE with an s-stage Runge-Kutta approach and consider a semi-discrete function, which is defined on the time grid $\{t_0,\dots,t_N\}$. We will in the following use superscripts to denote the time point of the discrete approximation, i.e. for the system dynamics we write

$$y_x^\nu = y_x(\nu,\alpha(\nu)), \quad \nu \in [t_0,T].$$

Furthermore we use y_x^{n+1} for $y_x^{t_{n+1}} = y_x(t_{n+1},\alpha(t_{n+1}))$ and set $\tau_n \overset{\text{def}}{=} t_{n+1}-t_n$ for $n=1,\dots,N$. Thus, an approximate solution of equation (2) is given by

$$\begin{cases} y_x^{n+1} = y^n + \tau_i \widetilde{\Phi}(y^n,\underline{a}^n,\tau_i) \overset{\text{def}}{=} \Phi(y^n,\underline{a}^n,\tau_i) \\ \quad y_x^0 = x \end{cases}, \tag{5}$$

where $\widetilde{\Phi}$ is the increment function and Φ the one-step map for some Runge-Kutta method. For a scheme with s stages one has s time points starting from t_n, then $\underline{a}^n$ denotes a set of control tuples $\underline{a}^n = \{a_0^n,\dots,a_{s-1}^n\} \subset A^s$. Note that the controls are a part of the numerical solution, i.e. we compute an element α^n of the control set $\mathcal{A}|_{[t_n,t_{n+1}]}$ (see Section 4) which is piecewise constant and defined by $\underline{a}^n$.

The other step of the numerical scheme relating to time is the approximation of the integral in equation (3), for which one can use a quadrature formula with q nodes. Consider an index set $J=\{0,\dots,q-1\}$. The approximation $G^\Delta(x,\tau)$ is given by

$$G^\Delta(x,\tau,\underline{a}^n) \overset{\text{def}}{=} \tau \sum_{i\in J} \omega_i g(y_x^{\nu_i},a_i^n)\,\mathrm{e}^{-\lambda\nu_i} \approx \int_t^{t+\tau} g(y_x(s,\alpha),\alpha(s))\,\mathrm{e}^{-\lambda s}\,ds, \tag{6}$$

[1] In the following we will refer to the numerical solution of a ODE as numerical integration, while we will refer to the numerical approximation of an integral as numerical quadrature.

where ν_i and ω_i are the nodes and weights of a quadrature formula, respectively, and

$$0 \leq \nu_i \leq 1, \quad \omega_i \geq 0, \quad \sum_{i \in J} \omega_i = 1.$$

Additionally, one needs a numerical approximation of the value function $v(x)$. For this purpose we consider a, for now unspecified, finite spatial grid composed of some points $\{x_1, \ldots, x_M\} \subset \Omega \subset \mathbb{R}^d$ and discrete function values $V_i = v(x_i)$. Furthermore, $I[V](y_x(t_{n+1}, \alpha))$ denotes a suitable interpolation of $v(y_x(t_{n+1}, \alpha))$ on Ω. Thus the spatial approximation is given by

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

Together, the semi-Lagrangian scheme can be written as

$$\begin{cases} v_j^n = \min\limits_{\underline{a}^n} \left\{ G^{\Delta}(x_j, \tau_n, \underline{a}^n) + I[V^{n+1}](x_j, \alpha^n) \right\} \\ v_j^N = u(x_j) \end{cases}. \tag{7}$$

In equation (7) v_j^n denotes the approximation of the value function for the time t_n at the spatial coordinate x_j. In particular we have $V_j^n = v_j^n$. The function $u(x)$ represents the terminating condition for the end time $T = t_N$.

Note that we use a comparison approach for the minimization in $\mathcal{A}|_{[t_n, t_{n+1}]}$. This means we consider a sampling A_Δ of the control domain A with R points and compare approximations of the value function at time point t_n for each element of the set

$$(A_\Delta)^s \stackrel{\text{def}}{=} \left\{ \left\{ a_0^n, \ldots, a_{s-1}^n \right\} : a_k^n \in A_\Delta,\, k = 0, \ldots, s-1 \right\}.$$

Furthermore, here and in the following we use the abbreviated notation

$$\min_{\underline{a}^n} \stackrel{\text{def}}{=} \min_{\underline{a}^n \in (A_\Delta)^s}.$$

Observe that in some situations, assuming the value function is differentiable, the minimization can be achieved by using the gradient of the value function. Generally, any minimisation procedure which uses only evaluations of the objective function, and not its derivatives, could be performed without changing the main steps of the scheme. In all cases, there is a fixed (or suitably bounded) cost d_c per individual minimization.

2.2. **Examples of semi-Lagrangian schemes.** Semi-Lagrangian schemes for the Hamilton-Jacobi-Bellman equation with different orders in time are presented in [FF94]. The simplest case is the combination of the Euler method for the ODE and rectangular rule for the quadrature. We get with $J = \{0\}$, $\nu_0 = 0$ and $\omega_0 = 1$

$$v_j^n = \min_{a_0^n} \left[\tau_n g(x_j, a_0^n) + I[V^{n+1}](x_j + \tau f(t_n, x_j, a_0^n)) \right]. \tag{8}$$

Note that in this section we use $I[V](x_j + \tau \widetilde{\Phi}(t_n, x_j, \underline{a}^n))$ instead of $I[V](x_j, \alpha^n)$ to emphasize the numerical integration scheme, with $\widetilde{\Phi}$ as introduced in equation (5).

A more sophisticated approach is to use the Heun formula for the solution of the ODE and the trapezoid rule for the quadrature. In this case we have $J = \{0, 1\}$, $\omega_0 = \omega_1 = 1/2$ and $\nu_0 = 0$, $\nu_1 = 1$. The complete scheme is of the form

$$\begin{aligned} v_j^n = \min_{a_0^n, a_1^n} \Bigg[& \frac{\tau_n}{2} g(x_j, a_0^n) + \frac{\tau_n}{2} g(x_j + \tau \widetilde{\Phi}(t_n, x_j, a_0^n, a_1^n, \tau), a_1^n) \\ & + I[V^{n+1}](x_j + \tau \widetilde{\Phi}(t_n, x_j, a_0^n, a_1^n, \tau)) \Bigg], \end{aligned} \tag{9}$$

with

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

Another example is the fourth order Runge-Kutta scheme RK4. The parameters are $J = \{0, 1, 2, 3\}$, $\omega_0 = \omega_2 = 1/6$, $\omega_1 = \omega_3 = 1/3$, $\nu_0 = 0$, $\nu_1 = \nu_2 = 1/2$, $\nu_3 = 1$. The resulting scheme is

$$y_x^{\nu_0} = x \qquad y_x^{\nu_1} = x + \tau \frac{k_0}{2}$$

$$y_x^{\nu_2} = x + \tau \frac{k_1}{2} \qquad y_x^{\nu_3} = x + \tau k_2$$

with

$$k_0 = f(t_n + \nu_0 \tau, x, a_0^n) \qquad k_1 = f(t_n + \nu_1 \tau, x + \tau \frac{k_0}{2}, a_1^n)$$

$$k_2 = f(t_n + \nu_2 \tau, x + \tau \frac{k_1}{2}, a_2^n) \qquad k_3 = f(t_n + \nu_3 \tau, x + \tau k_2, a_3^n)$$

$$\widetilde{\Phi}(t_n, x, a_0, a_1, a_2, a_3, \tau) = \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3)$$

and

$$v_j^n = \min_{a_0^n, a_1^n, a_2^n, a_3^n} \Bigg[\frac{\tau_n}{6} \Big(g(y_{x_j}^{\nu_0}, a_0^n) + 2g(y_{x_j}^{\nu_1}, a_1^n) + 2g(y_{x_j}^{\nu_2}, a_2^n) + g(y_{x_j}^{\nu_3}, a_3^n) \Big) + I[V^{n+1}](x_j + \tau \widetilde{\Phi}(t_n, x, a_0^n, a_1^n, a_2^n, a_3^n, \tau) \Bigg]. \tag{10}$$

The approach corresponds to the Butcher tableau

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \tag{11}$$

Remark 1. *As we can see in equations* (9) *and* (10)*, 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 $\mathcal{O}(d_c^2)$ evaluations of the trajectory for equation* (9)*. The complexity for the RK4 discretization is $\mathcal{O}(d_c^4)$ equation* (10)*. In general for an one-step method with s stages the complexity for the minimization is $\mathcal{O}(d_c^s)$.*

3. Sparse grids

The derivation of semi-Lagrangian schemes in the previous section requires spatial approximation of the solution at the time points t_n in each step. In particular, for the discretization of Bellman's DPP equation (3) 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 equation (7), e.g. Lagrangian, ENO, WENO, or FE interpolation, see [FF14]. We use sparse grids for the spatial approximation of the value function in more than one dimension. A detailed presentation and introduction to

this discretization approach can be found in [BG04, Gar13, Pfl10]. This section recalls the main ideas, where the representation follows [BGGK13, GK17].

For simplicity we consider a space domain $\Omega = [0,1]^d$, $d \in \mathbb{N}$. For a multi-index

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

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

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

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

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

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

In the following we consider two different sets of basis functions defined on the mesh $\Omega_{\underline{l}}$. The first one is constructed with linear hat functions

$$\varphi(x) = \max(1 - |x|, 0).$$

For a level l and an index j we define

$$\varphi_{l,j}(x) \stackrel{\text{def}}{=} \varphi(2^l x - j), \tag{12}$$

and d-dimensional piecewise d-linear hat functions

$$\varphi_{\underline{l},\underline{j}}(\underline{x}) \stackrel{\text{def}}{=} \prod_{t=1}^{d} \varphi_{l_t,j_t}(x_t), \tag{13}$$

where $\underline{x} = (x_1, x_2, \ldots, x_d) \in \Omega$.

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

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

For $V_{\underline{l}}$, we define hierarchical difference spaces $W_{\underline{l}}$ and write $V_{\underline{l}}$ as their discrete sum

$$W_{\underline{l}} \stackrel{\text{def}}{=} V_{\underline{l}} \setminus \bigoplus_{t=1}^{d} V_{\underline{l}-\underline{e_t}}, \qquad V_{\underline{l}} = \bigoplus_{\underline{k} \le \underline{l}} W_{\underline{k}}, \tag{14}$$

with $e_{\underline{t}}$ the t-th unit vector.

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

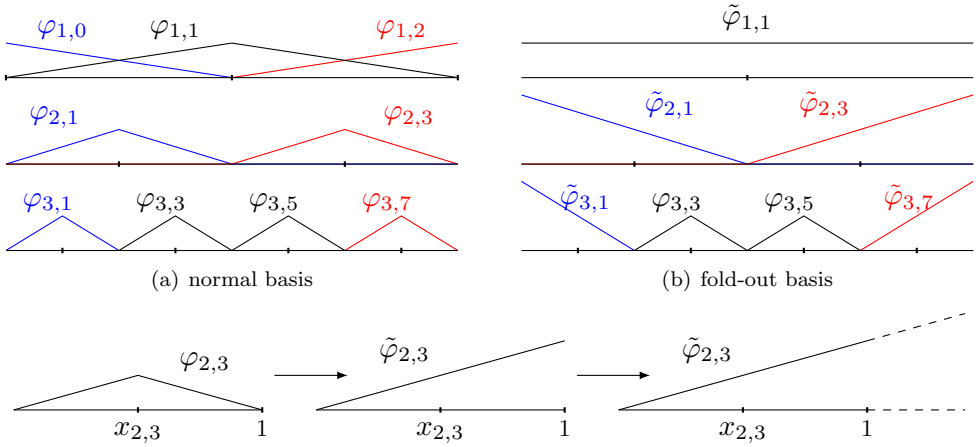
$$f(\underline{x}) = \sum_{\|\underline{l}\|_\infty \le n} \sum_{\underline{j} \in \mathcal{B}_{\underline{l}}} \alpha_{\underline{l},\underline{j}} \varphi_{\underline{l},\underline{j}}(\underline{x}) \tag{15}$$

where

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

The idea of a sparse grid is to take out those basis function $\varphi_{\underline{l},\underline{j}}$ which only have a small contribution to the representation of the interpolated function f [BG04, Gar13]. To measure this, we introduce the so-called Sobolev-space with dominating mixed derivative H^2_{mix} norm and the corresponding semi-norm:

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



(a) normal basis

(b) fold-out basis

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

FIGURE 1. Normal and fold out basis functions.

The function space

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

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

$$\|f_{\underline{l}}\|_2 \leq C(d) \cdot 2^{-2\|\underline{l}\|_1} |f|_{H^2_{mix}(\Omega)}, \tag{16}$$

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

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

The estimation (16) motivates the replacement of $\|\underline{l}\|_\infty \leq n$ in equation (15) by

$$\|\underline{l}\|_1 \leq n + d - 1. \tag{17}$$

The resulting sparse grid space

$$V_n^s \overset{\text{def}}{=} \bigoplus_{\|\underline{k}\|_1 \leq n+d-1} W_{\underline{k}} \tag{18}$$

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 full grids. This fact leads to a significant reduction of the computational complexity. Note also, that the error estimate for a function $f \in H^2_{mix}(\Omega)$ is

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

compared to a full grid, which has for $f \in H^2$, the classical Sobolev space,

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

For increasing level and dimension, the rule in equation (17) leads to a high number of points on the boundary of the domain in comparison to the inner area [Pfl10]. Here, an alternative is to use the so-called fold out ansatz function [Pfl10]. In this case, a sparse grid consists only of inner nodes and the reconstruction is extrapolated to and over the boundary, cf. Figure 1(c). An example of a sparse grid with fold out basis functions is given in Figure 2.

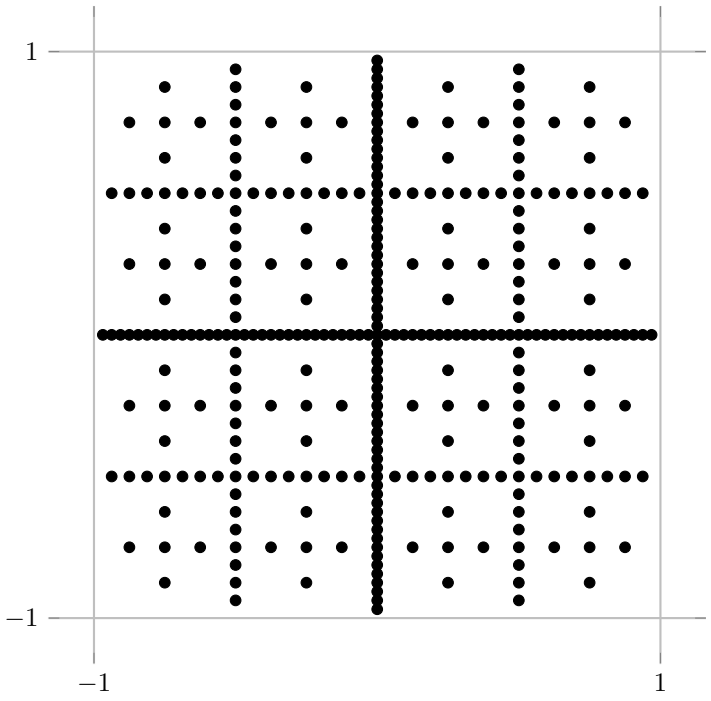


FIGURE 2. Sparse grid for level 6 on the domain $[-1,1]^2$ with fold out ansatz functions.

The second considered basis on Ω_l is constructed with B-splines. We provide a brief introduction to B-splines on sparse grids, following [Pfl10, VP16]. The definition of a cardinal B-spline, i.e. a B-spline with constant separation between nodes, is

$$b^0(x) \stackrel{\text{def}}{=} \chi_{[0,1)}(x), \qquad\qquad b^p(x) \stackrel{\text{def}}{=} \int_0^1 b^{p-1}(x-y)\,dy.$$

For level $l \in \mathbb{N}$ and an index $j \in I_l \stackrel{\text{def}}{=} \{1,3,5,\ldots,2^l-1\}$ we consider hierarchical B-splines $\varphi^p_{l,i} : [0,1] \to \mathbb{R}$ of the form

$$\varphi^p_{l,j}(x) \stackrel{\text{def}}{=} b^p\left(\frac{x}{h_l} + \frac{p+1}{2} - j\right)$$

with $h_l = 2^{-l}$. For $p = 1$ the $\varphi^p_{l,j}(x)$ is a piecewise linear basis function as in (12). The support of $\varphi^p_{l,j}(x)$ is given by

$$\operatorname{supp}(\varphi^p_{l,i}(x)) = [0,1] \cap [h_l\,(j-(p+1)/2)\,, h_l\,(j+(p+1)/2)]$$

and is larger than in the case of hat functions. This property leads in general to a larger computational effort for the sparse grids based on B-splines. However, this drawback is equalized by a higher approximation order of the B-splines with respect to the mesh size.

The d-dimensional hierarchical B-splines can be constructed analog to (13) with a tensor product approach. Similarly, the corresponding hierarchical B-spline subspaces are defined and (14) holds in this case as well. Therefore, one can use (18) to define a sparse grid space using the hierarchical B-spline spaces. See [VP16] for details, where additionally a B-spline basis with a modified behavior on the boundary is defined, which corresponds to the linear approach with fold out ansatz functions.

4. Semi-Lagrangian schemes with higher order time discretization

In this section we introduce a semi-Lagrangian scheme with higher order time discretization, which has only linear scaling with respect to the numerical costs for the optimization over the control set, a significant improvement in comparison to the approach outlined in Section 2.2. The main idea is to separate the optimization problem for each stage of the Runge-Kutta scheme. This approach is analog to the Dynamic Programming Principle (3).

An important aspect in the representation of the value function in equation (3) is the separation of the optimization problem in two parts. For a given initial state x we consider the contribution of the running costs up to the time τ and the remaining cost-to-go as the value function evaluated at $y_x(\tau, \alpha(\tau))$, i.e. $v(y_x(\tau, \alpha(\tau)))$.

The separation of the optimization problem is possible due to a rather general assumption on control, see Section 1 and [FF14]. Since we require the control to be a measurable function of time, it is possible to compose two different controls at some arbitrary time point $t^\star$. E.g. we can consider α_1, $\alpha_2 : [0, 2t^\star] \to \mathbb{R}$ and define $\alpha_3 [0, 2t^\star] \to \mathbb{R}$ to

$$\alpha_3(t) = \begin{cases} \alpha_1(t), & 0 \le t < t^\star, \\ \alpha_2(t), & t^\star \le t \le 2t^\star. \end{cases} \tag{19}$$

The property (19) has very strong consequences for the complexity of the minimization over the control set $\mathcal{A}$. Let us consider a finite time optimal control problem for a given initial state x

$$\min_{\alpha \in \mathcal{A}} J_x(\alpha) \tag{20}$$

with a cost functional [2]

$$J_x(\alpha) = \int_0^T g(y(s), \alpha(s))\, ds \tag{21}$$

and the corresponding value function as in (1). We define a restriction of (21) on some time interval $[t_i, t_j]$ with $0 \le t_i < t_j \le T$

$$J_x|_{[t_i,t_j]}(\alpha) \stackrel{\text{def}}{=} \int_{t_i}^{t_j} g(y(s), \alpha(s))\, ds. \tag{22}$$

The corresponding optimal control problem for $J_x|_{[t_i,t_j]}$ is

$$\min_{\alpha \in \mathcal{A}} J_x|_{[t_i,t_j]}(\alpha) \tag{23}$$

and we define (23) to be a subproblem of (20).

The set of minimizing control functions $\mathcal{A}_x|_{[t_i,t_j]}$ corresponding to $J_x|_{[t_i,t_j]}$ is

$$\begin{aligned} \mathcal{A}_x|_{[t_i,t_j]} &\stackrel{\text{def}}{=} \arg\min_{\alpha \in \mathcal{A}} J_x|_{[t_i,t_j]}(\alpha) \\ &= \left\{ \alpha^\star \in \mathcal{A} \,|\, \forall \alpha \in \mathcal{A} : J_x|_{[t_i,t_j]}(\alpha^\star) \le J_x|_{[t_i,t_j]}(\alpha) \right\}. \end{aligned}$$

Obviously, $\mathcal{A}_x|_{[0,T]}$ is the set of solutions to the original optimal control problem (20) on $[0, T]$ and, due to the property (19), we have

$$\mathcal{A}_x|_{[0,T]} \subset \mathcal{A}_x|_{[t_i,t_j]} \tag{24}$$

[2] To simplify the presentation we omit the term $e^{-\lambda s}$ in this section.

for all $0 \leq t_i \leq t_j < T$. Furthermore for the time intervals of the form $[t_i, T]$ with $t_i < T$ we can conclude

$$\mathcal{A}_x|_{[0,T]} \subset \mathcal{A}_x|_{[t_1,T]} \subset \mathcal{A}_x|_{[t_2,T]} \subset \ldots \subset \mathcal{A}_x|_{[t_{N-1},T]} \tag{25}$$

for an increasing sequence $0 = t_0 < t_1 < t_2 < \ldots < t_{N-1} < t_N = T$.

From the expression (25) we can derive two important properties in terms of the algorithmic complexity. The first one is called optimal substructure. A problem is said to have an optimal substructure if the optimal solution to the problem is contained within optimal solutions to its subproblems [CLRS09]. In the case of the optimal control problem (20) we consider as subproblems the finding of the set of minimizing control functions on each of the time intervals $[t_i, T]$. From (25) we obtain

$$\alpha^\star \in \mathcal{A}_x|_{[t_i,T]} \Rightarrow \alpha^\star \in \mathcal{A}_x|_{[t_j,T]}$$

for all $0 \leq t_i < t_j < T$, which shows, that the minimization of the control over $\mathcal{A}$ exhibits the optimal substructure.

The second property is the overlapping of subproblems, which means, that we can reuse the subproblems multiple times for the solution of the original problem. E.g. a set $\mathcal{A}_x|_{[t_j,T]}$ can be used for the computation of all sets $\mathcal{A}_x|_{[t_i,T]}$ with $0 \leq t_i < t_j < T$. For this purpose we can restrict the minimization of the cost functional $J_x|_{[t_i,T]}$ to the set $\mathcal{A}_x|_{[t_j,T]}$ to obtain $\mathcal{A}_x|_{[t_i,T]}$, i.e.

$$\min_{\alpha \in \mathcal{A}} J_x|_{[t_i,T]}(\alpha) = \min_{\alpha \in \mathcal{A}_x|_{[t_j,T]}} J_x|_{[t_i,T]}(\alpha).$$

Both properties — optimal substructure and overlapping subproblems — allow for an efficient minimization of the control. Due to the optimal substructure we can divide the original problem into less complex subproblems, i.e. we consider minimization over a smaller set of controls for each time interval $[t_i, T]$. The overlapping property, on the other hand, guarantees, that we have to solve each subproblem exactly once. Problems of this kind are said to have the Dynamic Programming property and can be computed efficiently, see [CLRS09]. The formulation of the DPP in equation (3) is an expression of this property for the value function (1).

4.1. **Composition methods.** In this section we derive a representation for the semi-Lagrangian scheme on time interval $[0, T]$ as a composition of semi-Lagrangian schemes for single time steps τ, which can be seen as a consequence of the Dynamic Programming property. Based on this observation, we then introduce Runge-Kutta methods obtained by composition approaches, in particular the Diagonally Implicit Symplectic Runge-Kutta (DISRK) methods. These methods will be used later to construct semi-Lagrangian schemes of higher order in time with linear complexity of the optimization problem. Let us first denote the semi-Lagrangian scheme (7) for the computation of the value function from Section 2 with step size τ_n by $\mathcal{S}^{\tau_n}$, i.e. for a spatial grid $\{x_1, x_2, \ldots, x_M\} \subset \Omega$ we have

$$\mathcal{S}^{\tau_n} : \mathbb{R}^{|\Omega|} \to \mathbb{R}^{|\Omega|}, \quad V^{n+1} \mapsto V^n$$

with

$$V_j^n = \left(S^{\tau_n}(V^{n+1})\right)_j \stackrel{\text{def}}{=} \min_{\underline{a}^n} \left\{ G^\Delta(x_j, \tau_n, \underline{a}^n) + I[V^{n+1}](x_j, \alpha^n) \right\}, \quad j = 1, \ldots, |\Omega|.$$

If we use the Dynamic Programming property, it is possible to express the discrete function values of $v(x)$, i.e. of the value function at time $t = 0$, as

$$V^0 = \mathcal{S}^{\tau_1}(\mathcal{S}^{\tau_2}(\ldots(\mathcal{S}^{\tau_N}(V^N))\ldots)) = \mathcal{S}^{\tau_1} \circ \mathcal{S}^{\tau_2} \circ \ldots \circ \mathcal{S}^{\tau_N}(V^N), \tag{26}$$

where $\{t_0, t_1, \ldots, t_N\}$ is a partition of $[0, T]$ with $0 = t_0 < t_1 < t_2 < \ldots < t_N = T$ and $\tau_i = t_{i+1} - t_i$ for $i = 1, \ldots, N$ are the corresponding step sizes.

We obtain expression (26) because, on the one hand, we can restrict the set of optimal controls after each time step τ_i without loosing the optimal solution for the complete time interval $[0, T]$, which is an expression of the optimal substructure of (25). On the other hand, we reuse the solution of the subproblems due to the overlapping in (25). Therefore we can express the semi-Lagrangian scheme for τ_i as a function of the semi-Lagrangian scheme for the time step τ_j with $i < j$. Altogether we can write the value function for $t = 0$ as a composition of semi-Lagrangian schemes applied to each time interval τ_i.

In the expression (26) each term of the form

$$\mathcal{S}^{\tau_i} \circ \mathcal{S}^{\tau_{i+1}} \circ \ldots \circ \mathcal{S}^{\tau_N} \tag{27}$$

can be considered a solution of the subproblem (23) on the time interval $[t_i, T]$ for the complete set of the grid points, i.e. $x \in \Omega$, whereby we associate the restricted value function

$$v|_{[t_i,T]}(x) \stackrel{\text{def}}{=} \min_{\alpha \in \mathcal{A}} J_x|_{[t_i,T]}(\alpha) \tag{28}$$

with the set of minimizers $\mathcal{A}_x|_{[t_i,T]}$.

The reduction of complexity due to the Dynamic Programming Principle is a clear advantage. The idea we present in this section is to mimic the Dynamic Programming property within a single step of a semi-Lagrangian scheme (7). As described in Section 2, in general the time discretization leads to an exponential complexity of the minimization problem. To overcome this drawback, we now consider Runge-Kutta methods with a structure similar to (26) and explain in the following how these methods can be used to construct a semi-Lagrangian scheme of higher order in time with linear scaling with respect to the numerical costs of the control minimization problem. In the following we will use the index of the form $\gamma_i\tau$ to denote parts of the numerical method within a single step of a semi-Lagrangian scheme, in contrast to the notation with index τ_i for the steps of the semi-Lagrangian scheme, see also Section 4.3. The construction is based on the following theorem from [HLW06].

Theorem 1. *Let Φ_τ, as defined in (5), be a one-step method of order p, i.e. the local error for Φ_τ is of order $\mathcal{O}(\tau^{p+1})$, with step size τ and*

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

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

$$\begin{aligned} \gamma_1 + \ldots + \gamma_s &= 1 \\ \gamma_1^{p+1} + \ldots + \gamma_s^{p+1} &= 0 \end{aligned} \tag{29}$$

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

With Theorem 1, the construction of higher order Runge-Kutta methods can be achieved by solving (29). However, the system (29) has only solutions for even p. Therefore in this work we consider methods derived from Theorem 1 by using the Implicit Midpoint Rule (with $p = 2$) as Φ_τ. These methods are called Diagonally Implicit Symplectic Runge-Kutta (DISRK) methods. Different schemes of this kind are given in [HLW06] and references therein. We recall that the Implicit Midpoint Rule has the following Butcher's tableau

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}. \tag{30}$$

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

$$\gamma_1 = \gamma_3 = \frac{1}{2 - 2^{1/(p+1)}}, \quad \gamma_2 = -\frac{2}{2 - 2^{1/(p+1)}} \tag{31}$$

with corresponding Butcher's tableau

$$
\begin{array}{c|ccc}
\frac{1}{2}a & \frac{1}{2}a & & \\
\frac{1}{2} & a & \frac{1}{2}-a & \\
1-\frac{1}{2}a & a & 1-2a & \frac{1}{2}a \\
\hline
 & a & 1-2a & a
\end{array} \tag{32}
$$

where $a = \dfrac{1}{2-2^{1/(p+1)}} \overset{(p=2)}{=} 1.35120719195966$, see [FQ10].
In the following, we refer to this method as DISRK_3. Note that it is not necessary to calculate the Butcher's tableau of the composition DISRK explicitly. The construction of the scheme allows to apply the implicit midpoint methods $\Phi_{\gamma_i\tau}$ with time steps $\gamma_i\tau$ given by equation (29) in a loop. Thus, the method in tableau (32) is equivalent to

$$\Phi_\tau^{\text{DISRK}_3} = \Phi_{\gamma_3\tau}^M \circ \Phi_{\gamma_2\tau}^M \circ \Phi_{\gamma_1\tau}^M$$

with γ_i from (31) and Φ_τ^M denotes the Implicit Midpoint Rule with step size τ.

As we are not restricted in the number of the composition steps, other solutions for an order 4 method are possible. E.g. we can set

$$\gamma_1 = \gamma_2 = \gamma_4 = \gamma_5 = \frac{1}{4-4^{1/(p+1)}}, \quad \gamma_3 = -\frac{4^{1/(p+1)}}{4-4^{1/(p+1)}} \tag{33}$$

The associated method will be denoted by DISRK_5.

Other examples for solutions to equation (29) are

$$\begin{aligned}
\gamma_1 = \gamma_7 &= 0.78451361047755726381949763 \\
\gamma_2 = \gamma_6 &= 0.23557321335935813368479318 \\
\gamma_3 = \gamma_5 &= -1.17767998417887100694641568 \\
\gamma_4 &= 1.31518632068391121888424973,
\end{aligned} \tag{34}$$

and

$$\begin{aligned}
\gamma_1 = \gamma_9 &= 0.39216144400731413927925056 \\
\gamma_2 = \gamma_8 &= 0.33259913678925943859974864 \\
\gamma_3 = \gamma_7 &= -0.70624617255763935980996482 \\
\gamma_4 = \gamma_6 &= 0.08221359629355080023149045 \\
\gamma_5 &= 0.79854399093482996339895035.
\end{aligned} \tag{35}$$

The methods are both of order 6. However, the method (35) 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 Section 5. In the following, we refer to DISRK_7 for (34) and DISRK_9 for (35).

The following method is of order 8 and will be denoted by DISRK_{17}.

$$
\begin{aligned}
\gamma_1 = \gamma_{17} &= 0.13020248308889008087881763 \\
\gamma_2 = \gamma_{16} &= 0.56116298177510838456196441 \\
\gamma_3 = \gamma_{15} &= -0.38947496264484728640807860 \\
\gamma_4 = \gamma_{14} &= 0.15884190655515560089621075 \\
\gamma_5 = \gamma_{13} &= -0.39590389413323757733623154 \\
\gamma_6 = \gamma_{12} &= 0.18453964097831570709183254 \\
\gamma_7 = \gamma_{11} &= 0.25837438768632204729397911 \\
\gamma_8 = \gamma_{10} &= 0.29501172360931029887096624 \\
\gamma_9 &= -0.60550853383003451169892108.
\end{aligned} \tag{36}
$$

Other methods with different properties are presented in [HLW06]. Essentially, the construction of a method with a composition according to Theorem 1 provides a solution to the non-linear Runge-Kutta order equations. However, multiple solutions can exist for the non-linear equation system.

4.2. **Semi-Lagrangian schemes with Implicit Midpoint Rule.** As emphasized in Section (2.2), using standard Runge-Kutta methods the complexity for optimizing the control is exponential in the number of the employed stages of the considered Runge-Kutta methods. To overcome this drawback, we are seeking to utilize the Dynamic Programming Principle to obtain a representation of a semi-Lagrangian scheme which is analog to (26). An important ingredient to achieve such a representation is the time discretization with DISRK methods.

In particular, we now derive a semi-Lagrangian scheme based on the Implicit Midpoint Rule. Subsequently, we will use this scheme analog to the Implicit Midpoint Rule in the derivation of the DISRK methods as an elementary building block and apply Theorem 1 to construct schemes of higher order in time.

Note that Theorem 1 can directly be used with the Implicit Midpoint Rule. However, the time discretization in a semi-Lagrangian scheme consists of two parts — numerical integration of the ODE system (5) and numerical quadrature of the integral for the running costs (6). Furthermore, the goal of the semi-Lagrangian scheme is to approximate the value function (1), not the state of the controlled system (2). This means, that to use Theorem 1, we have to analyse the discretization error for the value function in (7) with respect to time. For that we will consider the approximation of the running costs

$$\int_0^\tau g(y_x(s), \alpha(s))\, ds$$

as well as of the costs-to-go $v(y_x(\tau))$. In the following we present an approach to compute both quantities with a global approximation order of $p = 2$.

To ease the presentation we will combine the numerical integration and quadrature, i.e. let us consider the state vector $Y_x(t) \in \mathbb{R}^{d+1}$ in the form

$$Y_x(t) = \begin{pmatrix} Y_{x,1}(t) \\ Y_{x,2}(t) \end{pmatrix} \stackrel{\text{def}}{=} \begin{pmatrix} y_x(t) \\ \eta_x(t) \end{pmatrix} = \begin{pmatrix} y_x(t) \\ \int_0^\tau g(y_x(t))\, dt \end{pmatrix}, \tag{37}$$

with the corresponding differential equation

$$\dot{Y}_x(t) = F(t, Y) \stackrel{\text{def}}{=} \begin{pmatrix} f(y_x(t), \alpha(t)) \\ g(y_x(t)) \end{pmatrix}. \tag{38}$$

The Implicit Midpoint Rule with step size τ now yields, denoting $Y_x \stackrel{\text{def}}{=} Y_x(0)$,

$$Y_x(\tau) \approx Y_x^\tau = Y_x + \tau K, \tag{39}$$

with implicitly defined $K \in \mathbb{R}^{d+1}$

$$K = F\left((\tau/2), Y_x + (\tau/2)K\right).$$

From equation (38) we can derive for $k_y \in \mathbb{R}^d$ and $k_\eta \in \mathbb{R}$

$$K = \begin{pmatrix} k_y \\ k_\eta \end{pmatrix} = \begin{pmatrix} f(y_x(0) + (\tau/2)k_y, \alpha((\tau/2))) \\ g(y_x(\tau/2), \alpha(\tau/2)) \end{pmatrix}.$$

The last equality is due to g being independent of η. Thus we get

$$\begin{aligned} k_y &= f(y_x(0) + (\tau/2)k_y, \alpha(\tau/2)) = \frac{y_x^\tau - y_x(0)}{\tau} = \frac{y_x^\tau - x}{\tau}, \\ k_\eta &= g(y_x(\tau/2), \alpha(\tau/2)), \end{aligned}$$

where k_y is the solution of the first stage equation for the Implicit Midpoint Rule applied to the state dynamic (2) and y_x^τ is the approximation to the solution $y_x(\tau)$ by the Implicit Midpoint Rule.

With this joint formulation of the two time discretization aspects of one step of a semi-Lagrangian as one time discretization task, we pose the underlying approximation result.

Theorem 2. *We consider an approximation v_j^n of the value function $v(x)$, as defined in (1), by a semi-Lagrangian scheme in the form of (7) using the Implicit Midpoint Rule. Neglecting the impact of the spatial reconstruction operator I for the approximation of the costs-to-go $v(y_x(\cdot))$, the local time discretization error, i.e. for one semi-Lagrangian step, of the scheme is $\mathcal{O}(\tau^3)$. Based on Theorem 1 we can therefore, using $p = 2$, construct higher order composition schemes.*

Proof. We first consider the discretization error for the running costs. It is clear, that by the approach (39) we can compute the state of the controlled system, i.e. $Y_{x,1}$, with error order $p = 2$. However, for the computation of the running costs $Y_{x,2}$ we need an evaluation of $g(y_x(\tau/2), \alpha(\tau/2))$ and thus an approximation of $y_x(\tau/2)$. We use k_y from the calculation of $Y_{x,1}$ for a linear approximation of $y_x(\tau/2)$:

$$y_x(\tau/2) \approx x + (\tau/2)k_y = \frac{x + y_x^\tau}{2}. \tag{40}$$

Note that this can also be interpreted as the Implicit Euler Rule applied to approximate the value of y_x at $\tau/2$, i.e. the implicitly defined stages

$$k_y = f(y_x(0) + (\tau/2)k_y, \alpha(\tau/2))$$

have the same value for the Implicit Midpoint Rule with step size τ and the Implicit Euler Rule with step size $\tau/2$.

As stated in Section 2 we consider a piecewise constant control $\alpha \in \mathcal{A}$. From the viewpoint of time discretization, we consider an exact evaluation of the control function $\alpha(s)$ at the time point $s = \tau/2$. By this interpretation, we will not obtain any additional error with respect to $\alpha(s)$ due to the time discretization with the Implicit Midpoint Rule (39). However, we will achieve in general only a suboptimal solution in the minimization problem over the control set $\mathcal{A}$.

The approach (40) for the calculation of $y_x(\tau/2)$ leads to a linear approximation for $g(y_x(\tau/2), \alpha(\tau/2))$ if we assume $\partial g(x,y)/\partial x$ to be bounded by some constant C_g on the computational domain Ω, respectively $g(x,y)$ to be Lipschitz continuous with respect to the first argument as stated in Section 1. To estimate the error we can develop $g(x + (\tau/2)k_y, \alpha(\tau/2))$ into a Taylor

series around $y_x(\tau/2)$ and use, with $\xi \in [y_x(\tau/2), x + (\tau/2)k_y]$, the mean-value form of the remainder:

$$\begin{aligned} &g(y_x(\tau/2), \alpha(\tau/2)) - g(x + (\tau/2)k_y, \alpha(\tau/2)) \\ &= g(y_x(\tau/2), \alpha(\tau/2)) - g(y_x(\tau/2), \alpha(\tau/2)) \\ &- \underbrace{\partial g(\xi, \alpha(\tau/2))/\partial x}_{\leq C_g} \underbrace{(x + (\tau/2)k_y - y_x(\tau/2))}_{\mathcal{O}(\tau^2)}. \end{aligned} \tag{41}$$

The $\mathcal{O}(\tau^2)$ estimate is due to the linear approximation (40). If we plug $g(x + (\tau/2)k_y, \alpha(\tau/2))$ into equation (39) instead of k_η, we get a local error of $\mathcal{O}(\tau^3)$ in the approximation of $\eta_x(\tau)$ due to the multiplication of K with τ.

So far we have an approximation of the state dynamics and of the running costs of order $\mathcal{O}(\tau^2)$. As outlined in the beginning of this section, our goal is to apply Theorem 1 to the computation of the value function. Besides the calculation of the running costs, the semi-Lagrangian scheme (7) consists of the approximation for the costs-to-go $v(y_x(\tau))$. To estimate the local discretization error for the evaluation of $v(y_x(\tau))$ with respect to time we can provide an argument similar to equation (41). Here we neglect the impact of the control discretization and assume the optimal control $\alpha^\star$ to be known. Assuming that $v'(x)$ is bounded on Ω we can estimate

$$\begin{aligned} v(y_x(\tau)) - v(x + \tau k_y) &= v(y_x(\tau)) - v(y_x(\tau)) \\ &- \underbrace{v'(\xi)}_{\leq C_v} \underbrace{(x + \tau k_y - y_x(\tau))}_{\mathcal{O}(\tau^3)}. \end{aligned} \tag{42}$$

Please note that the approximation of the state in the Taylor expansion is of order $\mathcal{O}(\tau^3)$, which is due to the Implicit Midpoint Rule with a local error of $\mathcal{O}(\tau^3)$. Note that in the estimation (41) we use the linear approximation from equation (40) and thus have $\mathcal{O}(\tau^2)$. In other words, comparing (41) with equation (42), in the former one uses a k_y from a scheme with time step τ, and not $\tau/2$.

Altogether, we have an approximation of the value function $v(x)$ by a higher order semi-Lagrangian scheme with a local time discretization error of $\mathcal{O}(\tau^3)$. This means that we can apply Theorem 1 with $p = 2$ to construct higher order composition schemes. Note that here we neglect the impact of the spatial reconstruction operator I for the approximation of the costs-to-go $v(y_x(\tau/2))$. □

To simplify the presentation of the algorithm for a semi-Lagrangian scheme based on the DISRK methods in the following, we denote the approach for the calculation of $Y_x(\tau)$ with the linear approximation (40) as $\overline{\Phi}_\tau^M$ and the approximated state $Y_x(\tau)$ as $\overline{Y}_x^\tau$, i.e.

$$\begin{cases} \overline{Y}_x^\tau \overset{\text{def}}{=} Y_x + \tau \begin{pmatrix} f(x + (\tau/2)k_y, \alpha(\tau/2)) \\ g(x + (\tau/2)k_y, \alpha(\tau/2)) \end{pmatrix} = \overline{\Phi}_\tau^M(Y_x, \alpha, \tau), \\ k_y = f(x + \tau/2 k_y, \alpha(\tau/2)) \end{cases} \tag{43}$$

and analog to (37) we will use the notation

$$\overline{Y}_x^\tau = \begin{pmatrix} \overline{Y}_{x,1}^\tau \\ \overline{Y}_{x,2}^\tau \end{pmatrix}. \tag{44}$$

4.3. **Optimization of the control.** In this section we are going to derive a semi-Lagrangian method with higher order in time. Our main motivation is to preserve the Dynamic Programming property, which can be achieved by considering schemes in the composition form as described in the representation (26). Then, by the Dynamic Programming property we are allowed to separate the optimization of the control for each time interval and thus decrease the complexity of the minimization problem.

Based on Theorem 1, this can be accomplished by using special compositions with time steps $\gamma_i\tau$, as described for the Runge-Kutta methods in Section 4.1. To be more precise, let us consider a DISRK method of the form

$$\Psi_\tau = \Phi_{\gamma_s\tau} \circ \ldots \circ \Phi_{\gamma_2\tau} \circ \Phi_{\gamma_1\tau}. \tag{45}$$

With this, we want to derive the following representation of a semi-Lagrangian scheme

$$\mathcal{S}^\tau = \mathcal{S}^{\gamma_s\tau} \circ \ldots \circ \mathcal{S}^{\gamma_2\tau} \circ \mathcal{S}^{\gamma_1\tau}. \tag{46}$$

For positive γ_i we can use directly equation (7) to provide an algorithm for $\mathcal{S}^{\gamma_i\tau}$. However, we have to consider negative γ_i more carefully, i.e. it must be clarified how one attains a semi-Lagrangian scheme $\mathcal{S}^{\widetilde{\tau}}$ with $\widetilde{\tau} < 0$.

As already outlined at the beginning of Section 4.1, there is a connection between semi-Lagrangian schemes for the approximation of the value function and the subproblems of the form (23) for the time intervals $[t_i, T]$. A more detailed consideration shows, that $\mathcal{S}^{\tau_i}(x)$, whereby $\tau_i = t_{i+1} - t_i$ and the notation is as in Section 4.1, solves the subproblem (23)

$$\min_{\alpha\in\mathcal{A}} J_x|_{[t_i,t_{i+1}]}(\alpha),$$

i.e. we have

$$v(x)|_{[t_i,t_{i+1}]} \approx I[\mathcal{S}^{\tau_i}(V^{i+1})](x),$$

with $v(x)|_{[t_i,t_{i+1}]}$ from (28). Please note, that in expression (26) the step $\mathcal{S}^{\tau_i}$ solves

$$\min_{\alpha\in\mathcal{A}|_{[t_{i+1},T]}} J_x|_{[t_i,t_{i+1}]}(\alpha).$$

In this case, the restriction of $\mathcal{A}$ to the set $\mathcal{A}|_{[t_{i+1},T]}$ achieves that we consider optimal control over the time interval $[t_i, T]$ and not only on $[t_i, t_{i+1}]$ as for $\mathcal{S}^{\tau_i}$.

From the association of $\mathcal{S}^{\tau_i}$ with the problem (23) one could argue, that $\mathcal{S}^{\widetilde{\tau}}$ with $\widetilde{\tau} < 0$ solves (23) for some time points $\eta_i > \eta_j$ and $\widetilde{\tau} = \eta_j - \eta_i$, where we now use the notation η_i, η_j to increase the readability. Indeed, let us define a restriction of the cost functional on $[\eta_i, \eta_j]$ analog to (22) as

$$J_x|_{[\eta_i,\eta_j]}(\alpha) \stackrel{\text{def}}{=} \int_{\eta_i}^{\eta_j} g(y(s), \alpha(s))\, ds. \tag{47}$$

and consider

$$\min_{\alpha\in\mathcal{A}} J_x|_{[\eta_i,\eta_j]}(\alpha) \tag{48}$$

and the associated set of minimizers $\mathcal{A}|_{[\eta_i,\eta_j]}$. By definition of $\mathcal{A}|_{[\eta_i,\eta_j]}$ we have for an element $\alpha^\star \in \mathcal{A}|_{[\eta_i,\eta_j]}$

$$\begin{aligned}
&\forall \alpha \in \mathcal{A} : J_x|_{[\eta_i,\eta_j]}(\alpha^\star) \le J_x|_{[\eta_i,\eta_j]}(\alpha) \Leftrightarrow \\
&\forall \alpha \in \mathcal{A} : \int_{\eta_i}^{\eta_j} g(y(s), \alpha^\star(s))\, ds \le \int_{\eta_i}^{\eta_j} g(y(s), \alpha(s))\, ds \Leftrightarrow \\
&\forall \alpha \in \mathcal{A} : -\int_{\eta_j}^{\eta_i} g(y(s), \alpha^\star(s))\, ds \le -\int_{\eta_j}^{\eta_i} g(y(s), \alpha(s))\, dr \qquad (49)
\end{aligned}$$

where we exchange the integration boundaries for the last term in (49). Now we obtain

$$\forall \alpha \in \mathcal{A} : \int_{\eta_j}^{\eta_i} g(y(s), \alpha^\star(s))\, ds \ge \int_{\eta_j}^{\eta_i} g(y(s), \alpha(s))\, ds.$$

This means, we can characterize $\alpha^\star$ as a maximizing element for the cost functional

$$J_x|_{[\eta_j,\eta_i]}(\alpha) = \int_{\eta_j}^{\eta_i} g(y(s),\alpha(s))\,ds$$

over the set of controls $\mathcal{A}$, where we have $\eta_j < \eta_i$ for $J_x|_{[\eta_j,\eta_i]}$. Therefore we can identify the set of minimizers $\mathcal{A}|_{[\eta_i,\eta_j]}$ with the solution of the problem

$$\max_{\alpha\in\mathcal{A}} J_x|_{[\eta_j,\eta_i]}(\alpha).$$

From this derivation it is clear, that

$$\mathcal{A}|_{[\eta_i,\eta_j]} \cap \mathcal{A}|_{[\eta_j,\eta_i]} = \emptyset \tag{50}$$

for control sets $\mathcal{A}$ with

$$\exists\,\alpha,\beta\in\mathcal{A}\,:\; J_x|_{[\eta_i,\eta_j]}(\alpha) \neq J_x|_{[\eta_i,\eta_j]}(\beta) \tag{51}$$

i.e. for controls which are distinguishable with respect to $J_x|_{[\eta_i,\eta_j]}$. Indeed, let us assume

$$\mathcal{A}|_{[\eta_i,\eta_j]} \cap \mathcal{A}|_{[\eta_j,\eta_i]} \neq \emptyset$$

and consider $\widetilde{\alpha} \in \mathcal{A}|_{[\eta_i,\eta_j]} \cap \mathcal{A}|_{[\eta_j,\eta_i]}$. Clearly it holds $\widetilde{\alpha} \in \mathcal{A}|_{[\eta_i,\eta_j]}$ and $\widetilde{\alpha} \in \mathcal{A}|_{[\eta_j,\eta_i]}$ and we obtain

$$\max_{\alpha\in\mathcal{A}} J_x|_{[\eta_j,\eta_i]}(\alpha) = \min_{\alpha\in\mathcal{A}} J_x|_{[\eta_j,\eta_i]}(\alpha).$$

This is a contradiction to the assumption (51).

In particular, from (50) and (24), we obtain

$$\mathcal{A}|_{[\eta_i,\eta_j]} \cap \mathcal{A}|_{[0,T]} = \emptyset, \tag{52}$$

which shows, that (48) is not a subproblem of (20). Thus, by considering $\mathcal{S}^{\widetilde{\tau}}$ to be an approximation of the solution to (48), we cannot derive a representation of the form (26). To be precise, by adding a step $\mathcal{S}^{\widetilde{\tau}}$ with $\widetilde{\tau} < 0$ to a composition

$$\mathcal{S}^{\tau_i} \circ \mathcal{S}^{\tau_{i+1}} \circ \ldots \circ \mathcal{S}^{\tau_N}$$

with $\tau_k > 0$ for $k = i,\ldots,N$ we will obtain

$$\mathcal{S}^{\widetilde{\tau}} \circ \mathcal{S}^{\tau_i} \circ \mathcal{S}^{\tau_{i+1}} \circ \ldots \circ \mathcal{S}^{\tau_N} \tag{53}$$

which solves the problem

$$\max_{\alpha\in\mathcal{A}|_{[t_i,T]}} J_x|_{[\eta_j,\eta_i]}\,. \tag{54}$$

The set of solutions to (54), denoted by $\mathcal{A}|_{\widetilde{\tau}}$, has for controls distinguishable with respect to $J_x|_{[\eta_i,\eta_j]}$ an empty intersection with the set $\mathcal{A}|_{[\eta_j,T]}$, which is approximated by

$$\mathcal{S}^{-\widetilde{\tau}} \circ \mathcal{S}^{\tau_i} \circ \mathcal{S}^{\tau_{i+1}} \circ \ldots \circ \mathcal{S}^{\tau_N}.$$

Due to the property (25) we get

$$\mathcal{A}|_{\widetilde{\tau}} \cap \mathcal{A}|_{[0,T]} = \emptyset.$$

Altogether, by considering compositions of the form (53) we will not obtain the solution to the problem (20).

From the above considerations it is clear, that $\mathcal{S}^{\widetilde{\tau}}$ must be derived as an approximation to some subproblem of (20). Furthermore, if we preserve the optimal substructure and overlapping of subproblems, we can achieve the composition form (26) by the rationale as in the beginning of Section 4.1, which is based on the Dynamic Programming property.

Let us examine the derivations in (49). By exchanging the integration boundaries we can replace the min operator in (48) by the max. This can be performed vice versa. Thus we can consider $\mathcal{S}^{\widetilde{\tau}}$ as solving the problem

$$\max_{\alpha\in\mathcal{A}} J_x|_{[\eta_i,\eta_j]}(\alpha) \tag{55}$$

for some time points $\eta_i > \eta_j$ and obtain by (49) an equivalent characterization of $\mathcal{A}|_{[\eta_i,\eta_j]}$ as the solution of

$$\min_{\alpha\in\mathcal{A}} J_x|_{[\eta_j,\eta_i]}(\alpha). \tag{56}$$

It is clear, that (56) is a subproblem to (20) (see the definition in (23)), which preserves the Dynamic Programming property. Altogether, by associating $\mathcal{S}^{\widetilde{\tau}}$ with the problem (55) we can achieve the composition representation (26). Please note that due to the form

$$\mathcal{S}^{\widetilde{\tau}} \circ \mathcal{S}^{\tau_i} \circ \mathcal{S}^{\tau_{i+1}} \circ \ldots \circ \mathcal{S}^{\tau_N}$$

we must consider $\mathcal{S}^{\widetilde{\tau}}$ given the set $\mathcal{A}|_{[t_i,T]}$. This means we have to solve the problem (55) and not the problem (56), wherefore we would need the set of minimizers $\mathcal{A}|_{[\eta_i,T]}$. We use the following theorem from [BCD97], which provides a characterization of the value function for the reverse time direction, to derive an algorithm for (55). Note here, that we can assume autonomous dynamics.

Theorem 3 (Backward Dynamic Programming Principle)**.** *Under the assumptions on f and g from Section 1, for all $\tau > 0$ small enough and $\widetilde{x}$ so that there exists a $z \in \mathbb{R}^d$ and an optimal $\alpha^\star \in \mathcal{A}$ with $\widetilde{x} = y_z(\tau, \alpha^\star)$, it holds*

$$v(\widetilde{x}) = \max_{\alpha\in\mathcal{A}^-} \left(v(y_{\widetilde{x}}(-\tau,\alpha))\,\mathrm{e}^{\lambda\tau} - \int_0^\tau g(y_{\widetilde{x}}(-s,\alpha),\alpha(-s))\,\mathrm{e}^{\lambda s}\,ds \right) \tag{57}$$

where $\mathcal{A}^- = \{\alpha : [-T, t_0] \to A,\ \textit{measurable}\}$.

To clarify the connection between (55) and (57) we remark, that with the notation from (57) we have $\eta_j = 0$ and thus $\widetilde{\tau} = -\eta_i = -\tau$. By applying the substitution $r = \eta_i - s$ to (55) we obtain

$$\begin{aligned} J_x|_{[\eta_i,\eta_j]} &= \int_{\eta_i}^{\eta_j} g(y_x(r),\alpha(r))\,dr = \int_{\eta_i}^{0} g(y_x(r),\alpha(r))\,dr \\ &= -\int_0^{\eta_i} g(y_x(\eta_i - s),\alpha(\eta_i - s))\,ds. \end{aligned}$$

Furthermore we have $y_x(\eta_i - s) = y_{\widetilde{x}}(-s)$ due to x being denoted by z in Theorem 1. Finally we can identify the evaluation of the value function $v(y_x(-\tau,\alpha))$ with the restriction of the control set $\mathcal{A}$ to $\mathcal{A}|_{[t_i,T]}$.

Based on (57) we can provide the following scheme for the computation of $\mathcal{S}^{\widetilde{\tau}}$, i.e. for negative time steps, analog to (7)

$$v_j^{\gamma_i\tau} = \max_{A^n} \left\{ G^\Delta(x_j, \gamma_i\tau, A^n) + I[V^{\gamma_{i-1}\tau}](x_j, A^n) \right\}. \tag{58}$$

In equation (58) $V^{\gamma_{i-1}\tau}$ denotes the approximation of the value function from $s^{\gamma_{i-1}\tau}$.

For the computation of $G^\Delta(x_j, \gamma_i\tau, A^n)$ and $y_{x_j}^{\gamma_i\tau}$ we choose the scheme $\overline{\Phi}_\tau^M$ derived in Section 4.2. Thus, by Theorem 1, we achieve higher convergence order with respect to the time step size, if we neglect the effects of the control discretization and the error of the spatial reconstruction $I[V^{\gamma_{i-1}\tau}]$. In the next section we provide the complete algorithm for semi-Lagrangian schemes based on Runge-Kutta composition methods.

4.4. **Algorithm for high order semi-Lagrangian schemes with DISRK methods.** Algorithm 1 illustrates the idea of a semi-Lagrangian scheme with a DISRK method. $X \subset \mathbb{R}^d$ denotes the set of grid points from the spatial discretization, i.e. for $X = \{x_1, \dots, x_M\}$ we have

$$Y_X = \left(Y_{x_1}, Y_{x_2}, \dots, Y_{x_M}\right) \tag{59}$$

Utilizing the notation in (59) and definition of $\overline{Y}^\tau_x$ in equation (43), we introduce

$$\overline{Y}^\tau_X = \left(\overline{Y}^\tau_{x_1}, \overline{Y}^\tau_{x_2}, \dots, \overline{Y}^\tau_{x_M}\right)$$

which represents the time discretization within the semi-Lagrangian scheme. Analog to the notation in equations (37) and (44) we refer to the numerical integration of the state system (2) with $\overline{Y}^\tau_{X,1}$ and to the quadrature of the running costs with $\overline{Y}^\tau_{X,2}$, i.e.

$$\overline{Y}^\tau_{X,1} = \left(\overline{Y}^\tau_{x_1,1}, \overline{Y}^\tau_{x_2,1}, \dots, \overline{Y}^\tau_{x_M,1}\right)$$

and

$$\overline{Y}^\tau_{X,2} = \left(\overline{Y}^\tau_{x_1,2}, \overline{Y}^\tau_{x_2,2}, \dots, \overline{Y}^\tau_{x_M,2}\right).$$

With $<$ and $>$ in the lines 12 and 15 respectively we denote element wise comparison of the value function at the spatial grid X. This means, that the minimization/maximization is independent for each grid point.

The important part of the Algorithm 1 is the inner `for`-loop in the line 4, where the composition is calculated for given $\gamma_1, \dots, \gamma_s$. One can easily verify that the complexity of the control optimization is $\mathcal{O}(s \cdot d_c)$, where d_c denotes the number of discretization points of the control space and s is the number of stages of the DISRK scheme. Note, that the value function V is updated for each step $\gamma_i \tau$. This means, after each iteration of the `for`-loop in the line 4 we obtain a temporary value function $v^{\gamma_i \tau}$ as described in Section 4.3. The resulting value function after time τ is constructed via a sequence of $v^{\gamma_i \tau}$.

Algorithm 1 in this form is for finite time problems. For infinite time horizon control problems, e. g.with the cost functional (4), we have to adapt the stopping criteria of the `while`-loop of Algorithm 1 slightly. This can be achieved by comparing the computed value functions for two steps of the `while`-loop in the line 3. We then stop the iterations if the change in the value functions is less then some required tolerance TOL:

$$\frac{1}{|X_{ref}|} \left(\sum_{x \in X_{ref}} \left(I[V^k](x) - I[V^{k-1}](x)\right)^2 \right)^{\frac{1}{2}} \leq \text{TOL} \tag{60}$$

where $X_{ref} \subset \mathbb{R}^d$ denotes a set of spatial test points. Another approach is to use the maximum in the change of the value function

$$\max_{x \in X_{ref}} \left\{ I[V^k](x) - I[V^{k-1}](x) \right\} \leq \text{TOL}. \tag{61}$$

Algorithm 1: Semi-Lagrangian scheme for the Hamilton-Jacobi-Bellman equation with DISRK time discretization and optimization by comparison

Data: $\alpha_1, \dots, \alpha_{d_c}$ - sampling of the control space
$\gamma_1, \dots, \gamma_s$ - composition coefficients, T - end time, τ - time step
v_T - value function at T
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**
    for $i = 1, \dots, d_c$ **do**
      $\alpha = \alpha_i$
      $\overline{Y}^{\tau}_{X} = \overline{\Phi}^{M}_{\gamma_j \tau}(Y_X, \alpha(t), \tau)$ ▷ Approximation of the state dynamic and running costs as in (43)
      $V = \overline{Y}^{\tau}_{X,2} + I\left[V_{old}\right](\overline{Y}^{\tau}_{X,1})$ ▷ Approximation of the value function as in (7)
      if $\gamma_i \geq 0$ **then**
        if $V_{old} > V$ **then**
          $V_{old} = V$
      else
        if $V_{old} < V$ **then**
          $V_{old} = V$
  $t = t - \tau$
return V_{old}

5. Numerical tests

In this section we present numerical results for semi-Lagrangian schemes based on DISRK methods. We are mainly interested in the convergence rates for the error of the numerical approximation compared to a reference solution and in the computational complexity.

For the error estimation we consider analog to [GK17] the difference of the exact solution v_{ref} to the approximation v_{num} on a spatial test grid X_{ref} and approximate the $L^2(\Omega)$ respectively the $L^\infty(\Omega)$ norm of the error with

$$e_2^v \overset{\text{def}}{=} \frac{\|v_{ref} - v_{num}\|_2}{\sqrt{|X_{ref}|}} \approx \|v_{ref} - v_{num}\|_{L^2(\Omega)},$$

$$e_\infty^v \overset{\text{def}}{=} \|v_{ref} - v_{num}\|_\infty \approx \|v_{ref} - v_{num}\|_{L^\infty(\Omega)}. \tag{62}$$

In addition we provide convergence rates for the error estimates as

$$\rho_i(e) = \log_b\left(\frac{e_{i-1}}{e_i}\right), \tag{63}$$

where b is the refinement factor for the time discretization. In (63) e is one of the error estimates from (62) and i represents the i-th considered time discretization.

As a measure of the computational complexity we give the number of evaluations n^f of the state dynamics $f(y(s), \alpha(s))$ from (2) as a function of control space discretization. Furthermore we consider the rate of increase for n^f

$$\eta_j(n^f) = \log_2 \left(\frac{n_j^f}{n_{j-1}^f} \right),$$

with n_j^f as the number of evaluation of $f(y(s), \alpha(s))$ for the discretization of the control space with 2^j points, i.e. control level j.

5.1. **One dimensional system with non-linear dynamics.** As a first example we consider a numerical experiment, which is well suited to test the performance of a scheme for smooth solutions [FF94]. The state dynamics are given by

$$\begin{cases} \dot{y}(s) = f(y(s), s, \alpha(s)) = -y(s)\, \mathrm{e}^{-y(s)}\, \alpha(s)^2 \\ y(t_0) = 0, \end{cases} \tag{64}$$

and the expression for the running costs is

$$g(y_x(s, \alpha), \alpha(s)) = -y(s)\alpha(s)^2 - \mathrm{e}^{y(s)} \sin \frac{\pi \alpha(s)}{2}.$$

The exact value function for the infinite time horizon problem is [FF94]:

$$v_{ref}(x) = \mathrm{e}^{-y_x(s)}\,. \tag{65}$$

We approximate the value function in $\Omega = [0, 1]$ and the control is from $A = [0, 1]$.

The numerical tests with Algorithm 1 in the value iteration form are performed using Matlab. For the spatial interpolation we use cubic splines on an equidistant grid X_Ω with grid size of $\Delta\Omega = 10^{-3}$ for the DISRK_5 scheme and $\Delta\Omega = 10^{-4}$ for the DISRK_9 and DISRK_{17} schemes. The discretization of the control set is $\Delta A = 10^{-3}$ for the DISRK_5 scheme, $\Delta A = 10^{-4}$ for the DISRK_9 and $\Delta A = 10^{-5}$ for the DISRK_{17}.

To evaluate the error estimate we consider $X_{ref} = X_\Omega$. We decrease time discretization by $b = 10^{0.1}$, i.e. $\tau_i = 10^{-0.1} \cdot \tau_{i-1}$. We set the tolerance TOL for the `while`-loop in (61) to 10^{-14} and end time $T = 100.0$ as an additional criterion for the termination. This means, we stop the computation once the current time reaches $T = 100.0$ if the tolerance TOL in (61) has not been achieved.

Figure 3 presents the error estimates e_v^2 as a function of the time resolution. We can observe, that the convergence rate of the DISRK_5 method correlates very well with the theoretical rate of 4. The DISRK_9 method does not achieve the expected convergence rate of 6 exactly, shows however results which are very close to the theoretical value for the first 7 measurements. Similarly for DISRK_{17} we observe a convergence rate of about 8 for the first five steps. The stagnation of the error for the remaining part of the plot, in particular visible for DISRK_{17}, presumably can be explained by the effects of the other discretization aspects, i.e. the impact of the discretization of the control and the state space, respectively. In Table 1 we present more detailed values for the error estimates and convergence rates for each time discretization.

As a measure of computational effort we provide the number of evaluations of the state dynamics n^f as a function of the discretization of the control space in Table 2, where we consider 2^j points in the compare approach for the optimization. For these tests we choose a discretization of the state space with $\Delta\Omega = 10^{-2}$. Additional quantity is the increase rate $\eta_j(n^f)$. We observe a near linear dependency of n^f on the cardinality of the discrete control set A, i.e. $\eta_j(n^f) \approx 1.0$.

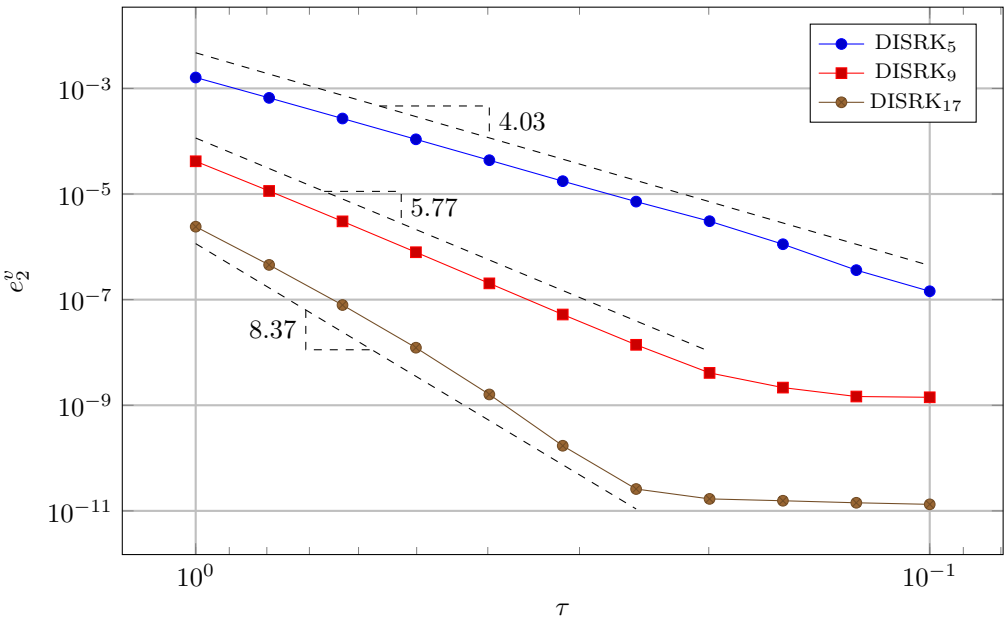


FIGURE 3. Estimation e_2^v of the $L^2(\Omega)$ error. We consider all error estimates for the calculation of the DISRK$_5$ regression line, the first 8 estimates for the regression line of DISRK$_9$ and the first 7 for DISRK$_{17}$.

	DISRK$_5$		DISRK$_9$		DISRK$_{17}$	
τ	e_2^v	$\rho_i(e)$	e_2^v	$\rho_i(e)$	e_2^v	$\rho_i(e)$
1.00_0	1.61_{-3}	–	4.17_{-5}	–	2.41_{-6}	–
7.94_{-1}	6.61_{-4}	3.86	1.14_{-5}	5.62	4.55_{-7}	7.24
6.31_{-1}	2.69_{-4}	3.91	3.04_{-6}	5.75	7.91_{-8}	7.6
5.01_{-1}	1.09_{-4}	3.94	7.90_{-7}	5.85	1.23_{-8}	8.08
3.98_{-1}	4.37_{-5}	3.96	2.03_{-7}	5.89	1.60_{-9}	8.85
3.16_{-1}	1.75_{-5}	3.97	5.23_{-8}	5.9	1.71_{-10}	9.71
2.51_{-1}	7.18_{-6}	3.87	1.40_{-8}	5.74	2.59_{-11}	8.19
2.00_{-1}	3.06_{-6}	3.71	4.10_{-9}	5.32	1.69_{-11}	1.87
1.58_{-1}	1.12_{-6}	4.36	2.16_{-9}	2.78	1.55_{-11}	0.35
1.26_{-1}	3.62_{-7}	4.91	1.46_{-9}	1.69	1.42_{-11}	0.38
1.00_{-1}	1.44_{-7}	4	1.42_{-9}	0.15	1.33_{-11}	0.3

TABLE 1. Estimations e_2^v of the $L^2(\Omega)$ errors and the corresponding convergence rates $\rho_i(e)$ at time steps τ for different DISRK methods. The theoretical convergence rate for DISRK$_5$ is 4, for DISRK$_9$ it is 6, and for DISRK$_{17}$ it is 8. Subscripts denote exponents in base 10, i.e. $1.0_{-n} = 1.0 \cdot 10^{-n}$.

To illustrate the advantage of semi-Lagrangian schemes based on DISRK methods, we compare the number of function calls to the Runge-Kutta 4 method, with Butcher tableau equation (11). For the Runge-Kutta 4 method we can observe $(2^j)^4$ evaluations of the state trajectory, where j denotes the control level. The corresponding number of functions calls n^f is $(2^j)^4 \cdot 4$, i.e. the state dynamics is evaluated 4 times for each control $a \in A$ due to 4 stages of the method. Please note, that DISRK are implicit methods. This means, that the number of function calls n^f also depends on the effort for the solution of the implicit equation for the stage k of the Implicit

control level	DISRK$_5$		DISRK$_9$		DISRK$_{17}$		RK$_4$	
	n^f	$\eta_j(n^f)$	n^f	$\eta_j(n^f)$	n^f	$\eta_j(n^f)$	n^f	$\eta_j(n^f)$
1	25	–	41	–	75	–	64	–
2	55	1.14	95	1.21	177	1.24	1,024	4
3	116	1.08	209	1.14	384	1.12	16,384	4
4	242	1.06	429	1.04	796	1.05	262,144	4
5	487	1.01	871	1.02	1,614	1.02	4,194,304	4
6	978	1.01	1,756	1.01	3,248	1.01	67,108,864	4
7	1,961	1	3,523	1	6,521	1.01	–	–
8	3,931	1	7,060	1	13,063	1	–	–
9	7,872	1	14,135	1	26,142	1	–	–
10	15,741	1	28,277	1	52,314	1	–	–

TABLE 2. Number of function evaluations n^f for the dynamics of the controlled system (2) for one time step as well as corresponding increase rates $\eta_j(n^f)$ for different control levels, i.e. 2^j comparison points.

Midpoint Rule, see Section 4. We use the `fsolve` Matlab function with the default trust-region algorithm and parameters TolX $= 10^{-15}$ and TolFun $= 10^{-15}$ to find k.

5.2. **Harmonic oscillator.** As a second example we consider the control of a harmonic oscillator as in [GK17]. The state dynamics are defined by

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

with $y \in \mathbb{R}^2$. The cost function is of the form

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

with $\alpha = 0.1$.

For the computational domain we choose $\Omega = [0,1]^2$ and for the control domain $A = [-3.5, 3.5]$. The discretization of the control space is obtained by an equidistant sampling, where in the following we denote the number of sampling points for the control domain with $|A_\Delta|$, compare Section 2. We choose A_Δ with 10^3 sampling points for the computation with the Implicit Midpoint Rule. Due to higher precision of the semi-Lagrangian schemes based on DISRK we consider different numbers of sampling points for these methods. To estimate the convergence rate with respect to the time step τ we need to neglect the impact of control and state space discretizations. In the following, the numerical experiments are computed with sampling sizes from $5.0 \cdot 10^4$ to $5.0 \cdot 10^6$. For the discretization of the value function we consider sparse grids with linear and 3rd order B-Splines, both with the fold out ansatz. We use the SG++ library [Pfl10, VP16] in the numerical implementation. The equation for the stage k of the Implicit Mipoint Rule is solved with the standard Newton algorithm, whereby we require the residuum of the implicit equation or the norm of the increment for k from the Newton step to be smaller than 10^{-12} as a terminating condition.

For this test case we can obtain a reference solution from the continuous time Riccati differential equation

$$\begin{cases} A^T P(t) + P(t)A - P(t)BR^{-1}B^T P(t) + Q = -\dot{P}(t) \\ P(T) = 0 \end{cases}, \tag{66}$$

see e.g. [FF14]. We choose $T = 0.1$ and solve (66) with a step size of 10^{-7}. We obtain the following reference solution for the value function

$$v_{ref}(x) = \frac{1}{2} x^T P x \tag{67}$$

with

$$P = \begin{pmatrix} 9.966236712679333 \cdot 10^{-2} & 4.982996318566705 \cdot 10^{-3} \\ 4.982996318566705 \cdot 10^{-3} & 3.326412035418859 \cdot 10^{-4} \end{pmatrix},$$

where x denotes a vector from the state space. The computation of the error estimates from (62) is performed on a domain $\Omega_{ref} = [0.25, 0.75]^2$, where we consider an equidistant grid X_{ref} of step size 0.5/200.0 in both spatial directions.

First we consider the semi-Lagrangian scheme based on the Implicit Midpoint Rule for the time discretization. The results are presented in the Figure 4. The goal of these computations is to evaluate the performance of a semi-Lagrangian scheme without composition as well as to estimate meaningful parameter ranges for the discretization of the control and state spaces. However, as mentioned at the beginning of the section, the parameters have to be adapted for DISRK methods of higher order in time.

Furthermore we will consider sparse grids base on fold out B-splines for the discretization of the state space in the following experiments, as the achieved precision compared to the computation time is better then for the fold out linear basis. Although the computation with B-splines is

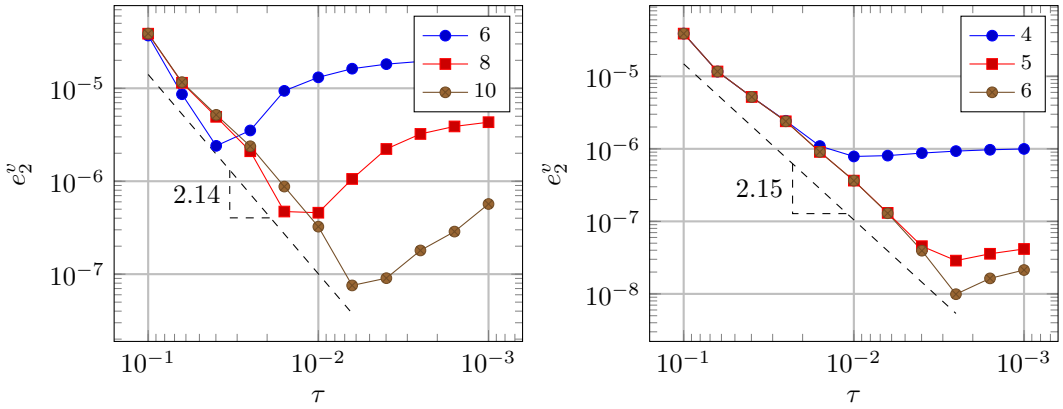


FIGURE 4. Estimation e_2^v of the $L^2(\Omega)$ error for the Implicit Midpoint Rule with $T = 0.1$. On the left with a linear fold out SG basis, on the right with fold out order 3 B-splines. The numbers in the legend denote different SG levels.

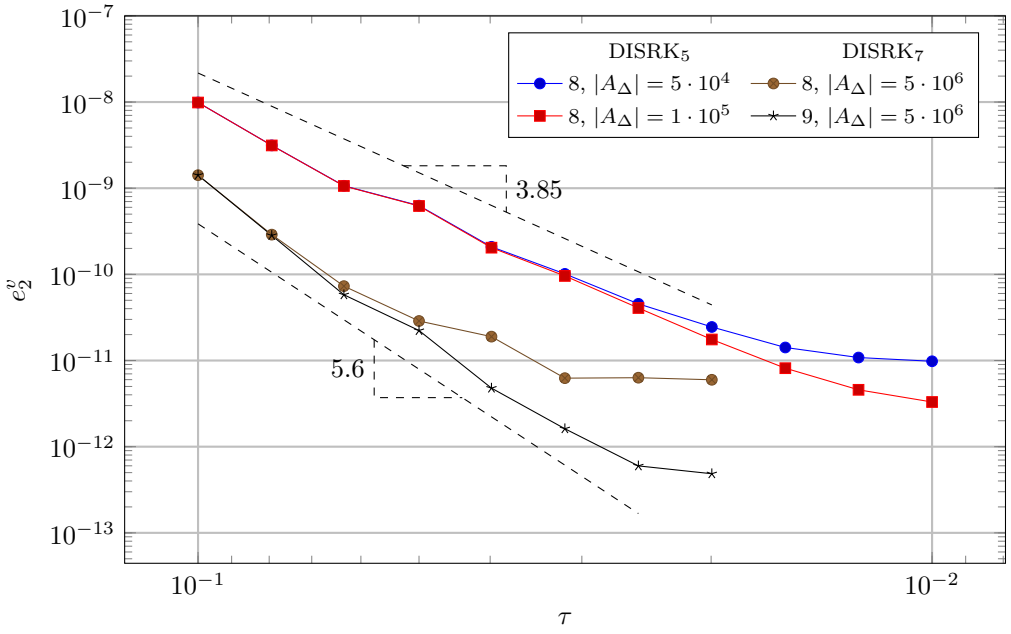


FIGURE 5. Estimation e_2^v of the $L^2(\Omega)$ error for the DISRK$_5$ and DISRK$_7$ methods with $T = 0.1$. The numbers in the legend denote different SG levels for the fold out order 3 B-spline basis. $|A_\Delta|$ is the number of sampling points for the control domain A. The regression is calculated with the first 8 points for DISRK$_5$ with level 8 SG discretization, $|A_\Delta| = 10^5$, and with the first 7 points for DISRK$_7$ with level 9 SG discretization.

more complex compared to a linear basis for the same spatial grid, see Section 3, we can choose a smaller level of the sparse grids in case of the B-splines.

	DISRK$_5$, B-splines level 8, $\lvert A_\Delta\rvert = 1.0_5$				DISRK$_7$, B-splines level 9, $\lvert A_\Delta\rvert = 5.0_6$			
τ	e_2^v	$\rho_i(e)$	e_∞^v	$\rho_i(e)$	e_2^v	$\rho_i(e)$	e_∞^v	$\rho_i(e)$
1.00_{-1}	9.87_{-9}	–	1.91_{-8}	–	1.42_{-9}	–	2.69_{-9}	–
7.94_{-2}	3.14_{-9}	4.98	6.05_{-9}	4.98	2.83_{-10}	6.99	5.64_{-10}	6.77
6.33_{-2}	1.06_{-9}	4.7	2.04_{-9}	4.71	5.80_{-11}	6.89	1.22_{-10}	6.65
5.00_{-2}	6.22_{-10}	2.32	1.20_{-9}	2.32	2.23_{-11}	4.15	5.55_{-11}	3.42
3.98_{-2}	2.04_{-10}	4.84	3.93_{-10}	4.84	4.78_{-12}	6.68	2.33_{-11}	3.78
3.16_{-2}	9.59_{-11}	3.28	1.86_{-10}	3.26	1.62_{-12}	4.71	9.20_{-12}	4.03
2.51_{-2}	4.07_{-11}	3.72	8.90_{-11}	3.2	5.99_{-13}	4.31	4.27_{-12}	3.34
2.00_{-2}	1.76_{-11}	3.65	4.63_{-11}	2.84	4.85_{-13}	0.91	6.75_{-12}	−1.99
1.58_{-2}	8.15_{-12}	3.33	2.88_{-11}	2.06	–	–	–	–
1.26_{-2}	4.57_{-12}	2.51	2.47_{-11}	0.66	–	–	–	–
1.00_{-2}	3.30_{-12}	1.42	2.25_{-11}	0.41	–	–	–	–

TABLE 3. Estimations e_2^v of the $L^2(\Omega)$ and e_∞^v of the $L^\infty(\Omega)$ errors and the corresponding convergence rates $\rho_i(e)$ at time steps τ for different DISRK methods. The theoretical convergence rate for the DISRK$_5$ is 4, for DISRK$_7$ - 6. Subscripts denote exponents in base 10, i.e. $1.0_{-n} = 1.0 \cdot 10^{-n}$.

Next we consider semi-Lagrangian schemes based on the DISRK$_5$ and DISRK$_7$ methods. The results are presented in the Figure 5, while detailed numbers can be found in Table 3. We consider results for the DISRK$_5$ scheme for a spatial discretization with sparse grid based on level 8 fold out B-splines and $|A_\Delta| = 5.0 \cdot 10^4$ respectively $|A_\Delta| = 1.0 \cdot 10^5$. Different control domain discretizations are provided to illustrate the effect of the discretization of the control space. The computations with the DISRK$_7$ scheme are performed with the discretization of the control space with $|A_\Delta| = 5.0 \cdot 10^6$ and fold out B-splines grids of level 8 and 9.

For both schemes — DISRK$_5$ and DISRK$_7$ — we can observe convergence rates higher then for a semi-Lagrangian scheme based on the Implicit Mipoint Rule, where we have $p \approx 2.15$ as in Figure 4. A more precise interpretation of results is difficult due to the overlap of impacts from different discretization parameters.

6. Conclusion and outlook

In this work we present semi-Lagrangian schemes with higher order time discretization for optimal control problems. The numerical method is in particular of interest with regard to the cost complexity of the optimization over the control space. The approach allows for a separation of the control optimization with respect to the individual stages of the Runge-Kutta scheme, by exploiting properties in correspondence with the Dynamic Programming Principle. Therefore the optimization costs increase linearly with the number of stages, but not exponentially as is the case for standard higher order Runge-Kutta time discretization approaches.

In the numerical tests we observe nearly linear complexity $\mathcal{O}(s \cdot d_c)$ of the constructed semi-Lagrangian schemes with respect to the costs d_c of one optimization and the number of stages of the Runge-Kutta method s. In comparison, in the general case an exponential complexity of $\mathcal{O}(d_c^s)$ holds for semi-Lagrangian schemes based on Runge-Kutta methods.

Besides the time discretization, semi-Lagrangian schemes are affected by the optimization of the control space as well as by the reconstructions methods used for the state space. The numerical tests in this work were performed with piecewise linear and B-spline based spatial discretization methods, where for two dimensions sparse grids were considered. For the optimization in the control space we used a simple discretization with an equidistant step size, on which we evaluated the function values and selected the optimum. It is clear from the numerical tests,

that such a simple optimization in the control has a significant impact on the overall runtime of the semi-Lagrangian scheme, in particular in view of the high accuracy which was needed to be on-par with the time and spatial discretization. In future work, a more efficient optimization procedure would need to be integrated into the numerical scheme.

References

[BCD97] Martino Bardi and Italo Capuzzo-Dolcetta. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equation*. Birkhäuser, 1997.

[BG04] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta Numerica*, 13:1–123, 2004.

[BGGK13] Olivier Bokanowski, Jochen Garcke, Michael Griebel, and Irene Klompmaker. An adaptive sparse grid semi-Lagrangian scheme for first order Hamilton-Jacobi Bellman equations. *Journal of Scientific Computing*, 55(3):575–605, 2013.

[CIR52] Richard Courant, Eugene Isaacson, and Mina Rees. On the solution of nonlinear hyperbolic differential equations by finite differences. *Comm. Pure Appl. Math.*, 5:243–255, 1952.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* The MIT Press, 2009.

[FF94] Maurizio Falcone and Roberto Ferretti. Discrete-time high-order schemes for viscosity solutions of hamilton-jacobi equations. *Numer. Math.*, 67(3):315–344, 1994.

[FF14] Maurizio Falcone and Roberto Ferretti. *Semi-Lagrangian Approximation Schemes for Linear and Hamilton-Jacobi Equations*. SIAM, 2014.

[FQ10] Kang Feng and Mengzhao Qin. *Symplectic Geometric Algorithms for Hamiltonian Systems.* Springer, 2010.

[Gar13] Jochen Garcke. Sparse grids in a nutshell. In J. Garcke and M. Griebel, editors, *Sparse grids and applications*, volume 88 of *Lecture Notes in Computational Science and Engineering*, pages 57–80. Springer, 2013.

[GK17] Jochen Garcke and Axel Kröner. Suboptimal Feedback Control of PDEs by Solving HJB Equations on Adaptive Sparse Grids. *Journal of Scientific Computing*, 70(1):1–28, 2017.

[HLW06] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric Numerical Integration. Structure-Preserving Algorithms for Ordinary Differential Equations.* Springer, 2006.

[Pfl10] Dirk Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. Verlag Dr. Hut, München, August 2010.

[VP16] Julian Valentin and Dirk Pflüger. Hierarchical gradient-based optimization with b-splines on sparse grids. In Jochen Garcke and Dirk Pflüger, editors, *Sparse Grids and Applications – Stuttgart 2014*, volume 109 of *Lecture Notes in Computational Science and Engineering*, pages 315–336. Springer, 2016.

Institut für Numerische Simulation, Universität Bonn, Wegelerstr. 6, 53115 Bonn and Fraunhofer SCAI, Schloss Birlinghoven, 53754 Sankt Augustin

E-mail address: garcke@ins.uni-bonn.de

Department für Mathematik und Informatik, Universität Basel, Spiegelgasse 1, 4051 Basel

E-mail address: ilja.kalmykov@unibas.ch