An adaptive sparse grid semi-lagrangian scheme for first order Hamilton-Jacobi Bellman equations

Olivier Bokanowski, Jochen Garcke, Michael Griebel, Irene Klompmaker

AN ADAPTIVE SPARSE GRID SEMI-LAGRANGIAN SCHEME FOR FIRST ORDER HAMILTON-JACOBI BELLMAN EQUATIONS

OLIVIER BOKANOWSKI, JOCHEN GARCKE,
MICHAEL GRIEBEL, AND IRENE KLOMPMAKER

ABSTRACT. We propose a semi-Lagrangian scheme using a spatially adaptive sparse grid to deal with non-linear time-dependent Hamilton-Jacobi Bellman equations. We focus in particular on front propagation models in higher dimensions which are related to control problems. We test the numerical efficiency of the method on several benchmark problems up to space dimension $d = 8$, and give evidence of convergence towards the exact viscosity solution. In addition, we study how the complexity and precision scale with the dimension of the problem.

1. INTRODUCTION

We are interested in solving the Hamilton-Jacobi Bellman (HJB) equation

$$v_t + \max_{\beta(t)\in\mathcal{B}} (f(x,\beta(t))\cdot\nabla v) = 0, \qquad t>0,\ \underline{x}\in\mathbb{R}^d, \tag{1a}$$

$$v(0,\underline{x}) = \varphi(\underline{x}), \qquad \underline{x}\in\mathbb{R}^d \tag{1b}$$

in a higher dimensional state space of dimension d. The *action space* $\mathcal{B}$, wherein the time-dependent control β takes its values, is a nonempty compact subset of $\mathbb{R}^m$ ($m \geq 1$), and the function $f : \mathbb{R}^d \times \mathcal{B} \to \mathbb{R}^d$, describing the *state dynamics*, is assumed to be Lipschitz continuous. This problem is closely related to the computation of the value function of optimal control problems [2]. Note here that the HJB equation (1a) is a particular case of the more general Hamilton-Jacobi (HJ) equation $v_t + H(x,\nabla v) = 0$.

In this paper, we shall focus on the approximation of a reachable set, either coded as $\Omega(t) := \{\underline{x}, v(t,\underline{x}) \leq 0\}$ or defined by its front $\partial\Omega(t)$. It is known from the work of Osher and Sethian [22] that front propagation problems can be solved by using level sets and HJ equations. Front propagation can be used for the determination of safety regions or for the treatment of avoidance problems [20, 21], for the computation of the function describing the minimal time to reach a set

$$\Omega(0) := \{\underline{x},\ \varphi(\underline{x}) \leq 0\}, \tag{2}$$

and for optimal trajectory and feedback control law reconstruction, cf. [2, Appendix]. Indeed, the solution of (1) is given by

$$v(t,\underline{x}) = \inf_{\beta\in L^\infty([0,t],\mathcal{B})} \varphi(y^\beta_{\underline{x}}(t)), \tag{3}$$

where $y^\beta_{\underline{x}} : [0,t] \to \mathbb{R}^d$ denotes the absolutely continuous solution of

$$\begin{cases} \dot{y}(t) = -f(y(t),\beta(t)) & \text{for } t\in\mathbb{R}_+ \text{ a.e.},\\ y(0) = \underline{x}. \end{cases} \tag{4}$$

Date: September 20, 2012.

Key words and phrases. sparse grids, Hamilton-Jacobi Bellman equation, front propagation, semi-Lagrangian scheme, adaptivity.

Hence $\Omega(t) \equiv \left\{ \underline{x} \in \mathbb{R}^d,\ \exists \beta \in L^\infty([0,t],\mathcal{B}),\ y^\beta_{\underline{x}}(t) \in \Omega(0) \right\}$ represents the set of points from which one can reach a given target $\Omega(0)$ in the time interval $[0,t]$ using some control β, where $\Omega(0)$ is defined by the given data φ such that (2) holds.

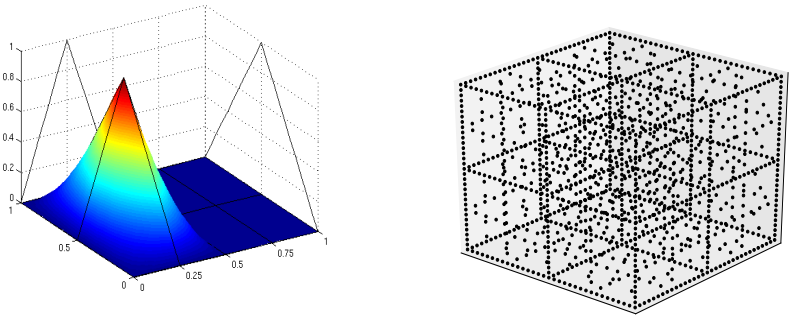
Various numerical methods have been proposed to determine approximations to the viscosity solution of $v_t + H(x, \nabla v) = 0$ (which includes the case of (1)). Crandall and Lions [6] studied first order monotone finite difference schemes, which converge to the solution. Note that monotone schemes are in general limited to at most first order accuracy [13]. Beyond that, higher order finite difference schemes, such as ENO schemes [23], have also been developed. These finite difference methods work quite efficiently for Cartesian meshes, but on unstructured meshes the schemes are more delicate [29]. Discontinuous Galerkin (DG or RKDG) methods, originally devised to solve conservation laws, can also be applied to HJ equations, with the flexibility for arbitrarily unstructured meshes [18]. Then, there are semi-Lagrangian (SL) schemes which are based on the discretization of the dynamic programming principle, see e.g. [11]. They can be easily implemented on arbitrary meshes. Here, it is mainly required to know how to interpolate from given values on a given mesh. A simple version of the SL scheme, which uses P_1 polynomial interpolation, is monotone and provides first order accuracy. Furthermore, higher order modifications exist, see e.g. [5, 7, 10]. Finally, adaptive schemes for solving the HJ(B) equation have been developed, see for instance [4, 16]. But altogether, the numerical treatment of HJ(B) equations remains a challenging problem, in particular in higher dimensions.

In this work we employ the sparse grid method, a special discretization technique which allows to cope with the curse of dimensionality to some extent. It is based on a hierarchical multilevel basis [8, 27] and a sparse tensor product construction. The underlying idea was first used for numerical integration [26]. Subsequently, the sparse grid method has been developed for the solution of partial differential equations [14, 28]. By now, it is also successfully used for, e.g., integral equations, stochastic differential equations, machine learning, or interpolation and approximation, see the overview article [3] and the references cited therein.

For the representation of a function f defined over a d-dimensional domain, the conventional sparse grid approach employs $\mathcal{O}(h_n^{-1} \cdot \log(h_n^{-1})^{d-1})$ grid points in the discretization process, where $h_n := 2^{-n}$ denotes the mesh width. It can be shown that the order of approximation to describe a function f, provided that certain mixed smoothness conditions hold, is $\mathcal{O}(h_n^2 \cdot \log(h_n^{-1})^{d-1})$. This is in contrast to conventional grid methods, which need $\mathcal{O}(h_n^{-d})$ for an accuracy of $\mathcal{O}(h_n^2)$, albeit for less stringent smoothness conditions. Thus, the curse of dimensionality of full grid methods arises for sparse grids to a much smaller extent. In case the smoothness conditions are not fulfilled, spatially adaptive sparse grids have been used with good success [3, 12, 25]. There, as in any adaptive grid refinement procedure, the employed hierarchical basis functions are chosen during the actual computation depending on the function to be represented.

In this paper we define a new semi-Lagrangian scheme on an adaptive sparse grid. We show that, for a particular kind of HJB equations related to the front propagation model, the number of grid points needed in higher dimensions to approximately represent the involved functions with a given threshold error can be small. Thus, we are able to circumvent the curse of dimensionality of standard grid approaches to some extent. There are two important ingredients that make things work: firstly, the spatial adaptivity of the sparse grid used in the scheme, and secondly, a particular type of boundary treatment using non-standard basis functions. We illustrate the feasibility of the method numerically for a set of front propagation examples for dimensions up to $d = 8$.

Note that the introduced sparse grid scheme is not monotone as the interpolation with sparse grids is not monotone [25]. Thus neither convergence towards the viscosity solutions of (1) nor stability can presently be guaranteed, even for the linear advection equation.



(a) Basis function $\phi_{(2,1),(1,1)}$ on grid $\Omega_{2,}$ (b) Three-dimensional sparse grid Ω_5^s

FIGURE 1. An example of a basis function $\phi_{\underline{l},\underline{j}}$ and a sparse grid Ω_n^s.

To this end, further work on the new scheme is in progress. Nevertheless, our numerical results give promising results.

This paper is organized as follows. In section 2 we describe the sparse grid structure we use to represent the data, the boundary treatment, and the adaptive refinement and coarsening procedures. Our new adaptive semi-Lagrangian sparse grid scheme is introduced and discussed in Section 3. Section 4 contains the results for various numerical examples. Finally we give some concluding remarks.

2. SPARSE GRIDS

For ease of presentation we will consider the domain $\Omega = [0,1]^d$ in this section. Let $\underline{l} = (l_1, \ldots, l_d) \in \mathbb{N}^d$ denote a multi-index. We define the anisotropic grid $\Omega_{\underline{l}}$ on Ω with mesh width $h_{\underline{l}} := (h_{l_1}, \ldots, h_{l_d}) := (2^{-l_1}, \ldots, 2^{-l_d})$. It has, in general, different but equidistant mesh widths h_{l_t} in each coordinate direction t, $t = 1, \ldots, d$. The grid $\Omega_{\underline{l}}$ thus consists of the points

$$x_{\underline{l},\underline{j}} := (x_{l_1,j_1}, \ldots, x_{l_d,j_d}), \tag{5}$$

with $x_{l_t,j_t} := j_t \cdot h_{l_t} = j_t \cdot 2^{-l_t}$ and $j_t = 0, \ldots, 2^{l_t}$. For any grid $\Omega_{\underline{l}}$ we define the associated space $V_{\underline{l}}$ of piecewise d-linear functions

$$V_{\underline{l}} := \text{span}\{\phi_{\underline{l},\underline{j}} \mid j_t = 0, \ldots, 2^{l_t},\ t = 1, \ldots, d\}, \tag{6}$$

which is spanned by the conventional basis of d-dimensional piecewise d-linear hat functions

$$\phi_{\underline{l},\underline{j}}(\underline{x}) := \prod_{t=1}^{d} \phi_{l_t,j_t}(x_t). \tag{7}$$

The one-dimensional functions $\phi_{l,j}(x)$ with support

$$[x_{l,j} - h_l, x_{l,j} + h_l] \cap [0,1] = [(j-1)h_l, (j+1)h_l] \cap [0,1]$$

are defined by

$$\phi_{l,j}(x) = \begin{cases} 1 - |x/h_l - j|, & x \in [(j-1)h_l, (j+1)h_l] \cap [0,1], \\ 0, & \text{otherwise}, \end{cases} \tag{8}$$

see Figure 1(a) for an example of a two-dimensional basis function.

The multi-index $\underline{l} \in \mathbb{N}^d$ denotes the level, i.e. the discretization resolution, be it of a grid $\Omega_{\underline{l}}$, of a space $V_{\underline{l}}$, or of a function $f_{\underline{l}}$, whereas the multi-index $\underline{j} \in \mathbb{N}^d$ gives the position of a grid point $x_{\underline{l},\underline{j}}$ or its corresponding basis function $\phi_{\underline{l},\underline{j}}$.

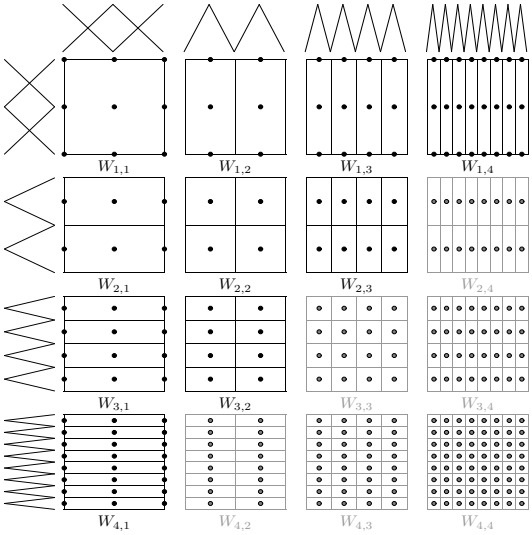


FIGURE 2. Supports of the basis functions of the hierarchical subspaces $W_{\underline{l}}$. All spaces are used for V_4, only the marked upper left triangle is used for V_4^s.

We now define a hierarchical difference space $W_{\underline{l}}$ via

$$W_{\underline{l}} \;:=\; V_{\underline{l}} \;\setminus\; \bigoplus_{t=1}^{d} V_{\underline{l}-\underline{e}_t}, \tag{9}$$

where $\underline{e}_t$ is the t-th unit vector. In other words, $W_{\underline{l}}$ is spanned by all $\phi_{\underline{k},\underline{j}} \in V_{\underline{l}}$ which are not included in any of the spaces $V_{\underline{k}}$ smaller[1] than $V_{\underline{l}}$. To complete the definition, we formally set $V_{\underline{l}} := \emptyset$, if $l_t = 0$ for at least one $t \in \{1,\dots,d\}$. As can be easily seen from (6) and (9), the definition of the index set

$$\mathsf{B}_{\underline{l}} \;:=\; \left\{ \underline{j} \in \mathbb{N}^d \;\middle|\; \begin{array}{ll} j_t = 1,\dots,2^{l_t}-1, & j_t \text{ odd}, t = 1,\dots,d, \text{ if } l_t > 1, \\ j_t = 0,1,2, & t = 1,\dots,d, \text{ if } l_t = 1 \end{array} \right\} \tag{10}$$

leads to

$$W_{\underline{l}} \;=\; \text{span}\{\phi_{\underline{l},\underline{j}} | \underline{j} \in \mathsf{B}_{\underline{l}}\}. \tag{11}$$

With these hierarchical difference spaces we now can define a multilevel subspace decomposition and write $V_{\underline{l}}$ as a direct sum of subspaces

$$V_{\underline{l}} \;:=\; \bigoplus_{k_1=1}^{l_1} \cdots \bigoplus_{k_d=1}^{l_d} W_{\underline{k}} = \bigoplus_{\underline{k} \le \underline{l}} W_{\underline{k}}. \tag{12}$$

Here and in the following "$\le$" refers to the element-wise relation for multi-indices. Furthermore, $|\underline{l}|_\infty := \max_{1\le t\le d} l_t$ and $|\underline{l}|_1 := \sum_{t=1}^d l_t$ are the discrete ℓ^∞- and the discrete ℓ^1-norm of $\underline{l}$, respectively.

The family of functions

$$\left\{ \phi_{\underline{l},\underline{j}} \,\middle|\, \underline{j} \in \mathsf{B}_{\underline{l}} \right\}_{\underline{l}=(1,\dots,1)}^{(n,\dots,n)} \tag{13}$$

is just the hierarchical basis [8, 27] of $V_n (:= V_{(n,\dots,n)})$, which generalizes the one-dimensional hierarchical basis, see Figure 3(a), to the d-dimensional case with a tensor product ansatz.

[1]We call a discrete space $V_{\underline{k}}$ smaller than a space $V_{\underline{l}}$ if $\forall_t k_t \le l_t$ and $\exists t : k_t < l_t$. In the same way a grid $\Omega_{\underline{k}}$ is smaller than a grid $\Omega_{\underline{l}}$.

Observe that the supports of the basis functions $\phi_{\underline{l},\underline{j}}(\underline{x})$, which span $W_{\underline{l}}$, are disjoint for $\underline{l} > 1$. Figure 2 gives a representation of the supports of the basis functions of the difference spaces W_{l_1,l_2} forming V_4.

Now, each function $f \in V_n$ can be represented as

$$f(\underline{x}) \;=\; \sum_{|\underline{l}|_\infty \le n} \sum_{\underline{j} \in \mathsf{B}_{\underline{l}}} \alpha_{\underline{l},\underline{j}} \cdot \phi_{\underline{l},\underline{j}}(\underline{x}), \tag{14}$$

where $\alpha_{\underline{l},\underline{j}} \in \mathbb{R}$ are the coefficients of the representation in the hierarchical tensor product basis. In one dimension it is easy to see that they specify what has to be added to the hierarchical representation of level $l-1$ to obtain that of level l. This generalizes to higher dimensions accordingly and specifies what has to be added to the representation in space $\bigoplus_{t=1}^{d} V_{\underline{l}-\underline{e}_t}$ to obtain a representation in $V_{\underline{l}}$.

The number of basis functions which describe a $f \in V_n$ in nodal or hierarchical basis is $(2^n+1)^d$. For example, a resolution of 17 points in each dimension, i.e. $n = 4$, for a ten-dimensional problem needs more than $2 \cdot 10^{12}$ coefficients, i.e. we encounter the curse of dimensionality.

On the other hand it was observed that for a function f with bounded second mixed derivatives it holds

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

where $f_{\underline{l}} := \sum_{\underline{j} \in \mathsf{B}_{\underline{l}}} \alpha_{\underline{l},\underline{j}} \cdot \phi_{\underline{l},\underline{j}}(\underline{x}) \in W_{\underline{l}}$ denotes its hierarchical components and $|f|_{H^2_{mix}} := \|\frac{\partial^{2d}}{\partial x_1^2 \ldots x_d^2} f\|_2$ is the H^2_{mix}-semi-norm, see [3, 28] for details.

Motivated by this dependence of the "importance" of the hierarchical components $f_{\underline{l}}$ on the size of the supports of the involved basis functions, i.e. $|\operatorname{supp} \phi_{\underline{l},\underline{j}}| = 2^d \cdot 2^{-\underline{l}}$, Zenger [28] and Griebel [14] introduced so-called *sparse grids*, where hierarchical basis functions with a small support, and therefore a small contribution to the function representation, are not included in the discrete space of level n anymore.

Formally, the sparse grid function space $V_n^s \subset V_n$ is defined as

$$V_n^s \;:=\; \bigoplus_{|\underline{l}|_1 \le n+d-1} W_{\underline{l}}, \tag{15}$$

where in the definition (12) for V_n in terms of hierarchical subspaces the condition $|\underline{l}|_\infty \le n$ is replaced by $|\underline{l}|_1 \le n+d-1$. In Figure 2 the employed subspaces $W_{\underline{l}}$ are given in black, the spaces omitted in comparison to (12) are given in grey.

Every $f \in V_n^s$ now can be represented, analogous to (14), as

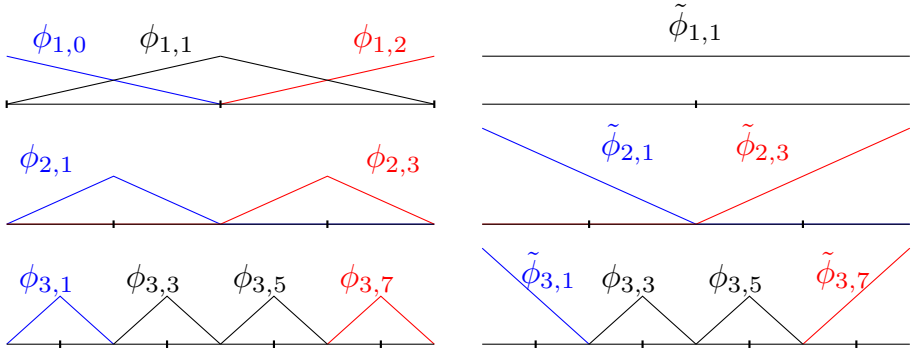
$$f_n^s(\underline{x}) \;=\; \sum_{|\underline{l}|_1 \le n+d-1} \sum_{\underline{j} \in \mathsf{B}_{\underline{l}}} \alpha_{\underline{l},\underline{j}} \phi_{\underline{l},\underline{j}}(\underline{x}). \tag{16}$$

The resulting grid which corresponds to the approximation space V_n^s is called sparse grid and is denoted by Ω_n^s, an example in three dimensions is given in Figure 1(b).

The sparse grid space V_n^s has a size of order $\dim V_n^s = \mathcal{O}(2^n \cdot n^{d-1})$, see [3]. It thus depends on the dimension d to a much smaller degree than V_n whose number of degrees of freedom is $\mathcal{O}(2^{nd})$. Note that for the approximation of a function f by a sparse grid function $f_n^s \in V_n^s$ the error relation

$$\|f - f_n^s\|_2 = \mathcal{O}\left(2^{-2n} \cdot n^{d-1}\right)$$

holds, provided that f fulfils the smoothness requirement $|f|_{H^2_{mix}} < \infty$ [3]. Therefore, sparse grids need much less points in comparison to a full grid to obtain an error of the same size.



(a) Hierarchical basis functions $\phi_{i,j}$ up to level 3 (b) Modified basis functions $\phi_{i,j}$ up to level 3

FIGURE 3. Standard and modified hierarchical basis functions.

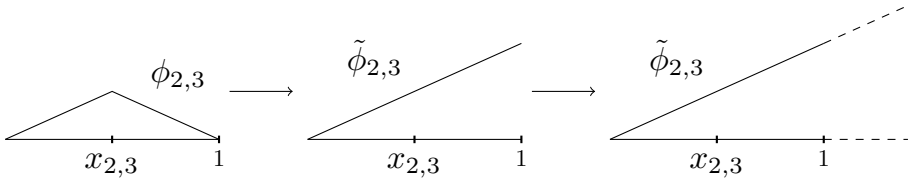


FIGURE 4. The modified hierarchical basis function $\tilde{\phi}_{2,3}$ stems from $\phi_{2,3}$ by folding up the half of the hat function near the boundary. It then can be straightforwardly extended to the outside of the domain Ω using linear extrapolation.

2.1. **Modified basis functions on the boundary.** Looking more closely at the number of basis functions used for a regular sparse grid of level n, we observe that the ratio of points on the boundary versus that in the interior grows significantly with increasing dimensionality [25], i.e. more and more grid points are spent on $\partial\Omega$. When dealing with functions that are zero or fixed on $\partial\Omega$, e.g. in case of Dirichlet boundary conditions, one could just work without the two basis functions $\phi_{1,0}$ and $\phi_{1,2}$ to avoid this effect. But since in our application the function values on the boundary are not known a priori, we can not employ this approach.

Instead, we proceed as follows: We still omit the grid points on the boundary but additionally modify the interior basis functions so that they extrapolate towards the boundary, as it was proposed in [25]. Figure 3(b) illustrates this modification for the case of level $n = 3$, the interior basis functions nearest to the boundary are "folded up" for any level, see also Figure 4 (mid). The d-dimensional basis functions are again obtained as tensor products of the one-dimensional ones in the same way as described in the previous section. This modification can be advantageous especially in settings where the accuracy close to the boundary is not required to be very high. In our case, this will correspond to the situation where the zero level is not located close to the boundary. Another advantage of these modified basis functions is the fact that we are able to extrapolate values for points outside of the domain, we then *linearly* extend the basis functions $\tilde{\phi}_{l,i}$, $i = 1, 2^l - 1$ to the exterior, which is illustrated in Figure 4 (right). This property will later be used for those examples where trajectories which leave the domain need to be considered.

2.2. **Spatially adaptive sparse grids.** The sparse grid structure (15) defines an a priori selection of grid points that is optimal if certain smoothness conditions are met, i.e. if the function has bounded second mixed derivatives, and no further knowledge of the function is known or used. If the aim is to approximate functions which either do not fulfil this

smoothness condition at all or show strongly varying behaviour due to finite but nevertheless locally large derivatives, then adaptive refinement may be used. There, depending on the characteristics of the problem and function at hand, adaptive refinement strategies decide which points and corresponding basis functions should be incrementally added to the sparse grid representation to increase the accuracy.

In the sparse grid setting, usually an error indicator stemming directly from the hierarchical basis is employed [12, 15, 25]: depending on the size of the hierarchical surplus $\alpha_{\underline{l},\underline{j}}$ it is decided whether a basis function should be marked for further improvement or not. This is based on two observations: First, the hierarchical surplus gives the absolute change in the discrete representation at point $x_{\underline{l},\underline{j}}$ due to the addition of the corresponding basis function $\phi_{\underline{l},\underline{j}}$, i.e. it measures its contribution to a given sparse grid representation (16) in the maximum-norm. And second, a hierarchical surplus represents discrete second mixed derivatives and hence can be interpreted as a measure of the smoothness of the considered function at point $x_{\underline{l},\underline{j}}$.[2]

In the adaptive procedure we use a set $\mathcal{I}$ to track the indices of the employed basis functions and denote the corresponding sparse grid by $\Omega_{\mathcal{I}}$ and the associated sparse grid space by $V_{\mathcal{I}}$, respectively. We start with a coarse initial sparse grid function $f_n^s \in V_n^s$ for some given small n as in (16). The index set is thus initialized as $\mathcal{I} := \{(\underline{l},\underline{j}) \mid |\underline{l}|_1 \leq n+d-1\}$. We proceed as follows: If, for any given index $(\underline{l},\underline{j}) \in \mathcal{I}$, we have

$$|\alpha_{\underline{l},\underline{j}}| \cdot \|\phi_{\underline{l},\underline{j}}\| > \varepsilon \tag{17}$$

for some given constant $\varepsilon > 0$, then the index will be *marked*. Here, $\|\cdot\|$ is typically either the L^∞- or L^2-norm, but other norms or weighted mixtures of norms are used in practice as well. If an index is marked, all its $2d$ so-called *children* will be added to the index set $\mathcal{I}$ to refine the discretization, i.e. all $(\tilde{\underline{l}},\tilde{\underline{j}})$ with $\tilde{\underline{l}} = \underline{l} + \underline{e}_t$ and $\tilde{\underline{j}} = \underline{j} + j_t\,\underline{e}_t \pm 1$ will be added to $\mathcal{I}$ for $t = 1,\ldots,d$. For the indices added that way it is possible that not all *parents* in all dimensions are already contained in the grid; note that in such cases, for algorithmic and consistency reasons, these missing parents have to be added to $\mathcal{I}$ as well. Thus for any $(\underline{l},\underline{j}) \in \mathcal{I}$ it holds that all parents $(\tilde{\underline{l}},\tilde{\underline{j}})$ with $\tilde{\underline{l}} \leq \underline{l}$ and $supp(\phi_{\tilde{\underline{l}},\tilde{\underline{j}}}) \cap supp(\phi_{\underline{l},\underline{j}}) \neq \emptyset$ are also in the index set $\mathcal{I}$. In other words, holes in the hierarchical structure are not allowed. The refinement step is repeated until no indices are added anymore. In Algorithm 1 we give the full adaptive refinement procedure. Note that if a global error criterion is available one can perform an additional outer loop with successively decreasing ε until the measured global error falls below a given threshold ε_{glob}.

In a similar way one can use the value $|\alpha_{\underline{l},\underline{j}}| \cdot \|\phi_{\underline{l},\underline{j}}\|$ to *coarsen* the grid in case of over-refinement. If this value is smaller than some coarsening constant η, and no children of $(\underline{l},\underline{j})$ are in $\mathcal{I}$, the index will be removed from this set. In Algorithm 2 we give the coarsening step, where the procedure is repeated until no indices are being removed. The coarsening will in particular be relevant once we consider time-dependent problems where the region in need of a higher resolution moves over the domain. This will be described more precisely in Section 3.

3. Semi-Lagrangian scheme

We will now use adaptive sparse grids in a new semi-Lagrangian scheme for Hamilton-Jacobi Bellman equations. Here, we focus on the equation

$$v_t(t,\underline{x}) + \max_{\beta(t)\in\mathcal{B}} (f(\underline{x},\beta(t)) \cdot \nabla v(t,\underline{x})) = 0, \qquad t \geq 0, \quad \underline{x} \in \Omega, \tag{18a}$$

$$v(0,\underline{x}) = \varphi(\underline{x}), \qquad \underline{x} \in \Omega. \tag{18b}$$

The state dynamics function f is assumed to be Lipschitz continuous. We are interested in the zero level set of $v(t,\cdot)$. The region $\Omega(t) := \{\underline{x} \,|\, v(t,\underline{x}) \leq 0\}$ is also called the

[2]Here, also many other approaches exist, which are based on interpolets, prewavelets or wavelets, cf. [12].

Algorithm 1: Spatially Adaptive Refinement

Data: initial index set $\mathcal{I}$, refinement threshold ε and function evaluation F
Result: refined index set $\mathcal{I}$, adaptive sparse grid approximation of F in $V_{\mathcal{I}}$
for *all indices* $(\underline{l},\underline{j}) \in \mathcal{I}$ **do**
  compute $F(\underline{x}_{\underline{l},\underline{j}})$ ▷ `evaluate` F `at initial grid points`
compute hierarchical values $\alpha_{\underline{l},\underline{j}}$ for all indices, see e.g. [15]
while *indices are added to* $\mathcal{I}$ **do**
  for $(\underline{l},\underline{j}) \in \mathcal{I}$ **do** ▷ `look at all indices`
    if $|\alpha_{\underline{l},\underline{j}}| \cdot \|\phi_{\underline{l},\underline{j}}\| > \varepsilon$ **then**
      for $t = 1,\dots,d$ **do** ▷ `hierarchical surplus is large`
        if $(\tilde{\underline{l}},\tilde{\underline{j}}) \notin \mathcal{I}$ *for* $\tilde{\underline{l}} = \underline{l} + \underline{e}_t$ *and* $\tilde{\underline{j}} \in \{\underline{j} + j_t\,\underline{e}_t \pm 1\}$ **then**
          $\mathcal{I} = \mathcal{I} \cup (\tilde{\underline{l}},\tilde{\underline{j}})$ ▷ `add children which are not in` $\mathcal{I}$
  check $\forall (\underline{l},\underline{j}) \in \mathcal{I}$ holds: $(\tilde{\underline{l}},\tilde{\underline{j}}) \in \mathcal{I}$ for $\tilde{\underline{l}} \le \underline{l}$ and $supp(\phi_{\tilde{\underline{l}},\tilde{\underline{j}}}) \cap supp(\phi_{\underline{l},\underline{j}}) \neq \emptyset$
  for *all added indices* $(\underline{l},\underline{j}) \in \mathcal{I}$ **do**
    compute $F(\underline{x}_{\underline{l},\underline{j}})$ ▷ `evaluate` F `at new grid points`
  compute hierarchical values $\alpha_{\underline{l},\underline{j}}$ for newly added indices, see e.g. [15]

Algorithm 2: Spatially Adaptive Coarsening

Data: index set $\mathcal{I}$, coarsening threshold η, and $\alpha_{\underline{l},\underline{j}}\ \forall (\underline{l},\underline{j}) \in \mathcal{I}$
Result: coarsened index set $\mathcal{I}$
while *indices are removed from* $\mathcal{I}$ **do**
  for $(\underline{l},\underline{j}) \in \mathcal{I}$ **do** ▷ `look at all indices`
    if $|\alpha_{\underline{l},\underline{j}}| \cdot \|\phi_{\underline{l},\underline{j}}\| < \eta$ **then** ▷ `hierarchical surplus is small`
      if $\forall t = 1,\dots,d$: $(\tilde{\underline{l}},\tilde{\underline{j}}) \notin \mathcal{I}$ *for* $\tilde{\underline{l}} = \underline{l} + \underline{e}_t$ *and* $\tilde{\underline{j}} \in \{\underline{j} + j_t\,\underline{e}_t \pm 1\}$ **then**
        $\mathcal{I} = \mathcal{I} \backslash (\underline{l},\underline{j})$ ▷ `remove if no children in` $\mathcal{I}$

(backward) "reachable set". As mentioned in the introduction, it corresponds to the set of points that can be reached at time $t \ge 0$ by some trajectory that starts from $\Omega(0) := \{\underline{x} \,|\, \varphi(\underline{x}) \le 0\}$, see [22].

In the following we will introduce a semi-Lagrangian (SL) scheme for the numerical treatment of (18). It is inspired by the dynamic programming principle, which is here:

$$v(t+\tau,\underline{x}) = \min_{\beta \in L^\infty([0,\tau];\mathcal{B})} v\left(t, y_{\underline{x}}^{\beta}(-\tau)\right) \quad \forall \tau \ge 0. \tag{19}$$

Together with $v(0,\underline{x}) = \varphi(\underline{x})$ this is equivalent to equation (18), see [2]. Here, $y_{\underline{x}}^{\beta}$ denotes the absolutely continuous solution of the corresponding state dynamics system (4). Hence $y_{\underline{x}}^{\beta}(-\tau)$ in (19) is a solution of (4) backwards in time, it gives the point from which $\underline{x}$ is reached under control β in the time interval $[t, t+\tau)$.

For the discretization of the time interval $[0,T]$, let K be the number of time steps, let $\tau := T/K$ be the time step and set $t_k := k\tau,\ k \in \{0,\cdots,K-1\}$. Note that from now on we will only consider controls that are constant during a time step, i.e. $\beta(t) \equiv b_k, t \in [t_k, t_k+\tau]$, and we will denote the corresponding trajectory as $y_{\underline{x}}^{b_k}(-\tau)$. Furthermore, the computation of $y_{\underline{x}}^{b_k}(-\tau)$ can be done in an approximate way without changing the core of the scheme, by e.g. using some high order Runge Kutta method. In the numerical examples of Section 4, there will be no need for such approximations since the considered $y_{\underline{x}}^{b_k}(-\tau)$ can be calculated analytically.

Let us for now consider a suitably chosen sparse grid $\Omega_{\mathcal{I}}$. The semi-Lagrangian (SL) scheme to solve (18) in the time interval $[0, T]$ on $\Omega_{\mathcal{I}}$ then takes the following form:

- Initialize $v_0 \in V_{\mathcal{I}}$ by interpolating φ.
- Iterate for $k = 0, \dots, K-1$,

$$v_{k+1}(\underline{x}) := \min_{b_k \in \mathcal{B}} v_k\left(y_{\underline{x}}^{b_k}(-\tau)\right) \quad \forall \underline{x} \in \Omega_{\mathcal{I}}. \tag{20}$$

Here, $v_k \in V_{\mathcal{I}}$ is the numerical solution computed by the scheme at time t_k and the value $v_k\left(y_{\underline{x}}^{b_k}(-\tau)\right)$ denotes the sparse grid interpolation of v_k at point $y_{\underline{x}}^{b_k}(-\tau)$. In general, depending on the initial function and the state dynamics, a specific treatment of the case $y_{\underline{x}}^{b_k}(-\tau) \notin \Omega$ is necessary; for our approach we will give details in the numerical section. Moreover note that for our experiments in Section 4 the minimization over the set $\mathcal{B}$ can be done in a straightforward way by evaluating the function values for each possible action $b_k \in \mathcal{B}$. In case this number is too large, or if the set $\mathcal{B}$ is infinite, then a minimisation procedure which uses only evaluations of the objective function, and not its derivatives, could be performed without changing the main steps of the SL-SG scheme. We refer for instance to [4], where this approach was successfully employed for a semi-Lagrangian scheme using Brent's minimization algorithm.

3.1. **Spatially adaptive semi-Lagrangian sparse grid scheme.** With an SL scheme using spatially adaptive sparse grids (SL-SG scheme), the employed selection of grid points needs to be changed from time step t_k to t_{k+1} for a fully efficient scheme to achieve the smallest grid for a given accuracy at each time step, i.e. for each k there is a corresponding adaptive sparse grid $\Omega_{\mathcal{I}(k)}$, defined by the index set $\mathcal{I}(k)$, with an approximate solution $v_k \in V_{\mathcal{I}(k)}$. To this end, we first discretize in time and then in space, this approach is often called Rothe-method.

To compute the approximate solution v_{k+1} for the next time step t_{k+1} the new index set $\mathcal{I}(k+1)$ is initialized by $\mathcal{I}(k)$. For this index set the effect of the actions is then evaluated and the approximation at time t_{k+1} is computed for $\mathcal{I}(k+1)$ as

$$v_{k+1}(\underline{x}_{\underline{l},\underline{j}}) := \min_{b_k \in \mathcal{B}} v_k\left(y_{\underline{x}_{\underline{l},\underline{j}}}^{b_k}(-\tau)\right) \quad \forall (\underline{l}, \underline{j}) \in \mathcal{I}(k+1).$$

With that, one can compute the hierarchical surplus $\alpha_{\underline{l},\underline{j}}$ for all $(\underline{l}, \underline{j}) \in \mathcal{I}(k+1)$, mark those indices with coefficients which are larger than the refinement constant ε, add the corresponding child indices to $\mathcal{I}(k+1)$ and repeat. In other words, we apply Algorithm 1 where the function evaluation $F(\underline{x})$ is $\min_{b_k \in \mathcal{B}} v_k\left(y_{\underline{x}}^{b_k}(-\tau)\right)$. After the new sparse grid $\Omega_{\mathcal{I}(k+1)}$ is sufficiently refined we apply Algorithm 2 to remove the superfluous indices up to some minimal level. This coarsening not only concerns children added newly in the refinement step which are not needed in the end, but is in particular relevant to take the movement of the region of interest in time into account, which corresponds to the state dynamics. In other words, the higher resolved area of the state space changes in time. The full spatially adaptive semi-Lagrangian Sparse Grid scheme is given in Algorithm 3.

3.2. **Non-smooth initial data.** We now explain how the scheme can be generalized to the case of non-smooth initial data of the form

$$\varphi(\underline{x}) := \min_{i=1,\dots,J} (a_i + \varphi_i(\underline{x})), \tag{21}$$

where $J \geq 1$, $a_i \in \mathbb{R}$ and where $\varphi_i : \mathbb{R}^d \to \mathbb{R}$ are some given functions. Each function φ_i will be assumed to be sufficiently smooth to allow for the efficient application of the

Algorithm 3: Adaptive SL-SG scheme

Data: suitable initial index set $\mathcal{I}$, refinement constant ε and coarsening constant η
Result: sequence of adaptive sparse grid solutions $v_k \in V_{\mathcal{I}(k)}$ for $k = 0, ..., K$
call Alg. 1 with $\mathcal{I}$, ε and $F(\underline{x}) = \varphi(\underline{x})$ ▷ `interpolate` φ `by` $v_0 \in V_{\mathcal{I}(0)}$
call Alg. 2 with $\mathcal{I}(0)$, η and v_0 ▷ `coarsen` v_0
for $k = 0, \ldots, K-1$ **do** ▷ `iterate in time`
  call Alg. 1 with $\mathcal{I}(k)$, ε and $F(\underline{x}) = \min_{b_k \in \mathcal{B}} v_k(y_{\underline{x}}^{b_k}(-\tau))$ ▷ `compute`
  $v_{k+1} \in V_{\mathcal{I}(k+1)}$ call Alg. 2 with $\mathcal{I}(k+1)$, η and v_{k+1} ▷ `coarsen` v_{k+1}

SL-SG scheme to the following subproblems for $i = 1, \ldots, J$:

$$v_t^{(i)}(t,\underline{x}) + \max_{\beta(t)\in\mathcal{B}} \Big(f(\underline{x},\beta(t)) \cdot \nabla v^{(i)}(t,\underline{x}) \Big) = 0, \qquad t \in (0,T), \quad \underline{x} \in \Omega, \tag{22a}$$

$$v^{(i)}(0,\underline{x}) = \varphi_i(\underline{x}), \qquad \underline{x} \in \Omega. \tag{22b}$$

Note that by using the equivalent formula (3) for initial data of the form (21), the solution of (18) is also given as

$$v(t,\underline{x}) = \min_{i=1,\ldots,J} \Big(a_i + v^{(i)}(t,\underline{x}) \Big).$$

Here each $v^{(i)}$ is the solution of the subproblem (22) associated to φ_i.

Indeed, it is well known (see [19], the "max-plus approach" of [1], and the references therein) that for a convex Hamiltonian

$$H(\underline{x},\underline{p}) := \max_{\underline{b}\in\mathcal{B}} \big(f(\underline{x},\underline{b}) \cdot \underline{p} \big)$$

the solution $v(t,\cdot)$ of $v_t + H(\underline{x}, \nabla v) = 0$ with initial data $v(0,\cdot) = \varphi(\cdot)$ has a "min-plus" property: For any φ decomposed in the form (21) we have $v(t,\underline{x}) = \min_i (a_i + (S^t\varphi_i)(\underline{x}))$, where S^t denotes the semigroup associating φ to its corresponding solution $v(t,\cdot) = S^t\varphi$. Therefore, we can apply an adaptive SL-SG scheme to each initial data φ_i from (21), where values and grid points are adapted with respect to φ_i. In order to obtain the value of $v(T,\underline{x})$ at the final computational time T and for a given point $\underline{x}$, we minimize the J values $a_i + v^{(i)}(T,\underline{x})$ by using the corresponding sparse grid representation of each $v^{(i)}(T,\cdot)$ which we computed separately up to time T. Hence this scheme needs J sparse grid functions. However the computation of the $v^{(i)}$ can be done independently and therefore completely in parallel. This generalized scheme will be applied to the example 5 in the following section.

4. Numerical results

We have tested the adaptive sparse grid procedure from Algorithm 3 on various numerical examples. We shall show two types of results. The first one concerns numerical convergence, and the second one deals with the scaling behaviour of the proposed scheme with respect to the dimension $d \geq 2$.

In our tests we only compute the L^∞- and L^2-error against the known exact solution v localized around the zero level set. This localization takes place for two reasons. First, the region around the zero level set is the important one for our applications, and second and more importantly, we can thus reduce the computational effort for the error calculation in higher dimensions. To this end, we define the region

$$Q_c^t(v) = \{x \,|\, |v(t,\underline{x})| \leq c\}$$

around the zero level set of the exact solution v at time t, where $c > 0$ is a given threshold. In general one would define $Q_c^t(v)$ such that $|v(t,\underline{x})|/\|\nabla v(t,\underline{x})\| \leq c$, but since in our examples we will have $\|\nabla v\| \sim 1$ near the zero level set we simplify the criterion and use

only $|v(t,\underline{x})| \leq c$. We now can localize any function u, in particular the exact solution v and its numerical approximation v_k, by setting the function to zero outside of $Q_c^t(v)$, i.e.

$$u_{loc}(\underline{x}) := \begin{cases} u(\underline{x}), & \text{if } \underline{x} \in Q_c^t(v), \\ 0, & \text{if } \underline{x} \notin Q_c^t(v). \end{cases} \tag{23}$$

To simplify the presentation we do not indicate the dependence of u_{loc} on $Q_c^t(v)$ since $Q_c^t(v)$ straightforwardly follows from the context for any employed u_{loc}. Now we can define the relative localized L^∞-error at time t_k by

$$\frac{\|v_{loc}(t_k,\cdot) - v_{k,loc}(\cdot)\|_{L^\infty}}{\|v_{loc}(t_k,\cdot)\|_{L^\infty}}.$$

To this end, let $v(t_k,\cdot)$ denote the exact solution at time t_k, whereas $v_k(\cdot)$ denotes the sparse grid approximation function at time t_k. Their localized counterparts v_{loc} and $v_{k,loc}$ are then defined by (23) using $Q_c^{t_k}(v)$. The discrete local L^∞-error at time t_k is now further approximated using an equidistant tensor grid $\mathcal{G}$ covering the whole domain, i.e.

$$e_{L^\infty_{loc}} = \frac{\max\limits_{\underline{x}\in\mathcal{G}} |v_{loc}(t_k,\underline{x}) - v_{k,loc}(\underline{x})|}{c}, \tag{24}$$

where $\|v_{loc}(t_k,\cdot)\|_{L^\infty} = c$.

The relative local L^2-error is defined in a similar fashion by

$$e_{L^2_{loc}} = \frac{\left(\sum\limits_{\underline{x}\in\mathcal{G}} |v_{loc}(t_k,\underline{x}) - v_{k,loc}(\underline{x})|^2\right)^{1/2}}{\left(\sum\limits_{\underline{x}\in\mathcal{G}} |v_{loc}(t_k,\underline{x})|^2\right)^{1/2}}. \tag{25}$$

For the computation of these localized relative errors only points $\underline{x} \in \mathcal{G} \cap Q_c^t(v)$ are now relevant due to the localization. In our examples we use $N_x = 70$ points in each direction for the grid $\mathcal{G}$ and $c = 0.2$. Note that in higher dimensions the error computation using the grid $\mathcal{G}$ will still dominate the computational time, even in the localized form. Let us remark that we observe with a relative local L^∞ error of roughly 10^{-2} already a quite good localization of a front in our numerical experiments.

To analyze the rate of convergence of the adaptive sparse grid scheme with respect to - for instance - the refinement constant ε, we compute the quantity

$$\varrho_{\varepsilon_j,e_j} := \frac{\log(e_j/e_{j-1})}{\log(\varepsilon_j/\varepsilon_{j-1})},$$

where e_j is the error corresponding to the refinement constant ε_j, see Section 2.2. Here, j enumerates the sequence of refinement constants used and it will later correspond to the rows in the tables of the numerical results. The order of the cost complexity ϱ_{N_j,e_j} is defined analogously, where N_j is the number of sparse grid points in the adaptive sparse grid which results from the refinement constant ε_j.

In the following, we first test our new adaptive SL-SG scheme on two linear examples and discuss its properties. This corresponds to the particular case of the control set $\mathcal{B}$ in (1) consisting of just one element. Then, we apply it to more involved nonlinear HJB equations.

In all examples we start with a sparse grid of level 3, which is also the minimal level for the coarsening procedure. Moreover, we always employ the modified basis functions of Figure 3(b) from section 2.1 in our sparse grid algorithms. The coarsening constant η corresponding to a given ε is set to $\eta = \varepsilon/5$ in all of the following examples. As norm in the refinement condition (17) we always use $\|.\|_\infty$.

$\underline{t=0}$

ε	N	$e_{L^\infty_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$	$e_{L^2_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$
8.00_{-3}	133	9.73_{-3}			1.15_{-2}		
2.00_{-3}	261	2.44_{-3}	1.00	-2.05	2.99_{-3}	0.97	-2.00
5.00_{-4}	517	6.08_{-4}	1.00	-2.03	7.50_{-4}	1.00	-2.02
1.25_{-4}	1,029	1.52_{-4}	1.00	-2.01	1.86_{-4}	1.01	-2.03
3.13_{-5}	2,053	3.81_{-5}	1.00	-2.00	4.67_{-5}	1.00	-2.00
7.81_{-6}	4,101	9.53_{-6}	1.00	-2.00	1.16_{-5}	1.00	-2.01
1.95_{-6}	8,197	2.38_{-6}	1.00	-2.00	2.88_{-6}	1.01	-2.02
4.88_{-7}	16,389	5.88_{-7}	1.01	-2.02	6.76_{-7}	1.05	-2.09

$\underline{t=1}$

ε	N	$e_{L^\infty_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$	$e_{L^2_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$
8.00_{-3}	105	1.14_{-1}			1.92_{-1}		
2.00_{-3}	201	2.85_{-2}	1.00	-2.14	4.79_{-2}	1.00	-2.14
5.00_{-4}	393	7.12_{-3}	1.00	-2.07	1.20_{-2}	1.00	-2.07
1.25_{-4}	777	1.78_{-3}	1.00	-2.03	2.99_{-3}	1.00	-2.04
3.13_{-5}	1,547	4.45_{-4}	1.00	-2.01	7.48_{-4}	1.00	-2.01
7.81_{-6}	3,081	1.11_{-4}	1.00	-2.01	1.87_{-4}	1.00	-2.01
1.95_{-6}	6,155	2.78_{-5}	1.00	-2.00	4.68_{-5}	1.00	-2.00
4.88_{-7}	12,297	6.95_{-6}	1.00	-2.00	1.17_{-5}	1.00	-2.00

TABLE 1. Example 1, convergence for $d = 2$, initial data ($t = 0$) and terminal data ($t = 1$), after $K = 12$ time steps.

Concerning the boundary treatment, we consider three different types of examples. At first there are problems where the trajectory is known to stay inside the domain, hence there is no need for a specific treatment for the boundary (Ex. 2 and 3). For the second type, we know that there is always at least one trajectory staying inside Ω and those leaving the domain can be safely ignored (Ex. 5). If this is not the case, i.e. if possibly all trajectories end up outside of Ω, we utilise the fact that our modified basis functions enable us to extrapolate function values even outside of the domain (see 2.1), and use the extrapolated values for the scheme if necessary (Ex. 1 and 4).

Example 1. We consider the following advection equation:

$$\begin{aligned} v_t + f \cdot \nabla v = 0, \qquad & t \geq 0,\ \underline{x} \in \Omega, \\ v(0, \underline{x}) = \varphi(\underline{x}), \qquad & \underline{x} \in \Omega, \end{aligned}$$

where $\Omega = (-2, 2)^d$, $f \equiv f(\underline{x}) = (-1, \ldots, -1)$ and

$$\varphi(\underline{x}) := \|\underline{x} - \underline{a}\|_2^2 - r_0^2, \quad \text{with } r_0 = 0.5.$$

The vector $\underline{a}$ is defined by the first d coordinates of the sequence $\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{5}, \frac{1}{7}, \frac{1}{11}, \frac{1}{13}\right)$, therefore $\underline{a}$ does not correspond to a sparse grid point. The level set $\{\underline{x} \,|\, \varphi(\underline{x}) = 0\}$ represents a sphere in $\mathbb{R}^d$ of radius r_0 centered at $\underline{a}$. This system describes the movement of the initial function φ, and hence its zero level set, in direction f. The exact solution for this advection equation is given by $v(t, \underline{x}) = \varphi(\underline{x} - ft)$.

We start with a convergence analysis for dimensions two and four. In Table 1 we first consider the two-dimensional case. For different refinement constants ε we give for $t = 0$ (initial data) and for $t = 1$ (terminal time) the number N of resulting sparse grid points, i.e. $N = |\mathcal{I}(t)|$, and the local errors $e_{L^\infty_{loc}}$ and $e_{L^2_{loc}}$ which were obtained.

We observe that $\varrho_{\varepsilon,e} \sim 1$, hence the computed errors $e_{L^\infty_{loc}}$ and $e_{L^2_{loc}}$ depend linearly on the parameter ε of the adaptive refinement procedure. We therefore can reduce the error in a controlled fashion by correspondingly reducing the parameter ε in the refinement procedure. Moreover, we approximately have $\varrho_{N,e} \sim -2$, hence the computed errors are of second order with respect to the total number of sparse grid points N.

$\underline{t=0}$

ε	N	$e_{L^\infty_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$	$e_{L^2_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$
8.00_{-3}	281	1.95_{-2}			2.28_{-2}		
2.00_{-3}	537	4.88_{-3}	1.00	-2.14	5.73_{-3}	1.00	-2.13
5.00_{-4}	1,049	1.20_{-3}	1.01	-2.09	1.42_{-3}	1.01	-2.09
1.25_{-4}	2,073	3.05_{-4}	0.99	-2.01	3.61_{-4}	0.99	-2.01
3.13_{-5}	4,121	7.62_{-5}	1.00	-2.02	9.00_{-5}	1.00	-2.02
7.81_{-6}	8,217	1.91_{-5}	1.00	-2.01	2.23_{-5}	1.01	-2.02
1.95_{-6}	16,409	4.76_{-6}	1.00	-2.00	5.39_{-6}	1.02	-2.05
4.88_{-7}	32,793	1.01_{-6}	1.12	-2.24	1.10_{-6}	1.15	-2.30

$\underline{t=1}$

ε	N	$e_{L^\infty_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$	$e_{L^2_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$
8.00_{-3}	225	2.28_{-1}			3.84_{-1}		
2.00_{-3}	417	5.69_{-2}	1.00	-2.25	9.59_{-2}	1.00	-2.25
5.00_{-4}	801	1.42_{-2}	1.00	-2.12	2.40_{-2}	1.00	-2.12
1.25_{-4}	1,569	3.56_{-3}	1.00	-2.06	5.99_{-3}	1.00	-2.06
3.13_{-5}	3,109	8.90_{-4}	1.00	-2.03	1.50_{-3}	1.00	-2.03
7.81_{-6}	6,177	2.22_{-4}	1.00	-2.02	3.75_{-4}	1.00	-2.02
1.95_{-6}	12,325	5.54_{-5}	1.00	-2.01	9.39_{-5}	1.00	-2.00
4.88_{-7}	24,609	1.39_{-5}	1.00	-2.00	2.37_{-5}	0.99	-1.99

TABLE 2. Example 1, convergence for $d = 4$, initial data ($t = 0$) and terminal data ($t = 1$), after $K = 12$ time steps.

To summarize, for both types of errors we observe the behaviour

$$e_{L^\infty_{loc}},\ e_{L^2_{loc}} \sim C_d \cdot \varepsilon \qquad \text{and} \qquad e_{L^\infty_{loc}},\ e_{L^2_{loc}} \sim C'_d \cdot N^{-2}, \tag{27}$$

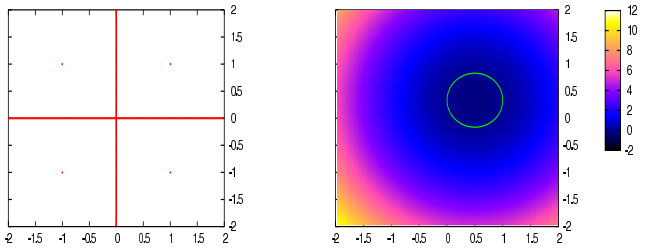
with C_d, C'_d constants independent of ε and N, respectively, but dependent on the dimension of the problem. This holds for both $t = 0$ and $t = 1$.

We observe the same type of behaviour for the 3-dimensional case (not shown here), as well as for the 4-dimensional case, see Table 2. Thus, (27) holds independent of the dimension. This is due to the specific structure of the initial data and the solution which is resolved by the adaptivity of our approach. All employed sparse grid points are on one of the coordinate axes, see Figure 5, therefore the adaptive sparse grid is just the superposition of d one-dimensional grids which explains (27) for all dimensions.

Moreover, we observe here that the number of sparse grid points for the terminal data ($t = 1$) is smaller than that for the initial data. This is at least in parts due to the evaluation outside of the domain in direction of the flow f in (20). There, instead of the exact solution v one needs to employ the discretized v_k, resulting in incoming data which is more easily representable by an adaptive sparse grid. We will not observe such a decrease of the number of necessary points in the general case, though.

Next, we consider the dependence of the complexity on the dimension d of the problem for a fixed threshold ε. The obtained results are presented in Table 3. We can observe that for time $t = 1$, the number N of points scales linearly with the dimension d of the problem, i.e. $N \sim d$. This almost perfect scaling can again be explained by the mentioned approximately additive structure of the solution which is detected by our adaptive sparse grid algorithm and which results in a pure axis-structure of the corresponding adaptive grid. Also the errors increase only slowly with the number d of dimensions by a factor of roughly $d/d-1$ per additional dimension, in other words, linearly in d. This shows that our adaptive procedure has very good scaling properties with respect to the dimension in simple cases.

Note here that the main bottleneck in runtime for higher dimensions is the error computation on the full grid and not the semi-Lagrangian sparse grid scheme itself. The method



(a) initial grid

(b) initial function with zero level set

(c) resulting grid

(d) resulting function with zero level set

FIGURE 5. Example 1 for $d = 2$ and $\varepsilon = 5 \cdot 10^{-4}$.

	$\underline{t=0}$				$\underline{t=1}$			
d	N	$e_{L^\infty_{loc}}$	$e_{L^2_{loc}}$	$\frac{N(d)}{N(d-1)}$	N	$e_{L^\infty_{loc}}$	$e_{L^2_{loc}}$	$\frac{N(d)}{N(d-1)}$
2	517	6.08_{-4}	7.50_{-4}		393	7.12_{-3}	1.20_{-2}	
3	781	9.05_{-4}	1.09_{-3}	1.51	595	1.07_{-2}	1.83_{-2}	1.51
4	1,049	1.20_{-3}	1.42_{-3}	1.34	801	1.42_{-2}	2.40_{-2}	1.35
5	1,321	1.49_{-3}	1.70_{-3}	1.26	1,011	1.78_{-2}	2.91_{-2}	1.26
6	1,597	1.77_{-3}	1.95_{-3}	1.21	1,255	2.14_{-2}	3.37_{-2}	1.24

TABLE 3. Example 1, scaling behaviour ($2 \leq d \leq 6$) for initial data ($t = 0$) and terminal data ($t = 1$), after $K = 12$ time steps, using $\varepsilon = 5 \cdot 10^{-4}$ for the adaptive procedure.

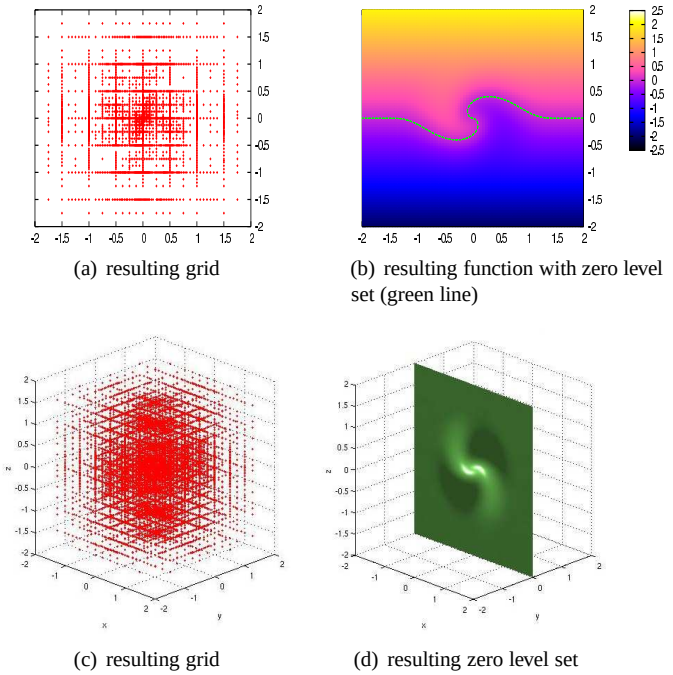
alone could actually be run in many more dimensions since its costs for the d-dimensional case involves only about d times the cost of the one dimensional case for this special test problem.

Remark 4.1. Notice that a spatially adaptive sparse grid representation of a function $f_1(x_1)$ that does not depend on the other variables $x_2, \ldots, x_d$ will have points located only on the x_1 axis. Similarly, a function of the form $f_1(x_1) + \cdots + f_d(x_d)$ will have a spatially adaptive sparse grid representation located only on the d principal axis. This explains the behavior in Figure 5, as well as for some initial data that will be used in Example 4 and 5 (see Figures 8 and 9, resp.).

Example 2. Next, we consider the following transport equation

$$\begin{aligned} v_t + f(\underline{x}) \cdot \nabla v = 0, \qquad & t \geq 0,\ \underline{x} \in \Omega, \\ v(0, \underline{x}) = \varphi(\underline{x}), \qquad & \underline{x} \in \Omega, \end{aligned}$$

where $\Omega = (-2, 2)^d$ and $t = 0.5$. In comparison to the first example the velocity field $f(\underline{x})$ is now $\underline{x}$-dependent and thus the solution has no longer an approximately additive structure. We will assume that $-f$ is an inward pointing flow, i.e. $(-f(\underline{x})) \cdot \nu \leq 0, \forall \underline{x} \in \partial\Omega$, where ν



(a) resulting grid

(b) resulting function with zero level set (green line)

(c) resulting grid

(d) resulting zero level set

FIGURE 6. Example 2 for $d = 2$ and $d = 3$, $\varepsilon = 5 \cdot 10^{-4}$.

is the outward normal on $\partial\Omega$, such that the backward characteristics of the flow stay inside the domain Ω.[3] Moreover, we choose the state dynamics as

$$f(\underline{x}) = a(\|x\|)\ f_1(\underline{x}),$$

where $a(r) := \left(\max\left(1 - \frac{r}{1.5}, 0\right)\right)^3$ is a scalar function, and $f_1(\underline{x})$ is a vector field corresponding to a rotation of $\mathbb{R}^d$, which is made precise in appendix A. The initial function is taken as $\varphi(\underline{x}) = x_2$. Since the characteristics $y_{\underline{x}}(-t)$ can be computed analytically (see appendix), the exact solution is known. It is given by

$$v(t, \underline{x}) = \varphi\left(y_{\underline{x}}(-t)\right).$$

For the numerical scheme, we also use the analytical formula for $y_{\underline{x}}(-\tau)$ where τ is the time step.

Note that the initial function $\varphi(x) = x_2$ can be represented exactly by a sparse grid. In Tables 4 and 5, we give the convergence results for dimension $d = 2$ and $d = 3$, respectively, observed at the terminal time $t = 0.5$. Again, we roughly obtain $\varrho_{\varepsilon,e} \sim 1$. Thus, also for this non-smooth function we have a control of the final error by a suitable adjustment of the refinement parameter ε. However, the number of grid points which are employed for a given refinement threshold is now much larger and the order with respect to N is worse than for the first example. This is due to the higher variability of the function (see Figure 6) which also exhibits larger second mixed derivatives. In higher dimensions this effect grows, the factor $\frac{N(d)}{N(d-1)}$ is about 5. We therefore need more and more grid points, but we are still able to achieve reasonable accuracies. In particular we observe no exponential scaling of the complexity with respect to d.

[3]Since $y_{\underline{x}}(-t)$ stays in Ω for all $\underline{x} \in \Omega$ the boundary needs no specific consideration and for this example we could use the standard hat basis of Figure 3(a) as well. Indeed we numerically observed the same accuracy with roughly the same number of points for the standard and modified basis of Figure 3, respectively.

$\underline{t = 0.5}$

ε	N	$e_{L^\infty_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$	$e_{L^2_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$
8.00_{-3}	281	2.44_{-1}			1.77_{-1}		
4.00_{-3}	539	1.33_{-1}	0.88	-0.93	9.61_{-2}	0.88	-0.94
2.00_{-3}	919	5.86_{-2}	1.18	-1.54	4.09_{-2}	1.23	-1.60
1.00_{-3}	1,409	3.20_{-2}	0.87	-1.41	1.98_{-2}	1.04	-1.69
5.00_{-4}	2,361	1.49_{-2}	1.10	-1.48	9.90_{-3}	1.00	-1.35
2.50_{-4}	3,683	6.03_{-3}	1.31	-2.04	4.41_{-3}	1.17	-1.82
1.25_{-4}	5,603	3.20_{-3}	0.92	-1.51	2.38_{-3}	0.89	-1.47
6.25_{-5}	8,531	1.69_{-3}	0.92	-1.52	1.28_{-3}	0.89	-1.47

TABLE 4. Example 2, convergence for $d = 2$, terminal data ($t = 0.5$), after $K = 10$ time steps.

$\underline{t = 0.5}$

ε	N	$e_{L^\infty_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$	$e_{L^2_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$
8.00_{-3}	1,297	3.75_{-1}			1.15_{-1}		
4.00_{-3}	2,507	2.26_{-1}	0.73	-0.76	6.09_{-2}	0.92	-0.97
2.00_{-3}	4,771	1.43_{-1}	0.67	-0.72	3.53_{-2}	0.79	-0.85
1.00_{-3}	8,451	7.01_{-2}	1.03	-1.24	1.79_{-2}	0.98	-1.18
5.00_{-4}	15,153	3.16_{-2}	1.15	-1.36	9.74_{-3}	0.88	-1.04
2.50_{-4}	25,213	1.63_{-2}	0.96	-1.31	5.27_{-3}	0.89	-1.21
1.25_{-4}	42,553	7.37_{-3}	1.14	-1.51	2.98_{-3}	0.82	-1.09
6.25_{-5}	72,267	4.06_{-3}	0.86	-1.12	1.71_{-3}	0.80	-1.05

TABLE 5. Example 2, convergence for $d = 3$, terminal data ($t = 0.5$), after $K = 10$ time steps.

$\underline{t = 0.5}$

d	N	$e_{L^\infty_{loc}}$	$e_{L^2_{loc}}$	$\frac{N(d)}{N(d-1)}$
2	2,361	1.49_{-2}	9.90_{-3}	
3	15,153	3.16_{-2}	9.74_{-3}	6.42
4	83,741	6.36_{-2}	7.72_{-3}	5.53
5	431,823	1.10_{-1}	7.84_{-3}	5.16

TABLE 6. Example 2, scaling behaviour ($2 \le d \le 5$) for terminal data ($t = 0.5$), after $K = 10$ time steps, using $\varepsilon = 5 \cdot 10^{-4}$ for the adaptive procedure.

Example 3. Now, we consider a nonlinear PDE example, the Eikonal equation

$$v_t + \|\nabla v\| = 0, \qquad t \ge 0,\ \underline{x} \in \Omega,$$
$$v(0, \underline{x}) = \varphi(\underline{x}), \qquad \underline{x} \in \Omega,$$

in $\Omega = (-2, 2)^d$. Note that this is a particular case of (1), since with $f(\underline{x}, \underline{b}) = \underline{b}$ for $\underline{b} \in \mathcal{B} := B(0, 1)$, i.e. the unit ball, we have $\max_{\underline{b} \in B(0,1)}(-f(\underline{x}, \underline{b}) \cdot \nabla v) = \|\nabla v\|$.

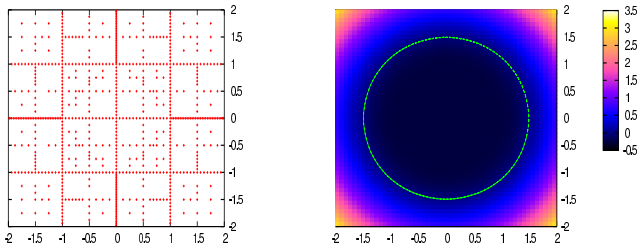
To obtain an exact solution to compare with, we consider the case $\varphi(\underline{x}) := q(\|\underline{x}\|)$ where $q : \mathbb{R}^+ \to \mathbb{R}$ is a non-decreasing function. Then, we can show that[4]

$$v(t, \underline{x}) = q((\|\underline{x}\| - t)_+). \tag{30}$$

Here, we use

$$q(z) := \frac{1}{2r_0}(z^2 - r_0^2)$$

[4]With $S_t(\underline{x}) := \{\underline{x} \in \mathbb{R}^d \mid \|\underline{x}\|_2 \le t\} = B(\underline{x}, t)$ we can show that $v(t, \underline{x}) = \inf_{y \in S_t(\underline{x})} \varphi(y) = \varphi(\underline{x}^*)$ where $\underline{x}^*$ is a point of $\mathbb{R}^d$ such that $\underline{x}^* := \arg\min_{\underline{y} \in S_t(\underline{x})} \|\underline{y}\|$.



(a) resulting grid

(b) resulting function with zero level set

FIGURE 7. Example 3 for $d = 2$, $\varepsilon = 2 \cdot 10^{-3}$.

$t = 1$

ε	N	$e_{L^\infty_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$	$e_{L^2_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$
8.00_{-3}	261	1.27_{-1}			1.30_{-1}		
2.00_{-3}	705	3.41_{-2}	0.95	-1.32	3.24_{-2}	1.00	-1.40
5.00_{-4}	1,889	1.12_{-2}	0.80	-1.13	1.02_{-2}	0.83	-1.17
1.25_{-4}	4,745	2.93_{-3}	0.97	-1.46	2.99_{-3}	0.88	-1.33
3.13_{-5}	11,153	8.75_{-4}	0.87	-1.41	7.83_{-4}	0.97	-1.57
7.81_{-6}	25,477	2.19_{-4}	1.00	-1.68	2.14_{-4}	0.93	-1.57
1.95_{-6}	57,641	6.24_{-5}	0.90	-1.54	6.30_{-5}	0.88	-1.50
4.88_{-7}	130,193	1.63_{-5}	0.97	-1.65	1.70_{-5}	0.94	-1.61

TABLE 7. Example 3, convergence for $d = 2$, terminal data ($t = 1$), after $K = 10$ time steps.

$t = 0$

d	N	$e_{L^\infty_{loc}}$	$e_{L^2_{loc}}$	$\frac{N(d)}{N(d-1)}$
2	261	2.44_{-3}	2.92_{-3}	
3	397	3.66_{-3}	4.45_{-3}	1.52
4	537	4.88_{-3}	5.81_{-3}	1.35
5	681	6.10_{-3}	6.98_{-3}	1.27

$t = 1$

d	N	$e_{L^\infty_{loc}}$	$e_{L^2_{loc}}$	$\frac{N(d)}{N(d-1)}$
2	705	3.41_{-2}	3.24_{-2}	
3	4,525	5.36_{-2}	3.95_{-2}	6.42
4	28,281	6.63_{-2}	4.73_{-2}	6.25
5	246,665	7.62_{-2}	4.48_{-2}	8.72

TABLE 8. Example 3, scaling behaviour ($2 \leq d \leq 5$) for initial data ($t = 0$) and terminal data ($t = 1$) after $K = 10$ time steps, using $\varepsilon = 2 \cdot 10^{-3}$ for the adaptive procedure.

with $r_0 = 0.5$, i.e. q is chosen such that $q(z) = 0$ for $z = r_0$ and $q'(r_0) = 1$. Hence, the zero level set $\{\underline{x} \,|\, \varphi(\underline{x}) = 0\}$ represents just the sphere of radius r_0.

In the numerical scheme we now employ the known optimal path $y_{\underline{x}}(-\tau) = \underline{x} - \tau \frac{\underline{x}}{\|\underline{x}\|}$ if $\|\underline{x}\| \geq \tau$, and $y_{\underline{x}}(-\tau) = 0$ otherwise. Thus, no error can come from a discretization of the continuous control and only interpolation errors from the SL scheme can occur.

Since we have the same initial function as in the first example, just moved to the center, we only give figures and results for $t = 1$ here. We observe a similar order $\varrho_{\varepsilon,e} \sim 1$ for $d = 2$ with respect to ε but we now need more grid points, i.e. we obtain $\varrho_{N,e} \sim -3/2$, see Table 7. The resulting grid and function for $t = 1$ are shown in Figure 7. Obviously, the solution at time $t = 1$ cannot be represented by an adaptive grid with points only on the axes like in Example 1. We now need to resolve the transition of the function from a flat region inside the zero level set to a steeper one outside. Furthermore, we have a non-smooth function due to the $(\cdot)_+$ in (30). This behaviour can however be handled well by

our adaptive discretization approach. Note in particular the sparse grid effect away from the area of non-smoothness in Figure 7.

The scaling behaviour is shown in Table 8. Although we now use a different value for ε, we observe for $t = 0$ the same factors $N(d)/N(d-1)$ as in Table 3 for the first example, where the same initial function was used. The scaling behaviour for $t = 1$ starts similar as in Example 2, but gets somewhat worse with rising d, which clearly shows that it depends on the function to be represented. But, again, we do not observe an exponential scaling with raising number of dimensions.

Example 4. In this example we consider the HJB equation

$$v_t + \max_{\underline{b}\in\mathcal{B}} \left(\sum_i b_i P_i \cdot \nabla v \right) = 0, \qquad t \geq 0,\ \underline{x} \in \Omega, \tag{31a}$$

$$v(0, \underline{x}) = \varphi(P^{-1}\underline{x}), \quad \underline{x} \in \Omega. \tag{31b}$$

where $d \geq 2$, $\Omega = (-2, 2)^d$, and $\mathcal{B} := \{\underline{b} = (b_1, \ldots, b_d),\ b_i = \pm 1\}$ is a set of 2^d column vectors, hence we have 2^d possible controls, and P_j denotes the j-th column vector of a given matrix $P \in \mathbb{R}^{d\times d}$. The definition of P and more details are given in appendix B. For the initial function, we consider radially symmetric data $\varphi(\underline{x}) := q(\|\underline{x}\|)$ as in Example 3. Hence, due to $P^2 = I_d$ and $v(0, \underline{x}) \equiv \varphi(P\underline{x})$, the zero level set at $t = 0$ is an ellipse in the first two dimensions, see Figure 8(a). The exact solution is given by

$$v(t, \underline{x}) \equiv \varphi\left((P\underline{x})^*_t\right), \tag{32}$$

see the appendix for details. Thus, the zero level set at time $t = 0.5$ describes a parallelogram with rounded edges in the first two dimensions, see Figure 8(d), whereas, in the other dimensions, it forms a rectangle with rounded edges. Note that we specifically designed this example to show the advantage of our adaptive sparse grid approach in situations where the function needs higher refinement in some dimensions and only small refinement in other dimensions.

For the numerical scheme we need to consider the explicit paths

$$y^{\underline{b}}_{\underline{x}}(-\tau) = \underline{x} - \tau(\sum_i b_i P_i) \equiv \underline{x} - \tau P\underline{b}$$

for all $\underline{b} \in \mathcal{B}$. Here, the first two components of $P\underline{b}$ take the four possible values $\pm(1, -2)$, $\pm(1, 0)$, and the other $d - 2$ values are given by $(\pm 1, \ldots, \pm 1)$. Let us emphasize that we now have to use an explicit minimization over all possible trajectories $y^{\underline{b}}_{\underline{x}}(-\tau)$ for the different actions $P\underline{b}$.

Since we are in a non-linear control setting, we have to reconsider the choice of the time step τ, which now needs to depend on ε. It is known from the approximation properties of regular grids that the error between the continuous solution and its discrete approximation depends on the time step τ and the spatial error ε_h in a mixed relation like $\mathcal{O}(\tau + \varepsilon_h/\tau)$, see [2, 9, 24]. In our adaptive sparse grid setting we do not have a mesh size h relating to the discretization error ε_h. Instead, we have a refinement constant ε which controls the error of our approximation. Therefore, to balance ε and τ accordingly, we choose here $\tau = c\sqrt{\varepsilon}$, with constant $c = 1$ in our case. This gives an overall error of $\mathcal{O}(\sqrt{\varepsilon} + \varepsilon/\sqrt{\varepsilon}) = \mathcal{O}(\sqrt{\varepsilon})$. A more detailed investigation of the relation between the error of the adaptive sparse grid discretization and the time step size is warranted, but beyond the scope of this paper.

In Figure 8 we show the initial and resulting grids as well as the level set of v for $t = 0$ and $t = 0.5$ for the two-dimensional case, and also the grids for the three-dimensional case. As can be seen in Figure 8(f) the resulting grid is essentially two-dimensional, since the rectangle-structure in the other dimensions can be represented by an adaptive grid which only has points on the axis in these coordinates. We also observe that a more diagonal structure of the function is disadvantageous for a sparse grid, since one needs a substantial number of points to represent such a diagonal formation by a locally refined sparse grid.

$t = 0$

ε	N	$e_{L^\infty_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$	$e_{L^2_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$
2.00_{-3}	263	3.66_{-3}			4.42_{-3}		
5.00_{-4}	519	9.03_{-4}	1.01	-2.06	1.10_{-3}	1.01	-2.05
1.25_{-4}	1,031	2.29_{-4}	0.99	-2.00	2.75_{-4}	1.00	-2.01
3.13_{-5}	2,055	5.72_{-5}	1.00	-2.01	6.84_{-5}	1.00	-2.02
7.81_{-6}	4,101	1.43_{-5}	1.00	-2.01	1.70_{-5}	1.01	-2.02
1.95_{-6}	8,199	3.57_{-6}	1.00	-2.00	4.16_{-6}	1.01	-2.03

$t = 0.5$

ε	K	N	$e_{L^\infty_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$	$e_{L^2_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$
2.00_{-3}	12	1,415	2.64_{-1}			2.31_{-1}		
5.00_{-4}	23	3,217	1.65_{-1}	0.34	-0.57	1.36_{-1}	0.38	-0.64
1.25_{-4}	45	7,529	9.10_{-2}	0.43	-0.70	7.46_{-2}	0.44	-0.71
3.13_{-5}	90	16,675	4.82_{-2}	0.46	-0.80	3.89_{-2}	0.47	-0.82
7.81_{-6}	179	36,967	2.53_{-2}	0.47	-0.81	2.01_{-2}	0.48	-0.83
1.95_{-6}	358	80,537	1.28_{-2}	0.49	-0.87	1.02_{-2}	0.49	-0.87

TABLE 9. Example 4, convergence for $d = 2$ for initial data ($t = 0$) and terminal data ($t = 0.5$) with $K = \frac{t}{\tau}$ for varying time step size $\tau = \sqrt{\varepsilon}$.

$t = 0$

ε	N	$e_{L^\infty_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$	$e_{L^2_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$
2.00_{-3}	399	4.88_{-3}			5.95_{-3}		
5.00_{-4}	783	1.19_{-3}	1.02	-2.10	1.45_{-3}	1.02	-2.09
1.25_{-4}	1,551	3.05_{-4}	0.98	-1.99	3.70_{-4}	0.99	-2.00
3.13_{-5}	3,087	7.62_{-5}	1.00	-2.01	9.24_{-5}	1.00	-2.02
7.81_{-6}	6,159	1.91_{-5}	1.00	-2.01	2.30_{-5}	1.00	-2.01
1.95_{-6}	12,303	4.76_{-6}	1.00	-2.01	5.49_{-6}	1.03	-2.07

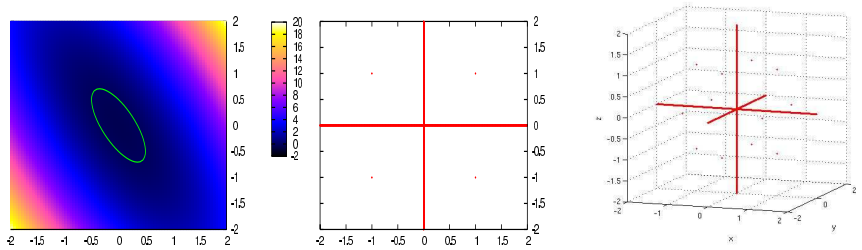
$t = 0.5$

ε	K	N	$e_{L^\infty_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$	$e_{L^2_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$
2.00_{-3}	12	1,523	2.59_{-1}			2.20_{-1}		
5.00_{-4}	23	3,425	1.63_{-1}	0.33	-0.57	1.27_{-1}	0.40	-0.68
1.25_{-4}	45	7,935	9.04_{-2}	0.43	-0.70	6.83_{-2}	0.45	-0.74
3.13_{-5}	90	17,467	4.81_{-2}	0.46	-0.80	3.54_{-2}	0.47	-0.83
7.81_{-6}	179	38,535	2.52_{-2}	0.47	-0.82	1.83_{-2}	0.48	-0.84
1.95_{-6}	358	83,653	1.28_{-2}	0.49	-0.87	9.26_{-3}	0.49	-0.88

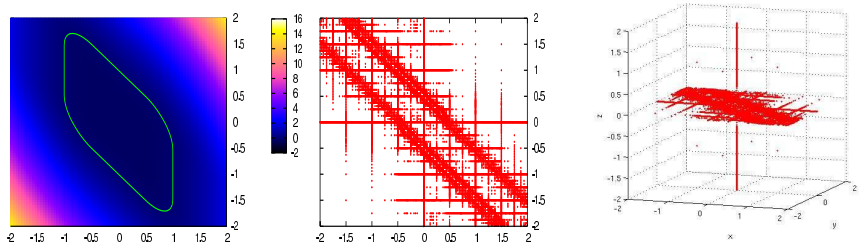
TABLE 10. Example 4, convergence for $d = 3$, initial data ($t = 0$) and terminal data ($t = 0.5$) with $K = \frac{t}{\tau}$ for varying time step size $\tau = \sqrt{\varepsilon}$.

d	N	$e_{L^\infty_{loc}}$	$e^{2d}_{L^\infty_{loc}}$	$\frac{N(d)}{N(d-1)}$
2	16,675	4.82_{-2}	4.82_{-2}	
3	17,467	4.81_{-2}	4.81_{-2}	1.05
4	18,263	4.79_{-2}	4.79_{-2}	1.05
5	19,063	4.78_{-2}	4.78_{-2}	1.04
6	19,867	–	4.76_{-2}	1.04

TABLE 11. Example 4, scaling behaviour for $2 \leq d \leq 6$ for terminal data ($t = 0.5$) with $\varepsilon = 3.125 \cdot 10^{-5}$ for the adaptive procedure and $K = 90$ time steps. For $d = 6$, the error is only measured in the $x_1 - x_2$ plane.



(a) initial function with zero level set for $d = 2$ (b) initial grid for $d = 2$ (c) initial grid for $d = 3$



(d) resulting function with zero level set for $d = 2$ (e) resulting grid for $d = 2$ (f) resulting grid for $d = 3$

FIGURE 8. Example 4 for $d = 2$ and $d = 3$, $\varepsilon = 3.125 \cdot 10^{-5}$.

Results for the case $d = 2$ and $d = 3$ are given in Tables 9 and 10, respectively. The initial grid shows a convergence rate as in the earlier examples, since the ellipsoidal structure has a form similar to the circle. As expected, the convergence order at $t = 0.5$ is now different due to the need for different time steps. We observe the predicted order of $\varrho_{\varepsilon,e} \sim 0.5$ for the refinement constant ε which controls our discretization error. For the number of grid points N we roughly observe an order of $\varrho_{N,e} \sim -0.8$.

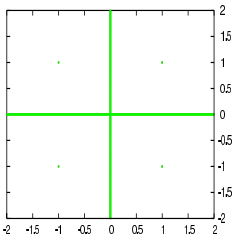
The scaling behaviour is displayed in Table 11, up to dimension $d = 6$. Note again that the error computation is the most time consuming part of the numerical procedure. Because it allows computationally cheap measurements in higher dimensions we additionally estimated the L^∞-error just for the $x_1 - x_2$ plane and denoted the result by $e^{2d}_{L^\infty_{loc}}$. The reduction to the $x_1 - x_2$ plane is justified in this example, as well as the following one, since the zero-level set in this plane reflects the dominant behaviour of the function at the final time, see Figures 8 and 9. Note that as long we could compute both errors $e_{L^\infty_{loc}}$ and $e^{2d}_{L^\infty_{loc}}$ they are here the same for the resulting function at $t = 0.5$. As can be expected from the picture of the grid in three dimensions, more dimensions only slightly increase the number of points since in the other dimensions only grid points on the axes are employed.

Example 5. At last, we consider the following problem with mixing fronts:

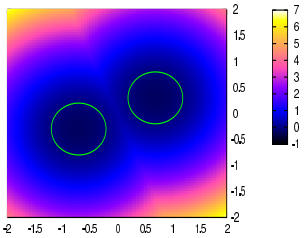
$$v_t + \max_{\underline{b} \in \mathcal{B}} \left(\sum_i b_i P_i \cdot \nabla v \right) = 0, \qquad t \geq 0,\ \underline{x} \in \Omega, \tag{33a}$$

$$v(0, \underline{x}) = \min(\varphi_1(P^{-1}\underline{x}), \varphi_2(P^{-1}\underline{x})), \quad \underline{x} \in \Omega, \tag{33b}$$

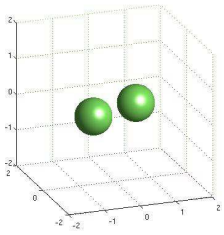
with $d \geq 2$, $\Omega = (-2, 2)^d$, $\varphi_1(x) := q(\|\underline{x} - \underline{r}\|)$ and $\varphi_2(x) := q(\|\underline{x} - \underline{s}\|)$, and q as in Example 3. Here, $\underline{r}, \underline{s}$ are the two points in $\mathbb{R}^d$ defined by $\underline{r} := (0.7, 0.3, 0, \ldots, 0)$ and $\underline{s} := -\underline{r}$. The matrix $P \in \mathbb{R}^{d \times d}$ is an orthogonal matrix, and as in Example 4, P_j denotes the j-th column vector of P and $\mathcal{B} = \{\underline{b} = (b_1, \ldots, b_d)^T,\ b_i = \pm 1\}$. The definition of P and more details are given in the appendix C.



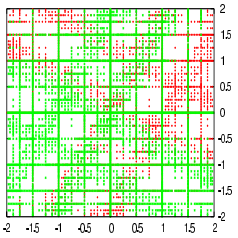
(a) overlay of both initial grids, $d = 2$



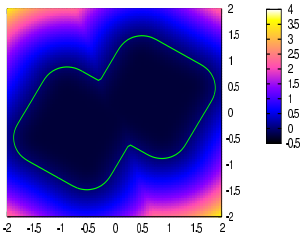
(b) initial function with zero level set for $d = 2$



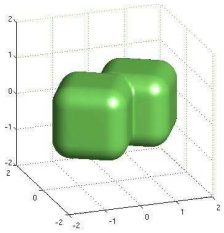
(c) initial zero level set for $d = 3$



(d) overlay of both resulting grids, $d = 2$



(e) resulting function with zero level set for $d = 2$



(f) resulting zero level set for $d = 3$

FIGURE 9. Example 5 for $d = 2$ and $d = 3$, $\alpha = \frac{\pi}{6}$, $\varepsilon = 5 \cdot 10^{-4}$, at time $t = 0$ in figures (a), (b), (c), and time $t = 0.5$ in figures (d), (e) and (f).

The exact solution here is

$$v(t, \underline{x}) = \min\left(\varphi\left((P^T(\underline{x} - \underline{r}))_t^*\right), \varphi\left((P^T(\underline{x} - \underline{s}))_t^*\right)\right).$$

In the first two dimensions the zero level set at time $t = 0.5$ describes the intersection of two rotated squares with rounded edges, in the other dimensions the level set forms a square with rounded edges, see Figure 9.

In this example, the initial data and the solution correspond to the mixing of two fronts and therefore are much less regular than in the previous examples. We numerically observed that the direct application of our adaptive SL-SG scheme from Section 3.1 is not efficient for dimensions $d \geq 3$ in this case. Hence, we now apply the generalized SL-SG scheme as described in Section 3.2 which is better suited for data of the form (33b).

For the generalized scheme, we employ two separate grids. For each initial data $\varphi_i, i = 1, 2$, we apply the SL-SG scheme and use the explicit paths $y_{\underline{x}}^{\underline{b}}(-\tau) = \underline{x} - \tau(\sum_i b_i P_i) \equiv \underline{x} - \tau P\underline{b}$ for all $\underline{b} \in \mathcal{B}$. Then, the minimum of both resulting functions is taken to obtain the solution at time t. The advantage of this approach is that the (costly) direct representation due to the discontinuity of the gradient of the $\min$-function is avoided.

We give results for the two- and three-dimensional case in Table 12 and Table 13, respectively. Here we only show the results for the terminal time, the results for $t = 0$ are equal to the ones from Example 1, but with twice the number of points. As before, we observe the predicted order of $\varrho_{\varepsilon,e} \sim 0.5$ for the convergence with respect to ε and roughly $\varrho_{N,e} \sim -0.8$ with respect to N.

The scaling behaviour is analyzed in Table 14, this time in up to 8 dimensions. We observe that the error, the complexity, and the scaling behaviour behave quite well and are similar to Example 4.

$t = 0.5$

ε	K	N	$e_{L^\infty_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$	$e_{L^2_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$
2.00_{-3}	12	2,192	9.43_{-2}			5.75_{-2}		
5.00_{-4}	23	5,560	3.44_{-2}	0.73	-1.08	2.52_{-2}	0.60	-0.89
1.25_{-4}	45	13,132	1.83_{-2}	0.45	-0.73	1.26_{-2}	0.50	-0.81
3.13_{-5}	90	31,046	8.89_{-3}	0.52	-0.84	6.44_{-3}	0.48	-0.78
7.81_{-6}	179	71,874	4.24_{-3}	0.53	-0.88	3.22_{-3}	0.50	-0.82

TABLE 12. Example 5, convergence for $d = 2$, terminal data ($t = 0.5$) with $K = \frac{t}{\tau}$ for varying time step size $\tau = \sqrt{\varepsilon}$.

$t = 0.5$

ε	K	N	$e_{L^\infty_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$	$e_{L^2_{loc}}$	$\varrho_{\varepsilon,e}$	$\varrho_{N,e}$
2.00_{-3}	12	2,408	8.91_{-2}			6.17_{-2}		
5.00_{-4}	23	5,976	4.13_{-2}	0.55	-0.85	2.66_{-2}	0.61	-0.92
1.25_{-4}	45	13,944	2.18_{-2}	0.46	-0.76	1.33_{-2}	0.50	-0.82
3.13_{-5}	90	32,630	1.06_{-2}	0.52	-0.85	6.66_{-3}	0.50	-0.81
7.81_{-6}	179	75,010	5.08_{-3}	0.53	-0.88	3.37_{-3}	0.49	-0.82

TABLE 13. Example 5, convergence for $d = 3$ for terminal data ($t = 0.5$) with $K = \frac{t}{\tau}$ for varying time step size $\tau = \sqrt{\varepsilon}$.

d	N	$e_{L^\infty_{loc}}$	$e^{2d}_{L^\infty_{loc}}$	$\frac{N(d)}{N(d-1)}$
2	5,560	3.44_{-2}	3.44_{-2}	
3	5,976	4.13_{-2}	3.61_{-2}	1.07
4	6,400	4.83_{-2}	3.78_{-2}	1.07
5	6,832	5.53_{-2}	3.95_{-2}	1.07
6	7,272	$-$	4.12_{-2}	1.06
7	7,720	$-$	4.29_{-2}	1.06
8	8,176	$-$	4.46_{-2}	1.06

TABLE 14. Example 5, scaling analysis for $2 \leq d \leq 8$ for terminal data ($t = 0.5$) with $\varepsilon = 5 \cdot 10^{-4}$ for the adaptive procedure and $K = 23$ time steps. For $d \geq 6$, for the same reasons as in Example 4, the error is only measured in the $x_1 - x_2$ plane.

5. CONCLUSION

We presented and implemented a new spatially adaptive semi-Lagrangian sparse grid scheme and tested it on a series of linear and nonlinear time-dependent Hamilton-Jacobi Bellman equations. In particular, we focused on the zero level set of the solution. The adaptive sparse grid is able to handle the representation of the front with reasonable precision in higher dimensions. This was tested up to $d = 8$ dimension in this work. From the numerical point of view, two main ingredients are crucial for our new approach: First, the adaptivity of the employed sparse grid, and, second, a special boundary treatment allow to keep the number of necessary sparse grids points relatively small. Furthermore we should emphasize that the sparse grid effect only works properly if the data has some potentially lower-dimensional structure or is roughly axis-aligned, e.g. after a suitable transformation.

As noted, the main computational burden in our SL-SG scheme is the evaluation of the adaptive sparse grid function. It was shown recently [17] that using a specific reordering of the steps of the point evaluations together with a GPU-based parallelisation one can achieve speed-ups of almost 50 in comparison to the standard implementation of adaptive

sparse grids. This approach can also be employed in our scheme and could be used to improve the runtime significantly.

Note that, for the moment, the proposed scheme neither has the monotony property that would give convergence towards the viscosity solution, nor has provable stability as far as we know. However, we think that this initial work is an encouraging step towards the construction of related schemes for the solution of HJB equations in higher dimensions that could remedy these drawbacks.

Let us also mention that extensions of the proposed scheme to more general situations are possible. For instance a sparse grid SL scheme can be straightforwardly defined for HJB equations with an additional cost term.

APPENDIX A. DETAILS FOR EXAMPLE 2

In order to define the vector field $f_1(\underline{x})$, let us first define two vectors $\underline{u}, \underline{v}$ of $\mathbb{R}^d$ as follows:

$$\underline{u} = (1, 0, \dots, 0)^T \quad \text{and} \quad \underline{v} = (0, 1, \dots, 1)^T/\sqrt{d-1}.$$

We then denote by A the operator such that $A(\alpha, \beta) = (-\beta, \alpha)$ in the basis $(\underline{u}, \underline{v})$, that is, $A(\alpha\underline{u} + \beta\underline{v}) = -\beta\underline{u} + \alpha\underline{v}$. We now decompose a vector $\underline{x}$ as $\underline{x} = P\underline{x} + (\underline{x} - P\underline{x})$, where $P\underline{x} = (\underline{x}, \underline{u})\underline{u} + (\underline{x}, \underline{v})\underline{v}$ is the projection on the plane $\mathrm{Vect}(\underline{u}, \underline{v})$. We have also

$$P\underline{x} = (x_1, y, \dots, y)^T, \quad \text{where } y = \frac{\sum_{i=2}^d x_i}{d-1}.$$

We can now define $f_1(\underline{x})$ in the following way:

$$f_1(\underline{x}) = A(P\underline{x}) := -(\underline{x}, \underline{v})\underline{u} + (\underline{x}, \underline{u})\underline{v}.$$

We then have

$$y_{\underline{x}}(-t) := R_{-ta(\|\underline{x}\|)}(P\underline{x}) + \underline{x} - P\underline{x},$$

where the operator R_θ is represented by the following rotation matrix in the basis $(\underline{u}, \underline{v})$

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}.$$

Hence, to compute $y_{\underline{x}}(-t)$ we have, using the notation $\theta = -ta(\|\underline{x}\|)$,

$$\begin{aligned} R_\theta P\underline{x} &= R_\theta\Big((\underline{x}, \underline{u})\underline{u} + (\underline{x}, \underline{v})\underline{v}\Big) \\ &= \Big(\cos\theta(\underline{x}, \underline{u}) - \sin\theta(\underline{x}, \underline{v})\Big)\underline{u} + \Big(\sin\theta(\underline{x}, \underline{u}) + \cos\theta(\underline{x}, \underline{v})\Big)\underline{v} \\ &= \Big(\cos\theta x_1 - \sin\theta(\underline{x}, \underline{v})\Big)\underline{u} + \Big(\sin\theta x_1 + \cos\theta(\underline{x}, \underline{v})\Big)\underline{v} \end{aligned}$$

and thus

$$y_{\underline{x}}(-t) = \underline{x} + \Big((\cos\theta - 1)x_1 - \sin\theta(\underline{x}, \underline{v})\Big)\underline{u} + \Big(\sin\theta x_1 + (\cos\theta - 1)(\underline{x}, \underline{v})\Big)\underline{v}$$

where $(\underline{x}, \underline{u}) = x_1$ and $(\underline{x}, \underline{v}) = \frac{1}{\sqrt{d-1}}\sum_{i\geq 2} x_i$.

APPENDIX B. DETAILS FOR EXAMPLE 4

We consider the equation

$$v_t + |v_{x_1} - v_{x_2}| + |-v_{x_2}| + \sum_{i=3,\dots,d} |v_{x_i}| = 0, \qquad t \geq 0, \quad \underline{x} \in \Omega, \tag{34a}$$

$$v(0, \underline{x}) = v_0(\underline{x}) = \varphi(P^{-1}\underline{x}), \qquad \underline{x} \in \Omega. \tag{34b}$$

where $d \geq 2$, $\Omega = (-2,2)^d$, and P is the matrix defined by

$$P := \begin{bmatrix} 1 & 0 & 0 & \dots & \dots & 0 \\ -1 & -1 & 0 & \dots & \dots & 0 \\ 0 & 0 & 1 & & & \\ \vdots & & 0 & \ddots & & \\ \vdots & & \vdots & & \ddots & \\ 0 & \dots & 0 & & & 1 \end{bmatrix}. \tag{35}$$

We denote by P_j the j-th column vector of P, i.e. $P = [P_1, \dots, P_d]$ and $\varphi(\underline{x}) := q(\|\underline{x}\|)$ as in Example 3. Then equation (34a) is equivalent to the HJB equation (31a). To obtain the exact solution for this problem, let $\underline{\xi}$ be the new coordinate vectors of $\mathbb{R}^d$ in the basis $(P_1, \dots, P_d)$ such that $\underline{x} = P\underline{\xi}$, and let $u(t,\underline{\xi}) := v(t,\underline{x}) = v(t,P\underline{\xi})$. Then we have $\partial_{\xi_i} u = \nabla_{\underline{x}} v \cdot (\partial_{\xi_i} \underline{x}) = \nabla_{\underline{x}} v \cdot P_i$. Hence, u is a solution of $u_t + \sum_{i=1,\dots,d} |u_{\xi_i}| = 0$ for $t \geq 0$ and $\underline{x} \in \mathbb{R}^d$, and $u(0,\underline{\xi}) = \varphi(\underline{\xi})$. The solution is given by $u(t,\underline{\xi}) = \min_{\underline{y} \in S_t(\underline{\xi})} \varphi(\underline{y})$, where $S_t(\underline{\xi}) := \{\underline{y},\ \|\underline{y} - \underline{\xi}\|_\infty \leq t\}$. Therefore, $u(t,\underline{\xi}) = \varphi(\underline{\xi}_t^*)$ where $\underline{\xi}_t^* := \operatorname{argmin}\{d(0,\underline{y}), \underline{y} \in S_t(\underline{\xi})\}$ is the orthogonal projection of 0 on the convex set $S_t(\underline{\xi})$. Since $S_t(\underline{\xi})$ is the d-dimensional box $\prod_{i=1,\dots,d}[\xi_i - t, \xi_i + t]$, the projection $\underline{\xi}_t^* = (\xi_{t,i}^*)$ can be computed component-wise and we have $\xi_{t,i}^* = \min(\max(0, \xi_i - t), \xi_i + t)$. This formula will now be denoted by

$$\underline{\xi}_t^* = \min(\max(0, \underline{\xi} - t), \underline{\xi} + t).$$

Finally we obtain the exact solution

$$v(t,\underline{x}) = u(t,\underline{\xi}) = \varphi(\underline{\xi}_t^*) = \varphi\left((P^{-1}\underline{x})_t^*\right) \equiv \varphi\left((P\underline{x})_t^*\right). \tag{36}$$

APPENDIX C. DETAILS FOR EXAMPLE 5

We consider

$$v_t + |\cos\alpha \cdot v_{x_1} + \sin\alpha \cdot v_{x_2}| + |-\sin\alpha \cdot v_{x_1} + \cos\alpha \cdot v_{x_2}| + \sum_{i=3,\dots,d} |v_{x_i}| = 0, \qquad t \geq 0,\ \underline{x} \in \Omega, \tag{37a}$$

$$v(0,\underline{x}) = \min(\varphi_1(P^{-1}\underline{x}), \varphi_2(P^{-1}\underline{x})), \qquad \underline{x} \in \Omega, \tag{37b}$$

where $d \geq 2$, $\Omega = (-2,2)^d$, and in this example P is the matrix defined by

$$P := \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & \dots & \dots & 0 \\ \sin\alpha & \cos\alpha & 0 & \dots & \dots & 0 \\ 0 & 0 & 1 & & & \\ \vdots & & 0 & \ddots & & \\ \vdots & & & \ddots & \ddots & \\ 0 & \dots & & & 0 & 1 \end{bmatrix}. \tag{38}$$

Let again P_j denote the j-th column vector of P and $\mathcal{B} = \{\underline{b} = (b_1, \dots, b_d)^T,\ b_i = \pm 1\}$. Then, equation (37a) is equivalent to the HJB equation (33a). Note that since $\varphi(x) := q(\|x\|)$ and P orthogonal, and with $\varphi_1(\underline{x}) := \varphi(\underline{x} - \underline{r})$ and $\varphi_2(\underline{x}) := \varphi(\underline{x} - \underline{s})$, we have $v(0,\underline{x}) \equiv \min\left(\varphi_1(\underline{x}), \varphi_2(\underline{x})\right)$ here. The exact solution for each initial data $\varphi_i, i = 1,2$ is obtained as in the previous example, and therefore

$$v(t,\underline{x}) = \min\left(\varphi\left((P^{-1}(\underline{x} - \underline{r}))_t^*\right), \varphi\left((P^{-1}(\underline{x} - \underline{s}))_t^*\right)\right) \quad \left(\text{with } P^{-1} = P^T\right)$$

holds.

Acknowledgments. This work was partially supported by the EU under the 7th Framework Programme Marie Curie Initial Training Network "FP7-PEOPLE-2010-ITN", SADCO project, GA number 264735-SADCO, and by the DFG priority programme SPP 1324 "Mathematical methods for extracting quantifiable information from complex systems"; Michael Griebel was also partially supported by the Hausdorff Research Institute for Mathematics in Bonn and by the Sonderforschungsbereich 611 *Singular phenomena and scaling in mathematical models* funded by the Deutsche Forschungsgemeinschaft.

REFERENCES

[1] M. Akian, S. Gaubert, and A. Lakhoua. The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis. *SIAM J. Control Optim.*, 47(2):817–848, 2008.

[2] M. Bardi and I. Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Systems and Control: Foundations and Applications. Birkhäuser, Boston, 1997.

[3] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

[4] E. Carlini, M. Falcone, and R. Ferretti. An efficient algorithm for Hamilton–Jacobi equations in high dimensions. *Comput. Vis. Sci.*, 7:15–29, 2004.

[5] E. Carlini, R. Ferretti, and G. Russo. A weighted essentialy non oscillatory, large time-step scheme for Hamilton Jacobi equations. *SIAM J. Sci. Comp.*, 27(3):1071–1091, 2005.

[6] M. Crandall and P.-L. Lions. Two approximations of solutions of Hamilton Jacobi equations. *Mathematics of Computation*, 43:1–19, 1984.

[7] K. Debrabant and E. Jakobsen. Semi-Lagrangian approximation schemes for linear and Hamilton-Jacobi equations. *Preprint*.

[8] G. Faber. Über stetige Funktionen. *Mathematische Annalen*, 66:81–94, 1909.

[9] M. Falcone. A numerical approach to the infinite horizon problem of deterministic control theory. *Applied Mathematics & Optimization*, 15:1–13, 1987. See also Corrigenda: A numerical approach to the infinite horizon problem of deterministic control theory. Applied Mathematics & Optimization, 23:213–214, 1991.

[10] M. Falcone and R. Ferretti. *Semi-Lagrangian approximation schemes for linear and Hamilton-Jacobi equations*. SIAM, to appear.

[11] M. Falcone, T. Giorgi, and P. Loreti. Level sets of viscosity solutions: Some applications to fronts and rendez-vous problems. *SIAM J. Appl. Math.*, 54(5):1335–1354, 1994.

[12] C. Feuersänger. *Sparse Grid Methods for Higher Dimensional Approximation*. Dissertation, Institut für Numerische Simulation, Universität Bonn, 2010.

[13] E. Godlewski and P.-A. Raviart. *Hyperbolic Systems of Conservation Laws*. SMAI. Ellipses, 1991.

[14] M. Griebel. A parallelizable and vectorizable multi-level algorithm on sparse grids. In W. Hackbusch, editor, *Parallel Algorithms for partial differential equations*, *Notes on Numerical Fluid Mechanics*, volume 31, pages 94–100. Vieweg Verlag, Braunschweig, 1991.

[15] M. Griebel. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. *Computing*, 61(2):151–179, 1998.

[16] L. Grüne. An adaptive grid scheme for the discrete Hamilton–Jacobi–Bellman equation. *Numer. Math.*, 75(3):319–337, 1997.

[17] A. Heinecke and D. Pflüger. Multi- and many-core data mining with adaptive sparse grids. In *Proceedings of the 8th ACM International Conference on Computing Frontiers*, CF '11, pages 29:1–29:10, New York, NY, USA, 2011. ACM.

[18] C. Hu and S.-W. Shu. A discontinuous Galerkin finite element method for Hamilton-Jacobi equations. *SIAM J. Sci. Comput.*, 21:666–690, 1999.

[19] V. P. Maslov. *Operational methods*. Mir Publishers, Moscow, 1976. Translated from the Russian by V. Golo, N. Kulman and G. Voropaeva.

[20] I. Mitchell, A. Bayen, and C. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Trans. Automat. Control*, 50(7):947–957, 2005.

[21] I. Mitchell and C. Tomlin. Level set methods for computation in hybrid systems. In B. Krogh and E. N. Lynch, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 310–323. New York: Springer-Verlag, 2000.

[22] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79:12–49, 1988.

[23] S. Osher and C.-W. Shu. High order essentially non-oscillatory schemes for Hamilton-Jacobi equations. *SIAM J. Numer. Anal.*, 28:907–922, 1991.

[24] S. Pareigis. Adaptive choice of grid and time in reinforcement learning. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *NIPS*, 1997.

[25] D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. Verlag Dr. Hut, München, Aug. 2010. Dissertation TU München.

[26] S. A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Dokl. Akad. Nauk SSSR*, 148:1042–1043, 1963. Russian, Engl. Transl.: Soviet Math. Dokl. 4:240–243, 1963.

[27] H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49:379–412, 1986.

[28] C. Zenger. Sparse grids. In W. Hackbusch, editor, *Parallel Algorithms for Partial Differential Equations, Proceedings of the Sixth GAMM-Seminar, Kiel, 1990*, volume 31 of *Notes on Num. Fluid Mech.*, pages 241–251. Vieweg-Verlag, 1991.

[29] Y.-T. Zhang and C.-W. Shu. High order WENO schemes for Hamilton-Jacobi equations on triangular meshes. *SIAM J. Sci. Comput.*, 24:1005–1030, 2003.

LAB. JACQUES-LOUIS LIONS, UNIV. PIERRE ET MARIE CURIE 75252 PARIS CEDEX 05 FRANCE AND UFR DE MATHÉMATIQUES, SITE CHEVALERET, UNIV. PARIS-DIDEROT, 75205 PARIS CEDEX AND PROJET COMMANDS, ENSTA PARISTECH, 828, BOULEVARD DES MARCHAUX, 91762 PALAISEAU CEDEX
E-mail address: boka@math.jussieu.fr

INSTITUR FÜR NUMERISCHE SIMULATION, UNIVERSTITÄT BONN AND FRAUNHOFER SCAI
E-mail address: garcke@ins.uni-bonn.de

INSTITUR FÜR NUMERISCHE SIMULATION, UNIVERSTITÄT BONN AND FRAUNHOFER SCAI
E-mail address: griebel@ins.uni-bonn.de

INSTITUT FÜR MATHEMATIK, TECHNISCHE UNIVERSITÄT BERLIN
E-mail address: klompmak@math.tu-berlin.de