ON THE NUMERICAL SYNTHESIS OF OPTIMAL FEEDBACK CONTROLLERS

Jorge Estrela da Silva
Electrical Engineering Department
Institute of Engineering of Porto
Portugal
jes@isep.ipp.pt

João Borges de Sousa
Electrical and Computer Engineering Department
Faculty of Engineering of Porto University
Portugal
jtasso@fe.up.pt

Abstract

We investigate the performance of feedback control laws obtained via dynamic programming techniques for deterministic continuous-time systems with continuous state-space $\mathbb{R}^n$. The control law relies on a value function which is calculated numerically with a semi-Lagrangian (SL) scheme. Both the numerical solver and the control law require interpolation. We propose a new algorithm that minimizes the impact of the discontinuities introduced by the state constraints for finite and infinite horizon optimal control problems.

Key words

Dynamic programming, feedback control, numerical methods.

1 Introduction

The design of optimal feedback controllers for continuous-time systems with continuous state-space is not a trivial task. This is mostly because, in general, it is not possible to find the appropriate solution in closed form, either by using the Pontryagin's maximum principle (PMP) (Pontryagin *et al.*, 1962) or dynamic programming (DP) (Bellman, 1957). The alternative is to use numerical methods. However, numerical methods pose their own problems and, most important, the accuracy of the employed method impacts the level of (sub-)optimality of the computed control law.

This paper focuses on the DP approach. The DP approach is based on the concept of the value function. The value function gives the optimal cost of the associated optimal control problem as a function of the initial state. The main appeal of the dynamic programming approach resides in the fact that the optimal control can be easily computed from the value function, by application of the DP (or, more specifically, the Verification Theorem). The crux of the problem resides in the computation of the value function.

The numerical computation of the value function is a very demanding task, specially as the system dimension grows. However, we remark that the value function must be computed only once for each optimal control problem and that such computation can be made at the design stage (i.e., offline), when more processing power is generally available. On the other hand, given the value function, the determination of the optimal control is a very amenable task for on-line computation. This is in contrast with receding horizon approaches, that may have to perform very demanding optimizations on each control cycle.

For continuous-time systems, the DP approach leads to the Hamilton-Jacobi-Bellman partial differential equation (HJB PDE), if a single control input is considered, or to the Hamilton-Jacobi-Isaacs partial differential equation (HJI PDE), if two adversarial inputs are considered (differential games framework (Krasovskii and Subbotin, 1988)). The value function is the viscosity solution of the HJB(I) PDE (see (Bardi and Capuzzo-Dolcetta, 1997) or (Fleming and Soner, 2006)). This fact is appealing given the large body of research on numerical schemes for PDE's.

Numerical methods for the solution of dynamic programming equations can be characterized as Lagrangian (based on following the system trajectories), Eulerian (computations are performed at fixed grid nodes) and semi-Lagrangian (a combination of the Eulerian and Lagrangian approaches). In general, these methods must perform interpolations of the value function or numerical approximations of the gradient of the value function (e.g., see (Mitchell *et al.*, 2005), (Cristiani and Falcone, 2009) and reference therein). Higher order methods have the potential to produce more accurate solutions. However, only those schemes that do not introduce oscillations in the solution should be used. Such oscillations could induce spurious stable equilibrium points or limit-cycles in the closed loop system preventing it from following the optimal trajectory. First order schemes do not introduce oscillations. However, their main drawback is the lower accuracy.

The main contribution of this paper is a new algorithm for state constrained optimal control problems

over a given time horizon. We extend a typical semi-Lagrangian scheme (Cristiani and Falcone, 2009) with ideas from receding horizon control. The main advantage of this algorithm is the reduction of diffusive effects near the boundary of the constrained region of operation (more specifically, near the boundary of the maximal controlled invariant set (Blanchini and Miani, 2008)).

In section 2 we present some background on dynamic programming. In section 3 we discuss some limitations of typical semi-Lagrangian algorithms. In section 4 we describe the proposed algorithm. The conclusions and final remarks are presented on section 5.

2 Dynamic programming based controller synthesis

Consider the cost functional

$$J(t_0, x_0, a) = \Psi(x(t_f)) + \int_{t_0}^{t_f} L(y(x_0, \tau, a), a(\tau))d\tau \quad (1)$$

and the optimal control problem (OCP)

$$\min_a J(t_0, x_0, a) \quad (2)$$

subject to:

$$\dot{x}(t) = f(x(t), a(t)) \quad (3)$$
$$x(t_0) = x_0 \quad (4)$$
$$x(t) \in \mathbb{X} \quad (5)$$

where $y(x_0, t, a)$ is the state of the system at time t when subject to the input sequence $a(.)$ and initial state $x(t_0) = x_0$; $L(x, a)$ and $\Psi(x)$ are the running and terminal costs, respectively; (3) describes the system dynamics, i.e., the system flow at state $x(t)$ when subject to input $a(t)$; $a(.)$ is drawn from $\mathcal{U}_a$, the space of measurable input sequences such that $a(t) \in U_a$; $\mathbb{X}$ defines the state constraints.

Consider also the following variation of the OCP presented above where

$$t_f = \min\{t : x(t) \in \mathcal{T}\} \quad (6)$$

and $\mathcal{T}$ is a given target set. This version of the OCP has a static solution, in the sense that it not depends on t_0.

The dynamic programming framework defines the concept of value function. The value function associated to the OCP composed by (2)–(5) is defined as follows:

$$V(t, x) = \min_{a \in \mathcal{U}_a} J(t, x, a) \quad (7)$$

For the static OCP the value function is independent of t and simply designated by $V(x)$.

The dynamic programming principle (DPP), also known as principle of optimality, for deterministic continuous-time system is expressed as follows:

$$V(t, x) = \inf_{a \in \mathcal{U}_a} \Big\{ \int_0^{\Delta} L(y(x, \tau, a), a(\tau))d\tau + V(t + \Delta, y(x, \Delta, a)) \Big\}, x \notin \mathcal{T}, t < t_f \quad (8)$$

where $y(x, t, a)$ is the state of the system at time t when subject to the input signal a and initial state $x(0) = x$. For the time-dependent case we define the terminal condition

$$V(t_f, x) = \Psi(x) \quad (9)$$

while for the static case we define the boundary condition

$$V(x) = \Psi(x), x \in \mathcal{T} \quad (10)$$

Consider the discrete-time version of the DPP:

$$V(t, x) = \min_{a \in U_a} \Big\{ \int_0^{\Delta} L(y_\Delta(x, \tau, a), a)d\tau + V(t + \Delta, y_\Delta(x, \Delta, a)) \Big\}, x \notin \mathcal{T}, t < t_f \quad (11)$$

where $y_\Delta(x, t, a) = y(x, t, [0, \Delta] \times a)$, i.e., the system trajectory for a constant input a during a period Δ. The solution of the discrete time dynamic programming problem converges to the solution of the continuous-time problem as Δ approaches zero (see (Fleming and Soner, 2006, Theorem 8.1)).

For static (time-invariant) problems, the discrete-time DPP can be simply cast as

$$V(x) = \min_{a \in U_a} \Big\{ \int_0^{\Delta} L(y_\Delta(x, \tau, a), a)d\tau + V(y_\Delta(x, \Delta, a)) \Big\}, x \notin \mathcal{T}, t < t_f \quad (12)$$

One of the main appeals of the dynamic programming approach is that, given a value function $V(x)$, the optimal control can be determined in state feedback form $f_c(x)$. For the discrete-time case, this is given by:

$$f_c(x) = \text{argmin}_{a \in U_a} \Big\{ \int_0^{\Delta} L(y_\Delta(x, \tau, a), a)d\tau + V(y_\Delta(x, \Delta, a)) \Big\} \quad (13)$$

The same idea can be applied to the time-dependent value function. However, the dependence on the time

variable is undesirable if the objective is to optimize the performance of the system for a large and undefined period of time or, ultimately, for the infinite horizon case. Since, in general, numerical solutions are employed, the extra dimension incurred by the time variable would lead to increased storage requirements. In what concerns the infinite horizon optimal control problem, we remark that, under certain conditions (see, e.g., (Bardi and Capuzzo-Dolcetta, 1997) and (Mitake, 2008)), the solution $V(t,x)$ converges to $V(x) - ct$ as $t \to -\infty$, where c (the ergodic cost) is related to the average cost of the large-time behaviour of the optimal trajectories. One such example is the well known Linear Quadratic Regulator (LQR) problem. For a stabilizable system, the standard LQR problem leads to optimal trajectories converging to the origin; in this case, the ergodic cost is $c = 0$, since we are assuming the usual zero running cost at the origin.

3 Standard semi-Lagrangian scheme

A semi-Lagrangian numerical solver was used for the computation of the value function. The semi-Lagrangian scheme implements a space discretization of the discrete-time DPP. We based our implementation on the description of (Cristiani and Falcone, 2009). In the present case, the value function is computed at the nodes of a regular grid covering the desired region $\mathbb{X}$ of the state space. We denote by Ω the set of grid nodes. A dimension independent algorithm was implemented as a C++ class. This algorithm was already used to solve problems of dimension 3 and 4. Here, for the sake of clarity, we will limit our numerical experiments to two-dimensional systems.

The user must write the code corresponding to $f(x,a,b)$ and $L(x,a,b)$. The computational space (limits and number of nodes along each dimension), the initial and boundary conditions must also be defined for each problem. Finally, the user must define the input range as a finite discrete set of values.

The main procedure consists of the application of the DPP (11) to every grid node. This procedure returns an approximation of the value function and corresponding optimal input at every grid node. For each considered input value a, the procedure computes $\left\{ \int_0^\Delta L(y_\Delta(x,\tau,a),a)d\tau + V(y_\Delta(x,\Delta,a)) \right\}$ and then chooses the minimum. This entails two main sub-procedures:

1. The numerical solution of the augmented set or ordinary differential equations $(\dot{x},\dot{c}) = (f(x,a), L(x,a))$ in order to obtain $y_\Delta(x,\Delta,a)$ and the cost to go from x to $y_\Delta(x,\Delta,a)$;
2. Interpolation of $V(x)$ in order to compute $V(y_\Delta(x,\Delta,a))$ from the values at the neighbouring nodes. More specifically, the algorithm performs a multilinear interpolation (linear for one dimension, bilinear for two dimensions, etc.) using the 2^n vertexes of the cell containing $y_\Delta(x,\Delta,a)$. Since the grid is regular, the cell can be determined by simple arithmetic and truncation.

The fixed step fourth order Runge-Kutta method was employed to solve the above mentioned set of ordinary differential equations (ODE). Other methods can also be easily used. Some trials were performed using a variable step ODE solver from the GNU Scientific Library (GSL) (Gough, 2003); however, for the considered Δ, the gain in accuracy was not noticeable and, on the other hand, the computation time was noticeably larger.

3.1 Finite and infinite horizon problems

The finite and infinite horizon optimal control problems are associated to time-dependent value functions $V(t,x)$. The computation of $V(t,x)$ starts from the terminal condition $V(0,x)$. Each posterior iteration corresponds to a progression backward in time. For a time-step Δ, the result of the first iteration, V_1, will be the approximation of $V(-\Delta,x)$, the result of the second iteration, V_2, will be the approximation of $V(-2\Delta,x)$ and so on. In practice, we are only interested in the final iteration. For the infinite horizon case, we assume that for a sufficient large number of iterations the value function will converge to the form $V(t,x) = V(x) - ct$. Therefore, for a basic implementation, only two grids must be kept: one holding the result from the previous iteration (for the first iteration, this will be $V(0,x)$) and other for the result of the current iteration. The results of iteration i are based exclusively on the results of iteration $i-1$ (Jacobi iterations).

We are mainly interested in state constrained problems. This is either because the engineering problems at hand so demand or simply due to the fact that the computational space must be bounded so that computations are feasible. The dimensions associated to periodic state variables (such as angular position, for instance) are not affected by the later aspect since the whole state space can be described by a compact set. However, for the general case, there is no simple way to compute $V(y_\Delta(x,\Delta,a))$ if $y_\Delta(x,\Delta,a)$ lies outside the defined computational space. In general, extrapolation schemes will introduce large errors. The safer approach consists of defining a sufficiently large computational space such that its boundaries do not affect the region of interest. Therefore, we assume the system must respect the state constraint $x \in \mathbb{X}$.

For the unconstrained case with smooth dynamics and running cost, the value function is continuous. However, the state constraints introduce discontinuities in the value function. By definition, $V(t,x) = \infty$ for $x \notin \mathbb{X}$. Moreover, the introduction of state-constraints leads to the concept of maximal controlled invariant set (MCIS). The MCIS is the set of states from which there is a trajectory meeting the state constraints afterwards. Depending on the system dynamics, state constraints and input constraints, some of the trajectories beginning on $\mathbb{X}$ cannot be kept inside it for all times. There-

fore, the cost associated to the respective initial state is defined as infinity. This is well captured by the infinite horizon problem. In this sense, the MCIS is defined as $\{x : \lim_{t\to-\infty} V(t,x) \neq \infty\}$.

The discontinuities create difficulties for the interpolation scheme. If one or more vertexes of the cell containing $y_\Delta(x,\Delta,a)$ are defined as infinity, special care has to be taken when performing the interpolation. One possibility, for these cases, is to define $V(t,x) = \infty$. This is a conservative approximation. Only trajectories passing through well defined cells (i.e., not containing vertexes defined as infinity) will be accepted as feasible. However, for certain systems and grid resolutions, this may lead to useless solutions. The forbidden nodes will tend to absorb nodes corresponding to eventually valid trajectories. Moreover, the absorption of nodes will have a cascading effect, i.e., nodes marked as forbidden will tend to absorb other nodes in their neighbourhood. This will happen if the reach set from node x, over the defined time-step, does not intersect a well defined cell. This effect is more pronounced as the Courant number decreases. After some iterations, the computed value function may be finite only on a small region or even infinite everywhere. Therefore, this method will produce a sub-approximation of the MCIS. Whenever the intersection of the reach set from x with the MCIS is a very narrow region (less than one grid cell), the method will disregard x and will converge to a sub-approximation of the MCIS. As explained, this will have a cascading effect. Moreover, some of the resulting trajectories will be sub-optimal in order to escape from the absorbing grid cells.

On the other hand, we may relax the definition of infinity and, instead, consider a very large value $K_{\inf}$ when performing the interpolation. This will produce an over-approximation of the MCIS. The cost near the boundary of the MCIS will be higher than the optimal, due to the introduction of $K_{\inf}$. This diffusive effect is informally designated as smearing. The extent of the smearing is analogous to the absorption effect of the previous method. It will also depend on the reach set from each node and on the grid resolution. The higher the grid resolution, the smaller the region affected by the smearing. Analogously to the previous case, the region affected by the smearing will produce sub-optimal results. This happens because the system tries to escape at all costs from the regions with the artificially higher value introduced by the smearing effect.

This method retains all the feasible trajectories but, on the other hand, it may exhibit "false positives". In this context, a false positive is a state x marked as finite (i.e., $V_i(x) < K_{\inf}$) for which no feasible trajectory can be generated by the resulting control law. This means that a false positive can either correspond to a state outside the exact MCIS or to a state belonging to the MCIS for which the synthesized controller cannot produce a feasible trajectory. If the system is known to have a nonempty equilibrium set S_e or a nonempty set S_s of cells free of smearing effects with finite values at their vertexes, these false positives can be identified by exhaustive simulation of the system trajectories. The trajectories are simulated until they either reach S_s or S_e. However, this is a time consuming procedure.

3.2 Numerical Example

Consider the infinite horizon optimal control of the continuous time linear system (state $x \in \mathbf{R}^2$ and input $u \in \mathbf{R}$)

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \qquad (14)$$

with cost functional

$$\min J(x_0,u) = \int_0^\infty (x^T(\tau)x(\tau) + u^2(\tau))d\tau \qquad (15)$$

for any given $x(0) = x_0$. It is well known that the optimal control for this problem is given by a time-invariant linear state feedback control law, i.e., $u(t) = -Kx(t)$. The associated value function, $V(x) = \min_u J(x,u)$, can also be derived in closed form. For the current example, we have

$$V(x) = x^T \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} x \qquad (16)$$

and $K = \begin{bmatrix} 1 & 1 \end{bmatrix}$. The closed loop system is exponentially stable with a single equilibrium point at the origin.

In what follows, consider the state-constraint $x(t) \in \mathbb{X}$, where $\mathbb{X} = \{x \in \mathbb{R}^2 : \|x\|_\infty \leq 10\}$, and the input constraint $|u(t)| \leq u_{\max}, u_{\max} > 0$ with $u_{\max} = 40$. This will have the following consequences:

1. The set $\mathcal{S}$ of initial states from which there is a trajectory confined to $\mathbb{X}$ (maximal invariant set) is a subset of $\mathbb{X}$; by simple arguments and manipulation of the system equations, it can be easily seen that the minimum distance to the boundary of $\mathbb{X}$ along the x_1 dimension, in the first and third quadrants of the $x_1 - x_2$ phase plane, is given by $|x_2| - u_{\max} \ln\left(\frac{u_{\max}+|x_2|}{u_{\max}}\right)$. See Fig. 1 for details of the boundary set, assuming different values for $u_{\max}$.
2. For certain regions of $\mathcal{S}$, the optimal control law for the constrained problem will be different from the one computed for the unconstrained problem. Obviously, if $u_{\max}$ is set too low, the control signal will be clipped on some regions of $\mathbb{X}$ (that happens, for instance, for $u_{\max} = 10$). On the other hand, if allowed, the absolute value of the optimal control will have to be higher than $|Kx|$ in some regions in order to ensure that trajectories stay inside $\mathcal{S}$.

It is possible to see on Fig. 1 that the controlled invariant set for the system controlled by $u = -Kx$, $\mathcal{S}_{\mathrm{LQ}R}$, is smaller than the maximal invariant set for both $u_{\max} = 20$ and $u_{\max} = 40$. For trajectories starting outside $\mathcal{S}_{\mathrm{LQ}R}$, the absolute value of the optimal control will be higher than $|Kx|$. The main challenge in this case, in terms of numerical scheme, resides in the fact that the optimal trajectories are tangent to the boundary of the invariant set in the curved segments depicted on Fig. 1. Therefore, the interpolation is affected by the discontinuity. However, in this particular example, the smearing effect is not very pronounced. Both methods present a good approximation of the MCIS.

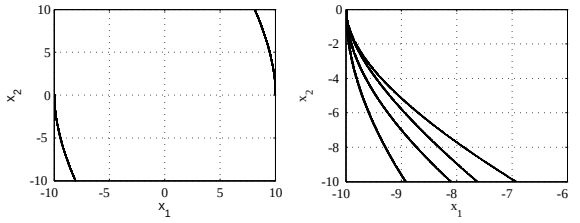

Figure 1. Left figure: boundary of the maximal invariant set for $u_{\max} = 40$. Right figure: detail of the boundary of the invariant set for different scenarios. From left to right: $|u(t)| \leq 40$; $|u(t)| \leq 20$; using the unconstrained LQR; $|u(t)| \leq 10$

The optimal control obtained by (13) with the numerical value function (numerical control law) converges to the exact one in most part of the computational space. The exceptions reside precisely on the neighbourhood of the curved boundary of the invariant set where the smearing occurs. As expected, in those regions, the absolute value of the resulting control is higher than the optimal. This effect can be attenuated by using a finer grid on those regions, as illustrated on Fig. 2 and 3 using the SL scheme with the relaxed notion of infinity. The trajectories are obtained by integration of the system equations with input $u(t)$ given by the numerical control law. At each control cycle, the actuation value is chosen from a set of 801 equally spaced values between $-u_{\max}$ and $u_{\max}$.

4 Semi-Lagrangian receding horizon scheme

In order to overcome the limitations of both of the methods previously described, we propose an algorithm (SLRH) based on concepts of receding horizon control. Recall that, for each x, the basic semi-Lagrangian scheme computes the corresponding optimal trajectory over one time-step. The key idea of the proposed algorithm is to compute the optimal trajectory for an horizon of $N_\Delta > 1$ time-steps when the trajectory does not end in a well defined cell. By optimizing over a larger horizon, the endpoint of the trajectory will have more chances of reaching a well defined cell. Let us define $\mathcal{V}_{i,N_\Delta} \equiv \{V_i, V_{i-1}, \ldots, V_{i-N_\Delta+1}\}, i \geq N_\Delta - 1$ and $\mathcal{V}_{i,N_\Delta} \equiv \{V_i, V_{i-1}, \ldots, V_0\}, i < N_\Delta - 1$. For $N_\Delta = 1$, this algorithm is equivalent to the basic SL scheme with the strict definition of infinity.

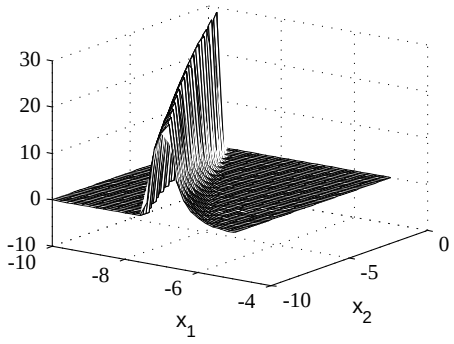

Figure 2. Difference between the optimal control derived from the 201×201 grid and the one derived from the 401×401 grid. The lower resolution leads to a sub-optimal higher actuation near the boundary of the invariant set.

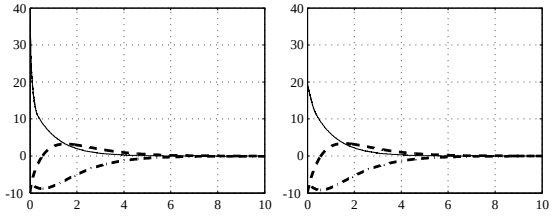

Figure 3. Evolution of $x_1(t)$ (dash-dot line) and $x_2(t)$ (dashed line) with $u(t)$ (solid line) given by the numerical DP control law for the LQR problem on a 201×201 grid (left) and a 401×401 grid (right)

As for the basic scheme, for solving the optimal control problem for a time horizon T, the algorithm starts with $V_0(x) = \Psi(x), \forall x \in \Omega$. It then proceeds iteratively, calling Algorithm 1 for $i = 1, \ldots, N_T$ where N_T is the smaller integer greater than T/Δ. The main algorithm is based on the procedure *ComputeVx*; this procedure is described by Algorithm 2. K_u is a constant that marks the node's value as undefined. All the cells to which that node belongs will be identified as not well defined.

With a sufficiently large N_Δ, this algorithm would be able to avoid any distortion (i.e., sub-optimality) introduced by the discontinuity at the boundary of the MCIS. However, the processing time and storage requirements grow exponentially with N_Δ. In order to minimize these requirements, we consider a reduced input space $U_{a,i}$ for the larger horizons. For many problems, one sensible approach is to consider only extremal controls when optimizing for more than one time-step (e.g., $U_{a,1} = U_a$ and $U_{a,i} = \{\min(U_a), \max(U_a)\}, i > 1$). Additionally, as soon as one valid cell is found, the procedure stops extending the optimization horizon. This accelerates the computation but, on the other hand, it may prevent the eval-

uation of better trajectories. In general, these two optimizations will lead to sub-optimality near the boundary.

Input: Ω, i, $\mathcal{V}_{i-1,N_\Delta}$, N_Δ
Output: $\mathcal{V}_{i,N_\Delta}$
foreach $x \in \{x : x \in \Omega \wedge V_{i-1}(x) \neq \infty\}$ **do**
  $\{V_i(x), \mathsf{defined}\} \leftarrow$ ComputeVx(x,$\mathcal{V}_{i-1,N_\Delta}$, N_Δ);
  if defined $= false$ **then**
    if $i \geq N_\Delta$ **then**
      $j \leftarrow i-1$;
      while $j > i - N_\Delta \wedge j > 0$ **do**
        $V_j(x) = \infty$;
        $j \leftarrow j-1$;
      end
    else
      $V_i(x) \leftarrow K_u$;
    end
  end
end
$\mathcal{V}_{i,N_\Delta} \leftarrow \{V_i\} \cup \mathcal{V}_{i-1,N_\Delta} \setminus \{V_{i-N_\Delta}\}$

Algorithm 1: Compute V_i using the semi-Lagrangian receding horizon scheme

The memory requirements for the proposed algorithm may seem higher than for the classical SL scheme. In fact, the classical SL scheme requires only the storage of two sets of values: the values computed at each grid node at the current iteration, V_i, and the values from the previous iteration V_{i-1}. The proposed algorithm requires the storage of $N+1$ sets of values. However, by careful tuning the maximum optimization horizon, N_Δ, and choosing a suitable $U_{a,i}$, the grid resolution can be lower than in the basic semi-Lagrangian scheme while still providing a better sub-approximation of the MCIS. Nevertheless, for the real-time implementation of the control law, the worst case optimization time should be kept within the processing capabilities. Therefore, the value of N_Δ must be chosen with that in mind.

4.1 Numerical example

Consider the following system:

$$\dot{x} = \begin{cases} \sin(x_2) \\ a \end{cases} \tag{17}$$

with $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}'$ and input $a \in U_a$, where U_a is a set of 31 equally spaced values from -0.25 to 0.25. We define $U_{a,i} = \{-0.25, 0.25\}, i > 1$. The system is constrained to $[-2,2] \times [-\pi/2, \pi/2]$. The grid resolution is 201×201. All the experiments are performed with $\Delta = 0.1$.

Fig. 4 shows the contour of the approximated MCIS for different values of N_Δ. For $N_\Delta = 1$, a significant contraction of the MCIS is observed. The results

Input: x, $\mathcal{V}_{i-1,N_\Delta}$, N_Δ
Output: $V_i(x)$, defined
$S_{\text{considered}} \leftarrow \{x\}$;
$V_i(x) \leftarrow \infty$;
level $\leftarrow 1$;
finished $\leftarrow$ false;
defined $\leftarrow$ false;
repeat
  $S^{\text{new}}_{\text{considered}} \leftarrow \emptyset$
  foreach $x \in S_{\text{considered}}$ **do**
    foreach $a \in U_{a,\text{level}}$ **do**
      Calculate $y_\Delta(x, \Delta, a)$;
      $c \leftarrow \int_0^\Delta L(y_\Delta(x,\tau,a), a) d\tau$;
      $v \leftarrow V_{i-\text{level}}(y_\Delta(x, \Delta, a))$;
      if v *is well defined* **then**
        if $V_i(x) > c + v$ **then**
          $V_i(x) \leftarrow c + v$;
          defined $\leftarrow$ true;
        end
      else
        $S^{\text{new}}_{\text{considered}} \leftarrow$
        $S^{\text{new}}_{\text{considered}} \cup \{y_\Delta(x, \Delta, a)\}$;
      end
    end
  end
  if level $= N_\Delta \vee i -$ level $= 0 \vee$ defined **then**
    finished $\leftarrow$ true
  else
    $S_{\text{considered}} \leftarrow S^{\text{new}}_{\text{considered}}$;
    if $S_{\text{considered}} = \emptyset$ **then**
      finished $\leftarrow$ true
    end
  end
until finished;

Algorithm 2: Compute $V_i(x)$ using up to N_Δ time-steps

improve significantly even with small values of N_Δ. However, we remark that this is highly dependent on the Courant number. A change in the grid resolution or time step leads to different results. In this case, the classical SL (using the relaxed definition of infinity) produces a noticeably rough over-approximation of the MCIS.

Table 1 shows the number of nodes in the approximation of the MCIS for different values of N_Δ. It also shows the CPU time for 68 iterations. This is the number of iterations required by the algorithm to converge to the approximation of the MCIS for $N_\Delta = 1$. For $N_\Delta \geq 40$, the computation times do not differ significantly. This is because the extra computation time near the boundary is negligible when compared to the computation of the remaining nodes. Moreover, the number of nodes for which the entire optimization horizon $N_\Delta \Delta$ must be used gets smaller as N_Δ increases.

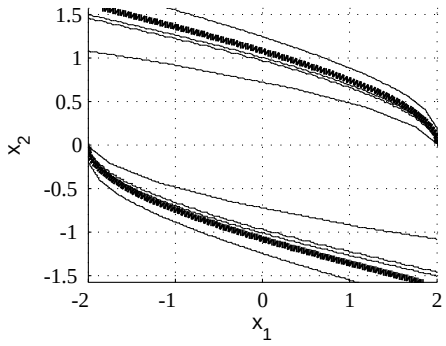

Figure 4. Computation of the MCIS. The thicker line corresponds to the exact MCIS. The inner contours correspond to the results of the SLRH with different N_Δ: 1, 2 and 4 (from the inner to the outer contour). The outer contour corresponds to the over-approximation obtained with the classical SL scheme.

Table 1. Approximation of the MCIS with the SLRH algorithm

N_Δ	Number of nodes	CPU Time (s)
1	17197	32
2	23351	38
3	23715	39
4	24303	39
10	25726	41
20	26035	42
40	26108	44
50	26120	44
60	26121	44
70	26121	45

5 Conclusions

The typical SL solver based on first order interpolation leads to sub-optimal optimal control laws for non-smooth value functions, with excessive or anticipated action near the non-smooth regions. Moreover, the obtained MCIS is an over-approximation of the exact MCIS. These effects can be minimized by refining the grid near those regions. However, memory constraints may sometimes prevent such approach. The alternative technique of avoiding cells near the boundary of the computed MCIS may lead to over-conservative sub-approximations of the MCIS. The proposed algorithm gives an accurate sub-approximation of the MCIS up to the grid resolution. The accuracy of the value function near the boundary of the MCIS can be adjusted by tuning the optimization horizon and the considered input space.

Acknowledgements

The first author was partially supported by the FCT PROTEC program. The second author was partially funded by the projects Persist and Noptilus.

References

Bardi, Martino and I. Capuzzo-Dolcetta (1997). *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Birkhauser Boston.

Bellman, R. (1957). *Dynamic programming*. Princeton University Press. Princeton.

Blanchini, Franco and Stefano Miani (2008). *Set-Theoretic Methods in Control*. Birkhauser. Boston.

Cristiani, E. and M. Falcone (2009). Fully-discrete schemes for the value function of pursuit-evasion games with state constraint. In: *Advances in Dynamic Games and Their Applications: Analytical and Numerical Developments* (Pierre Bernhard, Vladimir Gaitsgory and Odile Pourtallier, Eds.). Vol. 10 of *Annals of International Society of Dynamic Games*. pp. 178–205. Birkhäuser Boston.

Fleming, Wendell H. and Halil Mete Soner (2006). *Controlled Markov Processes and Viscosity Solutions*. Springer. New York.

Gough, Brian (2003). *GNU Scientific Library Reference Manual - 2nd Edition*. Network Theory Ltd.

Krasovskii, N.N. and A.I. Subbotin (1988). *Game-theoretical control problems*. Springer-Verlag. New York.

Mitake, Hiroyoshi (2008). Asymptotic solutions of hamilton-jacobi equations with state constraints. *Applied Mathematics and Optimization* **58**(3), 393–410.

Mitchell, I.M., A.M. Bayen and C.J. Tomlin (2005). A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control* **50**(7), 947–957.

Pontryagin, L. S., V. Boltyanskii, R. Gamkerelidze and E. Mischenko (1962). *The Mathematical Theory of Optimal Processes*. Interscience Publishers.