0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

1 AI-generated only 0%
Likely AI-generated text from a large-language model.

2 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

A Modified Iterative Method for Solving the Hamilton-Jacobi-Bellman Equation

Hartono[1, a)]

[1]*Department of Mathematics, Sanata Dharma University*
Kampus III Paingan, Maguwoharjo, Depok, Sleman,Yogyakarta, Indonesia

a) yghartono@usd.ac.id

Abstract. In the field of optimal control, the Hamilton-Jacobi-Bellman equation specifies both the necessary and sufficient condition for finding optimal control with respect to the intended objective function. The equation is a nonlinear partial differential equation which is generally intractable to be solved analytically. Hence, in order to obtain the solution of some optimal control problem formulated in the Hamilton-Jacobi-Bellman equation, it is necessary to develop some reliable and efficient numerical method. In this article, we propose a modified version of our iterative method, previously published in a journal, to solve the state-constrained Hamilton-Jacobi-Bellman equation. The modification is made on the way to update the value function of the objective function. In this new scheme, instead of updating the value function on each point on the domain, we select only some points neighboring a nominee of the optimal path making up the solution of the optimal control problem. Therefore, comparing to the old scheme, the computation results not only a reliable solution but it is also much faster and efficient.

INTRODUCTION

Optimal control is a field of Mathematics and Engineering that aims to find an optimal way to control a dynamical system. This kind of problem naturally occurs in decision making of many aspects such as science, engineering, finance and management [2,6]. Basically, to solve an optimal control problem we need to optimize some objective function under constraints of a differential equation system. In the literature, there are two ways to solve an optimal control problem. We can use a method of Pontryagin Minimum Principle or Bellman Dynamic Programming. The former method results in an open loop problem whose solution is easier to find and constitutes an optimal control along the optimal trajectory from the initial state. Unfortunately, this solution is not robust. If for some reason the state is off the optimal trajectory then the corresponding optimal control is no longer valid. On the other hand, using Bellman Dynamic Programming the optimal control problem will be formulated in Hamilton-Jacobi-Bellman equation [3,5].

In this article we will solve an optimal control problem with a state-constraint in the form of

$$\min_{u\in\Omega} J(u) = \int_0^1 L(x,u,t)\ dt + \Phi(x(1))$$

subject to

$$\frac{dx}{dt} = f(x,u,t), \quad t \in (0,1),\ x(0) = z$$

$$\Omega = \{u | h(x,u,t) \le 0\}, \quad t \in (0,T] \qquad (1)$$

This constrained problem can be converted to unconstrained problem by incorporating linear penalty terms in the objective function as follows.

$$\min_u P(u) = J(u) + \int_r^T \lambda\ h_\varepsilon(x,u,t)\ dt$$
$$\text{subject to}$$
$$\frac{dx}{dt} = f(x,u,t), \quad t \in (r,T),\ x(r) = z \qquad (2)$$

The corresponding Hamilton-Jacobi-Bellman equation of this unconstrained problem is the following.

$$\frac{\partial V}{\partial t} + \min_u\ (\nabla V \bullet f(x,u,t) + L(x,u,t) + \lambda\ h_\varepsilon(x,u,t)) = 0$$
$$V(T,x) = \Phi(x(T)) \qquad (3)$$

The solution of Hamiton-Jacobi-Bellman equation is a robust solution because it is defined over a time-space region that the optimal trajectory lays. Hence, the corresponding optimal control still can be traced if the state is off the optimal path. However, the Hamilton-Jacobi-Bellman equation is generally unsolvable analytically. Therefore, we should solve it numerically. There are some methods for solving it such as in [1, 3] and [5] to name but a few. However, most of them are devoted to solve general Hamilton-Jacobi-Bellman equation without constraint on control and state. Our proposed numerical method is unique because it aims to solve constrained Hamilton-Jacobi-Bellman equation using a penalty method.

MODIFIED ITERATIVE METHOD

This modified iterative method is a modification of method in the article [4]. The old method uses an iterative finite upwind difference method to evaluate the value function on all grid points in the time and space domain. The method we present here evaluate the value function on selected grid points in time and space domain without sacrificing accuracy of the computation. As a consequence, our new method will be faster and more efficient in terms of computational time. In the following part, we describe the method in detail and give a numerical simulation to show the effectiveness of the method.

First Iteration

Without loss of generalization, let us simplify the notation by taking an optimal control problem with 1 control variable u and 2 state variables $x_p \in [a_p,\ b_p],\ p = 1,\ 2$. First, we will discretize space and time interval into m and n partitions respectively. Therefore, we have

$$\Delta\ x_p = \frac{b_p - a_p}{m}, \quad x_{p,i} = a_p + (i-1)\ \Delta\ x_p, \quad p = 1,2 \quad , i = 1,\ 2, \dots,\ m+1 \qquad (4)$$

An appropriate shifting needs to be done such that this spatial discretization in any case contains the initial point. Let

$$s = arg\ \min_i \left| x_{p,i} - x_p(0) \right| \qquad (5)$$

then, with some adjustment for $i = 1, 2, \dots, m+1$ we obtain

$$x_{p,i} \leftarrow x_{p,i} + (x_p(0) - x_p(s)),\ a_p \leftarrow a_p + (x_p(0) - x_p(s)), \quad b_p \leftarrow b_p + (x_p(0) - x_p(s)) \qquad (6)$$

In this scheme, in order to work well the upwind finite difference method that we use need a stability condition as reported in , i.e.

$$n \geq \sum_{p=1}^{2} \frac{\|f_p\|_\infty}{\Delta\ x_p} \qquad (7)$$

Hence,

$$\Delta t = -\frac{1}{n}, \quad t_k = 1 + (k-1)\ \Delta t, \quad k = 1,\ 2, \dots, n \qquad (8)$$

is the backward partition of time interval [0,1]. We set the initial value function and control value as follows.

$$V_{i,j}^1 = \Phi(\vec{x}_{i,j}(1)), \quad i,j = 1,\ 2, \dots,\ m+1 \qquad (9)$$

$$u_{i,j}^1 = arg \min_u \left(f_1(t_1, \vec{x}_{i,j}, u)\frac{V_{i,j}^1 - V_{i-1,j}^1}{\Delta x_1} + f_2(t_1, \vec{x}_{i,j}, u)\frac{V_{i,j}^1 - V_{i,j-1}^1}{\Delta x_2} + L(t_1, \vec{x}_{i,j}, u)\right.$$
$$\left. + \lambda\, h_\varepsilon(t_1, \vec{x}_{i,j}, u)\right), \quad i, j = 2,\ 3, \dots,\ m+1. \qquad (10)$$

In the above formulas, $V_{i,j}^k \approx V(t_k, \vec{x}_{i,j})$ and $u_{i,j}^k \approx u\ (t_k, \vec{x}_{i,j})$ are value function and control variable at point $\vec{x}_{i,j} = (x_{1,i}, x_{2,j})$ and time t_k. The function $h_\varepsilon(t_1, \vec{x}_{i,j}, u)$ is smoothed form of constrain $h(t, x, u)$ evaluated at $(t_1, \vec{x}_{i,j}, u)$ and $\lambda$is a constant of explicit penalty term used to embed constraint to the objective function.

To prevent a trapezoidal propagation on the boundary of space domain for each time step, we add some artificial boundary conditions for control function. This boundary values are predicted using a linear extrapolation of the closest known points as the following.

$$u_{i,1}^1 = 2\ u_{i,2}^1 - u_{i,3}^1,\ u_{1,j}^1 = 2\ u_{2,j}^1 - u_{3,j}^1 \qquad (11)$$

To update value function for $i, j = 1,\ 2, \dots, m+1, \quad k = 1,\ 2, \dots, n$

$$V_{i,j}^{k+1} = -\frac{1+sign\ f_1}{2}\beta_1\ f_1(t_k, \vec{x}_{i,j}, u_{i,j}^k)\ V_{i+1,j}^k - \frac{1+sign\ f_2}{2}\beta_2\ f_2(t_k, \vec{x}_{i,j}, u_{i,j}^k)\ V_{i,j+1}^k$$
$$+\frac{1-sign\ f_1}{2}\beta_1\ f_1(t_k, \vec{x}_{i,j}, u_{i,j}^k)\ V_{i-1,j}^k + \frac{1-sign\ f_2}{2}\beta_2\ f_2(t_k, \vec{x}_{i,j}, u_{i,j}^k)\ V_{i-1,j}^k$$
$$+(1 + \textstyle\sum_{p=1}^2 \beta_p\ \left|f_p(t_k, \vec{x}_{i,j}, u_{i,j}^k)\right|\ )\ V_{i,j}^k - \Delta t\ L(t_k, \vec{x}_{i,j}, u_{i,j}^k)\ - \Delta t\ \lambda\ h_\varepsilon(t_k, \vec{x}_{i,j}, u_{i,j}^k) \qquad (12)$$

where $sign\ f_p$ denotes the sign of $f_p(t_k, \vec{x}_{i,j}, u_{i,j}^k)$ and $\beta_p = \frac{\Delta t}{\Delta x_p}, \quad p = 1,\ 2.$

For both ends of spatial domain, the value functions at those points are extrapolated using method similar to the (11).

$$V_{i,1}^{k+1} = 2\ V_{i,2}^{k+1} - V_{i,3}^{k+1},\ V_{i,m+1}^{k+1} = 2\ V_{i,m}^{k+1} - V_{i,m-1}^{k+1}$$
$$V_{1,j}^{k+1} = 2\ V_{2,j}^{k+1} - V_{3,j}^{k+1},\ V_{m+1,j}^{k+1} = 2\ V_{m,j}^{k+1} - V_{m-1,j}^{k+1} \qquad (13)$$

In addition, to update control values for $\boldsymbol{i, j = 2,\ 3, \dots, m+1, \quad k = 1,\ 2, \dots, n}$

$$u_{i,j}^{k+1} = arg \min_u \left(f_1(t_{k+1}, \vec{x}_{i,j}, u)\frac{V_{i,j}^{k+1} - V_{i-1,j}^{k+1}}{\Delta x_1} + f_2(t_{k+1}, \vec{x}_{i,j}, u)\frac{V_{i,j}^{k+1} - V_{i,j-1}^{k+1}}{\Delta x_2}\right.$$
$$\left. + L(t_{k+1}, \vec{x}_{i,j}, u) + \lambda\ h_\varepsilon(t_{k+1}, \vec{x}_{i,j}, u)\right). \qquad (14)$$

For left and right boundaries for control value

$$u_{i,1}^{k+1} = 2\ u_{i,2}^{k+1} - u_{i,3}^{k+1},\ u_{1,j}^{k+1} = 2\ u_{2,j}^{k+1} - u_{3,j}^{k+1} \qquad (15)$$

Up to this point, we already have $V_{i,j}^k$ and $u_{i,j}^k$ for $i, j = 1,\ 2,\ \dots, m+1, \quad k = 1,\ 2, \dots, n+1$. To find the optimal trajectory and control, we need to integrate forward the state equation from the starting point

$$\frac{d\vec{x}}{dt} = \vec{f}\ (t,\ \vec{x},\ u),\ \vec{x} = (x_1, x_2), \quad \vec{f} = (f_1,\ f_2) \qquad (16)$$

using predictor-corrector method as follows. Let us label the resultant path and control for predictor $\vec{y}_q = (y_{q,1}, y_{q,2}),\ u_q$ and $\vec{y}_c = (y_{c,1}, y_{c,2}),\ u_c$ for corrector with $\vec{y}_q(1) = \vec{y}_c(1) = \vec{x}_0$ and $u_c(1) = u\ (\vec{x}_0, t_{n+1})$ respectively. The control value used during the integration is the optimal control value from the closest grid point to the resultant state. Then, for $l = 2, \dots, n+1$

$$\vec{y}_q(l) = \vec{y}_c(l-1) - \Delta t\ \vec{f}\ (t_{-l+n+3},\ \vec{y}_q(l),\ u_c(l-1))$$
$$u_q(l) = u(\vec{x}_{i*,j*}(l),\ t_{-l+n+3}), \quad (i*, j*) = arg \min_{(i,j)} \left\|\vec{y}_q(l) - \vec{x}_{i,j}\right\|$$
$$\vec{y}_c(l) = \vec{y}_c(l-1) - \frac{1}{2}\Delta t\ (\vec{f}\ (t_{-l+n+3}, \vec{y}_c(l-1), u_c(l-1)) + \vec{f}\ (t_{-l+n+3}, \vec{y}_q(l), u_q(l))$$

$$u_c(l) = u(\vec{x}_{i*,j*}, t_{-l+n+3}), \quad (i*, j*) = arg \min_{(i,j)} \left\| \vec{y}_c(l) - \vec{x}_{i,j} \right\| \tag{17}$$

The resultant pair $(\vec{y}_c(l), u_c(l)), \ \ l = 1, \ 2, \ \ldots n+1$constitutes an optimal trajectory and control for all time stages. Moreover, the value function along the optimal trajectory can be obtained using the value function of the corresponding closest grid points. Similarly, the penalty value and objective function value can be determined by forward integration along the optimal trajectory of corresponding terms.

Second Iteration

Next, for the second iteration we firstly reduce the domain based on the optimal trajectory and control found in the first iteration. This reduction will save a lot of computational time and is done as following. First, we determine the maximum and the minimum value of resultant path and control. For $\boldsymbol{p = 1, \ 2}$ and $\boldsymbol{l = 1, \ 2, \ \ldots n+1}$

$$\begin{aligned} x_{p,\max} &= \max y_{c,p}(l), \quad x_{p,\min} = \min y_{c,p}(l) \\ u_{\max} &= \max u_c(l), \quad u_{\min} = \min u_c(l) \end{aligned} \tag{18}$$

For this iteration we double the number interval partitions m and set the region as follows.

$$\begin{aligned} &m_{new} = 2\, m, \quad \Delta x_{p,new} = \frac{x_{p,\max} - x_{p,\min}}{m_{new}}, \quad a_{p,new} = x_{p,\min} - d\ \Delta x_{p,new} \\ &b_{p,new} = x_{p,\max} + d\ \Delta x_{p,new}, \quad u_{low} = \lfloor u_{\min} \rfloor, \quad u_{up} = \lceil u_{\max} \rceil \end{aligned} \tag{19}$$

where $\lfloor z \rfloor$ fungsi floor, $\lceil z \rceil$ fungsi ceiling and $d$some given constant. This constant$d$ is a prescribed distance for selected grid points from the optimal trajectory. The shrinkage factor ρ_p can be evaluated using the following formula.

$$\rho_p = \frac{b_{p,new} - a_{p,new}}{b_p - a_p}, \quad p = 1, \ 2. \tag{20}$$

This shrinkage factor will be used to update distance d.

Then, we determine the number of time stages as previously explained in (7) and discretize the spatial domain as in (4). Shifting the spatial discretization to include the initial value using (5) and (6) can be done if needed. We select some grid points among the existing ones in the space domain based on the distance from the optimal trajectory resulted from the first iteration. In principle, we choose points $\vec{x}^d$ closed to the optimal trajectory $\vec{y}_c$ within some given distance$d$.

$$\|\vec{y}_c - \vec{x}^d\|_1 < d \tag{21}$$

Afterwards, we set the value function and control for these selected grid points. There are two kinds of grid points that need to be considered, i.e. interior and border points. Interior points are points that are surrounded by other selected points and therefore their initial value function is determined following (10). For border points, points that are directly adjacent to unselected points, at position (i, j) and time $k = 1$the value function will be one of the following possible linear extrapolation

$$\begin{aligned} V_{i,j}^k &= V_{i,j-1}^k + V_{i-1,j}^k - V_{i-1,j-1}^k \\ V_{i,j}^k &= V_{i-1,j}^k + V_{i,j+1}^k - V_{i-1,j+1}^k \\ V_{i,j}^k &= V_{i,j-1}^k + V_{i+1,j}^k - V_{i+1,j-1}^k \\ V_{i,j}^k &= V_{i+1,j}^k + V_{i,j+1}^k - V_{i+1,j+1}^k \\ V_{i,j}^k &= 2\ V_{i,j-1}^k - V_{i,j-2}^k \\ V_{i,j}^k &= 2\ V_{i-1,j}^k - V_{i-2,j}^k \\ V_{i,j}^k &= 2\ V_{i,j+1}^k - V_{i,j+2}^k \end{aligned}$$

$$V_{i,j}^k = 2\ V_{i+1,j}^k - V_{i+2,j}^k \qquad (22)$$

The right hand sides of the above formulas are the value function of the interior points set before. One of these value functions always exist due to the compactness of points close to the optimal trajectory for some (big enough) given distance d. Which extrapolation we choose really depends on the availability of interior points. Nevertheless, extrapolation involving more points is preferable because we want to have a better approximation to the true value.

Analogously, the initial control for interior points is determined using formula in (10) and for border points at position (i,j) and time $k = 1$will be one of the following possible linear extrapolation

$$\begin{aligned} u_{i,j}^k &= u_{i,j-1}^k + u_{i-1,j}^k - u_{i-1,j-1}^k \\ u_{i,j}^k &= u_{i-1,j}^k + u_{i,j+1}^k - u_{i-1,j+1}^k \\ u_{i,j}^k &= u_{i,j-1}^k + u_{i+1,j}^k - u_{i+1,j-1}^k \\ u_{i,j}^k &= u_{i+1,j}^k + u_{i,j+1}^k - u_{i+1,j+1}^k \\ u_{i,j}^k &= 2\ u_{i,j-1}^k - u_{i,j-2}^k \\ u_{i,j}^k &= 2\ u_{i-1,j}^k - u_{i-2,j}^k \\ u_{i,j}^k &= 2\ u_{i,j+1}^k - u_{i,j+2}^k \\ u_{i,j}^k &= 2\ u_{i+1,j}^k - u_{i+2,j}^k \end{aligned} \qquad (23)$$

The value function at next time stages follows from (12) for interior points and (22) for border points. Similarly, control for later time stages can be updated using (15) for interior points and (23) for border points. So far, we have $V_{i,j}^k$ and $u_{i,j}^k$ for $i,j = 1,\ 2,\ \dots, m+1, \quad k = 1,\ 2, \dots, n+1$, and therefore we have already updated all values. Next, we integrate forward the dynamical system to obtain optimal pair $(\vec{y}_c, u_c)$ for all time stages using predictor-corrector method as before. If during the integration, the resultant trajectory is off the selected grid points, we enlarge the distance d by one grid spacing and start again the integration with this new distance.

Like in the first iteration, next, we need to reduce the region size. To include these selected points for the next iteration the distance is updated as follows.

$$\gamma = max(\tfrac{1}{\rho_1}, \tfrac{1}{\rho_2}), \quad d_{new} = 2\ \gamma\ d. \qquad (24)$$

where $\rho_p, p = 1,2$ is the shrinkage factor of space interval between two consecutive iterations.

Next Iteration

Repeat all steps in the second iteration for later iterations. We stop the iteration if maximum iteration is reached or the difference between two consecutive value function at the initial state point are less than some prescribed tolerance.

NUMERICAL SIMULATION

In this section we will run a numerical simulation to show the effectiveness of the proposed method. The example taken from [4] is as follows.

$$\begin{aligned} &\min_u \quad J(u) = \int_0^1 ({x_1}^2 + {x_2}^2 + 0.005\ u^2) \\ &\text{subject to} \\ &\frac{dx_1}{dt} = x_2, \quad x_1(0) = 0 \\ &\frac{dx_2}{dt} = -x_2 + u, \quad x_2(0) = -1 \\ &g\ (t,x) = -8(t-0.5)^2 + 0.5 + x_2 \le 0, \quad t \in [0,1] \end{aligned} \qquad (25)$$

The last constraint constitutes a purely state constraint so that it does not give any information on how to choose a control that satisfies it. Therefore, it should be changed to a new constraint according to as following.

$$h\ (t,u,x) = 0.9\ g\ (t,x) + 0.1\ \tfrac{dg\ (t,x)}{dt} \qquad (26)$$

Furthermore, in order to be able to work with optimization routine this constraint need to be smoothed as follows.

$$h_\varepsilon = \begin{cases} 0, & h < -\varepsilon \\ \frac{1}{4}\frac{(h+\varepsilon)^2}{\varepsilon}, & -\varepsilon \le h \le \varepsilon \\ h, & h > \varepsilon \end{cases} \quad (27)$$

The above optimal control problem can be converted into Hamilton-Jacobi-Bellman (HJB) equation

$$\frac{\partial V}{\partial t} + \min_u \left(x_2 \frac{\partial V}{\partial x_1} + (u - x_2) \frac{\partial V}{\partial x_2} + {x_1}^2 + {x_2}^2 + 0.05\, u^2 + \lambda\, h_\varepsilon(t, x, u)\right) = 0$$
$$V(1, x) = 0 \quad (28)$$

In this numerical simulation we set $\lambda = 2,\ \varepsilon = 10^{-2}$and at the first iteration, we choose the region $-1 \le x_1 \le 1,\ -3 \le x_2 \le 1,\ -20 \le u \le 20,\ m = 8$. For various values of m the result of the computational results are displayed in Table 1 and Table 2.

TABLE 1. Computational value function

Iterations	Distance d	m	n	Penalty	Objective function	Value function	Control range
1	-	8	46	0.204	0.1763	0.4378	[-20,20]
2	2.00	16	164	0.020	0.2016	0.2186	[-6,14]
3	4.65	32	331	0.024	0.1921	0.2141	[-3,14]
4	9.26	64	658	0.003	0.2044	0.2099	[-3,14]
5	19.13	128	1331	0.003	0.2025	0.2078	[-3,14]

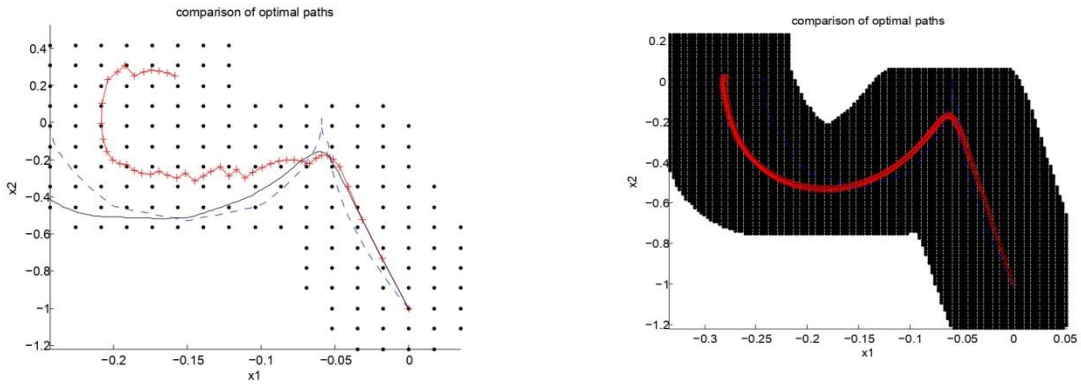
Columns in Table 1 shows us respectively the number iterations, distance, the number of partitions for space and time domain, penalty and objective value along the optimal path, value function at the initial point and control range that we used. We see that from iteration to iteration penalty value and value function decreases as expected. Meanwhile, the value function increases and the discrepancy between the value function and the objective function is getting smaller.

TABLE 2. Computational shrinkage factors

Iterations	Distance d	m	n	x_1 range	x_2 range	ρ_1	ρ_2	% m
1	-	8	46	[-1, 1]	[-3,1]	0.14	0.44	-
2	2.00	16	164	[-0.243, 0.034]	[-1.217,0.523]	1.24	0.86	60
3	4.65	32	331	[-0.310,0.034]	[-1.217,0.281]	1.06	1	65
4	9.26	64	658	[-0.316,0.050]	[-0.217,0.286]	1.06	0.97	63
5	19.13	128	1331	[-0.334,0.053]	[-0.217,0.238]	1.02	1	63

Table 2 provides some information related to the range of spatial domain and the shrinkage factor, i.e. the ratio of latter space interval length to former. The smaller the shrinkage factor is, the larger the reduction for the next iteration. The shrinkage factor nearly 1 indicates that the length of the space interval for the next iteration will not change much. Moreover, if the control range is fixed, this also means that the iteration is almost convergent and further improvement will be impossible. On the contrary, the shrinkage factor exceeding 1 indicates that the previous interval length is too short so that the method automatically adapts it to restore. The last column informs us the percentage of grid point evaluation compared to method without grid selection. It can be seen that for this simulation the method only evaluate not more than 65% of the complete set of grid points. Consequently, the efficiency of computational time can be increased about 35% and the best value function found is just insignificantly different from the result reported in.

The following figures shows the comparison between the optimal path generated by two consecutive iterations and MISER 3.3, a reliable software for solving optimal control problem in [7].

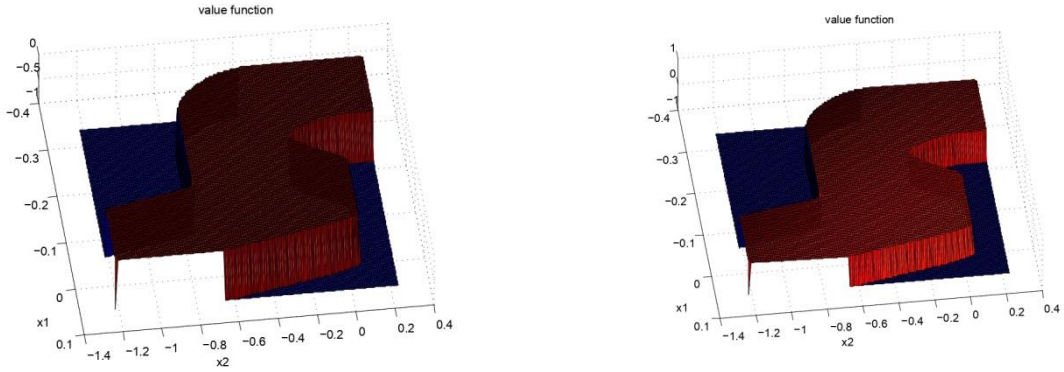


(a) (b)

FIGURE 1. Optimal path comparison of two consecutive iterations, i.e. iteration 2 in (a) and last iteration in (b)

Figure 1(a) shows optimal paths produced by first iteration (-+), second iteration (-) and MISER 3.3 (- -). It is noted that the optimal paths from the first two iterations and MISER 3.3 are different. However, in the last iteration in figure 1(b) this discrepancy disappears and the optimal finally coincide. This indicates that from time to time the method could improve the computation such that the optimal paths produced converge to the true solution.

Figures 2 below show the computational result from the last iteration for value function at initial time (t=0) in (a) and at the final time (t=1) in (b) . The similar basic S-shape in the figures indicate that the method is quite stable under proposed distance d.



(a) (b)

FIGURE 2. The value function in the last iteration at initial time (t=0) in (a) and final time (t=1) in (b)

CONCLUSION

In this article, we modify the method of iterative upwind finite difference method in order to improve the speed of computation. From the numerical simulation the new proposed method is much more efficient than the previous one. This is due to the selection of grid points evaluated in each iteration. Furthermore, in higher dimensions using the proposed method the computation could be even more efficient. However, it is necessary to investigate the minimal value of distance d further because choosing d too small will likely result in instability of the scheme.

REFERENCES

1. L. Grune, M. Falcone, R. Ferretti and W. M. McEneaney, *Numerical Methods for Optimal Control Problems* (Springer, Cham, 2019).
2. S. P. Sethi, *Optimal Control Theory* (Springer, Cham, 2019).
3. K. Dante, K. Kunisch and Z. Rao, *Hamilton-Jacobi-Bellman Equations* (De Gruyter, Berlin, 2018).
4. Hartono, L. S. Jennings and S. Wang, Pacific Journal Optimization **12(2)**, 379–397 (2016).
5. A. Festa, R. Guglielmi, C. Hermosilla et al, "Hamilton-Jacobi-Bellman Equations", in *Optimal Control: Novel Directions and Applications*, edited by D. Tonon et al. (Springer, Cham ,2017) pp. 127 – 261.
6. I. M. Ross, R. J. Proulx and M. Karpenko, "An Optimal Control Theory for the Traveling Salesman Problem and Its Variants," https://arxiv.org/abs/2005.03186, 2020.
7. L. S. Jennings, "Miser 3 Optimal Control Software", Theory and User Manual version 3, University of Western Australia, 2004.