Sequential Multiresolution Trajectory Optimization for Moving Targets

Sachin Jain* and Panagiotis Tsiotras†

Georgia Institute of Technology, Atlanta, GA 30332-0150, USA

In this paper we present two sequential multiresolution trajectory optimization algorithms for solving problems with moving targets and dynamically changing environments. For such problems, high accuracy is desirable only in the immediate future, yet the ultimate mission objectives should be accommodated as well. An intelligent trajectory generation for such situations is thus enabled by introducing the idea of multigrid temporal resolution to solve the associated trajectory optimization problem on a non-uniform grid across time that is adapted to: (i) immediate future, and (ii) potential discontinuities in the state and control variables.The numerical examples solved herein demonstrate the efficacy of the proposed technique.

I. Introduction

A common line of attack for solving nonlinear trajectory optimization problems in real time[1–4] is to break the problem into two phases: an offline phase and an online phase. The offline phase consists of solving the optimal control problem for various reference trajectories and storing these reference trajectories onboard for later online use. These reference trajectories are used to compute the actual trajectory online via a neighboring optimal feedback control strategy[5–8] typically based on the linearized dynamics. This approach requires extensive ground-based analysis and onboard storage capabilities.[9] Moreover, perturbations around the reference trajectories might not be small, and therefore applying the linearized equations may not be appropriate.

To illustrate the previous point, consider the problem of finding the optimal control that will steer the system from point A to the target point B under certain path constraints at a minimum cost. If the target point B is far off, then there is no real advantage of finding the optimal trajectory online with high precision from the starting point till the end. As we continue to move from point A towards the target point B, we can get more accurate information about the surrounding environment (path constraints), which may be different from what was assumed at the beginning when the trajectory was optimized. Moreover, the path constraints and the terminal constraints may also change as the vehicle progresses towards point B. For example, the target point B may not be stationary. One way of handling this problem is to use the receding horizon approach,[10–12] in which a trajectory that optimizes the cost function over a period of time, called the *planning horizon*, is designed. The trajectory is implemented over the shorter *execution time* and the optimization is performed again starting from the state that is reached at the end of the execution time. However, if the planning horizon length does not reach the target B, the trajectory found using this approach might not be optimal. One would like to solve the nonlinear trajectory optimization problem online for the whole time interval, but with high accuracy only near the current time. Recently, some work has been done in this direction by Kumar et al.[9] and Ross et al.[13] Kumar and Seywald[9] proposed a dense-sparse discretization technique in which the trajectory is discretized by placing N_D dense nodes close to the current time and N_S sparse nodes for the rest of the trajectory. The state values at some future node are accepted as optimal and are prescribed as the initial conditions for the rest of the trajectory. The remainder of the trajectory is again discretized using a dense-sparse discretization technique, and the whole process is repeated again. The algorithm can be stopped by using any adhoc scheme, for example, it can be terminated when the density

*Ph.D. candidate, School of Aerospace Engineering, Email: sachin.jain@gatech.edu.

†Professor, School of Aerospace Engineering, Email: tsiotras@gatech.edu. Associate Fellow AIAA.

of the dense nodes is less than or equal to the density of the sparse nodes. Ross et al.[13] also proposed a similar scheme by solving the discretized NLP problem on a grid with a certain number of nodes and then propagate the solution from the prescribed initial condition by integrating the dynamics of the system for a specified interval of time. The values of the integrated states at the end of the integration interval are taken as the initial condition for solving the NLP problem for the rest of the trajectory, again on a grid with a fixed number of nodes. The whole process is repeated until the terminal conditions are met.

In this paper, we present two algorithms, based on the Multiresolution Trajectory Optimization Algorithm (MTOA),[14,15] that autonomously discretize the trajectory with more nodes (finer grid) near the current time (not necessarily uniformly placed) and use fewer nodes (coarser grid) for the rest of the trajectory, the latter to capture the overall trend. Furthermore, if the states or controls are irregular in the vicinity of the current time, the algorithm will automatically further refine the mesh in this region to capture the irregularities in the solution more accurately. The generated grid is fully adaptive and can embrace any form depending on the solution.

The paper is organized as follows. We first formulate the trajectory optimization problem and discretize the continuous optimal control problem into an NLP problem. Next, we introduce two sequential trajectory optimization schemes for solving problems with moving targets and/or a dynamically changing environment. In due course of the paper, several challenging and practical examples are studied to demonstrate the efficacy of the proposed algorithms.

II. Problem Formulation

We wish to determine the state $\mathbf{x}(\cdot)$ and the control $\mathbf{u}(\cdot)$ that minimize the Bolza cost functional,

$$J = e(\mathbf{x}(\tau_f), \tau_f) + \int_{\tau_0}^{\tau_f} L(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau)\mathrm{d}\tau, \tag{1}$$

where $e : \mathbb{R}^{N_x} \times \mathbb{R}_+ \to \mathbb{R}$, $\tau \in [\tau_0, \tau_f]$, $\mathbf{x} : [\tau_0, \tau_f] \to \mathbb{R}^{N_x}$, $\mathbf{u} : [\tau_0, \tau_f] \to \mathbb{R}^{N_u}$, $L : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times [\tau_0, \tau_f] \to \mathbb{R}$, subject to the state dynamics

$$\dot{\mathbf{x}}(\tau) = \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau), \tag{2}$$

the state and control constraints

$$\mathbf{C}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) \leq 0, \tag{3}$$

where $\mathbf{C} : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times [\tau_0, \tau_f] \to \mathbb{R}^{N_c}$, the initial condition

$$\mathbf{x}(\tau_0) = \mathbf{x}_0, \tag{4}$$

and the terminal constraint

$$\mathbf{e}_f(\mathbf{x}(\tau_f), \tau_f) = 0, \tag{5}$$

where $\mathbf{e}_f : \mathbb{R}^{N_x} \times [\tau_0, \infty) \to \mathbb{R}^{N_e}$. The initial time τ_0 is assumed to be given and the final time τ_f can be fixed or free.

Note that the functions $\mathbf{C}$ and $\mathbf{e}_f$ are assumed to be given at time t_0, but may change as the vehicle moves from $\mathbf{x}_0$ to $\mathbf{x}(\tau_f)$. This change is *not known a priori* so it cannot be modeled via the explicit time-dependence of $\mathbf{C}$ and $\mathbf{e}_f$ in (3) and (5).

Next, we briefly describe the procedure for transcribing the continuous optimal control problem into an NLP problem. For more details, the reader is referred to Refs. [14, 15].

III. NLP Formulation: Discretizations on Dyadic Grids

Consider a set of dyadic grids of the form

$$\mathcal{V}_j = \{t_{j,k} \in [0, 1] : t_{j,k} = k/2^j,\ 0 \leq k \leq 2^j\}, \tag{6}$$

$J_{\min} \leq j \leq J_{\max}$, where j denotes the resolution level, k the spatial location, and $J_{\min},\ J_{\max} \in \{1, 2, \ldots\}$. We denote by $\mathcal{W}_j$ the set of grid points belonging to $\mathcal{V}_{j+1} \setminus \mathcal{V}_j$. Therefore,

$$\mathcal{W}_j = \{\hat{t}_{j,k} \in [0, 1] : \hat{t}_{j,k} = (2k + 1)/2^{j+1},\ 0 \leq k \leq 2^j - 1\}, \tag{7}$$

$J_{\min} \le j \le J_{\max} - 1$. Hence, $t_{j+1,k} \in \mathcal{V}_{j+1}$ is given by

$$t_{j+1,k} = \begin{cases} t_{j,k/2}, & \text{if } k \text{ is even,} \\ \hat{t}_{j,(k-1)/2}, & \text{otherwise.} \end{cases} \tag{8}$$

The subspaces $\mathcal{V}_j$ are nested

$$\mathcal{V}_{J_{\min}} \subset \mathcal{V}_{J_{\min}+1} \cdots \subset \mathcal{V}_{J_{\max}},$$

with $\lim_{J_{\max}\to\infty} \overline{\mathcal{V}_{J_{\max}}} = [0,1]$. The sequence of subspaces $\mathcal{W}_j$ satisfy

$$\mathcal{W}_j \cap \mathcal{W}_\ell = \varnothing, \quad j \neq \ell.$$

For simplicity, we denote $\mathbf{x}$ and $\mathbf{u}$ evaluated at $t_{j,k}$ by $\mathbf{x}_{j,k}$ and $\mathbf{u}_{j,k}$ respectively.

Since we will be working on dyadic grids, we first express the trajectory optimization problem stated in Section II on the unit interval $t \in [0,1]$ in terms of the new independent variable t using the following transformation,

$$\tau = t\,\Delta\tau + \tau_0, \tag{9}$$

where $\Delta\tau = \tau_f - \tau_0$.

We then convert the above mentioned optimal control problem into an NLP problem using a Runge-Kutta (RK) discretization.[14, 15] To this end, let a nonuniform grid of the form

$$\begin{aligned} \mathsf{G} = \{t_{j_i,k_i} : t_{j_i,k_i} \in [0,1],\ 0 \le k_i \le 2^{j_i},\ J_{\min} \le j_i \le J_{\max},\ \text{for } i = 0,\dots,N, \\ \text{and } t_{j_i,k_i} < t_{j_{i+1},k_{i+1}},\ \text{for } i = 0,\dots,N-1\}. \end{aligned} \tag{10}$$

A q-stage RK method for discretizing the state dynamics is given by

$$\mathbf{x}_{j_{i+1},k_{i+1}} = \mathbf{x}_{j_i,k_i} + h_{j_i,k_i}\Delta\tau \sum_{\ell=1}^{q} \beta^\ell \mathbf{f}^\ell_{j_i,k_i}, \tag{11}$$

where $\mathbf{f}^\ell_{j_i,k_i} = \mathbf{f}(\mathbf{y}^\ell_{j_i,k_i}, \mathbf{u}^\ell_{j_i,k_i}, t^\ell_{j_i,k_i})$, $\mathbf{y}^\ell_{j_i,k_i}$, $\mathbf{u}^\ell_{j_i,k_i}$, $t^\ell_{j_i,k_i}$ are the intermediate state, control, and time variables on the interval $[t_{j_i,k_i}, t_{j_{i+1},k_{i+1}}]$, given by

$$\mathbf{y}^\ell_{j_i,k_i} = \mathbf{x}_{j_i,k_i} + h_{j_i,k_i}\Delta\tau \sum_{m=1}^{q} \alpha^{\ell,m} \mathbf{f}^m_{j_i,k_i}, \tag{12}$$

where $h_{j_i,k_i} = t_{j_{i+1},k_{i+1}} - t_{j_i,k_i}$, $t^\ell_{j_i,k_i} = t_{j_i,k_i} + h_{j_i,k_i}\rho^\ell$, $\mathbf{u}^\ell_{j_i,k_i} = \mathbf{u}(t^\ell_{j_i,k_i})$, for $1 \le \ell \le q$, and q is referred to as the *stage*. In these expressions $\rho^\ell, \beta^\ell, \alpha^{\ell,m}$ are known constants with $0 \le \rho^1 \le \rho^2 \le \cdots \le 1$. The scheme is explicit if $\alpha^{\ell,m} = 0$ for $m \ge \ell$ and implicit otherwise. The cost functional is discretized by introducing a new state and then using a RK discretization as mentioned above. Then, the subsequent NLP problem is to find the variables $\mathbf{X},\ \mathbf{U},\ \tilde{\mathbf{U}},\ \tau_f$, that minimize

$$J = e(\mathbf{x}_{j_{N_t},k_{N_t}}, \tau_f) + \Delta\tau \sum_{i=0}^{N_t-1} \left(h_{j_i,k_i} \sum_{\ell=1}^{q} \beta^\ell L^\ell_{j_i,k_i} \right), \tag{13}$$

subject to the following constraints

$$\begin{aligned} \zeta_i &= 0, \quad i = 1,\dots,N_t-1, && (14)\\ \mathbf{x}_{j_0,k_0} &= \mathbf{x}_0, && (15)\\ \mathbf{e}_f(\mathbf{x}_{j_{N_t},k_{N_t}}, \tau_f) &= 0, && (16)\\ \mathbf{C}(\mathbf{X}, \tilde{\mathbf{X}}, \mathbf{U}, \tilde{\mathbf{U}}, \mathsf{G}, \tilde{\mathsf{G}}) &\le 0, && (17) \end{aligned}$$

where

$$\zeta_i = \mathbf{x}_{j_{i+1},k_{i+1}} - \mathbf{x}_{j_i,k_i} - h_{j_i,k_i}\Delta\tau \sum_{\ell=1}^{q} \beta^\ell \mathbf{f}^\ell_{j_i,k_i}, \qquad i = 0,\dots,N_t-1, \tag{18}$$

$$
\begin{aligned}
L^{\ell}_{j_i,k_i} &= L(\mathbf{y}^{\ell}_{j_i,k_i}, \mathbf{u}^{\ell}_{j_i,k_i}, t^{\ell}_{j_i,k_i}), \qquad i = 0,\dots,N_t-1, \\
\mathbf{X} &= \{\mathbf{x}_{j_0,k_0},\dots,\mathbf{x}_{j_{N_t},k_{N_t}}\}, \\
\mathbf{U} &= \{\mathbf{u}_{j_0,k_0},\dots,\mathbf{u}_{j_{N_t},k_{N_t}}\}, \\
\tilde{\mathsf{G}} &= \{t^{\ell}_{j_i,k_i} \in [0,1] : t^{\ell}_{j_i,k_i} \notin \mathsf{G},\ 0 \le i < N_t,\ 1 \le \ell \le q\}, \\
\tilde{\mathbf{X}} &= \{\mathbf{y}^{\ell}_{j_i,k_i} : t^{\ell}_{j_i,k_i} \in \tilde{\mathsf{G}}\}, \\
\tilde{\mathbf{U}} &= \{\mathbf{u}^{\ell}_{j_i,k_i} : t^{\ell}_{j_i,k_i} \in \tilde{\mathsf{G}}\}.
\end{aligned} \tag{19}
$$

If the optimal control problem does not have any constraints, then by RK discretization we mean RK discretizations that satisfy the conditions in Ref. [16]. If the optimal control problem has only pure control constraints, then by RK discretizations we mean RK discretizations that satisfy the Hager conditions;[16] alternatively, the coefficients of the RK scheme satisfy the conditions given in Ref. [17]. If the optimal control problem has state or mixed state/control constraints, then by RK discretizations we mean either Euler, Trapezoidal, or Hermite-Simpson discretization. The restriction to the above mentioned schemes stems from the fact that the convergence of these schemes for the optimal control problem has been demonstrated in the literature.[16–20]

We are now ready to present the proposed sequential trajectory optimization schemes.

IV. Sequential Trajectory Optimization

Consider a set of dyadic grids $\mathcal{V}_j$ and $\mathcal{W}_j$ as described in Eqs. (6) and (7). Suppose $g : \overline{\mathcal{D}} \to \mathbb{R}$ is specified on a grid G (given by (10)),

$$
\mathcal{U} = \{g_{j,k} : t_{j,k} \in \mathsf{G}\}, \tag{20}
$$

where $g_{j,k} = g(t_{j,k})$. By $\mathcal{I}^p(t; \mathcal{T}_{\mathsf{G}}(t))$ we denote the p-th order essentially nonoscillatory (ENO) interpolation of $\mathcal{U} = \{g_{j,k} : t_{j,k} \in \mathcal{T}_{\mathsf{G}}(t)\}$, where $\mathcal{T}_{\mathsf{G}}(t) = \{t_{j_\ell,k_\ell}\}_{\ell=i}^{i+p} \subseteq \mathsf{G}$, $0 \le i \le N-p-1$. The stencil $\mathcal{T}_{\mathsf{G}}(t)$ consists of one neighboring point on the left of t and one neighboring point on the right of t in the set G, with the remaining $p-1$ points selected from the set G that result in the least oscillatory polynomial. For more details on ENO interpolations the reader is referred to Refs. [21–23].

In order to solve an optimal control problem with a moving target and/or a dynamically changing environment, in this paper we present two sequential trajectory optimization algorithms. The basic idea behind the proposed algorithms is to solve the trajectory optimization problem at hand over the horizon $[\tau_0^1, \tau_f^1]$, and as we continue to move forward in time, we re-solve the optimization problem again on the new horizons $[\tau_0^i, \tau_f^i]$, where $i = 2,\dots,N_H$, using the solution of the previous horizon as an initial guess. Here $\tau_0^1 = \tau_0$, $\tau_0^{i-1} < \tau_0^i < \tau_f^{i-1}$, $i = 2,\dots,N_H$, and N_H is the number of horizons. If the final time is fixed, then

$$
\tau_f^1 = \tau_f^2 = \cdots = \tau_f^{N_H} = \tau_f. \tag{21}
$$

For further analysis, let

$$
\Delta\tau^i_{\mathrm{ro}} = \tau_0^{i+1} - \tau_0^i, \qquad i = 1,\dots,N_H-1, \tag{22}
$$

be the time interval after which we re-optimize the trajectory. The value of $\Delta\tau^i_{\mathrm{ro}}$ can be the same or different for all $i = 1,\dots,N_H-1$. For the case when $\Delta\tau^i_{\mathrm{ro}}$ are all the same for $i = 1,\dots,N_H-1$, that is, $\tau^i_{\mathrm{ro}} = \tau_{\mathrm{ro}}$, for all $i = 1,\dots,N_H-1$, and the final time is fixed, the number of horizons is given by

$$
N_H = \lfloor(\tau_f - \tau_0)/\Delta\tau_{\mathrm{ro}})\rfloor. \tag{23}
$$

Next, we present the sequential trajectory optimization algorithm STOA I.

IV.A. Sequential Trajectory Optimization Algorithm I (STOA I)

We first choose the minimum resolution level $J_{\min}$ based on the minimum time step required to achieve the desired accuracy in the regions of the solution where no constraints are active[a], the threshold $\epsilon(t)$ (the significance of which will be clear shortly), and the maximum resolution level $J_{\max}$. Then the proposed STOA

[a]The minimum time step required to achieve a desired accuracy in the regions of the solution where no constraints are active can be calculated using the well-known error estimation formulas for RK schemes.[16, 20, 24, 25]

I involves the following steps. First, we transcribe the continuous trajectory optimization problem into an NLP problem using a q-stage RK discretization as described in the previous section. We use trapezoidal discretization for the first iteration and switch to a high-order discretization for subsequent iterations. Next, we set $i = 1$, $\texttt{iter} = 1$, initialize $\mathsf{Grid}^i_{\text{iter}} = \mathcal{V}_{J_{\min}}$, and choose an initial guess for all NLP variables. Let us denote the set of initial guesses by $\mathcal{X}^i_{\text{iter}}$. The proposed sequential trajectory optimization algorithm then proceeds as follows:

STOA I

Step 1. Solve the NLP problem on $\mathsf{Grid}^i_{\text{iter}}$ with the initial guess $\mathcal{X}^i_{\text{iter}}$ on the horizon $[\tau^i_0, \tau^i_f]$. If $\mathsf{Grid}^i_{\text{iter}}$ has points from the level $\mathcal{W}_{J_{\max}-1}$, go to Step 4.

Step 2. **Mesh refinement**.

(a) i. If the problem either has pure state constraints or mixed constraints on states and controls, set $\Phi^i_{\text{iter}} = \{\mathbf{x}_{j,k}, \mathbf{u}_{j,k} : t_{j,k} \in \mathsf{Grid}^i_{\text{iter}}\}$, $N_r = N_x + N_u$.

ii. If the optimal control problem does not have any constraints or only pure control constraints are present, set $\Phi^i_{\text{iter}} = \{\mathbf{u}_{j,k} : t_{j,k} \in \mathsf{Grid}_{\text{iter}}\}$, $N_r = N_u$.

iii. In case no controls are present in the problem, set $\Phi^i_{\text{iter}} = \{\mathbf{x}_{j,k} : t_{j,k} \in \mathsf{Grid}^i_{\text{iter}}\}$

In the following, let Φ^i_{iter} denote the set constructed in Step 2a of the algorithm, that is, let $\Phi^i_{\text{iter}} = \{(\phi_\ell)_{j,k} : \ell = 1, \ldots, N_r,\ t_{j,k} \in \mathsf{Grid}_{\text{iter}}\}$.

(b) Initialize an intermediate grid $\mathsf{Grid}_{\text{int}} = \mathcal{V}_{J_{\min}-1}$, with function values

$$\Phi_{\text{int}} = \{(\phi_\ell)_{J_{\min},k} : (\phi_\ell)_{J_{\min},k} \in \Phi^i_{\text{iter}},\ \forall\, t_{J_{\min},k} \in \mathcal{V}_{J_{\min}},\ \ell = 1, \ldots, N_r\}, \tag{24}$$

and set $j = J_{\min} - 1$.

i. Find the points that belong to the intersection of $\mathcal{W}_j$ and $\mathsf{Grid}^i_{\text{iter}}$

$$\hat{T}_j = \{\hat{t}_{j,k_m} : \hat{t}_{j,k_m} \in \mathcal{W}_j \cap \mathsf{Grid}^i_{\text{iter}},\ \text{for } m = 1, \ldots, N_{\hat{t}},\ 1 \le N_{\hat{t}} \le 2^j - 1\}. \tag{25}$$

If $\hat{T}_j$ is empty go to Step 2c otherwise go to the next step.

ii. Set $m = 1$.

A. Compute the interpolated function values at point $\hat{t}_{j,k_m} \in \hat{T}_j$, $\hat{\phi}_\ell(\hat{t}_{j,k_m}) = \mathcal{I}^p(\hat{t}_{j,k_m}, \mathcal{T}_{\mathsf{Grid}_{\text{int}}}(\hat{t}_{j,k_m}))$, where $\hat{\phi}_\ell$ is the ℓ-th element of $\hat{\phi}$, for $\ell = 1, \ldots, N_r$.

B. Calculate the interpolative error coefficient d_{j,k_m} at the point $\hat{t}_{j,k_m}$,[b]

$$d_{j,k_m}(\phi) = \max_{\ell=1,\ldots,N_r} d_{j,k_m}(\phi_\ell) = \max_{\ell=1,\ldots,N_r} |\phi_\ell(\hat{t}_{j,k_m}) - \hat{\phi}_\ell(\hat{t}_{j,k_m})|. \tag{26}$$

If the value of d_{j,k_m} is below the threshold $\epsilon(\hat{t}_{j,k_m})$, then reject $\hat{t}_{j,k_m}$ and go to Step 2(b)iiF, otherwise add $\hat{t}_{j,k_m}$ to the intermediate grid $\mathsf{Grid}_{\text{int}}$ and move on to the next step.

C. Add to $\mathsf{Grid}_{\text{int}}$ N_{neigh} points on the left and N_{neigh} points on the right of the point $\hat{t}_{j,k_m}$ in $\mathcal{W}_j$.

D. If $N_{\text{neigh}} = 0$ add to $\mathsf{Grid}_{\text{int}}$ points belonging to the set

$$(\mathcal{V}_{\hat{J}} \cap [t_{j,k_m}, t_{j,k_m+1}]) \setminus \mathsf{Grid}_{\text{int}},$$

else add to $\mathsf{Grid}_{\text{int}}$ points belonging to the set

$$(\mathcal{V}_{\hat{J}} \cap [\hat{t}_{j,k_m-N_{\text{neigh}}}, \hat{t}_{j,k_m+N_{\text{neigh}}}]) \setminus \mathsf{Grid}_{\text{int}}.$$

Here $\hat{J} = \min\{j + \hat{j}, J_{\max}\}$, where $\hat{j} = 2$ if $\texttt{iter} = 1$ else $\hat{j} \ge 2$, $\hat{j}$ is the number of finer levels from which the points be added to the grid for refinement.

E. Add the function values at all the newly added points to Φ_{int}. If the function value at any of the newly added points is not known, we interpolate the function value at that point from the points in $\mathsf{Grid}^i_{\text{iter}}$ and their function values in Φ^i_{iter} using $\mathcal{I}^p(\cdot, \mathcal{T}_{\mathsf{Grid}^i_{\text{iter}}}(\cdot))$.

[b]Note that $\phi_\ell(\hat{t}_{j,k}) \in \Phi^i_{\text{iter}}$ for all $\hat{t}_{j,k} \in \hat{T}_j$ and $\ell = 1, \ldots, N_r$.

F. Increment m by 1. If $m \leq N_{\hat{t}}$ go to Step 2(b)iiA, otherwise move on to the next step.

iii. Set $j = j + 1$. If $j < J_{\max}$ go to Step 2(b)i, otherwise go to Step 2c.

(c) Terminate. The final nonuniform grid is $\mathsf{Grid}_{\text{new}} = \mathsf{Grid}_{\text{int}}$ and the corresponding function values are in the set $\Phi_{\text{new}} = \Phi_{\text{int}}$.

Step 3. Set $\texttt{iter} = \texttt{iter} + 1$. If the number of points and the level of resolution remain the same after the mesh refinement procedure, terminate. Otherwise, interpolate the NLP solution found in Step 1 on the new mesh $\mathsf{Grid}_{\text{new}}$, which will be the new initial guess $\mathcal{X}^i_{\text{iter}}$. Reassign the set $\mathsf{Grid}^i_{\text{iter}}$ to $\mathsf{Grid}_{\text{new}}$, and go to Step 1.

Step 4. New horizon:

(a) Set $\mathsf{Grid}^i = \mathsf{Grid}^i_{\text{iter}}$.

(b) Increment i by 1.

(c) Set $\tau^i_0 = \tau^{i-1}_0 + \Delta\tau^{i-1}_{\text{ro}}$.

(d) Terminate if $\tau^i_0 \geq \tau^{i-1}_f$, otherwise set $\texttt{iter} = 1$, $\mathsf{Grid}^i_{\text{iter}} = \mathcal{V}_{J_{\min}}$.

(e) Interpolate the solution of the previous horizon $[\tau^{i-1}_0, \tau^{i-1}_f]$ given on Grid^{i-1} to $\mathsf{Grid}^i_{\text{iter}}$, which will be our new initial guess $\mathcal{X}^i_{\text{iter}}$ for Step 1[c].

(f) Update information about the path constraints and the terminal constraints.

Step 5. Go to Step 1.

Remark 1. *Although the STOA I will work for any form of $\epsilon(t)$, we recommend using on each horizon H_i, $i = 1, \ldots, N_H$, the following expression,*

$$\epsilon(t) = \hat{\epsilon} E(\max\{0, t - \alpha\}), \tag{27}$$

where

$$\alpha = \begin{cases} t^{i+1}_0, & i = 1, \ldots, N_H - 1, \\ t^i_f, & i = N_H, \end{cases} \tag{28}$$

$\hat{\epsilon}$ be at least of order $h_{J_{\min}} = 1/2^{J_{\min}}$,[15] and $E : [0,1] \to \mathbb{R}^+$ is such that $E(0) = 1$. For example, one may choose $E(t) = e^{\beta(\max\{0, t-\alpha\})}$, where $\beta \in \mathbb{R}^+$, for $t \in [0,1]$. This choice implies that the threshold is constant, is equal to $\hat{\epsilon}$ for $t \in [0, \alpha]$, and it varies with time for $t \in (\alpha, 1]$. Such a choice stems from the fact that the solution should be calculated with high precision till the initial time of the next horizon.

Remark 2. *One should note that in STOA I each horizon $H_i = [\tau^i_0, \tau^i_f]$, $i = 1, \ldots, N_H$, is mapped to $[0,1]$ for discretizing the optimal control problem into NLP problem and hence the mesh refinement Step 2 is given on the transformed domain $[0,1]$.*

We demonstrate the above algorithm with the help of a simple, yet practical example, in which the terminal condition is assumed to be changing with time.

Example 1

Consider the Zermelo's problem taken from Ref. [8]. A ship must travel through a region of strong currents. The equations of motion of the ship are

$$\begin{aligned} \dot{x} &= V\cos\theta + u(x,y), & (29) \\ \dot{y} &= V\sin\theta + v(x,y), & (30) \end{aligned}$$

where θ is the heading angle of the ship's axis relative to the (fixed) coordinate axes, (x, y) represent the position of the ship, V is the magnitude of the ship's velocity relative to the water, and (u, v) are the

[c]It should be noted that although $\mathsf{Grid}^i_1 = \mathsf{Grid}^{i-1}_1$ on the transformed domain $[0,1]$ but both the grids Grid^{i-1}_1 and Grid^i_1 correspond to different time intervals, that is, $[\tau^{i-1}_0, \tau^{i-1}_f]$ and $[\tau^i_0, \tau^i_f]$ respectively.

velocity components of the current in the x and y directions, respectively. The magnitude and direction of the currents are assumed to be,

$$u = -Vy, \qquad v = 0, \tag{31}$$

and the ship's velocity V is assumed to be unity. The path constraint is the width of the river, and we assume

$$0 \leq x \leq 6.8. \tag{32}$$

The problem is to steer the ship in such a way so as to minimize the time necessary to go from a given point A to another given point B. For this specific example, we assume the coordinates of point A to be

$$x_{\mathrm{A}} = x(0) = 0, \qquad y_{\mathrm{A}} = y(0) = -4. \tag{33}$$

The target B is assumed to be moving. However, the trajectory of point B is not known in advance. Initially, the coordinates of B are taken to be as follows

$$x_{\mathrm{B}} = x(\tau_f) = 6, \qquad y_{\mathrm{B}} = y(\tau_f) = 1. \tag{34}$$

We assume (Step 4f of STOA I) that the information about the target is updated every time before the re-optimization is done on a new horizon. We also assume that the trajectory of the target is given by

$$x_{\mathrm{B}}(\tau) = 6 - 0.1\tau, \qquad y_{\mathrm{B}}(\tau) = 1 - 0.2\tau. \tag{35}$$

Hence, on each horizon H_i, where $i = 2, \ldots, N_H$, we have the following terminal constraints,

$$x(\tau_f^i) = 6 - 0.1\tau_0^i, \qquad x(\tau_f^i) = 1 - 0.2\tau_0^i. \tag{36}$$

For the sake of simplicity, and so that the proposed algorithm terminates in a finite number of iterations, we assume that if $\tau_0^i \geq 5$, for some $i \in [1, N_H]$, then

$$x(\tau_f^m) = 6 - 0.1\tau_0^i, \qquad y(\tau_f^m) = 1 - 0.2\tau_0^i, \tag{37}$$

for all $m = i, \ldots, N_H$.

We solved this problem on a grid with $J_{\min} = 2$ and $J_{\max} = 7$ for each horizon with

$$\epsilon(t) = 0.01\, e^{10 \max\{0, t-\alpha\}}, \quad i = 1, \ldots, N_H, \tag{38}$$

where α as in (28). The other parameters used in the simulation are $p = 3$ and $N_{\text{neigh}} = 0$. A fourth-order implicit Hermite-Simpson scheme[14] was used as a high-order scheme for discretizing the continuous optimal control problem into an NLP problem.

To solve this problem, we let $\Delta\tau_{\text{ro}}^i \approx 1\,\text{sec}$ ($i = 1, \ldots, N_H - 1$). One way for finding the initial conditions $(x(\tau_0^i), y(\tau_0^i))$ for the next horizon (H_i) is to integrate the dynamics of the system using the control found on the previous horizon (H_{i-1}) for a duration of $\Delta\tau_{\text{ro}}^i$ seconds and then use the integrated states at the end of the interval $[\tau_0^{i-1}, \tau_0^i]$ as the initial conditions for solving the NLP problem on the new horizon (H_i). For this example, we picked the initial time τ_0^i for each horizon H_i, $i = 1, \ldots, N_H$, as follows. For the first horizon we set $\tau_0^1 = 0$ and for subsequent horizons we choose

$$\tau_0^i = \min_{\tau}\{\tau \in \mathsf{Grid}_\tau^{i-1} : \tau \geq \tau_0^{i-1} + 0.95\}, \tag{39}$$

where $i = 2, \ldots, N_H$,

$$\mathsf{Grid}_\tau^{i-1} = \{\tau : \tau = (\tau_f^{i-1} - \tau_0^{i-1})t_{j,k} + \tau_0^{i-1}, \ \forall\, t_{j,k} \in \mathsf{Grid}^{i-1}\}. \tag{40}$$

The algorithm terminated after solving the problem on 6 horizons. The number of iterations taken by the algorithm before the algorithm terminated on each horizon ($\texttt{iter}_f$), the maximum resolution level reached on each horizon (J_f), the number of nodes used by the algorithm at the final iteration on each horizon (N_f), along with the initial and the final times for all the horizons are shown in Table 1.

The computed trajectory found using the proposed algorithm, along with the grid point distributions for different horizons are shown in Figures 1 and 2. In these figures, the initial point A is depicted by a square and the target point B is depicted by a cross. As pointed out earlier, the target B is assumed to

Table 1. Example 1. Target snapshots.

Horizon	$\mathtt{iter}_f$	J_f	N_f	τ_0	τ_f
H_1	3	4	9	0	5.6018
H_2	3	4	9	1.0503	5.5198
H_3	4	5	13	2.1677	5.4687
H_4	6	7	17	3.1993	5.4965
H_5	2	3	7	4.2043	5.5818
H_6	1	2	5	5.2374	5.7538

be non-stationary, and for convenience of the reader, in Figures 1, 2 all the previous locations of B are also shown in addition to the current position of target B. The optimal controls found for all the horizons are shown in Figure 3. From Figures 1(a), 3(a), we see that the proposed algorithm used only 9 points out of 129 points of the grid $\mathcal{V}_7$ for solving the given problem on the first horizon $[0, \tau_f]$. The grid point distribution 1(b) shows that the points from the finer resolution levels $\mathcal{V}_3$, $\mathcal{V}_4$ are concentrated only near the initial time. On the second horizon, we assume that the target B has moved to the new location. From Figures 1(c), 1(d), and 3(b), we again find that the algorithm used only 9 points for discretizing the trajectory and the points from the finer levels of resolution $\mathcal{V}_3$, $\mathcal{V}_4$ are again clustered near the current time. For the third horizon, the algorithm used 13 points to find the optimal solution. From the grid point distribution in Figure 1(f), it is evident that the algorithm started adding points from the finer resolution level, $\mathcal{V}_5$, near the location where there should be a switching in the control, since the ship is approaching the shore. Moving on to the fourth horizon, we see that, as the boat is approaching the shore, there should be a switching in the control. Hence, in order to capture this control switching, the algorithm further added points at the finer resolution levels $\mathcal{V}_6$, and $\mathcal{V}_7$, as can be observed from the grid point distribution for the fourth horizon (Figure 2(b)). For the fifth and sixth horizons, the algorithm used only 7 and 5 points respectively for computing the optimal solution. Since on the sixth horizon, we had $\tau_0^6 > 5$, the target was further assumed to be stationary located at

$$x(\tau_f^m) = 6 - 0.1\tau_0^6, \qquad y(\tau_f^m) = 1 - 0.2\tau_0^6, \tag{41}$$

for all $m = 6, \ldots, N_H$. Hence, the algorithm terminated after solving the problem on the sixth horizon. The overall CPU time taken by STOA I to solve this problem was 5.1 seconds. The combined trajectory and the control found on different horizons is shown in Figure 4.

Next, we incorporate the information of the trajectory profile of the target (35) in the optimal control problem itself. Since the trajectory profile of the target is assumed to be given for the optimal control problem at hand, the resulting problem can be solved in one go using MTOA.[14,15] The results found using MTOA are shown in Figure 4 and the overall CPU time taken by MTOA to solve this problem was 9.5 seconds. The minimum time (τ_f) to steer the ship from point A to the target point B found using MTOA is $\tau_f = 5.8637$. We also solved the same problem using STOA I. For comparison purposes the results found using STOA I are again shown in Figure 4. The number of iterations taken by the algorithm before the algorithm terminated on each horizon ($\mathtt{iter}_f$), the maximum resolution level reached on each horizon (J_f), the number of nodes used by the algorithm at the final iteration on each horizon (N_f), along with the initial and the final times for all the horizons are shown in Table 2. The overall CPU time to solve the problem using STOA I was 6.3 seconds. Hence, we see that the cost found by solving the problem using MTOA is less by 5^{-4} than the cost found using STOA I for the problem when the trajectory profile of the target is assumed to be known. However, we see that the overall CPU time taken by STOA I is about two-thirds of the overall CPU time taken by MTOA to solve the same problem.

IV.B. Sequential Trajectory Optimization Algorithm II (STOA II)

In this section, we present yet another sequential trajectory optimization scheme referred to as STOA II, which takes full advantage of the multiresolution structure of the grid in the mesh refinement procedure so that the previously computed information is retained, while moving from one horizon to the next. In order to avoid notational complexities, and without loss of generality, we will assume in this section that the time interval of interest is the unit interval $t \in [0, 1] = [\tau_0, \tau_f]$. Transformation (9) can be used to convert any optimal control problem from the domain $[\tau_0, \tau_f]$ to $[0, 1]$.

Table 2. Example 1. Target trajectory known.

Horizon	$\texttt{iter}_f$	J_f	N_f	τ_0	τ_f
H_1	3	4	9	0	5.9065
H_2	3	4	9	1.1075	5.8256
H_3	3	4	11	2.2870	5.8648
H_4	6	7	17	3.4051	5.8643
H_5	1	2	5	4.4810	5.8642
H_6	1	2	5	5.5184	5.8642

We choose the parameters $J_{\min}$, $J_{\max}$, and $\epsilon(t)$ as for the STOA I. Then the proposed STOA II involves the following steps. First, we transcribe the continuous trajectory optimization problem into an NLP problem using a q-stage RK discretization as described in the previous section. We use trapezoidal discretization for the first iteration and switch to a high-order discretization for subsequent iterations. Next, we set $i = 1$, $\texttt{iter} = 1$, $t_0^i = 0$, initialize $\mathsf{Grid}^i_{\text{iter}} = \mathcal{V}_{J_{\min}}$, and choose an initial guess for all NLP variables ($\mathcal{X}^i_{\text{iter}}$). Fix $\bar{J} = J_{\min} - 1$. The proposed sequential trajectory optimization algorithm proceeds as follows:

Step 1. Solve the NLP problem on $\mathsf{Grid}^i_{\text{iter}}$ with the initial guess $\mathcal{X}^i_{\text{iter}}$ on the horizon $[t_0^i, 1]$. If $\mathsf{Grid}^i_{\text{iter}}$ has points from the level $\mathcal{W}_{J_{\max}-1}$, go to Step 4.

Step 2. Find $\mathsf{Grid}_{\text{new}}$ using the mesh refinement step (Step 2) of STOA I.

Step 3. Set $\texttt{iter} = \texttt{iter} + 1$. If the number of points and the level of resolution remain the same after the mesh refinement procedure then terminate, otherwise interpolate the NLP solution found in Step 1 on the new mesh $\mathsf{Grid}_{\text{new}}$, which will be our new initial guess $\mathcal{X}^i_{\text{iter}}$, reassign the set $\mathsf{Grid}^i_{\text{iter}}$ to $\mathsf{Grid}_{\text{new}}$, and go to Step 1.

Step 4. New horizon.

(a) Set $\mathsf{Grid}^i = \mathsf{Grid}^i_{\text{iter}}$.

(b) Increment i by 1.

(c) Set $t_0^i = t_{\bar{J},i-1}$.

(d) If $i = 2^{\bar{J}} + 1$ terminate, else go to the next step.

(e) Set $\mathsf{Grid}^{i-} = \{t : t \in \mathsf{Grid}^{i-1} \text{ and } t \geq t_{\bar{J},i-1}\}$.

(f) If the number of points in the set $\{\mathsf{Grid}^{i-} \cap \mathcal{V}_{J_{\min}-1}\}$ is less than $p+1$, set $J_{\min} = J_{\min} + 1$.

(g) Set $\texttt{iter} = 1$, $\mathcal{V}_j = \mathcal{V}_j \setminus (\mathcal{V}_j \cap [0, t_{\bar{J},i-1}))$ (where $j = J_{\min} - 1, \ldots, J_{\max}$), and $\mathcal{W}_j = \mathcal{W}_j \setminus (\mathcal{W}_j \cap [0, t_{\bar{J},i-1}))$ (where $j = J_{\min} - 1, \ldots, J_{\max} - 1$). Find $\mathsf{Grid}_{\text{new}}$ using the mesh refinement step (Step 2) of STOA I with $\mathsf{Grid}^i_{\text{iter}} = \mathsf{Grid}^{i-}$.

(h) Increment $\texttt{iter}$ by 1 and reassign the set $\mathsf{Grid}^i_{\text{iter}}$ to $\mathsf{Grid}_{\text{new}}$.

(i) Interpolate the NLP solution given on Grid^{i-} to $\mathsf{Grid}^i_{\text{iter}}$, which will be our new initial guess $\mathcal{X}^i_{\text{iter}}$ for Step 1.

(j) Update the information about the path constraints and the terminal constraints.

Step 5. Go to Step 1.

Remark 3. *Although STOA II will work for any form of the threshold $\epsilon(t)$, we recommend choosing, on each horizon, H_i, $i = 1, \ldots, N_H$,*

$$\epsilon(t) = \hat{\epsilon} E(\max\{0, t - t_{\bar{J},i}\}), \tag{42}$$

where $\hat{\epsilon}$ is at least of order $h_{J_{\min}} = 1/2^{J_{\min}}$,[15] and $E : [0,1] \to \mathbb{R}^+$ such that $E(0) = 1$, $t \in [0,1]$. This choice implies that the threshold is constant and is equal to $\hat{\epsilon}$ for $t \in [0, t_{\bar{J},i}]$ and varies with time for $t \in (t_{\bar{J},i}, 1]$. Such a choice stems from the fact that the solution should be calculated with high precision till the initial time of the next horizon, which in this case would be $t_{\bar{J},i}$.

Example 2
In this example, we consider the re-entry guidance problem of an Apollo-type vehicle taken from Ref. [26]. The equations of motion during the flight of the vehicle through the Earth's atmosphere are as follows:

$$\begin{aligned}
\dot{v} &= -\frac{S}{2m}\rho v^2 c_{\mathrm{D}}(u) - \frac{g\sin\gamma}{(1+\xi)^2},\\
\dot{\gamma} &= \frac{S}{2m}\rho v c_{\mathrm{L}}(u) + \frac{v\cos\gamma}{R(1+\xi)} - \frac{g\cos\gamma}{v(1+\xi)^2},\\
\dot{\xi} &= \frac{v}{R}\sin\gamma,\\
\dot{\zeta} &= \frac{v}{1+\xi}\cos\gamma,
\end{aligned}$$

where v is the velocity, γ is the flight path angle, $\xi = h/R$ is the normalized altitude, h is the altitude above the Earth's surface, R is the Earth's radius, and ζ is the distance on the Earth's surface of a trajectory of an Apollo-type vehicle. The control variable is the angle of attack u. For the lift and drag the following relations hold:

$$c_{\mathrm{D}} = c_{\mathrm{D}_0} + c_{\mathrm{DL}}\cos u, \quad c_{\mathrm{D}_0} = 0.88, \quad c_{\mathrm{DL}} = 0.52, \tag{43}$$

$$c_{\mathrm{L}} = c_{\mathrm{L}_0}\sin u, \quad c_{\mathrm{L}_0} = -0.505. \tag{44}$$

The air density is assumed to satisfy

$$\rho = \rho_0 e^{-\beta R\xi}. \tag{45}$$

The values of the constants are

$$\begin{aligned}
R &= 209.0352\ (10^5\ \mathrm{ft}),\\
S/m &= 50,000\ (10^{-5}\ \mathrm{ft}^2\ \mathrm{slug}^{-1}),\\
\rho_0 &= 2.3769\times 10^{-3}(\mathrm{slug}\ \mathrm{ft}^{-3}),\\
g &= 3.2172\times 10^{-4}\ (10^5\ \mathrm{ft}\ \mathrm{s}^{-2}),\\
\beta &= 1/0.235\ (10^{-5}\ \mathrm{ft}^{-1}).
\end{aligned}$$

The cost functional to be minimized that describes the total stagnation point convective heating per unit area is given by the integral

$$J(u) = \int_0^{\tau_f} 10 v^3\sqrt{\rho}\,\mathrm{d}\tau. \tag{46}$$

The vehicle is to be maneuvered into an initial position favorable for the final splashdown in the Pacific. The data at the moment of entry are

$$v(0) = 0.35\ (10^5\ \mathrm{ft\ s}^{-1}), \qquad \gamma(0) = -5.75\ \mathrm{deg}, \tag{47}$$

$$\xi(0) = 4/R\ (h(0) = 400,000\ \mathrm{ft}), \qquad \zeta(0) = 0\ (10^5\ \mathrm{ft}). \tag{48}$$

The data prescribed at the unspecified terminal time t_f for this problem are

$$v(\tau_f) = 0.0165\ (10^5\ \mathrm{ft\ s}^{-1}), \qquad \gamma(\tau_f)\ \text{unspecified}, \tag{49}$$

$$\xi(\tau_f) = 0.75530/R\ (h(t_f) = 75530\ \mathrm{ft}), \qquad \zeta(\tau_f) = 51.6912\ (10^5\ \mathrm{ft}). \tag{50}$$

The angle of attack is constrained to be between ± 68 deg, that is,

$$|u| \le 68\,\mathrm{deg}\,. \tag{51}$$

We have used STOA II to solve this problem with $J_{\min} = 4$, and $J_{\max} = 7$. The threshold used for this problem was

$$\epsilon(t) = 0.01 e^{7\max\{0, t - t_{3,i}\}}, \quad i = 1, \ldots, N_H. \tag{52}$$

The other parameters used in the simulation for the mesh refinement step were $p = 3$ and $N_{\text{neigh}} = 1$. A fourth-order implicit Hermite-Simpson scheme[14] was used as a high-order scheme for discretizing the continuous optimal control problem into an NLP problem. The algorithm terminated after solving the problem on 8 horizons and the overall CPU time taken by the algorithm was 41.2 seconds, out of which 22 seconds were used to compute the solution on the first horizon H_1. For sake of brevity, we only show the time histories of the control u, along with the grid point distribution for different horizons, in Figures 5, 6, and 7. The number of iterations taken by the algorithm before the algorithm terminated on each horizon ($\mathtt{iter}_f$), the maximum resolution level reached on each horizon (J_f), and the number of nodes used by the algorithm at the final iteration on each horizon (N_f) are shown in Table 1.

Table 3. Example 2.

Horizon	$\mathtt{iter}_f$	J_f	N_f
H_1	2	5	24
H_2	2	7	27
H_3	1	7	24
H_4	1	4	11
H_5	1	4	9
H_6	1	4	7
H_7	3	7	17
H_8	1	7	13

IV.C. STOA I vs. STOA II

Both STOA I and STOA II have their own merits. STOA I will work for any user-specified time intervals ($\Delta\tau_{\rm ro}$), whereas the time intervals in STOA II are dyadic and fixed. On the other hand, STOA II takes full advantage of the multiresolution structure of the grid in the mesh refinement procedure. Most of the nodes in the grid for the new horizon are the nodes from the grid of the previous horizon. In STOA II most of the points of Grid_1^i consist of the points belonging to $\mathsf{Grid}^{i-} \subset \mathsf{Grid}^{i-1}$, for which the solution is already known. Hence, none of the previously computed information is lost while going from one horizon to the next. Therefore, in order to provide an initial guess $\mathcal{X}_1^i$, $i = 2, \ldots, N_H$, for starting the NLP solver on horizon H_i, the function values only at few additional points in the vicinity of the current time need to be interpolated from the solution found on the grid Grid^{i-1} during the previous horizon H_{i-1}. Moreover, in STOA I the algorithm always begins to iterate from the coarsest grid $\mathcal{V}_{J_{\min}}$. In STOA II, since most of the points of Grid_1^i consist of the points belonging to Grid^{i-}, the algorithm need not necessarily start from the coarsest grid, and in fact Grid_1^i may have nodes from finer scales resulting in faster convergence.

For both STOA I and STOA II, if the path constraints and the terminal constraints do not change drastically, the algorithm for each successive horizon converges pretty fast since the solution of the previous horizon is provided as an initial guess for solving the NLP problem on the current horizon. The CPU times achieved using the current implementation show the merits of the proposed algorithms in terms of speed. We should mention at this point that since all the computations presented in this paper were carried out in MATLAB, the reported CPU times can be significantly reduced by coding the algorithms in C or FORTRAN.

V. Conclusions

In this paper, we have proposed two sequential trajectory optimization schemes to solve optimal control problems with moving targets and/or under dynamically changing environments in a fast and efficient way. The proposed algorithms autonomously discretize the trajectory with more nodes near the current time (not necessarily uniformly placed) while using a coarser grid for the rest of the trajectory in order to capture the overall trend. Moreover, if the states or the controls are irregular at a certain future time, the mesh is further refined automatically at those locations as well. The final grid point distributions for all the horizons and for both the examples considered in this paper confirm these observations. Given their simplicity and efficiency, the proposed techniques offer a potential for online implementation for solving problems with moving targets and dynamically changing environments.

Acknowledgment: Partial support for this work has been provided by NSF award no. CMS 0510259 and NASA award no. NNX08AB94A.

References

[1]Seywald, H. and Cliff, E. M., "Neighboring Optimal Control Based Feedback Law for the Advanced Launch System," *Journal of Guidance, Control, and Dynamics*, Vol. 17, 1994, pp. 1154–1162.

[2]Lu, P., "Regulation About Time-Varying Trajectories: Precision Entry Guidance Illustrated," *Journal of Guidance, Control, and Dynamics*, Vol. 22, 1999, pp. 784–790.

[3]Jardin, M. R. and Bryson, A. E., "Neighboring Optimal Aircraft Guidance in Winds," *Journal of Guidance, Control, and Dynamics*, Vol. 24, 2001, pp. 710–715.

[4]Yan, H., Fahroo, F., and Ross, I. M., "Real-Time Computation of Neighboring Optimal Control Laws," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2002, AIAA 2002-4657.

[5]Breakwell, J. V., Speyer, J. L., and Bryson, A. E., "Optimization and Control of Nonlinear Systems Using the Second Variation," *SIAM Journal on Control*, Vol. 1, 1963, pp. 193–223.

[6]Kelly, H. J., "An Optimal Guidance Approximation Theory," *IEEE Transactions on Automatic Control*, Vol. 9, 1964, pp. 375–380.

[7]Speyer, J. L. and Bryson, A. E., "A Neighboring Optimum Feedback Control Scheme Based on Estimated Time-to-Go with Application to Re-entry Flight Paths," *AIAA Journal*, Vol. 6, 1968, pp. 769–776.

[8]Bryson, A. E. and Ho, Y.-C., *Applied Optimal Control: Optimization, Estimation, and Control*, Hemisphere Publishing Corporation, Washington, 1975.

[9]Kumar, R. R. and Seywald, H., "Dense-Sparse Discretization for Optimization and Real-Time Guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 19, 1996, pp. 501–503.

[10]Ohtsuka, T., "Quasi-Newton-Type Continuation Method for Nonlinear Receding Horizon Control," *Journal of Guidance, Control, and Dynamics*, Vol. 25, 2002, pp. 685–692.

[11]Bellingham, J., Kuwata, Y., and How, J., "Stable Receding Horizon Trajectory Control for Complex Environments," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2003, AIAA 2003-5635.

[12]Williams, P., "Application of Pseudospectral Methods for Receding Horizon Control," *Journal of Guidance, Control, and Dynamics*, Vol. 27, 2004, pp. 310–314.

[13]Ross, I. M., Gong, Q., and Sekhavat, P., "Low-Thrust, High Accuracy Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 30, 2007, pp. 921–933.

[14]Jain, S. and Tsiotras, P., "Multiresolution-Based Direct Trajectory Optimization," *Proceedings of 46th IEEE Conference on Decision and Control*, New Orleans, LA, 2007, pp. 5991–5996.

[15]Jain, S. and Tsiotras, P., "Trajectory Optimization using Multiresolution Techniques," *Journal of Guidance, Control, and Dynamics*, 2008, to appear.

[16]Hager, W. W., "Runge-Kutta Methods in Optimal Control and the Transformed Adjoint System," *Numerische Mathematik*, Vol. 87, 2000, pp. 247–282.

[17]Dontchev, A. L., Hager, W. W., and Veliov, V. M., "Second-Order Runge-Kutta Approximations in Control Constrained Optimal Control," *SIAM Journal on Numerical Analysis*, Vol. 38, 2000, pp. 202–226.

[18]Dontchev, A. L. and Hager, W. W., "The Euler Approximation in State Constrained Optimal Control," *Mathematics of Computation*, Vol. 70, 2000, pp. 173–203.

[19]Betts, J. T., *Practical Methods for Optimal Control using Nonlinear Programming*, SIAM, Philadelphia, PA, 2001.

[20]Betts, J. T., Biehn, N., and Campbell, S. L., "Convergence of Nonconvergent IRK Discretizations of Optimal Control Problems with State Inequality Constraints," *SIAM Journal on Scientific Computing*, Vol. 23, 2002, pp. 1981–2007.

[21]Harten, A., "Multiresolution Representation of Data: A General Framework," *SIAM Journal on Numerical Analysis*, Vol. 33, No. 3, 1996, pp. 1205–1256.

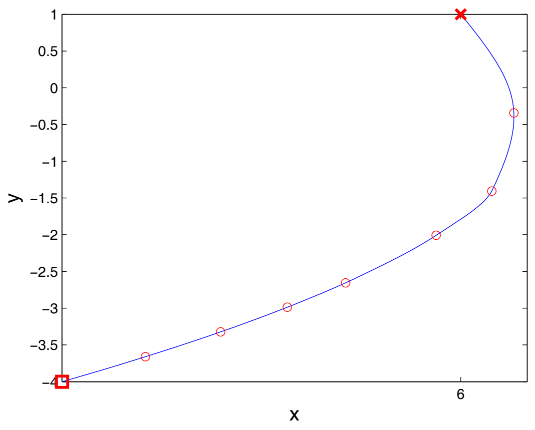
[22]Osher, S. and Fedkiw, R. P., *Level Set Methods and Dynamic Implicit Surfaces*, Springer, New York, 2003.

[23]Jain, S., Tsiotras, P., and Zhou, H.-M., "A Hierarchical Multiresolution Adaptive Mesh Refinement for the Solution of Evolution PDEs," *SIAM Journal on Scientific Computing*, 2007, Submitted.

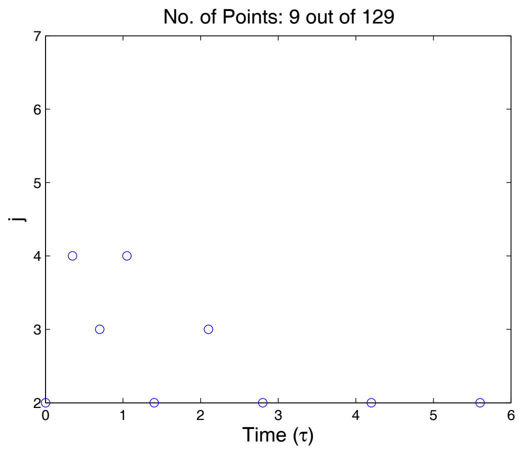
[24]Hairer, E., Norsett, S. P., and Wanner, G., *Solving Ordinary Differential Equations I. Nonstiff Problems.*, Springer-Verlag, New York, 1987.

[25]Hairer, E., Norsett, S. P., and Wanner, G., *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.*, Springer-Verlag, New York, 1991.

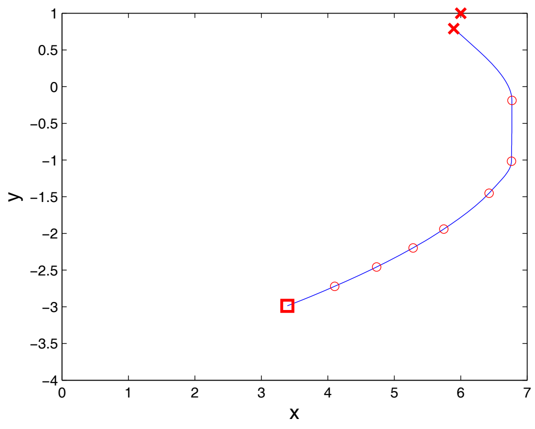
[26]Pesch, H. J., "Real-Time Computation of Feedback Controls for Constrained Optimal Control Problems. Part 2: A Correction Method Based on Multiple Shooting," *Optimal Control Applications & Methods*, Vol. 10, 1989, pp. 147–171.



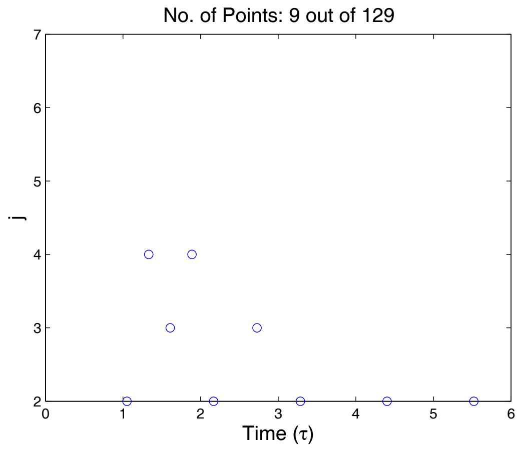
(a) Horizon 1. Trajectory.



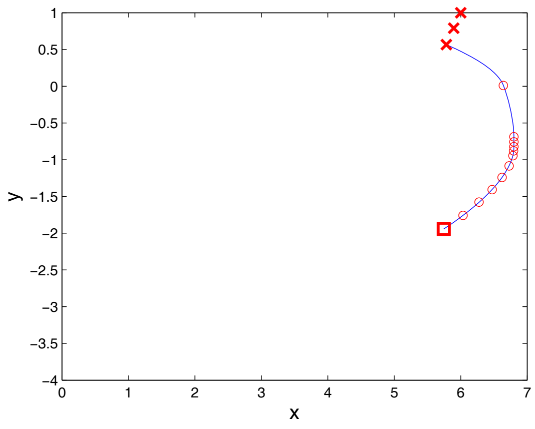
(b) Horizon 1. Grid point distribution.



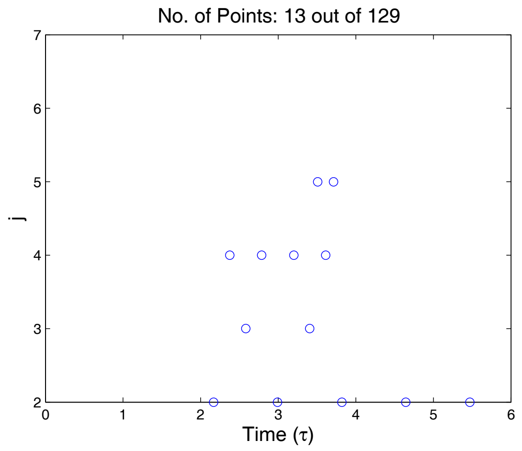
(c) Horizon 2. Trajectory.



(d) Horizon 2. Grid point distribution.

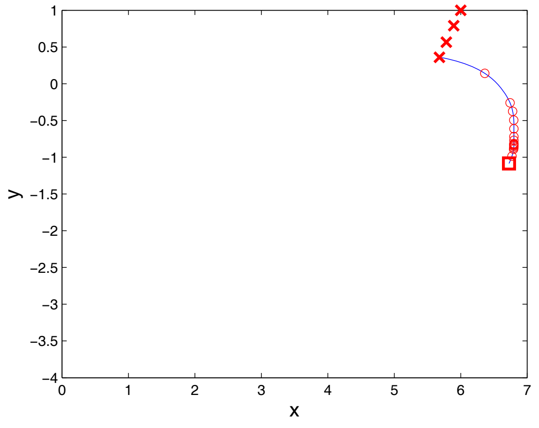


(e) Horizon 3. Trajectory.

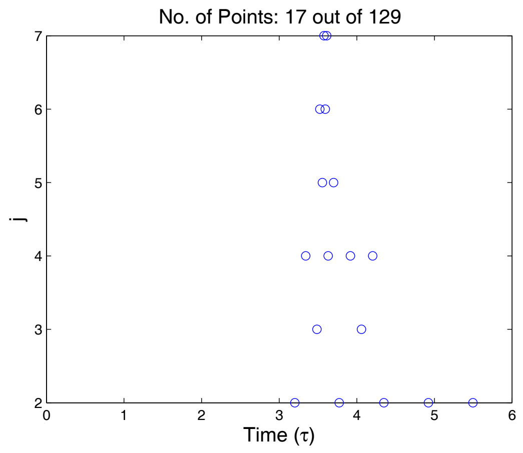


(f) Horizon 3. Grid point distribution.

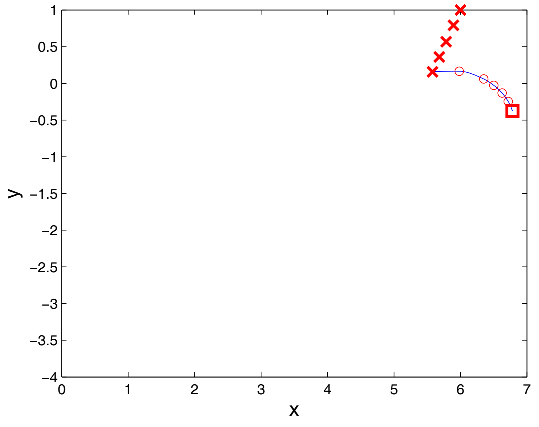
Figure 1. Example 1 (Target snapshots). Trajectory along with the grid point distributions for horizons 1, 2, and 3.



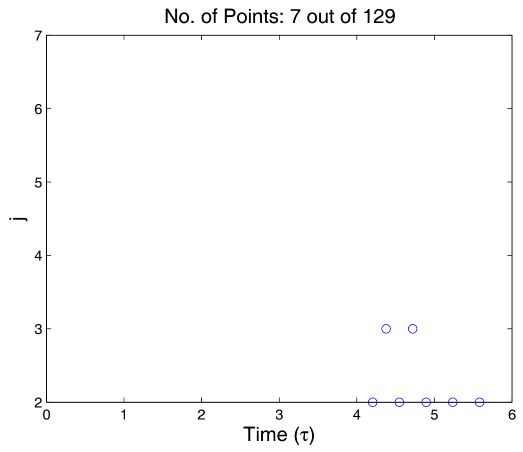
(a) Horizon 4. Trajectory.



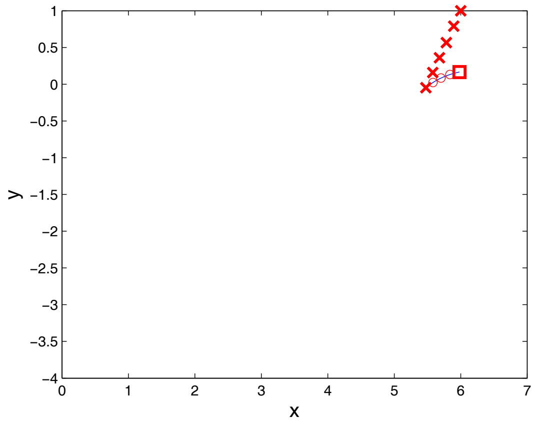
(b) Horizon 4. Grid point distribution.



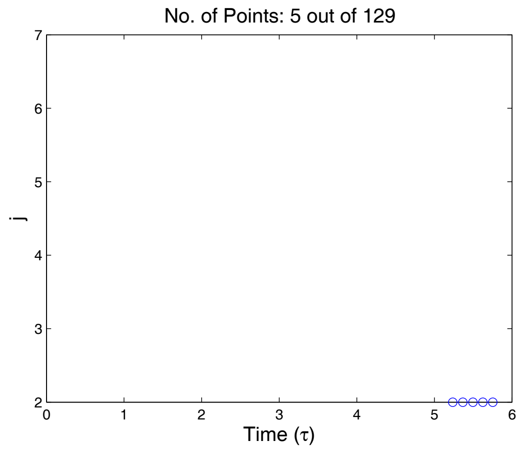
(c) Horizon 5. Trajectory.



(d) Horizon 5. Grid point distribution.

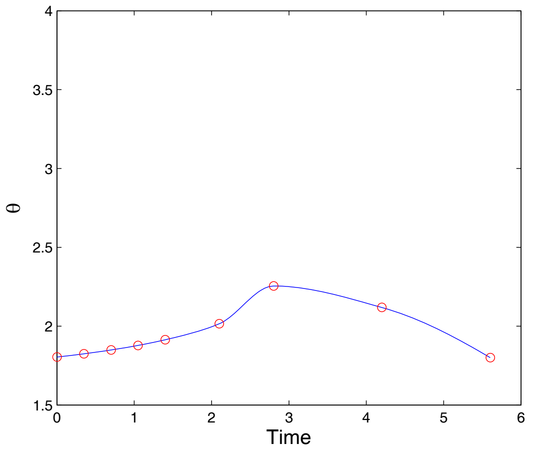


(e) Horizon 6. Trajectory.

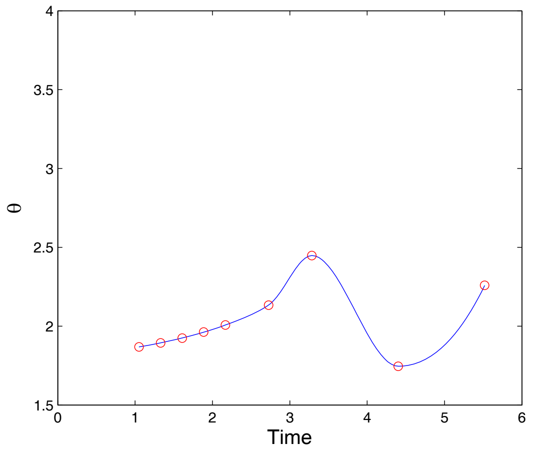


(f) Horizon 6. Grid point distribution.

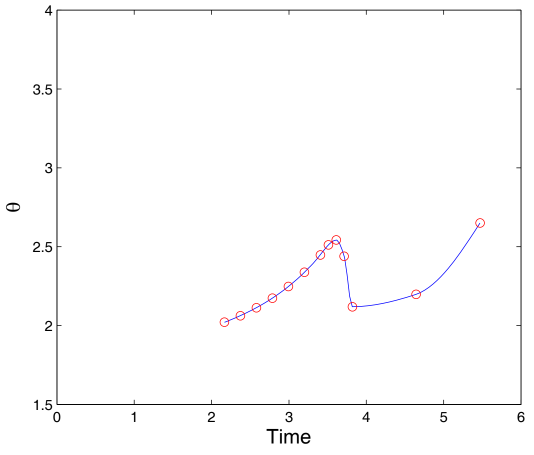
Figure 2. Example 1 (Target snapshots). Trajectory along with the grid point distributions for horizons 4, 5, and 6.



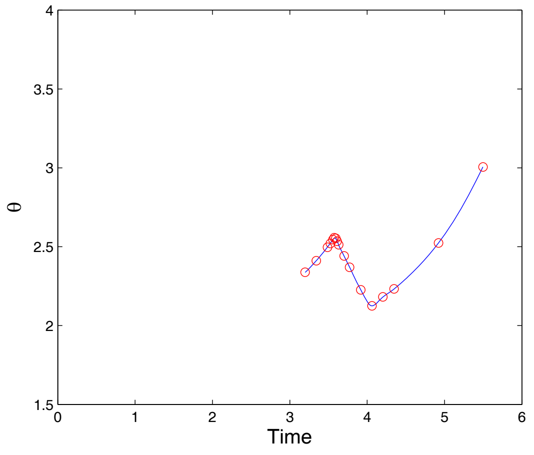
(a) Horizon 1.



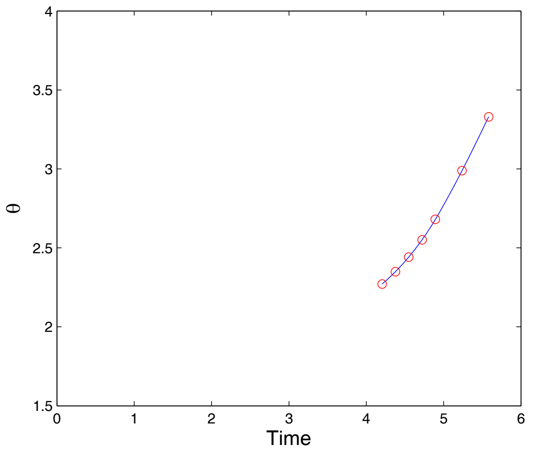
(b) Horizon 2.



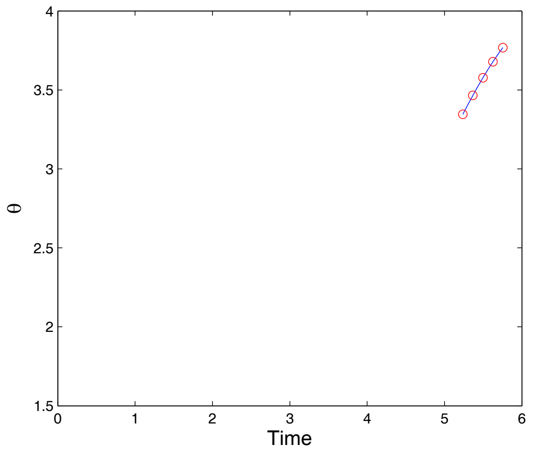
(c) Horizon 3.



(d) Horizon 4.

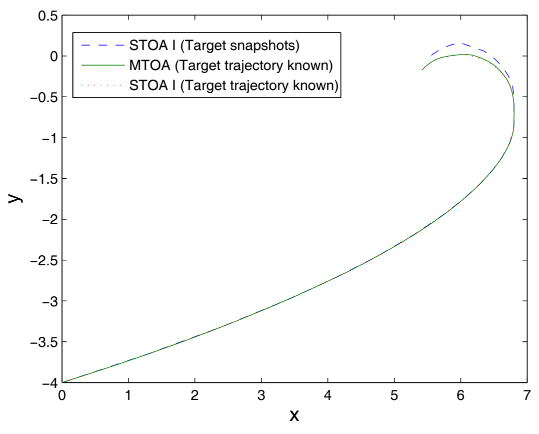


(e) Horizon 5.

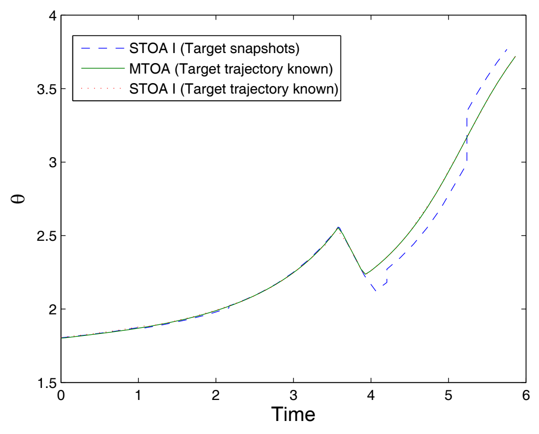


(f) Horizon 6.

Figure 3. Example 1 (Target snapshots). Time history of control θ for all horizons.

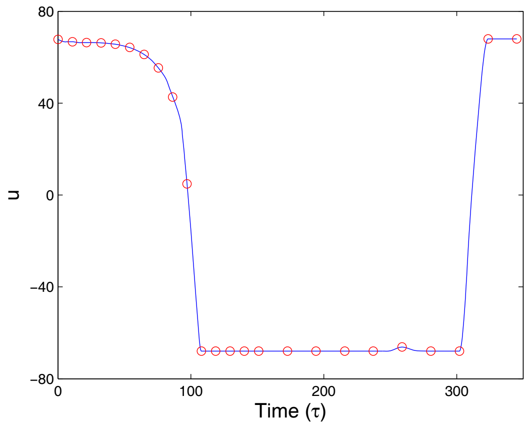


(a) Trajectory.

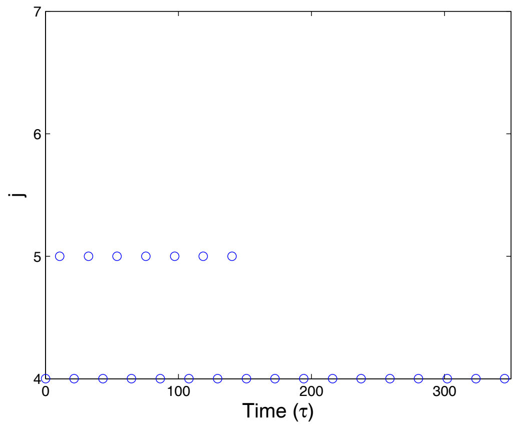


(b) Time history of control θ.

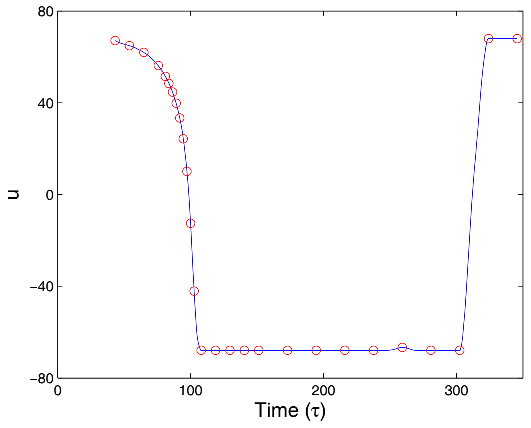
Figure 4. Example 1. Trajectory along with the time history of the control θ using three different multiresolution strategies.



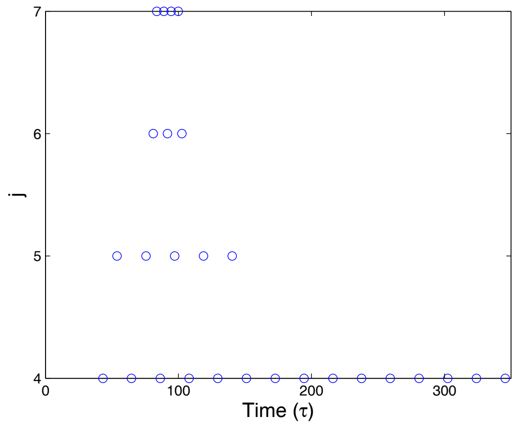
(a) Horizon 1. Control u.



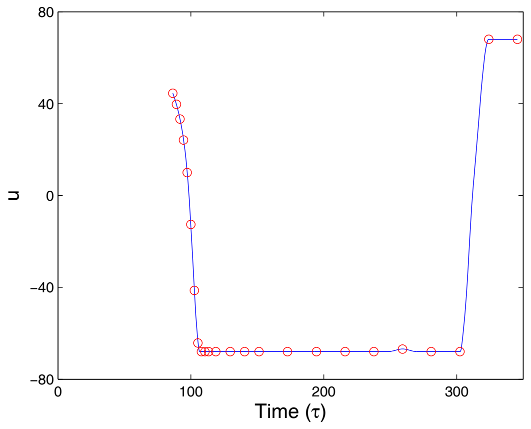
(b) Horizon 1. Grid point distribution.



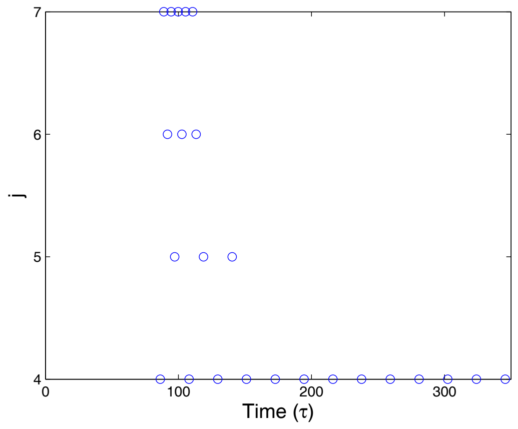
(c) Horizon 2. Control u.



(d) Horizon 2. Grid point distribution.

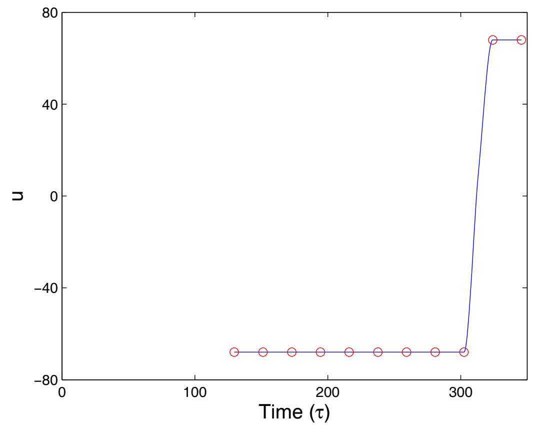


(e) Horizon 3. Control u.

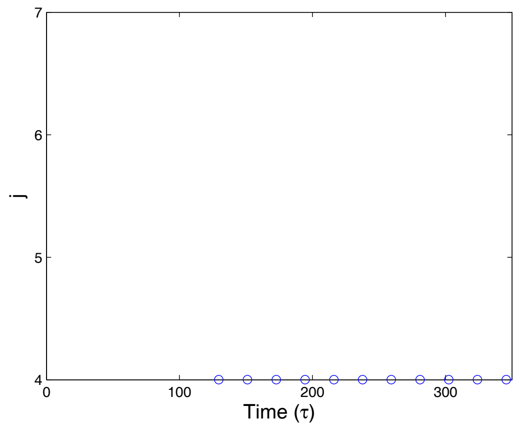


(f) Horizon 3. Grid point distribution.

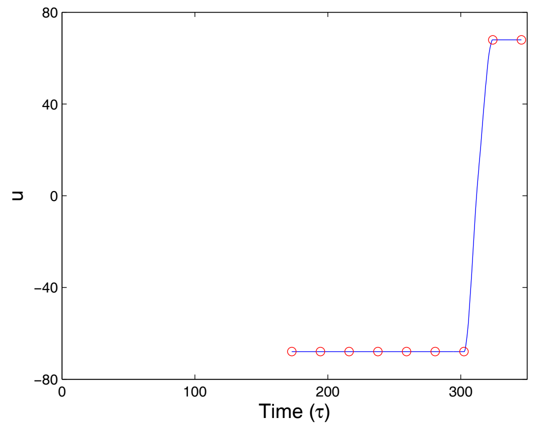
Figure 5. Example 2. Control time history and grid point distributions for horizons 1, 2, and 3.



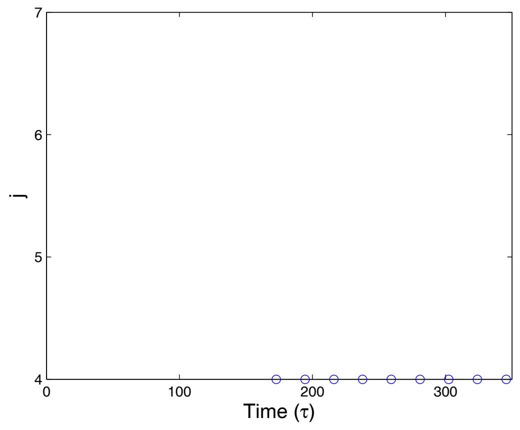
(a) Horizon 4. Control u.



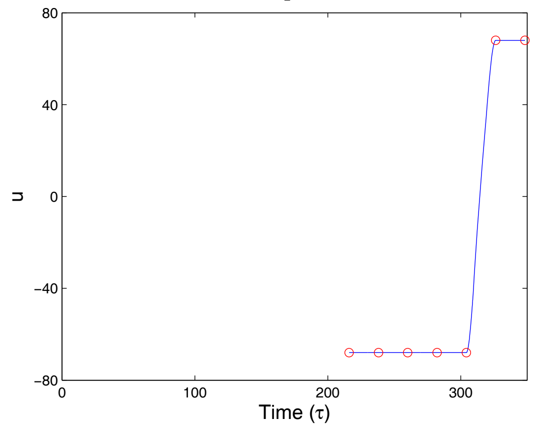
(b) Horizon 4. Grid point distribution.



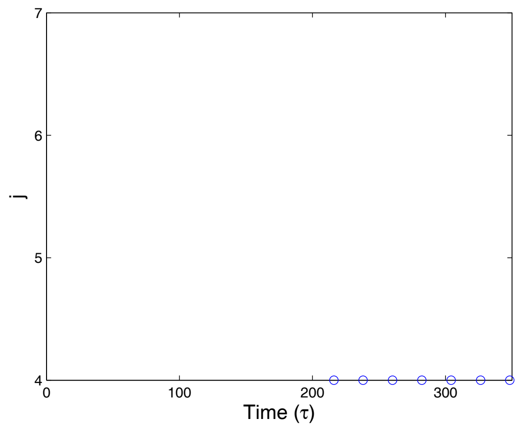
(c) Horizon 5. Control u.



(d) Horizon 5. Grid point distribution.

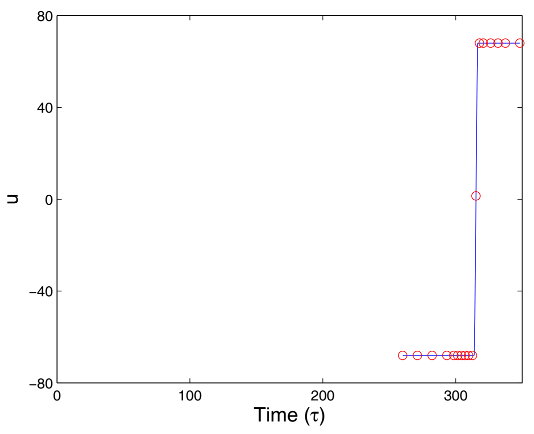


(e) Horizon 6. Control u.

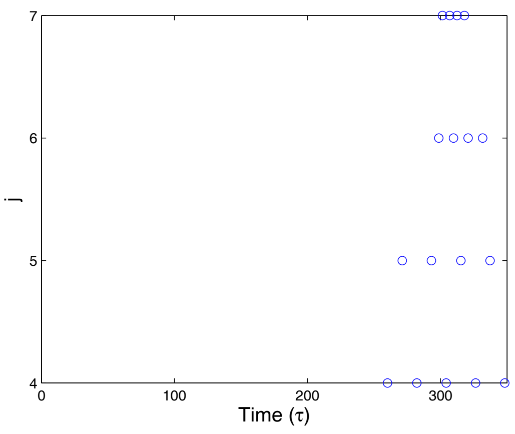


(f) Horizon 6. Grid point distribution.

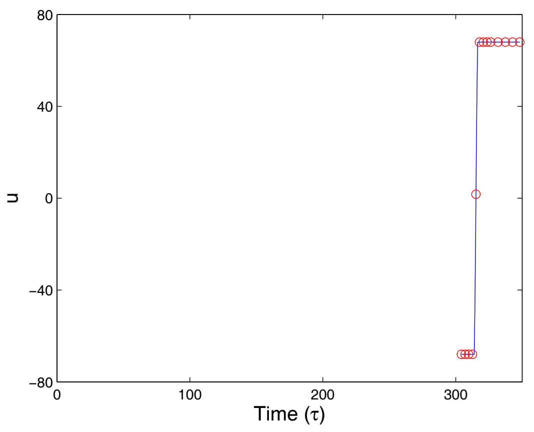
Figure 6. Example 2. Control time history and grid point distributions for horizons 4, 5, and 6.



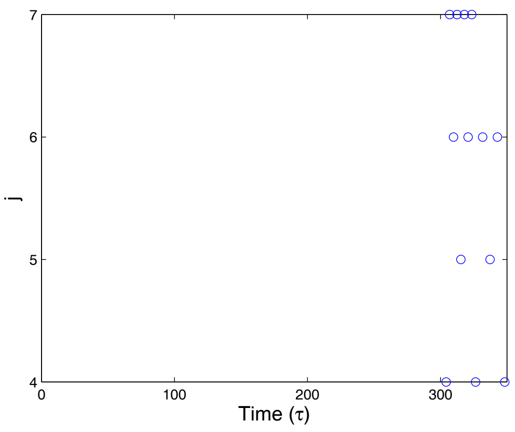
(a) Horizon 7. Control u.



(b) Horizon 7. Grid point distribution.



(c) Horizon 8. Control u.



(d) Horizon 8. Grid point distribution.

Figure 7. Example 2. Control time history and grid point distributions for horizons 7 and 8.