JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 32, No. 2, March–April 2009

Sequential Multiresolution Trajectory Optimization Schemes for Problems with Moving Targets

Sachin Jain* and Panagiotis Tsiotras†
Georgia Institute of Technology, Atlanta, Georgia 30332-0150

DOI: 10.2514/1.37899

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 algorithm is proposed, enabled by the introduction of the idea of multigrid temporal resolution to solve the associated trajectory optimization problem on a nonuniform grid across time. The grid is adapted to the immediate future and to potential discontinuities in the state and control variables.

Introduction

CONSIDER the problem of finding the optimal control that will steer a vehicle from point A to some target point B under certain path constraints with minimum cost. If the target point B is far off, then there is no real advantage to finding the optimal trajectory online with high precision from the starting point until the end. As we continue to move from point A toward 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 toward point *B*. For example, the target point B may not be stationary.

A common line of attack for solving such 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.

Another way of handling this problem is to use a receding-horizon approach [10–12]. In a receding-horizon approach, a trajectory that optimizes the cost function over a period of time, called the *planning horizon*, is designed first. 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 target B, the trajectory found using this approach might not be optimal. Hence, 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 and Seywald [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 using any ad hoc scheme. For example, it can be terminated when the density of the dense nodes is less than or equal to the density of the sparse nodes. Ross et al. [13] proposed a similar scheme by solving the discretized nonlinear programming (NLP) problem on a grid with a certain number of nodes and then propagating 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 recently proposed multiresolution trajectory optimization algorithm (MTOA) developed by the authors [14,15], which 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.

NLP Problem Formulation on Dyadic Grids

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

Presented as Paper 6980 at the AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, HI, 18–21 August 2008; received 4 April 2008; accepted for publication 12 November 2008.

*Ph.D. Candidate, School of Aerospace Engineering; sachin.jain@gatech.edu.

†Professor, School of Aerospace Engineering; tsiotras@gatech.edu. Fellow AIAA.

$$e\colon \mathbb{R}^{N_x}\times\mathbb{R}_+\to\mathbb{R},\qquad \tau\in[\tau_0,\tau_f],\qquad \mathbf{x}\colon[\tau_0,\tau_f]\to\mathbb{R}^{N_x}$$
$$\mathbf{u}\colon[\tau_0,\tau_f]\to\mathbb{R}^{N_u},\qquad L\colon \mathbb{R}^{N_x}\times\mathbb{R}^{N_u}\times[\tau_0,\tau_f]\to\mathbb{R}$$

subject to the state dynamics

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

and the state and control constraints

$$\mathbf{C}\left(\mathbf{x}(\tau),\mathbf{u}(\tau),\tau\right)\le 0 \tag{3}$$

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

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

and the terminal constraint

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

where $\mathbf{e}_f\colon \mathbb{R}^{N_x}\times\mathbb{R}_+\to\mathbb{R}^{N_e}$. 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 $\mathbf{C}(\mathbf{x},\mathbf{u},\tau)$ and $\mathbf{e}_f(\mathbf{x},\tau)$ are assumed to be given at time τ_0, but may change as the vehicle moves from $\mathbf{x}_0$ to $\mathbf{x}(\tau_f)$. This change is *not known a priori*, and so it cannot be modeled via the explicit time dependence of $\mathbf{C}(\mathbf{x},\mathbf{u},\tau)$ and $\mathbf{e}_f(\mathbf{x},\tau)$ in Eqs. (3) and (5).

All discretizations of the state dynamics, constraints, and performance index in Eqs. (1–5) will be performed on (nonuniform) grids induced by dyadic grids [14]. A uniform dyadic grid over the unit interval is a collection of points of the form

$$\mathcal{V}_j=\{t_{j,k}\in[0,1]\colon t_{j,k}=k/2^j, 0\le k\le 2^j\},\qquad J_{\min}\le j\le J_{\max} \tag{6}$$

where j denotes the resolution level, k is the spatial location, and $J_{\min}$ and $J_{\max}$ are positive integers. The set of grid points belonging to $\mathcal{V}_{j+1}\setminus\mathcal{V}_j$ is denoted by $\mathcal{W}_j$.

For simplicity, we henceforth denote $\mathbf{x}_{j,k}=\mathbf{x}(t_{j,k})$ and $\mathbf{u}_{j,k}=\mathbf{u}(t_{j,k})$. We convert the optimal control problem (1–5) into an NLP problem using Runge–Kutta (RK) discretizations [14,15]. To this end, let a nonuniform grid of the form

$$\begin{aligned}\mathsf{G}=\{&t_{j_i,k_i}\colon t_{j_i,k_i}\in[0,1],\qquad 0\le k_i\le 2^{j_i}\\ &J_{\min}\le j_i\le J_{\max}\quad \text{for } i=0,\dots,N\\ &t_{j_i,k_i}<t_{j_{i+1},k_{i+1}}\quad \text{for } i=0,\dots,N-1\}\end{aligned} \tag{7}$$

To express the trajectory optimization problem (1–5) on the unit interval $t\in[0,1]$ in terms of the new independent variable t, we use the following transformation:

$$t=\frac{\tau-\tau_0}{\tau_f-\tau_0} \tag{8}$$

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

$$\mathbf{x}_{j_{i+1},k_{i+1}}=\mathbf{x}_{j_i,k_i}+h_{j_i,k_i}\Delta\tau\sum_{\ell=1}^{q}\beta^\ell\mathbf{f}^\ell_{j_i,k_i} \tag{9}$$

where $\mathbf{f}^\ell_{j_i,k_i}=\mathbf{f}(\mathbf{y}^\ell_{j_i,k_i},\mathbf{u}^\ell_{j_i,k_i},t^\ell_{j_i,k_i})$, and $\mathbf{y}^\ell_{j_i,k_i}$, $\mathbf{u}^\ell_{j_i,k_i}$, and $t^\ell_{j_i,k_i}$ 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

$$\mathbf{y}^\ell_{j_i,k_i}=\mathbf{x}_{j_i,k_i}+h_{j_i,k_i}\Delta\tau\sum_{m=1}^{q}\alpha^{\ell,m}\mathbf{f}^m_{j_i,k_i} \tag{10}$$

where $h_{j_i,k_i}=t_{j_{i+1},k_{i+1}}-t_{j_i,k_i}$, $t^\ell_{j_i,k_i}=t_{j_i,k_i}+h_{j_i,k_i}\rho^\ell$, and $\mathbf{u}^\ell_{j_i,k_i}=\mathbf{u}(t^\ell_{j_i,k_i})$ for $1\le\ell\le q$, and q is referred to as the *stage*. In these expressions, ρ^ℓ, β^ℓ, and $\alpha^{\ell,m}$ 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 is implicit otherwise. The cost functional is discretized by introducing a new state and then using a RK discretization, as mentioned previously. Then the subsequent NLP problem [14,15] is to find the variables $\mathbf{X}$, $\mathbf{U}$, $\tilde{\mathbf{U}}$, and τ_f that minimize

$$J=e(\mathbf{x}_{j_{N_t},k_{N_t}},\tau_f)+\Delta\tau\sum_{i=0}^{N_t-1}\left(h_{j_i,k_i}\sum_{\ell=1}^{q}\beta^\ell L^\ell_{j_i,k_i}\right) \tag{11}$$

subject to the following constraints:

$$\zeta_i=0,\qquad i=1,\dots,N_t-1 \tag{12}$$

$$\mathbf{x}_{j_0,k_0}=\mathbf{x}_0 \tag{13}$$

$$\mathbf{e}_f(\mathbf{x}_{j_{N_t},k_{N_t}},\tau_f)=0 \tag{14}$$

$$\mathbf{C}\left(\mathbf{X},\tilde{\mathbf{X}},\mathbf{U},\tilde{\mathbf{U}},\mathsf{G},\tilde{\mathsf{G}}\right)\le 0 \tag{15}$$

where

$$\zeta_i=\mathbf{x}_{j_{i+1},k_{i+1}}-\mathbf{x}_{j_i,k_i}-h_{j_i,k_i}\Delta\tau\sum_{\ell=1}^{q}\beta^\ell\mathbf{f}^\ell_{j_i,k_i},\qquad i=0,\dots,N_t-1 \tag{16}$$

$$\begin{aligned}&L^\ell_{j_i,k_i}=L(\mathbf{y}^\ell_{j_i,k_i},\mathbf{u}^\ell_{j_i,k_i},t^\ell_{j_i,k_i}),\qquad i=0,\dots,N_t-1\\ &\mathbf{X}=\{\mathbf{x}_{j_0,k_0},\dots,\mathbf{x}_{j_{N_t},k_{N_t}}\},\qquad \mathbf{U}=\{\mathbf{u}_{j_0,k_0},\dots,\mathbf{u}_{j_{N_t},k_{N_t}}\}\\ &\tilde{\mathsf{G}}=\{t^\ell_{j_i,k_i}\in[0,1]\colon t^\ell_{j_i,k_i}\notin\mathsf{G}, 0\le i<N_t, 1\le\ell\le q\}\\ &\tilde{\mathbf{X}}=\{\mathbf{y}^\ell_{j_i,k_i}\colon t^\ell_{j_i,k_i}\in\tilde{\mathsf{G}}\},\qquad \tilde{\mathbf{U}}=\{\mathbf{u}^\ell_{j_i,k_i}\colon t^\ell_{j_i,k_i}\in\tilde{\mathsf{G}}\}\end{aligned} \tag{17}$$

Remark 1. For problems with pure control constraints, we use RK discretizations that satisfy the conditions in [16] or [17]. If the optimal control problem has state or mixed state/control constraints, then we use Euler, trapezoidal, or Hermite–Simpson discretization. The restriction to the aforementioned schemes stems from the fact that the convergence of these schemes for the optimal control problem has been demonstrated in the literature [16–20] (also see the relevant discussion in [14]).

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

Sequential Trajectory Optimization

Consider a set of dyadic grids $\mathcal{V}_j$ and $\mathcal{W}_j$, and suppose $g\colon[0,1]\to\mathbb{R}$ is specified on a grid G [given by Eq. (7)]:

$$\mathcal{U}=\{g_{j,k}\colon t_{j,k}\in\mathsf{G}\} \tag{18}$$

where $g_{j,k}=g(t_{j,k})$. Let $\mathcal{I}^p(t;\mathcal{T}_\mathsf{G}(t))$ denote the pth-order essentially nonoscillatory (ENO) interpolation of

$$\mathcal{U}=\{g_{j,k}\colon t_{j,k}\in\mathcal{T}_\mathsf{G}(t)\}$$

where

$$\mathcal{T}_\mathsf{G}(t)=\{t_{j_\ell,k_\ell}\}_{\ell=i}^{i+p}\subseteq\mathsf{G}$$

and $0\le i\le N-p-1$. 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 [21–23].

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 a suitably chosen horizon $[\tau_0^1, \tau_f^1]$. As we continue to move forward in time, we solve the optimization problem again on the new horizons $[\tau_0^i, \tau_f^i]$, where $i = 2, \ldots, N_H$, using the solution of the previous horizon as an initial guess. Here, $\tau_0^1 = \tau_0$ and $\tau_0^{i-1} < \tau_0^i < \tau_f^{i-1}$, where $i = 2, \ldots, N_H$, and N_H is the number of horizons. If the final time is fixed, then

$$\tau_f^1 = \tau_f^2 = \cdots = \tau_f^{N_H} = \tau_f \tag{19}$$

For further analysis, let

$$\Delta\tau_{\text{ro}}^i = \tau_0^{i+1} - \tau_0^i, \qquad i = 1, \ldots, N_H - 1 \tag{20}$$

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

$$N_H = \lfloor (\tau_f - \tau_0)/\Delta\tau_{\text{ro}} \rfloor \tag{21}$$

Next, we present the sequential trajectory optimization algorithm STOA I. In the following, and for the sake of simplicity, we denote $\mathbf{x}$ and $\mathbf{u}$ evaluated at $t_{j,k}$ by $\mathbf{x}_{j,k}$ and $\mathbf{u}_{j,k}$, respectively.

Sequential Trajectory Optimization Algorithm 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 in which no constraints are active,[‡] the threshold $\epsilon(t)$ (the significance of which will be clear shortly), and the maximum resolution level $J_{\max}$. Then the proposed sequential trajectory optimization algorithm (STOA 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 $i = 1$ and iter $= 1$, initialize $\text{grid}_{\text{iter}}^i = \mathcal{V}_{J_{\min}}$, and choose an initial guess for all NLP variables. Let us denote the set of initial guesses by $\mathcal{X}_{\text{iter}}^i$. The proposed sequential trajectory optimization algorithm then proceeds as follows:

1) Solve the NLP problem on $\text{grid}_{\text{iter}}^i$ with the initial guess $\mathcal{X}_{\text{iter}}^i$ on the horizon $[\tau_0^i, \tau_f^i]$. If $\text{grid}_{\text{iter}}^i$ has points from the level $\mathcal{W}_{J_{\max}-1}$, go to step 4.

2) Mesh refinement.

a) Initialization.

1. If the problem either has pure state constraints or mixed constraints on states and controls, set

$$\Phi_{\text{iter}}^i = \left\{ \mathbf{x}_{j,k}, \mathbf{u}_{j,k} \colon t_{j,k} \in \text{grid}_{\text{iter}}^i \right\}, \qquad N_r = N_x + N_u$$

2. If the optimal control problem does not have any constraints or only pure control constraints are present, set

$$\Phi_{\text{iter}}^i = \{ \mathbf{u}_{j,k} \colon t_{j,k} \in \text{grid}_{\text{iter}}^i \}, \qquad N_r = N_u$$

3. In case no controls are present in the problem, set

$$\Phi_{\text{iter}}^i = \{ \mathbf{x}_{j,k} \colon t_{j,k} \in \text{grid}_{\text{iter}}^i \}, \qquad N_r = N_x$$

In the following, let Φ_{iter}^i denote the set constructed in step 2a of the algorithm; that is, let

$$\Phi_{\text{iter}}^i = \{ \phi_\ell(t_{j,k}) \colon \ell = 1, \ldots, N_r, t_{j,k} \in \text{grid}_{\text{iter}}^i \}$$

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

$$\begin{aligned} \Phi_{\text{int}} = \{ \phi_\ell(t_{J_{\min},k}) \colon \phi_\ell(t_{J_{\min},k}) \in \Phi_{\text{iter}}^i \\ \forall\ t_{J_{\min},k} \in \mathcal{V}_{J_{\min}}, \quad \ell = 1, \ldots, N_r \} \end{aligned} \tag{22}$$

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

1. Find the points that belong to the intersection of $\mathcal{W}_j$ and $\text{grid}_{\text{iter}}^i$:

$$\begin{aligned} \hat{T}_j = \{ \hat{t}_{j,k_m} \colon \hat{t}_{j,k_m} \in \mathcal{W}_j \cap \text{grid}_{\text{iter}}^i, \quad \text{for } m = 1, \ldots, N_{\hat{t}} \\ 1 \le N_{\hat{t}} \le 2^j - 1 \} \end{aligned} \tag{23}$$

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

2. Set $m = 1$.

a. Compute the interpolated function values at points $\hat{t}_{j,k_m} \in \hat{T}_j$:

$$\hat{\phi}_\ell(\hat{t}_{j,k_m}) = \mathcal{I}^p(\hat{t}_{j,k_m}, \mathcal{T}_{\text{grid}_{\text{int}}}(\hat{t}_{j,k_m}))$$

where $\hat{\phi}_\ell$ is the ℓth element of $\hat{\phi}$ for $\ell = 1, \ldots, N_r$.

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

$$d_{j,k_m}(\phi) = \max_{\ell=1,\ldots,N_r} d_{j,k_m}(\phi_\ell) = \max_{\ell=1,\ldots,N_r} |\phi_\ell(\hat{t}_{j,k_m}) - \hat{\phi}_\ell(\hat{t}_{j,k_m})| \tag{24}$$

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 2b2f; otherwise, add $\hat{t}_{j,k_m}$ to the intermediate grid grid_{int} and move on to the next step.

c. To grid_{int}, add 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 $N_{\text{neigh}} = 0$, to grid_{int} add points belonging to the set

$$(\mathcal{V}_j \cap [t_{j,k_m}, t_{j,k_m+1}]) \setminus \text{grid}_{\text{int}}$$

else to grid_{int} add points belonging to the set

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

Here, $\hat{J} = \min\{j + \hat{j}, J_{\max}\}$, where $\hat{j} = 2$ if iter $= 1$, else $\hat{j} \ge 2$, and $\hat{j}$ is the number of finer levels from which the points are 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 $\text{grid}_{\text{iter}}^i$ and their function values in Φ_{iter}^i using $\mathcal{I}^p(\cdot; \mathcal{T}_{\text{grid}_{\text{iter}}^i}(\cdot))\Pi$.

f. Increment m by 1. If $m \le N_{\hat{t}}$, go to step 2b2a; otherwise, move on to the next step.

3) Set $j = j + 1$. If $j < J_{\max}$, go to step 2b1; otherwise, go to step 2c.

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

3) Set iter $=$ 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 grid_{new}, which will be the new initial guess $\mathcal{X}_{\text{iter}}^i$. Reassign the set $\text{grid}_{\text{iter}}^i$ to grid_{new}, and go to step 1.

4) New horizon.

a) Set $\text{grid}^i = \text{grid}_{\text{iter}}^i$.

b) Increment i by 1.

c) Set $\tau_0^i = \tau_0^{i-1} + \Delta\tau_{\text{ro}}^{i-1}$.

d) Terminate if $\tau_0^i \ge \tau_f^{i-1}$; otherwise, set iter $= 1$, $\text{grid}_{\text{iter}}^i = \mathcal{V}_{J_{\min}}$.

e) Interpolate the solution of the previous horizon $[\tau_0^{i-1}, \tau_f^{i-1}]$ given on grid^{i-1} to $\text{grid}_{\text{iter}}^i$, which will be our new initial guess $\mathcal{X}_{\text{iter}}^i$

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

[§]Note that $\phi_\ell(\hat{t}_{j,k}) \in \Phi_{\text{iter}}^i$ for all $\hat{t}_{j,k} \in \hat{T}_j$ and $\ell = 1, \ldots, N_r$:

for step 1. Note that $\text{grid}_1^i = \text{grid}_1^{i-1}$ on the transformed domain [0, 1], but both grid_1^{i-1} and grid_1^i correspond to different time intervals: that is, $[\tau_0^{i-1}, \tau_f^{i-1}]$ and $[\tau_0^i, \tau_f^i]$, respectively.

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

5) Go to step 1.

Remark 2. Although the STOA I will work for any form of $\epsilon(t)$, we recommend using the following expression on each horizon H_i $(i = 1, \ldots, N_H)$:

$$\epsilon(t) = \hat{\epsilon} E(\max\{0, t - \alpha\}) \tag{25}$$

where

$$\alpha = \begin{cases} t_0^{i+1}, & i = 1, \ldots, N_H - 1, \\ t_f^i, & i = N_H \end{cases} \tag{26}$$

$\hat{\epsilon}$ is at least of order $h_{J_{\min}} = 1/2^{J_{\min}}$ [14], 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, 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 until the initial time of the next horizon.

Remark 3. One should note that in STOA I, each horizon $H_i = [\tau_0^i, \tau_f^i]$ $(i = 1, \ldots, N_H)$ 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 preceding 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 Zermelo's problem taken from [8]. A ship must travel through a region of strong currents. The equations of motion of the ship are

$$\dot{x} = V \cos\theta + u(x, y) \tag{27}$$

$$\dot{y} = V \sin\theta + v(x, y) \tag{28}$$

where θ is the heading angle of the ship's axis relative to the (fixed) coordinate axes, x and y represent the position of the ship, V is the magnitude of the ship's velocity relative to the water, and u and v are the velocity components of the current in the x and y directions, respectively. The magnitude and direction of the currents are assumed to be

$$u(x, y) = -Vy, \qquad v(x, y) = 0 \tag{29}$$

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

$$0 \le x \le 6.8 \tag{30}$$

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

$$x_A = x(0) = 0, \qquad y_A = y(0) = -4 \tag{31}$$

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

$$x_B = x(\tau_f) = 6, \qquad y_B = y(\tau_f) = 1 \tag{32}$$

We assume (step 4f of STOA I) that the information about the target is updated before each reoptimization is done on a new horizon. We also assume that the trajectory of the target is given by

$$x_B(\tau) = 6 - 0.1\tau, \qquad y_B(\tau) = 1 - 0.2\tau \tag{33}$$

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

$$x_B^i = 6 - 0.1\tau_0^i, \qquad y_B^i = 1 - 0.2\tau_0^i \tag{34}$$

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

$$x_B^m = 6 - 0.1\tau_0^i, \qquad y_B^m = 1 - 0.2\tau_0^i \tag{35}$$

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

$$\epsilon(t) = 0.01 e^{10\max\{0, t-\alpha\}}, \qquad i = 1, \ldots, N_H \tag{36}$$

where α is defined in Eq. (26). The other parameters used in the simulation are $p = 3$ and $N_{\text{neigh}} = 0$. A fourth-order implicit Hermite–Simpson scheme [15] was used as a high-order scheme for discretizing the continuous-optimal-control problem into an NLP problem.

To solve this problem, we let $\Delta\tau_{\text{ro}}^i \approx 1$ s $(i = 1, \ldots, N_H - 1)$. One way to find the initial conditions $x(\tau_0^i)$ and $y(\tau_0^i)$ for the next horizon H_i is to integrate the dynamics of the system using the control found on the previous horizonH_{i-1} for a duration of $\Delta\tau_{\text{ro}}^i$ s 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

$$\tau_0^i = \min_{\tau}\{\tau \in \text{grid}_\tau^{i-1}: \tau \ge \tau_0^{i-1} + 0.95\} \tag{37}$$

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

$$\text{grid}_\tau^{i-1} = \{\tau: \tau = (\tau_f^{i-1} - \tau_0^{i-1})t_{j,k} + \tau_0^{i-1}, \quad \forall\ t_{j,k} \in \text{grid}^{i-1}\} \tag{38}$$

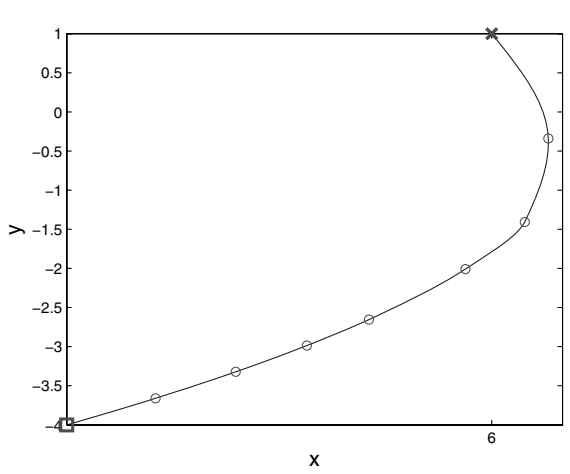
Correspondingly, the initial conditions for each horizon H_i $(i = 2, \ldots, N_H)$ were chosen to be the values of x and y at τ_0^i, which are known from the computations on the previous horizon H_{i-1}. We started STOA I with a linear initial guess. The algorithm terminated after solving the problem on 6 horizons. The number of iterations taken by the algorithm before the algorithm terminated on each horizon (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), and the initial and the final times for all the horizons are shown in Table 1.

The computed trajectory found using the proposed algorithm and the grid-point distributions for different horizons are shown in Figs. 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, target B is assumed to be nonstationary, and for the convenience of the reader, shown in Figs. 1 and 2 are all of the previous locations of point B in addition to the current position of target B. The optimal controls found for all the horizons are shown in Fig. 3.

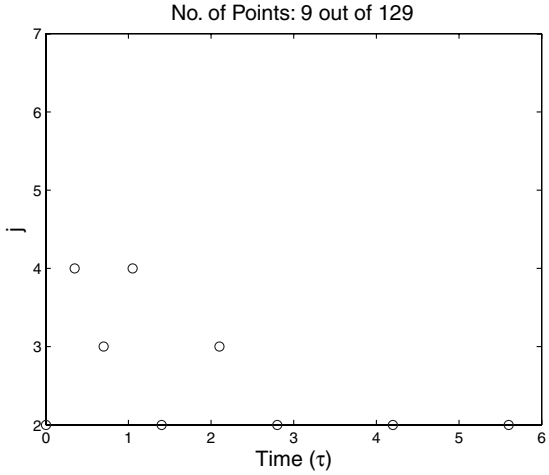
From Figs. 1a and 3a, we see that the proposed algorithm used only 9 points out of 129 points of grid $\mathcal{V}_7$ for solving the given problem on

Table 1 Example 1 (target snapshots): number of iterations on each horizon (iter_f), maximum resolution level J_f, number of nodes used by the algorithm at the final iteration on each horizon (N_f), and corresponding horizons

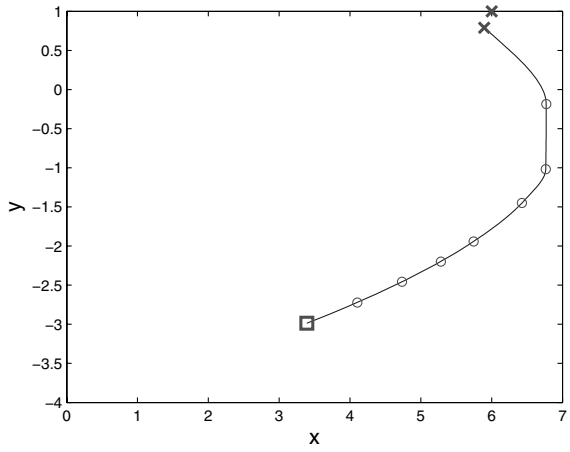
Horizon	iter_f	J_f	N_f	τ_0	τ_f
H_1	3	4	9	0	5.6018
H_2	3	4	9	1.0503	5.5198
H_3	4	5	13	2.1677	5.4687
H_4	6	7	17	3.1993	5.4965
H_5	2	3	7	4.2043	5.5818
H_6	1	2	5	5.2374	5.7538



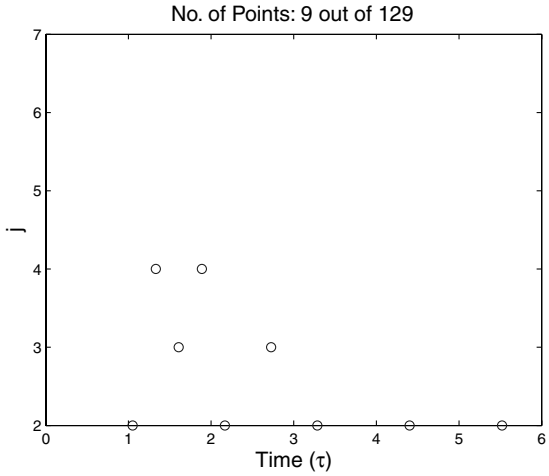
a) Horizon 1 trajectory



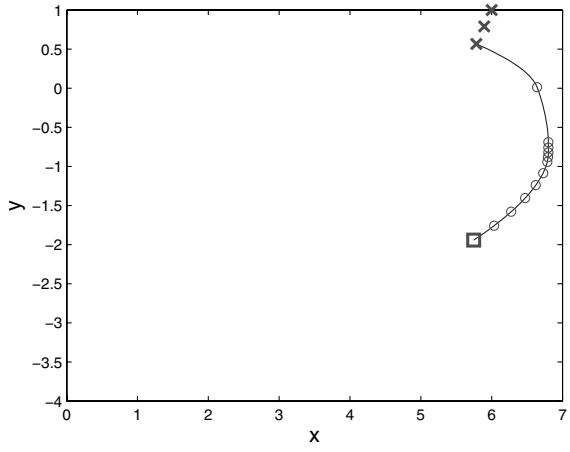
b) Horizon 1 grid point distribution



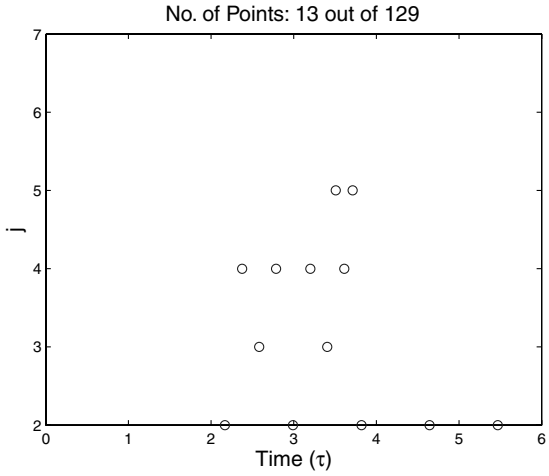
c) Horizon 2 trajectory



d) Horizon 2 grid point distribution



e) Horizon 3 trajectory



f) Horizon 3 grid point distribution

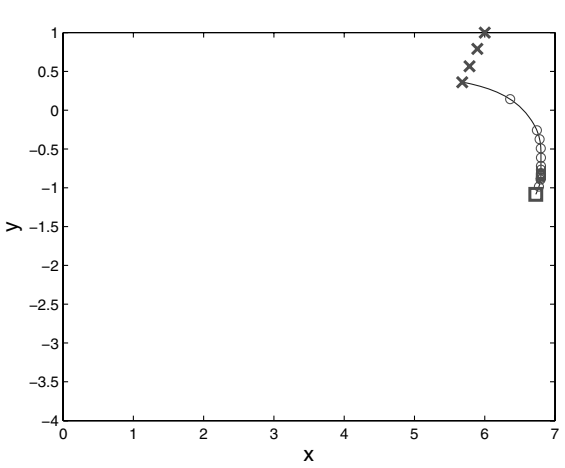
Fig. 1 Example 1 (target snapshots): trajectory and grid-point distributions for horizons 1, 2, and 3.

the first horizon $[0, \tau_f]$. The grid-point distribution in Fig. 1b shows that the points from the finer resolution levels $\mathcal{V}_3$ and $\mathcal{V}_4$ are concentrated only near the initial time.

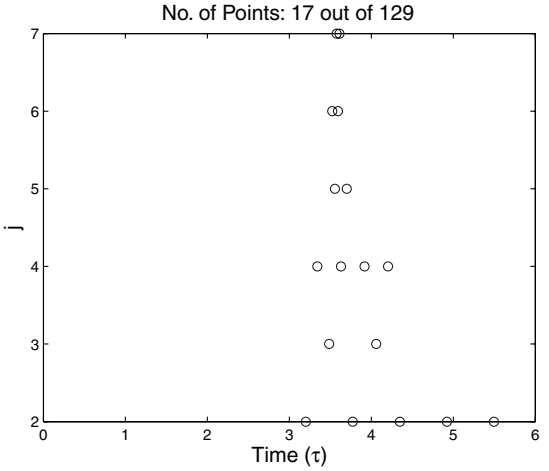
On the second horizon, we assume that target B has moved to the new location. From Figs. 1c, 1d, and 3b, we again find that the algorithm used only 9 points for discretizing the trajectory, and the points from the finer levels of resolution $\mathcal{V}_3$ and $\mathcal{V}_4$ are again clustered near the current time.

For the third horizon, the algorithm used 13 points to find the optimal solution. From the grid-point distribution in Fig. 1f, it is evident that the algorithm started adding points from the finer resolution level, $\mathcal{V}_5$, near the location at which there should be a switching in the control, because the ship is approaching the shore.

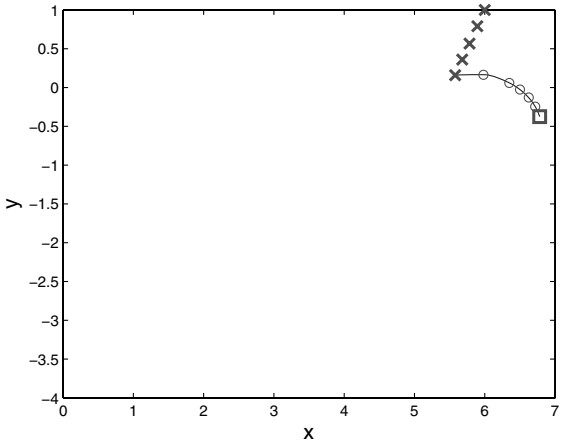
Moving on to the fourth horizon, we see that as the boat is approaching the shore there should be a switching in the control. Hence, to capture this control switching, the algorithm further added



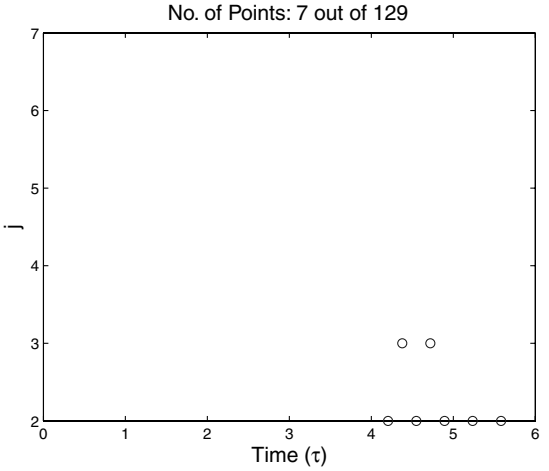
a) Horizon 4 trajectory



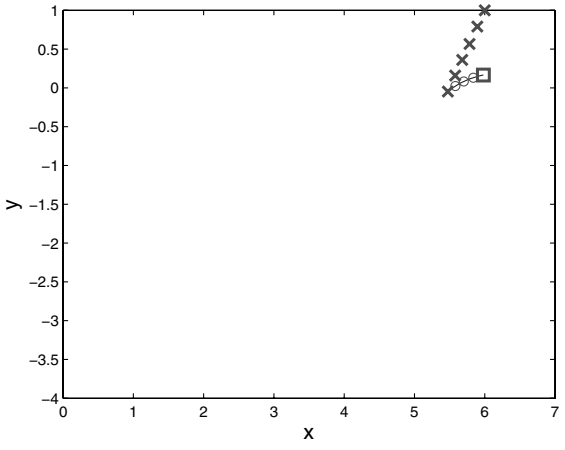
b) Horizon 4 grid point distribution



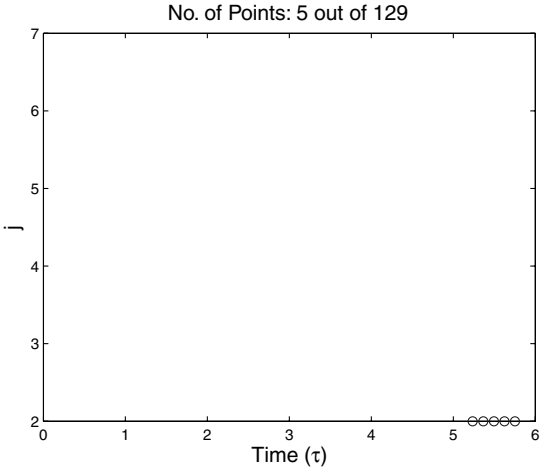
c) Horizon 5 trajectory



d) Horizon 5 grid point distribution



e) Horizon 6 trajectory



f) Horizon 6 grid point distribution

Fig. 2 Example 1 (target snapshots): trajectory and grid-point distributions for horizons 4, 5, and 6.

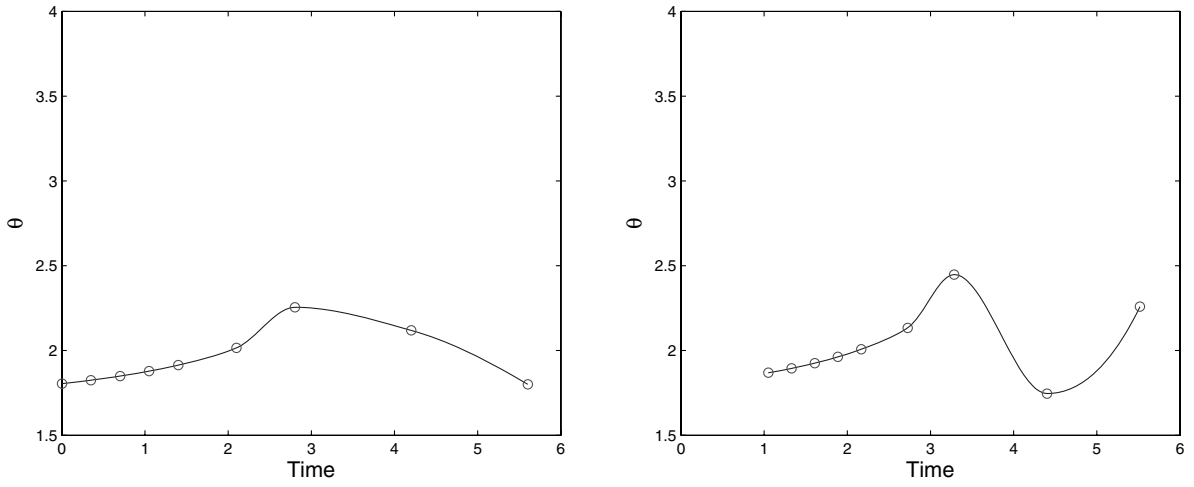
points at the finer resolution levels $\mathcal{V}_6$, and $\mathcal{V}_7$, as can be observed from the grid-point distribution for the fourth horizon (Fig. 2b).

For the fifth and sixth horizons, the algorithm used only 7 and 5 points, respectively, for computing the optimal solution. Because we had $\tau_0^6 > 5$ on the sixth horizon, the target was further assumed to be stationary located at

$$x_B^m = 6 - 0.1\tau_0^6, \qquad y_B^m = 1 - 0.2\tau_0^6 \tag{39}$$

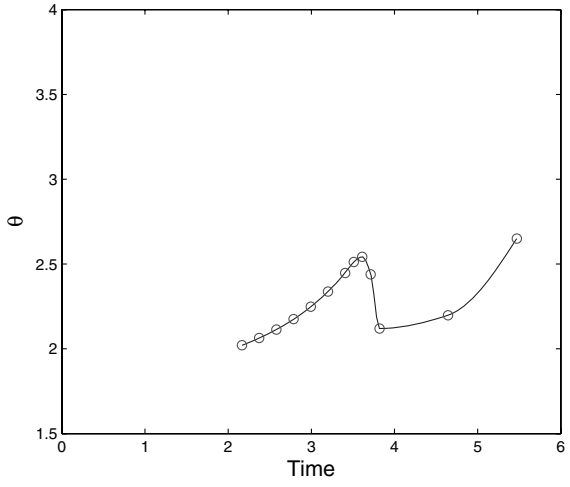
for all $m = 6, \ldots, N_H$. Hence, the algorithm terminated after solving the problem on the sixth horizon. The overall CPU time taken by STOA I to solve this problem was 5.1 s. The combined trajectory and the control found on different horizons are shown in Fig. 4.

Next, we incorporate the information of the trajectory profile of the target (33) in the optimal control problem itself. Because 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 a single step using the MTOA [14,15]. The results found using the MTOA are shown in Fig. 4. The overall CPU time taken by the MTOA to solve this problem was 9.5 s.

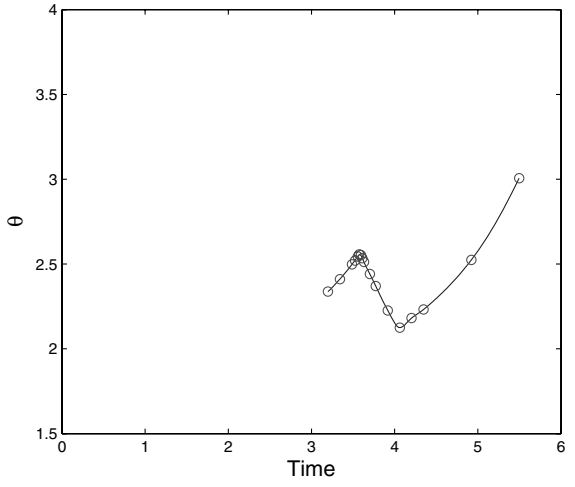


a) Horizon 1

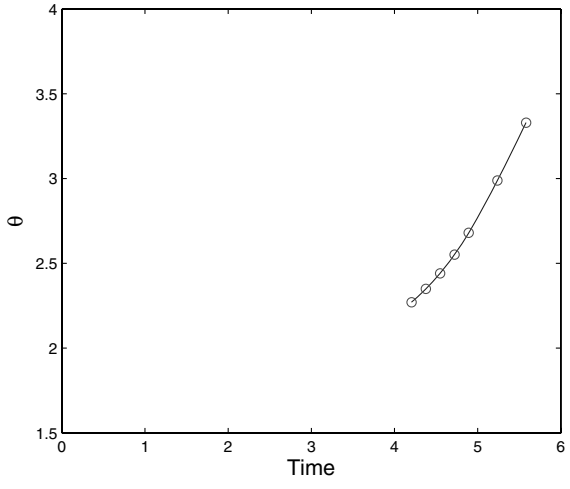
b) Horizon 2



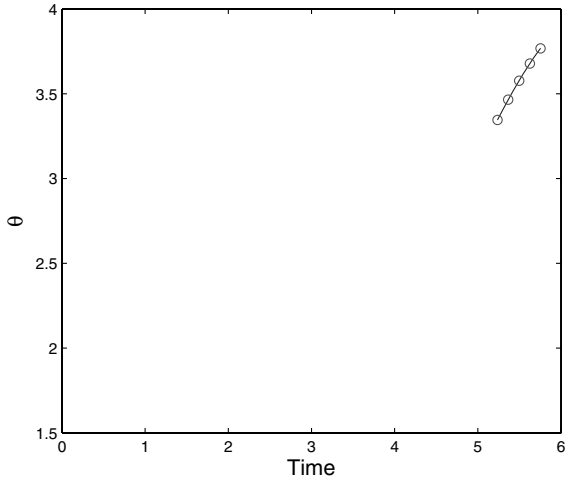
c) Horizon 3



d) Horizon 4



e) Horizon 5



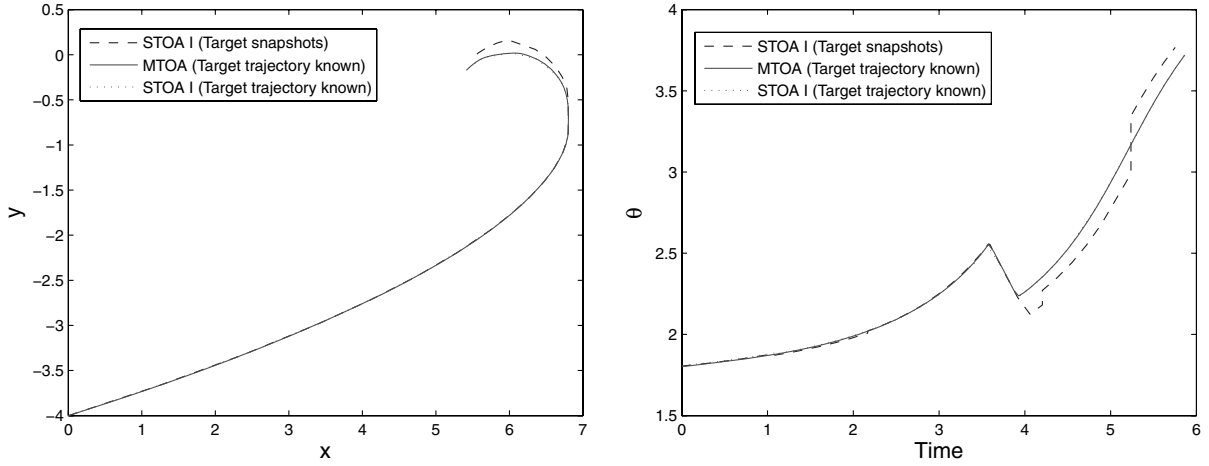
f) Horizon 6

Fig. 3 Example 1 (target snapshots): time history of control θ for all horizons.

We next solved the same problem using STOA I. For comparison purposes, the results found using STOA I are again shown in Fig. 4. The number of iterations taken by the algorithm before the algorithm terminated on each horizon (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), and 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 s. We see that the overall CPU time taken by STOA I is about two-thirds of the overall CPU time taken by the MTOA to solve the same problem.

Sequential Trajectory Optimization Algorithm II

In this section, we present yet another sequential trajectory optimization scheme referred to as the Sequential Trajectory



a) Trajectory **b) Time history of control θ**

Fig. 4 Example 1: trajectory and time history of the control θ using three different multiresolution strategies.

Optimization Algorithm II (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. 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 $[\tau_0, \tau_f] = [0, 1]$. Transformation (8) can be used to convert any optimal control problem from the domain $[\tau_0, \tau_f]$ to $[0, 1]$.

We choose the parameters $J_{\min}$, $J_{\max}$, and $\epsilon(t)$, as for the STOA I. 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 $i = 1$, iter $= 1$, and $\tau_0^i = 0$; initialize $\text{grid}_{\text{iter}}^i = \mathcal{V}_{J_{\min}}$; choose an initial guess for all NLP variables ($\mathcal{X}_{\text{iter}}^i$); and fix $\bar{J} = J_{\min} - 1$. The proposed sequential trajectory optimization algorithm proceeds as follows:

1) Solve the NLP problem on $\text{grid}_{\text{iter}}^i$ with the initial guess $\mathcal{X}_{\text{iter}}^i$ on the horizon $[\tau_0^i, 1]$. If $\text{grid}_{\text{iter}}^i$ has points from the level $\mathcal{W}_{J_{\max}-1}$, go to step 4.

2) Find grid_{new} using the mesh refinement step (step 2) of STOA I.

3) Set iter $=$ 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 grid_{new}, which will be our new initial guess $\mathcal{X}_{\text{iter}}^i$. Reassign the set $\text{grid}_{\text{iter}}^i$ to grid_{new}, and go to step 1.

4) New horizon.

a) Set $\text{grid}^i = \text{grid}_{\text{iter}}^i$.

b) Increment i by 1.

c) Set $\tau_0^i = t_{\bar{J},i-1}$.

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

e) Set $\text{grid}^{i-} = \{\tau\colon \tau \in \text{grid}^{i-1},\ \tau \geq \tau_0^i\}$.

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

g) Set iter $= 1$, $\mathcal{V}_j = \mathcal{V}_j \setminus (\mathcal{V}_j \cap [0, \tau_0^i))$ (where $j = \bar{J}, \ldots, J_{\max}$), and $\mathcal{W}_j = \mathcal{W}_j \setminus (\mathcal{W}_j \cap [0, \tau_0^i))$ (where $j = \bar{J}, \ldots, J_{\max} - 1$). Find grid_{new} using the mesh refinement step (step 2) of STOA I with $\text{grid}_{\text{iter}}^i = \text{grid}^{i-}$.

h) Empty $\text{grid}_{\text{iter}}^i$ and reassign $\text{grid}_{\text{new}} \rightarrow \text{grid}_{\text{iter}}^i$.

i) Interpolate the NLP solution given on grid^{i-} to $\text{grid}_{\text{iter}}^i$, which will be our new initial guess $\mathcal{X}_{\text{iter}}^i$ for step 1.

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

5) Go to step 1.

Remark 4. Although STOA II will work for any form of the threshold $\epsilon(t)$, on each horizon H_i ($i = 1, \ldots, N_H$), we recommend choosing

$$\epsilon(t) = \hat{\epsilon} E(\max\{0, t - t_{\bar{J},i}\}) \tag{40}$$

where $\hat{\epsilon}$ is at least of order $h_{J_{\min}} = 1/2^{J_{\min}}$ [14], and E: $[0, 1] \rightarrow \mathbb{R}^+$ such that $E(0) = 1$ and $t \in [0, 1]$. 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 until the initial time of the next horizon, which would be $t_{\bar{J},i}$ in this case.

Example 2. In this example, we consider the problem of minimizing the total stagnation-point convective heating per unit area during the reentry in the atmosphere of an Apollo-type capsule. This problem has been treated in [26] and, more recently, in [14]. The equations of motion and the initial and final boundary conditions can be found in these references. The angle of attack is constrained as

$$|u| \leq 68 \ \text{deg} \tag{41}$$

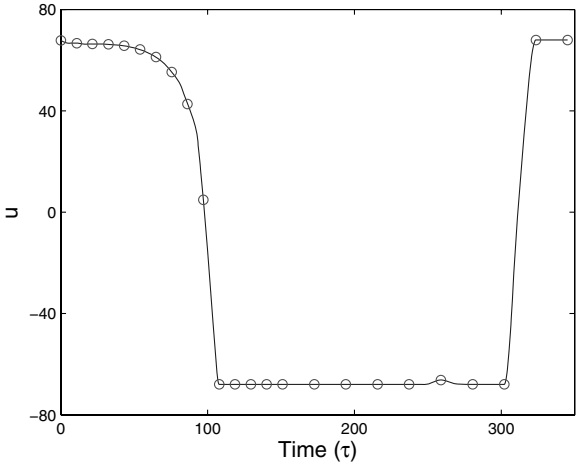
We have used STOA II to solve this problem with $J_{\min} = 4$ and $J_{\max} = 7$. The threshold used on each horizon H_i for this problem was

$$\epsilon(t) = 0.01 e^{7 \max\{0, t - t_{3,i}\}}, \qquad i = 1, \ldots, N_H \tag{42}$$

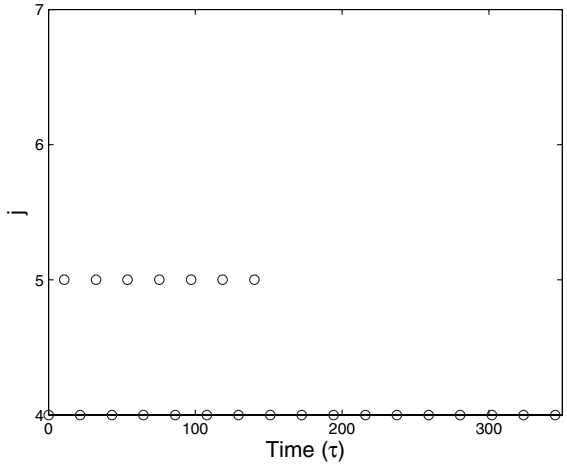
The other parameters used in the simulation for the mesh refinement step were $p = 3$ and $N_{\text{neigh}} = 1$. A fourth-order implicit Hermite–Simpson scheme [15] was used as a high-order scheme for discretizing the continuous-optimal-control problem into an NLP problem. We started STOA II with a linear initial guess. The algorithm terminated after solving the problem on 8 horizons, and the overall CPU time taken by the algorithm was 41.2 s, out of which 22 s were used to compute the solution on the first horizon H_1. For the sake of brevity, we only show the time histories of the control u, along with the grid-point distribution for different horizons, in Figs. 5–7. The number of iterations taken by the algorithm before the algorithm terminated on each horizon (iter_f), the maximum resolution level reached on each horizon (J_f), and the number of

Table 2 Example 1 (target trajectory known): number of iterations on each horizon (iter_f), maximum resolution level (J_f), number of nodes used by the algorithm at the final iteration on each horizon (N_f), and corresponding horizons

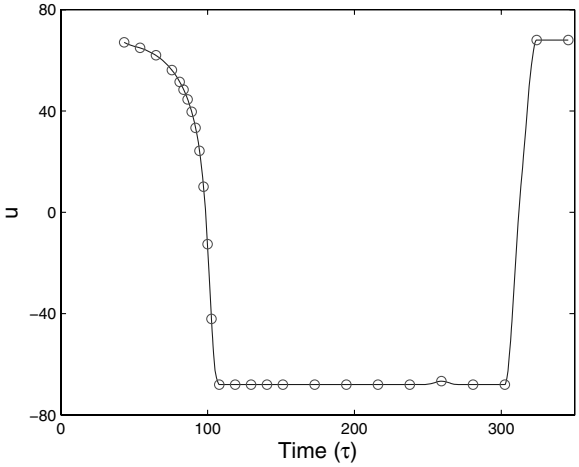
Horizon	iter_f	J_f	N_f	τ_0	τ_f
H_1	3	4	9	0	5.9065
H_2	3	4	9	1.1075	5.8256
H_3	3	4	11	2.2870	5.8648
H_4	6	7	17	3.4051	5.8643
H_5	1	2	5	4.4810	5.8642
H_6	1	2	5	5.5184	5.8642



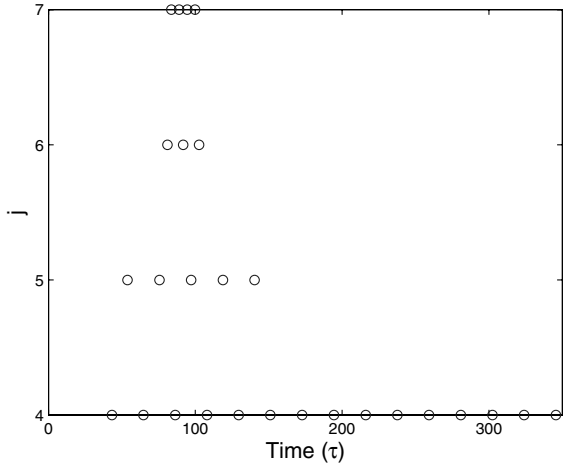
a) Horizon 1. control *u*



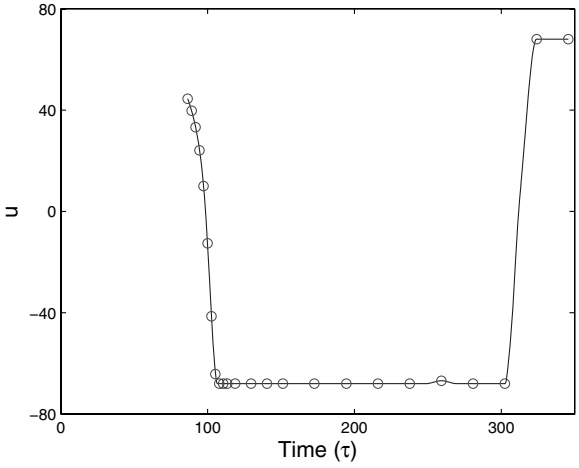
b) Horizon 1. grid-point distribution



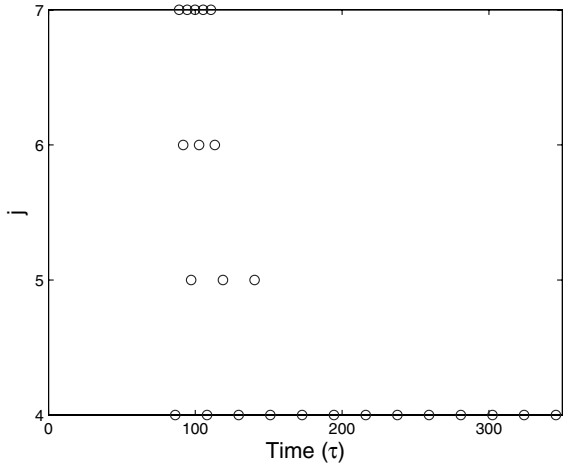
c) Horizon 2. control *u*



d) Horizon 2. grid-point distribution



e) Horizon 3. control *u*



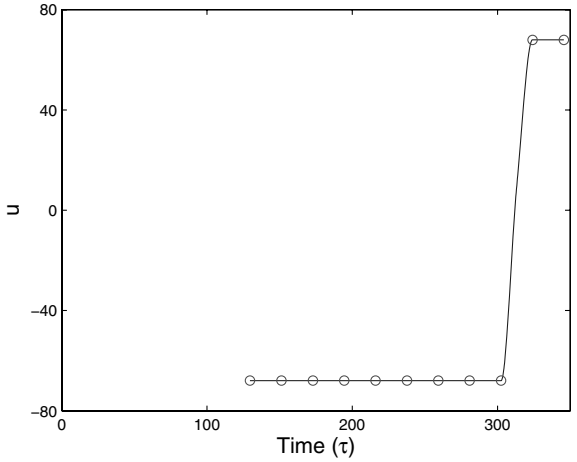
f) Horizon 3. grid-point distribution

Fig. 5 Example 2: control time-history and grid-point distributions for horizons 1, 2, and 3.

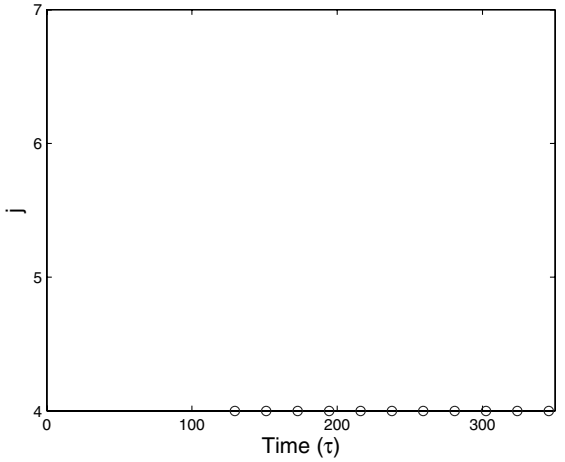
nodes used by the algorithm at the final iteration on each horizon (N_f) are shown in Table 3.

STOA I Versus STOA II

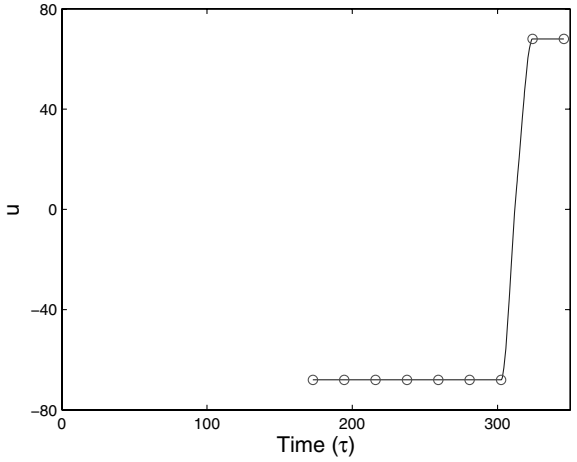
Both STOA I and STOA II have their own merits. STOA I will work for any user-specified time intervals ($\Delta\tau_{\text{ro}}$), whereas the time intervals in STOA II are dyadic and fixed. On the other hand, STOA II takes full advantage of the multiresolution structure of the grid in the mesh refinement 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 $\text{grid}^{i-} \subset \text{grid}^{i-1}$, for which the solution is already known. Hence, previously computed information is not lost, while



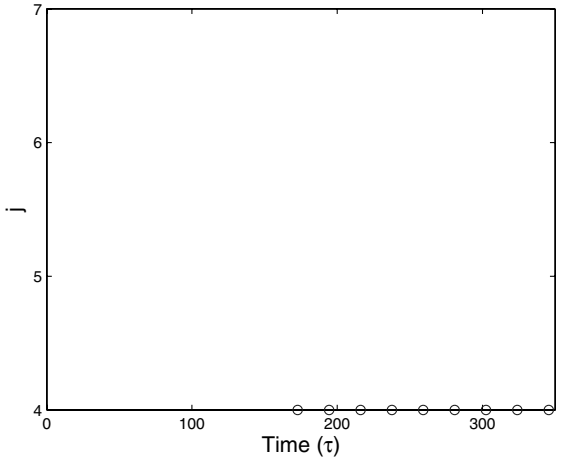
a) Horizon 4. control u



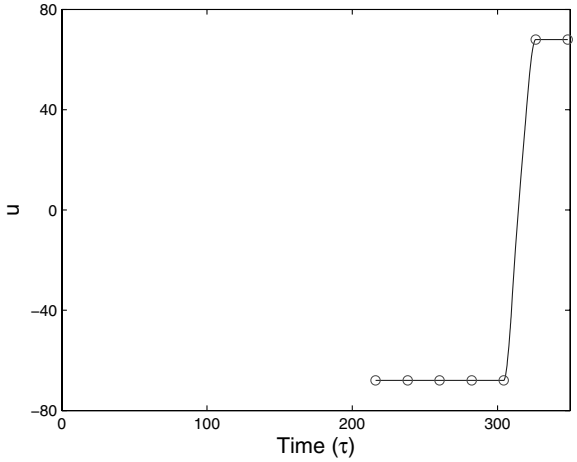
b) Horizon 4. grid-point distribution



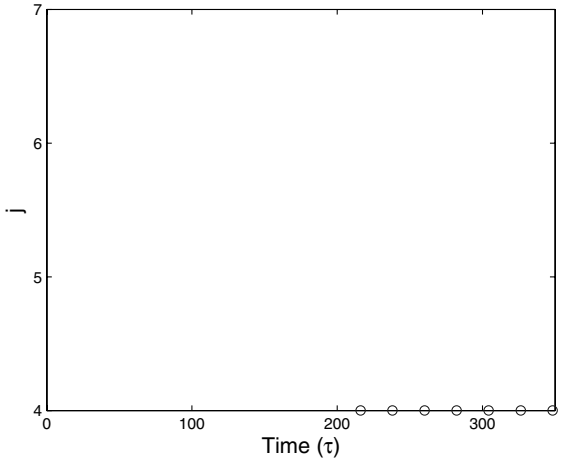
c) Horizon 5. control u



d) Horizon 5. grid-point distribution



e) Horizon 6. control u

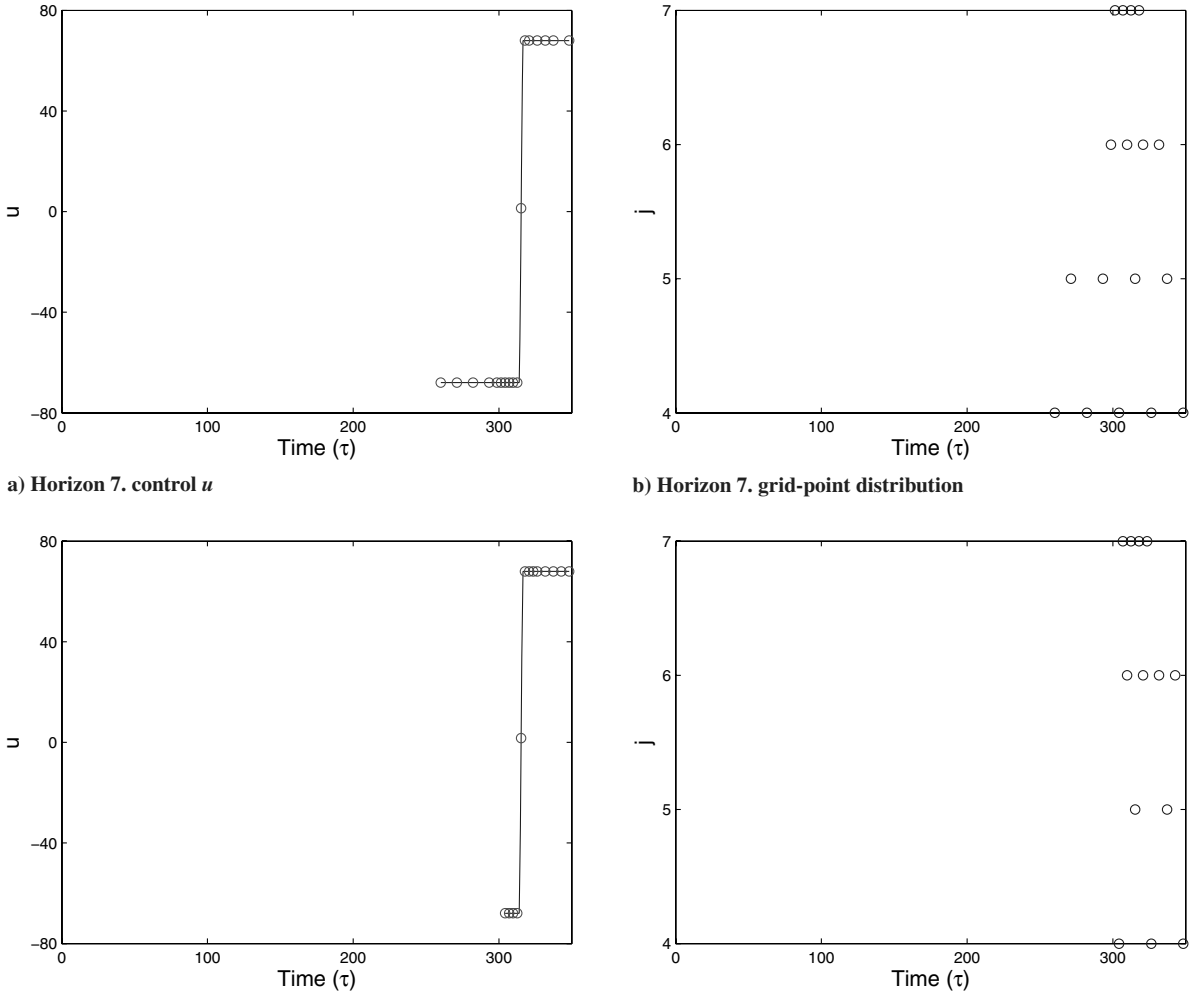


f) Horizon 6. grid-point distribution

Fig. 6 Example 2: control time-history and grid-point distributions for horizons 4, 5, and 6.

going from one horizon to the next. To provide an initial guess $\mathcal{X}_1^i$ for starting the NLP solver on horizon H_i $(i = 2, \ldots, N_H)$, the function values only need to be interpolated at a few additional points in the vicinity of the current time 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, because 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 grid_1^i may in fact 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 significantly, the algorithm for each successive horizon converges fast, because 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 because all of 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.



a) Horizon 7. control *u*

b) Horizon 7. grid-point distribution

c) Horizon 8. control *u*

d) Horizon 8. grid-point distribution

Fig. 7 Example 2: control time-history and grid-point distributions for horizons 7 and 8.

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 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. Further analysis is necessary to investigate the convergence of the proposed algorithms.

Table 3 Example 2: number of iterations on each horizon (iter_f), maximum resolution level (J_f), and number of nodes used by the algorithm at the final iteration on each horizon (N_f)

Horizon	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

Acknowledgments

Partial support for this work has been provided by National Science Foundation 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, No. 6, 1994, pp. 1154–1162. doi:10.2514/3.21327

[2] Lu, P., "Regulation About Time-Varying Trajectories: Precision Entry Guidance Illustrated," *Journal of Guidance, Control, and Dynamics*, Vol. 22, No. 6, 1999, pp. 784–790. doi:10.2514/2.4479

[3] Jardin, M. R., and Bryson, A. E., "Neighboring Optimal Aircraft Guidance in Winds," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 4, 2001, pp. 710–715. doi:10.2514/2.4798

[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, AIAA Paper 2002-4657, 2002.

[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 and Optimization*, Vol. 1, No. 2, Jan. 1963, pp. 193–223.

[6] Kelly, H. J., "An Optimal Guidance Approximation Theory," *IEEE Transactions on Automatic Control*, Vol. 9, No. 4, 1964, pp. 375–380. doi:10.1109/TAC.1964.1105747

[7] Speyer, J. L., and Bryson, A. E., "A Neighboring Optimum Feedback Control Scheme Based on Estimated Time-to-Go with Application to Reentry Flight Paths," *AIAA Journal*, Vol. 6, No. 5, 1968, pp. 769–776. doi:10.2514/3.4597

[8] Bryson, A. E., and Ho, Y.-C., *Applied Optimal Control: Optimization, Estimation, and Control*, Hemisphere, Washington, D.C., 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, No. 2, 1996, pp. 501–503. doi:10.2514/3.21648
[10] Ohtsuka, T., "Quasi-Newton-Type Continuation Method for Nonlinear Receding Horizon Control," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 4, 2002, pp. 685–692. doi:10.2514/2.4935
[11] Bellingham, J., Kuwata, Y., and How, J., "Stable Receding Horizon Trajectory Control for Complex Environments," AIAA Guidance, Navigation, and Control Conference and Exhibit, AIAA Paper 2003-5635, 2003.
[12] Williams, P., "Application of Pseudospectral Methods for Receding Horizon Control," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 2, 2004, pp. 310–314. doi:10.2514/1.5118
[13] Ross, I. M., Gong, Q., and Sekhavat, P., "Low Thrust, High Accuracy Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 4, 2007, pp. 921–933. doi:10.2514/1.23181
[14] Jain, S., and Tsiotras, P., "Trajectory Optimization using Multi-resolution Techniques," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 5, 2008, pp. 1424–1436. doi:10.2514/1.32220
[15] Jain, S., and Tsiotras, P., "Multiresolution-Based Direct Trajectory Optimization," *Proceedings of the 46th IEEE Conference on Decision and Control*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 2007, pp. 5991–5996.
[16] Hager, W. W., "Runge-Kutta Methods in Optimal Control and the Transformed Adjoint System," *Numerische Mathematik*, Vol. 87, No. 2, 2000, pp. 247–282. doi:10.1007/s002110000178
[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, No. 1, 2000, pp. 202–226. doi:10.1137/S0036142999351765
[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. doi:10.1090/S0025-5718-00-01184-4
[19] Betts, J. T., *Practical Methods for Optimal Control Using Nonlinear Programming*, Society for Industrial and Applied Mathematics, Philadelphia, 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, No. , 2002, pp. 1981–2007. doi:10.1137/S1064827500383044
[21] Harten, A., "Multiresolution Representation of Data: A General Framework," *SIAM Journal on Numerical Analysis*, Vol. 33, No. 3, 1996, pp. 1205–1256. doi:10.1137/0733060
[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* (to be published).
[24] Hairer, E., Norsett, S. P., and Wanner, G., *Solving Ordinary Differential Equations 1: Nonstiff Problems*, Springer–Verlag, New York, 1987.
[25] Hairer, E., Norsett, S. P., and Wanner, G., *Solving Ordinary Differential Equations 2: 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 and Methods*, Vol. 10, No. 2, 1989, pp. 147–171. doi:10.1002/oca.4660100206