Multiresolution-Based Direct Trajectory Optimization

S. Jain* and P. Tsiotras**
Georgia Institute of Technology, Atlanta, GA 30332-0150, USA

***Abstract*— In this paper we present a multiresolution-based approach for direct trajectory optimization. We transcribe the optimal control problem into a nonlinear programming (NLP) problem and solve the resulting NLP problem on an adaptive grid. The proposed algorithm automatically generates a grid to accurately capture the discontinuities and switchings in the state and control variables. The path constraints are handled with ease using the proposed technique without any additional computational complexity. The efficiency and accuracy of the proposed algorithm is demonstrated with the help of several examples.**

I. Introduction

It is difficult to find an analytic solution to a general trajectory optimization problem. Therefore, numerical techniques have been proposed in the literature, a nice survey of which can be found in [1]. In most numerical methods, one needs to use a high resolution (dense) grid to accurately capture any discontinuities or switchings in the state and/or control variables. This requires a large amount of computational resources, both in terms of CPU time and memory. Moreover, a large number of NLP variables can lead to ill conditioning of the discretized problem. In order to accurately capture any irregularities in the solution one would like to refine the mesh locally in the region close to the irregularity, instead of refining the mesh uniformly over the whole domain. Recently, some work has been done in this direction by Betts et al. [2], [3], Ross and Fahroo [4], [5], Gong et al. [6], Binder et al. [7], [8], [9], [10], and Schlegel et al. [11].

The method of Betts et al. [2], [3] selects the new grid points by solving an integer programming problem that minimizes the maximum discretization error (found by integrating the dynamics of the system) by subdividing the current grid. In [4], the authors use domain transformation techniques for generating the adaptive grids. The pseudospectral knotting method of Ross and Fahroo [5] generalizes the spectral patching method [12] by exchanging information across the patches in the form of event conditions associated with the optimal control problem, hence removing the restriction of continuity in the solution across the end points of the phases. The phase boundaries, termed as "knots" by the authors, can be fixed or free, with free knots constituting the part of the optimization process. On each phase, the problem is solved using the Legendre pseudospectral method [13] or Chebyshev pseudospectral method [14]. In order to improve the pseudospectral method, Gong et al. [6] present an algorithm in which the user specifies the number of nodes to be increased in a particular phase when the error between the computed optimal control of the two successive iterations is greater than the prescribed threshold. They use the gradient of the control to fix the location of the knots. Binder et al. [7], [9], [10] use wavelets to achieve better local resolution. They work in the wavelet space by using the wavelet-Galerkin approach to discretize the optimal control problem into an NLP problem and use a local error analysis of the states and a wavelet analysis of the control profile to add or remove wavelet basis functions. In [8] the authors use a direct shooting approach, where the optimal control problem is converted into an NLP problem by parameterizing the control profiles, combined with a wavelet analysis of the gradients of the Lagrangian function with respect to the parametrization functions at the optimal points in order to determine the regions that require refinement. For problems with state and/or control path constraints Schlegel et al. [11] use wavelet analysis of the control profile to determine the regions that require refinement.

In our continued effort for solving optimal control problems numerically [15], [16], we propose in this paper a novel multiresolution-based trajectory optimization technique using the ideas of [17] to solve the optimal control problem in a quick and efficient way. The proposed technique allows us to bypass solving any kind of secondary optimization problem for adding points to the mesh and, moreover, the criterion for deciding the mesh refinement regions is based on simple interpolations as opposed to integrations. At one go the proposed technique not only adds points to the grid but also removes points from the grid. On top of all this, the proposed multiresolution-based trajectory optimization technique can handle the constraints on states with ease without any further computational complexity.

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 briefly describe the mesh refinement algorithm of [17]. We then present the multiresolution-based trajectory optimization algorithm followed by several examples which show the efficiency and accuracy of the proposed algorithm. We conclude with a discussion on the advantages of the proposed algorithm over other existing adaptive methods [2], [3], [5], [6], [7], [8], [9], [10], [11] for solving the optimal control problems.

II. Problem Formulation

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

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

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

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

to the state dynamics

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

the boundary conditions

$$\mathbf{x}(\tau_0) = \mathbf{x}_0, \quad \mathbf{e}_f(\mathbf{x}(\tau_f), \tau_f) = 0, \tag{3}$$

where $\mathbf{e}_f : \mathbb{R}^{N_x} \times \mathbb{R}_+ \to \mathbb{R}^{N_e}$, and the state and control constraints

$$\mathbf{C}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) \leq 0, \tag{4}$$

where $\mathbf{C} : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times [\tau_0, \tau_f] \to \mathbb{R}^{N_c}$. The initial time τ_0 is assumed to be given and the final time τ_f can be fixed or free.

As stated above, it is very difficult to find an analytic solution to the trajectory optimization problem (1)-(4). In order to find a numerical solution to this problem, we transcribe the continuous optimal control problem into an NLP problem as described in the next section.

III. NLP Formulation

Consider a set of dyadic grids of the form

$$\mathcal{V}_j = \{t_{j,k} \in [0,1] : t_{j,k} = k/2^j,\ 0 \leq k \leq 2^j\}, \tag{5}$$

$J_{\min} \leq j \leq J_{\max}$, where j denotes the resolution level, k the spatial location, and $J_{\min},\ J_{\max} \in \mathbb{Z}_+^0$. We denote by $\mathcal{W}_j$ the set of grid points belonging to $\mathcal{V}_{j+1} \setminus \mathcal{V}_j$. Therefore,

$$\mathcal{W}_j = \{\hat{t}_{j,k} \in [0,1] : \hat{t}_{j,k} = (2k+1)/2^{j+1},\ 0 \leq k \leq 2^j - 1\}, \tag{6}$$

$J_{\min} \leq j \leq J_{\max} - 1$. An example of a dyadic grid with $J_{\min} = 0$ and $J_{\max} = 5$ is shown in Fig. 1. For

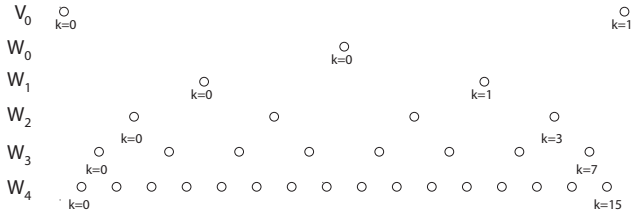


Fig. 1. Example of a dyadic grid.

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

The transformation $\tau = \Delta\tau \cdot t + \tau_0$, where $\Delta\tau = \tau_f - \tau_0$, can be used to express the trajectory optimization problem stated in Section II from any interval $[\tau_0, \tau_f]$ to $t \in [0,1]$. Hence, the trajectory optimization problem reduces to minimizing the following cost functional

$$J = e(\mathbf{x}(1), \tau_f) + \Delta\tau \int_0^1 L(\mathbf{x}(t), \mathbf{u}(t), t)\mathrm{d}t, \tag{7}$$

subject to the state dynamics

$$\frac{1}{\Delta\tau}\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t], \tag{8}$$

where $\mathbf{x} : [0,1] \to \mathbb{R}^{N_x}$, $\mathbf{u} : [0,1] \to \mathbb{R}^{N_u}$, the boundary conditions $\mathbf{x}(0) = \mathbf{x}_0$, $\mathbf{e}_f(\mathbf{x}(1), \tau_f) = 0$, and constraints $\mathbf{C}(\mathbf{x}(t), \mathbf{u}(t), t) \leq 0$.

Now we convert the above mentioned optimal control problem into an NLP problem using a Runge-Kutta (RK) discretization. To this end, let a nonuniform grid of the form

$$\begin{aligned} G \ = \ & \{t_{j_i,k_i} : t_{j_i,k_i} \in [0,1],\ 0 \leq k_i \leq 2^{j_i}, \\ & J_{\min} \leq j_i \leq J_{\max},\ \text{for } i = 1, \ldots, N_t,\ \text{and} \\ & t_{j_i,k_i} < t_{j_{i+1},k_{i+1}},\ \text{for } i = 1, \ldots, N_t - 1\}. \end{aligned} \tag{9}$$

Then a q-stage RK method for discretizing (8) is given by [1], [18]

$$\mathbf{x}_{j_{i+1},k_{i+1}} = \mathbf{x}_{j_i,k_i} + h_{j_i,k_i}\Delta\tau \sum_{\ell=1}^{q} \beta^\ell \mathbf{f}_{j_i,k_i}^\ell, \tag{10}$$

where $\mathbf{f}_{j_i,k_i}^\ell = \mathbf{f}(\mathbf{x}_{j_i,k_i}^\ell, \mathbf{u}_{j_i,k_i}^\ell, t_{j_i,k_i}^\ell)$, $\mathbf{x}_{j_i,k_i}^\ell,\ \mathbf{u}_{j_i,k_i}^\ell,\ t_{j_i,k_i}^\ell$ 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

$$\mathbf{x}_{j_i,k_i}^\ell = \mathbf{x}_{j_i,k_i} + h_{j_i,k_i}\Delta\tau \sum_{m=1}^{q} \alpha^{\ell,m} \mathbf{f}_{j_i,k_i}^m, \tag{11}$$

where $h_{j_i,k_i} = t_{j_{i+1},k_{i+1}} - t_{j_i,k_i}$, $t_{j_i,k_i}^\ell = t_{j_i,k_i} + h_{j_i,k_i}\rho^\ell$, $\mathbf{u}_{j_i,k_i}^\ell = \mathbf{u}(t_{j_i,k_i}^\ell)$, for $1 \leq \ell \leq q$, and q is referred to as the *stage*. In these expressions $\rho^\ell, \beta^\ell, \alpha^{\ell,m}$ are known constants with $0 \leq \rho^1 \leq \rho^2 \leq \cdots \leq 1$. The scheme is explicit if $\alpha^{\ell,m} = 0$ for $m \geq \ell$ and implicit otherwise.

Using (10) the defects of the discretization are given by

$$\zeta_i = \mathbf{x}_{j_{i+1},k_{i+1}} - \mathbf{x}_{j_i,k_i} - h_{j_i,k_i}\Delta\tau \sum_{\ell=1}^{q} \beta^\ell \mathbf{f}_{j_i,k_i}^\ell, \tag{12}$$

for $i = 1, \ldots, N_t - 1$.

For discretizing the cost functional (7), we introduce a new state $y(t)$ such that

$$\dot{y}(t) = \Delta\tau L(\mathbf{x}(t), \mathbf{u}(t), t)\mathrm{d}t, \quad y(0) = 0. \tag{13}$$

Then, using a q-stage RK method for discretizing (13) yields

$$y_{j_{N_t},k_{N_t}} = \Delta\tau \sum_{i=0}^{N_t-1} h_{j_i,k_i} \sum_{\ell=1}^{q} \beta^\ell L_{j_i,k_i}^\ell, \tag{14}$$

where $L_{j_i,k_i}^\ell = L(\mathbf{x}_{j_i,k_i}^\ell, \mathbf{u}_{j_i,k_i}^\ell, t_{j_i,k_i}^\ell)$. Hence, the cost functional (7) written in the discretized form is as follows

$$J = e(\mathbf{x}_{j_{N_t},k_{N_t}}, \tau_f) + \Delta\tau \sum_{i=1}^{N_t-1} \left(h_{j_i,k_i} \sum_{\ell=1}^{q} \beta^\ell L_{j_i,k_i}^\ell \right). \tag{15}$$

Let us now define the following sets

$$\mathbf{X} = \{\mathbf{x}_{j_1,k_1}, \ldots, \mathbf{x}_{j_{N_t},k_{N_t}}\}, \quad \mathbf{U} = \{\mathbf{u}_{j_1,k_1}, \ldots, \mathbf{u}_{j_{N_t},k_{N_t}}\},$$
$$\overline{G} = \{t_{j_i,k_i}^\ell \in [0,1] : t_{j_i,k_i}^\ell \notin G,\ 0 \leq i < N_t,\ 1 \leq \ell \leq q\},$$
$$\overline{\mathbf{X}} = \{\mathbf{x}_{j_i,k_i}^\ell : t_{j_i,k_i}^\ell \in \overline{G}\}, \quad \overline{\mathbf{U}} = \{\mathbf{u}_{j_i,k_i}^\ell : t_{j_i,k_i}^\ell \in \overline{G}\}.$$

As a result of the discretization the optimal control problem reduces to the NLP problem of finding the variables $\mathbf{X},\ \mathbf{U},\ \overline{\mathbf{U}},\ \tau_f$, that minimize (15) subject to the following constraints

$$\zeta_i = 0, \quad i = 1, \ldots, N_t - 1, \tag{16}$$
$$\mathbf{x}_{j_0,k_0} = \mathbf{x}_0, \tag{17}$$
$$\mathbf{e}_f(\mathbf{x}_{j_{N_t},k_{N_t}}, \tau_f) = 0, \tag{18}$$
$$\mathbf{C}(\mathbf{X}, \overline{\mathbf{X}}, \mathbf{U}, \overline{\mathbf{U}}, G, \overline{G}) \leq 0. \tag{19}$$

We use SNOPT [19] to solve (15)-(19). SNOPT is an NLP solver which is based on a sequential quadratic programming (SQP) algorithm.

Since the trajectory optimization problem can have discontinuities and switchings in the states and the controls, one way to accurately capture these discontinuities and switchings in the solution is to solve the NLP problem on a very fine mesh. However, this will require a lot of computational resources in terms of both CPU time and memory. Moreover, a large number of NLP variables can lead to ill-conditioning of the discretized problem. Therefore, in order to accurately capture the irregularities in the solution and alleviate these problems, we will only refine the mesh locally in the region of the irregularity using the multiresolution-based mesh refinement algorithm described in the next section. For more details on the mesh refinement algorithm the reader is referred to [17].

IV. The Mesh Refinement Algorithm

Consider a set of dyadic grids $\mathcal{V}_j$ and $\mathcal{W}_j$ as described in equations (5) and (6) before. Assume we are given a nonuniform grid $Grid_{\rm old}$ of the form (9) (that is, $Grid_{old} = G$), with any function $\phi : [0,1] \to \mathbb{R}$ specified on $Grid_{\rm old}$, $\Phi_{\rm old} = \{\phi_{j,k} : t_{j,k} \in Grid_{\rm old}\}$, where $\phi_{j,k} = \phi(t_{j,k})$.

Our aim now is to find a new grid $Grid_{\rm new}$, by adding or removing points from $Grid_{\rm old}$. This stems from the requirement for capturing the irregularity in ϕ more accurately. To this end, we initialize an intermediate grid $Grid_{\rm int} = \mathcal{V}_{J_{\min}}$, with function values

$$\Phi_{\rm int} = \{\phi_{J_{\min},k} : \phi_{J_{\min},k} \in \Phi_{\rm old},\ 0 \le k \le 2^{J_{\min}}\}, \quad (20)$$

and set $j = J_{\min}$. Then the mesh refinement algorithm proceeds as follows:

1) Find the points that belong to the intersection of $\mathcal{W}_j$ and $Grid_{\rm old}$

$$\hat{T} = \{\hat{t}_{j,k_i} : \hat{t}_{j,k_i} \in \mathcal{W}_j \cap Grid_{\rm old},\ \text{for } i = 1, \ldots, N_{\hat{t}}\}. \quad (21)$$

2) Set $i = 1$.
   a) Compute the interpolated function value at point $\hat{t}_{j,k_i} \in \hat{T}$, $\hat{\phi}(\hat{t}_{j,k_i})$, from the points in $Grid_{\rm int}$ and their function values in the set $\Phi_{\rm int}$ using an interpolating polynomial of degree p[1].
   b) Calculate the interpolative error coefficient[2] d_{j,k_i} at the point $\hat{t}_{j,k_i}$, $d_{j,k_i} = |\phi(\hat{t}_{j,k_i}) - \hat{\phi}(\hat{t}_{j,k_i})|$. If the value of d_{j,k_i} is below the threshold $\epsilon/2^{(j-J_{\min})/2}$, then reject $\hat{t}_{j,k_i}$ and goto Step 2f, otherwise add $\hat{t}_{j,k_i}$ to the intermediate grid $Grid_{\rm int}$ and move on to the next step.
   c) Add to $Grid_{\rm int}$ N_1 points on the left and N_1 points on the right neighboring to the point $\hat{t}_{j,k_i}$ in $\mathcal{W}_j$.
   d) Add to $Grid_{\rm int}$ $2N_2$ neighboring points at the next finer level $\{\hat{t}_{j+1,2k_i+\ell}\}_{\ell=-N_2+1}^{N_2}$.
   e) Add the function values at all the newly added points to $\Phi_{\rm int}$. 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 $Grid_{\rm old}$ and their function values in $\Phi_{\rm old}$ using the method described in [17].
   f) Increment i by 1. If $i \le N_{\hat{t}}$ goto Step 2a, otherwise move on to the next step.
3) Increment j by 1. If $j < J_{\max}$ goto Step 1, otherwise move on to the next step.
4) Terminate the algorithm. The final nonuniform grid is $Grid_{\rm new} = Grid_{\rm int}$ and the corresponding function values are in the set $\Phi_{\rm new} = \Phi_{\rm int}$.

It should be noted that the threshold $\epsilon/2^{(j-J_{\min})/2}$ changes with each level $\mathcal{W}_j$. The threshold becomes increasingly smaller as one goes from $j = J_{\min}$ to $j = J_{\max} - 1$. This is to account for the reduction in the interpolation error with the decrease in the distance between the interpolating points.

Now we are ready to present the proposed multiresolution-based trajectory optimization algorithm.

V. Multiresolution Trajectory Optimization Algorithm (MTOA)

We first transcribe the continuous trajectory optimization problem into an NLP problem using a q-stage RK discretization and select the minimum resolution level $J_{\min}$, the maximum resolution level $J_{\max}$, the threshold ϵ, the order of the interpolating polynomial p, and the parameters N_1, N_2 required for the mesh refinement algorithm based on the problem and the desired accuracy. Next, we initialize $Grid_{\rm old} = \mathcal{V}_{J_{\min}}$, choose an initial guess for all NLP variables, and denote the set of initial guesses by $Guess$. The proposed multiresolution-based trajectory optimization algorithm proceeds as follows:

1) Solve the NLP problem on $Grid_{\rm old}$ with the initial guess $Guess$.
2) *Mesh refinement*.
   a) If the problem either has pure state constraints or mixed constraints on states and controls: find $Grid_{{\rm new}_i}$, $i = 1, \ldots, N_u + N_{xc}$, where N_{xc} is the number of states involved in the constraints, by setting $\Phi_{\rm old}$ equal to the NLP solution of $\{u_n\}_{n=1}^{N_u}$, $\{x_n\}_{n=1}^{N_{xc}}$, in the mesh refinement algorithm mentioned in Section IV, set $N_r = N_u + N_{xc}$, and goto Step 2c.
   b) Find $Grid_{{\rm new}_i}$ by setting $\Phi_{\rm old}$ equal to the NLP solution of u_i, for $i = 1, \ldots, N_u$, in the mesh refinement algorithm mentioned in Section IV. Set $N_r = N_u$.
   c) The new final mesh will be $Grid_{\rm new} = \cup_{i=1}^{N_r} Grid_{{\rm new}_i}$.
3) Interpolate the NLP solution found in Step 1 on the new mesh $Grid_{\rm new}$, which will be our new initial guess $Guess$, reassign the set $Grid_{\rm old}$ to $Grid_{\rm new}$, and goto Step 1. The whole process is repeated until $Grid_{\rm old}$ has points from the finest level $\mathcal{W}_{J_{\max}-1}$. Once $Grid_{\rm old}$ has points from the level $\mathcal{W}_{J_{\max}-1}$ repeat Step 1 and terminate.

Remark 1: For increased robustness of the algorithm, in Step 2a) of MTOA, one can also perform mesh refinement

[1]For details on the computation of interpolating polynomials, the reader is referred to [17].

[2]The interpolative error coefficient is a measure of the local smoothness of the function.

based on all the states (irrespective of whether the corresponding states are involved in the constraints or not) and the constraints $\mathbf{C}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau)$ in addition to the mesh refinement based on the states involved in the constraints and the mesh refinement based on the controls.

It should be noted that since we first solve the problem on a coarse grid the solution converges pretty fast as the number of NLP variables is small. During the next iterations, we provide the solution of the previous iteration as an initial guess, and as we continue to iterate, our initial guess becomes more and more accurate. As a result, the solution to the NLP problem converges very fast even as the number of NLP variables increases from the previous iteration.

VI. Numerical Examples

Example 1: We consider the Moon landing problem, taken from [14]. The control problem is formulated as maximizing the final mass, and hence minimizing $J = -m(\tau_f)$. The equations of motion are given by $\dot{h} = v$, $\dot{v} = -g + T/m$, $\dot{m} = -T/(I_{\rm sp} g)$, where the state variables $h,\ v,\ m$ are altitude, velocity, and mass respectively. Control is provided by the thrust T, which is bounded by $0 \leq T \leq T_{\max}$. The final time τ_f is free. The other parameters in the problem are g, the gravity of the Moon, and $I_{\rm sp}$, the specific impulse of the spacecraft engine. The normalized parameters for the problem were chosen the same as in [14]: $T_{\max}/m_0 g = 1.1$, $I_{\rm sp} g/v_0 = 1$, $h(0)/h_0 = 0.5$, $v(0)/v_0 = -0.05$, $m(0)/m_0 = 1$, for any given set of initial conditions $h_0,\ v_0$, and m_0. Therefore, we have the following normalized initial conditions: $h(0) = 0.5$, $v(0) = -0.05$, $m(0) = 1.0$. For soft landing, we must have $h(\tau_f) = 0$, and $v(\tau_f) = 0$. In addition, for a physical meaningful trajectory, we must have $m(\tau_f) > 0$.

We solved this problem on a grid with $J_{\min} = 3$ and $J_{\max} = 9$. The other parameters used in the simulation are $p = 1$, $\epsilon = 10^{-2}$, and $N_1 = N_2 = 1$. For discretizing this problem into an NLP problem, we used the fourth-order explicit Runge-Kutta scheme $(q = 4)$ with the following parameters:

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

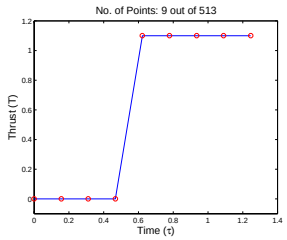
Since the problem does not have any state constraints or the mixed constraints on states and controls, thrust T was chosen to be the refinement criterion for constructing the new mesh in MTOA. The algorithm terminated in 7 iterations. Because of the space constraints we show the time history of the thrust T along with the grid point distribution only for iterations $1,\ 3,\ 6, \text{and } 7$ (Fig. 2). The grid point distributions in Fig. 2 show that with each iteration the algorithm adds points at finer resolution levels, and as a result the solution is getting more and more accurate. Moreover, as the solution gets more and more accurate, the algorithm also removes points at the coarser levels from the region where the solution is getting smoother. The grid point distribution at iteration 7 (Fig. 2(h)) shows that the regions where the solution is smooth are well represented by the coarse resolution levels; the higher resolution levels are needed only near the switching points in the thrust T, thus illustrating the efficiency of the proposed algorithm. From Fig. 2(g), we see that the algorithm was accurately able to capture the switching in the control using only two points. It should be noted that the algorithm used only 26 points out of 513 points of the grid $\mathcal{V}_9$ for calculating the final solution. One should also discern that the algorithm used 27 points at iteration 6 whereas used 26 points at iteration 7. At iteration 7, the algorithm removed some points at the coarser resolution levels and added points at the finer resolution level $\mathcal{W}_8$. This clearly demonstrates that the proposed strategy uses only the grid points that are actually necessary to attain a given precision, and the algorithm is able to add and remove points when and where is needed. The time history of mass m along with the phase portrait of the velocity v vs. the altitude h for the last iteration are shown in Fig. 3.

Example 2: Next we consider a minimum-energy problem with path constraint [20]. The problem is to find the control $u(t)$ that minimizes the cost function $J = 0.5 \int_0^1 u^2 \mathrm{d}\tau$, subject to the dynamics $\dot{x} = v$, $\dot{v} = u$, initial and final conditions, $x(0) = x(1) = 0$, $v(0) = -v(1) = 1$, and the path constraint, $x(\tau) \leq 0.04$.

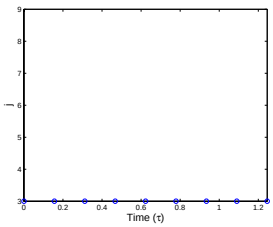
This is a problem with a second-order state variable inequality constraint and hence, the state x and the control u were chosen as the refinement criterion for constructing the new mesh. We solved this problem on a grid with $J_{\min} = 3$ and $J_{\max} = 10$. The other parameters used in the simulation are $p = 1$, $\epsilon = 10^{-2}$, and $N_1 = N_2 = 1$. For discretizing this problem into an NLP problem, we used an implicit Hermite-Simpson scheme, the defects of discretization for which can be found in [18]. It has been indicated in [21] that the Hermite-Simpson scheme is equivalent to a 3-stage Runge-Kutta scheme with the following parameters

0	0	0	0
1/2	5/24	1/3	−1/24
1	1/6	2/3	1/6
	1/6	2/3	1/6

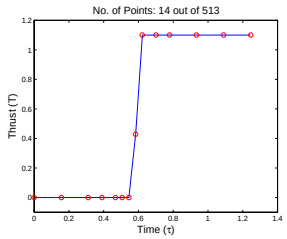
The algorithm terminated in 8 iterations. The time histories of states $x,\ v$, and the control u along with the grid point distribution for the final iteration are shown in Fig. 4. The grid point distribution (Fig. 4(d)) again shows that the regions where the solution is smooth are well represented by the coarse resolution levels; the higher resolution levels are needed only near the points where the state x enters and leaves the constraint. The number of grid points (N_g) used by the algorithm along with the error in the computed optimal cost ($E = |J_{\rm a} - J_{\rm n}|$, where $J_{\rm a}$ is the optimal cost found analytically [20] and $J_{\rm n}$ is the optimal cost found using the proposed algorithm) at each iteration are shown in Table I. The optimal cost found numerically, using the proposed technique, after the $8^{\rm th}$ iteration is 11.1111111, which is accurate up to 7 decimal places compared to the analytic solution ($J_{\rm a} = 4/0.36$). One should note that the proposed algorithm used only 51 points out of 1025 points of the grid $\mathcal{V}_{10}$. Table I also illustrates that with each iteration, $J_{\rm n}$ is converging to the analytic solution and the error is decreasing roughly by an order of magnitude which again shows the efficacy of the proposed algorithm.



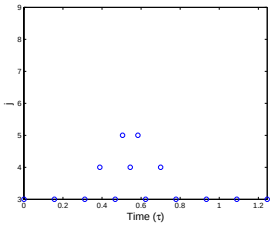
(a) Iteration 1: Time history of thrust T.



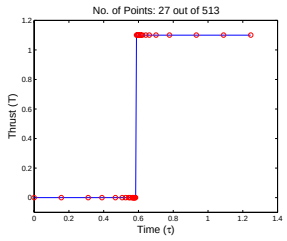
(b) Iteration 1: Grid point distribution.



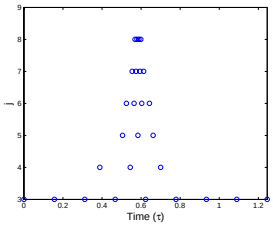
(c) Iteration 3: Time history of thrust T.



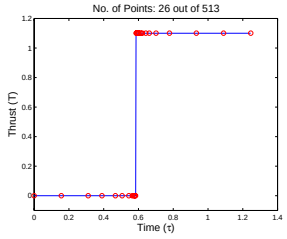
(d) Iteration 3: Grid point distribution.



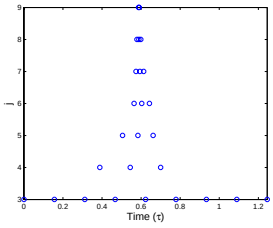
(e) Iteration 6: Time history of thrust T.



(f) Iteration 6: Grid point distribution.



(g) Iteration 7: Time history of thrust T.

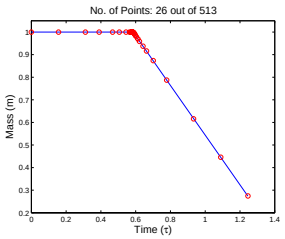


(h) Iteration 7: Grid point distribution.

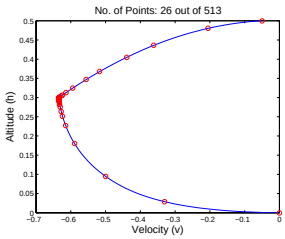
Fig. 2. Example 1. Time history of thrust T along with the grid point distribution for iterations 1, 3, 6, and 7.

VII. Comparison of MTOA with Existing Methods

One of the most well-known adaptive methods for solving optimal control problems is the method of Betts et al. [2], [3], [18] which requires the solution of an integer programming problem in order to refine the mesh, in addition to the solution of the NLP problem, which adds to the overall computational overhead. The MTOA proposed in this paper avoids the solution of any secondary optimization problem for adding points to the mesh. Only simple interpolations are needed to refine the mesh, which can be done on the fly using, for instance, Neville's algorithm. Furthermore, the MTOA does not involve any integrations, as opposed to the highly accurate integrations required in the method of Betts et al. [3], [18]. This again can be computationally expensive for nonlinear dynamics. Moreover, the algorithm of [2], [3], [18] can only add points to the grid, whereas

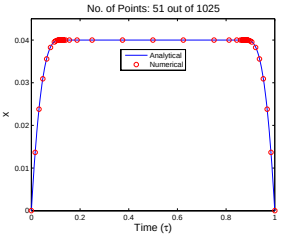


(a) Iteration 7: Time history of mass m.

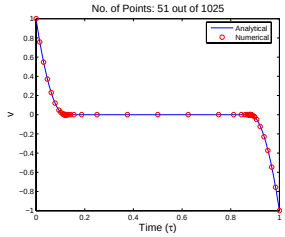


(b) Iteration 7: Phase portrait of v vs. h.

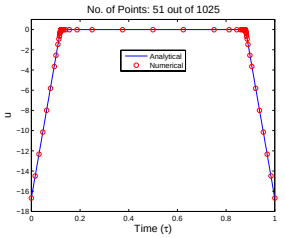
Fig. 3. Example 1. Phase portrait of velocity v vs. altitude h for iteration 7.



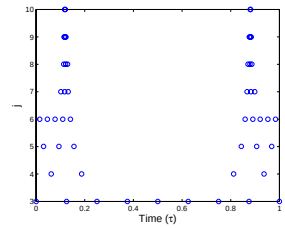
(a) Iteration 8: Time history of x.



(b) Iteration 8: Time history of v.



(c) Iteration 8: Time history of u.



(d) Iteration 8: Grid point distribution.

Fig. 4. Example 2. Time history of x, v, and u along with the grid point distribution for iteration 8.

MTOA is capable of not only adding points to the grid but also removing points from the grid when and where is needed. Finally, MTOA can handle state constraints without further computational complexity.

In a recent paper Ross and Fahroo [5] use the pseudospectral knotting method to handle problems with discontinuities and switches in states and control. However, as pointed out by Ross [22] "Soft knots[3] do not increase the speed of the algorithm; they are expected to improve accuracy. Consequently, the introduction of soft knots in the grid might significantly slow the algorithm." Moreover, in the pseudospectral knotting method, one needs to know a priori the approximate number and location of singularities in the solution. In [6] Gong et al. use the gradient of the control to fix the locations of the knots but incase the error between the computed control of two successive iterations is above the prescribed threshold, the number of nodes to be added to a particular phase is still defined by the user even before starting the algorithm. Moreover, the structure of the pseudospectral grid for each phase is fixed, irrespectively of the location of the knot. On the other hand, MTOA is fully autonomous. The user need not know a priori the number

[3]The reader is referred to [5] for the definition of soft knots.

TABLE I

EXAMPLE 2: NO. OF GRID POINTS ALONG WITH THE ERROR IN THE COMPUTED OPTIMAL COST AT EACH ITERATION.

Iteration	N_g	$J_{\rm n}$	$E = \lvert J_{\rm a} - J_{\rm n}\rvert$
1	9	11.1360	2.5×10^{-2}
2	13	11.1168	5.7×10^{-3}
3	17	11.1130	1.8×10^{-3}
4	27	11.1115	4.5×10^{-4}
5	33	11.1112	7.2×10^{-5}
6	39	11.1111	6.6×10^{-6}
7	45	11.1111	1.5×10^{-6}
8	51	11.1111	4.0×10^{-8}

of irregularities nor the locations of the irregularities in the solution. MTOA will automatically detect the regions of irregularities in the solution and autonomously add points accordingly when and where is needed. Furthermore, in the proposed MTOA, the grid for solving the NLP problem is fully adaptive. The algorithm can add and remove points anywhere in the grid, hence the grid can embrace any form depending on the irregularities in the solution, thus providing more flexibility in capturing any irregularities in the solution.

Finally, comparing MTOA to the third method [7], [9], [10], we find that in wavelet-Galerkin approach multiplication in the physical space becomes convolution in the wavelet space, which is computationally costly. In [8] and [11] the authors use wavelet analysis to determine the regions of irregularities in the solution, which results in additional computational overhead, as one needs to transform back and forth between the physical and wavelet domain. In MTOA we always work in the physical domain and operations like multiplication and differentiation are therefore performed fast compared to the wavelet domain. Furthermore, nonlinearities can be handled with ease. In addition, in the wavelet method of [8] and [11] one needs to interpolate the function values at the finest level every time one needs to perform the wavelet transform. This adds to the overall computational overhead. The mesh refinement technique of MTOA on the other hand, uses only the retained points in the grid to decide where to add or remove points from the grid. Hence there is no need of interpolating the function values at the finest level.

VIII. CONCLUSIONS

In this paper we have proposed a novel multiresolution-based approach for direct trajectory optimization. The generated grid is able to automatically adapt to any irregularities or discontinuities in the solution. The constraints on the states are dealt with ease in a straightforward manner without any additional computational overhead. The simplicity and efficiency of the proposed method allows one to perform rapid and accurate trajectory optimization. The proposed multiresolution trajectory optimization algorithm has been shown to have several advantages over the existing adaptive methods for solving optimal control problems via direct transcription.

Acknowledgment: This work was partially supported by NSF award no. CMS-0510259.

REFERENCES

[1] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, pp. 193–207, 1998.

[2] J. T. Betts and W. P. Huffman, "Mesh refinement in direct transcriptioin methods for optimal control," *Optimal Control Applications & Methods*, vol. 19, pp. 1–21, 1998.

[3] J. T. Betts, N. Biehn, S. L. Campbell, and W. P. Huffman, "Compensating for order variation in mesh refinement for direct transcription methods," *Journal of Computational and Applied Mathematics*, vol. 125, pp. 147–158, 2000.

[4] I. M. Ross, F. Fahroo, and J. Strizzi, "Adaptive grids for trajectory optimization by pseudospectral methods," in *AAS/AIAA Spaceflight Mechanics Conference*, Ponce, Puerto Rico, 2003, pp. 649–668.

[5] I. M. Ross and F. Fahroo, "Pseudospectral knotting methods for solving optimal control problems," *Journal of Guidance, Control, and Dynamics*, vol. 27, pp. 397–405, 2004.

[6] Q. Gong and I. M. Ross, "Autonomous pseudospectral knotting methods for space mission optimization," in *16th AAS/AIAA Space Flight Mechanics Meeting*, no. AAS 06-151, 2006.

[7] T. Binder, L. Blank, W. Dahmen, and W. Marquardt, "Grid refinement in multiscale dynamic optimization," RWTH Aachen, Aachen, Germany, Tech. Rep. LPT-2000-11, 2000.

[8] T. Binder, A. Cruse, C. A. C. Villar, and W. Marquardt, "Dynamic optimization using a wavelet based adaptive control vector parametrization strategy," *Computers and Chemical Engineering*, vol. 24, pp. 1201–1207, 2000.

[9] T. Binder, L. Blank, W. Dahmen, and W. Marquardt, "Iterative algorithms for multiscale state estimation, Part 1: Concepts," *Journal of Optimization Theory and Applications*, vol. 111, pp. 501–527, 2001.

[10] ——, "Iterative algorithms for multiscale state estimation, Part 2: Numerical investigations," *Journal of Optimization Theory and Applications*, vol. 111, pp. 529–551, 2001.

[11] M. Schlegel, K. Stockmann, T. Binder, and W. Marquardt, "Dynamic optimization using adaptive control vector parameterization," *Computers and Chemical Engineering*, vol. 29, pp. 1731–1751, 2005.

[12] F. Fahroo and I. Ross, "A spectral patching method for direct trajectory optimization," *Journal of the Astronautical Sciences*, vol. 48, pp. 269–286, 2000.

[13] G. Elnagar, M. A. Kazemi, and M. Razzaghi, "The pseudospectral Legendre method for discretizing optimal control problems," *IEEE Transactions on Automatic Control*, vol. 40, pp. 1793–1796, 1995.

[14] F. Fahroo and I. M. Ross, "Direct trajectory optimization by a Chebyshev pseudospectral method," *Journal of Guidance, Control, and Dynamics*, vol. 25, pp. 160–166, 2002.

[15] S. Jain and P. Tsiotras, "A solution of the time-optimal Hamilton-Jacobi-Bellman equation on the interval using wavelets," in *Proceedings of 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, 2004, pp. 2728–2733.

[16] ——, "The method of reflection for solving the time-optimal Hamilton-Jacobi-Bellman equation on the interval," in *13th IEEE Mediterranean Conference on Control and Automation*, Limassol, Cyprus, 2005, pp. 1062–1067.

[17] S. Jain, P. Tsiotras, and H.-M. Zhou, "Adaptive multiresolution mesh refinement for the solution of evolution PDEs," in *Proceedings of 46th IEEE Conference on Decision and Control*, New Orleans, LA, 2007, accepted.

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

[19] P. E. Gill, W. Murray, and M. A. Saunders, *User's Guide for SNOPT Version* 6, *A Fortran Package for Large-Scale Nonlinear Programming*, Stanford, CA, 2002.

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

[21] J. C. Butcher, "Implicit Runge–Kutta processes," *Mathematics of Computation*, vol. 18, pp. 50–64, 1964.

[22] I. M. Ross, "User's manual for DIDO (ver. pr.1β): A matlab application package for solving optimal control problems," Naval Postgraduate School, Monterey, CA, Tech. Rep. 04-01.0, 2004.