[2]Frank, P. M., and Ding, X., "Survey of Robust Residual Generation and Evaluation Methods in Observer Based Fault Detection Systems," *Journal of Process Control*, Vol. 7, No. 6, 1997, pp. 403–424.

[3]Campa, G., Fravolini, M. L., Napolitano, M. R., Del Gobbo, D., Seanor, B., Yu, G., and Gururajan, S., "On-Line Learning Neural Networks for Sensor Validation of a B777 Flying Model," *International Journal of Robust and Nonlinear Control*, Vol. 12, No. 11, 2002, pp. 987–1007.

[4]Lu, Y., Sundararajan, N., and Saratchandran, P., "Analysis of Minimal Radial Basis Function Network Algorithm for Real-Time Identification of Non-Linear Dynamic Systems," *IEE Proceedings on Control Theory and Application*, Vol. 4, No. 147, 2000, pp. 476–484.

Application of Pseudospectral Methods for Receding Horizon Control

Paul Williams*
Royal Melbourne Institute of Technology, Melbourne, Victoria 3001, Australia

Introduction

THE design of controllers for nonlinear systems is not always easy, and it is often proposed to segment trajectory optimization problems into offline and online control tasks, particularly for complex systems.[1] The offline phase is devoted to solving the full nonlinear optimal control problem, whereas the online phase usually involves employing a neighboring optimal feedback control strategy based on the linearized dynamics. Because the optimal trajectory is time varying, the neighboring feedback controller traditionally relies on solving the time-varying two-point boundary value problem by a backward sweep of the Riccati equation or by transition matrices.[2] This is often both time consuming and numerically unstable. Furthermore, there is no guarantee that perturbations around the reference trajectory will be small, and so applying the linearized equations may not be valid in some cases. To overcome these difficulties Ohtsuka and Fujii[3] proposed a receding-horizon control strategy for nonlinear systems based on the stabilized continuation method. Ohtsuka[4] has extended the method to time-varying systems. This method relies on the explicit integration of the states and costates derived from the calculus of variations and requires fine tuning the stabilization parameters to obtain satisfactory convergence. Yan et al.[5] present a method for solving linear quadratic optimal control problems by transforming the linear time-varying equations into a set of discrete linear algebraic equations using the Legendre pseudospectral method. Yan et al.[6] use the approach of Ref. 5 to generate the inner feedback loop of a two-degree-of-freedom control system. Their method is based on a neighboring optimal control strategy to minimize deviations from the optimal trajectory. Lu[1] has proposed a receding-horizon strategy for precision entry guidance based on an Euler–Simpson approximation of the quadratic programming (QP) problem. Lu[1] employs a Simpson approximation for the integral cost and Euler approximations for the state derivatives. In this way, Lu solves the QP problem analytically to obtain approximate control laws. Whereas the methods proposed by Yan et al.[5,6] and Lu[1] do not require any explicit integration, they are still based on the assumption that deviations from the reference trajectory are within the linear approximation.

Received 10 September 2003; revision received 9 November 2003; accepted for publication 18 November 2003.

*Ph.D. Candidate, School of Aerospace, Mechanical, and Manufacturing Engineering, 15 Ellindale Avenue, McKinnon; tethers@hotmail.com. Student Member AIAA.

In this Note, a method for solving nonlinear receding-horizon control problems by pseudospectral approximations is presented. The proposed solution method approximates the nonlinear receding-horizon control problem with successive linear approximations obtained via the method of quasi linearization. Each linear problem is solved by discretizing the two-point boundary value problem obtained from the calculus of variations using a Jacobi pseudospectral method. This allows the linear problem to be solved using matrix algebra. The approach is very similar to that used by Yan et al.,[5] except that a quasi linearization algorithm is first applied to the nonlinear problem. Another important difference between the proposed method and that of Yan et al.[5] is the way that the boundary conditions are incorporated. Yan et al.[5] use a matrix partitioning approach to obtain feedback control laws from an overdetermined set of equations. In this Note, however, the boundary conditions are applied directly in a similar manner to that done when solving partial differential equations by pseudospectral methods.[7] The method is applied to the difficult problem of retrieving a subsatellite using a flexible tether.

Theory

Consider the nonlinear receding horizon control problem of finding the control input $\boldsymbol{u}(t)$ that minimizes the performance index

$$J = \frac{1}{2}[\boldsymbol{M}_f\boldsymbol{x} - \psi]^T\boldsymbol{S}_f[\boldsymbol{M}_f\boldsymbol{x} - \psi]|_{t+T} + \frac{1}{2}\int_t^{t+T} \times \left\{\begin{bmatrix}(\boldsymbol{x}-\boldsymbol{x}_d)^T & (\boldsymbol{u}-\boldsymbol{u}_d)^T\end{bmatrix}\begin{bmatrix}\boldsymbol{Q} & \boldsymbol{N}\\ \boldsymbol{N}^T & \boldsymbol{R}\end{bmatrix}\begin{bmatrix}(\boldsymbol{x}-\boldsymbol{x}_d)\\(\boldsymbol{u}-\boldsymbol{u}_d)\end{bmatrix}\right\}\mathrm{d}\tau \quad (1)$$

subject to the state equations and initial conditions

$$\dot{\boldsymbol{x}} = f[\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau], \qquad \boldsymbol{x}(\tau = t) = \boldsymbol{x}(t) \quad (2)$$

where $\boldsymbol{x}(\tau) \in R^n$ is the state vector, $\boldsymbol{x}_d(\tau) \in R^n$ is the desired state vector, $\boldsymbol{u}(\tau) \in R^m$ is the vector of control inputs, $\boldsymbol{u}_d(\tau) \in R^m$ is the desired control input, $t \in R$ is the time, $\tau \in [t, t+T] \in R$ is the time variable used to predict the future system state, $\psi \in R^p$, $p \leq n$, is the desired vector of values for the linear combination of desired final states $[\boldsymbol{M}_f\boldsymbol{x}(t+T)]$, $\boldsymbol{Q}$ is a positive semidefinite matrix, $\boldsymbol{N}$ is a given matrix, $\boldsymbol{R}$ is a positive definite matrix, $\boldsymbol{S}_f$ is the positive semidefinite terminal weighting matrix, $\boldsymbol{M}_f$ is a given matrix, and T is the length of the future horizon. Note that there is a distinction between the state and control variables used for predicting the future state, parameterized by τ, and the actual state and control variables, parameterized by t. Once the problem defined by Eqs. (1) and (2) is solved, the actual control input is applied at the current time by $\boldsymbol{u}(t) = \boldsymbol{u}(\tau = t)$.

To solve this problem, we employ the method of quasi linearization in a manner similar to that by Jaddu[8] and Xu and Agrawal.[9] In this method, the performance index is expanded up to second order and the system equations are expanded to first order around nominal trajectories. The solution to the original problem is obtained by solving the sequence of resulting linear optimal control problems. This differs from other methods of quasi linearization that first apply the necessary conditions for optimality and then linearization, where an update parameter is employed to aid convergence.[10] Application of the method of quasi linearization to Eqs. (1) and (2) results in the following sequence of linear optimal control problems: To minimize

$$\tilde{J} = \frac{1}{2}[\boldsymbol{M}_f\tilde{\boldsymbol{x}} - \psi]^T\boldsymbol{S}_f[\boldsymbol{M}_f\tilde{\boldsymbol{x}} - \psi]|_{t+T} + \frac{1}{2}\int_t^{t+T} \times \left\{\begin{bmatrix}(\tilde{\boldsymbol{x}}-\boldsymbol{x}_d)^T & (\tilde{\boldsymbol{u}}-\boldsymbol{u}_d)^T\end{bmatrix}\begin{bmatrix}\boldsymbol{Q} & \boldsymbol{N}\\ \boldsymbol{N}^T & \boldsymbol{R}\end{bmatrix}\begin{bmatrix}(\tilde{\boldsymbol{x}}-\boldsymbol{x}_d)\\(\tilde{\boldsymbol{u}}-\boldsymbol{u}_d)\end{bmatrix}\right\}\mathrm{d}\tau \quad (3)$$

subject to

$$\dot{\tilde{\boldsymbol{x}}} = \underset{\sim}{\boldsymbol{A}}(\tau)\tilde{\boldsymbol{x}} + \underset{\sim}{\boldsymbol{B}}(\tau)\tilde{\boldsymbol{u}} + \underset{\sim}{\boldsymbol{w}}(\tau), \qquad \tilde{\boldsymbol{x}}(\tau = t) = \tilde{\boldsymbol{x}}(t) \quad (4)$$

where

$$\underset{\sim}{A}(\tau) = \left.\frac{\partial f[\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau]}{\partial \boldsymbol{x}}\right|_{\underset{\sim}{x}(\tau),\underset{\sim}{u}(\tau)} \tag{5}$$

$$\underset{\sim}{B}(\tau) = \left.\frac{\partial f[\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau]}{\partial \boldsymbol{u}}\right|_{\underset{\sim}{x}(\tau),\underset{\sim}{u}(\tau)} \tag{6}$$

$$\underset{\sim}{w}(\tau) = f[\underset{\sim}{x}(\tau), \underset{\sim}{u}(\tau), \tau] - \underset{\sim}{A}(\tau)\underset{\sim}{x} - \underset{\sim}{B}(\tau)\underset{\sim}{u} \tag{7}$$

and $\underset{\sim}{x}(\tau), \underset{\sim}{u}(\tau)$ are the state and control vectors from the preceding iteration, whereas $\tilde{\boldsymbol{x}}(\tau)$ and $\tilde{\boldsymbol{u}}(\tau)$ are the state and control vectors for the current iteration, respectively.

The Hamiltonian for the linear problem is given by

$$\mathcal{H} = \tfrac{1}{2}\Big[(\tilde{\boldsymbol{x}} - \boldsymbol{x}_d)^T\boldsymbol{Q}(\tilde{\boldsymbol{x}} - \boldsymbol{x}_d) + 2(\tilde{\boldsymbol{x}} - \boldsymbol{x}_d)^T\boldsymbol{N}(\tilde{\boldsymbol{u}} - \boldsymbol{u}_d) + (\tilde{\boldsymbol{u}} - \boldsymbol{u}_d)^T\boldsymbol{R}(\tilde{\boldsymbol{u}} - \boldsymbol{u}_d)\Big] + \tilde{\boldsymbol{\lambda}}^T[\underset{\sim}{A}(\tau)\tilde{\boldsymbol{x}} + \underset{\sim}{B}(\tau)\tilde{\boldsymbol{u}} + \underset{\sim}{w}(\tau)] \tag{8}$$

where $\tilde{\boldsymbol{\lambda}} \in R^n$ is the vector of costates for the current iteration. The costate equations and the corresponding boundary conditions are obtained from the calculus of variations,[2]

$$\dot{\tilde{\boldsymbol{\lambda}}} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}} = -\big[\boldsymbol{Q}(\tilde{\boldsymbol{x}} - \boldsymbol{x}_d) + \boldsymbol{N}(\tilde{\boldsymbol{u}} - \boldsymbol{u}_d) + \underset{\sim}{A}^T(\tau)\tilde{\boldsymbol{\lambda}}\big]$$
$$\tilde{\boldsymbol{\lambda}}(t+T) = \boldsymbol{M}_f^T\boldsymbol{S}_f[\boldsymbol{M}_f\tilde{\boldsymbol{x}}(t+T) - \psi] \tag{9}$$

The optimal control input is given by the stationary condition

$$\frac{\partial \mathcal{H}}{\partial \boldsymbol{u}} = 0 \Rightarrow \tilde{\boldsymbol{u}}(\tau) = \boldsymbol{u}_d - \boldsymbol{R}^{-1}\big[\underset{\sim}{B}^T(\tau)\tilde{\boldsymbol{\lambda}} + \boldsymbol{N}^T(\tilde{\boldsymbol{x}} - \boldsymbol{x}_d)\big] \tag{10}$$

Hence, the necessary conditions for the optimal control are given as follows

$$\dot{\tilde{\boldsymbol{x}}} = \underset{\sim}{A}(\tau)\tilde{\boldsymbol{x}} + \underset{\sim}{B}(\tau)\boldsymbol{u}_d - \underset{\sim}{B}(\tau)\boldsymbol{R}^{-1}\big\{\underset{\sim}{B}^T(\tau)\tilde{\boldsymbol{\lambda}} + \boldsymbol{N}^T[\tilde{\boldsymbol{x}} - \boldsymbol{x}_d]\big\} + \underset{\sim}{w}(\tau)$$
$$\dot{\tilde{\boldsymbol{\lambda}}} = -\boldsymbol{Q}[\tilde{\boldsymbol{x}} - \boldsymbol{x}_d] - \boldsymbol{N}[\tilde{\boldsymbol{u}} - \boldsymbol{u}_d] - \underset{\sim}{A}^T(\tau)\tilde{\boldsymbol{\lambda}} \tag{11}$$

with the boundary conditions

$$\tilde{\boldsymbol{x}}(\tau = t) = \boldsymbol{x}(t), \qquad \tilde{\boldsymbol{\lambda}}(t+T) = \boldsymbol{M}_f^T\boldsymbol{S}_f[\boldsymbol{M}_f\tilde{\boldsymbol{x}}(t+T) - \psi] \tag{12}$$

Equations (11) and (12) constitute a linear two-point boundary value problem (TPBVP). The TPBVP may be solved algebraically by pseudospectral methods following the idea used by Yan et al.[5] In this approach, the state and costate variables are expanded into Nth-degree Lagrange polynomials with the values of the states and costates at the Jacobi–Gauss–Lobatto (JGL) points as the expansion coefficients

$$\boldsymbol{x}_N(\xi) = \sum_{j=0}^{N}\boldsymbol{x}_j\phi_j(\xi), \qquad \boldsymbol{\lambda}_N(\xi) = \sum_{j=0}^{N}\boldsymbol{\lambda}_j\phi_j(\xi), \qquad \xi \in [-1, 1] \tag{13}$$

where $\phi_j(\xi)$ are the Lagrange interpolating polynomials. When this technique is used, the derivatives of the interpolating polynomials may be calculated exactly by way of a differentiation matrix. That is, we have

$$\dot{\tilde{\boldsymbol{x}}}(t_k) = \frac{2}{T}\sum_{j=0}^{N}D_{kj}\tilde{\boldsymbol{x}}_j, \qquad \dot{\tilde{\boldsymbol{\lambda}}}(t_k) = \frac{2}{T}\sum_{j=0}^{N}D_{kj}\tilde{\boldsymbol{\lambda}}_j \qquad k = 0, \ldots, N \tag{14}$$

where $t_k = (T/2)\xi_k + t + (T/2)$, $k = 0, \ldots, N$, and ξ_k, $k = 0, \ldots, N$, are the extrema of the Nth-degree Jacobi polynomial $P_N^{(\alpha,\beta)}$ on the interval $[-1, 1]$, including the endpoints, and D_{kj} are entries of the $(N+1) \times (N+1)$ differentiation matrix,[11]

$$D_{kj} = \begin{cases} \dfrac{\alpha - N(N+\alpha+\beta+1)}{2(\beta+2)} & k = j = 0 \\ \dfrac{N(N+\alpha+\beta+1) - \beta}{2(\alpha+2)} & k = j = N \\ \dfrac{P_N^{(\alpha,\beta)}(\tau_k)}{P_N^{(\alpha,\beta)}(\tau_j)}\dfrac{1}{(\tau_k - \tau_j)} & 1 \le k \ne j \le N-1 \\ \dfrac{(\alpha+\beta)\tau_k + \alpha - \beta}{2\left(1 - \tau_k^2\right)} & 1 \le k = j \le N-1 \\ -\dfrac{P_N^{(\alpha,\beta)}(\tau_0)}{P_N^{(\alpha,\beta)}(\tau_j)}\dfrac{1}{(\beta+1)(1+\tau_j)} & k = 0, 1 \le j \le N-1 \\ \dfrac{P_N^{(\alpha,\beta)}(\tau_N)}{P_N^{(\alpha,\beta)}(\tau_j)}\dfrac{1}{(\alpha+1)(1-\tau_j)} & k = N, 1 \le j \le N-1 \\ \dfrac{P_N^{(\alpha,\beta)}(\tau_j)}{P_N^{(\alpha,\beta)}(\tau_0)}\dfrac{(\beta+1)}{(1+\tau_j)} & j = 0, 1 \le k \le N-1 \\ -\dfrac{P_N^{(\alpha,\beta)}(\tau_j)}{P_N^{(\alpha,\beta)}(\tau_N)}\dfrac{(\alpha+1)}{(1-\tau_j)} & j = N, 1 \le k \le N-1 \\ -\dfrac{(\alpha+1)}{2(\beta+1)}\dfrac{P_N^{(\alpha,\beta)}(\tau_0)}{P_N^{(\alpha,\beta)}(\tau_N)} & k = 0, j = N \\ \dfrac{(\beta+1)}{2(\alpha+1)}\dfrac{P_N^{(\alpha,\beta)}(\tau_N)}{P_N^{(\alpha,\beta)}(\tau_0)} & k = N, j = 0 \end{cases} \tag{15}$$

Note that the factor $2/T$ in Eqs. (14) arises from the transformation from the computational domain $[-1, 1]$ to the physical domain $[t, t+T]$. Also note that Yan et al.[5] employ a Legendre differentiation matrix rather than the more general Jacobi differentiation matrix given by Eq. (15), although any differentiation matrix could be used in the Yan et al.[5] formulation. Because the differentiation matrix relates the derivative of the states and costates to the values of the states and costates via a linear matrix, the state and costate equations can be written as a set of linear algebraic equations by collocating at the JGL nodes as follows:

$$\frac{2}{T}\sum_{j=0}^{N}D_{kj}\tilde{\boldsymbol{x}}(t_j) - \big\{\underset{\sim}{A}(t_k) - \underset{\sim}{B}(t_k)\boldsymbol{R}^{-1}\boldsymbol{N}^T\big\}\tilde{\boldsymbol{x}}(t_k) + \underset{\sim}{B}(t_k)\boldsymbol{R}^{-1}\underset{\sim}{B}^T(t_k)\tilde{\boldsymbol{\lambda}}(t_k) = \underset{\sim}{B}(t_k)\boldsymbol{u}_d(t_k) + \underset{\sim}{B}(t_k)\boldsymbol{R}^{-1}\boldsymbol{N}^T\boldsymbol{x}_d(t_k) + \underset{\sim}{w}(t_k) \qquad k = 0, \ldots, N \tag{16}$$

$$\frac{2}{T}\sum_{j=0}^{N}D_{kj}\tilde{\boldsymbol{\lambda}}(t_j) + \{\boldsymbol{Q} - \boldsymbol{N}\boldsymbol{R}^{-1}\boldsymbol{N}^T\}\tilde{\boldsymbol{x}}(t_k) + \big\{\underset{\sim}{A}^T(t_k) - \boldsymbol{N}\boldsymbol{R}^{-1}\underset{\sim}{B}(t_k)\big\}\tilde{\boldsymbol{\lambda}}(t_k) = \{\boldsymbol{Q} - \boldsymbol{N}\boldsymbol{R}^{-1}\boldsymbol{N}^T\}\boldsymbol{x}_d(t_k) \qquad k = 0, \ldots, N \tag{17}$$

The boundary conditions may be incorporated by replacing the first n equations in Eq. (16) with

$$\tilde{x}(t_0) = \boldsymbol{x}(t) \tag{18}$$

and the last n equations in Eq. (17) with

$$\tilde{\boldsymbol{\lambda}}(t_N) - \boldsymbol{M}_f^T\boldsymbol{S}_f\boldsymbol{M}_f\tilde{\boldsymbol{x}}(t_N) = -\boldsymbol{M}_f^T\boldsymbol{S}_f\psi \tag{19}$$

This differs slightly from the method used by Yan et al.,[5] who solve the set of overdetermined equations [Eqs. (16–19)] using matrix partioning and QR decomposition, whereas in this work the state and costate equations at the boundaries are replaced with the appropriate boundary conditions. Equations (16–19) may be written in matrix form as

$$\mathcal{A}z = \mathcal{B} \tag{20}$$

where

b) Update the control trajectory using the solution from step 4a and Eq. (10).

c) Repeat steps 4a and 4b using the solution from the previous iteration as initial guess until the norm of the change in trajectory is within a prescribed tolerance or a given number of iterations is exceeded.

$$\mathcal{A} = \begin{bmatrix} \mathcal{A}^{xx} & \mathcal{A}^{x\lambda} \\ \mathcal{A}^{\lambda x} & \mathcal{A}^{\lambda\lambda} \end{bmatrix}, \qquad \mathcal{B} = \{\mathcal{B}^{x}, \mathcal{B}^{\lambda}\}^{T} \tag{21}$$

$$z = \{\tilde{\boldsymbol{x}}(t_0), \tilde{\boldsymbol{x}}(t_1), \ldots, \tilde{\boldsymbol{x}}(t_N), \tilde{\boldsymbol{\lambda}}(t_0), \tilde{\boldsymbol{\lambda}}(t_1), \ldots, \tilde{\boldsymbol{\lambda}}(t_N)\}^{T} \tag{22}$$

$$\mathcal{A}^{xx} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{D}_{10} & \boldsymbol{D}_{11} - \underset{\sim}{\boldsymbol{A}}(t_1) + \underset{\sim}{\boldsymbol{B}}(t_1)\boldsymbol{R}^{-1}\boldsymbol{N}^{T} & \boldsymbol{D}_{12} & \cdots & \boldsymbol{D}_{1N} \\ \boldsymbol{D}_{20} & \boldsymbol{D}_{21} & \boldsymbol{D}_{22} - \underset{\sim}{\boldsymbol{A}}(t_2) + \underset{\sim}{\boldsymbol{B}}(t_2)\boldsymbol{R}^{-1}\boldsymbol{N}^{T} & \cdots & \boldsymbol{D}_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{D}_{N0} & \boldsymbol{D}_{N1} & \boldsymbol{D}_{N2} & \cdots & \boldsymbol{D}_{NN} - \underset{\sim}{\boldsymbol{A}}(t_N) + \underset{\sim}{\boldsymbol{B}}(t_N)\boldsymbol{R}^{-1}\boldsymbol{N}^{T} \end{bmatrix} \tag{23}$$

$$\mathcal{A}^{x\lambda} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \underset{\sim}{\boldsymbol{B}}(t_1)\boldsymbol{R}^{-1}\underset{\sim}{\boldsymbol{B}}^{T}(t_1) & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \underset{\sim}{\boldsymbol{B}}(t_2)\boldsymbol{R}^{-1}\underset{\sim}{\boldsymbol{B}}^{T}(t_2) & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \underset{\sim}{\boldsymbol{B}}(t_N)\boldsymbol{R}^{-1}\underset{\sim}{\boldsymbol{B}}^{T}(t_N) \end{bmatrix} \tag{24}$$

$$\mathcal{A}^{\lambda x} = \begin{bmatrix} \boldsymbol{Q} - \boldsymbol{N}\boldsymbol{R}^{-1}\boldsymbol{N}^{T} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q} - \boldsymbol{N}\boldsymbol{R}^{-1}\boldsymbol{N}^{T} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{Q} - \boldsymbol{N}\boldsymbol{R}^{-1}\boldsymbol{N}^{T} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & -\boldsymbol{M}_f^{T}\boldsymbol{S}_f\boldsymbol{M}_f \end{bmatrix} \tag{25}$$

$$\mathcal{A}^{\lambda\lambda} = \begin{bmatrix} \boldsymbol{D}_{00} + \underset{\sim}{\boldsymbol{A}}^{T}(t_0) - \boldsymbol{N}\boldsymbol{R}^{-1}\underset{\sim}{\boldsymbol{B}}^{T}(t_0) & \boldsymbol{D}_{01} & \boldsymbol{D}_{02} & \cdots & \boldsymbol{D}_{0N} \\ \boldsymbol{D}_{10} & \boldsymbol{D}_{11} + \underset{\sim}{\boldsymbol{A}}^{T}(t_1) - \boldsymbol{N}\boldsymbol{R}^{-1}\underset{\sim}{\boldsymbol{B}}^{T}(t_1) & \boldsymbol{D}_{12} & \cdots & \boldsymbol{D}_{1N} \\ \vdots & \vdots & \ddots & \cdots & \vdots \\ \boldsymbol{D}_{(N-1)0} & \boldsymbol{D}_{(N-1)1} & \cdots & \boldsymbol{D}_{(N-1)(N-1)} + \underset{\sim}{\boldsymbol{A}}^{T}(t_{N-1}) - \boldsymbol{N}\boldsymbol{R}^{-1}\underset{\sim}{\boldsymbol{B}}^{T}(t_{N-1}) & \boldsymbol{D}_{(N-1)N} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} \end{bmatrix} \tag{26}$$

$$\mathcal{B}^{x} = \left\{\boldsymbol{x}(t), \underset{\sim}{\boldsymbol{B}}(t_1)\boldsymbol{u}_d(t_1) + \underset{\sim}{\boldsymbol{B}}(t_1)\boldsymbol{R}^{-1}\boldsymbol{N}^{T}\boldsymbol{x}_d(t_1) + \underset{\sim}{\boldsymbol{w}}(t_1), \ldots, \underset{\sim}{\boldsymbol{B}}(t_N)\boldsymbol{u}_d(t_N) + \underset{\sim}{\boldsymbol{B}}(t_N)\boldsymbol{R}^{-1}\boldsymbol{N}^{T}\boldsymbol{x}_d(t_N) + \underset{\sim}{\boldsymbol{w}}(t_N)\right\}$$

$$\mathcal{B}^{\lambda} = \left\{(\boldsymbol{Q} - \boldsymbol{N}\boldsymbol{R}^{-1}\boldsymbol{N}^{T})\boldsymbol{x}_d(t_0), \ldots, (\boldsymbol{Q} - \boldsymbol{N}\boldsymbol{R}^{-1}\boldsymbol{N}^{T})\boldsymbol{x}_d(t_{N-1}), -\boldsymbol{M}_f^{T}\boldsymbol{S}_f\psi\right\} \tag{27}$$

and $\boldsymbol{I}$ is an $n \times n$ identity matrix, $\boldsymbol{0}$ is an $n \times n$ zero matrix, and $\boldsymbol{D}_{kj} = [2D_{kj}/T]\boldsymbol{I}$.

The solution to Eq. (20) may be obtained using straightforward techniques from linear algebra. The procedure for implementing the receding-horizon control for the nonlinear problem may be summarized as follows.

1) Begin with receding horizon initialization: Select a suitable horizon length T and desired trajectory.

2) Apply quasi linearization: Expand the system dynamics to first order using Eqs. (4–7) and guess nominal trajectories for the interval $[t, t+T]$. A suitable guess would be the reference or desired trajectory.

3) Apply pseudospectral discretization: Select an appropriate discretization level N and the Jacobi parameters α and β. These parameters are problem dependent and may need to be adjusted to achieve the desired combination of speed and accuracy.

4) Solve the sequence of TPBVP:

a) Solve the linear system of Eqs. (20).

5) Apply receding-horizon controller: Apply the control input at the current time by interpolating the converged controls from step 4, and repeat steps 2–4 until $t = t_f$.

The proposed controller is relatively simple to program and does not require any explicit trajectory propagation techniques. In addition, the form of the equations is identical for time-varying and linear systems, which should be contrasted with the technique proposed by Ohtuska,[4] which requires significant modifications for implementation with each new system. In the case where the system dynamics are linear, step 4c may be omitted because convergence is achieved in one iteration. The proposed control method is attractive from the point of view of accuracy and computational speed. It retains the accuracy of indirect trajectory optimization techniques, while reducing the computation time by converting the TPBVP to a system of linear algebraic equations.

The proposed method combines a spectral method with quasi linearization. Sufficient conditions for the existence of a solution require that[9] 1) f is continuously differentiable, 2) $\boldsymbol{S}_f$ is positive semidefinite, 3) $\boldsymbol{R}$ is positive definite, and 4) $\boldsymbol{Q}$ is positive semidefinite. Xu and Agrawal[9] have proved that the method of quasi linearization converges to the solution of the nonlinear problem for systems that exhibit polynomial representations under the condition that $\boldsymbol{R}$ is sufficiently large. If the preceding assumptions are satisfied, then according to standard convergence results from spectral theory,[12,13] the proposed algorithm will converge at a spectral rate with N.

Numerical Example

The problem of retrieving a satellite connected to a large space platform by a flexible tether is considered. Two separate models of the tether system are used in this Note. The first, which is used in the control law, models the tether as a massless, inelastic rod. The only control input to the system is considered to be changes in tether length implemented by tension control at the deployer. The system under consideration is shown in Fig. 1. The mass of the subsatellite is considered to be negligible compared to the tether platform, and consequently, the tether platform is assumed to remain in an unperturbed Keplerian orbit. The equations of motion for the system may be derived via Lagrange's equations, and the resulting nondimensional equations for the in-plane motion are given by (see Ref. 14)

$$\theta'' = -2(\theta' + 1)(\Lambda'/\Lambda) - 3\sin\theta\cos\theta$$

$$\Lambda'' = \Lambda[(\theta' + 1)^2 + 3\cos^2\theta - 1] - u \tag{28}$$

where θ is the in-plane libration angle, $\Lambda(= l/L)$ is the nondimensional tether length, l is the tether length, L is the nominal deployed tether length, $u(= T/(m\omega^2 L))$ is the nondimensional tension control input, m is the mass of the subsatellite, $(\,)' = \mathrm{d}/\mathrm{d}(\omega t)$ is the nondimensional time derivative, and ω is the orbital angular velocity of the tether platform. The second model, which is used to simulate the controller, treats both the tether mass and flexibility by discretizing the tether into a series of point masses connected by springs.[14]

To demonstrate the speed of the proposed method compared to using a commerical nonlinear solver such as NPSOL,[15] we first consider the problem of minimizing the performance index

$$J = [10\theta^2 + 50(\Lambda - 1)^2]_{\tau=6} + \frac{1}{2}\int_0^6 [10\theta^2 + 5\theta'^2 + 10(\Lambda - 1)^2 + \Lambda'^2 + (u - 3)^2]\,\mathrm{d}\tau \tag{29}$$

subject to the state equations (28) and the initial conditions $(\theta, \theta', \Lambda, \Lambda')|_{\tau=0} = [-50\text{ deg}, 0, 0.3, 0]$. A comparison of the cost functions and computation time using the Legendre pseudospectral method (see Ref. 16) in NPSOL[15] with analytical Jacobians and default tolerances is presented in Table 1. Initial guesses for the trajectories are selected as $(\theta, \theta', \Lambda, \Lambda', u) = (0, 0, 1, 0, 3)$. Convergence for the quasi linearization algorithm is set as $1e$–7 for the L_2 norm of the change in trajectory at the JGL points. Both methods are implemented on a Pentium 4 2.4-GHz personal computer

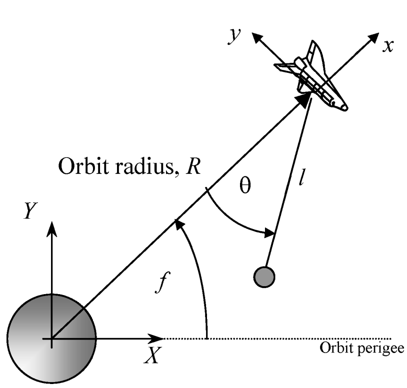

Fig. 1 Simplified representation of tethered satellite system.

Table 1 Comparison of cost function and CPU time for stationkeeping example

	NPSOL		Proposed method	
N	J	CPU, s	J	CPU, s
20	11.0382090	2.74	——	——
30	11.0312212	9.32	11.0275088	0.56
40	11.0311210	22.95	11.0310676	1.04
50	11.0311207	44.91	11.0311297	1.61
60	11.0311207	78.47	11.0311276	2.39

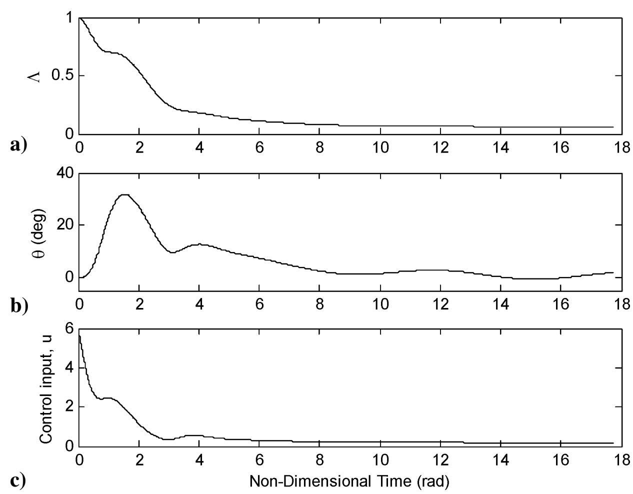

Fig. 2 Retrieval of subsatellite with a flexible tether using receding horizon control: a) nondimensional tether length, b) inplane tether libration angle, and c) nondimensional tether tension.

in MATLAB® 6.5. Table 1 illustrates that the proposed method is computationally superior to NPSOL and is extremely competitive in terms of the performance index. Note, however, that the quasi linearization algorithm did not converge for $N = 20$, whereas the direct method does. The computation time is still significantly longer for $N = 20$ for the direct method (2.74 s) than the proposed method for $N = 30$ (0.56 s). The fact that the algorithm did not converge for small N is a limitation of the method.

Retrieval control is recognized as more difficult than deployment control due to its inherent instability. The proposed controller is applied for the generation of an optimal feedback path online. To generate an optimal trajectory online, it is necessary to select the control parameters carefully. The control task is set as follows:

$$\boldsymbol{x}_0 = \{0, 0, 1, 0\}^T, \qquad \boldsymbol{x}_f = \{0, 0, 0.05, 0\}^T \tag{30}$$

with

$$\boldsymbol{S}_f = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{31}$$

$$\boldsymbol{Q} = \boldsymbol{I}, \qquad R = 1, \qquad M_f = 1 \tag{32}$$

The desired state is set as the desired final state, $\boldsymbol{x}_d = \boldsymbol{x}_f$, and the desired control input is set as the control required to maintain a constant length tether in the final equilibrium configuration, that is, $u_d = 0.15$. The horizon length is set as $T = 2$ rad, and the degree of discretization is set as $N = 30$. Numerical investigation reveals that, for $N = 30$, $\alpha = \beta = 0.94$ is optimum in terms of the rate of convergence for the tether problem. The control input is numerically integrated using ode45 in MATLAB with error tolerances set to 10^{-12}. The controller is implemented at nondimensional sampling times of

0.01 rad and not in a continuous manner. In this paper, we have ignored the computational delay between solving the control problem and implementation of the control because we are mainly concerned with the performance of the controller. Simulation parameters for the flexible model are selected as initial tether length = 20 km, tether stiffness, $EA = 60{,}000$ N, subsatellite mass = 500 kg, tether mass density = 1 kg/km, and orbital altitude = 400 km.

Numerical results are presented in Fig. 2. Figure 2 demonstrates the closed-loop stability of the retrieval process using the proposed control algorithm with a flexible tether. The retrieval time is approximately two orbits, although it may be noted from Fig. 2a that the main portion of the retrieval process is accomplished in approximately one orbit. It can also be seen that the control input and length variation is very smooth and that the tether tension remains positive throughout the retrieval process. It is evident that the proposed controller is able to treat effectively the unmodeled effects of tether mass and flexibility.

Conclusions

A new method for solving nonlinear receding-horizon control problems is proposed. The method does not require intensive computation and relies only on the solution of simultaneous linear equations. The algorithm is shown to be practical for online implementation in a tether system. The controller is demonstrated in the practical application of retrieving a flexible tether. The closed-loop performance is excellent in the face of unmodeled disturbances due to tether mass and flexibility.

References

[1] 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.

[2] Bryson, A. E., *Dynamic Optimization*, Addison–Wesley, Menlo Park, CA, 1999, pp. 315–318.

[3] Ohtsuka, T., and Fujii, H. A., "Real-Time Optimization Algorithm for Nonlinear Receding-Horizon Control," *Automatica*, Vol. 33, No. 6, 1997, pp. 1147–1154.

[4] Ohtsuka, T., "Time-Variant Receding Horizon Control of Nonlinear Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 1, 1998, pp. 174–176.

[5] Yan, H., Fahroo, F., and Ross, I. M., "Optimal Feedback Control Laws by Legendre Pseudospectral Approximations," *Proceedings of the American Control Conference*, IEEE Publications, Piscataway, NJ, 2001, pp. 2388–2393.

[6] Yan, H., Fahroo, F., and Ross, I. M., "Real-Time Computation of Neighbouring Optimal Control Laws," AIAA Paper 2002-4657, Aug. 2002.

[7] Fornberg, B., and Sloan, D.M., "A Review of Pseudospectral Methods for Solving Partial Differential Equations," *Acta Numerica*, Vol. 3, Aug. 1994, pp. 203–267.

[8] Jaddu, H., "Direct Solution of Nonlinear Optimal Control Problems Using Quasilinearization and Chebyshev Polynomials," *Journal of the Franklin Institute*, Vol. 339, Nos. 4–5, 2002, pp. 479–498.

[9] Xu, X., and Agrawal, S. K., "Finite-Time Optimal Control of Polynomial Systems Using Successive Suboptimal Approximations," *Journal of Optimization Theory and Applications*, Vol. 105, No. 2, 2000, pp. 477–489.

[10] Bryson, A. E., and Ho, Y.-C., *Applied Optimal Control*, Ginn and Co., Waltham, MA, 1969, pp. 234–236.

[11] Williams, P., "Jacobi Pseudospectral Method for Solving Optimal Control Problems," *Journal of Guidance, Control, and Dynamics* (to be published).

[12] Peyret, R., *Spectral Methods for Incompressible Viscous Flow*, Springer-Verlag, New York, 2000, pp. 70–73.

[13] Ross, I. M., and Fahroo, F., "Convergence of Pseudospectral Discretizations of Optimal Control Problems," *Proceedings of the 40th IEEE Conference on Decision and Control*, IEEE Publications, Piscataway, NJ, 2001, pp. 3175–3177.

[14] Williams, P., Blanksby, C., and Trivailo, P., "The Use of Electromagnetic Lorentz Forces as a Tether Control Actuator," 53rd International Astronautical Congress, Paper IAC-02-A.5.04, Oct. 2002.

[15] Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. A., "User's Guide to NPSOL 5.0: A Fortran Package for Nonlinear Programming," TR SOL 86-1, Stanford Optimization Lab., Stanford Univ., Stanford, CA, July 1998.

[16] Ross, I. M., and Fahroo, F., "Legendre Pseudospectral Approximations of Optimal Control Problems," *Lecture Notes in Control and Information Sciences*, Springer-Verlag (to be published).

Square Root Sigma Point Filtering for Real-Time, Nonlinear Estimation

Shelby Brunke*

University of Washington, Seattle, Washington 98195

and

Mark E. Campbell†

Cornell University, Ithaca, New York 14853

Introduction

THE objective of this Note is to present an elegant solution to a nonlinear estimation scheme that allows the development of accurate and robust system estimates in the presence of nonlinearities, noise, and faults, in real time. Traditional state estimation for linear systems has a rich, elegant theory, namely, the stochastic Kalman filter. Extension of Kalman filtering theory, especially guarantees, to nonlinear systems is nontrivial. The extended Kalman filter (EKF) (see Ref. 1) is an approximate extension of Kalman filtering techniques to nonlinear systems by linearizing the state dynamics at each time step. The EKF does not require linearity, stability, or time invariance of the system; it does require that the state dynamics are sufficiently differentiable, that is, the Jacobian exists. Although the EKF is quite flexible and well used in applications such as radar tracking and global positioning systems, disadvantages include sensitivities to required a priori information (initial state and noise environment), which can lead to bias and even divergence.[2]

One cause of the poor performance in the EKF is due to the linearization of the model dynamics.[3] A new nonlinear filter has been introduced[4] that is much simpler than the EKF, yet achieves better performance. Instead of truncating nonlinear dynamics to first order as with the EKF, the sigma point filter (SPF) [renamed from the unscented Kalman filter (see Ref. 4)] approximates the distribution of the state with a finite set of points. These points, called sigma points, are then propagated through the nonlinear dynamics. The mean and covariance of the distribution are then calculated as a weighted sum and outer product of the propagated points. Because nonlinear dynamics are used without approximation, that is, no derivatives are calculated as in the EKF, it is much simpler to implement and better results are expected. The performance of the SPF has been shown to be analytically equivalent to a truncated second-order EKF in accuracy, but without the need to calculate Jacobians or Hessians of the dynamics.[4] A schematic comparison of the SPF and EKF is given in Fig. 1.

The work here is twofold. First, a square root version of the SPF is developed. This is an elegant solution to the SPF because of its dependence on sigma points. Second, the new filter is compared to the traditional EKF using a variety of metrics to show that it is an excellent choice for real-time estimation of complex nonlinear systems. Finally, the square root SPF is implemented on a nonlinear F-15-like simulation, a good test because of the complex, uncertain, and dynamic environment, which demonstrates state/parameter estimation in the presence of a system fault.

Received 5 March 2003; revision received 5 August 2003; accepted for publication 9 November 2003.

*Mechanical Engineer; currently Scientist, Siemens Medical Solutions; shelby.brunke@siemens.com.

†Professor, Department of Mechanical and Aerospace Engineering; mc288@cornell.edu. Senior Member AIAA.