$$\bar{A} = \begin{bmatrix} -[\tilde{\ell}_{11}\boldsymbol{R}_1]^T & 0 & 0 & \cdots & 0 \\ \boldsymbol{I} & 0 & 0 & \cdots & 0 \\ [\tilde{\ell}_{12}\boldsymbol{R}_1^T]^T & -[\tilde{\ell}_{22}\boldsymbol{R}_2]^T & 0 & \cdots & 0 \\ 0 & \boldsymbol{I} & 0 & \cdots & 0 \\ 0 & [\tilde{\ell}_{23}\boldsymbol{R}_2^T]^T & -[\tilde{\ell}_{33}\boldsymbol{R}_3]^T & \cdots & 0 \\ 0 & 0 & \boldsymbol{I} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -[\tilde{\ell}_{nn}\boldsymbol{R}_n]^T \\ 0 & 0 & 0 & \cdots & \boldsymbol{I} \end{bmatrix} \tag{11}$$

which in turn confirms the orthogonality, viz. $\bar{\boldsymbol{B}}\bar{\boldsymbol{A}} = 0$. As can be seen from Eq. (11), the above null-space transformation results in a band matrix $\bar{\boldsymbol{A}}$ whose expression is simpler in appearance to that given in Eq. (5). Similar to Eq. (4), Eq. (11) is thus capable of nullifying the new Jacobian matrix $\bar{\boldsymbol{B}}$ on an orthogonal basis while simultaneously generating a new set of independent variables $\bar{\xi}_\omega$ after the null-space transformation. To eliminate the new Jacobian matrix $\bar{\boldsymbol{B}}$ in Eq. (9), Eq. (9) is premultiplied with the transpose of matrix $\bar{\boldsymbol{A}}$:

$$\bar{\boldsymbol{A}}^T\bar{\boldsymbol{M}}\bar{\boldsymbol{A}}\ddot{\bar{\xi}}_\omega = \bar{\boldsymbol{A}}^T\bar{\boldsymbol{F}} - \bar{\boldsymbol{A}}^T\bar{\boldsymbol{M}}\dot{\bar{\boldsymbol{A}}}\dot{\bar{\xi}}_\omega \tag{12}$$

which resembles the λ-free differential equation (6) derived in the previous section, except that an additional variable transformation is conducted herein. Unlike the matrix product $\boldsymbol{A}^T\boldsymbol{M}\boldsymbol{A}$ in Eq. (6), $\bar{\boldsymbol{A}}^T\bar{\boldsymbol{M}}\bar{\boldsymbol{A}}$ appearing in Eq. (12) can be expressed in a closed form as

$$\bar{\boldsymbol{A}}^T\bar{\boldsymbol{M}}\bar{\boldsymbol{A}} = \begin{bmatrix} \bar{m}_{11} & \bar{m}_{12} & 0 & 0 & \ldots & 0 & 0 \\ \bar{m}_{12}^T & \bar{m}_{22} & \bar{m}_{23} & 0 & \ldots & 0 & 0 \\ 0 & \bar{m}_{23}^T & \bar{m}_{33} & \bar{m}_{34} & \ldots & 0 & 0 \\ 0 & 0 & \bar{m}_{34}^T & \bar{m}_{44} & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & \bar{m}_{n-1,n-1} & \bar{m}_{n-1,n} \\ 0 & 0 & 0 & 0 & \ldots & \bar{m}_{n-1,n}^T & \bar{m}_{nn} \end{bmatrix} \tag{13}$$

where the submatrices $\bar{m}_{ij}$ can be described with the aid of indicial notation given below:

On-Diagonal:

$$\bar{m}_{ii} = \bar{\boldsymbol{j}}_i + \sum_{k=i}^{i+1}[\tilde{\ell}_{1k}\boldsymbol{R}_1]\bar{\boldsymbol{m}}_k[\tilde{\ell}_{1k}\boldsymbol{R}_1]^T, \quad \text{for } i = 1, \ldots, n-1$$

$$\bar{\boldsymbol{j}}_n + [\tilde{\ell}_{nn}\boldsymbol{R}_n]\bar{\boldsymbol{m}}_n[\tilde{\ell}_{nn}\boldsymbol{R}_n]^T, \quad \text{for } i = n$$

Off-Diagonal:

$$\bar{m}_{ij} = -[\tilde{\ell}_{ij}\boldsymbol{R}_i]\bar{\boldsymbol{m}}_j[\tilde{\ell}_{jj}\boldsymbol{R}_j]^T, \quad \text{for } \begin{matrix} i = 1, \ldots, n-1 \text{ and} \\ j = i+1 \end{matrix} \tag{14}$$

Due to the simple format of the band matrix, an inverse successive displacement scheme[5,6] can be applied to determine $\ddot{\bar{\xi}}_\omega$ in integrating Eq. (12) for simulation, which only demands a successive inversion of the 3×3 submatrices on the diagonal. As a result, only $6n$ matrix operations are needed for the determination of the cofactors in this successive inverse process, while $\frac{3}{2}n(3n+1)$ operations are deemed necessary for the inversion of a $3n \times 3n$ matrix given by Eq. (6). As such, significant time saving is achievable for the simulation of the MBD system in conjunction with the use of the band mass matrix resulting from an alternative variable transformation. Once the updated solution of $\ddot{\bar{\xi}}_\omega$ is obtained from Eq. (12), the original acceleration variables can then be determined through the use of Eqs. (7) and (11).

IV. Conclusions

An alternative variable transformation has been proposed and analyzed for the simulation of multibody dynamic systems. The alternative variable transformation can be used by incorporating a modified null-space method to transform the mass matrix of multibody dynamic equations into a closed-form band fashion, which results in an efficient computation for the acceleration variables during integration. An alternative variable transformation matrix has been developed to work directly with the multibody equations of motion without altering the inherent dynamic characteristics, and it eliminates the need for expensive computation of inversion of a large mass matrix which is required for the simulation of multibody dynamic systems. An articulated multibody model has been selected for an analytical derivation, in which it has shown that the computation can be saved $(3n + 1/4)$-fold due to the computational merits associated with the band matrix.

Acknowledgment

The author wishes to acknowledge the facility support of this investigation through National Chung-Cheng University.

References

[1]Park, K. C., Chiou, J.-C., and Downer, J. D., "Explicit-Implicit Stagered Procedure for Multibody Dynamics Analysis," *Journal of Guidance, Control, and Dynamics*, Vol. 13, No. 3, 1990, pp. 562–570.

[2]Chiou, J.-C., Park, K. C., and Farhat, C., "A Natural Partitioning Scheme for Parallel Simulation of Multibody Systems," *Proceedings of the AIAA 32nd Structures, Structural Dynamics, and Material Conference* (Baltimore, MD), April 1991 (AIAA Paper 91-1111).

[3]Ghaemmaghami, P., and Juang, J.-N., "A Controller Design for Three Flexible-Body Large Angle Maneuvers," *Analysis and Control of Nonlinear Systems*, edited by C. I. Byrnes, North-Holland, NY, 1988, pp. 431–436.

[4]Huang, J.-K., Yang, L.-F., and Juang, J.-N., "Large Planar Maneuvers for Articulated Flexible Manipulators," *Proceedings of the 1988 American Control Conference* (Minneapolis, MN), Aug. 1988 (ACC Paper 88-4119).

[5]Gerald, C. F., and Wheatley, P. O., *Applied Numerical Analysis*, Addison–Wesley, Reading, MA, 1989.

[6]Nobel, B., and Daniel, J. W., *Applied Linear Algebra*, Prentice–Hall, Englewood Cliffs, NJ, 1977, pp. 67–69.

Trajectory Optimization Using Parallel Shooting Method on Parallel Computer

D. J. Wirthman,* S.-Y. Park,† and S. R. Vadali‡
Texas A&M University, College Station, Texas 77843

Introduction

TRAJECTORY optimization problems arise frequently in the design of modern guidance and control systems. A variety of techniques are currently available for the solution of trajectory optimization problems.[1–3] They include shooting methods, gradient-based methods, and methods using nonlinear programming. This Note considers the family of shooting methods that are used when a high-accuracy solution is required. In shooting methods, the unknown conditions are estimated, and the differential equations are integrated. The initial estimates are then iteratively corrected using Newton's method.

Received Nov. 5, 1992; revision received July 6, 1994; accepted for publication July 7, 1994.

*Graduate Student, Department of Aerospace Engineering; currently Research and Development Engineer, Space Systems/Loral, Mail Stop G32, 3825 Fabian Way, Palo Alto, CA 94303. Member AIAA.

†Graduate Student, Department of Aerospace Engineering. Student Member AIAA.

‡Associate Professor, Department of Aerospace Engineering. Associate Fellow AIAA.

Difficulties can arise when using the shooting method. If the differential equations are unstable or very stiff, they will be very sensitive to the initial conditions. Consequently, very good initial estimates are required to start the method. Otherwise the forward integration can overflow, or the Jacobian matrix used in computing the corrections can be very ill-conditioned. Unfortunately, very good initial estimates are often difficult to obtain without prior knowledge of the solution. This difficulty can be alleviated by using the multiple shooting method. In this method, the original interval is divided up into m (not necessarily equal) subintervals. This decreases the sensitivity of the problem to the initial conditions as the integration is performed over shorter intervals. The concept of the multiple shooting method is readily adapted to parallel processing.

Reference 4 discusses the implementation of the parallel shooting method on a parallel computer with a hypercube architecture. The paper develops formulas for timing and efficiency estimates, but no actual examples are presented. The implementation of nonlinear programming techniques on a parallel computer is discussed in Refs. 5 and 6. Speedup factors in excess of 10 have been reported on specific problems. Reference 7 introduces a quasilinearization algorithm that is well suited for parallel implementation. A parallel scheme using the method of particular solutions is presented in Ref. 8.

Parallel Shooting

The parallel shooting method is simply a parallel implementation of the multiple shooting method. The parallel shooting method can be implemented as follows:

1) Commit one processor to a group of subintervals. Estimate all free parameters.

2) Each processor integrates independently across its subintervals. All processors including processor 0 pass on the computed state/costate values to a driver program handled by processor 0.

3) Each processor then independently integrates the differential equations across its intervals and computes the corresponding blocks of the Jacobian matrix and sends it to the driver program.

4) The driver program receives the blocks of the Jacobian matrix and forms them into one total Jacobian matrix. It then solves the linear set of equations for the corrections to the free parameters. A step size control algorithm is used to guarantee that the error in constraint satisfaction decreases from one iteration to the next.

5) The driver program passes sections of the corrections to the corresponding processors.

6) Return to step 2 and iterate until convergence.

Implementation Specifics

The parallel shooting algorithm was implemented on an nCUBE 6400 series parallel computer. It is a 64-processor, multiple-instruction, multiple-data-stream (MIMD) machine. The front end (host), which is essentially the user interface and file storage system, is a Sun Sparc workstation. Of the 64 processors, which are often referred to as nodes, 32 have 1 MB of memory each and the rest 4 MB each. There is no shared memory. Each processor has an average speed of 4 MIPS, with peak rate of 7.5 MIPS. Floating-point operations are performed at 2.5 MFLOPS.

The nCUBE utilizes a hypercube architecture. With the hypercube architecture, a process can be allocated a number of nodes equal to 2^k. Each node is linked directly with k other nodes. Thus, if $k = 3, 8$ nodes are allocated. Each of the nodes is linked directly to three other nodes. This can be pictured graphically as a cube.

The parallel shooting code uses the single-program, multiple-data model. In this model, the same program runs on all nodes. This is convenient for the parallel shooting method because each node performs essentially the same operations. The additional procedure for node 0 is handled by using a driver program that manages the nCUBE system. The driver program then simply calls the parallel shooting code based upon the node on which it is running. All node-specific tasks, mainly communication and integration, are handled by using statements based on the node number.

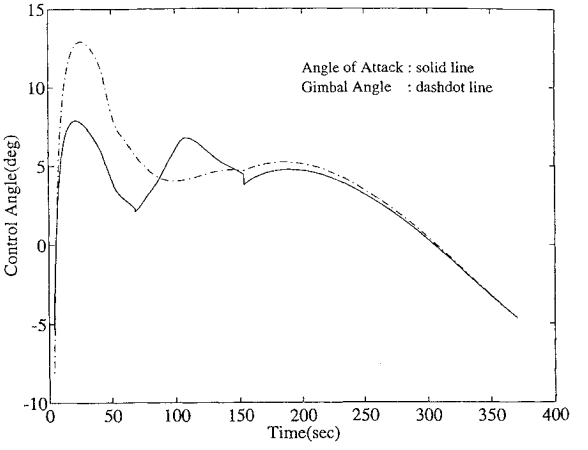

Fig. 1 Control variable history.

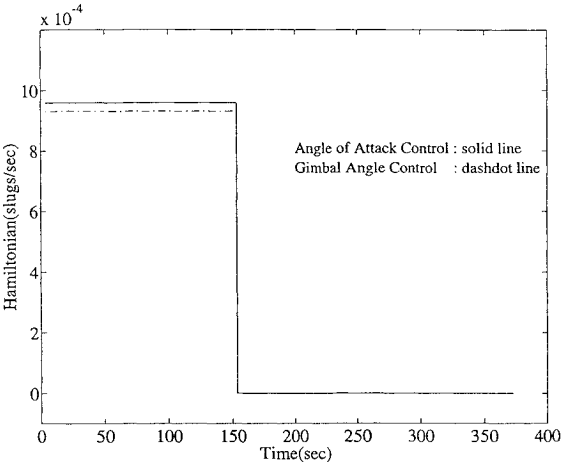

Fig. 2 Hamiltonian history.

Numerical Example

The parallel shooting method was used to solve for the rigorous optimal solution of a guidance problem for the optimal ascent of the Advanced Launch System.[9,10] The vehicle consists of a core, which contains three engines, and a booster, which contains seven engines. The engines burn continuously throughout the trajectory. The booster is separated when it runs out of fuel, which occurs at 153.54 s[9]. The final time is free. A spherical nonrotating Earth model is used. The differential equations for the two-dimensional version of the problem are given in Ref. 10. The lift and drag coefficients used are also tabulated in Ref. 10 as functions of Mach number and angle of attack. These data were interpolated using a combination of polynomials and cubic splines. After separation, the vehicle flies at hypersonic velocities, and the Mach number independence principle is assumed to hold. As a result, the coefficients are functions of the angle of attack only. The performance objectives are to maximize the final mass and place the payload at the perigee of an 80×150-nm transfer orbit. The Bulirsh-Stoer routine is used for integrating the differential equations. The integration time dominates considerably over the time required for linear algebra.

The first example presented uses the angle of attack as the control variable. A state variable inequality constraint is placed on the dynamic pressure by limiting it to 650 lb/ft^2. This is a first-order state inequality constraint giving rise to jumps in the altitude and velocity costates when the constraint becomes active. The entire trajectory is divided into 16 subintervals with 12 being in the first stage and 4 in the second stage. Two subintervals are dedicated to the segment of the trajectory where the inequality constraint is active. The initial guesses for the costates were obtained by solving the problem without the inequality constraint. The second example uses the thrust gimbal angle as the control variable instead of the angle of attack. The dynamic pressure constraint is not active for this example. The dependence of the aerodynamic coefficients on the angle of attack

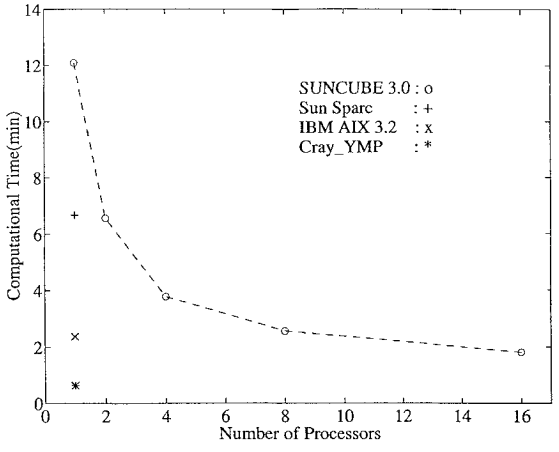

Fig. 3 Computational time vs number of processors for α control.

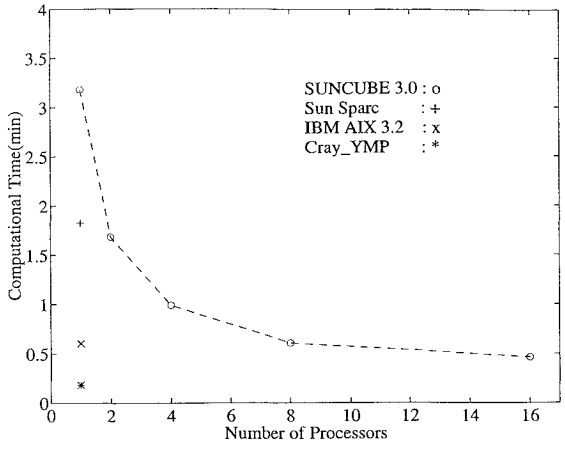

Fig. 4 Computational time vs number of processors for δ control.

are neglected to simplify the problem. The trajectory is divided into 10 subintervals during the first stage and 6 during the second stage. The initial guesses for the costates were the same as used for the previous example.

Figure 1 shows the control variable histories for the two examples. The inequality constraint is active for a short time, during the first stage, when the angle of attack is used as the control variable, as seen from Fig. 1. Figure 2 shows the Hamiltonian histories. The Hamiltonians are zero in the second stage as the final time is free. Figures 3 and 4 show the total computational and communication times (four iterations each) and speedup performance as a function of the number of processors for the two examples. Also included in these figures are the central processing unit (CPU) times for the same code to run on different serial computers. It is observed that the computational time for the second example is much lower than the first due to the elimination of the angle-of-attack dependency of the aerodynamic coefficients.

Conclusions

Implementation of a parallel shooting method on a parallel computer for solving a variety of optimal control and guidance problems has been presented. It is observed that a speedup of nearly 7 to 1 is achieved using 16 processors for the examples considered. Further improvements in performance might be achievable by parallelizing in the state domain.

Acknowledgment

This research is based in part upon work supported by the Texas Higher Education Coordinating Board under grant 999903-264.

References

[1]Bryson, A. E., and Ho, Y. C., *Applied Optimal Control*, Hemisphere, Washington, DC, 1975.

[2]Roberts, S. M., and Shipman, J. S., *Two-Point Boundary Value Problems: Shooting Methods*, American Elsevier, New York, 1972.

[3]Enright, P. J., and Conway, B. A.," Optimal Finite-Thrust Spacecraft Trajectories Using Collocation and Nonlinear Programming," *Journal of Guidance, Dynamics, and Control*, Vol. 14, No. 5, 1991, pp. 981–985.

[4]Keller, H. B., and Nelson, P., "Hypercube Implementations of Parallel Shooting," *Applied Mathematics and Computation*, Vol. 31, 1989, pp. 574–603.

[5]Betts, J. T., and Huffman, W. P., "Trajectory Optimization on a Parallel Processor," *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 2, 1991, pp. 431–439.

[6]Psiaki, M. L., and Park, K.-H., "Parallel Solver for Trajectory Optimization Search Directions," *Journal of Optimization Theory and Applications*, Vol. 73, No. 3, 1992, pp. 519–546.

[7]Menon, P. K. A., and Lehman, L. L., "A Parallel Quasi-Linearization Algorithm for Air Vehicle Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 9, No. 1, 1986, pp. 119–121.

[8]Miele, A., and Wang, T., "Parallel Computations of Two-Point Boundary-Value Problems via Particular Solutions," *Journal of Optimization Theory and Applications*, Vol. 79, No. 1, 1993, pp. 5–29.

[9]Shaver, D. A., and Hull, D. G., "Advanced Launch System Trajectory Optimization Using Suboptimal Control," AIAA Paper 90-3413-CP, Aug. 1990.

[10]Seywald, H., and Cliff, E. M., "A Feedback Control for the Advanced Launch System," AIAA Paper 91-2619-CP, Aug. 1991.

Maximum Likelihood Estimation of Fractional Brownian Motion and Markov Noise Parameters

Michael E. Ash*
Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts 02173
and
Matthew E. Skeen†
Cape Canaveral Air Force Station, Florida 32925

I. Introduction and Conclusions

THE parameters in a trend, random walk, exponentially correlated, and white noise Markov approximation to flicker noise [−1 log-log power spectral density (PSD) slope[1–3]] are estimated using maximum likelihood system identification. In addition, parameters for a fractional Brownian motion (fBm)[4,5] model of flicker noise plus trend or white noise are estimated using maximum likelihood gradient iterations, in contrast to previous work using brute force search to estimate a single fBm parameter.[6]

II. Maximum Likelihood Estimation

Let $z^N = [z_1, \ldots, z_N]^T$ be measurements at times $t_1, \ldots, t_N$ with joint probability density $p(z; \alpha)$ dependent on parameters $\alpha = [\alpha_1, \ldots, \alpha_m]^T$. Let $\Delta\alpha_j = \hat{\alpha}_j - \alpha_{j0}$ be adjustments from guesses α_{j0} towards the maximum likelihood estimates $\hat{\alpha}_j$ that minimize the negative log likelihood $\zeta(z^N; \alpha) = -\ln[p(z^N; \alpha)]$. A Taylor series expansion of the gradient of $\zeta(z^N; \alpha)$ yields

$$\sum_{j=1}^{m} A_{ij}\Delta\alpha_j = B_i, \qquad i = 1, \ldots, m \tag{1}$$

$$B_i = -\left.\frac{\partial\zeta(z^N;\alpha)}{\partial\alpha_i}\right|_{\alpha=\alpha_0}, \qquad A_{ij} = \left.\frac{\partial^2\zeta(z^N;\alpha)}{\partial\alpha_i\partial\alpha_j}\right|_{\alpha=\alpha_0} \tag{2}$$

Received June 15, 1993; revision received March 31, 1994; accepted for publication July 6, 1994.

*Principal Member of Technical Staff. Senior Member AIAA.
†Captain, U.S. Air Force.