JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 79, No. 1, OCTOBER 1993

Epsilon–Ritz Method for Solving Optimal Control Problems: Useful Parallel Solution Method

P. A. FRICK[1] AND D. J. STECH[2]

Communicated by T. L. Vincent

Abstract. Using Balakrishnan's epsilon problem formulation (Ref. 1) and the Rayleigh–Ritz method with an orthogonal polynomial function basis, optimal control problems are transformed from the standard two-point boundary-value problem to a nonlinear programming problem. The resulting matrix-vector equations describing the optimal solution have standard parallel solution methods for implementation on parallel processor arrays. The method is modified to handle inequality constraints, and some results are presented under which specialized nonlinear functions, such as sines and cosines, can be handled directly. Some computational results performed on an Intel Sugarcube are presented to illustrate that considerable computational savings can be realized by using the proposed solution method.

Key Words. Epsilon method, parallel computing, Walsh functions, hypercubes, Legendre polynomials, Chebyshev polynomials, Ritz method.

1. Introduction

Optimal control theory for nonlinear systems is well developed. Conventional computational solution methods for optimal control problems are, however, highly serial in nature. Usually algorithms involve an iterative process of solving the system equations forward in time, the adjoint equations backward in time, and then making some sensible adjustment to

[1]Professor and Dean, College of Engineering and Applied Science, University of Colorado at Colorado Springs, Colorado.

[2]Research Scientist, Frank J. Seiler Laboratory, United States Air Force Academy, Colorado Springs, Colorado.

0022-3239/93/1000-0031$07.00/0

the control functions before repeating the process. As a result, real-time implementation of nonlinear optimal control theory necessitates extremely fast solution times for these computationally intensive, two-point boundary-value problems, and the theory is rarely implemented in actual control hardware. New, computationally powerful parallel processors offer extremely high computation rates, and it is now possible to solve nonlinear optimal control problems quickly by adapting the problems for solution on parallel processors.

The authors have previously outlined and demonstrated a method in which many processors can be used concurrently to solve nonlinear optimal control problems (Ref. 2). This is accomplished by casting the dynamic optimization problem in its integral epsilon form (Ref. 1) and then using the Raleigh–Ritz method with the Walsh functions as a basis to convert the problem into a nonlinear programming problem.

This parametrization of the optimal control problem is unique and is quite different from recent efforts reported in Refs. 3, 4, 5 in that no differential equation (system or adjoint) is solved explicitly because the epsilon method is employed. It is this feature that allows a parallel implementation of the proposed method; i.e., the traditional sequential nature of generating solutions to differential equations either forward or backward is completely eliminated.

1.1. Epsilon Method. For readers not intimately familiar with the ε method, some of the convergence results are summarized below. We follow the derivation of Frick (Ref. 6) rather than that by Balakrishnan (Refs. 1, 7). Consider optimal control problems of the form

$$\min_u \int_0^T G(x, u; t)\, dt, \tag{1}$$

subject to the dynamic constraints

$$\dot{x}(t) = f(x, u; t), \tag{2}$$

where

$$x(0) = x_s.$$

For the problem (1)–(2), the composite cost functional

$$J(\varepsilon, x, u) = \int_0^T (1/2\varepsilon)\, \|e(t; \varepsilon)\|^2 + G(x, u; t)\, dt \tag{3}$$

is constructed, where the error function in the system dynamics is given by

$$e(t;\varepsilon) = x(t) - x_s - \int_0^t f(x, u; \tau)\, d\tau.$$

The composite cost functional (3) is now minimized simultaneously with respect to both x and u for a given $\varepsilon > 0$. If necessary, the process is subsequently repeated for a smaller value of ε or even a whole sequence $\{\varepsilon_j\}$ which decreases monotonically. This penalty type process is repeated until the error in the system dynamics is sufficiently small.

Convergence, subject to the usual boundedness, continuity, and convexity assumptions, is assured by the following result (Ref. 6).

1.2. Convergence Result. Consider the sequence of scalars $\{\varepsilon_j\} \downarrow 0$ monotonically decreasing. For the corresponding sequence of minimizing solutions to the integral ε problem [that is, $J(\varepsilon; x, u)$ of (3)] denoted by $\{x_0(\varepsilon_j), u_0(\varepsilon_j)\}$ and the associated sequence of error functions $\{e_0(\varepsilon_j)\}$,[3] we have the sequences

$$\{u_0(\varepsilon_j)\} \to u^*,$$

$$\{x_0(\varepsilon_j)\} \to x^*,$$

$$\{e_0(\varepsilon_j)/\varepsilon_j\} \to \lambda^*,$$

$$\{e_0(\varepsilon_j)\} \downarrow 0,$$

$$\{J(\varepsilon_j, x_0(\varepsilon_j), u_0(\varepsilon_j))\} \uparrow V(x^*, u^*),$$

as $\varepsilon \downarrow 0$. Here, x^*, u^*, and $V(x^*, u^*)$ are the optimal control, state, and cost for the optimal control problem (1)–(2).

Viewing the minimization of the composite functional $J(\varepsilon, x, u)$ of (3) as an optimization problem in $\mathscr{L}^2(0, T)$, either gradient methods (Ref. 6) or the Raleigh–Ritz method (Ref. 7) were used in solving problems of this type. For problems that require several values from the sequence $\{\varepsilon_j\}$ to reach a satisfactory problem solution, greater numerical stability is assured by adding an additional term to (3) as first proposed by Hestenes (Ref. 8).

The Raleigh–Ritz method (Ref. 9) with a trigonometric base was employed by Balakrishnan in solving the original differential epsilon problem (Refs. 1, 7). The use of the trigonometric functions posed two major practical problems. First, since either the differentiation or integration of the state variables is required, the full Fourier series (both sine and

[3]Note that the minimizing state, control, and error functions for a given ε depend on the size of $\varepsilon > 0$.

cosine functions) had to be used, which makes the incorporation of the boundary conditions very difficult indeed. The second problem is that, in truncating the approximating series to N terms say, the product of any two functions (in nonlinear systems for example) will now require $2N$ terms in the series. Difficulties in the incorporation of the initial conditions was first recognized by Jones and McCormick (Ref. 10) and motivated the development of the integral epsilon formulation (Ref. 6), the Sobolev space formulation by di Pillo et al. (Refs. 11, 12), and the use of gradient computational methods in the corresponding function space, to solve the epsilon problems.

The authors have demonstrated in Ref. 2 that using the Walsh functions as the set of basis functions in the Ritz method eliminates the above-mentioned problems associated with the use of the trigonometric functions. Two key properties of Walsh functions, and in particular the ease by which finite integrals for a Walsh series can be represented by a Walsh expansion and the finite group property of the Walsh functions (Ref. 13), are summarized in Appendix A (Section 7). In the process of applying the Walsh-based Ritz method, the optimal control problem is reduced to a nonlinear programming problem. By adopting a fairly simple vector-matrix notation (Ref. 14), the resulting nonlinear programming problem can be solved on any one of a number of types of parallel computers (Ref. 2), such as systolic processor arrays or conventional MIMD machines.

In this paper, it is shown that the proposed method can be implemented using orthogonal polynomial functions such as the Legendre and Chebyshev polynomials instead of the Walsh functions. An extension of the previous results is also provided by the formulation of a mechanism for dealing with nonpolynomial nonlinearities and inequality constraints when the Walsh function basis is used.

The proposed parallel solution method's utility for traditionally troublesome nonlinear optimal control problems such as the minimum time problem and nonlinear optimal control problems with inequality constraints is illustrated in a number of computational examples, using an Intel Sugarcube with eight processors operating in parallel.

2. Rayleigh–Ritz Solution of the Integral Epsilon Problem Using Legendre Polynomials

In Ref. 2, we outlined the procedure for converting an optimal control problem into a nonlinear programming problem. This can be accomplished by formulating the optimal control problem into its integral epsilon form (Ref. 6) and then using Walsh functions as a basis to convert the

unconstrained functional into a static function that can be minimized with respect to the Walsh function coefficients. Because Walsh functions were used to approximate the time-varying functions (state and control functions), the resulting optimal solutions have a distinct stair-step character.

We now show that the same development is possible with orthogonal polynomials such as the Legendre or Chebyshev polynomials.

2.1. Linear Case. To illustrate the procedure, we use a generic set of orthogonal polynomials as the set of basis functions for the development of the computational method. In Section 5, we illustrate the procedure by presenting computational examples using both the Walsh functions and the Legendre polynomials.

Consider, therefore, the following linear time-varying optimal control problem with a quadratic cost function:

$$\dot{x}(t) = A(t)\, x(t) + B(t)\, u(t) + C(t), \tag{4}$$

with initial conditions

$$x(0) = x_s$$

and cost functional given by

$$V(x, u) = (1/2) \int_0^{t_f} \{[x(t), Qx(t)] + [u(t), Ru(t)]\}\, dt, \tag{5}$$

where $x(t) \in R^n$, $u(t) \in R^m$, and $u(t)$ represents the inputs or controls.

This linear, time-varying optimal control problem can be solved using the integral epsilon method (Ref. 6). Consider the integral form of the system equation (4),

$$x(t) = x_s + \int_0^t A(\tau)\, x(\tau)\, d\tau + \int_0^t B(\tau)\, u(\tau)\, d\tau + \int_0^t C(\tau)\, d\tau.$$

If the system equations are viewed as a constraint, they can be adjoined to the cost functional by forming the error function

$$e(t) = x(t) - x_s - \int_0^t A(\tau)\, x(\tau)\, d\tau - \int_0^t B(\tau)\, u(\tau)\, d\tau - \int_0^t C(\tau)\, d\tau$$

and incorporating it into the cost functional (5) to form

$$J(x, u, \varepsilon) = (1/2\varepsilon) \int_0^{t_f} \|e(t, \varepsilon)\|^2\, dt + V(x, u).$$

In a similar manner, final state conditions (equality constraints at the final time) can be incorporated in the form of an additional penalty term with

$$\rho = x_s + \int_0^{t_f} A(t)\, x(t)\, dt + \int_0^{t_f} B(t)\, u(t)\, dt + \int_0^{t_f} C(t)\, dt - x_f,$$

where x_f is the desired final state, yielding the composite cost functional

$$J(x, u, \varepsilon) = (1/2\varepsilon) \int_0^{t_f} \|e(t)\|^2\, dt + \|\rho\|^2 + V(x, u). \tag{6}$$

The composite cost functional (6) is now minimized simultaneously with respect to both $x(t)$ and $u(t)$ for a given $\varepsilon > 0$. If necessary, the process is subsequently repeated for a smaller value of ε or even a whole sequence $\{\varepsilon_j\}$ which decreases monotonically. Convergence of the epsilon method is discussed in some detail in Refs. 1 and 6 and is not repeated here.

If the time-varying functions in (6) are approximated by a series of basis functions, then the dynamic optimization problem (6) is converted into a static optimization problem. We have, for example,

$$x(t) = XS(t),$$

or

$$e(t; \varepsilon) = ES(t),$$

where X and E are matrices of coefficients and

$$S(t) = [s_0(t), s_1(t), \ldots, s_{N-1}(t)]^T,$$

is a vector of orthogonal polynomials. These approximations are substituted into (6), giving an approximated composite cost function

$$J = \int_0^{t_f} \{(1/2\varepsilon)\, S^T(t)\, E^T E S(t) + (1/2)\, S^T(t)\, X^T Q X S(t) + (1/2)\, S^T(t)\, U^T R U S(t)\}\, dt + (1/2\varepsilon)\, S^T(t)\, \xi^T \xi S(t). \tag{7}$$

As is shown in the Appendix (Section 7), Eq. (7) can be simplified to yield

$$J = \operatorname{trace}\{(1/2\varepsilon)\, S^T(t)\, E^T(t)\, E^T E S(t) + (1/2\varepsilon)\, \xi^T \xi + (1/2)\, X^T Q X + (1/2)\, U^T R U\}. \tag{8}$$

By using the matrix-vector notation[4] of Brewer (Ref. 14) and the following result[5]:

$$\text{vec}(QX) = (I_N \otimes Q)\,\text{vec}(X),$$

Eq. (8) can be written as

$$J = (1/2\varepsilon)\,\text{vec}(E)^T (O \otimes I_n)\,\text{vec}(E) + (1/2\varepsilon)\,\text{vec}(\xi)^T (O \otimes I_n)\,\text{vec}(\xi)$$
$$+ (1/2)\,\text{vec}(X)^T (O \otimes Q)\,\text{vec}(X) + (1/2)\,\text{vec}(U)^T (O \otimes R)\,\text{vec}(U), \tag{9}$$

where

$$\text{vec}(E) = \{\text{vec}(X) - \text{vec}(G_s) - [(H^T \otimes I_n)\,\Lambda_\alpha]\,\text{vec}(X)$$
$$- [(H^T \otimes I_n)\,\Lambda_\beta]\,\text{vec}(U) - (H^T \otimes I_n)\,\text{vec}(C)\}, \tag{10}$$

and

$$\text{vec}(\xi) = \text{vec}(G_s) + \Gamma_\alpha \text{vec}(X) + \Gamma_\beta\,\text{vec}(U) + \Gamma_c\,\text{vec}(C) - \text{vec}(G_f). \tag{11}$$

In (9), O is a matrix which results from the orthogonal property of the polynomials. In (10) and (11), Λ and Γ are matrices of polynomial coefficients and H is the integration matrix for orthogonal polynomials (see Appendix A). To minimize J, the gradient is calculated simultaneously with respect to both $\text{vec}(X)$ and $\text{vec}(U)$ and set equal to zero:

$$\nabla_{\text{vec}(X)} J = (1/\varepsilon)(I_{Nn} - (H^T \otimes I_n)\,\Lambda_\alpha)^T (O^T \otimes I_n)\{(I_{Nn} - (H^T \otimes I_n)\,\Lambda_\alpha)\,\text{vec}(X)$$
$$- ([H^T \otimes I_n]\,\Lambda_\beta)\,\text{vec}(U) - \text{vec}(G_s)\} + (1/\varepsilon)\,\Gamma_\alpha^T (O^T \otimes I_n)\{\text{vec}(G_s)$$
$$+ \Gamma_\alpha\,\text{vec}(X) + \Gamma_\beta\,\text{vec}(U) + \Gamma_c\,\text{vec}(C) - \text{vec}(G_f)\}$$
$$+ (O^T \otimes Q)\,\text{vec}(X) = 0,$$
$$\nabla_{\text{vec}(U)} J = -(1/\varepsilon)[H^T \otimes I_n]\,\Lambda_\beta)^T (O^T \otimes I_n)\{(I_{Nn} - (H^T \otimes I_n)\,\Lambda_\alpha)\,\text{vec}(X)$$
$$- ([H^T \otimes I_n]\,\Lambda_\beta)\,\text{vec}(U) - \text{vec}(G_s)\} + (1/\varepsilon)\,\Gamma_\beta^T (O^T \otimes I_n)\{\text{vec}(G_s)$$
$$+ \Gamma_\alpha\,\text{vec}(X) + \Gamma_\beta\,\text{vec}(U) + \Gamma_c \text{vec}(C) - \text{vec}(G_f)\}$$
$$+ (O^T \otimes R)\,\text{vec}(U) = 0.$$

The resulting system of equations can be written in very simple form,

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} \text{vec}^*(X) \\ \text{vec}^*(U) \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \end{bmatrix}, \tag{12}$$

[4]Here, $\text{vec}(X)$ denotes a nN-vector formed by stacking the N columns of X above one another.
[5]$\otimes$ denotes the Kronecker matrix product described in Appendix 7.1.

where

$$K_{11} = (1/\varepsilon)(I_{Nn} - (H^T \otimes I_n)\,\Lambda_\alpha)^T (O \otimes I_n)(I_{Nn} - (H^T \otimes I_n)\,\Lambda_\alpha) + (O \otimes Q) + (1/\varepsilon)\,\Gamma_\alpha^T (O \otimes I_n)\,\Gamma_\alpha,$$

$$K_{12} = -(1/\varepsilon)(I_{Nn} - (H^T \otimes I_n)\,\Lambda_\alpha)^T (O \otimes I_n)([H^T \otimes I_n]\,\Lambda_\beta) + (1/\varepsilon)\,\Gamma_\alpha^T (O \otimes I_n)\,\Gamma_\beta,$$

$$K_{21} = -(1/\varepsilon)([H^T \otimes I_n]\,\Lambda_\beta)^T (O \otimes I_n)(I_{Nn} - (H^T \otimes I_n)\,\Lambda_\alpha) + (1/\varepsilon)\,\Gamma_\beta^T (O \otimes I_n)\,\Gamma_\alpha,$$

$$K_{22} = (1/\varepsilon)([H^T \otimes I_n]\,\Lambda_\beta)^T (O \otimes I_n)([H^T \otimes I_n]\,\Lambda_\beta) + (O \otimes R) + (1/\varepsilon)\,\Gamma_\beta^T (O \otimes I_n)\,\Gamma_\beta,$$

and

$$D_1 = (1/\varepsilon)(I_{Nn} - (H^T \otimes I_n)\,\Lambda_\alpha)^T (O \otimes I_n)(\mathrm{vec}(G_s) + [P^T \otimes I_n]\,\mathrm{vec}(C)) + (1/\varepsilon)\,\Gamma_\alpha^T (O \otimes I_n)\{G_f - G_s - \Gamma_c\,\mathrm{vec}(C)\}$$

$$D_2 = -(1/\varepsilon)([H^T \otimes I_n]\,\Lambda_\beta)^T (O \otimes I_n)(\mathrm{vec}(G_s) + [H^T \otimes I_n]\,\mathrm{vec}(C)), + (1/\varepsilon)\,\Gamma_\beta^T (O \otimes I_n)\{G_f - G_s - \Gamma_c\,\mathrm{vec}(C)\}.$$

For parallel computational implementation, the above K-matrix can be rewritten in a more convenient form by defining

$$M = \begin{bmatrix} I_{Nn} - (H^T \otimes I_n)\,\Lambda_\alpha & -(H^T \otimes I_n)\,\Lambda_\beta \\ 0 & 0 \end{bmatrix}, \tag{13}$$

$$L = \begin{bmatrix} O \otimes Q & 0 \\ 0 & O \otimes R \end{bmatrix} G = \begin{bmatrix} O \otimes I_n & 0 \\ 0 & 0 \end{bmatrix}, \tag{14}$$

$$F = \begin{bmatrix} \Gamma_\alpha & \Gamma_\beta \\ 0 & 0 \end{bmatrix}, \tag{15}$$

which then gives

$$K = (1/\varepsilon)\,M^T G M + L + (1/\varepsilon)\,F^T G F. \tag{16}$$

In (13)–(15), the integration matrix H and orthogonal property matrix O are dependent on the basis functions used. For example, if Walsh functions are used, $O = I$ and thus G becomes an identity. For Legendre polynomials, O is diagonal but it is not the identity matrix.

The solution to the linear time-varying optimal control problem is now obtained by solving (12) for vec*(X) and vec*(U) employing the

construction (16) using simple matrix operations. Since these matrix operations (mostly matrix inversion) have a variety of parallel implementations, the proposed algorithm has a number of possible parallel implementations (Refs. 15, 16). A parallel implementation for an eight processor Intel Sugarcube is given in Ref. 2 and is used in the numerical results presented in Section 5.

2.2. Remark. Note that there are no basic restrictions on the size of $\varepsilon > 0$ that can be used in solving Eq. (9). That is, any selection that will not render the K matrix singular can be used as a choice for ε. This in fact eliminates the penalty nature from the optimization process in the epsilon method (Ref. 17).

2.3. Nonlinear Case. Nonlinear optimal control problem can be solved by quasilinearizing the nonlinear problem and then applying the proposed algorithm to the resulting linear time-varying problem (see Ref. 2), and the quasilinearization is repeated if necessary. The procedure is summarized below for convenience.

For the nonlinear case, we consider the system

$$\dot{x}(t) = f(x, u; t), \tag{17a}$$

$$x(0) = x_s, \tag{17b}$$

where $x(t) \in R^n$, $u(t) \in R^m$, and with the cost functional given by

$$V(x, u) = (1/2) \int_0^T \{[x(t), Qx(t)] + [u(t), Ru(t)]\}\, dt.$$

Using a standard quasilinearization procedure (Ref. 18), the system (17) is approximated by the sequences of functions $\{x^k\}$, $\{u^k\}$, $k = 1, 2, \ldots$, and the linearized equations

$$\dot{x}^{k+1}(t) = A^{k+1}x^{k+1}(t) + B^{k+1}u^{k+1}(t) + C^k(t),$$

where

$$A^k(t) = \nabla_x f(x^k, u^k; t),$$

$$B^k(t) = \nabla_u f(x^k, u^k; t),$$

$$C^k(t) = f(x^k, u^k; t) - \nabla_x f(x^k, u^k; t)\, x^k(t) - \nabla_u f(x^k, u^k; t)\, u^k(t).$$

The composite cost functional for this case can therefore be written as

$$J(\varepsilon, x^k, u^k) = \int_0^T (1/2\varepsilon)\, \|e^k(t; \varepsilon)\|^2 + (1/2)\, x^k(t), Qx^k(t)]$$
$$+ (1/2)\, u^k(t), Ru^k(t)]\, dt, \tag{18}$$

where the error function $e^k(t;\varepsilon)$ is given by

$$e^k(t;\varepsilon) = x^k(t) - x_s - \int_0^t A^k x^k(\tau)\, d\tau - \int_0^t B^k u^k(\tau)\, d\tau - \int_0^t C^k(\tau)\, d\tau. \tag{19}$$

Note that we can now view the minimization problem (18)–(19) in exactly the same way as in Section 2.1.

Since a quasilinearization scheme is used to approximate the system equations (17), it is necessary to solve the matrix equation (16) iteratively for $k = 0, 1, 2, \ldots$. Again, there appears to be no restriction on the size of $\varepsilon > 0$ that can be chosen. This conjecture is tested against the results obtained from a number of computational examples in Section 5.

2.4. Remark. There is no compelling reason for using the quasilinearization procedure here. In fact, in his original computations Balakrishnan (Ref. 7) used a truncated Taylor expansion in approximating the nonlinear dynamics. In cases where the nonlinearities are polynomial in nature, no linearization should be required. Some speed-up could be realized in such cases, but the use of Brewer's vec notation (Ref. 14) becomes very cumbersome.[6]

3. Parallel Implementation

Solving the optimization problem of Section 2 [see Eq. (12)], and therefore the optimal control problem, simply boils down to solving the linear system of algebraic equations of the form

$$Ax = b.$$

Almost all noniterative methods for solving systems of equations of this type on parallel machines consist of the triangularization of the augmented matrix $[A \mid b]$ and then using backsubstitution to find the x-vector. The approach used here is that proposed by Bojanczyk (Ref. 19), modified for implementation on a linear array of processors.

Solutions to the matrix equation (12) using (16) involves not only the matrix inversion (or triangularization and backsubstitution) outlined above, but also a number of matrix multiplications and the formation of Kronecker products.

These operations, like many other numerical vector or matrix operations, have well-defined parallel implementations, both on parallel

[6]The authors are indebted to one of the reviewers for pointing to this shortcoming in the narrative.

computers and array processors such as systolic or wavefront arrays (Refs. 15, 16). This is a desirable feature since for parallel implementations it is usually very difficult to separate a particular parallel algorithm from the parallel architecture required to run it. Additionally, for such numerical operations it has been shown (Ref. 15) that, even with a time penalty for communicating between processors, as the size of the problem increases, the possible speedup from these parallel implementations is not bounded if the number of processors available is not bounded.

Algorithm Summary. The following summarizes the parallel implementation of the proposed algorithm on a linear array of processors where $p < n$.

Step 1. The matrix M of Eq. (13) is formed serially and is passed to the processor array.

Step 2. The array multiplies $M^T M$.

Step 3. L of Eq. (14) is sent to the appropriate processors and added to $M^T M$.

Step 4. D is added to the last processor.

Step 5. QR factorization is performed.

Step 6. Backsubstitution yields the Walsh coefficients for the optimal control u^* and state x^* functions (see Fig. 1).

Some numerical results are provided in the following sections.

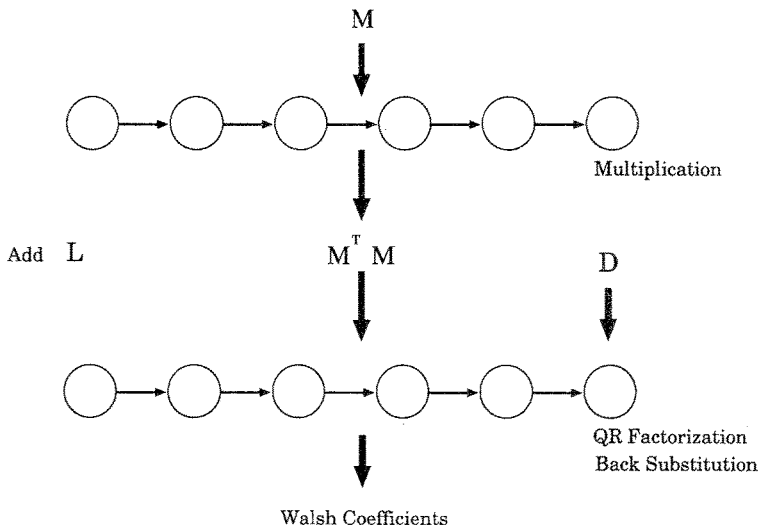

Fig. 1. Parallel implementation on a linear array.

4. Nonlinear Operations on Walsh Functions and Inequality Constraints

To handle other than polynomial nonlinearities in the dynamical equations, it is necessary to develop a method of performing nonlinear operations on the coefficients of the basis functions being used. For Walsh functions, this is fairly straightforward.

4.1. Nonlinear Oprations on Walsh Functions. A Walsh series made up of N Walsh functions can be described by a vector whose N elements give the value of the Walsh series for each of the N intervals.

For example, $\phi_2(t)$ can be described by

$$p_2 = [1, -1, 1, -1],$$

where each of the elements gives the value of $\phi_2(t)$ on each of its four different intervals. So the first 4 Walsh functions can be described by a vector with 4 elements, the first 8 Walsh functions by vectors with 8 elements, and so on. In this manner, the vector $\Psi(t)$, which is a vector of Walsh functions, can be described by a matrix. For example,

$$\Psi(t) = \begin{bmatrix} \phi_0(t) \\ \phi_1(t) \\ \vdots \\ \phi_N(t) \end{bmatrix} = \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_N \end{bmatrix} = \mathscr{P}.$$

Just as it is possible to approximate a function by

$$f(t) \approx F\Psi(t),$$

then

$$f(t) \approx F\mathscr{P}.$$

Since the Walsh functions are orthogonal, the $\mathscr{P}$-matrix is invertible and can be used to generate the Walsh series from the Walsh coefficients, and the Walsh coefficients from the Walsh series. In order to perform a nonlinear operation on a set of Walsh coefficients, it is only necessary to generate the Walsh series from the Walsh coefficients, perform the nonlinear operation elementwise on the Walsh series, and then convert back into Walsh coefficients. To demonstrate the technique, consider the following example.

4.2. Example. Let

$$g(t) = \sin(f(t)),$$

and we are to find the Walsh coefficients of $g(t)$ given the Walsh coefficients of $f(t)$. This is straightforward using the $\mathscr{P}$-matrix. From the above, we can write

$$\sin(F\mathscr{P}) = G\mathscr{P} \approx g(t),$$

and the coefficients of $g(t)$ can be found by

$$G = \sin(F\mathscr{P})\,\mathscr{P}^{-1}.$$

Other nonlinear functions can be performed on Walsh coefficients in a similar manner. Using this technique, the proposed method allows the solution of nonlinear optimal control problems with a variety of nonlinearities in the dynamical equations and provides a method for handling inequality constraints as illustrated in the next subsection.

4.3. Optimal Control Problems with Inequality Constraints. In most practical optimal control problems, some physical limits or constraints apply to the controls, and we adapt the proposed epsilon method to handle such inequality constraints. Consider, for example, the following problem:

$$\dot{x}(t) = f(x, u; t), \qquad x(t_0) = x_0, \tag{20}$$

with

$$V(x, u) = \int_0^{t_f} G(x, u; t)\, dt, \tag{21}$$

$$|u_i(t)| \leqslant c_i, \qquad i = 1, 2, \ldots, m, \tag{22}$$

where c is some constant m-vector. Using $\mathscr{P}$, the inequality constraint can be formulated in terms of the Walsh coefficients of the control. First, we approximate c and the control $u(t)$ by their Walsh series in vector form,

$$c = C_t = [c_t^0, c_t^1, \ldots, c_t^{N-1}],$$
$$u(t) = U_t = [u_t^0, u_t^1, \ldots, u_t^{N-1}].$$

Now, the inequality constraint can be written as

$$|u_t^i| \leqslant c_r^i, \qquad i = 0, 1, 2, \ldots, N-1.$$

Since

$$u^i(t) \approx U\mathscr{P} = Up_i, \qquad i = 0, 1, 2, \ldots, N-1,$$

the inequality can be written as

$$|Up_i| \leqslant c_t^i, \qquad i=0, 1, \ldots, N-1. \tag{23}$$

The problem defined by Eqs. (20) through (21) can be solved by minimizing the composite cost function obtained from the epsilon problem and using the Rayleigh–Ritz method with a Walsh function basis [see Ref. 2],

$$J=(1/2\varepsilon)\,\mathrm{vec}(E)^T\,\mathrm{vec}(E)+(1/2)\,\mathrm{vec}(X)^T\,(I_N\otimes Q)\,\mathrm{vec}(X)$$
$$+(1/2)\,\mathrm{vec}(U)^T\,(I_N\otimes R)\,\mathrm{vec}(U), \tag{24}$$

where

$$\mathrm{vec}(E)=\{\mathrm{vec}(X)-\mathrm{vec}(G_s)-[(P^T\otimes I_n)\,\Lambda_\alpha]\,\mathrm{vec}(X),$$
$$-[(P^T\otimes I_n)\,\Lambda_\beta]\,\mathrm{vec}(U)-(P^T\otimes I_n)\,\mathrm{vec}(C)\}.$$

In the above, P is the integration matrix for Walsh functions and the Λ's are matrices of Walsh series coefficients. To solve the inequality problem, this cost function must be minimized while meeting the inequality constraint on the Walsh coefficients of the control function in Eq. (23). This constrained optimization problem can be converted into an unconstrained optimization problem by adjoining the inequality constraint to the cost function as a penalty term, for example,

$$(1/2\gamma)\,\|\max(0, |Up_i|-c_{t_i})\|^2.$$

In this case, only the constraints which are violated will appear in the penalty term. One approach is to determine, at each iteration, which constraints are active and form a matrix $\mathscr{P}'$ and vector C_t' which consist of only the active constraints. This matrix and vector are then used to form the penalty term which must be in vec form,

$$(1/2\gamma)\,\|(\mathscr{P}'^T\otimes I_n)\,\mathrm{vec}(U)-\mathrm{vec}(C_t')\|^2.$$

This term is added to the cost function of (24) by defining

$$\mathrm{vec}(\mathscr{E})=(\mathscr{P}'^T\otimes I_n)\,\mathrm{vec}(U)-\mathrm{vec}(C_t')$$

and forming the cost functional

$$J=(1/2\varepsilon)\,\mathrm{vec}(E)^T\,\mathrm{vec}(E)+(1/2)\,\mathrm{vec}(X)^T\,(I_N\otimes Q)\,\mathrm{vec}(X)$$
$$+(1/2)\,\mathrm{vec}(U)^T\,(I_N\otimes R)\,\mathrm{vec}(U)+(1/2\gamma)\,\mathrm{vec}(\mathscr{E})^T\,\mathrm{vec}(\mathscr{E}).$$

This composite cost functional can now be minimized using the matrix operations of Section 2, as before.

4.4. Example. The proposed method is demonstrated with the following example:

$$\dot{x}_1 = x_2, \qquad x_1(0) = 0,$$

$$\dot{x}_2 = -x_1 + x_2 - x_1^2 x_2 + u, \qquad x_2(0) = 1,$$

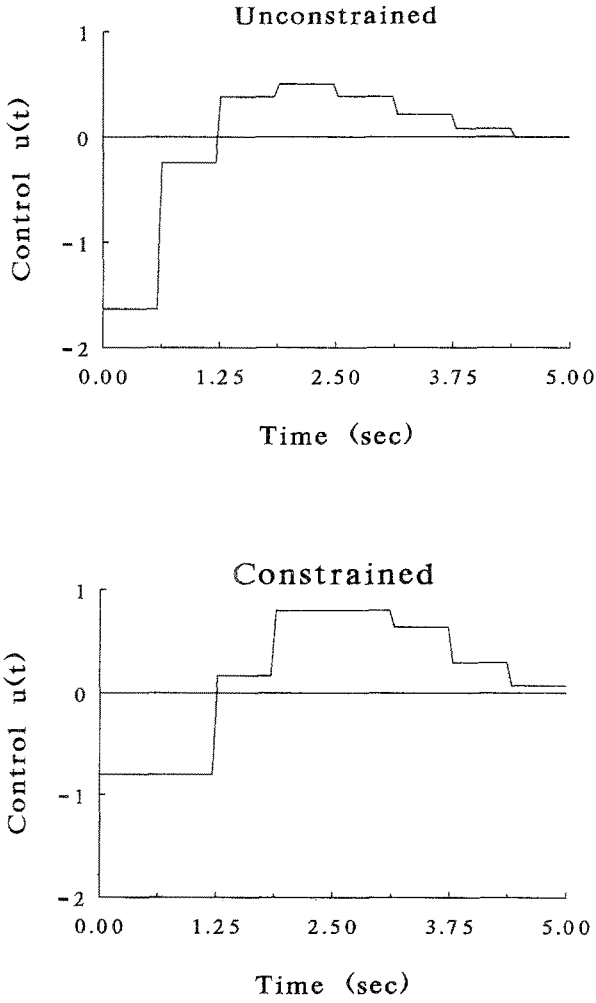

Fig. 2. Unconstrained and constrained controls using eight Walsh functions.

with

$$V = \int_0^5 (x_1^2 + x_2^2 + u^2)\, dt,$$

$$|u(t)| \leqslant 0.8.$$

Eight Walsh functions are used to solve the problem. First, the problem is solved with no constraint on the control. As illustrated in Fig. 2, the unconstrained optimal control solution would violate the inequality constraint while the optimal solution with the constraint in place yields a satisfactory solution. To demonstrate that the method can be efficiently implemented in parallel, the inequality problem was solved on an eight-processor Intel Sugarcube. The solution time on one processor was 38.8 seconds, while the solution time on all eight processors was 10.4 seconds, giving a speedup of 3.5.

5. Computational Results

5.1. Epsilon Method Using a Legendre Polynomial Basis. The proposed algorithm was implemented on an eight-processor Intel Sugarcube parallel computer using the first eight Legendre polynomials as the basis functions. For details, see Refs. 2, 20. The Van Der Pol oscillator problem with terminal state constraints was chosen as the example problem,

$$\dot{x}_1 = x_2,$$

$$\dot{x}_2 = -x_1 + x_2 - x_1^2 x_2 + u,$$

with

$$x_1(0) = 1, \qquad x_1(5) = -0.97,$$

$$x_2(0) = 0, \qquad x_2(5) = -0.96,$$

and

$$V = \int_0^5 (x_1^2 + x_2^2 + u^2)\, dt.$$

The problem took 14.1 seconds to solve on one node of the Sugarcube and 4.2 seconds to solve on all 8 nodes for a speedup of 3.5.

Solution results are summarized in Table 1. Notice that the method converges very quickly.

Table 1. Iterations 1–5 for the Van Der Pol problem.

Iteration	$(1/\varepsilon)\ \|e(t,\varepsilon)\|^2$	$(1/\varepsilon)\ \|\rho(t,\varepsilon)\|^2$	$J(\varepsilon, x, u)$
1	0.0307	0.0034	12.5862
2	0.0025	0.0001	3.4257
3	0.0040	0.0002	4.2447
4	0.0041	0.0002	4.2490
5	0.0041	0.0002	4.2490

Figure 3 shows the control generated by the method for both eight Walsh functions and eight Legendre polynomials. The controls generated by the epsilon method were tested by using them as inputs to a 4th-order Runge–Kutta solver of the nonlinear system state equations. The states are compared in Fig. 4, illustrating that both types of basis functions yield good results, Legendre polynomials providing perhaps a slightly better approximation for this type of terminal constraint problem. Neither basis gives good results for $N=4$.

5.2. Nonlinear Minimum Time Problem. Balakrishnan gave a method for solving minimum-time optimal control problems using the epsilon method (Ref. 7). Using Walsh functions and the Rayleigh–Ritz method, it is possible to generate a parallel solution method for minimum time problems. Consider the problem of moving a constantly thrusting space-

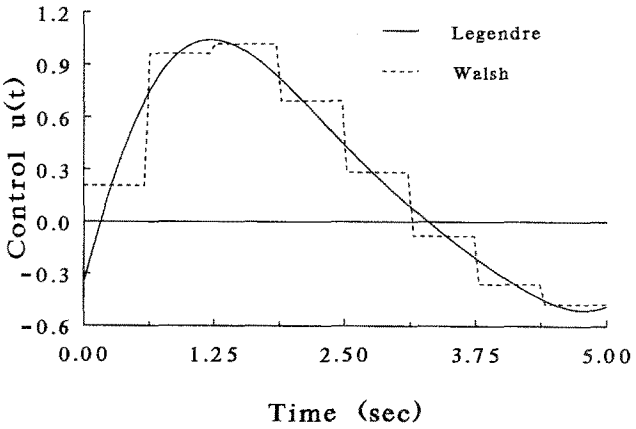

Fig. 3. Optimal control: Walsh functions versus Legendre polynomials.

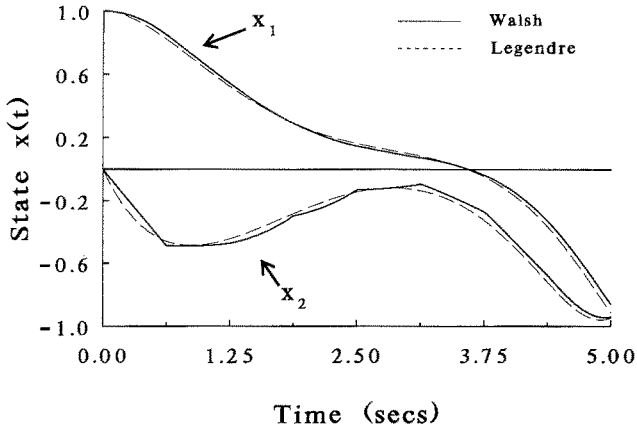

Fig. 4. Simulation results: Walsh functions versus Legendre polynomials.

craft from Earth to Mars orbit in minimum time. The dynamical equations for this problem are:

$$\dot{x}_1 = x_2, \tag{25a}$$

$$\dot{x}_2 = x_3^2/x_1 - \mu/x_1^2 + T\sin\theta/(m_0 + \dot{m}t), \tag{25b}$$

$$\dot{x}_3 = -x_2 x_3/x_1 + T\cos\theta/(m_0 + \dot{m}t), \tag{25c}$$

where $x_1(t)$ is the radial position, $x_2(t)$ is the radial velocity, $x_3(t)$ is the tangential velocity, and θ represents the thrust angle which is the control. Thrust T is constant throughout the entire flight, μ is the gravitational constant, m_0 is the initial mass, and $\dot{m}$ is the mass flow rate. The initial and final values for the state variables represent the initial and final orbits,

$$x_1(0) = 1.0, \qquad x_{1_{t_f}} = 1.525,$$

$$x_2(0) = 0.0, \qquad x_{2_{t_f}} = 0.0,$$

$$x_3(0) = 1.0, \qquad x_{3_{t_f}} = 0.8098.$$

Table 2. Minimum time versus number of Walsh functions.

Method	t_f	J_{final}
Steepest descent	3.951	3.959
Epsilon with 4 Walsh functions	4.018	4.042
Epsilon with 8 Walsh functions	3.949	3.975
Epsilon with 16 Walsh functions	3.921	3.946

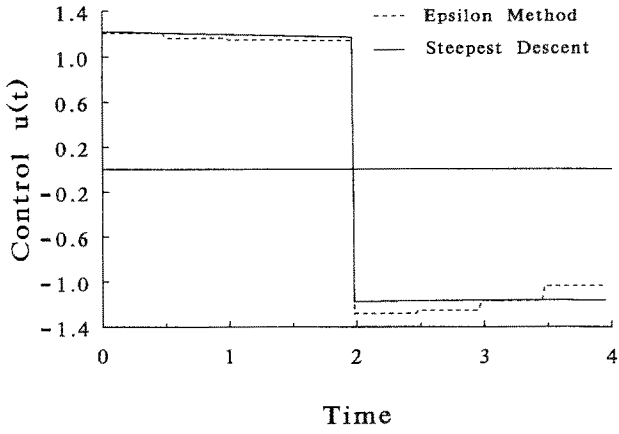

Fig. 5. Thrust angle: epsilon method with eight Walsh functions versus steepest descent.

The cost function to be minimized is given by

$$J = \int_0^{tf} 1 \, dt.$$

To obtain a reference, the problem was also solved by the method of steepest descent and the results are compared with the proposed method in the table and figures below. In Table 2, the minimum time obtained for the proposed method using 4, 8, and 16 Walsh functions and solution by steepest descent are compared. As for the Van Der Pol problem, the con-

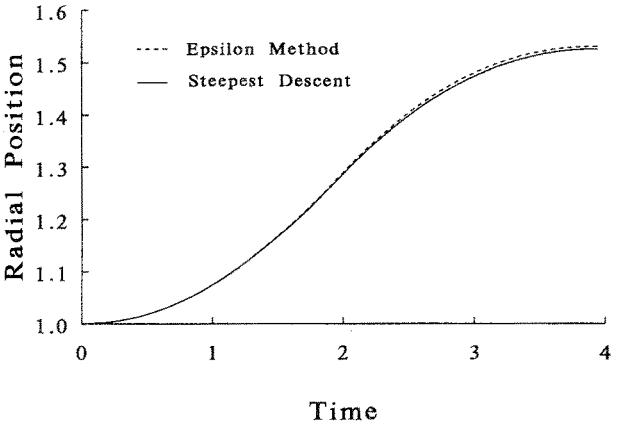

Fig. 6. Radial position: epsilon method with eight Walsh functions versus steepest descent.

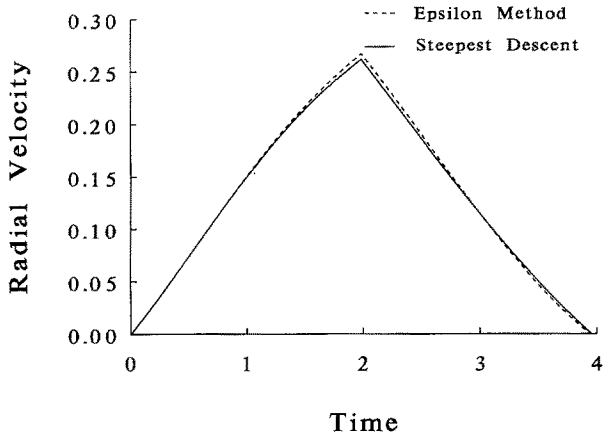

Fig. 7. Radial velocity: epsilon method with eight Walsh functions versus steepest descent.

trol generated is used to solve the dynamical equations using a 4th-order Runge–Kutta simulator to generate the state functions. Figures 5 through 8 compare the results obtained using the control generated by the epsilon method against the steepest descent solution. The epsilon solution method gives good results even when as few as eight Walsh functions are used.

6. Conclusions

A highly parallel solution method for nonlinear optimal control problems, first proposed by the authors in Ref. 2, was extended to use as its

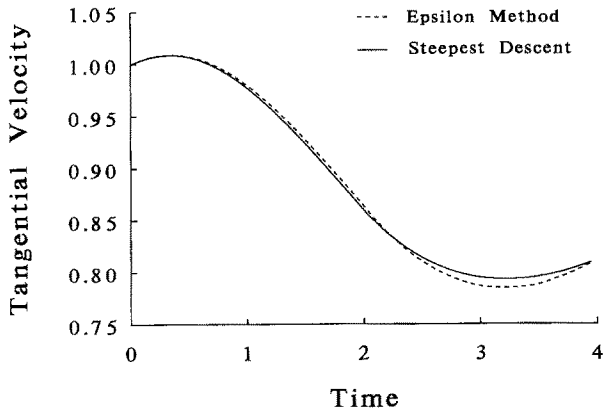

Fig. 8. Tangential velocity: epsilon method with eight Walsh functions versus steepest descent.

basis orthogonal polynomial functions such as Legendre and Chebyshev polynomials in place of the Walsh functions. Computational experiments indicate that the Walsh functions provided comparable approximation accuracy to the Legendre polynomials and require slightly less computation.

The proposed solution method was also adapted to handle optimal control problems with other than polynomial nonlinearities and problems that are subject to inequality constraints if Walsh functions are used as the basis functions. A similar adaptation for orthogonal polynomials is still under investigation.

Although the parallel computer used to demonstrate the proposed method consisted of only eight processors, the method lends itself to implementation on highly parallel computers or parallel processing arrays. For example, if the problem to be solved has two states and one control, and eight orthogonal functions are used to approximate the time-varying functions, then up to 300 processors in the form of a two-dimensional array may be used to solve the problem. We hope to explore the method's potential for extremely fast solution times using larger computer arrays.

7. Appendix

Relevant properties of the Walsh functions and orthogonal polynomials and some details on the developments and calculations of Section 2 are given below.

7.1. Properties of the Walsh Functions. Integral of Walsh Functions. It is well known (Ref. 21) that, for a given integer $p > 0$ and $N = 2^p - 1$, the integral of an N-vector of Walsh functions $\Psi(t)$ can be represented in a compact form,

$$\int_0^t \Psi(\tau)\, d\tau = P\Psi(t) = P_N \Psi(t),$$

where

$$P_N = \begin{bmatrix} P_{N/2} & -(1/2N)\, I_{N/2} \\ (1/2N)\, I_{N/2} & 0 \end{bmatrix},$$

with $P_1 = 1/2$.

Kronecker Matrix Product. Extensively used in Section 2, the Kronecker product of two matrices $A \otimes B$ is defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1q}B \\ a_{21}B & \cdots & a_{2q}B \\ \vdots & \vdots & \vdots \\ a_{p1}B & \cdots & a_{pq}B \end{bmatrix}.$$

Product of Two Walsh Functions. One of the most useful properties of Walsh functions is their group property under multiplication (Ref. 13). For any i, j with $0 \leqslant i \leqslant N$, $0 \leqslant j \leqslant N$, and $N = 2^p - 1$ for some integer $p > 0$, we have

$$\phi_i(t)\, \phi_j(t) = \phi_k(t),$$

where $k = i \otimes j$ and $\otimes$ denotes no-carry bitwise addition.

7.2. Properties of Orthogonal Polynomials. A polynomial is orthogonal with respect to some weight function $w(t)$. In general, orthogonal polynomials have the following property:

$$\int_a^b w(t)\, \phi_i(t)\, \phi_j(t)\, dt = \begin{cases} 0, & i \neq j, \\ \|\phi_i\|, & i = j. \end{cases} \tag{26}$$

For example, the weight function $w(t)$ for Legendre polynomials is unity and, if $a = 0$ and $b = t_f$, then

$$\|\phi_i\| = t_f / (2i + 1).$$

For Walsh functions, (26) is also true. In this case, the weight function is unity and

$$\|\phi_i\| = 1.$$

Orthogonal polynomials have interesting integration properties. In general, if $S(t)$ is a vector of orthogonal polynomials,

$$S(t) = \begin{bmatrix} s_0(t) \\ s_1(t) \\ \vdots \\ s_{N-1}(t) \end{bmatrix},$$

then

$$\int_0^t S(\tau)\, d\tau = HS(t),$$

where H is called the integration matrix. For example, the integration matrix for Legendre polynomials is given as

$$H_N = \frac{t_f}{2}\begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ -1/3 & 0 & 1/3 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1/5 & 0 & 1/5 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -1/(2N-3) & 0 & 1/(2N-3) \\ 0 & 0 & 0 & 0 & \cdots & 0 & -1/(2N-1) & 0 \end{bmatrix}.$$

This allows the integral of a vector of orthogonal polynomials to be found by a matrix-vector multiplication.

Finally, it can be shown that

$$S(t)\, S^T(t)\, c^T = \Lambda_c S(t), \tag{27}$$

where c is a vector of coefficients and Λ_c is a matrix of the coefficients from c. The resulting $S(t)\, S^T$ is dependent on the specific polynomials being used.

7.3. Quadratic Forms. A compact form for the integral of the quadratic forms in the cost function (7) can be obtained as follows:

$$S^T(t)\, MS(t) = \sum_{i=0}^{N-1} s_i(t) \sum_{j=0}^{N-1} m_{ij} s_j(t);$$

therefore,

$$\int_0^{t_f} S^T(t)\, MS(t)\, dt = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} m_{ij} \int_0^{t_f} s_i(t)\, s_j(t)\, dt.$$

Since the matrix O,

$$O \equiv \int_0^{t_f} s_i(t)\, s_j(t)\, dt,$$

is symmetric, we have

$$\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} m_{ij} \int_0^{t_f} s_i(t)\, s_j(t)\, dt = \sum_{i=0}^{N-1} m_{ii} o_{ii} = \text{trace}(MO).$$

O is a matrix which results from the orthogonal property of the particular polynomial being used.

7.4. Vector Form of Coefficients for the Error Penalty Term. To approximate $e(t;\varepsilon)$ in terms of orthogonal polynomials, the time-varying integrals in the penalty term must be approximated. This can be done using the multiplication and integral properties of orthogonal polynomials. Consider first the scalar case where $n=1$. Let

$$A(t)\approx \alpha S(t)=S(t)^T\alpha^T,$$

where α is a vector of coefficients, $\alpha=[\alpha_0,\alpha_1,\ldots,\alpha_{N-1}]$, and

$$x(t)\approx XS(t).$$

Then,

$$A(t)\,x(t)\approx XS(t)\,S^T(t)\,\alpha^T,$$

or using (27),

$$A(t)\,x(t)\approx X\Lambda_\alpha S(t).$$

For the scalar case,

$$\int_0^t A(\tau)\,x(\tau)\,d\tau\approx X\Lambda_\alpha HS(t). \tag{28}$$

Now consider the case where $n>1$. Then,

$$A(t)=\begin{bmatrix} a_{11}(t) & a_{12}(t) & \cdots & a_{1n}(t)\\ a_{21}(t) & & & \\ \vdots & & & \vdots \\ a_{n1}(t) & \cdots & & a_{nn}(t)\end{bmatrix},\qquad x(t)=\begin{bmatrix} x_1(t)\\ x_2(t)\\ \vdots\\ x_n(t)\end{bmatrix}.$$

Using (28), the time-varying integral for $n>1$ can be written as

$$\int_0^t A(\tau)\,x(\tau)\,d\tau\approx\begin{bmatrix} X_1\Lambda_{\alpha_{11}}HS(t)+\cdots+X_n\Lambda_{\alpha_{1n}}HS(t)\\ \vdots \qquad\qquad\qquad\qquad \vdots\\ X_1\Lambda_{\alpha_{n1}}HS(t)+\cdots+X_n\Lambda_{\alpha_{nn}}HS(t)\end{bmatrix}, \tag{29}$$

where

$$\alpha_{11}=[\alpha_{11}^0,\alpha_{11}^1,\ldots,\alpha_{11}^{N-1}].$$

Here the superscript denotes the coefficient number, so

$$a_{11}(t)\,x_1(t)=\alpha_{11}S(t)\,S^T(t)\,X_1.$$

Now the polynomial coefficients for (29) can be rearranged into vec notation,

$$\operatorname{vec}(X\Lambda_{\alpha}^{(3\dim)}P) = [(H^T \otimes I_n)\,\Lambda_{\alpha}^{(2\dim)}]\,\operatorname{vec}(X).$$

If the same approach is taken for the other integrals in $e(t;\varepsilon)$, then

$$\begin{aligned}\operatorname{vec}(E) = \{&\operatorname{vec}(X) - \operatorname{vec}(G_s) - [(H^T \otimes I_n)\,\Lambda_{\alpha}]\,\operatorname{vec}(X)\\ &- [(H^T \otimes I_n)\,\Lambda_{\beta}]\,\operatorname{vec}(U) - (H^T \otimes I_n)\,\operatorname{vec}(C)\}.\end{aligned}$$

7.5. Vector Form of Coefficients for the Error Penalty Term. Let $\rho(\varepsilon) \approx \xi S(t)$. The coefficients $\operatorname{vec}(\xi)$ must be found. Since $\rho(\varepsilon)$ is a vector of constants, only the zeroth polynomial coefficients are nonzero. Consider first the scalar case. Then,

$$\int_0^{t_f} A(t)\,x(t)\,dt \approx \int_0^{t_f} \alpha S(t)\,S^T(t)\,X^T\,dt.$$

Using the orthogonal property, this can be written as

$$\int_0^{t_f} \alpha S(t)\,S^T(t)\,X^T\,dt = \alpha O X^T.$$

For the case where $n > 1$, we have

$$\int_0^{t_f} A(t)\,x(t)\,dt = \begin{bmatrix} \int_0^{t_f} a_{11}(t)\,x_1(t)\,dt + \cdots + \int_0^{t_f} a_{1n}(t)\,x_n(t)\,dt \\ \vdots \\ \int_0^{t_f} a_{n1}(t)\,x_1(t)\,dt + \cdots + \int_0^{t_f} a_{nn}(t)\,x_n(t)\,dt \end{bmatrix}.$$

The above matrix can be written as

$$\begin{bmatrix} \alpha_{11}O & \cdots & \alpha_{1n}O \\ \vdots & & \vdots \\ \alpha_{n1}O & \cdots & \alpha_{nn}O \end{bmatrix} \begin{bmatrix} X_1^T \\ \vdots \\ X_n^T \end{bmatrix}.$$

Here, the X-vector is not in standard vec form, but it can be rearranged into standard vec form,

$$\begin{bmatrix} \left[\text{vec}\begin{pmatrix} \alpha_{11}O \\ \alpha_{12}O \\ \vdots \\ \alpha_{1n}O \end{pmatrix}\right]^T \\ \left[\text{vec}\begin{pmatrix} \alpha_{21}O \\ \alpha_{22}O \\ \vdots \\ \alpha_{2n}O \end{pmatrix}\right]^T \\ \vdots \\ \left[\text{vec}\begin{pmatrix} \alpha_{n1}O \\ \alpha_{n2}O \\ \vdots \\ \alpha_{nn}O \end{pmatrix}\right]^T \end{bmatrix} \text{vec}(X). \tag{30}$$

This matrix vector mutiplication gives the coefficients of the zeroth polynomial, with the rest being zero. To get a matrix of coefficients for the entire coefficient vector, a matrix Γ_α is formed. The first n rows are obtained from (30) and the rest of the matrix is zero. Using Γ_α, it is possible to write

$$\int_0^{t_f} A(t)\, x(t)\, dt = \int_0^{t_f} \alpha S(t)\, S^T(t)\, X\, dt = \Gamma_\alpha\, \text{vec}(X).$$

The same approach can be used to approximate the integral with $B(t)\, u(t)$. For the last integral a slightly different approach is used. Let

$$C(t) = CS(t).$$

Then

$$\int_0^{t_f} CS(t)\, dt = C \int_0^{t_f} S(t)\, dt.$$

The above integral is a vector with the first element being a one and the remaining N being zero and can be written in vec form,

$$\int_0^{t_f} CS(t)\, dt = \Gamma_c\, \text{vec}(C).$$

Now $\text{vec}(\xi)$ can be written as

$$\text{vec}(\xi) = \text{vec}(G_s) + \Gamma_\alpha\, \text{vec}(X) + \Gamma_\beta\, \text{vec}(U) + \Gamma_c\, \text{vec}(C) - \text{vec}(G_f).$$

References

1. BALAKRISHNAN, A. V., *On a New Computing Technique in Optimal Control*, SIAM Journal on Control, Vol. 6, No. 2, pp. 149–173, 1968.
2. FRICK, P. A., and STECH, D. J., *A New Approach for the Solution of Optimal Control Problems on Parallel Machines*, Proceedings of the 7th International Conference on Systems Engineering, Las Vegas, Nevada, pp. 286–292, 1990.
3. STYCZEN, K., *Trigonometric Approximation of Optimal Periodic Control Problems*, International Journal of Control, Vol. 43, No. 5, pp. 1531–1542, 1986.
4. SIRISENA, H. R., and CHOU, F. S., *An Efficient Algorithm for Solving Optimal Control Problems with Linear Terminal Constraints*, IEEE Transactions on Automatic Control, Vol. 21, No. 2, pp. 275–277, 1976.
5. TEO, K. L., GOH, C. J., and WONG, K. H., *A Unified Computational Approach to Optimal Control Problems*, Longman Scientific and Technical, London, England, 1991.
6. FRICK, P. A., *An Integral Formulation of the Epsilon Problem and a New Computational Approach to Control Function Optimization*, Journal of Optimization Theory and Applications, Vol. 13, No. 5, pp. 553–581, 1974.
7. BALAKRISHNAN, A. V., *On a New Computing Technique in Optimal Control and Its Applications to Minimal Time Flight Profile Optimization*, Journal of Optimization Theory and Applications, Vol. 4, No. 1, pp. 1–21, 1968.
8. HESTENES, M. R., *Multiplier and Gradient Methods*, Journal of Optimization Theory and Applications, Vol. 4, No. 5, pp. 303–332, 1969.
9. REKTORYS, K., *Variational Methods in Mathematics, Science, and Engineering*, D. Reidel, Dordrecht, Holland, 1977.
10. JONES, A. P., and MCCORMICK, G. P., *A Generalization of the Method of Balakrishnan to Inequality Constraints and Initial Conditions*, SIAM Journal on Control, Vol. 8, No. 2, pp. 219–225, 1970.
11. DI PILLO, G., GRIPPO, L., and LAMPARIELLO, F., *The Multiplier Method for Optimal Control Problems*, Proceedings of the CSEI Conference on Optimization Problems in Engineering and Economics, Naples, Italy, pp. 326–341, 1974.
12. DI PILLO, G., and GRIPPO, L., *A Computing Algorithm for the Application of the Epsilon Method to Identification and Optimal Control Problems*, Ricerche di Automatica, Vol. 3, No. 1, pp. 54–77, 1972.
13. FINE, N. J., *On the Walsh Functions*, Transactions of the American Mathematical Society, Vol. 65, No. 1, pp. 372–413, 1949.
14. BREWER, J. W., *Kronecker Products and Matrix Calculus in System Theory*, IEEE Transactions on Circuits and Systems, Vol. 25, No. 9, pp. 772–781, 1978.
15. BERTSEKAS, D. P., and TSITSIKIS, J. N., *Parallel and Distributed Computation–Numerical Methods*, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.
16. KUNG, S. Y., *VLSI Array Processors*, Prentice-Hall, Englewood Cliffs, New Jersey, 1988.
17. MIELE, A., *Recent Advances in Gradient Algorithms for Optimal Control Problems*, Journal of Optimization Theory and Applications, Vol. 17, Nos. 5–6, pp. 361–430, 1975.

18. BELLMAN, R. E., and KALABA, R. E., *Quasilinearization and Nonlinear Boundary-Value Problems*, Elsevier Press, New York, New York, 1965.
19. BOLANCZYK, A., BRENT, R. P., and KUNG, H. T., *Numerically Stable Solution of Linear Equations Using Mesh-Connected Processors*, SIAM Journal on Scientific and Statistical Computing, Vol. 5, No. 3, pp. 95–123, 1984.
20. HWANG, C., and CHEN, M., *Analysis and Optimal Control of Time-Varying Linear Systems via Shifted Legendre Polynomials*, International Journal of Control, Vol. 41, No. 5, pp. 1317–1330, 1985.
21. CHEN, C., and HSIAO, C., *Design of Piecewise Constant Gains for Optimal Control via Walsh Functions*, IEEE Transactions on Automatic Control, Vol. 20, No. 5, pp. 596–602, 1975.