[14] C. H. Lawson and R. J. Hanson, *Solving Least Squares Problems.* Englewood Cliffs, NJ: Prentice-Hall, 1974.
[15] M. B. Leahy and G. N. Saridis, "Compensation of industrial manipulator dynamics," *Int. J. Robotics Res.*, vol. 8, no. 4, pp. 73–84, 1989.
[16] S. K. Lin, "Microprocessor implementation of the inverse dynamic system for industrial robot control," in *Proc. 10th IFAC World Congress Automat. Contr.*, vol. 4, 1987, pp. 332–339.
[17] ——, *Identification of the inertia parameters of an industrial robot*, National Science Council, Taiwan, NSC Proj. Rep. NSC80-0404-E-009-31, 1991.
[18] ——, "An identification method for estimating the inertia parameters of a manipualtor," *J. Robotic Systems*, vol. 9, no. 4, pp. 505–528, 1992.
[19] ——, "Estimating minimal parameters of a manipulator for gravity load and friction," in *Proc. 12th IFAC World Congress Automat. Contr.*, vol. 3, 1993, pp. 305–310.
[20] H. Mayeda, K. Osuka, and Kangawa, "A new identification method for serial manipulator arms," in *Proc. IFAC 9th World Congress*, 1984, p. 2429–2434.
[21] H. Mayeda, K. Yoshida, and K. Osuka, "Base parameters of dynamics models for manipulators with rotational and translational joints," in *Proc. 1989 IEEE Conf. Robotics and Automation*, 1989, pp. 1523–1528.
[22] ——, "Base parameters of manipulator dynamics models," *IEEE Trans. Robotics and Automation*, vol. 6, pp. 312–321, 1990.
[23] H. Mayeda and K. Ohashi, "Base parameters of dynamic models for general open-loop kinematic chains," in *Proc. 5th Int. Symp. Robotics Research*, 1990, pp. 271–278.
[24] A. Mukerjee and D. H. Ballard, "Self-calibration in robot manipulations," in *Proc. 1985 IEEE Conf. Robotics and Automation*, 1985, pp. 1050–1057.
[25] H. B. Olsen and G. A. Bekey, "Identification of robot dynamics," in *Prob. 1986 IEEE Conf. Robotics and Automation*, 1986, pp. 1004–1010.
[26] K. Otani, T. Kakizaki, and K. Kogure, "Dynamic parameter identification of an industrial robot and its application to trajectory controls," in *Proc. 1992 IEEE/RSJ Conf. Intelligent Robots and Systems*, 1992, pp. 990–997.
[27] B. Raucent, J. C. Samin, and P. Y. Willems, "Identification of dynamic parameters of robot manipulator from external measurements," *European J. Mech. Eng.*, vol. 34, no. 1, pp. 21–26, 1989.
[28] B. Raucent, G. Bastin, J. C. Samin, and P. Y. Willems, "Identification of the barycentric parameters of robot manipulator from external measurements," *Automatica*, vol. 28, no. 5, pp. 1011–1016, 1992.
[29] R. Tsai and S. K. Lin, "Design and implementation of the torque control for a dc servo motor," in *Proc. 14th National Symp. Automat. Contr.*, Taiwan, 1990, pp. 565–572.
[30] J. T. Wen and D. S. Bayard, "New class of control laws for robotic manipulators, part 1: Non-adaptive case," *Int. J. Contr.*, vol. 47, pp. 1361–1385, 1988.
[31] H. West, E. Papadopoulos, S. Dubowsky, and H. Cheah, "A method for estimating the mass properties of a manipulator by measuring the reaction moments at its base," in *Proc. 1989 IEEE Conf. Robotics and Automation*, 1989, pp. 1510–1516.
[32] K. Yoshida, N. Ikeda, and H. Mayeda, "Experimental study of the identification methods for an industrial robot manipulator," in *Proc. 1992 IEEE/RSJ Conf. Intelligent Robots and Syst.*, 1992, pp. 263–270.

A Hardware Implementable Receding Horizon Controller for Constrained Nonlinear Systems

Shin-Yeu Lin

Abstract— **We present a two-phase parallel computing method for obtaining an implementable receding horizon control solution for constrained nonlinear systems. The phase 1 method solves a feasibility problem to find an approximate open-loop admissible control and horizon pair. The phase 2 method successively improves the admissible control to obtain an implementable open-loop receding horizon control solution. We briefly sketch an approach to realizing this two-phase method using VLSI array processors. Solution times for simulation examples estimated on the basis of current VLSI technology confirm that our controller is well suited to the stabilization of real-time processing, fast, constrained nonlinear systems.**

I. Introduction

For a nonlinear system with control constraints described by $\dot{x}(t) = f^0(x(t), u(t))$, $u(t) \in \Omega$, where $f^0: \mathfrak{R}^n \times \mathfrak{R}^p \to \mathfrak{R}^n$ is twice continuously differentiable and satisfies $f^0(0, 0) = 0$ and Ω is the set of admissible controls, Mayne and Michalska showed in [1] that feedback stabilization can be achieved by a conceptual receding horizon control. Subsequently, in [2] they proposed an implementable receding horizon controller which employed a hybrid system, $\dot{x}(t) = f^0(x(t), u(t))$, when $x(t) \notin W$; otherwise $\dot{x}(t) = Ax(t)$, where $A = f_x^0(0, 0) + f_u^0(0, 0)K$ is formed by applying a linear feedback control $u = Kx$ to the linearized system in a neighborhood W with small enough radius and centered at origin. Their algorithm for this implementable receding horizon controller first calculates an admissible control and horizon pair

$$[u_0, t_{f,0}] \in Z_W(x_0) \tag{1}$$

where the initial state $x_0 \notin W$ is assumed, the set $Z_W(\cdot)$: $Z_W(x) \equiv \{u \in S,\ t_f \in (0, \infty) | x^u(t + t_f;\ x, t) \in \delta W\}$, where S denotes the set of all piecewise continuous functions, and δW denotes the boundary of W. The algorithm then sets $h = 0$, $t_h = 0$ and performs the following process repeatedly to yield the receding horizon feedback control: It applies the obtained control u_h for $x \notin W$ or the linear feedback control Kx for $x \in W$ to the real system over $[t_h, t_h + \Delta t]$, where $\Delta t \in (0, \infty)$. Let x_{h+1} be the resulting state at $t_{h+1} (\equiv t_h + \Delta t)$, then if $x_{h+1} \in W$, the algorithm switches the control to $u = Kx$ over (t_{h+1}, ∞); otherwise, it calculates an improved control and horizon $[u_{h+1}, t_{f,h+1}]$ in the sense that

$$[u_{h+1}, t_{f,h+1}] \in Z_W(x_{h+1}), \text{ and } V(x_{h+1}, t_{h+1}, u_{h+1}, t_{f,h+1}) \leq V(x_{h+1}, t_{h+1}, u_h, t_{f,h} - \Delta t) \tag{2}$$

where $V(x, t, u, t_f) \equiv \int_t^{t+t_f} (1/2)(\|x^u(\tau; x, t)\|_Q^2 + \|u(\tau)\|_R^2)\, d\tau + \int_{t+t_f}^{\infty} (1/2)(\|x_L(s; x^u(t+t_f; x, t))\|_Q^2)\, ds$, in which R and Q are positive definite matrices, $\|y\|_A^2$ denotes $y^T Ay$, and x_L denotes the state trajectory in region W with feedback control $u = Kx$.

Mayne and Michalska showed in [2] and [3] that the above implementable receding horizon controller is globally asymptotically

Manuscript received October 10, 1992; revised September 24, 1993 and December 2, 1993. This work was supported by National Science Council of Taiwan under Grant NSC 82-0404-E009-356.

The author is with the Department of Control Engineering, National Chiao Tung University, Hsinchu, Taiwan.

IEEE Log Number 9402736.

stable in the absence of disturbance, and there exists a finite time $\bar{t}$ such that $x(\bar{t}) \in W$ provided the following three assumptions are satisfied:

1) For every $x \in \Re^n$, there exists a control function $u \in S$ and a $t_f \in (0, \infty)$ such that $u(t) \in \Omega$, $\forall t$, $x^u(t+t_f;\ x, t) \in W$.
2) The linearized system (at the origin) $\dot{x}(t) = Ax(t)$ is stabilizable in the region W.
3) $Kx \in \Omega$, $\forall x \in W$.

We see that stabilization of this implementable receding horizon controller is achieved at the cost of the high on-line computational complexity of solving the admissible control and horizon pair either from (1) or from (2). The purpose of this note is to develop a strategy for coping with this computational complexity for fast nonlinear systems. Our idea is to develop methods that can be implemented by very large scale integration (VLSI) array processors to obtain the receding horizon feedback control solution. The motivation for this idea is the many recent successes in the area of image processing [12]. To achieve our goal, we will first rewrite (1) and (2). Let W_1 be a subset of W that contains the origin $x = 0$, then the stabilization property will still hold for the implementable receding horizon controller if we replace W by W_1 in $Z_W(x_h)$ and the hybrid system. In particular, we let $W_1 = \{x \mid \|x\|_2 \le \epsilon\}$, where $\epsilon > 0$ is a small enough real number, and, for the sake of clarity, we use $Z_\epsilon = \{u \in S,\ t_f \in (0, \infty) \mid \|x^u(t+t_f;\ x, t)\|_2 < \epsilon\}$ instead of Z_W for W replaced by W_1. Thus, if $\epsilon = 10^{-3}$, $Q = I$, and (suppose) the magnitude of the assigned eigenvalues of $\dot{x} = Ax$ is around 10, then the value of $\int_{t+t_f}^{\infty} (1/2)(\|x_L(s;\ x^u(t+t_f;\ x, t))\|_Q^2)\, ds$ is approximately on the order of 10^{-8}, which is negligible. Hence, to evaluate the value of $V(t, x, u, t_f)$, we may consider the term $\int_t^{t+t_f} (1/2)(\|x^u(\tau, x, t)\|_Q^2 + \|u(\tau)\|_R^2)\, d\tau$ only. Accordingly, we can rephrase (1) and (2) as follows.

Calculate a control and horizon pair

$$[u_h', t_{f,h}] \in Z_\epsilon(x_h) \tag{3–h}$$

then solve the following optimal control problem approximately

$$\min_u \int_{t_h}^{t_h+t_{f,h}} \frac{1}{2}(\|x^u(\tau;\ x_h, t_h)\|_Q^2 + \|u(\tau)\|_R^2)\, d\tau \tag{4a–h}$$

$$u(\tau) \in \Omega, \quad t_h \le \tau \le t_h + t_{f,h}, \quad \|x(t_h + t_{f,h})\|_2 < \epsilon \tag{4b–h}$$

by applying a descent method for a finite number of iterations to obtain a control solution u_h that is better than u_h' in the sense of reducing the performance index (4a–h). For $h > 0$ we will use

$$\breve{u}_h(\cdot)\colon \breve{u}_h(\tau) \equiv \begin{cases} u_h(\tau), & t_{h+1} \le \tau \le t_{h+1} + t_{f,h} - \Delta t \\ 0, & t_{h+1} + t_{f,h} - \Delta t < \tau \le t_{h+1} + t_{f,h} \end{cases}$$

and $t_{f,h}$ as initial guesses in our method to seek the control and horizon pair $[u_{h+1}, t_{f,h+1}] \in Z_\epsilon(x_{h+1})$. The above procedure may seem redundant, because $[\breve{u}_h, t_{f,h}] \in Z_\epsilon(x_{h+1})$ already holds. In fact it is not, however, because this procedure ensures that the global stabilization of the receding horizon controller will not be affected by possible perturbations in state that might cause $[u_{h+1}, t_{f,h+1}]$ to become inadmissible.

II. Problem Statement

To circumvent some complications in presenting our method, we add one additional minor assumption, which holds for most physical systems to the control constraint set.

Assumption A0: Let Ω' be the actual physical control constraint set. We assume that there exists a nonempty convex polytope $\Omega \subseteq \Omega'$ such that this Ω meets the requirements in assumptions A1)–A3).

We see that if an optimization method aims to solve a nonlinear optimal control problem with exact terminal constraint $x(t_h+t_{f,h}) = 0$ but arbitrarily stops after a finite number of iterations, then the resulting terminal state $x(t_h+t_{f,h})$ will only lie within a neighborhood of 0. However, $x(t_h+t_{f,h})$ can be set to be arbitrarily close to zero after an arbitrarily large number of iterations. Thus, keeping in mind that we will stop our methods after an arbitrarily large number of iterations, we may replace the relaxed terminal constraint in (3–h) and (4a–h)–(4b–h) by the exact terminal constraint without violating the theoretical validity of the implementable receding horizon control. That is, we replace $Z_\epsilon(x_h)$ in (3–h) and $\|x(t_h + t_{f,h})\|_2 < \epsilon$ in (4b–h) by $Z_0(x_h) = \{u \in S,\ t_{f,h} \in (0, \infty) \mid x^u(t_h + t_{f,h};\ x_h, t_h) = 0\}$ and $x(t_h + t_{f,h}) = 0$, respectively. We will refer to (3–h) and (4a–h)–(4b–h) after these replacements as (3–h) and (4a–h)–(4b–h) with exact terminal constraints. Then (3–h) with exact terminal constraints is a feasibility problem. Using the slack variable s, we can reformulate (3–h) with exact terminal constraints as

$$\min_{u,\, t_f} \left\{ \int_t^{t+t_f} s^T(\tau)s(\tau)\, d\tau + H's^T(t+t_f)s(t+t_f) \,\middle|\, \dot{x}(\tau) - f^0(x(\tau), u(\tau)) + s(\tau) = 0, \quad \mathrm{a}u(\tau) + b \le 0, \quad t \le \tau \le t + t_f, \quad x(t+t_f) + s(t+t_f) = 0 \right\} \tag{5}$$

where the control constraint set Ω has been replaced by $\{u(t) \in \Re^p \mid au(t) + b \le 0,\ a \in \Re^{q\times p},\ b \in \Re^q\}$ based on assumption A0), H' is an arbitrary and positive penalty coefficient, and we neglect the subscript h for notational convenience. In the sequel, we will refer to (5) as the phase 1 problem.

Once (5) is solved approximately, then, letting $\tilde{t}_f$ be the obtained horizon, we may rewrite (4a–h)–(4b–h) with exact terminal constraints as

$$\min \left\{ \int_t^{t+\tilde{t}_f} \frac{1}{2}[x^T(\tau)Qx(\tau) + u^T(\tau)Ru(\tau)]\, d\tau \,\middle|\, \dot{x}(\tau) - f^0(x(\tau), u(\tau)) = 0, \quad \mathrm{a}u(\tau) + b \le 0, \quad t \le \tau \le t + \tilde{t}_f, \quad x(t+\tilde{t}_f) = 0 \right\}. \tag{6}$$

In the sequel, we will refer to (6) as the phase 2 problem.

III. The Two-Phase Method

A. Method for Solving the Phase 1 Problem

1) The Two-Level Method: Our method will be developed using nonlinear programming techniques. Therefore, we need to discretize (5) first. Let t_f be divided into N time intervals and let $s_i := \frac{t_f}{N} s_i$, $i = 0, \cdots, N-1$, and s_N be a new set of slack variables. Then, by the fact that multiplying the objective function of (5) by a positive constant results in an equivalent problem, (5) can be discretized as shown below

$$\min \left\{ \sum_{i=0}^{N-1} s_i^T s_i + Hs_N^T s_N \,\middle|\, x_{i+1} - x_i - \frac{t_f}{N} f(x_i, u_i, t_f) + s_i = 0, \quad x_0 = x(t), \quad au_i + b \le 0, \quad i = 1, 2, \cdots, N-1, \quad x_N + s_N = 0 \right\} \tag{7}$$

where f is the function resulting from using the Runge–Kutta method to discretize the state equation and $H = \frac{t_f}{N}H'$. Without loss of generality, we may set $H = 1$ in (7).

The solution set of (7) has a wider range than the minimum-time control solution, because the latter is one of the solutions of (7). Thus, a simpler version of the two-level, master-slave approach presented in [6] can be employed here. We use a very simple decision process for the master problem rather than a gradient iteration, as used in [6].

Master Problem:

$$\begin{cases} t_f(l+1) = t_f(l) + \delta t, & \text{if } \sum_{i=0}^{N} \hat{s}_i^T(t_f(l))\hat{s}_i(t_f(l)) > \epsilon_1 \\ \text{Stop}, & \text{otherwise}, \end{cases} \tag{8}$$

where $\delta t > 0$ is a small increment on the horizon, ϵ_1 is a very small positive real number, and $\hat{s}_i(t_f(l))$, $i = 0, \cdots, N$, denotes the optimal s of the following slave problem, under a $t_f(l)$ given from the master problem.

Slave Problem:

$$\min_{x,\,u,\,s} \left\{ \sum_{i=0}^{N} s_i^T s_i \middle| x_{i+1} - x_i - \frac{t_f(l)}{N} f(x_i,\, u_i,\, t_f(l)) + s_i = 0, \quad x_0 = x(t), \quad au_i + b \le 0, \quad i = 0, \cdots, N-1, \quad x_N + s_N = 0 \right\}. \tag{9}$$

Remark 1: We use ϵ_1 instead of 0 in (8) because we need only a good approximate solution for (7).

2) Combining the Scaled Gradient Projection Method with a Dual Jacobi Method for the Slave Problem: The slave problem has been solved in [6] using a combination of recursive quadratic programming and a dual gradient method that can be implemented by VLSI array processors. In the updating steps of that iterative method, the variable step sizes are determined based on one-dimensional minimization. The step sizes, however, were set as constants in [6] for the purpose of hardware implementation. In fact, the recursive quadratic programming method and the dual gradient method can be viewed as special versions of the scaled gradient projection method and the scaled gradient method, respectively, which uses a restrictive constant step size and converges under certain easily satisfied conditions [5]. Like the dual gradient method, the dual Jacobi method with constant step size is also a type of scaled gradient method. Thus, to solve the slave problem we will replace the variable step size in the recursive quadratic programming method by a constant step size and the dual gradient method by the dual Jacobi method with constant step size. Therefore, the iterative method (a combination of the scaled gradient projection method and the dual Jacobi method) we will use to solve the slave problem is almost the same as the method used in [6], and we will describe this method only briefly here because of the limitation on the length of this note. Following the notation in [6], the slave problem can be rewritten as follows

$$\min_{y,\,s} \left\{ \sum_{i=0}^{N} s_i^T s_i \middle| x_{i+1} - x_i - \frac{t_f}{N} f(y_i,\, t_f) + s_i = 0, x_0 = x(t), \quad a'u_i + b' + z_i = 0, \quad \underline{y_i} \le y_i \le \overline{y_i}, \quad i = 0, 1, \cdots, N-1, \quad x_N + s_N = 0 \right\} \tag{10}$$

where $y_i \equiv (x_i,\, u_i,\, z_i) \in \Re^{n+p+r}$, $\underline{y_i} \equiv (-\infty,\, \underline{u_i},\, 0)$, $\overline{y_i} \equiv (+\infty,\, \overline{u_i},\, +\infty)$, $u_N \equiv 0$, $z_N \equiv 0$, $\underline{u_i} = -\infty$ if u_i is unbounded from below, $\overline{u_i} = +\infty$ if u_i is unbounded from above, $z_i (\ge 0)$ are variables to convert the $r_i (\le q)$ nonsimple inequality constraints $a'u_i + b' \le 0$ into equality constraints, $f(y_i,\, t_f) \equiv f(x_i,\, u_i,\, t_f)$, and $t_f(l)$ has been replaced by t_f for the sake of notational convenience.

Let dx, du, and dz denote $(dx_0, \cdots, dx_N)$, $(du_0, \cdots, du_N)$, and $(dz_0, \cdots, dz_N)$, respectively, where $dx_0 \equiv 0$, $du_N \equiv 0$, and $dz_N \equiv 0$. The scaled gradient projection method uses the following iterations

$$y(k+1) = y(k) + dy^* \tag{11}$$

to solve (10), where $dy^* \equiv (dx^*,\, du^*,\, dz^*)$ associated with s^* is the solution of the quadratic approximation of (10) at $y(k)$ described in the following

$$\min_{dy,\,s} \left\{ \sum_{i=0}^{N} s_i^T s_i + \frac{1}{\gamma_1} dy_i^T dy_i \middle| E_i(y(k)) + dx_{i+1} - dx_i - \frac{t_f}{N} f_{y_i}(k) dy_i + s_i = 0, \quad dx_0 = 0, \quad E_N(y(k)) + dx_N + s_N = 0, \quad a'u_i(k) + b' + z_i(k) + a'du_i + dz_i = 0, \quad \underline{y_i} \le y_i(k) + dy_i \le \overline{y}_i, \quad i = 0, \cdots, N-1 \right\} \tag{12}$$

where γ_1 is a positive scalar, $E_i(y(k)) = x_{i+1}(k) - x_i(k) - \frac{t_f}{N} f(y_i(k),\, t_f)$, $i = 0, \cdots, N-1$, $E_N(y(k)) = x_N(k)$, $f_{y_i}(k) = (f_{x_i}(k),\, f_{u_i}(k),\, 0)$, in which $f_{x_i}(k) = (\partial f(y_i(k))/\partial x_i)$ and $f_{u_i}(k) = (\partial f(y_i(k))/\partial u_i)$, and $dy_i = (dx_i,\, du_i,\, dz_i)$. Because of the slack variables s_i, $i = 0, \cdots, N$, and the fact that Ω is a convex polytope [assumption A0], the constraint qualification for (12) is easily satisfied. Furthermore, the Hessian matrix of (12) is positive definite, so based on the strong duality theorem [8], we may obtain the solution of (12) by solving the corresponding dual problem shown below

$$\max \phi(\lambda) \tag{13}$$

where the dual function $\phi(\lambda)$ is defined as

$$\begin{aligned} \phi(\lambda) = \min_{dy \in Y - y(k),\, s} & \sum_{i=0}^{N} s_i^T s_i + \frac{1}{\gamma_1} dy_i^T dy_i + \sum_{i=0}^{N-1} \\ & \cdot \left\{ \lambda_{f,\,i} \left[E_i(y(k)) + dx_{i+1} - dx_i - \frac{t_f}{N} f_{y_i}(k) dy_i + s_i \right] \right. \\ & \left. + \lambda_{a',\,i} [a'u_i(k) + b' + z_i(k) + a'du_i + dz_i] \right\} \\ & + \lambda_{f,\,N} [E_N(y(k)) + dx_N + s_N] \end{aligned} \tag{14}$$

in which $\lambda = (\lambda_{f,\,1},\, \lambda_{a',\,1}, \cdots, \lambda_{f,\,N})$ is the vector of the Lagrange multiplier and each $\lambda_{f,\,i} \in \Re^n$, $i = 0, 1, \cdots, N$, $\lambda_{a',\,i} \in \Re^r$, $i = 0, 1, \cdots, N-1$.

Let λ^* be the optimal solution of the dual problem. Then the solution of the right-hand side of problem (14) with $\lambda = \lambda^*$ is the solution of (12), $(dy^*,\, ds^*)$. Furthermore $\sum_{i=0}^{N} s_i^{*T} s_i^* + \frac{1}{\gamma_1} dy_i^{*T} dy_i^* = \phi(\lambda^*)$. We use the dual Jacobi method with the following iteration to solve (13)

$$\lambda(j+1) = \lambda(j) + \gamma_2 (\mathrm{diag}\, [\nabla^2 \phi^u(\lambda(j))])^{-1} \nabla \phi(\lambda(j)) \tag{15}$$

where $\phi^u(\lambda)$ denotes the unconstrained dual function, that is, the minimization problem on the right-hand side of (14) with the constraint $dy \in Y - y(k)$ neglected; $\mathrm{diag}\, [\nabla^2 \phi^u(\lambda(j)]$ is a negative definite diagonal matrix formed from the diagonal terms of $\nabla^2 \phi^u(\lambda(j))$;

$\nabla\phi(\lambda(j))$, which is a function of $\hat{dy}$ and $\hat{s}$, is the gradient of ϕ with respect to λ at $\lambda(j)$; and $\hat{dy}$ and $\hat{s}$ represent the solution of the constrained minimization problem on the right-hand side of (14) with $\lambda = \lambda(j)$.

The formulas for calculating the values of each component of $\hat{dx}_i$, $\hat{du}_i$, $\hat{dz}_i$, and $\hat{s}_i$, $i = 1, \cdots, N$, and the values of each component of $\nabla_{\lambda_{f,i}}\phi(\lambda(j))$ and $\nabla_{\lambda_{a',i}}\phi(\lambda(j))$ are given in [6] and also appear in [7]. The formulas for the diagonal terms of $\nabla^2\phi^u(\lambda(j))$ are given in [7].

3) Convergence of the Phase 1 Method: The convergence of the phase 1 method is stated in the following proposition (the proof of the proposition is given in the Appendix).

Proposition 1: Let the point $(\hat{t}_f, \hat{x}, \hat{u}, \hat{s})$ be an optimal solution of (7) with zero objective value and let $t_f(\bar{l})$ satisfy $t_f(\bar{l}) - \delta t < \hat{t}_f \leq t_f(\bar{l})$, where $\bar{l}$ is a positive integer. Suppose δt, γ_1, and γ_2, in (8), (11), and (15), respectively, are small enough. Then i) the two-level method will not diverge in any iteration, and ii) if the initial point $y(0)$ in (11) is close enough to $\hat{y} = (\hat{x}, \hat{u}, \hat{z})$, when $t_f(l)$ in (9) equals $t_f(\bar{l})$, then the two-level method will terminate in at most $\bar{l}$ iterations.

To speed up this two-level method, the Newton method-based initial value processing used in [6] can also be used here to obtain a $y(0)$ that is close enough to $\hat{y}$ under a feasible horizon $t_f(\bar{l})$. This initial-value processing starts from a small t_f and uses the iterations $t_f(l+1) := t_f(l) - [\sum_{i=0}^{N} \hat{s}_i^T(t_f(l))\hat{s}_i(t_f(l))]/[(\frac{d}{dt_f}\{\sum_{i=0}^{N} \hat{s}_i^T(t_f(l))\hat{s}_i(t_f(l))\}]$ for the master problem, where the value of $(\frac{d}{dt_f})\{\sum_{i=0}^{N} \hat{s}_i^T(t_f(l))\hat{s}_i(t_f(l))\}$ can be obtained from the solution of slave problem (9) with $t_f = t_f(l)$ [6]. We will stop the initial-value processing when $\sum_{i=0}^{N} \hat{s}_i^T(t_f(l))\hat{s}_i(t_f(l)) < \epsilon_2(\epsilon_2 > \epsilon_1)$, and the corresponding solution y of the slave problem will be taken as $y(0)$.

As pointed out in Section II, we do not require the iterative method to iterate forever to meet the exact terminal constraint, so finite accuracy can be used to check for the convergence of the iterative method. Thus, we can use $|\nabla\phi(\lambda(j))|_\infty < \epsilon_3$ and $|y(k+1) - y(k)|_\infty < \epsilon_4$ as the convergence criteria for (15) and (11), respectively. The details of the algorithm steps of the phase 1 method can be found in [7]. We will not restate the algorithm here because of the limitation on the length of this note.

B. Method for Solving the Phase 2 Problem

Once the control and horizon pair $[\tilde{u}, \tilde{t}_f]$ is obtained from the phase 1 method, the phase 2 problem (6) is well defined. To solve (6) approximately, we start from the $\tilde{u}(i)$, $\tilde{z}(i)$, and $\tilde{x}(i)$, $i = 0, \cdots, N$, obtained from the solution of the phase 1 problem, and then use the scaled gradient projection method to improve the solution gradually in the sense of reducing the performance index of (6). We need to discretize (6) first. If we scale the objective function by $(N/\tilde{t}_f)$, the discretized problem is as shown in (16)

$$\begin{aligned}\min_y \Bigg\{ &\sum_{i=0}^{N} \frac{1}{2}[x_i^T Q x_i + u_i^T R u_i] \Big| x_{i+1} - x_i - \frac{\tilde{t}_f}{N} f(y_i, \tilde{t}_f) = 0,\\ &x_0 = x(t), \quad a' u_i + b' + z_i = 0,\\ &\underline{y_i} \leq y_i \leq \overline{y}_i, \quad i = 0, 1, \cdots, N-1, \quad x_N = 0 \Bigg\}. \end{aligned} \tag{16}$$

Let the feasible solution set of nonlinear constraints for (16) be defined as $\mathcal{X} = \{y \in \Re^{(n+p+r)N} | x_{i+1} - x_i - \frac{\tilde{t}_f}{N} f(y_i(k), \tilde{t}_f) = 0,\ a' u_i(k) + b' + z_i(k) = 0,\ \underline{y_i} \leq y_i \leq \overline{y}_i,\ i = 0, \cdots, N-1,\ x_N = 0\}$. Then starting from a feasible solution $\tilde{y}_i = (\tilde{x}_i, \tilde{u}_i, \tilde{z}_i)$, $i = 0, \cdots, N$, obtained from the phase 1 method, we may describe the scaled gradient projection method for (16) as

$$\begin{aligned} y(k+1) = y(k) + \arg\Bigg\{ &\min_{dy \in \mathcal{X} - y(k)} \sum_{i=0}^{N} \frac{1}{2\gamma_3}\\ &\cdot [dx_i^T \hat{Q}\, dx_i + du_i^T \hat{R}\, du_i + \eta\, dz_i^T\, dz_i]\\ &+ [Q x_i(k)]^T dx_i + [R u_i(k)]^T du_i \Bigg\} \end{aligned} \tag{17}$$

where $\mathcal{X} - y(k) \equiv \{dy \in \Re^{(n+p+r)N} | y(k) + dy \in \mathcal{X}\}$, $\hat{Q}$ and $\hat{R}$ denote the diagonal matrices formed from the diagonal terms of Q and R, respectively, and η is a small positive real number. The constrained minimization problem inside the big brackets in (17), however, is not easy to solve because the constraints in $\mathcal{X} - y(k)$ are nonlinear, making projection onto $\mathcal{X}$ very difficult. Yet, Miele *et al.* [4] and Mukai and Polak [10] have shown that such a nonlinear constrained minimization problem can be solved by projection onto the tangent plane and restoration back to nonlinear constraints. Their methods are not suitable for VLSI implementation. Based on their framework of projection and restoration, however, we can develop VLSI implementable methods for the problem.

1) The Projection Problem: The tangent plane of $\mathcal{X}$ at $y(k)$ is formed by linearizing the constraints in (16) at point $y(k)$. Thus, in the projection stage, we will solve

$$\begin{aligned}\min_{dy} \Bigg\{ &\sum_{i=0}^{N} \frac{1}{2\gamma_3}[dx_i^T \hat{Q} dx_i + du_i^T \hat{R} du_i + \eta dz_i^T dz_i]\\ &+ [Q x_i(k)]^T dx_i + [R u_i(k)]^T du_i | E_i(y(k))\\ &+ dx_{i+1} - dx_i - \frac{\tilde{t}_f}{N} f_{y_i}(k) dy_i = 0, \quad dx_0 = 0,\\ &\quad a' u_i(k) + b' + z_i(k) + a' du_i + dz_i = 0,\\ &\quad \underline{E}_N(y(k)) + dx_N = 0, \quad y_i \leq y_i(k) + dy \leq \overline{y}_i,\\ &\quad i = 0, 1, \cdots, N \Bigg\}. \end{aligned} \tag{18}$$

Let $dy^* = (dx^*, du^*, dz^*)$ be the optimal solution of (18). Then the point

$$y'(k) = y(k) + dy^* \tag{19}$$

defines the resulting projection point in the tangent plane of $\mathcal{X}$ at $y(k)$.

2) The Restoration Problem: For the sake of implementing the method in VLSI array processors, we have slightly modified the restoration method in [10]. We use the following iterations to restore $y'(k)$ to $\mathcal{X}$

$$x(m+1) = x(m) + \gamma_4 dx^* \tag{20}$$

with $x(0) = x'(k)$ and dx^* being the solution of the following problem of approximate restoration direction

$$\begin{aligned}\min_{dx,\, s} \Bigg\{ &\sum_{i=0}^{N} \mathcal{M} s_i^T s_i + dx_i^T dx_i | E_i(x(m)) + dx_{i+1} - dx_i\\ &- \frac{\tilde{t}_f}{N} f_{x_i}(x_i(m), u_i'(k), \tilde{t}_f) dx_i + s_i = 0,\\ &\quad i = 0, \cdots, N-1, \quad E_N(x(m)) + dx_N + s_N = 0 \Bigg\} \end{aligned} \tag{21}$$

where $\mathcal{M}$ is a very large positive scalar and s_i, $i = 0, \cdots, N$, are slack variables. Note that the control variables resulting from the projection stage have satisfied the inequality constraints and can be held fixed in the restoration stage; this property is due to Assumption A0).

Remark 2: If all the terms in (21) involving slack variables are deleted, problem (21) becomes the exact restoration problem solved in [10]. That exact restoration problem, however, is not guaranteed to have a solution; if it does not, a descent direction $dx^* = -\nabla[E^T(x(m))E(x(m))]$ is used [10]. The solution of (21) always exists and this solution is a descent direction for minimizing $E^T(x)E(x)$. Thus, the formulation of (21) is simple and eliminates the need to determine whether a solution exists.

3) The Dual Jacobi Method for Solving the Projection and Restoration Problems: Since problem (18) is similar to and problem (21) is simpler than the slave problem (9), they can be solved using a dual Jacobi method that is suitable for VLSI implementation. Let $\psi(\lambda)$ and $\Gamma(\lambda_f)$ denote the dual function of (18) and (21), respectively. Note that since (23) involves state equations only, $\Gamma(\lambda_f)$ is a function of λ_f only, in contrast to $\psi(\lambda)$, which is a function of λ, which in turn consists of λ_f and $\lambda_{a'}$. Let $(\check{d}x, \check{d}u, \check{d}z)$ and $(\bar{d}x, \bar{s})$ denote the terms required for calculating $\nabla\psi(\lambda)$ and $\nabla\Gamma(\lambda_f)$, respectively, as $(\hat{d}x, \hat{d}u, \hat{d}z, \hat{s})$ is needed in calculating $\nabla\phi(\lambda)$. The formulas for $(\check{d}_x, \check{d}u, \check{d}x)$, $(\bar{d}x, \bar{s})$, $\nabla\psi(\lambda)$, $\nabla\Gamma(\lambda_f)$, $\nabla^2\psi^u(\lambda)$, and $\nabla^2\Gamma(\lambda_f)$ can easily be derived. These formulas are not shown here because of space limitations. Then the following iterations of the dual Jacobi method can be used to solve (18) and (21), respectively

$$\lambda(j+1) = \lambda(j) + \gamma_5[\text{diag}\,\nabla^2\psi^u(\lambda(j))]^{-1}\nabla\psi(\lambda(j)) \tag{22}$$

$$\lambda_f(j+1) = \lambda_f(j) + \gamma_6[\text{diag}\,\nabla^2\Gamma(\lambda_f(j))]^{-1}\nabla\Gamma(\lambda_f(j)). \tag{23}$$

4) The Phase 2 Method and Its Convergence: It was pointed out above that if our methods for the projection and restoration problems converge, our methods will solve the constrained minimization problem inside the big brackets in (17). The convergence of our methods for the projection and restoration problems is trivial, as explained below.

The iterations of (20) can be viewed as a scaled gradient method for minimizing $E^T(x)E(x)$. Since $E^T(x)E(x)$ is at least locally convex, with $E^T(x)E(x) = 0$ if $x \in \mathcal{X}$, it can easily be shown using a procedure similar to that given in [5], that if γ_4 in (20) is small enough and satisfies $0 < \gamma_4 < (2K_4/\mathcal{K}(iv))$ and $y'(k)$ is close enough to $\mathcal{X}$, then the sequence generated by (22) with dx^* obtained from solving (23) will converge to a point in $\mathcal{X}$, where $\mathcal{K}^{(iv)} > 0$ is a Lipschitz constant such that $\|\nabla[E^T(x')E(x')] - \nabla[E^T(x'')E(x'')]\|_2 \leq \mathcal{K}^{(iv)}\|x' - x''\|_2$, $\forall x', x'' \in \Re^{n(N+1)}$, and K_4 satisfies the condition that $[\mathcal{M}\nabla E^T(x)\nabla E(x) + I - K_4 I]$ is nonnegative definite for every $x \in \Re^{n(N+1)}$.

Furthermore, if γ_5 in (22) and γ_6 in (23) are small enough, the convergence of the dual Jacobi method for solving (18) and (21) can be ensured by a proof similar to the proof of Proposition 1 showing the convergence of the dual Jacobi method when it is used to solve (12).

Then, if γ_3 in the scaled gradient projection method (17) is small enough, (17) will be a descent method. Consequently, the following proposition obviously holds; the proof is omitted.

Proposition 2: Suppose γ_3, γ_4, γ_5, and γ_6 are small enough. Then i) the scaled gradient projection method (17) will not diverge in any iteration, and ii) $\sum_{i=0}^{N}(1/2)[x_i^T(p_{\max})Qx_i(p_{\max}) + u_i^T(p_{\max})Ru_i(p_{\max})] < \sum_{i=0}^{N}(1/2)[\tilde{x}_i^T Q\tilde{x}_i + \tilde{u}_i^T R\tilde{u}_i]$, where $x_i(p_{\max})$, $u_i(p_{\max})$, $i = 0, \cdots, N$, is the solution resulting from method (17) performing $p_{\max}$ iterations, and $\tilde{x}_i$, $\tilde{u}_i$, $i = 0, \cdots, N$, is the solution obtained from the phase 1 method.

From the above proposition, we see that when we iterate (17) by an arbitrarily large number of iterations, $p_{\max}$, we have in fact solved the discretized phase 2 problem (16) approximately.

As indicated in [4] and [10], exact restoration is not necessary in the sequential projection and restoration process. Thus, in practice, the dual Jacobi method and the scaled gradient method (20) need only iterate for a sufficiently large finite number of iterations. We can use $|\nabla\psi(\lambda(j))|_\infty < \epsilon_5$, $|\nabla\Gamma(\lambda(j))|_\infty < \epsilon_6$, $|x(m+1) - x(m)|_\infty < \epsilon_7$ as the convergence criteria for (22), (23), and (20), respectively. The details of the algorithm steps for the phase 2 method can be found in [7].

IV. Hardware Computing Architecture and Time Complexity for the Two-Phase Method

A. A Modification of the Method

For the sake of hardware implementation, we will modify the convergence criteria for the iterative methods in the same manner as in [6]. We will assign arbitrary numbers of iterations $j_{\max}$ for the dual Jacobi method, $k_{\max}$ for the scaled gradient projection method in the phase 1 method, and $k_{\max}^r$ ($k_{\max}^r < k_{\max}$) for the scaled gradient method in the restoration stage of the phase 2 method. Thus, we may replace the convergence criteria described in Subsection III-A3 and III-B4 by checking whether the corresponding iteration index exceeds the assigned number.

B. VLSI Array Processor Architecture and Operations

From Section III, we see that the phase 1 method is almost the same as the two-level algorithm in [6] except for some minor modifications in the master problem and the dual method for solving slave problem. Therefore, the VLSI array processor architecture presented in [6] can be used to realize the phase 1 method with minor modifications in the details of some of the processing elements. Furthermore, in the phase 2 method, the methods for the projection stage (restoration stage) are a combination of the scaled gradient projection (scaled gradient) method and the dual Jacobi method. These methods have exactly the same iterative process as the method in the phase 1 method for solving the slave problem. Therefore, we may use the same VLSI array processor architecture to carry out the phase 2 method, with only some minor modifications in the processing elements that perform the operations of the scaled gradient projection method and dual Jacobi method. The modifications needed can be summarized as follows: First of all, we use three modes—0, 1, and 2—to distinguish the operations of the VLSI array processors in the phase 1 method and the projection stage and restoration stage of the phase 2 method, respectively. These modes can be used i) to control a multiplexer to select, for example, $k_{\max}$ or $k_{\max}^r$ in the processing element that checks the convergence of the scaled gradient projection method and scaled gradient method, and ii) to select a formula, for example, $\hat{d}x$, $\check{d}x$, or $\bar{d}x$, in the corresponding processing element. Since the phase 2 method begins as soon as the phase 1 method is completed and the projection stage and restoration stage in the phase 2 method alternate until convergence, modes 0, 1, and 2 will alternate frequently. To control these mode changes, we will use control logic circuitry design based on the satisfaction or nonsatisfaction of the convergence criteria. Thus, after making minor modifications to some of the processing elements, we may indeed use the VLSI array processor architecture presented in [6] to realize our two-phase parallel computing method.

C. Overall Time Complexity

To calculate the overall time complexity of the two-phase method in VLSI array processors, we need only to derive the time complexity corresponding to the dual Jacobi method in the complete two-phase method, because, as shown in [6], the execution of the dual method is the dominant term in the overall time complexity. Furthermore, the time complexity of the array processing elements should count as only that of one processing element, and the time complexity of the communication links is negligible [6]. Thus, the estimated time complexity for executing the two-phase method on VLSI array processors is

$$[m_s k_{\max} j_{\max} + p_{\max}(j_{\max} + k^r_{\max} j_{\max})][T_1 + T_2 + T_3] \qquad (24)$$

where m_s denotes the actual number of iterations that the phase 1 method takes and $T_1 + T_2 + T_3$ stands for the maximum serial time complexity of the processing elements, corresponding to the execution of one iteration of the dual Jacobi method for (15), (22), and (23). The individual time complexity formulas for T_1, T_2, and T_3 can be derived as follows: $T_1 = 2T_\otimes + \log_2(2n+2)T_\oplus$, $T_2 = 2T_\otimes + \log_2(n+p+3)T_\oplus$, and $T_3 = 1T_\otimes + 1T_\oplus$, where $T_\otimes$ and $T_\oplus$ denote the times for performing one multiplication and one addition, respectively, and n and p denote the number of states and controls, respectively.

V. Simulations

According to the work of Sharma *et al.* [11], $T_\otimes = 6.75$ ns for a 16×16 bit multiplication, and $T_\oplus \le 0.35$ ns for an addition. Then (24) becomes

$$[m_s k_{\max} j_{\max} + p_{\max}(j_{\max} + k^r_{\max} j_{\max})] \cdot [40.85 + 0.35 \log_2((2n+2)(n+p+3))] \text{ ns.} \qquad (25)$$

In the examples given in this section and [7], we use a second-order Runge–Kutta method to discretize the continuous problem. We set $W = \{x \mid \|x\|_\infty \le 0.05\}$, $\Delta t = dt = (t_{f,h}/N)$, $N = 30$, $\delta t = 0.2$ seconds, $k_{\max} = 40$, $k^r_{\max} = 10$, $j_{\max} = 40$, $p_{\max} = 30$, $\epsilon_1 = 0.0001$, $\epsilon_2 = 0.001$, and $\mathcal{M} = 100$. We select conservative values for the step-sizes γ_i, $i = 1, \cdots, 6$, respectively, as follows: 20, 0.2, 0.01, 0.1, 0.9, and 0.9.

Example: Consider the Van der Pol Oscillator described by $\dot{x}^\alpha = (1 - (x^\beta)^2)x^\alpha - x^\beta + u$, $\dot{x}^\beta = x^\alpha$, where x^α and x^β are state variables and u is the scalar control. We intend to find a receding horizon feedback control solution such that the control satisfies the instantaneous bound $|u| \le 0.7$ and drives the system from the initial state (0, 1) at time $t = 0$ to the origin (0, 0) asymptotically.

We use the following arbitrary initial values: $t_f = 3$ seconds, $u_i = 0$, $1 \le i \le N-1$, and $x^\alpha_i = x^\alpha_0 - (i/N)x^\alpha_0$, $x^\beta_i = x^\beta_0 - (i/N)x^\beta_0$, $i = 0, \cdots, N$. We choose $Q = \left[\begin{smallmatrix} 5 & 0 \\ 0 & 5 \end{smallmatrix}\right]$ and $R = 1$ in the phase 2 problem. Our test results show that the first horizon $t_{f,0} = 5.619$ seconds. Because we do not consider any disturbance here, the values of $t_{f,h}$, $h = 1, 2, \cdots$, are all equal to $t_{f,0}$, as expected. The receding horizon feedback control process stops when $h = 36$ because $\max\{|(x^\alpha)^{\hat{u}_{35}}(36\Delta t; x_{35}(t_{35}), t_{35})|, |(x^\beta)^{\hat{u}_{35}}(36\Delta t; x_{35}(t_{35}), t_{35})|\} < 0.05$. Then, we switch the control to a predesigned linear feedback control $\hat{u} = -6x^\alpha - 3x^\beta$. The final two-phase method-based implementable receding horizon feedback control solution $\hat{u}(t)$ and the resulting state trajectory $x^{\hat{u}}(t)$ are shown in Fig. 1. This state trajectory is indeed asymptotically stable. We find the performance index $V_h(\equiv V(x_h, t_h, u_h, t_{f,h}))$ for each $h = 0, 1, \cdots, 36$ is monotonic decreasing, so (2) is satisfied. In this example, $n = 2$, $p = 1$, and $r = 0$. When $h = 0$, we have $m_s = 11$. Then the estimated computation time for the two-phase method calculated from (25) is 1.28 ms. When $h > 0$, since the initial guess for the discretized phase 1 problem is already an admissible control and horizon pair, we have $m_s = 1$ for $h = 1, \cdots, 36$. The estimated computation time for the two-phase method calculated from (25) is 0.62 ms for each $h = 1, 2, \cdots, 36$. These execution times confirm the real-time applicability of our hardware-implementable receding horizon controller. The same conclusion can also be drawn from the example in [7].

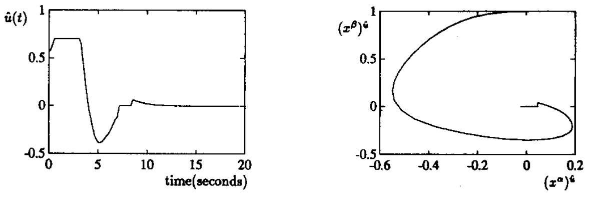

Fig. 1. The final feedback control solution and state trajectory of the example.

VI. Appendix

Proof of Proposition 1:

1) Proving i) is equivalent to proving that the combination of method (11) with method (15) converges. Since method (15) is a special version of the scaled gradient method and the dual function is continuously differentiable and bounded from above, then from the result in [5] method (15) converges if γ_2 is small enough; in particular, the value range of γ_2 is $0 < \gamma_2 < \underline{K}_2/\overline{\mathcal{K}}''$, where $\overline{\mathcal{K}}'' = \max_k \mathcal{K}''(k)$, and $\underline{K}_2 = \min_k K_2(k)$, in which k denotes the iteration index of method (11), $\mathcal{K}''(k)$ is the Lipschitz constant of the dual function $\phi(\lambda)$ such that $\|\nabla\phi(\lambda_1) - \nabla\phi(\lambda_2)\|_2 \le \mathcal{K}''(k)\|\lambda_1 - \lambda_2\|_2, \forall \lambda_1, \lambda_2 \in \mathfrak{R}^{n(N+1)+rN}$, and $K_2(k) = \min_j [\min_{\nu, \nu', (f, i), (a', i)} \{(\partial^2/\partial\lambda^{\nu^2}_{f,i})\phi^u(\lambda(j)), (\partial^2/\partial\lambda^{\nu'2}_{a',i})\phi^u(\lambda(j))\}]$ is a positive constant such that $\operatorname{diag}\nabla^2\phi^u(\lambda) + K_2(k)I \le 0$ for all iterations j of method (15). Because the dual function $\phi(\lambda)$ is quadratic and strictly concave, the limit point of the sequence $\{\lambda(j)\}$ generated by (15) will be λ^*, the optimal solution of (13). Consequently, the (dy^*, ds^*) corresponding to λ^* in (14) is the solution of (12). Thus, we may proceed to prove that method (11) converges. Method (11) is a special version of the scaled gradient projection method. Since the objective function of (10) is bounded from below and is continuously differentiable, then according to the result in [5], (11) will converge if γ_1 is small enough; in particular, $0 < \gamma_1 < 4/\mathcal{K}'$, where $\mathcal{K}'(> 0)$ is the Lipschitz constant such that

$$\left\| \nabla\left[\sum_{i=0}^{N} E_i^T(y')E_i(y')\right] - \nabla\left[\sum_{i=0}^{N} E_i^T(y'')E_i(y'')\right] \right\|_2 \le \mathcal{K}'\|y' - y''\|_2, \quad \forall y', y'' \in Y \cap U$$

where $Y = \{y \mid \underline{y_i} \le y_i \le \overline{y_i}, i = 0, \cdots, N\}$, $U = \{(u, z) \mid a'u_i + b' + z_i = 0,\ z_i \ge 0,\ i = 0, 1, \cdots, N\}$.

2) By assumption, $\sum_{i=0}^N \hat{s}_i^T \hat{s}_i = 0$. Since $t_f(\bar{l}) - \delta t < \hat{t}_f \le t_f(\bar{l})$ and δt is very small, there must exist a point $(\overline{x}, \overline{u}, \overline{z})$ near $(\hat{x}, \hat{u}, \hat{z})$ such that $(\overline{x}, \overline{u}, \overline{z})$ associated with $t_f(\bar{l})$ is optimal for (7) and $\sum_{i=0}^N \overline{s}_i^T \overline{s}_i = 0$ because of the property that $f^0(0, 0) = 0$. Clearly, the function $\sum_{i=0}^N E_i^T(y)E_i(y)$ is locally convex and has a local minimum at $y = \overline{y}$, because $\sum_{i=0}^N \overline{s}_i^T \overline{s}_i = 0$ and $\overline{s} = E(\overline{y})$. Now since

   a) $y(0)$ is close enough to $\overline{y} = (\overline{x}, \overline{u}, \overline{s})$, because by assumption, $y(0)$ is close enough to $\hat{y} = (\hat{x}, \hat{u}, \hat{z})$;

b) (11) with small enough γ_1 is a descent method [5] so that $\sum_{i=0}^{N} E_i^T(y(k))E_i(y(k))$, $k = 1, 2, \cdots$ is a decreasing sequence; and

c) $\sum_{i=0}^{N} E_i^T(y)E_i(y)$ is locally convex at point $\overline{y}$ and $\sum_{i=0}^{N} E_i^T(\overline{y})E_i(\overline{y}) = 0$,

then the stopping criteria of (8) should be satisfied at some iteration k of (11) when $t_f = t_f(\overline{l})$. This proves ii).

Acknowledgment

The author wishes to thank Prof. T. S. Chang for several helpful discussions.

References

[1] D. Q. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," *IEEE Trans. Automat. Contr.*, vol. 35, no. 7, pp. 814–824, July 1990.
[2] ——, "An implementable receding horizon controller for stabilization of nonlinear systems," in *Proc. 29th IEEE Conf. Dec. Contr.*, Dec. 1990, pp. 3396–3397.
[3] ——, "Robust receding horizon control," in *Proc. 30th IEEE Conf. Dec. Contr.*, 1991, pp. 64–69.
[4] A. Miele, J. Damoulakis, J. Cloutier, and J. Tietze, "Sequential gradient-restoration algorithm for optimal control problems with nondifferential constraints," *JOTA*, vol. 13, no. 2, pp. 218–255, 1974.
[5] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation.* London: Prentice-Hall, 1989.
[6] S.-Y. Lin, "A hardware implementable two-level parallel computing algorithm for general minimum-time control," *IEEE Trans. Automat. Contr.*, vol. 37, no. 5, pp. 589–603, May 1992.
[7] ——, "A two-phase parallel computing algorithm for the solution of implementable receding horizon control for constrained nonlinear systems," in *Proc. 32nd IEEE Dec. Contr.*, 1993.
[8] M. Bazaraa and C. Shetty, *Nonlinear Programming.* New York: Wiley, 1979.
[9] D. Luenberger, *Linear and Nonlinear Programming*, 2nd ed. Reading, MA: Addison-Wesley, 1984.
[10] H. Mukai and E. Polak, "On the use of approximations in algorithms for optimization problems with equality and inequality constraints," *SIAM J. Numer. Anal.*, vol. 15, pp. 674–693, 1978.
[11] R. Sharma, A. Lopaz, J. Michejda, S. Hillenius, J. Andrews, and A. Studwell, "A 6.75-ns 16 × 16-bits multiplier in single level method CMOS technology," *IEEE J. Solid State Circuits*, Aug. 1989.
[12] S. Y. Kung, *VLSI Array Processors.* Englewood Cliffs, NJ: Prentice-Hall, 1988.

Characterization and Computation of the Solution to the Optimal L_∞ Approximation Problem

Davut Kavranoğlu and Maamar Bettayeb

Abstract— **The characterization of the solution to the problem of approximating a given stable, proper, rational transfer function, in L_∞ norm sense, by a rational function with prescribed number of stable and unstable modes is developed. A simple state-space suboptimal computational algorithm for the solution is presented.**

I. Notation

L_∞ (H_∞)	the space of functions of a complex variable that are (analytic in the right half plane and) bounded on the $j\omega$-axis.
R (prefix)	restricted to be rational (RH_∞, etc.).
$\left[\begin{array}{c\|c} A & B \\ \hline C & D \end{array}\right]$	$C(sI - A)^{-1}B + D$.
$\\|G\\|_\infty$	$\sup_{\omega \in R} \sigma_{\max}(G(j\omega))$.
$G^\sim(s)$	$G^T(-s)$.
$B_\gamma H_\infty$	$\{G \mid G \in H_\infty,\ \\|G\\|_\infty \le \gamma\}$.
$G_{(n_1, n_2)}(s)$	$G(s) \in L_\infty$ with n_1 stable modes and n_2 unstable modes.
$In(A) = (\pi, \nu, \delta)$	$\pi = \pi(A)$, $\nu = \nu(A)$, $\delta = \delta(A)$ are respectively the numbers of eigenvalues of A having positive, negative, and zero real parts.

II. Introduction

The pioneering work of Adamjan *et al.* [1] on infinite dimensional operator approximation, specialized to dynamic linear systems of finite dimension, led to concrete closed form optimal model reduction algorithms [2]–[4]. These model reduction algorithms give the optimal solution in the Hankel norm sense for the approximation of a stable transfer function with McMillan degree n by another transfer function with r $(r < n)$ stable poles (with possible unstable modes). The same tools also led to an elegant solution to the H_∞ optimization problem [5].

The problem treated in this paper is new and recovers some known problems as special cases and is a step towards the general problem of approximating an unstable transfer function by another unstable transfer function with prescribed number of stable and unstable modes. A preliminary related work is reported in [6]. The L_∞ approximation problem treated in this paper is stated as follows.

Definition 1—L_∞ Approximation Problem: Given a $p \times m$ transfer function $G_n(s) \in RH_\infty$ with McMillan degree n, find a $p \times m$ transfer function $G_{(n_1, n_2)}(s) \in RL_\infty$ with at most n_1 stable and at

Manuscript received December 1, 1992; revised October 18, 1993. This work was supported by King Fahd University of Petroleum and Minerals Project SE/DISCRETE/142.

D. Kavranoglü is with the Department of Systems Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia.

M. Bettayeb is with the Department of Electrical Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia.

IEEE Log Number 9402737.