8. Model Predictive Control Using Volterra Series

8.1 Introduction

The preceding chapters have addressed the synthesis of controllers from Volterra series models. In Chapter 6, the concept of a partitioned inverse was introduced in the IMC framework to allow the explicit use of a Volterra series model in the controller design. Extensions were sketched for problematic dynamics, such as nonminimum phase behavior, although the derivations were carried out for the single-input-single-output (SISO) problem. In Chapter 7, a procedure was introduced to compensate for constraints, using classical techniques from constrained linear control theory. In the most general case, where multivariable interactions, problematic dynamics, and constraints are present, an MPC approach is the natural choice. As indicated in Chapter 6, there are natural extensions of the IMC algorithm to the MPC framework.

MPC involves the calculation of a set of manipulated variable moves that minimize an open-loop performance objective over a prediction horizon, subject to constraints on manipulated and controlled variables. Most MPC applications are based on linear finite convolution models, such as the step and impulse response models. An attractive feature of these models is the relative ease with which they can be obtained from plant data.

Many researchers have proposed predictive control schemes based on the direct use of nonlinear models. In nonlinear quadratic dynamic matrix control (García, 1984), a nonlinear model is used to compute the effect of past manipulated variables on the predicted output. A linear model, obtained by linearizing the nonlinear model at each sampling time, is used to compute the manipulated variable values. An advantage of this approach is that only one quadratic program is solved at each sampling interval versus solving a higher-order nonlinear program. Gattu and Zafiriou (1992) extended this algorithm by incorporating state estimation using a steady-state Kalman filter gain computed at each sampling time. Lee and Ricker (1993) modified García's algorithm by implementing an extended Kalman filter and by incorporating load disturbances on the states as additive stochastic signals. In Gattu and Zafiriou (1992), the Kalman filter was designed with white state noise characterized by a scalar-times-identity covariance matrix, which can lead to significant bias in the state estimates. Ricker and Lee (1995) implemented

their algorithm in an 8×8 nonlinear MPC scheme for the Tennessee Eastman test problem.

Li and Biegler (1988) extended the nonlinear internal model control design procedure (Economou and Morari, 1985) to a single-step method that included constraints on the states and inputs. They extended their algorithm to a multi-step strategy that linearizes a nonlinear model around a nominal trajectory (Li and Biegler, 1989).

Both the nonlinear quadratic dynamic matrix control (García, 1984; Gattu and Zafiriou, 1992; Lee and Ricker, 1993) and modified NLIMC (Li and Biegler, 1988; Li and Biegler, 1989) algorithms require a fundamental model of the process. Deriving these models can be very time consuming, and can be elusive if the process is not well understood. This motivates the use of the class of nonlinear models that have been discussed extensively in this book.

In this chapter, we develop the basic nonlinear MPC problem, with special emphasis on the use of Volterra series models. A variety of formulations will be described, as well as the numerical methods that can be employed to solve efficiently the resultant programming problem. Finally, a review of applications of Volterra series MPC will be presented.

8.2 General nonlinear MPC problem

The general nonlinear MPC problem involves the calculation of a sequence of future control moves that minimize a given objective function, subject to the constraint that the system evolves according to a nonlinear dynamic model. Mathematically, one can formulate this as follows:

$$\min_{\mathbf{U}} \quad \sum_{i=1}^{p} (\mathbf{y}_{ref}(k+i) - \mathbf{y}(k+i))^T \Gamma_Y (\mathbf{y}_{ref}(k+i) - \mathbf{y}(k+i)) + \sum_{j=1}^{M} \Delta\mathbf{u}(k+j-1) \Gamma_U \Delta\mathbf{u}(k+j-1)$$

subject to:

$$\begin{aligned} \mathbf{x}(k+i) &= f(\mathbf{x}(k), \mathbf{u}(k)) \\ \mathbf{y}(k) &= h(\mathbf{x}(k)) \\ \mathbf{U} &= [\mathbf{u}(k) \quad \mathbf{u}(k+1) \quad \ldots \quad \mathbf{u}(k+M-1)]^T \\ \mathbf{U}_{low} &\leq \mathbf{U} \leq \mathbf{U}_{high} \\ \Delta\mathbf{U}_{low} &\leq \Delta\mathbf{U} \leq \Delta\mathbf{U}_{high} \end{aligned} \tag{8.1}$$

where the last two lines denote the constraints on the actuator consisting of magnitude and rate constraints respectively. For simplicity, we have avoided

incorporating output constraints, which can give rise to infeasible programming problems. m and p are the lengths of the prediction horizon and the future move horizon respectively. Γ_Y and Γ_U are the weightings applied to the setpoint tracking and move suppression terms respectively in the overall objective function. These four quantities $(m, p, \Gamma_Y, \Gamma_U)$ are used to tune the model predictive controller.

In this formulation, we adopt the standard 2-norm metric for the objective function, and the nonlinear nature of the controller arises from the system model denoted by f and h. Of course, if one were to adopt a linear model, then a quadratic program would result from the standard linear MPC.

The general problem stated above is solved at a given time instant, and only the first elements of the control move vector are implemented. New measurements are then collected, the algorithm is "reset," and the procedure is repeated. This scheme allows for a number of variations, which can be characterized broadly by the following four elements of the basic algorithm (Ogunnaike and Ray, 1994):

- the use of a filter for the setpoint signal
- the formulation of the model for prediction of future outputs
- the calculation of the sequence of future moves from the programming problem in Equation (8.1)
- the update mechanism used to reconcile model predictions against actual measurements.

In this chapter, we will review a number of formulations of the Volterra series MPC problem, which differ primarily in the handling of one or more of these components of the algorithm. For ease of presentation, the algorithms will be presented for the SISO case.

8.3 Model predictive controller formulations for Volterra models

8.3.1 Standard impulse response form

The form of the Volterra series model that has been used in the previous chapters is in the so-called impulse-response form. The corresponding MPC formulation for the linear case is typically referred to as model algorithmic control (MAC) (Mehra et al., 1978; Rouhani and Mehra, 1982; Ogunnaike and Ray, 1994). If one considers a second-order Volterra model, then the following MPC formulation results:

$$\min_{\mathbf{U}} \quad \sum_{i=1}^{p} (y_*(k+i) - y(k+i))^T \Gamma_Y (y_*(k+i) - y(k+i)) + \sum_{j=1}^{M} \Delta u(k+j-1) \Gamma_U \Delta u(k+j-1)$$

subject to:

$$y(k+i) = \sum_{\ell=1}^{N} h_1(\ell)u(k+i-\ell) + \sum_{\ell=1}^{N}\sum_{m=\ell}^{N} h_2(\ell,m)u(k+i-\ell)u(k+i-m)$$

$$\mathbf{U} = [u(k) \;\; u(k+1) \;\; \ldots \;\; u(k+M-1)]^T$$
$$\mathbf{U}_{low} \leq \mathbf{U} \leq \mathbf{U}_{high}$$
$$\Delta\mathbf{U}_{low} \leq \Delta\mathbf{U} \leq \Delta\mathbf{U}_{high} \tag{8.2}$$

where the variable y_* denotes the filtered value of the setpoint, which is determined by the following expression (Harris, 1985):

$$y_*(k+1) = \alpha y_*(k) + (1-\alpha y_{ref}(k))$$

A number of authors have pointed out that, under certain conditions, there are analytical or simple numerical solutions to this problem. For example, in Haber (1989, 1995) a solution is presented for the unconstrained case in which the fourth-order objective function yields a cubic equation to be solved for the optimal control move. In the case of multiple feasible solutions, the authors recommend that the value of the control move that is closest to the present position should be selected.

There are a number of extensions to the basic MPC problem described above, including state estimation and filtering, which will be discussed in Section 8.3.4.

8.3.2 Dynamic matrix control

One of the most studied forms of linear model predictive control is the so-called step response formulation, also commonly known as dynamic matrix control (DMC). In this formulation, the model is formulated in the incremental change in the manipulated variable Δu. Among other advantages, it is more straightforward to design a controller with integral tracking in this formulation (Ogunnaike and Ray, 1994).

Haber (1995) proposed an extension of the step-response model for Volterra series using a so-called "incremental form." One can redefine the absolute control moves as follows:

$$u(k+i) = \Delta u(k+i) + \Delta u(k+i-1) + \ldots + \Delta u(k+1) + \Delta u(k) + u(k-1).$$

The predictive form of the Volterra series model becomes:

$$y(k+p) = \sum_{i=0}^{p+1} s_1(i)\Delta u^*(k+p-i) + \sum_{i=0}^{p+1}\sum_{j=i}^{p+1} s_2(i,j)\Delta u^*(k+p-i)\Delta u^*(k+p-j),$$

where

$$\Delta^* u(k+i) = \begin{cases} \Delta u(k+i) & 0 \le i \le p \\ u(k+i) & i < 0 \end{cases}$$

$$s_1(i) = \sum_{\nu=0}^{\min(i,N)} h(\nu)$$

$$s_2(i,j) = \sum_{\nu=0}^{\min(i,N)} \sum_{\mu=\nu}^{\min(j,N)} h(\nu,\mu) \quad i = 0,1,2,\ldots,p+1,$$

and N is the Volterra series model memory.

8.3.3 Generalized predictive control

In the linear generalized predictive control (GPC) formulation, a controlled autoregressive integrated moving average (CARIMA) input-output model is adopted (Clark et al., 1987a,b). Sommer (1994) proposes a nonlinear extension for Volterra, bilinear, and Hammerstein models. The AR-Volterra model representation is given as:

$$A(q^{-1})y(t) = q + 0 + B_1(q^{-1})u(t) + B_2(q_1^{-1}, q_2^{-1})u(t)u(t)$$

where A, B_1 and B_2 are polynomials in the delay operator q^{-1}:

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \ldots a_{N_y} q^{-n_a}$$
$$B_1(q^{-1}) = h_1(1)q^{-1} + h_1(2)q^{-2} + \ldots h_1(N_u)q^{-n_a}$$

and B_2 is defined as:

$$B_2(q_1^{-1}, q_2^{-1})u(t)u(t) = \sum_{i=1}^{N_u}\sum_{j=o}^{N_u} h_2(i,j)u(t-i)u(t-j).$$

Using the recursive Diophantine equation (Sommer, 1994), one can derive a predictive form for the Volterra model as follows:

$$y^*(k+p) = \alpha(q^{-1})y(k) + \alpha_0 + B_1(q^{-1})u(k+p) + B_2(q_1^{-1}, q_2^{-1})u(k+p)u(k+p).$$

8.3.4 State-space MPC

In this section, we consider the most general formulation of the Volterra series MPC problem, and allow for the general AR-Volterra model that was discussed in Chapter 3.

In formulating a state-space version of a Volterra-based MPC problem, one can define as the state vector the sequence of past inputs and outputs to the system:

$$\begin{aligned} x_1^a(k) &= y(k) & x_1^b(k) &= u(k-1) \\ &\vdots & &\vdots \\ x_{n_y+1}^a(k) &= y(k-n_y) & x_{n_u}^b(k) &= u(k-n_u) \end{aligned}$$

or

$$\underline{x}^a(k) = [x_1^a(k), \ldots, x_{n_y+1}^a(k)]^T \quad \underline{x}^b(k) = [x_1^b(k), \ldots, x_{n_u}^b(k)]^T$$

where $\underline{x}^a(k)$ and $\underline{x}^b(k)$ correspond to the output and input parts of the state respectively. The complete state is given by:

$$\underline{x}(k) = \begin{bmatrix} \underline{x}^a(k) \\ \underline{x}^b(k) \end{bmatrix}$$

The nonminimal realization becomes:

$$\begin{bmatrix} \underline{x}^a(k+1) \\ \underline{x}^b(k+1) \end{bmatrix} = \begin{bmatrix} 0 \cdots 0 & 0 & & \\ 1 & 0 & & \\ \ddots & \vdots & & \\ & 1 & 0 & \\ & 0 & 0 \cdots 0 & 0 \\ & & 1 & 0 \\ & & \ddots & \vdots \\ & & & 1 \; 0 \end{bmatrix} \begin{bmatrix} \underline{x}^a(k) \\ \underline{x}^b(k) \end{bmatrix} + \begin{bmatrix} f(\underline{x}(k), u(k)) \\ 0 \\ \vdots \\ 0 \\ u(k) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$y(k) = [1 \;\; 0 \;\; \cdots \;\; 0 \,|\, 0 \;\; \cdots \;\; 0] \begin{bmatrix} \underline{x}^a(k) \\ \underline{x}^b(k) \end{bmatrix}$$

or

$$\begin{aligned} \underline{x}(k+1) &= F(\underline{x}(k), u(k)) \\ y(k) &= h(\underline{x}(k)), \end{aligned}$$

where $f(\underline{x}(k), u(k))$ is the relevant form of the Volterra series model (e.g. AR-Volterra). State estimation can be incorporated into the MPC scheme using Equation (8.3):

$$\begin{aligned} [\hat{\underline{x}}(k+1|k)] &= [f(\hat{\underline{x}}(k|k), u(k))] \\ [\hat{\underline{x}}(k|k)] &= \left[\hat{\underline{x}}(k|k-1) + L\hat{d}(k|k)\right] \\ \hat{y}(k|k) &= h(\hat{\underline{x}}(k|k)), \end{aligned} \tag{8.3}$$

where the notation $(k+1|k)$ denotes the estimate at future sampling period $k+1$ based on information available at period k. L is the estimator gain and $\hat{d}(k|k)$ is the difference between the plant measurement and model prediction, i.e.

$$\hat{d}(k|k) = \tilde{y}(k) - \hat{y}(k|k-1).$$

Assuming an AR-Volterra model, the control action is computed by solving the following nonlinear programming problem:

$$\min_{\mathbf{U}} \quad \sum_{i=1}^{p}((y^*(k+i) - y(k+i))\Gamma_Y(y^*(k+i) - y(k+i)) + \sum_{j=1}^{m}\Gamma_U(\Delta u(k+j-1))^2$$

subject to:

$$y(k) = \sum_{j=1}^{N_y}\theta_j^1 y(k-j) + \sum_{j=1}^{N_u}h_1(j)u(k-j) + \sum_{j=1}^{N_u}\sum_{n=1}^{j}h_2(j,n)u(k-j)u(k-n) + \sum_{j=1}^{N_u}\sum_{n=1}^{j}\sum_{r=1}^{n}h_3(j,n,r)u(k-j)u(k-n)u(k-r)$$

$$\mathbf{U} = [u(k) \quad u(k+1) \quad \ldots \quad u(k+m-1)]^T$$
$$\mathbf{U}_{low} \leq \mathbf{U} \leq \mathbf{U}_{high}$$
$$\Delta\mathbf{U}_{low} \leq \Delta\mathbf{U} \leq \Delta\mathbf{U}_{high} \tag{8.4}$$

where p and m are the standard tuning parameters used in MPC, corresponding to the prediction and input move horizons respectively. The reference trajectory, $y^*(k+i)$ in Equation (8.4), is computed by subtracting the filtered value of the disturbance from the filtered value of the setpoint (defined in more detail below). Hernández and Arkun (1993) and Sriniwas et al. (1995) filtered the setpoint but used no filter in the feedback path. Placing a filter in the feedback path yields improved tuning (García and Morari, 1985; Prett and García, 1988; Ricker, 1990) and was very advantageous for the multivariable Volterra MPC case study in Maner and Doyle III (1997). This approach effectively treats the disturbance as a step passed through a first-order system. Wellons and Edgar (1987) originally proposed this disturbance model, and Ricker (1991) demonstrated that it can yield improved performance over using a step or ramp disturbance model.

The two filters $F_1(z)$ and $F_2(z)$ in Figure 8.1 were implemented in the same manner as the reference model in the work by Ricker (1990). For $F_1(z)$, a model was obtained for:

$$x_{r1}(k+1) = \Phi_{r1}x_{r1}(k) + \Gamma_{r1}\hat{d}(k|k)$$
$$y_{r1}(k) = x_{r1}(k).$$

The model for $F_2(z)$ was given by:

$$x_{r2}(k+1) = \Phi_{r2}x_{r2}(k) + \Gamma_{r2}r(k)$$
$$y_{r2}(k) = x_{r2}(k)$$

and

$$r^*(k+i) = y_{r2}(k+i) - y_{r1}(k+i), \quad i = 1, \ldots, p.$$

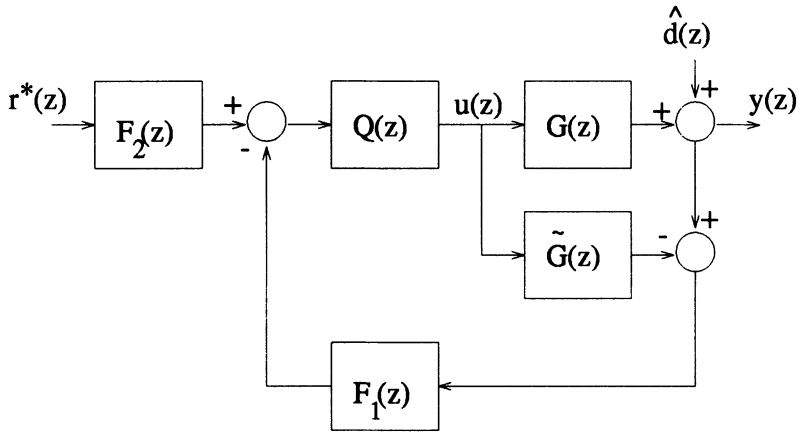

Fig. 8.1. Internal model control block diagram with two filter structure

Hence, the control problem at the beginning of sampling period k is as follows: compute the reference trajectory $r^*(k+i)$ over the prediction horizon p. Calculate the manipulated variable by solving the optimization problem in Equation (8.4). Update the state via Equation (8.3). Obtain the plant measurement $\tilde{y}(k+1)$. Compute $\hat{d}(k+1|k+1)$. Set $k = k+1$ and repeat the sequence.

8.3.5 Impact of model structure on NLP structure

The structure of the Volterra series model-based MPC offers opportunities for customized solution via efficient programming solvers. For example, a third-order Volterra model will lead to a sixth-order programming problem regardless of the value of the prediction horizon p. This same structure holds for AR-Volterra models. Though such a nonlinear program is more challenging to solve than the second-order (quadratic) program encountered in linear MPC, it is easier to solve than the nonlinear programs encountered in some other nonlinear MPC schemes. A widely used algorithm for unconstrained

optimization is the Broyden-Fletcher-Goldfarb-Shanno method (Dennis, Jr. and Schnabel, 1983). MINOS 5.1 (Murtagh and Saunders, 1987) employs a sequential quadratic programming approach to solving nonlinear constrained problems. Both of these solvers approximate the objective function using a quadratic form. Hence, if two objective functions are considered in which one objective function can be more accurately approximated by a quadratic form, the corresponding optimization problem could be expected to be solved more efficiently and reliably. In addition, one group of researchers (Chow et al., 1994) has developed an algorithm for unconstrained optimization using tensor methods that employs a fourth-order model of the objective function. In their case studies, the tensor method required significantly fewer iterations and function evaluations to solve most unconstrained optimization problems than standard methods based on quadratic models.

One can contrast this with the more general variations of nonlinear MPC that arise from more complex nonlinear models. Using a first principles process model directly in a nonlinear MPC scheme (Economou et al., 1986; Li and Biegler, 1988; Li and Biegler, 1989; Li and Biegler, 1990; Eaton and Rawlings, 1990) yields a general nonlinear program that may contain exponential nonlinearities due to Arrhenius temperature dependence or division of one state by another, as would be the case for computing the molecular weight in a polymerization reactor model. Since the quadratic approximation used by current nonlinear optimization algorithms may be a poor approximation for the objective function when a first principles model is employed, the efficiency of the nonlinear solver is reduced. Alternatively, one could employ various approximations of the system, as described in Section 8.4.

The general class of polynomial ARMA models that were discussed in Chapter 3 has the advantage that the model nonlinearities are polynomial nonlinearities, as opposed to exponential nonlinearities that do not appear in the form of a quadratic approximation. However, for the general case of nonlinearities in the y regressors, such as $y^2(k-1)y(k-2)$ and $y(k-1)u(k-2)$, the objective function becomes a higher-order function of the decision variables ($u_1(k), u_2(k), \ldots$, for example) as p is increased. For a modest value of $p = 5$, the resulting objective function could be 20th-order in the decision variables, for example. In general, it will be more difficult to approximate a higher-order objective function with a quadratic approximation, which could lead to an unreliable solution using current nonlinear optimization techniques.

8.4 Numerical approaches to problem solution

In Section 8.3, we developed several variations of the nonlinear MPC problem for Volterra series models. The solution of the resulting nonlinear programming (NLP) problem can take a number of approaches, several of which are described in this section.

8.4.1 Successive substitution

The nonlinear unconstrained MPC algorithm that results for a second-order Volterra model is conceptually quite similar to the corresponding IMC problem (Chapter 6), and its solution can be studied using the schematic diagram in Figure 8.2. Here, $\tilde{\mathbf{P}}_1[u]$ and $\tilde{\mathbf{P}}_2[u]$ denote the first-order and second-order terms of the Volterra model respectively. $\mathbf{P}$ is the actual nonlinear plant being controlled. $\tilde{\mathbf{P}}_1^{\dagger}$ is an inverse or pseudo-inverse of the linear model and is identical to the inverse used in linear MPC; ξ is the difference between a vector of desired future setpoints and a vector of future predicted outputs due to past inputs. It can be seen from Figure 8.2 that the second-order Volterra model-based algorithm is an extension of the constrained internal model control algorithm described by Ricker (1985). The second-order Volterra model

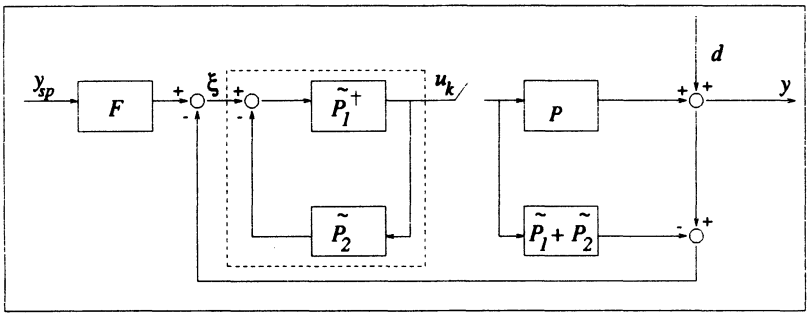

Fig. 8.2. Block diagram for the second-order Volterra model predictive controller

replaces the impulse response model and is used for output prediction. The corresponding Volterra controller, enclosed by the dashed box in Figure 8.2, replaces the linear model-based controller and computes the control action. For the constrained case, a constrained quadratic program is solved at each iteration through the $\tilde{\mathbf{P}}_1^{\dagger} - \tilde{\mathbf{P}}_2$ loop in Figure 8.2. Hence, the iterative loop guarantees that the control action calculated by the Volterra controller satisfies the constraints on the manipulated and controlled variables. The model equations for the SISO case are:

$$\begin{bmatrix} y(k+1) \\ y(k+2) \\ \vdots \\ y(k+p) \end{bmatrix} = \begin{bmatrix} h_1(1) & 0 & \cdots & 0 \\ h_1(2) & h_1(1) & \ddots & 0 \\ \vdots & \vdots & \ddots & h_1(1) \\ \vdots & \vdots & \ddots & h_1(1)+h_1(2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(p) & h_1(p-1) & \cdots & \sum_{i=1}^{p-m+1} h_1(i) \end{bmatrix} * \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+m-1) \end{bmatrix} + \begin{bmatrix} c(k+1) \\ c(k+2) \\ \vdots \\ c(k+p) \end{bmatrix} + \begin{bmatrix} f(k+1) \\ f(k+2) \\ \vdots \\ f(k+p) \end{bmatrix} \tag{8.5}$$

where

$$\begin{bmatrix} c(k+1) \\ c(k+2) \\ \vdots \\ c(k+p) \end{bmatrix} = \begin{bmatrix} h_1(2) & h_1(3) & \cdots & \cdots & h_1(N) & 0 \\ h_1(3) & h_1(4) & \cdots & h_1(N-1) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_1(p-1) & h_1(p) & \vdots & \vdots & \vdots & \vdots \\ h_1(p) & 0 & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \vdots & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} u(k-1) \\ u(k-2) \\ \vdots \\ u(k-N) \end{bmatrix} + \begin{bmatrix} d(k+1) \\ d(k+1) \\ \vdots \\ d(k+1) \end{bmatrix} + \begin{bmatrix} g(k+1) \\ g(k+2) \\ \vdots \\ g(k+p) \end{bmatrix} \tag{8.6}$$

and p, m, and N are the output prediction horizon, input move horizon, and model truncation order respectively. The future-future and future-past cross terms of the input correspond to $f(k+1), f(k+2), \ldots$, etc. The past-past cross terms of the input appear as $g(k+1), g(k+2), \ldots$, etc. $d(k+1)$ is the difference between the measured value of the process output and predicted model output at time k. Introducing vector notation, Equations (8.5) and (8.6) may be written in a more compact form as:

$$\mathbf{y} = \mathbf{G}\mathbf{u} + \mathbf{c} + \mathbf{f} \tag{8.7}$$

$$\mathbf{c} = \mathbf{H}\mathbf{u}_{past} + \mathbf{d} + \mathbf{g}. \tag{8.8}$$

Equations (8.7) and (8.8) are similar to the equations in Ricker (1985) except for the presence of the $\mathbf{f}$ and $\mathbf{g}$ terms, which arise due to the second-order model. $\mathbf{c}$ is the invariant contribution of past inputs and does not change with iterations around the $\tilde{\mathbf{P}}_1^{\dagger} - \tilde{\mathbf{P}}_2$ loop in Figure 8.2. $\mathbf{G}$ and $\mathbf{H}$ are matrices that contain impulse response coefficients. $\mathbf{u}_{past}$ is an historical record of past values of the input, and $\mathbf{u}$ is the linear contribution of the present and future values of the input. In addition, a setpoint trajectory $\mathbf{s}$ may be implemented via the following equations:

$$s(k+1) = \lambda y_{meas} + (1-\lambda) y_{sp}$$
$$s(k+i) = \lambda s(k+i-1) + (1-\lambda) y_{sp}, \qquad 2 \le i \le p$$

where λ is a tuning parameter (in addition to m and p) used to filter the setpoint.

To illustrate the computation of $\mathbf{f}$ and $\mathbf{g}$, consider the SISO case where $m = 2$, $p = 3$, and $N = 4$. Define $\mathbf{B}$, a matrix containing the second-order coefficients, as

$$\mathbf{B} = \begin{bmatrix} h_2(1,1) & h_2(1,2) & h_2(1,3) & h_2(1,4) \\ 0 & h_2(2,2) & h_2(2,3) & h_2(2,4) \\ 0 & 0 & h_2(3,3) & h_2(3,4) \\ 0 & 0 & 0 & h_2(4,4) \end{bmatrix}.$$

At time k, $u(k)$ and $u(k+1)$ are calculated. The entries in $\mathbf{f}$ are given by

$$\begin{aligned} f(k+1) &= [u(k) \;\; 0 \;\; 0 \;\; 0] * \mathbf{B} * [u(k) \;\; u(k-1) \;\; u(k-2) \;\; u(k-3)]^T \\ f(k+2) &= [u(k+1) \;\; u(k) \;\; 0 \;\; 0] * \mathbf{B} * \\ &\quad [u(k+1) \;\; u(k) \;\; u(k-1) \;\; u(k-2)]^T \\ f(k+3) &= [u(k+2) \;\; u(k+1) \;\; u(k) \;\; 0] * \mathbf{B} * \\ &\quad [u(k+2) \;\; u(k+1) \;\; u(k) \;\; u(k-1)]^T. \end{aligned} \tag{8.9}$$

Since $m = 2$, $u(k+1) = u(k+2)$ in Equation (8.9). The entries in $\mathbf{g}$ are given by

$$\begin{aligned} g(k+1) &= [0 \;\; u(k-1) \;\; u(k-2) \;\; u(k-3)] * \mathbf{B} * \\ &\quad [0 \;\; u(k-1) \;\; u(k-2) \;\; u(k-3)]^T \\ g(k+2) &= [0 \;\; 0 \;\; u(k-1) \;\; u(k-2)] * \mathbf{B} * [0 \;\; 0 \;\; u(k-1) \;\; u(k-2)]^T \\ g(k+3) &= [0 \;\; 0 \;\; 0 \;\; u(k-1)] * \mathbf{B} * [0 \;\; 0 \;\; 0 \;\; u(k-1)]^T. \end{aligned}$$

For the unconstrained case, the control action is computed at time k using the following algorithm:

Step 1. Set $i = 1$.

Step 2. Calculate $\mathbf{a}$ and $\mathbf{u}$ by solving the unconstrained least-squares control problem:

$$\begin{aligned} \mathbf{a} &= \mathbf{G}^T(\mathbf{s} - \mathbf{c} - \mathbf{f}) \\ \mathbf{u} &= (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{a}. \end{aligned} \tag{8.10}$$

Step 3. Determine if the condition in the following equation is satisfied, where Δ is the desired tolerance:

$$| \, u^{(i)}(k) - u^{(i-1)}(k) \, | \quad \leq \quad \Delta$$

and the superscript denotes the iteration step.

Step 4.

- If yes, set $u(k) = u^{(i)}(k)$. Close the switch in Figure 8.2 and implement $u(k)$.

- If no, recalculate $\mathbf{f}$ using $u^{(i)}(k)$ for present and future values of the input. Set $i = i+1$. Return to Step 2. (For the constrained case, $\mathbf{u}$ is calculated by solving a quadratic program rather than using the least-squares solution in Equation (8.10).)

Contraction mapping arguments can be made to show convergence of the successive substitution algorithm (Economou, 1985). Additional details of the algorithm as well as examples of the constraint handling and disturbance rejection capabilities appear in Maner (1993).

8.4.2 Quasi-Newton and related methods

A number of additional methods are available for efficient numerical solution of the MPC problem posed in Equation (8.1). For example, quasi-Newton techniques are available in standard optimization software toolkits, and their application to a Volterra series MPC problem is reported in Kirnbauer and Jörgi (1992). A novel application of ellipsoidal cutting plane techniques is reported in M'Sahli et al. (1998) for a Volterra MPC problem.

Rather than review standard approaches to constrained nonlinear optimization for this problem, we will highlight a number of new developments in this area, with an emphasis on the nonlinear MPC problem. Although not all of the methods described below have been directly developed for Volterra series models, they are relevant techniques that may be customized to the structure of the Volterra series MPC problem.

Arkun and co-workers (Simminger et al., 1991; Peterson et al., 1992) proposed a refinement of the extended quadratic dynamic matrix control (QDMC) algorithm (García, 1984) that retained the character of the basic linear DMC problem. This is particularly relevant for the Volterra MPC problem because of the natural partitioning of the model structure as discussed in Chapter 6. In effect, they reinterpret the disturbance vector according to the following definition:

$$d = d^{ext} + d^{nl}$$

where d^{ext} is constant across the horizon, and d^{nl} is integrated from the nonlinear model. This leads to a parameterized quadratic programming (QP) problem:

$$\begin{aligned} &\min_{\Delta U} \tfrac{1}{2}(\Delta U Q \Delta U + G^T(d^{nl})\Delta U) \\ &\text{s.t.} \qquad C\Delta U \geq c(d^{nl}). \end{aligned}$$

The correct value of the parameter d^{nl} is obtained from the solution of the following nonlinear program:

$$\min_{\Delta d^{nl}} \tfrac{1}{2} \left\| \begin{matrix} (y^{nl}(k+1) + d^{ext} - y^{el}(k+1)) \\ \vdots \\ (y^{nl}(k+p) + d^{ext} - y^{el}(k+p)) \end{matrix} \right\|$$

This NLP is solved using a Levenberg-Marquardt algorithm in the original reference.

Another approximation of the original QDMC problem is described by Patwardhan and Madhavan (1993). They employ a second-order approximation, which would be exact for second-order Volterra series models, and would involve a simple update for higher-order Volterra series models. They derive a second-order approximation using a Taylor series expansion at the current operating point and an integration of the sensitivity equations to obtain a bilinear approximation. They point out that the resultant algorithm combines

the properties of pseudo-Newton techniques (Li and Biegler, 1988; Li and Biegler, 1989; Li and Biegler, 1990) with variable sensitivity along the prediction horizon, and the fixed point linearization of Brengel and Seider (1989). The ultimate solution technique is MATLAB's *constr* command combined with analytical gradients from the sensitivity matrices.

A review of several of these techniques is provided in Mutha et al. (1997, 1998), where they compare the successive substitution technique from Section 8.4.1, the original extended QDMC, and a novel technique from the authors that is related to the second-order approximation of (Patwardhan and Madhavan, 1993). The crucial distinction is in the calculation of an updated linear approximation. The original extended QDMC approach (García, 1984) used an updated Jacobian that was parametrized by the operating point. The Volterra MPC controller described in Section 8.3 combined all the terms in the second-order expansion that contained future manipulated variable moves. Finally, the approach in Mutha et al. (1997, 1998) used both the level of the operating point plus the input signal to generate an approximation to the full nonlinear system.

8.4.3 Approximation solution methods

In addition to the numerical approximations described in Section 8.4.2, a number of approximations to the original control problem in Equation (8.1) have been employed. Most of these techniques involve some reduction in the number of unknown variables, such that an analytical solution could be obtained.

Some of these approximations include:

- unconstrained, one-step ahead prediction for the output variable (Sommer, 1994; Haber et al., 1999a)
- unconstrained, constant value of future manipulated variable moves (Haber, 1989; Haber, 1995; Wellers and Rake, 1998)
- unconstrained, constant increment of future manipulated variable moves (Haber, 1989, 1995)
- input blocking or changing the interval between input increments (Kirnbauer and Jörgi, 1992; Wellers and Rake, 1998).

8.5 Stability analysis

8.5.1 Nominal stability

One advantage of both Volterra models and AR-Volterra models is that simple open-loop stability conditions are available for these model classes, which can be used to develop closed-loop stability conditions. First, recall from Chapter 2 (Section 2.3.4) that Volterra models are BIBO stable. Similarly,

recall from Chapter 3 (Section 3.6.4) that AR-Volterra models are defined by Equation (3.16), repeated here for convenience:

$$y_k = y_0 + \sum_{i=1}^{p} \gamma_i y_{k-i} + \sum_{n=1}^{N} v_M^n(k),$$

where $v_M^n(k)$ is the n^{th}-degree nonlinear part of a $V_{(N,M)}$ Volterra model driven by the input sequence $\{u_k\}$. This model may be rewritten as the cascade connection of the $V_{(N,M)}$ model, followed by a linear $AR(p)$ model:

$$w_k = y_0 + \sum_{n=1}^{N} v_M^n(k)$$

$$y_k = \sum_{i=1}^{p} \gamma_i y_{k-i} + w_k.$$

Now, suppose $|u_k| \leq A$ for all k for some finite bound A. Since $V_{(N,M)}$ models are BIBO stable, it follows that there exists some finite B such that $|w_k| \leq B$ for all k. Hence, the overall model is BIBO stable if the linear $AR(p)$ model defined in this cascade decomposition is stable. In particular, the stability of an AR-Volterra model is seen to depend only on the pole locations of the linear autoregressive part of this model, and may be determined from the coefficients γ_i by standard results for linear discrete-time models (Elaydi, 1996, chapter 4).

If these stability conditions are satisfied, the AR-Volterra model is BIBO stable and nominal stability analysis of the closed-loop system is simplified considerably. Transfer function techniques such as IMC can be used in the analysis of MPC in the case of linear, unconstrained systems (Prett and García, 1988). Figure 8.1 is a block diagram of the IMC structure where filters may be used for setpoint and disturbance filtering, $\mathbf{F}_2(z)$ and $\mathbf{F}_1(z)$, respectively. The plant is denoted by $\mathbf{G}(z)$, and $\tilde{\mathbf{G}}(z)$ corresponds to the model of the process. $\mathbf{Q}(z)$ is the controller designed using $\tilde{\mathbf{G}}(z)$. Consider the nominal ($\mathbf{G}(z) = \tilde{\mathbf{G}}(z)$) case where $\mathbf{G}(z)$ is an AR-Volterra model. For the nominal case, there is no feedback in Figure 8.1 and $\mathbf{Q}(z)$ becomes a feedforward controller. Closed-loop stability is guaranteed if each of the three blocks in series ($\mathbf{F}_2(z)$, $\mathbf{Q}(z)$, and $\mathbf{G}(z)$) is stable. The setpoint filter is specified by the designer and contains stable transfer functions on its diagonal. Hence, $\mathbf{F}_2(z)$ is stable. The BIBO stability condition discussed here does not guarantee that the plant will be stable for an unbounded input signal. However, the input to the plant is the output of a control valve that is always physically bounded. Hence, closed-loop BIBO stability is guaranteed from the setpoint to the output provided the roots of the linear characteristic equation lie inside the unit circle.

Economou et al. (1986) employed the small gain theorem (Zames, 1966) and the IMC structure to obtain a closed-loop stability condition for control of

a nonlinear system using the inverse of the nonlinear process as the controller. The stability condition was also applied to the case in which a robustness filter was placed in the feedback path to de-tune the controller for plant-model mismatch.

Hernández (1992) considered the extended neighborhood stability of the inverse controller ($m = p = 1$) for two examples having a polynomial ARMA structure. The nonlinear model was approximated as a linear plus uncertain system. One of the contributions of his work was that the stability of the inverse of a linear plus uncertain system having an $M - \Delta$ structure could be determined without computing its inverse. Avoiding the construction of the analytical expression of the inverse is advantageous because it is impractical in general. If M_{22} and $(I - M_{11}\Delta)$ are invertible, the system may be transformed into an $M^I - \Delta$ configuration representing the inverse of the system using the following relationships (Hernández, 1992):

$$\begin{aligned}
\Delta^I &= \Delta \\
M_{11}^I &= M_{11} - M_{12}M_{22}^{-1}M_{21} \\
M_{12}^I &= M_{12}M_{22}^{-1} \\
M_{21}^I &= -M_{22}^{-1}M_{21} \\
M_{22}^I &= M_{22}^{-1}.
\end{aligned}$$

Conic sectors are used to bound the nonlinearities. For example, the following equation describes the bounding of a nonlinear function $f(\underline{x}(k), u(k))$ using a single cone with center $[A \quad b]$ and radius $[R_x \quad r_b]$.

$$f(\underline{x}, u) = [A \quad b]\begin{bmatrix}\underline{x} \\ u\end{bmatrix} + \Delta(\underline{x}, u)[R_x \quad r_b]\begin{bmatrix}\underline{x} \\ u\end{bmatrix} \tag{8.11}$$

where $\Delta(\underline{x}, u) \in [-1, 1]$ represents a scalar uncertainty. The first and second terms on the right-hand side of Equation (8.11) are linear and uncertain parts of the approximation respectively.

If the system is bounded in a region by conic sectors and it is assumed that the system does not leave this region, its inverse will be stable if $(I - M_{11}\Delta)$ and M_{22} are invertible and

$$\min_D \overline{\sigma}(DM_{11}^I D^{-1}) < 1. \tag{8.12}$$

The optimization problem in Equation (8.12) is a convex optimization problem that is solved for a constant scaling matrix D to provide a less conservative upper bound for μ, the structured singular value (Doyle, 1982). The condition in Equation (8.12) guarantees that the controller is exponentially stable. Hence, it is a more stringent stability guarantee than BIBO stability.

Though the stability of the inverse controller yields insight into the stability of the closed-loop system, the inverse controller is rarely used in practice because it usually requires very aggressive manipulated variable action. Aggressive manipulated variable profiles are undesirable because they cause excessive wear on control valves and may be an inefficient use of utilities such

as steam and cooling water. In addition, the inverse controller can perform poorly when plant-model mismatch is significant. Also, the inverse controller will be unstable for a model that contains a zero that lies outside the unit circle. These problems can be avoided if a relaxed inverse controller is used where $p > m$ in the MPC notation. However, there are no general stability results for the controller $\mathbf{Q}(z)$ in Figure 8.1 if $\mathbf{Q}(z)$ is the general nonlinear model predictive controller in Equation (8.2). However, Hernández (1992) proposed a method for studying the stability of the p-inverse controller. This controller is a particular case of the model predictive control algorithm where $m = 1$, $\Gamma_U = 0$, $\Gamma_Y(i) = 0$ for $i = 1, \ldots, p-1$, and $\Gamma_Y(p) = 1$. The analysis of the p-inverse controller consists of describing the nonlinear model as a linear plus time-varying uncertain model relating $y(k+p)$ to $u(k)$ where $p > 1$. The stability of the p-inverse controller is guaranteed provided that the inverse of the model relating $y(k+p)$ to $u(k)$ is stable. The number of uncertainty parameters increases as p increases. Hence if the uncertainty description is conservative, the increase in the number of uncertainty parameters results in additional conservatism limiting its usefulness. A conservative unstructured uncertainty description was employed by Hernández (1992).

BIBO closed-loop stability is guaranteed from the setpoint to the output for the nominal case provided the roots of the linear characteristic equation lie inside the unit circle and the condition in Equation (8.12) is satisfied for the p-inverse controller. In addition, if a μ analysis is also performed on the model and $\mu < 1$, each of the three operators in series ($\mathbf{F}_2(z)$, $\mathbf{Q}(z)$, and $\mathbf{G}(z)$) is exponentially stable and exponential closed-loop stability is guaranteed for the nominal case from the setpoint to the output.

8.5.2 Robust stability and performance

Robustness performance properties of general nonlinear MPC problems are the subject of active research. In the specific case of the Volterra series-based MPC controller, there are limited results presented in (Genceli and Nikolaou, 1995). Among the assumptions employed in their derivation are: (i) an ℓ_1 objective function metric, (ii) step-like disturbances, and (iii) an end-point condition is required that ensures that the input reaches an appropriate steady-state value at the end of a finite horizon. The stability conditions are based on Lyapunov arguments.

8.6 Application of Volterra series model predictive control

8.6.1 Customized nonlinear MPC approaches

In the area of advanced formulations of the nonlinear MPC problem, the following applications are reported. The second-order approximation technique

of Patwardhan and Madhavan (1993) was applied to two different examples: (i) a simple reactor with two inputs and two outputs, and (ii) a fermentor with two inputs and two outputs. Their results demonstrated improvement over both quasi-Newton methods and the original extended QDMC approach in terms of faster and smoother responses. An interesting aspect of their study was the consideration of extremum control, i.e. holding an optimum value at setpoint in a system with a parabolic steady-state locus. They also explored the robustness of their algorithm to parametric uncertainty in the process model.

Arkun and co-workers (Simminger et al., 1991; Peterson et al., 1992) applied their iterative QDMC technique to several process examples including a simple stirred tank reactor, and a more complex methyl methacrylate (MMA) polymerization reactor, operated in a semi-batch mode. The two inputs are jacket temperature and initiator flowrate, and the two outputs are the temperature and NAMW. In the development of their nonlinear model, they exploit the physical knowledge of the interplay between temperature and molecular weight. As a result, they are able to use a reduced isothermal model as a subcomponent of their molecular weight controller.

A comparison of several techniques is reported in Mutha et al. (1997) and Mutha et al. (1998) with application to several complex reactor systems. These include the MMA reactor mentioned previously (and studied in Chapter 9), and a semi-batch acrylonitrile-butadiene emulsion polymerization. The second example is fairly complex, and consists of three controlled outputs (conversion, percentage bound acrylonitrile, polydispersity) and four manipulated variables (monomer flow, activator flow, chain transfer agent flow, reactor temperature). In a number of simulated runs, they conclude that the successive substitution method described previously may have convergence problems, whereas their method outperforms conventional extended QDMC.

8.6.2 Volterra series systems

In addition to application studies with general programming techniques, there are also several reported studies of Volterra series-based MPC.

Dumont and co-workers (Dumont and Fu, 1993; Dumont et al., 1994) employ the Volterra-Laguerre model structure described in Section 3.6.3 in an adaptive GPC algorithm. They derive an analytical solution to the unconstrained, second-order problem under the assumptions of one step-ahead control action ($m = 1, p = 1$). Furthermore, they require that $\Gamma_U = 0$. The method is demonstrated on a simulated wood chip refiner motor load, in which the motor load is regulated using the refiner plate gap. The steady-state curve between these two variables exhibits parabolic properties, thus motivating the second-order approach. The second simulation example considered is a pH titration problem, in which strong base flow is adjusted to control a reactor pH. The physical system has the characteristics of a static

nonlinearity (titration curve) coupled with linear dynamics (mixing tank); hence, a Wiener model is developed with second-order dynamics and a third-order polynomial static nonlinearity. In comparisons with linear MPC, the adaptive nonlinear approach exhibits a significantly broader domain of robust performance.

Doyle and co-workers (Maner et al., 1996; Maner and Doyle III, 1997; Parker and Doyle III, 2001) have explored the application of Volterra series-based MPC to a number of chemical reactor problems, including polymerization systems and biochemical processes. They have considered second-order Volterra structures in Maner et al. (1996), third-order AR-Volterra structures in Maner and Doyle III (1997), and second-order Volterra-Laguerre structures in Parker and Doyle III (2001). The theoretical details of their approach have been described in this chapter.

Additional studies include the work described by Sommer (1994) detailing a simulated filling position control problem. M'Sahli et al. (1998) studied the nonlinear benchmark problem of the Van de Vusse kinetics scheme in a simple stirred tank reactor (also discussed in Chapter 9). Wellers and Rake (1998) also consider the Van de Vusse reactor as well as a pH neutralization process. Genceli and Nikolaou (1995) consider a simulated simple irreversible chemical reaction in a stirred tank with input magnitude and rate constraints. In Haber et al. (1999a,b), the application of Volterra series-based MPC to an unconstrained two-tank level control problem is described. They employ a recursive least-squares estimation technique to adapt the parameters of the model on-line.

8.6.3 Wiener and Hammerstein systems

A note about the sub-class of Wiener and Hammerstein systems is warranted, given their treatment in the modeling sections of Chapter 3. The block structured nature of these models gives rise to customized solution methods for the corresponding nonlinear MPC problem. This has been exploited by a number of authors in application case studies.

Kurth and Rake (1994a) describe the application of such a technique for the control of an experimental diesel engine. The main control objective was the regulation of mixed gas temperature using the fuel valve opening. The structure of the corresponding model was a static second-order nonlinearity coupled with a first-order linear dynamic component.

A polynomial Hammerstein model is also reported in Zhu and Seborg (1994a,b) with application to a simulated pH neutralization process. The nature of the pH process required that two overlapping models be constructed to capture fully the static nonlinear titration curve. The resultant MPC controller showed significant improvement over a simple PID controller.

Fruzzetti et al. (1997) carried out a formal analysis of a Hammerstein-based MPC algorithm, and provided results for nominal stability and asymptotic tracking. They described two examples: (i) a simulated SISO pH control

problem, and a (ii) 2×2 simulated distillation column. Their Hammerstein model for the pH problem includes up to fourth-order nonlinear terms for the input variable, and a linear output term. In effect, they construct an AR-Volterra model. Similarly, they construct a multivariable model for the distillation example that includes quadratic cross terms in both inputs and linear output terms, once again yielding an AR-Volterra model. All models were identified from data, and the results show a significant improvement for the nonlinear Hammerstein controller versus a linear MPC controller.

8.7 Summary

In this chapter, we have reviewed the development of several model predictive control algorithms that rely on a Volterra series system representation. As was described in preceding chapters, the Volterra models share the properties of linear impulse response models in relative ease of development and intuitive model structure.

A number of additional advantages of Volterra series-based MPC have been highlighted as well. These include the fact that the order of the resultant NLP problem is $2n$, where n is the order of the Volterra model. Consider the case of a second-order Volterra model ($n = 2$). While the resulting fourth-order nonlinear program is more computationally demanding than the quadratic program that arises in linear MPC, it is more readily solvable than the general nonlinear programs that arise in the NLIMC algorithm and nonlinear MPC based on polynomial ARMA models noted above. In particular, one group of researchers (Chow et al., 1994) has developed an algorithm for unconstrained optimization using tensor methods that employs a fourth-order model of the objective function. In their case studies, the tensor method required significantly fewer iterations and function evaluations to solve most unconstrained optimization problems than standard methods based on quadratic models. Another advantage of Volterra series-based model predicitve controllers is that analysis results (e.g. stability) are easier to derive, since the model has a nonlinear dependence on only the previous input variables and not on the previous output variables.