Chapter 9

APPLICATIONS ON OTHER PROBLEMS

In this chapter, the forward-backward splitting methods are applied to solve problems of stochastic programming, chemical engineering, variational inequalities, and some other problems.

9.1. Applications to Multistage Stochastic Programming

Multistage stochastic programming problems have received wide attention recently because of their application in engineering, management sciences, financial business, industry, economics, etc. Since most of such problems are large-scale type or with penalty expressions, they are difficult to solve. Numerical methods for them mostly depend on decomposition approaches currently. Some of the literature on these approaches may be found in the work of Cohen and Culioli [56], Rockafellar and co-workers [274], [295] – [298], [331], *etc.* In this section, we shall apply the EFBS algorithm to solve problems of multistage stochastic programming. The algorithm is a new decomposition approach for the problems. We call it *dynamical splitting* decomposition.

Consider the scenario tree in the next page with fixed transition probabilities, where nodes $i \in I$ are decision states, i^- is the unique predecessor of i, i^+ are various successors of i, and 0 is the initial state. Assume that J is the set of all terminal states, $u_i \in \mathbb{R}^{m_i}$ is a decision chosen in state i, policy u is $u : i \to u_i$ for all $i \in I$, and π_i is the probability of arriving at i.

Let $x_i \in \mathbb{R}^{n_i}$ be an auxiliary state vector, which accounts for effects of past decisions $u_0, \cdots, u_{i^-}$ on the choice of u_i, and be present in decision state $i \in I$.

We consider a multistage model here as follows:

$$\min_u \sum_{i \in I} \pi_i f_i(x_i, u_i), \tag{9.1.1}$$

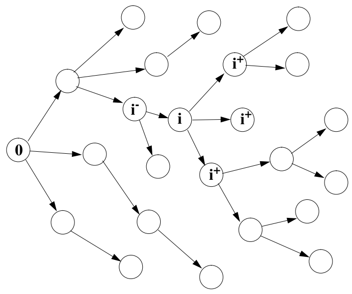

Fig. 9.1: Scenario Tree with Fixed Transition Probabilities

subject to

$$x_i = A_i x_{i^-} + B_i u_{i^-} + b_i \quad \text{for} \quad i \neq 0, \qquad x_0 = b_0, \tag{9.1.2}$$

where $f_i(x_i, u_i)$ is a convex cost function in decision state $i \in I$ and (9.1.2) is the dynamics.

In this section, we consider the following special cost function:

$$f_i(x_i, u_i) = p_i \cdot u_i + \varphi_i(u_i) + \rho_i(q_i - C_i x_i - D_i u_i) - c_i \cdot x_i, \tag{9.1.3}$$

where φ_i and ρ_i are convex, lower semi-continuous, and proper. The ρ_i is considered as a *tracking function*. It has the dualization:

$$\rho_i(z_i) = \sup_{v_i} \{ z_i \cdot v_i - \psi_i(v_i) \} \quad \text{on} \quad \mathbb{R}^{s_i} \tag{9.1.4}$$

and

$$\rho_i = \psi_i^*, \qquad \psi_i = \rho_i^*, \tag{9.1.5}$$

i.e., ρ_i and ψ_i are conjugate each other.

Some examples (refer to Rockafellar [268], [269]) of φ_i, ρ_i, and ψ_i are:

(1). LP structure:

$$\varphi_i(u_i) = \delta_{\mathbb{R}_+^{m_i}}(u_i), \qquad \psi_i(v_i) = \delta_{\mathbb{R}_+^{s_i}}(v_i). \tag{9.1.6}$$

That is, φ_i indicates a constraint $u_i \geq 0$ and ρ_i indicates a constraint $q_i - C_i x_i - D_i u_i \leq 0$.

(2). Extended LP structure:

$$\varphi_i(u_i) = \delta_{U_i}(u_i), \qquad \psi_i(v_i) = \delta_{V_i}(v_i), \qquad U_i, V_i \ \text{ – boxes }. \tag{9.1.7}$$

Namely, φ_i is a constraint $u_i \in U_i$ (upper and lower bound) and ρ_i gives linear penalties on $q_i - C_i x_i - D_i u_i$.

(3). Extended LQP structure:

$$\varphi_i(u_i) = \delta_{U_i}(u_i) + \tfrac{1}{2} u_i{\cdot}P_i u_i, \quad \psi_i(v_i) = \delta_{V_i}(v_i) + \tfrac{1}{2} v_i{\cdot}Q_i v_i, \tag{9.1.8}$$

where U_i and V_i are polyhedral, P_i and Q_i are positive semidefinite. The expressions in (9.1.8) imply: φ_i equals a linear constraint plus a quadratic cost on u_i and ρ_i equals piecewise linear-quadratic penalties on $q_i - C_i x_i - D_i u_i$.

From Lagrangian Duality of Rockafellar (1993), a reduced Lagrangian, for Problem (9.1.1) associated with (9.1.3), is

$$L(u,v) = \sum_{i \in I} \pi_i l_i(u_i, v_i) - a(u,v), \tag{9.1.9}$$

where

$$l_i(u_i, v_i) = p_i{\cdot}u_i + \varphi_i(u_i) + q_i{\cdot}v_i - \psi_i(v_i) - v_i{\cdot}D_i u_i, \tag{9.1.10}$$

and

$$\begin{aligned} a(u,v) &= \sum_{i\in I} \pi_i v_i{\cdot}C_i x_i + \sum_{i\in I} \pi_i c_i{\cdot}x_i \\ &= \sum_{i\notin J} \pi_i u_i{\cdot}E_i\{B_{i^+} y_{i^+}\} + \sum_{i\in I} \pi_i b_i{\cdot}y_i. \end{aligned} \tag{9.1.11}$$

Where $y_i \in \mathbb{R}^{n_i}$ is a dual state vector and satisfies:

$$y_i = \begin{cases} E_i\{A^*_{i^+} y_{i^+}\} + C^*_i v_i + c_i, & \text{for } i \notin J \\ C^*_i v_i + c_i, & \text{for } i \in J, \end{cases} \tag{9.1.12}$$

and E_i equals the conditional expectation in state i relative to all successor states i^+.

The dual problem related to $L(u,v)$ is:

$$\max_v \sum_{i\in I} \pi_i \big[q_i{\cdot}v_i - \psi_i(v_i) - \varphi^*_i\big(E_i\{B^*_{i^+} y_{i^+}\} + D^*_i v_i - p_i\big) - b_i{\cdot}y_i\big], \tag{9.1.13}$$

subject to dynamics (9.1.12).

From the results of Rockafellar [258], $(\bar u, \bar v)$ is a saddle point of $L(u,v)$, if and only if, $\bar u$ solves the primal problem (9.1.1) associated with (9.1.3), and $\bar v$ solves the dual problem (9.1.13).

Thus, solving the problems (9.1.1) and (9.1.13) is equivalent to solving the saddle point problem associated with $L(u,v)$. For convenience, we assume that a solution of Problem (9.1.1) exists later in this section discussion.

Note: $L(u,v)$ has the following features:

(a) L is convex in u and concave in v;

(b) L is separable in u and separable in v (separately);

(c) The different decision states of $L(u,v)$ are linked only by the form $a(u,v)$.

Similar to Section 7.5 and Example 2.3.3, solving the saddle point associated with $L(u,v)$ is the same as finding the zero point of the following monotone operator T:

$$T(u,v) = \begin{bmatrix} \Big\{\pi_i\big(p_i + \partial_{u_i}\varphi_i(u_i) - D^*_i v_i\big)\Big\}_{i\in I} - \Big\{\pi_i E_i\{B_{i^+} y_{i^+}\}\Big\}_{i\notin J} \\ \Big\{\pi_i\big(\partial_{v_i}\psi_i(v_i) - q_i + D_i u_i + C_i x_i\big)\Big\}_{i\in I} \end{bmatrix}. \tag{9.1.14}$$

Let

$$T_1(u,v) = T(u,v) - T_2(u,v), \quad T_2(u,v) = \begin{bmatrix} \left\{\pi_i\big(p_i + \partial_{u_i}\varphi_i(u_i)\big)\right\}_{i\in I} \\ \left\{\pi_i\big(\partial_{v_i}\psi_i(v_i) - q_i\big)\right\}_{i\in I} \end{bmatrix}. \quad (9.1.15)$$

Then, two extended FBS algorithms may be described similarly to Algorithm 8.1 and Algorithm 8.2 of Chapter 8. Here just an algorithm without line search is given as follows (H_k is always chosen as $H_k = \operatorname{diag}\big\{\{P_{k,i}\}_{i\in I}, \{Q_{k,i}\}_{i\in I}\big\}$, for each $k \geq 0$, in the algorithm):

ALGORITHM 9.1:

0. **Initial input**: Input an initial point (u^0, v^0), initial $\lambda_0 > 0$, an error bound $\varepsilon > 0$ and a initial matrix H_0. Let $k := 0$.
1. **EFBS iteration:** Use the EFBS scheme to get (u^{k+1}, v^{k+1}) from (u^k, v^k). That is,

$$\begin{aligned} u_i^{k+1} &= \operatorname*{argmin}_u \left[\pi_i\big(p_i{\cdot}u + \varphi_i(u)\big) + \frac{1}{2\lambda_k}\|u - P_{k,i}^{-1}\hat{u}_i^k\|^2\right], \\ \hat{u}_i^k &= P_{k,i}u_i^k + \lambda_k\pi_i \begin{cases} D_i^* v_i^k, & \text{for } i \in J \\ D_i^* v_i^k + E_i\{B_{i+}y_{i+}^k\}, & \text{for } i \notin J, \end{cases} \qquad i \in I; \end{aligned} \quad (9.1.16)$$

and

$$\begin{aligned} v_i^{k+1} &= \operatorname*{argmin}_v \left[\pi_i\big(\psi_i(v) - q_i{\cdot}v\big) + \frac{1}{2\lambda_k}\|v - Q_{k,i}^{-1}\hat{v}_i^k\|^2\right], \\ \hat{v}_i^k &= Q_{k,i}v_i^k - \lambda_k\pi_i[C_i x_i^k + D_i u_i^k], \qquad i \in I. \end{aligned} \quad (9.1.17)$$

Where x_i^k, y_i^k are the solutions of the dynamics (9.1.2) and (9.1.12) associated with given u_i^k and v_i^k.
2. **Primal-dual gap check:** Compute $f_k = \mathcal{F}(u^{k+1})$ and $g_k = \mathcal{G}(v^{k+1})$, namely,

$$f_k = \max_v L(u^{k+1}, v), \qquad g_k = \min_u L(u, v^{k+1}). \quad (9.1.18)$$

If

$$f_k - g_k \leq \varepsilon, \quad (9.1.19)$$

then stop iteration and output the ε-optimal solution $(u^*, v^*) \approx (u^{k+1}, v^{k+1})$, otherwise, go to the next step.

3. **Distance check:** If

$$\|u^{k+1} - u^k\|_{P_k} + \|v^{k+1} - v^k\|_{Q_k} \leq \varepsilon, \tag{9.1.20}$$

stop iteration and output the ε-optimal solution $(u^*, v^*) \approx (u^{k+1}, v^{k+1})$, otherwise, go to the next step. Here, in (9.1.20), $P_k = \text{diag}\left\{\{P_{k,i}\}_{i\in I}\right\}$ and $Q_k = \text{diag}\left\{\{Q_{k,i}\}_{i\in I}\right\}$.

4. **Constant modification:** Choose suitable $\lambda_{k+1} > 0$ and H_{k+1}, then let $k := k + 1$. Go back to Step 1.

The convergence analysis of the above algorithm is given by Theorem 7.7.3. More corollaries, similar to Theorem 7.7.5, may be deduced from the results in Chapter 5. From here we shall not go further.

9.2. An Application to Process Control in Chemical Engineering

In this section, we will consider applying the EBFS algorithm to the model-predictive control system design for the process control in chemical enginering.

Model-predictive control (MPC) is a generic term for a class of process control algorithms. The idea is simple: one obtains or derives a process model and uses it (on-line) to decide how to adjust the available manipulated variables in response to disturbances, changing production goals, etc. It is a form of *optional control* in the sense that one usually solves a mathematical optimization problem in order to make a decision. There are many variants with respect to details of implementation. Over the past 15 years or so, a bewildering array of MPC methods has evolved (with an equally bewildering/confusing array of acronyms). Ricker (1985, 1990), Morari and Ricker (1994) have applied sequential quadratic programming methods to MPC and made it into software recently. Garcia *et al.* (1989) recently reviewed the technical and economic incentives for the use of MPC and its evolutionary history. There are also several new books on the subject (*e.g.,* Prett and Garcia, 1988; Morari and Zafirion, 1989) and MPC has been the focus of several recent workshops. Many industrial applications have been reported (*e.g.,* Ricker *et al.* 1986, 1989), especially over the last eight years, and versions of MPC are being marketed by most of the major vendors of process control software. Thus MPC is a relatively mature control technology.

The method we are applying here is different from all of the existing versions of MPC. It not only converges faster, but also can do much longer time prediction, or more suitable in large scale cases than the existing Model Predictive Control Toolbox in *MATLAB* for our testing examples. We shall see the results in the next section.

Fig. 9.2: State-Space Plant Model with a LTI System

The model we are considering is a state-space modeling with a linear time-invariant (LTI) system, shown in Figure 9.2 of the above.

Consider the approximation of a plant, shown in Fig. 9.2. Then, a state-space representation is:

$$\tilde{x}_{i+1} = \tilde{A}\tilde{x}_i + \Gamma_u u_i + \Gamma_d d_i + \Gamma_w w_i \tag{9.2.1}$$

$$y_i = \bar{y}_i + z_i = \tilde{C}\tilde{x}_i + \tilde{D}u_i + z_i \tag{9.2.2}$$

where $\tilde{x}$ is a state vector, u represents the manipulated variables, d includes measured but freely-varying inputs (*i.e.*, measured disturbances), w is a vector of unmeasured disturbances, y is a vector of plant outputs, z is measurement noise, and $\tilde{A}$, Γ_u, Γ_d, Γ_w, $\tilde{C}$, and $\tilde{D}$ are constant matrices of appropriate size. The variable $\bar{y}_i$ $[= \tilde{C}\tilde{x}_i + \tilde{D}u_i]$ represents the plant output before the addition of measurement noise.

The above plant model is not part of the control system. Its only purpose is to represent the plant in simulations and analytical work (refer to Ricker, 1990). Therefore, the actual state-space model in the model-predictive control design, which we especially consider in this section, is:

$$\begin{aligned} x_{i+1} &= Ax_i + Bu_i, \\ h_i &= Cx_i + Du_i. \end{aligned} \tag{9.2.3}$$

Here $x_i \in \mathbb{R}^{n_s}$, $u_i \in \mathbb{R}^{n_c}$, $h_i \in \mathbb{R}^{n_o}$, y is replaced by h for the symbol convenience of our duality discussion, and all vectors and matrices have similar definitions as for the plant model. The model (9.2.3) provides a prediction of future plant outputs as a function of contemplated adjustments in the manipulated variables and estimated disturbances. The controller chooses the values of u to send to the plant such that the *predicted* plant outputs are optimal according to a specified criterion. This model is part of the control system, and its states are all known exactly. Furthermore, the structure and parameters of the model (9.2.3) are also known.

In this dissertation, we consider only one kind of designs of the MPC systems, for an application of the forward-backward splitting method. Since $D = 0$ for most cases we meet, we set $D = 0$ for the rest of this discussion.

In order to determine whether a given choice of u_k is optimal, we must be able to predict the effect on the plant output. In MPC, this is done for a finite "prediction horizon" of N sampling periods, starting with period $k+1$. In the following, we just use the representations:

$$u_{k+i} = u_{\{k+i|k\}}, \quad x_{k+i} = x_{\{k+i|k\}}, \quad h_{k+i} = h_{\{k+i|k\}}. \tag{9.2.4}$$

In MPC, for the time k, a special control problem is formulated as an optimization problem in which the manipulated variables, u_{k+i}, are to be calculated to minimize the objective function:

$$\min_{u(k)} J(k) = \sum_{i=1}^{N} \tfrac{1}{2} u_{k+i-1}^T P_i u_{k+i-1} + \tfrac{1}{2}[r_{k+i} - h_{k+i}]^T Q_i^{-1} [r_{k+i} - h_{k+i}], \qquad (9.2.5)$$

subject to

$$x_{i+1} = Ax_i + Bu_{k+i}, \quad x_1 \text{ given}, \qquad (9.2.6)$$

$$h_{k+i} = Cx_i, \qquad i = 1, \cdots, N; \qquad (9.2.7)$$

$$u_{k+i-1} \in U_i, \quad h_{k+i} \in W_i, \quad i = 1, \cdots, N \qquad (9.2.8)$$

where $u(k)^T = [u_k^T, \cdots, u_{k+N-1}^T]$, U_i and W_i are boxes, P_i and Q_i are nonnegative "tuning parameters" for MPC (and are assumed to be diagonal here to simplify the tuning process), h_{k+i} is a prediction vector of output at a time i sampling periods into the future (relative to the current time, k), r_{k+i} is the corresponding future setpoint at the period $k+i$.

This optimization problem is actually a special case of Problem $(\mathcal{O}')$ of Section 8.2. Hence, one can directly apply the algorithms in Chapter 8 to solve it. Here, we shall directly transform the above problem as a special case of the model problem $(\mathcal{P}_d)$ of Section 3.1 through a special structure.

The problem (9.2.5) – (9.2.8) can be reformulated as the equivalent problem:

$$\begin{aligned} \min_{u(k)\in U} \sum_{i=1}^{N} \Big[& \tfrac{1}{2} u_{k+i-1}^T P_i u_{k+i-1} + \delta_{W_i}\big(Cx_i\big) \\ & + \rho_{\mathbb{R}^{n_o}, Q_i}\big(r_{k+i} - Cx_i\big)\Big] \end{aligned} \qquad (9.2.9)$$

subject to dynamics (9.2.6), where $U = U_1 \times \cdots \times U_N$ and $\rho_{V,Q}$ is defined by (2.2.2).

Case 1. When $W_i = \mathbb{R}^{n_o}$, the indicator function disappears in (9.2.9) and $V_i = \mathbb{R}^{n_o}$. Hence, solving the problem (9.2.5) – (9.2.8) and its dual problem is equivalent to solving a saddle point problem on $U \times V$ associated with the following Lagrangian:

$$L\big(u(k), v\big) = \sum_{i=1}^{N} \Big[\tfrac{1}{2} u_{k+i-1}^T P_i u_{k+i-1} + r_{k+i}^T v_i - \tfrac{1}{2} v_i^T Q_i v_i \Big] - l\big(u(k), v\big) \tag{9.2.10}$$

where $v^T = [v_1^T, \cdots, v_N^T]$, $V = V_1 \times \cdots \times V_N$,

$$l(u(k), v) = \sum_{i=1}^{N} v_i^T C x_i = \sum_{i=1}^{N} y_i^T B u_{k+i-1} \tag{9.2.11}$$

and the dual states y_i satisfy the dynamics:

$$y_i = A^T y_{i+1} + C^T v_i, \qquad i = N, \cdots, 1, \quad \text{with} \quad y_{N+1} = 0. \tag{9.2.12}$$

Case 2. When W_i is a box with upper bound h_u and lower bound h_l,

$$\delta_{W_i}(s) = \rho_{\mathbb{R}_+^{n_o}, 0}(h_l - s) + \rho_{\mathbb{R}_-^{n_o}, 0}(h_u - s). \tag{9.2.13}$$

Hence, the problem (9.2.9) with (9.2.6) is a special case of the problem $(\mathcal{P}_d)$ of Section 3.1 with $t = k+1, \cdots, k+N$. Thus, it is the primal problem associated with the following L, U, and V:

$$L\big(u(k), v\big) = \sum_{i=1}^{N} \Big[\tfrac{1}{2} u_{k+i-1}^T P_i u_{k+i-1} + q_i^T v_i - \tfrac{1}{2} v_i^T \bar{Q}_i v_i \Big] - l\big(u(k), v\big) \tag{9.2.14}$$

where

$$l(u(k), v) = \sum_{i=1}^{N} v_i^T \bar{C} x_i = \sum_{i=1}^{N} y_i^T B u_{k+i-1}, \tag{9.2.15}$$

$$V = V_1 \times \cdots \times V_N, \quad V_i = \mathbb{R}^{n_o} \times \mathbb{R}_+^{n_o} \times \mathbb{R}_-^{n_o}, \tag{9.2.16}$$

$$q_i = \begin{bmatrix} r_{k+i} \\ h_l \\ h_u \end{bmatrix}, \quad \bar{C} = \begin{bmatrix} C \\ C \\ C \end{bmatrix}, \quad \bar{Q}_i = \begin{bmatrix} Q_i & & \\ & 0 & \\ & & 0 \end{bmatrix}, \tag{9.2.17}$$

$\bar{Q}_i = \text{diag}\{Q_i, 0, 0\}$, U is the same as that in Case 1, and y_i is determined by the dynamics (9.2.12) except replacing C by $\bar{C}$.

In general MPC, we are doing the simulation not only at one time k, but also at many other times. Hence, we actually need to solve the optimization problem (9.2.5) – (9.2.8) repeatly at each time $k = 1, 2, \cdots, J$ for the given duration J of the simulation.

Samilar to Chapter 8, two EFBS algorithms may be obtained for each case. When $P = \text{diag}\{P_1, \cdots, P_N\}$ and $Q = \text{diag}\{Q_1, \cdots, Q_N\}$ are positive definite, we give an EFBS algorithm (without line search), as an example, for the basic splitting of Case 1:

$$L_1(u(k), v) = -l(u(k), v), \qquad T_1 = \begin{pmatrix} \nabla_{u(k)} L_1 \\ -\nabla_v L_1 \end{pmatrix}. \tag{9.2.18}$$

Note: $H_k \equiv \text{diag}\{P, Q\}$ here.

ALGORITHM 9.2 (MPC Algorithm):

0. **Duration time input:** Set $k := 1$.
1. **Initial input:** Let $m := 0$. Input an initial point $\left(u(k)^0, v^0\right)$, initial $\lambda_0 > 0$, an error bound $\varepsilon > 0$ and the matrix $H = \text{diag}\{P, Q\}$, where $u(k)^T = [u_k^T, \cdots, u_{k+N-1}^T]$.
2. **EFBS iteration:** To get $(u(k)^{m+1}, v^{m+1})$ from $(u(k)^m, v^m)$, use the EFBS scheme (8.1.3) and Theorem 7.8.1.

$$\begin{aligned} u_{k+i-1}^{m+1} &= \mathcal{P}_{U_i}\left([I + \lambda_m P_i]^{-1} P_i^{-1} \hat{u}_{k+i-1}^m\right), \\ \hat{u}_{k+i-1}^m &= P_i u_{k+i-1}^m + \lambda_m B^T y_i^m, \quad i = 1, 2, \cdots, N; \end{aligned} \tag{9.2.19}$$

and

$$\begin{aligned} v_i^{m+1} &= \mathcal{P}_{V_i}\left([I + \lambda_m Q_i]^{-1} [\lambda_m r_{k+i} + Q_i^{-1} \hat{v}_i^m]\right), \\ \hat{v}_i^m &= Q_i v_i^m - \lambda_m C x_i^m, \qquad i = 1, \cdots, N. \end{aligned} \tag{9.2.20}$$

Where $\mathcal{P}_W$ is the projection operator from the appropriate whole space to the set W, x_i^m and y_i^m are the solutions of the dynamic systems (9.2.6), (9.2.12) associated with given u_{k+i-1}^m, v_i^m.

3. **Primal-dual gap check:** Compute

$$f_m = \max_v L(u(k)^{m+1}, v), \qquad g_m = \min_u L(u, v^{m+1}),$$

namely,

$$f_m = L(u(k)^{m+1}, \tilde{v}), \qquad g_m = L(\tilde{u}, v^{m+1}) \tag{9.2.21}$$

where

$$\begin{aligned} \tilde{v}_i &= \mathcal{P}_{V_i}\left[Q_i^{-1}\left(r_{k+i} - Cx_i^{m+1}\right)\right], \quad i = 1, \cdots, N; \\ \tilde{u}_i &= \mathcal{P}_{U_i}\left[P_i^{-1}\left(B^T y_i^{m+1}\right)\right], \qquad i = 1, 2, \cdots, N \end{aligned} . \tag{9.2.22}$$

If

$$f_m - g_m > \varepsilon, \tag{9.2.23}$$

go to next step. Otherwise, print out the ε-optimal solutions $u_k^* \approx u_k^{m+1}$ and the plant output $h_{k+1}^* \approx Cx_{k+1}^{m+1}$. Then, if $k \geq J$, stop iteration, otherwise, if $k < J$, let $k := k+1$ and go back to Step 1.

4. **Distance check:** If

$$\|u(k)^{m+1} - u(k)^m\|_P + \|v^{m+1} - v^m\|_Q > \varepsilon, \tag{9.2.24}$$

go to next step. Otherwise, print out the ε-optimal solutions $u_k^* \approx u_k^{m+1}$ and the plant output $h_{k+1}^* \approx Cx_{k+1}^{m+1}$. Then, if $k \geq J$, stop iteration, otherwise, if $k < J$, let $k := k+1$ and go back to Step 1.

5. **Constant modification:** Choose suitable $\lambda_{m+1} > 0$ and let $m := m+1$. Go back to Step 2.

9.3. A Numerical Example and Computing Results

The example in this section is the MPC testing example in the Department of Chemical Engineering.

Testing Example:

Consider the transfer function linear system (Morari and Ricker [214]):

$$\begin{bmatrix} h_1(s) \\ h_2(s) \end{bmatrix} = \begin{bmatrix} \frac{12.8e^{-s}}{16.7s+1} & \frac{-18.9e^{-3s}}{21.0s+1} \\ \frac{6.6e^{-7s}}{10.9s+1} & \frac{-19.4e^{-3s}}{14.4s+1} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix} \tag{9.3.1}$$

Its corresponding state-space systems are those in (9.2.6) and (9.2.7) with

$$A = \begin{bmatrix} a & b & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c & d & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \tag{9.3.2}$$

$$B = \begin{bmatrix} 1.0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1.0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad C^T = \begin{bmatrix} 0 & 0 \\ 0.7440 & 0 \\ -0.6788 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0.5786 \\ 0 & -0.5449 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -0.8789 & -1.3015 \\ 0.8199 & 1.2410 \end{bmatrix}, \tag{9.3.3}$$

where

$$a = 1.8542, \quad b = -0.8593, \quad c = 1.8864, \quad d = -0.8895. \tag{9.3.4}$$

The other parameters in (9.2.5) and (9.2.8) are

$$P_i = \begin{bmatrix} 10 & 0 \\ 0 & 12 \end{bmatrix}, \quad Q_i = \begin{bmatrix} 15 & 0 \\ 0 & 15 \end{bmatrix}, \quad r_{k+i} = \begin{bmatrix} 0.8 \\ 0 \end{bmatrix}, \tag{9.3.5}$$

$U_i = [-0.2, 1] \times [-0.2, 1]$, $W_i = \mathbb{R}^2$, and $V_i = \mathbb{R}^2$. The initial data are $u_i^0 = [0.01, 0.01]^T$ and $v_i^0 = [0, 0]^T$. The stopping bound $\varepsilon = 10^{-7}(2n_s + n_c + n_o)N$.

(1) Comparison with the MATLAB MPC Control Toolbox.

The duration of simulation is $J = 200$.

In the following table, 'FBS' indicates the algorithm 9.2, 'T.Box' represents the MPC Control Toolbox in MATLAB, 'time' is the total second of computing time.

The purpose of the comparison is to see how the computing time changes for both methods when the prediction horizon increases. We are not comparing the solutions here.

Table 9.1. Computing Results of Algorithm 9.2 and MPC Toolbox

N	FBS λ	FBS time	T.Box time	FBS λ	FBS time
50	0.5	6.7198	2.2667	0.3	9.0631
100	0.5	13.6425	10.1500	0.3	16.8770
200	0.4	25.4433	73.9167	0.3	30.1428
300	0.4	38.4788	283.4167	0.3	44.6052
400	0.4	51.3230	716.0167	0.3	53.9865
500	0.4	57.3683	1426.9	0.3	66.7203
600	0.4	69.5302	2455.2	0.3	80.9612
700	0.4	81.3750		0.3	93.8971
800	0.4	92.7659		0.3	107.9983
900	0.4	105.6901		0.3	122.4002
1000	0.4	116.0659		0.3	132.1192
5000	0.3	535.2189			
10000	0.2	1337.856			

(2). The influence of step sizes for N=100 and J=200.

Table 9.2. Influence of Step Sizes for Algorithm 9.2.

λ/time	λ/time	λ/time	λ/time	λ/time	λ/time	λ/time
0.1/40.8	0.2/22.3	.25/19.6	0.3/16.9	0.4/14.2	0.5/13.6	0.6/div.

In this table, 'div.' means *diverge.*

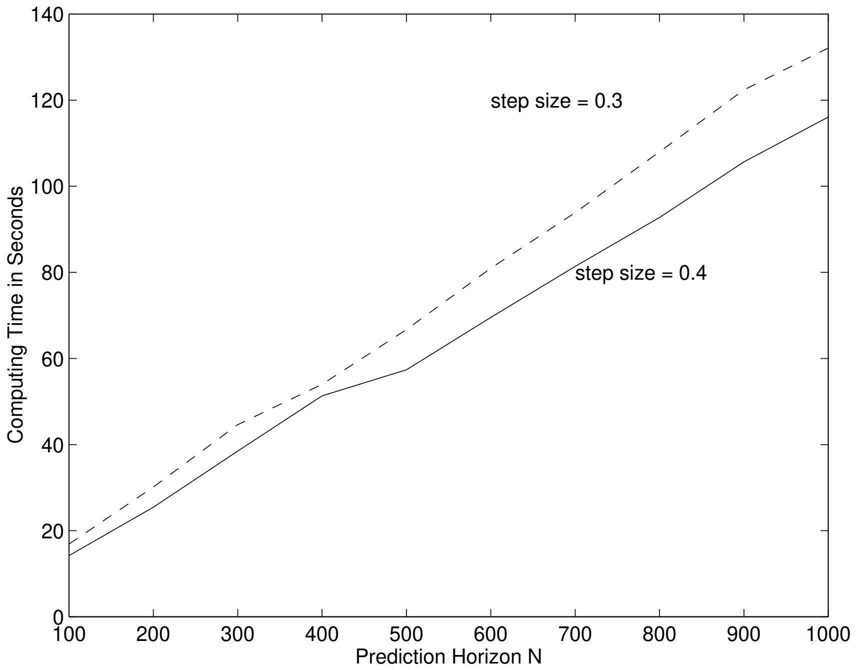

Figure 9.3: Computing Time of Alg. 9.2 for Fixed Step Size

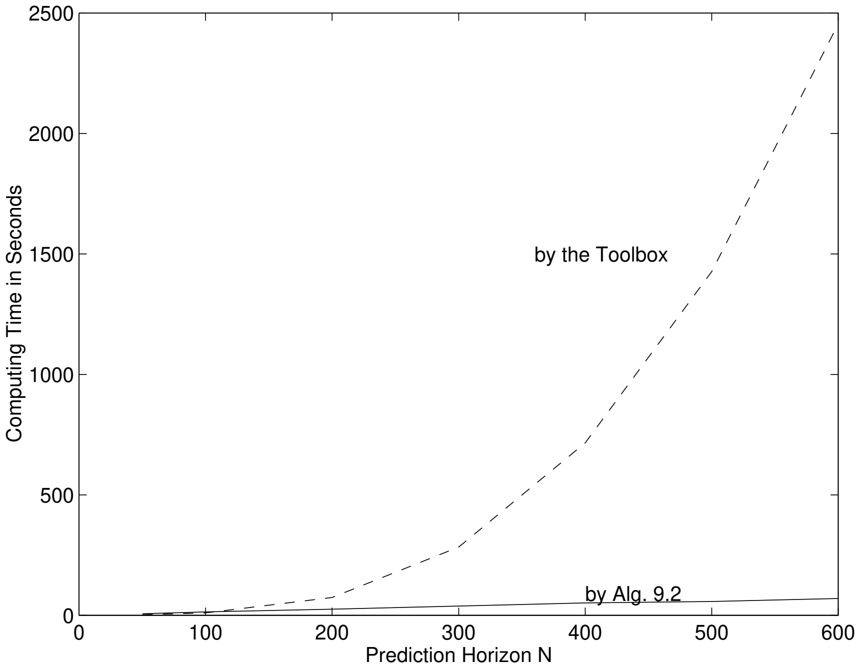

Figure 9.4: Comparison of Alg. 9.2 and the Toolbox

9.4. Applications to Variational Inequalities

Let Z be a nonempty, closed and convex set in $\mathbb{R}^n$, and $F : Z \to \mathbb{R}^n$ be a continuous and monotone mapping on Z. Consider the *variational inequality* problem of finding a $z^* \in Z$ such that

$$\langle F(z^*), z - z^* \rangle \geq 0, \qquad \forall z \in Z. \tag{9.4.1}$$

The variational inequality problem (9.4.1) has numerous applications in the mathematical and engineering sciences, for instance in differential equations, nonlinear optimization, traffic assignment, equilibrium problems in games, economics, and saddle-point problems. [See for example Baiocchi and Capelo (1984), Bertsekas and Gafni (1982), Bertsekas and Tsitsiklis (1989), Cohen and Chaplais (1988), Cottle, Giannessi, and Lions (1980), Dafermos (1990), Gabay (1983), Harker (1991), Kinderlehrer and Stampcchia (1980), Luo and Tseng (1992), Marcotte (1991), Patriksson (1993), Tseng (1991, 1990).]

For the problem (9.4.1), many theoretical discussions have been done and quite a few numerical methods have been suggested so far. Besides the articles and books in last paragraph, interested person may refer to Aganagic (1978), Al-Khayyal and Kyparisis (1991), Auchmuty (1989), Bakušinskiĭ and Polyak (1974), Browder (1966), Bruck (1975, 1977), Dafermos (1983), Fang and Peterson (1982), Glowinski, Lions, and Trémolières (1981), Hammond and Magnanti (1987), Harker (1988), Harker and Pang (1990), Larsson and Patriksson (1994), Marcotte (1986), Marcotte and Dussault (1989), Noor (1975, 1987, 1991, 1992), Patriksson (1994), Rockafellar (1969), Smith (1984), Solodov and Tseng (1994), Taji, Fukushima, and Ibaraki (1993), Zhu and Marcotte (1993).

Assume that the problem (9.4.1) has a solution. [Sufficient conditions for the existence of a solution to (9.4.1) can be found e.g. in Kinderlehrer and Stampcchia (1980), Glowinski, Lions, and Trémolières (1981), Harker and Pang (1990).] Let $H_k : Z \to \mathbb{R}$ be continuous and strictly monotone operators and $F^k = F_1^k + F_2^k$ be splitting strategies such that F_1^k and F_2^k are continuous and maximal monotone for all $k \geq 0$. If $\bar{z}_k$ is a solution of the following variational inequality problem:

$$\langle F^k(\bar{z}_k), z - \bar{z}_k \rangle \geq 0, \qquad \forall z \in Z, \tag{9.4.2}$$

then, we always assume that there exists a solution z^* of (9.4.1) such that $\bar{z}_k \to z^*$.

Consider the following algorithm to solve (9.4.1) now:

Algorithm 9.4 (without line search)

0. **Initial input:** Input an initial point $z^0 \in Z$, initial $\lambda_0 > 0$, an error bound $\varepsilon > 0$, an initial operator H_0, and an initial splitting $F^0 = F_1^0 + F_2^0$. Let $k := 0$.
1. **EFBS iteration:** Given an $z^k \in Z$, compute a new iterate $z^{k+1} \in Z$ such that

$$\langle H_k(z^{k+1}) + \lambda_k[F_2^k(z^{k+1}) + F_1^k(z^k)] - H_k(z^k), z - z^{k+1}\rangle \geq 0, \quad \forall z \in Z. \tag{9.4.3}$$

2. **Convergence check:** If z^k solves (9.4.3) in the error limit ε, or z^{k+1} is acceptable as a solution, for example

$$\langle z^{k+1} - z^k, H^k(z^{k+1}) - H_k(z^k)\rangle \leq \varepsilon^2, \tag{9.4.4}$$

   stop iteration and output the ε-optimal solution $z^* \approx z^{k+1}$. Otherwise, go to next step.
3. **Constant modification:** Choose suitable $\lambda_{k+1} > 0$, H_{k+1}, and $F^{k+1} = F_1^{k+1} + F_2^{k+1}$. Then let $k := k+1$ and go back to Step 1.

I call (9.4.3) an EFBS iteration because it is equivalent to

$$0 \in H_k(z^{k+1}) + \lambda_k[F_2^k(z^{k+1}) + F_1^k(z^k)] - H_k(z^k) + \mathcal{N}_Z(z^{k+1}), \tag{9.4.5}$$

or equivalently,

$$z^{k+1} = [H_k + \lambda_k(F_2^k + \mathcal{N}_Z)]^{-1}[H_k - \lambda_k F_1^k](z^k) \tag{9.4.6}$$

since $\lambda\mathcal{N}_Z = \mathcal{N}_Z$ if $\lambda > 0$.

Some special cases and equivalent steps of the above algorithm are given in the following.

The algorithm class of Tseng

Let $H_k \equiv D$, $F_1^k = F^k \equiv F$, and $\lambda_k \equiv 1$, where D is a $n \times n$ positive definite matrix. Then, Algorithm 9.4 becomes the *asymmetric projection algorithm* in Tseng [314].

The algorithm class of Patriksson

Let $F_1^k = F^k \equiv F$ and $\lambda_k \equiv 1$. Then, the EFBS iteration in Algorithm 9.4 becomes the search direction generation in the *cost approximation algorithm* with z^{k+1} replacing by a y^k in Patriksson [242].

The algorithm class of Cohen

Suppose $\varphi : Z \to \mathbb{R}$ be a strongly convex function in C^1 on Z. Let $H_k = \nabla\varphi$ and $F_1^k = F^k \equiv F$. Then, Algorithm 9.4 is basically the same as the *auxiliary problem principle* in Cohen [53] with the following subproblem for determining z^{k+1}:

$$\min_{z \in Z} \left\{ \varphi(z) + \left[\lambda_k F(z^k) - \nabla\varphi(z^k)\right]^T z \right\} \tag{9.4.7}$$

The algorithm class of Noor

Let $F_1^k = F^k \equiv F$, $H_k \equiv \Phi$, and $\lambda_k \equiv \lambda$, where $\lambda > 0$ and $\Phi : Z \to \mathbb{R}^n$ is continuous and strongly monotone. Then, Algorithm 9.4 becomes the *general algorithm* in Noor [226].

The algorithm class of Dafermos

Assume that $g : Z \times Z \to \mathbb{R}^n$ be a continuous mapping satisfying
(1) $g(z, z) = F(z), \qquad \forall z \in Z.$
(2) $\nabla_z g(z, w)$ is symmetric and positive definite for any fixed $z, w \in Z$.
Given $z^k \in Z$, an *iterative scheme* for z^{k+1} in Dafermos [64] is to solve the variational inequality subproblem

$$g\left(z^{k+1}, z^k\right)^T \left(z - z^{k+1}\right) \geq 0, \qquad \forall z \in Z. \tag{9.4.8}$$

Let $F_1^k = F^k \equiv F$, $\lambda_k \equiv 1$, and $H_k(z) = g(z, z^k) - F(z^k)$ in (9.4.3). Then, the variational inequality subproblems (9.4.3) and (9.4.8) are exactly same problems.

The algorithm class of Wu et al.

Assume that $h : Z \times Z \to \mathbb{R}$ be a continuous function of the form $h(z, w)$, strictly convex and in C^1 on Z with respect to z. Furthermore, suppose $\nabla_z h(z, z) := \nabla_z h(z, w) \mid_{w=z} = 0$. Given z^k, the subproblem for z^{k+1} in the general descent framework of Wu, Florian, and Marcotte [326] is

$$\min_{w \in Z} \left\{ F(z^k)^T w + h(w, z^k) \right\}. \tag{9.4.9}$$

Let $F_1^k = F^k \equiv F$, $\lambda_k \equiv 1$, and $H_k(w) = \nabla_w h(w, z^k)$ in (9.4.3). Then, the subproblems (9.4.3) and (9.4.9) are equivalent.

The general convergence analysis could be done similar to Chapter 5. Here, as corollaries of Theorem 5.6.5 and Theorem 5.6.6, we just give the following:

Theorem 9.4.1 (Quadratic convergence). *Suppose $F = F_1^k + F_2^k$, with $\{F_1^k\}$ uniformly strongly monotone with α and uniformly Lipschitz continuous with M on Z. For given $\lambda_k \equiv 1$, if F_1^k is twice differentiable and $H_k = \nabla F_1^k(z^k)$, then, the $\{z^k\}$ from (9.4.3) will converge to the unique solution z^* of the variational inequality (9.4.1) at square rate, when z^0 is near z^*.*

Theorem 5.6.6 (Superlinear convergence). *Suppose $F = F_1^k + F_2^k$ in which, on Z, $\{F_1^k\}$ is uniformly strongly monotone with α and uniformly Lipschitz continuous with M. If F_1^k is continuously differentiable and z^* is the solution of (9.4.1), then when $\{\nabla F_1^k(z^*)\}$ converges and we choose H_k with $H_k - \nabla F_1^k(z^*) \to 0$, then there must exist step sizes $\{\lambda_k\}$ with $\lambda_k \to 1$, such that the $\{z^k\}$ from (9.4.3) superlinearly converges to z^* for any initial z^0.*

9.5. Summary

In this dissertation, we expanded the ordinary forward-backward splitting method for finding a zero point of a maximal monotone operator to very generalized FBS methods and developed the convergence theory for both ordinary and extended methods. We applied our methods to problems of extended linear-quadratic programming, optimal control, saddle points, multistage stochastic programming, process control in chemical engineering, and variational inequalities in the previous sections and chapters. The numerical experiments of our methods and comparisons with some currently using methods show that our methods can fully utilize the decomposition structure of the problems we tested and converge much faster then the interior point methods and existing MPC toolbox in MATLAB for the large-scale cases of those problems. These numerical tests show the potential ability of the FBS methods to solve large-scale optimization problems.

Besides the above problems, the forward-backward splitting techniques in this dissertation may also be applied to many other probelms, especially to large-scale problems, such as problems of nondifferentiable optimization ([45], [54], [59], [220], [242]), problems of unconstrained and constrained optimization ([23], [51], [58], [114], [137], [242]), problems of partial differential equations ([11], [81], [82], [107], [243]), problems in nonlinear mechanics ([117]), complementarity problems ([60], [61], [184], [232]), and so on. In one word, the FBS techniques may be applied to any problems which could be transformed to zero point finding problems of monotone operators. However, the maximal monotonicity requirement of operators could be replaced by the local monotonicity of the operators or the maximal monotonicity of sequentially approaching operators under some mild assumptions. We shall leave this subject to our later research papers.

Although we showed the theoretic and numerical advantages of the FBS techniques, there are still some weak points which will influence the convergence and rates of the FBS algorithms. The key one is the automatic choices of the step sizes λ_k. Currently, we give the λ_k before we implement the algorithms. Hence, we have to estimate the Lipschitz constant of the T_1 and the strong monotone modulus of the operator. For some complicated problems, the estimation might be hard or might not be economic. This will restrict the applications of the algorithms. Thus, the automatic adjustments of the step sizes is a valuable research subject for these algorithms. This weak point might be overcome through the line searches through the merit functions of Patriksson [242]. For the special case of convex minimization, Cohen [51] successfully replaced the step size choices (through setting $\lambda_k = 1$) by line searches. However, it needs to be verified for general cases.