Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

MARC C. STEINBACH

Robust Process Control by Dynamic Stochastic Programming

ZIB-Report 04-20 (May 2004)

ROBUST PROCESS CONTROL BY DYNAMIC STOCHASTIC PROGRAMMING

MARC C. STEINBACH

ABSTRACT. Unnecessarily conservative behavior of standard process control techniques can be avoided by stochastic programming models when the distribution of random disturbances is known. In an earlier study we have investigated such an approach for tank level constraints of a distillation process. Here we address techniques that have accelerated the numerical solution of the large and expensive stochastic programs by a factor of six, and then present a refined optimization model for the same application.

0. INTRODUCTION

In [3] we have proposed a multistage stochastic programming extension of model predictive control techniques to allow for the explicit incorporation of uncertainty via scenario trees. As a concrete example, we have studied the separation of methanol and water in a continuously running process where the inflow into a buffer tank is assumed to be random with known distribution. The objective is to minimize the total energy consumption over a given planning horizon under hard level constraints at the buffer tank; an alternative approach with probabilistic level constraints has been studied in [5]. The model investigated in [3] is a tracking problem that decouples uncertainty from the process dynamics: the stochastic on-line problem minimizes the expected quadratic deviation from a nominal deterministic operation profile that is determined off-line. For the numerical solution of the linear-quadratic multistage stochastic programs we rely on a primal-dual interior method in combination with a *tree-sparse KKT solver* for the highly structured Newton step subproblems [10].

The current paper presents two techniques that are crucial for maximum efficiency in solving the stochastic on-line problem (and thus for future application to models with higher complexity): a problem reformulation enforcing relatively complete recourse, and a software tool generating custom code for the tree-sparse KKT solver, given a specific problem class. Finally we discuss a natural replacement of the tracking approach by an integrated optimization of uncertainty and process dynamics.

1. STOCHASTIC TRACKING MODEL

Consider a planning horizon of T periods in discrete time, $t = 0, 1, \ldots, T$, and a scenario tree with vertex set V. Let $L_t \subseteq V$ denote the level set of nodes at time t and $L \equiv L_T$ the set of leaves. Further let $0 \in L_0$ denote the root, $j \in L_t$ the "current" node, $i \equiv \pi(j) \in L_{t-1}$ its unique predecessor (if $t > 0$), and $S(j) \subseteq L_{t+1}$ its set of successors. We also need sets $V^* = V \setminus \{0\}$ and $V^t = \bigcup_{\tau=0}^{t} L_\tau$, and finally the node probabilities $p_j > 0$, $j \in V$.

As in [3], let ξ_t denote the tank inflow volume during $(t-1, t)$, $\hat{f}_t$ the target extraction during $(t, t+1)$, and f_t the actual extraction. The decision variables, $y_t = (x_t, u_t)$, consist of the tank filling volume x_t as state (with desired final value $\hat{x}_T$), and the local tracking

2000 *Mathematics Subject Classification.* 90C15, 90C06, 93C15, 93C95.

Key words and phrases. Process control, random disturbance, multistage stochastic program, relatively complete recourse, tree-sparse NLP.

Extended version of a contribution to the GAMM Annual Meeting 2004, March 21–27, Dresden, Germany.

error as control, $u_t = f_t - \hat{f}_t$. These quantities are random variables with realizations ξ_j, f_j, y_j in the scenario tree representation; only $\hat{f}_t$ is deterministic. Letting $h_j = \xi_j - \hat{f}_t$, the tracking model may then be written

$$\text{(1)} \qquad \underset{y}{\text{minimize}} \quad \sum_{j \in V \setminus L} \frac{1}{2} p_j u_j^2$$

$$\text{(2)} \qquad \text{subject to} \quad x_j = x_i - u_i + h_j \ \forall j \in V,$$

$$\text{(3)} \qquad x_j \in [x^-, x^+] \ \forall j \in V^*,$$

$$\text{(4)} \qquad u_j \in [u_t^-, u_t^+] \ \forall j \in V \setminus L,$$

$$\text{(5)} \qquad \sum_{k \in S(j)} \frac{p_k}{p_j} x_k = \hat{x}_T \ \forall j \in L_{T-1} \quad \text{(cycling constraint).}$$

Here $x_0 = h_0$ is fixed, and the cycling constraint prescribes $\hat{x}_T$ as *conditional expectation* of the final tank filling. It is not possible to end with $\hat{x}_T$ in every scenario since the extraction f_{T-1} must be chosen at time $T-1$ whereas the inflow ξ_{T-1} is only known at time T. In [3, §2.2] we have shown that u_{T-1} is uniquely determined by the cycling constraint so that the final stage can be eliminated completely. Letting $V_c := V^{T-1}$, $L_c := L_{T-1}$, and redefining x_j appropriately for $j \in L_c$, one obtains

$$\text{(6)} \qquad \underset{y}{\text{minimize}} \quad \sum_{j \in V_c \setminus L_c} \frac{1}{2} p_j u_j^2 + \sum_{j \in L_c} \frac{1}{2} p_j x_j^2$$

$$\text{(7)} \qquad \text{subject to} \quad x_j = x_i - u_i + h_j \ \forall j \in V_c,$$

$$\text{(8)} \qquad x_j \in [x_t^-, x_t^+] \ \forall j \in V_c^*,$$

$$\text{(9)} \qquad u_j \in [u_t^-, u_t^+] \ \forall j \in V_c \setminus L_c.$$

2. Relatively Complete Recourse

A multistage stochastic program is said to have *relatively complete recourse* if every feasible t-stage trajectory $y^t = (y_j^t)_{j \in V^t}$ can be extended to a feasible (T-stage) trajectory. In other words, if all local constraints up to time t can be satisfied, then it is possible to satisfy the constraints in higher stages $t+1, \dots, T$ as well: a trajectory y^t cannot "get stuck" later. The tracking problem under consideration does not have this desirable property, due to the cycling constraint at $t = T$.

In what follows, we construct a reformulation that does have relatively complete recourse (which is possible for every linearly constrained multistage stochastic program [8]). To this end, we first perform an outward recursion over the tree to define sets Y_j^{out} that contain the components y_j^t of every feasible t-stage trajectory. In an inward recursion we then determine *induced constraints* to obtain sets Y_j^{in} whose elements y_j have feasible extensions to all immediate successor nodes $k \in S(j)$. This is sufficient for relatively complete recourse since the tracking problem is Markovian: x_j depends only on the immediately preceding decision y_i. Given initial feasible sets

$$\text{(10)} \qquad Y_j = X_j \times U_j = [x_j^-, x_j^+] \times [u_j^-, u_j^+]$$

(where $X_0 = \{x_0\}$ and $U_j = \{0\}$ for $j \in L$), let $X_0^{\text{out}} := X_0$, $Y_0^{\text{out}} := Y_0$, and for $j \in V^*$ define recursively the attainable feasible sets

$$\text{(11)} \qquad X_j^{\text{out}} := X_j \cap (X_i^{\text{out}} - U_i + \xi_j), \quad Y_j^{\text{out}} := X_j^{\text{out}} \times U_j.$$

Here we use standard notation for sums and differences of subsets of a vector space,

$$\text{(12)} \qquad A \pm B := \{\, a \pm b \colon a \in A \text{ and } b \in B \,\},$$

$$\text{(13)} \qquad A \pm b := A \pm \{b\}, \quad a \pm B := \{a\} \pm B.$$

For $j \in L$, let now $X_j^{\text{in}} := X_j^{\text{out}}$, $U_j^{\text{in}} := U_j$, and $Y_j^{\text{in}} := Y_j^{\text{out}}$, to derive the induced feasible sets recursively for $j \in V \setminus L$,

$$X_j^{\text{in}} := X_j^{\text{out}} \cap (U_j + D_j^{\text{in}}), \quad U_j^{\text{in}} := U_j \cap (X_j^{\text{in}} - D_j^{\text{in}}), \tag{14}$$

$$Y_j^{\text{in}} := Y_j^{\text{out}} \cap \{\, (x_j, u_j) \colon x_j - u_j \in D_j^{\text{in}} \,\}, \tag{15}$$

where D_j^{in} denotes induced sets of feasible differences $x_j - u_j$,

$$D_j^{\text{in}} := \{\, x_j - u_j \colon x_j - u_j + \xi_k \in X_k^{\text{in}} \ \forall k \in S(j) \,\} = \bigcap_{k \in S(j)} (X_k^{\text{in}} - \xi_k). \tag{16}$$

Calculating the induced constraints for the tracking problem is particularly effective and inexpensive because of the specific structure: the (scalar) states and controls are linked via simple dynamic equations, $x_j = x_i - u_i + h_j$, they have lower and upper bounds, the initial state x_0 is fixed, and the cycling constraint yields narrow feasible intervals for the final states.

The desired reformulation of the tracking problem is obtained simply by replacing the inequality constraints $y_j \in Y_j$ with $y_j \in Y_j^{\text{in}}$. To see that it has indeed relatively complete recourse, nothing needs to be shown for $j \in L$. For $i \in V \setminus L$ and all $j \in S(i)$, the transition equation of node j maps Y_i^{in} into X_j^{in} by definition of D_i^{in}. But X_j^{in} contains precisely the local states x_j for which a feasible extension exists,

$$X_j^{\text{in}} = \{\, x_j \in X_j^{\text{out}} \colon \exists u_j \in U_j \colon x_j - u_j \in D_j^{\text{in}} \,\}. \tag{17}$$

This proves relatively complete recourse. Moreover, feasibility of the problem is obviously equivalent with $X_0^{\text{in}} \neq \emptyset$ since $X_0^{\text{in}} \subseteq X_0 = \{x_0\}$. This condition is immediately checked in the reformulation.

The construction shows that the principal restriction here is on the states, $x_j \in X_j^{\text{in}}$. It can further be shown that U_j^{in} contains precisely the local controls u_j that are instrumental in extending some $x_j \in X_j^{\text{in}}$ to a feasible trajectory,

$$U_j^{\text{in}} = \{\, u_j \in U_j \colon \exists x_j \in X_j^{\text{in}} \colon x_j - u_j \in D_j^{\text{in}} \,\}. \tag{18}$$

Thus $(x_j, u_j) \in Y_j^{\text{in}}$ implies $x_j \in X_j^{\text{in}}$ and $u_j \in U_j^{\text{in}}$, so that

$$Y_j^{\text{in}} = (X_j^{\text{in}} \times U_j^{\text{in}}) \cap \{\, (x_j, u_j) \colon x_j - u_j \in D_j^{\text{in}} \,\}. \tag{19}$$

In practice we wish to have simple bounds only, so we do not work with Y_j^{in} but rather replace X_j and/or U_j with X_j^{in} and/or U_j^{in}.

To see which problem formulation performs best, we combined X_j or X_j^{in} with U_j or U_j^{in} and tested all four combinations. A range of about 1000 test problems was generated by considering the setting in [3, §5] with various target profiles, random distributions, and inflow and extraction bounds. The discretization has 8 time periods and a scenario tree with a branching factor of 5, yielding $5^7 = 78125$ scenarios, 117187 variables, and 97656 constraints (plus bounds) for problem (6)–(7).

Interestingly, a speed-up by a factor of three was achieved with X_j^{in} and U_j, whereas the performance was roughly equal with all other combinations—even X_j^{in} and U_j^{in}. The speed-up is caused by a reduced number of interior point iterations. On the test problems, the number varies between 21 and 99 with the original formulation and between 6 and 36 with the reformulation. The solution times are reduced from 14.7–68.7 seconds to 4.2–25.3 seconds, at 0.7 seconds per iteration on a 1.5 GHz PC. The reformulation with relatively complete recourse thus leads to significantly faster convergence.

3. Code Generator

The tree-sparse KKT solver used in [3] implements block-level operations based on the BLAS and LAPACK linear algebra software libraries. Such an implementation is only moderately efficient when the blocks are very small (as is the case here), and it becomes

increasingly inefficient for sparse blocks of increasing size. Although a manual adaptation of the KKT code for a specific problem structure is straightforward, it is also tedious and error-prone—and hence impractical. To overcome these limitations, a software tool has been developed in the diploma thesis [7] that generates custom code based on a high-level description of the specific KKT structure. This is possible since the tree-sparse approach leaves very little choice for pivot selection, and only on the sub-block level. On the block level it employs a fixed elimination scheme that fully exploits the generic tree-sparse structure created by a natural classification of the constraints [10].

The code generator requires as input data the overall problem type (implicit, outgoing control, or incoming control; see [10]), the dimensions of each matrix block, the entry positions within each bock, and a classification of each individual entry. The classification indicates whether entry values are fixed per problem class (*static* entry) or problem instance (*constant* entry), or whether the values may differ among level sets L_t (*time dependent* entry) or nodes j (*node dependent*, or *stochastic* entry). The scenario tree is mostly independent of the structural specification, except that a minimum (or fixed) number of stages is given with the block specifications. (Currently the dimensions and entry positions and types must be identical across a stage.)

From these data, the code generator determines the positions and classification of the fill-in, then creates data structures and element-oriented (sparse) or block-oriented (dense) node operations, as appropriate, and finally generates four output files. The four files contain code implementing the KKT vector, matrix, inverse, and common structural information, respectively, as C++ class templates. The code is human-legible and can be edited manually. As we are working with very large scenario trees, the data structures are chosen so as to minimize memory requirements. For further details see [7].

The code generator was used to generate custom code for the tracking model, which was then slightly improved manually. With little effort, we have thus obtained an additional speed-up by a factor of two: computation times for the test problems are now reduced to 2.05–12.5 seconds, or 0.35 seconds per iteration.

4. Integrated Stochastic Process Model

A natural replacement of the tracking model discussed so far is an integrated approach where the dynamics of the distillation process are included in the stochastic optimization model, so that the expected energy consumption can be minimized rather than the expected deviation from a predetermined operation profile. This is clearly a substantial benefit which, however, has to be paid for by running a copy of the process model in every node of the scenario tree. As an example, consider an extended flash unit system consisting of the tank, a reboiler and total condenser; see Fig. 1 and [2, 5, 6]:

Tank:

$$\frac{d}{dt}M_F = \xi - F, \tag{20}$$

Reboiler:

$$0 = F - V_R - B, \tag{21}$$

$$M_R^L \frac{d}{dt}X_{1R} = FZ_1 - V_R Y_{1R} - BX_{1R}, \tag{22}$$

$$M_R^L \frac{d}{dt}H_R^L = FH_F - V_R H_R^V - BH_R^L + Q, \tag{23}$$

$$H_R^L = X_{1R}h_1^L(T_R) + X_{2R}h_2^L(T_R), \tag{24}$$

$$H_R^V = Y_{1R}h_1^V(T_R) + Y_{2R}h_2^V(T_R), \tag{25}$$

$$Y_{iR} = \gamma_i(T_R, X_{1R}, X_{2R})X_{iR}P_i^{vp}(T_R)/P_R, \quad i = 1,2, \tag{26}$$

$$1 = X_{1R} + X_{2R}, \qquad 1 = Y_{1R} + Y_{2R}, \tag{27}$$

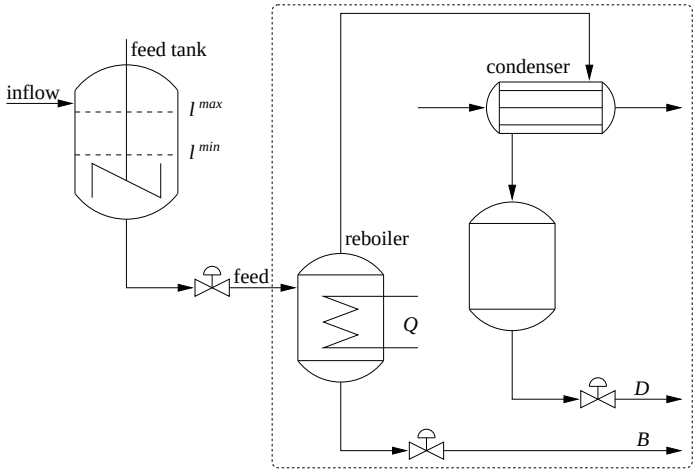

FIGURE 1. Flowsheet of a flash unit system

Condenser:

$$P_R - P_C = \frac{1}{2}\zeta_C \frac{(Y_{1R}M_1 + Y_{2R}M_2)(Y_{1R}\bar{V}_1^V(T_R) + Y_{2R}\bar{V}_2^V(T_R))}{(A_C^{FA})^2} V_R^2, \tag{28}$$

$$0 = V_R - D, \tag{29}$$

$$M_C^L \tfrac{d}{dt} X_{1C} = V_R Y_{1R} - D X_{1C}. \tag{30}$$

Here ξ, F, B, D, V_R denote molar liquid and vapor flow rates (tank inflow, feed, bottom, distillate, reboiler), Q is the reboiler heating power, M_F, M_R^L, M_C^L are molar liquid holdups (tank, reboiler, condenser), $X_{1R}, X_{1C}, Y_{1R}, Z_1$ and X_{2R}, Y_{2R} are the liquid and vapor mole fractions of methanol and water, respectively, M_1, M_2 are their respective molar masses, $H_F, H_R^L, H_R^V, h_i^L, h_i^V$ are molar liquid and vapor enthalpies, P_R, P_C are pressures, T_R is the reboiler temperature, and the remaining symbols denote constants and (empirical) functions. The model is valid if the inflow mixture has constant composition and temperature, the molar holdups M_R^L, M_C^L are constant, and the condenser reflux flow L_C is zero.

After eliminating $B, D, X_{2R}, Y_{2R}, H_R^L, H_R^V, P_R$ in a stable way, the process model can be formulated with four differential variables, $x = (M_F, X_{1R}, X_{1C}, T_R)$, two algebraic variables, $z = (V_R, Y_{1R})$, and two (physical) control variables $w = (F, Q)$, yielding a semi-implicit DAE of index one with stochastic disturbance in the first differential equation (the feed flow rate),

$$B(x)\tfrac{d}{dt}x = f(x, z, w) + \xi e_1, \qquad B(x) \text{ nonsingular}, \tag{31}$$

$$g(x, z) = 0, \qquad \frac{\partial g}{\partial z}(x, z) \text{ nonsingular}. \tag{32}$$

With piecewise constant control profiles and a suitable discretization of the DAE, the full model is then formulated as a nonlinear multistage stochastic program on a scenario tree (a tree-sparse NLP), which can be treated by suitable SQP or interior methods in combination with a tree-sparse KKT solver.

Observe first that collocation is *not* well suited for the DAE discretization: one cannot afford fine time grids since scenario trees grow exponentially with the number of stages. We therefore suggest a multiple shooting discretization with relaxed initial value problems to account for inconsistent iterates [1, 9]. This means that local DAE initial value problems of the following form have to be integrated numerically from node i at physical time τ_i to

node j at physical time τ_j, for every $j \in V^*$:

$$(x,z)(\tau_i) = (x_i, z_i), \tag{33}$$

$$B(x)\tfrac{d}{dt}x = f(x,z,w) + \xi e_1, \tag{34}$$

$$g(x,z) = g(x_i,z_i)e^{-\beta(\tau-\tau_i)}. \tag{35}$$

Thus one obtains values $x(\tau_j-) = F_j(x_i,z_i,w_i;\xi_j)$ that are needed for the continuity conditions, $x(\tau_j-)-x_j = 0$, and values $z(\tau_j-)$ that are thrown away: discretized algebraic equations are formulated pointwise as $g(x_j,z_j) = 0$, $j \in V$.

Defining the numerical control variables as $u_j := (z_j, w_j)$, the complete tree-sparse NLP may be written:

$$\underset{(x,u)}{\text{minimize}} \quad \sum_{j\in V\setminus L} p_j(\tau_j-\tau_i)e_4^* u_j \tag{36}$$

$$\text{subject to} \quad x_0 = \hat{x}_0, \tag{37}$$

$$x_j = F_j(x_i,u_i;\xi_j)\ \forall j \in V^*, \tag{38}$$

$$g(x_j,u_j) = 0\ \forall j \in V, \tag{39}$$

$$x_j \in [x_j^-, x_j^+]\ \forall j \in V^*, \tag{40}$$

$$u_j \in [u_j^-, u_j^+]\ \forall j \in V, \tag{41}$$

$$\sum_{k\in S(j)} \frac{p_k}{p_j} e_1^* x_k = \hat{M}_{FT}\ \forall j \in L_{T-1}. \tag{42}$$

From the tracking model (1)–(5), this problem differs mainly in the linear objective and the nonlinear transition equations. As before, the cycling constraint can be used to determine a control component: the feed extraction u_{j3}, $j \in L$. Although the final stage is still needed for the process model here (and thus not completely eliminated), it becomes *deterministic*. This means that no further branching is required in the scenario tree, yielding a substantial reduction in size. Moreover, relatively complete recourse can be achieved with respect to the tank level constraints, and their feasibility can be checked. The remaining constraints are not amenable to such preprocessing, however, due to the nonlinear process model. In particular, feasibility of the tank level constraints does not imply feasibility of the entire process (although the converse holds).

Assuming the same discretization as before, the integrated model has 78125 scenarios, 175781 nodes, 1328123 variables, and 976561 constraints. Even if the computational effort increases by a factor of 100, the stochastic NLP can be solved within 10–12 minutes on a 3 GHz PC—certainly an acceptable figure for real-time application to a relatively slow separation process.

5. Conclusion

We have considered multistage stochastic programming as an enhanced approach to robust process control when the distributions of random disturbances are known. The results demonstrate that this approach can yield very quick responses if the process dynamics are excluded from the real-time optimization by means of a tracking approach. Furthermore, our discussion indicates that the stochastic approach, although rather expensive in comparison to deterministic model predictive control, can even be feasible for an integrated treatment of uncertainty and moderately complex process dynamics.

Acknowledgement

This research has been supported by the Deutsche Forschungsgemeinschaft (DFG) under grant STE 566/1-1, within the Research Center *Online Optimization of Large Systems*.

REFERENCES

[1] H. G. BOCK, E. EICH, AND J. P. SCHLÖDER, *Numerical solution of constrained least squares boundary value problems in differential-algebraic equations*, in Numerical Treatment of Differential Equations, K. Strehmel, ed., Teubner Verlagsgesellschaft, Leipzig, 1988, pp. 269–280.

[2] H. A. GARCIA, R. HENRION, P. LI, A. MÖLLER, W. RÖMISCH, M. WENDT, AND G. WOZNY, *A model for the online optimization of integrated distillation columns under stochastic constraints*, Preprint 98-32, DFG Research Center „Echtzeit-Optimierung großer Systeme“, Nov. 1998.

[3] I. GARRIDO AND M. C. STEINBACH, *A multistage stochastic programming approach in real-time process control*, in Grötschel et al. [4], pp. 479–498.

[4] M. GRÖTSCHEL, S. O. KRUMKE, AND J. RAMBAU, eds., *Online Optimization of Large Scale Systems*, Springer, Berlin, 2001.

[5] R. HENRION, P. LI, A. MÖLLER, M. C. STEINBACH, M. WENDT, AND G. WOZNY, *Stochastic optimization for operating chemical processes under uncertainty*, in Grötschel et al. [4], pp. 457–478.

[6] R. HENRION, P. LI, A. MÖLLER, M. WENDT, AND G. WOZNY, *Optimization of a continuous distillation process under probabilistic constraints*, in Grötschel et al. [4], pp. 499–517.

[7] A. HUŢANU, *Code generator for sparse linear algebra in stochastic optimization*, master's thesis, "Politecnica" University of Bucharest, 2002.

[8] R. T. ROCKAFELLAR AND R. J. B. WETS, *Stochastic convex programming: Relatively complete recourse and induced feasibility*, SIAM J. Control Optim., 14 (1976), pp. 575–589.

[9] V. H. SCHULZ, H. G. BOCK, AND M. C. STEINBACH, *Exploiting invariants in the numerical solution of multipoint boundary value problems for DAE*, SIAM J. Sci. Comput., 19 (1998), pp. 440–467.

[10] M. C. STEINBACH, *Tree-sparse convex programs*, Math. Methods Oper. Res., 56 (2002), pp. 347–376.

MARC C. STEINBACH, KONRAD-ZUSE-ZENTRUM FÜR INFORMATIONSTECHNIK BERLIN, DEPARTMENT OPTIMIZATION, TAKUSTR. 7, 14195 BERLIN-DAHLEM, GERMANY

E-mail address: steinbach@zib.de

URL: http://www.zib.de/steinbach