Nonlinear Optimization with Many Degrees of Freedom in Process Engineering

Maame Yaa B. Poku and Lorenz T. Biegler*

Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania 15123

Jeffrey D. Kelly

Honeywell Industrial Solutions, 300 Yorkland Boulevard, Toronto, Ontario M2J 1S1, Canada

Applications of nonlinear optimization problems with many degrees of freedom have become more common in the process industries, especially in the area of process operations. However, most widely used nonlinear programming (NLP) solvers are designed for the efficient solution of problems with few degrees of freedom. Here we consider a new NLP algorithm, IPOPT, designed for many degrees of freedom and many potentially active constraint sets. The IPOPT algorithm follows a primal−dual interior point approach, and its robustness, improved convergence, and computational speed compared to those of other popular NLP algorithms will be analyzed. To demonstrate its effectiveness on process applications, we consider large gasoline blending and data reconciliation problems, both of which contain nonlinear mass balance constraints and process properties. Results on this computational comparison show significant benefits from the IPOPT algorithm.

1. Introduction

Consider a nonlinear programming (NLP) problem with equality constraints and variable bounds; these problems are often classified by variable types at the solution. *Nonbasic* variables are at their bounds at the solution, *basic* variables can be determined uniquely by the equality constraints (with other variables fixed), and the remaining variables, termed *superbasic* variables, represent the degrees of freedom available for optimization.[1] Most popular NLP solvers are designed to consider problems with few superbasic variables. For instance, linear programs and related successive linear programming (SLP) methods assume no superbasics at all. As observed in a number of studies,[2−5] these codes become inefficient when the number of superbasics becomes large. As a result, this study considers a new algorithm tailored for process applications with many degrees of freedom.

Large-scale nonlinear optimization problems tend to arise naturally in the physical sciences and engineering and are becoming more widely used in economics and management sciences. Such problems have become an important part of computer-aided process operations in areas such as optimal control,[2] parameter estimation,[6] data reconciliation,[7] and blending operations.[8−10] Many of these applications lead to NLP problems with many thousands of variables and also many degrees of freedom for optimization. Applicable solvers for these problems need to exploit sparsity and problem structure and usually require second-order information from the optimization model. Moreover, these methods also need to handle computational difficulties such as redundant (i.e., degenerate) constraints and lack of positive definiteness in the reduced Hessian matrix.

In the past 2 decades, the sequential quadratic programming (SQP)[5,11−13] algorithms have proven to be suitable for solving these kinds of problems. Extensions of these algorithms to deal with many degrees of freedom have been developed by Lucia and co-workers,[4,5] Betts and Frank,[14] and Sargent and Ding.[12] SQP solves a nonlinear optimization problem by successively solving a series of quadratic programming subproblems. At each iteration, a quadratic program (QP) is obtained from the nonlinear program through a quadratic approximation of the Lagrange function and a linear approximation of the constraints. This leads to a search direction and a line-search step size, which determines the next iterate.[11] Here, the presence of many inequality constraints can become a bottleneck during the solution of the QPs as a result of the identification of the active set of inequality constraints. Identification of the active set can increase exponentially with increasing problem size, and as a consequence, these algorithms have become less attractive.

Other popular NLP algorithms include reduced-gradient methods such as MINOS[1] and CONOPT,[15] which may not be well suited for problems with many superbasic variables. These methods approximate second-order information using dense quasi-Newton updates in the reduced space, and consequently the computational effort grows polynomially with problem size. Moreover, active set selection is a combinatorial process that may be expensive, especially on degenerate problems. Finally, augmented Lagrangian methods have been adapted to large-scale problems. For instance, LANCELOT incorporates second-order information and a bound-constrained trust region method to deal with negative curvature. These features allow interesting comparisons with SQP methods.

To overcome these limitations, we consider a novel full space barrier (or interior point) approach termed IPOPT. Here a novel filter line-search strategy, which ensures convergence of the barrier problem, is incorporated along with efficient ways to incorporate second-order information. Interior point methods were initially

* To whom correspondence should be addressed. Tel.: (412) 268-2232. Fax: (412) 268-7139. E-mail: lb01@andrew.cmu.edu.

10.1021/ie0341000 CCC: $27.50 © 2004 American Chemical Society
Published on Web 09/14/2004

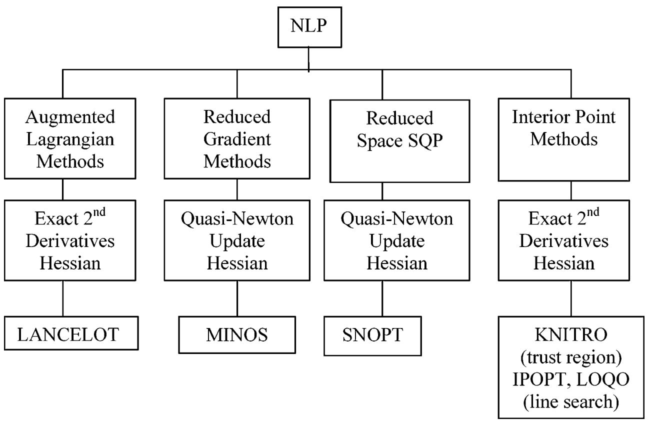
NLP
Augmented Lagrangian Methods
Reduced Gradient Methods
Reduced Space SQP
Interior Point Methods
Exact 2nd Derivatives Hessian
Quasi-Newton Update Hessian
Quasi-Newton Update Hessian
Exact 2nd Derivatives Hessian
LANCELOT
MINOS
SNOPT
KNITRO (trust region) IPOPT, LOQO (line search)

Figure 1. Summary of NLP solvers.

developed by Fiacco and McCormick.[16] Moreover, the past decade has seen intensive research[17−20] in interior point methods for general NLPs after encouraging work by Karmarkar.[21] In particular, the interior point approach overcomes the combinatorial bottleneck of current SQP methods in identifying the active set and appears to be especially well suited to problems with many degrees of freedom. An excellent description and comprehensive analysis of interior point methods can be found in work by Forsgren et al.[22] Lasdon et al. also extend primal−dual interior point procedures to solve small-to-medium size NLPs.[23]

Finally, to facilitate the general purpose calculation of second-order information, a mathematical programming modeling language is essential. In this study, we use AMPL, which provides automatic generation of first and second derivatives of constraint and objective functions.[24] The AMPL modeling language is linked to various state of the art NLP solvers, which can also be accessed over the Internet, and allows the user to choose easily among solvers and options that may improve solver performance.[24]

The objective of this study is to apply a new NLP algorithm to process engineering and economics problems with many degrees of freedom. As representative examples, we consider applications in gasoline blending and data reconciliation. The interior point algorithm used in this study follows a primal−dual interior point approach, and its robustness, improved convergence, and computational speed are compared to those of a number of popular NLP solvers. This study also addresses some further research questions in work done by Lasdon et al.[23] by extending interior point methods to solve large-scale problems. Section 2 gives a background summary of NLP solvers used in this study. The large-scale engineering applications, gasoline blending, and data reconciliation are discussed in section 3. Specific model formulations for these problems are presented in section 4, and numerical results and the solver comparison are given in section 5. Conclusions and directions for future work are presented in section 6.

2. Summary of NLP Solvers

Five other NLP solvers (LANCELOT,[3] MINOS,[1] SNOPT,[11] KNITRO,[17,25] and LOQO[19]) are summarized here and chosen in order to compare the relative efficiency of IPOPT.[20] All of these methods have been linked to AMPL to allow a straightforward comparison and are classified in Figure 1.

IPOPT Solver. To simplify the presentation of the IPOPT primal−dual interior point algorithm,[20] we consider NLPs rewritten with lower bounds of zero. Without loss of generality, the optimization problem (NLP) can be stated as

$$\begin{aligned} &\min f(x) \\ &\text{s.t.} \quad c(x) = 0 \\ &x \geq 0 \end{aligned} \tag{1}$$

We assume the objective function $f(x)$: $R^n \to R$ and the equality constraints $c(x)$: $R^n \to R^m$ with $m < n$ are sufficiently smooth. The bounds are now replaced by a logarithmic barrier term, which is added to the objective term to give

$$\begin{aligned} &\min \varphi_\mu(x) = f(x) - \mu \sum_i \log(x^{(i)}) \\ &\text{s.t.} \quad c(x) = 0 \end{aligned} \tag{2}$$

The barrier method solves a sequence of barrier problems (indexed by ℓ) for decreasing values of μ_ℓ with $\lim_{\ell\to\infty} \mu_\ell = 0$. Under mild assumptions, it can be shown that a sequence of $x^*(\mu_\ell)$ of (approximate) local solutions of eq 2 converges to a local solution of the original NLP (eq 1).[16,22] Since the exact solution $x^*(\mu_\ell)$ is not of interest for large values of μ_ℓ, the corresponding barrier problem is solved only to a relaxed accuracy ϵ_ℓ with $\lim_{\ell\to\infty} \epsilon_\ell = 0$. To solve the barrier problem for a fixed value of μ_ℓ, a primal−dual approach [22] is used, and with that, search directions for the primal and dual variables are generated.

For fixed values of μ_ℓ, the barrier problem eq 2 is solved using a Newton method, with directions determined by solving the linear system at iteration k:

$$\begin{bmatrix} \mathbf{W}(x_k,\boldsymbol{\lambda}_k) + \boldsymbol{\Sigma}_k & A(x_k) \\ A(x_k)^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} d_k \\ \boldsymbol{\lambda}_k \end{bmatrix} = - \begin{bmatrix} \nabla\varphi_\mu(x_k) \\ c(x_k) \end{bmatrix} \tag{3}$$

where λ_k is the vector of multiplier estimates at iteration k, $A(x_k) = \nabla c(x_k)$, $\mathbf{W}(x_k,\lambda_k)$ is the Hessian of the Lagrange function $L_\mu(x_k,\lambda_k)$ shown as

$$L_\mu(x_k,\lambda_k) = f(x_k) + \sum \lambda_k c(x_k) - \mu \sum_i \log(x_k^{(i)}) \tag{4}$$

and Σ_k is a diagonal matrix that represents the barrier term. Once the search direction is computed, a backtracking line-search procedure is used where a decreasing sequence of step sizes $\alpha_{k,l} \in (0, \alpha_k^{\max}]$ ($l = 0, 1, 2, ...$) is tried until some acceptance criterion is satisfied. Here $\alpha_k^{\max} = \max \{\alpha \in (0, 1]: x_k + \alpha_k d_k \geq (1 - \tau)x_k\}$ for a fixed parameter $\tau \in (0, 1]$, chosen close to 1. For the line-search filter method in IPOPT, the trial point is accepted if it improves a feasibility measure, i.e., $||c[x_k(\alpha_{k,l})]|| < ||c(x_k)||$, or if it improves the barrier function, i.e., $\varphi_\mu[x_k(\alpha_{k,l})] < \varphi_\mu(x_k)$. The motivation for filter methods[26] is to avoid finding a suitable penalty parameter for a merit function that trades off reductions in infeasibility with improvements in the barrier function. Assuming that Newton directions are usually "good" directions (e.g., when exact second-derivative information is used), filter methods can be more efficient than algorithms based on merit functions because they generally accept larger steps, $\alpha_{k,l}$. Moreover, to guarantee global convergence, several precautions are added to ensure that IPOPT terminates either at a KKT point or a point where the infeasibility is (locally) minimized.[20] IPOPT also exploits the sparsity of the KKT matrix in eq 3, and it also deals with negative curvature and dependent constraints through regularization of the KKT matrix in eq 3. When these are detected, positive quantities are added to the diagonal elements of the matrix in eq 3 in order to ensure a stable pivot sequence and a nonsingular matrix. Finally, to assess the impact of incorporating exact second-derivative information in eq 3 and to compare with quasi-Newton approximations, we also consider the L-BFGS option for IPOPT. Here the matrix $\mathbf{W}(x_k,\lambda_k)$ is approximated using a limited-memory BFGS quasi-Newton update in the full space.

LOQO and KNITRO Interior Point Solvers. The NLP solvers LOQO[19] and KNITRO[25] both implement algorithms for solving the barrier problem (2) for a sequence of barrier parameters, $\mu_l \rightarrow 0$, and are similar in concept to IPOPT. Both methods allow for second-derivative information and exploit the sparsity of the KKT matrix in eq 3. Also, penalty-based merit functions are used to promote convergence. The differences in these solvers are largely due to the steps that promote global convergence. LOQO uses a line search based on a penalty-based merit function. On the other hand, KNITRO applies the composite step trust region method by Byrd et al.;[17] here the KKT matrix (3) is decomposed to determine normal and tangential steps. One significant difference in KNITRO is that an iterative linear (conjugate-gradient) solver is used to determine the tangential step. This leads to a different search direction in the presence of negative curvature or ill-conditioning in eq 3. Generally, this trust region approach requires more iterations than the line-search methods of IPOPT and LOQO, although it can be more reliable on poorly conditioned problems.

Reduced-Space SQP Method. The NLP solver SNOPT[11] implements a SQP algorithm for solving large nonlinearly constrained optimization problems. This method has significant differences from interior point solvers. First, unlike the barrier approach, SNOPT applies an active set strategy determined from the solution of a QP. SNOPT also does not use second-order information; instead, quasi-Newton information is derived. Moreover, the KKT system solved in the QP (essentially eq 3 with $\Sigma_k = 0$) is solved by projection into the null space of the active constraints and does not exploit sparsity of $\mathbf{W}_k$. Finally, SNOPT uses a line search with an augmented Lagrangian merit function to guarantee convergence.

LANCELOT Augmented Lagrangian Method. The NLP solver LANCELOT[3] is based on a minimization of the augmented Lagrangian function through nested calculation loops. The outer level updates the Lagrange multipliers, while the inner loop updates the primal variables, x. This inner minimization applies a bound-constrained trust region minimization of the augmented Lagrangian function using Newton's method, similar to eq 3. As a trust region method, it also applies an iterative linear (conjugate gradient) solver.

Reduced-Gradient Methods. For this comparison, we describe two popular solvers, CONOPT[15] and MINOS[1]. CONOPT sets the nonbasic variables to their bounds, performs an implicit elimination of the basic variables by solving nonlinear constraints, and applies a quasi-Newton method to minimize the objective function in the space of the superbasic variables. CONOPT is a careful implementation that incorporates efficient nonlinear solvers and (re)partitioning of the variable sets. However, because CONOPT is currently not linked to AMPL, we did not consider this solver in our comparison. Nevertheless, the NLP solver MINOS also implements a reduced-gradient algorithm with quasi-Newton approximations and is similar to CONOPT. However, instead of an implicit elimination of the basic variables, MINOS first linearizes the constraints and uses this linearization for the elimination. The approach is based on a sequential linearly constrained (SLC) algorithm for nonlinear constraints. Both CONOPT and MINOS are suitable for large constrained problems with a mixture of linear and nonlinear constraints. They are most efficient for problems with (mostly) linear constraints and not too many superbasic variables (say, a few hundred).

Another popular method is SLP. SLP technology is the cornerstone of all solvers found in oil refinery and petrochemical large-scale planning systems.[27] SLP successively linearizes the objective and constraint functions and solves the resulting linear program. However, because linear programs terminate only at vertex solutions, SLP does not account explicitly for superbasic variables. Consequently, the problem performs poorly on problems with many superbasic variables. Moreover, there are few implementations of SLP methods, none of which are available with AMPL. For these reasons, SLP methods are not considered in this study.

3. Process Problem Classes with Many Degrees of Freedom

We describe two problem classes with many degrees of freedom that are frequently encountered in process operations. In this section, we consider blending problems, particularly for gasoline products, as well as a typical data reconciliation problem. Both problem types consist of nonlinear mass balances, which include concentration, quality, or temperature information. Moreover, their problem sizes can be extended through multiperiod formulations, and their degrees of freedom increase proportionally.

While these problems are nonconvex and admit to locally optimal solutions, we consider only local, and not global, optimization solvers here. Instead, we emphasize that the focus of the study is on computational performance and comparison of these local solvers. Nevertheless, this comparison is also relevant for most global solvers because they often require local solvers in their algorithms.

Gasoline Blending Problem. Gasoline blending problems represent large-scale multiperiod nonlinear programs with mass balance constraints, nonlinear blending properties, large-scale structure (particularly across multiperiods), and combinatorial aspects dealing with bounds and possible switching strategies. Gasoline is one of the most important refinery products as it can yield 60−70% of a typical refinery's total revenue.[28] Thus, tight control of blending operations can provide a crucial edge to the profitability of a refinery. Gasoline blending is featured in the dashed-line rectangle in the supply chain diagram for gasoline production and distribution in Figure 2.

There are various motivations to blend gasoline. First, there are variations in regional demands. For instance, gasoline blends supplied to the state of California must

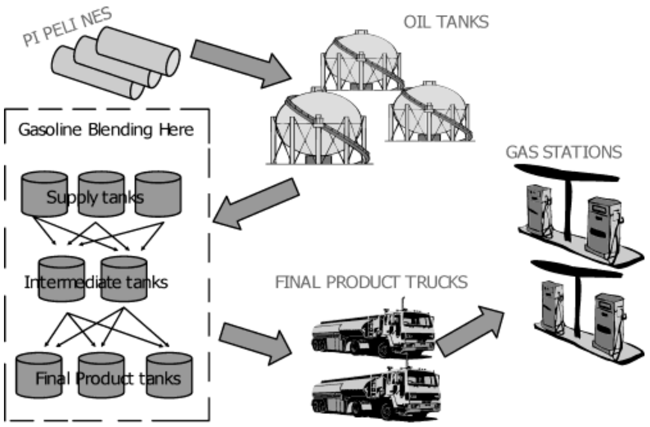

Figure 2. Gasoline production/supply chain diagram.

have a certain amount of alcohol additives to reduce the CO emissions in the atmosphere, as compared to the conventional gasoline blends in other areas. Second, seasonal variations account for particular blends during the summer and winter months. Finally, variations in process rundown quantities and qualities also account for the need to blend gasoline.

A generalization also known as the pooling problem is used to model many actual systems, which have intermediate mixing (or pooling) tanks in the blending process.[29] The pooling problem is a planning problem that arises when blending materials to produce products, e.g., from crude or refined petroleum.[10] Kelly and Mann[27] provide a new scheduling approach and problem formulation for crude oil blending. Blending systems are also encountered in other chemical process industries, such as chemical, pharmaceutical, cosmetics, and food. The classical blending problem arises in refinery processes where feeds with different attribute qualities or properties (e.g., sulfur composition, density or octane number, boiling point temperatures, and flow rate) are mixed together to obtain final products that are dispatched to several locations. There are usually several ways of satisfying the requirements, and the problem is posed as maximizing the difference between the revenue generated by selling the final blends and the cost of purchasing the feeds. The need for blending occurs when the requirements of a demand product are not met by a single feed.

Gasoline blending models often include nonconvex nonlinearities, which lead to the existence of several locally optimal solutions.[30] Given the high volume of sales of petroleum products, global optimization of the pooling and blending processes can lead to substantial savings in cost, resulting in higher profit margins. Pooling problems have been studied in both the operations research and chemical engineering literature since 1978, when Haverly[31] showed how difficult a small weight-based monoperiod fuel-oil blending problem is to solve to global optimality. Since then, several researchers have studied this problem. Simon and Azma[32] compared the performance of Exxon's SLP and SQP techniques with reduced-gradient methods. Floudas and Aggarwal[8] presented a search technique used to obtain global optima in the pooling problem. Floudas et al.[9] presented a number of literature pooling problems, allowing a quick assessment of local and global solution methods. Lodwick[33] developed methods to uncover redundancies, infeasibilities, and bound structures on variables for nonlinear constraints with applications to the pooling problem. Main[34] presented practical aspects of recursion techniques for large models. Ben-Tal et al.[35] reduced the duality gap of pooling problems using global minimization strategies. Greenberg[36] presented approaches for diagnosing infeasible linear programs for pooling problems, and Quesada and Grossmann[37] presented global optimization techniques for bilinear process networks with multicomponent flows. Amos et al.[38] introduced cumulative functions for distillation yields in pooling problems. Adjiman et al.[39] presented a global optimization method for general twice differentiable constrained NLPs. Adya and Sahinidis[30] introduced a new Lagrangian relaxation approach for developing lower bounds for the pooling problem, and Audet et al.[29] presented alternative formulations and methods for the pooling problem.[29] Finally, Tawarmalani and Sahinidis[10] addressed the development of an efficient solution strategy to obtain global optima of nonlinear programs. A general review of bilinear problems is given by Al-Khayyal.[40]

In this study, we do not consider global methods for pooling problems. Instead, the focus of this study is the performance of local solvers for this problem class. Local solvers can also lead to significant improvements, and solutions can be generated much faster, even for blend planning and scheduling applications. Moreover, efficient local solvers are often necessary components of global optimization algorithms. A general gasoline blending formulation is presented below:

$$\max \text{ Profit} = \sum_t \Big(\sum_k c_k f_{t,k} - \sum_i c_i f_{t,i}\Big)$$

$$\text{s.t.} \quad \sum_k f_{t,jk} - \sum_i f_{t,ij} + v_{t+1,j} = v_{t,j}$$

$$f_{t,k} - \sum_j f_{t,jk} = 0$$

$$\sum_k q_{t,k} f_{t,jk} - \sum_i q_{t,i} f_{t,ij} + q_{t+1,j} v_{t+1,j} = q_{t,j} v_{t,j} \tag{5}$$

$$q_{t,k} f_{t,k} - \sum_j q_{t,j} f_{t,jk} = 0$$

$$q_{k_{\min}} \le q_{t,k} \le q_{k_{\max}}$$

$$v_{k_{\min}} \le v_{t,k} \le v_{k_{\max}}$$

where indices i, j, k, and t refer to feeds, intermediates, products, and time, respectively, and the variables f, q, and v are flows, tank qualities, and tank volumes, respectively. The classical blending problem determines the optimal way to mix feeds directly into blends. The basic structure of the pooling problem is similar, except for one set of intermediate pools where the feeds are mixed prior to being directed to the final blends. Generally, there are three types of pools: source pools, having a single purchased feed as the input, intermediate pools with multiple inputs and outputs, and final pools, having a single final blend as the output. Also, if the jk pools have two or more outlet flows at the same time, then splitter equations $q_{t,jk1} - q_{t,jk2} = 0$ need to be added to enforce the same qualities on the outlet. The objective is derived through the input of the source pools and the output of the final pools. Since the qualities blend nonlinearly, bilinear terms are introduced in the model and the difficulty of the bilinear programming problem can be estimated by the number of bilinear

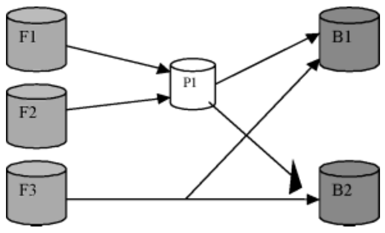

Figure 3. Haverly problem.

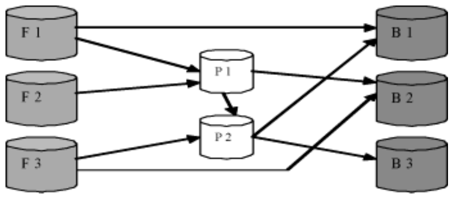

Figure 4. Audet and Hansen problem.

variables and constraints. Additional difficulties occur when the constraint gradients become linearly dependent and the KKT matrix in eq 3 becomes singular. Note that, for this case, a regularization procedure is introduced in IPOPT. The models in the optimization problems are formulated as bilinear programming models; the objective function remains linear, with the nonlinearities seen only in the constraints. The objective function of the gasoline blending model minimizes cost or maximizes profit of production of blends.

In the next section, we consider three categories of blending models. The first category consists of two simple (1-day) models [Haverly[31] (Figure 3) and Audet and Hansen[29] (Figure 4)] formulated as bilinear programming models, with the measure of difficulty seen in the increase in the number of blending tanks to product tanks. One quality was maintained in the tanks for both the Haverly and the Audet and Hansen models. The second category consists of extending the two simple (1-day) models to run on a multiperiod basis (25 days) and analyzing the various characteristics of the model formulation with independent constraints. For these multiday problems, tanks constitute pools with inventory. The third category applies the bilinear programming formulation to a Honeywell Industrial Solutions problem (Figure 5) with dependent constraints, which extends its mathematical programming model to run on a 15-day cycle with 48 different qualities (such as octane number, Reid vapor pressure, etc.) maintained in the tanks. The model diagram for these problems is shown in Figure 5.

Data Reconciliation Problem. Another large-scale engineering application is the *data reconciliation problem*. In any modern chemical plant, petrochemical process, or refinery, hundreds or even thousands of variables (such as flow rates, temperatures, pressures, levels, and compositions) are routinely measured and automatically recorded for process control, online optimization, or process economic evaluation.[7,41] Modern data systems facilitate the collection and processing of large data sets sampled with a frequency of minutes or even seconds. The motivation to have process data reconciled is to obtain reliable information that can be used for management planning, modeling, optimization, design of monitoring systems, instrument maintenance, and environmental compliance and to establish baselines for future work.[41]

Data reconciliation exploits redundancy in process data in order to determine measurement adjustments that lead to data sets that are consistent with the plant model. Here we define a system as redundant when the amount of available data (information) exceeds the minimum amount necessary to solve the simulation problem.[41] Data reconciliation has been developed to improve the accuracy of measurements by reducing the effect of errors in the data. Chemical process data inherently contain some degree of error, and these errors may be random or gross. Random errors are small errors due to the normal fluctuation of the process such as power supply fluctuations, network and signal conversion noise, and changes in ambient temperature.[7] On the other hand, gross errors are larger, systematic errors due to incorrect calibration or malfunction of the instruments, process leaks, and wear or corrosion of sensors. Thus, if the measurement is repeated with the same instrument under identical conditions, the contribution of a gross error will be the same.[41] Gross error detection is a companion technique to data reconciliation that has been developed to detect and identify gross errors. Thus, data reconciliation and gross data detection are applied together to improve the accuracy in measured data and to identify instrumentation problems that require special maintenance and correction. Moreover, detection of incipient gross errors can reduce maintenance costs and provide smoother plant operation. These methods can also be extended to detect faulty equipment[41] but are beyond the scope of this study.

A general formulation of the data reconciliation problem is presented below:

$$\min_{\mathbf{x},\mathbf{u}} \sum_t (\mathbf{y}_t - \mathbf{x}_t)^{\mathrm{T}} \boldsymbol{\Phi}^{-1} (\mathbf{y}_t - \mathbf{x}_t)$$
$$\text{s.t.} \quad \mathbf{f}_t(\mathbf{x},\mathbf{u}) = 0 \tag{6}$$
$$\mathbf{g}_t(\mathbf{x},\mathbf{u}) \le 0$$

where $\mathbf{y}$, $\mathbf{u}$, and $\mathbf{x}$ are vectors of measurements, unmeasured variables, and measured variables at time t, $\boldsymbol{\Phi}$ is the covariance matrix, and $\mathbf{f}$ and $\mathbf{g}$ are vectors of equality and inequality constraints at time t, respectively.

Two popular optimization methods for the nonlinear data reconciliation problem are the reduced-gradient and SQP methods, and several commercial data reconciliation and gross error detection software packages are available. The GAMS optimization package is used to solve the data reconciliation problem using the MINOS reduced-gradient algorithm.[12] Other software for data reconciliation and gross error detection include DATACON (Simulation Sciences Inc.), ROMeo (Simulation Sciences Inc.), DATREC (Elf Central Research), RECON (part of RECONSET of ChemPlant Technology sro, Czech Republic), VALI (Belsim sa), PRODUCTION BALANCE (Honeywell Industrial Solutions), and RAGE (Engineers India Limited).[41] Implementations have also been reported with ASPEN Plus using an SQP algorithm[42] for the data reconciliation problem.

As a test example characteristic of many data reconciliation problems, we consider a bilinear problem formulation for a steam metering problem first introduced by Serth and Heenan.[43] The constraints consist of total mass and energy balances with flows and temperatures measured for each stream. The steam metering process has 28 redundant measured streams flowing in and out of 11 nodes as depicted in Figure 6.

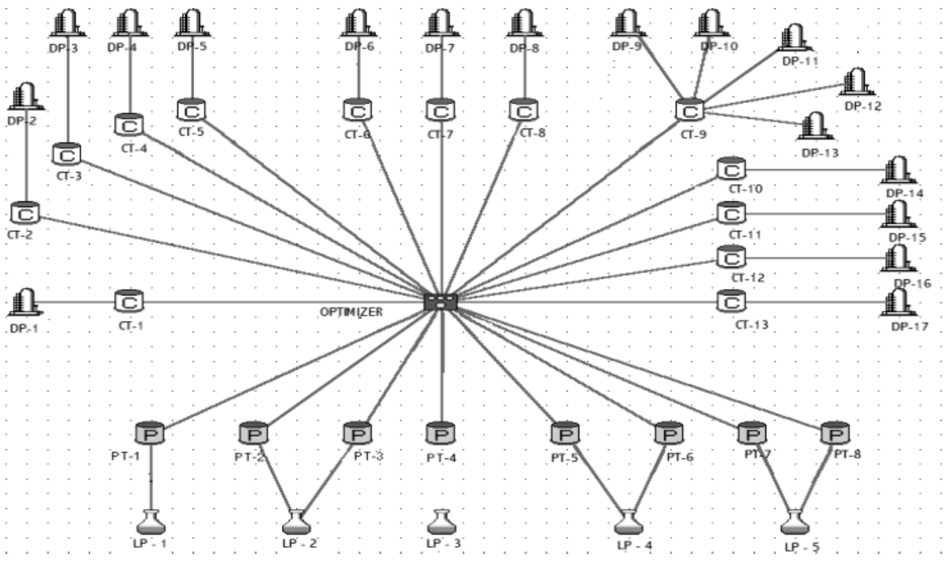

Figure 5. Honeywell industrial problem.

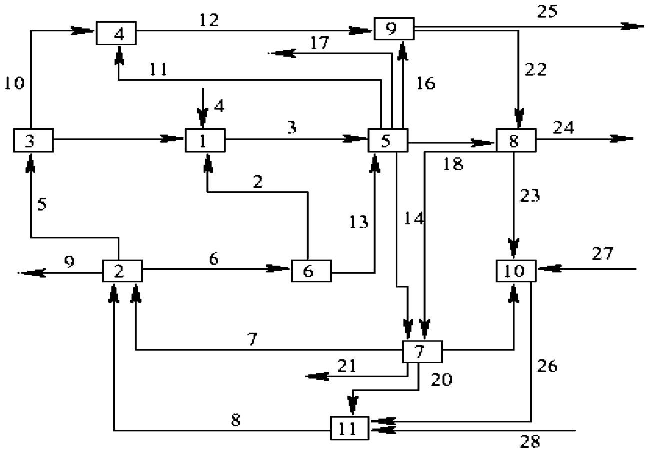

Figure 6. Steam metering problem.

The standard deviation, σ, of the true flow rates was taken to be 1.0% of the true values of the variables. The data generated for this model, which contain no gross errors, were obtained from summing the true values and its corresponding standard deviation values. The model's objective function was calculated using least-squares residual errors, and the (1-day) model formulated was extended to run on a multiperiod basis (up to 25 days).

4. Examples Considered

We selected 13 examples to compare the above NLP solvers for robustness and performance. Eight of these cases are gasoline blending problems, while five are based on the steam metering data reconciliation problem. For the gasoline blending problem, the following cases were analyzed: (i) 1-day Haverly model formulation (HM); (ii) 1-day Audet and Hansen model formulation (AHM); (iii) 25-day HM; (iv) 25-day AHM; (v) 1-day industrial model formulation (IHM); (vi) 5-day IHM; (vii) 10-day IHM; (viii) 15-day IHM. For the data reconciliation problem, the following cases were analyzed: (i) 1-day steam metering model formulation (SM); (ii) 5-day SM; (iii) 10-day SM; (iv) 15-day SM; (v) 25-day SM. All of these problems are nonconvex and may admit locally optimal solutions. For these solvers, we apply the following initialization strategy to obtain good starting points and attempt to determine globally optimal solutions (Figure 7).

In the phase I analysis, the quality, composition, or temperature variables are fixed and the bilinear equations become linear and redundant; these equations are dropped. The NLP now becomes an LP or QP problem containing only flow variables. For phase II, the solution from phase I is used to calculate the quality/composition/temperature variables. We then apply these values to the original NLP formulation.

5. Numerical Results and Discussion

Results from IPOPT are obtained from a Dual Pentium III 800 MHz machine running Linux. The other solvers are obtained from the NEOS Server for Optimization (http://www-neos.mcs.anl.gov) at Argonne National Laboratory from a variety of machines. Results from LANCELOT, KNITRO, and SNOPT are obtained from a SunOS 5.7 UltraSparc-III 360 MHz running Linux, results from LOQO are obtained from a Pentium III 601 MHz running Linux, and results from MINOS are obtained from an Intel Pentium IV 2.53 GHz running Linux. Note that although we usually find the global optimum, these solvers can guarantee only local solutions. Default options were used for all of the solvers. These specific options can be found at the URLs listed with the reference for each method. Tables 1−4 show the CPU times on these machines for illustration; to get a rough comparison, we also present normalized CPU times with respect to the machine used for IPOPT. In addition, other comparisons can be made with iteration counts, which represent the number of linear systems equivalent to eq 3 that were solved. Result tables for these 13 cases are presented below. Here N represents the number of variables, M is the number of equality constraints, and S is the number of superbasic variables at the solution.

In category I, we consider the results for the Haverly and the Audet and Hansen models. For the Haverly model, both KNITRO and LANCELOT find local solutions while the other solvers find the global optimum of 400. For the Audet and Hansen model, KNITRO and MINOS give different local solutions compared to the other solvers. These problems have few superbasic variables, all CPU times are small, and there is no significant difference in the solution times for these solvers. Note, however, that solvers that use exact

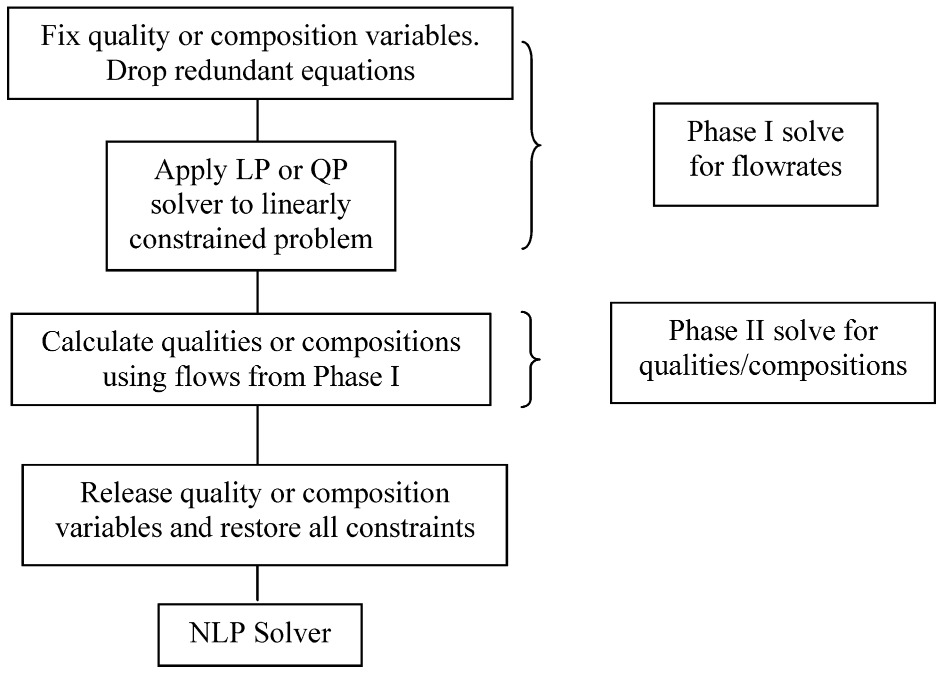
Fix quality or composition variables. Drop redundant equations
Apply LP or QP solver to linearly constrained problem
Phase I solve for flowrates
Calculate qualities or compositions using flows from Phase I
Phase II solve for qualities/compositions
Release quality or composition variables and restore all constraints
NLP Solver

Figure 7. Initialization strategy.

Table 1. Results of Gasoline Blending Models in Category I

	no. of iterations	objective	CPU (s)	normalized CPU (s)
HM Day 1 ($N = 13$, $M = 8$, $S = 8$)				
LANCELOT	62	100	0.10	0.05
MINOS	15	400	0.04	0.13
SNOPT	36	400	0.02	0.01
KNITRO	38	100	0.14	0.06
LOQO	30	400	0.10	0.08
IPOPT, exact	31	400	0.01	0.01
IPOPT, L-BFGS	199	400	0.08	0.08
AHM Day 1 ($N = 21$, $M = 14$, $S = 14$)				
LANCELOT	112	49.2	0.32	0.14
MINOS	29	0.00	0.01	0.03
SNOPT	60	49.2	0.01	<0.01
KNITRO	44	31.6	0.15	0.07
LOQO	28	49.2	0.10	0.08
IPOPT, exact	28	49.2	0.01	0.01
IPOPT, L-BFGS	44	49.2	0.02	0.02

Table 2. Results of Gasoline Blending Models in Category II

	no. of iterations	objective	CPU (s)	normalized CPU (s)
HM Day 25 ($N = 325$, $M = 200$, $S = 200$)				
LANCELOT	67	1.00×10^4	6.75	3.04
MINOS	801	6.40×10^3	1.21	3.83
SNOPT	739	1.00×10^4	0.59	0.27
KNITRO	>1000	*a*	*a*	*a*
LOQO	31	1.00×10^4	0.44	0.33
IPOPT, exact	47	1.00×10^4	0.24	0.24
IPOPT, L-BFGS	344	1.00×10^4	1.99	1.99
AHM Day 25 ($N = 525$, $M = 300$, $S = 350$)				
LANCELOT	149	8.13×10^2	26.8	12.1
MINOS	940	3.75×10^2	2.92	9.23
SNOPT	1473	1.23×10^3	1.47	0.66
KNITRO	316	1.13×10^3	17.5	7.88
LOQO	30	1.23×10^3	0.80	0.60
IPOPT, exact	44	1.23×10^3	0.25	0.25
IPOPT, L-BFGS	76	1.23×10^3	0.98	0.98

[a] Maximum iterations reached.

second derivatives [especially KNITRO, LOQO, and IPOPT (exact)] generally require fewer iterations. As a result, this set of results is meant to be a consistency check that shows the viability of all of the methods.

Category II extends these models from a 1-day to a 25-day period model. Here, the KNITRO solver exceeds the maximum iteration default limit of 1000 for the Haverly model. LANCELOT, MINOS, and KNITRO find different local solutions for the Audet and Hansen model. The smallest iteration counts are required by both LOQO and IPOPT (exact). Again, methods without exact second derivatives (including the L-BFGS option in IPOPT) generally require more iterations. Moreover, we believe that the excessive effort observed with KNITRO and LANCELOT can be explained by their trust region conjugate gradient (CG) method, which may accept a negative curvature direction even if it gives an insignificant reduction.

Category III considers various multiperiod instances of an industrial model provided by Honeywell. The models have 48 qualities, and by extending the model from 1 to 15 days, the NLP increases from 2003 to 31 743 variables. Most of the solvers have difficulty with this model beyond the first day. For the first day, the best solver performance is achieved by IPOPT. For the 5-day case, only MINOS and IPOPT (both options) were able to solve this problem. Note though that IPOPT (exact) was significantly more efficient than the other two methods. Moreover, because the IPOPT (L-BFGS) method required excessive solution, it was not considered for the larger problems.

Beyond 5 days, only IPOPT (exact) was able to provide a solution. Note that, even in this case, a larger CPU time was required. This problem is difficult for IPOPT because the redundancies in the constraints also make the KKT matrix ill-conditioned and expensive to factorize with the sparse solver. We suspect that LOQO and KNITRO failed for the same reasons. Compared to MINOS, the longer CPU times for IPOPT are entirely dependent on the sparse linear solver. MINOS, which is also run on the fastest computer, has very efficient strategies to identify redundant constraints and solve smaller linear systems much more quickly, as long as these systems are not too large. As a result, it solves

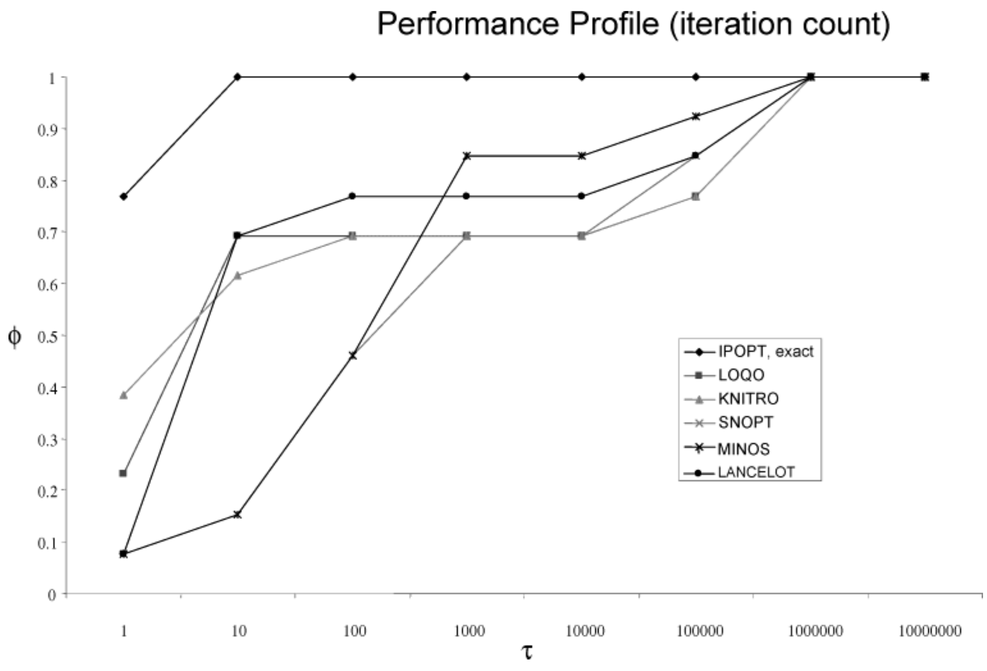

Figure 8. Solver performance chart.

Table 3. Results of Gasoline Blending Models in Category III

	no. of iterations	objective	CPU (s)	normalized CPU (s)
IHM Day 1 ($N = 2003$, $M = 1595$, $S = 1449$)				
LANCELOT	388	6.14×10^1	1.17×10^5	5.28×10^3
MINOS	2238	6.14×10^1	5.24×10^1	1.66×10^2
SNOPT	*a*	*a*	*a*	*a*
KNITRO	37	1.00×10^2	1.58×10^2	7.11×10^1
LOQO	*b*	*b*	*b*	*b*
IPOPT, exact	21	6.14×10^1	2.60	2.60
IPOPT, L-BFGS	52	6.14×10^1	8.89	8.89
IHM Day 5 ($N = 10\,134$, $M = 8073$, $S = 7339$)				
LANCELOT	*c*	*c*	*c*	*c*
MINOS	8075	1.39×10^5	3.08×10^2	9.74×10^2
SNOPT	*a*	*a*	*a*	*a*
KNITRO	*a*	*a*	*a*	*a*
LOQO	*b*	*b*	*b*	*b*
IPOPT, exact	39	1.39×10^5	1.06×10^3	1.06×10^3
IPOPT. L-BFGS	1000	1.39×10^5	2.91×10^5	2.91×10^5
IHM Day 10 ($N = 20\,826$, $M = 16\,074$, $S = 15\,206$)				
LANCELOT	*c*	*c*	*c*	*c*
MINOS	*a*	*a*	*a*	*a*
SNOPT	*a*	*a*	*a*	*a*
KNITRO	*a*	*a*	*a*	*a*
LOQO	*b*	*b*	*b*	*b*
IPOPT, exact	65	2.64×10^4	1.12×10^4	1.12×10^4
IHM Day 15 ($N = 31\,743$, $M = 25\,560$, $S = 23\,073$)				
LANCELOT	*c*	*c*	*c*	*c*
MINOS	*a*	*a*	*a*	*a*
SNOPT	*a*	*a*	*a*	*a*
KNITRO	*a*	*a*	*a*	*a*
LOQO	*b*	*b*	*b*	*b*
IPOPT, exact	110	4.15×10^4	7.25×10^4	7.25×10^4

[a] Solver failure. [b] Failure due to primal−dual infeasibility. [c] Failure due to insufficient memory allocation.

the 1-day and 5-day problems rather quickly. On the other hand, MINOS requires over 100 times as many iterations and function evaluations as IPOPT (exact) on these problems. Future developments for IPOPT are planned to address inefficiencies with the sparse linear solver.

Results of the data reconciliation model are given in Table 4. All of the solvers found the global solution in all cases. Note that we are minimizing the square of errors in the flowmeter readings, hence the small values for the objective function obtained. Here, KNITRO and

Table 4. Results of the Steam Metering Data Reconciliation Problem

	no. of iterations	objective	CPU (s)	normalized CPU (s)
SM Day 1 ($N = 40$, $M = 23$, $S = 17$)				
LANCELOT	11	3.53×10^{-5}	0.06	0.03
MINOS	41	3.53×10^{-5}	0.03	0.09
SNOPT	38	3.53×10^{-5}	0.04	0.02
KNITRO	2	3.53×10^{-5}	0.06	0.03
LOQO	19	3.53×10^{-5}	0.12	0.09
IPOPT, exact	2	3.53×10^{-5}	<0.01	<0.01
IPOPT, L-BFGS	3	3.53×10^{-5}	<0.01	<0.01
SM Day 5 ($N = 196$, $M = 111$, $S = 85$)				
LANCELOT	11	1.77×10^{-4}	0.19	0.09
MINOS	176	1.77×10^{-4}	0.14	0.44
SNOPT	178	1.77×10^{-4}	0.06	0.03
KNITRO	2	1.77×10^{-4}	0.06	0.03
LOQO	19	1.77×10^{-4}	0.04	0.03
IPOPT, exact	2	1.77×10^{-4}	0.02	0.02
IPOPT, L-BFGS	3	1.77×10^{-4}	0.02	0.02
SM Day 10 ($N = 391$, $M = 221$, $S = 170$)				
LANCELOT	10	3.53×10^{-4}	0.30	0.14
MINOS	318	3.53×10^{-4}	0.58	1.83
SNOPT	353	3.53×10^{-4}	0.23	0.10
KNITRO	2	3.53×10^{-4}	0.09	0.04
LOQO	19	3.53×10^{-4}	0.68	0.51
IPOPT, exact	2	3.53×10^{-4}	0.02	0.02
IPOPT, L-BFGS	3	3.53×10^{-4}	0.02	0.02
SM Day 15 ($N = 586$, $M = 331$, $S = 255$)				
LANCELOT	9	5.30×10^{-4}	0.45	0.20
MINOS	461	5.30×10^{-4}	1.55	4.90
SNOPT	528	5.30×10^{-4}	0.59	0.27
KNITRO	2	5.30×10^{-4}	0.10	0.05
LOQO	19	5.30×10^{-4}	1.04	0.78
IPOPT, exact	2	5.30×10^{-4}	0.03	0.03
IPOPT, L-BFGS	3	5.30×10^{-4}	0.04	0.04
SM Day 25 ($N = 976$, $M = 551$, $S = 425$)				
LANCELOT	11	8.83×10^{-4}	0.84	0.38
MINOS	793	8.83×10^{-4}	6.32	20.0
SNOPT	878	8.83×10^{-4}	1.72	0.77
KNITRO	2	8.83×10^{-4}	0.15	0.07
LOQO	20	8.83×10^{-4}	2.06	1.55
IPOPT, exact	2	8.83×10^{-4}	0.05	0.05
IPOPT, L-BFGS	3	8.83×10^{-4}	0.05	0.05

IPOPT require the fewest iterations. Moreover, note that the solvers SNOPT and MINOS require more iterations than the solvers that use second derivatives. Iteration counts for MINOS and SNOPT also increase significantly with the number of periods (and superbasic variables). Note that IPOPT (L-BFGS) requires only one more iteration than IPOPT (exact). This performance is likely due to the structure of the $\mathbf{W}(x,\lambda)$ matrix, which is dominated by diagonal terms.

All of these results can be summarized by the Dolan−Moré plot[44] in Figure 8. Here we define $t_{p,s}$ as the number of iterations required for problem p and solver s. We also define the performance ratio as $\rho_{p,s} = t_{p,s}/\min_s \{t_{p,s}\}$. Here we set $\rho_{p,s} = \infty$ for problems that could not be solved by a given solver. For each solver, we define $\phi(\tau)$ as the number of problems solved with $\rho_{p,s} \leq \tau$. A plot of these functions with respect to τ leads to the performance profile in Figure 8. Figure 8 gives a summarized overview of all of the tabulated results with the various solver performances. From values at $\tau = 1$, we see that IPOPT solves most of the problems with the fewest iterations. Moreover, as τ increases, IPOPT remains superior to the other solvers and is able to solve all of the problems presented.

6. Conclusions and Future Work

Large-scale nonlinear optimization problems have become an important part of computer-aided process operations. Many of these applications also have many degrees of freedom for optimization. For this purpose, NLP solvers that incorporate second-derivative information, exploit the sparsity of the KKT matrix, deal efficiently with large sets of active constraints, and handle dependent constraints and negative curvature are needed. These features are considered in IPOPT, a novel barrier solver that incorporates a line-search filter method.

This study compares IPOPT with five popular large-scale NLP solvers (MINOS, SNOPT, LOQO, KNITRO, and LANCELOT) on two problem classes in process operations: blending and data reconciliation. In particular, the first two solvers are widely used but do not use second-derivative information. By extending these problems to multiperiod formulations, NLPs are constructed that require several thousand degrees of freedom. For all of these cases, IPOPT appears to be well suited for process problems and outperforms the other solvers considered in this study, in particular, MINOS and SNOPT.

Future work will deal with improving problems with dependent constraints and negative curvature that are common with these problem classes. Currently, we deal with these through regularization of the KKT matrix in order to allow stable pivoting of a singular matrix. Additional features that include preprocessing to eliminate dependent constraints are currently being considered.

Acknowledgment

Support from Honeywell International and the Chemical Engineering Department at Carnegie Mellon University is gratefully acknowledged for this research. We also thank Dr. Andreas Waechter for help with IPOPT and Dr. Arvind Raghunathan for help with the AMPL interface.

Literature Cited

(1) Murtagh, A. B.; Saunders, M. *MINOS 5.0 Users' Guide*; Technical Report SOL 83.20; 1993 (see also http://www.ampl.com/BOOKLETS/ampl-minos.pdf).

(2) Cervantes, A. M.; Biegler, L. T. Large-scale DAE optimization using a simultaneous NLP formulation. *AIChE J.* **1998**, *44* (5), 1038.

(3) Conn, A. R.; Gould, I.; Toint, Ph. *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*; Number 17 in Springer Series in Computational Mathematics; Springer-Verlag: Heidelberg, 1992 (see also http://www-unix.mcs.anl.gov/~more/neos/lancelot/lancelot_options.html).

(4) Lucia, A.; Xu, J.; D'Couto, G. C. Sparse quadratic programming in chemical process optimization. *Ann. Oper. Res.* **1993**, *42*, 55−83.

(5) Lucia, A.; Xu, J. Chemical process optimization using Newton-like methods. *Comput. Chem. Eng.* **1990**, *14*, 119.

(6) Albuquerque, J. Parameter Estimation and Data Reconciliation for Dynamic Systems. PhD. Thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, 1996.

(7) Romagnoli, J. A.; Sanchez, M. C. *Data Processing and Reconciliation for Chemical Process Operations*; Academic Press: New York, 2000; p 2.

(8) Floudas, C. A.; Aggarwal, A. A decomposition strategy for global optimum search in the pooling problem. *ORSA J. Comput.* **1990**, *2* (3), 225−235.

(9) Floudas, L. R.; Haugland, D.; Jorsten, K. A bilinear approach to the pooling problems. *Optimization* **1992**, *24*, 165−180.

(10) Tawarmalani, M.; Sahinidis, N. V. *Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software and applications*; Kluwer Academic Publishers: Boston, 2002.

(11) Gill, P.; Murray, W.; Saunders, M. *SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization*; Report NA 97-2; 1997 (see also http://www-neos.mcs.anl.gov/neos/solvers/NCO: SNOPT-AMPL/).

(12) Sargent, R. W. H.; Ding, M. A new SQP algorithm for large-scale nonlinear programming. *SIAM J. Optim.* **2000**, *11* (3), 716−747.

(13) Ternet, D. New Approaches to a Reduced Hessian Successive Quadratic Programming Method for Large-Scale Process Optimization. Ph.D. Thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, 1998.

(14) Betts, J. T.; Frank, P. D. A sparse nonlinear optimization algorithm. *J. Optim. Theory Appl.* **1994**, *3*, 519−541.

(15) Drud, A. S. CONOPT−A Large Scale GRG Code. *ORSA J. Comput.* **1994**, 207−216.

(16) Fiacco, A. V.; McCormick, G. P. *Nonlinear Programming Sequential Unconstrained Minimization Techniques*; John Wiley: New York, 1990.

(17) Byrd, R. H.; Hribar, M. E.; Nocedal, J. An interior point algorithm for large-scale nonlinear programming. *SIAM J. Optim.* **1999**, *9* (4), 877−900 (also see http://www.ece.northwestern.edu/~rwaltz/knitro/options.html).

(18) El-Bakry, S. A.; Tapia, A. R. On the Formulation and Theory of the Newton Interior-Point Method for Nonlinear Programming. *J. Optim. Theory Appl.* **1996**, *89*, 507−541.

(19) Vanderbei, R. J.; Shanno, D. F. An Interior Point Method for Nonconvex Nonlinear Programming. *Comput. Optim. Appl.* **1999**, *13*, 232 (see also http://www.princeton.edu/~rvdb/tex/loqo/loqo405.pdf).

(20) Waechter, A. An Interior Point Algorithm for large-scale Nonlinear Optimization with Applications in Process Engineering. Ph.D. Thesis, Chemical Engineering Department, Carnegie Mellon University, Pittsburgh, PA, 2002 (see also http://www.coin-or.org).

(21) Karmarkar, N. A new polynomial-time for linear programming. *Combinatorics* **1984**, *4*, 373−395.

(22) Forsgren, A.; Gill, P. E.; Wright, M. H. Interior Methods for Nonlinear Optimization. *SIAM Rev.* **2002**, *44* (4), 525−597.

(23) Lasdon, L. S.; Plummer, J.; Yu, G. Primal−Dual and Primal Interior Point Algorithms for General Nonlinear Programs. *ORSA J. Comput.* **1995**, *7* (3), 321−332.

(24) Fourer, R.; Gay, D.; Kernighan, B. *AMPL: A Modeling Language for Mathematical Programming*; The Scientific Press: San Francisco, 1993.

(25) Nocedal, J.; Byrd, R.; Waltz, R. *Feasible Interior Point Methods Using Slacks for Nonlinear Optimization*; Technical Report OTC 2000/11; Optimization Technology Center, North-

western University: Evanston, IL, 2000 (also see http://www.ece.northwestern.edu/~rwaltz/knitro/options.html).

(26) Fletcher, R.; Leyffer, S. *Nonlinear Programming without a penalty function*; Technical Report NA/171, University of Dundee Numerical Analysis Report; University of Dundee: Dundee, Scotland, U.K., Sept 1997, revised Dec 1998.

(27) Kelly, J. D.; Mann, J. L. Crude Oil Blend Scheduling Optimization: An Application with Multimillion Dollar Benefits. *Hydrocarbon Process.* **2002**, 47−53; **2003**, 72−79.

(28) DeWitt, W. C.; Lasdon, S. L.; Waren, D. A.; Brenner, A. D. Omega: An Improved Gasoline Blending System for Texaco. *Interfaces* **1989**, *19*, 85−101.

(29) Audet, C.; Hansen, P.; Brimberg, J. *Pooling Problem: Alternate Formulations and Solutions Methods*; Les Cahiers du GERAD G 2000-23; 2000.

(30) Adhya, N.; Sahinidis, N. A Lagrangian Approach to the Pooling Problem. *Ind. Eng. Chem. Res.* **1999**, *38*, 1956−1972.

(31) Haverly, C. A. Studies of the Behavior of Recursion for the Pooling Problem. *SIGMAP Bull.* **1978**, 25.

(32) Simon, J. D.; Azma, H. M. Exxon experience with large scale linear and nonlinear programming applications. *Comput. Chem. Eng.* **1983**, *7* (5), 605−614.

(33) Lodwick, W. A. Preprocessing nonlinear functional constraints with application to the pooling problem. *ORSA J. Comput.* **1992**, *4* (2), 119−131.

(34) Main, R. A. Large recursion models: practical aspects of recursion techniques. In *Optimization in Industry*; Ciriani, T. A., Leachman, R. C., Eds.; John Wiley & Sons Ltd.: New York, 1993.

(35) Ben-Tal, A.; Eiger, G.; Gershovitz, V. Global minimization by reducing the duality gap. *Math. Program.* **1994**, *63*, 193−212.

(36) Greenberg, H. J. Analyzing the pooling problem. *ORSA J. Comput.* **1995**, *7* (2), 206−217.

(37) Quesada, I.; Grossmann, I. E. Global optimization of bilinear process networks with multicomponent flows. *Comput. Chem. Eng.* **1995**, *19* (2), 1219−1242.

(38) Amos, F.; Ronnqvist, M.; Gill, G. Modeling the pooling problem at the New Zealand refining company. *J. Oper. Res. Soc.* **1997**, *48*, 767−778.

(39) Adjiman, C. S.; Dallwig, S.; Floudas, C. A.; Neumaier, A. A global optimization method, αBB, for general twice-differentiable constrained NLPs—II, implementation and computational results. *Comput. Chem. Eng.* **1998**, *22*, 1160−1179.

(40) Al-Khayyal, F. A. Generalized bilinear programming—Part I models: Applications and linear programming relaxation. *Eur. J. Oper. Res.* **1992**, *60*, 306−314.

(41) Narasimhan, S.; Jordache, C. *Data Reconciliation & Gross Error Detection: An Intelligent Use of Process Data*; Gulf Publishing Co.: Houston, TX, 2000.

(42) Keenan, G. The Use of Optimization in the Evaluation of Plant Performance Data. AspenTech Users Conference, 2001.

(43) Serth, R. W.; Heenan, W. A. Gross Error Detection and Data Reconciliation in Stream Metering-Systems. *AIChE J.* **1986**, *32* (5), 733.

(44) Dolan, E.; Moré, J. J. Benchmarking optimization software with performance profiles. *Math. Program., Ser. A* **2002** (online).

Received for review August 29, 2003
Revised manuscript received January 22, 2004
Accepted July 30, 2004

IE0341000