Acceleration of Reduced Hessian Methods for Large-Scale Nonlinear Programming

C. Schmid, LT. Biegler

EDRC 06-131-92

ACCELERATION OF REDUCED HESSIAN METHODS FOR LARGE-SCALE NONLINEAR PROGRAMMING

Claudia Schmid and Lorenz T. Biegler
Chemical Engineering Department
Carnegie Mellon University
Pittsburgh. PA 15213

Submitted for Special Issue of *Computers and Chemical Engineers*
May, 1992

1. Introduction

In recent years, the search for economically viable process operating conditions has led to the increasing importance of optimization techniques as a tool in the design, control and retrofit of chemical processes. Since most process systems have an inherently nonlinear structure, development of a **robust** and efficient nonlinear programming strategy is a key issue. The two **major methods currently used to solve nonlinear** programming problems are the **linearized reduced gradient method (Murtagh and** Saunders, 1979) and Successive **Quadratic Programming (SQP) (Han, 1977; Powell,** 1977). The **linearized reduced gradient method, as implemented in MINOS, for example,** is best suited for cases **where all the nonlinearities are in the objective function and the** constraints **are linear. SQP, on the other hand, has emerged as the algorithm of choice** for **solving significantly nonlinear problems with a small to moderate number of variables; it consistently requires fewer function evaluations per Iteration than other** methods.

Two **main schools of thought have emerged for extending SQP to successfully handle larger systems. A number of studies have concentrated on sparse full-space methods and have developed efficient sparse successive quadratic programming solvers (Lucia and Xu, 1990; Nickel and Tolle, 1989). This approach appears to be a quite natural and general extension to SQP, but it also requires a more complex algorithm. The issues that must be addressed for these algorithms include calculation or approximation of the full Hessian matrix and treatment of indefinite Hessians and directions of negative curvature. The second approach focuses on decomposition of the search space in order to reduce the dimensionality of the problem. Given that a significant portion of process engineering problems have relatively few degrees of freedom, a characteristic which is exploited by decomposition, we find it useful to concentrate on reduced space or reduced Hessian SQP methods. Such methods create and solve a reduced space quadratic program with a reduced Hessian that is expected to be positive definite at the solution. Consequently, this matrix can be approximated by positive definite quasi-Newton formulae such as BFGS. In this study, we also show that these decomposition methods are also much easier to intexface with specialized Newton-based solution strategies for *existing process models.* In this way, an SQP-based optimization procedure can easily be tailored to particular classes of process problems.**

Abstract

Process optimization problems are frequently characterized by large models, with many variables and constraints but relatively few degrees of freedom. Thus, reduced Hessian decomposition methods applied to Successive Quadratic Programming (SQP) exploit the low dimensionality of the subspace of the decision variables, and have been very successful for a wide variety of process application. However, further development is needed for improving the efficient *large-scale* **use** of these tools.

In this **study we develop an improved SQP algorithm decomposition with coordinate** bases **that includes an inexpensive second order correction term. The resulting algorithm is 1-step Q-superlinearly convergent. More importantly, though, the resulting algorithm is largely independent of the specific decomposition steps. Thus, the inexpensive factorization of the coordinate decomposition, which lends itself very well to tailoring, can be applied in a reliable and efficient manner.**

With this efficient and easy-to-implement NLP strategy, we continue to improve the efficiency of the optimization algorithm by exploiting the mathematical structure of *existing* process engineering models. Here we consider the tailoring of a reduced Hessian method for the block tridiagonal structure of the model equations for distillation columns. This approach is applied to the Naphthali-Sandholm algorithm implemented within the UNIDIST and programs. Our reduced Hessian SQP strategy is incorporated within the package with only minor changes In the program's interface and data structures. Through this integration, reductions of 20% to 80% in the total CPU time are obtained compared to general reduced space optimization; an order of magnitude reduction Is obtained when compared to conventional sequential strategies.

Consequently, this approach shows considerable potential for efficient and reliable large-scale process optimization, particularly when complex Newton-based process models are already available.

For large systems, various decomposition strategies have been proposed which reduce the dimensionality of the problem and thus allow for an efficient algorithm. A decomposition strategy using coordinate basis matrices was used by Locke *et aL* (1983); while the decomposition itself is efficient, it frequently leads to inconsistent convergence results. To remedy this, Vasantharajan and Biegler (1988) investigate the use of orthogonal basis representations. Here, although the **computational effort per** iteration is **higher,** especially as the **number** of degrees of **freedom increases, the** resulting SQP method is more robust. However, orthogonal **projections are not always** easy to adapt to the sparsity structure of the process model. **In section 2 we briefly** describe the concept of decomposition for SQP with details on **the** use **of coordinate and** orthogonal bases.

While these reduced Hessian methods show much promise, they may still be inefficient when faced with large, complex process engineering problems. The performance of SQP for large-scale models can be improved considerably through a closer examination of the mathematical structure inherent to individual classes of chemical systems; for example, the tridiagonal structure of the Jacobian matrix for distillation problems. To tailor the SQP algorithm to take advantage of this underlying structure we adapt the reduced space SQP strategy so that the appropriate equation solver is used directly as part of the optimization procedure, in order to eliminate the dependent variables. In fact, any model specific structures and procedures used to generate the Newton step can be exploited directly here. As will be discussed later, coordinate basis representations are best suited for this task and thus we first discuss how to guarantee consistent convergence results with this decomposition strategy. Section 3 covers the details of our improved coordinate basis method and outlines three approaches for the calculation of a necessary second order correction term. We also present a series of test problem results using uncorrected orthogonal and coordinate bases as well as our improved algorithm.

Finally, using the improved coordinate bases algorithm, we exploit the structure of specific classes of process engineering problems in section 4. In particular, we consider the optimization of distillation columns and interface directly to the UNIDIST and NRDIST packages (Andersen *et* at, 1991). Here, the model equations form a block-tridiagonal matrix, and are solved using the Naphthali-Sandholm distillation algorithm. After making only minor changes within the interfaces and model solution algorithms, we are able to realize up to an order of magnitude increase in efficiency over

the conventional sequential procedure. The final section of the paper summarizes this work and briefly outlines future work.

2. Decomposition Strategies for SQP

In this section we discuss the background for successive quadratic programming and various decomposition strategies. The nonlinear programming problem (NLP) to be solved is **formulated** as.

$$\begin{aligned} \underset{z \in \mathbb{R}^n}{\text{Min}} \quad & \phi(z) \\ \text{s.t.} \quad & h(z) = 0 \qquad h: SR^n \to SR^m \\ & z^L \le z \le z^u \end{aligned} \tag{2.1}$$

Successive quadratic programming is motivated by a Newton method for the solution of the **Kuhn-Tucker optimality conditions; this can be shown to be equivalent to the solution of a sequence of quadratic programming (QP) subproblems. At the kth iteration, the QP subproblem has the form**

$$\begin{aligned} \underset{d}{\text{Min}} \quad & \nabla\phi(z_k)^T d + \tfrac{1}{2} d^T B^k d \\ \text{s.t.} \quad & h(z_k) + \nabla h(z_k)^T d = 0 \\ & z^L \le z_k + d \le z^u \end{aligned} \tag{2.2}$$

where d is the search direction and B^k is the Hessian of the Lagrange function. Convexity of (2.2) is guaranteed by calculating B^k using the BFGS matrix update formula with Powell damping (Powell. 1977). At each iteration, the following first-order Kuhn-Tucker optimality conditions must be satisfied (e.g.9 for bounds Inactive).

$$\begin{bmatrix} B & \nabla h \\ \nabla h^T & 0 \end{bmatrix} \begin{bmatrix} d \\ v \end{bmatrix} = - \begin{bmatrix} \nabla\phi \\ h \end{bmatrix} \tag{2.3}$$

where v is the vector of Lagrange multipliers. As the size of the NLP in (2.1) increases, the solution of the QP becomes increasingly expensive; B is a dense *nxn* matrix which must be updated and stored at each iteration, and sparsity is often not exploited. It has been demonstrated that the performance of the method can be considerably improved through a suitable change of basis representation. The new basis vectors are obtained by partitioning the search space into two subspaces, Z and Y, where the columns of Z span the null space of $\nabla h^T(z_k)$. After decomposition, the matrix to be updated is a projection of B onto a space whose dimension is given by the number of degrees of

freedom of the problem. This is suitable for many process engineering problems since in many instances this number is conveniently small (10 to 100). Further, the actual projected Hessian at the solution is expected to be positive definite at the solution, which is sufficient to guarantee optimality.

2.1 Decomposition

As indicated above, Z(zk) is an *wcfn-mj* matrix whose columns span the null space of $Vh^T(z_k)$. $Y(z_k)$ is an *wan* matrix chosen so that [Y *Z]* is nonsingular; Y and Z together span the entire search space. Let Q be a nonsingular matrix of order (n+mh given by

$$Q \; {}^{J[YZI\ on}_{'L\ \ o\ \ ij} \tag{2.4}$$

The search direction d is expressed as the sum of its components in the two subspaces.

$$d = Yp_Y + Zp_z \tag{2.5}$$

Premultiplying the system in (2.3) by Q, and substituting for d yields

$$\begin{bmatrix} Y^TBY & Y^TBZ & Y^TVh \\ ZPBYZPBZ & o & \\ Vh^TY & 0 & 0 \end{bmatrix} \begin{bmatrix} PY \\ Pz \\ V \end{bmatrix} = - \begin{bmatrix} Y^T\nabla\phi \\ Z^T\nabla\phi \\ h \end{bmatrix} \tag{2.6}$$

The Lagrange multipliers of the equality constraints are obtained from the solution of

$$Y^TBYp_Y + Y^TBZp_Z + V^IVhv = - Y\text{^}fy \tag{2.7}$$

Since exact values for these multipliers are only required at the solution of the NLP, once the algorithm has converged and Py = Pz= 0, (2.7) can be reduced to

$$Y\text{^}Vhv = - Y^T\nabla\phi \tag{2.8}$$

The reduced QP to be solved at each iteration can be written as

$$\begin{aligned} \text{Mln} \quad & f = (Z^T\nabla\phi(z_k) + Z^TBYp_Y)^Tp_Z + \tfrac{1}{2}p_Z^TZ^TB^kZp_Z \\ \text{S.t} \quad & Z^L \le z_k + Yp_Y + Zp_Z \le z^U \end{aligned} \tag{2.9}$$

where

$$IV = - (Vh^TY)^{-1}h \tag{2.10}$$

Note that this QP is *equivalent* to (2.2) for *any* nullspace representation Z and *any Y* that makes [Z Y] nonsingular.

In the objective function of (2.9). the matrix Z^TBY, which is of dimension *nx(n-m)* appears. It is usually "long and thin" and much larger than 2^BZ, which is of dimension *(n-m)x(n-ml.* Hence, in most reduced Hessian SQP methods there is incentive not to include this matrix. Moreover, if Yp_y is uniformly small as compared to Zp_z, we are further justified in neglecting this term. Alternatively, 2^BYp_Y can be estimated using a variety **of** approximations, as will be discussed later.

2.2 Choice of the basis matrices Y and Z

As long as Z spans the null space of $Vh^T(z_k)$, and [Y Z] is nonsingular, the choice of Y and Z is **essentially arbitrary. Previous studies have concentrated on three distinct basis representations for Y and Z.**

Orthonormal bases

Most references to SQP decomposition use orthonormal bases; the advantage here is that that the search space is well-conditioned and well-scaled. Nocedal and Overton (1985) have shown that this definition of Y and Z yields a 2-step Q-superlinearly convergent algorithm. However, the subspace matrices are obtained from a QR factorization at each iteration which makes this approach expensive and undesirable for large problems.

Orthogonal bases

A computationally less Intensive decomposition uses orthogonal bases (Vasantharajan and Biegler, 1988). The variables are partitioned into dependent variables, y e $9t^m$, and independent variables, x € Stf^{n-m}). z and Y are defined as

$$Z = \begin{bmatrix} I \\ -(\nabla_y h^T)^{-1} \nabla_x h^T \end{bmatrix} \qquad Y = \begin{bmatrix} \nabla_x h \ (Vyh)\text{-}n \end{bmatrix} \qquad (2.11)$$

Although *PY* and p_z calculated from this method are different from those obtained using orthonormal bases, it is easily shown that the actual moves, Ypy and Zp_z, are unchanged; hence the same convergence properties are observed. Compared to orthonormal representations, the work to determine Y and Z at each iteration is reduced considerably. However, the calculation of py and v using (2.8) and (2.10)

involves factorization of Y^h. Through the use of the Sherman-Morison formula, the range space move, py. and multipliers, v, are given by:

$$Pr = -[I - V(I + V^TV)^{-1} V^T] (\nabla_y h^T)^{-1} h$$

$$v = -(Vyh^{*1})^{*1} [I - V(I + V^TV)^{-1} V^T]^T Y^T \nabla\phi$$

where $V = (Vyh^1)^{*1} V_x h^T$. Since Vh is premultiplied by Y^1, the sparsity inherent to Vh cannot be exploited completely and, when *(n-m)* is large, this factorization can be expensive.

Coordinate bases

Gabay (1982) and Locke *et aL* (1983) propose the following definitions for Y and Z.

$$Z = \begin{bmatrix} I \\ -(\nabla_y h^T)^{-1} \nabla_x h^T \end{bmatrix} \qquad Y = \begin{bmatrix} 0 \\ I \end{bmatrix} \tag{2.12}$$

Here the effort required to determine the basis vectors is reduced even further. Moreover, factorization of Y^h is now equivalent to factorization of the model Jacobian $V_y h$. Model sparsity is maintained and py and v are obtained directly from a Newton step for solving h(z) = 0. Consequently, it is relatively straightforward to tailor this algorithm to take advantage of the particular structure of certain problem classes which can greatly improve the performance of SQP for complex problems.

2.3 **Comparison of orthogonal and coordinate bases**

A geometrical interpretation of the orthogonal bases and coordinate bases methods is given in Figure 1 for a two-dimensional problem with variables, z_x and 2fc. Hie initial point at the kth iteration is given by z°. The Z space move Is Identical for both decompositions and lies at a tangent to the space of the linearized equality constraint. Using orthogonal bases yields Ypy^1. which corresponds to a least squares projection: It represents the shortest distance between the current point and the subspace defkied by the linearized equality constraint, hence Ypy is minimized. Ypy^2 and Yp?^{s} are the two possible Y space moves using coordinate bases; one or the other is obtained depending on the choice of the independent variable.

Since coordinate bases result in the cheapest decomposition, and allow us to exploit the sparsity, or even more importantly, the particular structure inherent to the process

model when calculating p_Y. it follows that this method would be the method of choice for large-scale process optimization. However, other factors must be taken into consideration. In particular, we return to the objective function of the quadratic subproblem (2.9). The matrix *Z?BY,* of dimension *(n-m}xm* may be too large to store or to compute when m is large. We would like to neglect the term Z^TBYp_Y, reducing the objective function to

$$\mathbf{f} = V\text{^}zJ\ Zpz + |p_z{}^T Zn3^k Zpz \qquad (2.13)$$

In the extreme, when all the constraints are linear, the Y space move, Yp_Y, can be nonzero only at the first iteration, as the algorithm moves to a feasible point. At all subsequent iterations, any feasible search direction involves movement **only** in the Z space. **Hence, the term 2^3Ypy** becomes **zero after the first iteration and we are justified** in **making the above simplification.**

When using orthogonal bases, Yp_Y is **minimized. Although it may initially be larger than Zp_z, it has been shown that Ypy tends to zero faster than Zpz does. Hence, the objective function (2.13) above can be used without generating significant error.**

With **coordinate bases, on the other hand, the situation is different. For the problem** in **Figure 1, when z_x is used as the independent variable, Ypy^2 is obtained as the Y space move. This move, parallel to the Z2 axis, is almost identical to that obtained with orthogonal bases and the simplification in (2.13) is appropriate. However, if Z2 is chosen as the independent variable, the Y space move, given by Ypy^3 is much greater. In the limit, the Y space move is almost parallel to the space of the linearized equality constraint and Ypy becomes extremely large. Hence, the objective function (2.13) no longer adequately represents the situation.**

It follows that the performance of the coordinate method using (2.13) as the objective function of the QP subproblem depends to a large extent on the partitioning of the variables. We have observed inconsistent performance of the SQP algorithm with coordinate bases when solving a variety of test problems. Results illustrating this behavior are presented in the next section. Since it is difficult to determine a priori which are the "best" independent variables, and indeed; this choice may vaiy from one iteration to the next, it is essential to devise an alternate method to guarantee the robustness of the coordinate method if we are to exploit its advantages when solving large-scale **nonlinear problems.**

3. An Improved Coordinate Basis SQP Algorithm

In comparing orthogonal and coordinate basis methods, it is clear that the latter is simpler and cheaper. The term Y^h, which appears in the calculation of both py and the Lagrange multipliers of the equality constraints, reduces simply to V_yh. The Jacobian is untouched and the structure as well as any sparsity inherent to the model are maintained. The fact that the coordinate bases method requires less computational effort per iteration provides an additional benefit. The reduction is due to the fact that this approach does not require the matrix

$$[\, I + V^{*}V \quad I \qquad \text{where } V = (V_y h^{1})^{-1} V_x h^{T}$$

to be factorized, an operation which requires $O(n^2)$ + $O((n-m)^3)$ operations. While the dominant computational cost depends on the number of degrees of freedom in the problem, the difference in CPU time per iteration between the two decomposition strategies also becomes significant as the total number of variables becomes large. The only real drawback to the coordinate method is that in some instances it can exhibit poor convergence properties. If any of the linearized) constraints lies almost parallel to a coordinate direction, an unfortunate partitioning of the variables can lead to a very large Y space move which causes the simplification used to reduce the objective function of (2.9) to (2.13) to become invalid. Hence, robustness of the coordinate method Is not guaranteed. While the simplified objective function (2.13) is insufficient, exact evaluation of 2?BY may require too much effort for large problems. We propose an intermediate approach which utilizes an approximation of the term *ZJBYpy*- Through incorporation of the correction term we hope to enhance the robustness of the algorithm without compromising efficiency. We have studied three specific approaches for obtaining an approximation to Z^BYpy.

3*1 Finite difference correction

A first order approximation of Z^Ypy is given by a Taylor series expansion about the current point, x^.

$$\text{ZfcTBYp, = ZfcTVaUziJYpr - } Z^{\wedge}VLbk+Ypy) \text{ - } T^{\wedge}VU^{\wedge} \qquad (3.1)$$

where VL = Vϕ + Vh v . Since Z lies in the null space of Vh^T. Vh^T(Zk)Zk= 0, and (3.1) becomes

$$Z_k^T BYp_Y = Z_k^T V(t>(z_k+YpY) + Zk^Vhlzk+Ypy) v - ZjVQfad \quad (3.2)$$

Tests on a wide variety of problems have shown that this correction almost invariably improves the convergence rate of the coordinate method. However, the work involved may be considerable since an additional gradient evaluation at $z_k + Y_{py}$ is required at each iteration.

3.2 Broyden update correction

This method eliminates the need for an extra gradient evaluation. The rectangular matrix *Z^B* is approximated using Broyden's matrix update method. Letting A = 2^B_f the Broyden update is given by

$$A\text{«}i = Ak + \frac{(yk - A_k s_k) s_k^T}{s/sk} \quad (3\_3)$$

where s_k and y_k are defined as

$$s_k = z_{k+1} - z_k$$
$$y_k = Z_k^T (VL(z_{k+1}) . VLfeJl \quad (3\_4)$$

The Broyden matrix is initialized as Ao = (I 0] which is consistent with the initialization of the reduced Hessian approximation to B_o = I in the BFGS formula. The extra term in the objective function is obtained directly by postmultiplying A by Ypy. As long as the Broyden update formula yields a well-conditioned matrix, this method offers a viable alternative to using finite differences. Note though that additional overhead Is required for the storage of the matrix At.

3.3 limited memory Broyden update correction

Instead of using the Broyden update formula shown in (3.3) above, which requires storage of the full matrix At at each iteration, an approximation to the current matrix, A^!, can be obtained from a limited memory Broyden update (see Byrd and Nocedal, 1991). We define the update vectors St and y_t.

$$Si = z_{i+1} - z_i$$
$$y_t = Z_t^T [\nabla L(z_{i+1}) - \nabla L(z_i)] \quad (3.5)$$

The q most recent of these update vectors are stored as

$$S_{k+1} = [S_o \ldots . S_q] \qquad (3*6)$$
$$Y_{k+1} = (y_o \ldots . y_q]$$

Then $A^{\wedge}$ is calculated from

$$A_{k+1} = A_o + (Y_{k+1} - A_o Y_{k+1}) N_{k+1} r^J S_{k+1}{}^T \qquad (3.7)$$

where

$$(N_{k+1})_{ij} = \begin{cases} S_i^T A_j & \text{if } i \leq j \\ 1 & \text{otherwise} \\ 0 & \end{cases}$$

As for the full Broyden update method, we set Ao = [I 0]. Post-multiplication of both sides of (3.7) by Ypy allows us to approximate $^{\wedge}BY_{py}$ directly at each iteration without ever needing to construct the full matrix $A^{\wedge}$. Further, experience shows that 3 update vectors, i.e. q=3 in (3.6) above, are usually sufficient to obtain results comparable to the full Broyden method. This approach thus allows us to overcome the storage requirements of the previous method.

3.4 Update criterion

As mentioned above, using finite differences has the advantage of yielding consistent results while the Broyden methods are computationally cheaper. A natural extension is to include an update criterion which allows the algorithm to determine at which iteration it **requires the finite difference correction to yield satisfactoiy results, or if** the **Broyden correction is adequate. We adopt the update criterion proposed by Nocedal** and Overton **(1985) for the original, unconnected method.**

$$\| Y_{p_Y} \| < \overline{(k} J_{\overline{1}}{}^{v} \| Z_{pz} \| \qquad \textbf{(3.8)}$$

If (3.8) is satisfied, 11 Yp* 11 « 11 Zpz 11 and the Broyden update is used; otherwise the finite difference correction is selected. Biegler et *CLL* (1992), proved that when this criterion is used, the improved coordinate bases algorithm is 1-step superllnearly globally convergent. Figure 2 gives an outline of the main steps of the improved coordinate bases algorithm, including the update criterion.

3.5 Numerical Results

In this section, we briefly present sample results for three example problems. A more thorough numerical analysis is provided in Biegler *et oL* (1992). The first example is a 2-dimensional problem, with two variables and one constraint. The results are used to illustrate the effect of variable partitioning on the uncorrected coordinate bases method as compared to the orthogonal bases method. The results also demonstrate that this dependency can be largely eliminated through the addition of a second order correction term to the coordinate bases method. Examples 3.2 and 3.3 constitute generalizations of Example 3.1, allowing the effect of increasing the total number of variables and/or the number of degrees of freedom to be studied. CPU times **on** a SUN3 workstation are also included **to** assess **the** efficiency of the **various methods.**

Example **3.1**

$$\begin{aligned} &\textbf{Min} && \backslash \left(x_1^2 + x_2^2\right) \\ &\text{s.t.} && \textbf{Xjlx^-l) - 10X2 = 0} \qquad \textbf{(3.9)} \\ &\textbf{xp} = \textbf{0.1} && x_i^* = \textbf{0.0} \end{aligned}$$

This two dimensional problem has one degree of freedom; either x_x orxj can be chosen as the independent variable. Table 1 shows the total number of iterations required for convergence, using **a tolerance** of **10^{*15}.**

From the results in Table 1. it is apparent that the orthogonal bases method is relatively independent of variable partitioning. The coordinate bases method, on the other hand, requires almost twice as many iterations to converge when x^ is used as the independent variable compared to using x_x. When an approximation to $\mathbf{Z^T BY p_Y}$ is added, using either finite differences or a Broyden update, this trend is reversed and the dependence of performance on variable partitioning is reduced. For all the test cases, 3 update vectors were used for the limited memory update method. This has proved to be sufficient in most Instances to obtain results comparable to the full Broyden method. Tlie results in Table 1 clearly show that in terms of the number of iterations required for convergence, our improved coordinate basis algorithm is successful in overcoming problems associated with the pure coordinate method. The additional curvature information allows the algorithm to account for the fact that the Y space move may not be negligible compared to the Z space move. The effect of a poor choice of independent

variables is greatly reduced and the results are similar to those obtained with the orthogonal basis method.

Comparing the number of iterations required by the different correction methods, we see that the results using the Broyden correction are as good as those using the finite difference correction. For examples 3.2 and 3.3 we will see that in some instances the Broyden method even out-performs the improved coordinate bases method using finite differences. This trend is not true in general and may be attributed to the fact that, for this set of problems, the actual reduced Hessian at the solution is given by the identity matrix. **A°** = **[I 01** thus **provides** a **good initialization, favoring the Broyden update** method. **For models without this special structure, the finite difference correction tends to be more successful at resolving convergence problems caused by large Y space moves. However, it also requires more effort per Iteration since an extra gradient evaluation is required. To improve efficiency, we include the update criterion (3.8). If the criterion is satisfied, the Broyden update is used to obtain the correction term; if not, the algorithm resorts to finite differences. Thus, a compromise is met between the effort per iteration and the total number of iterations so as to yield the best overall results. In our experience, a good choice of the parameters** r\ **and v is given by**

$$\eta = \frac{\| Yp_Y \|}{\| Zp_Z \|} \quad \text{at the starting point and } 0;1 \pounds v \pounds 0.7$$

When the improved coordinate bases algorithm is applied to Example 3.1 with x^ **as the Independent variable, the update criterion is always satisfied. The Broyden update correction is used at every iteration for a total of 7 iterations. When Xj is chosen as the independent variable, the update criterion forces the algorithm to use a finite difference correction at the first two iterations, after which the Broyden correction is used for the remaining three. The total number of iterations is the same as when finite differences are used, but the extra gradient evaluation is only required for two out of the five Iterations. Consequently, using the update criterion yields a solution technique which requires less effort than either a pure finite difference or Broyden update method.**

Further, the inclusion of the update criterion Is necessary to prove that the improved coordinate bases algorithm is 1-step Q-superlinearly globally convergent. Figure 3 shows results for Example 3.1 using x^ as the independent variable. The ordlnate gives the ratio of the distance of the current point from the optimum to that of the previous point We compare the original, unconnected coordinate method to the improved method, using the update criterion. While the original method exhibits typical 2-step

superlinear convergence, the ratio for the corrected method decreases monotonically, reflecting the 1-step superlinear convergent nature of this algorithm.

Example 3.2

$$\text{Min} \quad \tfrac{1}{3} \sum_{i=1} Xi^2$$

$$\text{s.t.} \quad h, = xj\,(xj - 1) - lOxj = 0 \qquad j = 2 \ldots n \qquad (3.10)$$

$$x,^0 = 0.1 \qquad x_1^* = 0.0$$

Example 3.2 is a generalization of Example 3.1 and is intended to demonstrate the effect of increasing the total number of variables while maintaining only one degree of freedom. Table 2 gives the total number of iterations and the CPU time on a SUN3, for a total of 20 - 100 variables using $y^\wedge$ as the independent variable. The tolerance for convergence was set to 10^{-15}.

Example 3.3

$$\text{Min} \quad l \sum_{t=1}^{n} x\,t^2$$

$$\text{s.t.} \quad hj = Jfcn/2+j-1) - IOX^\wedge+J = 0 \quad j = 1 \ldots n/2 \qquad (3.11)$$

$$Xj^\circ = 0.1 \qquad Xj^* = 0.0$$

Also a generalization of Example 3.1_t this formulation allows both the total number of variables as well as the degrees of freedom to be increased. Variables $x_{n/2} + x$ to $y^\wedge$ were chosen as the Independent variables, which comprise one half of the total variables. The convergence tolerance was set to 10^{-15}. Table 3 gives the total number of iterations and the CPU time on a SUN3 for 20 - 100 variables.

The results reported in Tables 2 and 3 correspond to a poor choice of independent variables (large Y space moves) for Example 3.1 and 3.2 respectively. In terms of the number of iterations, the performance is very similar to that observed for Example 3.1. independent of the size of the problem. Again, the coordinate basis method requires significantly more iterations for convergence than the orthogonal method. When an approximation to $Z^T BYpy$ is included, calculated either via finite differences or using the Broyden update formula, the resulting improved coordinate basis method requires approximately the same number of iterations as the orthogonal method. As mentioned above, the Broyden method actually does slightly better than finite differences in some cases due to the special structure of the problem.

Let us now consider the total CPU time on a SUN3 until the convergence criterion is met. Even though the time per iteration is less for the coordinate method than for the orthogonal method, the extra iterations result in a greater total time. The results in Table 2 are for problems with only one degree of freedom. As expected, the Broyden update method is less expensive than using finite differences. While there is a significant improvement in total time as compared to the coordinate bases method, the results are not quite as good as for orthogonal bases. For so few degrees of freedom, the effort required to evaluate the correction term is comparable to the time necessary to invert the matrix in the orthogonal method, resulting in similar CPU times per iteration. As we increase the degrees of freedom, as shown in Table 3, the savings become **more pronounced** and our **improved** algorithm does **better** than both the orthogonal and uncorrected coordinate methods.

4. Tailoring the Reduced Space Algorithm

In the previous section, we demonstrated the effectiveness of a correction factor to improve the robustness of the coordinate bases decomposition method. After extensive numerical testing of the type outlined in section 3.5 above, we are confident that the resulting algorithm is sufficiently efficient and robust to apply to large, complex process engineering problems. In this section, we demonstrate how the optimization procedure can be coupled to a suitable equation solver which takes advantage of model structure. As will be shown, this results in considerable reductions In solution time. In particular, we focus on distillation calculations.

Two distinct approaches can be used to optimize the operating conditions of a distillation column; sequential or Integrated. This is illustrated in Figure 4. First, we select the independent variables for the system. Typical examples are the reflux ratio, the distillate rate, pressure and the flowrates of liquid and vapor side streams. In the sequential approach, these variables are fixed at an initial value, allowing the system of equations given by the material and eneigy balances on each stage to be solved. The independent variables are then updated using a suitable objective function. This procedure is repeated until optimality is achieved. The sequential approach requires that the model equations be solved to completion at each iteration. With an integrated approach, on the other hand, the process model is included directly as a set of equality constraints in the formulation of the NLP. The nonlinear model equations are then

solved as part of the optimization procedure. At present, most distillation optimization involves a sequential strategy since most existing optimization algorithms are inadequate to drive the complex models to optimality directly. As will be seen from the results presented below, our improved coordinate basis algorithm performs very well when used to optimize the separation of binary and ternary mixtures. We are able to obtain considerable reductions in the total number of iterations, as compared to a sequential strategy.

We can **still do better, however, at least in terms of the computational effort per iteration. General purpose SQP algorithms obtain the Y space move by applying Gauss elimination to the linearized equality constraints at each iteration. Instead, we propose to exploit the special structure of the mass and enthalpy balances for each stage of the column. Recalling the form of these equations, we realize that a relationship for** tray **i can depend at most on the state variables for stages i-1, i and 1+1. As a result, the Jacobian matrix for the model equations is quite sparse. The non-zero elements form a block tridiagonal structure as shown in Figure 4. In fact a major reason for the success of the sequential approach has been the development of highly efficient equation solvers with a special pivoting sequence that allows this block tridiagonal structure to be exploited. In particular, the Naphthali-Sandholm models UNIDIST and NRDIST, which are part of the SEPSIM process simulator (Anderson *et* aL, 1991), employ an efficient Thomas algorithm to obtain the Newton step for the distillation model. Our goal is to Incorporate this *existing* approach within the SQP framework to obtain the solution of the distillation equations. As discussed earlier, the coordinate basis algorithm is better suited for this Integration than orthogonal bases decomposition, since the Y space move is obtained directly from a Newton step for the solution of the model equations. Consequently, the structure of the Jacobian and any tailored procedure to calculate the Newton steps are fully exploited. Finally, coupling of the Naphthali-Sandholm solver with the optimization routine Is relatively straightforward. The UNIDIST and NRDIST Interfaces and data structures remain virtually the same and the physical properties need not be modified at all.**

4.1 Numerical results using the UNIDIST distillation model

Here, we observe the benefits of an integrated approach for the optimization of distillation columns, especially if the SQP algorithm is tailored to exploit the structure of the problem. We consider the separation of a binary benzene-toluene mixture and a ternary benzene-toluene-xylene mixture. These chemical systems allow the assumption of constant **molal overflow. Hence, for an initial investigation of the** behavior of **our algorithm, we can avoid the additional complication of energy balances** for each tray. **The distillation model within UNIDIST (see Appendix A), considers only** the mass **balances on each stage of the column and is sufficient to represent this** situation. **Temperature and composition effects are taken into account in the calculation of the equilibrium constants; we use the UNIFAC physical property** routines **that are part of UNIDIST.**

The problem formulations have *(N*C+2)* variables and *N*C* equality constraints, where *N* is the number of stages and C is the number of components. As decision variables we select the reflux ratio and the pressure within the column. The objective function to be minimized consists of the utility requirements for the column. It also includes a penalty term to ensure that the column pressure does not deviate excessively from 1 atm. The objective function is formulated as

$$\phi = Q_{condenser} + Q_{reboiler} + (P - 1)^2$$

Details of the model are described In Appendix A. Aside from the state equations, we also include purity constraints on the overhead and bottoms products.

$$\frac{v_{N,benzene}}{v_N}$$

$$\frac{l_{1,benzene}}{L_1} \leq_0.02$$

For both the binary and the ternary systems we look at two columns of different sizes with different amounts of feed. For the 12 tray column, feed Is introduced at the 5th plate while the 15th tray is the feed stage for the 36 tray column. No liquid or vapor side streams are considered. Further details on the initial conditions for the various cases may be found in Appendix A. Finally, the dependent variables were initialized at infeasible points, using the existing UNIDIST strategy for all of the cases we considered.

In Tables 4 and 5, we present the number of iterations and CPU time, respectively, required for convergence (tolerance = 10^{-6}). We consider the sequential approach as well as the integrated approach, both with a general dense Gauss elimination procedure and using the tailored approach for determining the Y-space move. Let us first consider the number of iterations. For the sequential approach, we include both the number of major (SQP) iterations as well as the total number of iterations. Since the model equations must be solved to completion for each major iteration, the latter number is much higher. Obviously, the **total number** of iterations **will** decrease significantly when passing from **the** sequential **to the integrated** approach. It **is** interesting to note that **for the four** sets of **results presented here, the number of iterations for the integrated approach is even less than the number of major Iterations for the sequential one. However, a comparison of iteration counts is not veiy meaningful since the effort per iteration is far less for the sequential approach. Instead, we need to look at the total CPU time, as given in Table 5. Here, even when a dense Gauss elimination procedure is used within the simultaneous method, the total time for convergence is consistently lower than for the sequential approach. When the algorithm is tailored to solve the linearized equality constraints more efficiently within the UNIDIST solver, the computational effort is reduced even further, especially as the size of the problems increases. Overall, there is over an order of magnitude reduction in the total CPU time and this savings increases with the size of the problem.**

4.2 Numerical results using the NRDIST distillation model

While the distillation model in UNIDIST was suillcient for the systems discussed in the previous section, here we will consider a system which requires the inclusion of the enthalpy balances. In particular, we look at the separation of the ternary benzene-water-ethanol mixture. The state equations are given by the model in NRDIST (see Appendix B). Temperature and composition effects are taken into account in the calculation of the equilibrium constants using the UNIFAC physical property routines within NRDIST.

Given a column with *N* stages and C components, the problem formulation includes *(N*(C+2)+2)* variables and *(N*(C+2))* equality constraints. We again use the reflux ratio and the pressure as independent variables. The objective function is similar to the one used in Section 4.1; the utility requirements of the column are to be minimized, which can be formulated as

• = Qcondcnscr + Qrcboilcr

The separation is carried out *in a 22* stage column, with two feed streams; on trays 18 and 21. We also include a purity constraint on the bottoms product, given by

$$\frac{ll_1 cthano1}{} \geq 0.9999$$

Further details on the initial conditions are included in Appendix B. The dependent variables **were** initialized at infeasible points, using the strategy provided within NRDIST.

Table **6 gives the total** number **of iterations** as **well as** the **CPU time on a** SUN3 **required** to converge **this problem within** a **tolerance of 10^{-7}. Results are given for both the** sequential **and the integrated approaches. While the two approaches require almost the** same **number of SQP iterations, the sequential approach requires an additional 543 Newton iterations in order to converge the state equations for the column at every major iteration. Consequently, the total CPU time for the integrated approach is considerably lower. When we use dense Gauss elimination to obtain the Newton step, we observe a 69% reduction in CPU time as compared to the sequential approach. When we tailor the SQP algorithm in order to exploit the special sparsity structure of the problem, we are able to reduce the time further by almost a factor of three. Overall,** there is almost **an** order of magnitude **reduction in CPU** time.

5. SUMMARY AND CONCLUSIONS

In this study we develop an SQP-based algorithm which is both efficient and robust when applied to existing complex, Newton-based process models. Here we develop an improved coordinate bases SQP algorithm which includes an update criterion that allows the method to determine whether a computationally inexpensive correction term (calculated via a Broyden update) is sufficient, or whether it must resort to a finite difference correction. This approach is 1-step superlinearly convergent and the results obtained on a set of test problems are very encouraging.

We then consider a specific class of process engineering problems, the optimization of the operating conditions of a distillation column. Our improved coordinate basis algorithm is particularly well-suited for such tailoring since the correction term

ensures robustness, while the definition of the Y-space maintains model sparsity. Thus, by coupling the optimization algorithm with an efficient equation solver developed to take advantage of the block tridiagonal structure of these problems, we obtain a tailored SQP algorithm which performs significantly better than existing methods. Moreover, this simultaneous approach takes advantage of the structure of process models and has the potential to consistently out-perform conventional "black-box" or sequential approaches to optimization.

Finally, **it is clear that this approach is not limited to the optimization of distillation columns. In particular, future work will employ a similar strategy to develop a tailored SQP algorithm for optimization of two point boundary value problems (using the Newton-based COLSYS package) as well as larger problems involving process flowsheets.**

ACKNOWLEDGEMENTS

Financial support from the Engineering Design Center, an NSF sponsored engineering research center at Carnegie Mellon University is gratefully acknowledged. Special thanks go to Dr. R. Gani for sharing the SEPSIM programs and to Prof. Jorge Nocedal for the use of the compact limited memory algorithm and for many useful suggestions regarding the correction term.

REFERENCES

Andersen, P. M., F. Genovese and J. Perregaard, "Manual for Steady State Simulator, SEPSIM,- Institut for Kemiteknik, DTH, Lyngby, Denmark (1991).

Berna, T., M. H. Locke and A. W. Westerberg, "A New Approach to Optimization of Chemical Processes," *AICHE* J., 26, p. 37 (1980).

Blegler L. T., J. Nocedal and C. Schmid, "Reduced Hessian Strategies for Large Scale Nonlinear Programming," Working paper (1991).

Byrd, R. and J. Nocedal, "A Compact Limited Memory Algorithm for Broyden Updates," Working Paper (1991)

Gabay, D., "Reduced Quasi-Newton Methods with Feasibility Improvement for Nonlineariy Constrained Optimization," Math. Programming Study, 16, p. 18 (1982).

Han, S-P, "A Globally Convergent Method for Nonlinear Programming," J. Opt. Theo. Applies, 22, p. 297 (1977)

Locke, M. H., R Edahl and A. W. Westerberg, "An Improved Successive Quadratic Programming Optimization Algorithm for Engineering Design Problems," *AICHE* J. Vol. 29, no. 5 (1983).

Lucia, A. and J. Xu, "Chemical Process Optimization Using Newton-Like Methods," 14, 2, p 119 *Computers and Chemical Engineering* (1990).

Nickel, R H. and J. W. Tolle, **"A** Sparse **Sequential Quadratic Programming Algorithm,"** *JOTA* **60, p.** 452 **(1989)**

Nocedal, J. and M. Overton, "Projected Hessian Updating Algorithms for Nonlinear Constrained **Optimization,"** SIAM **J. Num. Anal., Vol 22, no. 5, p. 945 (1985).**

Powell, M. J. D., "A Fast Algorithm for Nonlinear Constrained Optimization Calculations," ***1977 Dundee Conference on Numerical Analysis.*** **(1977).**

Vasantharajan S. and L. T. Biegler, "Large-Scale Decomposition for Successive Quadratic Programming," ***Computers and Chemical Engineering*** **(1988),**

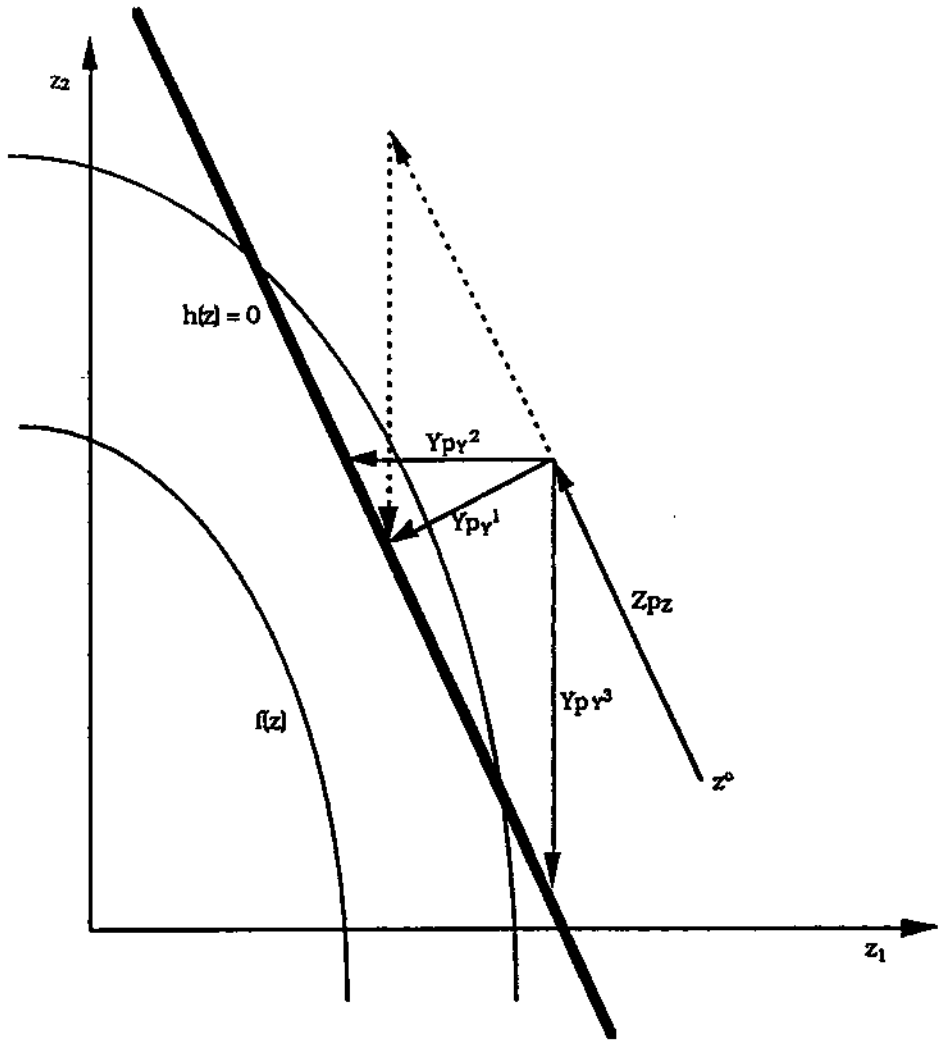


Figure 1. Geometrical Comparison of Various Decomposition Strategies

Legend

z_{lt} Z2 variables

f(z) objective function contours

h(z) = 0 linear equality constraints

z° current point

Zp_z Z space move

Yp_Y^1 Y space move for orthogonal bases

Ypy^2 Y space move for coordinate bases using Zi as the independent variable

Ypy^3 Y space move for coordinate bases using Z2 as the independent variable

——— Zpz and Ypy^3 modified due to addition of correction term

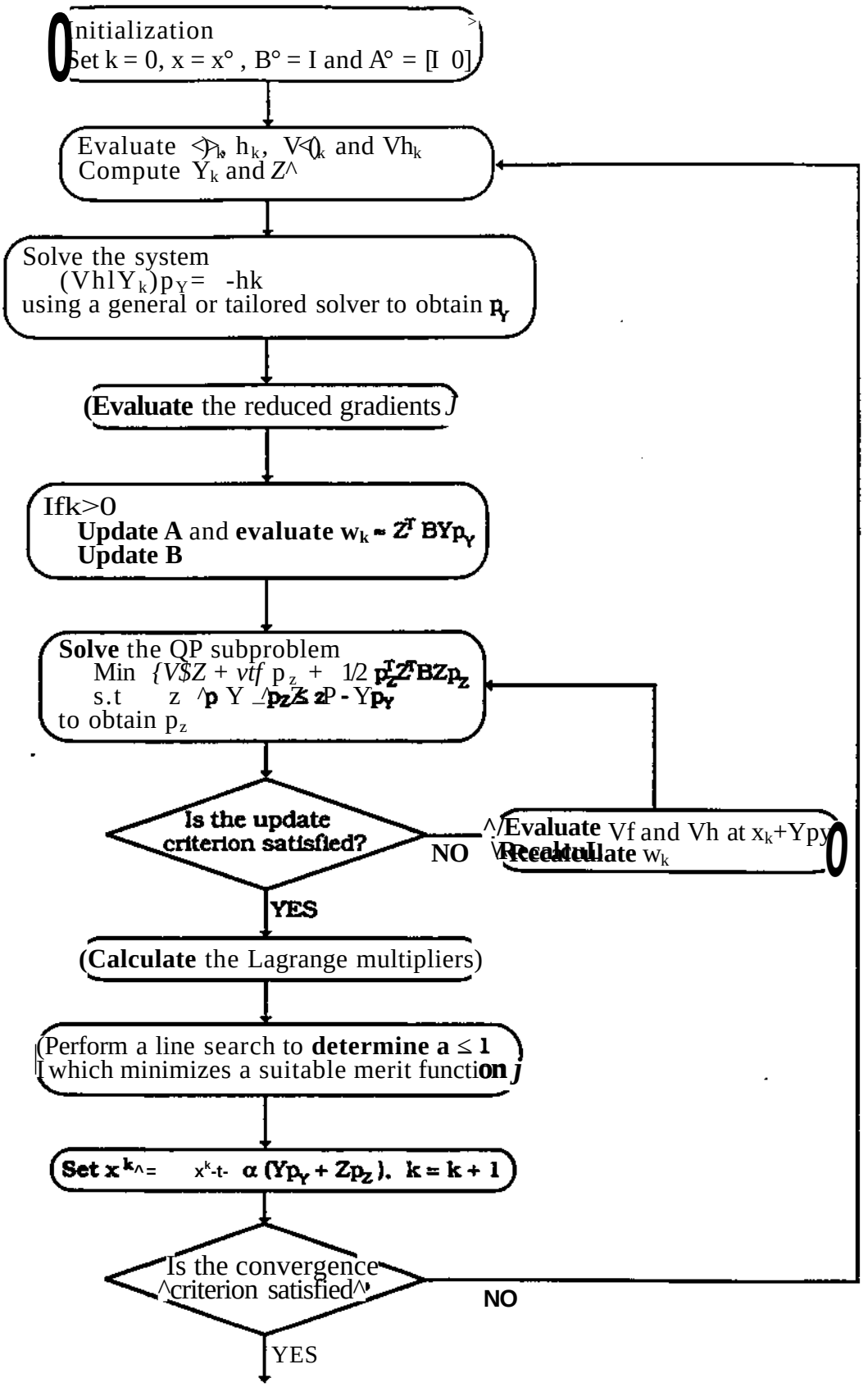


Figure 2. Improved coordinate bases algorithm

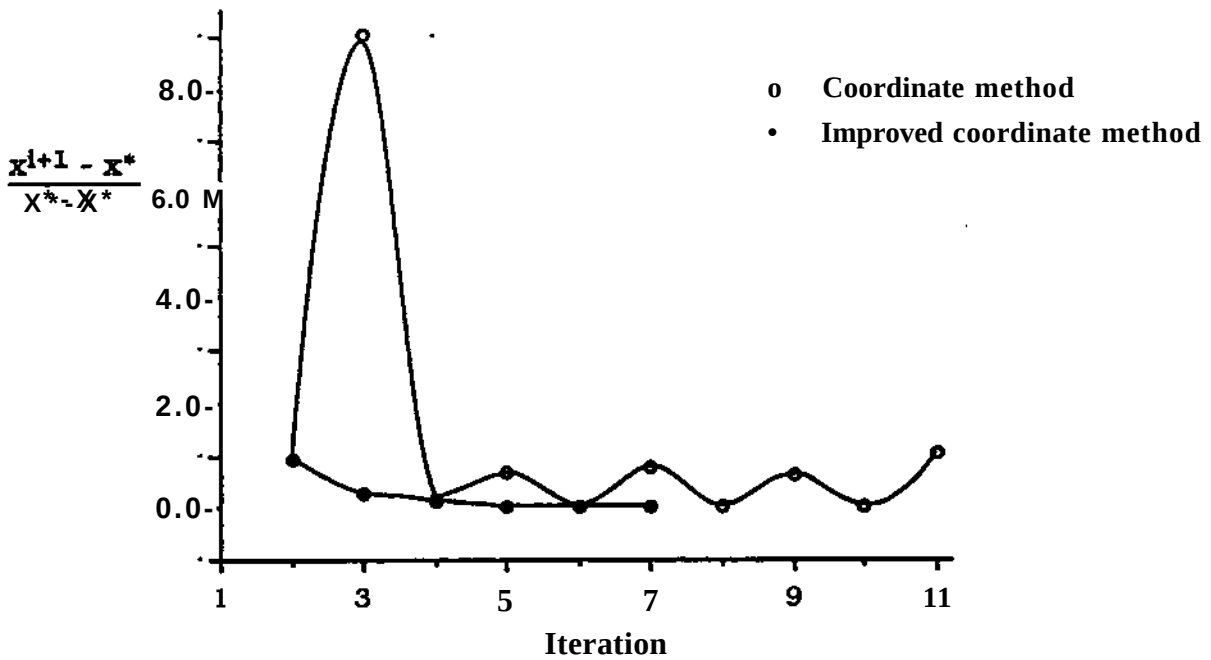
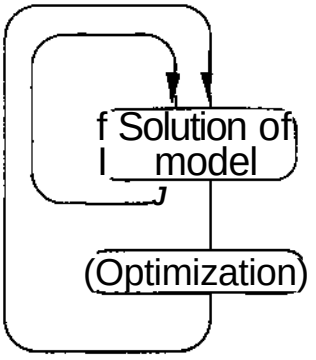


Figure 3. Comparison of the convergence properties

(a) Sequential approach: Convential modular mode



(b) Integrated approach: Equation **oriented** optimization

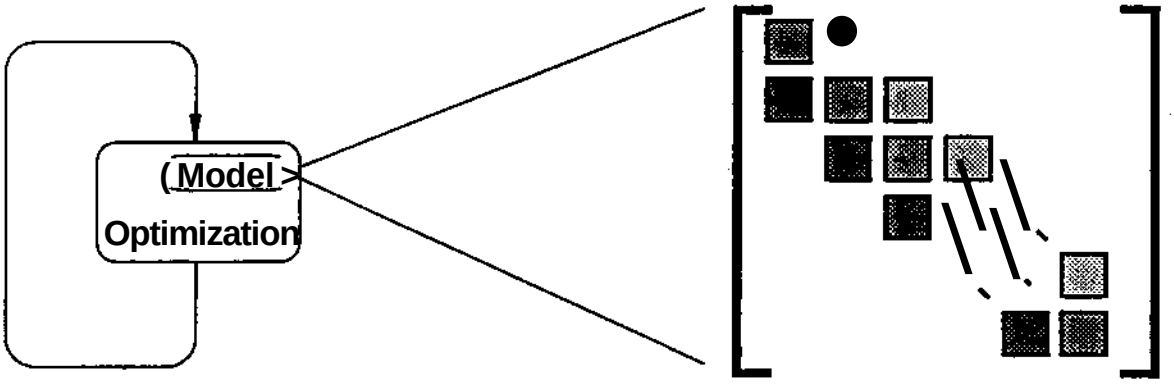


Figure 4. Comparison of sequential and Integrated approaches for distillation optimization

	Coordinate bases					
	Orthogonal bases	No correction	Finite difference correction	Broyden update correction	Limited memory Broyden	Update criterion
Independent variable = x_x	5	6	5	6	6	5
Independent variable = Xj	7	11	7	7	7	7

Table 1. Iterations until convergence for Example 3.1

Number of variables (Degrees of freedom)	Orthogonal bases		Coordinate bases: No correction		Coordinate bases: Finite difference correction		Coordinate bases: Broyden update correction	
	Its	Time	Its	Time	Its	Time	Its	Time
20(1)	8	3.9	10	4.6	7	3.7	8	3.7
40(1)	7	17.2	11	26.1	7	18.0	7	16.7
60 (1)	6	41.6	11	75.9	7	51.4	6	41.7
80(1)	6	94.5	11	168.5	7	112.4	7	107.5
100(1)	6	174.8	10	287.8	*T*	209.7	*T*	201.9

Table 2. Iterations and CPU time on a SUN3 for conveigence of Example 3.2

Number of variables (Degrees of freedom)	Orthogonal bases		Coordinate bases: No correction		Coordinate bases: **Finite difference** correction		Coordinate bases: Broyden update correction	
	Its	Time	Its	Time	Its	Time	Its	Time
20 (10)	9	5.5	14	6.5	9	5.1	9	4.9
40(20)	8	25.3	17	39.1	9	27.4	9	23.7
60(30)	9	73.4	18	119.1	10	90.4	9	66.2
80 (40)	8	160.3	18	232.0	10	221.0	9	140.3
100 (50)	8	299.8	17	458.8	10	377.2	9	258.6

Table 3. Iterations and CPU time on a SUN3 for conveigence of Example 3.3

Problem	Sequential approach		Integrated approach
	Major iterations	Total iterations	
2 components 12 trays	68	1760	16
2 components 36 trays	18	196	11
3 components 12 trays	16	222	10
3 components 36 trays	28	302	12

Table 4. Number of iterations for various distillation examples using the UNIDIST model.

Problem	**Sequential approach**	Integrated approach	
		General	**Tailored**
2 components 12 trays	**1022.8**	53.8	**42.9**
2 components 36 trays	**337.5**	223.6	**87.3**
3 components 12 trays	**210.1**	62.0	**42.5**
3 components 36 trays	**853.2**	**613.4**	**147.4**

Table 5. CPU time on a SUN3 for various distillation examples using the UNIDIST model.

Example problem 1: Separation of benzene-toluene using a 12 tray column

Feed stream:	Stage	Benzene	Toluene
	5	153.846 mol/s	195.654 mol/s

Distillate rate: 153.4 mol/s

	Initial point	Optimal solution
Reflux ratio	3.5	4.40
Pressure	1 atm	0.964 atm
Purity of benzene on top	0.753	0.980
Purity of benzene in bottoms	0.0266	0.0179
Objective function	10.919	17.543

Example problem 2: Separation of benzene-toluene using a 36 tray column

Feed stream:	Stage	Benzene	Toluene
	15	400 mol/s	500 mol/s

Distillate rate: 394 mol/s

	Initial point	Optimal solution
Reflux ratio	3.5	**1.48**
Pressure	1 atm	0.990 atm
Purity of benzene on top	**0.776**	0.990
Purity of benzene in bottoms	**0.013**	0.020
Objective function	**28.071**	20.516

Example problem 3: Separation of benzene-toluene-xylene using a 12 tray column

Feed stream:	Stage	Benzene	Toluene	Xylene
	5	153.846 mol/s	150mol/s	45.652 mol/s

Distillate rate: 153.4 mol/s

	Initial point	Optimal solution
Reflux ratio	3.5	3.59
Pressure	1 atm	0.972 atm
Purity of benzene on top	0.759	0.980
Purity of benzene in bottoms	0.0207	0.0179
Objective function	10.924	14.944

Example problem 4: Separation of benzene-toluene-xylene using a 36 tray column

Feed stream:	Stage	Benzene	Toluene	Xylene
	15	400 mol/s	410 mol/s	90 mol/s

Distillate rate: 394 mol/s

	Initial point	**Optimal solution**
Reflux ratio	3,5	1.3498
Pressure	1 atm	0.991 atm
Purity of benzene on top	0.776	0.990
Purity of benzene in bottoms	0.0130	0.020
Objective function	28.073	19.499

	Sequential approach	Integrated approach	
		General	Tailored
Iterations	8 (543 Newton)	9	9
CPU time on a **SUN3**	**1815.3**	**568.9**	**205.1**

Table 6. Number of iterations and CPU time on a SUN3 using the NRDIST model.

Appendix A* Distillation optimization using the UNIDIST model

Given a column with *N* stages and C components, we have a total of *NxC* relationships. At each stage, we have *C-l* component mass balances which may be expressed as

$$\left[\left(1 + \frac{S_n^L}{L_n}\right) + \zeta_{n,i}\left(1 + \frac{S_n^V}{V_n}\right)\right] l_{n,i} - \xi_{n-1,i}\, l_{n-1,i} \cdot 1_{n+1,i} \cdot f_{n,i} = o \qquad i = l,\ C\text{-}l$$

$l_{0,1}$ and 1_{N+U} **are set to zero. The total mass balance on each tray is given by**

$$\wedge_n \text{ " } \wedge_{n1} \text{ " } '*_{n2} \text{ " } — \text{ " } ' V_c = \wedge$$

The nomenclature used in the above expressions is as follows:

$l_{n,4}$	**molar liquid flowrate of component i on stage n**
L_n	**total liquid flowrate on stage n**
V_n	**total vapor flowrate on stage n**
fn_4	**feed of component i on stage n**
S_n^L	**liquid side stream on stage n**
S_n^v	**vapor side stream on stage n**
$L_{i,i,n}$	**$= K_{n17}^{V}\wedge$ where $K\wedge_t$ is the equilibrium ratio**

Appendix B. Distillation optimization using the NRDIST model

Given a column with *N* stages and C components, we have a total of *Nx(C+2)* relationships. At each stage, we have *C-l* component mass balances which may be expressed as

$$l_{n,j}\left(1+\frac{S_n^L}{L_n}\right)+v_{n,j}\left(1+\frac{S_n^V}{V_n}\right)-l_{n+1,j}-v_{n-i,j}-f_{n,j}=0 \qquad i=1,C,1$$

l_{04} and v_{N+14} are set to zero. At each stage we have a bubble point relation of the form

$$\sum_k K_{n,k}\,l_{n,k}-\sum_k l_{n,k}=0$$

as well as an enthalphy balance

$$(L_n+S_nL)\,h_n+(V_n+S_nV)\,H_n-L_{n+1}\,h_{n+1}-V_{n-1}H_{n-1}-Q_n=0$$

For the reboiler (n=l) and the condensor (n=N), the enthalpy balance is replaced by the specifications of the product flows:

$$\sum_k W-LN=0$$

$$\sum_k l_{1,k}-L_1=0$$

The nomenclature used in the above expressions is as follows:

$l_{n,k}$	molar liquid flowrate of component 1 on stage n
$L_n=\sum_j l_{n,j}$	total liquid flowrate on stage n
V_n	total vapor flowrate on stage n
$v_{n,k}=v_n\frac{K_{n,k}\,l_{n,k}}{\sum_k K_{n,k}\,l_{n,k}}$	molar vapor flowrate of component i on stage n
$f_{n,k}$	feed of component 1 on stage n
S_n^L	liquid side stream on stage n
S_n^V	vapor side stream on stage n
$K_{n,k}$	equilibrium ratio
h_n	enthalpy of liquid on stage n
H_n	enthalpy of vapor on stage n
Q_n	heat added at stage n

Example problem: Separation of benzene-water-ethanol using a 22 tray column

Feed streams:	Stage	Benzene	Water	Ethanol
	18	0.0 mol/s	5.1 mol/s	31.5 mol/s
	21	59.5 mol/s	20.92 mol/s	26.11 mol/s

Bottoms product: 28.7 mol/s benzene

Top product: 114.43 mol/s total

	Initial point	**Optimal solution**
Reflux ratio	**io-^{7}**	3xlO'^{8}
Pressure	1 atm	5.34788 atm
Purity of ethanol in bottoms	0,999827	0.9999
Utility requirements	25.03	24.34