A unified algorithm for flowsheet optimization

Y-D Lang
Carnegie Mellon University

Lorenz T. Biegler

Follow this and additional works at: http://repository.cmu.edu/chme
NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.
A Unified Algorithm For Flowsheet Optimization

by

Y-D Lang, L. T. Biegler

EDRC-06-18-86

September 1986
A UNIFIED ALGORITHM FOR FLOWSHEET OPTIMIZATION

V-D Laity and L. T. Biefler
Clujiulal Eifitiuunn Department
ClaiimMlo-McIion Univoiliiy
Pittsburgh, PA 15213

A flexible flowsheet (or flowsheet optimization) is developed on the FLOWTRAN process simulation. The optimization strategy combines the feasible and infeasible paid approaches as well as the simpler black box approaches. While the most efficient optimization strategy is often problem dependent, this paper presents guidelines that show which strategy is more efficient for a given problem. Also embedded within the algorithm is a new Broyden strategy for efficiently converging even complex flowsheets, without computing a new Jacobian. This allows for strategies "in-between" the infeasible and feasible path procedures. A ratio test based on the Kuhn-Tucker convergence test automatically and adaptively adjusts the optimization strategy.

The implementation on FLOWTRAN is discussed in detail and a number of examples are run to illustrate the flexibility of the implementation as well as demonstrate the effectiveness of the adaptive optimization strategy.

SCOPE

Flowsheet optimization has been an important area for chemical process design and has its origins in linear programming work in the early 60s (see Griffith and Stewart [13]). With the increasing use of flowsheeting tools, process simulation has become easier and more widespread. Currently, on commercial simulators, however, the optimization strategy is either ad hoc (i.e., often approached as a series of case studies) or involves a detached optimization algorithm that supplies sets of decision variables as new parametric cases to the process simulator. This "black box" approach was used by Giddy and coworkers [1.12] and Friedman and Pinder [11].
with direct search optimization strategies, as well as by Challand [18] and Freeman and Parker [11] with more sophisticated gradient-based optimization strategies. In these studies, any gradients that were required were evaluated by perturbing the decision variables and recalculting the entire flow sheet.

Recognizing that a complete flow sheet calculation is expensive for evaluating gradients, Isaacson [15] and Parker and Hughes [20] constructed reduced quadratic models by individual model perturbation at each base point. However, these required converged flow sheets for each trial point evaluation. Moreover, recent developments in nonlinear programming algorithms have cast flow sheet optimization in a new light. Using Successive Quadratic Programming (SQP) (Han [14], Powell [22]), Bertino, Locke and Westerberg [1] demonstrated with equation solving simulators that flow sheet convergence and optimization can proceed simultaneously. A number of researchers have applied this concept to sequential modular simulators [5, 9, 16, 17] with encouraging results. While differences exist among these studies in terms of calculating gradients and implementing the optimization algorithm all of them use an infeasible path approach to optimization; i.e. tear (or recycle) streams are solved simultaneously with the optimization problem using SQP to handle both tasks.

More recently, Biegler and Hughes [6, 7] advanced the concept of feasible variants, i.e. converging the flow sheet between SQP iterations. On a limited number of test problems this strategy generally required fewer SQP iterations. Reasons for this are not clear although it is easy to argue that converging the flow sheet at each iteration may help to correct problems with SQP resulting from inaccurate gradients or an inefficient line search strategy (such as the one originally proposed by Han and Powell). In more recent studies [3, 4], simply improving the line search algorithm and allowing for analytic gradient information where available also improved the performance of the infeasible path approach.

Finally, Kaisara [18] indicated that the feasible variant approach may not always be superior to infeasible path. This may be especially true if either the flow sheet is difficult to converge or the SQP algorithm has little difficulty in handling the infeasible path problems. Instead, he proposed a hybrid algorithm (IP) where the flow sheet is partially converged using a fixed number of Wegstein iterations between SQP iterations. Interestingly, this approach sometimes worked well even when compared to infeasible path. However, no criteria were given on how to choose the number of Wegstein iterations for partial convergence or how to apply this algorithm on flow sheets where the Wegstein algorithm may be inappropriate for convergence.

In this paper we develop criteria for intermediate flow sheet convergence and demonstrate this approach on a number of test problems. More importantly, however, this paper presents a unified strategy for flow sheet optimization within a fairly compact and easy to implement structure. Interestingly, this structure incorporates all approaches discussed so far and because of its implementation provides a great deal of flexibility in developing optimization strategies tailored to difficult optimization problems.

CONCLUSIONS AND SIGNIFICANCE

A flexible and efficient optimization strategy has been implemented and evaluated using the FLOWTRAN simulator. Due to the structure of the algorithms and in-line FORTRAN capabilities, the optimization implementation allows the following solution options:

- "black-box" optimization
- simultaneous convergence of recycle streams and design constraints using either Broyden or Newton methods.
- infeasible path optimization (IP)
- complete feasible variant optimization (CFV)
- partially converged flow sheet optimization with an embedded Broyden algorithm (EBOPT)
ID this policy we outline a simple donvaiOM for an improved "b-laco-l-ox" sirjiejy for recycle stream problems. A detailed ilioviation is then presented for an cnihddteJ Broyden method that partially converges flowsheets at intermediate uiiiiiiiiiuiui iterations. In addition, a heuristic strategy is presented that signals when pjM.j1 c silver gence 's desirable 01 not. The resulting EBOPT (Embedded Oruyden Optimization) method is fairly general and leads to the feasible path and feasible vonam algorithms as limiting cases.

The EBOPT strategy is compared to the infeasible path hybrid algorithm (IPH) developed by K.sala and some theoretical advantages of EBOPT are demonstrated. In particular, EBOPT can generally converge more complex flowsheeting problems more efficiently because of its Broyden capabilities. Also, EBOPT is not as prone to line search failures as IPH is.

Finally, the capabilities of the optimization implementation are demonstrated on five e*ampic problems. The first problem is characteristic of black-box optimization while the second illustrates the flowsheet convergence capabilities afforded by the EBOPT and IP algorithms.

The last three examples give a comparison of four algorithms, IP, EBOPT, CFV and IPH, on reasonably difficult and realistic flowsheet optimization problems. On all problems EBOPT performs better than either IP or CFV. IPH performs best on one problem but suffers premature line search failures on the other two.

1. Preliminary Theory and Concepts

The flowsheet optimization problem is given by

\begin{align}
\text{Min} & \quad F(x, y) \\
\text{st} & \quad h(x, y) \cdot y - w(x, y) \cdot 0 \\
& \quad c(x, y) \cdot 0 \\
& \quad g(x, y) \cdot 0 \\
& \quad x \in S \\
& \quad y \in Y
\end{align}

where 
- \( y \) - flowsheet decision variables
- \( w \) - calculated stream variables
- \( F \) - objective function
- \( h \) - leq- equations for converging the flowsheet
- \( c \) - additional equality constraints for optimisation
- \( g \) - inequality constraints

Examples of objective and constraint functions can be found in previous studies as well as in the case studies presented later in the paper. To solve this problem, the Successive Quadratic Programming algorithm essentially applies a modified quasi-Newton method to converge the optimally or Karush-Kuhn-Tucker (KKT) conditions of (MLP). To do this and maintain a consistent active set, the following quadratic program (QP) is solved at each iteration:

\begin{align}
\text{Min} & \quad v^T(x^l, y^l) \cdot d + \frac{1}{2} d^T B d \\
\text{st} & \quad b(x^l, y^l) \cdot \nabla(x^l, y^l) \cdot d \cdot 0 \\
& \quad c(x^l, y^l) \cdot \nabla(x^l, y^l) \cdot d \cdot 0 \\
& \quad s(x^l, y^l) \cdot H(x^l, y^l) \cdot d \leq 0
\end{align}

Here \( B \) is a BFGS (see (10)) update matrix to the Hessian of the Lagrange function with respect to \( x \) and \( y \). A detailed statement of this algorithm may be found in any of the above references and will not be given here. The version of the algorithm used in this implementation was developed in Biegler and Cuthrell [4] and includes the following features:
1. An adaptive augmented Lagrangian-based line search strategy is used to guarantee global convergence and allow full slopes in the region of the optimum. The latter property is not guaranteed by implementations of Norn (14) or Powell (22).

2. An automatic variable and constraint scaling strategy is included that gives good performance on flowsheeting problems. In addition a condition number is calculated for the 8 matrix in QP1 to determine when the problem is ill-conditioned or poorly scaled.

3. Because of portability, space and availability constraints, the Hal'weil subroutine VE02AD is used to solve the OP at each iteration. In Biegler and Cuivell (4) a more reliable and efficient OP code was used. However, there is no noticeable difference in function evaluations due to this substitution and, consequently, the FLOWTRAN implementation is not affected by this change.

This SOO algorithm forms the core of our unified optimization strategy.

To see how the optimization strategies compare from a geometric viewpoint, consider the sample flowsheet optimization problem in Figure la. Here only one decision variable, x, one target variable, y, and only one target equation, h, are required for the optimization problem. If one considers this problem from a case study perspective, one can trace a curve for Fix) vs. x (Fig. 1b) where each point on the curve represents a converged flowsheet. Expanding this problem in terms of both x and y yields the contour plot in Fig. 1c. Note that the optimum lies on the solid line which represents the target constraint and a nonlinear projection along this line gives the curve in Fig. 1b.

Using the case-study or "black box" approach the optimization algorithm is merely tied to the outside of the simulator and the simulator is responsible for converging the flowsheet for each evaluation of the optimization problem. Similarly, gradient calculations involve perturbation and convergence of the flowsheet for each decision variable. Thus, no information about flowsheet convergence is passed between the optimizer and simulator. In Figure 2 this can be seen in terms of the horizontal steps (in x) made by the optimizer and the vertical steps (in y) performed by the simulator. Note that these vertical steps usually represent flowsheet convergence by slowly converging lecya alnu' ilitinj mJ ih-lefore io)jesa-H He most lime consuming pen of He opl=>nieuen duOr

Since the infeasible path approach consititute information about the suHace and does not require flowsheet convergence until the optimum is found, movement occurs in both x and y as seen in Fig. 3a. This slap in x and y results from linearizing the constraints and approximating the surface contours as well as the curvature of the constraints. As shown above this approximation leads to a straightforward quadratic program. Gradients for (Q1) in found by perturbing the unconverged flowsheet. Also the expansive vertical steps for flowsheet convergence are avoided because flowsheet convergence is guaranteed as part of the solution to the optimization problem. In fact, as will be illustrated later, application of the infeasible path approach in the absence of degrees of freedom is equivalent to Newton's method.

To prevent an overextrapolation of the infeasible path approach it may be advantageous to ensure that the equality constraints be converged (or at least partially converged) at each iteration. Using the feasibility variant approach, the path for x and y is given by Figure 3b. Htm vertical steps from flowsheet convergence are introduced and one sees that the starting point for converging My) + 0 is given by the OP and is considerably better than in the "black-box" approach. Interestingly, the OP that is created and solved at each iteration is exactly the same, and requires the same effort at each iteration, as with infeasible path. Nota that with this strategy one assumes that the flowsheet can be solved readily by the racycla convergence algorithm.

In the next sections we develop a new approach for improving the performance of the optimization strategy. This strategy addresses some of the drawbacks with both infeasible path optimization and the feasibility variant strategy and also links both strategies more closely in terms of a unified framework. Before presenting this
strategy, however, it is useful to discuss further the differences between feasible variants and "black box" (or case-study) optimization.

2. Black Box Optimization vs. Optimization in x and y

Comparing Figure 2 to Figures 3a and 3b one sees that the main disadvantage in the optimization path of the first figure is due to lack of interaction between x and the dependent variables, y, in the optimization step. In fact, the main difference between Figures 2 and 3b is simply that with the feasible variant approach y is initialized much closer to the converged flowsheet. Generally, this leads to more efficient recycle convergence. The improved path however requires flowsheet perturbations in x and y to create a larger QP problem at each iteration. Using SQP with the "black box" approach requires the solution of a much smaller QP:

\[
\min \frac{d^1}{dx^1} \left( x_1 \right) + \frac{1}{2} d^T \left( x_1 \right) \left( x_1 \right) \]

s.t.
\[
g(x_1) + \frac{d^1}{dx^1} \left( x_1 \right) \leq 0
\]
\[
C(x_1) + \frac{dC}{dx} \left( x_1 \right) - 0
\]

where \( y = y(x_1) \mid h(x, y) = 0 \)

with the tear equations and tear variables removed. Note that the derivatives in the above QP are reduced gradients and require a converged flowsheet with each decision variable perturbation.

Because of the differences in the size of the optimization problem it is easy to see that the "black box" approach can be superior to the infeasible path or feasible variant methods when the number of tear variables greatly outnumbers the number of design variables and the flowsheet is not difficult to converge with conventional algorithms. The work per iteration for each approach can be approximated by:

Black Box

\[
\text{NFPI} \times \text{NRP} + \text{NX} + \text{NRC}
\]

Feasible Variant

\[
\text{NFPI} \times \text{E} + \text{NX} + \text{NY} + \text{NRI}
\]

Infeasible Path

\[
\text{NFPI} \times \text{E} + \text{NX} + \text{NY} + 1
\]

where:
- \( \text{NFPI} \) number of flowsheet passes per iteration
- \( \text{NRP} \) number of recycle iterations to converge perturbations in decision variables
- \( \text{NRC} \) number of recycle iterations to converge flowsheet at each new base point (vertical steps in Fig. 2)
- \( \text{NX} \) number of flowsheet decision variables
- \( \text{NY} \) number of flowsheet tear variables
- \( \text{E} \) fraction of equivalent flowsheet passes required for decision variable perturbations (partial flowsheet passes)
- \( \text{NRI} \) number of recycle iterations to converge flowsheet at new base point (vertical steps in Fig. 3b)

As seen from the above relationships, flowsheets with few degrees of freedom and many recycle components can be optimized more efficiently with the black box approach. Note also that \( \text{NRI} \) is expected to be less than \( \text{NRC} \). This occurs because the \( y \) variables have better initialization with the feasible variant approach and, as will be seen later, more efficient recycle convergence algorithms can be used with feasible variants. With the black box approach it is easy, however, to reduce \( \text{NRC} \) and allow the optimization path to be similar to the one followed by the feasible variant approach. This simply requires keeping track of how the dependent variables, \( y \), change with \( x \).

Consider a perturbation in variable \( x \) and a completely converged flowsheet for that perturbation. For the tear constraints, \( h(x, y) = 0 \), we have to a first order approximation:
where \( dx \cdot (0, 0, ..., 0) \). Solving this equation for \( dy/A^* \), give a column of the Hessian matrix \( V M x_i y^j \). Therefore, simply by saving the response of the \( y \) variables to all perturbations in \( x \), i.e.

\[
X = (dy, dy, dy, ..., y) \cdot (V M x_i y^j) \cdot y^j.
\]

we can use the solution of (QP2), \( d^x \), and write \( d_y \) as

\[
d_y = Y d^x.
\]

Note from QP1 that this equation solves the linearization of the tear constraints and therefore leads to the step in \( x \) and \( y \) given in Figure 3b. Also, it is interesting to note this approach leads to the same step that is generated by the Reduced Feasible Variant (RFV) algorithm described by Biaglar and Hughes (7).

However, because flowsheet convergence is the outer loop to several levels of iterative calculations, the convergence error in the tear equations can be relatively large. Therefore, in order to calculate the \( Y \) matrix correctly, the perturbation size needs to be chosen accurately. If we include second order corrections and the convergence error, \( \epsilon \), in our tear constraints we can write:

\[
dh = \epsilon V A J T \cdot (\partial h/\partial x)^T \cdot \partial y / \partial x^T \cdot \partial y / \partial x^T \cdot \partial h / \partial x^T\]

Rearranging this expression gives an order of magnitude estimate for the errors in the \( Y \) matrix:

\[
Y^* = \epsilon Y \cdot \partial y / \partial x^T \cdot (X|A^K|) \cdot \partial h / \partial x^T\]

Note that choosing a perturbation \( ax \) too small can lead to appreciable error due to convergence noise while a large perturbation \( ax \) leads to an error due to second order effects. To avoid these problems, use number of iterations \( \text{NIP} \) and \( \text{NIFC} \). may need to be large to force a small \( \epsilon \). Tilt it into box several flowsheet flows. convex (the) algorithms have only non-zero values and \( p^* \text{op}=\text{h} \). Where lies the target MP1 to be larger than it normally expects.

3. Development of an Embedded Broyden Strategy

To summarize the previous material and to introduce this section, consider problem (NLP) again:

\[
\text{(NLP) Min } F(x, y) \quad \text{s.t. } M(x, y) \cdot y - w(x, y) \cdot 0 \\
\quad c(x, y) \cdot 0 \\
\quad g(U, y) \cdot 0 \\
y \in \mathcal{Y}
\]

In the "black box" approach the \( y \) variables were eliminated and the constraints, \( xU, y \cdot 0 \), were always satisfied. \( \text{mv}r \) for perturbations of \( x \). The infeasible path (IP) and the complete feasible variant (CFV) algorithms deal with (NLP) explicitly in the space of \( x \) and \( y \) and solve (QPI) at each iteration. In addition, CFV converges the equations, \( h(x, y) \cdot 0 \), by adjusting the \( y \) variables iteratively. As mentioned above, it may be more efficient to solve the IP converging the constraints, \( h(x, y) \cdot 0 \), at each iteration since IP yields a converged flowsheet at the optimum anyway. If an efficient and reliable algorithm can be applied, one can handle both \( h \) and \( c \) by converging them simultaneously.

In this section we present an embedded Broyden approach for partial convergence within flowsheet optimization. This strategy incorporates the infeasible path and feasible variant approaches as limiting cases and can be viewed as a modification of the hybrid approach proposed by Kisala (18). In the previous section we observe that slowly converging recycle algorithms can lead to
inefficiency with the black box optimization algorithm. These algorithms log. 
WiMjbiom o- direct substitution) are also used in the feasible variant and hybrid 
approaches. Thus, for flowsheets that are difficult to solve, one can e-rloi 
convergence problems at intermediate points.

When tear variables and constraints are part of the optimization problem, we 
have enough gradient information from QP1 to allow also for more efficient 
convergence routines. In particular, Broyden routines have been used with good 
results (9, 19, 21) in converging complex flowsheets, even those with additional 
design constraints. This section therefore addresses two points.

1. HOW can a Broyden algorithm be embedded within the optimization 
strategy to (partially) converge the flowsheet at intermediate points?

2. What criterion should be used to decide whether (partial) convergence is 
necessary at intermediate points?

3. An embedded Broyden algorithm

we now derive a modified Broyden algorithm for partial convergence at 
intermediate points. For convenience we will use only the tear variables, y, and leer 
equations. nU,y) > 0. In presenting this derivation. Application of this method to 
additional design constraints, cU.y) = 0, and additional dependent variables is 
straightforward.

To converge or partially converge the equality constraints, consider the step in 
Figure 4. At point C, the gradients and values of the objective and constraint 
functions are evaluated and QP1 is constructed and solved. The search direction 
from QP1 and a suitable stepsize leads to point D, from which the equality (i.e. tear 
and any design) constraints may be converged. If one were to apply a Broyden 
method to converge the dependent variables at this point one would want to have 
the Jacobian (V^H)^T at point C to initialize the Broyden method. Since this is not 
available and it would be expensive to construct this information, we derive an 
update strategy based on the gradients evaluated at point C.

Let H* = [V^H]^T [V^H]^{-1} at point C and consider the Broyden formula (10):

\[ H_{k+1} = H_k + \frac{\Delta y^T}{\Delta x^T} \cdot H_k \cdot \frac{\Delta x}{\Delta y} \]

We note that this update relation can also be applied to the nonsquare matrix without 
violating any assumptions (see Dennis & More [10]) as to its derivation. Also from 
QP1 we see that the step from C to D is generated by H'd · t(e(x, y)). We can now 
apply the update formula to get H' at point 0:

Starting from point O we keep x constant and only change y to converge the 
flowsheet; thus i = 0 for the following iterations. Applying (B1) to H' gives the 
following relations:

\[ H_{k+1} = \begin{bmatrix} x^k \mid y^k \end{bmatrix} + \frac{(e(x, y) - y')}{\Delta y} \cdot \begin{bmatrix} x \mid y \end{bmatrix} \]

and

\[ \begin{bmatrix} x^k \mid y^k \end{bmatrix} \begin{bmatrix} 0 \mid J \end{bmatrix} = -H(e(x, y)) \]

Note that since J is determined by QP1 for the first step and H does not affect \( y \) 
in the later steps, we write the update formulae (B2), (B3) es:
This expression can be simplified further by noting that
\[ \mathbf{H}^{i+1} = \mathbf{H}^{i} + \left( \begin{bmatrix} \mathbf{I} & \mathbf{H}^{i} \end{bmatrix} \right)^{-1} \frac{\mathbf{g} - \mathbf{H}^{i} \mathbf{y}^{i}}{\mathbf{y}^{i}} \]

and
\[ \mathbf{H}^{i+1} = \mathbf{H}^{i} + \left( \begin{bmatrix} \mathbf{I} & \mathbf{H}^{i} \end{bmatrix} \right)^{-1} \frac{\mathbf{g} - \mathbf{H}^{i} \mathbf{y}^{i}}{\mathbf{y}^{i}} \]

for \( i \geq 1 \)

Testing of this approach on the problems given below shows that no more than 5 iterations are required to converge the flowsheet at intermediate points. As shown with the conventional Broyden approach, this method also handles the infeasible constraint easily.

3.2 Criteria for using the feasible Broyden method

As noted in previous studies, the choice of optimization strategy is often problem dependent. If the gradients are reasonably accurate and the flowsheet is only "mildly" nonlinear, then the infeasible path approach will converge smoothly. If, on the other hand, the flowsheet is highly nonlinear and difficult to converge (with complex units that are failure prone) then a feasible path approach with Broyden's method and appropriate safeguards could be more reliable and efficient. However, these characteristics are not always known a priori. Indeed, it is shown by Kisala [18]. partial flowsheet convergence may lead to more efficient performance in solving the optimization problem.

However, care must be taken in dealing with partial flowsheet convergence at each iteration. In particular, partial convergence can be detrimental to the line search algorithm in determining the stepsize for the next point.

In the SOP algorithm, a given stepsizes along the search direction, \( \mathbf{d} \), is accepted if a "sufficient" decrease is observed with some merit function, \( p \). In the algorithms of Han and Powell, this function is the exact penalty function; in our algorithm an augmented Lagrange function is used. In either case, it is well known (see e.g. Han 114)) that the search direction from QP1, \( \mathbf{d} \), is a descent direction for the merit function. Consequently, finding a nonzero stepsize is guaranteed, at least in theory, for the infeasible path algorithm.

For the feasible variant algorithms, one can also prove that a nonzero stepsizes will be found during the line search. This can be shown because all points in the line search have converged equality constraints and the IP solution, from a feasible point, is also a descent direction for the line search function \( f \).
However, lot flowsheets that are only partially converged, ont can guarantee illiat a stepsue with a decreased merit function will be found. A simple illustration of this is given in Figure S. From the QP base point at A, one sees the merit function can be viewed as a function of I along the search direction found by OP 1. Executing a fixed number of convergence iterations for a given stepsise, point B, say, may decrease or increase the objective function. In fact, for a fixed number of iterations, it is possible that the equality and inequality constraint infeasibilities may also increase. Consequently, it is possible that partial convergence may move the merit function value from point B to B'. Note that this behavior can occur arbitrarily close to point A, say, point C and thus lead to a line search failure - even though perfectly reasonable stepsizes exist for infeasible path.

For this reason we apply partial convergence only after the line search algorithm finds a stepsise. In this way we can avoid line search failures and also save some work at intermediate points.

As further justification for this safeguard we note that, by itself, the infeasible path algorithm converges quickly and takes full steps in the neighborhood of the optimum. For this case, partial convergence is usually not necessary. On the other hand, at the beginning of the optimization, the search direction may overextrapolate and lead to a point that is difficult to converge. Here it would be inefficient to partially converge the flowsheet during the line search. Instead, allowing the line search algorithm to find a more reasonable point first will save some effort.

Even with this safeguard, one is still not guaranteed that partial convergence leads to better performance for the optimization. One way to measure the success of partially converged points would be to compare merit functions from iteration to iteration. However, one still needs to know if, a priori, partial convergence is desirable or even necessary at a given point. In the next section we develop a strategy for dealing with this task. We should mention that this strategy is based on heuristics and, consequently, will not always guarantee improved performance compared to infeasible path.

Heuristic strategy for partial convergence

For SOP a common measure of Kuhn Tucker error (KTE) for problem INLP is given by (22):

$$KTE = \| \nabla f / \|_v \cdot K^{*}, \|_u \cdot \| x \cdot \| \cdot \| \cdot \| \cdot \|,$$

where $u$, $v$, and $l^*$ are multipliers calculated for QP1 for $g$, $h$, and $c$, respectively. Also, $(U, V, Y)$ is defined as max $(0, g^x, y^M)$. Reducing KTE to within a zero tolerance is a necessary and sufficient condition for satisfying the KKT conditions for (NLPL). Now, from Figure 4, if SOP finds a step from point C to point 0, one needs to determine if additional work is required by Broyden’s method to move to point E. Since this method converges only $c_i$ and $h$ a heuristic measure of how much improvement can be had is given by the ratio:

$$\text{URS} = \left( \sum_i \| u_i \cdot \| + \sum_i \| v_i \cdot \| \right) / \text{KTE}$$

If this ratio remains small, intermediate convergence is not necessary. If it remains consistently large, however, full or partial convergence may help to speed convergence. To use this ratio we propose two triggers for intermediate convergence.

$$\left( \| \nabla \cdot \| \cdot \| \cdot \| \cdot \| \cdot \| \right) / \text{KTE}$$

at point C. move to point D and converge $c_i$ and $h$ (to point E, say) until the relation

$$\left( \sum_i \| u_i \cdot \| + \sum_i \| v_i \cdot \| \right) / \text{KTE}$$

is satisfied.
if \( \left( \sum |v_i^A| + \sum |v_i^r| \right) / \text{KTE} \leq \varepsilon_1 \),

but on moving to point D,

\( \left( \sum |v_i^A| + \sum |v_i^r| \right) / \text{KTE} > \varepsilon_1 \),

use Broyden's method to converge to point E, say \( \mathbf{c}_1 \) and \( \mathbf{h}_1 \) so that (BR1) is satisfied.

Otherwise, both points C and D lie close to the constraints and intermediate convergence is not required. Note that by adjusting the \( \varepsilon \)'s, one can develop a full spectrum of methods between the infeasible path approach (\( \varepsilon_1 = 1, \varepsilon_2 = \infty \)) and the feasible variant strategy (\( \varepsilon_1 = 0, \varepsilon_2 = 0 \)).

In choosing these parameters, \( \varepsilon_1 \) should be set between zero and one to allow for partial convergence. \( \varepsilon_2 \) should be set small in order to avoid the first trigger at the next iteration. Our experience indicates that often very few Broyden iterations (1 or 2) are required to satisfy (BR1) even if \( \varepsilon_2 \) is small, (say 10^{-6}). \( \varepsilon_2 \) on the other hand, can be sufficiently greater than unity without hampering performance. This results because point C for the second trigger is sufficiently close to satisfying the constraints; a linearization from that point and a line search usually determine point D that is reasonably good without partial convergence. In fact, for the problems we solved, the second trigger for partial convergence was not necessary for good performance.

In addition to the above triggers we have also included the following conditions for intermediate convergence. First, if the current iteration is in the neighborhood of the optimum, applying Broyden iterations is usually not necessary since the constraints are close to being satisfied anyway. Therefore if \( \text{KTE} \leq 10 \varepsilon \), say, where \( \varepsilon \) is the Kuhn-Tucker tolerance, we do not apply intermediate convergence.

Also, it is possible that the flowsheet may not converge at all at an intermediate point. Consequently, we impose a maximum number of iterations for intermediate convergence. In our case studies successfully converged intermediate points never required more than 5 iterations.

We conclude this section by emphasizing that the above strategy is based on heuristics that, from our limited experience, work reliably and efficiently. Since very little theory governs the concept of partial convergence, we used these guidelines in our implementation. In the next section we discuss how the constants \( \varepsilon_1 \) and \( \varepsilon_2 \) were chosen and give a statement of the algorithm.

4. Algorithmic Statement and FLOWTRAN Implementation

Using the concepts stated above we now present an algorithmic statement of the Embedded Broyden Optimization (EBOPT) strategy and outline the features and options used in the FLOWTRAN implementation. In the algorithmic statement we assume the reader is somewhat familiar with the SOP algorithm and will not dwell on its details. The reader is referred to [4] for the line search strategy and update formulae.

4.1 Algorithm

Step 0) Set the SOP iteration counter, \( I = 0 \), and initialize the flowsheet with \( x^* \) and \( y^* \). \( x^* \) can be found by (partially) converging the flowsheet. Set \( \varepsilon \) as the Kuhn-Tucker tolerance.

Step 1) At \( (x^*, y^*) \) find the gradients for \( F, g, h \) and \( c \) with respect to \( x \) and \( y \). This can be done by direct loop perturbation [8] or chain ruling [3].

Step 2) Solve (QP1) given above to get the search direction \( d \) for \( x \) and \( y \). Evaluate KTE and (BR1) at iteration \( I \) using the expressions above. If \( \text{KTE}, \leq \varepsilon \), stop.
Step 3) Perform a Una search with a suitable merit (unction, f. to find a stop sue. 1. along d. Oef.ne \( \sum \gamma \cdot k \cdot d \cdot y \cdot x \cdot d \).

Step 4) Solve Broyden iteration counter \( k = 0 \) and evaluate the flowsheet (or \( \gamma \cdot k \cdot d \cdot y \cdot x \cdot d \)).

\[
\begin{align*}
1 \left( \sum_{i=1}^{n} \left| \frac{\partial f_i}{\partial x_i} \right| \right) & \left( \sum_{i=1}^{n} \left| \frac{\partial f_i}{\partial y_i} \right| \right) / \gamma \cdot k \cdot d \\
& \text{set } y'' \cdot y \text{ to step 7.}
\end{align*}
\]

Step 5) Set \( y'' \cdot y \) and apply the Broyden formulae \( (5) \) and \( (8) \) to \( y'' \) and \( y'' \).

\[
\sum_{i=1}^{n} \left| \frac{\partial f_i}{\partial x_i} \right| \cdot \sum_{i=1}^{n} \left| \frac{\partial f_i}{\partial y_i} \right| / \gamma \cdot k \cdot d
\]

then set \( y''' \cdot y'' \). If the above relation cannot be satisfied after five iterations \( k > 5 \), set \( y'' \cdot y \).

Step 6) Evaluate the gradients at \( (x''', y''') \) as in step 1. Update the Hessian matrix for QP1.

Step 7) Let \( i = 1 \) and go to step 2.

4.2 FLOWTRAN Implementation

The optimization capability in FLOWTRAN was installed by writing a type 2 (convergence) block. The structure and argument list for this block, called SCOPT, was the same as the existing recycle convergence block, SCVW. Because we did not change any code in FLOWTRAN, we implemented direct loop perturbation as the most straightforward way for evaluating gradients. Since FLOWTRAN generates end compiles a FORTRAN main program at run time, it can easily accommodate in-line FORTRAN and user written subroutines as part of the input data. This, in turn, allows the optimizer to evaluate partial flowsheet passes if needed for gradient evaluation.

Aim. The user's specification of the optimization problem can be made simply by a few lines of in-line FORTRAN to the input data for the problem.

SCOPT handles up to four phase streams (as does SCW), which it converges simultaneously, and up to a total of 40 decision and lever variables. Decision variables can be chosen from equipment parameters or feed streams. These variables are accessed through PUT statements that are common features in FLOWTRAN.

In addition, the user needs to specify a relative perturbation size (between \( 10^{-5} \) and \( 10^{-7} \) is recommended) and a relative Kuhn-Tucker tolerance (usually between \( 10^{-8} \) and \( 10^{-6} \)). It should be noted, however, that choosing a Kuhn-Tucker tolerance too small can result in line search failures and poor steps near the end of the run, because the gradients may not be accurate enough to satisfy the tolerance.

Another option in our implementation deals with the choice of test variables. In most optimization studies, stream flow rates, pressure and specific enthalpy are chosen. Temperature is not chosen because for multiphase streams, enthalpy is not realizable from temperature alone. However, for single phase streams calculation of enthalpy from temperature is usually direct and a level of iteration and some convergence noise are eliminated in the perturbation step. Since perturbing the temperature for single phase stream can lead to more accurate derivatives, we have added a T/H option.

Finally the \( T \) parameters need to be specified for the embedded Broyden algorithm. As mentioned above, \( f \), which defines the second trigger, can be fairly large. In our experience values of \( f > 3 \) have still led to good performance end this trigger was always inactive. Consequently, we have not used this test ellipt.

On the other hand \( f \) was set after some testing, to 0.01. As explained above, it takes surprisingly few Broyden iterations to satisfy this test. The most Important
parameter for determining intermediate convergence is therefore \( \eta \). Setting \( \eta \eta \eta = 0 \)
in our implementation leads to a feasible variant approach. Setting \( \eta \eta \eta = 1.0 \) yields the
infeasible path approach and intermediate values of \( \eta \) allow partial convergence. In
our testing, setting \( \eta \) to 0.4 yielded good results although this parameter is problem
dependent. However, on many problems, examination of the output shows that
performance of this strategy is not very sensitive to \( \eta \). In Table 1, the ranges of
\( \eta \) and \( \eta \) under which the same performance would be achieved are tabulated for the
Embedded Broyden Optimization (EBOPT) strategy.

5. Example Problems

The following five example problems were solved by a number of approaches.
A number of similar problems were solved in addition to these. However, for the
sake of brevity, we chose this set because it represents what can be expected from
the implementation in terms of performance and flexibility. The first problem is
essentially a black-box implementation on a single unit. The second problem
illustrates how the infeasible path and Broyden methods can be used to converge
flowsheets. The last three problems deal with moderately-sized flowsheets, some
with complex models. These allowed comparison of the embedded Broyden strategy
(EBOPT) with a number of recent and efficient optimization strategies. Due to the
flexibility of the algorithm and features of FLOWTRAN, none of these strategies was
difficult to implement.

Problem 1 - Black Box Optimization

The first problem deals with a single unit optimization of a 25 tray distillation
column with sidestreams. As illustrated in Figure 6, the distillation column problem,
which is solved by a Thiele-Geddes model (FRAKB), seeks to maximize the degree of
separation of its 5 components among its overhead, bottoms and sidestreams. The
decision variables are the fraction of feed to the two sidestreams and the distillate.
The only constraints are bounds on the decision variables as well as bounds on the
fraction of feed to the bottoms stream.

This problem is typical of many simple process optimization problems. Since
there are no recycles, SQP deals with this model in "black-box" fashion and solves
it completely each time it requires a function evaluation. Alternatively, an entire
flowsheet could easily have been treated instead of a single unit. The solution of
this problem is also given in Figure 6. This problem was solved to a relative Kuhn-
Tucker tolerance of \( 10^{-6} \).

Note from Table 1 that the performance of the optimization algorithm is
characteristic of the black-box approach. Because the model needs to be solved
several times it is not surprising that over 16 Simulation Time Equivalents (STE's
measured at the starting point) were required to optimize this three variable problem.
Because of the tight tolerance, seven iterations appears to be reasonable for this
case.

Problem 2 - Cavett Problem Simulation

To demonstrate the capability of the infeasible path and EBOPT methods for
Newton and Broyden convergence, respectively, we selected a modified form of the
Cavett problem, reported by Rosen and Pauls [23]. Here the number of stream
components was reduced from 16 to 11. Figure 7 illustrates the flowsheet where 21
and 22 were chosen as tears and Table 2 lists problem data and the converged
solution. This problem was first solved using the Wegstein convergence block in
FLOWTRAN with all of the default options. In this case 13 iterations were required
to converge the flowsheet to the default relative tolerance of 0.0005. Using the
same tolerance, this flowsheet was also converged using the infeasible path (IP) and
EBOPT methods in 4 and 6 iterations, respectively. However, comparing STE's for
this problem shows that these methods are not competitive with Wegstein. The
EBOPT method requires 26 flowsheet passes to construct a Jacobian matrix at each iteration, the IP incurs only 10 convergence Jacobian at each iteration.

Because of the effort required for the initial Jacobian, Broyden's method may not always be competitive for solving simulation problems. Embedded within an optimization strategy, however, where the Jacobian is calculated anyway, the Broyden method performs much more efficiently.

For the next three examples, we compare the IP and EBOPT strategies with the CFV (Complete Feasible Variant) [7] and IPH Unfeasible Path Hybrid [18] algorithms. The last two algorithms were implemented by using FLOWTRAN's Wegstein convergence block to (partially) converge the flowsheet between SOP iterations. For IPH, two Wegstein iterations were used between every SOP iteration, as suggested by Kisala [18], for CFV, the flowsheet was either converged to FLOWTRAN's default tolerance or until 30 Wegstein iterations had been exceeded.

On all problems, relative tolerances of $10^{-4}$ were used for the Kuhn-Tucker error. All problems were recycle flowsheets with complex unit operations and nonideal thermodynamics.

Problem 3 - Ammonia Process A

This problem was adapted from Parker and Hughes [20] and has been used in other studies [19, 18]. The problem statement is given in Figure 8 and in [20]. Because of different thermodynamic properties and fewer decision variables, values of the objective function are slightly lower in this study. The starting point and optimal solution for this problem were given in Table 3. As seen from Table 1, the double loop flowsheet with an equilibrium-based reactor is fairly easy to converge and optimize. Here the EBOPT and CFV approaches are close in performance, although EBOPT is slightly superior. Because no intermediate convergence was applied for the IP run, more iterations were required than with EBOPT. Interestingly, EBOPT, with the heuristic strategy described in Section 3, only had to use Broyden strategy after iterations 1 and 3. After this, SOP resulted in no improvement converging the process by itself.

Unfortunately for this problem, the IPH algorithm suffered a line search failure after 5 iterations. Restarting at this point resulted in a second line search failure after 3 additional iterations. In his study, Kisala [18] also reported a line search failure for Parker's ammonia problem. The reason for this, as explained in Section 3, may be that, because a fixed number of Wegstein iterations was applied for each function evaluation in the line search, a descent direction cannot be guaranteed and this method can be prone to failure.

Problem 4 - Methylichlorobenzene Process

This problem is adapted from an example in the FLOWTRAN manual [24]. Using the default costs and prices in the costing blocks, the optimization problem illustrated in Figure 9 was formulated. Nine decision variables were chosen for the optimization. These were listed along with their initial and optimal values in Table 4.

Because this problem contained a rigorous (and often unreliable) absorber model and the FORTRAN code for FLOWTRAN was not available to us, we were unable to provide error returns to the optimization algorithm and thus continue in the event of unit convergence failures. Obviously, error returns are a necessary feature in the implementation of any flowsheet optimization strategy, and the lack of this capability reflected how we could solve this problem.

From the results in Table 1, one sees that the EBOPT strategy required less effort than either the CFV or the IP strategies. However, to prevent premature termination due to failure in the absorber block, Intermediate recycle convergence was suppressed for EBOPT during the first two SQP iterations. The EBOPT algorithm
nee.led to apply Broyden's method only after iteration 4 in order to get satisfactory performance.

Also, due to 10% difficult! with the absorbent, the IP algorithm could not be converged from the starting point for EBOPT. From a slightly different starting point (shown in Table 4), 12 iterations were required to satisfy a Kuhn-Tucker tolerance slightly above 10⁻⁶. Again, because of the unreliable nature of the process units, a better and more consistent comparison could not be made.

The CFV algorithm required over 5 times the computational effort that EBOPT required. This represents the difficulty that SCVW has to converge this flowsheet at intermediate points. In fact, for SOP iterations 1, 2, 5 and 0, CFV required the maximum of 30 iterations without converging the flowsheet at these base points.

Again, as with the previous problem, IPH terminated with a line search failure after 12 iterations. This could be due to the descent direction line search problem explained in Section 3.

Problem 5 - Ammonia Process B

The flowsheet for this problem is given in Figure 10 along with the problem statement. The decision variables and their initial and optimal values are given in Table 5. Unlike problem 3, this ammonia process has a single loop design with 3 flash units. The unit operation and cost blocks for the reactor are taken from Chapters 9 and 10, respectively, of the FLOWTRAN manual. To make the problem more interesting, feed rates were chosen as decision variables and a constraint was imposed on the flow rate of the ammonia product. This type of constraint can be treated in a straightforward manner by all four of the algorithms compared.

Because of problems with error termination in FLOWTRAN, we suppressed the Broyden option for the first iteration in the EBOPT run. Even so, this run, as seen from Table 1, required only 47% of the effort as the infeasible path algorithm. Instead, Broyden iterations were applied after the 4ih, 5ih, 8ih and 9ih SOP iteration. However, the acceleration after the 3ih, 4ih and 5ih SOP iterations were not effective (and also did not lead to closer points) because they tended to using more than 5 Broyden iterations without satisfying the ratio test. This illustrates the difficulty of converging this flowsheet from intermediate points.

Similar, but more pronounced results were encountered with the CFV algorithm. Here the convergence algorithm was unable to converge the flowsheet for the first three SOP iterations. For these points the maximum of 30 Wegstein iterations was exceeded and, consequently, CFV required a lot of computational effort. On the other hand, the IPH algorithm did very well for this problem. Because it uses a fixed number of recycle iterations at intermediate points, the progress of the optimization was better than IP, but none of the convergence problems encountered with CFV, or to a lesser extent, with EBOPT, were observed here. Also for this problem there were no apparent difficulties with line search failures.

In summary, partial convergence of the flowsheet at intermediate optimization iterations led to better results on all of the recycle optimization problems than with either the IP or CFV algorithms. However, as shown in section 3, care must be taken to implement this strategy properly. Therefore, this study illustrates the potential of the EBOPT strategy for flowsheet optimization, although further work may be required to tune the algorithm for specific problems.
REFERENCES


(24) Seader, J.D., W.D. Saidar and AX. Pauls. FLOWTRAN Simulation - An Introduction. 2nd edition. CACHE Corp. (1977)
<table>
<thead>
<tr>
<th>Study Case</th>
<th>Objective Function</th>
<th>No. of Iterations</th>
<th>CPU Time (second)</th>
<th>STE's (6.74 seconds/STL)</th>
<th>STL's (6.74 seconds/STL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Disti 1/JUUI</td>
<td>7.33361</td>
<td>7</td>
<td>111.09</td>
<td>16.48</td>
<td></td>
</tr>
<tr>
<td>2. CJVCC problem</td>
<td></td>
<td>4</td>
<td>461.27</td>
<td>145.19</td>
<td>36.86</td>
</tr>
<tr>
<td>3. Ammonia A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. MCB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Ammonia B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Objective Function

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>(-3.67128)</td>
</tr>
<tr>
<td></td>
<td>(-3.23)</td>
</tr>
</tbody>
</table>

### Design Variables

1. Inlet Temp of Reactor \(147.226\) 180
2. Press of Reactor \(2950\) 3000
3. Inlet Temp of High Pressure Flash \(-24\) -24
4. Purge Fraction \(0.0802432\) 0.10
5. Conversion of Nitrogen \(45\) 41.0

### Tear Variables

1. Flowrate
   - \(N_2\): 2193.02 2170.0
   - \(H_2\): 568.426 612.0
   - \(NH_3\): 30.1334 16.3
   - \(Ar\): 85.7868 70.5
   - \(CH_4\): 104.604 125.0
2. Temperature 289.883 215.0
3. Pressure 2950.0 3000.0

### Design Variables

1. Inlet Temp of Reactor \(32.0401\) 32.0
2. Press of Reactor \(31.0401\) 31.0
3. Split Fraction of Absorber \(0.377398\) 0.33
4. Split Fraction of Absorber \(0.622602\) 0.67
5. Inlet Temp of Flash 227.422 270.0
6. Outlet Temp of Heat Exchanger 100.0 120.0

### Tear Variables

Flowrates:
1. HCl 0.0 0.0
2. Benzene 0.623078 0.0
3. HCl 82.1304 80.0
Temperature 100.00 120.0 100.00
Pressure 50.0 50.0 50.0
### Table 5: Result of Ammonia B Problem

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Optimum</th>
<th>Starting Pt.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>Objective Function</td>
<td>-24.9421</td>
<td>-20.859</td>
</tr>
<tr>
<td>B.</td>
<td>Design Variables</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. Inlet Temp. of Reactor</td>
<td>400.0</td>
<td>400.0</td>
</tr>
<tr>
<td></td>
<td>2. Inlet Temp. of 1st Flash</td>
<td>65.0</td>
<td>65.0</td>
</tr>
<tr>
<td></td>
<td>3. Inlet Temp. of 2nd Flash</td>
<td>35.0</td>
<td>35.0</td>
</tr>
<tr>
<td></td>
<td>4. Inlet Temp. of Recycle Compressor</td>
<td>75.3139</td>
<td>107.0</td>
</tr>
<tr>
<td></td>
<td>5. Purge Fraction (%)</td>
<td>0.814030</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>7. Flowrate of Feed 1</td>
<td>2626.66</td>
<td>2632</td>
</tr>
<tr>
<td></td>
<td>8. Flowrate of Feed 2</td>
<td>688.206</td>
<td>1648</td>
</tr>
</tbody>
</table>

### Tear Variables

<table>
<thead>
<tr>
<th></th>
<th>Flowrate</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$\text{H}_2$</td>
<td>1303.88</td>
<td>1648.0</td>
</tr>
<tr>
<td></td>
<td>$\text{H}_2$</td>
<td>3921.21</td>
<td>3676.0</td>
</tr>
<tr>
<td></td>
<td>$\text{NH}_3$</td>
<td>551.760</td>
<td>424.9</td>
</tr>
<tr>
<td></td>
<td>Ar</td>
<td>183.708</td>
<td>143.7</td>
</tr>
<tr>
<td></td>
<td>$\text{CH}_4$</td>
<td>2095.34</td>
<td>1657</td>
</tr>
<tr>
<td>2.</td>
<td>Temperature</td>
<td>75.3139</td>
<td>60</td>
</tr>
<tr>
<td>3.</td>
<td>Pressure</td>
<td>1901.49</td>
<td>1930</td>
</tr>
</tbody>
</table>

![Figure 1](image-url)

Figure 1: Simple flowsheet optimization.
Figur 3a

Inftaalblt PathOP) Optimisation Strategy

Figur 2

Black Box Optimisation Strategy
Figure 3b
Feasible Variant Optimization Strategy

Figure 4
Partially Converged Strategy for EBOPT

Contour of \( f(x,y) \)

\[ h(x,y) = 0 \]
Figure 5
Source of Line Search Failures with Partial Convergence

Figure 6
Distillation Column Optimization

Initial
- Obj. Fun - 6.42
- Sl/F - 0.2
- S2/F - 0.2
- OH/F - 0.3

Optimum
- Obj. Fun - 7.554
- Sl/F - 0.1
- S2/F - 0.2499
- OH/F - 0.3353
Partial Cavett flow - 11 components (see on 4 Paula)

- Clean with He 10 w/s method (IP)
- Charge with IR Br's method (EBOFT)
- MIID green of fr dom
Max (Before tax profit t 15x over 5 years)
s.t. NH3 in purge + 4.5 lb aola/hr
No liquid In compressors
100,000 tons NH3/yr
KH, Product Purity i 99.9X
1.8 i H2/N2 In S 3.3 combined feed