1983

Improved infeasible path optimization for sequential modular simulators.

Lorenz T. Biegler
Carnegie Mellon University

S Shivaram

IMPROVED INFEASIPLE PATH OPTIMIZATION FOR SEQUENTIAL MODULAR SIMULATORS PART I: THE INTERFACE

by

L.T. Diegler * S. Shivaram

December, 19^3

DRC-O6-M1-R3

Improved Infeasible Path Optimization for Sequential Modular Simulators Part 1: The Interface

by

L.T. Biegler and S. Shivaram

Working Paper - Preliminary Results

Recently, it was shown that chemical processes modeled by steady-state simulators could be optimized without repeatedly converging the process simulation. Instead, optimization and simulation of the process can be performed simultaneously (along an infeasible path), thus leading to much more efficient performance.

In this two-part study, we describe several improvements to this infeasible path approach. This first paper deals with improvements to the interface between the optimization algorithm and the process simulator. Here we are primarily concerned with obtaining the necessary functional and gradient information for the optimization with minimum simulation effort for the process. Thus, we consider the architecture of sequential modular simulators, the structure of process optimization problems and any sources of error in obtaining the necessary information for the optimization algorithm. To this end, we derive a chainruling algorithm that allows the incorporation of analytic derivative information for parts of the flowsheet and generally leads to less frequent evaluation of the process flowsheet.

This algorithm is demonstrated on three process optimization problems. The results indicate significant improvement in performance.

SCOPE

Simulation of chemical processes by flowsheeting programs has become a readily accepted and effective tool for process design and analysis. In virtually all commercial environments general-purpose simulators are based on the sequential modular approach. Here, all

equations and relationships defining the process are grouped into modules, each corresponding to some unit operation in the flowsheet. The order in which the modules are calculated normally parallels the flow of material in the process and is thus rigidly defined by the topology of the flowsheet.

While sequential modular process simulations are relatively easy to construct, analyze and run, the rigidly defined calculation sequence makes them useful only when doing performance or rating calculations. Design or optimization studies, on the other hand, are usually prohibitively expensive because the simulator lacks the flexibility in calculation order to accommodate them. To allow for greater flexibility when doing process simulation and optimization, several prototype simulators have been developed using equation-solving strategies (see e.g. Perkins (1983)). While these simulators easily accommodate design and optimization calculations, none has yet been able to handle problems as large as those routinely handled by sequential modular simulators. Also, several hybrid simulators that incorporate the desirable features of the above strategies have been developed. Termed simultaneous modular (see Biegler (1983)), this strategy has more flexibility than the sequential modular mode, has handled fairly large process problems, and appears to have great promise in future flowsheeting programs.

The inflexibility of sequential modular simulators is usually encountered when information flow reversals appear in the flowsheet. In design problems, for example, any process specification that cannot be introduced as a flowsheet input parameter must be determined iteratively. Process optimization can be thought of as an extended design problem in which decision variables must be manipulated to find **the** optimality conditions for the problem. To perform the optimization, sequential modular simulators can be very inefficient if

a feasible simulation is required for every function evaluation. To solve these problems more effectively, several researchers (Biegler and Hughes (1982), Chen and Stadtherr (1983), Hutchison, et al (1983)) proposed the infeasible path approach. Here the tear variables and recycle equations of the process simulation are added to the optimization problem. Thus, recycle convergence, which is usually the most inefficient part of the process simulation problem, is embedded within the optimization procedure; satisfaction of the recycle equations and convergence to the optimum occurs simultaneously. Although this strategy has been efficient and effective in solving process optimization problems, some problems in the implementation of this strategy can impair the efficiency of the method. This paper addresses the interface between the process simulator and the optimization algorithm. Here we need to consider the structure of the process optimization problem and determine functional and gradient calculation strategies that require fewer time consuming flowsheeting calculations while still maintaining the accuracy required for the optimization algorithm.

Background

To motivate the development of an improved interface, we briefly review the basis of the infeasible path strategy. Consider the block flowsheet given in Fig.1. Here each block represents a module or "black box" procedure. For simulation and optimization, the convergence or optimization algorithm is totally unaware of the process relationships within the modules. The only information that generally can be determined is the response of a module to prespecified inputs.

The simulation problem consists of solving the recycle equations $h(x,y) = y - w(x,y) = 0$. All other process equations are solved implicitly by the process modules and stream interconnections in order

to calculate w. To converge, we merely manipulate y .so that w and y match*

The process optimization problem posed by the engineer is usually expressed in the following form:

$$\min_{x} 0(x)$$

$$\text{s.t. } g(x) \leq 0$$

$$c(x) = 0$$

Here, he has selected a set of decision variables x, an objective function *ff(x)*, typically of economic form, and a set of process limitations or product specifications, g, and design constraints, c. To solve the optimization problem, the engineer naturally assumes that the process simulation problem has been converged before evaluating the constraint and objective functions. For the infeasible path algorithm this is not required. Instead, we augment the above problem by adding tear variables and recycle equations to form:

$$\min tf(x,y)$$

$$\text{s.t. } g(x,y) \leq 0$$

$$c(x,y) = 0$$

$$h(x,y) = y - w(x,y) = 0$$

Because of the "black box" nature of the flowsheet, the optimization problem must be defined explicitly by a set of parameters accessed through the simulator; we choose a vector of retention variables, r, to explicitly calculate the objective and constraint functions. From Figure 1, it is easy to see that these dependent variables are implicit functions of y and x. Thus the problem that we will deal with for flowsheet optimization is given by:

$$\text{Min } ^(r(x,y),\ x)$$

$$\text{s.t. } g(r(x,y),\ x) \leq 0$$

$$c(r(x_f y)_f\ x) = 0$$

$$l»(x,y) = y - w(x,y) = 0$$

$$V^{X} * X_{u}$$

$$y_{l} \leq y \leq y_{u}$$

To solve this problem, any optimization algorithm that handles nonlinear equality constraints, without requiring their convergence for each function evaluation, can be applied. Among the most efficient of these are the MINOS (Murtagh and Saunders (1978)) and SQP (Han (1977), Powell (1977)) algorithms. MINOS is designed to handle large, "mostly linear" problems efficiently since it performs a full optimization in the subspace of the constraint normals. However, since many of the active constraints are nonlinear tear equations, several constraint linearizations will be required before converging to the optimum. The SQP algorithm, on the other hand, minimizes a quadratic approximation of the Lagrangian with each constraint linearization. The resulting quadratic programming problem (QP) is formed using only one gradient and function evaluation from the nonlinear problem. The Hessian of the Lagrangian in the QP is derived from quasi-Newton updates (see Dennis and More, 1977).

Thus, the SQP algorithm solves a much simpler problem than MINOS for each linearization of the constraints. Previous studies (Powell (1977), SchittJcowski (1981), Stadtherr and Chen (1983)) have shown that SQP outperforms most of the other nonlinear programming algorithms on small problems, although their studies did not include MINOS.

At each iteration the following quadratic program is formulated and solved:

$$\underset{d}{\text{Min}} \quad ,l((z^{i})^{T}d + \frac{I}{2} d^{T} B d$$

$$\text{S.t.} \quad q(z^{l}) + Vg(z^{i})^{T} d \leq 0$$

$$c(z^{i}) + \nabla c(z^{i})^{T} d = 0$$

$$h(z^{i}) + V h(z^{1})T d = 0$$

$$z_{I} £ z £ z_{u}$$

$$z = \begin{bmatrix} x \\ y \end{bmatrix}$$

The QP solution, d, is then used as a search direction for the next point. A stepsize, X # is next chosen along this direction for which some merit function, measuring objective function improvement and constrianit infeasibility is minimized. Most studies have chosen the exact penalty function

$$(P(x) = f(x) + \sum_{j=1}^{m} r_j \max[0, g_j(x)] + \sum_{j=1}^{meq} s_j |h_j| \quad (*II)$$

as the merit function and have reported encouraging results. Others (Chamberlain et al (1979), Biegler and Hughes (1982), Chen and Stadtherr (1983)) have used the watchdog algorithm which uses either a Lagrangian or exact penalty function for the line search.

We defer further discussion of the SQP algorithm and our improvements over existing methods for our companion paper (Biegler and Cuthrell (1983)). In this paper we use the watchdog algorithm used in Biegler and Hughes (1982) and present a more efficient method for obtaining gradient and function information, from the process simulator. Specifically, we discuss strategies for chainruling derivative information, calculating and incorporating analytic Jacobians, and selecting the "best" flowsheet tear set for optimization.

Gradient Calculation Strategies

To obtain the gradient information for the QP, one can easily take advantage of the calculation,sequence prespecified by the engineer. The most straightforward way would be to perturb the entire flowsheet once the recycle streams are torn. Here a flowsheet pass is required to evaluate r and w, and is repeated for each perturbation of x and y. This strategy, termed direct loop perturbation, was described in a previous paper (Biegler and Hughes (1982)). Note from Figure 1 that design variable perturbations usually require

only partial flowsheet evaluations since modules and dependent variables upstream of design variables are unaffected by perturbation. For direct loop perturbation, the number of module evaluations required for gradient evaluation is:

$$NBE = \sum_{j=1}^{VCS} \left(\sum_{i=1}^{NT} T\,(NCP + 2)\ Jtij + \sum_{j=1}^{ND} £\ \text{ℓij} \right]$$

where
- NBE - number block evaluations
- NCP - number of components (component flow rates, pressure and enthalpy are perturbed because they thermodynamically define a stream)
- NT - number of tear streams
- tt$_{ij}$) - number of blocks from tear or design variable i to the jth terminus in the calculation sequence.
- ND - number of design variables
- VCS - number of termini of calculation sequence (number of tear streams plus any retention variables downstream of all tear streams).

As seen from the formula, gradient calculation using direct loop perturbation can be prohibitive. This method, however, is very easy to apply since it merely involves repeating the simulation calculation sequence for each perturbation. Consequently, one can implement the optimization routine merely by substituting an "optimization block" for the recycle convergence blocks.

A harder to implement gradient calculation strategy employs the concept of chainruling. Writing the gradients for the QP in terms of intermediate values gives:

$$V \text{ " } \ast\!\!- \ast\!\!- \qquad \text{for } Y = \wedge, {}_{8}, h \tag{2}$$

$$\nabla_x h = -\frac{\partial w}{\partial x}$$

$$v_y h - i \ -\frac{\partial w}{\partial y}$$

The partial derivatives $\frac{\partial \Psi}{\partial x}$ and $\frac{\partial \Psi}{\partial r}$ can be specified analytically since ϕ, g and c are written explicitly in terms of x and r. The derivatives $\frac{\partial w}{\partial x}$, $\frac{\partial w}{\partial y}$, $\frac{\partial r}{\partial y}$ and $\frac{\partial r}{\partial y}$ can be constructed by chaining Jacobian matrices for each unit according to the calculation sequence of the flowsheet. From Figure 1, it is easily seen that:

$$\frac{\partial w}{\partial y} = \frac{\partial y_1}{\partial y} \frac{\partial y_2}{\partial y_1} \cdots \frac{\partial y_n}{\partial y_{n-1}} \frac{\partial w}{\partial y_n} \qquad (3)$$

$$\frac{\partial w}{\partial x} = \frac{\partial y_k}{\partial x} \cdots \frac{\partial w}{\partial y_n}$$

$$\frac{\partial r}{\partial y} = \frac{\partial y_1}{\partial y} \frac{\partial y_2}{\partial y_1} \cdots \frac{\partial r}{\partial y_m}$$

$$\frac{\partial r}{\partial x} = \frac{\partial y_k}{\partial x} \cdots \frac{\partial y_m}{\partial y_m - 1} \frac{\partial r}{\partial y_m}$$

Note that the variables y and x initiate chains and the variables r and w terminate them. Although this method is more difficult to apply on sequential modular simulators than direct loop perturbation, it offers several advantages. First, the number of block evaluations for gradient calculation is:

$$NBE = NCS(NCP+2) + ND \qquad (4)$$

where NCS is the number of internal streams in the calculation sequence.

Note that design variable perturbations require far less effort since their derivative information is chained through modular Jacobians. These savings can be considerable if decision variables are encountered at the beginning of the calculation sequence. It should be mentioned, however, that chainruling requires extensive changes to the simulator executive with additional overhead for matrix manipulation, retrieval and storage. Also, the relative efficiency of chainruling

over direct loop perturbation is clearly dependent on the structure of the flowsheet and the optimization problem. For example, Fig. 2 presents a simple flowsheet where direct loop perturbation is significantly more efficient than chainruling.

The main advantage of the chainruling strategy is that it allows the prespecification of full or partial analytic Jacobians. While many unit operations do not have Jacobians that are easily calculated, some or all values of $\partial y_i / \partial x_j$ can be determined analytically for some units. These are described in the next section.

Analytic Jacobians

In this section we wish to explore the following concept:

Given that we know the type and structure of a particular module and its input and output streams, how many perturbations can be saved ?

Clearly, if nothing is assumed about the module, all input stream perturbations (flowrates> pressure and specific enthalpy) are required. One easily sees that all perturbations are required fpr module k even if only one row of $\partial y_i / \partial x_j$ cannot be derived analytically. Figure 3 gives Jacobians for six module types: mixer, splitter, simple component separator, valve, pump and compressor. (The last two module Jacobians require additional pressure perturbations to be calculated completely). More Jacobians can be derived for other units under certain conditions (e.g. heat exchangers with fixed heat duties and most units using ideal physical properties), but otherwise we require more than input-output information from the modules.

Once the Jacobians are calculated, the chainruling strategy follows the calculation sequence specified by the simulator. The number and configuration of tear streams presents no problem since Jacobians are defined as the partial derivative of all output stream elements with respect to all input stream elements. Therefore, the

propagation of the different chains is merely a bookkeeping task given directly by the calculation sequence.

Tear Set Selection

From the structure of the flowsheet and the type of gradient calculation, one can easily derive optimal tear strategies for the optimization problem.

The calculation sequence is largely determined by the objective function, constraints and decision variables as well as by the process flowsheet. This sequence extends from the variables x and y that are encountered furthest upstream to the variables r and w that are furthest downstream. From the flowsheet in Fig. 1 it is easy to see that the variables r_1 and x_3 do not participate in the optimimzation problem because they do not satisfy the above rule. r_1 is not not affected by any decision variables downstream from it and x_3 does not affect any upstream retention variables. Of course, such variables will only be encountered on acyclic portions of the flowsheet.

For direct loop perturbation, the optimal tear sequence could be found rigorously by minimizing the number of perturbations defined by equation (1) subject to the constraint that all loops be torn. To avoid this set covering problem, one could use the heuristic of minimizing the number of tear streams and choosing among these solutions the tear set that minimizes the effort of design variable perturbations.

From equation (4) one sees that the perturbation effort required by chainruling is independent of the tear sequence. The only effect in the ;gradient calculation strategy is the number of chains created and propagated. Also the optimization problem is reduced if fewer tear variables and equations are chosen. Thus we can use the heuristic (similar to the one above):

1) Choose a tear set that minimizes the number of tear variables.

2) Among the solutions choose the tear stream that has design variables toward the end of the loop.

The second step reduces the length of design variable chains even though the expected savings encountered here are generally negligible.

For simple flowsheets, tear streams can usually be chosen by inspection. For more complex ones, a number of methods can be applied (see Gundersen and Hertzberg (1983)) for minimal tear set selection.

The Role of Pressure in the Flowsheet

Most flowsheeting modules have a simple pressure dependence with either a prespecified outlet pressure or pressure drop. Otherwise, pressure is often a weak function of inlet flowrate or enthalpy, and may thus be removed from the tear set under the following conditions:

1) The outlet pressure of unit k is fixed or assigned by a decision variable.

2) Outlet pressure is not affected by inlet flows or enthalpy for all units downstream of unit k.

3) Inlet pressure does not affect outlet flows, enthalpy or retention variables for all units upstream of and including unit k.

The first condition creates a null row for pressure in the Jacobian $\mathbf{I} - \mathbf{I}$ of unit k. If the second and third conditions are satisfied, then through matrix multiplication the pressure tear equation becomes

$$p_{in} + A\,p = p_{out} \quad ,$$

which is independent of the other tear variables and can be deleted without affecting the solution of the QP. If in addition the pressure of unit k is fixed, then the columns corresponding to inlet pressure and the row corresponding to outlet pressure can be deleted from all modular Jacobians. Here pressure need not be considered at all during the chainruling process.

The above analysis cam apply to other tear variables such as trace flowrates and enthalpy as well, but the above conditions occur less often for these variables.

Inlet and outlet pressures are often related to each other by non-

differentiable equations of the form:

$$P_{out} = \min_j \{P_{in}^j\} \qquad j = 1, np$$

One can write this expression in an equivalent form to expose all the nondifferentiabilities.

$$P_{out} = P_1 + \min\{0,\ P_2 - P_1 + \min\{0,\ P_3 - P_2 + \min\{0, \ldots + P_{n-1} - P_{n-2} + \min\{0,\ P_n - P_{n-1}\} \ldots\}\}\}$$

where, for np = 2:

$$P_{out} = P_1 + \min(0, f)$$
$$f = P_2 - P_1$$

Now each nondifferentiable term, min (0,f) can be smoothed by fitting a quadratic function around the nondifferentiable point as shown in Fig. 4a. By enforcing the conditions for each min-term:

$$g(f) = \min(0,f)$$

$$g(-\epsilon) = -\epsilon \qquad dg/df(-\epsilon) = 1$$

$$g(\epsilon) = 0 \qquad dg/df(\epsilon) = 0$$

one obtains the approximation (Duran and Grossmann (1983)):

$$g(f) = \min(0,f) \approx \begin{cases} f, & f < -\epsilon \\ 0, & f > \epsilon \\ -\dfrac{f^2}{4\epsilon} + \dfrac{f}{2} - \dfrac{\epsilon}{4}, & -\epsilon \le f \le \epsilon \end{cases}$$

where ϵ is a prespecified tolerance

This approximation is everywhere differentiable and at the price of some inaccuracy introduced by ϵ, one can avoid convergence to nonoptimal points or premature termination.

As an example of the usefulness of this procedure, consider the very simple process flowsheet given in Fig. 4b. Here we wish to

maximize the vapor flowrate of the flash unit by adjusting the pressure of the inlet streams. Stream 1 and stream 2 ar6 mixed and thus determine the pressure of the flash drum. Now if $p_1 > p_2$ then p_2 has an effect on the flash unit and vice versa. However, when $p_1 = p_2$ forward difference pressure perturbations applied to the flowsheet will indicate that neither pressure has an effect on the flash regardless of the value of p_1. At this point the optimization algorithm will terminate because the incorrect "gradients" satisfy optimality conditions.

If we use the approximate quadratic form, then applying the optimization algorithm and choosing either pressure as a decision variable will lead to the lower pressure bound as the optimum. Here if we start at any pressure with $p_1 = p_2$, the gradients are $\frac{dP}{dP_1} = \frac{dP}{dP_2}$ from the quadratic approximation. Thus, both inlet pressures have an effect on the flash drum. Other nondifferentiable equations can be treated in the above manner once they are identified in the flowsheet. One obvious complication introduced by this strategy is that it always leads to nonlinear equations (even if the differentiable parts are linear) and to the possibility of multiple local optima.

Use of Simplified Models

Since most modules do not have analytic gradients and calculation by perturbation may be extremely time-consuming, it remains an open question if simple, nonlinear models can be found which give more efficient implementations. Several studies have demonstrated the effectiveness of simple models for complex unit operations and flowsheet simulation problems (Boston and Britt (1976), Chimowitz et al (1982) and Pierucci et al (1983)). Jirapongphan et al (1980), even developed an optimization strategy that involved fitting simple model parameters to the more rigorous ones and then applying the

optimization algorithm only to the simplified models.

However, it can easily be shown (see Biegler (1983)) that simple models can be detrimental to the determination of a process optimum. While simulation problems have termination criterion based on function values of the tear equations, optimality criteria are based on the gradients of the objective and constraint functions. Thus, using simplified models that have gradients different from the rigorous models can lead to the satisfaction of the optimality conditions at a non-optimal point. Therefore, while simple nonlinear models can be very useful for simulation, considerable analysis regarding accuracy and sensitivity needs to be done before they can be implemented correctly in an optimization framework.

Example Problems

Three process problems were chosen to test the effectiveness of the chainruling strategy. These examples have been described in previous papers in connection with the infeasible path strategy (Biegler and Hughes (1982),(1983)). All of the process optimization problems were solved using the SPAD simulator (Hughes et al (1981)).

The first two problems deal with the flash loop shown in Figure 5. Both problems have seven tear equations, nine decision variables with upper and lower bounds and six retention variables. The adiabatic flash pressure and the bottoms split fraction represent the two degrees of freedom in the process. Because the loop pressure is fixed by a decision variable, pressure need not be included as a tear equation.

The first problem seeks to maximize the molar flowrate of propane in the overhead flash vapor. A contour plot of the objective function surface in the reduced space of pressure and split fraction (see Biegler and Hughes 1982) shows the function to be monotonic and only slightly nonlinear. The second problem has a nonlinear combination of the flowrates of the overhead flash as its objective function. This function is highly nonlinear and non-

convex. We refer to these examples as the monotonic and ridge problem, respectively. More specific information about these problems is given in Table 1 and in Biegler and Hughes (1982).

The third problem is an optimization of a propylene chlorination process. The flowsheet is presented in Figure 6; seven equality constraints, sixteen decision variables, eight retention variables and three inequality constraints make up the optimization problem. More information about the nature of the variables and constraint functions is given in Table 2. A detailed description of the process model, including reactor design, kinetics and nonstandard modules has been presented in Biegler and Hughes (1983). In the process considered here however, the pressure drop through the reactor has been fixed at 69 kPa (10 psi). The objective of this process is to maximize net annual sales. Prices for the products and raw materials are also given in Table 2. Note that these prices are slightly different from the ones used in Biegler and Hughes (1983) and that the objective function converges to a different optimum.

The propylene chlorination process streams were torn so as to minimize the number of tear variables and the size of the optimization problem. Note that since the pressures of both loops are fixed by decision variables in the recycle compressor and the separating column, no pressure variable is needed in the tear set. However, several units are affected by inlet pressure, so pressure must be included in the Jacobians of the modules. Finally, note that the finishing column need not be evaluated during the calculation sequence of the flowsheet because it contains no retention variables.

The first two problems were run from two different starting points as in the previous study. Tear set initialization procedures were the same as described in the previous papers. The variable scale sets in all three problems were the same as the most successful scale sets in

previous studies; these are listed in Tables 1 and 2. None of the objective or constraint functions were scaled. An absolute Kuhn-Tucker tolerance of 10^{-3} was used for all runs. Relative perturbation sizes for gradient calculation were < c.tr to iO ^^A /£ .. All computer runs were performed en a DEC-2O computer at the Carnegie-Mellon Computing Center.

Results o£ Optimization Study

A comparison of the performance of the chainruling and direct loop perturbation method is given in Table 3. The results indicate that, in all cases but one, chainruling required far fewer module evaluations than direct loop perturbation. Simply by studying the two flowsheets, one finds that for the tear sets chosen, the following relations give the number of modular evaluations:

1) Flash Loop
   Direct Loop Perturbation: MBE = 4 + 381 + *UJ*
   Chainruling: N3E = 4 + 151 + 4(

2) Propylene Chlorination
   Direct Loop Perturbation: NBE » 11 + 1381 + 11^
   Chainruling: NBE = 11 + 681 + 1W

where i = # iterations

I = # additional line search

For the same i and *I* in both algorithms, it is easy to see that chainruling will have fewer module evaluations.

Table 3 also illustrates an important point about the accuracy of gradient calculation. In the first problem, both algorithms taXe identical full steps to the optimum. This occurs from either-starting point with a relative perturbation size of lo^{-3}A^V^ f° *

In the second problem a more interesting effect is observed. Note that if the perturbation size (71) is set to 10^{-2} for direct loop perturbation and 10^{-3} for chainruling, the number of iterations for both algorithms is the aawc from either starting point (their paths however are not identical). With direct loop perturbation, the entire loop is

evaluated with each perturbation of a tear variable. As the calculation procedure executes, each module receives the perturbed outlet stream of the previous module. Thus the magnitude of the relative perturbation size for each module is dependent on the previous modules and is usually smaller than that specified for the tear variable. With chainruling the perturbation size is fixed independently for each module. This explains the similar performance for different perturbation sizes.

The remaining results for the second problem also illustrate the importance of choosing appropriate perturbation sizes. Here direct loop perturbation fails from the ~~second~~ first starting point with $\eta = 10^{-3}$. While the chainruling algorithm always converges to the optimum, it requires, form the second starting point, almost twice as many iterations with $\eta = 10^{-2}$ than with $\eta = 10^{-3}$. We can observe qualitatively the effect of perturbation sizes on gradient error. From a Taylor series expansion on a single function f(x) we have:

$$\frac{f((1+\eta)x) - f(x)}{\eta x} = \frac{df}{dx} + 0(\eta x)$$

Furthermore, since the modular calculations are often subject to convergence tolerances, considerable noise can be present. The effect of noise can be given approximately by:

$$f(x) = \bar{f}(x) + \epsilon$$

where $\bar{f}(x)$ is the noise-free value and the noise, ϵ, is not a function of η. Thus the gradient is approximated by

$$\frac{\bar{f}((1+\eta)x) - \bar{f}(x) + (\epsilon_1 - \epsilon_2)}{\eta x} = \frac{df}{dx} + 0(\eta x) + 0\left(\frac{1}{\eta x}\right)$$

To reduce the error introduced by calculation noise, Crowe (1978) executes a fixed number of iterations during the perturbation step without enforcing a convergence tolerance. This procedure, however, would be hard to implement on existng software. Hutchison et al. (1983) choose a perturbation size

that minimizes the error in the above equation. This can only be done, however, at the expense of additional perturbations. In $SPAD_f$ we simply impose a tight convergence tolerance on modular calculations, thus making $(CJ-SJ)$ small. This usually helps to minimize the perturbation error.

CONCLUSIONS AND SIGNIFICANCE

This paper is the first half of a study that presents improvements in the infeasible path strategy. Here we address problems encountered when obtaining gradient and function information from the process simulator for the optimization algorithm and some possible remedies. In this context we present the chainruling algorithm and discuss its advantages over direct loop perturbation.

Chainruling allows analytic Jacobian information, to be specified when available, as well as incorporation of more flexible and accurate gradient calculation procedures. The most significant advantage of chainruling is that it usually leads to large reductions in the number of module evaluations in the perturbation step. A tear set strategy was outlined for both the chainruling and direct loop perturbation algorithms that minimizes the number of modular evaluations. Also, the role of pressure in the tear streams and perturbation procedure was analyzed aloug with a strategy for handling nondifferentiabilities in loop pressure and other functions.

Finally, we tested the above improvements on three process optimization problems. The results indicate that the chainruling algorithm can save up to 60% of the module evaluations required by direct loop perturbation. From this limited study it appears that chainruling is affected less by gradient error and thus less susceptible to failure. Although chainruling requires some overhead in processing and constructing Jacobian matrices, its advantages are readily apparent for large, complex process simulations where most of the computational effort is consumed by modular calculations.

References

Biegler, L.T., "Simultaneous Modular Simulation and Optimization", 2nd International Conference on Foundatons of Computer Aided Process Design, Snowmass, CO, (1983)

Biegler, L.T. and J.E. Cuthrell, "Improved Infeasible Path Optimization for Sequential Modular Simulators, Part 2 : The Optimization Algorithm", submitted to AIChE J., (1983)

Biegler, L.T. and R.R. Hughes, "Infeasible Path Optimization of Sequential Modular Simulators", AIChE J., 28, 6, p.994 (1982)

Biegler, L.T. and R. R. Hughes, "Process Optimization z A Case Study Comparison", Comp. and Chem. Engr., to appear (1983)

Boston, J.F and H.I. Britt, "A Radical Formulaton and Solution of the Single Stage Flash Problem", Comp. and Chem. Engr., 2, p.109 (1978)

Chamberlain, R.M., C. Lemarechal, H.C. Pedersen, and M.J.D. Powell, "The Watchdog Technique for Forcing Convergence in Algorithms for Constrained Optimization", DAMTP 80/NA1, University of Cambridge, (1979)

Chen, H-S and M.A. Stadtherr, "Strategies for Simultaneous Modular Flowsheeting and Optimization", 2nd International Conference on Foundations of Computer Aided Process Design, Snowmass, CO, (1983)

Chimowitz, E.H., S. Macchietto, T.F. Anderson and L.F. Stretzman, "Local Models for Representing Phase Equilibria", I & EC Proc. Der. Dev., to appear (1983)

Crowe, C. M., "Convergence Promotion of the Steady-State Simulation of Chemical Processes Using the Dominant Eigenvalue Method", 2nd International Congress on Computers and Chemical Engineering, Paris C-15, (1978)

Dennis, J.E. and J.J. Moore, "Quasi-Newton Methods, Motivation and Theory", SIAM Review, 19, 1, pp.46-89 (1977)

Duran, M. and I.E. Grossmann, personal communication, (1983)

Gunderson, T. and T. Hertzberg, "Partitioning and Tearing Chemical Process Flowsheets", Process Systems Engineering Symposium, Tokyo, (1982)

Han, S-P, " A Globally Convergent Method for Nonlinear Programming", J. Optimization Theory and Applications, 22, 3, p.297 (1977)

Hughes, R.R., R.K. Malik and L.T. Biegler, "SPAD-Simiiator for Process Analysis and Design", EES Report #52, University of Wisconsin, (1981)

Hutchison, H. P., S. Kaijaluoto, and W. Morton, "Process Optimization Using a Serial Cyclic Flowsheet Simulator", 3rd International Congress on "Computers and Chemical Engineering", Paris, (1983)

Jirapongphan, S., J.F. Boston, H.I. Britt, and L.B.Evans, "A Nonlinear Simultaneous Modular Algorithm for Process Flowsheet Optimization", presented at 80th AIChE Meeting, Chicago, (1980)

Kurtagh, B.A. and M. A. Saunders, Math. Prog., 14, p.41 (1978)

Perkins, J.D., "Equation Oriented Flowsheeting", 2nd IBternational Conference on Foundations of Computer Aided Process Design, Snovraass, CO, (1983)

Pierucci, S«. J., E.M. Ranzi and G.E. Biardi, "Solution of Recycle Problems in a Sequential Modular Approach", AIChE J. 28, 5, p.820 (1982)

Powell, M.J.D., "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations", presented at the 1977 Dundee Conference on Numerical Analysis (1977)

Schittfcowski, K. "The Nonlinear Programming Algorithm of Wilson, Han and Powell with an Augmented Lagrangian Type Line Search Function, Part 2: An Efficient Implementation with Linear Least Squares Subproblems", Numer. Hath., 38, p.115 (1981)

Stadtherr, M.A. and H.S. Chen, "Numerical Techniques for Process Optimization by Successive Quadratic Programming", 3rd International Congress on "Computers and Chemical Engineering", Paris, (1983)-~

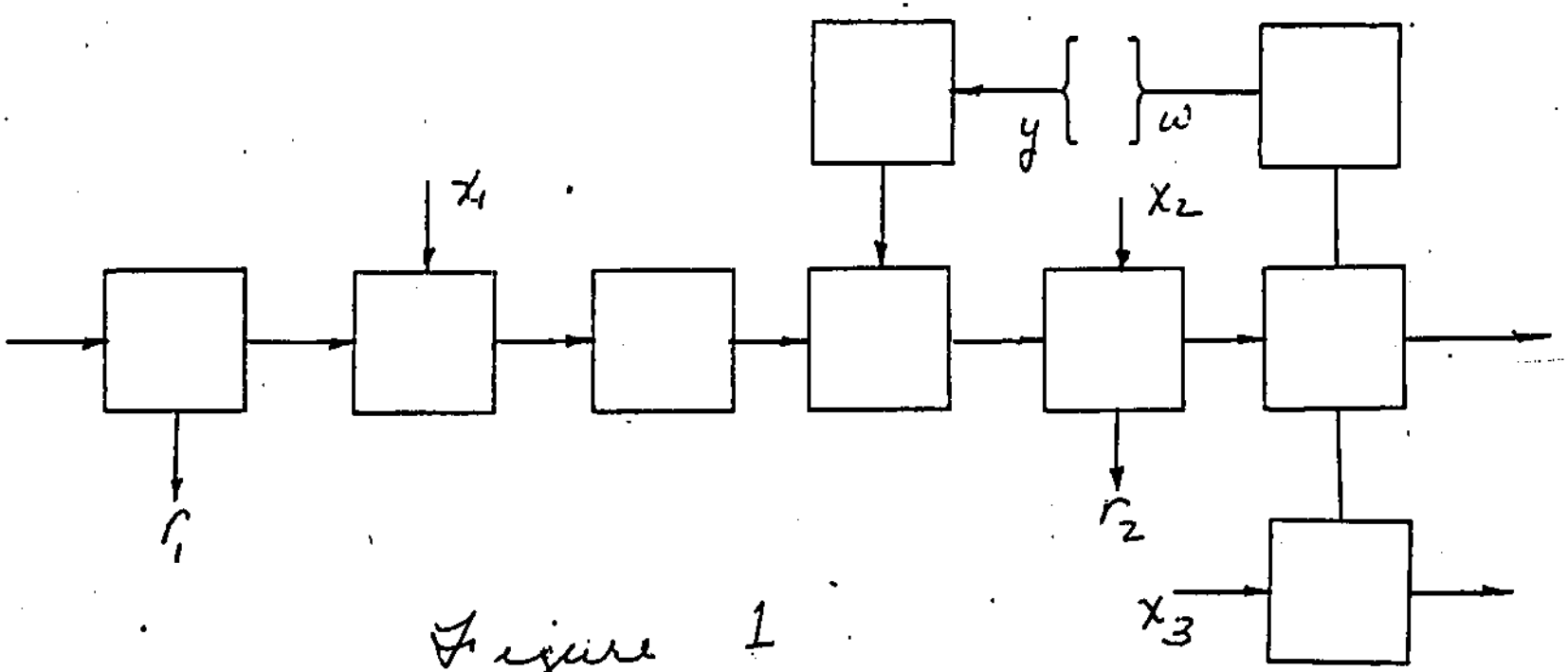


Figure 1

Simplified Process Sheet

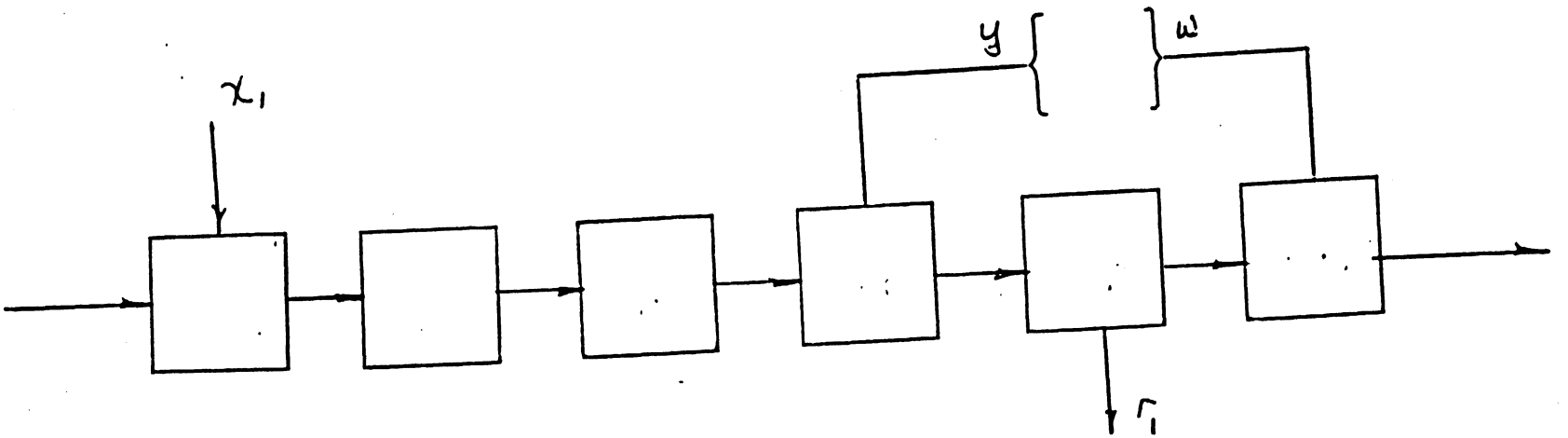


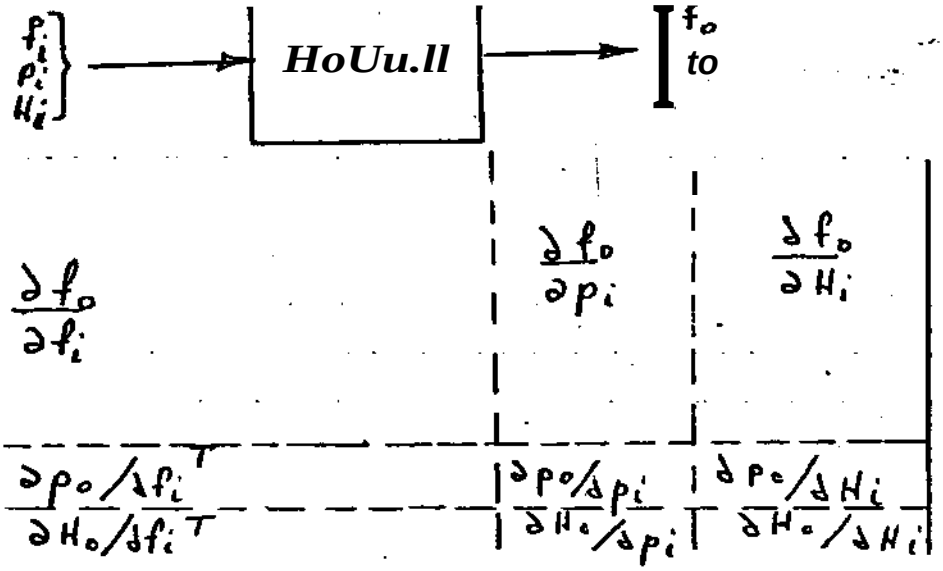
Figure 2

Comparison of Gradient Calculation Methods

Direct Loop Perturbation : $NBE = 6 + 3N$

Chainruling : $NBE = 1 + 5N$

$y \in R^N$, $N > 2$



f - flowrate
p " iiA*c*-f-t*4*e-
H - specifi enthalpy

$$\frac{\partial \rho_o}{\partial f_i} = \frac{\partial f_o}{\partial \rho_i} = \frac{\partial f_o}{\partial H_i} = \underline{0}$$

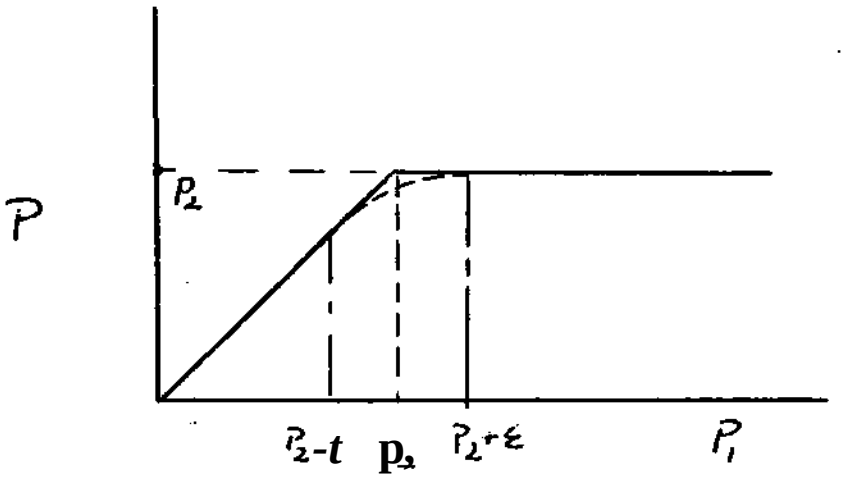
Also, $\frac{\partial \rho_o}{\partial H_i} = 0$, $\frac{\partial H_o}{\partial H_i} = 1$ for all six jacobians

	$\partial f_o/\partial f_i$	$\partial \rho_o/\partial \rho_i$	$\frac{\partial H_o}{\partial f_i}$	$\partial H_o/\partial \rho_i$
1. mixer.	$\underline{\underline{I}}$	0 or 1 see fig. 4	$\frac{H_i - H_o}{2 f_o}$	0
2. pump	$\underline{\underline{I}}$	0 or 1 + by perturbation	0 * (~~$\frac{\partial \rho_o}{\partial \rho_i}$~~)	by perturbation $-\rho/\eta$
3. combiner	$\underline{\underline{r}}$	0 or 1 + by perturbation 0	0 * (~~$\frac{\partial \rho_o}{\partial \rho_i}$~~)	by perturbation
4. valve	$\underline{\underline{I}}$	0	0	0
5. splitter	$\underline{\underline{3}}\underline{\underline{I}}$	1	0	0
6. component splitter	$\mathrm{diag}\{\underline{\underline{3}}_1 \ldots \underline{\underline{3}}_n\}$	0 or 1 by perturbation	0	0

* not identically zero, but negligible
+ depends on whether $\Delta\rho$ or ρ_o is fixed.

Figure 3

r



$$\hat{\pi} = \min\{P_1, P_2\}$$

$$\approx P_1 + \begin{cases} f, & f < -\varepsilon \\ 0, & f > \varepsilon \\ \text{£-\%£} - i, & |f| \le \varepsilon \end{cases}$$

$$-f- P_2 - P_1$$

Figure 4a: Quadratic Approximation to Nondifferentiable Pressure

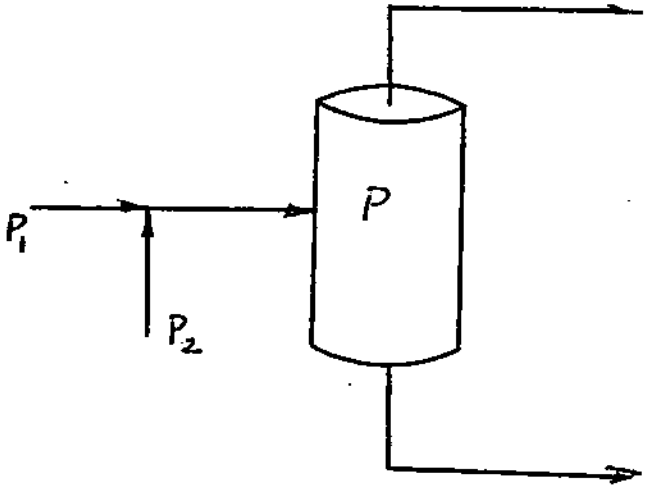


Fig. 4b

Flash with two inlet streams

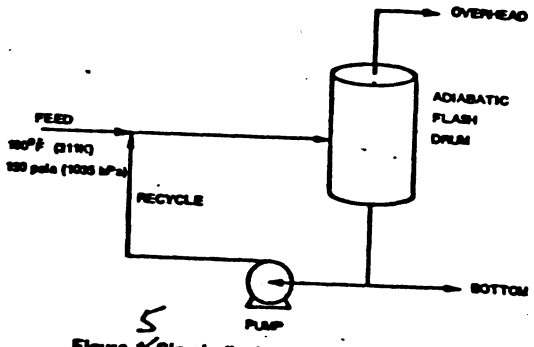


Figure 5. Simple flash process flowsheet.

FEED **(kg-mol/h)**

Propane	10
1-Butene	15
N-Butane	20
Trans-2-Butene	20
Cis-2-Butene	20
Pentane	10

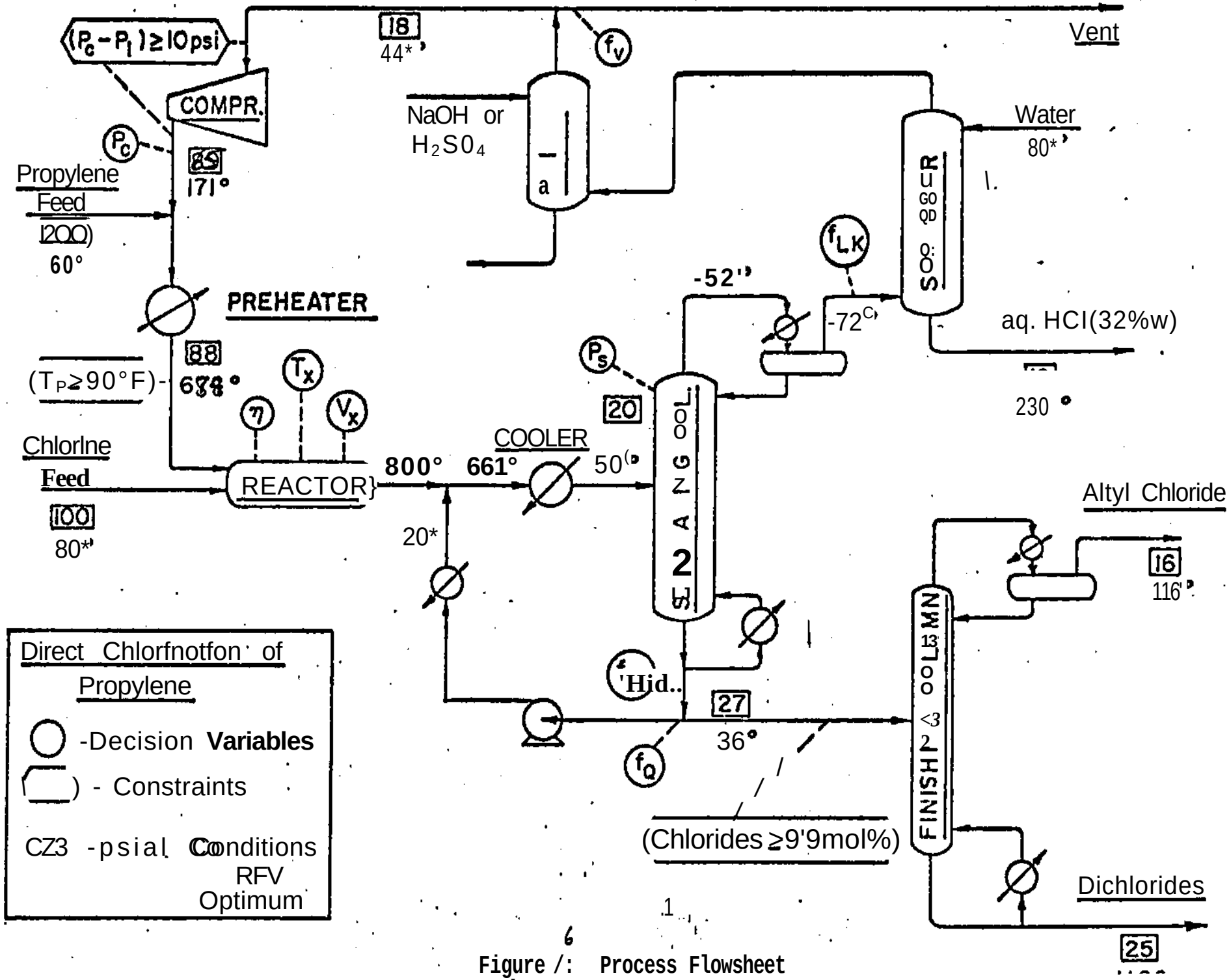


Figure /: Process Flowsheet

TABLE L FLASH PROBLEM SPECIFICATIONS

Objective Function		Constraints and Bounds
MonoUmie:		0.2 < xi < 0.8
Max	tfi)	69 ≤ x_s £345
x,f		05 9,^100 (i = 1,6)
Ridge:		
Max	$-\frac{1}{2}+h_t-\sqrt{p_5}$]	$y_i - w_i = h_i = 0$ (j = 1,7)

fadependent Variables

Si—Split fraction—Stream ©/Stream ®

xt—Adiabatic flash pressure, kPa

f,(i * 1,6)—Component flows—Stream (2) (kg-mol/h)

Specific enthalpy—Stream ©(w/kg-mol)

Retentate + [illegible]

ft (I «= 1,6>—Component flows—Stream © (kg-mol/h)

m>t (« * 1,6)—Component flows—Stream ® (kg-mol/h)

IDT —Specific enthalpy—Stream ® (w/kg-mol) "~"

Initialization (Standard Procedure)

x—Tabulated starting point values

f—Direct substitution (from ID), *after* a single pass through the simulation with x = x and y = 0.

Scaling—Runs were made with one of the following sets of scaling factors for the independent variables:

^	*i	*t	91	91	9s	94	9s	*U*	*n*
5 S	0	jZ	2	6	3	3	3	3.4	

for problems 1 and 2

J-'"£/">'*

$z = [x, y]$

$\bar{z}$ = scaled variables used by nonlinear problem

Problem Optima

/• /' 7.4.33 ., -£, ± 0.2 ""

x_2

Table 2
Propylene Chlorination
Problem Specifications

Objective function

$$QV.II \quad n_6 + 12.48\, r_7 + 4.91\, r_8$$

s.t.
$$(r_6 + r_7 + r_8)/(r_4 + r_5 + r_6 + r_7 + r_8) \leq A\text{-if}$$
$$r_3 \geq 90$$

		S
x_1 – C_3H_6/Cl_2 ratio to reactor		I
x_2 – reactor temperature (°F)		3
x_3 – reactor volume (ft^3)		I
x_4 – upper. column pressure (psia)		2
x_5 – split frac. to quench		
x_6 – split frac. to vent		-2
x_7 – [illegible] compressor pressure		2
x_8 – recovery Cl_2 in sep. tops		3
x_9 – recovery Allyl chloride in sep. bottoms		-1
y_1 – HCl (lb-mole/hr)	tear stream	
y_2 – Cl_2 "		
y_3 – C_3H_6 "		
y_4 – Allyl chloride "		
y_5 – Dichloropropane "		
y_6 – Dichloropropene "		
y_7 – Enthalpy (Btu/lb-mole)		3

r_1 – ~~inlet~~ reactor inlet temperature
r_2 – outlet [illegible] pressure
r_3 – inlet compressor pressure
r_4 – absorbed Cl_2 in sep. bottoms (lb-mole/hr)
r_5 – C_3H_6 in sep. bottoms "
r_6 – Allyl Chloride in sep. bottoms "
r_7 – Dichloropropane in sep. bottoms "
r_8 – Dichloropropene in sep. bottoms "

$z = [x, y]$, $\bar{z} = z/10^s$.
z – scaled variables used by nonlinear progr~~optimization~~

Table 3
Comparison of Gradient Calculation Strategies

CPU sec (#iterations) ⟨NBE⟩

	Direct Loop Perturbation		... Chainruling	
	$\eta = 10^{-2}$	$\eta = 10^{-3}$	$\eta = 10^{-2}$	$\eta = 10^{-3}$
1a	7.721 (5) ⟨194⟩	7.579 (5) ⟨194⟩	6.250 (5) ⟨79⟩	5.978 (5) ⟨79⟩
1b	7.914 (5) ⟨194⟩	7.693 (5) ⟨194⟩	6.076 (5) ⟨79⟩	6.035 (5) ⟨79⟩
2a	15.283 (10) ⟨388⟩	32.572 * (22) ⟨880⟩	13.364 (11) ⟨185⟩	12.705 (11) ⟨189⟩
2b	10.243 (7) ⟨274⟩	17.766 (12) ⟨460⟩	15.307 (13) ⟨203⟩	8.290 (7) ⟨113⟩
3	280.954 (39) ⟨5723⟩	228.533 (32) ⟨4614⟩	207.356 † (27) ⟨1890⟩	139.463 (20) ⟨1459⟩

Starting pts for 1 and 2: a – $[0.5, 40] = x_0^T$; b – $[0.4, 30] = x_0^T$

* terminated at 4.9965, far from optimum

† line search failure at 1482.99.

Table %\. Propylene Chlorination -- Summary of Optimization Results

saslci: 100 lb mol/hr iropylene feed

		Specifications			Direct Loop Perturbation	Chain ruling
		Lover Bound	Upper Bound	Start		
)b1ective - Net value added, $/hr		———	———	1010.13	1483.06	1483.08
Variables						
1. $C_3H_6/C^{\wedge}_2$ ratio to reactor	n	2	10	10**	4.854	4.934
2. reactor temperature (°F)	T_x	800	1100	1000	ft>u	f oo
3. reactor volume (ft^3)	V_X	1	15	10	/r	AT
4. sepn. column pressure (psia)	P_s	20	50	28	20	2.0
5. split, frac to quench	f_q	0.25	0.75	0.6	0.7-1	0.728
6. split frac. to vent	f_V	0.01	0.10	0.05	0.01	*.oi
7. compr. out. pres. (psia)	P_c	20	100	100**	iZH.	S_{s4sr}
8. fr. recovery $C\&_2$ in sep. tops	f_{LK}	0.95	0.99	0.99**	0.99	•0-.11
9. fr. recovery AC in sep. botts.	f_{HK}	0.95	0.99	0.99**	0.99	0.97
Constraints						
1. $(P_c-P_{in})_{comp}$ (psi)		10	none	74	SO.r	54.8
2. product purity (mol X)		99.0^	none	99.0-	• 77.C	≥ 99.0
3. reactor preheat (*F)	T_p	90	none	907.8	68PJ	6 <?<?./