FEASIBLE PATH OPTIMIZATION WITH SEQUENTIAL MODULAR SIMULATORS

by

L.T. Biegler and R.R. Hughes

DRC-06-62-84

December, 1984

FEASIBLE PATH OPTIMIZATION WITH SEQUENTIAL MODULAR SIMULATORS

L. T. Biegler* and R. R. Hughes
Department of Chemical Engineering
University of Wisconsin- Madison
Madison, WI 53706

*currently at Chemical Engineering Dept., Carnegie-Mellon University, Pittsburgh, PA 15213

FEASIBLE PATH OPTIMIZATION WITH SEQUENTIAL MODULAR SIMULATORS

L. T. Biegler and R. R. Hughes

ABSTRACT

Optimization of large process simulations has typically been inefficient because many time-consuming flowsheet evaluations are usually required. Here we present three optimization methods based on the efficient successive quadratic programming (SQP) algorithm of nonlinear programming. Quadratic/ Linear Approximation Programming (Q/LAP) and the Reduced and Complete Feasible Variant algorithms (RFV and CFV) perform efficiently and interface easily with most current sequential modular simulators. Two simple flash problem studies illustrate the mechanics of these algorithms and demonstrate their rapid performance. These results also help in formulating heuristics for the specification of algorithmic tuning parameters.

SCOPE

Virtually all steady-state chemical process simulators now in actual productive use are of the sequential modular type. The flowsheet is divided into unit modules which are connected by streams; units are modelled by pre-programmed subroutines in a large computer program where input describes the process flowsheet and problem. Calculation proceeds from module to module with trial-and-error iteration of recycle loops, using robust but not always efficient convergence algorithms.

Because the calculation sequence is largely determined by the flowsheet topology, little flexibility remains for design and optimization studies; these are often performed by repeated case studies simulating the process. Alternately, in order to optimize a process automatically, some investigators have treated the entire simulation as a "black box" subroutine and applied

direct search algorithms which require no derivative information. [1,2,3,4] Optimal designs have been obtained, but the computation has usually been lengthy. This is due both to the relative inefficiency of direct search methods, and to the need for convergence of the model for each trial point evaluated.

Hughes [5] and Parker and Hughes [6,7] have shown that faster optimization techniques can be developed if approximations to derivative information can be calculated. Their method, Quadratic Approximation Programming (QAP), required about *6*± simulation time equivalents to optimize an eleven-variable ammonia synthesis process.

In a recent paper [8] we outlined the development of sequential modular optimization and presented Quadratic/Linear Approximation Programming (Q/LAP), based on Successive Quadratic Programming (SQP)[9]. As an application we optimized the ammonia process of [6] in less than twelve simulation time equivalents.

More recently we developed an infeasible path optimization method (1POSEQ) [10] which converges and optimizes flowsheeting problems simultaneously. Though radically different from Q/LAP, IPOSEQ also uses SQP methods as its nucleus•

In this paper we restate the Q/LAP algorithm in detail and study the algorithm's tuning parameters. We also Introduce two new algorithms that are feasible variants of IPOSEQ. Like Q/LAP the Complete Feasible Variant (CFV) and Reduced Feasible Variant (RFV) algorithms obtain a converged simulation for each function evaluation (i.e., each step).

CONCLUSIONS AND SIGNIFICANCE

Three feasible path optimization algorithms have been shown to be effective and simple to implement and use in a sequential modular framework: Q/LAP

[8], and two new algorithms (CFV and RFV) that combine the best features of Q/LAP and IPOSEQ.

From analysis of these algorithms and repeated optimizations of two simple modular problems, we postulate the following heuristics about the algorithmic tuning parameters:

1. Scale the design variables so that the initial objective function gradient elements have absolute values between ten and one hundred.
2. Set perturbation sizes large enough to make stream variable responses in gradient calculation larger than the recycle convergence errors in the function evaluation.
3. Choose an optimization closure tolerance, ϵ, at least as large as the objective function responses in gradient calculation. [For CFV, increase the minimum tolerance to allow for the tear-equation error term, $|vh|$, in equation (11), below.]

The feasible variant algorithms can be programmed as self-contained subroutines that easily substitute for the recycle convergence algorithm. The calculation sequence is the same for the simulation and gradient evaluation steps and no changes are needed in the simulator executive.

Finally, based on the limited results of this study, it appears that RFV is probably the most efficient algorithm for sequential modular optimization, especially if gradients are calculated by numerical perturbation.

I. Method Development

Simulation problems typically involve many thousands of variables. In sequential modular simulation most of these are internal to self-contained unit modules. To keep the number of variables in the optimization problem small, we need only to specify the variables directly involved in evaluating the objective function and constraints. Thus, in the sequential modular environment, the optimization problem may be written as:

$$\underset{x}{\text{Min}} \; \Phi(x,r) \qquad (D$$

$$\text{s.t.} \quad g(x,r) \leq 0 \qquad (2)$$

$$c(x,r) = 0 \qquad (3)$$

$$h(x,y) = 0 \qquad (4)$$

$$r = r(x,y) \qquad (5)$$

where:

Φ - objective function

g - m inequality constraints

c - m_{eq} design (equality) constraints

h - I stream interconnection constraints

y - I vector of stream variables

x - n vector of design variables

r - dependent variables (only those that directly calculate Φ, g and c. Note r is a function of x and y)

The design variables, x, and the functions Φ, g and c are specified by the user as part of the optimization problem. Dependent or retention variables, r, are then added for direct calculation of these functions. In order to get this problem to a form small enough for successive quadratic programming, we make use of equations (4) and (5) to supply derivative information for Φ, g and c.

In Q/LAP, we first obtain a converged simulation and then decompose the flowsheet into component modules as shown in Figure 1. By perturbation of the inputs, x^k and y^k, we construct a linear model for each module, k, of the form:

$$\begin{Bmatrix} \Delta w^k \\ \Delta r^k \end{Bmatrix} = D^k A x^k + \sum_p A_p^k y_p^k$$

Using flowsheet interconnection relationships we then combine these module models to form a large, sparse linear system that approximates equations (4) and (5). We can now perturb the design variables of the linear system, in lieu of the

actual flowsheet, and obtain responses for the dependent variables, r. Here the Jt-vector y contains all stream variables in the decomposed flowsheet and typically has a dimension of several hundred or more. This decomposition strategy yields "reduced" gradients of Φ, g and c with respect to x but requires a sparse matrix solver and a special interface that alters the calculation sequence during the approximation calculation.

The feasible variant algorithms use the gradient calculation strategy of IPOSEQ which requires no special interface, just an optimization module that substitutes for the recycle convergence algorithm. As in Q/LAP, these algorithms converge the flowsheet for each function evaluation. However, they use a calculation sequence like that shown in Figure 2, i.e., one that includes all design and dependent variables in the convergence loop. The perturbation steps use the same calculation sequence as the flowsheet pass because only the tear and design variables are perturbed. Thus, the ^-vector y is much smaller than for Q/LAP. (typically, there are ten to one hundred tear variables) Equation (4) is modelled by linear approximations to the tear equations. Equation (5) remains embedded in the modules and is solved directly in the perturbation and simulation steps.

Once gradient information is obtained, we can set up the quadratic programming problem. For Q/LAP and RFV, we obtain gradients with respect to x only and write the quadratic program as:

$$\underset{d}{\text{Min}} \; [V_x \Phi^T d + 1/2 \; d^T B d] \qquad (6)$$

$$\text{s.t.} \quad g(x) + V_x g(x)^T i \leq 0$$

$$c(x) + V_x c(x)^T d \leq 0$$

Here the linear coefficients from the approximation to equation (4) are used to convert the finite difference derivatives into the desired gradients in x-space, at a given value of x. For Q/LAP, the coefficients from equation (5) are also needed, to eliminate the retention variables r.

For CFV we include the variable increments for both x and y in the quadratic program and add the recycle tear equations (4) as equality con* straints. Because *I* is small the quadratic program is written as:

$$\underset{d'}{\text{Min}}[V\langle f\rangle V + 1/2\ d'^{T}Bd''] \qquad (7)$$

$$\text{s.t.}\quad g(x,y) + Vg(x,y)V < 0$$

$$c(x,y) + Vc(x,y)V = 0$$

$$h(x,y) + Vh(x,y)V - 0$$

In both cases, the B matrix is approximated by quasi-Newton updating. The increment vector d in equation (6) is an n-vector search direction; d^ in equation (7) is of dimension n + £.

In the next section the three algorithms are formally stated. Then, there is a brief description of algorithmic tuning parameters. Finally, we show algorithmic performance and the effect of these tuning parameters by way of two simple flash optimizations.

II Algorithm

The statement of the Q/LAP, CFV and RFV algorithms follows below. Because many steps are shared among the algorithms, a single statement is given, with differences highlighted where they occur.

Set-up

Step JL. Define the process problem and outline the block diagram to simulate the process model.

Step 2^. Choose an objective function <I>, n decision variables x, m inequality constraint functions g, m_{eq} equality constraint functions c, and retention (dependent) variables r. The vector r should include only those dependent variables needed, with the decision variables, to calculate the objective and constraint functions.

Step 2- For Q/LAP:

a) Choose a calculation sequence and tear streams which will result in the fastest convergence of the nonlinear full function evaluation (FFE) for a given x vector.

b) Decompose the block diagram into N_m modules (Figure 1). Parker and Hughes [6] suggest that for the best decomposition, each module either should have one input stream and no decision variables or should consist of a single unit operations block with its associated cost block.

For CFV and RFV:

a) Choose a calculation sequence that contains all decision and retention variables, x and r, and all recycle loops within a single outer calculation loop (Figure 2). Frequently, the optimization problem is posed in a way that permits use of the natural calculation sequence for the simulation problem.

b) Tear the calculation and internal recycle loops, and choose *I* tear variables, y. As with IPOSEQ [10], these are usually the component flows, pressure, and specific enthalpy for each tear stream. If the pressure is fixed, or used as a decision variable anywhere in a loop, the pressure of the corresponding tear stream may be omitted.

Step 4. Set 1 « 1, x^i « x^1, initialize the tear variables, and converge the flowsheet simulation to yield (at $x^i \cdot x^1$), a full function evaluation, defined as:

FFE: $\bullet(x^1{}_f r^i)_{f!} g(x^1{}_f r^1),\ c(x^{\pm}, r^i)$

where $r^{i\,s}\ r(x^i, y^i)$

and y^1 satisfies $h(x^i, y^i) = y^i - w(x^i, y^i)\ \%\ 0$

[For CFV and RFV, $w(x^i, y^i)$ are the calculated tear stream values, given assumed values y^i ; for Q/LAP, w^i are the module output stream values, and y^i the module input stream values. After iterative convergence of the nonlinear

simulation all elements of h are either zero (for non-tear streams with Q/LAP) or smaller than the specified convergence tolerance(s) (for tear streams with any of the algorithms)]

Iteration

[Except where otherwise specified, x, y, and other variables are assumed to have values corresponding to the current iteration, i.]

Step Ju For Q/LAP:

a) By forward-difference perturbation of input streams and design variables, construct the following linear models for each module k:

$$Aw^k \approx D^k Ax^k + \sum_{p=1}^{N_{in}} A_p^k \, Ay_p^k \qquad (8)$$

$$Ar^k \approx D'^k \, Ax^k + \sum_{p=1}^{N_{in}} A'^k_p \, Ay^k_p \qquad (9)$$

where w - vector of output streams for module k

r - retention variables of module k

y_p - input stream p for module k

N_{in} - number of input streams for module k

x - vector of design variables for module k

Each stream vector is made up of all the component flows, stream pressure and the specific enthalpy. Perturbing the enthalpy yields smoother responses in retention variables. If temperature were perturbed, discontinuous changes in enthalpy would occur at phase changes.

b) Combine equation (8) with the input-output stream interconnections: $w_q^o = y_p^k$ if output stream q of module o is used as input stream p of module k. The result is a large sparse system of linear equations:

$$RAy = x \qquad (10)$$

Here Ay - vector of combined output streams, Ay_p for all p and k.

R - sparse coefficient matrix of the form $I - A_p$. A_p is a sparse matrix consisting of blocks of the A_p^k coefficient matrices from equation (8), ordered according to the stream sequence.

X - n columns, each corresponding to perturbations of one design variable. Depending on stream order, column j of x ^as elements of either $D^k Ax_j$ or zero.

Process feed streams are fixed and cannot be incremented, except when feed stream elements are used as design variables. Thus feed stream increments, if they occur, affect only Xt the right-hand side of equation (10). Process output stream increments are ignored unless they are defined as part of the retention vector.

c) Using each column of x in turn, solve for (Ay)^ for each numerical perturbation Ax_j. For the specified Ax_j and the calculated $(Ay)^j$% select proper contributions for equation (9), and find:

$$(Ar)^j \; E \underset{k=l}{\overset{N}{I}} \; (ArV = \underset{k=*l}{\overset{N}{\pounds}} \left[D^{-k} \Delta x_j^k + \sum_{p=1}^{N_{in}} A^{-k}_p (\Delta y_p^k)^j \right]$$

Evaluate the objective function gradient elements as:

$$\left| \frac{\partial \phi}{\partial x_j} \right| = \left| \frac{\Delta \phi}{\Delta x_j} \right| = \left| f_i(x + \Delta x_j, r + (\Delta r)^j) - f_i(x, r) \right| / \Delta x_j$$

Evaluate the constraint function gradients from similar equations in g(x,r) and c(x,r).

For CFV and RFV:

a) Starting from the tear stream, move backward along the calculation loop until an unperturbed design variable x_j is encountered. Perturb this design variable and calculate responses of the dependent and tear variables

in that module and all downstream modules of the calculation sequence. Dependent variables of upstream modules remain unaffected. Evaluate the gradients of ϕ, g, c, and h from equations of the form:

$$\frac{\partial \phi}{\partial x_j} \approx \left[\phi(x+\Delta x_j, r^*) - \phi(x,r)\right] / \Delta x_j$$

where $\Delta x_j \equiv \xi x_j$

ξ = a fractional perturbation factor

r^* = r values from the perturbed calculation

Repeat this step (5(a)) for each element of x.

b) One at a time, perturb the tear variables y^i, evaluate dependent variables and tear equations and calculate gradients of ϕ, g, c, and h from equations of the form:

$$\partial\phi/\partial y_j = [\phi(x,r^*) - \phi(x,r)]/\Delta y_j^i$$

For the RFV algorithm calculate the reduced gradients of ϕ, g, and c as follows:

$$\nabla_x \phi \equiv \frac{d\phi}{dx} = \left[\frac{\partial \phi}{\partial x}\right] - \left[\frac{\partial h}{\partial x}\right]\left[\frac{\partial h}{\partial y}\right]^{-1} \left[\frac{\partial \phi}{\partial y}\right]$$

Since the desired path maintains $h(x,y) \approx 0$, the reduced gradient of h is also zero.

Step 6. Update the Hessian matrix, B^i, using the BFGS [11] equation: (if $i=1$, $B^i=B^o$)

NOTE: Default value of B^o is I, the identity matrix. Other B^o matrices may be specified if desired, but they should be positive definite. One useful form is a diagonal matrix with weighted diagonal values.

If 1>1, apply the BFGS [11] equation:

$$B^{1} = B^{1-1} + \frac{zz^{T}}{z^{T}s} - \frac{B^{i-1}ss^{T}B^{i-1}}{s^{T}B^{i-1}s}$$

$$z = \theta\gamma + (1-\theta)B^{i-1}s$$

$$\theta = \begin{cases} 1 & \text{if } s^{T}\gamma > 0.2\, s^{T}B^{i-1}s \\ \dfrac{0.8\, s^{T}B^{i-1}s}{s^{T}B^{i-1}s - s^{T}\gamma} & \text{if } s^{T}\gamma < 0.2\, s^{T}B^{i-1}s \end{cases}$$

For Q/LAP and RFV: B^{i} has dimensions n x n, s and Y are n-vectors:

$$s = x^{i} - x^{i-1}$$

$$\gamma = \nabla_{x}L(x^{i}, u^{i-1}, t^{i-1}) - \nabla_{x}L(x^{i-1}, u^{i-1}, t^{i-1})$$

$$L(x,u,t) = \phi(x) + ug(x) + tc(x)$$

For CFV: B^{i} has dimensions (n+Jl) x (n+Jl), s and Y are (n+Jl)-vectors:

$$s = \begin{bmatrix} x^{i} \\ y^{i} \end{bmatrix} - \begin{bmatrix} x^{i-1} \\ y^{i-1} \end{bmatrix}$$

$$\gamma = \nabla L'(x^{i}, y^{i}, u^{i-1}, t^{1-1}, v^{i-1}) - \nabla L'(x^{i-1}, y^{i-1}, u^{i-1}, t^{i-1}, v^{i-1})$$

$$\nabla = \begin{bmatrix} \partial \\ \partial x \\ \\ \partial y \end{bmatrix}$$

$$L(x,y,u,t,v) = \phi(x,y) + ug(x,y) + tc(x,y) + vh(x,y)$$

In the above equations, u,t, and v are the QP multipliers from Step 7 of the previous iteration.

Step 1 ' Solve the quadratic program, using the gradient values from Step 5 and the matrix value from Step 6:

~~For Q/LAP + RFV~~;

$$\min_{d} \; (\nabla_{x}\phi)^{T}d + 1/2\; d^{T}\nabla^{2}_{xx}L\, d$$

$$\text{s.t.} \quad (\nabla g)^{T}d + g \leq 0$$

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

For RFV only;

$$\text{Evaluate} \quad u) = \left[\frac{\partial h^T}{\partial y}\right]^{-1} \left[\frac{\partial h^T}{\partial \alpha}\right] d$$

For CFV:

$$\begin{aligned} \underset{d'}{\text{Min}} \quad & (V<j))^T d' + 1/2\, d' V d' \\ \text{s.t.} \quad & (Vg)^T d' + g \le 0 \\ & (Vc)^T d'' + c - 0 \\ & (Vh)^T d' + h = 0 \end{aligned}$$

Step 8. Closure test: (Here the closure tolerance £> 0 has the same units as the objective function, <D. The vectors d, d% u, t and v come from the solution of the quadratic program in step 7.)

For Q/LAP and RFV:

$$\text{If} \quad ICV\hat{}x^1))^{1*}! + lugCx^1)! + ItcOc^1)! \le e \text{ , stop.} \qquad (10)$$

(x^i) is within tolerance of a Kuhn-Tucker point.

For CFV:

$$\text{If} \quad |\nabla\phi(x^i, y^j)d' + |ug^i(x^1, y|) + Itc\hat{}.y^1)! + |vh(x^{ii}_{y,})y^i)| < e, \text{ stop.} \qquad (11)$$

(x^i, y^i) is within tolerance of a Kuhn-Tucker point.

Step 9. Find a stepsizeA along the search direction, (d or d') that decreases P(x), an exact penalty function, or, if the exact penalty function has decreased monotonically in previous iterations, decreases a modified Lagrangian. (This is a modification of the "Watchdog" technique. For a more detailed description see Biegler [12])

For Q/LAP and RFV:

$$P(x^i + \lambda d) - P(x^i) \le 0.1A \quad [\nabla_x \phi(x^i)^T d + \phi(x^i) - P(x^i)]$$

$$P(x^i) = \phi(x^i) + \sum_{j=1}^{m} \mu_j \, [\max(0, g_j)] + \sum_{j=i}^{m_{eq}} T_j c_j$$

for the modified Lagrangian:

$$\mu_j = u_j^i$$

$$\mathbf{V} \quad t_j^i$$

for the exact penalty

$$\mu_j = 2|u_j^i|$$

$$T_j = 2|t_j^i| \operatorname{sgn}(c_j)$$

For CFV:

$$P[(x^i, y^i) + \lambda d'] - P(x^i, y^i) \leq 0.1X\{V\langle j\rangle (x^i, y^1)^T d' + \phi(x^i, y^i) - P(x^i, y^i)\}$$

$$P(x^i, y^i) = \phi(x S y y^{11}) + \sum_{j=1}^{m} [\max(o, g_j)] + \sum_{j=1}^{m_{eq}} T c_j + [\nu_j h_j$$

for the modified Lagrangian

$$\mu_j = u_j^i$$

$$T_j = t_j^i$$

$$Vj = Vj^i$$

for the exact penalty

$$\mu = 2|u^i|$$

$$\wedge = 2|t_j^1| \operatorname{sgn}(c_j)$$

$$j \qquad w_j = 2|v_j^i| \operatorname{sgn}(h_j)$$

Each point along the line search requires a full function evaluation, as defined (for i«l) in Step 4. The initialization of the tear variables for the iterative convergence is:

For Q/LAP: y^i (or stored values from previous point of the line search)

For RFV : $y^1 + Xo)^1$ (from Step 7)

For CFV : $y.^* + Xd\backslash.$ k=l,i

Step 10> If the line search succeeds, save the resulting FFE as point $1+1_f$ then increment i, and retutn to Step 5.

If the line search fails, or the QP of Step 7 does not generate a direction of descent, reset $B^i = B°$ (usually • I), and return to Step 7.

If re-initialization of B fails, with no further improvement, terminate calculation.

III • Adjustable Parameters for Optimization Algorithms

In this section we describe several tuning parameters that must be specified appropriately for efficient performance.

First a suitable scale factor must be found for the design variables. All of these variables have prespecified bounds* A scaling vector is chosen such that each design variable is divided by its corresponding element in the scaling vector. The elements of the scaling vector also multiply respective elements of the objective function and constraint gradient vectors.

The reasons for scaling are twofold. Some variables may have very large or very small gradients. Over the course of the optimization an accumulation of roundoff error can cause design variables with small gradients to be under-emphasized. If gradients are too large, the updating information for the Hessian is also large and can lead to a matrix that is ill-conditioned in the quadratic programming step. Also, for convenience the initial Hessian approximation is set to $\underline{\underline{I}}$. If this approximation is poor, the quadratic program may make poor progress initially or even generate a search direction that causes the line search to fail. Appropriate scaling of the gradients prevents these problems from occurring.

Perturbation factors need to be carefully chosen in the gradient calculation step. The perturbation factor is a small prespecified parameter £ such that

$$Ax_j = \max(£x_j, q) \quad \text{or} \qquad (12)$$
$$Ay_j = \max(^, q)$$

where q is a small preset number that prevents the perturbation from being zero. If the perturbation size is too large, Taylor series error due to nonlinear functions may be introduced. A perturbation size too small may lead to a response obscured by convergence error in recycle calculations.

Thus this factor must always be specified relative to the magnitude of the recycle convergence tolerance. If nondifferentiable or discontinuous functions are present, choosing an appropriate perturbation factor is often difficult and arbitrary because considerable error results from nonsmooth functions.

Finally, a Kuhn-Tucker convergence tolerance must be chosen for the optimization stopping criterion. This parameter has the same units as the objective function. The K-T error (LHS of equation (10) or (11)) is mainly influenced by the magnitude of the last search direction and the degree of infeasibility at the last base point. The search direction is in turn affected by any error in objective function and constraint gradients. Thus the K-T tolerance is difficult to determine and often is best determined a posteriori. By setting the tolerance to zero and allowing the optimization to run until a line search or quadratic programming failure occurs, the K-T error may then be examined at the last few base points.

IV. Example Problem

To illustrate the mechanics of the three algorithms and study some of their tuning parameters, a simple process problem was simulated and optimized. SPAD[13] installed on a UNIVAC 1100/82 computer, was used as the simulation framework for the optimization study. This simulator is small and flexible enough for interfacing with the optimization algorithm,and changes in the program executive can easily be made. Like large process simulators it has the same unit operations, stream handling and convergence capabilities for material and energy balances and equipment sizing. Unlike other simulators, however, SPAD could easily be installed on small inexpensive computers and executed cheaply with completely accessible code.

A flowsheet for the simple process problem is given in Figure 3. Hydrocarbon feed is mixed with a recycle and flashed adiabatically. The overhead is withdrawn as product and a fraction of the bottoms is removed. The rest is pumped around a recycle. This problem, although very simple, has many characteristics of complex sequential modular simulations. Recycle calculations are needed to converge a flowsheet made up of modules connected by streams. Iterative calculations are performed for the flash separation as well as for stream enthalpies. Also, an underlying set of subroutines is accessed for calculating physical properties. In this case, because hydrocarbons are present, simple physical property relations are used. A block diagram of this process as simulated on SPAD is given in Figure 4.*

For the optimization problem the flash pressure and the two bottoms splits were chosen as design variables. A linear equality constraint forces the sum of the two split fractions to unity. Bounds are also placed on the design variables.

Two different objective functions were used for the optimization study.

---

* Note that a special optimization block substitutes for the convergence block when feasible variant algorithms are used. Q/LAP retains the recycle convergence block but requires a separate interface to the simulator.

In the first set of optimizations, the objective was to maximize the amount of propane in the vapor product. The second objective was to maximize a nonlinear expression containing the first five component flow rates of the vapor product. Problem specifications are given in Figure 4.

For Q/LAP the flowsheet was first decomposed into the modules shown in Figure 4. Linear module models were constructed by perturbation of input streams and design variables. The models were then collected to form a large sparse linear system of equations. The incidence matrices of this system are presented in Figure 5. Here the blocks numbered horizontally and vertically correspond to the stream labels in Figure 4. The X's in the incidence matrix are the nonzero terms; an off-diagonal block of X^fs represents the coefficient matrix with output streams of the vertical label and input streams with the horizontal label. Note that the feed is not considered in this system because it is fixed throughout the optimization. As can be inferred from Figure 4 and the incidence matrix in Figure 5, row block 1 contains the models of the mix module; row blocks 2 and 3 contain flash module models; row blocks 4 and 5 contain split models and the last row block contains the pump model. Diagonal elements are, of course, equal to 1. Because the flowsheet has already been converged for the modelling step, the Wegstein block is removed and the torn stream is joined.

The three right-hand side columns correspond to perturbations of the three design variables. Note that the nonzero column elements only occur in the rows corresponding to the module where the design variable is present.

For CFV and RFV the set-up was much simpler. Here the tear algorithm or convergence block in, Figure 4 was merely replaced with an optimization block. Otherwise, the same flowsheet calcula- sequence was used as with simulation. The additional steps for optimization are all internal to the optimization block and do not affect the simulator's structure.

Monotonic Function Optimization

The first problem has a monotonic objective function with respect to the design variables. The optimum, as can be seen by inspection, is found at the lower bounds of the bottoms product split fraction and the flash pressure. A contour plot of this objective funciton for these two degrees of freedom is given in Figure 6. Note that the effect of the split fraction in the contour plot is very slight. This is due to the small enthalpy change in the pumping module. If the loop were completely adiabatic, the split fraction would have no effect on the objective function.

A summary of results is given in Tables 1, 2 and 3 for Q/LAP, RFV and CFV, respectively. Here, different values of the recycle convergence tolerance, perturbation factor and variable scaling vector were tried from two different starting points. The best result for Q/LAP required only one new base point and only 2.51 simulation time equivalents (STE's). Total CPU time was 6.426 seconds. The best monotonic problem solution with CFV required 5.812 CPU seconds, 2.27 STE's and also, only one new base point. RFV was the most efficient algorithm for this problem. It required only 4.613 CPU seconds, one new base point and 1.8 STE's.

Ridae Function Optimization

This problem has an optimum at the upper bound of the bottoms split fraction and at an intermediate flash pressure of 22.8 psia. A contour plot for this objective function with these two degrees of freedom is given in Figure 7. Again, different perturbation factors, convergence tolerances and scale factors were tried from two starting points. These are also listed in Tables 1 to 3. Here the best run for Q/LAP required six new base points, 12.811 CPU seconds and 18.98 STE's. CFV required almost twice as much effort as Q/LAP with 22.24 CPU seconds, 32.95 STE's and 10 new base points. RFV, on the other

hand, was only slightly slower than Q/LAP with 6 new base points, 13.2 CPU seconds and 19.5 STE.'s. Note that in Figures 6 and 7 the three algorithms follow the same path on the monotonic problem but diverge slightly on the ridge problem. In Figure 7 both Q/LAP and RFV follow similar paths while CFV takes smaller steps and requires more time for the optimiztion.

Because Q/LAP and RFV use the same quadratic programming step, any difference in performance is merely due to the accuracy of the gradient calculation strategy. CFV, on the other hand, solves a larger quadratic program that includes tear variables and equations at each step. Here, in addition to specifying bounds on design variables, restrictions on tear variables (e.g. molar flows ^ 0) are also imposed. Thus the search direction generated by CFV is more restricted than with RFV because tear variable bounds may be active. While this quadratic program generates smaller search directions, it also prevents large extrapolations on highly nonlinear surfaces.

V. Analysis of Adjustable Parameters for Flash Problems

The results in Tables 1, 2 and 3 offer some limited guidelines and heuristics for the determination of adjustable algorithmic parameters.

To scale the variables, a good rule of thumb for all three algorithms was to choose a scale vector with elements of integral powers of 10 such that the initial gradient has elements with absolute values between 10 and 100. Apparently, this heuristic makes the identity matrix a satisfactory approximation to the initial Hessian for these problems. Experience with these two process problems has shown that algorithmic performance is not very sensitive to small changes in scaling, but large deviations from a good scale vector (such as in run C-23 in Table 3) may lead to poor performance or premature termination.

Choosing an appropriate perturbation factor is especially important for Q/LAP and RFV. In both cases we use the interconnection (or tear) equations, $h(x,y) = 0$, to obtain reduced gradient information. However, we ignore the recycle convergence error during this step. Here:

$$\frac{\partial h}{\partial y}^T \Delta y + \frac{\partial h}{\partial x}^T \Delta x = h$$

where $\| h \|_\infty < \eta$ and η is the convergence tolerance. For each perturbation of x_j, we get:

$$\frac{dy}{dx_j} \approx \frac{\Delta y}{\Delta x_j} = - \left[\frac{\partial h}{\partial y}^T\right]^{-1} \left[\frac{\partial h}{\partial x_j}^T - \frac{h}{\Delta x_j}\right]$$

Now if $|(\partial h/\partial x_j)^T|$ is not $>>|h/\Delta x_j|$, then dy/dx_j will be in error. Also because,

$$\frac{d\phi}{dx} = \frac{\partial \phi}{\partial r}^T \left[\frac{\partial r}{\partial y}^T \frac{dy}{dx} + \frac{\partial r}{\partial x}\right] + \frac{\partial \phi}{\partial x}$$

$$\frac{dg}{dx} = \frac{\partial g}{\partial r}^T \left[\frac{\partial r}{\partial y}^T \frac{dy}{dx} + \frac{\partial r}{\partial x}\right] + \frac{\partial g}{\partial x}$$

$$\frac{dc}{dx} = \frac{\partial c}{\partial r}^T \left[\frac{\partial r}{\partial y}^T \frac{dy}{dx} + \frac{\partial r}{\partial x}\right] + \frac{\partial c}{\partial x}$$

the recycle convergence error affects all of the reduced gradients. Note in the above equations that the gradient error due to h becomes smaller as Δx_j increases. Thus, a rule of thumb is to choose the perturbation factors for y and x so that the effect of the perturbation is larger than the convergence tolerance. As an example, consider the monotonic flash problem solved by Q/LAP. Here the bottoms split has only a slight influence on the objective function. Thus its small gradient may easily be obscured by error if the perturbation size is too small. As seen in Figure 8, a perturbation size of 0.1 (run Q-3) approximates the true response surface well and leads to efficient location of the optimum, even though the convergence tolerance is only 10^{-4}. With a perturbation factor of 10^{-2} (run Q-7), however, the bottoms split has almost no effect on the approximated surface and this case terminates before locating the optimum.

For the ridge problem, the objective function gradients are not small and, as seen in Table 1, the optimum is always found as long as the perturbation factor is larger than the convergence tolerance. Figure 9 illustrates how Q/LAP successively approximates the response surface for run Q-10 in Table 1. Figures 10 and 11 show how the surface approximations are altered if the perturbation factor is too large (run Q-14) or too small (run Q-13).

For RFV, all of the unsuccessful runs in Table 2 had perturbation factors less than or equal to the convergence tolerance. In run R-14 in Table 2, RFV terminates at the optimum with the perturbation size and convergence tolerance both set to 10^{-3}. Here, however, the

gradient at the optimum is so inaccurate that the algorithm fails to recognize the optimum as a Kuhn-Tucker point (K-T error = 59.1) and concludes with a line search failure.

The perturbation factor is less difficult to choose for CFV than for RFV, but it still must be selected with caution. Here, tear equations are now part of the quadratic program and the values of h are included in the QP calculation. However, convergence error is also included in the Hessian update and in the termination criterion. While CFV is not as sensitive as Q/LAP or RFV in dealing with convergence error, choosing a larger perturbation factor does seem to help performance. To compare cases, consider run C-6 in Table 3 and run R-9 in Table 2. Both runs have perturbation factors and convergence tolerances of 10^{-3}. Here, the CFV run requires five base points but terminates at the optimum while the RFV run falls short of the optimum because of gradient inaccuracy.

Tables 1, 2 and 3 also give the Kuhn-Tucker error at the optimum for all of the successful runs. For the RFV and Q/LAP runs the K-T error for the monotonic problem is obviously small because the solution lies at a vertex. For both problems, choosing a K-T tolerance of the same order of magnitude as the perturbation factor worked reasonably well. For CFV, tear equation values form part of the Lagrangian and the K-T tolerance must be chosen a little larger. In Table 3 one sees the K-T error is about the same as the convergence tolerance. Here, care must be taken in choosing the K-T tolerance because the quadratic program seeks a point where the tear equations are exactly satisfied. Consequently if the K-T tolerance is chosen too small, CFV will merely spend a lot of time trying to enforce $h(x,y) = 0$ at the optimum.

In summary, the following heuristics are proposed for adjusting the parameters of the three algorithms:

1) Scale the variables so that the elements of the initial gradient vector have absolute values between 10 and 100.

2) Choose a perturbation factor such that its response is larger than the recycle convergence error.

3) Choose a K-T tolerance that is about the same as the objective function response to the perturbation factor. For CFV, this tolerance must be large enough to include the recycle error at the optimum.

On the above test problems the RFV algorithm is superior to CFV. The reasons for this are:

1) The inclusion of tear variables in the quadratic program leads to longer CPU times for the quadratic programming step in CFV.

2) Restricted search directions are calculated for CFV because bounds are imposed on the tear variables.

Several improvements suggest themselves from the analysis in this section. For CFV, an obvious improvement would be removal of **the** tear equations from the Lagrangian terms, the termination criterion **and** the updating equations. RFV and Q/LAP could be improved by including the convergence error in the calculation of the reduced gradients. Both improvements require additional implementation and should probably be made after further testing establishes the effectiveness of these algorithms.

VI. Comparison of Flash Problem Results

Table 4 presents a summary of flash problem results for five algorithms. In addition to the best runs of Q/LAP, RFV and CFV, we include results for IPOSEQ [10] from two starting points. We also present four optimization runs solved by CPX, a modified Complex [14] method that treats the simulation as a black box. Although CPX is much slower than any of the other algorithms, it is included because similar strategies are most often chosen for optimization of industrial problems [15]. Clearly, use of any of the other four algorithms results in significant savings in computational effort.

The RFV algorithm was generally the fastest on these simple problems. CFV tended to be slower because the additional bounds on tear variables led to smaller search directions. Q/LAP performed reasonably well in all four cases. Its main disadvantage, however, is the requirement of a more complicated gradient calculation strategy and interface to the simulator. Finally, in three of the four comparisons RFV was significantly faster than IPOSEQ. This suggests that feasible path algorithms should be considered if gradients are evaluated by extensive numerical perturbation.

In considering these feasible path methods over IPOSEQ, it is apparent that there could be trade-offs in performance. Both approaches require roughly the same effort from the simulator to set up the quadratic program. IPOSEQ does this at every iteration while the feasible path algorithms set up the QP at converged base points. Since IPOSEQ generally (but not always) requires more iterations, we find ourselves trading the effort required to converge the flowsheet (with feasible path) for the effort of evaluating the gradients and solving the QP (with IPOSEQ).

Finally, it should be noted that feasible path algorithms, unlike

IPOSEQ, offer the advantage of providing usable and improved solutions, even if the algorithm fails to converge. This feature and the performance characteristics observed above illustrate the potential attractiveness of these algorithms for process flowsheet optimization.

Acknowledgements

The authors express appreciation for the support of the Paul A. Gorman Fellowship from the International Paper Company Foundation (for L.T.B.) and of the Engineering Experiment Station of the University of Wisconsin.

REFERENCES

1. Friedman, P. and K. L. Pinder, "Optimization of a Simulation Model of a Chemical Plant," IEC Proc. Des. Dev., 11, 4, p. 512 (1972),
2. Adelman, A. and W. F. Stevens, "Process Optimization by the 'Complex' Method," AIChE J., 18, 1, p. 20 (1972).
3. Gaines, L. D. and J. L. Gaddy, "Process Optimization by Flowsheet Simulation," IEC Proc. Des. Dev. JL5, 1, p. 206 (1976).
4. Ballman, S. H. and J. L. Gaddy, "Optimization of Methanol Process by Flowsheet Simulation," IEC Proc. Des. Dev., JJ5, 3, p. 337 (1977).
5. Hughes, R. R., "Optimization Methods for Block Simulation," presented at VI Interamerican Congress of Chemical Engineers, Caracas (1975).
6. Parker, A. L. and R. R. Hughes, "Approximation Programming of Chemical Processes - 1," CACE, J5, 3, p. 123 (1981).
7. Parker, A. L. and R. R. Hughes, "Approximation Programming of Chemical Processes - 2," CACE, 1, 3, p. 135 (1981).
8. Biegler, L. T. and R. R. Hughes, "Approximation Programming of Chemical Processes with Q/LAP," Chem. Eng. Prog., J77, 4, p. 76 (1981).
9. Powell, M.J.D., "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," presented at 1977 Dundee Conference on Numerical Analysis (1977).
10. Biegler, L. T. and R. R. Hughes, "Infeasible Path Algorithms for Sequential Modular Optimization", AIChE J., 28, 6, p.994, (1982)
11. Dennis, J. E. and J. J. More, "Quasi-Newton Methods: Motivation and Theory", SIAM Review, 19, 1, p.46, (1977)
12. Biegler, L. T., PhD Thesis, University of Wisconsin, (1981)
13. Hughes, R. R., R. K. Malik and L. T. Biegler, "SPAD - Simulator for Process Analysis and Design", EES Report #52, University of Wisconsin, (1981)

14, Box, M. J., "A New Method of Constrained Optimization and a Comparison with Other Methods", Computer J., 8, 1, p.42 (1965)

15. Westerberg, A. W., "Optimization in Computer Aided Design", in Foundations of Computer Aided Process Design, (Mah and Seider, eds.)# p. 149, Engineering Foundation, New York, (1981)

Table 1: Flash Problem Optimization With Q/LAP

Run No.	Start[a]	Scale Index[b]	Conv. Tol.	Perturb. Fraction[d]	K-T Error[e]	Best Obj. Val.	CPU Secs.[g]	Base Pts.	Total STE's[h]
Monotonic Objective									
Successful runs									
Q-1	A	031*	10^{-4}	10^{-1}	$<10^{-7}$	7.233[f]	7.090	2	2.77
Q-2	A	"	10^{-3}	10^{-2}	$<10^{-7}$	"	6.104	2	3.88
Q-3	A	"	"	10^{-1}	$<10^{-7}$	"	5.981	2	3.80
Q-4	A	000	10-*	"	$<10^{-10}$	"	20.144	9	7.86
Q-5	A	030	"	"	$<10^{-7}$	"	10.624	4	4.15
Q-6	B	031*	"	"	$<10^{-4}$	"	6.426	2	2.51
Unsuccessful runs									
Q-7	A	031*	10"*	10^{-2}	0.13	7.130	13.594	3	5.30
Q-8	A	032	"	"	0.090	7.130	14.446	3	5.64
Q-9	A	020	"	"	0.094	7.124	14.205	3	5.54
Ridge Objective									
Successful runs									
Q-10	A	021*	10^{-4}	10^{-2}	$<10^{-3}$	6.935	28.981	10	42.9
Q-U	B	"	"	"	<io"*	6.937[f]	14.860	7	22.0
Q-12	B	"	"	10^{-3}	<10"*	"	12.811	6	19.0
Unsuccessful runs									
Q-13	B	"	10^{-4}	io-*	$<10^{-3}$	6.013	25.687	6	38.0
Q-14	B	"	"	10^{-1}	3.71	5.840	27.149	4	40.2

Table 1

Notes

a) Two staring points were used:

A - x - {0.5,40,0.5}

B - x - {0.4,30,0.6}

b) Scale index: each digit (numbered from the left) is the logarithm of the scale factor (i.e. the power of 10) applied to the corresponding variable as a divisor, and thus to the funtion gradient element as a multiplier.

(* indicates scale factors have been chosen to make the absolute value of the Initial scaled objective function elements, $|(y^{\wedge}) I^{*}$ fall between 10 and 100).

c) Tolerance fraction for iterative closure of tear variables at each base point calculation.

d) Perturbation fraction, ϵ , In equation (12)

e) The value of the left-hand-side of equation (10) or (11) when the calculation terminates. In these calculations, the test value, e, was set at zero, the calculation runs terminated with either line search or quadratic programming failure.

f) Maximum or optimum value closely approximated .

g) On UV-Madison Univac 1100/82.

h) STE = CPU time/t , where t - CPU time for simulation at the optimum. Value of t used depends on problem, and Convergence tolerance fraction, as follows

Convergence tolerance fraction	Monotonic objective	Ridge objective
10^{-3}	1.575	0.513
10^{-4}	2.563	0.675

Table 2: Flash Problem Optimization With RFV

Run No.	Start	Scale Index [a]	Conv. [b]	Perturb [c]	$K_{\sim}T_G$ [d]	Best	CPU [e] Sees.	Base Pts.	Total STE's [f]
Monotonic Objective									
Successful runs									
R-1	A	031*	10^{-4}	10^{-3}	$<10^{-7}$	7.233[g]	5.059	2	1.97
R-2	A	"	"	10^{-6}	$<10^{-8}$	"	6.814	3	2.66
R-3	A	032	10^{-3}	"	$<10^{-7}$	7.232	4.256	2	2.70
R-4	A	031*	10^{-4}	10^{-1}	$<10^{-7}$	7.233	4.613	2	1.80
R-5	B	"	10^{-3}	10^{-2}	$<10^{-7}$	"	8.237	5	5.23
R-6	B	032	"	"	$<10^{-7}$	"	7.893	5	5.01
R-7	B	031	10^{-4}	"	$<10^{-6}$	"	4.978	2	1.94
R-8	B	"	"	10^{-1}	$<10^{-7}$	"	6.232	2	2.43
Unsuccessful run									
R-9	B	031*	10^{-3}	10^{-3}	$<10^{-5}$	7.039	3.075	2	1.95
Ridge Objective									
Successful runs									
R-10	A	021*	10^{-4}	10^{-3}	$<10^{-3}$	6.938[g]	28.491	11	42.2
R-11	A	"	"	10^{-2}	$<10^{-2}$	6.929	26.895	9	39.8
R-12	B	"	"	10^{-3}	$<10^{-4}$	6.937[g]	13.200	7	19.6
R-13	B	"	10^{-3}	10^{-2}	$<10^{-3}$	6.935	11.211	6	21.9
R-14	B	"	"	10^{-3}	59.1	6.935	20.155	9	39.3
R-15	B	022	10^{-4}	"	$<10^{-3}$	6.937[g]	15.550	10	23.0
Unsuccessful runs									
R-16	A	021*	10^{-3}	10^{-3}	$<10^{-3}$	5.924	26.292	10	51.3
R-17	A	"	10^{-2}	"	$<10^{-3}$	6.392	14.001	9	32.4

Footnotes - see Table 1

Table 3: Flash Problem Optimization With CFV

Run No.	Specifications: Start [a]	Scale Index [b]: X	Scale Index [b]: Y	Conv. Tol. Fr. [c]	Perturb. Frac. [d]	Results: K-T Error [e]	Best Obj. Val.	CPU [g] Secs.	Base Pts.	Total STE's [h]
Monotonic Objective										
Successful runs										
C-1	A	222	2433334	10^{-3}	10^{-3}	0.40	7.233 [f]	15.811	3	10.04
C-2	A	"	"	"	10^{-2}	$<10^{-3}$	"	5.660	2	3.59
C-3	A	"	"	10^{-4}	"	$<10^{-4}$	"	5.812	2	2.27
C-4	A	"	"	"	10^{-3}	$<10^{-4}$	"	6.202	2	2.42
C-5	B	"	"	"	"	$<10^{-4}$	"	7.734	3	3.02
C-6	B	"	"	10^{-3}	"	$<10^{-3}$	"	12.080	5	7.67
C-7	B	"	1311113	10^{-4}	"	$<10^{-5}$	"	12.035	5	4.70
C-8	B	323	"	"	"	$<10^{-4}$	"	11.720	5	4.57
C-9	B	013	2224334	"	"	$<10^{-4}$	"	13.748	6	5.36
Unsuccessful run										
C-10	A	222	2433334	10^{-2}	10^{-3}	5.85	7.133	11.534	3	11.92
Ridge Objective										
Successful runs										
C-11	A	013	1113223	10^{-4}	10^{-3}	$<10^{-3}$	6.937 [f]	48.227	24	71.4
C-12	A	023	1224324	"	"	$<10^{-3}$	"	39.442	17	58.4
C-13	B	013	1113223	"	"	$<10^{-3}$	"	26.046	13	38.6
C-14	B	"	"	"	10^{-2}	$<10^{-3}$	6.936	22.240	11	33.0
C-15	B	212	"	"	10^{-3}	$<10^{-3}$	6.937 [f]	43.886	22	55.0
C-16	B	013	2224334	"	"	$<10^{-2}$	6.935	33.833	14	50.1
Unsuccessful runs										
C-17	B	013	1113223	10^{-2}	10^{-3}	22.2	6.057	23.598	9	60.5
C-18	B	"	"	10^{-3}	10^{-4}	100.	5.991	22.823	6	44.5
C-19	B	Oil	"	10^{-4}	10^{-3}	$<10^{-3}$	6.103	19.662	10	29.1
C-20	B	014	"	"	"	$<10^{-3}$	6.040	27.363	15	40.5
C-21	B	013	0002112	"	"	$<10^{-3}$	5.957	10.409	6	15.4
C-22	B	224	"	"	"	$<10^{-2}$	6.047	58.433	21	57.0
C-23	B	031	0000002	"	"	$<10^{-3}$	5.961	10.400	6	15.4

Footnotes - see Table 1

Table 4

Flash Problem Optimization; Summary of Best Results
(Lowest No. of STE's for Successful Runs)

Starting Point [a]	Algorithm	CPU secs. [g]	No. base points or iterations	Total STE's [b]
Monotonic Objective				
A	RFV	4.613	2	1.80
	CFV	5.812	2	2.27
	IPOSEQ	6.620	5	2.58
	Q/LAP	7.090	2	2.77
	CPX	82.129	273	52.1
B	RFV	4.978	2	1.94
	CFV	7.734	3	3.02
	IPOSEQ	6.660	5	2.60
	Q/LAP	6.426	2	2.51
	CPX	49.279	70	19.2
Ridge Objective				
A	RFV	26.895	9	39.8
	CFV	39.442	17	58.4
	IPOSEQ	12.856	10	19.0
	Q/LAP	28.981	10	42.9
	CPX	119.873	393	177.6
B	RFV	13.200	7	19.6
	CFV	22.240	11	33.0
	IPOSEQ	17.366	14	25.7
	Q/LAP	12.811	6	19.0
	CPX	136.468	463	202.2

Footnotes - see Table 1

Fig. 1 Flowsheet Module and Variables for Q/LAP

Fig. 2 Calculation Sequence and Variables for Feasible Variant Algorithms

Fig. 3 Flowsheet of Simple Flash Process

Fig. 4 Problem Data and Block Diagram

Fig. 5 Incidence Matrix for Q/LAP Solution of Flash Process

Fig. 6 Paths for Monotonic Flash Process

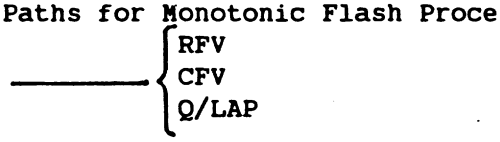


Fig. 7 Paths for Ridge Flash Process

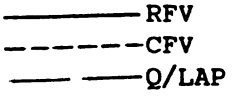


Fig. 8 Monotonic Problem - Effect of Perturbation Factor for Q/LAP

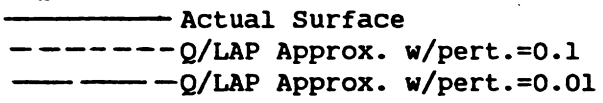


Fig. 9 Ridge Problem - Q/LAP Approximated Surface w/perturbation of 0.01

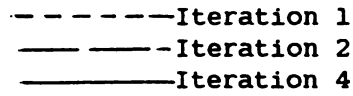


Fig.10 Ridge Problem - Q/LAP Approximated Surface w/perturbation of 0.1

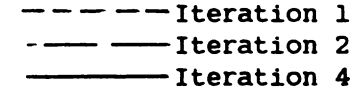
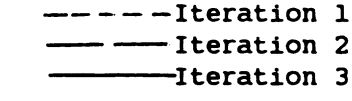
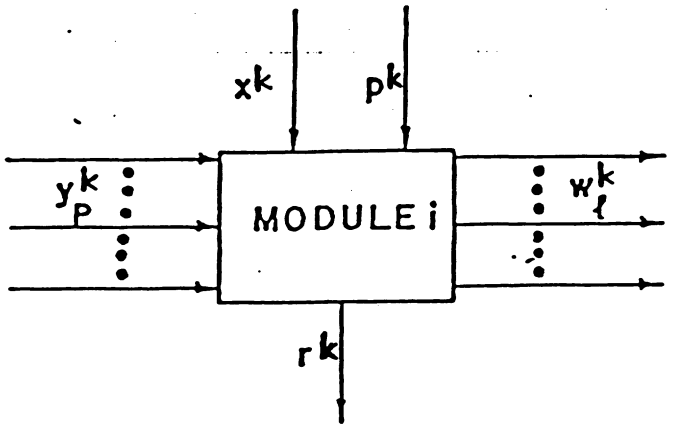


Fig.11 Ridge Problem - Q/LAP Approximated Surface w/perturbation of 10





y_p^k – INPUT STREAM VECTOR

x^k – DESIGN VARIABLE VECTOR

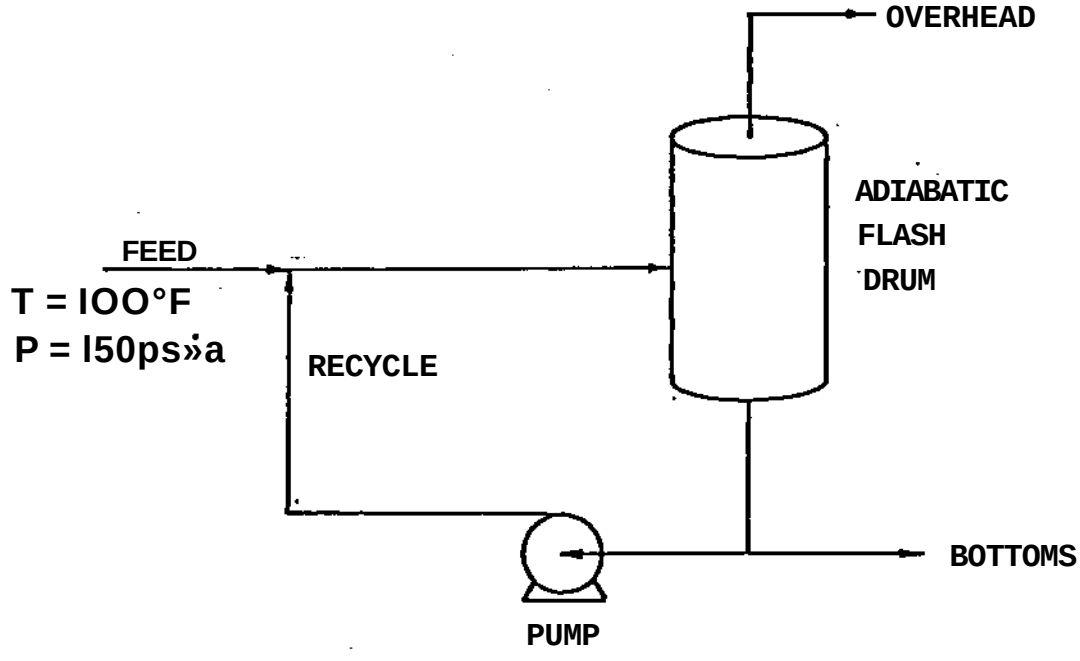
p^k – FIXED EQUIPMENT PARAMETERS

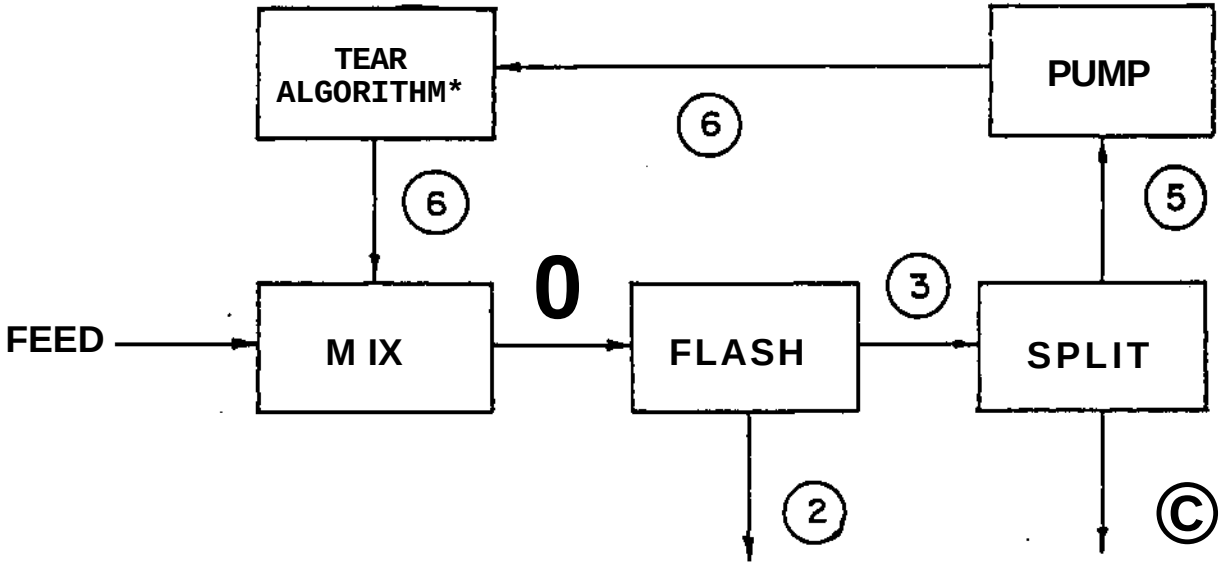
r^k – DEPENDENT (RETENTION) VARIABLE VECTOR

w_ℓ^k – OUTPUT STREAM VECTOR

FEED (lb-moles)

PROPANE	10
1 - BUTENE	15
N - BUTANE	20
TRANS - 2 - BUTENE	20
CIS - 2 - BUTENE	20
PENTANE	10





Component Flows
(lb-moles/hr) in Stream (2)

e_1 - Propane
e_2 - 1-Butene
e_3 - n-Butane
e_4 - t-2-Butene
e_5 - c-2-Butene

x_1 - split fraction of ©
x_2 - flash pressure (psia)
x_3 - split fraction of (T)

Monotonic Objective:

$$\text{Max}\ (e_1)$$

Nonlinear Objective:

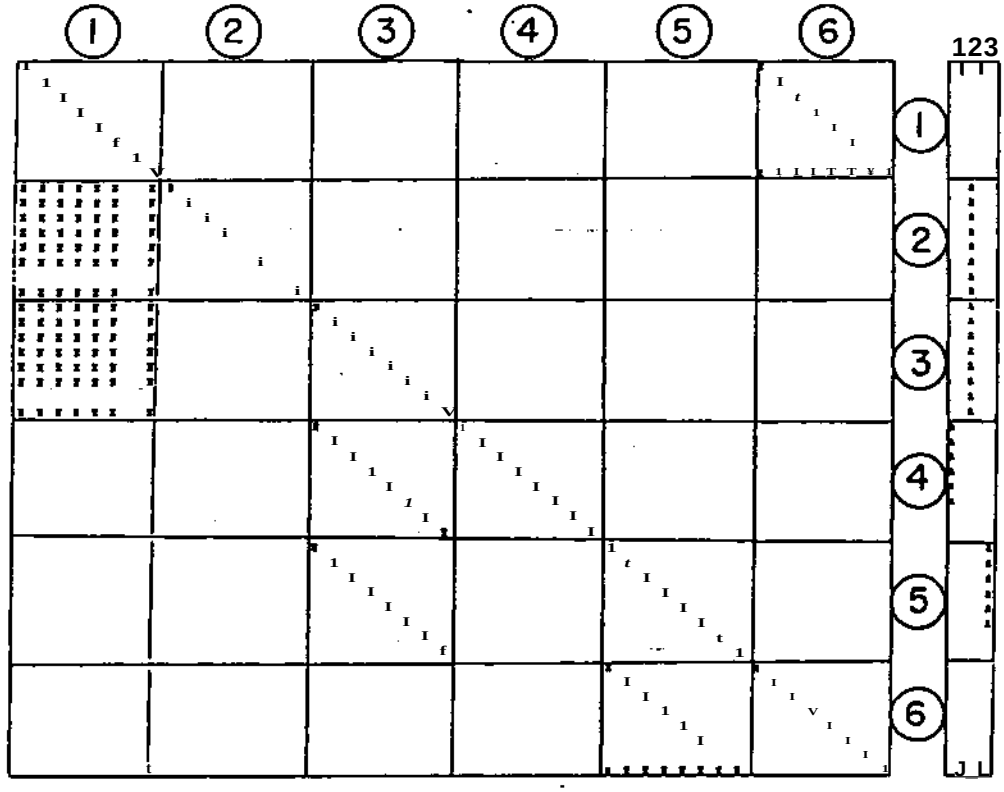
$$\text{Max}\ (\ x_1 x_2^{0.6} e_3^{0.9} e_4^{0.9} e_5^{0.8}\ )$$

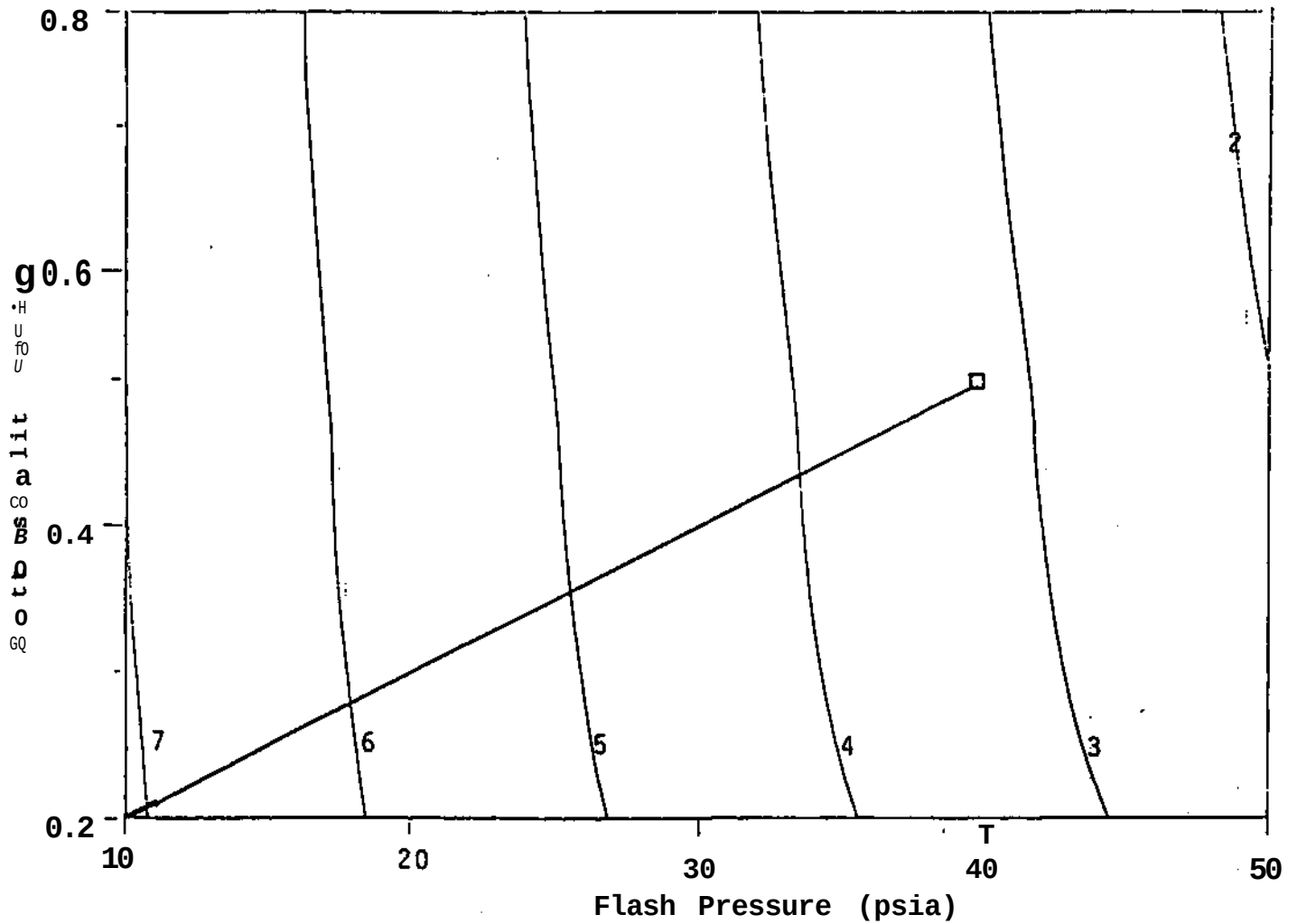
s.t. $0.2 \le x_1 \le 0.8$

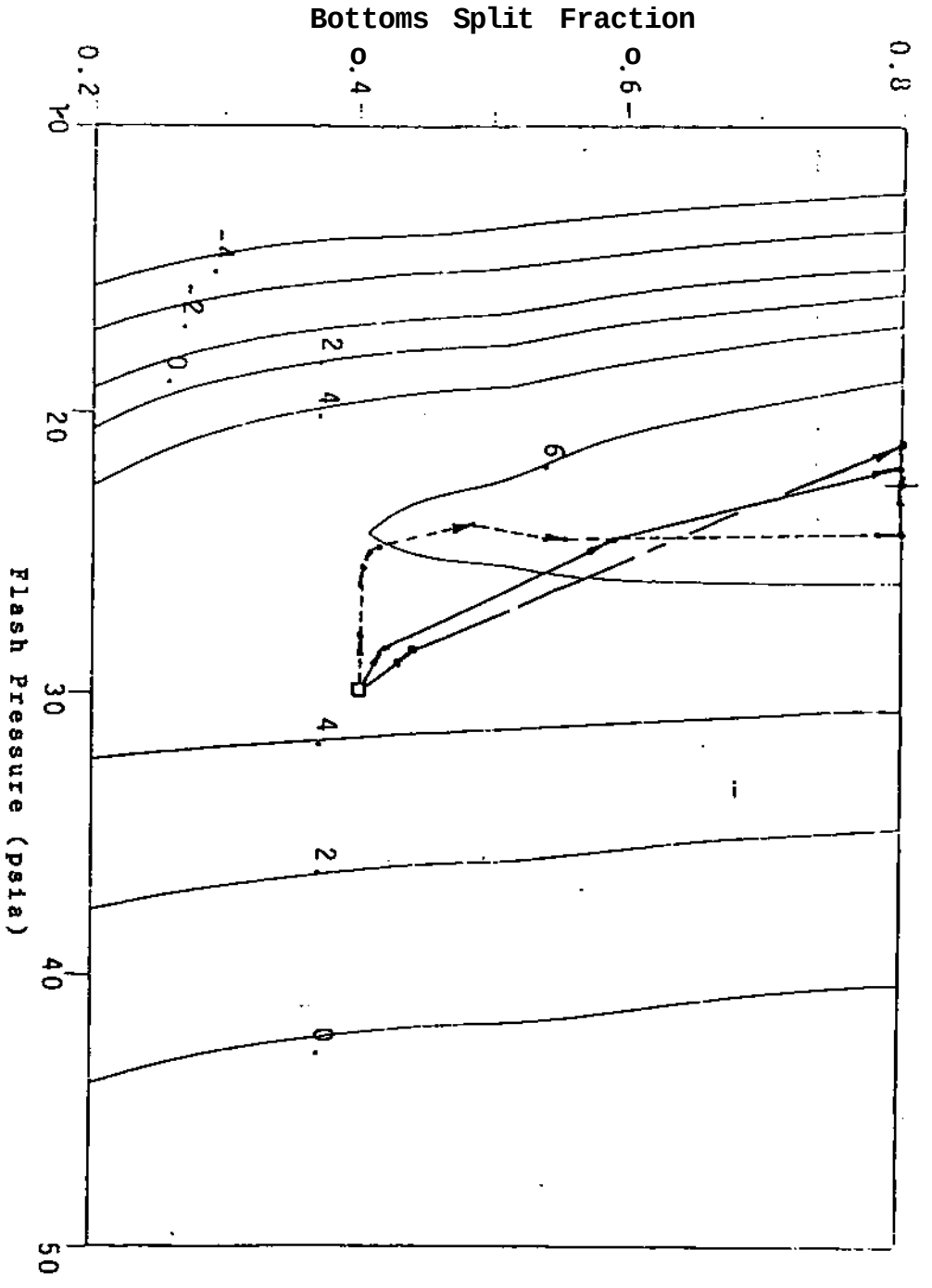
$10. \le x_2 \le 50.$

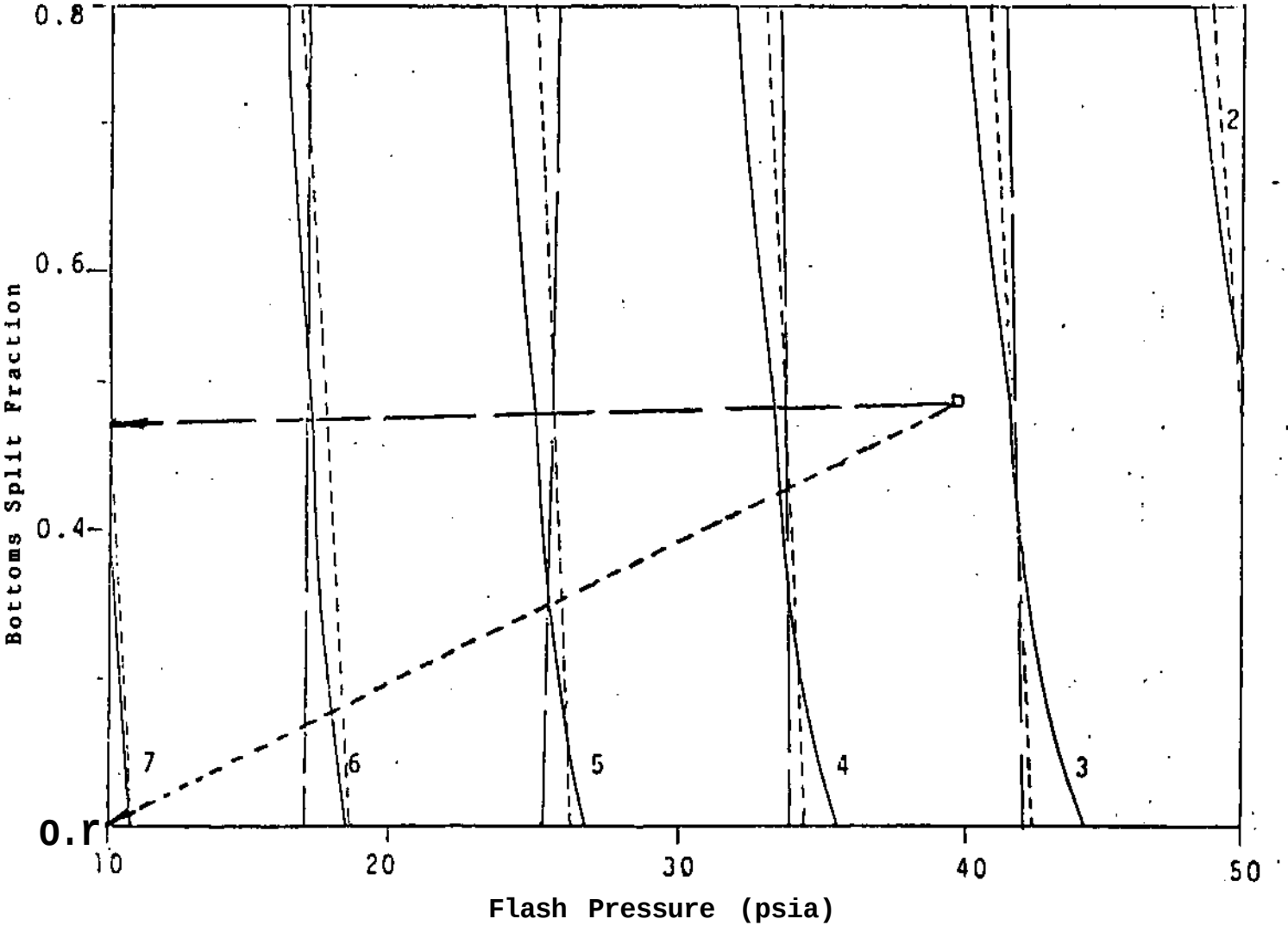
$x_1 + x_3 = 1.0$

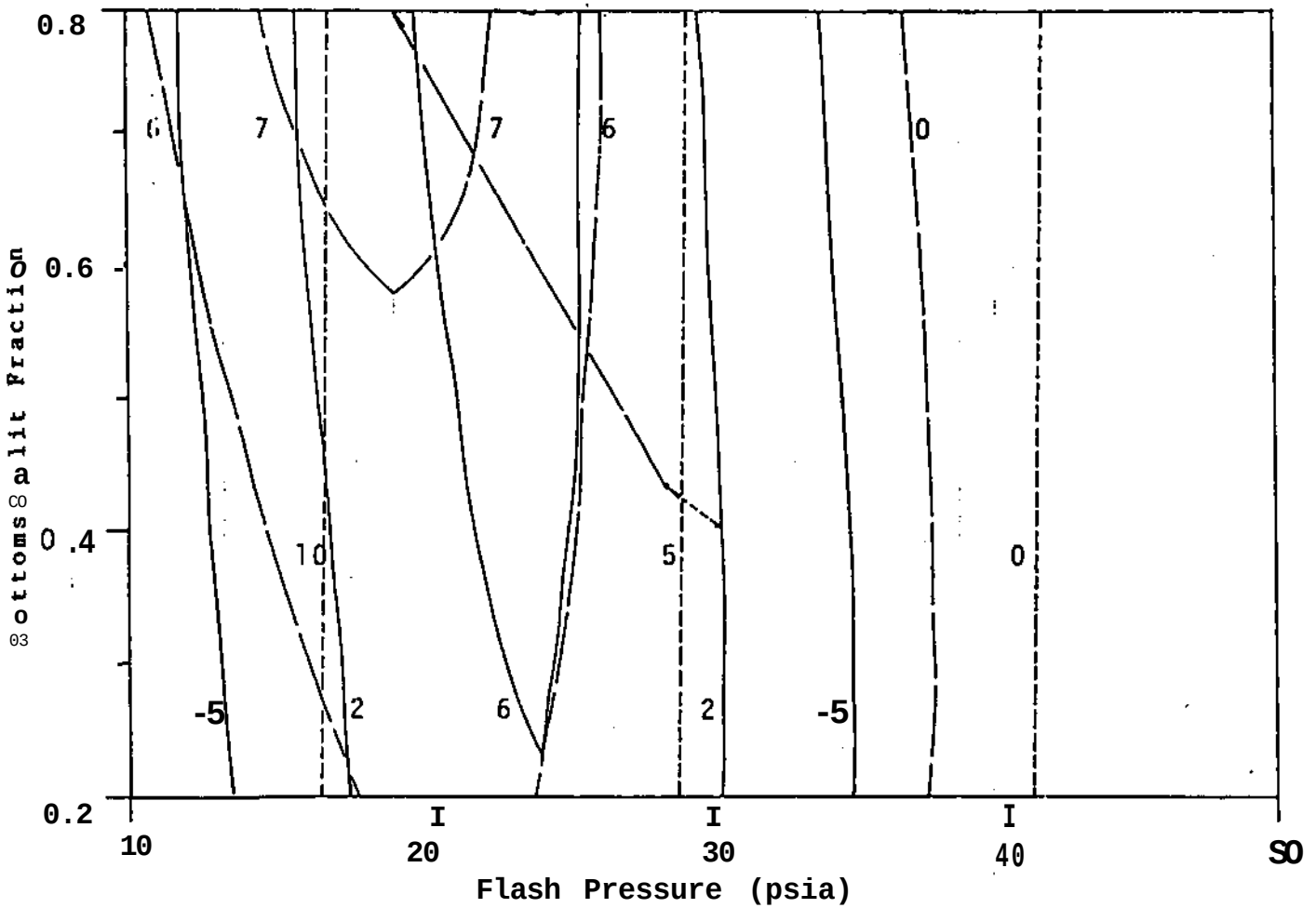
* for Q/LAP - Recycle Convergence Algorithm
for RFV and CFV - Optimization Algorithm

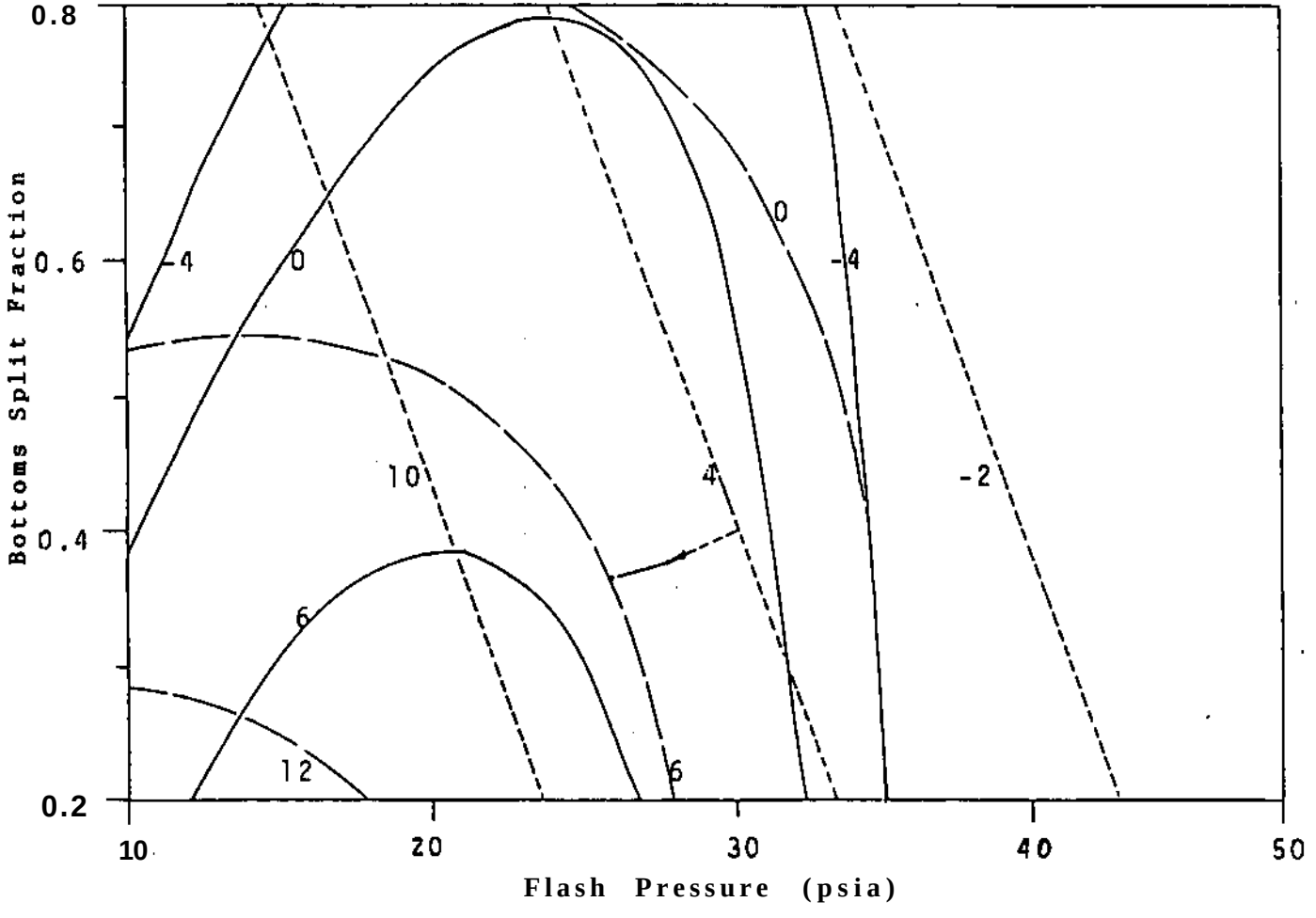

1
2
3
4
5
6
123
1
2
3
4
5
6

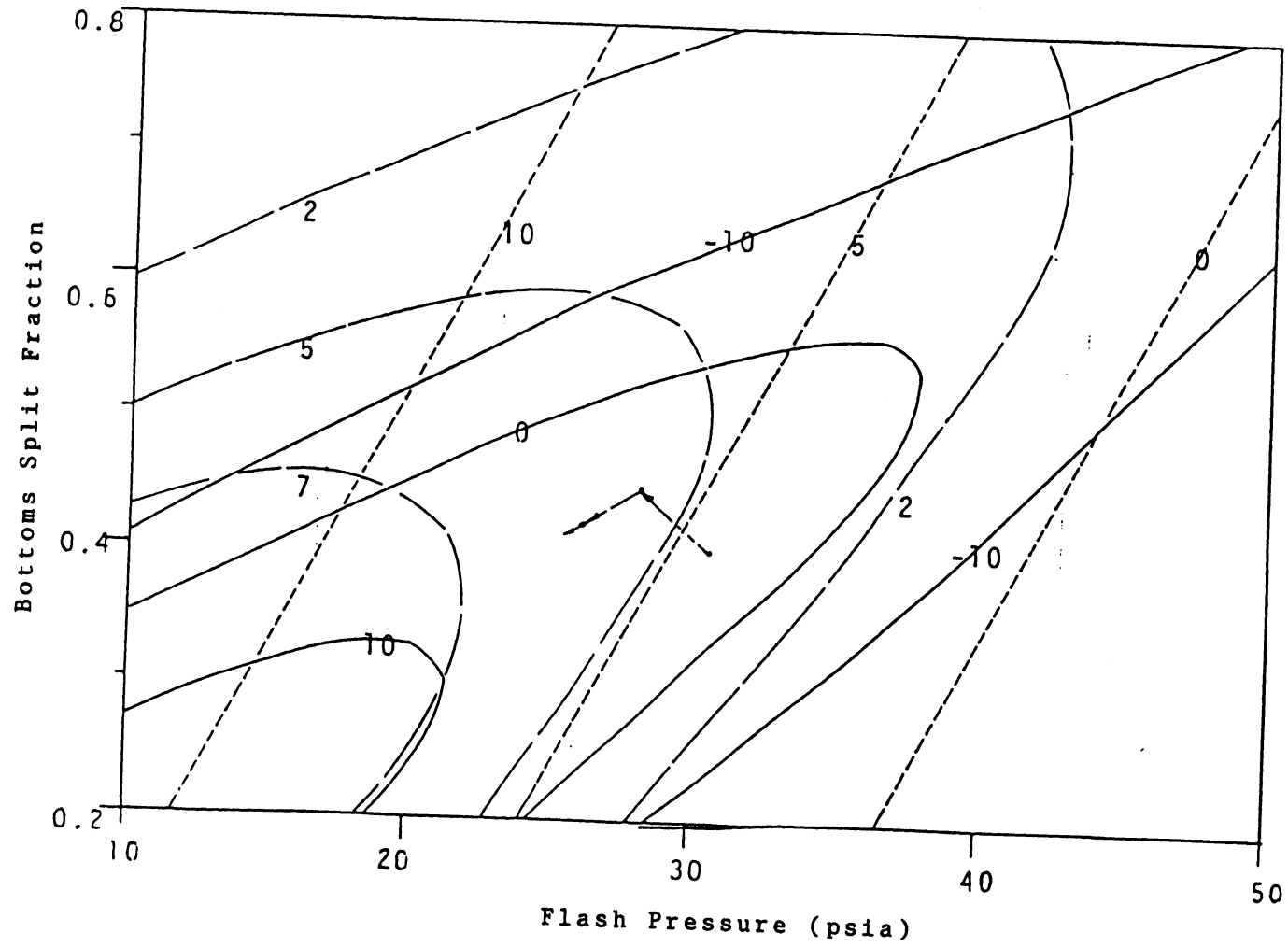

0.8
0.6
0.4
0.2
10
20
30
40
50
Flash Pressure (psia)
7
6
5
4
3
2
T

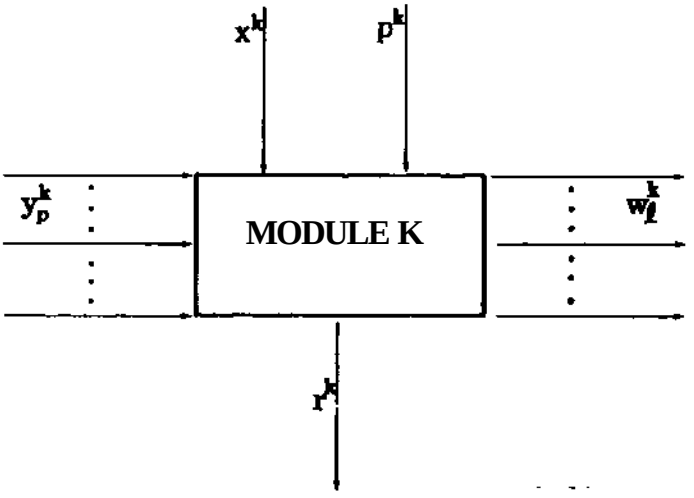

Bottoms Split Fraction
0.2
0.4
0.6
0.8
10
20
30
40
50
Flash Pressure (psia)
-4.
-2.
0.
2.
4.
6.
4.
2.
0.


0.8
0.6
0.4
0.r
Bottoms Split Fraction
10
20
30
40
50
Flash Pressure (psia)
2
3
4
5
6
7

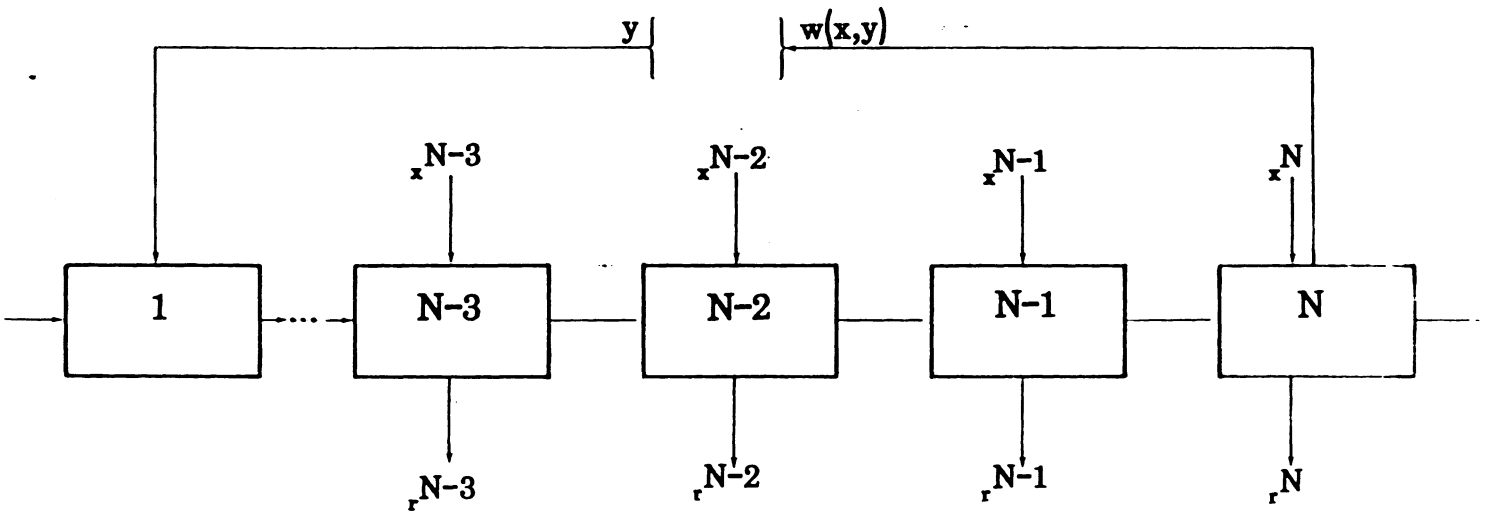

0.8
0.6
0.4
0.2
Bottoms Composition
6
7
7
6
0
10
5
0
-5
2
6
2
-5
10
20
30
40
50
Flash Pressure (psia)


0.8
0.6
0.4
0.2
Bottoms Split Fraction
10
20
30
40
50
Flash Pressure (psia)
-4
0
10
6
12
4
6
0
-4
-2


0.8
0.6
0.4
0.2
Bottoms Split Fraction
2
10
-10
5
0
5
0
7
2
-10
10
10
20
30
40
50
Flash Pressure (psia)



y_p^k - Input steam vector
x^k - Design variable vector
P^k - Fixed equipment parameters
r^k - Dependent (retention) variable vector
w_ℓ^k - Output steam vector



x^k - Design variables for kth module

r^k - Retention (dependent) variables for kth module

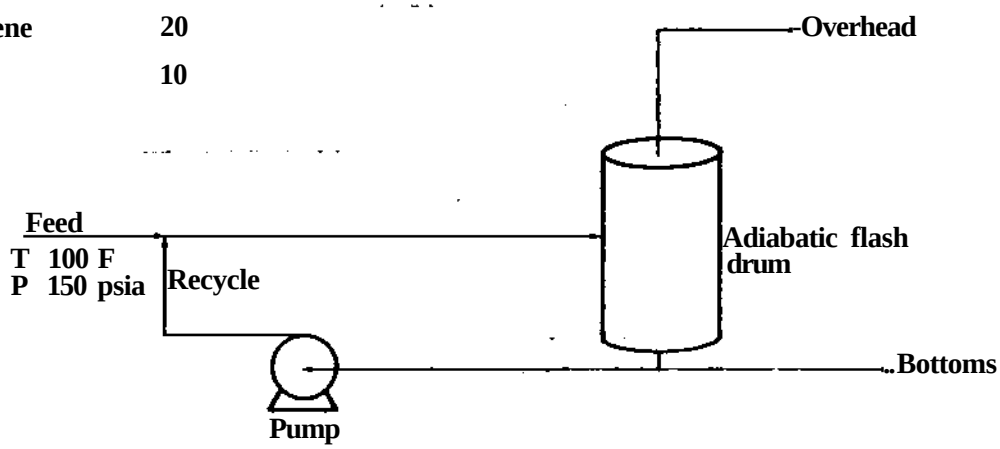
y - Tear variable vector (guessed)

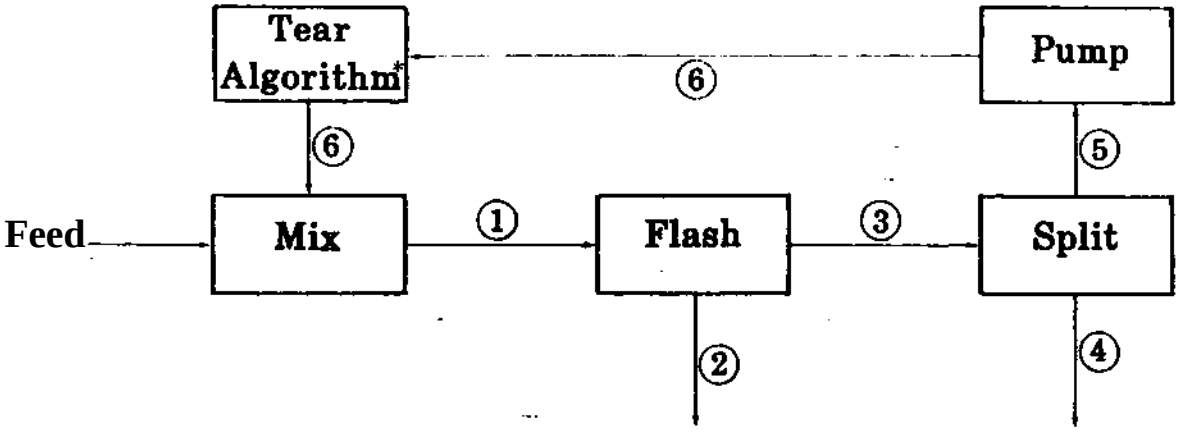
w(x,y) - Calculated tear stream vector

$h(x,y) \equiv y - w(x,y) = 0$, tear equations

Feed (lb-moles)

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





Component Flows (lb-moles/hr) in stream(2)

e_1- Propane
eg- 1-Butene
e_3- n-Butane
34- t-2-Butene
e^- c-2-Butene
x_2- split fraction of ^D
x_2- flash pressure (psia)
x_ - split fraction of ©

3

Monotonic Objective:

Max (e_x)

Nonlinear Objective:

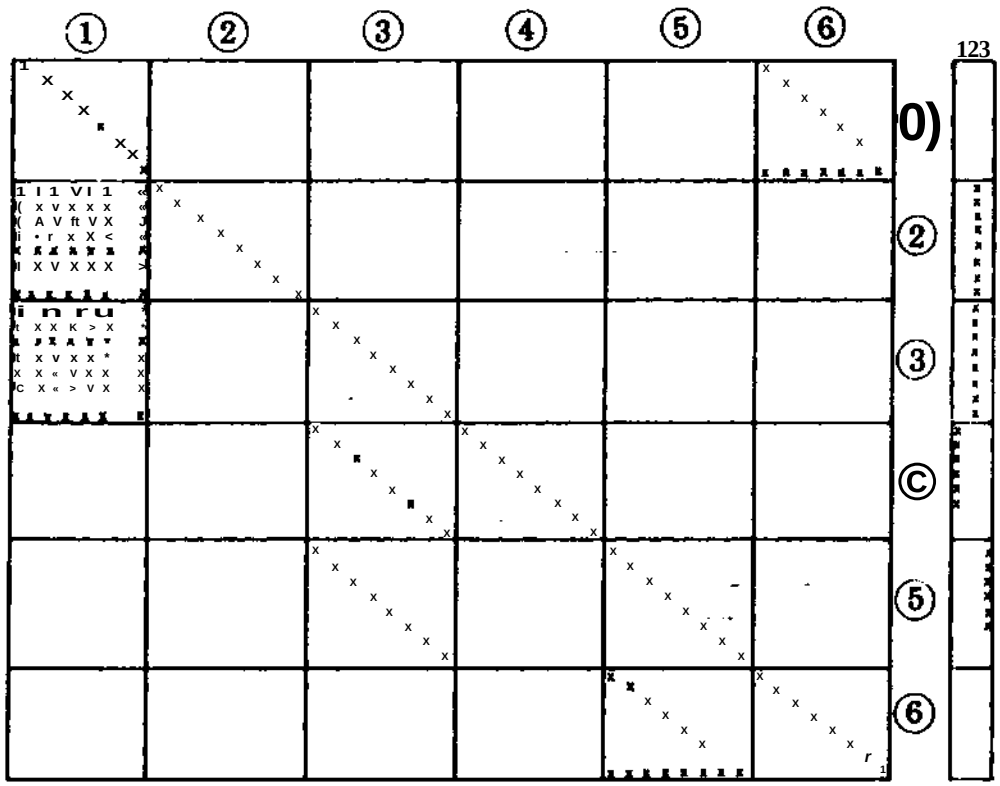
Max(e fe$_2$ 2e$_1$ +e + e$_4$ -e$_5$)

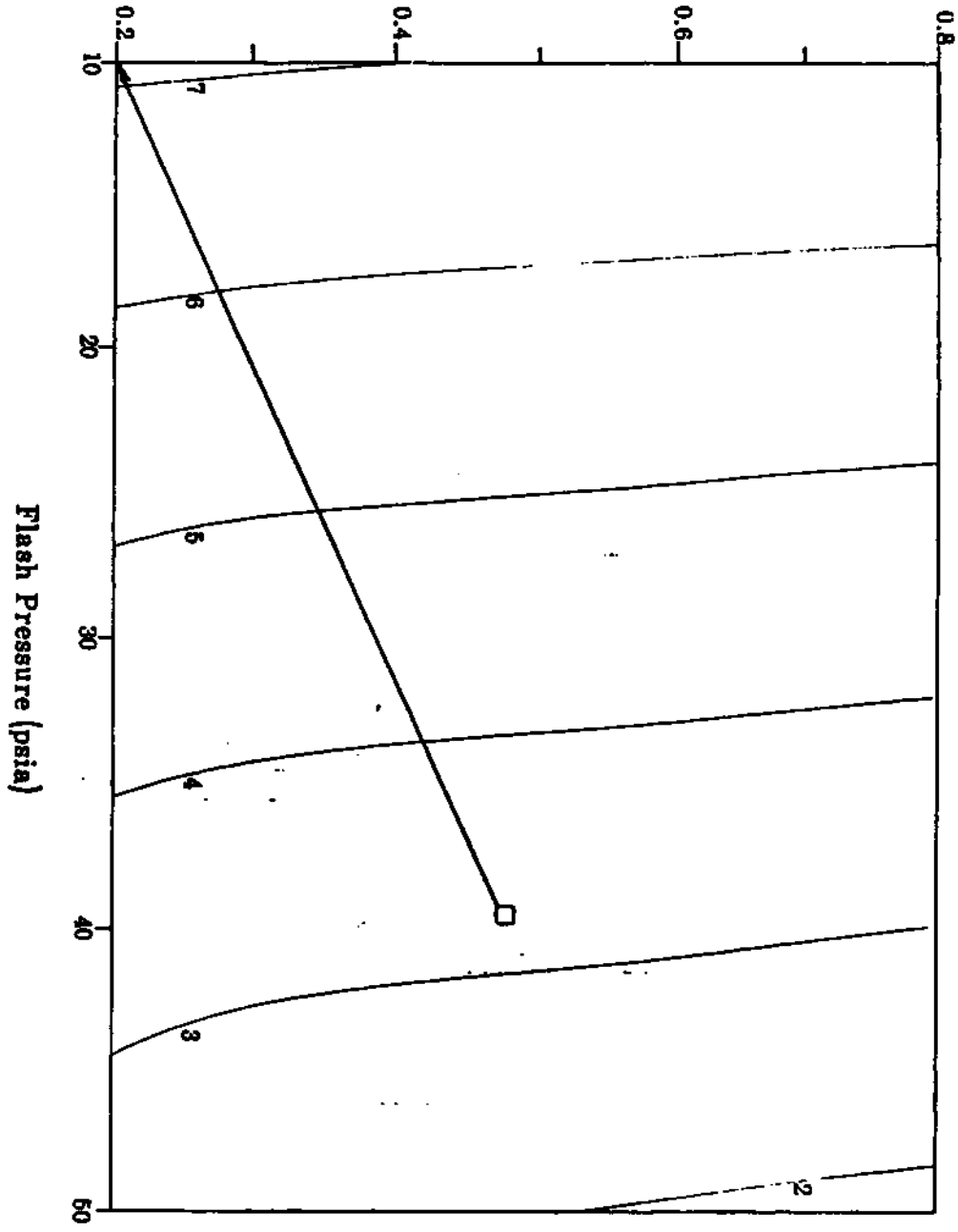
s. t. 0.2≤X!≤0.8

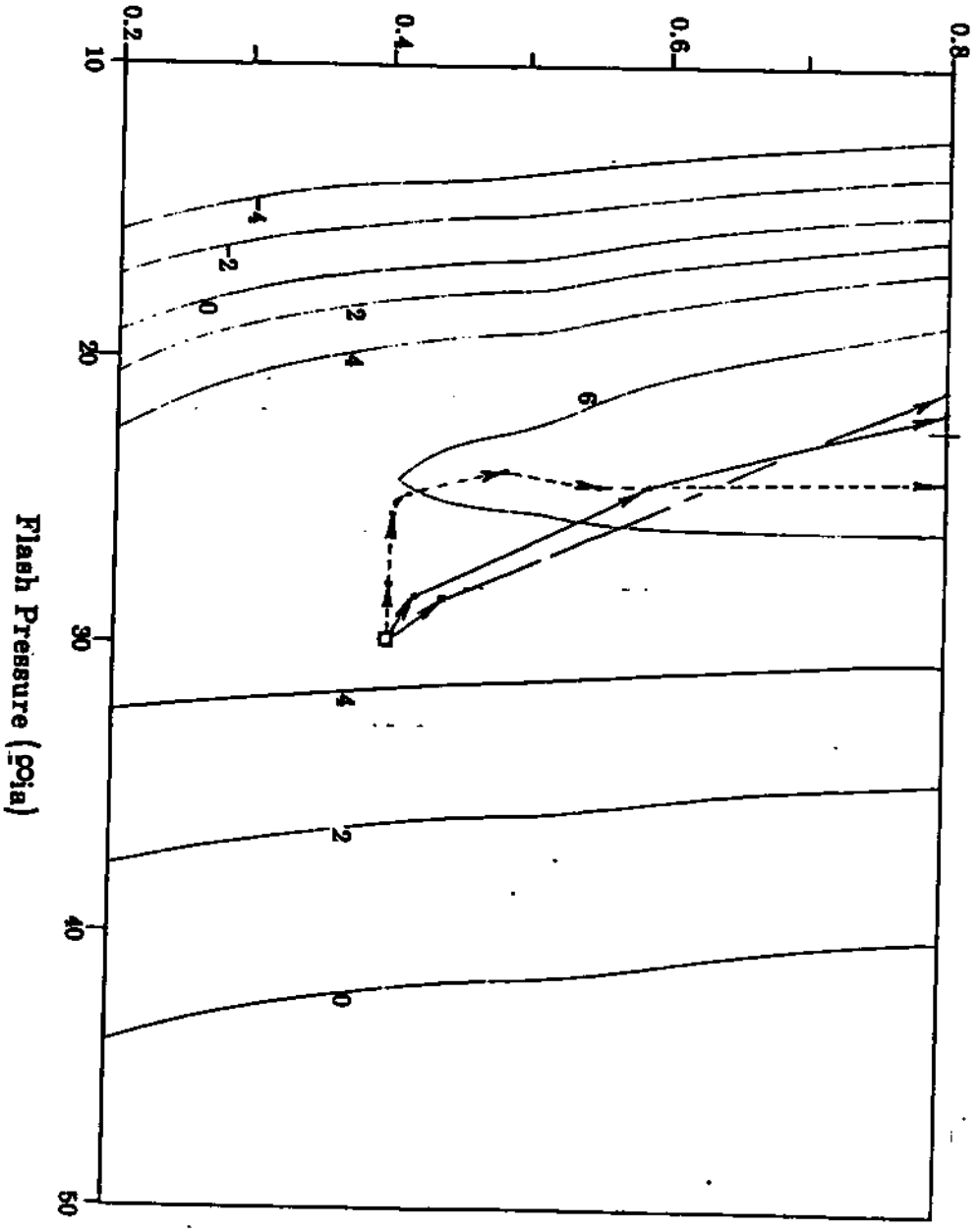
10.≤x_2≤50.

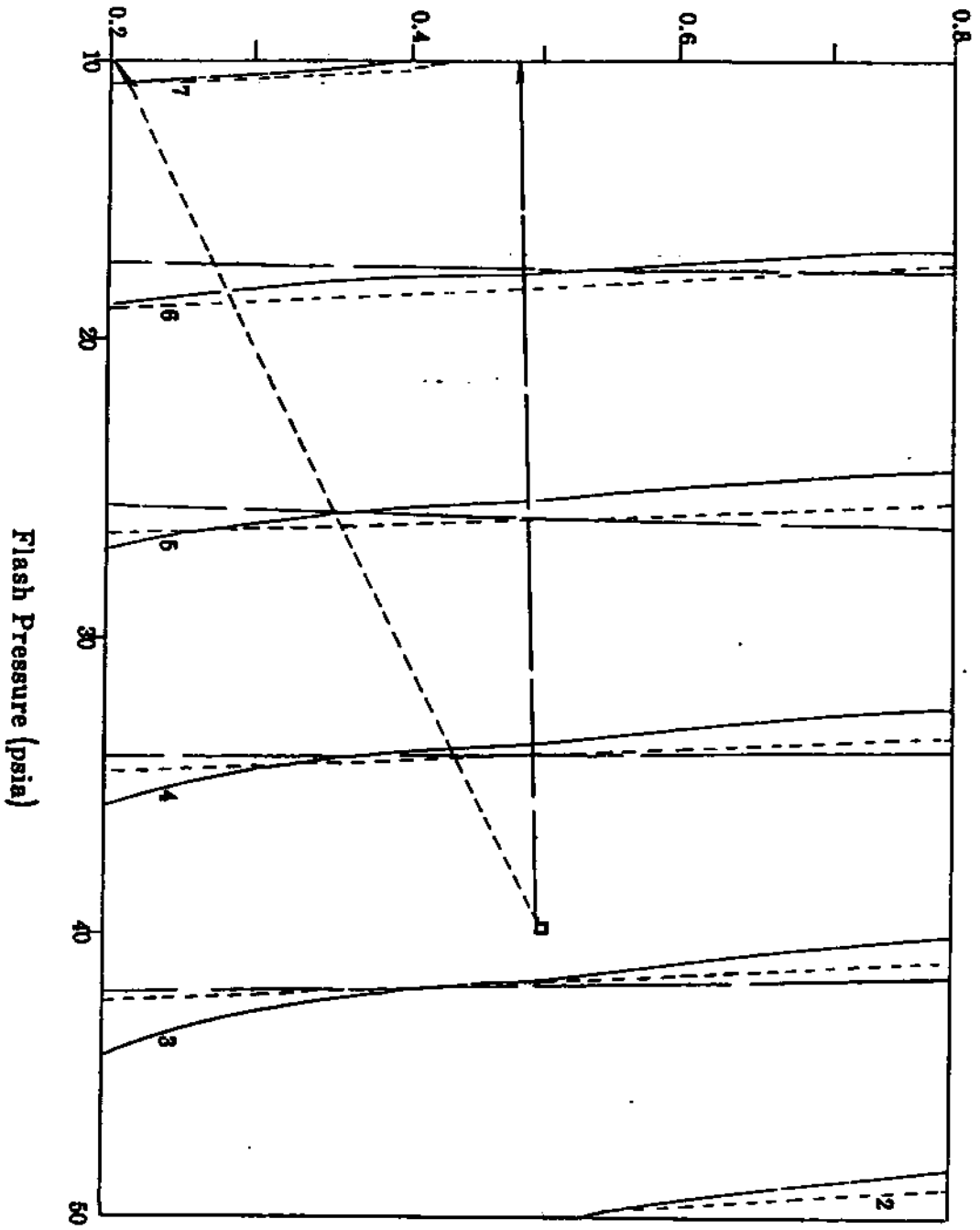
x_1+ x_3= 1.0

*for Q/LAP - Recycle convergence algorithm
for RFV - Optimization algorithm

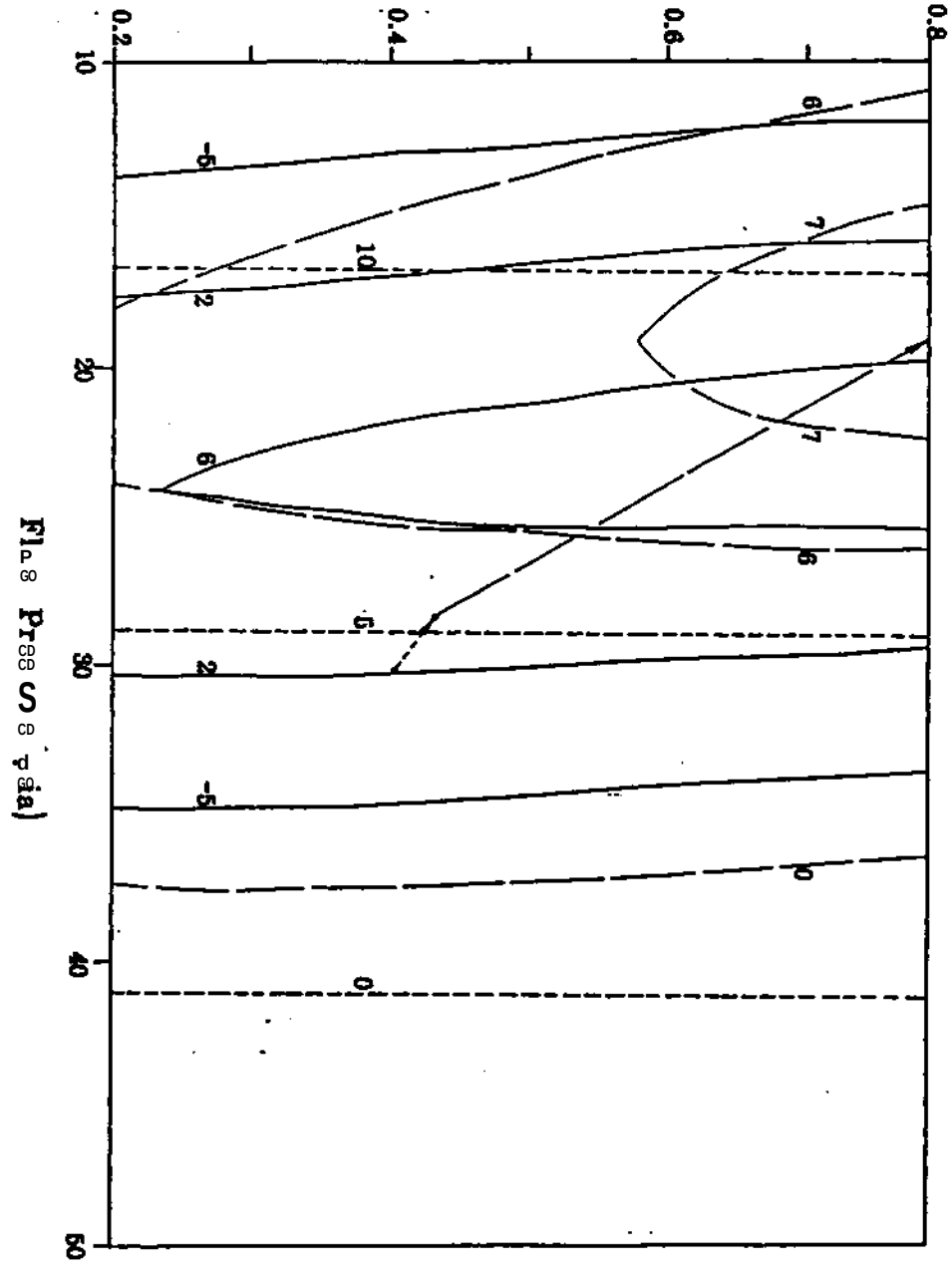

①
②
③
④
⑤
⑥
123
0)
②
③
©
⑤
⑥

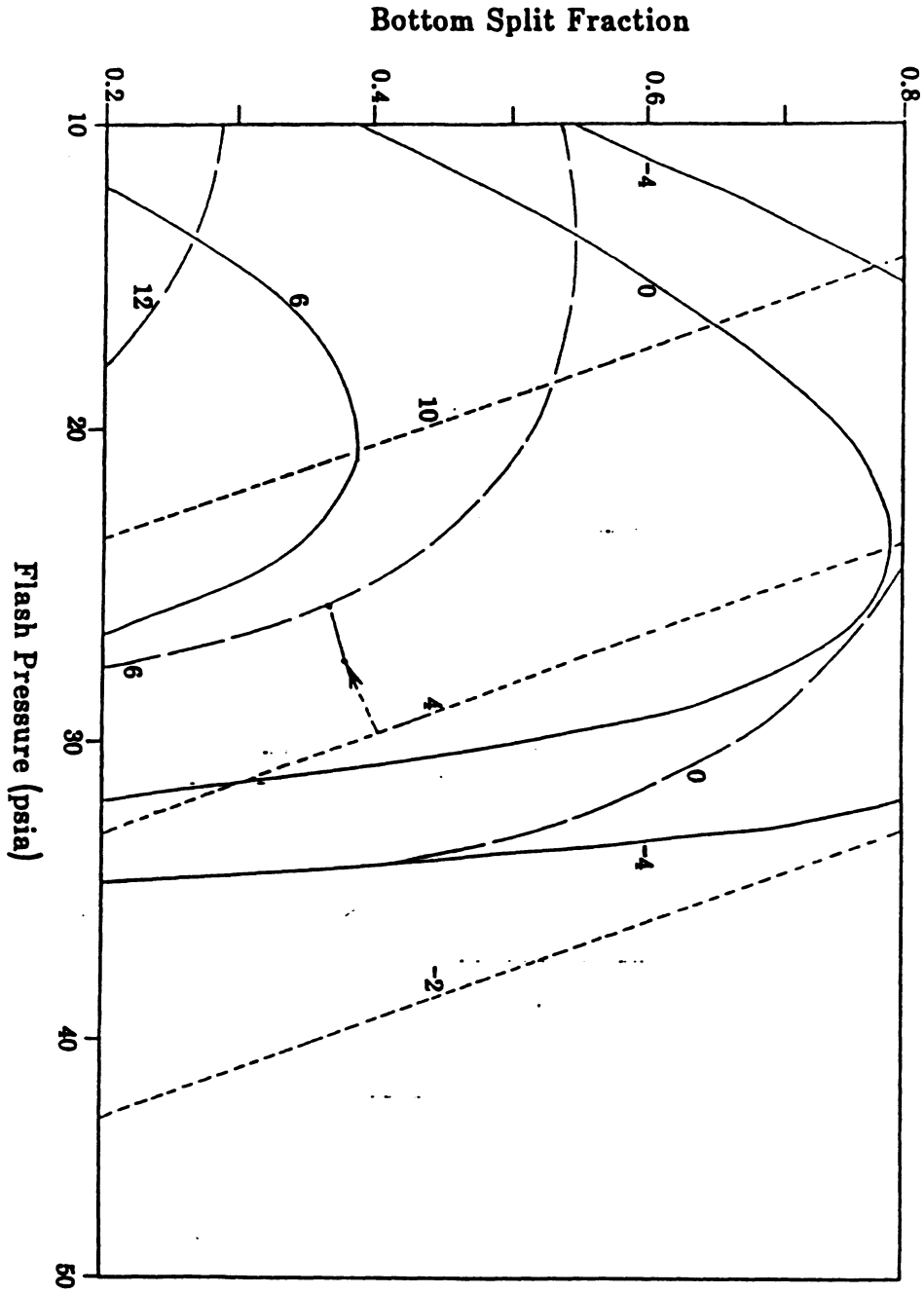

Bottom Split Fraction
0.2
0.4
0.6
0.8
10
20
30
40
50
Flash Pressure (psia)
7
6
5
4
3
2

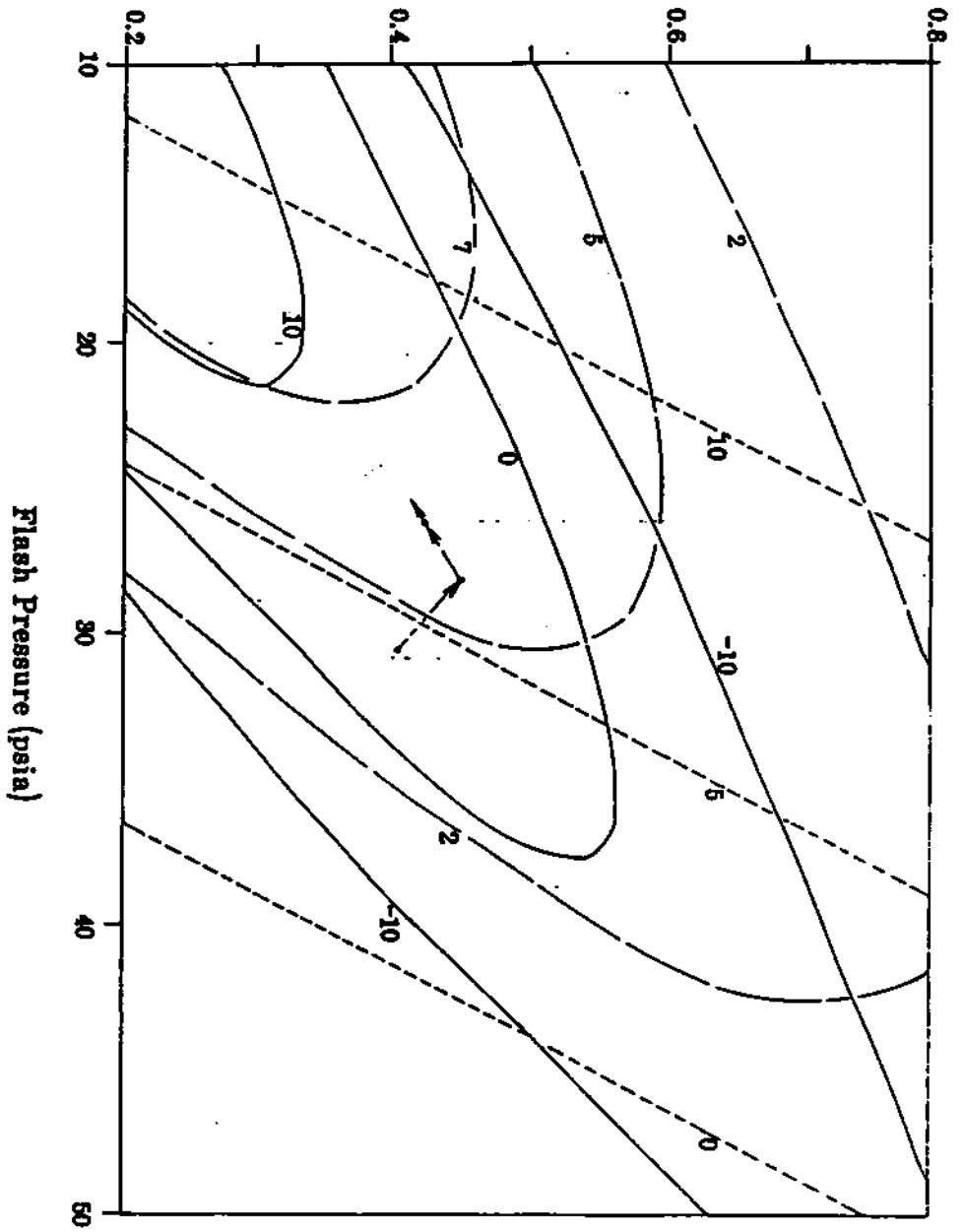

Bottom Split Fraction
0.2
0.4
0.6
0.8
10
20
30
40
50
Flash Pressure (psia)
-4
-2
0
2
4
6
4
2
0


Bottom Split Fraction
0.2
0.4
0.6
0.8
10
20
30
40
50
Flash Pressure (psia)
7
6
5
4
3
2

Bottom Split Fraction




Bottom Split Fraction
0.2
0.4
0.6
0.8
Flash Pressure (psia)
10
20
30
40
50
12
6
10
6
4
0
-4
-4
0
-2


Bottom Split Fraction
0.2
0.4
0.6
0.8
10
20
30
40
50
Flash Pressure (psia)
10
7
5
2
0
-10
5
2
-10
0