FUNCTION MINIMIZATION AND AUTOMATIC THERAPEUTIC CONTROL

BY

LINDA KAUFMAN

STAN-CS-228-71

JULY, 1971

COMPUTER SCIENCE DEPARTMENT

School of Humanities and Sciences

STANFORD UNIVERSITY

FUNCTION MINIMIZATION AND AUTOMATIC THERAPEUTIC CONTROL

Linda Kaufman
Computer Science 293
Professor Dorr
April 1971

Acknowledgments

I would like to thank Chris Schade for posing the problem and for explaining to me various biological aspects of the problem . I am indebted to Professor F.W.Dorr for his critical comments on the manuscript and to Michael Saunders, Stanley Eisenstat, and Richard Underwood for many interesting discussions and suggestions. I also wish to thank Laura Staggers for typing this paper.

1. Introduction

The problem considered in this paper was suggested by Cristy Schade [7] of the Department of Electrical Engineering at Stanford University. He is designing a computer system to assist in the care of patients who have suffered from cardiogenic shock. The treatment of such patients involves the administration of a vasoconstrictor drug over a long period of time, and it is rarely practical to always have a trained physician nearby adjusting the flow rate of the drug. Schade envisions his computer system as controlling the situation when a physician is unavailable. The computer would monitor the patient's blood pressure (and perhaps other aspects of his condition), update a mathematical model to correspond to the gathered information, and use the new model to determine the flow rate of the drug to get a desired blood pressure.

Schade has developed a system which treats a dog that has been injected with a drug that disables his natural blood pressure regulating system. A computer program has been written for the HP2116B computer which monitors the dog's blood pressure and adjusts the apparatus administering the vasoconstrictor drug to the dog. However, the algorithm employed to determine the successive flow rates does not always determine the optimal set of flow rates. By an optimal set we mean a set of flow rates over t intervals of time which will produce a set of blood pressures $\{y_j\}_{j=1}^{t}$ which minimizes

$$\sum_{j=1}^{t} (y_j - r_j)^2.$$

Here y_j is the blood pressure at time τ_j and r_j is the desired blood pressure at time τ_j set by the operator of the system. This paper develops an algorithm to determine the optimal set of flow rates.

Schade's mathematical model is based on a transversal filter system that weights the drug rates in the last n intervals of time to correspond to the patient's behavior. Symbolically, it is given by

$$Yj = \sum_{k=1}^{n} w_k x_{j-k+1} + w_0,$$

where

x_j = drug rate at time t_j,

y_j = blood pressure at time τ_j,

$\{w_k\}_{k=1}^{n}$ is a set of weights.

In matrix form the model can be written:

$$\begin{bmatrix} y_{j+1} \\ y_{j+n} \\ \\ y_{j+t} \end{bmatrix} = \begin{bmatrix} w_0 \\ w_0 \\ \\ w_0 \end{bmatrix} + \begin{bmatrix} w_n \cdots w_1 & & \\ & w_n \cdots w_1 & 0 \\ & & \\ 0 & & w_n \cdots w_1 \end{bmatrix} \begin{bmatrix} x_{j+2-n} \\ x_{j+1} \\ \\ x_{j+t+1-n} \\ x_{j+t} \end{bmatrix}$$

In practice, t is some positive integral multiple of n.

Since at time τ_{j+1} the values $x_{j-n+2}, \ldots, x_j$ are known, we can rewrite this system as $\underset{\sim}{y} = \bar{W} \underset{\sim}{x} + \underset{\sim}{h}$, where

$$h_\ell = \begin{cases} \sum_{k=\ell}^{n-1} w_{k+1} x_{j-k+\ell} + w_0, & \ell < n \\ w_0, & \ell \geq n \end{cases}$$

and

$$\bar{w}_{ij} = \begin{cases} 0 \text{ for } j > i \\ 0 \text{ for } i > n,\ j < i-n \\ w_{i+1-j} \text{ elsewhere.} \end{cases}$$

Thus, $\bar{W}$ is in lower triangular form.

Given this model, we can easily determine the future values of the blood pressure, y, for a given set of drug rates. Conversely, if we want the blood pressure to attain a certain level, using a back substitution process we can determine the required drug rate schedule/of the drug for the next t intervals of time. However, we must take several facts into consideration: i.e., flow rate

1. A negative amount of drug cannot be administered.

2. The apparatus does not allow a drug rate of more than 50 drops a minute.

3. The patient will die if his blood pressure exceeds certain bounds.

The last constraint is the one least often encountered but is, of course, the most important.

From this information one can formulate the following mathematical problem: Given r, $\bar{W}$, $\underset{\sim}{a}$, $\underset{\sim}{b}$, $\underset{\sim}{c}$, $\underset{\sim}{d}$, find x to

$$\text{minimize} \quad \sum_{i=1}^{t} (y_i - r_i)^2,$$

where

$$\underset{\sim}{y} = \bar{W}\underset{\sim}{x} + \underset{\sim}{h} ,$$

$$a_i < x_i < b_i,$$

$$c_i \leq y_i \leq d_i,$$

and r_i is the desired blood pressure at time τ_i. Usually the desired blood pressure is set to some goal so that actually r_i is the same for all i.

The problem outlined above is a quadratic programming problem that may be solved in several ways; however, when choosing a method we must observe the following criteria:

1) The method must be fast for systems whose number of variables is greater than 20, and whose constraints are four times the number of variables.

2) This particular problem constitutes only a small part of the total system. It is considered as a background job to which unused time cycles are allotted. Thus, it is unknown whether the procedure will actually be given sufficient time to complete the computation. Hence the method should be iterative, and all iterates should be feasible, i.e. they should satisfy the constraints.

3) Since the procedure is needed only when constraints are active, i.e., when x must satisfy some equality constraint, the method should work well in this situation.

The problem was approached in two ways, differing more in the actual formulation of the problem than in the algorithms involved.

In the first case, a method proposed by Goldfarb [1] based on Davidon's variable metric method for unconstrained optimization was applied to the problem of finding x to minimize

$$f(x) = \frac{1}{2} (\bar{W}\underset{\sim}{x} + \underset{\sim}{h}-\underset{\sim}{r})^T(\bar{W}\underset{\sim}{x} + \underset{\sim}{h}-\underset{\sim}{r})$$

where

$$\underset{\sim}{a} \leq \underset{\sim}{x} \leq \underset{\sim}{b}$$

and

$$c - \underset{\sim}{h} \leq \bar{W}\underset{\sim}{x} \leq \underset{\sim}{d} - h.$$

(A discussion of the algorithm and the computational experience appears in Section 3.)

However, it was discovered that a better approach was to set $z = \bar{W}\underset{\sim}{x} + \underset{\sim}{h}-\underset{\sim}{r}$, and consider the problem of determining $\underset{\sim}{z}$ to minimize $\frac{1}{2}\ \underset{\sim}{z}^{T}\underset{\sim}{z}$, where

$$\underset{\sim}{c} - \underset{\sim}{r} \leq \underset{\sim}{z} \leq \underset{\sim}{d} - \underset{\sim}{r}$$

and

$$\underset{\sim}{a} + \bar{W}^{-1}(\underset{\sim}{h}-\underset{\sim}{r}) \leq \bar{W}^{-1}\underset{\sim}{z} \leq \underset{\sim}{b} + \bar{W}^{-1}(\underset{\sim}{h}-\underset{\sim}{r}).$$

Section 4 contains a discussion of this approach. The method used is similar to Goldfarb's, but less complex. In fact, the ideas motivating both are the same, and it is these ideas which will be developed in the next chapter.

2. Basis of the Algorithms

Assume that we want to minimize a convex function

$$f(x) = f_0 + \underset{\sim}{a}^T \underset{\sim}{x} + \frac{1}{2} \underset{\sim}{x}^T G x, \tag{1}$$

where x is of dimension t and must lie in the domain D defined by

$$\underset{\sim i}{n}^T \underset{\sim}{x} \geq \ell_i, \quad i = 1,2,\ldots,m,$$

where $\underset{\sim i}{n}^T \underset{\sim i}{n} = 1$. Note that f is a convex function if G is a positive definite matrix.

Assume that prior calculations have indicated that the solution actually lies in the hyperplanes defined by

$$\underset{\sim l}{n}_i^T \underset{\sim}{x} = \ell_i, \quad i = 1,2,\ldots,q, \quad \text{where } 0 \leq q < m.$$

Let M denote the flat which is the intersection of these hyperplanes. If $g = \text{grad}\,(f(x))$, then by the mean value theorem, for any $\underset{\sim i}{x}$, $\underset{\sim i+1}{x}$,

$$\underset{\sim}{g}(\underset{\sim i+1}{x}) - \underset{\sim}{g}(\underset{\sim i}{x}) = G(\underset{\sim i+1}{x} - \underset{\sim i}{x}). \tag{2}$$

If we want $f(\underset{\sim i+1}{x})$ to be the global minimum of f where $\underset{\sim i+1}{x}$ is in M, then $g(\underset{\sim i+1}{x})$ must be orthogonal to M, i.e., $\underset{\sim}{g}(\underset{\sim i+1}{x})$ must be a linear combination of the unit normals to the hyperplanes whose intersection is M. Thus, if the columns of N are those normals, then

$$\underset{\sim}{g}(\underset{\sim i+1}{x}) = N \underset{\sim}{\alpha}, \tag{3}$$

and hence by substituting (3) into (2), we get

$$G^{-1}(N\,\underset{\sim}{\alpha} - \underset{\sim}{g}(\underset{\sim}{x}_i)) = \underset{\sim}{x}_{i+1} - \underset{\sim}{x}_i. \qquad (4)$$

Since $\underset{\sim}{x}_{i+1} - \underset{\sim}{x}_i$ is parallel to M, we must have

$$0 = N\cdot(\underset{\sim}{x}_{i+1} - \underset{\sim}{x}_i)$$

or, from (2),

$$0 = N^T G^{-1}(N\,o! - \underset{\sim}{g}(\underset{\sim}{x}_i)).$$

This implies

$$\alpha = (N^T G^{-1} N)^{-1}\,(N^T G^{-1}\,\underset{\sim}{g}(\underset{\sim}{x}_i)).$$

Substituting this back into (4) we get

$$\underset{\sim}{x}_{i+1} = \underset{\sim}{x}_i + (\,G^{-1}N(\,N^T G^{-1} N)^{-1}\,N^T G^{-1} - G^{-1})\,\underset{\sim}{g}(\underset{\sim}{x}_i)$$

or

$$\underset{\sim}{x}_{i+1} = \underset{\sim}{x}_i - \hat{P}\,G^{-1}\,\underset{\sim}{g}(\underset{\sim}{x}_i) \qquad (5)$$

where

$$\hat{P} = I - G^{-1}N(\,N^T G^{-1} N)^{-1} N^T.$$

Thus far we have assumed that the global minimum of f, where $\underset{\sim}{x}$ is constrained to D, corresponds to the minimum of f in M. However, we can distinguish two instances where this is not so:

Case 1: There is no guarantee that the point $\underset{\sim}{x}_{i+1}$ does not violate some other constraint of the problem. Hence, it would be preferable to write

$$\underset{\sim}{x}_{i+1} = \underset{\sim}{x}_i - \lambda\,\underset{\sim}{s}_i,$$

where $\underset{\sim}{s}_i = \hat{P}\,G^{-1}\,\underset{\sim}{g}(\underset{\sim}{x}_i)$ and $0 \leq \lambda \leq 1$. If $\bar{\lambda}$ is the minimum distance that can be traveled in direction $\underset{\sim}{s}$ before one of the inequality constraints becomes an equality constraint, then $\lambda = \min(1,\bar{\lambda})$. If $\lambda = 1$, then the minimum of f along $\underset{\sim}{s}_i$ has been found and hence the minimum in the corresponding hyperspace has been located. If this is not the case, then the barrier 'hit' must be added as a new constraint, and a new direction must be determined.

The value of $\bar{\lambda}$ can easily be calculated. Assume that the point $\underset{\sim}{x}_i$ lies in the interior of a region

$$\underset{\sim}{n}_j^T \underset{\sim}{x}_i > \ell_j, \qquad j = q+1,\ldots,m.$$

If $\underset{\sim}{x}_{i+1}$ is to lie in the closure of this domain, then

$$\underset{\sim}{n}_j^T (\underset{\sim}{x}_i - \bar{\lambda}\, \underset{\sim}{s}_i) \geq \ell_j, \qquad j = q+1,\ldots,m,$$

which implies

$$-\bar{\lambda}\, \underset{\sim}{n}_j^T \underset{\sim}{s}_i \geq \ell_j - \underset{\sim}{n}_j^T \underset{\sim}{x}_i, \quad j = q+1,\ldots,m,$$

$$\bar{\lambda}\, \underset{\sim}{n}_j^T \underset{\sim}{s}_i \leq \underset{\sim}{n}_j^T \underset{\sim}{x}_i - \ell_j, \quad j = q+1,\ldots,m,$$

$$\bar{\lambda} \leq \frac{\underset{\sim}{n}_j^T \underset{\sim}{x}_i - \ell_j}{\underset{\sim}{n}_j^T \underset{\sim}{s}_i}, \quad j = q+1,\ldots,m.$$

If we set

$$\lambda_j = \frac{\underset{\sim}{n}_j^T \underset{\sim}{x}_j - \ell_j}{\underset{\sim}{n}_j^T \underset{\sim}{s}_i}, \quad j = q+1,\ldots,m,$$

then

$$\bar{\lambda} = \min_{q+1 \leq j \leq m} \{\lambda_j \mid \lambda_j \geq 0\}.$$

Case 2: The global minimum for f where $x \in D$ actually lies in some flat M' where MCM' and $M \neq M'$, i.e., we have too many 'active' constraints. To check whether this is the case, we consider

$$u = (N^T N)^{-1} N \cdot \underset{\sim}{g}(\underset{\sim}{x}_{i+1}). \tag{6}$$

A necessary and sufficient condition that $f(\underset{\sim}{x}_{i+1})$ be a global minimum

is that $\|\hat{PG}^{-1}\underset{\sim}{g}(\underset{\sim}{x}_{i+1})\| = 0$ and $\underset{\sim}{u} \geq \underset{\sim}{\theta}$.

For a complete proof of this last statement see [1]. An intuitive idea of the proof is given below.

Obviously, if $\|\hat{PG}^{-1}\underset{\sim}{g}(\underset{\sim}{x}_{i+1})\| = 0$, then $f(\underset{\sim}{x}_{i+1})$ is the minimum of f for $x \in M$. If $\underset{\sim}{x}_{i+1}$ is not the global minimum, then we can proceed in the direction $-\underset{\sim}{g}(\underset{\sim}{x}_{i+1})$ to find a point $\bar{x}$ in D such that $f(\bar{x}) < f(\underset{\sim}{x}_{i+1})$. Since $\underset{\sim}{g}(\underset{\sim}{x}_{i+1}) = N\alpha$, we have

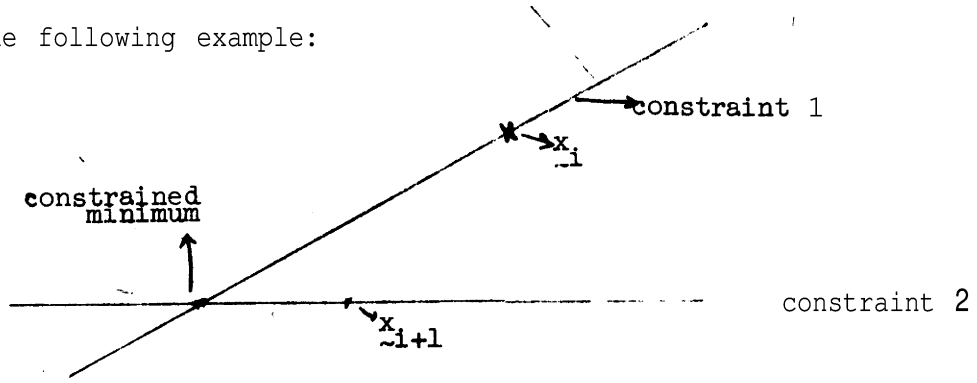
$$(N^T N)^{-1} N^T \underset{\sim}{g}(\underset{\sim}{x}_{i+1}) = (N^T N)^{-1} (N^T N)\alpha$$

$$= \underset{\sim}{\alpha},$$

and thus $u = \alpha$. Since the unit normals to the hyperplanes which constrain x point towards the interior of the region, a point $\bar{\underset{\sim}{x}}$ can be found such that $f(\bar{\underset{\sim}{x}}) < f(\underset{\sim}{x}_{i+1})$ if and only if at least one of the components of $\underset{\sim}{u}$ is negative.

Because of the above it is obvious that once $\underset{\sim}{x}_{i+1}$ has been computed we should check if $u_m \geq 0$. If this is not the case, the constraint corresponding to the most negative component of u should be dropped. Since there are only a finite number of constraints, this process should terminate in a finite number of steps. However, dropping constraints only after the minimum of f in M has been found might not be the best strategy. It might be preferable to drop a constraint whenever we find that $\underset{\sim}{u}$ has a negative component. In this case, if $\underset{\sim}{x}_{i+1}$ has been reached without adding any new constraints, then $f(\underset{\sim}{x}_{i+1})$ must be the minimum value of f(x) for $x \in D$. Moreover, if r constraints are dropped before a move, it is likely that r-1 repetitions. of the algorithm have been saved. Thus, it seems that we should

drop a hyperplane at the first opportunity. Unfortunately, there is a big disadvantage to this strategy. Although the dropped constraint cannot be picked up immediately, it is possible that after another constraint has been added, the dropped constraint might have to be reactivated. This is clear from the following example:



s-constrained minimum

In the above illustration, if constraint 1 is not deactivated, the minimum can be found in one step; otherwise two steps have to be taken.

Thus, we need a strategy guaranteeing that the function will be less if the constraint is dropped than if it is not. There is a w&l-known estimate (see [6]) that if

$$\|\hat{P}\, G^{-1}\, \underset{\sim}{g}\|_2 < \frac{1}{2}\, u_i\, \beta^{-1/2}, \tag{7}$$

where

$$\beta \geq \|(N^T N)^{-1}\|_2 \text{ and } u_i = \min \{u_j \mid u_j < 0\},$$

then the i^{th} hyperplane should be dropped.

3. Goldfarb's Method

In our problem the function f, mentioned in section 2, is

$$f(x) = \frac{1}{2}(\bar{W}x + \underset{\sim}{h}-\underset{\sim}{r})^T (\bar{W}x + \underset{\sim}{h}-\underset{\sim}{r}) .$$

Since $\bar{W}$ is lower triangular and w_1 is positive, $f(x)$ is certainly convex. Its gradient is. given by

$$\underset{\sim}{g}(\underset{\sim}{x}) = \bar{W}^T(\underset{\sim}{h}-\underset{\sim}{r} + \bar{W}\underset{\sim}{x}),$$

and its inverse Hessian is $G^{-1} = \bar{W}^{-1} \bar{W}^{-T}$. According to the theory developed in section 2, if we are at a point $\underset{\sim}{x}_i$ and the columns of the matrix N are the unit normals to the hyperplanes whose intersection defines a flat M, then the minimum of f in M is found at

$$\underset{\sim}{x}_{i+1} = \underset{\sim}{x}_i - \hat{P} G^{-1}\underset{\sim}{g}(\underset{\sim}{x}_i),$$

where

$$\hat{P} = I - G^{-1} N(N^T G^{-1}N)^{-1}N.$$

In our case it is quite costly to actually compute $-\hat{P} G^{-1}$, and it might be more practical to approximate it by some matrix H and set $\underset{\sim}{x}_{i+1}= \underset{\sim}{x}_i + H\cdot\underset{\sim}{g}(\underset{\sim}{x}_i)$. Donald Goldfarb in [2] presents an algorithm incorporating this idea. For a quadratic objective function in a system of dimension t, if the same q constraints are active for t-q iterations, his H is equal to $-\hat{P} G^{-1}$. His algorithm is analogous to Davidon's variable metric algorithm when no constraints are active.

The heart of Goldfarb's algorithm is his method for updating H. Let H_q represent the matrix H when q constraints are active. If we want to eliminate the q^{th} constraint from H, then

$$H_{q-1} = H_q + \frac{P_{q-1} \underset{\sim}{n}_q \underset{\sim}{n}_q^T P_{q-1}}{\underset{\sim}{n}_q^T P_{q-1} \underset{\sim}{n}_q}, \tag{8}$$

where $P_{q-1} = I - N_{q-1}(N_{q-1}^T N_{q-1})^{-1} N_{q-1}^T$. If we want. to add a $q+1^{st}$ constraint to the intersection, then

$$H_{q+1} = H_q - \frac{H_q \underset{\sim}{n}_{q+1} \underset{\sim}{n}_{q+1}^T H_q}{\underset{\sim}{n}_{q+1}^T H_q \underset{\sim}{n}_{q+1}}. \tag{9}$$

If the basis remains unchanged, then set $H_q = H_q + A + B$, where

$$A = - \frac{\underset{\sim}{\sigma}_i \underset{\sim}{\sigma}_i^T}{\underset{\sim}{\sigma}_i^T \underset{\sim}{y}_i} \tag{10}$$

$$B = - \frac{H_q \underset{\sim}{y}_i \underset{\sim}{y}_i^T H_q}{\underset{\sim}{y}_i^T H_q \underset{\sim}{y}_i}$$

where

$$\underset{\sim}{\sigma}_i = \lambda \underset{\sim}{s}_i = \underset{\sim}{x}_{i+1} - \underset{\sim}{x}_i$$

and

$$\underset{\sim}{y}_i = \underset{\sim}{g}(\underset{\sim}{x}_{i+1}) - \underset{\sim}{g}(\underset{\sim}{x}_i).$$

The chief features of (8) - (10) are:

(a) if $H_q = - \hat{P}_q G^{-1}$, then H_{q+1} given by (9) is equal to $-\hat{P}_{q+1} G^{-1}$;

(b) the term A in (10) insures that $H_q = -\hat{P}_q G^{-1}$ after t-q steps if $f(\underset{\sim}{x})$ is quadratic;

(c) the term B insures that mutually conjugate directions are searched, i.e.,

$$\underset{\sim}{s}_i^T \underset{-3}{G} \underset{\sim}{s}_j = 0, \; i \neq j, \text{ where } \underset{\sim}{s}_i = H_q \underset{\sim}{g}(\underset{\sim}{x}_i);$$

(d) The objective function will always be decreased at each iteration;

(e) H is a positive semi-definite matrix. Moreover, if $v = N_{q}\alpha$, then $v^T H v = 0$;

(f) if $H_q = \hat{P}_q G^{-1}$, then H_{q-1} given by (8) is not necessarily equal to $-\hat{P}_{q-1} G^{-1}$.

3.1 Observations

When none of the blood pressure constraints is active, which is true most of the time, each row in N^T has only one non-zero element and $(N^T N)^{-1} = I$. This means that the rows of H corresponding to active constraints on $\underset{\sim}{x}$ are zero. When the i^{th} constraint is dropped, the update of H according to equation (8) requires only adding 1 to h(i,i). Further, when only drug constraints are active, the vector $u = (N^T N)^{-1} N^T g$ (refer to section 2) is easy to obtain once g has been computed, and the quantity β in equation (7) is just 1. Thus, we can use a strategy for dropping constraints which assures us that we will not be deactivating a constraint which might have to be reactivated in a short time.

If $H = -\hat{P} G^{-1}$ and $s = H\underset{\sim}{g}(\underset{\sim}{x}_i)$, then the minimum of f along s is found at $\underset{\sim}{x}_{i+1} = \underset{\sim}{x}_i + s$. However, if H only approximates $\hat{P} G^{-1}$, then it isnecessary to compute λ such that $\underset{\sim}{x}_{i+1} = \underset{\sim}{x}_i + \lambda\underset{\sim}{s}$ minimizes f along s.

Since

$$f(x + \lambda s) = f(x) + \lambda \underset{\sim}{g}^T \underset{\sim}{s} + \frac{\lambda^2}{2} \underset{\sim}{s}^T G \underset{\sim}{s},$$

the minimum of f along s occurs when

$$\lambda = \frac{\underset{\sim}{g}^T \underset{\sim}{s}}{\underset{\sim}{s}^T G \underset{\sim}{s}} = \frac{\underset{\sim}{g}^T \underset{\sim}{s}}{(\bar{W}s)^T (\bar{W}s)} .$$

Since the computation of $\bar{W}s$ is required anyway to determine if any of the y constraints are active, the number of steps required to determine the minimum of f along s is effectively negligible.

The main disadvantage of the algorithm is that whenever a y (blood pressure) constraint is active, we must update $(N^TN)^{-1}$. Goldfarb [1] gives recursive relations for updating $(N^TN)^{-1}$ which require approximately $\frac{3}{2}q(q+t)$ operations every time a new column is added to N. We can do better by noting that N can be written as $N = QR$, where $Q^TQ = I$ and R is upper triangular (see (4)). In this case, $(N^TN)^{-1} = R^{-1}R^{-T}$. Adding a new column to R, as will be described in section 4.2, requires approximately $\frac{q}{2}(q+t)$ operations. Even this estimate is high when we consider the fact that, when blood pressure constraints are active, many of the other constraints will be drug rate constraints. Admittedly, in our formulation where the model is linear, the y constraint is rarely active, but when it is a patient's life is probably in peril.

3.2 The Matrix H

Probably the largest problems in the implementation of the whole method are the initialization and form of H_q. Several alternatives present themselves, none of which appears completely satisfactory.

Alternative 1: Since $H_0 = \bar{W}^{-1}\bar{W}^{-T}$, H_q can be initialized by determining the matrix $\bar{W}^{-1}$, forming the product $\bar{W}^{-1}\bar{W}^{-T}$ and applying eq. (9) q times. Because of the form of $\bar{W}$ (i.e., triangular with elements the same on the subdiagonals and diagonals), only $t^2/2$ steps are required to form $\bar{W}^{-1}$. However, another $t^3/6$ are needed to form the product. If we then add another qt^2 steps to obtain H_q, using (9), we find that for a 20×20 system on the IBM 360/67 approximately 1/3 second has elapsed before the routine has even been entered. This is entirely impractical, considering the application of the procedure.

Alternative 2: The matrix H_q can also be written as $H_q = \bar{W}^{-1} \cdot M_q \bar{W}^{-T}$, where $M_0 = I$ and

$$M_{i+1} = M_i - \frac{(M_i \bar{W}^{-T} n_{i+1})(M_i \bar{W}^{-T} n_{i+1})^T}{(\bar{W}^{-T} n_{i+1})^T M_i (\bar{W}^{-T} n_{i+1})} . \qquad (11)$$

This last formula is derived from eq. (9).

To update the H_q matrix when the constraint matrix has not changed, we use the formula

$$M_{q+1} = M_q + \bar{A} + \bar{B}$$

where

$$\bar{A} = - \frac{(\lambda M_q \bar{W}^{-T} \underset{\sim}{g})(\lambda M_q \bar{W}^{-T} \underset{\sim}{g})^T}{\lambda \underset{\sim}{s}_i^T \underset{\sim}{y}_i}$$

and

$$\bar{B} = - \frac{(M_q \bar{W}^{-T} \underset{\sim}{y}_i)(M_q \bar{W}^{-T} \underset{\sim}{y}_i)^T}{\underset{\sim}{y}_i^T H_q \underset{\sim}{y}_i},$$

where

$$\underset{\sim}{y}_i = \underset{\sim}{g}(\underset{\sim}{x}_{i+1}) - \underset{\sim}{g}(\underset{\sim}{x}_i).$$

Since the gradient $\underset{\sim}{g}$ has the form $\underset{\sim}{g}(\underset{\sim}{x}) = \bar{W}^{-T} \underset{\sim}{v}$, where $\underset{\sim}{v} = (\bar{W} \underset{\sim}{x} + (\underset{\sim}{h} - \underset{\sim}{r}))$, the formula for $\bar{A}$ and $\bar{B}$ can be written even more succinctly as

$$\bar{A} = - \frac{\lambda (M_q \cdot \underset{\sim}{v}_i)(M_q \cdot \underset{\sim}{v}_i)^T}{(M_q \underset{\sim}{v}_i)^T \underset{\sim}{z}_i},$$

$$\bar{B} = - \frac{(M_q \underset{\sim}{z}_i)(M_q \underset{\sim}{z}_i)^T}{\underset{\sim}{v}_i^T \underset{\sim}{n}_q \underset{\sim}{v}_i},$$

where

$$\underset{\sim}{v}_i = (\bar{W} \underset{\sim}{x}_i + (\underset{\sim}{h} - \underset{\sim}{r}))$$

and

$$\underset{\sim}{z}_i = \underset{\sim}{v}_{i+1} - \underset{\sim}{v}_i.$$

To drop the q^{th} constraint we set

$$M_{q-1} = M_q + \frac{(\bar{W}P_{q-1}\,\underset{\sim}{n}_q)(\bar{W}P_{q-1}\,\underset{\sim}{n}_q)^T}{\underset{\sim}{n}_q^T P_{q-1}\,\underset{\sim}{n}_q} \quad . \tag{12}$$

This alternative is not as impractical as it might appear, for the following reasons:

(a) The function of the matrix H is to project $\underset{\sim}{g}(\underset{\sim}{x})$ into the correct flat. In the algorithm it is used only to determine the correct direction $\underset{\sim}{s} = H\underset{\sim}{g}(\underset{\sim}{x}_i)$. Since we know that $\underset{\sim}{g}(\underset{\sim}{x}_i)$ has the form $W^{-T}\underset{\sim}{v}_i$, the $\underset{\sim}{s}_i$ vector is simply $\underset{\sim}{s}_i = \bar{W}^{-1}M_q\,\underset{\sim}{v}_i$. Although we now need $t^2/2$ more operations to compute $\underset{\sim}{s}_i$, we have also saved $t^2/2$ operations since we never need to compute $\underset{\sim}{g}(\underset{\sim}{x}_i)$ explicitly. Moreover, since $M_q\,\underset{\sim}{v}_1$ is computed here, to update the formula for $\bar{A}$ requires fewer steps.

(b) If only the $\underset{\sim}{x}$ constraints are active, the vector $P_{q-1}\underset{\sim}{n}_q$ is a unit vector which has a '1' or a '-1' in its q^{th} component. Thus, no work is required to form $\bar{W}\,P_{q-1}\underset{\sim}{n}_q$. Further, because $\bar{W}$ is lower triangular, the first q-1 elements of $\bar{W}\,P_{q-1}\underset{\sim}{n}_q$ are zero. Therefore, to drop a constraint requires $(t-q)^2/2$ operations, which admittedly is more than in the previous alternative.

(c) If only the x constraints are active, the formula (11) for forming M_q initially can also be simplified. Since $\bar{W}^{-T}$ is an upper triangular matrix, if $\underset{\sim}{n}_i$ corresponds to constraining the i^{th} component of $\underset{\sim}{x}$, then the last (t-i+1) components of the vector $\bar{W}^{-T}\underset{\sim}{n}_i$ will be zero. If the constraints on $\underset{\sim}{x}$ are initially added so that the last nonzero element of

the i^{th} column of W^T is above the last nonzero element in the $i+1$ column of N^T, then after $i+1$ applications of (11) M_{i+1} will have the form

$$M_{i+1} = \begin{bmatrix} A & 0 \\ 0 & D \end{bmatrix}$$

where A is a $j \times j$ symmetric matrix, and D is a diagonal matrix of rank t-j and the $i+1^{th}$ constraint corresponds to constraining x_j. Accordingly, forming the M_q matrix requires fewer steps than one might have anticipated originally.

Calculating a vector $a = \bar{W}^{-1}b$ is equivalent to finding the vector $\underset{\sim}{a}$ such that $\bar{W}\underset{\sim}{a} = b$. In our problem if $t > n$, where n is the dimension of the filter system used to determine $\bar{W}$, $\bar{W}^{-1}$ is a full lower triangular matrix with the same elements on the subdiagonal. However, $\bar{W}$ has a triangle of size t-n in its lower corner that is zero. If b has just one nonzero component, which is sometimes true when eq. (11) is applied, then naturally we should use $\bar{W}^{-1}$. In other cases we can do backsubstitution to find a and save $\frac{(t-n)^2}{2}$ operations.

This formulation still suffers from the fact that approximately $\frac{3}{4}qt^2$ operations are required to form the original matrix H_q. It might seem better to start with no constraints active and encounter them one at a time, but computational experience shows that most of the first constraints activated are later deactivated, and that it doesn't pay to compute s and minimize f along s each time a constraint is made active.

Although the form of H just presented is not that useful for our particular problem, the idea might be worth considering for other problems whose object function has the form $(\underset{\sim}{Ax}-\underset{\sim}{b})^T \underset{A}{} (\underset{\sim}{Ax}-\underset{\sim}{b})$.

Alternative 3: If we set $H = P\, G^{-1}$, we have

$$H = [I - i^{-1}\, \bar{W}^{-T}(N^T \bar{W}^{-1}\bar{W}^{-T}N)^{-1}\, N^T]\, \bar{W}^{-1}\bar{W}^{-T}.$$

If $A = \bar{W}^{-T}N$, then this becomes

$$H = [I - \bar{W}^{-1}A(A^TA)^{-1}\, N^T]\, \bar{W}^{-1}\bar{W}^{-T}$$

$$= \bar{W}^{-1}[I - A(A^TA)^{-1}\, A^T]\, \bar{W}^{-T}.$$

If we start out with the exact H, and change H just by updating A, then H will always be equal to $-P\, G$ (barring roundoff). Thus the update formulas for H when the constraints are not changed will never have to be used. Since $\underset{\sim}{q}$ is of the form $\bar{W}^T \underset{\sim}{v}$, then $s = H \cdot \underset{\sim}{g}$ simplifies to

$$\underset{\sim}{s} = \bar{W}^{-1}[I - A(A^TA)^{-1}\, A^T]\underset{\sim}{v}.$$

This looks horrendous, but if we put $H = QR$, where $QQ^T = I$, then $R^TR = A^TA$ and computing $(A^TA)^{-1}v$ requires only q^2 operations. On the average the calculation of s takes about $t^2/2 + q(3/2t+q)$ operations. The update of A' involves only updating R and is not prohibitive. Actually, using this alternative is the same as using the method to be described in the next section, except that fewer steps are required there.

Alternative 4: Begin with $H = I$ and forget that the inverse Hessian is known. Applying eq. (9) q times when only x constraints are active results in the matrix $\bar{H}$ where

$$\bar{h}_{ij} = \begin{cases} 0 & \text{when } i \neq j \\ 0 & \text{where constraint on } x_i \text{ is active and } i = j \\ 1 & \text{otherwise.} \end{cases}$$

Consequently, the initialization procedure requires no computation, and movement toward the required minimum begins immediately. This approach seems best when there are many constraints active, since H will equal $\hat{P}_G^{-1}$ after t-q iterations. However, usually about t/3 constraints are active and progress is slow; it takes twice as long to get to any point as it does in the method described in the next section.

4. The Algorithm in the Transformed System

As mentioned in section 2, the problem can be reformulated by setting $z = \bar{W}x + h - r$. In this case we want to minimize

$$f(z) = \frac{1}{2}(z^T z)$$

where

$$\bar{a} \leq z \leq b \tag{13}$$

$$\bar{c} \leq \bar{W}^{-1} z \leq \bar{d} \tag{14}$$

Eq. (13) corresponds to the blood pressure constraints, and (14) corresponds to the drug rate constraints.

In this situation the gradient of f is given by

$$g(z) = z$$

and the inverse Hessian of f is given by

$$G^{-1}(z) = I,$$

where I is the identity matrix.

The matrix $\hat{P}$ described in section 2 is then given by

$$\hat{P} = I - N(N^T N)^{-1} N^T$$

where the columns of N contain the unit normals to the hyperplanes which constrain z to a flat M. Therefore, according to (5), the minimum of $f(z)$ for z constrained to M is given by

$$z_{i+1} = z_i - (I - N(N^T N)^{-1} N^T) z_i \tag{15}$$

If M is defined by the q relations

$$n_i^T z = A_i, \quad i = 1, \ldots, q$$

(assuming q constraints active), where ℓ_i is known for i = 1, . . . ,q,

eq. (15) can be rewritten as

$$z_{i+1} = z_i - s_i$$

where

$$s_i = z_i - N(N^TN)^{-1}\ell.$$

If $(N^TN)^{-1}$ can be computed easily, then a matrix H to approximate $-\hat{P}G^{-1}$ is unnecessary, Further, the minimum of f along s in M, which is the minimum of f in M, can be determined immediately.

The main disadvantage to reformulating the problem is that now, when only the drug rate constraints are active, the matrix N is more 'full' than in the previous situation. Also, $(N^TN)^{-1}$ is not the identity matrix and we cannot easily obtain the upper bound on $\|(N^TN)\|_2^{-1}$ that eq. (7) required for a good criterion for dropping a hyperplane. Because of this we decided to drop a hyperplane whenever $-g(x_i)$ pointed to the interior of the region. This strategy had the disadvantage that constraints were sometimes dropped and then reactivated.

4.1. Determining $(N^TN)^{-1}\ell$.

Assume N^T is q x t. Since N can be written as N = QR where $Q^TQ = QQ^T = I$ and R is a q x q upper triangular matrix, we can write $(N^TN) = R^TQ^TQR = R^TR$.

In our algorithm we must compute the vector $u=(N^TN)^{-1}\ell$. This is-equivalent to determining u such that

$$(N^TN)u = \ell \quad \text{or} \quad R^TRu = \ell.$$

Since R is an upper triangular matrix, u can be determined using two backsolve operations; i.e.,

$$R^T m = \ell,$$

and

$$Ru = m.$$

Because each backsolve requires about $q^2/2$ steps, we spend almost as much time multiplying ℓ by $(N^TN)^{-1}$ to determine u as we do working with R. So instead of determining $(N^TN)^{-1}$, all we need to do is determine R. This can be done by performing q Householder transformations, which requires at most $(t-q/3)q^2$ steps, where t is the dimension of the system. In actual practice we can order the columns of N in such a way that the last n_k elements of the k^{th} column are zero, and $n_k \geq n_{k+1}$ for $k = 1,2,\ldots,q-1$. In this case, it is necessary to perform about $2(\sum_{k=1}^{q} \sum_{i}^{q} (t-n_k-i))$ operations. In practice $(t-n_k-i)$ is small.

As shown in Gill and Murray [1], updating R when a new constraint is added is not difficult. Let N represent the matrix whose columns are the unit normals to the intersection of q-1 hyperplanes in which z lies. Assume that a new constraint must be added.

Let $\bar{N} = [N \vdots n_q]$. If N = QR, then

$$\begin{bmatrix} N^TN & \vdots & N^Tn_q \\ \cdots & \vdots & \cdots \\ n_q^TN & \vdots & n_q^Tn_q \end{bmatrix} = \begin{bmatrix} R^TR & \vdots & N^Tn_q \\ \cdots & \vdots & \cdots \\ n_q^TN & \vdots & n_q^Tn_q \end{bmatrix}$$

If we set $\bar{R} = \begin{bmatrix} R & \vdots & r \\ 0 & \vdots & d \end{bmatrix}$, then

$$\bar{N}^T\bar{N} = \bar{R}^T\bar{R} = \begin{bmatrix} R^TR & \vdots & R^Tr \\ \cdots & \vdots & \cdots \\ r^TR & \vdots & r^Tr + d^2 \end{bmatrix}.$$

Hence, to determine $\bar{R}$ we need to find $\underset{\sim}{r}$ and d, where

$$R^T \underset{\sim}{r} = N^T \underset{\sim}{n}_q$$

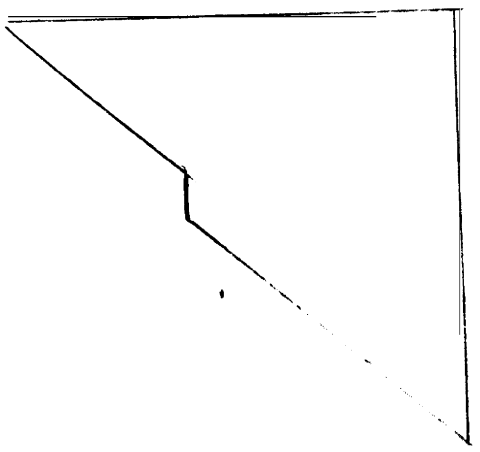
and

$$d=(\underset{\sim}{n}^T\underset{\sim}{n} - \underset{\sim}{r}^T\underset{\sim}{r})^{1/2} = (1- \underset{\sim}{r}^T\underset{\sim}{r})^{1/2}$$

Because Q is orthogonal the above square root is real.

Since R^T is a lower triangular matrix, the process requires one vector multiplication to find $N^T\underset{\sim}{n}q$ and one backsolve to determine $\underset{\sim}{r}$, and thus takes about $t \cdot q + q^2/2$ steps. In actual practice the nonzero elements of N and $\underset{\sim}{n}_q$ are known a priori and $\frac{q}{2}(t+q)$ would be a more accurate estimate of the number of operations required.

When a constraint is dropped the corresponding column must be deleted from R. The matrix then looks like



If the i^{th} column is eliminated-we have q-i elements below the diagonal which must be annihilated. Using Given's rotation matrices as suggested by Golub in [3] this can be done is $2(q-i)^2$ operations.

4.2 The algorithm in detail.

Using the results of the previous section, the whole algorithm to solve the problem can be written as follows:

1. Determine an initial guess $\underset{\sim}{x}_0$ with q constraints active, and set $\underset{\sim}{z}_0 = \bar{W}\,\underset{\sim}{x}_0 + \underset{\sim}{h} - \underset{\sim}{r}$. Compute $\bar{W}^{-1}$ if this has not been done previously. Determine the bounds ℓ_i, i = 1,...,4t, for the transformed system.

2. Set up the constraint matrix N_q where $\underset{\sim}{z}_0$ satisfies $\underset{\sim}{n}_i^T \underset{\sim}{z}_0 = \ell_i$, $1 \leq i \leq q$. Using Householder transformations, form the matrix R_q where $N = Q_q R_q$ and $Q_q^T Q_q = I$.

3. Compute the vector u where

$$R_q \underset{\sim}{u} = \underset{\sim}{m}$$

and

$$R_q^T \underset{\sim}{m} = \underset{\sim}{\ell}.$$

(Note that if step 3 is entered from steps 4 or 6, most of the elements of m have already been computed.)

Let $u_i = \min_{1 \leq j \leq q} u_j$.

4. If $u_i < 0$, delete the i^{th} column from N, using Given's rotation matrices, update R, set q = q-1, and go to 3.

If $u_i > 0$, then form $\underset{\sim}{s}_i = +\underset{\sim}{z}_i - N \cdot \underset{\sim}{u}$.

5. Determine

$$\lambda_i = \min_{\lambda_j \geq 0} \lambda_j = \frac{\underset{\sim}{n}_j^T \underset{\sim}{z}_{l-1} - \ell_j}{\underset{\sim}{n}_j^T \underset{\sim}{s}_i} \qquad j = q+1,\ldots,4t.$$

Here we are assuming that $\underset{\sim}{z}$ is constrained to lie in the &main &fined by

$$\underset{\sim j}{n}^T \underset{\sim}{z} \geq \ell_j, \qquad 1 \leq j \leq 4t.$$

Note that $\pm \underset{\sim j}{n}^T$ is either one of the rows of $\bar{W}^{-1}$ or a vector with only one nonzero component.

6. If $\lambda_i \leq 1$, add the corresponding column to N_q, form

$R_{q+1} = \begin{bmatrix} R_q & \vdots & \underset{\sim}{r} \\ 0 & \vdots & d \end{bmatrix}$, where

$$R_{q\sim}^T r = N_{q\sim i}^T$$

and

$$d = \text{sqrt}(1 - \underset{\sim}{r}^T \underset{\sim}{r});$$

set $q = q+1$, $\underset{\sim i+1}{z} = \underset{\sim i}{z} - \lambda_i \underset{\sim i}{s}$, and return to 3.

7. Set $\underset{\sim}{z} = \underset{\sim i}{z} - \underset{\sim i}{s}$ and $\underset{\sim}{x} = \bar{W}^{-1}(\underset{\sim}{z} + \underset{\sim}{r}-h)$.

x is the required drug schedule.

In our program we have ordered the constraints on $\underset{\sim}{z}$ so that the first 2t constraints correspond to eq. (14), and the last 2t constraints correspond to eq. (13). If we write the constraints as

$$\underset{\sim}{n}_j^T \underset{\sim}{z} \geq \ell_j, \qquad j = 1,2,\ldots,4t,$$

we notice that

$$\underset{\sim}{n}_j^T = -\hat{\underset{\sim}{w}}_j^T, \qquad 1 \leq j \leq t,$$

$$\underset{\sim}{n}_j^T = -\underset{\sim}{e}_{j-2t}^T, \qquad 2t+1 \leq j \leq 3t,$$

$$\underset{\sim}{n}_j^T = -\underset{\sim}{n}_{j-t}^T, \quad \begin{cases} t+1 \leq j \leq 2t \\ 3t+1 \leq j \leq 4t. \end{cases}$$

Here $\hat{\underset{\sim}{w}}_j^T$ is the j^{th} row of $\bar{W}^{-1}$ and $\underset{\sim}{e}_j$ is the unit vector in the j^{th} coordinate direction.

The fact that the matrix $\bar{W}^{-1}$ is part of the constraint matrix and enters into the transformation of variables simplifies the programming. Step 5 in the algorithm just outlined essentially requires the computation of $\bar{W}^{-1}\underset{\sim}{z}_i$ and $\bar{W}^{-1}\underset{\sim}{s}_i$. We know that

$$\underset{\sim}{z}_0 = \bar{W}\underset{\sim}{x}_0 + \underset{\sim}{h}-\underset{\sim}{r}$$

and

$$\underset{\sim}{z}_i = \underset{\sim}{z}_{i-1} - \lambda\, W\underset{\sim}{s}_{i-1}, \qquad i > 0,$$

which means

$$\bar{W}^{-1}\underset{\sim}{z}_0 = \underset{\sim}{x}_0 + \bar{W}^{-1}(\underset{\sim}{h}-\underset{\sim}{r}) \tag{16}$$

and

$$\bar{W}^{-1}\underset{\sim}{z}_i = \bar{W}^{-1}\underset{\sim}{z}_{i-1} - \lambda\, \bar{W}^{-1}\underset{\sim}{s}_{i-1}, \qquad i > 0. \tag{17}$$

Since $\bar{W}^{-1}(\underset{\sim}{n}-\underset{\sim}{r})$ is needed to compute the bounds for eq. (14), every quantity on the right-hand sides of (16) and (17) are known for $i \geq 0$. Thus, the computation of $\bar{W}^{-1}z$ in step 5 requires at most t multiplications.

Similarly, recovering x in step 7 does not involve a matrix multiplication. We are to set

$$x = \bar{W}^{-1}\underset{\sim}{z}_{i+1} - \bar{W}^{-1}(\underset{\sim}{h}-\underset{\sim}{r}).$$

But this means that

$$x = \bar{W}^{-1}\underset{\sim}{z}_i - \lambda\,\bar{W}^{-1}\underset{\sim}{s}_i - \bar{W}^{-1}(\underset{\sim}{h}-\underset{\sim}{r}). \qquad (18)$$

Since every quantity in (18) is known, $\underset{\sim}{x}$ can be obtained immediately.

5. Obtaining an Initial Guess

On the average the program can run about 10 times before any of the data is changed. Each time the drug values move up 1 time interval, so that the time represented by x_j during the i^{th} procedure call is the same time represented by x_{j-1} on the $(i+1)^{st}$ procedure call. If neither $\underline{r}$ nor $\bar{W}$ has changed, we may expect that the values $x_2,\ldots,x_t$ for the previous time interval might be excellent guesses for $x_1,\ldots,x_{t-1}$. In fact, if we allow the process sufficient time to attain the minimum in D during the last time interval, then during the next time interval only a few iterations should be required. An initial estimate for x_t can be obtained by looking at

$$e = (w_0 - r_t - \sum_{i=2}^{n} w_i x_{t-i+1})/w_1.$$

If $e < 0$, set $x_t = 0$. If $e > 50$, set $x_t = 50$; otherwise set $x_i = e$.

Usually when $\bar{W}$ is updated the changes in $\bar{W}$ are not that large. If the drug rate schedule from the previous call are still feasible, which is very likely, they may be used as initial guesses. However, when r is changed (which occurs much less often), it is wise to restart the process.

When restarting a process we have several options. We can set

$$x_{j+1} = \frac{r_{j+1}}{\sum_{k=1}^{n} w_k}, \qquad j = 0,\ldots,t. \tag{19}$$

This is the steady state solution which was Schade's original algorithm. These values are always feasible, but can be far from optimal. If other methods give answers violating the blood pressure constraints, the values given by (19) seem to be the only way to start.

Another way to start the process is to take advantage of the triangular shape of $\bar{W}$ and set $x_i = (h_i - r_i - \sum_{j=2}^{a} w_j x_{i-j+1})/w_1$, where $\ell = \min(i-1,n)$ and n is the dimension of the filter system used to determine the W's.

If x_i is greater than 50, set $x_i = 50$; if $x_i < 0$, set $x_i = 0$. If at any time a constraint on the blood pressure is violated, then revert to (16).

'The main problem with the method above is that too many constraints are usually activated that later must be dropped. The initialization procedure has been modified in several ways to help alleviate this problem.

(1) If $x_i = 0$ and $x_{i+1} = 50$, set $x_i = 10$ and recalculate x_{i+1}.

(2) If $x_i = 50$ and $x_{i+1} = 0$, set $x_i = 40$ and recalculate x_{i+1}.

(3) If the $\bar{W}$ matrix has been updated and $x_{i-1} = 50$ or $x_{i-1} = 0$, and $0 < x_i < 50$, then the column corresponding to the constraint on x_{i-1} is not included in the N matrix.

If the same constraint (upper and lower) holds for the first p components of x, and the next t-p components are unconstrained, then there is no need to call the function minimizing routine.

5.1 Computational Experience

The algorithm described in section 4 has been implemented in Algol W on the IBM 360/67. For a system of 20 variables and 80 constraints approximately 1/3 second is required to determine the correct drug rate schedule initially. When the matrix $\bar{W}$ has not been updated and the results of the previous time through the algorithm are used, the procedure requires less than 1/10 second. We think that the algorithm will be operationally practical when it is integrated into the real time system already running on an HP 2116B machine.

During the testing of the algorithm several facts were observed that are worth mentioning:

When computing λ_j we must insure that $\underset{\sim}{n}_j^T \underset{\sim}{s} > 0$. If the j^{th} hyperplane has just been dropped, it is possible that because of roundoff error the quantity $\underset{\sim}{n}_j^T \underset{\sim}{z}_i - \ell_j$ will be slightly negative. . the quantity $\underset{\sim}{n}_j^T \underset{\sim}{s}$ is tive, this constraint will be the first one encountered. Accordingly, we must check the sign of $\underset{\sim}{n}_j^T \underset{\sim}{s}$ before computing h_j.

If the $\bar{W}$ matrix has not been updated and we use information from the previous time interval, then z_{0i}, i = 1,...,t-1, need not be recomputed. However, it is still necessary to recompute R and the bounds on z. In 'fact, in this case more time may be spent computing R than in finding the minimum.

Our solutions supported the 'bang bang' principle of control theory. Often they indicated that the drug should be administered at 50 drops per minute for a period of time, then at 0 drops per minute, and finally back to 50 drops per minute. This is unfortunate from a computational point of view because it means that many constraints are active and more computer time is necessary per iteration. It is also unfortunate from an operational point of view. In this situation differences often arise between what actually occurs and what the model thinks has occurred. If an interval of time is considered to be 20 seconds, then a schedule of 50-o-50 is effectively a schedule of 50-3-50. Updating the model smoothes out some of the inconsistencies between the model and reality.

Roundoff error does accumulate but not catastrophically. Given the uncertainties in our data and the imprecision in our apparatus, we are obtaining solutions as accurately as we deserve.

Our solutions also verify the control theory principle that solutions of the problem in two distinct time intervals do not solve the problem when these two intervals are merged into one large interval. Of course, if this principle were not true, we could solve the problem once, and for successive time intervals tack on a local solution.

The active constraints can only be linearly dependent if the patient is dead, or if the person running the program sets the desired blood pressure outside the prescribed bounds. Therefore, linear dependence was not considered in the implemented algorithm although it can easily be detected by checking d when updating R.

When a blood pressure constraint is activated, only q^2 steps are required to update R as opposed to the formulation presented in section 3. Also, adding this constraint does not require added attention or code. This makes the whole program shorter and, hence, it can be more easily translated into HP assembly language so that it can be integrated into the system now running.

The procedure given in section 5 for determining z_0 often activates too many constraints which must be dropped. It was thought that if in the beginning no constraints were activated, then the ones hit during the execution of the main algorithm would still be active when the minimum of $f(z)$ for $z \in D$ was found. This was not the case. Many times the constraints encountered first were the ones later dropped. Furthermore, to construct R initially takes fewer steps than to add one column at a time.

In the table below an idea of the number of multiplications required in each phase of the program is given. In practice the quantity q is about $t/3$. Calculations which involve $O(t)$ operations have not been included.

Operation Count

Let t = dimension of the system;

q = number of active constraints;

n = dimension of the filter system or the number of nonzero diagonals of W.

1. Needed when $\bar{W}$ and r are updated:

a) initial guess of x	$(t^2 - n^2)/2$
b) computing $\bar{W}^{-1}$	$(n^2)/2$
c) initial guess of z	$(t^2 - n^2)/2$

2. Needed every time subroutine is called:

a) creating R	$(t-q/3)q^2/2$
b) bounds for z	$t^2/2$

3. Operations needed for first full iteration:

a) determining $\underset{\sim}{s}$	$q^2 + tq/2$
b) determining λ	$(t-q)t/2$

4. Dropping the i^{th} constraint:

a) updating R	$2(q-i)^2$
b) recomputing s and λ	$(q-i)^2/2 + q^2/2$

5. Adding a constraint:

a) updating R for drug rate constraint	$tq/q + q^2/2$
b) updating R for blood pressure constraint	$q^2/2$
c) recomputing s and λ	$q^2/2 + t^2/2$

REFERENCES

[1] P. E. Gill and W. Murray, "A numerically stable form of the simplex algorithm," National Physical Laboratory Maths Report 87, August 1970.

[2] D. Goldfarb, "Extension of Davidon's variable metric method to maximization under linear equality and inequality constraints," SIAM J. of Appl. Math 4 (1969), 739-764.

[3] G. H. Golub, "Matrix decompositions and statistical calculations," Stanford Computer Science Department Report CS 124, March 1969.

[4] E. Isaacson and H. Keller, Analysis of Numerical Methods, John Wiley and Sons, New York, 1966.

[5] H. P. Kunzi, W. Krelle, W. Oettli, Nonlinear Programming, Blaisdell Publishing Co., Waltham, Mass., 1966.

[6] J. B. Rosen, "The gradient projection method for nonlinear programming," SIAM J. of Appl. Math. 1 (1960), 181-217.

[7] C. M. Schade, "Optimal regulation of physiological systems via real time adaptive model synthesis," Ph.D. Dissertation, Stanford University, 1971.