SOLUTION OF A TWO STAGE INVENTORY PROBLEM USING DECOMPOSITION TECHNIQUES

by

CARLOS ALBERTO CELADA

S.B., University of Notre Dame
(1970)

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

December, 1971

Signature of Author . . . . . .
Department of Electrical Engineering, December 18, 1971

Certified by . . . . . .
Thesis Supervisor

Accepted by . . . . . .
Chairman, Departmental Committee on Graduate Students

SOLUTION OF A TWO STAGE INVENTORY PROBLEM USING DECOMPOSITION TECHNIQUES

by

Carlos Alberto Celada

Submitted to the Department of Electrical Engineering on December 18, 1971 in partial fulfillment of the requirements for the Degree of Master of Science.

ABSTRACT

A two stage inventory problem for a corporation composed of two manufacturing plants, one that manufactures an intermediate product and the other that manufactures the finished products is considered. The inventory problem is formulated as an optimization problem and solved using decomposition techniques of mathematical programming.

THESIS SUPERVISOR: Sanjoy K. Mitter
TITLE: Associate Professor of Electrical Engineering

ACKNOWLEDGEMENT

The author would like to thank Dr. Sanjoy Mitter who, acting as thesis advisor, provided invaluable encouragement and guidance in the development of this thesis. His patient suggestions for improving this author's style of writing are also gratefully acknowledged.

The author would also like to thank Robert Scott and Jaime Szajner who helped to carry out this work through their many helpful suggestions.

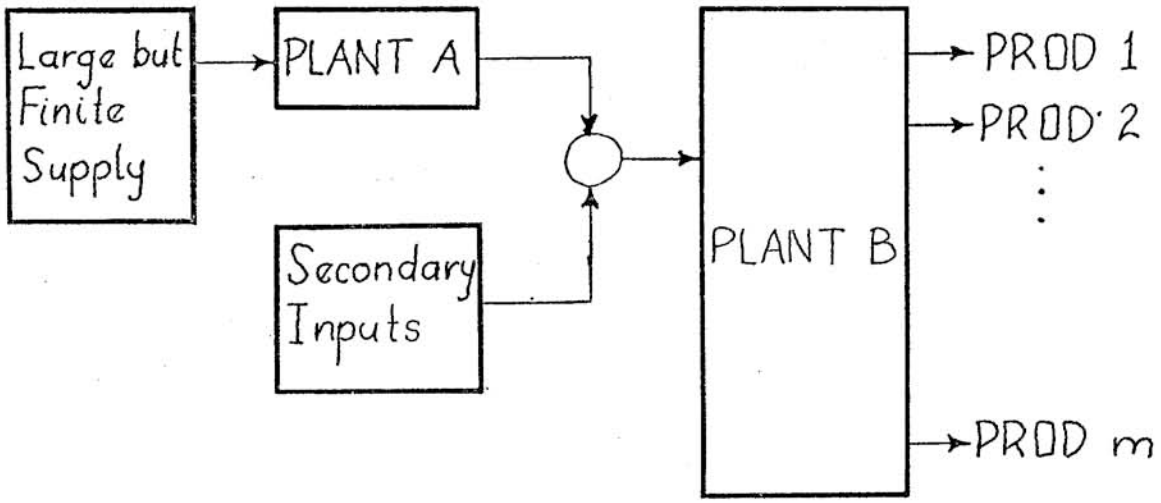
Last, but certainly not least, the author wishes to express his gratefulness to his parents who by their spiritual and monetary support have made his entire education possible.

TABLE OF CONTENTS

CHAPTER 1

PLANT DESCRIPTION

The development of this thesis will be around a manufacturing corporation composed of two main manufacturing plants: plant A and plant B. Plant A produces a single output product and takes its raw materials from a large but limited supply. Plant B is composed of several production lines, each one producing a different output product, but all having as their primary inputs the output product from plant A. In a schematic diagram, the two plants are therefore cascaded together in the following way:

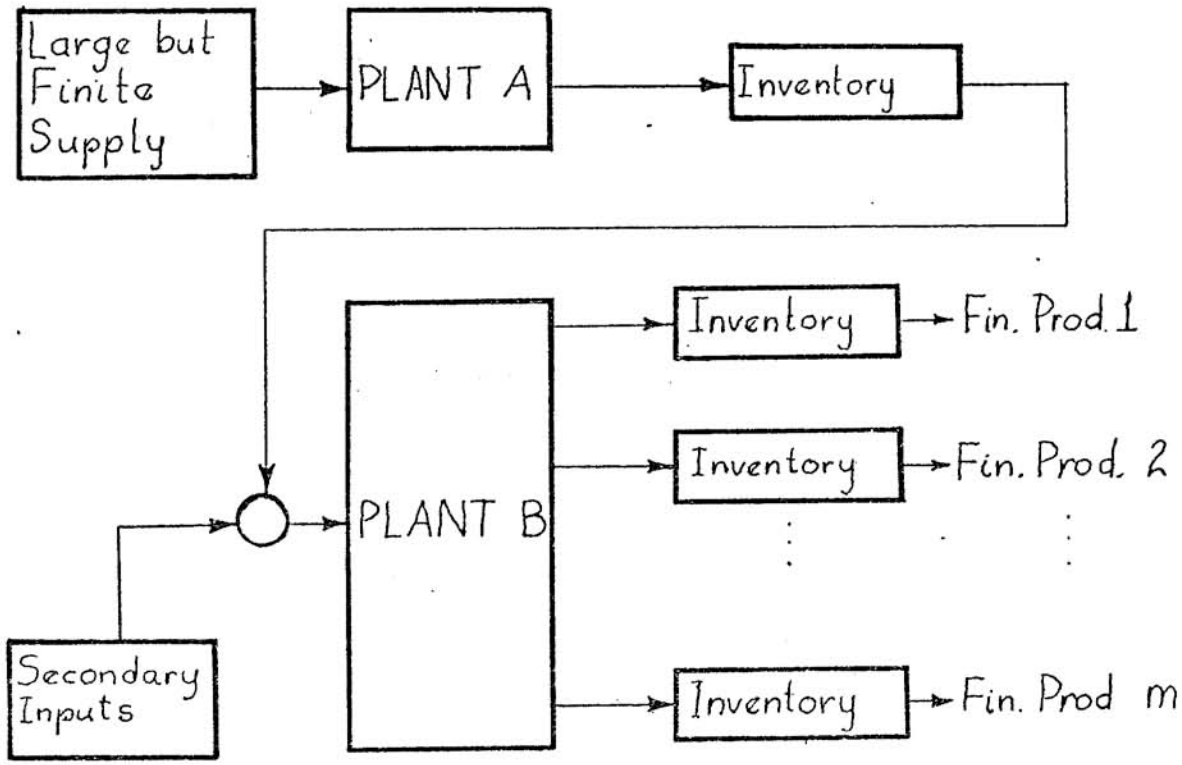


There are various industries that possess this internal structure. Steel industries, for instance, have a main iron processing furnace that distributes its single output product (steel) to other dependent industries that manufacture steel sheet, structural beams, pipes, wires and so forth. Similarly, textile industries usually have a main nylon,

dacron, or the like, manufacturing plant that distributes its output product among several other industries that produce a variety of output products. The petroleum industry would also be a perfect example of an industry having this same type of structure.

Now, the output products produced by plant B do not necessarily have to be distinguishable, that is, two subdivisions of B might very well produce identical products without disturbing in this way the structure of the system. An electrical power system, for instance, could fit into the same type of structure that we are considering, even when the output voltage from all the substations is identically the same.

Since inventory buildup is allowed for each one of the output products of plant B, as well as for that of plant A, the actual block diagram of the type of systems that will be considered is the following:



CHAPTER 2

STATEMENT OF THE PROBLEM AND FORMULATION OF THE MATHEMATICAL MODEL

1. Description of the Problem and Assumptions

The problem that will be dealt with specifically in this thesis is that of determining the optimum production schedules subject to the physical plant constraints being satisfied so as to maximize profits or minimize costs. That is, given the maximum production capacities, the maximum inventory levels, and the demand schedules for each one of the output finished products, the optimum production schedules for each one of the subdivisions as well as that for plant A must be determined.

The problem will be formulated as an open-loop optimal control problem taking the different inventory levels as the state variables of the system and the different input quantities for each one of the subdivisions of plant B as well as the input for plant A as the control variables of the system.

The demand schedules for each one of the outputs from plant B are assumed to be known through the entire period of time of interest. Consequently, the influence that the outside demands exert in the optimization is equivalent to that of a simple fixed parameter and not that of a dynamic variable, as would probably be the case in a more realistic situation. However, if the dynamics of the demand curve are incorporated into the model the optimization problem would become tremendously complicated. Besides, in the formulation of the econometric

model, in order to simulate the dynamics of the public demand, certain simplifying assumptions would undoubtedly have to be done, since it is virtually impossible to incorporate the entire economic system into the model. Therefore, before any optimization of the system is attempted, a thorough verification of the model with the real world would have to be performed. This, because of the magnitude of the system involved, is an almost impossible thing to do. Consequently, the incorporation of the dynamics of the demand schedules to the inventory model, besides being completely outside the scope of this thesis, would have a dubious contribution to the usefulness of the results.

Perhaps a better way of making the model somewhat more realistic would be to assume certain "expected" demand schedules to be known, but their true value through time to be actually uncertain. That is, instead of assuming perfectly deterministic demand curves, a certain degree of randomness in their actual values through time could be assumed. By specifying the demand curves in terms of their statistical parameters (mean and variance if a Gaussian distribution is assumed) it would be possible to simulate in the model the actual uncertainty that exists in a true economic market. However, the actual study of the effects of introducing a certain degree of randomness into the demand curves will not be done in this thesis and it is only recommended as a possibility for further research on this problem.

The demands through time for each of the finished products from plant B are not considered to be cumulative in case of failure to supply. That is, if a particular subdivision fails to supply the entire given demand for a particular period of time, the resulted excess demand will

be assumed lost as far as that subdivision is concerned, instead of adding it to future demands. In other words, what we are assuming in economic terms is that the number of different suppliers for each of the products sold is such that if a particular subdivision fails to supply his entire share of the demand, some other supplier will take the excess demand leaving, in this way, the overall economic system completely unaffected. This assumption is not only taken for the sake of simplicity, but also to be consistent with the assumption that the demand schedules are completely deterministic and fixed and thereby unaffected by the production policies of the particular company that is being dealed with.

Also, in order to be consistent with the two previous assumptions, the market price for the output products from plant A is assumed to be completely deterministic and known through time. For the sake of simplicity, the values of these market prices were considered to be constant through the entire period of time of interest. Actually, this detail could be very easily changed in the model; it was simply felt that this parameter was actually of very little importance as far as the dynamics of the model were concerned.

Another underlying assumption in the model is the fact that plant A is not allowed to sell any of its output product to outside consumers. This assumption was not only made for the sake of simplicity, but also because in actual practice it is found that the type of industry to which this model fits has zero or very little outside market for the product produced by the plant that in this model corresponds to plant A.

It should also be mentioned that in the actual optimization process the supplier of raw materials for plant A was assumed to be virtually unlimited in the sense that it imposed no limit on the maximum production allowed for that plant through the entire period of time considered. That is, the maximum of production for plant A was assumed to be determined by the internal production capacity of the plant itself and not by the availability, at the particular instant of time, of the necessary raw materials. Similarly it was assumed that the "secondary" inputs to plant B imposed to limitation on the production capabilities of any of the subdivisions at any instant of time. That is, the secondary inputs supply was considered ample enough to be able to supply all the subdivisions of plant B with the necessary materials even when they are simultaneously producing at full internal capacity. Finally, the market prices for both the raw materials for plant A and for the secondary inputs for plant B were assumed to be constant through the entire period of time considered. Again, the justification for these assumptions is just simply that in actual practice it is found that for the type of industry to which this model applies this is actually the case.

The amount of output coming out from a given subdivision was assumed to be directly proportional to the amount of input fed in. That is, the output function $o(t)$ was assumed to be related to the input function $u(t)$ by the following linear relation:

$$o(t) = k \cdot u(t) \tag{1}$$

where k, the proportionality factor, would correspond to the production efficiency of the subdivision. Also, it should be remarked that, as

equation (1) implies, the production process was assumed to be absolutely instantaneous. In other words, the production process was assumed to have zero processing delay. There is no physical justification for this assumption except that of simplifying the problem.. Actually, as it will be explained later, by modifying the structure of the model in the proper way the significance of this seemingly major assumption may be made to be no longer relevant.

2. Dynamics of the Problem

The dynamics of plant A are given by the following set of equations:

$$s_o(t) = \int_o^t [u_o(t)\ k_o - d_o(t)]dt$$

or in terms of a differential equation:

$$\frac{ds_o(t)}{dt} = u_o(t)\ k_o - d_o(t) \qquad (2)$$

where: $s_o(t)$ is the state of plant A and is the inventory level

$u_o(t)$ is the input being consumed by the plant at time t

k_o is the production efficiency of the plant

and $d_o(t)$ is the expected demand at time t

For plant B, the dynamics of each one of the subdivisions may be described by a similar set of equations. For instance, the dynamic equation for the i^{th} subdivision would be:

$$s_i(t) = \int_o^t [k_i\ u_i(t) - d_i(t)]dt$$

or in differential equation form:

$$\frac{d}{dt} s_i(t) = k_i\, u_i(t) - d_i(t) \qquad (3)$$

where $s_i(t)$ is the value of the inventory level of the i^{th} subdivision at time t

$u_i(t)$ is the input being applied to the i^{th} subdivision at time t

k_i is the production efficiency of the i^{th} subdivision

and $d_i(t)$ is the expected demand for the i^{th} subdivision at time t

Clearly, the dynamics of the entire plant B would be described by a set of m equations like equation (3), one for each subdivision.

If we let:

a) $\underline{s}(t)$ be the vector composed of the different state inventory levels for plants A and B

b) $\underline{u}(t)$ be the vector composed of the corresponding input variables

c) $\underline{d}(t)$ be the vector composed of the corresponding demands, and

d) K be the matrix containing the different production efficiencies in its diagonal and zeroes everywhere else,

then the entire dynamic structure of the model may be written in a conveniently compact matrix form. That is, if we let

$$\underline{s}(t) = [s_o(t), s_1(t), s_2(t), \ldots, s_m(t)]'$$

$$\underline{u}(t) = [u_o(t), u_1(t), u_2(t), \ldots, u_m(t)]'$$

$$\underline{d}(t) = [d_o(t), d_1(t), d_2(t), \ldots, d_m(t)]'$$

and

$$K = \begin{bmatrix} k_o & 0 & \cdots & 0 \\ 0 & k_1 & 0 & \cdots \\ 0 & 0 & k_2 & 0 \\ \vdots & & \ddots & 0 \\ 0 & 0 & \cdots & 0 \quad k_m \end{bmatrix}$$

then the m+1 equations that determine the dynamic structure of the system may be written by the following single matrix equation

$$\frac{d}{dt}\underline{s}(t) = K\underline{u}(t) - \underline{d}(t) \qquad (4)$$

where the derivative of $\underline{s}(t)$ is defined to be the vector composed of the derivatives of each one of the components of $\underline{s}(t)$.

At this point, is is important to realize that the way the dynamics of the model have been formulated, both plant A and all the subdivisions of B have been taken as single input single output subsystems. In the case of plant A, that was the way the problem was formulated to start with. But in the case of the subdivisions of B, in addition to the input proportioned by A, the subdivisions were assumed to require the so-called secondary inputs. Now the underlying assumption here is that these secondary inputs are merely auxilliary as far as production is concerned; that is, these secondary inputs cannot be transformed by themselves into marketable output, but serve merely as simple catalysts to the transformation of the input proportioned by plant A. Now since, as may be recalled, the supply of these secondary inputs was assumed to be unlimited as far as the capabilities of consumption of the plant was concerned, these inputs cannot have any influence on the dynamics of the

plants and consequently may be ignored. It is for this reason that each of the subdivisions were taken to be single input single output systems. Now, if the particular problem at hand cannot be reduced to a set of single input single output subsystems, as was assumed to be the case here, then an entirely different problem would have to be approached since the development that follows would simply not apply.

3. Cost Functional

The cost function or performance criterion that was formulated for the optimization of the system consists of three main items:

a) Inventory cost.

b) Change of production cost.

c) Sales profit.

The inventory cost item deserves little explanation since it is quite obvious that very seldom can an amount of inventory be stocked without having to incur in certain costs like: merchandise handling, warehouse rents, deterioration of the merchandise and so forth. Therefore, this item in the cost is designed to keep the inventory levels as low as possible without hurting the profits.

The Change in Production cost is perhaps the most obscure item in the cost function and deserves a little explanation. The reason why it is desirable to cost changes in production is that there is always a certain cost associated with every change in production. For steps up in production, for instance, there might be some hiring costs, but most importantly, there is always a certain time constant for the total production to catch up with the desired level; therefore, if an optimal

trajectory exists it must be such that sharp changes in production do not occur if it is to be feasible. For steps down in production this term has little meaning except for the fact that it might be necessary to incur in some layoff costs. This term, therefore, is primarily designed to keep the optimal solution from having abrupt changes in production so as to make it feasible.

Finally, the sales profit which contributes negatively to the cost needs very little explanation since its meaning is intuitively obvious. It should be enough to mention that some subdivisions may be more profitable than others, and consequently, in tight situations these subdivisions should be given preference over the others so as to obtain the "most profitable" distribution of resources.

Both the amounts of inventory and the net changes in production were charged in a quadratic fashion. In this way, large amounts of inventory or large changes in production were penalized proportionately much more severely than small amounts of inventory or small changes in production.

The Sales profit is, of course, directly proportional to the Total Sales. Clearly, the proportionality constant in this case is the marginal sales profit of the particular product, that is, sales price minus production cost per unit of merchandise.

Translating these ideas into mathematical terms, the cost functional may be expressed as follows

$$J = \int_{t_o}^{T} \sum_{i=0}^{m} [w_i \, s_i^2(t) + t\, c_i (du_i/dt)_t^2 - mp_i \, ts_i(t)]dt$$

where a) w_i is the inventory cost for the i^{th} subdivision

b) tc_i is the change of production cost for the i^{th} subdivision

c) mp_i is the marginal profit of the i^{th} subdivision

and d) $ts_i(t)$ is the total sales of the i^{th} subdivision at time t.

The Total Sales function is defined as follows:

$$ts_i(t) = \min\{(s_i(t-dt) + k_i u_i(t)),\ d_i(t)\} \qquad i=0,1,\ldots m$$

That is, all subdivisions are assumed to sell always as much as they can.

Now, defining a) a matrix W containing the inventory costs in its diagonal and zeroes everywhere else, b) a matrix T containing the corresponding change in production costs in its diagonal and zeroes everywhere else, and c) a third diagonal matrix M with the corresponding marginal profits, we can write the cost functional in a more convenient and familiar format. That is, defining:

$$W = \begin{bmatrix} w_o & & & \\ & w_1 & & \\ & & w_2 & \\ & & & \ddots \\ & & & & w_m \end{bmatrix} \qquad T = \begin{bmatrix} tc_1 & & & \\ & tc_2 & & \\ & & tc_3 & \\ & & & \ddots \\ & & & & tc_m \end{bmatrix}$$

and

$$M = \begin{bmatrix} mp_1 & & & \\ & mp_2 & & \\ & & mp_3 & \\ & & & \ddots \\ & & & & mp_m \end{bmatrix}$$

Then we can express the total cost in the following way:

$$J = \int_{t_o}^{T} [\underline{s}'(t)\, W\underline{s}(t) + \dot{\underline{u}}'(t)\, T\dot{\underline{u}}(t) + Mts(t)]dt \qquad (5)$$

where $\dot{\underline{u}}(t) = \frac{d}{dt}\,\underline{u}(t)$

and $\underline{ts}(t) = [ts_o(t),\ ts_1(t),\ ts_2(t),\ldots,\ ts_m(t)]'$

4. Optimization Constraints

The optimization must be performed subject to the physical constraints of the plants. Namely, the maximum inventory capacities and maximum production capacities. That is, the optimization must be performed subject to:

$$0 \le s_i(t) \le s_i^* \qquad i=0,\ldots,m$$

$$0 \le u_i(t) \le u_i^* \qquad t=t_o,\ldots,T$$

Or in vector form:

$$\begin{aligned} &0 \le \underline{s}(t) \le \underline{s}^* \qquad t=t_o,\ldots,T \\ &0 \le \underline{u}(t) \le \underline{u}^* \end{aligned} \qquad (6)$$

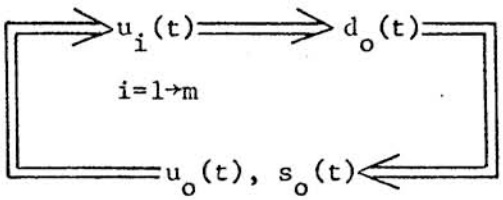
where $\underline{s}^* = [s_o^*, s_1^*, s_2^*, \ldots, s_m^*]'$

and $\underline{u}^* = [u_o^*, u_1^*, u_2^*, \ldots, u_m^*]'$

Finally, and most importantly, all feasible solutions must satisfy the material balance equation. That is, the sum of the inputs to plant B must always be smaller than or equal than the availability of the output from A for all $t \in [t_o, T]$. In mathematical terms:

$$\sum_{i=1}^{m} u_i(t) \le s_o(t-dt) + k_o\, u_o(t) \qquad (7)$$

As may be recalled, it is assumed in the optimization that the outside demands for the outputs from plant B are completely deterministic and known. But so far the demand curve for the output from plant A ($d_o(t)$) has not been specified. The determination of this demand curve is not as simple as adding up the demands divided by the production efficiencies for each of the subdivisions of B, as one might be led to believe. The reason for this is that since limited production capacities are assumed, saturation effects might occur; some subdivisions, for instance, anticipating a time of crisis might start producing in advance, producing in this way a backwards shift in the demand curve. Therefore, the demand for plant A is not known until the final production schedules for the subdivisions of plant B are known. Now the optimum production schedules for plant B depend on the production schedules for plant A; consequently, since the production schedules for plant A depend on the demand curve $d_o(t)$, there seems to be a vicious circle. That is:



However, as will be explained later, this difficulty may be overcome. For the time being it will suffice to remember that the constraint

$$d_o(t) = \sum_{i=1}^{m} u_i(t) \qquad t \in [t_o, T] \tag{8}$$

forms part of the constraint set of the optimization.

5. Summary

The open loop control problem may be stated as follows:

Determine the optimal trajectories $\underline{s}(t)$ and optimal controls $\underline{u}(t)$ for $t \in [t_o, T]$ that minimize the cost function given by equation (5) subject to equations (4), (6), (7) and (8). That is:

find the $\underline{s}(t)$ and $\underline{u}(t)$, $t \in [t_o, Y]$ that

$$\text{minimize} \quad J_T = \int_{t_o}^{T} [\underline{s}'(t)\, W\underline{s}(t) + \dot{\underline{u}}'(t)\, T\dot{\underline{u}}(t) - M_t\underline{s}(t)]dt$$

$$\text{subject to:} \quad \dot{\underline{s}}(t) = K\underline{u}(t) - \underline{d}(t)$$

$$0 \leq \underline{s}(t) \leq \underline{s}^*$$

$$0 \leq \underline{u}(t) \leq \underline{u}^*$$

$$\sum_{i=1}^{m} u_i(t) \leq s_o(t-dt) + k_o\, u_o(t)$$

$$d_o(t) = \sum_{i=1}^{m} u_i(t)$$

CHAPTER 3

DISCRETIZATION OF THE PROBLEM

The use of integrals and derivatives in the statement of the problem implies that the time increments dt that are considered are of infinitesimal magnitude and, furthermore, that the dynamics of the system are continuous. Actually, there is nothing intrinsically wrong with this approach, except that it is not quite realistic. Production changes are not made continuously, but rather they tend to be made by the week, by the month or so. By the same token, it makes better sense to talk of a production or demand of, for instance, so many cars per day, rather than a fraction of cars per unit of time. Therefore, it is not only convenient but also desirable to use a large by finite time base Δt instead of an infinitesimal one. Clearly, what this essentially amounts to is a discretization of time.

Now if time is discretized in this fashion, the integrals become summations and the derivatives differences, but the structure of the problem remains essentially the same, namely:

Find the $\underline{s}(n)$ and $u(n)$ $\quad n=0,\ldots,N$ that

$$\text{minimize} \quad J_T = \underline{s}'(N)W\,\underline{s}(N) - M\,\underline{ts}(N) + \sum_{n=0}^{N-1} [-M\,\underline{ts}(n) + \underline{s}'(n)W\,\underline{s}(n) + (\underline{u}(n) - \underline{u}(n+1))'\,T(\underline{u}(n) - \underline{u}(n+1))] \tag{9}$$

subject to:

$$\underline{s}(n+1) = \underline{s}(n) + K\,\underline{u}(n) - \underline{d}(n) \tag{10}$$

$$0 \leq \underline{s}(n) \leq \underline{s}^* \tag{11}$$

$$0 \leq \underline{u}(n) \leq \underline{u}^* \tag{12}$$

$$\sum_{i=1}^{m} u_i(n) \leq s_o(n) + k_o\, u_o(n) \tag{13}$$

$$d_o(n) = \sum_{i=1}^{m} u_i(n) \tag{14}$$

$$n=0,\ldots,N$$

where the nomenclature of the variables is the same as that used in the continuous case and the index "n" refers to the time $n(\Delta t) + t_o$, i.e.

$$\underline{s}(n) = \underline{s}(t_o + n\,\Delta t)$$

$$\underline{u}(n) = \underline{u}(t_o + n\,\Delta t)$$

for $n=0,\ldots,N$, where $N = \frac{T-t_o}{\Delta t}$.

There are other more subtle, but very significant, advantages that derive from this time discretization besides the conceptual ones previously mentioned. Firstly, by discretizing time in this way the number of control decisions along the optimal trajectory has been reduced from an infinite number of points to a finite number of points. In other words, the problem has been changed from an infinite dimensional one as far as control decisions are concerned to a finite dimensional one, thereby achieving a significant simplification in the mathematics of the optimization. Secondly, when time is discretized in this way, the outside demand curves have to be aggregated into isolated lumps across the time scale at Δt time intervals. Clearly, this concentration of the demands

into a finite number of lumps tends to flatten out short time oscillations (in particular noise) when they exist. Therefore under the presence of uncertainty, one of the results of this aggregation is that the probability distribution of the resulting points is significantly sharpened up around their expected values; obviously the greater the time interval Δt the more pronounced this effect becomes. Consequently, if a sufficiently large time discretization interval (Δt) is chosen, the assumption of a noiseless and deterministic demand curve may become much less crude than it might have previously appeared. And finally, also thanks to this time discretization, we can make the assumption of zero processing delay to be almost perfectly legitimate. That is, if we choose Δt to be a multiple or submultiple of the greatest processing delay among all the subdivisions, it is always possible to compensate for most processing delays by shifting the demand curves backwards in time by the corresponding processing delays. Now, the reason why this processing delay compensation is not admissible in the continuous case is because, due to the assumed limited production capacities, saturation effects might take place causing the equivalent "virtual" demand to be completely distorted. Consequently, since these saturation effects cannot be anticipated beforehand, a processing delay compensation of this type would be virtually impossible. If time is discretized, however, due to the already aggregated character of the demand curve, the distortion resulting from saturation effects is significantly less than in the continuous case. Consequently, a processing delay compensation of the type proposed here may be implemented successfully provided the time scale is adequately discretized.

Summarizing what has just been said, the discretization of the problem has the following advantages:

a) Helps in the interpretation of the problem with respect to the real world.

b) Transforms the problem into a finite dimensional problem as far as the number of control decision points is concerned.

c) Flattens out the noise that exists in the actual demand curve making the noiseless demand assumption somewhat more accurate.

d) It is possible to successfully compensate for non-zero processing delays (if they exist).

Now there are also some disadvantages associated with the time discretization of the problem. As will be explained later, this time discretization causes some severe difficulties in the actual design and operation of the optimization algorithm due to the inevitable discontinuities in the gradient function. However, all things considered, this author believes that time discretization does indeed pay off.

After this time discretization has been done, it is somewhat obvious that in order for the terminal time cost functional to be really meaningful it is necessary to incorporate an extra term that penalizes the system for passing out non-zero inventories to time stage N+1. That is, the system should pay for leaving non-zero inventories at the end of the time period considered so as to avoid leaving expensive inventories for "the other guys" to pay, as one might say. Therefore, it is necessary to add an extra term costing the inventories at time N+1 to the cost functional, i.e.

$$J_T = \underline{s}'(N+1)W\,\underline{s}(N+1) + \underline{s}'(N)W\,\underline{s}(N) - M\,\underline{ts}(N)$$
$$+ \sum_{n=0}^{N-1} \{\underline{s}'(n)W\,\underline{s}(n) - M\,\underline{ts}(n) \qquad (15)$$
$$+ [\underline{u}(n) - \underline{u}(n+1)]'\,T[\underline{u}(n) - \underline{u}(n+1)]\}$$

Therefore the optimization problem becomes: find the optimal trajectories and optimal controls

$$\underline{s}(n),\ \underline{u}(n) \quad \text{for } n=0,\ldots,N$$

that minimize J_T as given by equation (15) subject to the constraints given by equations (10) → (14) which will be rewritten below for ease of reference:

$$\underline{s}(n+1) = \underline{s}(n) + K\,\underline{u}(n) - \underline{d}(n) \qquad (10)$$

$$0 \le \underline{s}(n) \le \underline{s}^* \qquad (11)$$

$$0 \le \underline{u}(n) \le \underline{u}^* \qquad (12)$$

$$\sum_{i=1}^{m} u_i(n) \le s_o(n) + k_o\,u_o(n) \qquad (13)$$

$$d_o(n) = \sum_{i=1}^{m} u_i(n) \qquad (14)$$

$$n=0,\ldots,N$$

CHAPTER 4

DEVELOPMENT OF THE ALGORITHM

1. Decomposition of the Problem

The only equations that couple plant A with plant B and the subdivisions of B among themselves are the material balance equation (equation (13)) and the plant A demand equation (equation (14)). Therefore, the problem would be separable into m+1 single input-single output problems if equations (13) and (14) were not constraints. However, as shall be explained, it is possible to find a dual problem that is separable by means of Lagrangian functions.

Defining a Lagrangian function $L(\underline{c},\underline{p},\underline{u},d_o)$ as:

$$L = J_T + \sum_{n=0}^{N} \{c(n)[\sum_{i=1}^{m} u_i(n) - (s_o(n) + k_o\, u_o(n))] + p(n)[\sum_{i=1}^{m} u_i(n) - d_o(n)]\} \tag{16}$$

where $\underline{c} = [c(0),C(1),C(2),\ldots,C(N)]'$

and $\underline{p} = [p(0),p(1),p(2),\ldots,p(N)]'$

are Lagrange multiplier vectors.

And, defining $h(\underline{c},\underline{p})$ as:

$$h(\underline{c},\underline{p}) = \min_{(\underline{u},d_o)\in\Omega} L(\underline{c},\underline{p},\underline{u},d_o) \tag{17}$$

where Ω is the set of $\underline{u}(n)$, $d_o(n)$; n=0,...,N such that:

i) $\underline{s}(n+1) = \underline{s}(n) + K\,\underline{u}(n) - \underline{d}(n)$

ii) $0 \leq \underline{s}(n) \leq \underline{s}^*$

iii) $0 \leq \underline{u}(n) \leq \underline{u}^*$

Then it can be shown (see Appendix I) that the problem:

$$\underset{\underline{c},\underline{p}}{\text{maximize}}\ h(\underline{c},\underline{p}) \tag{18}$$

subject to $\underline{c} \geq 0$

is indeed the dual of the original problem (equations (10) → (15)).

Substituting the expression for J_T (equation (15)) into the Lagrangian function $L(\underline{c},\underline{p},\underline{u},d_o)$ (equation (16)), it is obtained:

$$\begin{aligned} L = {} & \underline{s}'(N+1)W\,\underline{s}(N+1) + \underline{s}'(N)W\,\underline{s}(N) - M\,\underline{ts}(N) \\ & + c(N)\Big[\sum_{i=1}^{m} u_i(N) - (s_o(N) + k_o\,u_o(N))\Big] \\ & + p(N)\Big[\sum_{i=1}^{m} u_i(N) - d_o(N)\Big] + \sum_{n=0}^{N-1} \{\underline{s}'(n)W\,\underline{s}(n) \\ & - M\,\underline{ts}(n) + [\underline{u}(n) - \underline{u}(n+1)]'\,T[\underline{u}(n) - \underline{u}(n+1)] \\ & + c(n)\Big[\sum_{i=1}^{m} u_i(n) - (s_o(n) + k_o\,u_o(n))\Big] \\ & + p(n)\Big[\sum_{i=1}^{m} u_i(n) - d_o(n)\Big]\} \end{aligned}$$

Now, if all the matrix and vector operations are expanded to the individual elements and the terms rearranged in the proper way, the equation above can be rewritten as follows:

$$
\begin{aligned}
L = & \sum_{i=0}^{m} \{w_i[s_i^2(N) + s_i^2(N+1)] - mp_i\ ts_i(N)\} \\
& + c(N)\{\sum_{i=1}^{m} u_i(N) - [s_o(N) + k_o\ u_o(N)]\} \\
& + p(N)[\sum_{i=1}^{m} u_i(N) - d_o(N)] \qquad (19) \\
& + \sum_{n=0}^{N-1} \sum_{i=0}^{m} \{w_i\ s_i^2(n) + Tc_i[u_i(n) - u_i(n+1)]^2 \\
& - mp_i\ ts_i(n) + u_i(n)[c(n) + p(n)]\} \\
& - \sum_{n=0}^{N-1} \{c(n)[s_o(n) + (k_o+1)u_o(n)] \\
& + p(n)[d_o(n) + u_o(n)]\}
\end{aligned}
$$

Therefore, as shown in equation (19), the Lagrangian function L is separable in the sense that it does not have any product terms of variables associated with different plants or different subdivisions within the same plant. That is, it may be written as a summation of a function depending on variables associated with the first plant, plus a function dependent on variables associated with the first subdivision of the second plant, plus a function dependent on variables associated with the second subdivision of the second plant, and so on. In other words, it may be written as:

$$L = f_o(\underline{c},\underline{p},\underline{u}_o,d_o) + f_1(\underline{c},\underline{p},\underline{u}_1) + \ldots + f_m(\underline{c},\underline{p},\underline{u}_m)$$

where $f_o = w_o[s_o^2(N) + s_o^2(N+1)] + mp_o\ ts_o(N)$

$$- c(N)[s_o(N) + k_o\ u_o(N)] - p(N)\ d_o(N)$$

$$+ \sum_{n=0}^{N-1} \{w_o\ s_o^2(n) + tc_o[u(n) - u_o(n+1)]^2$$

$$- mp_o\ ts_o(n) - c(n)[s_o(n) + k_o\ u_o(n)] - p(n)\ d_o(n)\}$$

and $\underset{i=1\to m}{f_i} = w_i[s_i^2(N) + s_i^2(N+1)] - mp_i\ ts_i(N) + u_i(N)[c(N) + p(N)]$

$$+ \sum_{n=0}^{N-1} \{w_i\ s_i^2(N) + tc_i[u_i(n) - u_i(n+1)]^2$$

$$- mp_i\ ts_i(n) + u_i(n)[c(n) + p(n)]\}$$

Therefore, the dual function $h(\underline{c},\underline{p})$ may be written as:

$$h(\underline{c},\underline{p}) = \underset{(\underline{u},d_o)\in\Omega}{\text{minimum}} \left[\sum_{i=0}^{m} f_i\right]$$

where Ω is the set of $\underline{u}(n)$, $d_o(n)$; $n=0,\ldots,N$

such that i) $\underline{s}(n+1) = \underline{s}(n) + K\ \underline{u}(n) - \underline{d}(n)$

ii) $0 \leq \underline{s}(n) \leq \underline{s}^*$

iii) $0 \leq \underline{u}(n) \leq \underline{u}^*$

However, since the constraint set Ω does not contain any coupling constraints between plants or between subdivisions of the same plant, the minimization that defines $h(\underline{c},\underline{p})$ is separable into m+1 independent minimizations. That is:

$$h(\underline{c},\underline{p}) = \min_{\Omega} f_o + \min_{\Omega} f_1 + \ldots + \min_{\Omega} f_m$$

$$= \min_{\Omega}\{w_o[s_o^2(N) + s_o^2(N+1)] - mp_o\ ts_o(N)$$

$$- c(N)[s_o(N) + k_o u_o(N)] - p(N)\ d_o'(N)$$

$$+ \sum_{n=i}^{N-1} \{w_o\ s_o^2(n) + tc_o[u_o(n) - u_o(n+1)]^2 - mp_o \quad ts(n)$$

$$- c(n)[s_o(n) + k_o\ u_o(n)] + p(n)\ d_o(n)\}\}$$

$$+ \sum_{i=1}^{m} \min_{\Omega}\{w_i[s_i^2(N) + s_i^2(N+1)] - mp_o\ ts_i(N)$$

$$+ u_i(N)[c(N) + p(N)]$$

$$+ \sum_{n=0}^{N-1} \{w_i\ s_i^2(n) + tc_i[u_i(n) - u_i(n+1)]^2$$

$$- mp_o\ ts_i(n) + u_i(n)[c(n) + p(n)]\}\}$$

Summarizing, the dual problem can be stated as:

$$\text{maximize}_{\underline{c},\underline{p}}\ h(\underline{c},\underline{p})$$

$$\text{subject to}\quad c \geq 0 \qquad \underline{c},\underline{p} \in R^m$$

where $$h(\underline{c},\underline{p}) = \min_{(\underline{u},d_o)\in\Omega} L(\underline{c},\underline{p},\underline{u},d_o)$$

$$= \min_{(\underline{u}_o,d_o)\in\Omega} f_o(\underline{c},\underline{p},\underline{u}_o,d_o) + \sum_{i=1}^{m} \min_{u_i\in\Omega} f_i(\underline{c},\underline{p},\underline{u}_i)$$

Or:

$$\text{maximize}_{\underline{p}\in R^N;\underline{c}\geq 0} \quad \min_{(\underline{u}_o,d_o)\in\Omega} f_o(\underline{c},\underline{p},\underline{u}_o,d_o) + \sum_{i=1}^{m} \min_{u_i\in\Omega} f_i(\underline{c},\underline{p},\underline{u}_i) \qquad (21)$$

As shown in Appendix I, if there exist a pair of vectors $\underline{c}^*,\underline{p}^*$ with the property that the controls resulting from the inner minimization $u(\underline{c}^*,\underline{p}^*)$, $d_o(\underline{c}^*,\underline{p}^*)$ solve the primal problem, then these same vectors $\underline{c}^*,\underline{p}^*$ also solve the dual, and vice versa. Therefore, it is possible to solve the problem by solving the dual, making use in this way of the decomposition properties inherent in the structure of the problem.

2. The Algorithm

The actual solution of the problem was done by means of a steepest ascent algorithm. The procedure was the following:

1) Initialize $\underline{c}_o,\underline{p}_o$; say $\underline{c}_o=0$, $\underline{p}_o=0$

2) Get $[\underline{v}_o,\hat{\underline{v}}_o]'$ $\nabla h(\underline{c}_o,\underline{p}_o)$

3) Get α^* such that

$$H_o(\alpha^*) = \max_{\alpha\geq 0} H(\alpha)$$

where $H(\alpha) = h(\underline{c}_o + \alpha\, \underline{v}_o,\ \underline{p}_o + \alpha\, \hat{\underline{v}}_o)$

4) Let $(\underline{c}_1,\underline{p}_1) = (\underline{c}_o + \alpha\underline{v}_o,\ \underline{p}_o + \alpha\, \hat{\underline{v}}_o)$.

5) Get direction of steepest ascent at point $(\underline{c}_1,\underline{p}_1)$; store it in $[\underline{v}_1,\hat{\underline{v}}_1]'$

6) Get α^* such that

$$H_1(\alpha^*) = \max_{\alpha\geq 0} H(\alpha)$$

where $H(\alpha) = h(\underline{c}_1 + \alpha\, \underline{v}_1,\ \underline{p}_1 + \alpha\, \hat{\underline{v}}_1)$

7) Make $(\underline{c}_1,\underline{p}_1) = (\underline{c}_1 + \alpha^*\, \underline{v}_1,\ \underline{p}_1 + \alpha^*\, \hat{\underline{v}}_1)$

8) If $(H_1-H_o) \geq H^*$

i) Make $H_o = H_1$

ii) Start a new cycle with step 5)

If $(H_1-H_o) < H^*$ proceed with step 9)

9) Get the direction of steepest ascent at point $(\underline{c}_1, \underline{p}_1)$ and store it in $[\underline{v}_1, \hat{\underline{v}}_1]$ over again

10) If any of the components of $[\underline{v}_1, \underline{v}_1]$ is greater than or equal to v*

i) make $H_o = H_1$

ii) start a new cycle with step 6)

If none of the components of $[\underline{v}_1, \hat{\underline{v}}_1]$ is greater than or equal to v*

STOP

3. Solution of the Local Problem

The solution of the local problem or inner minimization involves, as was derived before, the solution of m+1 nonlinear programming problems with linear constraints. Namely, it involves the solution of the maximization given by equation (20). That is, it is necessary to solve problems of the type:

$$\min_{\underline{u}} \{Tc[c(N), p(N), x(N), x(N+1)] + \sum_{n=0}^{N-1} F[\underline{c},\underline{p}, x(n), u(n), u(n+1)]\} \tag{22}$$

subject to:

$$x(n+1) = x(n) + k\, u(n) - d(n) \tag{23}$$

$$0 \leq s(n) \leq s^* \tag{24}$$

$$0 \leq u(n) \leq u^* \tag{25}$$

It is clear that equation (23) may be used to illuminate the x(n) terms from equation (22), reducing the problem to

$$\min\{\hat{Tc}[c(N),\ p(N),\ u(N),\ u(N-1)]$$

$$+ \sum_{n=0}^{N-1} \hat{F}[\underline{c},\underline{p},u(n)]$$

subject to:

$$0 \leq s(n) \leq s^*$$

$$0 \leq u(n) \leq u^*$$

The problem would be, therefore, reduced to a large quadratic programming problem with limiting constraints. The control constraints may very easily be taken care of by means of penalty functions of some kind, but the state constraints are not so easily complemented into the optimization algorithm. One way of taking care of those constraints without overcomplicating the problem is by means of an additional pricing mechanism that modifies the control bounds for each time stage in such a way that the state trajectory is forced to lay within the specified bounds.

That is, the problem is initially solved ignoring the state constraints; if the optimum state trajectory, so obtained, does not lay entirely within the specified region, then the control bounds at the critical points are modified appropriately. The problem is then solved a new time, again ignoring the state constraints, but now using the modified control bounds. The new optimal state trajectory is then checked to see if it lays entirely within the required bounds; if not, the control bounds are modified accordingly and the problem is solved over again. This procedure is repeated as many times as necessary until the resulting optimal controls obtained from the minimization yields a state trajectory that does not violate the constraints at any point.

The problem with this approach is that even though the algorithm is certain to converge, it would most probably take so much time that its implementation becomes impractical. Also, since the linear dynamic constraint that determines the structure of the system is built into the performance criterion, nice properties about the structure of the system are lost or ignored.

Another possible method of approach to this optimization is the use of the Discrete Minimum Principle. The advantages of this approach are that it is possible to derive certain generalizations about the nature of the optimal solutions. Even when the necessary conditions dictated by the Minimum Principles do not determine a closed criterion from which the optimal solutions may be derived, these necessary conditions are usually quite useful to test for optimality a solution obtained with some other method. In other words, even though the Minimum Principle does not always give directly a set of possible optimal trajectories, it usually provides useful clues as to the nature of the optimal solutions (if they exist).

Now, the problems at hand are essentially linear-regulator problems with quadratic performance criteria. In the familiar unconstrained discrete linear regulator it is possible to derive the optimal controls directly from the minimization of the Hamiltonian by simply equating:

$$\left.\frac{\partial H}{\partial u}\right|_{*} = 0$$

where the * means that the expression is evaluated along the optimal trajectory.

The control so obtained is in terms of the associated costate variables p(n). However, this dependence on the costate variables is easily removed from the control law by using the fact that:

$$p^*(n) = k^*(n)\ x^*(n)$$

where k*(n) satisfies a difference equation of the Riccati type.

In the constrained case, however, it is no longer legitimate to assume that the minimizing controls are those that make:

$$\left.\frac{\partial H}{\partial u}\right|_* = 0$$

since the values of u that satisfy the equation above may not lay entirely within the specified bounds. In other words, when the control variables are constrained to lay within certain values, it is possible to encounter some saturation effects when minimizing the Hamiltonian.

Now, unfortunately, due to the logic involved in the determination of the total sales factor,

$$Ts_i(n) = \min\{[s_i(n) + k_i\ u_i(n)],\ d_i(n)\}$$

which forms part of the performance criteria of the problems, the structure of the costate equations becomes tremendously complicated. Consequently, any attempt at getting something out of the necessary conditions dictated by the Minimum Principle would most certainly lead to a fabulous mathematical entanglement. Therefore, due to the anticipated mathematical complication, the application of the Minimum Principle conditions to this problem will be just postulated as a possibility for further research.

Now, since both the state and the control variables of the problem are constrained to lay within a closed region in N+1 dimensional space:

$$\begin{matrix} 0 \leq s(n) \leq s^* \\ 0 \leq u(n) \leq u^* \end{matrix} \qquad n=0 \rightarrow N$$

The optimal control problems are naturally suited to be solved by means of dynamic programming. If this were not the case, the number of discrete controls that would have to be considered at each stage in the optimization would be infinitely large. However, having the control variables limited in this way, the dynamic programming problem becomes finite dimensional and therefore solvable.

The fact that the state variables are also bounded from both sides further reduces the dimensionality of the control problem in the sense that the set of allowable controls may be also influenced by the set of allowable states. That is, during the actual implementation of the dynamic programming algorithm it is only necessary to consider those controls that a) are inside the specified limits ($0 \leq u(n) \leq u^*$), and b) yield states that lay inside the specified region ($0 \leq s(n) \leq s^*$); for "n" running from zero to N.

Another characteristic of the subproblems that makes the implementation of a Dynamic Programming algorithm specially suitable is that all the state vectors are single dimensional. And, with the exception of the problem corresponding to plant A which has a two dimensional control vector, all the rest of the problems have a one-dimensional control vector. If this were not the case, the use of dynamic programming techniques for the solution of these problems would most probably be impractical. In

order to appreciate the significance of this argument, it is helpful to consider the fact that if the state variables are discretized to M levels and the state vector is n-dimensional, then the number of memory locations necessary for the construction of the initial grid would be

$$M^n \text{ per time stage}$$

Therefore, it is not difficult to see how the memory requirements can very easily get out of hand when $n > 1$. For instance, if time is discretized to 10 levels and the state vector is of third order with each component discretized to 50 different levels, then it would be necessary to have:

$$10(50)^3 = 1{,}250{,}000 \text{ memory locations}$$

At this point is is helpful to remember that, had the local problem not been decomposed the way it was, the implementation of the relatively simple dynamic programming algorithm for the solution of this problem would not have been possible. In other words, it is only thanks to decomposition allowed in the dual function that it is possible to solve the inner problem with such, up to a certain extent, brute force methods as the dynamic programming algorithm.

4. Maximization of the Dual Function

As shown in Appendix II, due to the discrete nature of the solutions given by the Dynamic Programming routine, the gradient of the dual function $h(\underline{c},\underline{p})$ is not everywhere continuous. In fact, as also explained in Appendix II, $h(\underline{c},\underline{p})$ is a concave shell formed by intersecting hyperplanes

in (N+1)-dimensional space. Therefore, it is not difficult to see why the implementation of a steepest ascent algorithm, or any other kind of algorithm for that matter, is not by any means trivial.

The starting point for the first cycle was arbitrarily chosen to be zero. And, since the chances of hitting a point where the gradient is discontinuous are really nill, the direction of steepest ascent for this cycle was simply assumed to be the direction of the gradient at the starting point, that is, the gradient at point "zero." Now, since the suboptimal point resulting from a linear search in any direction will in the general case lay on an extreme point of the function where the gradient is discontinuous, the selection of the direction of steepest ascent for cycles other than the first gets quite complicated.

As explained in detail in Appendix III, there are various ways of characterizing the direction of steepest ascent at the extreme points of the dual function, but for the development of the numerical results obtained, a suboptimal criterion for the selection of these directions was adopted. It was simply taken to be the geometrical mean between the gradients of the two hyperplanes whose intersection contained the extreme point in question. This method does not, by any means, pretend to be close to the optimum or anything of that sort, but since it is very easily implemented and it guarantees to yield an increase in the function (see Appendix III), it was adopted for the sake of simplicity.

As can be seen from the numerical results obtained with this method of approach, there is plenty of room for improvement in the determination of the search direction at the beginning of each cycle. For instance,

if the algorithm happens to get caught in a situation where one of the hyperplanes that define the extreme point is moderately steeped, but flanked by two highly steeped hyperplanes with nearly perpendicular gradients, then the resulting path will be a zig-zag upwards along the moderately steeped hyperplane. This seemingly pathological case does not seem to be at all uncommon since examining the numerical results obtained it can be observed that this type of situation occurred in more than one case, making the iterative procedure somewhat inefficient. However, the implementation of more sophisticated techniques for the determination of the steepest ascent direction at the extreme points of the function, as those discussed in Appendix III, will be just recommended as a possibility for further research.

Finally, to conclude the discussion of the algorithm, it is only left to explain the details of the type of linear search used.

After a direction of search was chosen, a linear search was conducted in order to find the maximum of the function in that direction.

First, a preliminary search to locate the maximum between two known points was performed. Then, using a linear interpolation technique, the interval of uncertainty was narrowed down to a pre-established limit or to the point where the maximum occurred.

Setting the direction of search to be the vector $\underline{v}$ and incorporating, for the sake of notational convenience, the vectors $\underline{c}$,$\underline{p}$ as a single vector $\underline{r}$, i.e.

$$\underline{r} = \begin{bmatrix} \underline{c} \\ \hline \underline{p} \end{bmatrix}$$

Then it is possible to write the dual function $h(\underline{c},\underline{p})$ as:

$$h(\underline{c},\underline{p}) = h(\underline{r})$$

Now, defining the size of the step along the given direction of search to be "a", it is possible to define a function of the variable "a" for a given search direction $\underline{v}$ such that

$$H(a) = h(\underline{r}_o + a\ \underline{v})$$

where $\underline{r}_o$ is a given point in the space. Clearly H(a) is a scalar function whose values correspond to the intersection of the dual function $h(\underline{r})$ with a plane passing through $\underline{r}_o$ and containing the vector $\underline{v}$.

Now the linear search that was mentioned before solves the problem of finding the smallest non-negative "a", called a*, for which the function H(a) attains a local maximum. In fact, since, as shown in Appendix II, the function H(a) is concave, this local maximum is indeed the global maximum of H(a).

The procedure consists of four main stages and it uses the derivative

$$\frac{dH}{da} = \left(\frac{\partial h}{\partial \underline{r}}\right)' \frac{\partial \underline{r}}{\partial a} \qquad \text{where} \quad \underline{r} = \underline{r}_o + a\ \underline{v}$$

$$\frac{\partial h}{\partial \underline{r}} = \underline{\nabla} h(\underline{r})$$

$$\frac{\partial \underline{r}}{\partial a} = \underline{v}$$

$$\frac{dH}{da} = (\underline{\nabla} h)'\ \underline{v}$$

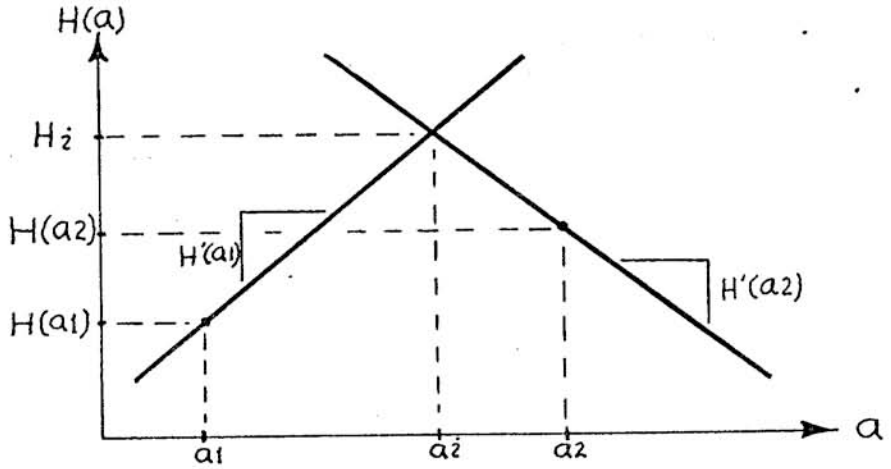
The first stage normalizes the $\underline{v}$ vector so that a step size a=1 is "acceptable." The second stage establishes bounds on a*, and the third stage interpolates linearly from the gathered data.

1st Stage: The vector $\underline{v}$ is normalized to unity length, i.e.

$$||\underline{v}|| = 1$$

2nd Stage: H(a) and dH/da are evaluated at points a=1,2,4,...,a1,a2 where a2 is the first of these points at which either H does not increase or dH/da becomes non-positive. It then follows that a* has to be bounded in the interval $a1 \leq a^* \leq a2$. If H(a2) is much smaller than H(a1), the function was evaluated at (2/3) a2 modifying the interval of uncertainty according to the values obtained.

3rd Stage: Using the values of the function and the slopes at a1 and a2 the function was interpolated linearly



$$H(a1) + H'(a1)(ai-a1) = H(a2) + H'(a2)(ai-a2)$$

$$ai = \frac{H(a1) - H(a2) + a2\ H'(a2) - a1\ H'(a1)}{H(a2) - H(a1)}$$

and $Hi = H(a1) + H'(a1)(ai-a1)$

4th Stage: H and dH/da is evaluated at a=ai. If

$$Hi - H(ai) \geq H^* \; ; \quad H^* \text{ arbitrary}$$

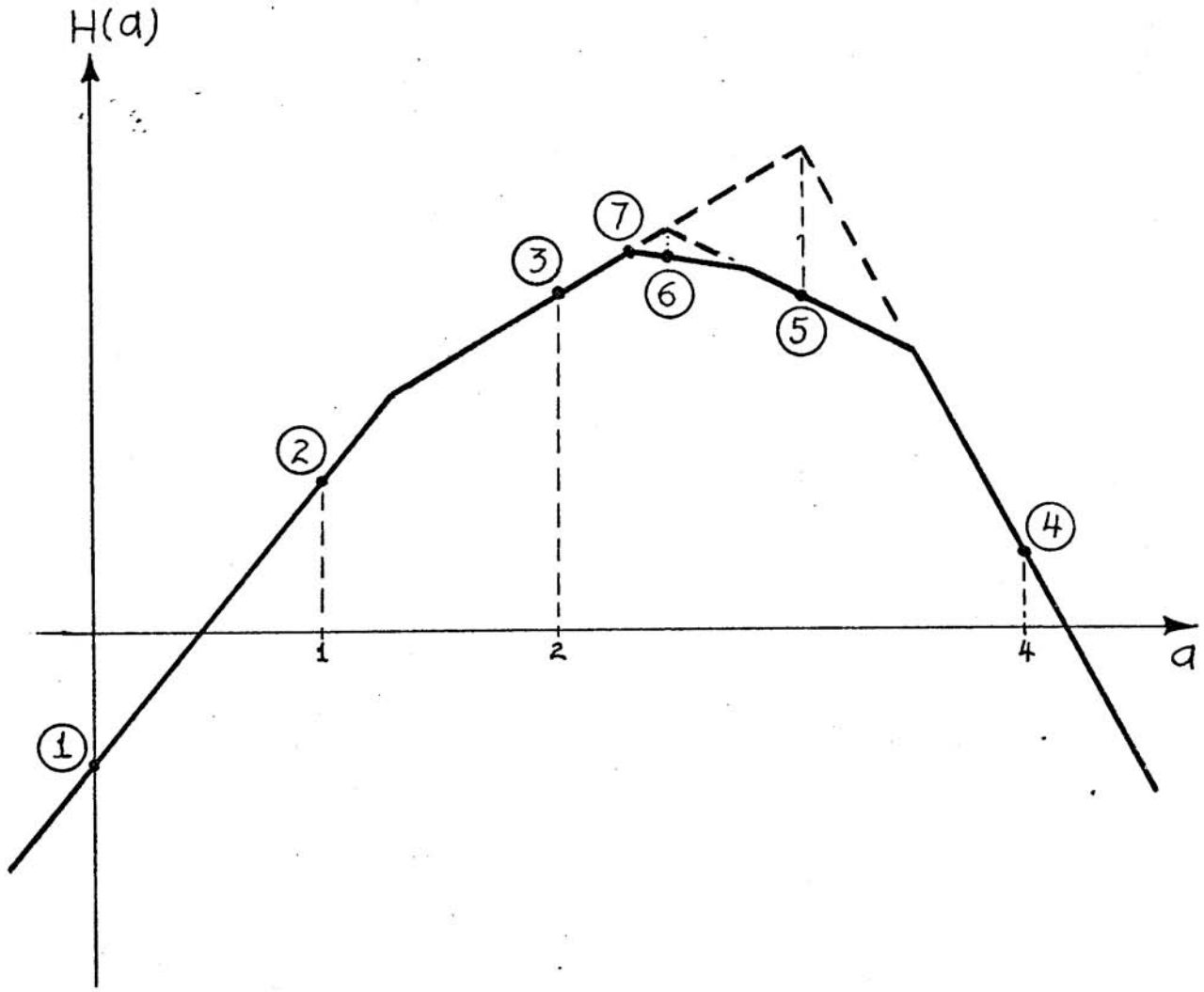
Then the interpolation is repeated between points a1 and ai or between ai and a2, depending on whether H'(ai) is negative or positive, respectively. If:

$$Hi - H(ai) \geq H^*$$

then the point resulting from the interpolation is accepted as the optimum.

The reason why a linear interpolation was chosen instead of a higher order one was because, as shown in Appendix II, the function $h(\underline{r})$ is a piecewise linear concave function in (N+1)-dimensional space. That is, it is formed by the intersection of a finite number of (N+1)-dimensional hyperplanes. Consequently, the scalar functions H(a) are also piecewise linear in their own 2-dimensional space. Obviously, the best interpolation technique is that that takes advantage of the linearity property of the function, that is, the linear interpolation technique.

In order to further illustrate the procedure of the linear search employed, following is a diagram that shows the sequence of points that would be evaluated in a hypothetical case where H(a) is given by the solid lines.


H(a)
a
1
2
4
1
2
3
4
5
6
7

CHAPTER 5

FORTRAN PROGRAM

A Fortran program that implements the algorithm described in the previous sections of this chapter was developed.

The program consists of a so-called main program and a number of auxilliary subroutines.

The main program may be considered the central manager. It controls the price of the intermediate product prices in such a way as to coordinate the optimization of the second plant subsystems. Using mathematical programming terminology, this part of the program is essentially a steepest ascent algorithm.

The linear search performed by the main program requires the solution of the subproblems (inner problem) and the resulting gradients at different points (different price vectors). This information is obtained by means of an auxilliary subroutine called the Subsystems Manager. This subroutine asks for the different schedules to the different subsystems (sequentially) given the specified price of the scarce materials. The actual subsystem optimization is done by a dynamic programming routine called the Subsystems Optimization routine. This last routine in turn makes use of three additional subroutines that calculate the intermediate and final stage costs and the inventory changes for each of the subdivisions.

Following is a listing of the sequential instructors of the actual program. The explanation of the variables used in the different routines is given in the comment lines included at the beginning of each routine.

```
C
C       SYSTEM'S MANAGER
C       METHOD-LINEAR INTERPOLATION
C
C       N IS THE TOTAL NUMBER OF SUBDIVISIONS.  KK IS THE NUMBER OF
C       TIME ELEMENTS. NW IS THE OUTPUT MACHINE LOGICAL CODE NUMBER. GT
C       IS THE GRADIENT TOLERANCE USED FOR STOPPING CRITERION. FT IS
C       THE MINIMUM CHANGE IN THE VALUE OF THE FUNCTION BETWEEN CYCLES
C       USED AS A STOPPING CRITERION.  NCMAX IS THE MAXIMUM NUMBER OF
C       CYCLES ALLOWED. C IS THE PRICES VECTOR. OT IS THE OPTIMAL
C       TRAJECTORIES MATRIX. OC IS THE OPT. CONTROLS MATRIX. DZ IS
C       THE FIRST PLANT DEMAND VECTOR. N IS THE TOTAL NUMBER OF
C       SUBDIVISIONS. LL IS THE NUMBER OF TIME INTERVALS.
C
      DIMENSION C(8), CH(8), GP(3), GNC(8), GN(8), GOPT(8),
     * OT(3,4), OC(3,4), SD(3), DZ(4)
      REAL MAG
      DATA N/3/, KK/4/
      K1 = KK + 1
      NW = 7
      GT = 0.1
      FT = 0.0005
      NSOL = 1
      NSMAX = 100
      NCMAX = 10
C
C     STARTING WITH ZERO COST VECTOR
C
      NC = 1
      WRITE (NW,203) NC
      KK2 = KK*2
      DO 1 K = 1,KK2
      C(K) = 0.0
    1 CONTINUE
      CALL SMGR (C,FI,GP,OT,OC,DZ)
      FP = FI
```

```
LINT  1
LINT  2
LINT  3
LINT  4
LINT  5
LINT  6
LINT  7
LINT  8
LINT  9
LINT 10
LINT 11
LINT 12
LINT 13
LINT 14
LINT 15
LINT 16
LINT 17
LINT 18
LINT 19
LINT 20
LINT 21
LINT 22
LINT 23
LINT 24
LINT 25
LINT 26
LINT 27
LINT 28
LINT 29
LINT 30
LINT 31
LINT 32
LINT 33
LINT 34
LINT 35
LINT 36
```

```
      DO 2 K = 1,KK2
      SD(K) = GP(K)
    2 CONTINUE
C
C     ROUTINE TO WRITE ALL PERTINENT INFORMATION AT THE
C     BEGINING OF A NEW CYCLE
C
    3 CONTINUE
      WRITE (NW,2[illegible]) FI
      WRITE (NW,230)
      WRITE (NW,218) (C(K) , K = 1,KK)
      WRITE (NW,231)
      WRITE (NW,218) (C(K), K = K1,KK2)
      WRITE (NW,232)
      DO 6 NS = 1,N
      WRITE (NW,233) (DT(NS,K),K = 1,KK)
      WRITE (NW,234) (DC(NS,K),K = 1,KK)
      IF (NS .EQ. 1) WRITE (NW,239) (DZ(K), K = 1,KK)
      IF (NS .EQ. N) GO TO 6
      WRITE (NW,236) NS
    6 CONTINUE
      WRITE (NW,201) (GP (K), K = 1,KK)
      WRITE (NW,220) (GP (K), K = K1,KK2)
C
C     ROUTINE TO STOP WHEN NONE OF THE "RELEVANT" COMPONENTS
C     OF THE GRADIENT ARE SUFFICIENTLY LARGE
C
    9 DO 12 K = 1,KK
      IF (SD(K) .GT. GT) GO TO 21
      IF (SD(K) .GT. -GT) GO TO 12
      IF (C(K) .GT. 0. ) GO TO 21
   12 CONTINUE
   15 DO 18 K = K1,KK2
      IF (ABS(SD(K)) .GT. GT) GO TO 21
   18 CONTINUE
      GO TO 117
```

```
C
C     NORMALIZATION AND ZEROING OF "USELESS" COMPONENTS OF THE
C     GRADIENT
C
   21 MAG = 0.0
      DO 27 K = 1,KK2
      GN(K) = SD(K)
      IF (K .GT. KK) GO TO 24
      IF (GN(K) .GT. 0.0) GO TO 24
      IF (C(K) .LE. 0.0) GN(K) = 0.0
   24 MAG = MAG + GN(K)**2
   27 CONTINUE
      MAG = MAG**0.5
      IF (MAG .LT. 1.0) GO TO 33
      DO 30 K = 1,KK2
      GN(K) = GN(K)/MAG
   30 CONTINUE
   33 WRITE (NW,221) (GN(K), K = 1,KK)
      WRITE (NW,222) (GN(K), K = K1,KK2)
      FPP = 0.0
      DO 36 K = 1,KK2
      FPP = FPP + GN(K)*GP(K)
   36 CONTINUE
      WRITE (NW,205)
      WRITE (NW,239) FPP
C
C     PRELIMINARY SEARCH
C
      KEY = 0
      AL = 0.0
      AR = 1.0
   39 CONTINUE
      DO 48 K = 1,KK
      CH(K) = C(K) + AR*GN(K)
      IF (CH(K)) 42,45,48
   42 AR = -C(K)/GN(K)
```

```
      KEY = 1                                                           LINT 108
      GO TO 39                                                          LINT 109
   45 IF (GN(K) .LT. 0.0) KEY = 1                                       LINT 110
   48 CONTINUE                                                          LINT 111
      DO 49 K = 1,KK                                                    LINT 112
      IF (CH(K) .GT. 0.0 + 1) GO TO 49                                  LINT 113
      IF (CH(K) .LT. -0.. -1)  GO TO 126                                LINT 114
      CH(K) = 0.0                                                       LINT 115
      IF (GN(K) .LT. 0.0) KEY = 1                                       LINT 116
   49 CONTINUE                                                          LINT 117
      DO 51 K = K1,KK2                                                  LINT 118
      CH(K) = C(K) + AR*GN(K)                                           LINT 119
   51 CONTINUE                                                          LINT 120
      CALL SMGR (CH, FN, GNG, CT, CC, DZ)                               LINT 121
      FPN = 0.0                                                         LINT 122
      DO 54 K = 1,KK2                                                   LINT 123
      FPN = FPN + GN(K)*GNG(K)                                          LINT 124
   54 CONTINUE                                                          LINT 125
      WRITE (NW,226) AR                                                 LINT 126
      WRITE (NW,229) FN                                                 LINT 127
      WRITE (NW,227)                                                    LINT 128
      WRITE (NW,239), FPN                                               LINT 129
C                                                                       LINT 130
C     ROUTINE TO REDUCE STEP SIZE WHEN THE VALUE OF F AT THE            LINT 131
C     NEW POINT IS MUCH SMALLER THAN AT THE PREVIOUS POINT              LINT 132
C                                                                       LINT 133
      IF (ABS(FP) .GT. ABS(FN)) GO TO 60                                LINT 134
   57 DC = ABS(FN)                                                      LINT 135
      GO TO 63                                                          LINT 136
   60 DC = ABS(FP)                                                      LINT 137
   63 IF (DC .EQ. 0.0) GO TO 66                                         LINT 138
      PD = (FP - FN)/DC                                                 LINT 139
      IF (PD .LE. 1..) GO TO 66                                         LINT 140
      AR = AR*(3.0/4.0)                                                 LINT 141
      KEY = 1                                                           LINT 142
      GO TO 39                                                          LINT 143
```

```
C
C     ROUTINE TO DECIDE WHETHER TO CONTINUE INCREASING
C     THE STEP SIZE OR NOT
C
   66 IF (FN .LE. FP) GO TO 78
      IF (FPN) 78,72,69
   69 IF (KEY .EQ. 1) GO TO 72
      AL = AP
      AR = 2.0*AR
      FP = FN
      FPP = FPN
      DO 71 K = 1,KK2
      GP(K) = GNG(K)
   71 CONTINUE
      GO TO 39
   72 FOPT = FN
      FP = FN
      DO 75 K = 1,KK2
      GP(K) = GNG(K)
      GOPT(K) = GNG(K)
      SD(K) = GNG(K)
   75 CONTINUE
      GO TO 111
C
C     LINEAR INTERPOLATION
C
   78 CONTINUE
      NSOL = NSOL + 1
      IF (NSOL .GE. NSMAX) GO TO 13
      AI = (FP - FN + AR*FPN - AL*FPP)/(FPN - FPP)
      FLI = FP + FPP*(AI - AL)
      WRITE (NW,206) AL,AR
      WRITE (NW,240) AI
      WRITE (NW,237)
      WRITE (NW,235) FLI
      DO 81 K = 1,KK2
```

```
      CH(K) = C(K) + AI*GN(K)                                          LINT 131
   81 CONTINUE                                                         LINT 132
      CALL SMGR (CH, FOPT, GOPT, UT, UC, DZ)                           LINT 133
      FPI = 0.0                                                        LINT 134
      DO 84 K = 1,KK2                                                  LINT 135
      FPI = FPI + GOPT(K)*GN(K)                                        LINT 136
   84 CONTINUE                                                         LINT 137
      WRITE (NW,226) AI                                                LINT 138
      WRITE (NW,229) FOPT                                              LINT 139
      WRITE (NW,227)                                                   LINT 140
      WRITE (NW,239) FPI                                               LINT 141
      KEY2 = 0.0                                                       LINT 142
C                                                                      LINT 143
C     ROUTINE TO DECIDE WHETHER TO CONTINUE NARROWING THE              LINT 144
C     INTERVAL OR NOT                                                  LINT 145
C                                                                      LINT 146
      IF (ABS(FLI - FOPT) .LE. FT) KEY2 = 1                            LINT 147
      IF (FPI) 93,99,87                                                LINT 148
   87 FP = FOPT                                                        LINT 149
      FPP = FPI                                                        LINT 150
      DO 90 K = 1,KK2                                                  LINT 151
      GP(K) = GOPT(K)                                                  LINT 152
   90 CONTINUE                                                         LINT 153
      AL = AI                                                          LINT 154
      IF (KEY2 .EQ. 1) GO TO 105                                       LINT 155
      GO TO 78                                                         LINT 156
   93 FN = FOPT                                                        LINT 157
      FPN = FPI                                                        LINT 158
      DO 96 K = 1,KK2                                                  LINT 159
      GNG(K) = GOPT(K)                                                 LINT 160
   96 CONTINUE                                                         LINT 161
      AR = AI                                                          LINT 162
      IF (KEY2 .EQ. 1) GO TO 105                                       LINT 163
      GO TO 78                                                         LINT 164
C                                                                      LINT 165
C     INITIALIZATION FOR A NEW CYCLE                                   LINT 166
```

```
C
   99  FP = FOPT                                                          LINT 217
       DO 102 K = 1,KK2                                                   LINT0218
       GP(K) = GOPT(K)                                                    LINT 219
       SD(K) = GOPT(K)                                                    LINT0220
  102  CONTINUE                                                           LINT0221
       GO TO 111                                                          LINT0222
  105  FP = FOPT                                                          LINT0223
       MAG1 = 0.0                                                         LINT0224
       MAG2 = 0.0                                                         LINT0225
       DO 106 K = 1,KK2                                                   LINT0226
       MAG1 = MAG1 + GP(K)**2                                             LINT0227
       MAG2 = MAG2 + GNG(K)**2                                            LINT 228
  106  CONTINUE                                                           LINT 229
       MAG1 = MAG1**0.5                                                   LINT0230
       MAG2 = MAG2**0.5                                                   LINT0231
       DO 108 K = 1,KK2                                                   LINT0232
       SD(K) = (GP(K)/MAG1 + GNG(K)/MAG2)/2.0                             LINT0233
       GP(K) = GOPT(K)                                                    LINT0234
  108  CONTINUE                                                           LINT0235
  111  DO 114 K = 1,KK2                                                   LINT0236
       C(K) = CH(K)                                                       LINT0237
  114  CONTINUE                                                           LINT0238
C                                                                         LINT0239
C      ROUTINE TO DECIDE WHETHER TO CONTINUE WITH A NEW                   LINT0240
C      CYCLE OR NOT                                                       LINT0241
C                                                                         LINT 242
       IF ((FP - FI) .LT. FT) GO TO 117                                   LINT0243
       FI = FP                                                            LINT0244
       NC = NC + 1                                                        LINT0245
       IF (NC .GT. NCMAX) GO TO 117                                       LINT0246
       WRITE (NW,203) NC                                                  LINT 247
       GO TO 3                                                            LINT0248
C                                                                         LINT 249
C      ROUTINE TO PRINTOUT THE FINAL OPTIMAL TRAJECTORIES AND             LINT 25
C      CONTROLS                                                           LINT 251
                                                                          LINT 252
```

```
C
 117  WRITE (NW,207) FOPT                                                    LINT 253
      WRITE (NW,208)                                                         LINT 254
      WRITE (NW,209) (C(K), K = 1,KK)                                        LINT 255
      WRITE (NW,210)                                                         LINT 256
      WRITE (NW,209) (GOPT(K), K = 1,KK)                                     LINT 257
      WRITE (NW,223)                                                         LINT 258
      WRITE (NW,209) (C(K), K = K1,KK2)                                      LINT 259
      WRITE (NW,224)                                                         LINT 260
      WRITE (NW,225)                                                         LINT 261
      WRITE (NW,209) (GOPT(K), K = K1,KK2)                                   LINT 262
      WRITE (NW,211)                                                         LINT 263
      WRITE (NW,212)                                                         LINT 264
      WRITE (NW,209) (OT(1,K), K = 1,KK)                                     LINT 265
      WRITE (NW,213)                                                         LINT 266
      NM1 = N - 1                                                            LINT 267
      DO 123 I = 1,NM1                                                       LINT 268
      WRITE (NW,214) I                                                       LINT 269
      WRITE (NW,209) (OT(I+1,K), K = 1,KK)                                   LINT 270
 123  CONTINUE                                                               LINT 271
      WRITE (NW,215)                                                         LINT 272
      WRITE (NW,212)                                                         LINT 273
      WRITE (NW,209) (OC(1,K), K = 1,KK)                                     LINT 274
      WRITE (NW,213)                                                         LINT 275
      DO 123 I = 1,NM1                                                       LINT 276
      WRITE (NW,214) I                                                       LINT 277
      WRITE (NW,209) (OC(I+1,K), K = 1,KK)                                   LINT 278
 123  CONTINUE                                                               LINT 279
      GO TO 130                                                              LINT 280
 126  WRITE (NW,204)                                                         LINT 281
 130  STOP                                                                   LINT 282
 207  FORMAT (//,5X,'VALUE OF DUAL FUNCTION =  ',E14.7)                      LINT 283
 208  FORMAT(///,5X,'GRADIENT ',8(E14.4))                                    LINT 284
 209  FORMAT (/////,5X,'CYCLE # ',I2)                                        LINT 285
 204  FORMAT (//,5X,'A COMPONENT OF THE COST VECTOR BECAME NEGATIVE')        LINT 286
 205  FORMAT (//,5X,'SLOPE OF DUAL FUNCTION')                                LINT 287
```

```
206  FORMAT (///,5X,'STEP SIZE LIMITS FOR LINEAR INTERPOLATION ARE',
     * F7.3,2X,'AND',F7.3)
207  FORMAT (1H1,4X,'MINIMUM OPERATING COST = ',E14.7)
208  FORMAT (//,5X,'OPTIMUM INTERMEDIATE PRODUCT COSTS'/)
209  FORMAT (5X,10(E13.5))
210  FORMAT (//,5X,'EXCESS DEMANDS'/)
211  FORMAT (1H1,16X,'OPTIMUM INVENTORY SCHEDULES')
212  FORMAT (/,26X,'PLANT 1'/)
213  FORMAT (//,20X,'PLANT 2'/)
214  FORMAT (//,7X,'FINISHED PRODUCT # ',I2/)
215  FORMAT (1H1,16X,'OPTIMUM PRODUCTION SCHEDULES')
216  FORMAT (1X,8(E14.4))
220  FORMAT (14X,8(E14.4))
221  FORMAT (/,5X,'SEARCH',3X,8(E14.4))
222  FORMAT (5X,'DIRECTION',8(E14.4))
223  FORMAT (//,5X,'OPTIMUM INTERMEDIATE PRODUCT DEMAND PRICES'/)
224  FORMAT (//,5X,'EXCESS DEMANDS WITH RESPECT TO THE')
225  FORMAT (5X,'OPTIMUM DEMANDS FOR PLANT ONE'/)
226  FORMAT (///,5X,'STEP SIZE "A" ALONG SEARCH DIRECTION =',F7.3/)
227  FORMAT ( /,5X,'SLOPE OF DUAL FUNCTION')
229  FORMAT (5X,'VALUE OF DUAL FUNCTION =  ',E14.7)
230  FORMAT (//,5X,'INTERMEDIATE PRODUCT PRICES' )
231  FORMAT ( /,5X,'PLANT 1 DEMAND PRICES' )
232  FORMAT (///,5X,'SCHEDULES FOR PLANT 1 '/)
233  FORMAT (5X,'INVENTORIES',E12.4,7(E14.4))
234  FORMAT (5X,'PRODUCTIONS',E12.4,7(E14.4))
235  FORMAT (5X,'DEMANDS',4X,F12.4,7(E14.4))
236  FORMAT (//,5X,'SCHEDULES FOR PLANT 2  SUBDIVISION # ',I2/)
237  FORMAT (/,5X,'LINEARLY INTERPOLATED')
238  FORMAT (5X,'VALUE OF THE FUNCTION  =  ',E14.7)
239  FORMAT (5X,'ALONG SEARCH DIRECTION =  ',E11.4)
240  FORMAT (/,5X,'OPTIMUM STEP SIZE =',F7.3)
     END
```

```
      SUBROUTINE SMGR (C, TCOST, G, OT, OC, DZ)
C
C     SUBSYSTEMS MANAGER ROUTINE
C
C     G IS THE GRADIENT VECTOR. OC IS THE OPT. CONTROL MATRIX.
C     CT IS THE OPT. TRAJECTORIES MATRIX. PE IS THE PRODUCTION
C     EFFICIENCIES VECTOR. DDZ IS THE FIRST PLANT DEMAND VECTOR.
C     N IS THE TOTAL NUMBER OF SUBDIVISIONS. KK IS THE NUMBER OF
C     TIME INTERVALS. LL IS THE NUMBER OF DEMAND DISCRETE LEVELS.
C     NW IS THE OUTPUT SOURCE LOGICAL NUMBER.
C
      DIMENSION OTV(4), OCV(4), COST(3), G(3), OC(3,4), OT(3,4),
     * C(8), DEMAND(3,4), PE(3), P(4), DDZ(4), DZ(4),SDD(4)
      DATA N/3/, KK/4/, LL/6/
      DATA PE/ 0.9, 0.7, 0.8/
      DATA (DEMAND(2,K), K = 1,KK)/ 15.0, 15.0, 37.5, 15.0/
      DATA (DEMAND(3,K), K = 1,KK)/17.5, 17.5, 43.75, 17.5/
      DATA DZMAX/65.0/
      KK2 = 2.0*KK
      K1 = KK + 1
      KEY = 1
      D3 = LL
      NW = 7
      DO 3 K = 1,KK
      P(K) = C(KK + K)
    3 CONTINUE
      NS = 1
      CALL SOPT (NS,C,P,DEMAND,CTV,OCV,DDZ,COST(NS))
      DO 6 K = 1,KK
      OT(NS,K) = OTV(K)
      OC(NS,K) = OCV(K)
      DZ(K) = DDZ(K)
    6 CONTINUE
      DO 10 NS = 2,N
      CALL SOPT (NS,C,P,DEMAND,CTV,OCV,DDZ,COST(NS))
      DO 10 K = 1,KK
```

```
      DT (NS,K) = DTV(K)
      DC(NS,K) = DCV(K)
   10 CONTINUE
      DO 18 K = 1,KK
      SOD(K) = 0.0
      DO 14 NS = 2,N
      SOD(K) = SOD(K) + DC(NS,K)
   14 CONTINUE
      G(K) = SOD(K) - DT(1,K) - DC(1,K)*PE(1)
      IF (KEY .GT. 0) GO TO 17
      D = 0.0
      SODN = SOD(K) - 0.5*DZMAX/(D3 - 1.0)
      DO 16 I = 1,LL
      D = D + DZMAX/(D3 - 1.0)
      IF (D .LE. SODN) GO TO 16
      SOD(K) = D
      GO TO 17
   16 CONTINUE
   17 G(K + KK) = SOD(K) - DZ(K)
   18 CONTINUE
      TCOST = 0.0
      DO 40 I = 1,N
      TCOST = TCOST + COST(I)
   40 CONTINUE
C
C     ROUTINE TO PRINT THE OPTIMAL TRAJECTORIES AND CONTROLS OF THE
C     INTERMEDIATE STAGES OF EACH CYCLE. IT IS ACTIVATED BY
C     REMOVING THE  GO TO 53  CARD
C
      GO TO 53
      WRITE (NW,63) (C(K),K = 1,KK)
      WRITE (NW,66) (C(K), K = K1,KK2)
      WRITE (NW,64) TCOST
      WRITE (NW,60)
      DO 50 I = 1,N
      WRITE (NW,61) (DT(I,K), K = 1,KK)
```

```
      WRITE (NW,62) (CC(I,K), K = 1,KK)
      IF (I .EQ. 1) WRITE (NW,62) (DZ(K),K = 1,KK)
   50 CONTINUE
      WRITE (NW,65) (G(K), K = 1,KK)
      WRITE (NW,66) (G(K), K = K1,KK2)
   53 CONTINUE
      RETURN
   60 FORMAT (//,5X,'OPTIMAL TRAJECTORY / OPTIMAL CONTROLS',
     * ' / (OPTIMAL DEMANDS)')
   61 FORMAT (/,3X,10(E12.4))
   62 FORMAT (3X,10(E12.4))
   63 FORMAT (//,5X,'COST VECTOR',8(E12.4))
   64 FORMAT (//,5X,'FUNCTION =  ',E14.7)
   65 FORMAT (//,5X,'GRADIENT    ',8(E12.4))
   66 FORMAT (16X,8(E12.4))
      END
```

```
      SUBROUTINE SOPT(NS,C,P,DEMAND,OT,OC,ODZ,OCOST)                    SOPT0001
C                                                                       SOPT0002
C     SUBSYSTEM OPTIMIZATION ROUTINE                                    SOPT0003
C     METHOD- DYNAMIC PROGRAMMING                                       SOPT0004
C                                                                       SOPT0005
C     JCOST,UOPT AND XK1OPT ARE THE COSTS, CONTROLS AND X(K+1)'S        SOPT0006
C     RESPECTIVELY, THAT FORM THE PRELIMINARY GRID. NS IS THE           SOPT0007
C     NUMBER OF THE SUBDIVISION. OCOST IS THE OPTIMAL COST.             SOPT0008
C     OT IS THE OPTIMAL TRAJECTORY VECTOR. OC IS THE OPTIMAL            SOPT0009
C     CONTROLS VECTOR. ODZ IS THE OPTIMAL DEMANDS VECTOR FOR THE        SOPT0010
C     INTERMEDIATE PRODUCT. II IS THE NUMBER OF DISCRETE LEVELS OF      SOPT0011
C     THE STATE. JJ IS THE NUMBER OF DISCRETE LEVELS OF THE CONTROL.    SOPT0012
C     KK IS THE NUMBER OF TIME INTERVALS. LL IS THE NUMBER OF DISCRETE  SOPT0013
C     LEVELS OF THE FIRST PLANT DEMAND. MINV IS THE MAXIMUM             SOPT0014
C     INVENTORIES VECTOR. MPROD IS THE MAXIMUM PRODUCTIONS VECTOR.      SOPT0015
C     DZMAX IS THE MAXIMUM PLANT 1 DEMAND                               SOPT0016
C                                                                       SOPT0017
C     IF THE INITIAL GRID, OPT. TRAJECTORIES AND OPT. CONTROLS OF       SOPT0018
C     EACH OPTIMIZATION ARE WANTED,THE CONSTANT  KW  DEFINED BELOW      SOPT0019
C     SHOULD BE GIVEN THE VALUE OF ZERO                                 SOPT0020
C                                                                       SOPT0021
      DIMENSION C(8), P(4), S(4), U(11), X(11), DZ(11),                 SOPT0022
     * UOPT(11,4), XK1OPT(11,4), DZOPT(11,4), OT(4), OC(4),             SOPT0023
     * ODZ(4), DEMAND(3,4)                                              SOPT0024
      REAL JCOST(11,4), MINV(3), MPROD(3)                               SOPT0025
      DATA II/6/, JJ/6/, KK/4/, LL/6/                                   SOPT0026
      DATA MINV/100.0, 40.0, 60.0/                                      SOPT0027
      DATA MPROD/ 50.0, 30.0, 35.0/                                     SOPT0028
      DATA DZMAX/65.0/                                                  SOPT0029
      NW = 0                                                            SOPT0030
      D1 = II                                                           SOPT0031
      D2 = JJ                                                           SOPT0032
      D3 = LL                                                           SOPT0033
      KW = 1                                                            SOPT0034
      IF (NS .EQ. 1) GO TO 4                                            SOPT0035
      DO 3 K = 1,KK                                                     SOPT0036
```

```
      D(K) = DEMAND (NS,K)                                                  SOPT  37
    3 CONTINUE                                                              SOPT  38
    4 XMAX = MINV(NS)                                                       SOPT  39
      UMAX = MPRCD(NS)                                                      SOPT  40
C                                                                           SOPT  41
C     COMPUTATION OF THE STATE AND CONTROL DISCRETIZED LEVELS               SOPT  42
C                                                                           SOPT  43
      X(1) = 0.0                                                            SOPT  44
      DO 6 I = 2,II                                                         SOPT  45
    6 X(I) = XMAX/(D1 - 1.0) + X(I-1)                                       SOPT  46
      U(1) = 0.0                                                            SOPT  47
      DO 9 J = 2,JJ                                                         SOPT  48
    9 U(J) = U(J-1) + UMAX/(D2 - 1.0)                                       SOPT  49
      DZ(1) = 0.0                                                           SOPT  50
      DO 12 L = 2,LL                                                        SOPT  51
   12 DZ(L) = DZ(L-1) + DZMAX/(D3 - 1.0)                                    SOPT  52
C                                                                           SOPT  53
C     START OF THE ALGORITHM                                                SOPT  54
C                                                                           SOPT  55
      K = KK                                                                SOPT  56
      IF (NS .GT. 1) GO TO 21                                               SOPT  57
      DO 18 I = 1,II                                                        SOPT  58
      KEY = 0                                                               SOPT  59
      DO 18 J = 1,JJ                                                        SOPT  60
      DO 18 L = 1,LL                                                        SOPT  61
      XK1 = F (NS,X(I),U(J),DZ(L))                                          SOPT  62
      IF (XK1 .GT. XMAX) GO TO 18                                           SOPT  63
      IF (XK1 .LT. 0.0) XK1 = 0.0                                           SOPT  64
      COST = TC (NS,X(I),XK1,U(J),DZ(L),C(K),P(K))                          SOPT  65
      IF (KEY .EQ. 0) GO TO 15                                              SOPT  66
      IF (COST .GE. JCOST(I,K)) GO TO 18                                    SOPT  67
   15 JCOST(I,K) = COST                                                     SOPT  68
      XK1OPT(I,K) = XK1                                                     SOPT  69
      UOPT(I,K) = U(J)                                                      SOPT  70
      DZOPT(I,K) = DZ(L)                                                    SOPT  71
   18 CONTINUE                                                              SOPT  72
```

```
      GO TO 23                                                          SOPT  073
   21 DO 27 I = 1,II                                                    SOPT  074
      DO 27 J = 1,JJ                                                    SOPT  075
      XK1 = F (NS,X(I),U(J),D(K))                                       SOPT  076
      IF (XK1 .GT. XMAX) GO TO 27                                       SOPT  077
      IF (XK1 .LT. 0.0) XK1 = 0.0                                       SOPT  078
      COST = TC (NS,X(I),XK1,U(J),D(K),C(K),P(K))                       SOPT  079
      IF (J .EQ. 1) GO TO 24                                            SOPT  080
      IF (COST .GE. JCOST(I,K)) GO TO 27                                SOPT  081
   24 JCOST(I,K) = COST                                                 SOPT  082
      XK1OPT(I,K) = XK1                                                 SOPT  083
      UOPT(I,K) = U(J)                                                  SOPT  084
   27 CONTINUE                                                          SOPT  085
C                                                                       SOPT  086
C     ITERATIVE PROCESS BACKWARDS IN TIME                               SOPT  087
C                                                                       SOPT  088
   28 K1 = KK - 1                                                       SOPT  089
      IF (NS .GT. 1) GO TO 41                                           SOPT  090
      DO 39 N = 1,K1                                                    SOPT  091
      K = KK - N                                                        SOPT  092
      DO 39 I = 1,II                                                    SOPT  093
      KEY = 0                                                           SOPT  094
      DO 39 J = 1,JJ                                                    SOPT  095
      DO 39 L = 1,LL                                                    SOPT  096
      XK1 = F (NS,X(I),U(J),DZ(L))                                      SOPT  097
      IF (XK1 .GT. XMAX) GO TO 39                                       SOPT  098
      IF (XK1 .LT. 0.0) XK1 = 0.0                                       SOPT  099
      XK1N = XK1 - 0.5*XMAX/(DI - 1.0)                                  SOPT  100
      DO 30 M = 1,II                                                    SOPT  101
      IF (X(M) .LE. XK1N) GO TO 30                                      SOPT  102
      IX = M                                                            SOPT  103
      GO TO 33                                                          SOPT  104
   30 CONTINUE                                                          SOPT  105
   33 COST = H (NS,X(I),U(J),UOPT(IX,K+1),C(K),P(K),DZ(L))              SOPT  106
     * + JCOST(IX,K+1)                                                  SOPT  107
      IF (KEY .EQ. 0) GO TO 36                                          SOPT  108
```

```
      IF (COST .GE. JCOST(I,K)) GO TO 39
   36 JCOST(I,K) = COST
      XK1OPT(I,K) = XK1
      UOPT(I,K) = U(J)
      DZOPT(I,K) = DZ(L)
   39 CONTINUE
      GO TO 52
   41 DO 50 N = 1,K1
      K = KK - N
      DO 50 I = 1,II
      DO 50 J = 1,JJ
      XK1 = H(NS,X(I),U(J),D(K))
      IF(XK1 .GT. XMAX) GO TO 50
      IF (XK1 .LT. 0.0) XK1 = 0.0
      XK1N = XK1 - 0.5*XMAX/(01 - 1.0)
      DO 43 M = 1,II
      IF (X(M) .LE. XK1N) GO TO 43
      IX = M
      GO TO 45
   43 CONTINUE
   45 COST = H (NS,X(I),U(J),UOPT(IX,K+1),C(K),P(K),D(K))
     * + JCOST(IX,K+1)
      IF (J .EQ. 1) GO TO 48
      IF (COST .GE. JCOST(I,K)) GO TO 50
   48 JCOST(I,K) = COST
      XK1OPT(I,K) = XK1
      UOPT(I,K) = U(J)
   50 CONTINUE
C
C     GRID PRINTING OUT ROUTINE
C
   52 IF (KW .GT. 0) GO TO 54
      IF (NS .GT. 1) GO TO 53
      WRITE (NW,115)
      GO TO 55
   53 WRITE (NW,116)
```

```
SOPT0 1-9
SOPT  11
SOPT  111
SOPT  112
SOPT  113
SOPT  114
SOPT  115
SOPT  116
SOPT  117
SOPT  118
SOPT  119
SOPT  120
SOPT  121
SOPT  122
SOPT  123
SOPT  124
SOPT  125
SOPT  126
SOPT  127
SOPT  128
SOPT  129
SOPT  130
SOPT  131
SOPT  132
SOPT  133
SOPT  134
SOPT  135
SOPT  136
SOPT  137
SOPT  138
SOPT  139
SOPT  140
SOPT  141
SOPT  142
SOPT  143
SOPT  144
```

```
   55  DO 56 I = 1,II                                                    SOPT 145
       WRITE (NW,110) X(I)                                               SOPT 146
       WRITE (NW,111) (XK1OPT(I,K), K = 1,KK)                            SOPT 147
       WRITE (NW,111) (JCOST(I,K), K = 1,KK)                             SOPT 148
       WRITE (NW,111) (UOPT(I,K), K = 1,KK)                              SOPT 149
       IF (NS .EQ. 1) WRITE (NW,111) (DZOPT(I,K), K = 1,KK)              SOPT 150
   56  CONTINUE                                                          SOPT 151
C                                                                        SOPT 152
C      PROCESSING OF THE GRID                                            SOPT 153
C                                                                        SOPT 154
   58  IX = 1                                                            SOPT 155
       OCOST = JCOST(1,1)                                                SOPT 156
       DO 60 I = 2,II                                                    SOPT 157
       IF (JCOST(I,1) .GE. OCOST) GO TO 60                               SOPT 158
       OCOST = JCOST (I,1)                                               SOPT 159
       IX = I                                                            SOPT 160
   60  CONTINUE                                                          SOPT 161
       OT (1) = X(IX)                                                    SOPT 162
       OC(1) = UOPT (IX,1)                                               SOPT 163
       IF (NS .EQ. 1) GO TO 69                                           SOPT 164
       DO 64 K = 2,KK                                                    SOPT 165
       XK1N = XK1OPT (IX,K-1) - 0.5*XMAX/(D1 - 1.0)                      SOPT 166
       DO 62 I = 1,II                                                    SOPT 167
       IF (X(I) .GT. XK1N) GO TO 63                                      SOPT 168
   62  CONTINUE                                                          SOPT 169
   63  IX = I                                                            SOPT 170
       IF (IX .GT. II) IX = II                                           SOPT 171
       OT(K) = X(IX)                                                     SOPT 172
       OC (K) = UOPT (IX,K)                                              SOPT 173
   64  CONTINUE                                                          SOPT 174
       GO TO 81                                                          SOPT 175
   69  OOZ(1) = DZOPT(IX,1)                                              SOPT 176
       DO 78 K = 2,KK                                                    SOPT 177
       XK1N = XK1OPT(IX,K-1) - 0.5*XMAX/(D1 - 1.0)                       SOPT 178
       DO 72 I = 1,II                                                    SOPT 179
       IF (X(I) .LE. XK1N) GO TO 72                                      SOPT 180
```

```
      IX = I                                                            SOPT 181
      GO TO 75                                                          SOPT 182
   72 CONTINUE                                                          SOPT 183
   75 OT(K) = X(IX)                                                     SOPT 184
      OC(K) = UOPT(IX,K)                                                SOPT 185
      OOZ(K) = DZOPT(IX,K)                                              SOPT 186
   76 CONTINUE                                                          SOPT 187
   81 IF (KK .NE. 0) GO TO 84                                           SOPT 188
      WRITE(NW,112) OCOST                                               SOPT 189
      WRITE(NW,113)                                                     SOPT 190
      WRITE(NW,111) (OT(K), K = 1,KK)                                   SOPT 191
      WRITE(NW,114)                                                     SOPT 192
      WRITE(NW,111) (OC(K), K = 1,KK)                                   SOPT 193
      IF (NS .GT. 1) GO TO 84                                           SOPT 194
      WRITE (NW,117)                                                    SOPT 195
      WRITE (NW,111) (OOZ(K), K = 1,KK)                                 SOPT 196
   84 CONTINUE                                                          SOPT 197
      RETURN                                                            SOPT 198
  110 FORMAT (/,4X,'X =',E15.5)                                         SOPT 199
  111 FORMAT (1X,10(E13.5))                                             SOPT 200
  112 FORMAT (1H1,2X,'OPTIMAL COST',F12.4//)                            SOPT 201
  113 FORMAT (//,3X,'OPTIMAL TRAJECTORY'/)                              SOPT 202
  114 FORMAT (//,3X,'OPTIMAL CONTROLS'/)                                SOPT 203
  115 FORMAT (1HC,4X,'XK1OPT / JOCST / UOPT / DZOPT')                   SOPT 204
  116 FORMAT (1HC,4X,'XK1OPT / JOCST / UOPT')                           SOPT 205
  117 FORMAT (//,3X,'OPTIMAL DEMANDS'/)                                 SOPT 206
      END                                                               SOPT 207
```

```
      FUNCTION F (NS,X,U,D)
C
C     DYNAMIC EQUATION
C
C     NS IS THE NUMBER OF THE SUBDIVISION. PE IS THE PRODUCTION
C     EFFICIENCIES VECTOR. X IS THE STATE. U IS THE APPLIED CONTROL
C     AND D IS THE DEMAND
C
      DIMENSION PE(3)
      DATA PE/0.9, 0.7, 0.8/
      F = X + PE(NS)*U - D
      RETURN
      END
```

```
      FUNCTION TC (NS,X,XK1,U,C,C,P)
C
C     TERMINAL COST EQUATION
C
C     NS IS THE SUBDIVISION NUMBER. X IS THE STATE VALUE. XK1 IS THE
C     NEXT STATE VALUE. U IS THE APPLIED CONTROL. D IS THE DEMAND.
C     C IS THE PRICE OF THE INTERMEDIATE PRODUCT. P IS THE PRICE
C     ASSOCIATED WITH THE DEMAND OF THE INTERMEDIATE PRODUCT. PE IS
C     THE PRODUCTION EFFICIENCIES VECTOR. IC IS THE INVENTORY COSTS
C     VECTOR. NP IS THE MARGINAL PROFITS VECTOR.
C
      DIMENSION PE(3)
      REAL IC(3), NP(3)
      DATA PE/0.9, 0.7, 0.8/
      DATA IC/ 0.02, 0.1 , 0.2/
      DATA NP/ 0.0, 5.0, 3.5/
      IF (NS.GT. 1) GO TO 5
      TC = IC(1)*(X**2 + XK1**2) - C*(PE(1)*U + X) - P*D
      RETURN
    5 TI = X + U*PE(NS)
      IF (TI .LT. 0) GO TO 10
      SP = NP(NS)*D
      GO TO 15
   10 SP = NP(NS)*TI
   15 TC = IC(NS)*(X**2 + XK1**2) - SP + (C + P)*U
      RETURN
      END
```

```
TCO     1
TCO     2
TCO     3
TCO     4
TCO     5
TCO     6
TCO     7
TCO     8
TCO     9
TCO    10
TCO    11
TCO    12
TCO    13
TCO    14
TCO    15
TCO    16
TCO    17
TCO    18
TCO    19
TCO    20
TCO    21
TCO    22
TCO    23
TCO    24
TCO    25
TCO    26
TCO    27
```

```
      FUNCTION H(NS,X,U,UK1,C,P,D)                                      RGCE0001
C                                                                       RGCE0002
C     REGULAR COST EQUATION                                             RGCE0003
C                                                                       RGCE0004
C     NS IS THE SUBDIVISION NUMBER. X IS THE PRESENT STATE. U IS        RGCE0005
C     THE PRESENTLY APPLIED CONTROL. UK1 IS THE CONTROL THAT WILL       RGCE0006
C     BE APPLIED AT THE NEXT TIME STAGE. C IS THE INTERMEDIATE          RGCE0007
C     PRODUCT PRICE. P IS THE PRICE ASSOCIATED WITH THE INTERMEDIATE    RGCE0008
C     PRODUCT DEMAND. IC IS THE INVENTORY COSTS VECTOR. NP IS THE       RGCE0009
C     MARGINAL PROFITS VECTOR. CHPC IS THE CHANGE IN PRODUCTION         RGCE0010
C     COSTS VECTOR.                                                     RGCE0011
C                                                                       RGCE0012
      DIMENSION CHPC(3), PE(3)                                          RGCE0013
      REAL IC(3), NP(3)                                                 RGCE0014
      DATA CHPC/ 0.1, 0.1, 0.1/                                         RGCE0015
      DATA PE/0.9, 0.7, 0.8/                                            RGCE0016
      DATA NP/ 6.0, 5.0, 3.5/                                           RGCE0017
      DATA IC/ 0.02, 0.13, 0.2/                                         RGCE0018
      IF (NS .GT. 1) GO TO 5                                            RGCE0019
      H = IC(1)*X**2 + CHPC(1)*(U - UK1)**2 - C*(PE(1)*U + X) - P*D     RGCE0020
      RETURN                                                            RGCE0021
    5 TI = X + U*PE(NS)                                                 RGCE0022
      IF (TI .LT. 0) GO TO 10                                           RGCE0023
      SP = NP(NS)*D                                                     RGCE0024
      GO TO 15                                                          RGCE0025
   10 SP = NP(NS)*TI                                                    RGCE0026
   15 H = IC(NS)*X**2 + CHPC(NS)*(U - UK1)**2 - SP + (C + P)*U          RGCE0027
      RETURN                                                            RGCE0028
      END                                                               RGCE0029
```

CHAPTER 6

NUMERICAL RESULTS

1. Numerical Data

The algorithm was tested by means of a hypothetical problem which had only two subdivisions in the second plant. This seemingly simple case, however, implied no loss in generality since the nature of the algorithm is such that if it works for a problem like this, it should work for a problem with any number of subdivisions in the second plant. The only reason for choosing a problem with only two subdivisions was just to simplify the computational procedures.

The values of the parameters chosen for the problem were intended to resemble typical values encountered in the real world, but as a whole they do not characterize or correspond to any real cooperation. The values of these parameters were taken as described by Figures 1 through 7.

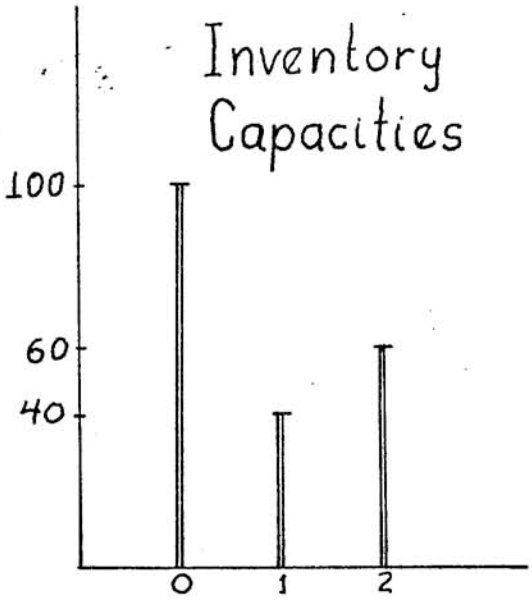


FIG 4

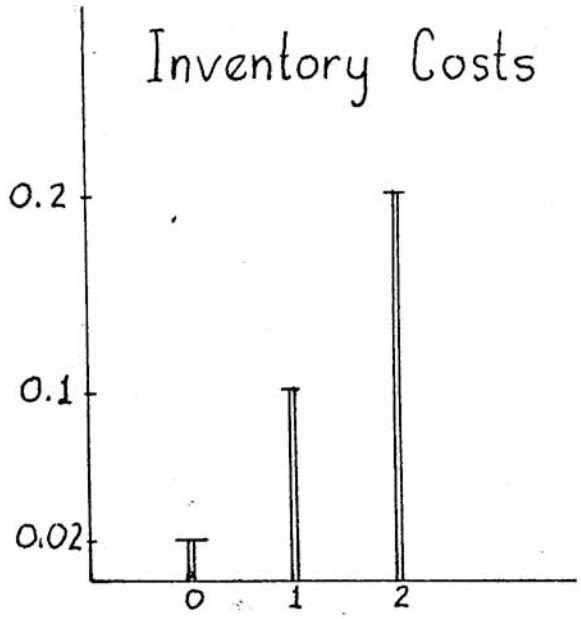


FIG 5

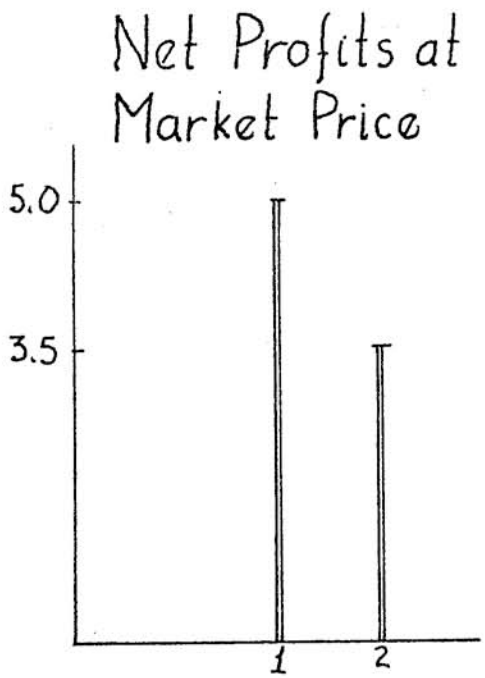


FIG. 6

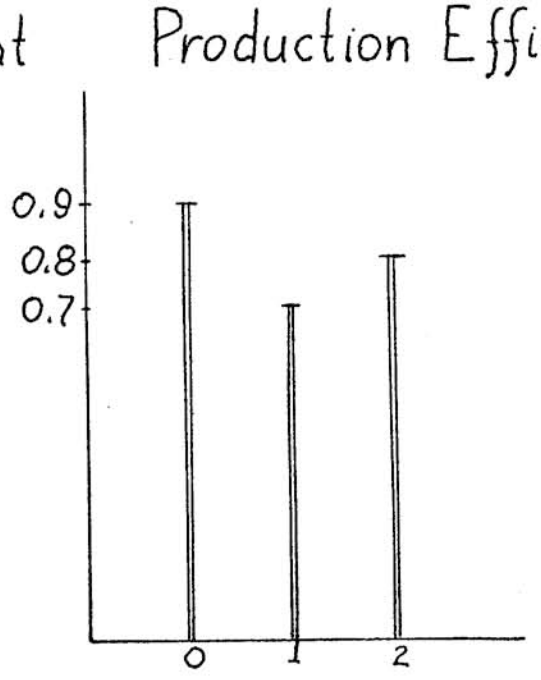


FIG. 7

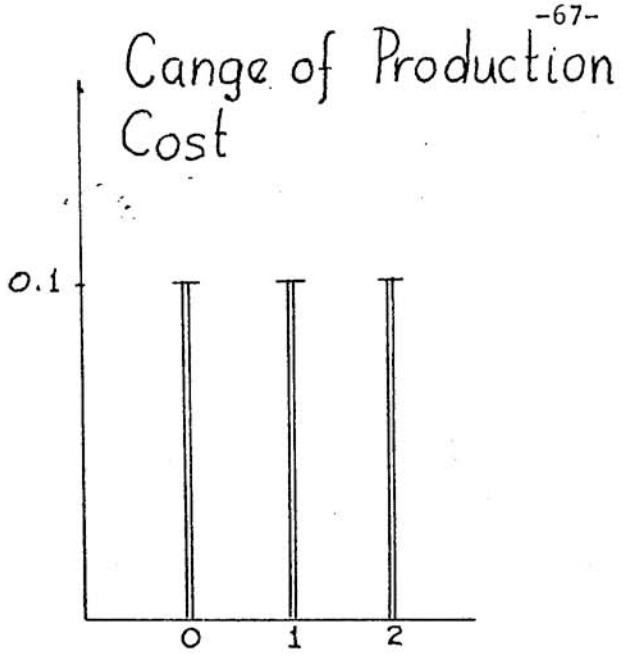


FIG. 1

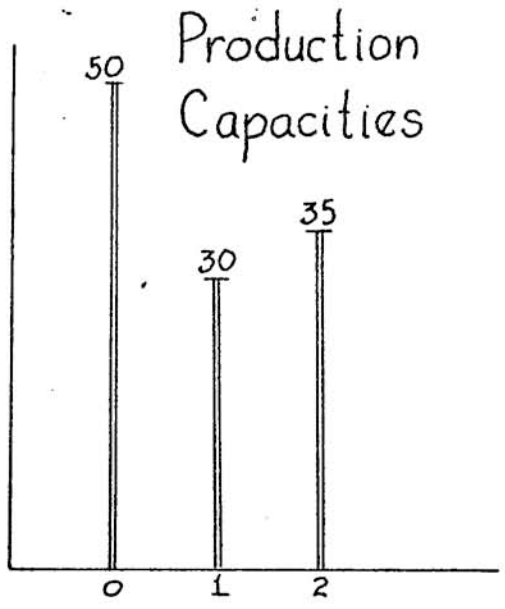
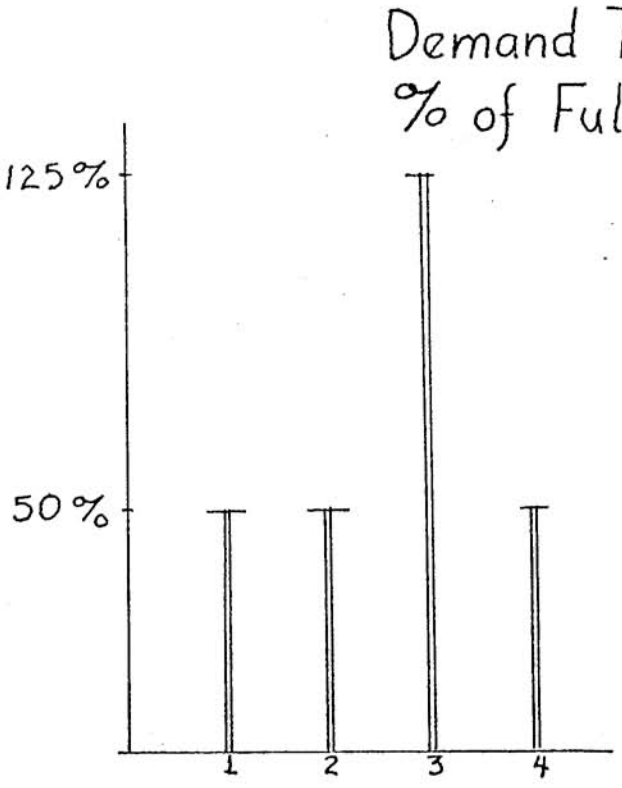


FIG. 2



Subdivision 1

D	15	15	37.5	15
t	1	2	3	4

Subdivision 2

D	17.5	17.5	43.75	17.5
t	1	2	3	4

FIG. 3

2. Results

The numerical results were split into cycles, each cycle corresponding to a different search direction. Initializing the price vectors to zero, that is, zero intermediate product prices and zero intermediate product demand prices, the algorithm was started.

The numerical results corresponding to the first ten cycles were the following:

CYCLE # 1

VALUE OF DUAL FUNCTION = -0.6420754E 03

INTERMEDIATE PRODUCT PRICES
0.0000E 00 0.0000E 00 0.0000E 00 0.0000E 00

PLANT 1 DEMAND PRICES
0.0000E 00 0.0000E 00 0.0000E 00 0.0000E 00

SCHEDULES FOR PLANT 1

INVENTORIES	0.0000E 00	0.0000E 00	0.0000E 00	0.0000E 00
PRODUCTIONS	0.5000E 02	0.5000E 02	0.5000E 02	0.5000E 02
DEMANDS	0.6500E 02	0.6500E 02	0.6500E 02	0.6500E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 1

INVENTORIES	0.0000E 00	0.8000E 01	0.1600E 02	0.0000E 00
PRODUCTIONS	0.3000E 02	0.3000E 02	0.3000E 02	0.2400E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 2

INVENTORIES	0.0000E 00	0.0000E 00	0.1200E 02	0.0000E 00
PRODUCTIONS	0.2800E 02	0.3500E 02	0.3500E 02	0.2100E 02

GRADIENT	0.1300E 02	0.2000E 02	0.2000E 02	0.1526E-04
	-0.7000E 01	0.0000E 00	0.0000E 00	-0.2000E 02
SEARCH	0.3452E 00	0.5311E 00	0.5311E 00	0.4052E-06
DIRECTION	-0.1859E 00	0.0000E 00	0.0000E 00	-0.5311E 00

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = 0.3766E 02

STEP SIZE "A" ALONG SEARCH DIRECTION = 1.000

VALUE OF DUAL FUNCTION = -0.6204851E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = 0.3394E 02

STEP SIZE "A" ALONG SEARCH DIRECTION = 2.000

VALUE OF DUAL FUNCTION = -0.6021069E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = 0.6320E 01

STEP SIZE "A" ALONG SEARCH DIRECTION = 4.000

VALUE OF DUAL FUNCTION = -0.6463389E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = -0.3410E 02

STEP SIZE LIMITS FOR LINEAR INTERPOLATION ARE 2.000 AND 4.000

OPTIMUM STEP SIZE = 2.593

LINEARLY INTERPOLATED
VALUE OF THE FUNCTION = -0.5983596E 03

STEP SIZE "A" ALONG SEARCH DIRECTION = 2.593

VALUE OF DUAL FUNCTION = -0.5993479E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = -0.2183E 02

STEP SIZE LIMITS FOR LINEAR INTERPOLATION ARE 2.000 AND 2.593

OPTIMUM STEP SIZE = 2.558

LINEARLY INTERPOLATED
VALUE OF THE FUNCTION = -0.5985815E 03

STEP SIZE "A" ALONG SEARCH DIRECTION = 2.558

VALUE OF DUAL FUNCTION = -0.5985813E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = 0.6320E 01

CYCLE # 2

VALUE OF DUAL FUNCTION = -0.5985713E 03

INTERMEDIATE PRODUCT PRICES
0.8830E 00 0.1358E 01 0.1358E 01 0.1036E-05

PLANT 1 DEMAND PRICES
-0.4755E 00 0.0000E 00 0.0000E 00 -0.1358E 01

SCHEDULES FOR PLANT 1

INVENTORIES	0.4000E 02	0.2000E 02	0.0000E 00	0.0000E 00
PRODUCTIONS	0.5000E 02	0.5000E 02	0.5000E 02	0.5000E 02
DEMANDS	0.6500E 02	0.6500E 02	0.6500E 02	0.6500E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 1

INVENTORIES	0.0000E 00	0.8000E 01	0.1600E 02	0.0000E 00
PRODUCTIONS	0.3000E 02	0.3000E 02	0.3000E 02	0.3000E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 2

INVENTORIES	0.0000E 00	0.0000E 00	0.1200E 02	0.0000E 00
PRODUCTIONS	0.2800E 02	0.3500E 02	0.3500E 02	0.2800E 02

GRADIENT	-0.2700E 02	0.1526E-04	0.2000E 02	0.1300E 02
	-0.7000E 01	0.0000E 00	0.0000E 00	-0.7000E 01
SEARCH	-0.8083E 00	-0.1887E 00	0.2703E 00	0.2983E 00
DIRECTION	-0.1606E 00	0.0000E 00	0.0000E 00	-0.1606E 00

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = 0.3336E 02

STEP SIZE "A" ALONG SEARCH DIRECTION = 1.000

VALUE OF DUAL FUNCTION = -0.5788396E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = -0.2749E 01

STEP SIZE LIMITS FOR LINEAR INTERPOLATION ARE 0.000 AND 1.000

OPTIMUM STEP SIZE = 0.623

LINEARLY INTERPOLATED
VALUE OF THE FUNCTION = -0.5775030E 03

STEP SIZE "A" ALONG SEARCH DIRECTION = 0.623

VALUE OF DUAL FUNCTION = -0.5778035E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = 0.3336E 02

CYCLE # 3

VALUE OF DUAL FUNCTION = -C.5778035E 03

INTERMEDIATE PRODUCT PRICES
0.3795E 00 0.1241E 01 0.1527E 01 0.1858E 00

PLANT 1 DEMAND PRICES
-0.5755E 00 0.0000E 00 0.0000E 00 -0.1459E 01

SCHEDULES FOR PLANT 1

INVENTORIES	0.4000E 02	C.2000E 02	0.0000E 00	C.0000E 00
PRODUCTIONS	0.5000E 02	C.5000E 02	0.5000E 02	C.5000E 02
DEMANDS	0.6500E 02	C.6500E 02	0.6500E 02	0.6500E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 1

INVENTORIES	0.0000E 00	C.8000E 01	0.1600E 02	C.0000E 00
PRODUCTIONS	0.3000E 02	C.3000E 02	0.3000E 02	0.3000E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 2

INVENTORIES	0.0000E 00	0.0000E 00	0.1200E 02	C.0000E 00
PRODUCTIONS	0.2800E 02	0.3500E 02	0.3500E 02	0.2800E 02

GRADIENT	-0.2700E 02	C.1526E-04	0.2000E 02	C.1300E 02
	-0.7000E 01	0.0000E 00	0.0000E 00	-C.7000E C1
SEARCH DIRECTION	-0.1792E 00	0.2857E 00	0.5560E 00	0.3614E 00
	-0.1946E 00	C.0000E 00	0.0000E 00	-C.1946E 00

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = C.233[illegible] 02

STEP SIZE "A" ALONG SEARCH DIRECTION = 1.000

VALUE OF DUAL FUNCTION = -C.5586218E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = 0.1013E 02

STEP SIZE "A" ALONG SEARCH DIRECTION = 2.000

VALUE OF DUAL FUNCTION = -C.5570886E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = 0.2236E 01

STEP SIZE "A" ALONG SEARCH DIRECTION = 2.118

VALUE OF DUAL FUNCTION = -C.5568237E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = 0.2236E 01

CYCLE # 4

VALUE OF DUAL FUNCTION = -0.5566237E 03

INTERMEDIATE PRODUCT PRICES
0.0000E 00 0.1846E 01 0.2705E 01 0.9514E 00

PLANT 1 DEMAND PRICES
-0.9878E 00 0.0000E 00 0.0000E 00 -0.1871E 01

SCHEDULES FOR PLANT 1

INVENTORIES	0.6000E 02	0.4000E 02	0.2000E 02	0.0000E 00
PRODUCTIONS	0.5000E 02	0.5000E 02	0.5000E 02	0.5000E 02
DEMANDS	0.6500E 02	0.6500E 02	0.6500E 02	0.6500E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 1

INVENTORIES	0.0000E 00	0.8000E 01	0.1600E 02	0.0000E 00
PRODUCTIONS	0.3000E 02	0.3000E 02	0.3000E 02	0.3000E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 2

INVENTORIES	0.0000E 00	0.0000E 00	0.0000E 00	0.0000E 00
PRODUCTIONS	0.2800E 02	0.2100E 02	0.2800E 02	0.2800E 02

```
GRADIENT     -0.4700E 02   -0.3400E 02   -0.7000E 01    0.1300E 02
             -0.7000E 01   -0.1400E 02   -0.7000E 01   -0.7000E 01

SEARCH        0.0000E 00   -0.8205E 00   -0.1689E 00    0.3137E 00
DIRECTION    -0.1689E 00   -0.3379E 00   -0.1689E 00   -0.1689E 00


SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION =    0.4644E 02


STEP SIZE "A" ALONG SEARCH DIRECTION =  1.000

VALUE OF DUAL FUNCTION =  -0.5416638E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION =  -0.1098E 02


STEP SIZE LIMITS FOR LINEAR INTERPOLATION ARE  0.000  AND  1.000

OPTIMUM STEP SIZE =  0.499

LINEARLY INTERPOLATED
VALUE OF THE FUNCTION  =  -0.5361592E 03


STEP SIZE "A" ALONG SEARCH DIRECTION =  0.499

VALUE OF DUAL FUNCTION =  -0.5382686E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION =    0.5237E 01
```

STEP SIZE LIMITS FOR LINEAR INTERPOLATION ARE 0.499 AND 1.000

OPTIMUM STEP SIZE = 0.629

LINEARLY INTERPOLATED
VALUE OF THE FUNCTION = -0.5375874E 03

STEP SIZE "A" ALONG SEARCH DIRECTION = 0.629

VALUE OF DUAL FUNCTION = -0.5375876E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = 0.5237E 01

CYCLE # 5

VALUE OF DUAL FUNCTION = -0.5375876E 03

INTERMEDIATE PRODUCT PRICES
0.0000E 00 0.1330E 01 0.2598E 01 0.1149E 01

PLANT 1 DEMAND PRICES
-0.1094E 01 -0.2124E 00 -0.1062E 00 -0.1977E 01

SCHEDULES FOR PLANT 1

INVENTORIES	0.0000E 00	0.0000E 00	0.0000E 00	0.0000E 00
PRODUCTIONS	0.5000E 02	0.5000E 02	0.5000E 02	0.5000E 02
DEMANDS	0.6500E 02	0.6500E 02	0.6500E 02	0.6500E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 1

INVENTORIES	0.0000E 00	0.8000E 01	0.1600E 02	0.0000E 00
PRODUCTIONS	0.3000E 02	0.3000E 02	0.3000E 02	0.3000E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 2

INVENTORIES	0.0000E 00	0.0000E 00	0.0000E 00	0.0000E 00
PRODUCTIONS	0.2800E 02	0.2100E 02	0.2800E 02	0.2800E 02

GRADIENT	0.1300E 02	C.6000E 01	0.1300E 02	C.1300E C2
	-0.7000E 01	-C.1400E 02	-0.7000E 01	-C.7000E C1
SEARCH	0.4273E 00	C.4198E CC	0.4273E 00	0.4273E CC
DIRECTION	-0.2301E 00	-0.2414E CC	-0.2301E 00	-C.2301E CC

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = C.2737E 02

STEP SIZE "A" ALONG SEARCH DIRECTION = 1.000

VALUE OF DUAL FUNCTION = -C.5617239E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = -0.5740E 02

STEP SIZE LIMITS FOR LINEAR INTERPOLATION ARE 0.000 AND 1.000

OPTIMUM STEP SIZE = 0.392

LINEARLY INTERPOLATED
VALUE OF THE FUNCTION = -0.5268469E 03

STEP SIZE "A" ALONG SEARCH DIRECTION = 0.392

VALUE OF DUAL FUNCTION = -C.5399722E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = -0.2345E 02

STEP SIZE LIMITS FOR LINEAR INTERPOLATION ARE 0.000 AND 0.392

OPTIMUM STEP SIZE = 0.134

LINEARLY INTERPOLATED
VALUE OF THE FUNCTION = -0.5339158E 03

STEP SIZE "A" ALONG SEARCH DIRECTION = 0.134

VALUE OF DUAL FUNCTION = -0.5339158E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = 0.2737E 02

CYCLE # 6

VALUE OF DUAL FUNCTION = -0.5339158E 03

INTERMEDIATE PRODUCT PRICES
0.5732E-01 0.1386E 01 0.2656E 01 0.1206E 01

PLANT 1 DEMAND PRICES
-0.1125E 01 -0.2448E 00 -0.1371E 00 -0.2008E 01

SCHEDULES FOR PLANT 1

INVENTORIES	0.0000E 00	0.0000E 00	0.0000E 00	0.0000E 00
PRODUCTIONS	0.5000E 02	0.5000E 02	0.5000E 02	0.5000E 02
DEMANDS	0.6500E 02	0.6500E 02	0.6500E 02	0.6500E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 1

INVENTORIES	0.0000E 00	0.8000E 01	0.1600E 02	0.0000E 00
PRODUCTIONS	0.3000E 02	0.3000E 02	0.3000E 02	0.3000E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 2

INVENTORIES	0.0000E 00	0.0000E 00	0.0000E 00	0.0000E 00
PRODUCTIONS	0.2800E 02	0.2100E 02	0.2800E 02	0.2800E 02

GRADIENT	0.1300E 02	C.6000E 01	0.1300E 02	C.1300E 02
	-0.7000E 01	-C.1400E 02	-0.7000E 01	-C.7000E C1
SEARCH	-0.1549E 00	-C.1707E 00	0.1677E 00	0.3290E 00
DIRECTION	-0.1771E 00	-0.3548E 00	-0.1771E 00	-C.1771E 00

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = C.1210E 02

STEP SIZE "A" ALONG SEARCH DIRECTION = 0.370

VALUE OF DUAL FUNCTION = -0.5318306E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = 0.4748E 01

CYCLE # 7

VALUE OF DUAL FUNCTION = -0.5318306E 03

INTERMEDIATE PRODUCT PRICES
0.0000E 00 0.1323E 01 0.2710E 01 0.1320E 01

PLANT 1 DEMAND PRICES
-0.1190E 01 -0.3759E 00 -0.2026E 00 -0.2073E 01

SCHEDULES FOR PLANT 1

INVENTORIES	0.0000E 00	0.0000E 00	0.0000E 00	0.0000E 00
PRODUCTIONS	0.5000E 02	0.5000E 02	0.5000E 02	0.5000E 02
DEMANDS	0.6500E 02	0.6500E 02	0.6500E 02	0.6500E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 1

INVENTORIES	0.0000E 00	0.8000E 01	0.1600E 02	0.0000E 00
PRODUCTIONS	0.3000E 02	0.3000E 02	0.3000E 02	0.3000E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 2

INVENTORIES	0.0000E 00	0.0000E 00	0.1200E 02	0.0000E 00
PRODUCTIONS	0.2600E 02	0.3500E 02	0.2300E 02	0.2800E 02

GRADIENT	0.1300E 02	0.2000E 02	0.1300E 02	0.1300E 02
	-0.7000E 01	0.0000E 00	-0.7000E 01	-0.7000E 01
SEARCH	0.4004E 00	0.6160E 00	0.4004E 00	0.4004E 00
DIRECTION	-0.2156E 00	0.0000E 00	-0.2156E 00	-0.2156E 00

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = 0.32[illegible]E 02

STEP SIZE "A" ALONG SEARCH DIRECTION = 1.000

VALUE OF DUAL FUNCTION = -0.5732649E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = -0.6918E 02

STEP SIZE LIMITS FOR LINEAR INTERPOLATION ARE 0.000 AND 1.000

OPTIMUM STEP SIZE = 0.273

LINEARLY INTERPOLATED
VALUE OF THE FUNCTION = -0.5229683E 03

STEP SIZE "A" ALONG SEARCH DIRECTION = 0.273

VALUE OF DUAL FUNCTION = -0.5337136E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = -0.2421E 02

STEP SIZE LIMITS FOR LINEAR INTERPOLATION ARE 0.000 AND 0.273

OPTIMUM STEP SIZE = 0.033

LINEARLY INTERPOLATED
VALUE OF THE FUNCTION = -0.5291233E 03

STEP SIZE "A" ALONG SEARCH DIRECTION = 0.033

VALUE OF DUAL FUNCTION = -0.5291238E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = -0.2421E 02

CYCLE # 8

VALUE OF DUAL FUNCTION = -0.5291238E 03

INTERMEDIATE PRODUCT PRICES
0.3339E-01 0.1374E 01 0.2751E 01 0.1361E 01

PLANT 1 DEMAND PRICES
-0.1200E 01 -0.3759E 00 -0.2206E 00 -0.2091E 01

SCHEDULES FOR PLANT 1

INVENTORIES	0.6000E 02	0.4000E 02	0.2000E 02	0.0000E 00
PRODUCTIONS	0.5000E 02	0.5000E 02	0.5000E 02	0.5000E 02
DEMANDS	0.6500E 02	0.6500E 02	0.6500E 02	0.6500E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 1

INVENTORIES	0.0000E 00	0.8000E 01	0.1600E 02	0.0000E 00
PRODUCTIONS	0.3000E 02	0.3000E 02	0.3000E 02	0.3000E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 2

INVENTORIES	0.0000E 00	0.0000E 00	0.1200E 02	0.0000E 00
PRODUCTIONS	0.2800E 02	0.2500E 02	0.2300E 02	0.2800E 02

CYCLE # 9

VALUE OF DUAL FUNCTION = -0.527 013E 03

INTERMEDIATE PRODUCT PRICES
0.0000E 00 0.1393E 01 0.2771E C1 0.1408E 01

PLANT 1 DEMAND PRICES
-0.1232E 01 -0.3759E 00 -0.2487E C0 -0.2116E 01

SCHEDULES FOR PLANT 1

INVENTORIES	0.0000E 00	C.0000E 00	0.0000E 00	C.0000E C0
PRODUCTIONS	0.5000E 02	C.5000E 02	0.5000E 02	C.5000E C2
DEMANDS	0.6500E 02	C.6500E 02	0.6500E 02	C.6500E C2

SCHEDULES FOR PLANT 2 SUBDIVISION # 1

INVENTORIES	0.0000E 00	C.8000E C1	0.1600E 02	0.0000E C0
PRODUCTIONS	0.3000E 02	0.3000E 02	0.3000E 02	C.3000E C2

SCHEDULES FOR PLANT 2 SUBDIVISION # 2

INVENTORIES	0.0000E 00	C.0000E C0	0.1200E 02	C.0000E C0
PRODUCTIONS	0.2800E 02	C.3500E 02	0.2800E 02	C.2800E C2

CYCLE # 9

VALUE OF DUAL FUNCTION = -0.5273013E 03

INTERMEDIATE PRODUCT PRICES
0.0000E 00 0.1393E 01 0.2771E 01 0.1408E 01

PLANT 1 DEMAND PRICES
-0.1232E 01 -0.3759E 00 -0.2467E 00 -0.2116E 01

SCHEDULES FOR PLANT 1

INVENTORIES	0.0000E 00	0.0000E 00	0.0000E 00	0.0000E 00
PRODUCTIONS	0.5000E 02	0.5000E 02	0.5000E 02	0.5000E 02
DEMANDS	0.6500E 02	0.6500E 02	0.6500E 02	0.6500E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 1

INVENTORIES	0.0000E 00	0.8000E 01	0.1600E 02	0.0000E 00
PRODUCTIONS	0.3000E 02	0.3000E 02	0.3000E 02	0.3000E 02

SCHEDULES FOR PLANT 2 SUBDIVISION # 2

INVENTORIES	0.0000E 00	0.0000E 00	0.1200E 02	0.0000E 00
PRODUCTIONS	0.2800E 02	0.3500E 02	0.2800E 02	0.2800E 02

GRADIENT	0.1300E 02	C.2000E 02	0.1300E 02	C.1300E C2
	-0.7000E 01	C.0000E 00	-0.7000E 01	-C.7000E C1
SEARCH	0.4004E 00	C.6156E 00	0.4004E 00	C.4004E CC
DIRECTION	-0.2156E 00	C.0000E 00	-0.2156E 00	-C.2156E CC

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = C.3247E 02

STEP SIZE "A" ALONG SEARCH DIRECTION = 1.000

VALUE OF DUAL FUNCTION = -0.5746543E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = -0.7029E 02

STEP SIZE LIMITS FOR LINEAR INTERPOLATION ARE 0.000 AND 1.000

OPTIMUM STEP SIZE = 0.228

LINEARLY INTERPOLATED
VALUE OF THE FUNCTION = -0.5203962E 03

STEP SIZE "A" ALONG SEARCH DIRECTION = 0.228

VALUE OF DUAL FUNCTION = -0.5330017E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = -0.3283E 02

STEP SIZE LIMITS FOR LINEAR INTERPOLATION ARE 0.000 AND 0.228

OPTIMUM STEP SIZE = 0.035

LINEARLY INTERPOLATED
VALUE OF THE FUNCTION = -0.5276633E 03

STEP SIZE "A" ALONG SEARCH DIRECTION = 0.035

VALUE OF DUAL FUNCTION = -0.5277773E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = -0.2421E 02

STEP SIZE LIMITS FOR LINEAR INTERPOLATION ARE 0.000 AND 0.035

OPTIMUM STEP SIZE = 0.015

LINEARLY INTERPOLATED
VALUE OF THE FUNCTION = -0.5273013E 03

STEP SIZE "A" ALONG SEARCH DIRECTION = 0.015

VALUE OF DUAL FUNCTION = -0.5273013E 03

SLOPE OF DUAL FUNCTION
ALONG SEARCH DIRECTION = 0.3247E 02

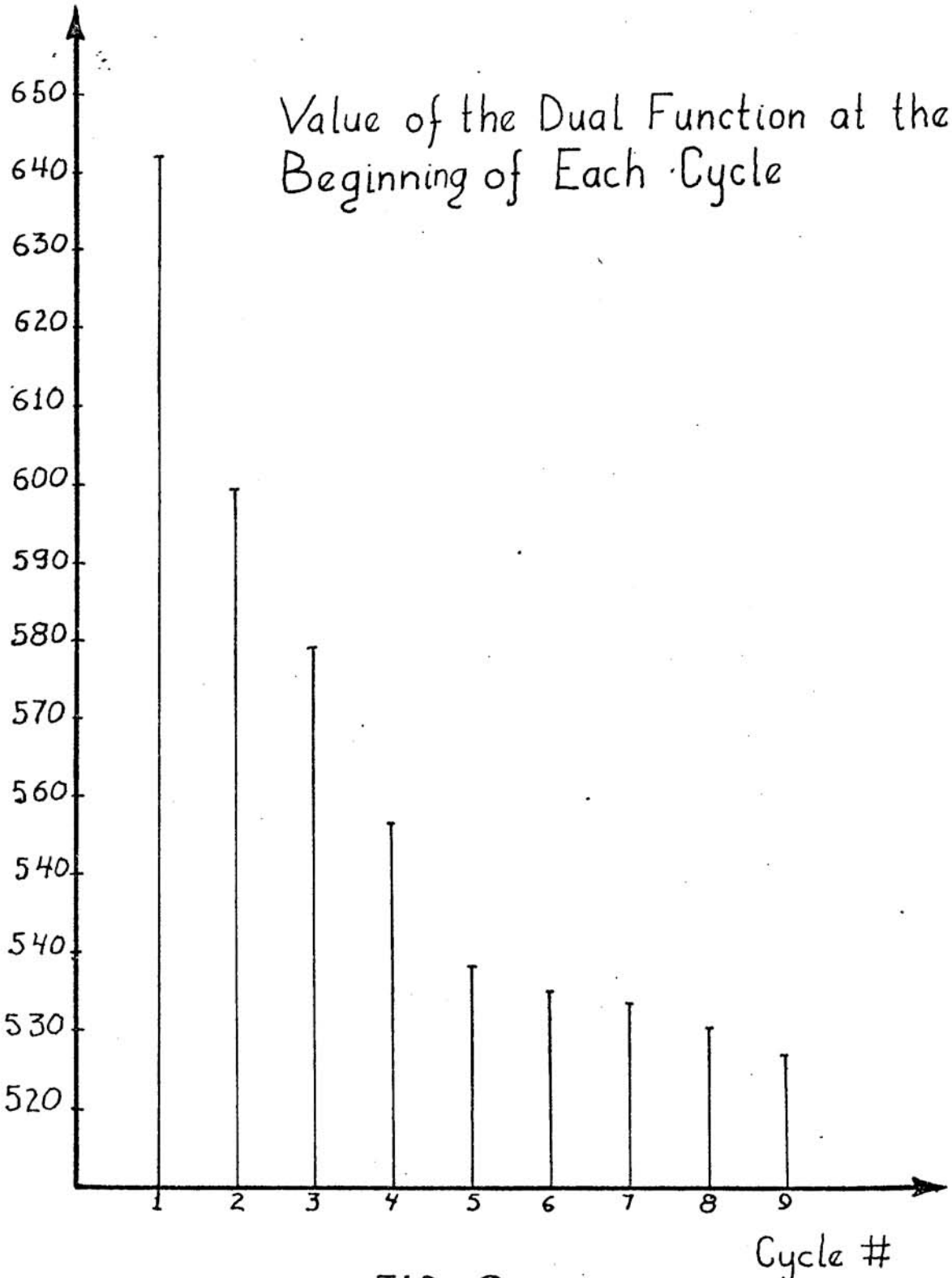

Value of the Dual Function at the Beginning of Each Cycle
650
640
630
620
610
600
590
580
570
560
540
540
530
520
1
2
3
4
5
6
7
8
9
Cycle #

FIG 8

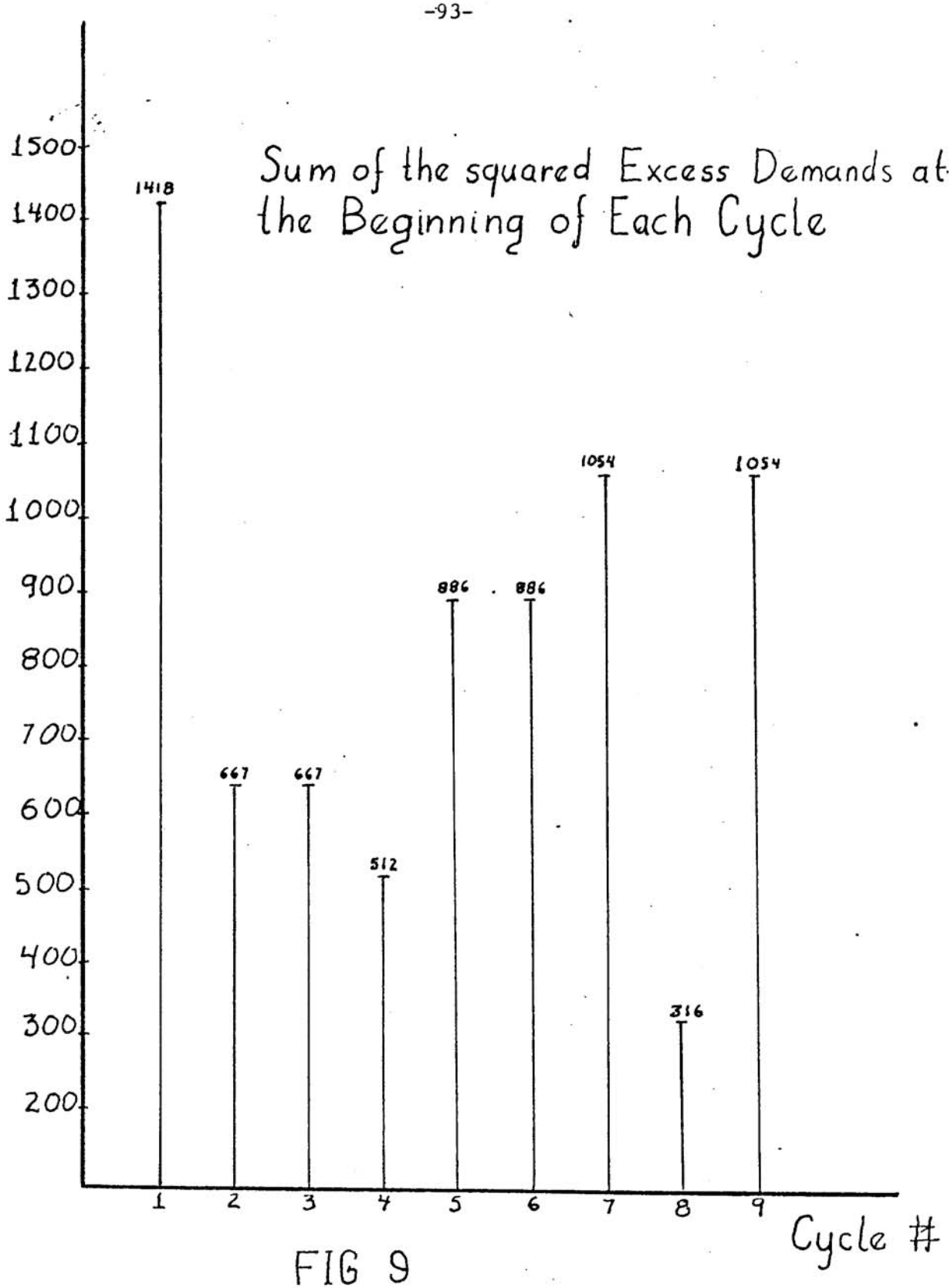

Sum of the squared Excess Demands at the Beginning of Each Cycle
1500
1400
1300
1200
1100
1000
900
800
700
600
500
400
300
200
1418
667
667
512
886
886
1054
316
1054
1
2
3
4
5
6
7
8
9
Cycle #

FIG 9

3. Discussion

Due to computer time limitations, the algorithm was not allowed to run for a sufficiently long period of time for it to converge to within the specified tolerances. However, by observing the development of the values of the variables through the cycels obtained, it is clear that the algorithm was indeed moving in the right direction and at a reasonable pace.

The cost functional seems to approach the optimum value in an asymptotic fashion (see Figure 8). Theoretically, however, due to the piecewise linear characteristic of the dual function (see Appendix II), the overall optimum should be obtained in a final number of steps.

The value of the sum of the squared excess demands is not very well behaved since it tends to oscillate wildly between cycles. The reason for these oscillations is that, as explained in Appendix II, the gradient of the dual function is discontinuous at some points and consequently the sum of the squared demands is discontinuous also. However, the general trend or the average of this sum should and does go down from cycle to cycle (see Figure 9). With a better criterion for the selection of the direction of search at the beginning of each cycle, most of these oscillations could be avoided. In fact, if the direction of steepest ascent is used as the search direction in each cycle, the sum of the squared demands should be monotonically decreasing from cycle to cycle. However, with a non-optimum method of selection of the search directions some oscillations, like the ones shown in Figure 9, are bound to occur due to zig-zagging problems.

4. An Alternative Method

An alternative way of solving this problem is by means of the Generalized Danzig Wolfe decomposition principle (see reference # 5). This method is specially suited to solve problems of the form:

$$\text{minimize } f(x) = \sum_{i=1}^{p} f_i(x_i) \tag{27}$$

where each f_i is a convex function, subject to a set of linear coupling constraints:

$$\sum_{i=1}^{p} A_i x_i = b \tag{28}$$

and to a set of possibly nonlinear constraints involving each x_i independently:

$$x_i \in S_i \qquad i=1,\ldots,p \tag{29}$$

where the sets S_i are convex.

Clearly, the inventory problem that was studied in this thesis fits, with minor modifications, into this general type of problems where (see Chapter 3) equation (9) corresponds to equation (27), equations (10), (13) and (14) correspond to equation (28), and equations (11) and (12) correspond to equation (29).

The main idea of this method consists of converting the nonlinearities of the problem into linear functions by means of grid linearization techniques (see reference # 2). For each variable x_i a grid of points is defined in such a way that

$$x_i^t \in S_i \qquad \text{for } t=1,\ldots,r$$

Each function is then replaced by its linearization on the grid $\{x_i^t\}$

$$f_i(x_i) = \sum_{t=1}^{r} \lambda_i^t \, f_i(x_i^t) \tag{30}$$

where

$$\sum_{t=1}^{r} \lambda_i^t = 1 \; ; \qquad \lambda_i^t \geq 0 \tag{31}$$

The associated value of x_i being:

$$x_i^a = \sum_{t=1}^{r} \lambda_i^t \, x_i^t \tag{32}$$

A linear approximation to the original problem is then obtained by substituting each f_i by its linear approximation and by substituting (32) into (28) obtaining:

$$\text{minimize } F(x) = \sum_{i,t} \lambda_i^t \, f_i(x_i^t) \tag{33}$$

subject to

$$\sum (A^i x_i^t)\lambda_i^t = b \tag{34}$$

$$\sum \lambda_i^t = 1 \; ; \qquad \lambda_i^t \geq 0 \qquad i=1,\ldots,p \tag{35}$$

Since the grid points x_i^t were defined to be elements of the sets S_i and since x_i^a, being a convex combination of the x_i^t, is also inside the convex sets S_i, equation (29) is no longer necessary in this new formulation.

The advantages of this method are mainly based on the inherent simplicity of its linear structure. Even though due to the great number of columns that are introduced by the linearization the problem cannot be solved using a straight simplex method, a procedure like the one introduced by Danzig and Wolf in their linear decomposition principle is perfectly suited for this problem. Problems with thousands of columns

have been solved using this method. Therefore, the use of this grid linearization approach to solve the original inventory problem is indeed promising.

Another advantage of this method is that the convergence process is from within the allowable region, that is to say that every single point along the convergence process is perfectly feasible (does not violate any constraint). This is a particularly desired property since it means that the algorithm can be stopped before convergence and still obtain a point that, besides it being close to the optimal, it is also feasible.

In contrast with this, the method that was actually used here to solve the problem provides absolutely no guarantee that any point along the convergence path will be feasible, unless it is the optimal. Due to the structure of the algorithm, the convergence path may come in and out of the feasible region from iteration to iteration without disturbing the convergence apparatus of the algorithm in the least. It may indeed stay inside the feasible region for the most part of the convergence path, the same way it may stay out of it; whether the algorithm converges from inside the feasible region or from outside of it depends for the most part on the starting point. However, as the algorithm approaches the optimal point, the convergence path starts jumping in and out of the feasible region due to the zig-zagging problems discussed in the previous section. This is clearly a major drawback of this method in comparison with the previously discussed Generalized Danzig Wolf method.

The study of the general feasibility and advantages of the Generalized Danzig-Wolf decomposition principle for the solution of the

inventory problem shall not be discussed any further in this work. It is nevertheless recommended as a possibility for further research on this problem.

APPENDIX I

JUSTIFICATION OF THE ALGORITHM BASED ON SADDLE POINT AND DUALITY THEORY FROM MATHEMATICAL PROGRAMMING

Consider the following problem:

$$\text{minimize} \quad f(\underline{x}) \tag{1}$$

subject to:

$$g_i(\underline{x}) \leq 0 \qquad i=1,\ldots,\hat{m} \tag{2}$$

$$g_i(\underline{x}) = 0 \qquad i=\hat{m},\ldots,m \tag{3}$$

$$\underline{x} \in S \quad R^n \quad (S \text{ compact}) \tag{4}$$

where f and the g_i are real valued functions defined over S.

The Lagrangian problem associated with this problem is:

$$L(\underline{x},\underline{u}) = f(\underline{x}) + \sum_{i=1}^{m} u_i \, g_i(\underline{x}) \tag{5}$$

Now, defining a saddle point of L to be a point $(\underline{x}^o,u^o)$ such that:

$$L(\underline{x}^o,\underline{u}^o) \leq L(\underline{x},\underline{u}^o) \qquad \text{for all} \quad x \in S$$

$$L(\underline{x}^o,\underline{u}^o) \geq L(\underline{x}^o,\underline{u}) \qquad \text{for all} \quad \underline{u} \in \Omega$$

where $\Omega = \{\underline{u} \mid u_i \geq 0 \quad \text{for} \quad i=1,\ldots,\hat{m}\}$

the following theorem may be stated:

Theorem 1:

Letting $\underline{x} \in S$ and $u_i \geq 0$, $i=1,\ldots,\hat{m}$, then $(\underline{x}^o,\underline{u}^o)$ is a saddle point for the Lagrangian function L if and only if

a) $\underline{x}^o$ minimizes $L(\underline{x},\underline{u}^o)$ over S

b) $g_i(\underline{x}^o) \leq 0$ for $i=1,\ldots,\hat{m}$

$g_i(\underline{x}^o) = 0$ for $i=\hat{m},\ldots,m$

c) $u_i^o g_i(\underline{x}^o) = 0$ for $i=1,\ldots,m$

The proof for this theorem follows directly from the Kuhn Tucker conditions given by the Kuhn Tucker theorem (see reference # 3).

Now the usefulness of the derivation of a systematic procedure for finding a saddle point of the Lagrangian function can be very clearly seen from the statement of the following theorem:

Theorem 2

If $(\underline{x}^o,\underline{u}^o)$ is a saddle point of the Lagrangian function given by equation (5), then $\underline{x}^o$ solves the problem (1) to (4).

Proof: Since (x^o,u^o) is a saddle point of L, condition (a) to (c) of Theorem (1) must hold

Letting $\underline{g}(\underline{x}) = [g_1(\underline{x}),g_2(\underline{x}),\ldots,g_{\hat{m}}(\underline{x}),\ldots,g_m(\underline{x})]'$

condition (a) becomes

$$f(\underline{x}^o) + \langle \underline{u}^o, g(\underline{x}^o)\rangle \leq f(x) + \langle \underline{u}^o, g(\underline{x})\rangle \qquad (6)$$

for all $x \in S$ satisfying (2) and (3).

From condition (b) and (c) it is known that:

$$g_i(\underline{x}^o) = 0 \quad \text{for } i=\hat{m},\ldots,m$$

and

$$u_i^o g_i(\underline{x}^o) = 0 \quad \text{for } i=1,\ldots,m$$

Therefore the scalar product of the price vector with the constraint vector must vanish at the point where the saddle point occurs, i.e.

$$\langle \underline{u}^o, \underline{g}(\underline{x}^o)\rangle = 0$$

Consequently, equation (6) becomes:

$$f(\underline{x}^o) \leq f(\underline{x}) + \langle \underline{u}^o, g(\underline{x}) \rangle$$

Now, the term $\langle \underline{u}^o, g(\underline{x}) \rangle$ is smaller than or equal to zero. Therefore, the following equation must hold

$$f(\underline{x}^o) \leq f(\underline{x})$$

for all $\underline{x}$ satisfying equation (2) → (4).

Q.E.D.

From Theorem 2 it is obvious that it is possible to solve the problem given by equations (1) to (4), in an indirect way; by finding the saddle point(s) of its Lagrangian function. However, up until now, nothing has been said as to how one would go about finding such a point; the discussion that follows addresses specifically to that question.

Consider a function h of the Lagrange multipliers $\underline{u}$ defined as:

$$h(\underline{u}) = \min_{x \in s} L(\underline{x}, \underline{u})$$

and let

$$X(\underline{u}) = \underline{x} \mid \begin{array}{l} \underline{x} \text{ minimizes } L(\underline{x},\underline{u}) \text{ over} \\ \text{the compact set } S \end{array}$$

The function $h(\underline{u})$ is called the dual function of f and its domain of definition is the set of vectors $\underline{u}$, $u_i \geq 0$ $i=1,\ldots,\hat{m}$, for which the function L has a bounded infimum over the set S, i.e.

$$D = \underline{u} \mid \begin{array}{l} u_i \geq 0 \text{ for } i=1,\ldots,\hat{m} \\ \text{and } \min_{\underline{x} \in S} L(\underline{x},\underline{u}) \text{ exists} \end{array} \qquad (7)$$

Summarizing, from the primal problem

$$\text{minimize} \quad f(x)$$

$$\text{subject to} \quad g_i(\underline{x}) \leq 0 \qquad i=1,\ldots,\hat{m}$$

$$g_i(\underline{x}) = 0 \qquad i=\hat{m},\ldots,m$$

$$\underline{x} \in S$$

The following dual problem has been defined:

$$\text{maximize} \quad h(\underline{u}) \tag{8}$$

$$\text{subject to} \quad \underline{u} \in D \quad \text{where D is given equation (7)} \tag{9}$$

The reason why this new problem is called the "dual" of the original problem is just to be consistent with the widely accepted linear programming nomenclature. That is, assuming the problem to be linear:

$$\text{minimize} \quad \langle \underline{c},\underline{x} \rangle$$

$$\text{subject to} \quad A\underline{x} \geq \underline{b}; \quad \underline{x} \geq 0$$

Then the Lagrangian function becomes:

$$L(\underline{x},\underline{u}) = \langle \underline{c},\underline{x} \rangle + \langle \underline{u},\underline{b}-A\underline{x} \rangle, \quad \underline{u} \geq 0$$

and

$$h(\underline{u}) = \min_{\underline{x}\geq 0} [\langle \underline{c},\underline{x} \rangle + \langle \underline{u},\underline{b}-A\underline{x} \rangle]$$

$$= \min_{\underline{x}\geq 0} [\langle \underline{c}-A'\underline{u},x \rangle + \langle \underline{b},\underline{u} \rangle]$$

This minimum exists if and only if $\underline{c}-A'\underline{u} \geq 0$ since otherwise $h(\underline{u})$ would blow to minus infinity. Therefore:

$$D = \{\underline{u} \mid A'\underline{u} \geq \underline{c}, \quad \underline{u} \geq 0\}$$

and

$$h(\underline{u}) = \min_{\underline{x}\geq 0} \langle \underline{b},\underline{u} \rangle = \langle \underline{b},\underline{u} \rangle$$

At this point it should be clear that the problem of maximizing $h(\underline{u})$ over set D is the standard linear programming formulation of the dual problem, i.e.

$$\text{maximize } \langle \underline{b}, \underline{u} \rangle$$

$$\text{subject to } A'\underline{u} \geq \underline{c} \; ; \; \underline{u} \geq 0$$

The following theorem is an immediate consequence of the definition of the dual as stated by equations (8) and (9):

Theorem 3

$h(\underline{u}) \leq f(\underline{x})$ for all $\underline{x}$ that satisfy the constraints given by equations (2) to (5) for all $\underline{u} \in D$

Proof:

$$h(\underline{u}) = \min_{\underline{x} \in S} L(\underline{x}, \underline{u}) \qquad \underline{u} \in D$$

$$= \min_{\underline{x} \in S} [f(x) + \langle \underline{u}, \underline{g}(\underline{x}) \rangle]$$

Therefore: $h(\underline{u}) \leq f(x) + \langle \underline{u}, \underline{g}(\underline{x}) \rangle$ for all $\underline{x} \in S$, $\underline{u} \in D$

But, if $\underline{x}$ satisfies equations (2) to (4), then:

$$\langle \underline{u}, \underline{g}(\underline{x}) \rangle = \sum_{i=1}^{m} u_i g_i(\underline{x}) = \sum_{i=1}^{\hat{m}} u_i g_i(\underline{x}) \leq 0$$

Consequently, if:

$$h(\underline{u}) \leq f(\underline{x}) + \langle \underline{u}, g(\underline{x}) \rangle$$

it follows that $h(\underline{u}) \leq f(x)$ for all $\underline{u} \in D$ and $\underline{x}$ satisfying equations (2) to (4).

Q.E.D.

Other results that follow immediately from Theorem 3 are stated below as corollaries:

Corollary 1

If $\inf\{f(\underline{x}) \mid \underline{x} \text{ satisfies (2) to (4)}\} = -\infty$

then the dual problem is unfeasible

Corollary 2

If $\sup\{h(\underline{u}) \mid \underline{u} \in D\} = +\infty$ then the primal is unfeasible

Corollary 3

If there exists an $\underline{x}^o$ satisfying equations (2) to (4), and a $\underline{u}^o \in D$ such that:

$$f(\underline{x}^o) = h(\underline{u}^o)$$

then $\underline{x}^o$ solves the primal and $\underline{u}^o$ solves the dual.

Proof: From Theorem 3 it is known that

$$h(\underline{u}) \leq f(\underline{x}) \quad \text{for all } \underline{x} \text{ satisfying (2) + (4) and for all } \underline{u} \in D.$$

Then since by assumption $\underline{u}^o \in D$

$$h(\underline{u}^o) \leq f(\underline{x})$$

But since, also by assumption, $h(\underline{u}^o) = f(\underline{x}^o)$ it follows that

$$f(\underline{x}^o) \leq f(\underline{x}) \quad \text{for all } \underline{x} \text{ satisfying (2) to (4)}$$

Similarly, since by assumption $\underline{x}^o \in D$

$$h(\underline{u}) \leq f(\underline{x}^o)$$

$$h(u) \leq h(\underline{u}^o) \quad \text{for all } \underline{u} \in D$$

Q.E.D.

Finally, these duality theory results have to be related with the previously derived saddle point theory results. This is best done by means of the following theorem.

Theorem 4

$$f(\underline{x}^o) = h(\underline{u}^o)$$

with $\underline{x}^o$ satisfying (2) to (4) and $\underline{u}^o \in D$, if and only if $(\underline{x}^o, \underline{u}^o)$ is a saddle point of $L(\underline{x},\underline{u})$.

Proof:

I) Sufficiency:

Since $\underline{x}^o$ satisfies (2) → (4):

$$g_i(\underline{x}^o) \leq 0 \qquad i=1,\ldots,\hat{m} \tag{10}$$

$$g_i(\underline{x}^o) = 0 \qquad i=\hat{m},\ldots,m \tag{11}$$

If $\underline{x}^o \notin X(\underline{u}^o) = \{\underline{x} \mid \underline{x}$ minimizes $L(\underline{x},\underline{u})$ over set $S\}$

then there is an $\underline{x}^1 \in X(\underline{u}^o)$ such that

$$h(\underline{u}^o) = f(\underline{x}^1) + \langle \underline{u}^o, \underline{g}(\underline{x}^1) \rangle \leq f(\underline{x}^o) + \langle \underline{u}^o, \underline{g}(\underline{x}^o) \rangle$$

But from assumption, $h(\underline{u}^o) = f(\underline{x}^o)$; therefore:

$$f(\underline{x}^o) \leq f(\underline{x}^o) + \langle \underline{u}^o, \underline{g}(\underline{x}^o) \rangle$$

which implies that: $$\langle \underline{u}^o, \underline{g}(\underline{x}^o) \rangle \geq 0 \tag{12}$$

Now, equation (12) together with equations (10) and (11) imply that

$$u_i^o \leq 0 \qquad i=1,\ldots,\hat{m}$$

which violates the assumption that $\underline{u}^o \in D$. Therefore, it may be concluded that:

$$\underline{x}^o \in X(\underline{u}^o) \tag{13}$$

Therefore: $f(\underline{x}^o) = h(\underline{u}^o) = f(\underline{x}^o) + \langle \underline{u}^o, g(\underline{x}^o) \rangle$

which implies that $\langle \underline{u}^o, g(\underline{x}^o) \rangle = 0$ (14)

By Theorem 1 equations (10), (11), (13) and (14) imply that $(\underline{x}^o, \underline{u}^o)$ is a saddle point of $L(\underline{x},\underline{u})$

II) Necessity:

Since, by assumption, $(\underline{x}^o, \underline{u}^o)$ is a saddle point of $L(\underline{x},\underline{u})$ from Theorem 1, it follows that:

i) $h(\underline{u}^o) = f(\underline{x}^o) + \langle \underline{u}^o, g(\underline{x}^o) \rangle$

ii) $\underline{x}^o$ satisfies (2) → (4)

iii) $\underline{u}^o \in D$

From ii) and iii): $\langle \underline{u}^o, g(\underline{x}^o) \rangle = 0$

Finally, from i): $h(\underline{u}^o) = f(\underline{x}^o)$

Q.E.D.

These results imply that one way of solving the problem given by equations (1) to (4) is by means of the following algorithmic procedure:

1) Solve the dual problem given by equations (8) and (9). Let the optimal multiplier values be given by the vector $\underline{u}^o$.

2) Get an $\underline{x}^o$ such that $\underline{x}^o \in X(\underline{u}^o)$

3) If $h(\underline{u}^o) \neq f(\underline{x}^o)$ return to step 2), but if $h(\underline{u}^o) = f(\underline{x}^o)$ then $\underline{x}^o$ is the solution of the primal.

Clearly, if $X(\underline{u})$ consists of a single point $x(\underline{u})$, the procedure described above should converge after the first iteration or not converge at all.

Even though it might be obvious to the reader, the author wishes nevertheless to emphasize the fact that the problem that this thesis addresses directly to belongs to the generalized type of problems given by equations (1) → (4). Consequently, the results that have been derived in this discussion should be directly applicable to the problem in the main body of this thesis.

That is, considering the primal problem:

minimize
$$J_T = \underline{S}'(N+1)W\,\underline{S}(N+1) + \underline{S}'(N)W\,\underline{S}(N) - M\,\underline{TS}(N) + \sum_{n=0}^{N-1} \{\underline{S}'(n)W\,\underline{S}(n) - M\,\underline{TS}(n) + [\underline{U}(n) - \underline{U}(n+1)]'\,T[\underline{U}(n) - \underline{U}(n+1)] \tag{15}$$

subject to
$$S(n+1) = \underline{S}(n) + K\,\underline{U}(n) - \underline{D}(n) \tag{16}$$

$$0 \leq \underline{S}(n) \leq \underline{S}^* \tag{17}$$

$$0 \leq \underline{U}(n) \leq \underline{U}^* \tag{18}$$

$$\sum_{i=1}^{m} u_i(n) \leq s_o(n) + k_o\,u_o(n) \tag{19}$$

$$d_o(n) = \sum_{i=1}^{m} u_i(n) \tag{20}$$

the associated Lagrangian function would be:

$$L = J_T + \sum_{i=0}^{N} c(n)\{\sum_{i=1}^{m} u_i(n) - [s_o(n) + k_o\, u_o(n)]\}$$

$$+ p(n)[\sum_{i=1}^{m} u_i(n) - d_o(n)]$$

where $\underline{c} = [c(1),c(2),\ldots,c(N)]'$

$\underline{p} = [p(1),p(2),\ldots,p(N)]'$

and $\underline{r} = \dfrac{\underline{c}}{\underline{p}}$

Then, defining a dual function $h(\underline{c},\underline{p}) = h(\underline{r})$ as:

$$h(\underline{r}) = \underset{(\underline{u},d_o)\in\Omega}{\text{minimum}}\ L(\underline{r},\underline{u},d_o)$$

where $\Omega = \left\{ \underline{u},d_o \;\middle|\; \text{Equations (16), (17) and (18) are satisfied for } n=0,\ldots,N \right\}$

and $U(\underline{r}) = \left\{ \dfrac{\underline{u}}{d_o} \;\middle|\; \underline{u} \text{ and } d_o \text{ minimize } L(\underline{r},\underline{u},d_o) \text{ over the set } \Omega \right\}$

a dual problem of the primal given by equations (15) to (20) may be defined as:

$$\underset{\underline{c} \geq 0}{\text{maximize}}\ h(\underline{r})$$

Then, from Theorem 4, it may be concluded that if there exists a vector of multipliers $\underline{r}^*$ with the property that some vector

$$\frac{\underline{u}^*}{d_o^*} \quad \varepsilon\ U(\underline{r}^*)$$

solves the dual, and $h(\underline{r}^*) = J_T(\underline{u}^*,d_o^*)$, then it follows that the vector

$$\frac{\underline{u}^*}{d_o^*}$$

solves the primal.

APPENDIX II

CONTINUITY AND CONCAVITY OF THE DUAL FUNCTION

In order to address the question of continuity it is necessary to first look at the behavior of the gradient. Recalling that the dual function was defined to be the result of an optimal control problem of the form:

$$h(\underline{c},\underline{p}) = \min_{\underline{u}} L(\underline{u},\underline{c},\underline{p})$$

or more conveniently:

$$h(\underline{r}) = \min_{\underline{u}} L(\underline{u},\underline{r}) \qquad (1)$$

$$\text{where} \quad \underline{r} = \frac{\underline{c}}{\underline{p}}$$

subject to certain constraints, it is obvious that the behavior of h(r) will be strongly influenced by the nature of the Lagrangian function $L(\underline{u},\underline{r})$. Now, clearly, since L is quadratic and the constraints are linear, $h(\underline{r})$ exists for any value of $\underline{r}$. However, since both the dynamics and the control variables $(\underline{u},d_o)$ have been discretized in the actual solution of the optimal control problem, it is obvious that the set of possible solutions to this minimization is finite. That is, there is only a finite number of possible optimal trajectories/optimal controls that can result from this inner minimization due to the internal discretization of variables done by the optimization algorithm used (Dynamic Programming). Consequently the solution to the inner minimization (equation (1)) cannot

vary continuously with $\underline{r}$. That is, for any cost vector $\underline{r}^*$ there is a non-zero neighborhood C containing $\underline{r}^*$ for which the solution to the inner minimization remains constant.

Examining the structure of the dual function $h(\underline{r})$ it is found that if the optimal trajectory and optimal controls for the inner problem remain constant, the dual function varies linearly with the cost vector

$$h(\underline{r}) = h(\underline{c},\underline{p}) = f[\underline{x}^*,\underline{u}^*,d_o]$$

$$+ \sum_{n=0}^{N} c(n) \sum_{i=1}^{m} u_i^*(n) - [s_o^*(n) - u_o^*(n)\ k_o]$$

$$+ \sum_{n=0}^{N} p(n) \sum_{i=1}^{m} u_i^*(n) - d_o^*(n)$$

where $\underline{s}^*$ and $\underline{u}^*$ are the optimal trajectory and optimal control vectors, respectively. And:

$$f[\underline{s}^*,\underline{u}^*] = \sum_{i=0}^{m} w_i[s_i^2(N) + s_i^2(N+1)] - mp_i\ ts_i(N)$$

$$+ \sum_{n=0}^{N-1} \{w_i\ s_i^2(n) + tc_i[u_i(n) - u_i(n+1)]^2$$

$$- mp_i\ ts_i(n)\} .$$

Now, clearly, the gradient of $h(\hat{\underline{r}})$ is given by:

$$\nabla h(\hat{\underline{r}}) = \begin{matrix} \sum_{i=1}^{m} u_i^*(1) - [s_o^*(1) - k_o u_o^*(1)] \\ \vdots \\ \sum_{i=1}^{m} u_i^*(N) - [s_o^*(N) - k_o u_o^*(N)] \\ \sum_{i=1}^{m} u_i(1) - d_o^*(1) \\ \vdots \\ \sum_{i=1}^{m} u_i^*(N) - d_o^*(N) \end{matrix}$$

where, again, $\underline{s}^*$ and $\underline{u}^*$ are the optimal trajectories and optimal controls, respectively, associated with the result of the inner minimization for a cost vector equal to $\hat{\underline{r}}$.

Therefore, if $\underline{s}^*$ and $\underline{u}^*$ are the optimal trajectories and optimal controls associated with equation (1) for $\underline{r} \in C$ then the dual function $h(\underline{r})$ may be written as:

$$h(\underline{r}) = f[\underline{s}^*,\underline{u}^*] + \langle \underline{r}, \nabla h(\underline{s}^*,\underline{u}^*) \rangle \quad \text{for} \quad \underline{r} \in C$$

Clearly since, as discussed before, there are only a finite number of solutions to the minimization of equation (1) and since, as just shown, the dual function behaves linearly with $\underline{r}$ for any of those solutions, the dual function $h(\underline{r})$ should be piecewise linear. That is, in the scalar case it should be formed by straight line segments like:

In the general case a 2(N+1)-dimensional vector would have to be considered and consequently h would be mapped to a 2(N+1)-dimensional space, i.e.

$$R^{2(N+1)} \rightarrow R$$

However, the form of $h(\underline{r})$ remains essentially the same. That is, instead of $h(\underline{r})$ being formed by segments of straight lines as in the one dimensional case, in this case it would be defined by a number of intersecting hyperplanes in $\hat{n}$ dimensional space where:

$$\hat{n} = 2(N+1) + 1$$

Clearly, the straight lines that correspond to the one-dimensional cost vector case could also be considered as two-dimensional hyperplanes.

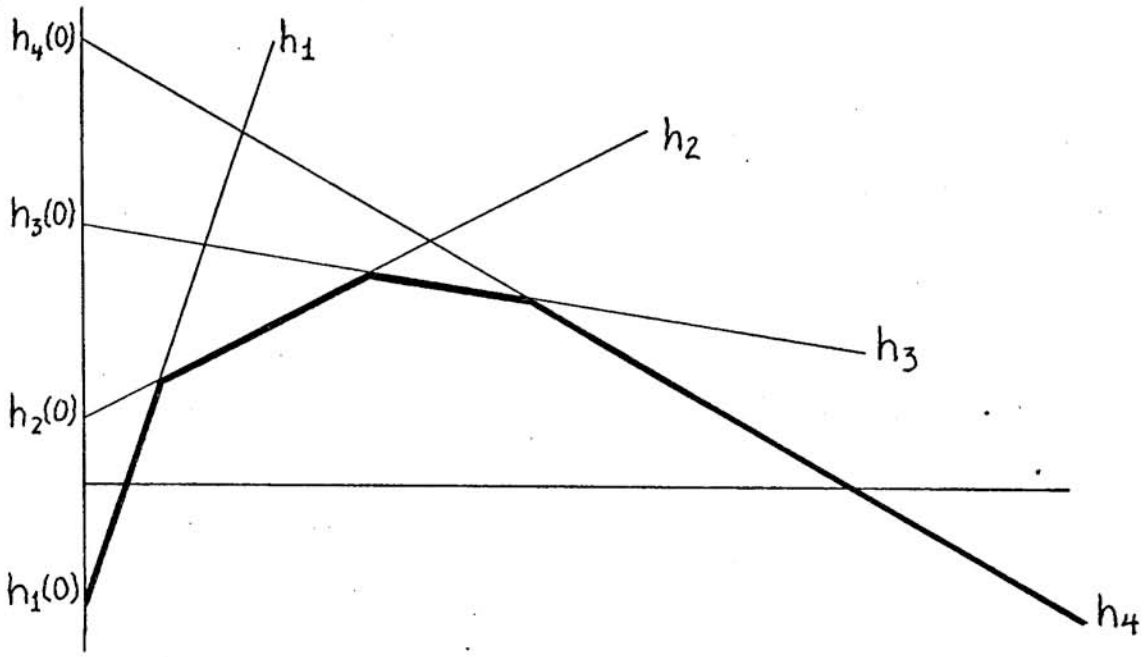
Going back to the one-dimensional cost vector it should be clear that evaluating the gradients and the value of the Lagrangian functions associated with each possible solutions to the inner problem for a given value of r (say $r=\phi$) it is possible to calculate the value of the dual function h for any allowable value of r. Recalling that h(r) was defined to be the minimum value among all possible Lagrangians for any value of r, it is clear that h(r) is just, simply, the upper lower bounds of the set of straight lines given by:

$$h_1 = h_1(r_o) + r\,\nabla h_1$$
$$\vdots$$
$$h_n = h_n(r_o) + r\,\nabla h_n$$

for any allowable value of r, where $h_i(c_o)$ is the value of L associated

with the i^{th} optimal trajectory (i=1,...,n) evaluated at $r=r_o$. And the ∇h_i's are the corresponding gradients associated with each one of these possible optimal trajectories.

For instance, if there are only four possible solutions to the inner problem with $h_i(0)$'s and ∇h_i's as plotted below, h(r) would be the thick line:



From these arguments, it should be clear that if h(r) were to be discontinuous, at least one of the possible optimal trajectories would have to have an infinite gradient associated with it. Now, in order to have a component of the gradient to be infinity, it would be necessary to have

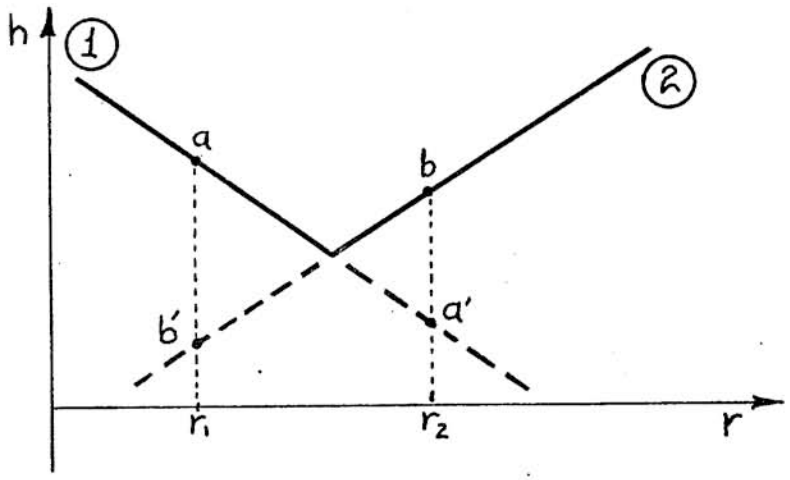
$$\sum_{i=1}^{m} u_i^* - (x_o^* + k_o u_o^*) = \infty$$

or:

$$\sum_{i=1}^{m} u_i^* - d_o = \infty$$

which is obviously impossible since both the optimal trajectories and optimal controls are constrained to lay within certain finite limits. Therefore, by contradiction, h(r) has to be continuous.

In fact, not only must h(r) be continuous, but also concave. In order to see this more clearly assume that h has a non-concave region like:



Clearly, point "a", which corresponds to the Lagrangian associated with solution 1 evaluated at r_2, and "b", which corresponds to the value of the Lagrangian of solution 2 evaluated at r_1, yield smaller values of h than the supposedly minimum values "b" and "a", respectively. Consequently, it may be concluded that h(r) cannot have a region with a configuration like the one assumed above. Obviously, this argument can be generalized to the point that if h(r) is to satisfy the minimality condition, its slope must be monotonically decreasing with "r", proving in this way that h(r) has to be concave.

Before trying to extend these arguments to the $\hat{n}$-dimensional cost vector, it might be helpful to remember that due to the "forced"

discretization of the optimal controls and optimal trajectories resulting from the inner minimization (equation (1)), it is obtained that the number of possible solutions to this inner problem is finite. That is, the solution to the inner problem does not change continuously with changing $\underline{r}$. In fact, as was explained before, for any $\underline{r}_o$ there is always a non-empty neighborhood C for which the optimal trajectories/optimal controls remain constant. Furthermore, it was argued that since the gradient of $h(\underline{r})$ associated with a given solution is independent of $\underline{r}$, if the values of the gradient and the Lagrangian, evaluated at a given point, are known for a particular solution to the inner problem, then it is possible to evaluate L associated with that solution for any value of $\underline{r}$; that value being the value taken by the hyperplane that passes through the known point and is perpendicular to the gradient.

Clearly, knowing the values of L associated with all possible solutions to the inner problem with their respective gradients, $h(\underline{r})$ may be constructed by just searching for the minimum value among all possible Lagrangians for each value of $\underline{r}$. Obviously, doing this in practice is a pretty hopeless case since there might be billions of different solutions to the inner minimization. But what is important to realize is that the dual function h is a multi-phase polyhedron bounded from above by a set of intersecting hyperplanes in $\hat{n}$-dimensional space. For obvious reasons, this argument shall not be supplemented with a diagram, but the author is nevertheless confident that the point has been gotten across properly.

Obviously, the question now is: is $h(\underline{r})$ continuous, and if so, is it convex?

As in the one dimensional case discussed before, in order to have a discontinuity in $h(\underline{r})$ it is necessary to have at least one optimal trajectory that yields a value of L equal to infinity for any value of $\underline{r}$. Clearly, since both the states and the controls are constrained to lay within a closed bounded convex set, the Lagrangian function cannot be unbounded at any point but at $\underline{r}=\infty$. Therefore, $h(\underline{r})$ must be continuous for all points within the allowable region of $\underline{r}$.

Now is $h(\underline{r})$ concave? This question shall be approached by exploring the function along an arbitrary direction $\underline{v}$.

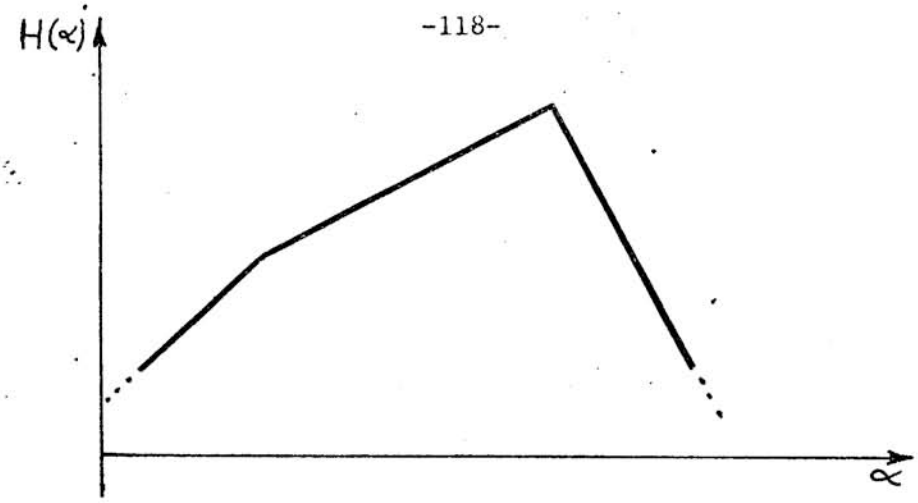
By evaluating $h(\underline{r})$ along a direction $\underline{r}$ it is possible to construct a two dimensional plot with respect to the step-size α along the given direction. That is, a scalar function $H(\alpha)$ can be defined such that:

$$H(\alpha) = h(\underline{r}_o + \alpha\, \underline{v})$$

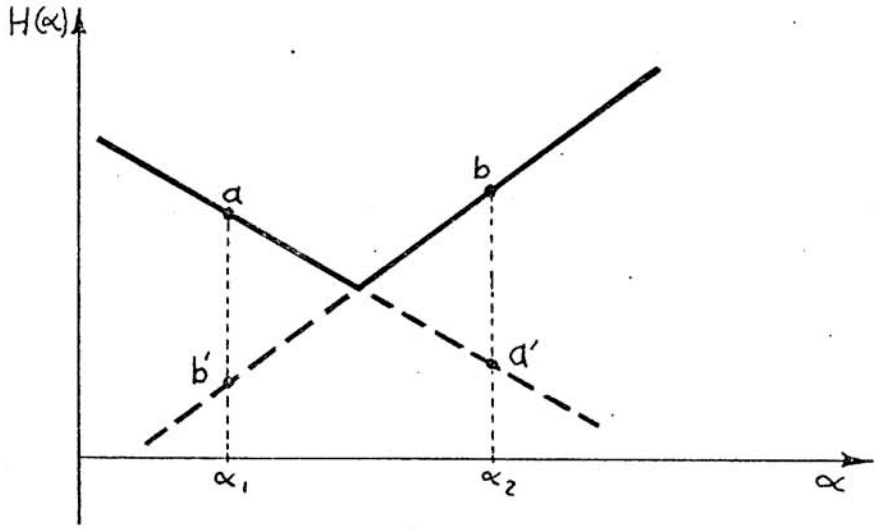
where $\underline{r}_o$ is an arbitrary point in the domain of h.

Now, clearly, the function $H(\alpha)$ is the function formed by the intersection of the dual function $h(\underline{r})$ with a hyperplane that passes through $\underline{r}_o$ and contains the vector $\underline{v}$.

Recalling that $h(\underline{r})$ is formed by a set of intersecting hyperplanes, it is clear that $H(\alpha)$ should have the form of a broken straight line as in the one dimensional case discussed before. That is, $H(\alpha)$ should look something like:



Using similar arguments as were used in the two dimensional case, it can be proved that $H(\alpha)$ has to be concave. That is, let us assume $H(\alpha)$ is not concave by allowing it to have a section like:



Now point "b'" belongs to the same inner minimization solution as that of "b" and "a'" belongs to the same of "a". Since if a given value of $\underline{r}$ corresponding to a given value of α is admissible for a given solution it is also admissible for any other solution, points "a'" and "b'" are as feasible as points "a" and "b" violating in this way the very definition of the dual function h. Namely, that $h(\underline{r})$ is equal to the minimum value

among the possible solutions for any admissible value of r. That is

$$h(\underline{r}) = \min_{u,d_o} L(\underline{u},d_o,\underline{r})$$

subject to certain constraints. Therefore, by contradiction, $H(\alpha)$ may not have non-concave regions as the one assumed in the above diagram. Now this argument can clearly be extended to the point that if the definition of $h(\underline{r})$ is not to be violated, $H(\alpha)$ must have a monotonically decreasing slope with increasing α. Therefore, $H(\alpha)$ must be concave.

Recalling that no constraints were imposed on the direction $\underline{v}$, it is obvious that these conclusions hold for any feasible direction $\underline{v}$. Consequently $h(\underline{r})$ must be everywhere concave.

APPENDIX III

DIRECTION OF STEEPEST ASCENT

As explained in Appendix II, the geometrical shape of the dual function is that of a piecewise linear concave hull in the m+1 dimensional space. That is, it is formed by the intersection of a finite number of m+1 dimensional hyperplanes. From these considerations, it is not difficult to see that the extreme point of this function along any direction s will, in the general case, be a point c where the gradient does not exist. That is, the directional extreme point c will generally lay on the intersection of two or more of these m+1 dimensional hyperplanes. Consequently, the optimum direction of search (direction of steepest ascent) after the first cycle has been concluded is by no means obvious.

In the special case where the directional extreme point lays on the hyperline formed by the intersection of two hyperplanes, the direction of steepest ascent is either the gradient of one of the hyperplanes or the projection of this hyperline on the R^m space.

In order to see this more clearly consider the problem in a three dimensional framework: calling the two planes Plane 1 and Plane 2 with G1 and G2 being their respective gradients, it is clear that due to the concavity property of the function, the only possible cases that may be had are those given by Figures 1, 2 and 3.

Clearly, in case 1 the direction of steepest ascent is G1, in case 2 d is the optimum direction and, finally, in case 3 the best direction is that given by G2.

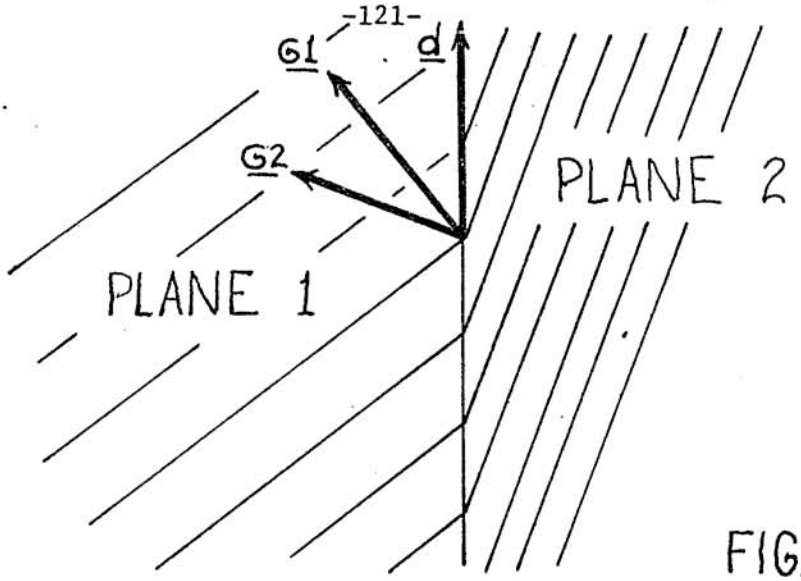


FIG. 1

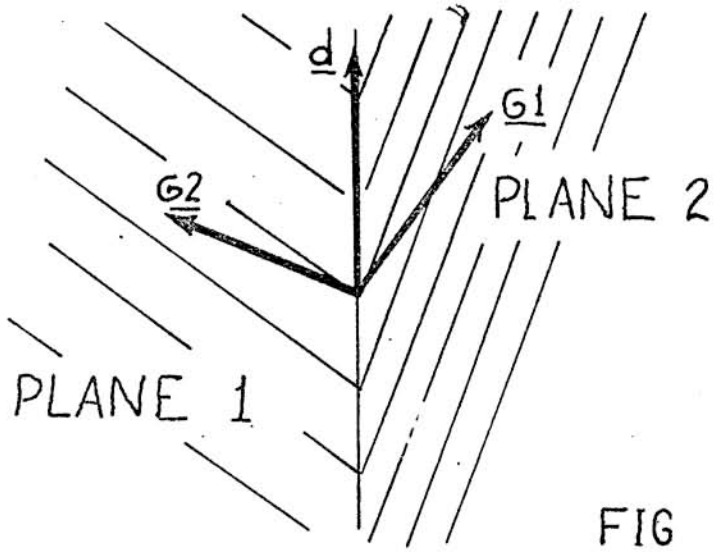


FIG 2

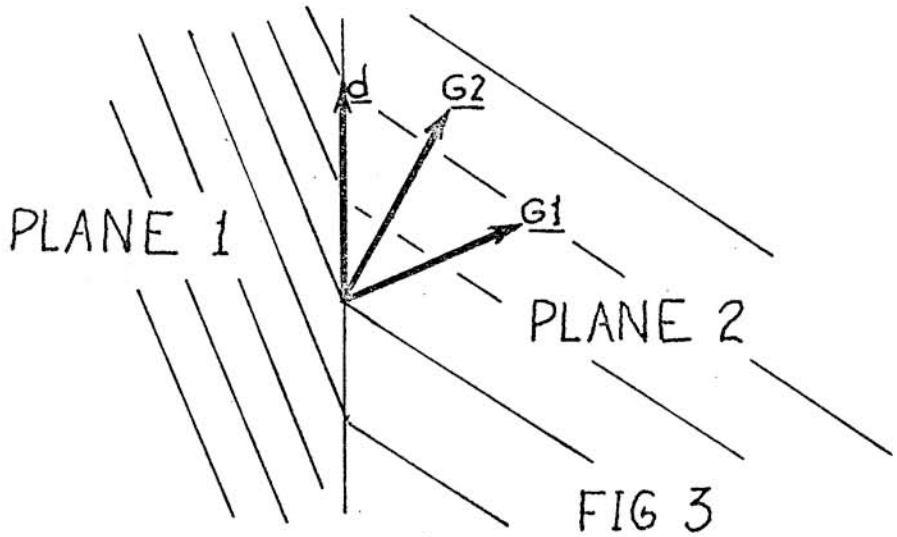


FIG 3

Even though it is difficult to interpret geometrically, the concept of the angle between two vectors in m-dimensional space has a very definite meaning. That is, defining α to be:

$$\alpha(\underline{v}_1,\underline{v}_2) = \cos^{-1} \frac{\underline{v}_1 \quad \underline{v}_2}{||\underline{v}_1|| \; ||\underline{v}_2||}$$

it may be shown that α has the same properties as that of a similar function that defines the angle between two vectors in three dimensional space. That is:

a) The angle between two parallel vectors equals zero.

b) The angle between perpendicular vectors equals 90 degrees.

c) $\alpha(\underline{v}_1,\underline{v}_2) + \alpha(\underline{v}_2,\underline{v}_3) \geq \alpha(\underline{v}_1,\underline{v}_3)$.

Using this angle concept in m dimensional space it is possible to characterize uniquely each one of the three cases given by Figures 1, 2, and 3 as a function of the angles between $\underline{G1}$ and $\underline{G2}$, $\underline{G1}$ and $\underline{d}$, and $\underline{G2}$ and $\underline{d}$; that is, as a function of $\alpha(\underline{G1}, \underline{G2})$, $\alpha(\underline{G1}, \underline{d})$ and $\alpha(\underline{G2}, \underline{d})$:

Case 1) $\alpha(\underline{G1}, \underline{d}) < \alpha(\underline{G2}, \underline{d})$

and $\alpha(\underline{G2}, \underline{G1}) < \alpha(\underline{G2}, \underline{d})$

Case 2) $\alpha(\underline{G1}, \underline{d}) < \alpha(\underline{G1}, \underline{G2})$

and $\alpha(\underline{G2}, \underline{d}) < \alpha(\underline{G1}, \underline{G2})$

Case 3) $\alpha(\underline{G2}, \underline{d}) < \alpha(\underline{G1}, \underline{d})$

and $\alpha(\underline{G1}, \underline{G2}) < \alpha(\underline{G1}, \underline{d})$

Therefore, a possible way of finding the direction of steepest ascent at a point that lays on an extreme hyperplane of $h(\underline{c})^*$ is to first identify the case with one of the three possibilities listed above. Then, as

explained before, if the particular situation at hand satisfies the conditions listed as Case 1, the direction of steepest ascent is that given by $\underline{G1}$, if the case corresponds to Case 2 then the optimal direction is $\underline{d}$ and finally if the case corresponds to Case 3 the best direction is that given by $\underline{G2}$.

Following similar guidelines, it is possible to find the directions of steepest ascent at extreme points formed by the intersection of more than two hyperplanes. However, this point shall not be studied further in this work.

APPENDIX IV

CONTROLLABILITY AND OBSERVABILITY ANALYSIS

Controllability

First of all, it is necessary to define what is to be understood by controllability in this type of problems.

Speaking in broad terms, one can define two basic types of controllability problems. One is the type usually associated with rocket guidance in which one is interested in making certain variables take particular values at a given point in time; this is usually referred to as the ballistic problem. The second type is the one in which one is concerned with making certain variables in a dynamical system follow a given trajectory and is usually referred to as the servomechanism controllability problem.

Intuitively, this inventory problem calls for a servomechanism controllability analysis. However, using the discrete model one might also view the problem as a ballistic one from point to point in the discrete sequence. Obviously, either approach is too constrictive since it is not absolutely essential to be able to cope with the demand at all points; actually the primary objective is the minimization of the cost function and not the fulfillment of the public demand. Nevertheless, either approach should give enough information to determine the limiting factors in the cases where there is a failure to meet the given demand, that is a lack of controllability. This constitutes valuable information

for the manager to help him justify reallocation of resources or future plant expansions.

Since the problem has been modeled in a discrete fashion, it is almost natural to study its controllability from a ballistic point of view, rather than as a servomechanism problem, as was initially suggested. Besides, the ballistic approach is considerably simpler to implement and perhaps even more enlightening than the servomechanism one.

The results from controllability theory that will be necessary for this analysis are stated below without proof. (For proofs see reference # 4.)

Controllability Theory

Theorem 1

An n-dimensional system defined by the set of equations:

$$\dot{\underline{x}}(t) = A\underline{x}(t) + B\underline{u}(t)$$

$$\underline{y}(t) = C\underline{x}(t)$$

is said to be controllable if the matrix

$$[B, AB, A^2B, \ldots, A^{n-1}B]$$

is of rank n.

Discretizing the system above we get

$$\underline{x}(n+1) - \underline{x}(n) = A\underline{x}(n) + B\underline{u}(n)$$

or

$$x(n+1) = (A+I)\underline{x}(n) + B\underline{u}(n) \qquad (1)$$

and

$$\underline{y}(n) = Cx(n) \qquad (2)$$

Surprisingly, the controllability condition for the discrete system turns out to be essentially the same as for the continuous system, i.e.

Theorem 2

Letting $A_o = A+I$, the system described by equations (1) and (2) is said to be observable if and only if:

$$[B, A_oB, A_o^2B, \ldots, A_o^{n-1}B]'$$

is of rank n.

As it was previously justified from pragmatical arguments, it is desired to perform a point to point controllability analysis. That is, it is desired to study the transfer capabilities between two points one time unit apart in the discrete sequence.

Unfortunately, the discrete version of the controllability condition, described above, can only be applied when terminal time is large with respect to initial time and optimally when $n \to \infty$. Therefore, this criterion is not applicable for the type of analysis that is desired to perform (transferability between points one time unit apart).

The structure of the problem is such that it is possible to identify m+1 quasi-independent subunits, the first being the first plant (intermediate product) and the rest being the individual subdivisions of the second plant (finished products).

Intuitively, one might be inclined to assume that the controllability of the subunits implies the controllability of the entire system. Actually, this is generally not true since there are constraints that tie these units together like the limited availability of resources for example. However, the converse is always true, that is, lack of

controllability of the subunits implies lack of controllability of the system. Therefore, it is always helpful to look at the subunits first before studying the controllability of the system as a whole.

As it turns out, examining the transfer capabilities of the subdivisions individually is a somewhat trivial operation. All the constraints involved in this phase of the analysis are linear and of the limiting type. That is, constraints of the form:

$$u_i(n) \leq u_i^* \quad \text{and} \quad s_i(n) \leq s_i^*$$

Actually, the problem is more difficult than just a simple check on the boundary values, as one might be led to believe from the discussion above. The reason for this is that a certain transfer between consecutive points is not possible does not necessarily imply that the given point is not reachable. For example, a given transfer might not be possible because of the limited production capacity, but perhaps if we allow for a higher inventory level on the previous point the production capacity limit might not be reached, getting in this way the previously unreachable point. Clearly, this procedure can be repeated as many times backwards in time until the given demand is reached. If in the process of going back in time time-zero is reached and the production is still short of the given demand, then the point might very safely be considered unreachable.

The important fact to consider here is not whether there are unreachable points or not since, as it was mentioned before, the main objective is to minimize the cost function and not to satisfy the public demand. What is actually intended to be obtained from this analysis is

information on which constraints and in what circumstances are the most frequent limiting factors that impede the fulfillment of the required demand. This information is clearly necessary for the justification and planning of relocations of resources and future plant expansions.

The iterative procedure proposed above to perform this type of analysis appears to be somewhat clumsy and inefficient since it requires a great number of operations. However, by virtue of its simplicity it is very easily implemented in a computer, making its application quite rewarding and worthwhile.

In conventional control theory terminology, observability usually refers to the question of whether the particular realization of the system being considered has all its state variables directly measurable in physical terms or not. However, the notion of observability that will be considered here is that used in modern control theory studies. That is, by an observability analysis, one is here interested not in whether the state variables of the system make any physical sense or not, but rather, on whether from the knowledge of the available inputs one is able to determine the values of the states or not.

The concepts of observability theory that will be necessary for this discussion are stated below without proof. (For proof see reference # 4)

Theorem 3

An n-dimensional system defined by the set of equations:

$$\dot{\underline{x}}(t) = A\,\underline{x}(t) + B\,\underline{u}(t)$$

$$\underline{y}(t) = C\,\underline{x}(t)$$

where A is an n×n matrix

B is an n×m matrix

and C is an r×n matrix

is said to be observable if and only if the matrix

$$[C, CA, CA^2, \ldots, CA^{n-1}]'$$

is of rank "n".

Discretizing the system above one gets:

$$\underline{x}(n+1) = (A+I)\underline{x}(n) + B\underline{u}(n) \tag{3}$$

$$\underline{y}(n) = C\underline{x}(n) \tag{4}$$

Again, the observability condition for the discrete system (above) results to be essentially the same as for the continuous system, i.e. Theorem 4

Letting $A_o = A+I$ the system described by equations (3) and (4) is said to be observable if the matrix:

$$[C, CA_o, CA_o^2, \ldots, CA_o^{n-1}]'$$

is of rank n.

Finally, from the controllability and observability analysis one is able to determine whether the particular realization of the system that is being considered is minimal or not. That is, one is able to determine whether the realization being considered is minimum dimensional or whether it is possible to find another realization with a smaller dimension than the one being considered. These results are stated in the following theorem also without proof (for proof, see reference # 4).

Theorem 5

A given realization is minimal if and only if it is both Controllable and Observable.

Clearly, for this analysis only the dynamic constraint equations should be considered. In the particular problem being studied in this thesis this constraint is given by the difference equation that determines the changes in inventory through time, i.e.

$$\underline{S}(n+1) = \underline{S}(n) + K\,\underline{U}(n) - \underline{D}(n)$$

Or, since matrix K is invertible

$$\underline{S}(n+1) = \underline{S}(n) + K\,\hat{\underline{U}}(n) \qquad (5)$$

where $\hat{\underline{U}}(n) = K^{-1}\underline{D}(n)$.

Using the terminology of equations (3) and (4) matrices A, B and C in equation (5) correspond to:

$A_o = I_{m+1}$ $(m+1) \times (m+1)$ identity matrix

$B = K$ where K is an $(m+1) \times (m+1)$ matrix with the production efficiencies of the different subdivisions in its diagonal.

$C = I$ since it may be assumed that all inventory levels are readily measurable.

Therefore, for this particular system it is obvious that both

$$[B, A_o B, A_o^2 B, \ldots, A^m B]'$$

and

$$[C, CA_o, CA_o^2, \ldots, CA^m]'$$

are of rank m+1; hence, from Theorems 2 and 4, the realization is both Controllable and Observable. Thus, from Theorem 5, it may be concluded that the realization at hand is minimum dimensional.

REFERENCES

1. George B. Danzig: "Linear Programming and Extensions," Princeton University Press, Princeton, New Jersey, 1963.

2. Leon S. Lasdon: "Optimization Theory for Large Systems," The Macmillan Company, London, 1970.

3. Willard I. Zangwill: "Nonlinear Programming: A Unified Approach," Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1969.

4. Roger W. Brockett: "Finite Dimensional Linear Systems," John Wiley and Sons, Inc., New York, 1970.