CHAPTER V

A Local Basis Simplex Method

1. Introduction

The two methods presented above, NDSDLP and PCSDLP, require the optimization of subproblems essentially independent of the main problem. This practice can be costly, leading to a great number of iterations. We describe below an adaptation of the Simplex Method for linear programs with stochastic structure. This method reduces the complications of stochastic linear programs by taking advantage of some fundamental properties of the basis. We call this approach a *local basis simplex method, LBSMPX* , because it relies on the near *square block triangularity* of the bases for these problems.

Block triangular linear programs, in general, have the form:

$$\begin{aligned} &\min \quad c_1x_1 + c_2x_2 + \cdots + c_Tx_T \\ &\text{subject to} \\ &\qquad A_{t1}x_1 + \cdots + A_{tt}x_t = b_t, t = 1, \ldots, T, \\ &\qquad\qquad x_t \geq 0, t = 1, \ldots, T. \end{aligned} \tag{1}$$

where $x_t \in R^{n_t}, b_t \in R^{m_t}, c_t \in R^{n_t}$, and the matrices A_{ij} are dimensioned accordingly.

The detached coefficient matrix is then:

$$A = \begin{pmatrix} A_{11} & & & \\ A_{21} & A_{22} & & \\ \vdots & \vdots & & \\ A_{T1} & A_{T2} & \cdots & A_{TT} \end{pmatrix} \tag{2}$$

and has dimension,$m \times n$, where $m = \sum_{t=1}^{T} m_t$ and $n = \sum_{t=1}^{T} n_t$.

Dantzig [15] introduced the idea of using an *artificial basis* for these programs in which square $(m_t \times m_t)$ basis blocks appeared along the diagonal. The *true basis* could be derived from the artificial basis by a set of *side conditions.* He observed that, because of the *persistence* property in dynamic linear programs, the additional computations would be few.

This concept of basis factorization and the use of *pseudo- basic* variables, as in Beale[6], have been applied in a variety of examples. Recent implementations are found in Kallio and Porteus[38],Perold and Dantzig[48],Fourer[22], and Propoi and Krivonozhko[52]. This last approach is close to the development here, using local bases as a factorization for multi-stage linear programs. A thorough and unified presentation of the relationships among basis factorization, partitioning, and decomposition can also be found in Winkler[67].

2. The Structure of the SDLP Basis

The program,SDLP, is another member of the class of block triangular linear programs as the following shows.

Lemma 1. *The program, SDLP, has the structure of a block triangular linear program, as in (1).*

Proof. Define

$$A_{tt} = \begin{pmatrix} A_t & & \\ & A_t & \\ & & \ddots \\ & & & A_t \end{pmatrix} \tag{3}$$

where A_t is repeated k_t times to correspond with each scenario in period t.

Define also

$$A^i_{t-1,t} = \begin{pmatrix} -B_{t-1} & & & \\ & -B_{t-1} & & \\ & & \ddots & \\ & & & -B_{t-1} \end{pmatrix} \tag{4}$$

where k_{t-1} repetitions of the matrix, $-B_{t-1}$, correspond to the possible outcomes of period $t-1$.

Then, we define $A_{t-1,t}$ as

$$A_{t-1,t} = \begin{pmatrix} A^1_{t-1,t} \\ A^2_{t-1,t} \\ \vdots \\ A^{k_t}_{t-1,t} \end{pmatrix}$$

The other A_{ij} matrices are void, so SDLP has the desired structure. ∎

The SDLP has other advantages that it shares with general multi-stage linear programs. In these programs, the number of additional columns required in finding the true basis from the artificial basis is limited. This bound means that storage requirements should not grow excessively with the problem size. This well-known property is stated in the following lemma:

Lemma 2. *For a dynamic linear program (a block triangular linear program where $A_{st} = 0$ for all $s > t+1$, ie. a staircase structure), the number of surplus and deficient columns in each block of the basis, B_{tt}, is bounded by:*

(i) $0 \leq l_1 \leq m_2$,

(ii) $-m_t \leq l_t \leq m_{t+1}, t = 2, \ldots, T-1$,

(iii) $l_T \geq -m_T$.

where $l_t + m_t$ is the number of basic columns in the basis from period t,and the basis has the form:

$$B = \begin{pmatrix} B_{11} & & & & \\ B_{21} & B_{22} & & & \\ 0 & B_{32} & B_{33} & & \\ & & & & \\ 0 & 0 & 0 & B_{TT-1} & B_{TT} \end{pmatrix}. \tag{5}$$

Proof. (ia). In order for B to be nonsingular B_{11} must have full rank. Therefore, $l_1 \geq 0$.

(ib) and (iia). $l_t \leq m_{t+1}$. Assume $l_t > m_{t+1}$, then B has greater than $m_t + m_{t+1}$ independent columns in period t, but the row rank of this block is less than or equal to $m_t + m_{t+1}$, a contradiction. Hence, $l_t \leq m_{t+1}$.

(iib) and (iii). $l_t \geq -m_t$. Assume $l_t < -m_t$, then there are greater than $\sum_{s=1}^{t} m_s$ independent columns in periods $1, 2, \ldots, t-1$, which again contradicts row rank, so $l_t \geq -m_t$.∎

The direct application of this lemma to the stochastic linear program, SDLP, would imply that for any period, t, there are at most $k_t \cdot m_{t+1}$ surplus columns in the basis. Because of the highly structured nature of SDLP, however, a much tighter bound can be found. We show this below in the following lemma. Murty [45] first observed this property for the two-stage case of programming under uncertainty with continuous distributions of the random variables in 1968.

Lemma 3. *For SDLP and for each scenario j in period t*

(i) $0 \leq l_1 \leq m_2$,

(ii) $-m_t \leq l_t^j \leq m_{t+1}, j = 1, \ldots, k_t; t = 2, \ldots, T-1$,

(iii) $-m_t \leq l_t^j; j = 1, \ldots, k_T$.

where l_t^j is the number of surplus columns in the basis for period t and scenario j, and m_t is the number of rows for a single scenario in period t.

Proof. (ia) This follows from Lemma 2.

(ib) and (iia). Assume that $l_t^j > m_{t+1}$. This implies that there are greater than $m_t + m_{t+1}$ independent columns) in the matrix :

$$D_t = \begin{pmatrix} A_t \\ -B_t \\ -B_t \\ \vdots \\ -B_t \end{pmatrix} \tag{6}$$

but, again, D_t has row rank $m_t + m_{t+1}$, a contradiction. Therefore, $l_t^j \leq m_{t+1}$.

(iib) and (iii). As before, the number of independent columns cannot exceed the row rank of the submatrix including that scenario, and the result follows.∎

From this result, we proceed to find an efficient implementation of the Simplex Method to the problem, SDLP, in which, a limited number of additional computations are used to perform the simplex routines of finding the true basis representation of a column, the cost row, and prices.

3. Finding the true basis representation of a column

To find the column representation, first define a square block triangular artificial basis for SDLP as

$$U = \begin{pmatrix} U_1 & & & & \\ V_1 & U_2^1 & & & \\ V_1 & & U_2^2 & & \\ & V_2^1 & & \cdot & \\ & V_2^2 & & & \cdot \\ & & & & \vdots \\ & & & V_{T-1}^{k_T - 1} & U_T^{k_T} \end{pmatrix} \tag{7}$$

where U_t^j consists of the rows of A_t^j for m_t independent columns in each period t, and V_t includes the corresponding rows in $-B_t$. For a constraint matrix, A, we also define

$$\alpha \equiv \{a_j \in A : a_j \text{ is a column in the artificial basis}\}.$$

We further define

$$\tau \equiv \{a_j \in A : a_j \text{ is a column in the true basis }\}.$$

The complements of α and τ are defined as $\overline{\alpha}$ and $\overline{\tau}$.

For every U_t, there also is a partition

$$U_t^j = \begin{pmatrix} P_t^j & \tilde{T}_t^j \\ \tilde{P}_t^j & T_t^j \end{pmatrix} \tag{8}$$

where P_t^j and T_t^j are square and non-singular, the columns in P_t^j are *pseudo-basic*, chosen from $\alpha \bigcap \overline{\tau}$, and T_t^j contains columns from the true basis. We define here

$$\psi \equiv \{k : a_j \in \alpha \bigcap \overline{\tau} \text{ where } a_j \text{ is the basic column for row } k\}.$$

To replace these columns in the basis at period t, there must be *surplus* basic variables from period $t-1$. We write these columns in terms of the

artificial basis as

$$U^{-1}\begin{pmatrix} A^j_{t-1} \\ -B^j_{t-1} \end{pmatrix} = \begin{pmatrix} P^j_{t-1} \\ Q^j_{t-1} \\ R^j_{t-1} \end{pmatrix} \tag{9}$$

where P^j_{t-1} consists of rows from A^j_{t-1} and is $m_{t-1} \times s^j_{t-1}$ (the number of surplus columns in period $t-1$ under scenario j). The rows corresponding to $-B^j_{t-1}$ are partitioned between Q^j_{t-1}, an $s^j_{t-1} \times s^j_{t-1}$ matrix, and R^j_{t-1}, a $\left(k_t \cdot m_t - s^j_{t-1}\right) \times s^j_{t-1}$ matrix.

The coefficient matrix,A, is then

$$U^{-1}A = \begin{pmatrix} & P_1 & & & & & \\ & & P^1_2 & & & & \\ & Q_1 & & \cdot & & & \\ & & & & P^{k_2}_2 & & \\ & & P^1_2 & & & & \\ & R_1 & & \cdot & & & \\ & & & & P^{k_2}_2 & & \\ & & Q^1_2 & & & & \\ I & & & \cdot & & & U^{-1}\tilde{A} \\ & & & & Q^{k_2}_2 & & \\ & & R^1_2 & & & & \\ & & & \cdot & & & \\ & & & & R^{k_2}_2 & & \\ & & & & \vdots & & \\ & & & & & \cdot \quad Q^{k_T}_{T-1} & \\ & & & & & \cdot & \\ & & & & & R^{k_T}_{T-1} & \end{pmatrix} \tag{10}$$

which, by row and column permutation, is

$$
\begin{pmatrix}
 & P_2^1 & & & & & & \\
Q_1 & & \cdot & & & & & \\
 & & & P_2^k & & & I & \\
 & Q_2^1 & & & & & & \\
 & & \cdot & & & & & \\
 & & & Q_2^{k_2} & & & & \\
 & & & & \cdot & & & \\
 & & & & & P_{T-1}^{k_{T-1}} & & \\
 & & & & & \vdots & & \\
 & & & & & Q_{T-1}^{k_{T-1}} & & \\
P_1 & & & & & & & U^{-1}\tilde{A} \\
R_1 & & & & & & & \\
 & & \cdot & & & & I & \\
 & & & \cdot & & & & \\
 & & & & & P_{T-1}^{k_{T-1}} & & \\
 & & & & \cdot & & & \\
 & & & & & R_{T-1}^{k_{T-1}} & &
\end{pmatrix}
\tag{11}
$$

The following allows us to use this partitioning for computations.

Lemma 4. *Relative to the artificial basis, U, the true basis, T, of SDLP has columns partitioned as in (11), where each Q_t^j is square and nonsingular.*

Proof. By the definition of the surplus columns in (9) and suitable permutations, we arrived at (11).

Next, define Q as the matrix, relative to the artificial basis, of surplus columns with rows in ψ, that is

$$Q = \begin{pmatrix} & P_2^1 & & \\ Q_1 & & \cdot & \\ & Q_2^1 & & \cdot \\ & & \cdot & P_{T-1}^{k_{T-1}} \\ & & & \cdot \\ & & & Q_{T-1}^{k_{T-1}} \end{pmatrix}$$

which is clearly nonsingular because U^{-1} and T are both nonsingular.

Now, index Q_t^j for all j and t along the diagonal as $Q_1, \ldots, Q_k$. Let Q_p be the first singular matrix in the list. Its row rank is $r_p < s_p$, the number of columns. In order for Q to be nonsingular, however, the row rank of the submatrix of the remaining columns, $\bar{r}_{p+1}$, is

$$\bar{r}_{p+1} > \sum_{j=p+1}^{k} s_j.$$

This violates the column rank, therefore, each Q_t^j is nonsingular.∎

Given that the Q_t^j's are nonsingular, we can proceed to eliminate the pseudo-basic columns from the basis. This procedure involves premultiplying $U^{-1}A$ by a product of matrices, $F_1, F_2^1, \ldots, F_{T-1}^{K_{T-1}}$. We first define

$$F_1 = \begin{pmatrix} Q_1^{-1} & & \\ & \cdot & \\ & & I \\ P_1 Q_1^{-1} & & \\ R_1 Q_1^{-1} & & \end{pmatrix} \tag{12}$$

and observe that

$$F_1U^{-1}A = \begin{pmatrix} I & \overline{P}_2^1 & & & Q_1^{-1} & \\ & Q_2^1 & & & & I \\ & & Q_{T-1}^{k_{T-1}} & & & \\ & . & & & -P_1Q_1^{-1} & \\ & & & & -R_1Q_1^{-1} & \\ & & & I & & \end{pmatrix} \tag{13}$$

We next define

$$F_2^1 = \begin{pmatrix} I & -\overline{P}_2^1(Q_2^1)^{-1} & \\ & (Q_2)^{-1} & \\ & -\overline{R}_2^1(Q_2^1)^{-1} & I \end{pmatrix}$$

and, in general,

$$F_t^l = \begin{pmatrix} I & -\overline{P}_t^l(Q_t^l)^{-1} & \\ & (Q_t^l)^{-1} & \\ & -\overline{R}_t^l(Q_t^l)^{-1} & I \end{pmatrix} \tag{14}$$

where $\overline{R}_t^l$ and $\overline{P}_t^l$ are the partitions of the surplus columns, relative to the previous F_s^j's. By repeated multiplication, the matrix relative to the true basis is found:

$$F_{T-1}^{k_{T-1}} \cdots F_1U^{-1}A = \begin{pmatrix} I & & \\ & F_{T-1}^k \cdots F_1 & F_{T-1}^{k_{T-1}} \cdots F_1U^{-1}\tilde{A} \\ & I & \end{pmatrix} \tag{15}$$

In order to facilitate finding the matrices, F_s^j, we note that the growth of nonzeroes in every period and scenario is limited. The following lemma states this.

Lemma 5. *After multiplication by $F_t^{l-1}, F_t^{l-2}, \ldots, F_t^1, \ldots, F_1$, the additional nonzero blocks in the surplus columns of scenario l in period t occur only in rows, for which, l's ancestor scenarios are basic.*

Proof. We proceed by induction on p, where we order all the blocks of surplus columns with $p = 1, \ldots, k$. For $p = 1$, the result is true trivially since no new nonzero blocks are added.

Assume this is true for all $p \leq n$. We look at the surplus columns in $n+1$, and assume, without loss of generality, that this is scenario l in period t. We then define

$$F_n \cdots F_1 \cdot \begin{pmatrix} P_t^l \\ \\ Q_t^l \\ \\ R_t^l \end{pmatrix} \equiv S_t^l \tag{16}$$

Next observe that every F_p in periods $1, \ldots, t-2$ has only identities in columns corresponding to P_t^l, Q_t^l, and R_t^l, so that only the F_{t-1}^l need be considered. If the scenarios are not ancestors of l at period t, then again by the hypothesis, they have no block entries that correspond to R_t^l, Q_t^l, or P_t^l, and so do not alter S_t^l.

Therefore, we have

$$S_t^l = F_{t-1}^l \cdot \begin{pmatrix} P_t^l \\ Q_t^l \\ \\ R_t^l \end{pmatrix} = \begin{pmatrix} & -P_{t-1}^l (Q_{t-1}^l)^{-1} & \\ I & (Q_{t-1}^l)^{-1} & \\ & & \\ & -R_{t-1}^l (Q_{t-1}^l)^{-1} & I \end{pmatrix} \begin{pmatrix} P_t^l \\ Q_t^l \\ \\ R_t^l \end{pmatrix} \tag{17}$$

and the only additional blocks occur where $-P_{t-1}^l (Q_{t-1}^l)^{-1} P_t^l$ is nonzero. By the hypothesis, the only nonzero blocks in S_{t-1}^l are in its ancestor rows from previous periods, so the only additional nonzero blocks in S_t^l will occur in these rows and the rows where columns are surplus in $t-1$, scenario l. This completes the induction. ∎

This lemma proves valuable in computing the columns relative to the true basis . Multiplications and storage are limited since only sections of each surplus block need to be considered. We next wish to compute the value of a column, z_i, where $z_i \in \overline{\tau}(t,l)$, the non –basic columns in period t and under scenario l.

We first observe that

$$F_n \cdots F_1 U^{-1} z_i = F_t^l \cdot F_{t-1}^{\bar{l}} \overline{z_i} \equiv \overline{z_i}^*, \tag{18}$$

where $\bar{l}$ is the first ancestor scenario of l, since for every F_j^p such that $j \notin \{t, t-1\}$ or $p \neq l$

$$F_j^p \overline{z_i} = \overline{z_i} \tag{19}$$

by the definition of F_j^p.

Next partition the components of the vector $\overline{z}_i$ as

$$\overline{a}_i^p \equiv (\overline{z}_{ji} : j \in \psi(t))$$

$$\overline{b}_i^p \equiv (\overline{z}_{ji} : j \in \psi(t+1))$$

$$\overline{a}_i^o = (\overline{z}_{ji} : j \in \overline{\psi}(t))$$

$$\overline{b}_i^o = (\overline{z}_{ji} : j \in \overline{\psi}(t+1)). \tag{20}$$

Then, $\overline{z}_i^*$ can be written as

$$\overline{z}_i^* = \begin{pmatrix} -P_{t-1}^l (Q_{t-1}^l)^{-1} \\ (Q_{t-1}^l)^{-1} \\ \\ -R_{t-1}^l (Q_{t-1}^l)^{-1} \end{pmatrix} \cdot \overline{a}_i^p + \begin{pmatrix} -P_t^l (Q_t^l)^{-1} \\ (Q_t^l)^{-1} \\ \\ -R_t^l (Q_t^l)^{-1} \end{pmatrix} \cdot \overline{b}_i^p + \begin{pmatrix} \overline{a}_i^o \\ \overline{b}_i^o \end{pmatrix} \tag{21}$$

With this representation, we form the following procedure, called **LBFTRN** (for local basis forward transformation) to find $\overline{z}_i^*$.

LBFTRN

Step 0. Identify an incoming column z_i. Find

$$\overline{z}_i = ((U_t^l)^{-1}a_i; (U_{t+1}^l)^{-1}(V_t^l((U_t^l)^{-1}a_i + b_i)) = (\overline{a}_i; \overline{b}_i) \text{as defined in (20).} \tag{22}$$

Step 1. Identify $\overline{a}_i^p$ and $\overline{b}_i^p$. Find

$$\begin{aligned} \overline{w} &\equiv (Q_{t-1}^l)^{-1}\overline{a}_i^p, \\ \overline{v} &\equiv (Q_{t-1}^l)^{-1}\overline{b}_i^p. \end{aligned} \tag{23}$$

Step 2. For all $j \in \bigcup_{s=1}^{t-2} \psi(s)$ (all preceding pseudo-basic columns) find

$$\overline{z}_{ji}^* = -\overline{P_{t-1}^l}(j,*) \cdot \overline{w} - \overline{P_t^l}(j,*) \cdot \overline{v} \tag{24}$$

for $j \in \psi(t-1)$,

$$\overline{z}_{ji}^* = \overline{w} - \overline{P_t^l}(j,*)\overline{v}, \tag{25}$$

for $j \in \psi(t)$,

$$\overline{z}_{ji}^* = \overline{v}, \tag{26}$$

for $j \in \bigcup_{s=1}^{t-2} \overline{\psi}(s)$,

$$\overline{z}_{ji}^* = -R_{t-1}^l(j,*)\overline{w} - R_t^l(j,*)\overline{v}, \tag{27}$$

and for $j \in \overline{\psi}(t-1)$,

$$\overline{z}_{ji}^* = -R_t^l(j,*)\overline{v}. \tag{28}$$

Step 3. For $j \in \overline{\psi}(t,l)$,

$$\overline{z}_{ji}^* = \overline{z}_{ji} + \overline{a}_{ji}^o, \tag{29}$$

and for $j \in \overline{\psi}(t+1, l)$

$$\overline{z}^*_{ji} = \overline{z}_{ji} + \overline{b}^o_{ji}. \tag{30}$$

LBFTRN results in the updated vector,$\overline{z}^*_i$. Therefore, from (21) and our previous results, the following proposition holds.

Proposition 1. *The representation of an incoming variable, z_i, relative to the current basis in SDLP can be found as $\overline{z}^*_i$ from the above procedures in LBFTRN.*

This routine allows for quick computations of an incoming column when there are few surplus columns. It requires current information of which rows belong to $\psi(t)$ and $\psi(t+1)$, the pseudo–basic columns in period t and period $t+1$, the representations of P^l_{t-1}, P^l_t, R^l_{t-1}, R^l_t, and the inverses of Q^l_{t-1} and Q^l_t. This storage requirement would be small, relative to the storage of the entire matrix, if few surplus columns are present.

4. Finding the Dual Prices and Pricing

The backwards transformation involved in computing the dual prices for SDLP can also be done more efficiently with the use of a square block triangular artificial basis. A savings here could be substantial because the pricing operation aften requires a majority of the effort in linear programming codes (viz., Fourer [22]). The method presented below requires only the present and previous scenario blocks' surplus column updates for computation of the reduced costs.

We assume again that we are in period t and at scenario l. The dual prices relative to the artificial basis are computed first, by transformations from the last period to the beginning. We define

$$\pi^l_T = c^u_T (U^l_T)^{-1} \text{for all } l = 1, \ldots, k_T. \tag{31}$$

and define π_t^l by the general recursion:

$$\pi_t^l = \left(c_t^u - [\sum_{l(t+1)} p_{t+1}^{l(t+1)} \pi_{t+1}^{l(t+1)}] V_t^l \right) (U_t^l)^{-1} \tag{32}$$

for all $l = 1, \ldots, k_t$, where $l(t+1)$ denotes all descendants of $l(t)$ at $t+1$ and $p_{t+1}^{l(t+1)}$ is the probability of each descendant.

Now, to find the prices relative to the true basis, we look for ρ such that

$$0 = c_j - \rho \cdot a_j \text{ for all } j \in \tau, a_j \in A. \tag{33}$$

We have from π defined in (32) and (33) that

$$0 = c_j - \pi \cdot a_j \text{ for all } j \in \tau \bigcap \alpha, \tag{34}$$

or, since $\pi = cU^{-1}$,

$$\begin{aligned} 0 &= c_j - c(U^{-1} a_j) \\ &= c_j - c_j \cdot I_j, \end{aligned} \tag{35}$$

where I_j is a unit column with identity in row j.

Thus, for c partitioned as

$$c^Q = (c_j : j \in \overline{\tau} \bigcap \alpha), \text{ and}$$

$$c^R = (c_j : j \in \overline{\tau} \bigcap \alpha), \tag{36}$$

we have from (35) and the definition of Q and R in (9),

$$\delta_j = c_j - \sum_{i \neq j} c^{Q_i} \cdot P_i - c^{Q_j} \cdot Q_j - c^R \cdot R_j \text{ for all } j \in \tau \bigcap \overline{\alpha}. \tag{37}$$

We seek next σ such that

$$0 = c_j - \sum_{i \neq j} (c_i^Q + \sigma_i) P_j - (c^{Q_j} + \sigma_j) Q_j - c^R R_j \text{ for all } j \in \tau \bigcap \overline{\alpha}. \tag{38}$$

We do this iteratively by finding first

$$\sigma_1 = \delta_1 Q^{-1} \tag{39}$$

and, in general,

$$\sigma_t^{l(t)} = [\delta_t^l - \sigma_t^{l(t-1)} P_t^l](Q_t^l)^{-1} \text{ for all } l = 1, \ldots, k_t. \tag{40}$$

We define ρ as

$$\rho = \begin{cases} c_j + \sigma_j, & \text{if } j \in \psi \\ c_j, & \text{if } j \in \overline{\psi}. \end{cases} \tag{41}$$

Hence, (33) holds by the construction of σ in (39) and (40) and the preservation of $\overline{c}^R = 0$.

The computation of ρ, using σ as defined in (39), allows pricing in an individual period t to involve only the inverse of Q_t^l. If pricing proceeds from period t to $t+1$, then, for a constant artificial basis, the previous prices need not be recomputed. This strategy may result in substantial savings by eliminating unnecessary multiplications for the other periods. It also saves on storage, since extraneous surplus block inverses can be ignored.

The local basis simplex procedures for finding dual prices are then:

LBBTRN

Step 0. For period t and scenario l, $c_\tau = (c_j(t,l) : j \in \tau \bigcap \overline{\alpha})$.

Step 1. Find $\delta_j(t,l)$ as defined in (37) for all $j \in \tau(t,l) \bigcap \overline{\alpha}$.

Step 2. Compute σ_t^l as in (40).

Step 3. Form ρ as in (41).

This procedure is then followed by LBPRCE which finds

$$\overline{c}_j = c_j - \rho \cdot a_j \tag{42}$$

for all $j \in \overline{\tau}(t,l)$. The incoming variable in this strategy is then chosen as the variable with the most negative reduced cost in that period and scenario. We define this as

$$\overline{c}_s(t,l) = \min_{j \in \overline{\tau}(t,l)} c_j - \rho \cdot a_j. \tag{43}$$

If $\overline{c}_s(t,l) \geq 0$, then the algorithm would proceed to the next scenario in period t, or, if $l = k_t$, the first scenario in period $t+1$ would be considered.

5. Updating the Pseudo-Bases and Surplus Blocks

After the incoming column is chosen in (43) a leaving column is selected by the minimum ratio criterion of the standard simplex method. This procedure, called CHUZR, finds

$$\frac{\overline{z}^*_{rs}}{\overline{\xi}_r} = \min_{\overline{z}_{is} > 0} \left(\frac{\overline{z}^*_{is}}{\overline{\xi}_i}\right), \tag{44}$$

where $\overline{z}^*_s$ is the representation of the incoming column found by LBFTRN and $\overline{\xi}_i$ is the current value of the right hand side in the i^{th} row.

The computations in CHUZR can also be reduced because nonzero entries in $\overline{z}^*_s$ are restricted to certain blocks as we discussed in Section 3. For updating the basis, s can be pseudo- basic or non-basic in period t (ie.,$s(t,l) \in \alpha \bigcap \overline{\tau}(t,l)$ or $s(t,l) \in \overline{\alpha} \bigcap \overline{\tau}(t,l)$), and s can enter the true artificial basis in period t or become surplus basic in period $t+1$ (that is, $r \in \tau(t-1,l) \bigcap \overline{\alpha}, \tau(t+1,l) \bigcap \alpha, \tau(t+1,l) \bigcap \overline{\alpha}$,or $\tau(t+1,l) \bigcap \alpha$). We discuss these cases below :

U1. $r \in \tau(t-1,l)$.

In this case s replaces a surplus column from the previous period. To update, we remove a column and row from $Q(t-1,l)$ and update the corresponding list of pseudo-vectors in period t. We also update $U(t,l)^{-1}$ by replacing the pseudo-basic column, $\psi(r)$, that corresponded to r with s.

This is done by adding an elementary matrix to the eta file of $U(t,l)^{-1}$ as in standard simplex codes. (Note that if $s \in \alpha$, the artificial basis remains unchanged.)

U2. $r \in \tau(t,l) \bigcap \alpha$.

Here, s replaces a currently true basic column in the artificial basis. If $s \in \alpha$, then we must replace the row in $Q(t,l)$ for which s is basic with the row for which r was basic. The artificial basis is unchanged. If $s \in \overline{\alpha}$, then we need only update the artificial basis, $U(t,l)$, as in **U1.**

U3.$r \in \tau(t+1,l) \bigcap \overline{\alpha}$.

In this case, s replaces a surplus column which is basic in the next period. We maintain the same artificial basis and update the surplus block,$Q(t+1,l)$. To do this, the column occupied by r in $Q(t+1,l)$ is replaced by the corresponding row entries of s. This is performed easily by premultiplying by an elementary matrix for this pivot.

U4. $r \in \tau(t+1,l) \bigcap \alpha$.

In this case, s replaces a basic column in the next period. This involves adding a column and row to $Q(t+1,l)$. The new row has entries from each of the surplus basic columns. The list of pseudo–basic variables in the next period must also be updated with the addition of the row for which r was basic. Again, the artificial basis remains unchanged.

The updating procedure can be confined to the current local artificial bases and current and following pseudo–bases. This property enables us to store only the present and following bases for updating purposes.

6. The Algorithm

The previous sections have presented the basic routines for a local basis simplex method. We discuss below the method's basic strategy and im-

plementation. The algorithm is similar to the nested decomposition approach of Chapter III in that it alternately follows a forward and backward procedure through the periods. This strategy is also similar to dynamic programming since the algorithm proceeds from one local optimum to the next. The basic method, call LBSMPX, follows.

LBSMPX

Step 0. Find an artificial basis. Do this by solving first

$$\begin{aligned} &\min\ c_1 x_1 \\ &\text{s. t. } A_1 x_1 = \xi, \\ &\qquad x_1 \geq 0, \end{aligned} \tag{45}$$

and proceeding to solve

$$\begin{aligned} &\min\ c_t x_t^l \\ &\text{s.t. } A_t x_t^l = \xi_t^l + B_{t-1} x_t^{l(t-1)}, \\ &\qquad x_t^l \geq 0, \end{aligned} \tag{46}$$

for all $(t,l), t = 1,\ldots,T, l = 1,\ldots,k_t$. If any of these programs has no feasible solution, use the last infeasible solution basis as U_t^l. Otherwise, we store each locally optimal basis as U_t^l and form the artificial basis from these matrices.

Step 0. Set $t = T-1, l = 1, NDRCTN = \text{'}BACK\text{'}, CFLAG = \text{'}NO\text{'}, MSTAT = \text{'}YES\text{'}, ITNO = 0$.

Step 1. Invert the current basis and find all basic variables values. This procedure, called INVERT, is equivalent to performing LBFTRN on the right hand side. If all variables have feasible values, $MSTAT = \text{'}YES\text{'}$. Else, $MSTAT = \text{'}NO\text{'}$.

*Step 2.*Form a Phase I objective row, if $MSTAT = \text{'}NO\text{'}$. Else, use the objective row,c_t. This routine is FORMC.

Step 3. Call LBBTRN for (t,l) to find the current prices. Set $ITNO = ITNO + 1$.

Step 4. Call PRICE for (t,l). If $\bar{c}_s \geq 0$, check basic variable values. If there exists $x(t', l')$ infeasible, set $MSTAT = 'NO'$, and

(a) If $t = T, l = k_T, NDRCTN = 'FORE'$,and $CFLAG = 'NO'$, if $MSTAT = 'YES'$, the solution is optimal,stop. If $MSTAT = 'NO'$ and the objective value, $x(1,1) > 0$,then the solution is infeasible, stop. If $x(1,1) = 0$, then, set $MSTAT = 'YES'$, and go to Step 2.

(b) If $t = 1, l = 1, NDRCTN = 'BACK'$, and $CFLAG = 'NO'$, then if $MSTAT = 'YES'$, the solution is optimal, stop. Else, if the objective $x(1,1) > 0$, then there is no feasible solution, stop. If $x(1,1) = 0$, then set $MSTAT = 'YES'$ and go to Step 2.

(c) If $t = T, l = k_T, NDRCTN = 'FORE'$, and $CFLAG = 'YES'$,set $t = T - 1, l = 1, NDRCTN = 'BACK', CFLAG = 'NO'$,set $MSTAT = YES$. If $ITNO < MAXIT$(the maximum number of iterations between reinversions),go to Step 2, else go to Step 1.

(d) If $t = 1, l = 1, NDRCTN = 'BACK'$, and $CFLAG = 'YES'$, then set $t = 2, l = 1, NDRCTN = 'BACK', CFLAG = 'NO', MSTAT = 'YES'$. If $ITNO < MAXIT$, go to Step 2. Else, go to Step 1.

(e) If $t < 1$ and $l < k_t$, then set $l = l + 1$. If $ITNO < MAXIT$, go to Step 2,else go to Step 1.

(f) If $1 < t < T, l = k_t, NDRCTN = 'BACK'$, set $t = t - 1, l = 1$, go to Step 2.

(g) If $1 < t < T, l = k_t, NDRCTN = 'FORE'$, set $t = t + 1, l = 1$, go to Step 2.

If $\bar{c}_s < 0$, set $CFLAG = 'YES'$.

Step 5. Call LBFTRN.

Step 6. Call CHUZR. If no r is found (ie., all $\overline{z}^*_{is} \leq 0$), then the solution is unbounded, stop.

Step 7. Call UPDATE to perform U1, U2, U3, or U4, depending on r, go to Step 2.

The preceding method leads to an optimal, unbounded, or infeasible solution as the following theorem states.

Theorem 1. *Assuming nondegeneracy (or suitable resolutions by choice of outgoing variable in CHUZR (see Dantzig [17])), the method, LBSMPX, terminates in a finite number of steps with an optimal solution, an unbounded feasible solution or an infeasibility criterion.*

Proof. Each pass through Step 7 results in a decreased objective value under nondegeneracy. Since there are a finite number of bases, the algorithm can pass Step 7 at most a finite number of times. The method would, therefore, terminate as in the simplex method as long as it takes at most a finite number of steps between passes of Step 7.

The algorithm does not return to Step 7 for each iteration that Step 4 results in (c), (d), (e), (f), or (g). However, in these cases, the algorithm progresses to the next scenario or period in either the forward or backward phase. Therefore, conditions (a) or (b) in Step 4 must be met after each phase.

When (a) or (b) are encountered in Step 4, the algorithm proceeds in the opposite direction, if Step 7 was ever encountered in the last pass through every scenario (the case, $CFLAG = 'YES'$). If no improvement was made, that is, if Step 7 was not passed in the last phase, the algorithm terminates. Therefore, LBSMPX always terminates in a finite number of steps. ∎

As in the Simplex Method, variations in LBSMPX may be used. The

most negative pricing strategy in PRICE, for instance, can be altered. We can also choose to proceed to the next period with a criterion other than $\bar{c}_s(t,l) \geq 0$. $\bar{c}_s(t,l) \geq -\epsilon$ could be used with ϵ decreasing in size as the optimal solution is approached.

LBSMPX uses the standard simplex techniques by efficiently partitioning the basis. As in the nested decomposition approach, NDSDLP, and the piecewise method, PCSDLP, LBSMPX concentrates on different sections of the basis one at a time. It differs with the previous methods, however, by continuously reflecting changes in the entire problem. The advantage of this property is that the global solution reflects the local optimization more quickly. The disadvantages, however, are that the method requires more computational effort in maintaining these instantaneous changes. These relative computational requirements are, again, discussed more closely in Chapter VII.

Chapter VI

The Relationship to

Dynamic Programming

1. Introduction

In Chapter I, the dynamic programming formulation for the general stochastic dynamic linear program, SDLP, was presented. In principle, SDLP could be solved exactly by this method. Since the random vectors, ξ_t, were allowed to have continuous distributions, the program at stage t was an infinite dimensional optimization problem. To avoid this complication, we approximated the distribution with discrete valued random vectors, ξ_t^i, and formulated a linear program, the solution of which we discussed above.

We considered linear programming techniques, but we could have used dynamic programming to solve this discrete distribution problem. In a production example, Beale, Forrest, and Taylor [7] estimated that four state variables in a time period could be handled. For larger problems, they concluded, an approximation must be used.

We chose the linear programming formulation because of the historically good performance of the Simplex Method and simplex–based algorithms. Using these algorithms, large problems with over a thousand rows and columns can be solved easily (see White [66] for a discussion of computational experience with large–scale linear optimization). Dynamic programming techniques using the standard computational procedure are generally limited to problems with six state variables and six decision variables (see Larson and Casti [41]). Advanced techniques can, however, be implemented

for larger problems in order to approximate and then refine the state and policy space. These methods prove very valuable for general transition equations and objective functions (see Larson [39]), but linear programming methods are most commonly used for problems with linear constraints and objectives. They do not require quantization of the state space or knowledge of the range of state variable values along the optimal path. Linear programming is, therefore, often more efficient than dynamic programming–type algorithms.

Our methods in Chapter III and IV actually combine these two techniques, although we have presented them as linear optimization strategies. In the following sections, a standard dynamic programming formulation of the stochastic linear program will be presented and its advantages will be discussed. We will then show that the piecewise method, PCSDLP, and the nested decomposition approach, NDSDLP, are simply different versions of general dynamic programming.

2. The Quantized State Space Approach

A standard simplification in dynamic programming is *quantization*. SDLP can be formulated in this manner by first creating a discrete approximation of the state space. This will enable us to form a recursion at every stage t and for every state y_t. We first define

$$y_t \equiv B_t x_t. \tag{1}$$

and quantize this vector as $\{y_t^1, y_t^2, \ldots, y_t^{l_t}\}$. The random right–hand sides will again have discrete values, $\{\xi_t^1, \xi_t^2, \ldots, \xi_t^{k_t}\}$.

Now, the following backward algorithm, BDP, can be used.

BDP

Step 0. For all y_T^i, find

$$z_T(y_{T-1}^i) = \mathrm{E}_{\xi_T}[z_T(y_{T-1}^i, \xi_T)], \tag{2}$$

where

$$\begin{aligned} z(y_{T-1}^i, \xi_T) = \ & \min \ c_T x_T \\ & \text{subject to} \\ & A_T x_T = y_{T-1}^i + \xi_T, \\ & x_T \geq 0. \end{aligned} \tag{3}$$

Set t=T. Go to Step 1.

Step 1. Set $t = t - 1$. For all y_{t-1}^j, find

$$z_t(y_{t-1}^j) = \mathrm{E}_{\xi_t}[z_t(y_{t-1}^j, \xi_t)], \tag{4}$$

where $z_t(y_{t-1}^j, \xi_t) = \min_{y_t^k}\{z_t((y_{t-1}^j, \xi_t), y_t^k)\}$, where

$$\begin{aligned} z_t((y_{t-1}^j, \xi_t), y_t^k) = \ & \min \ c_t x_t + z_{t+1}(y_t^k) \\ & \text{subject to} \\ & A_t x_t = \xi_t + y_{t-1}^j, \\ & -B_t x_t = y_t^k, \\ & x_t \geq 0. \end{aligned} \tag{5}$$

Step 2. If t=2, go to Step 3. Else, go to Step 1.

Step 3. Find $z_1 = z_1(0, b_1) = \min_{y_1^j} z_1((0, b_1), y_1^j)$, where $z_1((0, b_1), y_1^j)$ is as defined in (5) for $y_0 = 0$, $\xi_1 = b_1$. Stop.

Proposition. *The algorithm, BDP, converges in a finite number of steps to an optimal solution, $x_t^*(y_{t-1}^j)$, for $z_t(y_{t-1}^j)$, at every stage t of the program SDLP, under the following assumptions :*

I. The support of the random vector, ξ_t, is $\Xi_t = \{\xi_t^1, \xi_t^2, \ldots, \xi_t^k\}$.

II. The values of the inventories from period t are $B_t x_t$, where $B_t x_t \in Y_t = \{y_t^1, y_t^2, \ldots, y_t^l\}$.

Proof. The result follows directly from Bellman's Principle of Optimality and the finiteness of the Simplex Method for linear programming. ∎

BDP requires the solution of a great number of linear programs in Step 1. The storage requirements may be prohibitive for a large or detailed state space, Y_t. BDP does, however, have the advantage of solving small subproblems and of following a single pass to optimality recursively from period t back to 1. The method may even be implementable when one has a great deal of advance information about the admissible states in the solution. For more general problems, however, more efficient ways to characterize the state space are necessary. In the next section, we show that the algorithms, PCSDLP and NDSDLP, of Chapters III and IV are such methods, employing approximations for a continuous state space.

3. Relation to the Nested Decomposition Method

In this analysis, we consider the optimization problem, DP(t), at some time t :

$$z_t(y_{t-1}) = \mathrm{E}_{\xi_t}[z_t(y_{t-1}, \xi_t)], \tag{6}$$

where $z_t(y_{t-1}, \xi_t) = \min_{y_t} z_t((y_{t-1}, \xi_t), y_t)$ and

$$
\begin{aligned}
z_t((y_{t-1}, \xi_t) = \ & \min\ c_t x_t + z_{t+1}(y_t) \\
& \text{subject to} \\
& A_t x_t = y_{t-1} + \xi_t, \qquad (7) \\
& -B_t x_t = y_t, \\
& x_t \geq 0.
\end{aligned}
$$

Here, we would like to have an explicit representation of $z_{t+1}(y_t)$, which we approximated with the quantizations above. One way to find $z_{t+1}(y_t)$ is to exploit its convexity properties. The nested decomposition and piecewise approaches do this, as we show below.

For the nested decomposition approach, the following problem at stage t is solved instead of DP(t) :

$$
\begin{aligned}
& \min\ c_t x_t \quad + \theta_t \\
& \text{subject to} \\
& A_t x_t \quad = y_{t-1} + \xi_t, \\
& -(\overline{\pi}^k_{t+1} B_t) x_t + \theta_t \geq \rho^k_{t+1}, k = 1, \ldots, p, \qquad (ND(t)(a)) \\
& (-\sigma^l_{t+1} B_t) x_t \quad \geq (\sigma^l_{t+1} \tilde{\alpha}_{t+1}), l = 1, \ldots, q, (ND(t)(b)) \\
& \qquad x_t \geq 0,
\end{aligned}
\qquad (ND(t))
$$

where the constraints ND(t)(b) keep x_t feasible and ND(t)(a) forms an outer linearization of the convex function, $Q(x_t)$, as we showed in Chapter 3. The following lemma then shows the equivalence of DP(t) and ND(t).

Lemma 1. *For all t, $1 \leq t \leq T$, x^*_t is the optimal solution of $z_t(y_{t-1}, \xi_t)$ if and only if x^*_t solves:*

$$
\begin{aligned}
& \min \qquad c_t x_t + Q(x_t) \\
& \text{subject to } A_t x_t = y_{t-1} + \xi_t, \qquad (8) \\
& \qquad x_t \geq 0.
\end{aligned}
$$

Proof. For $t = T$, we have

$$\begin{aligned} Q_T(x_{T-1}) &= \mathrm{E}[Q_T(x_{T-1}, \xi_T)] \\ &= \mathrm{E}_{\xi_T}[\min c_T x_T | A_T x_T = \xi_T + B_{T-1} x_{T-1}, x_T \geq 0] \\ &= z_T(B_{T-1} x_{T-1}). \end{aligned} \tag{9}$$

At $t = T - 1$,

$$Q_{T-1}(x_{T-2}, \xi_{T-1}) = \min c_{T-1} x_{T-1} + Q_T(x_{T-1}) \tag{10}$$

subject to

$$\begin{aligned} A_{T-1} x_{T-1} &= \xi_{T-1} + B_{T-2} x_{T-2}, \\ x_{T-1} &\geq 0, \end{aligned}$$

which, by (9), is equivalent to :

$$\begin{aligned} \min \quad & c_{T-1} x_{T-1} + z_T(y_{T-1}) \\ \text{subject to} \quad & \\ & A_{T-1} x_{T-1} = \xi_{T-1} + B_{T-2} x_{T-2}, \\ & B_{T-1} x_{T-1} = y_{T-1}, \\ & x_{T-1} \geq 0. \end{aligned} \tag{11}$$

Therefore, the functions $z_{T-1}(y_{T-2}, \xi_{T-1})$ and $Q_{T-1}(x_{T-2}, \xi_{T-1})$ are identical and have correspondingly the same optimal solutions, x^*_{T-1}. The result follows by induction on t.∎

Lemma 1 shows that the cutting plane method used in the nested decomposition approach is equivalently a method for finding the function, $z_t(y_{t-1})$, in DP(t). In doing this, the constraints in ND(t)(a) linearize the convex function in the neighborhood of y_{t-1}.

In other words, at stage $t + 1$, for a given y^j_t, we find

$$
\begin{aligned}
z_{t+1}(y_t^j) &= \mathrm{E}_{\xi_t}[z_{t+1}(y_t^j, \xi_{t+1})] \\
&= \mathrm{E}_{\xi_t}[\pi_{t+1}^j(\xi_{t+1})(y_t^j + \xi_{t+1})], \\
&= \overline{\pi}_{t+1}^j y_t^j + \rho_{t+1}^j, \\
&= (\overline{\pi}_{t+1}^j B_t) x_t^j + \rho_{t+1}^j.
\end{aligned} \tag{12}
$$

By convexity and for $(\overline{\pi}_{t+1}^j B_t)x_t + \rho_{t+1}^j$ a support of $z_{t+1}(B_t x_t)$, we obtain

$$
z_{t+1}(B_t x_t) \geq (\overline{\pi}_{t+1}^j B_t)x_t + \rho_{t+1}^j. \tag{13}
$$

Now, we write DP(t) as

$$
\begin{aligned}
\min \quad & c_t x_t + \theta_t && (DPP(t)) \\
& A_t x_t = y_{t-1} + \xi_t, && (DPP(t)(a)) \\
& z_{t+1}(B_t x_t) \leq \theta_t, && (DPP(t)(b)) \\
& x_t \geq 0,
\end{aligned}
$$

and observe that, if (x_t, θ_t) is such that

$$
\theta_t < (\overline{\pi}_{t+1}^j B_t)x_t + \rho_{t+1}^j, \tag{14}
$$

then (x_t, θ_t) is infeasible in DPP(t). This corresponds (see Figure 1.) to finding a point within the feasibility space of the previously generated cutting planes that is not feasible for z_{t+1}. Another plane is added at the new x_t value and DPP(t) is solved again. The process is repeated until (x_t^*, θ_t^*) is found such that $\theta_t^* = z_{t+1}(B_t x_t^*)$. The following lemma results:

Lemma 2. *If (x_t^*, θ_t^*) is an optimal solution to ND(t) and $\theta_t^* = z_{t+1}(B_t x_t^*)$, then x_t^* is an optimal solution of DP(t).*

Proof. (x_t^*, θ_t^*) is optimal for the reduced constraints in ND(t) and x_t^* is feasible for DP(t) since $\theta_t^* = z_{t+1}(B_t x_t^*)$, therefore, x_t^* is optimal for DP(t).∎

From this lemma, NDSDLP can be seen as a method for finding an optimal solution to the dynamic program, DP(t). We further observe that the discreteness of the ξ_t distribution was not necessary for this development. The convexity of $z_{t+1}(y_t)$ is sufficient for the outer linearization to properly bound the objective function. The next section describes how PCSDLP uses the piecewise linearity of $z_{t+1}(y_t)$ to make local definitions of the dynamic programming recursion.

4. Relation to the Piecewise Method

For $\Xi_t = \{\xi_t^1, \xi_t^2, \ldots, \xi_t^k\}$, from Lemma 1, it follows that

$$\begin{aligned} z_{t+1}(B_t x_t) &= \sum_{j=1}^{k} p^j z_{t+1}(B_t x_t, \xi_t^j) \\ &= \sum_{j=1}^{k} p^j Q(x_t, \xi_t^j) \\ &= Q(x_t). \end{aligned} \tag{15}$$

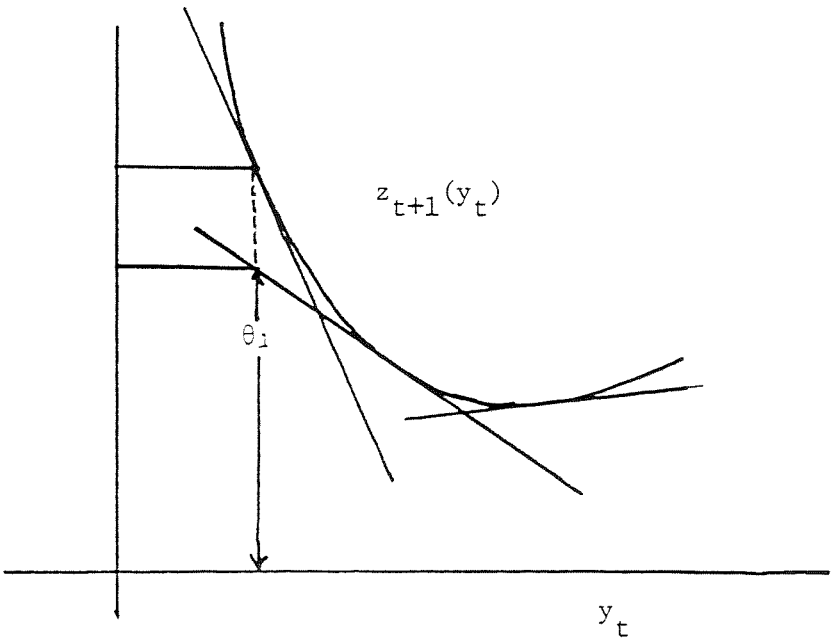

Figure 1. Outer linearizations.

Hence, z_t and Q have the same properties as functions of x_t. This implies that z_t is piecewise linear in x_t, as we showed for $Q(x_t)$ in Chapter 4. Therefore, we can write DP(t) as

$$\begin{aligned}\min\ c_t x_t \quad &+\textstyle\sum_{j=1}^{k} p^j c_{t+1}[(A_{t+1}^{B_j})^{-1}(B_t x_t+\xi_{t+1}^j)] \\ &+z_{t+2}(B_{t+1}[(A_2^{B_j})^{-1}(B_t x_t+\xi_{t+1}^j)])\end{aligned} \qquad (PW($$

subject to

$$\begin{aligned} A_t x_t &= y_{t-1} + \xi_t, \\ (A_2^{B_j})^{-1} B_t x_t &\geq -(A_2^{B_j})^{-1}\xi_{t+1}^j, j = 1,\ldots,k, \\ x_t &\geq 0. \end{aligned}$$

We proceed as in Chapter 4 to write PW(t) as

$$\begin{aligned} \min \quad & \tilde{c}_t x_t + k' \\ \text{subject to} \quad & \\ & A_t x_t = y_{t-1} + \xi_t, \\ & -\tilde{B}_t^j x_t \geq \tilde{\xi_{t+1}}^j, j = 1,\ldots,k, \\ & x_t \geq 0, \end{aligned} \qquad (PWW(t))$$

where k' is a constant, and $\tilde{c}_t$, $\tilde{B}_t^j$, and $\tilde{\xi_{t+1}}^j$ are the values relative to the subproblem bases as in (3.12) and (3.15).

Here, the function $z_t((y_{t-1}, \xi_t), y_t)$ is limited to the linear piece, for which, $-\tilde{B}_t^j x_t \geq \tilde{\xi_{t+1}}^j$ for all j. The piecewise algorithm proceeds by looking for improvements from x_t^k, the kth optimal solution found to PWW(t). A sequence,

$$z_t((y_{t-1}, \xi_t), y_t^1) > z_t((y_{t-1}, \xi_t), y_t^2) > \cdots > z_t((y_{t-1}, \xi_t), y_t^k), \qquad (16)$$

is found, such that, if $z_t((y_{t-1}, \xi_t), y_t^{k+1}) = z_t((y_{t-1}, \xi_t), y_t^k)$ for all feasible directions from y_t^k, then the x_t^k corresponding to $B_t x_t^k = y_t^k$ is optimal for

DP(t), and we can proceed to find $z_t(y_{t-1})$. (See Figure 2, and note that the optimum is determined for z_t only, so that z_{t+1} need not be at a minimum.)

From this development, we have

Lemma 3. *If the piecewise linear approach, PCSDLP, terminates with* $z_t((y_{t-1}, \xi_t), y_t^{k+1}) = z_t((y_{t-1}, \xi_t), y_t^k)$, *then the associated* x_t^k *is an optimal solution to* $z_t((y_{t-1}, \xi_t))$ *in DP(t).*

4. Conclusion

In Sections 3 and 4, we showed that the piecewise and nested decomposition methods found values of x_t that minimize the function z_t for a given state y_{t-1}. The following theorem, which follows directly from Lemmas 3 and 4, states this result.

Theorem 1. *The nested decomposition and piecewise methods presented in Chapters 3 and 4 obtain optimal values of* $z_t(y_{t-1}, \xi_t)$ *at each stage t of the*

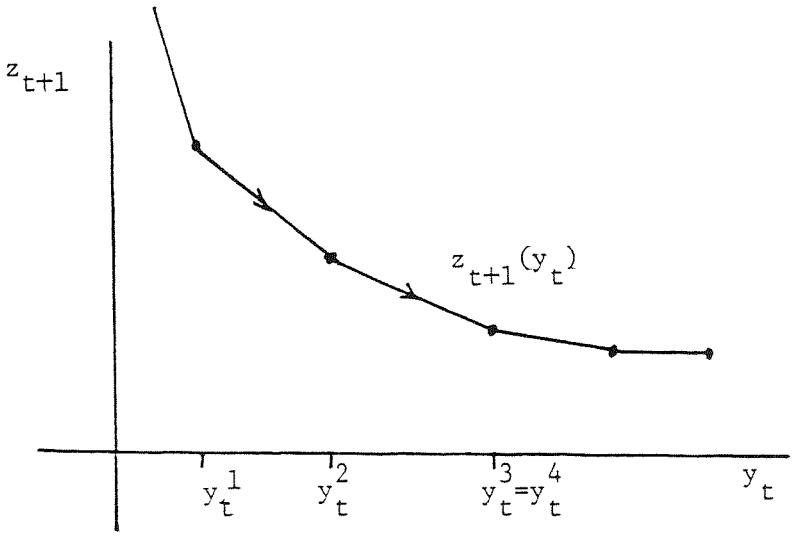

Figure 2. At y_t^3, $z_t((y_{t-1}, \xi_t), y_t^3) = z_t((y_{t-1}, \xi_t), y_t^4)$.

dynamic programming procedure.

PCSDLP and NDSDLP, when applied to this backward procedure, must each start from some given state, y_t. The dynamic programming approach of enumerating these states may be valuable here. If we let y_t take on representative values and evaluate each of the associated $z_t(y_{t-1})$ separately before proceeding to stage $t-1$, some computational effort may be saved in both the nested decomposition and piecewise approaches.

CHAPTER VII

Computational Results

and

Conclusions

1. Introduction

In this chapter, we synthesize our previous development of alternative methods for reducing the complexity of problems under uncertainty. We also present some results from solving these stochastic dynamic linear programs. In this presentation, we wish to show that the difficulties in solving complex stochastic programs are not so great that one should ignore uncertainties. We will demonstrate that the stochastic solution can be evaluated without prohibitive complications and that our techniques may prove beneficial in this evaluation.

Our strategy has been first to consider deterministic problems and then to extend them to the SDLP form. Our initial step in this approach dealt with the properties of the basis for different scenarios. In Chapter I, we showed that this analysis may result in finding an optimal deterministic solution. Beyond this, in Chapter 2, we showed that a bound on the value of the stochastic solution could be obtained before proceeding to solve SDLP. We next gave three algorithms for solving this problem. In the following sections, we present the computational properties of these algorithms in solving some examples of SDLP.

Section 2 describes our computational results on representative test problems and compares the algorithms' performance with standard linear pro-

gramming. In evaluating these methods, we also consider their usefulness in models of actual phenomena. In Section 3, we discuss areas in which stochastic linear programming has been applied, and we show that the solution of uncertainty models can lead to prudent decision–making. Following this analysis, in Section 4, we state our conclusions on solving stochastic linear programs and present directions for future research in this area.

2. Computational Results

The algorithms of Chapters III, IV, and V have been programmed in FORTRAN on the SCORE DECSystem 20 computer at Stanford University. This system is useful for observing the properties of algorithms, but it is not designed for solving the extremely large problems modeled in some stochastic dynamic linear programs. Our goal was not to draw definitive conclusions about the performance of each algorithm, but, instead, to observe some of the algorithms' properties, so that, on systems with larger processors, they may be implemented with a better understanding of their capabilities.

Since current linear programming packages take great advantage of the *sparsity* (the relative number of zero entries in the coefficient matrix), we compared our algorithms' performance with that of LPM–1, a program written by J. A. Tomlin and revised by G. Kochman at the Systems Optimization Laboratory. LPM–1 employs compact storage of the non–zero entries in the coefficient matrix, performs an LU–decomposition of the basis, retains the basis inverse in product form, and uses a merit counting sort for maintaining sparsity in the inverse (see Pfefferkorn and Tomlin [49]). This procedure has proved efficient in solving linear programs because the coefficient matrices are characteristically very sparse. Since staircase linear programs (and SDLP's) have considerable sparsity in their structure, it has been conjectured [47]

that efficient handling of these elements implicitly uses the structure and that, except on very large problems beyond the capacity of standard linear programming codes, no improvement can be made upon the packaged codes. Our procedures have an advantage over these codes because they only involve small sections of the program during an optimization. They do not require storage of the entire program parameters and, thus, may handle larger problems. Apart from this advantage, we want to observe our algorithms' performance on smaller problems that simplex codes handle easily.

The nested decomposition method, NDSDLP, was programmed in FORTRAN as NDST1. This program includes the basic algorithm presented in Chapter III, but does not include modifications, such as column passing. For solving individual linear programs at each node, NDST1 uses the procedures of LPM-1. It saves lists of the current basic variables at each node, but it does not maintain the LU-decomposition for each node. Instead, it keeps only one *ETA file* (the elementary matrices in the product form of the basis inverse) for the current node and reinverts the basis each time it encounters a new node. NDST1 also includes lower and upper bound capabilities on each variable and, hence, does not require Step 2′ of NDSDLP.

The piecewise approach, PCSDLP, was programmed as PCST1. It includes the feasibility routines of NDST1 for finding a primal feasible solution of SDLP, and it follows the procedures of PCSDLP given in Chapter IV. It does not, however, have capabilities in saving lists of subproblem bases. Alternatively, it considers each subproblem basis individually and, therefore, may be forced to solve larger than necessary master problems. PCST1 also uses lower and upper bounds on all variables and checks for possible upper bound violations of the equation (IV.4a), as we mentioned above. For the case of artificial variables in the subproblem basic sets, the upper bounds are

always included as additional constraints.

Storage in PCST1 is larger than that in NDST1 because of the need to maintain both the last solution for ς^{μ} and the solution for the current auxiliary problem of $\varsigma^{\mu,*}$. For this reason, ETA files are maintained for both problems.

LBSMPX was implemented in the program LBS1. This program uses the basic procedures of LPM–1 for the artificial basis, with updated transformations for the surplus basis blocks. It follows the cyclical pricing strategy described in Chapter V and uses a "most negative" pricing criterion in each period. Pricing is restricted to that period as long as the least reduced cost is negative. Updates of the surplus basis blocks, Q_t^j, are performed according to U1, U2, U3, and U4 in Chapter V, and the inverse of this matrix is stored explicitly.

We have chosen a set of representative problems to show the algorithms' computational properties. The data for the test cases appear in Table 1.

TABLE 1

Program Parameters

	Number of Constraints (m)	Number of Variables (n)	Number of Nodes (k)	Density (ρ^*)
PA1	10	22	3	.15
RD1	10	25	3	.25
NG1	19	35	3	.28
PA2	22	50	7	.09
RD2	22	57	7	.14
PA3	46	106	15	.04
RD3	46	121	15	.07

* (number of non-zero elements)/(total number of elements)

These examples were chosen to represent the types of problems found in applications.

The first cases, RD1, RD2, and RD3, were selected from a control problem in Davis [20] and modified to reflect a production planning model as in Beale *et al* [7]. In this format, the expected present value of profits is maximized for a firm planning the production and storage of a number of products that require common resources. Their sales are determined by an uncertain demand. The decision variables in this model are :

$x_{i,t}$ — the amount of product i sold in period t,

$y_{i,t}$ — the amount of i produced in period t,

$s_{i,t}$ — the amount of product i in stock at the end of period t.

The linear program for this problem is

$$
\begin{aligned}
&\max \textstyle\sum_{t=1}^{T} \sum_{i=1}^{q} (p_{i,t}x_{i,t} - c_{i,t}y_{i,t} - k_{i,t}s_{i,t}) \\
&\text{s. t.} \qquad \textstyle\sum_{i=1}^{q} y_{i,t} \leq R_t, \text{ for all } t, \\
&\qquad\qquad s_{i,t-1} + y_{i,t} - x_{i,t} = s_{i,t}, \text{ for all } t \text{ and } i, \\
&\qquad\qquad 0 \leq x_{i,t} \leq d_{i,t}, \text{ for all } t \text{ and } i,
\end{aligned} \tag{1}
$$

where R_t is the total capacity in period t, $p_{i,t}$ is the selling price of product i, $c_{i,t}$ is the production cost of product i, $k_{i,t}$ is the holding cost of stock in i, T is the planning horizon, and q is the number of products. In this example, we included random uncertainties in $d_{i,t}$ and formulated (1) as in SDLP for optimizing the expected value of profits over the planning horizon.

The examples, PA1, PA2, and PA3, were contributed by P. Abrahamson [1] and represent the discrete approximation of a continuous time linear program. This program represents the optimal control of a particle subject to state space constraints. It is written as

$$
\begin{aligned}
&\min \qquad \textstyle\int_0^3 u(t)e^{\alpha t}dt \\
&\text{s.t.} \quad x(t) - \textstyle\int_0^t (u(s) - x(s))ds = 1, \\
&\qquad\qquad u(t) \leq 1, \\
&\qquad\qquad x(t) \leq .7 - .2t, \\
&\qquad\qquad x(t), u(t) \geq 0,
\end{aligned} \tag{2}
$$

where we considered uncertainty in the state to state transitions for $x(t)$. The application of uncertainty to continuous problems, as suggested by this model, is an important area for possible future applications.

The last example, NG1, is derived from the exhaustible resource model of Chapter I. It includes uncertainty in investment in different technologies. This example is included, because the basis must be changed significantly from the myopic first period solution to the optimal stochastic solution. This process requires more surplus basic variables in the first period and demonstrates

TABLE 2

Sample Times and Iterations

	LPM-1		NDST1		PCST1		LBS1	
	Time (CPUs)	No. of Iter.	Time (CPUs)	No. of Iter.	Time (CPUs)	No. of Iter.	Time (CPUs)	No. of Iter.
PA1	.18	12	.14	12	.19	12	.20	11
RD1	.30	15	.23	12	.42	13	.33	15
NG1	.48	18	.83	29	1.01	23	.56	17
PA2	.94	29	1.71	29	2.41	30	1.05	28
RD2	1.59	31	.91	32	1.34	33	1.73	31
PA3	4.01	61	3.39	68	5.61	79	4.96	64
RD3	7.13	65	4.89	67	7.64	71	6.72	59

some of the complications of stochastic programs.

The solution times and number of simplex iterations required for the optimal solution of these problems are presented in Table 2. The times are in CPU seconds for solutions from a "cold start"(an infeasible basis), and represent computing time excluding time for data input. The results show that, on small programs, our algorithms may be competitive with codes using sparsity techniques alone, such as LPM-1. This is especially significant because NDST1, PCST1, and LBS1 are experimental codes and are not programmed for maximum computing efficiency. Small problems, however, are not representative of actual applications, and experience with larger programs is mandatory for true comparisons. Again, we do not intend to prove the superiority of our methods, but only to show how they perform.

TABLE 3

Sample Times per Iteration

	LPM–1 Time/No. of Iterations	NDST1 Time/No. of Iterations	PCST1 Time/No. of Iterations	LBS1 Time/No. of Iterations
PA1	.015	.012	.016	.018
RD1	.020	.019	.032	.022
NG1	.027	.028	.044	.033
PA2	.032	.059	.080	.037
RD2	.051	.028	.041	.056
PA3	.066	.049	.071	.078
RD3	.110	.073	.108	.114

NDST1 performs best overall on these examples, but, where it performs poorly we can observe some of its properties. In NG1, a large number of cuts were required on the first period and many suboptimizations were performed, slowing its convergence to an optimum. We also see here that these suboptimizations required NDST1 to perform 25 percent more iterations than PCST1, which maintains subproblem optimality. This difficulty also appears in PA2 where the added effort of reinversion at each pass from node to node led to NDST1's having a larger average time per iteration than LPM–1 (see Table 3.). We note that the small growth of non–zeroes in the basis inverse of PA2 is a good example of LPM–1's advantage in performing backward and forward transformations very quickly. On the other examples, because NDST1 solves the smallest problems, it requires the least amount of time per

iteration despite reinversion.

An interesting property observed in the solutions by NDST1 concerns the set of basic variables in the master problem as cutting planes are added from the subproblems. In our examples, we often saw that one set of basic variables would remain constant in the master problem from iteration to iteration and that the additional basic elements, would be chosen from another small set of variables. The algorithm would choose a member of the additional set on one iteration, replace it with another member in the next iteration, and then bring the first element back into the basis at master–subproblem optimality. For X_B, the constant set of basic variables, and $\{x_1, x_2\}$, the additional basic variables, the basis in the master problem would follow the pattern:

Iteration	Basic Set
1	$\{X_B\}$
2	$\{X_B, x_1\}$
3	$\{X_B, x_2\}$
4	$\{X_B, x_1, x_2\}$– optimal.

It has been observed [1] that the deterministic problem often brings additional variables into the basis and then adjusts them in subsequent iterations without dropping them from the basis. This observation led to the column passing technique we discussed above. Our observations indicate that the set of columns passed forward to the subproblems should include columns that were in the basis but were deleted. This would allow for the entire set of columns to obtain the optimal weights in a single pass.

The behavior we have observed may be indicative of the hedging effect in stochastic problems. In our example above, x_1 may be included to optimize the first descendant scenario and x_2 may correspond to optimizing with

TABLE 4

Number of First Period Optimizations

	NDST1	PCST1
	Number of Passes	Number of Passes
PA1	6	4
RD1	8	7
NG1	14	11
PA2	8	6
RD2	9	7
PA3	12	10
RD3	15	13

the second subproblem. The algorithm tries each of these activities before deciding on the optimal combination in Iteration 4. By including each of these potentially basic variables in the set corresponding to the columns passed forward, we may avoid excessive optimization.

PCST1 performed consistently worse than NDST1 based on CPU time, but its strength of not requiring suboptimization appears in Table 4. This chart displays the number of times the master problem in period 1 was solved. PCST1 requires fewer of these optimizations and passes to the subproblems. It thus decreases the possibility of lengthy suboptimization. PCST1 solves, however, a larger master problem than NDST1, because it includes the degeneracies associated with repeated subproblem bases. The basis list approach presented in Chapter 4 should alleviate some of the

difficulty here.

LBS1 also maintains more information than is necessary and its use of the triangular basis differs only slightly from the factorization for sparsity in LPM–1. LBS1's additional bookkeeping is most evident on smaller programs and begins to be efficient for larger problems as we start to see in RD3.

These examples presented some of the properties of NDST1, PCST1, and LBS1. Knowledge of these attributes should be helpful in determining what method to use for a specific problem. To determine this, as we stated above, the deterministic scenarios should be evaluated first and then the stochastic problem should be solved. Because of its simplicity, we would recommend NDST1 as a first method to try, followed by PCST1, if convergence to optimality is slowed by excessive suboptimization. LBS1's implementation is more determined by efficient coding and its evaluation should be made only once its procedures have been efficiently coded for larger problems on faster processors.

3. Areas of Application

We have mentioned that stochastic dynamic linear programs may be found in many real–world contexts. In this section, we discuss some actual applications of these problems in decision–making. We mention these examples to demonstrate the range of possible implementations of our techniques.

The first use of stochastic linear programs was by Dantzig and Ferguson [21] on a problem of airline scheduling with uncertain passenger demand. Using a modification of the transportation simplex method, they demonstrated that the net expected costs for meeting the carrier's demand could be reduced from \$1,666,000 to \$1,524,000 by considering the complete distribution of

demand instead of only the expected values of those demands.

More recent uses of stochastic linear programs include Manne's analysis in [44] of the decision to develop breeder fission technology. Another analysis of nuclear fuel choices appears in Avi-Itzhak and Connally [4], wherein, they use the scenario approach of assuming decisions based on deterministic solutions and evaluating the expected cost of these decisions under different future scenarios.

Other areas of application are found in the investment management of bank portfolios. Aghili *et al* [2] evaluated the decisions of a small midwestern bank in allocating their assets among a set of securities and loan positions. They used the bank management's criteria for constraints and formulated a two-stage stochastic linear program with uncertain interest rates and bond yields. Their results using only financial and accounting constraints reflected the extreme point property of optimal linear program solutions. Their model would have placed almost all of the bank's portfolio into municipal bonds, forgoing all commercial loans and mortgages.

The difficulty Aghili *et al*, encountered arose from their use of the risk neutral linear objective function. To improve their results, they incorporated management's concerns into additional policy constraints and obtained solutions that were much closer to the bank's own decisions. From this analysis, management became aware of the need to consider different future economic environments. Previously, they had used only one forecast for the future but were now able to consider several scenarios. This openness to changing money conditions gave the bank management more options to consider in their policy decisions and led them to reshape some of their ideas on the bank's operations.

The bank balance sheet model is a good example of decision makers'

use of a stochastic model. Difficulties often arise in the implementation of a model's optimal solution, but, in Aghili *et al*'s experience, the model served to inform management, and it became an important policy instrument. Another example of this utility of stochastic linear models appears in Gaither's study [24] of commercial fishing seasons. That research demonstrated that substantial savings could be accrued by proper fishing regulation in Alaska, but modifying those regulations required difficult policy decisions that would not be immediately forthcoming. Despite these problems, Gaither's model also played an important role in demonstrating to the decision makers how they should structure their strategy and what concerns they should have.

4. Conclusion

We have presented several approaches for solving linear models that involve decisions made over time in uncertain environments. We have showed that the resulting stochastic dynamic linear programs can be analyzed through the use of deterministic scenario solutions and the subproblem solution methods of nested decomposition, piecewise path following, and local basis factorization. This analysis rests upon the fundamental properties of the basis in linear programming, the understanding of which is crucial in understanding the characteristics of a stochastic solution.

The stochastic solution succeeds because it allows for a regularization of the extreme point solutions that occur in different deterministic scenarios. This property should be carefully considered in evaluating any stochastic solution because it can be a powerful stabilizing force in determining optimal decisions. Within the framework of the stochastic model, a single decision can be chosen that will reflect each of the future scenarios included in the model.

The methods we presented for solving these SDLP's demonstrated credible performance on smaller examples. These results are encouraging for future implementations on large–scale problems. This experimentation on large models should be the major direction for future research.

The true test of our algorithms' benefits and of the usefulness of the SDLP program, in general, must come from actual policy models. Our presentation of examples from production planning, exhaustible resources, and portfolio management demonstrate that many applications of SDLP are possible. The inclusion of uncertainty into a linear model extends the range of decision factors and helps stabilize policy strategies in dynamic settings. These attributes may bring significant gains to the use of models in decision–making.

Our primary goal has been to show that, by exploiting the stochastic linear program's special structure, models may be able to include uncertain parameters without becoming hopelessly complex. By proceeding from deterministic scenarios through the stochastic solution, the modeler can be aware of the value of proceeding to more complex stages and can develop a better understanding of the nature of the deterministic linear programming solution and its relation to the stochastic strategy. To complete this analysis most effectively, the computational costs of the stochastic program must be reduced. We hope that our work will help make these reductions possible, thereby, enabling models to become more powerful tools in affecting current decisions in an uncertain world.

BIBLIOGRAPHY

[1] Abrahamson, P. G., "A Nested Decomposition Approach for Solving Staircase Structured Linear Programs", **Proceedings of the Workshop on Large–Scale Linear Programming**, IIASA, Laxenburg, Austria, June 2–6, 1980.

[2] Aghili, P., R.H., Cramer and H.W. Thompson, "On the applicability of two-stage programming models to small bank balance sheet management", *J. Bank Res. J.*, pp. 246-256.

[3] Arrow, K.J., S. Karlin and H. Scarf, **Studies in the Mathematical Theory of Inventory and Production**, Stanford University, Stanford, CA, 1958.

[4] Avi-Itzhak, B. and T.J. Connolly, "The plutonium issue", Technical Report SOL 78-24, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, CA, September 1978.

[5] Avriel, M. and A.C. Williams, "The value of information and stochastic programming", *OR* 18, No. 5, Sept.-Oct. 1970.

[6] Beale, E.M.L., "The simplex method using pseudo-basic variables for structured linear programming problems", pp. 133-148 in *Recent Advances in Math. Prog.*, R.L. Graves and P. Wolfe (eds.), McGraw-Hill, New York, 1962.

[7] Beale, E., J.J.H. Forrest and C.J. Taylor, "Multi-time period stochastic programming", in *Stochastic Programming*, M.A.H. Dempster (ed.), Academic Press, New York, 1980.

[8] Bellman, R., Dynamic Programming, Princeton University Press, Princeton, NJ, 1957.

[9] Benders, J.F., "Partitioning procedures for solving mixed-variables programming problems", *Numerische Mathematik* 4, pp. 238-252, 1962.

[10] Bereanu, B., "Some numerical methods in stochastic linear programming under risk and uncertainty", in *Stochastic Programming*, M.A.H. Dempster (ed.), Academic Press, New York, 1980.

[11] Bisschop, J. and A. Meeraus, "Matrix augmentation and partitioning in the

updating of the basis inverse", *Math. Prog.* **13**, No. 3, pp. 241-254.

[12] Chao, H.P., "Exhaustible resource models: the value of information", Technical Report, Stanford University, August 1979.

[13] Chao, H.P., Private communication, May, 1980.

[14] Charnes, A. and W.W. Cooper, "Chance-constrained programming", *Management Science* **6**, No. 1, October 1959.

[15] Dantzig, G.B., "Upper bounds secondary constraints and block triangularity in linear programming", *Econometrica* **23**, pp. 174-183, April 1955.

[16] Dantzig, G.B., *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, 1963.

[17] Dantzig, G.B., "Solving staircase systems", draft of September 12, 1980.

[18] Dantzig, G.B. and A. Madansky, "On the solution of two-stage linear programs under uncertainty",*Proceedings, 4th Berkeley Symp. on Mathematical Statistics and Probability* , U.C. Press, Berkeley, pp. 165-176, 1961.

[19] Dantzig, G.B. and P. Wolfe, "Decomposition principle for linear programs", *OR* 8, No. 1, pp. 101-111, Jan.-Feb. 1960.

[20] Davis, R., "New jump conditions for state constrained optimal control problems", Ph.D. dissertation, Stanford University, 1980.

[21] Ferguson, A. and G.B. Dantzig, "The allocation of aircraft to routes: an example of linear programming under uncertain demands", *Management Science* **3**, pp. 45-73, 1956.

[22] Fourer, R., "Solving staircase l.p.'s by the simplex method", Technical Report SOL 79-18, Systems Optimization Laboratory, Stanford University, Stanford, CA, November 1979.

[23] Gabbay, H., "Multi-stage production planning", *Management Science* **25**, No. 11, pp. 1138-1148, Nov. 1979.

[24] Gaither, N., "A stochastic constrained optimization model for determining commercial fishing seasons", *Management Science* **26**, No. 2, Feb. 1980.

[25] Garstka, S. and D. Ruthenberg, "Computations in discrete stochastic programs without recourse", *OR* **21**, PP. 112-122, 1973.

[26] Garstka, S. and R.J.B. Wets, "On decision rules in stochastic programming",

Math. Prog. **7**, No. 2, pp. 117-143, 1974.

[27] Geoffrion, A.M., "Elements of large-scale mathematical programming", *Management Science* **16**, No. 11, pp. 652-675, July 1970.

[28] Glassey, C.R., "Dynamic linear programs for production scheduling", *OR* **19**, pp. 45-56, 1971.

[29] Glassey, C.R., "Nested decomposition and multi-stage linear programs", *Management Science* **20**, pp. 282-292.

[30] Grinold, R.C., "A new approach to multi-stage stochastic linear programs", pp. 19-29 in *Stochastic systems: Modelling, Identification and Optimization*, R.J.B. Wets (ed.); also *Management Science* 8, 1975.

[31] Gunderson, H.S., J.G. Morris, and H.E. Thompson, "Stochastic programming without recourse: a modification from an applications viewpoint", *JORS* **29**, No. 8, pp. 769-778, August 1978.

[32] Ho, J.K. and A.S. Manne, "Nested decomposition for dynamic models", *Math. Prog.* **6**, pp. 121-140, 1974.

[33] Holt, C., F. Modigliani, J. Muth, and H. Simon, *Planning Production, Inventories, and Work Force*, Prentice-Hall, Englewood Cliffs, NJ, 1960.

[34] Hotelling, H., "The economics of exhaustible resources", *Journal of Political Economy* **39**, pp. 137-175, 1931.

[35] Huang, C.C., W.T. Ziemba and A. Ben-Tal, "Bounds on the expectation of a convex function of a random variable with applications to stochastic programming", *OR* **25**, pp. 315-325, 1977.

[36] Huang, C.C., I. Vertinsky, W.T. Ziemba, "Sharp bounds on the value of perfect information", *OR* **25**, pp. 128-139, 1977.

[37] Kall, P., 'Computational methods for solving two-stage stochastic linear programming problems", *Journal of Appl. Math. and Physics* **30**, pp. 261-271, 1979.

[38] Kallio, M. and E.L. Porteus, "Decomposition of arborescent linear programs", *Math. Prog.* **13**, No. 3, pp. 348-355, Dec. 1977.

[39] Larson, R.E., "A survey of dynamic programming computational procedures", *IEEE Trans. on Automatic Control* **AC-12**, no. 6, pp. 764-774, 1967.

[40] Larson, R.E. and W.G. Keckler, "Application of dynamic programming to the control of water resource systems", *Automatica* **5**, No, 1, pp. 15-26, January 1969.

[41] Larson, R.E. and J.L. Casti, *Principles of Dynamic Programming*, Marcel Dekker, Inc., New York, 1978.

[42] Madansky, A., "Inequalities for stochastic linear programming problems", *Management Science* 6, pp. 197-204, 1960.

[43] Manne, A.S., "Waiting for the breeder", *Review of Economic Studies Symposium*, pp. 47-65, 1974.

[44] Manne, A.S., and R.G. Richels, 'A decision analysis of the U.S. breeder reactor program", *Energy*, **3**, pp. 747-767, 1978.

[45] Murty, K.G., "Two-stage linear program under certainty: a basic property of the optimal solution", *Zeitschrift fuer Wahrscheinlicheitstheorie und Verwandte Gebiete* **10**, pp. 284-288, 1968.

[46] Nordhaus, W.D., "The allocation of energy resources", Brooklyn Paper on Economic Activity, **3**, pp. 529-576, 1973.

[47] Perold, A., Private communication, February 1980.

[48] Perold, A.F. and G.B. Dantzig, "A basis factorization method for block triangular linear programs", Technical Report SOL 78-7, Systems Optimization Laboratory, Stanford University, April 1978.

[49] Pfefferkorn, C.E. and J.A. Tomlin, "Design of a linear programming system for ILLIAC IV", Technical Report SOL 76-8, Systems Optimization Laboratory, Stanford University, April 1976.

[50] Phillips, L.D. and H.J. Otway, "Limitations to human judgment: implications for system modellers", *Energy systems Analysis International Conference*, Dublin, Ireland, October 9-11, 1979.

[51] Propoi, A., "Models for educational and manpower planning: a dynamic linear programming approach", RM-78-20, IIASA, Laxenburg, Austria, May, 1978.

[52] Propoi, A. and V. Krivonozhko, 'The simplex method for dynamic linear programs", RR-78-14, IIASA, Vienna, Austria, September 1978.

[53] Rosen, J.B., "Convex partition programming", in *Recent Advances in Mathematical Programming*, R.L. Graves and P. Wolfe (eds.), McGraw-Hill, New York,

1963.

[54] Samuelson, P.A., "Lifetime portfolio selection by dynamic stochastic programming", *Rev. Econ. Stat.* **51**, pp. 239-246, 1969.

[55] Stancu-Minasian, S. and M.J. Wets, "A research bibliography in stochastic programming", *OR* **24**, No. 61, pp. 1078-1119, 1976.

[56] Van Slyke, R. and R. Wets, "L-shaped linear programs with applications to optimal control and stochastic linear programs", *SIAM J. of Appl. Math.* **17**, pp. 638-663, 1969.

[57] Van Slyke, R. and R. Wets, "Programming under uncertainty and stochastic control", *J. SIAM Control* **4**, No. 1, pp. 179-193, 1966.

[58] Walkup, D. and R. Wets, "Stochastic programs without recourse", *SIAM J. of Appl. Math.* **15**, pp. 1299-1314, 1967.

[59] Walkup, D. and R. Wets, "Stochastic programs without recourse, II: on the continuity of the objective", *SIAM J. of Appl. Math.* **17**, pp. 98-103, 1969b.

[60] Wets, R., "Programming under uncertainty: the complete problem", *Z. Wahrscheinlicheitstheorie und Verwandte Gebiete* **4**, pp. 319-339, 1966.

[61] Wets, R., "Programming under uncertainty: the solution set", *SIAM J. of Appl. Math.* **14**, pp. 1143-1151, 1966.

[62] Wets, R., "Programming under uncertainty: the equivalent convex program", *SIAM J. of Appl. Math.* **14**, pp. 89-105, 1966.

[63] Wets, R., "Characterization theorems for stochastic programs", *Math. Prog.* **2**, pp. 166-175, 1972.

[64] Wets, R., 'Stochastic programs with fixed recourse: the equivalent convex program", *SIAM Review* **16**, No. 3, July 1974.

[65] Wets, R., "Stochastic programs with recourse: a basic theorem for multi-stage problems", *Zeitschrift fuer Wahrscheinlicheitstheorie und Verwandte Gebiete* **21**, pp. 201-261, 1978.

[66] White, W.W., " A status report on computing algorithms for mathematical programming", *ACM Computing Survey* **5**, No. 3, September 1973.

[67] Winkler, C., "Basis factorization for block-angular linear programs", Technical Report SOL 74-19, Systems Optimization Laboratory, Stanford University,

December 1974.

[68] Yudin, D.B., "Mathematical methods of control under uncertainty problems and methods of stochastic programming", Soviet Radio, Moscow, 1974.

[69] Ziemba, W.T., "Transforming stochastic dynamic programming problems with nonlinear programs", *Management Science* **1**, pp. 450-462, 1971.

[70] Ziemba, W.T., "Computational algorithms for convex stochastic programs with simple recourse", *OR* 8, pp. 414-431, 1970.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SOL 80-29	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SOLUTION METHODS FOR STOCHASTIC DYNAMIC LINEAR PROGRAMS		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) John R. BIRGE		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0267
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research - SOL Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research - Dept. of the Navy 800 N. Quincy Street Arlington, VA 22217		12. REPORT DATE December 1980 13. NUMBER OF PAGES 131
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

large-scale linear programming
stochastic programming
dynamic linear programs
decomposition
expected value of perfect information
basis factorization

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SEE ATTACHED

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SOL 80-29: Solution Methods for Stochastic Dynamic Linear Programs,
John R. Birge

Linear programs have been formulated for many practical situations that require decisions made periodically through time. These dynamic linear programs often involve uncertainties. Deterministic solutions of these problems may lead to costly incorrect decisions, and, when a stochastic solution is attempted the problem may become too large. In this report, we present methods for reducing the computational cost of these stochastic programs, and we show conditions under which the stochastic program need not be solved. Our methods are based on the large-scale programming techniques of decomposition, partitioning, and basis factorization.