MASTER

RECEIVED BY TIC MAY 7 1980

DEVELOPMENT OF SOFTWARE FOR
COMPUTER ASSISTED MODEL SIMPLIFICATION
FINAL REPORT

PREPARED FOR
THE DEPARTMENT OF ENERGY
UNDER CONTRACT NUMBER DE-AC01-79EI-10597
APRIL 25, 1980

PREPARED BY
DBS CORPORATION
1901 NORTH MOORE STREET
ARLINGTON, VIRGINIA 22209

DBS

CORPORATION
1901 North Moore Street
Arlington, Virginia 22209
703—527-9060

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

DEVELOPMENT OF SOFTWARE FOR
COMPUTER ASSISTED MODEL SIMPLIFICATION
FINAL REPORT

PREPARED FOR
THE DEPARTMENT OF ENERGY
UNDER CONTRACT NUMBER DE-AC01-79EI-10597
APRIL 25, 1980

PREPARED BY
✓ DBS CORPORATION — 951 1156
1901 NORTH MOORE STREET
ARLINGTON, VIRGINIA 22209

DISCLAIMER

This book was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

NOTICE

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for any third party's use or the results of such use of any information, apparatus, product, or process disclosed in this report, or represents that its use by such third party would not infringe privately owned rights.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISTRIBUTION LIST

Harvey J. Greenberg 5 copies
Project Manager
Office of Analysis Oversight & Access, AA, EIA
Federal Building, 12th & Pennsylvania Avenue, N.W.
M.S. 4530, Room 4522
Washington, D.C. 20461

George M. Lady 1 copy
Director, Office of Analysis Oversight & Access
Federal Building, 12th & Pennsylvania Avenue, N.W.
M.S. 4530, Room 4522
Washington, D.C. 20461

Department of Energy 1 copy
Technical Information Center (TIC)
Post Office Box 62
Oak Ridge, Tennessee 37830

ABSTRACT

This constitutes the final report of DBS Corporation on the model simplification project. The purpose of the project is to develop computer assisted model simplification. The contributions of DBS to this project were an initial overall project assesment (task 1), contributions to design principles and testing procedures (task 2), specific experimental designs (task 3), and initial test results (task 4).

The main contributions of DBS to this project were in the area of LP matrix scaling, and particularly in the potential usefulness of shadow price information for model simplification. An algorithm for obtaining approximate shadow price information was developed and subjected to initial small-scale testing with promising results.

TABLE OF CONTENTS

	PAGE

ABSTRACT	i
INTRODUCTION	1
RETROSPECTIVE ASSESSMENT	3
TASKS 3 & 4 REPORT	4
THE ALGORITHM	6
MATHEMATICAL FORMULATION	8
NOTES ON BASIC TEST PROGRAM	12
LIST OF VARIABLES	18
LISTING OF BASIC TEST PROGRAM	19
TASK 1 REPORT	APPENDIX A
TASK 2 REPORT	APPENDIX E

FINAL REPORT

INTRODUCTION

This constitutes the final report of DBS Corporation on contract DE-ACC1-79-EI-10597 entitled "Development of Software for Computer-assisted Model Simplification". DBS is one of several contractors who contributed to this project.

The overall project was divided into four tasks:

Task 1 - Overall project assessment

Task 2 - Design principles and testing procedures

Task 3 - Specific experimental designs

Task 4 - Initial test results

DBS reports on task 1 and task 2 were submitted separately, but are included here as appendixes A and B for completeness. The DBS report on tasks 3 and 4 are combined and presented here.

The DBS task 1 report, overall project assessment, concluded that the development of software for computer aided model simplification was almost surely feasible, and promised to be of considerable value for many purposes. Some questions were raised, however, concerning the value of qualitative methods, which can only be answered by considerable experience. It was observed that without a natural metric on the rows and columns of an LP matrix, qualitative measures might either be too weak, or even misleading. It was also noted that matrix scaling could be a key to model simplification and diagnostics, and that there was, in fact, a

natural metric to impose -- namely that given by the shadow prices.

As a consequence, DBS was directed to focus its attention in task 2 on the implications of shadow price information to the project. In its task 2 report, DBS developed the concepts of Economic Correlation, Substitutability, and Complementarity in a natural definition based on shadow prices, as well as the natural rescaling of matrices with shadow prices and the benefits which could accrue in diagnosis and error tracing. In addition, DBS proposed an initial algorithm for finding approximate shadow prices in an efficient manner for cases in which they were not conveniently available as a by-product of problem solution.

DBS was consequently directed to focus the remainder of its effort under this contract to testing the feasibility of an algorithm for efficiently determining approximate shadow prices along the lines proposed in task 2. Thus the algorithm proposed in task 2 became a specific experimental design (task 3) and it was programmed and subjected to initial testing (task 4). During the course of this testing the initial proposed algorithm proved unsatisfactory and an entirely new algorithm, previously developed by DBS, was introduced. This new algorithm proved quite satisfactory in the initial (small-scale) testing. Because task 3 is partially contained in the task 2 report, and partially in task 4, we have chosen to report task 3 and 4 on a combined basis.

OVERALL ASSESSMENT (RETROSPECTIVE)

In retrospect, the views expressed by DBS in the initial assessment are essentially unchanged. The doubts about the usefulness of purely qualitative analysis remain until sufficient experience is obtained. The general feasibility and desirability of software systems which can aid in simplifying models, in diagnosing errors, and which can predict directions of changes for changed inputs is beyond question.

The strong conviction remains that shadow price information is potentially extremely useful in any such system, and that provision to incorporate it and exploit it should be pursued with vigor. To that end, the algorithm presented here should be subjected to large scale testing on realistic problems encountered at DOE, and further developed and modified as necessary.

TASKS 3 & 4 REPORT

This section represents the report of DBS Corporation on tasks 3 and 4 of Contract DE-ACC1-79-EI-10597 entitled "Development of Software for Computer-Assisted Model Simplification". Task 3 consisted of proposing specific experiental designs, while task 4 consisted of obtaining initial test results. Since a specific experimental design was presented in the task 2 report (the algorithm proposed there), which was subsequently adopted for initial testing, and since in the course of this testing an entirely new algorithm was introduced, the task 3 report is subsumed here.

The final task was devoted to exploring possible algorithms for efficiently finding approximate Lagrange multipliers ("shadow prices") for a linear program using techniques of the Generalized Lagrange Multiplier method (GLM). The initial algorithm proposed in task 2, together with numerous variations was subjected to initial tests. This algorithm was never completely satisfactory in the initial tests, and was replaced by a completely different algorithm during the course of task 4.

All of the testing was performed on small scale sample matrices (up to about 10x10) which were randomly generated, and non-sparse. Not all desirable tests and experiments could be carried out because of funding limitations, but it is felt that the algorithm presented here should

perform satisfactorily on large scale, sparse matrices.

The nature of the Lagrangian approach is to postulate multiplier values, pass through the matrix column by column maximizing payoff minus shadow cost, regardless of resource consumption -- and then to adjust the multipliers on successive passes to drive the resources actually consumed toward their constraint value.

Because of this approach, there is never any matrix inversion, so that only the original non-zeros need be stored, and there is no need for double precision calculations to avoid numerical difficulties. Experience with a number of test matrices ranging from 4x4 to 10x10 kernels indicates that satisfactory convergence is obtained in from 20 to 200 iterations. It is expected that, in actual large scale matrices of the type encountered in DOE, the performance will be a good deal better. Note that one pass accesses each non-zero matrix element only once, and performs essentially several column scalar products. It is therefore anticipated that the Lagrangian method could be a good deal faster, and use less working memory, for LPs of the typical variety at DOE.

A BASIC program listing, together with an explanatory description, is included in this report giving the implementation of the test program used. This program may be used as a guide to recoding the algorithm, or directly executed for further tests and experiments. The program was written and run on the TRS-80 microcomputer models I and II.

THE ALGORITHM

The algorithm initially proposed in the Task 2 report under this contract, in which a nonlinear (quadratic penalty) term was added to control activity levels was coded and tested. This method was never completely satisfactory, largely because the target activity levels had to be periodically reset, and this process tended to interfere with the Lagrange multiplier adjustment method originally proposed.

During the course of this phase of work an alternative algorithm, previously developed by DBS and used in other applications, was introduced and proved to perform much better in the tests. This algorithm uses a nonlinear penalty function on resource consumption (rather than on activities), centered at the value of the constraint. In this approach, the total consumption of each resource is kept track of at all times, and updated (differentially) each time an activity level is changed.

Fortunately the equations for the optimal activity level for an activity, given present values for all of the others (and hence resource consumption of all the rest), are analytically solvable and relatively simple. The penalty function on resource consumption is constructed in such a manner that it depends upon a postulated set of Lagrange multipliers, and has the effect of quadratically penalizing (in addition to the ordinary GLM linear Lagrangian penalty) deviations on either side of the constraint value (weighted by the shadow prices). The process

therefore consists of choosing some starting set of trial multipliers, and proceeding to optimize the activity levels for each activity, one at a time. After each pass through the activities, the trial multipliers are reset to the "effective multiplier values", which can be easily calculated as the derivatives of the Lagrangian with respect to resource levels at the current resource consumption level.

MATHEMATICAL FORMULATION

Let subscripts i denote activities, and subscripts j denote constraints. Furthermore, for appropriate subscripts, let x designate activities, y shadow prices, a matrix elements, c constraint values, and h "payoff" values. In this notation the LP problem is stated as

$$\text{MAXIMIZE } H = \sum_i h_i x_i \quad (1)$$

$$\text{subject to } \sum_i x_i a_{ij} \leq c_j \text{ all } j$$

with, possibly, further upper and lower bounding constraints on the x 's.

Following the GLM approach, we form the Lagrangian:

$$L = \sum_i h_i x_i - \sum_j y_j z_j (1 + b_j z_j) \quad (2)$$

where

$$z_j = \sum_i x_i a_{ij} - c_j \quad (3)$$

The z 's are seen to measure the deviation of actual resource consumption from the constraint values, and equation (2) is seen to be a standard GLM Lagrangian with an added factor, linear in z , on the penalty function. The b coefficients are introduced to control the "strength" of the penalty functions.

This form is chosen to be identical to the standard linear Lagrangian when the constraints are exactly met, and to additionally quadratically penalize deviations on either side. (All the standard theorems of GLM carry through for this form of Lagrangian.)

The Lagrangian must be slightly modified in case the added factor should be negative, a condition we shall call "below the slack limit". This is accomplished by freezing the value of any penalty term at the value it would have had as the factor crossed zero, namely:

$$\begin{aligned} & y_j/4b_j \quad \text{if} \quad 1+b_j z_j < 0 \\ & y_j z_j (1+b_j z_j) \quad \text{if} \quad 1+b_j z_j \geq 0 \end{aligned} \quad (4)$$

We can now calculate the "effective" shadow price for this form of Lagrangian at any trial solution point by taking the partial derivative with respect to the z values:

$$-\frac{\partial L}{\partial z_j} = \begin{cases} y_j(1+2b_j z_j) & \text{if } 1+2b_j z_j \geq 0 \\ 0 & \text{if } 1+2b_j z_j < 0 \end{cases} = y_j^{\text{eff}} \quad (5)$$

which will be used in the algorithm to adjust trial multiplier values from time to time.

As mentioned previously, a major property of this formulation is the fact that the optimum level of an activity can be calculated directly, which we now proceed to do. In what follows we shall use the symbol $\sum$ on a summation to denote sums over only terms above the "slack limit". (The Lagrangian derivatives vanish for terms below the "slack limit").

Taking the derivatives of the Lagrangian with respect to the

activities, and setting them to zero (the functions are properly concave) we get:

$$\begin{aligned}\frac{\partial L}{\partial x_i} &= h_i + \sum_j \frac{\partial L}{\partial z_j} \frac{\partial z_j}{\partial x_i} \\ &= h_i + \sum_j y_j (1+2b_j z_j) a_{ij} = 0\end{aligned}\quad (6)$$

Now suppose that we have a given trial solution with known x values (and known z values). We wish to recalculate the optimum value for a given single activity (the i -th say), with all other activities remaining at their present value. Denote present (original) values with zero superscripts. Then the effect on each of the z 's for a new value of the i -th activity is:

$$z_j = z_j^0 - x_i^0 a_{ij} + x_i \quad (7)$$

substitution into equation (6) yields:

$$\begin{aligned}h_i &= \sum_j y_j a_{ij} (1+2b_j (z_j^0 - x_i^0 a_{ij} + x_i a_{ij})) \\ &= \sum_j y_j a_{ij} (1+2b_j (z_j^0 - x_i^0 a_{ij})) + 2 \sum_j y_j a_{ij} b_j x_i a_{ij}\end{aligned}\quad (8)$$

from which the optimal value for the i -th activity is easily obtained as:

$$x_i = \frac{h_i - \sum_j y_j a_{ij} (1+2b_j (z_j^0 - x_i^0 a_{ij}))}{2 \sum_j y_j a_{ij}^2 b_j} \quad (9)$$

The optimal activity levels given by equation (9) must, of course, be

truncated by any upper or lower bounds in the problem, where zero is a lower bound in all cases. Care must be taken that in the evaluation of (9), no resource crosses its "slack limit", since the summations specifically exclude all resources below this limit. This is easily and efficiently accomplished algorithmically by maintaining a flag on each resource to give its slack limit status, and checking for no change when re-optimizing each activity. Such crossings seem to be relatively rare, and are easily handled when they occur by simply repeating the calculation whenever crossings are detected until convergence.

Finally, a simple upper bound can be maintained on the solution by the surrogate constraint method. After each iteration one uses the current shadow values to generate a single surrogate constraint (in effect sum all rows weighted by the shadow values) to get a reduced single row problem which is easily solved to give an upper bound. The minimum of all of these U.B.s is collected over all iterations to give the best current U.E.

NOTES ON BASIC TEST PROGRAM

These notes are supplied to aid in understanding the operation of the BASIC test program, of the underlying algorithms, and as guidance for reprogramming the algorithms in another language.

Lines 10-115 initialize the system, dimension working arrays, and request the user to input the desired number of rows and columns for a test problem. In particular, line 20 reserves string space, and line 30 declares all variables beginning with I through N to be type integer and variables beginning with T through V to be type string. All other variables will default to type real (single precision).

Lines 120-170 perform a random matrix generation with elements drawn uniformly from -1 to +1, for a non-zero density specified by the user. This feature was not used in the testing, and all tests were performed at density 1.

Lines 180-250 set up the SOLUTION to the test problem, which will be known in advance, and can hence be compared with the results of the algorithm. The user is prompted for the desired "kernel" size, which will be the number of non-zero activities (and non-zero shadow values as well) in the final solution. For convenience, these will be the first n rows and the first n columns of the generated test problem. The solution activities are randomly generated between 1 and 2.

Lines 400-590 calculate the constraint levels and objective function coefficients for which the previously specified activities and shadow prices are the solution. This completes the test problem generation, for which the solution is known (and saved for comparison purposes). The mathematics of this process were discussed in the Task 2 report under this contract. The constraints for the slack rows, and the objective function values for the inactive activities are, in this version, set well away from the points where they would become active (by 10 units). Time and funds did not permit testing of the behavior of the algorithm in the neighborhood of slackness, and some modifications may well be required.

Lines 1000-1070 initialize the solution algorithm. The current resource consumption vectors are cleared, the relative penalty strengths (b's) are all set to 1, the initial trial shadow prices are all set to 1, and the best global (across iterations) upper bound (EU), "distance from solution" (DV), and current distance (DS) initialized to "infinity". In addition, all activities are set initially to 0. Finally, the initial number of iterations to perform, NX, is set.

Lines 5000-5070 constitute the main control loop for an iteration. This loop is performed until the cumulative number of iterations NI equals the desired number NX. Within this loop, subroutines are called to capture the best solution so far, copy the current resource consumption vector to the consumption vector for the prior iteration, adjust the shadow prices, optimize the Lagrangian for each activity, and display results for the

iteration. The results displayed are: B2-the upper bound produced by this iteration; EU-the least upper bound found so far; DV-the best "solution distance" found so far; HT-the objective function value of the current solution; HS-the objective function value of the true solution; and DS-the "solution distance" of the current solution. "Solution distance" is computed as the root-mean-square distance of the constraint consumption vector from the constraint vector, weighted by the shadow prices. Lines 5050-5056 recompute directly (and display) resource consumption every 20 iterations, in order to guard against any gradual drift due to the differential updating of these quantities in the algorithm.

Lines 5080-5130 allow the user several options after the specified number of iterations have been performed. First he is given the option of continuing with further iterations. If not he is given a menu of display options which allow him to display various quantities, and compare them with the true solution. After the various displays, the user can continue with still further iterations if desired.

Lines 25200-25240 comprise a subroutine which computes the total shadow cost of resource usage by the current solution (ZA), the total shadow cost of the problem constraints (ZC), and the "distance" of the current resource usage from the constraints (DS), which is a simple Euclidean distance in the shadow price metric.

Lines 27000-27170 contain the subroutine which maximizes the

Lagrangian for a given column, and can be regarded as the heart of the algorithm. Lines 27010-27080 compute the new optimal level of activity i , for fixed levels of all other activities, taking into account the current total resource usage and the effect on it of changes in activity i . Thus the i -th activity is adjusted to maximize the overall Lagrangian, by direct implementation of equation (9) in the mathematical formulation section of this report. Line 27085 truncates this activity value at 0, to enforce non-negativity of activities. (If upper and lower bounds were introduced, this line would be replaced to enforce truncation conforming to the bounds.) Lines 27095-27140 update the overall resource consumption (differentially) and the shadow cost of column i to conform to the new level of activity. At the same time, the "slack limit" status of each resource (E) is computed in lines 27112-27114, and compared to the previous status for that resource ($ES(j)$). If a change of status has occurred, the new status is recorded and the slack limit crossing flag (EF) is set to indicate that a crossing has occurred (line 27118). When all rows have been so processed, the new activity level is recorded in line 27140. If the slack limit crossing flag is now found to be ON, indicating that some resource changed its slack limit status (and invalidating the optimization), a display of the slack limit status of each row is produced and the maximization step repeated. This process repeats until no status changes occur in the maximization. Lines 27145-27165 then capture certain global information before returning from the optimization subroutine. This information consists of the total value of the objective function (HT),

and two quantities (PM and PN) labelled "best consumer" and "best producer" to be used in the upper bound calculation.

Lines 27200-27280 are the subroutine which drives the individual column optimizer for a full pass of all columns, collects overall information, and produces an upper bound calculation (B2) for the iteration. Line 27210 computes the total shadow cost of all the constraints (ZC), to be used later in the surrogate-constraint bounding method. Line 27220 initializes the quantities PM and PN ("best consumer" and "best producer") to negative infinity. Line 27230 initializes the total objective function value (HT) to zero and proceeds to call the column optimization subroutine for all columns. After completion of the optimization pass, the upper bounding calculation is performed. Lines 27240 and 27250 are checks for unboundedness of the 1xNC surrogate constraint problem formed by summing all the rows of the original problem weighted by the shadow values. If either the "best producer" value (PN) or the "best producer" plus the "best consumer" value (PN+PM) exceeds 0 then the surrogate problem is unbounded. In the implementation 0 is replaced by a threshold (EP) to give better performance in producing bounds. The bound calculation then branches (line 27260) according to whether the constraint value of the surrogate problem (ZC) is positive or negative. If positive, then the bound is given from the "best consumer" in line 27270 by consuming the entire constraint, if the value is positive, or 0 if it is negative. In the case that the surrogate constraint is negative ("forced"

production -- even at negative payoff) then the bound is produced from the "best producer" in line 27280 by producing only what is necessary to meet the constraint.

Lines 27310-27370 give the shadow price adjustment algorithm which is executed between iterations. In this subroutine, each shadow price is adjusted to equal its "effective" value in the overall Lagrangian for the current resource consumption levels. This subroutine implements equation (5) of the mathematical formulation section, where the basic variable $BJ(J)$ is understood to incorporate already the factor of 2 multiplying $b(j)z(j)$ in eq. (5). The change in shadow prices for one iteration is limited by a factor of two in either direction for stability purposes in lines 27330 and 27340, which also eliminates the necessity to truncate at 0 below the slack limit, as specified in equation (5).

Lines 30100-30170 are a utility subroutine to replace the resource consumption vector for the prior iteration ($CA(J)$) with that of the current iteration ($CC(J)$), and in case the current iteration is the best so far, to record the necessary data (activity levels $XE(I)$, shadow prices $SB(J)$, resource consumption levels $CB(J)$, and best solution distance DV).

Finally lines 40000-40040 give the subroutines for displaying results.

LIST OF VARIABLES

The variables used by the BASIC test program, and their description are as follows:

A(I,J)	LP Matrix for column i, row j
X(I)	amount of i-th activity
H(I)	payoff per unit activity i (objective function)
S(J)	shadow price for constraint (row) j
C(J)	constraint value for row j
NC	number of columns
MR	number of rows
XS(I)	values of original activities (for display)
SS(J)	values of original shadow prices (for display)
SC(I)	shadow cost per unit activity i
CC(J)	constraint consumption row j, present iteration
CA(J)	constraint consumption row j, last iteration
NI	iteration number
BJ(J)	penalty "strength" row j (times 2)
ES(J)	slack limit CN flag row j
BU	current least upper bound to solution value
DV	best solution "distance" so far
DS	solution "distance", current solution
CB(J)	resource consumption, row j, best solution so far
XE(I)	activity for i-th col., best solution so far
SB(J)	shadow price row j, best solution so far
HT	objective function value, current solution
HS	objective function value, original (true) solution
ZA	total shadow cost, current resource consumption
ZC	total shadow cost, constraint values
XN	numerator in optimization step
XD	denominator in optimization step
EF	slack limit crossing flag, optimization step
E	slack limit status in optimization step
EP	tolerance in computing upper bounds
NX	number of iterations to perform

BASIC PROGRAM LISTING

```
-----
10 'LPTESTZ/VO8 (02/07/80)
20 CLEAR 1000
30 DEFINT I-N:DEFSTR T-V
40 INPUT"NUMBER OF COLUMNS";NC
50 INPUT"NUMBER OF ROWS";MR
60 DIM A(NC,MR),X(NC),H(NC),S(MR),C(MR),CA(MR)
70 DIM SC(NC),CC(MR),BJ(MR),ES(MR)
71 'BJ IS PENALTY STRENGTH, ES IS SLACK LIMIT ON FLAG
80 'A IS MATRIX,X ACTIVITIES, H OBJECTIVE,S SHADOW PRICES,
90 'C CONSTRAINTS, CA LAST ACTUAL CONSTRAINTS, CC CURRENT CONST.
100 'SC TOTAL SHADOW COST PER UNIT ACTIVITY
113 DIM XS(NC),SS(MR)
115 'XS SS ARE ORIG SOLN
120 INPUT"NON-ZERO MATRIX DENSITY (0.0 TO 1.0)";DN
130 PRINT "FILLING MATRIX"
140 FOR I=1 TO NC
150 FOR J=1 TO MR
160 IF RND(0)<DN THEN A(I,J)=RND(0)*2-1 ELSE A(I,J)=0
170 NEXT J: NEXT I
180 INPUT"KERNEL SIZE";KS
190 'SET SHADOW VALUES, ACTIVITIES ON KERNEL, 0 OUTSIDE
200 FOR I=1 TO NC
210 IF I>KS THEN X(I)=0 ELSE X(I)=RND(0)+1.0
215 XS(I)=X(I)'SAVE ORG.
220 NEXT I
230 FOR J=1 TO MR
240 IF J>KS THEN S(J)=0 ELSE S(J)=RND(0)+1.0
245 SS(J)=S(J)'SAVE ORG
250 NEXT J
400 'COMPUTE ACTUAL CONSTRAINT CCNSUMPTION AND SHADOW COST
410 FOR J=1 TO MR
420 CA(J)=0
430 FOR I=1 TO NC
440 CA(J)=CA(J)+X(I)*A(I,J)
450 NEXT I:NEXT J
460 'COMPUTE SHADOW COST PER UNIT ACTIVITY
470 FOR I=1 TO NC
480 SC(I)=0
490 FOR J=1 TO MR
500 SC(I)=SC(I)+S(J)*A(I,J)
510 NEXT J: NEXT I
520 'GENERATE CONSTRAINT LEVELS
530 FOR J=1 TO MR
540 IF S(J)>0 THEN C(J)=CA(J) ELSE C(J)=CA(J)+10.
550 NEXT J
```

```
560 'GENERATE OBJECTIVE FUNCTION
570 HS=0:FOR I=1 TONC
580 IF X(I) >0 THEN H(I)=SC(I) ELSE H(I)=SC(I)-10.
585 HS=HS+X(I)*H(I)
590 NEXT I
1000 'GENERATE STARTING CONDITIONS
1010 FOR J=1 TO MR
1013 CA(J)=0:CC(J)=0
1015 BJ(J)=1
1020 S(J)=1.
1040 NEXT J
1047 BU=1.E30
1048 DV=1.E30:DS=DV:EP=.001
1050 'SET INITIAL ACTIVITIES TO 0
1060 FOR I=1 TO NC:X(I)=0:NEXT I
1070 NX=20'SET INITIAL NUMBER OF ITERATIONS
5000 NI=NI+1
5010 GOSUB 30100'REPLACE CA WITH CC, CAPTURE BEST SOLUTION
5015 GOSUB 27310'ADJUST SHADOW PRICES
5020 PRINT "ITERATION NUMBER";NI
5030 GOSUB 27200'PERFORM ONE LAGRANGIAN OPTIMIZATION PASS THRU ACTIVITIES
5031 IFB2<BUTHENEU=B2'CAPTURE LEAST UPPER BOUND
5032 PRINT "B2 BU DV",B2,BU,DV
5035 GOSUB 25200' ACCUM DATA
5037 PRINT"HT HS DS",HT,HS,DS
5050 J=NI/20:IF NI<>J*20 THEN 5060'SKIP EXCEPT EVERY 20-TH
5052 PRINT"RECCMPUTING RESOURCES CONSUMED"
5053 FOR J=1 TO MR:X=0'AVOID DIFFERENTIAL DRIFT
5054 FOR I=1 TO NC:X=X+X(I)*A(I,J):NEXT I
5056 PRINT"X CC D",X,CC(J),X-CC(J):CC(J)=X:NEXT J
5060 'DONE ITERATION
5070 IF NI<NX THEN 5000
5080 N=0:INPUT"NUMBER OF EXTRA ITERATIONS";N
5090 IF N>0 THEN NX=NI+N:GOTO5000
5100 PRINT"1-PRINT CCNSTR","2-PRINT ACT","4-CONTINUE RUN"
5102 PRINT"5-DISPLAY ORG X","6-DISPL.ORG S"
5110 INPUT"CODE";I
5120 ON I GOSUB 40000,40010,40020,5080,40030,40040
5125 PRINT"TOUCH ANY KEY TO CCNTINUE"
5130 T=INKEY$:IFT=""THEN5130ELSE5100
25190 'COMPUTE ZA TOTAL CCST ACTUAL, ZC COST AT CONSTRAINTS
25200 ZA=0:ZC=0:DS=0
25210 FOR J=1 TO MR
25220 ZA=ZA+S(J)*CC(J)
25230 ZC=ZC+S(J)*C(J)
5231 D=S(J)*(C(J)-CC(J)):DS=DS+D*D
5232 NEXTJ
25233 DS=SQR(DS)
```

```
25240 RETURN
27000 'MAXIMIZE Z-FORM LAGRANGIAN FOR COL I
27010 XN=0:XD=0:EF=0
27020 FOR J=1TOMR:IFES(J)>0 THEN 27060'SKIP IF SLACK LIMIT
27030 X=S(J)*A(I,J)
27040 XN=XN+X*(1.+BJ(J)*(CC(J)-C(J)-X(I)*A(I,J)))
27050 XD=XD+X*A(I,J)*BJ(J)
27060 NEXT J
27070 'COMPUTE NEW OPTIMAL X
27080 X=(H(I)-XN)/XD
27085 IF X<0 THEN X=0
27090 'UPDATE RESOURCES DIFF. CHECK SLACK LIMIT
27095 SC(I)=0
27100 FOR J=1 TO MR
27110 CC(J)=CC(J)+A(I,J)*(X-X(I))
27111 SC(I)=SC(I)+S(J)*A(I,J)
27112 E=0
27114 IF BJ(J)*(CC(J)-C(J))<-1 THEN E=1
27116 IF E=ES(J) THEN 27120
27118 ES(J)=E:EF=1'SET SLACK CROSSING FLAG
27120 NEXT J
27130 'RESET X(I)
27140 X(I)=X:IF EF=1 THEN PRINT"SLACK LIMIT CROSSING":
FORJ=1TOMR:PRINTES(J);:NEXTJ:PRINT:GOTO27010
'ITERATE IF CROSSED SLACK LIMIT
27145 HT=HT+X(I)*H(I)
27150 IF SC(I)<0 THEN 27165
27155 P=H(I)/SC(I):IF P>PM THEN PM=P'BEST CONSUMER
27160 GOTO 27170
27165 P=-H(I)/SC(I):IF P>PN THEN PN=P'BEST PRODUCER
27170 RETURN
27200 'PERFORM FULL PASS
27210 ZC=0:FORJ=1TOMR:ZC=ZC+S(J)*C(J):NEXTJ
27220 PM=-1.E30:PN=-1.E30
27230 HT=0:FCRI=1TCNC:GOSUB27000:NEXTI
27240 IF PN>EP THEN B2=1.E30:RETURN'PRODUCER GAIN UNBOUNDED
27250 IF PN+PM>EP THEN B2=1.E30:RETURN'PRODUCER+CONSUMER UNBOUNDED
27260 IF ZC<0 THEN 27280
27270 IF PM>0 THEN B2=PM*ZC:RETURN ELSE B2=0:RETURN
27280 B2=-PN*ZC:RETURN'FORCED PRODUCTION
27300 'ADJUST LAMBDA S
27310 FOR J=1 TO MR
27320 F=1+BJ(J)*(CC(J)-C(J))
27330 IF F<.5 THEN F=.5
27340 IF F>2 THEN F=2
27350 S(J)=S(J)*F
27360 NEXT J
27370 RETURN
```

```
30100 'REPLACE CA WITH CC, CAPTURE BEST SOLUTION SC FAR
30102 IF DS>DV THEN 30140
30106 DV=DS
30110 FOR I=1TONC:XB(I)=X(I):NEXTI
30114 FCRJ=1TCMR:SB(J)=S(J):CB(J)=CC(J):NEXTJ
30140 FOR J=1 TO MR
30150 CA(J)=CC(J)
30160 NEXT J
30170 RETURN
40000 PRINT "C","CB","CC","S":FOR J=1TCMR:
PRINT C(J),CB(J),CC(J),S(J):NEXTJ:RETURN
40010 PRINT"X","XB","SC","H":FOR I=1TONC:
PRINTX(I),XB(I),SC(I),H(I):NEXTI:RETURN
40020 GOTO 5100
40030 PRINT"XS","XB","X ":FOR I=1TONC:
PRINTXS(I),XB(I),X(I):NEXT I:RETURN
40040 PRINT"SS","SE","CA","C":FCRJ=1TCMR:
PRINTSS(J),SE(J),CA(J),C(J):NEXTJ:RETURN
```

APPENDIX A

TASK 1 REPORT

CORPORATION 1901 NORTH MOORE STREET, ARLINGTON, VIRGINIA 22209, (703) 527-9060

APPENDIX A

October 19, 1979

Dr. Harvey Greenberg
Department of Energy
Room 4522
1200 Pennsylvania Avenue, N. W.
Washington, D. C. 20545

Subject: Contract No. DE-AC01-79EI-10597
Overall Project Assessment Topical Report

Dear Harvey:

This letter summarizes my initial reactions and thoughts covering the development of software for computer aided model simplification.

My initial reaction is that something along the lines of the proposed system will almost surely be feasible, and of considerable value for many purposes. I am somewhat less certain at this time about the particulars.

The usefulness of qualitative analysis (restricting information to signs of entries) in particular remains unclear to me. While undoubtedly in some cases it could be valuable, it is easy to construct artificial examples where it would be completely misleading. What would be encountered in actual practice cannot be foretold.

I also have some difficulty with the concept of causality - the tracing of the causes of infeasibility (or unboundedness) - since without any other knowledge every possible set of constraints whose removal would remove the infeasibility could be said mathematically to be the cause. In reality, of course, the analyst has a-priori knowledge of likely ranges of reasonable values, likely errors, etc. which must be used to impart meaning to "causality." Certainly interactive algorithms that can display likely candidates for such an analysts inspection would be valuable, but it is not yet clear to me on what basis such likelihood would be assessed.

One possibly very fruitful area of exploration occurred to me repeatedly in my review of the material - that if there were a "natural" rescaling of a matrix to where the true "importance" of the various coefficients were reflected in their

APPENDIX A (continued)

Harvey Greenberg
Contract No. DE-AC01-79EI-10597

-2-

October 19, 1979

numerical magnitude, many analytic tasks should be vastly simplified, various approximations and quantizations could be used, and the performance of any heuristic algorithms enhanced. Furthermore, the concepts of economic correlation, complementarity, substitutibility, etc. should be much more meaningful.

After several abortive attempts, I finally realized that the key to "correct" scaling is the shadow prices which are, of course, available after solution of an L.P. and, I believe, can probably be approximately obtained even before solution with much less computation than required to get the solution.

The shadow prices represent the marginal change in payoff (or cost) for changes in the constraint values, and hence convert units of constraint to units of ultimate payoff (objective function). Thus it seems most natural to rescale all rows by multiplication by the associated shadow prices. (Slack rows will, of course, disappear.)

Such row rescaling puts all the elements of a column on a comparable basis, but still allows free choice of column scaling. Several possibilities seem promising, including setting the max element (abs value) to one, setting the sum of negative elements to one, setting the self dot product to one, or perhaps most naturally addressing the dual and scaling just as was done for rows.

As a final note, it appears to me that the hashing algorithms currently used for value pool construction would be greatly speeded by using a bucket size of 1, since the length of the bucket search is then quite small. This observation applies equally to linear or list search organization. Buckets of larger size are appropriate for disk file access, where the cost of a search is small compared to a disk access, but not to in core data.

Sincerely,

A handwritten signature in dark ink, appearing to read "Hugh Everett, III". The signature is fluid and cursive, with a prominent "H" and "E".

Hugh Everett, III
Chairman

APPENDIX B

TASK 2 REPORT

APPENDIX B

DEVELOPMENT OF SOFTWARE FOR
COMPUTER-ASSISTED MODEL SIMPLIFICATION
TASK 2
DESIGN PRINCIPLES AND TESTING PROCEDURES
TOPICAL REPORT

PREPARED FOR
THE DEPARTMENT OF ENERGY
UNDER CONTRACT NUMBER DE-AC01-79EI-10597
DECEMBER 4, 1979

PREPARED BY
DR. HUGH EVERETT, III
DBS CORPORATION
1901 NORTH MOORE STREET
ARLINGTON, VIRGINIA 22209

NOTICE

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for any third party's use of the results of such use of any information, apparatus, product, or process disclosed in this report, or represents that its use by such third party would not infringe privately owned rights.

APPENDIX B (continued)

DISTRIBUTION LIST

Harvey J. Greenberg Project Manager Office of Analysis Oversight & Access, AA, EIA Federal Building, 12th & Pennsylvania Avenue, N. W. M. S. 4530, Room 4522 Washington, D. C. 20461	2 copies
George M. Lady Director, Office of Analysis Oversight & Access Federal Building, 12th & Pennsylvania Avenue, N. W. M.S. 4530, Room 4522 Washington, D. C. 20461	1 copy
Mr. William Kurator c/o Harvey J. Greenberg Office of Analysis Oversight & Access Federal Building, 12th & Pennsylvania Avenue, N. W. M.S. 4530, Room 4522 Washington, D. C. 20461	1 copy
Dr. Robert E. Bixby Industrial Engineering Department Northwestern University Evanston, Illinois 60201	1 copy
Dr. James E. Kalan Department of Management University of Florida Gainsville, Florida 32611	1 copy
Dr. John S. Maybee 265 Hapi Place Boulder, Colordao 80303	1 copy

APPENDIX B (continued)

ABSTRACT

The central focus of this Task II report on Design Principles and Testing Procedures is the potential usefulness of Shadow Price information for the model simplification project. Methods are suggested for obtaining approximate shadow prices efficiently, and for generating test problems with known characteristics and solutions. The potential usefulness of shadow price information for economic correlation studies, and for matrix rescaling to natural and comparable units, is discussed, as well as considerations for model execution, reporting of model contents and results, error tracing, aggregation specifications, and model audit procedures.

APPENDIX B (continued)

TABLE OF CONTENTS

	<u>PAGE</u>
ABSTRACT.	i
I. INTRODUCTION.	1
II. ALGORITHM FOR SHADOW PRICE GENERATION	2
III. TEST PROBLEM GENERATOR.	6
IV. SLACKNESS	8
V. ECONOMIC CORRELATION.	9
VI. MATRIX SCALING.	13
VII. CONSIDERATIONS.	15

APPENDIX B (continued)

I. INTRODUCTION

The Task 1 Report mentioned that a potentially very fruitful area of exploration appeared to exist in the area of matrix rescaling, so that the true importance of the various coefficients would be manifest in their values. It was suggested there that the key to such rescaling was the exploitation of shadow value information - either after a solution was in hand, or before by use of special algorithms for their rapid approximate generation.

This report expands upon that idea by presenting a suggested initial algorithm for shadow price generation (Section II), a test problem generation procedure for testing this algorithm or any other proposed algorithms (Section III), and a discussion of slackness and possible methods of dealing with the discontinuity in shadow price as slackness occurs (Section III). Section IV is devoted to a discussion of Economic Correlation, substitutability and complementarity and their natural definition using shadow prices. Section V discusses the natural rescaling of matrices using shadow prices, and its possible usefulness in diagnosis and error tracing. Finally, Section VI discusses considerations of simplification of model execution, reporting of model contents and results, error tracing, aggregation specifications, and model audit procedures.

Perhaps the most significant immediate design consideration for the project from this work is to be sure that the adopted data structure allows for the additional storage and retrieval of shadow price data, objective function data, and row and column rescaling factors.

II. ALGORITHM FOR SHADOW PRICE GENERATION

While after solution of a linear programming problem the shadow prices are usually readily available, it may be quite useful to have a method at hand for obtaining them quickly before actual solution, as a possible aid to more rapid solution, or as an early detector of (and diagnostic aid for tracing) infeasibility or unboundedness.

One suggested algorithm is based upon regarding a linear program within the Generalized Lagrange Multiplier (GLM)¹ framework. Within this framework the problem is formulated as:

$$\text{Problem: Max } \sum_i h_i x_i$$

$$\text{Subject to: } x_i \geq 0 \text{ all } i$$

$$\sum_i a_{ij} x_i \leq c_j \quad \text{all } j$$

where coefficients h_i are the objective function, x_i the activities, c_j the constraints, and a_{ij} the matrix.

The GLM form of this problem is to maximize, for given non-negative multipliers λ_j :

$$L = \sum_i h_i x_i - \sum_j \lambda_j \sum_i a_{ij} x_i$$

which decomposes into independent terms:

$$L = \sum_i L_i = \sum_i (h_i x_i - \sum_j \lambda_j a_{ij} x_i)$$

so the GLM technique, for given λ 's causes us to independently maximize each term:

¹ "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," Operations Research II, (3), (May-June 1963)

$$\begin{aligned}
 L_i &= h_i x_i - x_i \sum_j \lambda_j a_{ij} \\
 &= x_i (h_i - \sum_j \lambda_j a_{ij})
 \end{aligned}$$

We notice immediately a technical difficulty, namely that x_i will be 0 or ∞ according to the sign of $h_i - \sum_j \lambda_j a_{ij}$.

In order to stabilize the problem and allow a Lagrangian solution, we invert the classical physicists technique of linearization and introduce a slight non-linearity to make the problem strictly convex. A simple way to accomplish this is to add a slight quadratic penalty to the payoff function:

$$H = h_i x_i - c_i (x_i - x_i^0)^2$$

so that now

$$L_i = h_i x_i - c_i (x_i - x_i^0)^2 - x_i \sum_j \lambda_j a_{ij}$$

which can be analytically maximized by:

$$\begin{aligned}
 x_i^* &= x_i^0 + \frac{h_i - \sum_j \lambda_j a_{ij}}{2 c_i} \quad (\text{if } x_i^* > 0) \\
 &= 0 \quad (x_i^* \leq 0)
 \end{aligned}$$

The central task in the GLM approach is now to adjust the multipliers (and in our special case probably the c_i and x_i^0 as well) till all the constraints are met (or corresponding λ 's 0).

The proposed initial algorithm for adjusting the Lagrange multipliers is a simple, adaptive algorithm which has been widely used in the past and has been quite successful and robust.

APPENDIX B (continued)

The essence of this algorithm is to grow (or shrink) a Lagrange Multiplier after each iteration according to whether its corresponding constraint is exceeded (or slack). In addition, the factor itself is allowed to shrink (or grow) after each iteration according to whether the constraint was crossed (or not crossed) from the previous iteration. A constraint is "crossed" from one iteration to the next if it switches from slack to violated or vice-versa.

Mathematically, one starts with an arbitrary set of non-negative multipliers λ_j , and a set of adjustment factors, F_j , and proceed to iterate as follows (for the m -th iteration).

1. Maximize the Lagrangian for each column:

$$x_i^m = x_i^0 + (h_i - \sum_j \lambda_j^{m-1} a_{ij}) / 2c_i$$

2. Accumulate the "resource" consumption for each constraint:

$$c_j^m = \sum_i x_i a_{ij}$$

3. Modify the adjustment factors depending on whether a constraint was just crossed:

$$F_j^m = F_j^{m-1} * 1.3 \quad \text{if } (c_j^m - c_j) * (c_j^{m-1} - c_j) > 0$$

$$F_j^m = F_j^{m-1} * 0.25 \quad \text{if } (c_j^m - c_j) * (c_j^{m-1} - c_j) \leq 0$$

4. Adjust all the multipliers by:

$$\lambda_j^m = \lambda_j^{m-1} * (1 + F_j^m) \quad \text{if } c_j^m - c_j > 0$$

$$\lambda_j^m = \lambda_j^{m-1} / (1 + F_j^m) \quad \text{if } c_j^m - c_j \leq 0$$

5. Repeat 1 through 4 as required.

Of course, the coefficients 1.3 and 0.25 are arbitrary, but have served well in past cases.

In our case, we will also wish to adjust the quantities $\bar{x}_i^0$ around which the penalty function is formed, to equal the actual activity resulting from each iteration. The coefficients c_i should be chosen to be about equal to $\sum_j \lambda_j a_{ij}$ on each iteration.

III. TEST PROBLEM GENERATOR

In order to test the proposed method (as well as for other experiments) it is useful to have a method for generating test cases with controlled properties and known solutions.

The construction of such a test problem generator can be based upon the fact that given an arbitrary matrix, set of activities, and set of shadow prices, it is easy to construct the objective function and constraint values for which the given activities (and shadow prices) are the solution.

The method is quite simple, and easily understood within the GLM framework. Let x_i be the given activities, λ_j the given shadow prices, and a_{ij} the matrix. Then:

1. Compute the constraint values as

$$c_i = \sum_j x_i a_{ij} \quad \text{if } \lambda_j > 0 \text{ (not slack)}$$

$$> \sum_j x_i a_{ij} \quad \text{if } \lambda_j = 0 \text{ (slack)}$$

2. Compute the objective function values as

$$h_i = \sum_j \lambda_j a_{ij} \quad \text{if } x_i > 0$$

$$h_i < \sum_j \lambda_j a_{ij} \quad \text{if } x_i = 0$$

That the resulting problem has the desired solution is easily seen since the given activities maximize the Lagrangean and the constraints are exactly met for all $\lambda > 0$ and slack otherwise (see footnote ¹).

By using this technique, one can easily generate test problems with arbitrary sparseness, density of non-zero activities or slackness, degeneracy, or other structural properties. (note that if the number of non-zero activities does not equal the number of shadow values, the problem is degenerate and has multiple solutions, all with the same value of the objective function.)

By generating random test problems of given sparseness, or other properties, but with variable number of rows and columns, one can investigate in a meaningful way the scaling of solution time with problem size for various solution algorithms.

IV. SLACKNESS

While it is true that the shadow prices on a slack row are zero, the abrupt discontinuity between a finite value when not slack and 0 as slackness occurs might be too drastic for many rescaling and diagnostic purposes, especially in cases where some rows might be "just barely" slack.

In such cases it would probably be useful to impute to such a row a shadow value approaching the value it would have "if it were not slack," depending upon how close to the constraint the solution is, and approaching zero as the distance became significant.

To give meaning to the shadow value of a row if it were not slack, simply replace the original constraint on this row with its actual value in the solution (which creates a degeneracy in which this row can be in the solution basis) and PIVOT this row in.

In situations where the exact solution is not in hand, but one is attempting shadow price estimation by the algorithm suggested here, a modification where the target constraint levels are dynamically set near to actual levels might suffice to give reasonable values to "nearly binding" slack constraints.

V. ECONOMIC CORRELATION

One would like very much, as an aid to understanding and analysis as well as for estimating purposes, to have a measure of the "economic correlation" between two activities (or groups of activities) - the degree to which (and direction) one activity would be expected to change if another activity is changed.

To give precision to such a measure, let us suppose that we have the solution to a problem, and consider making a small change δ in the activity x_k . We now wish to inquire into the effect of this change on another activity x_j , and assume that all other activities are held constant.

In general, such a change will cause some constraints to be violated, others which were met to become slack, and still others which were slack to remain slack (if δ is small enough). We can, however, inquire as to what change ϵ in x_j will most closely restore the "equilibrium" - the original constraints.

For this inquiry to be meaningful, we need a metric on the space of constraint values, in order that distance in one constraint can be meaningfully compared and traded off with distance in another. Fortunately, there is such a measure, the shadow prices, which is in fact, the only metric which correctly measures the relative impact of marginal changes in constraint levels on the objective function and hence indirectly measures the relative "cost" of restoring original constraint levels after a given perturbation in activities.

This problem is most easily solved with vector notation. Given a vector $\vec{a}$, how much, α , of vector $\vec{b}$ should be added to $\vec{a}$ to get closest to the origin:

$$\text{Find } \min_{\alpha} (\vec{a} + \alpha \vec{b}) \cdot (\vec{a} + \alpha \vec{b})$$

This is easily minimized with respect to α and yields an optimum α of

$$\alpha = - \frac{\vec{a} \cdot \vec{b}}{\vec{b} \cdot \vec{b}}$$

which, under substitution, yields the minimum distance.

$$D^2 = (\vec{a} \cdot \vec{a}) - \frac{(\vec{a} \cdot \vec{b})^2}{(\vec{b} \cdot \vec{b})}$$

If we now normalize this distance (squared) to the length of vector $\vec{a}$ (make $\vec{a}$ a unit vector) we have

$$\frac{D^2}{(\vec{a} \cdot \vec{a})} = 1 - \frac{(\vec{a} \cdot \vec{b})^2}{(\vec{a} \cdot \vec{a})(\vec{b} \cdot \vec{b})}$$

which is an appropriate measure of how well the vector $\vec{b}$ can perform in compensating for a unit move away from the origin along vector $\vec{a}$. The value will range from 0 to 1 according to whether $\vec{b}$ is collinear with or orthogonal to $\vec{a}$.

An equivalent, and simpler measure is the complement,

$$\rho^2 = \frac{(\vec{a} \cdot \vec{b})^2}{(\vec{a} \cdot \vec{a})(\vec{b} \cdot \vec{b})}$$

which ranges from 0 for orthogonality to 1 for perfect collinearity and measures directly how well a move in the $\vec{a}$ direction from the origin can be compensated by a move in the $\vec{b}$ direction. To the extent that $\rho^2 < 1$ the remaining distance (squared) back to the origin would need to be recovered by further changes in other activities.

Returning to α , the amount of $\vec{b}$ required for optimal compensation for $\vec{a}$, it is convenient to normalize $\vec{a}$ and $\vec{b}$ to unit vectors in the $\vec{a}$ and $\vec{b}$ directions and to find α' , the amount of a unit vector in the $\vec{b}$ direction which optimally compensates for a unit vector in the $\vec{a}$ direction.

Replacing $\vec{a}$ with $\vec{a} / (\vec{a} \cdot \vec{a})^{1/2}$ and $\vec{b}$ with $\vec{b} / (\vec{b} \cdot \vec{b})^{1/2}$ in the equation for α yields the expression for unit vectors:

$$\alpha' = \frac{-\vec{a} \cdot \vec{b}}{(\vec{a} \cdot \vec{a})^{1/2} (\vec{b} \cdot \vec{b})^{1/2}}$$

which is seen to be the square root of ρ^2 and is hence conveniently identified (with a change of sign) as ρ :

$$\rho = \frac{\vec{a} \cdot \vec{b}}{(\vec{a} \cdot \vec{a})^{1/2} (\vec{b} \cdot \vec{b})^{1/2}}$$

This single quantity, ρ , therefore determines both the amount of a compensating $\vec{b}$ vector required, through

$$\alpha = -\rho \left(\frac{\vec{a} \cdot \vec{a}}{\vec{b} \cdot \vec{b}} \right)^{1/2}$$

and the remaining distance to the origin after optimal compensation:

$$D = (1 - \rho^2)^{1/2}$$

Translating these results to our situation of compensating for a perturbation δ in activity x_ℓ with a change α in activity x_i :

$$\vec{a} \cdot \vec{a} = \delta^2 \sum_j \lambda_j^2 a_{\ell j}^2$$

$$\vec{b} \cdot \vec{b} = \sum_j \lambda_j^2 a_{ij}^2$$

$$\vec{a} \cdot \vec{b} = \delta \sum_j \lambda_j^2 a_{\ell j} a_{ij}$$

so that

$$\rho = \frac{\sum_j \lambda_j^2 a_{\ell j} a_{ij}}{(\sum_j \lambda_j^2 a_{\ell j}^2)^{1/2} (\sum_j \lambda_j^2 a_{ij}^2)^{1/2}}$$

and

$$\alpha = -\rho \delta \left(\frac{\sum_j \lambda_j^2 a_{lj}^2}{\sum_j \lambda_j^2 a_{ij}^2} \right)^{1/2}$$

We see, therefore, that ρ , which is simply the scalar product of two columns scaled (weighted) by the shadow prices and renormalized to unit vectors is a natural measure for economic correlation, with a value of +1 corresponding to perfect substitutability and -1 corresponding to perfect complementarity. We have also seen that $1 - \rho^2$ measures degree to which substitutability or complementarity falls short and would need to be made up by adjustment in other activities.

The above observations also hold true between any two non-overlapping groups of activities whose levels are held in constant proportion, since they can be treated as single activities.

VI. MATRIX SCALING

In a linear program, one is quite free to rescale all rows and columns in a completely arbitrary manner. The original form of such a problem is usually stated in natural units, for which the analyst has an intuitive grasp, and must therefore always be available for analytic and diagnostic purposes where the analysts "feel" for the real world problem and magnitudes is important.

One can still inquire, however, into the possibility that another representation might, for some purposes, give a more useful "picture" and further aid the analyst in diagnostic and estimation activities. We are reminded here of the situation in Quantum Mechanics where physicists find it convenient to use "complimentary" views of the same situation for different purposes (waves vs. particles, position vs. momentum representation, etc.).

There appears, in fact, to be such a "complementary" representation of a linear program with just such intuitively appealing properties, and it is the result of rescaling all of the constraint rows of the matrix by multiplying by their shadow values, and then rescaling each of the columns to unit vectors (excluding the objective function).

The importance of this representation derives from the fact that the shadow values measure the change in objective function value obtained for marginal changes in activity levels (which act to change marginally the constraint levels which in turn must be restored by changes in other activities).

In this representation, all rows are put in comparable units, the units of the objective function (dollars for example), which also has very high intuitive content, and all coefficients of a given activity are now on a comparable scale so that relative importance is visible at a glance. Because of the further renormalization of the columns to unit vectors, each activity is in a sense comparable to the others in that it involves "one unit" in

constraint space (whether it produces or consumes, or however many constraints are involved). Furthermore, this "UNIT" is in units of the objective function, so that if units were for example, dollars, one could say that 1 unit each activity represented "a dollars worth" of effect on the problem constraints. The relative magnitudes of the rescaled objective function should then be of considerable interest.

Furthermore, in this representation, the notion of economic correlation, substitutability, and complementarity take on a particularly simple form (see the discussion on economic correlation)

$$\rho = \sum_j a_{lj} \cdot a_{ij}$$

for the correlation between activities l and i , and

$$\alpha = -\rho\delta$$

for the amount of activity i needed to best compensate for a perturbation δ in activity l . In this representation only the simple scalar product of columns is required.

Because of the unique properties of this representation, it is strongly suggested that it be considered for any proposed diagnostic aids, algorithms, substructure seeking techniques, heuristics, etc., to the extent that they depend upon quantitative values.

Finally, in practical applications, it is probably not wise to literally rescale by shadow values in the case of slack constraints, particularly if a constraint is "not-very" slack. Some number λ' between zero and its "not slack" value would seem appropriate, depending on how slack a constraint is.

VII. CONSIDERATIONS

A. Model Execution

The GLM approach may prove to give a much faster method of solving (at least approximately) a linear program, since the proposed algorithm produces not only shadow prices but activities as well.

Past experience with non-linear programs has usually been that the number of iterations required to obtain solution of a given accuracy seems to be nearly constant and independent of problem size. If this remains true of L.P.'s then there is hope of an effective method of approximate solution whose running time would be linear in the number of non-zero matrix elements, since the dominant time in performing an iteration is proportional to this quantity.

Another very useful technique for improving model execution, applicable in general to all models, algorithms, etc., which are somewhat complex and time consuming is the use of a software monitor to draw (on timer interrupt) a random sample of program locations. This sample directly estimates the distribution of running time throughout a program, and allows intelligent re-direction of programming effort to concentrate in the areas which are known to give potential payoff. There is at least one such package on the market (LEAP by Data Services Corporation) which not only gives time distribution by program memory location, but reports results by subroutines and specific call chains. This type of report is especially useful for analyzing highly subroutized FORTRAN programs.

B. Reporting of Model Contents and Results

Consideration should be given to reporting rescaled results or model contents as an alternative or in addition to any reported quantities currently contemplated, because of the high intuitive content of the objective function units (e.g. dollars), as well as the fact that all rows and columns are thereby put on a comparable, natural basis.

C. Tracing Errors

Again, the value of rescaled quantities should not be overlooked for essentially the same reasons. In particular, the behavior of the proposed algorithm for finding approximate multipliers might be especially useful in cases of infeasibility or unboundedness for helping isolate the causes.

D. Aggregation Specifications

While a complete specification of aggregation techniques is not yet clear (at least to this writer) it seems apparent that the rescaled matrix would be a good starting point because of the row and column comparability.

In particular, pairs of columns with scalar products approaching one (or minus one) would be prime candidates for aggregation. Perhaps similar treatment for the dual problem would aid in row aggregation. One can imagine "clustering" type algorithms on this basis to automatically perform some degree of aggregation.

E. Model Audit Procedures

In general, any such proposed procedures could possibly benefit from the rescaling proposed here, because of the comparability of all of the problem coefficients.

In particular, economic correlations computed as suggested here could be examined for reasonableness, with further, more intensive scrutiny in apparently contrainuitive cases.

**U.S. Department of Energy
Energy Information Administration
Office of Energy Information Services
1726 M St., N.W.
Washington, D.C. 20461**

T-CLASS MAIL
POSTAGE & FEES PAID
U.S. DEPT. OF ENERGY
PERMIT NO. G 20

FIRST CLASS MAIL

**OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, \$300**