

ORNL/TM-11261

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

Matrix Reduction Algorithms
for GRESS and ADGEN

J. E. Horwedel

OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401, FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

NTIS price codes—Printed Copy: A03 Microfiche A01

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.

Engineering Physics and Mathematics Division

**MATRIX REDUCTION ALGORITHMS
FOR GRESS AND ADGEN**

J. E. Horwedel*

*Computing and Telecommunications Division

DATE PUBLISHED — November 1989

NOTICE: This document contains information of a preliminary nature. It is subject to revision or correction and therefore does not represent a final report.

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400



CONTENTS

ABSTRACT	ix
1. INTRODUCTION	1
2. A MATHEMATICAL FOUNDATION FOR THE GRESS AND ADGEN CODE SYSTEMS	3
2.1 A MATHEMATICAL MODEL OF A COMPUTER PROGRAM	3
2.2 DIFFERENTIATING THE MATHEMATICAL MODEL	4
2.3 THE SOLUTION MATRIX	4
2.4 SOLVING THE DERIVATIVE MATRIX EQUATION	5
2.4.1 A Simple Example	5
2.4.2 Forward Substitution	7
2.4.3 Back Substitution	8
3. ALGORITHMS FOR REDUCING THE SIZE OF THE A MATRIX	11
3.1 FORWARD REDUCTION	11
3.2 BACK REDUCTION	13
3.3 PIPELINE REDUCTION	14
4. MATRIX REDUCTION APPLIED TO PRESTO-II	17
4.1 PRESTO-II BENCHMARK	17
4.2 PRESTO-II BENCHMARK WITH FORWARD AND BACK REDUCTION	18
4.3 PIPELINE REDUCTION	18
5. CONCLUSIONS AND RECOMMENDATIONS	23
REFERENCES	25

LIST OF FIGURES

Fig.		Page
2.1	A simple FORTRAN program to be used for derivative matrix generation	6
2.2	The components needed to create and solve the A matrix	6
2.3	The A matrix created by sample program	7
2.4	The matrices for solving $A*Y' = I$ for column one of Y'	7
2.5	Y' with column one resolved by forward substitution	8
2.6	The sub-matrices for solving $[A]^{tr}*[Y']^{tr} = I$ for column five of $[Y']^{tr}$	8
2.7	$[Y']^{tr}$ with the column five resolved by back substitution	9
3.1	Example program to demonstrate forward reduction	11
3.2	Complete A matrix used to demonstrate forward reduction	12
3.3	Sub-matrix extracted from a sample program A matrix using Forward Reduction	12
3.4	Sub-matrix extracted from a sample program A matrix using Back Reduction	13
3.5	The transposed A matrix for the example in Fig. 3.4	14
3.6	Transposed A matrix reduced by Pipeline Reduction	14
4.1	A plot showing the structure of the eight million row PRESTO-II $[A]^{tr}$ matrix	20

LIST OF TABLES

Table		Page
3.1	Reduction in data storage due to Forward Reduction algorithm applied to a sample problem	13
3.2	Reduction in data storage due to both Forward and Back Reduction algorithms applied to a sample problem	13
3.3	Data reduction due to Pipeline Removal algorithm	15
4.1	Resource requirements for creating and solving the A matrix created by PRESTO-II	17
4.2	Resource requirements for creating and solving the A matrix created by PRESTO-II with forward and back reduction	18
4.3	Frequency and cumulative percentage of variables defined and used within ten, 1000 row bands of the PRESTO-II $[A]^{tr}$ matrix	19
4.4	Results from seven executions of a simple Pipeline Reduction algorithm	21
4.5	Results from multiple executions of the Pipeline Reduction algorithm using the five year PRESTO-II sample problem	22

ABSTRACT

The GRESS Version 0.0 code system was developed to automate the implementation of derivative-taking capabilities in existing FORTRAN 77 computer models. The GRESS CHAIN option is used to calculate and report first derivatives of model results with respect to user selected input data by application of the calculus chain rule. The GRESS ADjoint matrix GENerator (ADGEN) option is used to calculate first derivatives of selected model results with respect to all input data. The first part of this paper presents the mathematical foundations and algorithms as presently implemented in GRESS Version 0.0. Examples are used to describe the implementation of both the CHAIN and ADGEN options. Due to excessive execution time and memory requirements with the CHAIN option users are often limited to propagating derivatives for just a few parameters. The ADGEN option allows an almost unlimited number of parameters (i.e., input data); however, the data storage requirement for an ADGEN application was more than 322 megabytes for a code that executes in 1 minute on a VAX 8600 computer.

The purpose for this paper is to present three new algorithms that could easily be implemented in GRESS Version 0.0 to dramatically reduce the data storage requirements and execution time for application of the ADGEN option. The new algorithms are described with examples. Test versions of these algorithms were implemented and tested. The application of these algorithms to the GRESS enhancement of the PRESTO-II computer model resulted in a significant reduction in execution time and a reduction in data storage requirements from 322 megabytes to 97 megabytes without any loss in the generality of the approach.



1. INTRODUCTION

In many areas of scientific computing, derivatives and sensitivities of model results to input parameters are often desired. Sensitivity analysis of computer-generated results consists of determining the effect of model data upon the calculated results of interest. The fields of sensitivity and uncertainty analyses have traditionally been dominated by statistical techniques when large-scale modeling codes are being analyzed. These methods are able to estimate sensitivities, generate response surfaces, and estimate response probability distributions. Because the statistical methods are computationally costly, they are usually applied only to problems with relatively small parameter sets. Deterministic methods, on the other hand, are very efficient and can handle large data sets, but generally require simpler models because of the considerable programming effort required for their implementation.

Since computer model equations can be differentiated analytically, sensitivities can be precisely defined and calculated in a deterministic fashion.¹⁻¹⁰ The deterministic approach is well suited to large-scale models for which direct perturbation of the model data becomes impractical from a cost standpoint. The main drawback to the deterministic approach has been the initial manpower investment to add the computational capability for calculating the necessary derivatives into existing computer models.

To circumvent this costly manpower investment and thus provide the means for model users to take advantage of the strengths of deterministic sensitivity analysis, two related software systems were developed to automate the implementation of these methods into existing FORTRAN 77 computer models. The first system, named GRESS (GRAdient-Enhanced Software System), uses a FORTRAN 77 precompiler, EXAP (EXtended Arithmetic Processor), to add derivative taking capabilities to existing FORTRAN 77 programs.¹⁰⁻¹⁷ GRESS, which has been thoroughly tested, calculates derivatives by applying the calculus chain rule to the model equations as the equations are being solved.

The second system, named ADGEN (ADjoint matrix GENerator), was developed as a GRESS option that provides the capability of automated implementation of the adjoint sensitivity methods into existing FORTRAN 77 models.¹⁸⁻²¹ ADGEN uses EXAP to automate the generation of an adjoint matrix from a computer model. Utility programs are then used to manipulate and solve the adjoint matrix for selected derivatives and sensitivities.

The purpose of this paper is to describe three new algorithms that, if implemented, would significantly reduce resource requirements for the adjoint matrix generation option (ADGEN) of GRESS Version 0.0. A general approach to the mathematical foundations is developed for GRESS and ADGEN that is based firmly in differential calculus and linear algebra. A mathematical model of a computer program, differentiating the mathematical model, and solving for derivatives is shown with examples. The numerical algorithms presently

implemented in the GRESS and ADGEN systems are described. Limitations are discussed.

The general approach is then used to develop the three matrix reduction algorithms: 1) Forward Reduction; 2) Back Reduction; and 3) Pipeline Reduction, that can easily be implemented in the present ADGEN system to significantly reduce the amount of data storage required by an ADGEN application. The matrix reduction algorithms are described with examples. The relative computational and storage requirements with and without matrix reduction are compared for the derivative enhancement of the PRESTO-II computer model.²¹

2. A MATHEMATICAL FOUNDATION FOR THE GRESS AND ADGEN CODE SYSTEMS

2.1 A MATHEMATICAL MODEL OF A COMPUTER PROGRAM

In a FORTRAN program, calculated left-hand-side variables are a function of previously defined left-hand-side variables and data, either through mathematical operations or read statements. This relationship can be expressed as

$$\bar{y} := \bar{f}(\bar{y}) \quad (1)$$

where the symbol, :=, indicates a value assignment (i.e., store) operation, the components of the column vector, \bar{y} , are all the terms on the left-hand-side of real number replacement statements, and the column vector, \bar{f} , represents the right-hand-side mathematical operations. The vector, \bar{y} , includes both model calculated results and data. Read statements are treated the same as setting a variable equal to a constant.

In a FORTRAN program a symbol cannot explicitly depend on itself. When a FORTRAN variable is redefined, mathematically, it is not the same variable. In the statement, $X := X + 5.0$, the X on the left and the X on the right represent two different locations in the solution vector, \bar{y} . Mathematically, the equation can be thought of as, $X_2 = X_1 + 5.0$

Therefore to represent equation (1) mathematically the dependence of a variable on itself must be considered explicitly. If we define

$$\frac{dy_i}{dy_i} = 1, \text{ for all } i. \quad (2)$$

Then equation (1) can be rewritten as

$$\bar{y} = \bar{f}(\bar{y}) \quad (3)$$

Consider the following four lines of FORTRAN.

```

READ (5,*) A, B, D
X = A ** 2 + B ** 2
W = A * X + D ** 2
Z = X ** 2 + W + A ** 3 + D ** 3

```

The components in Eq. (3) are

$$\bar{y} = \begin{bmatrix} A \\ B \\ D \\ X \\ W \\ Z \end{bmatrix} \quad \bar{f} = \begin{bmatrix} \text{input parameter} \\ \text{input parameter} \\ \text{input parameter} \\ A ** 2 + B ** 2 \\ A * X + D ** 2 \\ X ** 2 + W + A ** 3 + D ** 3 \end{bmatrix}$$

2.2 DIFFERENTIATING THE MATHEMATICAL MODEL

Differentiating Eq. (3), with respect to \bar{y} , yields

$$\frac{d\bar{y}}{d\bar{y}} = \frac{\partial \bar{f}}{\partial \bar{y}} \cdot \frac{d\bar{y}}{d\bar{y}} + I \quad (4)$$

where the identity matrix, I , provides the explicit dependence of a variable on itself necessary to make Eq. (4) meaningful. Eq. (4) can be rearranged, such that

$$\left[I - \frac{\partial \bar{f}}{\partial \bar{y}} \right] \bullet \frac{d\bar{y}}{d\bar{y}} = I \quad . \quad (5)$$

Eq. (5) can be represented in a more compact form as,

$$AY' = I \quad (6)$$

where

$$A = \left[I - \frac{\partial \bar{f}}{\partial \bar{y}} \right]$$

and

$$Y' = \frac{d\bar{y}}{d\bar{y}}.$$

Since FORTRAN equations are solved in a sequential fashion, FORTRAN variables are dependent on previously defined variables. Therefore,

$$\frac{\partial f_i}{\partial y_j} = 0, \text{ for } j \geq i$$

so that the matrix, $\frac{\partial \bar{f}}{\partial \bar{y}}$, is a lower triangular matrix with zeros on and above the diagonal. Therefore, the matrix, A , is nonsingular and invertible.

Solving Eq. (6) for Y' yields

$$Y' = A^{-1} \quad (7)$$

Since A is a lower triangular matrix, derivatives in the solution matrix Y' , can easily be resolved using forward substitution. Also, since the transpose of A is upper triangular, components of Y' can be calculated using back substitution as represented by Eq. (8).

$$[Y']^{tr} = [A^{tr}]^{-1} \quad . \quad (8)$$

The tr superscript is used to represent the transpose of the matrix.

2.3 THE SOLUTION MATRIX

The purpose of this section is to look at the contents of the solution matrix, Y' . The matrix, Y' , represented in Eq. (7) contains the total first derivatives of


```

E = 2.0
B = 3.0
D = 5.0
X = E ** 2 + B ** 2
W = E * X + D ** 2
Z = X ** 2 + W + E * D
END

```

Fig. 2.1. A simple FORTRAN program to be used for derivative matrix generation.

Shown in Fig. 2.2 are the various components used by GRESS to create the A matrix.

Row	Y	F	Result	Partial Derivatives
1	E	2.0		$\frac{\partial f}{\partial y} = 0$
2	B	3.0		$\frac{\partial f}{\partial y} = 0$
3	D	5.0		$\frac{\partial f}{\partial y} = 0$
4	X	$E ** 2 + B ** 2$	13.0	$\frac{\partial X}{\partial E} = 4$ $\frac{\partial X}{\partial B} = 6$
5	W	$E * X + D ** 2$	51.0	$\frac{\partial W}{\partial E} = 13$ $\frac{\partial W}{\partial D} = 10$ $\frac{\partial W}{\partial X} = 2$
6	Z	$X ** 2 + W + E * D$	230.0	$\frac{\partial Z}{\partial E} = 5$ $\frac{\partial Z}{\partial D} = 2$ $\frac{\partial Z}{\partial X} = 26$ $\frac{\partial Z}{\partial W} = 1$

Fig. 2.2. The components needed to create and solve the A matrix.

The A matrix created using the partial derivatives from the sample program is shown in Fig. 2.3. The row number identifies the dependent term in the equation, the column number identifies the independent term. The first FORTRAN variable defined during execution becomes row one in the A matrix, the second defined variable becomes row two, etc. The column is determined by the row in which the right-hand-side term was defined. For j less than i , the matrix element in row i , column j identifies the negative of the partial derivative of the term defined in row i with respect to the term defined in row j .

	Columns					
Rows	1	2	3	4	5	6
1	1					
2	0	1			(0)	
3	0	0	1			
4	-4	-6	0	1		
5	-13	0	-10	-2	1	
6	-5	0	-2	-26	-1	1

Fig. 2.3. The A matrix created by sample program.

2.4.2 Forward Substitution

Using the A matrix shown in Fig. 2.3, the Y' matrix for the sample program can be fully resolved by forward substitution. This is the same as simply calculating selected derivatives with the calculus chain rule. For example the derivative of \bar{y} with respect to E , could be calculated using the chain rule, as follows.

$$\begin{aligned} \frac{dE}{dE} &= 1, \frac{dB}{dE} = 0, \frac{dD}{dE} = 0 \\ \frac{dX}{dE} &= 2 * E = 4 \\ \frac{dW}{dE} &= X + E * \frac{dX}{dE} = 21 \\ \frac{dZ}{dE} &= 2 * X * \frac{dX}{dE} + \frac{dW}{dE} + D = 130 \end{aligned}$$

When using forward substitution to solve for the derivatives of \bar{y} with respect to E , only column one of Y' has to be resolved. The matrices set up for forward substitution are shown in Fig. 2.4. The non-zero locations in Y' are indicated with a question mark (?). To invert column one it is necessary to resolve every non-zero term in column one.

$$\begin{bmatrix} 1 & & & & & & \\ 0 & 1 & & & & & (0) \\ 0 & 0 & 1 & & & & \\ -4 & -6 & 0 & 1 & & & \\ -13 & 0 & -10 & -2 & 1 & & \\ -5 & 0 & -2 & -26 & -1 & 1 & \end{bmatrix} * \begin{bmatrix} 1 & & & & & & \\ ? & 1 & & & & & (0) \\ ? & ? & & & & & \\ ? & ? & 1 & & & & \\ ? & ? & ? & 1 & & & \\ ? & ? & ? & ? & 1 & & \\ ? & ? & ? & ? & ? & 1 & \\ & & & & & ? & 1 \end{bmatrix}$$

Fig. 2.4. The matrices for solving $A * Y' = I$ for column one of Y' .

By successively multiplying each row in the A matrix by column one in Y' , the unknown terms in column one can be quickly resolved. The Y' matrix with column one resolved is shown in Fig. 2.5.

$$\begin{bmatrix} 1 & & & & & & \\ 0 & 1 & & & & & \\ 0 & ? & 1 & & & & \\ 4 & ? & ? & 1 & & & \\ 21 & ? & ? & ? & 1 & & \\ 130 & ? & ? & ? & ? & 1 & \end{bmatrix} \quad (0)$$

Fig. 2.5. Y' with column one resolved by forward substitution.

The GRESS CHAIN option calculates the derivatives using forward substitution in memory. Derivatives with respect to declared parameters are propagated forward as the enhanced model is executing. The actual A matrix is never stored. At any given point during execution, the user can retrieve the total first derivatives for a calculated variable with respect to all the declared parameters. Since GRESS does not know *a priori* which results are of interest, it is necessary to propagate the derivatives for the user selected parameters through every row in the matrix. This causes the amount of execution time and memory required to calculate derivatives using the CHAIN option to increase rapidly with the number of declared parameters. In practice with large codes, the user is often limited to a few parameters per run.

2.4.3 Back Substitution

By transposing the A matrix from Fig. 2.3, and setting up the matrices in Eq. (8), back substitution can be used to solve for the derivatives of Y_i with respect to \bar{y} . The derivatives of Y_i with respect to \bar{y} are in the i^{th} column of $[Y']^{tr}$. Shown in Fig. 2.6, are the sub-matrices necessary to calculate the derivative of W with respect to \bar{y} (i.e., $\{E, B, D, X, W, Z\}$).

$$\begin{bmatrix} 1 & 0 & 0 & -4 & -13 & -5 \\ & 1 & 0 & -6 & 0 & 0 \\ & & 1 & 0 & -10 & -2 \\ (0) & & & 1 & -2 & -26 \\ & & & & 1 & -1 \\ & & & & & 1 \end{bmatrix} * \begin{bmatrix} 1 & ? & ? & ? & ? & ? \\ & 1 & ? & ? & ? & ? \\ & & 1 & ? & ? & ? \\ (0) & & & 1 & ? & ? \\ & & & & 1 & ? \\ & & & & & 1 \end{bmatrix}$$

Fig. 2.6. The sub-matrices for solving $[A]^{tr}*[Y']^{tr} = I$ for column five of $[Y']^{tr}$.

Back substitution can be used to resolve column five. Shown in Fig. 2.7. is the $[Y']^{tr}$ with column five completely resolved.

$$\begin{bmatrix} 1 & ? & ? & ? & 21 & ? \\ & 1 & ? & ? & 12 & ? \\ & & 1 & ? & 10 & ? \\ & (0) & & 1 & 2 & ? \\ & & & & 1 & ? \\ & & & & & 1 \end{bmatrix}$$

Fig. 2.7. $[Y']^{tr}$ with the column five resolved by back substitution.

From Fig. 2.7, we see that

$$\left[\frac{dw}{dy} \right]^{tr} = \begin{bmatrix} \frac{dW}{dE} \\ \frac{dW}{dB} \\ \frac{dW}{dD} \\ \frac{dW}{dX} \\ \frac{dW}{dW} \\ \frac{dW}{dZ} \end{bmatrix} = \begin{bmatrix} 21 \\ 12 \\ 10 \\ 2 \\ 1 \\ 0 \end{bmatrix}$$

The ADGEN system uses the GRESS adjoint matrix generation option to create the A matrix. ADGEN utilities are used to transpose the A matrix, and then to solve for the derivatives in a selected column. The A matrix created by GRESS Version 0.0 can be excessively large. For a moderately sized program, it is not uncommon for the A matrix to have on the order of 10^7 rows. If only the non-zero terms are stored, the data set can require as much as 200 megabytes of storage.

There are three pieces of information stored for each row.

- 1) Np = number of non-zero derivatives
- 2) C 's = column numbers for the right-hand-side terms
- 3) D 's = derivatives with respect to right-hand-side terms

For each row, values for items 2 and 3 are repeated Np times. The row number of the left-hand-side term is not stored, since it can be determined during processing. The present version uses four byte words for Np , D , and C . Therefore, eight bytes are required for each non-zero derivative in the row, and an additional four bytes for each row to store the derivative count, Np . The amount of storage required for either the forward or transpose matrix can be estimated as follows.

$$\text{Storage} = (\# \text{ of Rows}) * [4 + (\text{average } Np) * 8] \text{ bytes} \quad (9)$$

For a matrix with 10^7 rows with an average of two non-zero terms off the diagonal per row, the amount of storage can be estimated as follows.

10

$$\text{Storage} = (10^7) * [4 + (2 * 8)] \text{ bytes} = 200 \text{ megabytes}$$

In application, this amount of storage has been required for codes that used 1 minute of execution time on a VAX 8600 prior to enhancement.

3. ALGORITHMS FOR REDUCING THE SIZE OF THE A MATRIX

Presented in this chapter are three matrix reduction algorithms that can be easily implemented in the present GRESS and ADGEN systems to reduce the resource requirements required by an enhanced model. Mathematically, the matrix reduction techniques are based on the fact that the rows in the A matrix are linearly independent and the matrix is nonsingular. It can easily be proven that any sub-matrix from a linearly independent, nonsingular matrix, is also linearly independent and nonsingular.²² This means that if a row and corresponding column (i.e., row i , column i) are removed from the matrix, the remaining matrix is still linearly independent and nonsingular. Therefore, Eqs. (1-8) are still valid for any sub-matrix extracted from the A matrix. Since parameters and potential responses of interest must be declared by the user, and are "known" during execution, information is available that can be used to extract a problem dependent sub-matrix, thus reducing the amount of data stored, and the number of calculations required to solve for selected derivatives.

3.1 FORWARD REDUCTION

When creating the A matrix, partial derivatives are calculated and stored for every equation solved. However, only derivatives in rows that are dependent on user specified parameters are actually needed. Forward Reduction keeps track of parameter dependency. If any term on the right-hand side of an equation is dependent on a declared parameter, then the term being calculated is also dependent; and therefore, the row is needed. If there is no dependency on the right, then the row is not needed. For the example in Fig. 3.1, B and C are declared to be parameters. The variable, R , is specified as a result of interest for derivative calculation. Using Forward Reduction, a problem dependent sub-matrix can be extracted that includes those terms that are dependent on B and C .

```

B = 2
C = 3           Parameters of interest = B, C
D = 4
X = B + D
Y = D ** 2 + B ** 2
R = 7 * X + D ** 2   Result of interest = R
S = Y ** 2
END

```

Fig. 3.1. Example program to demonstrate forward reduction.

The complete A matrix is shown in Fig. 3.2. Since only derivatives with respect to parameters B and C are of interest, the problem dependent sub-matrix, shown in Fig. 3.3, can be extracted. The variable D was eliminated from the matrix using forward reduction. Though R is dependent on D , D is not included as a parameter; therefore, D may be treated as a constant. S was not eliminated by forward reduction because it is dependent on parameter B .

The symbol names (i.e, B, C, D, X, Y, R , and S) are included in Figs. 3.2 -- 3.6 to associate the terms in the matrix with the variables in the sample program.

	B	C	D	X	Y	R	S
B	1						
C	0	1			(0)		
D	0	0	1				
X	-1	-1	0	1			
Y	-4	0	-8	0	1		
R	0	0	-8	-7	0	1	
S	0	0	0	0	-40	0	1

Fig. 3.2. Complete A matrix used to demonstrate forward reduction.

Using the sub-matrix shown in Fig. 3.3, it is possible to calculate the derivatives of R with respect to B and C . These are the same derivatives that can be calculated with the complete A matrix, by either forward or back substitution. Since only the derivatives for specified results with respect to declared parameters are retrievable, there is no loss of usable information during the Forward Reduction process.

	B	C	X	Y	R	S
B	1					
C	0	1			(0)	
X	-1	-1	-1			
Y	-4	0	0	1		
R	0	0	-7	0	1	
S	0	0	0	-40	0	1

Fig. 3.3. Sub-matrix extracted from a sample program A matrix using Forward Reduction.

Table 3.1 shows the reduction in data storage that occurs in this sample program due to Forward Reduction. What can not be easily shown in a small program is the overall impact to execution time. However, in most applications, the amount of execution time to write the A matrix to disk is significant. By removing rows from the matrix, not only is the data storage reduced; but also, in most applications the execution time decreases. Reduction in execution time is discussed in the application to PRESTO-II in Chapter 4.

3.3 PIPELINE REDUCTION

The third algorithm discussed is Pipeline Reduction. This method takes advantage of the fact that in FORTRAN programs, memory locations tend to be re-used often. When a variable is re-defined, its FORTRAN symbol becomes associated with a different row in the matrix, and the associated column in the A matrix is truncated. The variable associated with that row will never appear on the right-hand side of an equation again. If a mapping variable (one that is not a parameter or a response) appears on the right-hand side of an equation only one time, then data storage can be reduced. The method is applied to the A matrix after completion. At that point, knowledge of the entire memory-use pattern is available. By reading the A matrix from bottom-to-top (read the last row first, then read the next-to-last row, etc.), it is possible to know where variables are used, as well as knowing where they were defined. If a variable is defined very near where it is used, and if the variable was never used again, then it is possible to apply the Pipeline Reduction algorithm to reduce the amount of data stored.

In the example, the FORTRAN variable, X , can be considered to be a mapping variable. The transposed A matrix from Fig. 3.4 is shown in Fig. 3.5. By using the calculus chain rule, the derivative information associated with the variable X can be moved to the column associated with R by simply multiplying the non-zero derivatives in column X by $\frac{\partial R}{\partial X}$ (i.e., 7), and adding them to the non-zero derivatives in column R .

	B	C	X	R
B	1	0	-1	0
C		1	-1	0
X			1	-7
R		(0)		1

Fig. 3.5. The transposed A matrix for the example in Fig. 3.4.

Once the processing has passed the row associated with X , that row can be dropped from the matrix. This is analogous to removing X from the FORTRAN model by re-writing the program. The reduced matrix is shown in Fig. 3.6.

	B	C	R
B	1	0	-7
C		1	-7
R	(0)		1

Fig. 3.6. Transposed A matrix reduced by Pipeline Reduction.

The reduction in data storage by removing column X' from the matrix is shown in Table 3.3.

Table 3.3. Data reduction due to Pipeline Removal algorithm.

<u>REDUCTION ALGORITHM</u>	<u>TOTAL STORAGE</u> (bytes)
None	76
Forward	56
Forward and Back	36
Forward, Back and Pipeline	28
RESULT = 63% reduction in data stored	

At first glance, a Pipeline Reduction algorithm does not necessarily appear to be very useful. However, as will be shown with the PRESTO-II example, a majority of variables in FORTRAN programs tend to be defined near where they are used. This results in a partially banded matrix. The Pipeline Reduction algorithm is primarily concerned with the memory locations that are used only near where they are defined (those in the band near the diagonal).

With forward substitution, as presently implemented in GRESS Version 0.0, there is no *a priori* knowledge as to where, and how often, a variable is used. As has been shown, once the matrix is created, information about variable usage is available. But what is also true is that once a variable is re-defined, we know that its associated column in the A matrix is truncated. By maintaining an output buffer large enough to hold several thousand rows of the matrix in memory, it would be possible to apply the Pipeline Reduction algorithm during the execution of the enhanced code. If ten thousand rows are kept in the matrix output buffer, in essence, the code can "see" ten thousand rows ahead as to how memory locations are being used. The required buffer size can be estimated using Eq. (6). If the average Np is two, then the amount of memory needed to maintain a buffer of 10,000 rows would be 50,000 four byte words. With the PRESTO-II application, discussed in Chapter 4, the band that contains 58.8% of the variables used in the model is only 10,000 rows wide.

4. MATRIX REDUCTION APPLIED TO PRESTO-II

The PRESTO-II computer model²³ was used as a benchmark for the GRESS Version 0.0 adjoint matrix generation option, ADGEN. A more complete discussion of the benchmark is included in Ref. 21. In this chapter, the computational and storage requirements for an ADGEN application with and without matrix reduction are compared for the derivative enhancement of the PRESTO-II computer model. The banded nature of the PRESTO-II derivative matrix is shown. A simple Pipeline Reduction algorithm is tested.

4.1 PRESTO-II BENCHMARK

PRESTO-II was developed as a non site-specific screening model for evaluating the possible health effects due to shallowland disposal of radioactive waste. The model has approximately 6,900 lines of coding. The PRESTO-II computer resource requirements are for the Barnwell sample problem included in Ref. 23. This problem calculates a time-dependent radiation dose to a man from transport of 42 radionuclides over a one thousand year time span.

PRESTO-II was enhanced with the EXAP precompiler, GRESS Version 0.0. Derivatives for two results with respect to 2800 parameters were calculated. The resource requirements for creating and solving the $[A]$ matrix are shown in Table 4.1. The PRESTO-II $[A]$ matrix had more than eight million rows and 322 megabytes of direct access storage were required to create the $[A]^{tr}$ matrix. The $[A]$ matrix is considered a scratch data set, and may be deleted once $[A]^{tr}$ is created.† However, the $[A]^{tr}$ matrix (144 megabytes) must remain active for the YSOLVE step. YSOLVE reads $[A]^{tr}$ and calculates the derivatives for one result with respect to all the declared parameters (i.e., a column of $[Y]^{tr}$). The YSOLVE step can be re-executed for additional results of interest that were specified during the execution of the enhanced model.

Table 4.1. Resource requirements for creating and solving the A matrix created by PRESTO-II.

<u>Job Step</u>	<u>Run Time (Min:Sec)</u>	<u>Data Set created</u>	<u>Storage (megabytes)</u>
Enhanced Presto-II	24:51	$[A]$	178
TMAT*	8:09	$[A]^{tr}$	144
YSOLVE (1 result)*	4:59		
			322 (total)

* TMAT and YSOLVE are described in "GRESS Version 0.0 User's Manual."

† A GRESS utility, TMAT, is used to convert the $[A]$ matrix into the $[A]^{tr}$ matrix.

4.2 PRESTO-II BENCHMARK WITH FORWARD AND BACK REDUCTION

Simple modifications to the GRESS/ADGEN run-time library routines were made to implement the Forward Reduction algorithm. Results, summarized in Table 4.2, show a decrease in the size of $[A]^{tr}$ from 144 megabytes to 86 megabytes due to the Forward Reduction algorithm, alone. Because of the nature of the algorithm changes, the intermediate step of creating a separate forward matrix, then transposing it, was also eliminated. This removed the TMAT utility from the calculational sequence.

Table 4.2. Resource requirements for creating and solving the A matrix created by PRESTO-II with forward and back reduction.

Job Step	Run Time (Min:Sec)	Data Set created	Storage (megabytes)
Enhanced model	24:01	$[A]^{tr}$	86
BREDUCE ^a	3:32	$[A]^{tr}$	11
YSOLVE	:34		
			97(total)

^a BREDUCE is test program for implementing the Back Reduction algorithm that creates a subset of $[A]^{tr}$.

The Back Reduction algorithm was implemented in a utility program, BREDUCE, to be executed after the creation of the A matrix. BREDUCE was used to create the sub-set of $[A]^{tr}$ that contains only those terms on which the results of interest (as specified by the user in the run step) depend. The results after execution of the Back Reduction algorithm show the amount of data storage to be reduced to 11 megabytes. The execution time to reduce the matrix and solve for derivatives for one response with respect to 2800 parameters is less than the time it takes to solve for one response with the entire matrix. The time to solve for a second result would be 34 seconds, as opposed to the nearly 5 minutes required to solve the unreduced matrix. In performing forward and back reduction as described no data or results of interest are removed from the A matrix. There is no loss in the generality of the adjoint approach as implemented in GRESS Version 0.0.

4.3 PIPELINE REDUCTION

The purpose of this section is to motivate the further study and development of Pipeline Reduction techniques. There are two ways to implement a Pipeline Reduction algorithm: (1) a Pipeline Reduction algorithm could be implemented on the forward run of an enhanced model to help reduce the size of the $[A]^{tr}$ matrix, prior to its creation; or (2) Pipeline Reduction can be implemented in a utility program to be executed after the matrix is created.

The use of pipeline reduction on the forward run would be designed to try to minimize the size of the actual matrix written to disk, by removing local variables

(those that are defined and then used only one time, near the point of definition). For small models, it is conceivable that the combination of Forward and Pipeline Reduction could result in the entire matrix being maintained in virtual memory until after the execution of the Back Reduction algorithm. For PRESTO-II, this would reduce the total required disk storage from 86 megabytes to less than 11 megabytes.

The advantage of implementing Pipeline Reduction algorithm in a utility program to be executed after the matrix is created, is that it could be used to further reduce the size of the $[A]^{tr}$ matrix prior to solving.

Shown in Fig. 4.1, is a plot of $[A]^{tr}$ showing its banded nature and sparseness. Each dot in the matrix plot represents a 10,000-by-10,000 sub-matrix with at least one non-zero partial derivative. It is not a density plot. Terms that are defined near where they are used, appear near the diagonal of the matrix. Terms that are used far away from where they are defined appear in sub-matrices above the diagonal. Pipeline Reduction is concerned with the terms that are defined near where they are used, and then, only used one or two times.

The banded nature of the PRESTO-II $[A]^{tr}$ matrix is shown in Table 4.3. To generate this data, the area within 10,000 rows of the diagonal for the PRESTO-II $[A]^{tr}$ matrix was broken into ten vertical bands, each 1000 rows high. Each non-zero partial derivative on the right-hand side of an equation is placed in a band relative to the row number (or band) of the equation. For example, a term that is defined within 1000 rows of where it appears on the right-hand side of an equation will be in band 1; a term defined within 2000 rows will be in band 2; etc. The results show that 58.8 percent of the non-zero derivatives are used within ten thousand rows of where they are defined.

Table 4.3. Frequency and cumulative percentage of variables defined and used within ten, 1000 row bands of the diagonal for the PRESTO-II $[A]^{tr}$ matrix.

<u>Band Width</u> <u>(1000 rows)</u>	<u>Frequency</u>	<u>Cumulative</u> <u>Frequency</u>	<u>Cumulative</u> <u>Percent</u>
1	6934351	6934351	49.7
2	242738	7177089	51.4
3	132713	7309802	52.4
4	143017	7452819	53.4
5	144551	7597370	54.4
6	131345	7728715	55.4
7	47409	7776124	55.7
8	252993	8029117	57.5
9	173120	8202237	58.7
10	3411	8205648	58.8

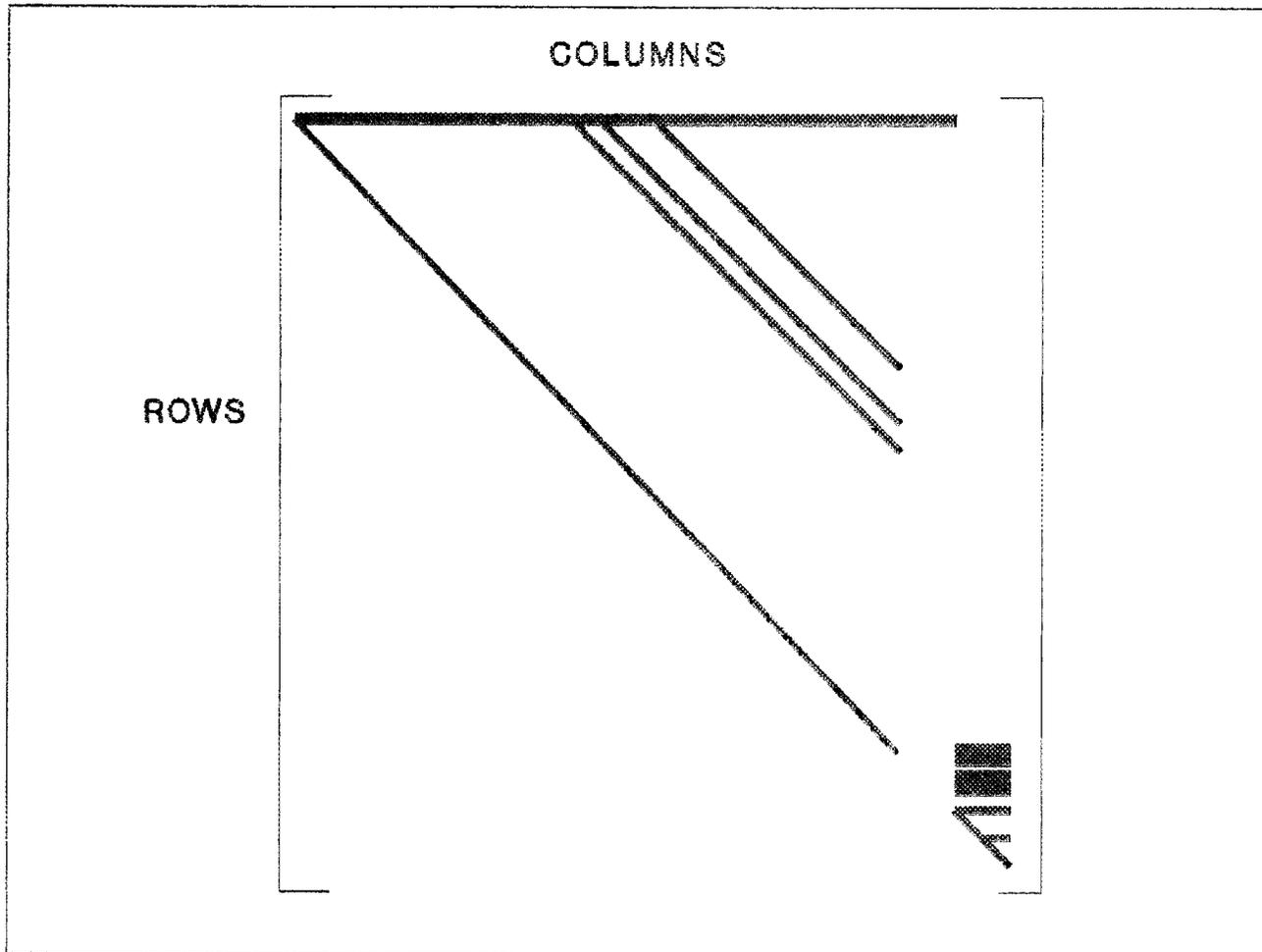


Fig. 4.1. A plot showing the structure of the eight million row PRESTO-II $[A]^{tr}$ matrix. (Each dot in this matrix represents a 10,000 row by 10,000 row sub-matrix with at least one non-zero location.)

The nature of FORTRAN programs is clearly shown in Fig. 4.1, and Table 4.3. FORTRAN programs work with a limited memory space. A significant number of memory locations are defined, or re-defined, and then used only one time, near the point of definition. After the $[A]^{tr}$ matrix is created, the complete history of the memory use is available. By simply reading the matrix from right-to-left (which is equivalent to looking at the last equation solved in the enhanced model, then next to the last equation, etc.), a simple algorithm could be implemented that checks the row number of the terms on the right-hand side of the equation and compares that to the row number of the equation itself. If the right-hand-side term is defined within a specified number of rows of where it is used, and it is never used again, then it can be flagged as a mapping term and removed by application of the Pipeline Reduction algorithm.

No attempt was made to implement Pipeline Reduction on the forward run of the model; however, a limited version of a Pipeline Reduction algorithm was implemented in a utility program for execution after the matrix is created. To keep it simple, only one term was removed from a column in a single pass through the matrix. However, the reduction program can be re-run using the reduced matrix as input. Each re-execution of the reduction program will remove additional terms from the matrix. Shown in Table 4.4 are the results from seven successive executions of the Pipeline Reduction algorithm.

Table 4.4. Results from seven executions of a simple Pipeline Reduction algorithm.

<u>Step</u>	<u>$[A]^{tr}$ Storage (megabytes)</u>	<u>Retrieval Time¹ (Seconds)</u>
No Reduction	322	299
Forward Reduction	85	127
Forward and Back Reduction	11	34
PLINE ² (1 executions)	8	24
PLINE (7 executions)	6	18

¹ The retrieval time is the execution time required to retrieve and report the derivatives for a requested result with respect to the 2800 declared parameters.

² PLINE is the utility program that implements a simplified version of the Pipeline Reduction algorithm.

Clearly a Pipeline Reduction algorithm can be used to reduce the data storage required to store the $[A]^{tr}$ matrix. By reducing the size of the matrix, not only are we saving storage, but the access time to retrieve a derivative is also decreasing. If the data storage were reduced to the point where only parameters and responses remained in the matrix, the matrix is completely solved. Also, the matrix is stored in a structure that allows quick access to any derivative requested. The potential for

using this methodology to automatically generate a response surface model should be investigated. To further understand this potential, the first 647,000 rows of the PRESTO-II matrix were further reduced using the Pipeline algorithm. (The first 647,000 rows represent running the PRESTO-II Barnwell sample problem for five years, rather than the full 1000 years.)

The results summarized in Table 4.5 were achieved by executing the simple pipeline routine 135 times. A more sophisticated implementation of the algorithm should be able to arrive at the same point with one or two executions; however, the results demonstrate the potential for such an algorithm.

Table 4.5. Results from multiple executions of the Pipeline Reduction algorithm using the five year PRESTO-II sample problem.

<u>Step</u>	<u>$[A]^{tr}$ Storage (megabytes)</u>	<u>Retrieval Time (Seconds)</u>
No Reduction	24.050	51
Forward Reduction	5.992	21
Forward and Back Reduction	0.260	6
PLINE (12 executions)	0.063	3
PLINE (75 executions)	0.033	1
PLINE (135 executions)	0.003	< 0.5

5. CONCLUSIONS AND RECOMMENDATIONS

The algorithms as presently implemented in the GRESS and ADGEN systems for calculating derivatives have limitations. With the GRESS CHAIN option, one can be limited to a few parameters due to limited memory available. Also, the execution time can increase significantly with each additional parameter of interest. Conversely, ADGEN can require an excessive amount of data storage. The A matrix for a moderately sized code could easily be greater than 200 megabytes.

Because of the size of the entire A matrix, the calculational efficiency of first extracting a problem dependent sub-matrix, and then calculating the derivatives of interest is apparent. The three matrix reduction algorithms presented in this report were used to extract a problem dependent sub-matrix from the PRESTO-II application, thus significantly reducing the amount of data storage and execution time required to calculate derivatives of selected results with respect to declared parameters.

The Forward and Back Reduction algorithms need to be implemented and fully tested as soon as possible. A more sophisticated Pipeline Reduction program for use after the matrix is created should be developed. This version should reduce the matrix as much as possible on a single pass, rather than, requiring multiple executions.

The possibility of using Pipeline Reduction on the forward pass warrants thorough testing. By maintaining a matrix buffer that includes 50,000 rows, it is possible to see the memory use pattern, 50,000 rows ahead of the row being output. For PRESTO-II, this is five times the band-width containing 59 percent of the non-zero terms. A fully implemented Pipeline Reduction algorithm on the forward run of the enhanced model could be used to significantly reduce data storage of the initial $[A]^{tr}$ data set.

REFERENCES

1. Tomovic, R. and Vukobratovic, M., *General Sensitivity Theory*, Elsevier North-Holland, Inc., New York (1972).
2. Stacey, W. M., *Variational Methods in Nuclear Reactor Physics*, Academic Press, New York (1974).
3. Greenspan, E., "Developments in Perturbation Theory," *Advances in Nuclear Science and Technology*, Vol. 9, Academic Press, New York (1976).
4. Oblow, E. M., "Sensitivity Theory for Reactor Thermal-Hydraulics Problems," *Nucl. Sci. Eng.* **68**, 322 (1978).
5. Frank, P. M., *Introduction to System Sensitivity Theory*, Academic Press, New York (1978).
6. Kedem, Gershon, "Automatic Differentiation of Computer Program," *ACM Transactions on Mathematical Software* **6**, No. 2, June 1980, pp. 150-165.
7. Cacuci, D. G., *J. Math. Phys.* (NY) **22**, 2794 (1981).
8. Cacuci, D. G., *J. Math. Phys.* (NY), **22**, 2803 (1981).
9. Weisbin, C. R. et al, "Sensitivity and Uncertainty Analysis of Reactor Performance Parameters," *Advances in Nuclear Science and Technology* **14**, Plenum Press, New York (1982).
10. Oblow, E. M., *An Automated Procedure for Sensitivity Analysis Using Computer Calculus*, ORNL/TM-8776, Oak Ridge National Laboratory, Oak Ridge, Tennessee, May 1983.
11. Oblow, E. M., "GRESS, Gradient-Enhanced Software System," ORNL/TM-9658, Oak Ridge National Laboratory, Oak Ridge, Tennessee, July 1985.
12. Oblow, E. M., F. G. Pin, R. Q. Wright, "Sensitivity Analysis Using Computer Calculus: A Nuclear Waste Isolation Application," *Nucl. Sci. Eng.* **94**, 46 (1986).
13. Worley, B. A., R. Q. Wright, F. G. Pin, W. V. Harper, "Application of an Automated Procedure for Adding a Comprehensive Sensitivity Calculation Capability to the ORIGEN2 Point Depletion and Radioactivity Decay Code," *Nucl. Sci. Eng.* **94**, 180 (1986).
14. Pin, F. G., B. A. Worley, E. M. Oblow, R. Q. Wright, W. V. Harper, "An Automated Sensitivity Analysis Procedure for the Performance Assessment of Nuclear Waste Isolation Systems," *Nucl. and Chem. Waste Man.* **6**, 255 (1986).
15. Worley, B. A., R. Q. Wright, F. G. Pin, *A Finite-Line Heat Transfer Code with Automated Sensitivity-Calculation Capability*, ORNL/TM-9975, Oak Ridge National Laboratory, Oak Ridge, Tennessee, September 1986.
16. Worley, B. A. and J. E. Horwedel, *A Waste Package Performance Assessment Code with Automated Sensitivity-Calculation Capability*, ORNL/TM-9976, Oak Ridge National Laboratory, Oak Ridge, Tennessee, September 1986.

17. Horwedel, J. E., Worley, B. A., Oblow, E. M., Pin, F. G., and Wright, R. Q., *GRESS Version 0.0 User's Manual*, ORNL/TM-10835, Oak Ridge National Laboratory, Oak Ridge, Tennessee, October 1988.
18. Pin, F. G. and E. M. Oblow, "Sensitivity Analysis of Predictive Models with an Automated Adjoint Generator," *Proceedings of Ninth Annual DOE Low-Level Waste Management Conference*, Denver, Colorado, August 25-27, 1987.
19. Pin, F. G., E. M. Oblow, J. E. Horwedel, and J. L. Lucius, "ADGEN: An Automated Adjoint Code Generator for Large-Scale Sensitivity Analysis," *Trans. Am. Nuc. Soc.* **55**, 311 (1987).
20. Horwedel, J. E., F. G. Pin, B. A. Worley, and E. M. Oblow, "EXAP-Precompiler for the GRESS and ADGEN Automated Sensitivity Calculation Systems." *Trans. Am. Nuc. Soc.* **56**, 301 (1988).
21. Worley, B. A., F. G. Pin, J. E. Horwedel, and E. M. Oblow, *ADGEN-ADjoint GENERator For Computer Models*, ORNL/TM-11037, Oak Ridge National Laboratory, Oak Ridge, Tennessee, May 1989.
22. Kohlman, B., *Elementary Linear Algebra*, The MacMillan Company, New York (1971).
23. Fields, D. E., et al., *PRESTO-II: A Low-Level Waste Environmental Transport and Risk Assessment Code*, ORNL-5970, Oak Ridge National Laboratory, Oak Ridge, Tennessee (1986).

INTERNAL DISTRIBUTION

- | | |
|--------------------------|---|
| 1. R. G. Alsmiller, Jr. | 30-34. F. G. Pin |
| 2. B. R. Appleton | 35. R. J. Raridon |
| 3. D. L. Barnett | 36. R. W. Roussin |
| 4. M. Beckerman | 37. T. H. Row |
| 5. A. G. Croff | 38. B. A. Worley |
| 6. J. R. Einstein | 39. R. Q. Wright |
| 7. C. W. Glover | 40-44. RSIC |
| 8. W. R. Hamel | 45-49. EP&MD Reports Office |
| 9-13. J. E. Horwedel | 50-51. Laboratory Records
Department |
| 14. J. P. Jones | 52. Laboratory Records,
ORNL-RC |
| 15. D. C. Kocher | 53. Document Reference
Section |
| 16. R. E. Maerker | 54. Central Research Library |
| 17-21. F. C. Maienschein | 55. ORNL Patent Section |
| 22-26. R. C. Mann | |
| 27. J. R. Merriman | |
| 28. E. M. Oblow | |
| 29. W. H. Pechin | |

EXTERNAL DISTRIBUTION

56. Office of Assistant Manager for Energy Research and Development, DOE-ORO, Oak Ridge, TN 37830
57. E. L. Albenesius, Savannah River Laboratory, U.S. DOE, P.O. Box A, Aiken, SC 29801
58. Donald Alexander, U.S. DOE, Civilian Radioactive Waste Management Program, 1000 Independence Ave., SW, Washington, DC 20585
59. R. E. Allan, Ontario Hydro, 700 University Ave., H18D16, Toronto, Ontario, M5G1X6, CANADA
60. M. Apted, Pacific Northwest Laboratory, P.O. Box 999, Richland, WA 99352
61. R. Baca, Idaho National Engineering Laboratory, EG&G Idaho, Inc., P.O. Box 1625, Idaho Falls, ID 83415
62. E. J. Bonano, Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185
63. L. C. Brown, Westinghouse Hanford Co., P.O. Box 1970, Richland, WA 99352
64. J. W. Cammann, Westinghouse Hanford Co., P.O. Box 1970, Richland, WA 99352
65. M. J. Case, Idaho National Engineering Laboratory, EG&G Idaho, Inc., P.O. Box 1625, Idaho Falls, ID 83415
66. C. Cole, Pacific Northwest Laboratory, P.O. Box 999, Richland, WA 99352
67. R. Collier, Research and Waste Management Division, U.S. DOE-ORO, Oak Ridge, TN 37830
68. J. R. Cook, Savannah River Laboratory, U.S. DOE, P.O. Box A, Aiken, SC 29801
69. R. Cranwell, Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185

70. R. Curl, Idaho National Engineering Laboratory, EG&G Idaho, Inc., P.O. Box 1625, Idaho Falls, ID 83415
71. J. B. Czarnecki, United States Geological Survey, Box 25046, Denver Federal Center, Denver, CO 80225
72. D. H. Dahlem, U.S. DOE-BWIPO, Richland, WA 99352
73. C. DeFigh-Price, Westinghouse Hanford Co., P.O. Box 1970, Richland, WA 99352
74. P. Doctor, Pacific Northwest Laboratory, P.O. Box 999, Richland, WA 99352
75. J. J. Dorning, Department of Nuclear Engineering and Physics, Thornton Hall, McCormick Road, University of Virginia, Charlottesville, VA 22901
76. B. R. Erdal, INC-DO, MS J-519, Los Alamos National Laboratory, Los Alamos, NM 87545
77. R. Erikson, Pacific Northwest Laboratory, P.O. Box 999, Richland, WA 99352
78. M. Foley, Pacific Northwest Laboratory, P.O. Box 999, Richland, WA 99352
79. J. R. Fowler, Savannah River Laboratory, U.S. DOE, P.O. Box A, Aiken, SC 29801
80. C. J. Geier, Westinghouse Hanford Co., P.O. Box 1970, Richland, WA 99352
81. R. M. Haralick, Boeing Clairmont Egtvedt Prof., Dept. of Electrical Engineering, Director, Intelligent Systems Lab, University of Washington, 402 Electrical Eng. Bldg., FT-10, Seattle, WA 98195
82. J. C. Helton, Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185
83. D. T. Hoxie, United States Geological Survey, Box 25046, Denver Federal Center, Denver, CO 80225
84. R. L. Iman, Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185
85. E. A. Jennrich, Rogers & Associates Engineering Corp., P.O. Box 330, Salt Lake City, UT 84110-0330
86. M. R. Jugan, Research and Waste Management Division, U.S. DOE-ORO, Oak Ridge, TN 37830
87. W. E. Kennedy, Pacific Northwest Laboratory, P.O. Box 999, Richland, WA 99352
88. C. Koplík, The Analytic Sciences Corporation, 1 Jacob Way, Reading, MA 10867
89. P. E. Lamont, U.S. DOE-BWIPO, Richland, WA 99352
90. D. Langmuir, Department of Chemistry and Geochemistry, Colorado School of Mines, Golden, CO 80401
91. W. C. Latting, U.S. DOE - Idaho Operations Office, 785 DOE Place, Idaho Falls, ID 83402
92. B. Lazas, Idaho National Engineering Laboratory, EG&G Idaho, Inc., P.O. Box 1625, Idaho Falls, ID 83415
93. W. L. Lee, R. F. Weston, Inc., 2301 Research Blvd., Rockville, MD 20850
94. J. E. Leiss, 13013 Chestnut Oak Drive, Gaithersburg, MD 20878
95. A. Liebetau, Pacific Northwest Laboratory, P.O. Box 999, Richland, WA 99352
96. S. J. Maheras, Idaho National Engineering Laboratory, EG&G Idaho, Inc., P.O. Box 1625, Idaho Falls, ID 83415

97. M. McKenzie-Carter, Idaho National Engineering Laboratory, EG&G Idaho, Inc., P.O. Box 1625, Idaho Falls, ID 83415
98. N. Moray, Dept. of Mechanical and Industrial Eng., University of Illinois, 1206 West Green Street, Urbana, IL 61801
99. B. Napier, Pacific Northwest Laboratory, P.O. Box 999, Richland, WA 99352
100. J. O. Neff, U.S. Department of Energy, Richland Operations Office, 505 King Ave., Columbus, OH 43201
101. B. L. Nitschke, Idaho National Engineering Laboratory, EG&G Idaho, Inc., P.O. Box 1625, Idaho Falls, ID 83415
102. M. D. Otis, Science Applications International Corp., 101 Park Ave., Idaho Falls, ID 83401
103. M. Piepho, Pacific Northwest Laboratory, P.O. Box 999, Richland, WA 99352
104. T. H. Pigford, Department of Nuclear Engineering, University of California, Berkeley, CA 94720
105. J. Rhoderick, U.S. DOE/OCRWM, 1000 Independence Ave., SW, Washington, DC 20585
106. C. Russomanno, U.S. DOE, Civilian Radioactive Waste Management Program, 1000 Independence Ave., S.W., Washington, DC 20585
107. B. Sager, Pacific Northwest Laboratory, P.O. Box 999, Richland, WA 99352
108. M. W. Shupe, U.S. DOE - Idaho Operations Office, 785 DOE Place, Idaho Falls, ID 83402
109. S. Sneider, Pacific Northwest Laboratory, P.O. Box 999, Richland, WA 99352
110. J. Sonnicksen, Westinghouse Hanford Co., P.O. Box 1970, Richland, WA 99352
111. H. W. Sturm, Savannah River Laboratory, U.S. DOE, P.O. Box A, Aiken, SC 29801
112. J. Sykes, Dept. of Civil Engineering, University of Waterloo, Waterloo, Ontario,, N2L3G1, CANADA
113. M. K. Thompson, U.S. DOE-BWIPO, Richland, WA 99352
114. D. Veneziano, Massachusetts Institute of Technology, Room 1-382, 77 Massachusetts Ave., Cambridge, MA 02139
115. E. L. Wilhite, Savannah River Laboratory, U.S. DOE, P.O. Box A, Aiken, SC 29801
116. M. F. Wheeler, Mathematics Department, University of Houston, 4800 Calhoun, Houston, TX 77204-3476
117. D. Wood, Westinghouse Hanford Co., P.O. Box 1970, Richland, WA 99352
- 118-405. For distribution as shown in TID-4500, Distribution category, UC-512 - Nuclear Waste Management