ARMY RESEARCH LABORATORY

Computational Modeling of Multicomponent Diffusion Using Fortran

by Michael Vincent Pasquariello

ARL-CR-575 **July 2006**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

ARL-CR-575 **July 2006**

Computational Modeling of Multicomponent Diffusion Using Fortran

Michael Vincent Pasquariello
University of Connecticut

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
July 2006	Final	13 April 2003–31 December 2005

4. TITLE AND SUBTITLE
Computational Modeling of Multicomponent Diffusion Using Fortran

5a. CONTRACT NUMBER
DAA17-03-C-0016

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Michael Vincent Pasquariello

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Connecticut

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: AMSRD-ARL-WM-BD
Aberdeen Proving Ground, MD 21005-5066

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
ARL-CR-575

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The objective of this thesis was to develop a Fortran software package, using three modules, in order to extract diffusion data from concentration profiles and to predict future concentration profiles. The first module will be a finite difference code that uses the multicomponent form of Fick's First Law, and the time evolution of concentration will be calculated using the implicit Crank-Nicholson method. The second module will take into account movements of boundaries between regions in the interdiffusion zone, via equations that assume local equilibrium, and take into account that mass must be conserved. The first and second module will be used to predict how measured concentration profiles will change with time. The purpose of the third module will be to extract diffusivity data from measured concentration profiles. This module will use a matrix inversion method to calculate the diffusivities. In conclusion, the original objective of this project was not met to its full completion. Several factors contributed to this shortcoming, but the primary obstacle was the correlation between the software and the input data. While the software ran successfully with many different known solutions, it did not perform well using actual concentration profile data from the U.S. Army Research Laboratory. Most likely, this is due to the limited amount of species data, the accuracy of the data itself, and the spacing between each data point. All efforts should be taken to obtain more accurate, smoother input data which will allow the software to run with fewer obstacles and, in turn, produce cleaner output data. Once the input data is appropriate, the programmer should return to "fine tune" the individual software programs to allow them to work with the new data. This may require using a filtering subroutine in order to accept only worthy data from the input stream.

15. SUBJECT TERMS
gun tube erosion, multicomponent diffusion, diffusion matrix

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Paul Conroy
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL	58	19b. TELEPHONE NUMBER (*Include area code*) 410-278-6114

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures v

Acknowledgments vi

1. Introduction 1

1.1 Background 1

1.2 Problem Statement 1

1.3 Approach 1

2. Literature Survey 1

2.1 Diffusion Equations for Predicting Concentration Profiles 1

2.1.1 Binary Systems....1

2.1.2 Multicomponent Systems With One Fast Diffuser 6

2.2 Finite-Difference Methods for Predicting Concentration Profiles....7

2.2.1 Explicit Forward-Difference Method (Euler Method) 8

2.2.2 Crank-Nicolson Method....9

2.2.3 Thomas Algorithm to Solve a Tridiagonal System of Equations....10

2.3 Inverse Methods Used to Calculate Transport Properties....13

2.3.1 Extracting Thermal Conductivity Values From Temperature Profiles 13

2.3.2 Extracting Diffusion Coefficients From Concentration Profiles....13

3. Fortran Software Development 17

3.1 Concentration Profile Predictor....17

3.2 Diffusivity Extractor....19

4. Software Limitations 20

4.1 Binary_effd....21

4.2 Inverse_rewrite....22

4.3 Ternary_separate 24

5. Conclusions 25

6. References 26

Appendix A. Binary_effd **27**

Appendix B. Inverse_rewrite **31**

Appendix C. Ternary_separate **37**

Distribution List **44**

List of Figures

Figure 1. Transient diffusion with constant diffusivity. 2
Figure 2. Concentration-distance curves for an instantaneous plane source. 3
Figure 3. Concentration-distance curves for the thin-film solution. 4
Figure 4. Finite difference grid for the explicit Euler method. 8
Figure 5. The Crank-Nicolson method stencil. 9
Figure 6. Graph showing the error function solution compared with the Fortran code solution. 18
Figure 7. Graph showing the thin-film solution compared with the Fortran program. 19
Figure 8. Graph showing the trigonometric approximation compared with the Fortran program. 20
Figure 9. Diffusivity matrix plotted for constant diffusivity of 1. 21
Figure 10. Thin film approximation using program. 22
Figure 11. Program output using ARL input data. 23

Acknowledgments

First and foremost, I would like to gratefully acknowledge the enthusiastic supervision of Dr. John E. Morral during this work. It was his guidance and continuous interest in the project that kept me motivated throughout this 4-year saga. I would have been lost without him.

A special thank you to Dr. Harold Brody for helping to spark my interest in materials science during my first year as a University of Connecticut undergraduate. I don't think he will ever know how much influence he had on my future academic decisions.

Thank you to Dr. Pamir Alpay for being an ideal mentor, one I could close the door and talk to about almost anything. He truly is a genuine professor that enjoys sharing his knowledge with eager students.

Thank you to Paul Conroy from the U.S. Army Research Laboratory for his continued support throughout this research project.

I would especially like to extend my heartfelt appreciation to my patient and loving wife, Harmony, for remaining by my side through all 10 years of my college career. It takes an incredible woman to put up with the mind of an engineer, and I will be eternally grateful for it.

Finally, I am forever indebted to my parents, Antonia and Mark, for their understanding, endless patience, and encouragement when it was most required. This thesis would be nothing without their tremendous confidence and belief in my academic abilities. They have created an environment in which following this path seemed so natural. It is to them whom I dedicate this thesis.

1. Introduction

1.1 Background

The U.S. Army Research Laboratory (ARL) in Aberdeen, MD, has expressed a need for a computer software package that can analyze ion-implantation data, extract diffusion coefficients from this data, and predict concentration profiles. Their current implantation data is used to study the effects of carbon diffusion in gun-tube barrels. A comprehensive software package such as this would save a tremendous amount of experimentation time and money for ARL and would therefore be very beneficial.

1.2 Problem Statement

The objective of this thesis was to develop a Fortran software package in order to extract diffusion data from concentration profiles and to predict future concentration profiles. This package needed to be stand-alone, user friendly, and have the ability to interface with ARL's existing computer code.

1.3 Approach

The approach to this problem began with an in-depth literature survey. This allowed the gathering of necessary equations and solution methods to solve the mathematical portion of the thesis. Using this newly acquired knowledge, a Fortran code was written in order to satisfy the requirements of ARL and to perform the necessary functions. This code was tested and compared against existing code and mathematical solutions to validate its effectiveness.

The thesis then covered various experiments to demonstrate the sensitivity of the code. The purpose of this step was to aid the reader in visualizing the adverse effects of altering parameters such as number of data points, time increments, and induced noise.

The thesis concluded with a discussion on the limitations of the software itself and possible recommendations that can be made in the future to improve the software package.

2. Literature Survey

2.1 Diffusion Equations for Predicting Concentration Profiles

2.1.1 Binary Systems

2.1.1.1 Constant Diffusivity. In reality, most cases of diffusion are transient or non steady-state ones. This applies, for example, to those cases in which the interstitial concentration C varies with time which results in a net accumulation or depletion of the diffusing species. In order to

model this situation, it is necessary to use the partial differential equation known as the diffusion equation which is given by (*1*)

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial C}{\partial x} \right). \tag{1}$$

For cases in which D is independent of composition, or where the range of composition is small, equation 1 reduces to (*1*)

$$\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2}, \tag{2}$$

which is known as Fick's second law. Figure 1 shows an example of transient diffusion in a binary system in which the diffusivity is constant with respect to composition.

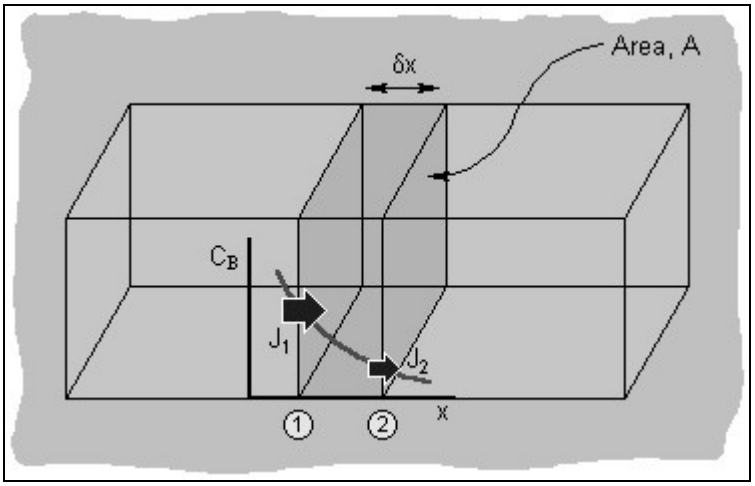

Figure 1. Transient diffusion with constant diffusivity.

If one were to apply Fick's second law to a semi-infinite solid and held the surface concentration constant, a common error function solution would be obtained (*2*). In order to obtain such a solution, the following assumptions can be made:

1. Before diffusion, the diffusing solute atoms in the solid are uniformly distributed with concentration of C_o.
2. The concentration at the surface, x = 0, is a constant value, $C_{s.}$

These boundary conditions can be stated as follows:

For t = 0, $C = C_O$ at $0 \leq x \leq \infty$,

For t > 0, $C = C_s$ at $x = 0$, and $C = C_0$ at $x = \infty$.

If these boundary conditions are applied to equation 2, the following solution is obtained:

$$\frac{C_x - C_o}{C_s - C_o} = 1 - erf\left(\frac{x}{2\sqrt{Dt}}\right), \tag{3}$$

where C_x represents the concentration at depth x after time t. The expression erf(x/2√Dt) is the Gaussian error function, values of which are given in mathematical tables for various x/2√Dt values. Equation 3 demonstrates the relationship between concentration, position, and time, namely, that C_x, being a function of the dimensionless parameter x/√Dt, may be determined at any time and position if the parameters C_o, C_s, and D are known.

Another solution to Fick's second law is known as the thin-film solution. Taking an infinite plane as the geometry, the total amount of substance M diffusing in the cylinder and unit cross section is given by (*1*)

$$M = \int_{-\infty}^{\infty} C dx\,. \tag{4}$$

After differentiating equation 4 and applying the appropriate derivation, equation 5 is given as (*1*)

$$C = \frac{M}{2\sqrt{\pi Dt}} \exp\left(\frac{-x^2}{4Dt}\right). \tag{5}$$

Therefore, this is the solution which describes the spreading by diffusion of an amount of substance M deposited at time t = 0 in the plane x = 0. Figure 2 shows typical distributions at six successive times.

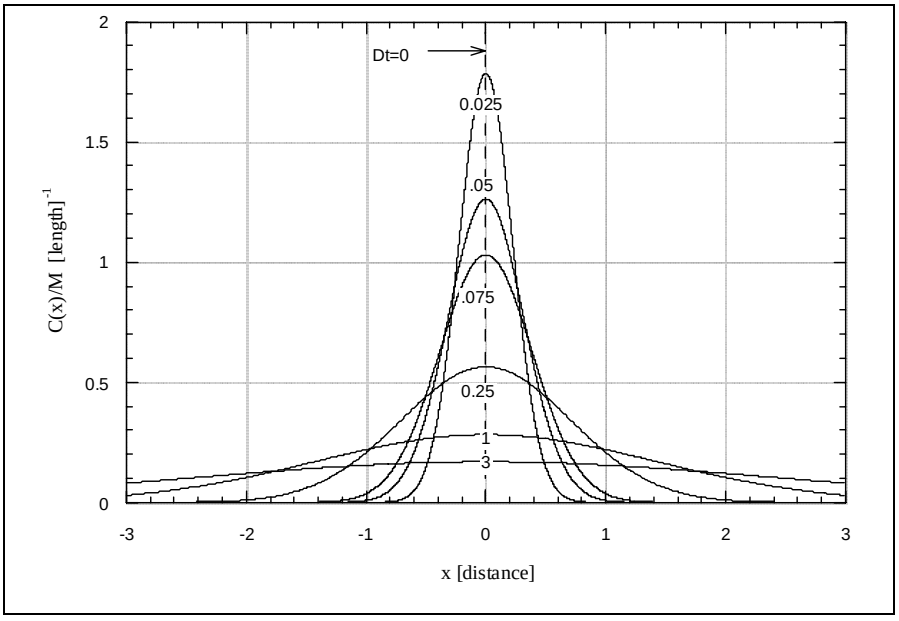

Figure 2. Concentration-distance curves for an instantaneous plane source.

For the thin-film solution, we can consider the solution for negative *x* to be reflected in the plane x = 0 and superimposed on the original distribution in the region x > 0. Since the original solution was symmetrical about x = 0 the concentration distribution for the semi-infinite plane is given by

$$C = \frac{M}{\sqrt{\pi Dt}} \exp\left(\frac{-x^2}{4Dt}\right). \tag{6}$$

A typical concentration distribution for the thin film solution is demonstrated in figure 3.

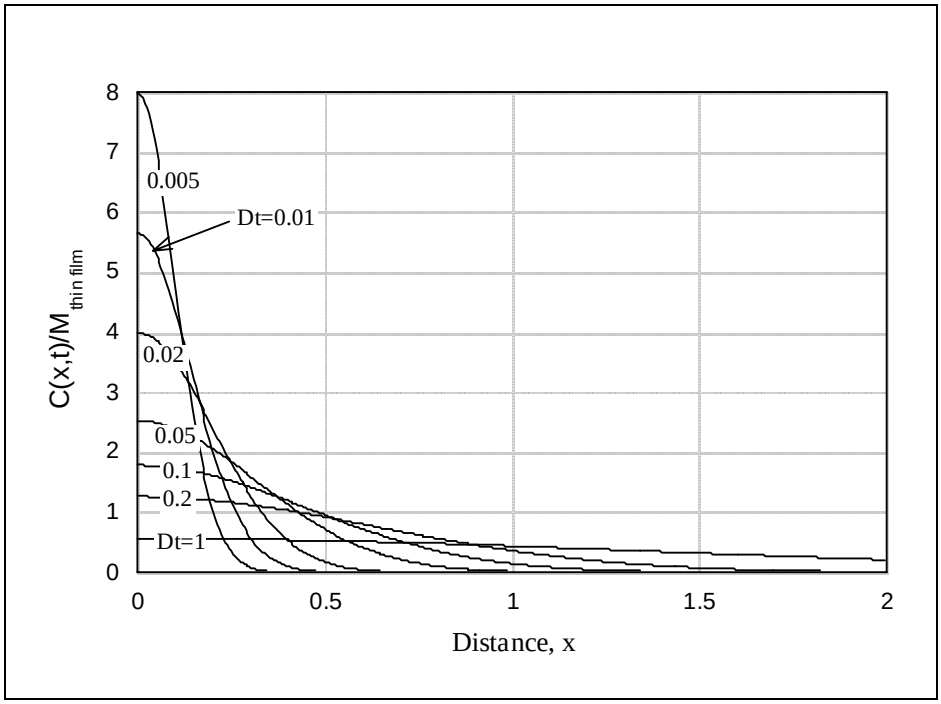

Figure 3. Concentration-distance curves for the thin-film solution.

A third solution to Fick's second law is based on a trigonometric solution. In this solution, it is assumed that the time and spatial variables are separable, that is (*1*)

$$C_B = \xi(x) \cdot \tau(t), \tag{7}$$

where ξ and τ are functions, as yet unknown, of *x* and *t* respectively. Substituting for C_B in Fick's second law gives (*1*)

$$\xi \frac{d\tau}{dt} = D_B \tau \frac{d^2\xi}{dx^2}, \tag{8}$$

and

$$\frac{1}{D_B \tau} \frac{d\tau}{dt} = \frac{1}{\xi} \frac{d^2\xi}{dx^2}, \tag{9}$$

where we now have only total differentials of our unknown functions. Since one side of the equation depends only on *t* and the other only on *x*, the equation can only hold for all *x* and *t* if both are equal to some constant, say $-k^2$ (*1*).

$$\frac{d\tau}{dt} = -k^2 D_B \tau \,. \tag{10}$$

$$\frac{d\xi}{dx} = -k^2 \xi \,. \tag{11}$$

These differential equations can now be solved to give the functions ξ and τ (*1*).

$$\tau = \exp(-k^2 D_B t) \,. \tag{12}$$

$$\xi = A_1 \sin(kx) + A_2 \cos(kx) \,. \tag{13}$$

The time-dependent concentration function C_B is therefore a sinusoidal composition fluctuation which decays exponentially with time. This is seen more clearly by taking $C_B = C_0$ at x = 0 at all times, and substituting for the wavelength of the sinusoidal variation λ:

$$C_B(x,t) = C_0 + \Delta C \sin\left(\frac{2\pi x}{\lambda}\right)\exp\left(-\frac{4\pi^2 D_B}{\lambda^2}t\right)$$

$$= C_0 + \Delta C \sin\left(\frac{2\pi x}{\lambda}\right)\exp\left(-\frac{t}{t_\Gamma}\right), \tag{14}$$

where

$$t_\Gamma = \frac{\lambda^2}{4\pi^2 D_B} \tag{15}$$

is the relaxation time, which is the time taken for a sinusoidal variation of wavelength λ to drop to 37% of its original amplitude.

2.1.1.2 Variable Diffusivity. When the diffusion coefficient *D* is a function of concentration *C*, the equation for one-dimensional (1-D) diffusion is (*3*)

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial C}{\partial x}\right). \tag{16}$$

Differentiation of equation 16 yields (*3*)

$$\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2} + \frac{\partial D}{\partial x}\frac{\partial C}{\partial x} \,. \tag{17}$$

Second-order-correct approximations to the partial derivations in equation 17 are (*3*):

$$\frac{\partial C^{n+1/2}}{\partial t_i} \cong \frac{C_i^{n+1} - C_i^n}{\Delta t}. \tag{18}$$

$$\frac{\partial^2 C^{n+1/2}}{\partial x_i^2} \cong \frac{1}{2}\left[\frac{C_{i+1}^{n+1} - 2C_i^{n+1} + C_{i-1}^{n+1}}{(\Delta x)^2} + \frac{C_{i+1}^{n} - 2C_i^{n} + C_{i-1}^{n}}{(\Delta x)^2}\right]. \tag{19}$$

$$\frac{\partial D^n}{\partial x_i} \cong \frac{D_{i+1}^n - D_{i-1}^n}{2\Delta x}. \tag{20}$$

$$\frac{\partial C^{n+1/2}}{\partial x_i} \cong \frac{1}{2}\left[\frac{C_{i+1}^{n+1} - C_{i-1}^{n+1}}{2\Delta x} + \frac{C_{i+1}^{n} - C_{i-1}^{n}}{2\Delta x}\right]. \tag{21}$$

The superscripts refer to the time dimension and subscripts denote the space dimension. Note that $\frac{\partial D}{\partial x}$ is evaluated at *n* rather than *n+1/2*, because the composition-dependent coefficient *D* cannot be calculated at the next time step *n+1* before the concentrations at *n+1* have been evaluated. Substituting the approximations, equations 18–21 into equation 17 yields (*3*)

$$C_{i-1}^{n+1}\left[-4D_i^n + D_{i+1}^n - D_{i-1}^n\right] + C_i^{n+1}\left[\frac{8(\Delta x)^2}{\Delta t} + 8D_i^n\right]$$

$$+ C_{i+1}^{n+1}\left[-4D_i^n - D_{i+1}^n - D_{i-1}^n\right]$$

$$= C_{i-1}^{n}\left[4D_i^n - D_{i+1}^n + D_{i-1}^n\right] + C_i^{n}\left[\frac{8(\Delta x)^2}{\Delta t} - 8D_i^n\right]$$

$$+ C_{i+1}^{n}\left[4D_i^n + D_{i+1}^n - D_{i-1}^n\right], \tag{22}$$

which has been arranged so that all the concentrations at the current time step (n) are on the right and the concentrations to be computed at the next time step (n+1) are on the left.

2.1.2 Multicomponent Systems With One Fast Diffuser

The basis for modeling multicomponent systems with one fast diffuser comes from the following equation:

$$\frac{\partial c_1^{total}}{\partial t} = \frac{\partial}{\partial x}(-J)^{total} \tag{23}$$

in which *x* represents distance and J^{total} is the total flux (*4*). Assuming the precipitate volume fraction is negligible, the multicomponent flux can be written as

$$J_1 = -\sum_{j=1}^{n-1} D_{1j} \frac{\partial c_j}{\partial x}, \tag{24}$$

in which component 1 is a fast diffuser (i.e., an interstitial atom) and components 2 through n-1 refer to slower moving alloying elements (i.e., substitutional atoms) (*5*).

Equation 24 can be rewritten in terms of an effective diffusivity described by the equation:

$$J_1 = -D_1^{eff} \frac{\partial c_1^m}{\partial x}. \tag{25}$$

Joining equations 24 and 25 yields (*6*)

$$D_1^{eff} = \frac{\sum_{j=1}^{n-1} D_{1j} \frac{\partial c_j^m}{\partial x}}{\frac{\partial c_1^m}{\partial x}}. \tag{26}$$

By assuming local equilibrium and no long-range diffusion by substitutional atoms, equation 26 can be simplified to

$$D_1^{eff} = D_{11} + \sum_{j=2}^{n-1} D_{1j} \left. \frac{\partial c_j^m}{\partial c_1^m} \right|_{c_k}. \tag{27}$$

Equation 27 represents only one independent concentration variable with only one degree of freedom.

Combining equations 23 and 25 and then differentiating c_1^{tot} gives

$$\frac{\partial c_1^{tot}}{\partial t} = \frac{\partial}{\partial x} D_1^{eff} \frac{\partial c_1^m}{\partial c_1^{tot}} \frac{\partial c_1^{tot}}{\partial x}. \tag{28}$$

Equation 28 is the one used to predict concentration profiles for multicomponent systems with one fast diffuser.

2.2 Finite-Difference Methods for Predicting Concentration Profiles

The objective of a finite-difference method for solving an ordinary differential equation (ODE) is to transform a calculus problem into an algebra problem by:

1. Discretizing the continuous physical domain into a discrete finite difference grid,
2. Approximating the exact derivatives in the initial-value ODE by algebraic finite difference approximations (FDAs),

3. Substituting the FDAs into the ODE to obtain an algebraic finite difference equation (FDE), and
4. Solving the resulting algebraic FDE.

2.2.1 Explicit Forward-Difference Method (Euler Method)

Consider the general nonlinear first-order ODE (*7*):

$$\bar{y}' = f\left(t, \bar{y}\right)\bar{y}(t_0) = \bar{y}_0 . \tag{29}$$

Choose point *n* as the base point and develop a finite difference approximation of equation 29 at that point. The finite-difference grid is illustrated in figure 4, where the *x* symbol denotes the base point for the finite difference approximation of equation 29.

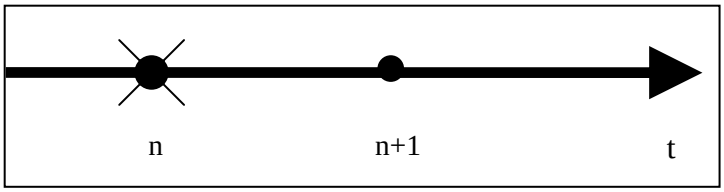

Figure 4. Finite difference grid for the explicit Euler method.

The first-order forward-difference, finite-difference approximation of $\bar{y}'$ is given by (*7*):

$$\bar{y}'\big|_n = \frac{\bar{y}_{n+1} - \bar{y}_n}{\Delta t} - \frac{1}{2}\bar{y}''(\tau_n)\Delta t . \tag{30}$$

Substituting equation 30 into equation 29 and evaluating $f\left(t, \bar{y}\right)$ at point *n* yields (*7*)

$$\frac{\bar{y}_{n+1} - \bar{y}_n}{\Delta t} - \frac{1}{2}\bar{y}''(\tau_n)\Delta t = f\left(t_n, \bar{y}_n\right) = \bar{f}_n . \tag{31}$$

Solving equation 31 for $\bar{y}_{n+1}$ gives (*7*)

$$\bar{y}_{n+1} = \bar{y}_n + \Delta t\,\bar{f}_n + \frac{1}{2}\bar{y}''(\tau_n)\Delta t^2 = \bar{y}_n + \Delta t\,\bar{f}_n + 0\left(\Delta t^2\right). \tag{32}$$

Truncating the remainder term, which is $0\left(\Delta t^2\right)$, and solving for y_{n+1} yields the explicit Euler finite difference equation (FDE) (*7*):

$$y_{n+1} = y_n + \Delta t f_n \; 0\left(\Delta t^2\right), \tag{33}$$

where the $0\left(\Delta t^2\right)$ term is included as a reminder of the order of the local truncation error. Several features of equation 33:

1. The FDE is explicit, since f_n does not depend on y_{n+1}.

2. The FDE requires only one known point. Hence, it is a single-point method.
3. The FDE requires only one derivative function evaluation (i.e., f(t,y)) per step.
4. The error in calculating y_{n+1} for a single step, the local truncation error, is $0\left(\Delta t^2\right)$.
5. The global (i.e., total) error accumulated after *N* steps is $0\left(\Delta t^2\right)$.

The explicit Euler method only has first-order accuracy and it is very unstable, therefore, it is impractical to use.

2.2.2 Crank-Nicolson Method

The Crank Nicolson (*1*) method is a more widely used finite difference method for solving partial differential equations and is set up using the following grid (figure 5).

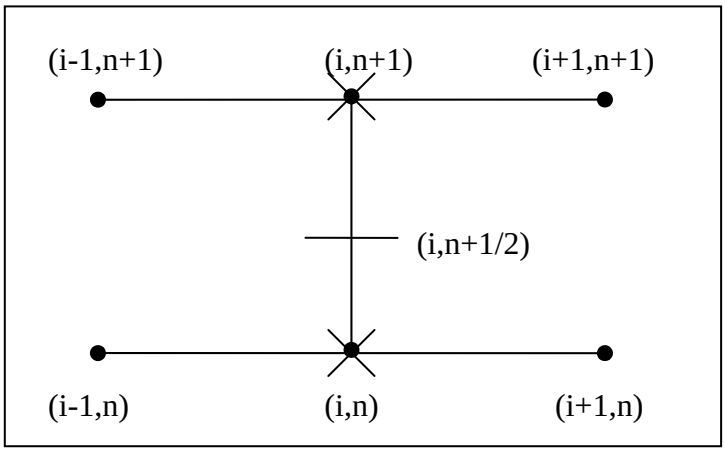

Figure 5. The Crank-Nicolson method stencil.

Crank and Nicolson in 1947 proposed approximating the partial derivative $\overline{f}_t$ at grid point (i,*n*+1/2) by the second-order centered-time approximation obtained by combining Taylor series for $\overline{f}_i^{n+1}$ and $\overline{f}_k^{n}$. Thus, (*7*)

$$\overline{f}_i^{n+1} = \overline{f}_i^{n+1/2} + \overline{f}_t\Big|_i^{n+1/2}\left(\frac{\Delta t}{2}\right) + \frac{1}{2}\overline{f}_{tt}\Big|_i^{n+1/2}\left(\frac{\Delta t}{2}\right)^2 + \frac{1}{6}\overline{f}_{ttt}\Big|_i^{n+1/2}\left(\frac{\Delta t}{2}\right)^3 + \ldots \tag{34}$$

$$\overline{f}_i^{n} = \overline{f}_i^{n+1/2} - \overline{f}_t\Big|_i^{n+1/2}\left(\frac{\Delta t}{2}\right) + \frac{1}{2}\overline{f}_{tt}\Big|_i^{n+1/2}\left(\frac{\Delta t}{2}\right)^2 - \frac{1}{6}\overline{f}_{ttt}\Big|_i^{n+1/2}\left(\frac{\Delta t}{2}\right)^3 + \ldots \tag{35}$$

Subtracting these two equations and solving for $\overline{f}_t\Big|_i^{n+1/2}$ gives

$$\overline{f}_t\Big|_i^{n+1/2} = \frac{\overline{f}_i^{n+1} - \overline{f}_i^{n}}{\Delta t} - \frac{1}{24}\overline{f}_{ttt}(\tau)\Delta t^2, \tag{36}$$

where $t^n \le \tau \le t^{n+1}$ [7]. Truncating the remainder term in equation 36 yields the second-order centered-time approximation of $\overline{f}_t$ (*7*):

$$f_t\Big|_i^{n+1/2} = \frac{f_i^{n+1} - f_i^n}{\Delta t}. \tag{37}$$

The partial derivative $\overline{f}_{xx}$ at grid point (*i*,*n*+1/2) is approximated by (*7*)

$$\overline{f}_{xx}\Big|_i^{n+1/2} = \frac{1}{2}\left(f_{xx}\Big|_i^{n+1} + \overline{f}_{xx}\Big|_i^{n} \right). \tag{38}$$

The order of the FDE obtained using equations 37 and 38 is expected to be $0(\Delta t)^2 + 0(\Delta x)^2$, but that must be proven from the MDE. The partial derivative $\overline{f}_{xx}$ at time levels *n* and *n+1* are approximated by the second-order centered-difference approximation

$$f_{xx}\Big|_i^n = \frac{f_{i+1}^n - 2f_i^n + f_{i-1}^n}{\Delta x^2}, \tag{39}$$

applied at time levels *n* and *n+1*, respectively (*7*). The resulting finite-difference approximation of the 1-D diffusion equation is (*7*)

$$\frac{f_i^{n+1} - f_i^n}{\Delta t} = \alpha \frac{1}{2}\left(\frac{f_{i+1}^{n+1} - 2f_i^{n+1} + f_{i-1}^{n+1}}{\Delta x^2} + \frac{f_{i+1}^{n} - 2f_i^{n} + f_{i-1}^{n}}{\Delta x^2} \right). \tag{40}$$

Rearranging equation 40 yields the Crank-Nicolson finite-difference equation:

$$-d f_{i-1}^{n+1} + 2(1+d) f_i^{n+1} - d f_{i+1}^{n+1} = d f_{i-1}^{n} + 2(1-d) f_i^{n} + d f_{i+1}^{n}, \tag{41}$$

where $d = \alpha \frac{\Delta t}{\Delta x^2}$ is the diffusion number (*7*).

The Crank-Nicolson method is unconditionally stable and accurate on a second order level. The solution at a given time level can be reached with much less computational effort by taking larger time steps. The time step is limited only by accuracy requirements.

2.2.3 Thomas Algorithm to Solve a Tridiagonal System of Equations

When a large system of linear algebraic equations has a special pattern, such as a tridiagonal pattern as in the Crank-Nicolson equation, it is usually worthwhile to develop special methods for that unique pattern. These methods are generally very efficient in computer time and storage. One algorithm that deserves special attention is the algorithm for tridiagonal matrices, often referred to as the Thomas algorithm.

To derive the Thomas algorithm, the Gauss elimination procedure is applied to a tridiagonal matrix *T*, modifying the procedure to eliminate all unnecessary computations involving zeros. Consider the matrix equation:

$$\mathrm{Tx} = \mathrm{b}, \tag{42}$$

where *T* is a tridiagonal matrix (*7*). Thus, (*7*)

$$T = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & ... & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & ... & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & ... & 0 & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & ... & 0 & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & 0 & ... & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & 0 & 0 & 0 & ... & 0 & a_{n,n-1} & a_{n,n} \end{bmatrix}. \tag{43}$$

Since all the elements of column 1 below row 2 are already zero, the only element to be eliminated in row 2 is a_{21}. Thus, replace row 2 by $R_2 - \left(\frac{a_{21}}{a_{11}}\right)R_1$. Row 2 becomes (*7*)

$$\begin{bmatrix} 0 & a_{22} - \left(\frac{a_{21}}{a_{11}}\right)a_{12} & a_{23} & 0 & 0 & ... & 0 & 0 & 0 \end{bmatrix}. \tag{44}$$

Similarly, only a_{32} in column 2 must be eliminated from row 3, only a_{43} in column 3 must be eliminated from row 4, etc. The eliminated element itself does not need to be calculated. In fact, storing the elimination multipliers, em = (a_{21}/a_{11}), etc., in place of the eliminated elements allows this procedure to be used as an LU factorization method. Only the diagonal element in each row is affected by the elimination. Elimination in rows 2 to *n* is accomplished as follows (*7*):

$$a_{i,i} = a_{i,i} - (a_{i,i-1} / a_{i-1,i-1}) \quad a_{i-1,i} \ (\mathrm{i} = 2,\ldots,\mathrm{n}) \tag{45}$$

Thus, the elimination step involves only 2n multiplicative operations to place *T* in upper triangular form.

The elements of the *b* vector are also affected by the elimination process. The first element b_1 is unchanged. The second element b_2 becomes (*7*)

$$b_2 = b_2 - \left(\frac{a_{21}}{a_{11}}\right)b_1. \tag{46}$$

Subsequent elements of the *b* vector are changes in a similar manner. Processing the *b* vector requires only one multiplicative operation, since the elimination multiplier, $em = \left(\frac{a_{21}}{a_{11}} \right)$, is already calculated. Thus, the total process of elimination, including the operation on the *b* vector, requires only *3n* multiplicative operations.

The *n x n* tridiagonal matrix *T* can be stored as an *n x 3* matrix *A'* since there is no need to store the zeros. The first column of matrix *A'*, elements $a'_{i,1}$, corresponds to the sub-diagonal of matrix *T*, elements $a_{i,i-1}$. The second column of matrix *A'*, elements $a'_{i,2}$, corresponds to the diagonal elements of matrix *T*, elements $a_{i,i}$. The third column of matrix *A'*, elements $a'_{i,3}$, corresponds to the super-diagonal of matrix *T*, elements $a_{i,i+1}$. The elements $a'_{1,1}$ and $a'_{n,3}$ do not exist. Thus, (*7*)

$$A' = \begin{bmatrix} - & a'_{1,2} & a'_{1,3} \\ a'_{2,1} & a'_{2,2} & a'_{2,3} \\ a'_{3,1} & a'_{3,2} & a'_{3,3} \\ \cdots & \cdots & \cdots \\ a'_{n-1,1} & a'_{n-1,2} & a'_{n-1,3} \\ a'_{n,1} & a'_{n,2} & - \end{bmatrix}. \tag{47}$$

When the elements of column 1 of matrix *A'* are eliminated, that is, the elements $a'_{i,1}$, the elements of column 2 of matrix *A'* become (*7*)

$$a'_{1,2} = a'_{1,2} \tag{48}$$

$$a'_{i,2} = a'_{i,2} - \left(\frac{a'_{i,1}}{a'_{i-1,2}} \right) a'_{i-1,3} \qquad (i = 2, 3, \ldots, n) \tag{49}$$

The *b* vector is modified as follows: (*7*)

$$b_1 = b_1 \tag{50}$$

$$b_i = b_i - \left(\frac{a'_{i,1}}{a'_{i-1,2}} \right) b_{i-1} \qquad (i = 2, 3, \ldots, n) \tag{51}$$

After $a'_{i,2}$ *(i = 2, 3,…,n)* and *b* are evaluated, the back substitution step is as follows: (*7*)

$$x_n = \frac{b_n}{a'_{n,2}}. \tag{52}$$

$$x_i = \frac{\left(b_i - a'_{i,3} x_{i+1} \right)}{a'_{i,2}}. \tag{53}$$

Pivoting destroys the tridiagonality of the system of linear algebraic equations, and thus cannot be used with the Thomas algorithm. Most large tridiagonal systems that represent real physical problems are diagonally dominant, so pivoting is not necessary. The Thomas algorithm, in a format suitable for programming for a computer, is summarized as follows:

1. Store the *n x n* tridiagonal matrix *T* in the *n x 3* matrix *A'*. The right-side vector *b* is an *n x 1* column vector.
2. Compute the $a'_{i,2}$ terms from equations 48 and 49. Store the elimination multipliers, em= $a'_{i,1}/a'_{i-1,2}$, in place of $a'_{i,1}$.
3. Compute the b_i terms from equations 50 and 51.
4. Solve for x_i by back substitution using equations 52 and 53.

2.3 Inverse Methods Used to Calculate Transport Properties

2.3.1 Extracting Thermal Conductivity Values From Temperature Profiles

Inverse determination of the thermal conductivity from measured temperature profiles has been the topic of research by many investigators (*3*). Most of these studies assume that the thermal conductivity is only a function of the spatial coordinate. However, thermal conductivities are temperature-dependent quantities in most practical engineering applications. Yeung developed a second-order finite-difference procedure for the inverse determination of the thermal conductivity in a one-dimensional heat conduction domain. In this case, the thermal conductivity of the material is reconstructed by using the available temperature data at discrete grid points. The numerical procedure is validated by comparing it to known examples. It is proven that, using this technique, *a priori* knowledge of the functional form for thermal conductivity is not required.

2.3.2 Extracting Diffusion Coefficients From Concentration Profiles

Since the governing equations for heat conduction and diffusion are similar, it is only natural to use the same procedure to investigate the diffusion coefficient in a concentration profile.

As stated earlier, in a 1-D formulation with the diffusing substance moving in the direction normal to a sheet of thickness 2a, the diffusion equation can be written as

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial C}{\partial x} \right) (0<x<a,\ t>0), \tag{54}$$

where *C* is the concentration of the diffusing substance, *t* is the time, *D* is the diffusion coefficient, and *x* is the distance coordinate measured from the center of the sheet (*3*).

Let the initial condition be (*3*)

$$C = C_0 (0<x<a,\ t = 0), \tag{55}$$

where C_0 is a constant concentration in the medium, and let the boundary conditions be

$$\frac{\partial C}{\partial x} = 0 \qquad (x = 0,\ t \geq 0) \tag{56}$$

$$D\frac{\partial C}{\partial x} = S\left(C_e - C\right) (x \approx a,\ t>0), \tag{57}$$

where *S* is the surface emission coefficient and C_e is the equilibrium concentration (*3*).

The first step in the inverse method is to present a finite-difference procedure for the calculation of the diffusion coefficient at discrete grid points. Let half of the medium thickness, *a*, be discretized with mesh width Δx in distance (thickness direction) and Δt in the time direction with grid points $x_j = j \cdot \Delta x$ (where j = 0, 1,…,n) and $t_i = i \cdot \Delta t$ (where i = 0, 1, 2…). The present procedure will assume that C(x,t) is known at grid points (x_j, t_i). Equation 54 can then be discretized as follows:

1. At the surface grid point with j = 0 and i > 0:

Applying forward difference to the time derivative of equation 54, we have (*3*)

$$\left(\frac{\partial C}{\partial t}\right)_0^i = \frac{C_0^{i+1} - C_0^i}{\Delta t}. \tag{58}$$

Applying the central difference to the distance derivative, we obtain

$$\left[\frac{\partial}{\partial x}\left(D\frac{\partial C}{\partial x}\right)\right]_0^i = \frac{\left(\frac{D_1^i + D_0^i}{2} \cdot \frac{C_1^i - C_0^i}{\Delta x} - D_0\alpha\frac{C_1^i - C_0^i}{\Delta x}\right)}{\frac{\Delta x}{2}}, \tag{59}$$

where the following has been set in equation 57:

$$\left(D\frac{\partial C}{\partial x}\right)_0^i D_0\alpha\frac{C_1^i - C_0^i}{\Delta x}, \tag{60}$$

by introducing an appropriate constant *a* to compensate the use of forward difference in the equation, which involves different errors than central difference (*3*). This also permits the avoidance of using the unknown surface emission coefficient *S*.

Equating equations 58 and 59 gives (*3*)

$$\frac{(\Delta x)^2}{\Delta t}\left(C_0^{i+1} - C_0^i\right) = D_0^i\left(2\alpha - 1\right)\left(C_0^i - C_1^i\right) + D_1^i\left(C_1^i - C_0^i\right). \tag{61}$$

2. At an internal grid point with 0 < j < n and i > 0:

Here we have (*3*)

$$\left(\frac{\partial C}{\partial t}\right)_j^i = \frac{C_j^{i+1} - C_j^i}{\Delta t} \tag{62}$$

and

$$\left[\frac{\partial}{\partial x}\left(D\frac{\partial C}{\partial x}\right)\right]_j^i = \frac{\left(\frac{D_{j+1}^i + D_j^i}{2}\cdot\frac{C_{j+1}^i - C_j^i}{\Delta x} - \frac{D_j^i + D_{j-1}^i}{2}\frac{C_j^i - C_{j-1}^i}{\Delta x}\right)}{\Delta x}. \tag{63}$$

Equating equations 62 and 63 yields (*3*)

$$\frac{2(\Delta x)^2}{\Delta t}\left(C_j^{i+1} - C_j^i\right) = D_{j-1}^i\left(C_{j-1}^i - C_j^i\right) + D_j^i\left(C_{j+1}^i - 2C_j^i + C_{j-1}^i\right)$$

$$+ D_{j+1}^i\left(C_{j+1}^i - C_j^i\right). \tag{64}$$

3. At the center grid point with j = n and i ≥ 0:

Due to symmetry, we can set $C_{j-1}^i = C_{j+1}^i$, $D_{j-1}^i = D_{j+1}^i$, and j = n in equation 64 to obtain (*3*)

$$\frac{(\Delta x)^2}{\Delta t}\left(C_n^{i+1} - C_n^i\right) = D_{n-1}^i\left(C_{n-1}^i - C_n^i\right) + D_n^i\left(C_{n-1}^i - C_n^i\right). \tag{65}$$

The next step in the inverse method occurs if $C\left(x,\bar{t}\right)$ and $C\left(x,\bar{t}+\Delta t\right)$ are known at evenly spaced grid points where $\bar{t}$ is the specified time and Δt is the time increment, and we are interested in finding the diffusion coefficient values at the grid points. From equations 63, 64, and 65, we can create the following system of linear equations:

$$\mathrm{Ad} = \mathrm{b}, \tag{66}$$

where *A* is an (n+1) × (n+1) matrix and *d* and *b* are (n+1) vectors (*3*). *A*, *d*, and *b* are subscripted from 0 to *n* as shown by the following:

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & & & & & \\ a_{1,0} & a_{1,1} & a_{1,2} & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ & & & & & a_{n,n-1} & a_{n,n} \end{bmatrix}.$$

$$d = \begin{bmatrix} D_0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ D_n \end{bmatrix} \quad b = \begin{bmatrix} b_0 \\ b_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ b_n \end{bmatrix}. \tag{67}$$

The elements of *d* are the unknown diffusion coefficient values at the grid points, and the elements of *A* and *b* are expressed as follows:

1. At the surface grid point with x = x_0 and t = $\bar{t}$: (*3*)

$$a_{0,0} = (2\alpha - 1)\left[C\left(x_0, \bar{t}\right) - C\left(x_1, \bar{t}\right)\right]. \tag{68}$$

$$a_{0,1} = C\left(x_1, \bar{t}\right) - C\left(x_0, \bar{t}\right). \tag{69}$$

$$b_0 = \frac{(\Delta x)^2}{\Delta t}\left[C\left(x_0, \bar{t} + \Delta t\right) - C\left(x_0, \bar{t}\right)\right]. \tag{70}$$

2. At an internal grid point with x = x_j (0<j<n) and t = $\bar{t}$: (*3*)

$$a_{j,j-1} = C\left(x_{j-1}, \bar{t}\right) - C\left(x_j, \bar{t}\right). \tag{71}$$

$$a_{j,j} = C\left(x_{j+1}, \bar{t}\right) - 2C\left(x_j, \bar{t}\right) + C\left(x_{j-1}, \bar{t}\right). \tag{72}$$

$$a_{j,j+1} = C\left(x_{j+1}, \bar{t}\right) - C\left(x_j, \bar{t}\right). \tag{73}$$

$$b_j = 2\frac{(\Delta x)^2}{\Delta t}\left[C\left(x_j, \bar{t} + \Delta t\right) - C\left(x_j, \bar{t}\right)\right]. \tag{74}$$

3. At the center grid point with x = x_n and t = $\bar{t}$: [3]

$$a_{n,n-1} = C\left(x_{n-1}, \bar{t}\right) - C\left(x_n, \bar{t}\right). \tag{75}$$

$$a_{n,n} = C\left(x_{n-1}, \bar{t}\right) - C\left(x_n, \bar{t}\right). \tag{76}$$

$$b_n = \frac{(\Delta x)^2}{\Delta t}\left[C\left(x_n, \bar{t} + \Delta t\right) - C\left(x_n, \bar{t}\right)\right]. \tag{77}$$

This system consists of a tridiagonal system of linear algebraic equations. The solution vector *d* is the diffusion coefficient vector. This system can be solved using the Thomas algorithm as previously mentioned.

3. Fortran Software Development

3.1 Concentration Profile Predictor

Constant Diffusivity

This program was written to predict future concentration profiles from an initial concentration profile and constant diffusivity value. The program begins by asking the user for the initial concentration profile file in .txt format. After storing this array, the program requests the constant diffusivity value from the user. The final request from the software is the time at which the user would like the concentration profile to be predicted. The software's output is both on screen and in a .txt file located in the same location as the initial concentration profile. The user can then easily import this file into Excel to see the new concentration profile. The user can also edit the position and time increments within the code itself to tailor the output to their liking.

As discussed in section 2.1.1.1, one solution to Fick's second law is the error function solution. One way to test the validity of this program is to compare it against this known solution, demonstrated by

$$\frac{C_x - C_o}{C_s - C_o} = 1 - erf\left(\frac{x}{2\sqrt{Dt}}\right). \tag{78}$$

This is accomplished by using an initial concentration profile based on equation 78 and setting a constant diffusivity. This diffusivity value is used both in equation 78 and in the program itself. Equal times were chosen as well. The result of this comparison is shown in figure 6.

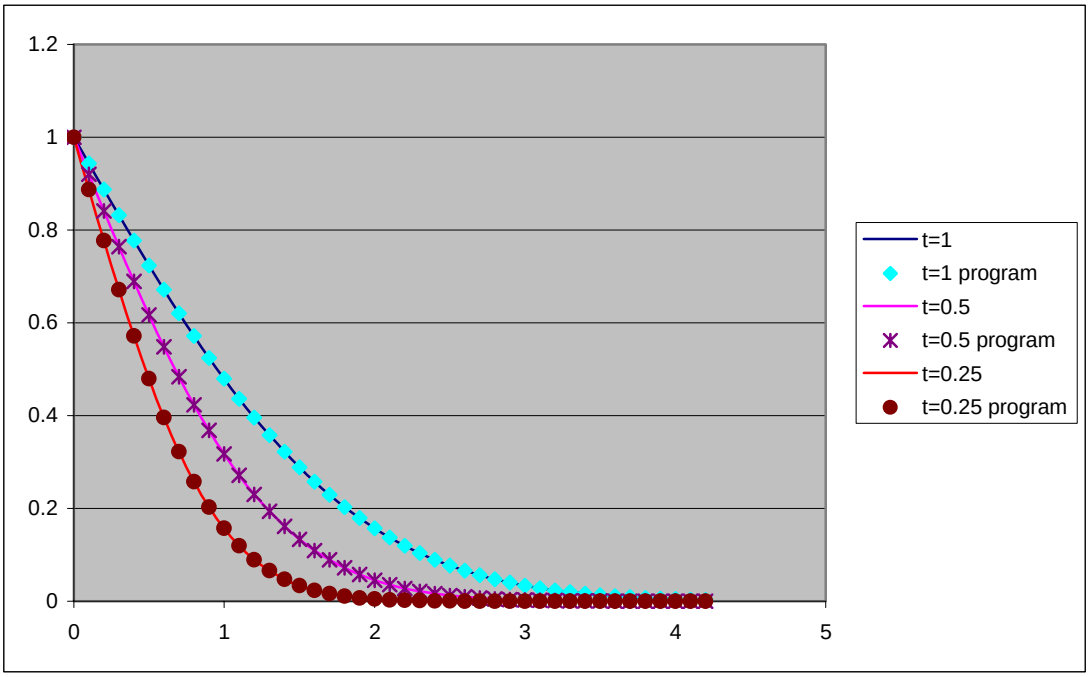

Figure 6. Graph showing the error function solution compared with the Fortran code solution.

One can see that the results match to four significant figures, thereby verifying the correct operation of the Fortran program.

Another solution known as the thin-film solution was discussed in section 2.1.1.1 as well. This research compares this solution to the Fortran program as well to further validate its functional use. The thin-film solution was given in equation 6, which is shown again here:

$$C = \frac{M}{\sqrt{\pi D t}} \exp\left(\frac{-x^2}{4Dt}\right). \tag{79}$$

In order to compare the program to the thin-film solution, an initial concentration profile based on equation 79 was used as well as a predetermined constant diffusivity. This diffusivity value was used both in equation 79 and in the program itself. Equal time increments were chosen as well. The result of this comparison is shown in figure 7.

One can see that the results match to four significant figures, thereby verifying the correct operation of the Fortran program.

A third solution of Fick's law is based on a trigonometric solution and is mentioned in section 2.1.1.1. This solution is based on the following:

$$C_B(x,t) = C_0 + \Delta C \sin\left(\frac{2\pi x}{\lambda}\right) \exp\left(-\frac{t}{t_\Gamma}\right). \tag{80}$$

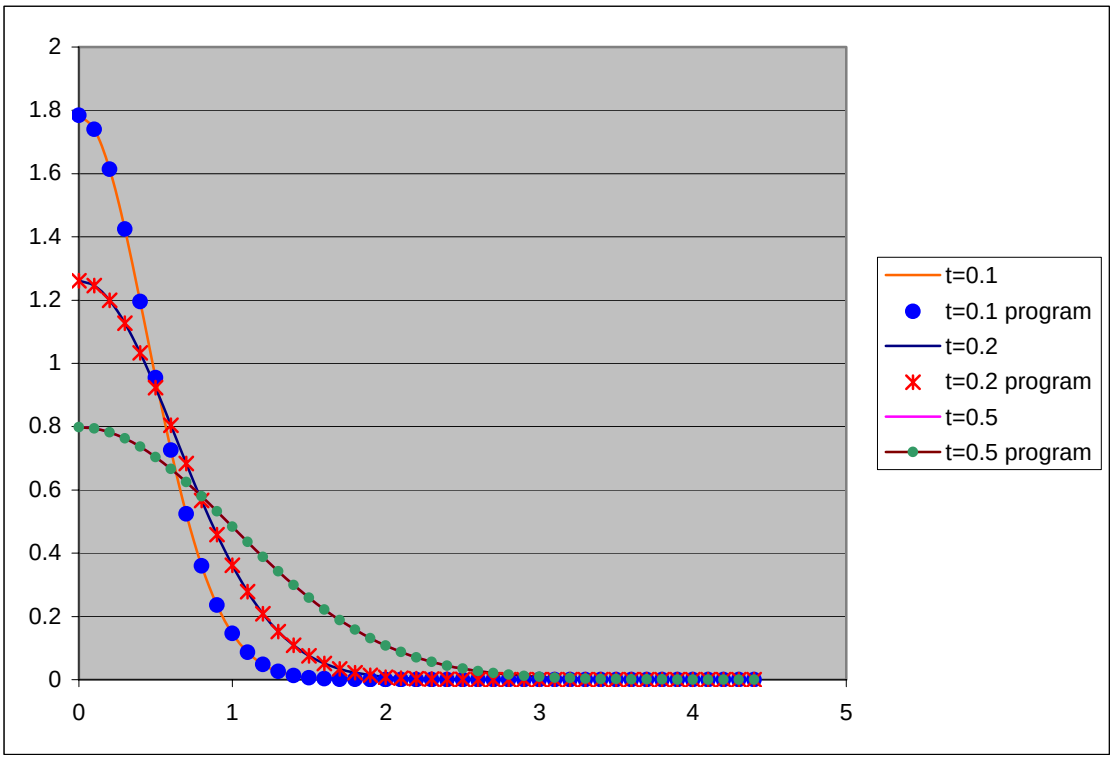

Figure 7. Graph showing the thin-film solution compared with the Fortran program.

In order to compare the program to the trigonometric solution, an initial concentration profile based on equation 80 was used as well as a predetermined constant diffusivity. This diffusivity value was used both in equation 80 and in the program itself. Equal time increments were chosen as well. The result of this comparison is shown below in figure 8.

One can see that the results match to four significant figures, thereby verifying the correct operation of the Fortran program.

3.2 Diffusivity Extractor

Constant Diffusivity

This program was written to extract diffusivity values from two concentration profiles. This matrix of diffusivity values can then be used to predict future concentration profiles with respect to both time and temperature. The program begins by asking the user for the two concentration profiles file in .txt format. The array size, time step, and time duration are then inputted. The software's output is both on screen and in a .txt file located in the same location as the concentration profiles. The user can then easily import this file into excel to see the graph of the diffusivity matrix.

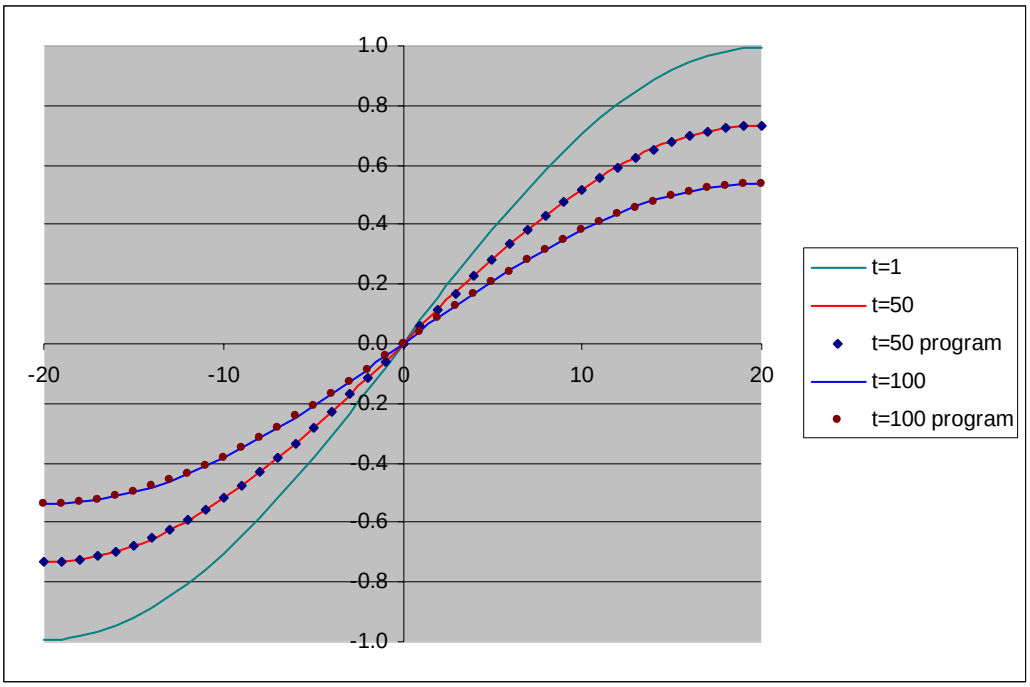

Figure 8. Graph showing the trigonometric approximation compared with the Fortran program.

In order to test this portion of the software, the thin-film solution was used again. A time of 0.005 s was inputted into equation 79 in order to generate one concentration profile. A second profile was generated using a time of 0.00503 s demonstrating a time step of 0.00003 s. In this case, the diffusivity was set as a constant value of 1 to generate both of these curves. In order for the program to be operational, it would need to extract a diffusivity matrix with the value 1 in each location. The resulting extraction is shown in graphical form in figure 9.

As one can see, the diffusivity values oscillate at first and finally converge to a value of 1.06, which leaves a 5% error since 1 is the true value. Since this is within the acceptable tolerance, this portion of the program is deemed operational.

4. Software Limitations

The Fortran programs written for this research project depend solely on input and output external files. These files are currently in .txt format which rely on exact formatting and data placement specifications. Slight alterations in either of these variables will inherently affect the operation of the main program. If these files are converted to spreadsheet files (i.e., Excel), data manipulation and representation will become more efficient. The programmer will need to write an SQL subroutine to allow standard Fortran output to be inserted into a spreadsheet.

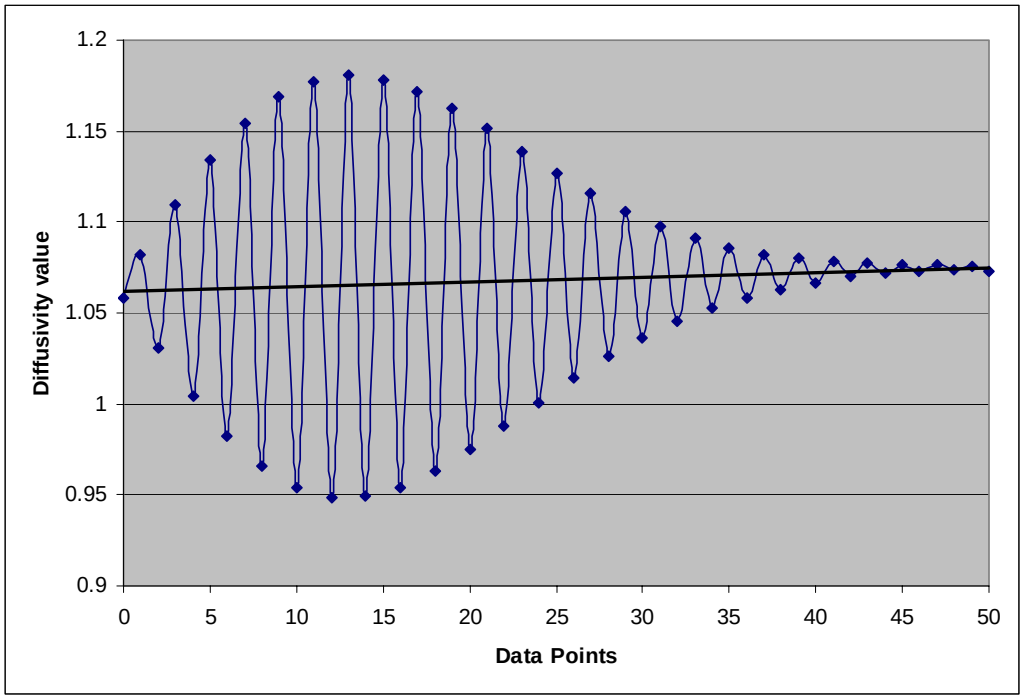

Figure 9. Diffusivity matrix plotted for constant diffusivity of 1.

Throughout this research, many different programs were written to solve different aspects of the problem such as binary_effd, inverse-rewrite , and ternary_separate and can be seen in appendices A, B, and C respectively.

4.1 Binary_effd

This program was written to act as a concentration profile predictor for use with constant diffusivity. It uses the condensed form of Fick's second law when 'D' is not a function of concentration. The program uses the Crank-Nicholson finite-difference method to iterate to future unknown time steps to predict concentration profiles for any given initial binary diffusion data set.

While this program runs successfully using known diffusion examples, the largest error seems to come from boundary condition determination. The program began with Dirichlet boundary conditions in which specify the value of the function at the surface and the finite difference only takes place between them (i.e., i = 2…n-1).

$$T = f(\mathbf{r}, t). \tag{81}$$

This scenario did not work with Fortran so the boundary conditions were changed to Neumann boundary conditions which specify the normal derivative of the function on the surface.

$$\frac{\partial T}{\partial n} = \hat{\mathrm{n}} \cdot \nabla T = f(\mathrm{r}, t). \tag{82}$$

The Neumann scenario proved effective when comparing the programs' output to known solutions. This program successfully uses the Thomas algorithm to solve for the concentration values at future unknown time steps using the previous known concentration values.

4.2 Inverse_rewrite

This program is designed to act as a diffusivity extractor which would extract the composition-dependent interdiffusion coefficients from the concentration profiles in a single diffusion couple. The procedure is based on the minimization of the difference between the profiles calculated by a finite difference scheme and the experimental profiles given by ARL. This program works flawlessly when modeled after the thin-film solution as seen in figure 10.

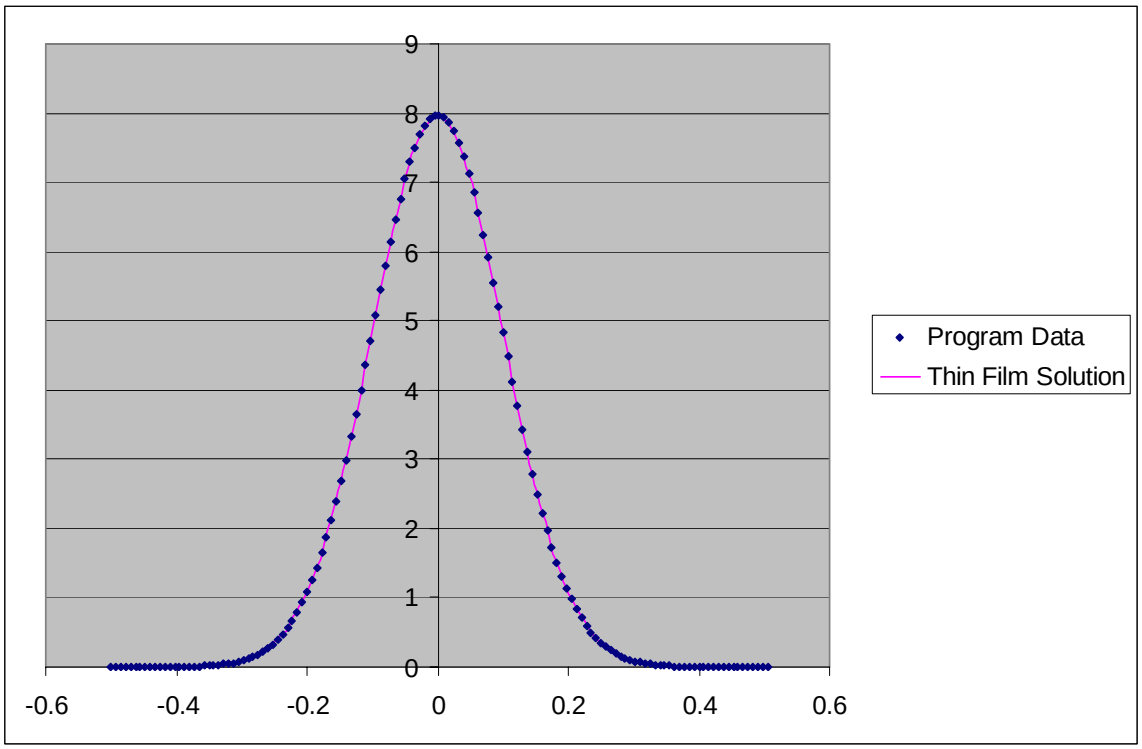

Figure 10. Thin film approximation using program.

This program does not perform as well when using actual data from ARL's test matrix. Unfortunately, due to the variation in the input data, the output does not converge on a specific value as seen in figure 11.

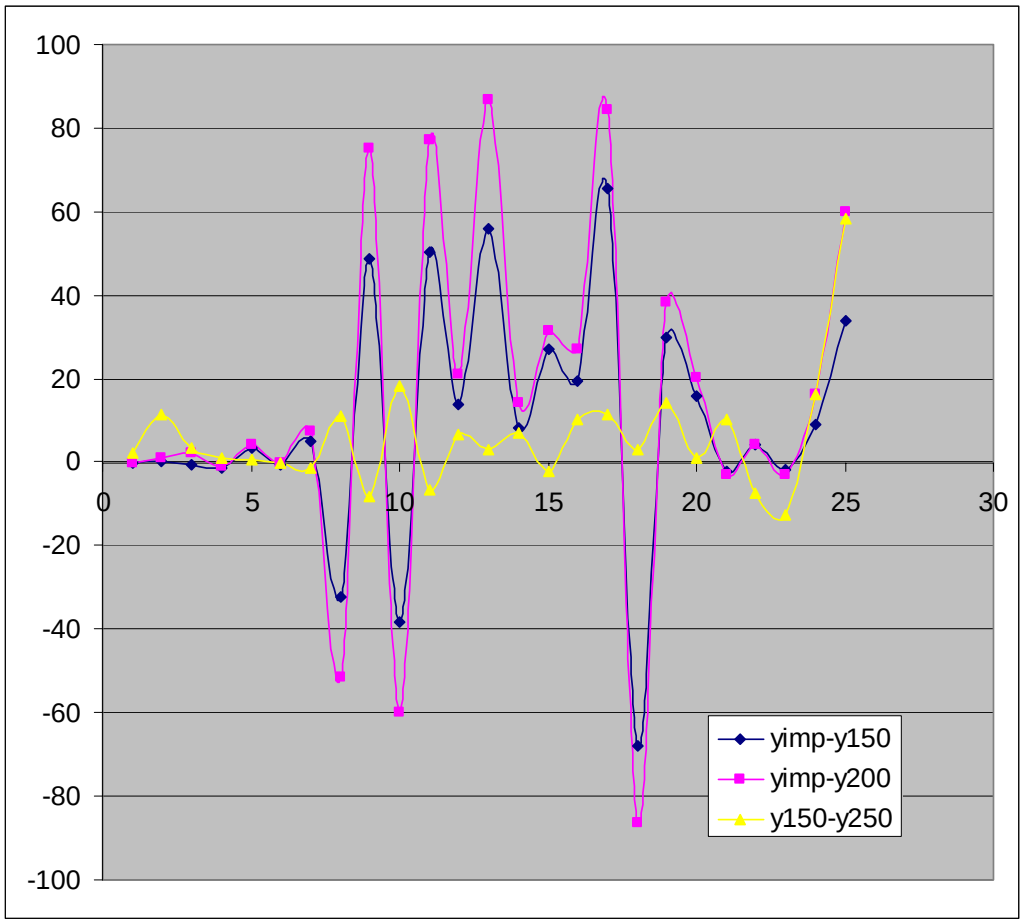

Figure 11. Program output using ARL input data.

Some effort was made to computationally induce noise to simulate experimental concentration scatter. A Gaussian noise with a standard deviation of 0.65% was applied to the raw data of ARL. Unfortunately, this caused the program's output to fluctuate even more. The inter-diffusion coefficient values were meant to be obtained along the entire diffusion path, instead of only at the intersection point of independent paths (Boltzmann-Matano; BZMA) (*8*) or instead of mean coefficient values (Krishtal; KMAZ) (*9*).

The researcher believes the error in this program focuses on the initial values that are introduced. It is assumed that a genetic algorithm would need to be developed to apply to the first iterative step. This algorithm would aid in the location of a true minimum and not a local one as seen in the current program. It is also alleged that the stopping criterion for this program was not developed properly. This criterion should allow a point at when it is reached, a test would be performed on the different terms to ensure they are of the same order of magnitude. In order to maintain this criterion constant during the while profile treatment, and in order to limit calculation time, a variable increment would need to be implemented. This increment could be used as follows: if the relative variation of the concentration exceeds 10%, the increment is

decreased so that the relative variation is no more than 1%. In that case, for the next iteration, it is necessary to employ the previously discussed genetic algorithm to determine the interdiffusion coefficients. This may explain the wide fluctuations seen in figure 11.

The researcher hypothesizes that a smoothing subcode would need to be applied on the raw ARL date before using the Fortran program on it. An appropriate smoothing function is described in equation 83.

$$C_i(x) = A_{i0} + \sum_{j=1}^{q} \frac{A_{ij}}{1 + (1 - B_{ij} \cdot m_{ij}) \cdot \exp\left(-\frac{x - p_{ij}}{r_{ij}}\right) + B_{ij} \cdot \exp\left(-\frac{n_{ij} \cdot (x - p_{ij})}{r_{ij}}\right)} \tag{83}$$

In order to use the smoothing procedure efficiently, an adequate number of functions (q) must be used. In this case, five should be sufficient. The average relative error on the concentration associated with the smoothing procedure amounts to 0.02%, and the maximum relative error, generally observed at extrema, never exceeds 0.5%.

Inverse solutions are known to be sensitive to changes in input data resulting from measurement and modeling errors. Hence, they may not be unique. Mathematically, the inverse problems belong to the class of *ill-posed* or *ill-conditioned* problems; that is, their solutions do not satisfy the general requirements of existence, uniqueness, and stability under small changes to the input data (*10*).

4.3 Ternary_separate

This program was written to scale the previous programs into ternary and multicomponent diffusion with constant and variable diffusivity. In the ternary program, Fortran would not recognize the second element array no matter how it was represented. A third element is not needed in the program as the third element is always determined by the balance of the other two to total 100%. It is here that a two-dimensional (2-D) array written in C++ would be easier to code and debug. Fortran's limitations make it difficult for the program to access a 2-D matrix and perform the necessary calculations while keeping track of each data point within the matrix. When forming the three-dimensional array known as 'C', Fortran continually disallowed a non-integer in the 'countreal' slot. This variable began as an integer but somehow was converted to a non-integer within the program itself.

Versions of the Crank-Nicholson method and the Thomas algorithm were again used in this program with adjustments made to account for the additional variables used in multicomponent diffusion. The finite-difference nomenclature is the same as that used for the binary diffusion case. This program had difficulty interpreting the bi-tridiagonal matrix that is generated from attempting to solve multicomponent diffusion. Only a single tridiagonal matrix is generated when solving a binary diffusion problem.

5. Conclusions

In conclusion, the original objective of this project was not met to its completion. Several factors contributed to this shortcoming, but the primary obstacle was the correlation between the software and the input data. While the software ran successfully with many different known solutions, it did not perform well using actual concentration profile data from ARL. Most likely, this is due to the limited amount of species data, the accuracy of the data itself, and the spacing between each data point. As with all software, the closer the data points and the less intense the noise is, the more accurate the solution will be. This is especially true when using a method such as the finite difference method, which is based solely on iteration.

In order to make this program successful, careful analysis of the input data must be made. All efforts should be taken to obtain more accurate, smoother input data which will allow the software to run with fewer obstacles and, in turn, produce cleaner output data. Once the input data is appropriate, the programmer should return to "fine tune" the individual software programs to allow them to work with the new data. This may require using a filtering subroutine in order to accept only worthy data from the input stream. There should be a minimum number of data points that the program can run on and still produce acceptable results. It is imperative that this number is determined so the program can be modified to run on a sufficient amount of data points. This part of the code would terminate the program if there were fewer data points than the program needed.

Finally, the concentration profile predictor should also be expanded to account for variable diffusivity.

6. References

1. Crank, J. *The Mathematics of Diffusion*; Oxford University Press: England, 1975.
2. Callister, William D. *Materials Science and Engineering*; John Wiley and Sons: Canada, 1997.
3. Yeung, W. K.; Lam, T. T. Second-Order Finite Difference Approximation for Inverse Determination of Thermal Conductivity. *Int. Journal of Heat and Mass Transfer* **1995**, *39* (17), 3685–3693.
4. Morral, J. E.; Dupen, B. M.; Law, C. C. Application of Commercial Computer Codes to Modeling the Carburizing Kinetics of Alloy Steels. *Metallurgical Transactions A* **1992**, *23a*, 2069–2071.
5. Kirkaldy, J. S. *Can. J. Phys*. **1958**, *36*, 899.
6. Dayananda, M. A.; Behnke, D. A. *Scripta Metall.* **1991** *11a*, 2187–2191.
7. Hoffman, J. D. *Numerical Methods for Engineers and Scientists*; New York: McGraw-Hill, 1992.
8. Philibert, J. *Diffusion et Transport de Matie`re dans les Solides*. Les e´ditions de Physique, 1985.
9. Krishtal M.; Mokrov A.; Akimov, V.; Zakharov, P. *Fiz Metal Metalloved* **1973**; 35:1234.
10. Ozisik, M. N. *Heat Conduction*; Chapters 12 and 14, 2nd ed.; John Wiley and Sons: New York, NY, 1993.

Appendix A. Binary_effd

Binary_effd

```
module setup_info

        real :: time, dt, dx, t
        real :: alpha, xpos, conc
        integer :: i
        real :: j

        data dt,dx,time /0.00005,0.01,.01/

end module setup_info

program binary_effd

        use setup_info
        implicit none
        interface

                subroutine fill_a(C,a,nx,effd)
                        integer :: nx
                        real, dimension(0:nx) :: C
                        real, dimension(0:nx,4) :: a
                        real, dimension(0:nx) :: effd
                end subroutine fill_a

                subroutine tridiag(C,a,nx)
                        integer :: nx
                        real, dimension(0:nx) :: C
                        real, dimension(0:nx,4) :: a
                end subroutine tridiag

        end interface


        integer, parameter :: NSEG = 40


        real, dimension (0:NSEG) :: effd !This is the efective D array
        real, dimension (0:NSEG,4) :: a
```

This appendix appears in its original form, without editorial change.

```
real, dimension (0:NSEG) :: C     !Initial condition
real, dimension (0:NSEG) :: xposa,conca
integer :: count
!character (len=1) :: tab = char(9)
!***Enter data and info

        open (unit=10,  file='conc.txt', status='old') !Opening the initial concentration profile
        open (unit=13,  file='ds.txt', status='old') !Opening the effective D array file
        open (unit=100, file='binaryoutputtrig.txt', status='unknown')

        !dx=1/NSEG
        !print*, 'dx is',dx
        alpha=dt/(dx*dx)
        !write(*,*)dx,dt,'   alpha is', alpha

        count=-1;
        5       read (10,*,END=15) xpos,conc
                count=count+1;
                xposa(count)=xpos;
                conca(count)=conc;
                go to 5

        15 if (count.EQ.0) then
                        print*, 'No data in file'
                else
                end if

        C=conca;
        !print*,C


        !Input effective D's into 'effd' array
        count=-1;
        6       read (13,*,END=16) effd
                count=count+1;
                go to 6

        16 if (count.EQ.0) then
                !print*, 'No data in file'
                else
                end if

        !print*, effd(0)


!***Crank Nicholson Method
```

```
t=100.0
count=0

do  j=0,time,0.005
        !if (t>time) exit
                !count=count+1
                !t=count*dt

                call fill_a(C,a,NSEG,effd)
                call tridiag(C,a,NSEG)        !Update C(i) to new time step


end do

print*, C
write (100,99) C;
99 format (1X, F10.4);

close (unit=100)
end program binary_effd



!************Subroutine Fill_a *********

subroutine fill_a(C,a,nx,effd)
use setup_info
integer :: nx
real, dimension(0:nx) :: C
real, dimension(0:nx,4) :: a
real, dimension(0:nx) :: effd



        do i=1,nx-1
                a(i,1) = -alpha/2.*effd(i)
                a(i,2) = 1. + (alpha*effd(i))
                a(i,3) = a(i,1)
                a(i,4) = C(i)*(1.-(alpha*effd(i))) + ((C(i-1)+C(i+1))*alpha/2.*effd(i))
        end do

        !print*, a(:,4);

        !This is for i=0
        a(0,1) = 0.*effd(0)
```

```
        a(0,2) = 1. + (alpha*effd(0))
        a(0,3) = -alpha*effd(3)
        a(0,4) = C(0)*(1.-(alpha*effd(0))) + (C(1)*alpha*effd(0))

        !This is for i=NSEG
        a(nx,1) = -alpha*effd(nx)
        a(nx,2) = 1. + alpha*effd(nx)
        a(nx,3) = 0.*effd(nx)
        a(nx,4) = C(nx)*(1.-(alpha*effd(nx))) + (C(nx-1)*alpha*effd(nx))

        !print*, a(nx,4);

end subroutine fill_a


!*********Tridiagonal matrix ************

subroutine tridiag(C,a,nx)

real, dimension(0:nx) :: C
real, dimension(0:nx,4) :: a
integer :: nx
real :: denom
integer :: j

C(0)=a(0,4)/a(0,2)
a(0,3)=a(0,3)/a(0,2)

do j=1,nx
        denom = a(j,2) - a(j,1)*a(j-1,3)
        C(j) = (a(j,4) - a(j,1)*C(j-1))/denom
        a(j,3) = a(j,3)/denom
end do

!a(nx,3) is 0
do j=nx-1,0,-1
        C(j) = C(j) - a(j,3)*C(j+1)
end do

end subroutine tridiag
```

Appendix B. Inverse_rewrite

Inverse_rewrite

```
module setup_info

        real :: time, dt, dx, t
        real :: alpha, xpos, conc, xpos2, conc2
        integer :: i
        real :: j

        data dt,dx,time /0.00005,0.01,10/


end module setup_info


program inverse_rewrite

        use setup_info
        implicit none
        interface

                subroutine fill_a(a,b,c,r,nx,yimp,y150)
                        integer :: nx
                        !real, dimension(0:nx) :: C
                        real, dimension(0:nx) :: a,b,c,r,h
                        !real, dimension(0:nx,0:nx) :: a
                        real, dimension(0:nx) :: yimp,y150
                end subroutine fill_a

                subroutine tridiag(nx,a,b,c,r,h)
                        integer :: nx
                        real, dimension(0:nx) :: h,a,b,c,r
                        !real, dimension(0:nx,0:nx) :: a
                end subroutine tridiag

        end interface



        integer, parameter :: NSEG = 67
```

This appendix appears in its original form, without editorial change.

```
!real, dimension (0:NSEG,0:NSEG) :: a
real, dimension (0:NSEG) :: h,a,b,c,r
real, dimension (0:NSEG) :: xposa,yimp,xpos2a,y150
integer :: count
!character (len=1) :: tab = char(9)



!***Enter data and info

        open (unit=10,  file='delta1.txt', status='old')
        open (unit=13,  file='delta2.txt', status='old')
        open (unit=100, file='output.txt', status='unknown')

        !dx=1/NSEG
        !print*, 'nx is',NSEG
        alpha=(dx*dx)/dt
        !write(*,*)dx,dt,'   alpha is', alpha

        count=-1;
        5       read (10,*,END=15) conc
                count=count+1;
                !xposa(count)=xpos;
                yimp(count)=conc;
                go to 5

        15 if (count.EQ.0) then
                        print*, 'No data in file'
                else
                end if

        !print*,yimp

        count=-1;
        6       read (13,*,END=16) conc2
                count=count+1;
                !xpos2a(count)=xpos2;
                y150(count)=conc2;
                go to 6

        16 if (count.EQ.0) then
                !print*, 'No data in file'
                else
```

```
                    end if

                !print*, y150


        !***Crank Nicholson Method

        t=100.0
        count=0

        !do  j=0,time,0.5
                !if (t>time) exit
                        !count=count+1
                        !t=count*dt

                        call fill_a(a,b,c,r,NSEG,yimp,y150)
                        call tridiag(NSEG,a,b,c,r,h)

!print*,'hnx is',h(NSEG)

        !end do

        print*, h
        write (100,99) h;
        99 format (1X, F10.4);

        close (unit=100)
        end program inverse_rewrite



        !*************Subroutine Fill_a *********

        subroutine fill_a(a,b,c,r,nx,yimp,y150)
        use setup_info
        integer :: nx
        real, dimension(0:nx) :: a,b,c,r
        !real, dimension(0:nx,0:nx) :: a
        real, dimension(0:nx) :: yimp,y150

        !*****This is for i=0********

                !a(1)=0!!!

                b(0) = 2.0*yimp(0)-3.0*yimp(1)+yimp(2)
                !print*,b(0)
```

```
        c(0) = -yimp(0)+yimp(1)
        !print*,c(0)

        r(0) = alpha*(y150(0)-yimp(0))
        !print*,'y150(0) is',y150(0)
        !print*,'alpha is',alpha
        !print*,r(0)


!*****This is for 0<i<NSEG********

        do i=1,nx-1
                a(i) = yimp(i-1)-yimp(i+1)
                b(i) = 4*(yimp(i+1)-2*yimp(i)+yimp(i-1))
                c(i) = yimp(i+1)-yimp(i-1)
                r(i) = 4*alpha*(y150(i)-yimp(i))
        end do

        !print*,'yimp(0) is',yimp(0)
        !print*,'yimp(1) is',yimp(1)
        !print*,'yimp(2) is',yimp(2)

        !print*,a(1);
        !print*,b(1);



!******This is for i=NSEG**********

        !a(nx,1) = 0
        b(nx) = 2*(yimp(nx-1)-yimp(nx))
        !a(nx,3) = 0
        r(nx) = alpha*(y150(nx)-yimp(nx))

        !print*, a(nx,4);

end subroutine fill_a


!*********Tridiagonal matrix ************

subroutine tridiag(nx,a,b,c,r,h)

!real, dimension (0:nx,0:nx) :: a
```

```
real, dimension(0:nx) :: a,b,c,r,h
real bet
real gam(100)
integer j

if(b(0).eq.0)pause 'tridiag:rewrite equations'



bet=b(0)
!print*, 'bet is',bet

h(0)=r(0)/bet
h(nx)=1.0
!print*,'h(0) is',h(0)

!**Decomposition and forward substitution
do 2 j=1,nx
        gam(j)=c(j-1)/bet
        !print*,gam(1)
        bet=b(j)-a(j)*gam(j)
        !print*,bet
        if (bet.eq.0)pause 'tridiag failed'  !Algorithm fails
        h(j)=(r(j)-a(j)*h(j-1))/bet

2 continue

!**Backsubstitution**

do 3 j=nx-1,0,-1
                h(j)=h(j)-gam(j+1)*h(j+1)
3 continue

return

end subroutine tridiag
```

INTENTIONALLY LEFT BLANK.

Appendix C. Ternary_separate

Ternary_separate

```
module setup_info

        real :: time, dt, dx, t
        !Xpos is the array of positions in the initial profile
        !Conc is the array of concentrations in the initial profile
        real :: alpha, xpos1, conc1, xpos2, conc2,dtt
        integer :: i,k,z
        real :: j

        data dt,dx,time /0.5,0.01,50/


end module setup_info

program ternary_separate

        use setup_info
        implicit none
        interface

                !The following subroutine is designed to make arrays that
                !follow the Crank Nicholson Finite Difference method. 'A' will
                !have four columns, representing the four coefficients in the
                !finite difference equation mentioned earlier which come before
                !C(i-1,j+1), C(i,j+1), C(i+1,j+1), and C(*,j)
                !'nx' is the maximum position of x
                !The 'Conc' array represents the known values of the concentrations
                !at the current time step

                subroutine fill_a(Conc,a,nx)
                        integer :: nx
                        real, dimension(0:nx) :: Conc
                        real, dimension(0:nx,4) :: a
                !       real :: dtt2,j
                end subroutine fill_a

                !The following subroutine is designed to solve the tridiagonal matrix
                !that was formed using the Crank Nicholson method
```

This appendix appears in its original form, without editorial change.

```
        !It uses the same variables as in subroutine 'fill_a'
        !This routine returns the concentration values at the next time step
        subroutine tridiag(Con,a,nx)
                integer :: nx
                real, dimension(0:nx) :: Con
                real, dimension(0:nx,4) :: a
        end subroutine tridiag

end interface


!This parameter defines the number of steps in the x direction
!For example, if the profile goes from -20 to 20, then NSEG=40
integer, parameter :: NSEG = 40
real, parameter :: timestep = 1000
!real, parameter :: dt=0.5


real, dimension (0:NSEG,4) :: a          !Coefficient array
real, dimension (0:NSEG,timestep) :: C     !Initial condition
real, dimension (0:NSEG) :: xposa,conca,xposb,concb,conarray
integer :: count
real :: countreal
!character (len=1) :: tab = char(9)



!***Enter data and info

        !This statement opens the text files containing the
        !two initial concentration profiles
        open (unit=10,  file='trig.txt', status='old')
        open (unit=11,  file='trig2.txt', status='old')
        !This statement opens the ouput file so the new concentration
        !profile at the current time step can be written
        open (unit=100, file='binaryoutputtrig.txt', status='unknown')

        !dx=1/NSEG
        !print*, 'dx is',dx
        alpha=dt/(dx*dx)
        !write(*,*)dx,dt,'   alpha is', alpha

        !This loop will take only the proper number of x positions and
        !concentrations, storing them into 'xposa' and 'conca',
        !leaving off the trailing zeros
        count=-1;
```

```
5       read (10,*,END=15) xpos1,conc1
        count=count+1;
        xposa(count)=xpos1;
        conca(count)=conc1;
        go to 5

15 if (count.EQ.0) then
                print*, 'No data in file'
        else
        end if

!Same for second profile
count=-1;
6       read (11,*,END=16) xpos2,conc2
        count=count+1;
        xposb(count)=xpos2;
        concb(count)=conc2;
        go to 6

16 if (count.EQ.0) then
                print*, 'No data in file'
        else
        end if



!Define the initial concentration array at time=0 for both elements




      !*************************************************************************
*****

        C(:,0)=concb;  !****change a to b to a to change elements


      !*************************************************************************
******



!C(:,2,0)=concb;

!print*, C(:,2,0)
```

```
!Begin the loop to solve over the entire time, incrementing
!by 0.5

t=0.0;
countreal=0.0;

dtt=time/.5; !Gives how large the time column is in the array
!print*,dtt


!print*, C(:,2,0)


!*****This is the set up for only one component******


do  j=0,15,0.5
                !if (t>time) exit
                        countreal=countreal+1 !monitor j increments
                        !t=countreal*dt

                        !Store one column of 3D array into 1D array 'conarray'
                        !This is done because the subroutine 'fill_a' only needs the
                        !concentrations and not the position and time subscripts
                        conarray=C(:,j)

                        !print*,z
                        !print*, conarray
                        !print*, C(:,1,0)

                        call fill_a(conarray,a,NSEG)            !Fill in the coefficients for the current time
step


                        !print*, C(1,i,0)
                        !print*, a(5,4)

                        call tridiag(conarray,a,NSEG)       !Update C(i) to new time step



                        !This loop will move the new concentration profile back into
                        !the 3D array with the appropriate position and time subscripts
                        do k=0,NSEG
```

```
                        C(k,countreal)=conarray(k)  !****Won't allow non-integer for countreal
slot
                    end do


end do

    !print*, countreal

    !print*, C(:,15)! ********* Test print to show concentration profile across all x positions
            ! for a specific element at a specific time step **********



    !write (100,99) C(:,275);        !Write the current concentration profile to the output file
    !99 format (1X, F7.4);


    close (unit=100)            !Close the output file
    end program ternary_separate !Close the program



    !************Subroutine Fill_a *********

    subroutine fill_a(Conc,a,nx)
    use setup_info                  !Uses the parameter module
    integer :: nx
    !real :: dtt2,z
    real, dimension(0:nx) :: Conc
    real, dimension(0:nx,4) :: a

    !print*,Conc

        !Loop to enter the four coefficients of the finite
        !difference equation
        do i=1,nx-1
            a(i,1) = -alpha/2.
            a(i,2) = 1. + alpha
            a(i,3) = a(i,1)
            a(i,4) = Conc(i)*(1.-alpha) + (Conc(i-1)+Conc(i+1))*alpha/2.
        end do

        !print*, a(:,4);

        !These next two sets are different because there is no previous
```

```
        !point for i=0 and there is no future point for i=NSEG

        !This is for i=0
        a(0,1) = 0.
        a(0,2) = 1. + alpha
        a(0,3) = -alpha
        a(0,4) = Conc(0)*(1.-alpha) + Conc(1)*alpha

        !This is for i=NSEG
        a(nx,1) = -alpha
        a(nx,2) = 1. + alpha
        a(nx,3) = 0.
        a(nx,4) = Conc(nx)*(1.-alpha) + Conc(nx-1)*alpha

        print*, Conc
        !print*,a(nx,4);

end subroutine fill_a


!*********Tridiagonal matrix ************

subroutine tridiag(Con,a,nx)

real, dimension(0:nx) :: Con
real, dimension(0:nx,4) :: a
integer :: nx
real :: denom !Represents denominator after calculation
integer :: j

!Initial values
Con(0)=a(0,4)/a(0,2)
a(0,3)=a(0,3)/a(0,2)

!This loop sets all ofthe known information to variables
do j=1,nx
        denom = a(j,2) - a(j,1)*a(j-1,3)
        Con(j) = (a(j,4) - a(j,1)*Con(j-1))/denom
        a(j,3) = a(j,3)/denom
end do

!This loop decides the next concentration at the next time step
!a(nx,3) is 0
do j=nx-1,0,-1
        Con(j) = Con(j) - a(j,3)*Con(j+1)
        !print*,Con(2)
```

```
end do

end subroutine tridiag
```

NO. OF COPIES	ORGANIZATION
1 (PDF ONLY)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA 8725 JOHN J KINGMAN RD STE 0944 FORT BELVOIR VA 22060-6218
1	US ARMY RSRCH DEV & ENGRG CMD SYSTEMS OF SYSTEMS INTEGRATION AMSRD SS T 6000 6TH ST STE 100 FORT BELVOIR VA 22060-5608
1	INST FOR ADVNCD TCHNLGY THE UNIV OF TEXAS AT AUSTIN 3925 W BRAKER LN AUSTIN TX 78759-5316
1	DIRECTOR US ARMY RESEARCH LAB IMNE ALC IMS 2800 POWDER MILL RD ADELPHI MD 20783-1197
3	DIRECTOR US ARMY RESEARCH LAB AMSRD ARL CI OK TL 2800 POWDER MILL RD ADELPHI MD 20783-1197

ABERDEEN PROVING GROUND

1	DIR USARL AMSRD ARL CI OK TP (BLDG 4600)

NO. OF
COPIES ORGANIZATION

1 HQDA
DAMO FDT
400 ARMY PENTAGON
WASHINGTON DC 20310-0460

1 DIRECTOR
US ARMY RSCH LAB
AMSRD ARL D
J MILLER
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 HQDA DIR R&D SAAL TR
W MORRISON
SUITE 9800
2511 JEFFERSON DAVIS HWY
ARLINGTON VA 22201

1 HQ US ARMY
MATERIEL CMD
9301 CHAPEK RD
FORT BELVOIR VA 22060-5527

1 US ARMY BMDS CMD
ADVANCED TECHLGY CTR
PO BOX 1500
HUNTSVILLE AL 35807-3801

1 OFC OF THE PRODUCT MGR
SFAE AR HIP IP
R DE KLEINE
PICATINNY ARSENAL NJ 07806-5000

1 US ARMY ARDEC
PROD BASE MODRNZTN AGENCY
AMSMC PBM
A SIKLOSI
PICATINNY ARSENAL NJ
07806-5000

1 US ARMY ARDEC
PROD BASE MODRNZTN AGENCY
AMSTA AR WES L LAIBSON
PICATINNY ARSENAL NJ
07806-5000

3 PM PEO ARMAMENTS
TANK MAIN ARMAMENT SYS
AMCPM TMA
AMCPM TMA 105
AMCPM TMA AS
H YUEN
PICATINNY ARSENAL NJ
07806-5000

NO. OF
COPIES ORGANIZATION

2 US ARMY ARDEC
AMSTA AR CCH B
C MANDALA
E FENNELL
PICATINNY ARSENAL NJ
07806-5000

1 US ARMY ARDEC
AMSTA AR CCS
PICATINNY ARSENAL NJ
07806-5000

1 US ARMY ARDEC
AMSTA AR WE
PICATINNY ARSENAL NJ
07806-5000

2 PM MAS
SFAE AMO MAS SMC
PICATINNY ARSENAL NJ
07806-5000

1 US ARMY ARDEC
AMSRD AAR AEM D
J LUTZ
BLDG 354
PICATINNY ARSENAL NJ
07806-5000

3 US ARMY ARDEC
AMSTA AAR AEE W
M MEZGER
D WIEGAND
P LU
BLDG 3022
PICATINNY ARSENAL NJ
07806-5000

1 US ARMY ARDEC
AMSRD AAR AIL F
G FERDINAND
BLDG 1
PICATINNY ARSENAL NJ
07806-5000

2 US ARMY ARDEC
AMSTA AR WEE
S WESTLEY
S BERNSTEIN
PICATINNY ARSENAL NJ
07806-5000

NO. OF
COPIES ORGANIZATION

1 US ARMY ARDEC
AMSRD AAR AEE W
S EINSTEIN
BLDG 382
PICATINNY ARSENAL NJ
07806-5000

1 US ARMY ARDEC
SFAE AMO CAS
J RUTKOWSKI
BLDG 171 M
PICATINNY ARSENAL NJ
07806-5000

1 US ARMY ARDEC
AMSRD AAR AEI W
B BRODMAN
BLDG 472
PICATINNY ARSENAL NJ
07806-5000

1 US ARMY ARDEC
AMSRD AAR AEE W
P OREILLY
BLDG 3028
PICATINNY ARSENAL NJ
07806-5000

1 US ARMY ARDEC
SFAE AMO CAS R
R CIRINCIONE
PICATINNY ARSENAL NJ
07806-5000

1 US ARMY ARDEC
AMSRD AAR AEE W
P HUI
BLDG 382
PICATINNY ARSENAL NJ
07806-5000

1 US ARMY ARDEC
AMSRD AAR AEE W
J OREILLY
BLDG 382
PICATINNY ARSENAL NJ
07806-5000

1 AMSTA AR FS
T GORA
PICATINNY ARSENAL NJ
07806-5000

NO. OF
COPIES ORGANIZATION

1 US ARMY ARDEC
AMSTA AR FS DH
PICATINNY ARSENAL NJ
07806-5000

1 US ARMY ARDEC
AMSRD AAR AEE W F(D)
R KOPMANN
BLDG 62N
PICATINNY ARSENAL NJ
07806-5000

1 US ARMY ARDEC
AMSRD AAR ATD
B MACHAK
BLDG 1
PICATINNY ARSENAL NJ
07806-5000

1 US ARMY ARDEC
AMSRD AAR AEM C
K CHUNG
BLDG 407
PICATINNY ARSENAL NJ 07806-5000

1 DIR BENET WEAPONS LAB
AMSTA AR CCB T
S SOPOK
WATERVLIET NY
12189-4050

1 DIR BENET WEAPONS LAB
AMSTA AR CCB TA
M AUDINO
WATERVLIET NY 12189-4050

1 DIR BENET WEAPONS LAB
AMST AAR CCB D
R HASENBEIN
WATERVLIET NY
12189-4050

2 CDR US ARMY RSRCH OFC
TECH LIB
D MANN
PO BOX 12211
RESEARCH TRIANGLE PARK NC
27709-2211

1 PM US ARMY TANK AUTOMOTIVE
CMD
AMCPM ABMS
T DEAN
WARREN MI 48092-2498

NO. OF
COPIES ORGANIZATION

1 PM US ARMY TANK AUTOMOTIVE
CMD
FIGHTING VEHICLES SYSTEMS
SFAE ASM BV
WARREN MI 48397-5000

1 PM ABRAMS TANK SYSTEM
SFAE ASM AB
WARREN MI 48397-5000

1 DIR HQ TRAC RPD
ATCD MA
FT MONROE VA 23651-5143

1 CDR
RADFORD ARMY
AMMUNITION PLANT
SMCAR QA HI LIB
RADFORD VA 24141-0298

1 COMMANDANT
USAFC&S
ATSF CN
P GROSS
FT SILL OK 73503-5600

4 CDR NAVAL RSRCH LAB
TECH LIBRARY
CODE 4410
K KAILASANATE
J BORIS
E ORAN
WASHINGTON DC 20375-5000

1 OFFICE OF NAVAL RSRCH
CODE 473 J GOLDWASSER
800 N QUINCY ST
ARLINGTON VA 22217-9999

5 COR NSWC
S MITCHELL
C MICHIENZI
J CONSAGA
C GOTZMER
TECHLIB
INDIAN HEAD MD 20640-5000

1 CDR
NAVAL SURFACE WARFARE CTR
CODE G30
GUNS & MUNITIONS DIV
DAHLGREN VA 22448-5000

NO. OF
COPIES ORGANIZATION

1 CDR
NAVAL SURFACE WARFARE CTR
CODE G32
GUNS SYSTEMS DIV
DAHLGREN VA 22448-5000

1 CDR
NSWC
CODE E23
TECHLIB
DAHLGREN VA 22448-5000

1 CDR
NSWC
R HUBBARD G33
DAHLGREN VA 22448-5000

2 CDR
NAVAL AIR WARFARE CTR
CODE 3895
T PARR
R DERR
CH1NA LAKE CA 93555-6001

1 CDR
NAVAL AIR WARFARE CTR
INFORMATION SCIENCE DIV
CHINA LAKE CA 93555-6001

1 WL MNME
ENERGETIC MATERIALS BR
2306 PERIMETER RD
STE 9
EGLIN AFB FL 32542-5910

1 DIR SANDIA NATL LABS
M BAER DEPT 1512
PO BOX 5800
ALBUQUERQUE NM 87185

1 DIR SANDIA NATL LABS
COMBUSTION RSRCH FACILITY
R CARUNG
LIVERMORE CA 94551-0469

2 DIR LLNL
L355
A BUCHINGHAM
M FINGER
PO BOX 808
LIVERMORE CA 94550-0622

NO. OF
COPIES ORGANIZATION

1 CIA
J BACKOFEN
RM 4PO7 NHB
WASHINGTON DC 20505

2 MILLERSVILLE UNIV
PHYSICS DEPT
C W PRICE
M NOLAN
MILLERSVILLE PA 17551

2 UNIV OF ILLINOIS
DEPT OF MECH INDUSTRY ENGR
H KRIER
R BEDDINI
144 MEB 1206 N GREEN ST
URBANA IL 61801-2978

5 PENNSYLVANIA STATE UNIV
DEPT OF MECHANICAL ENGRG
V YANG
K KUO
S THYNELL
G SETTLES
R YETTER
UNIV PARK PA 16802-7501

1 ARROW TECHLGY ASSOC INC
1233 SHELBURNE RD D 8
SOUTH BURLINGTON VT 05403

1 AAI CORPORATION
D CLEVELAND
PO BOX 126
HUNT VALLEY MD 21030-0126

2 ALLIANT TECHSYSTEMS INC
ALLEGHENY BALLISTICS LAB
W B WALKUP
T F FARABAUGH
PO BOX 210
ROCKET CTR WV 26726

3 ALLIANT TECHSYSTEMS INC
C AAKHUS MN07-LW54
R DOHRN MN07-LW54
D KAMDAR MN07-LW54
5050 LINCOLN DR
EDINA MN 55436

NO. OF
COPIES ORGANIZATION

4 ALLIANT TECHSYSTEMS INC
RADFORD ARMY AMMO PLANT
D A WORRELL
W J WORRELL
S RITCHIE
K BROWN
RADFORD VA 24141-0299

3 ST MARKS POWDER
GENERAL DYNAMICS ARM SYS
J DRUMMOND
J HOWARD
R PULVER
7121 COASTAL HWY
CRAWFORDVILLE FL 32327

1 GENERAL DYNAMICS ARM SYS
J TALLEY RM 1305
LAKESIDE AVE
BURLINGTON VT 05401

1 PRIMEX
BADGER ARMY AMMO PLANT
F E WOLF
BARABOO WI 53913

4 PRIMEX
E J KIRSCHKE
A F GONZALEZ
J DRUMMOND
D W WORTHINGTON
PO BOX 222
SAINT MARKS FL 32355-0222

2 PRIMEX
NHYLTON J BUZZETT
10101 9TH ST NORTH
ST PETERSBURG FL 33716

1 PAUL GOUGH ASSOC INC
P S GOUGH
1048 SOUTH ST
PORTSMOUTH NH 03801-5423

2 VERITAY TECHGY INC
R SALIZONI
J BARNES
4845 MILLERSPORT HWY
EAST AMHERST NY 14501-0305

NO. OF
COPIES ORGANIZATION

1 PRIMEX
E STEINER
DIR LARGE CAL R&D
PO BOX 127
RED LION PA 17356

1 SRI INTERNATIONAL
TECH LIB
PROPULSION SCIENCES DIV
333 RAVENWOOD AVE
MENLO PARK CA 94025-3493

ABERDEEN PROVING GROUND

1 CDR USAATC
CSTE DTC AT SL
R HENDRICKSEN
APG MD 21005

31 DIR USARL
AMSRD ARL WM
B RINGERS
AMSRD ARL WM BC
M BUNDY
J GARNER
P PLOSTINS
P WEINACHT
AMSRD ARL WM BD
W R ANDERSON
R A BEYER
A L BRANT
S W BUNTE
C F CHABALOWSKI
T P COFFEE
J COLBURN
P J CONROY
B E FORCH
B E HOMAN
S L HOWARD
P J KASTE
A J KOTLAR
C LEVERITT
K L MCNESBY
M MCQUAID
M S MILLER
A W MIZIOLEK
J B MORRIS
J A NEWBERRY
M J NUSCA
R A PESCE-RODRIGUEZ
G P REEVES
B M RICE
R C SAUSA
A W WILLIAMS

INTENTIONALLY LEFT BLANK.