

**Optimal Planting Patterns in the
Presence of Pathogen**

Jim Tilson

Tennessee Scholars Project

May 2, 1990

I would like to thank Dr. Suzanne Lenhart for all the help that she has given me during my undergraduate career at the University of Tennessee, and for the wonderful job she did advising me on this project.

I. INTRODUCTION

I.A. Problem

"Yield maximization in modern agriculture depends in part upon successful control of pathogen infestations. Disease control depends upon limiting infective incidents and, if one occurs, isolation of the infective agent from the remainder of the field. Carefully selected spatial patterns of two plant strains may offer a low cost, relatively easy to implement method of controlling pathogenic disease."

From Master's thesis abstract,

Mark T. Bullock, 1986

The purpose of this Tennessee Scholar's project is to investigate the relationship between spatial patterns of crops and crop yield at the end of the growing season. In this simulation there are two types of plant: the first plant has a higher yield than the second, but it also has a higher susceptibility to pathogen and disease than the second. In a disease-free setting, planting only the first type of plant is obviously the best way to maximize harvest; however, a sterile environment is rare, so some type of prevention should be taken. As Bullock points out, using a second plant to act as a buffer against infection in the first plant is environmentally and economically more rewarding. First of all, fewer chemicals would need to be used to prevent infection or to treat infected plants. Secondly, using chemicals to kill the pathogen can sometimes result in a strain of the pathogen which is resistant to the chemicals used to kill it off. Finally, the buffering plants could also be harvested at the end of the growing season, and part of the investment recouped. For further background on intercropping, see Vandermeer (Ref. 3).

I.B. Equations Used

The field is composed of a unit square with coordinates $(0,1) \times (0,1)$, divided into a grid of up to size 30×30 .

The equations that are used in this simulation are as follows:

$$(b_1)_t = a_1(b_1) - \alpha_1 P(b_1)$$

$$(b_2)_t = a_2(b_2) - \alpha_2 P(b_2)$$

where a_1, a_2 =coefficients of plant growth of plant 1 and plant 2, respectively, α_1, α_2 =coefficients of pathogen damage of the plants, respectively, and P =the amount of pathogen involved. For consistency, $a_1 > a_2$ and $\alpha_1 > \alpha_2$ in all runs; this means that plant 1 grows faster than plant 2, but that it also has less resistance to the pathogen.

The equation for the pathogen spread is based on the diffusion equation

$$P_t = P_{xx} + P_{yy} + \alpha_1 P(b_1) + \alpha_2 P(b_2)$$

where the second order partial derivative terms are the tendency of the pathogen to spread. The other terms give the growth of the pathogen.

The amounts of pathogen or plant mass at location (x,y) at time t are denoted by:

$P(x,y,t)$ = amount of pathogen at location (x,y) at time t

$\left. \begin{matrix} b_1(x,y,t) \\ b_2(x,y,t) \end{matrix} \right\}$ = amount of plant at location (x,y) at time t

and the amount of plant around the borders (i.e. $x=0, x=1, y=0$ or $y=1$) is 0 at all times. Thus the starting conditions are:

$$P(x,y,0) = 0$$

$$\left. \begin{matrix} b_1(x,y,0) \\ b_2(x,y,0) \end{matrix} \right\} = \text{given initial pattern}$$

where the plant mass at time $t=0$ is the initial pattern, and the field is clear of all pathogen.

The pathogen flows into the system from the sides. Pathogen flow is determined by the derivative terms $\partial P / \partial n$ for each edge, which are input by the user. The greater these terms are, the greater the amount of pathogen flowing into the field is.

I.C. Hypothesis

The hypothesis for this project is that a high amount of variation in the spatial pattern of the crop yields a greater harvest than a small amount of variation, since the highly resistant plant will act as a buffer against the pathogen for the lesser resistant, higher yielding plant. The three basic patterns to be tested and compared are a checkerboard, a peripherally-bordered field, and a field of rows of alternating plant type. I will use a FORTRAN program based on the above equations to simulate a growing season, and compare the total yields of the different patterns.

II. SIMULATIONS METHODS

II.A. Explicit Finite Difference Scheme

The first scheme which I used in the FORTRAN program was an explicit finite differencing scheme. The equations used in this program to calculate the change in plant mass and pathogen concentrations were as follows:

$$(b_1)_{i,j}^{n+1} = (b_1)_{i,j}^n + \Delta t [a_1(b_1)_{i,j} - \alpha_1(b_1)_{i,j} P_{i,j}^n]$$

$$(b_2)_{i,j}^{n+1} = (b_2)_{i,j}^n + \Delta t [a_2(b_2)_{i,j} - \alpha_2(b_2)_{i,j} P_{i,j}^n]$$

$$P_{i,j}^{n+1} = P_{i,j}^n + \Delta t \left[\frac{(P_{i+1,j}^n - P_{i,j}^n) + (P_{i-1,j}^n - P_{i,j}^n)}{\Delta x^2} + \frac{(P_{i,j+1}^n - P_{i,j}^n) + (P_{i,j-1}^n - P_{i,j}^n)}{\Delta y^2} + \alpha_1(b_1)_{i,j}^n P_{i,j}^n + \alpha_2(b_2)_{i,j}^n P_{i,j}^n \right]$$

The program inputs parameters from the user concerning grid size, grid pattern, length of time, coefficients of growth for plants and pathogen, and boundary conditions for the pathogen. It then runs through a loop computing the density of plant mass and pathogen at each time step, outputting the results to a file specified by the user. The arrays used to store the field information are three dimensional; rather than calculate each new amount, then copy all these amounts back into the matrix when all the calculations were finished, I alternated between the two matrices as to which was the current time step; it is much faster to flip back and forth by changing one variable pointing to the current time step and the next than it is to copy up to 2,700 numbers into new matrices. However, this program tended to run very slowly for any significantly-sized field; convergence for this method depends on the the inequality

$$\Delta t < \frac{\Delta x^2 \Delta y^2}{2(\Delta x^2 + \Delta y^2)}$$

Running this program on a 3x3 matrix representation of the field took very little time, but extending the field size to 30x30 resulted in very long run times. Because of the amount of time that this program took to complete a single run for a significantly-sized matrix, I rewrote it using an implicit scheme--specifically, successive over-relaxation using Crank-Nicolson's formula (Ref. 1).

II.B. Implicit Finite Difference Scheme

Using a finite difference implicit scheme frees the program from the restriction on the size of the time step which the program could use. Thus the speed of the run time was significantly increased. The formulae used in this version of the program were: for the plants,

$$(b_1)_{i,j}^{new} = (b_1)_{i,j}^{old} + \omega \frac{\left[(2 + \Delta t(a_1 - \alpha_1 P_{i,j}^n))(b_1)_{i,j}^n - (2 + \Delta t(-a_1 + \alpha_1 P_{i,j}^{old}))(b_1)_{i,j}^{old} \right]}{(2 + \Delta t(-a_1 + \alpha_1 P_{i,j}^{old}))}$$

$$(b_2)_{i,j}^{new} = (b_2)_{i,j}^{old} + \omega \frac{\left[(2 + \Delta t(a_2 - \alpha_2 P_{i,j}^n))(b_2)_{i,j}^n - (2 + \Delta t(-a_2 + \alpha_2 P_{i,j}^{old}))(b_2)_{i,j}^{old} \right]}{(2 + \Delta t(-a_2 + \alpha_2 P_{i,j}^{old}))}$$

and for the pathogen,

$$P_{i,j}^{new} = P_{i,j}^{old} + \omega \frac{\left[r(P_{i-1,j}^n + P_{i+1,j}^n + P_{i,j-1}^n + P_{i,j+1}^n + P_{i-1,j}^{new} + P_{i,j-1}^{new} + P_{i+1,j}^{old} + P_{i,j+1}^{old}) + (2 - 4r + \Delta t[\alpha_1(B_1)_{i,j}^n + \alpha_2(B_2)_{i,j}^n])P_{i,j}^n - (2 + 4r - \Delta t[\alpha_1(B_1)_{i,j}^{new} + \alpha_2(B_2)_{i,j}^{new}])P_{i,j}^{old} \right]}{2 + 4r - \Delta t(\alpha_1(B_1)_{i,j}^{new} + \alpha_2(B_2)_{i,j}^{new})}$$

where

$$r = \frac{\Delta t}{\Delta x \Delta y}$$

This program is somewhat similar to the finite difference program. The user inputs parameters to the program, which then reads the initial planting pattern data from the user-specified input file. The program runs through a loop, calculating the density of plants and pathogen at each time step and outputting the data from each time step to a user-specified file. However, the method of calculation is different. This method iteratively estimates what the amounts of plant mass and pathogen should be at the new time step. It uses both values at the current time step (superscripted "old") and the current

estimate for the new time step (superscripted "n") to iterate. The program makes estimates for each time step until the two-norm for the difference between matrices containing two successive estimates for each of the plants and the pathogen is less than some error amount specified by the user, i.e.

$$\|P^{new} - P^n\|_2, \|b_1^{new} - b_1^n\|_2, \|b_2^{new} - b_2^n\|_2 < \varepsilon$$

The program then updates the matrix. Because this program runs relatively fast, I did not bother with the alternating parallel matrix approach which I used in the finite differencing program.

IV. CONCLUSION

IV.A. Results of project

The first runs that I made did not seem to support my hypothesis. In fact, planting the field entirely with the high-yield, low-resistance plant gave the greatest amount of plant mass. However, as the coefficient of pathogen susceptibility for the buffering plant was decreased (i.e. its resistance increased), the checkerboard pattern eventually yielded more total plant mass than any other pattern run. My conclusion is that a high degree of variation in the planting arrangement is at its most effective if the buffering plant has a relatively high resistance to the pathogen in question, i.e. that (α_1/α_2) yields a relatively "large" number.

Following is a summary of several runs, giving the maximum amounts achieved by each planting pattern given growth coefficients for plants and pathogen. (See appendix, p. 12, for pattern name explanations)

$a_1=2.0, a_2=0.5, \alpha_1=4.0, \alpha_2=0.5, (\alpha_1/\alpha_2)=8.0$

Pattern:	Evenbar5	Max total plant:	6.4195
	Ckr5		6.4605
	Evenfrm		6.4728
	Allb1		6.5536

$a_1=4.0, a_2=1.0, \alpha_1=8.0, \alpha_2=0.5, (\alpha_1/\alpha_2)=16.0$

Pattern:	Evenfrm	Max total plant:	6.7196
	Evenbar5		6.7444
	Ckr5		6.8246
	Allb1		6.8269

$a_1=4.0, a_2=1.0, \alpha_1=8.0, \alpha_2=0.0625, (\alpha_1/\alpha_2)=128.0$

Pattern:	Evenfrm	Max total plant:	6.7226
	Evenbar5		6.8157
	Allb1		6.8268
	Ckr5		6.8898

IV.B. Ways Project Could Be Furthered

There is still a great deal of work that can be done with this project. One could add a factor controlling the maximum density of plant and pathogen growth to the equations, like the Lotka-Volterra model. Pathogen does not die out in this model; a term could be added to the pathogen equation for pathogen death. The boundary conditions for the pathogen could be made time dependent. The coefficients for pathogen growth and plant decay could be made different, instead of using the same one (α_1 , α_2) for both. Changing the flux pattern in the striped pattern case to be perpendicular to the stripes (rather than just all four sides) would be an interesting case to run. Certainly more patterns could be tried with this model. The formulae could be adjusted to allow the pathogen to flow diagonally; as it is now, pathogen would be unable to flow into the middle of the checkerboard pattern if the pathogen coefficient for either plant is set to 0.0. Some of the most difficult questions are how to quantitatively measure the amount of variation in the field, or how to maximize the harvest in the shortest amount of time. Much of the model used in this simulation is assumed to be under ideal conditions. These ideas are just a few of the ways this project could be furthered.

REFERENCES

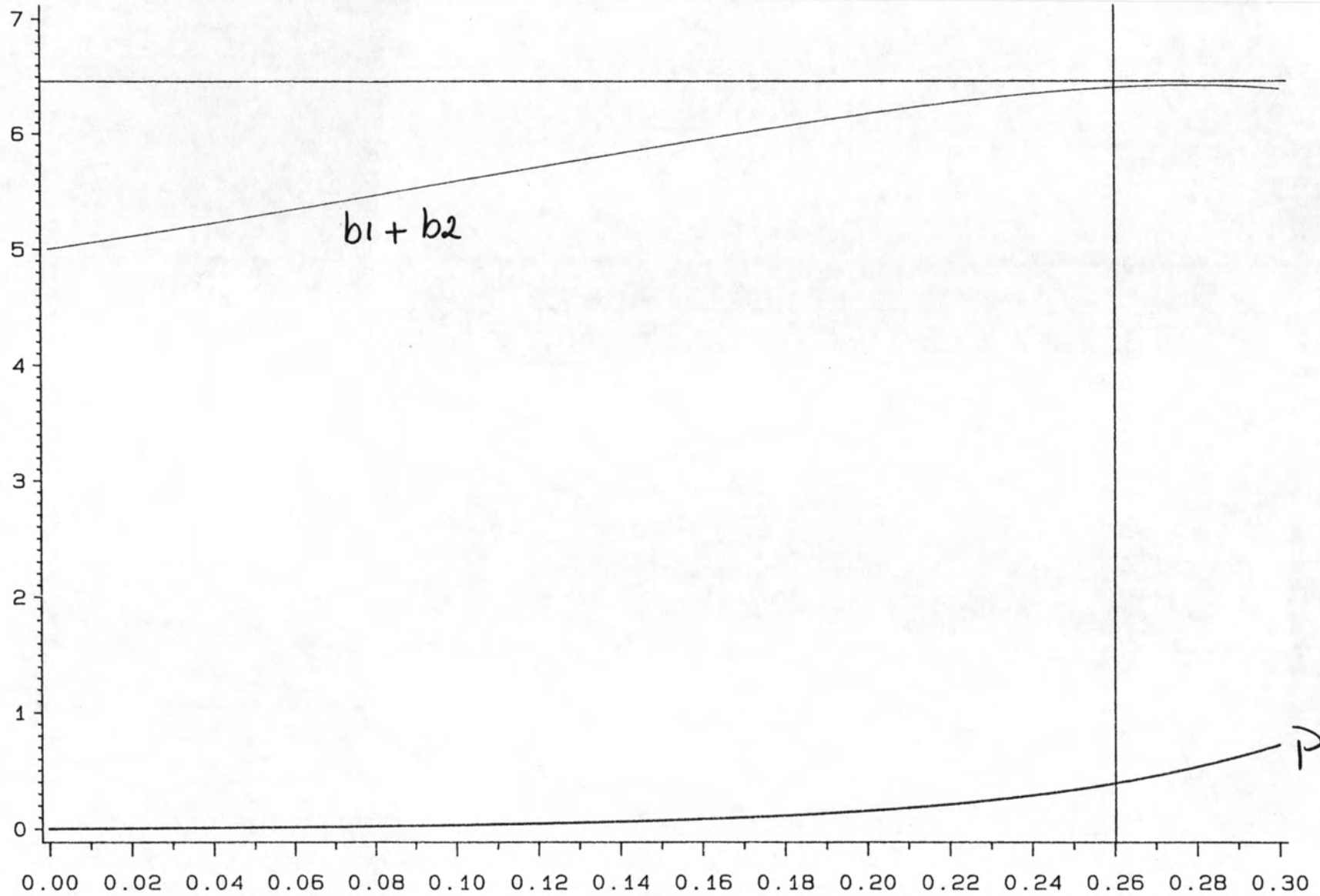
1. Ames, W., Numerical Methods for Partial Differential Equations, 2nd. Edition, Academic Press, New York, 1977
2. Bullock, M. K., A Spatial Simulation Model for Disease Spread in a Crop, M. S. thesis, University of Tennessee, Knoxville, 1986.
3. Vandermeer, J., The Ecology of Intercropping, Cambridge University Press, New York, 1989.

APPENDICES

Attached are some graphs of total biomass and pathogen versus time. The horizontal reference line shows the maximum amount of total plant mass attained by the checkerboard pattern for a given a_1 , a_2 , α_1 , α_2 combination, and the vertical reference line shows the time at which that maximum was reached. 'EVENBAR5' is a pattern of five stripes of alternating plant type, the buffering plant on the outside stripes and in the middle of the field, the fast-growing plant in two stripes inside the field. 'EVENFRM' is a frame pattern with the buffering plant surrounding a patch of fast-growing plant on all sides. 'CKR5' refers to a 5x5 checkerboard. 'ALLB1' is a field containing nothing but the fast-growing plant (no buffering). Also attached are copies of the two programs two programs which were used, the first one employing a finite differencing scheme, the second one an implicit scheme.

A1=2 A2=0.5 ALPHA1=4 ALPHA2=0.5 NAME=EVENBAR5

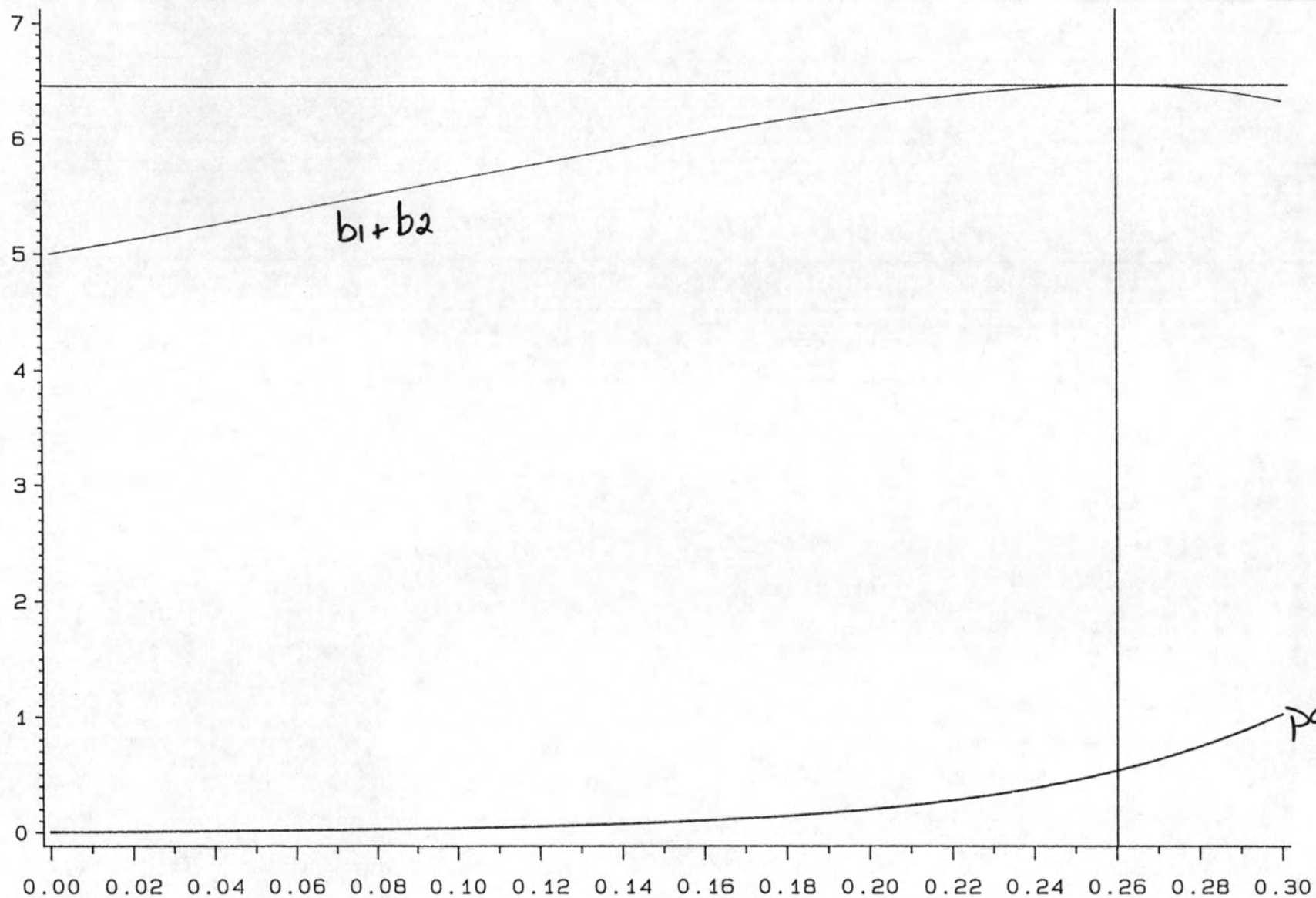
B1B2



TIME

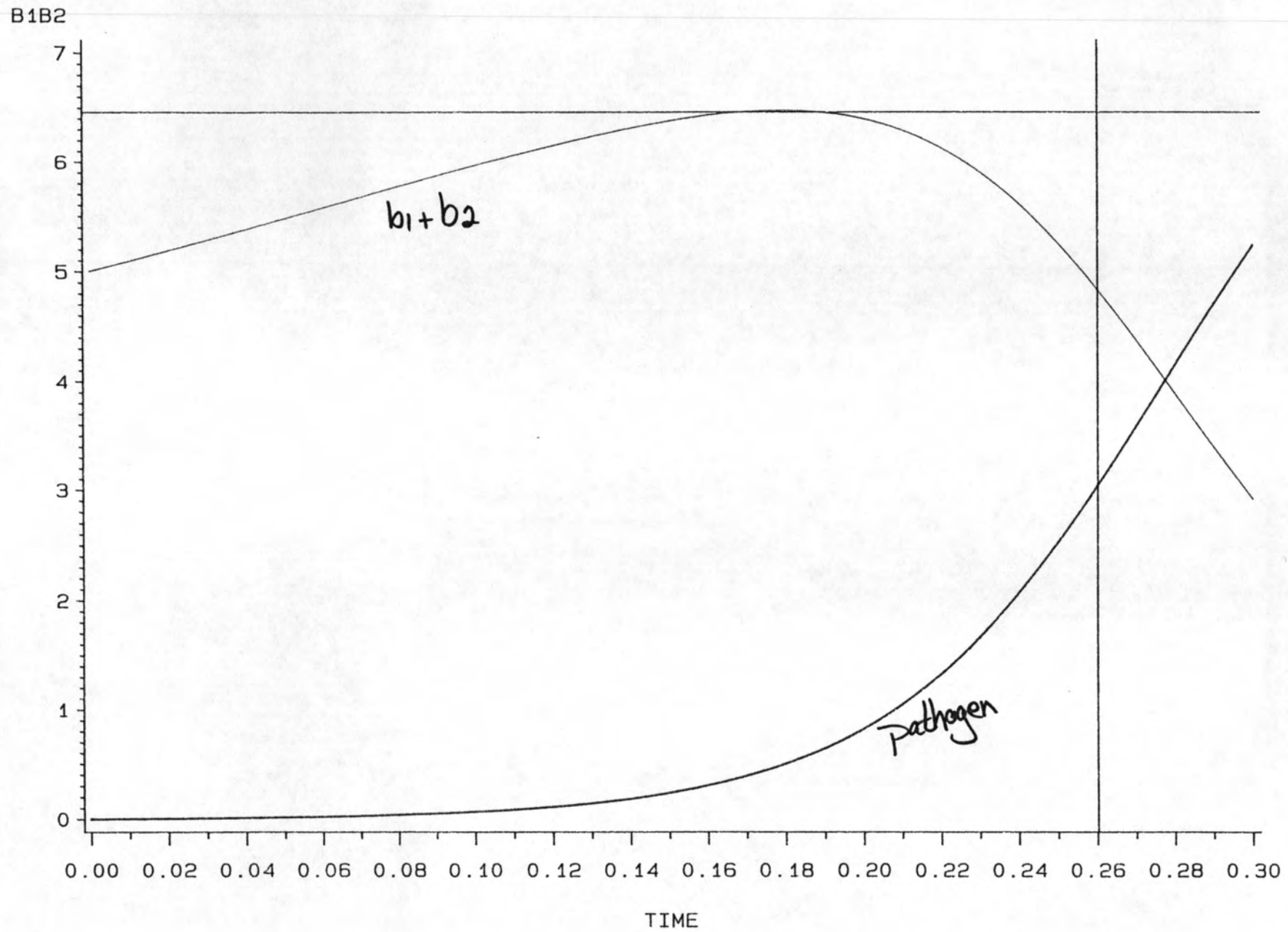
A1=2 A2=0.5 ALPHA1=4 ALPHA2=0.5 NAME=CKR5

B1B2

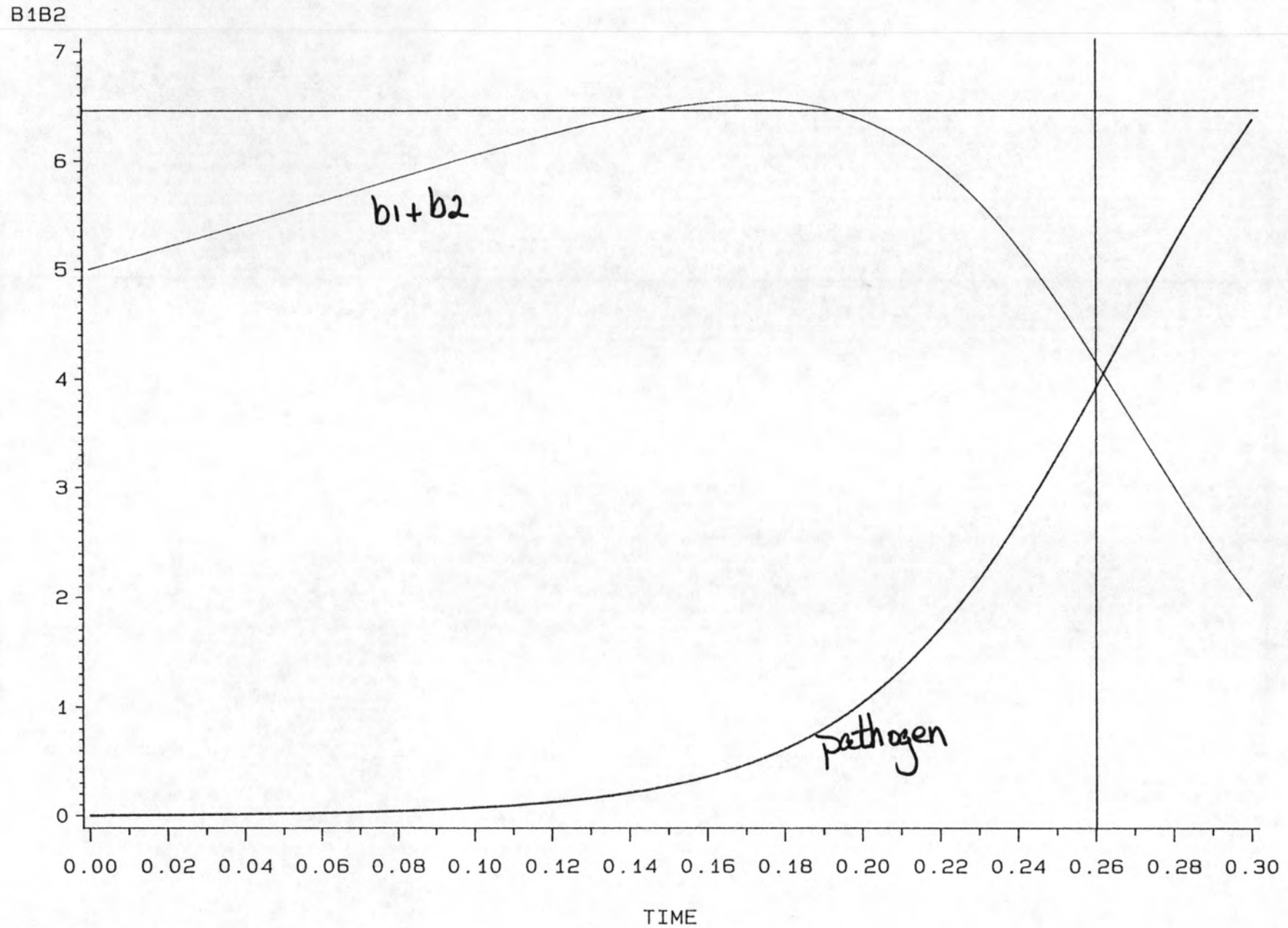


TIME

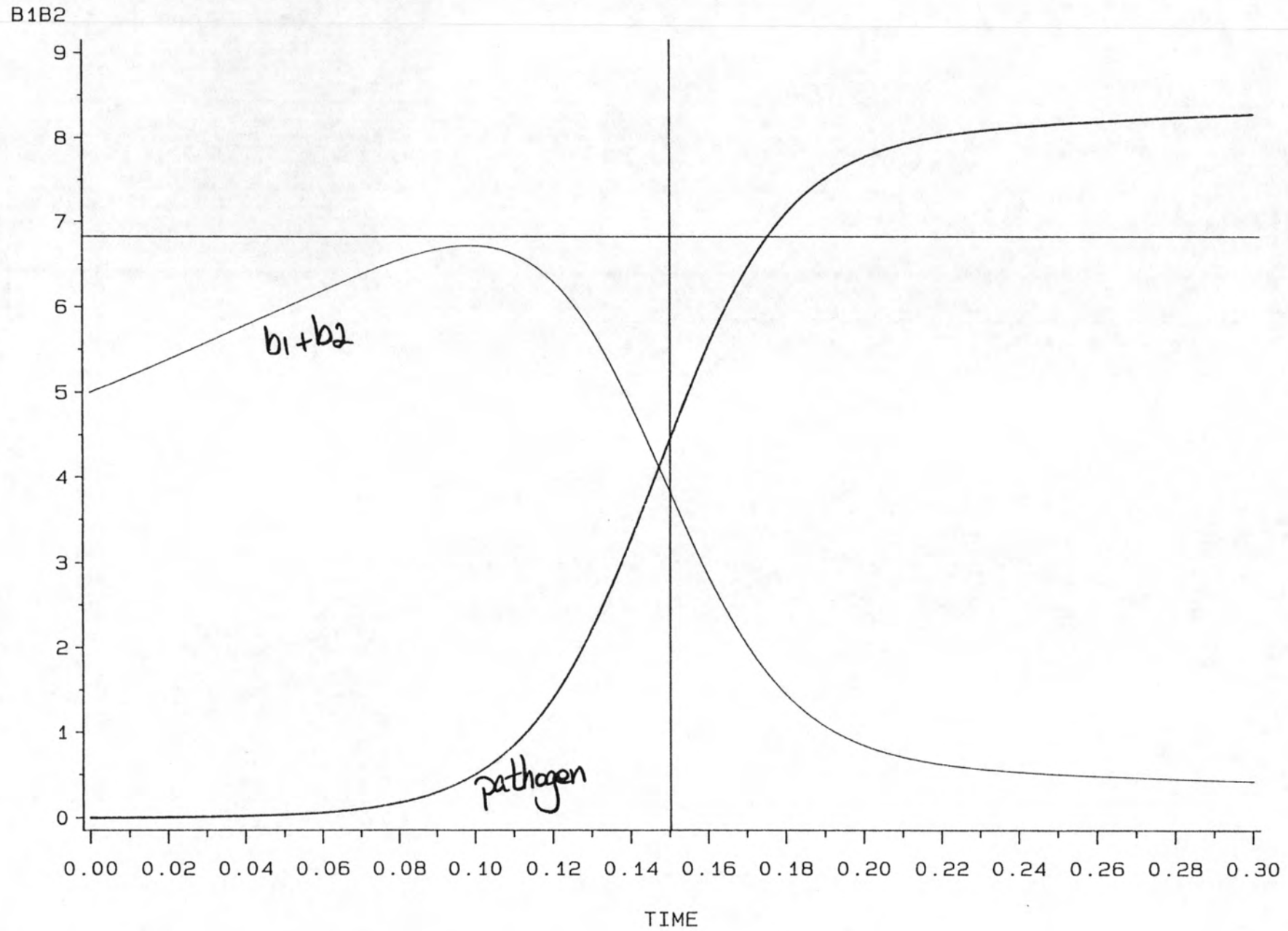
A1=2 A2=0.5 ALPHA1=4 ALPHA2=0.5 NAME=EVENTFRM



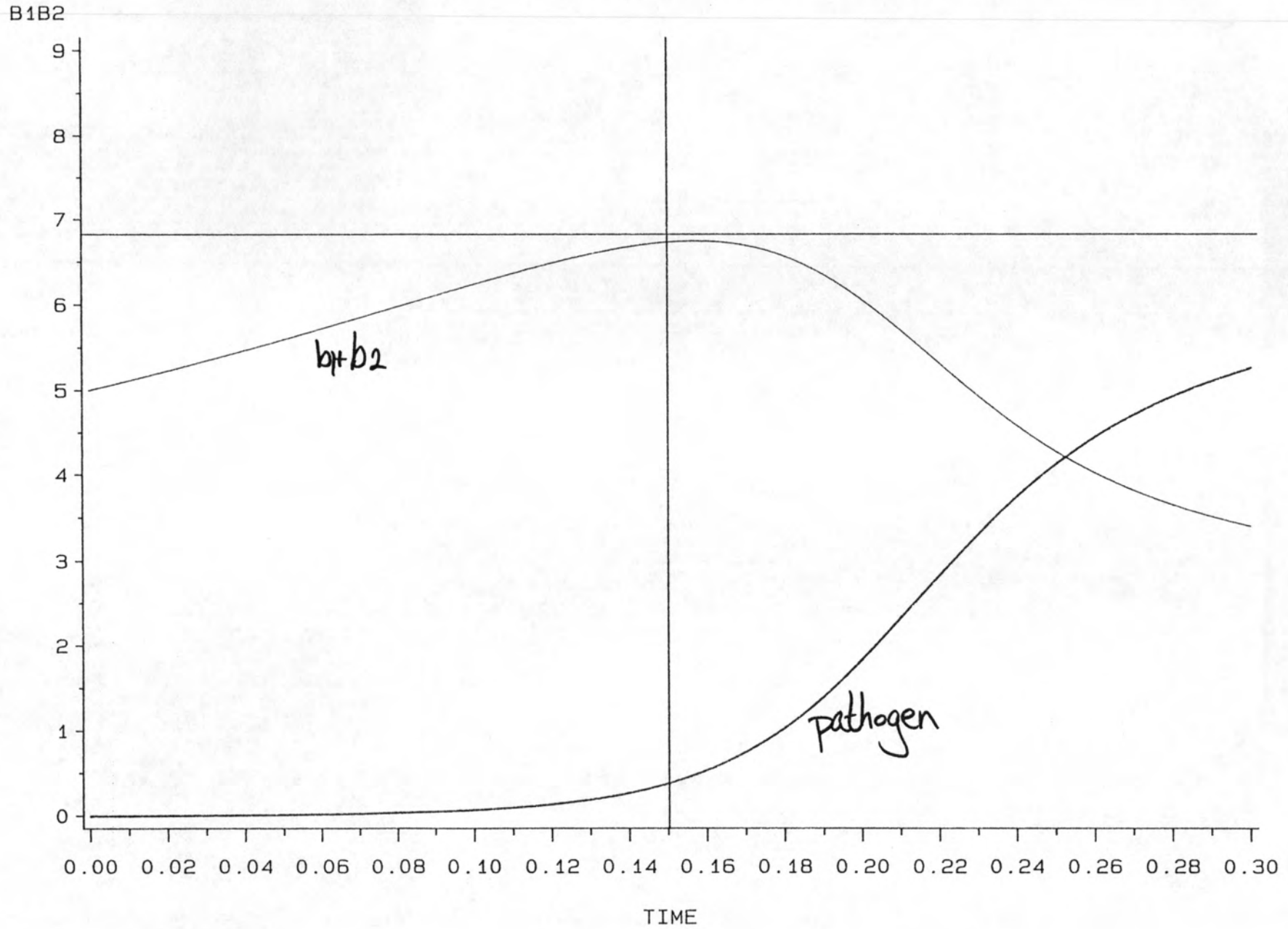
A1=2 A2=0.5 ALPHA1=4 ALPHA2=0.5 NAME=ALLB1



A1=4 A2=1 ALPHA1=8 ALPHA2=0.5 NAME=EVENTFRM

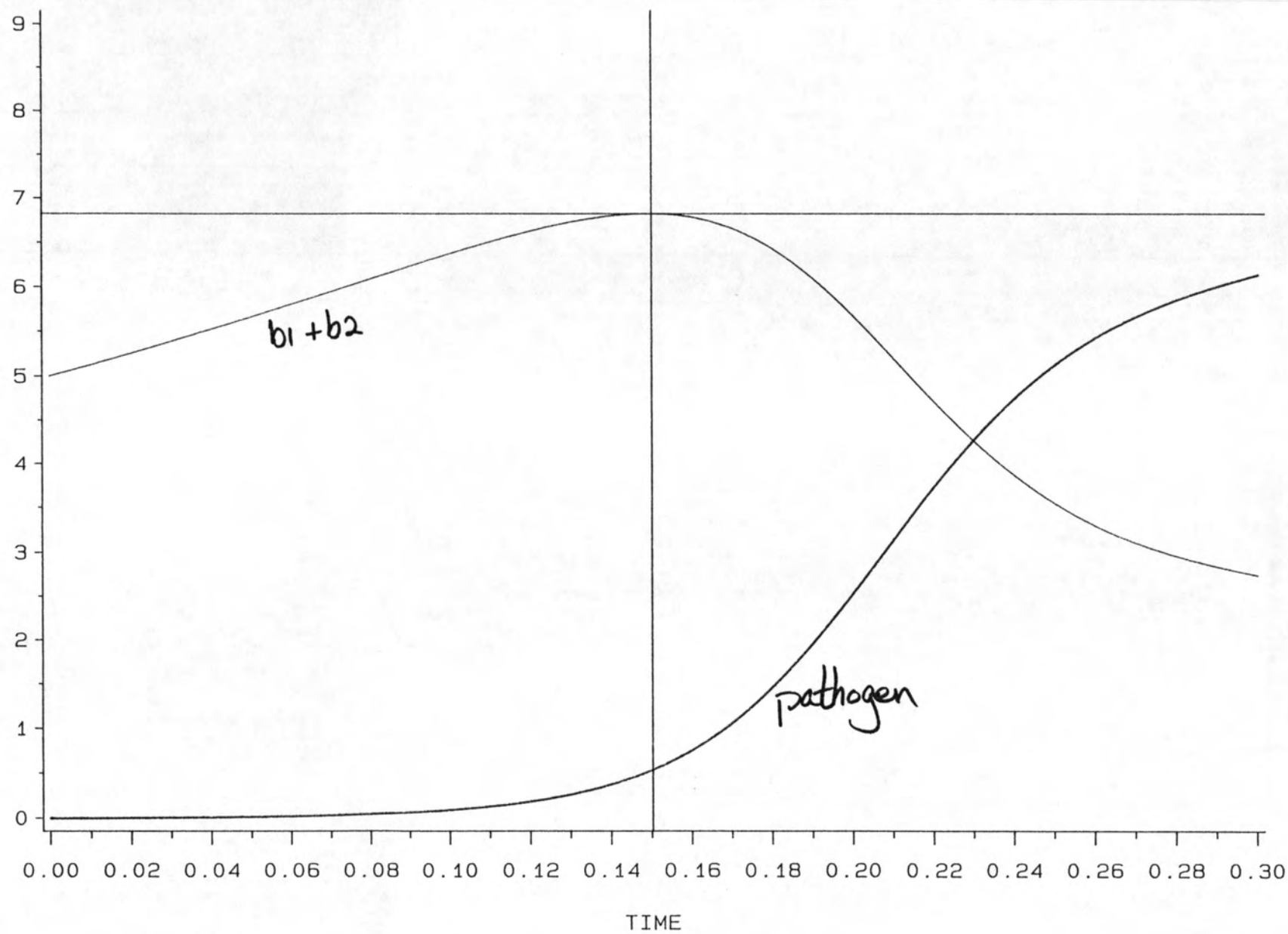


A1=4 A2=1 ALPHA1=8 ALPHA2=0.5 NAME=EVENBAR5



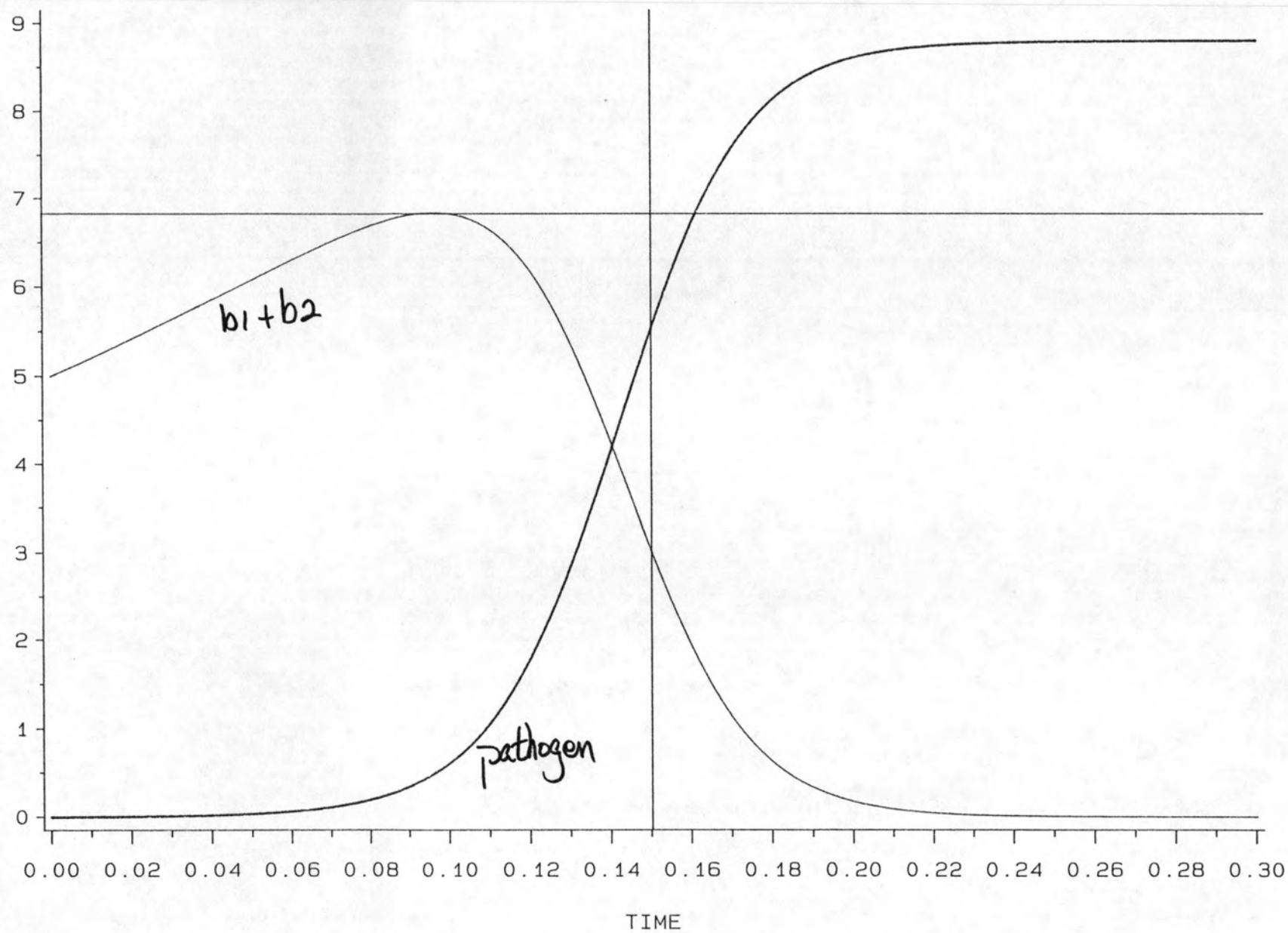
A1=4 A2=1 ALPHA1=8 ALPHA2=0.5 NAME=CKR5

B1B2

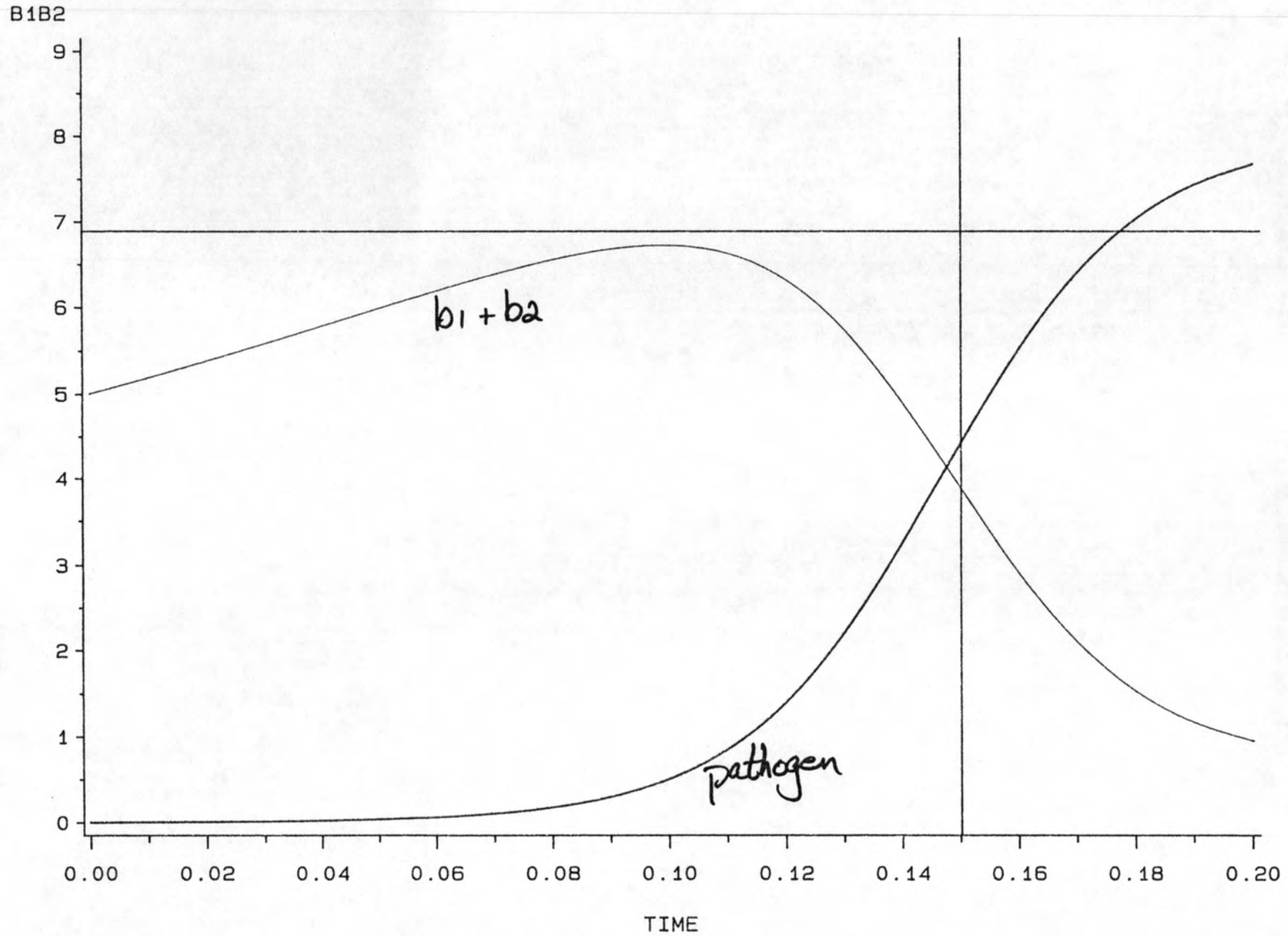


A1=4 A2=1 ALPHA1=8 ALPHA2=0.5 NAME=ALLB1

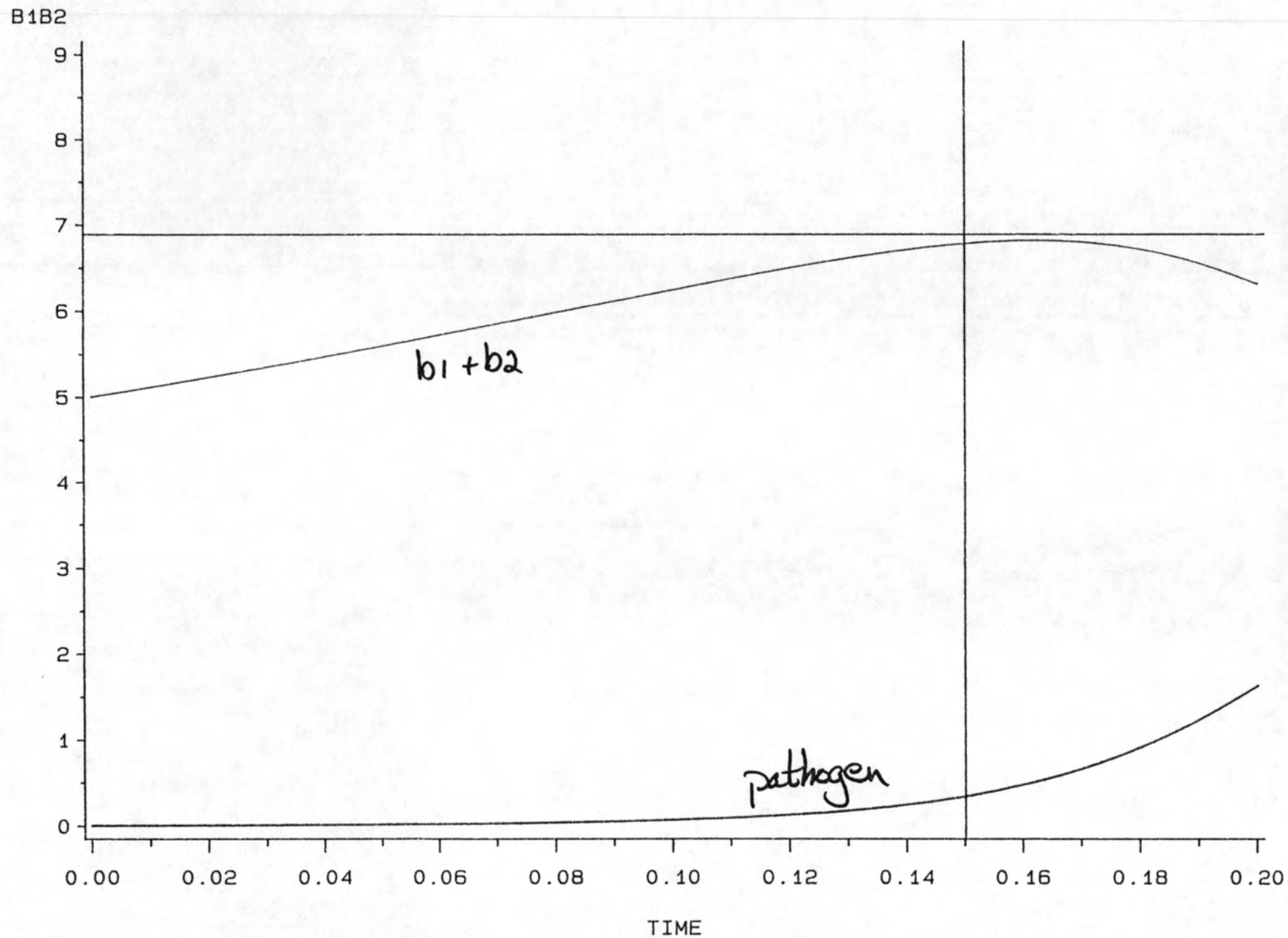
B1B2



A1=4 A2=1 ALPHA1=8 ALPHA2=0.0625 NAME=EVENFRM

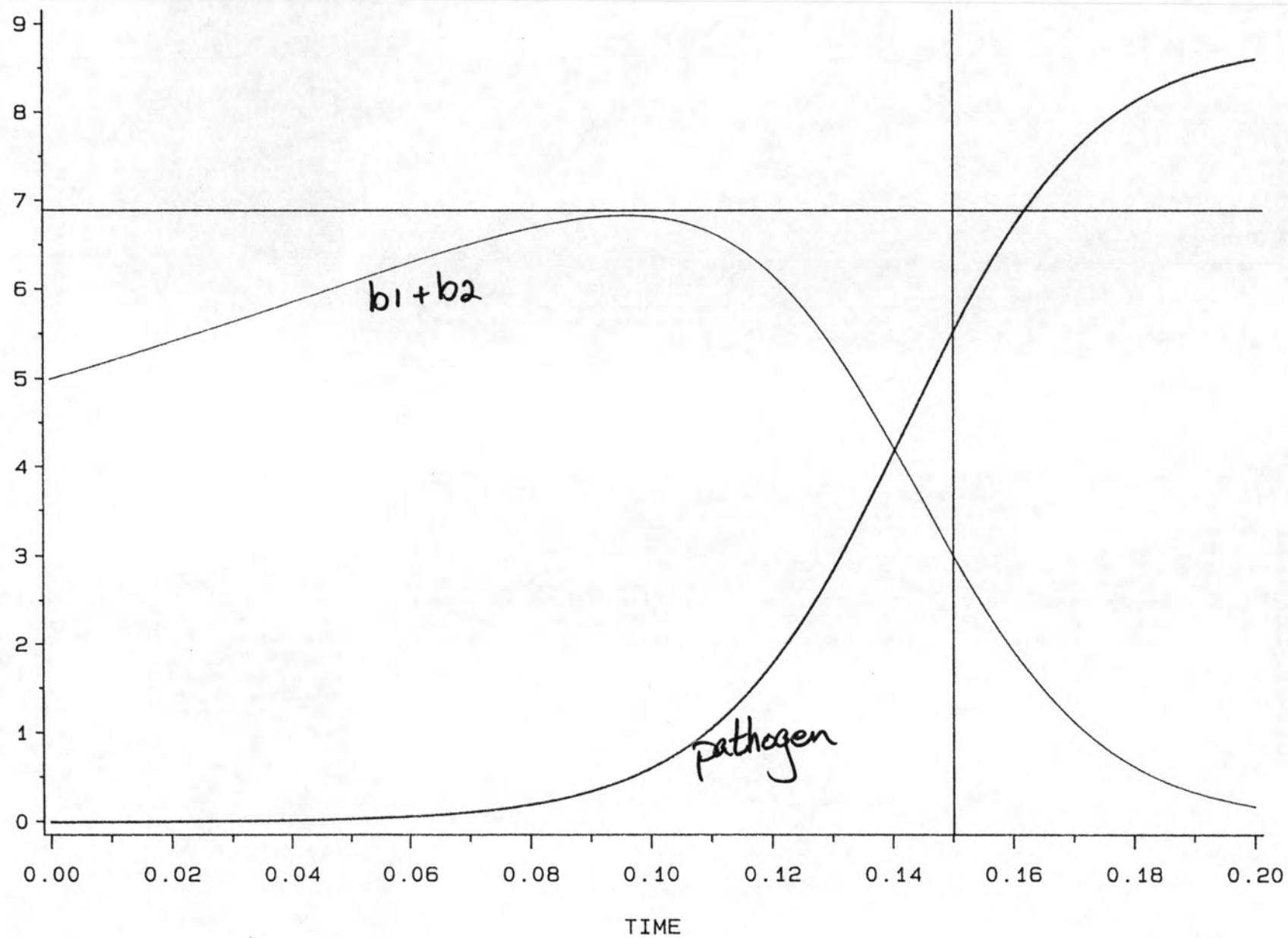


A1=4 A2=1 ALPHA1=8 ALPHA2=0.0625 NAME=EVENBAR5

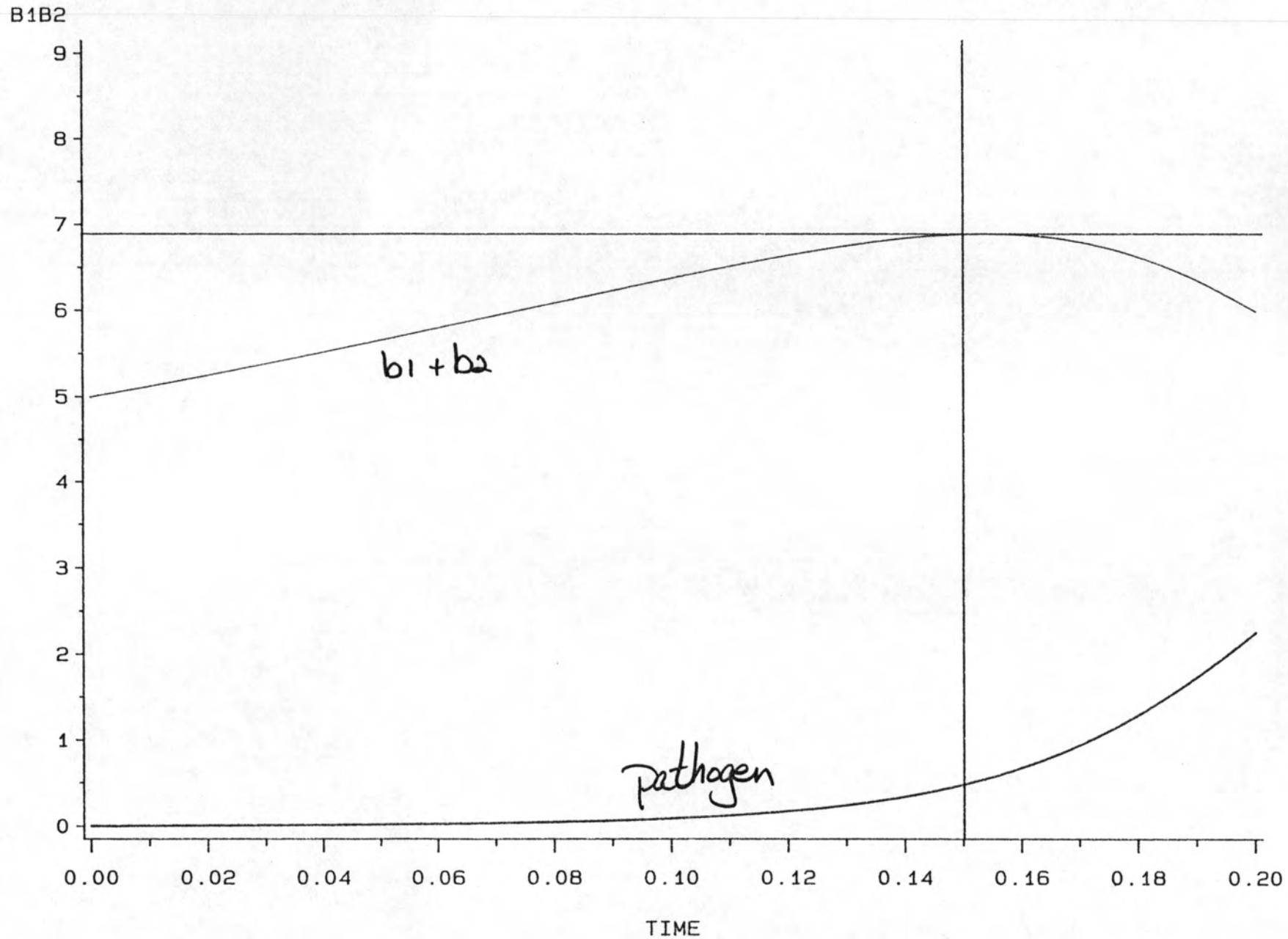


A1=4 A2=1 ALPHA1=8 ALPHA2=0.0625 NAME=ALLB1

B1B2



A1=4 A2=1 ALPHA1=8 ALPHA2=0.0625 NAME=CKR5



```

0001 *****
0002 *
0003 * PROGRAM: FINITEDIFF.FOR
0004 * AUTHOR: JIM TILSON
0005 * DATE: MAY 2, 1990
0006 * CLASS: TENNESSEE SCHOLARS, MATH 490
0007 *
0008 *****
0009 *
0010 * THIS PROGRAM SIMULATES THE GROWTH OF TWO KINDS OF PLANT IN A FIELD WHERE
0011 * A PATHOGEN IS PRESENT. GROWTH RATES OF THE PLANTS AND OF THE PATHOGEN
0012 * ARE CONTROLLED BY COEFFICIENTS ENTERED BY THE USER. DATA FOR THE INITIAL
0013 * FIELD PLANTING PATTERN ARE READ FROM A FILE SPECIFIED BY THE USER AND
0014 * RESULTS CONSISTING OF DATA FROM EACH TIME STEP ARE SENT TO AN OUTPUT FILE
0015 * ALSO SPECIFIED BY THE USER, IN A FORMAT READY FOR DATA ANALYSIS USING SAS.
0016 *
0017 * THE PROGRAM USES A FINITE DIFFERENCING SCHEME TO CALCULATE DATA FOR THE
0018 * N+1TH TIME STEP USING DATA FROM THE NTH TIME STEP.
0019 *
0020 *****
0021 *
0022 * INPUT: INPUT CONSISTS OF PARAMETERS, INPUT BY THE USER, AND A DATA FILE
0023 * (ALSO SPECIFIED BY THE USER) CONTAINING THE INFORMATION FOR THE
0024 * PLANTING PATTERN.
0025 *
0026 * OUTPUT: OUTPUT IS SENT TO A FILE AT EACH TIME STEP. IT IS FORMATTED FOR
0027 * USE BY SAS FOR ANY KIND OF STATISTICAL ANALYSIS WHICH THE USER
0028 * WOULD LIKE TO USE.
0029 *
0030 *****
0031 * FINITEDIFF--MAIN PROGRAM
0032 * IMPLICIT NONE
0033
0034
0035 * VARIABLE DECLARATIONS
0036
0037 REAL*8 LT ! LOOP VARIABLE
0038 REAL*8 TMAX ! MAXIMUM TIME
0039 REAL*8 DX, DY, DT ! CHANGE IN X, Y, TIME
0040 REAL*8 A1, A2, ALPHA1, ALPHA2 ! EQUATION COEFFS.
0041 REAL*8 B1(2, 30, 30) ! ARRAY FOR PLANT 1
0042 REAL*8 B2(2, 30, 30) ! ARRAY FOR PLANT 2
0043 REAL*8 P(2, 30, 30) ! ARRAY FOR PATHOGEN
0044 REAL*8 A, B, C, D ! PARTIAL EQN. PARAMS.
0045 REAL*8 TBOUND, BBOUND ! BOUNDARY FUNCTIONS
0046 REAL*8 LBOUND, RBOUND ! BOUNDARY FUNCTIONS
0047 REAL*8 X, Y ! COORDS. OF GRID SQUARE CNTR.
0048 INTEGER*4 LX, LY ! LOOP VARIABLES FOR GRID
0049 INTEGER*4 NOPTS ! NUMBER OF POINTS IN GRID
0050 INTEGER*4 OLD ! USED TO DENOTE OLD TIME STEP INFO IN GRIDS
0051 CHARACTER*10 OFNAME ! OUTPUT FILE NAME
0052 CHARACTER*10 IFNAME ! INPUT FILE NAME
0053
0054
0055 * INPUT PARAMETERS
0056 PRINT *, 'ENTER INPUT FILE NAME'
0057 READ *, IFNAME

```

```
0058      PRINT *, 'ENTER OUTPUT FILE NAME'
0059      READ *, OFNAME
0060      PRINT *, 'ENTER TMAX'
0061      READ *, TMAX
0062      PRINT *, 'ENTER A1, A2'
0063      READ *, A1, A2
0064      PRINT *, 'ENTER ALPHA1, ALPHA2'
0065      READ *, ALPHA1, ALPHA2
0066      PRINT *, 'ENTER TBOUND, BBOUND, RBOUND, LBOUND'
0067      READ *, TBOUND, BBOUND, RBOUND, LBOUND
0068
0069
0070      *      OPEN OUTPUT FILE
0071      OPEN(12, FILE=OFNAME, STATUS='NEW')
0072
0073
0074      *      SET VALUES
0075      DX = 1.0 / NO_PTS
0076      DY = 1.0 / NO_PTS
0077      DT = 0.48 * ((DX**2 * DY**2) / (DX**2 + DY**2))
0078      OLD = 1
0079
0080
0081      *      INITIALIZE GRID LOCATIONS
0082      CALL INITIALIZE_GRID(B1, B2, P, NO_PTS, OLD, IFNAME)
0083
0084
0085      *      TIME PASSES...
0086      DO LT = 0.0, TMAX, DT
0087          DO LY = 1, NO_PTS
0088              DO LX = 1, NO_PTS
0089                  *      OUTPUT CURRENT FIELD INFORMATION
0090                  CALL OUTPUT(B1,B2,P,OLD,OFNAME,IFNAME,NO_PTS,A1,A2,ALPHA1,ALPHA2,LT,DX,DY)
0091
0092
0093      *      CALCULATE PARAMETERS
0094      IF (LX .EQ. NO_PTS) THEN
0095          A = RBOUND
0096      ELSE
0097          A = (P(OLD, LX+1, LY) - P(OLD, LX, LY)) / DX
0098      END IF
0099
0100      IF (LX .EQ. 1) THEN
0101          B = LBOUND
0102      ELSE
0103          B = (P(OLD, LX-1, LY) - P(OLD, LX, LY)) / DX
0104      END IF
0105
0106      IF (LY .EQ. NO_PTS) THEN
0107          C = TBOUND
0108      ELSE
0109          C = (P(OLD, LX, LY+1) - P(OLD, LX, LY)) / DY
0110      END IF
0111
0112      IF (LY .EQ. 1) THEN
0113          D = BBOUND
0114      ELSE
```

```

0115             D = (P(OLD, LX, LY-1) - P(OLD, LX, LY)) / DY
0116             END IF
0117
0118
0119 *             CALCULATE NEW GRID VALUES
0120             P(3-OLD,LX,LY) = P(OLD,LX,LY) + DT * ((A+B)/DX + (C+D)/DY
0121             1             + ALPHA1*B1(OLD,LX,LY)*P(OLD,LX,LY) + ALPHA2*B2(OLD,LX,LY)*P(OLD,LX,LY))
0122             B1(3-OLD,LX,LY) = B1(OLD,LX,LY) + DT*(A1*B1(OLD,LX,LY) - ALPHA1*B1(OLD,LX,LY)*P(OLD,LX,LY))
0123             B2(3-OLD,LX,LY) = B2(OLD,LX,LY) + DT*(A2*B2(OLD,LX,LY) - ALPHA2*B2(OLD,LX,LY)*P(OLD,LX,LY))
0124             END DO
0125             END DO
0126             OLD = 3 - OLD
0127             END DO
0128
0129
0130 *             ITERATIONS DONE--OUTPUT FINAL STATE OF FIELD
0131             CALL OUTPUT(B1,B2,P,OLD,IFNAME,OFNAME,NO_PTS,A1,A2,ALPHA1,ALPHA2,LT,DX,DY)
0132
0133
0134 *             CLOSE FILE AND TERMINATE PROGRAM
0135             CLOSE(12)
0136             END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	1386	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	121	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	43656	PIC CON REL LCL NOSHR NOEXE RD WRT QUAD
Total Space Allocated	45163	

ENTRY POINTS

Address	Type	Name
0-00000000		FINITEDIFF\$MAIN

VARIABLES

[illegible]

2-May-1990 01:45:21
2-May-1990 01:45:07

VAX FORTRAN V5.3-50
SYS\$USER6:[JTILSON.MATH490]FINITEDIFF.FOR;7

Page 5

```
0001 *****
0002 *
0003 * SUBROUTINE INITIALIZE:
0004 *
0005 * THIS SUBROUTINE TAKES ITS INPUT FROM A FILE SPECIFIED BY THE USER. THIS
0006 * INPUT IS THE INITIAL PLANTING PATTERN FOR THE FIELD, CONSISTING OF:
0007 *
0008 * LINE 1: NUMBER OF POINTS IN GRID (30 MAX.)
0009 * SUBSEQUENT LINES: X AND Y RANGES, CONCENTRATIONS OF PLANTS AND PATHOGEN
0010 * IN RECTANGULAR BOX FORMED BY X AND Y COORDINATES
0011 * LAST LINE: CONTAINS SENTINEL 999 999 999 999 999 999 999 TO DENOTE
0012 * END OF FILE
0013 *
0014 *****
0015
0016 SUBROUTINE INITIALIZE_GRID(B1, B2, P, NO_PTS, OLD, IFNAME)
0017 *
0018 * PARAMETERS
0019 *
0020 REAL*8 B1(2,30,30), B2(2,30,30), P(2,30,30) ! PLANT AND PATHOGEN DATA
0021 INTEGER*4 OLD ! SIZE OF GRID
0022 INTEGER*4 NO_PTS ! NUMBER OF POINTS IN GRID
0023 CHARACTER*10 IFNAME ! INPUT FILE NAME
0024 *
0025 *
0026 * LOCAL VARIABLES
0027 *
0028 INTEGER*2 X, Y ! LOOP VARIABLES
0029 INTEGER*2 XMN, XMX, YMN, YMX ! X AND Y BOUNDS FOR PLANT AND PATHOGEN DATA
0030 REAL*8 B1VAL, B2VAL, PVAL ! PLANT AND PATHOGEN DATA
0031 *
0032 *
0033 * OPEN FILE
0034 OPEN (13, FILE=IFNAME, STATUS='OLD')
0035 *
0036 *
0037 * PRIMING READ
0038 READ (13,*) NO_PTS
0039 READ (13,*) XMN, XMX, YMN, YMX, B1VAL, B2VAL, PVAL
0040 *
0041 *
0042 * MAIN LOOP
0043 DO WHILE (XMN .NE. 999)
0044 DO X=XMN, XMX
0045 DO Y=YMN, YMX
0046 B1(OLD,X,Y) = B1VAL
0047 B2(OLD,X,Y) = B2VAL
0048 P(OLD,X,Y) = PVAL
0049 END DO
0050 END DO
0051 READ (13,*) XMN, XMX, YMN, YMX, B1VAL, B2VAL, PVAL
0052 END DO
0053 *
0054 *
0055 * CLOSE FILE, RETURN TO MAIN PROGRAM
0056 CLOSE(13)
0057 END
```

INITIALIZE_GRID
01

2-May-1990 01:45:21
2-May-1990 01:45:07

PROGRAM SECTIONS

Name	Bytes	Attributes									
0 \$CODE	419	PIC	CON	REL	LCL	SHR	EXE	RD	NOWRT	LONG	
2 \$LOCAL	140	PIC	CON	REL	LCL	NOSHR	NOEXE	RD	WRT	QUAD	
Total Space Allocated		559									

ENTRY POINTS

Address	Type	Name
0-00000000		INITIALIZE_GRID

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name	Address	Type	Name
2-00000010	R*8	B1VAL	2-00000018	R*8	B2VAL	AP-00000018@	CHAR	IFNAME	AP-00000010@	I*4	NO_PTS
AP-00000014@	I*4	OLD	2-00000020	R*8	PVAL	2-00000000	I*2	X	2-00000004	I*2	XMN
2-00000006	I*2	XMN	2-00000002	I*2	Y	2-00000008	I*2	YMN	2-0000000A	I*2	YMX

ARRAYS

Address	Type	Name	Bytes	Dimensions
AP-00000004@	R*8	B1	14400	(2, 30, 30)
AP-00000008@	R*8	B2	14400	(2, 30, 30)
AP-0000000C@	R*8	P	14400	(2, 30, 30)

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	FOR\$CLOSE		FOR\$OPEN

2-May-1990 01:45:21
2-May-1990 01:45:07

VAX FORTRAN V5.3-50
SYS\$USER6:[JTILSON.MATH490]FINITEDIFF.FOR;7

Page 7

```
0001 *****
0002 *
0003 * SUBROUTINE OUTPUT:
0004 *
0005 * THIS SUBROUTINE SENDS DATA FROM THE CURRENT TIME STEP TO THE FILE SPEC-
0006 * IFIED BY OFNAME.
0007 *
0008 *****
0009 SUBROUTINE OUTPUT(B1,B2,P,OLD,OFNAME,IFNAME,NO_PTS,A1,A2,ALPHA1,ALPHA2,LT,DX,DY)
0010 *
0011 * PARAMETERS
0012 *
0013 REAL*8 B1(2,30,30), B2(2,30,30), P(2,30,30) ! PLANT AND PATHOGEN DATA
0014 INTEGER*4 OLD ! POINTER TO OLD GUESS
0015 CHARACTER*10 OFNAME, IFNAME ! FILE NAMES
0016 INTEGER*4 NO_PTS ! NUMBER OF POINTS IN GRID
0017 REAL*8 A1, A2, ALPHA1, ALPHA2 ! GROWTH CONSTANTS
0018 REAL*8 LT ! CURRENT TIME STEP
0019 REAL*8 DX, DY ! GRID SQUARE DIMENSIONS
0020
0021 *
0022 * LOCAL VARIABLES
0023 *
0024 INTEGER*4 LX, LY ! LOOP VARIABLES
0025 REAL*8 SUMB1, SUMB2, SUMP, SUMB1B2 ! SUM VARIABLES
0026
0027 *
0028 CLEAR SUM VARIABLES
0029 SUMB1 = 0.0
0030 SUMB2 = 0.0
0031 SUMP = 0.0
0032
0033 *
0034 SUM UP CONCENTRATIONS OF PLANT, PATHOGEN DATA
0035 DO LX = 1,NO_PTS
0036 DO LY = 1,NO_PTS
0037 SUMB1 = SUMB1 + B1(OLD,LX,LY)
0038 SUMB2 = SUMB2 + B2(OLD,LX,LY)
0039 SUMP = SUMP + P(OLD,LX,LY)
0040 SUMB1B2 = SUMB1 + SUMB2
0041 END DO
0042 END DO
0043
0044 *
0045 MULTIPLY DENSITY BY SQUARE SIZE GIVING TOTAL MASS
0046 SUMB1 = SUMB1 * (DX*DY)
0047 SUMB2 = SUMB2 * (DX*DY)
0048 SUMP = SUMP * (DX*DY)
0049 SUMB1B2 = SUMB1B2 * (DX*DY)
0050
0051 *
0052 OUTPUT TO FILE
0053 WRITE (12, 50) IFNAME, NO_PTS, A1, A2, ALPHA1, ALPHA2, LT, SUMB1, SUMB2, SUMP, SUMB1B2
0054 50 FORMAT (' ', A8, X, I2, X, 9(F8.5, X))
0055
0056 *
0057 RETURN TO MAIN PROGRAM
```

0058 END

PROGRAM SECTIONS

Name	Bytes	Attributes									
0 \$CODE	369	PIC	CON	REL	LCL	SHR	EXE	RD	NOWRT	LONG	
1 \$PDATA	22	PIC	CON	REL	LCL	SHR	NOEXE	RD	NOWRT	LONG	
2 \$LOCAL	124	PIC	CON	REL	LCL	NOSHR	NOEXE	RD	WRT	QUAD	
Total Space Allocated		515									

ENTRY POINTS

Address	Type	Name
0-00000000		OUTPUT

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name	Address	Type	Name
AP-00000020@	R*8	A1	AP-00000024@	R*8	A2	AP-00000028@	R*8	ALPHA1	AP-0000002C@	R*8	ALPHA2
AP-00000034@	R*8	DX	AP-00000038@	R*8	DY	AP-00000018@	CHAR	IFNAME	AP-00000030@	R*8	LT
2-00000020	I*4	LX	2-00000024	I*4	LY	AP-0000001C@	I*4	NO_PTS	AP-00000014@	CHAR	OFNAME
AP-00000010@	I*4	OLD	2-00000000	R*8	SUMB1	2-00000018	R*8	SUMB1B2	2-00000008	R*8	SUMB2
2-00000010	R*8	SUMP									

ARRAYS

Address	Type	Name	Bytes	Dimensions
AP-00000004@	R*8	B1	14400	(2, 30, 30)
AP-00000008@	R*8	B2	14400	(2, 30, 30)
AP-0000000C@	R*8	P	14400	(2, 30, 30)

LABELS

Address	Label
1-00000000	50'

OUTPUT
01

2-May-1990 01:45:21
2-May-1990 01:45:07

VAX FORTRAN V5.3-50
SYS\$USER6:[JTILSON.MATH490]FINITEDIFF.FOR;7

Page 9

COMMAND QUALIFIERS

FORTRAN/NOOP/EXTEND_SOURCE/LIST FINITEDIFF.FOR

/CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/SHOW=(NODICTIONARY,NOINCLUDE,MAP,NOPREPROCESSOR,SINGLE)
/STANDARD=(NOSEMANTIC,NOSOURCE_FORM,NOSYNTAX)
/WARNINGS=(NODECLARATIONS,GENERAL,NOULTRIX,NOVAXELN)
/CONTINUATIONS=19 /NOCROSS_REFERENCE /NOD_LINES /EXTEND_SOURCE
/F77 /NOG_FLOATING /14 /NOMACHINE_CODE /NOOPTIMIZE /NOPARALLEL
/NOANALYSIS_DATA
/NODIAGNOSTICS
/LIST=SYS\$USER6:[JTILSON.MATH490]FINITEDIFF.LIS;2
/OBJECT=SYS\$USER6:[JTILSON.MATH490]FINITEDIFF.OBJ;1

COMPILATION STATISTICS

Run Time:	0.91 seconds
Elapsed Time:	4.42 seconds
Page Faults:	646
Dynamic Memory:	398 pages

```
0001 *****
0002 *
0003 * PROGRAM: IMPLICIT.FOR
0004 * AUTHOR: JIM TILSON
0005 * DATE: MAY 2, 1990
0006 * CLASS: TENNESSEE SCHOLARS, MATH 490
0007 *
0008 * THIS PROGRAM SIMULATES THE EFFECTS OF A PATHOGEN ON A FIELD USED TO GROW
0009 * TWO KINDS OF PLANTS, EACH WITH DIFFERENT GROWTH RATES AND DIFFERENT
0010 * LEVELS OF RESISTANCE TO THE PATHOGEN. IT USES SUCCESSIVE OVER-RELAXATION
0011 * AND CRANK-NICOLSON'S FORMULA TO CALCULATE THE NEW CONCENTRATIONS OF PLANT
0012 * AND PATHOGEN AT EACH TIME STEP.
0013 *
0014 *****
0015 *
0016 * INPUT: INPUT CONSISTS OF PARAMETERS, INPUT BY THE USER, AND A DATA FILE
0017 * (ALSO SPECIFIED BY THE USER) CONTAINING THE INFORMATION FOR THE
0018 * PLANTING PATTERN.
0019 *
0020 * OUTPUT: OUTPUT IS SENT TO A FILE AT EACH TIME STEP. IT IS FORMATTED FOR
0021 * USE BY SAS FOR ANY KIND OF STATISTICAL ANALYSIS WHICH THE USER
0022 * WOULD LIKE TO USE.
0023 *
0024 *****
0025 *
0026 * IMPLICIT--MAIN PROGRAM
0027 * IMPLICIT NONE
0028 *
0029 *
0030 * VARIABLE DECLARATIONS
0031 *
0032 * REAL*8 LT ! LOOP VARIABLE
0033 * REAL*8 TMAX ! MAXIMUM TIME
0034 * REAL*8 DX, DY, DT ! CHANGE IN X, Y, TIME
0035 * REAL*8 A1, A2, ALPHA1, ALPHA2 ! EQUATION COEFFS.
0036 * REAL*8 B1(30, 30) ! ARRAY FOR PLANT 1
0037 * REAL*8 B2(30, 30) ! ARRAY FOR PLANT 2
0038 * REAL*8 P(30, 30) ! ARRAY FOR PATHOGEN
0039 * REAL*8 TBOUND, BBOUND ! BOUNDARY FUNCTIONS
0040 * REAL*8 LBOUND, RBOUND ! BOUNDARY FUNCTIONS
0041 * REAL*8 X, Y ! COORDS. OF GRID SQUARE CNTR.
0042 * INTEGER*4 LX, LY ! LOOP VARIABLES
0043 * INTEGER*4 NOPTS ! NUMBER OF GRID SQUARES TO A SIDE
0044 * CHARACTER*10 OFNAME ! OUTPUT FILE NAME
0045 * CHARACTER*10 IFNAME ! INPUT FILE NAME
0046 *
0047 *
0048 * INPUT PARAMETERS
0049 *
0050 * PRINT *, 'ENTER INPUT FILE NAME'
0051 * READ *, IFNAME
0052 * PRINT *, 'ENTER OUTPUT FILE NAME'
0053 * READ *, OFNAME
0054 * PRINT *, 'ENTER TMAX'
0055 * READ *, TMAX
0056 * PRINT *, 'ENTER TIME STEP'
0057 * READ *, DT
```

```
0058      PRINT *, 'ENTER A1, A2'
0059      READ *, A1, A2
0060      PRINT *, 'ENTER ALPHA1, ALPHA2'
0061      READ *, ALPHA1, ALPHA2
0062      PRINT *, 'ENTER TBOUND, BBOUND, RBOUND, LBOUND'
0063      READ *, TBOUND, BBOUND, RBOUND, LBOUND
0064
0065
0066      OPEN(12, FILE=OFNAME, STATUS='NEW')
0067
0068
0069      *      INITIALIZE GRID
0070      CALL INITIALIZE_GRID(B1, B2, P, NO_PTS, IFNAME)
0071      DX = 1.0 / NO_PTS
0072      DY = 1.0 / NO_PTS
0073
0074
0075      *      MAIN LOOP
0076      DO LT = 0.0, TMAX-DT, DT
0077          CALL OUTPUT (B1,B2,P,OFNAME,IFNAME,NO_PTS,A1,A2,ALPHA1,ALPHA2,LT,DX,DY)
0078          CALL ITERAT (B1, B2, P, NO_PTS, DX, DY, DT, TBOUND, BBOUND, LBOUND, RBOUND, A1, A2, ALPHA1, ALPHA2)
0079      END DO
0080
0081
0082      *      ITERATIONS FINISHED--OUTPUT FINAL STATE OF FIELD
0083      CALL OUTPUT(B1,B2,P,OFNAME,IFNAME,NO_PTS,A1,A2,ALPHA1,ALPHA2,LT,DX,DY)
0084
0085
0086      *      TERMINATE PROGRAM
0087
0088      CLOSE(12)
0089      END
```

IMPLICIT\$MAIN
01

2-May-1990 01:19:50
2-May-1990 01:19:43

VAX FORTRAN V5.3-50
SYS\$USER6:[JTILSON.MATH490]IMPLICIT.FOR;12

Page 3

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	571	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	136	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	22072	PIC CON REL LCL NOSHR NOEXE RD WRT QUAD
Total Space Allocated	22779	

ENTRY POINTS

Address	Type	Name
0-00000000		IMPLICIT\$MAIN

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name	Address	Type	Name
2-000054A0	R*8	A1	2-000054A8	R*8	A2	2-000054B0	R*8	ALPHA1	2-000054B8	R*8	ALPHA2
2-000054C8	R*8	BBOUND	2-00005498	R*8	DT	2-00005488	R*8	DX	2-00005490	R*8	DY
2-0000546A	CHAR	IFNAME	2-000054D0	R*8	LBOUND	2-00005478	R*8	LT	2-000054F8	I*4	LX
2-000054FC	I*4	LY	2-00005500	I*4	NO_PTS	2-00005460	CHAR	OFNAME	2-000054D8	R*8	RBOUND
2-000054C0	R*8	TBOUND	2-00005480	R*8	TMAX	2-000054E0	R*8	X	2-000054E8	R*8	Y

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	R*8	B1	7200	(30, 30)
2-00001C20	R*8	B2	7200	(30, 30)
2-00003840	R*8	P	7200	(30, 30)

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	FOR\$CLOSE		FOR\$OPEN		INITIALIZE_GRID
	ITERAT		OUTPUT		

```

0001 *****
0002 *
0003 * SUBROUTINE OUTPUT:
0004 *
0005 * THIS SUBROUTINE SENDS DATA FROM THE CURRENT TIME STEP TO THE FILE SPEC-
0006 * IFIED BY OFNAME.
0007 *
0008 *****
0009
0010 SUBROUTINE OUTPUT(B1,B2,P,OFNAME,IFNAME,NO_PTS,A1,A2,ALPHA1,ALPHA2,LT,DX,DY)
0011 *
0012 * PARAMETERS
0013 *
0014 REAL*8 B1(30,30), B2(30,30), P(30,30) ! ARRAYS FOR PLANT, PATHOGEN DATA
0015 CHARACTER*10 OFNAME, IFNAME ! FILES
0016 INTEGER*4 NO_PTS ! NUMBER OF POINTS IN GRID
0017 REAL*8 A1, A2, ALPHA1, ALPHA2 ! GROWTH AND DECAY CONSTANTS
0018 REAL*8 LT ! CURRENT TIME STEP
0019 REAL*8 DX, DY ! GRID SQUARE DIMENSIONS
0020
0021 *
0022 * LOCAL VARIABLES
0023 *
0024 INTEGER*4 LX, LY ! LOOP VARIABLES
0025 REAL*8 SUMB1, SUMB2, SUMP, SUMB1B2 ! SUM VARIABLES
0026
0027 *
0028 * CLEAR SUM VARIABLES
0029 SUMB1 = 0.0
0030 SUMB2 = 0.0
0031 SUMP = 0.0
0032
0033 *
0034 * ADD DENSITIES
0035 DO LX = 1,NO_PTS
0036 DO LY = 1,NO_PTS
0037 SUMB1 = SUMB1 + B1(LX,LY)
0038 SUMB2 = SUMB2 + B2(LX,LY)
0039 SUMP = SUMP + P (LX,LY)
0040 SUMB1B2 = SUMB1 + SUMB2
0041 END DO
0042 END DO
0043
0044 *
0045 * MULTIPLY BY SIZE OF EACH SQUARE GIVING AMOUNT OF MASS
0046 SUMB1 = SUMB1 * (DX*DY)
0047 SUMB2 = SUMB2 * (DX*DY)
0048 SUMP = SUMP * (DX*DY)
0049 SUMB1B2 = SUMB1B2 * (DX*DY)
0050
0051 *
0052 * SEND TO FILE
0053 WRITE (12, 50) IFNAME, NO_PTS, A1, A2, ALPHA1, ALPHA2, LT, SUMB1, SUMB2, SUMP, SUMB1B2
0054 50 FORMAT (' ', A8, X, I2, X, 9(F8.5, X))
0055
0056 *
0057 * EXIT TO MAIN PROGRAM

```

OUTPUT

2-May-1990 01:19:50
2-May-1990 01:19:43

VAX FORTRAN V5.3-50
SYS\$USER6:[JTILSON.MATH490]IMPLICIT.FOR;12
Page 5

0058

END

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	348	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	22	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	124	PIC CON REL LCL NOSHR NOEXE RD WRT QUAD
Total Space Allocated	494	

ENTRY POINTS

Address	Type	Name
0-00000000		OUTPUT

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name	Address	Type	Name
AP-0000001C@	R*8	A1	AP-00000020@	R*8	A2	AP-00000024@	R*8	ALPHA1	AP-00000028@	R*8	ALPHA2
AP-00000030@	R*8	DX	AP-00000034@	R*8	DY	AP-00000014@	CHAR	IFNAME	AP-0000002C@	R*8	LT
2-00000020	I*4	LX	2-00000024	I*4	LY	AP-00000018@	I*4	NO_PTS	AP-00000010@	CHAR	OFNAME
2-00000000	R*8	SUMB1	2-00000018	R*8	SUMB1B2	2-00000008	R*8	SUMB2	2-00000010	R*8	SUMP

ARRAYS

Address	Type	Name	Bytes	Dimensions
AP-00000004@	R*8	B1	7200	(30, 30)
AP-00000008@	R*8	B2	7200	(30, 30)
AP-0000000C@	R*8	P	7200	(30, 30)

LABELS

Address	Label
1-00000000	50'

```

0001 *****
0002 *
0003 * SUBROUTINE INITIALIZE:
0004 *
0005 * THIS SUBROUTINE TAKES ITS INPUT FROM A FILE SPECIFIED BY THE USER. THIS
0006 * INPUT IS THE INITIAL PLANTING PATTERN FOR THE FIELD, CONSISTING OF:
0007 *
0008 * LINE 1: NUMBER OF POINTS IN GRID (30 MAX.)
0009 * SUBSEQUENT LINES: X AND Y RANGES, CONCENTRATIONS OF PLANTS AND PATHOGEN
0010 * IN RECTANGULAR BOX FORMED BY X AND Y COORDINATES
0011 * LAST LINE: CONTAINS SENTINEL 999 999 999 999 999 999 999 TO DENOTE
0012 * END OF FILE
0013 *
0014 *****
0015
0016 SUBROUTINE INITIALIZE_GRID(B1, B2, P, NO_PTS, IFNAME)
0017 *
0018 * PARAMETERS
0019 *
0020 REAL*8 B1(30,30), B2(30,30), P(30,30) ! ARRAYS WITH PLANT AND PATHOGEN DATA
0021 INTEGER*4 NO_PTS ! NUMBER OF POINTS IN GRID
0022 CHARACTER*10 IFNAME ! INPUT FILE NAME
0023
0024 *
0025 * LOCAL VARIABLES
0026 *
0027 INTEGER*2 X, Y, XMN, XMX, YMN, YMX ! X AND Y BOUNDARIES FOR EACH DATA LINE
0028 REAL*8 B1VAL, B2VAL, PVAL ! CONCENTRATIONS OF PLANT, PATHOGEN AT EACH SQUARE
0029
0030 *
0031 OPEN THE FILE
0032 OPEN (13, FILE=IFNAME, STATUS='OLD')
0033
0034 *
0035 PRIMING READ
0036 READ (13,*) NO_PTS
0037 READ (13,*) XMN, XMX, YMN, YMX, B1VAL, B2VAL, PVAL
0038
0039 *
0040 MAIN LOOP
0041 DO WHILE (XMN .NE. 999)
0042 DO X=XMN, XMX
0043 DO Y=YMN, YMX
0044 B1(X,Y) = B1VAL
0045 B2(X,Y) = B2VAL
0046 P(X,Y) = PVAL
0047 END DO
0048 END DO
0049 READ (13,*) XMN, XMX, YMN, YMX, B1VAL, B2VAL, PVAL
0050 END DO
0051
0052 *
0053 CLOSE FILE AND RETURN TO MAIN
0054 CLOSE(13)
0055 END

```

INITIALIZE_GRID
01

2-May-1990 01:19:50
2-May-1990 01:19:43

VAX FORTRAN V5.3-50
SYS\$USER6:[JTILSON.MATH490]IMPLICIT.FOR;12
Page 7

PROGRAM SECTIONS

Name	Bytes	Attributes									
0 \$CODE	398	PIC	CON	REL	LCL	SHR	EXE	RD	NOWRT	LONG	
2 \$LOCAL	140	PIC	CON	REL	LCL	NOSHR	NOEXE	RD	WRT	QUAD	
Total Space Allocated		538									

ENTRY POINTS

Address	Type	Name
0-00000000		INITIALIZE_GRID

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name	Address	Type	Name
2-000000010	R*8	B1VAL	2-000000018	R*8	B2VAL	AP-000000014@	CHAR	IFNAME	AP-000000010@	I*4	NO_PTS
2-000000020	R*8	PVAL	2-000000000	I*2	X	2-000000004	I*2	XMN	2-000000006	I*2	XM̄
2-000000002	I*2	Y	2-000000008	I*2	YMN	2-00000000A	I*2	YMX			

ARRAYS

Address	Type	Name	Bytes	Dimensions
AP-000000004@	R*8	B1	7200	(30, 30)
AP-000000008@	R*8	B2	7200	(30, 30)
AP-00000000C@	R*8	P	7200	(30, 30)

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	FOR\$CLOSE		FOR\$OPEN

```

0001 *****
0002 *
0003 * SUBROUTINE ITERAT:
0004 *
0005 * THIS SUBROUTINE CALCULATES INFORMATION AT THE N+1TH TIME STEP GIVEN THE
0006 * CONFIGURATION OF THE FIELD AT THE NTH TIME STEP.
0007 *
0008 * IT USES SUCCESSIVE OVER-RELAXATION, CRANK-NICOLSON'S SCHEME.
0009 *
0010 *****
0011 SUBROUTINE ITERAT (B1,B2,P,NO_PTS,DX,DY,DT,TBOUND,BBOUND,LBOUND,RBOUND,A1,A2,ALPHA1,ALPHA2)
0012 *
0013 * PARAMETERS
0014 *
0015 REAL*8 B1(30,30),B2(30,30),P(30,30) ! PLANT, PATHOGEN DATA
0016 INTEGER*4 NO_PTS ! NUMBER OF POINTS IN GRID
0017 REAL*8 DX,DY,DT ! DIFFERENTIALS
0018 REAL*8 TBOUND,BBOUND,LBOUND,RBOUND ! BOUNDARY FUNCTIONS
0019 REAL*8 A1,A2,ALPHA1,ALPHA2 ! GROWTH COEFFICIENTS
0020
0021 *
0022 * LOCAL VARIABLES
0023 *
0024 REAL*8 ACCERR ! ACCEPTABLE ERROR
0025 REAL*8 W ! 0 < W < 2 FOR SOR METHOD
0026 REAL*8 B1O(30,30),B2O(30,30),PO(30,30) ! PREVIOUS GUESS FOR N+1TH TIME STEP
0027 REAL*8 B1N(30,30),B2N(30,30),PN(30,30) ! NEXT GUESS FOR N+1TH TIME STEP
0028 REAL*8 R,K1,K2 ! COEFFICIENTS TO TIDY UP FORMULAS
0029 REAL*8 B1ERR, B2ERR, PERR ! ERROR BETWEEN GUESSES
0030 INTEGER*4 X,Y ! LOOP VARIABLES
0031 LOGICAL ACCFLG ! ACCEPTANCE FLAG
0032
0033
0034 * COPY MATRIX FOR OPERATIONS
0035 DO X=1,NO_PTS
0036 DO Y=1,NO_PTS
0037 B1O(X,Y) = B1(X,Y)
0038 B2O(X,Y) = B2(X,Y)
0039 PO(X,Y) = P(X,Y)
0040 END DO
0041 END DO
0042
0043
0044 * SET VARIABLES FOR ITERATIONS
0045 ACCFLG = .FALSE.
0046 R = DT/(DX**2)
0047 ACCERR = 0.001
0048 W = 1.35
0049
0050
0051 * MAIN LOOP
0052 DO WHILE (.NOT. (ACCFLG))
0053 * RESET ERROR COUNTERS
0054 B1ERR = 0.0
0055 B2ERR = 0.0
0056 PERR = 0.0
0057

```

```

0058
0059 *      CALCULATE PLANT MASS AT N+1TH TIME STEP
0060 DO X=1,NO_PTS
0061   DO Y=1,NO_PTS
0062     K1 = 2 + DT * (A1 - ALPHA1 * P(X,Y))
0063     K2 = 2 + DT * (-A1 + ALPHA1 * PO(X,Y))
0064     B1N(X,Y) = B1O(X,Y) + (W * (K1 * B1(X,Y) - K2 * B1O(X,Y))) / K2
0065     B1ERR = B1ERR + (B1N(X,Y) - B1O(X,Y)) ** 2
0066
0067     K1 = 2 + DT * (A2 - ALPHA2 * P(X,Y))
0068     K2 = 2 + DT * (-A2 + ALPHA2 * PO(X,Y))
0069     B2N(X,Y) = B2O(X,Y) + (W * (K1 * B2(X,Y) - K2 * B2O(X,Y))) / K2
0070     B2ERR = B2ERR + (B2N(X,Y) - B2O(X,Y)) ** 2
0071   END DO
0072 END DO
0073
0074
0075 *      PATHOGEN CALCULATIONS
0076 *      BOTTOM LEFT CORNER
0077 K1 = 2 - (2 * R) + (DT * (ALPHA1*B1(1,1) + ALPHA2*B2(1,1)))
0078 K2 = 2 + (2 * R) - (DT * (ALPHA1*B1N(1,1) + ALPHA2*B2N(1,1)))
0079 PN(1,1) = PO(1,1) + (W * (R * (DX*(LBOUND + BBOUND) + P(2,1) + P(1,2) + PO(2,1) + PO(1,2))
0080 1          + (K1 * P(1,1)) - (K2 * PO(1,1))) / K2)
0081 PERR = PERR + (PN(1,1) - PO(1,1)) ** 2
0082
0083
0084 *      BOTTOM MIDDLE ROW
0085 DO X=2,NO_PTS-1
0086   K1 = 2 - (3 * R) + (DT * (ALPHA1*B1(X,1) + ALPHA2*B2(X,1)))
0087   K2 = 2 + (3 * R) - (DT * (ALPHA1*B1N(X,1) + ALPHA2*B2N(X,1)))
0088   PN(X,1) = PO(X,1) + (W * (R * (DX*BBOUND + P(X-1,1) + P(X,2) + P(X+1,1) + PN(X-1,1) + PO(X,2) + PO(X+1,1))
0089 1          + (K1 * P(X,1)) - (K2 * PO(X,1))) / K2)
0090   PERR = PERR + (PN(X,1) - PO(X,1)) ** 2
0091 END DO
0092
0093
0094 *      BOTTOM RIGHT CORNER
0095 K1 = 2 - (2 * R) + (DT * (ALPHA1*B1(NO_PTS,1) + ALPHA2*B2(NO_PTS,1)))
0096 K2 = 2 + (2 * R) - (DT * (ALPHA1*B1N(NO_PTS,1) + ALPHA2*B2N(NO_PTS,1)))
0097 PN(NO_PTS,1) = PO(NO_PTS,1) + (W * (R * (DX*(RBOUND + BBOUND) + P(NO_PTS-1,1) + P(NO_PTS,2)
0098 1          + PN(NO_PTS-1,1) + PO(NO_PTS,2))
0099 2          + (K1 * P(NO_PTS,1)) - (K2 * PO(NO_PTS,1))) / K2)
0100 PERR = PERR + (PN(NO_PTS,1) - PO(NO_PTS,1)) ** 2
0101
0102
0103 *      MIDDLE ROWS
0104 DO Y = 2, NO_PTS - 1
0105
0106 *      LEFT EDGE
0107 K1 = 2 - (3 * R) + (DT * (ALPHA1*B1(1,Y) + ALPHA2*B2(1,Y)))
0108 K2 = 2 + (3 * R) - (DT * (ALPHA1*B1N(1,Y) + ALPHA2*B2N(1,Y)))
0109 PN(1,Y) = PO(1,Y) + (W * (R * (DX*LBOUND + P(2,Y) + P(1,Y-1) + P(1,Y+1) + PO(2,Y) + PN(1,Y-1) + PO(1,Y+1))
0110 1          + (K1 * P(1,Y)) - (K2 * PO(1,Y))) / K2)
0111 PERR = PERR + (PN(1,Y) - PO(1,Y)) ** 2
0112
0113
0114 *      INSIDE MIDDLE

```

```

0115      DO X = 2, NO_PTS - 1
0116          K1 = 2 - (4 * R) + (DT * (ALPHA1*B1(X,Y) + ALPHA2*B2(X,Y)))
0117          K2 = 2 + (4 * R) - (DT * (ALPHA1*B1N(X,Y) + ALPHA2*B2N(X,Y)))
0118          PN(X,Y) = PO(X,Y) + (W * (R * (P(X-1,Y) + P(X+1,Y) + P(X,Y-1) + P(X,Y+1)
0119              + PN(X-1,Y) + PN(X,Y-1) + PO(X+1,Y) + PO(X,Y+1))
0120              + (K1 * P(X,Y)) - (K2 * PO(X,Y))) / K2)
0121      PERR = PERR + (PN(X,Y) - PO(X,Y)) ** 2
0122  END DO
0123
0124  *
0125      RIGHT EDGE
0126      K1 = 2 - (3 * R) + (DT * (ALPHA1*B1(NO_PTS,Y) + ALPHA2*B2(NO_PTS,Y)))
0127      K2 = 2 + (3 * R) - (DT * (ALPHA1*B1N(NO_PTS,Y) + ALPHA2*B2N(NO_PTS,Y)))
0128      PN(NO_PTS,Y) = PO(NO_PTS,Y) + (W * (R * (DX*BOUND + P(NO_PTS-1,Y) + P(NO_PTS,Y+1)
0129          + PN(NO_PTS-1,Y) + PN(NO_PTS,Y-1) + PO(NO_PTS,Y+1))
0130          + (K1 * P(NO_PTS,Y)) - (K2 * PO(NO_PTS,Y))) / K2)
0131      PERR = PERR + (PN(NO_PTS,Y) - PO(NO_PTS,Y)) ** 2
0132  END DO
0133  *
0134  END MIDDLE ROWS
0135
0136  *
0137      TOP LEFT CORNER
0138      K1 = 2 - (2 * R) + (DT * (ALPHA1*B1(1,NO_PTS) + ALPHA2*B2(1,NO_PTS)))
0139      K2 = 2 + (2 * R) - (DT * (ALPHA1*B1N(1,NO_PTS) + ALPHA2*B2N(1,NO_PTS)))
0140      PN(1,NO_PTS) = PO(1,NO_PTS) + (W * (R * (DX*(TBOUND + LBOUND) + P(2,NO_PTS) + P(1,NO_PTS-1)
0141          + PO(2,NO_PTS) + PN(1,NO_PTS-1))
0142          + (K1 * P(1,NO_PTS)) - (K2 * PO(1,NO_PTS))) / K2)
0143      PERR = PERR + (PN(1,NO_PTS) - PO(1,NO_PTS)) ** 2
0144
0145  *
0146      TOP MIDDLE ROW
0147      DO X = 2, NO_PTS-1
0148          K1 = 2 - (3 * R) + (DT * (ALPHA1*B1(X,NO_PTS) + ALPHA2*B2(X,NO_PTS)))
0149          K2 = 2 + (3 * R) - (DT * (ALPHA1*B1N(X,NO_PTS) + ALPHA2*B2N(X,NO_PTS)))
0150          PN(X,NO_PTS) = PO(X,NO_PTS) + (W * (R * (DX*(TBOUND + RBOUND) + P(X-1,NO_PTS) + P(X+1,NO_PTS) + P(X,NO_PTS-1)
0151              + PN(X-1,NO_PTS) + PN(X,NO_PTS-1) + PO(X+1,NO_PTS))
0152              + (K1 * P(X,NO_PTS)) - (K2 * PO(X,NO_PTS))) / K2)
0153      PERR = PERR + (PN(X,NO_PTS) - PO(X,NO_PTS)) ** 2
0154  END DO
0155
0156  *
0157      TOP RIGHT CORNER
0158      K1 = 2 - (2 * R) + (DT * (ALPHA1*B1(NO_PTS,NO_PTS) + ALPHA2*B2(NO_PTS,NO_PTS)))
0159      K2 = 2 + (2 * R) - (DT * (ALPHA1*B1N(NO_PTS,NO_PTS) + ALPHA2*B2N(NO_PTS,NO_PTS)))
0160      PN(NO_PTS,NO_PTS) = PO(NO_PTS,NO_PTS) + (W * (R * (DX*(TBOUND + RBOUND) + P(NO_PTS-1,NO_PTS) + P(NO_PTS,NO_PTS-1)
0161          + PN(NO_PTS-1,NO_PTS) + PN(NO_PTS,NO_PTS-1))
0162          + (K1 * P(NO_PTS,NO_PTS)) - (K2 * PO(NO_PTS,NO_PTS))) / K2)
0163      PERR = PERR + (PN(NO_PTS,NO_PTS) - PO(NO_PTS,NO_PTS)) ** 2
0164
0165  *
0166      FIND 2-NORM OF ERROR TERMS--SQUARE ROOT OF SUM OF DIFFERENCE OF SQUARES
0167      B1ERR = B1ERR ** 0.5
0168      B2ERR = B2ERR ** 0.5
0169      PERR = PERR ** 0.5
0170
0171  *
0172      WITHIN ACCEPTABLE ERROR?

```

ITERAT

2-May-1990 01:19:50

VAX FORTRAN V5.3-50

Page 11

2-May-1990 01:19:43

SYS\$USER6:[JTILSON.MATH490]IMPLICIT.FOR,12

```
0172      IF ((B1ERR .LE. ACCERR) .AND. (B2ERR .LE. ACCERR) .AND. (PERR .LE. ACCERR)) THEN
0173      *      YES--SET FLAG FOR EXIT FROM SUBROUTINE
0174      ACCFLG = .TRUE.
0175      ELSE
0176      *      NO--UPDATE NEW GUESS AND TRY AGAIN
0177      DO X = 1,NO_PTS
0178      DO Y = 1,NO_PTS
0179      B1O(X,Y) = B1N(X,Y)
0180      B2O(X,Y) = B2N(X,Y)
0181      PO(X,Y) = PN(X,Y)
0182      END DO
0183      END DO
0184      END IF
0185
0186      END DO
0187
0188
0189      *      ITERATIONS FINISHED FOR THIS TIME STEP--RETURN NEW STATUS
0190      DO X = 1,NO_PTS
0191      DO Y = 1,NO_PTS
0192      B1(X,Y) = B1N(X,Y)
0193      B2(X,Y) = B2N(X,Y)
0194      P(X,Y) = PN(X,Y)
0195      END DO
0196      END DO
0197
0198
0199      *      RETURN TO MAIN PROGRAM
0200      END
```

ITERAT
01

2-May-1990 01:19:50
2-May-1990 01:19:43

VAX FORTRAN V5.3-50
SYS\$USER6:[JTILSON.MATH490]IMPLICIT.FOR:12

Page 12

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	3582	PIC CON REL LCL SHR EXE RD NOWRT LONG
2 \$LOCAL	43384	PIC CON REL LCL NOSHR NOEXE RD WRT QUAD
Total Space Allocated	46966	

ENTRY POINTS

Address	Type	Name
0-00000000		ITERAT

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name	Address	Type	Name
AP-00000030a	R*8	A1	AP-00000034a	R*8	A2	2-0000A8C0	R*8	ACCERR	2-0000A908	L*4	ACCFLG
AP-00000038a	R*8	ALPHA1	AP-0000003Ca	R*8	ALPHA2	2-0000A8E8	R*8	B1ERR	2-0000A8F0	R*8	B2ERR
AP-00000024a	R*8	BBOUND	AP-0000001Ca	R*8	DT	AP-00000014a	R*8	DX	AP-00000018a	R*8	DY
2-0000A8D8	R*8	K1	2-0000A8E0	R*8	K2	AP-00000028a	R*8	LBOUND	AP-00000010a	I*4	NO_PTS
2-0000A8F8	R*8	PERR	2-0000A8D0	R*8	R	AP-0000002Ca	R*8	RBOUND	AP-00000020a	R*8	TBOUND
2-0000A8C8	R*8	W	2-0000A900	I*4	X	2-0000A904	I*4	Y			

ARRAYS

Address	Type	Name	Bytes	Dimensions
AP-00000004a	R*8	B1	7200	(30, 30)
2-00005460	R*8	B1N	7200	(30, 30)
2-00000000	R*8	B10	7200	(30, 30)
AP-00000008a	R*8	B2	7200	(30, 30)
2-00007080	R*8	B2N	7200	(30, 30)
2-00001C20	R*8	B2O	7200	(30, 30)
AP-0000000Ca	R*8	P	7200	(30, 30)
2-00008CA0	R*8	PN	7200	(30, 30)
2-00003840	R*8	PO	7200	(30, 30)

ITERAT
01

2-May-1990 01:19:50
2-May-1990 01:19:43

VAX FORTRAN V5.3-50
SYS\$USER6:[JTILSON.MATH490]IMPLICIT.FOR:12

Page 13

COMMAND QUALIFIERS

FORTRAN/NOOP/LIST/EXTEND_SOURCE IMPLICIT

/CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/SHOW=(NODICTIONARY,NOINCLUDE,MAP,NOPREPROCESSOR,SINGLE)
/STANDARD=(NOSEMANTIC,NOSOURCE_FORM,NOSYNTAX)
/WARNINGS=(NODECLARATIONS,GENERAL,NOULTRIX,NOVAXELN)
/CONTINUATIONS=19 /NOCROSS_REFERENCE /NOD_LINES /EXTEND_SOURCE
/F77 /NOG_FLOATING /I4 /NOMACHINE_CODE /NOOPTIMIZE /NOPARALLEL
/NOANALYSIS_DATA
/NODIAGNOSTICS
/LIST=SYS\$USER6:[JTILSON.MATH490]IMPLICIT.LIS;1
/OBJECT=SYS\$USER6:[JTILSON.MATH490]IMPLICIT.OBJ;7

COMPILATION STATISTICS

Run Time:	1.89 seconds
Elapsed Time:	15.66 seconds
Page Faults:	897
Dynamic Memory:	558 pages