Spring 5-1990

Optimal Planting Patterns in the Presence of Pathogen

James Tilson
University of Tennessee - Knoxville

Follow this and additional works at: https://trace.tennessee.edu/utk_chanhonoproj

Recommended Citation
Tilson, James, "Optimal Planting Patterns in the Presence of Pathogen" (1990). University of Tennessee Honors Thesis Projects.
https://trace.tennessee.edu/utk_chanhonoproj/63

This is brought to you for free and open access by the University of Tennessee Honors Program at Trace: Tennessee Research and Creative Exchange. It has been accepted for inclusion in University of Tennessee Honors Thesis Projects by an authorized administrator of Trace: Tennessee Research and Creative Exchange. For more information, please contact trace@utk.edu.
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) x (0,1), divided into a grid of up to size 30x30.

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, \(\alpha_1, \alpha_2\)=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:

\[
P(x,y,t) = \text{amount of pathogen at location } (x,y) \text{ at time } t
\]
\[
b_1(x,y,t) = \text{amount of plant at location } (x,y) \text{ at time } t
\]
\[
b_2(x,y,t) = \text{amount of plant around the borders (i.e. } x=0, x=1, y=0 \text{ or } y=1 \text{) is } 0 \text{ at all times. Thus the starting conditions are:}
\]

\[
P(x,y,0) = 0
\]
\[
b_1(x,y,0) = \text{given initial pattern}
\]
\[
b_2(x,y,0) = 0
\]
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 $\frac{\partial}{\partial t}$ 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:

\[
(b_{1})_{i,j}^{n+1} = (b_{1})_{i,j}^{n} + \Delta t \left[ a_{1}(b_{1})_{i,j} - \alpha_{1}(b_{1})_{i,j} P_{i,j}^{n} \right]
\]
\[
(b_{2})_{i,j}^{n+1} = (b_{2})_{i,j}^{n} + \Delta t \left[ a_{2}(b_{2})_{i,j} - \alpha_{2}(b_{2})_{i,j} P_{i,j}^{n} \right]
\]
\[
P_{i,j}^{n+1} = P_{i,j}^{n} + \Delta t \left[ \left( \frac{P_{i,j+1}^{n} - P_{i,j}^{n}}{\Delta y} \right) + \left( \frac{P_{i,j-1}^{n} - P_{i,j}^{n}}{\Delta y} \right) + \left( \frac{P_{i+1,j}^{n} - P_{i,j}^{n}}{\Delta x} \right) + \left( \frac{P_{i-1,j}^{n} - P_{i,j}^{n}}{\Delta x} \right) + \alpha_{1}(b_{1})_{i,j} P_{i,j}^{n} + \alpha_{2}(b_{2})_{i,j} P_{i,j}^{n} \right]
\]

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
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,

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

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

and for the pathogen,

\[ P_{i,j}^{\text{new}} = P_{i,j}^{\text{old}} + \omega \left[ r\left( + P_{i-1,j}^{\text{old}} + P_{i+1,j}^{\text{old}} + P_{i,j-1}^{\text{old}} + P_{i,j+1}^{\text{old}} + P_{i,j}^{\text{new}} + P_{i-1,j}^{\text{new}} + P_{i,j}^{\text{new}} + P_{i,j+1}^{\text{new}} \right) + (2 - 4r + \Delta t \left[ \alpha_1(B_{i,j})^{\text{old}} + \alpha_2(B_{i,j})^{\text{old}} \right]) P_{i,j}^{\text{old}} \right] \]

\[ - \left( 2 + 4r - \Delta t \left[ \alpha_1(B_{i,j})^{\text{new}} + \alpha_2(B_{i,j})^{\text{new}} \right] P_{i,j}^{\text{new}} \right) \]

\[ 2 + 4r - \Delta t \left( \alpha_1(B_{i,j})^{\text{new}} + \alpha_2(B_{i,j})^{\text{new}} \right) \]

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.

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

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 \((\alpha_1 / \alpha_2)\) 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)

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Max total plant</th>
<th>(a_1)</th>
<th>(a_2)</th>
<th>(\alpha_1)</th>
<th>(\alpha_2)</th>
<th>((\alpha_1 / \alpha_2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evenbar5</td>
<td>6.4195</td>
<td>2.0</td>
<td>0.5</td>
<td>4.0</td>
<td>0.5</td>
<td>8.0</td>
</tr>
<tr>
<td>Ckr5</td>
<td>6.4605</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evenfrm</td>
<td>6.4728</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allb1</td>
<td>6.5536</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evenfrm</td>
<td>6.7196</td>
<td>4.0</td>
<td>1.0</td>
<td>8.0</td>
<td>0.5</td>
<td>16.0</td>
</tr>
<tr>
<td>Evenbar5</td>
<td>6.7444</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ckr5</td>
<td>6.8246</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allb1</td>
<td>6.8269</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evenfrm</td>
<td>6.7226</td>
<td>4.0</td>
<td>1.0</td>
<td>8.0</td>
<td>0.0625</td>
<td>128.0</td>
</tr>
<tr>
<td>Evenbar5</td>
<td>6.8157</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allb1</td>
<td>6.8268</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ckr5</td>
<td>6.8898</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 (alpha1, alpha2) 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


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 $a_1$, $a_2$, $\alpha_1$, $\alpha_2$ 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

b1 + b2

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

b1 + b2

pathogen

TIME
\[ b_1 + b_2 \]
A1=4  A2=1  ALPHAI=8  ALPHAI2=0.0625  NAME=CKR5

\[ b_1 + b_2 \]

name

pathogen

B1B2

0.00 0.02 0.04 0.06 0.08 0.10 0.12 0.14 0.16 0.18 0.20

TIME
0000 PROGRAM: FINITEDIFF.FOR
0001 AUTHOR: JIM TILSON
0002 DATE: MAY 2, 1990
0003 CLASS: TENNESSEE SCHOLARS, MATH 490
0004
0005 THIS PROGRAM SIMULATES THE GROWTH OF TWO KINDS OF PLANT IN A FIELD WHERE
0006 A PATHOGEN IS PRESENT. GROWTH RATES OF THE PLANTS AND OF THE PATHOGEN
0007 ARE CONTROLLED BY COEFFICIENTS ENTERED BY THE USER. DATA FOR THE INITIAL
0008 FIELD PLANTING PATTERN ARE READ FROM A FILE SPECIFIED BY THE USER AND
0009 RESULTS CONSISTING OF DATA FROM EACH TIME STEP ARE SENT TO AN OUTPUT FILE
0010 ALSO SPECIFIED BY THE USER, IN A FORMAT READY FOR DATA ANALYSIS USING SAS.
0011
0012 THE PROGRAM USES A FINITE DIFFERENCING SCHEME TO CALCULATE DATA FOR THE
0013 N+1TH TIME STEP USING DATA FROM THE NTH TIME STEP.
0014
0015 INPUT: INPUT CONSISTS OF PARAMETERS, INPUT BY THE USER, AND A DATA FILE
0016 (ALSO SPECIFIED BY THE USER) CONTAINING THE INFORMATION FOR THE
0017 PLANTING PATTERN.
0018
0019 OUTPUT: OUTPUT IS SENT TO A FILE AT EACH TIME STEP. IT IS FORMATTED FOR
0020 USE BY SAS FOR ANY KIND OF STATISTICAL ANALYSIS WHICH THE USER
0021 WOULD LIKE TO USE.
0022
0023 FINITEDIFF--MAIN PROGRAM
0024 IMPLICIT NONE
0025
0026 VARIABLE DECLARATIONS
0027 REAL*8 LT LOOP VARIABLE
0028 REAL*8 TMAX MAXIMUM TIME
0029 REAL*8 DX, DY, DT CHANGE IN X, Y, TIME
0030 REAL*8 A1, A2, ALPHA1, ALPHA2 EQUATION COEFFS.
0031 REAL*8 B1(2, 30, 30) ARRAY FOR PLANT 1
0032 REAL*8 B2(2, 30, 30) ARRAY FOR PLANT 2
0033 REAL*8 P (2, 30, 30) ARRAY FOR PATHOGEN
0034 REAL*8 A, B, C, D PARTIAL EQN. PARAMS.
0035 REAL*8 TBOUND, BBOUND BOUNDARY FUNCTIONS
0036 REAL*8 LBOUND, RBOUND BOUNDARY FUNCTIONS
0037 REAL*8 X, Y COORDS. OF GRID SQUARE CNTR.
0038 INTEGER*4 LX, LY LOOP VARIABLES FOR GRID
0039 INTEGER*4 NO_PTS NUMBER OF POINTS IN GRID
0040 INTEGER*4 OLD USED TO DENOTE OLD TIME STEP INFO IN GRIDS
0041 CHARACTER*10 OFNAME OUTPUT FILE NAME
0042 CHARACTER*10 IFNAME INPUT FILE NAME
0043
0044 INPUT PARAMETERS
0045 PRINT *, 'ENTER INPUT FILE NAME'
0046 READ *+, IFNAME
0047
FINITEDIFF$MAIN

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              *
0090          OUTPUT CURRENT FIELD INFORMATION
0091              CALL OUTPUT(B1, B2, P, OLD, OFNAME, IFNAME, NO_PTS, A1, A2, ALPHA1, ALPHA2, LT, DX, DY)
0092
0093  *
0094  CALCULATE PARAMETERS
0095  IF (LX .EQ. NO_PTS) THEN
0096      A = RBOUND
0097  ELSE
0098      A = (P(OLD, LX+1, LY) - P(OLD, LX, LY)) / DX
0099  END IF
0100
0101  IF (LX .EQ. 1) THEN
0102      B = LBOUND
0103  ELSE
0104      B = (P(OLD, LX-1, LY) - P(OLD, LX, LY)) / DX
0105  END IF
0106
0107  IF (LY .EQ. NO_PTS) THEN
0108      C = TBOUND
0109  ELSE
0110      C = (P(OLD, LX, LY+1) - P(OLD, LX, LY)) / DY
0111  END IF
0112
0113  IF (LY .EQ. 1) THEN
0114      D = BBOUND
0115  ELSE
0115 \[ D = \frac{P(OLD, LX, LY-1) - P(OLD, LX, LY)}{DY} \]
0116 \[ END IF \]
0117
0118
0119 \[ CALCULATE NEW GRID VALUES \]
0120 \[ P(3-OLD, LX, LY) = P(OLD, LX, LY) + DT \times \frac{(A+B)}{DX} + \frac{(C+D)}{DY} + \frac{ALPHA1 \times B1(OLD, LX, LY) \times P(OLD, LX, LY)}{P(OLD, LX, LY)} \]
0121 \[ B1(3-OLD, LX, LY) = B1(OLD, LX, LY) + DT \times \frac{(A1 \times B1(OLD, LX, LY) - ALPHA1 \times B1(OLD, LX, LY) \times P(OLD, LX, LY))}{P(OLD, LX, LY)} \]
0122 \[ B2(3-OLD, LX, LY) = B2(OLD, LX, LY) + DT \times \frac{(A2 \times B2(OLD, LX, LY) - ALPHA2 \times B2(OLD, LX, LY) \times P(OLD, LX, LY))}{P(OLD, LX, LY)} \]
0123 \[ END DO \]
0124 \[ OLD = 3 - OLD \]
0125 \[ END DO \]
0126
0127
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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bytes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CODE</td>
<td>1386</td>
<td>PIC CON REL LCL SHR EXE RD NOWRT LONG</td>
</tr>
<tr>
<td>$PDATA</td>
<td>121</td>
<td>PIC CON REL LCL SHR NOEXE RD NOWRT LONG</td>
</tr>
<tr>
<td>$LOCAL</td>
<td>43656</td>
<td>PIC CON REL LCL NOSHR NOEXE RD WRT QUAD</td>
</tr>
</tbody>
</table>

Total Space Allocated: 45163

**ENTRY POINTS**

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-00000000</td>
<td>FINITEDIFF$MAIN</td>
<td></td>
</tr>
</tbody>
</table>

**VARIABLES**

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-0000A920</td>
<td>R*8</td>
<td>A</td>
</tr>
<tr>
<td>2-0000A918</td>
<td>R*8</td>
<td>ALPHA2</td>
</tr>
<tr>
<td>2-0000A938</td>
<td>R*8</td>
<td>D</td>
</tr>
<tr>
<td>2-0000A8CA</td>
<td>R*8</td>
<td>IFNAME</td>
</tr>
<tr>
<td>2-0000A97C</td>
<td>I*4</td>
<td>LY</td>
</tr>
<tr>
<td>2-0000A95B</td>
<td>R*8</td>
<td>RBOUND</td>
</tr>
<tr>
<td>2-0000A96B</td>
<td>R*8</td>
<td>Y</td>
</tr>
<tr>
<td>2-0000A90B</td>
<td>R*8</td>
<td>A1</td>
</tr>
<tr>
<td>2-0000A948</td>
<td>R*8</td>
<td>B</td>
</tr>
<tr>
<td>2-0000A5F8</td>
<td>R*8</td>
<td>DT</td>
</tr>
<tr>
<td>2-0000A950</td>
<td>R*8</td>
<td>LBOUND</td>
</tr>
<tr>
<td>2-0000A980</td>
<td>I*4</td>
<td>NO_PTS</td>
</tr>
<tr>
<td>2-0000A940</td>
<td>R*8</td>
<td>TBOUND</td>
</tr>
<tr>
<td>2-0000A910</td>
<td>R*8</td>
<td>A2</td>
</tr>
<tr>
<td>2-0000A930</td>
<td>R*8</td>
<td>BBOUND</td>
</tr>
<tr>
<td>2-0000A8F0</td>
<td>R*8</td>
<td>DX</td>
</tr>
<tr>
<td>2-0000A8D0</td>
<td>R*8</td>
<td>LBOUND</td>
</tr>
<tr>
<td>2-0000A978</td>
<td>I*4</td>
<td>LT</td>
</tr>
<tr>
<td>2-0000A984</td>
<td>I*4</td>
<td>OLD</td>
</tr>
<tr>
<td>2-0000A960</td>
<td>R*8</td>
<td>X</td>
</tr>
<tr>
<td>2-0000A970</td>
<td>R*8</td>
<td>TMAX</td>
</tr>
<tr>
<td>2-0000A96D</td>
<td>R*8</td>
<td>OLD</td>
</tr>
</tbody>
</table>
SUBROUTINE INITIALIZE:

THIS SUBROUTINE TAKES ITS INPUT FROM A FILE SPECIFIED BY THE USER. THIS INPUT IS THE INITIAL PLANTING PATTERN FOR THE FIELD, CONSISTING OF:

LINE 1: NUMBER OF POINTS IN GRID (30 MAX.)
SUBSEQUENT LINES: X AND Y RANGES, CONCENTRATIONS OF PLANTS AND PATHOGEN IN RECTANGULAR BOX FORMED BY X AND Y COORDINATES
LAST LINE: CONTAINS SENTINEL 999 999 999 999 999 999 TO DENOTE END OF FILE

SUBROUTINE INITIALIZE_GRID(B1, B2, P, NO_PTS, OLD, IFNAME)

PARAMETERS

REAL*8 B1(2,30,30), B2(2,30,30), P(2,30,30) ! PLANT AND PATHOGEN DATA
INTEGER*4 OLD ! SIZE OF GRID
INTEGER*4 NO_PTS ! NUMBER OF POINTS IN GRID
CHARACTER*10 IFNAME ! INPUT FILE NAME

LOCAL VARIABLES

INTEGER*2 X, Y ! LOOP VARIABLES
INTEGER*2 XMN, XMX, YMN, YMX ! X AND Y BOUNDS FOR PLANT AND PATHOGEN DATA
REAL*8 B1VAL, B2VAL, PVAL ! PLANT AND PATHOGEN DATA

OPEN FILE
OPEN (13, FILE=IFNAME, STATUS='OLD')

PRIMING READ
READ (13,*) NO_PTS
READ (13,*) XMN, XMX, YMN, YMX, B1VAL, B2VAL, PVAL

MAIN LOOP
DO WHILE (XMN .NE. 999)
  DO X=XMN, XMX
    DO Y=YMN, YMX
      B1(OLD,X,Y) = B1VAL
      B2(OLD,X,Y) = B2VAL
      P (OLD,X,Y) = PVAL
    END DO
  END DO
  READ (13,*) XMN, XMX, YMN, YMX, B1VAL, B2VAL, PVAL
END DO

CLOSE FILE, RETURN TO MAIN PROGRAM
CLOSE(13)
INITIALIZE_GRID

PROGRAM SECTIONS

<table>
<thead>
<tr>
<th>Name</th>
<th>Bytes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CODE</td>
<td>419</td>
<td>PIC CON REL LCL SHR EXE RD NOWRT LONG</td>
</tr>
<tr>
<td>$LOCAL</td>
<td>140</td>
<td>PIC CON REL LCL NO SHR NO EXE RD WRT QUAD</td>
</tr>
</tbody>
</table>

Total Space Allocated: 559

ENTRY POINTS

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000000</td>
<td></td>
<td>INITIALIZE_GRID</td>
</tr>
</tbody>
</table>

VARIABLES

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Name</th>
<th>Address</th>
<th>Type</th>
<th>Name</th>
<th>Address</th>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000010</td>
<td>R*8</td>
<td>B1VAL</td>
<td>0000000018</td>
<td>R*8</td>
<td>B2VAL</td>
<td>00000018@</td>
<td>CHAR</td>
<td>IFNAME</td>
</tr>
<tr>
<td>000000014@</td>
<td>I*4</td>
<td>OLD</td>
<td>000000020</td>
<td>R*8</td>
<td>PVAl</td>
<td>000000000</td>
<td>I*2</td>
<td>X</td>
</tr>
<tr>
<td>000000006</td>
<td>I*2</td>
<td>XMX</td>
<td>000000002</td>
<td>I*2</td>
<td>Y</td>
<td>000000008</td>
<td>I*2</td>
<td>YMN</td>
</tr>
</tbody>
</table>

ARRAYS

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Name</th>
<th>Bytes</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000004@</td>
<td>R*8</td>
<td>B1</td>
<td>14400</td>
<td>(2, 30, 30)</td>
</tr>
<tr>
<td>000000008@</td>
<td>R*8</td>
<td>B2</td>
<td>14400</td>
<td>(2, 30, 30)</td>
</tr>
<tr>
<td>00000000C@</td>
<td>R*8</td>
<td>P</td>
<td>14400</td>
<td>(2, 30, 30)</td>
</tr>
</tbody>
</table>

FUNCTIONS AND SUBROUTINES REFERENCED

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FOR$CLOSE</td>
<td></td>
<td>FOR$OPEN</td>
</tr>
</tbody>
</table>
**SUBROUTINE OUTPUT:**

**PARAMETERS**

- REAL*8 B1(2,30,30), B2(2,30,30), P(2,30,30) : PLANT AND PATHOGEN DATA
- INTEGER*4 OLD : POINTER TO OLD GUESS
- CHARACTER*10 OFNAME, IFNAME : FILE NAMES
- INTEGER*4 NO_PTS : NUMBER OF POINTS IN GRID
- REAL*8 A1, A2, ALPHA1, ALPHA2 : GROWTH CONSTANTS
- REAL*8 LT : CURRENT TIME STEP
- REAL*8 DX, DY : GRID SQUARE DIMENSIONS

**LOCAL VARIABLES**

- INTEGER*4 LX, LY : LOOP VARIABLES
- REAL*8 SUMB1, SUMB2, SUMP, SUMB1B2 : SUM VARIABLES

**CLEAR SUM VARIABLES**

- SUMB1 = 0.0
- SUMB2 = 0.0
- SUMP = 0.0

**SUM UP CONCENTRATIONS OF PLANT, PATHOGEN DATA**

- DO LX = 1, NO_PTS
  - DO LY = 1, NO_PTS
    - SUMB1 = SUMB1 + B1(OLD, LX, LY)
    - SUMB2 = SUMB2 + B2(OLD, LX, LY)
    - SUMP = SUMP + P(OLD, LX, LY)
    - SUMB1B2 = SUMB1 + SUMB2
  - END DO
- END DO

**MULTIPLY DENSITY BY SQUARE SIZE GIVING TOTAL MASS**

- SUMB1 = SUMB1 * (DX*DY)
- SUMB2 = SUMB2 * (DX*DY)
- SUMP = SUMP * (DX*DY)
- SUMB1B2 = SUMB1B2 * (DX*DY)

**OUTPUT TO FILE**

- WRITE (12, 50) IFNAME, NO_PTS, A1, A2, ALPHA1, ALPHA2, LT, SUMB1, SUMB2, SUMP, SUMB1B2

**RETURN TO MAIN PROGRAM**
### Program Sections

<table>
<thead>
<tr>
<th>Name</th>
<th>Bytes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CODE</td>
<td>369</td>
<td>PIC CON REL LCL SHR EXE RD NOWRT LONG</td>
</tr>
<tr>
<td>$PDATA</td>
<td>22</td>
<td>PIC CON REL LCL SHR NOEXE RD NOWRT LONG</td>
</tr>
<tr>
<td>$LOCAL</td>
<td>124</td>
<td>PIC CON REL LCL NOPIC NOEXE RD WRT QUAD</td>
</tr>
</tbody>
</table>

**Total Space Allocated**: 515

### Entry Points

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-00000000</td>
<td>OUTPUT</td>
<td></td>
</tr>
</tbody>
</table>

### Variables

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP-000000020@R*8</td>
<td>A1</td>
<td>ALPHA1</td>
</tr>
<tr>
<td>AP-000000034@R*8</td>
<td>DX</td>
<td></td>
</tr>
<tr>
<td>2-000000020@1*4</td>
<td>LX</td>
<td>LY</td>
</tr>
<tr>
<td>AP-000000010@1*4</td>
<td>OLD</td>
<td>SUMB1</td>
</tr>
<tr>
<td>2-000000000@R*8</td>
<td>SUMP</td>
<td></td>
</tr>
<tr>
<td>AP-00000000024@R*8</td>
<td>A2</td>
<td>ALPHA2</td>
</tr>
<tr>
<td>AP-000000030@R*8</td>
<td>DY</td>
<td></td>
</tr>
<tr>
<td>AP-00000000024@I*4</td>
<td>IY</td>
<td>NO_PTS</td>
</tr>
<tr>
<td>2-000000000@R*8</td>
<td>SUMB1B2</td>
<td></td>
</tr>
</tbody>
</table>

### Arrays

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Name</th>
<th>Bytes</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP-0000000004@R*8</td>
<td>B1</td>
<td></td>
<td>14400</td>
<td>(2, 30, 30)</td>
</tr>
<tr>
<td>AP-0000000008@R*8</td>
<td>B2</td>
<td></td>
<td>14400</td>
<td>(2, 30, 30)</td>
</tr>
<tr>
<td>AP-000000000CB@R*8</td>
<td>P</td>
<td></td>
<td>14400</td>
<td>(2, 30, 30)</td>
</tr>
</tbody>
</table>

### Labels

<table>
<thead>
<tr>
<th>Address</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-000000000</td>
<td>50</td>
</tr>
</tbody>
</table>
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
/NODIAGNOSTICS
/NOSTANDARD_DATA
/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
**PROGRAM: IMPLICIT.FOR**  
**AUTHOR: JIM TILSON**  
**DATE: MAY 2, 1990**  
**CLASS: TENNESSEE SCHOLARS, MATH 490**  

This program simulates the effects of a pathogen on a field used to grow two kinds of plants, each with different growth rates and different levels of resistance to the pathogen. It uses successive over-relaxation and Crank-Nicolson's formula to calculate the new concentrations of plant and pathogen at each time step.

**INPUT: INPUT CONSISTS OF PARAMETERS, INPUT BY THE USER, AND A DATA FILE (ALSO SPECIFIED BY THE USER) CONTAINING THE INFORMATION FOR THE PLANTING PATTERN.**

**OUTPUT: OUTPUT IS SENT TO A FILE AT EACH TIME STEP. IT IS FORMATTED FOR USE BY SAS FOR ANY KIND OF STATISTICAL ANALYSIS WHICH THE USER WOULD LIKE TO USE.**

**VARIABLE DECLARATIONS**

```fortran
REAL*8 LT LOOP VARIABLE
REAL*8 TMAX MAXIMUM TIME
REAL*8 DX, DY, DT CHANGE IN X, Y, TIME
REAL*8 A1, A2, ALPHA1, ALPHA2 EQUATION COEFFS.
REAL*8 B1(30, 30) ARRAY FOR PLANT 1
REAL*8 B2(30, 30) ARRAY FOR PLANT 2
REAL*8 P (30, 30) ARRAY FOR PATHOGEN
REAL*8 TBOUND, BBOUND BOUNDARY FUNCTIONS
REAL*8 LBOUND, RBOUND BOUNDARY FUNCTIONS
REAL*8 X, Y COORDS. OF GRID SQUARE
INTEGER*4 LX, LY LOOP VARIABLES
INTEGER*4 NO_PTS NUMBER OF GRID SQUARES TO A SIDE
CHARACTER*10 OFNAME OUTPUT FILE NAME
CHARACTER*10 IFNAME INPUT FILE NAME
```

**INPUT PARAMETERS**

```fortran
PRINT *, 'ENTER INPUT FILE NAME'
READ *, IFNAME
PRINT *, 'ENTER OUTPUT FILE NAME'
READ *, OFNAME
PRINT *, 'ENTER TMAX'
READ *, TMAX
PRINT *, 'ENTER TIME STEP'
READ *, DT
```
IMPLICIT MAIN

0058 PRINT *, 'ENTER A1, A2'
0059 READ *, A1, A2
0060 PRINT *, 'ENTER ALPHA1, ALPHA2'
0061 READ *, ALPHA1, ALPHA2
0062 PRINT *, 'ENTER TB1OUND, BB1OUND, RB1OUND, LB1OUND'
0063 READ *, TB1OUND, BB1OUND, RB1OUND, LB1OUND

OPEN(12, FILE=OFNAME, STATUS='NEW')

0066 INITIALIZE GRID
0070 CALL INITIALIZE GRID(B1, B2, P, NO_PTS, IFNAME)
0071 DX = 1.0 / NO_PTS
0072 DY = 1.0 / NO_PTS

0074 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, TB1OUND, BB1OUND, RB1OUND, LB1OUND, A1, A2, ALPHA1, ALPHA2)
0079 END DO

0083 ITERATIONS FINISHED--OUTPUT FINAL STATE OF FIELD
0083 CALL OUTPUT(B1, B2, P, OFNAME, IFNAME, NO_PTS, A1, A2, ALPHA1, ALPHA2, LT, DX, DY)

0086 TERMINATE PROGRAM
0087 CLOSE(12)
0089 END
### PROGRAM SECTIONS

<table>
<thead>
<tr>
<th>Name</th>
<th>Bytes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CODE</td>
<td>571</td>
<td>PIC CON REL LCL</td>
</tr>
<tr>
<td>$pdata</td>
<td>136</td>
<td>PIC CON REL LCL</td>
</tr>
<tr>
<td>$local</td>
<td>22072</td>
<td>PIC CON REL LCL</td>
</tr>
</tbody>
</table>

**Total Space Allocated:** 22779

### ENTRY POINTS

- 0-00000000 IMPLICIT$MAIN

### VARIABLES

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Name</th>
<th>Address</th>
<th>Type</th>
<th>Name</th>
<th>Address</th>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-000054A0</td>
<td>R*8</td>
<td>A1</td>
<td>2-000054A8</td>
<td>R*8</td>
<td>A2</td>
<td>2-000054B0</td>
<td>R*8</td>
<td>ALPHA1</td>
</tr>
<tr>
<td>2-000054C8</td>
<td>R*8</td>
<td>BBOUND</td>
<td>2-00005498</td>
<td>R*8</td>
<td>DT</td>
<td>2-000054B8</td>
<td>R*8</td>
<td>DX</td>
</tr>
<tr>
<td>2-00005468</td>
<td>CHAR</td>
<td>IFNAME</td>
<td>2-000054D0</td>
<td>R*8</td>
<td>LBOUND</td>
<td>2-00005478</td>
<td>R*8</td>
<td>LT</td>
</tr>
<tr>
<td>2-000054FC</td>
<td>I*4</td>
<td>LBOUND</td>
<td>2-00005500</td>
<td>I*4</td>
<td>NO_PTS</td>
<td>2-00005460</td>
<td>CHAR</td>
<td>OFNAME</td>
</tr>
<tr>
<td>2-000054CD</td>
<td>R*8</td>
<td>TBOUND</td>
<td>2-000054B0</td>
<td>R*8</td>
<td>TMAX</td>
<td>2-000054E0</td>
<td>R*8</td>
<td>X</td>
</tr>
</tbody>
</table>

### ARRAYS

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Name</th>
<th>Bytes</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-00000000</td>
<td>R*8</td>
<td>B1</td>
<td>7200</td>
<td>(30, 30)</td>
</tr>
<tr>
<td>2-00001C20</td>
<td>R*8</td>
<td>B2</td>
<td>7200</td>
<td>(30, 30)</td>
</tr>
<tr>
<td>2-00005840</td>
<td>R*8</td>
<td>P</td>
<td>7200</td>
<td>(30, 30)</td>
</tr>
</tbody>
</table>

### FUNCTIONS AND SUBROUTINES REFERENCED

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FOR$CLOSE</td>
<td></td>
<td>FOR$OPEN</td>
<td></td>
<td>INITIALIZE_GRID</td>
</tr>
</tbody>
</table>
SUBROUTINE OUTPUT:

* THIS SUBROUTINE SENDS DATA FROM THE CURRENT TIME STEP TO THE FILE SPECIFIED BY OFNAME.

* *

SUBROUTINE OUTPUT(B1,B2,P,OFNAME,IFNAME,NO_PTS,A1,A2,ALPHA1,ALPHA2,LT,DX,DY)

* PARAMETERS

REAL*8 B1(30,30), B2(30,30), P(30,30) ARRAYS FOR PLANT, PATHOGEN DATA
CHARACTER*10 OFNAME, IFNAME FILES
INTEGER*4 NO_PTS NUMBER OF POINTS IN GRID
REAL*8 A1, A2, ALPHA1, ALPHA2 GROWTH AND DECAY CONSTANTS
REAL*8 LT CURRENT TIME STEP
REAL*8 DX, DY GRID SQUARE DIMENSIONS

LOCAL VARIABLES

INTEGER*4 LX, LY LOOP VARIABLES
REAL*8 SUMB1, SUMB2, SUMP, SUMB1B2 SUM VARIABLES

* CLEAR SUM VARIABLES
SUMB1 = 0.0
SUMB2 = 0.0
SUMP = 0.0

* ADD DENSITIES
DO LX = 1, NO_PTS
  DO LY = 1, NO_PTS
    SUMB1 = SUMB1 + B1(LX,LY)
    SUMB2 = SUMB2 + B2(LX,LY)
    SUMP = SUMP + P(LX,LY)
  END DO
END DO

* MULTIPLY BY SIZE OF EACH SQUARE GIVING AMOUNT OF MASS
SUMB1 = SUMB1 * (DX*DY)
SUMB2 = SUMB2 * (DX*DY)
SUMP = SUMP * (DX*DY)
SUMB1B2 = SUMB1B2 * (DX*DY)

* SEND TO FILE
WRITE (12, 50) IFNAME, NO_PTS, A1, A2, ALPHA1, ALPHA2, LT, SUMB1, SUMB2, SUMP, SUMB1B2
50 FORMAT (' ', A8, X, I2, X, 9(F8.5, X))

* EXIT TO MAIN PROGRAM
## Program Sections

<table>
<thead>
<tr>
<th>Name</th>
<th>Bytes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 $CODE</td>
<td>348</td>
<td>PIC CON REL LCL SHR EXE RD NOWRT LONG</td>
</tr>
<tr>
<td>1 $PDATA</td>
<td>22</td>
<td>PIC CON REL LCL SHR NOEXE RD NOWRT LONG</td>
</tr>
<tr>
<td>2 $LOCAL</td>
<td>124</td>
<td>PIC CON REL LCL NOSHR NOEXE RD WRT QUAD</td>
</tr>
</tbody>
</table>

Total Space Allocated: 494

## Entry Points

<table>
<thead>
<tr>
<th>Address Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-00000000</td>
<td>OUTPUT</td>
</tr>
</tbody>
</table>

## Variables

<table>
<thead>
<tr>
<th>Address Type</th>
<th>Name</th>
<th>Bytes</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP-00000001@ R*8</td>
<td>A1</td>
<td>ALPHAB1</td>
<td>(30, 30)</td>
</tr>
<tr>
<td>AP-00000002@ R*8</td>
<td>A2</td>
<td>ALPHAB2</td>
<td>(30, 30)</td>
</tr>
<tr>
<td>AP-00000003@ R*8</td>
<td>DX</td>
<td>IFNAME</td>
<td>(30, 30)</td>
</tr>
<tr>
<td>2-00000020</td>
<td>LX</td>
<td>OF NAME</td>
<td>(30, 30)</td>
</tr>
<tr>
<td>2-00000000</td>
<td>R*8 SUMB1</td>
<td>SUMP</td>
<td>(30, 30)</td>
</tr>
</tbody>
</table>

## Arrays

<table>
<thead>
<tr>
<th>Address Type</th>
<th>Name</th>
<th>Bytes</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP-00000004@ R*8</td>
<td>B1</td>
<td>7200</td>
<td>(30, 30)</td>
</tr>
<tr>
<td>AP-00000005@ R*8</td>
<td>B2</td>
<td>7200</td>
<td>(30, 30)</td>
</tr>
<tr>
<td>AP-00000006@ R*8</td>
<td>P</td>
<td>7200</td>
<td>(30, 30)</td>
</tr>
</tbody>
</table>

## Labels

<table>
<thead>
<tr>
<th>Address</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-00000000</td>
<td>50'</td>
</tr>
</tbody>
</table>
SUBROUTINE INITIALIZE:

THIS SUBROUTINE TAKES ITS INPUT FROM A FILE SPECIFIED BY THE USER. THIS INPUT IS THE INITIAL PLANTING PATTERN FOR THE FIELD, CONSISTING OF:

LINE 1: NUMBER OF POINTS IN GRID (30 MAX.)

SUBSEQUENT LINES: X AND Y RANGES, CONCENTRATIONS OF PLANTS AND PATHOGEN IN RECTANGULAR BOX FORMED BY X AND Y COORDINATES

LAST LINE: CONTAINS SENTINEL 999 999 999 999 999 999 999 TO DENOTE END OF FILE

SUBROUTINE INITIALIZE_GRID(B1, B2, P, NO_PTS, IFNAME)

PARAMETERS

REAL*8 B1(30,30), B2(30,30), P(30,30) ARRAYS WITH PLANT AND PATHOGEN DATA
INTEGER*4 NO_PTS NUMBER OF POINTS IN GRID
CHARACTER*10 IFNAME INPUT FILE NAME

LOCAL VARIABLES

INTEGER*2 X, Y, XMN, XMX, YMN, YMX X AND Y BOUNDARIES FOR EACH DATA LINE
REAL*8 B1VAL, B2VAL, PVAL CONCENTRATIONS OF PLANT, PATHOGEN AT EACH SQUARE

OPEN THE FILE
OPEN (13, FILE=IFNAME, STATUS='OLD')

PRIMING READ
READ (13,*) NO_PTS
READ (13,*) XMN, XMX, YMN, YMX, B1VAL, B2VAL, PVAL

MAIN LOOP
DO WHILE (XMN .NE. 999)
DO X=XMN, XMX
DO Y=YMN, YMX
B1(X,Y) = B1VAL
B2(X,Y) = B2VAL
P (X,Y) = PVAL
END DO
END DO
READ (13,*) XMN, XMX, YMN, YMX, B1VAL, B2VAL, PVAL
END DO

CLOSE FILE AND RETURN TO MAIN
CLOSE(13)
**INITIALIZE_GRID**

**PROGRAM SECTIONS**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bytes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CODE</td>
<td>398</td>
<td>PIC CON REL LCL SHR EXE RD NOWRT LONG</td>
</tr>
<tr>
<td>$LOCAL</td>
<td>140</td>
<td>PIC CON REL LCL NOSHR NOEXE RD WRT QUAD</td>
</tr>
</tbody>
</table>

Total Space Allocated 538

**ENTRY POINTS**

<table>
<thead>
<tr>
<th>Address Type Name</th>
<th>Address Type Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-0000000000</td>
<td>INITIALIZE_GRID</td>
</tr>
</tbody>
</table>

**VARIABLES**

<table>
<thead>
<tr>
<th>Address Type Name</th>
<th>Address Type Name</th>
<th>Address Type Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-000000010</td>
<td>R*B</td>
<td>B1VAL</td>
</tr>
<tr>
<td>2-000000020</td>
<td>R*B</td>
<td>PVAL</td>
</tr>
<tr>
<td>2-000000002</td>
<td>I*2</td>
<td>Y</td>
</tr>
<tr>
<td>2-000000018</td>
<td>R*B</td>
<td>B2VAL</td>
</tr>
<tr>
<td>2-0000000000</td>
<td>I*2</td>
<td>X</td>
</tr>
<tr>
<td>2-000000008</td>
<td>I*2</td>
<td>YMN</td>
</tr>
<tr>
<td>2-000000014</td>
<td>CHAR</td>
<td>IFNAME</td>
</tr>
<tr>
<td>2-000000004</td>
<td>I*2</td>
<td>XMN</td>
</tr>
<tr>
<td>2-000000006</td>
<td>I*2</td>
<td>XMX</td>
</tr>
</tbody>
</table>

**ARRAYS**

<table>
<thead>
<tr>
<th>Address Type Name</th>
<th>Address Type Name</th>
<th>Bytes Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP-000000004@</td>
<td>R*B</td>
<td>7200 (30, 30)</td>
</tr>
<tr>
<td>AP-000000005@</td>
<td>R*B</td>
<td>7200 (30, 30)</td>
</tr>
<tr>
<td>AP-000000006@</td>
<td>R*B</td>
<td>7200 (30, 30)</td>
</tr>
</tbody>
</table>

**FUNCTIONS AND SUBROUTINES REFERENCED**

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Type Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOR$CLOSE</td>
<td>FOR$OPEN</td>
</tr>
</tbody>
</table>
**SUBROUTINE ITERAT:**

This subroutine calculates information at the (n+1)th time step given the configuration of the field at the nth time step. It uses successive over-relaxation, Crank-Nicolson's scheme.

```fortran
SUBROUTINE ITERAT (B1,B2,P,NO_PTS,DX,DY,DT,TBOUND,BBOUND,LBOUND,RBOUND,A1,A2,ALPHA1,ALPHA2)

PARAMETERS
REAL*8 B1(30,30),B2(30,30),P(30,30) ! PLANT, PATHOGEN DATA
INTEGER*4 NO_PTS ! NUMBER OF POINTS IN GRID
REAL*8 DX,DY,DT ! DIFFERENTIALS
REAL*8 TBOUND,BBOUND,LBOUND,RBOUND ! BOUNDARY FUNCTIONS
REAL*8 A1,A2,ALPHA1,ALPHA2 ! GROWTH COEFFICIENTS

LOCAL VARIABLES
REAL*8 ACCERR ! ACCEPTABLE ERROR
REAL*8 W ! 0 < W < 2 FOR SOR METHOD
REAL*8 B10(30,30),B20(30,30),P0(30,30) ! PREVIOUS GUESS FOR (N+1)TH TIME STEP
REAL*8 B1N(30,30),B2N(30,30),PN(30,30) ! NEXT GUESS FOR (N+1)TH TIME STEP
REAL*8 R,K1,K2 ! COEFFICIENTS TO TIDY UP FORMULAS
REAL*8 B1ERR, B2ERR, PERR ! ERROR BETWEEN GUESSES
INTEGER*4 X,Y ! LOOP VARIABLES
LOGICAL ACCFLG ! ACCEPTANCE FLAG

COPY MATRIX FOR OPERATIONS
DO X=1,NO_PTS
  DO Y=1,NO_PTS
    B1(X,Y) = B1(X,Y)
    B2(X,Y) = B2(X,Y)
    P(X,Y)  = P(X,Y)
  END DO
END DO

SET VARIABLES FOR ITERATIONS
ACCFLG = .FALSE.
R = DT/(DX**2) ! NEXT GUESS FOR (N+1)TH TIME STEP
ACCERR = 0.001
W = 1.35

MAIN LOOP
DO WHILE (.NOT. (ACCFLG))
  DO Y=1,NO_PTS ! RESET ERROR COUNTERS
    B1ERR = 0.0
    B2ERR = 0.0
    PERR  = 0.0
  END DO

... (continued subroutine body) ...
```
CALCULATE PLANT MASS AT N+1TH TIME STEP

DO X = 1, NO_PTS
  DO Y = 1, NO_PTS
    K1 = 2 + DT * (A1 - ALPHA1 * P(X,Y))
    K2 = 2 + DT * (-A1 + ALPHA1 * PO(X,Y))
    B1N(X,Y) = B10(X,Y) + (W * (K1 * B1(X,Y) - K2 * B10(X,Y)) / K2)
    B1ERR = B1ERR + (B1N(X,Y) - B10(X,Y)) ** 2
  ENDDO
ENDDO

PATHOGEN CALCULATIONS

BOTTOM LEFT CORNER
K1 = 2 - (2 * R) + (DT * (ALPHA1*B1(1,1) + ALPHA2*B2(1,1)))
K2 = 2 + (2 * R) - (DT * (ALPHA1*B1N(1,1) + ALPHA2*B2N(1,1)))
P1N(1,1) = P0(1,1) + (W * (R * (DX*(BOUND + BBOUND) + P(2,1) + P(1,2) + P2(1,1) + P0(1,2)))
  + (K1 * P(1,1)) - (K2 * P0(1,1)) / K2)
PERR = PERR + (P1N(1,1) - P0(1,1)) ** 2

BOTTOM MIDDLE ROW
DO X = 2, NO_PTS - 1
  K1 = 2 - (3 * R) + (DT * (ALPHA1*B1(X,1) + ALPHA2*B2(X,1)))
  K2 = 2 + (3 * R) - (DT * (ALPHA1*B1N(X,1) + ALPHA2*B2N(X,1)))
P1N(X,1) = P0(X,1) + (W * (R * (DX*BOUND + BBOUND) + P(X-1,1) + P(X,2) + P(X+1,1) + P0(X-1,1) + P0(X,2) + P0(X+1,1))
  + (K1 * P(X,1)) - (K2 * P0(X,1)) / K2)
PERR = PERR + (P1N(X,1) - P0(X,1)) ** 2
ENDDO

BOTTOM RIGHT CORNER
K1 = 2 - (2 * R) + (DT * (ALPHA1*B1(NO_PTS,1) + ALPHA2*B2(NO_PTS,1)))
K2 = 2 + (2 * R) - (DT * (ALPHA1*B1N(NO_PTS,1) + ALPHA2*B2N(NO_PTS,1)))
P0N(NO_PTS,1) = P0(NO_PTS,1) + (W * (R * (DX*BOUND + BBOUND) + P(NO_PTS-1,1) + P0(NO_PTS-1,1) + P0(NO_PTS,2) + P0(NO_PTS,2))
  + (K1 * P(NO_PTS,1)) - (K2 * P0(NO_PTS,1)) / K2)
PERR = PERR + (P0N(NO_PTS,1) - P0(NO_PTS,1)) ** 2

MIDDLE ROWS
DO Y = 2, NO_PTS - 1
  LEFT EDGE
    K1 = 2 - (3 * R) + (DT * (ALPHA1*B1(1,Y) + ALPHA2*B2(1,Y)))
    K2 = 2 + (3 * R) - (DT * (ALPHA1*B1N(1,Y) + ALPHA2*B2N(1,Y)))
P1N(1,Y) = P0(1,Y) + (W * (R * (DX*BOUND + P(2,Y) + P(1,Y-1) + P(1,Y+1) + P0(2,Y) + P0(1,Y-1) + P0(1,Y+1))
  + (K1 * P(1,Y)) - (K2 * P0(1,Y)) / K2)
PERR = PERR + (P1N(1,Y) - P0(1,Y)) ** 2
ENDDO

INSIDE MIDDLE
DO X = 2, NO_PTS - 1
   K1 = 2 - (4 * R) + (DT * (ALPHA1*B1(X,Y) + ALPHA2*B2(X,Y)))
   K2 = 2 + (4 * R) - (DT * (ALPHA1*B1N(X,Y) + ALPHA2*B2N(X,Y)))
   PN(X,Y) = PO(X,Y) + (W * (R * (P(X-1,Y) + P(X+1,Y) + P(X,Y-1) + P(X,Y+1))
   + PN(X-1,Y) + PN(X,Y-1) + PO(X+1,Y) + PO(X,Y+1))
   + (K1 * P(X,Y) - (K2 * PO(X,Y))) / K2)
   PERR = PERR + (PN(X,Y) - PO(X,Y)) ** 2
END DO

* RIGHT EDGE
K1 = 2 - (3 * R) + (DT * (ALPHA1*B1(NO_PTS,Y) + ALPHA2*B2(NO_PTS,Y)))
K2 = 2 + (3 * R) - (DT * (ALPHA1*B1N(NO_PTS,Y) + ALPHA2*B2N(NO_PTS,Y)))
PN(NO_PTS,Y) = PO(NO_PTS,Y) + (W * (R * (DX*RBND + P(NO_PTS-1,Y) + P(NO_PTS,Y-1) + P(NO_PTS,Y+1)
   + PN(NO_PTS-1,Y) + PN(NO_PTS,Y-1) + PO(NO_PTS,Y+1))
   + (K1 * P(NO_PTS,Y) - (K2 * PO(NO_PTS,Y))) / K2)
   PERR = PERR + (PN(NO_PTS,Y) - PO(NO_PTS,Y)) ** 2
END DO

* TOP LEFT CORNER
K1 = 2 - (2 * R) + (DT * (ALPHA1*B1(1,NO_PTS) + ALPHA2*B2(1,NO_PTS)))
K2 = 2 + (2 * R) - (DT * (ALPHA1*B1N(1,NO_PTS) + ALPHA2*B2N(1,NO_PTS)))
PN(1,NO_PTS) = PO(1,NO_PTS) + (W * (R * (DX*(TBOUND + LBND) + P(2,NO_PTS) + P(1,NO_PTS-1)
   + PO(2,NO_PTS) + PN(1,NO_PTS-1))
   + (K1 * P(1,NO_PTS) - (K2 * PO(1,NO_PTS))) / K2)
   PERR = PERR + (PN(1,NO_PTS) - PO(1,NO_PTS)) ** 2

* TOP MIDDLE ROW
DO X = 2, NO_PTS - 1
   K1 = 2 - (3 * R) + (DT * (ALPHA1*B1(X,NO_PTS) + ALPHA2*B2(X,NO_PTS)))
   K2 = 2 + (3 * R) - (DT * (ALPHA1*B1N(X,NO_PTS) + ALPHA2*B2N(X,NO_PTS)))
   PN(X,NO_PTS) = PO(X,NO_PTS) + (W * (R * (DX*TBOUND + LBND) + P(X-1,NO_PTS) + P(X+1,NO_PTS) + P(X,NO_PTS-1)
   + PN(X-1,NO_PTS) + PN(X,NO_PTS-1) + PO(X+1,NO_PTS))
   + (K1 * P(X,NO_PTS) - (K2 * PO(X,NO_PTS))) / K2)
   PERR = PERR + (PN(X,NO_PTS) - PO(X,NO_PTS)) ** 2
END DO

* TOP RIGHT CORNER
K1 = 2 - (2 * R) + (DT * (ALPHA1*B1(NO_PTS,NO_PTS) + ALPHA2*B2(NO_PTS,NO_PTS)))
K2 = 2 + (2 * R) - (DT * (ALPHA1*B1N(NO_PTS,NO_PTS) + ALPHA2*B2N(NO_PTS,NO_PTS)))
PN(NO_PTS,NO_PTS) = PO(NO_PTS,NO_PTS) + (W * (R * (DX*(TBOUND + RBND) + P(NO_PTS-1,NO_PTS) + P(NO_PTS,NO_PTS-1)
   + PN(NO_PTS-1,NO_PTS) + PN(NO_PTS,NO_PTS-1))
   + (K1 * P(NO_PTS,NO_PTS) - (K2 * PO(NO_PTS,NO_PTS))) / K2)
   PERR = PERR + (PN(NO_PTS,NO_PTS) - PO(NO_PTS,NO_PTS)) ** 2

* FIND 2-NORM OF ERROR TERMS--SQUARE ROOT OF SUM OF DIFFERENCE OF SQUARES
B1ERR = B1ERR ** 0.5
B2ERR = B2ERR ** 0.5
PERR = PERR ** 0.5

* WITHIN ACCEPTABLE ERROR?
IF ((B1ERR .LE. ACCERR) .AND. (B2ERR .LE. ACCERR) .AND. (PERR .LE. ACCERR)) THEN
    YES--SET FLAG FOR EXIT FROM SUBROUTINE
    ACCFLG = .TRUE.
ELSE
    NO--UPDATE NEW GUESS AND TRY AGAIN
    DO X = 1, NO_PTS
        DO Y = 1, NO_PTS
            B10(X,Y) = B1N(X,Y)
            B20(X,Y) = B2N(X,Y)
            P0(X,Y) = PN(X,Y)
        END DO
    END DO
    END IF
    END DO

ITERATIONS FINISHED FOR THIS TIME STEP--RETURN NEW STATUS
    DO X = 1, NO_PTS
        DO Y = 1, NO_PTS
            B1(X,Y) = B1N(X,Y)
            B2(X,Y) = B2N(X,Y)
            P(X,Y) = PN(X,Y)
        END DO
    END DO
    END DO

RETURN TO MAIN PROGRAM
END
## PROGRAM SECTIONS

<table>
<thead>
<tr>
<th>Name</th>
<th>Bytes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 $CODE</td>
<td>3582</td>
<td>PIC CON REL LCL SHR EXE RD N0WRT LONG</td>
</tr>
<tr>
<td>2 $LOCAL</td>
<td>43384</td>
<td>PIC CON REL LCL N0SR N0EXE RD WRT QUAD</td>
</tr>
</tbody>
</table>

Total Space Allocated: 46966

## ENTRY POINTS

<table>
<thead>
<tr>
<th>Address Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-00000000</td>
<td>ITERAT</td>
</tr>
</tbody>
</table>

## VARIABLES

### Address Type Name

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Name</th>
<th>Bytes Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP-0000000030@</td>
<td>R*8</td>
<td>A1</td>
<td>2-00000000 (30, 30)</td>
</tr>
<tr>
<td>AP-0000000038@</td>
<td>R*8</td>
<td>ALPHA1</td>
<td>2-00000000 (30, 30)</td>
</tr>
<tr>
<td>AP-0000000024@</td>
<td>R*8</td>
<td>BBOUND</td>
<td>2-00000000 (30, 30)</td>
</tr>
<tr>
<td>2-0000A8DB@</td>
<td>R*8</td>
<td>K1</td>
<td>2-0000A8DO (30, 30)</td>
</tr>
<tr>
<td>2-0000A8F0@</td>
<td>R*8</td>
<td>PERR</td>
<td>2-0000A8F0 (30, 30)</td>
</tr>
<tr>
<td>2-0000A8BC@</td>
<td>R*8</td>
<td>W</td>
<td>2-0000A900 (30, 30)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Name</th>
<th>Bytes Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP-0000000034@</td>
<td>R*8</td>
<td>A2</td>
<td>2-00000000 (30, 30)</td>
</tr>
<tr>
<td>AP-000000003C@</td>
<td>R*8</td>
<td>ALPHA2</td>
<td>2-00000000 (30, 30)</td>
</tr>
<tr>
<td>AP-000000001C@</td>
<td>R*8</td>
<td>DT</td>
<td>2-00000000 (30, 30)</td>
</tr>
<tr>
<td>2-0000A8E8@</td>
<td>R*8</td>
<td>K2</td>
<td>2-0000A8FO (30, 30)</td>
</tr>
<tr>
<td>2-0000A8DO@</td>
<td>R*8</td>
<td>R</td>
<td>2-0000A904 (30, 30)</td>
</tr>
</tbody>
</table>

## ARRAYS

### Address Type Name

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Name</th>
<th>Bytes Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP-000000004@</td>
<td>R*8</td>
<td>B1</td>
<td>2-00000000 (30, 30)</td>
</tr>
<tr>
<td>2-00005460@</td>
<td>R*8</td>
<td>B1N</td>
<td>2-00005460 (30, 30)</td>
</tr>
<tr>
<td>2-00000000@</td>
<td>R*8</td>
<td>B10</td>
<td>2-00000000 (30, 30)</td>
</tr>
<tr>
<td>AP-000000008@</td>
<td>R*8</td>
<td>B2</td>
<td>2-00000000 (30, 30)</td>
</tr>
<tr>
<td>2-0000A8D8@</td>
<td>R*8</td>
<td>P</td>
<td>2-0000A8D8 (30, 30)</td>
</tr>
<tr>
<td>AP-00000000C@</td>
<td>R*8</td>
<td>P</td>
<td>2-00000000 (30, 30)</td>
</tr>
<tr>
<td>2-0000A8F0@</td>
<td>R*8</td>
<td>PN</td>
<td>2-0000A8F0 (30, 30)</td>
</tr>
<tr>
<td>2-0000A8BC@</td>
<td>R*8</td>
<td>PO</td>
<td>2-0000A8BC (30, 30)</td>
</tr>
</tbody>
</table>
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,NOUNTRIX,NOVAXELN)
/CONTINUATIONS=19 /NOCROSS_REFERENCE /NOD_LINES /EXTEND_SOURCE
/F77 /NOL_FLOATING /14 /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