A BASIC Program To Simulate Plane Wave Reflection

STEVE JEWELL
Advanced Measurement Systems, Inc., Fairfax, VA*

INTRODUCTION

Many problems in nature can be understood more readily using the computer. Determining how RF travels in a waveguide corridor is one such problem. Anechoic materials and a reflective wall pattern are typically used to provide the required shielding effectiveness: but how?

Many variables must be defined before the simulation can begin. The waveguide corridor dimensions, anechoic material characteristics, properties of plane wave propagation, and any physical space constraints should be known. Ideally, an artificial intelligence program could be created which would provide the perfect corridor for every installation.

To that end, a BASIC program is provided which models plane wave reflection (Program starts on page 64). Although it ignores absorption, the program does provide a rough basis for comparison of alternative floor plans and wall structures. Hopefully, the program will provide a basis for the design of future waveguide corridors which are smaller, cheaper, and have a higher shielding effectiveness.

THE PROGRAM

What It Does. The program, which executes on IBM or IBM compatibles with GWBASIC, allows the user to input a scaled wall configuration for evaluation. Next, an input vector (or series of vectors) which simulate the input RF are entered. One by one, the input vectors

A BASIC program which models plane wave propagation in a reflective space can assist designers of waveguide corridors to make estimates of shielding effectiveness.

are sent into the corridor until they are reflected out, or strike a pre-determined receive "wall." The direction of the vectors which succeed in navigating through the corridor to the receive wall are displayed at the conclusion of the simulation. All wall, vector, and output files can be stored for future retrieval.

What It Does Not Do. The program presently does not simulate the effects of anechoic materials in the corridor. Furthermore, it assumes perfect reflections from the walls, which may not be accurate with thin wall shields. The actual shielding effectiveness is not presented, since plane wave propagation is media dependent. The model further assumes that diffraction (the tendency of the plane wave "beam" to broaden) is insignificant, which is true only for far field conditions (transmit antenna at least 20 wavelengths from corridor). Finally, simulation accuracy is limited to the experience of the designer, the number of simulations run, and the ability of the installer to follow detailed plans.

What It Could Do. As is, the program is limited. However, it could easily be integrated into other programs to solve users' specific requirements. For example, the program could be modified to accommodate the addition of anechoic materials into the corridor. In addition, amplitude versus frequency/beamwidth data could be derived from the output data. Since the propagation characteristics of the beam are not defined, other plane wave phenomenon could also be modeled. Sound and light can be modeled this way, as long as they are present as plane waves, and the walls are reflective. In its compiled form, the program can be linked with other processing programs written in other languages.

PROGRAM INSTRUCTIONS

The program presented herein is called REFLECT.10. In order to run the program, an IBM PC (or compatible), with 512k of memory, DOS 2.0 or higher, GWBASIC, and a graphics compatible monitor are needed. Although an 8088 based machine can run REFLECT.10, 286- or 386-based machines will perform the larger simulations much faster.

Preparation. Power up your machine and load BASIC (your DOS or BASIC manuals can assist you in this). Place a blank, formatted disk in your "A" drive and you're ready to run. Using the provided program listing, enter the program and save it under REFLECT.10.

*See Advertisement on Page 333.

REFLECT.10 PROGRAM

```
500 REM **          REFLECT.10        S.Jewell/AMS Fairfax, VA          **
510 REM **   This program takes input wall endpoints and input vector(s)  **
520 REM **   and draws the vectors' path across the screen.               **
530 REM **   VERSION 1.0 - QUICKBASIC dimensioning, w/CGA/EGA/VGA display **
540 REM
550 DIM WSX(100),WSY(100),WEX(100),WEY(100),WM(100),WB(100)' wall data
560 DIM RDIST(100),VX(100),VY(100),VM(1000),VB(1000)      ' Distance and
570 DIM VS(1000),VSY(1000),VSX(1000)                      ' vector data
580 DIM VSX1(1000),VSY1(1000),VM1(1000),VB1(1000),VS1(1000)' vector data
590 SCREEN 2:KEY OFF                                       ' set screen mode
600 PI = 3.14159:DISTANCE=0:NV=0:MAXDISTANCE=100000!  'initialize variables
610 ROUNDERR=.0004     '*******     ROUNDOFF ERROR VARIABLE
615 DELA = 3000        '*******     DELAY TIME VARIABLE (3000 FOR 8088 CPU)
620 BLANKLINE$="                                                      "
630 BIGBLANK$ = "
        "
640 I=1:MAXX=1000:MAXY=1000:PINTERSECT=999:DUM=0
650 WINDOW (0,0) - (MAXX,MAXY)
660 REM **  This area reserved for user defined variables                **
690 PINTERSECT=999:DISTANCE = 0:CLS:LOCATE 1,1:PRINT"  REFLECT .10"
700 LOCATE 10,18:PRINT "MAIN MENU"
710 LOCATE 12,15:PRINT "1. Create a wall file"
720 LOCATE 13,15:PRINT "2. Retrieve a wall file"
730 LOCATE 14,15:PRINT "3. Run Simulation"
740 LOCATE 15,15:PRINT "4. Help Menu"
750 LOCATE 16,15:PRINT "5. Exit"
760 LOCATE 20,15:INPUT "Enter Selection Number ==> ",NU
770 ON NU GOTO 1000, 2000, 3000, 6000, 7000
1000 CLS:I=1:WR=0                  ' ** WALL FILE CREATION SUBROUTINE **
1010 FOR JJ=1 TO 8
1020 LOCATE JJ,1:PRINT BLANKLINE$
1030 NEXT JJ
1040 LOCATE 1,1:INPUT "Enter the starting x coordinate  ",WSX(I)
1060 LOCATE 2,1:INPUT "Enter the starting y coordinate  ",WSY(I)
1080 LOCATE 3,1:INPUT "Enter the ending x coordinate  ",WEX(I)
1100 LOCATE 4,1:INPUT "Enter the ending y coordinate  ",WEY(I)
1120 IF WSX(I)<WEX(I) THEN 1150
1130 TEMP = WEX(I):WEX(I)=WSX(I):WSX(I)=TEMP
1140 TEMP = WEY(I):WEY(I)=WSY(I):WSY(I)=TEMP  'starting x must be < ending x
1150 LINE (WSX(I),WSY(I)) - (WEX(I),WEY(I))   'draws the line just input
1160 I=I+1
1170 LOCATE 5,1:PRINT "Enter an 'R' if this is the"
1180 LOCATE 6,1:INPUT "receive wall (return to continue) ",R$
1190 IF R$="R" OR R$="r" THEN 1210
1200 GOTO 1220
1210 WR=I-1                   'Keeps track of which wall is the receive wall
1220 INPUT "Enter an 'X' to exit wall entry ",X$
1230 IF X$="X" OR X$="x" THEN 1250
1240 GOTO 1010
1250 IMAX = I-1:LOCATE 7,1:PRINT "Enter Filename (8 digits max.)  "
1260 LOCATE 8,1:INPUT "for storage ... ";FM1$
1265 IF LEN(FM1$)>8 THEN 1250
1270 OPEN "a:"+FM1$+".wal" FOR OUTPUT AS #1      ' ** STORES WALL FILE **
1280 FOR EZ=1 TO IMAX
1290 WRITE #1, WSX(EZ),WSY(EZ),WEX(EZ),WEY(EZ)
1300 NEXT EZ
1310 WRITE #1,999999!,0,0,0    ' 999999 is end of wall data indicator
1320 WRITE #1,WR
1330 CLOSE #1
1340 GOTO 690
2000 REM   ******   Retrieve File Subroutine   *******
2010 CLS:FILES "a:*.wal"
2020 LOCATE 24,10:INPUT "Enter filename for retrieval ... ",RF$
2022 IF RIGHT$(RF$,4)=".wal" OR RIGHT$(RF$,4)=".WAL" THEN 2030
2025 LOCATE 24,10:PRINT "Please add '.wal' extension to filename         "
2027 FOR CC=1 TO DELA:NEXT CC
2028 GOTO 2020
2030 OPEN "a:"+RF$ FOR INPUT AS #1
2040 FOR RT = 1 TO 1000
2050 INPUT #1, WSX(RT),WSY(RT),WEX(RT),WEY(RT)
2060 IF WSX(RT)=999999! THEN 2080           ' Detects end of receive file
2070 NEXT RT
2080 INPUT #1,WR                            ' Accepts receive wall value
2090 IMAX=RT-1
2100 CLOSE #1
2110 GOTO 690
3000 CLS:J=1
3010 IF IMAX>0 THEN 3040
3020 PRINT "Please create or retrieve file first"  ' Error trap for no wall
3030 FOR DRE =1 TO DELA:NEXT DRE:GOTO 690          ' file (includes delay)
3040 FOR QQ = 1 TO IMAX
3050 LINE (WSX(QQ),WSY(QQ)) - (WEX(QQ),WEY(QQ))    ' Draws wall file
3060 NEXT QQ
3080 LOCATE 1,15:PRINT "*****   SIMULATION ROUTINE   *****"
3090 LOCATE 3,5:TOTDIST=0
3100 INPUT "Choose retrieved (R), or newly created (C) vector file .. ",M$
3110 IF M$="c" OR M$="C" THEN 3260
3130 IF M$="r" OR M$="R" THEN 3140
3135 LOCATE 3,62:PRINT "     ";:GOTO 3090
3140 CLS:FILES "a:*.vf"         ' ** VECTOR FILE RETRIEVAL SUBROUTINE **
3150 LOCATE 22,10:INPUT "Enter vector file name for retrieval .. ",VFNM$
3152 IF RIGHT$(VFNM$,3)=".VF" OR RIGHT$(VFNM$,3)=".vf" THEN 3160
3155 LOCATE 22,10:PRINT "Please add '.VF' extension to filename          "
```

Alternatively, copies of the listed program can be obtained as described in the **Tips on Use** section given below. Now run the program by pressing the "F2" key. If problems occur, see the **Tips on Use** section or refer to your BASIC manual for debugging suggestions. Otherwise, you should see the main menu (Figure 1). Options are summarized below.

```
REFLECT .10   MAIN MENU

1.   Create a Wall File
2.   Retrieve a Wall File
3.   Run Simulation
4.   Help Menu
5.   Exit

Enter Selection Number==>
```

FIGURE 1. *The main menu for the REFLECT.10 program. The user selects an option by entering the number corresponding to the desired action, and pressing the "return" key.*

1. *Create a Wall File.* This selection is used to describe the reflective surface configuration. For waveguide corridors, it is used to describe the shielded wall locations. The walls are entered by successive prompts for the starting and ending x coordinate (horizontal), and the starting and ending y coordinate (vertical). The wall inputs are scaled so that the end points are from 1 to 999, with the point 999,999 in the upper right-hand portion of the screen. The wall configuration is displayed graphically as it is entered, and the user is prompted to provide the location of the receive wall. After the last wall is entered, an eight-digit file name must be provided to save the wall file.
2. *Retrieve a Wall File.* When this option is chosen, all wall files which have been created and saved to the "A"

REFLECT.10 Program...Continued

```
3157 FOR CC=1 TO DELA:NEXT CC
3158 GOTO 3150
3160 OPEN "a:"+VFNM$ FOR INPUT AS #1
3170 INPUT #1, TYPEVEC
3175 INPUT #1,VSX1,VSY1,SA,EA,NV,VM1(1),VB1(1),VS1(1),DUM,DUM,DUM
3180 IF TYPEVEC=0 THEN 3220      ' Typevec = 0 for single-vector files
3190 FOR EZ=2 TO NV:
3200 INPUT #1, VSX1,VSY1,SA,EA,NV,VM1(EZ),VB1(EZ),VS1(EZ),DUM,DUM,DUM
3210 NEXT EZ
3220 CLOSE #1
3230 FOR QQ = 1 TO IMAX
3240 LINE (WSX(QQ),WSY(QQ)) - (WEX(QQ),WEY(QQ))     ' Redraw wall file
3250 NEXT QQ:GOTO 3610
3260 CLS:LOCATE 1,15:PRINT "Vector Sweep File Creation"
3270 LOCATE 3,12:INPUT "Enter the starting x coordinate  ",VSX1
3280 LOCATE 4,12:INPUT "Enter the starting y coordinate  ",VSY1
3290 LOCATE 5,12:INPUT "Enter the starting angle in degrees ",SA
3300 LOCATE 6,12:INPUT "Enter the ending angle in degrees ",EA
3310 IF SA<EA THEN 3330
3320 TEMP = EA:EA=SA:SA=TEMP  'starting angle must be < ending x
3330 IF SA<>EA THEN 3350
3340 NV=1:THETAINC=0:GOTO 3370 ' still needs vm/vb calculations
3350 LOCATE 15,12:INPUT "Enter the number of vectors drawn ",NV
3360 THETAINC = (EA-SA)/(NV-1)
3370 FOR W=1 TO NV         ' **** Subroutine converts to slope delta format  ***
3380 ANGLE = SA + ((W-1)*THETAINC)
3390 IF ANGLE>(90) THEN 3410
3400 VM1(W)=TAN((ANGLE*PI)/180):VB1(W)=VSY1 - VM1(W)*VSX1:VS1(W)=1:GOTO 3420
3410 VM1(W)=TAN((ANGLE*PI)/180):VB1(W)=VSY1 - VM1(W)*VSX1:VS1(W)=0
3420 NEXT W
3430 LOCATE 17,15:PRINT "Enter Filename for  "
3440 LOCATE 18,15:INPUT "storage (8 letters max.) ";FM1$
3450 OPEN "a:"+FM1$+".vf" FOR OUTPUT AS #1:TYPEVEC=1
3460 WRITE #1, TYPEVEC
3470 FOR EZ=1 TO NV
3480 WRITE #1, VSX1,VSY1,SA,EA,NV,VM1(EZ),VB1(EZ),VS1(EZ),DUM,DUM,DUM
3490 NEXT EZ
3500 CLOSE #1
3610 BB=1:IF TYPEVEC=0 THEN 3640      'BB counts the number of vectors done
3620 FOR BB=1 TO NV
3630 VSX(1)=VSX1:VSY(1)=VSY1:VM(1)=VM1(BB):      'Initializes loop variables
3635 VB(1)=VB1(BB):VS(1)=VS1(BB):DCOUNT=0:J=1    'for next input vector
3640 CLS:FOR QQ = 1 TO IMAX
3650 LINE (WSX(QQ),WSY(QQ)) - (WEX(QQ),WEY(QQ))
3660 NEXT QQ
3670 ' *************     Begin single vector loop     ***************
3680 IF VM(J)<99999! THEN 3700
3690 VTHETA = PI/2+.00001:' error trap for straight up
3700 VTHETA = ATN(VM(J)):LOCATE 1,1:IF VTHETA<>0 THEN 3720
3710 VTHETA =.00001 'arbitrarily assigned value for lines straight right/left
3720 IF VM(J)>0 THEN 3780 ' adjusts vtheta value depending on sense
3730 IF VS(J)=0 THEN 3760
3740 VTHETA = 2*PI + VTHETA:GOTO 3800
3760 VTHETA = PI + VTHETA:GOTO 3800
3780 IF VS(J)=1 THEN 3800
3790 VTHETA = PI + VTHETA
3800 LOCATE 2,1:PRINT "vector # ";BB;" of ";NV
3810 FOR A=1 TO IMAX               'converts all walls to slope/delta format
3820 IF WEX(A)<>WSX(A) THEN 3840
3830 WM(A)= 99999!:GOTO 3870             ' error trap for straight up
3840 IF WEY(A)<>WSY(A) THEN 3860
3850 WM(A)=.00001:GOTO 3870              ' error trap for sideways
3860 WM(A)= (WEY(A)-WSY(A))/(WEX(A)-WSX(A))
3870 WB(A)= WSY(A) - WM(A)*WSX(A)
3880 NEXT A
3900 REM  ****  Iteration loop for reflection path  ****
3920 FOR Z=1 TO IMAX
3930 IF VM(J)<>WM(Z) THEN 3950
3940 WM(Z)=WM(Z)+.0001
3950 VX(Z)=(WB(Z)-VB(J))/(VM(J)-WM(Z))
3960 VY(Z)=VM(J)*VX(Z)+VB(J)
3970 NEXT Z
3980 VMAX=MAXX               'This section finds the closest intersecting line
3990 RDIST(BB)=0             'Initializes receive distance to 0
4000 IF VS(J)=1 THEN 4210
4020 FOR Q1=1 TO IMAX
4030 DIF=(VSX(J)+(MAXX*ROUNDERR/100))-VX(Q1)      ' roundoff compensation
4040 IF Q1=PINTERSECT THEN 4170   ' PINTERSECT is the previously intersected
4050 IF DIF<=0 THEN 4170          ' wall.  This keeps roundoff errors from
4060 IF DIF>VMAX THEN 4170        ' causing false reflections.
4070 IF Q1=PINTERSECT THEN 4170
4080 IF (WSX(Q1)-(MAXX*ROUNDERR))>VX(Q1) THEN 4170 'checks for start/end point
4090 IF (WEX(Q1)+(MAXX*ROUNDERR))<VX(Q1) THEN 4170 'on line (w/roundoff error)
4100 IF WEY(Q1)>WSY(Q1) THEN 4140
4110 IF (WSY(Q1)+(MAXY*ROUNDERR))<VY(Q1) THEN 4170
4120 IF (WEY(Q1)-(MAXY*ROUNDERR))>VY(Q1) THEN 4170   'handles wsy()>wey() and
4130 GOTO 4160
4140 IF (WSY(Q1)-(MAXY*ROUNDERR))>VY(Q1) THEN 4170   'wey()>wsy()
4150 IF (WEY(Q1)+(MAXY*ROUNDERR))<VY(Q1) THEN 4170
4160 VMAX=DIF:INTERSECT=Q1                  'sets new intersection point wall
4170 NEXT Q1
4180 PINTERSECT=INTERSECT     'PINTERSECT is the previous intersecting wall
4190 IF VMAX=MAXX THEN 4420
```

Continued on page 69

disk will be displayed. Note that each of the wall files has the .wal file extension. You must enter this extension to retrieve the file. This option is useful when you wish to display previously entered wall configurations on which to run the simulation.

3. *Run Simulation.* This option actually runs the vector file sweep. The first display after this option is run is shown in Figure 2. The wall file retrieved is drawn on the display, and the user is prompted to retrieve a vector file or create a vector file.

If the create a vector file option is selected, the user will be prompted to select the starting point for the vector file sweep. Next starting and ending angles are

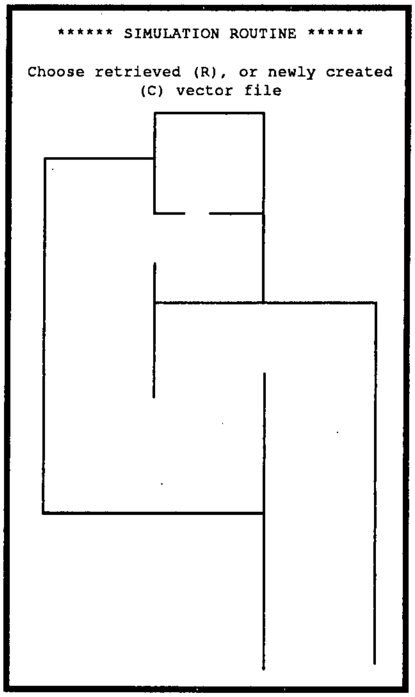

FIGURE 2. *The run simulation screen displays the selected wall configuration, then prompts the user for the vector file to be used. The example file shown is based on an actual waveguide corridor (see Reference 1). The example vector file selected starts at the midpoint of the entrance to the corridor, and sweeps from an angle of 25° to 155° in 100 discrete steps. The receive wall for this corridor is the horizontal wall at the top of the display.*

selected. The angles are defined as: 0° (straight right), 90° (straight up), and the maximum, 180° (straight left). If the starting and ending angles are identical you'll be prompted for the eight-digit file name. Otherwise the number of vectors needed to run the simulation (10-25 is good for beginners) must be provided.

If the retrieve-a-vector-file option is selected, all vector files which have been created and saved to the A disk will be displayed. Note that each of the vector files has the .vf file extension. You must enter this extension to retrieve the file. This option is useful when you wish to retrieve a specific radiation pattern.

Once the vector file is selected, the simulation begins to run. An example of the simulation display is shown in Figure 3. When a vector either hits the receive wall, or is reflected out of the corridor, the program stops momentarily, then proceeds to the next vector. The distance the vector traversed to that point is displayed.

The simulation runs until all vectors have run through the corridor. At the conclusion of the run, you may review the summarized data, or return to the main menu. If you elect to view the data (only the vectors which made it to the receive wall are listed), the option to

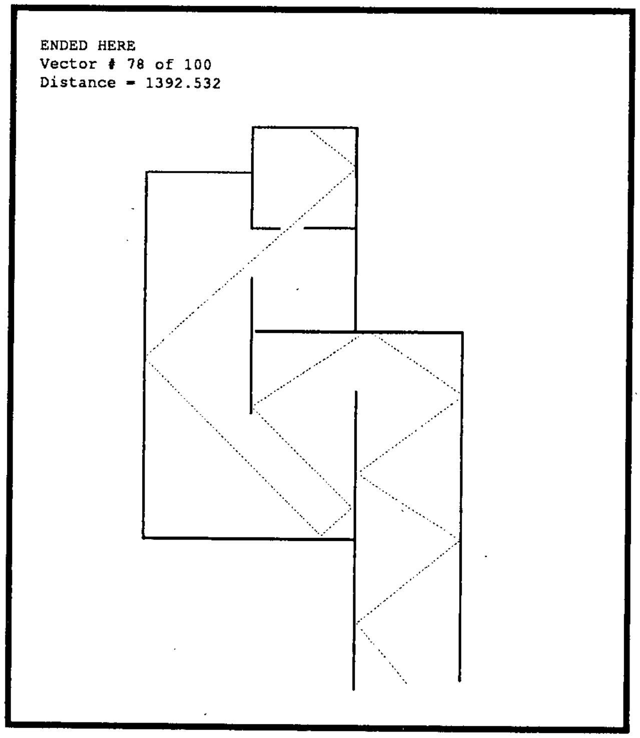

FIGURE 3. A typical vector as it proceeds from the input to the receive wall. The example vector shown (#78) is the shortest path (of the 100 attempted) to the receive wall. Figure 4 demonstrates that this is the case.

save the data is provided (See Table 1). Data files which are saved in this fashion have the .dat file extension.

4. *Help Menu.* Due to space constraints, the program provided does not provide a help menu. However, the space is provided for users to install their own help menu. Copies of the program are provided with a help menu in place.
5. *Exit.* Exit returns you to BASIC. Alternatively, compiled versions of the program will return you to DOS or the application program.

HOW IT WORKS

GENERAL

Some programmers may wish to add functions to this program, or rearrange it to suit a specific need. Although the program is annotated, an overview is presented here. Some key variables are also provided to assist the programmer in interpreting which parameters to change. For ease of interpretation, line numbers are provided in parenthesis after their reference.

Continued on page 78

Receive Location	Starting x	y	Vector #	Angle	Distance Traveled
14	537.50	100.00	3	27.63	8031
14	537.50	100.00	18	47.32	1530
14	537.50	100.00	24	55.20	1674
14	537.50	100.00	28	60.45	1581
14	537.50	100.00	38	73.59	1433
14	537.50	100.00	54	94.60	5593
14	537.50	100.00	78	126.11	1393
14	537.50	100.00	85	135.30	1599
14	537.50	100.00	96	149.75	2233

please hit "S" to save or return to proceed

***TABLE 1.** Received Signal Output Summary. This table catalogs all vectors which succeeded in making it to the receive wall (wall # 14). Each point has the same starting point, and the shortest distance to the receive wall is found to be vector # 78, which traverses the corridor as shown in Figure 3. Other vectors also have made their way to the receive wall, but vectors 3 and 54 will suffer significant space loss on their way.*

REFLECT.10 Program...Continued

```
4200 GOTO 4390
4210 VMAX = MAXX
4220 FOR Q1=1 TO IMAX
4230 DIF=(VX(Q1)+(MAXX*ROUNDERR/100))-VSX(J) ' includes roundoff compensation
4240 IF Q1=PINTERSECT THEN 4360
4250 IF DIF<=0 THEN 4360
4260 IF DIF>VMAX THEN 4360
4270 IF (WSX(Q1)-(MAXX*ROUNDERR))>VX(Q1) THEN 4360'checks for start/end point
4280 IF (WEX(Q1)+(MAXX*ROUNDERR))<VX(Q1) THEN 4360'on line (w/roundoff error)
orrection)
4290 IF WEY(Q1)>WSY(Q1) THEN 4330
4300 IF (WSY(Q1)+(MAXY*ROUNDERR))<VY(Q1) THEN 4360
4310 IF (WEY(Q1)-(MAXY*ROUNDERR))>VY(Q1) THEN 4360   'handles wsy()>wey() and
4320 GOTO 4350
4330 IF (WSY(Q1)-(MAXY*ROUNDERR))>VY(Q1) THEN 4360   'wey()>wsy()
4340 IF (WEY(Q1)+(MAXY*ROUNDERR))<VY(Q1) THEN 4360
4350 VMAX=DIF:INTERSECT=Q1
4360 NEXT Q1
4370 PINTERSECT=INTERSECT
4380 IF VMAX=MAXX THEN 4440
4390 LINE (VSX(J),VSY(J)) - (VX(INTERSECT),VY(INTERSECT))
4400 IF VX(INTERSECT)>0 OR VX(INTERSECT)<MAXX THEN 4450
4410 IF VY(INTERSECT)>0 OR VY(INTERSECT)<MAXY THEN 4450
4420 IF WR<>INTERSECT THEN 4440
4430 RDIST(BB)=DISTANCE+SQR((VX(INTERSECT)-VSX(J))^2+(VY(INTERSECT)-VSY(J))^2)
4435 LOCATE 3,1:PRINT "distance = ";RDIST(BB)
4440 LOCATE 1,1:PRINT "ENDED HERE":FOR DRE=1 TO DELA:NEXT DRE:GOTO 5290
4450 IF WR=INTERSECT THEN 4430
4455 LASTDIST=SQR((VX(INTERSECT)-VSX(J))^2+(VY(INTERSECT)-VSY(J))^2)
4460 DISTANCE=DISTANCE +LASTDIST
4462 IF LASTDIST>.001 THEN 4470
4463 DCOUNT=DCOUNT+1:IF DCOUNT<10 THEN 4470      'Protects against roundoff
4465 IF DCOUNT<5 THEN 4470                        'errors which slow run time
4466 LOCATE 10,15:PRINT"STUCK IN LOOP ERROR":FOR P=1 TO DELA:NEXT P
4468 GOTO 5290
4470 LOCATE 3,1:PRINT "distance = ";DISTANCE
4480 IF DISTANCE >MAXDISTANCE THEN 4440
4490 VSX(J+1)=VX(INTERSECT):VSY(J+1)=VY(INTERSECT)
4500 REM *** This portion of the program determines what the next line is ***
4510 WTHETA = ATN(WM(INTERSECT))                ' Converts intersected wall into
4520 IF WTHETA>0 THEN 4540                      ' angular data
4530 WTHETA = PI + WTHETA
4540 IF VTHETA<WTHETA THEN 4580
4550 THETA = VTHETA-WTHETA:GOSUB 5750
4560 PSI=THETA
4570 GOTO 4600
4580 THETA = WTHETA-VTHETA:GOSUB 5750
4590 PSI=THETA
4600 IF VM(J)<0 THEN 4880
4610 IF VS(J)=0 THEN 4740 ' 16 possible outcomes described in following eq.'s
4620 IF WM(INTERSECT)<0 THEN 4680                  ' Case 1
4630 IF VM(J)>WM(INTERSECT) THEN 4660
4640 THETA = WTHETA+PSI:GOSUB 5750
4650 GOTO 5180
4660 THETA = WTHETA-PSI:GOSUB 5750
4670 GOTO 5180
4680 IF (WTHETA-VTHETA)>(PI/2) THEN 4710          ' Case 2
4690 PSI = WTHETA - VTHETA:THETA = WTHETA+PSI
4700 GOTO 5180
4710 PSI = WTHETA - VTHETA:THETA = WTHETA+PSI
4730 GOTO 5180
4740 IF WM(INTERSECT)<0 THEN 4820                  ' Case 3
4750 IF (VTHETA-WTHETA)>PI THEN 4790
4760 PSI = ABS(WTHETA + PI - VTHETA)
4770 THETA = WTHETA + PI + PSI:GOTO 5180
4790 PSI = VTHETA - PI - WTHETA
4800 THETA = WTHETA + PI - PSI:GOTO 5180
4820 IF (VTHETA-WTHETA)<(PI/2) THEN 4860          ' Case 4
4830 PSI = WTHETA + PI - VTHETA
4840 THETA = WTHETA + PI + PSI:GOTO 5180
4860 THETA = WTHETA - PSI:GOSUB 5750
4870 GOTO 5180
4880 IF VS(J)=0 THEN 5050                          ' Case 5-6 or 7-8?  vm(j)<0
4890 IF WM(INTERSECT)<0 THEN 4970
4900 IF (VTHETA - ((3*PI)/2))>WTHETA THEN 4940   ' Case 5
4910 PSI = VTHETA - PI - WTHETA:
4920 THETA = WTHETA + PI - PSI:GOTO 5180
4940 PSI=(WTHETA+PI)-(VTHETA-PI)
4950 THETA = WTHETA + PSI:GOTO 5180
4970 THETA=VTHETA-WTHETA:GOSUB 5750
4980 IF (VTHETA - PI)>WTHETA THEN 5020            ' Case 6
4990 PSI = WTHETA - (PI + VTHETA)
5000 THETA = WTHETA + PI + PSI:GOTO 5180
5020 PSI = VTHETA - PI - WTHETA
5030 THETA = WTHETA + PI - PSI:GOTO 5180
5050 IF WM(INTERSECT)<0 THEN 5140                 ' Case 7 or 8?
5060 IF (VTHETA-(PI/2))>WTHETA THEN 5100          ' Case 7
5070 PSI = VTHETA-WTHETA:THETA = WTHETA - PSI
5080 GOSUB 5750
5090 GOTO 5180
5100 PSI = (WTHETA+PI)-VTHETA
5110 THETA = WTHETA + PI + PSI:GOSUB 5750
5130 GOTO 5180
5140 IF WTHETA<VTHETA THEN 5170    ' Case 8
```

Continued on page 78

REFLECT.10 Program...Continued from page 69

```
5150 THETA = WTHETA + PSI:GOTO 5180
5170 THETA = WTHETA - PSI                    ' Concludes 16 combos
5180 NVTHETA = THETA             ' **** BEGIN STORAGE AND NEW VECTOR
5190 IF NVTHETA<PI THEN 5210
5200 THETA = THETA-PI
5210 VM(J+1)=TAN(THETA)                      'sets next vector variables
5220 VB(J+1)=VSY(J+1)-VM(J+1)*VSX(J+1)
5230 VS(J+1)=1
5240 IF NVTHETA<(PI/2) THEN 5270
5250 IF NVTHETA>((3*PI)/2) THEN 5270
5260 VS(J+1)=0                               'sets next vector sense
5270 J=J+1
5280 GOTO 3670
5290 TOTDIST=TOTDIST+DISTANCE                ' increases total distance
5300 IF TYPEVEC=0 THEN 5330
5310 PINTERSECT=999:DUM=0:DISTANCE=0
5320 NEXT BB
5330 LOCATE 1,1:PRINT "                                                 "
5340 LOCATE 1,23:PRINT "Simulation ended with total distance = ";TOTDIST
5350 LOCATE 2,23:PRINT " please hit Return to show received data, or"
5360 LOCATE 3,23:INPUT " 'C' to continue ",IMPUT$
5370 IF IMPUT$="C" OR IMPUT$="c" THEN 5610
5380 XR=1    ' xr is the received wall designator
5390 CLS:COUNTS=1:LOCATE 2,20:PRINT "Received Signal Output Summary"
5400 LOCATE 3,10:PRINT "Receive     Starting           Vector       Distance"
5410 LOCATE 4,10:PRINT "location    x         y        #      angle traveled"
5420 FOR OO=1 TO NV
5430 IF RDIST(OO)=0 THEN 5580
5440 COUNTS=COUNTS+1
5450 IF INT((COUNTS+5)/22)<>((COUNTS+5)/22) THEN 5500
5460 LOCATE 23,10:INPUT "Please hit return to continue ",YT$
5470 CLS:COUNTS=1
5480 LOCATE 3,10:PRINT "Receive     Starting           Vector       Distance"
5490 LOCATE 4,10:PRINT "location    x         y        #      angle traveled"
5500 LOCATE (COUNTS+5),10
5510 IF NV<>1 THEN 5530
5520 ANGL=SA:GOTO 5540
5530 ANGL=(EA-SA)*(OO-1)/(NV-1)+SA
5540 PRINT USING "  ###";WR;
5550 PRINT USING "     ###.##";VSX(1),VSY(1);
5560 PRINT USING "     ####";OO;
5570 PRINT USING "      ###.##";ANGL;:PRINT USING "    #####";RDIST(OO)
5580 NEXT OO
5590 LOCATE 24,10
5600 INPUT "please hit 'S' to save or return to proceed ... ",OOG$
5610 IF OOG$="S" OR OOG$="s" THEN 5630
5620 GOTO 690
5630 CLS:LOCATE 5,15:PRINT "DATA SAVE ROUTINE"
5640 LOCATE 10,10:INPUT "Enter the eight digit filename for save ... ",GNAM$
5645 IF LEN(GNAM$)>8 THEN 5640
5650 OPEN "a:"+GNAM$+".dat" FOR OUTPUT AS #1
5660 FOR CC=1 TO NV
5670 IF RDIST(CC)=0 THEN 5720
5680 IF NV<>1 THEN 5700
5690 ANGL=SA:GOTO 5710
5700 ANGL=(EA-SA)*(CC-1)/(NV-1)+SA
5710 WRITE #1, CC,VSX1,VSY1,ANGL,RDIST(CC)
5720 NEXT CC
5730 CLOSE #1
5740 GOTO 690
5750 REM  **** Subroutine for modulo calculation (unit circle) ****
5760 REM  ****   Uses variable theta and returns theta        ****
5770 INCOME=THETA
5780 IF THETA < 2*PI THEN 5800
5790 THETA = THETA-2*PI
5800 IF THETA > 0 THEN 5820
5810 THETA = THETA + 2*PI
5820 RETURN
6000 '******    HELP MENU SUBROUTINE    ********
6010 GOTO 690
7000 KEY ON:END
```

THE FLOWCHART

As the flowchart (Figure 4) illustrates, the program starts with an initialization section. Next, the PC screen is set for a 1000 x 1000 graphics grid for plotting, and the main menu is displayed (700). The user selects an option (760), which must first define the wall file.

The wall files are created and retrieved in lines 1000-1340 and 2000-2110, respectively. The wall files are stored with the starting x less than the ending x for all walls, and WR (1210) stores which wall number is the receive wall. IMAX (1250) is the number of walls stored.

After the wall file is provided, the vector file is defined. If a vector file is retrieved, the file format provides the slope, intercept point, and sense (1 for x increasing, 0 for x decreasing), along with 3 user defined dummy variables (dum, 3175).

All intersections and reflections used in this program use the slope delta format (lines defined as y = mx + b) to determine where lines intersect. Therefore, angular data is converted to slope delta format prior to the processing of all data. Once the slope and delta are known, the equations are solved for x, and the value of the y intercepts is determined directly.

Error traps must be provided for walls or vectors which are straight up/down or left/right, to eliminate slopes of zero or infinity. Furthermore, their intersections with other lines can cause round-off errors since the intercept with the y axis (b) may be quite large for lines straight up and down, for example. If round-off errors are repetitive, the vector file may become trapped in a tiny square caused by bad approximations. Lines 4040, 4240, and 4466 deal with this problem.

Once the closest intersecting line is known, the reflection from that wall is determined. Eight possible outcomes are possible with a known vector's sense/slope, and the wall's slope (4600-5160). With the reflection angle known, the next vector's parameters are described (5210-5260).

The next lines handle the received signal output summary, and its associated storage (5330-5740). The modulo subroutine (5750-5820) limits the values that theta can take from 0° to 360°. If the values are greater than 360° (2*pi radians), 360° (or an integral number of 360) are substracted from the angle value. Similarly, integral numbers of 360° can be added to input thetas with negative angles to make them positive. Lines from 6000 on can be used for a Help Menu, and line 7000 ends the program.

The following variables are helpful in modifying the program for a specific user application.

maxx and maxy
These variables define the boundaries of the workspace. Their values are defaulted to 1000, and any input units should be scaled to take maximum advantage of this space.

rounderr
Denotes amount of round-off error which can be compensated.

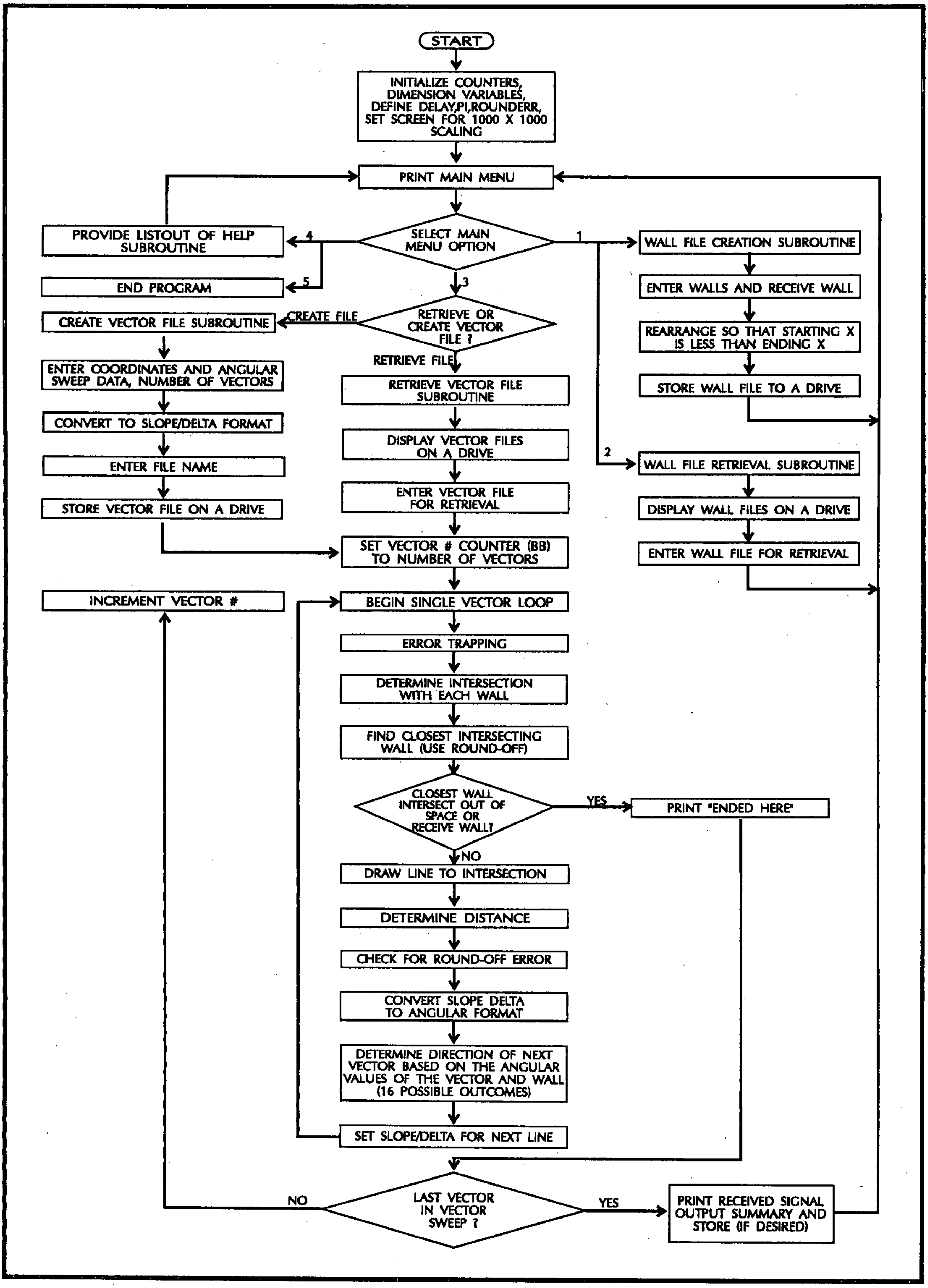

FIGURE 4. *Flow Chart shows the general flow of the REFLECT.10 program. Note that nearly all of the code is contained in the run simulation (#3) portion of the program.*

This variable limits the accuracy of measurement, but prevents routine calculation snags from stopping operation of the program.

Maxdistance

This variable limits the length of the simulation by ending the simulation whenever the distance of the path propagation exceeds this value. It is useful because it stops program execution if the input vector has no energy left. The default value is 100,000 distance units, but laser and other coherent sources (in a reflective environment) will require a much higher value.

bb and j

Counters for the vector # and reflection #, respectively.

dela

Determines the delay associated with the speed of the computer. Default is 3000 (for an 8088 machine). Longer delays may be needed for faster machines.

TIPS ON USE

This section offers suggestions to make using REFLECT.10 easier. Some general precautions are suggested, followed by some advice for the novice.

DO NOT

- Overlap wall lines. Overlapping causes the program to misinterpret which wall it is intersecting and can provide false results. If you make an entry error during the wall entry, finish the entry process, stop the program, enter a line editor program (EDLIN is provided with most DOS machines), and change the offending line. See your DOS manual for more information about EDLIN.
- Be disheartened if the program does not work when you first enter it. If you have difficulty debugging BASIC programs, find a friend who can help, or learn how to debug the program yourself.

DO

- Modify the input vector size for the system you're using. Some newer systems avoid the 64 kB constraints on the amount of memory BASIC data can use. The program provided shows all the dimensioned data for 1000 possible input vectors (per vector file). If you have problems with an older machine or BASIC version, reduce the dimensioned variables (DIM statements) down to 100 for all lines 560-580.
- Use the same starting point for all starting transmit vector inputs (500,100 is recommended). This will allow you to change the walls for a vector input, or the vector input for a specific wall interchangeably.
- Use a receive wall instead of a narrow point (to simulate a receive antenna size, for example). Remember that the RF leakage through the corridor is what's being measured.
- Remember basic waveguide theory. Equations are available which describe the attenuation (at lower frequencies) provided by the long, thin corridor. The smaller the dimension of this corridor, the less low frequency shielding/absorption is required.
- Use the control-break keys to stop program execution at any point.
- Ignore screen distortion. Although the monitor you have will provide an elongated (in width) version of your design, the math will provide the correct answers.
- Use multiple transmit antenna locations if your initial location is successful. No design is final until it shields emissions from every direction.
- Use graph paper to scale your wall pattern. Wall entry is difficult to perform without some reference with which to make measurements. Graph paper will allow you the flexibility to create your wall pattern and the

Continued on page 333