

Computation and Visualization of Groundwater Flow

Seven Classic Cases Using Finite Difference Methods

Dr. Charles J. Paradis
Department of Geosciences
University of Wisconsin at Milwaukee
2023

DRAFT MANUSCRIPT

1. Introduction

Herbert F. Wang and Mary P. Anderson wrote the seminal textbook, *Introduction to Groundwater Modeling: Finite Difference and Finite Element Methods* in 1982. The finite difference methods in their textbook, written over 40 years ago, remain highly relevant today, as evident by their use in MODFLOW 6, the United States Geological Survey's latest version of their finite difference groundwater flow model. John W. Backus developed, and IBM released, one of the first widely used computer programming languages, Fortran, in 1957. Fortran, now approaching 70 years old, also remains highly relevant today, as evident by its use to program every version of MODFLOW since its inception in 1983. Like MODFLOW, Fortran continues to develop new capabilities to adapt to the demands of scientific and engineering computing; Fortan 2023 is its 12th version, and MODFLOW 6 was released in 2017 and updated (MODFLOW 6.4.2) in 2023.

The finite difference method using Fortran has proven to be one of the most reliable methodological approaches in all of computational hydrogeology. However, Fortran does not directly support visualization of its output, e.g., hydraulic heads versus space and/or time. Gnuplot is a graphing utility that supports visualization of computational outputs via two- and three-dimensional plots. Gnuplot was first released in 1986, and like Fortran, continues to develop new capabilities to adapt to the demands of scientific and engineering visualization; Gnuplot 5 was released in 2015 and updated (Gnuplot 5.4) in 2023.

Computation of groundwater flow can be performed accurately, rapidly, and freely using an open-source compiler, such as GFortran, to compile programs written in Fortran that can be executed from the command line on various operating systems. Visualization of groundwater flow can be similarly performed using an open-source graphing utility, such as Gnuplot, to generate two- and three-dimensional plots of hydraulic heads versus space and/or time; programs written in Gnuplot can also be executed from the command line on various operating systems.

What follows is a demonstration of the finite difference method for computation and visualization of groundwater flow based on seven classic cases from Wang and Anderson (1982). The computational programs are written in Fortran, like Wang and Anderson (1982), but have been updated to exclude the use of obsolescent or deleted features and to include the use of modern programming practices. For example, the programs are written explicitly, as opposed to implicitly, thus avoiding any potentially incorrect compiler specific assumptions regarding the types of declared variables. The program loops are written with a single entry point and a single exit point, thus avoiding the use of go to statements that can potentially lead to confusion and errors. The Fortran programs are also written to include outputs for the visualization utility, Gnuplot, to use as inputs to generate plots of hydraulic head; the visualization programs are written in Gnuplot's scripting language.

The reader is assumed to be versed in the fundamentals of differential calculus and hydrogeology but need not be versed in the finite difference method nor computer programming. Therefore, the finite difference method and computer programming are given sufficient explanation, by theory and application, to allow the reader to begin developing a skill set in computational hydrogeology.

2. Groundwater Flow

The three-dimensional, heterogeneous, anisotropic, transient groundwater flow equation can be written as:

$$\frac{\partial}{\partial x} \left(K_x \frac{\partial h}{\partial x} \right) + \frac{\partial}{\partial y} \left(K_y \frac{\partial h}{\partial y} \right) + \frac{\partial}{\partial z} \left(K_z \frac{\partial h}{\partial z} \right) = S_s \frac{\partial h}{\partial t} - W \quad (1)$$

where: h =hydraulic head [L], x , y , z =space [L], t =time [T], K =hydraulic conductivity [L/T], S_s =specific storage 1/[L], and W =source/sink of water 1/[T], note that subscripts x , y , and z on K indicate direction

Equation (1) can be simplified for steady flow with no source/sink of water as:

$$\frac{\partial}{\partial x} \left(K_x \frac{\partial h}{\partial x} \right) + \frac{\partial}{\partial y} \left(K_y \frac{\partial h}{\partial y} \right) + \frac{\partial}{\partial z} \left(K_z \frac{\partial h}{\partial z} \right) = 0 \quad (2)$$

Equation (2) can be further simplified for homogeneous flow as:

$$K_x \frac{\partial^2 h}{\partial x^2} + K_y \frac{\partial^2 h}{\partial y^2} + K_z \frac{\partial^2 h}{\partial z^2} = 0 \quad (3)$$

Finally, Equation (3) can be simplified for isotropic flow in two-dimensions as:

$$\frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2} = 0 \quad (4)$$

Equation (4) describes two-dimensional, homogeneous, isotropic, steady flow, is a form of the Laplace equation, and is the starting point for computation and visualization of groundwater flow.

3. Case One: Two-dimensional Steady Flow Near a Well

Equation (4) can describe two-dimensional steady flow near a well using the finite difference method and then solved computationally under specified boundary conditions. The spatial derivatives in Equation (4) can be approximated by the finite difference method as:

$$\frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2} \approx \frac{\frac{h_{i,j+1} - h_{i,j}}{\Delta x} - \frac{h_{i,j} - h_{i,j-1}}{\Delta x}}{\Delta x} + \frac{\frac{h_{i+1,j} - h_{i,j}}{\Delta y} - \frac{h_{i,j} - h_{i-1,j}}{\Delta y}}{\Delta y} \quad (5)$$

where: subscript i=row index, subscript j=column index, Δx , Δy =grid space [L]

Equation (5) can be simplified as:

$$\frac{h_{i,j+1} - 2h_{i,j} + h_{i,j-1}}{(\Delta x)^2} + \frac{h_{i+1,j} - 2h_{i,j} + h_{i-1,j}}{(\Delta y)^2} = 0 \quad (6)$$

If Δx is equal to Δy , Equation (6) can be simplified and solved for the hydraulic head at the center node as:

$$\frac{h_{i,j+1} + h_{i,j-1} + h_{i+1,j} + h_{i-1,j}}{4} = h_{i,j} \quad (7)$$

Equation (7) states that the hydraulic head at the center node is equal to the arithmetic mean of its four surrounding nodes. A unique solution to Equation (4) can be obtained by first specifying boundary conditions in the form of specified heads (Dirichlet), specified flow (Neumann), or some combination of the two (mixed). For this case, the heads are specified on all four sides of a finite difference model that is four rows by four columns (Figure 1). The four nodes inside of the boundary conditions (Figure 1) can be solved using Equation (7) by Gauss-Seidel iteration as:

$$\frac{h_{i,j+1}^m + h_{i,j-1}^m + h_{i+1,j}^m + h_{i-1,j}^m}{4} = h_{i,j}^{m+1} \quad (8)$$

where: superscript m=iteration index

For this case, an initial guess ($m=0$) at the heads of the four inner nodes is made (Figure 1) and then Equation (8) placed at the upper and left-most inner node to compute $h_{2,2}$, $h_{2,3}$, $h_{3,2}$, and $h_{3,3}$ while using the newly computed heads ($m+1$) to the left and/or above the center node (Figure 1). Equation (8) can be used iteratively to sweep through the inner nodes, from top to bottom and left to right, until the changes in head from one iteration to the next are negligible and the finite difference solution has converged.

Figure 1 Four by four finite difference model with specified head boundary conditions on all four sides, four inner nodes can be solved by an initial guess ($m=0$) followed by Gauss-Seidel iteration, top model shows initial guess, bottom model show results after one iteration ($m+1$)

heads:

8.04	8.18	8.36	8.53
7.68	8.00	8.50	8.41
7.19	7.00	8.00	8.33
6.82	7.56	7.99	8.29

number of iterations = 0

heads:

8.04	8.18	8.36	8.53
7.68	7.84	8.15	8.41
7.19	7.65	8.03	8.33
6.82	7.56	7.99	8.29

number of iterations = 1

The finite difference method using Gauss-Seidel iteration can be performed using a computational program written in Fortran (Figure 2). The program contains four major sections: declare, define, compute, and output. The declare section (lines 6 through 10) explicitly states the types of variables, e.g., integer, real, or parameter; a parameter type is a constant value and must be defined immediately. The define section (lines 11 through 26) assigns numeric values to the declared variables; these are not constant values and can be re-assigned during computation. The computation section (lines 27 through 43) sweeps through the inner nodes, using Equation (8) (lines 35 and 36), to compute the heads, the largest change in heads between iterations (lines 37 through 40), and determine if either the computation has converged or if the maximum iterations has been reached (line 29). The output section (lines 44 through 55) is divided into two sub-sections. Sub-section one (lines 45 through 50) outputs the heads and the number of iterations to the command line (Figure 3). Sub-section two (lines 51 through 55) outputs the heads to a text file (Figure 4) that can be used as in input file for visualization by Gnuplot.

Figure 2 Computation Fortran program for Case One

```

1  program code_ex01
2 ! Similar to Figure 2.7 in Wang & Anderson (1982)
3 ! Computer program with Gauss-Seidel iteration for the region-near-a-well.
4 ! Code written by C.J. Paradis (paradisc@uwm.edu)
5 implicit none
6 ! DECLARE
7 integer :: i, j, numit, maxit ! Indices & iteration counters/limits
8 real :: converg, error, maxerr, oldval ! Converge, errors, & old values
9 integer,parameter :: row=4, col=4 ! Set constant grid size
10 real,dimension(row,col) :: heads ! Heads array
11 ! DEFINE
12 heads(1,1:4)=(/8.04,8.18,8.36,8.53/) ! Specified head top
13 heads(4,1:4)=(/6.82,7.56,7.99,8.29/) ! Specified head bottom
14 heads(2:3,1)=(/7.68,7.19/) ! Specified head left
15 heads(2:3,4)=(/8.41,8.33/) ! Specified head right

```

```

16 ! Inner head initial guesses
17 heads(2,2)=(8.)
18 heads(3,3)=(8.)
19 heads(2,3)=(8.5)
20 heads(3,2)=(7.)
21 numit=0 ! Set iteration counter
22 maxit=10 ! Set maximum iterations
23 converg=0.01 ! Set convergence criteria
24 error=0. ! Initialize error
25 maxerr=2.*converg ! Enable initial entry to computation loop
26 oldval=0. ! Initialize old value
27 ! COMPUTE
28 do
29 if(maxerr<=converg .or. numit>=maxit) exit
30 numit=numit+1
31 maxerr=0.
32   do i=2,row-1
33     do j=2,col-1
34       oldval=heads(i,j)
35       heads(i,j)=(heads(i-1,j)+heads(i+1,j) &
36                   +heads(i,j-1)+heads(i,j+1))/4.
37       error=abs(heads(i,j)-oldval)
38       if(error>maxerr) then
39         maxerr=error
40       end if
41     end do
42   end do
43 end do
44 ! OUTPUT
45 ! Write to command line
46 write(*,*) "heads:"
47 write(*,10) ((heads(i,j),j=1,col),i=1,row)
48 10 format(1x,4f8.2)
49 write(*,11) numit
50 11 format(" number of iterations = ", i3)
51 ! Write to text file
52 open(unit=100, file="head_ex01.txt")
53 write(100,12) ((heads(i,j),j=1,col),i=1,row)
54 12 format(1x,4f8.2)
55 close(100)
56 end program code_ex01

```

Figure 3 Command line output from the Fortran program (lines 45 through 50 in Figure 2)

heads:

8.04	8.18	8.36	8.53
7.68	7.93	8.19	8.41
7.19	7.68	8.05	8.33
6.82	7.56	7.99	8.29

number of iterations = 4

Figure 4 Text file output from the Fortran program (lines 51 through 55 in Figure 2)

```
8.04 8.18 8.36 8.53
7.68 7.93 8.19 8.41
7.19 7.68 8.05 8.33
6.82 7.56 7.99 8.29
```

The computational program written in Fortran (Figure 2) is annotated to provide a more detailed explanation of how it is constructed and how it works. The first-time reader of a Fortran program will likely benefit greatly from repeatedly reading the program carefully, followed by copying, pasting, compiling, and executing the program from the command line on their own computer, and finally changing the program slightly to better understand how its constructed and how it works, e.g., change the boundary conditions, change the initial guesses at the inner nodes, change the convergence criterion, change the maximum number of iterations.

The text file output of the hydraulic heads in space from the Fortran program (Figure 4) can be plotted and contoured using a visualization program written in Gnuplot's scripting language (Figure 5). The program begins by setting the terminal type and size (line 3) and the name and file type of the output (line 4). The specifics of the plot's font, title, axes labels and ranges, style, and key placement are then set (lines 5 through 12). The plot is then set to display contours across a range and interval, an interpolation method is chosen, and the display of data used for interpolation is turned off (lines 13 through 16). A table to temporarily store the contour data is set and a three-dimensional surface plot of the contours, based on the head values from the Fortran output (Figure 4), is created but not displayed (lines 17 through 19). Finally, a two-dimensional plot of the contour lines with labels is created (line 20 through 21).

Figure 5 Visualization Gnuplot program for Case One

```
1 #Plot heads with contours
2 reset
3 set terminal pngcairo size 800,800
4 set output "cont_ex01.png"
5 set font "Times New Roman, 12"
6 set title "Case 01: 2D Steady Flow Near Well"
7 set xlabel "Distance along X"
8 set ylabel "Distance along Y"
9 set xrange [0:3]
10 set yrange [3:0]
11 set style line 1 linecolor rgb "blue" linewidth 2
12 set key below
13 set contour
14 set cntrparam levels incremental 6.5, 0.25, 8.5
15 set cntrparam bspline
16 unset surface
17 set table $contour
18 splot "head_ex01.txt" matrix
19 unset table
20 plot $contour linestyle 1 title "Iso-heads" with lines,\n    $contour every 6 title "" with labels font "Times New Roman, 8"
```

The plot of contoured heads in space (Figure 6) can greatly enhance the visualization of the text file output from the Fortran program (Figure 4). For this case, it is clear that the direction of groundwater flow is towards the left-hand corner of the finite difference model (Figure 6). The importance of coupling visualization with computation will become clearer as more complex cases of groundwater flow are considered.

Case 01: 2D Steady Flow Near Well

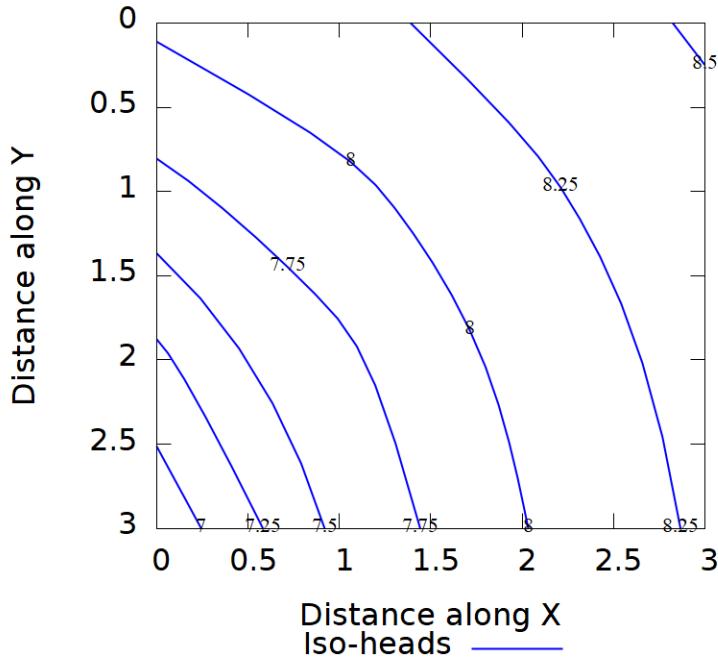


Figure 6 Image file output of the Gnuplot program (Figure 5)

4. Case Two: Two-dimensional Steady Regional Flow

For this case, only the computation and visualization programs and their outputs are included. For cases 3 through 7, only the computation programs are included. This is a draft manuscript and its completion is pending additional time and resources.

Figure 7 Computation Fortran program for Case Two

```
1  program code_ex02
2 ! Similar to Figure 2.10 in Wang & Anderson (1982)
3 ! Computer program for regional flow example using Gauss-Seidel iteration.
4 ! Code written by C.J. Paradis (paradisc@uwm.edu)
5 implicit none
6 ! DECLARE
7 integer :: i, j, numit, maxit ! Indices & iteration counters/limits
8 real :: converg, error, maxerr, oldval ! Converge, errors, & old vals
9 real :: dx ! grid spacing
10 integer,parameter :: row=7, col=13 ! Set constant grid size
11 real,dimension(row,col) :: heads ! Heads array
12 ! DEFINE
13 ! Fill array with placeholders
14 do i=1,row
15     do j=1,col
16         heads(i,j)=100.
17     end do
18 end do
19 numit=0 ! Set iteration counter
20 maxit=200 ! Set maximum iterations
21 converg=0.001 ! Set convergence criteria
22 error=0. ! Initialize error
23 maxerr=2.*converg ! Enable initial entry to computation loop
24 oldval=0. ! Initialize old value
25 dx=20. ! Grid spacing
26 do j=2,col-1
27     heads(1,j)=0.02*dx*(j-2)+100. ! Top specified-head boundary
28 end do
29 ! COMPUTE
30 do
31 if(maxerr<=converg .or. numit>=maxit) exit
32 numit=numit+1
33 maxerr=0.
34     do i=1,row
35         heads(i,1)=heads(i,3) ! Left specified-flux boundary
36         heads(i,col)=heads(i,col-2) ! Right specified-flux boundary
37     end do
38     do j=1,col
39         heads(row,j)=heads(row-2,j) ! Bottom specified-flux boundary
40     end do
41     do i=2,row-1
42         do j=2,col-1
43             oldval=heads(i,j)
44             heads(i,j)=(heads(i-1,j)+heads(i+1,j) &
45             +heads(i,j-1)+heads(i,j+1))/4.
46             error=abs(heads(i,j)-oldval)
47             if(error>maxerr) then
```

```

48                      maxerr=error
49      end if
50      end do
51  end do
52 end do
53 ! OUTPUT
54 ! Write to command line
55 write(*,*) "heads:"
56 write(*,10) ((heads(i,j),j=1,col),i=1,row)
57 10 format(1x,13f8.2)
58 write(*,11) numit
59 11 format(" number of iterations = ", i3)
60 ! Write to text file
61 open(unit=100, file="head_ex02.txt")
62 write(100,12) ((heads(i,j),j=2,col-1),i=1,row-1)
63 12 format(1x,11f8.2)
64 close(100)
65 end program code_ex02

```

Figure 8 Command line output from the Fortran program (lines 54 through 59 in Figure 7)

heads:

```

100.40 100.00 100.40 100.80 101.20 101.60 102.00 102.40 102.80 103.20 103.60 104.00 103.60
100.78 100.63 100.78 101.03 101.34 101.66 101.99 102.33 102.65 102.95 103.21 103.35 103.21
101.05 100.98 101.05 101.22 101.45 101.71 101.99 102.26 102.53 102.76 102.93 103.00 102.93
101.22 101.17 101.22 101.35 101.53 101.75 101.98 102.22 102.44 102.62 102.74 102.79 102.74
101.32 101.28 101.32 101.42 101.58 101.77 101.98 102.19 102.38 102.54 102.64 102.68 102.64
101.35 101.31 101.35 101.45 101.60 101.78 101.98 102.18 102.36 102.51 102.61 102.64 102.61
101.32 101.28 101.32 101.42 101.58 101.77 101.98 102.19 102.38 102.54 102.64 102.68 102.64
number of iterations = 109

```

Figure 9 Visualization Gnuplot program for Case Two

```

1 #Plot heads with contours
2 reset
3 set terminal pngcairo size 1600,800
4 set output "cont_ex02.png"
5 set font "Times New Roman, 12"
6 set title "Case 02: 2D Steady Regional Flow"
7 set xlabel "Distance along X"
8 set ylabel "Distance along Y"
9 set xrange [0:10]
10 set yrange [5:0]
11 set style line 1 linecolor rgb "blue" linewidth 2
12 set key below
13 set contour
14 set cntrparam levels incremental 100, 0.25, 104
15 set cntrparam bspline
16 unset surface
17 set table $contour
18 splot "head_ex02.txt" matrix
19 unset table
20 plot $contour linestyle 1 title "Iso-heads" with lines,
21     $contour every 6 title "" with labels font "Times New Roman, 8"

```

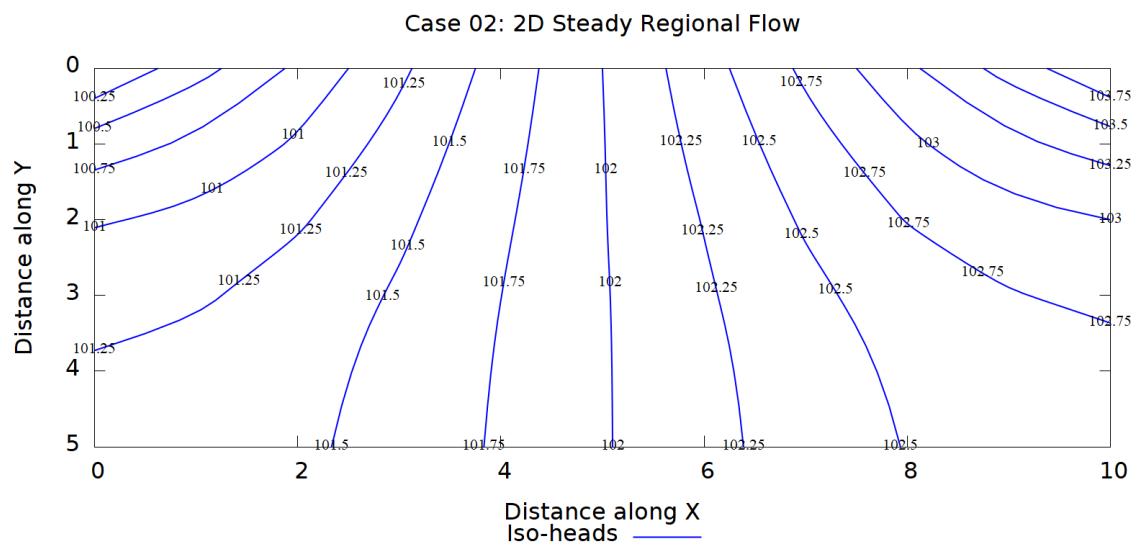


Figure 10 Image file output of the Gnuplot program (Figure 9)

5. Case Three: Two-dimensional Steady Flow to a Well

Figure 11 Computation Fortran program for Case Three

```

1  program code_ex03
2 ! Similar to Figure 3.5 in Wang & Anderson (1982)
3 ! Finite difference program for solving the drawdown example.
4 ! Code written by C.J. Paradis (paradisc@uwm.edu)
5 implicit none
6 ! DECLARE
7 integer :: i, j, numit, maxit ! Indices & counters
8 real :: converg, error, maxerr, oldval ! Converge, errors, & old vals
9 real :: dx, omega ! Grid spacing & relaxation
10 real :: Q, T, R ! Aquifer parameters and recharge
11 integer,parameter :: row=12, col=12 ! Set constant grid size
12 real,dimension(row,col) :: heads, Rfield ! Heads and recharge arrays
13 real,dimension(row,col) :: radfield, conheads ! Radial & constant heads
14 ! DEFINE
15 ! Fill arrays with placeholders
16 do i=1,row
17   do j=1,col
18     heads(i,j)=10.
19     Rfield(i,j)=0.
20     radfield(i,j)=0.
21     conheads(i,j)=0.
22   end do
23 end do
24 numit=0 ! Set iteration counter
25 maxit=100 ! Set maximum iterations
26 converg=0.001 ! Set convergence criteria
27 error=0. ! Initialize error
28 maxerr=2.*converg ! Enable initial entry to computation loop
29 oldval=0. ! Initialize old value
30 dx=200. ! Grid spacing
31 omega=1.8 ! Relaxation
32 Q=-2000. ! Pumping rate
33 T=300. ! Transmissivity
34 R=Q/(dx*dx) ! Recharge rate
35 ! Specified head boundary at radial distance from well
36 do i=row-1,1,-1
37   do j=2,col
38     radfield(i,j)=(((row-1-i)*dx)**2.+((2-j)*dx)**2.)**(.1./2.)
39   end do
40 end do
41 do i=1,row-1
42   do j=2,col
43     if(radfield(i,j)>=1950.) then
44       conheads(i,j)=10. ! Constant head boundary at approx. 2000 m
45     end if
46   end do
47 end do
48 Rfield(row-1,2)=R ! Pumping well location
49 ! COMPUTE
50 do
51   if(maxerr<=converg .or. numit>=maxit) exit
52   numit=numit+1

```

```

53 maxerr=0.
54   do j=1,col
55     heads(col,j)=heads(col-2,j) ! Bottom specified-flux boundary
56   end do
57   do i=1,row
58     heads(i,1)=heads(i,3) ! Left specified-flux boundary
59   end do
60   do i=2,row-1
61     do j=2,col
62       if(conheads(i,j)==10.) then
63         heads(i,j)=conheads(i,j)
64       else
65         oldval=heads(i,j)
66         heads(i,j)=(heads(i-1,j)+heads(i+1,j) &
67           +heads(i,j-1)+heads(i,j+1)+dx*dx*Rfield(i,j)/T)/4.
68         heads(i,j)=omega*heads(i,j)+(1.-omega)*oldval
69         error=abs(heads(i,j)-oldval)
70         if(error>maxerr) then
71           maxerr=error
72         end if
73       end if
74     end do
75   end do
76 end do
77 ! OUTPUT
78 ! Write to command line
79 write(*,*) "heads:"
80 write(*,10) ((heads(i,j),j=1,col),i=1,row)
81 10 format(1x,12f8.2)
82 write(*,11) numit
83 11 format(" number of iterations = ", i3)
84 ! Write to text file
85 open(unit=100, file="head_ex03.txt")
86 write(100,12) ((heads(i,j),j=2,col),i=1,row-1)
87 12 format(1x,11f8.2)
88 close(100)
89 end program code_ex03

```

Figure 12 Command line output from the Fortran program (lines 78 through 83 in Figure 11)

heads:

10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00
9.89	9.88	9.89	9.90	9.93	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00
9.76	9.76	9.76	9.79	9.83	9.88	9.93	10.00	10.00	10.00	10.00	10.00	10.00
9.62	9.61	9.62	9.66	9.71	9.77	9.84	9.92	10.00	10.00	10.00	10.00	10.00
9.47	9.45	9.47	9.51	9.57	9.65	9.74	9.83	9.92	10.00	10.00	10.00	10.00
9.28	9.26	9.28	9.34	9.42	9.53	9.63	9.74	9.84	9.93	10.00	10.00	10.00
9.05	9.02	9.05	9.14	9.26	9.39	9.53	9.65	9.77	9.88	10.00	10.00	10.00
8.77	8.71	8.77	8.92	9.09	9.26	9.42	9.57	9.71	9.83	9.93	10.00	
8.41	8.26	8.41	8.67	8.92	9.14	9.34	9.51	9.66	9.79	9.90	10.00	
7.96	7.50	7.96	8.41	8.77	9.05	9.28	9.47	9.63	9.76	9.89	10.00	
7.50	5.84	7.50	8.26	8.71	9.02	9.26	9.45	9.61	9.76	9.88	10.00	
7.96	7.50	7.96	8.41	8.77	9.05	9.28	9.46	9.62	9.76	9.89	10.00	

number of iterations = 51

Figure 13 Visualization Gnuplot program for Case Three

```
1 #Plot heads with contours
2 reset
3 set terminal pngcairo size 800,800
4 set output "cont_ex01.png"
5 set font "Times New Roman, 12"
6 set title "Case 01: 2D Steady Flow Near Well"
7 set xlabel "Distance along X"
8 set ylabel "Distance along Y"
9 set xrange [0:3]
10 set yrange [3:0]
11 set style line 1 linecolor rgb "blue" linewidth 2
12 set key below
13 set contour
14 set cntrparam levels incremental 6.5, 0.25, 8.5
15 set cntrparam bspline
16 unset surface
17 set table $contour
18 splot "head_ex01.txt" matrix
19 unset table
20 plot $contour linestyle 1 title "Iso-heads" with lines,
21     $contour every 6 title "" with labels font "Times New Roman, 8"
```

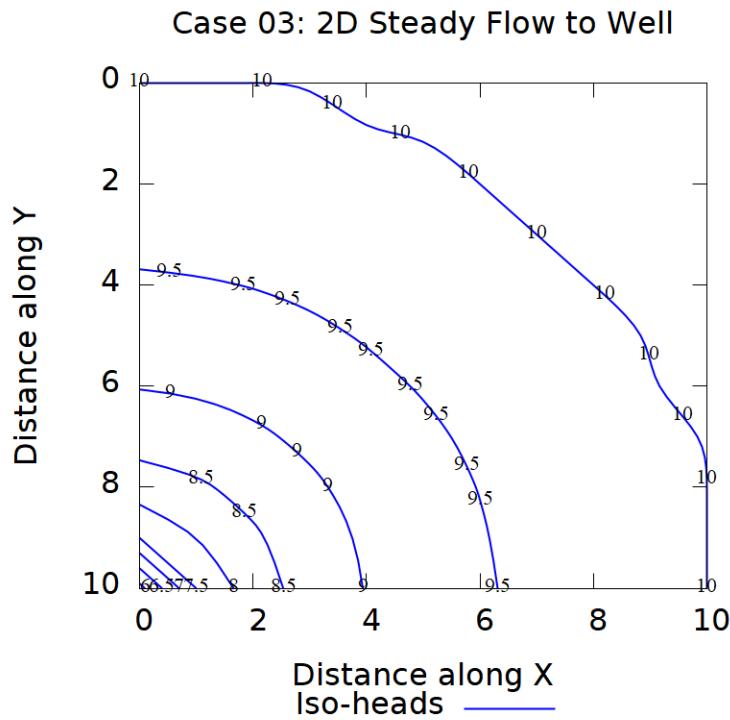


Figure 14 Image file output of the Gnuplot program (Figure 13)

6. Case Four: Two-dimensional Steady Flow Through a Dam

Figure 15 Computation Fortran program for Case Four

```
1 program code_ex04
2 ! Similar to Figure 3.10 in Wang & Anderson (1982)
3 ! Computer program for solving 1D seepage thru dam, Dupuit assumptions.
4 ! Code written by C.J. Paradis (paradisc@uwm.edu)
5 implicit none
6 ! DECLARE
7 integer :: i, j, numit, maxit ! Indices & counters
8 real :: converg, error, maxerr, oldval ! Converge, errors, & old vals
9 integer,parameter :: row=6, col=4 ! Set constant grid size
10 real,dimension(row,col) :: H, V ! Linear & non-linear (Dupuit) heads
11 ! DEFINE
12 ! Fill arrays with placeholders
13 do i=1,row
14     do j=1,col
15         H(i,j)=0. ! Linear
16         V(i,j)=H(i,j)**2. ! Non linear
17     end do
18 end do
19 numit=0 ! Set iteration counter
20 maxit=100 ! Set maximum iterations
21 converg=0.01 ! Set convergence criteria
22 error=0. ! Initialize error
23 maxerr=2.*converg ! Enable initial entry to computation loop
24 oldval=0. ! Initialize old value
25 ! Left specified head boundary
26 do i=2,row-1
27     H(i,1)=4.
28     V(i,1)=H(i,1)**2.
29 end do
30 ! Right specified head boundary
31 do i=2,row-1
32     H(i,col)=3.
33     V(i,col)=H(i,col)**2.
34 end do
35 ! Initial guess at inner nodes
36 do i=2,row-1
37     do j=2,col-1
38         H(i,j)=4.
39         V(i,j)=H(i,j)**2.
40     end do
41 end do
42 ! COMPUTE
43 do
44 if(maxerr<=converg .or. numit>=maxit) exit
45 numit=numit+1
46 maxerr=0.
47 do j=1,col
48     V(1,j)=V(1+2,j) ! Top specified-flux boundary
49     V(row,j)=V(row-2,j) ! Bottom specified-flux boundary
50 end do
51 do i=2,row-1
52     do j=2,col-1
```

```

53         oldval=V(i,j)
54         V(i,j)=(V(i,j+1)+V(i,j-1)+V(i-1,j)+V(i+1,j))/4.
55         error=abs(V(i,j)-oldval)
56         if(error>maxerr) then
57             maxerr=error
58         end if
59     end do
60 end do
61 end do
62 do i=1,row
63     do j=1,col
64         H(i,j)=V(i,j)**(1./2.) ! Linearize heads
65     end do
66 end do
67 ! OUTPUT
68 ! Write to command line
69 write(*,*) "heads:"
70 write(*,10) ((H(i,j),j=1,col),i=1,row)
71 10 format(1x,4f8.2)
72 write(*,11) numit
73 11 format(" number of iterations = ", i3)
74 ! Write to text file
75 open(unit=100, file="head_ex04.txt")
76 write(100,12) ((H(i,j),j=1,col),i=2,row-1)
77 12 format(1x,4f8.2)
78 close(100)
79 end program code_ex04

```

Ex. 04: 2D Steady Flow thru Dam

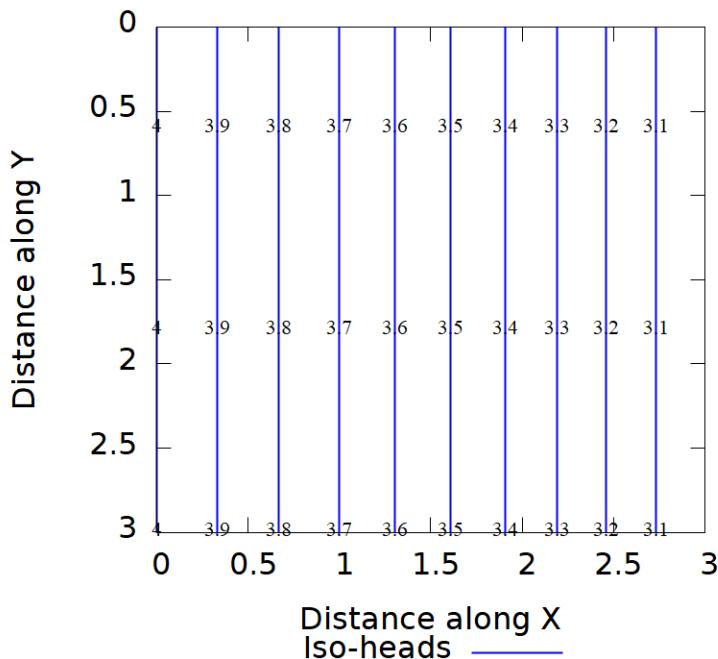


Figure 16 Visualization of Fortan program (Figure 15)

7. Case Five: Two-dimensional Steady Flow to a Phreatic Well

Figure 17 Computation Fortran program for Case Five

```
1 program code_ex05
2 ! Similar to Figure 3.12 in Wang & Anderson (1982)
3 ! Computer program for solving phreatic drawdown, Dupuit assumptions.
4 ! Code written by C.J. Paradis (paradisc@uwm.edu)
5 implicit none
6 ! DECLARE
7 integer :: i, j, numit, maxit ! Indices & counters
8 real :: converg, error, maxerr, oldval ! Converge, errors, & old vals
9 real :: dx, omega ! Grid space & relaxation
10 real :: Q, K, R ! Aquifer parameters and recharge
11 integer,parameter :: row=12, col=12 ! Set constant grid size
12 real,dimension(row,col) :: V, H ! Linear and non-linear (Dupuit) heads
13 real,dimension(row,col) :: Rfield ! Recharge field
14 real,dimension(row,col) :: radfield, conheads ! Radial dist. & boundaries
15 ! DEFINE
16 ! Fill arrays with placeholders
17 do i=1,row
18     do j=1,col
19         H(i,j)=10. ! Linear
20         V(i,j)=H(i,j)**2. ! Non linear
21         Rfield(i,j)=0.
22         radfield(i,j)=0.
23         conheads(i,j)=0.
24     end do
25 end do
26 numit=0 ! Set iteration counter
27 maxit=100 ! Set maximum iterations
28 converg=0.001 ! Set convergence critera
29 error=0. ! Initialize error
30 maxerr=2.*converg ! Enable initial entry to computation loop
31 oldval=0. ! Initialize old value
32 dx=200. ! Grid spacing
33 omega=1.8 ! Relaxation
34 Q=-2000. ! Pumping rate
35 K=30. ! Conductivity
36 R=Q/(dx*dx) ! Recharge rate
37 ! Specified head boundary at radial distance from well
38 do i=row-1,1,-1
39     do j=2,col
40         radfield(i,j)=(((row-1-i)*dx)**2.+((2-j)*dx)**2.)**(.1./2.)
41     end do
42 end do
43 do i=1,row-1
44     do j=2,col
45         if(radfield(i,j)>=1950.) then
46             conheads(i,j)=10.***2. ! Constant head boundary at ~ 2000 m
47         end if
48     end do
49 end do
50 Rfield(row-1,2)=R ! Pumping well location
51 ! COMPUTE
52 do
```

```

53 if(maxerr<=converg .or. numit>=maxit) exit
54 numit=numit+1
55 maxerr=0.
56   do j=1,col
57     V(col,j)=V(col-2,j) ! Bottom specified-flux boundary
58   end do
59   do i=1,row
60     V(i,1)=V(i,3) ! Left specified-flux boundary
61   end do
62   do i=2,row-1
63     do j=2,col
64       if(conheads(i,j)==10.**2.) then
65         V(i,j)=conheads(i,j)
66       else
67         oldval=V(i,j)
68         V(i,j)=(V(i-1,j)+V(i+1,j)+V(i,j-1)+V(i,j+1)+&
69         dx*dx*2.*Rfield(i,j)/K)/4.
70         V(i,j)=omega*V(i,j)+(1.-omega)*oldval
71         error=abs(V(i,j)-oldval)
72         if(error>maxerr) then
73           maxerr=error
74         end if
75       end if
76     end do
77   end do
78 end do
79 do i=1,row
80   do j=1,col
81     H(i,j)=V(i,j)**(1./2.) ! Linearize heads
82   end do
83 end do
84 ! OUTPUT
85 ! Write to command line
86 write(*,*) "heads:"
87 write(*,10) ((H(i,j),j=1,col),i=1,row)
88 10 format(1x,12f8.2)
89 write(*,11) numit
90 11 format(" number of iterations = ", i3)
91 ! Write to text file
92 open(unit=100, file="head_ex05.txt")
93 write(100,12) ((H(i,j),j=2,col),i=1,row-1)
94 12 format(1x,11f8.2)
95 close(100)
96 end program code_ex05

```

Ex. 05: 2D Steady Flow to Phreatic Well

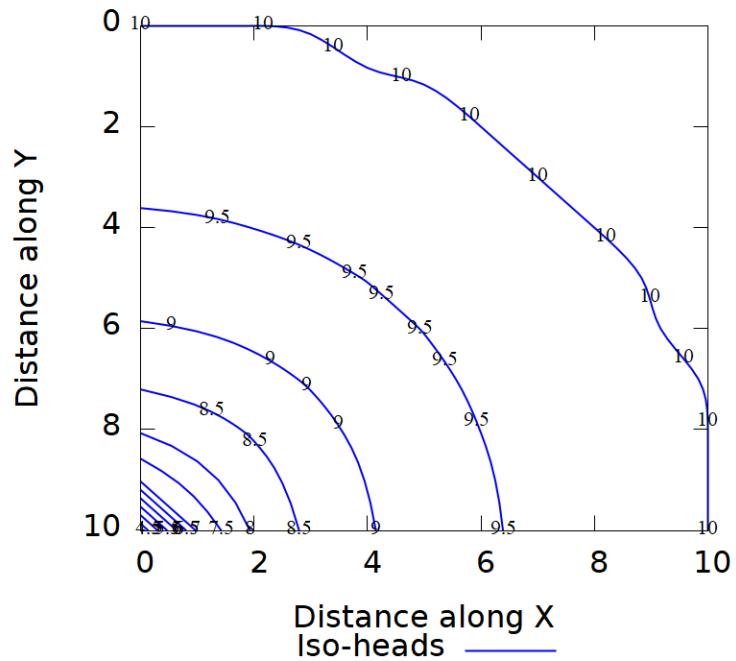


Figure 18 Visualization of Fortan program (Figure 17)

8. Case Six: One-dimensional Transient Flow with Head Change

Figure 18 Computation Fortran program for Case Six

```
1 program code_ex06
2 ! Similar to Figure 4.2 in Wang & Anderson (1982)
3 ! Computer program for solving reservoir problem, explicit technique
4 ! Code written by C.J. Paradis (paradisc@uwm.edu)
5 implicit none
6 ! DECLARE
7 integer :: i, j, n, nend, kount1, kount2, kount3 ! Indices & counters
8 integer, parameter :: row=100,col=11 ! Set constant grid size
9 real :: delx, delt, ho, hr, time ! Space & time steps, boundaries, & time
10 real :: T, S ! Aquifer parameters
11 real :: A, B ! Computational placeholders
12 real,dimension(col) :: hold, hnew ! Old heads and new heads
13 real,dimension(row,col) :: head ! Output heads
14 real,dimension(col,row+1) :: tranhead ! Transposed output heads
15 ! DEFINE
16 delx=10. ! Space step
17 delt=5. ! Time step
18 T=0.02 ! Aquifer transmissivity
19 S=0.002 ! Aquifer storativity
20 ho=16. ! Specified head at inlet
21 hr=11. ! Specified head at outlet
22 ! Fill arrays initial heads
23 do i=1,col
24     hold(i)=ho
25     hnew(i)=ho
26 end do
27 ! Change outlet head to sudden decrease
28 hold(col)=hr
29 hnew(col)=hr
30 nend=100
31 time=0.
32 kount1=0
33 kount2=2
34 kount3=1
35 ! COMPUTE & OUTPUT
36 open(unit=100, file="head_ex06.txt")
37 write(*,10) "Heads","Time" ! Write header for heads and time
38 10 format(/,39x,a5,52x,a4/)
39 do n=1,nend
40     do i=2,col-1
41         A=(delt*T)/S
42         B=(hold(i+1)-2.*hold(i)+hold(i-1))/(delx**2.)
43         hnew(i)=hold(i)+A*B
44     end do
45     do i=2,col-1
46         hold(i)=hnew(i)
47     end do
48     time=time+delt
49     kount1=1+kount1
50     do j=1,col
51         head(kount3,j)=hnew(j)
52     end do
```

```

53     kount3=kount3+1
54     if(kount1==kount2) then ! Write results every other time step
55     write(*,11) (hnew(i),i=1,col),time
56     11 format(1x,11f8.2,1f12.2)
57     kount1=0 ! Reset counter 1 at every other time step
58   end if
59 end do
60 do i=1,col
61   tranhead(i,1)=(i-1)*delx
62 end do
63 do j=1,col
64   do i=1,row
65     tranhead(j,i+1)=head(i,j)
66   end do
67 end do
68 !OUTPUT
69 write(100,12) ((tranhead(i,j),j=1,row+1),i=1,col)
70 12 format(1x,101f8.2)
71 close(100)
72 end program code_ex06

```

Ex. 06: 1D Unsteady Flow Head Change

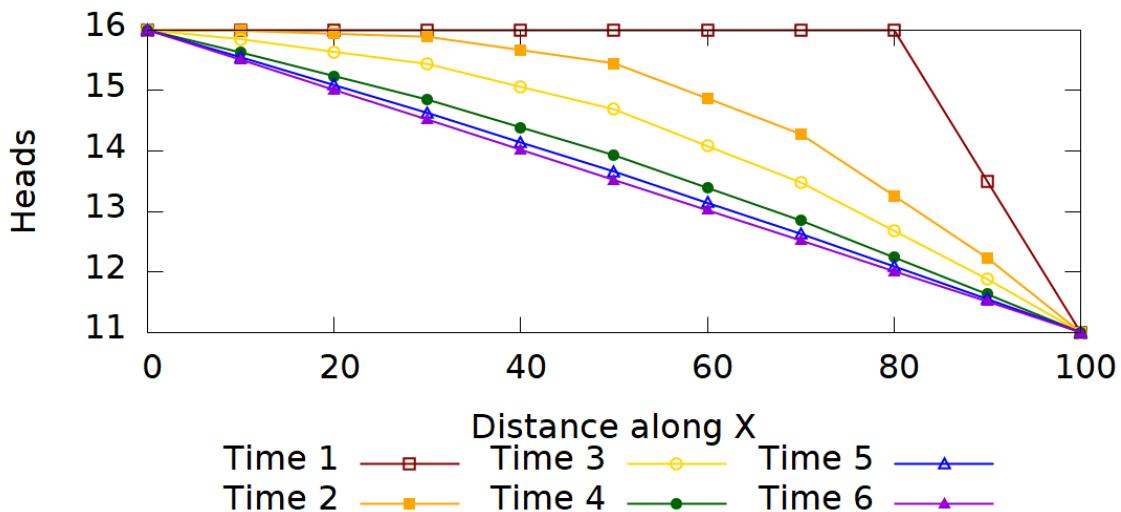


Figure 19 Visualization of Fortan program (Figure 18)

9. Case Seven: Two-dimensional Transient Flow to a Well

Figure 20 Computation Fortran program for Case Seven

```

1  program code_ex07
2 ! Similar to Figure 4.6 in Wang & Anderson (1982)
3 ! Computer program for solving transient flow to a confined well
4 ! Code written by C.J. Paradis (paradisc@uwm.edu)
5 implicit none
6 !DECLARE
7 integer :: i, j, numit, maxit, n, nend
8 integer, parameter :: row=23, col=23
9 real, dimension (row,col) :: Hnew, Hold, R, DD
10 real :: delx, delt, Q, T, S, Ho, time
11 real :: alpha, converg, error, maxerr, oldval
12 real :: C1, C2, H1, H2
13 ! DEFINE
14 delx=100.
15 delt=0.01
16 Q=-2000.
17 T=300.
18 S=0.002
19 Ho=10.
20 time=0.
21 numit=0
22 maxit=200
23 nend=16
24 alpha=0.5
25 converg=0.001
26 error=0.
27 maxerr=2.*converg
28 oldval=0.
29 do i=1,row
30   do j=1,col
31     Hold(i,j)=Ho
32     Hnew(i,j)=Ho
33     R(i,j)=0.
34     DD(i,j)=0.
35   end do
36 end do
37 R(row-1,2)=Q/(delx*delt)
38 H1=0.
39 H2=0.
40 ! COMPUTE
41 do n=1,nend
42 maxerr=2.*converg
43 numit=0
44 time=time+delt
45   do
46     if(maxerr<=converg .or. numit>maxit) exit
47     numit=numit+1
48     maxerr=0.
49     do i=2,row-1
50       do j=2,col-1
51         oldval=Hnew(i,j)
52         H1=(Hold(i-1,j)+Hold(i+1,j)+Hold(i,j-1)+Hold(i,j+1))/4.
```

```

53      H2=(Hnew(i-1,j)+Hnew(i+1,j)+Hnew(i,j-1)+Hnew(i,j+1))/4.
54      C1=(delx*delx*S)/(4.*T*delt)
55      C2=1./(C1+alpha)
56      Hnew(i,j)=C2*(alpha*H2+C1*Hold(i,j)+(1.-alpha)*&
57      (H1-Hold(i,j))+(delx*delx*R(i,j))/(4.*T))
58      error=abs(Hnew(i,j)-oldval)
59      if(error>maxerr) then
60          maxerr=error
61      end if
62      end do
63  end do
64  do i=2,row-1
65      Hnew(i,1)=Hnew(i,3)
66      Hnew(i,col)=Hnew(i,col-2)
67  end do
68  do j=2,col-1
69      Hnew(1,j)=Hnew(3,j)
70      Hnew(row,j)=Hnew(row-2,j)
71  end do
72 end do
73 do i=1,row
74     do j=1,col
75         Hold(i,j)=Hnew(i,j)
76     end do
77 end do
78 do i=2,row-1
79     do j=2,col-1
80         DD(i,j)=Ho-Hnew(i,j)
81     end do
82 end do
83 if(n<=nend-1) then
84     delt=delt*1.5
85 end if
86 if(delt>5.0) then
87     delt=5.0
88 end if
89 end do
! OUTPUT
90 write(*,10) time, delt, numit
91 10 format("time = ",f8.2,/, "delt = ",f8.2,/, "numit = ",i8)
92 write(*,20) ((DD(i,j),j=2,col-1),i=2,row-1)
93 20 format(1x,21f8.2)
94 open(unit=100, file="head_ex07.txt")
95 write(100,30) ((Hnew(i,j),j=2,col-1),i=2,row-1)
96 30 format(1x,21f8.2)
97 close(100)
98 end program code_ex07

```

Ex. 07: 2D Transient Flow to Well

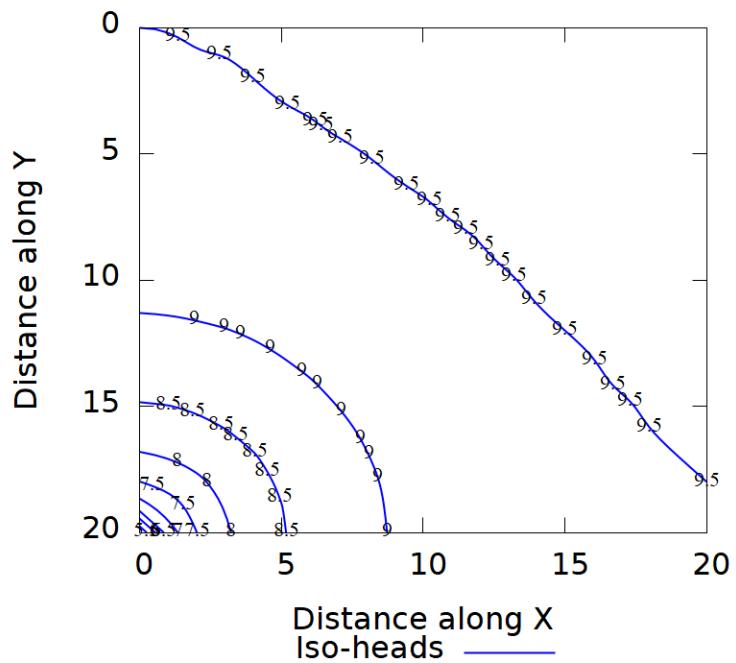


Figure 21 Visualization of Fortan program (Figure 20)