CE 5366 Water Resources Management
Concepts and Tools for Decision Making

by
Theodore G. Cleveland
Department of Civil, Environmental, and Construction Engineering
Texas Tech University

January 2018

Suggested citation: Cleveland, T. G. 2018. "Water Resources Management; Concepts and Tools for Decision Making." CE 5366 Lecture Notes, Department of Civil, Environmental, and Construction Engineering, Texas Tech University.

Contents

1. Introduction

words

1.1. What is Water Resources Management?

Water resources management is:

> The activity of planning, developing, distributing and managing the optimum use of water resources. Water resource management planning considers the competing demands for/ on water (quantity, quality, location, and timing) and seeks to allocate water on an equitable basis to satisfy all uses and demands.

Paraphrasing WRSPM:

> The central purpose of water resources planning and management activities is to address and, if possible, answer questions such as:
> How can renewable yet finite water resources (e.g. rivers, estuaries, lakes, glaciers, and coastal zones) best be managed and used? How can this management and use be accomplished in an environment of uncertain supplies and uncertain and increasing demands, and consequently of increasing conflicts among individuals having different interests in the management of a river and its basin?

Key concepts/terms are

- Finite (limited) water resource.
- How much, when, how yummy.
- Allocate to many.
- Equity/conflicts.
- "Best be managed ..."

1.1.1. What is "Allocate"

Allocate means

1: to apportion for a specific purpose or to particular persons or things : distribute allocate tasks among human and automated components
2: to set apart or earmark : designate allocate a section of the building for special research purposes

If the resource is abundant then allocation is unnecessary. However, water resources are identified as finite, hence limited in scope both spatially and temporally.

Because the resource is scarce; allocation will necessarily deny the resource to some, and supply it to others.

1.1.2. What is "Equitable"

Equitable means

1: having or exhibiting equity : dealing fairly and equally with all concerned an equitable settlement of the dispute
2: existing or valid in equity as distinguished from law an equitable defense

Because the resource is scarce; allocation will necessarily deny the resource to some, and supply it to others. Equity implies some kind of ?fairness? in that allocation.

1.2. Decision Making

The allocation requires decisions be made. Decisions (policy) may involve the activity of an individual decision maker or collective decision maker(s). These two classifications may make decisions quite differently, based on their roles, emotions, and goals.

1.2.1. How do individuals and groups make decisions?

Some people may seem indecisive or inconsistent. They may avoid making decisions as long as possible. The same reluctance to make decisions also happens in organizations or institutions. The decision avoidance may be an individual or collective desire to address uncertainty, in waiting, more information may become available.

On the other hand, some people and groups may be accused of "jumping to conclusions" or making a decision "without considering all the facts." The decision acceleration may be an individual or collective assessment of the utility of more information,

or a value judgement of the importance of the actual situation to longer term goals, or acknowledgement of urgency, or just plain laziness – all of these explanations are clearly situational dependent.

This dichotomy itself contains a valuable implication that, in general, decisions should be made by collecting and weighing various elements in a rational way. Furthermore, alternatives must exist, or there is no decision to make.

Decision making contains deep psychological aspects. People's choices depend upon how alternatives are presented – consider Figure 1.

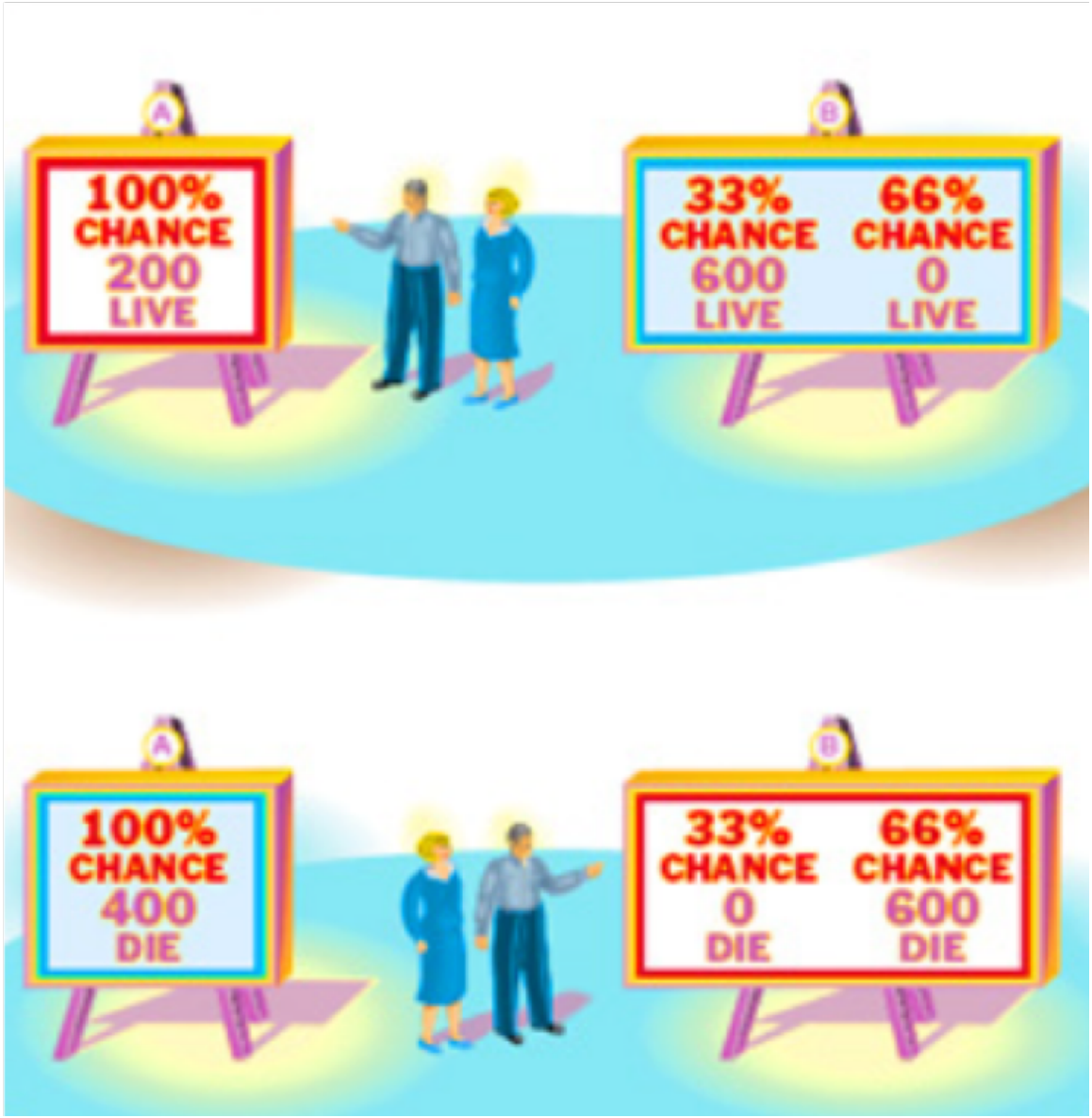


Figure 1. Identical decisions. Upper panel is "Survival Based"; Lower panel is "Mortality Based" .

Consider the decision A or B, the actual outcomes in terms of probability and number of survivors of the decision.

If we make decision A. The probability P of the outcomes are $P(S = 200) = \frac{1}{1}; P(S = 0) = \frac{0}{1}$. If we make decision B. The probability P of the outcomes are $P(S = 600) = \frac{1}{3}; P(S = 0) = \frac{2}{3}$.

Expectation in this structure is the product of the value of the outcome and the probability of that outcome. Thus $E(S_A) = 1.0 \times 200 + 0.0 \times 0 = 200$ people, and $E(S_B) = \frac{1}{3} \times 600 + \frac{2}{3} \times 0 = 200$ people.

Whichever decision is chosen A or B has the same expected outcome (in terms of expectation in the statistical sense); 200 people survive. So one might conclude it does not matter which decision we make – but that's not the whole story.

While decision A guarantees 200 survive, decision B has a 66% chance or zero survivors – which is intended here to illustrate the influence of uncertainty in decision making.

Furthermore, how the situation is presented (Survival Based or Mortality Based) can have a severe emotional influence on how one would make the choice.

Lastly, consider the exogenous influences. If the people in the example pay taxes, then the cost of the decision could influence the actual decision.

This simple example illustrates the challenges of something seemingly as simple as making a decision.

1.3. Challenge of Water Resources Management

There are actually only a few challenges (at a very high cognitive level):

- Define alternatives (present decision points)
- Define value
- Define outcomes
- Define equity; "what is fair" for a situation
- Implement tools to support (defend) the decision
- Implement the decision(s)

1.3.1. Support Tools

One of the bullet items is tools to support the decision. These include economics or some other way to value the alternatives (cost and benefit), as well as measure "fairness" Then some policy (or procedure) to rank the value(s). Another policy or procedure to guide the decision – in some instances we could hand off the decision to an algorithm (Narrow Domain AI). We already let such algorithms fly aircraft with

us in them, recommend products for us to purchase, detect fraudulent fiscal activity and such. Another important factor in the support/implementation component is what level of control do we have.

- Do we control the system?
  SCADA operating a water distribution network – fully autonomous with little human intervention.
- ... or just influence the system
  Voluntary compliance versus risk of apprehension for water quality management

2. Engineering Economy

words

2.1. Time Value of Money

2.1.1. Example

2.2. Comparing Alternatives

2.2.1. Example

2.3. Perspective

2.3.1. Example

2.4. Readings

2.5. Exercises

3. Title

words

3.1. Topic 1

3.1.1. Example

3.2. Topic 2

3.2.1. Example

3.3. Topic 3

3.3.1. Example

3.4. Readings

3.5. Exercises

4. Optimization Tools – Linear Programming as a Decision Support Component

words

4.1. Overview of LP as a Decision Support Tool

4.1.1. Example

4.2. Primal-Dual Relationship

4.2.1. Example

4.3. Interpreting Results

4.3.1. Example

4.4. Readings

4.5. Exercises

5. Solving Linear Programs using R

words

5.1. Installing `lpSolve` package

5.1.1. Example

5.2. Solving a LP using `lpSolve`

5.2.1. Example

5.3. Interpreting Results

5.3.1. Example

5.4. Readings

5.5. Exercises

6. Optimization Tools - Non-Linear Programming as a Decision Support Component

words

6.1. Nonlinear Programming

6.1.1. Example

6.2. Unconstrained Non-Linear Programs

6.2.1. Example

6.3. Constrained Non-Linear Programs

6.3.1. Example

6.4. Readings

6.5. Exercises

7. Solving Non-Linear Programs using R

Solving constrained non-linear optimization problems of any meaningful size is non-trivial. Sometimes just finding a feasible solution is difficult (especially for large-scale problems). This chapter presents the **R** package `nloptr` and its dependencies for solving non-linear probelms.

7.1. Packages

The first step is to install the package `nloptr` from the CRAN. Figure 30 shows the procedure to install the package using the package manager. Be sure to select "Install Dependencies" so that the manager installs related programs needed for the solver tools to work. The package itself is a front-end (interface) for several different packages contained in the CRAN, all of which need to be present on your computer for the tool to work.

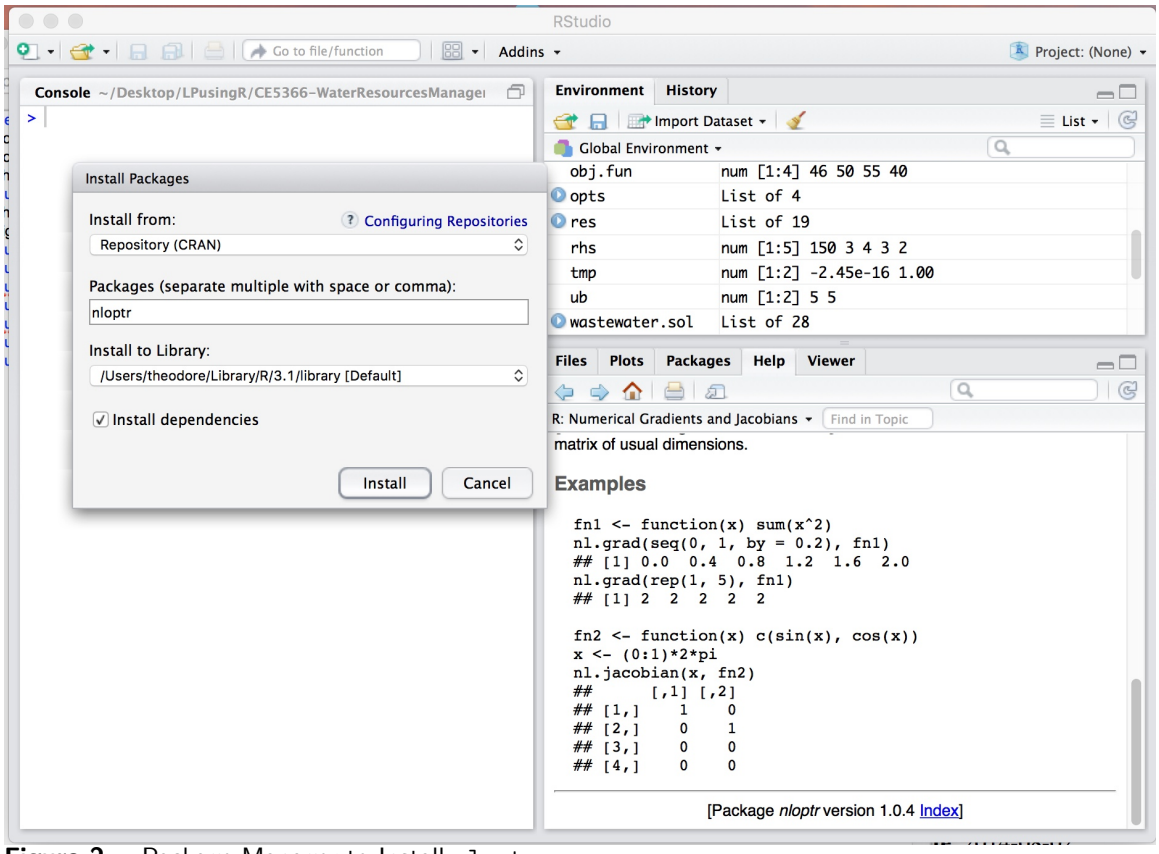


Figure 2. Package Manager to Install `nloptr`.

Upon successful install the rather simple message in Figure 31 is displayed that shows the install statistics and the location of the binaries on your computer.

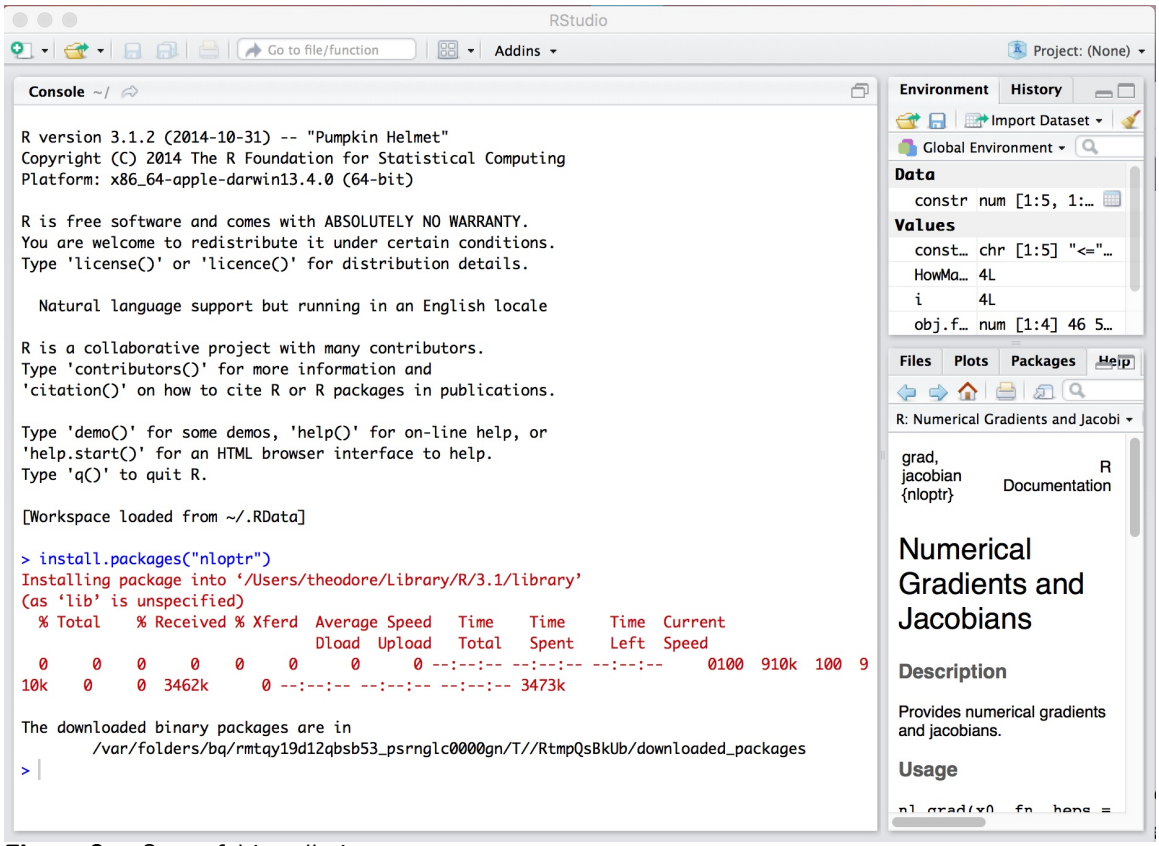


Figure 3. Succesful installation `nloptr`.

7.1.1. Example 1

The example in Figure 32 is used to test the install

$$\begin{aligned} &\textit{minimize} \quad x_1 \times x_4 \times (x_1 + x_2 + x_3) + x_3 \\ &\textit{subject to} \end{aligned}$$

$$\begin{array}{rcl} x_1 \times x_2 \times x_3 \times x_4 & \geq & 25 \\ \hline x_1^2 + x_2^2 + x_3^2 + x_4^2 & = & 40 \\ \hline x_1 & \geq & 1 \\ x_2 & \geq & 1 \\ x_3 & \geq & 1 \\ x_4 & \geq & 1 \\ \hline x_1 & \leq & 5 \\ x_2 & \leq & 5 \\ x_3 & \leq & 5 \\ x_4 & \leq & 5 \end{array}$$

Figure 4. Non-Linear Program Structure for Example 1 Problem.

Listing 20 is the example script lifted directly from the package documentation showing how the mathematics in Figure 32 is converted into the syntax for the non-linear program solver.

Listing 1. R code to construct and solve a NLP Model.

```
# Example problem, number 71 from the Hock-Schittkowsky test suite.
#
# \min_{x} x1*x4*(x1 + x2 + x3) + x3
# s.t.
#    x1*x2*x3*x4 >= 25
#    x1^2 + x2^2 + x3^2 + x4^2 = 40
#    1 <= x1,x2,x3,x4 <= 5
#
# we re-write the inequality as
#   25 - x1*x2*x3*x4 <= 0
#
# and the equality as
#   x1^2 + x2^2 + x3^2 + x4^2 - 40 = 0
#
# x0 = (1,5,5,1)
#
# optimal solution = (1.00000000, 4.74299963, 3.82114998, 1.37940829)

library('nloptr')
#
# f(x) = x1*x4*(x1 + x2 + x3) + x3
#
eval_f <- function( x ) {
  return( list( "objective" = x[1]*x[4]*(x[1] + x[2] + x[3]) + x[3],
                "gradient" = c( x[1] * x[4] + x[4] * (x[1] + x[2] + x[3]),
                                x[1] * x[4],
                                x[1] * x[4] + 1.0,
                                x[1] * (x[1] + x[2] + x[3]) ) ) )
}
# constraint functions
# inequalities
eval_g_ineq <- function( x ) {
  constr <- c( 25 - x[1] * x[2] * x[3] * x[4] )
  grad   <- c( -x[2]*x[3]*x[4],
               -x[1]*x[3]*x[4],
               -x[1]*x[2]*x[4],
               -x[1]*x[2]*x[3] )
  return( list( "constraints"=constr, "jacobian"=grad ) )
}
# equalities
eval_g_eq <- function( x ) {
  constr <- c( x[1]^2 + x[2]^2 + x[3]^2 + x[4]^2 - 40 )
  grad   <- c(  2.0*x[1],
                2.0*x[2],
                2.0*x[3],
                2.0*x[4] )
  return( list( "constraints"=constr, "jacobian"=grad ) )
}
# initial values
x0 <- c( 1, 5, 5, 1 )
# lower and upper bounds of control
lb <- c( 1, 1, 1, 1 )
ub <- c( 5, 5, 5, 5 )
local_opts <- list( "algorithm" = "NLOPT_LD_MMA",
                    "xtol_rel"  = 1.0e-7 )
opts <- list( "algorithm" = "NLOPT_LD_AUGLAG",
              "xtol_rel"  = 1.0e-7,
              "maxeval"   = 1000,
              "local_opts" = local_opts )
res <- nloptr( x0=x0,
               eval_f=eval_f,

               lb=lb,
               ub=ub,
               eval_g_ineq=eval_g_ineq,
               eval_g_eq=eval_g_eq,
               opts=opts)

print( res )
```

An important observation in the listing is that gradients and Jacobians are supplied as analytical functions of the input variables. Later we will attempt to replace these with numerical approximations (to generalize the problem solving tool somewhat).

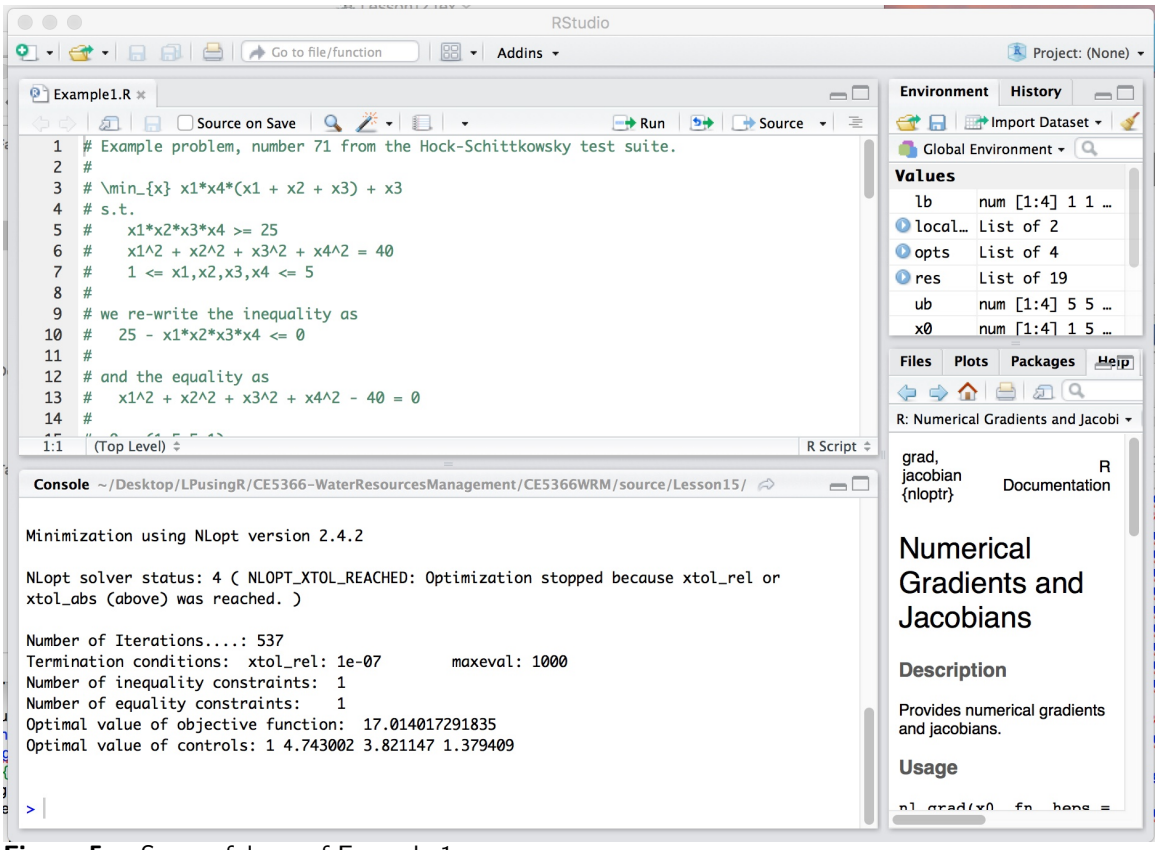


Figure 5. Successful run of Example 1.

Figure 33 is the RStudio output when the example is run. With the successful run, we conclude that the package is installed and runs as anticipated.

7.1.2. Example 2

Another test example from pg 209 of (**?**) is

$$
\begin{aligned}
&\textit{minimize} \quad x_1 \times x_2^2 \\
&\textit{subject to}
\end{aligned}
$$

$2 - x_1^2 \times -x_2^2$	$\geq$	0
x_1	$\geq$	-5
x_2	$\geq$	-5
x_1	$\leq$	5
x_2	$\leq$	5

Figure 6. Non-Linear Program Structure for Example 2 Problem.

Listing 21 is the script to solve this example. Observe there is no inequality constraint, so that part of the solver call is suppressed.

Listing 2. R code to construct and solve a NLP Model.

```
# Example problem, pg 209 Gill, Murray, and Wright
#
# \min_{x} x1*(x2^2)
#
# s.t.
#    2 - x1^2 - x2^2 >= 0
#
# we re-write the inequality as
#   x1^2 + x2^2 - 2 <= 0
#
#
# x0 = (-1,-1)
#
# optimal solution = (-0.81650,-1.1547)

library('nloptr')
#
# f(x) = x1*x4*(x1 + x2 + x3) + x3
#
eval_f <- function( x ) {
  return( list( "objective" = x[1] * x[2]^2,
                "gradient" = c( x[2]^2,2*x[1]*x[2])
              )
          )
}
# constraint functions
# inequalities
eval_g_ineq <- function( x ) {
  constr <- c(x[1]^2 + x[2]^2 -2 )
  grad   <- c( 2*x[1],
               2*x[2] )
  return( list( "constraints"=constr, "jacobian"=grad ) )
}
# equalities
# eval_g_eq <- function( x ) {
#   constr <- c( x[1]^2 + x[2]^2 + x[3]^2 + x[4]^2 - 40 )
#   grad   <- c(  2.0*x[1],
#                 2.0*x[2],
#                 2.0*x[3],
#                 2.0*x[4] )
#   return( list( "constraints"=constr, "jacobian"=grad ) )
# }
# initial values
x0 <- c( -1,-1 )
# lower and upper bounds of control
lb <- c( -5, -5 )
ub <- c( 5, 5 )
local_opts <- list( "algorithm" = "NLOPT_LD_MMA",
                    "xtol_rel"  = 1.0e-7 )
opts <- list( "algorithm" = "NLOPT_LD_AUGLAG",
              "xtol_rel"  = 1.0e-7,
              "maxeval"   = 1000,
              "local_opts" = local_opts )
res <- nloptr( x0=x0,
               eval_f=eval_f,

               lb=lb,
               ub=ub,
               eval_g_ineq=eval_g_ineq,
#               eval_g_eq=eval_g_eq,
               opts=opts)

print( res )
```

Figure 35 shows a successful run (and the optimal solution) for the example problem. So now we have a rudimentary understanding of the solver's syntax.

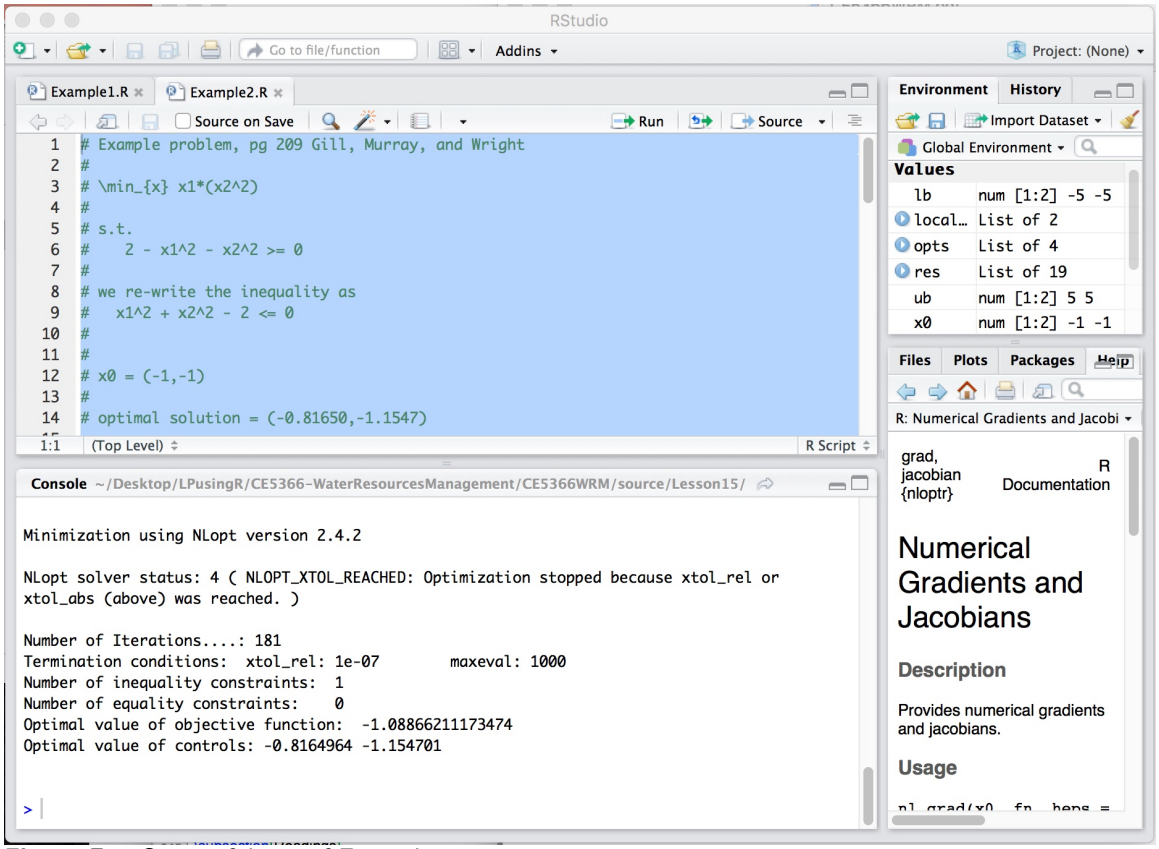


Figure 7. Successful run of Example 2.

7.2. Topic 2 : An Equitable Price Strategy

7.2.1. Example: Bus Transit Price Structure

An transit company has determined the average cost to provide service on a two-route bus system is \$1.20 per customer. The company wants to establish a fare schedule to account for the trip length of the rider and the ridership on each route. The route with the greater ridership may pay a fare that is less than the fare paid by users of the route with lower ridership. Similarly, users of the longer route may have to pay a fare greater than users on the shorter route.

Figure 36 depicts the layout of the bus system. Table 4 lists information regarding the two routes.

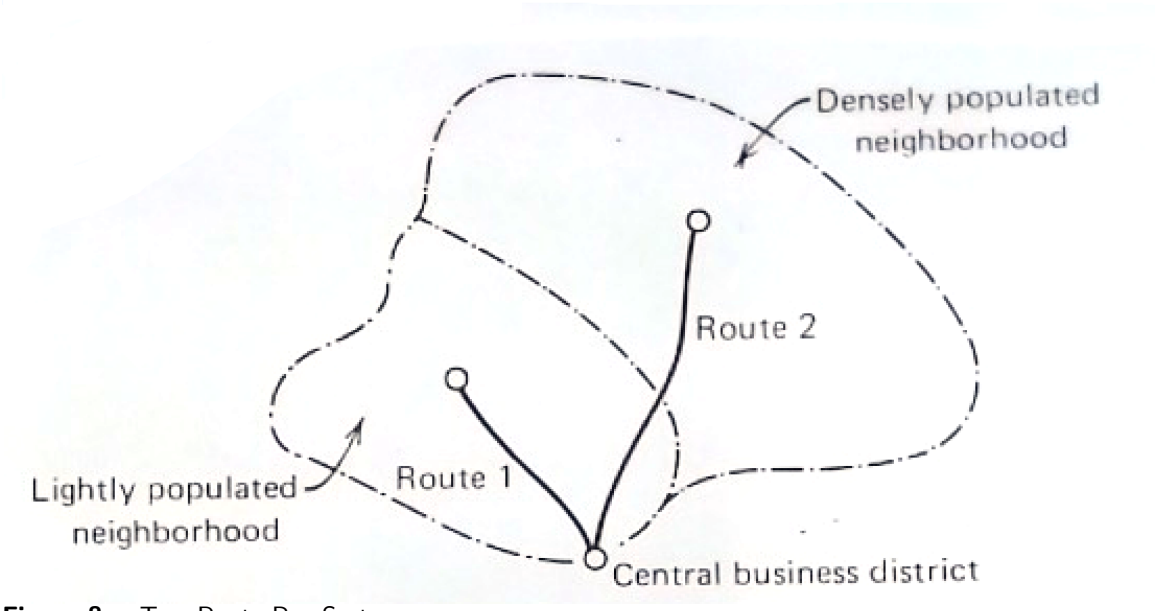


Figure 8. Two-Route Bus System.

Table 1. Transit System Statistics.

Route i	Ridership Average Trip Length L_i	n_i, Average daily ridership
1	10 miles	600
2	15 miles	1,400

One metric of equity is that an equitable system would minimize the variance of the travel cost among its customers. This measure of effectiveness (a weighted variance equation) is

$$V = \frac{1}{N} \sum_{i=1}^{N} n_i (r_i - \bar{r})^2 \tag{1}$$

where V is the variance of the travel cost; r_i is the fare box revenue paid by each customer on route i; $\bar{r}$ is the average fare box revenue paid by each customer [1] ; and N is the total number of customers.

Using the supplied information formulate a mathematical model to determine an equitable bus fare schedule that charges each customer on a cost per passenger-mile basis, then find an optimal solution using constrained (or unconstrained) non-linear programming in **R**.

Analysis Let the price to be charged on each route be the decision variable equal to the cost per passenger-mile on each route.

$p_1 =$ unit fare in cost per passenger-mile on route 1
$p_2 =$ unit fare in cost per passenger-mile on route 2

The merit function is the weighted variance of travel cost among the customers:

$$V = \frac{n_1}{N}(r_1 - \bar{r})^2 + \frac{n_2}{N}(r_2 - \bar{r})^2$$

The revenue from a unit fare for each route is:

$$r_1 = p_1 \times L_1 = 10p_1$$
$$r_2 = p_2 \times L_2 = 15p_2$$

Substituting these values into the merit function yields:

$$V = \frac{600}{2000}(10p_1 - 1.20)^2 + \frac{1400}{2000}(15p_2 - 1.20)^2$$

or

$$V = 0.3(10p_1 - 1.20)^2 + 0.7(15p_2 - 1.20)^2$$

The bus company is assumed to be a not-for-profit operation so that it is constrained to have revenue equal costs. This constraint is expressed as:

$$p_1(n_1L_1) + p_2(n_2L_2) = \bar{r}N$$

or

$$p_1(600\dot{1}0) + p_2(1400\dot{1}5) = \$1.20\dot{2}000$$

or

[1] $\bar{r} = \frac{1}{N}\sum_{i=1}^{N} n_i r_i$

$6p_1 + 21p_2 = \$2.40$

Problem Set-Up
Using the constraints and performance criteria we can construct the optimization problem as

$minimize \quad V = 0.3(10p_1 - 1.20)^2 + 0.7(15p_2 - 1.20)^2$
subject to

$6p_1$	$+ 21p_2$	$=$	$\$2.40$
p_1		$\geq$	0
	p_2	$\geq$	0

Solution Listing **??** is the **R** script that can be used to find the optimal solution to the problem.

7.3. Topic 3 : Water Resources Allocation/Operation Problem

7.3.1. Example: Water Delivery Price Structure

7.4. Topic 4: Water Quality Allocation Problem

7.4.1. Example: Reactor-Tank System

Consider a biological reactor system using the two-vessel feedback schematic in Figure 37. The system uses influent substrate, S, as an energy source (think food) to grow biomass, X, in the first vessel. The mixed liquor is then transferred to a clarifier tank where the liquid is decanted and the biomass is concentrated. The decanted liquid is returned to the environment (you can visualize the diagram as a water treatment system, where the substrate is some pollutant). The concentrated biomass is recycled back to the reactor tank as a way to ensure that the biomass population is sufficient for substrate removal.

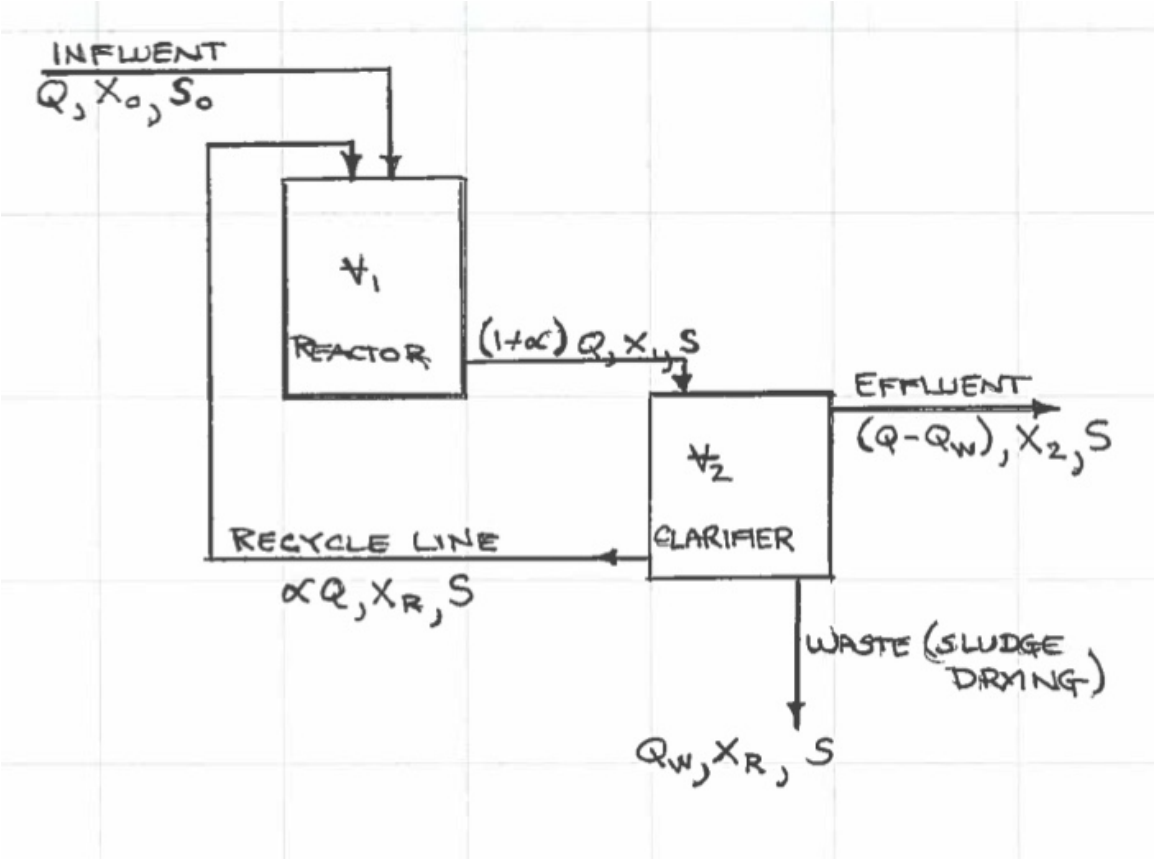


Figure 9. Feedback Biological Reactor Schematic.

In practice a portion of the concentrated biomass is wasted from the system to maintain constant system volumes (and control sludge age and other important features exogenous to the example).

The cost per vessel is a function of the vessel volume in gallons and is expressed as

$$C = \$25.119 \times V^{0.6} \tag{2}$$

The equilibrium inflow conditions are $Q = 1 \times 10^6$ gallons per day, $S_0 = 200$ ppm, and $X_0 = 0$ ppm.[2]

The system design guidelines call for the clarifier overflow loading rate to be $Q_O = 800$ gal/day/ft^2, the clarifier depth h to be 8 feet, and the effluent substrate concentration to be $S = 20$ ppm.

Past design experience with similar systems suggests a cell growth rate of $\mu = 1.876$ days^{-1} and a biosolids yield coefficient $Y = \frac{X}{S} = 0.449$.

Using this information design a minimum cost reactor system to satisfy the desired output water quality.

Analysis
The reactor mass balance is used to establish various constraints for the problem.

The biosolids mass balance is:

$$V_1 \frac{dX_1}{dt} = QX_0 + \alpha QX_R - (1+\alpha)QX_1 + \mu X_1 V_1 \tag{3}$$

Now divide by V_1 to obtain:

$$\frac{dX_1}{dt} = \frac{Q}{V_1}X_0 + \alpha\frac{Q}{V_1}X_R - (1+\alpha)\frac{Q}{V_1}X_1 + \mu X_1 \tag{4}$$

Introduce the hydraulic retention time of the reactor as $T_1 = \frac{V_1}{Q}$, and substitute this value into the equation to obtain:

$$\frac{dX_1}{dt} = \frac{X_0}{T_1} + \alpha\frac{X_R}{T_1} - (1+\alpha)\frac{X_1}{T_1} + \mu X_1 \tag{5}$$

The substrate (pollutant) mass balance is

$$V_1 \frac{dS}{dt} = QS_0 + \alpha QS - (1+\alpha)QS - \frac{\mu}{Y}X_1 V_1 \tag{6}$$

Now divide by V_1 to obtain:

$$\frac{dS}{dt} = \frac{Q}{V_1}S_0 + \alpha\frac{Q}{V_1}S - (1+\alpha)\frac{Q}{V_1}S - \frac{\mu}{Y}X_1 \tag{7}$$

Again substitute the hydraulic retention time to obtain:

$$\frac{dS}{dt} = \frac{S_0}{T_1} + \alpha\frac{S}{T_1} - (1+\alpha)\frac{S}{T_1} - \frac{\mu}{Y}X_1 \tag{8}$$

Repeat the mass balance analysis for the clarifier, in the clarifier we assume there are no further reactions so the substrate concentrations that enter the clarifier leave

[2] S is the concentration of the substrate or pollutant; X is the concentration of biosolids that have been grown from the substrate.

undiminished, so we only consider the biosolids within the clarifier. The clairifier biosolids mass balance is:

$$V_2 \frac{dX_2}{dt} = (1+\alpha)QX_1 - (Q - Q_w)X_2 - (\alpha Q + Q_w)X_R \tag{9}$$

At equilibrium, $\frac{dX_1}{dt} = 0; V_2 \frac{dX_2}{dt} = 0; \frac{dS}{dt} = 0;$

Apply this condition to the three equations, first the reactor biosolids balance,

$$\frac{X_0}{T_1} + \alpha \frac{X_R}{T_1} - (1+\alpha)\frac{X_1}{T_1} + \mu X_1 = 0 \tag{10}$$

Multiply by the hydraulic retention time to obtain:

$$0 = X_0 + \alpha X_R - (1+\alpha)X_1 + \mu X_1 T_1 = 0 \tag{11}$$

Next the substrate balance,

$$\frac{S_0}{T_1} + \alpha \frac{S}{T_1} - (1+\alpha)\frac{S}{T_1} - \frac{\mu}{Y} X_1 = 0 \tag{12}$$

Again multiply by the hydraulic retention time to obtain:

$$S_0 + \alpha S - (1+\alpha)S - \frac{\mu}{Y} X_1 T_1 = 0 \tag{13}$$

Combine the terms involving S to obtain

$$X_1 T_1 = \frac{Y}{\mu}(S_0 - S) \tag{14}$$

Next the clarifier biosolids balance:

$$(1+\alpha)QX_1 - (Q - Q_w)X_2 - (\alpha Q + Q_w)X_R = 0 \tag{15}$$

Expand and divide by Q_w to obtain:

$$\frac{QX_1 + \alpha QX_1 - QX_2 + Q_w X_2 - \alpha QX_R - Q_w X_R}{Q_w} = 0 \tag{16}$$

Factor out and isolate $X_2 - X_R$ to obtain:

$$\frac{Q}{Q_w}[X_1 + \alpha X_1 - X_2 - \alpha X_R] = X_R - X_2 \tag{17}$$

Refactor to obtain

$$\frac{(1+\alpha)X_1 - X_2 - \alpha X_R}{X_R - X_2} = \frac{Q_w}{Q} \tag{18}$$

Equations 55,58 , and 62 are now employed to build a meaningful constraint set from the performance guidelines.

Multiply Equation 55 by the hydraulic retention time to obtain:

$$0 = X_0 T_1 + \alpha X_R T_1 - (1+\alpha) X_1 T_1 + \mu X_1 T_1 \cdot T_1 = 0 \tag{19}$$

Now substitute Equation 58 into the result above to obtain:

$$0 = X_0 T_1 + \alpha X_R T_1 - (1+\alpha)\frac{Y}{\mu}(S_0 - S) + Y(S_0 - S)T_1 = 0 \tag{20}$$

Now refactor the result to obtain:

$$[X_0 + \alpha X_R + Y(S_0 - S)]T_1 - \alpha\frac{Y}{\mu}(S_0 - S) = \frac{Y}{\mu}(S_0 - S) \tag{21}$$

Again, rewrite Equation 55 as

$$X_0 + \mu X_1 T_1 = (1+\alpha)X_1 - \alpha X_R \tag{22}$$

Substitute into Equation 62 to obtain:

$$\frac{X_0 + \mu X_1 T_1 - X_2}{X_R - X_2} = \frac{Q_w}{Q} \tag{23}$$

Now substitute Equation 58 to obtain:

$$\frac{X_0 + Y(S_0 - S) - X_2}{X_R - X_2} = \frac{Q_w}{Q} \tag{24}$$

Here we have removed any dependency on T_1 and α which are the design variables, so the only mass balance constraint is Equation 65.

Now consider the relationship of clarifier and reactor volumes. The design overflow rate actually sizes the clarifier – its volume must be

$$V_2 = \frac{(1+\alpha)Q}{Q_0} \cdot h \tag{25}$$

The hydraulic retention time (a design variable) provides the required size of the reactor as

$$V_1 = QT_1 \tag{26}$$

Now we have enough description to construct the optimization problem.

Problem Set-Up
Using the constraints and performance criteria we can construct the optimization problem as

$$\begin{aligned} &\textit{minimize } C \;=\; 25.12(Q)^{0.6}(T_1)^{0.6} + 25.12(\tfrac{Q\cdot h}{Q_0})^{0.6}(1+\alpha)^{0.6} \\ &\textit{subject to} \end{aligned}$$

$$\begin{array}{rcl} [X_0 + \alpha X_R + Y(S_0 - S)]T_1 - \alpha\frac{Y}{\mu}(S_0 - S) & = & \frac{Y}{\mu}(S_0 - S) \\ T_1 & \geq & 0 \\ \alpha & \geq & 0 \end{array}$$

Now for the particular problem we will substitute in numerical values from the known quantities.

$Q = 1 \times 10^6$ thus

$25.12(\frac{Q\cdot h}{Q_0})^{0.6} = 25.12(\frac{1\times 10^6 \cdot 8}{800})^{0.6} = 6309.9,$

and

$25.12(Q)^{0.6} = 25.12(1 \times 10^6)^{0.6} = 100,010.$

The remaining values are $X_0 = 0$, $X_R = 10,000$, $S_0 = 200$, and $S = 20$.

The resulting non-linear program in the design variables T_1 and α is

$$\begin{aligned} &\textit{minimize } C \;=\; 100,010T_1^{0.6} + 6309.9(1+\alpha)^{0.6} \\ &\textit{subject to} \end{aligned}$$

$$\begin{array}{rcl} 10,000\alpha \cdot T_1 + 80.82T_1 - 43.081\alpha & = & 43.081 \\ T_1 & \geq & 0 \\ \alpha & \geq & 0 \end{array}$$

7.5. Topic 5 : Extending the Toolkit – Using numerical derivatives to construct gradients and Jacobians

The approach so far involved defining objective function(s) and constraint function(s), however in both components analytical derivatives (and Jacobians) were employed.[3] For small scale problems where such analysis is straightforward, this is a fine approach – but for large scale problems, especially when the functions themselves are complicated and are not polynomials (hence the derivatives may themselves be complex even if we could find them) we may prefer to numerical derivatives – which leaves us (the engineers) only with the task of defining the objective and the constraint functions (and we hope numerical approximations of the various derivatives are well behaved).

To use numerical approximations the package `nloptr` contains a pair of functions to compute the gradient of a scalar valued function of vector arguments (the objective function) and compute the Jacobian of a vector valued function of vector arguments (the constraints) using the function names `nl.grad` and `nl.jacobian` respectively.

7.5.1. Example 2A Numerical Gradient and Jacobian

The test example from pg 209 of (**?**) is repeated, but instead of analytical derivatives, the two functions are used to construct the optimization call. Figure 38 states the problem (yes it is identical to the above problem). The fundamental change in the

$$
\begin{array}{l}
\textit{minimize} \quad x_1 \times x_2^2 \\
\textit{subject to}
\end{array}
$$

$2 - x_1^2 \times -x_2^2$	$\geq$	0
x_1	$\geq$	-5
x_2	$\geq$	-5
x_1	$\leq$	5
x_2	$\leq$	5

Figure 10. Non-Linear Program Structure for Example 2 Numerical Problem.

script will the the addition of the two functions named `objfn` and `constfn1` that accept as input the current value of the solution vector and return the various functional results. The function names are arbitrary, but i selected them to be meaningful (to me at least) and not to interfere with names used by the package. Listing 22 shows the code for the two new functions.

Listing 3. R code for functions `objfn` and `constfn1`.

```
# Define the objective function
#
```

[3]The various components are analytical because we used calculus to find the gradient of the objective and the directional derivatives of the constraints.

```
# f(x) = x1*(x2^2)
#
objfn <- function(x) {
  objfn <- x[1]*x[2]^2;
  return(objfn)
}
# Define the constraint function
#   x1^2 + x2^2 - 2 <= 0
#
constfn1 <- function(x){
  constfn1 <- x[1]^2 + x[2]^2 -2 ;
  return(constfn1)
}
```

These two functions are then substituted into the optimization interface functions and their respective derivatives[4] as shown in Listing 23

Listing 4. R code for functions `objfn` and `constfn1`.

```
# Construct the evaluation structure for nloptr
#
eval_f <- function( x ) {
  return( list( "objective" = objfn(x),
                "gradient" = nl.grad(x,objfn). # << Here we use the numerical method for
                    gradient
              )
          )
}
# Construct the constraint functions
# Inequalities
eval_g_ineq <- function( x ) {
  constr <- constfn1(x);
  grad <- nl.jacobian(x,constfn1) # << Here we use the numerical method for Jacobian
  return( list( "constraints"=constr, "jacobian"=grad ) )
}
```

The entire problem structure for the **R** script is shown in Listing 24 and a complete solution is displayed in Figure 39.

Listing 5. R code for Example 2A – Numerical Derivatives.

```
# Example problem, pg 209 Gill, Murray, and Wright
#
# min_{x} x1*(x2^2)
# s.t.
#   x1^2 + x2^2 - 2 <= 0
#
# x0 = (-1,-1)
#
# optimal solution = (-0.81650,-1.1547)
#
# Numerical approximations to compute the gradient of the objective function
# and the jacobian of the constraint set via the nloptr methods: nl.grad and nl.jacobian

library('nloptr')
# Define the objective function
# f(x) = x1*(x2^2)
#
objfn <- function(x) {
  objfn <- x[1]*x[2]^2;
  return(objfn)
}
# Define the constraint function
#   x1^2 + x2^2 - 2 <= 0
#
constfn1 <- function(x){
  constfn1 <- x[1]^2 + x[2]^2 -2 ;
  return(constfn1)
}
# Construct the evaluation structure for nloptr
#
eval_f <- function( x ) {
  return( list( "objective" = objfn(x),
                "gradient" = nl.grad(x,objfn)
              )
          )
}
# Construct the constraint functions
```

[4]The gradient in the case of the objective function using `nl.grad`, and the Jacobian in the case of the constraint function using `nl.jacobian`

```
# Inequalities
eval_g_ineq <- function( x ) {
  constr <- constfn1(x);
  grad <- nl.jacobian(x,constfn1)
  return( list( "constraints"=constr, "jacobian"=grad ) )
}
# Set initial values
x0 <- c( -1,-1 )
# Set lower and upper bounds of control
lb <- c( -5, -5 )
ub <- c( 5, 5 )
# Begin the optimization call(s)
local_opts <- list( "algorithm" = "NLOPT_LD_MMA",
                    "xtol_rel"  = 1.0e-7 )
opts <- list( "algorithm" = "NLOPT_LD_AUGLAG",
              "xtol_rel"  = 1.0e-7,
              "maxeval"   = 1000,
              "local_opts" = local_opts )
res <- nloptr( x0=x0,
               eval_f=eval_f,
               lb=lb,
               ub=ub,
               eval_g_ineq=eval_g_ineq,
# Activate next line if there are equality constraints
#               eval_g_eq=eval_g_eq,
              opts=opts)
print( res )
```

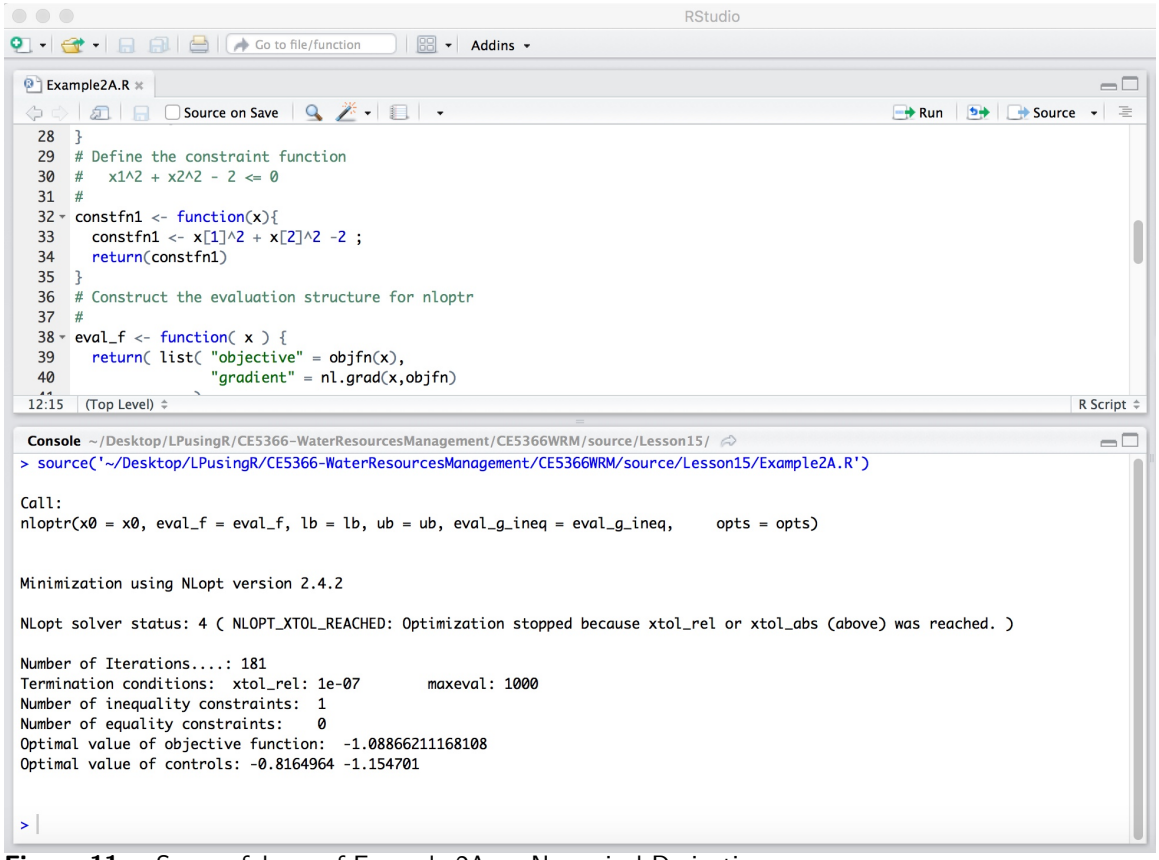


Figure 11. Successful run of Example 2A — Numerical Derivatives.

7.6. Topic 5: Water Resources Allocation/Operation Problem

7.6.1. Example

7.7. Topic 6: Water Quality Allocation Problem

7.7.1. Example

7.8. Readings

7.9. Exercises

1. Solve the following constrained non-linear optimization problem.

   $$\begin{array}{l} \textit{minimize} \quad (x_1 - x_2)^2 + (x_2 - x_3)^4 \\ \textit{subject to} \end{array}$$

$x_1 + x_1(x_2)^2 + (x_3)^4$			$=$	3
x_1			$\geq$	-5
	x_2		$\geq$	-5
		x_3	$\geq$	-5
x_1			$\leq$	5
	x_2		$\leq$	5
		x_3	$\leq$	5

   A solution should exist at $x^* = (1, 1, 1)$.

2. Solve the following constrained non-linear optimization problem.

   $$\begin{array}{l} \textit{minimize} \quad (x_1)^3 - 6(x_1)^2 + 11x_1 + x_3 \\ \textit{subject to} \end{array}$$

$(x_1)^2 + (x_2)^2 - (x_3)^2$			$\leq$	0
$(x_1)^2 + (x_2)^2 + (x_3)^2$			$\geq$	4
		$(x_3)^2$	$\leq$	5
x_1			$\geq$	0
	x_2		$\geq$	0
		x_3	$\geq$	0
x_1			$\leq$	$+Inf$
	x_2		$\leq$	$+Inf$
		x_3	$\leq$	$+Inf$

   A solution should exist at $x^* = (0, \sqrt{(}2), \sqrt{(}2))$. Of special note, the objective function does not contain x_2; its value is strictly established by the constraint requirements.

3. Water Resources Allocation Problem 1 (Next time)

4. Water Quality Allocation Problem 2 (Next time)

8. Simulation Tools - A Decision Consequence Estimation Component

The purpose of a simulation in the context of this document is to predict the response of a system to the decisions; that is what is the anticipated state of the system after the decisions have been applied. Did the decisions improve the situation, or make it worse, or no detectable change.

8.1. Simulation and Models

define a model

8.1.1. Example

8.2. Statistical Models

8.2.1. Example

8.3. Analog Models

Analog models are models where analogs of the real system to be studied are employed. Column, Jar, and bench-scale studies in Environmental Engineering are examples of analog (and physical) models of a real system. Flume and estuary scale physical models are examples of analog (and scaled) models of a real system. Electric circuit analogs have been built to simulate behavior of river systems and groundwater aquifers – actual instances of these kinds of models are today only likely to be found in museums; computer models have largely replaced these kind of analog models.

In general analog models are outside the scope of this document, they are useful, they are expensive, and they are uncommon (with the exception of the Environmental Engineering laboratory models).

8.3.1. Example

8.4. Digital Models

8.4.1. Example

Use an on-line network simulator to make a decision.

8.5. Readings

8.6. Exercises

9. Simulation Models – Riverine Systems

words

9.1. QUAL2E On-Line

QUAL2E On-Line is a web accessible implementation of QUAL2E that performs the model computations on a remote server and when completed, the submitter can download and process the results. Uploading model input files and a supervisory file requires server credentials, but otherwise the same input and output files of the circa 1995 version of QUAL2E is implemented.

QUAL2E, is a one-dimensional river and stream water quality model that simulates temperature, dissolved oxygen nutrients, pH, periphyton, macrophytes, phytoplankton, and sediment diagenesis. QUAL2E simulates a steady hydraulics representation of stream responses with diurnal variability in boundary conditions.

A newer model,Version 6 of QUAL2Kw allows continuous simulation with non-steady, non-uniform flow using kinematic wave flow routing, however the source code is not publicly available. Water quality simulations include nutrient dynamics, algal production, dissolved oxygen with the impact of benthic and carbonaceous demand, pH, and alkalinity.

9.1.1. Example

9.2. Interpreting Results

9.2.1. Example

9.3. Readings

9.4. Exercises

10. Simulation Models – Aquifer Systems

10.1. USGS-MOC On-Line

This chapter demonstrates pumping allocation based on total demands for consumption and availability of water for the certain aquifer, and an example of a groundwater simulation-optimization activity. The optimization model is created using linear programming for aquifer management to minimize the cost of the pumping and conveyance to a customer, to meet a water demand requirement, with constraints designed to prevent contaminated lake water from encroaching into the aquifer. The chapter is presented as a case study for a simple example to illustrate the organizational and computational steps involved.

Figure 12 shows a plan view of the aquifer system being examined. The 10 km 10 km square aquifer is divided into 25 cells of (2km 2km each). Well fields in each cell are assumed to behave as if all pumping occurs at the cell centroid, and the resulting heads are averaged throughout the cell.

The aquifer has impervious boundaries on three sides and a lake is on the forth side. The lake water quality is poor, so keeping the water level of the aquifer above a certain level should be taken into consideration to prevent water encroaching from the lake into the aquifer. Therefore, water level at the distance of 1 km and 3 km from the lake should be maintained at +0.64 m and +0.95 m respectively, where the lake level is the datum (+0.0 m).

Precipitation with a rate of N=100 mm/yr is replenishing the aquifer uniformly over the entire area. The entire replenishment drains through the aquifer into the lake. The aquifer is a homogeneous-isotropic aquifer and has a transmissivity of T=1000 m^2/day.

A single consumer (city) exists in Cell 18, with a demand of $7.0 Mm^3/yr$. The cost to pump water and deliver to Cell 18 is $(\$1.00 + \$0.50/km))/Mm^3$

The distance from a pumping cell to cell 18 expressed in cartesian coordinates is $D_{i->18} = \sqrt{(x_i - x_{18})^2 + (y_i - y_{18})^2}$ where the x and y values are the coordinates to the cell centers in kilometers (using the lower right corner is the coordinate system origin).

For example the cost to pump 1 Mm^3 of water from Cell 18 is $1.00, whereas to cost to pump 1 Mm^3 water from Cell 3 is $1.00 + $3.00 = $4.00. The first $1.00 is the pumping cost and the remainder is the transport cost to deliver water to Cell 18.

The management objective is to supply the demand at Cell 18 at minimum cost, while maintaining the minimal water levels in Cells 16 – 25 to protect the water quality of the aquifer (and prevent lake water from entering the aquifer).

To construct the management model we can first specify the decision variables which are P_i the pumping rate inMm^3 in the $i - th$ cell. Then we need to specify the cost

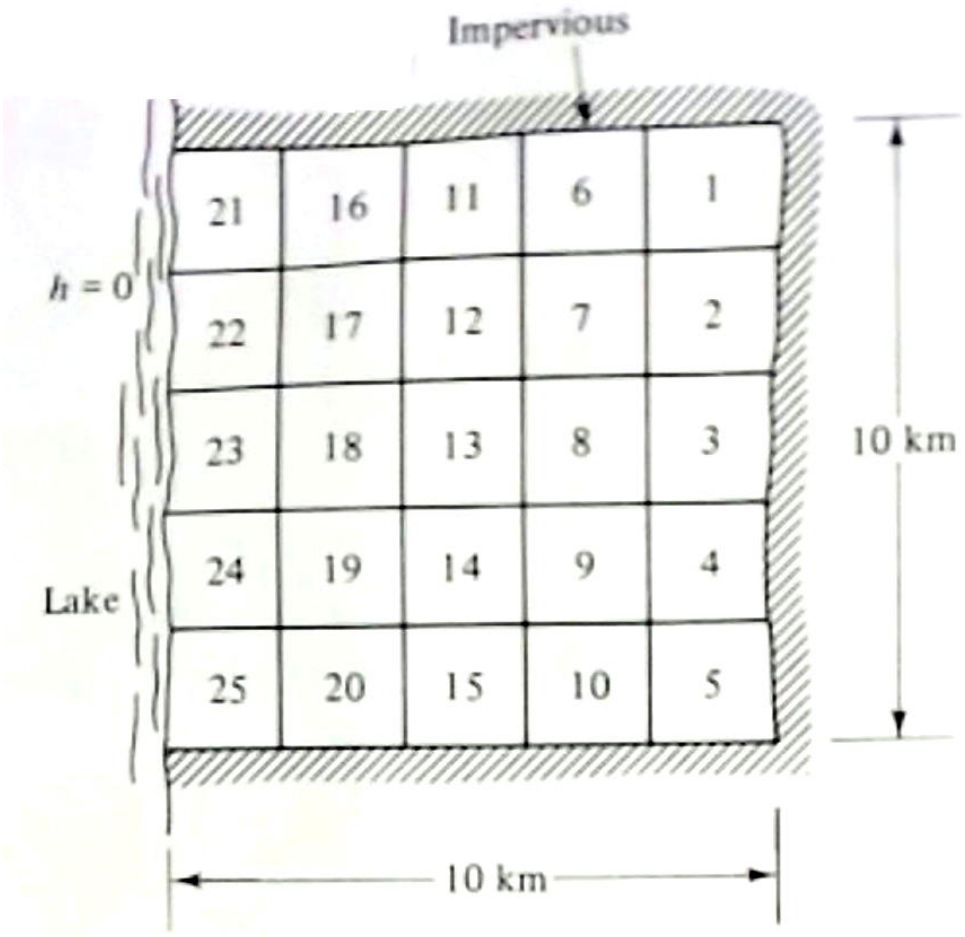


Figure 12. 25-cell Aquifer System Schematic.

function

$$C = \sum_{i=1}^{25} (1 + \sqrt{(x_i - x_{18})^2 + (y_i - y_{18})^2}) \times P_{i_{i=1}}^{25} \tag{27}$$

Next is the influence on the aquifer heads (or drawdown) at location i caused by unit pumping at location j. These will form the various constraint equations. There are 25 cells in the aquifer that can have pumping, and 10 cells with restrictive constraints (the drawdown requirements).

10.2. Groundwater Hydraulics Model to Build the Influence Coefficients

The pumping decisions affect the aquifer head, so a tool to estimate the response to the decision variables is created using a computer model of the aquifer. In the example, the system is to be operated at steady conditions and the aquifer is a homogeneous-isotropic confined aquifer, so the effect of one unit of pumping at a location can be determined external to the optimization process. First, one needs to

develop a groundwater hydraulic model of the system.

Confined Aquifer Flow

We will address approximating heads in a confined aquifer, first in one spatial dimension, then in two spatial dimensions. The extension to a third spatial dimension is relatively straightforward if one can picture each 2D horizontal slice as a layer.

10.2.1. Finite-Difference Methods – 2 Spatial Dimensions

If we perform an analysis in the same way as we did to arrive at Equation **??** except now include another direction (the y-direction) we will have an aquifer in two spatial dimensions. The governing equation becomes

$$S\frac{\partial h}{\partial t} = \frac{\partial}{\partial x}(T_x\frac{\partial h}{\partial x}) + \frac{\partial}{\partial y}(T_y\frac{\partial h}{\partial y}) \tag{28}$$

The meanings of the terms are the same, except the transmissivity terms now have subscripts to indicate they can have different values depending on direction.

Then as before we will construct the difference-equation model from a multiple-cell balance model of the aquifer at a cell of interest, then extend the equations to cover the entire model domain.

Figure 13 is a plan view schematic of a aquifer with flow to be computed in two directions (x and y). The cell indexing convention in the sketch is that rows are indexed by the letter j and columns are indexed by the letter i. This naming convention is arbitrary; in some instances it may be preferable to reverse the convention. The schematic also shows the assumed flow directions; for column i, flows are upward in the drawing, and for row j, flows are from left to right. If indeed the opposite is true for a given set of boundary conditions and material properties, then the flows will be computed as negative numbers – hence the convention here is that "positive flow" is right and up.

As in the one-dimensional development the storage term is

$$\frac{dM_{water}}{dt}\Big|_{cell} = \rho_w S_s \Delta x \Delta y \Delta z \frac{\partial h_i}{\partial t} \tag{29}$$

where h_i is the head in the $i-$th cell.

The mass flows entering the $i-\text{th}, j-\text{th}$ cell are:

$$M_{Inflow} = Q_{left} + Q_{bottom} = \rho_w K_x \Delta y \Delta z \frac{h_{i-1,j} - h_{i,j}}{\Delta x} + \rho_w K_y \Delta x \Delta z \frac{h_{i,j-1} - h_{i,j}}{\Delta y} \tag{30}$$

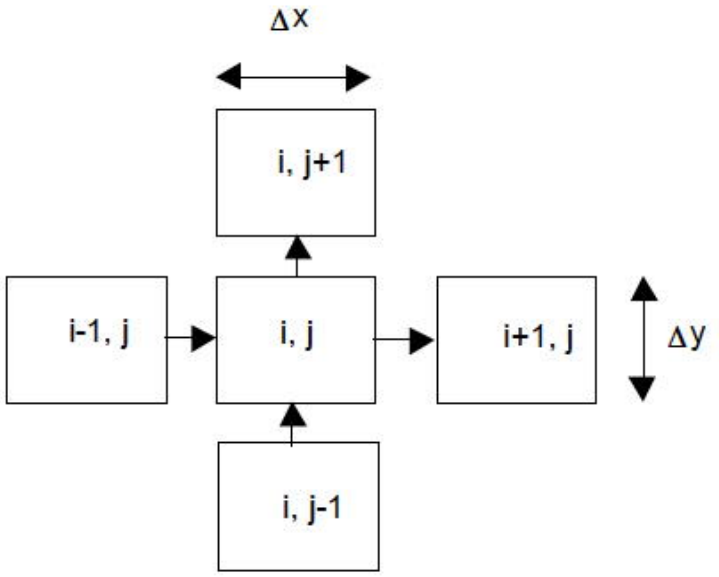


Figure 13. Plan view schematic of 2-dimensional multiple cell balance computational stencil.

The mass flows leaving the $i-\text{th}, j-\text{th}$ cell are:

$$M_{Inflow} = Q_{right} + Q_{top} = \rho_w K_x \Delta y \Delta z \frac{h_{i,j} - h_{i+1,j}}{\Delta x} + \rho_w K_y \Delta x \Delta z \frac{h_{i,j} - h_{i,j+1}}{\Delta y} \quad (31)$$

Now write the entire balance equation

$$\begin{aligned} \rho_w S_s \Delta x \Delta y \Delta z \frac{\partial h_i}{\partial t} = [\rho_w K_x \Delta y \Delta z \tfrac{h_{i-1,j} - h_{i,j}}{\Delta x} + \rho_w K_y \Delta x \Delta z \tfrac{h_{i,j-1} - h_{i,j}}{\Delta y}] - \\ [\rho_w K_x \Delta y \Delta z \tfrac{h_{i,j} - h_{i+1,j}}{\Delta x} + \rho_w K_y \Delta x \Delta z \tfrac{h_{i,j} - h_{i,j+1}}{\Delta y}] \end{aligned} \quad (32)$$

Next replace $S_s \Delta z$ with the storage coefficient S, and the $K_{x,y} \Delta z$ with the transmissivity $T_{x,y}$ terms, and divide by the density of the fluid and the cell plan view area $\Delta x \Delta y$ to obtain a more compact form of the difference equation.

$$\begin{aligned} S \frac{\partial h_i}{\partial t} = [\tfrac{1}{\Delta x} T_x \tfrac{h_{i-1,j} - h_{i,j}}{\Delta x} + \tfrac{1}{\Delta y} T_y \tfrac{h_{i,j-1} - h_{i,j}}{\Delta y}] - \\ [\tfrac{1}{\Delta x} T_x \tfrac{h_{i,j} - h_{i+1,j}}{\Delta x} + \tfrac{1}{\Delta y} T_y \tfrac{h_{i,j} - h_{i,j+1}}{\Delta y}] \end{aligned} \quad (33)$$

As in the one-dimensional case, lets again consider steady flow (we will do transient flows later on)

$$\begin{aligned} 0 = [\tfrac{1}{\Delta x} T_x \tfrac{h_{i-1,j} - h_{i,j}}{\Delta x} + \tfrac{1}{\Delta y} T_y \tfrac{h_{i,j-1} - h_{i,j}}{\Delta y}] - \\ [\tfrac{1}{\Delta x} T_x \tfrac{h_{i,j} - h_{i+1,j}}{\Delta x} + \tfrac{1}{\Delta y} T_y \tfrac{h_{i,j} - h_{i,j+1}}{\Delta y}] \end{aligned} \quad (34)$$

Also as in the one-dimensional case, we will approximate the spatial variation of the material properties (transmissivity) as arithmetic mean values between two cells, so making the following definitions:

$$\begin{aligned} A_{i,j} &= \tfrac{1}{2\Delta x^2}(T_{x,(i-1,j)} + T_{x,(i,j)}) \\ B_{i,j} &= \tfrac{1}{2\Delta x^2}(T_{x,(i,j)} + T_{x,(i+1,j)}) \\ C_{i,j} &= \tfrac{1}{2\Delta y^2}(T_{y,(i,j-1)} + T_{y,(i,j)}) \\ D_{i,j} &= \tfrac{1}{2\Delta y^2}(T_{y,(i,j)} + T_{y,(i,j+1)}) \end{aligned} \tag{35}$$

Substitution into the difference equation yields

$$0 = A_{i,j}h_{i-1,j} + B_{i,j}h_{i+1,j} - (A_{i,j} + B_{i,j} + C_{i,j} + D_{i,j})h_{i,j} + C_{i,j}h_{i,j-1} + D_{i,j}h_{i,j+1} \tag{36}$$

As before we can explicitly write the cell equation for $h_{i,j}$ as

$$h_{i,j} = \frac{[A_{i,j}h_{i-1,j} + B_{i,j}h_{i+1,j} + C_{i,j}h_{i,j-1} + D_{i,j}h_{i,j+1}]}{[A_{i,j} + B_{i,j} + C_{i,j} + D_{i,j}]} \tag{37}$$

This difference equation represents an approximation to the governing flow equation, the accuracy depending on the cell size. Boundary conditions are applied directly into the analogs (another name for the difference equations) at appropriate locations on the computational grid. Also as in the one-dimensional case we can generate solutions either by iteration or solving the resulting linear system.

10.2.2. Generalizing the Boundary Conditions

In the prior examples the boundary conditions for the problems were kind of glossed over. We applied a fixed head boundary on the left and right edges of the rectangular domain, and zero-flux boundary at the top and bottom edges. A useful improvement is to allow the user to choose which type by supplying information in the input file. I find the easiest way (as we are just learning) is to assume the entire model is always surrounded by a constant head condition and use a mask to tell the program when that is not true.

The code fragments for making this change are pretty straightforward, and are displayed in Listing 6. We need to read in boundary indicators for the top, bottom, left, and right boundaries. Then convert them into numeric values for later. Here I choose to use a zero to indicate a zero-flux boundary and any non-zero (usually a 1) to indicate a fixed head boundary.

Next we apply the conditions within the solver loop by assigning the compued value of the adjacent cell (either above, below, left, or right as appropriate). The reaminder of the script is unchanged, of coude we need to include the new input values in the data file as in Listing 7.

Listing 6. Script fragments for implementing generalized boundary conditions.

```
### Up in the input read section of the script ########
# add boundary conditions 0= fixed head, 1= no flow
boundarytop <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
     FALSE))
boundarybottom <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
boundaryleft <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul
    = FALSE))
boundaryright <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul
     = FALSE))
hydhead <-(readLines(zz, n = nrows, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
    FALSE))
.........
### Convert to numeric ######
boundarytop <-as.numeric(unlist(strsplit(boundarytop,split=" ")))
boundarybottom <-as.numeric(unlist(strsplit(boundarybottom,split=" ")))
boundaryleft <-as.numeric(unlist(strsplit(boundaryleft,split=" ")))
boundaryright <-as.numeric(unlist(strsplit(boundaryright,split=" ")))
..........
#### Apply the conditions ####
for (iter in 1:maxiter){
# Boundary Conditions
# Top and Bottom
for(jcol in 1:ncols){
    if(boundarytop[jcol] == 0){hydhead[1,jcol]<-hydhead[2,jcol]} #no-flow at top
    if(boundarybottom[jcol] == 0){hydhead[nrows,jcol]<-hydhead[2,jcol]} #no-flow at bottom
    # otherwise values are fixed head
}
for(irow in 1:nrows){
    if(boundaryleft[irow] == 0){hydhead[irow,1]<-hydhead[irow,2]} #no-flow at left
    if(boundaryright[irow] == 0){hydhead[irow,ncols]<-hydhead[irow,ncols-1]} #no-flow at
        right
    # otherwise values are fixed head
}
  for (irow in 2:(nrows-1)){
  ..........
```

Listing 7. Input file for 2D vertical slice confined aquifer with low permeability inclusion using a generalized boundary mask. The comments in the listing should be removed for running the program.

```
1
10
1
13
23
1e-8
5000
5 15 25 35 45 55 65 75 85 95 105 115 125 135 145 155 165 175 185 195 205 215 225
0.5 1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5 9.5 10.5 11.5 12.5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  # Boundary Conditions along Top
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  # Boundary Conditions along Bottom
1 1 1 1 1 1 1 1 1 1 1 1 1                      # Boundary Conditions along Left
1 1 1 1 1 1 1 1 1 1 1 1 1                      # Boundary Conditions along Right
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 0.001 0.001 0.001 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 0.001 0.001 0.001 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 0.001 0.001 0.001 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
```

```
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 0.001 0.001 0.001 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 0.001 0.001 0.001 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 0.001 0.001 0.001 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
```

Addition of the generalized boundary conditions expands the utility of our script as illustrated in the next example.

Example 4: 2D Vertical Slice in for Computing Heads (Pore Pressure) under a Dam

Figure 14 is a schematic of a dam built upon a permeable ground layer 80 meters thick (segment A to I). The dam has a base 60 meters wide (segment B to F), with an upstream water depth of 10 meters. The downstream side of the dam is at 0 meters depth (otherwise its not a very good dam!). A sheetpile cutoff wall is installed beneath the dam (segment C to D to U to E). The ground layer has a hydraulic conductivity of $K = 1 \times 10^{-4}$ meters per second.

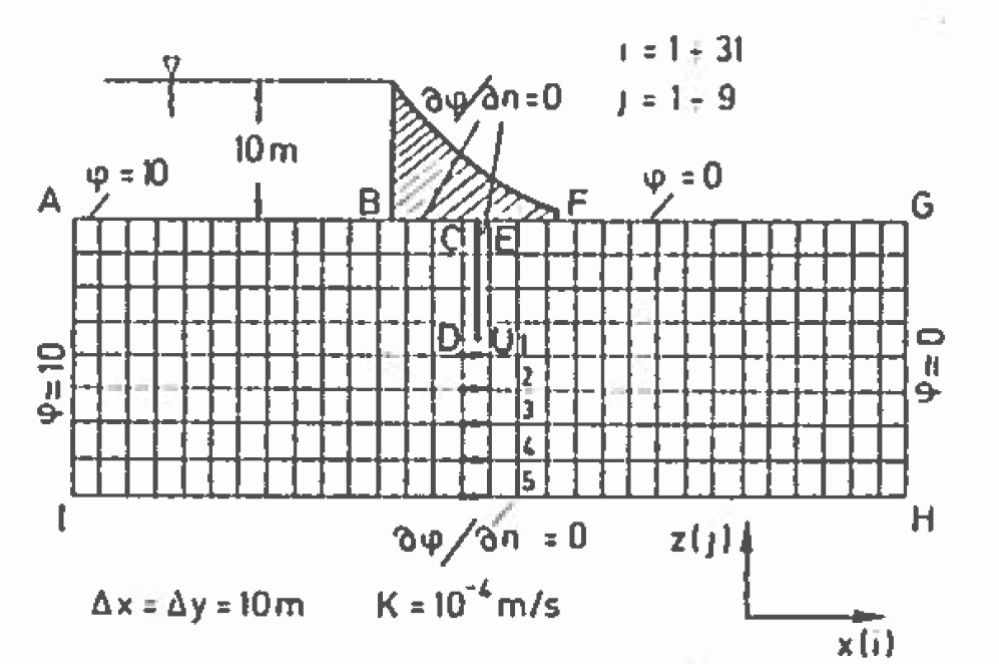


Figure 14. Schematic of a dam on permeable soil with the sheet pile curtain underneath the dam.

Important engineering questions are what is the pore water pressure under the dam, and is what is the total seepage under the dam? The pore water pressure can be found by solving for heads under the dam then subtracting the elevation of the computation location elative to a datum. The flow is found by Darcy's law applied under the dam

(shown as locations 1,2,3,4, and 5 in the sketch), which in turn requires computation of head under the dam. Thus the questions are answered by finding the head distribution under the dam.

The flow field (mathematically) extends an infinite distance upstream and downstream, but as a practical matter the contribution to seepage far upstream of the dam is negligible, and hence is approximated by the finite domain depicted.

Using the tools we have already built we can simply build an input file, run our script and determine the head distribution (and thus compute the discharges under the dam. There are two ways to conceptualize the model domain, we will examine both.

The first is to represent the domain as shown, and make the following specifications in the boundary condition information, and we will treat the sheetpile cutoff wall as a low permeability inclusion (much like the the prior example). The boundary conditions are:

1. The segment from A to B is a constant head boundary with value equal to 10.
2. The segment from B to F is a zero-flux boundary.
3. The segment from B to C to D to U to E to F should be treated as a zero-flux boundary, but our mask does not extend into the interior – however the sheetpile itself can be approximated by providing a very small permeability. Alternately we could (should) modify the code to handle interior boundaries – but that is outside the scope of this chapter.
4. The segment from F to G is a constant head boundary with value equal to 0.
5. The segment form G to H is a constant head boundary with value equal to 0.
6. The segment from H to I is a zero-flux boundary.
7. The segment from I to A is a constant head boundary with value equal to 10.

Listing 8 is the input file that is used that incorporates the boundary conditions stated above. Observe how all we did was change the Δx and Δy specifications, modified the boundary condition arrays, and set the row and column counts.

Listing 8. Input file for 2D vertical slice for Dam Seepage Example.

```
10
10
1
9
31
1e-12
5000
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240
     250 260 270 280 290 300
0 10 20 30 40 50 60 70 80
1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
```

```
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0000001 0.0000001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0000001 0.0000001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0000001 0.0000001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0000001 0.0000001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0000001 0.0000001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.00010
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
```

Figure 15 is a screen capture of the code running and producing the contour plot. I was unable to force the plot to plot the zero head contour, however the problem is symmetric, so this line should be the mirror image of the 10 meter contour on the left of the image.

The contour plot itself (even with these cautions) is a little deceptive, so I added a write command at the end of the script to write the contents of the hydraulic head array to a file that can be loaded into Excel to make the requisite flow calculations.

A screen capture of the output data loaded into an Excel spreadsheet is shown on Figure 16. The red shaded area in the figure is where the sheetpile cutoff wall is located.

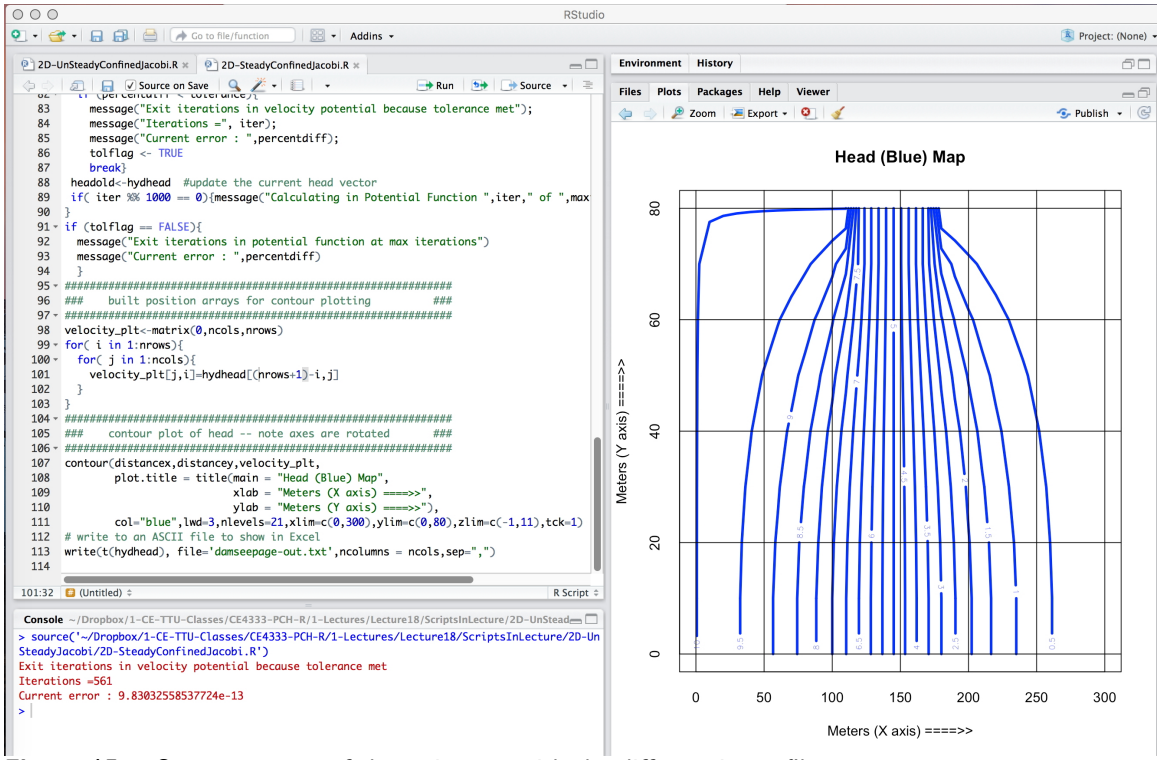


Figure 15. Screen capture of the script run with the different input file.

If we apply Darcy's law to the orange portion in the vertical direction (where flow mush be vertical to seep under the dam) the result is about 3.76 m^3 per day, per meter of width (perpendicular to Figure Figure 14). These computations are shown in the spreadsheet beneath the output data.

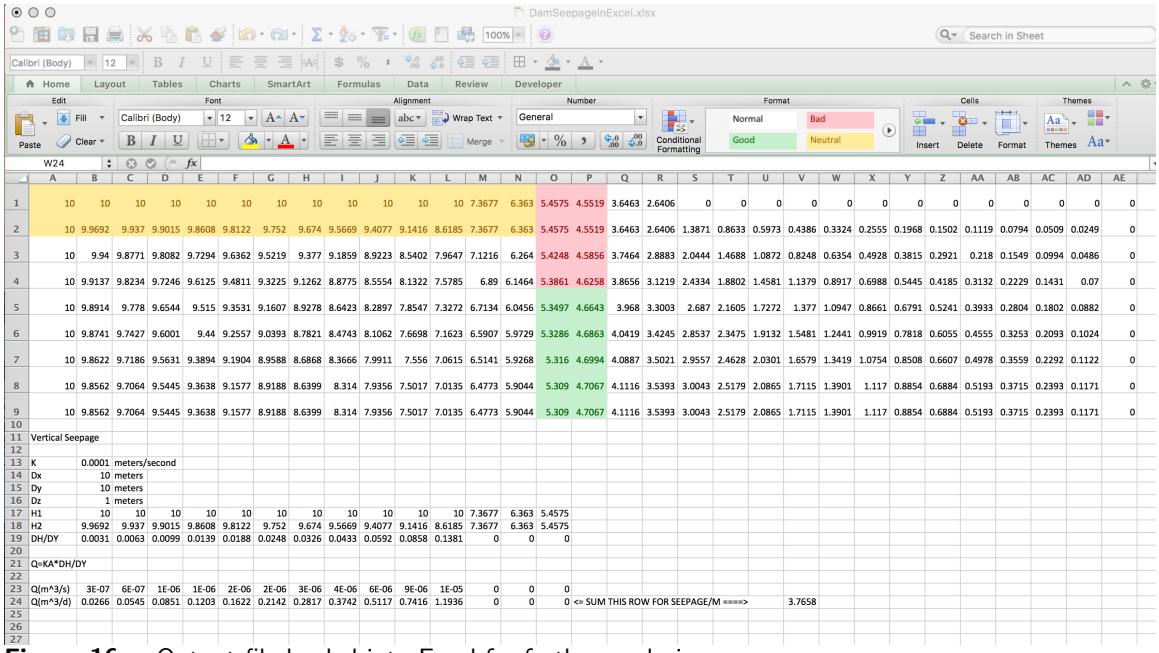


Figure 16. Output file loaded into Excel for further analysis.

This example (like most roll-your-own) requires some specific knowledge of hydraulics to interpret the results, in this case knowing to apply Darcy's law along the bottom of the reservoir to determine the flow rate, and to integrate (sum up) the individual cell flows to obtain total flow.

The second way to solve this example, and perhaps better is to take advantage of the symmetry and cut the domain in half, as in Figure 17. In this method we can actually specify the sheetpile as a boundary, and we will obtain the same results, but only need to supply half as much input data.

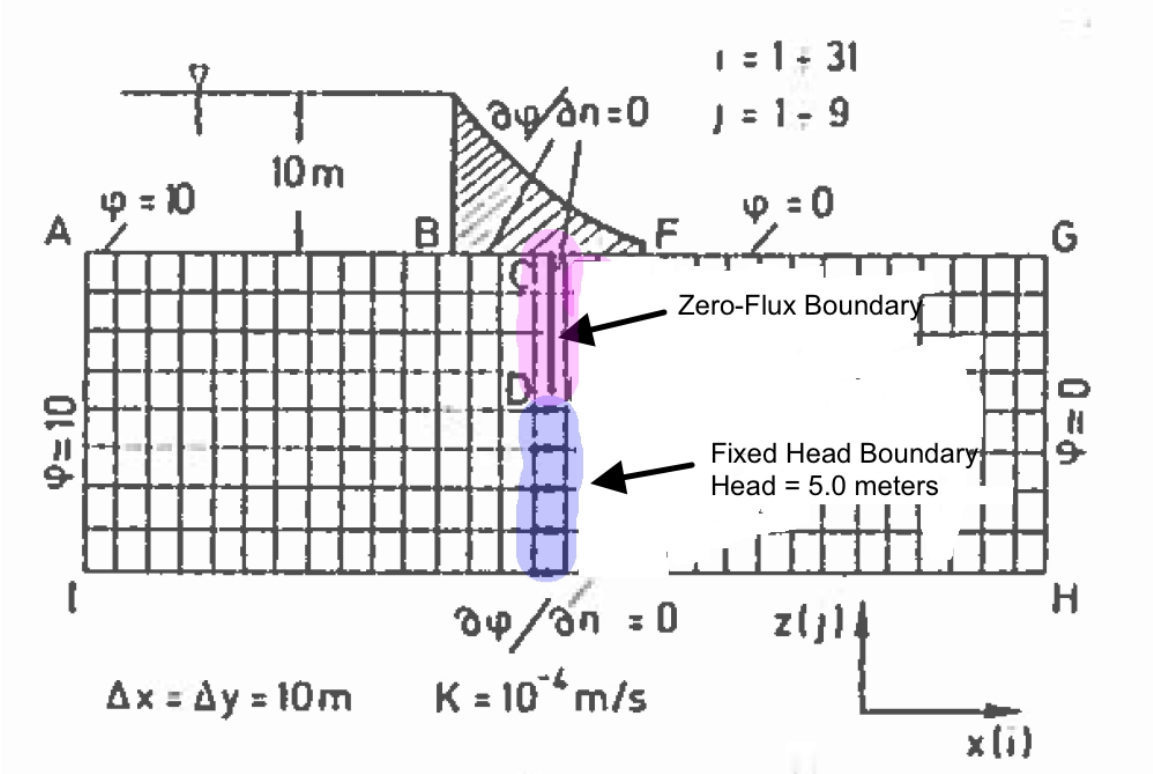


Figure 17. Output file loaded into Excel for further analysis.

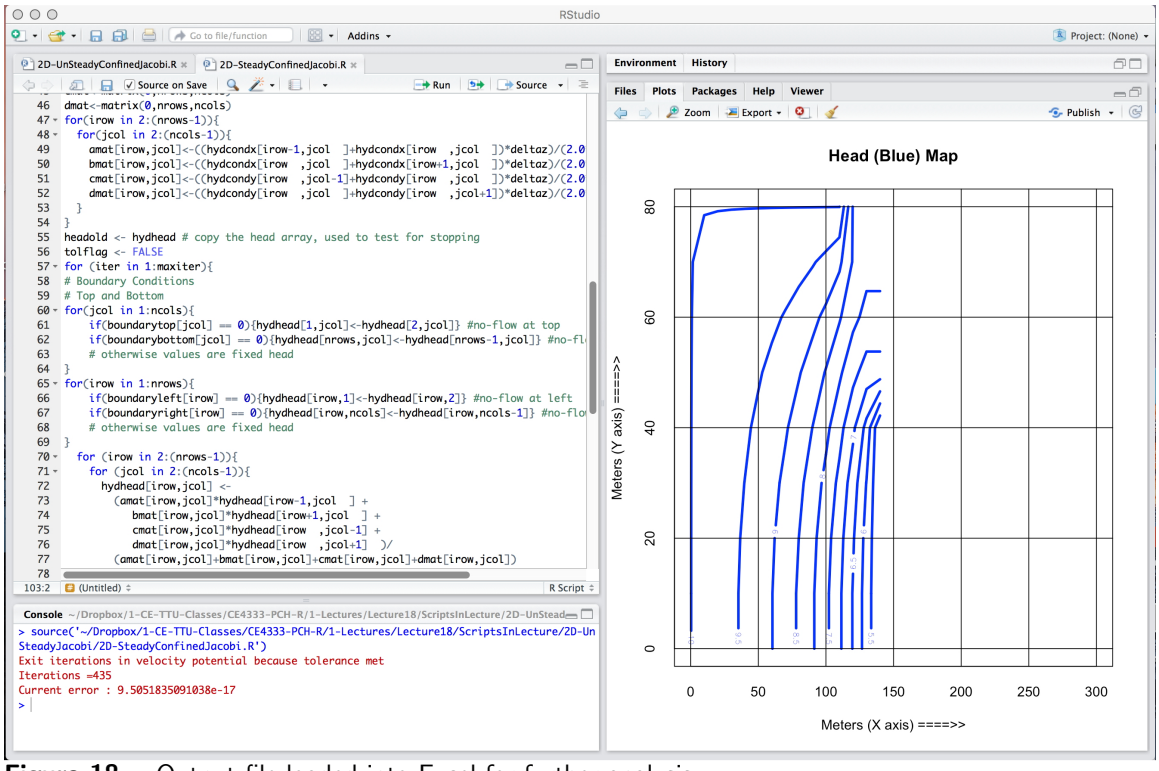
Listing 9 is the input file for this symmetry based approach.

Listing 9. Input file for 2D vertical slice for Dam Seepage Example using Symmetry.

```
10
10
1
9
15
1e-16
5000
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140
0 10 20 30 40 50 60 70 80
1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1
0 0 0 0 1 1 1 1 1
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 5
10 10 10 10 10 10 10 10 10 10 10 10 10 10 5
10 10 10 10 10 10 10 10 10 10 10 10 10 10 5
10 10 10 10 10 10 10 10 10 10 10 10 10 10 5
```

```
 10  10  10  10  10  10  10  10  10  10  10  10  10  10  5
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
 0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001  0.0001
      0.0001  0.0001
```

Figure 18 is the result of using the half-domain approach.



```
dmat<-matrix(0,nrows,ncols)
for(irow in 2:(nrows-1)){
  for(jcol in 2:(ncols-1)){
    amat[irow,jcol]<-((hydcondx[irow-1,jcol  ]+hydcondx[irow  ,jcol  ])*deltaz)/(2.0
    bmat[irow,jcol]<-((hydcondx[irow  ,jcol  ]+hydcondx[irow+1,jcol  ])*deltaz)/(2.0
    cmat[irow,jcol]<-((hydcondy[irow  ,jcol-1]+hydcondy[irow  ,jcol  ])*deltaz)/(2.0
    dmat[irow,jcol]<-((hydcondy[irow  ,jcol  ]+hydcondy[irow  ,jcol+1])*deltaz)/(2.0
  }
}
headold <- hydhead # copy the head array, used to test for stopping
tolflag <- FALSE
for (iter in 1:maxiter){
# Boundary Conditions
# Top and Bottom
for(jcol in 1:ncols){
    if(boundarytop[jcol] == 0){hydhead[1,jcol]<-hydhead[2,jcol]} #no-flow at top
    if(boundarybottom[jcol] == 0){hydhead[nrows,jcol]<-hydhead[nrows-1,jcol]} #no-fl
    # otherwise values are fixed head
}
for(irow in 1:nrows){
    if(boundaryleft[irow] == 0){hydhead[irow,1]<-hydhead[irow,2]} #no-flow at left
    if(boundaryright[irow] == 0){hydhead[irow,ncols]<-hydhead[irow,ncols-1]} #no-flo
    # otherwise values are fixed head
}
  for (irow in 2:(nrows-1)){
    for (jcol in 2:(ncols-1)){
      hydhead[irow,jcol] <-
        (amat[irow,jcol]*hydhead[irow-1,jcol  ] +
           bmat[irow,jcol]*hydhead[irow+1,jcol  ] +
           cmat[irow,jcol]*hydhead[irow  ,jcol-1] +
           dmat[irow,jcol]*hydhead[irow  ,jcol+1]  )/
        (amat[irow,jcol]+bmat[irow,jcol]+cmat[irow,jcol]+dmat[irow,jcol])
```

Console ~/Dropbox/1-CE-TTU-Classes/CE4333-PCH-R/1-Lectures/Lecture18/ScriptsInLecture/2D-UnStead

```
> source('~/Dropbox/1-CE-TTU-Classes/CE4333-PCH-R/1-Lectures/Lecture18/ScriptsInLecture/2D-Un
SteadyJacobi/2D-SteadyConfinedJacobi.R')
Exit iterations in velocity potential because tolerance met
Iterations =435
Current error : 9.5051835091038e-17
> 
```

Figure 18. Output file loaded into Excel for further analysis.

If we repeat the flow calculations, this time along the portion at the bottom of the

sheetpile, we obtain nearly the same result, 3.74 m^3 per day, per meter of width. This second way of solving the problem is more correct because we didn't have to choose an artificial value to mimic the effect of the sheetpile.

10.2.3. Internal sources/sinks to handle a discharge (well) at a specific location

The last concept we will add here is to illustrate a means to approximate the effect of a localized source (recharge or an injection well) or sink (pumping well). To incorporate these kinds of inputs we add the terms to Equation 28 to obtain Equation 38

$$S\frac{\partial h}{\partial t} = \frac{\partial}{\partial x}(T_x\frac{\partial h}{\partial x}) + \frac{\partial}{\partial y}(T_y\frac{\partial h}{\partial y}) + R - Q \tag{38}$$

where R and Q are recharge and pumping expressed in dimensions of $\frac{L^3}{T}$.

The resulting difference equation is

$$\begin{aligned} 0 = [\tfrac{1}{\Delta x}T_x\tfrac{h_{i-1,j}-h_{i,j}}{\Delta x} + \tfrac{1}{\Delta y}T_y\tfrac{h_{i,j-1}-h_{i,j}}{\Delta y}]- \\ [\tfrac{1}{\Delta x}T_x\tfrac{h_{i,j}-h_{i+1,j}}{\Delta x} + \tfrac{1}{\Delta y}T_y\tfrac{h_{i,j}-h_{i,j+1}}{\Delta y}] \\ +\tfrac{R_{i,j}}{\Delta x\Delta y} - \tfrac{Q_{i,j}}{\Delta x\Delta y} \end{aligned} \tag{39}$$

These additions are incorporated into our program by only adding a few code fragments in certain places. Listing 10 shows the various added components to handle a well field. Observe we can incorporate recharge as if it were a well with a negative pumping rate so for this document it is not considered separately, although in many practical cases that might be a preferable way to incorporate the process.

Listing 10. Input file for 2D Steady flow with generalized boundary conditions and wells.

```
# 2D Steady Confined -- With Boundary Arrays -- And Pumping Array
# 2D Aquifer Flow Model using Jacobi Iteration
# deallocate memory
rm(list=ls())
zz <- file("wellfield.dat", "r") # Open a connection named zz to file named input.dat
# read the simulation conditons
deltax <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
deltay <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
deltaz <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
nrows <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
ncols <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
tolerance <- as.numeric(readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
maxiter <- as.numeric(readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
distancex <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
    FALSE))
distancey <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
    FALSE))
# add boundary conditions 0= fixed head, 1= no flow
boundarytop <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
     FALSE))
```

```
boundarybottom <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
boundaryleft <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul
    = FALSE))
boundaryright <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul
    = FALSE))
hydhead <-(readLines(zz, n = nrows, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
    FALSE))
# hydhead is now the initial condition array #
hydcondx <-(readLines(zz, n = nrows, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
    FALSE))
hydcondy <-(readLines(zz, n = nrows, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
    FALSE))
# add pumping array
pumping <-(readLines(zz, n = nrows, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
    FALSE))
close(zz)
# split the multiple column strings into numeric components for a vector
distancex <-as.numeric(unlist(strsplit(distancex,split=" ")))
distancey <-as.numeric(unlist(strsplit(distancey,split=" ")))
boundarytop <-as.numeric(unlist(strsplit(boundarytop,split=" ")))
boundarybottom <-as.numeric(unlist(strsplit(boundarybottom,split=" ")))
boundaryleft <-as.numeric(unlist(strsplit(boundaryleft,split=" ")))
boundaryright <-as.numeric(unlist(strsplit(boundaryright,split=" ")))
hydhead <-as.numeric(unlist(strsplit(hydhead,split=" ")))
hydcondx <-as.numeric(unlist(strsplit(hydcondx,split=" ")))
hydcondy <-as.numeric(unlist(strsplit(hydcondy,split=" ")))
pumping <-as.numeric(unlist(strsplit(pumping,split=" ")))
# convert the numeric vectors into matrices for easier indexing
hydhead <- matrix(hydhead,nrow=nrows,ncol=ncols,byrow = TRUE)
hydcondx <-matrix(hydcondx,nrow=nrows,ncol=ncols,byrow = TRUE)
hydcondy <-matrix(hydcondy,nrow=nrows,ncol=ncols,byrow = TRUE)
pumping <-matrix(pumping,nrow=nrows,ncol=ncols,byrow = TRUE)
# here we perform the velocity potential calculations
# built the transmissivity arrays
amat<-matrix(0,nrows,ncols)
bmat<-matrix(0,nrows,ncols)
cmat<-matrix(0,nrows,ncols)
dmat<-matrix(0,nrows,ncols)
qrat<-matrix(0,nrows,ncols)
for(irow in 2:(nrows-1)){
  for(jcol in 2:(ncols-1)){
    amat[irow,jcol]<-((hydcondx[irow-1,jcol  ]+hydcondx[irow  ,jcol  ])*deltaz)/(2.0*deltax
        ^2)
    bmat[irow,jcol]<-((hydcondx[irow  ,jcol  ]+hydcondx[irow+1,jcol  ])*deltaz)/(2.0*deltax
        ^2)
    cmat[irow,jcol]<-((hydcondy[irow  ,jcol-1]+hydcondy[irow  ,jcol  ])*deltaz)/(2.0*deltay
        ^2)
    dmat[irow,jcol]<-((hydcondy[irow  ,jcol  ]+hydcondy[irow  ,jcol+1])*deltaz)/(2.0*deltay
        ^2)
    qrat[irow,jcol]<-(-1.0)*pumping[irow,jcol]/(deltax*deltay)
  }
}
headold <- hydhead # copy the head array, used to test for stopping
tolflag <- FALSE
for (iter in 1:maxiter){
# Boundary Conditions
# Top and Bottom
for(jcol in 1:ncols){
    if(boundarytop[jcol] == 0){hydhead[1,jcol]<-hydhead[2,jcol]} #no-flow at top
    if(boundarybottom[jcol] == 0){hydhead[nrows,jcol]<-hydhead[nrows-1,jcol]} #no-flow at
        bottom
    # otherwise values are fixed head
}
for(irow in 1:nrows){
    if(boundaryleft[irow] == 0){hydhead[irow,1]<-hydhead[irow,2]} #no-flow at left
    if(boundaryright[irow] == 0){hydhead[irow,ncols]<-hydhead[irow,ncols-1]} #no-flow at
        right
    # otherwise values are fixed head
}
  for (irow in 2:(nrows-1)){
    for (jcol in 2:(ncols-1)){
      hydhead[irow,jcol] <-
          (                       qrat[irow,jcol] +
           amat[irow,jcol]*hydhead[irow-1,jcol  ] +
           bmat[irow,jcol]*hydhead[irow+1,jcol  ] +
           cmat[irow,jcol]*hydhead[irow  ,jcol-1] +
           dmat[irow,jcol]*hydhead[irow  ,jcol+1]  )/
        (amat[irow,jcol]+bmat[irow,jcol]+cmat[irow,jcol]+dmat[irow,jcol])
    }
  }
  # test for stopping iterations
  percentdiff <- sum((hydhead-headold)^2)
  if (percentdiff < tolerance){
    message("Exit iterations in velocity potential because tolerance met");
    message("Iterations =", iter);
    message("Current error : ",percentdiff);
    tolflag <- TRUE
    break}
 headold<-hydhead  #update the current head vector
 if( iter %% 1000 == 0){message("Calculating in Potential Function ",iter," of ",maxiter, "
      iterations")}
```

```
}
if (tolflag == FALSE){
  message("Exit iterations in potential function at max iterations")
  message("Current error : ",percentdiff)
  }
#################################################################
###    built position arrays for contour plotting          ###
#################################################################
velocity_plt<-matrix(0,ncols,nrows)
for( i in 1:nrows){
  for( j in 1:ncols){
    velocity_plt[j,i]=hydhead[(nrows+1)-i,j]
  }
}
#################################################################
###    contour plot of head -- note axes are rotated       ###
#################################################################
contour(distancex,distancey,velocity_plt,
        plot.title = title(main = "Head (Blue) Map",
                           xlab = "Meters (X axis) ====>>",
                           ylab = "Meters (Y axis) ====>>"),
        col="blue",lwd=3,nlevels=21,xlim=c(0,300),ylim=c(0,300),zlim=c(-1,40),tck=1)
# write to an ASCII file to show in Excel
write(t(hydhead), file='damseepage-out.txt',ncolumns = ncols,sep=",")
message("min head : ",min(hydhead))
```

Example 5: 4 Wells in a rectangular aquifer Figure 19 is a rectangular aquifer with 4 wells as shown. The aquifer thickness is 1 meters. The aquifer is surrounded with a constant head boundary of 30 meters. The hydraulic conductivity is $K = 0.033\ m/day$.

Using a 10 *meter* $\times$ 10 *meter* grid spacing estimate the pumping rate in each well so that the head within the rectangular area defined by the well field is no greater than 15 meters.

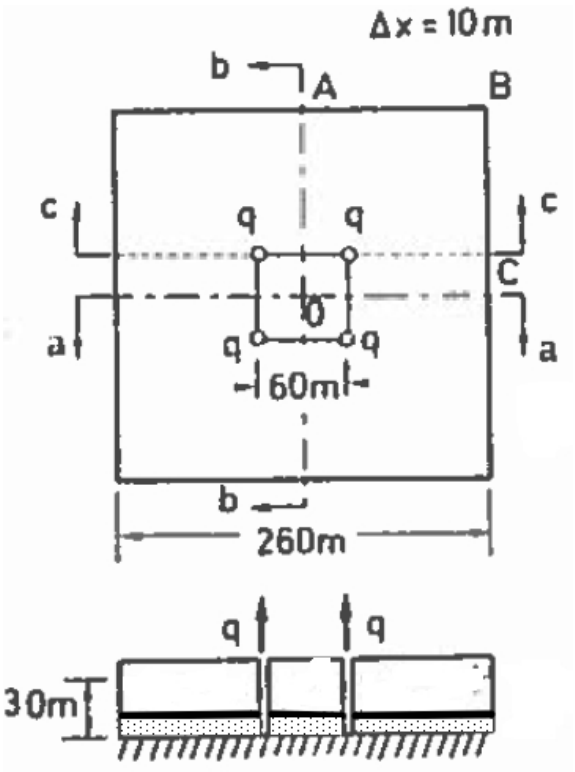


Figure 19. Rectangular aquifer with 4 wells.

Listing 11 is the input file for the modified script for the example problem. The file takes several pages in this document — as the input requirements become substantial, one would write separate scripts just to build input files to structure them properly. We are essentially constructing a specialized database, and manually building that database can become tedious whereas we can automatically populate portions of the database with a script.

Listing 11. Input file for 4 well example.

```
10
10
1
28
28
1e-16
5000
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240
    250 260 270
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240
    250 260 270
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
    0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
```

```
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
     0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033 0.033
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
     0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
     0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
     0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
     0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
     0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
     0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
     0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
```

```
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.430 0.000 0.000 0.000 0.000
    0.430 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.430 0.000 0.000 0.000 0.000
    0.430 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
```

Figure 20 is a screen capture of the script run with the datafile above. The contour plot shows the “cone of depression” around the well field. The script has an added message that reports the smallest head, and that report was used to adjust the pumping rates at the four wells until the value was at the prescribed 15 meters.

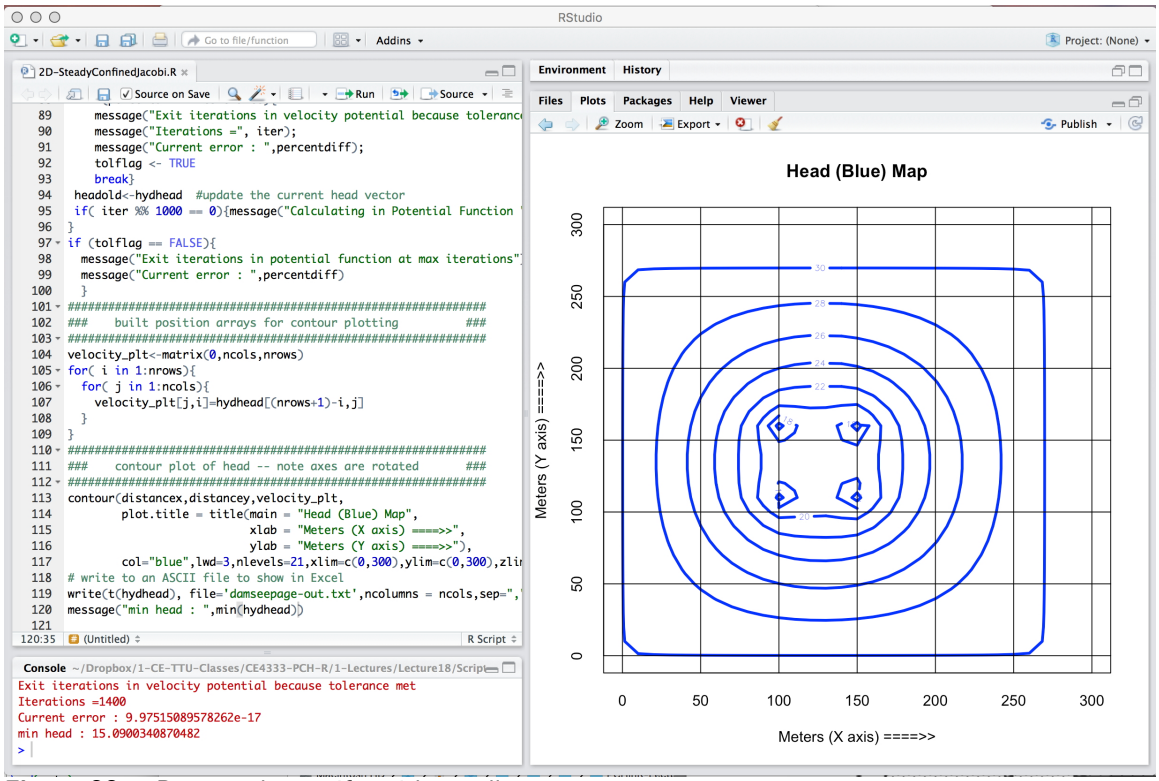


Figure 20. Rectangular aquifer with 4 wells.

The value for a single well is 0.430 $\frac{m^3}{day}$ (using the units of hydraulic conductivity as a guide). If the goal was then to specify particular pumps, that flow value and the required lift would be used to specify and purchase pumps for the system.

For our problem, it is a simple matter to modify Example 5 to perform the needed tasks to construct the constraint equation coefficients. We need to run a single simulation to determine the pre-development head distribution in the aquifer system, then run 25 simulations with pumping in a single aquifer cell to generate a head distribution that represents the steady state head distribution in the aquifer caused by the pumping in one cell, starting in cell 1 and ending in cell 25. Then each of these distributions is subtracted from the pre-development head distribution to construct the drawdown induced by each pump. Because superposition can be applied, these resulting arrays become the coefficients in the constraint matrix. Listing 12 is a listing that performs the 26 simulations, then writes the output to a file named `influence-matrices-out.txt`.

Listing 12. R code to simulate response to groundwater pumping.

```
# 2D Steady Confined -- With Boundary Arrays -- And Pumping Array
# 2D Aquifer Flow Model using Jacobi Iteration
# deallocate memory
rm(list=ls())
#here we work through the file names
filenames <- c("base-case.txt",
               "pump1.txt","pump2.txt","pump3.txt","pump4.txt","pump5.txt",
               "pump6.txt","pump7.txt","pump8.txt","pump9.txt","pump10.txt",
               "pump11.txt","pump12.txt","pump13.txt","pump14.txt","pump15.txt",
               "pump16.txt","pump17.txt","pump18.txt","pump19.txt","pump20.txt",
               "pump21.txt","pump22.txt","pump23.txt","pump24.txt","pump25.txt")
outfile <- file("influence-matrices-out.txt","w")
```

```
for (indexFile in 1:26){
zz <- file(filenames[indexFile], "r") # Open a connection named zz to file named input.dat
# read the simulation conditons
deltax <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
deltay <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
deltaz <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
nrows <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
ncols <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
tolerance <- as.numeric(readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
maxiter <- as.numeric(readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
distancex <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
    FALSE))
distancey <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
    FALSE))
# add boundary conditions 0= fixed head, 1= no flow
boundarytop <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
     FALSE))
boundarybottom <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown",
    skipNul = FALSE))
boundaryleft <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul
    = FALSE))
boundaryright <- (readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul
     = FALSE))
hydhead <-(readLines(zz, n = nrows, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
    FALSE))
# hydhead is now the initial condition array #
hydcondx <-(readLines(zz, n = nrows, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
     FALSE))
hydcondy <-(readLines(zz, n = nrows, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
     FALSE))
# add pumping array
pumping <-(readLines(zz, n = nrows, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul =
    FALSE))
close(zz)
# split the multiple column strings into numeric components for a vector
distancex <-as.numeric(unlist(strsplit(distancex,split=" ")))
distancey <-as.numeric(unlist(strsplit(distancey,split=" ")))
boundarytop <-as.numeric(unlist(strsplit(boundarytop,split=" ")))
boundarybottom <-as.numeric(unlist(strsplit(boundarybottom,split=" ")))
boundaryleft <-as.numeric(unlist(strsplit(boundaryleft,split=" ")))
boundaryright <-as.numeric(unlist(strsplit(boundaryright,split=" ")))
hydhead <-as.numeric(unlist(strsplit(hydhead,split=" ")))
hydcondx <-as.numeric(unlist(strsplit(hydcondx,split=" ")))
hydcondy <-as.numeric(unlist(strsplit(hydcondy,split=" ")))
pumping <-as.numeric(unlist(strsplit(pumping,split=" ")))
# convert the numeric vectors into matrices for easier indexing
hydhead <- matrix(hydhead,nrow=nrows,ncol=ncols,byrow = TRUE)
hydcondx <-matrix(hydcondx,nrow=nrows,ncol=ncols,byrow = TRUE)
hydcondy <-matrix(hydcondy,nrow=nrows,ncol=ncols,byrow = TRUE)
pumping <-matrix(pumping,nrow=nrows,ncol=ncols,byrow = TRUE)
# debug
# print(boundarytop)
# print(boundarybottom)
# print(boundaryleft)
# print(boundaryright)
print(hydhead)
print(hydcondx)
print(pumping)
# here we perform the velocity potential calculations
# built the transmissivity arrays
amat<-matrix(0,nrows,ncols)
bmat<-matrix(0,nrows,ncols)
cmat<-matrix(0,nrows,ncols)
dmat<-matrix(0,nrows,ncols)
qrat<-matrix(0,nrows,ncols)
for(irow in 2:(nrows-1)){
  for(jcol in 2:(ncols-1)){
    amat[irow,jcol]<-((hydcondx[irow-1,jcol  ]+hydcondx[irow  ,jcol  ])*deltaz)/(2.0*deltax
        ^2)
    bmat[irow,jcol]<-((hydcondx[irow  ,jcol  ]+hydcondx[irow+1,jcol  ])*deltaz)/(2.0*deltax
        ^2)
    cmat[irow,jcol]<-((hydcondy[irow  ,jcol-1]+hydcondy[irow  ,jcol  ])*deltaz)/(2.0*deltay
        ^2)
    dmat[irow,jcol]<-((hydcondy[irow  ,jcol  ]+hydcondy[irow  ,jcol+1])*deltaz)/(2.0*deltay
        ^2)
    qrat[irow,jcol]<-(-1.0)*pumping[irow,jcol]/(deltax*deltay)
  }
}
headold <- hydhead # copy the head array, used to test for stopping
drawdown <- hydhead # copy the head array, used for drawdown calculation
tolflag <- FALSE
for (iter in 1:maxiter){
# Boundary Conditions
# Top and Bottom
for(jcol in 1:ncols){
```

```
    if(boundarytop[jcol] == 0){hydhead[1,jcol]<-hydhead[2,jcol]} #no-flow at top
    if(boundarybottom[jcol] == 0){hydhead[nrows,jcol]<-hydhead[nrows-1,jcol]} #no-flow at
        bottom
    # otherwise values are fixed head
}
for(irow in 1:nrows){
    if(boundaryleft[irow] == 0){hydhead[irow,1]<-hydhead[irow,2]} #no-flow at left
    if(boundaryright[irow] == 0){hydhead[irow,ncols]<-hydhead[irow,ncols-1]} #no-flow at
        right
    # otherwise values are fixed head
}
  for (irow in 2:(nrows-1)){
    for (jcol in 2:(ncols-1)){
      hydhead[irow,jcol] <-
          (                       qrat[irow,jcol] +
           amat[irow,jcol]*hydhead[irow-1,jcol  ] +
           bmat[irow,jcol]*hydhead[irow+1,jcol  ] +
           cmat[irow,jcol]*hydhead[irow  ,jcol-1] +
           dmat[irow,jcol]*hydhead[irow  ,jcol+1]  )/
        (amat[irow,jcol]+bmat[irow,jcol]+cmat[irow,jcol]+dmat[irow,jcol])
    }
  }
  # test for stopping iterations
  percentdiff <- sum((hydhead-headold)^2)
  if (percentdiff <= tolerance){
    message("Exit iterations in velocity potential because tolerance met");
    message("Iterations =", iter);
    message("Current error : ",percentdiff);
    tolflag <- TRUE
    break}
 headold<-hydhead  #update the current head vector
 if( iter %% 1000 == 0){message("Calculating in Potential Function ",iter," of ",maxiter, "
      iterations")}
}
if (tolflag == FALSE){
  message("Exit iterations in potential function at max iterations")
  message("Current error : ",percentdiff)
  }
#############################################################
###    built position arrays for contour plotting          ###
#############################################################
velocity_plt<-matrix(0,ncols,nrows)
for( i in 1:nrows){
  for( j in 1:ncols){
    velocity_plt[j,i]=hydhead[(nrows+1)-i,j]
  }
}
#############################################################
###    contour plot of head -- note axes are rotated       ###
#############################################################
contour(distancex,distancey,velocity_plt,
        plot.title = title(main = "Head (Blue) Map",
                           xlab = "Meters (X axis) ====>>",
                           ylab = "Meters (Y axis) ====>>"),
        col="blue",lwd=3,nlevels=21,tck=1)
# write to an ASCII file to show in Excel

if( indexFile == 1){
  basecase <- hydhead;
  drawdown <- drawdown - hydhead
  write(paste(filenames[indexFile]," predevelopment head "),file=outfile,ncolumns=1)
  write(t(hydhead), file=outfile,ncolumns = ncols,sep=",")
}
if (indexFile > 1){
  drawdown <- basecase - hydhead
  write(paste(filenames[indexFile], " drawdown matrix "),file=outfile,ncolumns=1)
  write(t(drawdown), file=outfile,ncolumns = ncols,sep=",")
}
}
close(outfile)
```

Listing 13. Representative Input File(s) for Groundwater Simulation Model.

```
--- REFERENCE CASE --
2000
2000
365
7
7
1e-12
5000
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 0 0 0 0 0 0
1 0 0 0 0 0 0
1 1 1 1 1 1 1
0 0 0 0 0 0 0
0 2.795639 7.1792 10.46687 12.65865 13.75454 13.75454
0 2.795639 7.1792 10.46687 12.65865 13.75454 13.75454
0 2.795639 7.1792 10.46687 12.65865 13.75454 13.75454
```

```
0 2.795639 7.1792 10.46687 12.65865 13.75454 13.75454
0 2.795639 7.1792 10.46687 12.65865 13.75454 13.75454
0 2.795639 7.1792 10.46687 12.65865 13.75454 13.75454
0 2.795639 7.1792 10.46687 12.65865 13.75454 13.75454
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
-4E05 -4E05 -4E05 -4E05 -4E05 -4E05 -4E05
-4E05 -4E05 -4E05 -4E05 -4E05 -4E05 -4E05
-4E05 -4E05 -4E05 -4E05 -4E05 -4E05 -4E05
-4E05 -4E05 -4E05 -4E05 -4E05 -4E05 -4E05
-4E05 -4E05 -4E05 -4E05 -4E05 -4E05 -4E05
-4E05 -4E05 -4E05 -4E05 -4E05 -4E05 -4E05
-4E05 -4E05 -4E05 -4E05 -4E05 -4E05 -4E05
--- PUMP IN CELL 5 ACTIVE ---
2000
2000
365
7
7
1e-21
5000
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 0 0 0 0 0 0
1 0 0 0 0 0 0
1 1 1 1 1 1 1
0 0 0 0 0 0 0
0 2.79 7.22 10.55 12.77 13.88 13.88
0 2.79 7.22 10.55 12.77 13.88 13.88
0 2.79 7.22 10.55 12.77 13.88 13.88
0 2.79 7.22 10.55 12.77 13.88 13.88
0 2.79 7.22 10.55 12.77 13.88 13.88
0 2.79 7.22 10.55 12.77 13.88 13.88
0 2.79 7.22 10.55 12.77 13.88 13.88
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
2920.0 1000.0 1000.0 1000.0 1000.0 1000.0 1000.0
-4E05 -4E05 -4E05 -4E05 -4E05 -4E05 -4E05
-4E05 -4E05 -4E05 -4E05 -4E05 -4E05 -4E05
-4E05 -4E05 -4E05 -4E05 -4E05 -4E05 -4E05
-4E05 -4E05 -4E05 -4E05 -4E05 -4E05 -4E05
-4E05 -4E05 -4E05 -4E05 -4E05 -4E05 -4E05
-4E05 -4E05 -4E05 -4E05 -4E05 6E05 -4E05
-4E05 -4E05 -4E05 -4E05 -4E05 -4E05 -4E05
```

Listing 14. R code to construct influence matrix from groundwater simulation model.

```
base-case.txt  predevelopment head
0,2.795639,7.1792,10.46687,12.65865,13.75454,13.75454
0,2.795639,7.1792,10.46687,12.65865,13.75454,13.75454
0,2.795639,7.1792,10.46687,12.65865,13.75454,13.75454
0,2.795639,7.1792,10.46687,12.65865,13.75454,13.75454
0,2.795639,7.1792,10.46687,12.65865,13.75454,13.75454
0,2.795639,7.1792,10.46687,12.65865,13.75454,13.75454
0,2.795639,7.1792,10.46687,12.65865,13.75454,13.75454
pump1.txt  drawdown matrix
0,0.330387,1.000676,1.752327,2.700742,4.131255,4.131255
0,0.3303868,1.000676,1.752327,2.700743,4.131255,4.131255
0,0.3076552,0.9193137,1.555562,2.218647,2.822041,2.822041
0,0.2762686,0.8133615,1.331962,1.796241,2.116223,2.116223
0,0.2492751,0.7259019,1.162682,1.518133,1.730387,1.730386
0,0.2342335,0.6782889,1.074731,1.383223,1.556804,1.556804
0,0.2342337,0.6782892,1.074731,1.383223,1.556804,1.556804
pump2.txt  drawdown matrix
0,0.3076554,0.919314,1.555562,2.218646,2.822041,2.822041
0,0.3076552,0.9193138,1.555562,2.218647,2.822041,2.822041
0,0.2990002,0.8947236,1.528726,2.278337,3.425436,3.425436
```

```
0,0.2806617,0.8318541,1.386282,1.940539,2.436205,2.436205
0,0.261227,0.7657485,1.244011,1.661331,1.942641,1.94264
0,0.2492751,0.7259019,1.162682,1.518133,1.730387,1.730386
0,0.2492754,0.7259022,1.162682,1.518133,1.730386,1.730386
pump3.txt  drawdown matrix
0,0.2762689,0.8133618,1.331961,1.796241,2.116223,2.116223
0,0.2762687,0.8133616,1.331962,1.796241,2.116223,2.116223
0,0.2806617,0.8318541,1.386282,1.940539,2.436205,2.436205
0,0.2839586,0.8471106,1.440775,2.143427,3.251854,3.251854
0,0.2806617,0.8318541,1.386282,1.940539,2.436205,2.436205
0,0.2762686,0.8133615,1.331962,1.796241,2.116223,2.116223
0,0.2762689,0.8133618,1.331961,1.796241,2.116223,2.116223
pump4.txt  drawdown matrix
0,0.2492754,0.7259022,1.162682,1.518133,1.730386,1.730386
0,0.2492752,0.725902,1.162682,1.518133,1.730387,1.730386
0,0.261227,0.7657485,1.244011,1.661331,1.942641,1.94264
0,0.2806617,0.8318541,1.386282,1.940539,2.436205,2.436205
0,0.2990002,0.8947236,1.528726,2.278337,3.425436,3.425436
0,0.3076552,0.9193137,1.555562,2.218647,2.822041,2.822041
0,0.3076554,0.919314,1.555562,2.218646,2.822041,2.822041
pump5.txt  drawdown matrix
0,0.2342337,0.6782892,1.074731,1.383223,1.556804,1.556804
0,0.2342335,0.678289,1.074731,1.383223,1.556804,1.556804
0,0.2492751,0.7259019,1.162682,1.518133,1.730387,1.730386
0,0.2762686,0.8133615,1.331962,1.796241,2.116223,2.116223
0,0.3076552,0.9193137,1.555562,2.218647,2.822041,2.822041
0,0.3303868,1.000676,1.752327,2.700743,4.131255,4.131255
0,0.330387,1.000676,1.752327,2.700742,4.131255,4.131255
pump6.txt  drawdown matrix
0,0.3531186,1.082038,1.949091,3.182838,2.700742,2.700742
0,0.3531184,1.082038,1.949091,3.182839,2.700742,2.700742
0,0.3163101,0.9439038,1.582398,2.158956,2.218646,2.218646
0,0.2718756,0.7948689,1.277641,1.651943,1.79624,1.79624
0,0.2373232,0.6860553,1.081353,1.374935,1.518133,1.518132
0,0.2191918,0.6306759,0.9867803,1.248312,1.383222,1.383222
0,0.2191921,0.6306762,0.9867803,1.248312,1.383222,1.383222
pump7.txt  drawdown matrix
0,0.3163104,0.9439041,1.582398,2.158956,2.218646,2.218646
0,0.3163102,0.9439039,1.582398,2.158956,2.218646,2.218646
0,0.3086838,0.933003,1.644334,2.675825,2.278336,2.278336
0,0.2817578,0.8350902,1.38611,1.881948,1.940538,1.940538
0,0.2537442,0.7394895,1.183068,1.52532,1.66133,1.66133
0,0.2373232,0.6860553,1.081353,1.374935,1.518133,1.518132
0,0.2373235,0.6860555,1.081353,1.374935,1.518132,1.518132
pump8.txt  drawdown matrix
0,0.2718758,0.7948692,1.277641,1.651943,1.79624,1.79624
0,0.2718756,0.794869,1.277641,1.651943,1.79624,1.79624
0,0.2817578,0.8350902,1.38611,1.881948,1.940538,1.940538
0,0.2905524,0.8776237,1.549762,2.549203,2.143426,2.143426
0,0.2817578,0.8350902,1.38611,1.881948,1.940538,1.940538
0,0.2718756,0.7948689,1.277641,1.651943,1.796241,1.79624
0,0.2718758,0.7948692,1.277641,1.651943,1.79624,1.79624
pump9.txt  drawdown matrix
0,0.2373235,0.6860555,1.081353,1.374935,1.518132,1.518132
0,0.2373233,0.6860553,1.081353,1.374935,1.518132,1.518132
0,0.2537442,0.7394895,1.183068,1.52532,1.66133,1.66133
0,0.2817578,0.8350902,1.38611,1.881948,1.940538,1.940538
0,0.3086838,0.933003,1.644334,2.675825,2.278336,2.278336
0,0.3163101,0.9439038,1.582398,2.158956,2.218646,2.218646
0,0.3163104,0.9439041,1.582398,2.158956,2.218646,2.218646
pump10.txt  drawdown matrix
0,0.2191921,0.6306762,0.9867803,1.248312,1.383222,1.383222
0,0.2191919,0.630676,0.9867803,1.248312,1.383222,1.383222
0,0.2373232,0.6860553,1.081353,1.374935,1.518133,1.518132
0,0.2718756,0.7948689,1.277641,1.651943,1.79624,1.79624
0,0.3163101,0.9439038,1.582398,2.158956,2.218646,2.218646
0,0.3531183,1.082038,1.949091,3.182839,2.700742,2.700742
0,0.3531186,1.082038,1.949091,3.182838,2.700742,2.700742
pump11.txt  drawdown matrix
0,0.4126584,1.301535,2.51255,1.949091,1.752326,1.752326
0,0.4126582,1.301534,2.51255,1.949091,1.752326,1.752326
0,0.3325915,0.9793947,1.547298,1.582397,1.555561,1.555561
0,0.2576003,0.736155,1.11485,1.27764,1.33196,1.33196
0,0.2089504,0.5927744,0.8983084,1.081352,1.162681,1.162681
0,0.1860188,0.5276835,0.8042571,0.9867796,1.07473,1.07473
0,0.1860191,0.5276838,0.8042571,0.9867795,1.07473,1.07473
pump12.txt  drawdown matrix
0,0.3325917,0.979395,1.547298,1.582397,1.555561,1.555561
0,0.3325915,0.9793948,1.547298,1.582397,1.555561,1.555561
0,0.337667,1.058295,2.080102,1.644333,1.528725,1.528725
0,0.2839416,0.8360141,1.330756,1.386109,1.386281,1.386281
0,0.2346687,0.6710641,1.020799,1.183067,1.24401,1.24401
0,0.2089504,0.5927744,0.8983084,1.081352,1.162681,1.162681
0,0.2089507,0.5927747,0.8983084,1.081352,1.162681,1.162681
pump13.txt  drawdown matrix
0,0.2576005,0.7361552,1.11485,1.27764,1.33196,1.33196
0,0.2576003,0.736155,1.11485,1.27764,1.33196,1.33196
0,0.2839416,0.8360141,1.330756,1.386109,1.386281,1.386281
0,0.3147354,0.9932036,1.986051,1.549761,1.440774,1.440774
0,0.2839416,0.8360141,1.330756,1.386109,1.386281,1.386281
0,0.2576003,0.736155,1.11485,1.27764,1.33196,1.33196
0,0.2576005,0.7361552,1.11485,1.27764,1.33196,1.33196
```

```
pump14.txt  drawdown matrix
0,0.2089507,0.5927747,0.8983084,1.081352,1.162681,1.162681
0,0.2089505,0.5927745,0.8983084,1.081352,1.162681,1.162681
0,0.2346687,0.6710641,1.020799,1.183067,1.24401,1.24401
0,0.2839416,0.8360141,1.330756,1.386109,1.386281,1.386281
0,0.337667,1.058294,2.080102,1.644333,1.528725,1.528725
0,0.3325915,0.9793947,1.547298,1.582397,1.555561,1.555561
0,0.3325917,0.979395,1.547298,1.582397,1.555561,1.555561
pump15.txt  drawdown matrix
0,0.1860191,0.5276838,0.8042571,0.9867795,1.07473,1.07473
0,0.1860189,0.5276836,0.8042571,0.9867796,1.07473,1.07473
0,0.2089504,0.5927744,0.8983084,1.081352,1.162681,1.162681
0,0.2576003,0.736155,1.11485,1.27764,1.33196,1.33196
0,0.3325915,0.9793947,1.547298,1.582397,1.555561,1.555561
0,0.4126581,1.301534,2.51255,1.949091,1.752326,1.752326
0,0.4126584,1.301535,2.51255,1.949091,1.752326,1.752326
pump16.txt  drawdown matrix
0,0.5522649,1.84317,1.301533,1.082036,1.000674,1.000674
0,0.5522647,1.84317,1.301533,1.082036,1.000674,1.000674
0,0.3437973,0.9359858,0.9793938,0.9439022,0.9193117,0.9193115
0,0.2169837,0.5775819,0.7361541,0.7948673,0.8133595,0.8133593
0,0.1548593,0.4212038,0.5927735,0.6860536,0.7258999,0.7258997
0,0.1299142,0.3596003,0.5276826,0.6306743,0.6782869,0.6782867
0,0.1299144,0.3596006,0.5276826,0.6306741,0.6782867,0.6782867
pump17.txt  drawdown matrix
0,0.3437976,0.9359861,0.9793938,0.943902,0.9193115,0.9193115
0,0.3437974,0.9359859,0.9793938,0.9439022,0.9193117,0.9193115
0,0.425451,1.484766,1.058294,0.9330014,0.8947216,0.8947214
0,0.2816729,0.7796077,0.8360132,0.8350886,0.8318521,0.8318519
0,0.1920385,0.5159784,0.6710632,0.7394879,0.7657465,0.7657463
0,0.1548593,0.4212038,0.5927735,0.6860536,0.7258999,0.7258997
0,0.1548596,0.4212041,0.5927734,0.6860535,0.7258997,0.7258997
pump18.txt  drawdown matrix
0,0.2169839,0.5775822,0.736154,0.7948671,0.8133593,0.8133593
0,0.2169837,0.577582,0.7361541,0.7948672,0.8133594,0.8133593
0,0.2816729,0.7796077,0.8360132,0.8350886,0.8318521,0.8318519
0,0.4005058,1.423163,0.9932027,0.877622,0.8471086,0.8471084
0,0.2816729,0.7796077,0.8360132,0.8350886,0.8318521,0.8318519
0,0.2169837,0.5775819,0.7361541,0.7948673,0.8133595,0.8133593
0,0.2169839,0.5775822,0.736154,0.7948671,0.8133593,0.8133593
pump19.txt  drawdown matrix
0,0.1548596,0.4212041,0.5927734,0.6860535,0.7258997,0.7258997
0,0.1548594,0.4212039,0.5927735,0.6860536,0.7258998,0.7258997
0,0.1920386,0.5159784,0.6710632,0.7394879,0.7657465,0.7657463
0,0.2816729,0.7796077,0.8360132,0.8350886,0.8318521,0.8318519
0,0.425451,1.484766,1.058294,0.9330014,0.8947216,0.8947214
0,0.3437973,0.9359858,0.9793938,0.9439022,0.9193117,0.9193115
0,0.3437976,0.9359861,0.9793938,0.943902,0.9193115,0.9193115
pump20.txt  drawdown matrix
0,0.1299144,0.3596006,0.5276826,0.6306741,0.6782867,0.6782867
0,0.1299142,0.3596004,0.5276826,0.6306742,0.6782868,0.6782867
0,0.1548593,0.4212038,0.5927735,0.6860536,0.7258999,0.7258997
0,0.2169837,0.5775819,0.7361541,0.7948673,0.8133595,0.8133593
0,0.3437973,0.9359858,0.9793938,0.9439022,0.9193117,0.9193115
0,0.5522646,1.84317,1.301533,1.082036,1.000674,1.000674
0,0.5522649,1.84317,1.301533,1.082036,1.000674,1.000674
pump21.txt  drawdown matrix
0,0.9003386,0.5522642,0.4126565,0.3531159,0.3303839,0.3303839
0,0.9003384,0.552264,0.4126566,0.353116,0.330384,0.3303839
0,0.2733495,0.3437967,0.3325899,0.3163078,0.3076525,0.3076523
0,0.1116778,0.216983,0.2575987,0.2718733,0.2762659,0.2762658
0,0.063589,0.1548586,0.2089488,0.2373209,0.2492724,0.2492722
0,0.04886441,0.1299135,0.1860172,0.2191895,0.2342308,0.2342306
0,0.04886467,0.1299138,0.1860172,0.2191894,0.2342306,0.2342306
pump22.txt  drawdown matrix
0,0.2733497,0.3437969,0.3325898,0.3163077,0.3076523,0.3076523
0,0.2733495,0.3437967,0.3325899,0.3163078,0.3076524,0.3076523
0,0.7386667,0.4254503,0.3376654,0.3086815,0.2989975,0.2989973
0,0.2252606,0.2816723,0.28394,0.2817555,0.280659,0.2806588
0,0.09695324,0.1920379,0.2346671,0.2537419,0.2612243,0.2612241
0,0.063589,0.1548586,0.2089488,0.2373209,0.2492724,0.2492722
0,0.06358926,0.1548589,0.2089488,0.2373208,0.2492722,0.2492722
pump23.txt  drawdown matrix
0,0.1116781,0.2169833,0.2575987,0.2718731,0.2762658,0.2762658
0,0.1116779,0.2169831,0.2575987,0.2718732,0.2762659,0.2762658
0,0.2252607,0.2816723,0.28394,0.2817555,0.280659,0.2806588
0,0.7239421,0.4005052,0.3147338,0.2905501,0.2839559,0.2839557
0,0.2252606,0.2816723,0.28394,0.2817555,0.280659,0.2806588
0,0.1116778,0.216983,0.2575987,0.2718733,0.2762659,0.2762658
0,0.1116781,0.2169833,0.2575987,0.2718731,0.2762658,0.2762658
pump24.txt  drawdown matrix
0,0.06358926,0.1548589,0.2089488,0.2373208,0.2492722,0.2492722
0,0.06358905,0.1548587,0.2089488,0.2373209,0.2492724,0.2492722
0,0.09695325,0.1920379,0.2346671,0.2537419,0.2612243,0.2612241
0,0.2252606,0.2816723,0.28394,0.2817555,0.280659,0.2806588
0,0.7386667,0.4254503,0.3376654,0.3086815,0.2989975,0.2989973
0,0.2733495,0.3437966,0.3325899,0.3163078,0.3076525,0.3076523
0,0.2733497,0.3437969,0.3325898,0.3163077,0.3076523,0.3076523
pump25.txt  drawdown matrix
0,0.04886467,0.1299138,0.1860172,0.2191894,0.2342306,0.2342306
0,0.04886446,0.1299136,0.1860172,0.2191895,0.2342307,0.2342306
0,0.06358901,0.1548587,0.2089488,0.2373209,0.2492724,0.2492722
```

```
0,0.1116778,0.216983,0.2575987,0.2718733,0.2762659,0.2762658
0,0.2733495,0.3437966,0.3325899,0.3163078,0.3076525,0.3076523
0,0.9003384,0.5522639,0.4126566,0.3531161,0.3303841,0.3303839
0,0.9003386,0.5522642,0.4126565,0.3531159,0.3303839,0.3303839
```

Next we analyze the output from the program to construct the influence coefficient matrix.

Listing 15. R code to construct influence matrix from groundwater simulation model.

```
# Make Influence Table for LP Soultion to Groundwater Allocation Problem
rm(list=ls())
zz <- file("influence-matrices-out.txt", "r") # Open a connection named zz to file named
    input.dat
# influence-matrices contain 26 tables, each 7X7+label
# ignore the first 8 rows, then capture the rest.
alist <-character(0)
amatrix <- array(0,dim=c(7,7,25))
bmatrix <- array(0,dim=c(25,25))
label <-readLines(zz, n = 1, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul = FALSE)
dummy <-readLines(zz, n = 7, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul = FALSE)
for (i in 1:25){
  label <-readLines(zz, n = 1, ok = TRUE, warn = TRUE,
                              encoding = "unknown", skipNul = FALSE) #clobber the existing
                                   label
  alist <-readLines(zz, n = 7, ok = TRUE, warn = TRUE,encoding = "unknown", skipNul = FALSE
      )
# split the list
  temparray<-as.numeric(unlist(strsplit(alist,split=",")))
# convert into a matrix
  temparray <- matrix(temparray,nrow=7,ncol=7,byrow = TRUE)
# now move temp array into the big array
  for (j in 1:7){
    for (k in 1:7){
      amatrix[j,k,i] <- temparray[j,k];
    }
  }
}
close(zz)
# split the list
# rearrange the result into 49X49 array
for (ipump in 1:25){
  icell <- 0
  for (j in 6:2){
    for(i in 2:6){
      icell <-icell+1
      bmatrix[icell,ipump] <- amatrix[i,j,ipump]
      }
  }
}
# write the result
outfile <- file("influence-table-output.txt","w")
write(bmatrix,file=outfile,ncolumns=25,sep=",")
close(outfile)
```

Listing 16. Influence Table Output File (for building the LP model.

```
4.131255,2.822041,2.116223,1.730387,1.556804,2.700743,2.218647,1.796241,
1.518133,1.383223,1.752327,1.555562,1.331962,1.162682,1.074731,1.000676,
0.9193137,0.8133615,0.7259019,0.6782889,0.3303868,0.3076552,0.2762686,
0.2492751,0.2342335
2.822041,3.425436,2.436205,1.942641,1.730387,2.218647,2.278337,1.940539,
1.661331,1.518133,1.555562,1.528726,1.386282,1.244011,1.162682,0.9193138,
0.8947236,0.8318541,0.7657485,0.7259019,0.3076552,0.2990002,0.2806617,
0.261227,0.2492751
2.116223,2.436205,3.251854,2.436205,2.116223,1.796241,1.940539,2.143427,
1.940539,1.796241,1.331962,1.386282,1.440775,1.386282,1.331962,0.8133616,
0.8318541,0.8471106,0.8318541,0.8133615,0.2762687,0.2806617,0.2839586,
0.2806617,0.2762686
1.730387,1.942641,2.436205,3.425436,2.822041,1.518133,1.661331,1.940539,
2.278337,2.218647,1.162682,1.244011,1.386282,1.528726,1.555562,0.725902,
0.7657485,0.8318541,0.8947236,0.9193137,0.2492752,0.261227,0.2806617,
0.2990002,0.3076552
.... MANY ROWS BUILT FROM INFLUENCE MATRICES ....
0.2762659,0.280659,0.2839559,0.280659,0.2762659,0.2718732,0.2817555,
0.2905501,0.2817555,0.2718733,0.2575987,0.28394,0.3147338,0.28394,0.2575987,
0.2169831,0.2816723,0.4005052,0.2816723,0.216983,0.1116779,0.2252607,
0.7239421,0.2252606,0.1116778
0.2492724,0.2612243,0.280659,0.2989975,0.3076525,0.2373209,0.2537419,
0.2817555,0.3086815,0.3163078,0.2089488,0.2346671,0.28394,0.3376654,
0.3325899,0.1548587,0.1920379,0.2816723,0.4254503,0.3437966,0.06358905,
0.09695325,0.2252606,0.7386667,0.2733495
0.2342307,0.2492724,0.2762659,0.3076525,0.3303841,0.2191895,0.2373209,
0.2718733,0.3163078,0.3531161,0.1860172,0.2089488,0.2575987,0.3325899,
0.4126566,0.1299136,0.1548587,0.216983,0.3437966,0.5522639,0.04886446,
```

```
0.06358901,0.1116778,0.2733495,0.9003384
```

10.3. The LP Framework and Solution

Listing 17. R script for generating the LP model.

```
# Build the LP model
library(lpSolve)
# read in the influence matrix
zz <- read.csv("influence-table-output.txt",header=FALSE)
# build the cost vector
xdist <- c(1,3,5,7,9)
ydist <- c(1,3,5,7,9)
x0 <- 5
y0 <- 3
cost <- numeric(0)
icell <- 0
for (j in 5:1){
  for( i in 5:1){
    icell <- icell+1
    distance <- sqrt((xdist[i]-x0)^2 + (ydist[j]-y0)^2)
    cost[icell] <- 1.0 + 0.5*distance
#    print(paste(icell," dist= ",distance," row= ",i," col=",j,"cost = ",cost[icell]))
  }
}
# build the demand constraint
demand <- rep(1,25)
# build the constraint values
crhs <- c(0.00,0.00,0.00,0.00,0.00,
          0.00,0.00,0.00,0.00,0.00,
          0.00,0.00,0.00,0.00,0.00,
          6.27,6.27,6.27,6.27,6.27,
          2.15,2.15,2.15,2.15,2.15,
          7.00,
          0.00,0.00,0.00,0.00,0.00)
#
# build the LP constraint matrix
#
cmatrix <- matrix(0,nrow=31,ncol=25)
for (i in 1:25){
  for (j in 1:25){
    cmatrix[i,j] <- zz[i,j]
  }
}
cmatrix[26,] <- demand
# force Zeros in cells 21-25
cmatrix[27,] <- c(rep(0,20),1,rep(0,4))
cmatrix[28,] <- c(rep(0,21),1,rep(0,3))
cmatrix[29,] <- c(rep(0,22),1,rep(0,2))
cmatrix[30,] <- c(rep(0,23),1,rep(0,1))
cmatrix[31,] <- c(rep(0,24),1         )
#
# build the LP constraint directions
#
cdirection <- character(0)
for (i in 1:15){
  cdirection[i] <- ">="
}
for (i in 16:25){
  cdirection[i] <- "<="
}
for (i in 26:31){
  cdirection[i] <- "="
}
#
# send to the LP solver
#
prod.sol <- lp("min", cost, cmatrix, cdirection, crhs, compute.sens = TRUE)
#accessing to R output
message("Objective Value = : ", prod.sol$objval)
decisionMatrix <- matrix(0,nrow=5,ncol=5)
index <- 0
for (irow in 1:5){
  for (jcol in 1:5){
    index <- index+1
    decisionMatrix[irow,jcol] <- prod.sol$solution[index]
  }
}
message("Decision (Pumping) Matrix = : ")
print(decisionMatrix, byrow=TRUE)
message("Constraint Check = : ")
constCheck <- cmatrix%*%prod.sol$solution
print(cbind(constCheck,cdirection,crhs))
prod.sol$duals #includes duals of constraints and reduced costs of variables
#sensibility analysis results
```

```
prod.sol$duals.from
prod.sol$duals.to
prod.sol$sens.coef.from
prod.sol$sens.coef.to
# test alternate solutions
value <- t(cost)%*%c(rep(0,15),1.160381,1.797931 ,1.627169 ,1.797931,1.160381, 0.000000,
    0.000000, 0.000000, 0.000000, 0.000000)
message("Alternate Solution Objective Value = : ",value)
decisionVector <- c(rep(0,15),1.160381,1.797931 ,1.627169 ,1.797931,1.160381, 0.000000,
    0.000000, 0.000000, 0.000000, 0.000000)
index <- 0
for (irow in 1:5){
  for (jcol in 1:5){
    index <- index+1
    decisionMatrix[irow,jcol] <- decisionVector[index]
  }
}
print(decisionMatrix, byrow=TRUE)
constCheck <- cmatrix%*%c(rep(0,15),1.160381,1.797931 ,1.627169 ,1.797931,1.160381,
    0.000000, 0.000000, 0.000000, 0.000000, 0.000000)
message("Constraint Equation Values = :")
print(cbind(constCheck,cdirection,crhs))

# test alternate solutions
value <- t(cost)%*%c(rep(0,15),0.5801905,2.414519 ,1.627169 ,1.797931,0.5801905, 0.000000,
    0.000000, 0.000000, 0.000000, 0.000000)
message("Alternate Solution Objective Value = : ",value)
decisionVector <- c(rep(0,15),0.5801905,2.414519 ,1.627169 ,1.797931,0.5801905, 0.000000,
    0.000000, 0.000000, 0.000000, 0.000000)
index <- 0
for (irow in 1:5){
  for (jcol in 1:5){
    index <- index+1
    decisionMatrix[irow,jcol] <- decisionVector[index]
  }
}
print(decisionMatrix, byrow=TRUE)
constCheck <- cmatrix%*%c(rep(0,15),0.5801905,2.414519 ,1.627169 ,1.797931,0.5801905,
    0.000000, 0.000000, 0.000000, 0.000000, 0.000000)
message("Constraint Equation Values = :")
print(cbind(constCheck,cdirection,crhs))
# value <- t(cost)%*%c(rep(0,15),0.5801905,2.414519 ,1.627169 ,1.797931,0.5801905,
    0.000000, 0.000000, 0.000000, 0.000000, 0.000000)
# print(value)
# # check constrainst for alternates
# check <- cmatrix%*%c(rep(0,15),1.160381,2.414519 ,1.627169 ,1.797931,0.000000, 0.000000,
    0.000000, 0.000000, 0.000000, 0.000000)
# print(check)
# check <- cmatrix%*%c(rep(0,15),0.5801905,2.414519 ,1.627169 ,1.797931,0.5801905,
    0.000000, 0.000000, 0.000000, 0.000000, 0.000000)
# print(check)
```

10.4. Readings

10.5. Exercises

11. Supply Allocation under Uncertainty - Multi-Stage Approach

Introduction of uncertainty into an allocation problem creates a stochastic programming situation. There are a variety of methods to address stochastic programming, the next two lessons consider a case where the problem is linear enough for a linear programming solution. Using LP for stochastic problems results in a large increase in the number of variables and constraints to account for alternative scenarios or in the use of chance constraints. These two approaches are explored in this lesson and the next – in this lesson we will examine two-stage linear programming by means of a simple example. The example will be repeated in the next lesson using chance constraints.

11.1. Multi-Stage LP Allocation Example

A water master for a region can divert water from an unregulated stream to three consumers: a municipality, an industrial concern, and an agricultural concern. A schematic of the system is shown in Figure 25.

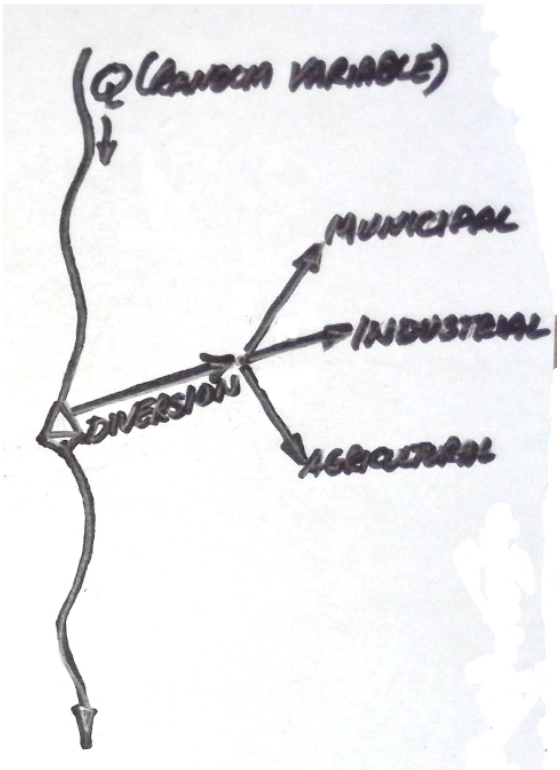


Figure 21. Schematic of water supply allocation problem.

The industrial and agricultural concern are expanding and would like to know how much water they can expect. If the water supply is insufficient, the expansion plans will be curtailed.

The unregulated streamflow is a random variable, and the goal of the water master is to maximize benefit to the water supply authority.

Let T_i represent the contracted quantity of water the supplier will deliver to each user i. If the water is delivered, the resulting net benefits to the local economy are estimated to be NB_i. However, if the water is not delivered, water must be obtained from an alternate source, or demand must be curtailed by rationing of the municipal users, reduced production by industrial users, or reduced irrigation by the agricultural users. Such deficits reduce the net benefits to user i by a cost C_i per unit of water not supplied. Each user can provide the water master with an estimate of the maximum amount of water they can use, T_i^{max}.

Q, the total water available, is a random variable described by some probability distribution. The water master's goal is to maximize the expected net benefit to the region by allocating water to the three users.

11.2. Linear Programming Model Framework

Let the deficit for a delivery be denoted by the variable $D_{i,j}$, where the i subscript refers to a particular user, and the j subscript refers to the particular random supply scenario.

The water masters allocation problem is then to maximize expected benefit. Recall an expected cost (or benefit) is the product of the probability of an outcome and the value (cost) of the outcome.

In this example the benefits of the deliveries is assured (probability = 1), but the costs of the deficits are stochastic, hence the benefit function is

$$NB = NB_1 \cdot T_1 + NB_2 \cdot T_2 + NB_3 \cdot T_3 - E(C_1 \cdot D_{1,Q} + C_2 \cdot D_{2,Q} + C_3 \cdot D_{3,Q}) \quad (40)$$

The constraints to be applied are

$$T_1 + T_2 + T_3 - [D_{1,Q} - D_{2,Q} - D_{3,Q}] \leq Q \quad (41)$$

and

$$0 \leq T_i - D_{i,Q} \leq T_i^{max} \quad (42)$$

11.2.1. Approximating the Random Variable

To approximate the value of the expectation, we can approximate the distribution of Q with a discrete distribution as in Figure 22. For a continuous distribution, some kind of discretization will be required to keep the LP manageable.

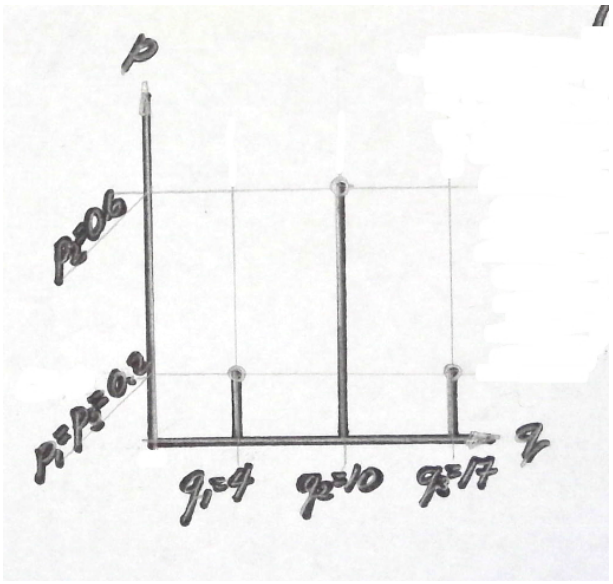


Figure 22. Streamflow Probability Distribution.

The approximation allows the estimation of the expected cost (of failed delivery) to be manifest in the objective function, and the supply availability in the constraint set. The model in this instance (3 customers, 3 possible supply rates) has a total of 12 decision variables; the three contracted deliveries and nine deficits (from the 3 possible supply rates).

The cost function becomes:

$$\begin{aligned} NB = \quad & NB_1 \cdot T_1 + NB_2 \cdot T_2 + NB_3 \cdot T_3 \quad -p_1(C_1 \cdot D_{1,1} + C_2 \cdot D_{2,1} + C_3 \cdot D_{3,1}) \\ & \qquad\qquad\qquad\qquad\qquad\qquad\quad -p_2(C_1 \cdot D_{1,2} + C_2 \cdot D_{2,2} + C_3 \cdot D_{3,2}) \\ & \qquad\qquad\qquad\qquad\qquad\qquad\quad -p_3(C_1 \cdot D_{1,3} + C_2 \cdot D_{2,3} + C_3 \cdot D_{3,3}) \end{aligned} \tag{43}$$

The constraint set becomes:

$$
\begin{array}{llllll}
T_1 + T_2 + T_3 & -D_{1,1} - D_{2,1} - D_{3,1} & & & \leq & q_1 \\
T_1 + T_2 + T_3 & & -D_{1,2} - D_{2,2} - D_{3,2} & & \leq & q_2 \\
T_1 + T_2 + T_3 & & & -D_{1,3} - D_{2,3} - D_{3,3} & \leq & q_3 \\
\hline
T_1 & -D_{1,1} & & & \leq & T_1^{max} \\
T_2 & - D_{2,1} & & & \leq & T_2^{max} \\
T_3 & - D_{3,1} & & & \leq & T_3^{max} \\
\hline
T_1 & & -D_{1,2} & & \leq & T_1^{max} \\
T_2 & & - D_{2,2} & & \leq & T_2^{max} \\
T_3 & & - D_{3,2} & & \leq & T_3^{max} \\
\hline
T_1 & & & -D_{1,3} & \leq & T_1^{max} \\
T_2 & & & - D_{2,3} & \leq & T_2^{max} \\
T_3 & & & - D_{3,3} & \leq & T_3^{max} \\
\hline
T_1 & -D_{1,1} & & & \geq & 0 \\
T_2 & - D_{2,1} & & & \geq & 0 \\
T_3 & - D_{3,1} & & & \geq & 0 \\
\hline
T_1 & & -D_{1,2} & & \geq & 0 \\
T_2 & & - D_{2,2} & & \geq & 0 \\
T_3 & & - D_{3,2} & & \geq & 0 \\
\hline
T_1 & & & -D_{1,3} & \geq & 0 \\
T_2 & & & - D_{2,3} & \geq & 0 \\
T_3 & & & - D_{3,3} & \geq & 0
\end{array}
\tag{44}
$$

The solution to this Linear Program would provide the water master with the optimal allocation of contracted (promised delivery) as well as an estimate (implicit in the solution) of the resource allocations to make when supply is insufficient.

11.2.2. Technological and Probability Values

Assume for this example that Table 2 contains the benefit and cost information for the three uses and the maximum value that each user can assimilate.

Table 2. Water Resource Allocation Technological Values.

Use	Index	T_i^{max}	NB_i	C_i
Municipal	1	2	100	250
Industrial	2	3	50	75
Agricultural	3	5	30	60

Table 3 contains the streamflow distribution for the region.

The values in these two tables are inserted into their representative locations in the

Table 3. Streamflow Probability Distribution.

Flow Index, j	q_j	p_j
1	4	0.20
2	10	0.60
3	17	0.20

linear program structure (cost function and constraint set). Once these values are supplied, then we have the necessary set-up for a Linear Program and can apply our tools to its solution. So the linear program to be solved is depicted in Figure 23

11.3. Linear Programming Model Solution

To solve the LP we need to translate the framework from Figure 23 into a structure for our solver tool (`lpSolve`) to process. The process to solve the problem in **R** is to build the cost function weights, then the constraint matrix coefficients, then the right hand side, and finally the sense of the constraints. Once these are built, the objects can be sent to the program for a solution.

For this example, the cost function unit values are hand-entered and the probabilities are stored in a separate vector. The coefficient for the cost function is created by element-by-element multiplication of the two vectors. The result is what will be sent to the LP solver algorithm.

The coefficient matrix is constructed row-wise, again by hand. The right-hand-side matrix is also hand constructed. The sense of the inequalities is constructed semi-automatically by taking advantage of how the tableau is structured. Finally all the elements are sent to the algorithm as shown in Listing 18 builds and solves the example problem.

$$
\begin{array}{lll}
\textit{maximize} & 100T_1 + 50T_2 + 30T_3 & -0.2(250D_{1,1} + 75D_{2,1} + 60D_{3,1}) \\
 & & -0.6(250D_{1,2} + 75D_{2,2} + 60D_{3,2}) \\
 & & -0.2(250D_{1,3} + 75D_{2,3} + 60D_{3,3})
\end{array}
$$

subject to

$$
\begin{array}{lllllll}
T_1 + T_2 + T_3 & -D_{1,1} - D_{2,1} - D_{3,1} & & & & \leq & 4 \\
T_1 + T_2 + T_3 & & -D_{1,2} - D_{2,2} - D_{3,2} & & & \leq & 10 \\
T_1 + T_2 + T_3 & & & -D_{1,3} - D_{2,3} - D_{3,3} & & \leq & 14 \\
\hline
T_1 & -D_{1,1} & & & & \leq & 2 \\
T_2 & - D_{2,1} & & & & \leq & 3 \\
T_3 & - D_{3,1} & & & & \leq & 5 \\
\hline
T_1 & & -D_{1,2} & & & \leq & 2 \\
T_2 & & - D_{2,2} & & & \leq & 3 \\
T_3 & & - D_{3,2} & & & \leq & 5 \\
\hline
T_1 & & & -D_{1,3} & & \leq & 2 \\
T_2 & & & - D_{2,3} & & \leq & 3 \\
T_3 & & & - D_{3,3} & & \leq & 5 \\
\hline
T_1 & -D_{1,1} & & & & \geq & 0 \\
T_2 & - D_{2,1} & & & & \geq & 0 \\
T_3 & - D_{3,1} & & & & \geq & 0 \\
\hline
T_1 & & -D_{1,2} & & & \geq & 0 \\
T_2 & & - D_{2,2} & & & \geq & 0 \\
T_3 & & - D_{3,2} & & & \geq & 0 \\
\hline
T_1 & & & -D_{1,3} & & \geq & 0 \\
T_2 & & & - D_{2,3} & & \geq & 0 \\
T_3 & & & - D_{3,3} & & \geq & 0
\end{array}
$$

Figure 23. Linear Program Structure for Water Supply Allocation Problem.

Once the script run, the output then needs to be interpreted. Figure 24 is a screen capture of the results. The optimal solution is to contract deliveries to each user at the full amount they request. The expected net benefit is 425 benefit units. There is a 20% chance that users 2 and 3 will have deficits and the water master will incur costs. There is an 80% chance that all users will get their contracted water, and a 20% chance that there will be excess water available. If the system had storage the 20% excess cases could be used to help satisfy the one case when the system does not have enough input.

A useful collateral result is that when the water input is too small to satisfy all the contracts, the solution also suggests how to make the allocations in the limited case – the municipal users are supplied their full amount, industry receives the remainder of the available water (and has a deficit of 1 unit), and agriculture is denied its entire request. The allocations would change if the unit benefits reflected the value of the water differently to the three users and/or the deficit costs changed.

Listing 18. R code to construct and solve the LP Model for Water Allocation problem.

```
# Build the LP model
library(lpSolve)
#
probabilities <- c(1,1,1,0.2,0.2,0.2,0.6,0.6,0.6,0.2,0.2,0.2);
unit_cost     <- c(100,50,30,-250,-75,-60,-250,-75,-60,-250,-75,-60);
prices <- probabilities*unit_cost
#
# constraint matrix
constraint <- array(0,dim=c(21,12));
# manual build
constraint[1,]  <- c(1,1,1,-1,-1,-1,0,0,0,0,0,0)
constraint[2,]  <- c(1,1,1,0,0,0,-1,-1,-1,0,0,0)
constraint[3,]  <- c(1,1,1,0,0,0,0,0,0,-1,-1,-1)
constraint[4,]  <- c(1,0,0,-1,0,0,0,0,0,0,0,0)
constraint[5,]  <- c(0,1,0,0,-1,0,0,0,0,0,0,0)
constraint[6,]  <- c(0,0,1,0,0,-1,0,0,0,0,0,0)
constraint[7,] <- c(1,0,0,0,0,0,-1,0,0,0,0,0)
constraint[8,] <- c(0,1,0,0,0,0,0,-1,0,0,0,0)
constraint[9,] <- c(0,0,1,0,0,0,0,0,-1,0,0,0)
constraint[10,] <- c(1,0,0,0,0,0,0,0,0,-1,0,0)
constraint[11,] <- c(0,1,0,0,0,0,0,0,0,0,-1,0)
constraint[12,] <- c(0,0,1,0,0,0,0,0,0,0,0,-1)
constraint[13,]  <- c(1,0,0,-1,0,0,0,0,0,0,0,0)
constraint[14,]  <- c(0,1,0,0,-1,0,0,0,0,0,0,0)
constraint[15,]  <- c(0,0,1,0,0,-1,0,0,0,0,0,0)
constraint[16,] <- c(1,0,0,0,0,0,-1,0,0,0,0,0)
constraint[17,] <- c(0,1,0,0,0,0,0,-1,0,0,0,0)
constraint[18,] <- c(0,0,1,0,0,0,0,0,-1,0,0,0)
constraint[19,] <- c(1,0,0,0,0,0,0,0,0,-1,0,0)
constraint[20,] <- c(0,1,0,0,0,0,0,0,0,0,-1,0)
constraint[21,] <- c(0,0,1,0,0,0,0,0,0,0,0,-1)
#
rhs <- numeric(0)
rhs <- c(4,10,17,2,3,5,2,3,5,2,3,5,0,0,0,0,0,0,0,0,0)
#
direction <- character(0)
for (i in 1:3){
  direction[i] <- "<="
}
for (i in 4:12){
  direction[i] <- "<="
}
for (i in 13:21){
  direction[i] <- ">="
}
print(prices)
print(cbind(constraint,direction,rhs))
# send to the LP solver
prod.sol <- lp("max", prices, constraint, direction, rhs, compute.sens = TRUE)
# #accessing to R output
print(prod.sol$objval)
print(prod.sol$solution)
```

11.4. Readings

11.5. Exercises

1. Determine the required penalty to ensure at least 1 unit of water is delivered to the agricultural sector.[5] Interpret the results of your solution.

2. Suppose there is a fourth user, the in-stream requirement to ensure health of the stream itself. If the in-stream requirement is 1 flow unit, how does the allocation change for the original problem and for the cost-incentivized agricultural adjustment? Document the new Linear Program to accommodate the 4th user.

[5]The objective function structure is in a penalty-function structure, and can be used to determine costs to assign to produce economically influenced results.

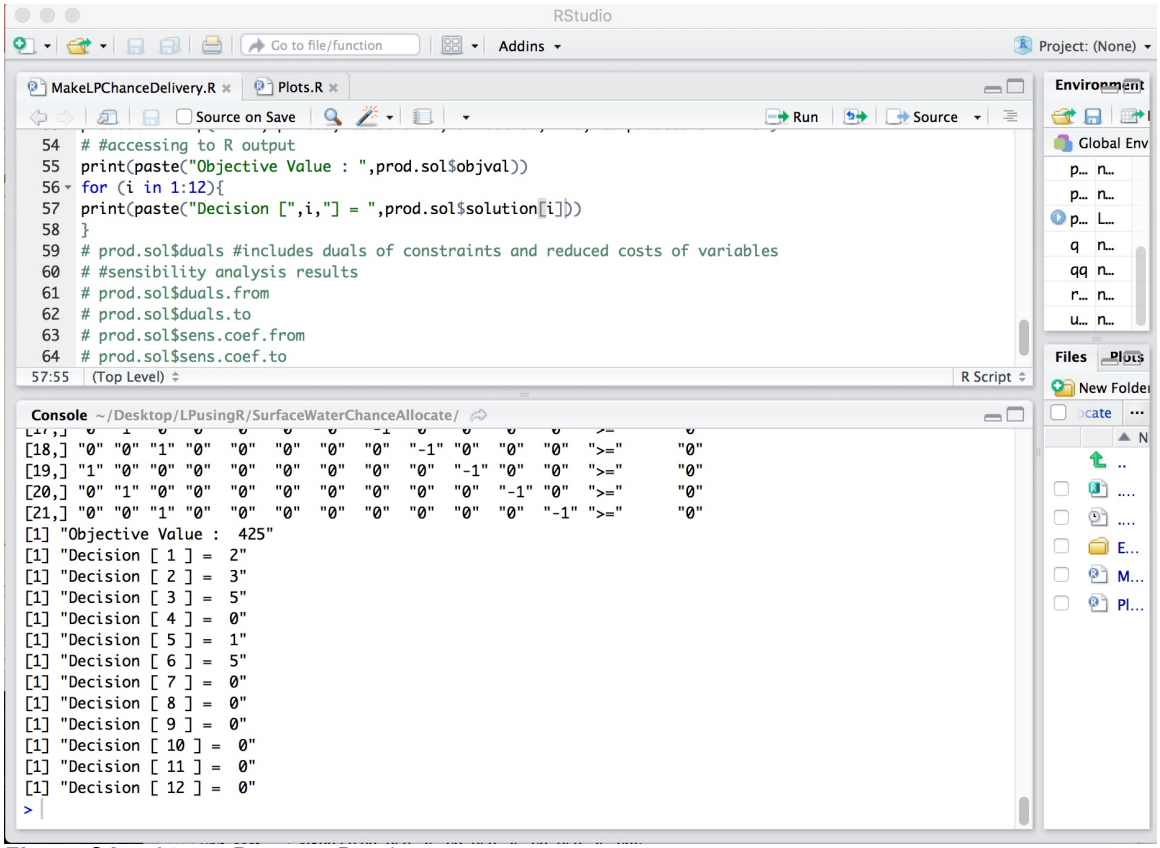


Figure 24. Linear Program Results.

12. Supply Allocation under Uncertainty - Chance-Constraint Approach

Adding uncertainty in the penalty-function approach of the previous chapter greatly increases the the size of an otherwise straightforward linear program. If the problem is reformulated, the increase in LP size can be mitigated using the concept of chance constraints.

12.1. Chance-Constraints

12.1.1. Example

A water master for a region can divert water from an unregulated stream to three consumers: a municipality, an industrial concern, and an agricultural concern. A schematic of the system is shown in Figure 25.

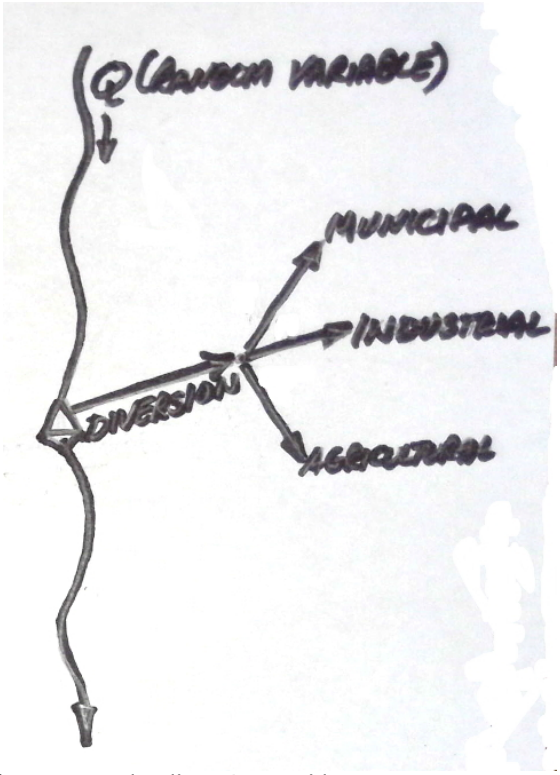


Figure 25. Schematic of water supply allocation problem.

The industrial and agricultural concern are expanding and would like to know how much water they can expect. If the water supply is insufficient, the expansion plans will be curtailed.

The unregulated streamflow is a random variable, and the goal of the water master is to maximize benefit to the water supply authority. Suppose, that instead of estimating

shortage cost coefficients C_i, the manager wants to specify the reliability with which targets are met.

The subtle change in perspective changes the LP considerably.

12.2. Approximating the Random Variable

The random variable is approximated in the same way in that the same distribution is used, however a cumulative distribution function $F(Q)$ which returns the cumulative probability of the random variable Q is used. Figure 26 depicts such a function.

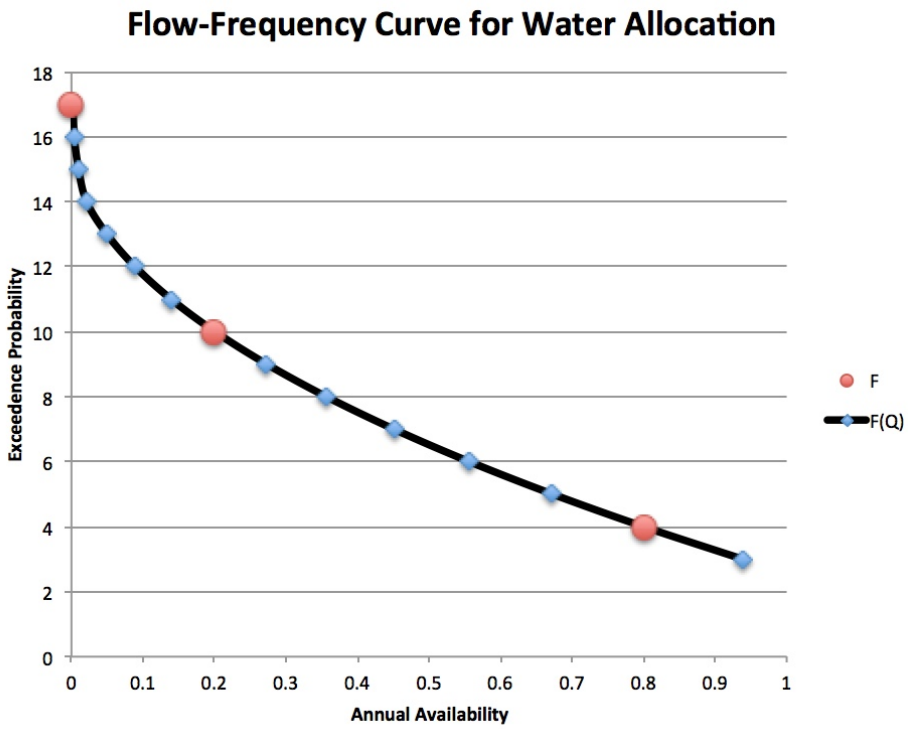


Figure 26. Flow-frequency curve.

The curve depicts the exceedance probability on the horizontal axis and the flow value on the vertical axis. For instance, the probability of observing a flow of 4 units or greater is 80%, whereas the probability of observing a a flow of 10 units or greater is only 20%.

12.3. The LP Framework and Solution

The LP is now somewhat simpler in that the entire problem is much more compact as shown in Figure 27

To illustrate, lets suppose we want to use the median (50%) flow reliability. From

$$
\begin{aligned}
&\textit{maximize} \quad 100T_1 + 50T_2 + 30T_3 \\
&\textit{subject to}
\end{aligned}
$$

$T_1 + T_2 + T_3$	$\leq$	$F^{-1}(Q)$
T_1	$\leq$	2
T_2	$\leq$	3
T_3	$\leq$	5

Figure 27. Linear Program Structure for Water Supply Allocation Problem.

the plot, the median value is about 7 flow units. Listing 19 shows the resulting LP model.

Listing 19. R code to construct and solve the LP Model for Water Allocation problem.

```
# Build the LP model
library(lpSolve)
# chance constained LP
prices <- c(100,50,30);
#
# constraint matrix
constraint <- array(0,dim=c(4,3));
# manual build
constraint[1,]  <- c(1,1,1)
constraint[2,]  <- c(1,0,0)
constraint[3,]  <- c(0,1,0)
constraint[4,]  <- c(0,0,1)

#
rhs <- numeric(0)
rhs <- c(7,2,3,5)
#
direction <- character(0)
for (i in 1:4){
  direction[i] <- "<="
}
print(prices)
print(cbind(constraint,direction,rhs))
# #
# # send to the LP solver
# #
prod.sol <- lp("max", prices, constraint, direction, rhs, compute.sens = TRUE)
# #accessing to R output
print(paste("Objective Value : ",prod.sol$objval))
for (i in 1:3){
print(paste("Decision [",i,"] = ",prod.sol$solution[i]))
}
```

When the model is run, the result is shown in Figure 28 and represents the optimal allocations for flows that can be expected 50% of the time.

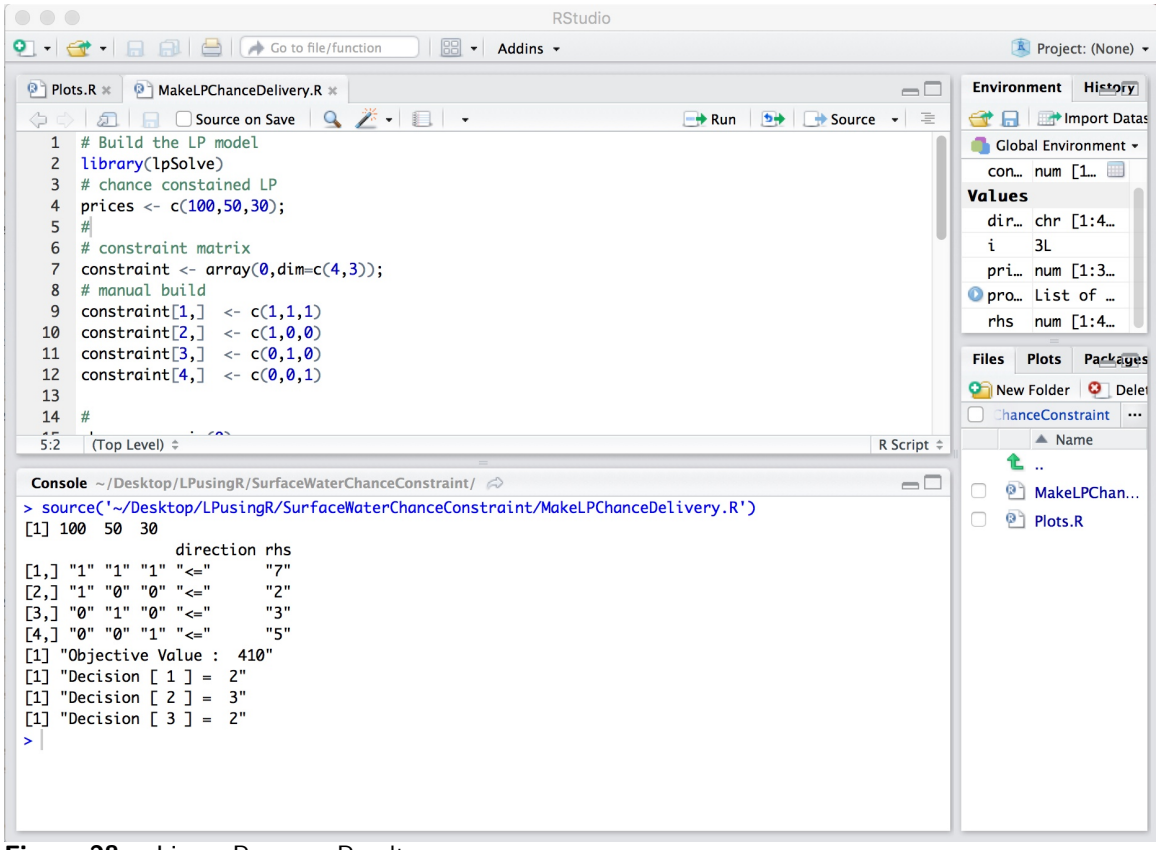


Figure 28. Linear Program Results.

12.4. Readings

12.5. Exercises

1. Assume the flow frequency curve is given by Figure 29. Determine the 80% reliable allocation for the water supply system.

2. If the prices are changed so that the agricultural and industrial benefit are the same, what are the 80% allocations? Be aware there may be multiple equivalent allocations, try to enumerate all the allocations.

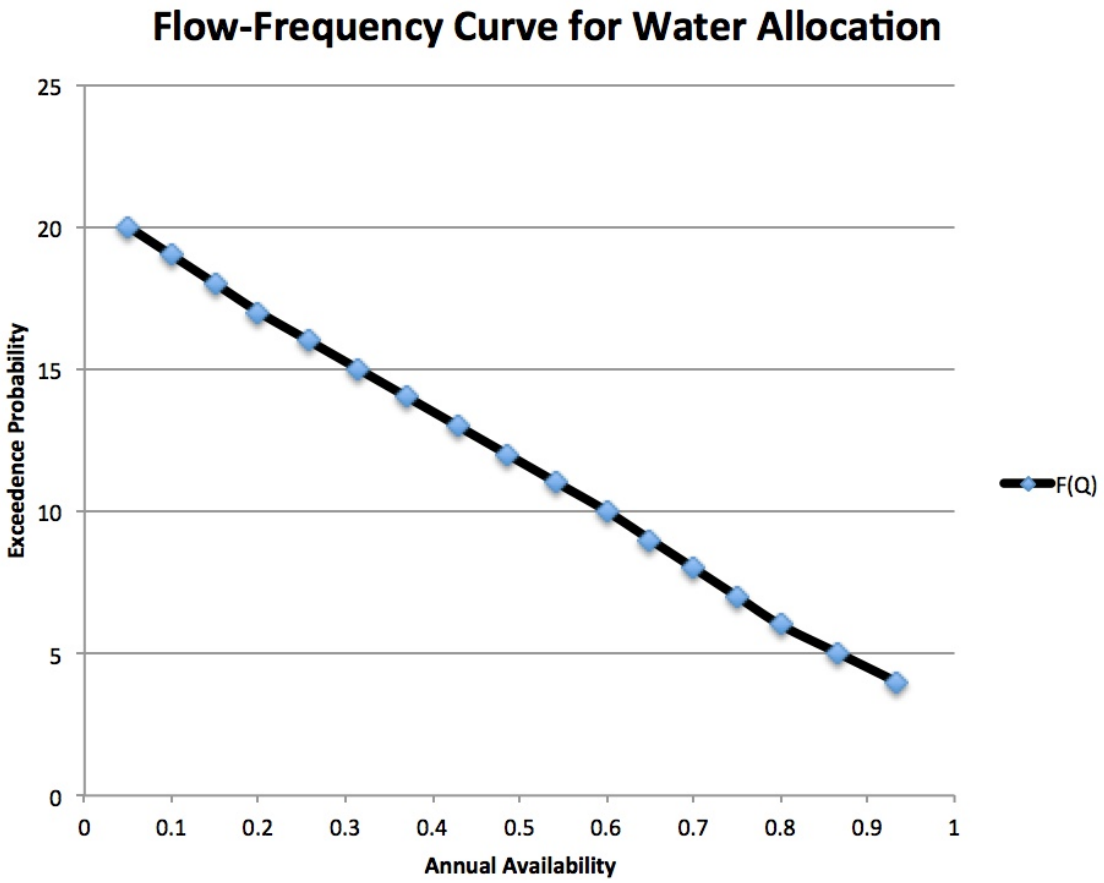


Figure 29. Flow Frequency Curve.

13. Waste Load Allocation

words

13.1. Water Quality Technological Functions

13.1.1. Example

13.2. Simulation-Optimization

13.2.1. Example

13.3. Topic 3

13.3.1. Example

13.4. Readings

13.5. Exercises

14. Constrained Non-Linear Programming

words

14.1. Topic 1

14.1.1. Example

14.2. Topic 2

14.2.1. Example

14.3. Topic 3

14.3.1. Example

14.4. Readings

14.5. Exercises

15. Solving constrained Non-Linear programs in R

Solving constrained non-linear optimization problems of any meaningful size is non-trivial. Sometimes just finding a feasible solution is difficult (especially for large-scale problems). This chapter presents the **R** package `nloptr` and its dependencies for solving non-linear probelms.

15.1. Packages

The first step is to install the package `nloptr` from the CRAN. Figure 30 shows the procedure to install the package using the package manager. Be sure to select "Install Dependencies" so that the manager installs related programs needed for the solver tools to work. The package itself is a front-end (interface) for several different packages contained in the CRAN, all of which need to be present on your computer for the tool to work.

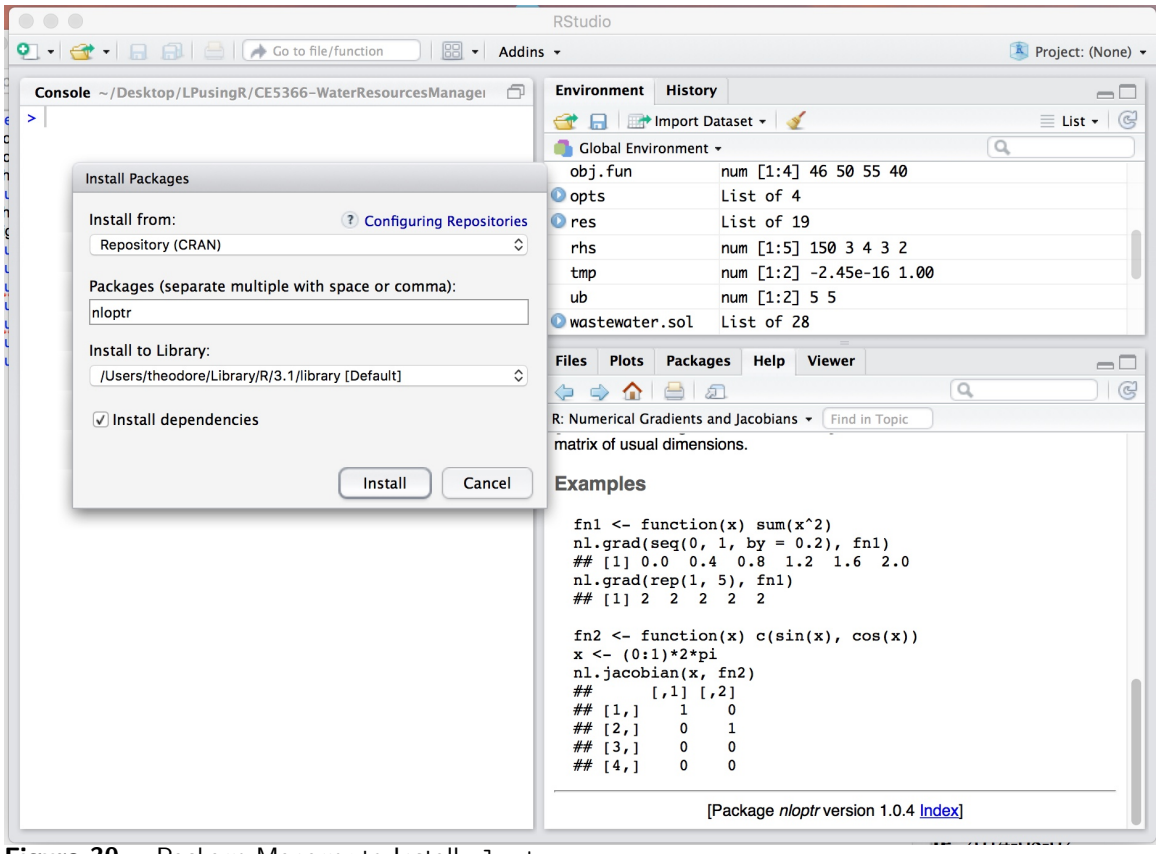


Figure 30. Package Manager to Install `nloptr`.

Upon successful install the rather simple message in Figure 31 is displayed that shows the install statistics and the location of the binaries on your computer.

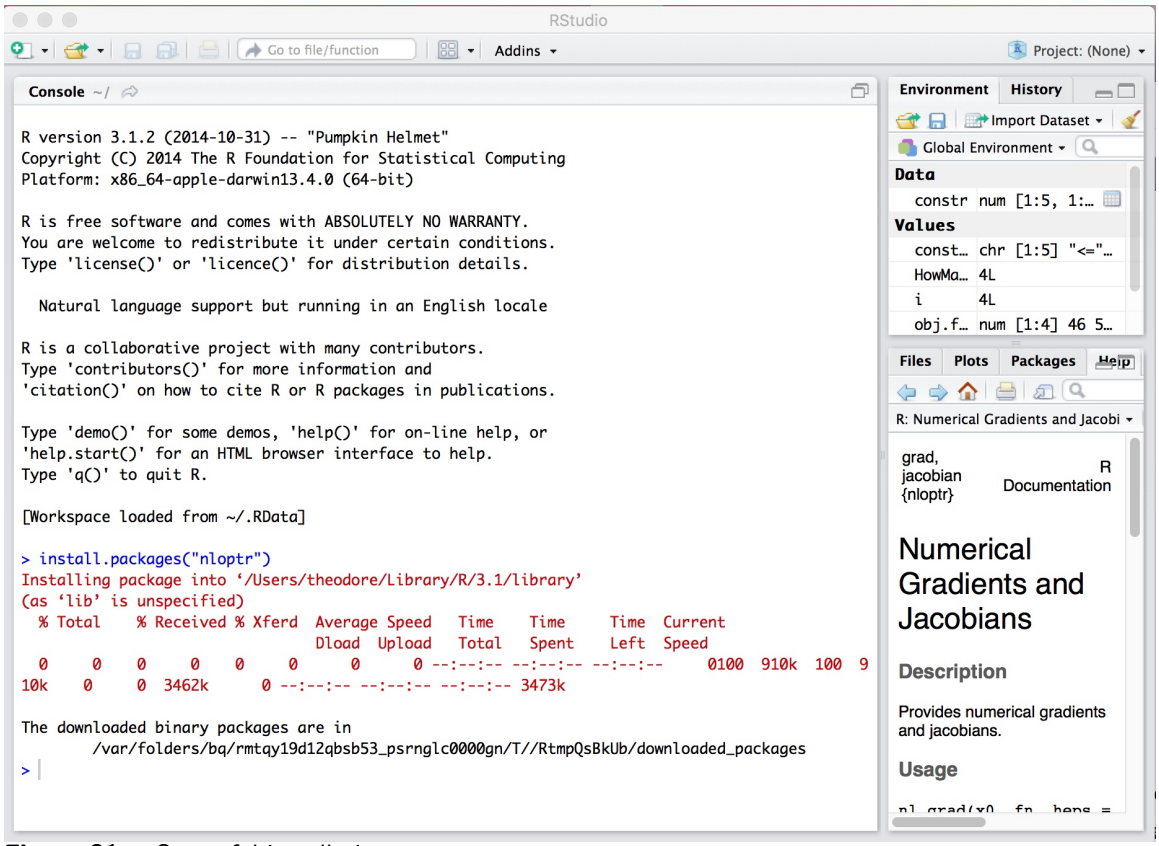


Figure 31. Succesful installation `nloptr`.

15.1.1. Example 1

The example in Figure 32 is used to test the install

$$
\begin{aligned}
&\textit{minimize} \quad x_1 \times x_4 \times (x_1 + x_2 + x_3) + x_3 \\
&\textit{subject to}
\end{aligned}
$$

$$
\begin{array}{rcl}
x_1 \times x_2 \times x_3 \times x_4 & \geq & 25 \\
\hline
x_1^2 + x_2^2 + x_3^2 + x_4^2 & = & 40 \\
\hline
x_1 & \geq & 1 \\
x_2 & \geq & 1 \\
x_3 & \geq & 1 \\
x_4 & \geq & 1 \\
\hline
x_1 & \leq & 5 \\
x_2 & \leq & 5 \\
x_3 & \leq & 5 \\
x_4 & \leq & 5
\end{array}
$$

Figure 32. Non-Linear Program Structure for Example 1 Problem.

Listing 20 is the example script lifted directly from the package documentation showing how the mathematics in Figure 32 is converted into the syntax for the non-linear program solver.

Listing 20. R code to construct and solve a NLP Model.

```
# Example problem, number 71 from the Hock-Schittkowsky test suite.
#
# \min_{x} x1*x4*(x1 + x2 + x3) + x3
# s.t.
#    x1*x2*x3*x4 >= 25
#    x1^2 + x2^2 + x3^2 + x4^2 = 40
#    1 <= x1,x2,x3,x4 <= 5
#
# we re-write the inequality as
#   25 - x1*x2*x3*x4 <= 0
#
# and the equality as
#   x1^2 + x2^2 + x3^2 + x4^2 - 40 = 0
#
# x0 = (1,5,5,1)
#
# optimal solution = (1.00000000, 4.74299963, 3.82114998, 1.37940829)

library('nloptr')
#
# f(x) = x1*x4*(x1 + x2 + x3) + x3
#
eval_f <- function( x ) {
  return( list( "objective" = x[1]*x[4]*(x[1] + x[2] + x[3]) + x[3],
                "gradient" = c( x[1] * x[4] + x[4] * (x[1] + x[2] + x[3]),
                                x[1] * x[4],
                                x[1] * x[4] + 1.0,
                                x[1] * (x[1] + x[2] + x[3]) ) ) )
}
# constraint functions
# inequalities
eval_g_ineq <- function( x ) {
  constr <- c( 25 - x[1] * x[2] * x[3] * x[4] )
  grad   <- c( -x[2]*x[3]*x[4],
               -x[1]*x[3]*x[4],
               -x[1]*x[2]*x[4],
               -x[1]*x[2]*x[3] )
  return( list( "constraints"=constr, "jacobian"=grad ) )
}
# equalities
eval_g_eq <- function( x ) {
  constr <- c( x[1]^2 + x[2]^2 + x[3]^2 + x[4]^2 - 40 )
  grad   <- c(  2.0*x[1],
                2.0*x[2],
                2.0*x[3],
                2.0*x[4] )
  return( list( "constraints"=constr, "jacobian"=grad ) )
}
# initial values
x0 <- c( 1, 5, 5, 1 )
# lower and upper bounds of control
lb <- c( 1, 1, 1, 1 )
ub <- c( 5, 5, 5, 5 )
local_opts <- list( "algorithm" = "NLOPT_LD_MMA",
                    "xtol_rel"  = 1.0e-7 )
opts <- list( "algorithm" = "NLOPT_LD_AUGLAG",
              "xtol_rel"  = 1.0e-7,
              "maxeval"   = 1000,
              "local_opts" = local_opts )
res <- nloptr( x0=x0,
               eval_f=eval_f,

               lb=lb,
               ub=ub,
               eval_g_ineq=eval_g_ineq,
               eval_g_eq=eval_g_eq,
               opts=opts)

print( res )
```

An important observation in the listing is that gradients and Jacobians are supplied as analytical functions of the input variables. Later we will attempt to replace these with numerical approximations (to generalize the problem solving tool somewhat).

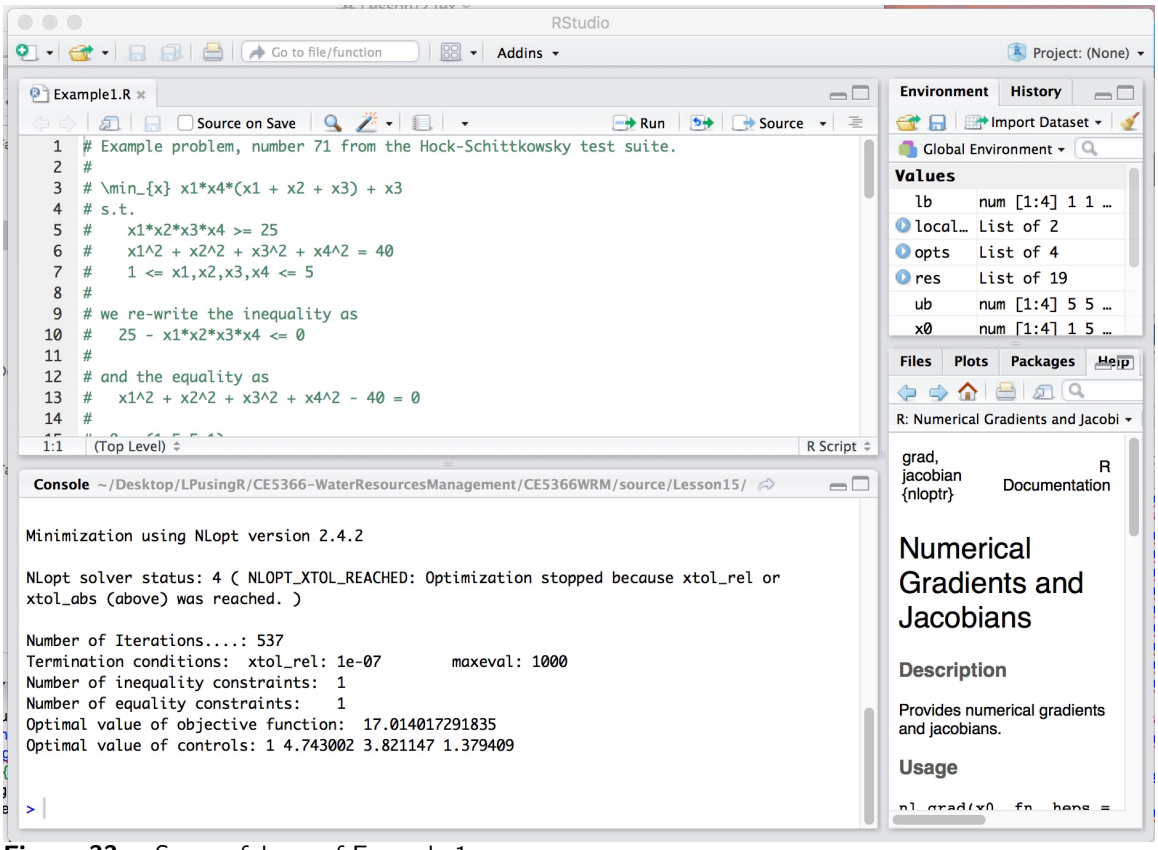


Figure 33. Successful run of Example 1.

Figure 33 is the RStudio output when the example is run. With the successful run, we conclude that the package is installed and runs as anticipated.

15.1.2. Example 2

Another test example from pg 209 of (**?**) is

$$
\begin{aligned}
&\textit{minimize} \quad x_1 \times x_2^2 \\
&\textit{subject to}
\end{aligned}
$$

$$
\begin{array}{rcl}
2 - x_1^2 \times -x_2^2 & \geq & 0 \\
\hline
x_1 & \geq & -5 \\
x_2 & \geq & -5 \\
\hline
x_1 & \leq & 5 \\
x_2 & \leq & 5
\end{array}
$$

Figure 34. Non-Linear Program Structure for Example 2 Problem.

Listing 21 is the script to solve this example. Observe there is no inequality constraint, so that part of the solver call is suppressed.

Listing 21. R code to construct and solve a NLP Model.

```
# Example problem, pg 209 Gill, Murray, and Wright
#
# \min_{x} x1*(x2^2)
#
# s.t.
#    2 - x1^2 - x2^2 >= 0
#
# we re-write the inequality as
#   x1^2 + x2^2 - 2 <= 0
#
#
# x0 = (-1,-1)
#
# optimal solution = (-0.81650,-1.1547)

library('nloptr')
#
# f(x) = x1*x4*(x1 + x2 + x3) + x3
#
eval_f <- function( x ) {
  return( list( "objective" = x[1] * x[2]^2,
                "gradient" = c( x[2]^2,2*x[1]*x[2])
              )
          )
}
# constraint functions
# inequalities
eval_g_ineq <- function( x ) {
  constr <- c(x[1]^2 + x[2]^2 -2 )
  grad   <- c( 2*x[1],
               2*x[2] )
  return( list( "constraints"=constr, "jacobian"=grad ) )
}
# equalities
# eval_g_eq <- function( x ) {
#   constr <- c( x[1]^2 + x[2]^2 + x[3]^2 + x[4]^2 - 40 )
#   grad   <- c(  2.0*x[1],
#                 2.0*x[2],
#                 2.0*x[3],
#                 2.0*x[4] )
#   return( list( "constraints"=constr, "jacobian"=grad ) )
# }
# initial values
x0 <- c( -1,-1 )
# lower and upper bounds of control
lb <- c( -5, -5 )
ub <- c( 5, 5 )
local_opts <- list( "algorithm" = "NLOPT_LD_MMA",
                    "xtol_rel"  = 1.0e-7 )
opts <- list( "algorithm" = "NLOPT_LD_AUGLAG",
              "xtol_rel"  = 1.0e-7,
              "maxeval"   = 1000,
              "local_opts" = local_opts )
res <- nloptr( x0=x0,
               eval_f=eval_f,

               lb=lb,
               ub=ub,
               eval_g_ineq=eval_g_ineq,
#               eval_g_eq=eval_g_eq,
               opts=opts)

print( res )
```

Figure 35 shows a successful run (and the optimal solution) for the example problem. So now we have a rudimentary understanding of the solver's syntax.

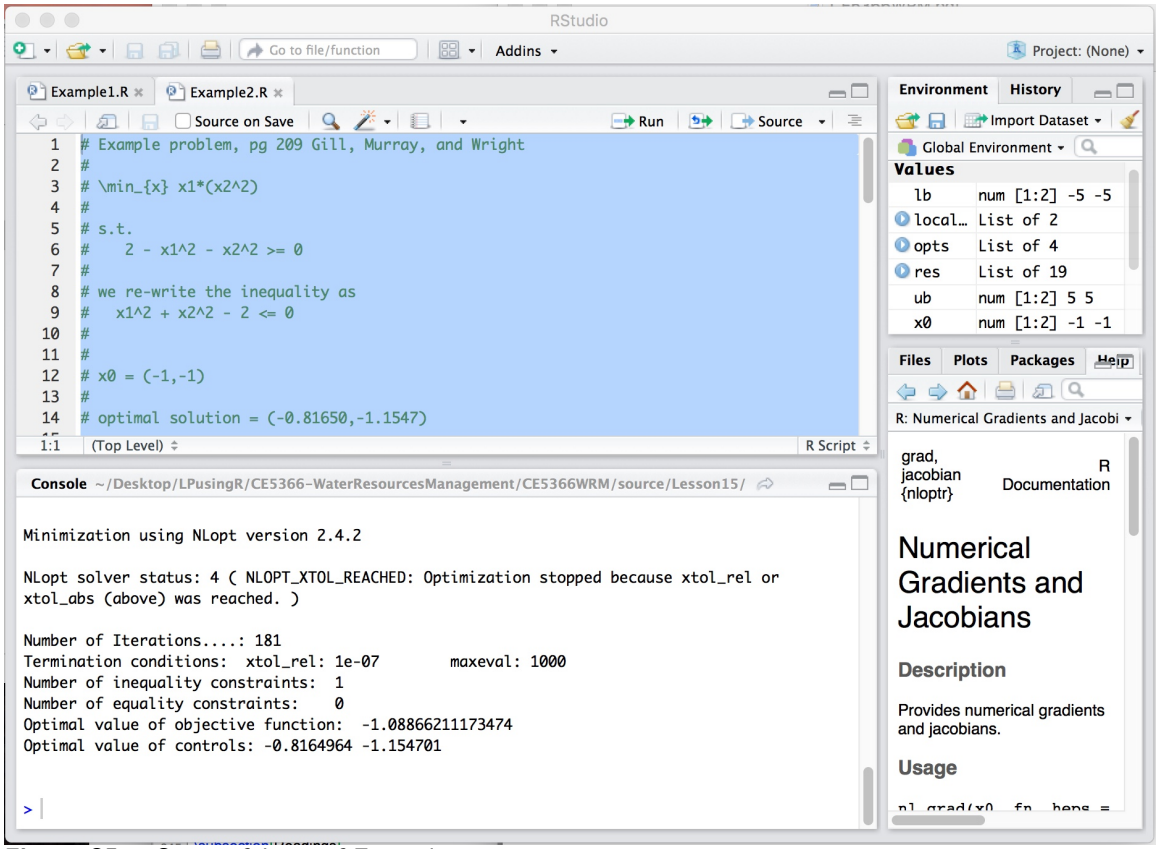


Figure 35. Successful run of Example 2.

15.2. Topic 2 : An Equitable Price Strategy

15.2.1. Example: Bus Transit Price Structure

An transit company has determined the average cost to provide service on a two-route bus system is \$1.20 per customer. The company wants to establish a fare schedule to account for the trip length of the rider and the ridership on each route. The route with the greater ridership may pay a fare that is less than the fare paid by users of the route with lower ridership. Similarly, users of the longer route may have to pay a fare greater than users on the shorter route.

Figure 36 depicts the layout of the bus system. Table 4 lists information regarding the two routes.

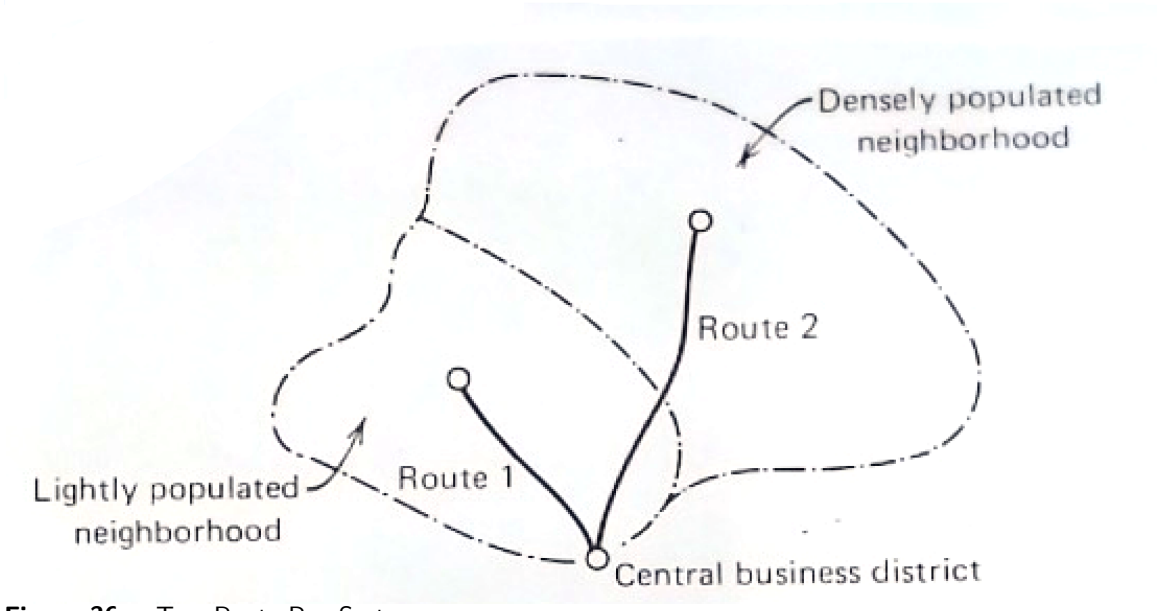


Figure 36. Two-Route Bus System.

Table 4. Transit System Statistics.

Route i	Ridership Average Trip Length L_i	n_i, Average daily ridership
1	10 miles	600
2	15 miles	1,400

One metric of equity is that an equitable system would minimize the variance of the travel cost among its customers. This measure of effectiveness (a weighted variance equation) is

$$V = \frac{1}{N} \sum_{i=1}^{N} n_i (r_i - \bar{r})^2 \tag{45}$$

where V is the variance of the travel cost; r_i is the fare box revenue paid by each customer on route i; $\bar{r}$ is the average fare box revenue paid by each customer [6] ; and N is the total number of customers.

Using the supplied information formulate a mathematical model to determine an equitable bus fare schedule that charges each customer on a cost per passenger-mile basis, then find an optimal solution using constrained (or unconstrained) non-linear programming in **R**.

Analysis Let the price to be charged on each route be the decision variable equal to the cost per passenger-mile on each route.

$p_1 =$ unit fare in cost per passenger-mile on route 1
$p_2 =$ unit fare in cost per passenger-mile on route 2

The merit function is the weighted variance of travel cost among the customers:

$V = \frac{n_1}{N}(r_1 - \bar{r})^2 + \frac{n_2}{N}(r_2 - \bar{r})^2$

The revenue from a unit fare for each route is:

$r_1 = p_1 \times L_1 = 10p_1$
$r_2 = p_2 \times L_2 = 15p_2$

Substituting these values into the merit function yields:

$V = \frac{600}{2000}(10p_1 - 1.20)^2 + \frac{1400}{2000}(15p_2 - 1.20)^2$

or

$V = 0.3(10p_1 - 1.20)^2 + 0.7(15p_2 - 1.20)^2$

The bus company is assumed to be a not-for-profit operation so that it is constrained to have revenue equal costs. This constraint is expressed as:

$p_1(n_1L_1) + p_2(n_2L_2) = \bar{r}N$

or

$p_1(600\dot{1}0) + p_2(1400\dot{1}5) = \$1.20\dot{2}000$

or

[6] $\bar{r} = \frac{1}{N}\sum_{i=1}^{N} n_i r_i$

$6p_1 + 21p_2 = \$2.40$

Problem Set-Up
Using the constraints and performance criteria we can construct the optimization problem as

$minimize \quad V = 0.3(10p_1 - 1.20)^2 + 0.7(15p_2 - 1.20)^2$
subject to

$6p_1$	$+ 21p_2$	$=$	$\$2.40$
p_1		$\geq$	0
	p_2	$\geq$	0

Solution Listing **??** is the **R** script that can be used to find the optimal solution to the problem.

15.3. Topic 3 : Water Resources Allocation/Operation Problem

15.3.1. Example: Water Delivery Price Structure

15.4. Topic 4: Water Quality Allocation Problem

15.4.1. Example: Reactor-Tank System

Consider a biological reactor system using the two-vessel feedback schematic in Figure 37. The system uses influent substrate, S, as an energy source (think food) to grow biomass, X, in the first vessel. The mixed liquor is then transferred to a clarifier tank where the liquid is decanted and the biomass is concentrated. The decanted liquid is returned to the environment (you can visualize the diagram as a water treatment system, where the substrate is some pollutant). The concentrated biomass is recycled back to the reactor tank as a way to ensure that the biomass population is sufficient for substrate removal.

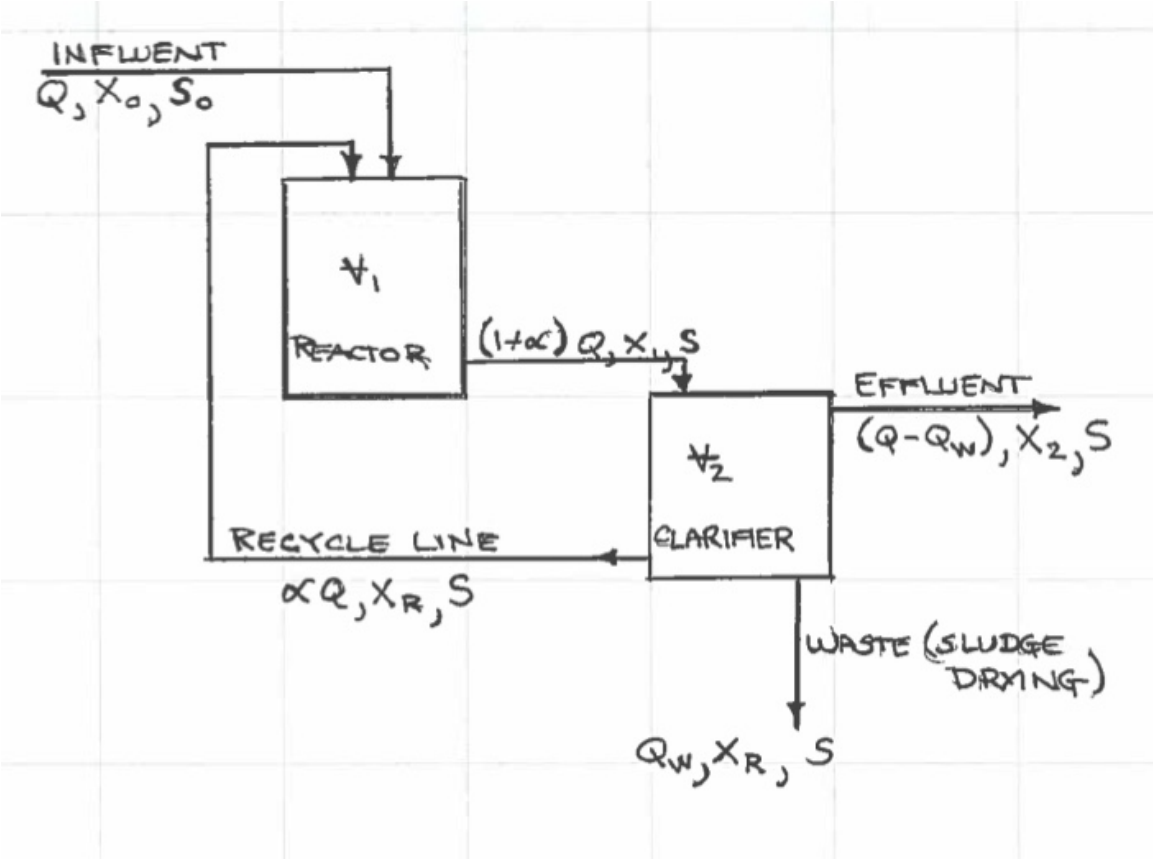


Figure 37. Feedback Biological Reactor Schematic.

In practice a portion of the concentrated biomass is wasted from the system to maintain constant system volumes (and control sludge age and other important features exogenous to the example).

The cost per vessel is a function of the vessel volume in gallons and is expressed as

$$C = \$25.119 \times V^{0.6} \tag{46}$$

The equilibrium inflow conditions are $Q = 1 \times 10^6$ gallons per day, $S_0 = 200$ ppm, and $X_0 = 0$ ppm.[7]

The system design guidelines call for the clarifier overflow loading rate to be $Q_O = 800$ gal/day/ft^2, the clarifier depth h to be 8 feet, and the effluent substrate concentration to be $S = 20$ ppm.

Past design experience with similar systems suggests a cell growth rate of $\mu = 1.876$ days^{-1} and a biosolids yield coefficient $Y = \frac{X}{S} = 0.449$.

Using this information design a minimum cost reactor system to satisfy the desired output water quality.

Analysis
The reactor mass balance is used to establish various constraints for the problem.

The biosolids mass balance is:

$$V_1 \frac{dX_1}{dt} = QX_0 + \alpha QX_R - (1+\alpha)QX_1 + \mu X_1 V_1 \tag{47}$$

Now divide by V_1 to obtain:

$$\frac{dX_1}{dt} = \frac{Q}{V_1}X_0 + \alpha\frac{Q}{V_1}X_R - (1+\alpha)\frac{Q}{V_1}X_1 + \mu X_1 \tag{48}$$

Introduce the hydraulic retention time of the reactor as $T_1 = \frac{V_1}{Q}$, and substitute this value into the equation to obtain:

$$\frac{dX_1}{dt} = \frac{X_0}{T_1} + \alpha\frac{X_R}{T_1} - (1+\alpha)\frac{X_1}{T_1} + \mu X_1 \tag{49}$$

The substrate (pollutant) mass balance is

$$V_1 \frac{dS}{dt} = QS_0 + \alpha QS - (1+\alpha)QS - \frac{\mu}{Y}X_1 V_1 \tag{50}$$

Now divide by V_1 to obtain:

$$\frac{dS}{dt} = \frac{Q}{V_1}S_0 + \alpha\frac{Q}{V_1}S - (1+\alpha)\frac{Q}{V_1}S - \frac{\mu}{Y}X_1 \tag{51}$$

Again substitute the hydraulic retention time to obtain:

$$\frac{dS}{dt} = \frac{S_0}{T_1} + \alpha\frac{S}{T_1} - (1+\alpha)\frac{S}{T_1} - \frac{\mu}{Y}X_1 \tag{52}$$

Repeat the mass balance analysis for the clarifier, in the clarifier we assume there are no further reactions so the substrate concentrations that enter the clarifier leave

[7] S is the concentration of the substrate or pollutant; X is the concentration of biosolids that have been grown from the substrate.

undiminished, so we only consider the biosolids within the clarifier. The clairifier biosolids mass balance is:

$$V_2 \frac{dX_2}{dt} = (1+\alpha)QX_1 - (Q - Q_w)X_2 - (\alpha Q + Q_w)X_R \tag{53}$$

At equilibrium, $\frac{dX_1}{dt} = 0; V_2 \frac{dX_2}{dt} = 0; \frac{dS}{dt} = 0;$

Apply this condition to the three equations, first the reactor biosolids balance,

$$\frac{X_0}{T_1} + \alpha \frac{X_R}{T_1} - (1+\alpha)\frac{X_1}{T_1} + \mu X_1 = 0 \tag{54}$$

Multiply by the hydraulic retention time to obtain:

$$0 = X_0 + \alpha X_R - (1+\alpha)X_1 + \mu X_1 T_1 = 0 \tag{55}$$

Next the substrate balance,

$$\frac{S_0}{T_1} + \alpha \frac{S}{T_1} - (1+\alpha)\frac{S}{T_1} - \frac{\mu}{Y} X_1 = 0 \tag{56}$$

Again multiply by the hydraulic retention time to obtain:

$$S_0 + \alpha S - (1+\alpha)S - \frac{\mu}{Y} X_1 T_1 = 0 \tag{57}$$

Combine the terms involving S to obtain

$$X_1 T_1 = \frac{Y}{\mu}(S_0 - S) \tag{58}$$

Next the clarifier biosolids balance:

$$(1+\alpha)QX_1 - (Q - Q_w)X_2 - (\alpha Q + Q_w)X_R = 0 \tag{59}$$

Expand and divide by Q_w to obtain:

$$\frac{QX_1 + \alpha QX_1 - QX_2 + Q_w X_2 - \alpha QX_R - Q_w X_R}{Q_w} = 0 \tag{60}$$

Factor out and isolate $X_2 - X_R$ to obtain:

$$\frac{Q}{Q_w}[X_1 + \alpha X_1 - X_2 - \alpha X_R] = X_R - X_2 \tag{61}$$

Refactor to obtain

$$\frac{(1+\alpha)X_1 - X_2 - \alpha X_R}{X_R - X_2} = \frac{Q_w}{Q} \tag{62}$$

Equations 55,58 , and 62 are now employed to build a meaningful constraint set from the performance guidelines.

Multiply Equation 55 by the hydraulic retention time to obtain:

$$0 = X_0 T_1 + \alpha X_R T_1 - (1+\alpha) X_1 T_1 + \mu X_1 T_1 \cdot T_1 = 0 \tag{63}$$

Now substitute Equation 58 into the result above to obtain:

$$0 = X_0 T_1 + \alpha X_R T_1 - (1+\alpha)\frac{Y}{\mu}(S_0 - S) + Y(S_0 - S)T_1 = 0 \tag{64}$$

Now refactor the result to obtain:

$$[X_0 + \alpha X_R + Y(S_0 - S)]T_1 - \alpha\frac{Y}{\mu}(S_0 - S) = \frac{Y}{\mu}(S_0 - S) \tag{65}$$

Again, rewrite Equation 55 as

$$X_0 + \mu X_1 T_1 = (1+\alpha)X_1 - \alpha X_R \tag{66}$$

Substitute into Equation 62 to obtain:

$$\frac{X_0 + \mu X_1 T_1 - X_2}{X_R - X_2} = \frac{Q_w}{Q} \tag{67}$$

Now substitute Equation 58 to obtain:

$$\frac{X_0 + Y(S_0 - S) - X_2}{X_R - X_2} = \frac{Q_w}{Q} \tag{68}$$

Here we have removed any dependency on T_1 and α which are the design variables, so the only mass balance constraint is Equation 65.

Now consider the relationship of clarifier and reactor volumes. The design overflow rate actually sizes the clarifier – its volume must be

$$V_2 = \frac{(1+\alpha)Q}{Q_0} \cdot h \tag{69}$$

The hydraulic retention time (a design variable) provides the required size of the reactor as

$$V_1 = QT_1 \tag{70}$$

Now we have enough description to construct the optimization problem.

Problem Set-Up
Using the constraints and performance criteria we can construct the optimization problem as

$$\begin{aligned} &\textit{minimize } C \;=\; 25.12(Q)^{0.6}(T_1)^{0.6} + 25.12(\tfrac{Q\cdot h}{Q_0})^{0.6}(1+\alpha)^{0.6} \\ &\textit{subject to} \end{aligned}$$

$$\begin{array}{rcl} [X_0 + \alpha X_R + Y(S_0 - S)]T_1 - \alpha\frac{Y}{\mu}(S_0 - S) & = & \frac{Y}{\mu}(S_0 - S) \\ T_1 & \geq & 0 \\ \alpha & \geq & 0 \end{array}$$

Now for the particular problem we will substitute in numerical values from the known quantities.

$Q = 1 \times 10^6$ thus

$25.12(\frac{Q\cdot h}{Q_0})^{0.6} = 25.12(\frac{1\times 10^6 \cdot 8}{800})^{0.6} = 6309.9,$

and

$25.12(Q)^{0.6} = 25.12(1 \times 10^6)^{0.6} = 100,010.$

The remaining values are $X_0 = 0$, $X_R = 10,000$, $S_0 = 200$, and $S = 20$.

The resulting non-linear program in the design variables T_1 and α is

$$\begin{aligned} &\textit{minimize } C \;=\; 100,010T_1^{0.6} + 6309.9(1+\alpha)^{0.6} \\ &\textit{subject to} \end{aligned}$$

$$\begin{array}{rcl} 10,000\alpha \cdot T_1 + 80.82T_1 - 43.081\alpha & = & 43.081 \\ T_1 & \geq & 0 \\ \alpha & \geq & 0 \end{array}$$

15.5. Topic 5 : Extending the Toolkit – Using numerical derivatives to construct gradients and Jacobians

The approach so far involved defining objective function(s) and constraint function(s), however in both components analytical derivatives (and Jacobians) were employed.[8] For small scale problems where such analysis is straightforward, this is a fine approach – but for large scale problems, especially when the functions themselves are complicated and are not polynomials (hence the derivatives may themselves be complex even if we could find them) we may prefer to numerical derivatives – which leaves us (the engineers) only with the task of defining the objective and the constraint functions (and we hope numerical approximations of the various derivatives are well behaved).

To use numerical approximations the package `nloptr` contains a pair of functions to compute the gradient of a scalar valued function of vector arguments (the objective function) and compute the Jacobian of a vector valued function of vector arguments (the constraints) using the function names `nl.grad` and `nl.jacobian` respectively.

15.5.1. Example 2A Numerical Gradient and Jacobian

The test example from pg 209 of (**?**) is repeated, but instead of analytical derivatives, the two functions are used to construct the optimization call. Figure 38 states the problem (yes it is identical to the above problem). The fundamental change in the

$$
\begin{array}{l}
\textit{minimize} \quad x_1 \times x_2^2 \\
\textit{subject to}
\end{array}
$$

$2 - x_1^2 \times -x_2^2$	$\geq$	0
x_1	$\geq$	-5
x_2	$\geq$	-5
x_1	$\leq$	5
x_2	$\leq$	5

Figure 38. Non-Linear Program Structure for Example 2 Numerical Problem.

script will the the addition of the two functions named `objfn` and `constfn1` that accept as input the current value of the solution vector and return the various functional results. The function names are arbitrary, but i selected them to be meaningful (to me at least) and not to interfere with names used by the package. Listing 22 shows the code for the two new functions.

Listing 22. R code for functions `objfn` and `constfn1`.

```
# Define the objective function
#
```

[8] The various components are analytical because we used calculus to find the gradient of the objective and the directional derivatives of the constraints.

```
# f(x) = x1*(x2^2)
#
objfn <- function(x) {
  objfn <- x[1]*x[2]^2;
  return(objfn)
}
# Define the constraint function
#   x1^2 + x2^2 - 2 <= 0
#
constfn1 <- function(x){
  constfn1 <- x[1]^2 + x[2]^2 -2 ;
  return(constfn1)
}
```

These two functions are then substituted into the optimization interface functions and their respective derivatives[9] as shown in Listing 23

Listing 23. R code for functions `objfn` and `constfn1`.

```
# Construct the evaluation structure for nloptr
#
eval_f <- function( x ) {
  return( list( "objective" = objfn(x),
                "gradient" = nl.grad(x,objfn). # << Here we use the numerical method for
                    gradient
              )
          )
}
# Construct the constraint functions
# Inequalities
eval_g_ineq <- function( x ) {
  constr <- constfn1(x);
  grad <- nl.jacobian(x,constfn1) # << Here we use the numerical method for Jacobian
  return( list( "constraints"=constr, "jacobian"=grad ) )
}
```

The entire problem structure for the **R** script is shown in Listing 24 and a complete solution is displayed in Figure 39.

Listing 24. R code for Example 2A – Numerical Derivatives.

```
# Example problem, pg 209 Gill, Murray, and Wright
#
# min_{x} x1*(x2^2)
# s.t.
#   x1^2 + x2^2 - 2 <= 0
#
# x0 = (-1,-1)
#
# optimal solution = (-0.81650,-1.1547)
#
# Numerical approximations to compute the gradient of the objective function
# and the jacobian of the constraint set via the nloptr methods: nl.grad and nl.jacobian

library('nloptr')
# Define the objective function
# f(x) = x1*(x2^2)
#
objfn <- function(x) {
  objfn <- x[1]*x[2]^2;
  return(objfn)
}
# Define the constraint function
#   x1^2 + x2^2 - 2 <= 0
#
constfn1 <- function(x){
  constfn1 <- x[1]^2 + x[2]^2 -2 ;
  return(constfn1)
}
# Construct the evaluation structure for nloptr
#
eval_f <- function( x ) {
  return( list( "objective" = objfn(x),
                "gradient" = nl.grad(x,objfn)
              )
          )
}
# Construct the constraint functions
```

[9]The gradient in the case of the objective function using `nl.grad`, and the Jacobian in the case of the constraint function using `nl.jacobian`

```
# Inequalities
eval_g_ineq <- function( x ) {
  constr <- constfn1(x);
  grad <- nl.jacobian(x,constfn1)
  return( list( "constraints"=constr, "jacobian"=grad ) )
}
# Set initial values
x0 <- c( -1,-1 )
# Set lower and upper bounds of control
lb <- c( -5, -5 )
ub <- c( 5, 5 )
# Begin the optimization call(s)
local_opts <- list( "algorithm" = "NLOPT_LD_MMA",
                    "xtol_rel"  = 1.0e-7 )
opts <- list( "algorithm" = "NLOPT_LD_AUGLAG",
              "xtol_rel"  = 1.0e-7,
              "maxeval"   = 1000,
              "local_opts" = local_opts )
res <- nloptr( x0=x0,
               eval_f=eval_f,
               lb=lb,
               ub=ub,
               eval_g_ineq=eval_g_ineq,
# Activate next line if there are equality constraints
#               eval_g_eq=eval_g_eq,
              opts=opts)
print( res )
```

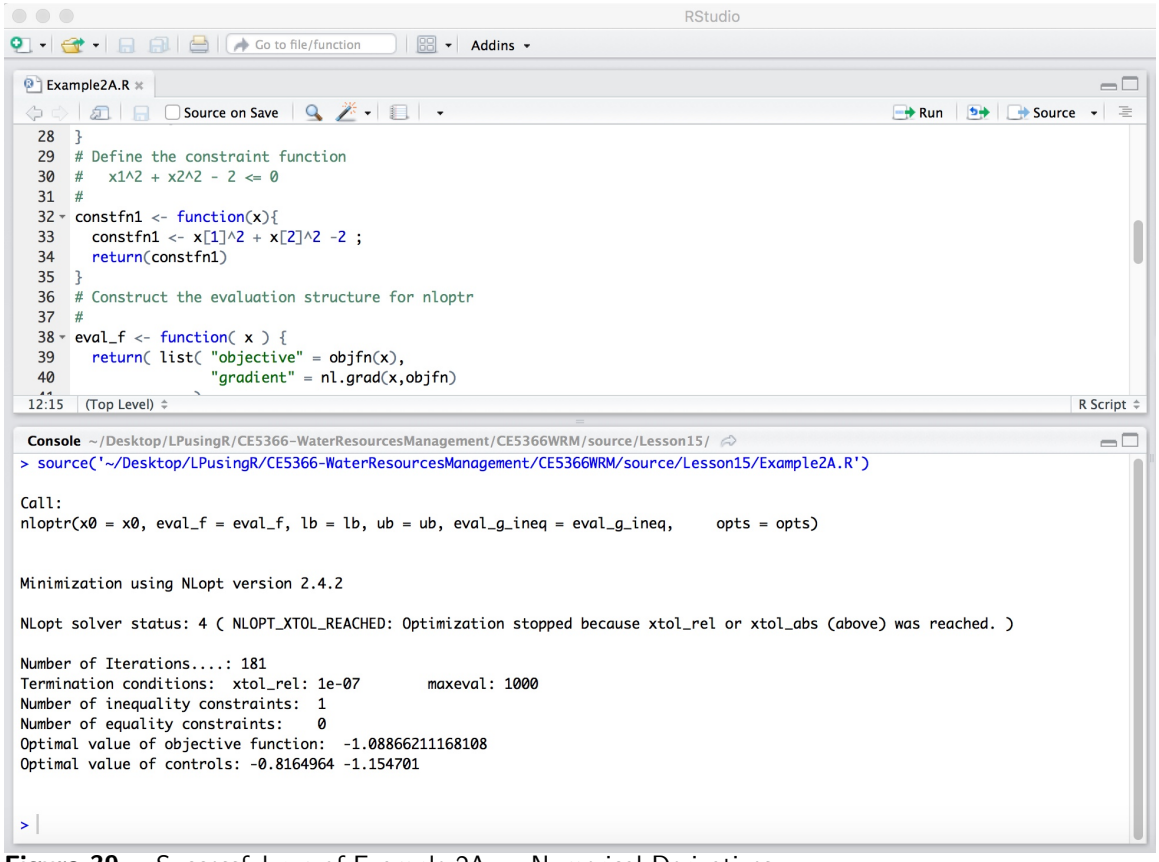


Figure 39. Successful run of Example 2A — Numerical Derivatives.

15.6. Topic 5: Water Resources Allocation/Operation Problem

15.6.1. Example

15.7. Topic 6: Water Quality Allocation Problem

15.7.1. Example

15.8. Readings

15.9. Exercises

1. Solve the following constrained non-linear optimization problem.

   $$\begin{array}{l} \textit{minimize} \quad (x_1 - x_2)^2 + (x_2 - x_3)^4 \\ \textit{subject to} \end{array}$$

$x_1 + x_1(x_2)^2 + (x_3)^4$			$=$	3
x_1			$\geq$	-5
	x_2		$\geq$	-5
		x_3	$\geq$	-5
x_1			$\leq$	5
	x_2		$\leq$	5
		x_3	$\leq$	5

   A solution should exist at $x^* = (1, 1, 1)$.

2. Solve the following constrained non-linear optimization problem.

   $$\begin{array}{l} \textit{minimize} \quad (x_1)^3 - 6(x_1)^2 + 11x_1 + x_3 \\ \textit{subject to} \end{array}$$

$(x_1)^2 + (x_2)^2 - (x_3)^2$			$\leq$	0
$(x_1)^2 + (x_2)^2 + (x_3)^2$			$\geq$	4
		$(x_3)^2$	$\leq$	5
x_1			$\geq$	0
	x_2		$\geq$	0
		x_3	$\geq$	0
x_1			$\leq$	$+Inf$
	x_2		$\leq$	$+Inf$
		x_3	$\leq$	$+Inf$

   A solution should exist at $x^* = (0, \sqrt{(}2), \sqrt{(}2))$. Of special note, the objective function does not contain x_2; its value is strictly established by the constraint requirements.

3. Water Resources Allocation Problem 1 (Next time)

4. Water Quality Allocation Problem 2 (Next time)

16. Title

words

16.1. Topic 1

16.1.1. Example

16.2. Topic 2

16.2.1. Example

16.3. Topic 3

16.3.1. Example

16.4. Readings

16.5. Exercises

17. Title

words

17.1. Topic 1

17.1.1. Example

17.2. Topic 2

17.2.1. Example

17.3. Topic 3

17.3.1. Example

17.4. Readings

17.5. Exercises

18. Title

words

18.1. Topic 1

18.1.1. Example

18.2. Topic 2

18.2.1. Example

18.3. Topic 3

18.3.1. Example

18.4. Readings

18.5. Exercises

19. Title

words

19.1. Topic 1

19.1.1. Example

19.2. Topic 2

19.2.1. Example

19.3. Topic 3

19.3.1. Example

19.4. Readings

19.5. Exercises

20. Title

words

20.1. Topic 1

20.1.1. Example

20.2. Topic 2

20.2.1. Example

20.3. Topic 3

20.3.1. Example

20.4. Readings

20.5. Exercises

21. Title

words

21.1. Topic 1

21.1.1. Example

21.2. Topic 2

21.2.1. Example

21.3. Topic 3

21.3.1. Example

21.4. Readings

21.5. Exercises

22. Title

words

22.1. Topic 1

22.1.1. Example

22.2. Topic 2

22.2.1. Example

22.3. Topic 3

22.3.1. Example

22.4. Readings

22.5. Exercises

23. Title

words

23.1. Topic 1

23.1.1. Example

23.2. Topic 2

23.2.1. Example

23.3. Topic 3

23.3.1. Example

23.4. Readings

23.5. Exercises

24. Title

words

24.1. Topic 1

24.1.1. Example

24.2. Topic 2

24.2.1. Example

24.3. Topic 3

24.3.1. Example

24.4. Readings

24.5. Exercises

References

Asquith, W.H., and Slade, R.M., 1997, Regional equations for estimation of peak-streamflow frequency for natural basins in Texas: U.S. Geological Survey Water Resources Investigations Report 96–4307, `http://pubs.usgs.gov/wri/wri964307/`.

Asquith, W.H., and Roussel, M. S., 2009, Regression equations for estimation of annual peak-streamflow frequency for undeveloped watershed in Texas using an L-moment-based, PRESS-minimized, residual adjusted approach. U.S. Geological Survey Scientific Investigations Report 2009–5087.

Asquith, W.H., Herrmann, G.R., and Cleveland, T.G., 2013, Use of generalized additive modeling for regionalization of a discharge measurement database in Texas. Journal of Hydrologic Engineering, American Society of Civil Engineers, *in press*

Gordon, N.D., T.A. McMahon, B.L. Finlayson, C.J. Gippel, R.J. Nathan, 2004, Stream Hydrology: An Introduction for Ecologists (second edition). John Wiley, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, 429 p.

U.S. Geological Survey, 2009, Streamflow measurements for Texas: USGS National Water Information System, accessed March 1, 2009, `http://waterdata.usgs.gov/tx/nwis/measurements/?site_no=STATIONID&agency_cd=USGS`.