Optimal Flow Capacity of a Chlorine Disinfection System

Jacob Turner

Computational Methods III

9/13/2018

Introduction

Problems within environmental engineering often require the use of a numerical method to solve for a specific parameter under measurable conditions. This parameter would typically be far too complex to solve manually, and typically dictates an important aspect of the design of a system. Therefore, using a language such as FORTRAN to program an algorithm that can solve for the desired parameter is an imperative skill that will prove invaluable to future engineers. The exercise at hand consists of solving for an optimal flow rate to properly size a design of a chlorination disinfection system. The design will factor in the flow rate able to be treated 95% of the time. The other 5 % of the time, the flow will be over capacity. Though this may be a necessary trade off because it will likely require substantially fewer resources to maintain. The probability density function (PDF) describing the variation in flow rate that was used in the program will be discussed in further detail in the methodology section. The primary objective is to successfully solve for the design flow of the system by use of the Romberg integration algorithm. In addition, students will also be able to practice writing a program that utilizes a numerical method to solve a real-world problem. Finally, a sensitivity analysis will be performed to understand how the optimal flow is affected by changing other parameters.

Methodology

The overall approach, governing equations for the probability density function, and a description of the algorithms used will be discussed in this section. The goal of the exercise was to determine the

maximum flow at which 95% of the daily water demand will be less than the capacity. To find this flow capacity one must evaluate the probability density function of the flow. A probability density function is a model described by a function of a random variable where the area under the curve of the equation is the probability that the value of the variable lies within the interval of the PDF (Nykamp). The PDF (Eqn.1) was determined from direct measurements of the water demand at the plant. This means that the maximum flow that is less than or equal to 95% of the daily water demand can be written as the integral of the PDF, shown in Eqn. 1 (Lang, Lab Handout, 2018).

$$P(X \leq Q_{0.95}) = \int_{e^{\gamma}}^{Q_{0.95}} f_x(x)dx = 0.95 \qquad \text{(Eqn. 1)}$$

The lower limit of the integral shown is the minimum flow or daily water demand on the system, this is a known value. The upper limit is the maximum daily water demand so that 95% of the time the flow is under the design capacity, this is the value being solved for within the program. For example, integrating from the lower limit to the upper limit of $Q_{0.98}$ would return a maximum flow so that the integral would equal 0.98. In other words, maximum flow capacity would be sufficient at handling the daily demand 98% of the time. This concept is important because it will be discussed in the sensitivity analysis. The PDF evaluated by the integral is a log Pearson type III probability distribution (Lang, Handout, 2018). This PDF is shown on the following page in Eqn. 2 (Lang, Handout, 2018).

$$f_x(x) = \frac{1}{\alpha x \Gamma(\beta)} \frac{(\ln(x)-\gamma)^{(\beta-1)}}{\alpha^{(\beta-1)}} e^{-[\frac{\ln(x)-\gamma}{\alpha}]} \; ; \; x > e^{\gamma} \qquad \text{(Eqn. 2)}$$

Where:

α is a measured scale parameter

β is a measured shape parameter

γ is a measured location parameter [$\ln(\frac{m^3}{day})$]

$\Gamma(\beta)$ is the Gamma function

x is the average daily water demand ($\frac{m^3}{day}$)

To solve this problem for the maximum daily flow a program was written in FORTRAN that integrates the PDF by utilizing Romberg integration with the trapezoidal rule and Richardson's Extrapolation. The trapezoidal rule is similar to Riemann summation in that it approximates the area under a curve, instead of using trapezoids, this results in a more accurate estimate (*Khan Academy*). The number of trapezoids under the curve of the function are then increased. As the number of trapezoids approaches infinity the area estimate of the interval increases in accuracy. The Trapezoidal Rule is shown below in Eqn. 3 (Chapra & Canale, 2015, chapter 21 pg. 605).

$$I = \frac{(b-a)[f(a)+f(b)]}{2} \qquad \text{(Eqn. 3)}$$

A Romberg Integration Algorithm was used to solve for the integral of the PDF over the upper and lower limit of flow (Lang, Lecture, 2018). Romberg integration combines the Trapezoidal rule and Richardson's Extrapolation to estimate the integral of a function. Richardson's Extrapolation is used to improve the rate of convergence and the accuracy of a numerical method (UBC Mathematics Dept). Richardson's extrapolation uses two previous estimates from the numerical method to create a new estimate with a higher degree of accuracy (Chapra & Canale, 2015, chapter 22, pg. 638).

$$I_{j,k} = \frac{4^{(k-1)} I_{j+1,k-1} - I_{j,k-1}}{4^{(k-1)} - 1} \qquad \text{(Eqn. 4)}$$

Application

Romberg integration can be used to solve a wide range of engineering problems. The approach to solving this problem involved writing a general module subroutine to integrate over any function. This subroutine uses a Romberg Integration Algorithm with the trapezoidal rule (shown above in Eqn. 3) and Richardson's Extrapolation (shown above in Eqn.4). The PDF in Eqn. 2 was written into an external function that was interfaced into the subroutine so that it could integrate over the desired region. The parameter values used in this external function are shown below in Table1. In order to solve for the upper limit of the integral, the user is asked to input an initial value for the flow capacity and the percentile desired. A search algorithm was the created around the subroutine call statement to find the correct upper limit that equaled the desired percentile. This was done by placing the Romberg subroutine call statement in a Do loop. Search algorithms are often employed to work problem backwards or to use a known value to assess how close the theoretical value is to the actual and what direction to adjust the value to converge to the actual answer. The result of the integral from the initial guess was then compared to the desired percentile. If the result was under or over the desired percent, the program added or subtracted an arbitrary amount to the upper limit. This new upper limit was then sent back into the integration subroutine. This process was repeated until a result within a certain margin of error relative to the desired percentile was found.

Table 1) Parameters Used to Solve the PDF

Parameter	Value	Units
α	0.3204	N/A
β	2.75	N/A
γ	2.96	ln($\frac{m^3}{day}$)]
$P(X \leq Q_{0.95})$	0.95	(%)

Results and Discussion

The goal of the exercise was to determine the maximum flow capacity so that there is only a 5% chance that the daily water demand will exceed the design flow. The program iterates the integration subroutine to solve for the upper limit that results in 0.95, or the desired 95% probability. At this probability the maximum design flow is calculated to be 128.60 (m^3/day). This result indicates that there is only a 5% chance that the daily water demand will exceed this value and the chlorine disinfection system will be over capacity. To explore the relationship between the parameters a sensitivity analysis was performed. This showed the relationship between the change in flow and the percentile probability is exponential. In other words, as the probability of exceeding daily water demand approaches 0% the necessary flow capacity for the system drastically increases. This relationship is shown below in Figure 1.

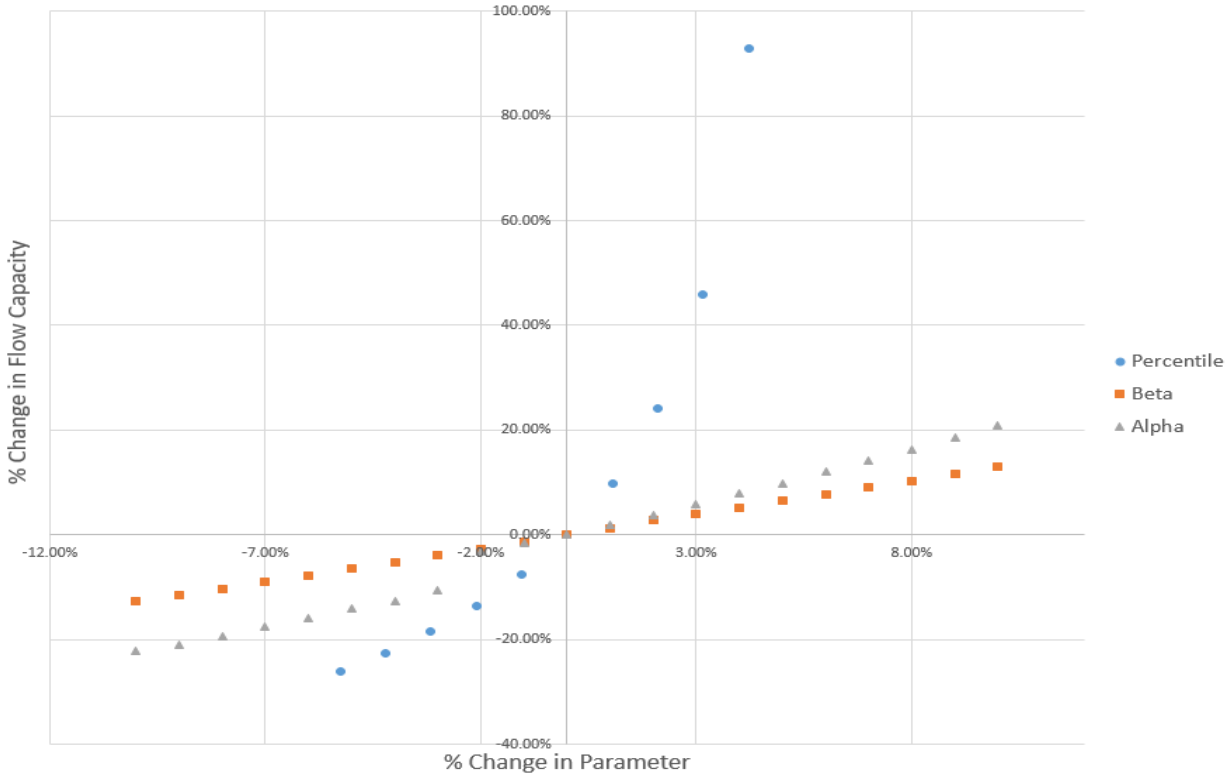

Figure 1) Percent Change in Flow Capacity vs. Percent Change in Parameter

This exponential relationship between the percentile and the flow capacity indicates that it is increasingly less efficient to design a maximum capacity with a higher probability. In order to combat this dilemma additional safety or control features may need to be implemented to handle the 5% chance of excess daily water demand on the system. The sensitivity analysis shows the relationship between α and the flow capacity to also have an exponential relationship, albeit much smaller than with the percentile. Finally, β and the flow capacity appear to have a linear relationship with one another.

Conclusions

Environmental engineering requires the use of complex methods and tools to determine solutions to intricate problems. Using a programming language such as FORTRAN to solve custom real-world problems is an essential skill for students to develop. Determining the maximum design flow of the chlorine disinfection system at a 95% probability was a primary goal in this lab exercise. In addition, exploring the relationships of the parameters in the PDF with the maximum flow were important goals. Exploring the relationship between α, β and how they affected the flow capacity was done through a sensitivity analysis. The program determined a maximum design flow of 128.60 (m^3/day) to allow for only a 5% chance of exceeding the capacity. Lastly, it was shown to be inefficient to design a Flow capacity with a probability significantly lower than 5% because the flow capacity necessary will increase exponentially and this would require significantly more resources.

References

1) Nykamp, Duane. Q. (n.d.). "The idea of a probability density function." *Math Insight*, <https://mathinsight.org/probability_density_function_idea> (Sep. 18, 2018).
2) *Khan Academy*. "Understanding the trapezoidal rule." (n.d.). Khan Academy, <https://www.khanacademy.org/math/ap-calculus-ab/ab-integration-new/ab-6-2/a/understanding-the-trapezoid-rule> (Sep. 18, 2018).
3) Israel, Robert. (n.d.). "Richardson Extrapolation." UBC Mathematics, <http://www.math.ubc.ca/~israel/m215/rich/rich.html> (Sep. 18, 2018).
4) Steven ,C Chapra and Raymond P. Canale (2015). *Numerical Methods for Engineers*, 7th Edition. (Chapter 21 pg 605 & Chapter 22 pg 638), McGraw-Hill Education, Singapore.

Appendix

Table 2) Raw Data Showing Relationship Between Flow and Percentile

% Changed in Percentile	*Percentile*	*% Chg. In Flow due to Percentile*	*Flow*
-5.26%	**90.00%**	-26.21%	94.9
-4.21%	**91.00%**	-22.63%	99.5
-3.16%	**92.00%**	-18.51%	104.8
-2.11%	**93.00%**	-13.61%	111.1
-1.05%	**94.00%**	-7.54%	118.9
0.00%	**95.00%**	0.00%	128.6
1.05%	**96.00%**	9.80%	141.2
2.11%	**97.00%**	24.03%	159.5
3.16%	**98.00%**	45.96%	187.7
4.21%	**99.00%**	92.76%	247.89

Table 3) Raw Data Showing Relationship Alpha, Beta, and Flow

Change in Parameter	*Flow due to Change in Alpha*	*% Chg. In Flow due to Alpha*	*Flow due to Change in Beta*	*% Chg. in Flow due to beta*
-10.00%	100.00	-22.18%	112.20	-12.68%
-9.00%	100.89	-21.09%	113.80	-11.44%
-8.00%	102.74	-19.31%	115.30	-10.27%
-7.00%	104.56	-17.61%	116.90	-9.03%
-6.00%	106.54	-15.85%	118.50	-7.78%
-5.00%	108.48	-14.18%	120.10	-6.54%
-4.00%	110.20	-12.72%	121.70	-5.29%
-3.00%	113.10	-10.50%	123.40	-3.97%
-2.00%	123.66	-3.24%	125.10	-2.65%
-1.00%	126.01	-1.63%	126.80	-1.32%
0.00%	128.50	0.00%	128.50	0.00%
1.00%	130.89	1.86%	130.20	1.32%
2.00%	133.40	3.81%	132.00	2.72%
3.00%	135.95	5.80%	133.70	4.05%
4.00%	138.55	7.82%	135.14	5.17%
5.00%	141.20	9.88%	136.80	6.46%
6.00%	143.90	11.98%	138.45	7.75%
7.00%	146.65	14.12%	140.11	9.04%
8.00%	149.46	16.31%	141.77	10.32%
9.00%	152.31	18.53%	143.42	11.61%
10.00%	155.23	20.80%	145.08	12.90%

Program Script

```
!This module establishes parameters, reals, and integers
module vars
   integer, parameter :: wp = selected_real_kind(15)
   real(wp), parameter :: alpha=0.3204, beta=2.75, lambda=2.96,ll=exp(lambda),epsi =  0.0001!0.000001
   real(wp) :: valu,ul,percent
   integer :: maxit, n
   !ll=lower limit
   !ul=upper limit, solving for this
   !epsi= exit criteria
   !n=iteration
end module vars
!_____________________________________________________________________________
module sub
use vars
implicit none
   contains
   subroutine romberg(ll,ul,f,valu,epsi,maxit,n)
      implicit none
      !Global variables
      !----------------------------
      real(wp), intent(in) :: ll, epsi
      real(wp), intent(out) :: valu
      real(wp), intent(inout) :: ul
      integer, intent(in) :: maxit
      integer, intent(out) :: n
```

```
!interface external function here
Interface
   function f(xa)
   use vars
   implicit none
   real(wp), intent(in) :: xa
   real(wp) :: f
   end function f
end interface
!local var
!-----------------------------
real(wp), dimension(maxit+1,maxit+1) :: t
real(wp) :: ratio, step, s1, s2, s2sum, s2i, sfinal
integer :: k,i, j, nt
!initialize local vars
nt = 2
ratio = 2 !long/short
t=0
step = (ul-ll)/nt
s1 = (f(ll) + f(ul))/2
s2 = 0 !intialize s2 before sum
do k = 1, (nt-1) !Sum s2
   s2sum = f(ll+ (k*step))
   s2 = s2 + s2sum
end do
t(1,1) = step*(s1+s2)
n=1 !iterations
do
```

```
        nt = 2*nt !double traps every time
        step = (ul-ll)/nt
        sfinal = 0 !initalize
        !summing s2 here before extrapolations
        do i = 1, (nt/2) !this sum is getting stuck
           s2i= f(ll + step*((2.0*i) - 1.0))
           sfinal = sfinal + s2i


        end do
        s2 = s2 + sfinal
        t(n+1,1) = step*(s1 + s2)
        do j = 2, (n+1) !extrapolations formula
           t(n+1,j) = (ratio**(2.0*(j-1))*t(n+1,j-1)-t(n,j-1))/(ratio**(2.0*(j-1))-1)
        end do
        !check each iteration against the stopping criteria
        if (abs(t(n+1,n) - t(n+1, n+1)) < 0.000001) then !Converged
           valu = t(n+1,n+1)
           exit
        else if (n > maxit) then ! too many iterations
           write(*,*) "The max iterations allowed were hit! n = ",n
           exit
        end if
        n=n+1
      end do
   end subroutine romberg
end module sub
!_____________________________________________________________________________
```

```
!This module contains all the input and output
!_____________________________________________________________________________
! Written by Jacob Turner
! This program will determine the design flow for a disinfection system
!_____________________________________________________________________________
program romberg_integration
use vars
use sub
implicit none
real(wp) :: answer
real(wp) :: max_flow, perc
!interface for function here
!------------------------------
Interface
   function f(xa)
      use vars
      implicit none
      real(wp), intent(in) :: xa
      real(wp) :: f
   end function f
end interface
!ask user for probability percent and an initial guess
write(*,*)
write(*,*) "Enter the desired probability percentile"
read(*,*) answer
percent = answer/100
write(*,*)
write(*,'(A,F4.2)') "Your desired percentile is ", percent
```

```
write(*,*)
maxit = 30
write(*,'(A,I2,A)') "Stopping after ", maxit," iterations."
write(*,*)
write(*,*) "what is your initial guess for the upper limit at this percentile?"
read(*,*) ul



do
   !Call subroutine with romberg
   call romberg(ll,ul,f,valu,epsi,maxit,n)
   if (abs(valu-percent) < epsi) then
      perc = valu*100
      write(*,'(A,f8.4,A)') "Upper limit reached at ",perc," %"
      max_flow = ul
      exit
   else if (valu>percent) then
      !write(*,'(A,f8.6,A)') "Integration result ",valu," is too high."
      if (abs(valu-percent)>0.05) then
      ul = ul - 4
      else if (abs(valu-percent)<0.05 .or. abs(valu-percent)>0.01) then
      ul = ul - .1
      else if (abs(valu-percent)<0.01) then
      ul = ul - .0001
      end if
   else if (valu<percent) then
      !write(*,'(A,f8.6,A)') "Integration result ",valu," is too low."
```

```
      if (abs(valu-percent)>0.05) then
      ul = ul + 3
      else if (abs(valu-percent)<0.05 .or. abs(valu-percent)>0.01) then
      ul = ul + .1
      else if (abs(valu-percent)<0.01) then
      ul = ul + .0001
      end if
   end if

end do
write(*,'(A,f10.6,A)') "Maximum design flow is ", max_flow, " (m^3/day)"
write(*,'(A,I3)') "Iterations necessary to converge", n
stop
end program romberg_integration
!_____
function f(x)
   use vars
   implicit none
   real(wp), intent(in) :: x
   real(wp) :: f, one, two, three
   one = 1/(alpha*x*Gamma(beta))
   two = ((log(x) - lambda)**(beta-1))/(alpha**(beta-1))
   three = exp(-1*((log(x)-lambda)/alpha))
   f = one*two*three
   return
end function f]0;jdt385@CNRS-LinuxMint: ~/Documents/program[01;32mjdt385@CNRS-
LinuxMint[00m:[01;34m~/Documents/program[00m$ gfortran ch.f90 -o hi.exe

]0;jdt385@CNRS-LinuxMint: ~/Documents/program[01;32mjdt385@CNRS-
LinuxMint[00m:[01;34m~/Documents/program[00m$ ./hi.exe
```

Enter the desired probability percentile

95

Your desired percentile is 0.95

Stopping after 30 iterations.

what is your initial guess for the upper limit at this percentile?

200

Upper limit reached at 95.0055 %

Maximum design flow is 128.599999 (m^3/day)

Iterations necessary to converge 6

]0;jdt385@CNRS-LinuxMint: ~/Documents/program[01;32mjdt385@CNRS-LinuxMint[00m:[01;34m~/Documents/program[00m$ ./hi.exe

Enter the desired probability percentile

90

Your desired percentile is 0.90

Stopping after 30 iterations.

what is your initial guess for the upper limit at this percentile?

120

Upper limit reached at 89.9941 %

Maximum design flow is 94.900000 (m^3/day)

Iterations necessary to converge 5

]0;jdt385@CNRS-LinuxMint: ~/Documents/program[01;32mjdt385@CNRS-LinuxMint[00m:[01;34m~/Documents/program[00m$ exit

exit