Diffusion Equation

Impact of Soil Phosphate Level on an Ecologically Sensitive Lake

Dikshita Kothari
500539170

Date: 8th April 2022

In this report, the impact of soil phosphate level in a section of the Murrumbidgee River is examined. The river empties into an ecologically sensitive lake. A dam separates the river and the lake to prevent the lake from becoming polluted.

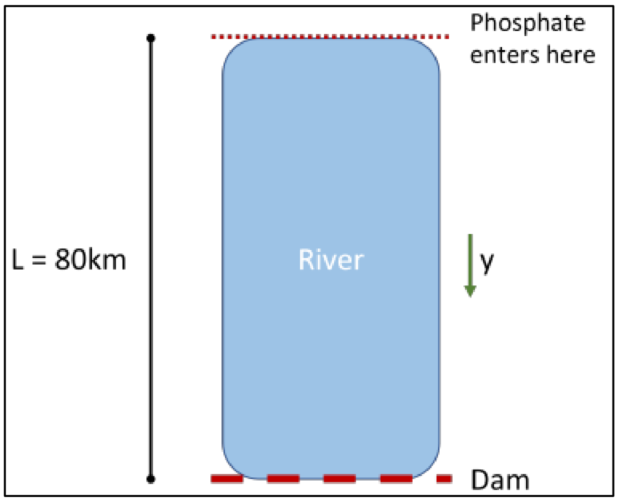

Figure 1. Simplified river schematic

Section 1: Analytical Solution to the Diffusion Equation

In this section, an analytical solution to the diffusion equation is developed. The concentration of phosphate is evaluated as a function of space and time. This concentration (C), can be modelled using the governing equation given by,

$$\frac{\partial}{\partial t}C(y,t) = D\frac{\partial^2}{\partial y^2}C(y,t) \tag{1}$$

This can be written as,

$$C_t = DC_{yy} \tag{2}$$

1. **Initial Conditions and Boundary Conditions**
   It is given that the water is initially unpolluted. Therefore, the initial condition is given by,
   $$C(y,0) = 0 \tag{3}$$
   At the start of the river section, the concentration of water is 50 mg/L. Therefore, the boundary condition at the start (i.e., at y = 0 m) is a Dirichlet boundary condition given by,
   $$C(0,t) = \varphi_0 = 50 \tag{4}$$
   The dam separating the river from the lake is insulated and stops outflow of polluted water. This means that the rate of change of concentration or gradient at the dam is a Neumann boundary condition given by,
   $$C_y(L,t) = \varphi_1 = 0 \tag{5}$$
   Here, L is the length of the river section given as 80 km.
2. **Steady State Solution**

There is a non-zero Dirichlet boundary condition where the phosphate enters the river and a Neumann boundary condition at the dam. Due to these mixed boundary conditions the concentration of phosphate in the river C(y,t) can be represented as a sum of the steady state solution and the homogenous solution given as,

$$C(y,t) = C^H(y,t) + C^{SS}(y) \tag{6}$$

The boundary conditions are as stated in equation (4) and (5). Therefore, the steady state solution can be represented as,

$$C^{SS}(y) = \varphi_0 + \varphi_1 y = 50 \tag{7}$$

The steady state solution is time independent.

The first derivative of the steady state solution with respect to time is given by,

$$C_t^{SS}(y) = 0 \tag{8}$$

The second derivative of the steady state solution with respect to space is given by,

$$C_{yy}^{SS}(y) = 0 \tag{9}$$

Plugging these values into the governing equation (1), we get both sides equal to zero. This shows that the steady state solution satisfies the governing equation.

3. **Separation of Variables**

The homogeneous solution $C^H(y,t)$ can be written as a product of two functions.

$$C^H(y,t) = F(y) \times G(t) \tag{10}$$

Where, $F(y)$ is function of space and $G(t)$ is a function of time.

Differentiating (10) with respect to time,

$$C_t^H(y,t) = F(y) \times G_t(t) \tag{11}$$

Differentiating (10) twice with respect to space,

$$C_{yy}^H(y,t) = F_{yy}(y) \times G(t) \tag{12}$$

Plugging (11) and (12) in the governing equation (1) we get,

$$\frac{G_t(t)}{DG(t)} = \frac{F_{yy}(y)}{F(y)} = -p^2 \tag{13}$$

This equation holds true for all y and t. The left-hand side does not depend on y and the right-hand side does not depend on t. Hence each side must be a constant. This is a non-trivial solution as $-p^2$ will always be a negative constant. Here, p^2 is the separation constant.

Rearranging we get the spatial ODE as,

$$F_{yy}(y) + p^2 F(y) = 0 \tag{14}$$

And the temporal ODE as,

$$G_t(t) + Dp^2 G(t) = 0 \tag{15}$$

Therefore, we have obtained two ODEs after separating $C^H(y,t)$ into two functions of y and t independently.

4. **General Solution to Homogeneous Part**

The boundary conditions of the steady state solution can be given by,

$$C^{SS}(0) = \varphi_0 = 50 \tag{16}$$

$$C_y^{SS}(L) = \varphi_1 = 0 \tag{17}$$

The boundary conditions for the homogenous solution can be derived from,

$$C^H(y,t) = C(y,t) - C^{SS}(y) \tag{18}$$

$$C_y^H(y,t) = C_y(y,t) - C_y^{SS}(y) \tag{19}$$

At the Dirichlet boundary (i.e., y = 0 m),

$$C^H(0,t) = C(0,t) - C^{SS}(0) = 50 - 50 = 0 \tag{20}$$

At the Neumann boundary (i.e., y = L = 80 km),

$$C_y^H(L,t) = C_y(L,t) - C_y^{SS}(L) = 0 - 0 = 0 \tag{21}$$

The homogenous boundary conditions can be used to derive the homogenous solution.

A possible solution to the spatial ODE (14) is,

$F(y) = A\cos py + B\sin py$ (22)

The first derivative of (22) with respect to space can be given as,

$F_y(y) = -Ap\sin py + Bp\cos py$ (23)

At y = 0, the homogeneous boundary condition requires that $F(y)$ is zero. Therefore, A = 0.

At y = L, the homogeneous boundary condition requires that $F_y(y)$ is zero. For this to be true,

$p = \frac{\left(n-\frac{1}{2}\right)\pi}{L}$ (24)

Where n is the mode.

Therefore,

$F(y) = B\sin py$ (25)

For each mode,

$F_n(y) = B_n \sin py$ (26)

A possible solution to the temporal ODE for each mode is,

$G_n(t) = b_n e^{-\lambda_n{}^2 t}$ (27)

Combining (26) and (27) we get the homogeneous solution for each mode as,

$C_n^H(y,t) = F_n(y) \times G_n(t) = B_n^* \sin\left(\frac{\left(n-\frac{1}{2}\right)\pi}{L}y\right) e^{-\lambda_n{}^2 t}$ (28)

The homogeneous solution can be given by,

$C^H(y,t) = \sum_{n=1}^{\infty} B_n^* \sin\left(\frac{\left(n-\frac{1}{2}\right)\pi}{L}y\right) e^{-\lambda_n{}^2 t}$

(29)

Where, B_n^* is the homogeneous initial condition.

5. Complete Solution

The initial condition to the homogeneous problem can be determined as,

$C^H(y,0) = C(y,0) - C^{SS}(y) = 0 - (\varphi_0 + \varphi_1 y) = 0 - 50 = -50$ (30)

The Fourier coefficient can be calculated as follows,

$B_n = \frac{2}{L}\int_0^L -50 \times \sin\left(\frac{\left(n-\frac{1}{2}\right)\pi}{L}y\right) dy$ (31)

$B_n = -\frac{100}{L}\left[-\frac{\cos(py)}{p}\right]_0^L = -\frac{100}{L}\left[-\frac{0-1}{p}\right] = -\frac{100}{Lp} = -\frac{100}{\left(n-\frac{1}{2}\right)\pi}$ (32)

Therefore, we can write the homogeneous solution as,

$C^H(y,t) = \sum_{n=1}^{\infty} -\frac{100}{\left(n-\frac{1}{2}\right)\pi} \sin\left(\frac{\left(n-\frac{1}{2}\right)\pi}{L}y\right) e^{-\lambda_n{}^2 t}$ (31)

The complete solution can be written as,

$C(y,t) = 50 + \sum_{n=1}^{\infty} -\frac{100}{\left(n-\frac{1}{2}\right)\pi} \sin\left(\frac{\left(n-\frac{1}{2}\right)\pi}{L}y\right) e^{-\lambda_n{}^2 t}$ (32)

6. Interpretation of results

a) The homogeneous solution can be written as shown.

$C^H(y,t) = F(y) \times G(t)$ (33)

F(y) is a function which describes the concentration of phosphate over the length of the river or at different points in the river independent of time. Whereas G(t) describes the concentration of phosphate in the river at a particular time independent of the location in the river.

b) The complete solution is given by,

$$C(y,t) = 50 + \sum_{n=1}^{\infty} -\frac{100}{\left(n-\frac{1}{2}\right)\pi} \sin\left(\frac{\left(n-\frac{1}{2}\right)\pi}{L} y\right) e^{-\lambda_n{}^2 t} \tag{34}$$

However, it is not practical to sum infinite modes in order to find the homogeneous equation. Adding larger number of eigen functions will give a better approximation of the initial condition. This is illustrated by the graph below. The first 10 kilometres are illustrated for clarity.

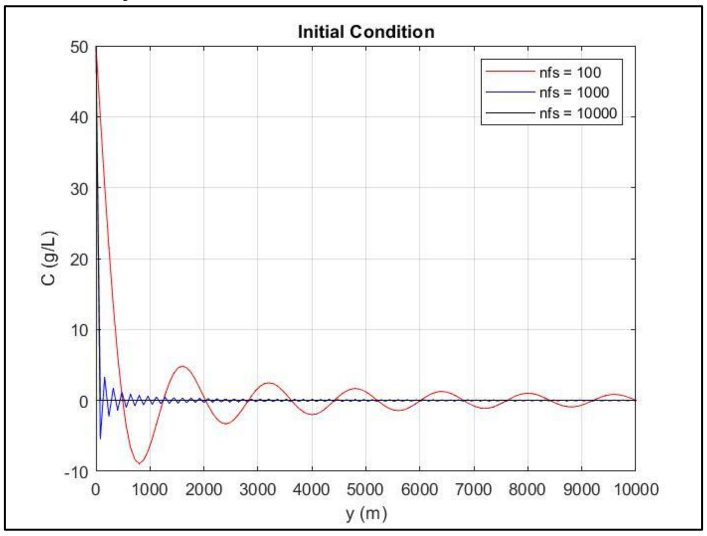

Figure 2. Effect of summing more eigenfunctions on the initial condition.

We know that the river is initially unpolluted implying that the concentration of phosphate in the river is zero at t = 0. From the graph it is evident that larger number of Fourier terms result in a better approximation to this initial condition.

7. **Time at which the concentration of phosphate exceeds permissible limit.**

The maximum permissible phosphate level in the river is 50 µg/L. The time taken for different parts of the river to exceed this is tabulated as below.

Length from start (kilometres)	Time Taken (hours)
20	22.22
40	88.89
60	197.22
80	311.11

Table 1. Time taken to exceed permissible limit.

Section 2: Numerical Solution to the Diffusion Equation

In this section, a numerical solution to the diffusion equation is discussed and explored. The concentration of phosphate is evaluated using the stencils given below.

$$C_i^{n+1} = \begin{cases} C_i^n + \frac{D\Delta t}{\Delta y^2}\left(C_{i+1}^n - 2C_i^n + C_{i-1}^n\right) & i = 2{:}ny-1 \\ C_i^n + \frac{D\Delta t}{\Delta y^2}\left(C_{i-1}^n - C_i^n\right) & i = ny \end{cases} \tag{35}$$

1. **Order of Accuracy**

The orders of accuracy for the two stencils are shown in the table below:

Stencil	Scheme	Order of Accuracy
$i = 2{:}ny - 1$	Forward in time	2nd
	Central in space	4th
$i = ny$	Forward in time	2nd
	Central in space	4th

Table 2. Order of accuracy of the given stencil.

The schemes and orders of accuracy are explained below.

a) Forward in Time

The Taylor series forward in time expansion can be written as,

$$C^{n+1} = C^n + \Delta t C_t^n + \frac{\Delta t^2}{2!} C_{tt}^n + \frac{\Delta t^3}{3!} C_{ttt}^n + \cdots \tag{36}$$

$$C_t^n = \frac{C^{n+1} - C^n}{\Delta t} - \frac{\Delta t^2}{2!} C_{tt}^n - \frac{\Delta t^3}{3!} C_{ttt}^n - \cdots \tag{37}$$

The higher orders of Δt are truncated in the forward in time approximation which is,

$$C_t^n = \frac{C^{n+1} - C^n}{\Delta t} - O(\Delta t^2) \tag{38}$$

This is known as truncation error. Therefore, the forward scheme is 2nd order accurate.

b) Central in Space

The Taylor series forward in space expansion can be written as,

$$C_{i+1} = C_i + \Delta y C_{i_y} + \frac{\Delta y^2}{2!} C_{i_{yy}} + \frac{\Delta y^3}{3!} C_{i_{yyy}} + \frac{\Delta y^4}{4!} C_{i_{yyyy}} \ldots \tag{39}$$

The Taylor series backward in space expansion can be written as,

$$C_{i-1} = C_i - \Delta y C_{i_y} + \frac{\Delta y^2}{2!} C_{i_{yy}} - \frac{\Delta y^3}{3!} C_{i_{yyy}} + \frac{\Delta y^4}{4!} C_{i_{yyyy}} \ldots \tag{40}$$

Adding equation (39) and (40),

$$C_{i+1} + C_{i-1} = 2C_i + 2\frac{\Delta y^2}{2!} C_{i_{yy}} + 2\frac{\Delta y^4}{4!} C_{i_{yyyy}} \ldots \tag{41}$$

$$C_{i_{yy}} = \frac{C_{i+1} + C_{i-1} - 2C_i}{\Delta y^2} - \frac{\Delta y^4}{2} C_{i_{yyyy}} - \cdots \tag{42}$$

The higher orders of Δy are truncated in the central approximation which is,

$$C_{i_{yy}} = \frac{C_{i+1} + C_{i-1} - 2C_i}{\Delta y^2} - O(\Delta y^4) \tag{43}$$

This is known as truncation error. Therefore, the central scheme is 4th order accurate.

2. MATLAB implementation

The given scheme was implemented in MATLAB. The maximum number of grid points is 257 and its corresponding spatial step is 312.5. The maximum stable time step can be calculated as,

$$\Delta t \leq \frac{\Delta y^2}{2D} \tag{44}$$

Therefore, Δt is chosen as 206.

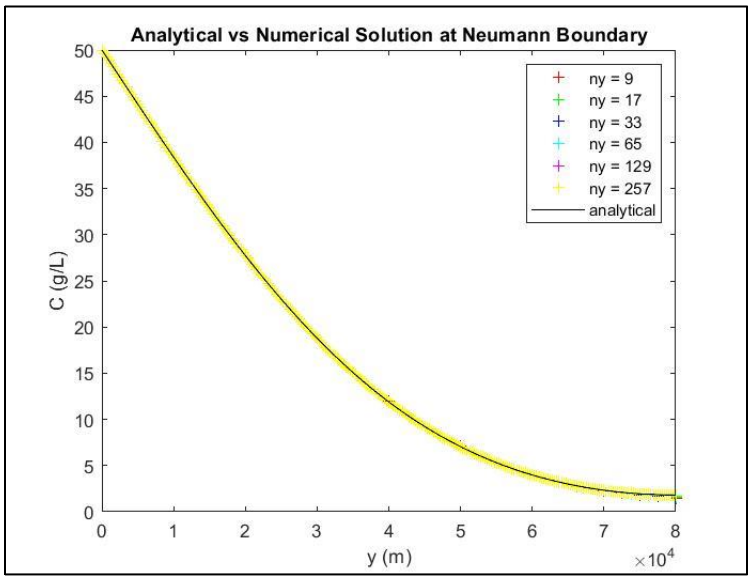

Figure 3. Numerical solution for different step sizes.

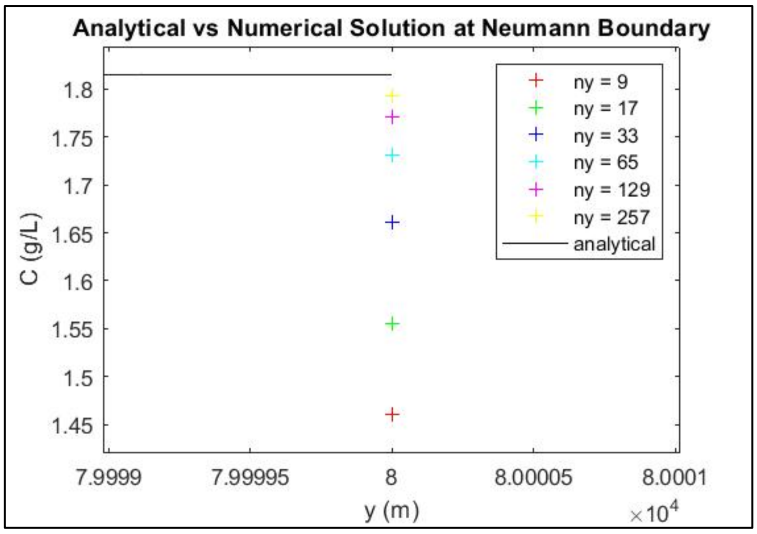

Figure 4. Zoomed plot near the Neumann boundary.

The observation is that the numerical solution is closer to the exact solution when the number of grid points are increased. The grid size is proportional to the accuracy.

3. Verification of Order of Accuracy

L^1 error is the mean absolute error, L^2 the rms, and L^∞ the maximum error. We can calculate this for the solution on multiple grid sizes. The error norms are tabulated as,

Grid Size (ny)	L^1	L^2	L^∞	$O(L^1)$	$O(L^2)$	$O(L^\infty)$
9	6.6714e+3	37.6765	0.3541	-	-	-
17	3.3736e+3	23.3922	0.2600	1.0721	0.7494	0.4857
33	1.9471e+3	13.1726	0.1544	0.8287	0.8658	0.7857
65	1.0431e+3	6.9951	0.0839	0.9207	0.9337	0.8997
129	543.2242	3.6154	0.0438	0.9519	0.9629	0.9483
257	285.4614	1.8511	0.0225	0.9335	0.9712	0.9664

Table 3. Error norms for different grid sizes.

It is observed that with increasing grid size the error values decreases. This is as expected. Therefore, the maximum grid size 257 is used for further analysis.

However, the order of error is obtained is different from what was expected.

4. Numerical Modifications

a) Addition of Source Term

The governing equation is modified by including a source term which accounts for the phosphate that the riverbed absorbs. The value of α is 1.17×10^{-6} and $C_{eq} = 1$ µg/L. The new governing equation is given by,

$$\frac{\partial}{\partial t}C(y,t) - D\frac{\partial^2}{\partial y^2}C(y,t) = \alpha\left(C_{eq} - C(y,t)\right) \tag{45}$$

The Central in Space, Forward in Time (FTCS) scheme is used to formulate the new equation for $i = 2{:}ny - 1$.

The central in space scheme is given by,

$$C_{yy}{}_i^n = \frac{C_{i+1}^n - 2C_i^n + C_{i-1}^n}{\Delta y^2} \tag{46}$$

The forward in time scheme is given by,

$$C_t{}_i^n = \frac{C_i^{n+1} - C_i^n}{\Delta t} \tag{50}$$

The governing equation can therefore be adjusted as,

$$C_i^{n+1} = \begin{cases} \alpha\Delta t(C_{eq} - C_i^n) + \frac{D\Delta t}{\Delta y^2}(C_{i+1}^n - 2C_i^n + C_{i-1}^n) + C_i^n & i = 2{:}ny - 1 \\ \alpha\Delta t(C_{eq} - C_i^n) + \frac{D\Delta t}{\Delta y^2}(C_{i-1}^n - C_i^n) + C_i^n & i = ny \end{cases} \tag{51}$$

The maximum stable time step is chosen according to the equation,

$$\Delta t \leq \frac{\Delta y^2}{4D} \tag{52}$$

It is found to be 103. This value is chosen for the evaluation. The best grid size is 257. The concentration of phosphate is checked after 14 days.

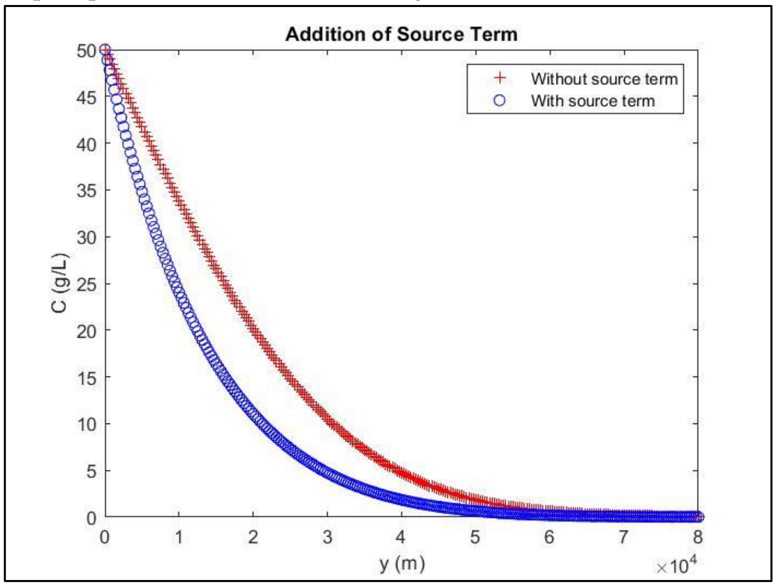

Figure 5. Comparison of pollutant concentration with and without the source term.

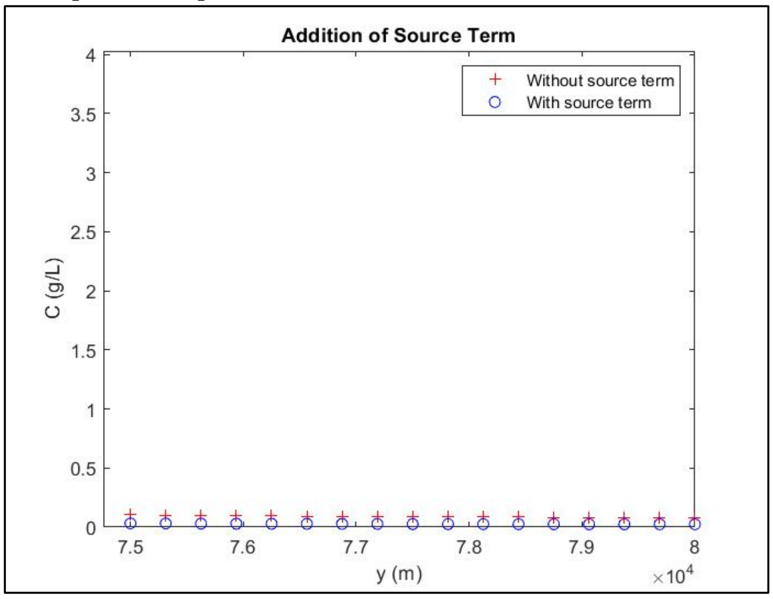

Figure 6. Comparison of pollutant concentration at the Neumann Boundary.

It is observed that the addition of the source term has a large effect on the phosphate levels in the river. Without the source term the concentration of phosphate at the Neumann boundary is 0.0815 g/L whereas with the source term it is four times less equal to 0.0240 g/L. It is confirmed that the riverbed is indeed absorbing phosphate from the river.

b) Increasing the α parameter

A particular chemical can be added to the river to artificially increase the value of α such that $\alpha = 2.0 \times 10^5$.

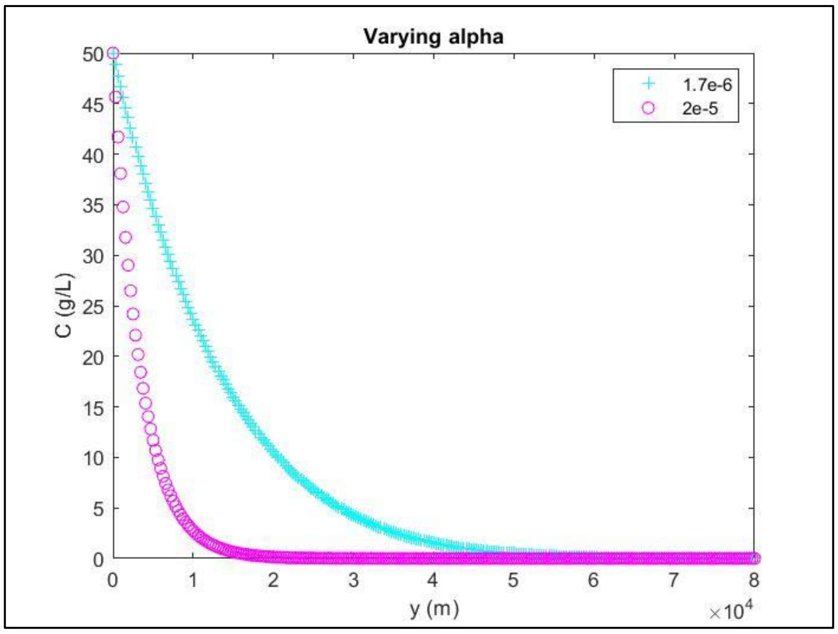

Figure 7. Effect of variation in alpha on concentration.

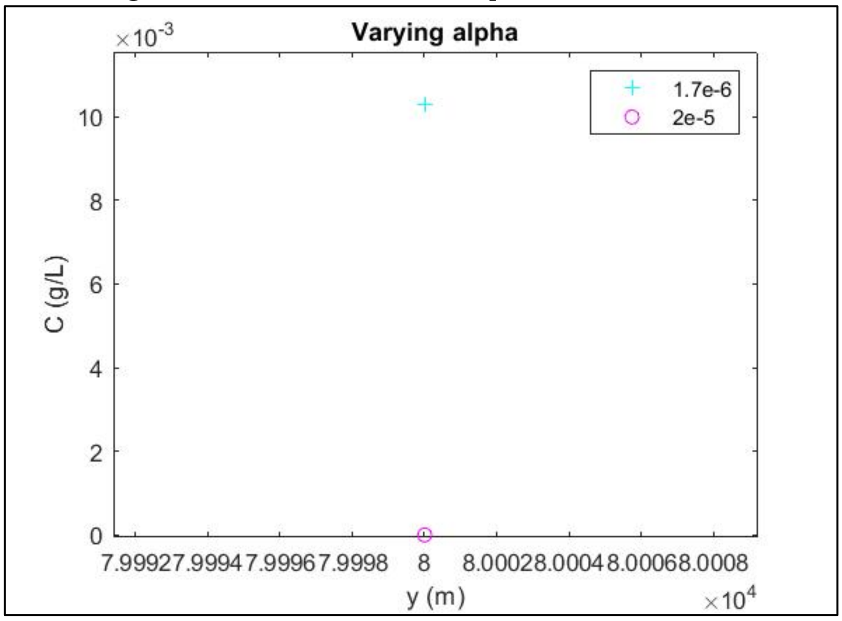

Figure 8. Effect of variation in alpha on concentration at the Neumann boundary.

The change in the value of alpha had a large effect on the concentration level. For $\alpha = 1.17 \times 10^{-6}$, the concentration at the Neumann boundary or at the dam is 0.0103 g/L and with $\alpha = 2.0 \times 10^{5}$, the concentration is 1.0078 µg/L which is much less than the limiting concentration which is 50 µg/L. This value remains the same even after 100 days. The lake has a limiting concentration of 100 µg/L. Therefore, the dam can be removed without causing harm to the ecologically sensitive lake.

Conclusion

The analytical and numerical solution was solved and compared. The increase in the number of eigen functions added results in a better approximation of the initial condition in the analytical solution. The river becomes polluted after a while therefore a chemical is added into it to stop the spread of the pollutant and can result in removal of the dam. The order of accuracy for the numerical scheme was discussed. Large number of spatial grid points results in a better numerical approximation.

APPENDIX

1. ANALYTICAL SOLUTION

```
clc
clear all
close all

timeend = 1000000;    % Ending time
nt = 101;             % Number of temoral points
plotFlag = 1;         % If want to plot

% Spatial (y) parameters
L      = 80000;                  % Size of the domain
ystart = 0;                      % Start of computational domain (m)
yend   = L;                      % End of computational domain (m)
ny     = 101;                    % Number of grid points along the  vector
dy     = (yend-ystart)/(ny-1);   % Spatial step size, delta y (m)
y      = ystart:dy:yend;         % Vector of grid points in y

% Temporal (t) parameters
timestart = 0;                           % Starting time (0 seconds)
dt        = (timeend-timestart)/(nt-1);     % Temporal step size, delta t (s)
time      = timestart:dt:timeend;        % Vector of times

% Phyiscal parameters
phi0     = 50;                       % Temperature at y=0
phi_1    = 0;                        % Temperature at y=L
D        = 2.37*10^2                 % Diffusion equation constant
d        = sqrt(D)                   % Diffusion equation constant
toxicity = 0;                        % Toxic flag

% Calculate B coefficients using a for loop
nfs = 1000;               % Number of fourier terms
B = zeros(1,nfs);         % Initialise B vector
lambda=zeros(1,nfs);      % Initialise lambda vector
p = zeros(1,nfs);         % Initialise p vector
for n = 1:nfs;            % Compute values
    p(n) = (n-0.5)*pi/L;
    B(n)= -(2*phi0/(L*p(n)));
    lambda(n)=d*p(n);
end


%% Solve for C using three for loops in time, space, and Fourier series

% Loop through time
for i = 1:nt               % For each time

    t = time(i);           % time in seconds

    % Vector of zeros to initialise the Fourier series solution.
    % This should be re-initialised at each new time step.
    C = zeros(1,ny);

    % Loop through space
    for j = 1:ny         % For each grid point
        C(j) = phi0;         % Add the steady state solution
        % Loop through the Fourier series
        for n = 1:nfs
            C(j)  = C(j) + B(n)*sin(p(n)*y(j))*exp(-lambda(n)^2*t);  % Calculate
series sum for T at y(i) and t
        end
    end

    % Plot the solution if requested
    if plotFlag==1
        plot(y,C);
        ylim([0 51])
        grid on;
        xlabel('y (m)');             % Do not forget the unit from the plot
        ylabel('C (g/L)');
        title('Diffusion in the river')
        pause(0.001);                % Animate the plot (use instead of pause)'
    end

    if toxicity == 0             % Check toxicity flag
        if C(ny) <= 50*10^-3     % Check against limit
            toxicity = 0
        else
            toxicity = 1         % River is toxic
            T = t;               % Save time
        end
    end
end
```

2. NUMERICAL SOLUTION

```
clc
close all

plotFlag = 1;

% Spatial (x) parameters analytical
L      = 80000;                     % Size of the domain
ystart = 0;                      % Start of computational domain (m)
yend   = ystart + L;             % End of computational domain (m)
nya    = 1000;                     % Spatial grid points
dya     = (yend - ystart)/(nya - 1);  % Spatial step size, delta r (m)
ya      = ystart: dya : yend;         % Vector of grid points

% Spatial y parameters numerical
ny =  [9, 17, 33, 65, 129, 257];
for i = 1:length(ny)
    dy(i) = (yend - ystart)/(ny(i) - 1);  % Spatial step size, delta r (m)
end
y1 = ystart: dy(1) : yend;
y2 = ystart: dy(2) : yend;
y3 = ystart: dy(3) : yend;
y4 = ystart: dy(4) : yend;
y5 = ystart: dy(5) : yend;
y6 = ystart: dy(6) : yend;

% Phyiscal parameters
phi0     = 50;                        % Temperature at Y=0
phi_1    = 0;                         % Temperature at Y=L
```

```matlab
D        = 2.37*10^2;                   % Diffusion equation constant
d        = sqrt(D);                     % Square root of Diffusion constant

% Temporal (t) parameters
timestart = 0;                              % Starting time (0 seconds)
timeend   = 86400*28;                       % Ending Time 28 days
dt        = 206;                            % Temporal step size, delta t (s)
nt        = (timeend - timestart)/dt + 1;   % Time grid points
time      = timestart:dt:timeend;           % Vector of times

% Initialise the solution arrays - Tn (T^n) and Tnp1 (T^(n+1)) and exact
% solution TExact
Cn1 = zeros(1,ny(1));
Cn2 = zeros(1,ny(2));
Cn3 = zeros(1,ny(3));
Cn4 = zeros(1,ny(4));
Cn5 = zeros(1,ny(5));
Cn6 = zeros(1,ny(6));

Cnp11 = zeros(1,ny(1));
Cnp12 = zeros(1,ny(2));
Cnp13 = zeros(1,ny(3));
Cnp14 = zeros(1,ny(4));
Cnp15 = zeros(1,ny(5));
Cnp16 = zeros(1,ny(6));

CExact = zeros(1,nya);    % Exact fourier series solution for temperature
%%
% Specify the initial conditions (already zeros)

% Enforce the boundary conditions - These will never change in the simulation
Cn1(1)       = 50;
Cn2(1)       = 50;
Cn3(1)       = 50;
Cn4(1)       = 50;
Cn5(1)       = 50;
Cn6(1)       = 50;

Cnp11(1)     = 50;
Cnp12(1)     = 50;
Cnp13(1)     = 50;
Cnp14(1)     = 50;
Cnp15(1)     = 50;
Cnp16(1)     = 50;


%Fourier parameters - pre-compute B and lambda vectors
nfs     = 100;                  % Number of fourier terms
B       = zeros(1,nfs);         % Initialise B vector
lambda  = zeros(1,nfs);         % Initialise lambda vector
p       = zeros(1,nfs);         % Initialise p vector
for n = 1:nfs;
    p(n)        = (n-0.5)*pi/L;
    B(n)        = -(2*phi0/(L*p(n)));
    lambda(n)   = d*p(n);
end


%% Main solution loop
```

```
% Loop through time
for i=2:nt                   % loop starts from 2 (not 1) to ensure that the
analytical and numerical solutions are compared at the same time point in the mean
error calc below

    t=time(i);               % current time for outputted solution

    CExact = zeros(1,nya);

    %Analytical solution (same as Week 3)
    for j = 1:nya                  % For each grid point
        CExact(j) = phi0;         % Add the steady state solution

        % Loop through the Fourier series
        for n = 1:nfs
            CExact(j)  = CExact(j) + B(n)*sin(p(n)*ya(j))*exp(-lambda(n)^2*t);  %
Calculate series sum for T at r(i) and t
        end
    end

    % 1 % Numerical solution - note that we only need to solve for j=2 to nr-1
    % since the first and last points are fixed by the boundary condition
    sigma   =  D*dt/(dy(1)^2);
    for j   =   2:ny(1)-1
        Cnp11(j) =   sigma*Cn1(j+1)+(1-2*sigma)*Cn1(j)+sigma*Cn1(j-1);
    end
        Cnp11(ny(1)) =   Cn1(ny(1))+sigma*(Cn1(ny(1)-1)-Cn1(ny(1)));

        % 2 % Numerical solution - note that we only need to solve for j=2 to nr-1
    % since the first and last points are fixed by the boundary condition
    sigma   =  D*dt/(dy(2)^2);
    for j   =   2:ny(2)-1
        Cnp12(j) =   sigma*Cn2(j+1)+(1-2*sigma)*Cn2(j)+sigma*Cn2(j-1);
    end
        Cnp12(ny(2)) =   Cn2(ny(2))+sigma*(Cn2(ny(2)-1)-Cn2(ny(2)));

        % 3 % Numerical solution - note that we only need to solve for j=2 to nr-1
    % since the first and last points are fixed by the boundary condition
    sigma   =  D*dt/(dy(3)^2);
    for j   =   2:ny(3)-1
        Cnp13(j) =   sigma*Cn3(j+1)+(1-2*sigma)*Cn3(j)+sigma*Cn3(j-1);
    end
        Cnp13(ny(3)) =   Cn3(ny(3))+sigma*(Cn3(ny(3)-1)-Cn3(ny(3)));

        % 1 % Numerical solution - note that we only need to solve for j=2 to nr-1
    % since the first and last points are fixed by the boundary condition
    sigma   =  D*dt/(dy(4)^2);
    for j   =   2:ny(4)-1
        Cnp14(j) =   sigma*Cn4(j+1)+(1-2*sigma)*Cn4(j)+sigma*Cn4(j-1);
    end
        Cnp14(ny(4)) =   Cn4(ny(4))+sigma*(Cn4(ny(4)-1)-Cn4(ny(4)));

        % 5 % Numerical solution - note that we only need to solve for j=2 to nr-1
    % since the first and last points are fixed by the boundary condition
    sigma   =  D*dt/(dy(5)^2);
    for j   =   2:ny(5)-1
        Cnp15(j) =   sigma*Cn5(j+1)+(1-2*sigma)*Cn5(j)+sigma*Cn5(j-1);
    end
```

```
        Cnp15(ny(5)) =   Cn5(ny(5))+sigma*(Cn5(ny(5)-1)-Cn5(ny(5)));

        % 6 % Numerical solution - note that we only need to solve for j=2 to nr-1
    % since the first and last points are fixed by the boundary condition
    sigma   =  D*dt/(dy(6)^2);
    for j   =   2:ny(6)-1
        Cnp16(j) =   sigma*Cn6(j+1)+(1-2*sigma)*Cn6(j)+sigma*Cn6(j-1);
    end
        Cnp16(ny(6)) =   Cn6(ny(6))+sigma*(Cn6(ny(6)-1)-Cn6(ny(6)));

    Plot analytical vs numerical solution
    if plotFlag==1
        plot(y1,Cnp11, 'r+', y2,Cnp12, 'g+', y3,Cnp13, 'b+', y4,Cnp14, 'c+',
y5,Cnp15, 'm+', y6,Cnp16,'y+',  ya, CExact, 'k-')
        xlabel('y (m)')
        ylabel('C (g/L)')
        ylim([0 50])
        title('Analytical vs Numerical Solution')
        h=legend('ny = 9','ny = 17','ny = 33','ny = 65','ny = 129','ny =
257','analytical');
        set(h,'location','northeast');
        pause(0.01);
    end

    % Copy solution to initial conditions for next iteration. Note only
    % copy the computed values otherwise you may overwrite the boundary
    % conditions!
    Cn1=Cnp11;
    Cn2=Cnp12;
    Cn3=Cnp13;
    Cn4=Cnp14;
    Cn5=Cnp15;
    Cn6=Cnp16;

End

clc
clear all
close all

plotFlag = 1;

% Spatial (x) parameters analytical
L      = 80000;                    % Size of the domain
ystart = 0;                        % Start of computational domain (m)
yend   = ystart + L;               % End of computational domain (m)
ny     = 257;                      % Spatial grid points
dy     = (yend - ystart)/(ny - 1); % Spatial step size, delta r (m)
y      = ystart: dy : yend;        % Vector of grid points

% Phyiscal parameters
phi0      = 50;                        % Temperature at r=0
phi_1     = 0;                         % Temperature at r=L
D         = 2.37*10^2;                 % Diffusion equation constant
d         = sqrt(D);                   % Square root of Diffusion constant
alpha     = 1.17*10^-6;                % Source term parameter
Ceq       = 1*10^-6;                   % Equilibrium concentration

% Temporal (t) parameters
```

```
timestart = 0;                            % Starting time (0 seconds)
timeend   = 86400*14;                     % Ending Time 14 days
dt        = 103;                          % Temporal step size, delta t (s)
nt        = (timeend - timestart)/dt + 1;  % Time grid points
time      = timestart:dt:timeend;          % Vector of times

% Initialise the solution arrays - Tn (T^n) and Tnp1 (T^(n+1)) and exact
% solution TExact
Cn = zeros(1,ny);
Cnp1 = zeros(1,ny);
CN = zeros(1,ny);
CNp1 = zeros(1,ny);

% Enforce the boundary conditions - These will never change in the simulation
Cn(1) = 50;
Cnp1(1) = 50;
CN(1) = 50;
CNp1(1) = 50;

%% Main solution loop

% Loop through time
for i=2:nt                      % loop starts from 2 (not 1) to ensure that the
analytical and numerical solutions are compared at the same time point in the mean
error calc below

    % Numerical solution without the source term - note that we only need to solve
for j=2 to nr-1
    % since the first and last points are fixed by the boundary condition
    sigma   =  D*dt/(dy^2);
    for j   =   2:ny-1
        Cnp1(j) =   sigma*Cn(j+1)+(1-2*sigma)*Cn(j)+sigma*Cn(j-1);
    end
        Cnp1(ny) =   Cn(ny)+sigma*(Cn(ny-1)-Cn(ny));

    % Numerical solution with the source term
    sigma   =  D*dt/(dy^2);
    for j   =   2:ny-1
        CNp1(j) =   alpha*dt*(Ceq - CN(j)) + sigma*CN(j+1)+(1-
2*sigma)*CN(j)+sigma*CN(j-1);
    end
        CNp1(ny) =   alpha*dt*(Ceq - CN(j)) + CN(ny) + sigma*(CN(ny-1)-CN(ny));


    % Plot analytical vs numerical solution
    if plotFlag==1
        plot(y,Cnp1, 'r+', y,CNp1, 'bo')
        xlabel('y (m)')
        ylabel('C (g/L)')
        ylim([0 50])
        title('Addition of Source Term')
        h=legend('Without source term', 'With source term');
        set(h,'location','northeast');
        pause(0.01);
    end

    % Copy solution to initial conditions for next iteration. Note only
    % copy the computed values otherwise you may overwrite the boundary
    % conditions!
```

```matlab
    Cn=Cnp1;
    CN=CNp1;

end

clc
clear all
close all

plotFlag = 1;

% Spatial (x) parameters analytical
L      = 80000;                  % Size of the domain
ystart = 0;                      % Start of computational domain (m)
yend   = ystart + L;             % End of computational domain (m)
ny     = 257;                    % Spatial grid points
dy     = (yend - ystart)/(ny - 1);  % Spatial step size, delta r (m)
y      = ystart: dy : yend;      % Vector of grid points

% Phyiscal parameters
phi0      = 50;                      % Temperature at r=0
phi_1     = 0;                       % Temperature at r=L
D         = 2.37*10^2;               % Diffusion equation constant
d         = sqrt(D);                 % Square root of Diffusion constant
alpha     = [1.17*10^-6, 2*10^-5];   % Source term parameter
Ceq       = 1*10^-6;                 % Equilibrium concentration

% Temporal (t) parameters
timestart = 0;                          % Starting time (0 seconds)
timeend   = 86400*100;                   % Ending Time 14 days
dt        = 103;                        % Temporal step size, delta t (s)
nt        = (timeend - timestart)/dt + 1;   % Time grid points
time      = timestart:dt:timeend;       % Vector of times

% Initialise the solution arrays - Tn (T^n) and Tnp1 (T^(n+1)) and exact
% solution TExact
Cn = zeros(1,ny);
Cnp1 = zeros(1,ny);
CN = zeros(1,ny);
CNp1 = zeros(1,ny);

% Enforce the boundary conditions - These will never change in the simulation
Cn(1) = 50;
Cnp1(1) = 50;
CN(1) = 50;
CNp1(1) = 50;

%% Main solution loop

% Loop through time
for i=2:nt                   % loop starts from 2 (not 1) to ensure that the
analytical and numerical solutions are compared at the same time point in the mean
error calc below

    % Numerical solution with the old alpha 1.7e-6
    sigma   =  D*dt/(dy^2);
    for j   =   2:ny-1
        CNp1(j) =   alpha(1)*dt*(Ceq - CN(j)) + sigma*CN(j+1)+(1-
2*sigma)*CN(j)+sigma*CN(j-1);
```

```
    end
        CNp1(ny) =   alpha(1)*dt*(Ceq - CN(j)) + CN(ny) + sigma*(CN(ny-1)-CN(ny));

    % Numerical solution with the new alpha 2e-5source term
    sigma   =  D*dt/(dy^2);
    for j   =   2:ny-1
        Cnp1(j) =   alpha(2)*dt*(Ceq - Cn(j)) + sigma*Cn(j+1)+(1-
2*sigma)*Cn(j)+sigma*Cn(j-1);
    end
        Cnp1(ny) =   alpha(2)*dt*(Ceq - Cn(j)) + Cn(ny) + sigma*(Cn(ny-1)-Cn(ny));


%     % Plot analytical vs numerical solution
%     if plotFlag==1
%         plot(y,CNp1, 'c+', y,Cnp1, 'mo')
%         xlabel('y (m)')
%         ylabel('C (g/L)')
%         ylim([0 50])
%         title('Varying alpha')
%         h=legend('1.7e-6', '2e-5');
%         set(h,'location','northeast');
%         pause(0.01);
%     end

    % Copy solution to initial conditions for next iteration. Note only
    % copy the computed values otherwise you may overwrite the boundary
    % conditions!
    Cn=Cnp1;
    CN=CNp1;

end
%meanError = mean(abs(CExact-Cnp1)./CExact);
```