Journal of Applied Engineering Mathematics

Volume 11, December 2024

Temperature Modeling of a Cement Pipe

Cameron Stoker and Ryan Jurges

Mechanical Engineering Department
Brigham Young University
Provo, Utah 84602
stoke29@byu.edu, rjurges@byu.edu

Abstract

Cement pipes are commonly used in water management projects, often buried underground but sometimes exposed to air. Understanding the temperature distribution is crucial, especially to assess the risk of freezing in above-ground pipes under cold conditions. Using the heat equation, modeled as a long hollow cylinder with type 1 and 2 boundary conditions, the method of separation of variables reveals that pipe temperatures reach steady state in less than a second. Therefore, water in above-ground cement pipes is unlikely to freeze unless a significant external temperature gradient exists at the exterior of the pipes.

Nomenclature

- u = Temperature distribution
- u_{ss} = Steady state temperature distribution
- U = Transient temperature distribution
- t = Time
- r = Radial coordinate
- k = Conduction coefficient
- a = Thermal diffusivity
- T_0 = Initial temperature of pipe
- T_1 = Temperature at interior of pipe
- F_2 = Temperature gradient at exterior of pipe
- r_1 = Interior diameter
- r_2 = Exterior diameter

Introduction

The purpose of modeling is to identify the temperature distribution of the interior of a pipe. This is useful information for structures and energy systems that pump water in above ground pipes for various purposes. This information can determine the possibility and timing of ice forming on the outside of pipes in cold weather conditions. Pipe interior temperatures often match that of the flowing fluid closely because of high convection conditions. Therefore, it is often more useful to know exterior pipe temperatures. The solution can also be altered to determine the heat loss from the pumping fluid to the environment. The solutions are simplified, and can be altered with differing boundary conditions to obtain different information.

Model

Heat Equation

To model a cement pipe in above-ground convective conditions, it is represented with a 1-D homogeneous heat equation. The representation is simplified to a long, hollow cylinder and it is assumed that the temperature variation is only significant in the radial direction. Therefore, temperature dependence with angular and height coordinates are neglected. The heat equation in cylindrical coordinates is shown below in Equation 1.

$$\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} = \frac{1}{a^2}\frac{\partial u}{\partial t} \tag{1}$$

The boundary condition for the interior of the pipe is assumed to be Dirichlet, as seen in Equation 2. This represents a constant temperature at the interior due to high convection between the water and the pipe. For the exterior, there is a constant temperature gradient as represented by Equation 3, which is a Neumann boundary condition.

$$u(r_1, t) = T_1 \tag{2}$$

$$\frac{\partial u(r_2, t)}{\partial r} = F_2 \tag{3}$$

Initial conditions, seen in Equation 4, of the pipe are assumed to be at room temperature at t=0.

$$u(r, 0) = u_0(r) = T_0 \tag{4}$$

Solution

In the method of separation of variables, the steady state solution and transient solutions are found separately. They are then combined using Equation 5.

$$u(r, t) = u_{ss}(r) + U(r, t) \tag{5}$$

To find the steady state solution, the general solution of the heat equation without a time variable is found. This corresponds to Equation 6 where c_1 and c_2 depend on the boundary conditions and are shown in Equations 7 and 8 respectively.

$$u_{ss}(r) = c_1 ln(r) + c_2 \tag{6}$$

$$c_1 = r_2 F_2 \tag{7}$$

$$c_2 = T_1 - r_2 F_2 \ln r_1 \tag{8}$$

The transient solution is defined by rearranging Equation 5 to get Equation 9. Because the boundary conditions for the steady state solution and full solution are the same, this results in homogeneous boundary conditions for the transient problem. The new boundary conditions are shown in Equations 10 and 11. The initial condition is also modified and is shown in Equation 12.

$$U(r, t) = u(r, t) - u_{ss}(r) \tag{9}$$

$$U(r_1, t) = u(r_1, t) - u_{ss}(r_1) = T_1 - T_1 = 0 \tag{10}$$

$$\frac{\partial U(r_2, t)}{\partial r} = \frac{\partial u(r_2, t)}{\partial r} - \frac{\partial u_{ss}(r_2)}{\partial r} = F_2 - F_2 = 0 \tag{11}$$

$$U(r, 0) = u(r, 0) - u_{ss}(r) = T_0 - u_{ss}(r) \tag{12}$$

We can now solve for the transient solution by solving the basic problem with the method of separation of variables. First, it is assumed that the transient solution is of the form given in Equation 14. Equation 15 is yielded when substituting Equation 14 into Equation 13. Since each side of the equation depends on a single variable, it can be concluded that they are equal to a constant μ.

$$\frac{\partial^2 U}{\partial r^2} + \frac{1}{r}\frac{\partial U}{\partial r} = \frac{1}{a^2}\frac{\partial U}{\partial t} \tag{13}$$

$$U(r, t) = R(r)T(t) \tag{14}$$

$$\frac{R''}{R} + \frac{1}{r}\frac{R'}{R} = \frac{1}{a^2}\frac{T'}{T} = \mu \tag{15}$$

When solving Equation 15 for R, this creates an eigenvalue problem that can be solved using the generalized bessel equation, which yields eigenfunctions given by Equation 16 and the characteristic Equation 17 for its eigenvalues where the eigenvalues are defined by Equation 18.

$$R_n(r) = \frac{J_0(\lambda_n r)}{J_0(\lambda_n r_1)} - \frac{Y_0(\lambda_n r)}{Y_0(\lambda_n r_1)} \tag{16}$$

$$-J_0(\lambda r_1)Y_1(\lambda r_2) + J_1(\lambda r_2)Y_0(\lambda r_1) = 0 \tag{17}$$

$$\mu_n = -\lambda_n^2 \tag{18}$$

Now that the eigenvalues have been found, solving for T becomes easy. Solving Equation 15 for T is done by using the method of separation of variables for ordinary differential equations. This yields Equation 19.

$$T_n(t) = e^{-a^2\lambda_n^2 t} \tag{19}$$

The solution is then found by using Equation 20 where a_n, $R_n(r)$, and $T_n(t)$ are given by Equations 21, 16, 19 respectively.

$$U(r,t) = \sum_{i=1}^{\infty} a_n R_n(r) T_n(t) \tag{20}$$

$$a_n = \frac{\int_{r_1}^{r_2} U(r,0) R_n(r) r dr}{\int_{r_1}^{r_2} R_n(r)^2 r dr} \tag{21}$$

The full solution for the problem can then be found by substituting Equations 6 and 20, which yields Equation 22 below.

$$u(r,t) = (r_2F_2)ln(r) + T_1 - r_2F_2 \ln r_1 + \sum_{i=1}^{\infty} a_n R_n(r) T_n(t) \tag{22}$$

Conclusions

Using summations to solve the heat equation is a valuable method for modeling temperature in cylindrical systems. The figures clearly show that a cement pipe's temperature quickly reaches steady-state conditions. An exterior temperature gradient of -3000 K/m represents an exceptionally high heat flux, likely only in extreme cold environments. Since power plants often discharge wastewater at

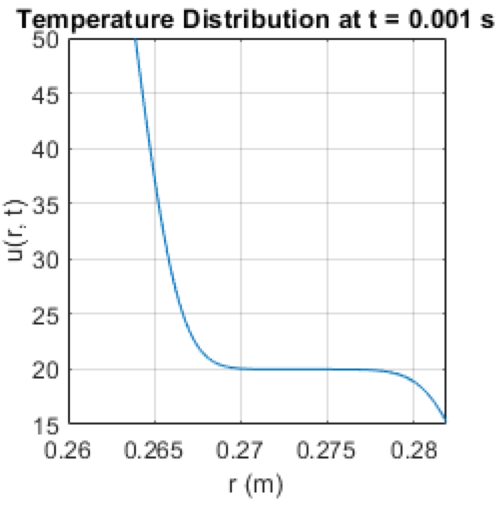

Figure 1: Radial temperature distribution of the pipe modeled shortly after the initial condition. Values include common specifications of a 10-inch cement pipe with $T_1 = 50$, $T_0 = 20$, and $F_2 = -3000$.

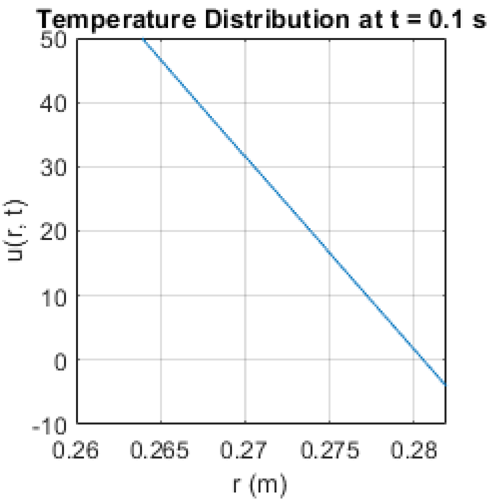

Figure 2: Radial pipe temperature distribution at steady state. Conditions are the same as the previous figure.

around 50°C, a cement pipe would only face a risk of ice buildup under severe winter conditions at this temperature. Boundary conditions can be adjusted in the model if other physical processes are to be modeled.

Acknowledgements

We would like to express heartfelt gratitude to Dr. Soloviev for his compilation of notes and teaching about transient initial-boundary value problems.

Appendix

MATLAB Code

```
clf, clc

% Value Inputs

r1 = 0.2639; % m, pipe specs

r2 = 0.2819; % m

T1 = 50; % Inside fluid temperature

t2 = -3000; % Outside fluid
    temperature gradient

C = 20; % Initial pipe temperature

a = 0.002; % Thermal diffusivity
    in heat equation

num = 40; % Number of summations
    in transient solution

d = 173; % Set as approximate
    x-distance between eigenvalues

t_value = 0.001;

%% Steady State solution

c1 = t2*r2;
c2 = T1 - c1 * log(r1);
Uss = @(r) c1 * log(r) + c2;

% Plot Steady State Solution
r_values = linspace(r1, r2, 100);  % Define
    r-values for plotting

figure(1)
plot(r_values, Uss(r_values))

%% Transient Solution

% Finding Eigenvalues: Characteristic Equation
char = @(l) - besselj(0, l*r1)
    .* bessely(1, l*r2) + besselj(1, l*r2)
    .* bessely(0, l*r1);

% Plot to visualize
l_range = linspace(60, 2000, 1000); % Avoid
    zero (singular point)
char_vals = arrayfun(char, l_range);
plot(l_range, char_vals); % Plot
    to visualize solutions
yline(0)

% Find eigenvalues: Roots of
    Characteristic Equation
lam = zeros(num, 1);
lam(1) = fzero(char, [0.01 d*1.1]);

for j = 2:num
    % Find next root interval,
        somewhere around the last one +d
    int = [lam(j-1)+d/3 (lam(j-1)+d*1.4)];
    lam(j) = fzero(char, int);
end
% disp(lam);


% Transient Solution accumulation
% Initialize U as a function handle
U = @(r, t) 0;

for n = 1:num % Loop to accumulate
    terms in the series solution

    % Eigenfunctions Rn
    Rn = @(r) besselj(0, lam(n)*r)/
        besselj(0, lam(n)*r1)
        - bessely(0, lam(n)*r)/
        bessely(0, lam(n)*r1);

    % Defining Tn
    Tn = @(t) exp(-a .* lam(n)^2 .* t);

    % Finding summation term an
    int_top = @(r) (C - Uss(r)) .* Rn(r) .* r;
    int_bot = @(r) Rn(r).^2 .* r;
    an = integral(int_top, r1, r2) /
        integral(int_bot, r1, r2);

    % Define the nth term in
        the series
    Un = @(r, t) an .* Rn(r)
        .* Tn(t);  % Ensure Rn
        and Tn are evaluated

    % Accumulate each term into U
    U = @(r, t) U(r, t) + Un(r, t);
        % Update U with the new term
end

%% Solution

% Ensure uss and U are evaluated numerically
u = @(r, t) double(Uss(r)) + double(U(r, t));

% Create a plot of the
    solution U(r, t) at a specific time
r_values = linspace(r1, r2, 100);
    % Define r-values for plotting

u_values = u(r_values, t_value);
    % Compute u(r, t)

% Plot the solution U(r, t)
    at the given t
figure(2);
figure('Position', [100, 100, 250, 250]);
plot(r_values, u_values);
    % Line plot with specified width
xlabel('r (m)');   % Label for x-axis
ylabel('u(r, t)'); % Label for y-axis
title(['Temperature Distribution at
    t = ', num2str(t_value), ' s']);
grid on;          % Display grid
```