Contents

1 Introduction

1.1 Adobe Huts

In climates characterized by hot days and cold nights, adobe huts are used to mitigate the effects off extreme external temperatures on the inside of the hut. The walls require a large and relatively long input of heat from the sun and from the surrounding air before they warm through to the interior. After the sun sets and the temperature drops, the wall will continue to transfer the heat it built up during the day to the interior for several hours. This causes a time lag effect on the internal temperature of the hut relative to the outside.

2 Geometry

- Spherical coordinates because hut is a half sphere
- Walls are uniform so there is no angular dependence based on wall composition
- Hut is small so there is no angular dependence on how much sun/heat the wall is getting
- Hut is a half sphere so that interface of wall and ground is parallel to the radius, thus removing angular dependence

Temperature in the hut depends only on distance from the center of the hut, i.e. the radius.

3 Model Derivation

3.1 The Outside

For the sake of simplicity, the temperature outside the hut is assumed to be uniform and follow the following pattern base on time

$$T_{ext}(t) = max(0, \sin(\pi t))$$

Where t is the time scale.

3.2 The Hut

General Heat Equation:

$$\rho c_p \frac{\partial T}{\partial t} = k\nabla^2 T$$

Where

- ρ is the density of the material in $\frac{kg}{m^3}$
- c_p is the heat capacity of the material in $\frac{J}{kg \cdot K}$
- k is the thermal conductivity of the material in $\frac{W}{m \cdot K}$

Translating this to spherical coordinates and dropping any θ and ϕ dependent terms, as this model has no angular dependence, gets

$$\rho c_p \frac{\partial T}{\partial t} = k\left(\frac{\partial^2 T}{\partial r^2} + \frac{2}{r}\frac{\partial T}{\partial r}\right)$$
$$\frac{\partial T}{\partial t} = \left(\frac{k}{\rho c_p}\right)\left(\frac{\partial^2 T}{\partial r^2} + \frac{2}{r}\frac{\partial T}{\partial r}\right)$$

and from these we get the heat equation for both the wall of the hut and the air inside of it.

$$\frac{\partial T_{wall}}{\partial t} = \left(\frac{k}{\rho c_p}\right)_{wall}\left(\frac{\partial^2 T}{\partial r^2} + \frac{2}{r}\frac{\partial T}{\partial r}\right)$$
$$\frac{\partial T_{air}}{\partial t} = \left(\frac{k}{\rho c_p}\right)_{air}\left(\frac{\partial^2 T}{\partial r^2} + \frac{2}{r}\frac{\partial T}{\partial r}\right)$$

3.3 Boundary Conditions

Since the model is being solved numerically, there are certain boundary conditions that need to be put in place to make sure that the model doesn't stray too far from reality.

The first is that the temperature of the air where it touches the wall and the temperature of the wall where it touches the air must be the same. This gives that

$$BC1 := T_{wall}(r_a, t) = T_{air}(r_a, t)$$

Where r_a is the radius of the inside of the hut.

The second, and related condition is that the heat flux at the interface of the wall and air has to be the same through both the wall and air.

$$BC2 := k_{wall}\frac{\partial T_{wall}}{\partial t}\bigg|_{r_a} = k_{air}\frac{\partial T_{air}}{\partial t}\bigg|_{r_a}$$
$$BC2 := \frac{\partial T_{wall}}{\partial t}\bigg|_{r_a} = \frac{k_{air}}{k_{wall}}\frac{\partial T_{air}}{\partial t}\bigg|_{r_a}$$

The third is that the heat flux at the interface of the wall and the surroundings is proportional to the difference in temperature between the surroundings and the wall at the interface.

$$BC3 := k_{wall}\frac{\partial T_{wall}}{\partial t} = h(T_{wall}(r_b, t) - T_{ext}(t))$$
$$BC3 := \frac{\partial T_{wall}}{\partial t} = \frac{h}{k_{wall}}(T_{wall}(r_b, t) - T_{ext}(t))$$

Where h is the heat transfer coefficient of the material the wall is made of and r_b is the radius of the hut from center to outer edge.

The final is a regularity condition at the center of the hut.

$$BC4 := \frac{\partial T_{air}}{\partial r}\bigg|_{0} = 0$$

3.3.1 List of Boundary conditions

$$BC1 := T_{wall}(r_a, t) = T_{air}(r_a, t)$$
$$BC2 := \frac{\partial T_{wall}}{\partial t}\bigg|_{r_a} = \frac{k_{air}}{k_{wall}}\frac{\partial T_{air}}{\partial t}\bigg|_{r_a}$$
$$BC3 := \frac{\partial T_{wall}}{\partial t} = \frac{h}{k_{wall}}(T_{wall}(r_b, t) - T_{ext}(t))$$
$$BC4 := \frac{\partial T_{air}}{\partial r}\bigg|_{0} = 0$$

3.4 Nondimensionalization

Define:

$$t = \tau \tilde{t}$$
$$r = L\tilde{r}$$
$$T = T_0 + (\Delta T)\theta$$

So the equation worked out in 3.2 becomes:

$$\frac{\Delta T}{\tau}\frac{\partial \theta}{\partial \tilde{t}} = \frac{k}{\rho c_p}\frac{\Delta T}{L^2}\left(\frac{\partial^2 \theta}{\partial \tilde{r}^2} + \frac{2}{\tilde{r}}\frac{\partial \theta}{\partial \tilde{r}}\right)$$
$$\frac{\partial \theta}{\partial \tilde{t}} = \frac{k}{\rho c_p}\frac{\tau}{L^2}\left(\frac{\partial^2 \theta}{\partial \tilde{r}^2} + \frac{2}{\tilde{r}}\frac{\partial \theta}{\partial \tilde{r}}\right)$$

Drop Tildes

$$\frac{\partial \theta}{\partial t} = \frac{k}{\rho c_p}\frac{\tau}{L^2}\left(\frac{\partial^2 \theta}{\partial r^2} + \frac{2}{r}\frac{\partial \theta}{\partial r}\right)$$
$$\frac{\partial \theta}{\partial t} = \frac{k}{\rho c_p}\frac{\tau}{L^2}\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial \theta}{\partial r}\right)$$

So the nondimensionalized equations become:

$$\frac{\partial \theta}{\partial t} = \left(\frac{k}{\rho c_p}\right)_{wall}\frac{\tau}{L^2}\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial \theta}{\partial r}\right)$$
$$\frac{\partial \theta}{\partial t} = \left(\frac{k}{\rho c_p}\right)_{air}\frac{\tau}{L^2}\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial \theta}{\partial r}\right)$$

3.4.1 Variable Values and Parameters

Variable	Description	Value	Units
k_{wall}	thermal conductivity of brick	0.75	$Wm^{-1}K^{-1}$
k_{air}	thermal conductivity of air	4e-2	$Wm^{-1}K^{-1}$
ρ_{wall}	density of the wall/brick	1.8e+3	kgm^{-3}
ρ_{air}	density of the air	1.2	kgm^{-3}
cp_{wall}	heat capacity of the wall/brick	840	$Jkg^{-1}K^{-1}$
cp_{air}	heat capacity of the air	700	$Jkg^{-1}K^{-1}$
r_b	radius of the hut (from center to external edge of wall)	3.0	m
r_a	radius of the hut (from center to internal edge of wall)	2.7	m
h	heat transfer coefficient of brick	2.0	$Wm^{-2}K^{-1}$
τ	time scale (12 hours)	4.32e+4	s

For the sake of simplicity, some numbers that remain constant are gathered together as parameters

Parameter	Equation	Value
P_1	$\left(\frac{k}{\rho c_p}\right)_{wall}\frac{\tau}{L^2}$	2.4e-e
P_2	$\left(\frac{k}{\rho c_p}\right)_{air}\frac{\tau}{L^2}$	0.23
P_3	$\frac{r_a}{r_b}$	0.9
P_4	$\frac{k_{air}}{k_{wall}}$	5.3e-2
P_5	$h\frac{r_b}{k_{wall}}$	8

3.5 Method of Lines

Method of lines involves discretizing the problem spatially and leaving it continuous temporally. In this case, the hut is divided into n 'shells' each of which represent a single 'line'. Each shell has a width $h = 1/(n-1)$. $n-1$ because the outermost shell is considered to be shell 0.

The function of the temperature in the hut $T(r,t)$ then becomes $\Phi = \{\phi_0(t), ..., \phi_{n-1}(t)$ where each $\phi_i(t)$ represent the temperature in the wall at a given radius at time t.

So the heat equations become:

$$\frac{\partial \phi_i}{\partial t} = P_1\left(\frac{\partial^2 \phi_i}{\partial r^2} + \frac{2}{r}\frac{\partial \phi_i}{\partial r}\right) \to P_3 < r < 1 \quad , \quad 0 < i < n \cdot P_3$$
$$\frac{\partial \phi_i}{\partial t} = P_2\left(\frac{\partial^2 \phi_i}{\partial r^2} + \frac{2}{r}\frac{\partial \phi_i}{\partial r}\right) \to 0 < r < P_3 \quad , \quad n \cdot P_3 < i < n-1$$

Which when expanded and simplified using finite differences, gets:

$$\frac{\partial \phi_i}{\partial t} = P_1\left(\frac{1}{rh}(\phi_{i+1} - \phi_{i-1}) + \frac{1}{h^2}(\phi_{i+1} - 2\phi_i + \phi_{i-1})\right) \to P_3 < r < 1 \quad , \quad 0 < i < n \cdot P_3$$
$$\frac{\partial \phi_i}{\partial t} = P_2\left(\frac{1}{rh}(\phi_{i+1} - \phi_{i-1}) + \frac{1}{h^2}(\phi_{i+1} - 2\phi_i + \phi_{i-1})\right) \to 0 < r < P_3 \quad , \quad n \cdot P_3 < i < n-1$$

And then for BC1 and BC2, you get:

$$\phi_{n \cdot P3} = \frac{P_4\phi_{n \cdot P3+2} - \phi_{n \cdot P3-2}}{P_4 - 1}$$

BC3 becomes

$$\phi_0 = T_{ext}(t) + \frac{T_{ext}(t) - \phi_1}{2P_5 h}$$

And BC4 becomes

$$\phi_{n-1} = \phi_{n-2}$$

3.6 Runge-Kutta

Since the actual equations for $\phi_i(t)$ aren't known, the change in temperature at each step is computed using Runge-Kutta approximation.

$$\phi_i(t + \Delta t) = \phi_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Where

$$k_1 = \frac{\partial \phi_i}{\partial t}$$

$$k_2 = \frac{\partial}{\partial t}\left(\phi_i + \frac{h \cdot k_1}{2}\right)$$

$$k_3 = \frac{\partial}{\partial t}\left(\phi_i + \frac{h \cdot k_2}{2}\right)$$

$$k_4 = \frac{\partial}{\partial t}\left(\phi_i + h \cdot k_3\right)$$

4 Appendices

4.1 Model Diagram

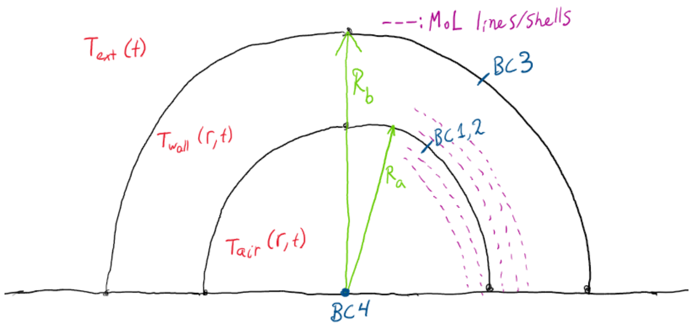

Figure 1: Rough Sketch of the Hut

4.2 Plots

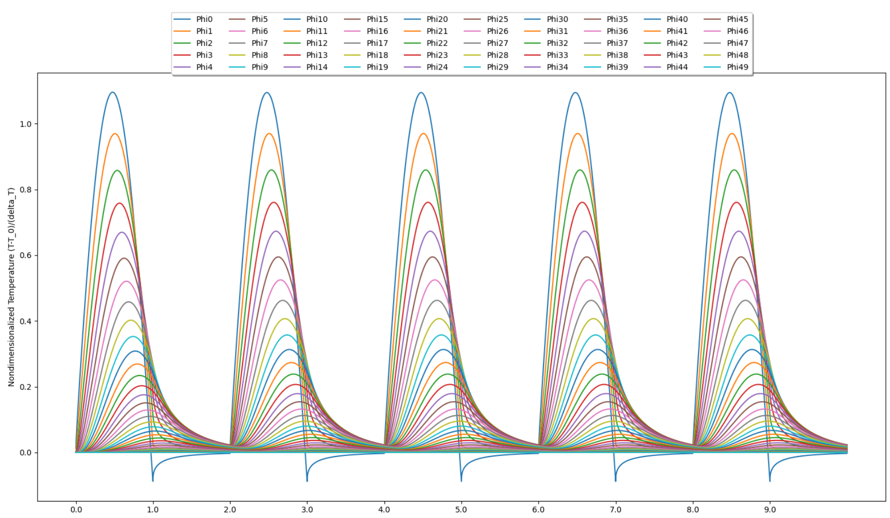

Figure 2: 2D Plot of temperature in each shell vs. Time

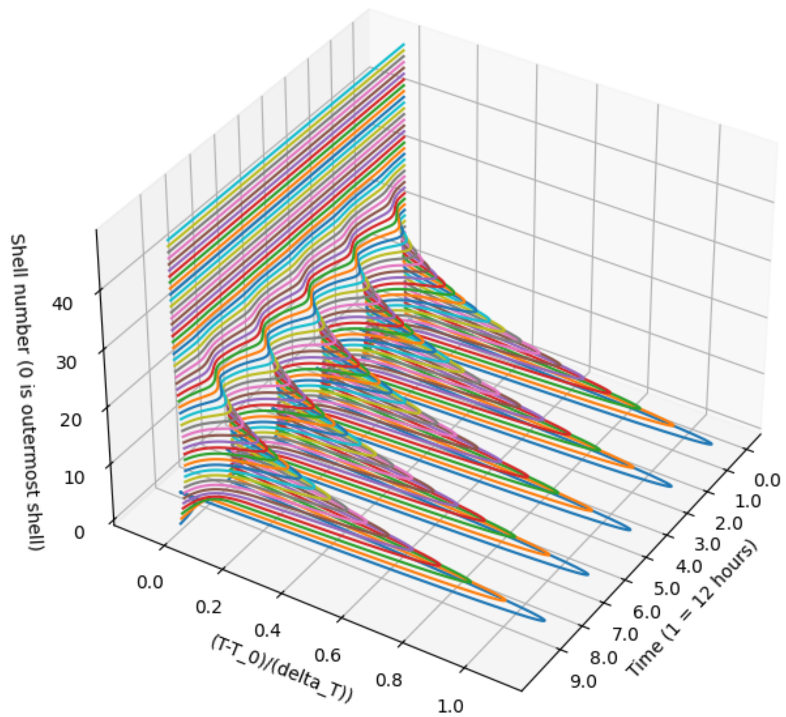

Figure 3: 3D Plot of temperature in each shell vs. Time

4.3 Code

Final_Project\Adobe_Hut.py

```python
# By: Daniel Shklyarman
#     100851439

# A simplified model of heat transfer
# in an adobe hut

# The adobe hut is made out of brick and has no windows
# It is assumed to be small enough that it always receives
# an equal amount of sunlight/heat on all sides so that there
# is no angular dependence

# Imports
import numpy as np
import math
import matplotlib.pyplot as plt
import time

# Physical Constants

# | Name     | Description                                        | Value    | Units          |
# |---------|--------------------------------------------------|---------|---------------|
# | k_wall   | thermal conductivity of brick                      | 0.75     | W m^-1 K^-1    |
# | k_air    | thermal conductivity of air                        | 4e-2     | W m^-1 K^-1    |
# | rho_wall| density of the wall/brick                          | 1.8e+3   | kg m^-3        |
# | rho_air  | density of the air                                 | 1.2      | kg m^-3        |
# | cp_wall  | heat capacity of the wall/brick                    | 840      | J kg^-1 K^-1   |
# | cp_air   | heat capacity of the air                           | 700      | J kg^-1 K^-1   |
# | R_b      | radius of the hut (from center to outside          | 3.0      | m              |
# | R_a      | radius of the hut (from center to internal wall)   | 2.7      | m              |
# | h        | heat transfer coefficient of brick                 | 2.0      | W  m^-2  K^-1  |
# | tau      | time scale (12 hours)                              | 4.32e+4  | s              |
# |---------|--------------------------------------------------|---------|---------------|

# number of 'lines' through the hut
n = 50

# Parameters

# P1 = [k/(rho*cp)]_wall * [tau/(R_b)^2]
P1 = 2.4e-3
# P2 = [k/(rho*cp)]_air * [tau/(R_b)^2]
P2 = 0.23
# P3 = R_a / R_b
P3 = 0.9
# P4 = k_air / k_wall
P4 = 5.3e-2
# P5 = h * R_b / k_wall
P5 = 8

# Width of a 'line'
# r0 = 1
# r0 - (N-1)*delta_r = r_(N-1) = 0
h = 1/(n-1)
```

```python

# External Temperature as a function of time
def u_ext(t):
    return max(0, np.sin(np.pi * t))

# Array of 'lines'
# the phi at Phi[0] is the external edge
# progressive phi's go closer to the center
Phi = np.zeros(n)

# Set temperatures at time = 0
T_init = 0
for i in range(len(Phi)):
    Phi[i] = T_init

# Number of measurement taken per tau
# aka dt
resolution = 1000

# Number of time steps
it_number = 10000

# Array to store temperature values over time
Phi_history = np.zeros([n,it_number-1])

# Finite Difference estimate of the derivative wrt time
def dphi_dt(temp_vec, n, i, parameter):
    temp_i_minus_1 = temp_vec[0]
    temp_i = temp_vec[1]
    temp_i_plus_1 = temp_vec[2]

    curr_r = ((n-1-i)*h)**2

    dphi_dt = parameter * (
                          ((n-1)**2)*(temp_i_minus_1 - 2*temp_i + temp_i_plus_1) +
                          ((n-1)/curr_r)*(temp_i_plus_1 - temp_i_minus_1)
                          )

    return dphi_dt

# Runge-Kutta method, only returns the increment not
# the full new estimate, hence the '+=' in the loop
def Runge_Kutta(temp_vec, dphi_dt, h, i, n, parameter):
    k1 = dphi_dt(temp_vec, n, i, parameter)

    k2_temp_vec = temp_vec + [temp*(h*k1)/2 for temp in temp_vec]
    k2 = dphi_dt(k2_temp_vec, n, i, parameter)

    k3_temp_vec = temp_vec + [temp*(h*k2)/2 for temp in temp_vec]
    k3 = dphi_dt(k3_temp_vec, n, i, parameter)

    k4_temp_vec = temp_vec + [temp*(h*k3) for temp in temp_vec]
    k4 = dphi_dt(k4_temp_vec, n, i, parameter)

    return (h/6) * (k1 + 2*k2 + 2*k3 + k4)

```

```
# Live Plotting stuff
# x = np.linspace(0,50,50)
# y = Phi

# plt.ion()

# fig,ax = plt.subplots(figsize=(10,5))
# line, = ax.plot(x,y)

# plt.title("Heat in an Adobe Hut")

# plt.xlabel("Lines from the outside in")
# plt.ylabel("Temperature")

# plt.ylim(-1, 2)

# The actual fancy stuff
for t in range(1, it_number):
    for i in range(len(Phi)):
        # BC3
        if i == 0:
            Phi[i] = u_ext(t/resolution) + (u_ext(t/resolution) - Phi[1])/(2*P5*h)
            Phi_history[i,t-1] = Phi[i]

        # Heat equation in wall
        elif i < n*P3:
            temp_vec = [Phi[i-1],Phi[i],Phi[i+1]]
            Phi[i] += Runge_Kutta(temp_vec, dphi_dt , h, i, n, P1)
            Phi_history[i,t-1] = Phi[i]

        # BC1/2
        elif i == (n - n*P3):
            Phi[i] = (P4*Phi[i+1] - Phi[i-1]) / (P4-1)
            Phi_history[i,t-1] = Phi[i]

        # Heat equation in the air
        elif i > n-n*P3 and i < n-1:
            temp_vec = [Phi[i-1],Phi[i],Phi[i+1]]
            Phi[i] += Runge_Kutta(temp_vec, dphi_dt , h, i, n, P2)
            Phi_history[i,t-1] = Phi[i]

        # BC4
        else:
            Phi[i] = Phi[i-1]
            Phi_history[i,t-1] = Phi[i]

    if t < 100:
        print(Phi_history[:,t-1][:5])

    # Also Live Plotting Stuff
    # new_y = Phi

    # line.set_xdata(x)
    # line.set_ydata(new_y)

    # fig.canvas.draw()
```

```python

    # fig.canvas.flush_events()

    # time.sleep(0.01)


# Ask user for what plot they want
plot = input('What kind of plot would you like? \n * 2             = 2d plot \n * 3 = 3d plot \n * [other input] = no plot \n')

# Create 2D plot
if plot == '2':
    # Ask user if they want the legend since the legend is very crowded due to havin n lines
    legend = input("Legend or no legend? (Y/N , other input assumes N)")

    # Create Plot
    for i in range(0, n):
        plt.plot(np.linspace(0,it_number, it_number-1), Phi_history[i], label=f'Phi{i}')

    if legend == 'Y':
        plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.15), ncol=10, fancybox=True, shadow=True)

    # Make the plot a bit nicer to look at/read
    plt.xticks(np.arange(0,it_number, step=resolution), np.arange(0, it_number/resolution, step=1))
    plt.xlabel('Time (1 = 12 hours)')
    plt.ylabel('Nondimensionalized Temperature (T-T_0)/(delta_T)')

    plt.show()

# Create 3D plot
elif plot == '3':
    # Create Plot
    ax = plt.figure().add_subplot(projection='3d')
    for i in range(0, n):
        ax.plot(np.linspace(0,it_number, it_number-1), Phi_history[i], zs=i, label=f'Phi{i}')

    # Make the plot a but nicer to look at/read
    ax.set_xticks(np.arange(0,it_number, step=resolution), np.arange(0, it_number/resolution, step=1))
    ax.set_xlabel('Time (1 = 12 hours)')
    ax.set_ylabel('(T-T_0)/(delta_T))')
    ax.set_zlabel('Shell number (0 is outermost shell)')

    plt.show()

else:
    print("Have a nice day.")
```