A Study of Cooking Temperatures Using PDEs

Mark Do, Mckale Chung, Carmelli Dao, Miekale Smith, Odette Beaudin

April 7, 2025

The heat equation is a fundamental topic within the realm of PDEs. The use of the 1D heat equation enables the determination of analytical solutions on simple domains and the ability to model complex engineering problems. Depending on the application, the heat equation can be modeled using spherical or cartesian coordinates. A simpler application of the heat equation could be used to determine how the temperature of a rod may vary along said rod. A representation of the rod can be seen in Figure 1

The 1D heat equation is a second-order linear partial differential equation. This equation is derived from the principles of the conservation of energy, Fourier's law of heat transfer, as well as the energy of a body with uniform properties. To model the heat equation, there is a need for both initial and boundary conditions (B.C). Some common boundary conditions used to model the heat equation include the Dirichlet B.C, Neumann B.C, as well as the Robin (mixed) B.C.

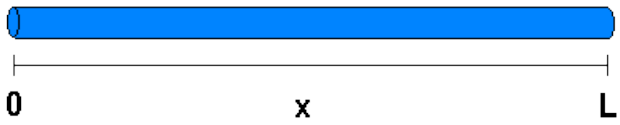


Figure 1: 2D Rod [1]

This project leverages the idea of the 1D heat equation to examine the problem of cooking temperatures for varying eggs.

Part 1: 1D Heat Equation in Spherical Coordinates

1.A Problem Definition

The general form of the one-dimensional heat equation in cartesian coordinates is modeled by:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \tag{1}$$

Where, α represents the thermal diffusivity in m/s^2 of the material. In this case the material being modeled is the chicken egg. This formula assumes that there is no heat generation and that the thermal diffusivity remains constant.

For simplicity, in this project it will be assumed that an egg is a perfectly spherical object. Therefore to properly analyze heat diffusion in the egg, the heat equation should be modeled in spherical coordinates. The three-dimensional representation of the heat equation in spherical coordinates is as follows:

$$\frac{\partial T}{\partial t} = \alpha \nabla^2 T(r, \theta, \phi) = \alpha \left(\frac{1}{r} \frac{\partial^2}{\partial r^2}(rT) + \frac{1}{r^2 \sin\theta} \frac{\partial}{\partial \theta} \left(\sin\theta \frac{\partial T}{\partial \theta} \right) + \frac{1}{r^2 \sin^2\theta} \frac{\partial^2 T}{\partial \phi^2} \right) \tag{2}$$

However, the egg is assumed to be perfectly spherical, so the equation can be simplified to:

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{1}{r^2} \frac{\partial}{\partial r} (r^2 \frac{\partial T}{\partial r}) \right) \tag{3}$$

The initial condition based on the definition of the problem is that the initial temperature, at time zero, of the egg will be equal to the refrigerator temperature in which the egg is stored. The average temperature of a refrigerator is 2 degrees Celsius.

Initial Condition: $T(r, 0) = 2$ for all r ϵ (0, R))

The perfectly spherical egg will be placed in boiling water at 100 degrees Celsius and cooked until the temperature across the entire egg reaches 80 degrees Celsius for 10 seconds. Therefore, there should be a derelict boundary condition that keeps the surface of the egg restricted to 100 degrees Celsius due to the boiling water. Additionally, in order to prevent heat flow across the center of the egg, the following Neumann Boundary conditions was added.

Dirichlet Boundary Condition: $T(R, t) = 100$ for all t $\geqslant 0$

Neumann Boundary Condition: $\frac{\partial T}{\partial r}\big|_{r=0} = 0$

1.B.1 Approximating the Material Properties and Dimensions of Spherical Eggs

To properly estimate the time for an egg to be *fully cooked*[1], the geometrical and material properties must first be established. The following section details the assumptions and processes undertaken to determine these values.

The yolk, white, and shell percentages for quail, chicken, and ostrich eggs are found below in Table . For quail eggs, an average was taken for the eggs of Gray, Brown and White Quail eggs. All the following measurements are based on averages, as factors such as the age of the mother fowl and egg size may influence the composition and ratios of the egg components [2].

Egg Type	**Shell (%)**	**Yolk (%)**	**White (%)**
Quail [4]	13.64	32.69	53.66
Chicken [2]	11.00	31.15	57.85
Ostrich [3]	19.90	26.38	53.68

Table 1: Percent composition of eggs by type.

Component	**Thermal Conduct (W/m·K)**	**Density (kg/m^3)**	**Specific Heat (J/kg·K)**
Shell	0.4560	2300	888
Yolk	0.3370	1036	2093
White	0.594	1036	3935

Table 2: Summarized Thermal Properties of Non-Frozen Egg Components from Sabliov et al [5]

The thermal properties of each type of egg was approximated by taking a weighted sum of the shell, yolk, and white. This assumes that each egg is a homogeneous solution, rather than a layered structure. It also assumes that the yolk, shell, and albumin for each egg type have the same individual thermal properties and any air inside the eggs is arbitrary. For example, to find the thermal conductivity of the quail egg model.

$$c_{quail} = 0.1364 * 0.4560 + 0.3269 * 0.3370 + 0.5366 * 0.594 = 0.491$$

The results are summarized in the table below.

Egg Type	**Thermal Conduct (W/m·K)**	**Density (kg/m^3)**	**Specific Heat (J/kg·K)**
Quail	0.491	1208.47	2917.06
Chicken	0.499	1175.04	3026.05
Ostrich	0.499	1287.1	2840.5

Table 3: Thermal properties of quail, chicken, and ostrich eggs.

To approximate the dimensions of spherical eggs, we may approximate the volume of an egg shape and find the radius that produces a sphere of the same volume. Many resources provide the length and width of an egg, along with the milliliter volume of the egg, but this does not take into account the shell. Nobua Yamamoto proposes an equation that models an egg shape [6]:

$$(x^2 + y^2)^2 = ax^3 + (a - b)xy^2 \tag{4}$$

[1]for the rest of this report, a fully cooked egg refers to an egg cooked to 80 degrees Celcius for 10 seconds, as outlined in the problem definition.

where a is the major length of the egg, and b is a term that helps describe the y-distance as x is moved from 0 to a. Yamamoto provides the equation for y as a function of x as:

$$y = \pm \frac{\sqrt{(a-b) - 2x + \sqrt{4bx + (a-b)^2}} \cdot \sqrt{x}}{\sqrt{2}} \tag{5}$$

Since the shape is symmetric over the x-axis, the tallest part of the egg will be half of the minor length (or width). The source indicates that as long as b is kept lower than a, we may assume that this function is concave down, which means the point that represents half the width will be the highest point. Therefore, to estimate the egg curve of an egg, we have the following requirements:

$$\frac{dy}{dx}(x_a) = 0$$

$$y(x_a) = \frac{1}{2} \text{ width}$$

Due to the complexity of the derivative, we used the product rule and the SciPy library to determine when both of these conditions were met. The code in Appendix A returns the unknown values of x and b that estimate the 'true' egg-shaped curve of an egg with a given length and width. Yamamoto provides a function for the volume given these parameters:

$$\text{volume} = \frac{\pi}{2}\left(\frac{(a+b)^3 a}{6b} - \frac{a^3}{6} - \frac{a^2 b}{2} - \frac{(a+b)^5 - (a-b)^5}{60b^2}\right)$$

By equating the volume found with the above equation to the volume of a sphere, the radius of each egg approximation was found. The results are summarized below:

Egg Type	Length a (cm)	Width (cm)	x at $f_{\max}$	b	Volume (cm^3)	Radius (cm)
Ostrich [7]	15	13	7.6473	4.3580	1323.98	6.8118
Quail [4]	3.5	2.7	1.8516	1.9134	13.1644	1.4647
Chicken [2]	6.7	4.3	3.4092	5.0428	57.8532	2.3993

Table 4: Summary of Egg Volume and Spherical Radius Approximations

Using this information, a corresponding graph for the shape of the quail egg was generated as below:

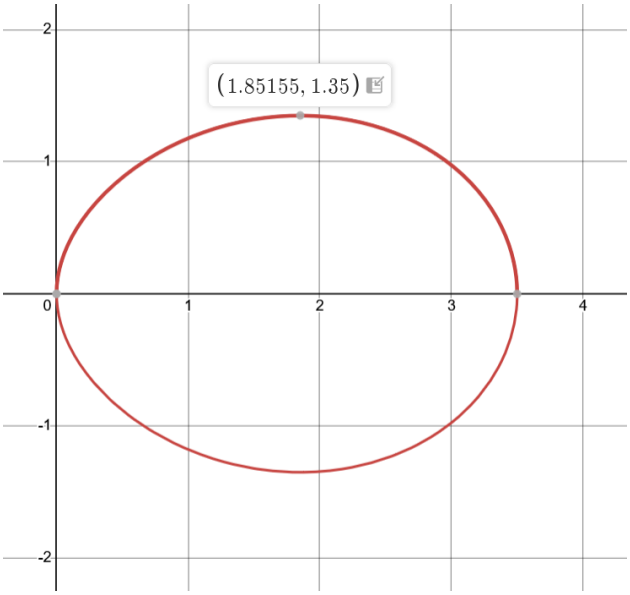


Figure 2: Graph of the Quail Egg using Yamamoto's Egg Shape Function

1.B.2 Deriving the Heat Solver

As derived in section 1.A, the partial differential equation (PDE) for heat conduction in spherical cooridinates, assuming radial symmetry is as follows:

$$\frac{\partial T}{\partial t} = \alpha(\frac{1}{r^2}\frac{\partial}{\partial r}(r^2\frac{\partial T}{\partial r}))$$

To solve this PDE numerically, the explicit finite difference method is used.
Let:

- T_i^k be the temperature at radius node i and time step k,
- $r_i = i \cdot \Delta r$, where Δr is the radial step size,
- Δt be the time step size.

The time derivative can be approximated using the Euler forward equation.

$$\left.\frac{\partial T}{\partial t}\right|_i^k \approx \frac{T_i^{k+1} - T_i^k}{\Delta t} \tag{6}$$

To discretize the spatial component, r, we use central difference approximation [8], shown by the following formula:

$$\left.\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial T}{\partial r}\right)\right|_i \approx \frac{T_{i+1}^k - 2T_i^k + T_{i-1}^k}{\Delta r^2} + \frac{T_{i+1}^k - T_{i-1}^k}{r_i \Delta r} \tag{7}$$

Then we can substitute the spatial and time components into the original PDE to get:

$$T_i^{k+1} = T_i^k + \alpha\Delta t\left[\frac{T_{i+1}^k - 2T_i^k + T_{i-1}^k}{\Delta r^2} + \frac{T_{i+1}^k - T_{i-1}^k}{r_i \Delta r}\right] \tag{8}$$

From this expression, the temperature at each $k+1$ step can be computed, using the known thermal diffusivity, α. The python function generated form this formula is shown in Appendix 2.

1.B.3 Results

Using the equation derived in 1.B.2 above, the time required to fully cook a chicken, quail, and ostrich egg can be numerically calculated. The following fiqures show the temperature (degrees Celsius) versus time (seconds) for each of the eggs.

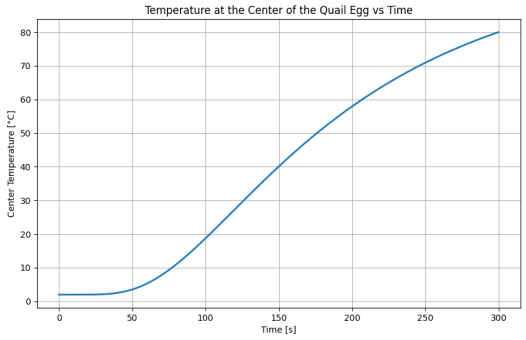


Figure 3: Temperature versus time of the quail egg in boiling water.

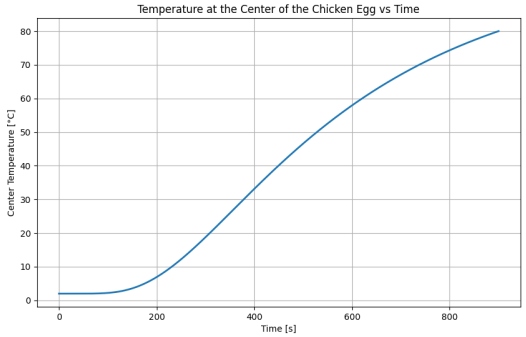


Figure 4: Temperature versus time of the chicken egg in boiling water.

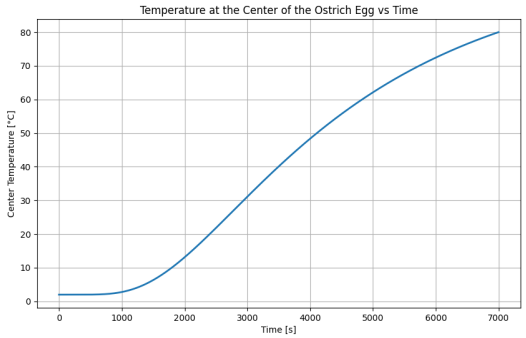


Figure 5: Temperature versus time of the ostrich egg in boiling water.

The following table summarizes the results, showing the exact time when the eggs reach the

fully cooked condition.

Egg Type	**Final Time (seconds)**	**Final Time (minutes**
Quail	275.2	4.6
Chicken	821.8	13.7
Ostrich	5951.9	99.2

Table 5: Time required to cook each egg type.

1.C Verification of Results

To verify the simulation results, a real chicken egg was placed in already boiling water and cooked for 13 minutes and 42 seconds (13.7 minutes). This was the time calculated by the solver described in section 1.B for a chicken egg to reach 80 degrees Celsius for 10 seconds.

As seen in the image below, the end result of this experiment was a hard-boiled egg. The yolk was crumbly, not runny, like a hard boiled egg, however the egg did not appear to be overcooked. An overcooked hard-boiled egg will develop a greenish or grayish ring around the yolk, which did not appear on this egg.

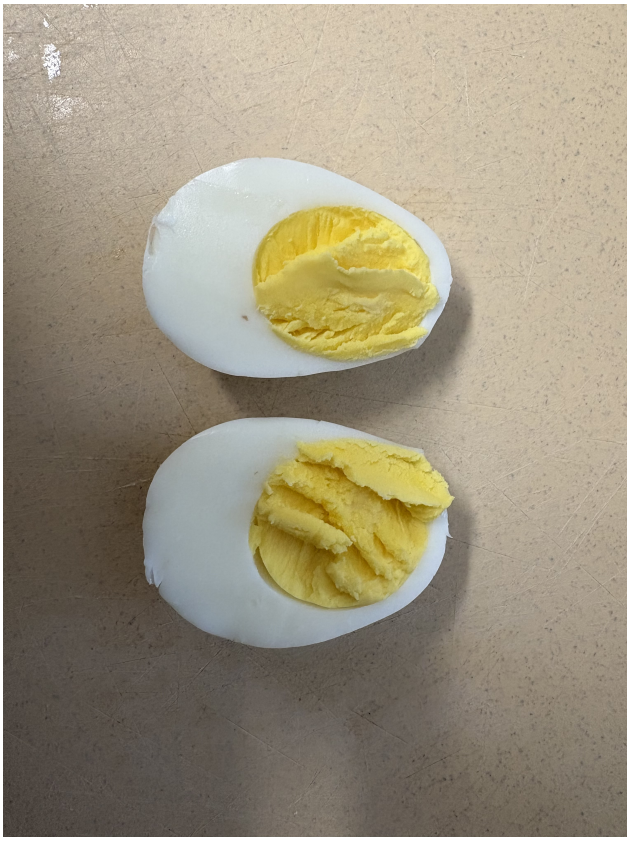


Figure 6: The chicken egg after 13.9 minutes in the boiling water.

Therefore, the solver created for this project produces a well-cooked hard-boiled egg, at least for a chicken egg. Due to cost, the quail and ostrich eggs were not tested.

1.D Lowering Energy Consumption

The following section will examine a method to decrease the consumption of fuel used to cook an egg. This section will solely examine a chicken egg, since this general method is not dependent on the type of egg chosen.

One way to lower the energy consumption of fully cooking an egg is to increase the rate of diffusion of the temperature throughout the egg, by increasing the egg's thermal diffusivity. The formula for thermal diffusivity is as follows.

$\alpha = \frac{k}{\rho Cp}$

Where k is the thermal conductivity, ρ is the density, and Cp is the specific heat of the material. Therefore, increasing the thermal diffusivity can be done by removing the shell which has a high density when compared to the yolk and egg white. This method of cooking an egg is known a poaching. For simplicity, it is assumed the egg will remain spherical when cooked in this way.

After removing the shell, the material composition of the egg will change. The new composition is shown below.

Egg Type	**Shell (%)**	**Yolk (%)**	**White (%)**
Chicken	0	35	65

Table 6: Percent composition of the chicken egg without the shell.

Then, using the method to calculate each material property as described in section 1.B, the new material values can be calculated and are shown in the following table.

The results are summarized in the table below.

Egg Type	**Thermal Conduct (W/m·K)**	**Density (kg/m^3)**	**Specific Heat (J/kg·K)**
Chicken	0.504	1036	3290.3

Table 7: Thermal properties of quail, chicken, and ostrich eggs.

By assuming new values, the time to fully cook the chicken egg becomes 780 seconds, or 13.0 minutes based off of results from the solver used in part 1.B. To compare energy required, the mass of the chicken egg with the shell was determined using the volume from Table 4. The equation below was used to determine the mass of the hard boiled egg.

$$\rho = \frac{m}{V} \tag{9}$$

The given mass then becomes 0.0599 kg. The heat equation was then used to determine the amount of energy required to completely boil the egg. The formula is shown below.

$$Q = mc\Delta T \tag{10}$$

where Q is the energy required (in kJ), m is the mass of the egg (in kg), c is the specific heat capacity (in kJ/kg·°K) and ΔT is the temperature change (in °K). Using this formula, it was determined that the energy required to boil the egg was 21, 920 J.

To calculate the energy required to poach the egg, there will be an assumption that the egg remains the same volume, since the thickness of the egg is very small. Given the new density of 1036 kg/m^3, the energy required to poach the egg then becomes 19, 326 J, which is less energy compared to boiling the egg. A summary of the results can be seen below.

Cooking Type	Thermal Conduct (W/m·K)	Density (kg/m^3)	Energy (J)
Boiling	0.499	1175	21920
Poaching	0.504	1036	19326

Table 8: Summarized Values

Part 2. 1D heat equation in Cartesian Coordinates

2.A Analytical and Numerical Solutions

The analytical solution can be found can be found in Appendix C. It gives the following solution:

$$u(x,t) = \sum_{n=1}^{N} C_n \sin(n\pi x) e^{-2(n\pi)^2 t} + 2x$$

where the Fourier coefficients C_n are defined as:

$$C_n = \begin{cases} -\dfrac{4}{\pi}, & \text{if } n = 1 \\ \dfrac{1-\cos((n+1)\pi)}{(n+1)\pi} - \dfrac{1-\cos((1-n)\pi)}{(1-n)\pi} + \dfrac{4\cos(n\pi)}{n\pi}, & \text{if } n \geq 2 \end{cases}$$

The following graph was generated from this:

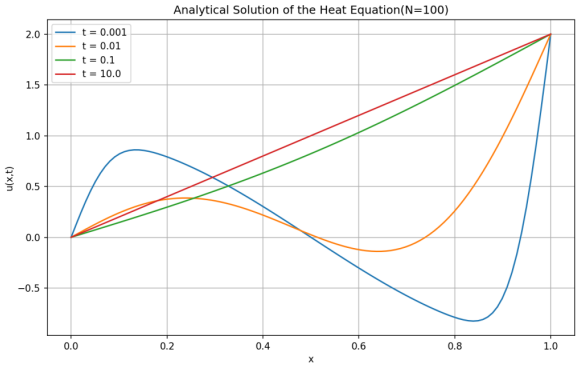


Figure 7: Analytical Solution Graphs to the 1D Heat Equation

The numerical solver for this Equation may be found in Appendix D. This was done in python to generate the below graph, approximating the solution at different time step sizes.

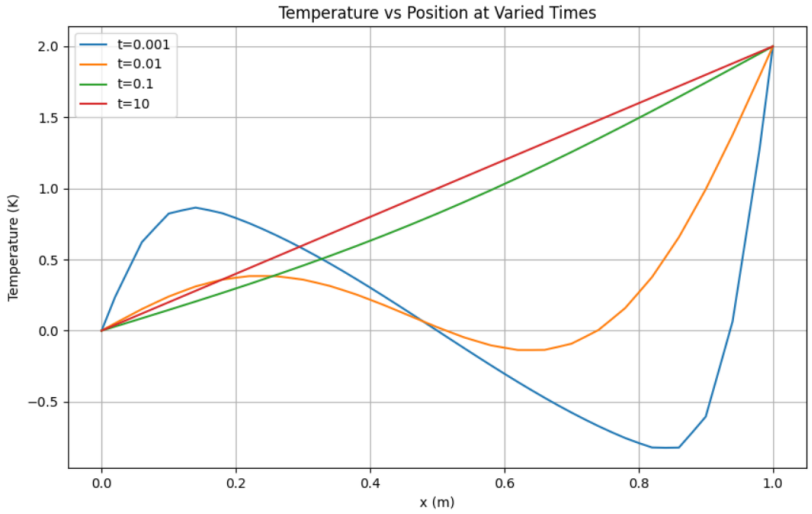


Figure 8: Numerical Solution Graphs to the 1D Heat Equation

Higher accuracy can be achieved using the analytical solution because it provides an exact graph of the function solution. The numerical solver cannot take an upper limit as infinity, so a limit of 100 was used. As this limit is increased the accuracy also increases, though there will always be errors due to the grid resolution. As t grows larger, the linear 2x term becomes dominant, and the function loses the sinusoidal influence.

2.B Effect of Grid Resolution and Series Truncation on Solution Accuracy

For the numerical solution, making grid spacing finer in both time and space increased the accuracy of the final solution. The Forward Time Central Space scheme used in the solver has a first order error in the t term and second order error in the x term. However, since decreasing delta x increases F, and F has to be less than 0.5 for stability in this scheme: a reasonable delta t was chosen to give acceptable iteration count, then delta x was back calculated from fixing F at 0.5

Increasing the number of terms kept in the analytical solution also increases accuracy of the final solution.

References

[1] Partial Differential Equations: An Introduction (Heat Equation), Duke University, CCP. Available: https://sites.math.duke.edu/education/ccp/materials/engin/pdeintro/pde1.html.

[2] D. U. Ahn, S. M. Kim, and H. Shu, "Effect of egg size and strain and age of hens on the solids content of chicken eggs," Poultry Science, vol. 76, no. 6, pp. 914-919, 1997, doi: 10.1093/ps/76.6.914. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0032579119411851.

[3] A.-G. M. El-Shawaf et al, "Chemical, microbial and nutritional evaluation of ostrich eggs compared to hen's egg," in *Proc. [Online]. Available: https://api.semanticscholar.org/CorpusID:251390104,2019.

[4] J. Bagh. et al. Body weight, egg production, and egg quality traits of gray, brown, and white varieties of Japanese quail (Coturnix coturnix japonica) in coastal climatic condition of Odisha. Vet World. 2016 Aug;9(8):832-6. doi: 10.14202/vetworld.2016.832-836. Epub 2016 Aug 10. PMID: 27651670; PMCID: PMC5021831.

[5] C. M. Sabliov et al, "Cooling of shell eggs with cryogenic carbon dioxide: A finite element analysis of heat transfer," LWT - Food Science and Technology, vol. 35, no. 7, pp. 568-574, 2002, doi: 10.1006/fstl.2002.0915. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0023643802909151.

[6] N. Yamamoto, "Equation of Egg Shaped Curve," nyjp07.com, Apr. 29, 2007. https://nyjp07.com/index_egg_E.html.

[7] C. Ad, C. M. Perrins, C. J. O. Harrison, and Reader's Digest Association, Birds - Their Life, Their Ways, Their World. Pleasantville, N.Y.: Reader's Digest Association, 1979, p. 169.

[8] J. Thibault and M. A. G. Maiga, "On finite-difference solutions of the heat equation in spherical coordinates," *Numerical Heat Transfer*, vol. 11, no. 3, pp. 297–318, 1987.

A Code for 1.B.1

Given Yamamoto's half egg equation:

$$h(x,a,b) = \frac{\sqrt{(a-b) - 2x + \sqrt{4bx + (a-b)^2}} \cdot \sqrt{x}}{\sqrt{2}} \tag{11}$$

For the derivative:

$$A = a - b, \quad B = 4b, \quad C = A^2 \tag{12}$$

$$f(x) = \sqrt{A - 2x + \sqrt{Bx + C}} \tag{13}$$

$$g(x) = \frac{\sqrt{x}}{\sqrt{2}} \tag{14}$$

The derivatives of each term:

$$f'(x) = \frac{1}{2\sqrt{A - 2x + \sqrt{Bx + C}}} \left(-2 + \frac{B}{2\sqrt{Bx + C}} \right) \tag{15}$$

$$g'(x) = \frac{1}{2\sqrt{2x}} \tag{16}$$

The product rule may now be applied to solve for the full derivative. The following code solves for the b and x terms that satisfy these conditions:

$$\frac{dy}{dx}(x_a) = 0$$

$$y(x_a) = \frac{1}{2} \text{ width}$$

```
class eggSystem:
def __init__(self, major_length, minor_length, x_guess=2, b_guess=1):
    self.a = major_length  # Major axis length
    self.target_value = minor_length / 2  # Half of the minor axis length
    self.x_guess = x_guess  # Initial guess for x
    self.b_guess = b_guess  # Initial guess for b
    self.x = None  # Solution for x
    self.b = None  # Solution for b
    self.volume = None  # Computed volume

def egg_function(self, x, a, b):
    return np.sqrt((a - b) - 2*x + np.sqrt(4*b*x + (a - b)**2)) * np.sqrt(x) / np.sqrt(2)

def egg_func_derivative(self, b, a, x):
    A = a - b
    B = 4 * b
    C = A ** 2
```

```
        term1 = np.sqrt(A - 2 * x + np.sqrt(B * x + C))
        term2 = np.sqrt(x) / np.sqrt(2)

        # Derivative of term 1
        f_prime = (1 / (2 * np.sqrt(A - 2 * x + np.sqrt(B * x + C)))) * (-2 + (B / (2 * np.sqr

        # Derivative of term 2
        g_prime = 1 / (2 * np.sqrt(2 * x))

        # Product rule
        dy_dx = f_prime * term2 + term1 * g_prime
        return dy_dx

    def egg_system(self, vars, a, target_value):
        x, b = vars
        eq1 = self.egg_function(x, a, b) - target_value  # Function should be target_value
        eq2 = self.egg_func_derivative(b, a, x)  # Derivative should be 0
        return [eq1, eq2]

    def solve_egg_system(self):
        solution = fsolve(self.egg_system, [self.x_guess, self.b_guess], args=(self.a, self.ta
        self.x, self.b = solution
        return self.x, self.b
```

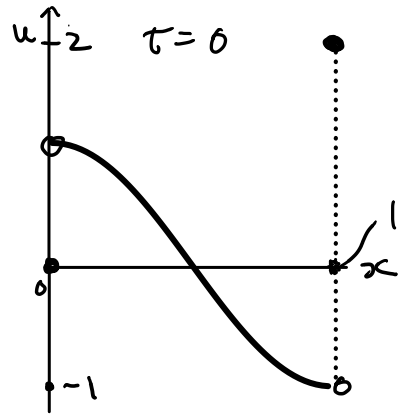
B Code for 1.B.2

```
def spherical_heat_solver(r, t, r_points, dt, alpha, t_water, t_init):
r = r / 100
dr = r / r_points
t_points = int(t / dt)


temp = np.zeros((t_points, r_points))

# Set boundary conditions
temp[:, -1] = t_water  # Python uses -1 for last index
temp[0,:] = t_init

# Calculate Fourier number for stability check
F = alpha * dt / (dr**2)
if F > 0.5:
    raise ValueError("Stability condition violated: F > 0.5")


done_counter = 0
end_index = 0

# Main simulation loop

for k in range(1, t_points): # Temp steps
    temp[k, 0] = temp[k-1, 0] + alpha * dt / (dr**2) * (2 * temp[k-1, 1] - 2* temp [k-1, 0]

    # Interior Points
    for i in range(1, r_points - 1): # Radial positions
        r_i = i*dr # radial position
        temp[k, i] = temp[k-1, i] + alpha * dt * (
            (temp[k-1, i+1] - 2 * temp[k-1, i] + temp[k-1, i-1]) / (dr**2) +
            (2 / (r_i)) * ((temp[k-1, i+1] - temp[k-1, i]) / dr)
        )
    # 80°C Check

    if temp[k,0] >= 80:
        print(f"Cooked at time(s): {k*dt:.1f}")
        return temp[:k+1, :], k*dt

return temp, t
```

C Analytical Solution to 1D Heat Equation

Analytical:

$PDE: u_t = 2u_{xx}, \quad x \in (0,1), t \in (0,\infty)$
$BCs: u(x=0,t) = 0,\ u(x=1,t) = 2, \quad t \in [0,\infty)$
$IC: u(x,t=0) = \cos(\pi x), \quad x \in (0,1)$



let $u = v + \phi$

let $\phi = u(x=0) + x\left(\frac{u(x=1) - u(x=0)}{1}\right)$

$= 2x$

BC:

① $u(x=0) = V(x=0) + \phi(x=0) = 0 + 0 = 0$

② $u(x=1) = V(x=1) + \phi(x=1) = 0 + 2(1) = 2 \Rightarrow V(x=1) = 0$

$$\frac{\partial(v+\phi)}{\partial t} = 2\frac{\partial(v+\phi)}{\partial x^2}$$

$$\frac{\partial v}{\partial t} + \frac{\partial \phi}{\partial t} = 2\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 \phi}{\partial x^2}\right), \quad \frac{\partial \phi}{\partial t} = 0, \quad \frac{\partial^2 \phi}{\partial x^2} = 0$$

$$\frac{\partial v}{\partial t} = 2\frac{\partial^2 v}{\partial x^2} \qquad \text{let } v = XT$$

$$XT' = 2X''T$$

$$\frac{T'}{2T} = \frac{X''}{X} = \lambda = -\mu^2$$

① $X'' - X\lambda = 0$:

$\lambda > 0 \Rightarrow X(x) = Ae^{\lambda t} + B$ From B.C 1 : $X(x) = 0$

$\lambda = 0 \Rightarrow X(x) = At + B$ from BC 1 : $X(x) = 0$

$\lambda < 0 \Rightarrow X(x) = A\sin(\mu x) + B\cos(\mu x)$

Using BC #① $0 = A\sin(0) + B\cos(0) \Rightarrow B = 0$

BC #③ $0 = A\sin(\mu) \Rightarrow \mu = \pi n, \ n \in \mathbb{Z}$

$X(x) = \sin(\pi n x)$

(2) $\frac{T'}{2T} = \lambda = -\mu^2$

$$\int \frac{1}{T}\,dy = \int 2\lambda\,dt$$

$$\ln(T) = 2\lambda t + C_0$$

$$T = C e^{2\lambda t} = C_n e^{-2(\pi n)^2 t}$$

Combining:

$$V = \sum_{n=1}^{\infty} C_n \sin(\pi n x)\, e^{-2(\pi n)^2 t}$$

$$\therefore \quad u = \sum_{n=1}^{\infty} C_n \sin(\pi n x)\, e^{-2(\pi n)^2 t} + 2x$$

Solving C_n:

Using IC: $T(t=0) = \cos(\pi n)$

Fourier Form of:

$$f(x) = \frac{a_0}{2}^{\nearrow 0} + \sum_{n=1}^{\infty} a_n \cos\left(\frac{n\pi}{L}x\right)^{\nearrow 0} + \sum_{n=1}^{\infty} b_n \sin\left(\frac{n\pi}{L}x\right)$$

$$= \sum_{n=1}^{\infty} C_n \sin(\pi n x)$$

$$b_n = \frac{\langle f(x), \phi_n(x)\rangle}{\langle \phi_n(x), \phi_n(x)\rangle} = \frac{\int_0^1 (\cos(\pi x) - 2x)\sin(n\pi x)\,dx}{\int_0^1 \sin^2(n\pi x)\,dx}$$

(3)

$$\int_0^1 (\cos(\pi x) - 2x)\sin(n\pi x)\,dx = \int_0^1 \cos(\pi x)\sin(n\pi x)\,dx \quad (1)$$

$$- \int_0^1 x\sin(n\pi x)\,dx \quad (2)$$

(1) $\displaystyle\int_0^1 \cos(\pi x)\sin(n\pi x)\,dx$

$$= \frac{1}{2}\int_0^1 \sin(\pi(n-1)x) + \sin(\pi(n+1)x)\,dx$$

$$= \frac{1}{2}\left(\frac{-\cos(\pi(n-1)x)}{\pi(n-1)} - \frac{\cos(\pi(n+1)x)}{\pi(n+1)} \right)_0^1$$

$$= \frac{1}{2}\left(\frac{-\cos(\pi(n-1))}{\pi(n-1)} - \frac{\cos(\pi(n+1)x)}{\pi(n+1)} - \left(-\frac{1}{\pi(n-1)} - \frac{1}{\pi(n+1)} \right) \right)$$

$$= \frac{1}{2}\left(\frac{1-\cos(\pi(n-1))}{\pi(n+1)} + \frac{1-\cos((1-n)\pi)}{\pi(n-1)} \right) \quad \leftarrow \cos(1-n) = \cos(1+n) \text{ if } n \in \mathbb{Z}$$

$$= \frac{n(1-\cos(n+1))}{\pi(n^2-1)}$$

(2) $\displaystyle -2\int_0^1 x\sin(\pi n x)\,dx$

let $u = x$ $\quad v = \frac{-\cos(\pi n x)}{\pi n}$

$du = 1$ $\quad dv = \sin(\pi n x)$

$$= -2\left(-\frac{x\cos(\pi n x)}{\pi n}\Big|_0^1 + \int_0^1 \frac{\cos(\pi n x)}{\pi n}\,dx \right)$$

$$= -2\left(-\frac{x\cos(\pi n x)}{\pi n}\Big|_0^1 + \left(\frac{\sin(\pi n x)}{\pi^2 n^2} \right)\Big|_0^1 \right) \quad \sin(\pi n x) \to 0$$

$$= \frac{2\cos(\pi n)}{\pi n} = \frac{2(-1)^n}{\pi n}$$

$$\int_0^1 (\cos(\pi x) - 2x)\sin(n\pi x)\,dx = \frac{n(1-\cos(n+1))}{\pi(n^2-1)} + \frac{2(-1)^n}{\pi n}$$

undefined @ 1

$$\int_0^1 \cos(\pi x)\sin(\pi n x)\,dx \Big|_{n=1}$$

$$= \int_0^1 \cos(\pi x)\sin(\pi x)\,dx$$

$$= \int_0^1 \frac{1}{2}\sin(2\pi x)\,dx = \frac{1}{2}\left[-\frac{\cos(2\pi x)}{2\pi}\right]_0^1 = 0$$

$$\therefore \text{ @ } n=1 \quad = \quad 0 + \frac{2\cos(\pi)}{\pi n} = \frac{-2}{\pi}$$

$$\text{@ } n>2 \quad = \quad \frac{n(1-\cos(n+1))}{\pi(n^2-1)} + \frac{2(-1)^n}{\pi n}$$

(3) $$\int_0^1 \sin^2(n\pi x)\,dx$$

$$= \frac{1}{2}\int_0^1 1 - \cos(2\pi n x)\,dx$$

$$= \frac{1}{2}\left(x - \frac{\sin(2\pi n x)}{2\pi n}\right)\Bigg|_0^1$$

$$= \frac{1}{2}\left(1 - \frac{\sin(2\pi n}{2\pi n} - 0 - \frac{\sin(0)}{2\pi n}\right) = \boxed{\frac{1}{2}}$$

∴ calculating b_n:

$$b_1 = \frac{-4}{\pi}$$

$$b_{n>1} = \frac{2n(1-\cos(n+1))}{\pi(n^2-1)} + \frac{4(-1)^n}{\pi n}$$

$$V(x,t) = \sum_{n=1}^{\infty} C_n \sin(\pi x n)\, e^{-2(n\pi)^2 t}$$

$$V(x,t) = \sum_{n=1}^{\infty} C_n \sin(\pi x n)\, e^{-2(n\pi)^2 t} + 2x$$

$$C_n \in \begin{cases} n=1, & \frac{-4}{\pi} \\ n>1, & \frac{2n(1-\cos(n+1))}{\pi(n^2-1)} + \frac{4(-1)^n}{\pi n} \end{cases}$$

D Numerical Solution for the 1D Heat Equation

The following functions numerically approximate the solution to the 1D heat equation using forward time central spacing.

```
def iterate(space, F, timestep_for_print=1):
for i in range(1, space.shape[0]):
    for j in range(1, space.shape[1] - 1):
        space[i,j] = (1-2*F) * space[i-1,j] + F * space[i-1,j-1] + F * space[i-1,j+1]
        if ((i+ 1,j + 1) == space.shape):
            print(f'{(1-2*F) * space[i-1,j]} + {F * space[i-1,j-1]} + {F * space[i-1,j+1]}
            print(f'{i}, {j}: {space[i,j]}')
            print(space[i-10:i+1, :])
            print("============")

def analytical_solver(times: List[float], order: int = 100, num_points: int = 100):
def C_n(n : int):
    if n == 1:
        return -4 / np.pi
    else:
        return (1 - np.cos((n+1)*np.pi))/ (n+1) / np.pi - (1-np.cos((1-n)*np.pi)) / (1-n)/

def f_x(x : np.array, t : np.array, order : int) -> float:
    if order < 1:
        raise Exception("Order must be at least 2")
    # Shape: (len(t), len(x))
    values =  2 * np.tile(x, (t.shape[0], 1))

    for n in range(1, order + 1):
        coeff = C_n(n)
        sin_term = np.sin(np.pi * n * x)  # shape: (len(x),)
        decay = np.exp(-2 * (np.pi * n)**2 * t[:, None])  # shape: (len(t), 1)
        values += coeff * sin_term[None, :] * decay  # broadcasted multiplication
    return values  # shape: (len(t), len(x))

times = np.array(times)
x_points = np.linspace(0, 1, num_points)

values = f_x(x_points, times, order)

plt.figure(figsize=(10, 6))

for i, t in enumerate(times):
    plt.plot(x_points, values[i, :], label=f"t = {t}")

plt.xlabel("x")
plt.ylabel("u(x,t)")
```

```
plt.title(f"Analytical Solution of the Heat Equation(N={order})")
plt.legend(loc="upper left")
plt.grid(True)
plt.savefig("plots/2a_analytical.png", dpi=300)
plt.show()
```

E Ownership:

1. Mark Do: 2
2. Mckale Chung: 1 (Knower of all egg geometry)
3. Carmelli Dao: 1
4. Miekale Smith: 2
5. Odette Beaudin: 1