FINITE DIFFERENCE METHODS FOR THE RADIALLY SYMMETRIC HEAT EQUATION APPLIED TO THE THERMODYNAMICS OF EGGS

A thesis

submitted by

Lea Levi

in partial fulfillment of the requirements

for the degree of

Master of Science

in

Mathematics

TUFTS UNIVERSITY

May 2025

Adviser: Christoph Börgers

Abstract

We approximate a generic large chicken egg by a sphere of the same volume and solve the radially symmetric heat equation using finite difference methods. We introduce four increasingly refined models that use measurements of external water temperature to approximate the internal temperature and physical consistency of the egg. We implement each method using different cooking techniques and compare the outputs to real boiled eggs.

Özet

Amerikan standartlarında “large” boyutundaki bir tavuk yumurtası, eşdeğer hacme sahip bir küre şeklinde modellenmiştir. Bu modelde, radyal simetrik ısı iletim denklemi sonlu farklar yöntemiyle çözülmüştür. Yumurtanın çevresindeki suyun sıcaklığına ilişkin veriler kullanılarak, yumurtanın iç sıcaklığı ve pişme durumu tahmin edilmiştir. Pişme seviyesini giderek daha hassas biçimde öngören dört farklı model geliştirilmiştir. Her bir model farklı pişirme yöntemlerine uygulanmış ve elde edilen sonuçlar, gerçek haşlanmış yumurtalarla karşılaştırılarak değerlendirilmiştir.

To my brothers Roni and Leon, and my cats Ela and Limon.

Acknowledgements

My heartfelt gratitude goes to my advisor, Christoph Börgers, for the patience with which he teaches. Christoph was my first math professor in college; he ignited my passion for applied mathematics way back in 2020. He retires from Tufts University this semester, while I graduate with his fingerprints all over my creative process. In this full circle moment, I hope retirement gives him plenty of time to listen to Schönberg with his radioactive cat.

I would also like to acknowledge my housemate, Anat Katz, for inspiring the idea behind this project. Thanks to the way she cooks her noodles, I was introduced to an application of the heat equation that excited me enough to write a thesis.

Of course, profound thanks to all my past and present coaches and teammates, for being my sturdiest support system for over a decade. Your pride in me has made this journey more meaningful.

Lastly, this would not have been possible without my parents, Beyza and Daniel Levi. Their unconditional sacrifices are what made this education possible for me. Their blind trust in me is the reason I can approach anything with blind confidence. I hope they never cease to set the bar too high, so I never cease to overachieve.

Contents

List of Tables

List of Figures

Finite Difference Methods for the Radially Symmetric Heat Equation Applied to the Thermodynamics of Eggs

Chapter 1

Introduction

1.1 Motivation and Setup

The objective of this project is to model the internal consistency of a shelled egg. According to the American Egg Board [1], chicken egg white begins coagulating at $62.5°C$, and solidifies at $65°C$. Additionally, the yolk begins coagulating at $65°C$, is jammy (coagulated but not solid) at $68°C$, and solidifies completely at $70°C$. These thresholds are used to describe the following temperature profiles:

Soft Boil: coagulated egg white and runny (liquid) yolk. Then, egg white must surpass $62.5°C$, and the yolk must be heated to no more than $65°C$.

Medium Boil: completely solid egg white and jammy yolk. Then, egg white must surpass $65°C$, and the yolk must be heated to at least $68°C$, but to no more than $70°C$.

Hard Boil: completely solidified egg white and yolk. Then, egg white must surpass $65°C$, and the yolk must surpass $70°C$.

In the rest of this paper we construct and refine a model for the internal temperature of an egg and use the above definitions to classify its internal consistency.

1.2 Estimation By Sphere

The egg is estimated by two concentric spheres in order to simplify the geometry of the problem. This construction can be observed in Figure 1.1. According to the American Egg Board's online cooking school [2], five large chicken eggs contain one cup of liquid egg product, and thus an average large chicken egg contains approximately 48 ml of liquid. Then, we can solve for $\frac{4}{3}\pi r_{egg}^3 = 48\text{cm}^3$ to construct a sphere that has the same volume as an average large egg, which corresponds to

a radius of 2.255cm. Additionally, an average large egg contains $\frac{1}{16}$ cups of yolk, which translates to $\frac{4}{3}\pi r_{yolk}^3 = 15$ ml, corresponding to 1.53cm.

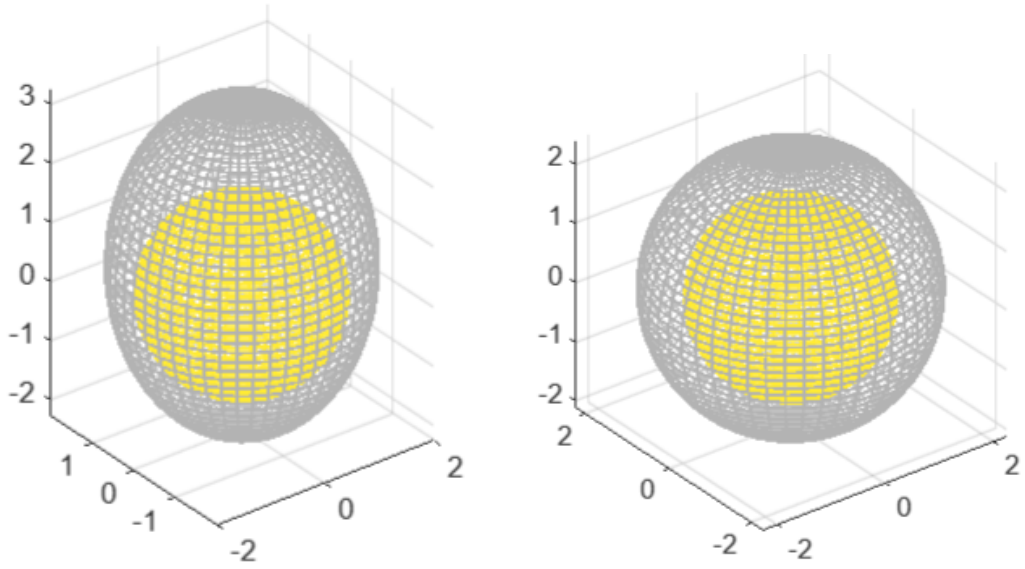

Figure 1.1: Estimation of Egg by Sphere of Same Volume, Normalized

1.3 The Heat Equation and Coefficients of Thermal Diffusivity

Let $E \subset \mathbb{R}^3$ be some fixed region, and let ∂E be its boundary. Let u denote the temperature across the domain, and ∇u denote its gradient. Then, near a point $x \in \partial E$, there is flow of thermal energy (denoted $\mathbf{q}$) across ∂E, that is proportional to and in the direction of $-\nabla u$. The negative sign follows from Fourier's Law, which states that thermal energy flows in from high concentration to low. The constant of proportionality is called thermal conductivity and denoted with κ.

$$\mathbf{q} = -\kappa \nabla u$$

The total amount of energy in the region E is [8]:

$$\int_E e(x,t)\, dx$$

where $e(x)$ is the density of thermal energy, or energy per unit volume $\frac{J}{m^3}$. Using the divergence theorem, its rate of change across the entire region is:

$$\int_E \frac{\partial e}{\partial t}\, dV = -\int_{\partial E} \mathbf{q}\cdot n\, dA = \int_{\partial E} \big(\kappa \nabla u\big)\cdot n\, dA = \int_E \nabla\cdot\big(\kappa\nabla u\big)\, dV$$

Since this result holds for any regular region $E \subset \mathbb{R}^3$, the integrands must be equal pointwise. Then,

$$\frac{\partial e}{\partial t} = \nabla\cdot\big(\kappa\nabla u\big)$$

To turn this into an equation for u, we relate the thermal energy to the temperature. The specific heat, denoted c, is a measure of energy required to heat up 1 kg of material by $1°C$, and has the unit $\frac{J}{kg\cdot^\circ C}$. The density, denoted ρ, has unit $\frac{kg}{m^3}$. So, the volumetric heat capacity, described by $c\rho$ has the unit $\frac{J}{kg\cdot^\circ C}\cdot\frac{kg}{m^3} = \frac{J}{m^3\cdot^\circ C}$. Additionally, u has the unit $°C$, and thus $uc\rho$ has the unit $\frac{J}{m^3}$, which is also the unit for the thermal energy density. Then, we can describe the density of energy per unit volume by $e = c\rho u$. Therefore,

$$\frac{\partial}{\partial t}\big(c\rho u\big) = \nabla\cdot\big(\kappa\nabla u\big)$$

$$\implies \frac{\partial u}{\partial t} = \frac{1}{c\rho}\nabla\cdot\big(\kappa\nabla u\big)$$

Additionally, if κ is constant across the material,

$$\frac{\partial u}{\partial t} = \frac{\kappa}{c\rho}\nabla^2 u$$

We now introduce α, the thermal diffusivity coefficient defined by $\alpha = \frac{\kappa}{\rho c}$. Intuitively, $\frac{k}{c\rho}$ compares a material's ability to transfer heat to its ability to retain it, so α describes the rate of transfer of heat.

$$\frac{\partial u}{\partial t} = \alpha\nabla^2 u \tag{1.1}$$

We consult the May 2006 edition of Journal of Food Engineering [3] for the density, conductivity, and specific heat of liquid egg products, which are found in Table 1.1.

	Egg White	Egg Yolk
κ	$0.55\ \frac{J}{sec \cdot m \cdot^\circ C}$	$0.39\ \frac{J}{sec \cdot m \cdot^\circ C}$
ρ	$1026.6\ \frac{kg}{m^3}$	$1134.3\ \frac{kg}{m^3}$
c	$3.552\ \frac{J}{kg \cdot^\circ C}$	$2.62\ \frac{J}{kg \cdot^\circ C}$

Table 1.1: Density, Conductivity, and Specific Heat of Liquid Egg Products

Remark 1.3.1 Using this definition and the units from the table above, we calculate the unit of thermal diffusivity as:

$$\alpha = \frac{\kappa}{\rho c} = \frac{J}{sec \cdot m \cdot K} \cdot \frac{m^3}{kg} \cdot \frac{K \cdot kg}{J} = \frac{m^2}{sec}$$

We compute α_{white} and α_{yolk} as follows:

$$\alpha_{yolk} = \frac{\kappa_{yolk}}{\rho_{yolk} \cdot c_{yolk}} = \frac{0.39 \frac{J}{sec \cdot m \cdot^\circ C}}{1134.3 \frac{kg}{m^3} \cdot 2.62 \frac{J}{kg \cdot^\circ C}} = 1.31 \times 10^{-7} \frac{m^2}{sec}$$

$$\alpha_{white} = \frac{\kappa_{white}}{\rho_{white} \cdot c_{white}} = \frac{0.55 \frac{J}{sec \cdot m \cdot^\circ C}}{1026.6 \frac{kg}{m^3} \cdot 3.552 \frac{J}{kg \cdot^\circ C}} = 1.51 \times 10^{-7} \frac{m^2}{sec}$$

Additionally, some of the simpler models we introduce in Chapter 2 assume the egg is made of homogeneous liquid, which requires an additional α_{egg} that we calculate

using the following weighted average by mass:

$$\alpha_{egg} = 0.33 \cdot \alpha_{yolk} + 0.67 \cdot \alpha_{white} = 0.33 \cdot 1.31 \times 10^{-7} \frac{m^2}{sec} + 0.67 \cdot 1.51 \times 10^{-7} \frac{m^2}{sec} = 1.44 \times 10^{-7} \frac{m^2}{sec}$$

We normalize the radius to simplify the domain. This way, our spatial points will correspond to the percent distance from the center, which makes the later results significantly more interpretable. Recall that our total radius is $r_{egg} = 2.255\text{cm}$, so we must convert our unit from $\frac{m^2}{sec}$ to $\frac{r_{egg}^2}{sec}$.

$$\frac{m^2}{r_{egg}^2} = \frac{(100\text{cm})^2}{(2.255\text{cm})^2} = 1966.56$$

The above equality implies $1\frac{m^2}{sec} = 1966.56\frac{r_{egg}^2}{sec}$, and thus we must multiply each of our α values by this number, so that the unit of our thermal diffusivity coefficient is appropriate for a normalized radius. In the rest of the paper, we will be using the following α values:

$$\alpha_{yolk} = 1.31 \times 10^{-7} \frac{m^2}{sec} = 2.576 \times 10^{-4} \frac{r_{egg}^2}{sec}$$

$$\alpha_{white} = 1.51 \times 10^{-7} \frac{m^2}{sec} = 2.970 \times 10^{-4} \frac{r_{egg}^2}{sec}$$

$$\alpha_{egg} = 1.44 \times 10^{-7} \frac{m^2}{sec} = 2.832 \times 10^{-4} \frac{r_{egg}^2}{sec}$$

Although we have established in the previous section that the egg white and yolk have different α values, at this initial stage we treat the egg as if it contains homogeneous liquid that is 33% yolk and 67% egg white by mass. Hence, this section gives the form of the heat equation for constant α only. The two materials' thermal diffusivity coefficients will be distinguished in Chapter 3.

As we have derived in the previous section, for constant coefficients, the general form of the heat equation is:

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u$$

where $\nabla^2 u$ is the Laplacian of u, which represents the coordinate-wise spatial derivative of the temperature. This can be interpreted as the distribution of heat within the boundary. The general form of the Laplacian is as follows:

$$\alpha\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

1.4 Radially Symmetric Heat Equation

Let's start with the Laplacian in the Euclidean space:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial x^2} \tag{1.2}$$

Remark 1.4.1 We may also denote the partial derivative using u_r. For the rest of the calculations, we use u_r and $\dfrac{\partial u}{\partial r}$ interchangably. ⋄

The Laplacian can be transformed into spherical coordinates using the system

$$\begin{aligned} x &= r\ \sin\ (\theta)\ \cos\ (\phi) \\ y &= r\ \sin\ (\theta)\ \sin\ (\phi) \\ z &= r\ \cos\ (\theta) \end{aligned} \tag{1.3}$$

We will not be walking through the steps of this transformation in this paper, as the process is lengthy and not relevant enough to the problem. A detailed derivation is included in the bibliography [5]. After applying the transformation, the Laplacian in spherical coordinates (r, θ, ϕ) is given by:

$$\nabla^2 u = \frac{1}{r^2}\left(r^2 u_r\right)_r + \frac{1}{r^2 \sin\theta}\left(\sin\theta u_\theta\right)_\theta + \frac{1}{r^2 \sin^2\theta} u_{\phi\phi} \tag{1.4}$$

Because of radial symmetry, which will be explained more carefully in Section 2.2, u depends only on the radial distance r and time t, and not on the angular

coordinates θ and ϕ. Therefore, the terms involving derivatives with respect to θ and ϕ vanish, leaving behind:

$$\nabla^2 u = \frac{1}{r^2}\left(r^2 u_r\right)_r \tag{1.5}$$

Substituting the radially symmetric Laplacian into the original heat equation gives us the following:

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

We expand the radial derivatives in the following way:

$$\frac{\partial}{\partial r}\left(r^2 \frac{\partial u}{\partial r}\right) = 2r\frac{\partial u}{\partial r} + r^2 \frac{\partial^2 u}{\partial r^2}$$

Substituting this back gives us the heat on a homogeneous sphere:

$$\frac{\partial u}{\partial t} = \alpha\left(\frac{2}{r}\frac{\partial u}{\partial r} + \frac{\partial^2 u}{\partial r^2}\right) = \alpha\left(\frac{2}{r}u_r + u_{rr}\right) \tag{1.6}$$

Chapter 2

Problem Formulation

In order to use our definitions from 1.1, we need the temperature across the egg's interior. However, the only measurements we have access to are 1) the initial condition of the egg, and 2) the temperature of the water in which we place it. In this chapter, we use these two measurements to construct an initial value problem, which we solve numerically in chapters 3 and 4.

2.1 Motivation for Numerical Methods

We are often interested in solving complex equations that describe how physical phenomena evolve over time and space. Almost all of these equations, like the heat equation we are interested in, are often impossible to solve exactly using traditional mathematical methods, especially when the problem is defined over irregular shapes or has complicated boundary conditions.

This is where numerical schemes come into play. Numerical schemes are a set of computational tools designed to approximate the solutions of equations that cannot be solved exactly. Instead of producing an exact formula for the solution, these schemes break down the problem into small, manageable pieces, allowing us to compute an approximate solution step by step. By discretizing the problem (i.e., dividing continuous time and space into smaller segments), numerical schemes allow us to simulate the behavior of these equations to any desired level of accuracy. Simply put, numerical schemes transform continuous equations into a sequence of discrete approximations.

2.1.1 Finite Difference Methods

Finite difference methods are a class of numerical techniques used to approximate solutions to partial differential equations. In essence, we:

- Replace derivatives with difference quotients
- Approximate the solution on a discrete grid of points
- Iterate in time to simulate evolution

The accuracy of a difference scheme depends on the size of the time and space intervals. Smaller intervals generally produce more accurate solutions, but require more computational resources. Additionally, schemes can become unstable if the intervals are not chosen carefully, leading to wildly incorrect solutions. In the next chapter, we will explore several numerical schemes, focusing on their mechanics, accuracy, and stability. We will also implement and provide visuals for each method, and explain their implications on the egg cooking problem.

2.2 Application to the Problem

Temperature should be modeled as a function of time (t) and space (x, y, z). Recall from Section 1.2 that the shape is radially symmetric, and thus any two points equidistant from the center of the object will be at the same temperature. Then, we can describe the temperature at a point by t and r only, which is a crucial step, as it reduces an otherwise 4-dimensional system (x, y, z, t) to a 2-dimensional system (t and r).

Let us define a sequence $\{r_j\}_{j=0}^{n} = j\Delta r$ such that $j = 0, 1, 2, ..., \frac{1}{\Delta r}$. This sequence represents the set of discrete r-values at which we compute the value of u. For this example, our spatial step size Δr is 0.02. Or, equivalently, we have cut the radius into 50 even pieces.

Similarly, let $\{t^n\}_{n=0}^{\infty} = n\Delta t$ such that $n = 0, 1, 2, ...$ represents the step count for the time discretization. The following example uses $\Delta t = 0.25$. Since the diffusion coefficient describes behavior of heat per second, $\Delta t = 0.25$ divides a second into 4 equal pieces, and thus one minute of simulation requires 240 computations.

Remark 2.2.1 We introduce the notation $u(j\Delta r, n\Delta t) = u(r_j, t^n) = u_j^n$. ♢

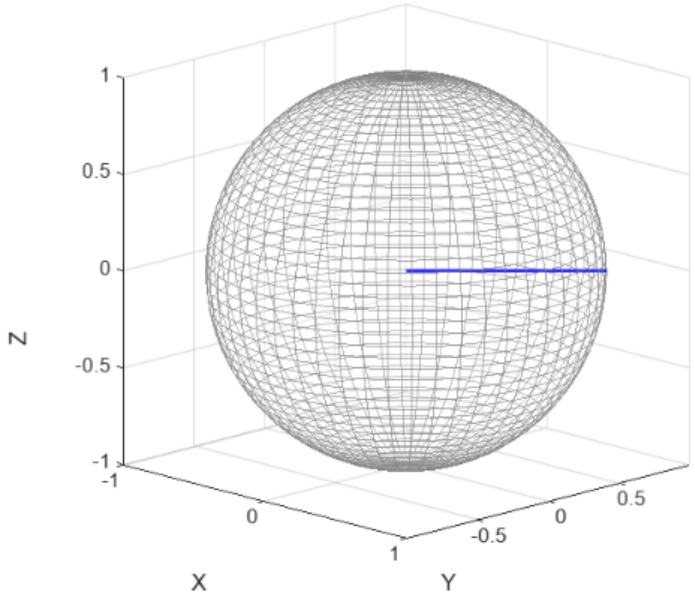

Figure 2.1: 3D Visual of the Domain of r

An additional advantage of the "radius axis" is that we are able to observe the internal temperature of the entire egg in the form of a single line. This is because the blue line in Figure 2.1 shares its temperature profile with every other line that connects the center to the boundary. All the figures we provide in this paper visualize the temperature along this blue line segment.

At $t = 0$, or t^0, we assume the egg to be uniformly at room temperature, except at the egg shell boundary. At this boundary, which corresponds to $r_{50} = 1$, we measure the water temperature as a function of time and call it $f(t)$. Since the solution at $r_{50} = 1$ is given by the water temperature $f(t)$, here we assign a Dirichlet boundary condition. For now, we assume that the water temperature is constant at $100°C$.

Radial symmetry, together with differentiability of u as a function of x, y, z, and t implies that $\frac{\partial u}{\partial r}(0, t) = 0$ for all t. Then, we assign a condition at the center of the egg r_0 that specifies the lack of heat flux through the origin. Geometrically, this means that thermal energy travels towards or away from the center, but never through it. This corresponds to a Neumann boundary condition at $r = 0$ such that $\frac{\partial}{\partial r} u(r = 0, t) = 0$.

Now, we can use the radially symmetric heat equation from 1.5, which states that

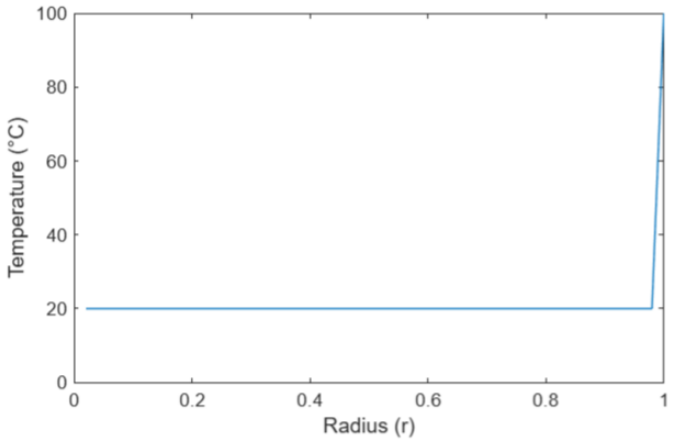

Figure 2.2: Initial Temperature Profile

the next time step can be calculated using an operation on the current temperature profile. So, given that our shape is a sphere with the aforementioned boundary conditions, and that the temperature profile at 0 minutes looks like Figure 2.2, we can construct an initial value problem and numerically compute the temperature profile at any time we want.

Figure 2.3 shows the computed temperature profile 2 minutes after the egg has been placed in boiling water. This profile was calculated using the first numerical method we describe in the following chapter. The pink line is located at $68°C$, which is the threshold at which the entire egg is coagulated. Then, this figure implies it takes 2 minutes to cook the outer 20% of the egg, and that after 2 minutes, the inner 20% has yet to change temperature at all.

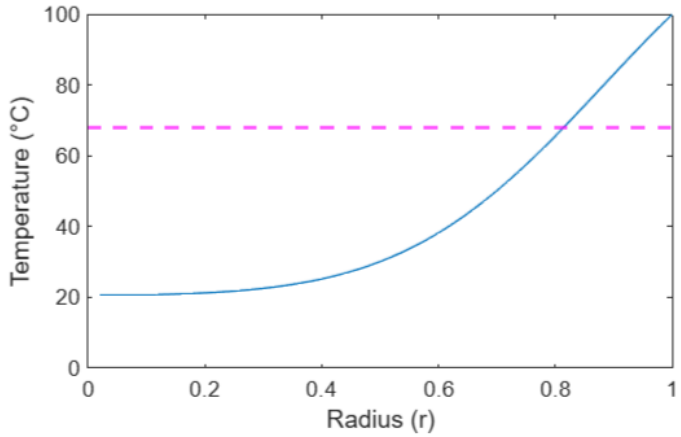

Figure 2.3: Internal Temperature Profile After 2 Minutes

We can use the same method to approximate the temperature profile at 4 minutes, 6 minutes, 8 minutes, and 10 minutes. Observe that the graph for 10 minutes suggests a completely coagulated egg, as every part of the radius has surpassed the pink line.

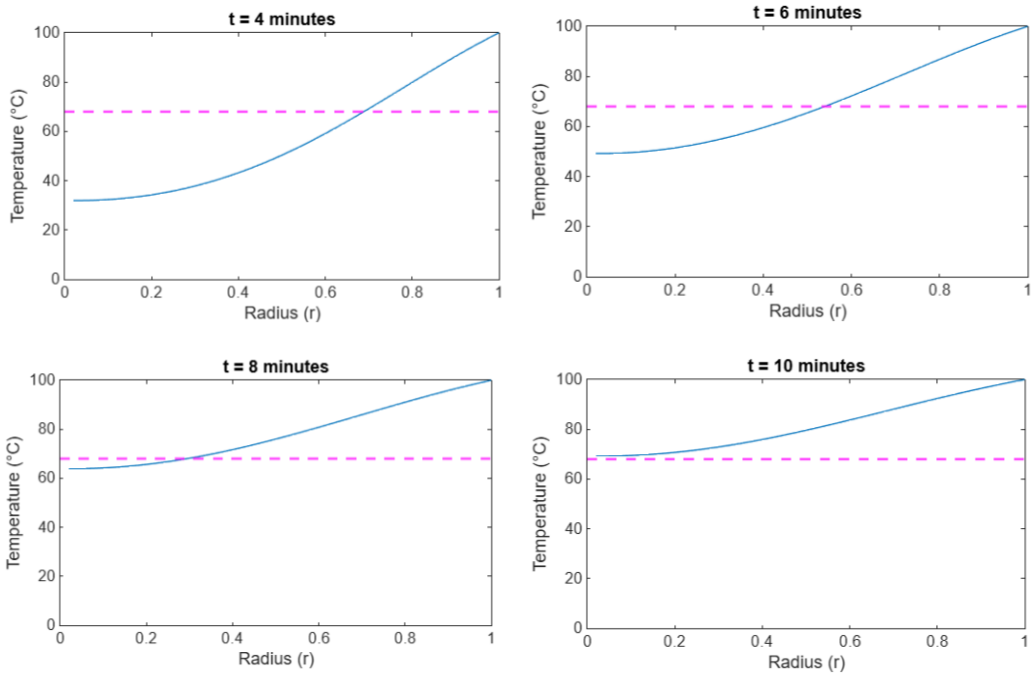

Figure 2.4: Additional Temperature Profiles

In order to visualize the internal temperature across time, we incorporate a second new into the figure. One may visualize this step as “stacking" the lines from Figure 2.4 according to their t-coordinates, as pictured in the upper left image from Figure 2.5.

Note that as we decrease the amount of time between calculations, which corresponds to deceasing the size of Δt, the lines begin to create a denser mesh blanket. Once they are dense enough, this blanket represents almost every coordinate in the sphere, at almost every point in time. In other words, the output $u(r_j,\ t^n)$ will approximate the temperature at time t^n at location r_j.

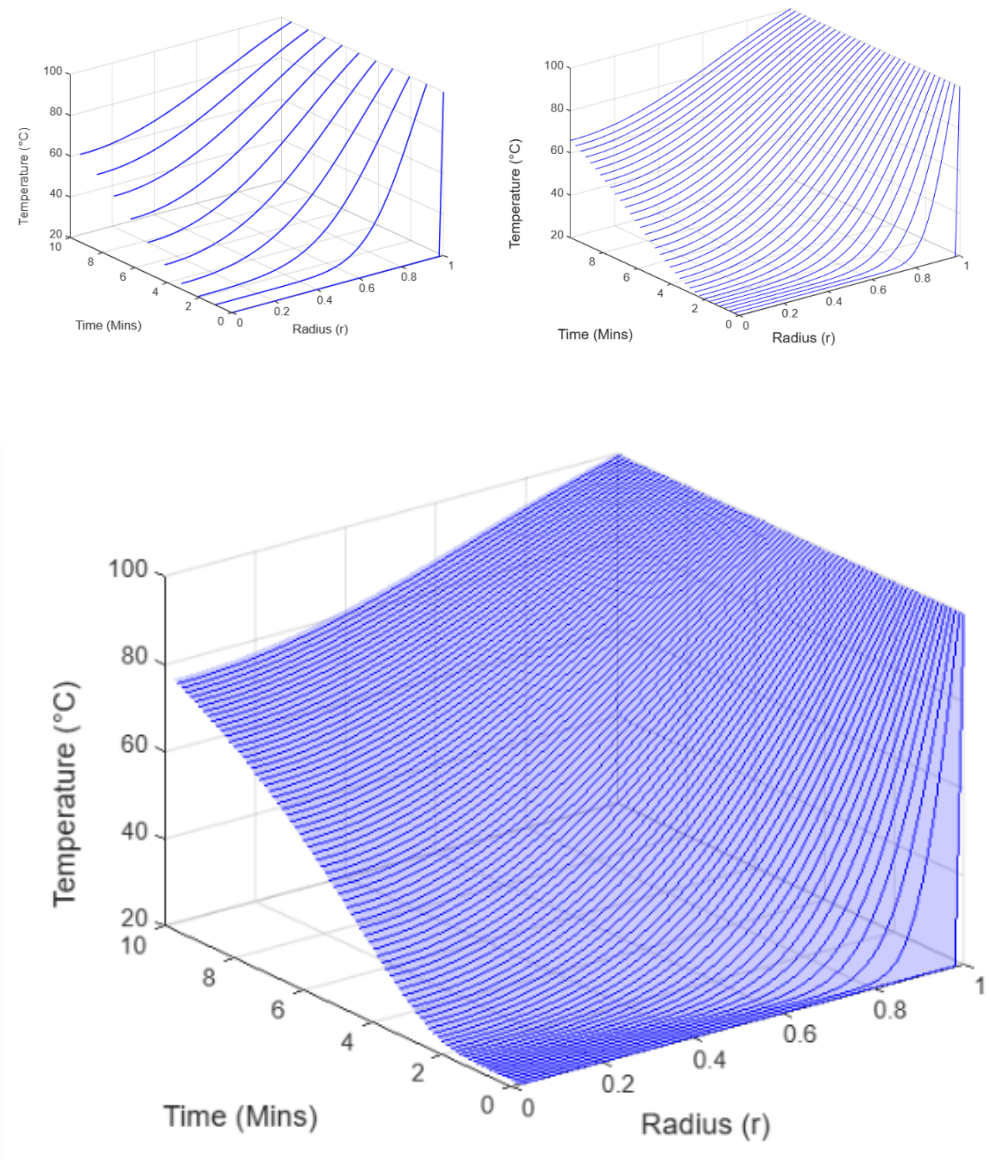

Figure 2.5: 3D Visualization of Temperature Surface

Chapter 3

Numerically Solving the Heat Equation on a Homogeneous Sphere

3.1 Method 1

Euler's Method is the simplest way to numerically solve differential equations in cases where traditional methods prove insufficient. Simply put, it uses tangent line approximations to solve initial value problems. For example, given a problem of the form:

$$\frac{\partial u}{\partial t} = f(t,\ u),\ u(t_0) = u_0$$

Euler's Method estimates the solution by discretizing the time domain into small increments of size Δt. Starting from the initial condition $u_0 = u(t_0)$, the next value u_1 is computed using the slope of the solution $f(t_0,\ u_0)$ at the initial point:

$$u_1 = u_0 + \Delta t \cdot f(t_0, u_0)$$

This process is repeated to generate the approximate trajectory of the solution:

$$u_{n+1} = u_n + \Delta t \cdot f(t_n,\ u_n)$$

Euler's Method essentially "walks" along the curve of the solution by following the direction of the slope at each step. The accuracy of the method depends very heavily on the step size Δt; smaller step sizes yield more accurate approximations but require significantly more computational effort. Additionally, although Euler's method is quite simple and inefficient in comparison to other methods we will expand on, by assigning Δt small enough we can still get arbitrarily close to a real solution.

Let us begin deriving a numerical scheme that approximates a first order consistent solution to the heat equation. Recall that the radially symmetric heat equation

for homogeneous material is:

$$\frac{\partial u}{\partial t} = \alpha\left(\frac{2}{r}\frac{\partial u}{\partial r} + \frac{\partial^2 u}{\partial r^2}\right) \tag{3.1}$$

Using the forward Euler method, we approximate the time derivative by:

$$\frac{\partial u}{\partial t} \approx \frac{u_j^{n+1} - u_j^n}{\Delta t} \tag{3.2}$$

Similarly, we use upwind differencing to approximate the spatial derivative by:

$$\frac{2}{r}\frac{\partial u}{\partial r}\Big|_{r=r_j} \approx \frac{2}{r_j} \cdot \frac{u_{j+1}^n - u_j^n}{\Delta r} \tag{3.3}$$

This choice is justified both numerically and physically: Near the origin, the coefficient $\frac{2}{r_j}$ becomes large, and using a backward difference (which would rely on values closer to the singularity at $r = 0$) can introduce instability. Additionally, our question focuses mainly on heat traveling towards the center, so using upwind differencing approximates the partial derivative with respect to a grid point that is higher in temperature. This aligns with the physical geometry of the problem because of Fourier's Law.

Finally, we use central differencing to approximate the second r derivative:

$$\frac{\partial^2 u}{\partial r^2}\Big|_{r=r_j} \approx \frac{\partial}{\partial r}\left(\frac{u_j^n - u_{j-1}^n}{\Delta r}\right) = \frac{(u_{j+1}^n - u_j^n) - (u_j^n - u_{j-1}^n)}{\Delta r^2} = \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta r^2} \tag{3.4}$$

Central differencing was chosen based on its second order nature, which will be analyzed further in section 3.1.2.

Substituting (3.2), (3.3), and (3.4) into (3.1), we have:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \alpha\left(\frac{2}{r_j} \cdot \frac{u_{j+1}^n - u_j^n}{\Delta r} + \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta r^2}\right)$$

Rearranging for u_j^{n+1} gives us the final numerical scheme:

$$u_j^{n+1} = u_j^n + \alpha \cdot \Delta t\left(\frac{2}{r_j} \cdot \frac{u_{j+1}^n - u_j^n}{\Delta r} + \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta r^2}\right) \tag{3.5}$$

This scheme solves for the temperature at the interior of the egg; recall that we assign Dirichlet and Neumann boundary conditions at $r = 1$ and $r = 0$, respectively. Since $r = 1$ corresponds to the egg shell, the temperature at this point is prescribed by the water temperature function $f(t)$, which corresponds to a Dirichlet boundary condition expressed as $u(r = 1, t) = f(t)$. Additionally, since thermal energy should only flow either towards or away from the center, there should be no flux through the origin. So, the boundary condition at $r = 0$ assumes $\frac{\partial u}{\partial r}(r = 0, t) = 0$.

A MATLAB implementation of this scheme can be found in the Appendix. In this implementation, the Neumann boundary condition at the center of the sphere is handled by enforcing symmetry: the temperature at the center is set equal to the temperature at the first interior point. This is implemented in the code by setting $u(0, t) = u(\Delta r, t)$, which approximates zero radial heat flux at $r = 0$ without encountering the singularity in the PDE at the origin. At the outer boundary $r = 1$, the water temperature function directly assigns the value of $u(r = 1, t)$.

3.1.1 von Neumann Stabiliy Analysis

For explicit timestepping schemes solving the heat equation, stability typically requires $\frac{\alpha \Delta t}{\Delta r^2} \leq C$, where C is some constant that depends on our scheme. This constant is called the Courant-Friedrichs-Lewy (CFL) condition for stability [6] .

We will derive the condition using von Neumann analysis, which is based on decomposing the numerical solution into Fourier modes and examining how these modes evolve over time. The Fourier wavenumber is commonly denoted k. The key idea behind the analysis is to substitute a trial solution of the form $u_j^n = G^n e^{ikj\Delta r}$ into the difference equation and derive an amplification factor G. This amplification factor corresponds to the stability of the discrete equations, and explains the behavior of a small perturbation in the initial condition. Then, depending on G, we determine whether this perturbation grows or decays between each subsequent computational step. Since we do not want numerical errors to grow uncontrollably, we solve for a constraint on the time step Δt and spatial discretization Δr that ensures $|G| \leq 1$ for all r_j and t^n. Such a G will render our scheme von Neumann stable.

Note that this is a mode-wise analysis, so it only checks how a single Fourier mode behaves. But actual numerical solutions are combinations of modes, and interactions between modes can cause instability even when each individual mode is stable. Additionally, the Fourier expansion assumes periodic boundary conditions, even though our model has Dirichlet and Neumann boundaries. This means that a method can be von Neumann stable and still blow up near the boundary. Finally, the analysis only relates to PDE's with constant coefficients, so it will lose applicability once we turn to our non-homogeneuos model in Chapter 4.

However, the analysis is fast and simple, and it is useful because it gives explicit stability conditions. It is also widely applicable for standard problems like the heat equation on a unit sphere. Moreover, each Fourier mode corresponds to a "wavelength" of information, and stability of all these modes means no particular spatial

pattern in the solution grows over time. This notion is especially useful in diffusion problems where mode behavior reflects physical propagation, so von Neumann stability ensures that a small perturbation in the initial condition will eventually smooth out. Hence, although von Neumann analysis is not rigorous, it does check for a necessary condition.

Let us fix some small Δt and Δr , and substitute $u_j^n = G^n e^{ikj\Delta r}$ into the numerical scheme from (3.5):

$$G^{n+1}e^{ikj\Delta r} = G^n e^{ikj\Delta r} + \alpha\Delta t\left(\frac{G^n e^{ikj\Delta r}\cdot\left(e^{ik\Delta r} - 2 + e^{-ik\Delta r}\right)}{\Delta r^2} + \frac{2}{r_j}\frac{G^n e^{ikj\Delta r}\cdot\left(e^{ik\Delta r} - 1\right)}{\Delta r}\right)$$

Dividing both sides by $G^n e^{ikj\Delta r}$:

$$G = 1 + \alpha\cdot\Delta t\left(\frac{e^{ik\Delta r} - 2 + e^{-ik\Delta r}}{\Delta r^2} + \frac{2}{r_j}\frac{e^{ik\Delta r} - 1}{\Delta r}\right)$$

Using identities $e^{ik\Delta r} + e^{-ik\Delta r} = 2\cos(k\Delta r)$ and $e^{ik\Delta r} = \cos(k\Delta r) + i\sin(k\Delta r)$, we rewrite the previous expression as:

$$G = 1 + \alpha\cdot\Delta t\left(\frac{2\cos(k\Delta r) - 2}{\Delta r^2} + \frac{2}{r_j}\frac{\cos(k\Delta r) + i\sin(k\Delta r) - 1}{\Delta r}\right)$$

Dividing G into its real and imaginary parts, we have:

$$G = 1 + \alpha\Delta t\left(\frac{2\cos(k\Delta r) - 2}{\Delta r^2} + \frac{2\cos(k\Delta r) - 2}{r_j\Delta r}\right) + i\alpha\Delta t\frac{2\sin(k\Delta r)}{r_j\Delta r}$$

Since G represents the amplification of a perturbation, we are looking for $|G| \le 1$, or equivalently $|G|^2 \le 1$. We prefer to use $|G|^2$ in this case because it is real valued and differentiable, which will become useful when we search for extrema. Note that the local extrema of $|G|$ and $|G|^2$ occur at the same points.

$$|G|^2 = \left(1 + \alpha\Delta t\left(\frac{2\cos(k\Delta r) - 2}{\Delta r^2} + \frac{2\cos(k\Delta r) - 2}{r_j\Delta r}\right)\right)^2 + \left(\alpha\Delta t\frac{2\sin(k\Delta r)}{r_j\Delta r}\right)^2 \le 1$$

Notice that by construction, $r_j = j\Delta r$ such that $j = 1, 2, ..., \frac{1}{\Delta r}$. We do not include $j = 0$ because we assume no heat flux at the center point, so no calculation

is done at r_0. Then, let us fix j and rewrite the equation such that $r_j = j\Delta r$. Note that we can fix j because we are assuming the entire solution is a pure Fourier mode. Thus, what happens at one spatial index reflects what happens everywhere, because the mode behaves identically across space up to a phase shift. So, by fixing j we are observing the time evolution of that single Fourier mode, which is independent of the spatial location because the PDE and scheme are translation-invariant in space. Therefore, fixing j just simplifies the algebra without restricting generality.

$$|G|^2 = \left(1 + \frac{\alpha\Delta t}{\Delta r^2}\left(2\cos(k\Delta r) - 2 + \frac{2\cos(k\Delta r) - 2}{j}\right)\right)^2 + \left(\frac{\alpha\Delta t}{\Delta r^2}\frac{2\sin(k\Delta r)}{j}\right)^2$$

Let $\frac{\alpha\Delta t}{\Delta r^2} = \gamma$ and $k\Delta r = \theta$:

$$|G|^2 = \left(1 + 2\gamma(\cos(\theta) - 1) + \frac{2\gamma}{j}(\cos(\theta) - 1)\right)^2 + \left(\frac{2\gamma}{j}\sin(\theta)\right)^2$$

$$= 1+4\gamma(\cos(\theta)-1)+\frac{4\gamma}{j}(\cos(\theta)-1)+4\gamma^2(\cos(\theta)-1)^2+\frac{8\gamma^2}{j}(\cos(\theta)-1)^2+\frac{4\gamma^2}{j^2}(\cos(\theta) - 1)^2+\frac{4\gamma^2}{j^2}\sin^2(\theta) \tag{3.6}$$

Isolating the red and blue terms:

$$\frac{4\gamma^2}{j^2}(\cos(\theta) - 1)^2+\frac{4\gamma^2}{j^2}\sin^2(\theta) = \frac{4\gamma^2}{j^2}\cos^2(\theta) - \frac{8\gamma^2}{j^2}\cos(\theta) + \frac{4\gamma^2}{j^2}+\frac{4\gamma^2}{j^2}\sin^2(\theta)$$

$$= \frac{4\gamma^2}{j^2}\cos^2(\theta)+\frac{4\gamma^2}{j^2}\sin^2(\theta)-\frac{8\gamma^2}{j^2}\cos(\theta) + \frac{4\gamma^2}{j^2} = \frac{4\gamma^2}{j^2}(\cos^2(\theta)+\sin^2(\theta))-\frac{8\gamma^2}{j^2}\cos(\theta)+\frac{4\gamma^2}{j^2}$$

$$= -\frac{8\gamma^2}{j^2}\cos(\theta) + \frac{8\gamma^2}{j^2} = -\frac{8\gamma^2}{j^2}(\cos(\theta) - 1)$$

Substituting this back into (3.6):

$$|G|^2 = 1+4\gamma(\cos(\theta)-1)+\frac{4\gamma}{j}(\cos(\theta)-1)+4\gamma^2(\cos(\theta)-1)^2+\frac{8\gamma^2}{j}(\cos(\theta)-1)^2-\frac{8\gamma^2}{j^2}(\cos(\theta)-1) := \mathbf{G}(\theta,\gamma)$$

In a discrete Fourier transform over a finite number of grid points, the wavenumber k would only take on integer values. But in von Neumann analysis, we are doing a continuous mode analysis, so we treat k as a real-valued variable, as if we have an infinitely fine spectrum of frequencies. Then, it is meaningful to ask how a single

mode evolves for any real k, so let us find the value of k that maximizes $|G|^2$. Since this value will represent the worst-case scenario, computing the stability condition for this k will ensure that it holds for all other values.

To find the critical points with respect to k, we will analyze the derivative of $\mathbf{G}(\theta, \gamma)$ with respect to $\theta = k\Delta r$.

$$\frac{\partial}{\partial\theta}\mathbf{G}(\theta,\gamma) = -4\gamma\sin(\theta) - \frac{4\gamma}{j}\sin(\theta) - 8\gamma^2\sin(\theta)(\cos(\theta)-1) - \frac{16\gamma^2}{j}\sin(\theta)(\cos(\theta)-1) + \frac{8\gamma^2}{j^2}\sin(\theta)$$

$$= \sin(\theta)\left(-4\gamma - \frac{4\gamma}{j} - 8\gamma^2\cos(\theta) + 8\gamma^2 - \frac{16\gamma^2}{j}\cos(\theta) + \frac{16\gamma^2}{j} + \frac{8\gamma^2}{j^2}\right) = 0$$

Let $A = -4\gamma - \frac{4\gamma}{j} + 8\gamma^2 + \frac{16\gamma^2}{j} + \frac{8\gamma^2}{j^2}$ and $B = 8\gamma^2 + \frac{16\gamma^2}{j}$, and rewrite:

$$\frac{\partial}{\partial\theta}\mathbf{G}(\theta,\gamma) = \sin(\theta)\Big(A - B\cos(\theta)\Big) = 0$$

This expression equals 0 when $\sin(\theta) = 0$, or when $A - B\cos(\theta_*) = 0$. Note that the second solution corresponds to $\theta_* = \arccos\left(\frac{A}{B}\right)$, which exists when $\frac{A}{B} \in [-1, 1]$. Note also that $\gamma = \frac{\alpha\Delta t}{\Delta r^2}$, and $\alpha, \Delta t, \Delta r \in \mathbb{R}^+$, so $\gamma > 0$. Then, $B = 8\gamma^2 + \frac{16\gamma^2}{j} > 0$, and thus the fraction $\frac{A}{B}$ is defined everywhere. Now let us find the bounds on γ such that a solution of the form $\theta_* = \arccos\left(\frac{A}{B}\right)$ exists.

$$-1 \le \frac{A}{B} \le 1 \implies -B \le A \text{ and } A \le B$$

1) $$-B \le A \implies -8\gamma^2 - \frac{16\gamma^2}{j} \le -4\gamma - \frac{4\gamma}{j} + 8\gamma^2 + \frac{16\gamma^2}{j} + \frac{8\gamma^2}{j^2}$$

$$\implies 4\gamma + \frac{4\gamma}{j} \le 16\gamma^2 + \frac{32\gamma^2}{j} + \frac{8\gamma^2}{j^2} \implies 1 + \frac{1}{j} \le 4\gamma + \frac{8\gamma}{j} + \frac{2\gamma}{j^2}$$

$$\implies j^2 + j \le \gamma\,(4j^2 + 8j + 2) \implies \frac{j^2 + j}{4j^2 + 8j + 2} \le \gamma$$

2) $$A \le B \implies -4\gamma - \frac{4\gamma}{j} + 8\gamma^2 + \frac{16\gamma^2}{j} + \frac{8\gamma^2}{j^2} \le 8\gamma^2 + \frac{16\gamma^2}{j}$$

$$\implies -4\gamma - \frac{4\gamma}{j} + \frac{8\gamma^2}{j^2} \le 0 \implies \frac{8\gamma^2}{j^2} \le 4\gamma + \frac{4\gamma}{j} \implies \frac{2}{j^2}\gamma \le 1 + \frac{1}{j}$$

$$\implies \gamma \leq \frac{j^2 + j}{2}$$

Using **1)** and **2)**, for any fixed j, $\frac{A}{B} \in [-1, 1]$ when $\frac{j^2 + j}{4j^2 + 8j + 2} \leq \gamma \leq \frac{j^2 + j}{2}$.

Additionally, note that $\mathbf{G}(\theta, \gamma)$ is 2π-periodic and even, so we only need to consider critical points in the range $[0,\ \pi]$. Then, we can see that for any fixed γ, $\mathbf{G}(\theta, \gamma) = |G|^2$ has critical points at 0, π, and θ_* such that $A = B\cos(\theta_*)$. Recall from (3.5):

$$|G|^2 = \mathbf{G}(\theta, \gamma) = \left(1 + 2\gamma(\cos(\theta) - 1) + \frac{2\gamma}{j}(\cos(\theta) - 1)\right)^2 + \left(\frac{2\gamma}{j}\sin(\theta)\right)^2$$

1) Critical point at $\theta = 0$:

$$\mathbf{G}(0, \gamma) = \left(1 + 2\gamma(\cos(0) - 1) + \frac{2\gamma}{j}(\cos(0) - 1)\right)^2 + \left(\frac{2\gamma}{j}\sin(0)\right)^2$$

$$= \left(1 + 2\gamma \cdot 0 + \frac{2\gamma}{j} \cdot 0\right)^2 + \left(\frac{2\gamma}{j} \cdot 0\right)^2 = (1)^2 = 1$$

Then, at the critical point $\theta = 0$, $|G|^2 = 1$ for all values of γ. Since we want γ that ensures $|G|^2 \leq 1$, and as shown above, all values of γ satisfy this condition, we may disregard $\theta = 0$ in the context of the stability condition.

2) Critical point at $A = B\cos(\theta_*)$:

$$\frac{\partial^2}{\partial\theta^2}\mathbf{G}(\theta, \gamma) = \frac{\partial}{\partial\theta}\left(\sin(\theta)\Big(A - B\cos(\theta)\Big)\right) = A\cos(\theta) - B\cos^2(\theta) + B\sin^2(\theta)$$

Since we have $A = B\cos(\theta_*)$, $A\cos(\theta_*) - B\cos^2(\theta_*) + B\sin^2(\theta_*) = B\sin^2(\theta_*)$.

Also, recall that $\theta_* \in (0, \pi)$, so $\sin^2(\theta_*) > 0$. Then, $\frac{\partial^2}{\partial\theta^2}\mathbf{G}(\theta_*, \gamma) = B\sin^2(\theta_*) > 0$. According to the second derivative test, this implies θ_* is a minimum. If this critical point minimizes $|G|^2$, then it does not threaten stability for any value of γ. Then, we can conclude that this critical point may also be disregarded in the context of the stability condition, as it does not reveal any information about the constraint on γ.

3) Critical point at $\theta = \pi$:

$$\mathbf{G}(\pi,\gamma) = \left(1 + 2\gamma(\cos(\pi) - 1) + \frac{2\gamma}{j}(\cos(\pi) - 1)\right)^2 + \left(\frac{2\gamma}{j}\sin(\pi)\right)^2 = \left(1 - 4\gamma - \frac{4\gamma}{j}\right)^2 \leq 1$$

$$\implies \left|1 - 4\gamma - \frac{4\gamma}{j}\right| \leq 1 \implies -1 \leq 1 - 4\gamma - \frac{4\gamma}{j} \leq 1 \implies -2 \leq -4\gamma - \frac{4\gamma}{j} \leq 0$$

$$\implies 0 \leq 4\gamma + \frac{4\gamma}{j} \leq 2 \implies 0 \leq \gamma\left(1 + \frac{1}{j}\right) \leq \frac{1}{2} \implies \gamma \leq \frac{1}{2\left(1 + \frac{1}{j}\right)}$$

By construction, $j = 1, 2, 3, ..., \frac{1}{\Delta r}$. Then, $j = 1$ maximizes the denominator, and thus provides the strictest bound on γ. Then, we conclude that for $\gamma \in \left(0, \frac{1}{4}\right]$, this critical point satisfies $|G|^2 \leq 1$ for all j.

Since $\theta = \pi$ is the only critical point that provides a restriction on the value of γ, we can conclude that the entire scheme is stable $\forall j$ when $\gamma \in \left(0, \frac{1}{4}\right]$.

$$\gamma \in \left(0, \frac{1}{4}\right] \implies \frac{\alpha\Delta t}{\Delta r^2} \leq \frac{1}{4}, \text{ so the stability condition is } \Delta t \leq \frac{\Delta r^2}{4\alpha}.$$

Figure 3.1 visualizes one complete θ-period of $|G|^2$ for $\gamma = 0.2$, which is below the stability threshold of 0.25. Notice that at $\theta = 0$, $|G|^2 = 1$ as expected. Notice also the second local maximum at π. Since $\gamma < 0.25$, $|G|^2 \leq 1$, as required. Notice finally, that there is a local minimum between 0 and π, which is the critical point θ_* such that $A = B\cos(\theta_*)$.

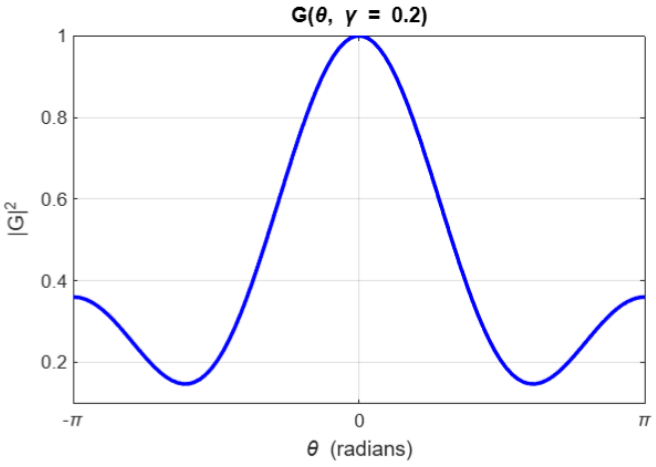

Figure 3.1: $|G|^2$ for γ Below the Stability Threshold

Figure 3.2 visualizes $|G|^2$ when γ is exactly at the stability threshold. As the analysis suggested, $|G|^2 \leq 1$ for all θ. Figure 3.3 visualizes $|G|^2$ for $\gamma = 0.3$, which is above the stability threshold. Notice that $\max_{\theta \in [-\pi,\pi]} \left(|G|^2\right) > 1$, which renders the scheme unstable. Figure 3.9 shows the effect of unstable step sizing in the context of the model output.

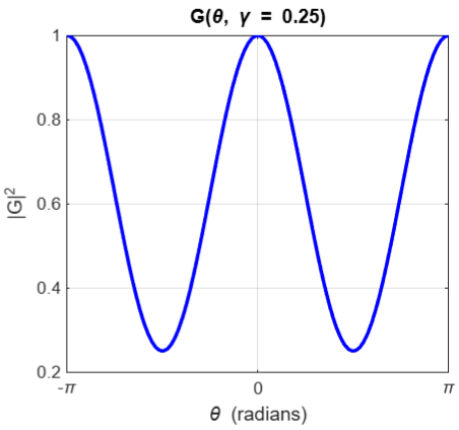

Figure 3.2: $|G|^2$ for γ At Threshold

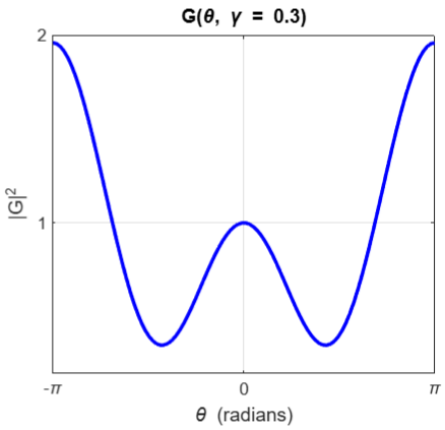

Figure 3.3: $|G|^2$ for γ Above Threshold

The next four figures analyze the $|G|^2$ surface on the $\gamma - \theta$ grid. Figures 3.4 and 3.5 show the surface $|G|^2$ from different angles, for $\gamma \in (0, 0.4]$ and $\theta \in [-\pi, \pi]$. A red line is added at $\gamma = 0.25$, marking the computed stability condition.

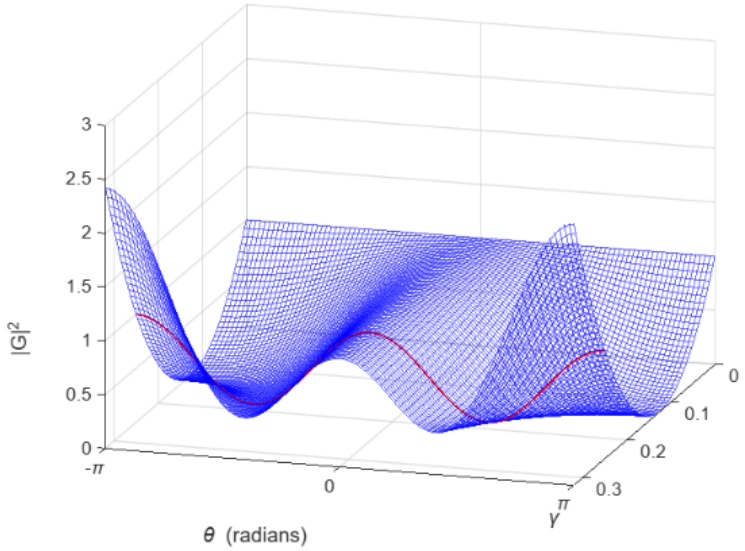

Figure 3.4: $|G|^2$ on (θ, γ)-grid

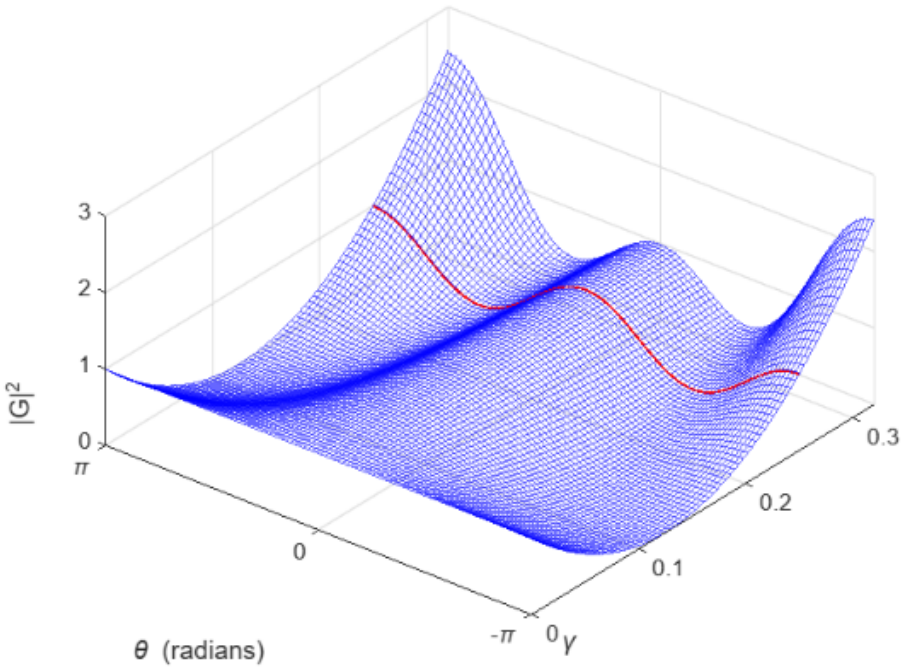

Figure 3.5: $|G|^2$ on (θ, γ)-grid, Rotated

Figure 3.6 adds a plane at $|G|^2 = 1$ so that we may observe the γ values for which $|G|^2 > 1$ for some $\theta \in [-\pi, \pi]$.

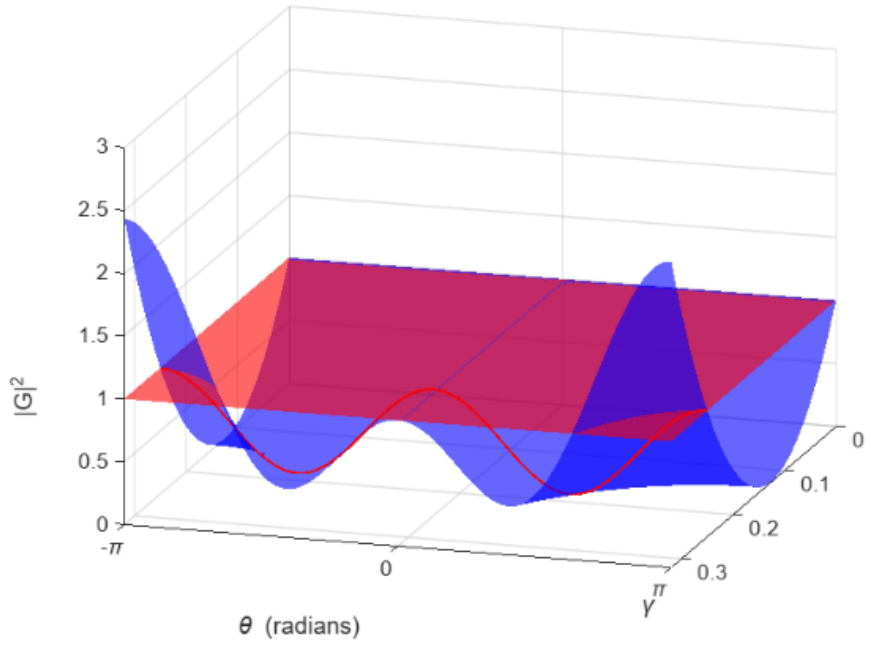

Figure 3.6: Behavior of $|G|^2$ at γ Threshold

Finally, Figure 3.7 is a birdseye view of Figure 3.6. This angle makes it easier to see the exact points at which the blue surface surpasses the red plane, which corresponds to the γ values for which the scheme becomes unstable. Observe that the right hand side of the red line, which represents the stability threshold, coincides exactly with all the γ values for which $|G|^2 > 1$ for some θ.

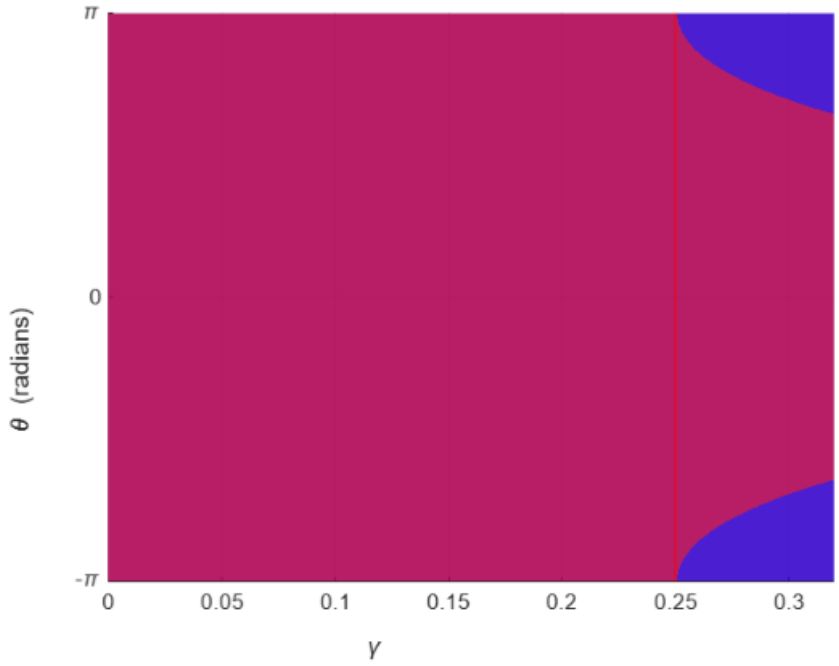

Figure 3.7: Behavior of $|G|^2$ at γ Threshold, Birdseye

Additionally, Figures 3.8 and 3.9 provide a comparison between the outputs of the numerical scheme for stable and unstable step sizing. The only difference between the two outputs is the size of Δt.

Notice that the unstable version claims that the internal temperature of the egg reaches 10^{94} $^\circ C$ within 10 minutes, which simply cannot be true as the core of Earth is about 5200 $^\circ C$. All remaining simulation using Method 1 run with step sizes that satisfy $\Delta t \leq \dfrac{\Delta r^2}{4\alpha}$.

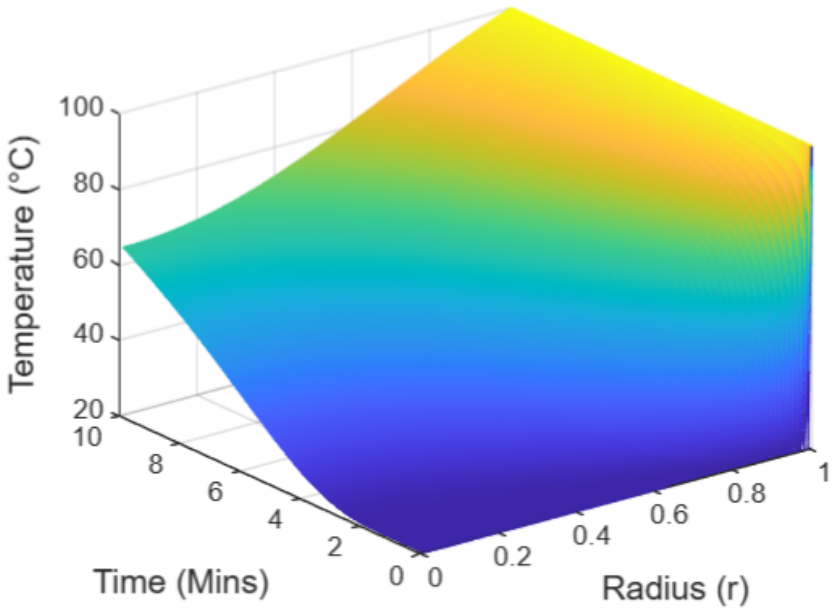

Figure 3.8: Numerical Scheme Output for $\gamma = 0.2$

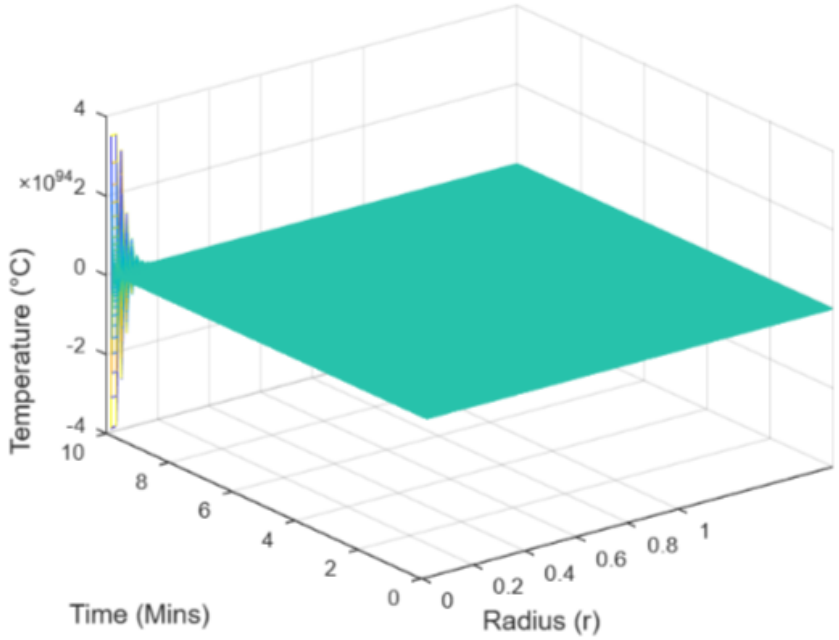

Figure 3.9: Numerical Scheme Output for $\gamma = 0.3$

3.1.2 Truncation Error

Radius Axis: For a function $u(r,t)$ that is four times differentiable in space,

$$u(r+\Delta r,t) = u(r,t)+\Delta r\frac{\partial u}{\partial r}(r,t)+\frac{\Delta r^2}{2}\frac{\partial^2 u}{\partial r^2}(r,t)+\frac{\Delta r^3}{6}\frac{\partial^3 u}{\partial r^3}(r,t)+\frac{\Delta r^4}{12}\frac{\partial^4 u}{\partial r^4}(r,t)+o(\Delta r^4)$$

Let us take any $u_j^n = u(r_j,t^n)$, and rewrite $u_{j+1}^n = u(r_j+\Delta r,t^n)$ and $u_{j-1}^n = u(r_j-\Delta r,t^n)$ using the above Taylor expansion about u_j^n:

$$u_{j+1}^n = u_j^n+\Delta r\frac{\partial u}{\partial r}(r_j,t^n)+\frac{\Delta r^2}{2}\frac{\partial^2 u}{\partial r^2}(r_j,t^n)+\frac{\Delta r^3}{6}\frac{\partial^3 u}{\partial r^3}(r_j,t^n)+\frac{\Delta r^4}{12}\frac{\partial^4 u}{\partial r^4}(r_j,t^n)+o(\Delta r^4)$$

$$u_{j-1}^n = u_j^n-\Delta r\frac{\partial u}{\partial r}(r_j,t^n)+\frac{\Delta r^2}{2}\frac{\partial^2 u}{\partial r^2}(r_j,t^n)-\frac{\Delta r^3}{6}\frac{\partial^3 u}{\partial r^3}(r_j,t^n)+\frac{\Delta r^4}{12}\frac{\partial^4 u}{\partial r^4}(r_j,t^n)+o(\Delta r^4)$$

We are approximating $\dfrac{\partial u}{\partial r}$ by $\dfrac{u_{j+1}^n-u_j^n}{\Delta r}$. So,

$$\frac{u_{j+1}^n-u_j^n}{\Delta r} = \frac{1}{\Delta r}\Big(\cancel{u_j^n} + \Delta r\frac{\partial u}{\partial r}(r_j,t^n) + \frac{\Delta r^2}{2}\frac{\partial^2 u}{\partial r^2}(r_j,t^n)$$

$$+ \frac{\Delta r^3}{6}\frac{\partial^3 u}{\partial r^3}(r_j,t^n) + \frac{\Delta r^4}{12}\frac{\partial^4 u}{\partial r^4}(r_j,t^n) + o(\Delta r^4) - \cancel{u_j^n}\Big)$$

$$= \frac{\partial u}{\partial r}(r_j,t^n)+\frac{\Delta r}{2}\frac{\partial^2 u}{\partial r^2}(r_j,t^n) + \frac{\Delta r^2}{6}\frac{\partial^3 u}{\partial r^3}(r_j,t^n) + \frac{\Delta r^3}{12}\frac{\partial^4 u}{\partial r^4}(r_j,t^n) + o(\Delta r^3)$$

Note that the error term (in red) has leading coefficient $\dfrac{\Delta r}{2}$, so our truncation error for this approximation is first order.

We are also approximating $\dfrac{\partial^2 u}{\partial r^2}$ by $\dfrac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{\Delta r^2}$. Then,

$$\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta r^2} = \frac{1}{\Delta r^2}\Big(\Big(u_j^n + \Delta r\frac{\partial u}{\partial r}(r_j,t^n) + \frac{\Delta r^2}{2}\frac{\partial^2 u}{\partial r^2}(r_j,t^n) + \frac{\Delta r^3}{6}\frac{\partial^3 u}{\partial r^3}(r_j,t^n)$$

$$+ \frac{\Delta r^4}{12}\frac{\partial^4 u}{\partial r^4}(r_j,t^n) + o(\Delta r^4)\Big) - 2u_j^n$$

$$+ \Big(u_j^n - \Delta r\frac{\partial u}{\partial r}(r_j,t^n) + \frac{\Delta r^2}{2}\frac{\partial^2 u}{\partial r^2}(r_j,t^n) - \frac{\Delta r^3}{6}\frac{\partial^3 u}{\partial r^3}(r_j,t^n)$$

$$+ \frac{\Delta r^4}{12}\frac{\partial^4 u}{\partial r^4}(r_j,t^n) + o(\Delta r^4)\Big)\Big)$$

$$= \frac{1}{\Delta r^2}\Big(\Delta r^2\frac{\partial^2 u}{\partial r^2}(r_j,t^n) + \frac{\Delta r^4}{6}\frac{\partial^4 u}{\partial r^4}(r_j,t^n) + o(\Delta r^4)\Big)$$

$$= \frac{\partial^2 u}{\partial r^2}(r_j,t^n) + \frac{\Delta r^2}{6}\frac{\partial^4 u}{\partial r^4}(r_j,t^n) + o(\Delta r^2)$$

The error term (in red) has coefficient $\frac{\Delta r^2}{6}$, so our truncation error for this approximation is second order.

By construction we choose $\Delta r << 1$, so $\Delta r^2 << \Delta r$. So, the total truncation error in space has dominating coefficient proportional to Δr, and thus the method is first order consistent in space.

Time Axis: For a function $u = u(r,t)$ that is twice differentiable in time,

$$r(r,t+\Delta t) = u(r,t) + \Delta t\frac{\partial u}{\partial t}(r,t) + \frac{\Delta t^2}{2}\frac{\partial^2 u}{\partial t^2}(r,t) + o(\Delta t^2)$$

We approximate $\frac{\partial u}{\partial t}$ by $\frac{u_j^{n+1} - u_j^n}{\Delta t}$. This computation is analogous to the first derivative approximation in space. Thus, we have shown the scheme is first order consistent in time.

3.1.3 Implementation: The Medium Boil

In this section, we will use the current model to predict the internal temperatures of eggs boiled for 7, 8, and 9 minutes. The simulation assumes the egg is taken right out of the fridge and has uniform internal temperature of $6°C$.

Figure 3.10 is an example water temperature function. Notice that the surrounding water is at $100°C$ for 7 minutes. After removing the eggs from the pot, we rinse them under cold tap water, which was measured at $8°C$. We extend the water temperature function in this way to account for the fact that heat does not stop diffusing towards the center as soon as it is removed from boiling water. The energy that has already entered the egg's boundary needs additional time to continue traveling, and thus modeling the cooling process of the system makes the figure more accurate to the real world.

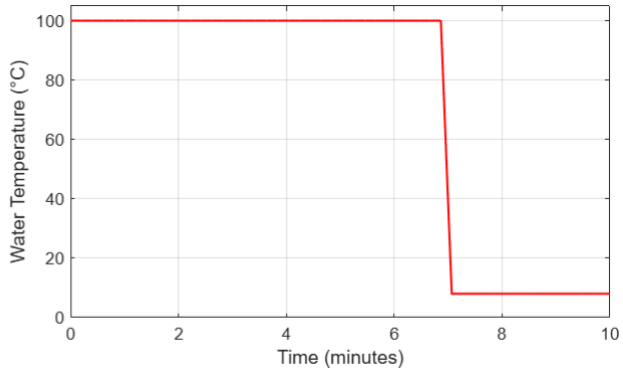

Figure 3.10: Water Temperature Function Example: 7-Minute Boil

Figure 3.11 shows the surface generated using the above water temperature function and Method 1, with Δt = 0.1 and Δr=0.01. The cyan plane is located at $68°C$, the temperature at which the yolk is halfway coagulated. The purple plane is located at $70°C$, the temperature at which the yolk fully solidifies. Additionally, the red dashed line is the water temperature function, and is located at $r = 1$, which represents the egg shell boundary. Notice that a large portion of the egg yolk ($0 \leq r \leq 0.678$) has not surpassed the planes, which suggests that the center of the yolk must still be runny.

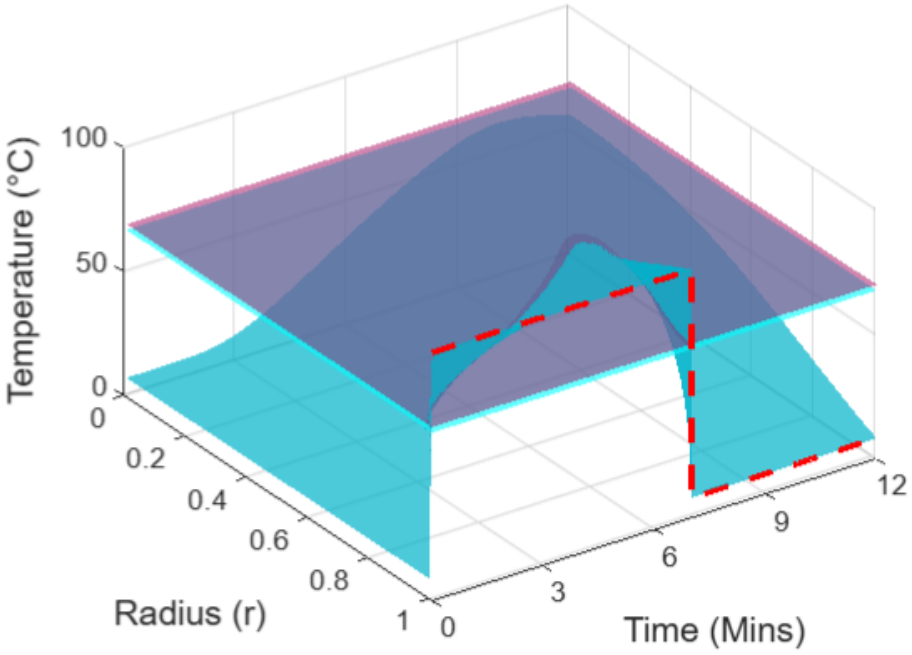

Figure 3.11: 7-Minute Boil, Method 1, $\Delta t = 0.1$, $\Delta r = 0.01$

Figure 3.12 visualizes the same surface from a different angle so we can observe temperature at $r = 0$, which represents the center of the egg and is highlighted in dark blue.

Figure 3.13 visualizes the same plane from a birdseye view. This perspective makes it easier to tell the percentage of the radius that has surpassed the two planes. The dashed black line is located at $r = 0.678$, which is when the egg white transitions to the egg yolk. Notice that only a small part of the yolk has surpassed $68°C$, and an even smaller part has surpassed $70°C$. Then, we can once again conclude that the core of the egg is not coagulated after being placed in boiling water for 7 minutes.

Figure 3.14 is a picture of a fridge-cold egg that was placed in boiling water for 7 minutes and rinsed under tap water for 3 minutes. Notice that the egg white is solid, and the yolk is still runny. This outcome is exactly what the model suggested.

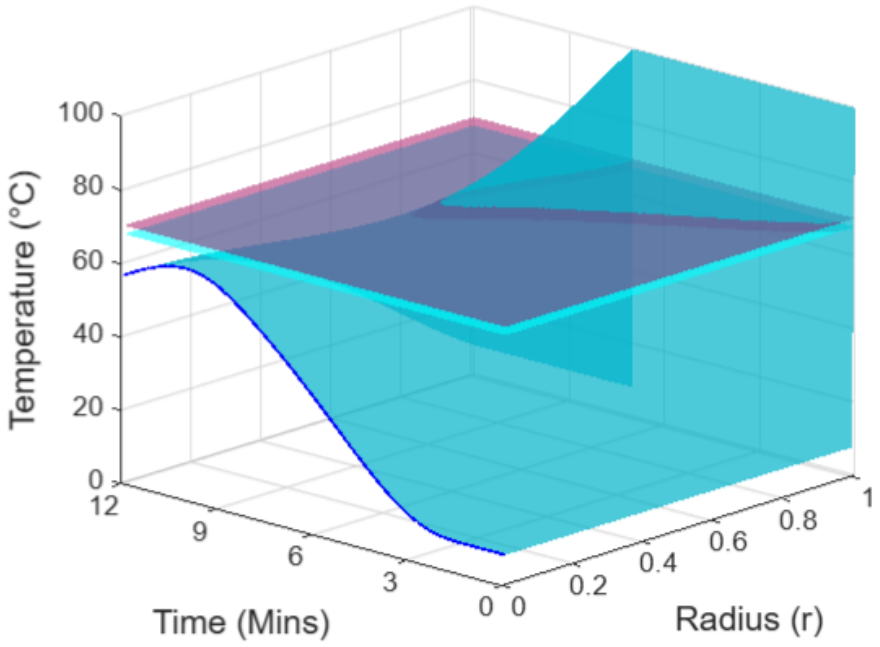

Figure 3.12: 7-Minute Boil, Method 1, Rotated, $\Delta t = 0.1$, $\Delta r = 0.01$

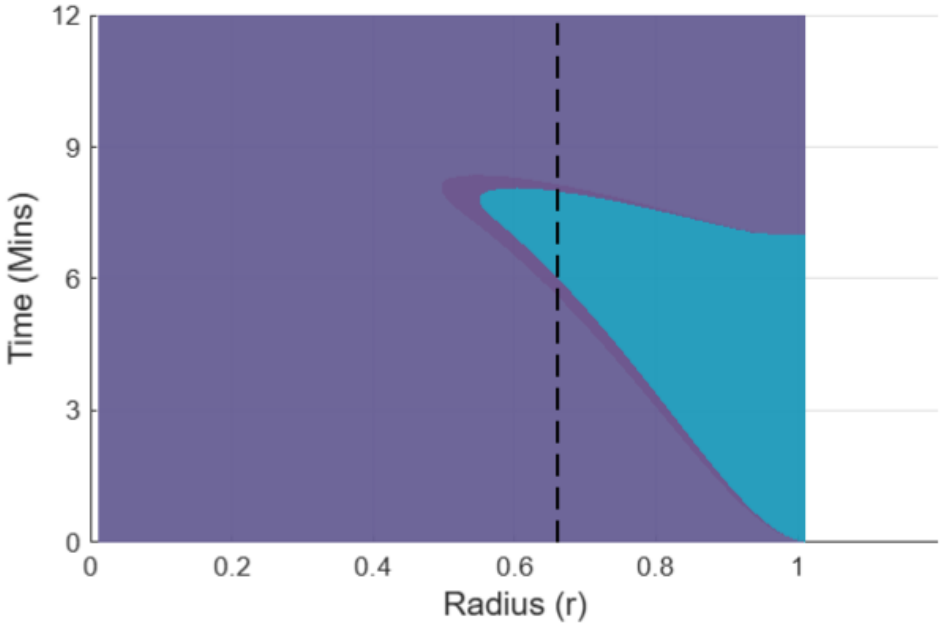

Figure 3.13: 7-Minute Boil, method 1, Birdseye, $\Delta t = 0.1$, $\Delta r = 0.01$

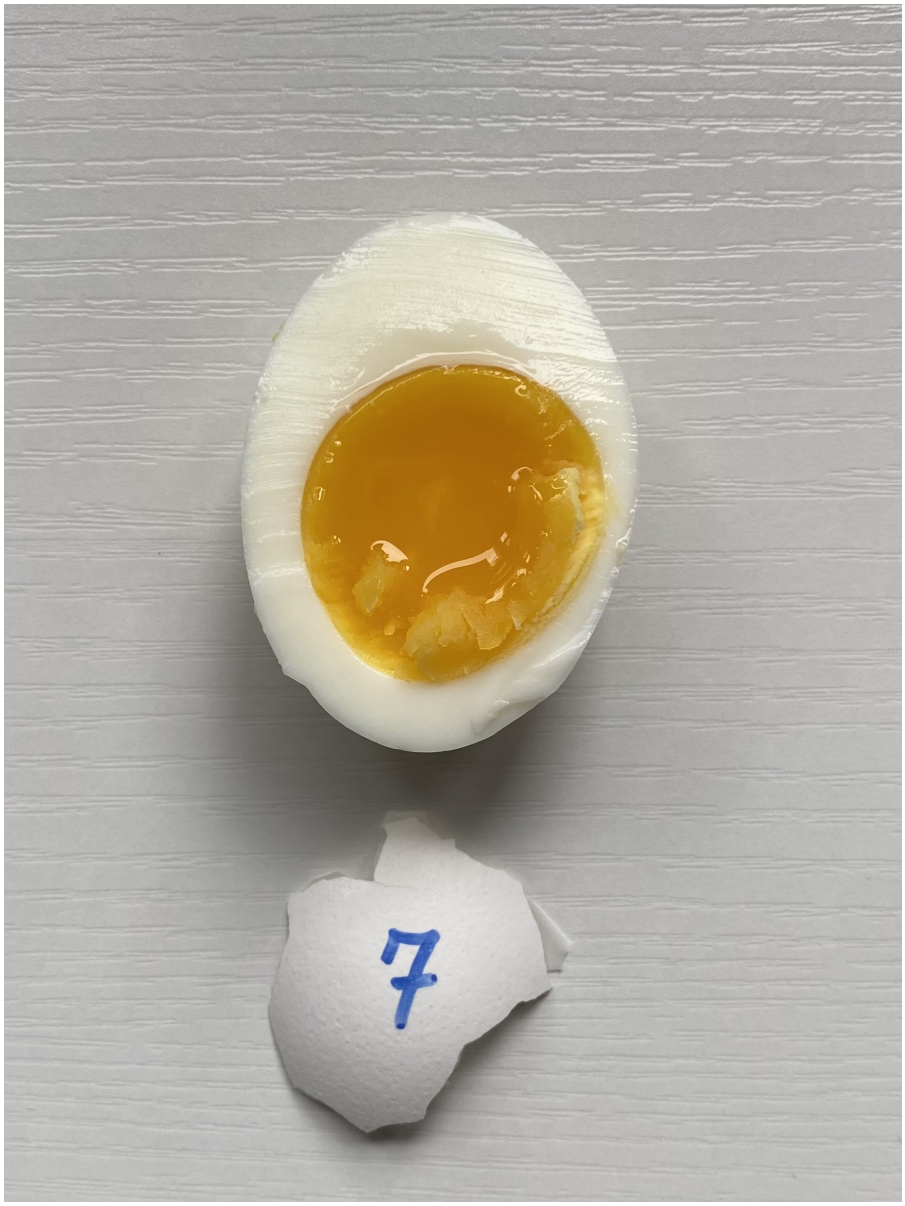

Figure 3.14: 7-Minute Boil, Real Egg

Figure 3.15 is the output of an 8 minute boil, using the same model and step sizes. The red dashed line is located at $r = 1$, so it refers to the water temperature function. Observe that in this figure, the entire egg has surpassed the cyan plane located at $68°C$, so the yolk must be completely coagulated. Additionally, only some of the yolk has surpassed the purple plane located at $70°C$, meaning that only some of it has solidified. Then, this figure suggests that the center is neither runny nor liquid, which fits our definition of a medium boil from section 1.1.

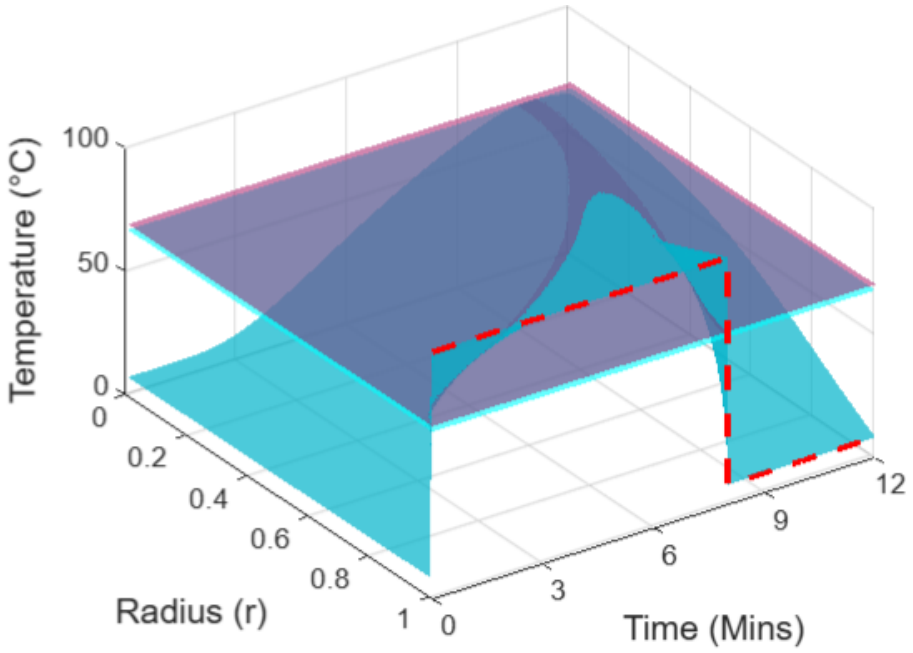

Figure 3.15: 8-Minute Boil, Method 1, $\Delta t = 0.1$, $\Delta r = 0.01$

Figure 3.16 rotates the same surface so we can observe the temperature at the center. Notice that the blue line at $r = 0$ has its vertex between the two planes, further solidifying the medium boil idea. Figure 3.17 is the same surface from above, with a black dashed line at the threshold between egg white and egg yolk.

Finally, Figures 3.18 and 3.19 are pictures of fridge-cold eggs placed in boiling water for 8 minutes and rinsed under tap water for 3 minutes. Observe that for each egg, the yolk is not completely solidified, but it is also not liquid enough to move. This fits with our definition of a medium boil from Section 1.1. In the next sections, we will refer back to the 8-minute boil as the gold standard for a medium boiled egg.

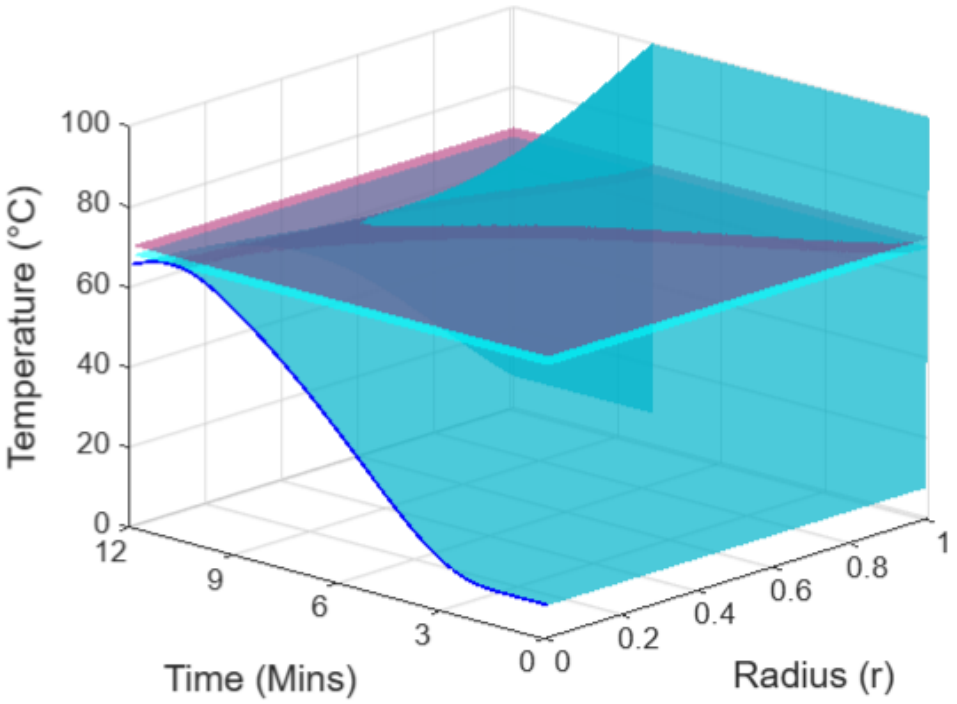

Figure 3.16: 8-Minute Boil, Method 1, Rotated, $\Delta t = 0.1$, $\Delta r = 0.01$

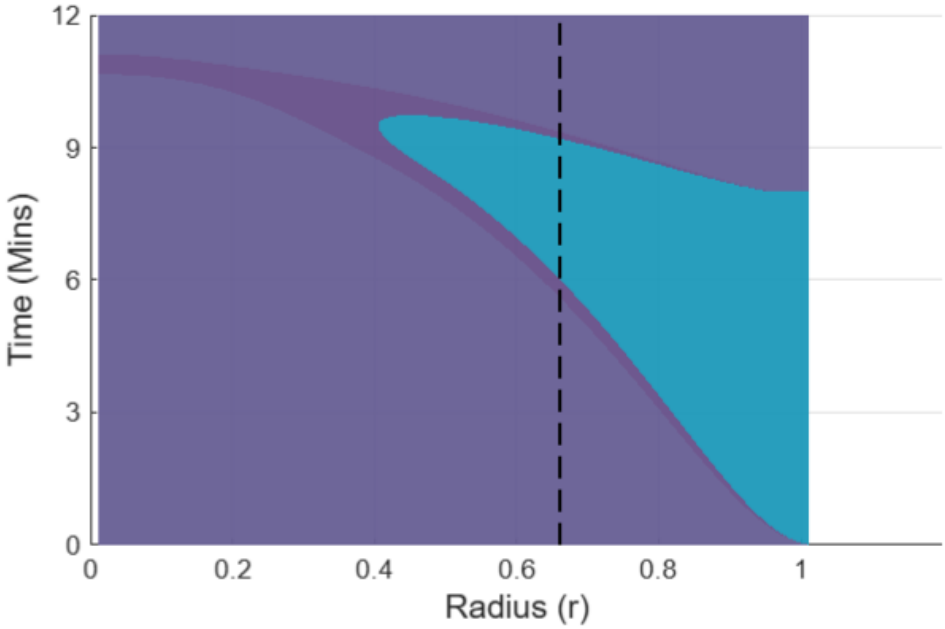

Figure 3.17: 8-Minute Boil, Method 1, Birdseye, $\Delta t = 0.1$, $\Delta r = 0.01$

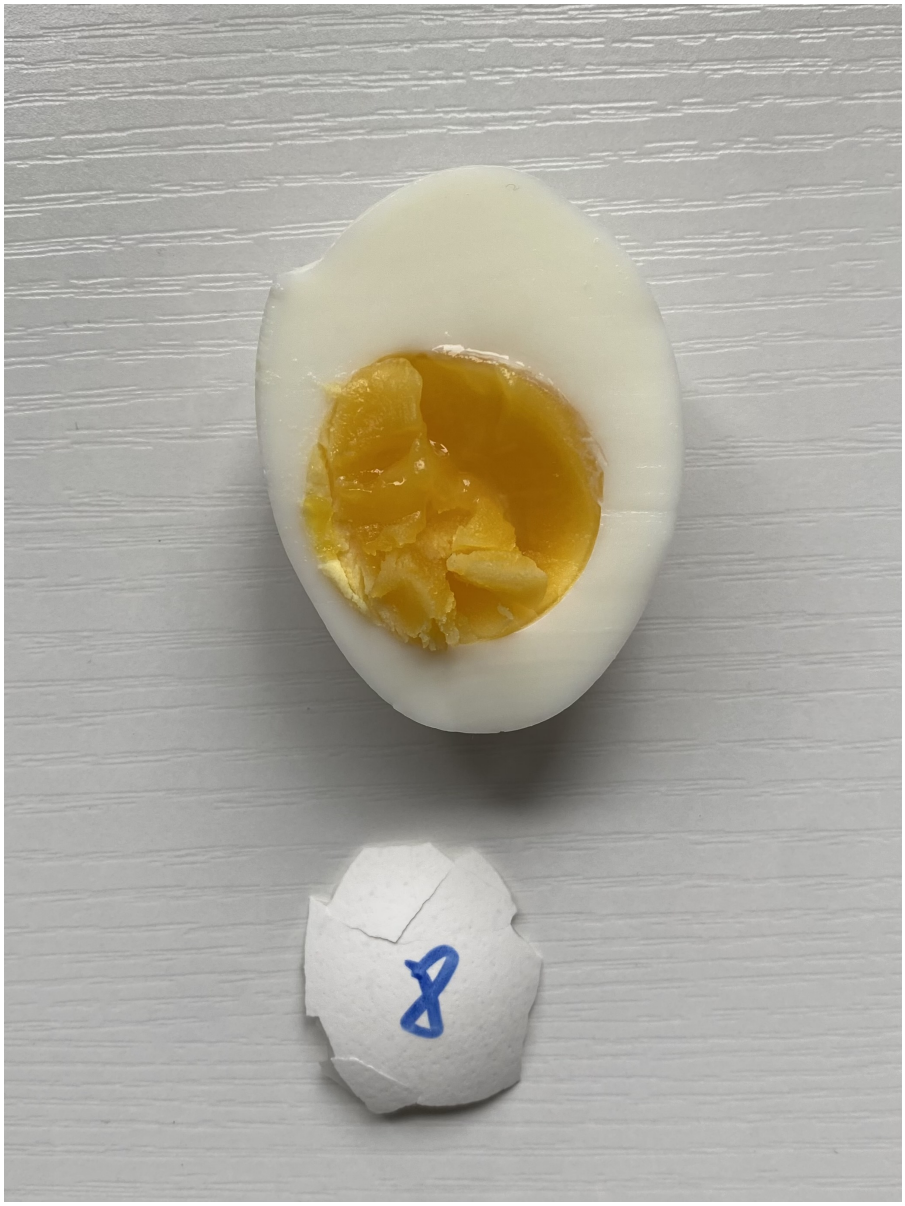

Figure 3.18: 8-Minute Boil, Real Egg

Figure 3.19: 8-Minute Boil, Real Egg

Figure 3.20 simulates a 9-minute boil using the same step sizes as the previous figures. As before, the red dashed line represents the water temperature, and the cyan and purple planes are located at $68°C$ and $70°C$, respectively. Notice from the figure that the entire egg has surpassed the purple plane, meaning that the yolk is completely solidified, and thus the egg should be hard boiled.

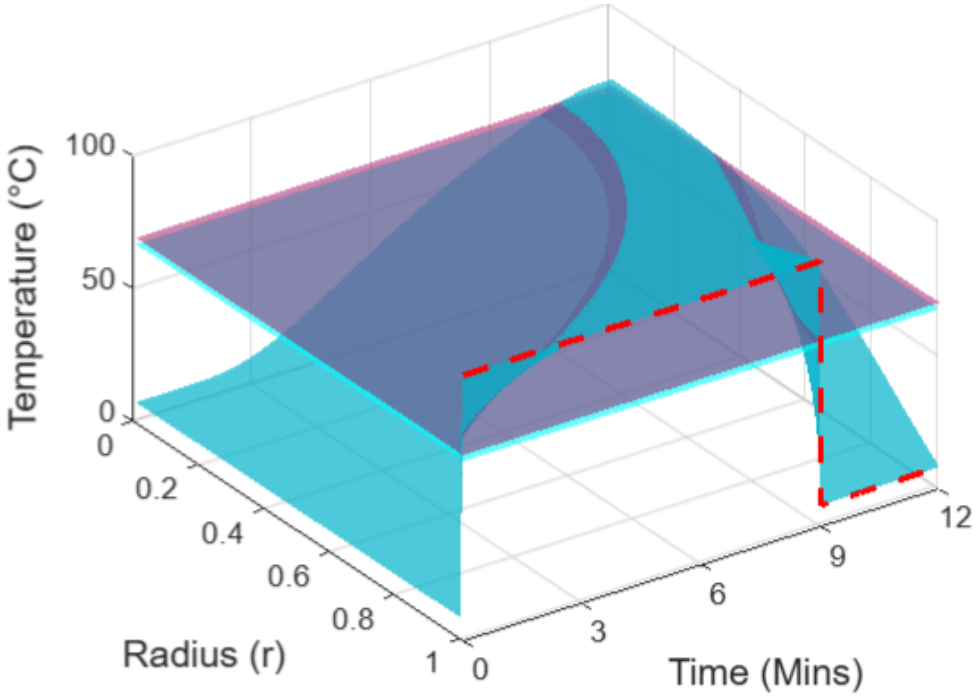

Figure 3.20: 9-Minute Boil, Method 1, $\Delta t = 0.1$, $\Delta r = 0.01$

Figure 3.21 rotates the same surface. Observe the blue line at $r = 0$, and notice that the internal temperature has surpassed $70°C$ everywhere in the egg. Figure 3.22 is a birsdeye view of the 9-minute boil.

Figure 3.23 is a fridge-cold egg boiled for 9 minutes and rinsed under tap water for 3 minutes. Observe that the yolk has completely solidified, so 9 minutes is too long for the purpose of a medium boil.

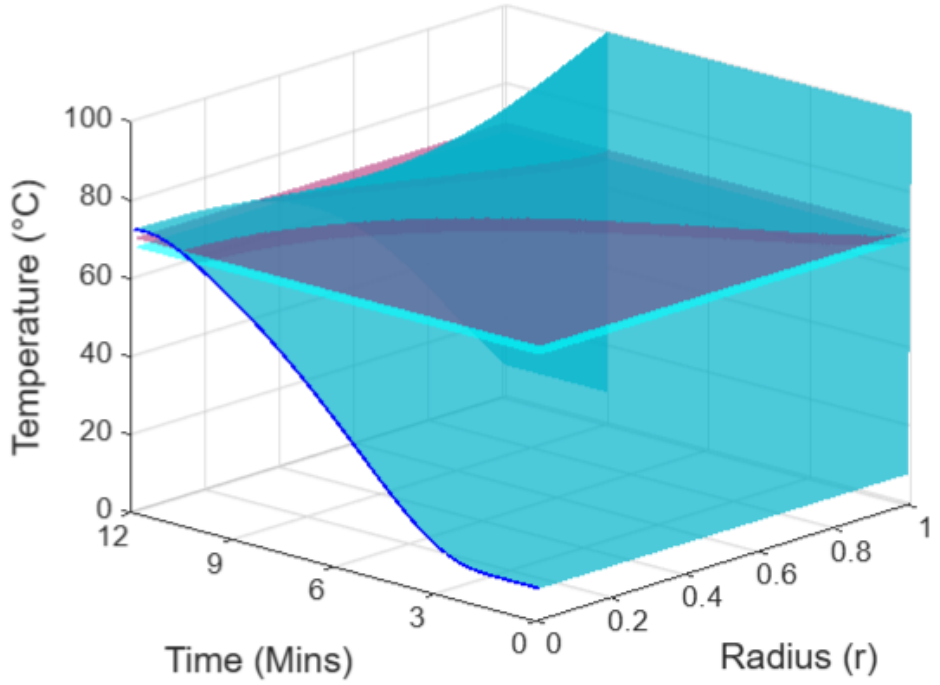

Figure 3.21: 9-Minute Boil, Method 1, Rotated, $\Delta t = 0.1$, $\Delta r = 0.01$

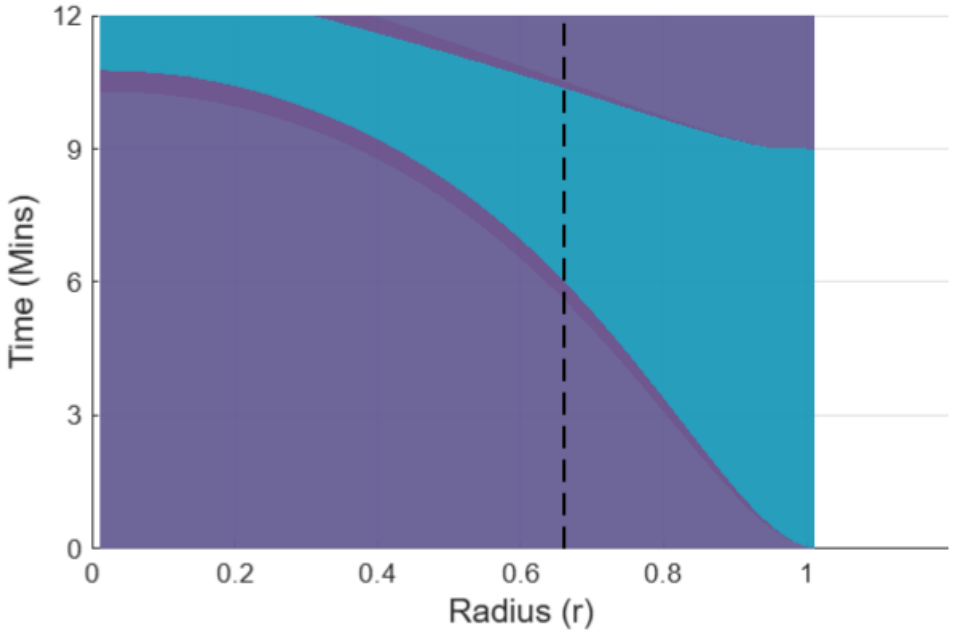

Figure 3.22: 9-Minute Boil, Method 1, Birdseye, $\Delta t = 0.1$, $\Delta r = 0.01$

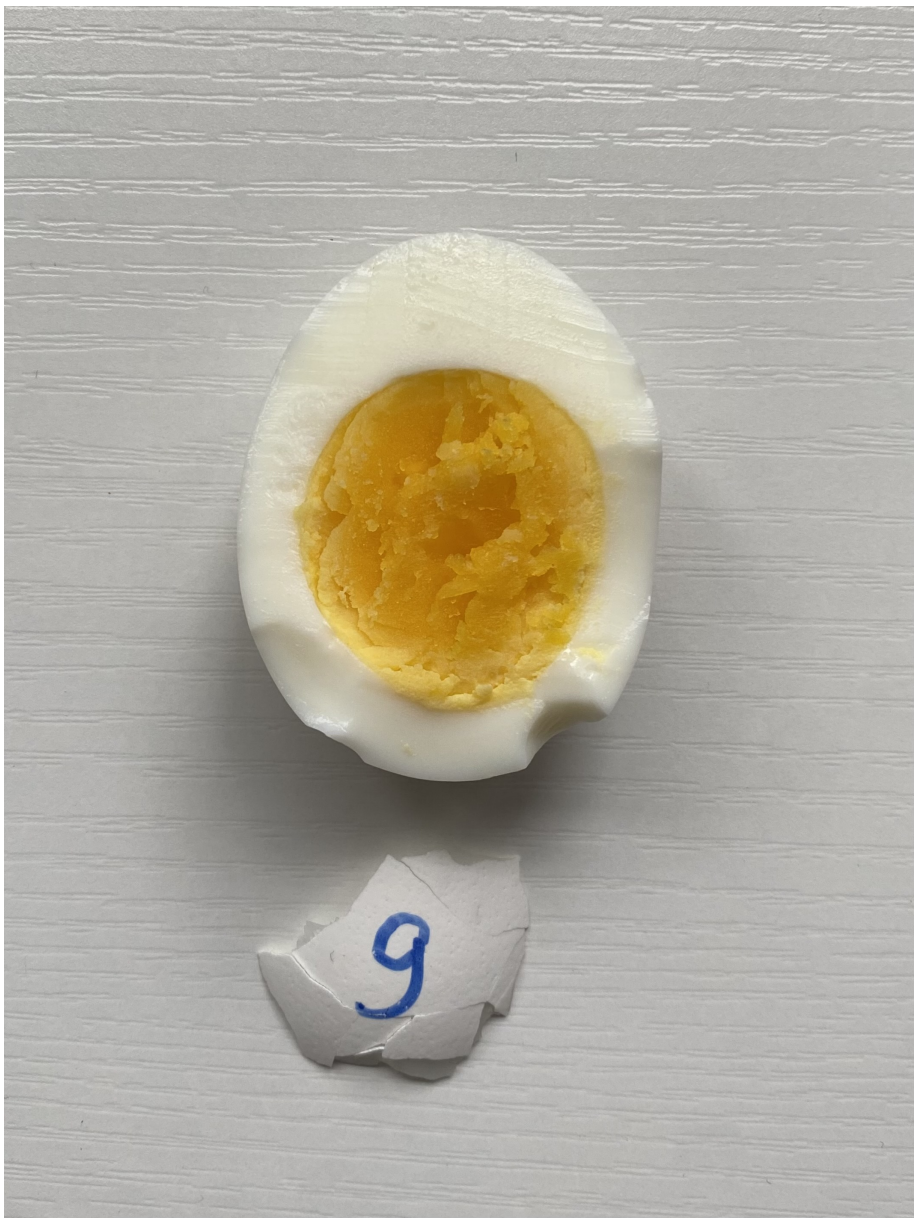

Figure 3.23: 9-Minute Boil, Real Egg

3.2 Method 2

We approximate the derivative at some point (r_j, t^n) using a linear combination of $u(r_j, t^n)$, $u(r_j + \Delta r, t^n)$, and $u(r_j - \Delta r, t^n)$, where Δr is some very small increment of discretization.

$$u'(r_j, t^n) \approx \alpha \cdot u(r_j - \Delta r, t^n) + \beta \cdot u(r_j, t^n) + \gamma \cdot u(r_j + \Delta r, t^n) \tag{3.7}$$

Using a Taylor expansion about $u(r_j, t^n)$, we can rewrite $u(r_j - \Delta r, t^n)$ and $u(r_j + \Delta r, t^n)$ as:

$$u(r_j - \Delta r, t^n) = u(r_j, t^n) - u'(r_j, t^n)\Delta r + \frac{u''(r_j, t^n)}{2}\Delta r^2 - \frac{u'''(r_j, t^n)}{6}\Delta r^3$$

$$u(r_j + \Delta r, t^n) = u(r_j, t^n) + u'(r_j, t^n) \cdot \Delta r + \frac{u''(r_j, t^n)}{2}\Delta r^2 + \frac{u'''(r_j, t^n)}{6}\Delta r^3$$

Rewriting (3.7) using these expansions:

$$\begin{aligned} u'(r_j, t^n) \approx \alpha &\left(u(r_j, t^n) - u'(r_j, t^n)\Delta r + \frac{u''(r_j, t^n)}{2}\Delta r^2 - \frac{u'''(r_j, t^n)}{6}\Delta r^3 \right) + \beta\ u(r_j, t^n) \\ &+\gamma \left(u(r_j, t^n) + u'(r_j, t^n)\Delta r + \frac{u''(r_j, t^n)}{2}\Delta r^2 + \frac{u'''(r_j, t^n)}{6}\Delta r^3 \right) \end{aligned} \tag{3.8}$$

We are interested in the coefficients of the functions, so we re-factor the above expression in the following way:

$$u'(r_j, t^n) \approx (\alpha + \beta + \gamma)\ u(r_j, t^n) + \Delta r\,(\gamma - \alpha)\ u'(r_j, t^n) + \frac{\Delta r^2}{2}(\alpha + \gamma)\ u''(r_j, t^n) + \frac{\Delta r^3}{6}(\gamma - \alpha)\ u'''(r_j, t^n) \tag{3.9}$$

Observe that on the left hand side of (3.9), $u'(r_j, t^n)$ has coefficient 1, and $u(r_j, t^n)$, $u''(r_j, t^n)$, and $u'''(r_j, t^n)$ all have coefficient 0. Using this, we derive the following set of equations:

$$(\alpha + \beta + \gamma) \cdot u(r_j, t^n) = 0 \cdot u(r_j, t^n) \implies \alpha + \beta + \gamma = 0$$

$$\Delta r(\gamma-\alpha)\cdot u'(r_j,t^n) = 1\cdot u'(r_j,t^n) \implies \gamma\Delta r - \alpha\Delta r = 1$$

$$\frac{\Delta r^2}{2}(\alpha+\gamma)\cdot u''(r_j,t^n) = 0\cdot u''(r_j,t^n) \implies \left(\alpha\frac{\Delta r^2}{2}+\gamma\frac{\Delta r^2}{2}\right) = 0$$

Since we have only three unknowns, and the above equations are linearly independent, we can not specify a fourth equation using the coefficient of $u'''(r_j,t^n)$, as this may cause the system to have no real solutions. Then, we assign the term $\frac{\Delta r^3}{6}(\gamma-\alpha)\,u'''(r_j,t^n)$ as the dominating, the order of which will determine the scheme's consistency in time.

We have now redefined our initial problem in the following way:

$$\begin{bmatrix} 1 & 1 & 1 \\ -\Delta r & 0 & \Delta r \\ \frac{\Delta r^2}{2} & 0 & \frac{\Delta r^2}{2} \end{bmatrix}\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix} = \begin{bmatrix}0\\1\\0\end{bmatrix}$$

Which has the solution $\alpha = \frac{1}{2\Delta r}$, $\beta = 0$, $\gamma = \frac{-1}{2\Delta r}$.

Substituting this solution into 3.8 provides us with the following approximation:

$$u'(r_j,t^n) = -\frac{1}{2\Delta r}u(r_j-\Delta r,t^n)+0\cdot u(r_j,t^n)+\frac{1}{2\Delta r}u(r_j+\Delta r,t^n) = \frac{u(r_j+\Delta r,t^n)-u(r_j-\Delta r,t^n)}{2\Delta r} \tag{3.10}$$

Furthermore, substituting the same solution into 2.8 gives:

$$u'(r_j) \approx 0\cdot u(r_j)+1\cdot u'(r_j)+0\cdot u''(r_j)+\Delta r^3\left(\frac{\gamma-\alpha}{6}\right)u'''(r_j) = u'(r_j)-\frac{\Delta r^2}{6}u'''(r_j)$$

Observe that the above equation means $\frac{\Delta r^2}{6}u'''(r_j,t^n)$ approximates the dominating error in our scheme, which implies that the error is of order $o(\Delta r^2)$, or that it is second order consistent in space.

Let us return now to the original heat equation from Chapter 1. Recall that

$$\frac{\partial u}{\partial t} = \alpha\left(\frac{\partial^2 u}{\partial r^2}+\frac{2}{r}\frac{\partial u}{\partial r}\right)$$

As discussed in the previous section, we may use a central differencing scheme for the second derivative, which is second order consistent:

$$\frac{\partial^2 u}{\partial r^2}(r_j, t^n) = \frac{u(r_j - \Delta r, t^n) - 2u(r_j, t^n) + u(r_j + \Delta r, t^n)}{\Delta r^2} \tag{3.11}$$

Finally, we can substitute (3.10) and (3.11) into the heat equation to obtain this approximation to the time derivative:

$$\frac{\partial u}{\partial t} = \alpha\left(\frac{u(r_j - \Delta r) - 2u(r_j) + u(r_j + \Delta r)}{\Delta r^2} + \frac{2}{r}\left(\frac{u(r_j + \Delta r) - u(r_j - \Delta r)}{2\Delta r}\right)\right) \tag{3.12}$$

We may now substitute (3.12) and (3.2) into (3.1) and write:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \alpha\left(\frac{2}{r_j}\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta r} + \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta r^2}\right) \Longrightarrow$$

$$u_j^{n+1} = u_j^n + \alpha \cdot \Delta t\left(\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta r^2} + \frac{1}{r_j}\frac{u_{j+1}^n - u_{j-1}^n}{\Delta r}\right) \tag{3.13}$$

The code for this scheme, which can be found in the appendix, implements the boundary conditions in the same way that Method 1 does. Note that this means the singularity at the center is handled by setting $u(0, t) = u(\Delta r, t)$, which is a first-order approximation. Using a first-order accurate approximation at $r = 0$ is acceptable in this context because it only affects a single point in the domain, while the rest of the scheme uses second-order accurate spatial discretization. As a result, the overall scheme can still exhibit second-order convergence in space, since the lower-order treatment at the singular point has a negligible impact on the global error as the grid is refined.

3.2.1 von Neumann Stability Analysis

We will derive the stability condition for the scheme using von Neumann analysis, as done for the previous method. We start by replacing r_j by $j\Delta r$ and rewriting the

scheme:

$$u_j^{n+1} = u_j^n + \frac{\alpha\Delta t}{\Delta r^2}\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n + \frac{u_{j+1}^n - u_{j-1}^n}{j}\right)$$

Substituting $u_j^n = G^n e^{ikj\Delta r}$:

$$G^{n+1}e^{ikj\Delta r} = G^n e^{ikj\Delta r} + \frac{\alpha\Delta t}{\Delta r^2}G^n e^{ikj\Delta r}\left(e^{ik\Delta r} - 2 + e^{-ik\Delta r} + \frac{e^{ik\Delta r} - e^{-ik\Delta r}}{j}\right)$$

Dividing both sides by $G^n e^{ikj\Delta r}$:

$$G = 1 + \frac{\alpha\Delta t}{\Delta r^2}\left(e^{ik\Delta r} - 2 + e^{-ik\Delta r} + \frac{e^{ik\Delta r} - e^{-ik\Delta r}}{j}\right)$$

Substituting $\gamma = \dfrac{\alpha\Delta t}{\Delta r^2}$ and $\theta = k\Delta r$:

$$G = 1 + \gamma\left(e^{i\theta} - 2 + e^{-i\theta} + \frac{e^{i\theta} - e^{-i\theta}}{j}\right)$$

Using the identities $e^{i\theta} = \cos(\theta) + i\sin(\theta)$ and $e^{-i\theta} = \cos(\theta) - i\sin(\theta)$:

$$G = 1+\gamma\left(\cos(\theta) + i\sin(\theta) - 2 + \cos(\theta) - i\sin(\theta) + \frac{\cos(\theta) + i\sin(\theta) - \big(\cos(\theta) - i\sin(\theta)\big)}{j}\right)$$

$$= 1 + \gamma\left(2\cos(\theta) - 2 + \frac{2i\sin(\theta)}{j}\right) = (1 + 2\gamma(\cos(\theta) - 1)) + i\left(\frac{2\gamma\sin(\theta)}{j}\right)$$

$$\implies |G|^2 = (1 + 2\gamma(\cos(\theta) - 1))^2 + \left(\frac{2\gamma\sin(\theta)}{j}\right)^2$$

Recall from the previous stability analysis that we want to keep $|G|^2 \leq 1$. Notice that j appears only in the denominator of a positive fraction. Since $j = 1, 2, 3, ..., \dfrac{1}{\Delta r}$, the value of j that maximizes $|G|^2$ is $j = 1$. Recall that $r_j = j\Delta r$, so this conclusion implies $r_j = \Delta r$ maximizes $|G|^2$, which means that the computation closest to the center is the most affected by small perturbations. Since $j = 1$ maximizes the instability, computing the stability condition for this point will suffice, as this will ensure that the condition holds for all other j. Then, we have:

$$|G|^2 = (1 + 2\gamma(\cos(\theta) - 1))^2 + \left(2\gamma \sin(\theta)\right)^2$$
$$= 1 + 4\gamma\cos(\theta) - 4\gamma + 4\gamma^2 - 8\gamma^2\cos(\theta) + 4\gamma^2\cos^2(\theta) + 4\gamma^2\sin^2(\theta)$$

Using the identity $\sin^2(\theta) + \cos^2(\theta) = 1$:

$$|G|^2 = 1 + 4\gamma\cos(\theta) - 4\gamma + 8\gamma^2 - 8\gamma^2\cos(\theta)$$

We want $\mathbf{G}(\theta, \gamma) \leq 1$ for all Fourier wavenumbers k, so we can analyze the critical points of $\mathbf{G}(\theta, \gamma)$ with respect to $\theta = k\Delta r$. Since this function is even and periodic, it suffices to focus our analysis on $\theta \in [0, \pi]$.

$$\frac{\partial}{\partial\theta}\mathbf{G}(\theta,\gamma) = \frac{\partial}{\partial\theta}\Big[1 - 4\gamma + 8\gamma^2 + 4\gamma\cos(\theta) - 8\gamma^2\cos(\theta)\Big] = -4\gamma\sin(\theta) + 8\gamma^2\sin(\theta) = 0$$

$$\implies \sin(\theta) = 0, \text{ so } \theta \in \{0, \pi\}$$

Let us substitute these critical points into $\mathbf{G}(\theta, \gamma)$. **For $\theta = 0$:**

$$\mathbf{G}(0,\gamma) = 1 - 4\gamma + 8\gamma^2 + 4\gamma\cos(0) - 8\gamma^2\cos(0) = 1 - 4\gamma + 8\gamma^2 + 4\gamma - 8\gamma^2 = 1$$

So, for the critical point at $\theta = 0$, $|G|^2 = 1$ independent of γ, and thus the critical point does not impact the stability condition.

For $\theta = \pi$:

$$\mathbf{G}(\pi,\gamma) = 1 - 4\gamma + 8\gamma^2 + 4\gamma\cos(\pi) - 8\gamma^2\cos(\pi) = 1 - 4\gamma + 8\gamma^2 - 4\gamma + 8\gamma^2 = 1 - 8\gamma + 16\gamma^2$$

Then, $|G|^2 \leq 1$ when $1 - 8\gamma + 16\gamma^2 \leq 1$, or equivalently when $\gamma \leq \frac{1}{2}$. Since $\gamma = \frac{\alpha\Delta t}{\Delta r^2}$, the stability condition is $\frac{\alpha\Delta t}{\Delta r^2} \leq \frac{1}{2}$, or $\Delta t \leq \frac{\Delta r^2}{2\alpha}$

The code for this scheme, which can be found in the appendix, includes a test for the stability condition: an error message is generated if the step sizes violate the stability condition.

3.2.2 Truncation Error

Radius Axis: For a function $u(r,t)$ that is four times differentiable in space,

$$u(r+\Delta r,t) = u(r,t)+\Delta r\frac{\partial u}{\partial r}(r,t)+\frac{\Delta r^2}{2}\frac{\partial^2 u}{\partial r^2}(r,t)+\frac{\Delta r^3}{6}\frac{\partial^3 u}{\partial r^3}(r,t)+\frac{\Delta r^4}{12}\frac{\partial^4 u}{\partial r^4}(r,t)+o(\Delta r^4)$$

Let us take any $u_j^n = u(r_j,t^n)$, and rewrite $u_{j+1}^n = u(r_j+\Delta r,t^n)$ and $u_{j-1}^n = u(r_j-\Delta r,t^n)$ using the above Taylor expansion about u_j^n

$$u_{j+1}^n = u_j^n+\Delta r\frac{\partial u}{\partial r}(r_j,t^n)+\frac{\Delta r^2}{2}\frac{\partial^2 u}{\partial r^2}(r_j,t^n)+\frac{\Delta r^3}{6}\frac{\partial^3 u}{\partial r^3}(r_j,t^n)+\frac{\Delta r^4}{12}\frac{\partial^4 u}{\partial r^4}(r_j,t^n)+o(\Delta r^4)$$

$$u_{j-1}^n = u_j^n-\Delta r\frac{\partial u}{\partial r}(r_j,t^n)+\frac{\Delta r^2}{2}\frac{\partial^2 u}{\partial r^2}(r_j,t^n)-\frac{\Delta r^3}{6}\frac{\partial^3 u}{\partial r^3}(r_j,t^n)+\frac{\Delta r^4}{12}\frac{\partial^4 u}{\partial r^4}(r_j,t^n)+o(\Delta r^4)$$

We are approximating $\dfrac{\partial u}{\partial r}$ by $\dfrac{u_{j+1}^n-u_{j-1}^n}{2\Delta r}$. So,

$$\begin{aligned}\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta r} &= \frac{1}{2\Delta r}\Big(u_j^n+\Delta r\frac{\partial u}{\partial r}(r_j,t^n)+\frac{\Delta r^2}{2}\frac{\partial^2 u}{\partial r^2}(r_j,t^n)+\frac{\Delta r^3}{6}\frac{\partial^3 u}{\partial r^3}(r_j,t^n)\\ &\quad+\frac{\Delta r^4}{12}\frac{\partial^4 u}{\partial r^4}(r_j,t^n)+o(\Delta r^4)\\ &\quad-\Big(u_j^n-\Delta r\frac{\partial u}{\partial r}(r_j,t^n)+\frac{\Delta r^2}{2}\frac{\partial^2 u}{\partial r^2}(r_j,t^n)-\frac{\Delta r^3}{6}\frac{\partial^3 u}{\partial r^3}(r_j,t^n)\\ &\quad+\frac{\Delta r^4}{12}\frac{\partial^4 u}{\partial r^4}(r_j,t^n)+o(\Delta r^4)\Big)\Big)\\ &= \frac{1}{2\Delta r}\Big(2\Delta r\frac{\partial u}{\partial r}(r_j,t^n)+\frac{\Delta r^3}{3}\frac{\partial^3 u}{\partial r^3}(r_j,t^n)+o(\Delta r^4)\Big)\\ &= \frac{\partial u}{\partial r}(r_j,t^n)+\frac{\Delta r^2}{6}\frac{\partial^3 u}{\partial r^3}(r_j,t^n)+o(\Delta r^3)\end{aligned}$$

Note that the error term (in red) has coefficient $\frac{\Delta r^2}{6}$, so our truncation error for this approximation is second order consistent.

We are also approximating $\frac{\partial^2 u}{\partial r^2}$ by $\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta r^2}$, which we have previously shown to be second order consistent.

So, the total truncation error in space has dominating coefficient proportional to Δr^2, and thus the method is second order consistent in space.

Time Axis: We approximate $\frac{\partial u}{\partial t}$ by $\frac{u_j^{n+1} - u_j^n}{\Delta t}$, which we have previously shown to be first order, so the scheme is first order consistent in time.

Putting these results together,

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + o(\Delta t) = \alpha\left(\frac{\partial^2 u}{\partial r^2} + \frac{1}{r_j}\frac{\partial u}{\partial r} + o(\Delta r^2)\right)$$

Recall from the stability analysis, the CFL condition requires $\Delta t \leq \frac{\Delta r^2}{2}$. Then, for any stable implementation of the scheme, Δt is bounded above by Δr^2. Then, the total truncation error is bounded above by Δr^2. This ensures that neither time nor space dominates the local truncation error, because both the spatial and temporal errors behave proportionally to Δr^2.

3.2.3 Implementation: 8-Minute Boil and Jenn Segal's Method

Figures 3.24, 3.25, and 3.26 are different angles of an 8-minute boil simulation using Method 2. Recall from the last section that this water temperature function should produce a medium boil. Observe in the figures that the outputs of this numerical method agree with the medium boil definition from section 1.1.

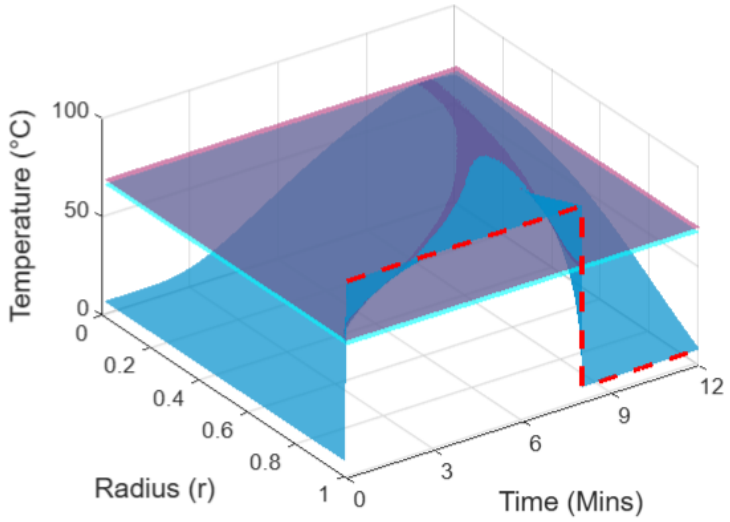

Figure 3.24: 8-Minute Boil, Method 2, $\Delta t = 0.1$, $\Delta r = 0.01$

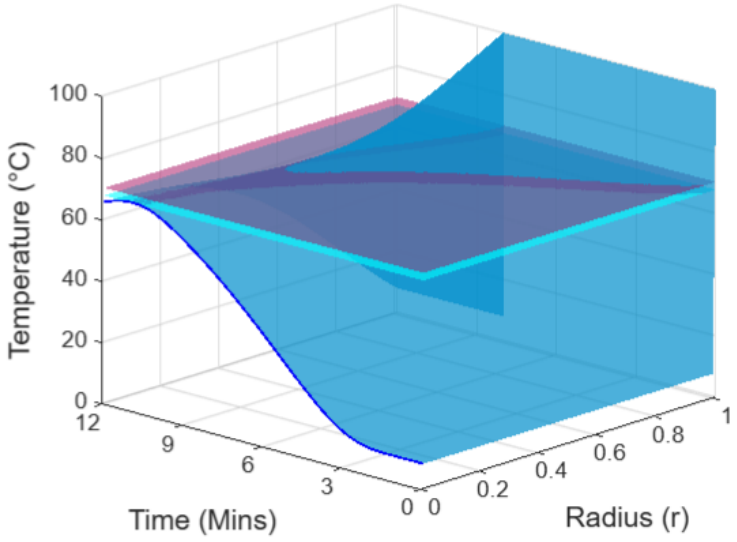

Figure 3.25: 8-Minute Boil, Method 2, Rotated, $\Delta t = 0.1$, $\Delta r = 0.01$

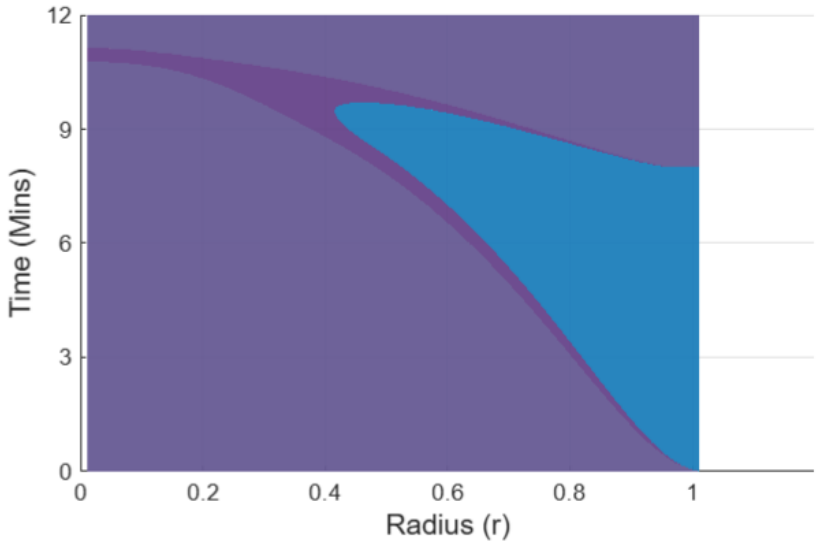

Figure 3.26: 8-Minute Boil, Method 2, Birdseye, $\Delta t = 0.1$, $\Delta r = 0.01$

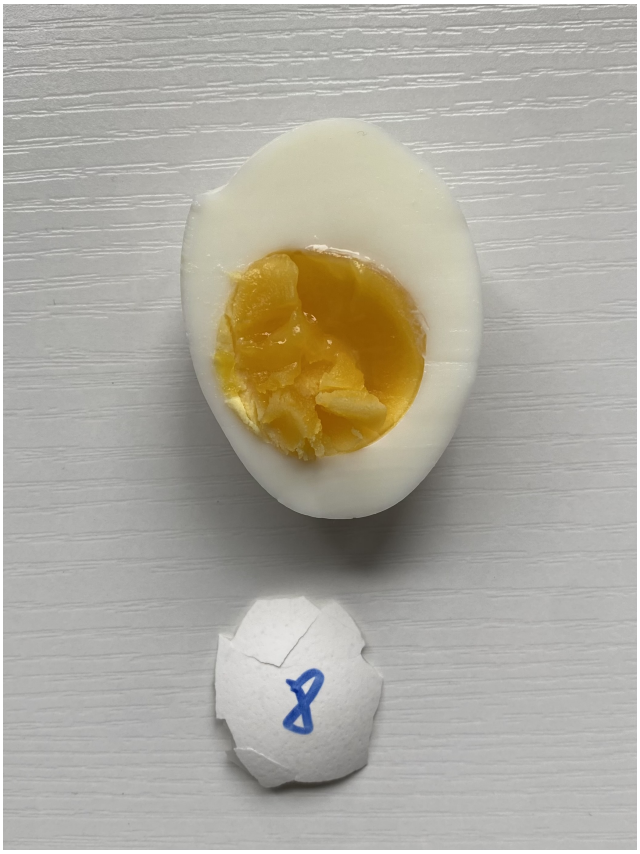

Figure 3.27: 8-Minute Boil, Real Egg

Jenn Segal's Method In this section we analyze the model's output for another common hard-boil method [7]. This method consists of putting fridge cold eggs in cold tap water in a saucepan and heating up the contents until a bubbling boil. During our experiment, the water reached this boil at approximately 9.1 minutes. Then, we turn off the stove and let the contents sit in the pot for 10 minutes, before running the egg under cold tap water at $8°C$ for 2 minutes. Finally, we fill the pot with room temperature water ($22°C$) and wait a couple minutes to ensure that the internal temperature of the egg has approximately reached equilibrium. Figure 3.28 is a graph of the water temperature that is described by the article. The derivatives for $0 \leq t \leq 9.1$ and $9.1 < t \leq 19.1$ were calculated using Newton's Law of Cooling, based on measurements conducted every 2 minutes.

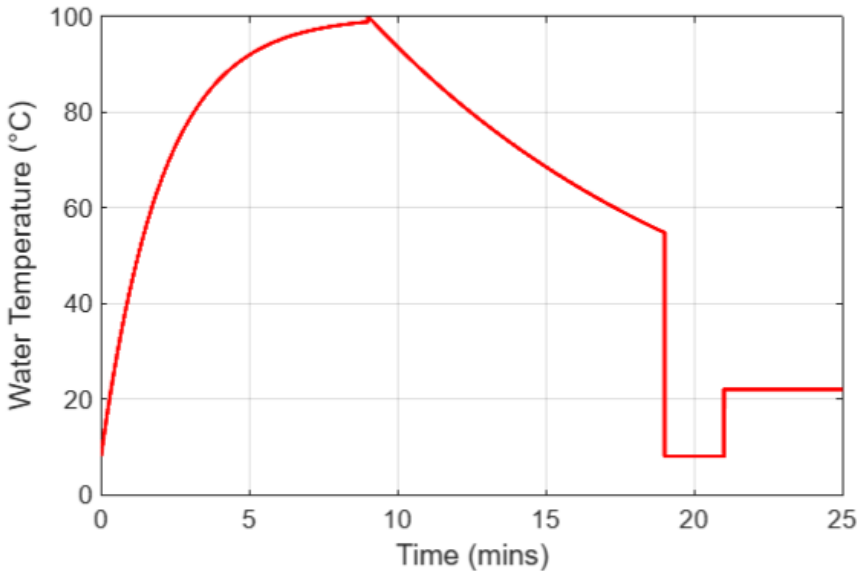

Figure 3.28: Jenn Segal's Water Temperature Function

We use this water temperature function and the numerical method described in this section with $\Delta t = 0.1$ and $\Delta r = 0.01$ to create Figures 3.29, 3.30, and 3.31. Notice that the figures quite clearly support the hypothesis that the egg will be hard boiled, as every part of the blue surface has surpassed the purple plane, which represents the solidification point of the yolk.

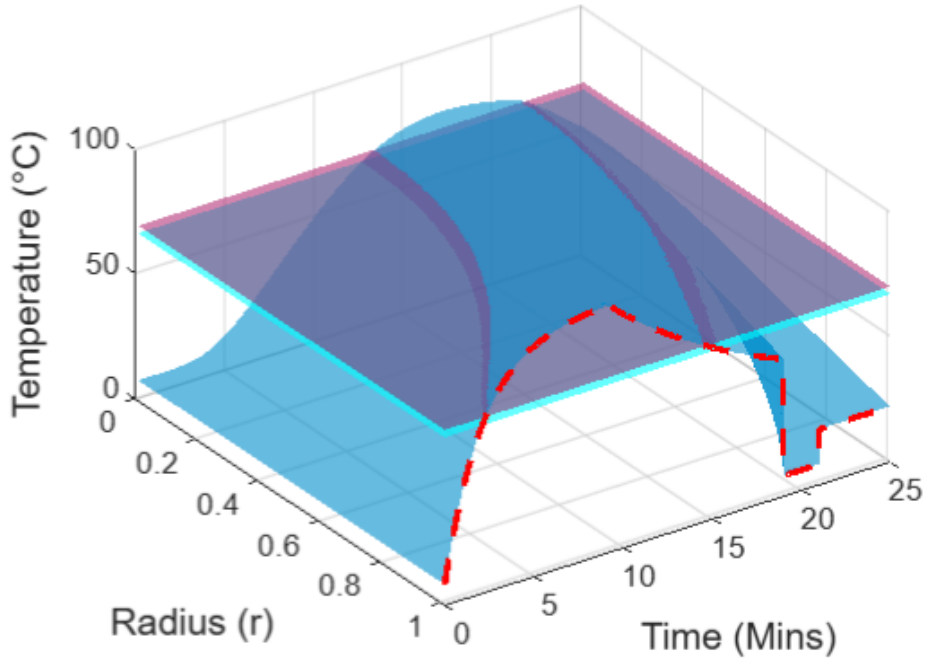

Figure 3.29: Jenn Segal, Method 2, $\Delta t = 0.1$, $\Delta r = 0.01$

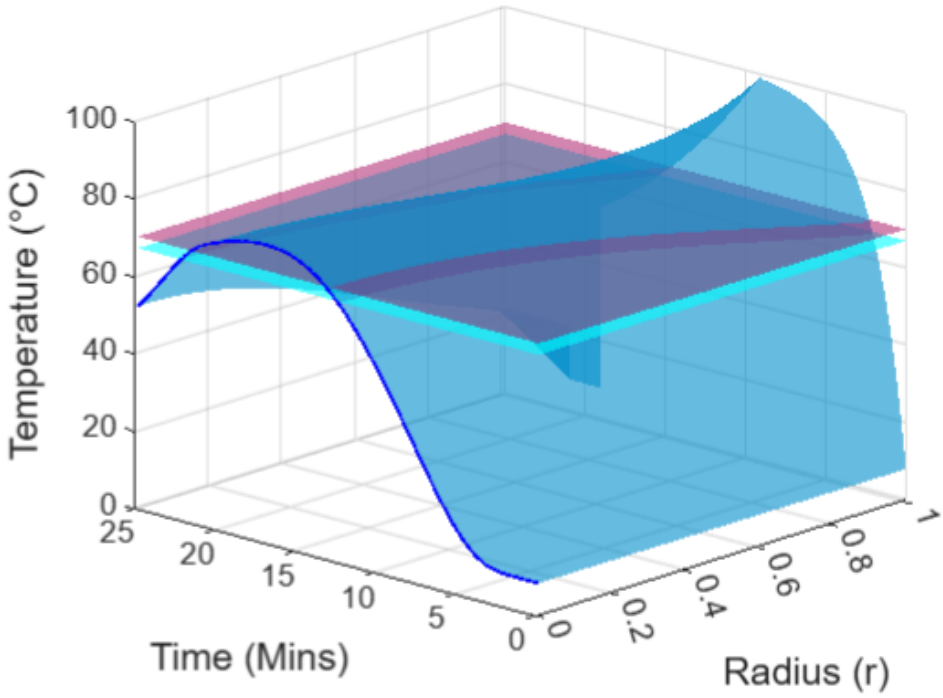

Figure 3.30: Jenn Segal, Method 2, Rotated, $\Delta t = 0.1$, $\Delta r = 0.01$

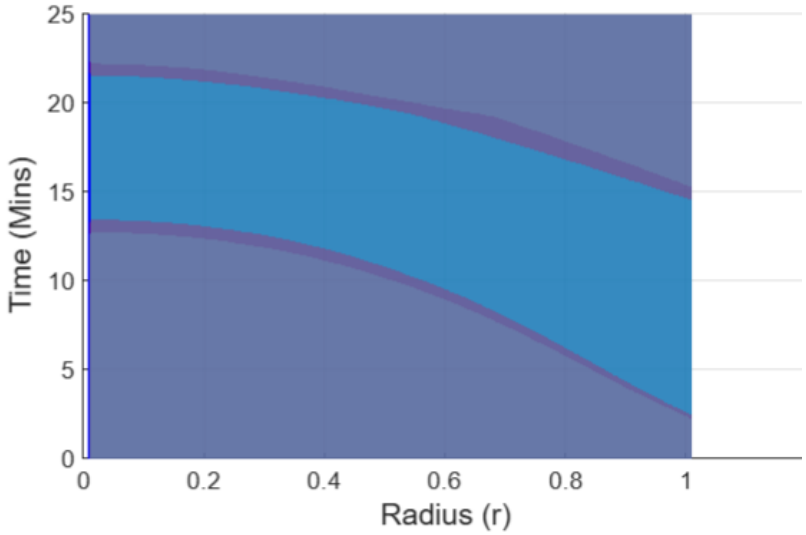

Figure 3.31: Jenn Segal, Method 2, Birdseye, $\Delta t = 0.1$, $\Delta r = 0.01$

Figure 3.32 is a fridge-cold egg cooked using this method. Observe that it is indeed hard boiled.

Figure 3.32: Jenn Segal, Real Egg

3.3 Comparison of Method 1 and Method 2

Method 1 and Method 2 are both consistent, so they approximate the same equation as Δt and Δr approach 0. In this section, we compare the surfaces generated with different methods and step sizes for a 10-minute boil followed by a 5-minute rest in tap water. Figures 3.33 and 3.34 visualize the surfaces generated using Δt = 0.2, and Δr = 0.04.

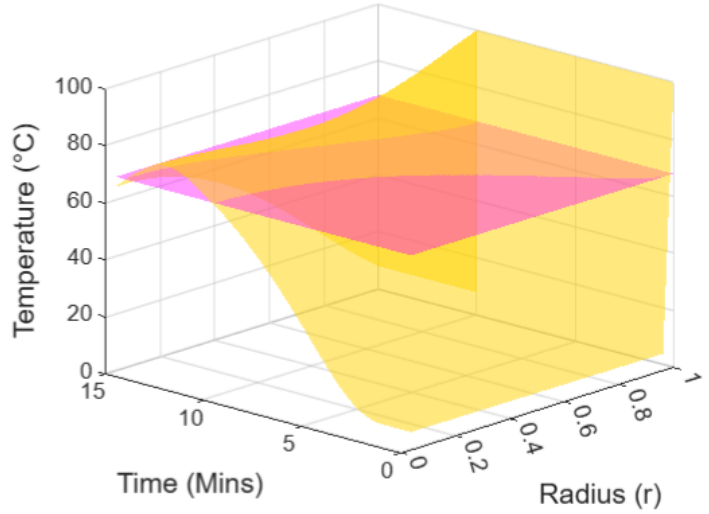

Figure 3.33: Method 1, Δt = 0.2, Δr = 0.04

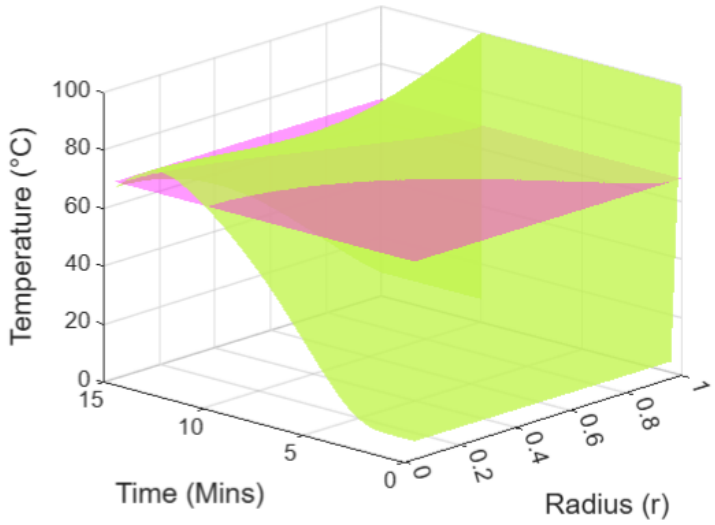

Figure 3.34: Method 2, Δt = 0.2, Δr = 0.04

Figures 3.35 and 3.36 visualize the same two surfaces from a birdseye view. Notice that Method 1 claims that the egg reaches a hard boil earlier than the output of Method 2, but they agree in their end result.

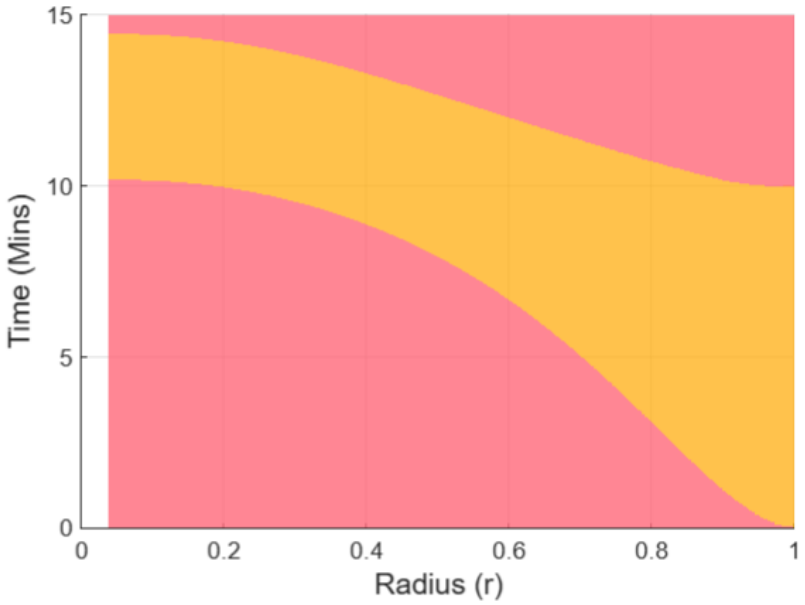

Figure 3.35: Method 1, $\Delta t = 0.2$, $\Delta r = 0.04$, Birdseye

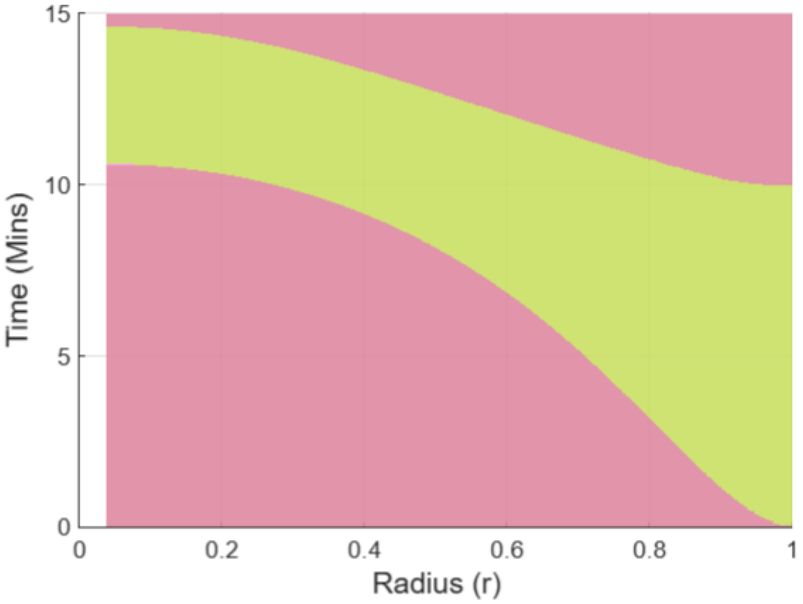

Figure 3.36: Method 2, $\Delta t = 0.2$, $\Delta r = 0.04$, Birdseye

Figure 3.37 plots the differences between the two solutions across the entire simulation. The maximum absolute value error between the two solutions is $2.846°C$.

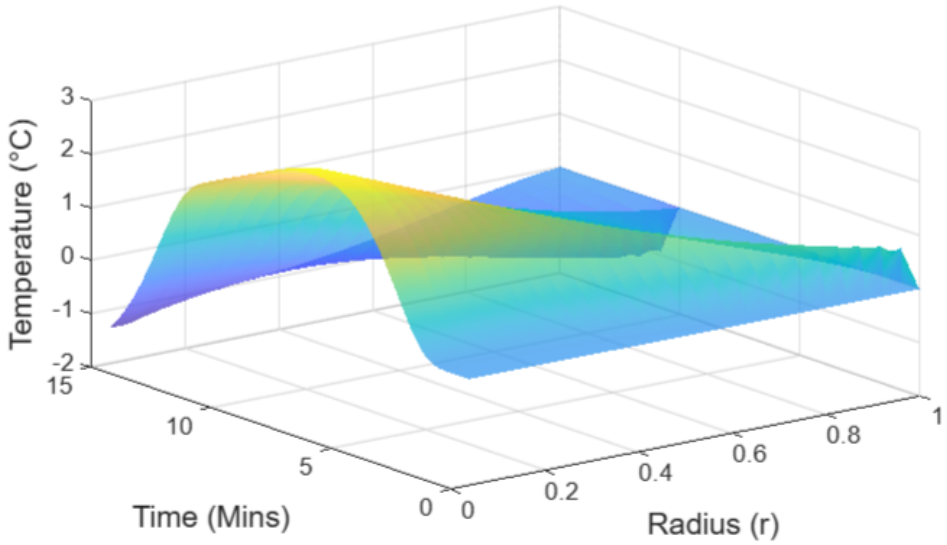

Figure 3.37: Difference between Method 1 and Method 2, $\Delta t = 0.2$, $\Delta r = 0.04$

Figure 3.38 plots the difference between solutions produced with $\Delta t = 0.1$ and $\Delta r = 0.02$. The maximum absolute value for these step sizes is $1.462°C$.

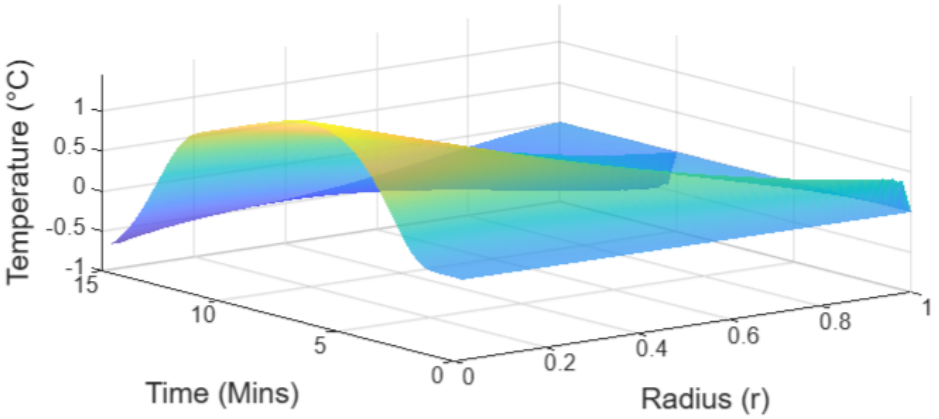

Figure 3.38: Difference between Method 1 and Method 2, $\Delta t = 0.1$, $\Delta r = 0.02$

Figure 3.39 plots the difference between solutions produced with $\Delta t = 0.05$ and $\Delta r = 0.01$. The maximum absolute value for these step sizes is $0.740°C$.

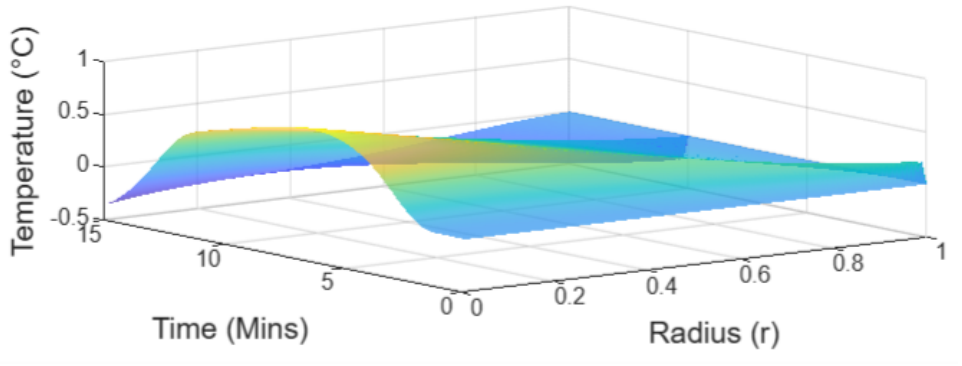

Figure 3.39: Difference between Method 1 and Method 2, $\Delta t = 0.05$, $\Delta r = 0.01$

Figure 3.40 plots the difference between solutions produced with $\Delta t = 0.025$ and $\Delta r = 0.005$. The maximum absolute value for these step sizes is $0.3725°C$.

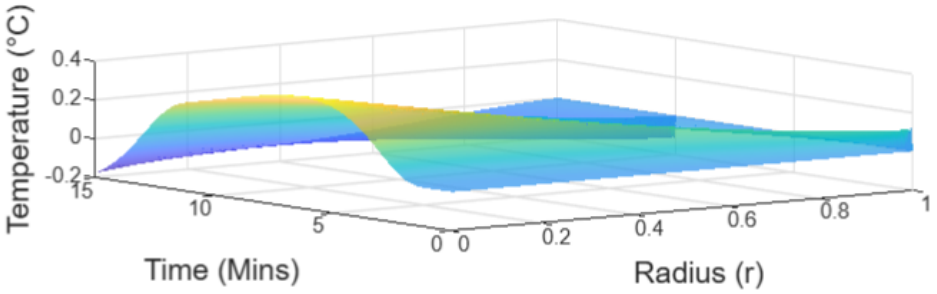

Figure 3.40: Difference between Method 1 and Method 2, $\Delta t = 0.025$, $\Delta r = 0.005$

Table 3.1 plots the maximum absolute value difference between the two methods against 4 different pairs of step sizes. Observe that as Δt and Δr are each divided by 2, the total error is also approximately divided by 2.

Δt	Δr	Maximum absolute value difference ($°C$)
0.2	0.04	2.826
0.1	0.02	1.462
0.05	0.01	0.740
0.025	0.005	0.3725

Table 3.1: Maximum Difference Between Methods 1 and 2 with Grid Refinement

Chapter 4

Generalization To Radius-Dependent Non-Homogeneous Thermal Diffusivity

Until this point, α was defined using a weighted average of α_{white} and α_{yolk}. In this chapter we part the sphere into its two distinct materials, and separate their density ρ, thermal conductivity κ, and specific heat c. This adjustment will render the model more accurate to a real egg.

Recall from Section 1.3 that the egg yolk has radius 1.53cm, and the whole egg has radius 2.255 cm. Then, solving for $\frac{r_{yolk}}{r_{egg}} = \frac{1.53}{2.255} = 0.678$ tells us that the threshold where the values transition should be located at $r = 0.678$.

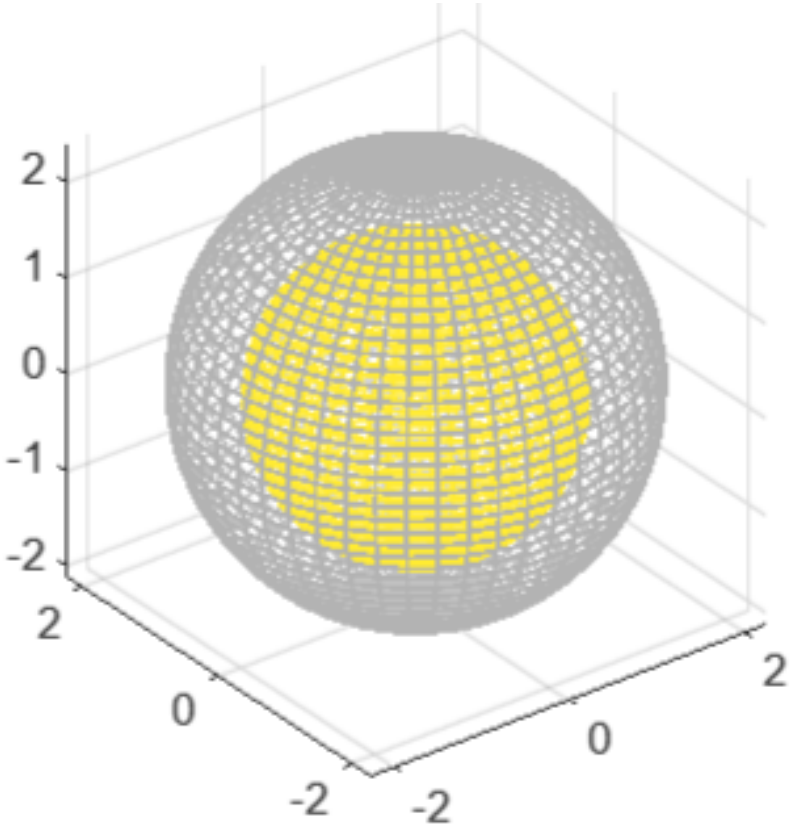

Figure 4.1: Egg Estimated by Sphere, Normalized

After constructing the threshold, we describe the behavior of each variable at $r = 0.678$. These functions are constant across each material and transition smoothly at the threshold. Additionally, we need $\alpha(r)$ and $\kappa(r)$ to be at least once differentiable everywhere, as we will use their derivatives in the next two schemes we introduce. With no motivation other than satisfying the two above conditions, we construct the following equations:

$$\alpha(r) = \alpha_{yolk} + \frac{\alpha_{white} - \alpha_{yolk}}{1 + e^{-200(r-0.678)}} = 2.576 \times 10^{-4} + \frac{3.94 \times 10^{-5}}{1 + e^{-200(r-0.678)}}$$

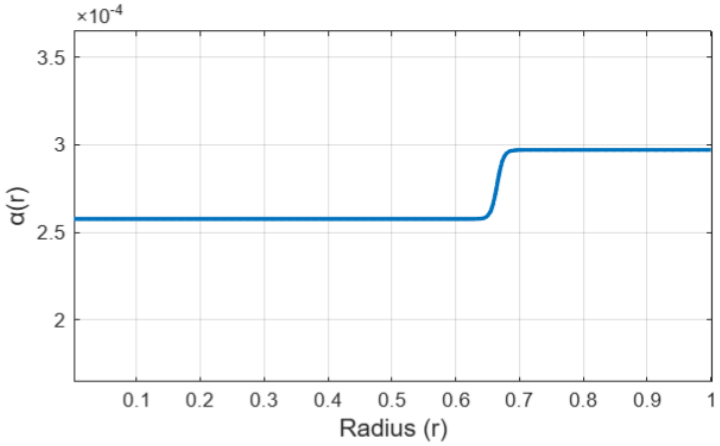

Figure 4.2: α as a Function of r

$$\kappa(r) = \kappa_{yolk} + \frac{\kappa_{white} - \kappa_{yolk}}{1 + e^{-200(r-0.678)}} = 0.39 + \frac{0.16}{1 + e^{-200(r-0.678)}}$$

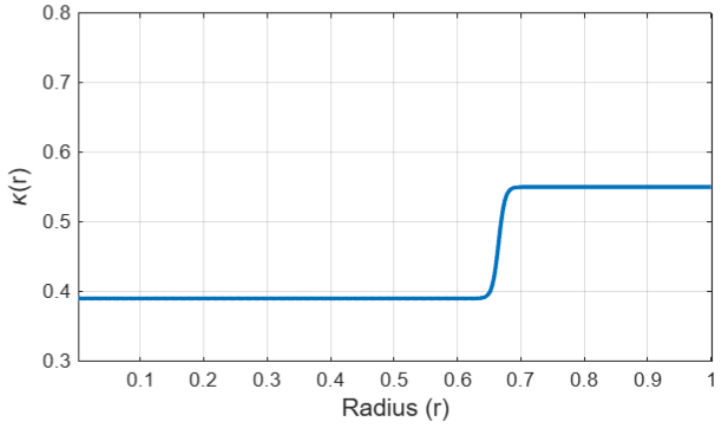

Figure 4.3: κ as a Function of r

$$c(r) = c_{yolk} + \frac{c_{white} - c_{yolk}}{1 + e^{-200(r-0.678)}} = 2.62 + \frac{0.932}{1 + e^{-200(r-0.678)}}$$

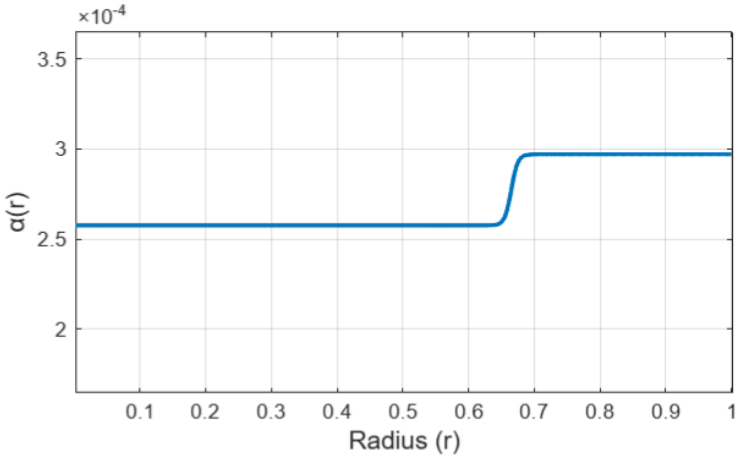

Figure 4.4: c as a Function of r

$$\rho(r) = \rho_{white} + \frac{\rho_{white} - \rho_{yolk}}{1 + e^{-200(r-0.678)}} = 1026.6 - \frac{107.7}{1 + e^{-200(r-0.678)}}$$

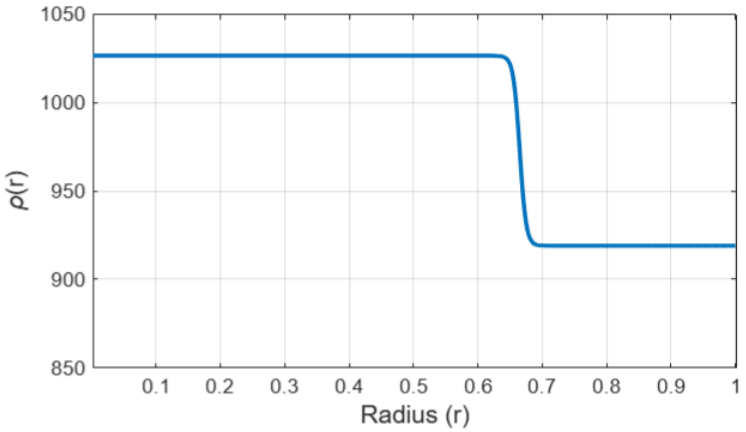

Figure 4.5: ρ as a Function of r

4.1 Heat Equation with Spatially Varying Conductivity

Recall from Section 1.3 that the general form of the heat equation is

$$\frac{\partial u}{\partial t} = \frac{1}{c\rho}\nabla \cdot \left(\kappa \nabla u\right)$$

Since κ is no longer constant across space, we need to calculate $\nabla \cdot \big(\kappa(r)\nabla u\big)$:

Let $\mathbf{F} = \kappa(r)\nabla u = \kappa(\sqrt{x^2+y^2+z^2})\dfrac{u_r}{r}\begin{bmatrix} x \\ y \\ z \end{bmatrix}$, and recall that $\nabla \cdot \mathbf{F} = \dfrac{\partial \mathbf{F}_x}{\partial x} + \dfrac{\partial \mathbf{F}_y}{\partial y} + \dfrac{\partial \mathbf{F}_x}{\partial z}$.

$$\frac{\partial \mathbf{F}_x}{\partial x} = \frac{\partial}{\partial x}\left(\kappa(r)\frac{x \cdot u_r}{r}\right) = \frac{\partial}{\partial x}\left(\kappa(r)\frac{u_r}{r}\right)\cdot x + \kappa(r)\frac{u_r}{r} = \frac{\left(\kappa(r)\dfrac{u_r}{r}\right)_r}{r}x^2 + \kappa(r)\frac{u_r}{r}$$

And similarly, $\dfrac{\partial \mathbf{F}_y}{\partial y} = \dfrac{\left(\kappa(r)\dfrac{u_r}{r}\right)_r}{r}y^2 + \kappa(r)\dfrac{u_r}{r}$ and $\dfrac{\partial \mathbf{F}_z}{\partial z} = \dfrac{\left(\kappa(r)\dfrac{u_r}{r}\right)_r}{r}z^2 + \kappa(r)\dfrac{u_r}{r}$.

Then, $\nabla \cdot \mathbf{F} = \dfrac{\left(\kappa(r)\dfrac{u_r}{r}\right)_r}{r}(x^2+y^2+z^2) + 3\kappa(r)\dfrac{u_r}{r} = \left(\kappa(r)\dfrac{u_r}{r}\right)_r \cdot r + 3\kappa(r)\dfrac{u_r}{r}$.

$$\left(\kappa(r)\frac{u_r}{r}\right)_r \cdot r + 3\kappa(r)\frac{u_r}{r} = \left(\left(\frac{-1}{r^2}\kappa(r) + \frac{1}{r}\frac{\partial \kappa}{\partial r}\right)\frac{\partial u}{\partial r} + \frac{\kappa(r)}{r}\frac{\partial^2 u}{\partial r^2}\right)\cdot r + 3\kappa(r)\frac{\partial u}{\partial r}$$

$$= \left(\frac{-1}{r^2}\kappa(r)\frac{\partial u}{\partial r} + \frac{1}{r}\frac{\partial \kappa}{\partial r}\frac{\partial u}{\partial r} + \frac{\kappa(r)}{r}\frac{\partial^2 u}{\partial r^2}\right)\cdot r + 3\kappa(r)\frac{\partial u}{\partial r}$$

$$= \frac{-\kappa(r)}{r}\frac{\partial u}{\partial r} + \frac{\partial \kappa}{\partial r}\frac{\partial u}{\partial r} + \kappa(r)\frac{\partial^2 u}{\partial r^2} + \frac{3\kappa(r)}{r}\frac{\partial u}{\partial r}$$

$$= \kappa\frac{\partial^2 u}{\partial r^2} + \frac{\partial \kappa}{\partial r}\frac{\partial u}{\partial r} + \frac{2\kappa}{r}\frac{\partial u}{\partial r}$$

This form of the heat equation generalizes the previous model for spatially varying thermal conductivity. Hence, we want to ensure that this approach is consistent with those of the previous chapter. Notice that assuming constant α implies the thermal diffusivity having no dependence on the radial coordinate. Then, let us set $\dfrac{\partial \kappa}{\partial r} = 0$ and simplify the equation to $\kappa\dfrac{\partial^2 u}{\partial r^2} + \dfrac{2\kappa}{r}\dfrac{\partial u}{\partial r}$. Then,

$$\frac{\partial u}{\partial t} = \frac{1}{c\rho}\left(\kappa\frac{\partial^2 u}{\partial r^2} + \frac{2\kappa}{r}\frac{\partial u}{\partial r}\right) = \alpha\left(\frac{\partial^2 u}{\partial r^2} + \frac{2}{r}\frac{\partial u}{\partial r}\right)$$

Note that this equation is consistent with the radially symmetric Laplacian from Section 1.5 and the homogeneous models from Chapter 3.

4.2 Method 3

We want to approximate a solution to the following equation:

$$\frac{\partial u}{\partial t} = \frac{1}{c(r)\rho(r)}\nabla\cdot(\kappa(r)\nabla u) = \frac{1}{c(r)\rho(r)}\left(\frac{2\kappa}{r}\frac{\partial u}{\partial r} + \frac{\partial \kappa}{\partial r}\frac{\partial u}{\partial r} + \kappa\frac{\partial^2 u}{\partial r^2}\right)$$

$$= \frac{2\alpha(r)}{r}\frac{\partial u}{\partial r} + \alpha(r)\frac{\partial^2 u}{\partial r^2} + \frac{1}{c(r)\rho(r)}\frac{\partial \kappa}{\partial r}\frac{\partial u}{\partial r}$$

We use the same differencing methods from Section 2.2 and approximate the derivatives in the following way:

$$\frac{\partial u}{\partial t} = \frac{u_j^{n+1} - u_j^n}{\Delta t}$$

$$\frac{2\alpha}{r}\frac{\partial u}{\partial r} = \frac{2\alpha_j}{r_j}\left(\frac{u_{j+1} - u_j}{\Delta r}\right)$$

$$\alpha\frac{\partial^2 u}{\partial r^2} = \alpha_j\left(\frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta r^2}\right)$$

$$\frac{\partial \kappa}{\partial r}\frac{\partial u}{\partial r} = \left(\frac{\kappa_{j+1} - \kappa_j}{\Delta r}\right)\left(\frac{u_{j+1} - u_j}{\Delta r}\right)$$

Substituting the approximations into the heat equation:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{2\alpha_j}{r_j}\left(\frac{u_{j+1} - u_j}{\Delta r}\right) + \alpha_j\left(\frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta r^2}\right) + \frac{1}{c_j\rho_j}\left(\frac{\kappa_{j+1} - \kappa_j}{\Delta r}\right)\left(\frac{u_{j+1} - u_j}{\Delta r}\right) \tag{4.1}$$

Solving for u_j^{n+1} gives the following numerical scheme:

$$u_j^{n+1} = u_j^n + \Delta t\left(\frac{2\alpha_j}{r_j}\left(\frac{u_{j+1} - u_j}{\Delta r}\right) + \alpha_j\left(\frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta r^2}\right) + \frac{1}{c_j\rho_j}\left(\frac{\kappa_{j+1} - \kappa_j}{\Delta r}\right)\left(\frac{u_{j+1} - u_j}{\Delta r}\right)\right)$$

An implementation of this scheme can be found in the appendix. The boundary conditions are handled the same way as the previous two methods.

4.2.1 Implementations: 8-Minute Boil and Double Cooling Method

Figures 4.6, 4.7, and 4.8 visualize the scheme's output for an 8-minute boil. Observe that the shape of the surface suggest the egg should be medium boiled.

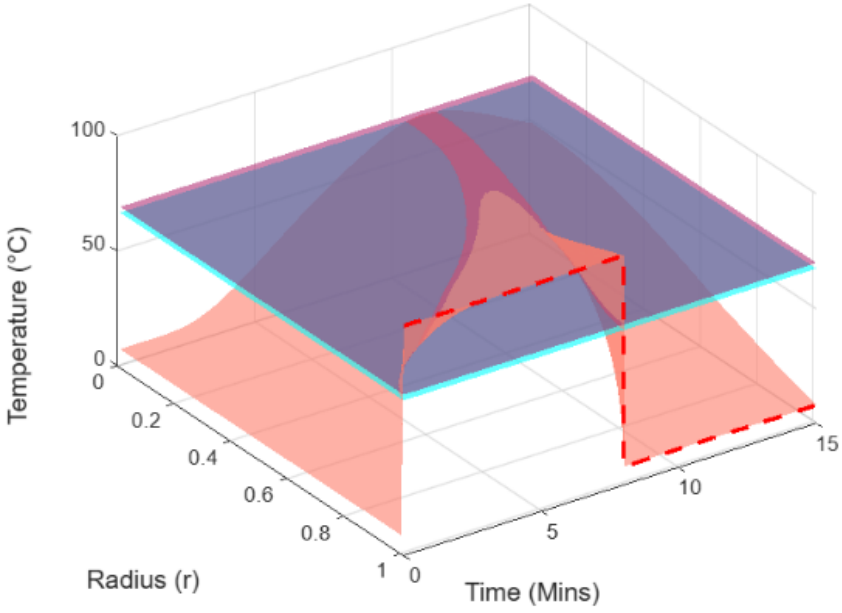

Figure 4.6: 8-Minute Boil, Method 3, $\Delta t = 0.1$, $\Delta r = 0.01$

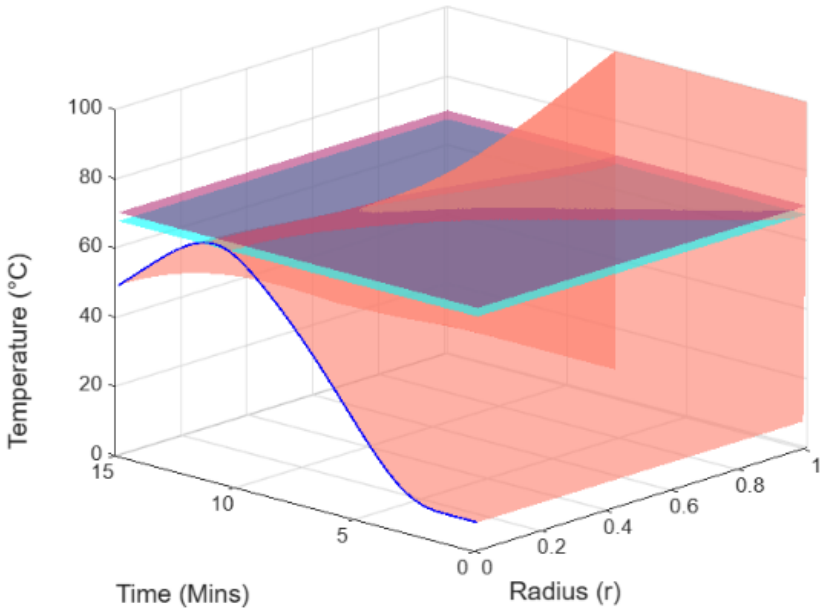

Figure 4.7: 8-Minute Boil, Method 3, Rotated, $\Delta t = 0.1$, $\Delta r = 0.01$

Double Cooling Mehod

In this section, we construct and test a method of cooking eggs with no stove. We boil 1 L of water in a kettle, and place the egg in a heat-safe bowl that is big enough to submerge it completely. We then periodically measure the temperature of the water near the eggshell and use Newton's law of cooling to build the function plotted in Figure 4.8.

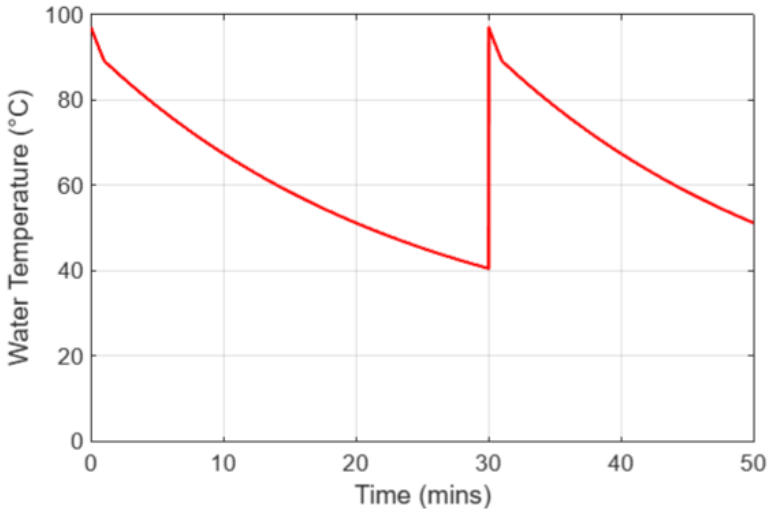

Figure 4.8: Double Cooling, Water Function

Since the water begins cooling as soon as it is poured out of the kettle, the initial temperature of the water next to the eggshell was measured at $97°C$. Additionally, the water came in contact with fridge cold eggs as soon as it was poured, so it cooled very rapidly in the first minute of the experiment. After the initial minute, the cooling became more gradual, and we used Newton's law of cooling to approximate its slope based on measurements that were 2 minutes apart. Then, we let the egg sit in the bowl for 30 minutes, after which the water was measured at $40.3°C$. At 30 minutes, we pour out the warm water and repeat the process. The motivation behind doing it twice will become apparent in Figure 4.9. During the second cycle, we let the egg sit in the water for 20 minutes. At the end of 50 minutes, we pour out the warm water and peel the egg.

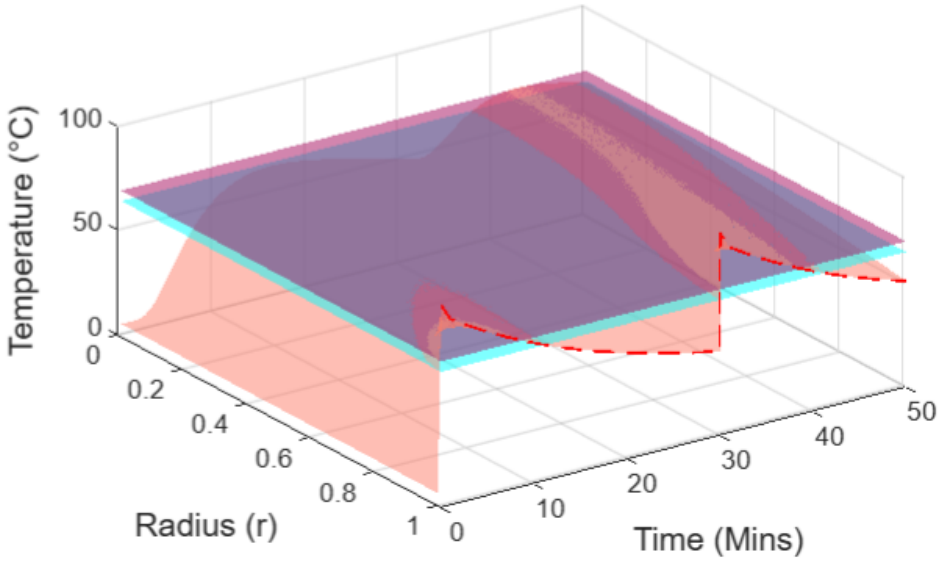

Figure 4.9: Double Cooling, Method 3, $\Delta t = 0.1$, $\Delta r = 0.01$

Observe in Figure 4.9 that the first boil does not provide the egg with enough total thermal energy to coagulate it fully. However, at the end of the second cooling, the egg reaches a hard boil. Figure 4.10 rotates the same surface to show how the central temperature evolves through time.

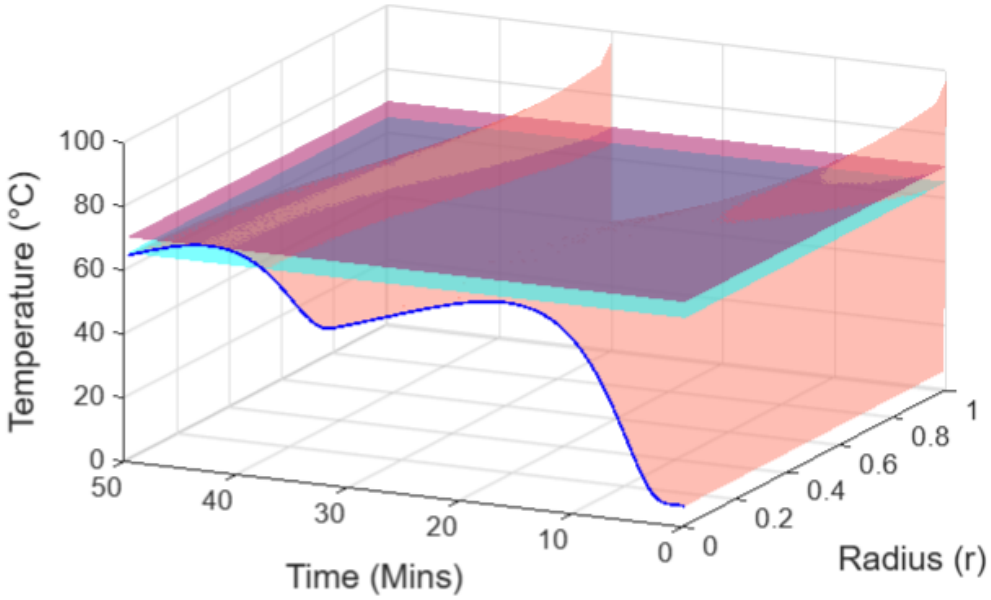

Figure 4.10: Double Cooling, Method 3, Rotated, $\Delta t = 0.1$, $\Delta r = 0.01$

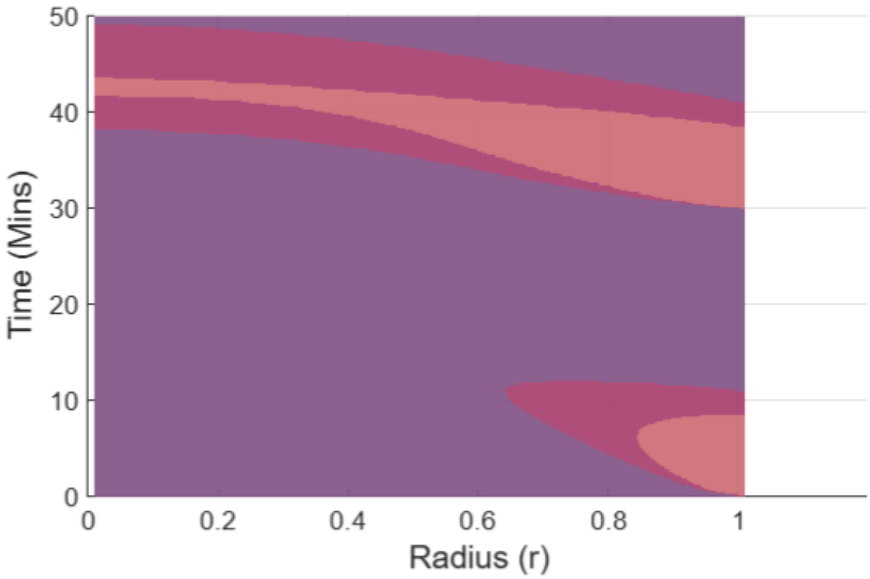

Figure 4.11: Double Cooling, Method 3, Birdseye, $\Delta t = 0.1$, $\Delta r = 0.01$

Figure 4.12: Double Cooling, Real Egg

Figure 4.13: Double Cooling, Real Egg

Figure 4.14: Double Cooling, Real Egg, Performed Live at Thesis Defense on 18 April 2025

4.3 Method 4

Recall the second order approximation to $\frac{\partial u}{\partial r}$ from Section 3.2:

$$\frac{\partial u}{\partial r} = \frac{u_{j+1} - u_{j-1}}{2\Delta r}$$

We approximate the two following derivatives using this second order method:

$$\frac{2\alpha}{r}\frac{\partial u}{\partial r} = \frac{2\alpha_j}{r_j}\left(\frac{u_{j+1} - u_{j-1}}{2\Delta r}\right) = \frac{\alpha_j}{r_j}\left(\frac{u_{j+1} - u_{j-1}}{\Delta r}\right)$$

$$\frac{\partial \kappa}{\partial r}\frac{\partial u}{\partial r} = \left(\frac{\kappa_{j+1} - \kappa_{j-1}}{2\Delta r}\right)\left(\frac{u_{j+1} - u_{j-1}}{2\Delta r}\right)$$

Additionally, recall that the following approximation is already second order:

$$\alpha\frac{\partial^2 u}{\partial r^2} = \alpha_j\left(\frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta r^2}\right)$$

Finally, recall from Section 3.1 that $\frac{\partial u}{\partial t}$ is estimated first order using $\frac{u_j^{n+1} - u_j^n}{\Delta t}$.

Our scheme solves the equation:

$$\frac{\partial u}{\partial t} = \frac{2\alpha}{r}\frac{\partial u}{\partial r} + \alpha\frac{\partial^2 u}{\partial r^2} + \frac{1}{c\rho}\frac{\partial \kappa}{\partial r}\frac{\partial u}{\partial r}$$

Substituting the approximations:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{2\alpha_j}{r_j}\left(\frac{u_{j+1} - u_{j-1}}{2\Delta r}\right) + \alpha_j\left(\frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta r^2}\right) + \frac{1}{c_j\rho_j}\left(\frac{\kappa_{j+1} - \kappa_{j-1}}{2\Delta r}\right)\left(\frac{u_{j+1} - u_{j-1}}{2\Delta r}\right)$$

Solving for u_j^{n+1} gives the following numerical scheme:

$$u_j^{n+1} = u_j^n + \Delta t\left(\frac{2\alpha_j}{r_j}\left(\frac{u_{j+1} - u_{j-1}}{2\Delta r}\right) + \alpha_j\left(\frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta r^2}\right) + \frac{1}{c_j\rho_j}\left(\frac{\kappa_{j+1} - \kappa_{j-1}}{2\Delta r}\right)\left(\frac{u_{j+1} - u_{j-1}}{2\Delta r}\right)\right)$$

An implementation of this scheme can be found in the appendix. The boundary conditions are handled the same way as the previous three methods.

4.3.1 Implementation: 8-Minute Boil and Periodic Cooking

Figure 4.15 visualizes the model's output for an 8-minute boil. The pink surface in these figures has surpassed the cyan plane located at $68°C$, and only a small portion of the radius has surpassed the purple plane located at the solidification temperature ($70°C$). These results agree with the definition of a medium boil.

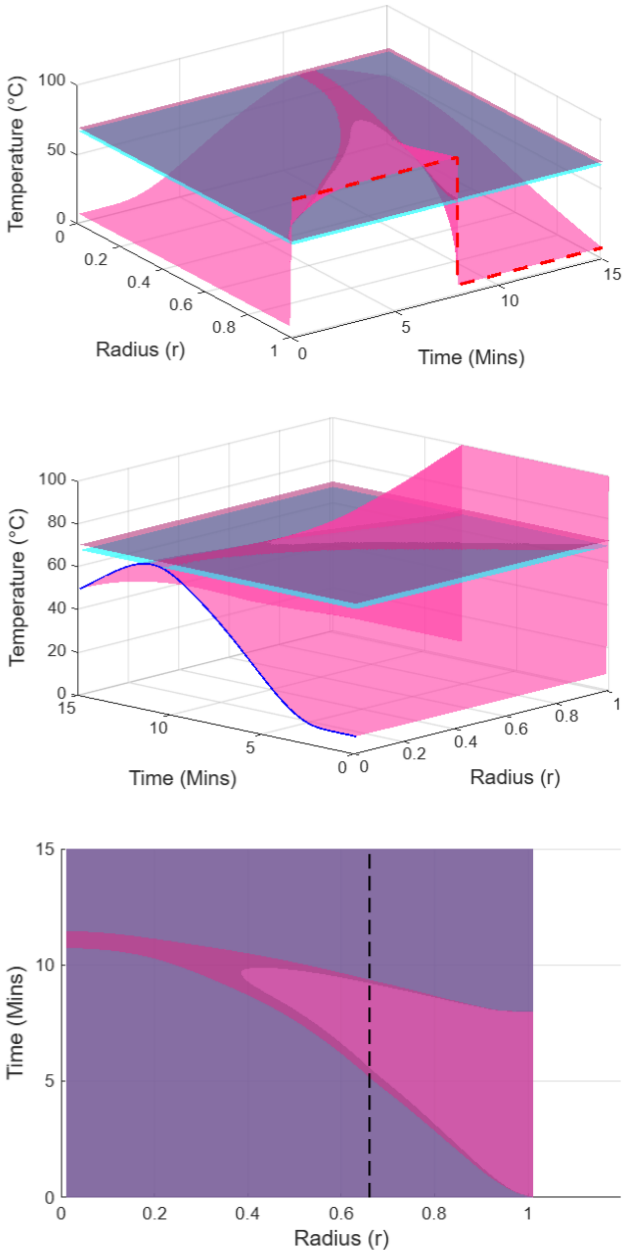

Figure 4.15: 8-Minute Boil, Method 4, $\Delta t = 0.1$, $\Delta r = 0.01$

Periodic Cooking

An article called Periodic Cooking of Eggs [4] was published in Communications Engineering in February 2025. In this section, we explain what the method entails, and we use Method 4 to numerically test whether our results agree with those from the article.

Periodic cooking claims to achieve the perfect boiled egg by submerging it in boiling water (100°C) for two minutes, then in lukewarm water (30°C) for two minutes, and alternating for a total of 16 cycles over 32 minutes. This approach "balances a velvety yolk and a soft, solid white" by ensuring that both components reach their optimal cooking temperatures, while also not staying above them for an extended period of time. The method also enhances the egg's nutritional profile by "increasing its polyphenol content." While it is definitely more time consuming than any other traditional method of making boiled eggs, it "yields superior taste and texture, making it a valuable technique for culinary enthusiasts." In order to conceptualize the mathematical construction of our experiment, Figure 4.16 is an abbreviated version of the water temperature function, which includes only one out of the 16 cycles.

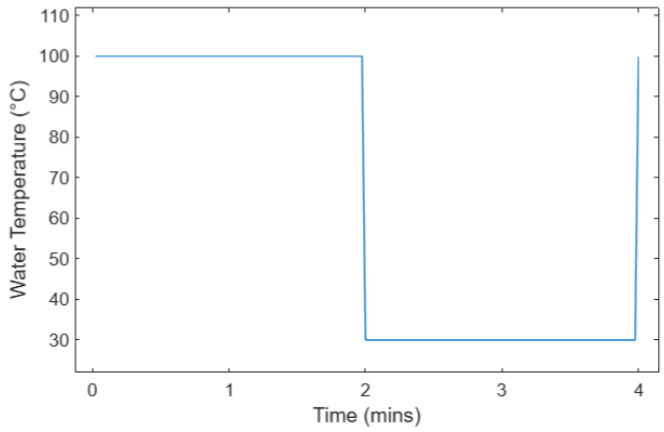

Figure 4.16: One Cycle of Water Temperature for Periodic Cooking Method

Figure 4.17 simulates periodic cooking using Method 4 and Δt = 0.1 and Δr = 0.01. The line at r = 1 plots all 16 cycles of the water temperature equation from Figure 4.16. The cyan plane is located at the coagulation temperature of the yolk ($65°C$) and the purple plane is located at the solidification temperature of the yolk

$(70°C)$.

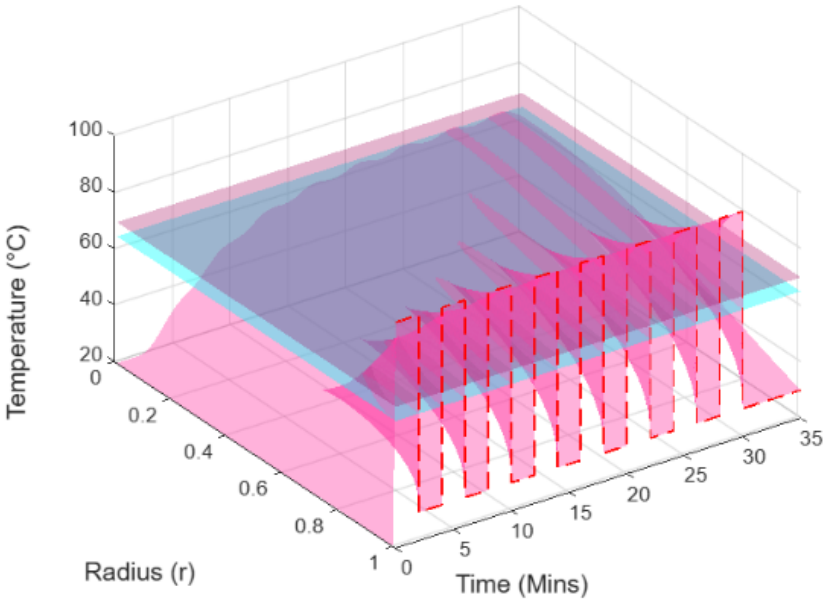

Figure 4.17: Periodic Cooking, Method 4, $\Delta t = 0.1$, $\Delta r = 0.01$

Figure 4.18 is a rotated version of the same surface. The blue line at $r = 0$ shows that the center of the egg yolk oscillates in temperature, and the simulation ends soon after the yolk has reached coagulation $(65°C)$.

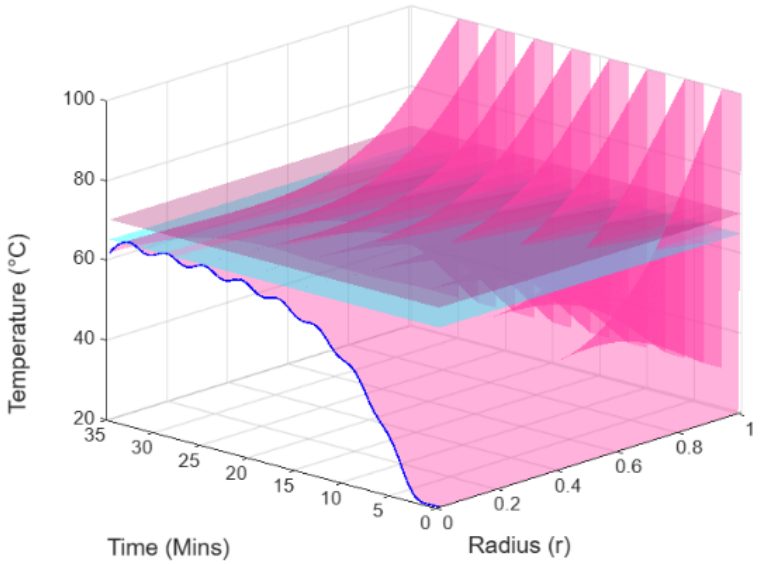

Figure 4.18: Periodic Cooking, Method 4, Rotated, $\Delta t = 0.1$, $\Delta r = 0.01$

Figure 4.19 visualizes the same surface from a birdseye view, as this perspective makes it easier to observe cookedness. Notice that the egg white periodically crosses its cooking point ($65°C$, cyan plane), and each repetition of these cycles influences a larger portion of the egg.

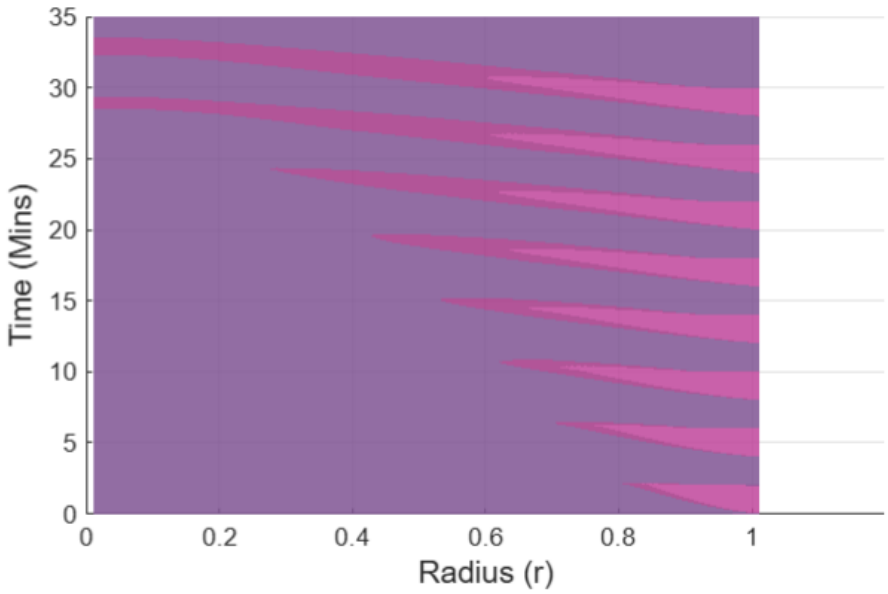

Figure 4.19: Periodic Cooking, Method 4, Birdseye, $\Delta t = 0.1$, $\Delta r = 0.01$

This image supports the claimed advantages of "periodic cooking." Since the external temperature alternates periodically, any part of the egg that reaches its cooking point begins to cool down very soon after. According to the original source, this makes the whole egg less likely to overcook and have a rubbery texture, and chemically makes it more nutritious than other traditional methods.

Figure 4.20: Periodic Cooking, Real Egg

Bibliography

[1] American Egg Board, *Coagulation/thickening*, 2023. Accessed: 2025-04-11.

[2] ——, *Egg size conversion chart and tips*, 2023. Accessed: 2025-04-11.

[3] Jane Selia dos Reis Coimbra, Ana L. Gabas, Luis A. Minim, Edwin E. Garcia Rojas, Vânia R. N. Telis, and Javier Telis-Romero, *Density, heat capacity and thermal conductivity of liquid egg products*, Journal of Food Engineering, 74 (2006), pp. 186–190.

[4] Emilia Di Lorenzo, Francesca Romano, Lidia Ciriaco, Nunzia Iaccarino, Luana Izzo, Antonio Randazzo, Pellegrino Musto, and Ernesto Di Maio, *Periodic cooking of eggs*, Communications Engineering, 4 (2025).

[5] PlanetMath Contributors, *Derivation of the laplacian from rectangular to spherical coordinates*. https://planetmath.org/DerivationOfTheLaplacianFromRectangularToSphericalCoordinates, 2013. Accessed: 2025-03-22.

[6] R. R. Rosales, MIT Department of Mathematics, *Lecture notes for 18.300: Principles of applied mathematics*. https://math.mit.edu/classes/18.300/Notes/Notes_vNSA.pdf, 2025. Accessed: 2025-05-01.

[7] Jennifer Segal, *How to make hard-boiled eggs*, 2024. Accessed: 2025-04-11.

[8] Stanford University, *Heat equation*. https://web.stanford.edu/class/math220b/handouts/HEATEQN.pdf, 2003. Lecture notes for Math 220B, accessed 2025-04-11.

Appendix A

Code for Numerical Method Implementations

A.1 Method 1

```
function U_time = method1(dt, dr, u_0, max_time_min, water_temp_func, alpha_value)
    % method1 Solves the heat equation in a sphere using
    % Euler's method, use with a customizable water temperature function
    %
    % Inputs:
    %   dt             - Time step size
    %   dr             - Space step size (radial step)
    %   u_0            - Initial egg temperature (uniform within egg, scalar)
    %   max_time_min   - Maximum time for simulation (in minutes)
    %   water_temp_func - Function handle for the water temperature equation (f(t))
    %
    % Output:
    %   U_time         - Matrix of temperatures (rows: time, columns: spatial points)

    % Parameters
    alpha = alpha_value; % Thermal diffusivity
```

```
18      R = 1;              % Radius of the sphere
19      T_max = floor((max_time_min * 60) / dt); % Convert max time to
            seconds and determine steps
20      N = floor(R / dr) + 1; % Number of spatial points

22      % initialization
23      r = linspace(0, R, N);
24      u = ones(1, N) * u_0;
25      U_time = zeros(T_max, N);



28      % Time iteration using Euler's method
29      for t = 1:T_max

31      u_new = u;

33      for i = 2:N-1
34          u_new(i) = u(i) + alpha * dt * ...
35                    ( ((u(i+1) - 2*u(i) + u(i-1)) / dr^2) + ( (2 / r(i))
                          * (u(i+1) - u(i)) ) / dr);
36      end

38      u_new(1) = u_new(2);
39      u_new(end) = water_temp_func(t*dt);

41      u = u_new;

43      U_time(t, :) = u;
44  end
```

A.2 Method 2

```
function U_time = method2(dt, dr, u_0, max_time_min, water_temp_func, alpha_value)
    % method2 Solves the heat equation in a sphere using explicit
    % finite differences. Use with a custom water function!
    %
    % Inputs:
    %   dt             - Time step size
    %   dr             - Space step size (radial step)
    %   u_0            - Initial temperature (scalar)
    %   max_time_min   - Maximum time for simulation (in minutes)
    %   water_temp_func - Function handle for the water temperature equation (f(t))
    %   alpha_value    - Thermal diffusivity
    %
    % Output:
    %   U_time         - Matrix of temperatures (rows: time, columns: spatial points)

    % Parameters
    alpha = alpha_value; % Thermal diffusivity
    R = 1;               % Radius of the sphere
    T_max = floor((max_time_min * 60) / dt); % Convert max time to seconds and determine steps
    N = floor(R / dr) + 1; % Number of spatial points
```

```
23  % Stability check (explicit method requires dt <= dr^2 / (2 *
        alpha))
24  if dt > dr^2 / (2 * alpha)
25      error('Time step dt is too large for stability. Choose dt <= dr
            ^2 / (2 * alpha).');
26  end

28  % Initialization
29  r = linspace(0, R, N);
30  u = ones(1, N) * u_0;
31  U_time = zeros(T_max, N);

33  % Time iteration using explicit finite differences
34  for t = 1:T_max
35      u_new = u;

37      % Update for interior points
38      for i = 2:N-1
39          laplacian = (u(i-1) - 2*u(i) + u(i+1)) / dr^2;
40          radial_term = (1 / r(i)) * ((u(i+1) - u(i-1))/dr);
41          u_new(i) = u(i) + dt * alpha * (laplacian + radial_term);
42      end

44      % Apply boundary conditions
45      u_new(1) = u_new(2);      % Neumann condition at r = 0
46      u_new(end) = water_temp_func(t * dt); % Dirichlet condition at
            r = R

48      % Update solution
```

```
        u = u_new;
        U_time(t, :) = u;
    end
end
```

A.3 Method 3

```
function U_time = method3(dt, dr, u_0, max_time_min, water_temp_func,
    alpha_func, kappa_func, c_func, rho_func)
    % method3 Solves the heat equation with spatially varying
        coefficients
    % using an explicit Euler scheme in a radially symmetric sphere.
    %
    % Inputs:
    %   dt             - Time step size
    %   dr             - Spatial step size
    %   u_0            - Initial temperature (scalar)
    %   max_time_min   - Simulation time in minutes
    %   water_temp_func - Function handle for water temperature f(t)
    %   alpha_func     - Function handle for thermal diffusivity (r)
    %   kappa_func     - Function handle for thermal conductivity (r)
    %   c_func         - Function handle for specific heat c(r)
    %   rho_func       - Function handle for density (r)
    %
    % Output:
    %   U_time         - Temperature matrix [time x radius]


    % Setup
    R = 1;
```

```
T_max = floor((max_time_min * 60) / dt);
N = floor(R / dr) + 1;

r = linspace(0, R, N);
alpha = arrayfun(alpha_func, r);
kappa = arrayfun(kappa_func, r);
c = arrayfun(c_func, r);
rho = arrayfun(rho_func, r);

u = ones(1, N) * u_0;
U_time = zeros(T_max, N);

for t = 1:T_max
    u_new = u;

    for i = 2:N-1
        ri = r(i);
        alpha_i = alpha(i);
        ci = c(i);
        rhoi = rho(i);
        kappai = kappa(i);
        kappa_ip1 = kappa(i+1);

        % Numerical scheme (from the image)
        term1 = (2 * alpha_i / ri) * ((u(i+1) - u(i)) / dr);
        term2 = alpha_i * ((u(i+1) - 2*u(i) + u(i-1)) / dr^2);
        term3 = (1 / (ci * rhoi)) * ((kappa_ip1 - kappai) / dr) *
            ((u(i+1) - u(i)) / dr);
```

```
49            u_new(i) = u(i) + dt * (term1 + term2 + term3);
50        end

52        % Boundary conditions
53        u_new(1) = u_new(2); % Neumann at r = 0
54        u_new(end) = water_temp_func(t * dt); % Dirichlet at r = R

56        u = u_new;
57        U_time(t, :) = u;
58    end
59 end
```

A.4 Method 4

```
1   function U_time = method4(dt, dr, u_0, max_time_min,
        water_temp_func, alpha_func, kappa_func, c_func, rho_func)
2   % method4 solves the spherical heat equation with variable
3   % coefficients using an explicit finite difference method.
4   %
5   % Inputs:
6   %   dt             - Time step size
7   %   dr             - Spatial step size
8   %   u_0            - Initial temperature (scalar)
9   %   max_time_min   - Simulation time in minutes
10  %   water_temp_func - Function handle for water temperature f(t)
11  %   alpha_func     - Function handle for (r)
12  %   kappa_func     - Function handle for (r)
13  %   c_func         - Function handle for c(r)
14  %   rho_func       - Function handle for (r)
```

```
15      %
16      % Output:
17      %   U_time         - Temperature matrix [time x radius]
18
19      % Setup grid and time
20      R = 1;
21      T_max = floor((max_time_min * 60) / dt);
22      N = floor(R / dr) + 1;
23
24      r = linspace(0, R, N);
25
26      % Evaluate spatially-varying coefficients
27      alpha = arrayfun(alpha_func, r);
28      kappa = arrayfun(kappa_func, r);
29      c = arrayfun(c_func, r);
30      rho = arrayfun(rho_func, r);
31
32      % Initialize temperature profile
33      u = ones(1, N) * u_0;
34      U_time = zeros(T_max, N);
35
36      % Time stepping loop
37      for t = 1:T_max
38          u_new = u;
39
40          for j = 2:N-1
41              rj = r(j);
42              alpha_j = alpha(j);
43              kappa_j = kappa(j);
```

```
            kappa_jp1 = kappa(j+1);
            cj = c(j);
            rhoj = rho(j);

            % Finite difference approximations
            dudr_centered = (u(j+1) - u(j-1)) / (2 * dr);
            d2udr2 = (u(j+1) - 2*u(j) + u(j-1)) / (dr^2);
            dkappadr = (kappa_jp1 - kappa(j-1)) / (2 * dr);

            % Numerical scheme terms
            term1 = (2 * alpha_j / rj) * dudr_centered;
            term2 = alpha_j * d2udr2;
            term3 = (1 / (cj * rhoj)) * dkappadr * dudr_centered;

            u_new(j) = u(j) + dt * (term1 + term2 + term3);
        end

        % Apply boundary conditions
        u_new(1) = u_new(2); % Neumann at r=0
        u_new(end) = water_temp_func(t * dt); % Dirichlet at r=R

        % Update
        u = u_new;
        U_time(t, :) = u;
    end
end
```