APPM 2360 Project 1 Write-up

Authors: Carson Gano, Gabriel Ohnstad, Nghi Tran

Introduction

In 2023 AFRLP Inc. won the contract for the new 10 billion dollar Meta Compound in Boulder, CO, enclosing multiple data centers, cubicles, and Mark Zuckerberg's -*REDACTED*-. Our team of AFRLP Engineers has been tasked with the very important task of ensuring the correct temperature ranges. We will need to evaluate our models for heat affects order to maintain the 71-85$^\circ F$ temperature range necessary for -*REDACTED*- according to Pet-Smart's Lizard Ownership Manual.

We will need to account for the affects of the ambient temperature outside $A(t)$, the data centers and -*REDACTED*- guests $H(t)$, as well as our provisioned HVAC systems $Q(t)$. We can model these affects in a differential equation, $\frac{dT}{dt} = A(t) + H(t) + Q(t)$, and monitor the affects they have on temperature to evaluate the functionality of our system.

As our calculations will involve complicated differential equations that need to be graphed, we are attempting to develop a numerical system to properly graph these equations. Our chosen software is MATLAB.

Simplified Model

Initially, we want to start with a simple model for our system that we can build off of as we add more complexity. To do this, we will model the most measurable and consistent parameter affecting the temperature, the outside environment $A(t)$.

Methods

From Newton's Cooling Law, we know that the change in temperature of a system out of equilibrium is the difference in temperature multiplied by a coefficient of heat exchange k. As such, we can represent the outside temperature as $M(t)$ and define our simplified system as $A(t) = \frac{dT}{dt} = k(M(t) - T(t))$, the equation for our system ignoring $H(t)$ and $Q(t)$ for now. Additionally, for simplicity, we will generalize the environment $M(t)$ to be a constant, and thus we get the non-homogeneous linear equation: *(see Appendix A4 - part a.)*

$$\frac{dT}{dt} = k(M - T(t))$$

Before solving, it is useful to analyze if this function has a solution using **Picard's Theorem**, which states that if both $\frac{dT}{dt}$ and $\frac{d^2T}{dt^2}$ are continuous, then the solution can be evaluated and is unique. As the only t dependent parameter, $T(t)$ always exists and is in the numerator, the solution must be unique. *(see Appendix A2)*

We solve this equation using the integrating factor method

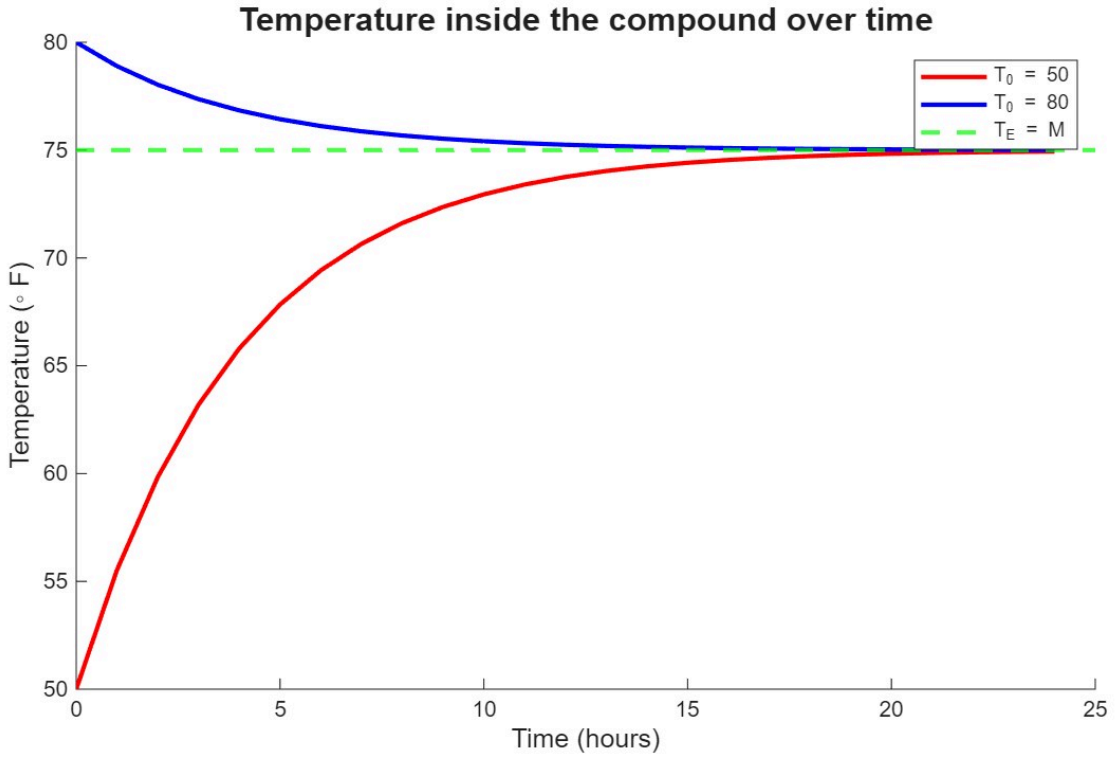


Figure A.1: Temperature inside compound without considering other elements

and get $T(t) = M + (T_0 - M)e^{kt_0}e^{-kt}$ *(see Appendix A3 - A4)*

We also examined how the heating properties of our compound affect its internal temperature to determine which material would be best to build the compound and ensure its functionality.

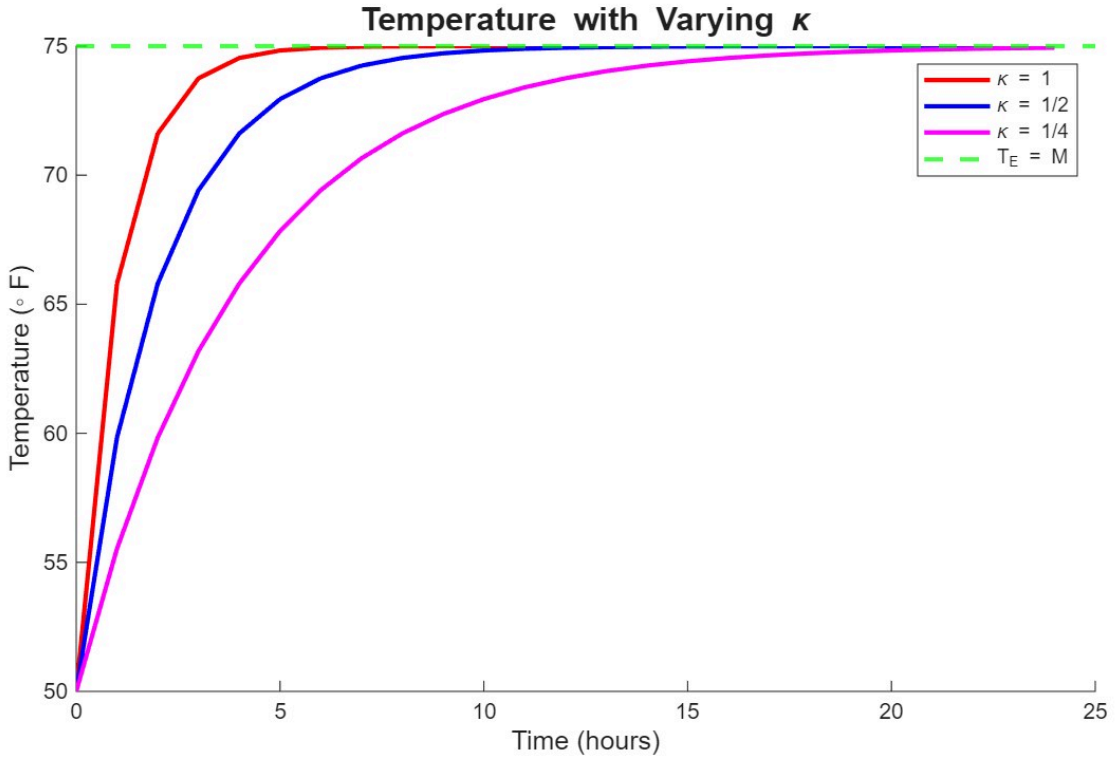


Figure A.2: How different construction materials affect the compound's temperature changes

From *Figure A.2*, we can see that while changing κ does not affect the equilibrium solution, the rate of change of temperature is changed: a bigger κ implies a greater rate of change, and vice versa. The quality of the compound's insulation could change κ, with quality insulation resulting in a slower heat loss (smaller κ), and poor insulation resulting in a sharper decrease in temperature (larger κ).

Numerical Calculator

Methods:

We are going to use the Runge-Kutta fourth order estimation to create our numeric approximation for graphing. This method operates on the principle of estimating the **slope between two values** of the differential equation and then moving along that slope to approximate the rough movement of the equation.

This is an approximation, but operates off the principles that guide the definition of a derivative:

$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

in that the estimate approaches the actual equation as our intervals (h) approach 0. We can cut down h by simply calculating more steps over the same interval, getting higher accuracy at the cost of compute power. Think of it like if you stretched out a human costume over the rough shape of a human and then had a lizard person climb into it, it would roughly approximate a human with an imperceptible error. Our goal is to balance speed with accuracy, tuning h to meet our criteria and comparing our answers to a solvable differential equation to measure accuracy

Technical Approach

First, we need a test differential equation in MATLAB to test our code against. We are using $\frac{dT}{dt} = 0.25(75 - T), T(0) = 50$ as this sample equation. We are given the rk4.m function to work with as well, which takes in $t_0, t_f, n, y_0, f(t)$ as variables. As such, the only function we really need to introduce is $\frac{dT}{dt}$. After completion, we get a full graph (*Figure B.1*) of the estimation of the equation.

After this, we need to compare the error to the actual solution of the differential equation. Luckily our original equation is very easily differentiable, so we can compare a graph of it directly. Plotting $T = 75 - 25e^{-\frac{t}{4}}$ alongside our graph, we get:

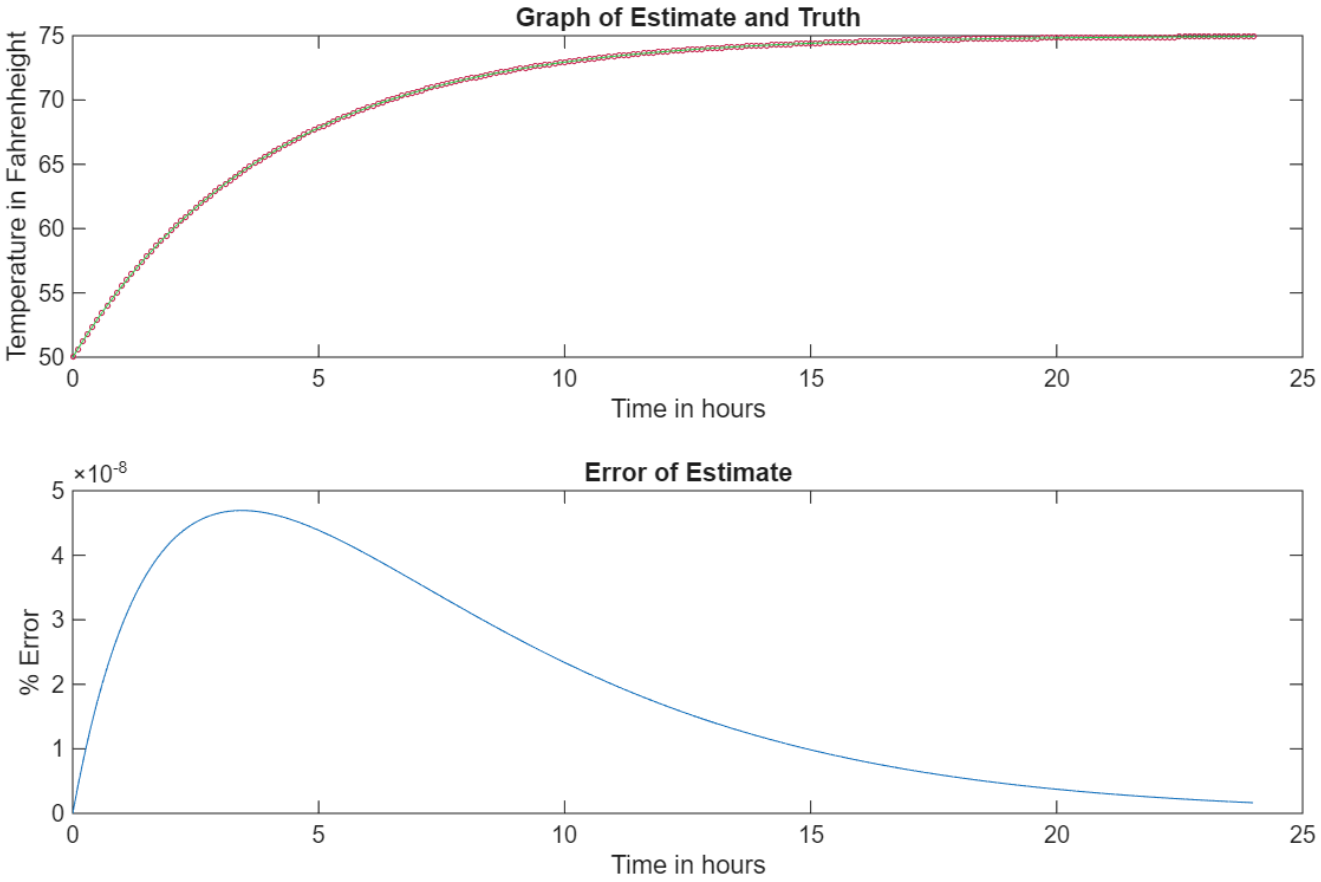


Figure B.1: Testing Numerical Calculator's accuracy based on Estimates and Errors

At this scale the values look almost entirely identical. Looking at the % deviation of the estimate from the actual value gives a closer view of the actual error, where the error is on the order of 10^{-8}%, almost entirely negligible.

Results

The rk4.m estimation method has proven incredibly proficient, and certainly suitable for usage in our calculations. We were surprised with the shape of the error graph, since we assumed that the error would increase over time, but it seems to spike only initially before climbing back down, seemingly correcting itself. Quite impressive!

Improving our model

Accounting for Daily Weather Cycles

In part A, our model simplified $M(t)$ to be a constant value. In actuality, ambient temperature varies by season and time of day, and we need to model this fluctuation to get a proper estimation.

Now that our initial function has been solved and our numerical calculator is ready, we can transform $M(t)$ into a sinusoidal function to illustrate daily temperature ranges, as well as testing summer and winter average temperatures to ensure we don't end up with a frozen or overheated Zucc.

Methods:

Using the measured temperature fluctuations of a given day in Boulder, we will assume a sinusoidal oscillation with a peak to valley amplitude of 24 degrees, and a variable starting temperature M_0, being 75 in the summer and 35 in the winter. We can then plug in this variable temperature $T_{out}(t)$ to our initial equation $\frac{dT}{dt}(t) = 0.25(T_{out} - T)$ and solve to get our complimentary internal temperature.

We would also like to find our maximum values of our graphs to the minute precision, so that in the case of Colorado randomly forgetting which season it is and dropping to -20, we can rush Zuckerberg to his -*Redacted*- before the coldest time of the day to prevent him from turning into a pop-sickle.

Technical Approach:

For our main model, $T_{out}(t) = M_0 - 12\cos\left(\frac{\pi(t-f)}{12}\right), T(0) = 65$. In our summer system, plugging in $M_0 = 75$, and solving the differential equation (proof in appendix $FigC_1$), we get the equation $T(t) = 75 - 14.3e^{-t/4} - \frac{36}{9+\pi^2}(3\cos(\theta) + \pi\cos(\theta))$, where $\theta = \frac{\pi(t-5)}{12}$. From plotting these two equations in MATLAB, we get:

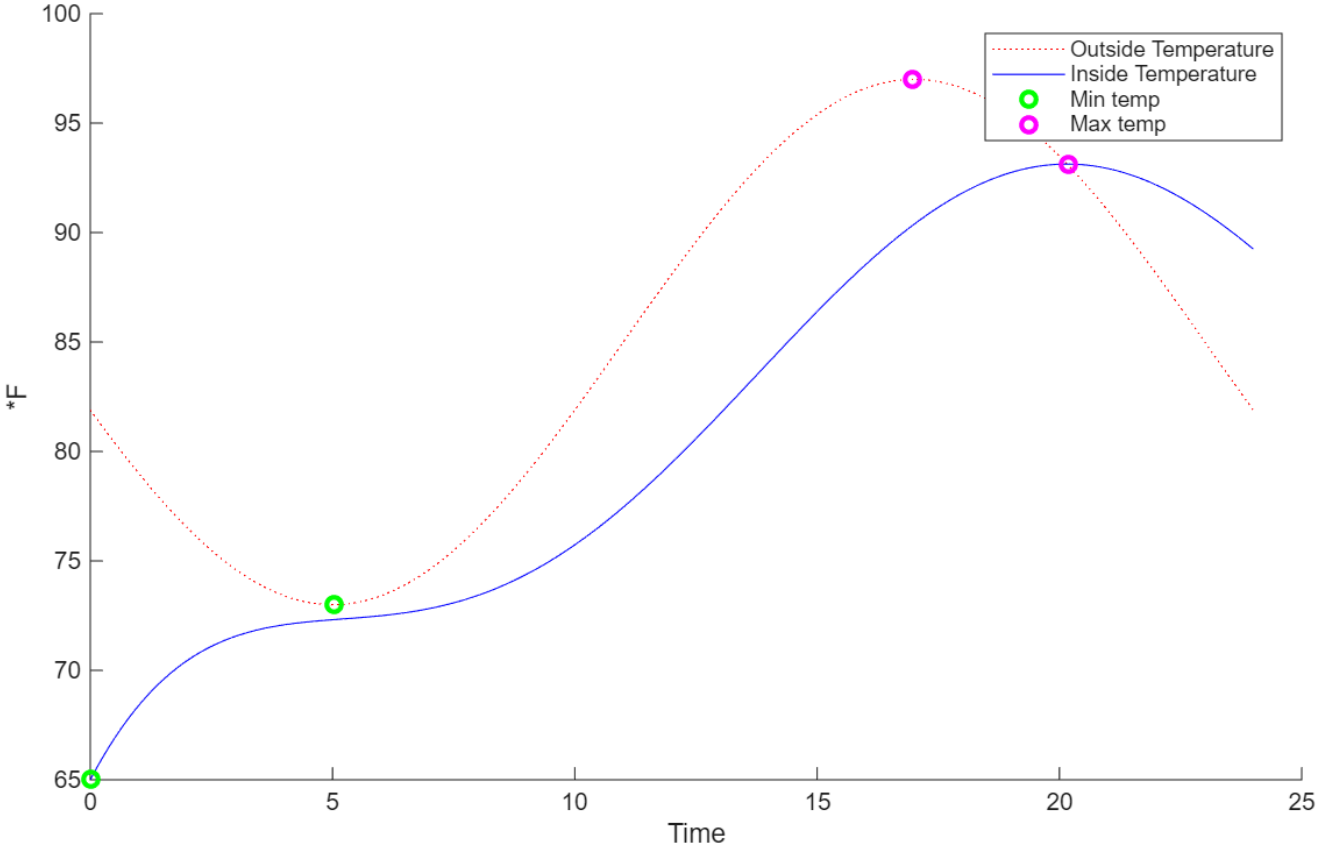
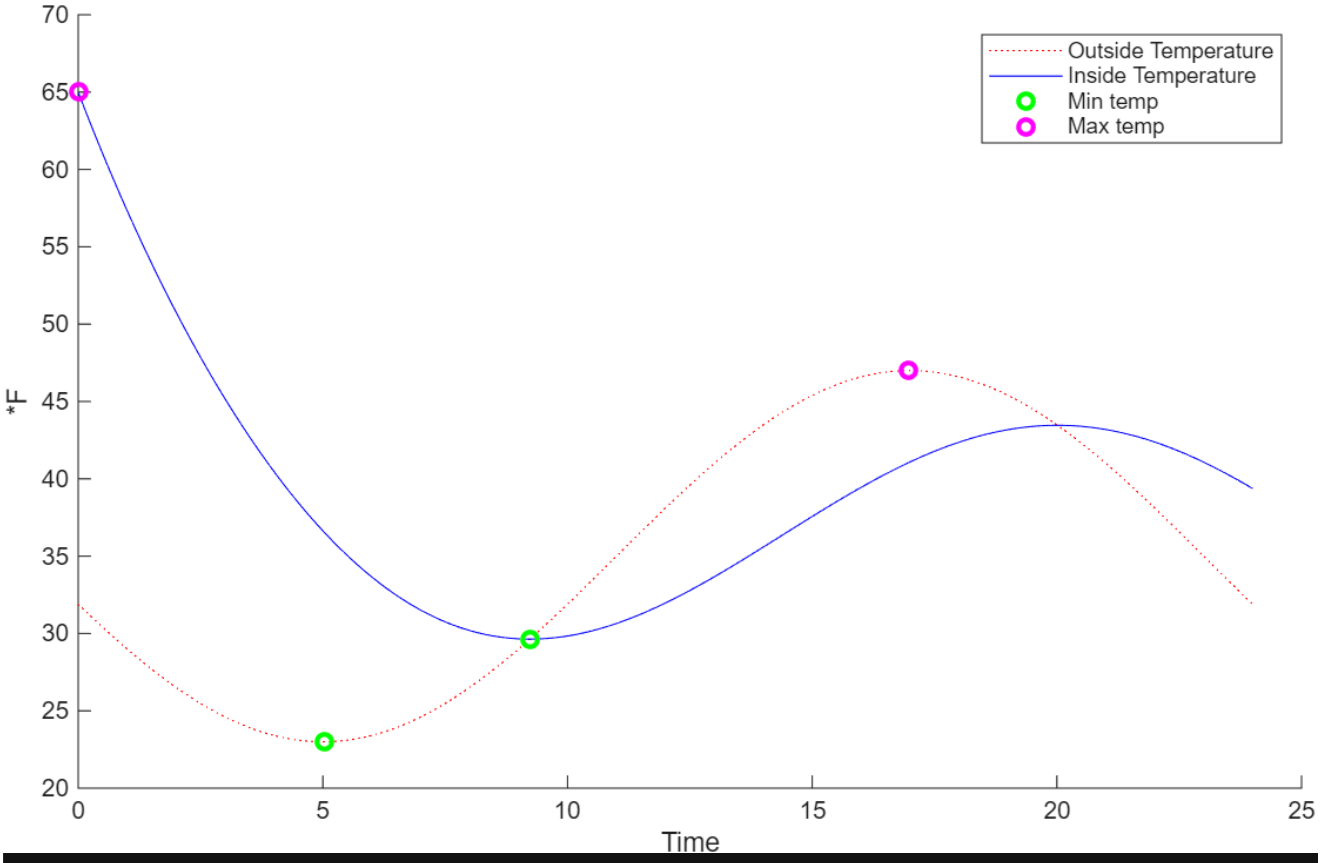


Figure C.1: Comparison of external & internal temperatures of the compound within a 24-hour time frame in the summer

By plotting over a range of $24hours * 60min$, we get a precision of minutes, and can output our max and min values from the range with the function, getting: *Inside temperature: Minimums and Maxs of 64.56°F and 83.19°F at 6:58 and 20:08 Outside temperature: Minimums and Maxs of 63.00°F and 87.00°F at 5:00 and 16:59*

We want to also get a representation of this data in the winter, so we resolve with $M_0 = 35$. We modified the matlab code to be flexible, taking in any M_0 as a perameter of the equation,

and then inputted C into our $T(t)$ equation (work in $FigC_2$ appendix) to get




```
Inside temperature: Minimums and Maxs of 29.64 and 65.00 at  9:13 and  0:00
Outside temperature: Minimums and Maxs of 23.00 and 47.00 at  5:00 and 16:59
```

Figure C.2: Comparison of external & internal temperatures of the compound within a 24-hour time frame in the winter

Results

We can observe that the inside temperature is slightly out of phase with the out, in addition to having a lower amplitude. Its minimums and maximums also occur at intersections with the outside temperature, as $T_{out} - T_{in} = 0 = \frac{dT}{dt}$.

In both cases, $F(t)$ trends towards $M(t)$ initially, before reaching a sort of phase equilibrium with it after the first few hours. You can see this especially well with the graph extending up to a week

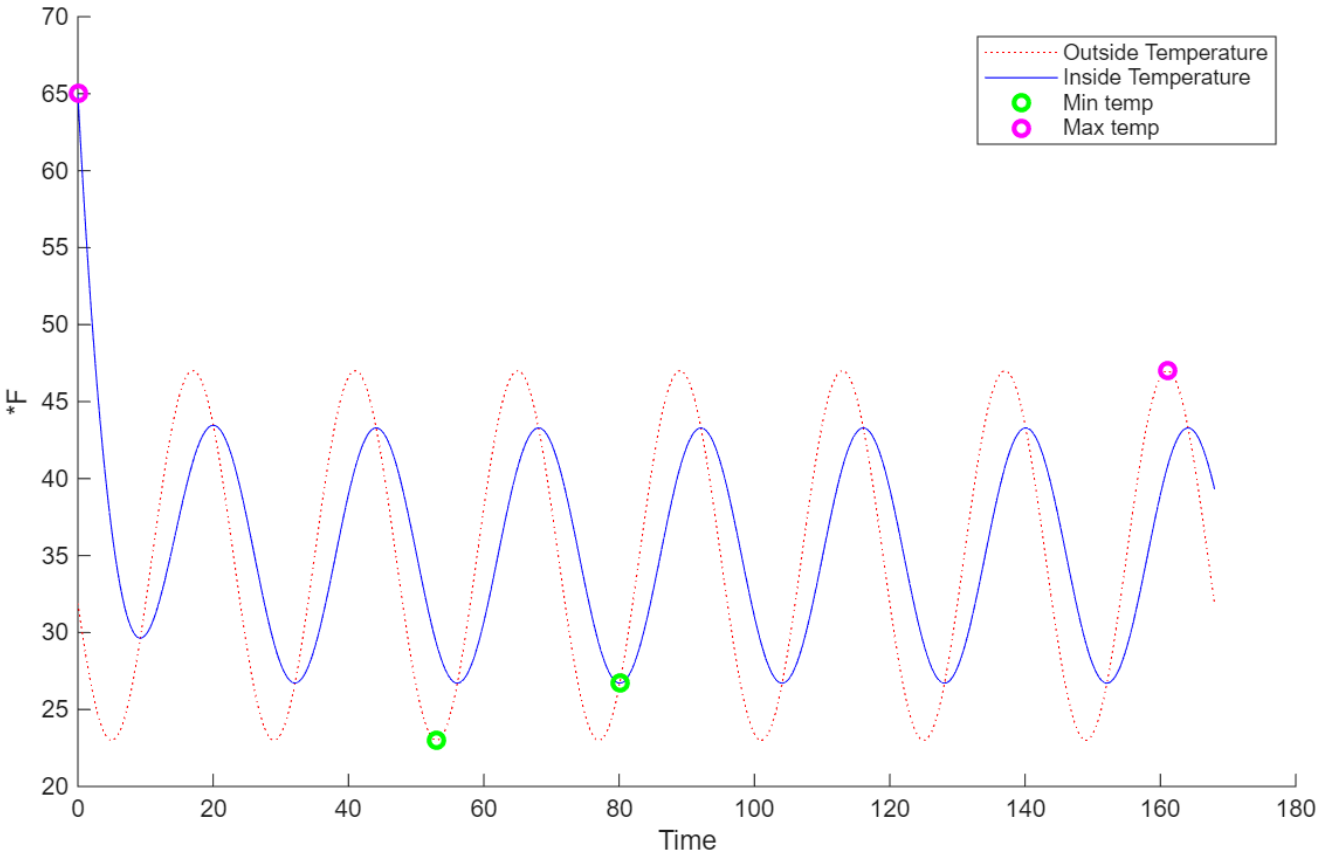


Figure C.3: Comparison of external & internal temperatures of the compound over a 180 hours time-frame

as over time the wave of internal temperature stabilizes. This results from the fact that as $t \to \infty, Ce^{-t/4} \to 0$, and as a result

$$T(t, C) = M_0 - Ce^{-t/4} - \frac{36}{9 + \pi^2}(3\cos(\theta) + \pi\cos(\theta))$$

approaches

$$T(t) = M_0 - \frac{36}{9 + \pi^2}(3\cos(\theta) + \pi\cos(\theta))$$

which is a constantly oscillating function. This also means that the initial value of T becomes irrelevant over time, and no matter how hot it is it will eventually reach the same posture.

We can see that our current model would allow the Zucc to be outside of his habitat requirements, and we must introduce more heat-generating elements to keep Mark toasty.

Effects of Human Interaction with System

Goal:

Our goal now is to add another layer, and model the effects of people occupying the compound in question. This will first be done without the additional effects of Repti Coolers ^{TM} & heat lamps in mind, making our equation much simpler to start (however, we will look at air conditioning later).

Technical Approach:

Again, we shall be approximating a given function utilizing fourth-order Runge-Kutta estimation. This time, our given function to model heat input is

$$H(t) = 7\,\mathrm{sech}\left(\frac{3}{4}(t-10)\right)$$

where H(t) is used to represent lighting and data center heat. We will manipulate this function to be used in MATLAB with the Runge-Kutta function, to do so we need to solve it analytically,

$$H(t) = 7\,\mathrm{sech}\big(\tfrac{3}{4}(t-10)\big)$$

where $H(t)$ represents lighting and data center heat.
We will manipulate this function to be used in MATLAB with the Runge-Kutta function. To do so, we need to solve it analytically:

$$\frac{dy}{dt} = H(t) = 7\,\mathrm{sech}\big(\tfrac{3}{4}(t-10)\big), \quad y(t_0) = y_0$$

which allows us to then input into our Runge-Kutta solver, with the following steps:

$$\begin{aligned} t_j &= t(j), \\ H_j &= H(j), \\ \\ k_1 &= \Delta t \cdot 7\,\mathrm{sech}\big(\tfrac{3}{4}(t_j - 10)\big), \\ k_2 &= \Delta t \cdot 7\,\mathrm{sech}\big(\tfrac{3}{4}\big((t_j + \tfrac{\Delta t}{2}) - 10\big)\big), \\ k_3 &= \Delta t \cdot 7\,\mathrm{sech}\big(\tfrac{3}{4}\big((t_j + \tfrac{\Delta t}{2}) - 10\big)\big), \\ k_4 &= \Delta t \cdot 7\,\mathrm{sech}\big(\tfrac{3}{4}((t_j + \Delta t) - 10)\big), \\ \\ H_{j+1} &= H_j + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned}$$

Now using this to format our MATLAB code, we can simulate the estimated heating effects of compound usage, without temperature control systems. After running this we are left with the output of the following graphs:

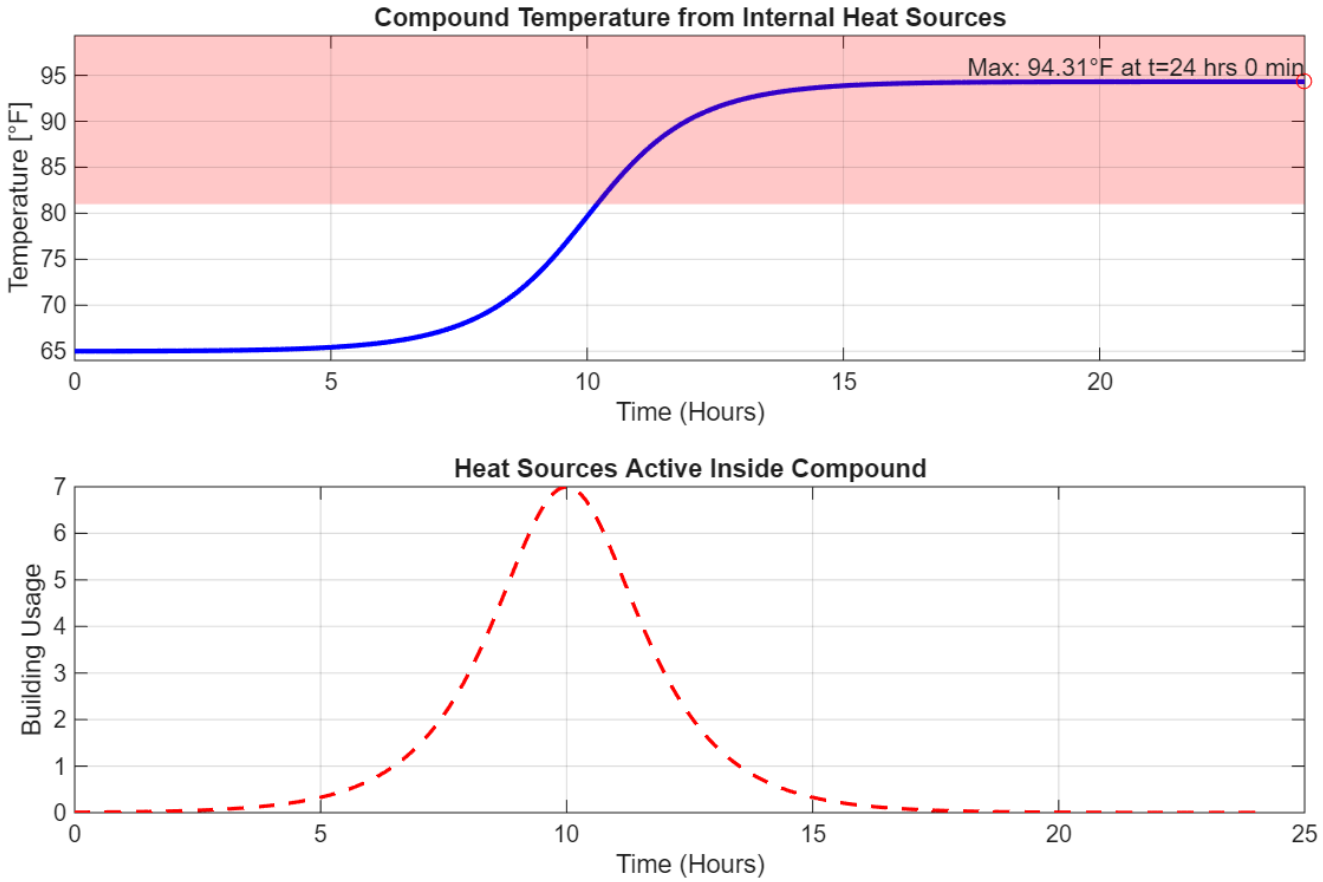


Figure D.1: Overall temperature in the compound in a day from internal heat sources, without temperature control

In *Figure D.1*, we can see the overall temperature in the compound (top) and the active usage of the compound (bottom). On the top we can also see the outputted max temperature of 94.31$^{\circ}F$, which is shown to occur right at the 24 hour mark, which is the limit for where we are focused as it is one full day. We can think of the top graph, which shows our estimation for $H(t)$, to represent the total heat buildup within a compound, as our model currently only accepts inputs from interior heating and has no way to dissipate that heat. We are later given a constraint that states that equipment inside will be damaged by the heat should it rise above 8$^{\circ}F$, so that 'no go' area is shown here as the area above 81$^{\circ}F$ on the y-axis.

Effects of Reptile Temperature Control

Goal:

Adding onto the models we have so far built up, we now need to introduce the effects of Repti Coolers ^{TM} and heat lamps, we will just call this "temperature control". This will be again, calculated separately from other heating elements (outdoor temperature and human produced heating). We want to build this final model to then integrate all our heating and cooling models into one.

Technical Approach

We are given the equation $Q(t) = \kappa_d(T_d - T)$ to model our temperature control effects, where $Q(t)$ represents the effects air conditioners and furnaces on our buildings heat. T_d is our

thermostats set temperature, and will be constant, T is the buildings overall temperature at any given moment, and κ_d is either the quantity of furnaces and air conditioners present in the building, or the efficiency of those devices, where installing more or better devices will change the ability to effectively cool or heat the air.

We can use our Runge-Kutta solver to model this equation, which when setting up will look like:

$$
\begin{aligned}
t_j &= t(j), \\
Q_j &= Q(j), \\
k_1 &= \Delta t\, \kappa d\big(T_d - T(t_j)\big), \\
k_2 &= \Delta t\, \kappa d\big(T_d - T(t_j + \tfrac{\Delta t}{2})\big), \\
k_3 &= \Delta t\, \kappa d\big(T_d - T(t_j + \tfrac{\Delta t}{2})\big), \\
k_4 &= \Delta t\, \kappa d\big(T_d - T(t_j + \Delta t)\big), \\
Q_{j+1} &= Q_j + \tfrac{1}{6}\big(k_1 + 2k_2 + 2k_3 + k_4\big).
\end{aligned}
$$

Using MATLAB, we now want to model four different scenarios for our input values: $T(0) = 65, \kappa_d = 0.2)$, $T(0) = 65, \kappa_d = 2.0)$, $T(0) = 95, \kappa_d = 0.2)$, $T(0) = 95, \kappa_d = 2.0)$ with $T_d = 77$ to model different initial temperatures, and different rates of cooling/heating will impact the overall heat in the building over time. We can now run our MATLAB script to get the following figure plotting all four variations over a 24 hour interval:

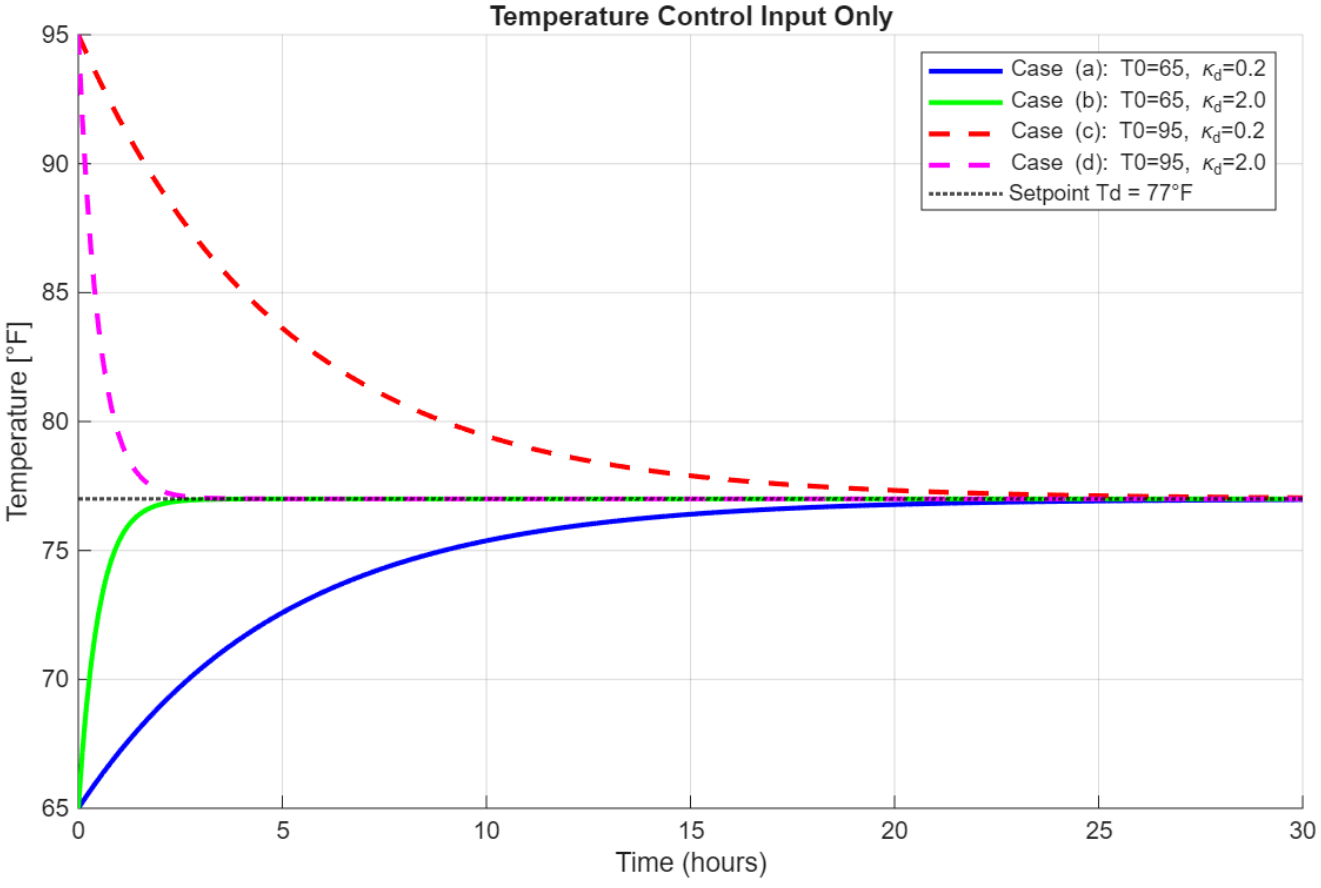


Figure E.1: Effects of temperature control measures on the internal temperature of the compound, observed with different internal temperatures and control strengths

We can make the observation that all functions will approach our thermostats set point of 77 $^\circ F$, denoted by the black asymptote line. With a larger κ_d , we see a much more rapid change in air temperature.

Full Model

We will now take into consideration the effects of of outside temperature, $A(t)$, people, lights and our Meta AI Slop data centers, $H(t)$, and heat lamps/Repti Coolers ^{TM}, $Q(t)$. We achieve this by including each function in $\frac{dT}{dt}$ to understand how each affects T, and finally adding everything in a single DE to obtain an accurate model for the temperature inside the building. Our goal is to burn enough rain forests to keep Mark warm until his mind is uploaded to the Metaverse.

Effects of lights & AI Slop data centers $H(t)$

We are provided an IVP to find $\frac{dT}{dt}$ with $H(t)$ considered: $\frac{dT}{dt} = 7\ \text{sech}\left[\frac{3}{4}(t-10)\right] + 2(77-T)$,

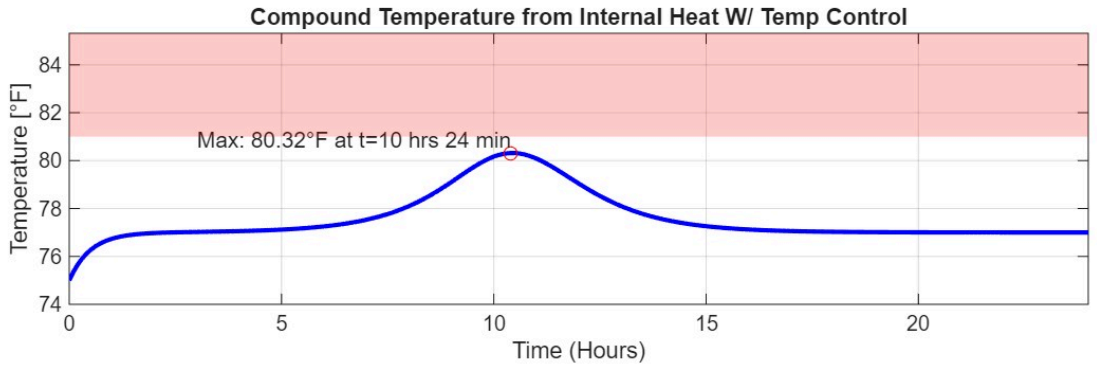


Figure F.1: Temperature change within the compound under the effects of temperature control measures

From *Figure F.1*, we find that the maximum temperature within the building is 80.32F at 10:24AM, after which the temperature drops and is maintained at approximately 77F for the rest of the day. Our damage threshold is 81F, and since $T_{max} < 81$, we narrowly avoid overheating Zuck.

Repti Coolers^{TM} are critical to ensure the equipment within the building functioning smoothly. We prove this by simulating the same compound on a hot weekend day, but without any data

centers or Repti Coolers^{TM}, which means $H(t) = Q(t) = 0$. We model this using our given IVP: $\frac{dT}{dt} = 0.2585 - 10\cos[\frac{\pi(t-5)}{12}] - T$, $T(0) = 75$

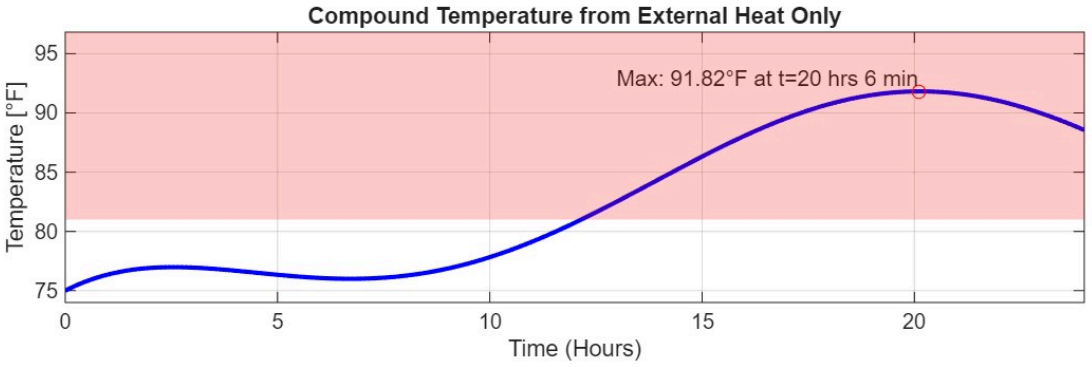


Figure F.2: Heat accumulating within the building from external heat, without regulation of Repti Coolers^{TM}*

In *Figure F.2*, we see that the temperature crosses the threshold 81 at 12:09PM, and remains above the threshold throughout the day, with the highest temperature being 91.82$^{\circ}F$ at 8:06PM. So Repti Coolers is definitely important!

Including Devices (ACs & Furnaces)

With active Repti Coolers ^{TM} & heat lamps, the maximum temperature inside the compound depends on how powerful these devices are.

We use another MATLAB code to model how these devices affect the compounds temperature

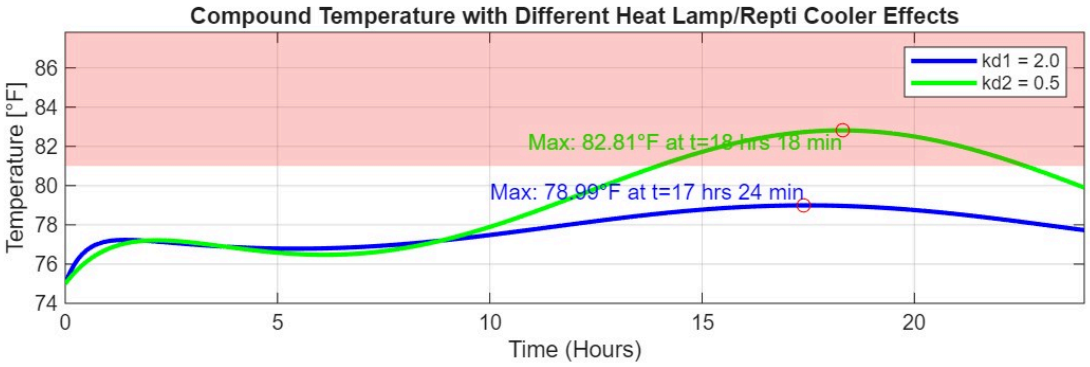


Figure F.3: Temperature inside the compound under different heat lamps/Repti Coolers^{TM} conditions

If the devices have a scaling factor $\kappa_d = 2$, the maximum temperature is 78.99°F. If the devices have a scaling factor $\kappa_d = 0.5$, the maximum temperature is 82.1°F.

```
The equipment was exposed to damaging temperature for 8.7 hours
The equipment was exposed to damaging temperature for 522 minutes.
```

From *Figure F.3* we can see that if $\kappa_d = 0.5$, the devices are exposed to damaging temperatures for up to 8.7 hours (522 minutes). Therefore, it is important to consider how powerful our heat lamps/Repti Coolers ^{TM} devices are and how they would affect the temperature, so that we can be sustainable.

Putting it together

We now add up all factors: outside ambience, modeled by the function $M(t) = 0.2585 - 10\cos[\frac{\pi(t-5)}{12}]$, people, machinery, and lights, and heating and cooling devices to understand how the temperature inside this building might fluctuate over a weekend (72 hours)

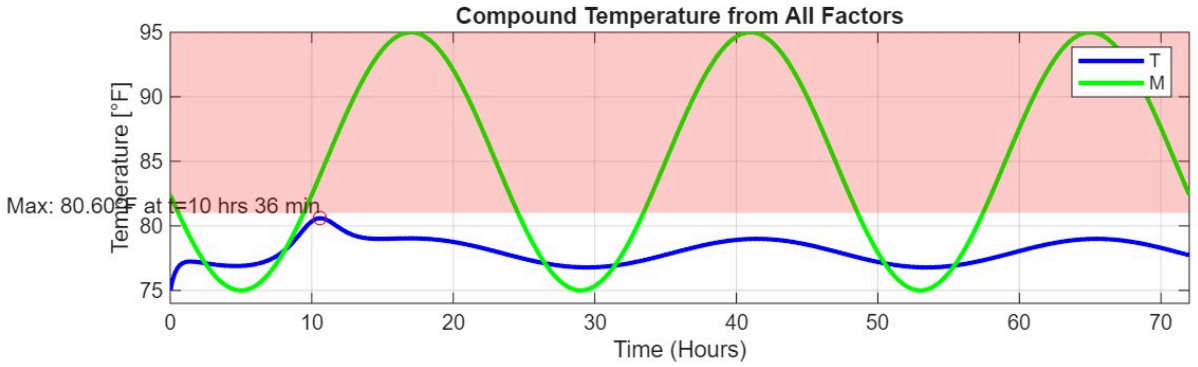


Figure F.4: Comparing internal compound temperature with outside temperature in a 24-hour span

From *Figure F.4*, we can see that the temperature curve (blue), fluctuates throughout the day in the form of a wave function. Therefore, it is safe to say that M does not approach a particular steady state, but rather, it's steady state is periodic, limited by a maximum and minimum value.

We also observe that there is a correlation between the fluctuation of the temperature curve and the outside temperature curve (yellow). As the outside temperature goes up, so does the compound's internal temperature, and vice-versa. We conclude that it is safe to say that the outside temperature drives the direction of the rate of change in temperature inside the compound: whether the building heats up, or cool down depends on whether the outside is heating up, or cooling down.

Our solution curve can be described as follows: as the compound powers up and people start to enter, it's interior rapidly heats up until the first 10 hours has passed, after which it begins cooling down steadily. After 20 hours, the compound's temperature is mainly decided by the outside temperature, if it's heating up outside, the compound heats up, and vice versa.

Conclusion

From our study, our temperatures should oscillate between 75-95$^\circ F$, which will keep Zuckerburg in the correct temperature range. Additionally, the data center load from the metaverse simulations will help heat up the rest of the world, so that Mark can vacation further north and get his amphibious side on when his Florida estate is underwater.

see Appendix & Code sections below

Appendix A

$$\frac{dT}{dt} = \kappa[M(t) - T(t)] + H(t) + Q(t)$$

1) The differential equation is a **linear, constant coefficient, nonhomogeneous** equation. If $Q(t) = tT$, the DE would be a **variable coefficient** equation.

2) For Picard's theorem to guarantee the existence of unique solutions, $M(t)$, $H(t)$ and $Q(t)$ must guarantee **continuity for the DE on the predetermined interval** (existence) and $f_T(t, T)$ **must be continuous on the interval** (uniqueness). This can be interpreted as the house having a **unique flow of temperature as time goes on.**

3)

$$\frac{dT}{dt} = \kappa[M(t) - T(t)] + H(t) + Q(t)$$

$$\frac{dT}{dt} - \kappa T(t) = \kappa M(t) + H(t) + Q(t)$$

Solve for $\mu(t)$**:**

We have $p(t) = \kappa$

$$\int p(t) = \kappa t$$

Therefore $\mu(t) = e^{\kappa t}$

Multiply both sides by Integrating Factor:

$$e^{\kappa t}[\frac{dT}{dt} - \kappa T(t)] = e^{\kappa t}[\kappa M(t) + H(t) + Q(t)]$$

$$\frac{d}{dt}[e^{\kappa t}T] = e^{\kappa t}[\kappa M(t) + H(t) + Q(t)]$$

$$e^{\kappa t}T = \int e^{\kappa t}[\kappa M(t) + H(t) + Q(t)]dt + C$$

$$T = \frac{\int e^{\kappa t}[\kappa M(t)+H(t)+Q(t)]dt + C}{e^{\kappa t}}$$

4)

a) "No people in it and the lights and machinery are off," so $H(t) = 0$
"No furnaces or air conditioners are running," so $Q(t) = 0$
"The outside temperature is constant with the value M," so $M(t) = M$

Our new DE is $\frac{dT}{dt} = \kappa[M - T(t)]$

b) The DE has equilibrium solutions if $\frac{dT}{dt} = 0$
$\kappa[M - T(t)] = 0$ if $T(t) = M$

c) If $T < M$, $\frac{dT}{dt} > 0$
If $T > M$, $\frac{dT}{dt} < 0$
T goes towards M as $t \to \infty$, so $T(t) = M$ is a stable equilibrium solution. This implies that **the temperature within the building would eventually be the same as that of the ambient temperature.**

Using solution from part 3:

$$T = \frac{\int e^{\kappa t}[\kappa M(t)+H(t)+Q(t)]dt + C}{e^{\kappa t}}$$

Apply the conditions from part 4a:

$$T = \frac{\kappa M \int e^{\kappa t} dt + C}{e^{\kappa t}}$$

$$T = M + Ce^{-\kappa t}$$

Apply initial condition $T(t_0) = T_0$

$$T_0 = M + Ce^{-\kappa t_0}$$

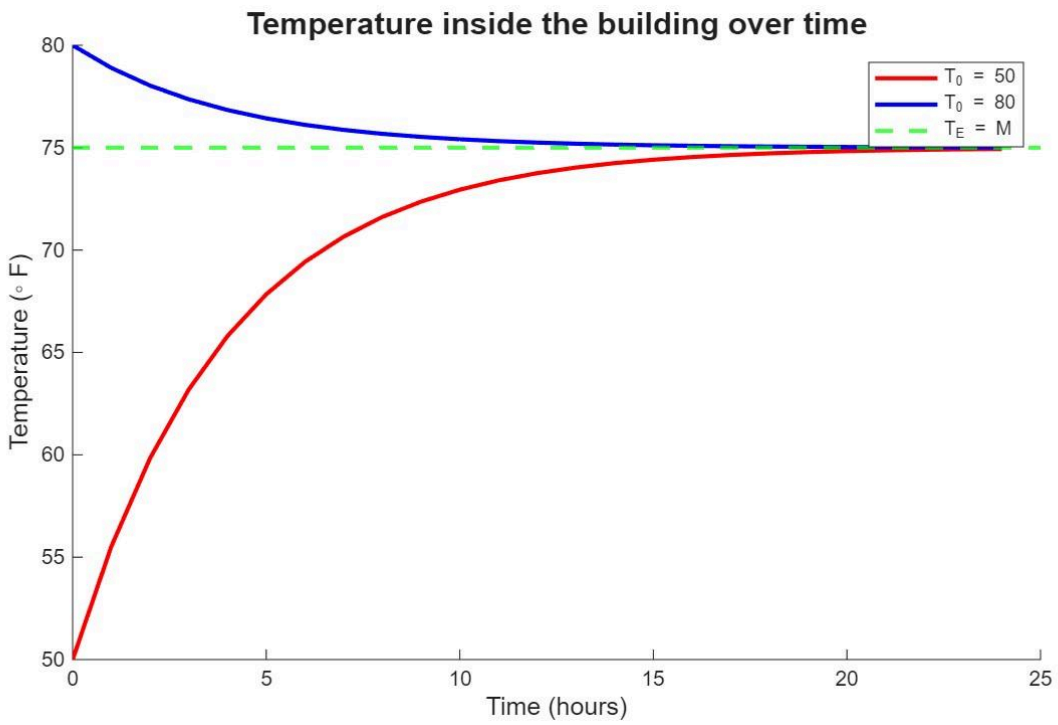
$$T_0 - M = Ce^{-\kappa t_0}$$

$$C = (T_0 - M)e^{\kappa t_0}$$

General solution:

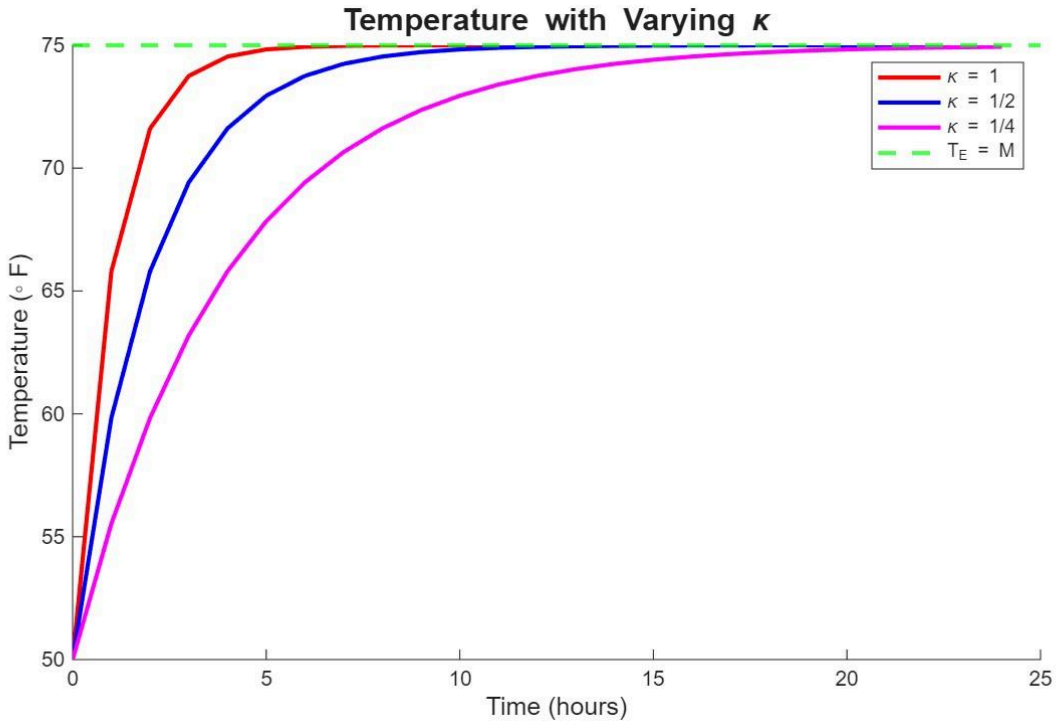
$$T(t) = M + (T_0 - M)e^{\kappa t_0}e^{-\kappa t}$$

d)



These solutions confirm part c)'s answer: $T(t) = M$ is a stable equilibrium solution

e)



From the graph, we can see that **while changing κ does not affect the equilibrium solution, the rate of change of temperature is changed: a bigger κ implies a greater rate of change, and vice versa.** The quality of the building's insulation could change κ, with quality insulation resulting in a slower heat loss (smaller κ), and poor insulation resulting in a sharper decrease in temperature (larger κ).

f)

$$T(t) = M + (T_0 - M)e^{\kappa t_0}e^{-\kappa t}$$

"The difference between the building's temperature and the outside temperature is e^{-1} of the initial difference."

$$T(t) - M_0 = e^{-1}(T_0 - M_0)$$

Using the general solution:

$$(T_0 - M_0)e^{-1} = (T_0 - M)e^{\kappa t_0}e^{-\kappa t}$$

$$e^{-1} = e^{\kappa t_0}e^{-\kappa t}$$
$$e^{-1} = e^{\kappa(t_0 - t)}$$
$$-1 = \kappa(t_0 - t)$$
$$-1 = -\kappa(t - t_0)$$
$$-1 = -\kappa\Delta t$$
$$\Delta t = \frac{1}{\kappa}$$

Since the time constant resembles the change in time, **the units should be hours**

In part 4f), we saw that a smaller κ implies smaller heat loss. Since Δt is **inversely proportional** to κ, it can be inferred that **a larger** Δt **would slow the building's response to outside temperature; therefore, we want a larger time constant**

B1:

Derivation of diffeq:

Q.

$$\frac{dT}{dt} = 0.25(75-T), \quad T(0)=50$$

$$T_g = T_p + T_h$$

T_h: $\frac{dT}{dt} = -\frac{T}{4}$

$$T = Ce^{-\frac{t}{4}}$$

$T_p = Ve^{-\frac{t}{4}}$

$$Ve^{-\frac{t}{4}} = \frac{75}{4}$$

$$V = \int \frac{75}{4} e^{\frac{t}{4}}$$

$$V = 75e^{\frac{t}{4}}$$

$$T_p = 75$$

$$T_g = 75 + Ce^{-\frac{t}{4}}$$

$$T(0) = 50 = 75 + C$$

$$C = -25$$

$$T_g = 75 - 25e^{-\frac{t}{4}}$$

C1: Solving Summer Diffeq

$$\frac{dI}{dt} = k(M(t) - T(t)),$$

$$M(t) = M_0 - 12\cos\left[\frac{\pi(t-5)}{12}\right], \quad M_0 = 75, \; T_0 = 65$$

Th: $\frac{dT}{dt} = -kT$

$$T = Ce^{-kt}$$

Tp: $v'e^{kt} = k\cdot\left(75 - 12\cos\left(\frac{\pi(t-5)}{12}\right)\right)$

$$v = \int 75e^{kt} - 12k\int \cos\left(\frac{\pi(t-5)}{12}\right)e^{kt}$$

$$k\int 75e^{kt} = 75e^{kt}$$

$12k\int \cos\left(\frac{\pi(t-5)}{12}\right)e^{kt}$? subbing in x for simplicity

$dx = \frac{\pi}{12}$ $\int x = \frac{12}{\pi}$

$$= \cos(x)e^{kt}$$

$u = \cos(x)$ $du = \frac{\pi\sin(x)}{12}$

$dv = e^{kt}$ $v = \frac{e^{kt}}{k}$

$$= 12\cos(x)e^{kt} + \int \pi\sin(x)e^{kt}$$

$$\pi\int -\sin(x)e^{kt}$$

$u = -\sin x$ $du = -\frac{\pi}{12}\cos x$

$dv = e^{kt}$ $v = \frac{e^{kt}}{k}$

$$= -\pi\sin x e^{kt} + \frac{\pi^2}{12k}\int \cos x e^{kt}$$

$$12k\int \cos(x)e^{kt} = 12\cos(x)e^{kt} + \pi\sin x e^{kt} - \frac{\pi^2}{12k}\int \cos x e^{kt}$$

$$\left(12k+\frac{\pi^2}{12k}\right)\int \cos(x)e^{kt} = 12\cos(x)e^{kt} + \pi \sin x\, e^{kt}$$

$$= \left(\frac{12k}{144k^2+\pi^2}\right)\cdot 12\cos(x)e^{kt} + \pi \sin x\, e^{kt}$$

$$\int \cos(x)e^{kt} = \frac{144k\cos(x)e^{kt} + 12\pi \sin x e^{kt}}{144k^2+\pi^2}$$

$\cdot 12k$ $\cdot 12k$

$$= \frac{(k12\cos(x)+\pi\sin(x))ke^{kt}}{k^2+\frac{\pi^2}{144}}$$

$$I = \frac{\left(k12\cos\left(\frac{\pi(t-5)}{12}\right)+\pi\sin\left(\frac{\pi(t-5)}{12}\right)\right)ke^{kt}}{k^2+\frac{\pi^2}{144}}$$

$$V = \left(75-\frac{\left(k12\cos\left(\frac{\pi(t-5)}{12}\right)+\pi\sin\left(\frac{\pi(t-5)}{12}\right)\right)k}{k^2+\frac{\pi^2}{144}}\right)e^{kt}$$

$$T_p = 75-\frac{\left(k12\cos\left(\frac{\pi(t-5)}{12}\right)+\pi\sin\left(\frac{\pi(t-5)}{12}\right)\right)k}{k^2+\frac{\pi^2}{144}}$$

$$T_g = 75+Ce^{-kt}-\left(\frac{\left(k12\cos\left(\frac{\pi(t-5)}{12}\right)+\pi\sin\left(\frac{\pi(t-5)}{12}\right)\right)k}{k^2+\frac{\pi^2}{144}}\right)$$

$k=0.25$:

$$T_g = 75+Ce^{-\frac{t}{4}}-36\left(\frac{3\cos\left(\frac{\pi(t-5)}{12}\right)+\pi\sin\left(\frac{\pi(t-5)}{12}\right)}{9+\pi^2}\right)$$

$$T_0 = 65 = 75 + C - 36\left(\frac{3\cos\left(-\frac{5\pi}{12}\right) + \pi\sin\left(-\frac{5\pi}{12}\right)}{9+\pi^2}\right)$$

$$C = -10 - 4.308 = -14.3$$

$$T_g = 75 - 14.3\,e^{-\frac{t}{4}} - 36\left(\frac{3\cos\left(\frac{\pi(t-5)}{12}\right) + \pi\sin\left(\frac{\pi(t-5)}{12}\right)}{9+\pi^2}\right)$$

$$T_g \text{ simplified} = 25 - 14.3\,e^{-\frac{t}{4}} - 1.907\left(3\cos(\phi) + \pi\sin(\phi)\right)$$

$$\left[\phi = \frac{\pi(t-5)}{12}\right]$$

C_2: Solving Winter Diffeq

$$T_y = 35 + Ce^{-\frac{t}{4}} - 36\left(\frac{3\cos\left(\frac{\pi(t-5)}{12}\right) + \pi\sin\left(\frac{\pi(t-5)}{12}\right)}{9+\pi^2}\right)$$

Resolving for C_2:

$65 = M_0 + C + 4.308$ ← unchanged by M_0

So we can make a function of C:

$C = 65 - M_0 - 4.308$

And use matlab

And now we just sub in C_2 and M_0 2 for the original eq:

Solving by hand:

$C_2 = 25.692$ ✓

Matlab:

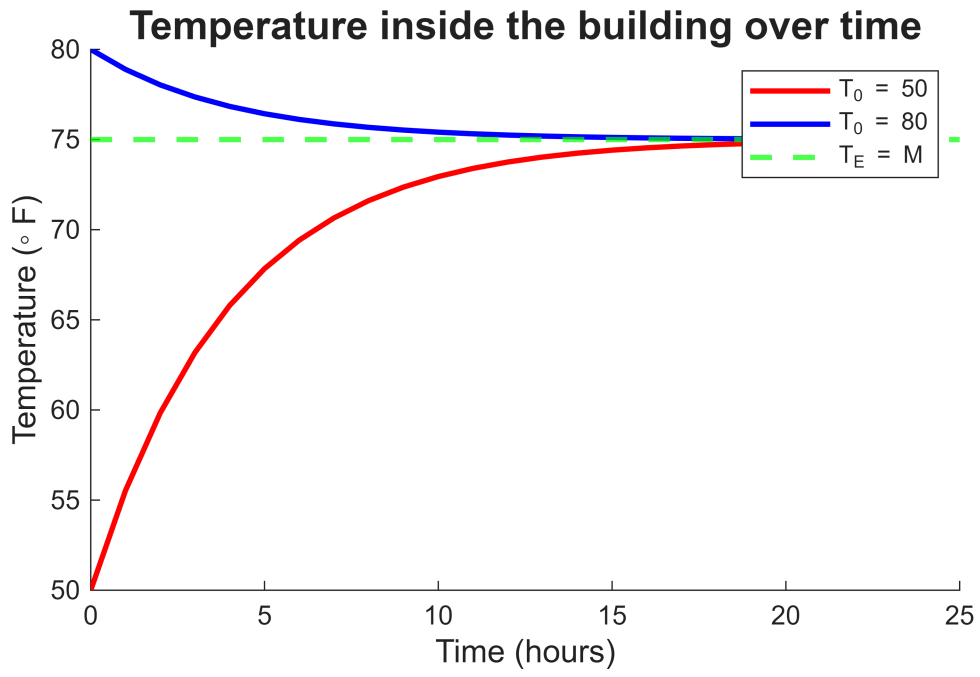
$C_2 = 25.692$ ✓

We are good to go!

Project 1

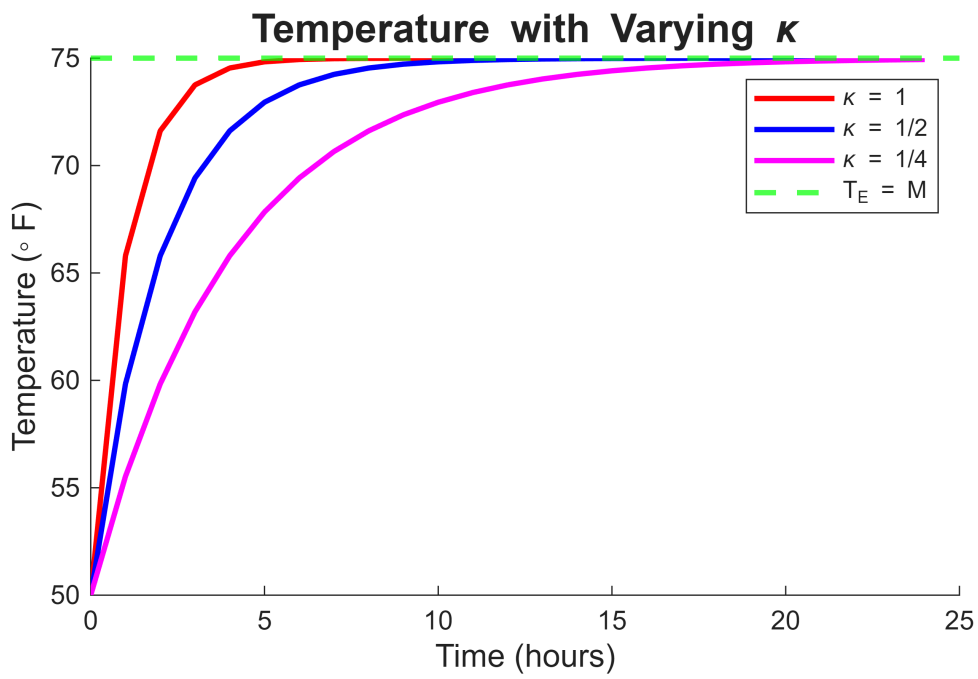
4e) Confirming Equilibrium at T = M

```
% Set variables
M_0 = 75;            % Temp of environment
t = 0:1:24;          % Time
kappa = 0.25;

% Solution functions with t = 0
T_50 = M_0 + (50-M_0)*exp(kappa*0)*exp(-kappa.*t); % T_0 = 50
T_80 = M_0 + (80-M_0)*exp(kappa*0)*exp(-kappa.*t); % T_0 = 80

% Plot
figure ();
hold on;             % Multiple plots
plot(t, T_50,'r','LineWidth',2);
plot(t, T_80,'b','LineWidth',2);
yline(M_0,'g--','LineWidth',2);                    % Equilibrium
hold off;

% Label graph
title('Temperature inside the building over time','FontSize',15);
xlabel('Time (hours)','FontSize',12);
ylabel('Temperature (◦ F)','FontSize',12);
legend('T_0 = 50', 'T_0 = 80','T_E = M');
```



4f) Effect of Kappa on T & dT/dt

```
% Solution functions with varrying kappa
T_k1 = M_0 + (50-M_0)*exp(1*0)*exp(-1.*t); % kappa = 1
T_k1_2 = M_0 + (50-M_0)*exp(1/2*0)*exp(-1/2.*t); % kappa = 1/2
T_k1_4 = M_0 + (50-M_0)*exp(1/4*0)*exp(-1/4.*t); % kappa = 1/4

% Plot the solutions with varying kappa
figure ();
hold on;
plot(t, T_k1, 'r', 'LineWidth', 2);
plot(t, T_k1_2, 'b', 'LineWidth', 2);
plot(t, T_k1_4, 'm', 'LineWidth', 2);
yline(M_0,'g--','LineWidth',2);
hold off;

% Label graph
title('Temperature with Varying \kappa','FontSize',15);
xlabel('Time (hours)','FontSize',12);
ylabel('Temperature (◦ F)','FontSize',12);
legend('\kappa = 1', '\kappa = 1/2', '\kappa = 1/4','T_E = M');
```



```
clear;close;clc;
```

This file acts to test the functionality of rk4.m's Runga-Kutta Approximation

Functions:

The function dt acts as our test differential equation, modeling Fig B1

Approximation

Variables for our RK4 are as below:

```
ti = 0; % t initial
tf = 24; % t final
npts = 240; % # of steps for our estimation
y0 = 50; % Starting T
f = @dt; % Reference equation

% Our unsolved differential equation
function T = dt(to,To)
    T = 75/4 - To/4;
end

% Calls rk4.m and stores it in a matrix

[test,Test] = rk4(ti,tf,npts,y0,f);
```

Real Value Calculation

Variable

```
ttrue = linspace(ti,tf,npts+1);
Ttrue = zeros(1,npts+1);

% Real Value Equation:
function T = dtrue(t)
    T = 75 - 25 * (exp(1)^(-t/4));
end

% Real Value Graph
for i = 1 : npts+1
    Ttrue(i) = dtrue(ttrue(i));
end
```

Calculation of error over time

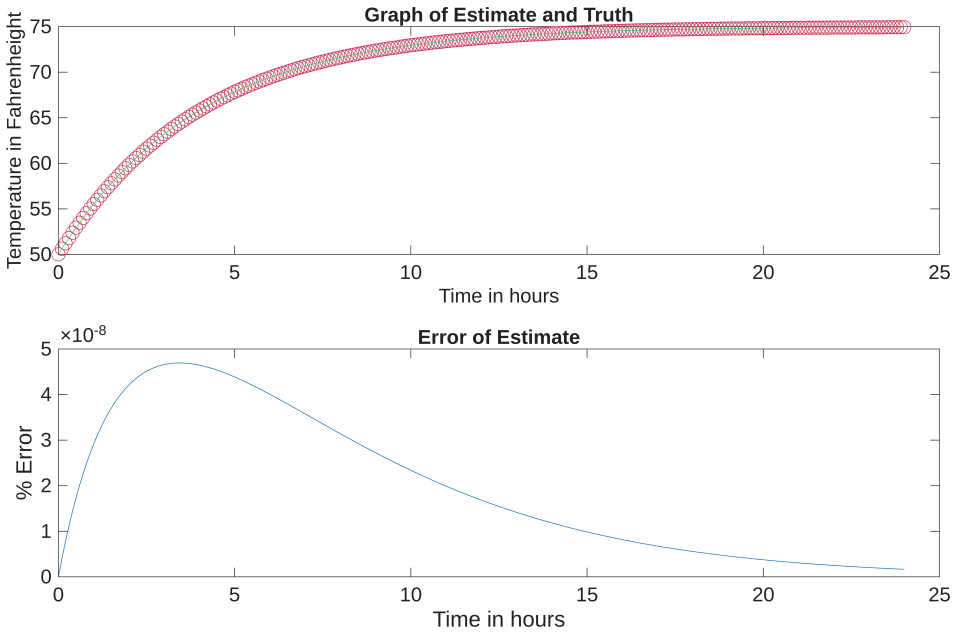
Error Matrix

```
Err = zeros(1, npts+1);
```

```
% Error Equation
for i = 1 : npts+1
    Err(i) = 100 * (Ttrue(i) - Test(i))/Ttrue(i);
end
```

Plotting

```
figure
tiledlayout
nexttile
% T of t graph
pest = plot(test,Test,"-o");
fontsize(15,"points")
title("Graph of Estimate and Truth","FontSize",15)
xlabel("Time in hours")
ylabel("Temperature in Fahrenheight")
pest.Color = '#cc2f59';
pest.MarkerSize = 10;
hold on;
ptrue=plot(ttrue,Ttrue);
ptrue.Color = '#2fcc4e';
hold off
nexttile
% Error Graph
perror = plot(test, Err);
fontsize(15,"points")
title("Error of Estimate","FontSize",15)
xlabel("Time in hours")
ylabel("% Error")
```



```
clear;close;clc;
```

Section C graphing:

Variables

Variables of the calculation range:

```
ti = 0; % t initial
tf = 168; % t final
npts = 1440; % step count
t = linspace(ti,tf,npts); % full t range for output
% C  and M control the C and Mo constants in the T(t) equation, alter them
% to change the starting temperature
Mo = 35;
```

C(Mo) calculation

```
function C=Ccalc(Mo)
    C=100-Mo-4.308;
end

% calculate C
C = Ccalc(Mo);
```

M(t) calculation

Function

```
function Tm = Tout(t, Mo)
    Tm = Mo - 12*cos(pi*((t-5)/12));
end

% Matrix
M = Tout(t,Mo);
```

T(t) calculation

Function

```
function T = Tin(t,Mo,C)
    theta = (pi*(t-5))/12;
    Th = C*(exp(-1*(t/4)));
    Tp = (36/(9+(pi^2)))*(3*cos(theta)+pi*sin(theta));
    T = Mo+Th-Tp;
end

% Matrix
```

```
T = Tin(t,Mo,C);
```

Critical points

```
[Mmin, tMmin] = min(M);
[Mmax, tMmax] = max(M);
[Tmin, tmin] = min(T);
[Tmax, tmax] = max(T);

function timeout = minutescalc(x, t)
    time=t(x);
    hours = floor(time);
    minutes = round((60 * (time - hours)));
    % Prevents the 6 hours and 60 minutes output I was getting
    if minutes == 60
        minutes = minutes - 1;
    end
    timeout = [hours,minutes];
end

% Grabbing max and min times
tmins = [tmin, tMmin];
Tmins = [Tmin, Mmin];
tmaxs = [tmax, tMmax];
Tmaxs = [Tmax, Mmax];

% Printing max/min times
Tspoints = [Tmin, Tmax, minutescalc(tmin,t), minutescalc(tmax,t)];
Mspoints = [Mmin, Mmax, minutescalc(tMmin,t), minutescalc(tMmax, t)];
formatspec1 = 'Inside temperature: Minimums and Maxs of %2.2f and %2.2f at
%2.0f:%02.0f and %2.0f:%02.0f\n';
formatspec2 = 'Outside temperature: Minimums and Maxs of %2.2f and %2.2f at
%2.0f:%02.0f and %2.0f:%02.0f\n';
fprintf(formatspec1, Tspoints)
```

```
Inside temperature: Minimums and Maxs of 26.71 and 100.00 at 80:05 and  0:00
```

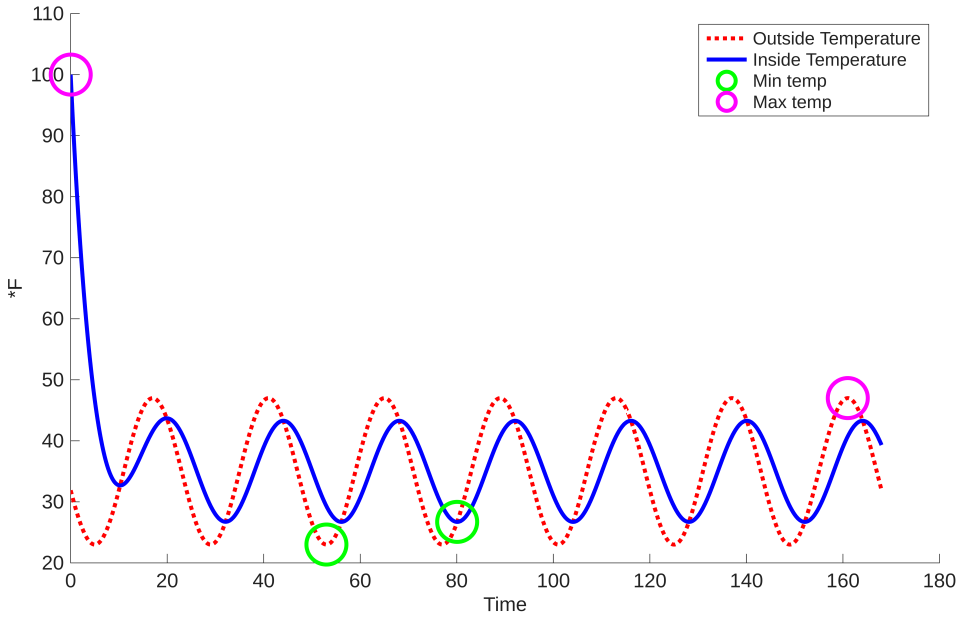
```
fprintf(formatspec2, Mspoints)
```

```
Outside temperature: Minimums and Maxs of 23.00 and 47.00 at 53:00 and 160:59
```

Plotting

```
figure
hold on
ToutPlot = plot(t,M,":r",LineWidth=3);
TinPlot = plot(t,T,"-b",LineWidth=3,MarkerSize=10);
TminPlot = plot(t(tmins),Tmins,"oG",MarkerSize=30,LineWidth=3);
TmaxPlot = plot(t(tmaxs),Tmaxs,"oM",MarkerSize=30,lineWidth=3);
xlabel("Time")
```

```
ylabel("*F")
fontsize(15,"points")
legend("Outside Temperature","Inside Temperature","Min temp","Max temp")
hold off
```



```m
% rk4_internal_heat.m
clc;
clear;

% Parameters
T0 = 65;                    % Initial temperature
t0 = 0;                     % Initial time
tf = 24;                    % Final time
dt = 0.1;                   % Time step
N = floor((tf - t0)/dt);    % Number of steps

% Time vector
t = t0:dt:tf;
T = zeros(1, length(t));    % Preallocate solution
T(1) = T0;                  % Initial condition

% Define internal heat source H(t)
H = @(t) 7 * sech((3/4)*(t - 10));    % Heat from people/lights/machines
%       ^^^  Change first value to modify input heat sources

% Define derivative function dT/dt
dTdt = @(t, T) H(t);          % No losses, only accumulation

% RK4 Integration
for i = 1:N
    ti = t(i);
    Ti = T(i);

    k1 = dt * dTdt(ti, Ti);
    k2 = dt * dTdt(ti + dt/2, Ti + k1/2);
    k3 = dt * dTdt(ti + dt/2, Ti + k2/2);
    k4 = dt * dTdt(ti + dt, Ti + k3);

    T(i+1) = Ti + (1/6)*(k1 + 2*k2 + 2*k3 + k4);
end

% Find and display max temperature and when it occurs
[max_T, idx_max] = max(T);      % Max value and its index
time_max_T = t(idx_max);        % Time at which max occurs

fprintf('Maximum temperature: %.2f °F\n', max_T);
```

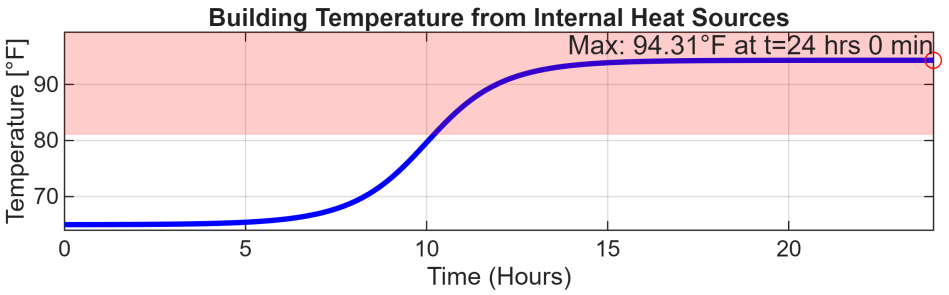
```
Maximum temperature: 94.31 °F
```

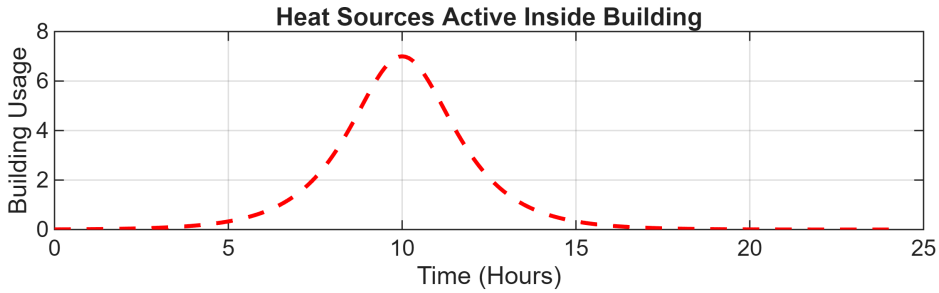
```m
fprintf('Time of maximum temperature: %.2f hours\n', time_max_T);
```

```
Time of maximum temperature: 24.00 hours
```

```m

% Plot the result
figure;
```

```
y_max = max_T + 5;

% Plot temperature T(t)
subplot(2, 1, 1);
plot(t, T, 'b-', 'LineWidth', 2);
xlabel('Time (Hours)');
ylabel('Temperature [°F]');
title('Building Temperature from Internal Heat Sources');
xlim([0 24]);        % Fix x-axis to 0–24 hours
ylim([min(T)-1, y_max]);  % Pad lower limit slightly for visibility
grid on;

% Add a red area to denote unsafe temps
hold on;
y_fill = 81 * ones(size(t));
y_max = max(T) + 5;
fill([t, fliplr(t)], [y_fill, y_max * ones(size(t))], ...
     'r', 'FaceAlpha', 0.2, 'EdgeColor', 'none');
hold off;

% Add text to the graph where max temp occurs
hold on; % Keep the current plot
plot(time_max_T, max_T, 'ro'); % Mark the max temperature point
time_hours = floor(time_max_T); % Get the integer hours
time_minutes = round((time_max_T - time_hours) * 60); % Round minutes to nearest
integer
text(time_max_T, max_T, sprintf('Max: %.2f°F at t=%d hrs %d min', max_T,
time_hours, time_minutes), ...
    'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'right');
hold off; % Release the plot

% Plot H(t) for comparison

%{
hold on;
plot(t, H(t), 'r--', 'LineWidth', 1.5);
legend('T(t)', 'H(t)');
%}

% Create a separate subplot for building usage
subplot(2, 1, 2);
plot(t, H(t), 'r--', 'LineWidth', 1.5);
xlabel('Time (Hours)');
ylabel('Building Usage');
title('Heat Sources Active Inside Building');
grid on;
```

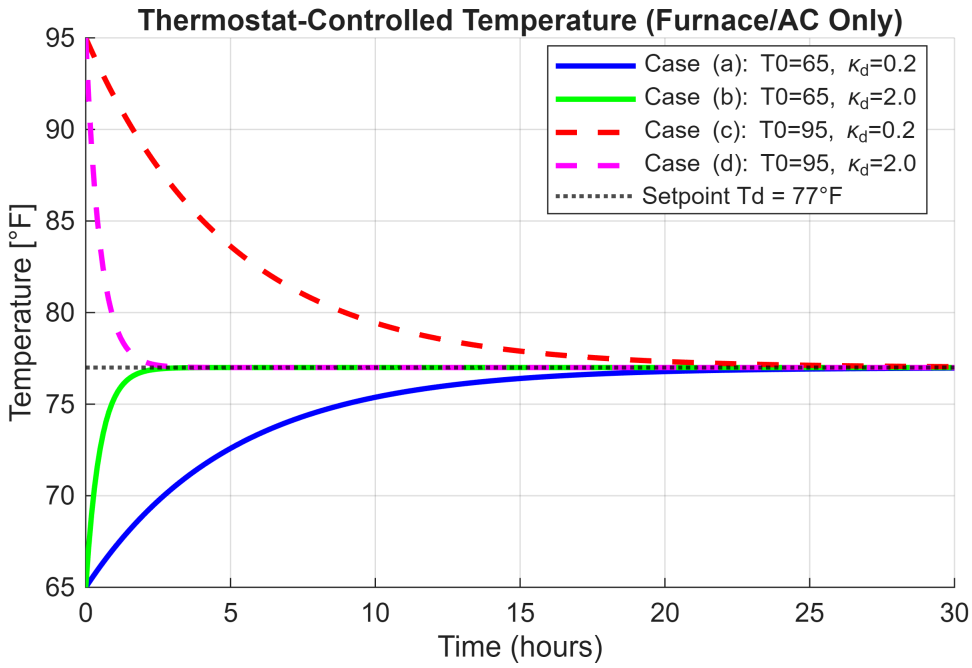





```
% --- Helper function for sech ---
function y = sech(x)
    y = 1 ./ cosh(x);
end
```

```
% rk4_thermostat_compare.m
clc;
clear;

% Simulation parameters
Td = 77;              % Thermostat setpoint
t0 = 0;
tf = 30;
dt = 0.1;
t = t0:dt:tf;
N = length(t);

% Case definitions
cases = {
    65, 0.2, 'b-', 'Case (a): T0=65, \kappa_d=0.2';
    65, 2.0, 'g-', 'Case (b): T0=65, \kappa_d=2.0';
    95, 0.2, 'r--','Case (c): T0=95, \kappa_d=0.2';
    95, 2.0, 'm--','Case (d): T0=95, \kappa_d=2.0';
};

% Prepare plot
figure;
hold on;

% Loop through cases
for i = 1:4
    T0 = cases{i,1};
    kappa_d = cases{i,2};
    style = cases{i,3};
    label = cases{i,4};

    k = kappa_d;

    % Initialize temperature vector
    T = zeros(1, N);
    T(1) = T0;

    % Define the differential equation
    dTdt = @(t, T) -k * (T - Td);

    % RK4 integration
    for j = 1:N-1
        tj = t(j);
        Tj = T(j);

        k1 = dt * dTdt(tj, Tj);
        k2 = dt * dTdt(tj + dt/2, Tj + k1/2);
        k3 = dt * dTdt(tj + dt/2, Tj + k2/2);
        k4 = dt * dTdt(tj + dt, Tj + k3);
```

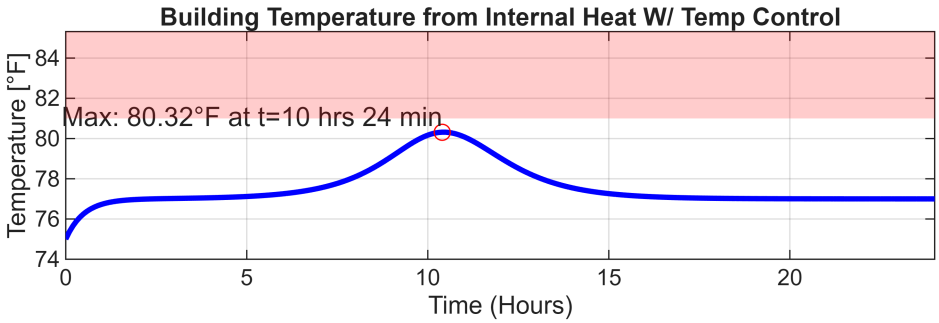
```
        T(j+1) = Tj + (1/6)*(k1 + 2*k2 + 2*k3 + k4);
    end

    % Plot the result
    plot(t, T, style, 'LineWidth', 2, 'DisplayName', label);
end

% Finalize plot
yline(Td, 'k:', 'LineWidth', 1.5, 'DisplayName', 'Setpoint Td = 77°F');
xlabel('Time (hours)');
ylabel('Temperature [°F]');
title('Thermostat-Controlled Temperature (Furnace/AC Only)');
legend('Location', 'best');
grid on;
```



5.1

```
% ////////////////////////////////////////////////
% (RUN ONE SECTION AT A TIME OR CHANGE VAIRABLE NAMES)
% ////////////////////////////////////////////////

% rk4_internal_heat_w_ac.m
clc;
clear;

% Parameters
T0 = 75;                    % Initial temperature
t0 = 0;                     % Initial time
tf = 24;                    % Final time
dt = 0.1;                   % Time step
N = floor((tf - t0)/dt);    % Number of steps

% Time vector
t = t0:dt:tf;
T = zeros(1, length(t));    % Preallocate solution
T(1) = T0;                  % Initial condition

% Define internal heat source H(t)
H = @(t, T) 7 * sech((3/4)*(t - 10)) + 2 * (77 - T);    % Heat from people/lights/
machines
%          ^^^  Change first value to modify input heat sources

% Define derivative function dT/dt
dTdt = @(t, T) H(t, T);          % No losses, only accumulation

% RK4 Integration
for i = 1:N
    ti = t(i);
    Ti = T(i);

    k1 = dt * dTdt(ti, Ti);
    k2 = dt * dTdt(ti + dt/2, Ti + k1/2);
    k3 = dt * dTdt(ti + dt/2, Ti + k2/2);
    k4 = dt * dTdt(ti + dt, Ti + k3);

    T(i+1) = Ti + (1/6)*(k1 + 2*k2 + 2*k3 + k4);
end

% Find and display max temperature and when it occurs
[max_T, idx_max] = max(T);      % Max value and its index
time_max_T = t(idx_max);        % Time at which max occurs

% Convert time to duration
time_duration = duration(floor(time_max_T), mod(time_max_T*60, 60), 0); % Convert
to hours and minutes
```

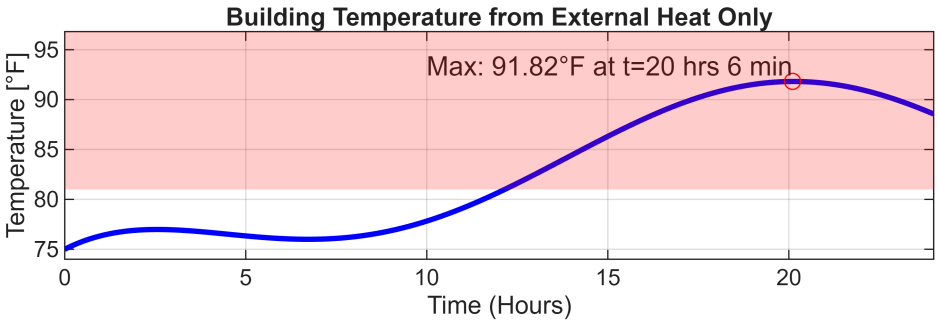
```
% Plot the result
figure;
y_max = max_T + 5;

% Plot temperature T(t)
subplot(2, 1, 1);
plot(t, T, 'b-', 'LineWidth', 2);
xlabel('Time (Hours)');
ylabel('Temperature [°F]');
title('Building Temperature from Internal Heat W/ Temp Control');
xlim([0 24]);        % Fix x-axis to 0–24 hours
ylim([min(T)-1, y_max]);  % Pad lower limit slightly for visibility
grid on;

% Add text to the graph where max temp occurs
hold on; % Keep the current plot
plot(time_max_T, max_T, 'ro'); % Mark the max temperature point
time_hours = floor(time_max_T); % Get the integer hours
time_minutes = round((time_max_T - time_hours) * 60); % Round minutes to nearest
integer
text(time_max_T, max_T, sprintf('Max: %.2f°F at t=%d hrs %d min', max_T,
time_hours, time_minutes), ...
    'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'right');
hold off; % Release the plot

% Add a red area to denote unsafe temps
hold on;
y_fill = 81 * ones(size(t));
y_max = max(T) + 5;
fill([t, fliplr(t)], [y_fill, y_max * ones(size(t))], ...
     'r', 'FaceAlpha', 0.2, 'EdgeColor', 'none');
hold off;
```



5.2 (RUN ONE SECTION AT A TIME OR CHANGE VAIRABLE NAMES)

```
% Parameters
T0 = 75;                    % Initial temperature
t0 = 0;                     % Initial time
tf = 24;                    % Final time
dt = 0.1;                   % Time step
N = floor((tf - t0)/dt);    % Number of steps

% Time vector
t = t0:dt:tf;
T = zeros(1, length(t));    % Preallocate solution
T(1) = T0;                  % Initial condition

% Define internal heat source H(t)
H = @(t, T) 0.25 * (85 - 10 * cos((pi * (t - 5) / 12)) - T);    % Heat from people/
lights/machines
%          ^^^  Change first value to modify input heat sources

% Define derivative function dT/dt
dTdt = @(t, T) H(t, T);          % No losses, only accumulation

% RK4 Integration
for i = 1:N
    ti = t(i);
    Ti = T(i);

    k1 = dt * dTdt(ti, Ti);
    k2 = dt * dTdt(ti + dt/2, Ti + k1/2);
    k3 = dt * dTdt(ti + dt/2, Ti + k2/2);
```

```
    k4 = dt * dTdt(ti + dt, Ti + k3);

    T(i+1) = Ti + (1/6)*(k1 + 2*k2 + 2*k3 + k4);
end

% Find and display max temperature and when it occurs
[max_T, idx_max] = max(T);      % Max value and its index
time_max_T = t(idx_max);        % Time at which max occurs

% Convert time to duration
time_duration = duration(floor(time_max_T), mod(time_max_T*60, 60), 0); % Convert
to hours and minutes

% Plot the result
figure;
y_max = max_T + 5;

% Plot temperature T(t)
subplot(2, 1, 1);
plot(t, T, 'b-', 'LineWidth', 2);
xlabel('Time (Hours)');
ylabel('Temperature [°F]');
title('Building Temperature from External Heat Only');
xlim([0 24]);        % Fix x-axis to 0–24 hours
ylim([min(T)-1, y_max]);  % Pad lower limit slightly for visibility
grid on;

% Add text to the graph where max temp occurs
hold on; % Keep the current plot
plot(time_max_T, max_T, 'ro'); % Mark the max temperature point
time_hours = floor(time_max_T); % Get the integer hours
time_minutes = round((time_max_T - time_hours) * 60); % Round minutes to nearest
integer
text(time_max_T, max_T, sprintf('Max: %.2f°F at t=%d hrs %d min', max_T,
time_hours, time_minutes), ...
    'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'right');
hold off; % Release the plot

% Add a red area to denote unsafe temps
hold on;
y_fill = 81 * ones(size(t));
y_max = max(T) + 5;
fill([t, fliplr(t)], [y_fill, y_max * ones(size(t))], ...
     'r', 'FaceAlpha', 0.2, 'EdgeColor', 'none');
hold off;
```

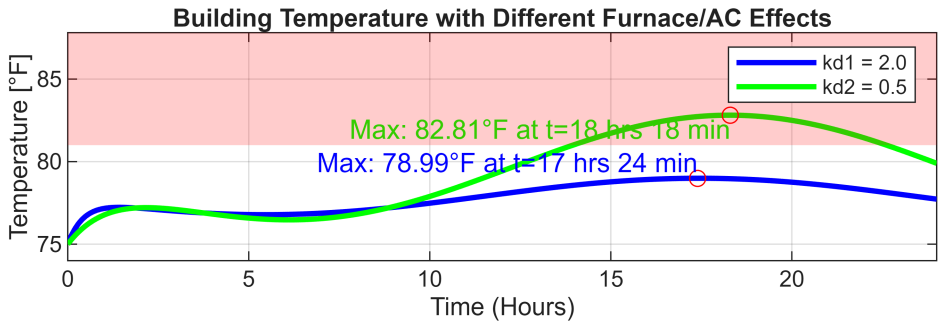


5.3

```
% Parameters
T0 = 75;                    % Initial temperature
t0 = 0;                     % Initial time
tf = 24;                    % Final time
dt = 0.1;                   % Time step
N = floor((tf - t0)/dt);    % Number of steps
kd1 = 2;                    % Effect of furnaces and ACs
kd2 = .5;

% Time vector
t = t0:dt:tf;
T1 = zeros(1, length(t));   % Preallocate solution for kd1
T2 = zeros(1, length(t));   % Preallocate solution for kd2
T1(1) = T0;                 % Initial condition for kd1
T2(1) = T0;                 % Initial condition for kd2

% Define internal heat source H(t)
H1 = @(t, T) 0.25 * (85 - 10 * cos((pi * (t - 5) / 12)) - T) + kd1*(77 - T);    %
Heat from people/lights/machines, including effect of furnaces & ACs
%          ^^^  Change first value to modify input heat sources
H2 = @(t, T) 0.25 * (85 - 10 * cos((pi * (t - 5) / 12)) - T) + kd2*(77 - T);

% Define derivative function dT/dt
dT1dt = @(t, T) H1(t, T);
dT2dt = @(t, T) H2(t, T);
```

```
% RK4 Integration (merged loop for kd1 and kd2)
for i = 1:N
    ti = t(i);

    % kd1
    Ti1 = T1(i);
    k1_1 = dt * dT1dt(ti, Ti1);
    k2_1 = dt * dT1dt(ti + dt/2, Ti1 + k1_1/2);
    k3_1 = dt * dT1dt(ti + dt/2, Ti1 + k2_1/2);
    k4_1 = dt * dT1dt(ti + dt, Ti1 + k3_1);
    T1(i+1) = Ti1 + (1/6)*(k1_1 + 2*k2_1 + 2*k3_1 + k4_1);

    % kd2
    Ti2 = T2(i);
    k1_2 = dt * dT2dt(ti, Ti2);
    k2_2 = dt * dT2dt(ti + dt/2, Ti2 + k1_2/2);
    k3_2 = dt * dT2dt(ti + dt/2, Ti2 + k2_2/2);
    k4_2 = dt * dT2dt(ti + dt, Ti2 + k3_2);
    T2(i+1) = Ti2 + (1/6)*(k1_2 + 2*k2_2 + 2*k3_2 + k4_2);
end

% Find and display max temperature for kd1
[max_T1, idx_max1] = max(T1);      % Max value and its index
time_max_T1 = t(idx_max1);         % Time at which max occurs

% Find and display max temperature for kd2
[max_T2, idx_max2] = max(T2);      % Max value and its index
time_max_T2 = t(idx_max2);         % Time at which max occurs

% Plot the result
figure;
y_max = max([max_T1, max_T2]) + 5;

% Plot temperature T(t)
subplot(2, 1, 1);
plotT1 = plot(t, T1, 'b-', 'LineWidth', 2); hold on;
plotT2 = plot(t, T2, 'g-', 'LineWidth', 2);
xlabel('Time (Hours)');
ylabel('Temperature [°F]');
title('Building Temperature with Different Furnace/AC Effects');
xlim([0 24]);        % Fix x-axis to 0–24 hours
ylim([min([T1,T2])-1, y_max]);  % Pad lower limit slightly for visibility
grid on;

% Add text to the graph where max temps occur
plot(time_max_T1, max_T1, 'ro'); % Mark the max temperature point for kd1
time_hours1 = floor(time_max_T1); % Get the integer hours
time_minutes1 = round((time_max_T1 - time_hours1) * 60); % Round minutes to nearest
integer
```

```
text(time_max_T1, max_T1, sprintf('Max: %.2f°F at t=%d hrs %d min', max_T1,
time_hours1, time_minutes1), ...
    'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'right', 'Color', 'b');

plot(time_max_T2, max_T2, 'ro'); % Mark the max temperature point for kd2
time_hours2 = floor(time_max_T2); % Get the integer hours
time_minutes2 = round((time_max_T2 - time_hours2) * 60); % Round minutes to nearest
integer
text(time_max_T2, max_T2, sprintf('Max: %.2f°F at t=%d hrs %d min', max_T2,
time_hours2, time_minutes2), ...
    'VerticalAlignment', 'top', 'HorizontalAlignment', 'right', 'Color', 'g');
hold off; % Release the plot

% Add a red area to denote unsafe temps
hold on;
y_fill = 81 * ones(size(t));
fill([t, fliplr(t)], [y_fill, y_max * ones(size(t))], ...
     'r', 'FaceAlpha', 0.2, 'EdgeColor', 'none');
hold off;

legend([plotT1 plotT2], sprintf('kd1 = %.1f', kd1), sprintf('kd2 = %.1f', kd2));
```




```
% Equipment is only exposed to unsafe temps if kd = 0.5, so we'll concentrate on
its plot

t_dmg = sum(T2 > 81) * dt;        % total hours
t_dmg_min = t_dmg * 60;           % total minutes

disp("The equipment was exposed to damaging temperature for " + t_dmg + " hours");
```

```
The equipment was exposed to damaging temperature for 8.7 hours
```

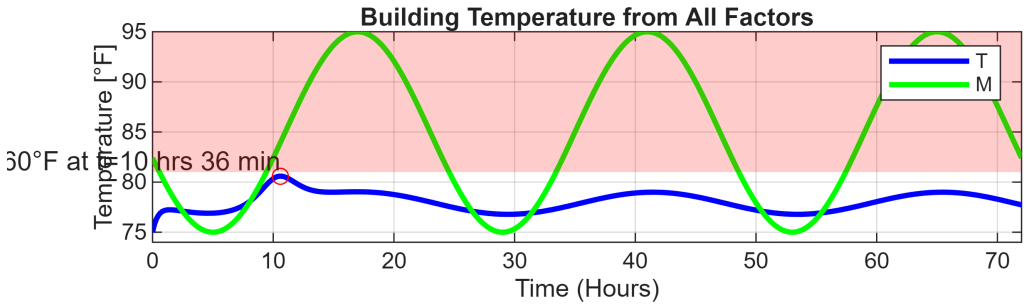
```
disp("The equipment was exposed to damaging temperature for " + t_dmg_min + "
minutes.");
```

```
The equipment was exposed to damaging temperature for 522 minutes.
```

5.4

```
% Parameters
T0 = 75;                    % Initial temperature
t0 = 0;                     % Initial time
tf = 72;                    % Final time
dt = 0.1;                   % Time step
N = floor((tf - t0)/dt);    % Number of steps

% Time vector
t = t0:dt:tf;
T = zeros(1, length(t));    % Preallocate solution
T(1) = T0;                  % Initial condition

% Define internal heat source H(t)
H = @(t, T) 0.25 * (85 - 10 * cos((pi * (t - 5) / 12)) - T) + 7 * sech((3/4)*(t -
10)) + 2*(77-T);    % Heat from people/lights/machines
%          ^^^  Change first value to modify input heat sources

% Define derivative function dT/dt
dTdt = @(t, T) H(t, T);          % No losses, only accumulation

% RK4 Integration
for i = 1:N
    ti = t(i);
    Ti = T(i);

    k1 = dt * dTdt(ti, Ti);
    k2 = dt * dTdt(ti + dt/2, Ti + k1/2);
    k3 = dt * dTdt(ti + dt/2, Ti + k2/2);
    k4 = dt * dTdt(ti + dt, Ti + k3);

    T(i+1) = Ti + (1/6)*(k1 + 2*k2 + 2*k3 + k4);
end

% Find and display max temperature and when it occurs
[max_T, idx_max] = max(T);      % Max value and its index
time_max_T = t(idx_max);        % Time at which max occurs

% Convert time to duration
time_duration = duration(floor(time_max_T), mod(time_max_T*60, 60), 0); % Convert
to hours and minutes
```

```
% Plot the result
figure;
y_max = max_T + 5;

% Define M(t)
M = 85 - 10 * cos((pi * (t - 5) / 12));

% Plot temperature T(t) and M(t) function
subplot(2, 1, 1);
plotT = plot(t, T, 'b-', 'LineWidth', 2);
hold on;
plotM = plot(t, M, 'g-', 'LineWidth', 2);
xlabel('Time (Hours)');
ylabel('Temperature [°F]');
title('Building Temperature from All Factors');
xlim([0 72]);        % Fix x-axis to 0–72 hours
ylim([min(T)-1, 95]);  % Pad lower limit slightly for visibility
grid on;
hold off;

% Add text to the graph where max temp occurs
hold on; % Keep the current plot
plot(time_max_T, max_T, 'ro'); % Mark the max temperature point
time_hours = floor(time_max_T); % Get the integer hours
time_minutes = round((time_max_T - time_hours) * 60); % Round minutes to nearest
integer
text(time_max_T, max_T, sprintf('Max: %.2f°F at t=%d hrs %d min', max_T,
time_hours, time_minutes), ...
    'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'right');
hold off; % Release the plot

% Add a red area to denote unsafe temps
hold on;
y_fill = 81 * ones(size(t));
y_max = 95;
fill([t, fliplr(t)], [y_fill, y_max * ones(size(t))], ...
     'r', 'FaceAlpha', 0.2, 'EdgeColor', 'none');
hold off;

% Legend
legend([plotT plotM], "T", "M");
```


Building Temperature from All Factors
Temperature [°F]
Time (Hours)
T
M
95
90
85
80
75
0
10
20
30
40
50
60
70
30°F at 10 hrs 36 min