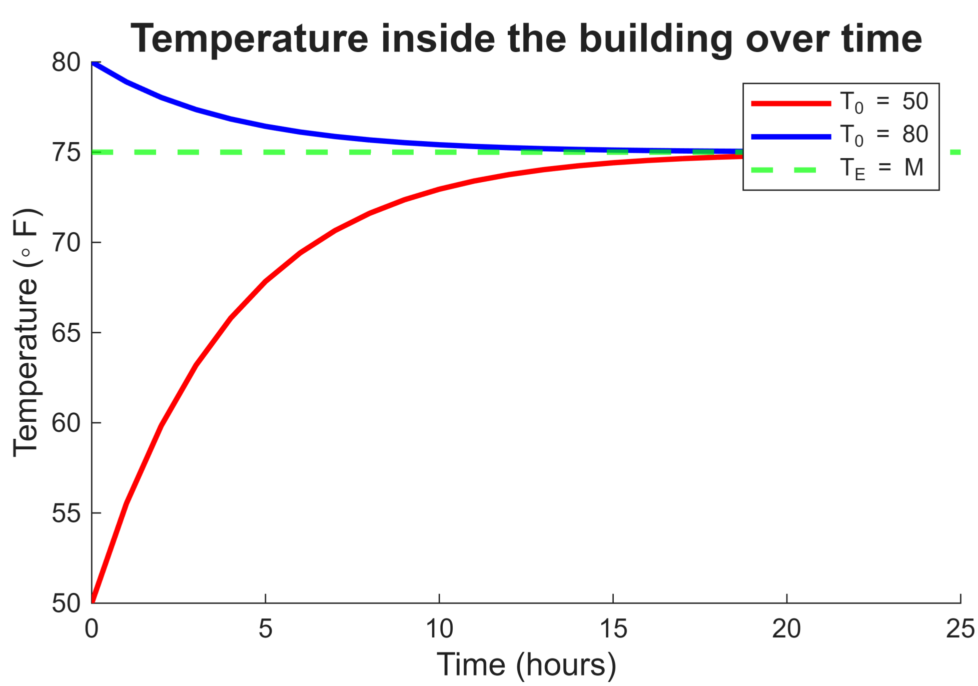
Project 1

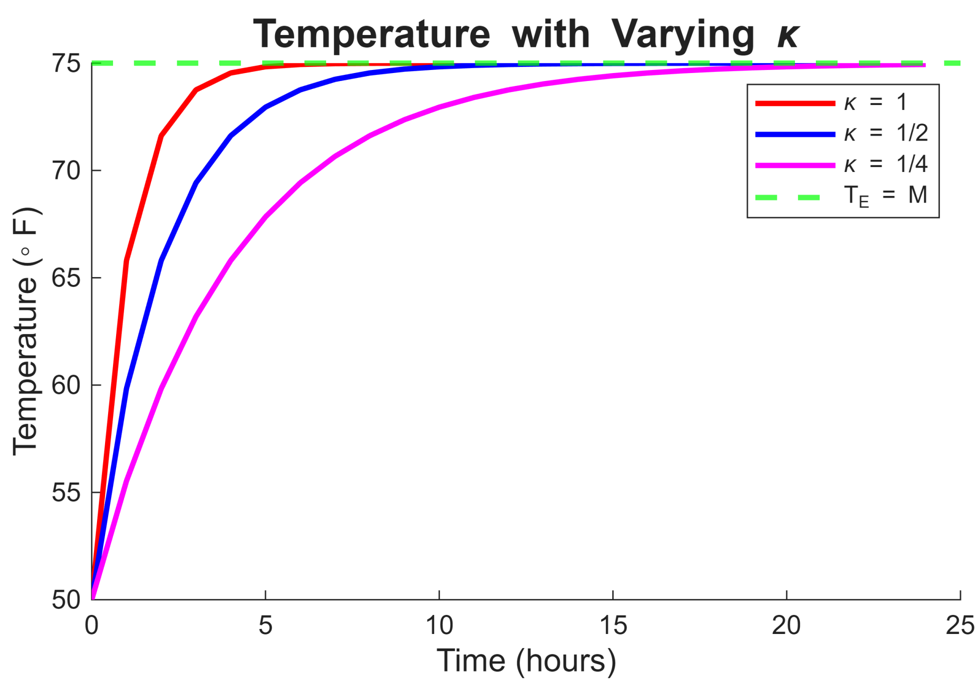
4e) Confirming Equilibrium at T = M

```
% 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

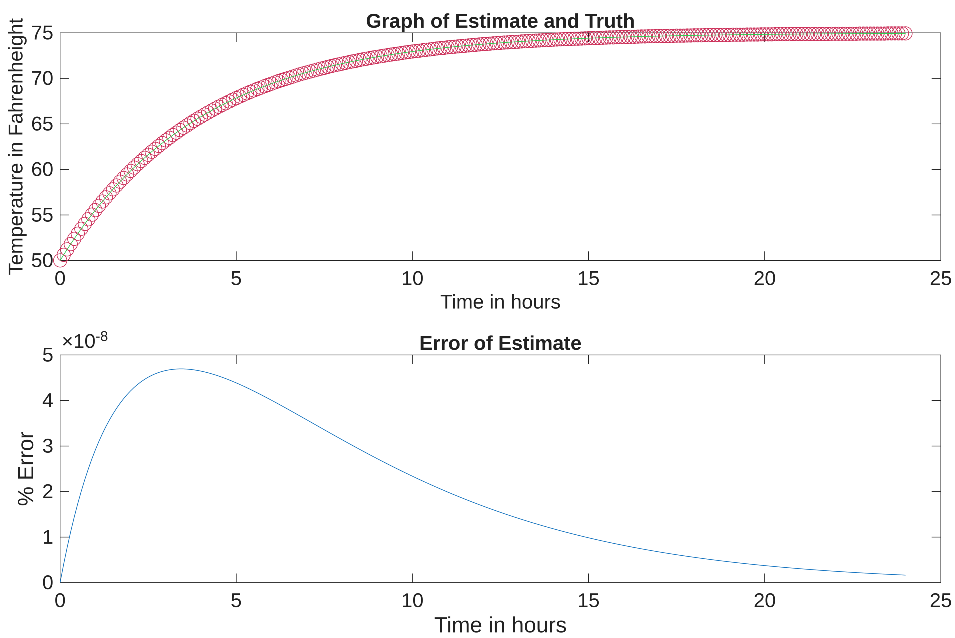
Error Matrix

```
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
```

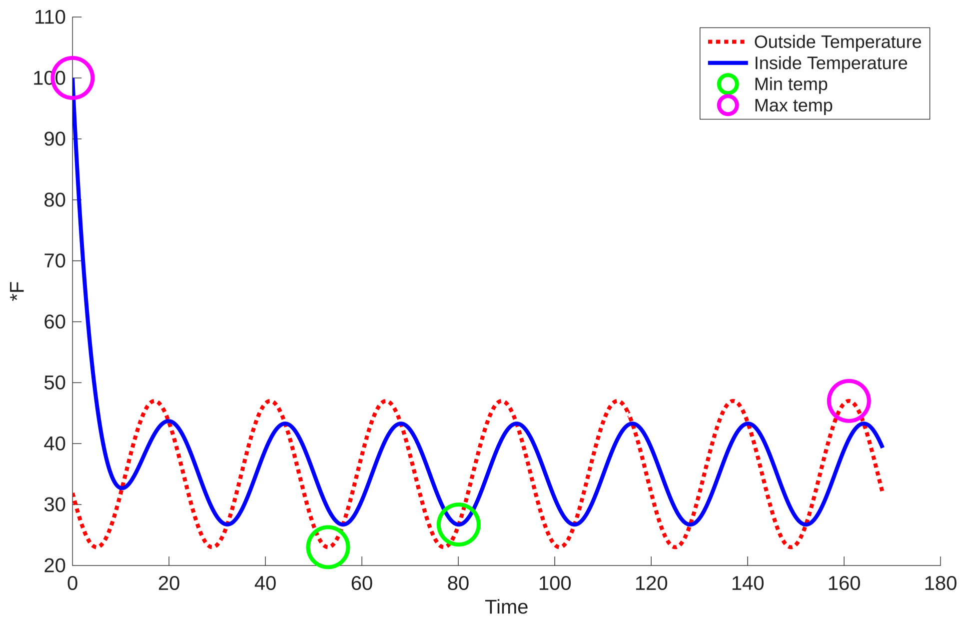
```
fprintf(formatspec2, Mspoints)
```

```
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
```



```
% 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);
```

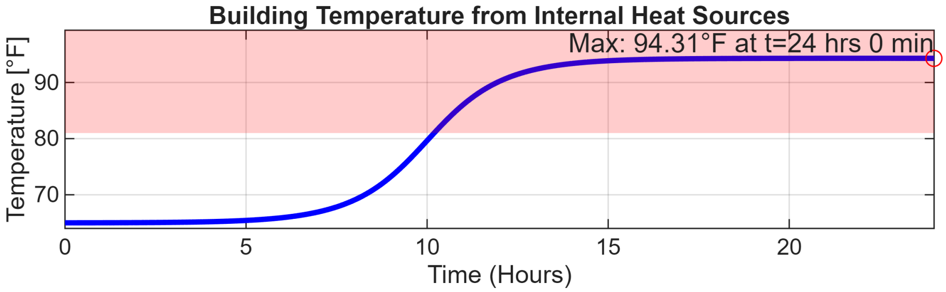
```
Maximum temperature: 94.31 °F
```

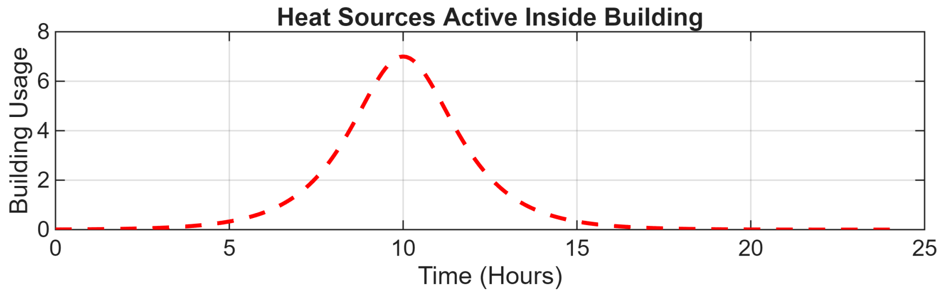
```
fprintf('Time of maximum temperature: %.2f hours\n', time_max_T);
```

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

```

% 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;
```

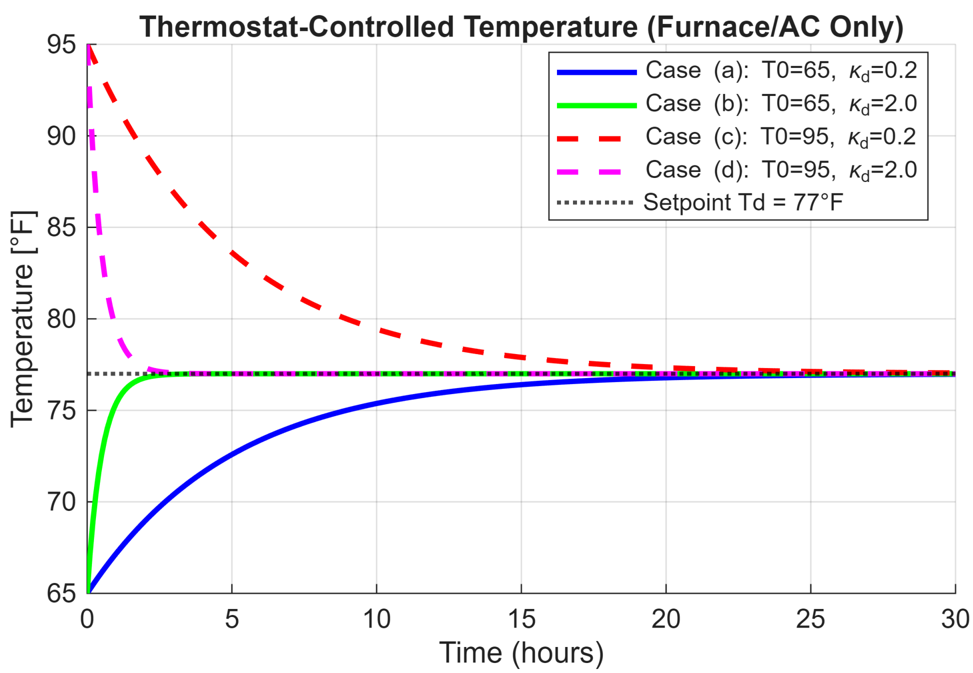





```
% --- Helper function for sech ---
function y = sech(x)
    y = 1 ./ cosh(x);
end
```

```matlab
% 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);
```

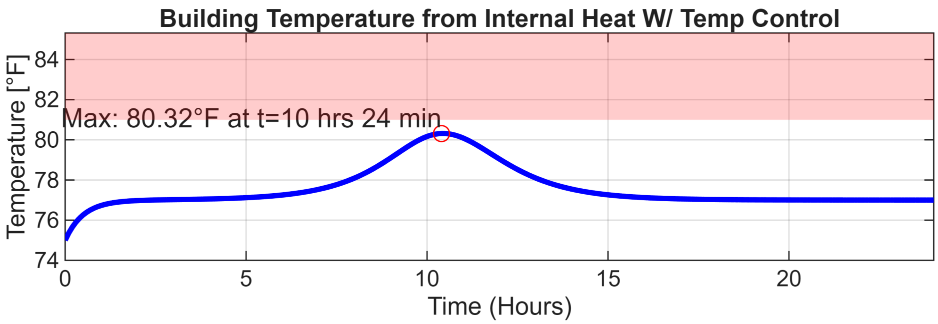
```
        T(j+1) = Tj + (1/6)*(k1 + 2*k2 + 2*k3 + k4);
    end

    % 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
```

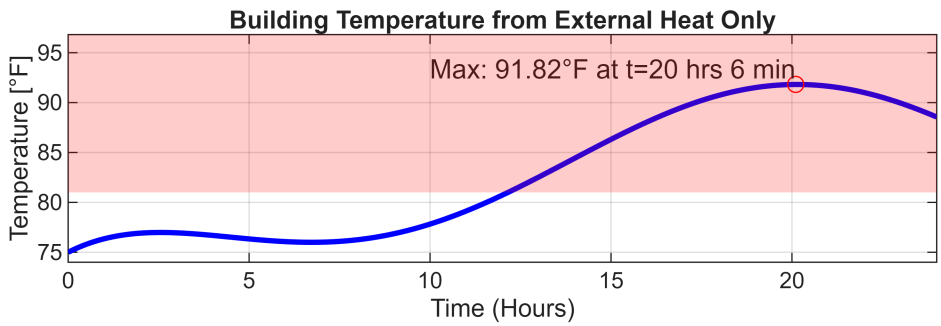
```
% Plot the result
figure;
y_max = max_T + 5;

% 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;
```

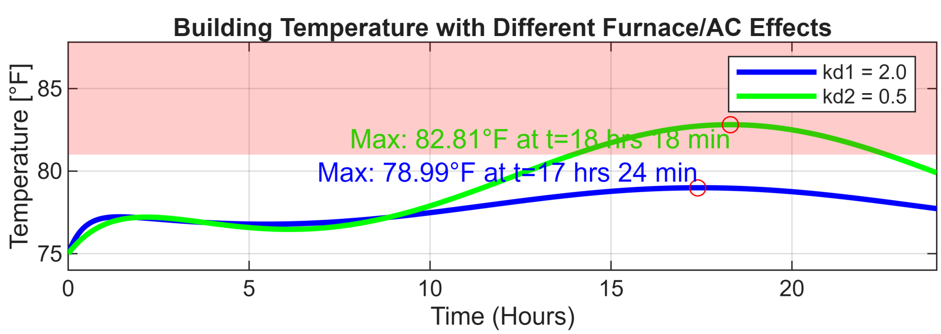


5.3

```
% 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
```

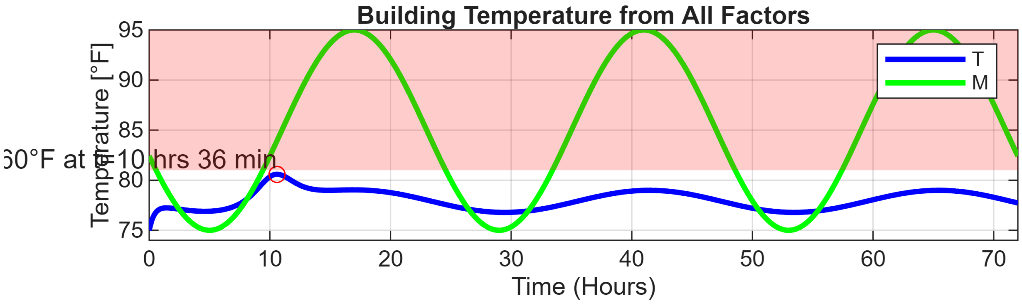
```
disp("The equipment was exposed to damaging temperature for " + t_dmg_min + "
minutes.");
```

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