Modeling Passive Solar House in Massachusetts

By Gati Aher and Zachary Sherman

All measurements are in metric

Design Constraints

- small house

```
min_floor_space = 9.2903; % m^2
max_floor_space = 37.1612; % m^2
```

- reasonably comfortable in winter in Boston climate

```
min_indoor_air_temp = 17; % deg Celsius
max_indoor_air_temp = 25; % deg Celsius
```

- no direct sunlight through south-facing windows at noon in summer

Minimal Implementation

Define Materials

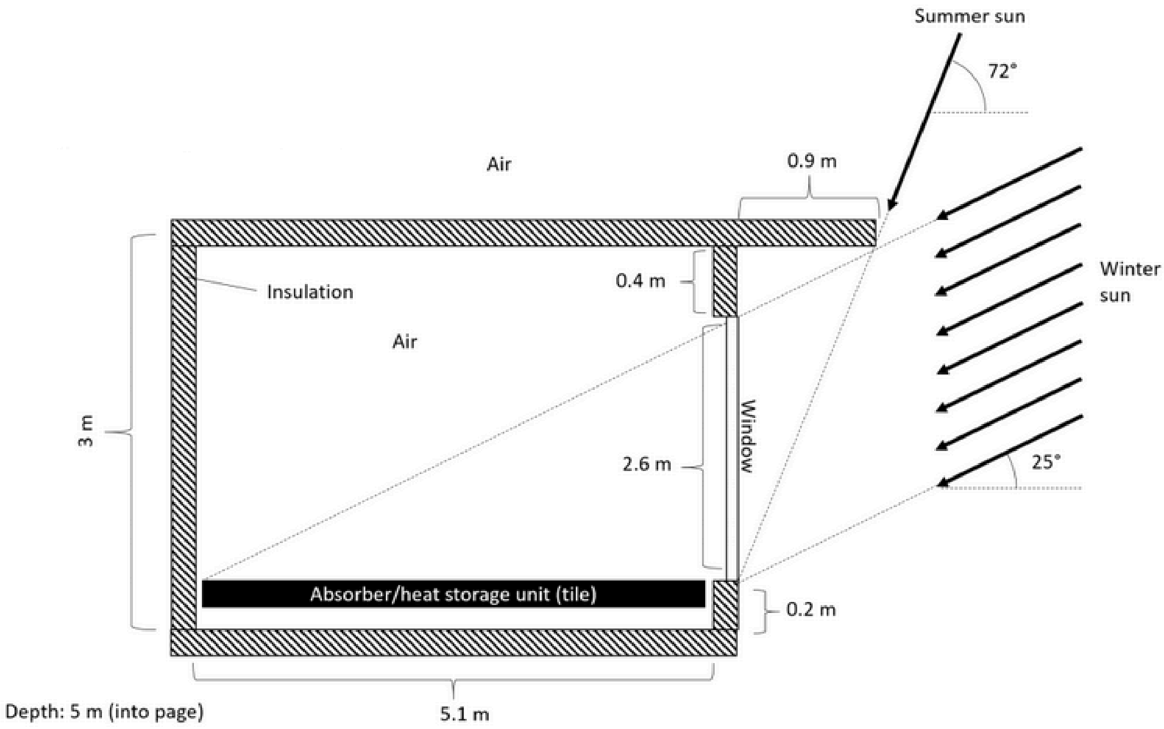
```
tile_storage = PureThermalStorage("recycled ceramic tile", 800, 3000)
```

```
tile_storage = 
  PureThermalStorage with properties:

                LongName: "recycled ceramic tile"
    SpecificHeatCapacity: 800
                 Density: 3000
```

```
fiberglass_insulation = SolidPureThermalResistance("fiberglass", 0.04)
```

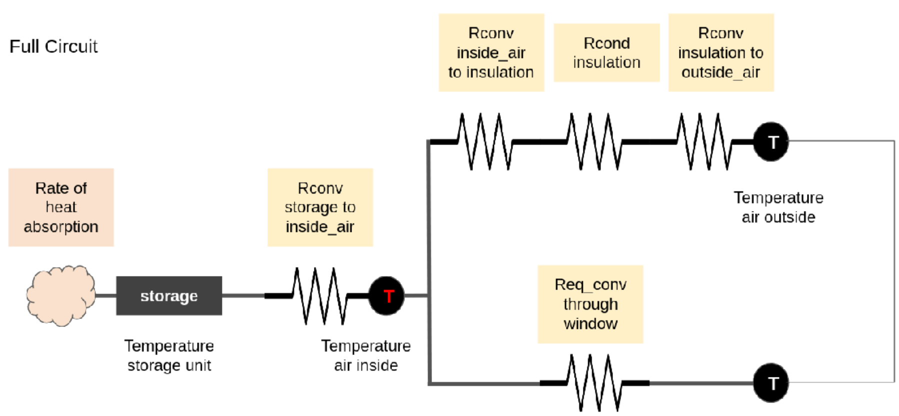
```
fiberglass_insulation = 
  SolidPureThermalResistance with properties:

               LongName: "fiberglass"
    ThermalConductivity: 0.0400
```

Define Base Model



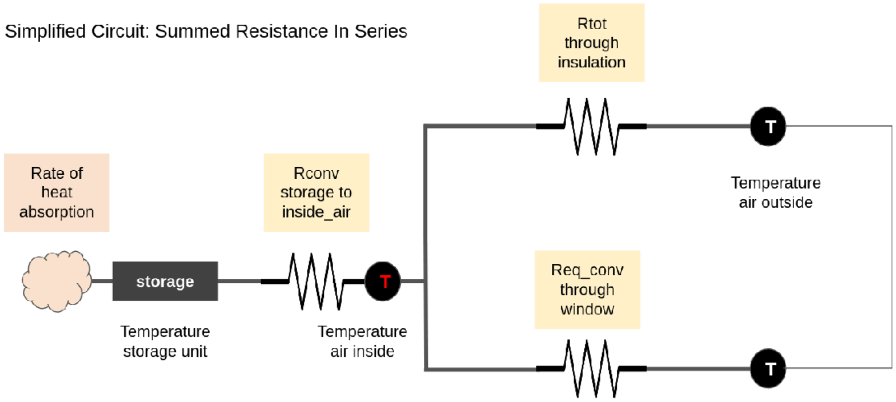
Define Base Measurements

Draw Circuit Diagram to Understand Resistance Network



Sum resistance in series: $R_{\text{totseries}} = R_1 + R_2 + \ \ldots + R_n$

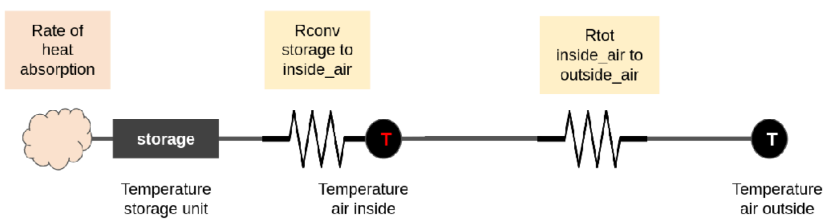
Simplified Circuit: Summed Resistance In Series



Sum the two resistances in parallel

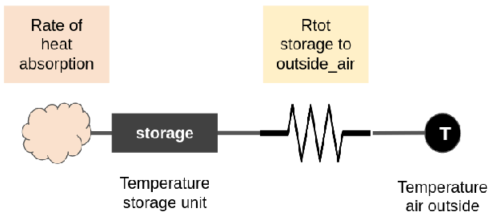
$$R_{\text{totparallel}} = \frac{R_1 R_2}{R_1 + R_2}$$

Simplified Circuit: Summed Resistance In Parallel



Sum in series to get resistance network in most simplified form

Simplified Circuit: Summed Resistance In Series



Find a solution for the house's temperature over the course of serveral days

1. Write an ordinary differential equation to model the heat storage's rate of temperature change over time.

$$C_{\text{storage}} \frac{\mathrm{dT}_{\text{storage}}}{\mathrm{dt}} = \frac{\mathrm{dU}_{\text{storage}}}{\mathrm{dt}} = Q'_{\text{intostorage}} - Q'_{\text{ourofstorage}} = Q'_{\text{in}} - Q'_{\text{out}}$$

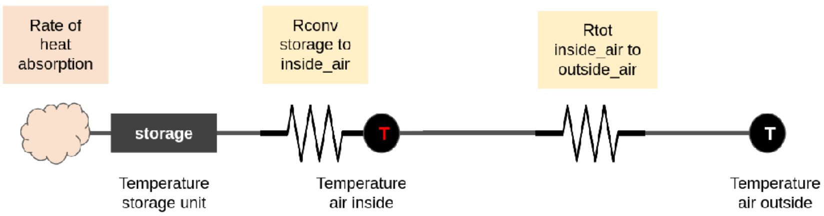
$$Q'_{\text{out}} = \frac{T_{\text{storage}} - T_{\text{outsideair}}}{R_{\text{totstoragetooutsideair}}}$$

final ODE for $\frac{\mathrm{dT}_{\text{storage}}}{\mathrm{dt}}$ is y'

$$y' = \frac{\left(Q'_{\text{in}} - \frac{y - T_{\text{outsideair}}}{R_{\text{totstoragetooutsideair}}}\right)}{C_{\text{storage}}}$$

2. Solve for $T_{\text{insideair}}$

Simplified Circuit: Summed Resistance In Parallel



Heat flow through a circuit is the same for resistors in series.

$$\frac{T_{\mathrm{storage}} - T_{\mathrm{outsideair}}}{R_{\mathrm{totstoragetooutsideair}}} = \frac{T_{\mathrm{insideair}} - T_{\mathrm{outsideair}}}{R_{\mathrm{totinsideairtooutsideair}}}$$

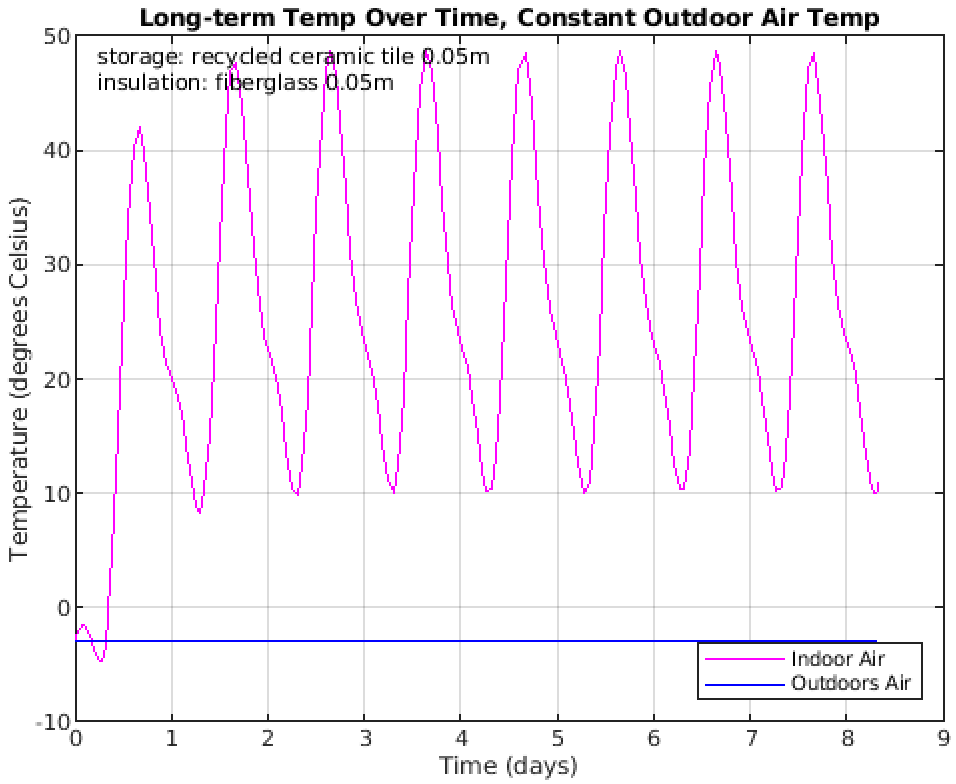
By rearranging

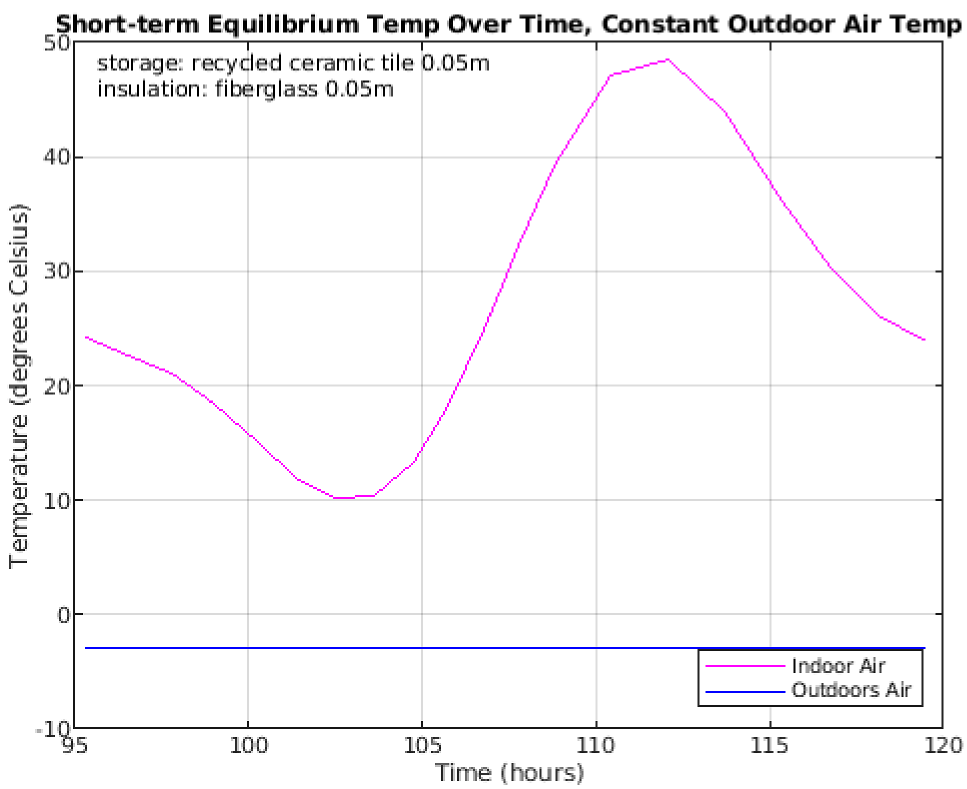
$$T_{\mathrm{insideair}} = T_{\mathrm{outsideair}} + R_{\mathrm{totinsideairtooutsideair}} \frac{T_{\mathrm{storage}} - T_{\mathrm{outsideair}}}{R_{\mathrm{totstoragetooutsideair}}}$$

Model With Constant Outside Air Temperature

```
ODEHelper(200, "constant", 0.05, tile_storage, 0.05, fiberglass_insulation, true)
```

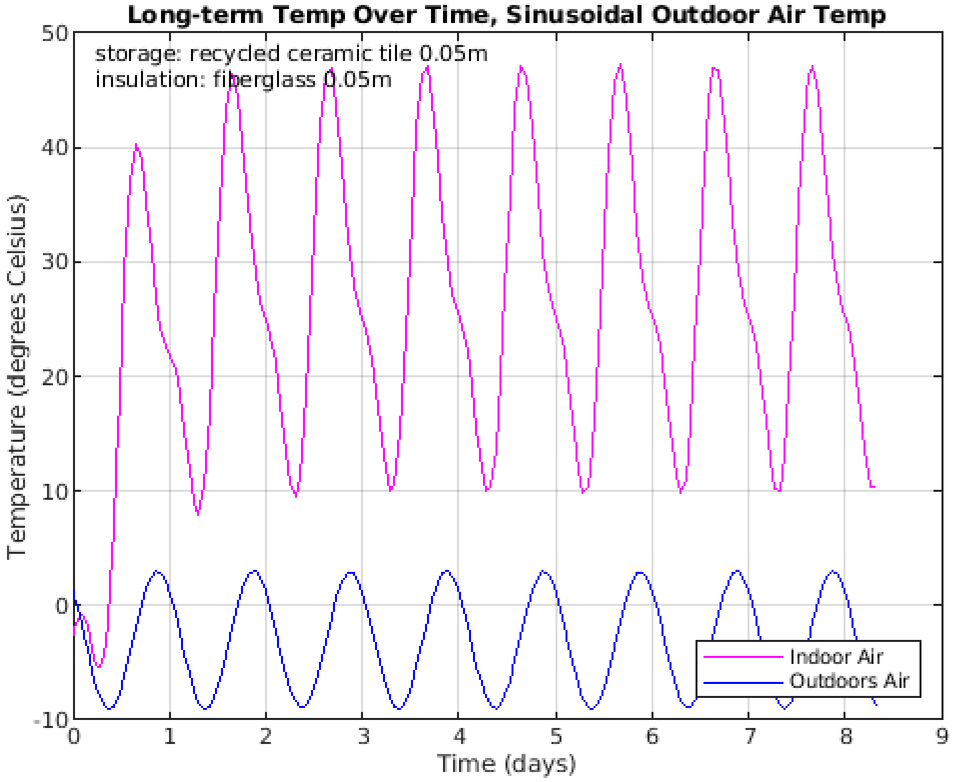
```
Warning: MATLAB has disabled some advanced graphics rendering features by switching to software
 OpenGL. For more information, click here.
```

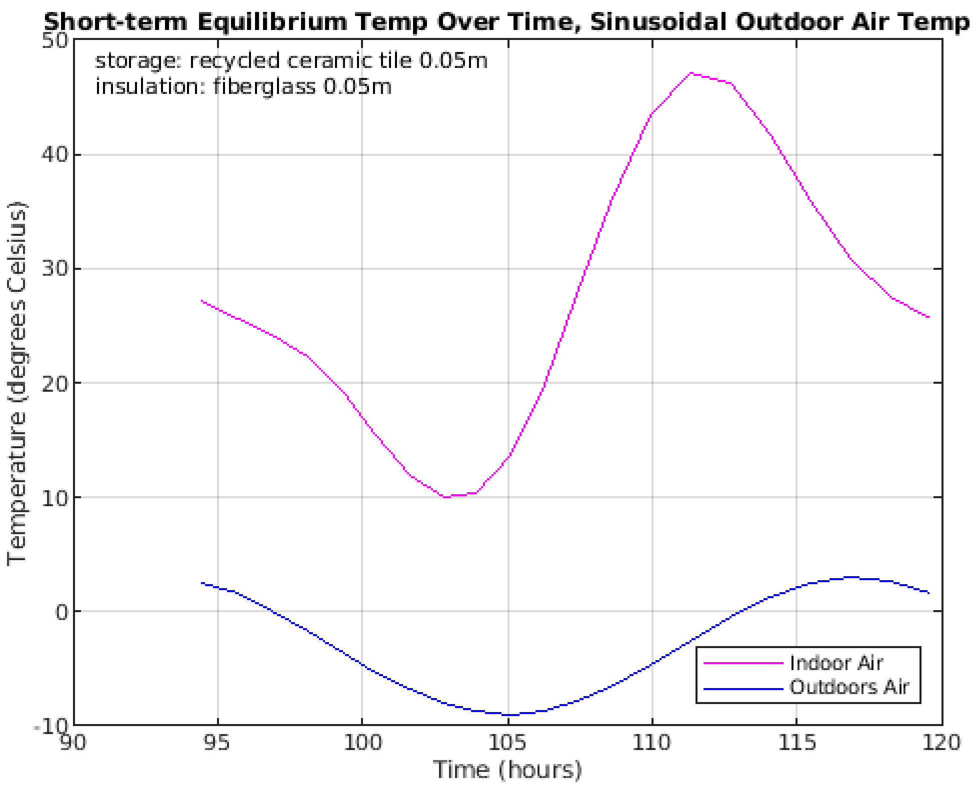




For Model With Sinusoidal Outside Air Temperature

```
ODEHelper(200, "sinusoidal", 0.05, tile_storage, 0.05, fiberglass_insulation, true)
```





Optimization

Find best thickness for storage material and insulation

Compare different materials

Goal: temperature fluctuation should be between 17-25 degrees Celcius

Fiddle with:

- Number of hours to run simulation
- storage material thickness
- insulation material thickness
- material type

```
masonry_storage = PureThermalStorage("masonry", 1000, 2300)
```

```
masonry_storage = 
  PureThermalStorage with properties:

              LongName: "masonry"
    SpecificHeatCapacity: 1000
               Density: 2300
```

```
water_storage = PureThermalStorage("water", 4200, 1000)
```

```
water_storage = 
  PureThermalStorage with properties:

              LongName: "water"
    SpecificHeatCapacity: 4200
               Density: 1000
```

```
carpet_insulation = SolidPureThermalResistance("carpet", 0.05)
```

```
carpet_insulation = 
  SolidPureThermalResistance with properties:

             LongName: "carpet"
    ThermalConductivity: 0.0500
```

```
gypsum_plaster_insulation = SolidPureThermalResistance("gypsum plaster", 0.5)
```

```
gypsum_plaster_insulation = 
  SolidPureThermalResistance with properties:

             LongName: "gypsum plaster"
    ThermalConductivity: 0.5000
```

```
aircrete_insulation = SolidPureThermalResistance("aircrete", 0.15)
```

```
aircrete_insulation = 
  SolidPureThermalResistance with properties:

             LongName: "aircrete"
    ThermalConductivity: 0.1500
```

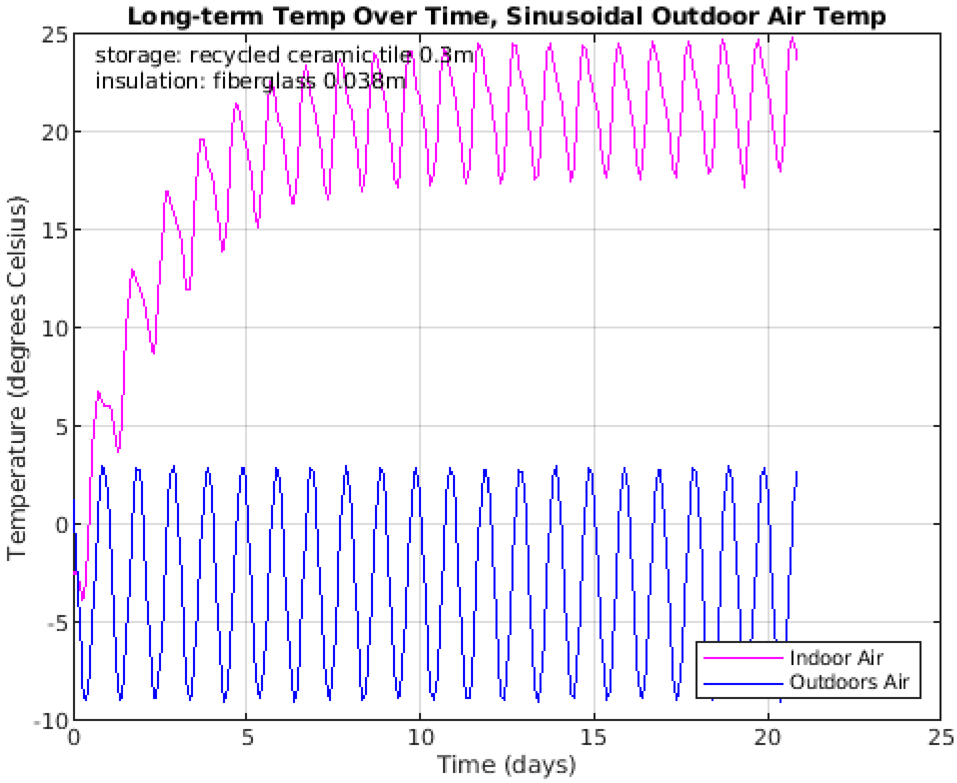
Try out 12 Combinations and compare required thicknesses

- tile + fiberglass (base)
- tile + carpet

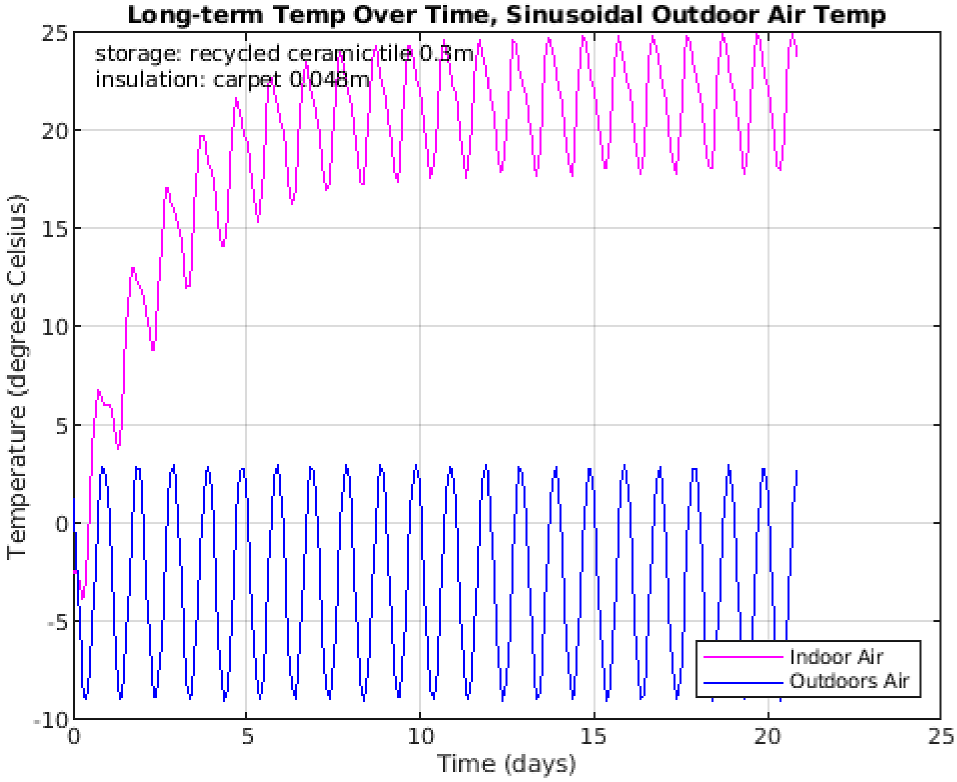
- tile + gypsum plaster
- tile + aircrete
- masonry + fiberglass (base)
- masonry + carpet
- masonry + gypsum plaster
- masonry + aircrete
- water + fiberglass (base)
- water + carpet
- water + gypsum plaster
- water + aircrete

Tests

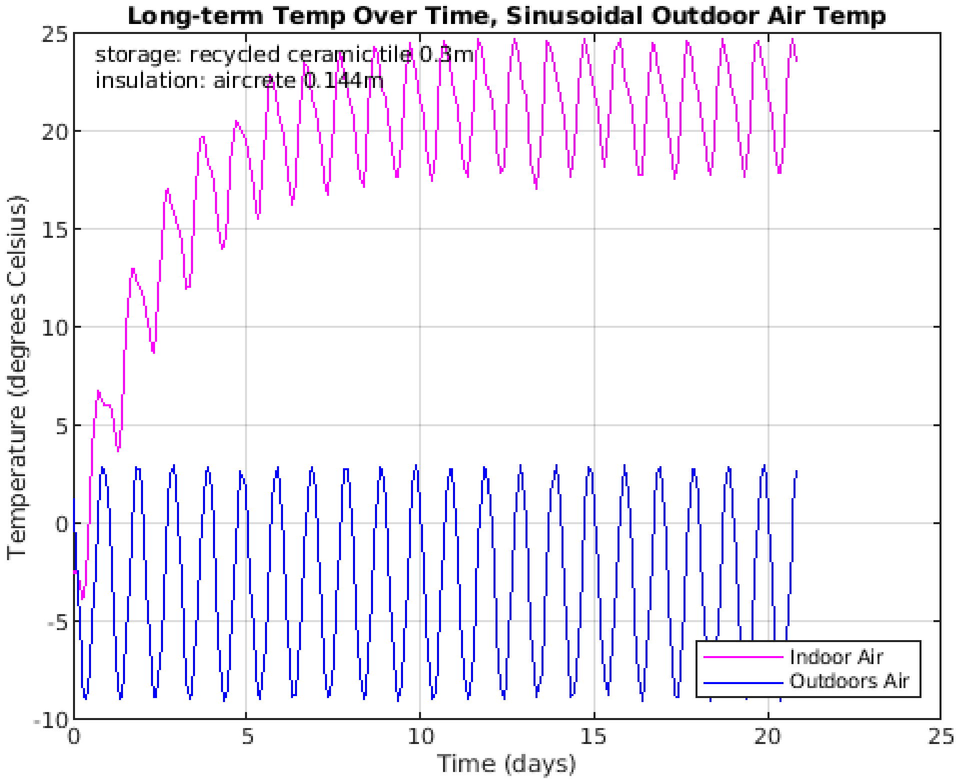
```
tile_storage_thickness = 0.30;
masonry_storage_thickness = 0.30;
water_storage_thickness = 0.15;

fiberglass_insulation_thickness = 0.038;
carpet_insulation_thickness = 0.048;
aircrete_insulation_thickness = 0.144;
gypsum_plaster_insulation_thickness = 0.485;

ODEHelper(500, "sinusoidal", tile_storage_thickness, tile_storage, fiberglass_insulatio
```



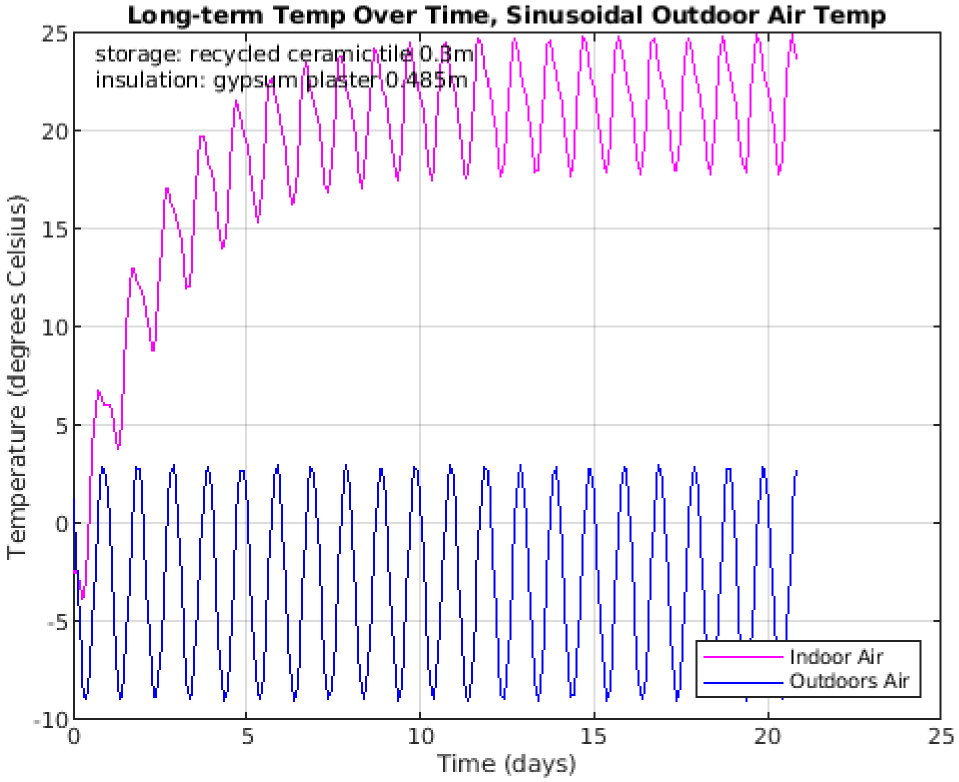

```
ODEHelper(500, "sinusoidal", tile_storage_thickness, tile_storage, carpet_insulation_th
```




```
ODEHelper(500, "sinusoidal", tile_storage_thickness, tile_storage, aircrete_insulation_
```

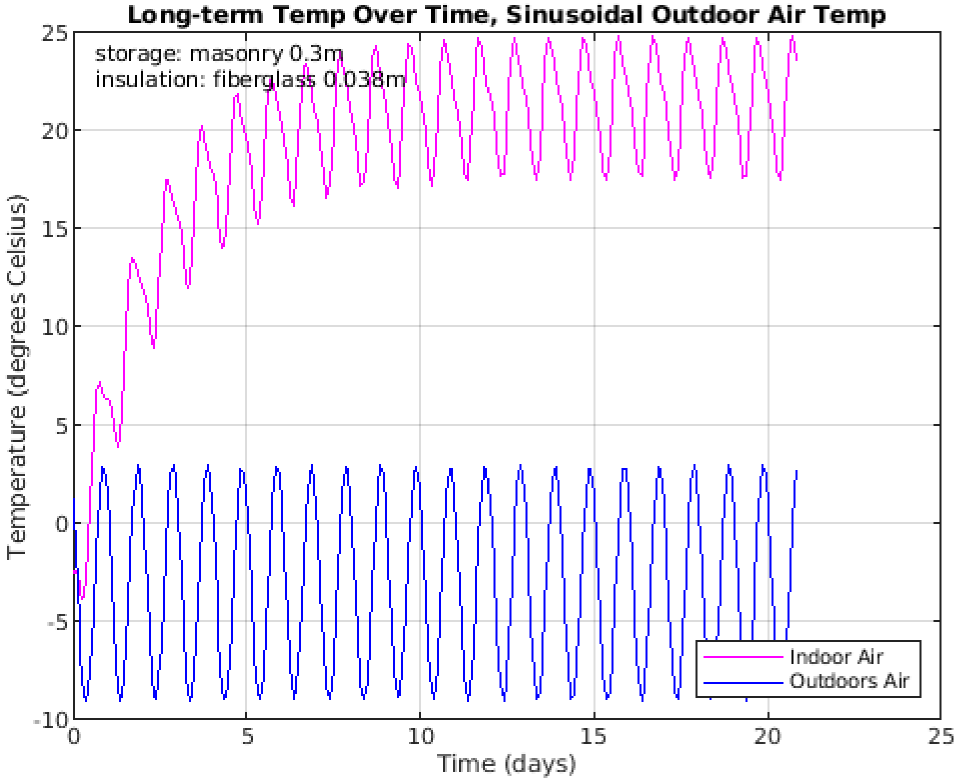


```
ODEHelper(500, "sinusoidal", tile_storage_thickness, tile_storage, gypsum_plaster_insul
```



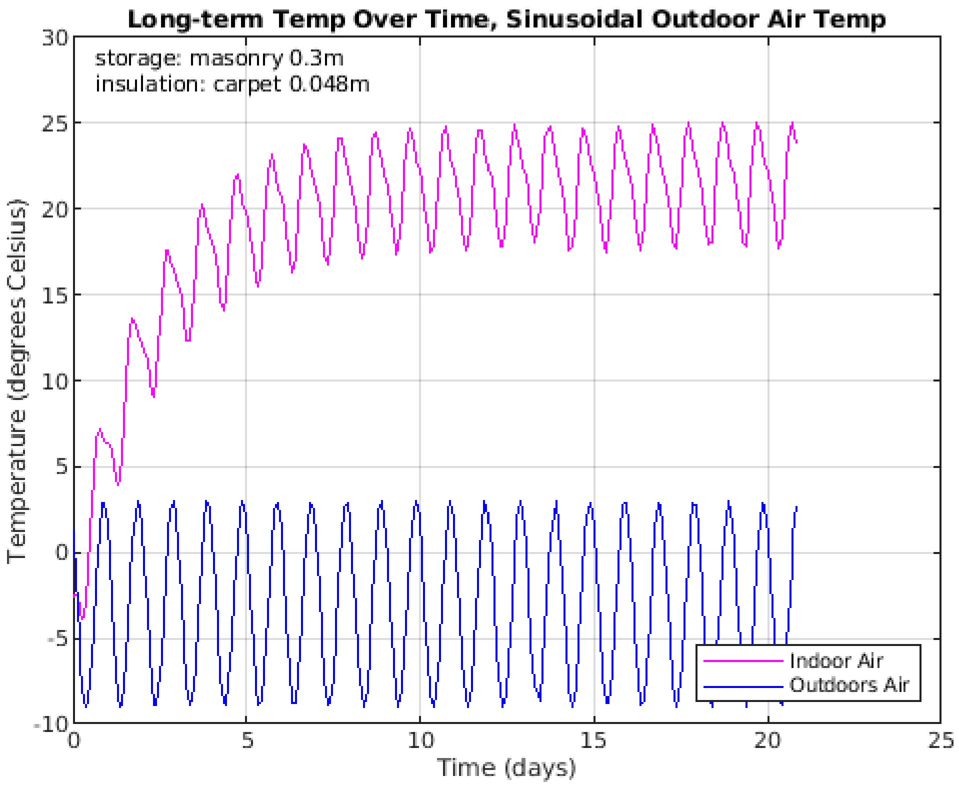
- masonry + fiberglass

```
ODEHelper(500, "sinusoidal", masonry_storage_thickness, masonry_storage, fiberglass_ins
```



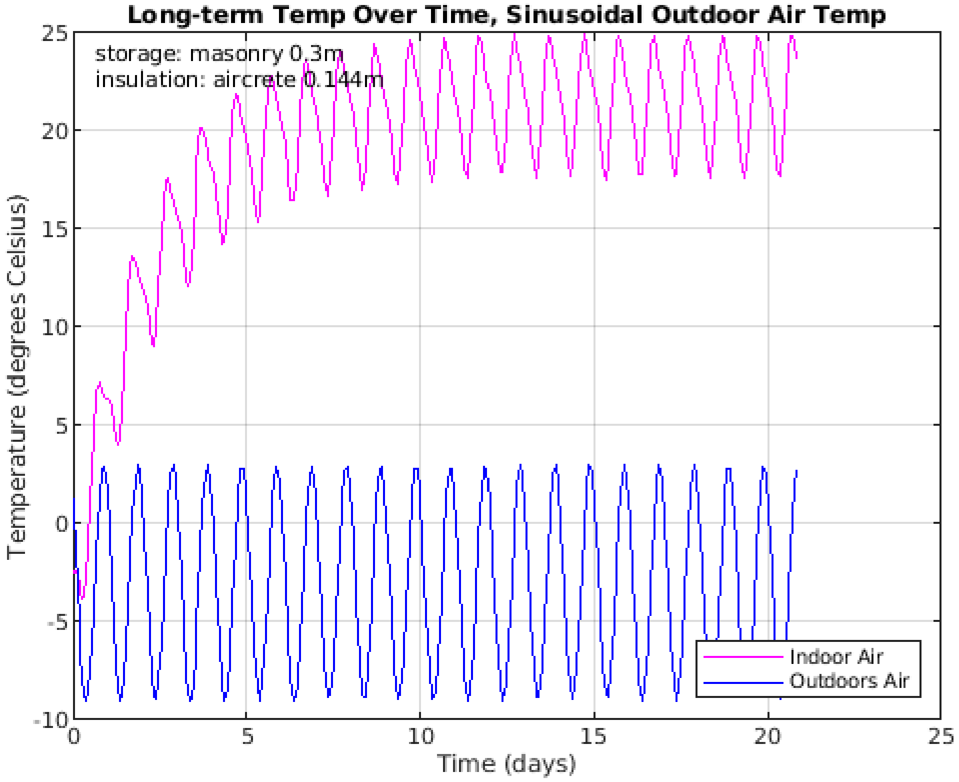
- masonry + carpet

```
ODEHelper(500, "sinusoidal", masonry_storage_thickness, masonry_storage, carpet_insulat
```



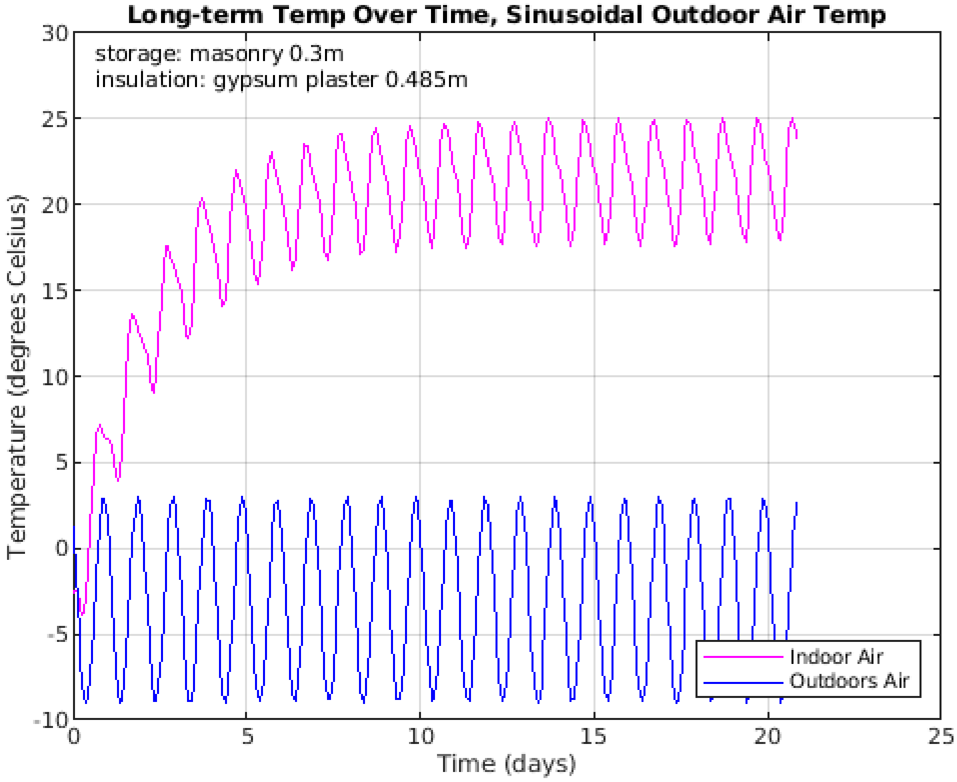
- masonry + aircrete

```
ODEHelper(500, "sinusoidal", masonry_storage_thickness, masonry_storage, aircrete_insul
```



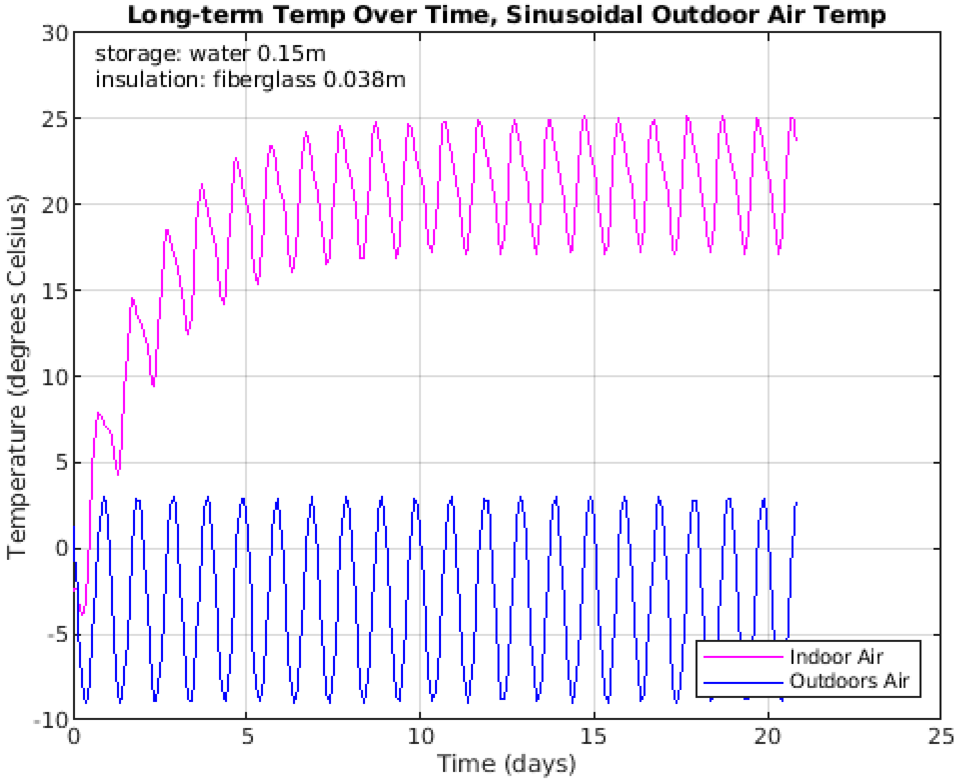
- masonry + gypsum plaster

```
ODEHelper(500, "sinusoidal", masonry_storage_thickness, masonry_storage, gypsum_plaster
```



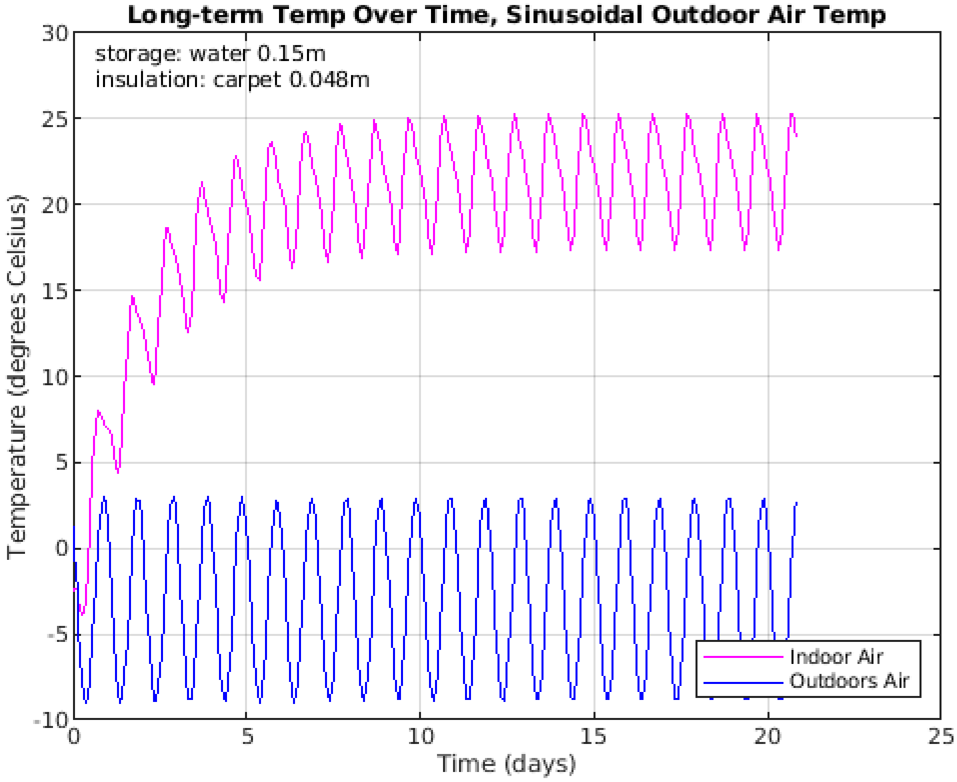
- water + fiberglass

```
ODEHelper(500, "sinusoidal", water_storage_thickness, water_storage, fiberglass_insulat
```



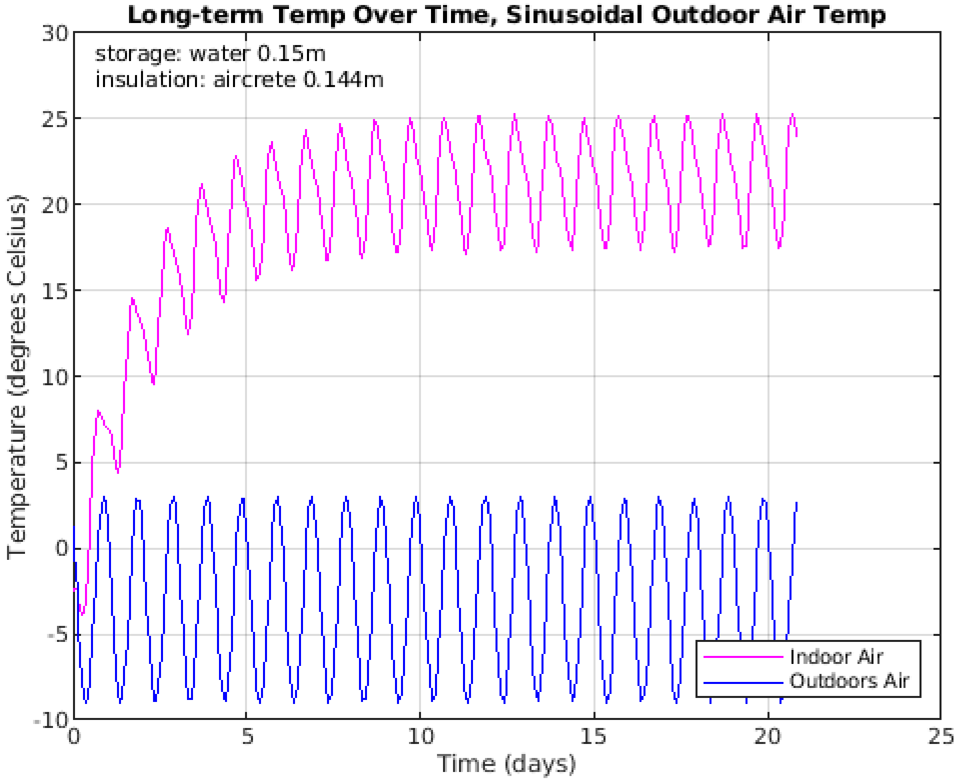
- water + carpet

```
ODEHelper(500, "sinusoidal", water_storage_thickness, water_storage, carpet_insulation_
```



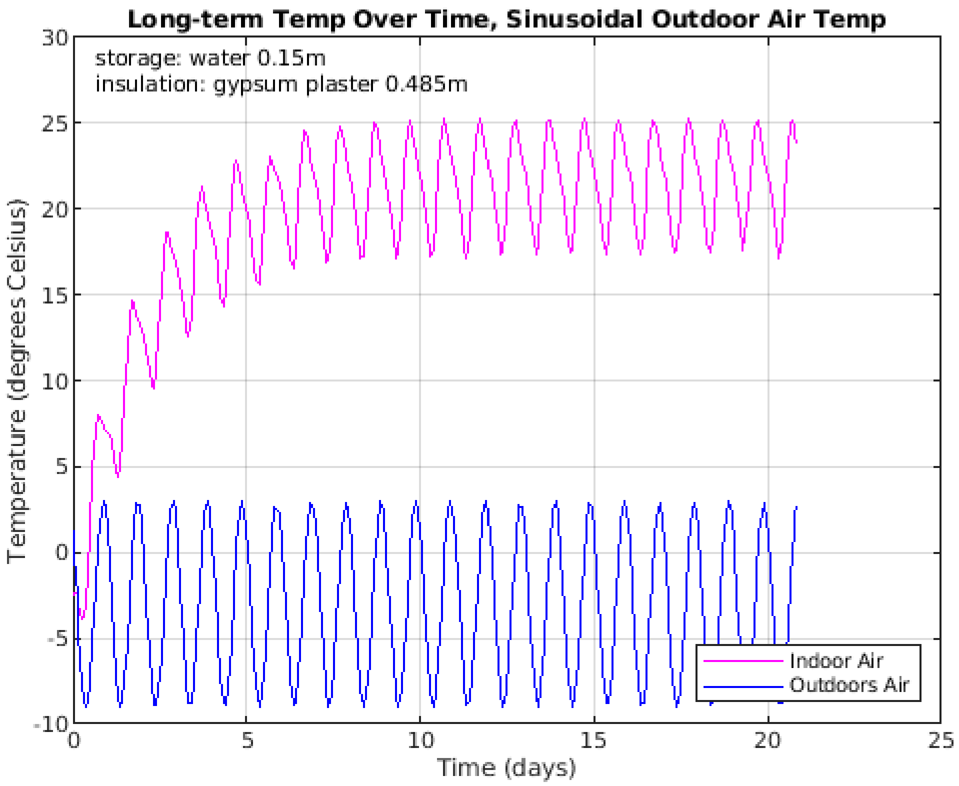
- water + aircrete

```
ODEHelper(500, "sinusoidal", water_storage_thickness, water_storage, 0.144, aircrete_in
```



- water + gypsum plaster

```
ODEHelper(500, "sinusoidal", water_storage_thickness, water_storage, gypsum_plaster_ins
```



Optimization Conclusions

```
% table of ideal thermal storage thicknesses
S.Name = [tile_storage.LongName; masonry_storage.LongName; water_storage.LongName];
S.SpecificHeatCapacity = [tile_storage.SpecificHeatCapacity; masonry_storage.SpecificHe
S.Density = [tile_storage.Density; masonry_storage.Density; water_storage.Density];
S.IdealThickness = [tile_storage_thickness; masonry_storage_thickness; water_storage_th
storage_table = struct2table(S)
```

storage_table = 3×5 table

	Names	SpecificHeatCapacity	Density	IdealThickness	Name
1	"recycled ...	800	3000	0.3000	"recycled ...
2	"masonry"	1000	2300	0.3000	"masonry"
3	"water"	4200	1000	0.1500	"water"

```
% table of idea thermal insulation thicknesses
I.Name = [fiberglass_insulation.LongName; carpet_insulation.LongName; aircrete_insulati
I.ThermalConductivity = [fiberglass_insulation.ThermalConductivity; carpet_insulation.T
I.IdealThickness = [fiberglass_insulation_thickness; carpet_insulation_thickness; aircr
insulation_table = struct2table(I)
```

insulation_table = 4×3 table

	Name	ThermalConductivity	IdealThickness
1	"fiberglass"	0.0400	0.0380

	Name	ThermalConductivity	IdealThickness
2	"carpet"	0.0500	0.0480
3	"aircrete"	0.1500	0.1440
4	"gypsum pl...	0.5000	0.4850

Appraisal

The base model predicts a house that is slightly too warm, with an equilibrium air temperature of approximately 30 °C. Although in theory, a house that is too hot is just as uncomfortable as a house that is too cold, in reality, cooling down a passive solar house is a fairly trivial task. A smaller window could be used, the absorber's effectiveness or surface area could be reduced, and/or the thermal mass could be smaller. In each of these cases, this just means making the house slightly less efficient. For our final model, we will adjust these parameters as necessary to allow for a comfortable indoor air temperature.

Model of Q'_{in} Heat Absorbed by the Absorber/Heat Storage Unit

To simplify the model we make the following assumptions:

- Heat storage unit is at a spatially uniform temperature
- All solar radiation hitting the window is absorbed by the heat storage unit

With these assumptions Q'_{in} can be calculated as $Q'_{\mathrm{in}} = q\,A_{\mathrm{window}}$ the product of the normal solar flux, q(t), and the window area, A_{window}.

```
function nsf = q(t)
    %Input:
    %   t: (single row-vector of numbers) time in seconds
    %Output:
    %   nsf: (single row-vector of numbers) normal solar flux through south-facing
    %   window in Massachusetts. Units of W/m^2
    nsf = -361 * cos(t.*(pi/(12*3600))) + 244 * cos(t.*(pi/(6*3600))) + 210;
end

function heatAbsorbed = Qinprime(t,Awindow)
    %Input:
    %   t: (single row-vector of numbers) time in seconds
    %   Awindow: (number) area of window in m^2
    %Output:
    %   heatAbsorbed: (single row-vector of numbers) rate of heat absorbed by the
    %   absorber/heat storage. Units in Watts
    heatAbsorbed = q(t) .* Awindow;
end
```

Surface Area Helper Functions

```
function SA = surfaceAreaRectPrism(depth, length, thickness)
    %Input:
```

```
    %   depth: (number) depth of model house in meters
    %   length: (number) length of model house in meters
    %   height: (number) height of model house in meters
    %Output:
    %   SA: (number) surface area of inside of model house. Units in m^2
    SA = 2 * (depth * length + length * thickness + thickness * depth);
end

function SA = innerSurfaceAreaInsulation(window_area, depth, length, height, ~)
    %Return solid surface area that inner air convection occurs on
    %Input:
    %   Awindow: (number) area of window in m^2
    %   depth: (number) depth of model house in meters
    %   length: (number) length of model house in meters
    %   height: (number) height of model house in meters
    %Output:
    %   SA: (number) area of model house in contact with inside air. Units in m^2
    sa_total = surfaceAreaRectPrism(depth, length, height);
    SA = sa_total - window_area;
end

function SA = outerSurfaceAreaInsulation(window_area, depth, length, height, thickness)
    %Return solid surface area that outer air convection occurs on
    %Input:
    %   Awindow: (number) area of window in m^2
    %   depth: (number) depth of model house in meters
    %   length: (number) length of model house in meters
    %   height: (number) height of model house in meters
    %   thickness: (number) wall thickness in model house in meters
    %Output:
    %   SA: (number) area of model house in contact with outside air. Units in m^2
    sa_total = surfaceAreaRectPrism(depth + 2*thickness, length + 2*thickness, height +
    sa = sa_total - window_area;
    % subtract area covered by overhang join
    SA = sa - ((depth + 2*thickness) * thickness);
end
```

Plotting Helper

```
function [] = plotHelper(annotation, plot_title, t, t_label, ~, T_inside_air, T_outside
    % Plots Heat Storage, Indoor Air, Outdoors Air over time
    figure
    plot(t, T_inside_air, 'm')
    hold on
    plot(t, T_outside_air, 'b')
    grid on
    xlabel(t_label)
    ylabel('Temperature (degrees Celsius)')
    title(plot_title)
    legend('Indoor Air', 'Outdoors Air', 'Location', 'southeast')

    % for easy identification
```

```
    text(0.025,0.95,annotation,'Units','normalized','FontSize',8)

    ax = gca;
    ax.FontSize = 8;
    hold off
end

function [] = plotHelperWithTStorage(annotation, plot_title, t, t_label, T_storage, T_i
    % Plots Heat Storage, Indoor Air, Outdoors Air over time
    figure
    plot(t, T_storage, 'r')
    hold on
    plot(t, T_inside_air, 'm')
    plot(t, T_outside_air, 'b')
    grid on
    xlabel(t_label)
    ylabel('Temperature (degrees Celsius)')
    title(plot_title)
    legend('Indoor Air', 'Outdoors Air', 'Location', 'southeast')

    % for easy identification
    text(0.025,0.95,annotation,'Units','normalized','FontSize',8)

    ax = gca;
    ax.FontSize = 8;
    hold off
end
```

ODE Helper

Model Base House

```
function [C_storage, Rtot_inside_air_to_outside_air, Rtot_storage_to_outside_air, b_win

    % define material properties
    air_indoors = LiquidPureThermalResistance('air indoors', 15);
    air_outdoors = LiquidPureThermalResistance('air outdoors', 30);
    % use equivalent heat transfer coefficient (heq) of double-paned window
    window = LiquidPureThermalResistance('double-paned window', 0.7);

    % define model dimensions
    b_model_depth = 5;
    b_model_length = 5.1;
    b_model_height = 3;

    % window
    b_window_height = 2.4;
    b_window_area = b_window_height * b_model_depth;

    % storage
    b_storage_volume = b_model_depth * b_model_length .* storage_thickness;
    % convection surface area
    b_storage_surface_area = surfaceAreaRectPrism(b_model_depth, b_model_length, storag
```

```
    % insulation
    % convection surface area
    b_insulation_inner_surface_area = innerSurfaceAreaInsulation(b_window_area, b_model
    b_insulation_outer_surface_area = outerSurfaceAreaInsulation(b_window_area, b_model

    % calculate resistances
    Rconv_storage_to_inside_air = Rconv(air_indoors, b_storage_surface_area);
    Rconv_inside_air_to_insulation = Rconv(air_indoors, b_insulation_inner_surface_area
    Rcond_insulation = Rcond(insulation_material, b_insulation_inner_surface_area, insu
    Rconv_insulation_to_outside_air = Rconv(air_outdoors, b_insulation_outer_surface_ar
    Req_conv_through_window = Rconv(window, b_window_area);

    % simplify resistance network
    Rtot_through_insulation = Rconv_inside_air_to_insulation + Rcond_insulation + Rconv

    % final return values
    C_storage = C(storage_material, b_storage_volume);
    Rtot_inside_air_to_outside_air = (Rtot_through_insulation * Req_conv_through_window
    Rtot_storage_to_outside_air = Rconv_storage_to_inside_air + Rtot_inside_air_to_outs
end
```

ODE Solver

```
% function [] = ODEHelper(t_end, air_temp_function, air_temp_name, storage_thickness, s
function [] = ODEHelper(t_end, air_temp_outside_model_type, storage_thickness, storage_

    [air_temp_outside_name, air_temp_outside_function] = air_temp_outside_model(air_tem

    [C_storage, Rtot_inside_air_to_outside_air, Rtot_storage_to_outside_air, b_window_a
    yprime = @(t, y) (Qinprime(t, b_window_area) - (y - air_temp_outside_function(t)) /

    % convert T_end from hours to seconds
    t_end_sec = t_end * 60 * 60;

    % ODE constants
    T_storage_0 = -3;
    tspan = [1 t_end_sec];
    % solve ODE for heat storage temp
    [t, y] = ode45(yprime, tspan, T_storage_0);

    % solve for air temp
    T_outside_air = air_temp_outside_function(t);
    T_inside_air = T_outside_air + Rtot_inside_air_to_outside_air * (y - T_outside_air)

    %convert t from seconds to days for nice plots
    t_days = t / (60 * 60 * 24);
    % make annotation string for nice plots
    annotation = {"storage: " + storage_material.LongName + " " + storage_thickness + "
        , "insulation: " + insulation_material.LongName + " " + insulation_thickness +

    plotHelper(annotation, "Long-term Temp Over Time, " + air_temp_outside_name, t_days
```

```
    if (nargin > 6 & plot_short_term)
        %plot temperature over course of one day at equilibrium
        %convert t from seconds to hours for nice plots
        t_hours = t / (60 * 60);
        % find indexes of 24 hour range in equilibrium time
        % eyeball check that equilibrium was reached at least 1/2 of the way through th
        s = ceil(size(t_hours, 1) * (1/2));
        e = find(t_hours>t_hours(s)+24);
        e = e(1); % end 1-day period in hours
        plotHelper(annotation, "Short-term Equilibrium Temp Over Time, " + air_temp_out
    end

end
```