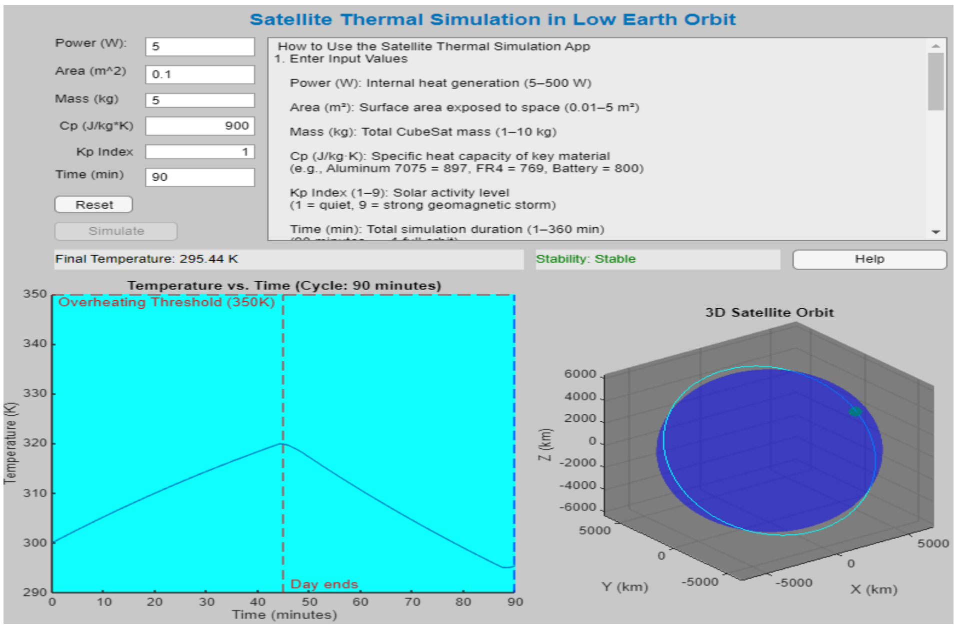
MATLAB Thermal Analysis of Small Satellite



SPRING 2025

Engin-170-2473

5/23/2025

Jesse Hart

1. Introduction and Motivation

Briefly tell us why you pick this topic and the main function of your App.

In introductory physics class 3 I enjoyed learning about thermodynamics. I learned how to solve different equations to model heat transfer in rods.Outside of class my interest in space resulted in reading about CubeSats. CubeSats can be an exciting real world application of thermodynamics. This led to the creation of this app that applies a simplified thermodynamic model for a CubeSat in orbit, similar to the one I learned about in introductory physics 3, where the app simulates the thermal behavior of CubeSats as they orbit Earth.

2. Background

Describe the basic theory, equation and/or principles related to this project. List the
main function used in your code.

This project models the thermal behavior of a satellite in Low Earth Orbit (LEO) using principles of energy balance from thermodynamics. As a CubeSat orbits Earth, it experiences alternating periods of sunlight and eclipse, resulting in different heat input. Understanding how internal heat and radiative cooling affect the satellite's temperature is crucial to ensure it remains within safe operating limits.

The core equation used in this simulation is a simplified first order differential equation derived from conservation of energy:

dT/dt = (Qin - Quote) / (M * Cp)

Where:

- T is temperature (K) in Kelvins
- Qin is the total heat input (internal power + solar heating when in sunlight)
- Quot = sigma * A * T^4
- M is the satellite mass (kg)
- Cp is the specific heat capacity (J/kg*K)
- Sigma is the Stefan-Boltzman constant (5.67 * 10^-8 W/m^2*K^4)

This equation is solved using MATLAB’s ode15s solver, which is suitable for the large data in this simulation.

Main Functions used in the Code

- ode15s() - Solves the temperature differential equation over time.

- plot()/plot3() - Visualizes temperature vs. time, and the satellites 3d orbital path.
- set() - Updates the satellite marker during animation
- yline()/xline() - Annotates key thresholds on the temperature graph
- uialert() - Alerts the user in case of invalid input values.
- cla() - Clears axes for new simulations.
- uifigure components - Used for interactive GUI elements

Orbit Animation;

The satellite's orbit is animated using trigonemetric functions to simulate a circular orbit in a 3D Earth-Centered interior frame.

The orbit assumes;
A constant altitude of approximately 400km
An inclination angle of 51.6, matching the international space station
A full orbital period of 90 minutes

As the satellite moves, its position is updated in real time;
x(t) = acos(theta)
y(t)=asin(theta)cos(i)
z(t)=asin(theta)sin(i)

Where a is the orbital radius, theta is the angular position, and i is the inclination angle.

The satellite marker changes color based on its temperature: green for stable conditions, and red if the temperature exceeds 350 K (the overheating threshold).

3. Flowchart and Pseudocode

Use either a flowchart or pseudocode to describe how your App is coded.

(START)

↓

User inputs: Power, Area, Mass, C_p, K_p, Time

↓

Calculate time span (convert min → sec)

↓

Define initial temperature ($T_0 = 300$ K)

↓

Solve $dT/dt = (Q_{in} - Q_{out})/(M \cdot C_p)$ using ode15s

↓

GENERATE OUTPUTS

→ Plot Temperature vs. Time

→ Animate 3d orbit

↓

Check: Is final Temp > 350 K

→ Yes → Show "OVERHEATING" (red marker warning!

→ No → Show "STABLE" green marker

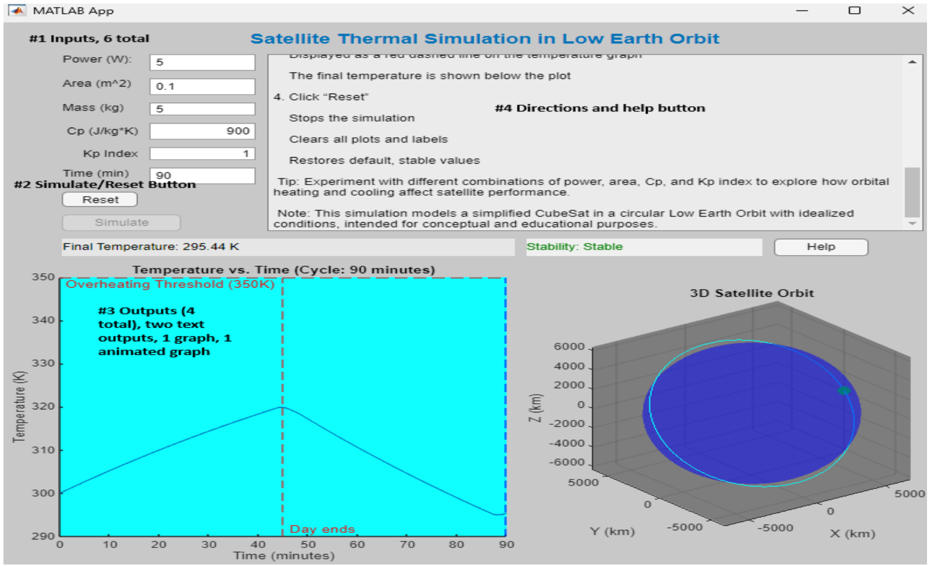
↓

Display Final temperature and Stability label

↓

END / wait for reset

4. Design and Layout



At the top of the app is a title label "Satellite Thermal Simulation in Low Earth Orbit".

The layout is divided using a grid system to group components:

1. Input (6 total) parameters - numeric fields for
    - Power (W)
    - Area (m^2)
    - Mass(kg)
    - Cp (J/kg*K)
    - Kp Index
    - Time (min)
    - Each input field includes a tooltip with valid ranges. The directions explain typical values.

2. Buttons:
    - "Simulate" runs the simulation with user inputs - it initiates the thermodynamic model and plots it in a temperature vs time graph, starts a orbital animation synced with time, it also outputs the final temperature and satellite stability status
    - "Reset" Stops the simulation and animation, as well as restores the default values for the inputs
3. Outputs (4 total):
   Temperature Graph

- Temperature vs Time graph, displays the results of the simulation
- Red dashed line marks overheating threshold of 350 Kelvin
- Vertical line marks day-night cycle and orbit completion

3D Oribt Animation

- 3d plot visualizes the satellite orbiting earth at ~400km altitude with a 51.6 degree inclination
- The satellite marker turns green if it is stable, turns red if its overheating
- The Earth is represented with a semi-transparent sphere, the orbital path is drawn around it.

Final Temperature Display - displays final temperature

Stability - Displays if the satellites thermodynamics are stable or overheating according to the model

4. Directions/Help Button
   - The directions explain how to use program
   - Help button reminds user to use reset button, which will reset all inputs to default and to hover over each input for range of inputs

Visual design was intended to be easy to read, color coding (green/red) is used for immediate status feedback. Input constraints and tooltips improve robustness and prevent user errors. The Reset function ensures that the simulation can be reset at any time.

5. Results

Validation

$M = 2\,kg$
$C_p = 900\,J/kg \cdot K$
$t_{phase} = 2700\,s$
$T_0 = 300\,K$
$A = 0.1\,m^2$
$\sigma = 5.67 \cdot 10^{-8}$
$\alpha = 0.6$
$S = 1361\,W/m^2$
$K_p = 1$
$P = 5\,W$

Phase 1 Daylight

$Q_{in\,day} = P + K_p \cdot S \cdot A \cdot \alpha$
$5 + 1 \cdot 1361 \cdot 0.1 \cdot 0.6 = 86.66\,W$
$Q_{out\,day} = \sigma \cdot A \cdot T^4$
Avg Temp $\approx 310\,K$
$Q_{out\,day} = 5.67 \cdot 10^{-8} \cdot 0.1 \cdot 310^4 = 55.41126\,W$
$P_{net\,day} = 86.66 - 55.4$
$P_{net\,day} = 31.24874\,W$
$Q_{day} = 31.2 \cdot 2700 \approx 84571.548$

$Q_{in\,night} = 5\,W$

$Q_{out\,night} = \sigma \cdot A \cdot T^4$
Avg Temp $\approx 240\,K$
$Q_{out\,night} = \sigma \cdot 0.1 \cdot 290^4$
$= 5.67 \cdot 10^{-8} \cdot 0.1 \cdot 290^4 \approx 40.1\,W$
$P_{net\,night} = 5 - 40.1 = -35.1\,W$
$Q_{night} = -35.1 \cdot 2700$
$Q_{night} = -94777.64829\,J$

$Q_{total} = Q_{day} + Q_{night}$
$Q_{tot} \approx 84371 + (-94777)$
$Q_{tot} \approx -10406\,J$

$$T_f = T_0 + \frac{Q_{total}}{m \cdot c_p}$$

$$T_f = 300 + \left(\frac{-10406}{2 \cdot 900}\right)$$

$$T_f = 300 - 5.7$$

$$T_f \approx 294.21$$

Final Temperature: 295.44 K

This matches and validates the apps output.

6. Conclusion

This project successfully demonstrates a simplified thermal simulation of a small satellite in low earth orbit using matlab app designer. With core thermodynamic principles and orbital mechanics, the app allows users to explore how internal power, material properties, and solar activity affect satellite temperature over time.

The combination of a temperature time graph and a 3d orbital animation provides both quantitative and visual insight into the satellite behavior. Key features such as real time stability feedback, overheating detection, and an interactive user interface make the app an effective educational and design tool.

Future improvements could include a more advanced thermal model, live Kp index integration, and more advanced and expanded orbital mechanics.

This was a rewarding project that deepend my understanding of thermodynamics, space systems and matlab.

Appendix

```
properties (Access = private)
    Property % Description

    StopSimulation = false;
  end


  methods (Access = private)
  % ODE function for temperature change
  function dTdt = temperatureODE(app, t, T, P, A, M, Cp, Kp)
    sigma = 5.67e-8; % Stefan-Boltzmann constant (W/m²·K⁴)
    s = 1361; % solar constant (W/m^2)
    alpha = 0.6; %Absorptivity of satellite surface approximation
    cycle_time = 5400; %Day night cycle, 45 minutes in seconds
    day_length = 2700; % day length in seconds

    if mod(t, cycle_time) < day_length
      Qin = P + (Kp * s * A * alpha); %Heat input during the day is power + (Kp *solar constant * area * alpha (absoprtivity)
    else
      Qin = P; % Heat input during night is just the power input from the payload
    end

    %Radioactive heat loss
```

```
    Qout = sigma * A * T^4; % Stefan-Boltzman constant * Area * Temperature ^4

    %Differential equation for temperature change
    dTdt = (Qin - Qout) ./ (M * Cp);
    end
```

```
 app.SimulateButton.Enable = 'off'; %Disables simulate button during simulation


        % Get inputs
        P = app.PowerWEditField.Value; % power of payload
        A = app.Aream2EditField.Value; %surface area of satellite
        M = app.MasskgEditField.Value; % mass
        Cp = app.CpEditField.Value; % specific heat
        Kp = app.SolarFlareEditField.Value; % Kp index
        t_max = app.TimeminEditField.Value * 60; % convert minutes to seconds
        if P <= 0 || A <= 0 || M <= 0 || Cp <= 0 || t_max <= 0 %no negative inputs
           uialert(app.UIFigure, 'All inputs must be positive numbers.', 'Invalid Input');
           app.SimulateButton.Enable = 'on';
              return;
        end
        %Initial temperature
        T0 = 300; % Kelvins
        tspan = 0:1:t_max; %1 second intervals creates a better plot
        %solve ODE
        [t, T] = ode15s(@(t, T) app.temperatureODE(t, T, P, A, M, Cp, Kp), tspan, T0); %Anyomous function in ode15s, ode15s
can handle the data set better than ode45

        if T(end) > 350 % 350 Kelvin is the critical failure point
           app.StabilityLabel.Text = sprintf('Stability: Overheating %.2f K', T(end)); % alerts user of overheating
           app.StabilityLabel.FontColor = [1, 0, 0]; % Red for overheating
        else
           app.StabilityLabel.Text = sprintf('Stability: Stable %,2f K', T(end)); % informs user on label that the satellite is
stable
           app.StabilityLabel.FontColor = [0, 0.5, 0]; % green for stable
        end

        %plot temperature graph
        cla(app.TempAxes); % clear app
        plot(app.TempAxes, (t / 60), T); % plot (t is put back into minutes)
        hold(app.TempAxes, 'on');
        yline(app.TempAxes, 350, '--r', 'Overheating Threshold (350K)', 'LabelHorizontalAlignment', 'left',
'LabelVerticalAlignment','bottom');
        full_cycle = 90; %LEO orbit is 90 minutes
        day = full_cycle / 2; % day has sunlight
        time_minutes = t_max / 60; % convert to minutes
        hold(app.TempAxes, 'on');
        xline(app.TempAxes, day, '--r', 'Day ends', 'LabelOrientation', 'horizontal', 'LabelVerticalAlignment', 'bottom'); %
shows when day stops, and night begins (only Qin is power during night)
        xline(app.TempAxes, full_cycle, '--b', '1 Orbit', 'LabelOrientation','horizontal','LabelVerticalAlignment','bottom');
%night is over, back to day, completion of 1 cycle or orbit
        hold(app.TempAxes, 'off');

        xlim(app.TempAxes, [0 t_max / 60]);
        app.TempAxes.XLabel.String = 'Time (minutes)';
        app.TempAxes.YLabel.String = 'Temperature (K)';
        app.TempAxes.Title.String = sprintf('Temperature vs. Time (Cycle: %d minutes)', time_minutes);
        app.TemperatureLabel.Text = sprintf('Final Temperature: %.2f K', T(end));
```

```
%%%%%%%%%%% Orbit Animation
% Constants for orbit
orbit_period = 90 * 60;             % 90 minutes in seconds
omega = 2 * pi / orbit_period;       % angular velocity (rad/sec)
inclination = deg2rad(51.6);        % inclination in radians
a = 6771;                           % km (Earth radius + 400 km altitude)
% Precompute orbit path for plotting (optional background path)
theta = linspace(0, 2*pi, 360);
x_orbit = a * cos(theta);
y_orbit = a * sin(theta) * cos(inclination);
z_orbit = a * sin(theta) * sin(inclination);
% Plot orbit and Earth
cla(app.OrbitAxes);
plot3(app.OrbitAxes, x_orbit, y_orbit, z_orbit, 'c');  % orbit path
hold(app.OrbitAxes, 'on');
[xe, ye, ze] = sphere(50);
surf(app.OrbitAxes, 6371*xe, 6371*ye, 6371*ze, ...
   'FaceColor', 'b', 'EdgeColor', 'none', 'FaceAlpha', 0.3); % Earth
satelliteMarker = plot3(app.OrbitAxes, 0, 0, 0, 'ro', ...
   'MarkerSize', 8, 'MarkerFaceColor', 'r');
axis(app.OrbitAxes, 'equal');
xlabel(app.OrbitAxes, 'X (km)');
ylabel(app.OrbitAxes, 'Y (km)');
zlabel(app.OrbitAxes, 'Z (km)');
title(app.OrbitAxes, '3D Satellite Orbit');
grid(app.OrbitAxes, 'on');
% Animate using ode15s time and temperature outputs
app.StopSimulation = false;
for i = 1:length(t)
if app.StopSimulation
   disp('Simulation stopped by user.');
      break;
end
      angle = omega * t(i);  % satellite’s angular position at t(i)
      x_sat = a * cos(angle);
      y_sat = a * sin(angle) * cos(inclination);
      z_sat = a * sin(angle) * sin(inclination);
      % Color changes with temperature
      if T(i) > 350
          color = 'r';
      else
          color = 'g';
end
% Update satellite marker position and color
set(satelliteMarker, 'XData', x_sat, 'YData', y_sat, ...
          'ZData', z_sat, ...
          'MarkerFaceColor', color, ...
'MarkerEdgeColor', color);
   drawnow;
   pause(0.001);
 end
```

References

NASA. (2014). Thermal Control Subsystem Design for CubeSats. NASA Technical Report NASA/TP-2014-216648. https://ntrs.nasa.gov/citations/20140016958

FAU College of Engineering. (2017). Thermal Analysis of CubeSat in Low Earth Orbit. https://public.eng.fau.edu/design/fcrar2017/papers/CubeSatThermalAnalysis.pdf

MathWorks. (n.d.). Solve Stiff Differential Equations—ode15s. https://www.mathworks.com/help/matlab/ref/ode15s.html

SmallSat Institute. (n.d.). Structures, Materials, and Mechanisms. https://www.nasa.gov/smallsat-institute/sst-soa/structures-materials-and-mechanisms/

Cardin, J. (2018). Analysis of a passive thermal control system for low Earth orbit satellite applications. Advances in Astronautics Science and Applications, 5(1), 1–7. https://medcraveonline.com/AAOAJ/AAOAJ-05-00130.pdf

MathWorks. (n.d.). Mission Analysis with the Orbit Propagator Block. MATLAB & Simulink Aerospace Blockset Examples. https://www.mathworks.com/help/aeroblks/mission-analysis-with-the-orbit-propagator-block.html