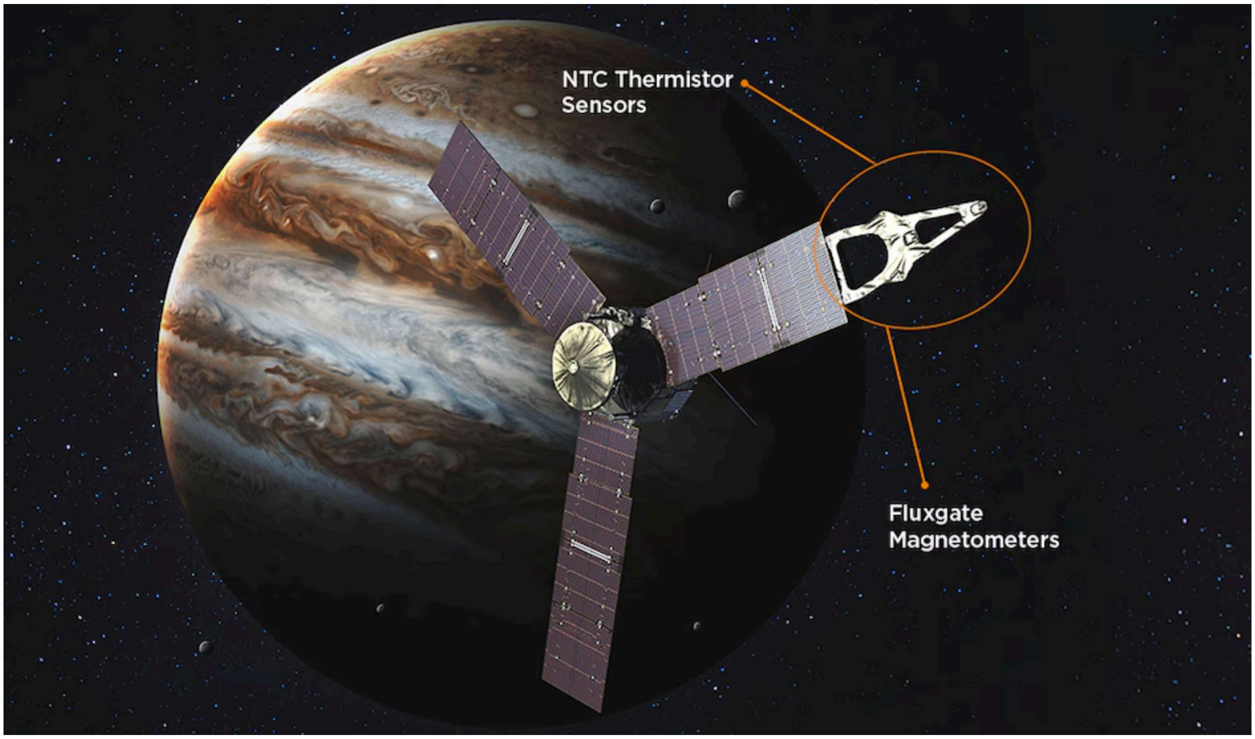
Report on Space-Grade Pressure and Temperature Sensors

Introduction



Space-Grade Sensors: Overview

Space-grade sensors are designed to withstand harsh space conditions, including extreme temperatures, radiation, vacuum, and mechanical stress. These sensors are crucial for spacecraft and satellites, ensuring reliable data collection and system functionality in space.

Key Features

1. **Radiation Tolerance**: Designed to resist damage from cosmic rays and solar radiation, ensuring long-term performance.
2. **Temperature Range**: Operates across extreme temperatures, from -150°C to +150°C, ensuring consistent functionality in space's fluctuating thermal environment.
3. **Vacuum Compatibility**: Built to function in the vacuum of space, preventing outgassing and material degradation.
4. **Material Requirements**: Uses corrosion-resistant, lightweight materials to maintain durability and performance.

Examples of Space-Grade Sensors

- **Accelerometers**: Measure acceleration forces and vibrations, aiding in spacecraft navigation.
- **Gyroscopes**: Measure rotational motion for orientation control.
- **Pressure Sensors**: Monitor atmospheric pressure within spacecraft or on planetary surfaces.
- **Strain Gauges**: Track mechanical strain on spacecraft, ensuring structural integrity.
- **Optical Sensors**: Capture images and analyze light for scientific data and remote sensing.

Some Companies That Make Nasa-Verified Space-Grade Sensors:

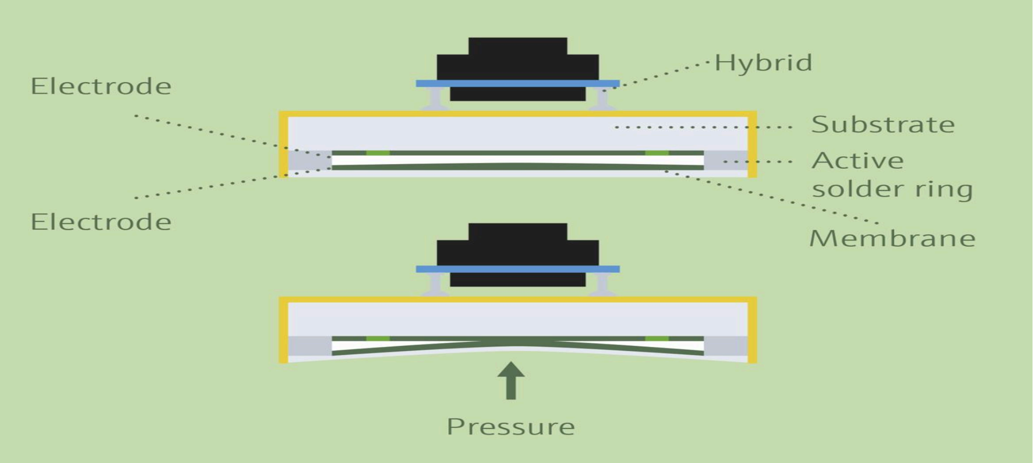
1. **TE Connectivity** – Produces reliable pressure and temperature sensors for space missions.
2. **Honeywell Aerospace** – Provides aerospace-grade sensors used in NASA missions.
3. **Sensonor (Spectris)** – Specializes in high-performance space sensors.
4. **Broadcom (Avago Technologies)** – Manufactures sensors for space applications.
5. **Ball Aerospace** – Known for space-grade environmental sensors.
6. **Northrop Grumman Innovation Systems** – Supplies sensors for space exploration.
7. **Moog Inc.** – Designs space-grade sensors for NASA missions.

1. Basic Functioning of Space-Grade Pressure and Temperature Sensors

1.1 Pressure Sensors

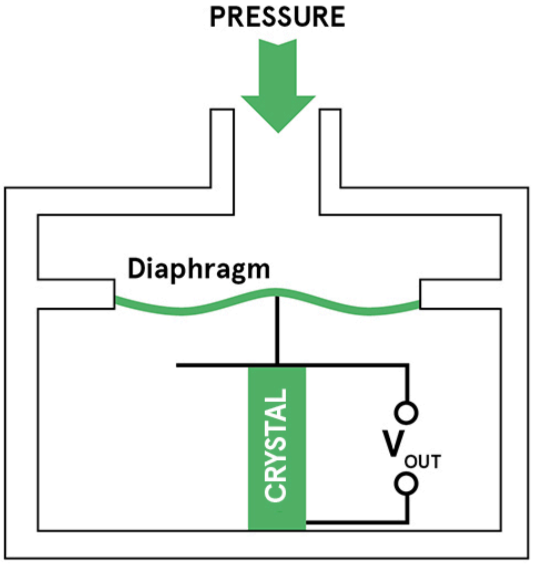
Pressure sensors are used to measure the **atmospheric pressure** at various altitudes, which is critical for assessing the environment around spacecraft, satellites, and rovers. In space applications, pressure sensors typically function using one of the following principles:

- **Capacitive Sensors**: These sensors operate by measuring the change in capacitance between two electrodes as pressure changes. As atmospheric pressure increases or decreases, the diaphragm in the sensor deform which alters the capacitance.



-

- **Piezoelectric Sensors**: These sensors utilize materials that generate a voltage when subjected to pressure. The voltage signal is then measured and used to determine the pressure.

- 

EXAMPLES

Piezoelectric (Honeywell 26PC): Used for monitoring atmospheric pressure and rocket fuel systems.

Capacitive (Barocap®): Used in spacecraft cabins to monitor air pressure.

Strain Gauge (Futek LCM): Used for high-precision pressure measurement in aerospace.

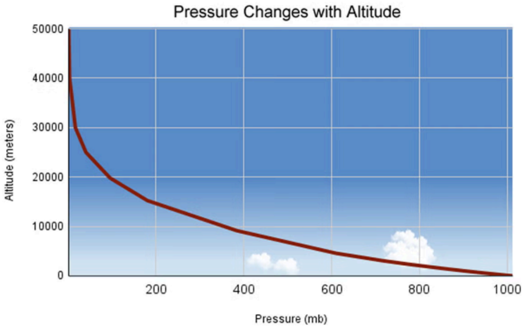
Pressure Calculation in Space: Pressure in the atmosphere decreases exponentially with altitude. The barometric formula can be used to model pressure as a function of height:

$P(h)=P0 \cdot \exp(-h/H)$

Where:

- P0 is the sea-level pressure.
- h is the altitude.
- H is the scale height for pressure.

This equation allows the calculation of pressure at various altitudes, with a noticeable decrease in pressure as altitude increases.



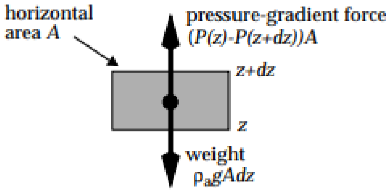


Figure 2-3 Vertical forces acting on an elementary slab of atmosphere

The atmosphere exerts an upward pressure force *P(z)A* on the bottom of the slab and a downward pressure force *P(z+dz)A* on the top of the slab; the net force, *(P(z)-P(z+dz))A*, is called the *pressure-gradient force*. Since *P(z)* > *P(z+dz)*, the pressure-gradient force is directed upwards. For the slab to be in equilibrium, its weight must balance the pressure-gradient force:

$$\rho_a g A dz = (P(z) - P(z+dz))A \qquad (2.3)$$

Rearranging yields

$$\frac{P(z+dz) - P(z)}{dz} = -\rho_a g \qquad (2.4)$$

The left hand side is ***dP/dz*** by definition. Therefore

$$\frac{dP}{dz} = -\rho_a g \qquad (2.5)$$

Now, from the ideal gas law,

$$\rho_a = \frac{PM_a}{RT} \qquad (2.6)$$

where M_a is the molecular weight of air and *T* is the temperature. Substituting *(2.6)* into *(2.5)* yields:

$$\frac{dP}{P} = \frac{M_a g}{RT} dz \qquad (2.7)$$

We now make the simplifying assumption that *T* is constant with altitude; as shown in Figure 2-2, *T* varies by only 20% below 80 km. We then integrate *(2.7)* to obtain

$$\ln P(z) - \ln P(0) = -\frac{M_a g}{RT} z \qquad (2.8)$$

which is equivalent to

$$P(z) = P(0)\exp\left(-\frac{M_a g}{RT} z\right) \qquad (2.9)$$

Equation *(2.9)* is called the *barometric law*. It is convenient to define a *scale height H* for the atmosphere:

$$H = \frac{RT}{M_a g} \qquad (2.10)$$

leading to a compact form of the Barometric Law:

$$P(z) = P(0)e^{-\frac{z}{H}} \qquad (2.11)$$

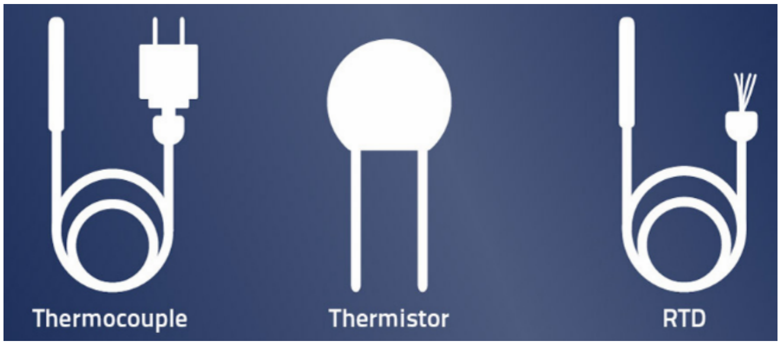
For a mean atmospheric temperature *T* = 250 K the scale height is *H* = 7.4 km. The barometric law explains the observed exponential dependence of *P* on *z* in Figure 2-2; from equation *(2.11)*, a plot of *z* vs. ln *P* yields a straight line with slope -*H* (check out that the slope in Figure 2-2 is indeed close to -7.4 km). The small fluctuations in slope in Figure 2-2 are caused by variations of temperature with altitude which we neglected in our derivation.

The vertical dependence of the air density can be similarly formulated. From *(2.6)*, ρ_a and *P* are linearly related if *T* is assumed constant, so that

$$\rho_a(z) = \rho_a(0)e^{-\frac{z}{H}} \qquad (2.12)$$

1.2 Temperature Sensors

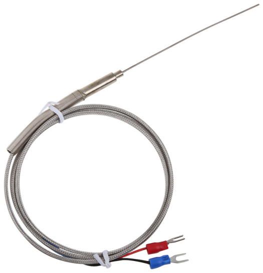
Temperature sensors measure the **ambient temperature** in the surrounding environment. In space applications, **thermistors** or **RTDs (Resistance Temperature Detectors)** are commonly used. These sensors change their resistance based on temperature, and the resistance change can be measured and converted into a temperature value.



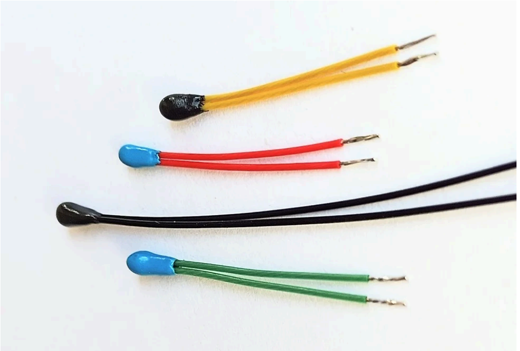
EXAMPLES

RTD Sensors (PT100): Used for precise temperature measurements in spacecraft and rockets.

Thermocouples (Type K): Used in high-temperature environments like spacecraft engines.



Thermistors (NTC): Used in thermal control systems of satellites.



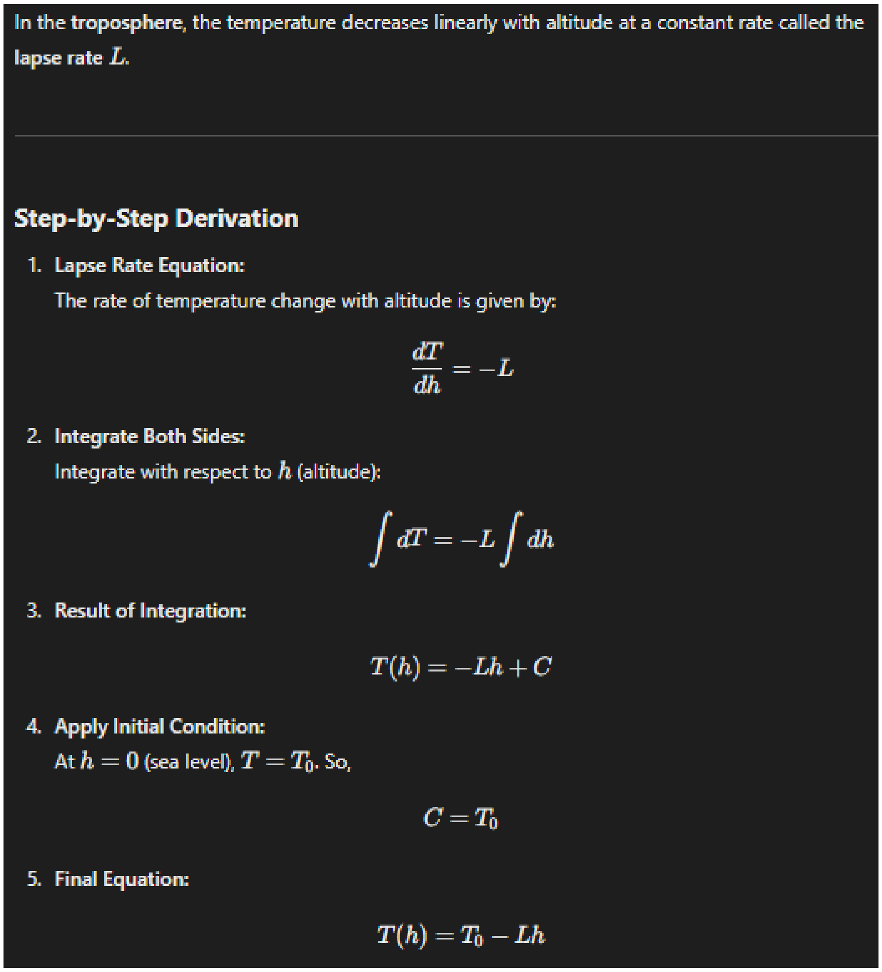
Temperature Calculation in Space

: Temperature decreases linearly with altitude in the troposphere at a constant lapse rate, which is typically around **6.5°C per kilometer**. The temperature at a certain altitude hh can be calculated as:

$T(h)=T0-L\cdot h$

Where:

- T0 is the sea-level temperature.
- L is the temperature lapse rate.
- h is the altitude



In the **troposphere**, the temperature decreases linearly with altitude at a constant rate called the **lapse rate** L.

Step-by-Step Derivation

1. **Lapse Rate Equation:**
   The rate of temperature change with altitude is given by:

$$\frac{dT}{dh} = -L$$

2. **Integrate Both Sides:**
   Integrate with respect to h (altitude):

$$\int dT = -L \int dh$$

3. **Result of Integration:**

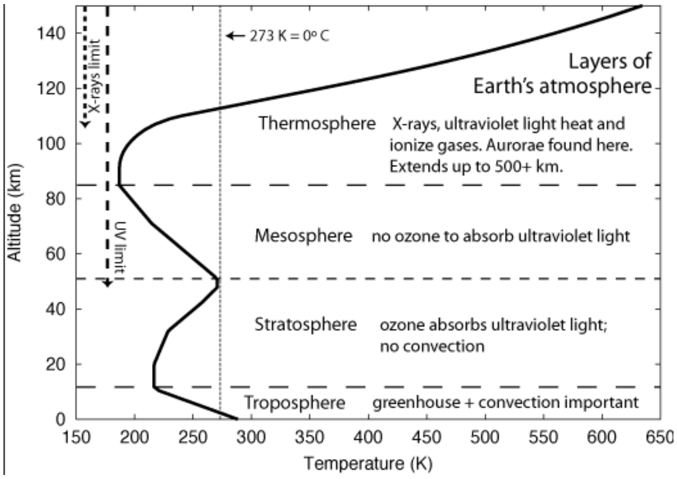
$$T(h) = -Lh + C$$

4. **Apply Initial Condition:**
   At $h = 0$ (sea level), $T = T_0$. So,

$$C = T_0$$

5. **Final Equation:**

$$T(h) = T_0 - Lh$$

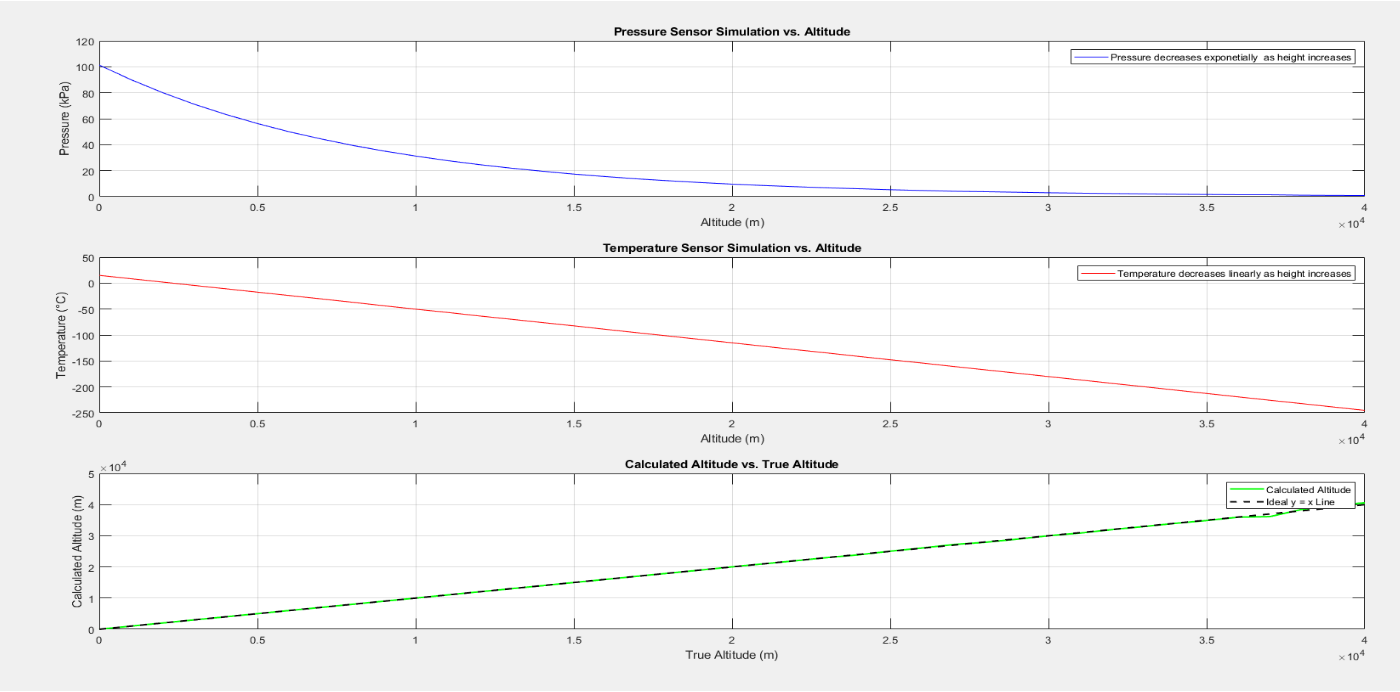


2. Sensor Simulation

In the MATLAB simulation provided in the earlier code, the pressure and temperature were calculated at different altitudes ranging from **0 to 40,000 meters**. The sensor output was simulated with the addition of **noise**, representing real-world inaccuracies that can arise due to environmental conditions or sensor limitations.

- **Pressure Output**: The pressure values decrease exponentially with altitude, and noise was added to simulate measurement error.
- **Temperature Output**: The temperature decreases linearly with altitude, with noise introduced to replicate sensor inaccuracy.

Furthermore, the sensor data can be used to estimate the **altitude** from the measured pressure using the inverse of the barometric formula.



CODE

```
% MATLAB Simulation for Space-Grade Pressure and Temperature Sensor (With Altitude)

% Constants
P0 = 101.325;  % Sea-level pressure in kPa
T0 = 288.15;   % Sea-level temperature in Kelvin
L = 0.0065;    % Temperature lapse rate in K/m
H = 8500;      % Scale height for pressure (in meters)
h_max = 40000; % Maximum height (in meters) for space simulation (40 km)

% Generate Altitude Data
heights = 0:1000:h_max;  % Altitudes from 0 to 40,000 meters (every 1 km)

% Calculate Pressure and Temperature as a function of Height
pressure_values = P0 * exp(-heights / H);  % Exponential decay for pressure
temperature_values = T0 - L * heights;    % Linear temperature decrease

% Simulate Sensor Output (Adding Noise)
pressure_output = pressure_values + randn(size(pressure_values)) * 0.05; % Adding
noise
temperature_output = temperature_values - 273.15 + randn(size(temperature_values)) *
0.1; % Convert Kelvin to Celsius and add noise

% Plotting the Results
figure;

% Plot a: Pressure vs. Altitude
subplot(3,1,1);
plot(heights, pressure_output, 'b');
xlabel('Altitude (m)');
ylabel('Pressure (kPa)');
title('Pressure Sensor Simulation vs. Altitude');
grid on;
legend('Pressure decreases exponetially  as height increases');


% Plot b: Temperature vs. Altitude
subplot(3,1,2);
plot(heights, temperature_output, 'r');
xlabel('Altitude (m)');
ylabel('Temperature (°C)');
title('Temperature Sensor Simulation vs. Altitude');
grid on;
legend('Temperature decreases linearly as height increases');
```

```
% Calculating True Altitude from Measured Pressure (Inverse Calculation)
calculated_heights = H * log(P0 ./ pressure_output); % Inverse of the pressure-altitude relationship

% Plot c: Calculated vs True Altitude (from Pressure Measurement)
subplot(3,1,3);
plot(heights, calculated_heights, 'g', 'LineWidth', 1.5);
hold on;
plot(heights, heights, 'k--', 'LineWidth', 1.5); % y = x reference line
hold off;
xlabel('True Altitude (m)');
ylabel('Calculated Altitude (m)');
title('Calculated Altitude vs. True Altitude');
grid on;
legend('Calculated Altitude', 'Ideal y = x Line');



% Analysis of Sensor Behavior
% Simulating extreme space-like conditions (low altitude)
extreme_heights = [0, 5000, 10000, 20000, 30000, 40000]; % Altitudes from 0 to 40 km

% Displaying sensor outputs under extreme conditions
fprintf('Sensor Output under Extreme Altitude Conditions:\n');
for h = extreme_heights
    pressure_reading = P0 * exp(-h / H) + randn() * 0.05;
    temperature_reading = T0 - L * h - 273.15 + randn() * 0.1; % Convert Kelvin to Celsius
    fprintf('Altitude: %.1f m -> Pressure: %.2f kPa, Temperature: %.2f °C\n', ...
        h, pressure_reading, temperature_reading);
end
```

3.Challenges Faced by Space-Grade Sensors

1. **Extreme Environmental Conditions**

   - **Temperature Extremes**: Space can reach temperatures between -150°C and +150°C. These extremes can affect sensor accuracy as components expand or contract.
   - **Vacuum and Low Pressure**: In the near-vacuum of space, pressure sensors must operate effectively in the absence of atmospheric pressure.

2. **Radiation Exposure**

   - **Radiation Damage**: High levels of cosmic rays and solar flares can physically damage sensor components, impacting their performance.
   - **Signal Degradation**: Radiation can interfere with sensor electronics, causing signal loss and measurement errors.

3. **Noise and Interference**

   - **Sensor Noise**: Internal electronics and environmental factors (like temperature fluctuations) can distort pressure and temperature readings.
   - **Electromagnetic Interference (EMI)**: Other spacecraft electronics may generate EMI, affecting sensor accuracy.

4. **Calibration and Drift**

   - **Calibration Challenges**: Accurate calibration is crucial but difficult due to limited maintenance opportunities in space.
   - **Long-Term Drift**: Over time, sensors may drift, leading to inaccuracies in long-duration missions.

5. **Power Consumption**

   - Power is a limited resource, so space-grade sensors must be energy-efficient to extend mission lifetimes.

Solutions to Address Challenges

1. **Thermal Control Systems**
    - Thermal insulation, heaters, and radiators maintain sensor temperatures within optimal operating ranges.
2. **Radiation-Hardened Components**
    - Special components resist radiation, ensuring the sensors' long-term reliability.
3. **Noise Filtering and Shielding**
    - Noise reduction and electromagnetic shielding techniques protect sensors from internal and external disturbances.
4. **Calibration Techniques**
    - Regular self-calibration ensures accurate measurements over time, minimizing the need for manual intervention.
5. **Power Efficiency**
    - Low-power sensors and energy-saving modes extend the mission's lifespan while maintaining functionality.

This approach to addressing space sensor challenges ensures their durability and reliability for the demanding conditions of space missions.

BONUS TASK

Simulating and Filtering Space Sensor Disturbances

Objective

The goal of this experiment is to simulate periodic fluctuations in pressure and temperature data of a space-grade sensor due to:

1. Mechanical vibrations during launch.
2. Orbital temperature cycles (transition between sunlight and shadow).
3. Engine vibrations and reaction wheel disturbances.

These disturbances are modeled as sinusoidal components, and a filtering approach is applied to mitigate their effects.

Methodology

1. Adding Sinusoidal Disturbances

We introduced sinusoidal oscillations in pressure and temperature data to mimic real-world disturbances:

- Low-frequency oscillations (0.0001 cycles/m) to represent orbital thermal cycles.
- Amplitude variations in pressure (~0.5 kPa) and temperature (~2°C).

2. Analyzing the Impact

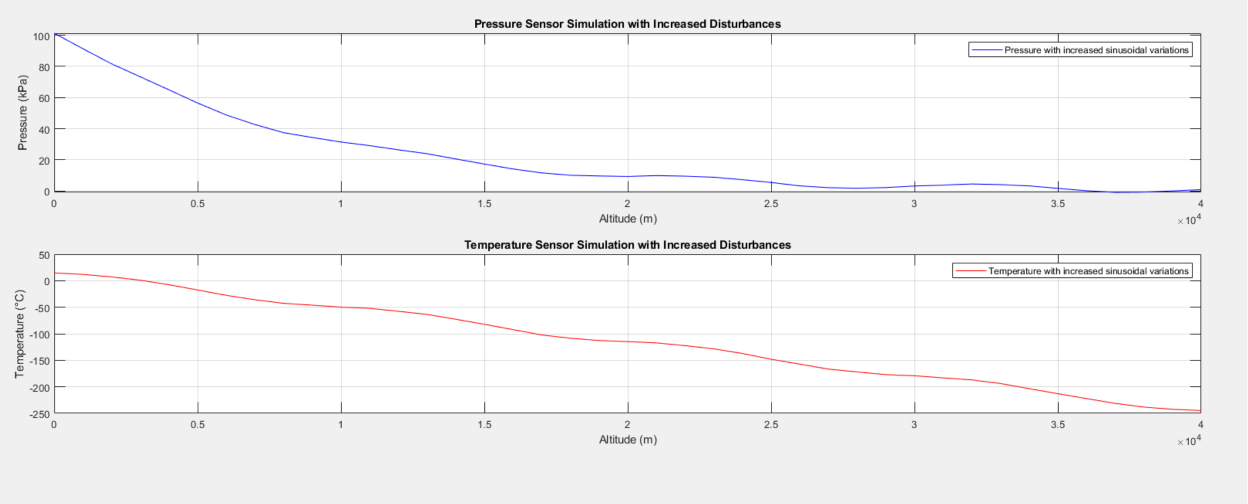
- Plotting the raw data with sinusoidal disturbances shows how sensor readings deviate from ideal values.
- The altitude was calculated using the noisy pressure data.

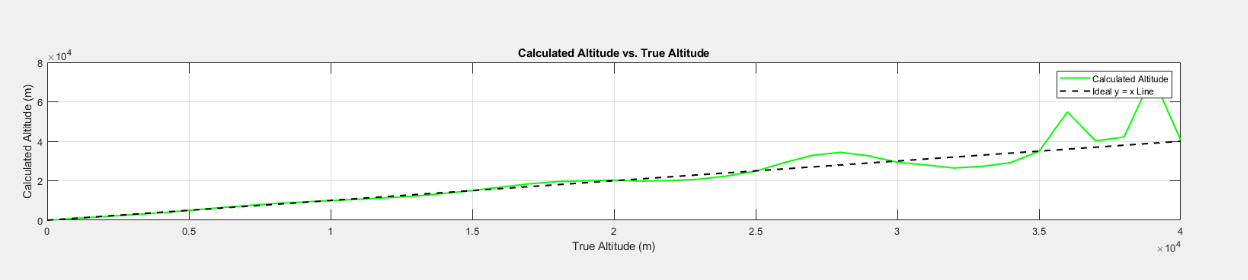
3. Filtering Approach

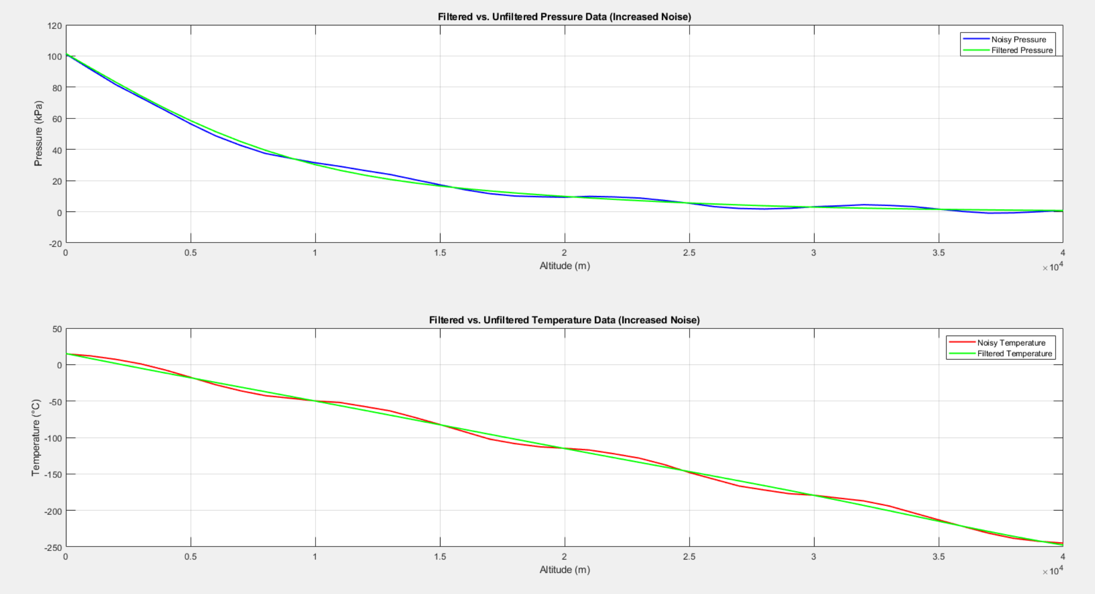
To mitigate oscillations, we applied a 4th-order Butterworth low-pass filter:

This filter effectively removes the high-frequency oscillations while preserving the overall trend.

WE CAN SEE THERE ARE SINOSUIDAL NOISE SUPERPOSED WITH ALL OUTPUTS







CODE

```matlab
% MATLAB Simulation for Space-Grade Pressure and Temperature Sensor
% (With Altitude & Periodic Disturbances)
% Constants
P0 = 101.325;  % Sea-level pressure in kPa
T0 = 288.15;   % Sea-level temperature in Kelvin
L = 0.0065;    % Temperature lapse rate in K/m
H = 8500;      % Scale height for pressure (in meters)
h_max = 40000; % Maximum height (in meters) for space simulation (40 km)
% Generate Altitude Data
heights = 0:1000:h_max;  % Altitudes from 0 to 40,000 meters (every 1 km)
% Calculate Pressure and Temperature as a function of Height
pressure_values = P0 * exp(-heights / H);  % Exponential decay for pressure
temperature_values = T0 - L * heights;    % Linear temperature decrease
% Simulate Sensor Output
pressure_output = pressure_values + randn(size(pressure_values)) * 0.2; % Increased
noise
temperature_output = temperature_values - 273.15 + randn(size(temperature_values)) *
0.5; % Increased noise
% Add Sinusoidal Disturbances to Simulate Vibrations
freq = 0.0001; % Lower frequency (0.0001 cycles/m = 0.1 cycles/km)
amplitude_pressure = 2.0; % Increased amplitude of pressure oscillations
amplitude_temperature = 6; % Increased amplitude of temperature oscillations
pressure_output = pressure_output + amplitude_pressure * sin(2 * pi * freq *
heights);
temperature_output = temperature_output + amplitude_temperature * sin(2 * pi * freq *
heights);
% Plotting the Results
figure;
% Plot 1: Pressure vs. Altitude
subplot(3,1,1);
plot(heights, pressure_output, 'b');
xlabel('Altitude (m)');
ylabel('Pressure (kPa)');
title('Pressure Sensor Simulation with Increased Disturbances');
grid on;
legend('Pressure with increased sinusoidal variations');
% Plot 2: Temperature vs. Altitude
subplot(3,1,2);
plot(heights, temperature_output, 'r');
xlabel('Altitude (m)');
ylabel('Temperature (°C)');
title('Temperature Sensor Simulation with Increased Disturbances');
grid on;
legend('Temperature with increased sinusoidal variations');
% Calculated Altitude vs. True Altitude
calculated_heights = H * log(P0 ./ pressure_output);
```

```
% Ensure third plot is displayed correctly
figure;
subplot(3,1,3);
plot(heights, calculated_heights, 'g', 'LineWidth', 1.5);
hold on;
plot(heights, heights, 'k--', 'LineWidth', 1.5); % y = x reference line
hold off;
xlabel('True Altitude (m)');
ylabel('Calculated Altitude (m)');
title('Calculated Altitude vs. True Altitude');
grid on;
legend('Calculated Altitude', 'Ideal y = x Line');
% Apply a Low-Pass Filter to Remove Sinusoidal Noise
fs = 1; % Sampling frequency (1 sample per km altitude step)
fc = 0.05; % Cutoff frequency in cycles/sample (0.05 cycles/km)
Wn = fc / (fs/2); % Correctly normalized cutoff frequency
[b, a] = butter(4, Wn, 'low'); % 4th order Butterworth filter
filtered_pressure = filtfilt(b, a, pressure_output);
filtered_temperature = filtfilt(b, a, temperature_output);
% Plot 4: Filtered vs. Unfiltered Data
figure;
subplot(2,1,1);
plot(heights, pressure_output, 'b', heights, filtered_pressure, 'g', 'LineWidth',
1.5);
xlabel('Altitude (m)');
ylabel('Pressure (kPa)');
title('Filtered vs. Unfiltered Pressure Data (Increased Noise)');
grid on;
legend('Noisy Pressure', 'Filtered Pressure');
subplot(2,1,2);
plot(heights, temperature_output, 'r', heights, filtered_temperature, 'g',
'LineWidth', 1.5);
xlabel('Altitude (m)');
ylabel('Temperature (°C)');
title('Filtered vs. Unfiltered Temperature Data (Increased Noise)');
grid on;
legend('Noisy Temperature', 'Filtered Temperature');
% Analysis of Sensor Behavior in Extreme Conditions
extreme_heights = [0, 5000, 10000, 20000, 30000, 40000]; % Altitudes from 0 to 40 km
fprintf('Sensor Output under Extreme Altitude Conditions:\n');
for h = extreme_heights
   pressure_reading = P0 * exp(-h / H) + randn() * 0.2; % Increased noise
   temperature_reading = T0 - L * h - 273.15 + randn() * 0.5; % Convert Kelvin to
Celsius, increased noise
   fprintf('Altitude: %.1f m -> Pressure: %.2f kPa, Temperature: %.2f °C\n', ...
       h, pressure_reading, temperature_reading);
end
```

Results & Discussion

- Raw sensor data showed clear oscillations due to added sinusoidal disturbances.
- Filtered sensor data was smoother, indicating that the noise was successfully suppressed.
- Altitude calculations based on noisy pressure readings showed errors, but filtering improved accuracy.

Key Takeaways

✅ Sinusoidal disturbances can significantly impact sensor accuracy.
✅ A low-pass filter effectively mitigates high-frequency oscillations.
✅ Filtering is crucial in space missions to ensure reliable sensor readings.

Conclusion

- ✅ This experiment successfully simulated and analyzed space sensor disturbances, highlighting the impact of periodic fluctuations on pressure and temperature readings.

- ✅ **Low-pass filtering** effectively reduced high-frequency noise, significantly improving the accuracy of sensor measurements.

- ✅ Space-grade sensors must operate in **harsh conditions**, including extreme temperatures, radiation exposure, and mechanical vibrations.

- ✅ Ensuring **sensor reliability** requires advanced materials, precise calibration, and robust signal processing techniques.

- ✅ As space missions become **longer and more complex**, the development of **more resilient and efficient sensors** will be crucial for future aerospace advancements. 🚀

REFERENCES:

Download link for MATLAB:-
https://in.mathworks.com/help/install/ug/install-products-with-internet-connection.html
Resources for studying space-grade sensors:-
https://www.te.com/en/whitepapers/sensors/sensors-in-space.html