SES/RAS 598: Space Robotics and AI

Lecture 1: Course Introduction & State Estimation Overview

Dr. Jnaneshwar Das

Arizona State University
School of Earth and Space Exploration

Spring 2025

Lecture Outline

Course Structure

- **Meeting Times:** Tu/Th 10:30-11:45am

Course Structure

- **Meeting Times:** Tu/Th 10:30-11:45am
- **Location:** PSF 647

Course Structure

- **Meeting Times:** Tu/Th 10:30-11:45am
- **Location:** PSF 647
- **Course Components:**
  - Assignments (20%)
  - Midterm Project (20%)
  - Final Project (50%)
  - Class Participation (10%)

Course Structure

- **Meeting Times:** Tu/Th 10:30-11:45am
- **Location:** PSF 647
- **Course Components:**
  - Assignments (20%)
  - Midterm Project (20%)
  - Final Project (50%)
  - Class Participation (10%)
- **Prerequisites:**
  - Linear algebra, calculus, probability theory
  - Python programming with NumPy, SciPy
  - Basic computer vision concepts
  - Linux/Unix systems experience

Course Resources

- **Recommended Books:**
  - Probabilistic Robotics (Thrun, Burgard, Fox)
  - Optimal State Estimation (Simon)
  - Pattern Recognition and Machine Learning (Bishop)

Course Resources

- **Recommended Books:**
  - Probabilistic Robotics (Thrun, Burgard, Fox)
  - Optimal State Estimation (Simon)
  - Pattern Recognition and Machine Learning (Bishop)
- **Interactive Tutorials:**
  - Sensor Fusion
  - Parameter Estimation
  - Gaussian Processes

Course Resources

- **Recommended Books:**
  - Probabilistic Robotics (Thrun, Burgard, Fox)
  - Optimal State Estimation (Simon)
  - Pattern Recognition and Machine Learning (Bishop)
- **Interactive Tutorials:**
  - Sensor Fusion
  - Parameter Estimation
  - Gaussian Processes
- **Required Software:**
  - Linux OS
  - ROS2
  - Python with scientific computing libraries

Why State Estimation?

- **Real-World Applications:**
  - Mars rover navigation
  - Drone flight control
  - Satellite attitude determination

Why State Estimation?

- **Real-World Applications:**
  - Mars rover navigation
  - Drone flight control
  - Satellite attitude determination
- **Key Challenges:**
  - Sensor noise and uncertainty
  - Environmental dynamics
  - Resource constraints

Why State Estimation?

- **Real-World Applications:**
  - Mars rover navigation
  - Drone flight control
  - Satellite attitude determination
- **Key Challenges:**
  - Sensor noise and uncertainty
  - Environmental dynamics
  - Resource constraints
- **Impact on Space Exploration:**
  - Autonomous navigation
  - Precision landing
  - Sample collection

Least Squares Estimation

- **Mathematical Foundation:**

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n} (y_i - h(\theta))^2$$

Least Squares Estimation

- **Mathematical Foundation:**

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n} (y_i - h(\theta))^2$$

- **Key Properties:**
  - Minimizes squared error
  - Optimal for Gaussian noise
  - Computationally efficient

Least Squares Estimation

- **Mathematical Foundation:**

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n} (y_i - h(\theta))^2$$

- **Key Properties:**
  - Minimizes squared error
  - Optimal for Gaussian noise
  - Computationally efficient
- **Applications:**
  - Sensor calibration
  - Trajectory estimation
  - Parameter identification

Implementation Example: Least Squares Estimation

```python
import numpy as np
from scipy.optimize import minimize

class LeastSquaresEstimator:
    def __init__(self, measurements, measurement_model):
        self.y = measurements          # Measurement vector
        self.h = measurement_model    # Measurement model function

    def cost_function(self, theta):
        """Compute sum of squared errors."""
        residuals = self.y - self.h(theta)
        return np.sum(residuals**2)

    def estimate(self, theta_init):
        """Find parameters that minimize squared error."""
        result = minimize(self.cost_function, theta_init,
                        method='Nelder-Mead')
        return result.x  # Return optimal parameters
```

- **Principle:**

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \prod_{i=1}^{n} p(y_i|\theta)$$

Maximum Likelihood Estimation

- **Principle:**

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \prod_{i=1}^{n} p(y_i|\theta)$$

- **Connection to Least Squares:**
  - Equivalent under Gaussian assumptions
  - More general framework
  - Handles different noise models

Maximum Likelihood Estimation

- **Principle:**

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \prod_{i=1}^{n} p(y_i|\theta)$$

- **Connection to Least Squares:**
  - Equivalent under Gaussian assumptions
  - More general framework
  - Handles different noise models
- **Space Applications:**
  - Orbit determination
  - Attitude estimation
  - Sensor fusion

Implementation Example: Maximum Likelihood Estimation

```python
import numpy as np
from scipy.stats import norm
from scipy.optimize import minimize

class MLEstimator:
    def __init__(self, measurements, measurement_model):
        self.y = measurements          # Measurement vector
        self.h = measurement_model    # Measurement model function

    def neg_log_likelihood(self, theta):
        """Compute negative log-likelihood."""
        residuals = self.y - self.h(theta)  # Assuming Gaussian noise model
        return -np.sum(norm.logpdf(residuals))

    def estimate(self, theta_init):
        """Find parameters that maximize likelihood."""
        result = minimize(self.neg_log_likelihood, theta_init,
                          method='Nelder-Mead')
        return result.x  # Return optimal parameters
```

State-Space Models

- **System Dynamics:**

$$
\begin{aligned}
x_{k+1} &= Ax_k + Bu_k + w_k \\
y_k &= Cx_k + v_k
\end{aligned}
$$

State-Space Models

- **System Dynamics:**

$$x_{k+1} = Ax_k + Bu_k + w_k$$
$$y_k = Cx_k + v_k$$

- **Components:**
  - State vector x_k
  - Input vector u_k
  - Measurement vector y_k
  - Process noise w_k
  - Measurement noise v_k

Case Study: Mars Rover Navigation

- **State Variables:**
  - Position (x, y, z)
  - Orientation (roll, pitch, yaw)
  - Velocities

Case Study: Mars Rover Navigation

- **State Variables:**
  - Position (x, y, z)
  - Orientation (roll, pitch, yaw)
  - Velocities
- **Sensors:**
  - Visual odometry
  - Inertial measurement unit (IMU)
  - Sun sensors

Case Study: Mars Rover Navigation

- **State Variables:**
  - Position (x, y, z)
  - Orientation (roll, pitch, yaw)
  - Velocities
- **Sensors:**
  - Visual odometry
  - Inertial measurement unit (IMU)
  - Sun sensors
- **Challenges:**
  - Wheel slippage
  - Varying terrain
  - Limited computational resources

Implementation Example: State-Space Model

```python
import numpy as np
from scipy.stats import multivariate_normal

class LinearStateSpaceModel:
    def __init__(self, A, B, C, Q, R):
        self.A = A  # State transition matrix
        self.B = B  # Input matrix
        self.C = C  # Measurement matrix
        self.Q = Q  # Process noise covariance
        self.R = R  # Measurement noise covariance

    def propagate_state(self, x, u=None):
        """Propagate state forward one step."""
        w = multivariate_normal.rvs(mean=np.zeros(x.shape), cov=self.Q)
        if u is not None:
            return self.A @ x + self.B @ u + w
        return self.A @ x + w

    def get_measurement(self, x):
        """Get noisy measurement of current state."""
        v = multivariate_normal.rvs(mean=np.zeros(self.C.shape[0]), cov=self.R)
        return self.C @ x + v
```

Preparation for Next Lecture

- **Review:**
  - Matrix operations
  - Probability concepts
  - Basic Python programming

Preparation for Next Lecture

- **Review:**
  - Matrix operations
  - Probability concepts
  - Basic Python programming
- **Setup:**
  - Install Linux if needed
  - Configure ROS2 environment
  - Test Python scientific libraries

Preparation for Next Lecture

- **Review:**
  - Matrix operations
  - Probability concepts
  - Basic Python programming
- **Setup:**
  - Install Linux if needed
  - Configure ROS2 environment
  - Test Python scientific libraries
- **Reading:**
  - Skim Kalman filter basics
  - Review assigned papers
  - Explore interactive tutorials

Questions?

Thank you!

Contact: jdas5@asu.edu