Intro
- for my project, I decided to dig into the kalman filter because I've only seen it covered at a surface level in previous classes and I wanted to finally understand how it works

- variables
- noises also have covariance

Kalman Filter
- what I learned is that the kalman filter is essentially a two step process
- the first step predicts the mean and covariance of a probability distribution describing the state estimate given the current state, any commands to the system, and the system dynamics
- the second step updates the prediction with information gained from the system's sensors by weighting the difference between the actual output and the predicted output with the reliability of the sensors

- observer because it builds up state estimate from history of inputs and outputs
- filter because it filters out process and measurement noise
- linear dynamics – matrix multiplications
- state – position, velocity
- inputs – vel commands, torques, forces
- output – values reported by sensors
- F – system dynamics
- G – how inputs drive dynamics
- H – how the state maps to the sensor measurements
- returns mean and covariance of state estimate, distribution of likely states
- prediction – next state from current state according to dynamics, accounts for info lost due to errors in motion model
- update – corrects prediction with info from sensors
- innovation nu – difference between actual output and predicted output
- S – predicted output covariance plus measurement noise covariance
- R – if large, sensor readings more believable than prediction, if low, prediction more believable than sensors

EKF
- the kalman filter assumes that the dynamics of the system are linear
- the extended kalman filter then is just the kalman filter applied to nonlinear systems by linearizing the system dynamics

- ekf just kalman filter for nonlinear systems
- linearize F and H matrices

SLAM
- next, I wanted to see how the kalman filter can be applied to the problem of simultaneous localization and mapping
- this is done by estimating the state of the robot and the state of landmarks in the environment at the same time
- the state is defined to include the positions of landmarks as well as the position of the robot
- the output is define to include the relative position of the robot to each landmark as well as any proprioceptive measurements

- rather than only estimate state of robot, also estimate state of landmarks in environment
- new state includes (x, y) position of landmarks
- now the measurements are distances from the robot to each landmark (either x and y difference or range and bearing)

Data Association
- you now have the problem of data association – which landmark is each measurement associated with?
- this problem can be solved with the mahalanobis distance
- if a measurement has a high mahalanobis norm with a particular landmark, then it is associated with that landmark
- if no measurement has a high mahalanobis norm, then you add a new landmark to the map

- problem of deciding which landmark each measurement associated with
- can do this with mahalanobis distance
- if no landmark has sufficiently high norm, add new landmark to map

Simulation
- next I simulated slam using the ROS package gmapping and gazebo simulator for the jackal robot
- it does a decent job until the end when it decides that what was previously a wall is now free space

- simulated slam using the ROS package gmapping and gazebo simulator for the jackal robot
- does a decent job until the end when it decides that what was previously a wall is now free space

Next Steps
- next I want to compare my own implementation of the kalman filter and slam to gmapping

- compare implementation of kalman filter and slam to gmapping

Intro
- for my project, I decided to dig into the kalman filter because I've only ever seen it covered at a surface level and I wanted to finally understand how it works

Kalman Filter
- what I learned is that it is essentially a two step process
- the first step predicts the mean and covariance of a probability distribution describing the state of the robot using the current state, the commands to the system, and the system dynamics
- the second step updates this prediction with information gained from the sensors by weighting the difference between the actual output and the predicted output with the reliability of the sensors

SLAM
- next, I wanted to see how the kalman filter could be used for SLAM
- this can be done by modifying the state to include any landmarks detected in the environment and modifiying the output include the relative position of the robot and each landmark

Data Association
- now you have to decide which landmark each measurement is associated with
- this problem can be solved with the mahalanobis distance
- if it is high for a particular landmark, then the measurement is associated with that landmark
- if it is low for all landmarks, then you add a new landmark to the state

Simulation
- lastly, I simulated slam using the ROS package gmapping and the gazebo simulator for the jackal robot
- as you can see, it works pretty well until the end when it decides that what was previously a wall is now free space

Next Steps
- in the future, I want to compare my own implementation of the kalman filter and slam to gmapping