6.2 Building maps with the Extended Kalman Filter

January 2, 2021

1   6.2 Building maps with the Extended Kalman Filter - The Nirvana shopping mall

The managers behind the **Nirvana shopping mall**, in the scope of their *experiencing the future* plan, have contacted **UMA-MR** (our brand-new mobile robotics company at UMA) looking for mobile robots able to guide their visitors between different points of interest in their facilities, like information points, the entrance to relevant (paying well) shops, rescue points, etc.

These managers have placed identifying marks close to the points of interest they want to consider, but do not know their exact location in the mall. **Our mission as engineers at *UMA-MR* is to build a map of the mall containing such points, so the robot can operate within it.** We are going to use for that an **Extended Kalman Filter (EKF)**.

Fortunately, our company has developed a system able to provide the exact location of our robots at each time instant, which are also equipped with a range-and-bearing nosiy sensor able to detect the identifying marks and take measurements to them.

1.1   6.2.1 Formalizing the problem

Since we are going to build a map of N landmarks, the position of those landmarks in the map m are the random variables to be estimated. In this way, the state vector in the mapping case is defined as:

$$m = [m_1, m_2, \cdots, m_N] = [x_1, y_1, \underbrace{x_2, y_2}_{\text{Position of landmark 2}}, \ldots, x_N, y_N]^T, \quad len(m) = 2N$$

In other words, we pursuit the estimation of the probability distribution:

$$p(m|z_{1:t}, x_{1:t})$$

being $z_{1:t}$ the sensor measurements taken until time instant t, and $x_{1:t}$ the robot poses from which those measurements were acquired. Recall that a sensor measurement is related to the pose x and the map m by means of the **observation function**:

$$z = h(x, m) + e \qquad e \sim N(0, Q)$$

Two assumptions are made when building maps using EFK.

Assumption 1: each landmark position is estimated independently For simplifying the problem, it is usual to assume that the estimation of the position of the landmarks is independent one to another, only depending each one on its observations, so:

$$p(m_i|z_{1:t}, x_{1:t}) = p(m_i|z_{1:t}^i, x_{1:t})$$

which results in the following simplification:

$$p(m|z_{1:t}, x_{1:t}) = p(m_1, m_2, \cdots, m_N|z_{1:t}, x_{1:t}) = \prod_{k=1}^{N} p(m_i|z_{1:t}^i, x_{1:t})$$

As a consequence of this, the estimation of, for example, 3 landmarks simultaneously is the same as using three concurrent and independent EKFs for estimating them.

Assumption 2: the map is static Unlike the localization case, where the robot pose changed over time, in this case it is assumed that the map is static, that is, the landmarks are still. This means that in the state transition model $m_k = m_{k-1}$, that is:

$$m_t = A_t m_{t-1} + B_t u_t + \epsilon_t \quad (A = I, u = 0, \epsilon = 0)$$

So good news here!, there is no need for a prediction step in the EKF, we only have to model the correction (update) one.

1.2   6.2.2 Developing the EKF filter for mapping the mall

```
[1]: %matplotlib notebook

import sys
import time

import numpy as np
from numpy import random
from scipy import linalg
import matplotlib
import matplotlib.pyplot as plt
```

```
sys.path.append("..")
from utils.tcomp import tcomp
from utils.AngleWrap import AngleWrap
from utils.DrawRobot import DrawRobot
from utils.PlotEllipse import PlotEllipse
from utils.Jacobians import J2
from utils.unit6.MapCanvas import MapCanvas
```

1.2.1 The provided robot

Take a look at the following `EFKMappingRobot()` class, which is already provided by your colleagues at **UMA-MR**, modeling **a robot equipped with a range and bearing sensor**, which is able to keep the following information: - `true_pose`: The exact pose of the robot in the environment, which is perfectly known. - `Q`: Uncertainty of the range and bearing sensor, which has the form: $\Sigma_{r\theta} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}$ - `xEst`: Vector with the estimated state, in this case the position of the M observed landmarks: $[x_1, y_1, x_2, y_2, \ldots, x_M, y_M]^T$. Its size changes over time with the detection of previously unobserved landmarks. - `PEst`: uncertainty associated with those predictions, with size $(M \times 2, M \times 2)$. Its size also changes over time. - `MappedLandmarks`: A vector with length equal to the number of landmarks in the map (N), which elements can take the following values: - `-1` if the landmark with that index has not benn seen yet. - `idx_in_xEst`: an odd number indicating the position of that landmark in `xEst`. For example, if during the robot operation in a map with 5 landmarks, it first detect the landmark with id 2, and later the one with id 4, the content of this vector would be `MappedLandmarks=[-1,-1,1,-1,3]`.

and to perform the following actions:

- `step()`: Performs a motion command, without noise.
- `observe()`: Returns a range and bearing measurement (in polars) to a given landmark in the map.
- `get_random_observation()`: Returns a range and bearing measurement (in polars) to a random landmark.

```
[2]: class EFKMappingRobot():
         def __init__(self, true_pose, sigma_r, sigma_theta, n_features):
             # Robot description
             self.true_pose = true_pose
             self.Q = np.diag([sigma_r, sigma_theta])**2

             # Map -- Initially empty
             self.xEst = np.empty((0, 0))
             self.PEst = np.empty((0, 0))
             self.QEst = 1.0*self.Q

             self.MappedLandmarks = -1*np.ones((n_features,1), int)

         def step(self, u):
```

```python
        self.true_pose = tcomp(self.true_pose, u)

    def observe(self, idx, world, noisy=True):
        """ Generate a observation of a feature in our world

            Args:
                world: Complete map of all landmarks in the world
                idx: Landmark to observe (index in world matrix)
                noisy: Add noise to z, prorportional to self.Q

            Returns:
                z: One range and bearing observation
        """

        z = np.empty((2, 1))
        delta = world[:, [idx]] - self.true_pose[0:2, :]

        # Range
        z[0, :] = np.sqrt(np.sum(delta**2))
        # Bearing
        z[1, :] = np.arctan2(delta[1, 0],delta[0, 0]) - self.true_pose[2, 0]
        z[1, :] = AngleWrap(z[1, :])

        if noisy:
            z = z + np.sqrt(self.Q)@random.randn(2,1)

        return z

    def get_random_observation(self, world, noisy=True):
        iLandmarks = world.shape[1]
        iLandmark = random.randint(iLandmarks)
        z = self.observe(iLandmark, world, noisy)
        return z, [iLandmark]
```

1.2.2   The prediction step

As commented, the map is considered static, so the prediction step is reduced to consider as the predicted landmarks' positions the ones estimated in the previous step. The same holds for the predicted uncertainty, so this steps results in something like this:

$$\text{def } \texttt{ExtendedKalmanFilter}(m_{t-1}, \Sigma_{t-1}, z_t):$$

Prediction.

$$\bar{m}_t = m_{t-1} \qquad\qquad\qquad\qquad (1.\ \text{Map prediction})$$

$$\bar{\Sigma}_t = \Sigma_{t-1} \qquad\qquad\qquad\qquad (2.\ \text{Uncertainty of prediction})$$

4

1.2.3 ASSIGNMENT 1: Implementing the naive prediction step

You are tasked to implement the previous behavior in the following function, which performs the prediction step.

```python
[3]: def prediction_step(robot: EFKMappingRobot):
         """ Performs the prediction step of the EKF algorithm for mapping
                robot: Robot base (contains state map: xEst, PEst)

             Returns: Nothing. But it modifies the state in robot
                xPred: Predicted position of the landmarks
                PPred: Predicted uncertainty of the landmarks positions
         """

         # We assume that the map is static
         xPred = robot.xEst
         PPred = robot.PEst

         return xPred, PPred
```

You can **test your function** with the next code:

```python
[4]: # TRY IT!
     xVehicleTrue = np.vstack([0.5, 0.7, 0]) # We know the exact robot pose at any␣
      ↪moment
     robot = EFKMappingRobot(xVehicleTrue, 1, 0.8, 1)
     robot.xEst = np.vstack([.5, .7])
     robot.PEst = np.diag([1.32, 0.8])
     [xPred,PPred] = prediction_step(robot)
     print('xPred:\n' + str(xPred))
     print('PPred:\n' + str(PPred))
```

```
xPred:
[[0.5]
 [0.7]]
PPred:
[[1.32 0.  ]
 [0.   0.8 ]]
```

Expected output:

```
xPred:
[[0.5]
 [0.7]]
PPred:
[[1.32 0.  ]
 [0.   0.8 ]]
```

1.2.4 Observing a landmark for first time

When the sensor onboard the robot detects a landmark for the first time, there is no need to do the EKF update step (indeed, since there is not previously knowledge about the landmark, there is nothing to update). Instead, we have to properly modify 1) the vector of estimated landmark positions, and 2) their associated uncertainties, to accommodate this new information:

1. **Modifying the state vector**: Insert the position of the new observed landmark $[x_{M+1}, y_{M+1}]$, using the sensor measurement $z_k = [r_k, \theta_k]$, at the end of the vector containing the estimated positions xEst, so:

$$xEst = [x_1, y_y, \cdots, x_M, y_M, x_{M+1}, y_{M+1}]$$

Since the measurment is provided in polar coordinates in the robot local frame, we have to convert them first to cartensians and then to the world frame using the robot pose $[x_v, y_v]'$. The function in charge of doing so can be defined as:

$$f(x_v, z_k) = \begin{bmatrix} x_{M+1} \\ y_{M+1} \end{bmatrix} = \begin{bmatrix} x_v \\ y_v \end{bmatrix} + r_k \begin{bmatrix} cos\alpha_k \\ sin\alpha_k \end{bmatrix}, \quad \alpha_k = \theta_k + \theta_v$$

2. **Extending the covariance matrix**. In order to acomodate the uncertainty regarding the position of the new landmark, we have to extend the covariance matrix in the following way:

$$PEst = \begin{bmatrix} [\Sigma_{xy}^1]_{2\times2} & \cdots & 0_{2\times2} \\ \vdots & \ddots & \vdots \\ 0_{2\times2} & \cdots & [\Sigma_{xy}^{M+1}]_{2\times2} \end{bmatrix}_{2n\times2n}$$

Notice that the covariance Σ_{xy}^{M+1} stands for the uncertainty in the measurement expressed in the world cartesian coordinates, retrieved by:

$$\Sigma_{xy}^{M+1} = J\Sigma_{r\theta}^{M+1}J^T$$

being $\Sigma_{r\theta}^{M+1}$ the uncertainty characterizing the sensor measurements (QEst in our code), and J (jGz in our code) the jacobian of the function $f(x_v, z_k)$ that expresses the measurement in global coordinates, which is:

$$J = \begin{bmatrix} \partial x/\partial r & \partial x/\partial \theta \\ \partial y/\partial r & \partial y/\partial \theta \end{bmatrix} = \begin{bmatrix} cos\alpha & -rsin\alpha \\ sin\alpha & rcos\alpha \end{bmatrix}$$

1.2.5 ASSIGNMENT 2: Incorporating a landmark detected for first time

Your work here is to: - Complete the get_new_landmark_jacobians() to compute the jacobian J. - Complete the incorporate_new_landmark() method to modify the state vector xEst and the covariance matrix PEst as explained above. We will make use of the linalg.block_diag() function at this point.

```python
[5]: def get_new_landmark_jacobians(Xv, z):
         """ Calculate the jacobian for transforming an observation to the world frame

             Args:
                 Xv: True pose of our robot
                 z: Observation of a landmark. In polar coordinates from the p.o.v.␣
         ↪of our robot.

             Returns:
                 2x2 matrix containing the corresponding jacobian.
         """
         r, a = z[0], z[1]
         c, s = np.cos(a), np.sin(a)

         jGz = np.array([
             [c, -r*s],
             [s, r*c]
         ])
         return jGz[:,:,0]
```

Is it working properly? **Try it!**

```python
[6]: # TRY IT!
     z = np.vstack([1.2,0.35])
     Xv = np.vstack([2, 2.1, 0])
     jGz = get_new_landmark_jacobians(Xv, z)
     print('jGz:\n' + str(jGz))
```

```
jGz:
[[ 0.93937271 -0.41147737]
 [ 0.34289781  1.12724726]]
```

Expected output:

```
jGz:
[[ 0.93937271 -0.41147737]
 [ 0.34289781  1.12724726]]
```

```python
[7]: def incorporate_new_landmark(robot: EFKMappingRobot, z, iLandmark, xPred, PPred):
         """ Incorporates the information relative to a new ladmark to our system
             robot: Robot base (contains state map: xEst, PEst)
             z: Observation of a landmark
             iLandmark: Index of z in the world map
             xPred: Predicted map
             PPred: Uncertainty of the prediction

             Returns: Nothing. But it modifies the state in robot
         """
         # This is a new feature, so add it to the map
```

```python
    # The observation is in the local frame of the robot, it has to
    # be translated to the global frame
    lmAngle   = robot.true_pose[2,] + z[1]
    xLandmark = np.vstack([ robot.true_pose[0,]+(z[0]*np.cos(lmAngle)) , robot.
→true_pose[1,]+(z[0]*np.sin(lmAngle)) ])

    # Add it to the current state
    nStates = xPred.size

    if nStates == 0:
        robot.xEst = xLandmark
    else:
        robot.xEst = np.vstack([robot.xEst, xLandmark]) #Each new feature two
→new rows

    # Compute the jacobian
    jGz = get_new_landmark_jacobians(robot.true_pose, z) #Dimension 2x2

    # Build a matrix M incorporating the jacobian to multiply the extendend PEst
→matrix by it (see below)
    if nStates != 0:
        # note we don't use jacobian w.r.t vehicle since the pose doesn't have
→uncertainty
        M = np.vstack([
                np.hstack([np.identity(nStates), np.zeros((nStates, 2))]),
                np.hstack([np.zeros((2, nStates)), jGz])
            ])
    else:
        # First landmark observed!
        M = jGz

    # print(M)
    # print(nStates)
    robot.PEst = M@linalg.block_diag(robot.PEst, robot.QEst)@M.T
    # print(robot.PEst)

    #This can also be done directly PEst = [PEst,zeros(nStates,2);
                                        # zeros(2,nStates),
                                        # jGz*QEst*jGz']

    #remember this landmark as being mapped: we store its ID for the state vector
    robot.MappedLandmarks[iLandmark] = robot.xEst.size-2 #Always an odd number
```

1.2.6 The correction (update) step

Once a landmark has been detected and its provided information (location and uncertainty) has been properly incorporated to our mapping system, such an information can be updated with new measurements of such a landmark. For doing so, the EKF algorithm performs the following steps:

Correction.

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1} \qquad \text{(3. Kalman gain)}$$
$$m_t = \bar{m}_t + K_t(z_t - h(x_t, \bar{m}_t)) \qquad \text{(4. Map estimation)}$$
$$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t \qquad \text{(5. Uncertainty of estimation)}$$
$$\texttt{return } m_t, \Sigma_t$$

Notice that the map (landmark locations) is estimated according to the map estimation in the previous time step $t - 1$, the *Kalman gain*, and the error (also called ***innovation***) between the obsevation taken by the sensor (z_t) and the one computed by the observation model given the robot pose and the predicted map, that is $z_t - h(x_t, \bar{m}_t)$.

In such equations, H ($\texttt{jH}$ in our code) represents the jacobian of the observation model. The shape of such a jacobian is $(2, M \times 2)$, and has the following form:

$$H = \begin{bmatrix} 0 & 0 & \cdots & \underbrace{\begin{bmatrix} jHxl_{11} & jHxl_{12} \\ jHxl_{21} & jHxl_{22} \end{bmatrix}}_{\text{Jacobian for the observed landmark, } jHxl} & \cdots & 0 & 0 \\ 0 & 0 & \cdots & & \cdots & 0 & 0 \end{bmatrix}$$

$$jHxl = \begin{bmatrix} (x_l - x)/d & (y_l - y)/d \\ -(y_l - y)/d^2 & (x_l - x)/d^2 \end{bmatrix}$$

where $[x_l, y_l]$ is the position of the landmark, $[x, y]$ is the robot location, and $d = \sqrt{(x_l - x)^2 + (y_l - y)^2}$.

1.2.7 ASSIGNMENT 3: Updating the knowledge about an observed landmark

Your job at this point is:

- To implement the function $\texttt{get\_observation\_jacobian()}$ returning the jacobian of the observed landmark, that is, $jHxl$.
- To complete the $\texttt{update\_step()}$ method that performs the update step of the EKF algorithm.

```
[8]:  def get_observation_jacobian(xPred, xLandmark):
          """ Calculate the jacobian of the observation model.

              Needed to update a landmark we have already seen.
              Hint. Similar to the one described in unit 5 (Localization)

              Args:
                  xPred: True pose of our robot.
```

```python
            xLandmark: Estimated pose of a landmark in our map. World p.o.v in
    ↪cartesian coordinates.
                Does not contain an angle.

        Return:
            jHxl: 2x2 matrix containing the corresponding jacobian.
        """
    delta = xLandmark - xPred[0:2]

    xdist = delta[0]
    ydist = delta[1]
    r = np.sqrt( xdist**2 + ydist**2)
    r2 = r**2
    jHxl = np.array([
        [ xdist/r  ,  ydist/r],
        [-ydist/r2 , xdist/r2]
    ])
    return jHxl[:,:,0]
```

Let's see if your implementation is right:

```python
[9]: # TRY IT!
     xLandmark = np.vstack([2.5,2])
     xPred = np.vstack([2, 2.1, 0])
     jHxl = get_observation_jacobian(xPred, xLandmark)
     print('jHxl:\n' + str(jHxl))
```

```
jHxl:
[[ 0.98058068 -0.19611614]
 [ 0.38461538  1.92307692]]
```

Expected output:

```
jHxl:
[[ 0.98058068 -0.19611614]
 [ 0.38461538  1.92307692]]
```

```python
[10]: def update_step(robot, z, iLandmark, xPred, PPred):
          """ Performs the update step of EKF
              robot: Robot base (contains state map: xEst, PEst)
              z: Observation of a landmark
              iLandmark: Index of z in the world map
              xPred: Predicted map
              PPred: Uncertainty of the prediction

              Returns: Nothing. But it modifies the state in robot
          """
          # Find out where it is in state vector
```

```python
        landmarkIndex = robot.MappedLandmarks[iLandmark[0], 0]

        # xLandmark is the current estimation of the position of the
        # landmard "FeatureIndex"
        xLandmark = xPred[landmarkIndex:landmarkIndex+2]

        # DONE Predicts the observation
        zPred = robot.observe(0, xLandmark, noisy=False) # Hint: use robot.observe␣
→function

        # Get observation Jacobians
        jHxf = get_observation_jacobian(robot.true_pose, xLandmark)

        # Fill in state jacobian
        # (the jacobian is zero except for the observed landmark)
        jH = np.zeros((2, xPred.size))
        jH[:, landmarkIndex:landmarkIndex+2] = jHxf

        #
        # Kalman update
        #
        Innov = z-zPred # Innovation
        Innov[1] = AngleWrap(Innov[1])
        S = jH@PPred@jH.T + robot.QEst
        K = PPred@jH.T@linalg.inv(S) # Gain
        robot.xEst = xPred + K@Innov
        robot.PEst = (np.eye(robot.PEst.shape[0]) - K@jH)@PPred
        #robot.PEst = PPred - K@S@K.T # Alternative way

        #ensure P remains symmetric
        robot.PEst = 0.5*(robot.PEst+robot.PEst.T)
```

1.2.8 ASSIGNMENT 4: Putting all together in the EKF algorithm

Now that you have implemented the building blocks of the EKF filter for mapping the **Nirvana shopping mall**, it is time to write a simple function `EKFMapping()` putting them together. For that, **you have to** call each method with the appropriate parameters.

```python
[11]: def EKFMapping(robot: EFKMappingRobot, z, iLandmark):
          """ EFK algorithm for mapping

                  robot: Robot base (contains state map: xEst, PEst)
                  z: Observation of a landmark
                  iLandmark: Index of z in the world map

              Returns: Nothing. But it modifies the state in robot
```

```
    """

    # Do prediction step
    [xPred, PPred] = prediction_step(robot)

    # Check if feature observed is in map
    if robot.MappedLandmarks[iLandmark] > -1:
        update_step(robot, z, iLandmark, xPred, PPred)

    else:
        # This is a new feature, so add its information to the map
        incorporate_new_landmark(robot, z, iLandmark, xPred, PPred)
```

1.3   6.2.3 Testing the mapping system

1.3.1   Playing with one landmark

Let's consider that the mall has only one landmark to get things started (`nLandmarks=1`). The following function provides a demo where the robot is commanded to follow a squared trajectory while observing a landmark after each movement.

The **Nirvana** managers are curious about the state and dimensions of the variables storing the estimated positions `xEst` and their associated uncertainties `Pest`, so we show their content after each 5 iterations of the algorithm.
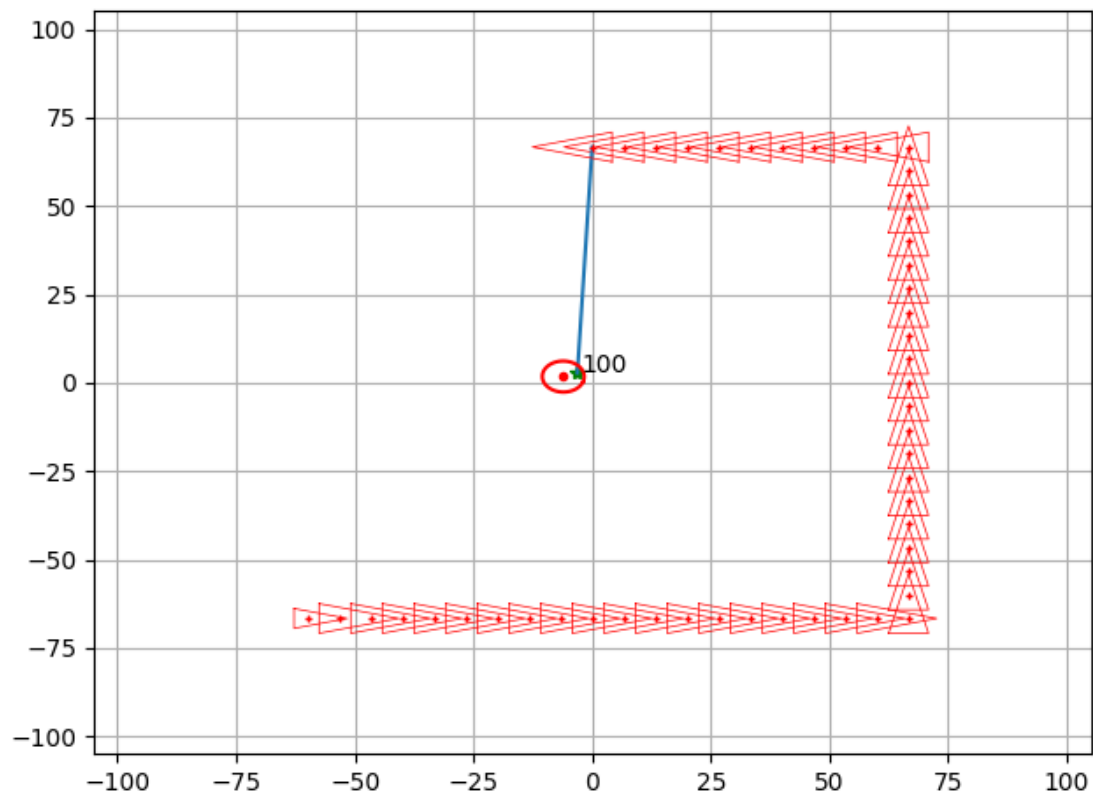
Fig. 1: Example run of the EKF algorithmn for mapping (only one landmark). it shows the true pose (in red), the real pose of the landmark (as a green star), and the estimation from the EKF algorithm (pose and confidence ellipse).

```python
[14]: def demo_ekf_mapping(robot,
                           Map,
                           nLandmarks,
                           mode='non_stop',
                           logger=None,
                           nSteps=100, # Number of motions
                           turning= 40, # Number of motions before turning (square
      ↪path)
                           print_each=2):

          %matplotlib notebook
          if mode == 'step_by_step':
              matplotlib.use('TkAgg')

          # storing the number of times a landmark has been seen
          # also store the handler to the graphical info shown
          canvas = MapCanvas(nLandmarks)

          canvas.ax.plot(Map[0, :], Map[1, :], 'g*')
          hObsLine = canvas.ax.plot([0,0], [0,0], linestyle=':')

          # Control action
          u = np.zeros((3, 1))
          u[0] = (2.*MapSize/1.5)/turning
          u[1] = 0.

          # Start the loop!
          for k in range(nSteps):
              #
              # Move the robot
              #
              u[2]=0.
              if k%turning == turning-1:
                  u[2] = np.pi/2

              robot.step(u) # Perfectly known robot pose

              z, iLandmark = robot.get_random_observation(world=Map)

              # Update the "observedtimes" for the feature and plot the reading
              canvas.increment_observed_times(iLandmark)
              canvas.PlotNumberOfReadings(robot.true_pose, iLandmark, Map)
```

```python
            EKFMapping(robot, z, iLandmark)

            # Print map evolution each 5 steps
            if not k%5:
                with np.printoptions(precision=3):
                    print('Iteration: ' + str(k))
                    print('Estimated xEst:\n' + str(robot.xEst))
                    print('Estimated PEst:\n' + str(robot.PEst))
                    print('-------------------------------')

            # Log important values
            if logger is not None:
                logger.log(k, robot, Map)

            # Drawings
            if k%print_each == print_each-1:
                DrawRobot(canvas.fig, canvas.ax,robot.true_pose,
 ↪'r')#plot(xVehicleTrue(1),xVehicleTrue(2),'r*')
                canvas.DoMapGraphics(robot) # Draw estimated poitns (in black) and
 ↪ellipses
                plt.axis([-MapSize-5, MapSize+5, -MapSize-5, MapSize+5]) # Set
 ↪limits again
                plt.draw()

                if mode == 'step_by_step':
                    plt.waitforbuttonpress(-1)
                elif mode == 'visualize_process':
                    plt.pause(0.2)
                elif mode == 'non_stop':
                    pass # non stop!

    # Final drawings
    %matplotlib inline
    if logger is not None:
        logger.plot()
```

```python
[15]: #mode = 'step_by_step'
      mode = 'visualize_process'
      #mode = 'non_stop'

      # WORLD MAP
      # Num features/landmarks considered within the map
      nLandmarks = 1
      # Generation of the map
      MapSize = 100
      Map = MapSize*random.rand(2,nLandmarks)-MapSize/2
```
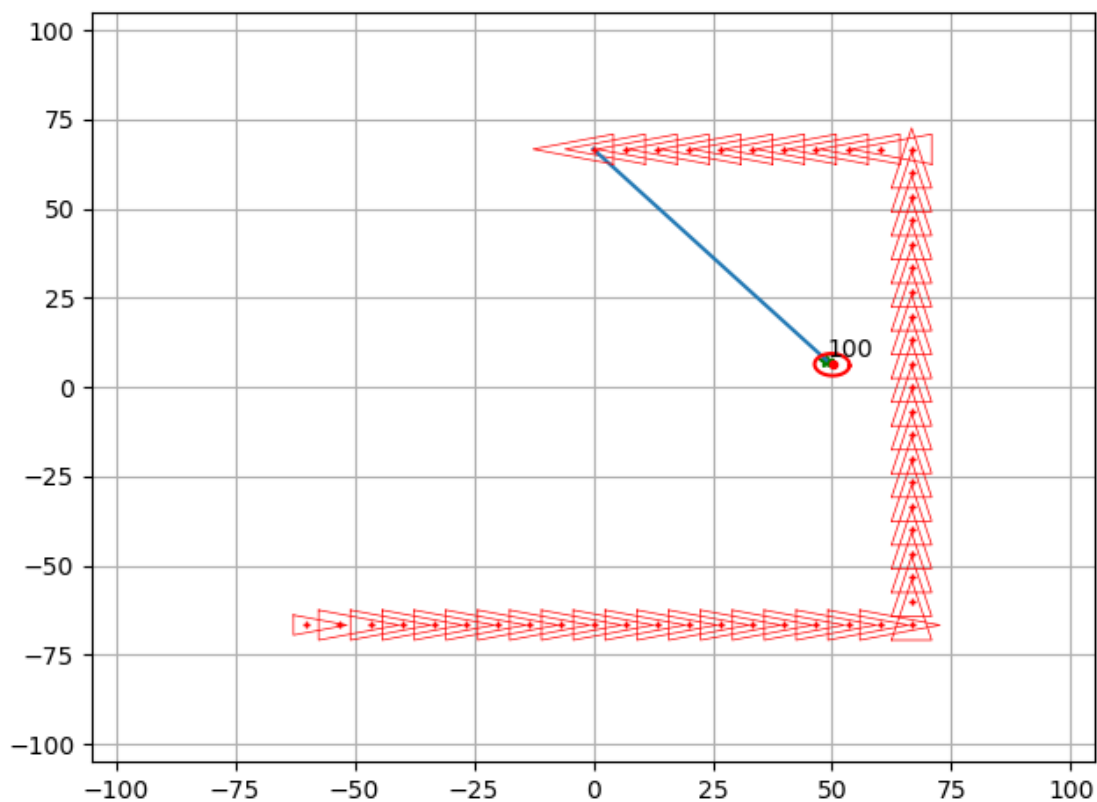
```python
# ROBOT
# Covariances for our very bad&expensive sensor (in the system <d,theta>)
Sigma_r = 8.0
Sigma_theta = 8*np.pi/180
# Initial robot pose
xVehicleTrue = np.vstack([-MapSize/1.5, -MapSize/1.5, 0.]) # We know the exact␣
 ↪robot pose at any moment


robot = EFKMappingRobot(xVehicleTrue, Sigma_r, Sigma_theta, nLandmarks)

demo_ekf_mapping(robot, Map ,nLandmarks, mode=mode)
```



```
Iteration: 0
Estimated xEst:
[[48.245]
 [ 3.712]]
Estimated PEst:
[[ 142.349 -124.215]
 [-124.215  260.929]]
----------------------------------
Iteration: 5
Estimated xEst:
[[54.879]
```

```
 [ 2.38 ]]
Estimated PEst:
[[ 22.595 -18.926]
 [-18.926  40.709]]
----------------------------------
Iteration: 10
Estimated xEst:
[[52.347]
 [ 5.644]]
Estimated PEst:
[[12.578 -9.412]
 [-9.412 18.999]]
----------------------------------
Iteration: 15
Estimated xEst:
[[51.694]
 [ 9.218]]
Estimated PEst:
[[ 8.514 -5.629]
 [-5.629 11.107]]
----------------------------------
Iteration: 20
Estimated xEst:
[[51.927]
 [ 8.834]]
Estimated PEst:
[[ 6.309 -3.624]
 [-3.624  7.174]]
----------------------------------
Iteration: 25
Estimated xEst:
[[51.575]
 [ 9.319]]
Estimated PEst:
[[ 4.909 -2.405]
 [-2.405  4.945]]
----------------------------------
Iteration: 30
Estimated xEst:
[[50.808]
 [ 8.553]]
Estimated PEst:
[[ 3.924 -1.607]
 [-1.607  3.592]]
----------------------------------
Iteration: 35
Estimated xEst:
[[50.636]
```

```
 [ 8.147]]
Estimated PEst:
[[ 3.218 -1.087]
 [-1.087  2.756]]
----------------------------------
Iteration: 40
Estimated xEst:
[[51.061]
 [ 7.107]]
Estimated PEst:
[[ 2.717 -0.748]
 [-0.748  2.222]]
----------------------------------
Iteration: 45
Estimated xEst:
[[50.842]
 [ 7.412]]
Estimated PEst:
[[ 2.266 -0.529]
 [-0.529  1.867]]
----------------------------------
Iteration: 50
Estimated xEst:
[[50.337]
 [ 7.85 ]]
Estimated PEst:
[[ 1.81  -0.406]
 [-0.406  1.613]]
----------------------------------
Iteration: 55
Estimated xEst:
[[50.198]
 [ 6.784]]
Estimated PEst:
[[ 1.378 -0.402]
 [-0.402  1.383]]
----------------------------------
Iteration: 60
Estimated xEst:
[[50.394]
 [ 5.648]]
Estimated PEst:
[[ 1.166 -0.387]
 [-0.387  0.873]]
----------------------------------
Iteration: 65
Estimated xEst:
[[49.906]
```

```
 [ 6.551]]
Estimated PEst:
[[ 0.862 -0.102]
 [-0.102  0.507]]
----------------------------------
Iteration: 70
Estimated xEst:
[[50.53 ]
 [ 6.328]]
Estimated PEst:
[[ 0.708 -0.04 ]
 [-0.04   0.459]]
----------------------------------
Iteration: 75
Estimated xEst:
[[50.575]
 [ 6.317]]
Estimated PEst:
[[ 0.647 -0.029]
 [-0.029  0.44 ]]
----------------------------------
Iteration: 80
Estimated xEst:
[[50.477]
 [ 6.181]]
Estimated PEst:
[[ 0.617 -0.026]
 [-0.026  0.425]]
----------------------------------
Iteration: 85
Estimated xEst:
[[50.574]
 [ 6.128]]
Estimated PEst:
[[ 0.591 -0.025]
 [-0.025  0.412]]
----------------------------------
Iteration: 90
Estimated xEst:
[[50.147]
 [ 6.16 ]]
Estimated PEst:
[[ 0.569 -0.022]
 [-0.022  0.399]]
----------------------------------
Iteration: 95
Estimated xEst:
[[50.142]
```

```
  [ 6.167]]
Estimated PEst:
[[ 0.549 -0.019]
 [-0.019  0.387]]
--------------------------------
```

1.3.2 Considering a larger number of landmarks

Once our EKF implementation is working with one landmark, let's try it in a scenario with 5 landmarks. Again, the content of the `xEst` and `Pest` is shown after each 5 iterations of the algorithm.
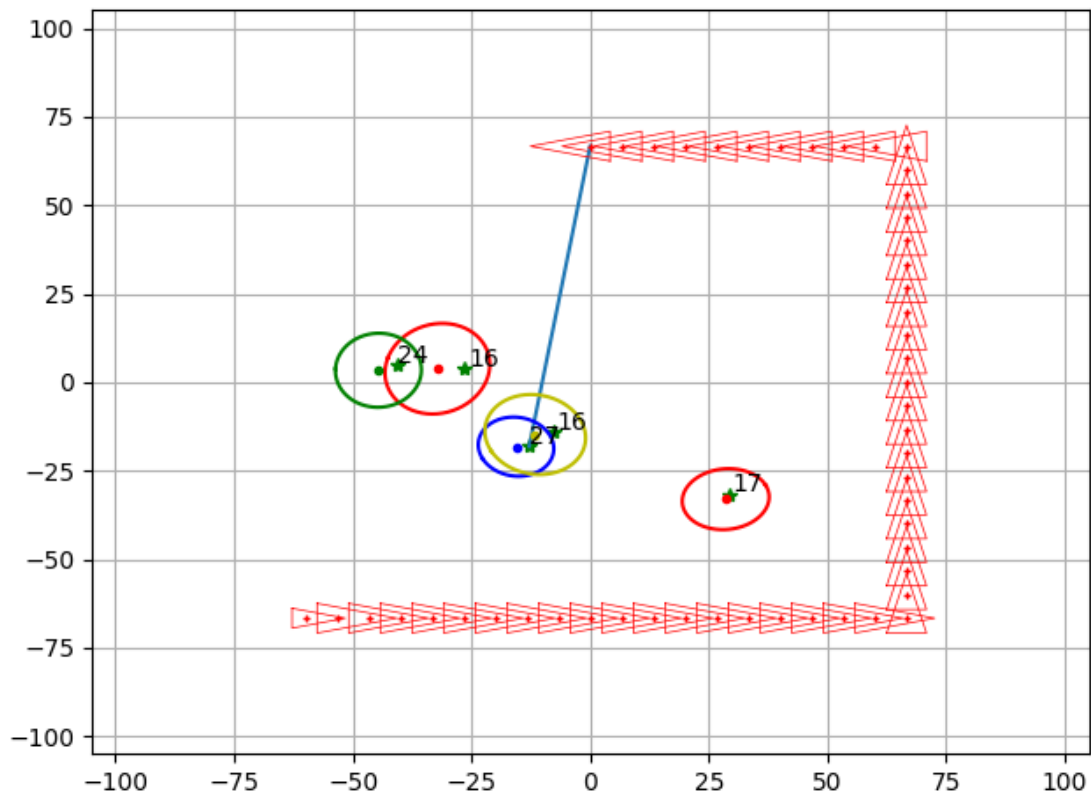

Fig. 2: Execution of the EKF algorithmn for mapping (multiple landmarks). Same as in Fig 1., each landmark is accompanied by a number of times observed.

```
[17]: #mode = 'step_by_step'
      mode = 'visualize_process'
      #mode = 'non_stop'

      # WORLD MAP
      # Num features/landmarks considered within the map
      nLandmarks = 5
      # Generation of the map
      MapSize = 100
```

```
Map = MapSize*random.rand(2,nLandmarks)-MapSize/2

# ROBOT
# Covariances for our very bad&expensive sensor (in the system <d,theta>)
Sigma_r = 8.0
Sigma_theta = 7*np.pi/180
# Initial robot pose
xVehicleTrue = np.vstack([-MapSize/1.5, -MapSize/1.5, 0.]) # We know the exact␣
 ↪robot pose at any moment

robot = EFKMappingRobot(xVehicleTrue, Sigma_r, Sigma_theta, nLandmarks)

demo_ekf_mapping(robot, Map ,nLandmarks, mode=mode)
```
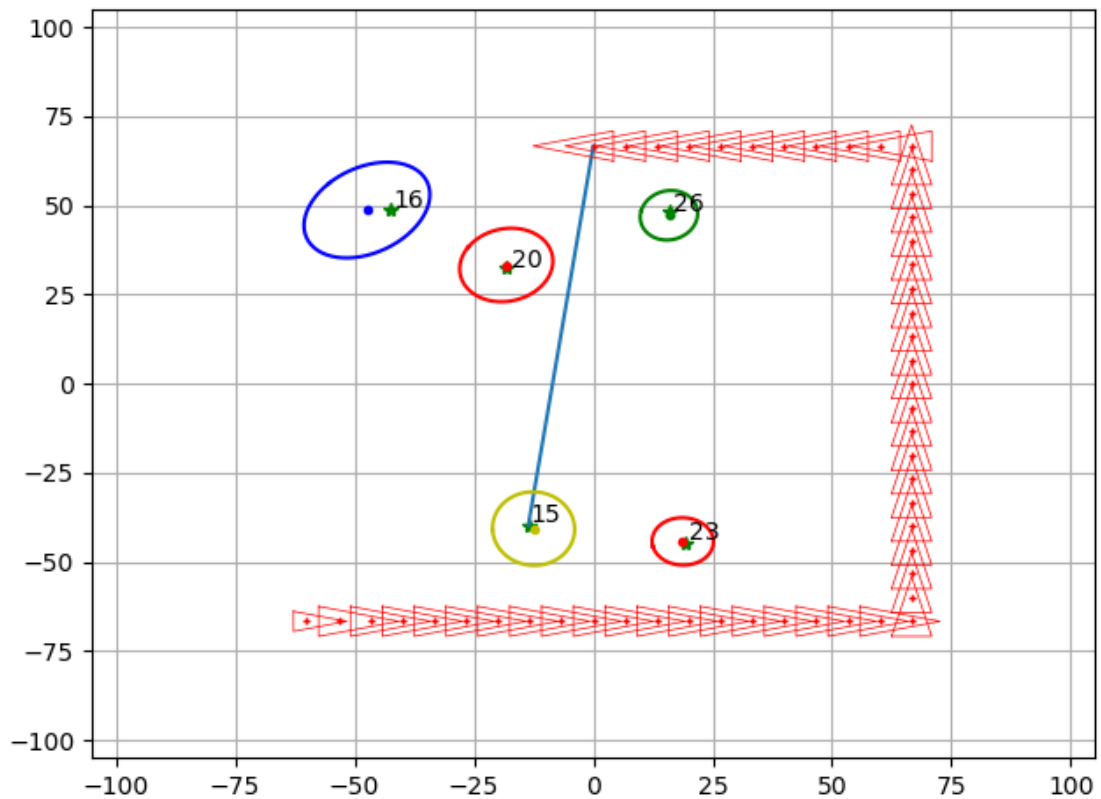


```
Iteration: 0
Estimated xEst:
[[ 23.676]
 [-50.071]]
Estimated PEst:
[[ 65.864  -9.774]
 [ -9.774 115.247]]
--------------------------------
Iteration: 5
```

```
Estimated xEst:
[[ 23.676]
 [-50.071]
 [-65.682]
 [ 47.731]
 [-10.107]
 [-42.036]
 [ 38.702]
 [ 31.415]
 [ -3.238]
 [ 22.936]]
Estimated PEst:
[[  65.864   -9.774    0.        0.        0.        0.        0.        0.
      0.        0.    ]
 [  -9.774  115.247    0.        0.        0.        0.        0.        0.
      0.        0.    ]
 [    0.       0.     195.496    6.532    0.        0.        0.        0.
      0.        0.    ]
 [    0.       0.       6.532   64.324    0.        0.        0.        0.
      0.        0.    ]
 [    0.       0.       0.        0.      28.039    7.267    0.        0.
      0.        0.    ]
 [    0.       0.       0.        0.       7.267   18.275    0.        0.
      0.        0.    ]
 [    0.       0.       0.        0.        0.        0.     173.556 -102.804
      0.        0.    ]
 [    0.       0.       0.        0.        0.        0.    -102.804  160.467
      0.        0.    ]
 [    0.       0.       0.        0.        0.        0.        0.        0.
    133.538  -36.29 ]
 [    0.       0.       0.        0.        0.        0.        0.        0.
    -36.29    82.939]]
----------------------------------
Iteration: 10
Estimated xEst:
[[ 23.676]
 [-50.071]
 [-62.357]
 [ 49.037]
 [-10.107]
 [-42.036]
 [ 21.718]
 [ 50.756]
 [ -3.238]
 [ 22.936]]
Estimated PEst:
[[ 65.864  -9.774   0.       0.       0.       0.       0.       0.       0.
     0.    ]
```

```
[ -9.774 115.247   0.       0.       0.       0.       0.       0.       0.
    0.     ]
[  0.       0.      97.434    8.944    0.       0.       0.       0.       0.
    0.     ]
[  0.       0.       8.944   33.386    0.       0.       0.       0.       0.
    0.     ]
[  0.       0.       0.       0.      28.039    7.267    0.       0.       0.
    0.     ]
[  0.       0.       0.       0.       7.267   18.275    0.       0.       0.
    0.     ]
[  0.       0.       0.       0.       0.       0.      37.086 -15.549    0.
    0.     ]
[  0.       0.       0.       0.       0.       0.     -15.549   23.005    0.
    0.     ]
[  0.       0.       0.       0.       0.       0.       0.       0.     133.538
  -36.29  ]
[  0.       0.       0.       0.       0.       0.       0.       0.     -36.29
   82.939]]
--------------------------------
Iteration: 15
Estimated xEst:
[[ 19.153]
 [-47.067]
 [-62.357]
 [ 49.037]
 [-11.288]
 [-39.241]
 [ 21.252]
 [ 52.619]
 [ -2.342]
 [ 32.878]]
Estimated PEst:
[[ 29.615    5.272    0.       0.       0.       0.       0.       0.       0.
    0.     ]
 [  5.272   23.009    0.       0.       0.       0.       0.       0.       0.
    0.     ]
 [  0.       0.      97.434    8.944    0.       0.       0.       0.       0.
    0.     ]
 [  0.       0.       8.944   33.386    0.       0.       0.       0.       0.
    0.     ]
 [  0.       0.       0.       0.       8.336    3.799    0.       0.       0.
    0.     ]
 [  0.       0.       0.       0.       3.799   13.864    0.       0.       0.
    0.     ]
 [  0.       0.       0.       0.       0.       0.      27.299 -10.249    0.
    0.     ]
 [  0.       0.       0.       0.       0.       0.     -10.249   15.115    0.
    0.     ]
```

```
[ 0.      0.      0.      0.      0.      0.      0.      0.      62.786
  -10.99 ]
 [ 0.      0.      0.      0.      0.      0.      0.      0.      -10.99
   36.343]]
----------------------------------
Iteration: 20
Estimated xEst:
[[ 19.21 ]
 [-47.37 ]
 [-59.693]
 [ 43.746]
 [-13.491]
 [-41.81 ]
 [ 21.533]
 [ 50.524]
 [ -2.342]
 [ 32.878]]
Estimated PEst:
[[ 10.233   5.225   0.      0.      0.      0.      0.      0.      0.
    0.   ]
 [  5.225   9.757   0.      0.      0.      0.      0.      0.      0.
    0.   ]
 [  0.      0.     64.346  10.692   0.      0.      0.      0.      0.
    0.   ]
 [  0.      0.     10.692  24.422   0.      0.      0.      0.      0.
    0.   ]
 [  0.      0.      0.      0.      4.95    0.754   0.      0.      0.
    0.   ]
 [  0.      0.      0.      0.      0.754   9.476   0.      0.      0.
    0.   ]
 [  0.      0.      0.      0.      0.      0.     23.825  -8.142   0.
    0.   ]
 [  0.      0.      0.      0.      0.      0.     -8.142  12.405   0.
    0.   ]
 [  0.      0.      0.      0.      0.      0.      0.      0.      62.786
  -10.99 ]
 [  0.      0.      0.      0.      0.      0.      0.      0.      -10.99
   36.343]]
----------------------------------
Iteration: 25
Estimated xEst:
[[ 16.959]
 [-47.73 ]
 [-59.7  ]
 [ 48.97 ]
 [-13.491]
 [-41.81 ]
 [ 19.636]
```

```
 [ 50.606]
 [ -7.71 ]
 [ 33.154]]
Estimated PEst:
[[ 3.93   2.29   0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 2.29   7.607  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     37.702 10.021 0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     10.021 17.298 0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     4.95   0.754  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     0.754  9.476  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     20.829 -6.321 0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     -6.321 10.287 0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     43.003 -3.853]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     -3.853 22.796]]
----------------------------------
Iteration: 30
Estimated xEst:
[[ 17.37 ]
 [-45.866]
 [-53.126]
 [ 50.505]
 [-13.491]
 [-41.81 ]
 [ 18.776]
 [ 48.798]
 [ -7.71 ]
 [ 33.154]]
Estimated PEst:
[[ 2.897  0.758  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.758  4.881  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     27.279 9.036  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     9.036  14.015 0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     4.95   0.754  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     0.754  9.476  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     16.362 -3.732 0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     -3.732 7.493  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     43.003 -3.853]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     -3.853 22.796]]
----------------------------------
Iteration: 35
Estimated xEst:
[[ 17.571]
 [-44.047]
 [-52.08 ]
 [ 47.905]
 [-13.491]
 [-41.81 ]
 [ 18.776]
```

```
 [ 48.798]
 [ -7.71 ]
 [ 33.154]]
Estimated PEst:
[[ 2.200e+00 -1.002e-02  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [-1.002e-02  3.031e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  2.136e+01  8.213e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  8.213e+00  1.217e+01  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  4.950e+00  7.535e-01
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  7.535e-01  9.476e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   1.636e+01 -3.732e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
  -3.732e+00  7.493e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  4.300e+01 -3.853e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00 -3.853e+00  2.280e+01]]
----------------------------------
Iteration: 40
Estimated xEst:
[[ 17.571]
 [-44.047]
 [-52.08 ]
 [ 47.905]
 [-14.039]
 [-41.721]
 [ 18.051]
 [ 48.418]
 [-10.091]
 [ 36.081]]
Estimated PEst:
[[ 2.200e+00 -1.002e-02  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [-1.002e-02  3.031e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  2.136e+01  8.213e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  8.213e+00  1.217e+01  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  4.604e+00  6.704e-01
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
```

```
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  6.704e-01  8.479e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   1.188e+01 -1.398e+00  0.000e+00  0.000e+00]
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
  -1.398e+00  5.354e+00  0.000e+00  0.000e+00]
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  3.145e+01  7.148e-01]
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  7.148e-01  1.774e+01]]
---------------------------------
Iteration: 45
Estimated xEst:
[[ 17.571]
 [-44.047]
 [-47.734]
 [ 48.61 ]
 [-13.299]
 [-42.38 ]
 [ 17.175]
 [ 48.235]
 [-11.958]
 [ 36.508]]
Estimated PEst:
[[ 2.200e+00 -1.002e-02  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [-1.002e-02  3.031e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  1.695e+01  7.365e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  7.365e+00  1.103e+01  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  4.295e+00  5.943e-01
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  5.943e-01  7.802e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   1.087e+01 -9.806e-01  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
  -9.806e-01  4.932e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  2.424e+01  2.325e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  2.325e+00  1.509e+01]]
---------------------------------
Iteration: 50
Estimated xEst:
[[ 17.886]
```

```
 [-44.409]
 [-47.734]
 [ 48.61 ]
 [-12.848]
 [-41.423]
 [ 17.175]
 [ 48.235]
 [-14.98 ]
 [ 36.155]]
Estimated PEst:
[[ 2.126e+00 -3.109e-03  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [-3.109e-03  2.797e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  1.695e+01  7.365e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  7.365e+00  1.103e+01  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  3.780e+00  4.160e-01
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  4.160e-01  6.700e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   1.087e+01 -9.806e-01  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
  -9.806e-01  4.932e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  1.663e+01  3.156e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  3.156e+00  1.198e+01]]
--------------------------------
Iteration: 55
Estimated xEst:
[[ 18.199]
 [-44.099]
 [-47.734]
 [ 48.61 ]
 [-12.601]
 [-42.482]
 [ 15.767]
 [ 48.683]
 [-19.06 ]
 [ 35.109]]
Estimated PEst:
[[ 1.984  0.023  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.023  2.507  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.    16.95   7.365  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     7.365 11.025  0.     0.     0.     0.     0.     0.    ]
```

```
[ 0.     0.     0.     0.     3.569 0.336  0.      0.      0.      0.    ]
[ 0.     0.     0.     0.     0.336 6.26   0.      0.      0.      0.    ]
[ 0.     0.     0.     0.     0.     0.     9.693 -0.673  0.      0.    ]
[ 0.     0.     0.     0.     0.     0.    -0.673  4.595  0.      0.    ]
[ 0.     0.     0.     0.     0.     0.     0.      0.    13.879  2.928]
[ 0.     0.     0.     0.     0.     0.     0.      0.     2.928 10.878]]
----------------------------------
Iteration: 60
Estimated xEst:
[[ 18.309]
 [-44.302]
 [-47.734]
 [ 48.61 ]
 [-12.239]
 [-42.023]
 [ 16.266]
 [ 47.268]
 [-17.473]
 [ 34.358]]
Estimated PEst:
[[ 1.92   0.028  0.      0.      0.      0.      0.      0.      0.      0.    ]
 [ 0.028  2.399  0.      0.      0.      0.      0.      0.      0.      0.    ]
 [ 0.     0.    16.95   7.365  0.      0.      0.      0.      0.      0.    ]
 [ 0.     0.     7.365 11.025  0.      0.      0.      0.      0.      0.    ]
 [ 0.     0.     0.      0.     3.231  0.169  0.      0.      0.      0.    ]
 [ 0.     0.     0.      0.     0.169  5.55   0.      0.      0.      0.    ]
 [ 0.     0.     0.      0.     0.      0.     8.446 -0.527  0.      0.    ]
 [ 0.     0.     0.      0.     0.      0.    -0.527  4.292  0.      0.    ]
 [ 0.     0.     0.      0.     0.      0.     0.      0.    11.703  2.592]
 [ 0.     0.     0.      0.     0.      0.     0.      0.     2.592  9.973]]
----------------------------------
Iteration: 65
Estimated xEst:
[[ 18.719]
 [-44.561]
 [-47.734]
 [ 48.61 ]
 [-12.239]
 [-42.023]
 [ 16.266]
 [ 47.268]
 [-17.498]
 [ 34.63 ]]
Estimated PEst:
[[ 1.788e+00 -1.552e-03  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [-1.552e-03  2.181e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
```

```
[ 0.000e+00   0.000e+00   1.695e+01   7.365e+00   0.000e+00   0.000e+00
  0.000e+00   0.000e+00   0.000e+00   0.000e+00]
[ 0.000e+00   0.000e+00   7.365e+00   1.103e+01   0.000e+00   0.000e+00
  0.000e+00   0.000e+00   0.000e+00   0.000e+00]
[ 0.000e+00   0.000e+00   0.000e+00   0.000e+00   3.231e+00   1.693e-01
  0.000e+00   0.000e+00   0.000e+00   0.000e+00]
[ 0.000e+00   0.000e+00   0.000e+00   0.000e+00   1.693e-01   5.550e+00
  0.000e+00   0.000e+00   0.000e+00   0.000e+00]
[ 0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00
  8.446e+00  -5.275e-01   0.000e+00   0.000e+00]
[ 0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00
 -5.275e-01   4.292e+00   0.000e+00   0.000e+00]
[ 0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00
  0.000e+00   0.000e+00   8.720e+00   1.914e+00]
[ 0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00
  0.000e+00   0.000e+00   1.914e+00   8.414e+00]]
--------------------------------
Iteration: 70
Estimated xEst:
[[ 18.779]
 [-44.317]
 [-47.734]
 [ 48.61 ]
 [-12.239]
 [-42.023]
 [ 14.971]
 [ 47.663]
 [-18.413]
 [ 34.44 ]]
Estimated PEst:
[[ 1.751e+00  -1.183e-02   0.000e+00   0.000e+00   0.000e+00   0.000e+00
   0.000e+00   0.000e+00   0.000e+00   0.000e+00]
 [-1.183e-02   2.118e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00
   0.000e+00   0.000e+00   0.000e+00   0.000e+00]
 [ 0.000e+00   0.000e+00   1.695e+01   7.365e+00   0.000e+00   0.000e+00
   0.000e+00   0.000e+00   0.000e+00   0.000e+00]
 [ 0.000e+00   0.000e+00   7.365e+00   1.103e+01   0.000e+00   0.000e+00
   0.000e+00   0.000e+00   0.000e+00   0.000e+00]
 [ 0.000e+00   0.000e+00   0.000e+00   0.000e+00   3.231e+00   1.693e-01
   0.000e+00   0.000e+00   0.000e+00   0.000e+00]
 [ 0.000e+00   0.000e+00   0.000e+00   0.000e+00   1.693e-01   5.550e+00
   0.000e+00   0.000e+00   0.000e+00   0.000e+00]
 [ 0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00
   5.994e+00  -4.165e-01   0.000e+00   0.000e+00]
 [ 0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00
  -4.165e-01   3.338e+00   0.000e+00   0.000e+00]
 [ 0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00   0.000e+00
   0.000e+00   0.000e+00   7.652e+00   1.567e+00]
```

```
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  1.567e+00  7.753e+00]]
----------------------------------
Iteration: 75
Estimated xEst:
[[ 18.779]
 [-44.317]
 [-49.7  ]
 [ 47.797]
 [-12.239]
 [-42.023]
 [ 14.389]
 [ 47.477]
 [-18.219]
 [ 34.88 ]]
Estimated PEst:
[[ 1.751 -0.012  0.     0.     0.     0.     0.     0.     0.     0.   ]
 [-0.012  2.118  0.     0.     0.     0.     0.     0.     0.     0.   ]
 [ 0.     0.    10.855  4.361  0.     0.     0.     0.     0.     0.   ]
 [ 0.     0.     4.361  9.01   0.     0.     0.     0.     0.     0.   ]
 [ 0.     0.     0.     0.     3.231  0.169  0.     0.     0.     0.   ]
 [ 0.     0.     0.     0.     0.169  5.55   0.     0.     0.     0.   ]
 [ 0.     0.     0.     0.     0.     0.     5.478 -0.365  0.     0.   ]
 [ 0.     0.     0.     0.     0.     0.    -0.365  3.082  0.     0.   ]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     6.138  1.032]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     1.032  6.709]]
----------------------------------
Iteration: 80
Estimated xEst:
[[ 18.779]
 [-44.317]
 [-47.784]
 [ 49.553]
 [-12.155]
 [-41.825]
 [ 14.389]
 [ 47.477]
 [-18.776]
 [ 33.843]]
Estimated PEst:
[[ 1.751 -0.012  0.     0.     0.     0.     0.     0.     0.     0.   ]
 [-0.012  2.118  0.     0.     0.     0.     0.     0.     0.     0.   ]
 [ 0.     0.     9.152  3.476  0.     0.     0.     0.     0.     0.   ]
 [ 0.     0.     3.476  8.324  0.     0.     0.     0.     0.     0.   ]
 [ 0.     0.     0.     0.     3.065 -0.025  0.     0.     0.     0.   ]
 [ 0.     0.     0.     0.    -0.025  4.916  0.     0.     0.     0.   ]
 [ 0.     0.     0.     0.     0.     0.     5.478 -0.365  0.     0.   ]
 [ 0.     0.     0.     0.     0.     0.    -0.365  3.082  0.     0.   ]
```

```
 [ 0.    0.    0.    0.    0.    0.    0.    0.    5.148  0.633]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.633  5.92 ]]
----------------------------------
Iteration: 85
Estimated xEst:
[[ 18.757]
 [-44.1  ]
 [-47.416]
 [ 48.7  ]
 [-12.155]
 [-41.825]
 [ 14.389]
 [ 47.477]
 [-18.57 ]
 [ 33.557]]
Estimated PEst:
[[ 1.734 -0.021  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [-0.021  2.053  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     6.957  2.294  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     2.294  7.211  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     3.065 -0.025  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.    -0.025  4.916  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     5.478 -0.365  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.    -0.365  3.082  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     4.446  0.407]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     0.407  5.145]]
----------------------------------
Iteration: 90
Estimated xEst:
[[ 18.831]
 [-43.892]
 [-47.416]
 [ 48.7  ]
 [-12.155]
 [-41.825]
 [ 15.831]
 [ 46.837]
 [-18.57 ]
 [ 33.557]]
Estimated PEst:
[[ 1.701 -0.033  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [-0.033  1.933  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     6.957  2.294  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     2.294  7.211  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     3.065 -0.025  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.    -0.025  4.916  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     2.815  0.575  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.575  2.21   0.     0.    ]
```

```
[ 0.     0.     0.     0.     0.     0.     0.     0.     4.446  0.407]
[ 0.     0.     0.     0.     0.     0.     0.     0.     0.407  5.145]]
--------------------------------
Iteration: 95
Estimated xEst:
[[ 18.606]
 [-44.136]
 [-47.416]
 [ 48.7  ]
 [-12.155]
 [-41.825]
 [ 14.861]
 [ 46.97 ]
 [-18.227]
 [ 33.294]]
Estimated PEst:
[[ 1.669 -0.032  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [-0.032  1.823  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     6.957  2.294  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     2.294  7.211  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     3.065 -0.025  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.    -0.025  4.916  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     1.894  0.301  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.301  2.064  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     3.813  0.429]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     0.429  4.259]]
--------------------------------
```

1.3.3   Thinking about it (1)

Having completed these trials, you will be able to **answer the following questions**:

In the **one landmark** case:

- Discuss the evolution of the variables `xEst` and `Pest`, including their dimensions.

  The more the robot moves, and therefore more measurementes of the same landmark are taken, we can see that PEst values starts to decrease (At first, it goes down very fast, bu later the uncertainty is slow down). XEst starts at a close value, and quickely gets quite close to the real one. The dimensions of XEst are 2x1 anr those of PEst are 2x2. The dimensions don't var because we are working with just one landmark.

In the **five landmarks** case:

- Why and how the content of the variables `xEst` and `Pest` has change? Discuss their size.

  It's similar to what a I said before, the only difference is that now in each movement of the robot, only one landmark is observed, then only the uncertainty decreases and the positions is corrected for one landmark per robot move. xEst and PEst start with size zero, and ends with 10x1 and 10x10 respectively. This is because when whe see a new landmark, we add two rows on xEst and a 2x2 block with 0 correlation on PEst

- What structure does the matrix of covariances have? Is there any kind of correlation among the observations of different landmarks?

  It have an structure of 2*(N X N) where N is the number of landmarks. It have a diagonal that is a "block" of 2x2. The correlation between these blocks is 0.

1.3.4 Getting performance results

As is normal, the contracting company requires some information about how well our EFK implementation performs. For that, your colleagues have implemented a logger, which is meant to store some information each loop regarding the method performance and plot it at the end of its execution. **Execute the following code cells and take a look at that plots!**

```python
[19]: import pandas as pd

class Logger():
    """ Logs info about the covariance and error of a map.

        Attrs:
            n_features: Number of features in the world.
            log_error: Matrix to store the error in the fitting for each␣
 ↪landmark.
            log_det: Matrix to store the determinant of the covariance matrix␣
 ↪for each landmark.

    """
    def __init__(self, n_steps, n_features):
        """ Initializes each matrix to log the information

            Args:
                n_steps: Maximum number of steps our robot will take.
                n_features: Number of features in the world.
        """
        self.n_features = n_features
        self.log_error = np.empty((n_steps,n_features))
        self.log_det = np.empty((n_steps,n_features))

    def log(self, k: int, robot: EFKMappingRobot, Map: np.ndarray):
        """ Computes relevant info about the error and covariances.

            It is called once per loop in the demo.

            Args:
                k: Number of iteration we are at. Range: [0, n_steps)
                robot:
                Map:
        """
        for idx in range(self.n_features):
```

```python
                tid = robot.MappedLandmarks[idx,0]
                if tid <= -1:
                    self.log_error[k,idx]= np.Inf
                    self.log_det[k,idx]=np.Inf
                else:
                    self.log_det[k,idx] = np.linalg.det(robot.PEst[tid:tid+2,tid:
↪tid+2])
                    self.log_error[k,idx] = np.sqrt(np.sum((robot.xEst[tid:tid+2,0]␣
↪- Map[:,idx])**2))

    def plot(self):
        """ Plot all relevant figures. It is called at the end of the demo"""
        fig1 , ax1 =plt.subplots(1, 1, sharex=True)
        fig2 , ax2 =plt.subplots(1, 1, sharex=True)
        fig2.tight_layout()
        fig1.tight_layout()

        df1 = pd.DataFrame(data= self.log_error, columns = ['Landmark {}'.
↪format(i) for i in range(self.n_features)])
        ax1.set_title('Error between map and est')
        df1.plot(ax = ax1)
        df2 = pd.DataFrame(data=np.log(self.log_det), columns=['Landmark {}'.
↪format(i) for i in range(self.n_features)])
        ax2.set_title('Det. of covar.')
        df2.plot(ax = ax2)
```

```python
[20]:  #mode = 'step_by_step'
       mode = 'visualize_process'
       #mode = 'non_stop'

       # WORLD MAP
       # Num features/landmarks considered within the map
       nLandmarks = 5
       # Generation of the map
       MapSize = 100
       Map = MapSize*random.rand(2,nLandmarks)-MapSize/2

       # ROBOT
       # Covariances for our very bad&expensive sensor (in the system <d,theta>)
       Sigma_r = 8.0
       Sigma_theta = 7*np.pi/180
       # Initial robot pose
       xVehicleTrue = np.vstack([-MapSize/1.5, -MapSize/1.5, 0.]) # We know the exact␣
↪robot pose at any moment

       robot = EFKMappingRobot(xVehicleTrue, Sigma_r, Sigma_theta, nLandmarks)
```

```
nSteps=100
logger = Logger(n_features=nLandmarks, n_steps=nSteps)

demo_ekf_mapping(robot,
                 Map,
                 nLandmarks,
                 logger=logger,
                 mode='non_stop',
                 nSteps=nSteps)
```
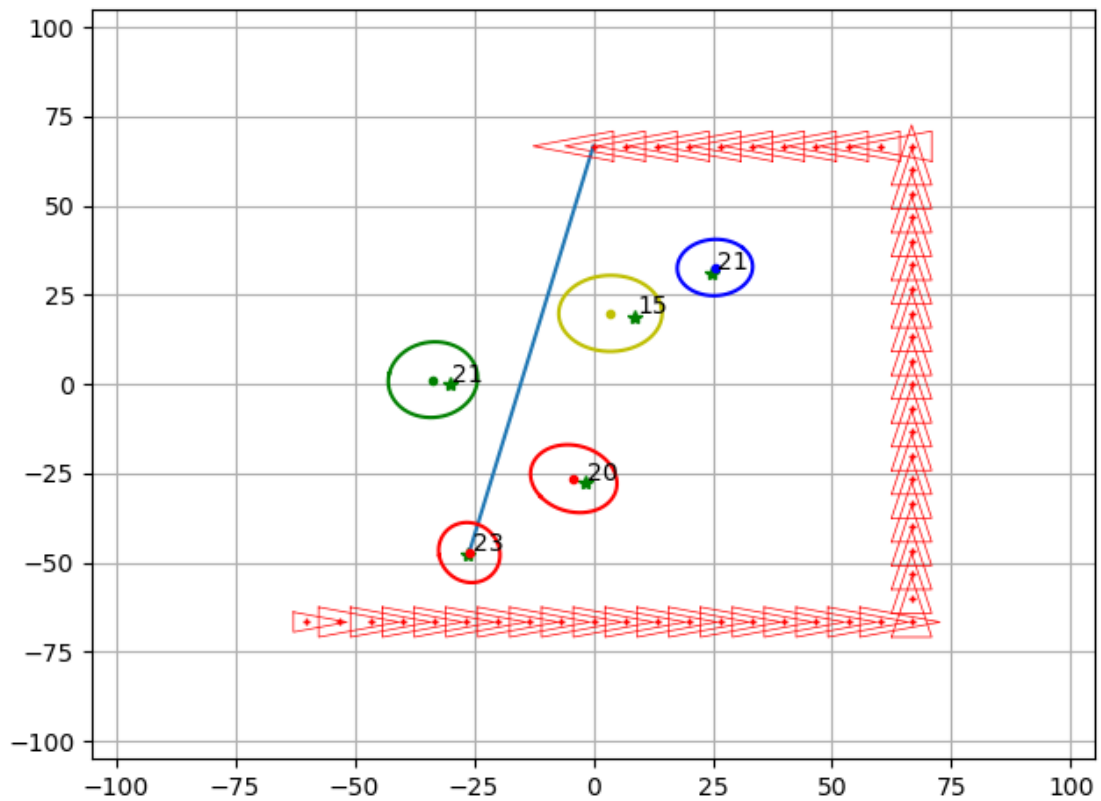


```
Iteration: 0
Estimated xEst:
[[-12.816]
 [-21.656]]
Estimated PEst:
[[65.917 -2.151]
 [-2.151 66.415]]
----------------------------------
Iteration: 5
Estimated xEst:
[[-12.816]
 [-21.656]
 [ 33.095]
```

```
 [ 35.514]
 [ 22.687]
 [ 16.831]
 [-40.615]
 [  3.564]]
Estimated PEst:
[[  65.917   -2.151    0.       0.       0.       0.       0.       0.   ]
 [  -2.151   66.415    0.       0.       0.       0.       0.       0.   ]
 [   0.       0.     184.872 -110.125    0.       0.       0.       0.   ]
 [   0.       0.    -110.125  164.333    0.       0.       0.       0.   ]
 [   0.       0.       0.       0.     134.436  -66.94     0.       0.   ]
 [   0.       0.       0.       0.     -66.94   127.617    0.       0.   ]
 [   0.       0.       0.       0.       0.       0.      24.156   -0.618]
 [   0.       0.       0.       0.       0.       0.      -0.618   21.502]]
-----------------------------------
Iteration: 10
Estimated xEst:
[[-12.816]
 [-21.656]
 [ 33.095]
 [ 35.514]
 [ 10.433]
 [ 22.98 ]
 [-37.516]
 [  4.044]
 [-22.429]
 [-42.212]]
Estimated PEst:
[[  65.917   -2.151    0.       0.       0.       0.       0.       0.
     0.       0.   ]
 [  -2.151   66.415    0.       0.       0.       0.       0.       0.
     0.       0.   ]
 [   0.       0.     184.872 -110.125    0.       0.       0.       0.
     0.       0.   ]
 [   0.       0.    -110.125  164.333    0.       0.       0.       0.
     0.       0.   ]
 [   0.       0.       0.       0.      64.383  -26.848    0.       0.
     0.       0.   ]
 [   0.       0.       0.       0.     -26.848   54.432    0.       0.
     0.       0.   ]
 [   0.       0.       0.       0.       0.       0.      18.192   -0.372
     0.       0.   ]
 [   0.       0.       0.       0.       0.       0.      -0.372   16.094
     0.       0.   ]
 [   0.       0.       0.       0.       0.       0.       0.       0.
     8.597    5.833]
 [   0.       0.       0.       0.       0.       0.       0.       0.
     5.833   18.228]]
```

```
-----------------------------------
Iteration: 15
Estimated xEst:
[[ -9.621]
 [-23.311]
 [ 22.587]
 [ 31.994]
 [ 10.433]
 [ 22.98 ]
 [-37.516]
 [  4.044]
 [-24.862]
 [-43.11 ]]
Estimated PEst:
[[ 21.331   1.436   0.      0.      0.      0.      0.      0.      0.
    0.   ]
 [  1.436  32.24    0.      0.      0.      0.      0.      0.      0.
    0.   ]
 [  0.      0.     53.354 -21.514   0.      0.      0.      0.      0.
    0.   ]
 [  0.      0.    -21.514  36.06    0.      0.      0.      0.      0.
    0.   ]
 [  0.      0.      0.      0.     64.383 -26.848   0.      0.      0.
    0.   ]
 [  0.      0.      0.      0.    -26.848  54.432   0.      0.      0.
    0.   ]
 [  0.      0.      0.      0.      0.      0.     18.192  -0.372   0.
    0.   ]
 [  0.      0.      0.      0.      0.      0.     -0.372  16.094   0.
    0.   ]
 [  0.      0.      0.      0.      0.      0.      0.      0.      2.791
    0.482]
 [  0.      0.      0.      0.      0.      0.      0.      0.      0.482
    8.928]]
-----------------------------------
Iteration: 20
Estimated xEst:
[[ -9.621]
 [-23.311]
 [ 26.482]
 [ 30.962]
 [ 11.095]
 [ 19.45 ]
 [-37.485]
 [  3.069]
 [-25.135]
 [-43.887]]
Estimated PEst:
```

```
[[ 21.331   1.436    0.        0.        0.        0.        0.        0.        0.
    0.    ]
 [  1.436  32.24     0.        0.        0.        0.        0.        0.        0.
    0.    ]
 [  0.      0.      30.288    -8.93     0.        0.        0.        0.        0.
    0.    ]
 [  0.      0.      -8.93     17.702    0.        0.        0.        0.        0.
    0.    ]
 [  0.      0.       0.        0.       40.216  -11.223    0.        0.        0.
    0.    ]
 [  0.      0.       0.        0.      -11.223   29.051    0.        0.        0.
    0.    ]
 [  0.      0.       0.        0.        0.        0.       14.937    0.064    0.
    0.    ]
 [  0.      0.       0.        0.        0.        0.        0.064   12.972    0.
    0.    ]
 [  0.      0.       0.        0.        0.        0.        0.        0.        2.556
    0.048]
 [  0.      0.       0.        0.        0.        0.        0.        0.        0.048
    6.725]]
----------------------------------
Iteration: 25
Estimated xEst:
[[  0.145]
 [-24.146]
 [ 24.462]
 [ 33.171]
 [ 11.095]
 [ 19.45 ]
 [-37.485]
 [  3.069]
 [-25.603]
 [-44.457]]
Estimated PEst:
[[  7.212  -1.276    0.        0.        0.        0.        0.        0.        0.
    0.    ]
 [ -1.276  12.119    0.        0.        0.        0.        0.        0.        0.
    0.    ]
 [  0.      0.      24.466    -6.012    0.        0.        0.        0.        0.
    0.    ]
 [  0.      0.      -6.012   13.68      0.        0.        0.        0.        0.
    0.    ]
 [  0.      0.       0.        0.       40.216  -11.223    0.        0.        0.
    0.    ]
 [  0.      0.       0.        0.      -11.223   29.051    0.        0.        0.
    0.    ]
 [  0.      0.       0.        0.        0.        0.       14.937    0.064    0.
    0.    ]
```

```
[ 0.      0.      0.      0.      0.      0.      0.064  12.972   0.
    0.   ]
 [ 0.      0.      0.      0.      0.      0.      0.      0.      2.439
   -0.042]
 [ 0.      0.      0.      0.      0.      0.      0.      0.     -0.042
    5.712]]
--------------------------------
Iteration: 30
Estimated xEst:
[[  0.145]
 [-24.146]
 [ 24.462]
 [ 33.171]
 [ 10.973]
 [ 19.468]
 [-32.42 ]
 [  2.026]
 [-25.777]
 [-44.792]]
Estimated PEst:
[[ 7.212 -1.276  0.      0.      0.      0.      0.      0.      0.      0.    ]
 [-1.276 12.119  0.      0.      0.      0.      0.      0.      0.      0.    ]
 [ 0.     0.     24.466 -6.012  0.      0.      0.      0.      0.      0.    ]
 [ 0.     0.     -6.012 13.68   0.      0.      0.      0.      0.      0.    ]
 [ 0.     0.      0.      0.     28.354 -4.765  0.      0.      0.      0.    ]
 [ 0.     0.      0.      0.     -4.765 19.402  0.      0.      0.      0.    ]
 [ 0.     0.      0.      0.      0.      0.     11.286  0.976  0.      0.    ]
 [ 0.     0.      0.      0.      0.      0.      0.976 10.044  0.      0.    ]
 [ 0.     0.      0.      0.      0.      0.      0.      0.      2.26  -0.064]
 [ 0.     0.      0.      0.      0.      0.      0.      0.     -0.064  4.684]]
--------------------------------
Iteration: 35
Estimated xEst:
[[  0.39 ]
 [-24.429]
 [ 23.835]
 [ 33.531]
 [ 10.973]
 [ 19.468]
 [-32.451]
 [  1.909]
 [-25.788]
 [-45.652]]
Estimated PEst:
[[ 6.477 -0.936  0.      0.      0.      0.      0.      0.      0.      0.    ]
 [-0.936 10.237  0.      0.      0.      0.      0.      0.      0.      0.    ]
 [ 0.     0.     17.453 -2.488  0.      0.      0.      0.      0.      0.    ]
 [ 0.     0.     -2.488  9.266  0.      0.      0.      0.      0.      0.    ]
```

```
[ 0.     0.     0.     0.    28.354 -4.765  0.     0.     0.     0.    ]
[ 0.     0.     0.     0.    -4.765 19.402  0.     0.     0.     0.    ]
[ 0.     0.     0.     0.     0.     0.    10.077  1.205  0.     0.    ]
[ 0.     0.     0.     0.     0.     0.     1.205  9.155  0.     0.    ]
[ 0.     0.     0.     0.     0.     0.     0.     0.     2.184 -0.053]
[ 0.     0.     0.     0.     0.     0.     0.     0.    -0.053  4.393]]
--------------------------------
Iteration: 40
Estimated xEst:
[[ -2.554]
 [-24.196]
 [ 23.835]
 [ 33.531]
 [ 10.973]
 [ 19.468]
 [-32.451]
 [  1.909]
 [-26.124]
 [-45.974]]
Estimated PEst:
[[ 5.015e+00 -3.367e-01  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [-3.367e-01  7.264e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  1.745e+01 -2.488e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00 -2.488e+00  9.266e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  2.835e+01 -4.765e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00 -4.765e+00  1.940e+01
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   1.008e+01  1.205e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   1.205e+00  9.155e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  2.047e+00 -1.720e-02]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00 -1.720e-02  4.100e+00]]
--------------------------------
Iteration: 45
Estimated xEst:
[[ -2.554]
 [-24.196]
 [ 23.668]
 [ 32.254]
 [  8.694]
```

```
[ 20.481]
[-31.939]
[  1.904]
[-26.268]
[-45.891]]
Estimated PEst:
[[ 5.015e+00 -3.367e-01  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [-3.367e-01  7.264e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  1.490e+01 -1.492e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00 -1.492e+00  8.075e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  2.140e+01 -1.664e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00 -1.664e+00  1.515e+01
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   8.931e+00  1.303e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   1.303e+00  8.634e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  1.925e+00 -4.000e-03]
 [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00 -4.000e-03  3.854e+00]]
--------------------------------
Iteration: 50
Estimated xEst:
[[ -2.481]
 [-24.447]
 [ 24.457]
 [ 33.249]
 [  6.729]
 [ 20.721]
 [-32.23 ]
 [  2.238]
 [-26.312]
 [-46.758]]
Estimated PEst:
[[ 4.652e+00 -2.643e-01  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [-2.643e-01  6.625e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00  1.242e+01 -9.543e-01  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
 [ 0.000e+00  0.000e+00 -9.543e-01  7.189e+00  0.000e+00  0.000e+00
   0.000e+00  0.000e+00  0.000e+00  0.000e+00]
```

```
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  1.667e+01 -5.082e-01
  0.000e+00  0.000e+00  0.000e+00  0.000e+00]
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00 -5.082e-01  1.261e+01
  0.000e+00  0.000e+00  0.000e+00  0.000e+00]
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
  7.952e+00  1.284e+00  0.000e+00  0.000e+00]
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
  1.284e+00  8.168e+00  0.000e+00  0.000e+00]
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
  0.000e+00  0.000e+00  1.869e+00 -7.258e-03]
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
  0.000e+00  0.000e+00 -7.258e-03  3.742e+00]]
---------------------------------
Iteration: 55
Estimated xEst:
[[ -2.481]
 [-24.447]
 [ 24.457]
 [ 33.249]
 [  3.411]
 [ 22.452]
 [-31.384]
 [  1.267]
 [-26.434]
 [-46.611]]
Estimated PEst:
[[ 4.652 -0.264  0.     0.     0.     0.     0.     0.     0.     0.   ]
 [-0.264  6.625  0.     0.     0.     0.     0.     0.     0.     0.   ]
 [ 0.     0.    12.419 -0.954  0.     0.     0.     0.     0.     0.   ]
 [ 0.     0.    -0.954  7.189  0.     0.     0.     0.     0.     0.   ]
 [ 0.     0.     0.     0.    13.435 -0.107  0.     0.     0.     0.   ]
 [ 0.     0.     0.     0.    -0.107 10.785  0.     0.     0.     0.   ]
 [ 0.     0.     0.     0.     0.     0.     5.875  1.027  0.     0.   ]
 [ 0.     0.     0.     0.     0.     0.     1.027  6.997  0.     0.   ]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     1.818 -0.022]
 [ 0.     0.     0.     0.     0.     0.     0.     0.    -0.022  3.636]]
---------------------------------
Iteration: 60
Estimated xEst:
[[ -2.481]
 [-24.447]
 [ 24.109]
 [ 35.613]
 [  5.942]
 [ 21.308]
 [-33.057]
 [  0.546]
 [-26.368]
```

```
 [-46.962]]
Estimated PEst:
[[ 4.652 -0.264  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [-0.264  6.625  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.    10.133 -0.95   0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.    -0.95   6.246  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.    11.114 -0.052  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.    -0.052  9.286  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     4.969  0.837  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.837  6.371  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     1.773 -0.041]
 [ 0.     0.     0.     0.     0.     0.     0.     0.    -0.041  3.535]]
----------------------------------
Iteration: 65
Estimated xEst:
[[ -3.124]
 [-23.662]
 [ 26.923]
 [ 33.303]
 [  6.745]
 [ 21.034]
 [-34.213]
 [ -0.489]
 [-26.368]
 [-46.962]]
Estimated PEst:
[[ 4.373 -0.301  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [-0.301  6.161  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     7.446 -0.928  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.    -0.928  4.761  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     9.462 -0.065  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.    -0.065  8.015  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     4.602  0.704  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.704  6.085  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     1.773 -0.041]
 [ 0.     0.     0.     0.     0.     0.     0.     0.    -0.041  3.535]]
----------------------------------
Iteration: 70
Estimated xEst:
[[ -2.136]
 [-23.708]
 [ 26.461]
 [ 33.374]
 [  2.48 ]
 [ 19.873]
 [-33.963]
 [  0.556]
 [-26.368]
```

```
 [-46.962]]
Estimated PEst:
[[ 4.139 -0.34   0.     0.     0.     0.     0.     0.     0.     0.    ]
 [-0.34   5.77   0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     6.643 -0.852  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.    -0.852  4.05   0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     7.297 -0.009  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.    -0.009  6.255  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     4.296  0.553  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.553  5.809  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     1.773 -0.041]
 [ 0.     0.     0.     0.     0.     0.     0.     0.    -0.041  3.535]]
----------------------------------
Iteration: 75
Estimated xEst:
[[ -2.405]
 [-24.287]
 [ 26.461]
 [ 33.374]
 [  2.532]
 [ 19.512]
 [-33.883]
 [  0.88 ]
 [-26.099]
 [-46.648]]
Estimated PEst:
[[ 3.973 -0.403  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [-0.403  5.429  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     6.643 -0.852  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.    -0.852  4.05   0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     5.968 -0.055  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.    -0.055  5.322  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     4.033  0.424  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.424  5.559  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     1.744 -0.074]
 [ 0.     0.     0.     0.     0.     0.     0.     0.    -0.074  3.419]]
----------------------------------
Iteration: 80
Estimated xEst:
[[ -2.642]
 [-26.388]
 [ 25.704]
 [ 33.418]
 [  2.532]
 [ 19.512]
 [-33.883]
 [  0.88 ]
 [-26.082]
```

```
 [-46.904]]
Estimated PEst:
[[ 3.73  -0.514  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [-0.514  4.855  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     5.784 -0.563  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.    -0.563  3.682  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     5.968 -0.055  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.    -0.055  5.322  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     4.033  0.424  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.424  5.559  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     1.697 -0.133]
 [ 0.     0.     0.     0.     0.     0.     0.     0.    -0.133  3.201]]
------------------------------------
Iteration: 85
Estimated xEst:
[[ -2.642]
 [-26.388]
 [ 26.464]
 [ 33.621]
 [  2.532]
 [ 19.512]
 [-33.883]
 [  0.88 ]
 [-26.162]
 [-46.83 ]]
Estimated PEst:
[[ 3.73  -0.514  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [-0.514  4.855  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     3.552  0.095  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.095  2.709  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     5.968 -0.055  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.    -0.055  5.322  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     4.033  0.424  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.424  5.559  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     1.679 -0.155]
 [ 0.     0.     0.     0.     0.     0.     0.     0.    -0.155  3.086]]
------------------------------------
Iteration: 90
Estimated xEst:
[[ -3.567]
 [-26.89 ]
 [ 25.565]
 [ 33.097]
 [  2.588]
 [ 19.468]
 [-33.883]
 [  0.88 ]
 [-26.162]
```

```
[-46.83 ]]
Estimated PEst:
[[ 3.45  -0.53   0.     0.     0.     0.     0.     0.     0.     0.    ]
 [-0.53   4.054  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     2.929  0.111  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.111  2.597  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     5.432 -0.025  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.    -0.025  4.897  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     4.033  0.424  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.424  5.559  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     1.679 -0.155]
 [ 0.     0.     0.     0.     0.     0.     0.     0.    -0.155  3.086]]
----------------------------------
Iteration: 95
Estimated xEst:
[[ -4.215]
 [-27.019]
 [ 25.475]
 [ 32.677]
 [  2.588]
 [ 19.468]
 [-33.224]
 [  1.156]
 [-26.094]
 [-46.799]]
Estimated PEst:
[[ 3.365 -0.52   0.     0.     0.     0.     0.     0.     0.     0.    ]
 [-0.52   3.833  0.     0.     0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     2.496  0.073  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.073  2.493  0.     0.     0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     5.432 -0.025  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.    -0.025  4.897  0.     0.     0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     3.665  0.235  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.235  4.818  0.     0.    ]
 [ 0.     0.     0.     0.     0.     0.     0.     0.     1.664 -0.165]
 [ 0.     0.     0.     0.     0.     0.     0.     0.    -0.165  2.959]]
----------------------------------
```
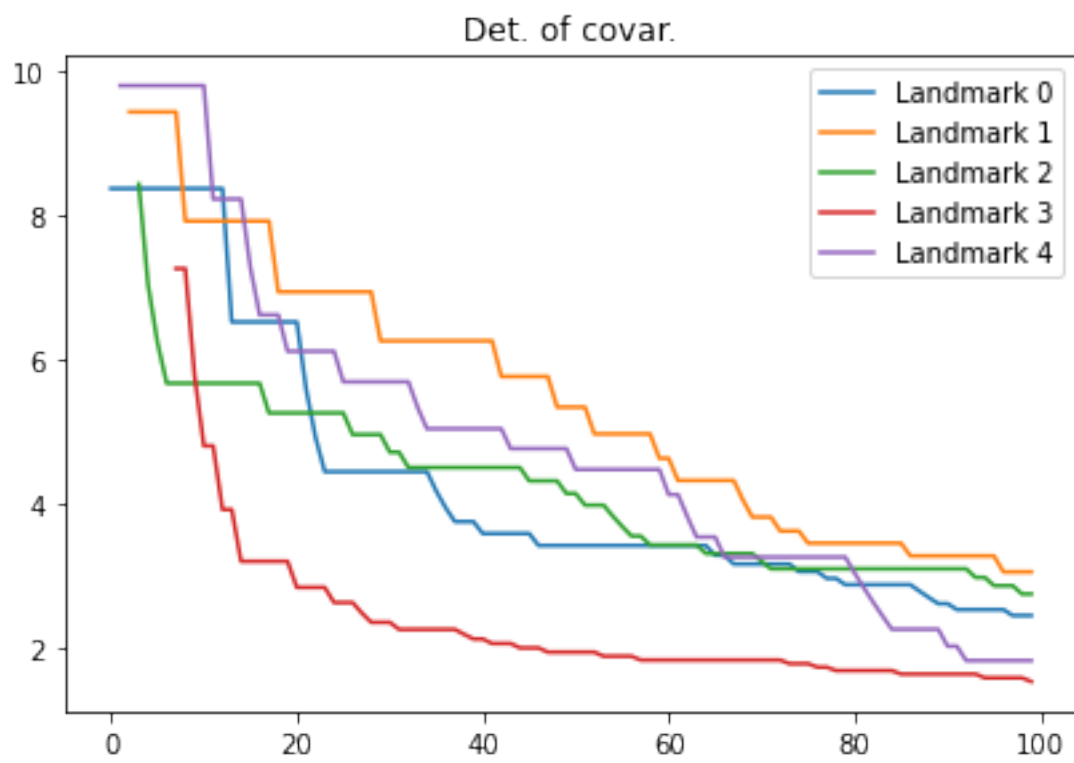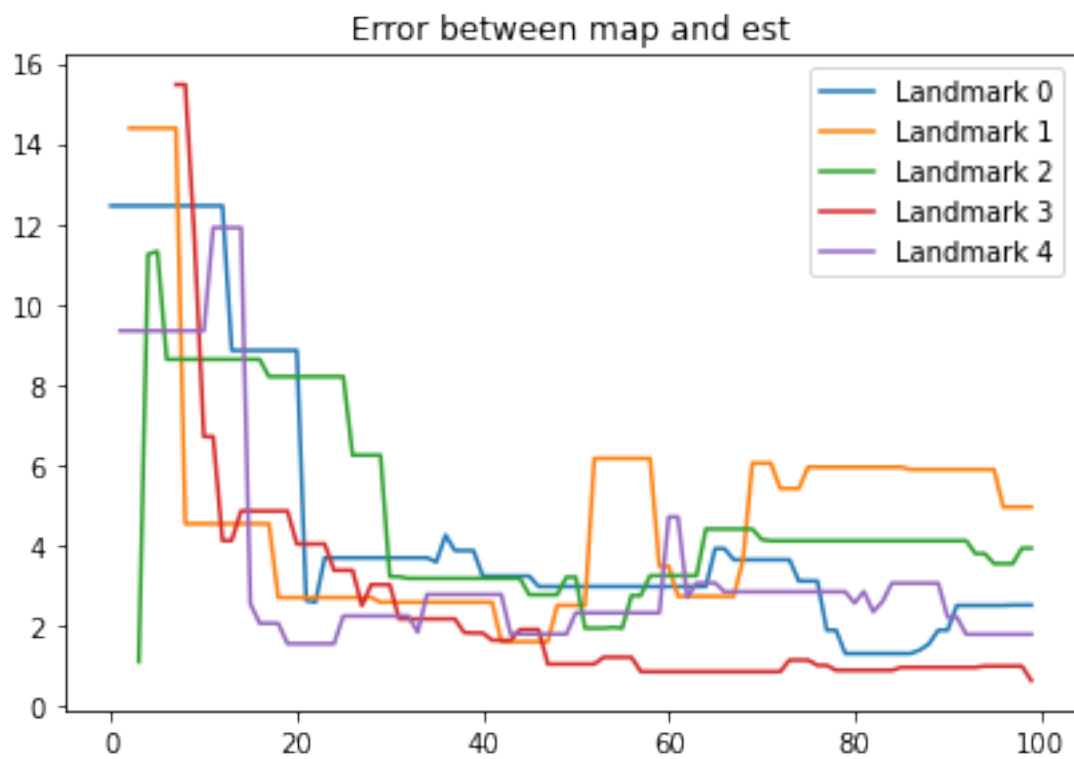
Error between map and est

Det. of covar.

1.3.5 Thinking about it (2)

Having taken a look at the logger and its output, you will be able to **answer the following questions**:

- What information is shown in the figures produced by the logger?

  The logarithm of the determinant of covariance matrices and the logarithm of the error with respect the time

- The information about the error and the determinant of the covariance is provided for first time at different iterations of the algorithm for each landmark. Is that an error? Why is this happening?

  No. This happens because we only measure one distance to one landmark in one instance of time. Therefore, the first one we measure will fall down before the last one we measure

- The error associated to each landmark not always decreases with new observations. Why could this happen?

  Because of the uncertainty of the sensor, it could happen that we may drift away from the expected position of the landmark.

- On the contrary, the determinant of the covariance matrix associated to each landmark always decreases when new observations are available. Is this an error? Why?

  No. This happens because the more measures I have of something, the less uncertainty I have. Therefore, by having more measurements and reducing uncertainty, the value of the determinant of the covariance matrix goes down.