Miniset 4

Thursday, September 29, 2022 4:14 PM

Robotic Localization and Mapping - Miniset 4
More Gaussians, Poses/Pose Uncertainty, and Uncertainty Propagation

Prof. Joshua Mangelson
Brigham Young University
(Problems Courtesy of Ryan Eustice.)

September 28, 2021

More Gaussians and Linearization Practice

MGL-Q1. Ryan has collected four points (in 2D space); let's call this set $'r'$. He has computed their mean and (biased) sample covariance to be:

$$\mu_r = \begin{bmatrix}4\\2\end{bmatrix} \quad \Sigma_{rr} = \begin{bmatrix}4 & 2\\2 & 8\end{bmatrix}.$$

And Ed has collected six points (from the same 2D space); let's call it set $'e'$, finding their mean and (biased) sample covariance to be:

$$\mu_e = \begin{bmatrix}-1\\0\end{bmatrix} \quad \Sigma_{rr} = \begin{bmatrix}6 & 3\\3 & 6\end{bmatrix}.$$

(a) If Ryan computed the sum (over all r points) of $xx^\top$, what value would he have computed? (Show work and provide a numerical answer.)

(b) If Ed computed the sum (over all e points) of $xx^\top$, what value would he have computed? (Show work and provide a numerical answer.)

We now wish to compute the mean and sample covariance of all ten points (which we will denote $e + r$).

(c) What is the mean, μ_{e+r}? (Show work and provide a numerical answer.)

(d) What is the (biased) sample variance, Σ_{e+r}? (Show work and provide a numerical answer.)

(a) $n\left[\Sigma_{xx} + \mu_x\mu_x^T\right] =$

$4\left[\begin{bmatrix}4 & 2\\2 & 8\end{bmatrix} + \begin{bmatrix}4\\2\end{bmatrix}\begin{bmatrix}4 & 2\end{bmatrix}\right] = 4\left[\begin{bmatrix}4 & 2\\2 & 8\end{bmatrix} + \begin{bmatrix}16 & 8\\8 & 4\end{bmatrix}\right]$

$\boxed{= \begin{bmatrix}80 & 40\\40 & 48\end{bmatrix}}$

(b) $6\left[\begin{bmatrix}6 & 3\\3 & 6\end{bmatrix} + \begin{bmatrix}-1\\0\end{bmatrix}\begin{bmatrix}-1 & 0\end{bmatrix}\right] = 6\left[\begin{bmatrix}6 & 3\\3 & 6\end{bmatrix} + \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}\right]$

$\boxed{= \begin{bmatrix}42 & 18\\18 & 36\end{bmatrix}}$

(c) $\frac{4}{10}\mu_r + \frac{6}{10}\mu_e$

$= \frac{4}{10}\begin{bmatrix}4\\2\end{bmatrix} + \frac{6}{10}\begin{bmatrix}-1\\0\end{bmatrix} = \boxed{\begin{bmatrix}-1\\4/5\end{bmatrix}}$

(d) $\frac{4}{10}\Sigma_{ee} + \frac{6}{10}\Sigma_{rr} = \begin{bmatrix}1.6+3.6 & .2+1.8\\.2+1.8 & 3.2+3.6\end{bmatrix} = \boxed{\begin{bmatrix}5.2 & 2\\2 & 6.8\end{bmatrix}}$

MGL-Q2. Consider a two-dimensional robot whose position is denoted as r_x and r_y. Note that the robot is a "point" robot and has no orientation. Assume a robot is estimating the position of a landmark whose position is denoted by f_x and f_y. The joint state vector is thus $[r_x, r_y, f_x, f_y]^\top$.

The robot is equipped with a sensor that measures two quantities:

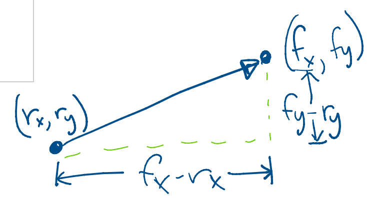
- The dot product of the vector from the robot to the landmark with the vector $[\sqrt{2}, \sqrt{2}]^\top$.
- The square of the distance between the robot and the landmark.

Each observation is corrupted by iid Gaussian noise with mean 0 and variance 1.

a) Write the observation model for the sensor.

b) What is the Jacobian of the observation model with respect to the state?

c) What is the Jacobian of the observation model with respect to the noise?



$X = \begin{bmatrix}r_x\\r_y\\f_x\\f_y\end{bmatrix}$

$Z_{t,1} = \begin{bmatrix}f_x - r_x\\f_y - r_y\end{bmatrix} \cdot \begin{bmatrix}\sqrt{2}\\\sqrt{2}\end{bmatrix}$

$Z_{t,2} = (f_y - r_y)^2 + (f_x - r_x)^2$

$\delta_t = N(0, 1)$

$+ y\sqrt{2} + \delta_{t1}$ $(f_x - r_x)\sqrt{2} + (f_y - r_y)\sqrt{2} + \delta_{t1}$

$\sim_t$

(a) $$Z_t = \begin{bmatrix} x\sqrt{2} + y\sqrt{2} + \delta_{t1} \\ x^2 + y^2 + \delta_{t2} \end{bmatrix} = \begin{bmatrix} (f_x-r_x)\sqrt{2} + (f_y-r_y)\sqrt{2} + \delta_{t1} \\ (f_x-r_x)^2 + (f_y-r_y)^2 + \delta_{t2} \end{bmatrix}$$

(b) $$\frac{\partial Z}{\partial x} = \begin{bmatrix} \frac{\partial z_1}{\partial r_x} & \frac{\partial z_1}{\partial r_y} & \frac{\partial z_1}{\partial f_x} & \frac{\partial z_1}{\partial f_y} \\ \frac{\partial z_2}{\partial r_x} & \frac{\partial z_2}{\partial r_y} & \frac{\partial z_2}{\partial f_x} & \frac{\partial z_2}{\partial f_y} \end{bmatrix} = \begin{bmatrix} -\sqrt{2} & -\sqrt{2} & \sqrt{2} & \sqrt{2} \\ 2r_x-2f_x & 2r_y-2f_y & 2f_x-2r_x & 2f_y-2r_y \end{bmatrix}$$

(c) $$\frac{\partial Z}{\partial \delta} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

There are two noise distributions (z_1 and z_2 have different noise)

Stochastic Coordinate Transformations

Rigid-body coordinate transforms are used all the time in robotics: e.g., sensors make measurements with respect to their own reference frame, which then has to be transformed to the robot frame, and from there to the world frame and...you get the point. The goal of this exercise is to help you become comfortable with coordinate frame composition.

In lecture we learned about the Smith, Self, Cheesemen coordinate frame notation for 3-DOF and 6DOF parameterizations of pose. The 6-DOF versions as described in Appendix A of Ryan Eustice's thesis are implemented in the code provided, namely:

- **ssc_head2tail**() performs the head-to-tail frome composition $x_{ik} = x_{ij} \oplus x_{jk}$.
- **ssc_inverse**() performs the inverse frame transformation $x_j i = \ominus x_{ij}$
- **ssc_tail2tail**() performs the tail-to-tail frame composition $x_{jk} = \ominus x_{ij} \oplus x_{ik}$

Each of the above functions returns the 6-DOF pose vector result of the coordinate frame operation and (optionally) the resulting analytical Jacobian. Use these functions to answer the scenario described below. For these problems, load the provided data file data.mat into your MATLAB workspace.

Suppose that we have a robot instrumented with two sensors s_1 and s_2. Sensor s_1 has a static pose in the robot coordinate frame given by the variable x_{rs_1}, and sensor s_2 has a static pose defined with respect to (w.r.t.) sensor s_1 given by variable $x_{s_1 s_2}$. These transforms are assumed to be known exactly (i.e. well-calibrated).

The robot's own pose is less certain and is expressed w.r.t. a world reference frame w. The robot's pose sampled at time t_1, x_{wr1}, has a mean and covariance given by variables μ_{wr1} and Σ_1, respectively. And at a short time later, t_2, after making an odometry move of a couple meters, the robot has a pose x_{wr2} with mean and covariance given by variables μ_{wr2} and Σ_2, respectively. The cross-covariance between the robot's pose at time t_1 and t_2 is given by the 6×6 matrix variable Σ_{12}.

SCT-Q0. **Extra Credit -** The code given to you above is Matlab code, I would rather it be in python, but haven't had time to convert it over. If you convert it to python and show results for each function verifying that the code works correctly (and give me permission to use your code in future classes), I will give you 2 points extra credit on this assignment.

PASS

SCT-Q1. What is the pose of the sensor s_1 w.r.t. sensor s_2? What is the Jacobian of the associated transformation?

```
X_prime = Xs2s1

   -0.3734
   -0.8706
   -0.2346
   -0.4647
   -0.6602
   -0.9595
```

head2tail(X_{ij}, X_{jk})

↳ X_{ik}

have X_{ij} be j defined in i

X_{jk} be k defined in j

get k defined in i

```
   -0.4647
   -0.6602
   -0.9595
```

X jk be le dan get ve defined in!

```
Jminus =
```

J_{21}

```
   -0.4534   -0.8898   -0.0523         0    0.7815   -0.4491
    0.6468   -0.2880   -0.7061   -0.2346   -0.2640    0.2514
   -0.6132    0.3540   -0.7061    0.8706   -0.2640   -0.2181
         0         0         0   -0.7266    0.7331    0.6940
         0         0         0   -0.8189   -0.4058   -0.4481
         0         0         0    0.4456    0.2576   -1.1317
```

SCT-Q2. What is the mean pose of sensor s_1 in reference frame w at time t_1? What is the Jacobian of the associated transformation? To first order, what is the covariance of s_1's pose in reference frame w at time t_1?

```
X_ws1Q2 =

  501.0000
  500.0000
   -1.0000
    0.0873
    0.1222
    0.2618


JplusQ2 =

    1.0000         0         0         0   -1.0000         0    1.0000         0         0         0         0         0
         0    1.0000         0    1.0000         0    1.0000         0    1.0000         0         0         0         0
         0         0    1.0000         0   -1.0000         0         0         0    1.0000         0         0         0
         0         0         0    0.9732    0.2608         0         0         0         0    1.0000   -0.0000         0
         0         0         0   -0.2588    0.9659         0         0         0         0         0    1.0000    0.0000
         0         0         0    0.1186    0.0318    1.0000         0         0         0         0   -0.0000    1.0000


P_lcQ2 =

   99.9998   14.1355   22.7650    0.0793   -0.0224   -0.4068
   14.1355  100.3467  -76.3478    0.0507    0.2078    0.1908
   22.7650  -76.3478   99.9501   -0.1411    0.0619   -0.2292
    0.0793    0.0507   -0.1411    0.0027   -0.0001    0.0019
   -0.0224    0.2078    0.0619   -0.0001    0.0028    0.0024
   -0.4068    0.1908   -0.2292    0.0019    0.0024    0.0080
```

SCT-Q3. What is the mean pose, Jacobian, and first-order covariance of sensor s_2's pose in reference frame w at time t_1? (Hint: think chain rule.)

```
X_ws2Q3 =

  500.8247
  499.0763
   -1.2618
   -0.5224
   -0.5180
   -0.6708


JplusQ3 =

    1.0000         0         0         0   -1.2618    0.9237    1.0000         0         0         0         0         0
         0    1.0000         0    1.2618         0    0.8247         0    1.0000         0         0         0         0
         0         0    1.0000   -0.9237   -0.8247         0         0         0    1.0000         0         0         0
         0         0         0    0.9016   -0.7155         0         0         0         0    1.0000   -0.0000         0
         0         0         0    0.6216    0.7833         0         0         0         0         0    1.0000   -0.0000
         0         0         0   -0.4464    0.3543    1.0000         0         0         0         0    0.0000    1.0000


P_lcQ3 =

   99.2364   14.3436   22.4868    0.0742    0.0518   -0.4468
   14.3436  100.2796  -76.3087   -0.1567    0.1641    0.2611
   22.4868  -76.3087  100.2404   -0.1560   -0.0768   -0.1351
    0.0742   -0.1567   -0.1560    0.0038   -0.0000   -0.0030
    0.0518    0.1641   -0.0768   -0.0000    0.0026    0.0027
   -0.4468    0.2611   -0.1351   -0.0030    0.0027    0.0097
```

SCT-Q4. Plot the 3-sigma (x, y) confidence ellipses for global robot poses x_{wr1} and x_{wr2} in the world frame. (Hint: extract the 2×2 (x, y) submatrix for each $6x6$ covariance matrix and use the draw_ellipse() function to plot.) Note that each pose has approximately $10m$ of error in the world reference frame w. Now compute the relative pose $x_{r_1r_2}$, i.e., the robo pose at time t_2 w.r.t. the robot pose at time t_1 and plot its first-order $(x, y)3 - sigma$ confidence ellipse. Comment on what you see and explain why.

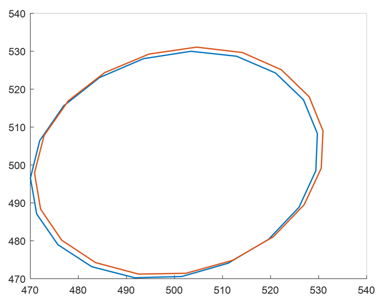
```
X_r1r2Q4 =
```

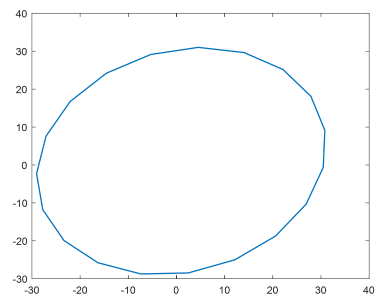
```
X_r1r2Q4 =

    1.0000
    1.0000
    0.2500
         0
         0
         0


JQ4 =

   -1.0000         0         0         0   -0.2500    1.0000    1.0000         0         0         0         0         0
         0   -1.0000         0    0.2500         0   -1.0000         0    1.0000         0         0         0         0
         0         0   -1.0000   -1.0000    1.0000         0         0         0    1.0000         0         0         0
         0         0         0   -1.0000         0         0         0         0         0    1.0000         0         0
         0         0         0         0   -1.0000         0         0         0         0         0    1.0000         0
         0         0         0         0         0   -1.0000         0         0         0         0         0    1.0000
```





The ellipse is the same size, but centered on the origin.

SCT-Q5. Sometimes, analytically computing the Jacobian is too error-prone and/or tedious. Instead, we can numerically compute the Jacobian with a good degree of accuracy. Excerpted below from the textbook "Multiple View Geometry" by Hartley and Zissermen is the outline of an algorithm for numerically computing a vector-valued function's Jacobian.

> Numerical differentiation may be carried out as follows. Each independent variable x_i is incremented in turn to $x_i + \delta$, the resulting function value is computed using the routine provided for computing f and the derivative is computed as a ratio. Good results have been found by setting the value δ to the maximum of $|10^{-4} \times x_i|$ and 10^{-6}. This choice seemingly gives a good approximation to the derivative. In practice, there seems to be little disadvantage in using numerical differentiation, though for simple functions f one may prefer to provide a routine to compute J, partly for aesthetic reasons, partly because of a possible slightly improved convergence and partly for speed.

Rewrite your software for SCT-Q1 to SCT-Q4 to use numerically computed Jacobians instead of the analytically evaluated ones returned by the `ssc_*` family of functions. Compare your results to those obtained using the analytical Jacobians.