CS455/655

Project 2. Consensus Filters for Sensor Networks (100 points)

Project Deadline: May 1st, 2018. Each student has to submit both **hard copy** and **electronic copy** of the project report.

1. **Hard copy submission in the class:** Return the hardcopy of your project report and **DO NOT** include source code in your hardcopy submission.

2. **Electronic copy submission: Include source code in your electronic submission.**
   Name/Zip your files as: "PR2-First_Lastname" then email your project report to Bravehung@yahoo.com: Before 11.30pm May 1st, 2018.

Case 1. Estimate single cell (single scalar value): 50 points

We randomly generate a connected network of 10 nodes in the area of 4x4. The cell is located at the center of this area. The ground truth of the measurement at this location is 50.

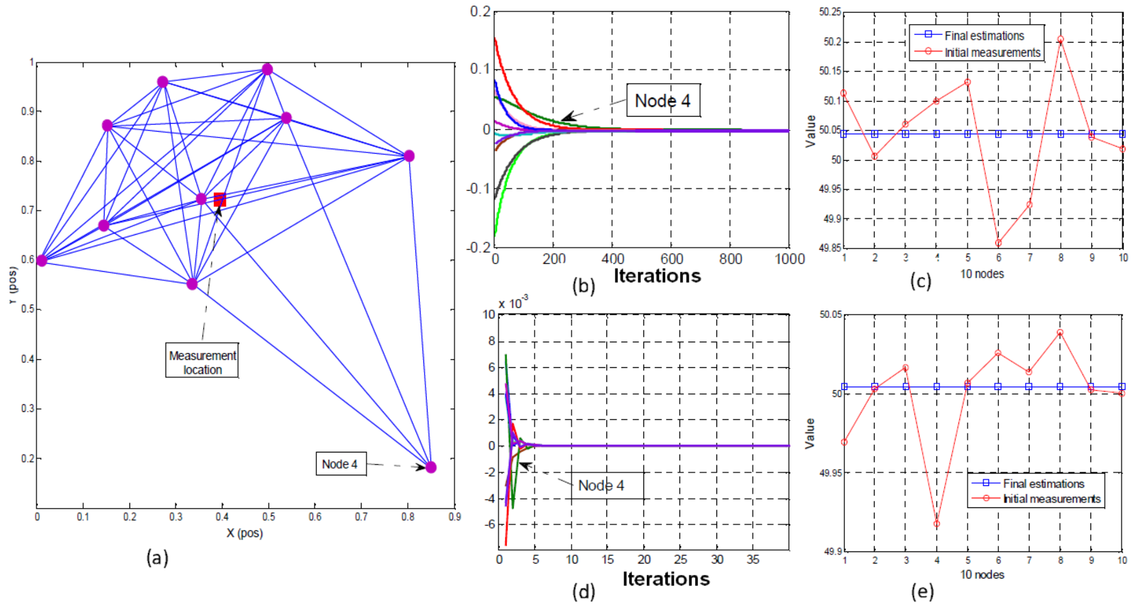
Each node makes a measurement as

$$m_i^1 = F^1 + n_i^1. \tag{1}$$

here $F^1 = 50$, and n_i^1 is the Gaussian noise, $N(0, V_i^1)$, with $V_i^1 = \frac{\|q_i - \overline{q}\|^2 + c_v}{(r_i^s)^2}$, $c_v = 0.01$, $r_1^s = r_2^s = ... = r_{10}^s = 1.6$, and $\overline{q} = \frac{1}{10}\sum_{i=1}^{10} q_i$. The initial condition for running the Consensus Filter 1 is $x_i^1(l = 0) = m_i^1$.

1. Show the results of the convergence of consensus filter 1 (Weighted Average Consensus) associated with two different weights, i.e., *weight design 1* and *weight design 2*. Explain the obtained results.(UG: 20points; G:15points)
2. Show the results of the convergence of consensus filter 2 (Average Consensus) with both Max-degree and Metropolis weights for a network of 10 nodes (area of 4x4), and a network of 50 nodes (area of 20x20). Explain the obtained results. (UG: 20points; G:15points)
3. Show the convergence of the node which has the smallest number of neighbors and the node which has the largest number of neighbors. Observe the obtained results and give explanation. (UG: 15points; G:10points)

$$(x_i^1(l) - E^1)$$



4. Show the convergence of the node which has the smallest number of neighbors and the node which has the largest number of neighbors in the dynamic network case where the node's neighbors are changing over time. Observe the obtained results and give explanation. (**Grad Students Only**): 10points

Some notes to be considered:

1. **Since we implement this consensus for a static sensor network (*t* is constant) the weight may not get updated in case both node i and its neighbors do not sense the cell. Therefore we should assume that at least one of node i's neighbors can sense cell k. (You can try to enlarge the sensing range.)**
2. **When node *i* can not sense the cell, you may try to set up the weight is empty instead of zero to avoid wrong update.**

Case 2. Estimate multiple cells (scalar field): 50 points

We model the scalar field of interest as

$$F = \Theta\Phi^T, \tag{2}$$

here $\Theta = [\theta_1, \theta_2, ..., \theta_K]$, and $\Phi = [\phi_1, \phi_2, ..., \phi_K]$, where j is the index, and K is the total number of function distributions. We can rewrite Equation (2) as

$$F = \sum_{j=1}^{K} \theta_j \phi_j, \tag{3}$$

here ϕ_j is a function representing the density distribution, and θ_j is the weight of the density distribution of the function ϕ_j.

We can model the function ϕ_j as a bivariate Gaussian distribution:

$$\phi_j = \frac{1}{\sqrt{det(C_j)(2\pi)^2}} e^{\frac{-1}{2}(Z-\mu_j^x)C_j^{-1}(Z-\mu_j^y)^T}, j \in [1, 2, ..., K].$$

here $Z = [x, y]$, $[\mu_j^x \; \mu_j^y]$ is the mean of the distribution of function ϕ_j, and C_j is the covariance matrix (positive definite) and it is represented by:

$$C_j = \begin{bmatrix} (\sigma_x^j)^2 & c_j^o \sigma_{xy}^j \\ c_j^0 \sigma_{xy}^j & (\sigma_y^j)^2 \end{bmatrix},$$

where c_j^0 is a correlation factor.

In this project, we model the scalar field F using four multiple variate Gaussian distributions ($K = 4$) with $\Theta = [20 \; 50 \; 35 \; 40]$, and each one is represented as:

$$\phi_1 = \frac{1}{\sqrt{det(C_1)(2\pi)^2}} e^{\frac{-1}{2}(Z-\mu_1)C_1^{-1}(Z-\mu_1)^T},$$

here $Z = [x, y]$, $\mu_1 = [2, 2]$, $C_1 = \begin{bmatrix} 2.25 & 0.2999 \\ 0.2999 & 2.25 \end{bmatrix}$, with the correlation factor $c_1^0 = 0.1333$.

$$\phi_2 = \frac{1}{\sqrt{det(C_2)(2\pi)^2}} e^{\frac{-1}{2}(Z-\mu_2)C_2^{-1}(Z-\mu_2)^T},$$

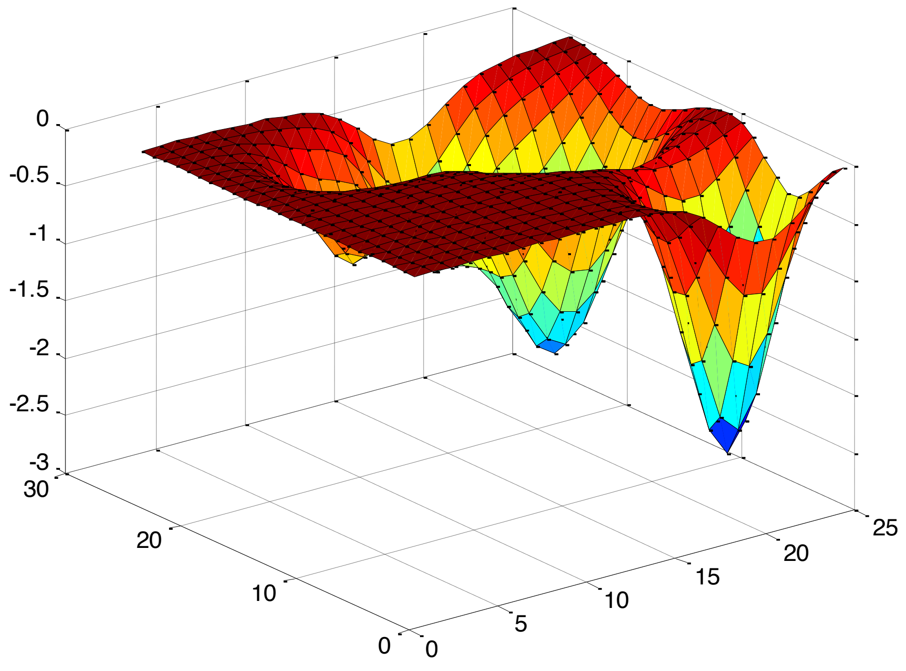
here $\mu_2 = [1, 0.5]$, $C_2 = \begin{bmatrix} 1.25 & 0.1666 \\ 0.1666 & 1.25 \end{bmatrix}$, and the correlation factor $c_2^0 = c_1^0$.

$$\phi_3 = \frac{1}{\sqrt{det(C_3)(2\pi)^2}} e^{\frac{-1}{2}(Z-\mu_3)C_3^{-1}(Z-\mu_3)^T},$$

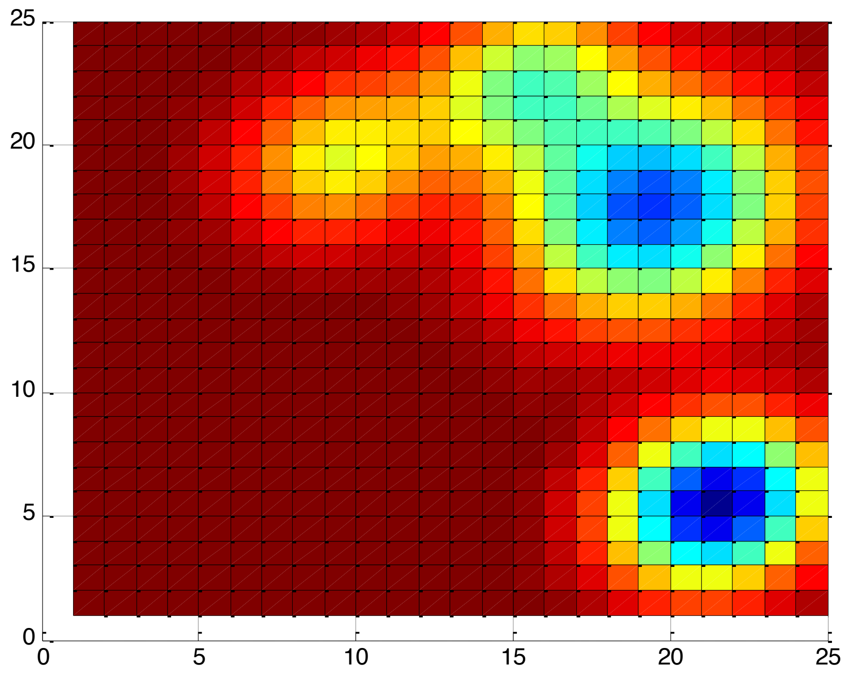
here $\mu_3 = [4.3, 3.5]$, $C_3 = C_2$, and the correlation factor $c_3^0 = c_2^0$.

$$\phi_4 = \frac{1}{\sqrt{det(C_4)(2\pi)^2}} e^{\frac{-1}{2}(Z-\mu_4)C_4^{-1}(Z-\mu_4)^T},$$

here $\mu_4 = [3, -3]$, $C_4 = C_3$, and with the correlation factor $c_4^0 = c_3^0$.



Scalar field F in 3D



Scalar field F in 2D

The field F has a size $x \times y = 25 \times 25$, and it is partitioned into $25 \times 25 = 625$ cells. You can set variables x and y run as: *0 to 25* with scale of *1* as presented in the above scalar field figures.

1. Generate a connected network of 30 nodes to **cover the entire area**. You can select the node's sensing range (may be $r^s_i = 5$). (UG: 10points; G:10points)
2. Running the Consensus 1 (Weighted Average Consensus) to obtain the estimate at each cell of the field F. Then, build the map of this scalar field. (UG: 25points; G:20points)
3. Plot the error between the build map and the original one in both 2D and 3D (ignore if difficult), respectively, and give explanation for the obtained results. (UG: 15points; G:10points)
4. Running the Consensus 2 (Average Consensus) to find out the confidence (weight) of the estimate at each cell, then plot the confidence (weight) in both 2D and 3D. **(Graduate Students Only): 10 points**
5. **Optional to any student: If you would like to apply flocking control in project 1 to do scalar field mapping, you can get 10% bonus point for this project.**

Note: If you feel difficult to model the scalar field F, you can download the .txt data file or Matlab data file of the field F. The data file will contain a valued matrix of *25x25*, and all you need to do is to assign each scalar value with its own *(x, y)* coordinate.