

Assignment #1

Instructor: Sundeep Chepuri

Name: Vineeth S, SR No.: 16543

PART A: Problem 1 Cramer-Rao Lower Bound (CRLB)

(Solution) From the range difference equations we have,

$$r_{ij} = d_i - d_j + n_{ij}$$

$$n_{ij} = w_i - w_j$$

We will use a set of non redundant range equations,

$$r_{12} = d_1 - d_2 + n_{12}$$

$$r_{23} = d_2 - d_3 + n_{23}$$

$$r_{34} = d_3 - d_4 + n_{34}$$

Concisely in vector notation we can write as,

$$\begin{bmatrix} r_{12} \\ r_{23} \\ r_{34} \end{bmatrix} = \begin{bmatrix} d_1 - d_2 \\ d_2 - d_3 \\ d_3 - d_4 \end{bmatrix} + \begin{bmatrix} n_{12} \\ n_{23} \\ n_{34} \end{bmatrix}$$

$$\underline{\mathbf{r}} = \underline{\mathbf{h}}(\underline{\mathbf{x}}) + \underline{\mathbf{n}}$$

$$\underline{\mathbf{r}} \sim \mathcal{N}(\underline{\mathbf{h}}(\underline{\mathbf{x}}), \underline{\Sigma}_{\mathbf{n}})$$

where $\underline{\Sigma}_{\mathbf{n}}$ is the noise covariance matrix, given by,

$$\underline{\mathbf{n}} = \begin{bmatrix} n_{12} \\ n_{23} \\ n_{34} \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$$

$$\underline{\mathbf{n}} = \underline{\mathbf{A}}\underline{\mathbf{w}}$$

$$\underline{\Sigma}_{\mathbf{n}} = E[\underline{\mathbf{n}}\underline{\mathbf{n}}^T]$$

$$= E[\underline{\mathbf{A}}\underline{\mathbf{w}}\underline{\mathbf{w}}^T \underline{\mathbf{A}}]$$

$$= \underline{\mathbf{A}}E[\underline{\mathbf{w}}\underline{\mathbf{w}}^T] \underline{\mathbf{A}}^T$$

$$= \underline{\mathbf{A}}\underline{\Sigma}_{\mathbf{w}} \underline{\mathbf{A}}^T$$

With this we can write the likelihood and log-likelihood functions as,

$$P(\underline{\mathbf{r}}; \underline{\mathbf{x}}) = \frac{1}{(2\pi)^{3/2} |\underline{\Sigma}_{\mathbf{n}}|^{1/2}} \exp\left(-\frac{1}{2}(\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}}))^T \underline{\Sigma}_{\mathbf{n}}^{-1} (\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}}))\right)$$

$$\ln P(\underline{\mathbf{r}}; \underline{\mathbf{x}}) = K - \frac{1}{2}(\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}}))^T \underline{\Sigma}_{\mathbf{n}}^{-1} (\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}}))$$

$$\begin{aligned}
&= K - \frac{1}{2} \sum_{i=1}^3 \sum_{j=1}^3 (r_i - h_i(\underline{\mathbf{x}})) [\boldsymbol{\Sigma}_{\mathbf{n}}^{-1}]_{ij} (r_j - h_j(\underline{\mathbf{x}})) \\
\frac{\partial}{\partial \underline{\mathbf{x}}} \ln P(\underline{\mathbf{r}}; \underline{\mathbf{x}}) &= \frac{1}{2} \sum_{i=1}^3 \sum_{j=1}^3 [\boldsymbol{\Sigma}_{\mathbf{n}}^{-1}]_{ij} (r_i - h_i(\underline{\mathbf{x}})) \frac{\partial h_j(\underline{\mathbf{x}})}{\partial \underline{\mathbf{x}}} + \frac{\partial h_i(\underline{\mathbf{x}})}{\partial \underline{\mathbf{x}}} (r_j - h_j(\underline{\mathbf{x}}))
\end{aligned}$$

We can find the elements of the Fischer Information Matrix one element at a time,

$$\frac{\partial^2}{\partial x^2} \ln P(\underline{\mathbf{r}}; \underline{\mathbf{x}}) = \frac{1}{2} \sum_{i=1}^3 \sum_{j=1}^3 [\boldsymbol{\Sigma}_{\mathbf{n}}^{-1}]_{ij} ((r_i - h_i(\underline{\mathbf{x}})) \frac{\partial^2 h_j(\underline{\mathbf{x}})}{\partial x^2} - \frac{\partial h_i(\underline{\mathbf{x}})}{\partial x} \frac{\partial h_j(\underline{\mathbf{x}})}{\partial x} + (r_j - h_j(\underline{\mathbf{x}})) \frac{\partial^2 h_i(\underline{\mathbf{x}})}{\partial x^2} - \frac{\partial h_j(\underline{\mathbf{x}})}{\partial x} \frac{\partial h_i(\underline{\mathbf{x}})}{\partial x})$$

$$\begin{aligned}
[\mathbf{I}(\underline{\mathbf{x}})]_{11} &= -E\left[\frac{\partial^2}{\partial x^2} \ln P(\underline{\mathbf{r}}; \underline{\mathbf{x}})\right] \\
&= \sum_{i=1}^3 \sum_{j=1}^3 [\boldsymbol{\Sigma}_{\mathbf{n}}^{-1}]_{ij} \frac{\partial h_i(\underline{\mathbf{x}})}{\partial x} \frac{\partial h_j(\underline{\mathbf{x}})}{\partial x} \\
&= \frac{\partial \underline{\mathbf{h}}(\underline{\mathbf{x}})}{\partial x}^T \boldsymbol{\Sigma}_{\mathbf{n}}^{-1} \frac{\partial \underline{\mathbf{h}}(\underline{\mathbf{x}})}{\partial x}
\end{aligned}$$

$$\begin{aligned}
[\mathbf{I}(\underline{\mathbf{x}})]_{12} &= -E\left[\frac{\partial^2}{\partial x \partial y} \ln P(\underline{\mathbf{r}}; \underline{\mathbf{x}})\right] \\
&= \sum_{i=1}^3 \sum_{j=1}^3 [\boldsymbol{\Sigma}_{\mathbf{n}}^{-1}]_{ij} \frac{\partial h_i(\underline{\mathbf{x}})}{\partial x} \frac{\partial h_j(\underline{\mathbf{x}})}{\partial y} \\
&= \frac{\partial \underline{\mathbf{h}}(\underline{\mathbf{x}})}{\partial x}^T \boldsymbol{\Sigma}_{\mathbf{n}}^{-1} \frac{\partial \underline{\mathbf{h}}(\underline{\mathbf{x}})}{\partial y}
\end{aligned}$$

$$\begin{aligned}
[\mathbf{I}(\underline{\mathbf{x}})]_{21} &= -E\left[\frac{\partial^2}{\partial y \partial x} \ln P(\underline{\mathbf{r}}; \underline{\mathbf{x}})\right] \\
&= \sum_{i=1}^3 \sum_{j=1}^3 [\boldsymbol{\Sigma}_{\mathbf{n}}^{-1}]_{ij} \frac{\partial h_i(\underline{\mathbf{x}})}{\partial y} \frac{\partial h_j(\underline{\mathbf{x}})}{\partial x} \\
&= \frac{\partial \underline{\mathbf{h}}(\underline{\mathbf{x}})}{\partial y}^T \boldsymbol{\Sigma}_{\mathbf{n}}^{-1} \frac{\partial \underline{\mathbf{h}}(\underline{\mathbf{x}})}{\partial x}
\end{aligned}$$

$$\begin{aligned}
[\mathbf{I}(\underline{\mathbf{x}})]_{22} &= -E\left[\frac{\partial^2}{\partial y \partial y} \ln P(\underline{\mathbf{r}}; \underline{\mathbf{x}})\right] \\
&= \sum_{i=1}^3 \sum_{j=1}^3 [\boldsymbol{\Sigma}_{\mathbf{n}}^{-1}]_{ij} \frac{\partial h_i(\underline{\mathbf{x}})}{\partial y} \frac{\partial h_j(\underline{\mathbf{x}})}{\partial y} \\
&= \frac{\partial \underline{\mathbf{h}}(\underline{\mathbf{x}})}{\partial y}^T \boldsymbol{\Sigma}_{\mathbf{n}}^{-1} \frac{\partial \underline{\mathbf{h}}(\underline{\mathbf{x}})}{\partial y}
\end{aligned}$$

where,

$$\frac{\partial \underline{\mathbf{h}}(\underline{\mathbf{x}})}{\partial y} = \begin{bmatrix} \frac{\partial h_1(\underline{\mathbf{x}})}{\partial y} & \frac{\partial h_2(\underline{\mathbf{x}})}{\partial y} & \frac{\partial h_3(\underline{\mathbf{x}})}{\partial y} \end{bmatrix}$$

$$\begin{aligned}
\frac{\partial h_i(\underline{\mathbf{x}})}{\partial x} &= \frac{\partial}{\partial x} (d_i - d_{i+1}) \\
&= \frac{\partial}{\partial x} (\|\mathbf{x} - \mathbf{x}_i\|_2 - \|\mathbf{x} - \mathbf{x}_{i+1}\|_2) \\
&= \frac{x - x_i}{\|\mathbf{x} - \mathbf{x}_i\|_2} - \frac{x - x_{i+1}}{\|\mathbf{x} - \mathbf{x}_{i+1}\|_2} \\
&= \frac{x - x_i}{d_i} - \frac{x - x_{i+1}}{d_{i+1}}
\end{aligned}$$

Given an $\underline{\mathbf{x}}$, we can find the elements of Fischer Information Matrix and invert it to get the CRLB.

$$\mathbf{C}_{\hat{\mathbf{x}}} \geq [\mathbf{I}^{-1}(\underline{\mathbf{x}})]$$

$$Var[\hat{x}] \geq [\mathbf{I}^{-1}(\underline{\mathbf{x}})]_{11}$$

$$Var[\hat{y}] \geq [\mathbf{I}^{-1}(\underline{\mathbf{x}})]_{22}$$

PART A: Problem 2 Maximum Likelihood Estimator (MLE)

(Solution) We want obtain the maximum likelihood estimator (MLE) for $\underline{\mathbf{x}}$. The estimator that aximizes likelihood function also maximizes log-likelihood, since log function is a monotonically increasing function.

We have our likelihood and log-likelihood functions from Problem 1 as,

$$\begin{aligned}
 P(\underline{\mathbf{r}}; \underline{\mathbf{x}}) &= \frac{1}{(2\pi)^{3/2} |\Sigma_n|^{1/2}} \exp\left(-\frac{1}{2}(\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}}))^T \Sigma_n^{-1} (\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}}))\right) \\
 \ln P(\underline{\mathbf{r}}; \underline{\mathbf{x}}) &= K - \frac{1}{2}(\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}}))^T \Sigma_n^{-1} (\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}})) \\
 \hat{\underline{\mathbf{x}}} &= \arg \max_{\underline{\mathbf{x}}} \ln P(\underline{\mathbf{r}}; \underline{\mathbf{x}}) \\
 &= \arg \max_{\underline{\mathbf{x}}} K - \frac{1}{2}(\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}}))^T \Sigma_n^{-1} (\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}})) \\
 &= \arg \min_{\underline{\mathbf{x}}} \frac{1}{2}(\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}}))^T \Sigma_n^{-1} (\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}})) \\
 &= \arg \min_{\underline{\mathbf{x}}} J(\underline{\mathbf{x}})
 \end{aligned}$$

We can proceed with this minimization as an optimization problem, using any of the available descent algorithms. We will use a simple iterative gradient descent algorithm with a fixed step size α such that $0 < \alpha < \frac{2}{\lambda_{max}}$, where λ_{max} is the largest eigenvalue of Σ_n^{-1} (for which the gradient descent theoretically converges if objective function is Lipschitz).

$$\begin{aligned}
 \hat{\underline{\mathbf{x}}}^{(k+1)} &= \hat{\underline{\mathbf{x}}}^{(k)} - \alpha \nabla J(\underline{\mathbf{x}}) \\
 &= \hat{\underline{\mathbf{x}}}^{(k)} + \alpha \nabla \underline{\mathbf{h}}(\underline{\mathbf{x}})^T \Sigma_n^{-1} (\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}}))
 \end{aligned}$$

where,

$$\nabla \underline{\mathbf{h}}(\underline{\mathbf{x}}) = \frac{\partial \underline{\mathbf{h}}(\underline{\mathbf{x}})}{\partial \underline{\mathbf{y}}} = \begin{bmatrix} \frac{\partial h_1(\underline{\mathbf{x}})}{\partial x} & \frac{\partial h_2(\underline{\mathbf{x}})}{\partial x} & \frac{\partial h_3(\underline{\mathbf{x}})}{\partial x} \end{bmatrix}$$

$$\begin{aligned}
 \frac{\partial h_i(\underline{\mathbf{x}})}{\partial x} &= \frac{\partial}{\partial x} (d_i - d_{i+1}) \\
 &= \frac{\partial}{\partial x} (\|\mathbf{x} - \mathbf{x}_i\|_2 - \|\mathbf{x} - \mathbf{x}_{i+1}\|_2) \\
 &= \frac{x - x_i}{\|\mathbf{x} - \mathbf{x}_i\|_2} - \frac{x - x_{i+1}}{\|\mathbf{x} - \mathbf{x}_{i+1}\|_2} \\
 &= \frac{x - x_i}{d_i} - \frac{x - x_{i+1}}{d_{i+1}}
 \end{aligned}$$

PART A: Problem 3 Best Linear Unbiased Estimator (BLUE) estimator for θ

(Solution) From the range difference equations we have,

$$\begin{aligned} r_{ij} &= d_i - d_j + n_{ij} \\ r_{ij} + d_j &= d_i + n_{ij} \\ r_{ij}^2 + 2r_{ij}d_j + d_j^2 &= d_i^2 + e_{ij} \end{aligned}$$

where d_i and e_{ij} are given by,

$$\begin{aligned} d_i^2 &= \|\mathbf{x} - \mathbf{x}_i\|_2^2 = \|\mathbf{x}\|^2 + \|\mathbf{x}_i\|^2 - 2\mathbf{x}_i^T \mathbf{x} \\ e_{ij} &= n_{ij}^2 = (w_i - w_j)^2 \end{aligned}$$

Substituting in range equation,

$$\begin{aligned} r_{ij}^2 &= d_i^2 - d_j^2 - 2r_{ij}d_j + e_{ij} \\ r_{ij}^2 &= (\|\mathbf{x}\|^2 + \|\mathbf{x}_i\|^2 - 2\mathbf{x}_i^T \mathbf{x}) - (\|\mathbf{x}\|^2 + \|\mathbf{x}_j\|^2 - 2\mathbf{x}_j^T \mathbf{x}) - 2r_{ij}d_j + e_{ij} \\ r_{ij}^2 &= \|\mathbf{x}_i\|^2 - \|\mathbf{x}_j\|^2 - 2(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{x} - 2r_{ij}d_j + e_{ij} \end{aligned}$$

Let $\gamma_{ij} = \|\mathbf{x}_i\|^2 - \|\mathbf{x}_j\|^2$,

$$r_{ij}^2 = \gamma_{ij} - 2(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{x} - 2r_{ij}d_j + e_{ij}$$

$$\begin{aligned} \begin{bmatrix} r_{12}^2 \\ r_{13}^2 \\ r_{14}^2 \\ r_{23}^2 \\ r_{24}^2 \\ r_{34}^2 \end{bmatrix} &= \begin{bmatrix} -2(\mathbf{x}_1 - \mathbf{x}_2)^T & -2r_{12} & 0 & 0 \\ -2(\mathbf{x}_1 - \mathbf{x}_3)^T & 0 & -2r_{13} & 0 \\ -2(\mathbf{x}_1 - \mathbf{x}_4)^T & 0 & 0 & -2r_{14} \\ -2(\mathbf{x}_2 - \mathbf{x}_3)^T & 0 & -2r_{23} & 0 \\ -2(\mathbf{x}_2 - \mathbf{x}_4)^T & 0 & 0 & -2r_{24} \\ -2(\mathbf{x}_3 - \mathbf{x}_4)^T & 0 & 0 & -2r_{34} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} + \begin{bmatrix} e_{12} \\ e_{13} \\ e_{14} \\ e_{23} \\ e_{24} \\ e_{34} \end{bmatrix} + \begin{bmatrix} \gamma_{12} \\ \gamma_{13} \\ \gamma_{14} \\ \gamma_{23} \\ \gamma_{24} \\ \gamma_{34} \end{bmatrix} \\ \begin{bmatrix} r_{12}^2 \\ r_{13}^2 \\ r_{14}^2 \\ r_{23}^2 \\ r_{24}^2 \\ r_{34}^2 \end{bmatrix} &= \begin{bmatrix} -2(x_1 - x_2)^T & -2(y_1 - y_2)^T & -2r_{12} & 0 & 0 \\ -2(x_1 - x_3)^T & -2(y_1 - y_3)^T & 0 & -2r_{13} & 0 \\ -2(x_1 - x_4)^T & -2(y_1 - y_4)^T & 0 & 0 & -2r_{14} \\ -2(x_2 - x_3)^T & -2(y_2 - y_3)^T & 0 & -2r_{23} & 0 \\ -2(x_2 - x_4)^T & -2(y_2 - y_4)^T & 0 & 0 & -2r_{24} \\ -2(x_3 - x_4)^T & -2(y_3 - y_4)^T & 0 & 0 & -2r_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} + \begin{bmatrix} e_{12} \\ e_{13} \\ e_{14} \\ e_{23} \\ e_{24} \\ e_{34} \end{bmatrix} + \begin{bmatrix} \gamma_{12} \\ \gamma_{13} \\ \gamma_{14} \\ \gamma_{23} \\ \gamma_{24} \\ \gamma_{34} \end{bmatrix} \end{aligned}$$

Concisely in vector notation,

$$\mathbf{r} = \mathbf{H}\theta + \mathbf{e} + \boldsymbol{\gamma}$$

where,

$$\underline{\mathbf{r}} = \begin{bmatrix} r_{12}^2 \\ r_{13}^2 \\ r_{14}^2 \\ r_{23}^2 \\ r_{24}^2 \\ r_{34}^2 \end{bmatrix}, \underline{\mathbf{H}} = \begin{bmatrix} -2(x_1 - x_2)^T & -2(y_1 - y_2)^T & -2r_{12} & 0 & 0 \\ -2(x_1 - x_3)^T & -2(y_1 - y_3)^T & 0 & -2r_{13} & 0 \\ -2(x_1 - x_4)^T & -2(y_1 - y_4)^T & 0 & 0 & -2r_{14} \\ -2(x_2 - x_3)^T & -2(y_2 - y_3)^T & 0 & -2r_{23} & 0 \\ -2(x_2 - x_4)^T & -2(y_2 - y_4)^T & 0 & 0 & -2r_{24} \\ -2(x_3 - x_4)^T & -2(y_3 - y_4)^T & 0 & 0 & -2r_{34} \end{bmatrix}, \underline{\theta} = \begin{bmatrix} x \\ y \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}, \underline{\mathbf{e}} = \begin{bmatrix} e_{12} \\ e_{13} \\ e_{14} \\ e_{23} \\ e_{24} \\ e_{34} \end{bmatrix}, \underline{\gamma} = \begin{bmatrix} \gamma_{12} \\ \gamma_{13} \\ \gamma_{14} \\ \gamma_{23} \\ \gamma_{24} \\ \gamma_{34} \end{bmatrix}$$

But here the mean of error $\underline{\mathbf{e}}$ is no longer $\underline{\mathbf{0}}$ and hence we should subtract the mean of the error from above observations to make error zero mean.

$$E[\underline{\mathbf{e}}] = \begin{bmatrix} E[e_{12}] \\ E[e_{13}] \\ E[e_{14}] \\ E[e_{23}] \\ E[e_{24}] \\ E[e_{34}] \end{bmatrix} = 2\sigma^2 \underline{\mathbf{1}}_{6 \times 1}$$

$$\begin{aligned} E[e_{ij}] &= E[(w_i - w_j)^2] \\ &= \text{Var}[w_i - w_j] + E[w_i - w_j] \\ &= \text{Var}[w_i] + \text{Var}[w_j] \\ &= 2\sigma^2 \end{aligned}$$

We can now have a model with zero mean as,

$$\begin{aligned} \underline{\mathbf{r}} - \underline{\gamma} - 2\sigma^2 \underline{\mathbf{1}}_{6 \times 1} &= \underline{\mathbf{H}}\underline{\theta} + \underline{\mathbf{e}} - 2\sigma^2 \underline{\mathbf{1}}_{6 \times 1} \\ \underline{\mathbf{r}}' &= \underline{\mathbf{H}}\underline{\theta} + \underline{\mathbf{e}}' \\ \underline{\mathbf{r}}' &= \underline{\mathbf{r}} - \underline{\gamma} - 2\sigma^2 \underline{\mathbf{1}}_{6 \times 1} \\ \underline{\mathbf{e}}' &= \underline{\mathbf{e}} - 2\sigma^2 \underline{\mathbf{1}}_{6 \times 1} \\ \underline{\mathbf{e}}' &\sim \mathcal{N}(0, \underline{\mathbf{C}}) \end{aligned}$$

where $\underline{\mathbf{C}}$ is the covariance matrix of $\underline{\mathbf{e}}'$ which is given by,

$$\begin{aligned} \underline{\mathbf{C}} &= E[(\underline{\mathbf{e}}' - E[\underline{\mathbf{e}}']) (\underline{\mathbf{e}}' - E[\underline{\mathbf{e}}'])^T] \\ &= E[\underline{\mathbf{e}}' \underline{\mathbf{e}}'^T] \\ &= E[(\underline{\mathbf{e}} - 2\sigma^2 \underline{\mathbf{1}}_{6 \times 1})(\underline{\mathbf{e}} - 2\sigma^2 \underline{\mathbf{1}}_{6 \times 1})^T] \\ &= E[\underline{\mathbf{e}}\underline{\mathbf{e}}^T - \underline{\mathbf{e}}(2\sigma^2 \underline{\mathbf{1}}_{6 \times 1})^T - (2\sigma^2 \underline{\mathbf{1}}_{6 \times 1})\underline{\mathbf{e}}^T + (2\sigma^2 \underline{\mathbf{1}}_{6 \times 1})(2\sigma^2 \underline{\mathbf{1}}_{6 \times 1})^T] \\ &= E[\underline{\mathbf{e}}\underline{\mathbf{e}}^T] - (2\sigma^2 \underline{\mathbf{1}}_{6 \times 1})(2\sigma^2 \underline{\mathbf{1}}_{6 \times 1})^T \end{aligned}$$

We can find element by element of $\underline{\mathbf{C}}$ as,

$$\begin{aligned} [\underline{\mathbf{C}}]_{11} &= E[e_{12}^2] - 4\sigma^4 \\ &= E[(w_1 - w_2)^2(w_1 - w_2)^2] - 4\sigma^4 \\ &= E[(w_1^2 - 2w_1w_2 + w_2^2)(w_1^2 - 2w_1w_2 + w_2^2)] - 4\sigma^4 \end{aligned}$$

$$\begin{aligned}
&= E[w_1^4 + w_2^4 + 6w_1^2w_2^2] - 4\sigma^4 \\
&= 3\sigma^4 + 3\sigma^4 + 6\sigma^4 - 4\sigma^4 \\
&= 8\sigma^4
\end{aligned}$$

$$\begin{aligned}
[\mathbf{C}]_{12} &= E[e_{12}e_{13}] - 4\sigma^4 \\
&= E[(w_1 - w_2)^2(w_1 - w_3)^2] - 4\sigma^4 \\
&= E[(w_1^2 - 2w_1w_2 + w_2^2)(w_1^2 - 2w_1w_3 + w_3^2)] - 4\sigma^4 \\
&= E[w_1^4 + w_1^2w_3^2 + w_1^2w_2^2 + w_2^2w_3^2] - 4\sigma^4 \\
&= 3\sigma^4 + \sigma^4 + \sigma^4 + \sigma^4 - 4\sigma^4 \\
&= 2\sigma^4
\end{aligned}$$

$$\begin{aligned}
[\mathbf{C}]_{13} &= E[e_{12}e_{14}] - 4\sigma^4 \\
&= 2\sigma^4
\end{aligned}$$

$$\begin{aligned}
[\mathbf{C}]_{14} &= E[e_{12}e_{23}] - 4\sigma^4 \\
&= 2\sigma^4
\end{aligned}$$

$$\begin{aligned}
[\mathbf{C}]_{15} &= E[e_{12}e_{24}] - 4\sigma^4 \\
&= 2\sigma^4
\end{aligned}$$

$$\begin{aligned}
[\mathbf{C}]_{16} &= E[e_{12}e_{34}] - 4\sigma^4 \\
&= E[(w_1 - w_2)^2(w_3 - w_4)^2] - 4\sigma^4 \\
&= E[(w_1^2 - 2w_1w_2 + w_2^2)(w_3^2 - 2w_3w_4 + w_4^2)] - 4\sigma^4 \\
&= E[w_1^2w_3^2 + w_1^2w_4^2 + w_2^2w_3^2 + w_2^2w_4^2] - 4\sigma^4 \\
&= \sigma^4 + \sigma^4 + \sigma^4 + \sigma^4 - 4\sigma^4 \\
&= 0
\end{aligned}$$

Similarly we compute the matrix \mathbf{C} by identifying the pattern,

$$\mathbf{C} = 2\sigma^2 \begin{bmatrix} 4 & 1 & 1 & 1 & 1 & 0 \\ 1 & 4 & 1 & 1 & 0 & 1 \\ 1 & 1 & 4 & 0 & 1 & 1 \\ 1 & 1 & 0 & 4 & 1 & 1 \\ 1 & 0 & 1 & 1 & 4 & 1 \\ 0 & 1 & 1 & 1 & 1 & 4 \end{bmatrix}$$

Hence the best linear unbiased estimator (BLUE estimator),

$$\hat{\underline{\theta}} = (\mathbf{H}^T \mathbf{C}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{C}^{-1} \underline{\mathbf{r}}'$$

PART B: Problem 1 Estimated location of the Transvahan using MLE estimator

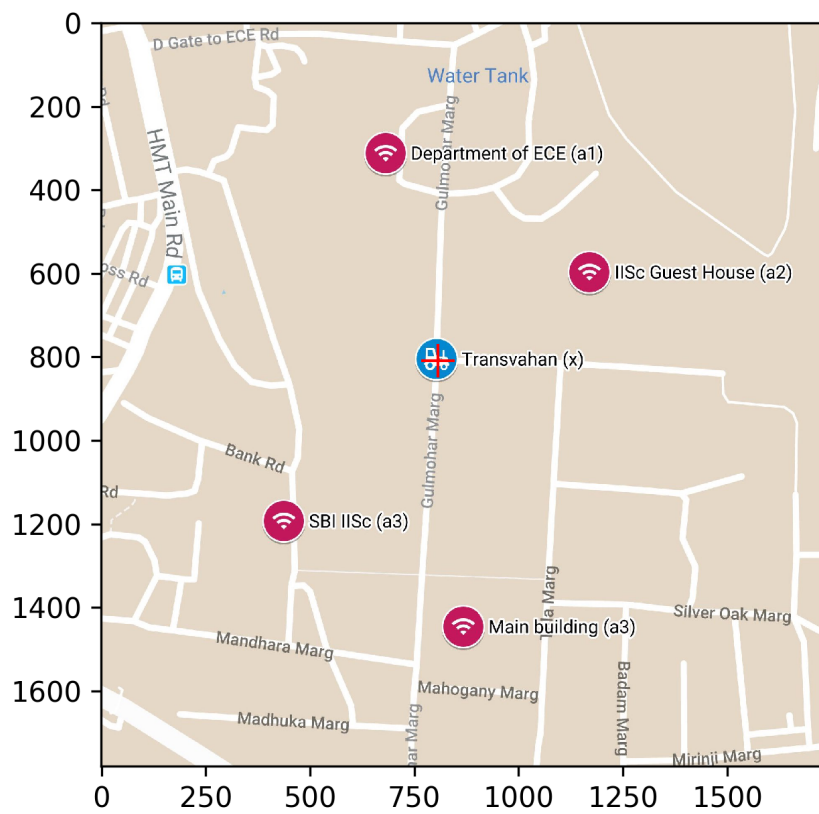


Figure 1: Estimated location of Transvahan using MLE estimator

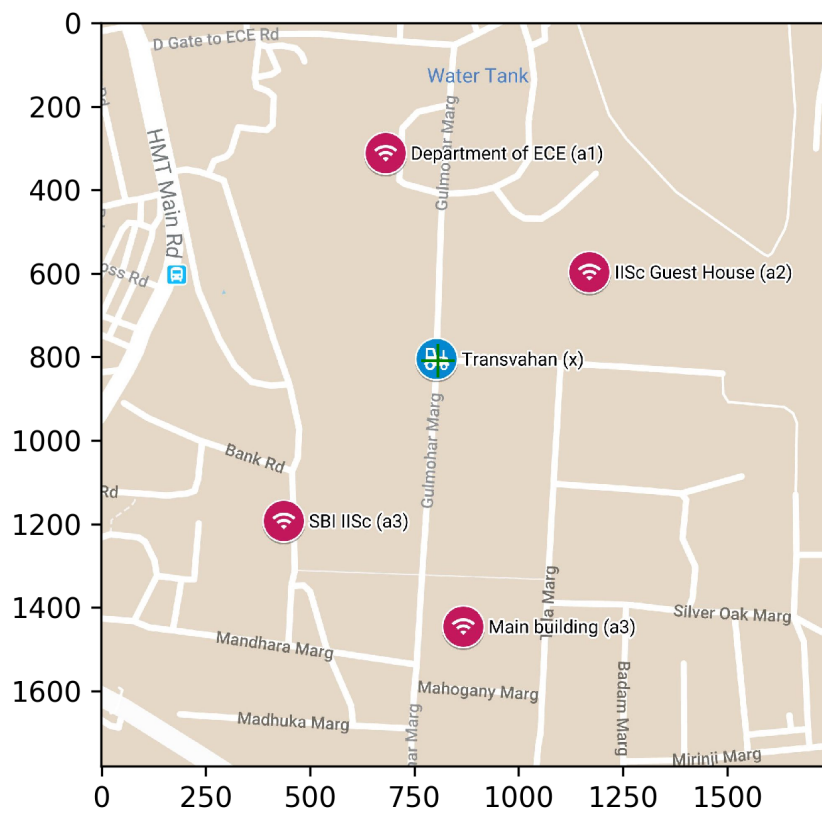
PART B: Problem 2 Estimated location of the Transvahan using BLUE estimator

Figure 2: Estimated location of Transvahan using BLUE estimator

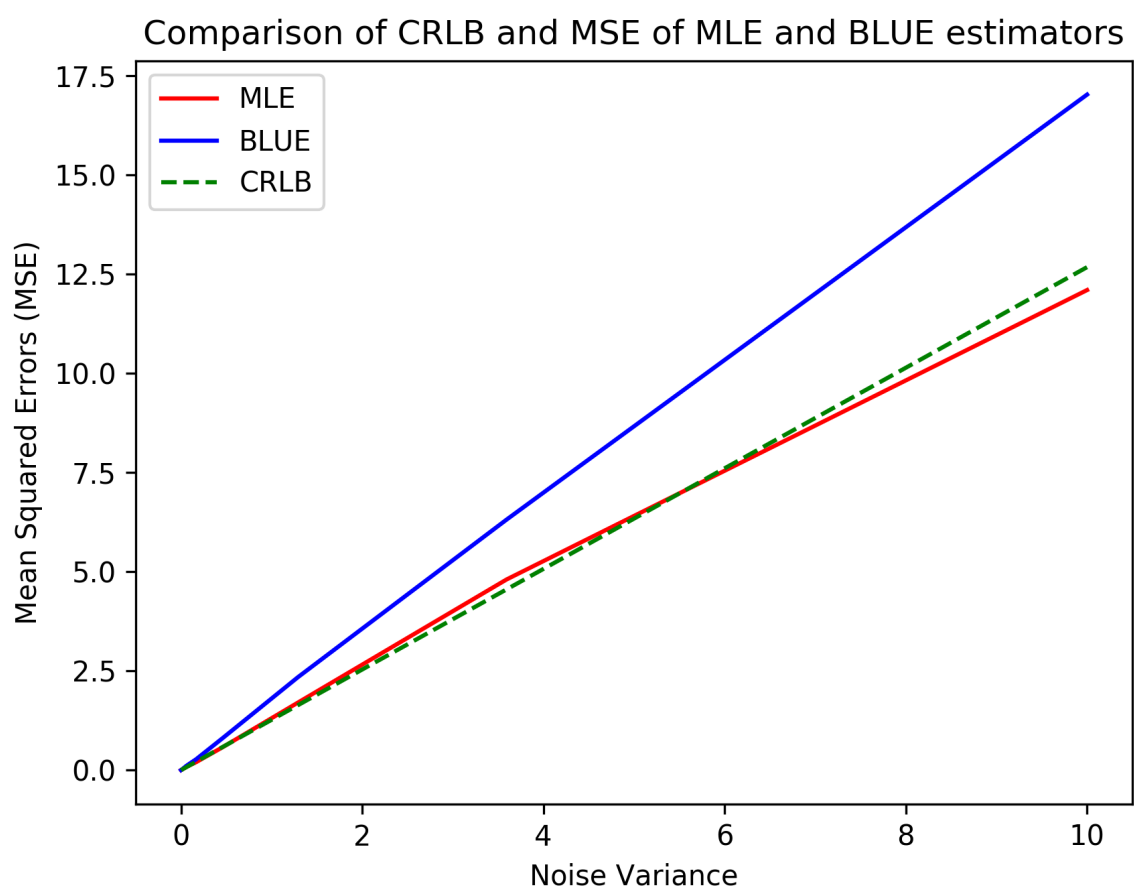
PART B: Problem 3 Comparison of CRLB and mean squared error of MLE & BLUE estimators

Figure 3: Comparison of CRLB and mean squared error of MLE & BLUE estimators

PART C: Appendices (Python codes)

The entire program is written in four python files - main.py, CRLB.py, MLE.py and BLUE.py. The codes are provided below and are also available at <https://github.com/vineeths96/TDOA-Localization> GitHub repository. (Repository access is private as of now. Access can be made available, if necessary).

main.py

```

1  import os
2  import numpy as np
3  import scipy.io
4  import matplotlib.pyplot as plt
5
6  from MLE import MLE_TDOA
7  from BLUE import BLUE_TDOA
8  from CRLB import CRLB
9
10
11 def MSE_MLE(noisy_distances, anchor_location, x_0, target_location):
12     (xdim, ydim, zdim) = noisy_distances.shape
13     mse = np.zeros(zdim)
14
15     for i in range(zdim):
16         for j in range(ydim):
17             x_estimate = MLE_TDOA(noisy_distances[:, j, i], anchor_location, x_0)
18             u = x_estimate - target_location
19             mse_y = np.linalg.norm(u) ** 2
20             mse[i] = mse[i] + mse_y
21
22     mse = mse / ydim
23
24     return mse
25
26
27 def MSE_BLUE(noisy_distances, anchor_location, sigma2, target_location):
28     (xdim, ydim, zdim) = noisy_distances.shape
29     mse = np.zeros(zdim)
30
31     for i in range(zdim):
32         for j in range(ydim):
33             x_estimate = BLUE_TDOA(noisy_distances[:, j, i], anchor_location, sigma2[i])
34             u = x_estimate - target_location
35             mse_y = np.linalg.norm(u) ** 2
36             mse[i] = mse[i] + mse_y
37
38     mse = mse / ydim
39
40     return mse
41
42
43 def main():
44     try:
45         os.makedirs('./results')
46     except:
47         pass
48

```

```

49     TDOA_data = scipy.io.loadmat('./data/TDOA_data.mat')
50     anchor_location = TDOA_data['anchor_location']
51     anchor_location = anchor_location.astype(int)
52     noisy_distances = TDOA_data['noisy_distances']
53     target_location = TDOA_data['target_location']
54     target_location = target_location.astype(float)
55     target_location = [target_location[0][0], target_location[1][0]]
56     sigma2 = TDOA_data['sigma2']
57
58     # Problem 1
59     x_0 = [0, 0]
60     x = MLE_TDOA(noisy_distances[:, 1, 1], anchor_location, x_0)
61     map = plt.imread('./data/mapimage.jpeg')
62     filename = './results/MLE_map_loc.png'
63
64     plt.figure()
65     plt.imshow(map)
66     plt.plot(x[0], x[1], '+', color='red', markersize=12)
67     plt.savefig(filename, dpi=300)
68
69     # Problem 2
70     x = BLUE_TDOA(noisy_distances[:, 1, 1], anchor_location, sigma2[1])
71     map = plt.imread('./data/mapimage.jpeg')
72     filename = './results/BLUE_map_loc.png'
73
74     plt.figure()
75     plt.imshow(map)
76     plt.plot(x[0], x[1], '+', color='green', markersize=12)
77     plt.savefig(filename, dpi=300)
78
79     # Problem 3
80     MSE_MLE_value = MSE_MLE(noisy_distances, anchor_location, x_0, target_location)
81     MSE_BLUE_value = MSE_BLUE(noisy_distances, anchor_location, sigma2, target_location)
82     CRLB_value = CRLB(target_location, anchor_location, sigma2)
83     filename = './results/MSE.png'
84
85     plt.figure()
86     plt.plot(sigma2, MSE_MLE_value, color='red', label='MLE')
87     plt.plot(sigma2, MSE_BLUE_value, color='blue', label='BLUE')
88     plt.plot(sigma2, CRLB_value, 'g--', label = 'CRLB')
89     plt.title('Comparison of CRLB and MSE of MLE and BLUE estimators')
90     plt.xlabel('Noise Variance')
91     plt.ylabel('Mean Squared Errors (MSE)')
92     plt.legend()
93     plt.savefig(filename, dpi=300)
94
95
96 if __name__ == '__main__':
97     main()

```

CRLB.py

```

1  import numpy as np
2
3
4  def CRLB(x, anchor_location, sigma2):
5      CRLB_value = np.zeros(len(sigma2))
6      DIMENSIONS = 2
7      num_anchors = anchor_location.shape[1]
8
9      for ind in range(len(sigma2)):
10         d = np.zeros(num_anchors)
11         h = np.zeros(num_anchors-1)
12         I = np.zeros([DIMENSIONS, DIMENSIONS])
13         delH = np.zeros([DIMENSIONS, (num_anchors-1)])
14
15         C = sigma2[ind] * np.array([[2, -1, 0], [-1, 2, -1], [0, -1, 2]])
16         C_inverse = np.linalg.inv(C)
17
18         for j in range(num_anchors):
19             d[j] = np.linalg.norm(x - anchor_location[:, j])
20
21         for j in range(num_anchors-1):
22             h[j] = d[j] - d[j + 1]
23
24         for j in range(num_anchors-1):
25             for k in range(DIMENSIONS):
26                 delH[k, j] = ((x[k] - anchor_location[k, j]) / d[j]) - ((x[k] - anchor_location[k, j + 1]) / d[j + 1])
27
28         for i in range(num_anchors-1):
29             for j in range(num_anchors-1):
30                 I[0, 0] = I[0, 0] + C_inverse[i, j] * (delH[0, i] * delH[0, j] + delH[0, j] * delH[0, i])
31                 I[0, 1] = I[0, 1] + C_inverse[i, j] * (delH[1, i] * delH[0, j] + delH[1, j] * delH[0, i])
32                 I[1, 1] = I[1, 1] + C_inverse[i, j] * (delH[1, i] * delH[1, j] + delH[1, j] * delH[1, i])
33                 I[1, 0] = I[0, 1]
34
35         I_inverse = np.linalg.inv(I/2)
36
37         CRLB_value[ind] = I_inverse[0, 0] + I_inverse[1, 1]
38
39     return CRLB_value
40
41
42 if __name__ == '__main__':
43     CRLB()

```

MLE.py

```

1  import numpy as np
2
3
4  def MLE_TDOA(noisy_distances, anchor_location, x_0):
5      ALPHA = 0.5
6      NUM_ITER = 100
7      DIMENSIONS = 2
8
9      num_anchors = anchor_location.shape[1]
10
11     x = np.zeros([2, NUM_ITER])
12     d = np.zeros(num_anchors)
13     h = np.zeros(num_anchors-1)
14     r = np.zeros(num_anchors-1)
15     delH = np.zeros([DIMENSIONS, (num_anchors-1)])
16
17     x[:, 0] = x_0
18     C = np.array([[2, -1, 0], [-1, 2, -1], [0, -1, 2]])
19     C_inverse = np.linalg.inv(C)
20
21     for iter in range(NUM_ITER - 1):
22         for j in range(num_anchors):
23             d[j] = np.linalg.norm(x[:, iter] - anchor_location[:, j])
24
25         for j in range(num_anchors - 1):
26             h[j] = d[j] - d[j + 1]
27             r[j] = noisy_distances[j] - noisy_distances[j + 1]
28
29         for j in range(num_anchors - 1):
30             for k in range(DIMENSIONS):
31                 delH[k, j] = ((x[k, iter]-anchor_location[k, j])/d[j]) - ((x[k, iter]-anchor_location[k, j+1])/d[j + 1])
32
33             x[:, iter + 1] = x[:, iter] + ALPHA * delH @ C_inverse @ (r - h)
34
35     return x[:, -1]
36
37
38 if __name__ == '__main__':
39     MLE_TDOA()

```

BLUE.py

```

1  import numpy as np
2
3
4  def BLUE_TDOA(noisy_distances, anchor_location, sigma2):
5      r = np.zeros(6)
6      r_dash = np.zeros(6)
7      H = np.zeros([6, 5])
8      gamma_vec = np.zeros(6)
9
10     C = np.array([[4, 1, 1, 1, 1, 0], [1, 4, 1, 1, 0, 1], [1, 1, 4, 0, 1, 1], [1, 1, 0, 4, 1, 1],
11                  [1, 0, 1, 1, 4, 1], [0, 1, 1, 1, 1, 4]])
12     C_inverse = np.linalg.inv(C)
13
14     num_anchors = anchor_location.shape[1]
15
16     ind = 0
17     for i in range(num_anchors):
18         for j in range(i + 1, num_anchors):
19             r[ind] = noisy_distances[i] - noisy_distances[j]
20             gamma_vec[ind] = np.linalg.norm(anchor_location[:, i]) ** 2 - np.linalg.norm(anchor_location[:, j]) ** 2
21             r_dash[ind] = r[ind]**2 - gamma_vec[ind] - 2*sigma2
22             ind = ind + 1
23
24     ind = 0
25     for anchor_ind in range(num_anchors - 1):
26         ls = list(range((anchor_ind), num_anchors-1))
27         for loc in ls:
28             H[ind, 0] = -2*(anchor_location[0, anchor_ind] - anchor_location[0, (loc+1)])
29             H[ind, 1] = -2*(anchor_location[1, anchor_ind] - anchor_location[1, (loc+1)])
30
31             H[ind, (loc+2)] = -2*r[ind]
32             ind = ind + 1
33
34     H_transpose = np.transpose(H)
35     W = np.linalg.inv(H_transpose @ C_inverse @ H) @ H_transpose @ C_inverse
36     x = W @ r_dash
37     xy = x[:2]
38
39     return xy
40
41
42 if __name__ == '__main__':
43     BLUE_TDOA()

```
