E1 244: Detection and Estimation Theory **(Due: 02/03/20)**

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,

$$\begin{bmatrix} r_{12} \\ r_{23} \\ r_{34} \end{bmatrix} = \begin{bmatrix} d_1 - d_2 \\ d_2 - d_3 \\ d_3 - d_4 \end{bmatrix} + \begin{bmatrix} n_{12} \\ n_{23} \\ n_{34} \end{bmatrix}$$
$$\underline{\mathbf{r}} = \underline{\mathbf{h}}(\underline{\mathbf{x}}) + \underline{\mathbf{n}}$$
$$\underline{\mathbf{r}} \sim \mathcal{N}(\underline{\mathbf{h}}(\underline{\mathbf{x}}), \mathbf{\Sigma_n})$$

where $\mathbf{\Sigma_n}$ is the noise covariance matrix, given by,

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

$$\underline{\mathbf{n}} = \mathbf{A}\underline{\mathbf{w}}$$
$$\begin{aligned} \mathbf{\Sigma_n} &= E[\underline{\mathbf{n}}\underline{\mathbf{n}}^T] \\ &= E[\mathbf{A}\underline{\mathbf{w}}\underline{\mathbf{w}}^T\mathbf{A}] \\ &= \mathbf{A}E[\underline{\mathbf{w}}\underline{\mathbf{w}}^T]\mathbf{A}^T \\ &= \mathbf{A}\Sigma_w\mathbf{A}^T \end{aligned}$$

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

$$P(\underline{\mathbf{r}};\underline{\mathbf{x}}) = \frac{1}{(2\pi)^{3/2}|\sum_n|^{1/2}} exp(-\frac{1}{2}(\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}}))^T\Sigma_n^{-1}(\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}})))$$
$$lnP(\underline{\mathbf{r}};\underline{\mathbf{x}}) = K - \frac{1}{2}(\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}}))^T\mathbf{\Sigma_n^{-1}}(\underline{\mathbf{r}} - \underline{\mathbf{h}}(\underline{\mathbf{x}}))$$

$$= K - \frac{1}{2}\sum_{i=1}^{3}\sum_{j=1}^{3}(r_i - h_i(\underline{\mathbf{x}}))[\mathbf{\Sigma_n^{-1}}]_{ij}(r_j - h_j(\underline{\mathbf{x}}))$$

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

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

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

$$\begin{aligned}
[\mathbf{I}(\underline{\mathbf{x}})]_{11} &= -E[\frac{\partial^2}{\partial x^2} lnP(\underline{\mathbf{r}};\underline{\mathbf{x}})] \\
&= \sum_{i=1}^{3}\sum_{j=1}^{3}[\mathbf{\Sigma_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 \mathbf{\Sigma_n^{-1}} \frac{\partial \underline{\mathbf{h}}(\underline{\mathbf{x}})}{\partial x} \\
[\mathbf{I}(\underline{\mathbf{x}})]_{12} &= -E[\frac{\partial^2}{\partial x \partial y} lnP(\underline{\mathbf{r}};\underline{\mathbf{x}})] \\
&= \sum_{i=1}^{3}\sum_{j=1}^{3}[\mathbf{\Sigma_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 \mathbf{\Sigma_n^{-1}} \frac{\partial \underline{\mathbf{h}}(\underline{\mathbf{x}})}{\partial y} \\
[\mathbf{I}(\underline{\mathbf{x}})]_{21} &= -E[\frac{\partial^2}{\partial y \partial x} lnP(\underline{\mathbf{r}};\underline{\mathbf{x}})] \\
&= \sum_{i=1}^{3}\sum_{j=1}^{3}[\mathbf{\Sigma_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 \mathbf{\Sigma_n^{-1}} \frac{\partial \underline{\mathbf{h}}(\underline{\mathbf{x}})}{\partial x} \\
[\mathbf{I}(\underline{\mathbf{x}})]_{22} &= -E[\frac{\partial^2}{\partial y \partial y} lnP(\underline{\mathbf{r}};\underline{\mathbf{x}})] \\
&= \sum_{i=1}^{3}\sum_{j=1}^{3}[\mathbf{\Sigma_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 \mathbf{\Sigma_n^{-1}} \frac{\partial \underline{\mathbf{h}}(\underline{\mathbf{x}})}{\partial y}
\end{aligned}$$

where,

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

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

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

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

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

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

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,

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

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 $\mathbf{\Sigma_n^{-1}}$ (for which the gradient descent theoretically converges if objective function is Lipschitz).

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

where,

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

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

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

(Solution) From the range difference equations we have,

$$r_{ij} = d_i - d_j + n_{ij}$$
$$r_{ij} + d_j = d_i + n_{ij}$$
$$r_{ij}^2 + 2r_{ij}dj + d_j^2 = d_i^2 + e_{ij}$$

where d_i and e_{ij} are given by,

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

Substituting in range equation,

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

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

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

$$\begin{bmatrix} r_{12}^2 \\ r_{13}^2 \\ r_{14}^2 \\ r_{23}^2 \\ r_{24}^2 \\ r_{34}^2 \end{bmatrix} = \begin{bmatrix} -2(\mathbf{x}_1 - \mathbf{x}_2)^T & -2r_{12} & 0 & 0 \\ -2(\mathbf{x}_1 - \mathbf{x}_3)^T & 0 & -2r_{13} & 0 \\ -2(\mathbf{x}_1 - \mathbf{x}_4)^T & 0 & 0 & -2r_{14} \\ -2(\mathbf{x}_2 - \mathbf{x}_3)^T & 0 & -2r_{23} & 0 \\ -2(\mathbf{x}_2 - \mathbf{x}_4)^T & 0 & 0 & -2r_{24} \\ -2(\mathbf{x}_3 - \mathbf{x}_4)^T & 0 & 0 & -2r_{34} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} + \begin{bmatrix} e_{12} \\ e_{13} \\ e_{14} \\ e_{23} \\ e_{24} \\ e_{34} \end{bmatrix} + \begin{bmatrix} \gamma_{12} \\ \gamma_{13} \\ \gamma_{14} \\ \gamma_{23} \\ \gamma_{24} \\ \gamma_{34} \end{bmatrix}$$

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

Concisely in vector notation,

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

where,

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

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.

$$E[\underline{\mathbf{e}}] = \begin{bmatrix} E[e_{12}] \\ E[e_{13}] \\ E[e_{14}] \\ E[e_{23}] \\ E[e_{24}] \\ E[e_{34}] \end{bmatrix} = 2\sigma^2 \underline{\mathbf{1}}_{\mathbf{6x1}}$$

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

We can now have a model with zero mean as,

$$\begin{gathered} \underline{\mathbf{r}} - \underline{\gamma} - 2\sigma^2 \underline{\mathbf{1}}_{\mathbf{6x1}} = \mathbf{H}\underline{\theta} + \underline{\mathbf{e}} - 2\sigma^2 \underline{\mathbf{1}}_{\mathbf{6x1}} \\ \underline{\mathbf{r}'} = \mathbf{H}\underline{\theta} + \underline{\mathbf{e}'} \\ \underline{\mathbf{r}'} = \underline{\mathbf{r}} - \underline{\gamma} - 2\sigma^2 \underline{\mathbf{1}}_{\mathbf{6x1}} \\ \underline{\mathbf{e}'} = \underline{\mathbf{e}} - 2\sigma^2 \underline{\mathbf{1}}_{\mathbf{6x1}} \\ \underline{\mathbf{e}'} \sim \mathcal{N}(0, \mathbf{C}) \end{gathered}$$

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

$$\begin{aligned} \mathbf{C} &= E[(\underline{\mathbf{e}'} - E[\underline{\mathbf{e}'}])(\underline{\mathbf{e}'} - E[\underline{\mathbf{e}'}])^T] \\ &= E[\underline{\mathbf{e}'}\underline{\mathbf{e}'}^T] \\ &= E[(\underline{\mathbf{e}} - 2\sigma^2 \underline{\mathbf{1}}_{\mathbf{6x1}})(\underline{\mathbf{e}} - 2\sigma^2 \underline{\mathbf{1}}_{\mathbf{6x1}})^T] \\ &= E[\underline{\mathbf{e}}\underline{\mathbf{e}}^T - \underline{\mathbf{e}}(2\sigma^2 \underline{\mathbf{1}}_{\mathbf{6x1}})^T - (2\sigma^2 \underline{\mathbf{1}}_{\mathbf{6x1}})]\underline{\mathbf{e}}^T + (2\sigma^2 \underline{\mathbf{1}}_{\mathbf{6x1}})(2\sigma^2 \underline{\mathbf{1}}_{\mathbf{6x1}})^T] \\ &= E[\underline{\mathbf{e}}\underline{\mathbf{e}}^T] - (2\sigma^2 \underline{\mathbf{1}}_{\mathbf{6x1}})(2\sigma^2 \underline{\mathbf{1}}_{\mathbf{6x1}})^T \end{aligned}$$

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

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

$$
\begin{aligned}
&= E[w_1^4 + w_2^4 + 6w_1^2w_2^2] - 4\sigma^4 \\
&= 3\sigma^4 + 3\sigma^4 + 6\sigma^4 - 4\sigma^4 \\
&= 8\sigma^4 \\
[\mathbf{C}]_{12} &= E[e_{12}e_{13}] - 4\sigma^4 \\
&= E[(w_1 - w_2)^2(w_1 - w_3)^2] - 4\sigma^4 \\
&= E[(w_1^2 - 2w_1w_2 + w_2^2)(w_1^2 - 2w_1w_3 + w_3^2)] - 4\sigma^4 \\
&= E[w_1^4 + w_1^2w_3^2 + w_1^2w_2^2 + +w_2^2w_3^2] - 4\sigma^4 \\
&= 3\sigma^4 + \sigma^4 + \sigma^4 + \sigma^4 - 4\sigma^4 \\
&= 2\sigma^4 \\
[\mathbf{C}]_{13} &= E[e_{12}e_{14}] - 4\sigma^4 \\
&= 2\sigma^4 \\
[\mathbf{C}]_{14} &= E[e_{12}e_{23}] - 4\sigma^4 \\
&= 2\sigma^4 \\
[\mathbf{C}]_{15} &= E[e_{12}e_{24}] - 4\sigma^4 \\
&= 2\sigma^4 \\
[\mathbf{C}]_{16} &= E[e_{12}e_{34}] - 4\sigma^4 \\
&= E[(w_1 - w_2)^2(w_3 - w_4)^2] - 4\sigma^4 \\
&= E[(w_1^2 - 2w_1w_2 + w_2^2)(w_3^2 - 2w_3w_4 + w_4^2)] - 4\sigma^4 \\
&= E[w_1^2w_3^2 + w_1^2w_4^2 + w_2^2w_3^2 + +w_2^2w_4^2] - 4\sigma^4 \\
&= \sigma^4 + \sigma^4 + \sigma^4 + \sigma^4 - 4\sigma^4 \\
&= 0
\end{aligned}
$$

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),

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

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

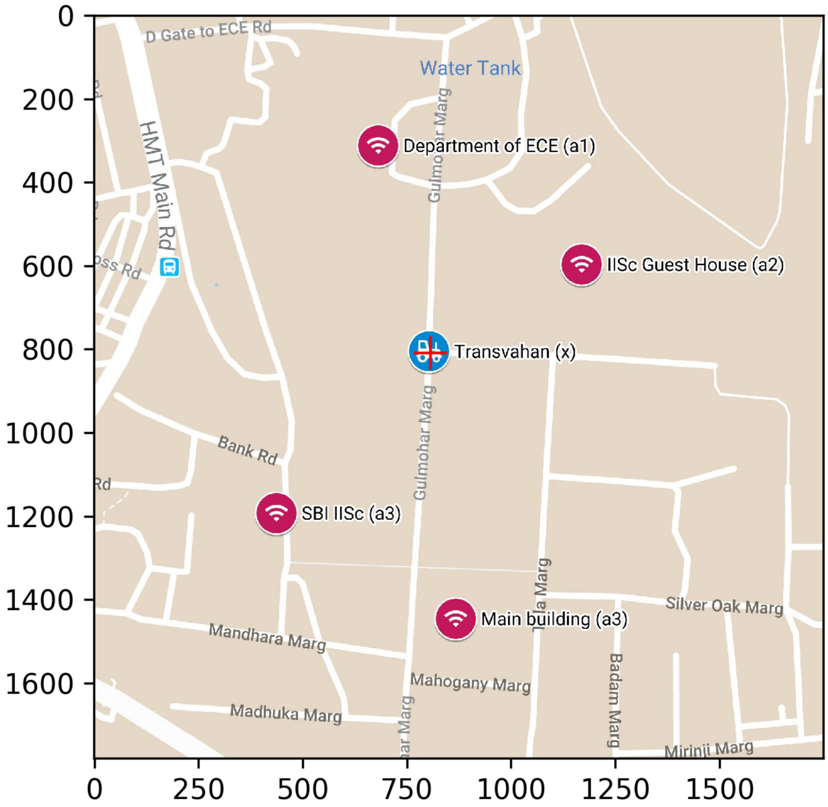

Figure 1: Estimated location of Transvahan using MLE

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

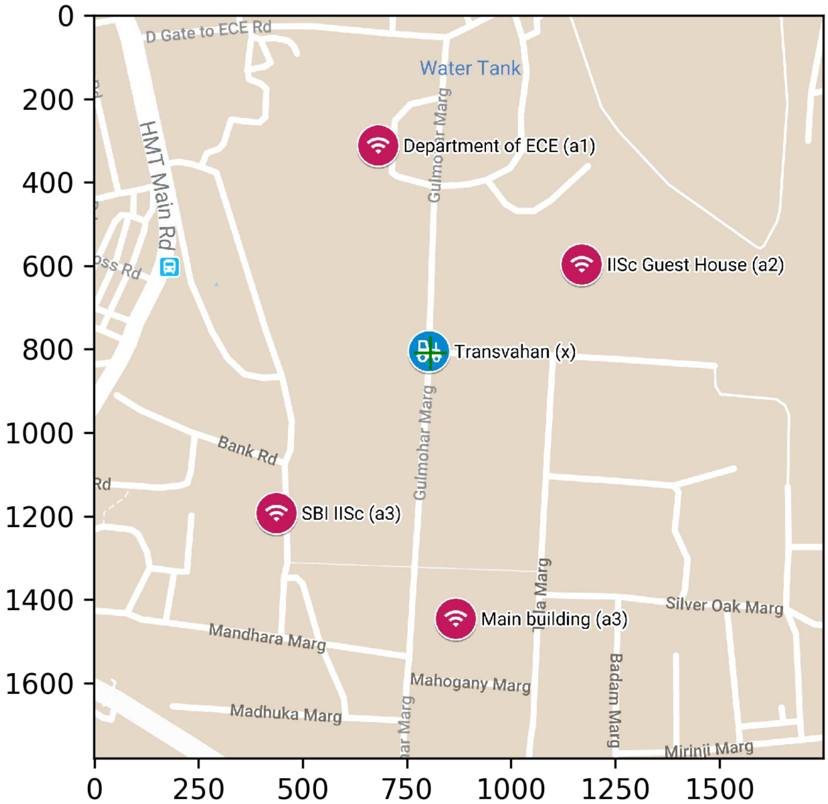

Figure 2: Estimated location of Transvahan using MLE

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

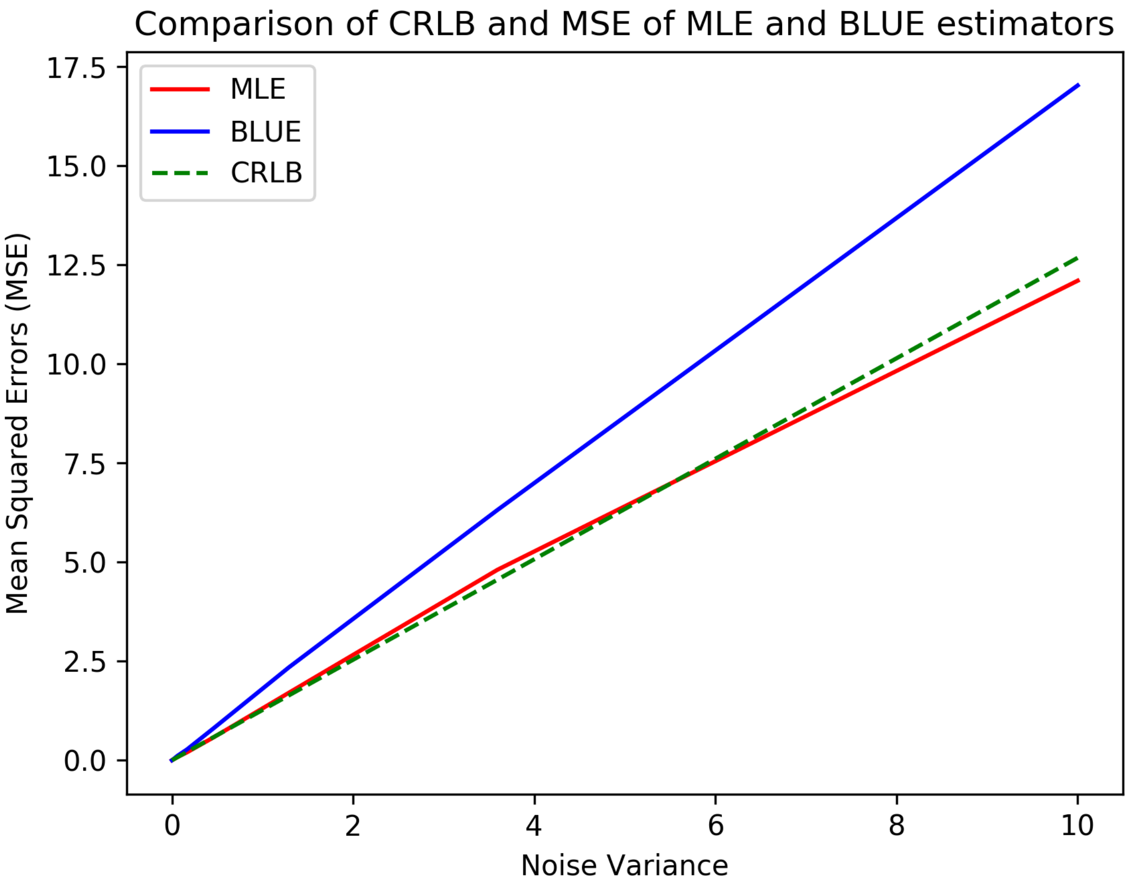

Figure 3: Estimated location of Transvahan using MLE

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 me made available, if necessary).

main.py

```python
import os
import numpy as np
import scipy.io
import matplotlib.pyplot as plt

from MLE import MLE_TDOA
from BLUE import BLUE_TDOA
from CRLB import CRLB


def MSE_MLE(noisy_distances, anchor_location, x_0, target_location):
    (xdim, ydim, zdim) = noisy_distances.shape
    mse = np.zeros(zdim)

    for i in range(zdim):
        for j in range(ydim):
            x_estimate = MLE_TDOA(noisy_distances[:, j, i], anchor_location, x_0)
            u = x_estimate - target_location
            mse_y = np.linalg.norm(u) ** 2
            mse[i] = mse[i] + mse_y

    mse = mse / ydim

    return mse


def MSE_BLUE(noisy_distances, anchor_location, sigma2, target_location):
    (xdim, ydim, zdim) = noisy_distances.shape
    mse = np.zeros(zdim)

    for i in range(zdim):
        for j in range(ydim):
            x_estimate = BLUE_TDOA(noisy_distances[:, j, i], anchor_location, sigma2[i])
            u = x_estimate - target_location
            mse_y = np.linalg.norm(u) ** 2
            mse[i] = mse[i] + mse_y

    mse = mse / ydim

    return mse


def main():
    try:
        os.makedirs('./results')
    except:
        pass

```

```
    TDOA_data = scipy.io.loadmat('./data/TDOA_data.mat')
    anchor_location = TDOA_data['anchor_location']
    anchor_location = anchor_location.astype(int)
    noisy_distances = TDOA_data['noisy_distances']
    target_location = TDOA_data['target_location']
    target_location = target_location.astype(float)
    target_location = [target_location[0][0], target_location[1][0]]
    sigma2 = TDOA_data['sigma2']

    # Problem 1
    x_0 = [0, 0]
    x = MLE_TDOA(noisy_distances[:, 1, 1], anchor_location, x_0)
    map = plt.imread('./data/mapimage.jpeg')
    filename = './results/MLE_map_loc.png'

    plt.figure()
    plt.imshow(map)
    plt.plot(x[0], x[1], '+', color='red', markersize=12)
    plt.savefig(filename, dpi=300)

    # Problem 2
    x = BLUE_TDOA(noisy_distances[:, 1, 1], anchor_location, sigma2[1])
    map = plt.imread('./data/mapimage.jpeg')
    filename = './results/BLUE_map_loc.png'

    plt.figure()
    plt.imshow(map)
    plt.plot(x[0], x[1], '+', color='green', markersize=12)
    plt.savefig(filename, dpi=300)

    # Problem 3
    MSE_MLE_value = MSE_MLE(noisy_distances, anchor_location, x_0, target_location)
    MSE_BLUE_value = MSE_BLUE(noisy_distances, anchor_location, sigma2, target_location)
    CRLB_value = CRLB(target_location,anchor_location,sigma2)
    filename = './results/MSE.png'

    plt.figure()
    plt.plot(sigma2, MSE_MLE_value, color='red', label='MLE')
    plt.plot(sigma2, MSE_BLUE_value, color='blue', label='BLUE')
    plt.plot(sigma2, CRLB_value, 'g--', label = 'CRLB')
    plt.title('Comparison of CRLB and MSE of MLE and BLUE estimators')
    plt.xlabel('Noise Variance')
    plt.ylabel('Mean Squared Errors (MSE)')
    plt.legend()
    plt.savefig(filename, dpi=300)


if __name__ == '__main__':
    main()
```

CRLB.py

```
import numpy as np


def CRLB(x, anchor_location, sigma2):
    CRLB_value = np.zeros(len(sigma2))
    DIMENSIONS = 2
    num_anchors = anchor_location.shape[1]

    for ind in range(len(sigma2)):
        d = np.zeros(num_anchors)
        h = np.zeros(num_anchors-1)
        I = np.zeros([DIMENSIONS, DIMENSIONS])
        delH = np.zeros([DIMENSIONS, (num_anchors-1)])

        C = sigma2[ind] * np.array([[2, -1, 0], [-1, 2, -1], [0, -1, 2]])
        C_inverse = np.linalg.inv(C)

        for j in range(num_anchors):
            d[j] = np.linalg.norm(x - anchor_location[:, j])

        for j in range(num_anchors-1):
            h[j] = d[j] - d[j + 1]

        for j in range(num_anchors-1):
            for k in range(DIMENSIONS):
                delH[k, j] = ((x[k] - anchor_location[k, j]) / d[j]) - ((x[k] - anchor_location[k, j + 1]) / d[j + 1])

        for i in range(num_anchors-1):
            for j in range(num_anchors-1):
                I[0, 0] = I[0, 0] + C_inverse[i, j] * (delH[0, i] * delH[0, j] + delH[0, j] * delH[0, i])
                I[0, 1] = I[0, 1] + C_inverse[i, j] * (delH[1, i] * delH[0, j] + delH[1, j] * delH[0, i])
                I[1, 1] = I[1, 1] + C_inverse[i, j] * (delH[1, i] * delH[1, j] + delH[1, j] * delH[1, i])
                I[1, 0] = I[0, 1]

        I_inverse = np.linalg.inv(I/2)

        CRLB_value[ind] = I_inverse[0, 0] + I_inverse[1, 1]

    return CRLB_value


if __name__ == '__main__':
    CRLB()
```

MLE.py

```python
import numpy as np


def MLE_TDOA(noisy_distances, anchor_location, x_0):
    ALPHA = 0.5
    NUM_ITER = 100
    DIMENSIONS = 2

    num_anchors = anchor_location.shape[1]

    x = np.zeros([2, NUM_ITER])
    d = np.zeros(num_anchors)
    h = np.zeros(num_anchors-1)
    r = np.zeros(num_anchors-1)
    delH = np.zeros([DIMENSIONS, (num_anchors-1)])

    x[:, 0] = x_0
    C = np.array([[2, -1, 0], [-1, 2, -1], [0, -1, 2]])
    C_inverse = np.linalg.inv(C)

    for iter in range(NUM_ITER - 1):
        for j in range(num_anchors):
            d[j] = np.linalg.norm(x[:, iter] - anchor_location[:, j])

        for j in range(num_anchors - 1):
            h[j] = d[j] - d[j + 1]
            r[j] = noisy_distances[j] - noisy_distances[j + 1]

        for j in range(num_anchors - 1):
            for k in range(DIMENSIONS):
                delH[k, j] = ((x[k, iter]-anchor_location[k, j])/d[j]) - ((x[k, iter]-anchor_location[k, j+1])/d[j + 1])

        x[:, iter + 1] = x[:, iter] + ALPHA * delH @ C_inverse @ (r - h)

    return x[:, -1]


if __name__ == '__main__':
    MLE_TDOA()
```

BLUE.py

```
import numpy as np


def BLUE_TDOA(noisy_distances, anchor_location, sigma2):
    r = np.zeros(6)
    r_dash = np.zeros(6)
    H = np.zeros([6, 5])
    gamma_vec = np.zeros(6)

    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],
                  [1, 0, 1, 1, 4, 1], [0, 1, 1, 1, 1, 4]])
    C_inverse = np.linalg.inv(C)

    num_anchors = anchor_location.shape[1]

    ind = 0
    for i in range(num_anchors):
        for j in range(i + 1, num_anchors):
            r[ind] = noisy_distances[i] - noisy_distances[j]
            gamma_vec[ind] = np.linalg.norm(anchor_location[:, i]) ** 2 - np.linalg.norm(anchor_location[:, j]) ** 2
            r_dash[ind] = r[ind]**2 - gamma_vec[ind] - 2*sigma2
            ind = ind + 1

    ind = 0
    for anchor_ind in range(num_anchors - 1):
        ls = list(range((anchor_ind), num_anchors-1))
        for loc in ls:
            H[ind, 0] = -2*(anchor_location[0, anchor_ind] - anchor_location[0, (loc+1)])
            H[ind, 1] = -2*(anchor_location[1, anchor_ind] - anchor_location[1, (loc+1)])

            H[ind, (loc+2)] = -2*r[ind]
            ind = ind + 1

    H_transpose = np.transpose(H)
    W = np.linalg.inv(H_transpose @ C_inverse @ H) @ H_transpose @ C_inverse
    x = W @ r_dash
    xy = x[:2]

    return xy


if __name__ == '__main__':
    BLUE_TDOA()
```