All the tasks are completed using the MATLAB wrapper for the GTSAM library.

Task 1. 2D Graph SLAM

A. Function to read in the 2D Intel dataset in G2O format.
- Opening the required 2D dataset file.
- Using the 'textscan' function in MATLAB to extract the required information.
- The extracted information is then converted to usable matrix form.

B. Batch Solution.
- For the batch solution the entire graph is constructed using the edge information in the g2o file and solved altogether at the end.
- First, a nonlinear 2D graph is initialized and a prior factor is added with a noise model of sigma = 0.3
- Next the edge constraints are added to the graph, along with the initial estimates of the vertices.
- The Gauss Newton solver is implemented to optimize the given data, which results in the map shown in Figure 1.
- The upper triangular information matrix provided in the g2o file is used to calculate the required noise covariance matrix which is further on passed on as an input argument when the corresponding edge data is added to the graph. **This function is used in every implementation further on in the assignment.**

C. Incremental Solution.
- In this implementation we optimize the trajectory incrementally with each node as the graph data is added gradually.
- First, as was done in the batch implementation, a nonlinear 2D graph is initialized and a prior factor is added with a noise model of sigma = 0.3
- Then with the initial estimate as the origin, isam2 solver is used to find the initial estimate.
- The calculated initial estimate is used in conjunction with the graph values of the particular node to solve for estimate of the next node. This process is then repeated for the number of nodes (vertices) in the graph.
- In addition, to the steps described above, the graph is reinitialized at each iteration to reduce the use of memory allocation in processing as the same factor graph gets overwritten for the number of nodes in the graph. Since the estimates calculated after each iteration is the only useful information, the graph can be reinitialized without the fear of losing any data for computation.

- The optimized trajectory is then plotted against the initial estimates as shown in Figure 2.

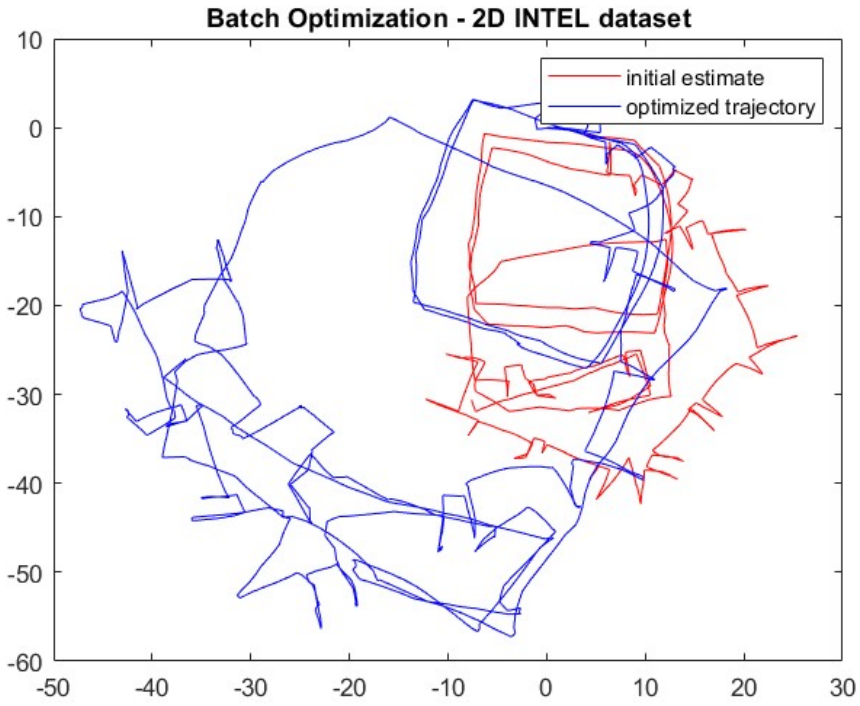

Figure 1. Batch Solution using GaussNewton solver on 2D Intel dataset.

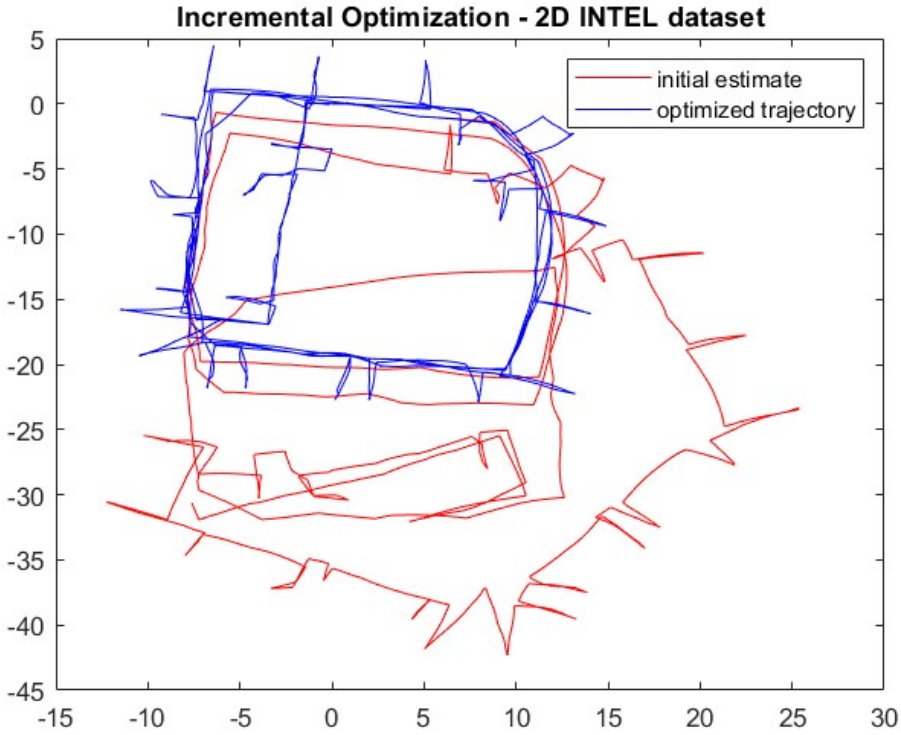

Figure 2. Incremental Solution using iSAM2 solver on 2D Intel dataset.

Task 2. 3D Graph SLAM

A. Function to read in the 3D Parking Garage dataset in G2O format.
   - Opening the required 3D dataset file.
   - Using the 'textscan' function in MATLAB to extract the required information.
   - The extracted information is then converted to usable matrix form.

B. Batch Solution.
   - For the batch solution the entire graph is constructed using the edge information in the g2o file and solved altogether at the end.
   - First, a nonlinear graph is initialized and a prior factor is added with a noise model of sigma = 1e-5
   - Next the edge constraints are added to the graph, along with the initial estimates of the vertices. The MATLAB function 'quat2rotm' is used to calculate the rotation matrix for the given quaternions corresponding to each node.
   - The Gauss Newton solver is implemented to optimize the given data, which results in the map shown in Figure 3 and Figure 4.

C. Incremental Solution.
   - In this implementation we optimize the trajectory incrementally with each node as the graph data is added gradually.
   - First, as was done in the batch implementation, a nonlinear graph is initialized and a prior factor is added with a noise model of sigma = 1e-5
   - Then with the initial estimate as the origin, isam2 solver is used to find the initial estimate.
   - The calculated initial estimate is used in conjunction with the graph values of the particular node to solve for estimate of the next node. This process is then repeated for the number of nodes (vertices) in the graph. The MATLAB function 'quat2rotm' is used to calculate the rotation matrix for the given quaternions corresponding to each node.
   - In addition, to the steps described above, the graph is reinitialized at each iteration to reduce the use of memory allocation in processing as the same factor graph gets overwritten for the number of nodes in the graph. Since the estimates calculated after each iteration is the only useful information, the graph can be reinitialized without the fear of losing any data for computation.
   - The optimized trajectory is then plotted against the initial estimates as shown in Figure 5 and Figure 6.

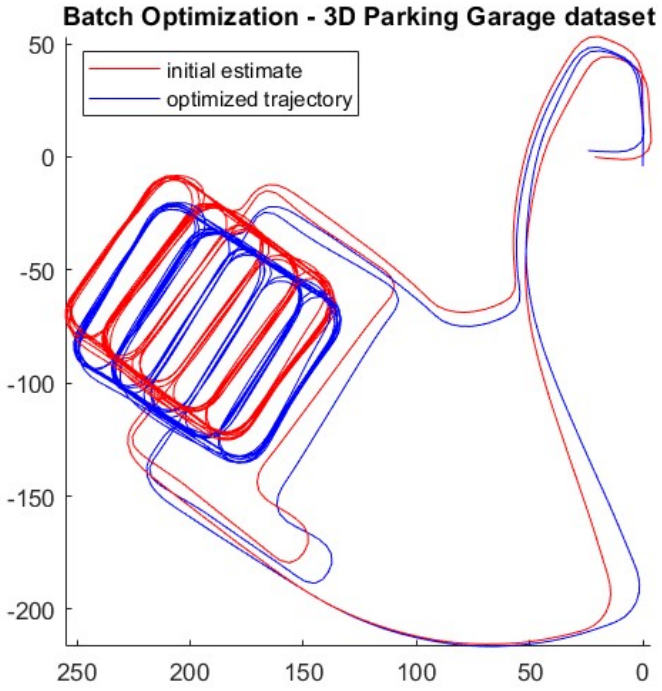

Figure 3. Top-Down View - Batch Solution using GaussNewton solver on 3D Park Garage dataset.

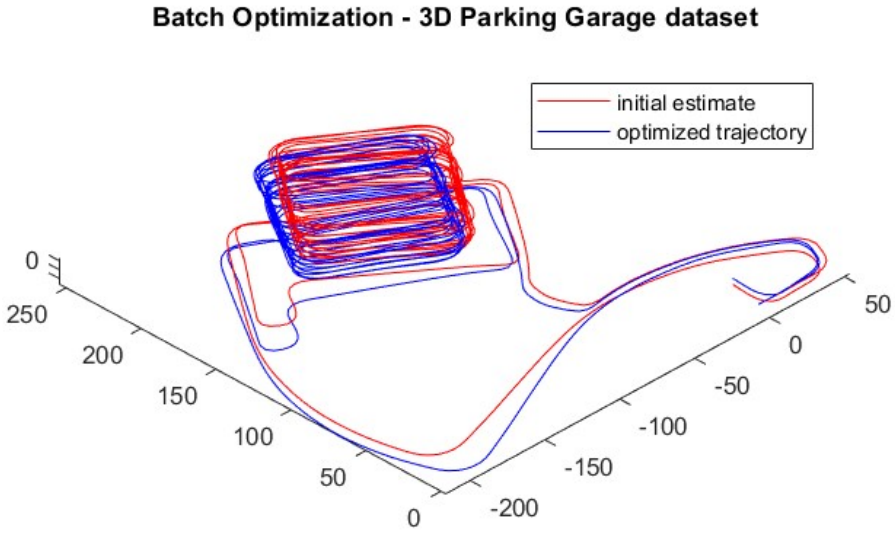

Figure 4. Isometric View - Batch Solution using GaussNewton solver on 3D Park Garage dataset.

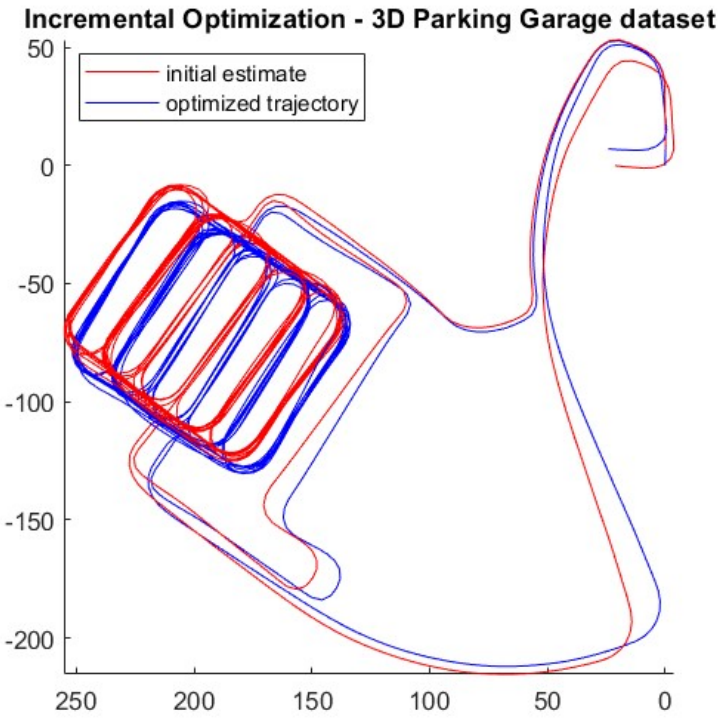

Figure 5. Top-Down View - Incremental Solution using iSAM2 solver on 3D Park Garage dataset.

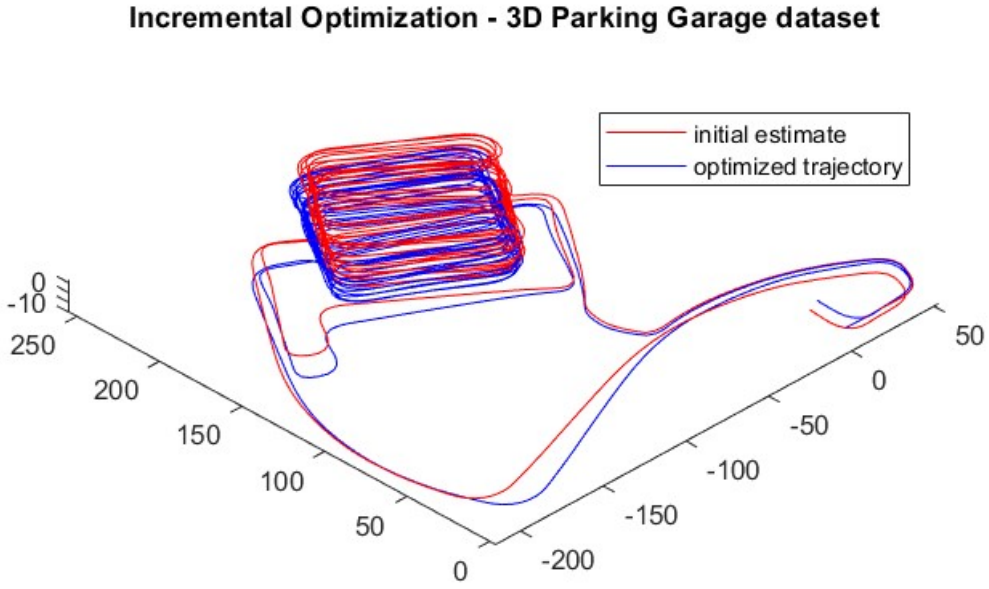

Figure 6. Isometric View - Incremental Solution using iSAM2 solver on 3D Park Garage dataset.

Conclusion:

It can be clearly seen from the above implementations that iSAM2 produces a more accurate result in comparison to a batch optimizer with GaussNewton solver. In the case of the batch computation using the GaussNewton solver the entire graph is generated and then optimized on the basis of the initial estimates provided by the g2o file. Whereas, in the case of iSAM2 the edge information is added to the graph incrementally while the iSAM2 solver is also implemented simultaneously for each node (vertex). Hence, the results are generated for each node which are then further used as the estimates for the next iteration. Therefore, where the batch implementation depends on the initial estimates provided by the g2o file, incremental implementation depends totally on the edge information fed into the nonlinear graph and hence, produces a better estimate of trajectory than the batch implementation.

It is observed that there is quite some difference of the trajectory estimated between the batch solution and the incremental solution in the two cases of the 2D Intel data and the 3D park garage data. As in the case of the 2D dataset the iSAM2 solver is able to improve the trajectory from the batch solution a lot , but there is not much difference in case of the 3D dataset. This is because of the poor initial estimates provided for the 2D dataset, which in turn produces a bad solution in case of the batch implementation. But since the initial estimates provided of the 3D dataset are almost accurate, the batch solution for 3D produces the same result as anticipated using the incremental implementation.