COMP0130 Robot Vision and Navigation
Coursework 2: Graph-based Optimisation and SLAM

Jeffrey Li, Dannielle Lee
zcabbli@ucl.ac.uk, ucabdl5@ucl.ac.uk

March 24, 2024

Question 1

1. a. A diagram of the structure of the factor graph is displayed below:

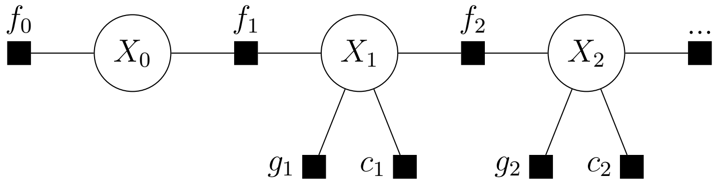


Figure 1: Factor Graph to Predict and Update Robot Pose Using GPS and Compass Sensors.

The vertices, X_n, represent the vehicle state vertices describing the robot pose at each timestep. There are three types of edges - f_n, representing the vehicle process model edges, g_n, representing the GPS measurements, and finally c_n, representing the compass measurements. It should be noted that f_0 is not an edge representing the process model, but rather the prior, setting initial conditions for the initial state vertex, X_0.

We would implement the vertices and edges using format specified by *g2o* framework for graph optimization, which we would define `computeError` and `linearizeOplus` for edges and `oplus` for state vertices. We will follow the same notation as specified in the appendix, where vehicle state is denoted as x_k and the horizontal position of the vehicle is denoted as x_k. This is not to be confused with the state vertices X_n.

In the current context, each state comprises a translation (x, y) and a rotational component ψ. While the translation (x, y) naturally resides within a Euclidean space, the rotational component ψ extends across the non-Euclidean rotation group SO(2) in 2D within this context. Furthermore, there is also compass measurements that also extends across the same non-Euclidean rotation group. To address the challenge of singularities, g2o adopts the strategy of treating the underlying space as a manifold. Within this framework, an operator denoted as $\boxplus$ is introduced, facilitating the mapping of a local variation Δx from the Euclidean space to a variation on the manifold. By utilizing this operator, a new error function can be devised. A manifold, in this context, refers to a mathematical space that may not be Euclidean on a global scale but can be approximated as Euclidean on a local scale. [1]

The $\boxplus$ operator, defined as $\boxplus : \texttt{Dom}(\mathbf{x}_i)\times\texttt{Dom}(\mathbf{\Delta x}_i) \rightarrow\texttt{Dom}(\mathbf{x}_i)$, plays a crucial role in handling angular discontinuities by enforcing the domain of $[-\pi, \pi]$ on any angular measurements or operations. This discontinuity is particularly significant in our solution, especially when the vehicle executes turns, which may lead to deviations in state estimates and discrepancies between process edges and compass measurements, as illustrated in **Question 1C**.

Vehicle State Vertex X_n:

- `oplus`: This function will take in an updating vector $\Delta \mathrm{x}$ the same dimension as the vehicle state.
$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta \mathrm{x} \tag{1}$$
Note that here we need to normalize the heading of the state to account for angular discontinuity.
$$\mathrm{x}_{k+1}(3) = \texttt{g2o.stuff.normalize\_theta}(\mathrm{x}_{k+1}(3)) \tag{2}$$

Prior Edge f_0:

- `computeError`: Where z is the initial condition
$$errorZ = \mathrm{x}_0 - z \tag{3}$$
We account for angular discontinuity with the following:
$$errorZ(3) = \texttt{g2o.stuff.normalize\_theta}(error(3)) \tag{4}$$
- `linearizeOplus`: Here the Jacobian J is simply the identity matrix
$$J = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Vehicle Process Edge f_n:

- `computeError`:
$$v_k = (\Delta T_k M(\psi_k))^{-1}(\mathrm{x}_{k+1} - \mathrm{x}_k) - u_k \tag{6}$$
We would need to normalize the heading component to account for angular discontinuity:
$$v_k(3) = \texttt{g2o.stuff.normalize\_theta}(v_k(3)) \tag{7}$$

- `linearizeOplus`: We first denote the following variables

$$\Delta x = x_{k+1} - x \tag{8}$$

$$\Delta y = y_{k+1} - y \tag{9}$$

Here as the edge is binary, connected states vertex from time step k to $k+1$, there would be two Jacobian matrices for the error, J_1 is with respect to x_k and J_2 to x_{k+1}.

$$J_1 = \Delta T^{-1} \begin{bmatrix} -cos(\psi_k) & -sin(\psi_k) & -\Delta x sin(\psi_k) + \Delta y cos(\psi_k) \\ sin(\psi_k) & -cos(\psi_k) & -\Delta x cos(\psi_k) - \Delta y sin(\psi_k) \\ 0 & 0 & -1 \end{bmatrix} \tag{10}$$

$$J_2 = (\Delta T_k M(\psi_k))^{-1} \tag{11}$$

Computation of the Jacobian J_1 is different as M composes of ψ_k which is a constant with respect to $\mathrm{x_{k+1}}$.

GPS Measurement Edge g_n:

- `computeError`:

$$w_k^G = z_k^G - \begin{bmatrix} x_k \\ y_k \end{bmatrix} - M(\psi_k) \begin{bmatrix} \Delta x_G \\ \Delta y_G \end{bmatrix} \tag{12}$$

- `linearizeOplus`: Taking derivative of the error w_k^G with respect to state x_k, we get Jacobian matrix J. Since the error term doesn't have a component for heading ψ_k, the Jacobian component for it is zero.

$$J = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \tag{13}$$

Compass Measurement Edge c_n:

- `computeError`:

$$w_k^C = z_k^C - \begin{bmatrix} \psi_k \end{bmatrix} - \begin{bmatrix} \Delta \psi_C \end{bmatrix} \tag{14}$$

Note that here we need to handle angular discontinuity, thus we would in fact get:

$$w_k^C = \texttt{g2o.stuff.normalize\_theta}(w_k^C) \tag{15}$$

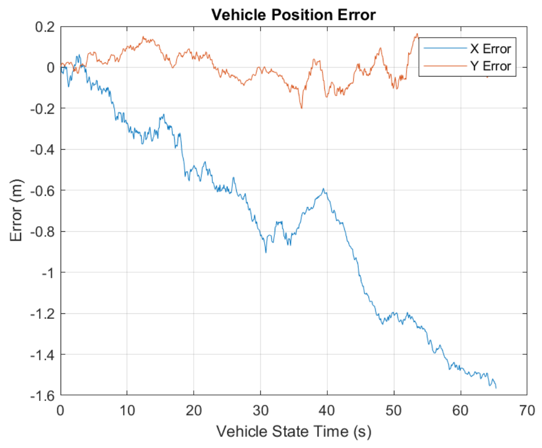
- `linearizeOplus`: Computing Jacobian matrix of error with respect to vehicle state x_k. Here the error is only in terms of the heading, thus the Jacobian for x_k and y_k is simply zero.

$$J = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix} \tag{16}$$

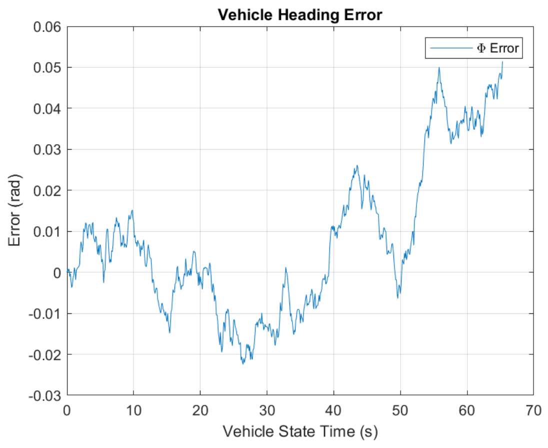
b. i. Functions modified for this question are the `computeError` and `linearizeOplus` methods of **VehicleKinematicsEdge** class, and the `handlePredictToTime` method of **DriveBotSLAMSystem** class.

ii. In this scenario, the vehicle has no access to any observations and estimations rely solely on the process model for vehicle kinematics. In terms of the vehicle position and heading error (Figure 2), they display a rather noisy and random behaviour due to the perturbation of noise added to the control signal and differs between runs. To better visualize the underlying behaviour of the state error, we generated figures for when the system is not perturbed by noise (Figure 3) by setting `perturbWithNoise` flag set to false. Here we would clearly see that the absolute error increases drastically after around 35 seconds, which is when the vehicle does its first anti-clockwise turn. For heading error, it drops slightly and flattens after this rapid increase.

    The reason why the absolute errors increase so rapidly after turning is that the control signal used is rather abrupt for both heading and position (Figure 4), where we would observe sudden spikes in speed and angular velocity inputs during the turn, which makes it difficult to accurately estimate the state of the vehicle using the process model only given the set time step length that is currently set as 0.1. We experimented with lower $DT = 0.01, 0.001$ as well as less rapid inputs and found that the error significantly dropped. Theoretically, given small enough DT, the error would tend to 0 in the case without perturbation of noise.

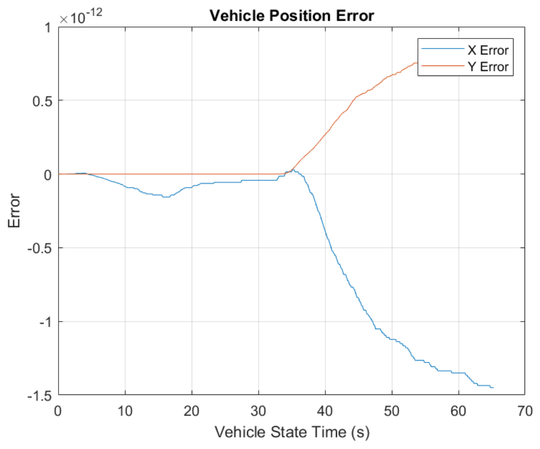


(a) Position Error

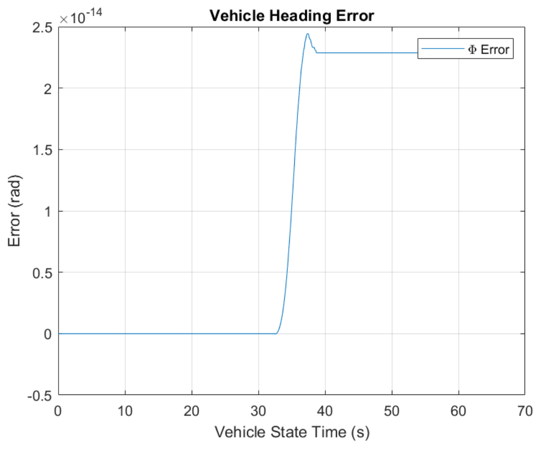


(b) Heading Error

Figure 2: Question 1B State Error

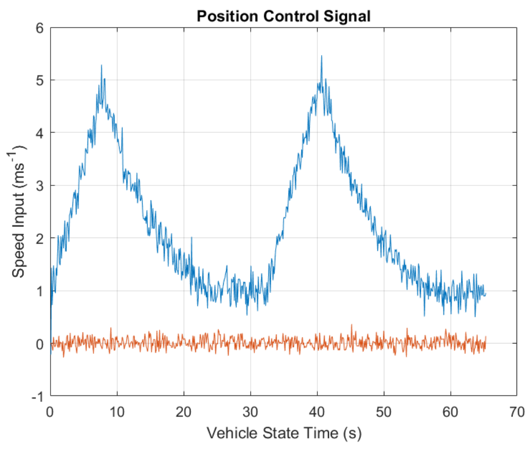


(a) Position Error

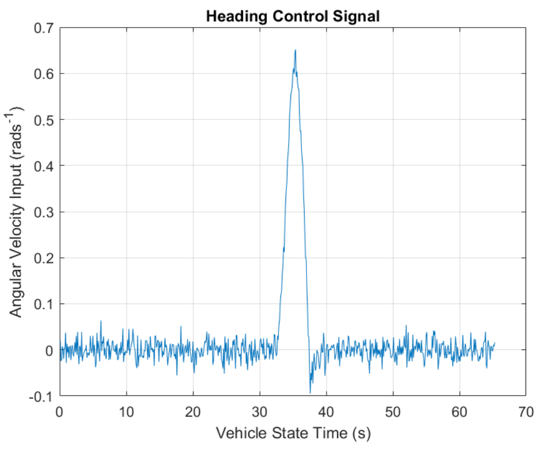


(b) Heading Error

Figure 3: Question 1B State Error With No Perturbation



(a) Position Control

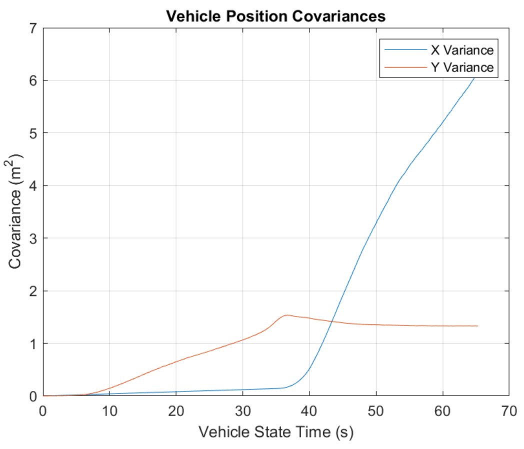


(b) Heading Control

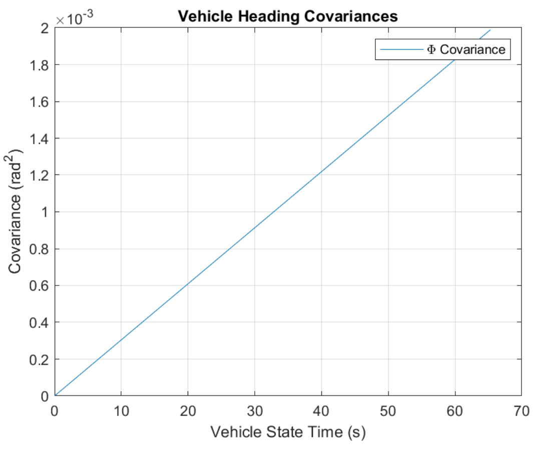
Figure 4: Question 1B Control Signals

As for the covariance of vehicle position (Figure 5), the covariance in the x coordinate of the vehicle is initially small and increases slowly, which also increases drastically at around 35 seconds. On the other hand, the covariance of the y coordinate initially increases quite rapidly but then reduces slightly and plateaus after around 35 seconds.

CONTINUED



(a) Position Covariance



(b) Heading Covariance

Figure 5: Question 1B State Covariance

Similar to the phenomenon observed towards the conclusion of Lecture 6C with the Particle filter demonstration, covariance tends to increase in the direction perpendicular to the direction of motion. This could be explained by the covariance estimation method, which involves the inverse of the Hessian evaluated through Laplace Approximation on the state vertex at each time step. When the vehicle accelerates parallel to the x axis, it results in a larger Hessian component related to x and a smaller component related to y (due to minimal acceleration noise in the y direction). Consequently, the x component of the inverse Hessian becomes small, while the y component becomes large, leading to larger covariance in the y direction and smaller covariance in the x direction. The same rationale applies when the vehicle moves parallel to the y axis after a turn. Regarding covariance in vehicle heading, it appears to increase linearly with time due to accumulating uncertainty over time without any observations.

We have verified that these behaviours in covariance are not the result of having noise in the system, as the same behaviour arises while having the `perturbWithNoise` flag set to false.

We observed that the residual value remains at zero through the entire optimization, which is because, in this scenario, there are no observation measurements fed into the system, thus residual value being the metric for evaluating the goodness of fit between estimated state and observation measurements would be zero since the model can fit the process model perfectly.

c. i. The *q1_c.m* file has been modified so that the system configuration would enable compass measurements. We verify that it is indeed faulty by observing the covariance ellipse during the run. The centre of the ellipse represents the current estimate (mean) of the (x, y) location of the vehicle. Running the faulty implementation, we would get a run where the ellipse would run off while turning (See Figure 6).

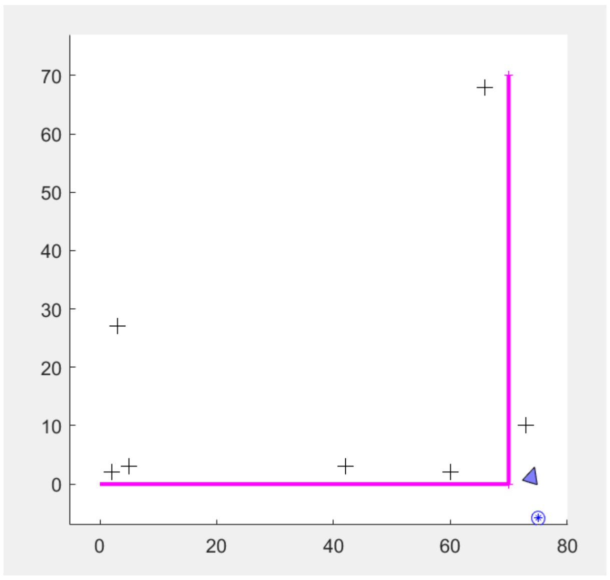
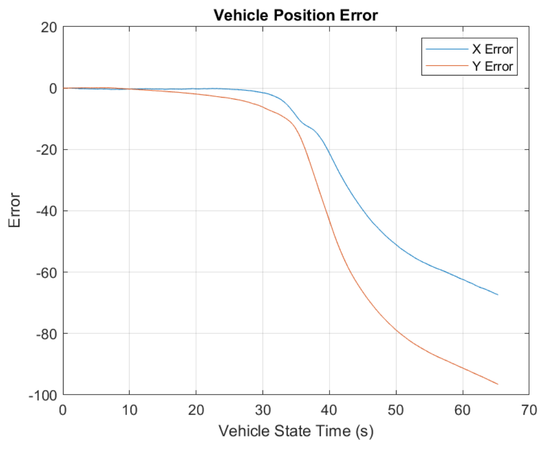


Figure 6: Covariance Ellipse drifting off when turning

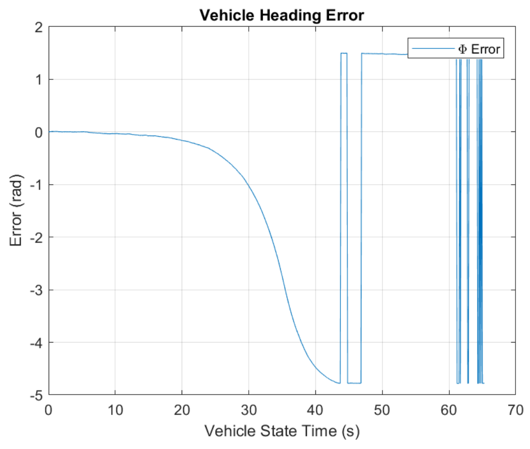
This observation is further substantiated by a scrutiny of the State Error (Figure 7) and Chi-Squared metrics (Figure 8). Noteworthy is the persistent divergence of the error in the (x, y) coordinates, reaching magnitudes in the order of hundreds. Simultaneously, the Chi-Squared value rises significantly to the order of 10^5. It is crucial to recognize that the *Levenberg-Marquardt* algorithm seeks to minimize the Chi-Squared value, and the substantial increase observed here strongly suggests a fault.

The root cause of this problem stems from the gradual divergence of state estimates from the true vehicle state. The manifestation of this error becomes apparent during turns. Notably, both the error and chi-squared values remain relatively low until around 35 seconds. It is strongly indicated that the issue is linked to angles not being normalized within the $[-\pi, \pi]$ range.

CONTINUED



(a) Position Error



(b) Heading Error

Figure 7: Faulty Compass Error Characteristics

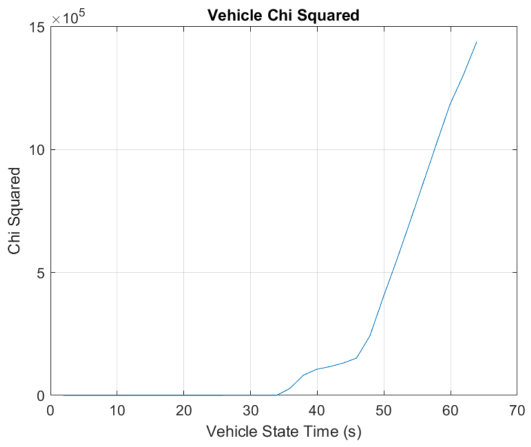


Figure 8: Faulty Compass Chi-Squared Characteristics

ii. The compass is fixed by modifying the `computeError` method of **Compass-MeasurementEdge** class, where an additional `g2o.stuff.normalize_theta` function is used to normalize the angle error.

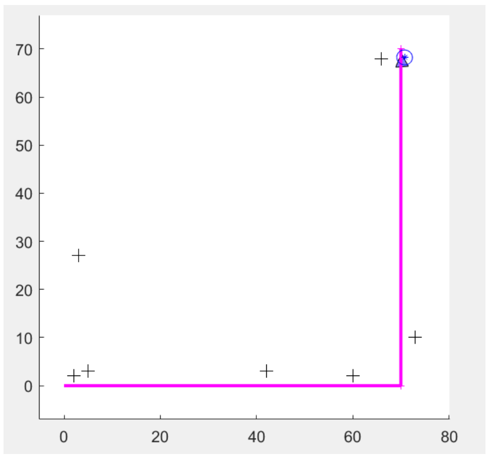
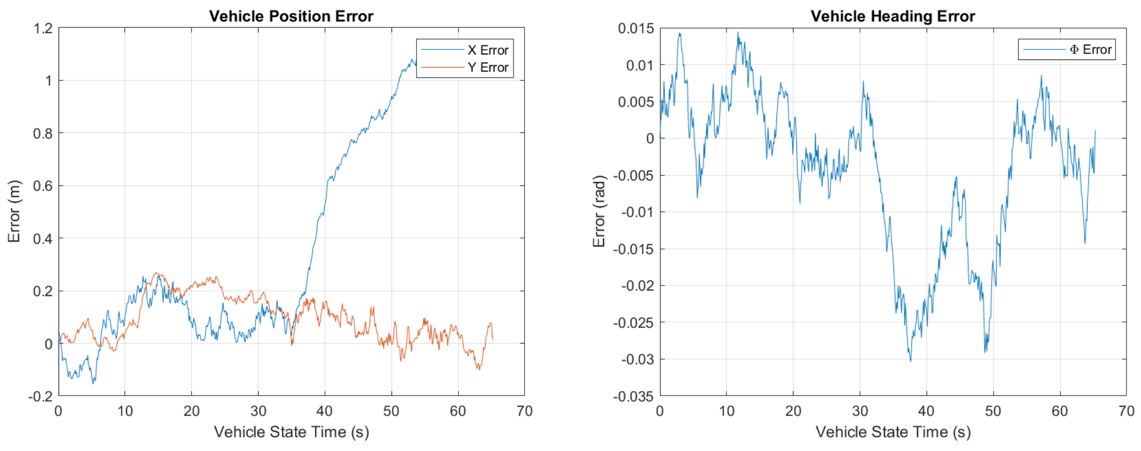


Figure 9: Covariance Ellipse remains intact with the vehicle



(a) Position Error (b) Heading Error

Figure 10: Fixed Compass Error Characteristics

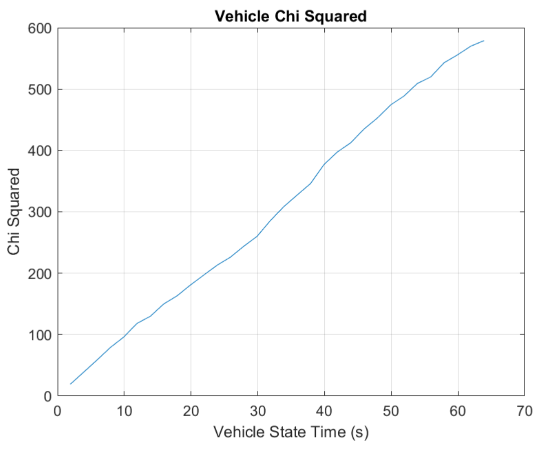


Figure 11: Fixed Compass Chi-Squared Characteristics

CONTINUED

We demonstrate that this resolved the issue, as depicted in Figure 9, where the covariance ellipse of the vehicle's position remains closely associated with the vehicle throughout the entire run. This affirmation is further supported by the State Error and Chi-Squared plots (Figure 10 11), where the error ceases to consistently escalate as time goes on after approximately 35 seconds, and the Chi-Squared value is kept comparatively low. Despite this, the state error might still display some noise, and the Chi-Squared value keeps rising. It's important to emphasize that this escalation is a result of the system's inherent noise, and it is unrelated to the implementation of the compass measurement edge.

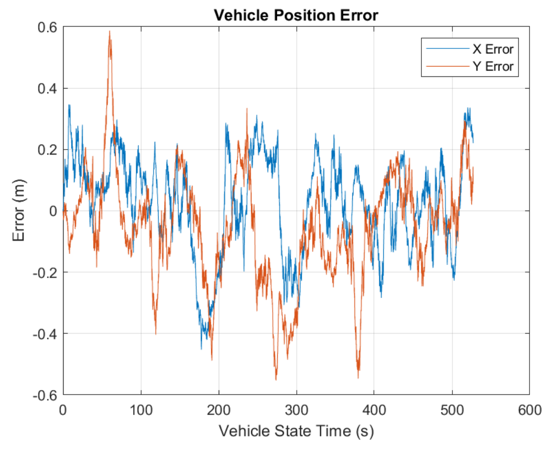
d. i. Functions modified for this question are the `computeError` and `linearizeOplus` methods of **GPSMeasurementEdge** class, and the `handleGPSObservationEvent` method of **DriveBotSLAMSystem** class.

ii. Looking at the heading error (Figure 12), an initial observation would be that it is very low (close to zero), except occasionally, the heading error would have a sudden spike. This might be caused by angular discontinuities in computing the heading, or the fact that the GPS measurement is in 2D, so sudden errors may arise from estimating the heading from the x and y coordinates. As for the behaviour of position error (Figure 12, we would see that this resembles some form of random noise, which is expected due to the additional perturbation of noise added to the GPS measurements.

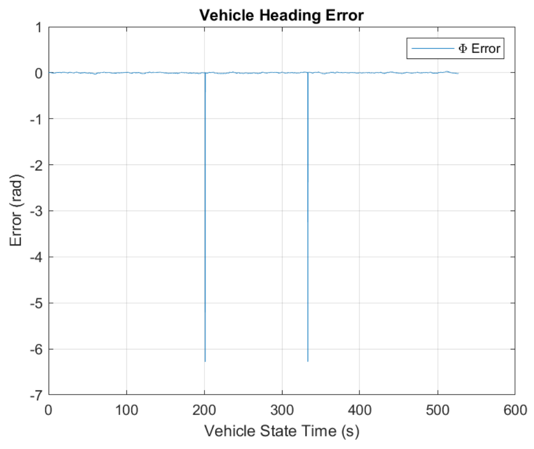
In terms of the position and heading covariance (Figure 13) of the vehicle, we would observe an oscillating behaviour for covariance, and at the very end, covariance shoots off positively.

The reason for the oscillating behaviour could be caused by the infrequent GPS observations received, though not exactly at the same period as the period between sensor measurements is not entirely deterministic due to the dither time and handling of other events. The covariance reaches its maximal value when it is furthest from the periodic measurements at both ends. Periodic measurements "smooth" the uncertainties resulting in the oscillation in covariance where information from the future improves information from the past.

As for the very end, where the covariance shoots off, this may be due to not having any further GPS measurements while the run still hasn't terminated, which increases the vehicle's uncertainty both in its position and heading. This resembles what we saw in **Lecture 6C** with the plume end, behaving like a conventional filter.

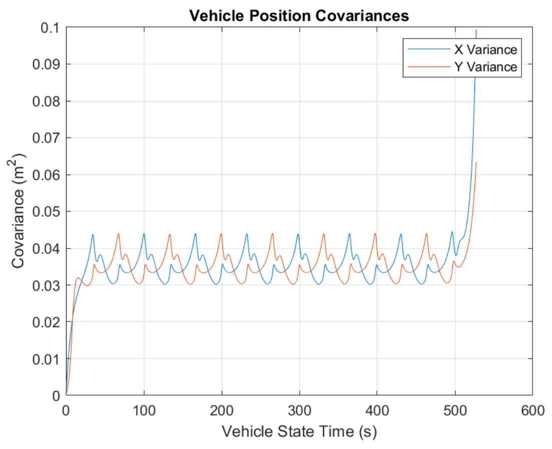


(a) Position Error

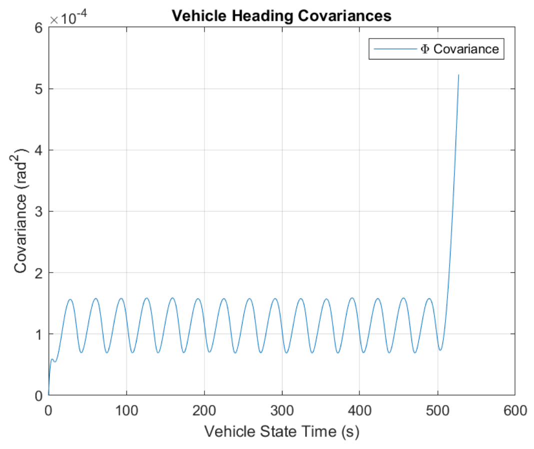


(b) Heading Error

Figure 12: GPS Observation Only Error



(a) Position Covariance



(b) Heading Covariance

Figure 13: GPS Observation Only Covariance

e. i. The optimisation time and Chi-Squared curves have been plotted in Figure 14 below. We can observe a range of optimisation times with an increasing trend as time progresses where there are some sections where it would drop close to 0 optimisation time. The optimisation times reach their maximum as the Vehicle State Time approaches 35 seconds, suggesting that the optimiser takes longer to find the optimal solution as more data is added to the system

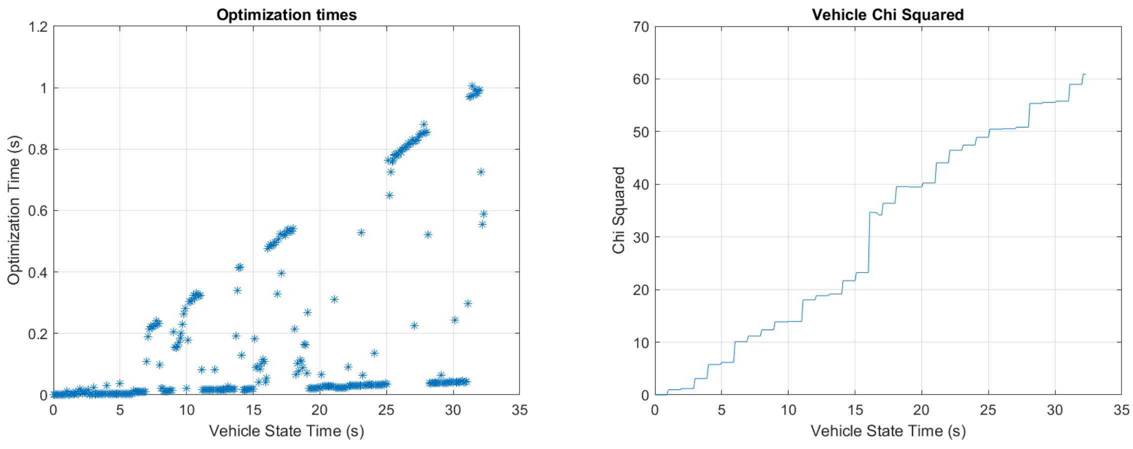
The Chi-Squared curve exhibits a pattern of increasing behaviour, characterized by discrete jumps and moments of ”flattening.” This phenomenon arises due to the fixed time interval between GPS measurements, typically set at 1 second. Upon closer examination of the Chi-Squared plot, it becomes evident that within each 5-second interval of vehicle state time, there are approximately

CONTINUED

5 discrete jumps. However, this pattern may not always hold true, as there exists a predefined dither time between successive GPS measurements.

The occurrence of these jumps can be attributed to the introduction of new GPS measurements, which create disparities between the estimated state derived from the process model and the observed values from GPS. These disparities, known as residuals, are minimized through least-squares optimization. Nevertheless, the estimated state derived from the process model may gradually deviate from the true state of the vehicle over time. Since GPS measurements provide the absolute position of the true state, albeit with some inherent random noise, any discrepancies result in an increase in the residual/Chi-Squared value, leading to these sudden jumps.

Upon examining both plots, similarities become apparent: when the optimization time is minimal, it aligns with segments in the Chi-Squared plot where the Chi-Squared value stabilizes or ”flattens.” This correlation is logical, considering that no additional GPS measurements are acquired during these intervals. Instead, the graph expands solely through kinematic edges and the addition of new state vertices, facilitating faster optimization processes.



(a) Optimisation Time

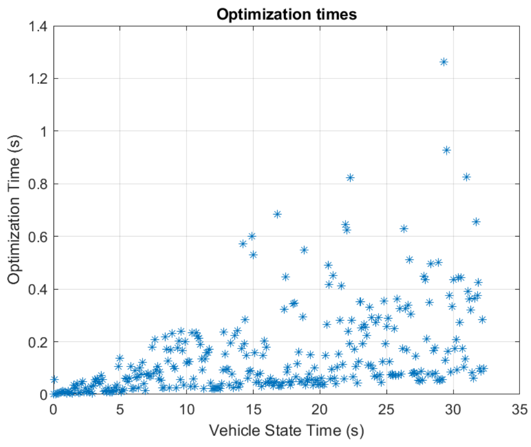
(b) Chi-Squared

Figure 14: Using GPS Only

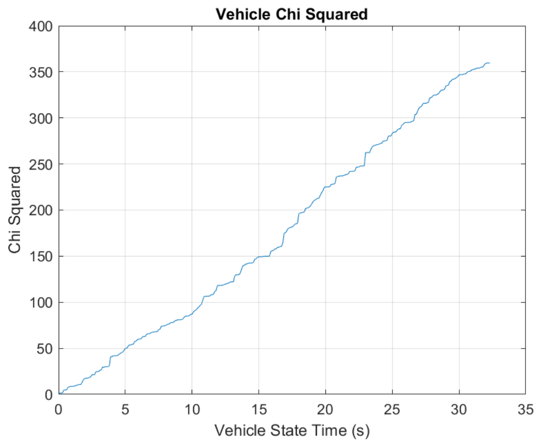
ii. Enabling the compass, we obtain a new set of behaviours of optimisation time and chi-squared, as displayed in Figure 15 below.

The behaviour depicted in the chi-squared plot exhibits a more gradual ascent and a broader span compared to previous observations. This smoother trend arises from the compass measurements occurring at a frequency ten times higher than that of GPS measurements. Consequently, both sensors provide measurements more frequently, reducing the duration of the ”flattened” sections during which no measurements are taken. Moreover, the incorporation of additional sensors contributes to the expanded range observed. Although compasses offer accuracy within short time intervals, over time, they may deviate from the true state, contributing to the overall range observed in the plot.

The optimisation times vary over a larger range, compared to Figure 14 and there are less points lying near 0 seconds for optimisation time. This may be due to the previous discussion about the resemblance of optimisation time and ”flattened” area in Chi-Squared plots. As explained previously, due to enabling both sensors, the ”flattened” areas have shrunk, thus there would be less optimization done with sole kinematics edges, and the overall optimization time increased.



(a) Optimisation Time



(b) Chi-Squared

Figure 15: Using GPS and Compass

Question 2

2. a. A diagram of the structure of the factor graph is displayed below:

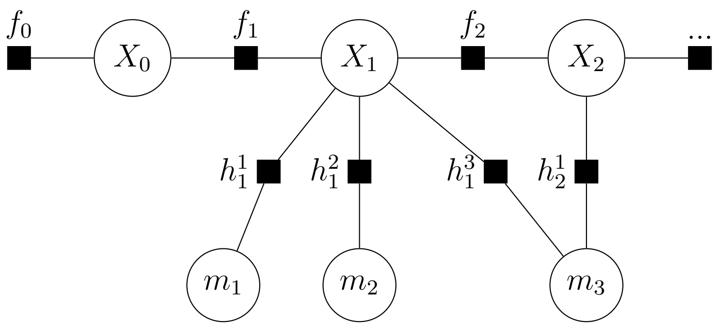


Figure 16: Factor Graph to Predict and Update Robot Pose Using Landmark Observations.

Similar as in **Question 1A** - X_n represent the vehicle state vertices describing the robot pose at each timestep, f_n, representing the vehicle odometry and f_0 represents the prior. These all have the same definition as stated in the previous question.

In addition to this we have the new vertex m_n, which represent the landmark state vertices and new edge h_n^i, representing the Landmark Range Bearing edges (where n is the index of the vehicle state vertex and i is the index of the edge). They have formulation stated below.

Similar to what was stated in **Question 1A**, in the current SLAM context, we now have the heading component ψ and landmark measurment bearing component that spans across the non-linear Euclieand rotation space SO(2), thus they must also be handled with the $\boxplus$ operator for mapping if local variations to a manifold.

Here the $\boxplus$ operator is defined same as before $\boxplus : \texttt{Dom}(\mathbf{x}_i) \times \texttt{Dom}(\boldsymbol{\Delta}\mathbf{x}_i) \rightarrow \texttt{Dom}(\mathbf{x}_i)$, is important in dealing with angular discontinuities, as it enforces the domain of $[-\pi, \pi]$ on any angular measurements or operations. This discontinuity is particularly important for our solution in this case as which may cause our state and landmark estimates to deviate away from the true state for process and landmark observation edges.

Landmark State Vertex m_n:

- The Landmark State Vertex is given by:

$$\mathbf{m}_n = \begin{bmatrix} x_n \\ y_n \end{bmatrix} \tag{17}$$

- Each landmark is initialised in the `LandmarkRangeBearingEdge` class, where it takes the range, azimuth and elevation measurements of the landmark relative to the platform frame, to compute the position of the landmark using the inverse observation model equations below:

$$x_i = x_k + r\cos(\beta + \phi_k) \tag{18}$$

$$y_i = y_k + r\sin(\beta + \phi_k) \tag{19}$$

Landmark Range Bearing Edge h_n^i:

- `computeError`:

$$w_k^L = z_k^L - \begin{bmatrix} r_k^i \\ \beta_k^i \end{bmatrix} \tag{20}$$

  Where $w_k^L(2)$ has to be handled as follow:

$$w_k^L(2) = \texttt{g2o.stuff.normalize\_theta}(w_k^L(2)) \tag{21}$$

- `linearizeOplus`: We define the following variables

$$\Delta x = x_{k+1} - x_k \tag{22}$$

$$\Delta y = y_{k+1} - y_k \tag{23}$$

$$r = \sqrt{\Delta x^2 + \Delta y^2} \tag{24}$$

and computing the Jacobian matrix of error with respect to the vehicle state x_k, we obtain:

$$J_1 = \begin{bmatrix} -\frac{\Delta x}{r} & -\frac{\Delta y}{r} & 0 \\ \frac{\Delta y}{r^2} & -\frac{\Delta x}{r^2} & -1 \end{bmatrix} \tag{25}$$

$$J_2 = -\begin{bmatrix} -\frac{\Delta x}{r} & -\frac{\Delta y}{r} \\ \frac{\Delta y}{r^2} & -\frac{\Delta x}{r^2} \end{bmatrix} \tag{26}$$

b. i. Functions modified for this question are the `handleLandmarkObservationEvent` method of `DriveBotSLAM` and `initialize`, `computeError` and `linearizeOplus` methods of `LandmarkRangeBearingEdge`.

ii. We observe in Figure 17 that overall the optimization time increases due to having a larger and larger graph as the run progresses and more landmark observation edges attached. We also observed that there are occasional drops in optimization times. This drop is most likely due to not having any landmarks in the line of sight of the vehicle, which significantly reduced the rate at which the graph increases and that no new landmark observation edges would be added, thus resulting in a sudden drop in optimization time. However, we would still observe that the optimization time for these sudden drops also increases as time progresses as the graph is still getting bigger.

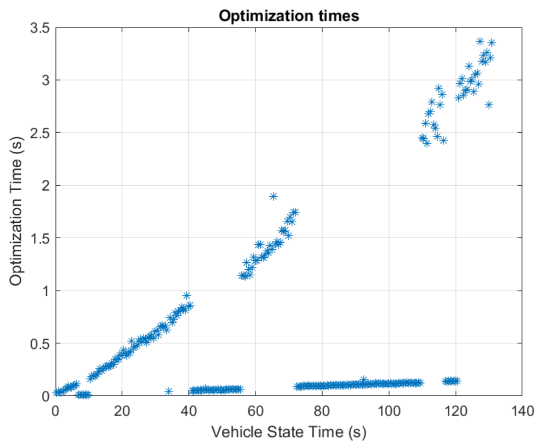


Figure 17: Optimizing Time for running SLAM on single closed loop

CONTINUED

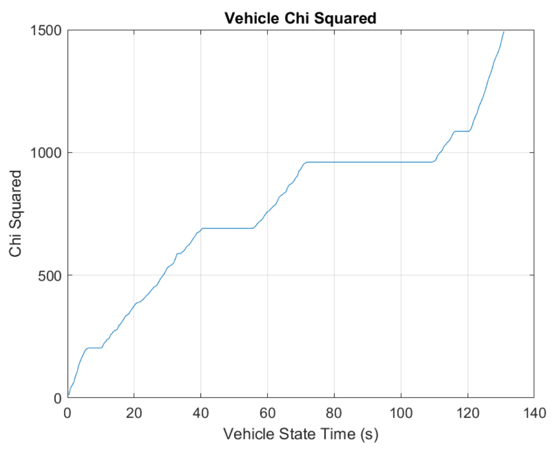
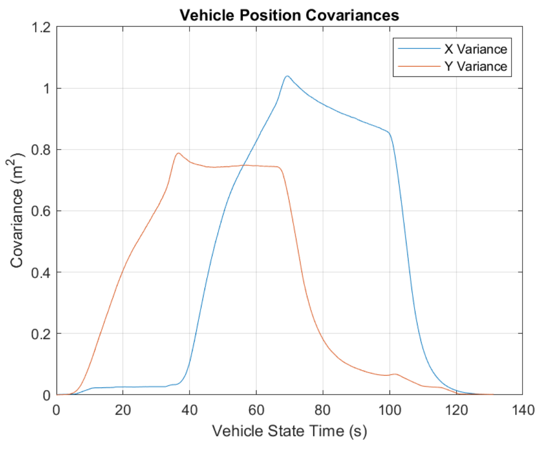


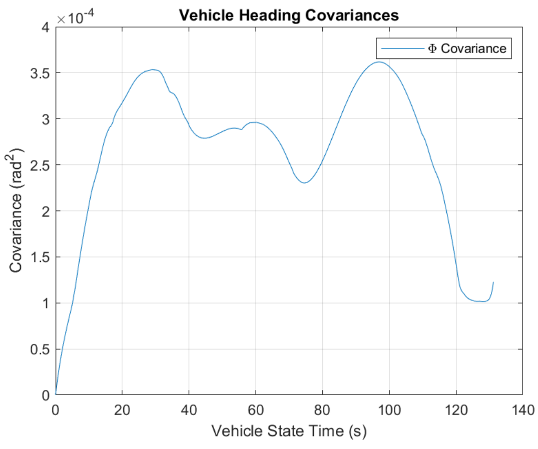
Figure 18: Chi-Squared for running SLAM on single closed loop

Regarding the Chi-Squared plot (Figure 18), the initial observation reveals a continuous upward trend, interrupted by periods of flattening precisely when the optimization time decreases. This reduction typically occurs when the landmark falls out of the vehicle's line of sight, leading to a decrease in optimization efforts. Consequently, the flattening of the plot compensates for the drop in optimization time, reflecting the absence of new landmark observations during these intervals.

This behaviour aligns with our discussion in **Question 1B**, where the system operates solely based on the vehicle's process model when no new landmark observations are available. Consequently, during these periods of flattening, there is no increase in residual, as there are no new observations to compare against the predicted state. However, the residual also doesn't decrease during these intervals, as there are no new observations to refine the estimated state's alignment with previously observed landmark information. Thus, the flattening of the Chi-Squared plot serves as a reflection of the system's reliance on internal state propagation in the absence of external observations.



(a) Position Covariance



(b) Heading Covariance

Figure 19: Covariance for running SLAM on single closed loop

As for the covariance curves (Figure 19), a similar shape is seen for the x and y variances, with some minor differences. We note down the following key time steps $K(i)$ for the covariance analysis:

- $K(1) \approx 36s$ (Vehicle's first anticlockwise turn)
- $K(2) \approx 70s$ (Vehicle's second anticlockwise turn)
- $K(3) \approx 100s$ (Vehicle's third anticlockwise turn)

Concerning the variance in the x direction, it starts off relatively low but escalates swiftly at $K(1)$, experiences a minor dip at $K(2)$, and then rapidly declines at $K(3)$. As for the variance in the y, it initially undergoes a rapid increase, followed by a slight decrease at $K(1)$, and then a swift drop from $K(2)$ to $K(3)$, after which it gradually decreases further.

The initial behaviour at $K(1)$ can be explained similarly as in **Question 1B**, where the vehicle accelerating along one axis would reduce variance in that axis and would cause an increase in variance in the perpendicular axis. The initial gradual decrease in y variance at $K(1)$ after the rapid increase is due to the vehicle's first turn. The rapid decrease in y and the initial gradual decrease in x variance at $K(2)$ is caused by observing the top right corner landmark while performing the second turn. The sharp decline in x variance and the subsequent gradual decrease in y variance at $K(3)$ are attributed to loop closure. During loop closure, the vehicle re-observes the initial landmarks, providing a high level of certainty about its position, especially as these landmarks were observed when the vehicle was initially at the origin. This reaffirms the vehicle's confidence in its position, leading to the observed reduction in variance.

Regarding the heading covariance, we observe an initial increase in variance up to $K(1)$, followed by a slight decrease at $K(2)$. Subsequently, from $K(2)$ onwards, the variance gradually decreases again. However, be-

CONTINUED

yond $K(3)$, we witness a subsequent increase in variance, rapidly followed by a sharp decline, with a minor upward deviation towards the conclusion.

In contrast to the situation described in Question 1B, where the estimation of vehicle state relies solely on vehicle kinematics edges, in this case, the covariance in heading does not increase when the vehicle turns at $K(1)$ and $K(2)$. This is because the vehicle can observe landmarks during the turn. Although the vehicle's confidence in these landmark observations decreases further into the run, the range and bearing information still provide some insight into its heading, thereby reducing variance in heading.

The increase in variance from $K(2)$ onwards occurs because, after the second turn, there are no landmarks within the sensing range of the vehicle, leading to increased uncertainty in its heading. Similarly to the covariance in position, the decrease from $K(3)$ onwards is attributed to loop closure, as the vehicle observes landmarks near the origin, allowing it to derive its heading more confidently from the range and bearing information of these landmarks. Finally, the sharp increase in variance towards the end may be attributed to the vehicle going out of range in observing the landmark around $(3, 28)$, causing a slight loss in certainty in its heading, although this effect is mitigated as the vehicle can still observe landmarks near the origin.

It is noteworthy that the variance in heading is at the scale of 10^{-4} and position at the scale of 10^{0}, which is rather low even relative to the scale of its measurement units, thus overall the vehicle is rather certain about its position and heading.

c. i. Using MATLAB's `isa` function, the total number of `VehicleStateVertex` and `LandmarkStateVertex` instances within the constructed graph were extracted. Then, using a similar approach, the total number of `LandmarkRangeBearingEdge` instances were extracted.

   Using this, the number of vehicle poses stored (i.e. the number of `VehicleStateVertex` stored in the graph), the number of landmarks initialised (i.e. the number of `LandmarkStateVertex` stored in the graph), average number of observations made by a robot at each time step (i.e. the average number of `LandmarkRangeBearingEdge` per `VehicleStateVertex`), and the average number of observations received per landmark (i.e. the average number of `LandmarkRangeBearingEdge` per `LandmarkStateVertex`) were determined.

   ii. The results from the above computations are presented in Table 1 below:

Property	Quantity
Number of Poses	5273
Number of Landmarks	7
Average Number of Observations per Timestep	0.6118
Average Number of Observations Received by Landmark	460.9

Table 1: Properties of the Constructed Graph.

The results displayed above imply that the constructed graph is very densely connected - each landmark receives an average of approximately 461 observations which can indicate that a large portion of the data is redundant in terms of accurately determining the state of the vehicle. Given the 5273 poses of the vehicle and 7 landmarks stored in the graph, this suggests that several simplification approaches can be implemented to reduce the storage requirements of the system whilst maintaining a similar level of accuracy.

Another point to note is that the average number of observations made by the vehicle per timestep is less than 1 - that is, not every timestep has an associated observation. This implies that there are many poses stored in the graph that do not contribute to the accuracy of the solution. Since the vehicle moves mostly in straight lines, many stored poses are redundant and could be eliminated from the graph.

d. Right before loop closure (See Figure 20), landmarks in proximity to the vehicle's starting point exhibit comparatively low covariance, as illustrated by the smaller ellipse. The diminished covariance of landmark locations earlier in the trajectory is attributed to the vehicle's heightened confidence in its location estimate, particularly when it is closer to its origin. This confidence stems from the vehicle's certainty about its initial position, leading to increased confidence in landmark locations, as determined by the Landmark Observation Model based on the vehicle's state.

   Conversely, landmarks situated farther along the trajectory demonstrate increasing covariance. Additionally, a substantial covariance in the vehicle's position estimate is evident. This is expected as there would be an increased uncertainty in its position as it progresses along the trajectory, being further away from its origin. Consequently, this leads to heightened uncertainty in the locations of the landmarks since the vehicle has encountered each landmark only once.

   After the loop closure (See Figure 21), we noticed that the covariance in vehicle position shrunk drastically as a result of loop closure, this is because the vehicle has observed the landmarks very close to the origin once again, asserting the fact that the vehicle has returned to the origin, thus gaining confidence in the vehicle's state

   Consequently, the landmark covariance shrunk overall. However, those earlier

CONTINUED

in the trajectory don't have as much of a change compared to the ones further along the trajectory, though there is also a slight decrease in covariance shown by the ellipse reduction in size. This is expected as the vehicle is already rather certain with these earlier landmark positions. Another interesting observation is that the landmark located at around $(3, 28)$ has shrunk drastically compared to the landmark located at the top right corner of the trajectory. This is because this landmark is more recently observed before observing the landmarks near the origin, which allows this landmark to be smoothed, demonstrating the smoothing behaviour of SLAM systems.

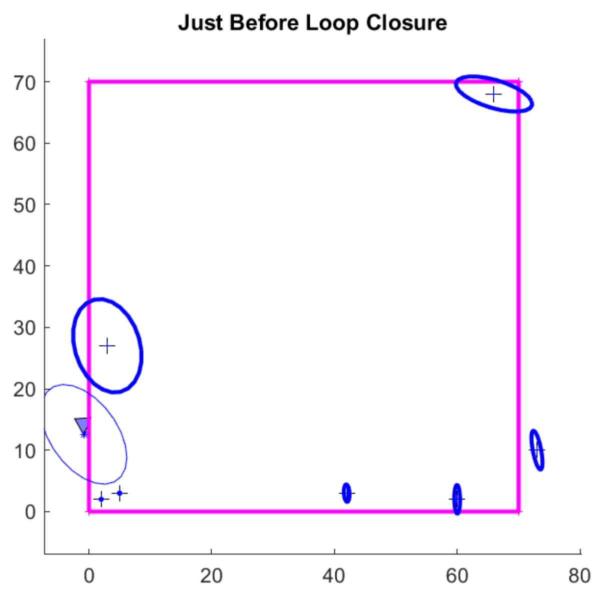


Figure 20: Before Loop Closure

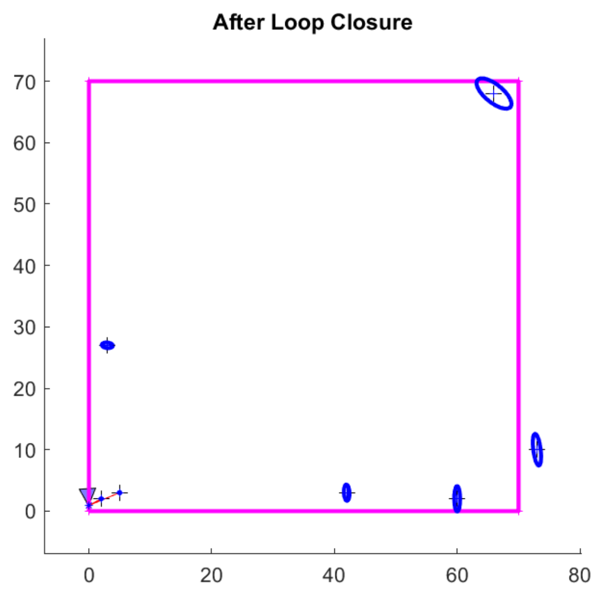


Figure 21: After Loop Closure

Uncertainty is calculated by looking at the square root of the determinant of the covariance of each landmark, we computed the amount by which the uncertainty has changed for each landmark. As the covariance may vary for different runs due to random noise, we set a random seed and obtained the following to 4 significant figures:

Landmark Index	**Uncertainty Before**	**Uncertainty After**	**Change**
1	1.010×10^{-3}	9.792×10^{-4}	3.132×10^{-5}
2	1.339×10^{-3}	1.291×10^{-3}	4.806×10^{-5}
3	7.227×10^{-2}	6.682×10^{-2}	5.454×10^{-3}
4	0.1237	0.1095	1.423×10^{-2}
5	0.2433	0.1736	6.971×10^{-2}
6	1.636	0.5693	1.066
7	4.606	4.541×10^{-2}	4.561

Table 2: Uncertainty Before and After Loop Closure

As stated previously, uncertainty for all landmarks decreased after loop closure, where significant decrease in uncertainty in the 7th landmark, which is the landmark observed just before loop closure located at $(3, 28)$.

Question 3

3. a. i. Here the SLAM system relies purely on Landmark Observations. This might be a sensible strategy as the landmark vertices essentially become

the intermediary node between state vertices. This not only greatly reduced the number of edges (as the prediction edges are all removed) but also vehicle state vertices that are not linked via the landmark vertices may not be considered as part of the optimization, which may speed up optimization. It is likely to be successful when we have dense landmarks and when the motion of the vehicle is regular and predictable, and it is likely to fail when the landmarks are sparse or when the initial node (that the prior edge is attached to) has no connection with the rest of the graph and also when motion of the vehicle is irregular and difficult to predict and model.

ii. The function modified for this question is the `deleteVehiclePredictionEdges` method of the `DriveBotSLAMSystem` class.

iii. Removing all prediction edges fails and Removing all but the first edge works. The reason why removing all prediction edges fails might be because by convention of the code, no Landmark Vertices are attached to the initial node via the Landmark Observation edges, and the Landmark vertices only start to attach to the state vertices after the initial state vertex (See Figure 22). Thus with all process edges removed, the the initial vehicle state node and the prior edge that is attached to it would have no connection with any of the state vertices within the graph (See Figure 23).

The prior acts as a soft constraint or anchor for these optimization processes, giving the initial conditions on where the vehicle could be. Without the prior, the optimization could not be done. As a result of this, it would result in very wrongly calculated Hessian in the `computeHB` method in **SparseOptimizer** class, which propagates to error in the computation of the covariance in the `computeMarginals` method and messes up the optimization process.

We verified that this was the source of the error by removing only the first process edge and keeping the rest of the process edges, which produced the same error.

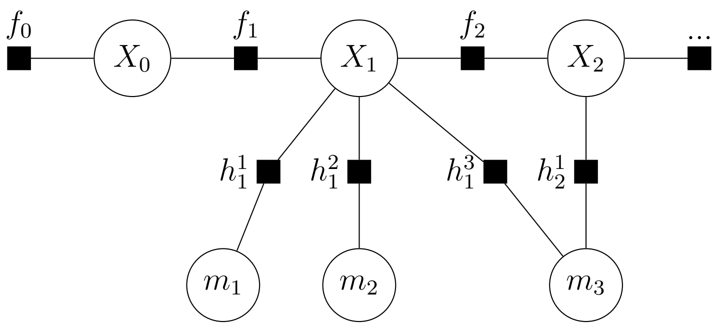


Figure 22: Subsection of a Graphical Model of the SLAM system

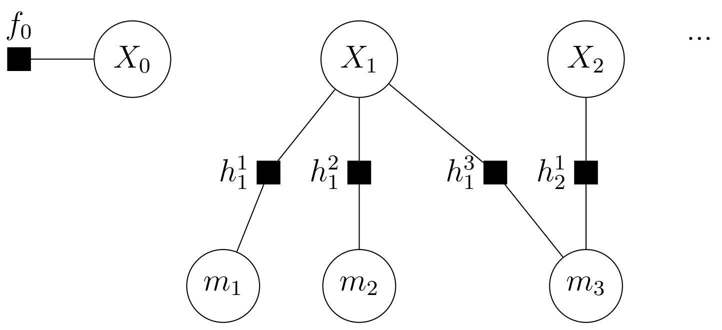


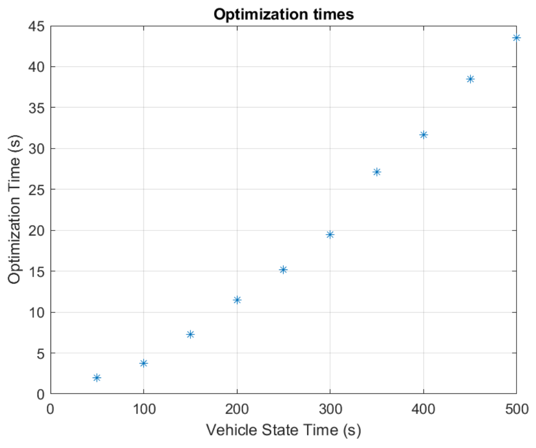
Figure 23: Subsection of a Graphical Model With all Process Edge Removed

iv. In terms of optimization time (Figure 24) we would see improvement where at all vehicle state time steps, the edge removal method would have less optimization time than the full SLAM system. We would also observe significant improvement in terms of Chi-Squared (Figure 25) where once again the edge removal method results in lower Chi-Squared at all time steps compared with the full SLAM system.

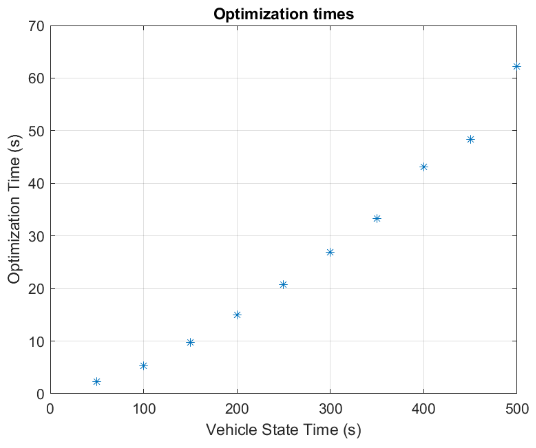
    However, if we look at the covariance plots (Figure 26 27), we would observe the opposite, where the covariance for the edge removal method is almost ten times greater than the full SLAM and also has immense variability from time step to time step.

    Ignoring covariance, this method for removing edges undoubtedly proves successful, given its faster optimization and resulting lower residuals. However, when considering covariance, the determination of its success hinges on the specific task at hand. For instance, if applied to robotic platforms with limited computational resources prioritizing rapid SLAM implementation over stringent uncertainty control, this method would excel. Conversely, if ensuring high certainty in the platform's state throughout operation takes precedence over computational efficiency and optimization speed, opting for the complete SLAM system might be preferable.

CONTINUED

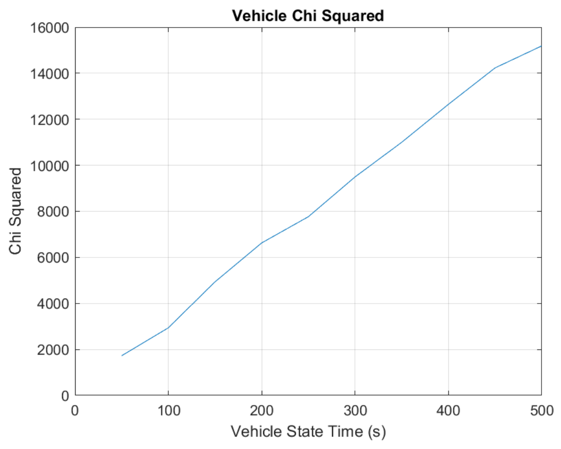


(a) With Edge Removal

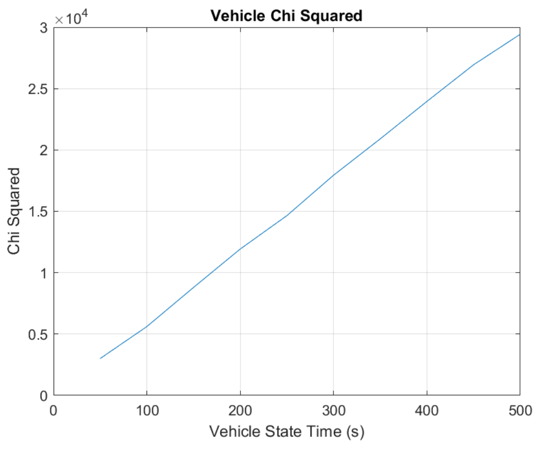


(b) Full SLAM System

Figure 24: Optimization Time Comparison between Edge Removal and Full SLAM

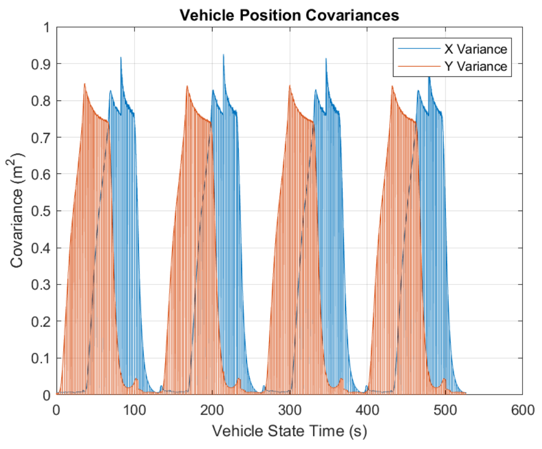


(a) With Edge Removal

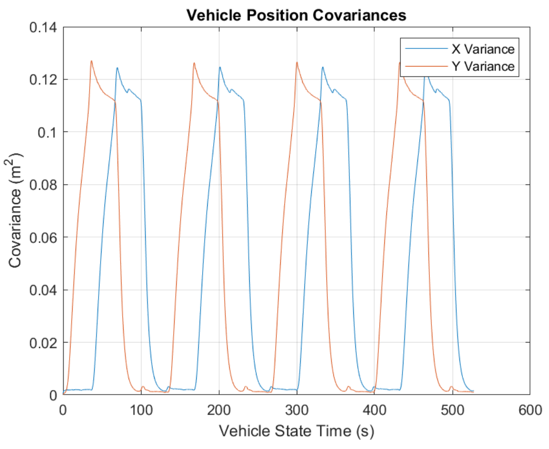


(b) Full SLAM System

Figure 25: Chi-Squared Comparison between Edge Removal and Full SLAM

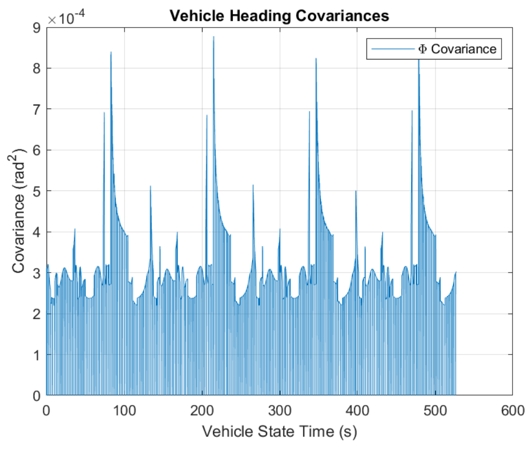


(a) With Edge Removal

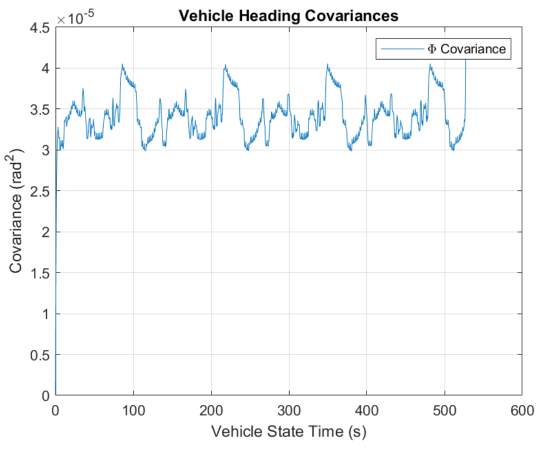


(b) Full SLAM System

Figure 26: Position Covariance Comparison between Edge Removal and Full SLAM



(a) With Edge Removal



(b) Full SLAM System

Figure 27: Heading Covariance Comparison between Edge Removal and Full SLAM

b. i. Graph pruning is a graph simplification approach to reduce the large storage requirements of a SLAM system. By pruning redundant edges and vertices, the solution maintains a similar level of accuracy whilst drastically reducing the size of the constructed graph.

One heuristic of graph pruning is based on density [2]. Given the large number of observations received by each landmark, many observations are in fact redundant and do not contribute significantly to the accuracy of the solution. Therefore, by removing edges and vertices with high density, i.e. where the graph is more likely to have stored redundant information, we can keep the size of the graph lower whilst maintaining an acceptable level of accuracy. Pruning based on density also allows for the algorithm to run

CONTINUED

without having to examine the information associated with each vertex, saving computational cost.

ii. The implementation of the graph pruning algorithm above involved the modification of the **DriveBotSLAMSystem** class. The method `pruneGraph` was implemented, along with the method `setPruneOn` to enable graph pruning only when requested.

    The density of each vertex was measured using the scale-invariant density to ensure a more even spread in the graph after pruning [2]. The scale-invariant density, denoted as SID hereafter, was computed for each vertex as below:

$$d\left(v_i\right) = \frac{1}{\pi} \sum_{k \in \{1,\ldots,n\} \setminus \{i\}} \left\| v_k - v_i \right\|^{-1} \tag{27}$$

    where v_n is the position of the n-th state vertex (both landmark and vehicle).

    To save computational cost, the SID for each vertex was computed taking into account its distance from only its 20 nearest neighbours. Care was taken to avoid removing information from when the vehicle was making a turn, so as to retain solution accuracy at points where the vehicle dynamics are changing rapidly. Once the SID of all vertices were obtained, the maximum of these were found, and if this corresponded to a vehicle state vertex then the associated landmark observation edges were pruned. The reason for removing the landmark range bearing edges instead of the vertices was to maintain the stability of the solution without requiring vertex marginalisation.

    The implementation was tested using the same configuration as in **Question 3a**, to compare the performance of the SLAM System after graph pruning with that after process edge removal.

iii. The optimisation times for the simulation with graph pruning have been plotted and displayed in Figure 28. The optimisation times are almost identical to the edge removal case of **Question 3a**, confirming that graph pruning has significantly reduced the complexity of the solution to the point that its optimisation times are as low as the case with no process model edges.

    The Chi-Squared curve in Figure 29 displays an interesting behaviour, where it steadily increases for 300 seconds then drops. The values are much lower than both the edge removal and full SLAM System case in **Question 3a**, perhaps due to its lower residuals than in the case of edge removal, but less edges in the graph than in the full SLAM case.

    The position and heading covariance curves in Figure 30 display similar behaviour as that of the full SLAM System in Figures 26 and 27, albeit with a higher range. The covariances are consistently lower than that of the SLAM System with edge removal, but slightly higher than the full system, indicating that pruning successfully retained an acceptable level of accuracy within the solution.

Overall, the graph pruning implementation successfully demonstrated the potential of the algorithm. The results of this simple implementation suggest with confidence that a more complex pruning method will yield a comparable level of accuracy whilst reducing optimisation times.

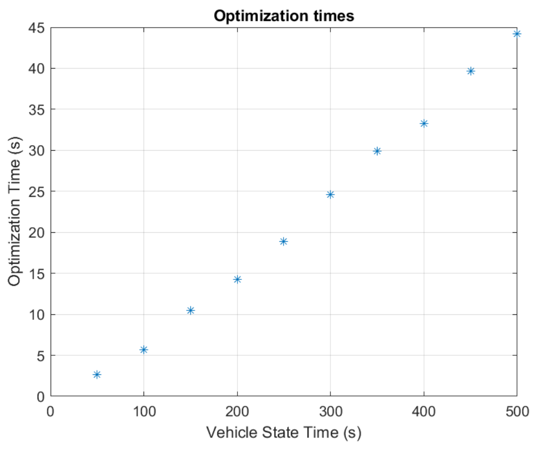


Figure 28: Optimization Time with Graph Pruning

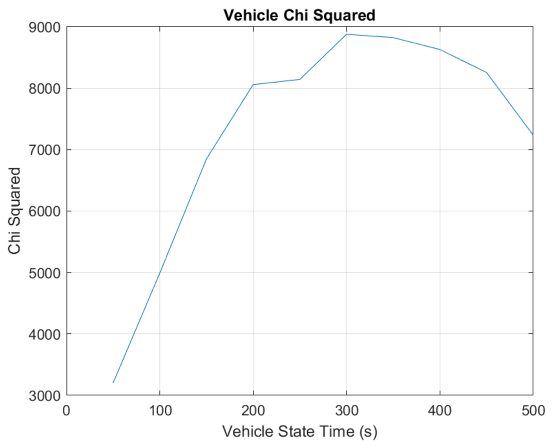
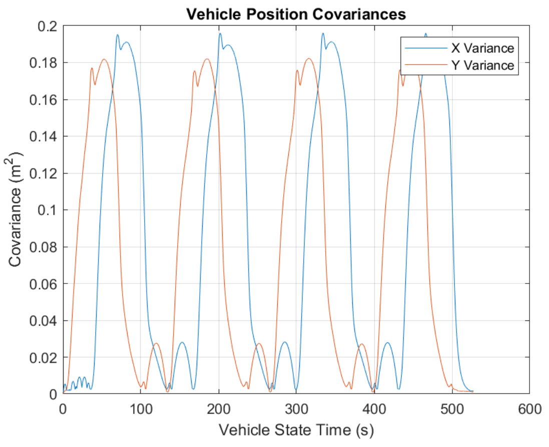
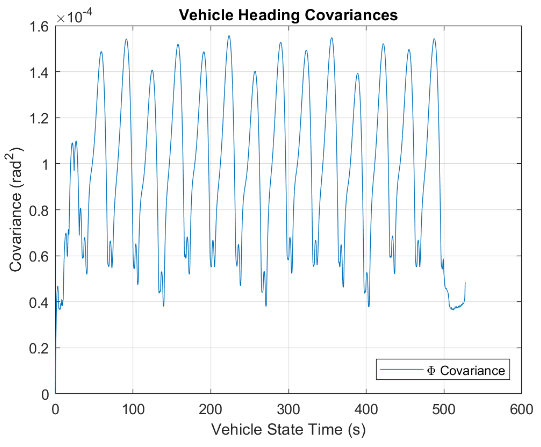


Figure 29: Chi-Squared with Graph Pruning

CONTINUED



(a) Position Covariance



(b) Heading Covariance

Figure 30: Covariance with Graph Pruning

References

[1] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011. doi: 10.1109/ICRA.2011.5979949.

[2] Gerhard Kurz, Matthias Holoch, and Peter Biber. Geometry-based graph pruning for lifelong slam, 2021.

END OF COURSEWORK