Student Number: 5973058

Student Name: Gabriella Low Chew Tung

Degree Programme: Master of Science Engineering and Policy Analysis

Faculty: Faculty of Technology, Policy and Management

# Computational Reproducibility in Modelling and Simulation Studies

A Case Study on Data Assimilation Algorithms for Agent-based Modelling and Simulation

Graduation Committee

| | |
|---|---|
| Chair | Prof.dr.ir. Alexander Verbraeck |
| First Supervisor | Dr. Yilin Huang |

# Table of Contents

Executive Summary ..... 2
I. Introduction ..... 3
1.1. Reproducibility Research ..... 3
1.1.1 Data Assimilation for Agent-based Modelling and Simulation ..... 3
1.1.2 Knowledge gap ..... 4
1.1.3 Contributions ..... 5
1.1.4 Relevance to degree programme ..... 5
1.1.5 Thesis Structure ..... 6
II. Research Design ..... 7
1.2 Research Questions ..... 7
1.2.1 Main Research Question ..... 7
1.2.2 Sub-questions ..... 7
1.3 Research Design Flow Diagram ..... 8
1.3.1 Reproducibility Experiment ..... 9
1.3.2 Particle Filter Sensitivity Analysis ..... 9
1.3.3 Hardware and Software implementation Specification ..... 10
1.3.4 Data Sources and Availability ..... 10
III. Reproducibility and Replicability ..... 11
1.4 Why conduct reproducibility and replicability research in M&S? ..... 11
1.5 Benefits of Reproducible Research ..... 11
1.6 Factors contributing to the Reproducibility Crisis ..... 12
1.7 What solutions have been proposed? ..... 12
1.7.1 The Code Review Process ..... 13
1.8 How to assess a successful reproduction/replication success? ..... 13
1.9 Reproducibility Frameworks ..... 14
IV. Data Assimilation for agent-based models ..... 16
1.10 Data Assimilation Background ..... 16
1.10.1 Recursive Bayesian Estimation Problem Formulation ..... 16
1.11 Particle Filter ..... 18
1.11.1 General Particle Filter Algorithm – SIR Filter ..... 18
1.11.2 Particle Filter for ABMS ..... 19
V. Reproducibility Assessment ..... 21
1.12 Case Study Selection ..... 21
1.13 Case Study Overview ..... 21
1.13.1 Case Study Methodology ..... 22

1.13.2 Case Study Experimental Setup ..... 23
1.14 Reproducibility Experimental Setup ..... 23
1.14.1 Method – Reproducing Figures ..... 24
1.14.2 Method – Reproducing Tabulated Results ..... 24
1.14.3 Computational Workflow of Case Study Code Repository ..... 26
1.15 Results ..... 27
1.15.1 Reproduction of Figures of Bus Trajectories ..... 27
1.15.2 Reproduction of Table 3: Sensitivity analysis of maxDemand and dynamic change rate 33
1.16 Discussion ..... 36
1.16.1 Challenges ..... 36
1.16.2 Recommendations ..... 38
VI. Particle Filter Sensitivity Analysis ..... 40
1.17 Problem Formulation ..... 40
1.18 Particle Filter Parameters ..... 43
1.19 Experimental Setup ..... 43
1.20 Particle Filter Sensitivity Analysis Results ..... 44
1.20.1 Experiment 1 – Varying Number of Particles and Jitter and Resample Window = 1 45
1.20.2 Experiment 2 – Varying Number of Particles and Resample Window; Jitter = 0.0005 48
1.20.3 Discussion ..... 51
VII. Final Discussion ..... 53
1.21 Challenges for reproduction attempt ..... 53
1.22 Parameter Specifications for Particle Filter ..... 54
1.23 Guidelines for reproduction process ..... 59
VIII. Conclusion ..... 62
1.24 Limitations ..... 63
1.25 Recommendations/Future work ..... 63
1.26 Reflection ..... 63
References ..... 65
Appendix A: Experiment Log ..... 69
1.27 Experiment 1 ..... 70
1.27.1 Figure 5 ..... 70
1.27.2 Figure 6 ..... 72
1.27.3 Figure 7 ..... 75
1.27.4 Figure 8 ..... 77
1.27.5 Figure 9 ..... 80

1.28 Sensitivity Analysis of maxDemand and dynamic change rate $\xi$ ........ 83
1.28.1 Errors ........ 85
Appendix B: Reproducibility Experiments of Table 3 ........ 87
1.29 Experiment 1-Reproducing Table 3 using the original data files ........ 87
1.30 Experiment 2 – Reproducing Table 3 using reproduced data ........ 89
1.30.1 Experiment 2A – Using models calibrated by BusSim-truth for IncreaseRate and using models calibrated by BusSim-stochastic for maxDemand ........ 89
1.30.2 Experiment 2B – Using models calibrated to BusSim-truth ........ 90
Appendix C: Error Logs ........ 91
1.31 Test 1 – Preliminary execution ........ 91
1.32 Test 2 – Check if codebase runs with the provided data files ........ 103
Appendix D: Particle Filter Sensitivity Analysis Supporting Material ........ 105
1.33 Particle Filter Algorithm According to Publication and Code ........ 105

# Executive Summary

In recent times, the *'reproducibility crisis'* has become a cause for concern in the scientific community. Many disciplines from psychology and neuroscience to machine learning and ecology have been promoting initiatives towards ensuring the replicability and reproducibility of the findings reported in research publications. However, ensuring reproducibility and replicability of research faces many challenges such as differing definitions of the terms across disciplines, lack of incentives towards reproducing/replicating already published work and no standard methods for assessing a successful reproduction or replication. Reproducible research is fundamental to the scientific process, helps to ensure the credibility of scientific research and facilitates the dissemination and advancement of scientific knowledge. This crisis is especially relevant in the field of Modelling and Simulation and other computational sciences which rely on computer simulations to support findings yet there is a dearth of adequately detailed documentation to facilitate successful reproductions.

This project focussed on investigating the computational reproducibility of a research publication in the field of data assimilation for agent-based simulations. Agent-Based Modelling and Simulation (ABMS) is a computational method frequently employed to study complex socio-technical systems. Data assimilation techniques for ABMS is an emerging research area that seeks to incorporate real-time data into the model to improve its predictive capabilities. However, due to its novelty, reproducibility studies of these experiments are lacking. As this is a young research field, with various new methodologies being published, it is important to support verification and validation processes to advance scientific developments in the field such that the methods can be suitably adopted by applied researchers for future studies.

The main challenges of the reproduction process were identified as code quality and missing dependencies; ambiguous or missing specifications regarding the methodology and inconsistencies between textual descriptions and implemented code. Evaluation of reproducibility was also considered from the perspective of statistical metrics on one hand and qualitative reproducibility frameworks on the other hand. Furthermore, the experiment also highlighted the importance of computational provenance to connect the published results to the code or software used to generate them.

A series of practical steps to guide the workflow of future reproduction studies was drafted along with guiding questions to deduce computational workflows from publications and their code repositories when workflows to produce published results are missing.

A sensitivity analysis was employed to examine the influence of filter parameters including the number of particles, the resampling window, and the jitter standard deviation on the data assimilation algorithm's estimation accuracy to verify the implementation and reproducibility of the particle filter algorithm used in the case study. From this experiment and based on literature, key elements that should be specified in future data assimilation for ABMS studies to ensure reproducibility of the research were identified.

In summary, this thesis project addressed the research gap in data assimilation for ABMS by conducting a reproducibility study of a research publication employing the Particle Filter technique. Key results from the original publication were reproduced and the original and reproduced results were compared. A reproducibility protocol was formulated to guide researchers in future reproducibility studies and with respect to data assimilation for agent-based simulations, a list of key parameters and considerations that should be reported for studies applying the particle filter to ABMS was devised.

# I. Introduction

## 1.1. Reproducibility Research

The use of simulation studies and computational science has become increasingly common across many disciplines. However, there remains the ever-present need to ensure that the results of computational experiments uphold reproducibility which is considered a core principle of the scientific process. Reproducibility is of great importance as evidenced by concerns in recent years of the 'Reproducibility crisis' in science (Baker, 2016).

Taylor et al. (2015) discussed reproducibility in research as one of the grand challenges in the field of Modelling and Simulation (M&S). It is less well-studied in M&S with works such as Fitzpatrick (2019) discussing the difficulty of reproducibility in stochastic simulations, particularly agent-based simulations. The failure of many simulation studies to ensure reproducibility has contributed to the concerns about credibility in the simulation field(Dalle, 2012). The results of these simulations are used to inform decision-making and for forecasting. Yet the situation is that without reproducibility, how reliable can the outcomes and insights from these experiments be?

There is no consensus on the definitions of 'reproducibility' and 'replicability' across the scientific disciplines(Gundersen, 2021; Plesser, 2018). The definitions of these terms differ on whether the original author's digital artefacts are used to verify the findings as compared to independently creating the necessary digital artefacts based on the research article's texts(National Academies of Sciences & Medicine, 2019). In this thesis project, the definitions used are aligned with the terminology used in National Academies of Sciences and Medicine (2019).

*"**Reproducibility** is obtaining consistent results using the same input data; computational steps, methods, and code; and conditions of analysis."*

*"**Replicability** is obtaining consistent results across studies aimed at answering the same scientific question, each of which has obtained its own data."*

Alternatively, to understand the M&S literature on replicating agent-based models (ABMs), it is also necessary to discuss the term '*replication*' as a verification method used by researchers in computational social science. The following definition of the replication process shall be referred to as 'model replication' in the sections that follow.

***Model replication*** is a researcher's implementation of another researcher's conceptual model which has been previously implemented. The implementation of the replicated model must differ from the original model implementation in some manner such as algorithms, hardware and authors, among others.(Wilensky & Rand, 2007) It is a means of verifying that a given simulation's reported results can be reproduced by someone starting from scratch(Axelrod, 1997).

Due to the different terminologies used across disciplines, it is important to identify the meaning of reproducibility or replicability with regards to a researcher's intention when studying the topic and associated literature.

### 1.1.1 Data Assimilation for Agent-based Modelling and Simulation

ABMS is a type of simulation popularly employed in the social sciences and it consists of the existence of many agents interacting with each with no central direction(Axelrod, 1997). Traditionally, ABMs have been used to comprehend fundamental processes and not to accurately represent a real-world system(Axelrod, 1997).

However, in recent times, there has been an increase in the availability and application of sensor technologies which has generated a plethora of data. Data availability alongside increasing computational power have spurred research in agent-based modelling and simulation (ABMS) towards applications for which it could not be used previously(Ghorbani et al., 2023).

When using ABMS, the results can quickly diverge from the reality after the initial execution of the simulation due to various model uncertainties. To overcome such shortcomings, researchers have looked to incorporate real-time data into these simulation models using data assimilation techniques to improve the forecasting capabilities of ABMS(Malleson et al., 2020). Such models could provide up-to-date information and more accurate short-term forecasts to policymakers allowing for improved situational awareness and decision-making. It could be used as a real-time management tool in fields such as disaster management, emergency evacuation and crowd management, among other applications (Malleson, 2018). This potential application of ABMS represents a shift in the usage of ABMs as solely a tool for exploration and understanding of basic processes to usage of ABMs to more accurately represent real-world systems.

Data Assimilation is an approach for combining prior knowledge of a system in the form of a numerical model with new data (also called measurements or observations). (van Leeuwen, 2015) This is commonly done to calculate the best possible estimate of the model state. Data assimilation practices originated in ocean and weather-prediction communities and have been adopted by many research fields including the geosciences, economics and biology, among others. (Evensen et al., 2022)

For ABMs, conventional data assimilation methods like the Kalman filter and its associated derivations are difficult to apply because these models are usually specified as computer programs, lack analytical form and display elements of non-Gaussianity, high dimensionality and nonlinearity in their systems. For such problems, the particle filter algorithm is often employed(Wang & Hu, 2015). This algorithm can be used to solve complex nonlinear, non-Gaussian on-line (real-time) estimation problems that cannot be solved analytically. Particle filter methods are flexible, easily implemented and generally applicable in different settings but are generally computationally inefficient (Doucet et al., 2001).

### 1.1.2 Knowledge gap

The aim of applying data assimilation methods to ABMS is to realise real-time simulations that employ ABMs and can be used to make improved forecasts of real-world systems. However, there are challenges inherent to achieving computational reproducibility in the area of real-time simulation studies. For example, in the case of real-time distributed simulations, analysis of the simulations can be difficult due to the usually non-deterministic nature of the simulations(McLean & Fujimoto, 2000). Sources of non-determinism include external inputs such as a human operator in human-in-the-loop simulations or lacking complete control of the experimental conditions(Dalle, 2012; McLean & Fujimoto, 2000). Such non-determinism can also occur in future real-time ABMs and thus, should be considered by these researchers.

On the other hand, in the ABMS field, several research papers have addressed the subject of *model replication.* This is a key concern as simulation studies shared via publications are difficult to replicate or even fully understand without detailed specifications of the model itself (Axelrod, 1997). Replicating simulations has been proven a worthwhile process as studies like Axelrod (1997) and Edmonds and Hales (2003) have found that it can reveal bugs and implementation issues which can have either a minor or significant impact on the overall simulation results.

In short, data assimilation for ABMS presents a more complex reproducibility problem in which reproducibility of both the model and the data assimilation algorithm need to be accounted for.

As data assimilation for ABMS is an emerging field, standard methodologies and implementations have yet to be established. Most implementations are domain-dependent and therefore, are formulated for their specific model (Tang & Malleson, 2022). This sentiment was echoed in the critique of Monti et al. (2023) on data assimilation for ABMS research in which it is suggested that successful results were dependent on the use of specific toy models which could be adjusted for better performance.

Current research should be documented and verified to facilitate the reuse and reapplication of their methodologies on alternative problems by future researchers. In order to advance the field, existing research should be reproducible and verifiable yet there exists few reproducibility or replication studies to facilitate this matter. The lack of reproducibility studies and lack of guidelines to ensure reproducibility of published studies constitutes a research gap that inhibits the credibility and reliability of this young field.

With new approaches for data assimilation for ABMS regularly published in the literature, verification of these methodologies through reproducibility studies is key to establishing confidence in them and enabling other researchers to more easily adopt and reuse these methods for their own studies.

In essence, this research project aimed to address this research gap by conducting a reproducibility study of the particle filter approach employed in real-time ABMS.

### 1.1.3 Contributions

Despite the widespread usage of simulation studies, they are often poorly designed, analysed, and reported (Morris et al., 2019). Reporting guidelines can address this matter as they facilitate reproducible research, enhance transparency, improve documentation practices, build trust in methodological research and facilitate knowledge dissemination. (Plavén-Sigray et al., 2017; Williams et al., 2024).

Furthermore, amidst the reproducibility crisis, scientific journals have implemented policies aimed at mitigating the problem while research communities have urged the adoption of reproducible research practices through various initiatives(Eglen & Nüst, 2019). Despite improvements, reproducibility research in computational experiments remains outside of mainstream practice due to its perception as time-consuming work with little payoff(Freire & Chirigati, 2018). To facilitate the execution of future reproducibility studies, a protocol for conducting reproducibility experiments was drafted in this thesis project and insights into the challenges encountered were derived from the experiment.

In short, this project's output included insights into the challenges encountered when conducting a reproducibility study, a protocol formulated to guide similar future reproducibility experiments and lastly, documentation tips for the reporting of relevant parameters for the particle filter when used in data assimilation for ABMS studies to facilitate future reproduction and reuse.

### 1.1.4 Relevance to degree programme

In the MSc Engineering and Policy Analysis programme, M&S is one of the primary learning lines. In the role of policy analysts, agent-based modelling and simulation can be used to model complex socio-technical systems to gain insights and design interventions that can be used to tackle the global grand challenges. Therefore, it is of utmost importance that these M&S studies meet reproducibility and replicability standards such that the insights gained are reliable not only in the eyes of the modeller but to stakeholders and the scientific community. Furthermore, reproducibility of science is crucial for ensuring credibility of scientific claims and discoveries.

Therefore, it is especially important for evidence-based policymaking that makes use of scientific knowledge in decision-making.

### 1.1.5 Thesis Structure

The structure of the thesis is as follows. Chapter II describes the research design and research questions. Chapter III provides background on reproducibility research and Chapter IV details the theory behind data assimilation for agent-based simulation emphasising the particle filter method and its formulation for ABMs. Chapter V discusses the reproducibility experiment giving an overview of the case study, the experimental setup and results of the reproducibility study. Chapter VI conducts a sensitivity analysis on key parameters of the particle filter algorithm used in the case study. Chapter VII summarises the key contributions of this study in the form of a list of challenges to the reproduction process, reporting tips for the particle filter and the reproducibility protocol. Chapter VIII concludes the study and reflects on the results and limitations of this work.

# II.Research Design

This research project's methodology was divided into two sections. The first section focussed on the reproducibility experiment and the second section focussed on the verification of the particle filter. The project made use of a case study i.e. a selected research publication to be reproduced which is detailed in Chapter 1.12. This section first describes the research questions followed by the presentation of the research design flow diagram and corresponding methodology.

## 1.2 Research Questions

### 1.2.1 Main Research Question

***What are the essential elements of data assimilation for ABMS studies applying the particle filter that must be specified to facilitate reproducibility?***

Based on existing reproducibility research, this project used primarily quantitative methods to conduct the reproducibility study. Insights gained were based on both the subjective experience of the researcher and quantitative analysis of the experimental results. Similar approaches were used in Luijken et al. (2024), Zhang and Robinson (2021) and Edmonds and Hales (2003).

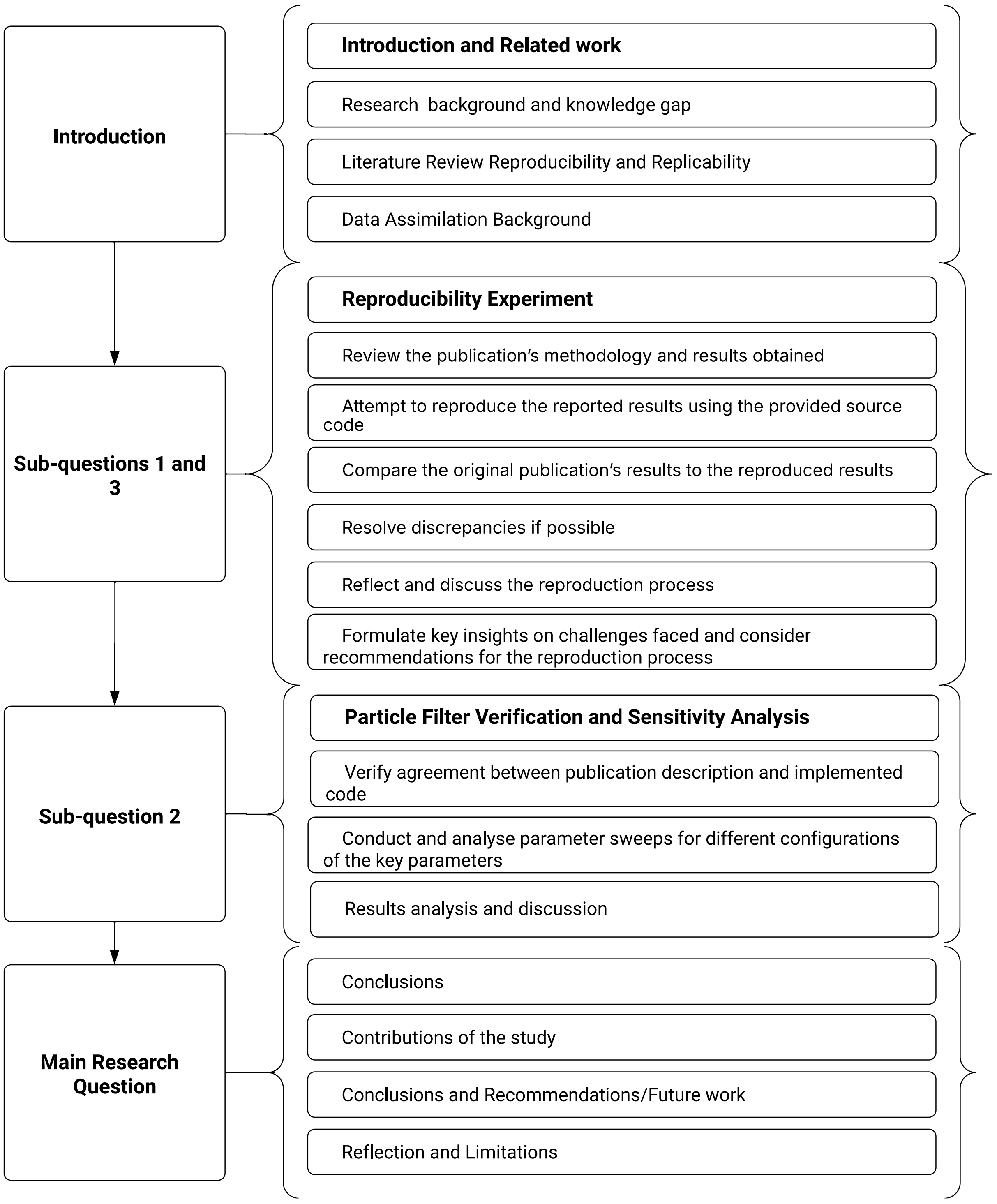
The reproduction was conducted using the original codebase and analysis scripts provided by the case study's authors. This involved running the scripts and determining whether the output was the same as results reported in the original study. Similarity between the original and reproduced results are dependent on the reporting of the results and can be difficult to ascertain such as with figures(Eglen & Nüst, 2019; Gundersen, 2021). Therefore, it remained the choice of the researcher on how to assess success of the reproducibility attempt. An experimental log was maintained throughout the process to document all steps during the reproduction as was similarly done in Axtell et al. (1996).

The main research question is divided into the following sub-questions:

### 1.2.2 Sub-questions

1. *What challenges arise in attempting to reproduce a data assimilation for agent-based simulation study using the particle filter?*
2. *Which properties/parameters of the particle filter method should be reported to facilitate reproducibility efforts in data assimilation for agent-based simulation studies?*
3. *What procedure can be used to guide future reproducibility efforts in data assimilation for agent-based simulation studies?*

## 1.3 Research Design Flow Diagram



Introduction
Introduction and Related work
Research background and knowledge gap
Literature Review Reproducibility and Replicability
Data Assimilation Background
Sub-questions 1 and 3
Reproducibility Experiment
Review the publication's methodology and results obtained
Attempt to reproduce the reported results using the provided source code
Compare the original publication's results to the reproduced results
Resolve discrepancies if possible
Reflect and discuss the reproduction process
Formulate key insights on challenges faced and consider recommendations for the reproduction process
Sub-question 2
Particle Filter Verification and Sensitivity Analysis
Verify agreement between publication description and implemented code
Conduct and analyse parameter sweeps for different configurations of the key parameters
Results analysis and discussion
Main Research Question
Conclusions
Contributions of the study
Conclusions and Recommendations/Future work
Reflection and Limitations

*Figure 1: Research Design Flow Diagram*

### 1.3.1 Reproducibility Experiment

The methodology employed for this experiment was based on the reproducibility and replicability literature as well as the code verification procedures used by journal reviewers and independent reproducibility code review initiatives like CODECHECK. The reproducibility experiment focussed on the computational reproducibility of the publication's results rather than the model reproducibility. The results from this section are used to address sub-questions 1 and 3.

The general research steps are described as follows:

**Step 1: Review the case study's methodology and results obtained**

In this step, the goal was to understand the aim and methodology of the case study and to determine the main results to be reproduced. Following the style of code review initiatives, relevant results of the case study included available figures/plots and tables among other forms of result reporting.

**Step 2: Attempt to reproduce the reported results using the provided source code**

Step 2 involved the attempted reproduction of the figures and tables presented in the original research paper using the available source code.

**Step 3: Compare the original publication's results to the reproduced results**

The original study's results and that of the reproduction were compared using a suitable assessment method from literature.

**Step 4: Resolve discrepancies if possible**

This step involved examining the discrepancies between the original and reproduced results and investigated the cause. Step 2 and Step 3 were repeated until either satisfactory results were achieved, or considerable attempts had been made with minimal success. All challenges encountered and changes made were documented/recorded in an experimental log to keep track of errors encountered and solutions employed to resolve them.

**Step 5: Discussion and Conclusion**

In this step, the researcher reflected on the reproduction process of the case study to distil insights into the factors that facilitated and hindered the process and to devise a protocol for conducting reproducibility experiments. Insights into the main challenges encountered were formulated based on the experiment log.

All steps and findings from the reproducibility process were documented accordingly regardless of the success or failure of the attempted reproduction.

### 1.3.2 Particle Filter Sensitivity Analysis

This section of the research investigated the effect of fundamental parameters of the particle filter algorithm implemented in the case study on the accuracy of the estimated system states. The methodology was similar to the study by Cho et al. (2020) which studied the effect of select conditions on the particle filter applied to discrete event simulation models. Furthermore, the sensitivity analysis aided in understanding the sensitivity of the case study's methodology by exploring more of the parameter space than was reported in the published study similar to the approaches taken by Edmonds and Hales (2003) and Galán and Izquierdo (2005).

This analysis aimed to explore the impact of different particle filter configurations on the case study's framework by investigating the influence of the parameters on the accuracy of the state estimates. It also served to verify the code implementation used in the original publication and to identify any reporting discrepancies.

The steps for this section are as follows:

**Step 1: Verify agreement between the algorithm described in the publication and the code**

Agreement between algorithm description and experimental setup in the publication and the corresponding code implementation was examined. Any differences between the publication and the code implementation were noted.

**Step 2: Identify the target parameters for the sensitivity analysis based on particle filter theory, literature and implementation parameters.**

**Step 3: Implement and conduct parameter sweep of the selected parameters.**

**Step 4: Discussion and Conclusion**

The effects of the parameters on the state estimates were recorded, evaluated and compared to the default configuration used in the original publication.

### 1.3.3 Hardware and Software implementation Specification

The methodology for this research project was implemented using the hardware and software specified in Table 1.

*Table 1: Specifications of hardware and software used in the thesis project*

| **Hardware used** | **Description** |
|---|---|
| System | LENOVO ThinkPad P14s Gen 5 AMD |
| Processor | AMD Ryzen 7 PRO 8840HS w/ Radeon 780M Graphics, 3301 Mhz, 8 Core(s), 16 Logical Processor(s) |
| RAM | 16 GB |
| **Software used** | **Description** |
| Operating System | Microsoft Windows 11 Home |
| Operating System Version | 10.0.26100 Build 26100 |
| Integrated Development Environment (IDE) | Visual Studio Code May 2025 (version 1.101.2) Release date: June 12, 2025 |
| Python | Python 3.13.1 |
| Package Manager | pip 25.1.1 |
| Provider of pseudo-random number generator | Numpy 2.2.4<br>Command: *numpy.random.seed*<br>*More details in code repository specifying the random seeds used.* |

### 1.3.4 Data Sources and Availability

No external raw datasets were required or used in this thesis research project. All relevant materials for the case study were sourced from the publisher's website. Relevant materials included the research article, supporting materials and associated archived code repositories.

The code repository for used for this thesis project is available at https://github.com/5690-cosh/test-bus-sim.git .

# III. Reproducibility and Replicability

In this section, the concept of reproducibility and replicability are explored to illustrate the significance of the crisis particularly in the M&S field. The approaches taken by different actors in the research community to conduct reproducibility and replication studies and to address the crisis are also discussed.

## 1.4 Why conduct reproducibility and replicability research in M&S?

*Literature in M&S frequently address the issue of model replication, therefore in this short section, replication mainly refers to model replication used for verification of simulation models.*

The inability to independently verify and reproduce simulation experiments has created a credibility gap in M&S(Yilmaz, 2011). Successful replications of simulation studies can demonstrate that an experiment's findings are repeatable and were not an exception(Yilmaz & Ören, 2013). Yet the replication of computer models is rarely conducted by independent researchers across the various disciplines that rely on computational models (Axelrod, 1997; Yilmaz & Ören, 2013). This statement was echoed especially in the ABMS field by the literature review of Zhang and Robinson (2021) which found that replication-related research in ABMS was quite poor when reviewing six prominent journals.

Without verification via model replication, it is possible that published findings are incorrect due to programming errors, mistakes in the analysis or reporting of results or misrepresentation of the simulation experiment(Axelrod, 1997). The replication of experiments by independent researchers helps to discern whether a study's results can be attributed to crucial model assumptions or artefacts created from arbitrary choices during design, implementation or runtime of the simulation(Galán & Izquierdo, 2005). Furthermore, simulation studies proposing a novel method are likely to report more positive results when compared to existing methods(Wilensky & Rand, 2007).

These insights into the importance of the replicability issue in M&S also encompass the reproducibility issue because if simulation results are not reproducible then it cannot be expected that they are replicable(Epskamp, 2019).

In the instance of replicating simulation studies to evaluate statistical methods - *replicability as defined by National Academies of Sciences and Medicine (2019)* - the lack of sufficient detail in reporting on a statistical method can result in limited understanding of the method's strengths and limitations(Williams et al., 2024). This hinders the reproducibility and comprehensibility of findings which may lead to misuse of statistical methods when adopted by other researchers.

This thesis project focussed not on replicability or model replication but on reproducibility which serves as a basic requirement for replicability(Epskamp, 2019).

## 1.5 Benefits of Reproducible Research

The many potential benefits of reproducible research have been discussed in literature. (Edmonds & Hales, 2005; Hernandez & Colom, 2025; Miłkowski et al., 2018; Taylor et al., 2017) These benefits include:

- Improving the quality of scientific writing
- Detecting research fraud and bias
- The advancement of operational knowledge

- Enabling the reuse of knowledge and data
- Testing of novel simulation methods i.e. the validation of new algorithms, analysis methods or experimentation techniques which would require detailed specifications to facilitate reuse.
- Increasing credibility and confidence of research findings
- Supporting reliable and accurate research results
- Increasing research visibility and impact
- Builds confidence and trust in the simulation mechanism used in the study

## 1.6 Factors contributing to the Reproducibility Crisis

It is a common occurrence that scientific articles cannot be reproduced in their entirety due to insufficiently detailed documentation(Hernandez & Colom, 2025; Heroux, 2015; National Academies of Sciences & Medicine, 2019). Factors contributing to this situation in the computational sciences with an emphasis on simulation studies include:

- A large proportion of researchers lack formal training in software development
- Insufficiently detailed publications
- Unavailable source code
- Lack of clean code which impairs interpretability, reuse and extension of the code
- The use of software or models that are not Open Source and cannot be disclosed at the source code level
- The continuous evolution of computer systems and architecture which inhibits long-term reproducibility
- Lack of information on all software and system dependencies necessary for code execution
- Difficulty sharing code as there is no agreement amongst journal policies on repositories, metadata or computational provenance.

(Chirigati et al., 2013; Dalle, 2012; National Academies of Sciences & Medicine, 2019; Nüst & Eglen, 2021; Stodden et al., 2013; Stodden et al., 2018; "Supporting computational reproducibility through code review," 2021; Trisovic et al., 2020)

## 1.7 What solutions have been proposed?

To address the reproducibility crisis, policies have been implemented by several scientific journals and initiatives have been organised by research communities centred on teaching and encouraging reproducibility practices(Nüst & Eglen, 2021). Code verification and code peer review are methods that have been proposed and implemented to improve code quality and increase trust in published results. Some journals have instituted systematic code reviews which require peer reviewers to verify code functionality, reproducibility of findings and suitability of the documentation("Supporting computational reproducibility through code review," 2021). Other journal policies require that authors release a replication package in a repository upon publication to enable reproducibility and transparency (Trisovic et al., 2020). However, the challenge is that code verification and review demands extra effort of reviewers, thus, the onus is on authors to facilitate the reproducibility of their work as much as possible in their submissions to journals("Supporting computational reproducibility through code review," 2021).

### 1.7.1 The Code Review Process

In this section, the review process used by code review initiatives - CODECHECK and the Replicated Computational Results (RCR) review process - is discussed to provide a general understanding of reproducibility experiment conducted in this thesis project.

The CODECHECK initiative (Nüst & Eglen, 2021) offers to researchers a service to independently verify their computational workflows and provides a certification assuring that a research article's computational workflow can be executed. The CODECHECK process is akin to that of the peer-review process commonly used by journals. It checks whether the code can generate the output that it claims to create and therefore, constitutes the verification of the computational workflow but excludes validation. The verification process relies on the author providing all relevant materials required for reproduction and on communication between the codechecker (i.e. the code reviewer) and the author for when issues arise. The process is repeated until reproducibility is proven or it is concluded that workflow is irreproducible. Assessing a successful reproduction is dependent on the judgement of the codechecker as it can be difficult to determine whether reproduced figures are identical to those in the original article. Additionally, stochastic simulations frequently generate different results and outputs can even differ based on the operating system. Despite this challenge, the act of reproducing a research article's findings can improve transparency and uncover the submission's shortcomings(Nüst & Eglen, 2021).

A code review process similar to CODECHECK is the RCR review process of the computational journal ACM Transactions on Mathematical Software (TOMS). This process is an optional component of the peer review process that is conducted upon the researcher's request to independently confirm that a research manuscript's results are reproducible(Heroux, 2015). The review procedure is similar to that of CODECHECK in that it focusses on reproducing computational results and relies on cooperation between the code reviewer and the author. The RCR reviewer also determines whether the manuscript's results are successfully reproduced and whether they support the research article's findings(Heroux, 2015).

## 1.8 How to assess a successful reproduction/replication success?

There is no single measure that can describe replication success(Aarts et al., 2015). Some reproducibility or replication efforts, like CODECHECK, do not employ specific metrics but rely on the subjective assessment of the code reviewer. The RepliSims project by Luijken et al. (2024) used the equivalence of results between original study and the replication to assess replicability in other words by assuming that results turn out similar if simulated data, implemented computations and software functionalities are similar. Additionally, Aarts et al. (2015) used five statistical measures to evaluate replicability including significance levels and p-values, subjective assessments, effect sizes, and meta-analyses of effect sizes.

Muradchanian et al. (2021) also reported on the difficulty of assessing replication success and the lack of any standardised approaches. In Muradchanian et al. (2021), replication studies were conducted with the aim of investigating and comparing different metrics of replication success while accounting for differing levels of publication bias.

The Replication Standard(Axtell et al., 1996) is one approach to assessing the success of replications and is referenced in several replication studies in M&S (Yilmaz & Ören, 2013; Zhang & Robinson, 2021). It describes categories of equivalence testing for measuring replication success: numerical identity, distributional equivalence and relational equivalence. Numerical identity involves the reproduction of the exact reported results and is typically not expected for any stochastic simulations unless information on random seeds has been specified. Distributional equivalence can be determined by demonstrating that two models produce distributions of results that are statistically indistinguishable. This was determined by formulating a null hypothesis and using the Kolmogorov-Smirnov statistical test. Relational equivalence refers to two models producing the same internal relationship among their results.

In essence, there are a variety of approaches both quantitative and qualitative that can be used to evaluate a successful reproduction. In this thesis project, the methods selected are based on the type of results reproduced and the information available in the original publication.

## 1.9 Reproducibility Frameworks

Frameworks have been drafted in the literature to classify the degree or level of reproducibility of a research publication. In this section, different frameworks and their criteria are reviewed as they present a different means of evaluating reproducibility apart from the dichotomous view of reproducible or not.

Gundersen (2021) emphasised the relation of the researcher's definition of the results to the means used to judge a successful reproduction. Usually, an experiment is considered reproducible if the hypothesis supported by the original agrees with the ones that is supported by the reproduced results. This research article uses the following framework to define reproducibility types and classify reproducibility experiments using degrees of reproducibility. **Outcome reproducible** refers to when the same analysis, interpretation and result are produced. **Analysis reproducible** refers to when the same analysis and interpretation can be done but outcomes are not identical to the original. **Interpretation reproducible** refers to when the neither the analysis nor outcome are the same, but the interpretation of the analysis leads to the same conclusion. The quality and level of documentation can also be used to classify reproductions as follows:

- **R1 Description** – textual description
- **R2 Code** – code and textual description
- **R3 Data** – data, textual description
- **R4 Experiment** – data, code and textual description

Another framework was developed by Raghupathi et al. (2022) for conducting and evaluating reproducibility studies based on the division of the documentation into three sections: Method, Data and Experiment. If these three sections are available such that other researchers can reproduce the results, the study is **R1: Experiment Reproducible**. **R2: Data Reproducible** refers to only having access to the method and data, therefore, only similar findings can be deduced. **R3: Method Reproducible** implies the sole availability of the method section via which researchers can reproduce results.

Dalle (2012) discusses the reproducibility of simulation-based experiments and defines reproducibility as the capacity to produce similar but not necessarily identical numerical data by running a similar simulation. Dalle (2012) presents four levels of reproducibility for the case of computer simulations.

**L1: deterministic, identical computation** – The original experiment can be rerun exactly, and the procedure uses the same source code to reimplement the same computation.

**L2: non-deterministic, identical computation** – At L2, non-deterministic aspects of the simulation study cannot be reproduced exactly but results should be similar and not significantly differ from the original.

**L3: identical scenario and instrumentation** - At L3, the simulation study is reimplemented based on detailed documentation of the study's model and setup. Implementation differences can lead to the production of different results.

**L4: similar scenario and instrumentation** - At L4, a similar experiment can be executed but its experimental conditions are not identical to those of the original.

Chirigati et al. (2013) describes a process for creating and reviewing reproducible research papers and uses three criteria to characterise the reproducibility level of experiments. The first criteria, **transparency**, refers to the degree of availability of the source code and data such as whether limited data or the code scripts are accessible. The **portability** criteria refers to whether the research experiment can be reproduced in an identical, similar or different computational environment. The last criteria, **coverage**, refers to the completeness of the experimental pipeline provided.

These frameworks reflect different aspects of reproducibility that are considered important for scientific research. The primary two perspectives evident in these frameworks are the reproducibility of the result or findings as compared to the quality and availability of documentation and digital artefacts. Raghupathi et al. (2022) orients its classification system solely on the documentation. On the other hand, Chirigati et al. (2013) while also focussing on availability of documentation and digital artefacts, instead defines reproducibility by key characteristics of research that can be achieved by the provision of certain documentation. Gundersen (2021) differentiates between the two perspectives quite distinctly and Dalle (2012) combines them in their framework. Of course, these two viewpoints influence each other as the documentation quality directly affects the degree of similarity that can be achieved between the original and reproduced results. Hence, the quality of documentation can be seen as indicative of the degree of result reproducibility that is achievable.

In short, these frameworks provide another avenue to evaluate the reproducibility of a research article that differs from the use of reproducibility metrics that classify the attempt as a success or failure solely based on results.

# IV. Data Assimilation for agent-based models

This section provides background on data assimilation and the state estimation problem formulation.

## 1.10 Data Assimilation Background

Data Assimilation is a method for combining prior knowledge of a system in the form of a dynamical model with new data (i.e. observations or measurements) from that system to calculate an estimate of the distribution of the true state of the system under study (van Leeuwen, 2015; Wikle & Berliner, 2007). This methodology can be used for state estimation, parameter estimation or to characterise the best model controls (Evensen et al., 2022).

The key components of a data assimilation problem are:

- The observation or measurement model relating the observations to the state
- An a priori dynamical systems model describing the evolution of the system state
- An algorithm to combine the observations and the model

Data assimilation is grounded in Bayesian methodology and can be formulated as a recursive Bayesian estimation problem. The foundation of this approach is Bayes' theorem which can be derived by expressing the joint probability of the system state $x$ and measurements $y$ in terms of conditional probability distributions (pdfs) (Evensen et al., 2022; van Leeuwen, 2015; Wikle & Berliner, 2007).

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x)$$

Bayes' Theorem $$p(x|y) = \frac{p(y|x)}{p(y)}p(x)$$

### 1.10.1 Recursive Bayesian Estimation Problem Formulation

The data assimilation problem can be formulated as a recursive Bayesian estimation or filtering problem for a discrete-time system in which the posterior distribution is recursively computed when observations become available. The following description is adapted from Evensen et al. (2022), Wikle and Berliner (2007), Gordon et al. (1993) and the course notes of H. Driessen (personal communication, March 6th, 2025).

Assume that a system model and observation model are available. The state vector a time $k$, $x_k \in \mathbb{R}^n$ , is assumed to evolve according to the system model

**Dynamical model** $x_{k+1} = f_k(x_k, w_k)$

$f_k$ is the system transition function and $w_k$ is the process noise. Assume the pdf of $w_k$ is known.

Observations $y_k \in \mathbb{R}^p$ become available at regular time intervals and are related to the state vector via the observation model.

**Observation Model** $y_k = h_k(x_k, v_k)$

$h_k$ is the observation function and $v_k$ is the observation noise with a known pdf.

For a period of $T$ assimilation windows, the prior pdf of the system state can be formulated as:

$$p(x_{0:T}) = p(x_0)p(x_1|x_0)p(x_2|x_1)\dots p(x_k|x_{k-1})$$

$$p(x_{0:T}) = p(x_0)\prod_{k=1}^{T} p(x_k|x_{k-1})$$

$p(x_k|x_{k-1})$ is the evolution density i.e. the probabilistic mode of the dynamical model and $p(x_0)$ is the prior pdf of the initial state.

Assume that the incoming observations are independent and the associated errors are uncorrelated. At time step $k$, a set of observations $y_k = \{y_i : i = 1, \dots, k\}$ are available.

The likelihood function for the observation vector can be written as the product of the independent likelihoods for each assimilation window.

$$p(y_{1:T}|x_{0:T}) = \prod_{k=1}^{T} p(y_k|x_k)$$

Bayes' Theorem can then be rewritten as

$$p(x_{0:T}|y_{1:T}) \propto p(x_0)\prod_{k=1}^{T} p(x_k|x_{k-1})p(y_k|x_k)$$

This formulation demonstrates that as observations arrive, the previous state estimate is updated by treating it as the prior pdf for the current estimate.

Given this formulation, to construct the posterior pdf i.e. the pdf of the current state given all the available information, $p(x_k|y_k)$, two steps must be performed recursively: prediction and update.

It is assumed that the posterior pdf from the previous assimilation window is available when a new observation arrives at the current time, $k$. The posterior pdf of the previous window, $p(x_{k-1}|y_{1:k-1})$, becomes the prior for the current assimilation window and is used to determine the forecast distribution $p(x_k|y_{1:k-1})$, and the updated distribution i.e. the posterior for the current window, $p(x_k|y_k)$.

**PREDICTION STEP**

$$p(x_k|y_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1}$$

**UPDATE STEP**

$$p(x_k|y_{1:k}) = \frac{p(y_k|x_k)p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})}$$

In essence, sequential data assimilation consists of iterating between the prediction and update steps to recursively compute the posterior pdf.

## 1.11 Particle Filter

### 1.11.1 General Particle Filter Algorithm – SIR Filter

There are several variations of the particle filter. In this section, a generic and basic form of the particle filter commonly known as the bootstrap filter is described according to Arulampalam et al. (2002); Doucet et al. (2001); Gordon et al. (1993). This filter uses the idea of Sampling-Importance-Resampling (SIR) and is also termed the SIR Particle Filter.

The concept behind the particle filter is to represent the posterior pdf using a set of random samples (i.e. particles) and corresponding weights. As the number of particles becomes very large, the set becomes an equivalent representation of the posterior pdf(Arulampalam et al., 2002).

In essence, the particle filter is an algorithm for propagating and updating a set of random samples $\left\{x_{k-1}^{(i)}: i = 1, \dots, N\right\}$ from $p(x_{k-1}|y_{k-1})$ to obtain a set of samples $\left\{x_k^{(i)}: i = 1, \dots, N\right\}$ which are approximately distributed as $p(x_k|y_k)$ (Gordon et al., 1993).

At time $k$ , a set of equally weighted random samples (i.e. particles) $\left\{\left\{x_{k-1}^{(i)}, \gamma_{k-1}^{(i)}\right\}: i = 1, \dots, N\right\}$ is available.

**PREDICT STEP**

Draw samples $x_k^{(*i)}$ from the evolution density $p\left(x_k \middle| x_{k-1}^{(i)}\right), i = 1, \dots, N$ where $i$ refers the index of the sample. This is equivalent to propagating each sample $x_{k-1}^{(i)}$ through the system model:

$$x_k^{*(i)} = f_{k-1}\left(x_{k-1}^{(i)}, w_{k-1}^{(i)}\right)$$

$w_k^{(i)}$ is a sample drawn from the pdf of the process noise $p(w_{k-1})$.

**REWEIGHT OR UPDATE STEP**

On receipt of measurement $y_k$, a normalised weight, $\gamma_k^{(i)}$, is calculated for each prior sample $i$ that is proportional to the likelihood function.

$$\gamma_k^{(i)} \propto p(y_k|x_k^{(*i)})$$

$$\gamma_k^{(i)} = \frac{p(y_k|x_k^{(*i)})}{\sum_{j=1}^{N} p\left(y_k \middle| x_k^{(*i)}\right)}$$

**RESAMPLE STEP**

Resample from $\left\{x_k^{(*i)}: i = 1, \dots, N\right\}$with probability proportional to $\gamma_k^{(i)}$ to generate a set of random samples $\left\{x_k^{(i)}: i = 1, \dots, N\right\}$ for time $k$ for which the weights are reset to $\gamma_k^{(i)} = \frac{1}{N}$.

If $N$ is sufficiently large, then the new set of samples $\left\{\left\{x_k^{(i)}, \gamma_k^{(i)}\right\}: i = 1, \dots, N\right\}$ approximates the updated posterior distribution $p(x_k|y_k)$.

#### Considerations for applying the particle filter

The widespread use of the particle filter is owed to the simplicity of implementing it in code and its real-world applicability as it does not place restrictions on the form of the system and observation models nor does it restrict the pdfs of the process noise and measurement noise. However, there are properties of the algorithm that can greatly affect its performance and need to be considered. Principally, the number of samples or particles required to produce satisfactory approximations of the pdfs is difficult to determine and depends on the dimensionality of the state space, the number of required time steps and the overlap between the prior pdf and likelihood function(Gordon et al., 1993). Another concern is particle degeneracy which manifests over time when all particles except one have negligible weights(Arulampalam et al., 2002). Resampling used in the SIR particle filter is employed to mitigate the effects of particle degeneracy; however, excessive resampling decreases the filter's efficiency due to sample impoverishment. This phenomenon occurs when a few particles with high weights are resampled frequently such that the population of particles becomes almost identical and there is a reduction in particle diversity(Arulampalam et al., 2002).

### 1.11.2 Particle Filter for ABMS

The particle filter is the data assimilation technique commonly applied to agent-based models in several scientific publications (Kreuger & Osgood, 2015; Lueck et al., 2019; Malleson et al., 2020; Murata & Tanaka, 2025; Oswald et al., 2024; Rai & Hu, 2013; Wang & Hu, 2015). This algorithm is used because it is non-parametric and hence appropriate to apply to systems that exhibit non-linear and non-Gaussian behaviours such as agent-based models.

#### *1.11.2.1 Formalisation of Agent-Based Models*

To apply the recursive Bayesian estimation problem framework to an agent-based model, it is necessary to first formalise it as a state-space model. The following is based on the formalisation used in Grazzini et al. (2017) and Wang and Hu (2015) and the work of Liu (2001).

The state-space model is a common dynamic system composed of the observation equation and the state equation which can be represented by a Markov process.

The system state vector of the ABM at time $k$, $x_k$, consists of the set of the states of all agents in the model with $x_{i,k}$ referring to the state of each agent $i$ in the model. This is denoted

$$\boldsymbol{x_k} = \{x_{i,k}\}, i = 1, \dots, N$$

In Wang and Hu (2015), the state-space model representing the agent-based model is formulated with additive process noise $Q_t$, and $ABMTransition$ referring to the state transition function of the agent-based model.

$$\boldsymbol{x_{k+1}} = ABMTransition(\boldsymbol{x_k}) + Q_t$$

The observation model and observation vector need to be defined though this is based on the type of sensor being used and the observation data expected.

$$Y = \{y_1, y_2, \dots, y_T\}$$

$$y_k = h(x_k, v_k)$$

The importance weight of a particle is calculated by comparing the ground truth state vector to the state vector of each particle. In the bootstrap filter, the importance weight is proportional to the likelihood function $p(y_k|x_k)$ which is specified by the modeller. There are many possible choices for the importance weighting function(Doucet et al., 2001). In Wang and Hu (2015), the likelihood is specified as a multivariate Gaussian distribution whereas Kreuger and Osgood (2015) uses a negative binomial distribution as the likelihood function to compute the particle weights.

#### *1.11.2.2 Considerations for applying data assimilation to ABMS*

Due to the nature of ABMS, there exists challenges that must be overcome if data assimilation algorithms like the particle filter are to successfully be applied.

**Data Association problem**

The data association problem which concerns how to map the real-world observations onto the state-space of the agent-based model has not been solved and nor researched in some studies as the identical twin framework is often used and simplifying assumptions made(Malleson et al., 2020)

**High-dimensional state vectors**

On another note, ABMs are typically high-dimensional systems. Thus, when applying the particle filter algorithm, as the dimensionality of the system increases, the number of particles required to produce a suitable estimate grows exponentially along with the computational power required(Malleson et al., 2020)

.

# V. Reproducibility Assessment

## 1.12 Case Study Selection

To select a case study, various papers that specifically described a simulation study implementing the particle filter on agent based models were reviewed. The selection was further narrowed by the availability of original source code from the publication; the researcher's familiarity with the paper's study domain and the complexity of the models used as the thesis project's time constraints had to be considered.

The article Kieu et al. (2020) titled "Dealing with uncertainty in agent-based models for short-term predictions" with a citation number of 35 according to The Royal Society Publishing was selected as the case study for this reproducibility experiment. The research article included a description of the ABM in the Appendix. The original source code was also available via Zenodo archive and Github. All relevant materials were sourced from the following locations:

- https://royalsocietypublishing.org/doi/10.1098/rsos.191074
- https://doi.org/10.5281/zenodo.3549633

## 1.13 Case Study Overview

The study's objective was to improve the accuracy of short-term forecasts using ABMs by performing dynamic state estimation to reduce the uncertainty of the model's current system state. Three models were developed in the case study and each was executed for 6000 seconds and 20 bus stops.

**BusSim-truth**

BusSim-truth can be briefly described as a bus route model that generates synthetic GPS data for each bus on the route over a simulation runtime of 6000 seconds for which a single simulation timestep is 10 seconds. The model contains two agent classes - buses and bus stops - and other parameters. A bus can be in one of the following states during the simulation: MOVING when on the road; DWELLING when stopping at a bus stop; FINISHED when its service has ended or IDLE. Every timestep, each bus agent checks whether its planned dispatch time was larger than the next timestep. If yes, then the bus state is checked and if it is MOVING and not at a bus stop then the bus speed is compared to the traffic speed. If its speed is slower than the traffic speed then the bus speed increases else, it remains constant. Alternatively, if the bus is approaching a bus stop and it is the last bus stop then its state changes to FINISH and speed changes to zero else the bus state will change to DWELLING at the bus stop.

The model is both stochastic and dynamic and is used to generate synthetic 'pseudo-truth' GPS data in the form of two datasets: historical GPS data and real-time GPS data.

The level of stochasticity and dynamicity of the model can be controlled to reflect bus route systems which are either stable or changing over time.

The level of stochasticity in BusSim-truth is controlled by the equation for the passenger arrival rate at each bus stop, $m$. The arrival rate represents the number of passengers arriving at each bus stop per minute.

$$Arr_m = \mathcal{U}(minDemand, maxDemand)\ , \quad m = 1, \dots, M$$

If $minDemand = maxDemand$, then BusSim-truth would become a deterministic model.

The level of dynamicity in the model is controlled using the $dynamic\ change\ rate, \xi$, parameter. This parameter is used to vary the arrival rate $Arr_m$ and traffic speed, $V$, throughout the simulation run according to the following equation:

$$V = V \cdot \left(1 - \frac{t}{T} \cdot \frac{100}{\xi}\right)$$

$$Arr_m = Arr_m \cdot \left(1 - \frac{t}{T} \cdot \frac{100}{\xi}\right)$$

Note: It was observed that the equations above from the case study publication were not in agreement with the corresponding code implementation.

**BusSim-deterministic**

This model is deterministic and runs the exactly the same way during each model run when initialised with the same parameters.

**BusSim-stochastic**

This model is stochastic but not dynamic. It is similar to BusSim-truth in that the passenger arrival rate at each bus stop is initialised according to the earlier equation but the arrival rates and traffic speed once initialised do not change over time. The *maxDemand* parameter has influence on this model and there is no dynamic change rate parameter.

### 1.13.1 Case Study Methodology

The case study explored the application of parameter calibration and a data assimilation technique, i.e. particle filter, to reduce the uncertainties in the predicted bus trajectories of bus operation models. The primary steps involved were:

1. Generate pseudo ground truth data for real-time GPS data and historical GPS data.
2. Calibrate BusSim-deterministic and BusSim-stochastic using the historical GPS data via the Cross Entropy Method.

   The Cross-Entropy Method (CEM) was used to calibrate the model parameter vector $S_t$of the models. The model parameter vector consists of the arrival rate at each bus stop $m$, the departure rate at each bus stop, $m$, and the traffic speed.

   $$S_t = [\, Arr_m^t, Dep_m^t, V^t ]\,, m = 1 \dots M$$
3. Using the particle filter, update the states of the calibrated models to the real-time GPS data to produce more accurate short-term forecasts.

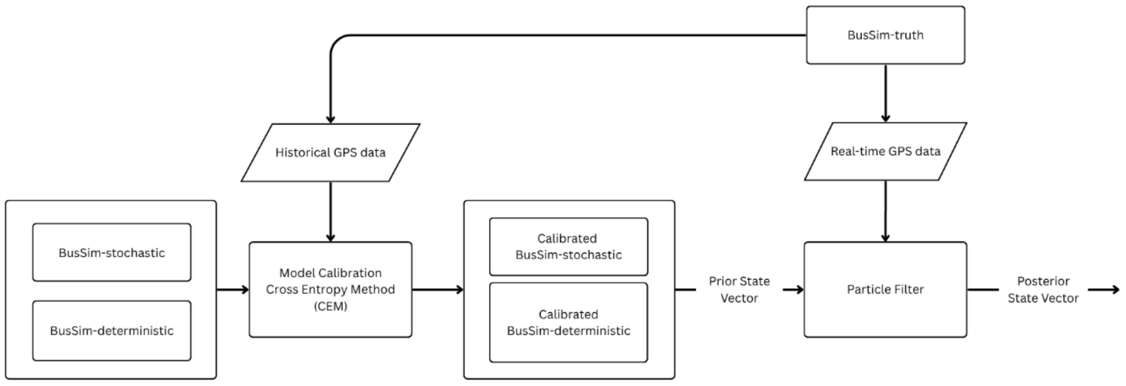


*Figure 2: Case Study Methodology Workflow. Derived from the description in Kieu et al. (2020)*

### 1.13.2 Case Study Experimental Setup

The numerical experiment setup aimed to evaluate the predicted results of BusSim-deterministic and BusSim-stochastic as compared to the synthetic 'ground-truth' data under different scenarios. "Scenario 1: No calibration (benchmark)" serves as a benchmark and evaluates the models' predictions without parameter calibration or data assimilation. Under "Scenario 2: parameter calibration", the model state vectors of the two models are calibrated using CEM and the predictions of bus trajectories evaluated. Under "Scenario 3: applying a particle filter", the particle filter is applied to the calibrated models, BusSim-deterministic and BusSim-stochastic. The observation vector $O_t$ is provided to the two models and used to correct the prediction of their future system states $X_t$. A sensitivity analysis was done in the case study to compare the prediction error in each of the scenarios by varying two parameters: $maxDemand$ and $dynamic\ change\ rate$ individually. The prediction error was evaluated using root mean square error (RMSE) between the real-time and predicted bus trajectories.

*Table 2: Experimental Setup of Scenarios used in the Case Study Kieu et al. (2020)*

| Scenario | CEM Parameter Calibration | Particle Filter |
|---|---|---|
| 1 | ✗ | ✗ |
| 2 | ✓ | ✗ |
| 3 | ✓ | ✓ |

## 1.14 Reproducibility Experimental Setup

Upon reviewing the case study publication, 5 figures and one table were identified for reproduction along with the corresponding interpretations of results. These outputs were selected because they were produced using data generated by the available codebase. One figure used an external dataset that was not provided and was thus excluded from the reproduction. There is no single metric or standard that can be used to assess a successful reproduction. Therefore, in this experiment, the methods chosen were based on the type of results being reproduced and the availability of data and code.

The computational workflow of the case study was derived based on the information in the research article and the information in the code. Figure 3 displays the general workflow as it was comprehended for the reproduction task.

### 1.14.1 Method – Reproducing Figures

The reproducibility of the case study's figures was quantitatively assessed for successful reproduction. The metric used in the case study to compare the performance of each experimental Scenario was the root mean square error (RMSE). Therefore, the RMSE between the real-time bus trajectory and BusSim-stochastic predicted trajectory as well as the error between the real-time bus trajectory and BusSim-deterministic predicted trajectory were calculated for each of the figures both the published originals and the reproductions. 50 simulation replications each using a different random seed were executed for the reproduction of each figure. The RMSE values of the published figures were then compared to the RMSE values of the reproduced figures for similarity.

#### *1.14.1.1 Extracting Data from Figures*

The datasets corresponding to the figures were not included in the downloaded codebase. Consequently, data had to be extracted from the case study's figures to estimate the RMSE values associated with each figure. This was done using the free web-based software, WebPlotDigitzer 5.2 (Rohatgi), for data extraction.

WebPlotDigitzer was used to extract 15 data points from the first 3 bus trajectories of each figure. These points were used to estimate the RMSE values to compare to the RMSE metric of reproduced versions of the figures.

A limitation of this approach was that only 15 datapoints were used resulting in reduced accuracy of the estimated RMSE values from the case study figures. Only 15 datapoints were extracted for each model's trajectories due to limited time available for the manual data extraction using WebPlotDigitzer.

### 1.14.2 Method – Reproducing Tabulated Results

The only numerical results included in the case study were in the form of a table presenting the results of the sensitivity analysis of the model parameters *maxDemand* and *dynamic change rate* which influence the stochasticity and dynamicity of the BusSim-truth model. The table evaluated the prediction error using RMSE for each Scenario and model configuration.

The metric selected to evaluate similarity was whether the original study's effect would be contained in 95% confidence interval of the reproduced estimates. The confidence interval was calculated for the estimated RMSE values of each of the three experimental scenarios. The original table's values were then compared to the reproduced values to check whether they fell within the estimated confidence intervals.

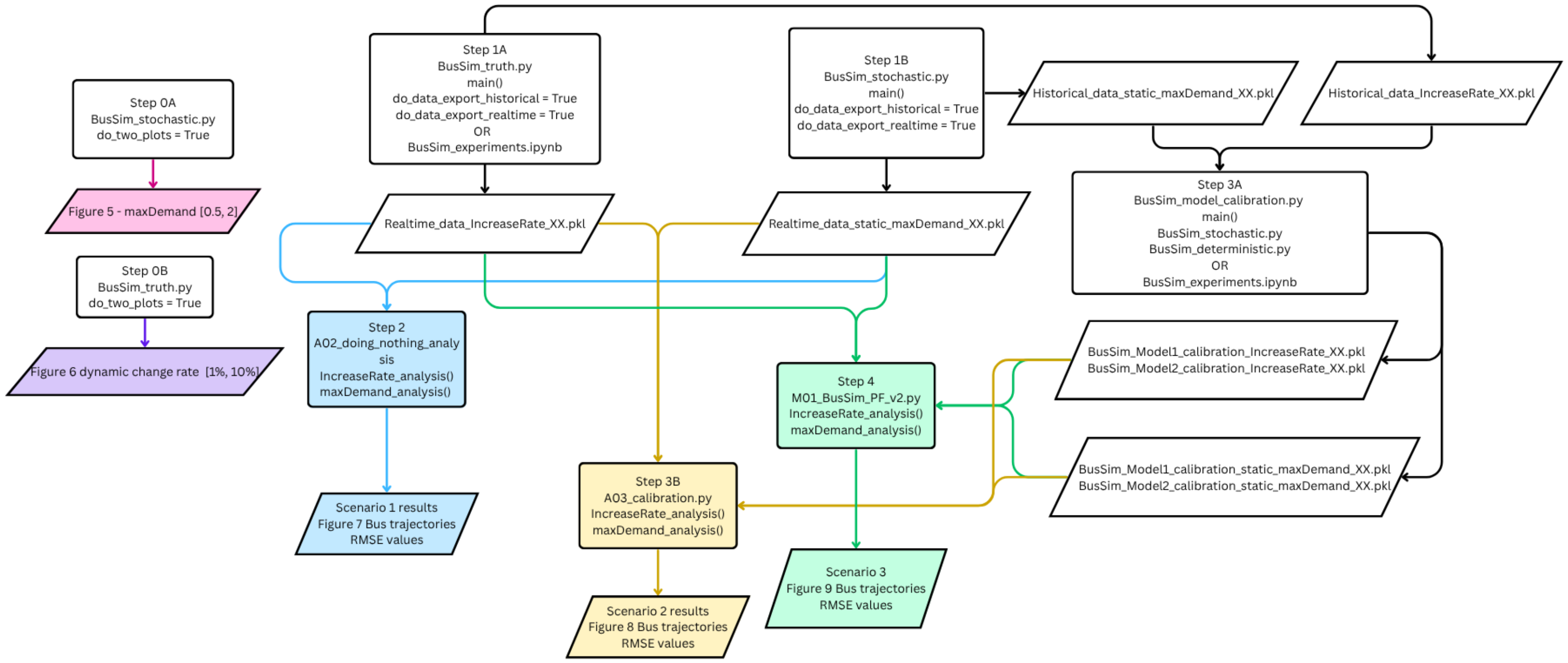
Number of simulation replications used, $n_{sim}$ = 100

$$Monte\ Carlo\ Standard\ Error\ of\ Estimate: \sqrt{\frac{95\ \times (1-95)}{100}} = 2.18$$

Number of replications used in the original study = 10

The number of simulation replications chosen was motivated by both practicality and to reduce the Monte Carlo standard error. The use of 100 replications ensured the reproduced results had a higher power than the original study's results. As such, 100 replications should have been sufficient to reproduce the estimates of the original study and/or show whether the power of the original study was suitable or not.

## 1.14.3 Computational Workflow of Case Study Code Repository



Step 0A
BusSim_stochastic.py
do_two_plots = True
Figure 5 - maxDemand [0.5, 2]
Step 0B
BusSim_truth.py
do_two_plots = True
Figure 6 dynamic change rate [1%, 10%]
Step 1A
BusSim_truth.py
main()
do_data_export_historical = True
do_data_export_realtime = True
OR
BusSim_experiments.ipynb
Realtime_data_IncreaseRate_XX.pkl
Step 1B
BusSim_stochastic.py
main()
do_data_export_historical = True
do_data_export_realtime = True
Realtime_data_static_maxDemand_XX.pkl
Historical_data_static_maxDemand_XX.pkl
Historical_data_IncreaseRate_XX.pkl
Step 3A
BusSim_model_calibration.py
main()
BusSim_stochastic.py
BusSim_deterministic.py
OR
BusSim_experiments.ipynb
Step 2
A02_doing_nothing_analysis
IncreaseRate_analysis()
maxDemand_analysis()
Scenario 1 results
Figure 7 Bus trajectories
RMSE values
Step 3B
A03_calibration.py
IncreaseRate_analysis()
maxDemand_analysis()
Scenario 2 results
Figure 8 Bus trajectories
RMSE values
Step 4
M01_BusSim_PF_v2.py
IncreaseRate_analysis()
maxDemand_analysis()
Scenario 3
Figure 9 Bus trajectories
RMSE values
BusSim_Model1_calibration_IncreaseRate_XX.pkl
BusSim_Model2_calibration_IncreaseRate_XX.pkl
BusSim_Model1_calibration_static_maxDemand_XX.pkl
BusSim_Model2_calibration_static_maxDemand_XX.pkl

*Figure 3: Computational workflow of the case study*

## 1.15 Results

### 1.15.1 Reproduction of Figures of Bus Trajectories

First, 5 graphs, Figures 5 to 9 of the case study's article, were selected for reproduction. The reproduction of Figures 7 to 9 from the publication are discussed in this section. *(See Appendix A for the other reproduced figures and more details.)*

Reproducing these figures was facilitated by the provision of the plotting code in the code repository and the Jupyter notebook describing the basic workflow. The descriptions from the publication though helpful were at times ambiguous or incomplete. There were no datasets included in the repository that could be linked to the plots, therefore, the data files required to create these plots had to be generated. The figures generated were not identical to the originals due to the lack of random seed management in the case study's codebase.

In the case study's article, the figures for each scenario were compared visually and did not include any associated metrics. The reproducibility comparison took a quantitative approach instead. To quantitatively assess the similarity of the original and reproduced figures, the RMSE was calculated for each figure over 50 replications and the RMSE of the case study's original figures was estimated.

A function, *rmse()*, was provided in the case study's code and used for the testing. However, this function was implemented differently from what was reported in the case study's publication.

$$rmse = \sqrt{\left(\frac{1}{T}\sum_{k=1}^{T}(\hat{y}_k - y_k)\right)^2}$$

In the process of reproducing the study, another function *rmse_fixed()* was implemented to evaluate the prediction error using the correct RMSE formula.

$$rmse\_fixed = \sqrt{\frac{1}{T}\sum_{k=1}^{T}(\hat{y}_k - y_k)^2}$$

The mean RMSE was calculated for both the case study's incorrectly implemented *rmse* function and the correct RMSE formula implementation using *rmse_fixed*. These results are displayed in Table 3 and Table 4 for the original published figures and for the reproduced figures, respectively.

*Table 3: Estimated RMSE prediction error values for BusSim-deterministic and BusSim-stochastic calculated using the case study's incorrect implementation of the RMSE formula and the correct RMSE formula. RMSE values were estimated using datapoints extracted from published figures in Kieu et al. (2020) using WebPlotDigitizer software.*

| Figure | Scenario | Case Study Estimated RMSE BusSim-deterministic | Case Study Estimated RMSE BusSim-stochastic | Estimated RMSE BusSim-deterministic | Estimated RMSE BusSim-stochastic |
|---|---|---|---|---|---|
| 7 | 1 | 3151 | 4021 | 13864 | 13708 |
| 8 | 2 | 254 | 2880 | 721 | 10090 |
| 9 | 3 | 51 | 39 | 160 | 178 |

*Table 4: Reproduced RMSE prediction error values for BusSim-deterministic and BusSim-stochastic calculated using the case study's incorrect implementation of the RMSE formula and the correct RMSE formula. The average RMSE was calculated over 50 simulation replications.*

| Figure | Scenario | Case Study Mean RMSE BusSim-deterministic | Case Study Mean RMSE BusSim-stochastic | Mean RMSE BusSim-deterministic | Mean RMSE BusSim-stochastic |
|---|---|---|---|---|---|
| 7 | 1 | 354 | 221 | 6361 | 5479 |
| 8 | 2 | 57 | 60 | 4073 | 4493 |
| 9 | 3 | 88 | 48 | 3095 | 2576 |

The results in the first table showing the estimated RMSE values of the published figures agreed with the case study's visual comparison demonstrating that the prediction error (RMSE) decreased from Scenario 1 to Scenario 3 with Scenario 3 having the best performance when using the results of the correctly implemented RMSE function. In the reproduced results in Table 4, a similar decreasing trend was observed in the RMSE of the reproduced results from Scenario 1 to Scenario 3. These findings agreed with the conclusions drawn from the figures in the research article where the author concluded that Figure 9 (scenario 3) appeared to demonstrate substantial improvement in prediction performance compared to Figures 7 and 8 (Scenarios 1 and 2).

The spread of RMSE values over 50 replications are visualised in the boxplot diagrams below. These plots also demonstrated the reduced prediction error for Figure 9 as compared to Figure 7 and 8 from the case study.

Additionally, examples of the reproduced figures from the experiment were visually compared to the original published figures as shown in the Figure 7: Original Figure 7 Scenario 1: no calibration (benchmark). Source Kieu et al. (2020)Figure 7-12 of this thesis project.

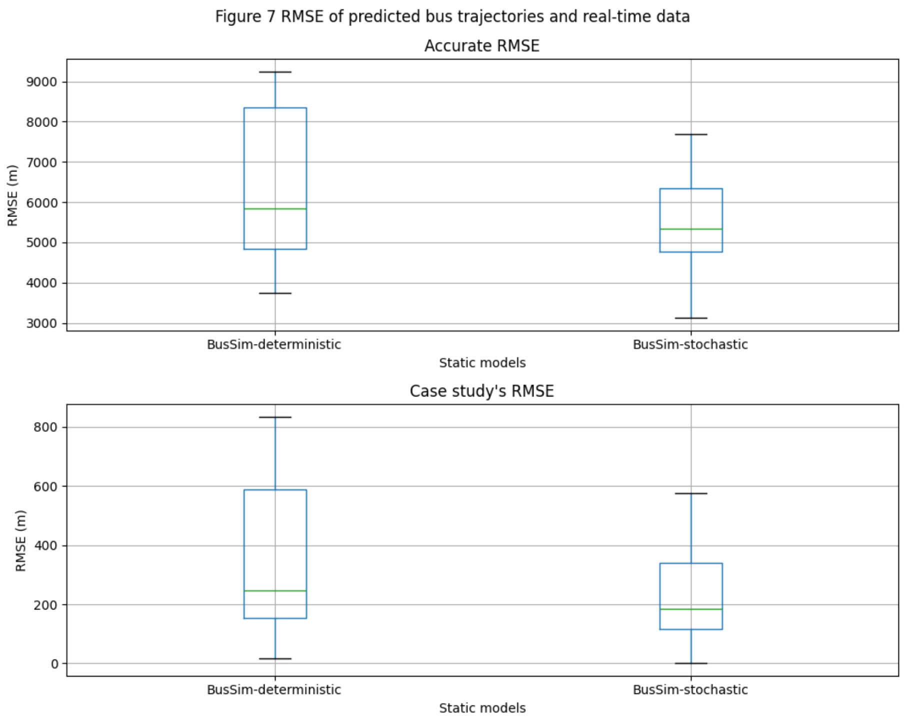


*Figure 4: Boxplot showing the spread of the case study RMSE and correct RMSE formulas for Kieu et al. (2020) Figure 7 using experiment Scenario 1 across 50 simulation replications for the models BusSim-deterministic and BusSim-stochastic*

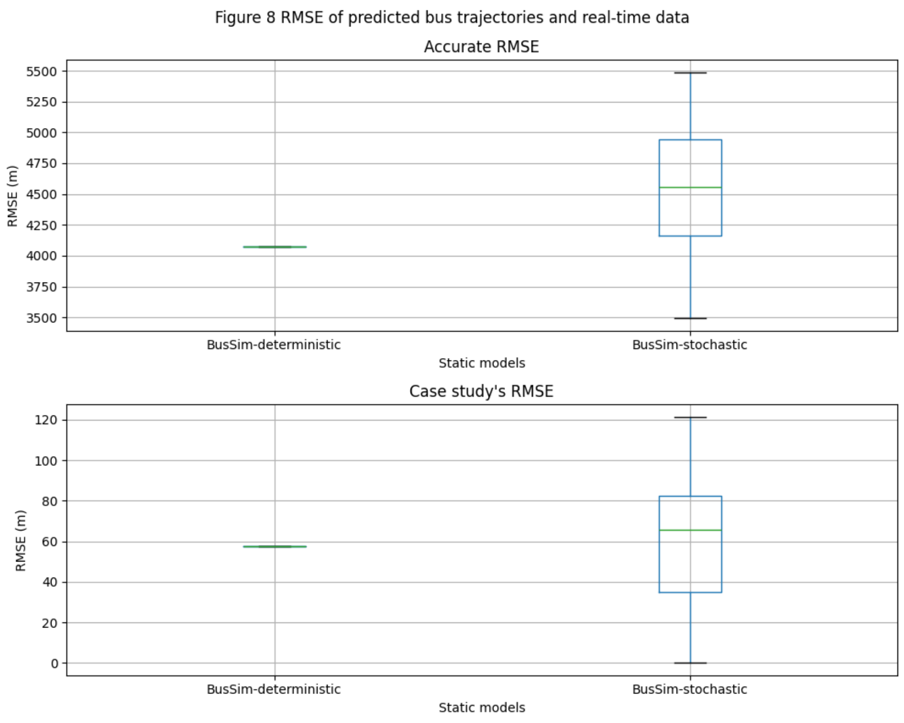


*Figure 5: Boxplot showing the spread of the case study RMSE and correct RMSE formulas for Kieu et al. (2020) Figure 8 using experiment Scenario 2 across 50 simulation replications for the models BusSim-deterministic and BusSim-stochastic*

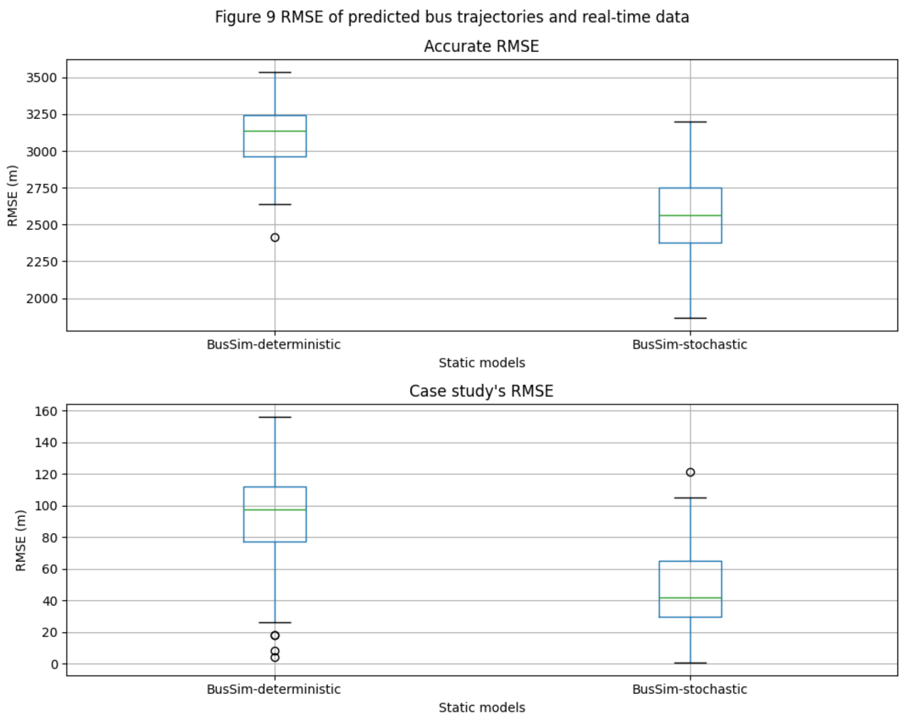


*Figure 6: Boxplot showing the spread of the case study RMSE and correct RMSE formulas for Kieu et al. (2020) Figure 9 using experiment Scenario 3 across 50 simulation replications for the models BusSim-deterministic and BusSim-stochastic*

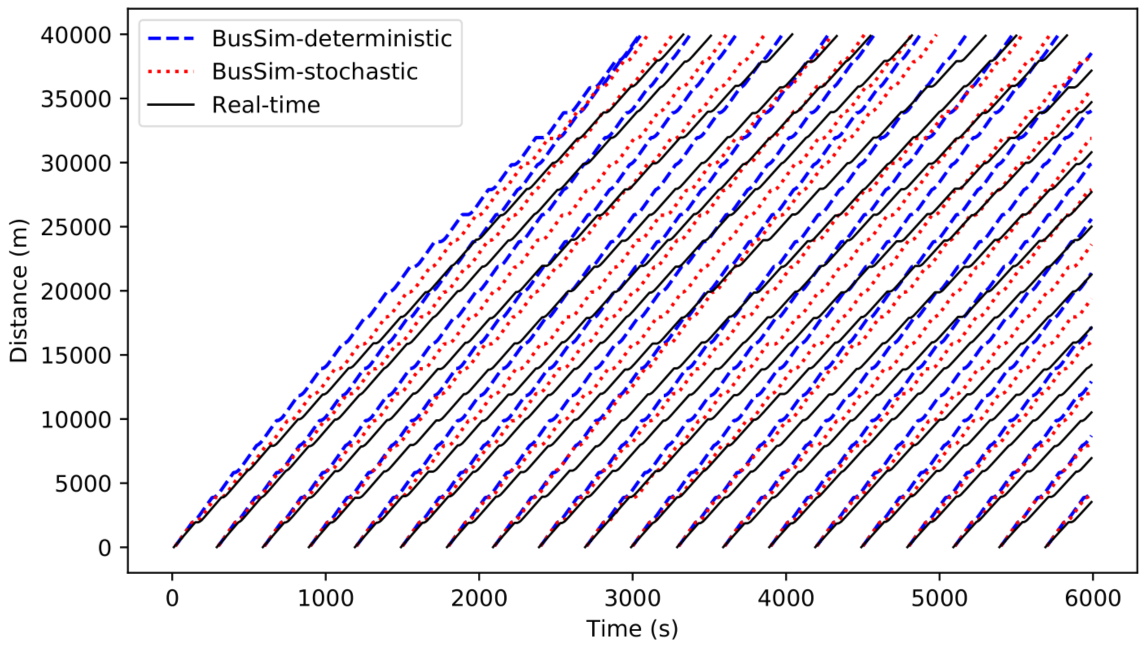


*Figure 7: Original Figure 7 Scenario 1: no calibration (benchmark). Source Kieu et al. (2020)*

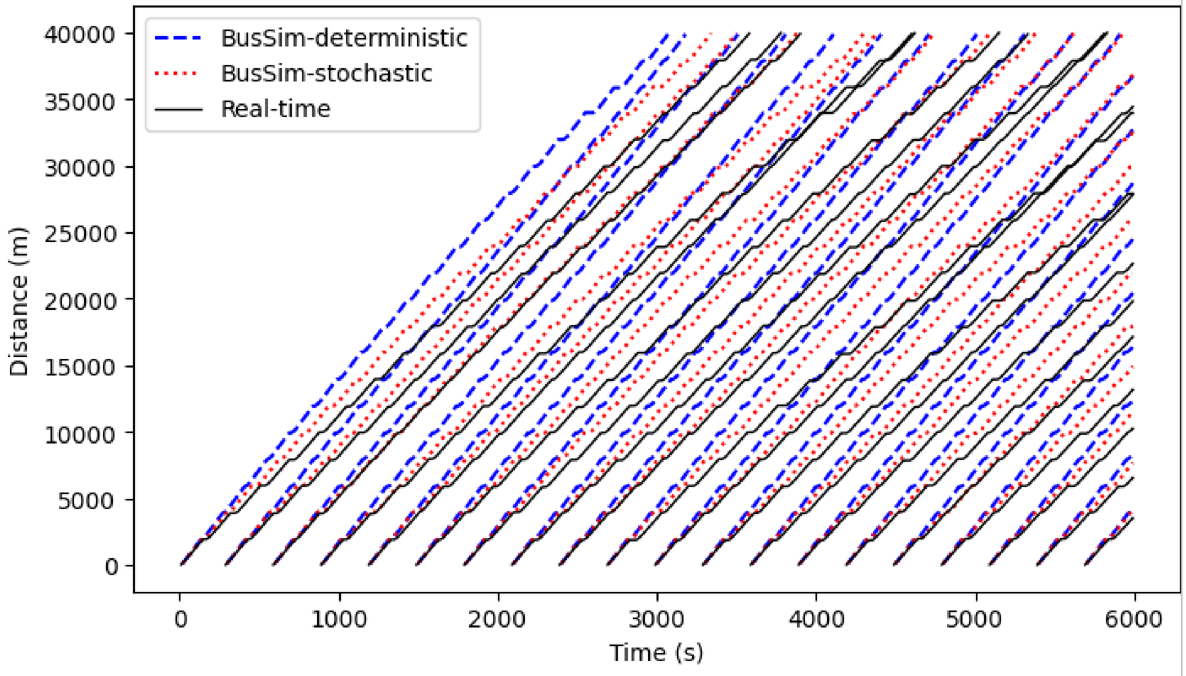


*Figure 8: Reproduced Figure 7 Scenario 1: no calibration (benchmark).*

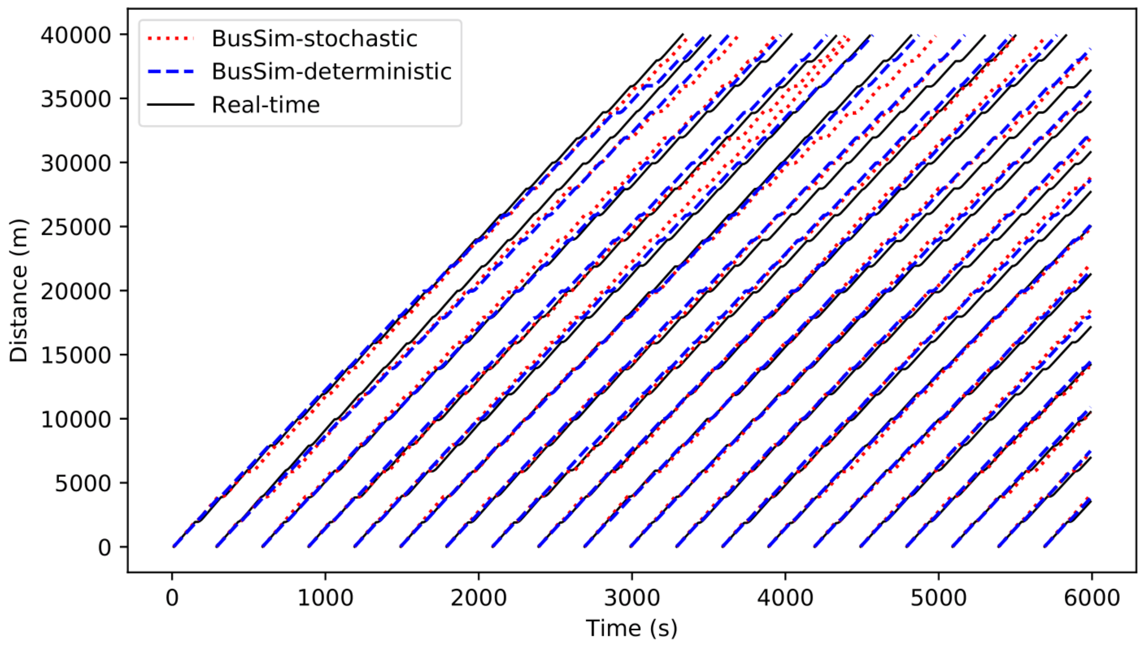


*Figure 9: Original Figure 8 Scenario 2: parameter calibration using cross-entropy method. Source Kieu et al. (2020)*

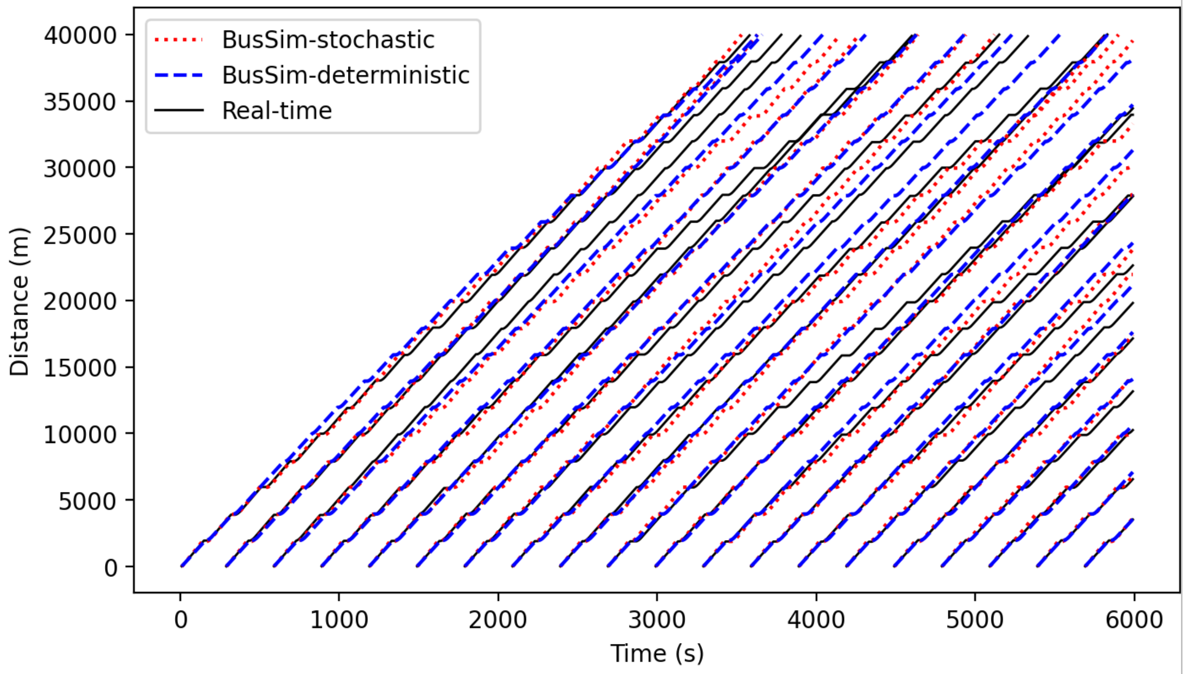


*Figure 10: Reproduced Figure 8 Scenario 2: parameter calibration using cross-entropy method*

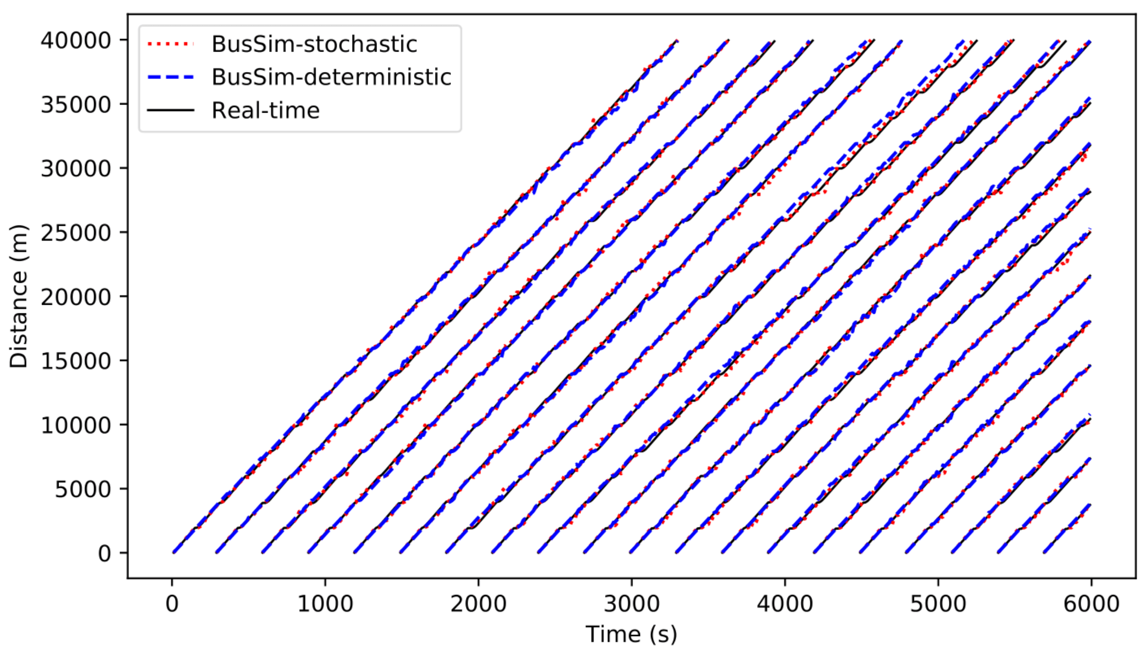


*Figure 11: Original Figure 9 Scenario 3: Parameter Calibration and Particle Filter. Source on Kieu et al. (2020)*

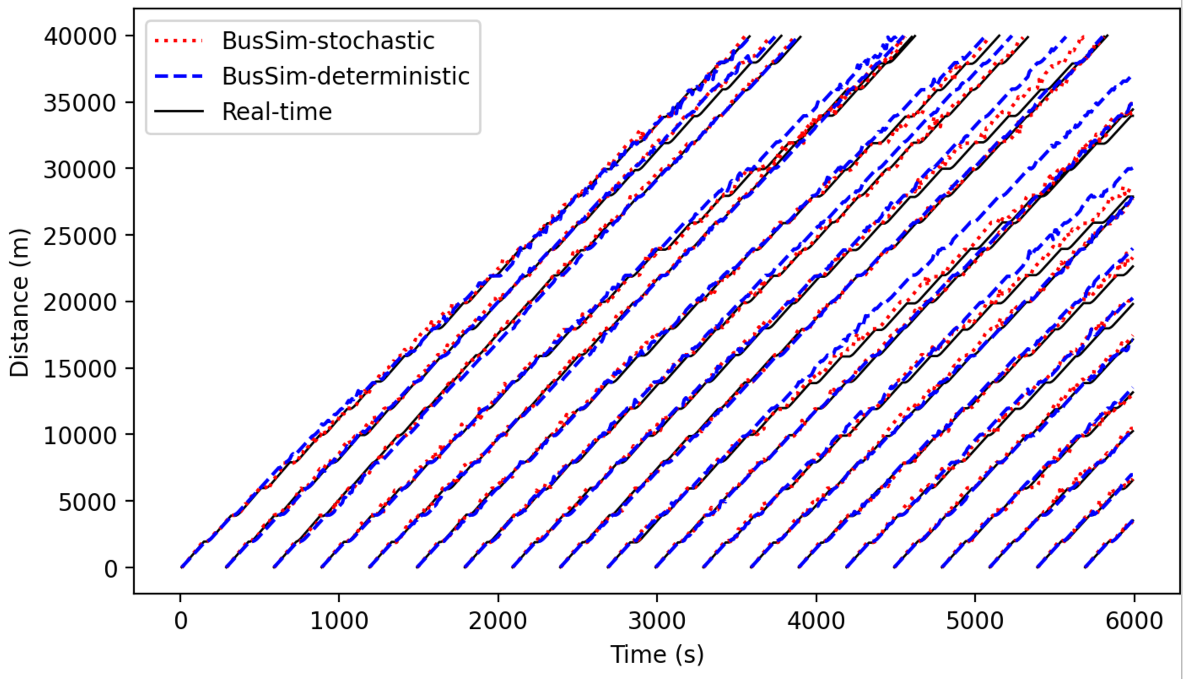


*Figure 12: Reproduced Figure 9 Scenario 3: Parameter Calibration and Particle Filter*

### 1.15.2 Reproduction of Table 3: Sensitivity analysis of maxDemand and dynamic change rate

The case study result *Table 3* of the original publication contained results of a sensitivity analysis conducted using the average RMSE of 10 replications to evaluate the error between each of the static model bus trajectories and the real-time bus trajectories. The main difficulty in reproducing this result stemmed from ambiguities and inconsistencies between the published descriptions and the code structure. A few of these inconsistencies and ambiguities are summarised below.

The research paper did not specify which model, BusSim-deterministic or BusSim-stochastic, was used for the RMSE calculations. Short experiments were conducted using both BusSim-deterministic and BusSim-stochastic which can be found in the Appendix B. However, it was assumed that BusSim-stochastic was used for the sensitivity analysis and these results are presented in experiments 1 and 2 below.

Another issue was that the original data files included in the archived code repository were not generated with the parameter configurations used to reproduce the table's results and/or included some of the parameter configurations while missing others. New files were generated using the codebase to correspond to the range used for the *dynamic change rate* parameter and *maxDemand* parameter in the table. 100 simulation replications each using a different initialised random seed were used to generate the table's results.
Therefore, all historical, real-time and calibration datafiles were generated for the following ranges:

$$dynamic\ change\ rate \in \{0, 2.5, 5, 7.5, 10, 12.5, 15, 17.5\}$$
$$maxDemand \in \{0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5\}$$

This experiment used the provided functions in the study's repository to generate the results. However, as previously mentioned, the case study's implemented RMSE function differed from the RMSE function stated in the publication. Therefore, an RMSE function with the correct formula was implemented and both formulae were used to reproduce the table's results similar to what was done to reproduce the figures.

Lastly, the research paper specified that the models were calibrated to historical data produced by BusSim-truth and while this is true for the *dynamic change rate* sensitivity analysis, the code structure implied that the models were calibrated to data produced by BusSim-stochastic for the *maxDemand* analysis. These assumptions were followed for the reproduction process. This ambiguity may be due to the similarity between BusSim-stochastic and the more complex dynamic model BusSim-truth. BusSim-stochastic lacks the dynamicity of BusSim-truth as the *arrival rate* and *traffic speed* at each bus stop remains constant throughout the simulation run after initialisation. For BusSim-truth, *arrival rate* and *traffic speed* are influenced primarily by the *dynamic change rate* with the arrival rate also influenced by *maxDemand*. BusSim-stochastic lacked this dynamic nature and did not have a dynamic change rate parameter. While this understanding can be derived from the code, it was not specified in the publication which claimed that calibration were done to BusSim-truth data. Furthermore, the case study had clear descriptions of BusSim-truth model but lacked specification of the companion static ABMs BusSim-deterministic and BusSim-stochastic and how they differed from BusSim-truth.

*Table 5: BusSim-stochastic reproduced Average RMSE results using Case Study rmse function to reproduce results of Table 3 Kieu et al. (2020)*

| maxDemand | values | Scenario 1 RMSE | Scenario 2 RMSE | Scenario 3 RMSE |
|---|---|---|---|---|
| | 0.5 | 292 | 58 | 49 |
| | 1 | 250 | 37 | 13 |
| | 1.5 | 230 | 91 | 9 |
| | 2 | 230 | 87 | 63 |
| | 2.5 | 243 | 192 | 13 |
| | 3 | 212 | 138 | 30 |
| | 3.5 | 184 | 42 | 47 |
| | 4 | 169 | 94 | 24 |
| | 4.5 | 168 | 210 | 54 |
| dynamic change rate | values | Scenario 1 RMSE | Scenario 2 RMSE | Scenario 3 RMSE |
| | 0 | 226 | 32 | 27 |
| | 2.5 | 237 | 63 | 23 |
| | 5 | 195 | 166 | 36 |
| | 7.5 | 227 | 85 | 33 |
| | 10 | 234 | 119 | 60 |
| | 12.5 | 233 | 165 | 59 |
| | 15 | 233 | 127 | 52 |
| | 17.5 | 259 | 133 | 115 |

*Table 6: BusSim-stochastic reproduced Average RMSE results using the correct formula to reproduce Table 3 Kieu et al. (2020)*

| maxDemand | values | Scenario 1 RMSE | Scenario 2 RMSE | Scenario 3 RMSE |
|---|---|---|---|---|
| | 0.5 | 5734 | 3345 | 1698 |
| | 1 | 5327 | 3033 | 1061 |
| | 1.5 | 5264 | 3642 | 920 |
| | 2 | 5381 | 3562 | 1176 |
| | 2.5 | 5575 | 4461 | 1075 |
| | 3 | 5434 | 4217 | 1923 |
| | 3.5 | 5504 | 4335 | 3060 |
| | 4 | 5385 | 4315 | 1836 |
| | 4.5 | 5335 | 4598 | 1277 |
| dynamic change rate | values | Scenario 1 RMSE | Scenario 2 RMSE | Scenario 3 RMSE |
| | 0 | 5598 | 3521 | 1429 |
| | 2.5 | 5482 | 3192 | 1341 |
| | 5 | 5253 | 3831 | 2075 |
| | 7.5 | 5435 | 3279 | 1698 |
| | 10 | 5630 | 3800 | 2163 |
| | 12.5 | 5456 | 3892 | 2537 |
| | 15 | 5524 | 4301 | 2849 |
| | 17.5 | 5650 | 4404 | 3339 |

Table 5 and Table 6 showed that case study's Table 3 could not be reproduced identically. From the reproduced results, scenario 3 appeared to generate the lowest estimation errors compared to scenarios 1 and 2. This interpretation agreed with the conclusions made in the original study.

In the bar charts below, the 95% confidence intervals have been plotted as error bars for comparison with the original table 3 results. This was done using the case study's RMSE formula. The majority of the original RMSE values obtained did not lie with the 95% CI of the reproduced results. The exception to this was Scenario 1 results for the sensitivity analysis of the dynamic change rate parameter shown in Figure 14. In which all of the original estimates from Scenario 1 do lie in the 95% CI of the reproduced result. The successful reproduction in terms of 95% CI for Scenario 1 of the dynamic change rate analysis and failure for others may be indicative of incorrect assumptions made for the calibration and particle filter steps. The calibration using CEM would not have produced calibrated model parameters numerically identical to the ones used in the original study and due to the extensive computational resources and time needed to run the model calibrations, the model calibration itself was not replicated but executed once to produce the optimised parameters. For the maxDemand sensitivity analysis the majority of the original case study RMSE values did not lie within the 95% CI.

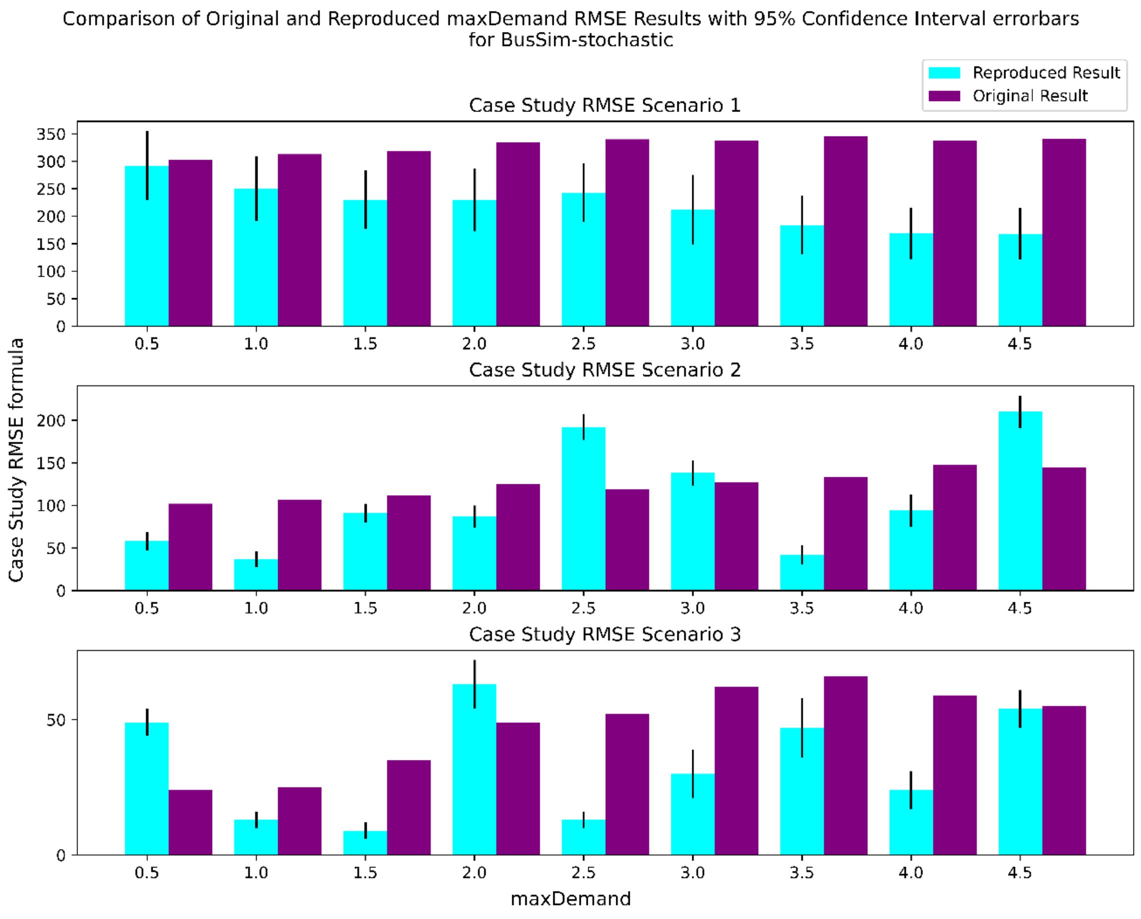


*Figure 13: Bar chart comparing the original values from the maxDemand sensitivity analysis from Table 3 Kieu et al. (2020) to the reproduced RMSE using the Case Study rmse formula and the corresponding 95% confidence intervals of the reproduced results indicated by straight black lines to the .*

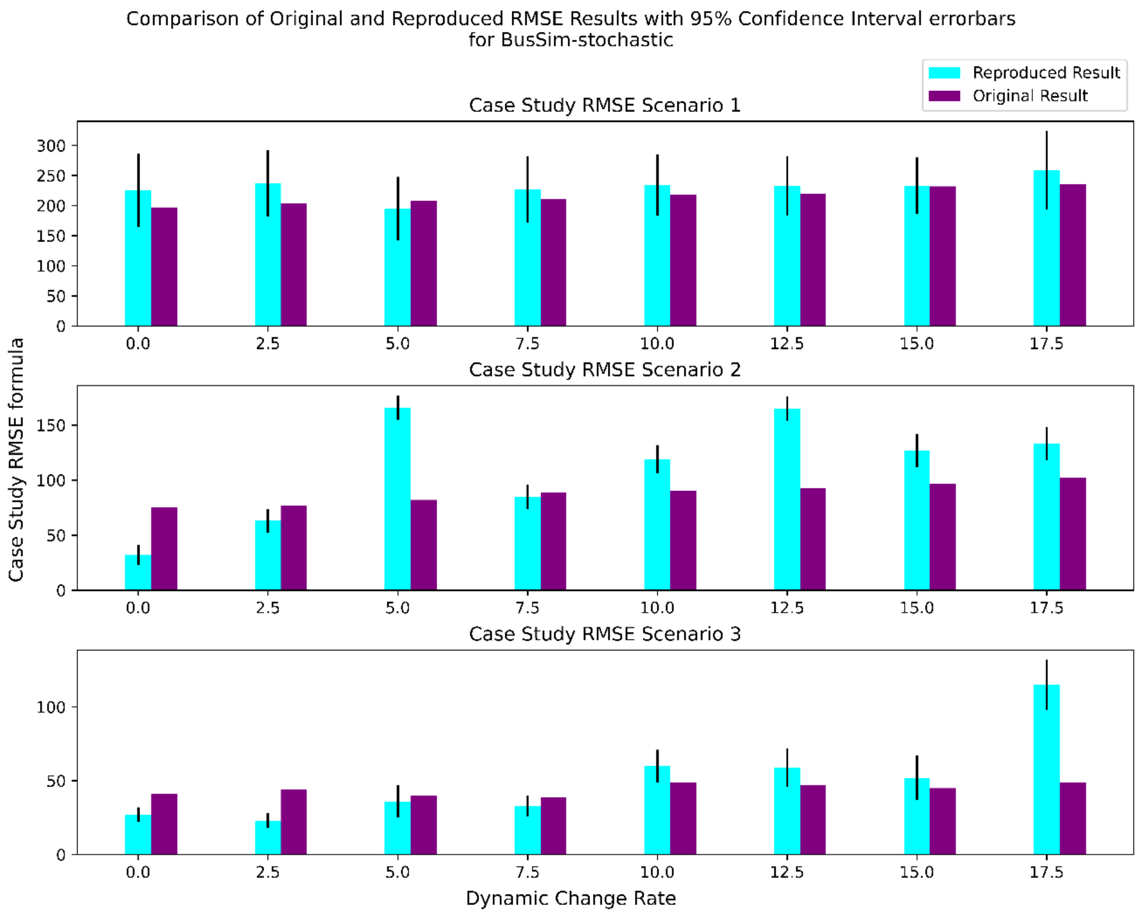


*Figure 14: Bar chart comparing the original values from the dynamic change rate sensitivity analysis from Table 3 Kieu et al. (2020) to the reproduced RMSE using the Case Study rmse formula and the corresponding 95% confidence intervals of the reproduced results indicated*

## 1.16 Discussion

### 1.16.1 Challenges

The reproduction of the case study's results was facilitated by the associated code retrieved from the Zenodo repository, the case study's modular code implementation, the provision of a Python Jupyter notebook that depicted a basic version of the code's execution workflow and a qualitative description of the experimental setup and methodology in the research article. Yet despite the case study's use of such methods, there were still issues to be improved upon and several challenges to the reproduction of the case study's results.

Upon initial review, the codebase appeared well-organised, and a Python notebook was included which demonstrated the steps to generate plots resembling the figures from the publication and which reflected the general experimental procedure described in the publication. Datasets were included in the code package that had been generated by the simulation and had been archived with the code. However, the README.md file was not instructive beyond referring to the research publication and the Python notebook for more information. The example Python notebook provided used these datasets to generate similar bus trajectory plots to those in the publication.

Initial challenges to the reproducibility experiment consisted of basic programming errors often encountered when running another programmer's code in addition to documentation failures. The python notebook was instructive of the experimental workflow used by the author; however, it fell short of being a proper replication package as it did not link directly to the publication and did not generate the plots provided. Issues related to code quality though tedious could be resolved such as simple file path errors. The main challenges for the reproduction came from discrepancies between the documentation of the code and the experiment described in the code and missing parameter specifications and ambiguous descriptions.

Another challenge was the evaluation of the non-numerical results like the figures. The datasets used to make the graph were not made available in the code repository. To circumvent this issue, data extraction was conducted using WebPlotDigitizer. However, this approach carried a small margin of error(Burda et al., 2017).

The chosen reproducibility metric was determining whether the original results lay within the 95% confidence interval of the reproduced results. While easy to implement and applicable for numerical input from tables, this method has several limitations. It ignored the uncertainty in the original study which was high given the low simulation replication number used, however, there was also a high probability that the method could have indicated replication failure when the estimates were the same(Heyard; Heyard et al., 2025).

It should be noted that a major limitation of this reproducibility study was the lack of communication with the study's author due to time constraints of the project. Furthermore, more research is still required into the applicability of different reproducibility metrics to ascertain statistical equivalence of results. Therefore, a limitation of this experiment is that only one method, the 95% confidence interval, was used and a single metric is often insufficient to capture the reproducibility success or failure.

#### *1.16.1.1 Documentation failures encountered*

- README.md was not sufficiently detailed and lacked information on the structure of the project and the functionality of the files. The file only referred the reviewer to a Python notebook for details.
- Missing information on dependencies
  - There was no requirements.txt or environment.yml files included in the downloaded project. No information was provided on the Python packages used nor the Python version used.

##### 1.16.1.1.1 Code Quality Issues

- File path errors encountered due to the use of absolute paths in the code. Although the author did acknowledge that the working directory would have to be changed for the some of the code to be executed in the primary Python notebook.
- Outdated comments were found. For instance, some comments referred to components or files that had been renamed by older names.
- No random seed management implemented which prevented the reproducibility of identical figures

##### 1.16.1.1.2 Code-Publication discrepancies

- Formulae used in model description differed from the publication
- Instructions in code comments and the format of the code differed from the experimental setup described by the article.

##### 1.16.1.1.3 Evaluation using Frameworks

Using the reproducibility framework of Gundersen (2021) that was briefly explained in earlier sections, the reproduction of this experiment can be considered as reproducibility type R4 Experiment due to the level of documentation provided in the form of data, code and textual description. However, this classification does not account for the missing parameter specification and contextual information, nor does it account for the inconsistencies between the implemented code and the textual description. The research paper itself can be classified as interpretation reproducible. It was only partially analysis reproducible due to the ambiguity concerning the experimental frame used to conduct the original case study analysis. Lastly, it was not outcome reproducible as the exact outcomes cannot be regenerated. A similar analysis can be conducted to what was done in the case study, but the missing parameter specifications and/or data files prevent the analysis from being numerically identical. On another note, using the framework of Dalle (2012), the reproducibility study can be considered at level *"L2 non-deterministic, identical computation"* due to the provision of the data files and source code and because the results are similar to the original study but not exactly the same. Yet, the missing components and ambiguities are also not considered in this framework.

The criteria of Chirigati et al. (2013) can be used to account for the ambiguities in the research paper. The case study was partially transparent as it made available some of the data and the executable scripts, the partial pipeline was included therefore the publication showed partial coverage, and it was somewhat portable as it could be reproduced in a different environment but no information about software dependencies had been included in the package.

From the viewpoint of the researcher who conducted the reproducibility experiment, the case study's findings and interpretations were indeed reproducible i.e. interpretation reproducible as both the original and reproduced results confirmed that the Scenario 3 performed better than Scenarios 1 and 2. Yet challenges remained and there were many aspects of the study that could have been improved upon not only for documentation but also to improve the study's design.

### 1.16.2 Recommendations

A major issue in the reproduction process was the missing information relating the figures in the publication to the project code. While some information was provided on the key parameters for the figure through a narrative description, several parameter specifications were missing. Inconsistencies were also found when between the publication's description and the implemented code and code comments.

For researchers to address this issue, one solution is to include metadata that can link the figures to the process used to create them. Drawing from the work of Chirigati et al. (2013), the use of deep captions for figures which would consists of the workflow used to derive the plot, the underlying libraries invoked by the workflow, and the input data would allow these figures to be reproduced more easily. Furthermore, Dalle (2012) mentions that when addressing reproducibility issues, the concept of traceability of simulations should be considered such that a publication's results can be traced to the software elements used to produce them.

A similar concept to traceability, is the idea of provenance of workflows. Provenance is considered vital for facilitating result reproducibility and knowledge re-use in the research community. The provenance of a result would contain detailed specifications of the procedure and data used to generate it such that other researchers could reproduce and validate the results(Davidson & Freire, 2008). While partial details of the execution steps were specified in the case study's materials, there was no documentation specifically addressing the computational provenance of its results. Consequently, a key step in this reproduction process was the knowledge gained in comprehending and mapping the computational workflow required to produce the results. In other words, attempting to create a retrospective provenance of the case study's results uncovered the missing information and ambiguities in the provided documentation This step facilitated both the understanding of the process and documentation of the workflow being followed for the reproduction as shown in Figure 3. Therefore, giving more attention to the capture of provenance information in scientific studies would be advantageous for authors in preparing their publications.

# VI. Particle Filter Sensitivity Analysis

The particle filter sensitivity analysis is explained in this section. The case study did not report the assumptions and key parameters of the particle filter in the publication though parameters were hard-coded in the algorithm implementation. It can be assumed that the hard-coded filter parameter values were the ones used to conduct the analysis reported in the publication. This was the assumption made in the previous reproducibility experiment. Alternatively, by exploring more of the parameter space, it was possible that there was a configuration of the filter parameters that provided a closer match to the reported results.

In this section, first the setup of the data assimilation problem is described followed by the parameter sweeps for the identified parameters used in the sensitivity analysis.

## 1.17 Problem Formulation

Aim: To calculate a posterior probability for the state vector $X_t$ given prior distributions from a model and data from observations.

The full model state is denoted by state vector $X_t$. For simplicity, it is assumed in the case study (Kieu et al., 2020) that the state vector is fixed in size and initialised considering all of the agents that will ever be in the system. If an agent is not yet present in the system yet, then its variables are set to 0.

This state-space model can be represented by the state-space vector which consists of the observation vector $O_t$ and the model parameter vector $S_t$.

$$X_t = [O_t, S_t]$$

$$X_t = \left[c_j^t, s_j^t, v_j^t, Occ_j^t, Arr_m^t, Dep_m^t, V^t\right], m = 1, \dots, M$$

$M$ : Number of bus stops.

$j = 1, \dots\ N_a$

Observation vector: $O_t = \left[c_j^t, s_j^t, v_j^t, Occ_j^t\right]$

$c_j^t - current\ status\ of\ bus\ (IDLE, MOVING, DWELLING\ or\ FINISHED)\ at\ time\ t$

$s_j^t - coordinate\ of\ bus\ locations\ on\ a\ 1D\ lattice\ at\ time\ t$

$v_j^t - current\ speed\ of\ bus\ j$

$Occ_j^t - current\ occupancy\ of\ bus\ j\ at\ time\ t$

Model Parameter vector: $S_t = [Arr_m^t\ \ Dep_m^t\ \ V^t]$

$Arr_m - arrival\ rate\ of\ passengers\ to\ stop\ m\ per\ second$

$Dep_m - departure\ rate\ of\ passengers\ from\ stop\ m\ per\ second$

$V_t - traffic\ speed\ in\ ms^{-1}$

$Arr_m = U(minDemand, maxDemand), m = 1, \dots, M$

$Dep_m = ordered(U(0.05,0.5))$

$Dep_M = 1$

$initial\ traffic\ speed = 14\ ms^{-1}$

$N_a$: Number of agents i.e. buses

The problem can be represented using the form of a general nonlinear discrete-time stochastic dynamic system.

$$\dot{X}_t = f(X_t) + \epsilon_t.$$

$$X_t \in \mathcal{X}$$

$$\mathcal{X} \in \mathbb{R}^n$$

The process noise $\epsilon_t$ is not explicitly modelled in the case study.

The observation model is linear and defined as a measurement of the vector $O_t$ , the GPS data and subset of the model system state $X_t$ at time $t$. It is assumed the assimilation window is fixed in size and that data is assimilated at each time step of the simulation. The observation model is not specified mathematically in the publication. The following equation is based on the code and the qualitative description in the research article where $y_t$ represents the observation data as mapped onto the subset of the observation vector $O_t$.which is a subset of the full state vector $X_t$.

$$y_t = O_t$$

Figure 15 shows the general format of the particle filter algorithm in relation to the other components in the experimental setup. Figure 16 describes the case study's implementation of the particle filter in a flowchart. See Appendix D for the particle filter algorithm transcribed from the implemented code in pseudocode.

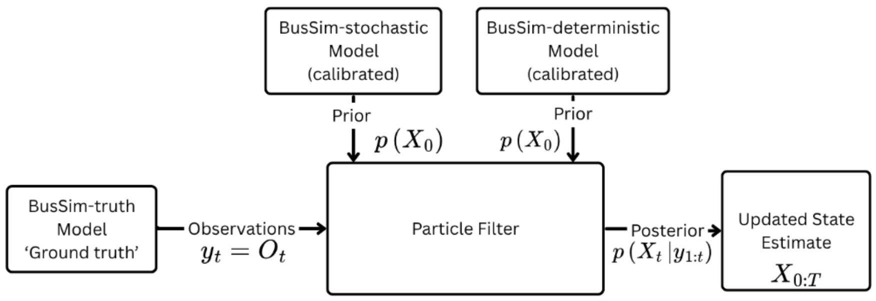


*Figure 15: Block Diagram of Particle Filter setup for case study*

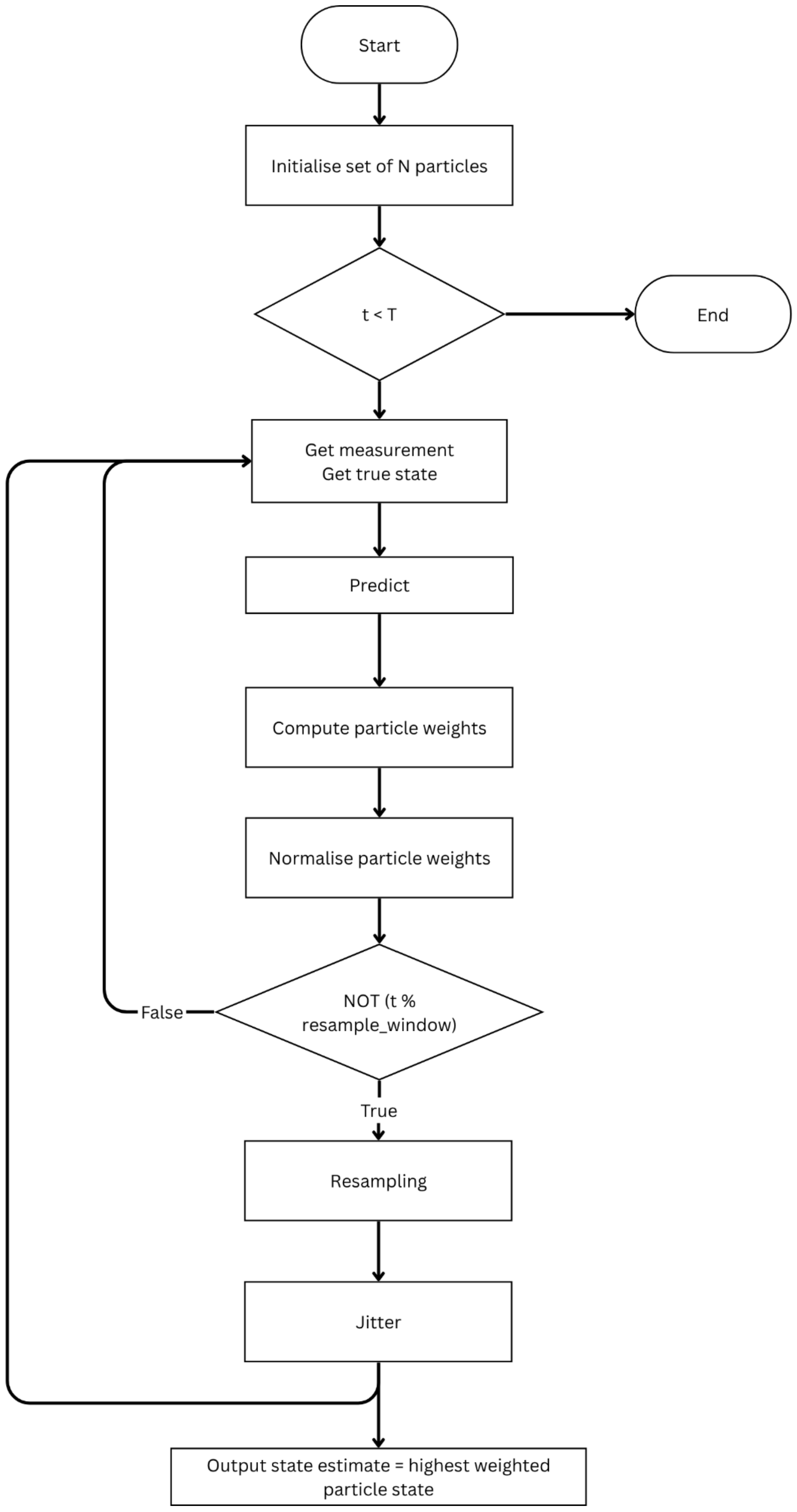

Start
Initialise set of N particles
t < T
End
Get measurement
Get true state
Predict
Compute particle weights
Normalise particle weights
NOT (t % resample_window)
False
True
Resampling
Jitter
Output state estimate = highest weighted particle state

*Figure 16: Flowchart of Particle Filter Algorithm reflecting the coded implementation. See Appendix D for the pseudocode.*

## 1.18 Particle Filter Parameters

The following parameters were varied to examine their effect on the estimation accuracy of the case study's proposed experimental framework. The parameters that were varied are shown in Table 7 for a fixed configuration of the BusSim-truth model - $dynamic\ change\ rate = 7$ and $maxDemand = 2$.

The configurable filter parameters in the case study's particle filter implementation included the number of particles, the resampling window and the jitter noise standard deviation.

The algorithm had been implemented such that a measurement arrived each timestep of the simulation and therefore the assimilation window = 1 timestep. However, this was not variable in the implementation but implicit in the structure of the code. Additionally, measurement noise was not specified in the case study publication nor included as a variable in the code although there was commented code such that measurement noise could be added if desired.

Therefore, the sensitivity analysis focussed on the primary filter parameters of the code implementation.

**Number of particles**

The number of particles specifies the size of the set of particles used to approximate the probability distribution of the system state variable.

**Resample window**

This parameter controls the frequency of resampling and jitter. It is specified as an integer representing the time interval between resampling executions. Resampling is used to mitigate particle degeneracy and also causes sample impoverishment.

**Jitter Noise Standard Deviation**

An independent jitter is drawn from a random Gaussian white noise $j \sim \mathcal{N}(0, \sigma^2)$ in which $\sigma$, the jitter standard deviation, is predefined by the modeller. The jitter is added to each particle's model parameter state vector $S_t$ instead of the whole state vector $X_t$. This procedure occurs after the resampling step to increase particle diversity and mitigate particle degeneracy. The predefined parameter was not specified in the original publication but was hard coded in the case study's codebase as 0.0005.

## 1.19 Experimental Setup

For the following experiments, a fixed model configuration was used by setting the two main parameters $dynamic\ change\ rate = 7$ (called $IncreaseRate$ in the code scripts) and $maxDemand = 2$. This experiment was conducted using the calibrated BusSim-stochastic and calibrated BusSim-deterministic models in accordance with the experimental design of fixing all other components of the case study's framework and using the data provided for the testing. This setup was shown in Figure 15.

The Numpy module global random seed was initialised for these experiments to ensure reproducibility and were specified in the code. 10 replications using 10 different seeds were executed for each combination of the three filter parameters according to the ranges specified in the table below. This resulted in 210 different combinations of the three parameters and for each combination, 10 simulation replications were executed. This is the same number of replications used in the case study's sensitivity analysis though the low number of replications is also a limitation of the experiment.

*Table 7: Parameters investigated for particle filter sensitivity analysis*

| Filter Parameter | Case Study Code Implemented value | Values |
|---|---|---|
| Number of particles | 500 | [10, 100, 500, 1000, 2000, 3000] |
| Resampling window | 1 | [1, 10, 30, 60, 100] |
| Jitter noise standard deviation | 0.0005 | [0, 0.0005, 0.001, 0.01, 0.1, 1, 10] |

The metrics used to evaluate the prediction error in this experiment was the RMSE as was used in the case study's sensitivity analysis. Both the case study's incorrectly implemented rmse function and the implementation the correct RMSE formula were used.

The root mean squared error was calculated for the trajectories i.e. positions of the buses predicted by each model and the ground truth trajectories and the mean RMSE was calculated over $M = 10$ replications.

$$Average\ RMSE = \frac{1}{M}\sum_{k=1}^{M} RMSE$$

## 1.20 Particle Filter Sensitivity Analysis Results

In this section, results of the sensitivity analysis are discussed. For visualisation, one parameter value was fixed, and variation of the other two parameters was shown.

The sensitivity analysis was conducted using the framework used to generate Figure 9 of the case study publication. Therefore, the estimated RMSE values of Figure 9 calculated based on extracted data points can be used as a point of comparison to determine whether the hard-coded values of the filter parameters were the values used in the original case study's experiments.

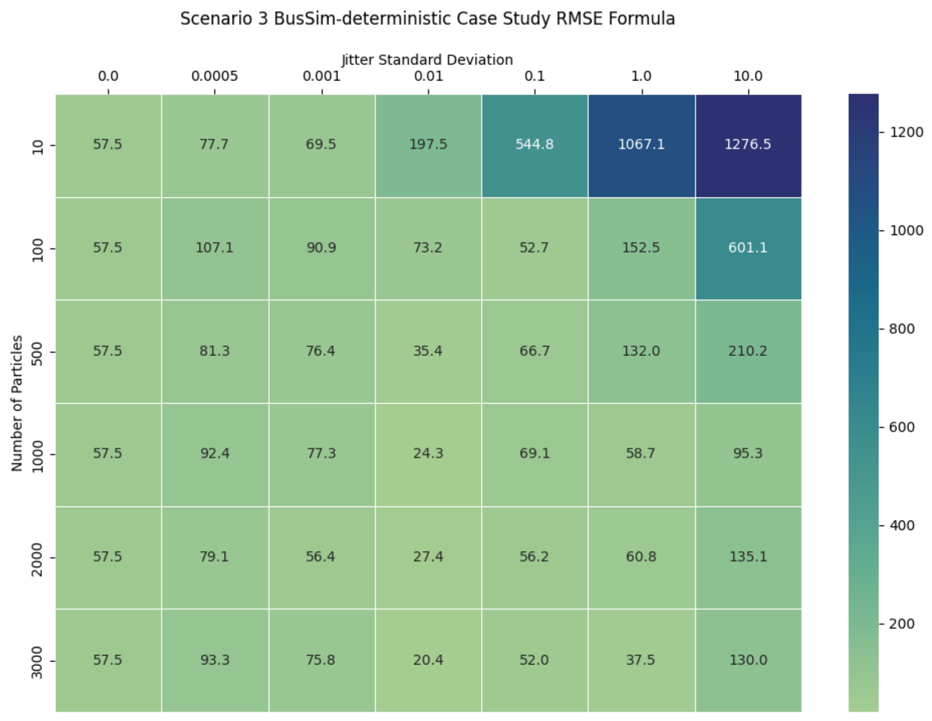
*Table 8: Estimated RMSE values of the case study Figure 9 calculated from datapoints extracted using WebPlotDigitizer*

| Figure | Scenario | Case Study Estimated RMSE BusSim-deterministic | Estimated RMSE BusSim-deterministic | Case Study Estimated RMSE BusSim-stochastic | Estimated RMSE BusSim-stochastic |
|---|---|---|---|---|---|
| 9 | 3 | 51 | 160 | 39 | 178 |

### 1.20.1 Experiment 1 – Varying Number of Particles and Jitter and Resample Window = 1

The figures display heatmaps showing the variation in the RMSE value according to the rmse() function of the case study and the rmse_fixed() function that implemented the correct formula.

#### *1.20.1.1 RMSE results for BusSim-deterministic*



| Number of Particles | 0.0 | 0.0005 | 0.001 | 0.01 | 0.1 | 1.0 | 10.0 |
|---|---|---|---|---|---|---|---|
| 10 | 57.5 | 77.7 | 69.5 | 197.5 | 544.8 | 1067.1 | 1276.5 |
| 100 | 57.5 | 107.1 | 90.9 | 73.2 | 52.7 | 152.5 | 601.1 |
| 500 | 57.5 | 81.3 | 76.4 | 35.4 | 66.7 | 132.0 | 210.2 |
| 1000 | 57.5 | 92.4 | 77.3 | 24.3 | 69.1 | 58.7 | 95.3 |
| 2000 | 57.5 | 79.1 | 56.4 | 27.4 | 56.2 | 60.8 | 135.1 |
| 3000 | 57.5 | 93.3 | 75.8 | 20.4 | 52.0 | 37.5 | 130.0 |

*Figure 17: Average RMSE of 10 replications for BusSim-deterministic predictions under scenario 3 and varying the jitter standard deviation and number of particles. Calculated using case study's rmse() function*

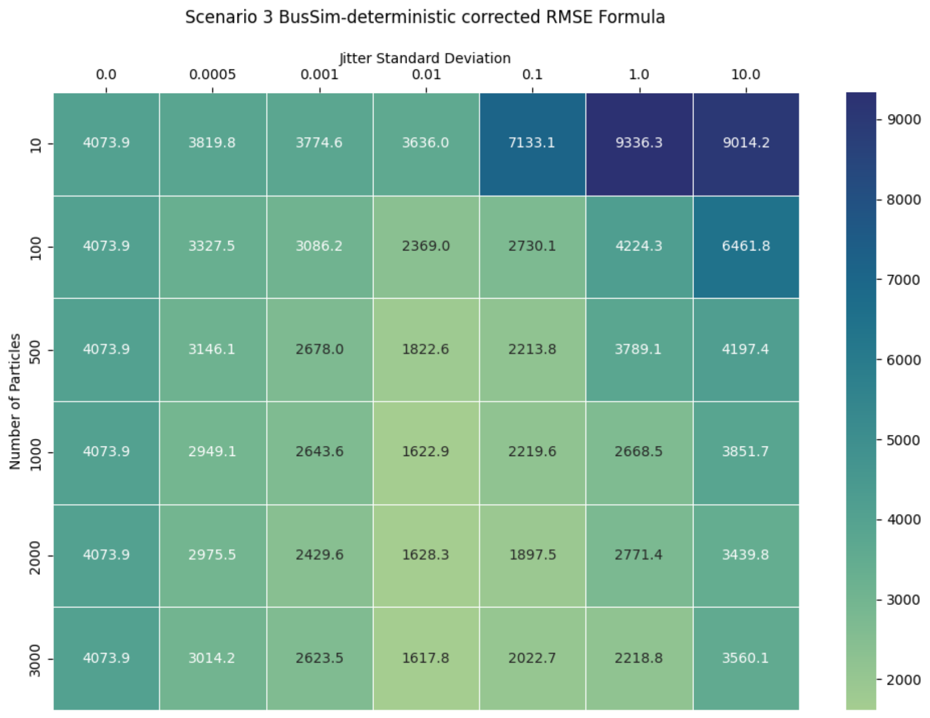


*Figure 18: Average RMSE of 10 replications for BusSim-deterministic predictions under scenario 3 and varying the jitter standard deviation and number of particles. Calculated using correct implementation of RMSE formula*

The RMSE values resulting from the jitter standard deviation = 0 confirm the deterministic nature of BusSim-deterministic. This experiment used the setup of Scenario 3 such that the BusSim-deterministic model had been calibrated before the application of the particle filter algorithm. Without the jitter standard deviation to diversify the particles, all the particles evolve in the same way during the simulation leading to the same output. This output not only confirmed the deterministic nature of BusSim-deterministic but also demonstrated that increasing the number of particles when using a deterministic model did not improve prediction performance. Another observation is that the prediction error was lowest when the jitter standard deviation was 0.01. This value was larger than the case study's hard-coded value of 0.0005 and demonstrated improved performance. By tuning the jitter standard deviation, prediction error can be reduced accordingly. This result was reflected in both the calculation using the case study's incorrectly implemented rmse function and the rmse_fixed function that correctly implemented the RMSE formula. For most of the jitter values used, increasing the number of particles, decreased the prediction error. However, this trend did not consistently apply to larger sets of 3000 particles, at which point the drop in prediction error is less significant or the error increased. Therefore, the use of 500 particles in the case study's code can be considered a practical choice for implementation though the jitter value of 0.0005 that was used could have been selected more carefully to improve predictive accuracy.

### *1.20.1.2 RMSE results for BusSim-stochastic*

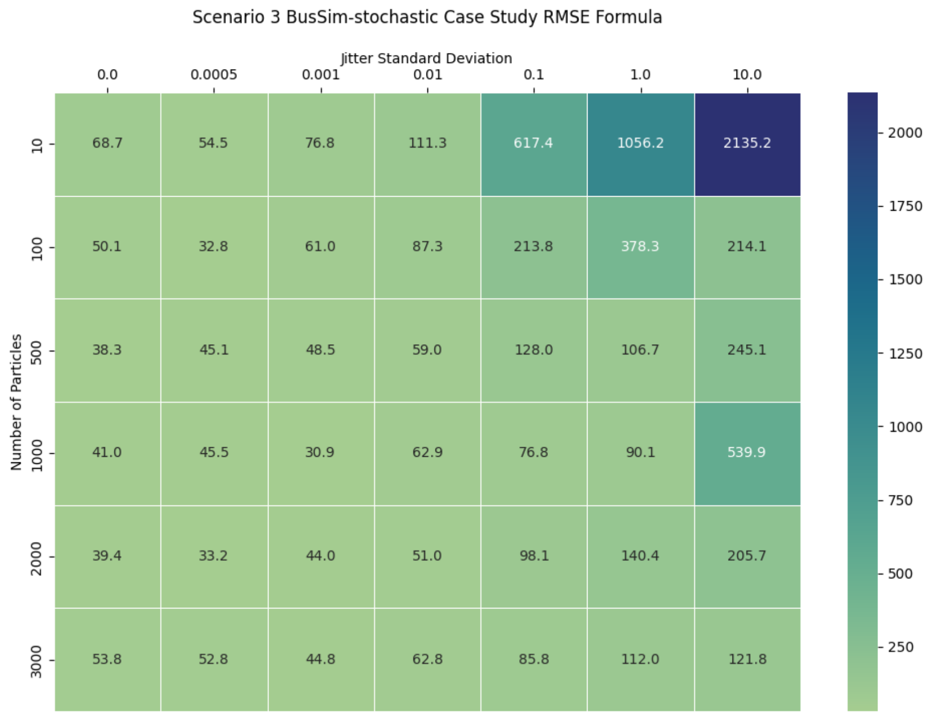


*Figure 19: Average RMSE of 10 replications for BusSim-stochastic predictions under scenario 3 and varying the jitter standard deviation and number of particles. Calculated using case study's rmse() function*

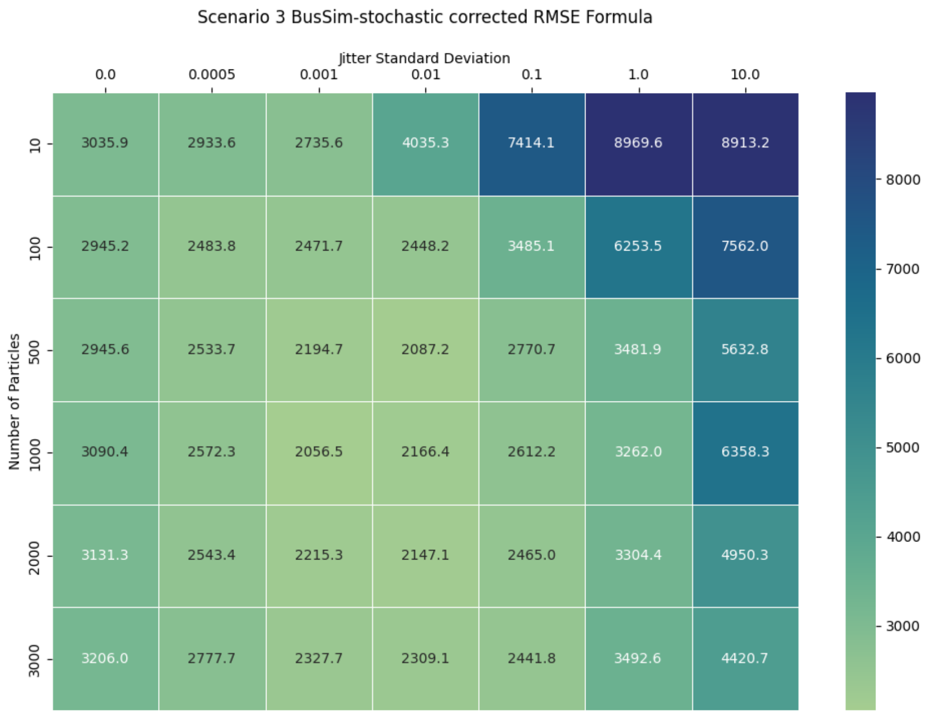


*Figure 20: Average RMSE of 10 replications for BusSim-deterministic predictions under scenario 3 and varying the standard deviation and number of particles. Calculated using correct implementation of RMSE formula*

For BusSim-stochastic, the results using the case study rmse function and the corrected rmse_fixed function differed greatly as compared to the results of BusSim-deterministic discussed previously. The variability of the results may be indicative of the need for more simulation runs to produce a more accurate estimate due to the stochasticity of BusSim-stochastic. The inherent stochasticity of the model can be seen in the variability of the prediction error results when there is no jitter being added to the state vector and demonstrates that this uncertainty in the stochastic model aids in producing more accurate estimations than the BusSim-deterministic model. For most of the jitter values, increasing the number of particles above 2000 did not significantly improve performance. The lowest RMSE value resulted from a combination of 1000 particles and 0.001 jitter standard deviation. The differences between the case study rmse and the corrected rmse_fixed also illustrated the impact on findings due to programming errors like this as the BusSim-stochastic model RMSE values varied greatly between the two formulae.

### 1.20.2 Experiment 2 – Varying Number of Particles and Resample Window; Jitter = 0.0005

The jitter standard deviation was fixed to 0.0005 i.e. the parameter value hard-coded in the case study's code repository. Recall that jittering is only implemented in the algorithm when resampling occurs. Therefore, the resample window parameter also controlled the frequency of adding jitter to the model parameter vector. The results were visualised in the heatmaps below.

### 1.20.2.1 RMSE results for BusSim-deterministic

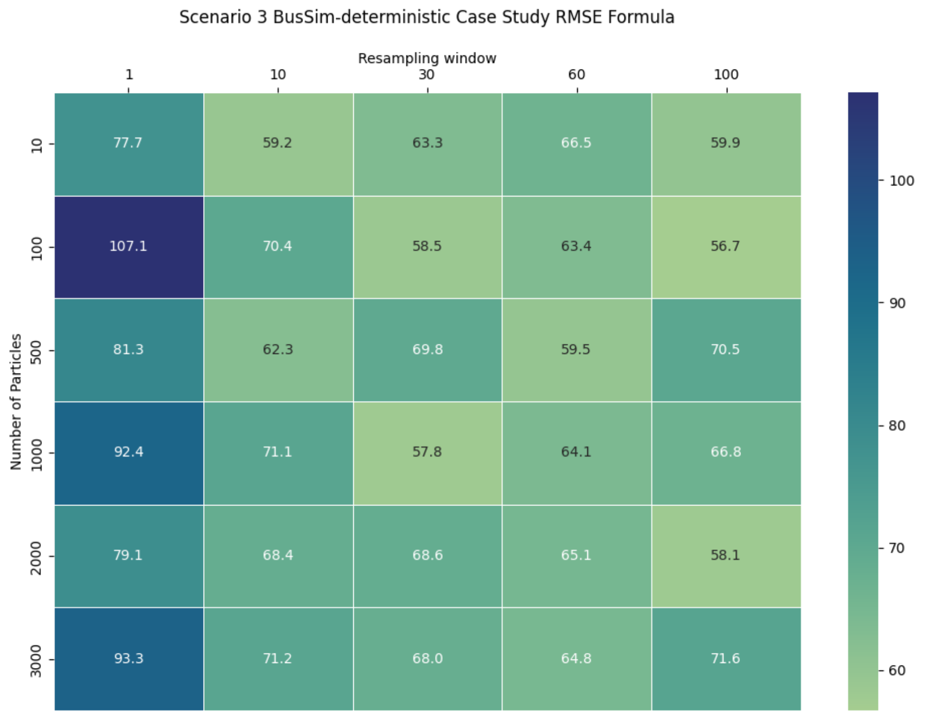


*Figure 21: Average RMSE of 10 replications for BusSim-deterministic predictions under scenario 3 and varying the resampling window and number of particles. Calculated using case study's rmse() function*

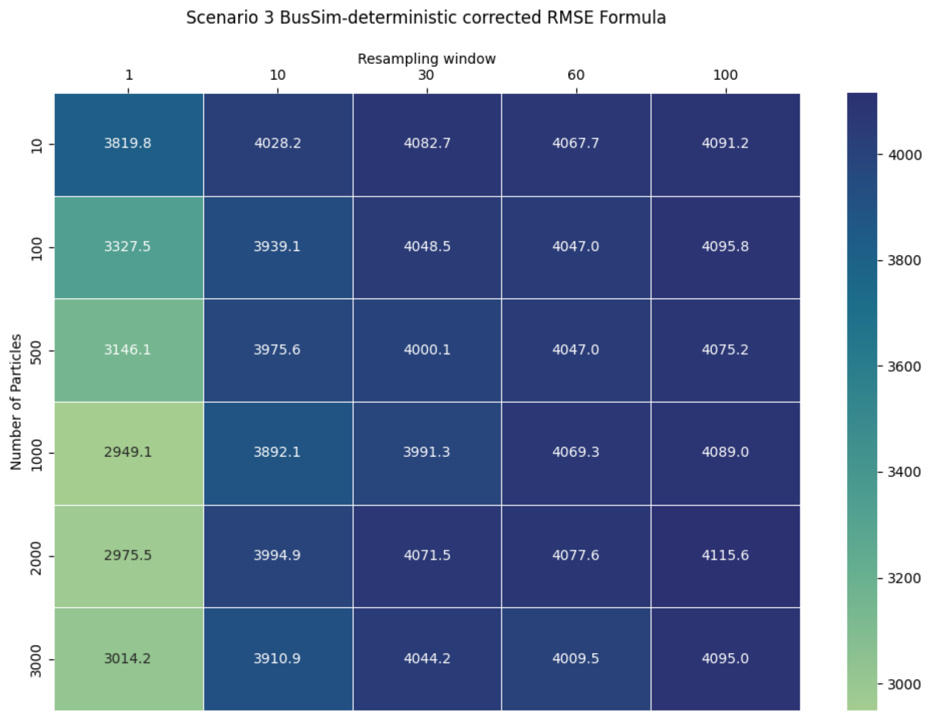


*Figure 22: Average RMSE of 10 replications for BusSim-deterministic predictions under scenario 3 and varying the resampling window and number of particles. Calculated using correct implementation of RMSE formula.*

For BusSim-deterministic, the prediction error increased with the increasing resampling window as seen in the figure using the correct RMSE formula. This observation follows from the previous experiment on the matter of deterministic nature of BusSim-deterministic and how without the jitter to add noise and improve particle diversity, the model was incapable of accurately representing the pdf of the system state vector and differed from the ground truth state. Increasing the resample window resulted in a drastic increase in prediction error, therefore, it was reasonable to assume that the resample window parameter was set to 1 in the case study's original implementation. Furthermore, the results produced using the case study's rmse function vs the rmse_fixed function once again show the unreliability of results calculated using the case study's incorrect rmse function and how it can affect the overall analysis.

### *1.20.2.2 RMSE results for BusSim-stochastic*

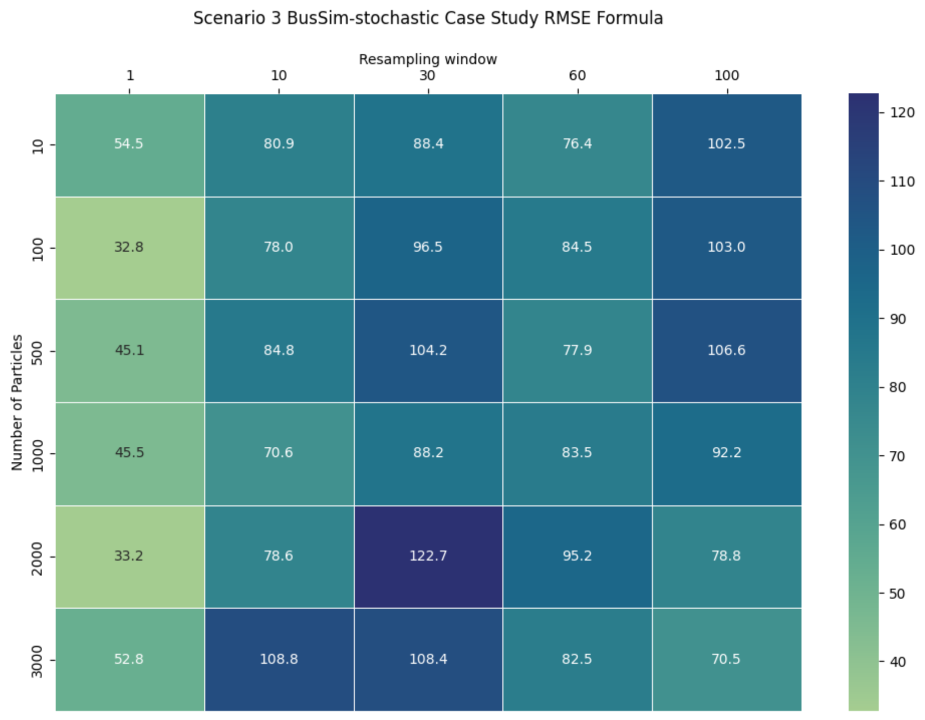


*Figure 23: Average RMSE of 10 replications for BusSim-stochastic predictions under scenario 3 and varying the resampling window and number of particles. Calculated using case study's rmse() function*

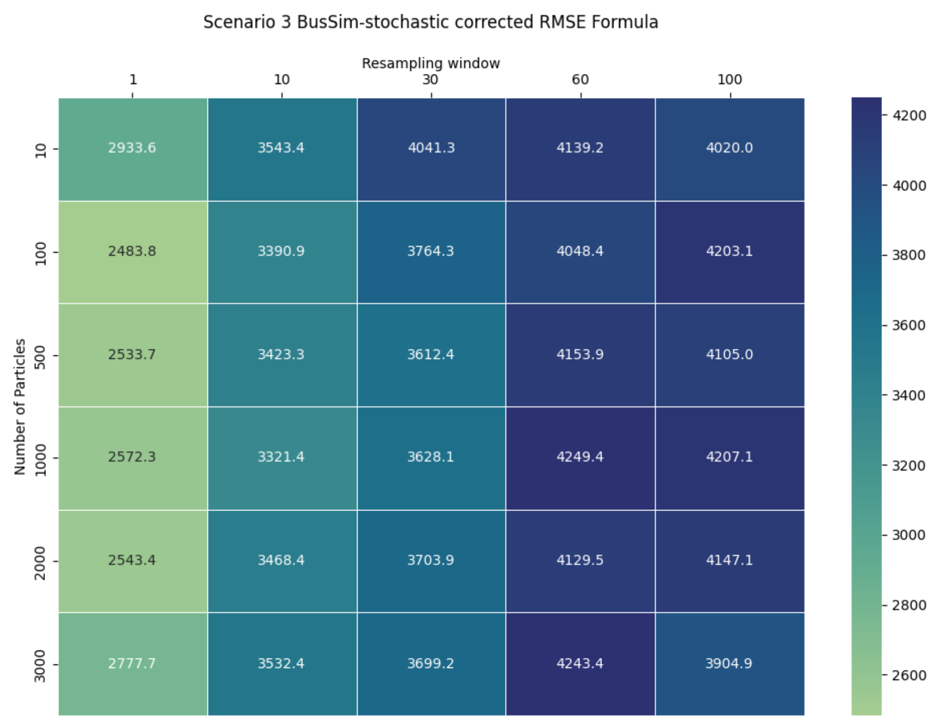


*Figure 24: Average RMSE of 10 replications for BusSim-stochastic predictions under scenario 3 and varying the resampling window and number of particles. Calculated using correct implementation of RMSE formula.*

The BusSim-stochastic model also demonstrated that increasing the resample window produced more inaccurate estimates likely due to particle degeneracy occurring without sufficient use of resampling to mitigate the problem. This was observed in both heatmaps despite the different rmse formulae used. Once again, it was observed that increasing the number of particles used did not guarantee improved estimation accuracy as values of 2000 and 3000 while computationally expensive did not significantly reduce the prediction error.

### 1.20.3 Discussion

The results of these experiments demonstrated that the jitter standard deviation appeared to have a strong influence on the estimation accuracy when using the case study's implemented particle filter. This finding was in agreement with the findings of Malleson et al. (2020) in which the amount of jitter added to the particles was found to be important as an overly large value can cause the particles to deviate greatly from the true underlying system state. Jitter was demonstrated to have a more significant impact on BusSim-deterministic. This was likely due to lack of stochasticity in the model which can be considered equivalent to having small or no process noise being modelled. In such cases of small or no process noise being modelled, the particle filter is not considered to be an appropriate algorithm to apply. BusSim-stochastic with its inherent stochasticity was less influenced by the jitter parameter as shown when the jitter standard deviation was 0.

The number of particles employed had less of an impact than initially expected. The sensitivity analysis proved that there appears to be a threshold at which point increasing the number of particles did not improve estimation accuracy as it increased the prediction error. For particle sets of 500, 1000 or 2000, the prediction error often varied slightly either decreasing or increasing as the number of particles increased. Using 3000 particles was proven to be an impractical choice requiring more computational time and power yet frequently resulting in worsening estimation accuracy. The hard-coded parameter of 500 particles can be considered a suitable choice though 1000 particles would have worked as well with slightly better performance for some filter parameter configurations.

Based on the corrected RMSE calculations, BusSim-deterministic tended to have higher prediction errors than BusSim-stochastic. This would suggest that the stochasticity of the models contribute to the prediction error. This finding contradicted a statement made in the case study's article in which the two static models performed relatively the same under Scenario 2. This could be due to the difference in the calibration settings between the case study's original experiment and the reproducibility experiment. There was ambiguity regarding which model's historical data (BusSim-truth or BusSim-stochastic) should have been used to calibrate the two static models and ultimately, the models had been calibrated to historical data generated by BusSim-stochastic.

Furthermore, the hard-coded filter parameter values of 500 particles, 0.0005 jitter standard deviation and resampling window of 1 timestep, were estimated to have RMSE values of 51 and 39 for BusSim-deterministic and BusSim-stochastic, respectively. These values can be compared to the average RMSE values form the heatmaps calculated using the case study's *rmse()* function, 81.3 for BusSim-deterministic and 45.1 for BusSim-stochastic under the same parameter configuration. Differences between the values generated can be attributed to the low estimation accuracy of the RMSE estimates calculated from extracting the datapoints in addition to differences in the calibration of the static models. This challenge underscored the need the provide the datasets that accompany plots or to make the datasets used in plots numerically reproducible.

Other key parameters that should be investigated in future work include the assimilation window and the weight function used as both parameters are known to be crucial to the performance of the particle filter.

# VII. Final Discussion

This project has conducted a reproducibility experiment for the selected case study with the aim of extracting the key elements that should be documented and reported to facilitate reproducibility of data assimilation for ABMS studies. In this section, the insights gained from the experiments are discussed, summarised and distilled into guidelines for future reproducibility studies and for those conducting data assimilation for ABMS studies who desire to ensure that their work is reproducible.

## 1.21 Challenges for reproduction attempt

In response to the first sub-question, the main challenges encountered in the reproduction process were identified based on the researcher's subjective experience.

> *What challenges arise in attempting to reproduce a data assimilation for agent-based simulation study using PF?*

*Table 9: Summary of Reproducibility Challenges*

| Category | Challenge |
|---|---|
| Code/Packaging errors | • File path errors due to use of absolute paths and not relative paths.<br>• Outdated comments that were never removed. E.g. filenames were changed but comments still refer to old names.<br>• No specification of software dependencies. Lack of suitable requirements.txt or environment.yml file.<br>• Only partial execution pipeline described. The code relies on previously generated files, but the Jupyter notebook provided does not guide through the generation of all the necessary files.<br>• Lack of random seed management to facilitate numerically reproducible results. |
| Publication - Code implementation Inconsistencies | 1. Model formulae and equations described in publication differ from code implementation. For example, the RMSE formula implemented in code did not match the publication.<br>2. Inconsistencies between the textual description and the code implementation.<br>3. Datasets specific to the plots or the results in the publication not included in the codebase |
| Missing/ Ambiguous Specifications | • Insufficient metadata for plot reproduction.<br>• Missing real-time datasets or "pseudo-ground truth" datasets.<br>• Ambiguous documentation of experimental frame used. |

Several of the challenges encountered while attempting to reproduce the case study's results aligned with insights gained from existing literature. The identified challenges in the reproduction process were very similar to the categories of replication problems identified in Axelrod (1997): ambiguity in published descriptions, gaps in the published data and unambiguous yet incorrect published descriptions. Examples from these categories were observed during the reproduction process such as inconsistencies between the textual description and the code implementation.

One of the main hindrances to the reproducibility process was the lack of metadata and computational provenance for each of the reported results. This impeded the reproduction process as there was much ambiguity concerning the experimental frame and parameter settings used to generate each result.

On another note, the nature of data assimilation for ABMS was not adequately accounted for in the codebase of the case study. To reproduce the results and figures exactly would have required the provision of the original ground truth data used to execute the experiments. Alternatively, had the case study initialised and recorded the random seeds used in the experiments, similar datasets to those used in the case study's results could have been regenerated under the identical twin experiment framework that was employed. Had datasets from the real applications been used, it would have been necessary to provide access to those datasets in order to reproduce the results. Additionally, specification of the random seed would not only have contributed to achieving reproducibility of the data assimilation and parameter calibration processes but also would have ensured the repeatability of each simulation run of one of the agent-based models.

During the reproduction process, another major issue was discrepancies between the reported formulae used and the code implementation i.e. there was a gap between what the author reported and what was actually programmed. This appeared mainly in the form of the incorrectly coded RMSE formula *rmse()*. Such implementation mistakes can spark reasonable doubt in the research publication's results. Reproducibility experiments, therefore, prove useful not only to achieve computational reproducibility of results but to verify implementation details which is important for future researchers who wish to reimplement or replicate the methodology for another application.

## 1.22 Parameter Specifications for Particle Filter

*Which properties/parameters of the research methods need to be reported to facilitate reproducibility efforts in data assimilation for agent-based simulation studies?*

By reviewing the implementation of the case study's particle filter algorithm and comparing with other data assimilation for ABMS literature and particle filter theory, the following properties and parameters of the particle filter algorithm were distilled and considered important for the reproduction and potential reuse of the particle filter by future researchers.

*Table 10: Summary of Particle Filter parameters for specification*

| Parameters | Justification for specifying the parameter |
|---|---|
| Number of particles | Indicative of computational resources required for execution and the most important hyper-parameter with direct influence on estimation accuracy(Malleson et al., 2020). |
| Weight function | The weight function is typically selected by the modeller and proportional to the likelihood function. |

| | |
|---|---|
| | The functions used for data assimilation on agent-based models vary across research articles. |
| Process noise | Some studies explicitly model process noise while others consider it negligible.<br>The particle filter is not the most appropriate algorithm for estimation of models with small process noise as if there is no process noise, particle collapse may occur(Gordon et al., 1993). |
| Jittering standard deviation | Impacts the particle variance based on experiments.<br>Important parameter as it is used to mitigate sample impoverishment from resampling. |
| Resample window | Important parameter to mitigate particle degeneracy and increasing particle variance over time. |
| Improved state estimate calculation from the set of particles | It is necessary to describe how the state estimate is obtained from the posterior pdf, for instance, via taking the mean state of all particles or by using the state of the highest weighted particle. (Hu & Wu, 2019) |
| Error evaluation metric | This value should be specified and verified whether RMSE, MSE or L2-norm, SSE, or otherwise. |
| Assimilation window | The time interval dictating when the next observation is assimilated. |
| System state vector | The system state vector needs to be specified as it defines the states of the particles(Hu & Wu, 2019) and the dimensionality of the system. As the dimensionality increases so does the number of particles required to suitably estimate the system state(Malleson et al., 2020). |

The case study reported its main contribution to be the framework of using parameter calibration (CEM) and data assimilation together to improve short-term model forecasts, yet the paper lacked documentation on the implementation of the particle filter and the values of the filter's parameters. While it was possible to conduct the reproductions by assuming that the filter parameter configuration hardcoded in the case study's source code was the same used to produce the published results, this assumption may not always be a suitable particularly in circumstances in which the source code published may not be the same version as the one used to produce the published results.

Table 10 lists key elements of an implemented particle filter that should be considered in future studies. The importance of these parameters are summarised in the table and explained in more detail in the following paragraphs.

**Assimilation Window**

Regarding the implementation of the particle filter in the code, the algorithm's implicit use of an assimilation window of 1 by assimilating an observation at each timestep implied that the parameter could not be varied or experimented with. Due to the importance of the assimilation window as emphasised in literature such as Cho et al. (2020), this parameter should be included as a variable filter parameter in particle filter implementations.

**Number of Particles**

The number of particles used is an important metric to include as generally the more particles used the better the accuracy of the state estimate as well as the higher the computational cost(Hu & Wu, 2019). However, from the sensitivity analysis experiment, it was shown for the case study, there seemed to be a limit to this increase in performance because for the highest numbers of particles, the reduction in prediction error was either minimal or did not occur as the error increased instead due to worse estimates. Therefore, the choice of the number of particles should be practically motivated such that estimation accuracy is sufficient for the respective problem and also reasonable to implement regarding computational cost.

**Process Noise**

The modelling of the process noise in the dynamical model is of importance and can improve the quality of the data assimilation(Wang & Hu, 2015). The researcher must choose how best to model the process noise. In Wang and Hu (2015), the process noise is modelled by adding noise to the state vectors of the particles after re-sampling such that the particles re-sampled from the same parent particle can have states that differ from each other. Contrary to this idea, Lueck et al. (2019) assume process noise is negligible in their implementation and they rely on the inherent uncertainty in the agent-based model. It should be noted that the particle filter is not a suitable algorithm in the case when the process noise is zero. Furthermore, if the process noise is very small, particle collapse will occur in a few iterations due to the resampling step reducing the diversity of the particles(Arulampalam et al., 2002). The effect of small or no process noise in the system model was demonstrated in the earlier sensitivity analysis in which the prediction error of BusSim-deterministic was evaluated without any jittering. Without uncertainty or noise in the system model, the prediction error increased, and the particles were not an accurate approximation of the posterior pdf of the system state variable.

**Performance Evaluation Metric**

The formula of the metric used for the prediction error is significant because as shown in the case study, it can be implemented incorrectly, and it is important to know whether the entire state vector or only a subset of the state vector is being evaluated. In the case study, the RMSE was calculated using only the bus trajectories which represent a subset of the system state vector $X_t$.

**Weight Function**

The weights of the particle filter are usually updated by the likelihood function but more generally can be updated using any importance function(Gustafsson et al., 2002). In the reviewed literature, the definition of the likelihood function used for the weight calculation varies across research articles applying the particle filter to agent-based models(Kreuger & Osgood, 2015; Malleson et al., 2020; Wang & Hu, 2015). Additionally, the weight function and the method for extracting the state estimate from the posterior pdf of the system state variable should be implemented using a modular approach such that alternative functions can be employed to achieve better state estimates.

**Measurement Model and Measurement Noise**

The measurement or observation model is a key part of the data assimilation problem setup and as such, should be stated explicitly. The observation equation and its statistical equivalent the likelihood function impact the weighting of the particle filter as the weights of the particles are proportional to the likelihood function. The measurement model defines the mapping between the system state vector and the observation data. The model should be formulated based on the sensors used to collect the data. The measurement noise is also an important part of the model because the more accurate the observation data (i.e. the less noise) the better the estimation results.(Hu & Wu, 2019). The system state vector of the ABMs in the case study  was well-described in the article, however, the observation model was not formally specified but only qualitatively described. Moreover, the case study did not specify whether any observation noise was added which was also reflected in the code.

**Further considerations**

To reproduce and evaluate the efficacy of the data assimilation method, it was necessary to fix other components in the experiment: the dynamic model used (BusSim-stochastic and BusSim-deterministic) and the 'real-world' model producing the ground truth data (BusSim-truth). Therefore, in addition to specifying the features and parameters of the filtering algorithm used, it is also necessary to specify the experimental frame employed for the data assimilation experiments. For example, it was observed that Figure 9 of the case study, did not specify the values for the maxDemand and dynamic change rate parameters used and therefore lacked information on the specific model configuration and datasets used for Scenario 3 in which both parameter calibration and data assimilation were applied.

Apart from this, a key element is to identify whether the data assimilation for ABMS study used collected sensor data from the real system or an identical twin framework. The real-time data used in the original study needs to be made available whether from the real system or generated from a simulation model. Alternatively, the model run used to generate the real-time data could be setup to reproduce the same data during each simulation run. The availability of this real-time data dictates the level of reproducibility of the overall study.

In Figure 25, a diagram is shown containing the various elements that should be specified for a study applying the particle filter to an ABM and the data that should be specified and stored in such research to ensure reproducibility of results.

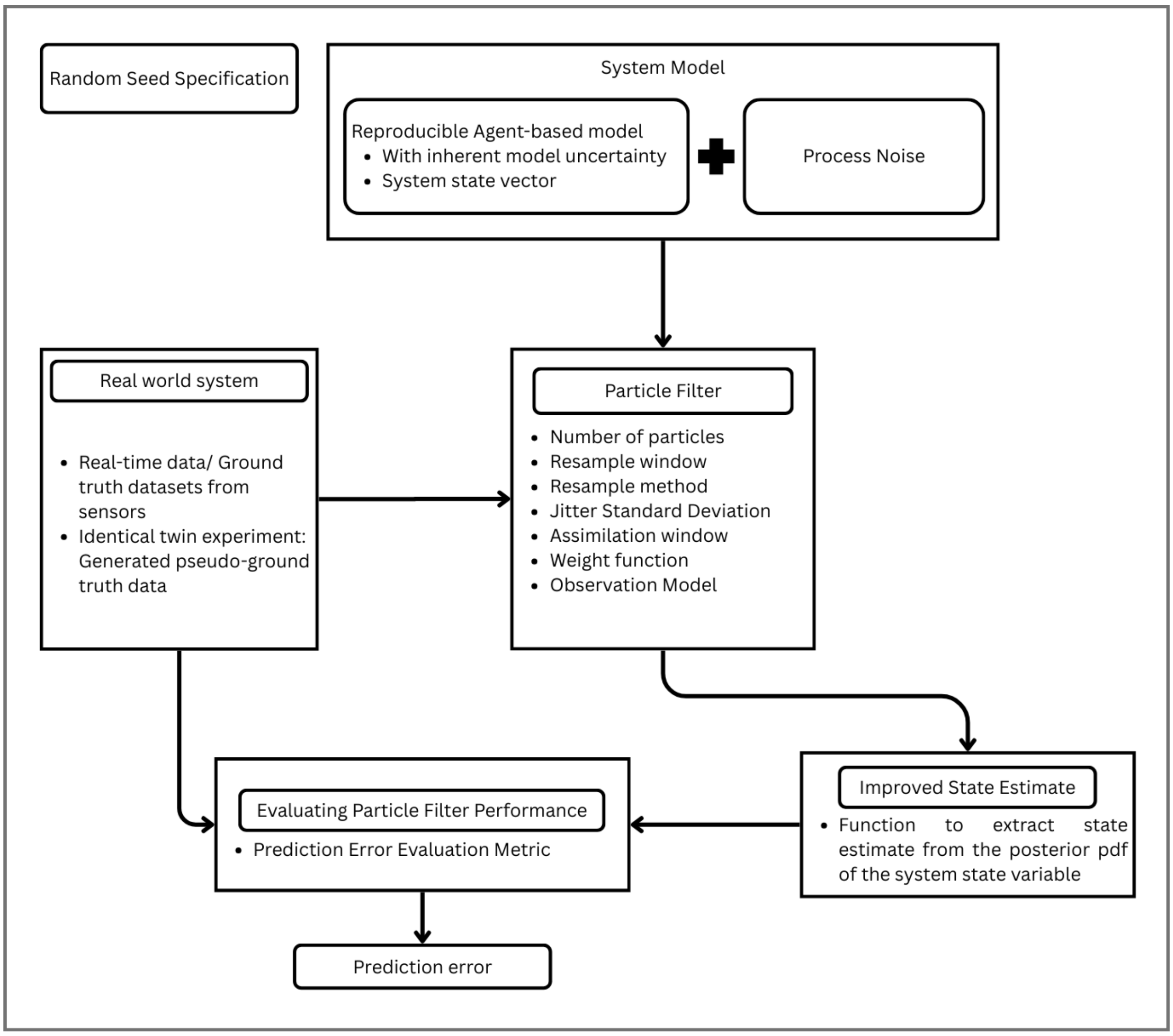

Random Seed Specification
System Model
Reproducible Agent-based model
• With inherent model uncertainty
• System state vector
Process Noise
Real world system
• Real-time data/ Ground truth datasets from sensors
• Identical twin experiment: Generated pseudo-ground truth data
Particle Filter
• Number of particles
• Resample window
• Resample method
• Jitter Standard Deviation
• Assimilation window
• Weight function
• Observation Model
Evaluating Particle Filter Performance
• Prediction Error Evaluation Metric
Improved State Estimate
• Function to extract state estimate from the posterior pdf of the system state variable
Prediction error

*Figure 25: Diagram showing the generalised elements and parameters of data assimilation for ABMS study that require specification to facilitate reproducibility of results.*

## 1.23 Guidelines for reproduction process

*What procedure can be used to guide future reproducibility efforts in data assimilation for agent-based simulation studies?*

In Table 11, general steps have been drafted to guide future reproducibility studies. These guidelines are based on the experience of the researcher during the reproduction of the case study as well as insights gathered from the relevant literature.

*Table 11: Protocol for Reproducibility Experiments*

| | Steps | Description |
|---|---|---|
| 1 | Gather all relevant materials from the selected research study for the reproducibility attempt | This would include the research article, supporting materials, all digital artefacts such as archived code and data. Any resources provided by the author as part of the research compendium. |
| 2 | Ascertain the software requirements/dependencies and hardware requirements for the study if any. | Descriptions of software and hardware requirements may be included in the publication by the author or be included in a README or other form of code documentation.<br><br>Determine whether the reproducibility attempt is feasible based on the accessibility of the required software and the hardware resources available to the reproduction researcher. |
| 3 | Determine the aim, general experimental setup and methodology of the study | Review the research article and the research compendium and note the general steps of the study to be followed. |
| 4 | Select the results from the research article to reproduce | Results can take different forms including figures/plots, numerical data in tables, qualitative descriptions or conclusions drawn from the study |
| 5 | Map or trace the workflow required to produce each of the results and formulate as a diagram. | For each result, attempt to trace which datasets, functions and parameters were used to generate it. This requires reviewing both the descriptions in the publication as well as the code provided. Note any processes or parameters for which there is no information or where the steps used to create the result are ambiguous.<br><br>Check for the use of literate programming for example in the form of Jupyter notebooks that may provide benchmark examples explaining the workflow used or which may already reproduce results from the publication.<br><br>In the case of missing or ambiguous information, the reproduction researcher may wish to contact the author of the paper.<br>Alternatively, assumptions can be made based on reasonable justifications and noted for future reference. |

| | | |
|---|---|---|
| 6 | Assess the reproducibility level of the results using a reproducibility framework from literature. | Using the map created in the previous step, determine the expected degree of reproducibility of the result based on the specificity of the workflow and the information that is missing. |
| 7 | Select potential metrics or measures that could be used to assess the reproducibility of each result. | Statistical analyses can be used for numerical results depending on the information available. Subjective assessments may be required for qualitative results.<br><br>Note the number of simulations (i.e. sample size) used in the estimation of results in the original study and how this might affect the choice of metric. |
| 8 | Recreate the results | Execute the code required to recreate the results and store the appropriate information required for later analysis.<br><br>Keep a log of all errors encountered, steps taken to resolve them and assumptions made.<br><br>Depending on the quality of code available, this step will require possible debugging. |
| 9 | Evaluate reproduced results using suitable metrics | Evaluate the reproduction using the previously selected metrics. Note any discrepancies found. |
| 10 | Repeat steps 7-9 until satisfactory results have been obtained or sufficient effort has been expended in pursuit of reproduced results. | The metric chosen may not be the most suitable depending on the power of the original study or the reproduced study.<br><br>Debug and identify the causes of the discrepancies. Resolve them if possible and record them otherwise if they impede the reproduction process. |
| 11 | Include a reflection or subjective assessment on the reproduction process. | Note limitations of both the original study and the reproduction attempt. Reflect on assumptions made and any ambiguities found. |

The following questions in Table 122 can be used to collect the required information needed to formulate the case study's computational workflow similar to Figure 3. The questions in this table can aid in step 5 of the reproducibility protocol. See the Appendix A: Experiment Log for an example of these questions applied to the case study for the process of reproducing figures and tables.

*Table 12: Guiding questions for deducing computational workflows*

| Questions | Description |
|---|---|
| What is the result as described in the publication? | For a graph, what is plotted on the axes?<br>For numerical results, this could refer to a calculated metric. |
| What is the interpretation/conclusion drawn from the result? | List the conclusions or insights drawn from the result by the author in the original study. |
| Where is the code to generate this result located? | Has the code to generate this result been provided? If the code to generate this result is provided, list the files or functions used to produce the result. |
| How is this result generated/defined in the code? | Review the source code and/or comments for the description of the result and check whether it agrees with the description in the publication. |
| Which datasets or files are required to reproduce this result? | Determine which datasets are required to produce the result. |
| Which model is used to produce the data/results? | If the dataset was generated by the code, determine which model or process was used. If the dataset required is external to the project, check whether it was included in the relevant materials. |
| What are the parameter settings required to produce the results using the model? | List the parameters of the model used to generate the datasets. Any assumptions made are justified and recorded. |

# VIII.Conclusion

This thesis project has investigated the reproducibility of a published simulation study in the field of data assimilation for ABMS.

From this attempt, challenges characteristic of the reproduction process were identified. These challenges were aligned with the difficulties typically experienced by other researchers conducting reproducibility studies and/or replication studies which were explained in the earlier chapters. The errors can largely be divided into issues of code quality and software dependencies; ambiguous or missing specifications regarding the methodology and inconsistencies between textual descriptions and implemented code which is similar to the categories derived from the replication experience in Axelrod (1997). Another challenge was the assessment of a successful reproduction. The use of a single quantitative metric such as the confidence interval had several limitations. Another avenue for assessing reproducibility apart from metrics is the use of frameworks for classifying the experiments. This is a more accessible means of assessing reproducibility especially for researchers interested in ensuring the reproducibility of their own work by others as the emphasis is in the communication and transparency of the research rather than the correctness of the study. Though this does not detract from the importance of proper simulation study design. Statistical metrics may be more suited for validating the adequacy and statistical power of study's design in the case of replication whereas the frameworks focus on evaluating only the level of documentation and reporting which are indicative of the degree of result reproducibility that can be achieved.

Further to these insights, a protocol was derived from the reproduction attempt to provide guidance to future researchers attempting to reproduce other scientific works. The protocol along with the challenges can aid other researchers in the execution of future reproducibility studies and provides insights as to what to expect when navigating published code repositories.

The particle filter algorithm employed in the study was also tested by conducting a sensitivity analysis for key parameters and comparing the implementation to existing literature. Though this part of the study was limited in the range of its exploration, it helped to investigate the influence of the parameters: number of particles, jitter standard deviation and the resampling window on the estimation accuracy of the results. Based on this analysis and that of existing literature, some key properties of the particle filter that should be specified in future studies were identified. Apart from this, elements required to reproduce data assimilation for ABMS research were also indicated including the specifying the experimental frame and whether an identical twin framework is being used.

These insights into data assimilation for ABMS resulting from this thesis can be considered a step towards encouraging reproducible and reusable research in this field such that both researchers interested in conducting reproducibility studies and those interested in ensuring their research is reproducible can benefit from this project's findings.

Note that this study has only sought to verify the computational workflow and implementation of the case study and not to validate its methodology. It should be noted that computational reproducibility is a basic requirement for science and that higher levels of such as replicability should also be considered in one's research.

## 1.24 Limitations

- Firstly, the findings of this thesis project were based on a single reproducibility attempt. To test the generalisability and applicability of the proposed protocol would require further reproducibility experiments to be conducted.
- There is always a bias in reproducibility studies related to the level of expertise of the researcher who attempted the reproduction. This must be accounted for when assessing the results of this project.
- The author of the case study was not contacted during the reproducibility study. While it is not always necessary to contact the author if sufficient details and code artefacts are available, many code review processes rely on open communication between the author and reviewer for the code review for journals. However, this study was more than a code review and the reviewer also took on the role of an independent researcher who wished to not only reproduce but understand the method proposed in the original paper for potential future applications.
- Many reproducibility guidelines and those that promote open science practices, have advocated for the registration of reproducibility studies by publishing the study protocol prior to engaging in the research. This was not done for this study.
- Limited computational power and time was available to increase the number of simulation replications and reduce the Monte Carlo Standard Error.
- Extracting data from figures using WebPlotDigitizer has been found to have a small margin of error(Burda et al., 2017) which would have affected the accuracy of the RMSE estimates.

## 1.25 Recommendations/Future work

An important topic of future research is how to assess the results of a reproducibility study. Many researchers have published studies on this subject whether the methods be statistical analyses or subjective in nature and based on expert opinion. Works such as Gundersen (2021) also call for the research community to question and define what exactly is meant by the 'result' that replication studies try to compare. The most suitable metric to use for a reproducibility study depends on the result being reproduced and the amount of data and information available in the original publication. A possible future work could involve extensive testing of a variety of statistical measures for assessing reproducibility to better understand the properties best captured by each metric and potentially derive guidelines for selecting a diverse set of metrics for a reproducibility experiment such as was done for transportation metrics in Jafino et al. (2020).

On the subject of data assimilation for ABMS, an expanded sensitivity analysis exploring the choice of weight function, the assimilation window, measurement noise and other parameters not directly investigated in this thesis would be useful to better understand how critical these choices are to the accuracy of the algorithm when applied to ABMS.

## 1.26 Reflection

Research in data assimilation applied to ABMS required an understanding of not only the data assimilation algorithm but the ABM being used. The stochasticity of both elements needed to be considered. It was initially thought that the ABM could be thought of as a black-box model because the data assimilation algorithm did not impact the model structure. However, during the reproduction process, it was necessary to understand the model in order to debug the errors that were encountered as well as to understand the parameters included in the state vector.

Reproducing the case study showed that even research publications with seemingly organised and detailed source code and text description can contain mistakes and lack information that can be easily missed. One such example was the incorrectly implemented *rmse* function.

While this project has focussed on verification and the computational reproducibility of published research, there are other perspectives on the issue which have not been touched upon. Reproducibility is seen as a standard requirement of scientific research, but replicability, reusability, robustness and generalisability of the research are all higher standards for which repeatability and reproducibility are necessary requirements. To ensure reproducibility, many journals and in response researchers are providing code and data to accompany their research publications, however, there are some disagreements on this point.

The opinion paper by Miłkowski et al. (2018) discusses that in computational studies, model validation efforts are more valuable than model verification. The author states that many researchers fall into the trap of writing too much which sometimes results in giving too much detail to some aspects and too little in others. Miłkowski et al. (2018) proposed that research papers should contain only the information needed to recreate the model and assess its relationship to the intended target. The article differentiates between the purpose of publication and the purpose of code repositories in that repositories facilitate code reuse and reproducibility whereas publication serves a different function from code.

By taking this perspective, one could argue that reproducibility is not the requirement of the research article but of the archived digital artefacts that support the research. Therefore, standards can be created for the publication of research articles which would differ from the publication or release of the associated software artefacts in repositories.

On another note, the research process is not linear workflow but an iterative one and a learning process. This makes it difficult for researchers under pressure to publish or meet deadlines capable of ensuring their work is reproducible. Given the difficulty in assessing reproducibility using metrics, the current approach of code review processes to leave the decision of reproducible or not up to the code reviewer is a sensible decision. The promotion of the use of reproducibility frameworks in such reviews would be helpful to identify the level of reproducibility to be expected from independent researchers interested in reusing or reapplying published methods. In doing this, publications could be given a credibility or reusability score like a citation score depending on the number of researchers able to independently adapt or reproduce a particular research paper.

# References

Aarts, A., Anderson, J., Anderson, C., Attridge, P., Attwood, A., Axt, J., Babel, M., Bahník, Š., Baranski, E., Barnett-Cowan, M., Bartmess, E., Beer, J., Bell, R., Bentley, H., Beyan, L., Binion, G., Borsboom, D., Bosch, A., Bosco, F., & Pen, M. (2015). Estimating the reproducibility of psychological science. *Science*, *349*. https://doi.org/10.1126/science.aac4716

Arulampalam, M. S., Maskell, S., Gordon, N., & Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, *50*(2), 174-188. https://doi.org/10.1109/78.978374

Axelrod, R. (1997). Advancing the Art of Simulation in the Social Sciences. In R. Conte, R. Hegselmann, & P. Terna, *Simulating Social Phenomena* Berlin, Heidelberg.

Axtell, R., Axelrod, R., Epstein, J. M., & Cohen, M. D. (1996). Aligning simulation models: A case study and results. *Computational & Mathematical Organization Theory*, *1*(2), 123-141. https://doi.org/10.1007/BF01299065

Burda, B. U., O'Connor, E. A., Webber, E. M., Redmond, N., & Perdue, L. A. (2017). Estimating data from figures with a Web-based program: Considerations for a systematic review. *Research Synthesis Methods*, *8*(3), 258-262. https://doi.org/https://doi.org/10.1002/jrsm.1232

Chirigati, F. S., Troyer, M., Shasha, D. E., & Freire, J. (2013). A Computational Reproducibility Benchmark. *IEEE Data Eng. Bull.*, *36*(4), 54-59.

Cho, Y., Huang, Y., & Verbraeck, A. (2020). Strategic Use of Data Assimilation for Dynamic Data-Driven Simulation. In V. V. Krzhizhanovskaya, G. Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos, & J. Teixeira, *Computational Science – ICCS 2020* Cham.

Dalle, O. (2012, 9-12 Dec. 2012). On reproducibility and traceability of simulations. Proceedings of the 2012 Winter Simulation Conference (WSC),

Davidson, S. B., & Freire, J. (2008). *Provenance and scientific workflows: challenges and opportunities* Proceedings of the 2008 ACM SIGMOD international conference on Management of data, Vancouver, Canada. https://doi.org/10.1145/1376616.1376772

Doucet, A., De Freitas, N., & Gordon, N. (2001). *Sequential Monte Carlo methods in practice*. Springer.

Edmonds, B., & Hales, D. (2003). Replication, Replication and Replication: Some hard lessons from model alignment. *Jasss-The Journal Of Artificial Societies And Social Simulation*, *6*, U227-U253.

Edmonds, B., & Hales, D. (2005). Computational Simulation as Theoretical Experiment. *The Journal of Mathematical Sociology*, *29*(3), 209-232. https://doi.org/10.1080/00222500590921283

Eglen, S., & Nüst, D. (2019). CODECHECK: An open-science initiative to facilitate sharing of computer programs and results presented in scientific publications. *Septentrio Conference Series*(1). https://doi.org/10.7557/5.4910

Epskamp, S. (2019). Reproducibility and Replicability in a Fast-Paced Methodological World. *Advances in Methods and Practices in Psychological Science*, *2*(2), 145-155. https://doi.org/10.1177/2515245919847421

Evensen, G., Vossepoel, F. C., & Van Leeuwen, P. J. (2022). *Data assimilation fundamentals: A unified formulation of the state and parameter estimation problem*. Springer Nature.

Fitzpatrick, B. G. (2019). Issues in Reproducible Simulation Research. *Bulletin of mathematical biology*, *81(1)*, 1–6. https://doi.org/https://doi.org/10.1007/s11538-018-0496-1

Freire, J., & Chirigati, F. (2018). Provenance and the different flavors of computational reproducibility. *IEEE Data Engineering Bulletin*, *41*(1), 15.

Galán, J. M., & Izquierdo, L. (2005). Appearances Can Be Deceiving: Lessons Learned Re-Implementing Axelrod's 'Evolutionary Approach to Norms'. *Journal of Artificial Societies and Social Simulation*, *8*.

Ghorbani, A., Ghorbani, V., Nazari-Heris, M., & Asadi, S. (2023). Data Assimilation for Agent-Based Models. *Mathematics*, *11*(20).

Gordon, N. J., Salmond, D. J., & Smith, A. F. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE proceedings F (radar and signal processing),

Grazzini, J., Richiardi, M. G., & Tsionas, M. (2017). Bayesian estimation of agent-based models. *Journal Of Economic Dynamics & Control*, *77*, 26-47. https://doi.org/10.1016/j.jedc.2017.01.014

Gundersen, O. E. (2021). The fundamental principles of reproducibility. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *379*(2197), 20200210. https://doi.org/doi:10.1098/rsta.2020.0210

Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson, R., & Nordlund, P.-J. (2002). Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, *50*(2), 425-437.

Hernandez, J. A., & Colom, M. (2025). Reproducible research policies and software/data management in scientific computing journals: a survey, discussion, and perspectives [Original Research]. *Frontiers in Computer Science*, *Volume 6 - 2024*. https://doi.org/10.3389/fcomp.2024.1491823

Heroux, M. A. (2015). Editorial: ACM TOMS Replicated Computational Results Initiative. *ACM Trans. Math. Softw.*, *41*(3), Article 13. https://doi.org/10.1145/2743015

Heyard, R. *A scoping review on metrics to quantify reproducibility*. Retrieved 19 July 2025 from http://rachelhey.github.io/reproducibility_metrics/

Heyard, R., Pawel, S., Frese, J., Voelkl, B., Würbel, H., McCann, S., Held, L., Wever, K. E., Hartmann, H., Townsin, L., & Zellers, S. (2025). A scoping review on metrics to quantify reproducibility: a multitude of questions leads to a multitude of metrics. *Royal Society Open Science*, *12*(7), 242076. https://doi.org/doi:10.1098/rsos.242076

Hu, X., & Wu, P. (2019). A Data Assimilation Framework for Discrete Event Simulations. *ACM Trans. Model. Comput. Simul.*, *29*(3), Article 17. https://doi.org/10.1145/3301502

Jafino, B. A., Kwakkel, J., & Verbraeck, A. (2020). Transport network criticality metrics: a comparative analysis and a guideline for selection. *Transport Reviews*, *40*(2), 241-264. https://doi.org/10.1080/01441647.2019.1703843

Kieu, L.-M., Malleson, N., & Heppenstall, A. (2020). Dealing with uncertainty in agent-based models for short-term predictions. *Royal Society Open Science*, *7*(1), 191074. https://doi.org/10.1098/rsos.191074

Kreuger, K., & Osgood, N. (2015, 6-9 Dec. 2015). Particle filtering using agent-based transmission models. 2015 Winter Simulation Conference (WSC),

Liu, J. S. (2001). *Monte Carlo strategies in scientific computing*. Springer.

Lueck, J., Rife, J. H., Swarup, S., & Uddin, N. (2019, 8-11 Dec. 2019). Who goes there? Using an agent-based simulation for tracking population movement. 2019 Winter Simulation Conference (WSC),

Luijken, K., Lohmann, A., Alter, U., Claramunt Gonzalez, J., Clouth, F. J., Fossum, J. L., Hesen, L., Huizing, A. H. J., Ketelaar, J., Montoya, A. K., Nab, L., Nijman, R. C. C., Penning de Vries, B. B. L., Tibbe, T. D., Wang, Y. A., & Groenwold, R. H. H. (2024). Replicability of simulation studies for the investigation of statistical methods: the RepliSims project. *Royal Society Open Science*, *11*(1), 231003. https://doi.org/10.1098/rsos.231003

Malleson, N. (2018). *Data Assimilation for Agent-Based Modelling*. https://urban-analytics.github.io/dust/index.html

Malleson, N., Minors, K., Kieu, L.-M., Ward, J. A., West, A., & Heppenstall, A. (2020). Simulating Crowds in Real Time with Agent-Based Modelling and a Particle Filter. *Journal of Artificial Societies and Social Simulation*, *23*(3), 3. https://doi.org/10.18564/jasss.4266

McLean, T., & Fujimoto, R. (2000, 28-31 May 2000). Repeatability in real-time distributed simulation executions. Proceedings Fourteenth Workshop on Parallel and Distributed Simulation,

Miłkowski, M., Hensel, W. M., & Hohol, M. (2018). Replicability or reproducibility? On the replication crisis in computational neuroscience and sharing only relevant detail. *Journal of Computational Neuroscience*, *45*(3), 163-172. https://doi.org/10.1007/s10827-018-0702-z

Monti, C., Pangallo, M., De Francisci Morales, G., & Bonchi, F. (2023). On learning agent-based models from data. *Scientific Reports*, *13*(1), 9268. https://doi.org/10.1038/s41598-023-35536-3

Morris, T. P., White, I. R., & Crowther, M. J. (2019). Using simulation studies to evaluate statistical methods. *Statistics in Medicine*, *38*(11), 2074-2102. https://doi.org/https://doi.org/10.1002/sim.8086

Muradchanian, J., Hoekstra, R., Kiers, H., & van Ravenzwaaij, D. (2021). How best to quantify replication success? A simulation study on the comparison of replication success metrics. *Royal Society Open Science*, *8*(5), 201697. https://doi.org/doi:10.1098/rsos.201697

Murata, R., & Tanaka, K. (2025). Dynamic Estimation of Customer Movements by Agent-Based Simulation with Particle Filter. In P. Mathieu & F. De la Prieta, *Advances in Practical Applications of Agents, Multi-Agent Systems, and Digital Twins: The PAAMS Collection* Cham.

National Academies of Sciences, E., & Medicine. (2019). *Reproducibility and Replicability in Science*. The National Academies Press. https://doi.org/doi:10.17226/25303

Nüst, D., & Eglen, S. (2021). CODECHECK: an Open Science initiative for the independent execution of computations underlying research articles during peer review to improve reproducibility [version 2; peer review: 2 approved]. *F1000Research*, *10*(253). https://doi.org/10.12688/f1000research.51738.2

Oswald, Y., Malleson, N., & Suchak, K. (2024). An Agent-Based Model of the 2020 International Policy Diffusion in Response to the COVID-19 Pandemic with Particle Filter [Article]. *JASSS*, *27*(2), Article 3. https://doi.org/10.18564/jasss.5342

Plavén-Sigray, P., Matheson, G. J., Schiffler, B. C., & Thompson, W. H. (2017). The readability of scientific texts is decreasing over time. *eLife*, *6*, e27725. https://doi.org/10.7554/eLife.27725

Plesser, H. E. (2018). Reproducibility vs. Replicability: A Brief History of a Confused Terminology [Opinion]. *Frontiers in Neuroinformatics*, *Volume 11 - 2017*. https://doi.org/10.3389/fninf.2017.00076

Raghupathi, W., Raghupathi, V., & Ren, J. (2022). Reproducibility in Computing Research: An Empirical Study. *IEEE Access*, *10*, 29207-29223. https://doi.org/10.1109/ACCESS.2022.3158675

Rai, S., & Hu, X. (2013). Behavior pattern detection for data assimilation in agent-based simulation of smart environments. Proceedings - 2013 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2013,

Rohatgi, A. *WebPlotDigitizer*. In (Version 5.2) https://automeris.io

Stodden, V., Guo, P., & Ma, Z. (2013). Toward Reproducible Computational Research: An Empirical Analysis of Data and Code Policy Adoption by Journals. *PLoS ONE*, *8*(6), e67111. https://doi.org/10.1371/journal.pone.0067111

Stodden, V., Seiler, J., & Ma, Z. (2018). An empirical analysis of journal policy effectiveness for computational reproducibility. *Proceedings of the National Academy of Sciences*, *115*(11), 2584-2589. https://doi.org/doi:10.1073/pnas.1708290115

Supporting computational reproducibility through code review. (2021). *Nature Human Behaviour*, *5*(8), 965-966. https://doi.org/10.1038/s41562-021-01190-w

Tang, D., & Malleson, N. (2022). DATA ASSIMILATION WITH AGENT-BASED MODELS USING MARKOV CHAIN SAMPLING [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2205.01616

Taylor, S. J. E., Anagnostou, A., Fabiyi, A., Currie, C., Monks, T., Barbera, R., & Becker, B. (2017, 3-6 Dec. 2017). Open science: Approaches and benefits for modeling & simulation. 2017 Winter Simulation Conference (WSC),

Taylor, S. J. E., Khan, A., Morse, K. L., Tolk, A., Yilmaz, L., Zander, J., & Mosterman, P. J. (2015). Grand challenges for modeling and simulation: simulation everywhere—from cyberinfrastructure to clouds to citizens. *SIMULATION*, *91*(7), 648-665. https://doi.org/10.1177/0037549715590594

Trisovic, A., Durbin, P., Schlatter, T., Durand, G., Barbosa, S., Brooke, D., & Crosas, M. (2020). *Advancing Computational Reproducibility in the Dataverse Data Repository Platform* Proceedings of the 3rd International Workshop on Practical Reproducible Evaluation of Computer Systems, Stockholm, Sweden. https://doi.org/10.1145/3391800.3398173

van Leeuwen, P. J. (2015). Nonlinear Data Assimilation for high-dimensional systems. In *Nonlinear Data Assimilation* (pp. 1-73). Springer International Publishing. https://doi.org/10.1007/978-3-319-18347-3_1

Wang, M., & Hu, X. (2015). Data assimilation in agent based simulation of smart environments using particle filters. *Simulation Modelling Practice and Theory*, *56*, 36-54. https://doi.org/https://doi.org/10.1016/j.simpat.2015.05.001

Wikle, C. K., & Berliner, L. M. (2007). A Bayesian tutorial for data assimilation. *Physica D: Nonlinear Phenomena*, *230*(1), 1-16. https://doi.org/https://doi.org/10.1016/j.physd.2006.09.017

Wilensky, U., & Rand, W. (2007). Making Models Match: Replicating an Agent-Based Model. *Journal of Artificial Societies and Social Simulation*, *10*(4), 2. https://www.jasss.org/10/4/2.html

Williams, C., Yang, Y., Lagisz, M., Morrison, K., Ricolfi, L., Warton, D. I., & Nakagawa, S. (2024). Transparent reporting items for simulation studies evaluating statistical methods: Foundations for reproducibility and reliability. *Methods in Ecology and Evolution*, *n/a*(n/a). https://doi.org/https://doi.org/10.1111/2041-210X.14415

Yilmaz, L. (2011). Reproducibility in modeling and simulation research. *SIMULATION*, *87*(1-2), 3-4. https://doi.org/10.1177/0037549710387316

Yilmaz, L., & Ören, T. (2013). Toward Replicability-Aware Modeling and Simulation: Changing the Conduct of M&S in the Information Age. In A. Tolk (Ed.), *Ontology, Epistemology, and Teleology for Modeling and Simulation: Philosophical Foundations for Intelligent M&S Applications* (pp. 207-226). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-31140-6_11

Zhang, J., & Robinson, D. T. (2021). Replication of an agent-based model using the Replication Standard. *Environmental Modelling & Software*, *139*, 105016. https://doi.org/https://doi.org/10.1016/j.envsoft.2021.105016

# Appendix A: Experiment Log

17/04/2025

Preamble:

After much experimentation and debugging, the case study code now produces plots that are visually similar to the plots from the publication base on face validation. Details of this process can be found in the change/log and README.md file of the github repository.

I will now proceed to list a series of experimental checks to be done to confirm reproducibility of the results.

- Experiment 1a: Reproducing Publication Results
    - maxDemand low passenger high passenger plot max Demand= 0.5% and maxDemand = 2%
    - Dynamic change rate plot 1% and 10%
    - This will require produce plots for each scenario with setting maxDemand = 2 and dynamic change rate = 7%.
- Experiment 1b: Reproducing Publication Results (10 replications per scenario experiment and average RMSE values reported)
    - Producing the sensitivity analysis RMSE results
    - maxDemand range(0.5, 4.5, 0.5)
    - dynamic change rate range(0,17.5,2.5)
- Experiment 2: Using the codebase with the original data and calibration files to reproduce the results from the paper. (Essentially using doing experiment 1 using the data files present. Recall that BuSim_experiments.ipynb does not regenerate certain data files which are included in the Data/ and required for further calculations).
- Experiment 3: Reproducing the data provided in the codebase download i.e. the original data and calibration files.

## 1.27 Experiment 1

### 1.27.1 Figure 5

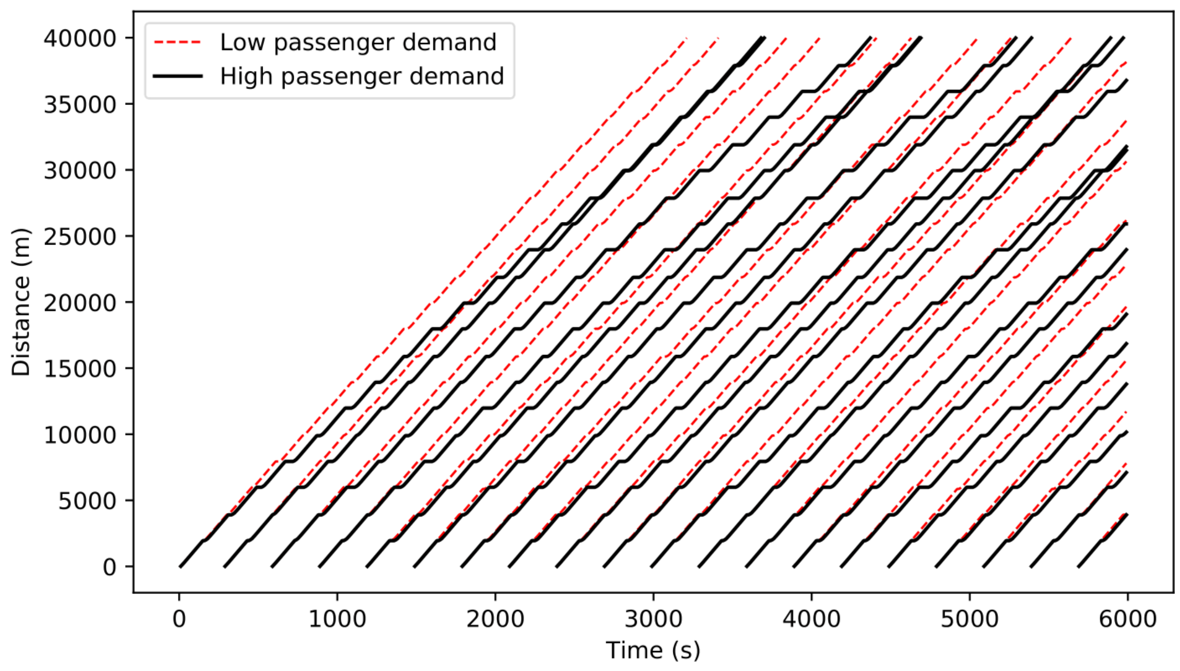


*Figure 26: Original Figure 5 Plot: Synthetic bus GPS trajectory at low and high passenger demand. Red, dashed lines are bus trajectories when maxDemand equals 0.5, while black, colid lines re bus trajectories when maxDemand equals 2*

*Figure description from publication:*

Figure 5 gives insight into the differences in bus trajectories when maxDemand=0.5 and 2. When maxDemand=minDemand=0.5, BusSim-truth reduces to a deterministic model.

*Interpretation of the Figure from publication:*

As maxDemand increases there are more delays for individual buses and it is less likely that a stable headway can be maintained between the buses.

*Where is the code to generate this plot found?*

The code to generate this plot is found in BusSim_stochastic.py instead of BusSim_truth.py. The codebase appears to have been set up to use BusSim-stochastic for this plot using maxDemand = 0.5 and maxDemand = 3. The tester has extracted the plot generation code from the BusSim_stochastic.py and adapted it for use with BusSim-truth model as indicated by the research publication.

*What is plotted on the figure according to the code?*

The code plots variables x and t. X refers to the trajectory of each bus trajectory refers to a list storing bus positions after each model timestep.

*Which data files are required to reproduce this figure?*

No data files are required to produce this figure as it is generated by running BusSim-truth.

*Which model is used to produce the necessary files?*

According to the publication, BusSim-truth. However, the codebase was setup to use BusSim-stochastic.

*What are the parameter settings required for the model to produce the data files?*

maxDemand = 0.5

maxDemand = 2

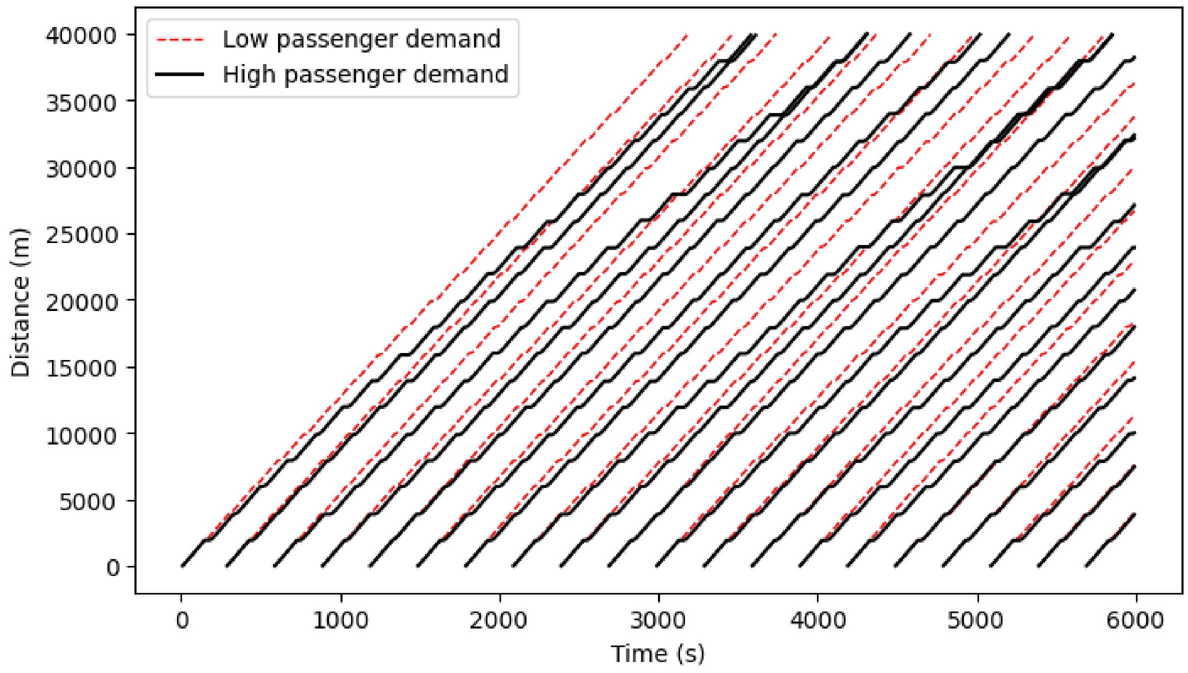
Unspecified parameters required to produce plots:

| Unspecified parameters | Assumed value for testing |
|---|---|
| dynamic change rate, ξ<br>( variable name: *IncreaseRate* ) | 1 |

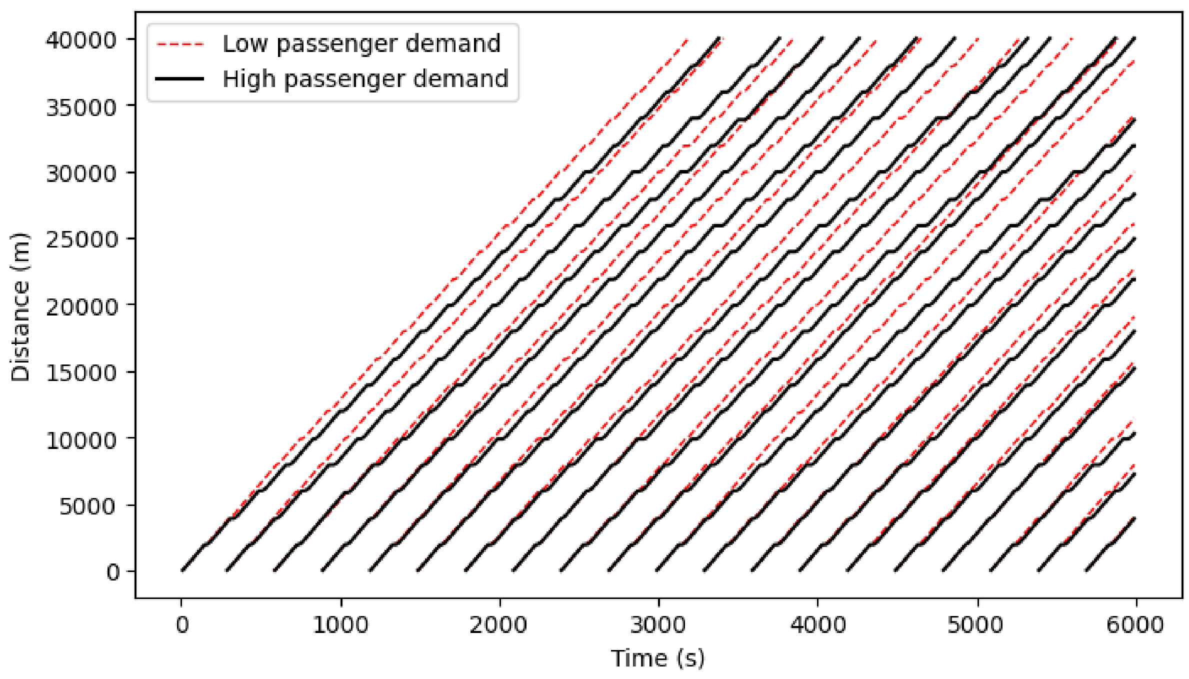
*Note: If BusSim-stochastic were used to create this figure as indicated by the code, no dynamic change rate parameter would be required because the BusSim-stochastic is a static model.*

Reproduced Figure

Reproduced figure 5 using BusSim-truth



Reproduced figure 5 using BusSim-stochastic



### 1.27.2 Figure 6

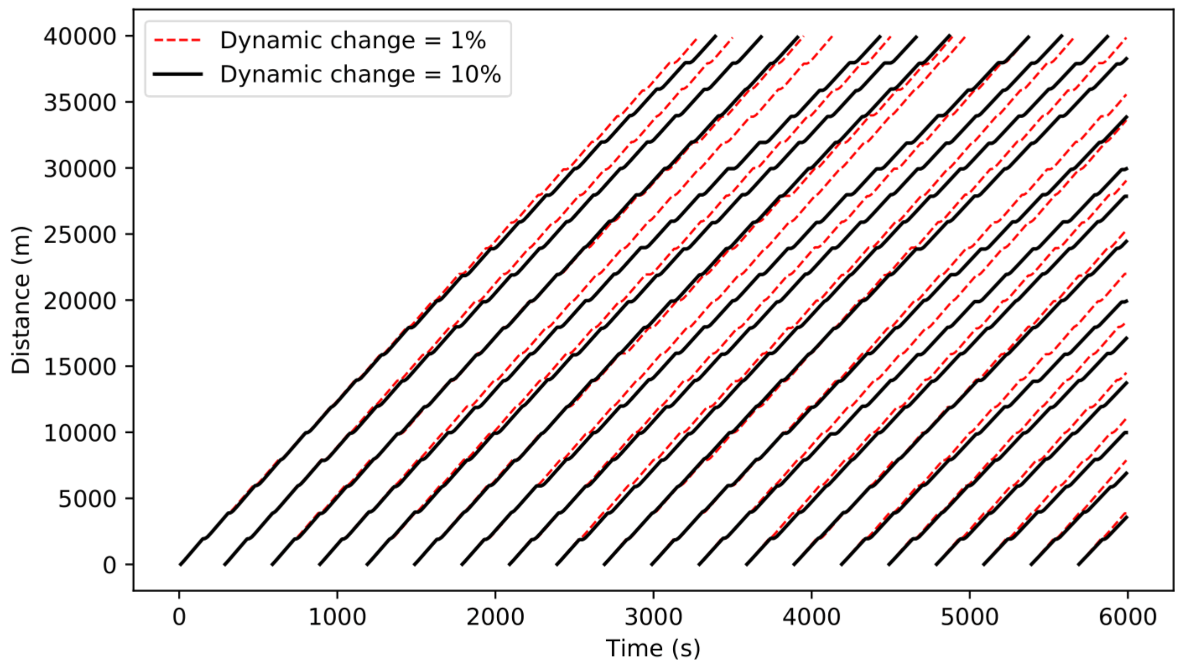


*Figure 27: Original Figure 6 plot: Synthetic bus GPS trajectory with two different values of ξ.*

*Figure description from publication:*

Dynamic change rate parameter, ξ, is equal to 1% and 10%.

*Interpretation of the Figure from publication:*

Because arrival rate and traffic speed gradually change, there is little change in bus trajectories of BusSim-truth with $\xi = 1\%$ and $\xi = 10\%$. As time passes there are more delays for BusSim-truth $\xi = 10\%$ because there are more passengers (higher arrival rate) and the buses are travelling slower (lower traffic speed).

*Where is the code to generate this plot found?*

Code to generate figure 6 appears to be located in the main function of BusSim_truth.py.

*What is plotted on the figure according to the code?*

The code plots variables x and t. x refers to the trajectory of each bus. trajectory refers to a list storing bus positions after each model timestep.

*Which data files are required to reproduce this figure?*

No data files are required for this process because they are generated by the model.

*Which model is used to produce the necessary files?*

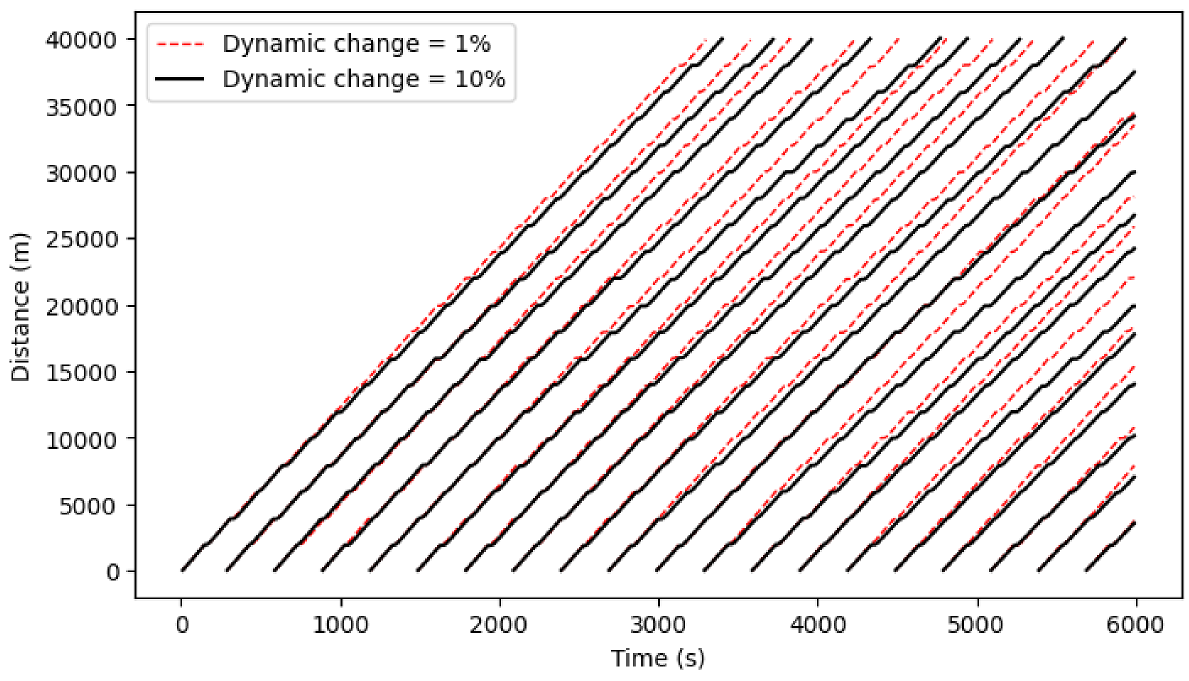
BusSim-truth is used.

*What are the parameter settings required for the model to produce the data files?*

$\xi = 1\%$

$\xi = 10\%$

| Unspecified parameters | Assumed value | Comments |
|---|---|---|
| maxDemand | 1 | This value and others were set in the code but not specified in the publication explicitly. As no changes were made to the code that produces this plot the parameters set in the code are assumed to be correct. |



Reproduced Figure 6

### 1.27.3 Figure 7

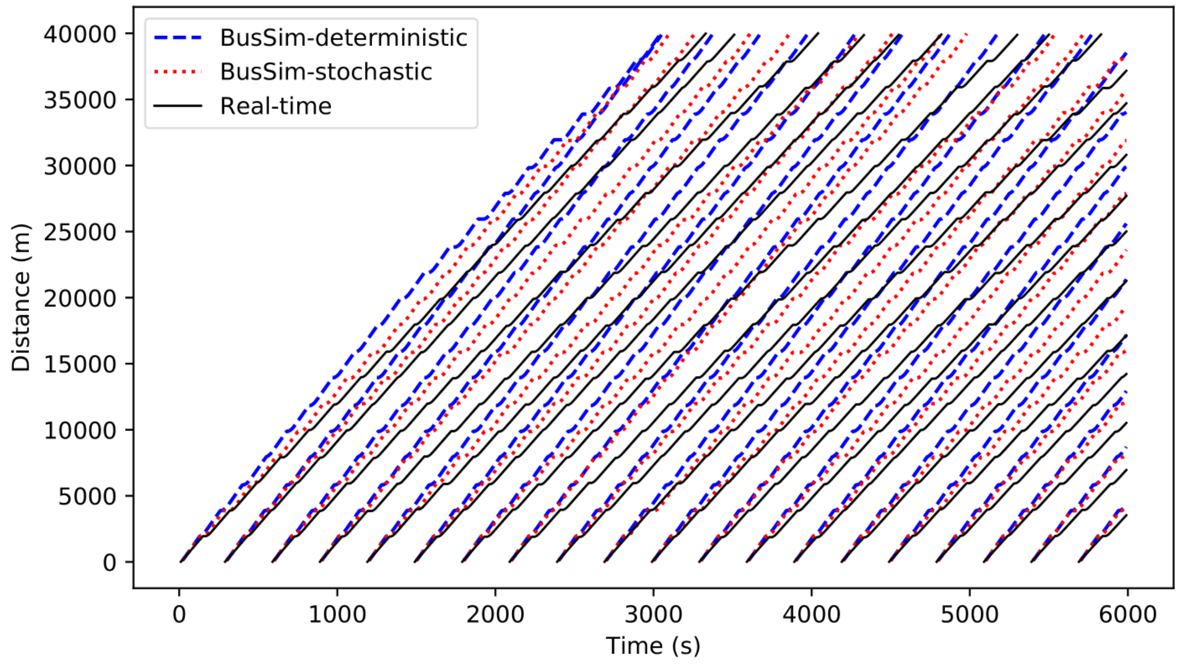


*Figure 28: Original Figure 7 Plot: Prediction results from scenario 1: no calibration*

*Figure description from publication:*

Scenario 1: no calibration (benchmark)

An example of prediction results where maxDemand = 2 and $\xi = 7\%$.

*Interpretation of the Figure from publication:*

Both static models poorly predict the trajectories of the 'real' buses. Gaps between the real trajectories widen as the buses operate ( as the distance and time increase). Models are shown to diverge from reality which is expected because the models do not have optimal parameters to capture the bus route operations.

*Where is the code to generate this plot found?*

Code to generate this plot adapted from A02_doing_nothing_analysis.IncreaseRate_analysis() method.

*What is plotted on the figure according to the code?*
The bus trajectories for all the models.

$$x, t\ , x2, t2, x3, t3$$

*Which data files are required to reproduce this figure?*

BusSim-truth needs to be executed beforehand to create Real-time_data_IncreaseRate_X.pkl files. This is then read into memory and compared to BusSim-deterministic and BusSim-stochastic which a re executed using random parameters to generate bus trajectories.

To create this graph, the Realtime_data_IncreaseRate_',str(IncreaseRate),'.pkl' files generated by BusSim_truth are required. This is first done by executing the main function of BusSim_truth.py and setting 'do_data_export_realtime' = True . Realtime_data_increaseRate_XX.pkl was generated.

*Which model is used to produce the necessary files?*

BusSim-truth produces files beforehand.

BusSim-deterministic an BusSim-stochastic

*What are the parameter settings required for the model to produce the data files?*

BusSim-deterministic and BusSim-stochastic are implemented using random parameters based on equation 4.1 and 4.1 of the publication for arrival rate and departure rate which are generated from uniform distributions. (m refers to bus stops)

$$Arr_m = U(minDemand, maxDeamnd)\ , m = 1, \dots, M$$

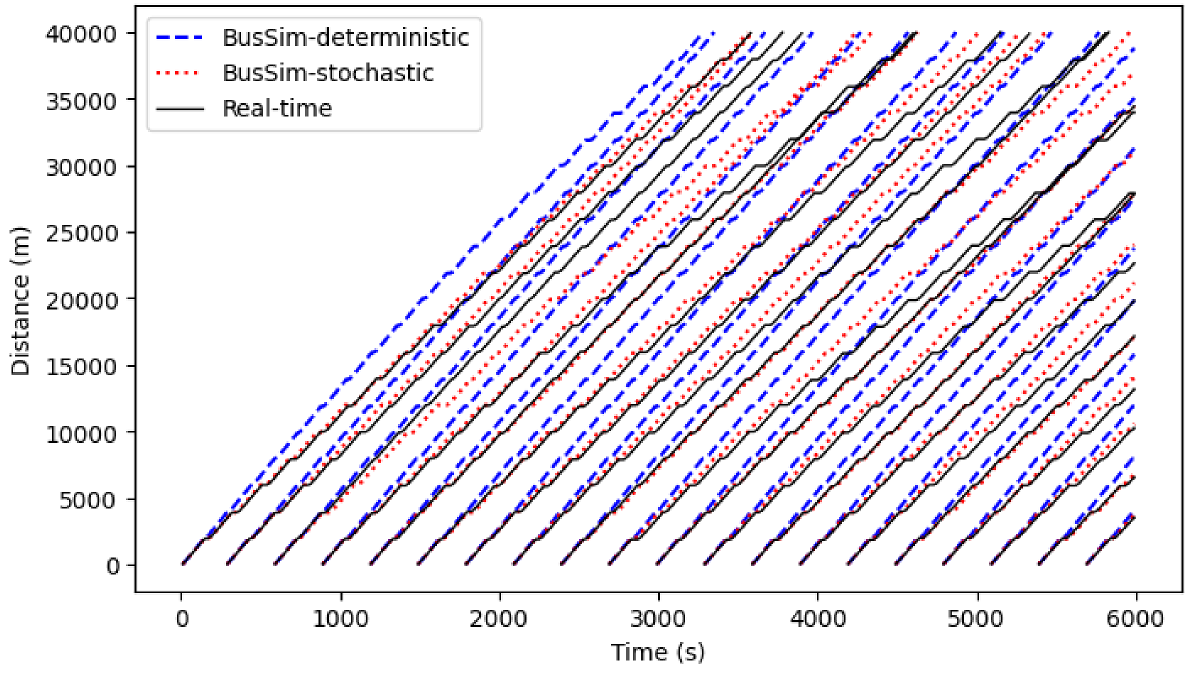
$$Dep_m = ordered\big(U(0.05{,}0.5)\big) and\ Dep_M = 1, \qquad m = 1, \dots, M$$

BusSim-truth produces synthetic real-time trajectories to which the static models' trajectories are compared.

maxDemand = 2

dynamic change rate $\xi\ = 7\%$

Reproduced Figure 7



### 1.27.4 Figure 8

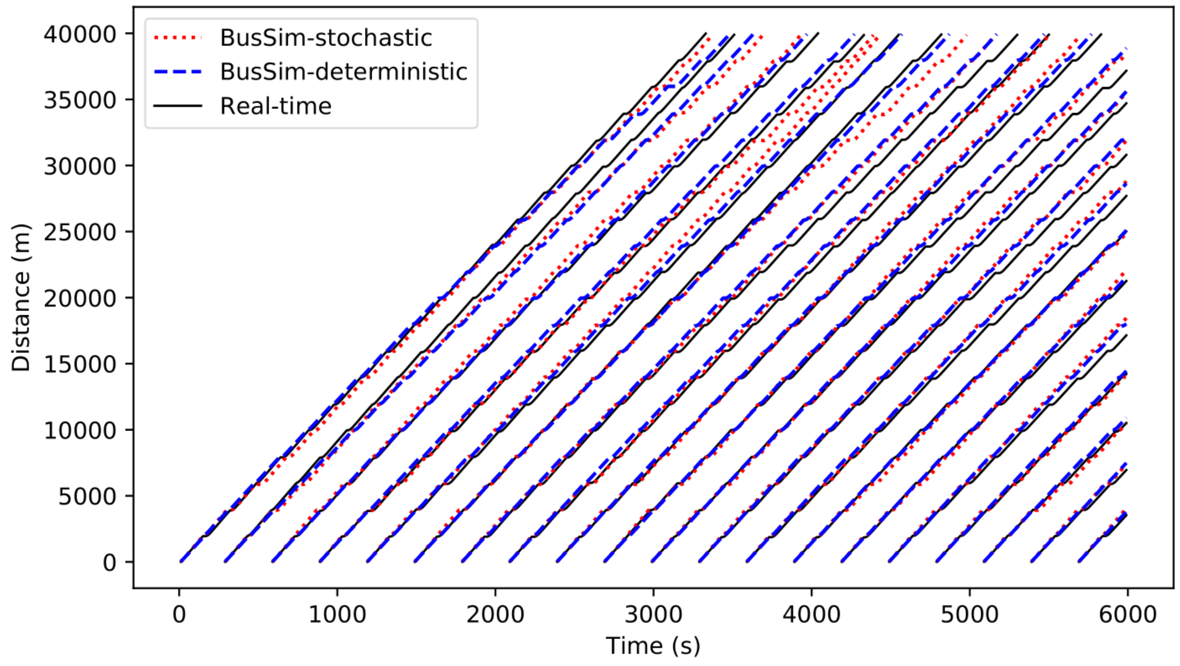


*Figure 29: Original Figure 8 Plot: prediction results from scenario 2: parameter calibration*

*Figure description from publication:*

Figure shows an example of the comparison between BusSim-deterministic and BusSim-stochastic versus synthetic 'real-time'GPS data where maxDemand = 2 and $\xi = 7\%$.

*Interpretation of the Figure from publication:*

Both models outperform models in scenario 1. During 1st quarter of the route when distance is less than 1000, there are few visible gaps between the predicted trajectories and synthetic real-time data. However, increasing divergence is still a problem. Gaps between predicted and synthetic trajectories widen by time. This is because BusSim-stochastic and BusSim-deterministic are static models i.e. their model states do not change over time, whereas the synthetic data comes from a dynamic system. At $\xi$=7%, both passenger demand and the traffic speed are changing rapidly. BusSim-stochastic shows no improvement in prediction performance in comparison to BusSim-deterministic. This shows that the changing system state is the main source of the prediction error, not the deterministic or stochastic nature of the models.

*Where is the code to generate this plot found?*

A03_calibration.py

*What is plotted on the figure according to the code?*

The bus trajectories generated by BusSim-stochastic and BusSim-deterministic versus the synthetic real-time trajectories generated by the BusSim-truth.

Variables: x, t, x2, t2, x3, t3

*Which data files are required to reproduce this figure?*

Synthetic historical data is required to create this figure. The research paper states that the models are calibrated against historical data. It does not state that the historical data should be generated by BusSim_truth. However, this can be assumed based on the specification of $\xi = 7\%$ as the static models do not have this parameter. Moreover, it aligns with the setup of the study in which synthetic GPS data is generated by BusSim-truth for the purpose of parameter calibration and particle filtering. The data files for the setting maxDemand =2 and $\xi = 7\%$ do not exist among the provided data files in the codebase.

BusSim_experiments.ipynb also states that the static models are calibrated against data generated by BusSim-truth. However, the codebase upon downloading it from the repository was set up such that model calibration was done using the data files, *"./Data/Historical_data_static_maxDemand_"*. These files are read by the unpickle_initiation function of BusSim_model_calibration.py which contains the code for model calibration. According to the comments in BusSim_model_calibration.py, the Historical_static_maxDemand data files are supposed to be generated using BusSim_static_v2.py which does not exist but can be assumed to refer to BusSim_stochastic.py. If the data were generated using one of the static models then there is no IncreaseRate parameter to deal with. This generates BusSim_ModelX_calibration_static_maxDemand data files.

On another note, the analysis of the performance of the calibrated static models versus synthetic real-time data is executed by code in the script A03_calibration.py. The setup of this code was such that it calls upon provided data files BusSim_ModelX_Calibration_IncreaseRate_XX . There were no code excerpts in the codebase that were setup to produce BusSim_ModelX_Calibration_IncreaseRate_XX files by calibrating models using Historical_Data_IncreaseRate_XX files which were provided in the code download and can be generated as well by the code.

*Which model is used to produce the necessary files?*

BusSim-truth used to produce synthetic historical data for calibration and real-time data for performance analysis. The real time data file required for this plot was produced during the generation of previous plot Figure 7 used in scenario 1 benchmarking. This file will be used for the generation of Figure 8.

BusSim-deterministic and BusSim-stochastic are used by the code in BusSim_model_calibration.py to generate BusSim_Model1_Calibration_XX and BusSim_Model2_calibration_XX files. These static models then executed in A03_calibration.py using the calibrated parameter settings to generate bus trajectories and for performance analysis against synthetic real-time data.

*What are the parameter settings required for the model to produce the data files?*

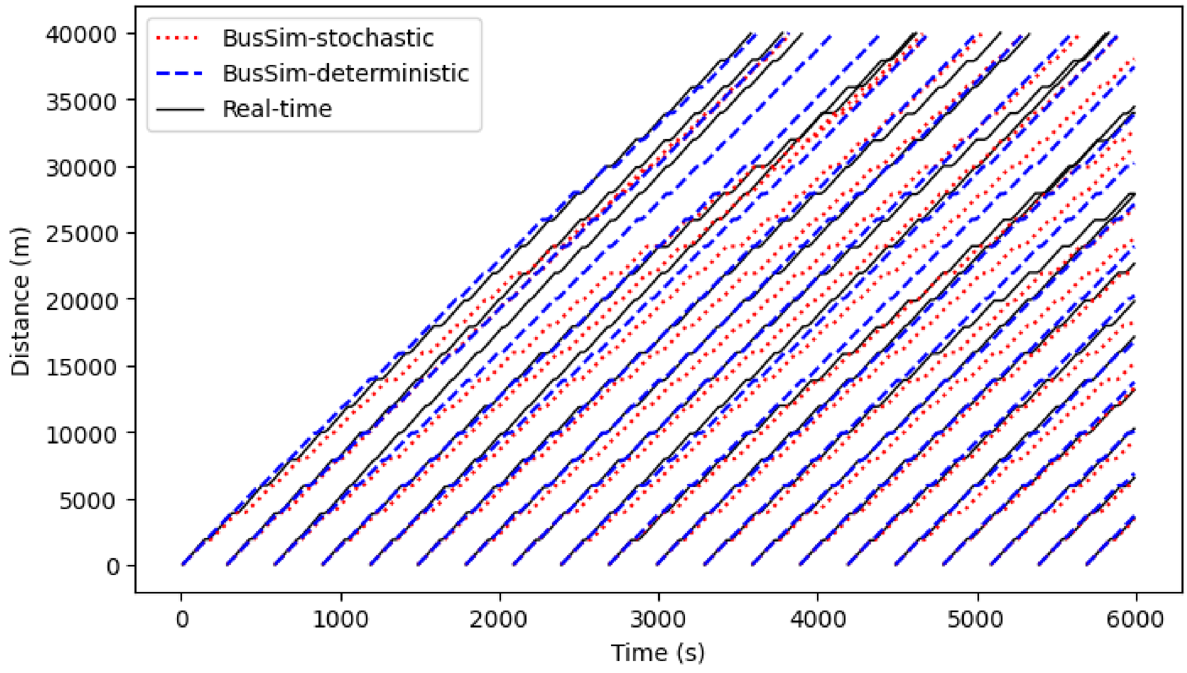
maxDemand = 2

$\xi = 7\%$

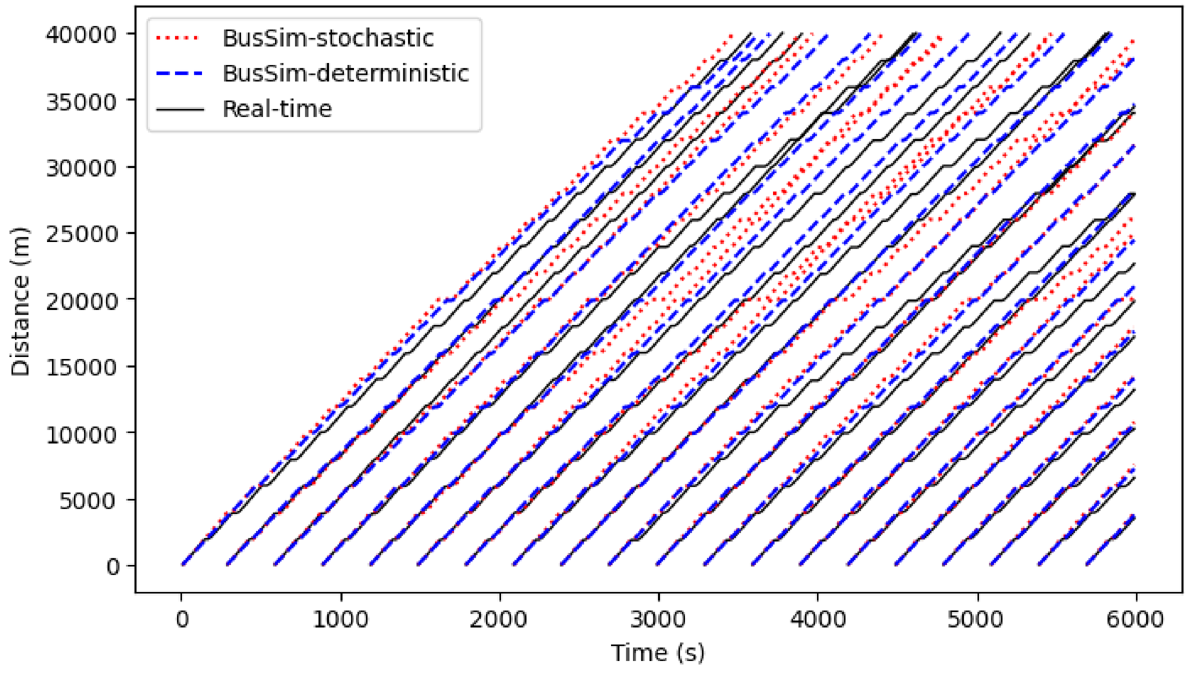
*Procedure*

The tester aims to follow the description in the research publication that would suggest the synthetic historical data was generated with maxDemand = 2 and $\xi = 7\%$. These parameters require the use of BusSim-truth for the generation of historical data. The code for this historical data generation for BusSim_truth.py involves conducting 20 replications and taking the standard deviation and meand of the produced GPS data. This functionality is edited by the tester such that number of replications is set to 1. (this functionality may be used for replications in subsequent experiments).

Reproduced Figure calibrated using BusSim truth



Reproduced figure using calibration to busSim stochastic



### 1.27.5 Figure 9

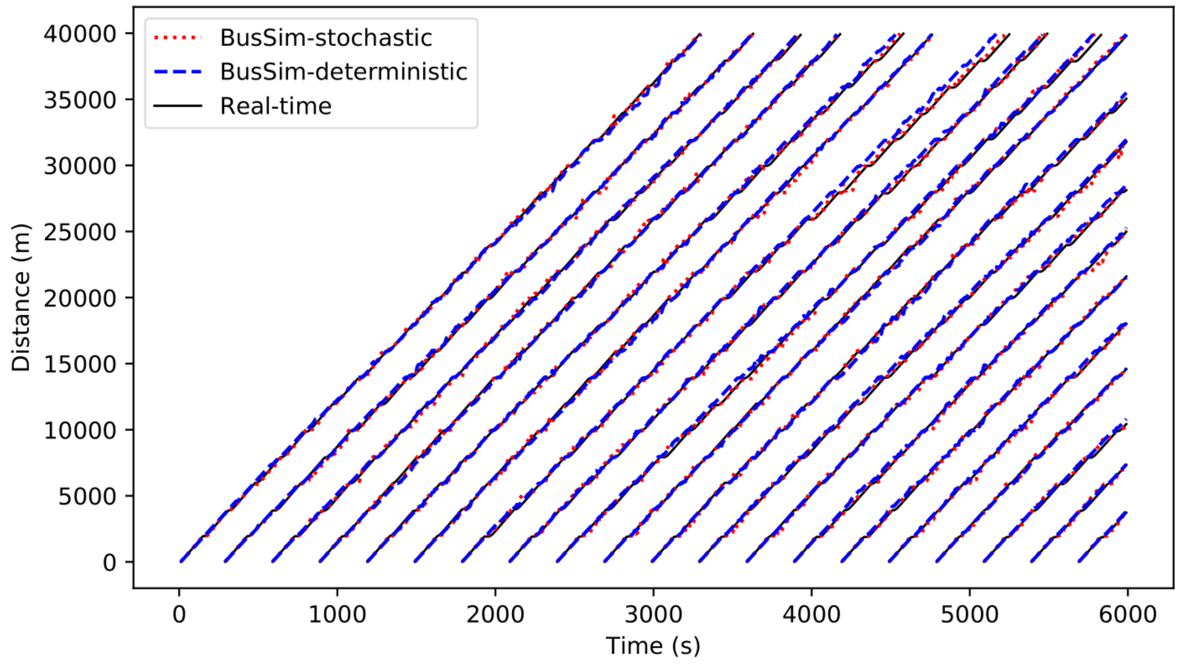


*Figure 30: Original Figure 9 plot: Prediction results from scenario 3: parameter calibration and particle filtering*

*Figure description from publication:*

This figure illustrates the results after the models have been calibrated and have 'real-time' data assimilated into them during runtime.

*Interpretation of the Figure from publication:*

Figure 9 shows a better prediction performance compared to figures 7 and 8. There are still observable gaps between the prediction and synthetic 'real-time' GPS data because the underlying models do not know the underlying stochasticity and dynamicity in the synthetic data but improvements appear to be substantial.

*Where is the code to generate this plot found?*

M01_BusSim_PF_v2.py

*What is plotted on the figure according to the code?*

Bus trajectories from BusSim-stochastic and BusSim-deterministic

*Which data files are required to reproduce this figure?*

Data files required:

Realtime_data_IncreaseRate_XX.pkl

BusSim_ModelX_calibration_IncreaseRate_XX.pkl

*Which model is used to produce the necessary files?*

Realtime_data_IncreaseRate_X.pkl is produced by running BusSim_truth.py. BusSim_ModelX_Calibration_IncreaseRate_X.pkl is produced in the generation of figure 8 using all three models and BusSim_model_calibration.py.

M01_BusSim_PF_v2.py contains the code for executing scenario 3 particle filter.

*What are the parameter settings required for the model to produce the data files?*

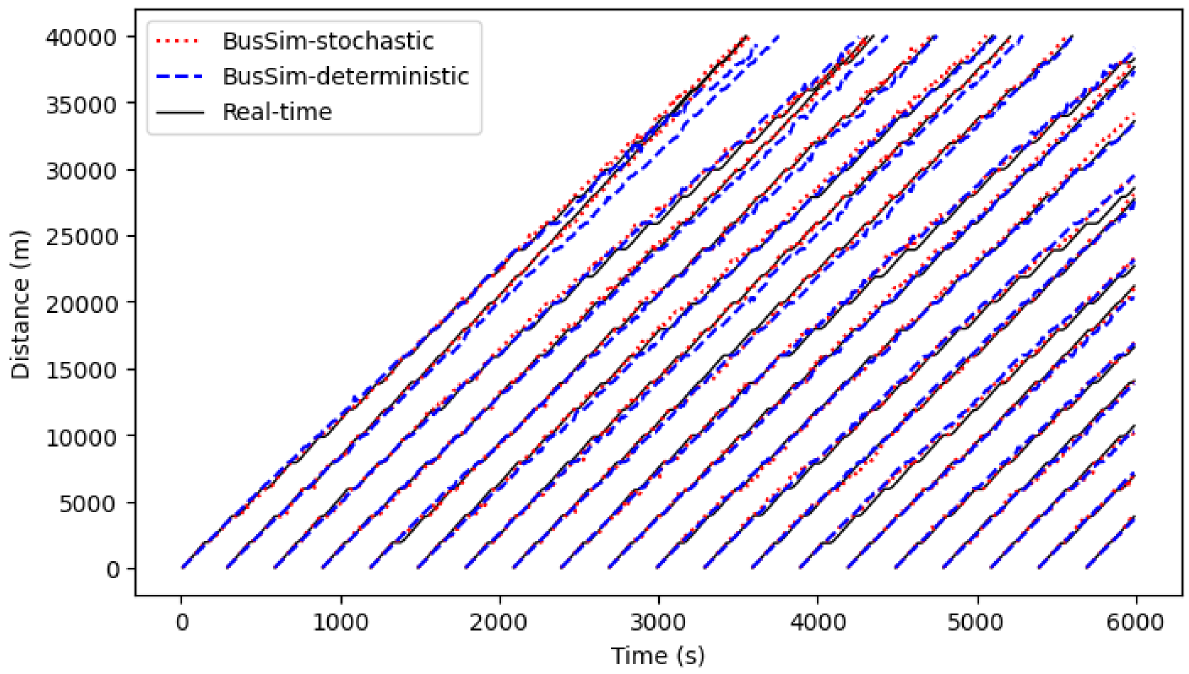
| Unspecified parameters | Assumption | Note |
|---|---|---|
| maxDemand | 2 | Assume value is the same as previous figures. |
| IncreaseRate | 7% | Assume value is the same as previous figures. |

Preliminary Justification: It is assumed by the tester that the maxDemand and IncreaseRate (i.e. $\xi$) are equal to 2 and 7% respectively as this was the parameter setup in the previous figures. The author of the publication compares the three figures 7,8 and 9. It is therefore logical to reason that such a comparison is possible when the parameter settings for maxDemand and IncreaseRate are fixed across the scenarios.

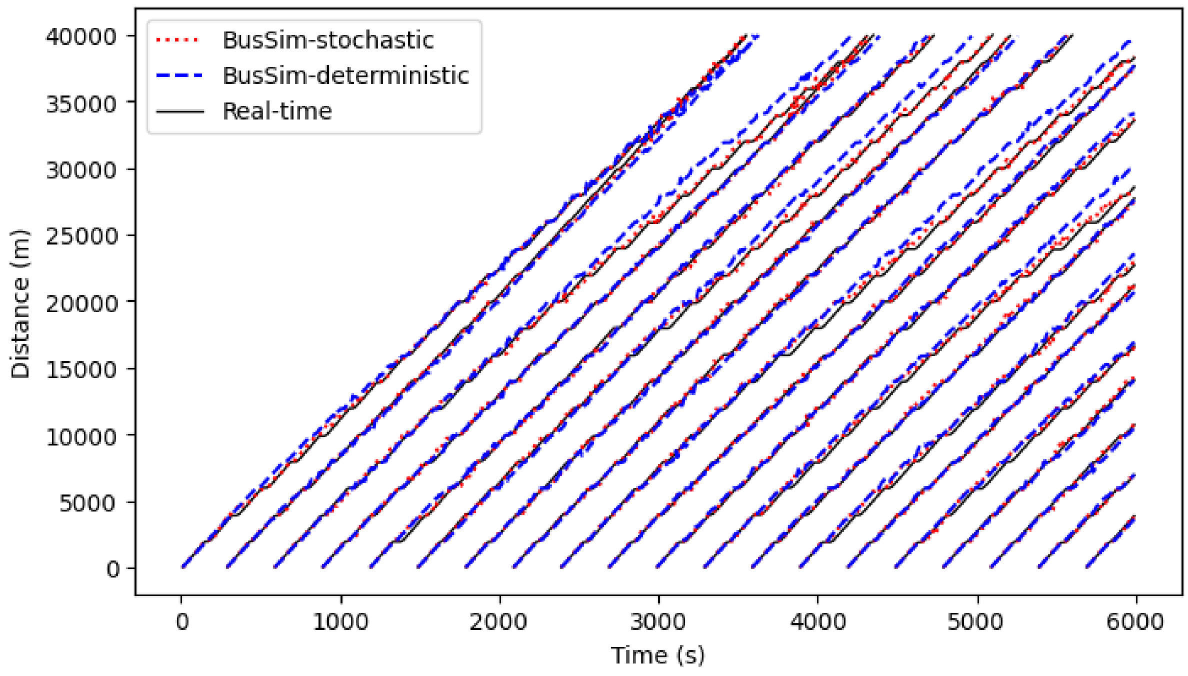
**However, the research publication was accompanied by supporting materials including a Latex Project of the published article.** In this study the plot referenced and embedded at the location of Figure 9 in the publication is called "*Fig_PF_IncreaseRate_5.pdf*". Provided figure naming was done accurately, this suggest that the parameter setting should be IncreaseRate = 5%. This counteracts the above assumption. The tester assumes that the name of the pdf file is descriptive of its contents i.e. the Figure 9 pdf file used IncreaseRate= 5%.

| Unspecified parameters | Assumption | Note |
|---|---|---|
| maxDemand | 2 | Assume value is the same as previous figures. |
| IncreaseRate | 5% | Assume value according to Latex Project pdf file naming. |

Reproduced using bus simtruth calibration



Reproduced using bussim stochastic calibration maxdemand = 2



## 1.28 Sensitivity Analysis of maxDemand and dynamic change rate $\xi$

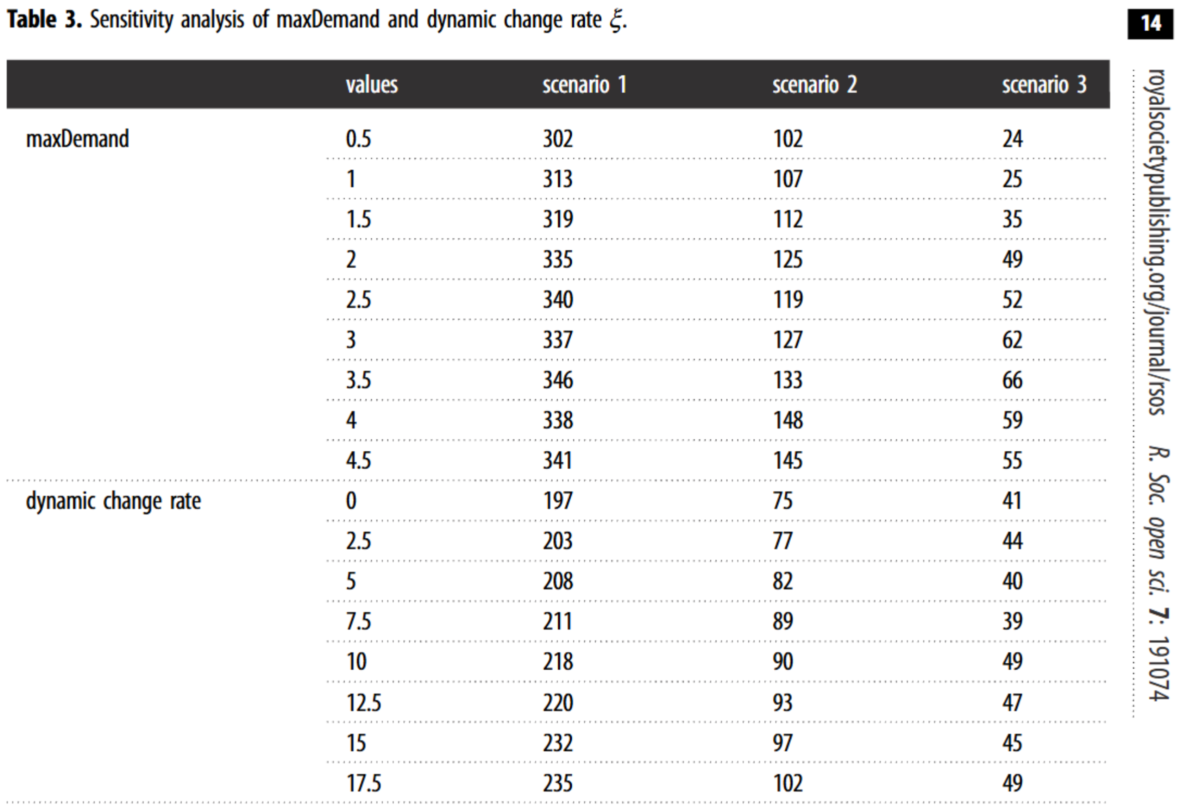


**Table 3.** Sensitivity analysis of maxDemand and dynamic change rate $\xi$.

14

royalsocietypublishing.org/journal/rsos *R. Soc. open sci.* **7**: 191074

| | values | scenario 1 | scenario 2 | scenario 3 |
|---|---|---|---|---|
| maxDemand | 0.5 | 302 | 102 | 24 |
| | 1 | 313 | 107 | 25 |
| | 1.5 | 319 | 112 | 35 |
| | 2 | 335 | 125 | 49 |
| | 2.5 | 340 | 119 | 52 |
| | 3 | 337 | 127 | 62 |
| | 3.5 | 346 | 133 | 66 |
| | 4 | 338 | 148 | 59 |
| | 4.5 | 341 | 145 | 55 |
| dynamic change rate | 0 | 197 | 75 | 41 |
| | 2.5 | 203 | 77 | 44 |
| | 5 | 208 | 82 | 40 |
| | 7.5 | 211 | 89 | 39 |
| | 10 | 218 | 90 | 49 |
| | 12.5 | 220 | 93 | 47 |
| | 15 | 232 | 97 | 45 |
| | 17.5 | 235 | 102 | 49 |

*Figure 31: Table 3 sourced from case study publication Kieu et al. (2020)*

*Table description from publication:*

The same experiments as described in scenarios 1 to 3, are repeated at different values of maxDemand and dynamic change rate $\xi$. To increase robustness, 10 replications have been made for each experiment and the average root mean square error (RMSE) values reported. RMSE is calculated as the difference in prediction bus location and synthetic 'real-time'bus location.

*Interpretation of the Table from publication:*

It is clear that scenario 3 (i.e. the combination of parameter calibration and Data Assimilation) outperforms the other two scenarios.

*Where is the code to generate this table found?*

There is no code that directly generates the table but there is code that generates results similar to the RMSE values in the table. Two functions *IncreaseRate_analysis* and *maxDemand_analysis* that execute a parameter sweep for the dynamic change rate in range [1,20] and maxDemand in range [0.5, 4.5] respectively. The main python files A02_doing_nothing_analysis.py, A03_calibration.py and M01_BusSim_PF_v2.py which correspond to the original experiment's 3 scenarios all contain their respective versions of *IncreaseRate_analysis* and *maxDemand_analysis*.

*What does the data in the table represent according to the code?*

The data represents the RMSE between the real-time bus trajectories i.e. ground truth and the bus trajectories of BusSim-stochastic and the RMSE values between the real-time bus trajectories i.e. ground truth and the bus trajectories of BusSim-deterministic.

*Which data files are required to reproduce this figure?*

The calibration files required are:

*BusSim_Model1_calibration_IncreaseRate_X.pkl*

*BusSim_Model2_calibration_IncreaseRate_X.pkl*

*BusSim_Model1_calibration_static_maxDemand_X.pkl*

*BusSim_Model2_calibration_static_maxDemand_X.pkl*

The real-time files required are:

Realtime_data_IncreaseRate_X.pkl

Realtime_data_static_maxDemand_X.pkl

*Which model is used to produce the necessary files?*

BusSim-truth and BusSim-stochastic create the real-time data files required

The file BusSim_model_calibration.py contains the functions necessary to produce the calibration files using BusSim-stochastic and BusSim-deterministic.

*What are the parameter settings required for the model to produce the data files?*

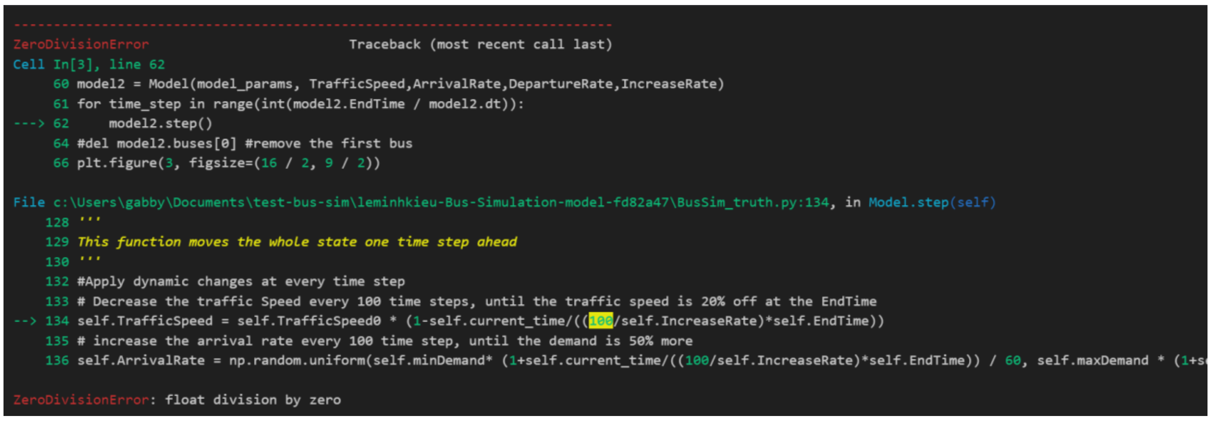
| Parameters | Values | |
|---|---|---|
| maxDemand | [0.5 , 4.5] | Assume static maxDemand. Historica and realtime static maxdemand data files generated by BusSim_stochastic. IncreaseRate = 0 |
| IncreaseRate | [0 , 17.5] | maxDemand = 1 |
| Number of replications | 10 | |

*Procedure*

1. Generate real-time and historical data for IncreaseRate = [0,17.5] and maxDemand = [0.5,4.5]
2. Execute A02_doing_nothing_analysis IncreaseRate_analysis and maxDemand_analysis functions with do_reps = True and Numreps = 10.
3. Generate BusSim_ModelX_calibration data for BusSim-stochastic and BusSim-deterministic
4. Execute A03_calibration.py IncreaseRate_analysis and maxDemand_analysis.
5. Execute M01_BusSim_PF_v2.py IncreaseRate_analysis and maxDemand_analysis.

### 1.28.1 Errors

The sensitivity analysis for dynamic change rate is conducted in the range [0,17.5] with intervals of 2.5 ( i.e. range(0,20,2.5)). However, IncreaseRate = 0, produces a divide-by-zero error.



```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[3], line 62
     60 model2 = Model(model_params, TrafficSpeed,ArrivalRate,DepartureRate,IncreaseRate)
     61 for time_step in range(int(model2.EndTime / model2.dt)):
---> 62     model2.step()
     64 #del model2.buses[0] #remove the first bus
     66 plt.figure(3, figsize=(16 / 2, 9 / 2))

File c:\Users\gabby\Documents\test-bus-sim\leminhkieu-Bus-Simulation-model-fd82a47\BusSim_truth.py:134, in Model.step(self)
    128 '''
    129 This function moves the whole state one time step ahead
    130 '''
    132 #Apply dynamic changes at every time step
    133 # Decrease the traffic Speed every 100 time steps, until the traffic speed is 20% off at the EndTime
--> 134 self.TrafficSpeed = self.TrafficSpeed0 * (1-self.current_time/((100/self.IncreaseRate)*self.EndTime))
    135 # increase the arrival rate every 100 time step, until the demand is 50% more
    136 self.ArrivalRate = np.random.uniform(self.minDemand* (1+self.current_time/((100/self.IncreaseRate)*self.EndTime)) / 60, self.maxDemand * (1+s

ZeroDivisionError: float division by zero
```

Upon closer inspection, the tester noted that the equations that used the $\xi$ parameter in the BusSim-truth model differed from the equations included in *Appendix A. The BusSim model* of the publication.

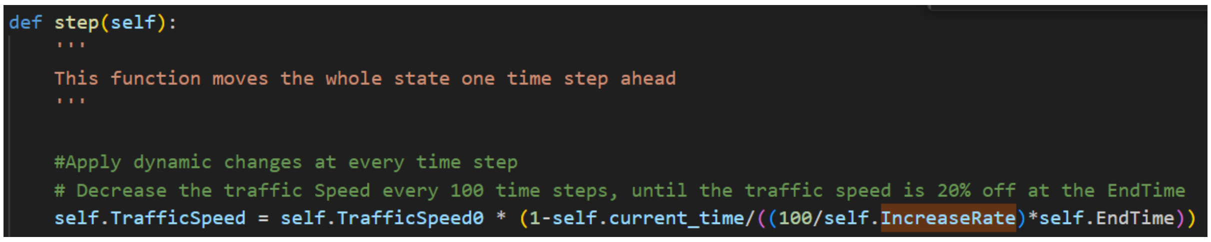
$\xi$ represents the change in passenger demand or surrounding traffic speed. This is modelled as the following equations according to the research publication.

$$V = V \cdot \left(1 - \frac{t}{T} \cdot \frac{100}{\xi}\right) \tag{A8}$$

$$Arr_m = Arr_m \cdot \left(1 - \frac{t}{T} \cdot \frac{100}{\xi}\right) \tag{A9}$$

The article then states that for $\xi > 0$ , the equation gradually reduces surrounding traffic speed V and increases the arrival rate $\mathrm{Arr}_m$ which would lead to more bus delays and congestion.

However, the contents of the paper are in conflict with the formulae implemented in code for BusSim-truth.



```
def step(self):
    '''
    This function moves the whole state one time step ahead
    '''

    #Apply dynamic changes at every time step
    # Decrease the traffic Speed every 100 time steps, until the traffic speed is 20% off at the EndTime
    self.TrafficSpeed = self.TrafficSpeed0 * (1-self.current_time/((100/self.IncreaseRate)*self.EndTime))
```

self.TrafficSpeed=self.TrafficSpeed0 *(1-self.current_time/((100/self.IncreaseRate)*self.EndTime))

self.ArrivalRate=np.random.uniform(self.minDemand* (1+self.current_time/((100/self.IncreaseRate)*self.EndTime)) / 60, self.maxDemand * (1+self.current_time/((100/self.IncreaseRate)*self.EndTime)) / 60, self.NumberOfStop)

According to the code, the equations are implemented as follows:

$$V = V \cdot \left(1 - \frac{t}{\frac{100}{\xi} \cdot T}\right)$$

$$Arr_m = Arr_m \cdot \left(1 + \frac{t}{\left(\frac{100}{\xi}\right) \cdot T}\right)$$

As such, there exist a discrepancy between the formulae described and those implemented.

In the current form of the code, IncreaseRate = 0 cannot be executed.

When $\xi = 0$, $Arr_m$ and V should remain constant over time. Therefore the brackets in each equations can be removed. The tester will use this approach to generate data for IncreaseRate =0. This will involve editing BusSim_truth.py Model class temporarily.

**NOTES**

- Two different datasets were generated. Realtime_static_maxDemand_XX.pkl and Historical_data_static_maxDemand_XX.pkl were generated using BusSim-truth model and BusSim-stochastic model. BusSim-truth is aligned with the description in the research publication. BusSim-stochastic is aligned with the comments and setup of the codebase upon download.
- The replications programmed into the codebase are only for the historical data. Real-time data files are generated without replications. Historical data files are generated over a number of replications for which the mean and standard deviation of the GPS data and the model parameters used are saved in the pickle file. This applies to the generation of both static_maxDemand and IncreaseRate datasets.

*RMSE formula error*

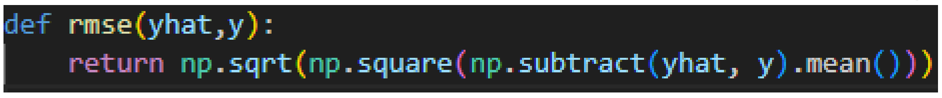
The RMSE formula used for the sensitivity analysis is specified in the research article as follows:

$$RMSE = \sqrt{\left(\frac{1}{T}\right) \sum_{k=1}^{T} (\widehat{y_k} - y_k)^2}$$

$\widehat{y_k}$: the bus location at time $k$ from the model prediction

$y_k$: the bus location at time $k$ from the synthetic 'real-time' data

The RMSE function provided and used in the code is calculated according to a different formula.



```
def rmse(yhat,y):
    return np.sqrt(np.square(np.subtract(yhat, y).mean()))
```

$$RMSE = \sqrt{\left(\left(\frac{1}{T}\right) \sum_{k=1}^{T} (\widehat{y_k} - y_k)\right)^2}$$

# Appendix B: Reproducibility Experiments of Table 3

## Table 3: Sensitivity Analysis of maxDemand and dynamic change rate $\xi$

It is noted that the BusSim_stochastic.py file is used to create the synthetic ground truth data for the maxDemand sensitivity analysis. -> Historical Data static maxDemand, realtime data static maxDemand. BusSim_truth.py only used to generate data for IncreaseRate parameter sweep.

The publication does not state which model was used for the sensitivity analysis as the table appears to report results from only one of the model.

Number of replications $R = 10$

$$RMSE = \sqrt{\frac{1}{T}\sum_{k=1}^{T}(\hat{y}_k - y_k)^2}$$

Reported value $Average\ RMSE = \frac{\sum_{i=1}^{R} RMSE_i}{R}$

**Assumption: BusSim_stochastic is used for the sensitivity analysis.**

## 1.29 Experiment 1-Reproducing Table 3 using the original data files

The scenarios were executed for 10 replications using the data files available in the original Zenodo download. It should be noted that maxDemand sensitivity analysis uses files generated by BusSim_stochastic as its ground truth whereas the increaserate sensitivity analysis uses files generated by BusSim_truth. The latter makes sense as the static models BusSim-stochastic and BusSim-deterministic do not have an IncreaseRate parameter.

Number of replications = 10

BusSim-deterministic results

| maxDemand | scenario_1 | scenario_2 | scenario_3 |
|---|---|---|---|
| 0.5 | 245.0 | 80.0 | 15.0 |
| 1.5 | 380.0 | 8.0 | 31.0 |
| 2.5 | 257.0 | 195.0 | 115.0 |
| 3.5 | 232.0 | 184.0 | 78.0 |
| 4.5 | 241.0 | 67.0 | 24.0 |
| IncreaseRate | scenario_1 | scenario_2 | scenario_3 |
| 1 | 247 | 12 | 23 |
| 3 | 378 | 54 | 18 |
| 5 | 258 | 4 | 17 |
| 7 | 212 | 54 | 9 |
| 9 | 315 | 44 | 44 |
| 11 | 285 | 45 | 50 |
| 13 | 321 | 22 | 44 |
| 15 | 379 | 88 | 30 |

|  |  |  |  |
|---|---|---|---|
| 17 | 229 | 66 | 10 |
| 19 | 292 | 21 | 30 |

BusSim-stochastic results

| maxDemand | scenario_1 | scenario_2 | scenario_3 |
|---|---|---|---|
| 0.5 | 266.0 | 86.0 | 14.0 |
| 1.5 | 279.0 | 33.0 | 9.0 |
| 2.5 | 297.0 | 146.0 | 27.0 |
| 3.5 | 152.0 | 93.0 | 30.0 |
| 4.5 | 99.0 | 52.0 | 111.0 |
| IncreaseRate | scenario_1 | scenario_2 | scenario_3 |
| 1 | 202 | 85 | 33 |
| 3 | 250 | 103 | 22 |
| 5 | 256 | 31 | 36 |
| 7 | 211 | 31 | 25 |
| 9 | 232 | 42 | 61 |
| 11 | 351 | 30 | 41 |
| 13 | 261 | 238 | 61 |
| 15 | 243 | 47 | 20 |
| 17 | 226 | 41 | 22 |
| 19 | 223 | 161 | 40 |

## 1.30 Experiment 2 – Reproducing Table 3 using reproduced data

The same experiment as experiment 1 is rerun using datasets that have been generated using the code instead of the datasets that were stored in the github repository/ zenodo archive at the time of download. The range of the IncreaseRate analysis will be [0,17.5] with step size of 2.5. This contrasts with the hard-coding of the archived code that used a range of [1,20] step size = 2.

### 1.30.1 Experiment 2A – Using models calibrated by BusSim-truth for IncreaseRate and using models calibrated by BusSim-stochastic for maxDemand

BusSim-stochastic

| maxDemand | scenario_1 | scenario_2 | scenario_3 |
|---|---|---|---|
| 0.5 | 178.0 | 39.0 | 39.0 |
| 1.5 | 254.0 | 21.0 | 13.0 |
| 2.5 | 277.0 | 108.0 | 24.0 |
| 3.5 | 208.0 | 51.0 | 42.0 |
| 4.5 | 141.0 | 193.0 | 78.0 |
| IncreaseRate | scenario_1 | scenario_2 | scenario_3 |
| 0.0 | 270.0 | 50.0 | 40.0 |
| 2.5 | 192.0 | 31.0 | 34.0 |
| 5.0 | 340.0 | 65.0 | 21.0 |
| 7.5 | 251.0 | 26.0 | 24.0 |
| 10.0 | 227.0 | 54.0 | 34.0 |
| 12.5 | 340.0 | 244.0 | 96.0 |
| 15.0 | 371.0 | 183.0 | 104.0 |
| 17.5 | 245.0 | 114.0 | 53.0 |

BusSim-deterministic

| maxDemand | scenario_1 | scenario_2 | scenario_3 |
|---|---|---|---|
| 0.5 | 348.0 | 89.0 | 9.0 |
| 1.5 | 340.0 | 108.0 | 45.0 |
| 2.5 | 358.0 | 80.0 | 84.0 |
| 3.5 | 172.0 | 176.0 | 102.0 |
| 4.5 | 274.0 | 28.0 | 121.0 |
| IncreaseRate | scenario_1 | scenario_2 | scenario_3 |
| 0.0 | 420.0 | 63.0 | 25.0 |
| 2.5 | 354.0 | 16.0 | 18.0 |
| 5.0 | 478.0 | 56.0 | 45.0 |
| 7.5 | 376.0 | 38.0 | 19.0 |
| 10.0 | 400.0 | 38.0 | 16.0 |
| 12.5 | 328.0 | 138.0 | 95.0 |
| 15.0 | 429.0 | 42.0 | 49.0 |
| 17.5 | 567.0 | 124.0 | 113.0 |

### 1.30.2 Experiment 2B – Using models calibrated to BusSim-truth

BusSim-stochastic

| maxDemand | scenario_1 | scenario_2 | scenario_3 |
|---|---|---|---|
| 0.5 | 181.0 | 76.0 | 47.0 |
| 1.5 | 263.0 | 95.0 | 10.0 |
| 2.5 | 240.0 | 58.0 | 124.0 |
| 3.5 | 182.0 | 101.0 | 103.0 |
| 4.5 | 133.0 | 188.0 | 197.0 |
| IncreaseRate | scenario_1 | scenario_2 | scenario_3 |
| 0.0 | 270.0 | 54.0 | 42.0 |
| 2.5 | 192.0 | 34.0 | 36.0 |
| 5.0 | 340.0 | 87.0 | 26.0 |
| 7.5 | 251.0 | 33.0 | 18.0 |
| 10.0 | 227.0 | 40.0 | 38.0 |
| 12.5 | 340.0 | 245.0 | 91.0 |
| 15.0 | 371.0 | 194.0 | 80.0 |
| 17.5 | 245.0 | 125.0 | 59.0 |

BusSim-deterministic

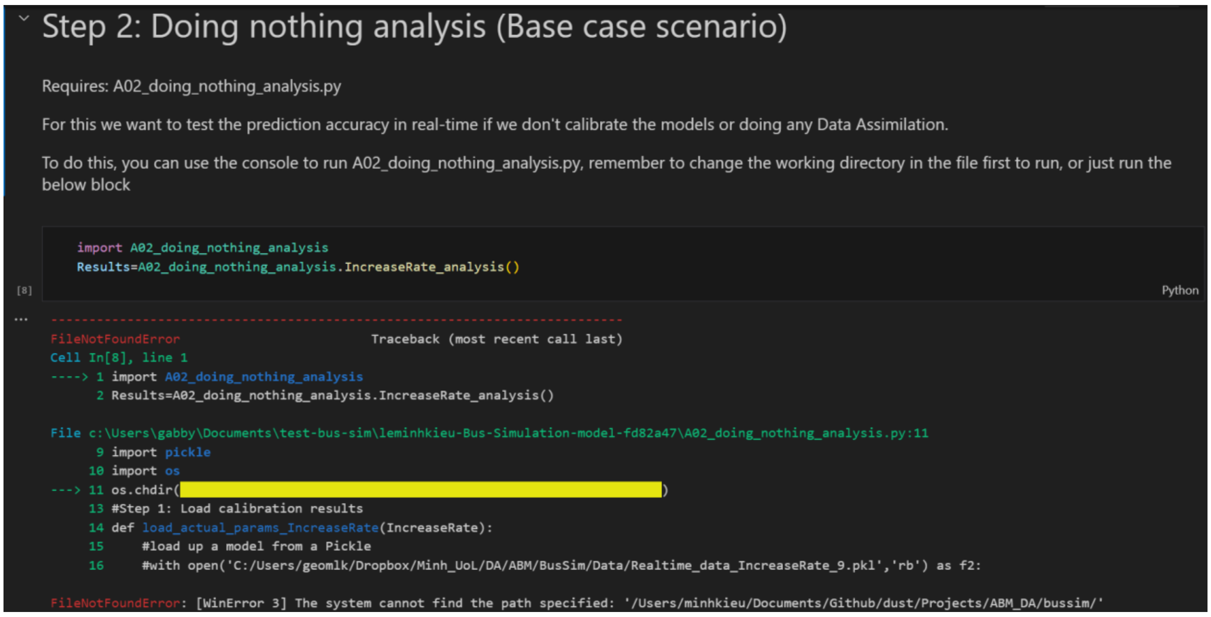
| maxDemand | scenario_1 | scenario_2 | scenario_3 |
|---|---|---|---|
| 0.5 | 373.0 | 1.0 | 8.0 |
| 1.5 | 349.0 | 78.0 | 18.0 |
| 2.5 | 270.0 | 176.0 | 40.0 |
| 3.5 | 202.0 | 112.0 | 23.0 |
| 4.5 | 274.0 | 57.0 | 57.0 |
| IncreaseRate | scenario_1 | scenario_2 | scenario_3 |
| 0.0 | 420.0 | 63.0 | 30.0 |
| 2.5 | 354.0 | 16.0 | 11.0 |
| 5.0 | 478.0 | 56.0 | 42.0 |
| 7.5 | 376.0 | 38.0 | 16.0 |
| 10.0 | 400.0 | 38.0 | 16.0 |
| 12.5 | 328.0 | 138.0 | 91.0 |
| 15.0 | 429.0 | 42.0 | 34.0 |
| 17.5 | 567.0 | 124.0 | 111.0 |

# Appendix C: Error Logs

## 1.31 Test 1 – Preliminary execution

07/04/2025

Note: This log was generated for the first try at generating the data included in the BusSim codebase. The data from this attempt can be found in test_v1_data/.



Step 2: Doing nothing analysis (Base case scenario)

Requires: A02_doing_nothing_analysis.py

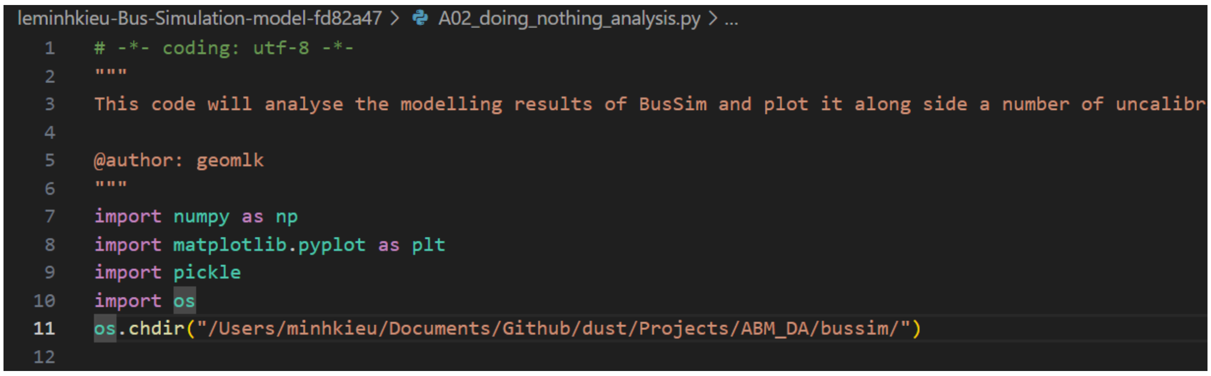
For this we want to test the prediction accuracy in real-time if we don't calibrate the models or doing any Data Assimilation.

To do this, you can use the console to run A02_doing_nothing_analysis.py, remember to change the working directory in the file first to run, or just run the below block

```
import A02_doing_nothing_analysis
Results=A02_doing_nothing_analysis.IncreaseRate_analysis()
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[8], line 1
----> 1 import A02_doing_nothing_analysis
      2 Results=A02_doing_nothing_analysis.IncreaseRate_analysis()

File c:\Users\gabby\Documents\test-bus-sim\leminhkieu-Bus-Simulation-model-fd82a47\A02_doing_nothing_analysis.py:11
      9 import pickle
     10 import os
---> 11 os.chdir([redacted])
     13 #Step 1: Load calibration results
     14 def load_actual_params_IncreaseRate(IncreaseRate):
     15     #load up a model from a Pickle
     16     #with open('C:/Users/geomlk/Dropbox/Minh_UoL/DA/ABM/BusSim/Data/Realtime_data_IncreaseRate_9.pkl','rb') as f2:

FileNotFoundError: [WinError 3] The system cannot find the path specified: '/Users/minhkieu/Documents/Github/dust/Projects/ABM_DA/bussim/'
```

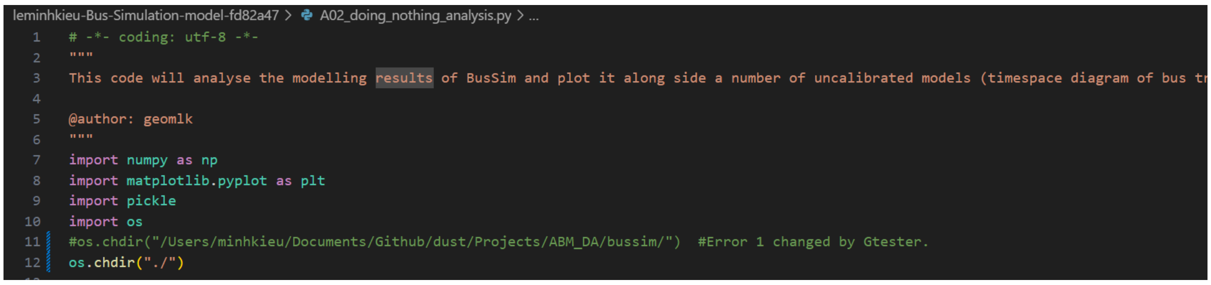
This error was due to the use of absolute paths in A02_doing_nothing_analysis.py.



leminhkieu-Bus-Simulation-model-fd82a47 > A02_doing_nothing_analysis.py > ...

```
# -*- coding: utf-8 -*-
"""
This code will analyse the modelling results of BusSim and plot it along side a number of uncalibr

@author: geomlk
"""
import numpy as np
import matplotlib.pyplot as plt
import pickle
import os
os.chdir("/Users/minhkieu/Documents/Github/dust/Projects/ABM_DA/bussim/")

```

Error fixed by changing the relative path.

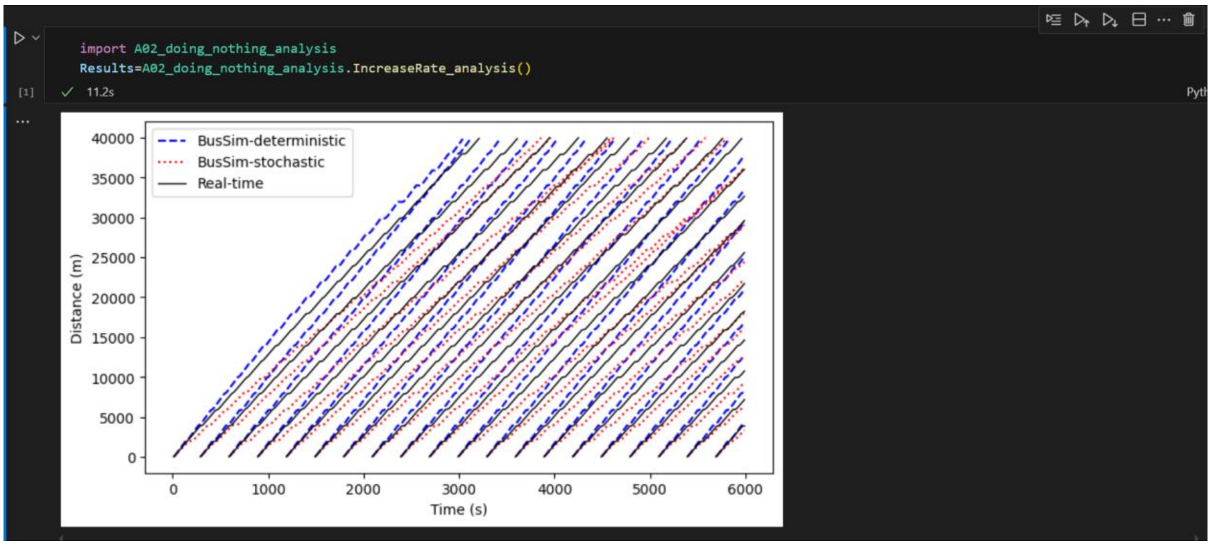


leminhkieu-Bus-Simulation-model-fd82a47 > A02_doing_nothing_analysis.py > ...

```
# -*- coding: utf-8 -*-
"""
This code will analyse the modelling results of BusSim and plot it along side a number of uncalibrated models (timespace diagram of bus tr
@author: geomlk
"""
import numpy as np
import matplotlib.pyplot as plt
import pickle
import os
#os.chdir("/Users/minhkieu/Documents/Github/dust/Projects/ABM_DA/bussim/")  #Error 1 changed by Gtester.
os.chdir("./")
```

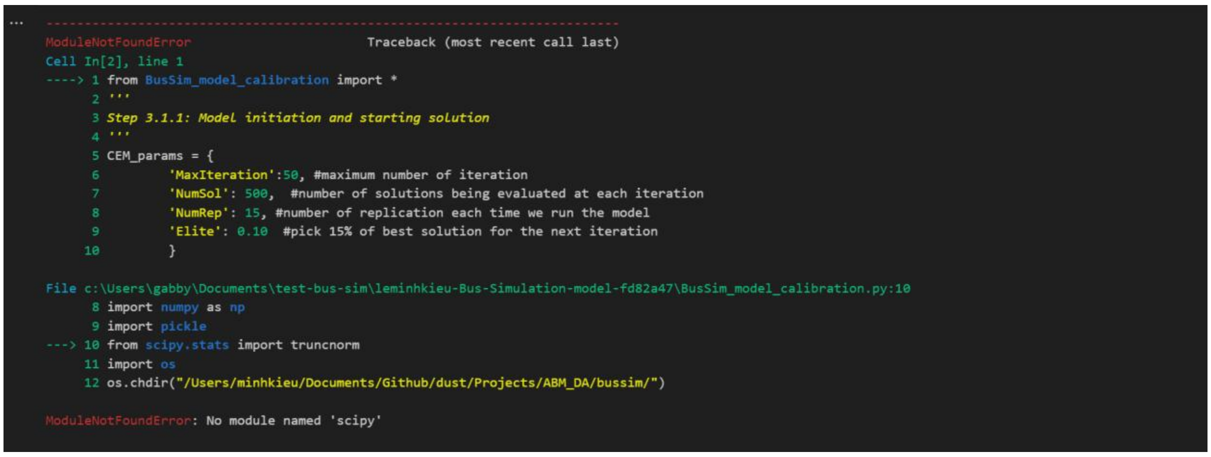
Error resolved. Code in cell now runs.

Note in the Jupyter Notebook. The author indicated that the working directory would have to be changed when running the file from the command line. He also stated that the cell block could also be run alternatively. Does this count as acknowledgement of the use of his absolute file path or not?



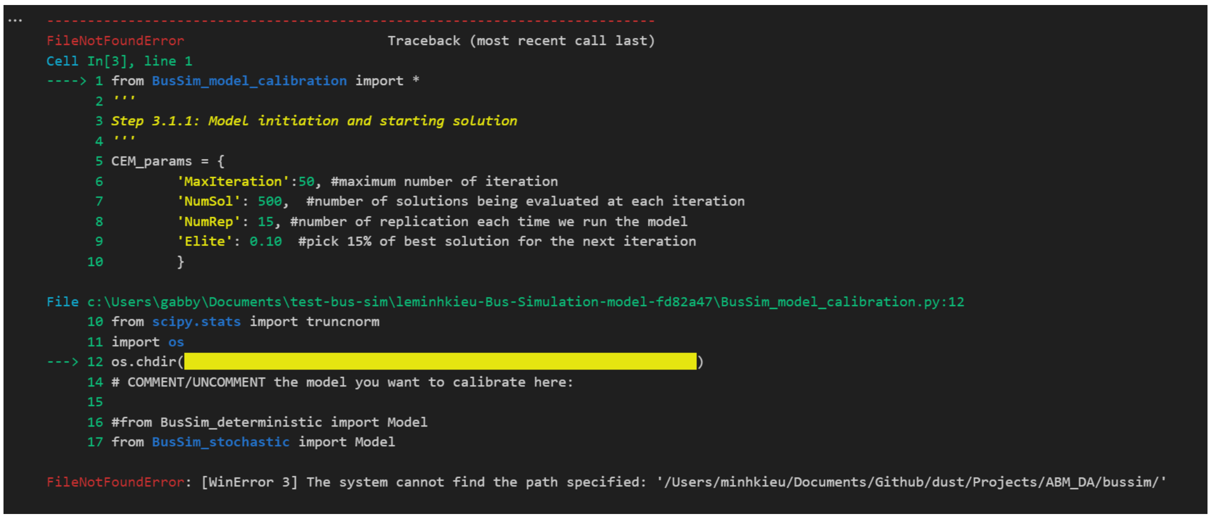
```
import A02_doing_nothing_analysis
Results=A02_doing_nothing_analysis.IncreaseRate_analysis()
```

Step 3.1 error. Module not found error because Scipy is require by one of the .py files specifically BusSim_model_calibration.py. error resolved by installing Scipy module



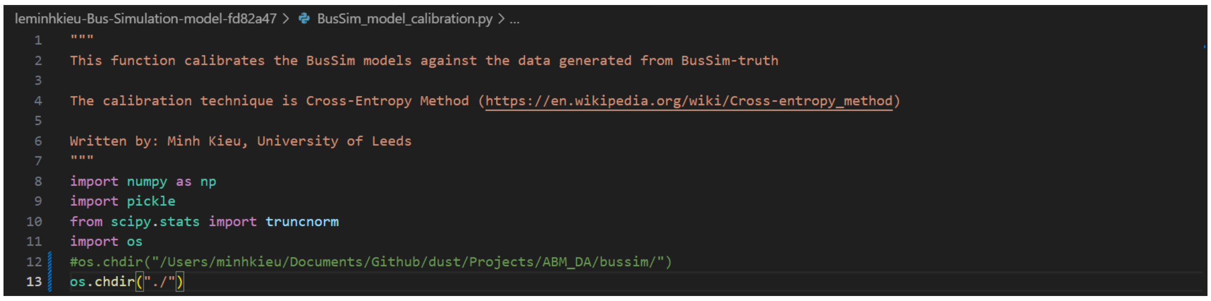
```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[2], line 1
----> 1 from BusSim_model_calibration import *
      2 '''
      3 Step 3.1.1: Model initiation and starting solution
      4 '''
      5 CEM_params = {
      6         'MaxIteration':50, #maximum number of iteration
      7         'NumSol': 500,  #number of solutions being evaluated at each iteration
      8         'NumRep': 15, #number of replication each time we run the model
      9         'Elite': 0.10  #pick 15% of best solution for the next iteration
     10         }

File c:\Users\gabby\Documents\test-bus-sim\leminhkieu-Bus-Simulation-model-fd82a47\BusSim_model_calibration.py:10
      8 import numpy as np
      9 import pickle
---> 10 from scipy.stats import truncnorm
     11 import os
     12 os.chdir("/Users/minhkieu/Documents/Github/dust/Projects/ABM_DA/bussim/")

ModuleNotFoundError: No module named 'scipy'
```

Step 3.1 error. FileNotFoundError.



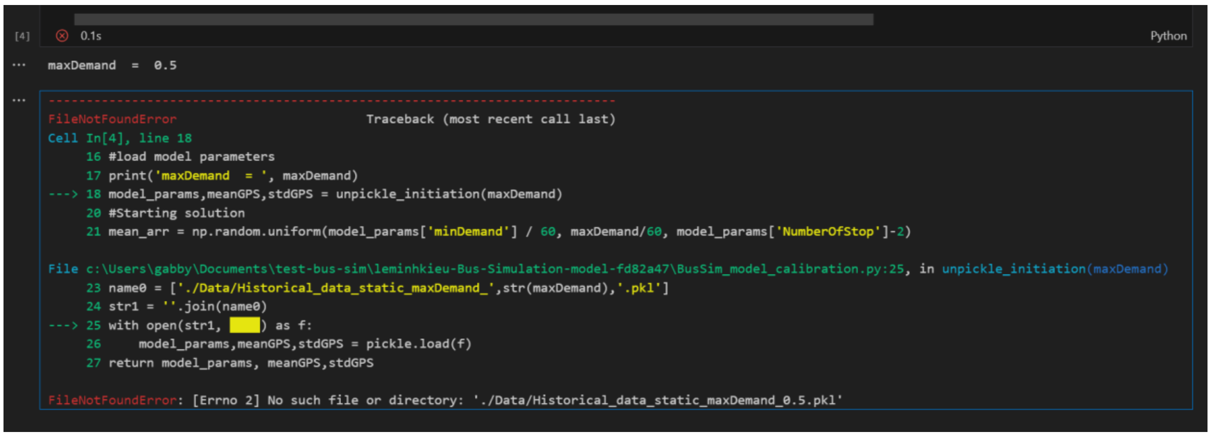
```
...  ---------------------------------------------------------------------------
     FileNotFoundError                         Traceback (most recent call last)
     Cell In[3], line 1
     ----> 1 from BusSim_model_calibration import *
           2 '''
           3 Step 3.1.1: Model initiation and starting solution
           4 '''
           5 CEM_params = {
           6         'MaxIteration':50, #maximum number of iteration
           7         'NumSol': 500,  #number of solutions being evaluated at each iteration
           8         'NumRep': 15, #number of replication each time we run the model
           9         'Elite': 0.10  #pick 15% of best solution for the next iteration
          10         }

     File c:\Users\gabby\Documents\test-bus-sim\leminhkieu-Bus-Simulation-model-fd82a47\BusSim_model_calibration.py:12
          10 from scipy.stats import truncnorm
          11 import os
     ---> 12 os.chdir(                                                              )
          14 # COMMENT/UNCOMMENT the model you want to calibrate here:
          15 
          16 #from BusSim_deterministic import Model
          17 from BusSim_stochastic import Model

     FileNotFoundError: [WinError 3] The system cannot find the path specified: '/Users/minhkieu/Documents/Github/dust/Projects/ABM_DA/bussim/'
```

Error corrected by changing the file path.



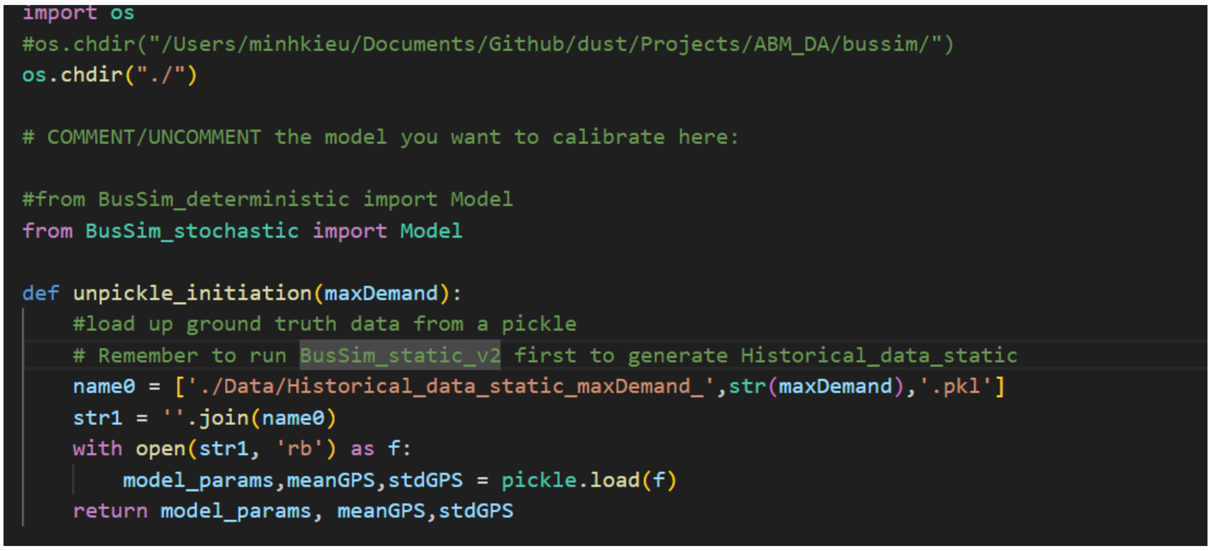
```
leminhkieu-Bus-Simulation-model-fd82a47 > BusSim_model_calibration.py > ...
"""
This function calibrates the BusSim models against the data generated from BusSim-truth

The calibration technique is Cross-Entropy Method (https://en.wikipedia.org/wiki/Cross-entropy_method)

Written by: Minh Kieu, University of Leeds
"""
import numpy as np
import pickle
from scipy.stats import truncnorm
import os
#os.chdir("/Users/minhkieu/Documents/Github/dust/Projects/ABM_DA/bussim/")
os.chdir("./")
```

Error #3 is similar to Error #1.



```
[4]   ⊗ 0.1s                                                                    Python

...  maxDemand  =  0.5

...  ---------------------------------------------------------------------------
     FileNotFoundError                         Traceback (most recent call last)
     Cell In[4], line 18
          16 #load model parameters
          17 print('maxDemand  =  ', maxDemand)
     ---> 18 model_params,meanGPS,stdGPS = unpickle_initiation(maxDemand)
          20 #Starting solution
          21 mean_arr = np.random.uniform(model_params['minDemand'] / 60, maxDemand/60, model_params['NumberOfStop']-2)

     File c:\Users\gabby\Documents\test-bus-sim\leminhkieu-Bus-Simulation-model-fd82a47\BusSim_model_calibration.py:25, in unpickle_initiation(maxDemand)
          23 name0 = ['./Data/Historical_data_static_maxDemand_',str(maxDemand),'.pkl']
          24 str1 = ''.join(name0)
     ---> 25 with open(str1,     ) as f:
          26     model_params,meanGPS,stdGPS = pickle.load(f)
          27 return model_params, meanGPS,stdGPS

     FileNotFoundError: [Errno 2] No such file or directory: './Data/Historical_data_static_maxDemand_0.5.pkl'
```

Error due to missing file that was not regenerated by code. Why was this file not generated by the code? Is it a data file from the real-world? Unlikely given this is a identical twin framework.



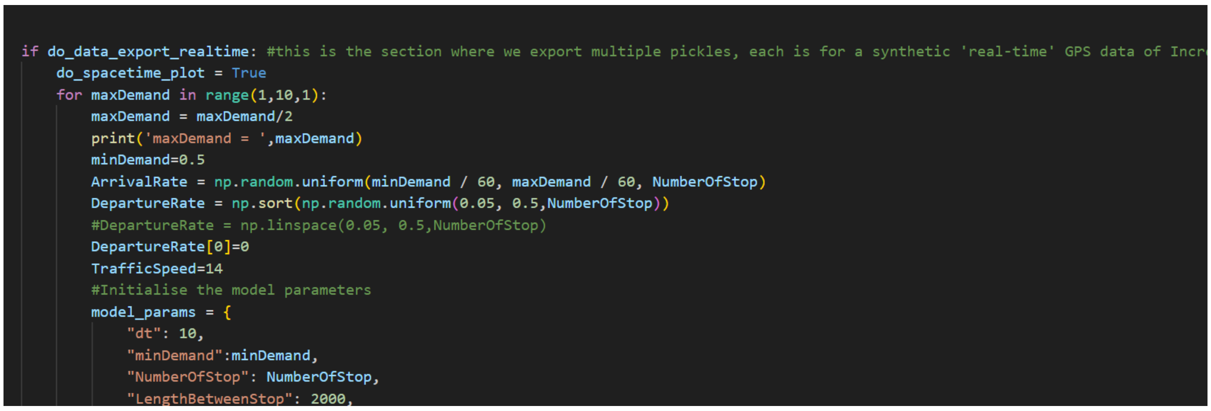
```python
import os
#os.chdir("/Users/minhkieu/Documents/Github/dust/Projects/ABM_DA/bussim/")
os.chdir("./")

# COMMENT/UNCOMMENT the model you want to calibrate here:

#from BusSim_deterministic import Model
from BusSim_stochastic import Model

def unpickle_initiation(maxDemand):
    #load up ground truth data from a pickle
    # Remember to run BusSim_static_v2 first to generate Historical_data_static
    name0 = ['./Data/Historical_data_static_maxDemand_',str(maxDemand),'.pkl']
    str1 = ''.join(name0)
    with open(str1, 'rb') as f:
        model_params,meanGPS,stdGPS = pickle.load(f)
    return model_params, meanGPS,stdGPS
```

Comment in the code indicates that there should be a BuSim_static_v2.py file to run first to generate the static data required from the calibration analysis. But this file is not included in the codebase provided.

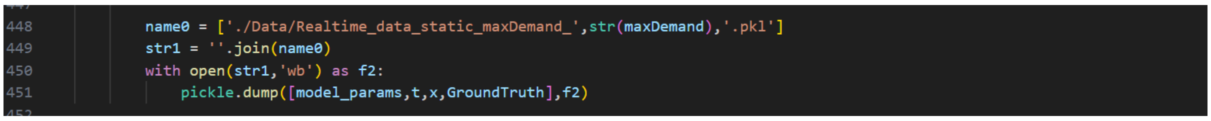
08/04/2025 – continuation

The jupyter notebook containing the experiments does not include any cells nor instructions to generate the historical static max demand files nor the real-time static max demand files. By examining the comments of BusSim_model_Calibration.py, it can be seen that the Historical_data_static needs to be generated by running the BusSim_static_v2. BusSim_static_v2 does not exist. However, upon examining the code of BusSim_stochastic.py, there is code to generate these files present in its 'main' function. This is evidence of filename changes that occurred but were not properly recorded in the description and comments of the codebase.

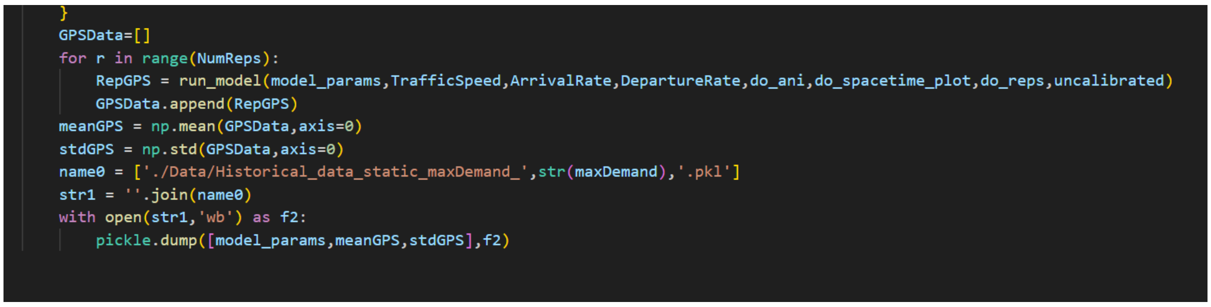
BusSim_stochastic.py



```python
if do_data_export_realtime: #this is the section where we export multiple pickles, each is for a synthetic 'real-time' GPS data of Incr
    do_spacetime_plot = True
    for maxDemand in range(1,10,1):
        maxDemand = maxDemand/2
        print('maxDemand = ',maxDemand)
        minDemand=0.5
        ArrivalRate = np.random.uniform(minDemand / 60, maxDemand / 60, NumberOfStop)
        DepartureRate = np.sort(np.random.uniform(0.05, 0.5,NumberOfStop))
        #DepartureRate = np.linspace(0.05, 0.5,NumberOfStop)
        DepartureRate[0]=0
        TrafficSpeed=14
        #Initialise the model parameters
        model_params = {
            "dt": 10,
            "minDemand":minDemand,
            "NumberOfStop": NumberOfStop,
            "LengthBetweenStop": 2000,
```

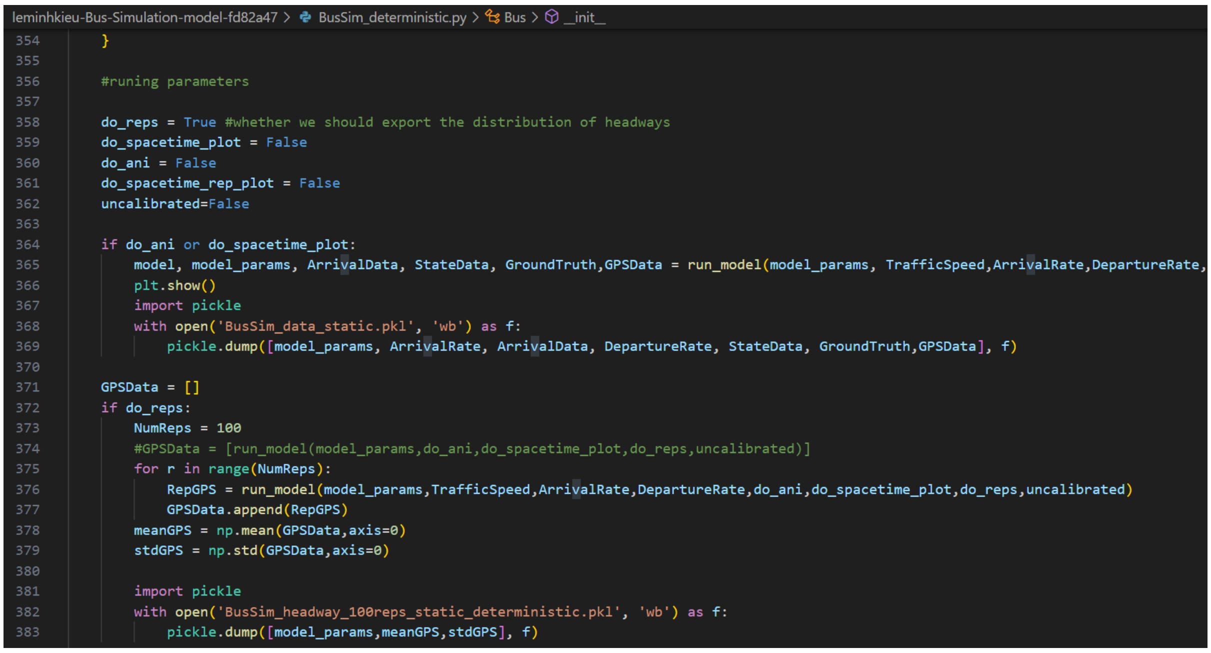


```python
            name0 = ['./Data/Realtime_data_static_maxDemand_',str(maxDemand),'.pkl']
            str1 = ''.join(name0)
            with open(str1,'wb') as f2:
                pickle.dump([model_params,t,x,GroundTruth],f2)
```



```python
    }
    GPSData=[]
    for r in range(NumReps):
        RepGPS = run_model(model_params,TrafficSpeed,ArrivalRate,DepartureRate,do_ani,do_spacetime_plot,do_reps,uncalibrated)
        GPSData.append(RepGPS)
    meanGPS = np.mean(GPSData,axis=0)
    stdGPS = np.std(GPSData,axis=0)
    name0 = ['./Data/Historical_data_static_maxDemand_',str(maxDemand),'.pkl']
    str1 = ''.join(name0)
    with open(str1,'wb') as f2:
        pickle.dump([model_params,meanGPS,stdGPS],f2)
```

There is also file generation possible existing in BusSim_determinsitic.py. However the filenames in this generation do not seem to be present in the original files provided with the codebase.



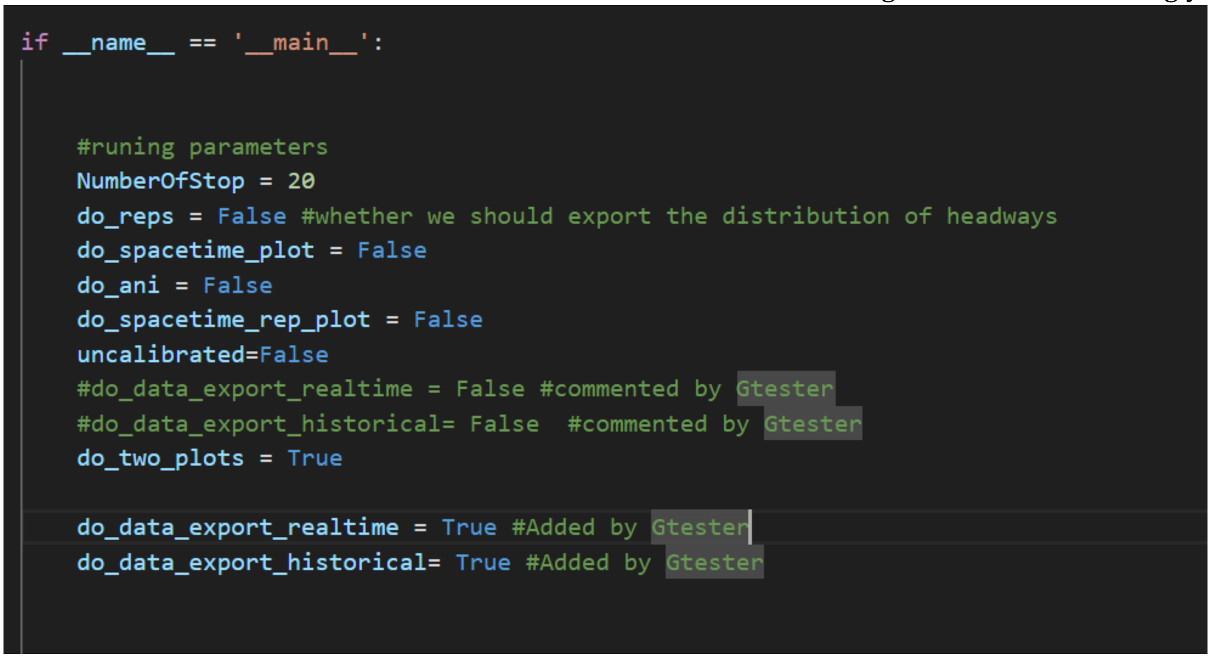
leminhkieu-Bus-Simulation-model-fd82a47 > BusSim_deterministic.py > Bus > __init__

```python
    }

    #runing parameters

    do_reps = True #whether we should export the distribution of headways
    do_spacetime_plot = False
    do_ani = False
    do_spacetime_rep_plot = False
    uncalibrated=False

    if do_ani or do_spacetime_plot:
        model, model_params, ArrivalData, StateData, GroundTruth,GPSData = run_model(model_params, TrafficSpeed,ArrivalRate,DepartureRate,
        plt.show()
        import pickle
        with open('BusSim_data_static.pkl', 'wb') as f:
            pickle.dump([model_params, ArrivalRate, ArrivalData, DepartureRate, StateData, GroundTruth,GPSData], f)

    GPSData = []
    if do_reps:
        NumReps = 100
        #GPSData = [run_model(model_params,do_ani,do_spacetime_plot,do_reps,uncalibrated)]
        for r in range(NumReps):
            RepGPS = run_model(model_params,TrafficSpeed,ArrivalRate,DepartureRate,do_ani,do_spacetime_plot,do_reps,uncalibrated)
            GPSData.append(RepGPS)
        meanGPS = np.mean(GPSData,axis=0)
        stdGPS = np.std(GPSData,axis=0)

        import pickle
        with open('BusSim_headway_100reps_static_deterministic.pkl', 'wb') as f:
            pickle.dump([model_params,meanGPS,stdGPS], f)
```

Both BusSim_stochastic.py and BusSim_deterministic.py are static model compared to the dynamic BusSim_truth.py.

The static models are supposed to be calibrated against synthetic GPS data generated by the BusSim_truth model according to the publication. So why is it that the calibration data is generated by the static stochastic model BusSim_stochastic.py.

I ran BusSim_stochastic.py to generate the necessary files. A figure was produced and saved to the general directory. Figure was not saved to the figures folder. The data required was not produced. Specific parameters have to be set to true to generate the necessary data. These lines were commented out and changed accordingly.

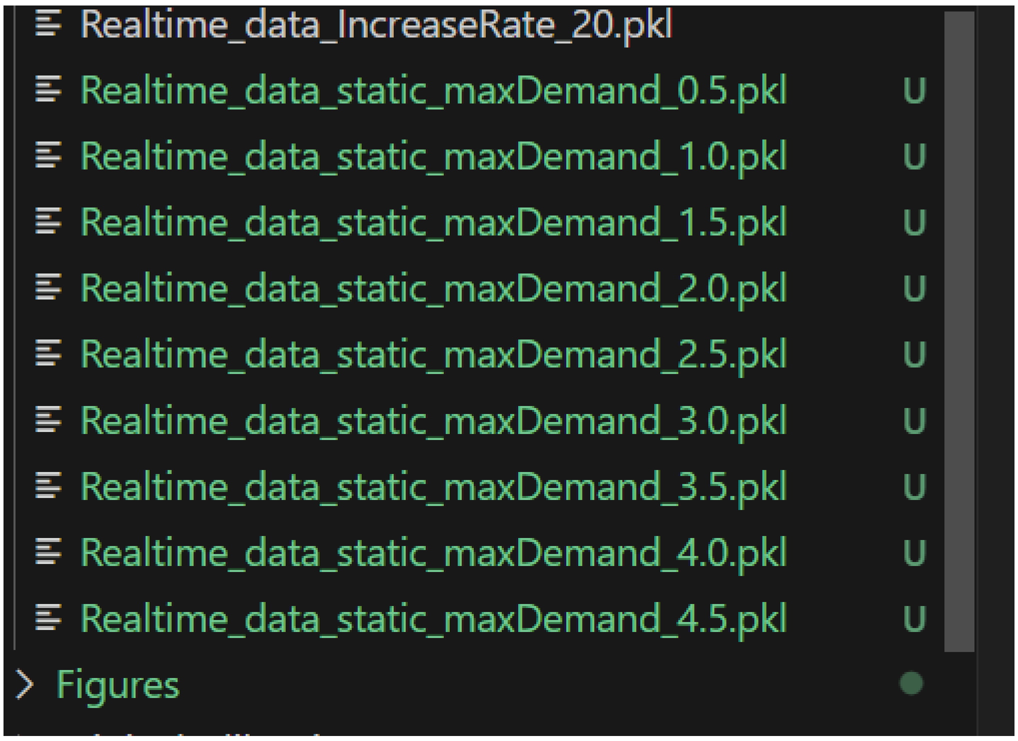


```
if __name__ == '__main__':


    #runing parameters
    NumberOfStop = 20
    do_reps = False #whether we should export the distribution of headways
    do_spacetime_plot = False
    do_ani = False
    do_spacetime_rep_plot = False
    uncalibrated=False
    #do_data_export_realtime = False #commented by Gtester
    #do_data_export_historical= False  #commented by Gtester
    do_two_plots = True

    do_data_export_realtime = True #Added by Gtester
    do_data_export_historical= True #Added by Gtester
```

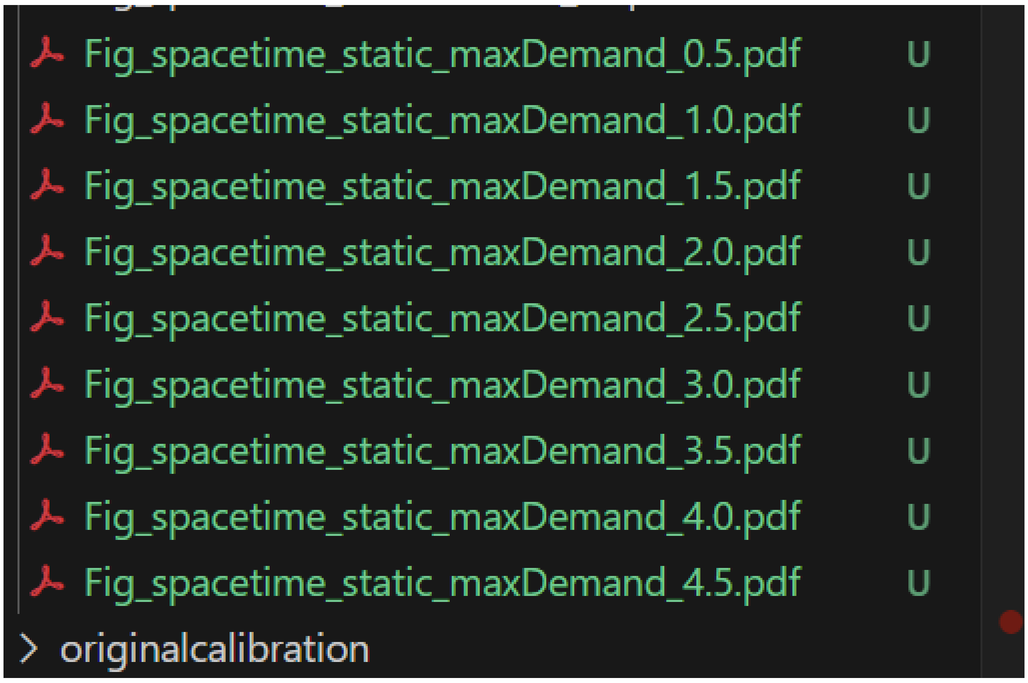
The script was then rerun.

Multiple plots were produced during this run and saved to the Figures/ folder. The figures appeared in popup windows during the run which I had to close continuously.

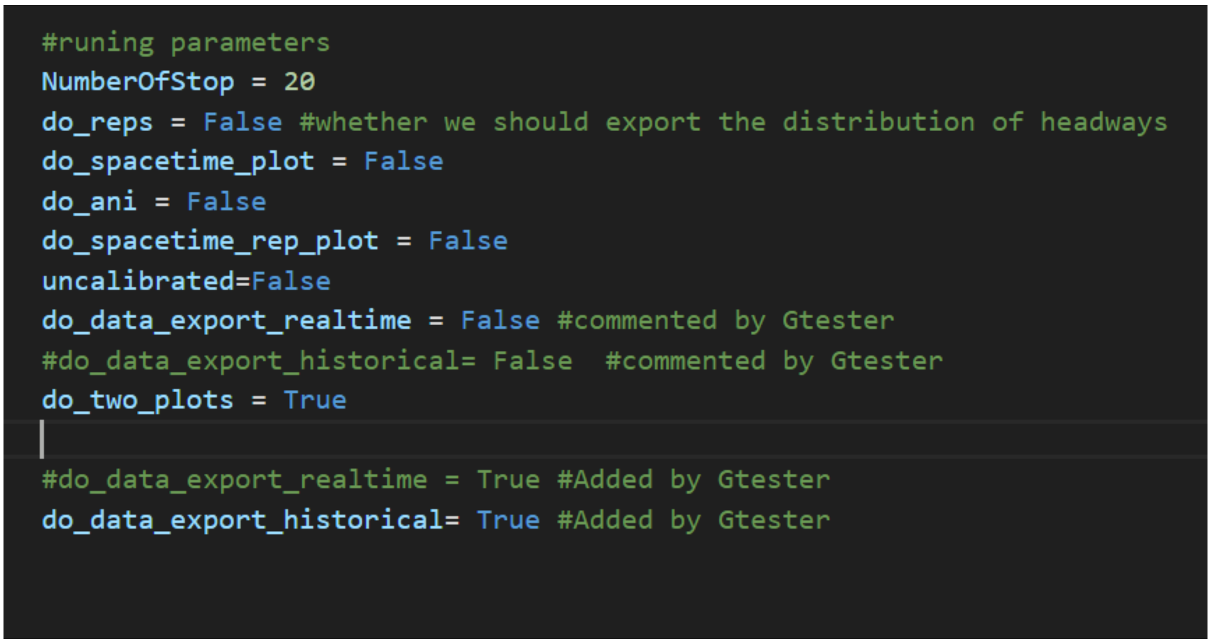
These files were generated and saved to Data/.



```
Realtime_data_IncreaseRate_20.pkl
Realtime_data_static_maxDemand_0.5.pkl    U
Realtime_data_static_maxDemand_1.0.pkl    U
Realtime_data_static_maxDemand_1.5.pkl    U
Realtime_data_static_maxDemand_2.0.pkl    U
Realtime_data_static_maxDemand_2.5.pkl    U
Realtime_data_static_maxDemand_3.0.pkl    U
Realtime_data_static_maxDemand_3.5.pkl    U
Realtime_data_static_maxDemand_4.0.pkl    U
Realtime_data_static_maxDemand_4.5.pkl    U
> Figures
```


```
Fig_spacetime_static_maxDemand_0.5.pdf   U
Fig_spacetime_static_maxDemand_1.0.pdf   U
Fig_spacetime_static_maxDemand_1.5.pdf   U
Fig_spacetime_static_maxDemand_2.0.pdf   U
Fig_spacetime_static_maxDemand_2.5.pdf   U
Fig_spacetime_static_maxDemand_3.0.pdf   U
Fig_spacetime_static_maxDemand_3.5.pdf   U
Fig_spacetime_static_maxDemand_4.0.pdf   U
Fig_spacetime_static_maxDemand_4.5.pdf   U
> originalcalibration
```

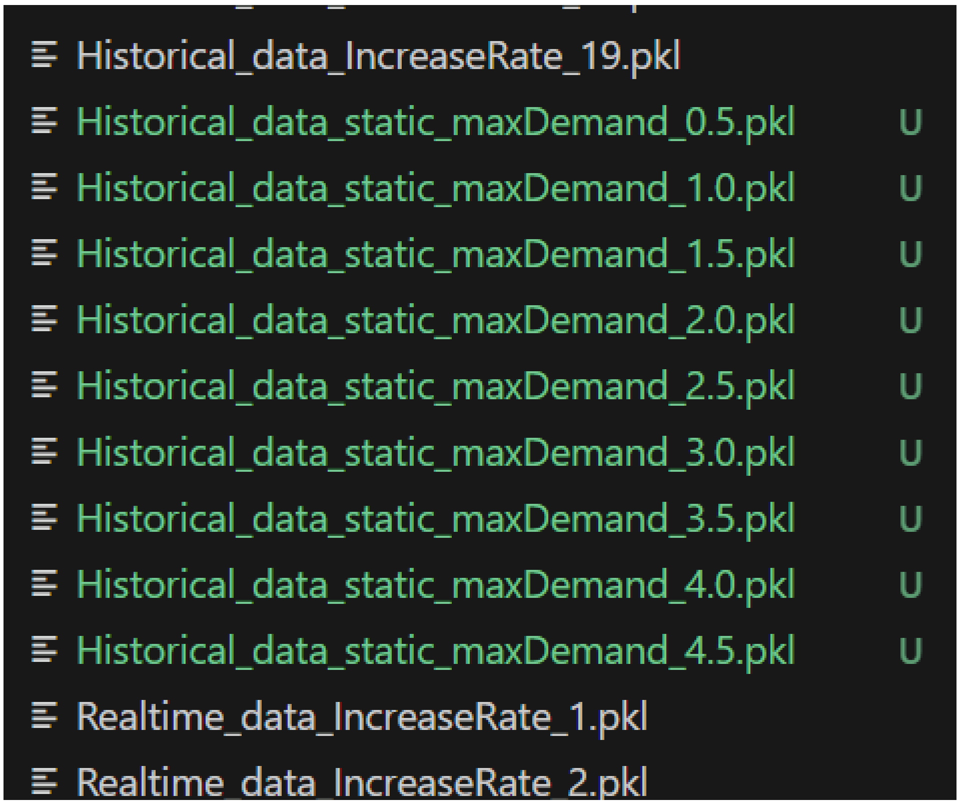
For some reason the historical static max demand data was not generated so the code was changed to as follows and rerun.


```
#runing parameters
NumberOfStop = 20
do_reps = False #whether we should export the distribution of headways
do_spacetime_plot = False
do_ani = False
do_spacetime_rep_plot = False
uncalibrated=False
do_data_export_realtime = False #commented by Gtester
#do_data_export_historical= False  #commented by Gtester
do_two_plots = True

#do_data_export_realtime = True #Added by Gtester
do_data_export_historical= True #Added by Gtester
```

Historical static max demand data now generated.

The error was resolved and step 3.1 cell now executes.

Calibration step will likely take some time.

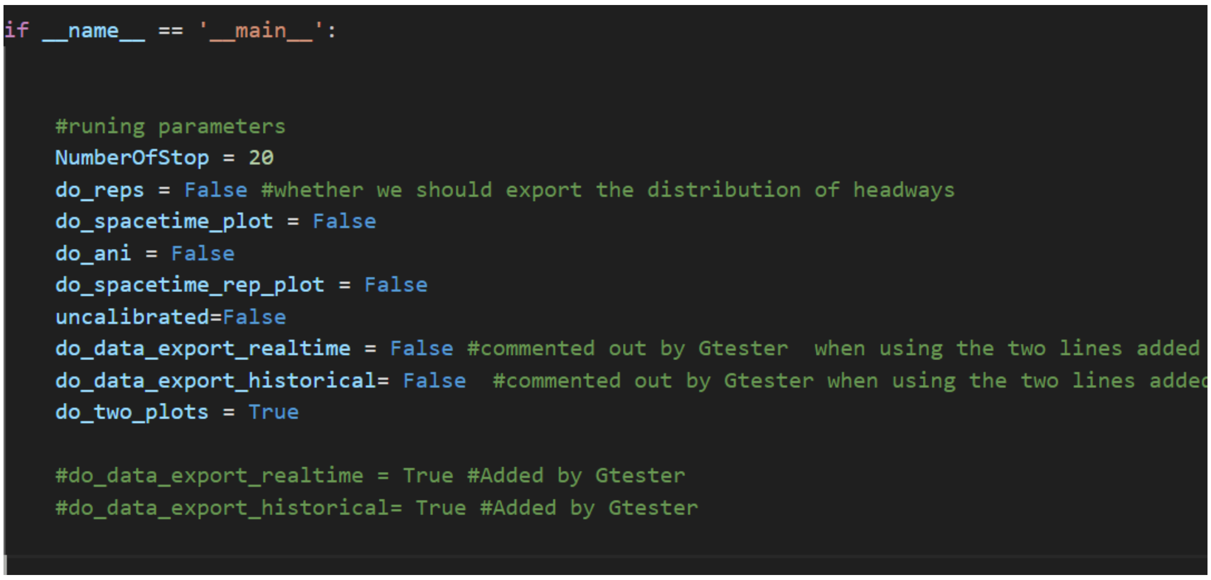


```
Historical_data_IncreaseRate_19.pkl
Historical_data_static_maxDemand_0.5.pkl     U
Historical_data_static_maxDemand_1.0.pkl     U
Historical_data_static_maxDemand_1.5.pkl     U
Historical_data_static_maxDemand_2.0.pkl     U
Historical_data_static_maxDemand_2.5.pkl     U
Historical_data_static_maxDemand_3.0.pkl     U
Historical_data_static_maxDemand_3.5.pkl     U
Historical_data_static_maxDemand_4.0.pkl     U
Historical_data_static_maxDemand_4.5.pkl     U
Realtime_data_IncreaseRate_1.pkl
Realtime_data_IncreaseRate_2.pkl
```

No new figures were generated from this execution of BusSim_stochastic.py.

Note: BusSim_determinsitic.py also seems to capable of generating data but this will not be executed until such data is required by the simulation experiments.

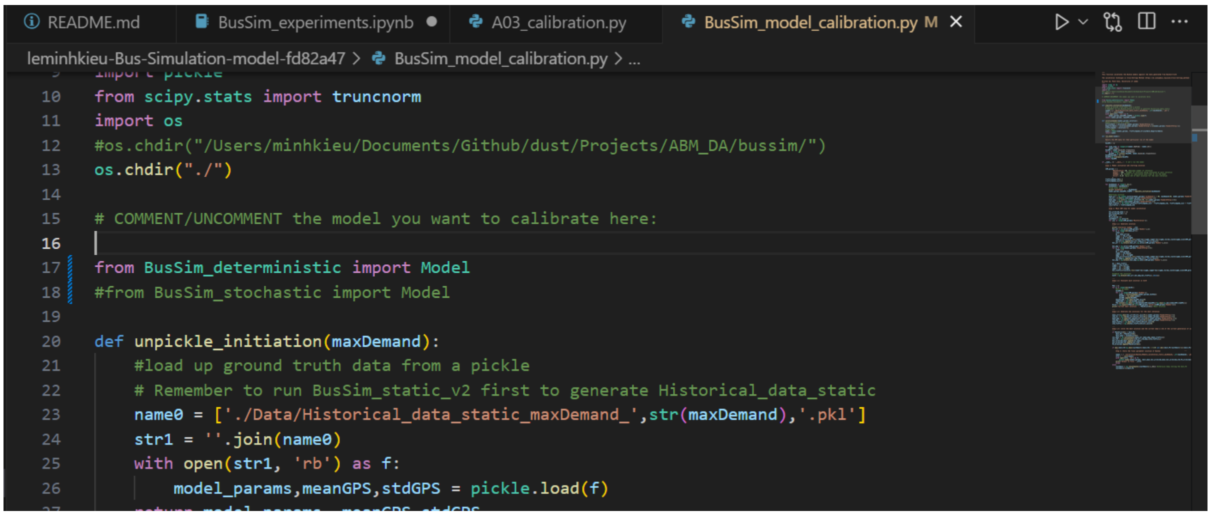
Returned the BusSim_stochastic.py code to its original setup.



```
if __name__ == '__main__':


    #runing parameters
    NumberOfStop = 20
    do_reps = False #whether we should export the distribution of headways
    do_spacetime_plot = False
    do_ani = False
    do_spacetime_rep_plot = False
    uncalibrated=False
    do_data_export_realtime = False #commented out by Gtester  when using the two lines added
    do_data_export_historical= False  #commented out by Gtester when using the two lines added
    do_two_plots = True

    #do_data_export_realtime = True #Added by Gtester
    #do_data_export_historical= True #Added by Gtester
```

Further notes:

Successfully calibrated BusSim_stochastic.py. Still need to calibrate BusSim_determinsitic.py. In BusSim_model_calibration.py the desired model, BusSim_deterministic, was uncommented/commented as instructed in the comments.


README.md
BusSim_experiments.ipynb
A03_calibration.py
BusSim_model_calibration.py M
leminhkieu-Bus-Simulation-model-fd82a47 > BusSim_model_calibration.py > ...

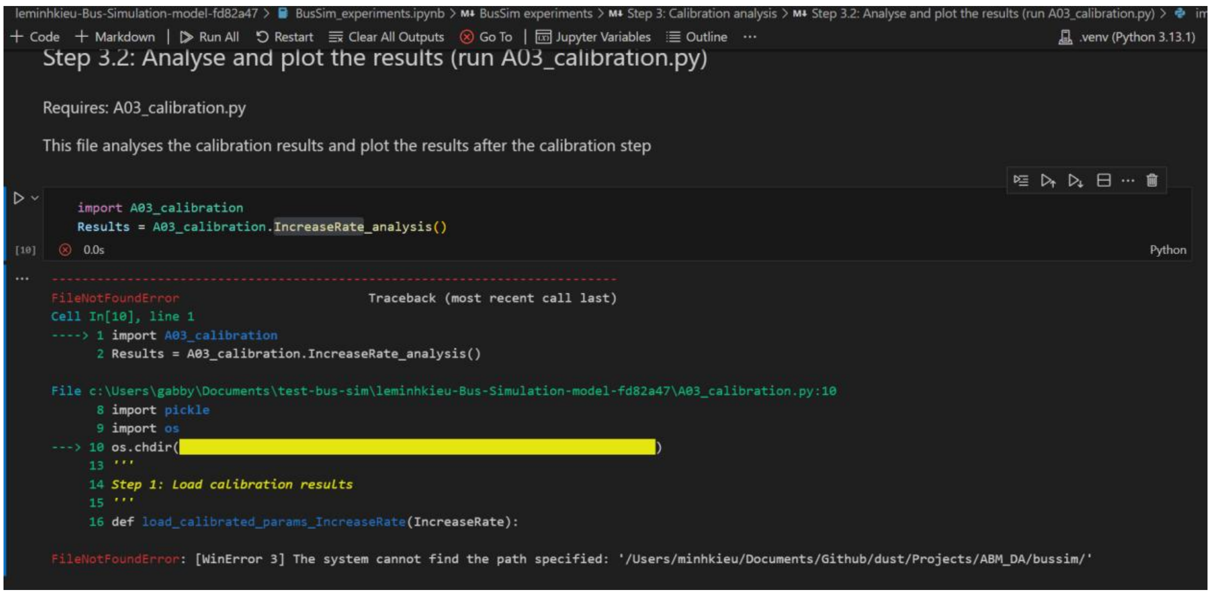
```
from scipy.stats import truncnorm
import os
#os.chdir("/Users/minhkieu/Documents/Github/dust/Projects/ABM_DA/bussim/")
os.chdir("./")

# COMMENT/UNCOMMENT the model you want to calibrate here:

from BusSim_deterministic import Model
#from BusSim_stochastic import Model

def unpickle_initiation(maxDemand):
    #load up ground truth data from a pickle
    # Remember to run BusSim_static_v2 first to generate Historical_data_static
    name0 = ['./Data/Historical_data_static_maxDemand_',str(maxDemand),'.pkl']
    str1 = ''.join(name0)
    with open(str1, 'rb') as f:
        model_params,meanGPS,stdGPS = pickle.load(f)
```

Step 3.1 cell was then re-run to calibrate BusSim_deterministic model.

However, because the code does not change the naming convention, BusSim_deterministic run overwrote the first datafile for BusSim_stochastic.py marked BusSim_Model2_calibration_static_maxdemand. Upon noticing this change, I discarded the change and reverted to the old file using the version control i.e. git.

Re-execute BusSim_deterministic.py which will save its files as BusSim_Model1_calibration_static_maxdemand.py.

Error encountered in 'step 3.2 analyse and plot results'. FileNotFoundError. Similar to previous errors that have been encountered due to the use of absolute file paths.


leminhkieu-Bus-Simulation-model-fd82a47 > BusSim_experiments.ipynb > M↓ BusSim experiments > M↓ Step 3: Calibration analysis > M↓ Step 3.2: Analyse and plot the results (run A03_calibration.py) > im

Step 3.2: Analyse and plot the results (run A03_calibration.py)

Requires: A03_calibration.py

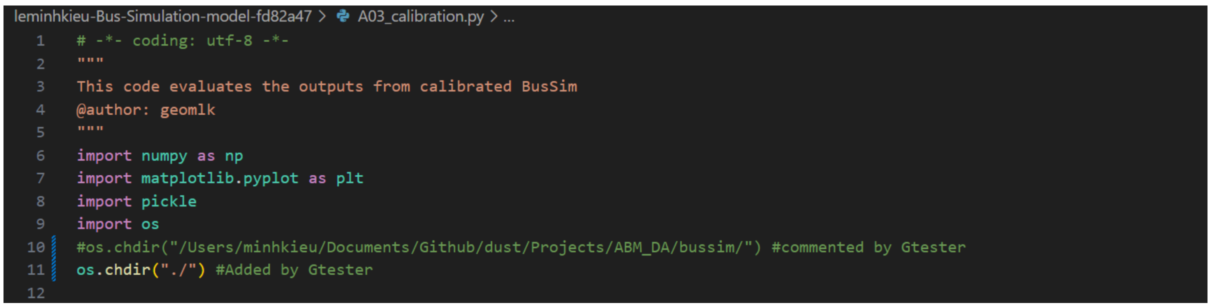
This file analyses the calibration results and plot the results after the calibration step

```
import A03_calibration
Results = A03_calibration.IncreaseRate_analysis()
```

[10] 0.0s

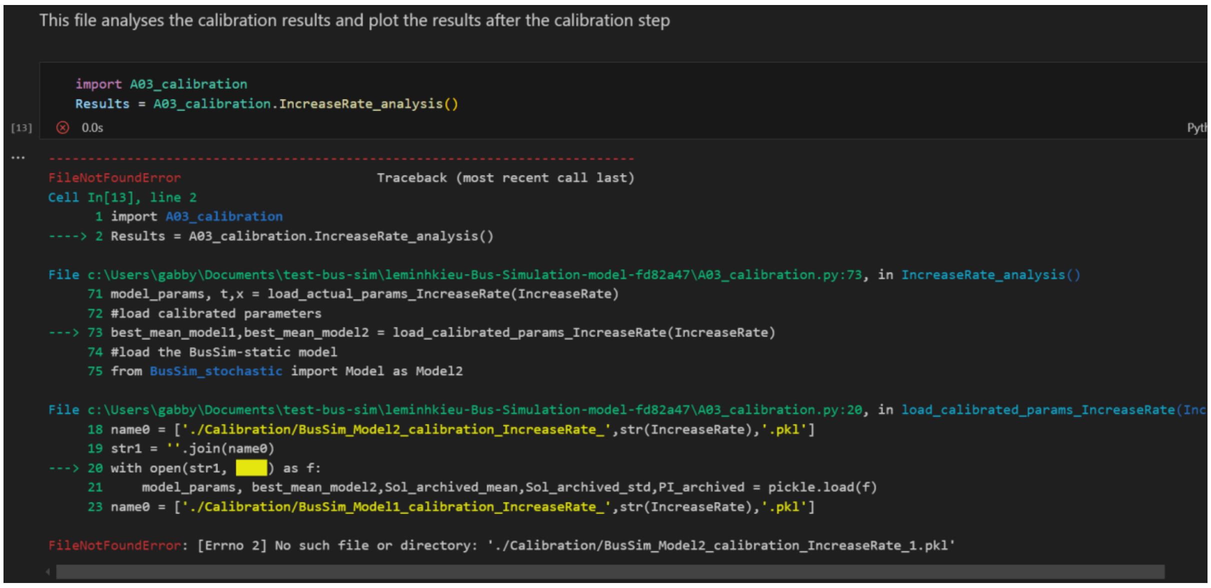
```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[10], line 1
----> 1 import A03_calibration
      2 Results = A03_calibration.IncreaseRate_analysis()

File c:\Users\gabby\Documents\test-bus-sim\leminhkieu-Bus-Simulation-model-fd82a47\A03_calibration.py:10
      8 import pickle
      9 import os
---> 10 os.chdir([illegible])
     13 '''
     14 Step 1: Load calibration results
     15 '''
     16 def load_calibrated_params_IncreaseRate(IncreaseRate):

FileNotFoundError: [WinError 3] The system cannot find the path specified: '/Users/minhkieu/Documents/Github/dust/Projects/ABM_DA/bussim/'
```

Fixed the error by changing the file path.


leminhkieu-Bus-Simulation-model-fd82a47 > A03_calibration.py > ...

```
# -*- coding: utf-8 -*-
"""
This code evaluates the outputs from calibrated BusSim
@author: geomlk
"""
import numpy as np
import matplotlib.pyplot as plt
import pickle
import os
#os.chdir("/Users/minhkieu/Documents/Github/dust/Projects/ABM_DA/bussim/") #commented by Gtester
os.chdir("./") #Added by Gtester

```

Step 3.2 cell error. Previous cell did not generate the calibration data using the Historical_data_IncreaseRate_XX.pkl data. There is no code included to do this automatically. I will have to read the calibration code and attempt to regenerate the data.

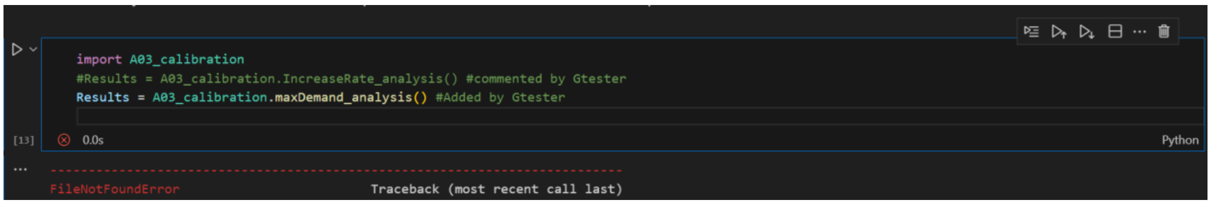

This file analyses the calibration results and plot the results after the calibration step

```
import A03_calibration
Results = A03_calibration.IncreaseRate_analysis()
```

[13] 0.0s

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[13], line 2
      1 import A03_calibration
----> 2 Results = A03_calibration.IncreaseRate_analysis()

File c:\Users\gabby\Documents\test-bus-sim\leminhkieu-Bus-Simulation-model-fd82a47\A03_calibration.py:73, in IncreaseRate_analysis()
     71 model_params, t,x = load_actual_params_IncreaseRate(IncreaseRate)
     72 #load calibrated parameters
---> 73 best_mean_model1,best_mean_model2 = load_calibrated_params_IncreaseRate(IncreaseRate)
     74 #load the BusSim-static model
     75 from BusSim_stochastic import Model as Model2

File c:\Users\gabby\Documents\test-bus-sim\leminhkieu-Bus-Simulation-model-fd82a47\A03_calibration.py:20, in load_calibrated_params_IncreaseRate(Inc
     18 name0 = ['./Calibration/BusSim_Model2_calibration_IncreaseRate_',str(IncreaseRate),'.pkl']
     19 str1 = ''.join(name0)
---> 20 with open(str1, [illegible]) as f:
     21     model_params, best_mean_model2,Sol_archived_mean,Sol_archived_std,PI_archived = pickle.load(f)
     23 name0 = ['./Calibration/BusSim_Model1_calibration_IncreaseRate_',str(IncreaseRate),'.pkl']

FileNotFoundError: [Errno 2] No such file or directory: './Calibration/BusSim_Model2_calibration_IncreaseRate_1.pkl'
```

Note there are two analyses that can be run IncreaseRate analysis and MaxDemand analysis. If I change the command from IncreaseRate analysis to Maxdeman analysis from A03_calibration.py. The corresponding figures should be generated.

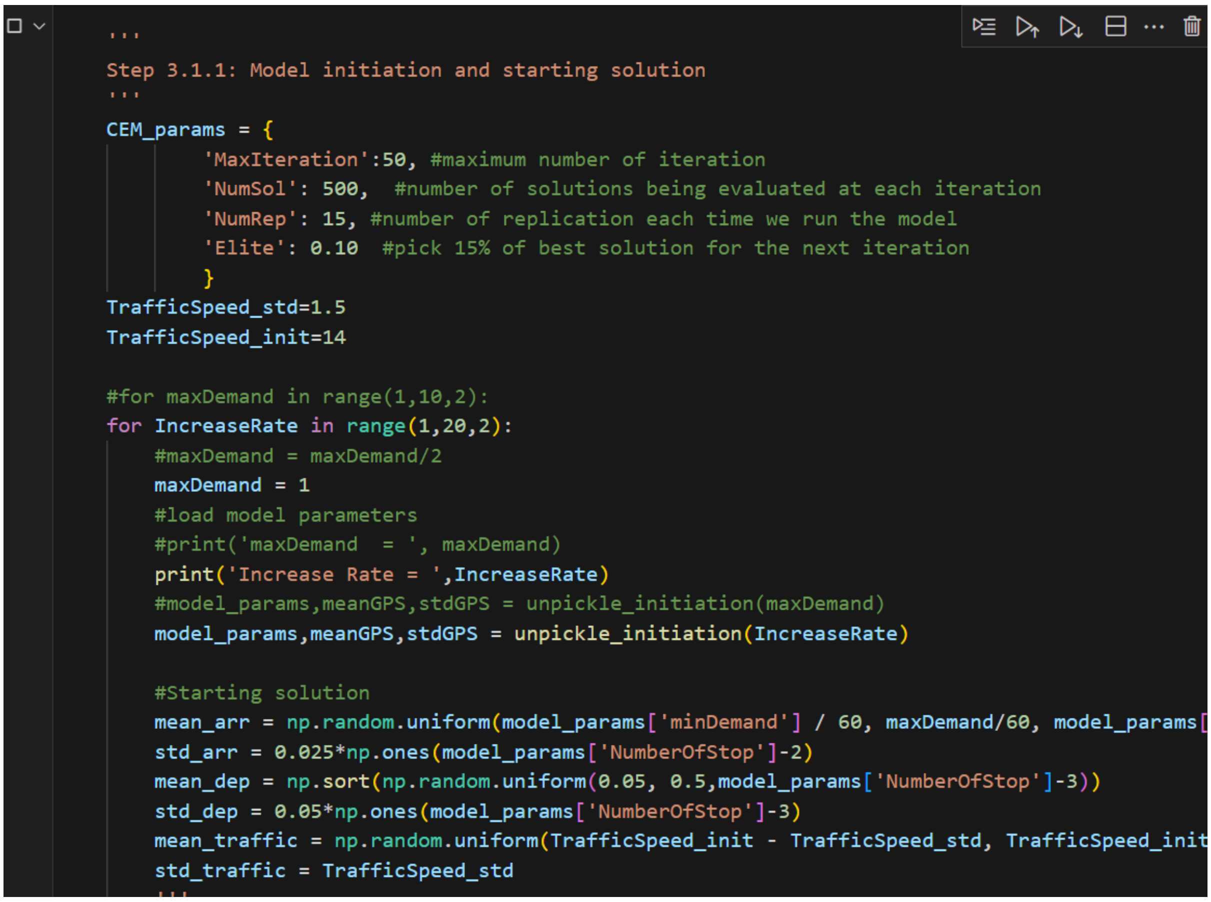


```
import A03_calibration
#Results = A03_calibration.IncreaseRate_analysis() #commented by Gtester
Results = A03_calibration.maxDemand_analysis() #Added by Gtester

[13]  0.0s                                                    Python
...
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
```

Attempted solution to the IncreaseRate_analysis missing data problem:

When BusSim_truth was sued to generate the historical_increaserate.pkl files maxDemand was set to 1 and the loop was for IncreaseRate (1,21).

I will copy the calibration cell and change the for loop to reflect the IncreaseRate for loop previously used.



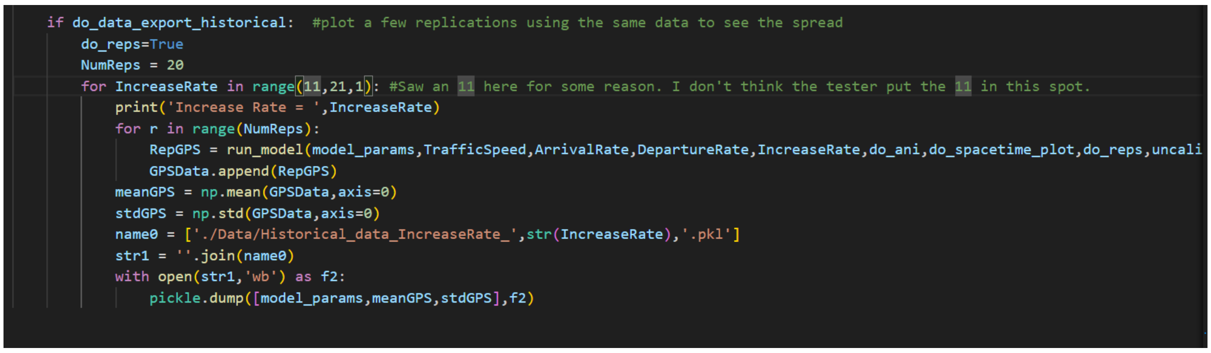
```
'''
Step 3.1.1: Model initiation and starting solution
'''
CEM_params = {
        'MaxIteration':50, #maximum number of iteration
        'NumSol': 500,  #number of solutions being evaluated at each iteration
        'NumRep': 15, #number of replication each time we run the model
        'Elite': 0.10  #pick 15% of best solution for the next iteration
        }
TrafficSpeed_std=1.5
TrafficSpeed_init=14

#for maxDemand in range(1,10,2):
for IncreaseRate in range(1,20,2):
    #maxDemand = maxDemand/2
    maxDemand = 1
    #load model parameters
    #print('maxDemand  = ', maxDemand)
    print('Increase Rate = ',IncreaseRate)
    #model_params,meanGPS,stdGPS = unpickle_initiation(maxDemand)
    model_params,meanGPS,stdGPS = unpickle_initiation(IncreaseRate)

    #Starting solution
    mean_arr = np.random.uniform(model_params['minDemand'] / 60, maxDemand/60, model_params[
    std_arr = 0.025*np.ones(model_params['NumberOfStop']-2)
    mean_dep = np.sort(np.random.uniform(0.05, 0.5,model_params['NumberOfStop']-3))
    std_dep = 0.05*np.ones(model_params['NumberOfStop']-3)
    mean_traffic = np.random.uniform(TrafficSpeed_init - TrafficSpeed_std, TrafficSpeed_init
    std_traffic = TrafficSpeed_std
```

Created a cell that runs the calibration using the historical_data_IncreaseRate.pkl files with maxDemand = 1 which is the same setting as BusSim_truth when the hisotrical data was created. Executing the cell or BusSim_stochastic first and results should save as Model2 calibration IncreaseRate results. The cell will be re-executed for BusSim-deterministic.

Calibrating using IncreaseRate files for BusSim_stochastic.py took 274 minutes 36.5 seconds, Calibrating using IncreaseRate for BusSim_deterministic.py. Due to laptop shutdown overnight, did not get to record the time taken for the cell to run but it was a long time. Completed the cell execution for IncreaseRate 15, 17 and 19 on 09/04/2025. Took 70 minutes 26.9 seconds.

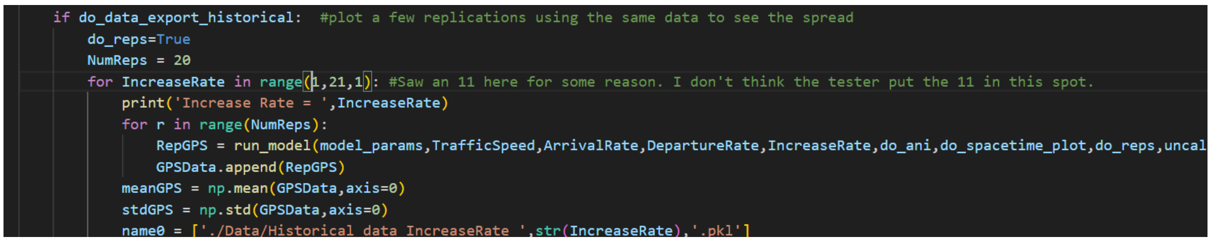
09/04/2025

Possible error (Not sure)



```
if do_data_export_historical:  #plot a few replications using the same data to see the spread
    do_reps=True
    NumReps = 20
    for IncreaseRate in range(11,21,1): #Saw an 11 here for some reason. I don't think the tester put the 11 in this spot.
        print('Increase Rate = ',IncreaseRate)
        for r in range(NumReps):
            RepGPS = run_model(model_params,TrafficSpeed,ArrivalRate,DepartureRate,IncreaseRate,do_ani,do_spacetime_plot,do_reps,uncali
            GPSData.append(RepGPS)
        meanGPS = np.mean(GPSData,axis=0)
        stdGPS = np.std(GPSData,axis=0)
        name0 = ['./Data/Historical_data_IncreaseRate_',str(IncreaseRate),'.pkl']
        str1 = ''.join(name0)
        with open(str1,'wb') as f2:
            pickle.dump([model_params,meanGPS,stdGPS],f2)
```

In BusSim_truth main function, saw an 11 in this function instead of a 1. Thought this was strange. Changed it to 1.



```
if do_data_export_historical:  #plot a few replications using the same data to see the spread
    do_reps=True
    NumReps = 20
    for IncreaseRate in range(1,21,1): #Saw an 11 here for some reason. I don't think the tester put the 11 in this spot.
        print('Increase Rate = ',IncreaseRate)
        for r in range(NumReps):
            RepGPS = run_model(model_params,TrafficSpeed,ArrivalRate,DepartureRate,IncreaseRate,do_ani,do_spacetime_plot,do_reps,uncal
            GPSData.append(RepGPS)
        meanGPS = np.mean(GPSData,axis=0)
        stdGPS = np.std(GPSData,axis=0)
        name0 = ['./Data/Historical_data_IncreaseRate_',str(IncreaseRate),'.pkl']
```

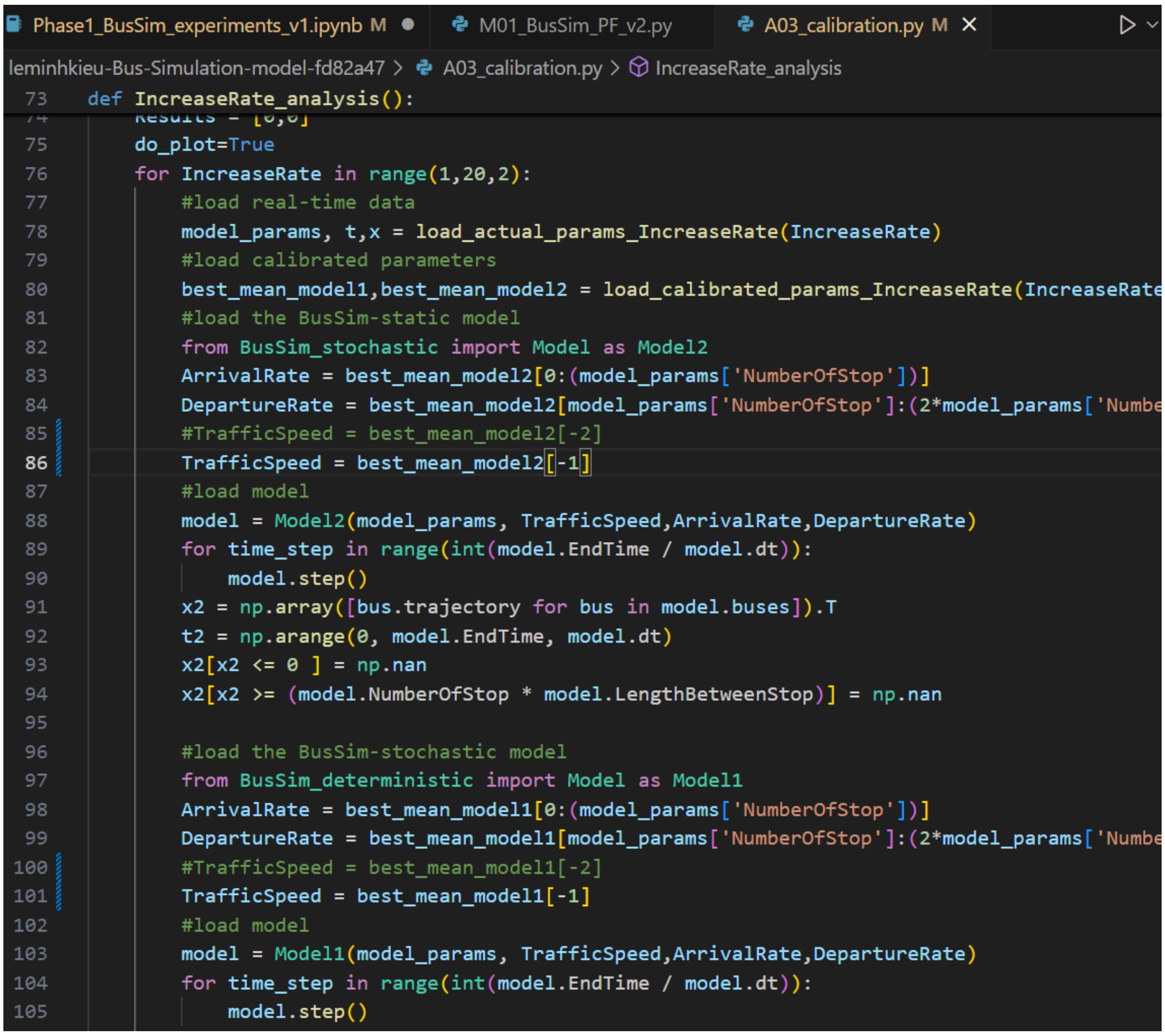
There was a persistent error in the calibration of files in which the plots produced for scenario 2 were erroneous. It was thought that the calibration data was not being generated properly and so time was input to investigate why this was occurring and reruns were done that took a lengthy period of time.

After examining the code and the difference between the original calibration files and my own calibration files, changes were noted.

The original files: 'BusSim_Model2_calibration_IncreaseRate_X' contained arrays of bestmean value of size 42.

The output from my runs for these files were of size 41.

After examining the publication as well as delving deeper into the CEM code. I noticed that the solutions drawn for the traffic parameter were the last solutions in the dataset I generated. Therefore in the setup to run busmodelstochastic and bussim-deterministic there was an indexing error that utilised traffic data from the 2nd to last location in the array instead of the final element of the array. What this final value is and where it came form remains to be understood. But once this change was made such that the traffic data was intialised with the right value. The plots came to resemble something closer to the publication. Thought the sizes of the different datasets will make it difficult to compare datasets for testing.


Phase1_BusSim_experiments_v1.ipynb M
M01_BusSim_PF_v2.py
A03_calibration.py M
leminhkieu-Bus-Simulation-model-fd82a47 > A03_calibration.py > IncreaseRate_analysis

```python
def IncreaseRate_analysis():
    Results = [0,0]
    do_plot=True
    for IncreaseRate in range(1,20,2):
        #load real-time data
        model_params, t,x = load_actual_params_IncreaseRate(IncreaseRate)
        #load calibrated parameters
        best_mean_model1,best_mean_model2 = load_calibrated_params_IncreaseRate(IncreaseRate
        #load the BusSim-static model
        from BusSim_stochastic import Model as Model2
        ArrivalRate = best_mean_model2[0:(model_params['NumberOfStop'])]
        DepartureRate = best_mean_model2[model_params['NumberOfStop']:(2*model_params['Numbe
        #TrafficSpeed = best_mean_model2[-2]
        TrafficSpeed = best_mean_model2[-1]
        #load model
        model = Model2(model_params, TrafficSpeed,ArrivalRate,DepartureRate)
        for time_step in range(int(model.EndTime / model.dt)):
            model.step()
        x2 = np.array([bus.trajectory for bus in model.buses]).T
        t2 = np.arange(0, model.EndTime, model.dt)
        x2[x2 <= 0 ] = np.nan
        x2[x2 >= (model.NumberOfStop * model.LengthBetweenStop)] = np.nan

        #load the BusSim-stochastic model
        from BusSim_deterministic import Model as Model1
        ArrivalRate = best_mean_model1[0:(model_params['NumberOfStop'])]
        DepartureRate = best_mean_model1[model_params['NumberOfStop']:(2*model_params['Numbe
        #TrafficSpeed = best_mean_model1[-2]
        TrafficSpeed = best_mean_model1[-1]
        #load model
        model = Model1(model_params, TrafficSpeed,ArrivalRate,DepartureRate)
        for time_step in range(int(model.EndTime / model.dt)):
            model.step()
```

This same issue was present in the M01_PF_v2 file. This same correction was made to intialise the traffic parameter properly.

## 1.32 Test 2 – Check if codebase runs with the provided data files

The following errors were encountered in trying to execute the BusSim_experiments.ipynb notebook that contained the case study's experiment workflow.

Step 1

- Cells executed without error.
- Data files produced:
    - "Historical_data_IncreaseRate_X.pkl" , X in [1,19]
        - 19 historical data files regenerated but there are 20 historical data files in the provided Data/ folder.
    - "Realtime_data_IncreaseRate_X.pkl" X in [1,20] produced.
        - 20 files produced.
    - Fig_spacetime_IncreaseRate_X , X in [1,20] produced.

Step 2

- FileNotFoundError.
    - Change path in A02_doing_nothing_analysis.
- Cell executes after fix was applied.
- Files generated:
    - Fig_do_nothing_IncreaseRate_X.pdf where X in [1,19]
    - Fig_do_nothing_results.pdf
    - No data files produced or saved to Data/

Step 3.1

- FileNotFoundError
    - Change path in BusSim_model_calibration
- Cell started executing but takes a while so stopped it prematurely but it was executing.

Notes on step 3:

- The calibration is set up for BusSim_stochastic by default.

Step 3.2

- FileNotFoundError.
    - Change path in A03_calibration
- Cell executed successfully.
- Files generated:
    - Fig_calibration_IncreaseRate_X.pdf in X range(1,19,2).
    - Fig_calibration_results_IncreaseRate

Step 4

- Executes successfully without error. No path change required.
- Files generated:
    - Fig_PF_IncreaseRate_X.pdf , X in range(1,19,2)
    - Fig_calibration_results_IncreaseRate.pdf

# Appendix D: Particle Filter Sensitivity Analysis Supporting Material

## 1.33 Particle Filter Algorithm According to Publication and Code

The particle filter algorithm transcribed below is based on the case study's implemented code.

Initialise set of $N_p$ particles, $P_0 = \left\{\langle X_0^{(i)}, w_0^{(i)}\rangle, i = 1, \dots, N_p\right\}, X_0^{(i)} = \left\{O_t^{(i)}, S_t^{(i)}\right\}, t = 0$

Initialise $self.states = \left\{X_0^{(i)}, i = 1, \dots, N_p\right\}$

**PREDICT**

For $i = 1{:}\,N_p$

- Assign previous state estimate to particle state
  $X_t^{(i)} = X_{t-1}^{(i)}$
- Advance the timestep of the particle
  $X_{t+1}^{(i)} = f\left(X_t^{(i)}\right)$

$self.states = \left\{X_{t+1}^{(i)}, i = 1, \dots, N_p\right\}$

$t = t + 1$

**IMPORTANCE WEIGHTING**

- Take the observation vectors of each particle
  $states = \left\{O_t^{(i)}, i = 1, \dots, N_p\right\}$
- Calculate the distance metric using Frobenius norm of the difference between each particle's observation vector and the measured state
- $d_t^{(i)} = \left\|O_t^{(i)} - y_t^{(i)}\right\|_2 \;\forall\, i \in \{1, \dots, N_p\}$
- Weight computation: $w_t^{(i)} = \frac{1}{\max\left(d_t^{(i)}, 1E-99\right)}$
- Normalise weights $w_t^{(i)} = \frac{w_t^{(i)}}{\sum_{i=1}^{N_p} w_t^{(i)}}$

**RESAMPLE**

Generate a new set of particles from the current set with the likelihood of drawing a particle proportional to its weight

If $NOT\ (t\ \%\ resample\ window)$

- Draw a random number for each particle

$U^{(i)} = \frac{(i-1)+U}{N_p} \forall\, i \in \left\{1, \dots, N_p\right\}$

- Construct the CDF i.e. an array of cumulative sum of the weights
  - Initialise CDF, $c_0 = 0$

```
        ○ $c^{(i)} = c^{(i-1)} + w_t^{(i)} \ \forall\, i \in \{1, \dots, N_p\}$
While $i < N_p$ and $j < N_p$
    If $U^{(i)} < c^{(j)}$then
    Assign particle to new set of particles $X_t^{(*i)} = X_t^{(j)}$
                    $i = i + 1$
    Else
                    $j = j + 1$
```

**JITTER**

Add a random gaussian white noise to the components of the model parameter vector $S_t$.

$$Arr_m = Arr_m + j \sim \mathcal{N}\left(0, arr_{std}^2\right)$$

$$Dep_m = Dep_m + j \sim \mathcal{N}\left(0, dep_{std}^2\right)$$

$$Traffic_m = Traffic_m + j \sim \mathcal{N}\left(0, traffic_{std}^2\right)$$

```
Endwhile
```