Eerland, W J et al 2017 Cambridge Rocketry Simulator – A Stochastic Six-Degrees-of-Freedom Rocket Flight Simulator. *Journal of Open Research Software,* 5: 5, DOI: https://doi.org/10.5334/jors.137

SOFTWARE METAPAPER

Cambridge Rocketry Simulator – A Stochastic Six-Degrees-of-Freedom Rocket Flight Simulator

Willem J. Eerland, Simon Box and András Sóbester
University of Southampton, GB
Corresponding author: Willem J. Eerland
(w.j.eerland@soton.ac.uk)

The Cambridge Rocketry Simulator can be used to simulate the flight of unguided rockets for both design and operational applications. The software consists of three parts: The first part is a GUI that enables the user to design a rocket. The second part is a verified and peer-reviewed physics model that simulates the rocket flight. This includes a Monte Carlo wrapper to model the uncertainty in the rocket's dynamics and the atmospheric conditions. The third part generates visualizations of the resulting trajectories, including nominal performance and uncertainty analysis, e.g. a splash-down region with confidence bounds. The project is available on SourceForge, and is written in Java (GUI), C++ (simulation core), and Python (visualization). While all parts can be executed from the GUI, the three components share information via XML, accommodating modifications, and re-use of individual components.

Keywords: Java; C++; Python; rocket; simulation; simulator; stochastic; design
Funding Statement: The authors gratefully acknowledge the funding provided under research grant EP/L505067/1 from the Engineering and Physical Sciences Research Council, and the support of industry sponsor Cunning Running Software Ltd. The research data and code generated as part of this study is available at https://doi.org/10.5281/zenodo.161850

(1) Overview

Introduction

The Cambridge Rocketry Simulator is designed for use with unguided rockets including model rockets, High Power Rockets (HPR), and sounding rockets. Typically, these will perform a sub-orbital flight, collect some data, and deploy a parachute for recovery to Earth. The software models the flight dynamics of the rocket in six-degrees-of-freedom and of the parachute descent in three-degrees-of-freedom. It has a range of uses. Firstly, it can guide the design process. For example, it allows the engineer to select the appropriate size motor for a desired apogee; or to design the fins for an appropriate margin of stability; or to optimise the timing of stage separation or parachute deployment. Secondly, it enables the operator to predict the landing location, which helps in determining the required launch safety exclusion zone, as well as facilitating the retrieval of the reusable components of the rocket. In order to map uncertainties in the dynamics and the atmospheric conditions into confidence bounds around the predicted landing location, a Monte Carlo approach is combined with a numerical integration scheme.

Most research performed on predicting the trajectories of projectiles is done by numerical integration. The exception is the work of Chudinov [8], who predicts the trajectories by applying an analytical solution. While this approach is beneficial in terms of computation costs, the prediction is limited to modelling a point mass with a quadratic drag force. At the other end of the spectrum, there are advanced simulators such as that described by Sahu [17], where the authors use advanced coupled computational fluid dynamics/rigid body dynamics to predict the rocket behaviour even as it enters supersonic speeds. However, as the additional computational cost is significant due to the calculation of unsteady aerodynamics associated with supersonic flight, it is computationally costly to perform an uncertainty analysis of parameters via a Monte Carlo method. While there are other methods available for non-linear uncertainty propagation, the Monte Carlo method is capable of handling large state spaces and arbitrary input distributions, making it the most widely used approach. That does not take away that advanced simulations remain invaluable when testing the performance of new control methods, as performed by Gomez and Miikkulainen [12], who test their control method for a finless rocket via simulations. Uncertainty propagation is also useful for Impact Point Prediction (IPP), where the impact point of a projectile is determined while it is in-flight. Yuan et al. [20] incorporates the

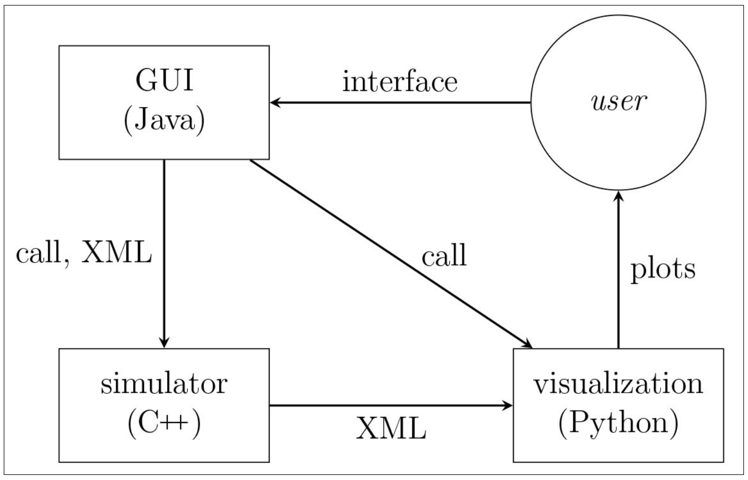
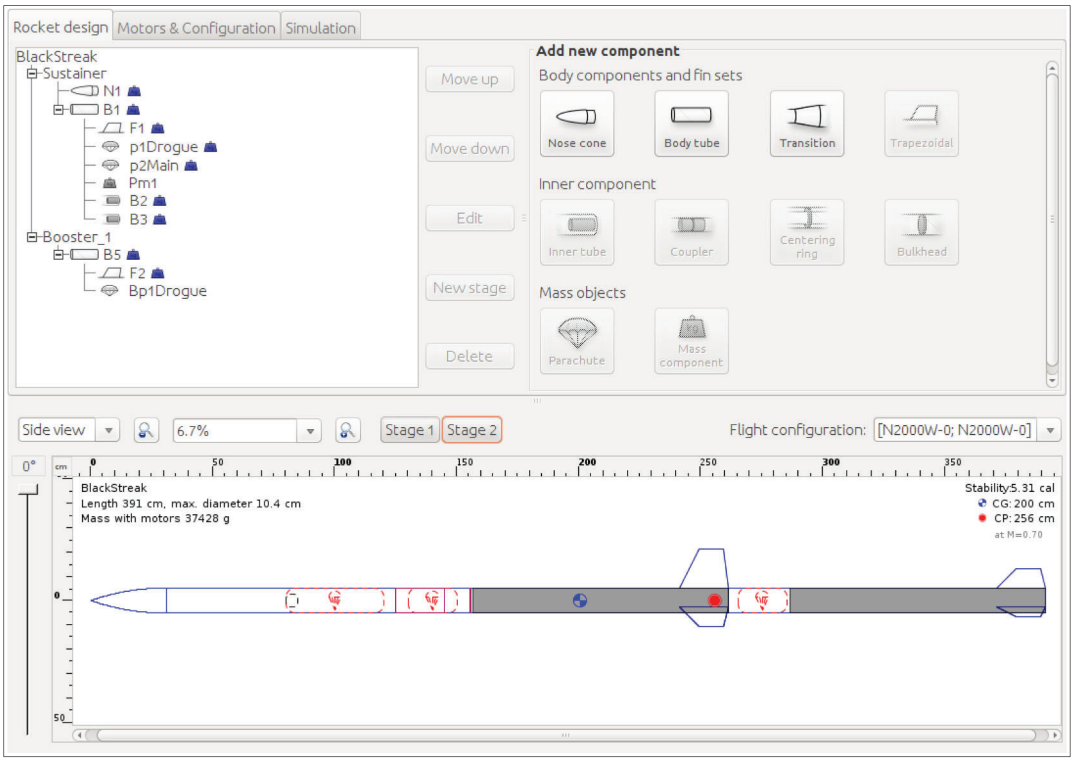


Figure 1: A schematic representation of the three components and the corresponding programming languages.

uncertainty due to the presence of wind to increase the accuracy on the confidence bounds on the landing location. Rogers [16] applies an uncertainty analysis not only to predict the impact point, but also to *guide* the rocket such that it avoids objects and reaches the target location without collision. With the exception of the analytical solution, all research is done via a six-degrees-of-freedom rocket model and parameters describing the projectile's properties.

There are various software packages that facilitate conceptual studies of rocket design, including commercial solutions such as RockSim [1], open-source solutions such as OpenRocket [13], and even computer games such as the Kerbal Space Program [18]. All these packages simulate the performance of a rocket under nominal conditions, returning the expected performance to the user. While RockSim Pro does display confidence bounds around the predicted landing location to the user, it is only available to USA citizens.

The Cambridge Rocketry Simulator has four advantages over these software packages. Firstly, it is free and open-source [6]. Secondly, the physics model is verified, peer-reviewed, and published [5]. Thirdly, by using a Monte Carlo wrapper, it incorporates uncertainties in both rocket dynamics and atmospheric conditions, making it possible to produce a splash-down area with confidence bounds. Finally, the atmospheric model supports a 3-dimensional wind vector, air density, and air temperature, all as a function of altitude. These atmospheric data may be populated from a recent meteorological forecast to maximize the



BlackStreak
Length 391 cm, max. diameter 10.4 cm
Mass with motors 37428 g

Stability: 5.31 cal
CG: 200 cm
CP: 256 cm
at M=0.70

Figure 2: GUI for designing a rocket. It contains all the basic components required to design a rocket.

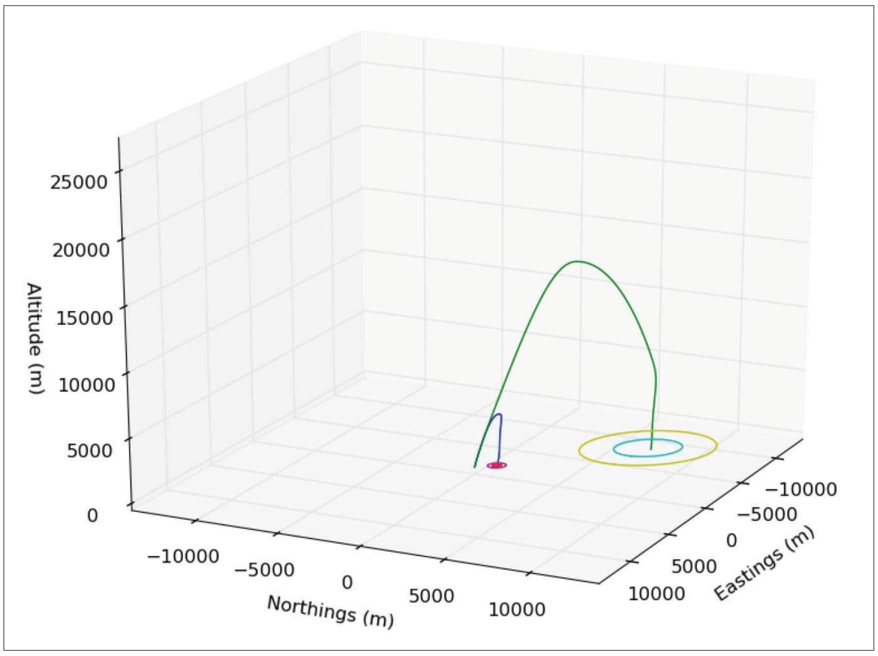


Figure 3: Mean trajectories of both stages of a two-stage rocket including a staged parachute descent. The 1 and 2 σ bounds of confidence in landing position are also shown.

accuracy of predictions. And with the introduction of the third version of the Cambridge Rocketry Simulator, as presented in this paper, parts of the Graphical User Interface (GUI) from Open-Rocket [13] are integrated to assist the user in their rocket design. The software package is useful in predicting the trajectories of rockets in academic activities [3, 19, 7, 10], or when sampling the atmosphere using sounding rockets. Without the prediction of the trajectories, or more specifically, the confidence bounds of the landing location, you may never get permission from the aviation authorities to launch your rocket [9]. Furthermore, the output of the simulator, the rocket trajectories, have been used in the trajectory modelling research described by Eerland, Box, and Sóbester [11].

The remainder of this paper covers the implementation and architecture, quality control, the availability, and reuse potential of the software.

Implementation and architecture

The software package consists of three components, each written in a different language to suit their individual purpose: A GUI, a simulator, and a visualisation module. A

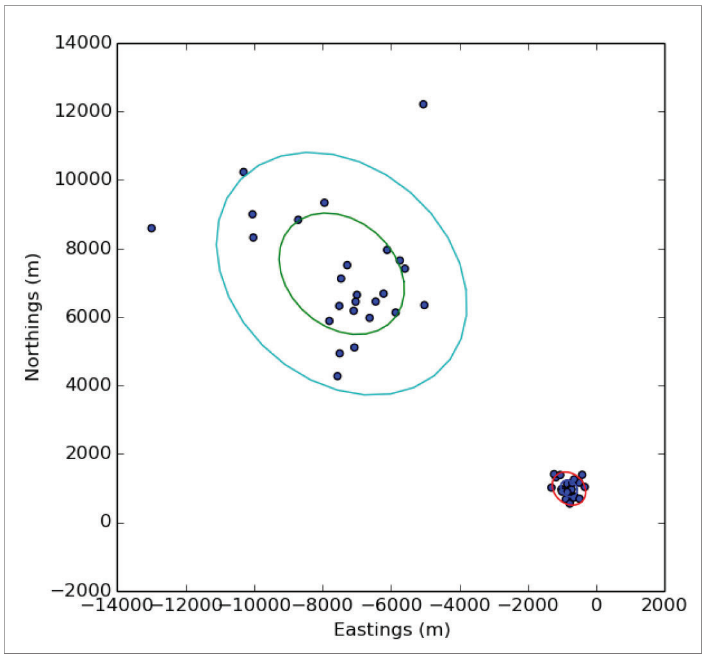


Figure 4: Landing position confidence bounds from Figure 3 along with the individual landing locations of 25 Monte Carlo iterations.

schematic representation of the three components, and how they interact, is presented in **Figure 1**.

The GUI used to design the rocket is coded in Java, where the start-up file SwingStartup.java is located at gui/swing/src/net/sf/openrocket/startup/. This starts the design page as seen in **Figure 2**, where the user can select the different components to build the rocket. More details about the GUI, and the corresponding internal structure can be found in Niskanen [14]. The example here shows a two-stage rocket. Once the rocket is designed, all the parameters are passed to the simulator core (written in C++) via an Extensible Markup Language (XML) datasheet. These parameters are required in the simulation, and consist of the moments of inertia, centre of pressure, drag coefficient, and the thrust curve. A full list of parameters and a description of the components are available in the user guide user_guide.pdf, which is located at doc/.

The core of the physical simulation, the rocketc source code, is located in cpp/ which includes a Makefile that compiles the binary and moves it to the simulator folder, found in simulator/. The Java GUI calls the simulation binary, after which the simulation starts by reading the input XML generated by the Java code. The XML file SimulationInput.xml can be found in the Data folder at Data/. Upon completion of the simulation an output XML file SimulationOutput.xml is generated in the same folder.

This output of the simulation is then input to the Python visualization code which is located in Plotter/. This presents the user with an overview of the flight trajectories and the splash-down area. The splash down-area includes confidence bounds, as seen in **Figures 3** and **4**.

When running the software under Linux, the three folders, simulator/, Data/, and Plotter/, are expected to be located in the users home directory ~/.camrocsim/. Therefore, to prepare the system for execution from source code, a script called prepare_linux.sh is included in the repository. This copies the three relevant folders to this location. The information presented in this section, and build instructions for Windows, are available in the README.md file found in the repository.

Quality control

The performance of the simulator has been evaluated by comparison to telemetry data recorded in rocket flights as described by Box, Bishop, and Hunt [5].

The performance and stability of the GUI and the visualiser have been tested in two rounds of user-testing, including cross-platform testing, which was done on Ubuntu 14.04 *Trusty Tahr* and Windows 10.0 (Build 10240).

Furthermore, there are unit tests available for the GUI and simulator, using JUnit for the GUI (Java), and GoogleTest for the simulator (C++). Instructions on how to run these are included in the README.md file.

(2) Availability

Operating system

The Cambridge Rocketry Simulator is able to function on any operating system that supports a standard Java and Python installation, which includes Linux and Windows. In all systems the simulator can be set-up by checking out the code repository. Detailed instructions on how to start the program and continue development are included in the documentation.

Programming language

Java ≥ 1.7
Python ≥ 2.7
C++ ≥ C++ 98

Additional system requirements

No special requirements.

Dependencies

The following C++ library is a required dependency:

Boost ≥ 1.58

The following Python libraries are a required dependency:

numpy ≥ 1.5
matplotlib ≥ 1.1
scipy ≥ 0.16.0

List of contributors

1. Willem Eerland (developer).
2. Simon Box (developer).
3. The project includes sources from OpenRocket [13] (GNU GPL).

Software location

Archive

Name: Zenodo
Persistentidentifier:https://doi.org/10.5281/zenodo.161850
Licence: GPL v3
Publisher: Zenodo
Version published: 3.1
Date published: 19/10/2016

Code repository

Name: SourceForge
Persistent identifier: https://sourceforge.net/p/camrocsim/code/ci/b888389c281788805b2680190dcda3b706dd8327/
Licence: GPL v3
Version published: 3.1
Date published: 19/10/2016

Language

English

(3) Reuse potential

The modular design of the Cambridge Rocketry Simulator's source code makes it inherently reusable. The GUI may be reused for applications where rocket design is required but simulation and visualization are not. The simulator itself – as described above – is a stand-alone program callable from the command-line, accepting a XML simulation input file as its single parameter. Therefore this can be ported into other applications.

Furthermore, individual classes of the simulator have reuse potential. For example the RKF45.CPP class is a general ordinary differential equation solver which implements the fourth/fifth order Runge-Kutta algorithm with Fehlberg step-size control [15]. This may be used in any dynamics simulation application.

Similarly, the VMATHS.CPP class may be used for modelling the six-degrees-of-freedom dynamics of rigid bodies. In particular the implementation of quaternions in this class is helpful for modelling unconstrained rotation without the need to deal with the singularities that arise when using Euler angles [2].

To assist users and developers, there is a user guide (USER_GUIDE.PDF) available in the DOC/ directory. This includes information on the required structure of the XML files, and instructions on how to launch the simulator and plotter via the command line.

The website Cambridge Rocketry [4] is devoted to producing open-source software for simulating high power rockets, where besides the Cambridge Rocketry Simulator, an Octave/Matlab version of the simulator core named the Cambridge Rocketry Toolbox is available. The preferred method for contributors to contact the developers is via the SourceForge webpage https://sourceforge.net/p/camrocsim/, where questions can be asked on the forum, and the developers may be contacted directly.

Acknowledgements
The authors thank the two anonymous reviewers whose comments and suggestions helped improve this manuscript and software.

Competing Interests
The authors have no competing interests to declare.

References

1. **Apogee** 2008 *RockSim v9*. URL: http://www.apogeerockets.com (visited on 31/01/2017).
2. **Baraff, D** 1997 "An introduction to physically based modeling: Rigid body simulation I – Unconstrained rigid body dynamics". In: *SIGGRAPH '97 Tutorial notes*.
3. **BBC** 2011 *Rocket launch for Bolton students on Mull of Galloway*. URL: http://www.bbc.co.uk/news/uk-england-manchester-14760906 (visited on 31/01/2017).
4. **Box, S** 2011 *Cambridge Rocketry*. URL: http://cambridgerocket.sourceforge.net (visited on 31/01/2017).
5. **Box, S, Bishop, C** and **Hunt, H** 2010 "Stochastic six-degree-of-freedom flight simulator for passively controlled high-power rockets". In: *Journal of Aerospace Engineering* 24.1, pp. 31–45. DOI: https://doi.org/10.1061/(ASCE)AS.1943-5525.0000051. eprint: http://eprints.soton.ac.uk/73938/.
6. **Box, S** and **Eerland, W J** 2016 *Cambridge Rocketry Simulator*. URL: https://sourceforge.net/p/camrocsim/ (visited on 31/01/2017).
7. **Buchanan, G** et al 2015 "The Development of Rocketry Capability in New–Zealand World Record Rocket and First of Its Kind Rocketry Course". In: *Aerospace* 2.1, p. 91. ISSN: 2226–4310. DOI: https://doi.org/10.3390/aerospace2010091
8. **Chudinov, P S** 2003 "Analytical investigation of point mass motion in midair". In: *European Journal of Physics* 25.1, p. 73. DOI: https://doi.org/10.1088/0143-0807/25/1/010
9. **Commercial Space Transportation** 2007 *Supplemental Application Guidance for Unguided Suborbital Launch Vehicles (USLVs)*. Tech. rep. Federal Aviation Administration (FAA), URL: https://www.faa.gov/.
10. **Courtney, M** and **Courtney, A** 2009 "Measuring thrust and predicting trajectory in model rocketry". In: *arXiv preprint*, arXiv:0903.1555. URL: https://arxiv.org/abs/0903.1555.
11. **Eerland, W J, Box, S** and **Sóbester, A** 2016 "Modeling the dispersion of aircraft trajectories using Gaussian processes". In: *Journal of Guidance, Control, and Dynamics*. DOI: https://doi.org/10.2514/1.G000537. eprint: http://eprints.soton.ac.uk/399818/.
12. **Gomez, F J** and **Miikkulainen, R** 2003 "Active Guidance for a Finless Rocket Using Neuroevolution". In: *Genetic and Evolutionary Computation – GECCO 2003: Genetic and Evolutionary Computation Conference Chicago, IL, USA, July 12–16, Proceedings, Part II*. Ed. by Cantú-Paz , E. et al. Berlin, Heidelberg: Springer, July, pp. 2084–2095. ISBN: 978-3-540-45110-5. DOI: https://doi.org/10.1007/3-540-45110-2_105
13. **Niskanen, S** 2015 *OpenRocket*. URL: http://openrocket.sourceforge.net (visited on 31/01/2017).
14. **Niskanen, S** 2013 *OpenRocket technical documentation*. Tech. rep. URL: http://openrocket.sourceforge.net (visited on 31/01/2017).
15. **Press, W, Teukolsky, S, Vetterling, W** and **Flannery, B** 2007 *Numerical Recipes*. Cambridge University Press.
16. **Rogers, J** 2014 "Stochastic Model Predictive Control for Guided Projectiles Under Impact Area Constraints". In: *Journal of Dynamic Systems, Measurement, and Control* 137.3 (Oct.), pp. 034503–8. DOI: https://doi.org/10.1115/1.4028084
17. **Sahu, J** 2008 "Time-accurate numerical prediction of free-flight aerodynamics of a finned projectile". In: *Journal of Spacecraft and Rockets* 45.5, pp. 946–954. DOI: https://doi.org/10.2514/1.34723
18. **Squad** 2011 *Kerbal Space Program*. URL: http://www.kerbalspaceprogram.com (visited on 31/01/2017).
19. **TU Delft** 2016 *DARE: Delft Aerospace Rocket Engineering*. URL: http://dare.tudelft.nl (visited on 31/01/2017).
20. **Yuan, T, Bar-Shalom, Y, Willett, P** and **Hardiman, D** 2014 "Impact point prediction for thrusting projectiles in the presence of wind". In: *Aerospace and Electronic Systems, IEEE Transactions on* 50.1, pp. 102–119. DOI: https://doi.org/10.1117/12.930875

How to cite this article: Eerland, W J, Box, S and Sóbester, A 2017 Cambridge Rocketry Simulator – A Stochastic Six-Degrees-of-Freedom Rocket Flight Simulator. *Journal of Open Research Software,* 5: 5, DOI: https://doi.org/10.5334/jors.137

Published: 16 June 2016 **Accepted:** 26 January 2017 **Published:** 21 February 2017