Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

USING THE TOOL COMMAND LANGUAGE FOR A FLIGHT SIMULATION FLIGHT DYNAMICS MODEL

Frank Morlang[1], and Steffen Strassburger[2]

[1]Private Person, Querumer Strasse 20A, Braunschweig, GERMANY
[2]Dept. Information Technology in Production and
Logistics, Ilmenau University of Technology, Ilmenau, GERMANY

ABSTRACT

This paper introduces a methodology for simulating flight dynamics utilizing the Tool Command Language (Tcl). Tcl, created by John Ousterhout, was conceived as an embeddable scripting language for an experimental Computer Aided Design (CAD) system. Tcl, a mature and maturing language recognized for its simplicity, versatility, and extensibility, is a compelling contender for the integration of flight dynamics functionalities. The work presents an extension method utilizing Tcl's adaptability for a novel type of flight simulation programming. Initial test findings demonstrate performance appropriate for the creation of human-in-the-loop real-time flight simulations. The possibility for efficient and precise modeling of future complicated distributed simulation elements is discussed, and recommendations regarding subsequent development priorities are drawn.

1 INTRODUCTION

Tcl was created by Dr. John Ousterhout in the late 1980s at the University of California at Berkeley (Flynt 2012). He and his team were involved in work related to enhancing simulations with macro languages. After the realization of several project specific solutions that were incompatible for other projects, they decided to develop an interpreter language that could be easily extended with new functionalities and usable with any project. This reveals that Tcl has its roots in the simulation domain. With Tcl, existing project libraries can be turned into new commands within Tcl itself. Such a set then represents a language for the domain specific application to be created. With its ease of integration with C and C++ and its compatibility with further programming languages, Tcl is the ideal choice for a fundament a flight simulation scripting language can be built on.

The non-linear equations of flight mechanics serve as the foundation for modeling flight dynamics (Stengel 2022). Consideration is given to the external forces and moments for gravity, propulsion, and aerodynamics. The coefficients of aerodynamic force and moment are parametrized as nonlinear functions of the following: dynamic pressure, aircraft slats/flaps configuration, angular velocities, control surface deflections, and flying circumstances (Mach number (M) or Angle of Attack (AoA). These are computed with the help of look-up tables that are built at regular locations throughout the whole flight domain. The use of quaternions for aircraft orientation representation is a key component that insures the prevention of large data rates and singularities comparable to the more popular Euler angle representation of orientation (Cooke et al. 1992).

Look-up tables replace intensive runtime computations with simpler data retrieval and row/column indexing operations. Both, the processing burden and the execution times are improved as a result of this. To summarize the benefits of employing table look-up, the following are some of the advantages:

- The process of retrieving a table value is far more efficient than carrying out complex calculations. When real-time simulations are taken into consideration, this acceleration can be of great significance.
- Complex computations are abstracted away into a direct data retrieval procedure, which results in a reduction in memory usage. This reduction in the necessity for calculation repetitions also benefits the CPU load. The abstraction makes the software codebase easier to understand.
- Modification and expansion need of the simulation model can easily be fulfilled by changed or added table entries.

Quaternions, which Sir William Rowan Hamilton discovered in 1843 while looking for a generalization of complex numbers, offer a useful way to update orientations (Shoemake 1985). They are particularly useful in three-dimensional computer graphics, robotics, and physics, as they provide a way to represent rotations in three-dimensional space without suffering from the gimbal lock problem that can occur with Euler angles. A quaternion is more concise in comparison to a rotation matrix (Jia 2013), provides smoother interpolation, and requires less computational resources (Perdana 2025).

The fact that no available flight dynamics model extension for Tcl currently exists founded the motivation for an own development. Further usage is designated to future programming tasks in the area of human-in-the-loop real-time flight simulations, especially with a focus on distributed simulation, where loosely coupled approaches based on Tcl are subject of current development work (Morlang and Strassburger 2024) integrating the commercial flight simulation software "X-Plane" (Laminar Research 2025a) in a Web Live, Virtual and Constructive (WebLVC) / High Level Architecture (HLA) environment. X-Plane derives its flight dynamics model from an aircraft's geometry wireframe using a process called "blade element theory" (Laminar Research 2025b). A Tcl based table look-up data driven simulation solution perfectly fits in as an addition to the X-Plane component. The objective of this paper is to assess to what extent a pure Tcl code flight dynamics extension can fulfill real-time simulation requirements and to identify the computation acceleration potential of fragments developed in C.

The other sections of this paper are structured as follows. Section 2 gives an overview about related work and exposes the addressed gap. Section 3 describes the derivation of the requirements for the rate at which the model has to recalculate the aircraft's dynamics. Section 4 presents implementation details of a first flight dynamics Tcl extension version describing exported methods as well as the realization of an additional C accelerated complement. Performance results are presented and discussed in section 5. Section 6 closes with a conclusion containing an outlook to next steps.

2 RELATED WORK AND ADDRESSED GAP

Extensions to Tcl are modules that extend Tcl's core functionalities. Their architecture is described in Welch and Thomas (2000). In general, one can distinguish between two types of them:

- Extensions that are written in Tcl itself. They are a pragmatic way to enhance Tcl by domain specific commands when high performance of computationally intensive needs is not of utmost importance.
- Extensions that are written in C or C++ that cover tasks where high-performance processing is requested.

Extensions form an essential character of the Tcl programming language, where an easy syntax with a small command set can be significantly enhanced to tailored solutions, thus representing the special adaptability and flexibility of this programming language. One important feature of extensions is their ability to be used as components for the development of other extensions. The following gives an overview about some of Tcl's relevant extensions.

Tcl is in most cases referred as Tcl/Tk. This reveals Tcl's elemental extension from the very beginning named Tk (Toolkit) (Ousterhout 1991). Tk is a graphical user interface framework extension for Tcl providing various widgets and their associated event bindings.

An important collection of Tcl extensions is called Tcllib and can be regarded as a kind of reference source for extensions covering over 400 packages in over 100 modules (Kupries 2015). The intention of Tcllib is to gather often needed functions from a broad variety of domains in one sink to be regarded and used by developers as one source to rely on with respect to stability and availability.

One of the most widely deployed database engines called SQLite has its roots as an extension to Tcl (Gaffney et al. 2022). It is the only SQL database engine specifically designed to work with Tcl (Hipp 2009). One of SQLite's features is its server lessness, a full-featured SQL implementation where a complete database is stored in a single disk file.

Developing Tcl extensions in C or C++ can benefit from a special extension itself, called CriTcl (Landers and Wippler 2002). Originally developed as a package which embedded C fragments in Tcl code with the need of C compilation at runtime, today its features allow to compile the C code once and then distribute it as an extension without the need of compilation at runtime (Kupries 2016).

A mathematical framework extension called Odielib performs 2d and 3d graphics related math operations in a C accelerated way (Woods 2017). Vectors and matrices optimized for a 4x4 affine transformation are the internal fundamental representation of the extension's functions. Among other math domains Odielib supports quaternions.

Rocket Engine (Morlang 2022) and Atmosim (Morlang 2023) are two flight simulation related extensions developed in pure Tcl code. The Rocket Engine extension arose from the need for a transition from a spacecraft thrustless flight dynamics model to a rocket-propelled one in order to perform thrustless descent/approach trajectory analyses as well as rocket-propelled ascent phase investigations. The Atmosim extension has been realized to facilitate the development of an atmosphere HLA federate application in Tcl.

The use of scripting languages in the field of six degrees of freedom (6DOF) flight dynamics covers several target programming languages. For instance, the Python Flight Mechanics Engine (AeroPython Team 2016) provides an Application Programming Interface (API) with several packages to model aircraft's flight physics. The Aerospace Toolbox (The MathWorks, Inc. 2025) delivers aerospace math related functions for the MATLAB scripting language. McFlight (Ferreira da Silva 2018) is a collection of F-16 Fighting Falcon aircraft model scripts for the high-level, numerically oriented programming language Scilab.

JSBSim (Berndt 2004) is an open-source flight dynamics model. Its usage potential covers two use cases. On the one hand, it can be used in an end-user-friendly way as a standalone application; on the other hand, it can act as a library to be fused in larger simulation frameworks, like its integration into the FlightGear (Perry 2004) open-source flight simulator as the default flight physics engine. In standalone mode, aircraft specifics must be configured in Extensible Markup Language (XML) input files. For integration purposes, it has to be used as a C++ software library with the associated complexity of dependencies on other libraries, linking and compilation.

This work addresses the gap of an available flight dynamics model extension to bring flight physics to the Tcl world on pure scripting level to be combined with other Tcl extensions for facing the challenge of larger distributed simulation needs.

3 METHOD

The approach for the realization of a flight dynamics model extension for Tcl is shown in Figure 1. At first, the equations of motions are implemented in pure Tcl. Secondly, if the performance is not sufficient, parts are rewritten in C and used as CriTcl packages (Kupries 2016) inside the extension to replace the pure Tcl counterparts until the model update rate fulfills the requirement. Update rate refers to the frequency at which a flight dynamics model updates its calculations to predict the motion of an aircraft. What is this requirement or in other words: what does ">= X Hz ?" (Figure 1) mean?

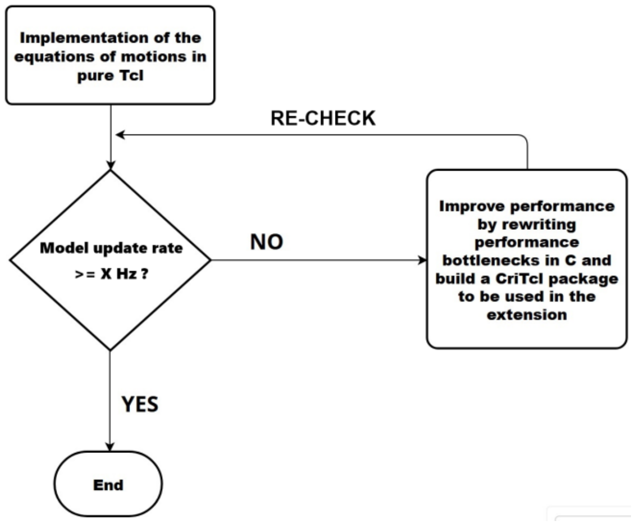
Implementation of the equations of motions in pure Tcl
RE-CHECK
Model update rate >= X Hz ?
NO
Improve performance by rewriting performance bottlenecks in C and build a CriTcl package to be used in the extension
YES
End

Figure 1: Method of the extension development.

An answer to this question can be derived from Table 1 which gives an overview of some flight simulation related work and the addressed flight dynamics model update rates. Especially Eklund and Korenberg (2000) refer to 60 Hz as "the industry standard". Against this background, we take a flight dynamics model update of 60 Hz as the core requirement in our approach (Figure 1).

Table 1: Flight dynamics model update rates in literature.

Reference	**Flight Dynamics Model Update Rate**
Allerton (2010)	50 Hz
Cooke et al. (1992)	54/57 Hz (mentioning of every 17.6/18.8 ms in the reference)
Eklund and Korenberg (2000)	60 Hz
European Aviation Safety Agency (2018)	60 Hz
Litt et al. (2022)	60 Hz
Marshall et al. (1995)	31.25/33/40/62.5 Hz
Meng et al. (2008)	60 Hz
Perry (2004)	60 Hz (mentioning of in 1/60th of a second in the reference)
Primatesta et al. (2023)	50/60 Hz
Willis (2021)	30 Hz

4 IMPLEMENTATION

The extension bases on the Tcl object system TclOO (Fellows 2010) and uses math::calculus (Markus 2004), a sub-module of Tcllib (Kupries 2015) which implements, among other algorithms, the integration of a function over an interval. Exported methods, that can be invoked from outside the object's context, start with a lower-case letter, whereas private methods start with an upper-case letter. Private methods can only be used from another method in the class. So far, we implemented the following exported methods:

- computeAirspeed
    - This method computes the speed of the aircraft's center of gravity in component form of the linear velocities defined in the aircraft body axes.
- computeAlpha
    - This method computes the aircraft's angle of attack with respect to the aircraft body axes.
- computeAlphaWingIncidence
    - This method just adds the wing incidence to alpha. This is done to consider that the wing can be offset referring to the aircraft axes. In addition, there can be a twist along the wing reducing the actual angle of attack of the wing.
- computeAlphaDot
    - This method computes the time derivative of the aircraft's angle of attack.
- computeBeta
    - This method computes the sideslip angle.
- computeBetaDot
    - o This method computes the time derivative of the sideslip angle.
- computeLift
    - This method computes the lift generated by an airfoil.
- computeDrag
    - This method computes the drag generated by an airfoil.
- computeSideForce
    - This method computes the side force. When viewed from above and the fuselage of the aircraft is not aligned with the direction of flight, there will be an incident angle with the wind. As a consequence, the fuselage generates a force in the direction along the wing.
- Computation of the body frame forces
    - These are computed from the aerodynamic and propulsive terms.
        - computeBodyFrameForceFx
        - computeBodyFrameForceFy
        - computeBodyFrameForceFz
- U,V,Z relevance (U is the velocity in the forward direction along the aircraft fuselage, V is the velocity along the starboard wing direction and Z refers to the velocity along the direction perpendicular to the underside of the aircraft fuselage.)
    - Computation of the body frame accelerations
        - These are derived from the body frame forces.
            - computeUDot
            - computeVDot
            - computeZDot
    - Computation of the body frame aerodynamic velocities
        - These are integrated from the body frame accelerations.
            - computeU_{Aero}
            - computeV_{Aero}
            - computeZ_{Aero}

All these methods are bundled in a TclOO class named fdmModel in a file named fdmTcl.tcl. To use the extension, one has to source the file with "source fdmTcl.tcl" and create an object of the class fdmModel, e.g. with "set appobj [fdmModel new]". After that, the methods can be invoked in an ensemble command form of OBJECT METHODNAME args…, e.g. "set airspeed [$appobj computeAirspeed $u $v $z]".

In addition to these methods, we implemented a table-look realization in Tcl based on math::interpolate (Markus and Kenny 2004) and Euler angles to Quaternions conversion. Converting Euler angles to quaternions and back is a common practice in flight simulation. On the one hand, the conversion to quaternions avoids gimbal lock and is associated with an improved numerical stability in large rotation or high frequency update situations; on the other hand, the conversion back to Euler angles is often needed for simplified aircraft orientation visualization of flight dynamics accompanying components (e.g. image generator). To get a first idea about the C acceleration potential, we developed an additional C accelerated (Figure 1) Euler angles to Quaternions conversion counterpart as CriTcl package inside the Tcl extension. This was realized with CriTcl's critcl::cproc functionality which defines a C function and sets up the corresponding Tcl command (Landers and Wippler 2002). With that, the CriTcl package mode was used to generate a package using the C part as a shared library, invokable as an added command inside the Tcl extension.

5 RESULTS

5.1 Performance

First processing performance results refer to an experimental setup of a Windows 11 Enterprise OS machine with Intel® Core™ i7-8650U CPU at 1.9 GHz and 32 GB RAM running Magicsplat Tcl/Tk 8.6.16 for Windows (Nadkarni 2025). The tests refer to a data-driven approach of a generic aircraft case assuming the existence and availability of proper tables and performance data (e.g. lift and drag coefficients representing airfoil specifics) for the aircraft under consideration. Figure 2 to Figure 5 show the processing performance of the implemented exported methods of the developed flight dynamics model Tcl extension. Methods neither containing numerical integration nor table look-up are consolidated in Figure 2. Their processing performance never exceeded 120 microseconds (Figure 2). The computation methods of the body frame aerodynamic velocities are presented in Figure 3. These contain numerical integration from the body frame accelerations and remained below 1000 microseconds (Figure 3). Targeting a flight dynamics model update rate of 60 Hz, we have chosen 20 integration steps, because this means 1200 integrations per second, where even a sampling frequency of 200 Hz can be considered acceptable (Bara et al. 2019). Two interpolating table look-up pure Tcl code implementations, differing in a doubled row count, are gathered in Figure 4. Outliers of their processing performance never surpassed 400 microseconds (Figure 4). To get an impression of the C acceleration potential we implemented two Euler angles to Quaternions conversions, one in pure Tcl and one with C acceleration code. We observed a boost factor of 10 (Figure 5).

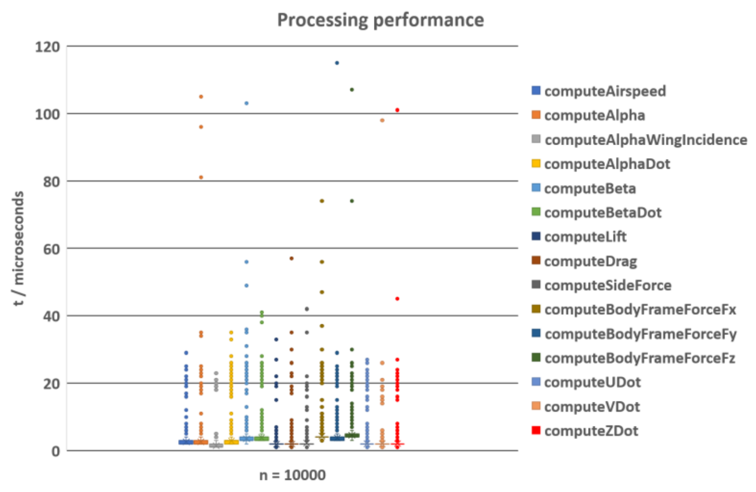

Figure 2: Processing performance of implemented methods.

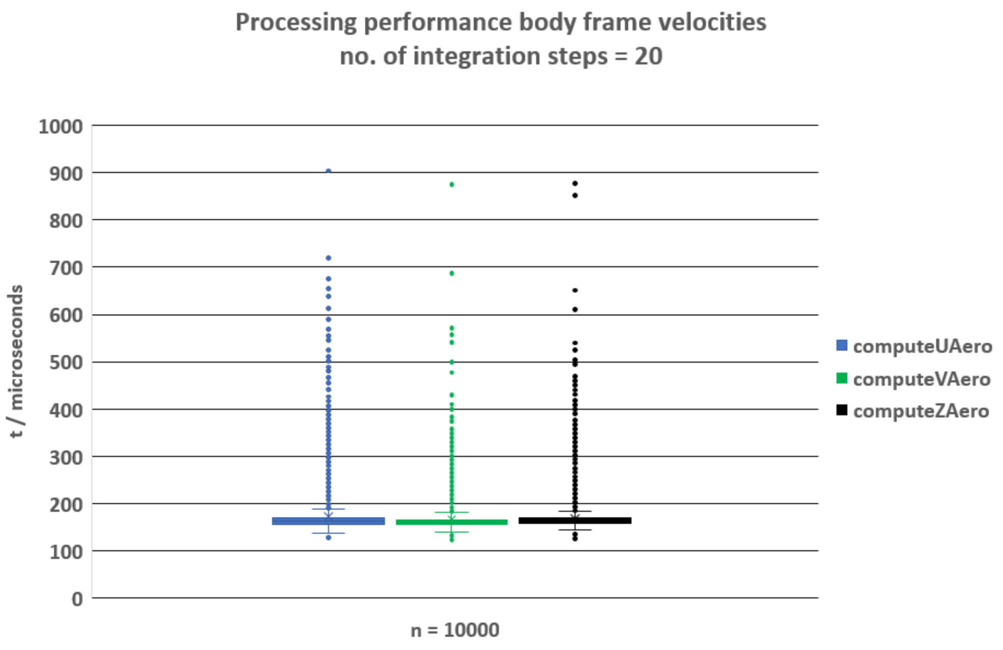

Figure 3: Processing performance body frame velocities.

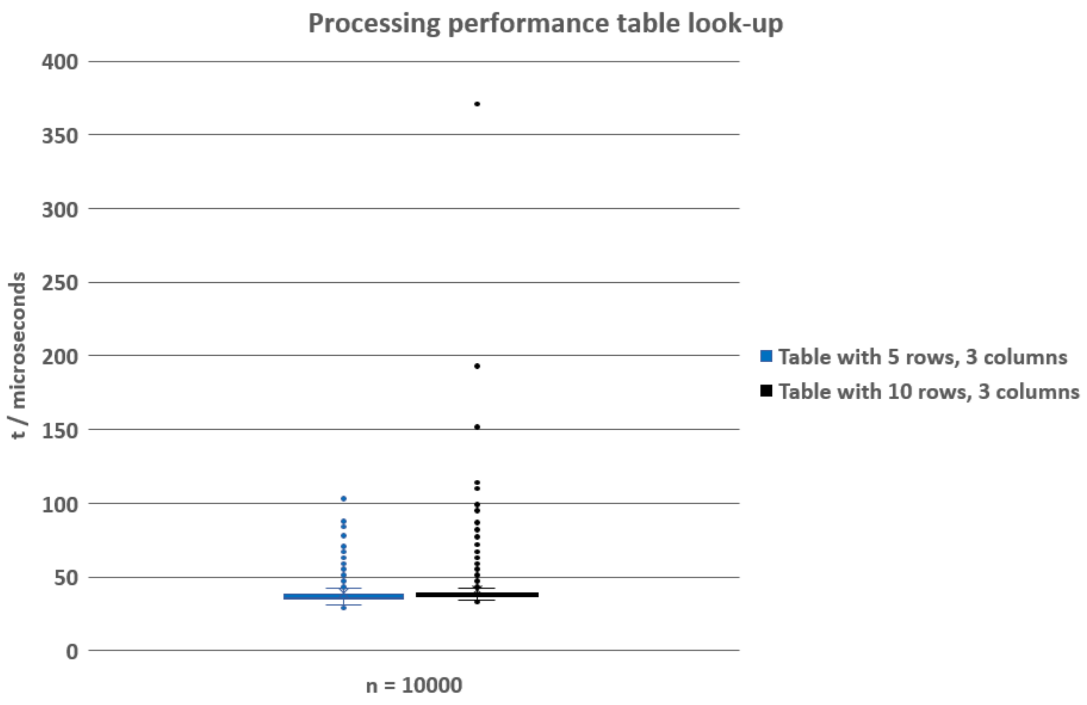

Figure 4: Processing performance table look-up.

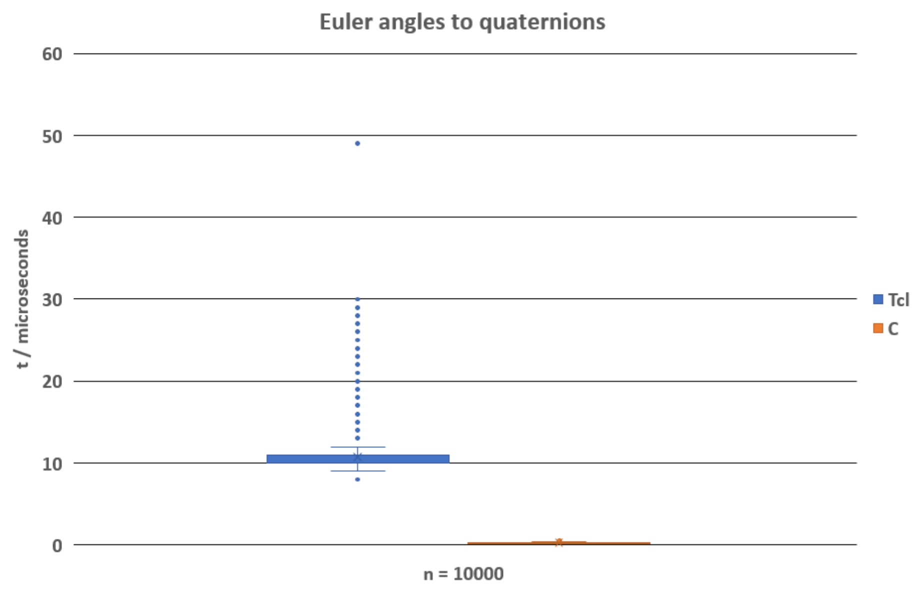

Figure 5: Processing performance Euler angles to quaternions.

The whole process of updating one step of a flight dynamics model can be summarized as computations of the following values (Allerton 2009):

- angles of attack and sideslip
- coefficients of aerodynamic forces
- aerodynamic moments in pitch, roll and yaw
- body lift, drag and side force
- engine forces and moments
- gear forces and moments
- body frame forces
- body frame accelerations
- body frame aerodynamic velocities
- wind components
- turbulence components
- earth velocities
- latitude and longitude rates
- aircraft position
- body rates in stability axes
- body frame moments in stability axes
- body frame moments in the body frame
- body frame angular accelerations
- body rates
- quaternions
- direction cosine matrix
- Euler angles

These calculations cover three computation types, table-lookup, numerical integration as well as simple ones containing only additive, trigonometric and multiplication terms. Table 2 gives estimations about the processing times for one flight dynamics model calculation step based on the maximum outlier and average values of Figure 2 to Figure 5.

Table 2: Processing time estimations based on the maximum and average values.

Computation Type	**No.**	**Processing Time (**maximum)	**Processing Time (**average)
Simple	48	(48 * 120) microseconds = 5760 microseconds	(48 * 5) microseconds = 240 microseconds
Table Look-Up	48	(48 * 400) microseconds = 19200 microseconds	(48 * 50) microseconds = 2400 microseconds
Integration	13	(13 * 1000) microseconds = 13000 microseconds	(13 * 200) microseconds = 2600 microseconds
Sum		**37960 microseconds**	**5240 microseconds**

A 60 Hz flight dynamics model rate requests an update every 16666 microseconds. The processing time estimations (Table 2) reveal that the pure Tcl code approach is not capable of fulfilling hard real-time conditions, where the 16666 microseconds deadline must always be met, but that soft real-time needs can be fulfilled. Hard real-time means that no deadline failures are allowed (Jensen 1994), whereas soft real-time permits limit misses (Stangeland 2015).

5.2 Accuracy

Against the background of a lack of real test flight data, we have chosen a simulated X-Plane flight simulator aircraft of the company X-Aerodynamics (X-Aerodynamics 2019a) as reference for accuracy testing. X-plane has a high ranking in its physical fidelity (Craighead et al. 2007). The company X-Aerodynamics is known to provide high-fidelity flight dynamics for own developments as well as for other companies offering X-Plane flight simulator aircraft (X-Aerodynamics 2019b). For an accuracy test we compared the lift during a 49 seconds level flight scenario (Figure 6) of the Tcl flight dynamics model with the lift of a simulated Partenavia P68B aircraft (X-Aerodynamics 2019c) in X-Plane. The deviation between the two lift curves (Figure 6) ranges between 0.06% and 5.20%. This translates to an accuracy between 99.94% and 94.80% and lies above the 92.00% rated as a high-fidelity dynamic model by Do et al. (2023).

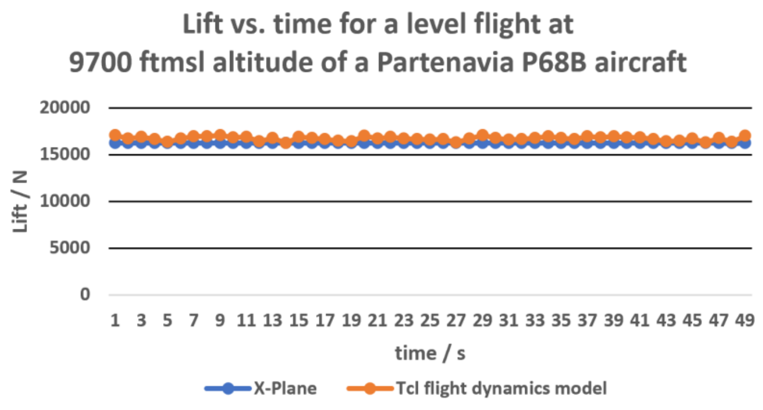

Figure 6: Lift comparison to X-Plane flight simulator reference.

6 CONCLUSION

This paper has identified the gap of a flight dynamics model extension package for the Tool Command Language (Tcl). We have shown that a pure Tcl code extension development approach can meet the needs for a 60 Hz soft real-time flight dynamics model without being forced to accelerate on C code level (Figure 1), but that a hard-real-time request cannot be fulfilled with a standard Windows 11 hard-/software setup in combination with code based on pure Tcl. Typical calculation methods can benefit from a C accelerated performance boost by a factor of ten. This shows the potential of solving the hard-real-time request / pure Tcl bottleneck by replacing all exported script level methods of the extension by C accelerated counterparts (Figure 1). A lift comparison scenario revealed a lift accuracy of high-fidelity level. Future work will focus on the implementation of these C accelerated extension methods as well as optimization for the extension's usage in the context of distributed flight simulation development based on HLA. A future release of the extension as part of the Tcllib (Kupries 2015) collection in combination with its conditions for use is planned.

REFERENCES

AeroPython Team. 2016. PyFME Python Flight Mechanics Engine. https://pyfme.readthedocs.io/en/latest/, accessed 5th June.

Allerton, D.J. 2009. *Principles of Flight Simulation*. Chichester: John Wiley & Sons.

Allerton, D.J. 2010. "The Impact of Flight Simulation in Aerospace". *The Aeronautical Journal* 114(1162):747–756.

Bara, F., P. Capone, R. Monstein, S. Godio, and G. Guglieri. 2019. "Implementation of a Comprehensive Mathematical Model for Tilt-rotor Real-time Flight Simulation". *In Proceedings of the 45th European Rotorcraft Forum*, September 17th-20th, Warsaw, Poland, 880-892.

Berndt, J.S. 2004. "JSBSim: An Open Source Flight Dynamics Model in C+". In *Proceedings of the AIAA Modeling and Simulation Technologies Conference and Exhibit*, August 16th-19th, Providence, USA, 1-27.

Cooke, J.M., M.J. Zyda, D.R. Pratt, and R.B. McGhee. 1992. "NPSNET: Flight Simulation Dynamic Modeling Using Quaternions". In Presence: Teleoperators & Virtual Environments, 1(4), 404-420. Monterey: Dudley Knox Library / Naval Postgraduate School.

Craighead, J., R. Murphy, J. Burke, and B. Goldiez. 2007. "A Survey of Commercial & Open Source Unmanned Vehicle Simulators". In *Proceedings of the 2007 IEE International Conference on Robotics and Automation*, April 10th-14th, Rome, Italy, 852-857.

Do, M. H., C.E. Lin, and Y.C. Lay. 2023. "Validation of the Flight Dynamics Engine of the X-Plane Simulator in Comparison with the Real Flight Data of the Quadrotor UAV Using CIFER". *Drones* 7(9):548.

Eklund, J. M. and M.J. Korenberg. 2000. "Simulation of Aircraft Pilot Flight Controls Using Nonlinear System Identification". *Simulation* 75(2):72-81.

European Aviation Safety Agency. 2018. "Certification Specifications for Aeroplane Flight Simulation Training Devices". CS-FSTD(A).

Fellows, D. 2010. Adventures in TclOO. https://www.tclcommunityassociation.org/wub/proceedings/Proceedings-2010/DonalFellows/Adventures-in-TclOO.pdf, accessed 26th March 2025.

Ferreira da Silva, A. 2018. McFlight. https://github.com/fsandre/mcflight, accessed 5th June.

Flynt, C. 2012. *Tcl/Tk: A Developer's Guide*. Amsterdam: Elsevier.

Gaffney, K. P., M. Prammer, L. Brasfield, D.R. Hipp, D. Kennedy, and J.M. Patel. 2022. "SQLite: Past, Present, and Future". In *Proceedings of the VLDB Endowment*, 15(12).

Hipp, D.R. 2009. SQLite - The World's Most Popular TCL Extension. http://www.tclcommunityassociation.org/wub/proceedings/Proceedings-2009/proceedings/sqlitetcl/tcl2009-sqlite.pdf, accessed 17th March 2025.

Jensen, E.D. 1994. "Eliminating the Hard/Soft Real-time Dichotomy". *Embedded Systems Programming* 7(10):28–35.

Jia, Y. B. 2013. Quaternions and Rotations. https://graphics.stanford.edu/courses/cs348a-17-winter/Papers/quaternion.pdf, accessed 19th June 2025.

Kupries, A. 2015. Plumbing the Kitchen Sink — The Tcl Standard Library. https://www.tcl-lang.org/community/tcl2015/assets/talk8/Tcllib.pdf, accessed 18th March 2025.

Kupries, A. 2016. C Runtime In Tcl. https://www.tcl-lang.org/community/tcl2016/assets/talk35/critcl-paper.pdf, accessed 17th March 2025.

Laminar Research. 2025a. XPlane12. https://www.x-plane.com/, accessed 3rd April.

Laminar Research. 2025b. How X-Plane Works. https://www.x-plane.com/desktop/how-x-plane-works/, accessed 3rd April.

Landers, S., J.C. Wippler. 2002. CriTcl - Beyond Stubs and Compilers. http://www.tcl-lang.org/community/tcl2002/archive/Tcl2002papers/wippler-critcl/critcl.pdf, accessed 17th March 2025.

Litt, J.S., T.S. Sowers, H. Buescher, and R. Jansen. 2022. "Implementation Approach for an Electrified Aircraft Concept Vehicle in a Research Flight Simulator". In AIAA SCITECH 2022 Forum, p. 2306.

Markus, A. 2004. math::calculus - Integration and Ordinary Differential Equations. https://core.tcl-lang.org/tcllib/doc/trunk/embedded/md/tcllib/files/modules/math/calculus.md, accessed 26th March 2025.

Markus, A. and K.B. Kenny. 2004. math::interpolate - Interpolation Routines. https://core.tcl-lang.org/tcllib/doc/trunk/embedded/md/tcllib/files/modules/math/interpolate.md, accessed 27th March 2025.

Marshall, S.R., V.I. Chung, and D. Martinez. 1995. Transport Delays Associated with the NASA Langley Flight Simulation Facility. https://ntrs.nasa.gov/api/citations/19950023033/downloads/19950023033.pdf, accessed 25th March 2025.

Meng, K.W., M.C. Hung, D.L. Yang, and Y.C. Chung. 2008. "Implementation of an Intelligent HLA-Compliant Application Layer Gateway for Real-Time Flight Simulation". In *Proceedings of the 8th International Conference on Intelligent Systems Design and Applications*, November 26th-28th, Kaohsiung, Taiwan, 421-426.

Morlang, F. 2022. "Rocket Engine–A Rocket Engine Propulsion Package In The Tool Command Language (Tcl)". *Webology* 19(3): 1665– 1673.

Morlang, F. 2023. "Atmosim: An Atmosphere Simulation Package in the Tool Command Language (Tcl)". *Tuijin Jishu/Journal of Propulsion Technology* 44(4):43–52.

Morlang, F. and S. Strassburger. 2024. "The Space Liner Federation – Distributed Space Vehicle Simulation Based on Loose Coupling". In *Proceedings of the 2024 IEEE Aerospace Conference*, March 2nd-9th, Big Sky, USA, 1-7.

Nadkarni, A.P. 2025. Magicsplat Tcl/Tk for Windows. https://www.magicsplat.com/tcl-installer/index.html, accessed 27th March 2025.

Ousterhout, J.K. 1991. "An X11 Toolkit Based on the Tcl Language". In *Proceedings of the Winter 1991 USENIX Conference*, January 21st-25th, Dallas, USA, 105-116.

Perdana, M. R. 2025. Quaternion-Based Representation of Aircraft Rotation in 3D Navigation Systems Using X-Plane 12 Simulation. https://www.researchgate.net/publication/388451912_Quaternion-Based_Representation_of_Aircraft_Rotation_in_3D_Navigation_Systems_Using_X-Plane_12_Simulation, accessed 19th June 2025.

Perry, A.R. 2004. The FlightGear Flight Simulator. https://www.usenix.org/legacy/events/usenix04/tech/sigs/full_papers/perry/perry.pdf, accessed 25th March 2025.

Primatesta, S., F. Barra, F. Godio, S. Guglieri, and P. Capone. 2023. "Implementation of a Comprehensive Real-time Flight Simulator for XV-15 Tilt-rotor Aircraft". In AIAA SCITECH 2023 Forum, p. 0336.

Shoemake, K. 1985. "Animating Rotation with Quaternion Curves". In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, July 22nd-26th, San Francisco, USA, 245-254.

Stangeland, N. 2015. "Alternatives to Classic Real Time - A Literature Study". Master's thesis, Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim.

Stengel, R.F. 2022. *Flight Dynamics*. Princeton: Princeton University Press.

The MathWorks, Inc. 2025. Aerospace Toolbox. https://www.mathworks.com/products/aerospace-toolbox.html, accessed 5th June.

Welch, B., M. Thomas. 2000. The Tcl Extension Architecture. https://www.usenix.org/legacy/publications/library/proceedings/tcl2k/full_papers/welchextension/welchextension.pdf, accessed 18th March 2025.

Willis, C.A. 2021. Two Published Flight Dynamics Models Rewritten in Rust and Structures as an ECS. https://scholar.afit.edu/cgi/viewcontent.cgi?article=5914&context=etd, accessed 25th March 2025

Woods, D.W. 2017. Odielib: A C Accelerated Math Library for Tcl. https://www.tcl-lang.org/community/tcl2017/assets/talk96/Paper.pdf, accessed 17th March 2025.

X-Aerodynamics. 2019a. X-Aerodynamics - About X-Aerodynamics. https://www.x-aerodynamics.com/about-x-aero, accessed 17th June 2025.

X-Aerodynamics. 2019b. X-Aerodynamics - X-Aero Projects. https://www.x-aerodynamics.com/portfolio, accessed 17th June 2025.

X-Aerodynamics. 2019c. X-Aerodynamics - X-Aero Products - Partenavia P68B Aircraft. https://www.x-aerodynamics.com/copy-of-portfolio, accessed 17th June 2025.

AUTHOR BIOGRAPHIES

FRANK MORLANG is a researcher in the area of real-time simulation. He earned his Diploma Degree in Materials Science at Technical University of Darmstadt, Germany. His research interests include air traffic management and flight simulation and the traffic integration of future suborbital winged passenger transport vehicles in normal air traffic. His private e-mail address is frank.morlang@freenet.de.

STEFFEN STRASSBURGER is a professor at the Ilmenau University of Technology and head of the Group for Information Technology in Production and Logistics. Previously he was head of the "Virtual Development" department at the Fraunhofer Institute in Magdeburg, Germany and a researcher at the Daimler Chrysler Research Center in Ulm, Germany. He holds a Doctoral and a Diploma degree in Computer Science from the University of Magdeburg, Germany. He has been involved with HLA-based distributed simulation since 1997. His further research interests include automatic simulation model generation and general interoperability topics within the digital factory and Industry 4.0 context. His email address is steffen.strassburger@tu-ilmenau.de.