# Another F-16 simulation? A user-friendly Julia, MATLAB, and Simulink flight simulation package suitable for undergraduate teaching

Duc NGUYEN[1,a *] and Mark LOWENBERG[1,b]

[1]University of Bristol, Queen's Building, University Walk, Bristol, BS8 1TR, United Kingdom

[a]duc.nguyen@birstol.ac.uk, [b]m.lowenberg@bristol.ac.uk

* corresponding author

**Keywords:** flight simulation, aircraft dynamics, education, Julia, MATLAB, Simulink

**Abstract.** We present a compact and user-friendly simulation package of the F-16 aircraft, which is based on aerodynamic data from the 1979 report by NASA. Developed across Julia, MATLAB, and Simulink, the package was designed for education and early research training. Priorities were given for clarity and ease-of-use over advanced functionality while preserving the full six-degree-of-freedom equations of motion and nonlinear aerodynamics. Three model variants are provided: 4th-, 8th-, and 12th-order, each progressively expanding the simulated dynamics. In addition to solving the nonlinear equations of motion, some common functions for flight dynamics analysis are provided, including trimming and linearisation, parameter sweep using parallel simulations, 3D trajectory display, and a unique real-time plotting capability in Julia. Benchmarking demonstrates excellent performance, particularly for Julia, which achieves sub-millisecond runtimes for typical time-integration tasks. The package enables learners to explore core flight-dynamics principles and nonlinear behaviours such as spins and deep stalls within a minimal coding structure. This work aims to lower the entry barrier to flight-simulation education and provide a flexible platform for undergraduate teaching and introductory research.

## Introduction

The nonlinear 6-degree-of-freedom (DoF) aircraft equations of motion are a challenging topic to teach. Based on the author's experience, the reason can be attributed to the large jump from simple linear representation (transfer function and state-space) to nonlinear equations with complex coupling terms and lookup tables. Rather unfortunately, many students consider linear feedback control as one of the harder units in an aerospace engineering degree. This reduces the number of students choosing to explore aircraft dynamics further – typically as an optional unit or via a final-year research project. Students who choose this topic typically go through an 'adjustment period' of transitioning from solving handwritten equations to complex and (often) closed-source flight simulations. These software are typically constructed for a research environment that prioritises functionalities and expandability, but not necessarily user friendliness. This results in a steep learning curve and requires major effort for onboarding new lab members. Similarly, experienced control engineers from other sectors may also run into the same challenges when first introduced to atmospheric flight systems.

To address the bottleneck problem above, we propose a 'lightweight' flight simulation that prioritises simplicity over functionality – one that preserves the full 6 DoF equations of motion without advanced features that can overwhelm newcomers. This means omitting additional control surfaces (flaps, spoilers, etc.), actuator and engine modelling, atmospheric modelling, and a complex coding structure with multiple nested functions. The focus is on helping learners

grasp the fundamentals behind the equations of motion for aircraft and the structure of an aircraft flight dynamics simulation—including elements such as lookup tables, time integration, and initial conditions. Other key features, such as trajectory visualisation, trimming, and linearisation, should still be included, ideally in a way that does not overwhelm the learners. This is the opposite approach to some existing open-source flight simulations, most famously the University of Minnesota F-16 simulation [1], its recent update by one of the original authors [2], and the reduced model described in the textbook by Stevens and Lewis [3]. Furthermore, we do not favour the block-diagram approach as seen in a few Simulink examples [4], as they tend to be complex visually and may impede the user's understanding of the nonlinear equations of motion (especially if 'lazy' black box blocks are used, such as those provided in Simulink's Aerospace Blockset package [4]).

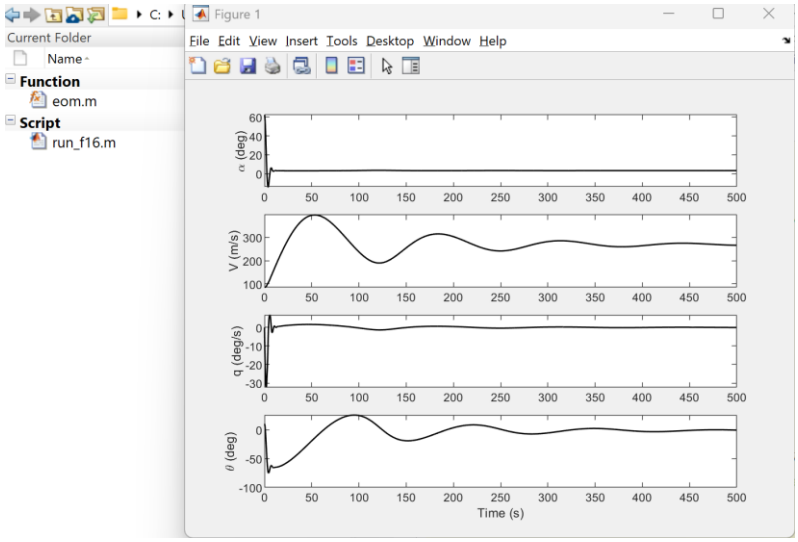



**Fig. 1. 4th-order MATLAB simulation, showing the simple simulation structure with only 2 files totalling 96 lines of code.**

## Key features

Our simulation package, which can be downloaded from GitHub [5], is based on the F-16 aerodynamic data published in the 1979 NASA report [6]. Some well-known MATLAB/Simulink packages based on this database have been published, most notably those from the University of Minnesota [1, 2]. The package presented in this paper is a collection of 4 x 3 = 12 small simulations, which are built around the 4 coding environments used (MATLAB, Simulink, MATLAB with C-code generation, and Julia) and 3 levels of complexity in the equations of motion ($4^{th}$, $8^{th}$, and $12^{th}$-order). The choice of multiple simulation environments covers a wide range of user bases. Regarding the 3 variants of the equations of motion, our rationale is:

- $4^{th}$-order: limits the aircraft motion to the longitudinal plane only. This is the ideal starting point for newcomers because the model contains all elements of a nonlinear aircraft simulation (coupled states, force and moment coefficients, and lookup table) while still being significantly simpler than the full 6 DoF equations. The 4 states modelled in the simulation are angle of attack, velocity, pitch rate, and pitch angle. The code to run the $4^{th}$-order time integration in MATLAB (including nonlinear aerodynamic tables) has only 96 lines, split between two files (see Fig. 1).

- 8th-order: expands the dynamics to include full 6 DoF. This implementation omits the 4 uncoupled equations: three representing the aircraft's location in the flat-Earth XYZ axes, plus the heading angle.

- 12th-order: contains the 4 coupled states mentioned above, along with an aircraft trajectory plotting routine.

All models assume that the air density is constant, with the default value set to 0.458 kg/m$^3$ (30,000 ft altitude). This assumption removes the coupling between air density and the aircraft Z-axis coordinate, thereby preventing an additional coupling term in the equations of motion. In almost all entry-level flight dynamics texts, only the short-term dynamics is considered, which does not involve significant changes in air density. If a different altitude is desired, users can easily change the air density constant in the code.

In addition to time integration of the flight dynamics equations of motion, a few additional features are included:

- Trimming and linearisation routines to facilitate some basic open-loop analysis.

- Parallel simulation routine to run a parameter sweep. The included example determines the trim point at different stabilator deflections by running multiple simulations for a long duration, then collecting the aircraft states at the end of each simulation.

- For the 12th-order model in Julia, the code includes a plotting routine that displays the result while the differential equation solver is still running (see Fig. 2). This is similar to the 'scope' blocks in Simulink. Visualising the solutions in real-time could be useful for detecting simulations that diverge or encounter numerical instability, which tend to result in unreasonably long or infinite run time with no easy way to alert the user. To the best of the authors' knowledge, this code is the first flight visualisation package for Julia in the public domain.

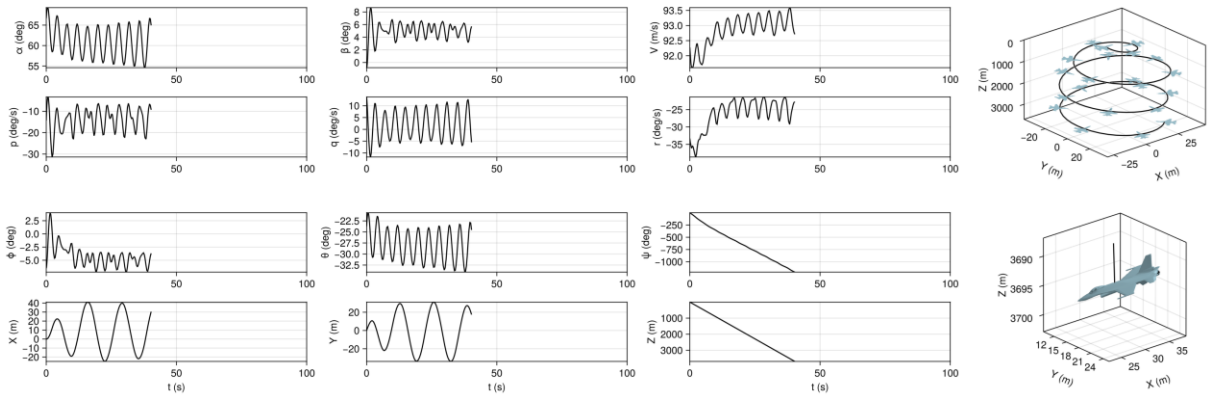



**Fig. 2. 12th-order Julia simulation with plot-while-simulating feature, showing the aircraft in a spin at high angle of attack. The lower-right trajectory plot is drawn to scale.**

Another notable feature is that the equations of motion are written in the wind axes instead of the typical body axis (i.e., using $\alpha$, $\beta$, and $V$ instead of body-axis velocities $u$, $v$, and $w$). The wind-axis representation provides an alternative reference to other simulations available. Regarding the orientation equations, we use Euler angles instead of quaternions to provide a closer link to

the typical textbook equations of motion. Experienced users can easily convert the simulations to a body-axis system and/or quaternions.

**Performance**

Coding best practices were employed to create a light code base with high speed. The run times of two 500-second simulations (one $4^{th}$- and one $8^{th}$-order) with variable steps and 1e-4 relative tolerance are shown in Table I. Results were obtained on a laptop with a 24-core Intel i9-13950HX processor and 32 GB of RAM.

**Table I. 500-second simulation run times in seconds.**

| ENVIRONMENT | $4^{TH}$-ORDER | $8^{TH}$-ORDER |
|---|---|---|
| MATLAB | 1.14 | 18.3 |
| Simulink | 0.776 | 0.702 |
| MATLAB with C-code generation | 0.0616 | 0.108 |
| Julia | 0.00209 | 0.0194 |

Table II shows the run times of a parameter sweep using parallel simulation, which involves 7826 simulations of duration 500 seconds split between 23 parallel workers. The same solver settings as above were used.

**Table II. Parallel simulation run times in sec., excluding overheads and post-processing.**

| ENVIRONMENT | $4^{TH}$-ORDER |
|---|---|
| Simulink | 81.8 |
| Julia | 18.3 |

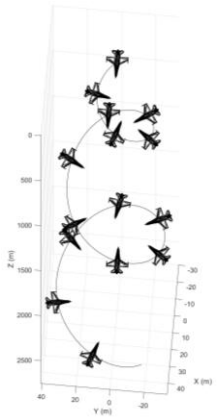
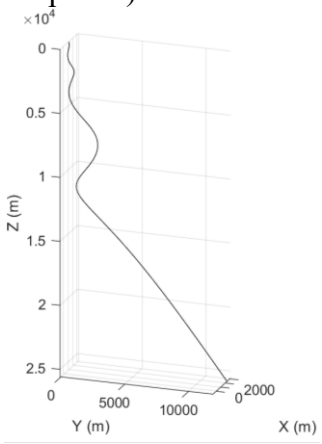
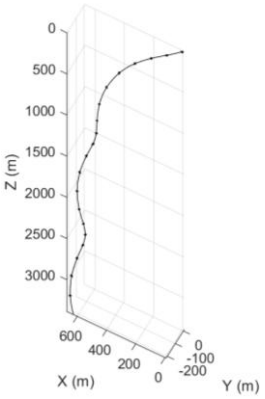
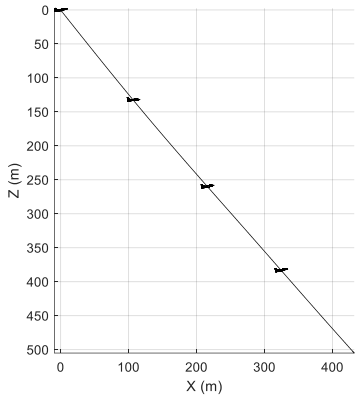
**Key results**

The $8^{th}$- and $12^{th}$-order simulations contain a set of control inputs (CI) and initial conditions (IC) that demonstrate some well-known nonlinear dynamics behaviours of the F-16. Most of these arise from the combination of nonlinear aerodynamics and aft centre-of-gravity placement. Table III summarises those cases. The airframes displayed are drawn to scale at 2-second intervals.

**Acknowledgements**

This work was supported by the UK Engineering and Physical Sciences Research Council, grant number EP/Y014545/1.

**Table III. Nonlinear dynamics demonstrations in the 8th- and 12th-order simulations.**

| | CI 1 | CI 2 |
|---|---|---|
| **IC 1** | Fully developed spin  | Spin self-recovery (back to deep stall)   |
| **IC 2** | Spin entry from deep stall   | Deep stall   |

## References


[1] Balas, G. "Non-Linear F-16 Aircraft Model", dept.aem.umn.edu/~balas/darpa_sec/SEC.Software.html [retrieved 21 July 2025].

[2] Bhattacharya, R. "F16-Model-Matlab", github.com/isrlab/F16-Model-Matlab [retrieved 31 July 2025].

[3] Stevens, B. L., Lewis, F. L., and Johnson, E. N., Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems, Wiley, 2015. doi: 10.1002/9781119174882

[4] MathWorks. "Fly the De Havilland Beaver", uk.mathworks.com/help/aeroblks/fly-the-dehavilland-beaver.html [retrieved 21 July 2025].

[5] Nguyen, D. H., and Lowenberg, M. H. "F-16_Bristol", github.com/duchn7/F-16_Bristol.git [retrieved 8 November 2025].

[6] Nguyen, L. T., Ogburn, M. E., Gilbert, W. P., Kibler, K. S., Brown, P. W., and Deal, P. L. "Simulator Study of Stall/Post-Stall Characteristics of a Fighter Airplane with Relaxed Longitudinal Static Stability," NASA Technical Paper 1538. NASA, Langley Research Center, Hampton, Virginia, 1979.