The Aeronautical Journal (2023), **127**, pp. 41–56
doi:10.1017/aer.2022.45

REGULAR PAPER

Verification framework for control theory of aircraft

O. A. Jasim and S. M. Veres*

Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, UK
*Corresponding author email: s.veres@sheffield.ac.uk

Received: 18 May 2021; **Revised:** 13 December 2021; **Accepted:** 8 April 2022

Keywords: Control theory; Higher order logic; Formal methods; Control theorems; Control systems of aircraft

Abstract

A control system verification framework is presented for unmanned aerial vehicles using theorem proving. The framework's aim is to set out a procedure for proving that the mathematically designed control system of the aircraft satisfies robustness requirements to ensure safe performance under varying environmental conditions. Extensive mathematical derivations, which have formerly been carried out manually, are checked for their correctness on a computer. To illustrate the procedures, a higher-order logic interactive theorem-prover and an automated theorem-prover are utilised to formally verify a nonlinear attitude control system of a generic multi-rotor UAV over a stability domain within the dynamical state space of the drone. Further benefits of the procedures are that some of the resulting methods can be implemented onboard the aircraft to detect when its controller breaches its flight envelope limits due to severe weather conditions or actuator/sensor malfunction. Such a detection procedure can be used to advise the remote pilot, or an onboard intelligent agent, to decide on some alterations of the planned flight path or to perform emergency landing.

Nomenclature

b	torque constant of the rotor
l	aerodynamic force constant of the rotor
$\boldsymbol{r}$	position (translational) vector of x, y, z in the world frame
$\boldsymbol{u}$	control input vector
$\boldsymbol{u}_d$	represents uncertainty and disturbances
$\boldsymbol{q}$	quaternions elements
I	symmetric and positive-definite inertia matrix
$\hat{I}$	estimated matrix of the inertia matrix I
T	time sub-domain
F_B	total force of thrusters
V	Lyapunov function
Q	symmetric positive-definite matrix
K_ω, K_q	positive-definite diagonal gain matrices

Greek Symbol

τ	torque vector
τ_d	disturbance torque vector
ϕ	roll of the aircraft
θ	pitch of the aircraft
ψ	yaw of the aircraft
ω	angular velocity vector
$\dot{\omega}$	angular acceleration vector
ξ	error vector

$\dot{\xi}$	error rate vector
η	error and error rate vector
Ω	rotor angular velocity
ℓ	length from the centre of mass of the multi-rotor to the rotor
γ	known positive constant bound
δ	positive constant
ε	positive constant
$\lambda_{min}, \lambda_{max}$	lower and upper constant bound, respectively

1.0 Introduction

Control law design for aircraft normally combines control engineering knowledge with tests of stability and smoothness of control responses under disturbances. This often takes the form of an iterative process of control law refinements accompanied with wind tunnel and in flight tests. Given a particular open loop dynamical model, control engineering relies on mathematical theory that enables the design of a flight controller. When the flight envelope is defined, it introduces numerical values for the design, which need to be carried through derivations and proofs of stability and acceptable handling within the flight envelope. In this paper manual derivations are replaced by theorem proving methods for robust control performance. The procedures presented go beyond algebraic computation and use higher-order logic, including handling of functionals, operators, stability and levels of smoothness measures. The formal approach of proofs on a computer has two advantages: avoiding mistakes of manual mathematical derivations and fast robustness calculations against parameters of the flight envelope required.

The approach taken in this paper fits into one of the three stages of formally verifiable controller design as outlined in [1], where robust control theory verification is followed by verification of the software used for implementation. A third stage is to prove that the quantisation affects on a digital controller do not significantly affect the results in stage one of the controller theory used. In implementations of aviation software, verification is often followed by redundancy-based safety analysis for critical sensors and actuators using voting principles, the use and effect of these lies outside the scope of this paper.

Realtime code-verification normally consists systematically checking the correctness of the encoded controller such as in [2, 3] and [4]. This paper focuses on verification of the control principles before it is implemented in realtime code. An interactive theorem prover (ITP) is used for performance verification under bounded nominal ambient conditions. Also monitoring of stability and performance are checked in flight using an automated theorem prover (ATP) for excessive conditions, including some sensor or actuator failures.

There have been prior initiatives on verification of safety-critical and cyber-physical systems. Such are the European Integrated Tool Chain for Model-based Design of Cyber-Physical Systems (INTO-CPS) [5], where a Functional Mockup Interface (FMI) has been developed for integrating the formal verification of Cyber-Physical Systems using the PVS (Prototype Verification System) [6] theorem prover with model-based software to co-simulate these systems. This approach integrates simulated models in model-based tools such as Modelica [7], Simulink/Matlab [8] or 20-sim [9] with the FMI interface to verify the control system according to the required specifications using formal methods.

The FMI is also implemented in [10] using Isabelle/Unifying Theory of Programming (UTP) [11], where Modelica is used to model the control system of a train, which is encoded in Isabelle/UTP within an FMI framework for co-simulation. Another project is the ERATO Metamathematics for Systems Design (MMSD) [12], where a framework is developed to use formal methods to verify automotive-related applications. Other projects are conducted by the verification team at the NASA Langley Research Center [13] such as [14, 15] and [16]. Though these approaches use formal methods to verify control systems, the derivations of control laws are not covered by verification before implementing them. In addition, there is an absence of onboard real-time stability monitoring of these systems by formal methods.

https://doi.org/10.1017/aer.2022.45 Published online by Cambridge University Press

There have been several studies conducted to formally verify control systems. In [17], a framework for reasoning about the transfer function and steady state errors in feedback control systems using the higher-order-logic (HOL) Light [18] theorem prover is proposed. Adnan et al. [19] formalised the theory of Laplace transforms using an ITP to analyse linear control systems. In [20], formal verification of coordinate transformations is proposed for aircraft control systems using the Coq [21] theorem prover. An autonomous vehicle controller is formalised and proven in [22] using the PVS theorem prover to illustrate its correctness in terms of stability of the Jacobian matrix and then co-simulated using the PVSio environment in the PVS prover. Denman et al. [23] presented a method for the implementation of a formal Nichols plot analysis of a flight control system using the MetiTarski ATP [24] with the aim of stability verification. Araiza-Illan et al. [25] proposed a framework for automatically translating the system block diagrams modelled in Simulink into the Why3 [26] prover for verification. Verifying hybrid control systems using differential dynamic logic can be carried out in the KeYmaera prover [27]. In [28, 29] abstractions of the closed-loop behaviour have been constructed for a class of controllers using KeYmaera.

The framework proposed here is different from these approaches as we are verifying the correctness of the manually derived control law of an aviation system at the design stage before the encoding and testing/simulation step, which is followed by real-time monitoring of stability by an ATP onboard the craft. To the best of our knowledge this initial verification step has not been carried out for robust control systems at this level of HOL-complexity in any prior work.

The novelty of this paper is that it demonstrates that the presented combination of interactive theorem provers are suitable for verifying the correctness of robust control theory for a prescribed flight envelope of a rotary wing craft. Although prior work suggested this may be a possibility, this is a first evidence of this kind. This involves formal stability analysis to guarantee system robustness under variable ambient conditions. Also a method of continuous stability monitoring is proposed through an onboard automated theorem prover. The latter methods are implemented in Isabelle [30] and MetiTarski [24]. The associated HOL codes have been made available online for use by the aviation community.

Our verification framework has been applied to a control scheme of a generic quad-copter presented in [31], which consists of a nonlinear attitude controller to deal with modelling uncertainty and external disturbances. The design is based on the well-known dynamic inversion technique [32, 33]. Lyapunov's method [34] has been used as part of the attitude controller design and to analyse the system to ensure its stability. We have applied our verification scheme to the attitude control system, including all definitions, assumptions, formal derivations and performance proofs. Lyapunov stability of the control system is formalised and proven in Isabelle/HOL.

We have implemented the onboard stability analysis in the MetiTarski ATP. This system is suitable to check whether the aircraft is in its permitted flight envelope and it can inform the pilot or an autonomous agent to perform some emergency action.

The paper is structured as follow: Section 2 describes theorem proving tools, where Isabelle and MetiTarski provers are introduced; Section 3 presents the proposed verification framework; Section 4 demonstrates the applicability of the framework toa multi-rotor control system and describes onboard stability monitoring; Section 5 concludes the work.

2.0 Theorem proving

Computer-based theorem proving relies on a set computational tools in some logical system that can be used to prove the soundness and correctness of mathematical arguments. There are two main types of theorem proving: Automated Theorem Proving (ATP), which proves mathematical statements automatically without human interaction, and Interactive Theorem Proving (ITP), which automates steps of formal proofs by aid of a developer guiding the process of proof. The automated steps rely on mathematical logic and automated reasoning techniques. In this paper an ITP will be used for the verification framework proposed. ITPs are proof assistants, which formally define and prove mathematical theorems.

https://doi.org/10.1017/aer.2022.45 Published online by Cambridge University Press

Therefore, their user can implement a mathematical theory by defining assumptions and some valid logical statements to start with. Then the ITP procedure is used to prove a sequence of statements, relying on available formal theories, and also by using existing logical methods and techniques or some external resources, such as ATP tools.

The distinction between ITP and ATP systems is not only that ATP systems fully automate proofs but that ATPs tend to have restricted expressivity. Unlike ITPs, they cannot handle the highest order mathematical theories. Instead, they tend to be used to prove less complex mathematical formulae containing inequalities over real numbers, quantified variables, complex algebra and special functions. On the other hand, ITP systems have the ability to aid formalising and proving mathematical theories involving higher order logic, techniques of abstract mathematics, including the concept of convergence, continuity and linear and nonlinear operators to represent dynamical systems.

To verify complex mathematical theories, ATPs can be utilised locally in an ITP to prove a step in the proof by adding their packages to the ITP software. A large number of theorem provers are listed on the "System on TPTP" [35] web site, including the most powerful ATPs that can be used to prove mathematical statements automatically. A good example of a useful ATP, from a control engineering point of view, is *MetiTarski* [24], which is a first-order logic (FOL) prover designed to work over the field of real numbers on decidable problems.

Model checking is one of the well-known tools that can be applied to engineering models and to formally verify and validate their correctness [36]. Systems are modelled as finite state transition systems, or timed automata, and their properties are expressed, for instance, in propositional temporal logic. Checking of system properties is reduced to a graph search and exhaustive exploration of all possible abstracted states. However, model checkers are not which structures to infinite dimensional state space properties, and they cannot be used to prove the correctness of robust control systems. In contrast, ITPs can be employed to verify infinite dimensional state space systems and other abstract mathematical constructs.

2.1 Isabelle/HOL

Isabelle is an interactive higher-order-logic (HOL) theorem prover, a kind of proof assistant, based on automated reasoning techniques and logical calculus. It is written using ML [37], a functional programming language, in which rules are presented as propositions and the proofs are structured by combining rules using the λ-calculus. Isabelle provides the ability to express mathematical theorems of control engineering in formulae of formal logic and prove them using the automated derivation tools provided. When a proof is completed in Isabelle, one can be certain that the mathematical theorem at hand is valid.

Isabelle/HOL is one of Isabelle's platforms with quantifiers over domains of variables and formal semantics. Isabelle/HOL has a proof-language called *Isar*, which structures the proofs in a human readable and understandable mathematical formal text for both users and computers. The mathematical formulae can be formalised and proven in Isar with the aid of Isabelle's logical tools.

Examples of such tools are the *simplifier*, which performs operation and reasoning on equations; the *classical reasoner* that carries out long chains of reasoning procedures; and *algebraic decision procedures* using advanced pattern matching by the package *sledgehammer* for automatically finding the proofs based on already-proven theorems in Isabelle's library. The latter also calls external FOL (first-order-logic) provers (ATPs) and Satisfiability Modulo Theories (SMT) solvers [38]. Isabelle has been chosen in this work for its rich logical and automation techniques since it has a large library, which contains most of the mathematical theories that are useful in control systems theory proofs.

The use of Isabelle is justified as advanced robust and adaptive control theory heavily uses concepts of convergence in limits, differentiation and integration, complex analysis, abstract vector spaces, spaces of functionals and nonlinear operators, stability theory, and formal measures of control performance. Isabelle also supports other logical systems such as Hoare logics for systems verification [39] in addition to the ability to generate executable codes of the proven statements. There is a wide range of syntax in Isabelle/HOL, the most common and useful Isabelle/HOL expressions and symbols are summarised in Table 1.

https://doi.org/10.1017/aer.2022.45 Published online by Cambridge University Press

Table 1. *Isabelle/HOL symbols and expressions*

Expression	Description
$\rightarrow$	mapping from value to value or function to function
$\longrightarrow$	refers to "*imply*" in HOL
$\Rightarrow$	used to define a function with its corresponding variable types, e.g. *real* $\Rightarrow$ *real* is a function from real to real numbers
$\Rightarrow$	refers to "*imply*" in the *Isar* language of Isabelle. For instance $x = 0 \Rightarrow y = x$ means that $x = 0$ is an assumption and $y = x$ is to be proven
$'$	refers to mapping of a function into a set of functions. For instance x' A means that x is a function which maps to another function in a set of functions A. For nonlinear operator definition in HOL like mapping signal to another signal, see prior work in [1]
$\forall$	means "*for all*" or "*for any*" over a domain of variables
$\bigwedge$	stands for "*for universal all*" which applies to all assumptions and/or proof statements. For instance, $\bigwedge x.\ \mathrm{P}x \longrightarrow \mathrm{R}x$ means that x is extended to the implied statement Rx while $\forall x.\ \mathrm{P}x \longrightarrow \mathrm{R}x$ means for all x in the assumption only
$\exists$	means "*there exist*" or "*there is*" over a domain of variables
$\exists!$	means "*there is only one*" over a domain of variables
$\wedge$	refers to the logical "*and*"
$\vee$	refers to the logical "*or*"
x'	the 1st time derivative of x (x'' the 2nd time derivative of x)
$\lvert x \rvert$	refers to absolute value of x
xi	returns the i^{th} element of the vector x
$\bullet$	an operator for the *dot product* of two vectors
$*_v$	an operator for the multiplication of a matrix and a vector
$*_s$	an operator for the multiplication of a scalar value and a vector
$**$	an operator for the multiplication of two matrices
($\lambda\ t.\ x\ t$)	this is equivalent to the function $x(t)$ with dependency on an argument (t)
norm(x)	the Euclidean norm of a vector or a matrix
SUP*(x)*	the supremum value of x over its domain set

2.2 MetiTarski

MetiTarski is a FOL automated theorem prover, which is designed as a combination of the resolution prover *Metis* [40] and a decision procedure tool *QEPCAD* [41] to conduct proofs of theorems stated algebraically over the field of real numbers. It can be utilised to prove universally quantified inequalities of transcendental and other functions as well, for instance *ln*, *log*, *exp*, etc. MetiTarski reduces problems to decidable ones by replacing some functions with their upper and lower bounds. It can be made to perform proofs automatically with the aid of three reasoners: QEPCAD, Z3 [42] and Mathematica. It can, for instance, be used to check the stability of an unmanned aerial vehicle (UAV) during flight, due to its ability to prove inequalities, which occur in the theory of dynamical stability, at a reasonable speed.

3.0 A Control theory verification framework

A framework is introduced here to perform the verification steps of the control theory that was applied at the stage of aircraft design. Following an engineer's initial control design of an aircraft, a thorough process can be started to verify the correctness of the control system using this framework. Such a computerised verification process is able to deal with stability proofs of aircraft dynamics under actuators constraints. It is able confirm or it can make flight envelope specifications more precise. In addition, its

https://doi.org/10.1017/aer.2022.45 Published online by Cambridge University Press

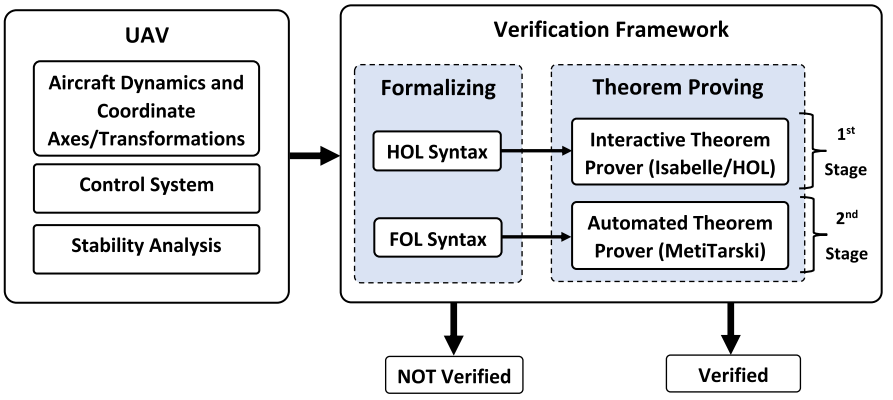

Figure 1. *UAVs verification framework.*

methods can be extended to onboard stability monitoring of aircraft to enhance the decision making process for aircraft safety. The framework is intended as a complementary work to conventional verification processes such as software and control code verification. It is not to be mixed up with control system code verification, which concerns itself with the correctness of implementation in code and also with numerical inaccuracies due to digital processing, sensor signal processing, control surface and engine actuator errors.

An example framework is outlined in Fig. 1. It consists of two stages:

(i) The ITP represented by the *Isabelle/HOL* is used to prove the mathematical derivation of the designed control system and its stability analysis.

(ii) The ATP represented by the *MetiTarski* prover is deployed to continuously check aircraft stability during the flight.

To perform the first stage, the aircraft's dynamical parameters need to be included in the Isabelle/HOL prover to carry out the verification process. The work starts with formalising the aircraft's dynamics in the HOL syntax of the Isabelle prover. The aircraft's formal dynamical theory, which need to be included in an ITP representation, contains the following:

(a) the dynamical equations of motion,

(b) the coordinate system in the rigid body frame,

(c) the transformations between the world and body frames,

(d) the controller design and its stability analysis.

Other needed concepts are also formalised in HOL such as definitions of the time domain, signals, real vectors and matrices with their properties, etc.

In a second stage, the aircraft's stability analysis is to be formalised in the FOL syntax of the MetiTarski prover for verification of stability conditions onboard the craft. The derivative of the Lyapunov function is formalised in order to check that the Lyapunov stability criterion is satisfied for the current state. If the criterion is not satisfied that can be indicative of the aircraft being outside its nominal stability domain, i.e. it violates its flight envelope.

4.0 A Case study: multi-rotor verification

This section illustrates the verification stages within framework in some detail. The stages in Fig. 1 are considered in separate subsections. The Isabelle and MetiTarski codes of these demonstrations are downloadable by aerospace engineers from an online repository [43].

https://doi.org/10.1017/aer.2022.45 Published online by Cambridge University Press

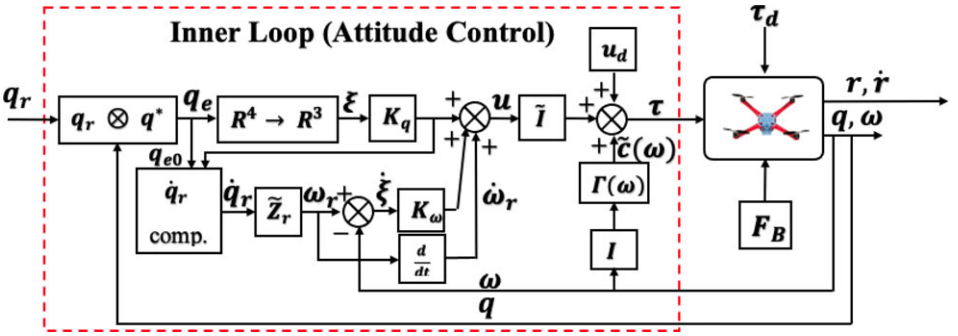

Figure 2. *The inner control loop of the multi-rotor controller [31] for attitude control. q is the measured and q_r is the desired attitude, q_e is its error in quaternions, ξ is angular error vector, ω, ω_r are the angular velocity and reference, K_q, K_ω are control gains, $I, \tilde{I}$ are the inertia matrix and its estimate. $\Gamma(\omega)$ is a cross-product matrix for the Coriolis forces such that $\Gamma(\omega)I\omega = \omega \times I\omega$ is satisfied and u_d is torque disturbance of the attitude.*

4.1 Verification using the Isabelle/HOL prover

The attitude controller of a generic multi-rotor UAV, as proposed in [31], is considered here. The aircraft's attitude dynamics are controlled by a robust nonlinear controller that takes into account the dynamical modelling uncertainties and payload disturbances. Figure 2 illustrates the attitude controller.

Simulation cannot be used to fully prove that the UAV's control system is robust for all possible flight conditions within its flight envelope. To ensure correctness of the controller robustness stated in [31], the controller design's derivations and stability analysis are to be verified using the Isabelle/HOL prover. The Isabelle/HOL software system is applied for this purpose due to its rich library of mathematical theorems, which are required to perform the UAV's control system verification.

The verification process using Isabelle/HOL is illustrated in Fig. 3, which consists of two parts: formalisation (HOL Platform) and proof procedures (Proof Tools). The first part starts by formalising the multi-rotor UAV system into the Isabelle/HOL syntax such as the coordinate system, rotational dynamics, time-domain functions, proposed assumptions and the aircraft's stability analysis. The implementation of the control design and aircraft dynamics includes a series of Isabelle items such as *definition*, *lemma* and *theorem*.

In this demo some new lemmas were needed to be stated and proven, which did not yet exist in Isabelle as that library is still under development. The formalisation also required importing some proven mathematical theories and lemmas from the prover's library, which were used in formalising the control system equations with the controller's assumptions and definitions.

The theories implemented under an *HOL* formalism will be described here to illustrate the proof procedures. The *HOL.thy* is at the core of *HOL* formalism, which includes definitions of real numbers (*real.thy*), functions (*fun.thy*), sets (*set.thy*), etc. These are necessary in most formal procedures. The *Quaternions.thy* can be used for quaternion definitions, operations and also to represent aircraft attitude. A multi-variable analysis package *Analysis.thy* can be exploited for function operations over real numbers. The *Finite_Cartesian_Product.thy* and *Inner_Product.thy* can be utilised for definitions and operations over real vectors, *L2_norm.thy* and *Norm_Arth.thy* are used for real vector norms and their operations, respectively.

A time sub-domain has been defined as $T = \{t |\ t \in [0, \infty)\}$ and relied on in definitions of sets of vector signals. For instance, the concept of a function of time (i.e. signal) over $[0, \infty)$ and its mapping to another signal is an operator. Formal nonlinear operators have been defined within Isabelle to express unknown dynamics, also used in a previous work [1], to map signals to signals by dynamical operators. These nonlinear operators are very important in descriptions and proofs of controller robustness. It is

https://doi.org/10.1017/aer.2022.45 Published online by Cambridge University Press

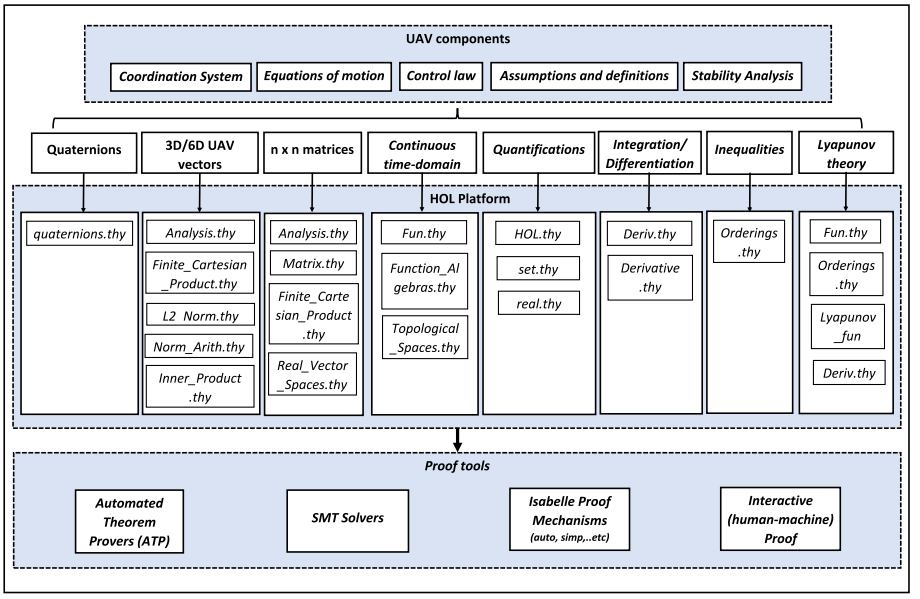

Figure 3. *Formalising and proving UAV's controller in Isabelle/HOL theorem prover.*

worth mentioning that in the Isabelle/HOL it was possible to define such operations using the "*locale*" syntax [30].

The mentioned theory *Analysis.thy* includes definitions of real vectors (*Finite_Cartesian_Product.thy*), vector norms (*L2_norm.thy, Norm_Arth.thy*) and their operations. These theories enable the definitions of the aircraft's three-dimensional rotation vectors and their norms including the torque, angular velocity and acceleration vectors, where each component of a vector is represented as a continuous time-domain function for signals. The continuous functions defined in *Fun.thy*, *Function_Algebras.thy* and *Topological_Spaces.thy* theories have been heavily relied on.

Matrices are formalised in *Matrix.thy*, and their operations and derivations performed in *Analysis.thy*, *Finite_Cartesian_Product.thy* and *Real_ Vector_Spaces.thy*. The rate change of the multi-rotor's attitudes, i.e. velocities and accelerations, are formalised by time derivatives in the theories *Deriv.thy* and *derivative.thy*.

Multi-rotor controller design requires several robustness assumptions, which need inequalities over the field of real-numbers. Fortunately, such inequalities have been defined in the Isabelle prover in the theory *Orderings.thy*. This is an important feature for any robust control design.

The second part "Theorem Proving" in Fig. 1 is an interactive process between the designer/engineer and the automated proving tools that the Isabelle prover supports. The role of designer/engineer is to exploit the ITP step-by-step the achieve a proof, where the system is not able to find a proof automatically, by simplifying the statement into several steps. Each step should be proven using the provided automated tools before moving to the next one, otherwise the prover will not pass the statement. Examples of the automated tools supported in Isabelle are: *CVC4*, *Z3*, *SPASS*, *E prover*, *Remote_Vampire* and *SMT solvers*. In addition, Isabelle has its own automatic proving tools such as *auto*, *simp*, *blast*, etc. Most of the control systems verified in this work required an interaction with the prover due to the design complexity making it not possible for the prover to solve them automatically.

The Isabelle code used in the case study is too long to be stated here, only the relevant definitions and proofs are shown, while the complete code can be found in an onlinerepository [43]. Formalisation and proofs are illustrated in the remainder of this section.

https://doi.org/10.1017/aer.2022.45 Published online by Cambridge University Press

The multi-rotor attitude dynamics in [31],

$$I\dot{\boldsymbol{\omega}} + \Gamma(\boldsymbol{\omega})I\boldsymbol{\omega} + \boldsymbol{\tau}_d = \boldsymbol{\tau}, \tag{1}$$

where $I \in \Re^{3\times3}$ is a symmetric and positive-definite inertia matrix of the craft about its mass centre. $\boldsymbol{\tau}_d(t) = [\tau_{d\phi}(t)\ \tau_{d\theta}(t)\ \tau_{d\psi}(t)]^T \in \Re^3$ are unknown disturbances torques with ϕ, θ and ψ being the roll, pitch and yaw angles, respectively. $\boldsymbol{\tau}(t) = [\tau_\phi(t)\ \tau_\theta(t)\ \tau_\psi(t)]^T \in \Re^3$ is the torque vector of the onboard controller in the body frame which produces the multi-rotor motion. These can be formalised by the following code in Isabelle/HOL:

```
definition "att_dyms ω ω' I C Γ τ τ_d = (∀t ∈ T. (∀ω. ω ∈ D3_vec_set) ∧ (∀i.((λt. ω$i) has_derivative
(λt. ω'$i))(at t within T)) ∧ I_mat I ∧ C_fun C Γ I ω ∧ τ = I *v ω' + C + τ_d)"
```

and the torque vector $\boldsymbol{\tau}$ in [31],

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} \ell l(\Omega_2^2 - \Omega_4^2) \\ \ell l(-\Omega_1^2 + \Omega_3^2) \\ b(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{bmatrix}, \tag{2}$$

is defined by bounding all rotor angular velocities Ω_i with their maximum value $\Omega_{\max}$ as

```
definition "torq_fun τ = ((∃ Ω1 Ω2 Ω3 Ω4. |Ω1| < Ω_max ∧ |Ω2| < Ω_max ∧ |Ω3| < Ω_max ∧ |Ω4| < Ω_max ∧ τ ∈ D3_vec_set ∧ τ$1 =
ℓ * l * (Ω2^2 - Ω4^2) ∧ τ$2 = ℓ * l * (-Ω1^2 + Ω3^2) ∧ τ$3 = b * (-Ω1^2 + Ω2^2 - Ω3^2 + Ω4^2)))"
```

where ℓ is the length from the centre of mass of the multi-rotor to the rotor, and l and b are the aerodynamic force and torque constants of the rotors. The control input $\boldsymbol{u}$ [31],

$$\boldsymbol{u} = \dot{\boldsymbol{\omega}}_r + K_\omega\dot{\boldsymbol{\xi}} + K_q\boldsymbol{\xi}, \tag{3}$$

and the control torque [31],

$$\boldsymbol{\tau} = \hat{I}\boldsymbol{u} + \boldsymbol{u}_d + \hat{\boldsymbol{c}}(\boldsymbol{\omega}), \tag{4}$$

are defined in the prover through the following code:

```
definition "cont_law (τ :: (real, 3)vec) I_hat u u_d C_hat = (τ = I_hat *v u + u_d + C_hat)"
definition "cont_u_def (u :: (real, 3)vec) ω'_ref K_ω K_q ξ' ξ = (u = ω'_ref + K_ω *v ξ' + K_q *v ξ)"
```

where $\hat{I}$ is an estimated matrix of the inertia matrix I of the craft, $\boldsymbol{u}$ represents a new input vector to be designed later on in Equation (3), $\boldsymbol{\omega}(t)$ is the angular velocities vector, $\dot{\boldsymbol{\omega}}(t)$ and $\dot{\boldsymbol{\omega}}_r(t)$ are the angular acceleration and reference angular acceleration vectors, respectively. $K_\omega = diag[k_{\omega_1}k_{\omega_2}k_{\omega_3}] \in \Re^{3\times3}$ is a positive-definite diagonal gain matrix setting the error gains in feedback and $K_q = diag[k_{q_1}k_{q_2}k_{q_3}] \in \Re^{3\times3}$ is a positive-definite diagonal gain matrix. $[q_{e_1}\ q_{e_2}\ q_{e_3}]^T$ is a term defined to reduce the the dimensions of quaternions. $\hat{\boldsymbol{c}}(\boldsymbol{\omega})$ is an estimate of $\boldsymbol{c}(\boldsymbol{\omega}) = \Gamma(\boldsymbol{\omega})I\boldsymbol{\omega}$ as based on $\hat{I}$ and measured $\boldsymbol{\omega}$. The additional term $\boldsymbol{u}_d$ is added to render the effects of uncertainty and disturbances in order to guarantee robustness against these effects; $\boldsymbol{u}_d$ will be defined later, to counter these, in Equation (14). The derivative of $\dot{\boldsymbol{\omega}}$ [31] is

$$\begin{aligned} \dot{\boldsymbol{\omega}} &= I^{-1}\hat{I}\boldsymbol{u} + I^{-1}\boldsymbol{u}_d + I^{-1}[\boldsymbol{\Delta}(\boldsymbol{\omega}) - \boldsymbol{\tau}_d] \\ &= \boldsymbol{u} + (I^{-1}\hat{I} - \mathbb{I})\boldsymbol{u} + I^{-1}\boldsymbol{u}_d + I^{-1}[\boldsymbol{\Delta}(\boldsymbol{\omega}) - \boldsymbol{\tau}_d] \\ &= \boldsymbol{u} + I^{-1}\boldsymbol{u}_d - \boldsymbol{y} \end{aligned} \tag{5}$$

where

$$\boldsymbol{y} = [\mathbb{I} - I^{-1}\hat{I}]\boldsymbol{u} - I^{-1}[\boldsymbol{\Delta}(\boldsymbol{\omega}) - \boldsymbol{\tau}_d]\ ,\ \ \boldsymbol{\Delta}(\boldsymbol{\omega}) = \hat{\boldsymbol{c}}(\boldsymbol{\omega}) - \boldsymbol{c}(\boldsymbol{\omega}) \tag{6}$$

https://doi.org/10.1017/aer.2022.45 Published online by Cambridge University Press

is formalised in Isabelle/HOL based on the *att_dyms*, *cont_u* and *cont_law* (see the the repository [43] for the details). The closed-loop error dynamics [31] is

$$\dot{\boldsymbol{\eta}} = A\boldsymbol{\eta} + G[\boldsymbol{y} - I^{-1}\boldsymbol{u}_d] \tag{7}$$

where $\boldsymbol{\eta} = [\boldsymbol{\xi}^T \ \ \dot{\boldsymbol{\xi}}^T]^T \in \Re^{6\times 1}$ and

$$A = \begin{bmatrix} 0^{3\times 3} & \mathbb{I}^{3\times 3} \\ -K_q^{3\times 3} & -K_\omega^{3\times 3} \end{bmatrix}, \ G = \begin{bmatrix} 0^{3\times 3} \\ \mathbb{I}^{3\times 3} \end{bmatrix}, \tag{8}$$

are implemented as

```
lemma Eq_29 :
assumes "∀t. t ∈ T"and"(set_of_definitions ω ω_ref ω' ω'_ref u u_d ξ ξ' ξ'' q q' q_r q'_r q_e τ τ_d η y ζ C C_hat
Δ A G Γ Z_t Q P K_q K_ω I I_hat)"
shows "η' = A *_v η + G *_v (y − (matrix_inv(I) *_v u_d))"
proof -
have "ξ'' = ω'_ref − ω'" using assms ddot_error_fun_def set_of_definitions_def by metis
thus ? thesis by (smt G_mat_def assms(2) exhaust_3 set_of_definitions_def)
qed
```

The *set_of_definitions* in the above code are definitions, which rely on many available, pre-defined definitions in Isabelle. The flight envelope assumptions proposed in [31] are as follows.

Assumption 1,

$$\sup(\|\dot{\boldsymbol{\omega}}_r\|) < \alpha. \tag{9}$$

Assumption 2,

$$\lambda_{\min} \leq \|I^{-1}\| \leq \lambda_{\max}. \tag{10}$$

$$\|\mathbb{I} - I^{-1}\hat{I}\| \leq \delta \leq 1, \tag{11}$$

where $\lambda_{\min} > 0$ *and* $\lambda_{\max} > 0$ *a lower and upper bound, respectively;* $\delta > 0$ *is a constant.*

Assumption 3 ,

$$\|\boldsymbol{\tau}_d\| \leq \gamma, \tag{12}$$

where $\gamma > 0$ *is a known upper constant bound. The above assumptions are formalised in Isabelle/HOL as follows:*

```
definition "assump1 ω'_ref = ((SUP t ∈ T. norm(ω'_ref)) < α)"
definition "assump2 I I_hat = (I_mat I ∧ I_hat_mat I_hat ∧ λ_min ≤ norm(matrix_inv(I)) ∧ norm(matrix_inv(I)) ≤ λ_max ∧
norm(mat 1 − ((matrix_inv(I)) ** I_hat)) ≤ δ)"
definition "assump3 (τ_d :: (real, 3)vec) = (norm(τ_d) ≤ γ)"
```

Stability analysis of the attitude controller as stated in Equations (38)–(47) [31] is implemented in Isabelle/HOL using a set of definitions in (*definition*), several lemmas in (*lemma*) and short theorems in terms of (*theorem*). This structure of using several *lemmas* and *theorems* during the proof is due to the fact that the reasoning system of the theorem prover cannot handle long proofs with many assumptions, i.e. the Isabelle system is unable to prove statements, which have many equations, if they are formalised in only one *lemma* or *theorem* style. However, stability analysis starts by defining the candidate Lyapunov function V [31],

$$V(\boldsymbol{\eta}) = \boldsymbol{\eta}^T Q\boldsymbol{\eta} > 0 \ , \ \forall \boldsymbol{\eta} \neq 0 \tag{13}$$

https://doi.org/10.1017/aer.2022.45 Published online by Cambridge University Press

which is formalised as a *definition* in Isabelle/HOL:

```
definition "Lyapunov V η = (∀t ∈ T. if (η :: (real, 6)vec) ≠ 0 then (∃a. V(η) = (a :: real) ∧ continuous_on D6_vec_set V ∧
V(η) > 0) else V(η) = 0)"
```

where $Q \in \Re^{6\times 6}$ is a symmetric positive-definite matrix. Taking the candidate Lyapunov function V, the time derivative of Lyapunov function is derived and the derivations in Equations (39)–(41) in [31] are proven symbolically and detailed in the online repository. An outline is shown below:

```
theorem Stb_Eq_39_41 :
assumes "∀η. η ≠ 0" and "Lyapunov V η" and "V(η) = η•(Q*v η)" and "A_mat A" and "η' = A*vη+G*v(y−(matrix_inv(I)*vu_d))"
and "(∀t ∈ T. ((λt. V(η)) has_derivative (λt. V'(η)))(at t within T))"
shows "V'(η) = −(η • (P *v η)) + 2 * (((η v* Q) v* G) • (y − matrix_inv(I) *v u_d))"
proof -
⋮
qed
```

The term $\boldsymbol{u_d}$ [31],

$$\boldsymbol{u}_d = \begin{cases} \dfrac{\zeta(\boldsymbol{\eta}, t)}{\|G^T Q\boldsymbol{\eta}\|} G^T Q\boldsymbol{\eta}, & \text{if } \|G^T Q\boldsymbol{\eta}\| \geq \mu \\ \dfrac{\zeta(\boldsymbol{\eta}, t)}{\mu} G^T Q\boldsymbol{\eta}, & \text{if } \|G^T Q\boldsymbol{\eta}\| < \mu. \end{cases} \tag{14}$$

is defined then as the derivative in Equation (43) [31]. This is performed using the Cauchy-Schwartz inequality (see "*theorem Eq_43*" in the repository). Based on the $\zeta(\boldsymbol{\eta}, t)$ definition, $\varepsilon > 0$, $\lambda_{\min} > 0$,

$$\zeta(\boldsymbol{\eta}, t) \geq \frac{\varepsilon}{\lambda_{\min}}, \tag{15}$$

and the upper bound of the norm of $\boldsymbol{y}$ derived in Equation (45) (31) (see "*theorem Eq_45*" in the repository), $\zeta(\boldsymbol{\eta}, t)$ in (15) is obtained (see "*theorem Eq_46*" in the repository). The terms $\boldsymbol{u_d}$ and $\zeta(\boldsymbol{\eta}, t)$ are implemented in the prover as "u_d*_def*" and "*zeta_def*" respectively:

```
definition "u_d_def u_d G Q ζ η = (∀t ∈ T. if(norm(transpose(G) *v (Q *v η)) ≥ μ) then (u_d = (ζ/norm(transpose(G) *v (Q *v
η))) *s (transpose(G) *v (Q *v η))) else (u_d = (ζ/μ) *s (transpose(G) *v (Q *v η))))"
definition "zeta_def ζ (y :: (real, 3)vec) = (∀ t ∈ T. ∃ε. ε > 0 ∧ norm(y) ≤ ε → ζ ≥ ε/λ_min)"
```

Note that the short arrow $\rightarrow$ in the code refers to *implies*, while the longer $\longrightarrow$ refers to *convergence* in HOL.

Finally, based on all the above definitions and assumptions, it has been verified that the proposed control system is asymptotically stable since the time derivative of Lyapunov function [31],

$$\dot{V}(\boldsymbol{\eta}) = -\boldsymbol{\eta}^T P\boldsymbol{\eta} + 2\boldsymbol{\eta}^T QG(\boldsymbol{y} - I^{-1}\frac{\zeta(\boldsymbol{\eta}, t)}{\|G^T Q\boldsymbol{\eta}\|} G^T Q\boldsymbol{\eta}) < 0, \tag{16}$$

$$\dot{V}(\boldsymbol{\eta}) = -\boldsymbol{\eta}^T P\boldsymbol{\eta} + 2\boldsymbol{\eta}^T QG(\boldsymbol{y} - I^{-1}\frac{\zeta(\boldsymbol{\eta}, t)}{\mu} G^T Q\boldsymbol{\eta}) < 0. \tag{17}$$

is strictly negative for $\forall \boldsymbol{\eta} \neq 0$ where $A^T Q + QA = -P$. It has also been proven that the tracking error converges to zero as the time converges to infinity ($\|\boldsymbol{\eta}\| \longrightarrow 0$). The code below illustrates the symbolic proof in the Isabelle theorem prover.

https://doi.org/10.1017/aer.2022.45 Published online by Cambridge University Press

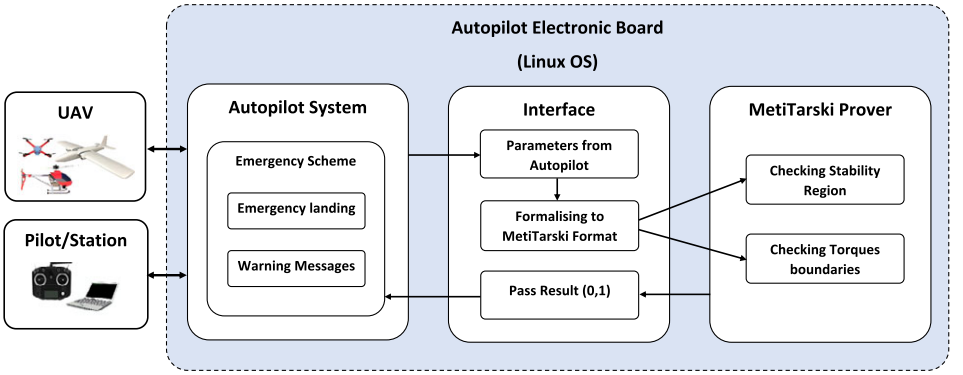

Figure 4. *Onboard verification framework of UAVs.*

theorem *Stb_Eq_47_49* :
assumes "$\forall t.\ t \in T$" **and** "(*set_of_definitions* $\omega\ \omega_{ref}\ \omega'\ \omega'_{ref}\ u\ u_d\ \xi\ \xi'\ \xi''\ q\ q'\ q_r\ q'_r\ q_e\ \tau\ \tau_d\ \eta\ y\ \zeta\ C$
$C_{hat}\ \Delta\ A\ G\ \Gamma\ Z_t\ Q\ P\ K_q\ K_\omega\ I\ I_{hat}$)" **and** "*assump1* ω'_{ref}" **and** "*assump2* $I\ I_{hat}$" **and** "*assump3* τ_d" **and** "$\forall \eta.\ \eta \neq 0$" **and**
"*Lyapunov V* η" **and** "$V(\eta) = \eta \bullet (Q *_v \eta)$" **and** "$(\forall t \in T.\ ((\lambda t.\ V(\eta))\ has\_derivative$
$(\lambda t.\ V'(\eta)))(at\ t\ within\ T))$" **and** "$\omega' = u - y + matrix\_inv(I) *_v u_d$" **and** "$\eta' = A *_v \eta + G *_v (y - (matrix\_inv(I) *_v u_d))$" **and**
"$V'(\eta) = -(\eta \bullet (P *_v \eta)) + 2 * (((\eta v^* Q) v^* G) \bullet (y - matrix\_inv(I) *_v u_d))$"
shows "$norm(transpose(G) *_v (Q *_v \eta)) \geq \mu \Longrightarrow V'(\eta) < 0$"
and "$norm(transpose(G) *_v (Q *_v \eta)) < \mu \Longrightarrow V'(\eta) < 0$" **and** "$((\lambda t.norm(\eta)) \longrightarrow 0)(at\ t\ within\ T)$"
proof -
show "$norm(transpose(G) *_v (Q *_v \eta)) \geq \mu \Longrightarrow V'(\eta) < 0$" **using** *assms Eq_19 rel_simps*(93) **by** *metis*
then show "$norm(transpose(G) *_v (Q *_v \eta)) < \mu \Longrightarrow V'(\eta) < 0$" **using** *assms Eq_19 rel_simps*(93) **by** *metis*
show "$((\lambda t.norm(\eta)) \longrightarrow 0)(at\ t\ within\ T)$" **using** *assms* **by auto**
qed

4.2 Onboard verification for a safe flight using MetiTarski

The control system of the UAV can be designed, simulated and verified at the modelling/design stage. The verified controller can then be converted into code and implemented on the autopilot system, which controls the aircraft trajectory. The aircraft can be exposed to gusts of wind, and the payload may also vary. Both may cause, in the extremes, unstable flight when the flight envelope is violated. The autopilot should provide a warning when the aircraft has entered an unstable region of its dynamical states as that may cause a crash or damage. This monitoring of the occurrence of instability can be implemented in an ATP tool, for instance the MetiTarski prover. The MetiTarski ATP has been chosen here to verify the controller stability of the aircraft due to its ability to deal with algebraic inequalities over numerical real numbers quickly. Unlike the previous verification stage using Isabelle, MetiTarski proves inequalities automatically, without the need for any interaction with a pilot or engineer.

MetiTarski can be implemented on the autopilot's electronic board such as PixHawk [44] or Navio2 [45] Raspberry Pi and many others. Interaction between the autopilot and MetiTarski is easy to create in practice. A possible system architecture is illustrated in Fig. 4. The applicability of this approach is demonstrated here in simulation by interfacing the Simulink/Matlab model with MetiTarski and testing the stability of the control system. The multi-rotor model used was based on the realistic MathWorks Simulink model from [46].

For the stability analysis stated in Equations (16) and (17), the time derivative of the Lyapunov function needs to be tested to check whether it is negative definite or not. If it is not negative definite, then

https://doi.org/10.1017/aer.2022.45 Published online by Cambridge University Press

this indicates that the control system is out of its stability region, hence the autopilot can pass a warning message to the pilot to take some suitable action or perform emergency landing.

The verification process starts by formalising the stability Equations (16) and (17) into a FOL syntax. The parameters of these equations are then passed from the autopilot system to the MetiTarski prover via an interface. The test is conducted in the MetiTarski prover to test the condition ($\dot{V}(\boldsymbol{\eta}) < 0$). The stability equations, Equations (16) and (17), were simplified using symbolic computations in Matlab before formalising them into FOL syntax. The parameters included in the stability equations were passed from Simulink/Matlab to MetiTarski to perform the monitoring. The following code illustrates stability checking in the MetiTarski prover as based on Equation (16) (the full code can be found in the online repository [43]:

```
fof(Stability_Eq47, conjecture, ![E_1, E_2, E_3, E_4, E_5, E_6, Phi, Theta] :?[Y_1, Y_2, Y_3, Zeta_E] :

%assumptions
(E_1 = 0.0037 & E_2 = 0.004964 & E_3 = 0.014124 & E_4 = 0.0504 & E_5 = 0.05748 & E_6 = 0.03166
& Phi > −1.5708 & Phi < 1.5708 & Theta > −1.5708 & Theta < 1.5708
& abs(Y_1) <= (0.04231 ∗ (180.7904 + (0.9 ∗ abs(E_4)) + (16 ∗ abs(E_1))) + (171.47 ∗ (0.332 + 0.4231)))
& abs(Y_2) <= (0.04231 ∗ (180.7904 + (0.9 ∗ abs(E_5)) + (16 ∗ abs(E_2))) + (171.47 ∗ (0.332 + 0.4231)))
& abs(Y_3) <= (0.04231 ∗ (180.7904 + (0.0064 ∗ abs(E_6)) + (25 ∗ abs(E_3))) + (171.47 ∗ (0.332 + 0.4231)))
& Zeta_E > 0 & Zeta_E >= sqrt(Y_1^2 + Y_2^2 + Y_3^2)/171.045
%implies
=> .... < 0)).
```

and for Equation (17),

```
fof(Stability_Eq49, conjecture, ![E_1, E_2, E_3, E_4, E_5, E_6, Phi, Theta] :?[Y_1, Y_2, Y_3, Zeta_E] :

%assumptions
(E_1 = 0.001227 & E_2 = 0.001241 & E_3 = 0.007062 & E_4 = 0.0168 & E_5 = 0.01437 & E_6 = 0.01583
& Phi > −1.5708 & Phi < 1.5708 & Theta > −1.5708 & Theta < 1.5708
& abs(Y_1) <= (0.04231 ∗ (180.7904 + (0.9 ∗ abs(E_4)) + (16 ∗ abs(E_1))) + (171.47 ∗ (0.332 + 0.4231)))
& abs(Y_2) <= (0.04231 ∗ (180.7904 + (0.9 ∗ abs(E_5)) + (16 ∗ abs(E_2))) + (171.47 ∗ (0.332 + 0.4231)))
& abs(Y_3) <= (0.04231 ∗ (180.7904 + (0.0064 ∗ abs(E_6)) + (25 ∗ abs(E_3))) + (171.47 ∗ (0.332 + 0.4231)))
& Zeta_E > 0 & Zeta_E >= sqrt(Y_1^2 + Y_2^2 + Y_3^2)/171.045
%implies
=> .... < 0)).
```

The system entering an unstable region has been demonstrated in Simulink- Matlab by applying an external force to the nonlinear multi-rotor's dynamics to simulate a strong gust of wind. The control system tried to overcome this dynamical change and it failed to keep the system stable because the external force was high and continuous. Without stability monitoring, this could have lead to the craft crashing.

The approach has been implemented not only to check Lyapunov stability but also to check if the limits of the system's robustness parameters have been exceeded. This has been achieved by checking whether the maximum value of one or all of the rotor angular velocities $0 < \Omega_i < \Omega_{\max}$ have been exceeded. This can be known from checking the torques for $\|\tau\| < \tau_{\max}$, where $\tau_{\max}$ can be obtained from the known maximum angular velocity of each propeller $\Omega_{\max}$ and Equation (2). In the simulation this meant that the craft was unable to compensate against the external wind and caused the craft unable to follow its prescribed flight path. For more details, see our previous work [47]. In such a situation the pilot, or autonomous agent, may decide not to insist on the planned flight path and make the most of the developing situation to save the craft by replanning.

https://doi.org/10.1017/aer.2022.45 Published online by Cambridge University Press

All that has been demonstrated in simulation can also be implemented on a real UAV. This can be achieved as follows. First, implement MetiTarski on the autopilot's electronic board with the stability and verification framework. Second, implement the interface between the autopilot and MetiTarski prover. Finally, implement the emergency decision framework inside the autopilot code as in Fig. 4.

The steps of a single cycle of stability testing, in a Simulink free onboard environment, can be implemented as follows:

- An interface is created for the autopilot which enables communication with the MetiTarski prover.
- The required parameters are communicated by the autopilot to the MetiTarski.
- The interface formalises the stability condition in FOL for MetiTarski.
- MetiTarski tests stability and communicates the result to the autopilot through the interface.
- The autopilot informs the human pilot or the onboard decision making software.

Using these methods the autopilot can send warning messages to the pilot, or in case of an autonomous mission, to an onboard intelligent software agent, to decide on a suitable counter manoeuvre [48].

5.0 Conclusions

A novel verification framework has been introduced for safety-critical control systems by applying the power of higher-order-logic-based interactive theorem provers and first-order logic-based automated theorem provers to verify the control system of unmanned aerial vehicles and to ensure aircraft control system stability and performance. The framework relies on two stages, the first is for verifying the design of the control system and its stability and the second is onboard monitoring of the aircraft's stability to ensure flight safety.

The framework has been demonstrated on a robust attitude controller of a generic multi-rotor UAV to verify the correctness of the design and stability analysis in addition to onboard monitoring of the conditions of its dynamical stability while the aircraft is flying. The aircraft's attitudes have been controlled by a nonlinear robust controller, which was designed to take into account dynamical uncertainty and external disturbances.

The methods needed in the verification stages go significantly beyond symbolic computation of inequalities for the Lyapunov theory, which can be based on first order logic (FOL). To prove robust control theory, higher order logic (HOL) concepts are needed such as quantification over sets of functions and operators as used in Isabelle HOL. These techniques were not found in prior literature on aviation control systems.

The paper illustrates the applicability and the power of interactive theorem provers relying on HOL. The methodological approach is supported by working code in an online repository [43]. This has been an initial effort that may encourage the use of such methods for control system verification in aviation and for safety-critical systems in general in the future.

References

[1] Jasim, O.A. and Veres, S.M. Towards formal proofs of feedback control theory, 2017 21st International Conference on System Theory, Control and Computing (ICSTCC), 2017, pp 43–48. doi: 10.1109/ICSTCC.2017.8107009.

[2] Feron, E. From control systems to control software, *IEEE Control Syst. Mag.*, 2010, **30**, (6), pp 50–71. doi: 10.1109/MCS.2010.938196.

[3] Jobredeaux, R.J. Formal Verification of Control Software, PhD Thesis, Georgia Institute of Technology, 2015.

[4] Wang, T. Credible Autocoding of Control Software, PhD Thesis, Georgia Institute of Technology, 2015.

[5] INTO-CPS, [Accessed: 16 May 2021] (May 2021). URL http://projects.au.dk/into-cps/

https://doi.org/10.1017/aer.2022.45 Published online by Cambridge University Press

[6] Owre, S., Rushby, J.M. and Shankar, N. PVS: A prototype verification system, in: Kapur, D. (Ed.), *Automated Deduction—CADE-11*, Springer, 1992, Berlin, Heidelberg, pp 748–752.
[7] Tiller, M. *Introduction to Physical Modeling with Modelica*, vol. **615**, Springer Science & Business Media, 2012, New York.
[8] M. Inc., Simulink/Matlab, [Accessed: 16 May 2021] (May 2021). URL https://uk.mathworks.com/
[9] 20-sim, [Accessed: 16 May 2021] (May 2021). URL https://www.20sim.com/
[10] Zeyda, F., Ouy, J., Foster, S. and Cavalcanti, A. Formalising cosimulation models, International Conference on Software Engineering and Formal Methods, Springer, 2017, Berlin, Heidelberg, pp 453–468.
[11] Isabelle/UTP, [Accessed: 16 May 2021] (May 2021). URL https://www-users.cs.york.ac.uk/simonf/utp-isabelle/
[12] ERATO Team, ERATO MMSD, [Accessed: 16 May 2021] (May 2021). URL http://www.jst.go.jp/erato/hasuo/en/
[13] NASA Langley Formal Methods Research Program, [Accessed: 16 May 2021] (May 2021). URL https://shemesh.larc.nasa.gov/fm/index.html
[14] Denman, W. and Muñoz, C. Automated real proving in PVS via MetiTarski, International Symposium on Formal Methods, Springer, 2014, pp 194–199.
[15] Muáoz, C.A. Formal methods in air traffic management: The case of unmanned aircraft systems (invited lecture), International Colloquium on Theoretical Aspects of Computing, Springer, 2015, pp 58–62.
[16] Munoz, C.A., Dutle, A., Narkawicz, A. and Upchurch, J. Unmanned aircraft systems in the national airspace system: A formal methods perspective, *ACM SIGLOG News*, 2016, **3**, (3), pp 67–76.
[17] Hasan, O. and Ahmad, M. Formal analysis of steady state errors in feedback control systems using hol-light, 2013 Design, Automation Test in Europe Conference Exhibition (DATE), 2013, pp 1423–1426. doi: 10.7873/DATE.2013.290.
[18] Harrison, J. HOL Light: A tutorial introduction, in: Srivas, M. and Camilleri, A. (Eds), *Formal Methods in Computer-Aided Design*, Springer, 1996, Berlin, Heidelberg, pp 265–269.
[19] Rashid, A. and Hasan, O. Formal analysis of linear control systems using theorem proving, in: Duan, Z. and Ong, L. (Eds), *Formal Methods and Software Engineering*, Springer International Publishing, 2017, Cham, pp 345–361.
[20] Ma, Z. and Chen, G. Formal derivation and verification of coordinate transformations in theorem prover Coq, 2017 International Conference on Dependable Systems and Their Applications (DSA), 2017, pp 127–136. doi: 10.1109/DSA.2017.29.
[21] Huet, G.K.G. and Paulin-Mohring, C. The coq proof assistant: A tutorial, Tech Rep 178 [Online]. Available: http://cs.swan.ac.uk/csoliver/ok-sat-library/OKplatform/ExternalSources/sources/Coq/Tutorial.pdf, National Institute of Research in Information and Automation (INRIA), 2009.
[22] Domenici, A., Fagiolini, A. and Palmieri, M. Integrated simulation and formal verification of a simple autonomous vehicle, in: Cerone, A. and Roveri, M. (Eds), *Software Engineering and Formal Methods*, Springer International Publishing, 2018, Cham, pp 300–314.
[23] Denman, W., Zaki, M.H., Tahar, S. and Rodrigues, L. Towards flight control verification using automated theorem proving, in: Bobaru, M., Havelund, K., Holzmann, G.J. and Joshi, R. (Eds), *NASA Formal Methods*, Springer, 2011, Berlin, Heidelberg, pp 89–100.
[24] Paulson, L.C. Metitarski: Past and future, International Conference on Interactive Theorem Proving, Springer, 2012, pp 1–10.
[25] Araiza-Illan, D., Eder, K. and Richards, A. Formal verification of control systems' properties with theorem proving, 2014 UKACC International Conference on Control (CONTROL), 2014, pp 244–249. doi: 10.1109/CONTROL.2014.6915147.
[26] Filliâtre, J.-C. and Paskevich, A. Why3 — where programs meet provers, in: Felleisen, M. and Gardner, P. (Eds), *Programming Languages and Systems*, Springer, 2013, Berlin, Heidelberg, pp 125–128.
[27] Platzer, A. Differential dynamic logic for hybrid systems, *J. Autom. Reason.*, 2008, **41**, (2), pp 143–189. doi: 10.1007/s10817-008-9103-8. URL https://doi.org/10.1007/s10817-008-9103-8
[28] Aréchiga, N., Loos, S.M., Platzer, A. and Krogh, B.H. Using theorem provers to guarantee closed-loop system properties, 2012 American Control Conference (*ACC*), IEEE, 2012, pp 3573–3580. doi: 10.1109/ACC.2012.6315388.
[29] Aréchiga, N. and Krogh, B. Using verified control envelopes for safe controller design, 2014 American Control Conference, IEEE, 2014, pp 2918–2923. doi: 10.1109/ACC.2014.6859307.
[30] Nipkow, T., Paulson, L.C. and Wenzel, M. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, vol. **2283**, Springer Science and Business Media, 2002, Berlin, Heidelberg. doi: 10.1007/3-540-45949-9.
[31] Jasim, O.A. and Veres, S.M. A robust controller for multi rotor UAV, *Aerospace Sci. Technol.*, 2020, **105**, 106010.
[32] Spong, M.W., Hutchinson, S. and Vidyasagar, M. *Robot Modeling and Control*, vol. **3**, Wiley, 2006, New York.
[33] Sciavicco, L. and Siciliano, B. *Modelling and Control of Robot Manipulators*, *Springer Science and Business Media*, 2012, Berlin, Heidelberg. doi: 10.1007/978-1-4471-0449-0.
[34] Slotine, J.-J.E., Li, W. *Applied Nonlinear Control*, vol. **199**, Prentice-Hall, 1991, Englewood Cliffs, New Jersey.
[35] Sutcliffe, Geoff and Suttner, Christian, System on TPTP, [Accessed: 16 May 2021]. (2001). URL http://www.tptp.org/cgi-bin/SystemOnTPTP
[36] Clarke, E.M., Grumberg, O. and Peled, D. *Model Checking*, MIT Press, 1999, London.
[37] Milner, R. A theory of type polymorphism in programming, *J. Comput. Syst. Sci.*, 1978 17, (3), pp 348–375.
[38] Barrett, C. and Tinelli, C. Satisfiability modulo theories, in: *Handbook of Model Checking*, Springer, 2018, pp 305–343.
[39] Nipkow, T. Hoare logics in Isabelle/HOL, in: *Proof and System-Reliability*, Springer, 2002, Berlin, Heidelberg, pp 341–367.
[40] Hurd, J. First-order proof tactics in higher-order logic theorem provers, Design and Application of Strategies/Tactics in Higher Order Logics, number NASA/CP-2003-212448 in NASA Tech *Rep*, 2003, pp 56–68.
[41] Brown, C.W. QEPCAD B: A program for computing with semi-algebraic sets using cads, *ACM SIGSAM Bull.*, **37**, (4), 2003, pp 97–108.

https://doi.org/10.1017/aer.2022.45 Published online by Cambridge University Press

[42] De Moura, L. and Bjørner, N. Z3: An efficient SMT solver, Tools and Algorithms for the Construction and Analysis of Systems, 2008, Berlin, Heidelberg, pp 337–340.
[43] Jasim, O.A. and Veres, S.M. Isabelle and MetiTraski Code for Controller Verification of an Attitude Controller for Multirotor UAV, [Accessed: 14 May 2021] (May 2021). URL https://github.com/ojrobotics/fvuav
[44] PixHawk, [Accessed: 16 May 2021] (2021). URL http://pixhawk.org/
[45] Emlid. Navio2, [Accessed: 16 May 2021] (May 2021). URL https://emlid.com/navio/
[46] Horton, B. Modelling simulation and control of a quadcopter, MATLAB Academic Conference. Australia and New Zealand, 2016, pp 4–14, [Accessed: 16 May 2021]. URL https://uk.mathworks.com/videos/modelling-simulation-and-control-of-a-quadcopter-122872.html
[47] Jasim, O.A. and Veres, S.M. Nonlinear attitude control design and verification for a safe flight of a small-scale unmanned helicopter, 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), 2019, pp 1652–1657. doi: 10.1109/CoDIT.2019.8820310.
[48] Lusk, P.C., Glaab, P.C., Glaab, L.J. and Beard, R.W. Safe2ditch: Emergency landing for small unmanned aircraft systems, *J. Aerospace Inf. Syst.*, 2019, pp 327–339.

Cite this article: Jasim O. A. and Veres S. M. (2023). Verification framework for control theory of aircraft. *The Aeronautical Journal*, **127**, 41–56. https://doi.org/10.1017/aer.2022.45