Formal Analysis of Linear Control Systems using Theorem Proving

Adnan Rashid and Osman Hasan
School of Electrical Engineering and Computer Science (SEECS),
National University of Sciences and Technology (NUST), Islamabad, Pakistan
{adnan.rashid,osman.hasan}@seecs.nust.edu.pk

Abstract

Control systems are an integral part of almost every engineering and physical system and thus their accurate analysis is of utmost importance. Traditionally, control systems are analyzed using paper-and-pencil proof and computer simulation methods, however, both of these methods cannot provide accurate analysis due to their inherent limitations. Model checking has been widely used to analyze control systems but the continuous nature of their environment and physical components cannot be truly captured by a state-transition system in this technique. To overcome these limitations, we propose to use higher-order-logic theorem proving for analyzing linear control systems based on a formalized theory of Laplace transform method. For this purpose, we have formalized the foundations of linear control system analysis in higher-order logic so that a linear control system can be readily modeled and analyzed. The report presents a new formalization of Laplace transform and the formal verification of its properties that are frequently used in the transfer function based analysis to judge the frequency response, gain margin and the phase margin and stability of a linear control system. We also formalize the active realizations of various controllers, like Proportional-Integral-Derivative (PID), Proportional-Integral (PI), Proportional-Derivative (PD), and various active and passive compensators, like lead, lag and lag-lead. For illustration, we present a formal analysis of an unmanned free-swimming submersible vehicle using the HOL Light theorem prover.

Keywords: Control Systems, Higher-order Logic, Theorem Proving

1 Introduction

Linear control systems are widely used to regulate the behavior of many safety-critical applications, such as process control, aerospace, robotics and transportation. The first step in the analysis of a linear control system is the construction of its equivalent mathematical model by using the physical and engineering laws. For example, in the case of electrical systems, we need to model the currents and voltages passing through the electrical components and their interactions in the corresponding electrical circuit using the system governing laws, such as Kirchhoff's current law (KCL) and Kirchhoff's voltage law (KVL). The mathematical model is then used to derive differential equations describing the relationship between the inputs and outputs of the underlying system. The next step in the analysis of a linear control system is to solve these equations to obtain a transfer function, which is in turn used to assess many interesting control system characteristics, such as frequency response, phase margin and gain margin. However, solving these equations in time domain is not so straightforward as they usually involve the integral and differential operators. Laplace transform, which is an integral based transform method, is thus used to convert these differential equations to their equivalent algebraic equations in s-domain by converting the differential and integral operations into multiplication and division operators, respectively. This algebraic equation can be quite easily solved to obtain the corresponding transfer function, frequency response, gain margin and the phase margin and perform the stability analysis of the given control system.

Traditionally, the linear control system analysis is performed using paper-and-pencil proof methods. However, these methods are human-error prone and cannot be relied upon for the analysis of safety-critical applications. Moreover, there is always a risk of misusing an existing mathematical result as this manual analysis method does not provide the assurance that a mathematical law would be used only if all of its required assumptions are valid. Computer simulation and numerical methods are also frequently used to analyze linear control systems. However, they also compromise the accuracy of the results due to the involvement of computer arithmetic and the associated round-off errors. Computer algebra systems (CAS), such as Mathematica [1], are also used for the Laplace transform based analysis of linear control systems. However, CAS are primarily based on unverified symbolic algorithms and thus there is no formal proof to ascertain the accuracy of their analysis results. Given the inaccurate nature of all the above-mentioned analysis techniques, they are not very suitable to analyze control

systems used in safety-critical domains, where even a slight error in analysis may lead to disastrous consequences, including the loss of human lives.

To overcome the above-mentioned limitations, Model checking [2] has been also used to analyze control systems [3, 4] but the continuous nature of their environment and physical components cannot be truly captured by a state-transition system in this technique. Higher-order logic is expressive enough to capture the continuous aspects of control systems and their environment. Thereafter, proof assistants, like HOL, have been used to reason about these formal models to judge the desired characteristics of control systems [5, 6, 7]. However, all these existing works focus on the verification of the transfer functions for a control system and, to the best of our knowledge, no prior work dealing with the formal analysis of dynamics of a linear control system exists in the literature of higher-order-logic theorem proving.

In this report, we present a framework to conduct the formal analysis of dynamical characteristics of a linear control system using higher-order-logic theorem proving. The main idea behind the proposed framework, depicted in Fig. 1, is to formalize all the foundational components of a linear control system to facilitate formal modeling and reasoning about linear control systems within the sound core of a theorem prover. For this purpose, we built upon the higher-order-logic formalizations of Multivariable calculus [8] and a library of analog components, like resistor, capacitor and inductor [9]. Firstly, we present a *new formalization of Laplace transform*, which includes its formal definition and the formal verification of some of its classical properties, which are frequently used to reason about the transfer function of an n-order system. We also formalized some widely used *characteristics of linear control systems*, such as frequency response, gain margin and phase margin, which can be used for the stability analysis of a linear control system. Moreover, we formalize the *active realizations of various controllers*, such as Proportional-Integral-Derivative (PID), Proportional-Integral (PI), Proportional-Derivative (PD), Proportional (P), Integral (I) and Derivative (D) and various *active and passive compensators*, such as lag, lead and lag-lead.

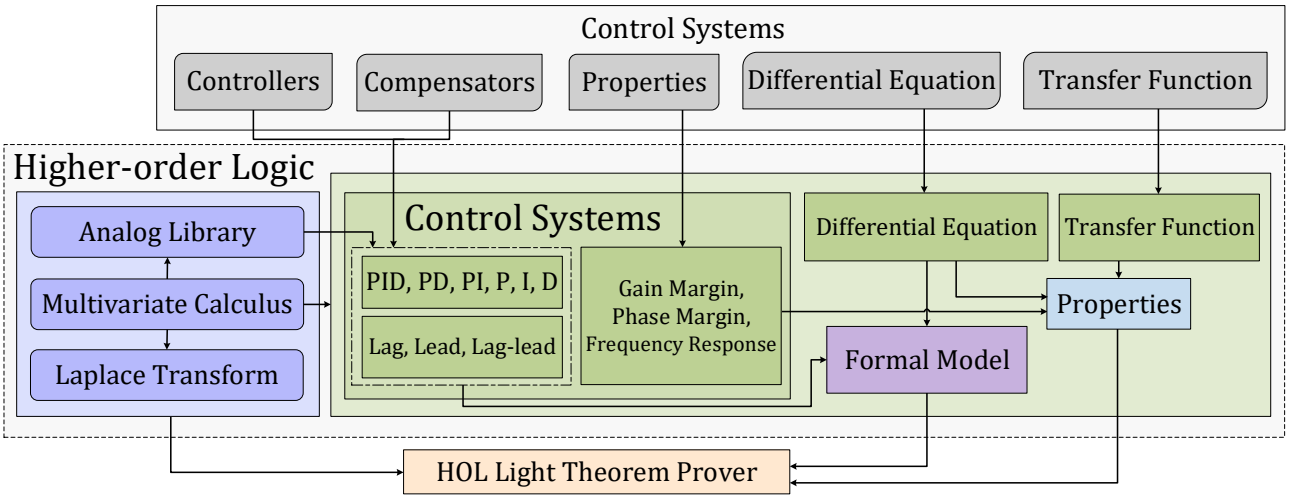


Figure 1: Proposed Framework

The proposed framework, depicted in Fig. 1, allows us to build a formal model of the given linear control system, based on the active realizations of its controllers and compensators, the passive realizations of compensators and differential equations. Moreover, it also allows to formalize the behavior of the given linear control system in terms of its differential equation, transfer function specification and its properties, such as phase margin, frequency response and gain margin. We can then use these formalized models and properties to verify an implication relationship between them, i.e., model implies its specification. In order to demonstrate the effectiveness of our proposed formalization, we formalize the linear control system of an unmanned free-swimming submersible vehicle [10]. We have used the HOL Light theorem prover [11] for the proposed formalization in order to build upon its multivariable calculus theories. We have also developed a tactic that can be used to automatically verify the transfer function of any control system up to $20^{th} order$. The report demonstrates its effectiveness by using it for the verification of the controllers, compensators and the unmanned submersible vehicle.

2 Related Work

Johnson [3] used a symbolic model checker (SMV) for verifying the safety properties of servo-loop control systems. Similarly, hybrid systems have been analysed by capturing their dynamical behaviour as a state-space model [4]. However, various abstractions were considered in these model checking based verification efforts as the dynamical behaviour of the control systems can only be modeled completely using differential equations.

Arthan et al. [12] proposed a system, ClawZ, which provides a translation of Mathworks Simulink models of a discrete-time control system into the formal Z language specification. Next, a controller implementation in Ada corresponding to this specification is verified in ProofPower. Similarly, a feedback control system is modeled and analyzed using the DOVE environment [13]. *Arechiga et al.* [14] used an automated theorem prover KeYmaera for the formal verification of the safety properties of the closed-loop control systems with sampled-time controllers. Due to the limited expressiveness of the underlying logic in these automated theorem provers, the continuous nature of the models is again abstracted in the formal modeling process. These abstractions compromise the completeness of the analysis.

Hoare logic based formal reasoning support for a single-input single-output continuous-time control system in frequency domain has also been proposed in [15]. The main idea is to use formalized block diagrams in a compositional way in the HOL98 theorem prover. This formalization has been used to analyse the phase and gain shift properties of a control system. However, the formalization is just limited for the analysis of systems that can be expressed using a block diagram with a tree structure. *Boulton et al.* [16] presented an automated symbolic method, which replaces the typical Nicole and Bode plots with their formal models. This method involves a merger of a CAS and theorem prover, i.e., Maple, and the higher-order-logic theorem prover PVS. Maple is used to find the verification conditions, which are in turn discharged using PVS. The involvement of Maple in the framework somewhat compromises the accuracy of the corresponding analysis.

Hasan et al. presented a formalization of the block diagrams in HOL Light and used it to reason about the transfer function and the steady-state error analysis of a feedback control system [5]. *Ahmed et al.* used this formalization of block diagrams to verify the steady-state error of a unity feedback control system [6]. Similarly, *Beillahi et al.* formalized of the signal flow graphs in HOL Light, which can be used to formally verify transfer functions of linear control systems [7]. However, all these existing works in HOL Light focus on the verification of the transfer functions for a control system and thus cannot cater for the dynamical analysis of the control systems.

3 Multivariable Calculus Theories in HOL Light

An N-dimensional vector is formalized in the multivariable theory of HOL Light as a $\mathbb{R}^{\mathbb{N}}$ column matrix of real numbers [8]. All of the multivariable calculus theorems are verified for functions with an arbitrary data-type $\mathbb{R}^{\mathbb{N}} \rightarrow \mathbb{R}^{\mathbb{M}}$.

A complex number is defined as a 2-dimensional vector, i.e., a $\mathbb{R}^2$ matrix.

Definition 3.1. Cx and ii

```
⊢ ∀ a. Cx a = complex (a, &0)
⊢ ii = complex (&0, &1)
```

`Cx` : $\mathbb{R} \rightarrow \mathbb{R}^2$ is a type casting function that accepts a real number and returns its corresponding complex number with the imaginary part equal to zero, where the `&` operator type casts a natural number to its corresponding real number. Similarly, `ii` (iota) represents a complex number having the real part equal to zero and the magnitude of the imaginary part equal to 1.

Definition 3.2. Re, Im, lift and drop

```
⊢ ∀ z. Re z = z$1
⊢ ∀ z. Im z = z$2
⊢ ∀ x. lift x = (lambda i. x)
⊢ ∀ x. drop x = x$1
```

The function `Re` accepts a complex number (2-dimensional vector) and returns its real part. Here, the notation `z$i` represents the i^{th} component of vector `z`. Similarly, `Im` takes a complex number and returns its imaginary part. The function `lift` accepts a variable of type $\mathbb{R}$ and maps it to a 1-dimensional vector with the input variable as its single component. Similarly, `drop` takes a 1-dimensional vector and returns its single element as a real number.

Definition 3.3. Exponential Function
⊢ ∀ x. exp x = Re (cexp (Cx x))

The complex exponential and real exponentials are represented as `cexp` : $\mathbb{R}^2 \to \mathbb{R}^2$ and `exp` : $\mathbb{R} \to \mathbb{R}$ in HOL Light, respectively [17].

Definition 3.4. Vector Integral and Real Integral
⊢ ∀ f i. integral i f = (@y. (f has_integral y) i)
⊢ ∀ f i. real_integral i f = (@y. (f has_real_integral y) i)

The function `integral` represents the vector integral and is defined using the Hilbert choice operator `@` in the functional form. It takes the integrand function `f` : $\mathbb{R}^N \to \mathbb{R}^M$, and a vector-space `i` : $\mathbb{R}^N \to \mathbb{B}$, which defines the region of convergence, and returns a vector $\mathbb{R}^M$, which is the integral of `f` on `i`. The function `has_integral` represents the same relationship in the relational form. Similarly, the function `real_integral` accepts an integrand function `f` : $\mathbb{R} \to \mathbb{R}$ and a set of real numbers `i` : $\mathbb{R} \to \mathbb{B}$ and returns the real-valued integral of function `f` over `i`.

Definition 3.5. Vector Derivative
⊢ ∀ f net. vector_derivative f net = (@f'. (f has_vector_derivative f') net)

The function `vector_derivative` takes a function `f` : $\mathbb{R}^1 \to \mathbb{R}^M$ and a `net` : $\mathbb{R}^1 \to \mathbb{B}$, which defines the point at which `f` has to be differentiated, and returns a vector of data-type $\mathbb{R}^M$, which represents the differential of `f` at `net`. The function `has_vector_derivative` defines this relationship in the relational form.

Definition 3.6. Limit of a function
⊢ ∀ f net. lim net f = (@l. (f → l) net)

The function `lim` accepts a `net` : $\mathbb{R}^1 \to \mathbb{B}$ and a function `f` : $\mathbb{A} \to \mathbb{R}^M$ and returns `l` : $\mathbb{R}^M$, i.e., the value to which `f` converges at the given `net`.

4 Formalization of Laplace Transform

Mathematically, Laplace transform is defined for a function $f : \mathbb{R}^1 \to \mathbb{C}$ as [18]:

$$\mathcal{L}[f(t)] = F(s) = \int_0^\infty f(t)e^{-st}dt,\ s\ \epsilon\ \mathbb{C} \tag{1}$$

We formalize Equation 1 in HOL Light as follows:

Definition 4.1. Laplace Transform
⊢ ∀ s f. laplace_transform f s = integral {t| &0 <= drop t} (λt. cexp (--(s * Cx (drop t))) * f t)

The function `laplace_transform` accepts a complex-valued function `f` : $\mathbb{R}^1 \to \mathbb{R}^2$ and a complex number `s` and returns the Laplace transform of `f` as represented by Equation 1. In the above definition, we used the complex exponential function `cexp` : $\mathbb{R}^2 \to \mathbb{R}^2$ because the return data-type of the function `f` is $\mathbb{R}^2$. Here, the data-type of `t` is $\mathbb{R}^1$ and to multiply it with the complex number `s`, it is first converted into a real number by using `drop` and then it is converted to data-type $\mathbb{R}^2$ using `Cx`. Next, we use the vector function `integral` (Definition 3.4) to integrate the expression $f(t)e^{-i\omega t}$ over the positive real line since the data-type of this expression is $\mathbb{R}^2$. The region of integration is {`t | &0 <= drop t`}, which represents the positive real line. Laplace transform was earlier formalized using a limiting process as [19]:

```
⊢ ∀ s f.  laplace f s = lim at_posinfinity (λb.  integral
          (interval [lift (&0), lift b]) (λt.  cexp (--(s * Cx (drop t))) * f t))
```

However, the HOL Light definition of the integral function implicitly encompasses infinite limits of integration. So, our definition covers the region of integration, i.e., $[0, \infty)$, as {`t | &0 <= drop t`} and is equivalent to the definition

of Laplace transform given in [19]. However, our definition considerably simplifies the reasoning process in the verification of Laplace transform properties since it does not involve the notion of limit.

The Laplace transform of a function f exists, if f is piecewise smooth and is of exponential order on the positive real line [18]. A function is said to be piecewise smooth on an interval if it is piecewise differentiable on that interval.

Definition 4.2. Laplace Exists

```
⊢ ∀ s f. laplace exists f s ⇔
  (∀ b. f piecewise_differentiable_on interval [lift (&0),lift b] ) ∧
  (∃ M a. Re s > drop a ∧ exp_order_cond f M a)
```

The function `exp_order_cond` in the above definition represents the exponential order condition necessary for the existence of the Laplace transform [19, 18]:

Definition 4.3. Exponential order Function

```
⊢ ∀ f M a. exp_order f M a ⇔ &0 < M ∧ (∀ t. &0 <= t ⇒ norm (f (lift t)) <= M * exp (drop a * t))
```

We used Definitions 4.1, 4.2 and 4.3 to formally verify some of the classical properties of Laplace transform, given in Table 1. The properties namely linearity, frequency shifting, differentiation and integration were already verified using the formal definition of the Laplace transform [19]. We formally verified these using our new definition of the Laplace transform. Moreover, we formally verified some new properties, such as, time shifting, time scaling, cosine and sine-based modulations and the Laplace transform of a n-order differential equation, which plays a vital role in the formal verification of the transfer function of the n-order linear control system, as described in the next paragraph. The assumptions of these theorems describe the existence of the corresponding Laplace transforms. For example, the predicate `laplace_exists_higher_deriv` in the theorem corresponding to the n-order differential equation ensures that the Laplace of all the derivatives up to the n^{th} order of the function `f` exist. Similarly, the predicate `differentiable_higher_derivative` provides the differentiability of the function `f` and its higher derivatives upto the n^{th} order. The verification of these properties not only ensures the correctness of our definitions but also plays a vital role in minimizing the user effort in reasoning about Laplace transform based analysis of systems, as will be depicted in Sections 5 and 6 of this report.

Table 1: Properties of Laplace Transform

Mathematical Form	Formalized Form
	Integrability
$e^{-st}f(t)$ *integrable on* $[0,\infty)$	`⊢ ∀ f s. laplace_exists f s ⇒ (λt. cexp (--(s * Cx (drop t))) * f t) integrable_on {t | &0 <= drop t}`
	Linearity
$\mathcal{L}[\alpha f(t)+\beta g(t)] = \alpha F(s)+\beta G(s)$	`⊢ ∀ f g s a b. laplace_exists f s ∧ laplace_exists g s ⇒ laplace_transform (λt. a * f t + b * g t) s = a * laplace_transform f s + b * laplace_transform g s`
	Frequency Shifting
$\mathcal{L}[e^{s_0 t}f(t)] = F(s-s_0)$	`⊢ ∀ f s s0. laplace_exists f s ⇒ laplace_transform (λt. cexp (s0 * Cx (drop t)) * f t) s = laplace_transform f (s - s0)`
	First-order Differentiation in Time Domain
$\mathcal{L}\left[\frac{d}{dt}f(t)\right] = sF(s)-f(0)$	`⊢ ∀ f s. laplace_exists f s ∧ (∀t. f differentiable at t) ∧ laplace_exists (λt. vector_derivative f (at t)) s ⇒ laplace_transform (λt. vector_derivative f (at t)) s = s * laplace_transform f s - f (lift (&0))`
	Higher-order Differentiation in Time Domain
$\mathcal{L}[\frac{d^n}{dt^n}f(t)] = s^nF(s) - \sum_{k=1}^{n} s^{k-1}\frac{d^{n-k}f(0)}{dx^{n-k}}$	`⊢ ∀ f s n. laplace_exists_higher_deriv n f s ∧ (∀t. differentiable_higher_derivative n f t) ⇒ laplace_transform (λt. higher_vector_derivative n f t) s = s pow n * laplace_transform f s - vsum (1..n) (λx. s pow (x - 1) * higher_vector_derivative (n - x) f (lift (&0)))`
	Integration of Time Domain

$\mathcal{L}\left[\int_0^t f(\tau)d\tau\right] = \frac{1}{s}F(s)$	⊢ ∀ f s. &0 < Re s ∧ laplace_exists f s ∧ laplace_exists (λx. integral (interval [lift (&0),x]) f) s ∧ (∀x. f continuous_on interval [lift (&0),x]) ⇒ laplace_transform (λx. integral (interval [lift (&0),x]) f) s = Cx(&1) / s * laplace_transform f s
Time Shifting	
$\mathcal{L}[f(t-t_0)u(t-t_0)] = e^{-t_0 s}F(s)$	⊢ ∀ f s t0. &0 < drop t0 ∧ laplace_exists f s ⇒ laplace_transform (shifted_fun f t0) s = cexp (--(s * Cx (drop t0))) * laplace_transform f s
Time Scaling	
$\mathcal{L}[f(ct)] = \frac{1}{c}F\left(\frac{s}{c}\right),\ 0 < c$	⊢ ∀ f s c. &0 < c ∧ laplace_exists f s ∧ laplace_exists f(s / Cx c) ⇒ laplace_transform (λt. f(c % t)) s = Cx(&1) / Cx c * laplace_transform f(s / Cx c)
Modulation (Cosine and Sine Based)	
$\mathcal{L}[f(t)cos(\omega_0 t)] = \frac{F(s-j\omega_0)}{2} + \frac{F(s+j\omega_0)}{2}$	⊢ ∀ f s w0. laplace_exists f s ⇒ laplace_transform (λt. ccos (Cx w0 * Cx (drop t)) * f t) s = laplace_transform f (s − ii * Cx w0) / Cx(&2) + laplace_transform f (s + ii * Cx w0) / Cx(&2)
$\mathcal{L}[f(t)cos(\omega_0 t)] = \frac{F(s-j\omega_0)}{2j} - \frac{F(s+j\omega_0)}{2j}$	⊢ ∀ f s w0. laplace_exists f s ⇒ laplace_transform (λt. csin (Cx w0 * Cx (drop t)) * f t) s = laplace_transform f (s − ii * Cx w0) / Cx(&2) * ii − laplace_transform f (s + ii * Cx w0) / Cx(&2) * ii
n-order Differential Equation	
$\mathcal{L}\left(\sum_{k=0}^{n}\alpha_k\frac{d^k y}{dt^k}\right) = F(s)\sum_{k=0}^{n}\alpha_k s^k - \sum_{k=0}^{n}\sum_{i=1}^{k} s^{i-1}\frac{d^{k-i}f(0)}{dt^{k-i}}$	⊢ ∀ f lst s n. laplace_exists_higher_deriv n f s ∧ (∀x. differentiable_higher_derivative n f x) ⇒ laplace_transform (λt. diff_eq_n_order n lst f t) s = laplace_transform f s * vsum (0..n) (λk. EL k lst * s pow k) - vsum (0..n) (λk. EL k lst * vsum (1..k) (λi. s pow (i - 1) * higher_vector_derivative (k - i) f (lift (&0))))

The generalized linear differential equation describes the input-output relationship for a generic n-order linear control system [10]:

$$\sum_{k=0}^{n}\alpha_k\frac{d^k}{dt^k}y(t) = \sum_{k=0}^{m}\beta_k\frac{d^k}{dt^k}x(t), \quad m \leq n \tag{2}$$

where $y(t)$ is the output and $x(t)$ is the input to the system. The constants α_k and β_k are the coefficients of the output and input differentials with order k, respectively. The greatest index n of the non-zero coefficient α_n determines the order of the underlying system. The corresponding transfer function is obtained by setting the initial conditions equal to zero [10]:

$$\frac{Y(s)}{X(s)} = \frac{\sum_{k=0}^{m}\beta_k s^k}{\sum_{k=0}^{n}\alpha_k s^k} \tag{3}$$

We verified the transfer function, given in Equation 3, for the generic n-order linear control system as the following HOL Light theorem.

Theorem 4.1. Transfer Function of a Generic n-order Linear Control System

```
⊢ ∀ y x m n inlst outlst s. (∀t. differentiable_higher_deriv m n x y t) ∧
   laplace_exists_of_higher_deriv m n x y s ∧ zero_init_conditions m n x y ∧
   diff_eq_n_order_sys m n inlst outlst y x ∧ ∼(laplace_transform x s = Cx (&0)) ∧
   ∼(vsum (0..n) (λt. EL t outlst * s pow t) = Cx (&0))
            ⇒ laplace_transform y s / laplace_transform x s = vsum (0..m) (λt. EL t inlst * s pow t) / vsum (0..n) (λt. EL t outlst * s pow t)
```

The first assumption ensures that the functions `y` and `x` are differentiable up to the n^{th} and m^{th} order, respectively. The next assumption represents the Laplace transform existence condition upto the n^{th} order derivative of function `y` and m^{th} order derivative of the function `x`. The next assumption models the zero initial conditions for both of the functions `y` and `x`, respectively. The next assumption represents the formalization of Equation 2

and the last two assumptions provide the conditions for the design of a reliable linear control system. Finally, the conclusion of the above theorem represents the transfer function given by Equation 3. The verification of this theorem is very useful as it allows to automate the verification of the transfer function of any linear control system as described in Sections 5 and 6 of the report. The formalization, described in this section, took around 2000 lines of HOL Light code and around 130 man-hours. The proof script is available at [20].

5 Formalization of Linear Control Systems Foundations

A general closed-loop control system is depicted in Figure 2. Here, $X(s)$ and $Y(s)$ represent the Laplace transforms of the time domain input $x(t)$ and the output $y(t)$, respectively. $G(s)$ and $H(s)$ represent the forward path and the feedback path transfer functions, respectively. Similarly, $G(s)H(s)$ is the open loop transfer function of the system and $Y(s)/X(s)$ is the closed loop transfer function [21]. Table 2 presents the formalization of the frequency response, phase margin and gain margin of this control system. These properties are used to study the dynamics of a linear control system in the frequency domain and to perform its stability analysis.

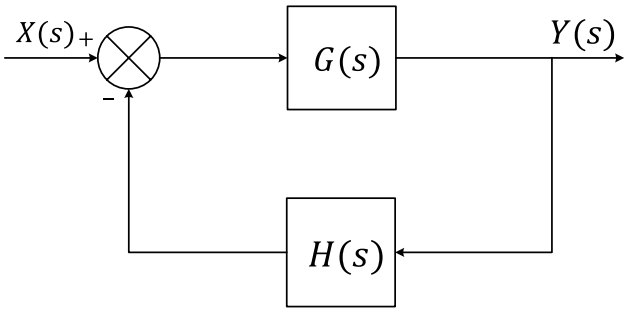


Figure 2: Closed Loop Control System

The frequency response is used to analyze the dynamics of the system by studying the impact of different frequency components on the intended behaviour of the given linear control system. We also formally verified the frequency response of a generic n-order system based on assumptions that are very similar to the ones used for Theorem 4.1.

Table 2: Properties of Linear Control Systems

Mathematical Form	Formalized Form
Frequency Response	
$M(j\omega) = M(s)\|_{(j\omega)} = \left.\frac{Y(s)}{X(s)}\right\|_{(j\omega)} = \frac{Y(j\omega)}{X(j\omega)}$	`⊢ ∀ y x w. frequency_response x y w =` $\frac{\texttt{laplace\_transform y (ii * Cx w)}}{\texttt{laplace\_transform x (ii * Cx w)}}$
Frequency Response of an n-order System	

$\frac{Y(j\omega)}{X(j\omega)} = \frac{\sum_{k=0}^{m} \beta_k (j\omega)^k}{\sum_{k=0}^{n} \alpha_k (j\omega)^k}$	`⊢ ∀ y x m n inlst outlst s.`
	`(∀t. differentiable_higher_deriv m n x y t) ∧`
	`laplace_exists_of_higher_deriv m n x y w ∧`
	`zero_init_conditions m n x y ∧`
	`diff_eq_n_order_sys m n inlst outlst y x ∧`
	`non_zero_denom_cond n x w outlst ⇒`
	`frequency_response x y w =`
	$\frac{\texttt{vsum (0..m) (λt. EL t inlst * (ii * Cx w) pow t)}}{\texttt{vsum (0..n) (λt. EL t outlst * (ii * Cx w) pow t)}}$

Phase Margin	
$[\angle G(j\omega)H(j\omega)]_{\omega=\omega_{gc}} + 180^o$	`⊢ ∀ g h wgc. phase_margin g h wgc =` `pi + Arg (g (ii * Cx wgc) * h (ii * Cx wgc))`
Gain Margin	
$\left[20log_{10}\left\|G(j\omega)H(j\omega)\right\|_{\omega=\omega_{pc}}\right] dB$	`⊢ ∀ g h wpc. gain_margin_db g h wpc = &20 *` $\frac{\texttt{log (norm (g (ii * Cx wpc) * h (ii * Cx wpc)))}}{\texttt{log (\&10)}}$

Phase margin and gain margin provide useful information about controlling the stability of the system [21]. Phase margin represents 180^o shifted phase angle of the open loop transfer function evaluated at the gain crossover frequency (ω_{gc}), which is the frequency at which the magnitude of the open loop transfer function is equal to 0 dB. The gain margin represents the magnitude of the open loop transfer function evaluated at the phase crossover frequency (ω_{pc}), which is the frequency at which the resultant phase curve of the open loop gain has a phase of 180^o. In our formal definitions of these notions, the function `Arg(z)` represents the argument of a complex number `z`.

The controllers form the most vital part of any control system as they are mainly responsible for the correct operation of every component of the underlying system. Controllers are modeled using their active realizations based on an electrical circuit, which comprises of an inverting operational amplifier (op-amp) with unity gain, and two components, i.e., C_A and C_B, which are shown as rectangular boxes in Figure 3a. The boxes C_A and C_B contain different configurations of the passive components, i.e., resistors and capacitors [22]. By making an appropriate choice of these passive components, we obtain various controllers, such as P, I, D, PI, PD, PID [10]. For the analysis of these controllers, we first need to formalize them in higher-order logic. This step requires a formal library of analog components [23, 20], describing the voltage-current relationships of resistor, capacitors and inductors, and the KCL and KVL, which model the currents and voltages in an electrical circuit.

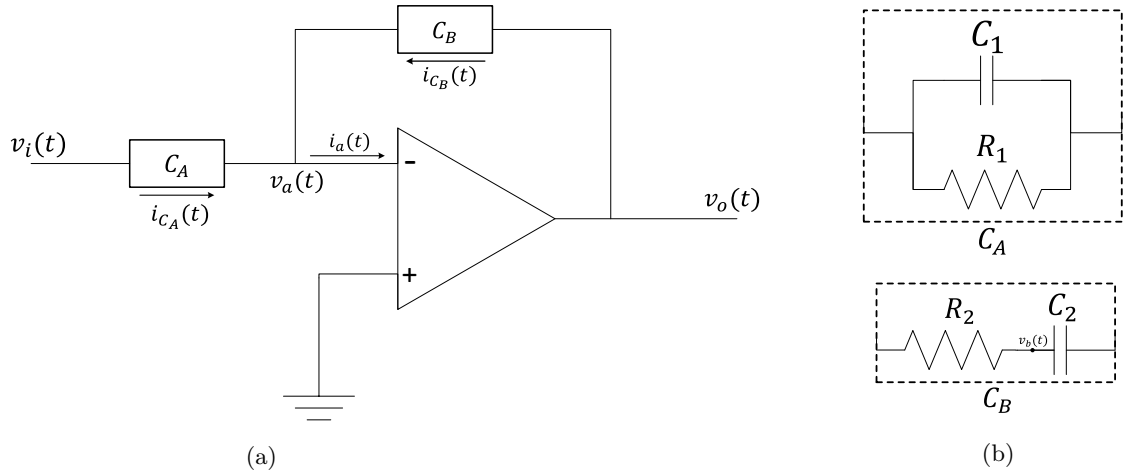


Figure 3: Controller (a) Generic Active Realization (b) PID Configuration

The PID controller, depicted in Figure 3b, can be formalized as follows:

Definition 5.1. Implementation of the PID Controller

```
⊢ ∀ C1 R1 Vi R2 C2 Vo Vb Va. pid_controller_implem Vi Vo Va Vb C1 C2 R1 R2 ⇔
        (∀t. &0 < drop t ⇒ kcl [λt. capacitor_current C1 (λt.  Vi t - Va t) t;
                                λt. resistor_current R1 (λt. Vi t - Va t) t;
                                λt. resistor_current R2 (λt. Vb t - Va t) t] t ∧
        (∀t. &0 < drop t ⇒ kcl [λt. resistor_current R2 (λt.  Va t - Vb t) t;
                                λt. capacitor_current C2 (λt. Vo t - Vb t) t] t ∧
        (∀t. &0 < drop t ⇒ Va t = Cx (&0))
```

where `Vi` and `Vo` are the input and the output voltages, respectively, having data type $\mathbb{R}^1 \to \mathbb{C}$, and `Va` and `Vb` are the voltages at nodes a and b, respectively. The functions `resistor_current` and `capcitor_current` are the currents across the resistor and capacitor, respectively. The function `kcl` accepts a list of currents across the components of the circuit and a time variable `t` and returns the predicate that guarantees that the sum of all the currents leaving a particular node at time `t` is zero. The first conjunct of the above definition represents the application of KCL across node a. Similarly, the second conjunct models the KCL at node b, whereas the last conjunct provides the voltage across the non-inverting input of the op-amp using the virtual ground condition, as shown in Figure 3a. We also develop a simplification tactic `KCL_SIMP_TAC`, which simplifies the implementations of the PID controller as well as the other controllers and compensators of Tables 3 and 4.

Next, we model the dynamical behaviour of the PID controller using the n-order differential equation:

Definition 5.2. Behavioural Specification of PID Controller

```
⊢ ∀ R1 R2 C1 C2.
     inlst_pid_contr R1 R2 C1 C2 = [--Cx (&1); --Cx (R2 * C2 + R1 * C1); --Cx (R1 * R2 * C1 * C2)]
⊢ ∀ R1 C2. outlst_pid_contr R1 C2 = [Cx (&0); Cx (R1 * C2)]
⊢ ∀ Vo R1 R2 C1 C2 Vi t. pid_controller_behav_spec R1 R2 C1 C2 Vi Vo t ⇔
                diff_eq_n_order 1 (outlst_pid_contr R1 C2) Vo t =
                diff_eq_n_order 2 (inlst_pid_contr R1 R2 C1 C2) Vi t
```

We verified the behavioural specification based on the implementation of the PID controller as the following theorem:

Theorem 5.1. Verification of the Dynamical Behavioural Specification

```
⊢ ∀ R1 R2 C1 C2 Vi Va Vb Vo t.
        &0 < R1 ∧ &0 < R2 ∧ &0 < C1 ∧ &0 < C2 ∧
        (∀t. differentiable_higher_derivative Vi Vo Vb t) ∧
        pid_controller_implem Vi Vo Va Vb C1 C2 R1 R2
                 ⇒ (&0 < drop t ⇒ pid_controller_behav_spec R1 R2 C1 C2 Vi Vo t)
```

The first four assumptions model the design requirement for the underlying system. The next assumption provides the differentiability of the higher-order derivatives of `Vi`, `Vo` and `Vb` up to the order 1, 2 and 2, respectively. The last assumption presents the implementation for the PID controller. Finally, the conclusion presents its behavioral specification. We also develop a simplification tactic `DIFF_SIMP_TAC`, which simplifies the behavioural specifications of the PID controller as well as the other controllers and compensators presented in Tables 3 and 4.

Next, we verified the transfer function of the PID controller as follows:

Theorem 5.2. Verification of the Transfer Function

```
⊢ ∀ R1 R2 C1 C2 Vi Vo s t. &0 < R1 ∧ &0 < R2 ∧ &0 < C1 ∧ &0 < C2 ∧
  ∼(laplace_transform Vi s = Cx (&0)) ∧ ∼(Cx R1 * Cx C2 * s = Cx (&0)) ∧
  (∀t. differentiable_higher_derivative Vi Vo t) ∧
  laplace_exists_higher_deriv Vi Vo s ∧ zero_initial_conditions Vi Vo ∧
  (∀t.  pid_controller_behav_spec R1 R2 C1 C2 Vi Vo t)
```

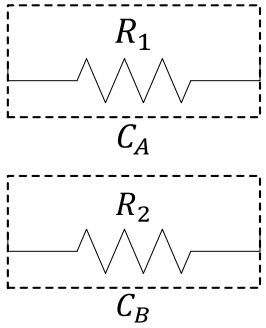
$$\Rightarrow \frac{\texttt{laplace\_transform Vo s}}{\texttt{laplace\_transform Vi s}} = \frac{\texttt{--}\big(\texttt{Cx}(\texttt{R1}*\texttt{C1}*\texttt{R2}*\texttt{C2})*\texttt{s pow 2} + (\texttt{Cx}(\texttt{R2}*\texttt{C2}) + \texttt{Cx}(\texttt{C1}*\texttt{R1}))*\texttt{s} + \texttt{Cx}(\&1)\big)}{\texttt{Cx}(\texttt{R1}*\texttt{C2})*\texttt{s}}$$

The first six assumptions present the design requirements for the underlying system. The next two assumptions provide the differentiability and the Laplace existence condition for the higher-order derivatives of `Vi` and `Vo` up to the order 2 and 1, respectively. The next assumption models the *zero initial conditions* for the voltage functions `Vi` and `Vo`. The last assumption presents the behavioural specification of the PID controller. Finally, the conclusion of Theorem 5.2 presents its required transfer function. By judicious selection of the configuration of passive components, we obtain various controllers, such as P, I, D, PI, PD and perform the above-mentioned analysis for all of them.

Compensators are widely used in control systems, to improve their frequency response, steady-state error and the stability and hence, act as a fundamental block of a control system. Like controllers, the compensators are also modeled using their active realizations. A compensator uses the same analog circuit, which was used for the controllers, presented in Figure 3a, by making an appropriate choice of the passive components C_A and C_B, as given in Table 3. It acts as a lag-compensator under the condition $R_2C_2 > R_1C_1$, whereas for the case of $R_1C_1 > R_2C_2$, it acts as a lead-compensator. The configurations of the passive components for the controllers and compensators, and their formalization is presented in Table 3. The details about their analysis can be found in [20].

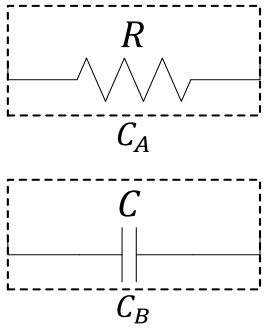
Table 3: Modeling of Active Realization of Different Controllers and Compensators

Configuration	Implementation
	Proportional (P)



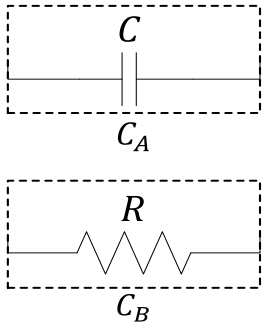

```
⊢ ∀ R1 Vi R2 Vo Va. p_controller_implem Vi Vo Va R1 R2 ⇔
   (∀t. &0 < drop t
            ⇒ kcl [λt. resistor_current R1 (λt. Vi t - Va t) t;
                   λt. resistor_current R2 (λt. Vo t - Va t) t] t) ∧
   (∀t. &0 < drop t ⇒ Va t = Cx (&0))
```

Integral (I))



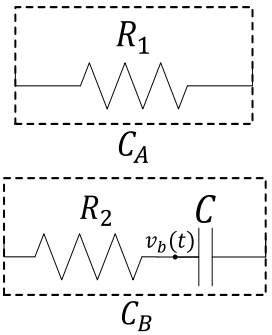

```
⊢ ∀ R Vi C Vo Va. i_controller_implem Vi Vo Va C R ⇔
   (∀t. &0 < drop t
            ⇒ kcl [λt. resistor_current R (λt. Vi t - Va t) t;
                   λt. capacitor_current C (λt. Vo t - Va t) t] t) ∧
   (∀t. &0 < drop t ⇒ Va t = Cx (&0))
```

Derivative (D)



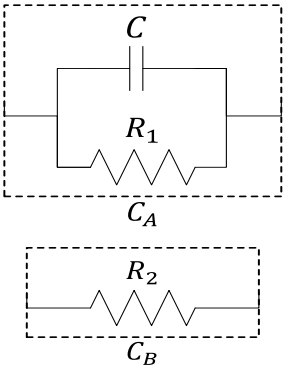

```
⊢ ∀ C Vi R Vo Va. d_controller_implem Vi Vo Va C R ⇔
   (∀t. &0 < drop t
            ⇒ kcl [λt. capacitor_current C (λt. Vi t - Va t) t;
                   λt. resistor_current R (λt. Vo t - Va t) t] t) ∧
   (∀t. &0 < drop t ⇒ Va t = Cx (&0))
```

Proportional-Integral (PI)



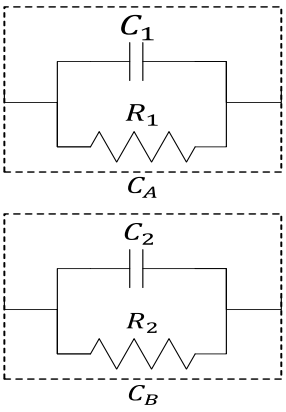

```
⊢ ∀ R1 Vi R2 C Vo Vb Va.
   pi_controller_implem Vi Vo Va Vb C R1 R2 ⇔
   (∀t. &0 < drop t
            ⇒ kcl [λt. resistor_current R1 (λt. Vi t - Va t) t;
                   λt. resistor_current R2 (λt. Vb t - Va t) t] t) ∧
   (∀t. &0 < drop t
            ⇒ kcl [λt. resistor_current R2 (λt. Va t - Vb t) t;
                   λt. capacitor_current C (λt. Vo t - Vb t) t] t) ∧
   (∀t. &0 < drop t ⇒ Va t = Cx (&0))
```

Proportional-Derivative (PD)




```
⊢ ∀ C R1 Vi R2 Vo Va.
   pd_controller_implem Vi Vo Va C R1 R2 ⇔
   (∀t. &0 < drop t
            ⇒ kcl [λt. capacitor_current C (λt. Vi t - Va t) t;
                   λt. resistor_current R1 (λt. Vi t - Va t) t;
                   λt. resistor_current R2 (λt. Vo t - Va t) t] t) ∧
   (∀t. &0 < drop t ⇒ Va t = Cx (&0))
```

Compensator




```
⊢ ∀ C1 R1 Vi C2 R2 Vo Va.
   compensator_implem Vi Vo Va C1 C2 R1 R2 ⇔
   (∀t. &0 < drop t
            ⇒ kcl [λt. capacitor_current C1 (λt. Vi t - Va t) t;
                   λt. resistor_current R1 (λt.  Vi t - Va t) t;
                   λt. capacitor_current C2 (λt.  Vo t - Va t) t;
                   λt. resistor_current R2 (λt.  Vo t - Va t) t] t) ∧
   (∀t. &0 < drop t ⇒ Va t = Cx (&0))
```

Compensators are also modeled using their passive realizations based on an electrical circuit, which comprises of two components, i.e., C_A and C_B, which are shown as rectangular boxes in Figure 4. The boxes C_A and C_B

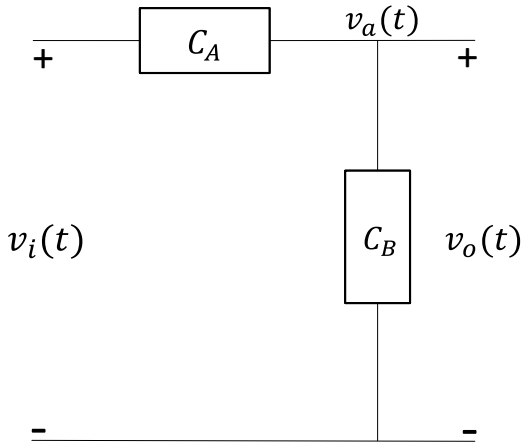
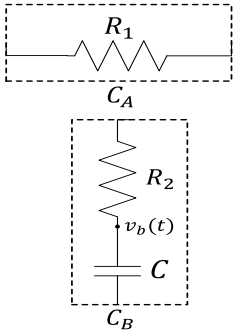
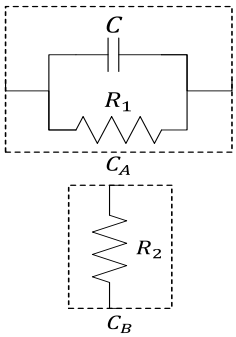
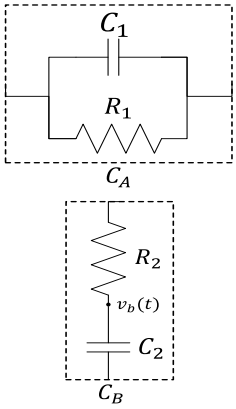


Figure 4: Generic Passive Realization of Compensator

contain different configurations of the passive components, i.e., resistors and capacitors. By making an appropriate choice of these passive components, we obtain various compensators, such as lag, lead and lag-lead [10]. The configurations of the passive components for the compensators and their formalization in HOL Light is presented in Table 4. The details about their analysis can be found in [20].

The formalization presented in this section took around 300 lines of HOL Light code and around 14 man-hours, which clearly illustrates the effectiveness of our foundational formalization, presented in the previous section, as the verification steps involved the linearity, differentiation in time domain, n-order differential properties of Laplace transform and the transfer function of an n-order linear control system.

Table 4: Modeling of Passive Realization of Different Compensators

Configuration	Implementation
Lag Compensator	
	⊢ ∀ R1 Vi R2 C Vb Va Vo. lag_compensator_implem Vi Vo Va Vb C R1 R2 ⇔ (∀t. &0 < drop t ⇒ kcl [λt. resistor_current R1 (λt. Vi t - Va t) t; λt. resistor_current R2 (λt. Vb t - Va t) t] t) ∧ (∀t. &0 < drop t ⇒ kcl [λt. resistor_current R2 (λt. Va t - Vb t) t; λt. capacitor_current C (λt. (λt. Cx (&0)) t - Vb t) t] t) ∧ (∀t. &0 < drop t ⇒ Va t = Vo t)
Lead Compensator	
	⊢ ∀ C R1 Vi R2 Vo Va. lead_compensator_implem Vi Vo Va C R1 R2 ⇔ (∀t. &0 < drop t ⇒ kcl [λt. capacitor_current C (λt. Vi t - Va t) t; λt. resistor_current R1 (λt. Vi t - Va t) t; λt. resistor_current R2 (λt. (λt. Cx (&0)) t - Va t) t] t) ∧ (∀t. &0 < drop t ⇒ Va t = Vo t)
Lag-lead Compensator	
	⊢ ∀ C1 R1 Vi R2 C2 Vb Va Vo. lag_lead_compensator_implem Vi Vo Va Vb C1 C2 R1 R2 ⇔ (∀t. &0 < drop t ⇒ kcl [λt. capacitor_current C1 (λt. Vi t - Va t) t; λt. resistor_current R1 (λt. Vi t - Va t) t; λt. resistor_current R2 (λt. Vb t - Va t) t] t) ∧ (∀t. &0 < drop t ⇒ kcl [λt. resistor_current R2 (λt. Va t - Vb t) t; λt. capacitor_current C2 (λt. (λt. Cx (&0)) t - Vb t) t] t) ∧ (∀t. &0 < drop t ⇒ Va t = Vo t)

6 Unmanned Free-Swimming Submersible Vehicle

Unmanned Free-Swimming Submersible (UFSS) vehicles are a kind of autonomous underwater vehicles (AUVs) that are used to perform different tasks and operations in the submerged areas of the water. These vehicles have their own power and control systems, which are autonomously operated and controlled by the onboard computer system without any involvement of human assistance as it is difficult for humans to work in an underwater environment. UFSS vehicles are used in many safety-critical domains to perform different tasks, such as underwater navigation and object detection [24], performing deep sea rescue and salvage operations [25], searching for seamines [26] and securing sea harbour [26]. Due to their wider usage in the above-mentioned safety-critical applications, an accurate analysis of their control system is of utmost importance.

We present a formal analysis of the pitch control system of a UFSS vehicle. The pitch control system is responsible for the uninterrupted operation and functionality of the UFSS vehicle by manipulating different parameters, such as, elevator surface, pitch angle [10]. Figure 5 depicts its block diagram.

The dynamics of the UFSS vehicle are represented by its corresponding differential equation, which presents the relationship between the pitch command angle $\theta_e(t)$ and the pitch angle $\theta(t)$, and is given as follows:

$$\frac{d^4\theta}{dt^4} + 3.456\frac{d^3\theta}{dt^3} + (3.207 + 0.25K_2)\frac{d^2\theta}{dt^2} + (0.616 + 0.1088K_2 + 0.25K_1)\frac{d\theta}{dt} + (0.1088K_1 + 0.0416) = 0.25K_1\frac{d\theta_e}{dt} + 0.1088K_1 \tag{4}$$

We formalize the above differential equation as follows:

Definition 6.1. Differential Equation of the UFSS Pitch Control System

```
⊢ ∀ K1. inlst_ufsv K1 = [Cx (#0.1088) * Cx K1; Cx (#0.25) * Cx K1]
⊢  ∀ K1 K2. outlst_ufsv K1 K2 =
[Cx (#0.1088) * Cx K1 + Cx (#0.0416); Cx (#0.25) * Cx K1  + Cx (#0.1088) * Cx K2 +  Cx (#0.6106);
Cx (#0.25) * Cx K2 + Cx (#3.207); Cx (#3.456); Cx (&1)]
⊢ diff_eq_ufsv inlst_ufsv outlst_ufsv theta thetae K1 K2 ⇔
   (∀t. diff_eq_n_order 4 (outlst_ufsv K1 K2) theta t = diff_eq_n_order 1 (inlst_ufsv K1) thetae t)
```

where `thetae` and `theta` represent the input and the output of the pitch control system and `K1` and `K2` are the pitch gain and pitch rate sensor gain, respectively. The symbol # is used to represent a decimal number of data type $\mathbb{R}$ in HOL Light and is same as symbol & for the integer literal of data type $\mathbb{R}$.

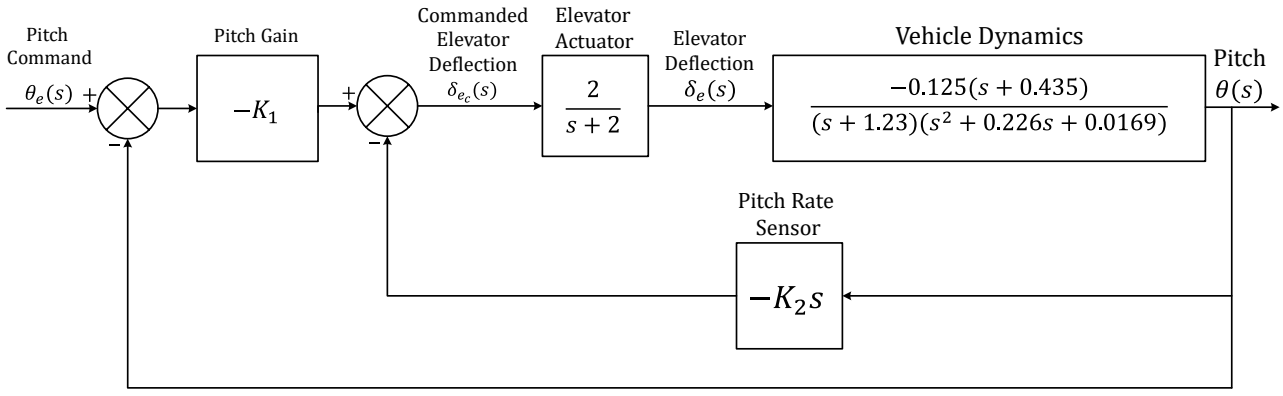


Figure 5: Pitch Control Model for Unmanned Free-swimming Submersible Vehicle

The transfer function of the pitch control of the UFSS vehicle is as follows:

$$\frac{\theta(s)}{\theta_e(s)} = \frac{0.25K_1s + 0.1088K_1}{s^4 + 3.456s^3 + (3.207 + 0.25K_2)s^2 + (0.6106 + 0.1088K_2 + 0.25K_1)s + (0.1088K_1 + 0.0416)} \tag{5}$$

We verified the above transfer function as the following HOL Light theorem:

Theorem 6.1. Transfer Function of the Closed Loop System

```
⊢ ∀ thetae theta s K1 K2. (∀t. differentiable_higher_deriv theta thetae t) ∧
  laplace_exists_of_higher_deriv theta thetae s ∧ zero_init_conditions theta thetae ∧
  diff_eq_ufsv inlst_ufsv outlst_ufsv theta thetae K1 K2 ∧ non_zero_denominator_condition theta s
        ⇒ laplace_transform theta s / laplace_transform thetae s =
                                  (Cx (#0.25) * Cx K1) * s + Cx (#0.1088) * Cx K1
 s pow 4 + Cx (#3.456) * s pow 3 + (Cx (#0.25) * Cx K2 + Cx (#3.207))
   * s pow 2  + (Cx (#0.25) * Cx K1 + Cx (#0.1088) * Cx K2 + Cx (#0.6106))
                                       * s  + Cx (#0.1088) * Cx K1 + Cx (#0.0416)
```

The first two assumptions present the differentiability and the Laplace existence condition of the higher-order derivatives of `thetae` and `theta` upto order 1 and 4, respectively. The next assumption provides the *zero initial*

conditions for `thetae` and `theta`. The next assumption presents the differential equation specification for the pitch control system of UFSS vehicle. The final assumption models the non-negativity of the denomerator of the transfer function presented in the conclusion of the above theorem. We also verified the open loop transfer function $\theta(\mathtt{s})/\delta_{\mathtt{e}}(\mathtt{s})$, frequency response (open and closed loop) and gain margin, for the UFSS vehicle and the details can be found from [20].

The distinguishing feature of the formally verified Theorem 6.1 as compared to traditional analysis methods is its generic nature, i.e., all of the variables and functions are universally quantified and can thus be specialized in order to obtain the results for some given values. Moreover, all of the required assumptions are guaranteed to be explicitly mentioned along with the theorem due to the inherent soundness of the theorem proving approach. The high expressiveness of the higher-order logic enables us to model the differential equation and the corresponding transfer function in their true continuous form, whereas, in model checking they are mostly discretized and modeled using a state-transition system, which can certainly compromise the accuracy of the analysis.

In order to facilitate control engineers to use our formalization for analyzing their linear control system problems, we developed an automatic tactic `TRANSFER_FUN_TAC`, which automatically verifies the transfer function of the systems upto 20^{th}-order, which covers most of the real-world systems. This tactic was successfully used for the automatic verification of the transfer functions of the controllers, compensators and the pitch control system of the UFSS vehicle. This automatic verification tactic only requires the differential equation and the transfer function of the underlying system and automatically verifies the transfer function. The formal analysis of the UFSS vehicles took only 25 lines of code and about half an hour, thanks to our automatic tactic and the foundational formalization of Section 4.

7 Conclusion

This report presented a higher-order-logic theorem proving based approach for the formal analysis of the dynamical aspects of linear control systems using theorem proving. The main idea behind the proposed framework is to use a formalization of Laplace transform theory in higher-order logic to formally analyze the dynamic aspects of linear control systems. For this purpose, we develop a new formalization of Laplace transform theory, which includes its formal definition and verification of its properties, such as linearity, frequency shifting, differentiation and integration in time domain, time shifting, time scaling, cosine and sine-based modulation and the Laplace transform of an n-order differential equation, which are used for the verification of the transfer function of a generic n-order linear control system. Moreover, the report also presents the formal verification of some widely used linear control system characteristics, such as frequency response, phase margin and the gain margin, using the verified transfer function, which can be used for the stability analysis of a linear control system. We also formalize the active realization of various controllers, such as PID, PD, PI, P, I, D, and various compensators, such as lag and lead. Finally, we formalize the passive realization of the various compensators, such as lag, lead and lag-lead and verified the corresponding behavioral (differential equation) and the transfer function specifications. To facilitate the usage of these formalizations in analyzing real-world linear control systems, we developed some simplification and automatic verification tactics, in particular the tactic `TRANSFER_FUN_TAC`, which automatically verifies the transfer function of any real-world linear control system based on its differential equation. These foundations can be used to analyze a wide range of linear control systems and for illustration purposes, the report presents the formal analysis of an unmanned free-swimming submersible vehicle.

In future, we plan to link the proposed formalization with Simulink so that the users can provide the system model as a block diagram. This diagram can be used to extract the corresponding transfer function [27], which can in turn be formally verified, almost automatically, to be equivalent to the corresponding block diagram based on the reported formalization and reasoning support. The other future direction is to use our formalization of Laplace transform for analyzing non-linear systems [28, 29, 30]. The idea is to model the dynamics of the system using any non-linear differential equation and then after its linearization, use the Laplace transform theory in order to verify the properties of any control system under consideration.

References

[1] MD Lutovac and DV Tošić. Symbolic Analysis and Design of Control Systems using Mathematica. *International Journal of Control*, 79(11):1368–1381, 2006.

[2] Osman Hasan and Sofiène Tahar. Formal verification methods. *Encyclopedia of Information Science and Technology, IGI Global Pub*, pages 7162–7170, 2015.

[3] M Edwin Johnson. Model Checking Safety Properties of Servo-loop Control Systems. In *Dependable Systems and Networks*, pages 45–50. IEEE, 2002.

[4] Ashish Tiwari and Gaurav Khanna. Series of Abstractions for Hybrid Automata. In *Hybrid Systems: Computation and Control*, pages 465–478. Springer, 2002.

[5] Osman Hasan and Muhammad Ahmad. Formal Analysis of Steady State Errors in Feedback Control Systems using HOL-Light. In *Design, Automation and Test in Europe*, pages 1423–1426, 2013.

[6] Muhammad Ahmad and Osman Hasan. Formal Verification of Steady-State Errors in Unity-Feedback Control Systems. In *Formal Methods for Industrial Critical Systems*, pages 1–15. Springer, 2014.

[7] Sidi Mohamed Beillahi, Umair Siddique, and Sofiène Tahar. Formal Analysis of Power Electronic Systems. In *Formal Engineering Methods*, pages 270–286. Springer, 2015.

[8] John Harrison. The HOL Light Theory of Euclidean Space. *Journal of Automated Reasoning*, 50(2):173–190, 2013.

[9] S. H. Taqdees and O. Hasan. Formally Verifying Transfer Functions of Linear Analog Circuits. IEEE Design & Test, `http://save.seecs.nust.edu.pk/pubs/2017/DTnA_2017.pdf`, 2017.

[10] Norman S Nise. *Control Systems Engineering*. John Wiley & Sons, 2007.

[11] J. Harrison. HOL Light: A Tutorial Introduction. In *Formal Methods in Computer-Aided Design*, volume 1166 of *LNCS*, pages 265–269. Springer, 1996.

[12] Rob Arthan, Paul Caseley, Colin O'Halloran, and Alf Smith. ClawZ: Control laws in Z. In *Formal Engineering Methods, 2000. ICFEM 2000. Third IEEE International Conference on*, pages 169–176. IEEE, 2000.

[13] Brendan Mahony. The DOVE Approach to the Design of Complex Dynamic Processes. In *NASA Conference Publication*, pages 167–188. NASA; 1998, 2002.

[14] Nikos Aréchiga, Sarah M Loos, André Platzer, and Bruce H Krogh. Using Theorem Provers to Guarantee Closed-loop System Properties. In *American Control Conference (ACC), 2012*, pages 3573–3580. IEEE, 2012.

[15] Richard J Boulton, Ruth Hardy, and Ursula Martin. A Hoare Logic for Single-input Single-output Continuous-time Control Systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 113–125. Springer, 2003.

[16] Richard J Boulton, Hanne Gottliebsen, Ruth Hardy, Tom Kelsey, and Ursula Martin. Design Verification for Control Engineering. In *International Conference on Integrated Formal Methods*, pages 21–35. Springer, 2004.

[17] HOL-Light Transcendental Theory. `https://github.com/jrh13/hol-light/blob/master/Multivariate/transcendentals.ml`, 2017.

[18] R. J. Beerends, H. G. Morsche, J. C. Van den Berg, and E. M. Van de Vrie. *Fourier and Laplace Transforms*. Cambridge University Press, Cambridge, 2003.

[19] S. H. Taqdees and O. Hasan. Formalization of Laplace Transform Using the Multivariable Calculus Theory of HOL-Light. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 744–758. Springer, 2013.

[20] Adnan Rashid. Formal Analysis of Linear Control Systems using Theorem Proving. `http://save.seecs.nust.edu.pk/projects/falcstp`, 2017.

[21] Smarajit Ghosh. *Control Systems*, volume 1000. Pearson Education, 2010.

[22] Katsuhiko Ogata and Yanjuan Yang. Modern Control Engineering. 1970.

[23] S. H. Taqdees. Formalization of Laplace Transform Using the Multivariable Calculus Theory of HOL-Light. `http://save.seecs.nust.edu.pk/Downloads/thesis_hira.pdf`, 2014.

[24] Hayato Kondo and Tamaki Ura. Navigation of an AUV for Investigation of Underwater Structures. *Control Engineering Practice*, 12(12):1551–1559, 2004.

[25] Robert L Wernli. Low Cost UUV's for Military Applications: Is the Technology Ready? In *Pacific Congress on Marine Science and Technology*, 2001.

[26] Scott Willcox, Jerome Vaganay, Robert Grieve, and Jeff Rish. The Bluefin BPAUV: An Organic Widearea Bottom Mapping and Mine-hunting Vehicle. *Unmanned Untethered Submersible Technology*, 2001.

[27] Robert Babuska and Stefano Stramigioli. Matlab and Simulink for Modeling and Control. *Delft University of Technology*, 1999.

[28] Jan Willem Polderman and Jan C Willems. Introduction to the Mathematical Theory of Systems and Control. *Lecture notes, University of Twente, Department of Applied Mathematics Available from http://wwwhome. math. utwente. nl/˜ poldermanjw/onderwijs/DISC/mathmod/book. pdf*, 1998.

[29] Béla Lantos and Lőrinc Márton. *Nonlinear Control of Vehicles and Robots*. Springer Science & Business Media, 2010.

[30] M Massaro and R Lot. Application of Laplace Transform Techniques to Non-linear Control Optimization. *Proc of the multibody dynamics*, pages 25–28, 2007.