Design verification for control engineering

To appear in Proceedings of the Fourth International Conference on Integrated Formal Methods, LNCS xxxx, Springer 2004

Richard J Boulton, Hanne Gottliebsen[1], Ruth Hardy[2], Tom Kelsey[2], and Ursula Martin[1]

[1] Queen Mary University of London,
Ursula.Martin@dcs.qmul.ac.uk
[2] University of St Andrews

Abstract. We introduce control engineering as a new domain of application for formal methods. We discuss design verification, drawing attention to the role played by diagrammatic evaluation criteria involving numeric plots of a design, such as Nichols and Bode plots. We show that symbolic computation and computational logic can be used to discharge these criteria and provide symbolic, automated, and very general alternatives to these standard numeric tests. We illustrate our work with reference to a standard reference model drawn from military avionics.

1 Introduction

To control an object means to influence its behaviour so as to achieve a desired goal. Control systems may be natural mechanisms, such as cellular regulation of genes and proteins by the gene control circuitry in DNA. They may be man-made – an early mechanical example was Watt's steam governor – but today most man-made control systems are digital, for example fighter aircraft or CD drives. In genomics we want to identify the control mechanism from observations of its properties: in engineering we want to solve the dual problem of constructing a system with certain properties. Traditionally, control is treated as a mathematical phenomenon, modelled by continuous or discrete dynamical systems. Numerical computation is used to test and simulate these models, for example Matlab is an industry standard in avionics. A largely separate process is then used in the implementation of such as continuous model as a digital (discrete) controller.

Block diagrams are a standard engineering representation of dynamical systems, obtained from a Laplace transform. In a recent paper [3] we added assertions about phase and gain of a signal to block diagrams and devised and implemented a simple Hoare logic. We were able to emulate mechanically engineer's informal reasoning about phase and gain. We also prototyped symbolic, automated, very general alternatives to some standard numeric tests used in engineering design, and it is this work which forms the main result of this paper.

We replace numeric plots with symbol manipulation in the computer algebra system Maple and the theorem prover PVS. This in turn exploited our Maple-PVS system [12], and Gottliebsens PVS continuity checker [13].

Control engineering is a large subject: we intend to focus on those aspects which are to a control engineer fairly standard and widely used in practice, see for example Ogata [25]. Optimal control assumes that a model of the system is available and one wants to optimise its behaviour, using the calculus of variations and so forth: for example pre-computing a desired flight-path for a spacecraft. Feedback control compensates for uncertainty in the model by using feedback to correct for deviations from desired behaviour: for example if the spacecraft strays off course. Models vary according to the application: for example differential equations are used when modelling a continuous signal, but these are replaced by difference equations when modelling a sampled signal as used in digital systems. In reasoning about such systems we are interested not only in the solutions, but in their properties. These include the time response, stability, frequency response and behaviour under perturbation. The time response considers features such as the time taken for a property of the system (e.g. the cruising speed of a car) to reach the desired value, and by how far it overshoots before settling at the desired value. Stability analysis considers whether the system will always settle into a steady state following a change to the input(s). An output of an unstable system may increase out of control or oscillate. Frequency response considers the amplitude and phase shift of the output signal when the system is presented with a sinusoidal input. The analysis considers input signals with a range of frequencies.

In practice systems are rarely linear: non-linear systems are generally treated locally by linearising at points of interest, but global behaviour is the subject of much research and raises subtle questions in differential and algebraic geometry.

In classical control a Laplace transform is applied to a linear system to obtain a representation as a transfer function, a rational function over the complexes. Analysis of properties, such as frequency and response of the control system, is in terms of the position of its roots and poles in the complex plane. So called modern control considers a state-space representation, which replaces a single differential equation with a system of simultaneous equations in the state variables, and analyses the system via properties of the eigenvalues of a related matrix. Both frameworks can be extended from SISO (single input/output) to MIMO (multiple input/output) systems.

Block diagrams are often used to represent systems with feedback graphically, for example in classical control a block diagram is a directed graph whose edges are labelled by rational functions over the complexes. They also allow more general representation of components described only by their input/output behaviour.

Software such as the widely used Mathworks Simulink [21], the industry standard in avionics and automotive applications, supports numerical tests and simulations. A number of standard tests are used for prediction and analysis: for example the Nichols plot is a numerical test which investigates stability. It

displays the steady state behaviour of a classical control system in terms of the phase and gain of a sinusoidal input. We shall see below how some of the control requirements of fighter aircraft are specified in terms of acceptable paths in this plot [26].

In practice man-made control systems are typically digital embedded software systems, which use sampled, rather than continuous time. These can be modelled as discrete dynamical systems (difference equations), which again admit a transform representation via the z-transform, and an analogous state-space representation, investigated as before using matrix algebra. The design of a digital controller, for example in avionics applications, typically involves analysis as above in continuous time: it is then passed to a software team for implementation as a discrete digital system. It has been suggested that this process is a likely source of error: indeed apparently "similar" continuous and discrete systems may have very different stability properties. The ubiquity of such embedded controllers, for example in cars and domestic appliances, has led to increased interest in methods of generating assured code straight from a high level design [2].

The study of control in the context of computer science is an emerging area: we identify some strands of work which complement our own:

- The most well developed is the field of hybrid systems, which models certain control systems as automata with discrete transitions which are then amenable to model checking [20] and theoretical analysis [18]. Alur and Dill [5] introduced timed automata, state-transition diagrams annotated with timing constraints using finitely many real-valued clock variables which can be used to model discrete dynamical systems
- In the 1970s Arbib and Manes [1] studied categorical models of linear control: more recently various categories with feedback have been much studied, especially traced monoidal [16] categories, which are models for linear logic. These seem to obey similar algebraic laws to feedback diagrams, though as far as we know the connection has not been developed formally. Tourlas [14] has studied reasoning about general diagram languages.
- Less attention has been paid to the classical dynamical systems representations. Perhaps the closest foundational work is Edalat's [9] extension of classical domain theory to analysis and dynamical systems. Tiwari [29] allows abstraction of dynamical systems to a level where model checking can be used.
- Our own work on light formal methods for mathematical systems [6, 7] was a precursor to this work: we devised an assertion language and lint-like checker for NAG Ltd's AXIOM system.
- The widespread use of Simulink suggests that effective formal verification techniques for block diagrams could have significant impact. The ClawZ system of Arthan et al [2], developed for QinetiQ, is a first step: it translates discrete-time models, described using Simulink, into formal specifications in Z. A controller implementation in an Ada-like programming language can then be verified against these Z specifications using the ProofPower mechanised proof assistant. QinetiQ used ClawZ in a case study of the braking

systems of the Eurofighter, and are addressing issues of concurrency in this framework using CSP/FDR. Mahony [23] has used similar ideas in his DOVE system.

In a pilot study [3] we developed a Hoare-style logic and a verification condition generator for a restricted class of block diagrams, essentially those with a tree structure. Hoare logics [17] were originally studied by Hoare, Floyd and others to give an axiomatic basis for programming, and continue to be used for a variety of applications, for example Java byte-code verification [24]. As far as we know our work is the first to investigate Hoare-style logics for feedback systems. We attached assertions to nodes in the diagram: the key observation was that phase and gain were compositional, and hence we could reason about them locally, and propagate our reasoning through the diagram to deduce properties of a classical frequency-response analysis. Following Gordon's approach [11] the logic was mechanised in the HOL98 theorem proving system, allowing goal-directed reasoning, machine assistance in the details of the proof, and automatic generation of verification conditions, the logical formulas that must ultimately be proved to justify an assertion in the Hoare logic. The verification conditions themselves are pure predicate logic formulas, that is, they do not involve the constructs of our logic.

2 Design requirements for aerospace applications

As one might expect the design and verification of flight control laws, and their implementation in flight control systems of civilian or military aircraft, is a mature technology involving many specialist teams in engineering companies and certification authorities - Pratt [26] provides a comprehensive account. Producing a new plane typically takes around three years from the official programme start to the first test flight, and can cost upwards of a billion dollars. The customer for a civilian aircraft might expect to get their first plane about a year after the first test flight: the flight test programme for military aircraft may be significantly longer as they are more likely to contain novel technologies to give enhanced performance, and these need to be tested across a greater range of highly dynamic missions, a wider flight envelope and varying payloads. The flight control algorithms represent between five and ten per cent of the flight control law (FCL) software on an aircraft, and this in turn represents around a quarter of the activity of the flight control system (FCS): the rest is associated with safety monitoring, redundancy management and so on. Typically development of the FCL and FCS follows the "V-model", moving down through the specification of the aircraft, system, equipment and components to manufacturing and coding and then up through an integration path involving suitable validation/verification at each level. Design engineers at the component level initially produce a control law for a (continuous) model of the airplane to meet certain design requirements: this is subsequently implemented as a digital controller and then incorporated in the design and test cycle for the full FCS.

The controller is designed to obtain certain desired behaviours of the controlled system: for example we might want it to be stable, and to behave in a specified way, to within an error bound, in response to a given input function, such as a step function, or a sine wave. The desired behaviours constitute design requirements. For larger systems these may be expressed in two parts: a high level description of a property, for example "good flying qualities", and a corresponding evaluation criterion, expressed as a precise mathematical property that the controlled system is required to have to discharge the design requirement. It is these properties that are often expressed as showing that a particular curve avoids a partcular region, and discharged by plotting a graph.

As an example we consider the Garteur HIRM (High Incidence Research Model) benchmark [10]. This was produced in the early nineties as a challenge problem in robust control: the design problem was to produce a control augmentation system for HIRM, an aircraft typical of modern combat aircraft. In general terms the aim of the flight control system is to give good handling qualities across the specified flight envelope and also provide robustness to unmodelled plant dynamics, modelling uncertainties and variations in operating point within the flight envelope. Acceptable noise and disturbance rejection must also be demonstrated.

In general simple approximate models are chosen so as to enhance understanding of the principles of the design. The aircraft definition provides a continuous model of the aircraft dynamics in terms of the flight conditions (which describe things such as aeroplane geometry, altitude, mass number etc), and the response of the aircraft to control inputs (in terms of displacement, velocity and acceleration). Models are also needed of atmospheric conditions and so forth. Thus for example HIRM is described in terms of 11 inputs (relating to taileron, canard and rudder deflections, throttle and wind), 16 states (velocities, roll pitch and yaw rates and angles, centre of gravity[1] etc) and 20 outputs.

Initial design of the flight control laws is carried out against this reference model, to meet the high level design requirements. Pratt [26] provides a thorough account of many of these for modern fighter aircraft. We give just two examples to indicate some of the subtleties of design of modern fighter aircraft:

- Good handling qualities are those that offer precise control in the various modes of operation of the aircraft, with low pilot workload. While the former can be estimated from the model or judged by experiment, for the latter evaluation by pilots is typically used, and then correlated with parameters in aircraft response which the pilot uses in performing the task. Thus for example the phugoid is a low-frequency mode which causes oscillations in pitch and speed: this can be controlled by pilots but requires attention, so poor phugoid characteristics make for poor handling. Pilot induced oscillations are unwanted oscillations resulting from the pilots attempt to control the aircraft, and require subtle analysis particularly of unexpected behaviours of fly-by-wire systems.

[1] The centre of gravity may change as the plane burns fuel and "releases items"

– **aeroservoelasticity** or structural coupling is the name given to the interaction between the flight control system of an aircraft and the oscillations of the airframe. The sensors of the flight control system detect not only the motion of the aircraft, which provides the required feedback to the control system, but also high frequency oscillations due to resonances of the airframe, which can feed into the control loop and cause instability. To solve this problem notch filters are introduced to attenuate the resonances. However these can also add a phase lag, which itself interferes with stability. Correcting the phase lag with a phase advance filter introduces further coupling at high frequencies. Pratt [26] provides a detailed survey of how this design conflict was resolved for the Eurofighter, and the techniques used in the evaluation criteria.

Notice that both these problems are made more challenging by the use of more complex automated control systems, lighter more flexible materials in the airframe, and the need to take account of a variety of payloads.

For the HIRM model the design requirements comprise:

– Control strategy
– Pilot commands
– Robustness considerations for the design envelope, modelling, measurement and hardware implementation
– Robustness requirements
– Performance requirements
– Scheduling considerations

and each requirement comes with a corresponding evaluation criterion. The evaluation criteria comprise several hundred numeric tests in the form of response to specified inputs, generally expressed graphically. We expand below on one such test, the Nichols plot, which occurs in about half the criteria.

3 Diagrams as evaluation criteria

The Laplace transform provides an algebraic representation of a linear dynamical system as a transfer function $G(s)$, that is a quotient of two polynomials in a complex variable s. Under this representation the input $X(s)$ to $G(s)$ is transformed to $Y(s) = G(s)X(s)$. Transfer functions provide the traditional block diagram representation used in Simulink. Thus for example the HIRM rudder actuator has transfer function

$$\frac{1}{1 + 0.0191401s + 0.000192367s^2}$$

.

The behaviour under particular inputs is often represented graphically. For simplicity we describe only the "frequency response" analysis of such a system via Nichols plots. If a sine wave is input to a stable system it can be shown that the

steady state output will also be a sine wave, of the same frequency ω as the input, but with different phase and amplitude. The difference between the output and input phases is called the "phase margin", $arg(G(I\omega)) = arg(Y(I\omega)/X(I\omega))$, and ratio of the output to the input amplitudes is called the "gain margin", that is $|G(I\omega)| = |Y(I\omega)|/|X(I\omega)|$. Here I denotes the square root of -1.

Thus a simple evaluation criterion for a design requirement might take the form

- Phase margin greater than 40 degrees
- Gain margin of G greater than 20 dB

The Nichols plot of G allows us to express more complex design requirements. It plots the gain against the phase in decibels for different values of the input frequency ω, and is given parametrically by

$$x = arg(G(I\omega t)) \tag{1}$$

$$y = 20ln(|G(I\omega t)|)/ln(10). \tag{2}$$

Thus one design requirement for the HIRM benchmark is a lengthy description

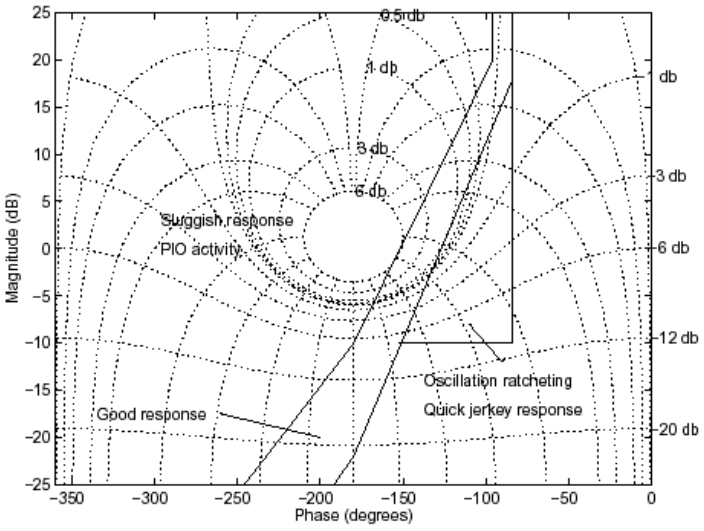

Fig. 1. Handling requirements for HIRM model

of "acceptable handling qualities" that balance aircraft performance against pilot discomfort, and avoid pilot induced oscillations. The evaluation criteria are that a sequence of Nichols plots all lie in the region marked "good response" in Fig. 1. Other evaluation criteria used in HIRM require that various Nichols plots avoid a hexagonal exclusion region around $(-\pi, 0)$.

Other plots used in evaluation criteria include Bode and Nyquist plots in the frequency domain, and analysis of the response to ramp and step inputs in the time domain. Matlab [22] commands such as `nichols, bode` draw the respective plots of a given transfer function, and for small examples GUI tools allow the user to manipulate the plot to a required form, and obtain a modified form of the input that generates this output.

4 Computing with real numbers

The methods above require us to discharge verification requirements which take the form of showing that a particular curve in $\mathbf{R} \times \mathbf{R}$ lies in a particular region bounded by straight lines, or more generally that a function is positive or continuous in an interval. We briefly summarise the relevant methods from numeric and symbolic computation, and computational logic.

Numerical methods Numerical methods are the standard, and almost universal, approach to computational support for analysis of control and dynamical systems, and in particular the discharge of such requirements.

These are widely available through standard commercial libraries such as NAG and MatLab which generate numeric or graphical output, from which various properties of the system may be inferred. In addition such systems can readily accommodate other inputs, for example from measurement devices, or other numerical procedures, such as curve fitting. For many problems, for example the investigation of chaotic phenomena, there are no alternative standard techniques. The main advantage of numerical systems is that they will always give an answer, and with sufficient user expertise are accepted as doing so sufficiently quickly and accurately, with established protocols for testing and error analysis. However the output, and properties derived from it, will be usealways be numeric and not analytic, and support for investigating other properties of the solution, for example continuity or behaviour under parameters, may be limited. Thus verifying a Nichols plot for a parameterised curve will involve drawing a series of plots for different values of the parameter.

Symbolic computation Symbolic computation systems, such as Maple, contain a variety of symbolic algorithms that might in principle be used to discharge our verification requirements.

We note that we cannot hope for a fully automatic test for an arbitrary function q to be positive in an interval $[a, b]$, as this is a version of the zero-constant problem and hence undecidable, even for q an expression generated by a variable x, the *sin*, *exp* and *modulus* functions [28]. There is much theoretical work on techniques for analysing questions of this kind for particular q, for example quantifier elimination, which is decidable for polynomial functions q, but can still be doubly exponential, and more effective methods such as cylindrical algebraic decomposition, CAD, which still become infeasible for complicated

inequalities. Jirstrand [18] experimented with CAD for polynomial differential equations with constraints, and solved some problems involving stationarity, stability, and curve following, but was only able to tackle small examples.

Computer algebra systems test for analytic properties such as continuity, convergence or differentiability by using numeric or symbolic root-finding algorithms to find possible points of failure of the required property, which again reduces to the zero constant problem. While in principle parameters can be handled, computer algebra systems do not generally handle pre and side conditions well [12]. Consider the equation

$$y'(x) + y/(2\sqrt{x-a}) = \ln(b-x)\exp(-\sqrt{x-a}) \tag{3}$$

for which Maple returns

$$y(x) = \exp(-\sqrt{x-a})((x-b)(\ln(b-x)-1)+C) \tag{4}$$

This supposed solution is only defined if $x > a$ and $x < b$, so that if $b < a$ it is not defined anywhere on the real line. Maple has produced this solution because the `dsolve` procedure, for finding a solution of an equation valid in an interval V, does does not check what the interval V might be, or the required conditions that functions are continuous on it.

Computational logic and real number theorem proving Computational logic means the use of a computer to produce a proof in some formal system. A variety of systems have been built, some motivated by experimentation in a particular formal system, others by the need for increasingly complex practical verification, especially hardware and distributed systems. We mention below other applications of computational logic to verification of control systems. Such systems may be hard to use, but the pay-off is that the user can be confident that the results that they produce are correct. SRI's PVS [27] - prototype verification system - is based on sequent calculus and provides a collection of powerful primitive inference mechanisms including propositional and quantifier rules, induction, rewriting, and decision procedures for linear arithmetic. The implementations of these mechanisms are optimised for large proofs: there is support for proof strategies and a powerful brute force search mechanism called grind. PVS has a rich higher-order type system supporting overloading of operators, subtypes and dependent types, and mechanisms for parametric specifications. It has been widely used in applications, particularly aerospace work.

As we saw in Section 1 computational logic has impacted control engineering: here we concentrate on those aspects useful in the discharge of design requirements.

Some form of real analysis has been implemented in many theorem provers, both because it forms the basis of much mathematics and, increasingly, because of its use in hardware verification: for example Harrison's work on floating point verification [15]. PVS has a basic built-in theory of the reals: the axiomatisation is via the least upper bound principle, every non-empty set of reals bounded

above has a least upper bound. Dutertre [8] extended this and developed a library for real analysis in PVS, including definitions and basic properties of convergence, limits, continuity and differentiation. Further development of real analysis in PVS is described in [13], where the transcendental functions were incorporated by defining the functions in terms of power series and constructing on top of the basic definitions a large lemma database of routine results about elementary functions, that is functions built up from rational functions of a variable x together with *cos*, *sin*, *exp* and *log*. Typical entries are:

$$\sin(\frac{\pi}{2}) = 1 \qquad \cos(\pi) = -1$$
$$\sin(x) = \cos(\frac{\pi}{2} - x) \quad \frac{d}{dx}(\sin(x)) = \cos(x)$$

Particularly useful for our work are procedures that attempt to check if a function has a property in a closed interval [13]: typical properties are continuity, convergence and differentiability. The checker relies on results which follow the text-book development of continuity (sums of continuous functions are continuous and so on), augmented with our database and a collection of standard results about the continuity of elementary functions.

The method used for this checking is what one might call the High School method. It is based on the theorems that the constant functions and the identity function are continuous everywhere, and that well-founded combinations using the following operators are also continuous: addition, subtraction, multiplication, division, absolute value and function composition. Also, the functions exp, cos, sin and tan are continuous everywhere in their domains. The checker can be used to check the continuity and limiting behaviour of functions such as

$$e^{x^2+|1-x|} \text{ or } -\pi - 1 + \pi * e^{1-\cos(x)}.$$

As continuity is undecidable [4], this will not always work, however for many examples it is sufficient. Note also that if a proof fails using the checker we can always go back and prove the result from first principles and add it to the database. The evolving nature of the database makes it hard to provide efficiency measures. At first sight it might appear that using a computational logic engine for such symbolic analysis gives one no advantage over a computer algebra system, and the considerable disadvantage that it is very much harder to use. However as we have seen computer algebra systems have a number of disadvantages, and computational logic engines like PVS have the advantage that the results they produce are correct and unambiguous. Our methods may fail if the lemma database does not contain an appropriate result: however if they report success this will always be justified by a proof, rather than, as in a computer algebra system, failure of a root finding algorithm or similar.

Maple-PVS Our Maple-PVS system [12] combines symbolic computation and computational logic by providing restricted invocation of PVS from Maple to perform the checks we describe above. These are called from Maple by a simple

pipe-lined interface. As well as direct calls from Maple to verify continuity and so on, we are able to build these checks into other Maple procedures. For example, it is straightforward to write a harness for a Maple routine to solve differential equations which checks the validity of the pre-conditions for the routine to be correct.

5 Symbolic diagram testing with Maple-PVS

We described above how classical control engineering gives rise to design requirements expressed geometrically, typically in the form of plots such as the Nichols plots. In this section we show how to discharge these requirements automatically in some cases using symbolic computation and computational logic embedded in our Maple-PVS system.

Our approach is based in the following simple idea. Tests based on such plots concern showing that a given curve $y = f(x)$ is bounded above (or below) by a given line $y = g(x)$ in an interval $[a, b]$, or equivalently that

$$\forall x \in [a, b], g(x) - f(x) > 0. \tag{5}$$

The following two theorems each provide a sufficient condition for the inequality to hold which is readily verified symbolically for certain lines and curves.

Theorem 1. *Given real-valued functions f, g defined throughout an interval $[a, b]$ and satisfying*

1. *g is linear in $[a, b]$*
2. *$f(x)$ is continuous and twice differentiable in $[a, b]$*
3. *$f(x)$ is monotonic increasing and concave in $[a, b]$, ie $f^{'}(x) > 0$ and $f^{'}(x)$ is monotonically decreasing in $[a, b]$, ie $f^{'}(x) > 0$ and $f^{''}(x) < 0$ in $[a, b]$.*
4. *$g(b) > f(b)$*
5. *$g^{'}(b) < f^{'}(b)$*

then $f(x) < g(x)$ throughout the region $[a, b]$.

The proof of this is straightforward. Let $g(x) = px + q$. The continuous function $g(x) - f(x)$ attains its maximum and minimum values in an interval at endpoints or at zeroes of $p - f^{'}(x)$. Now for any x in $[a, b]$ we have $p - f^{'}(x) < p - f^{'}(b) = g^{'}(b) - f^{'}(b) < 0$, so the minimum value of $g(x) - f(x)$ is either $g(b) - f(b) > 0$, or $g(a) - f(a) \geq g(b) - f(b) + (f'(b) - p)(b - a) > 0$.

Theorem 2. *Given real-valued functions f, g defined throughout an interval $[a, b]$ and satisfying*

1. *g is linear in $[a, b]$*
2. *$f(x)$ is continuous and twice differentiable in $[a, b]$*
3. *$f(x)$ is monotonic increasing and convex in $[a, b]$, ie $f^{'}(x) > 0$ and $f^{'}(x)$ is monotonically increasing in $[a, b]$, ie $f^{'}(x) > 0$ and $f^{''}(x) > 0$ in $[a, b]$.*
4. *$g(b) > f(b)$*

5. $g(a) > f(a)$

then $f(x) < g(x)$ *throughout the region* $[a, b]$.

Similar results hold for $f(x)$ monotonic decreasing, and with a and b interchanged.

Theorem 1 ensures the correctness of the following test for condition (5):

1. compute the verification conditions (1)-(5) above
2. verify that conditions (1)-(5) hold

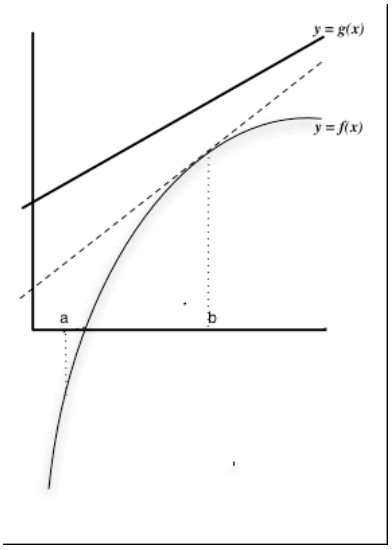

Fig. 2. Conditions of Theorem 1

For any $g(x)$ linear and $f(x)$ continuous and twice differentiable in $[a, b]$, we can partition $[a, b]$ into intervals determined by the zeroes and points of inflection of $g(x) - f(x)$ and $g^{'}(x) - f^{'}(x)$, and hence apply one or other of these theorems in each interval to determine whether f is bounded above by g throughout.

In the case of the Nichols plot of a transfer function $G(s)$, which is a rational function (ie a quotient of two polynomials), the curve $y = f(x)$ will be given parametrically by

$$x = arg(G(I\omega t)) \tag{6}$$

$$y = 20ln(|G(I\omega t)|)/ln(10) \tag{7}$$

and its derivatives $f^{'}(x), f^{''}(x)$ will be quotients of polynomials in cos, sin and x.

We carried out a prototype implementation of the test using our Maple-PVS system described above. Maple readily computes the verification conditions, which require computing and simplifying derivatives. PVS is used to discharge them, using its tests for continuity and inequality of elementary functions.

Example

As an example consider the transfer function

$$G(s) = k.(s^2 + 2cs + d)^{-1} \tag{8}$$

and Nichols plot in the region $[-\pi, -\pi/2]$. This is the graph of $|G(Iw)|$ against $20log_{10}|tan^{-1}(Im(G(Iw))/Re(G(Iw)))|$, given parametrically by

$$x := \arg(k.(-t^2 + 2Ict + d)^{-1}) \tag{9}$$

$$y := 20ln\,|\,k.(-t^2 + 2Ict + d)^{-1}\,|\,/ln(10). \tag{10}$$

Then, as an example, Condition (3) requires us to calculate dy/dx and show positive in $[-\pi, -\pi/2]$. Maple shows

$$dy/dx = 20\ sin(x)(-c\ Q + c^2cos(x) + d^2cos(x))/Q/(c\ cos(x) + Q)/ln(10) \tag{11}$$

where

$$Q = (c^2cos(x)^2 + d^2sin(x)^2)^{1/2}. \tag{12}$$

In this case the PVS test that dy/dx is positive in the desired region follows the usual informal human reasoning - parse the expression into a quotient of sums of products, use the lemma database to identify the sign of each component - for example *sin* and *cos* are both negative in the given interval - and hence deduce the sign of the products, then the sums, then the quotient.

It is hard to provide a complexity or timing analysis of our method, since it is built on top of an evolving lemma database. For the kind of applications we consider the inputs are not random, but typically consist of a large number of tests in rather similar format - for example Nichols plots are usual computed over the interval $[-\pi, -\pi/2]$, and in the HIRM model nearly all satisfy the conditions of Theorem 1. Hence it is worthwhile optimising the system by extending the lemma database as required. Our method is related to what is called, in computer graphics, the method of Lipschitz bounds, devised by Kalra and Barr [19] to handle ray-tracing as a technique for visualising implicit surfaces.

6 Analysis

We are aiming to find automated symbolic methods of design testing, to replace the use of numerical graph plots which are verified by inspection.

We prototyped our method for Theorem 1 and another similar test, and performed the necessary computations in Maple and PVS. Maple can readily provide the support that we need to compute derivatives, simplify and so on.

PVS contains definitions and properties of elementary functions and likewise provides much of what we need, and the necessary material is under constant development, particularly in support of NASA's verification work on air traffic control algorithms.

There are a number of other symbolic computation and theorem proving technologies which could be used to implement this approach to design testing - in particular it would be fairly straightforward to eliminate the dependency on Maple altogether and use a rewrite engine to compute and simplify derivatives. Longer term there is no obstacle in principle to embedding the whole process as an automated call from other engines, like Simulink.

Note that unlike numeric tests our methods are not samples, but provide verification, subject to the correctness of the underlying Maple-PVS implementation, for all values in the interval. We hope to be able to use similar methods for robust control, where we will need to consider parameterised curves, and to extend to non-linear systems. Current techniques like Bode and Nichols plots are a heritage of the pre-computer days when they really were plotted on graph paper. However the availability of methods like ours encourages speculation about other design verification methods which while impossible to plot and eyeball may be susceptible to symbolic verification.

Acknowledgements

This research supported by QinetiQ, EPSRC grants GR/M98340 and GR/L48256, and by an EPSRC studentship to the third author. We are indebted to Manuela Bujorianu, John Hall, Rick Hyde, and Yoge Patel for their insights into control engineering, and to Rob Arthan and Colin O'Halloran for helpful discussions.

References

1. M Arbib and E Manes Machines in a category SIAM review 57 (1974), 163-192
2. R. Arthan, P. Caseley, C. O'Halloran, and A. Smith. ClawZ: Control laws in Z. In *Proc. 3rd IEEE International Conference on Formal Engineering Methods (ICFEM 2000)*, York, September 2000.
3. Richard J. Boulton, Ruth Hardy and Ursula Martin A Hoare Logic for Single-Input Single-Output Continuous-Time Control Systems In *Proceedings 6th International Workshop on Hybrid Systems, Computation and Control*, pages 113-125, Springer LNCS vol 2623, 2003
4. G Cherlin Rings of continuous functions: decision problems, In *Model theory of algebra and arithmetic*, Lecture Notes in Mathematics 834, pages 44–91, Springer 1980
5. D Dill A theory of timed automata Theoretical Computer Science , 126, pages 183-235,1994
6. Martin Dunstan, Tom Kelsey, Ursula Martin and Steve Linton Lightweight formal methods for computer algebra systems In *ISSAC'98: Proc ACM International Symposium on Symbolic and Algebraic Computation*, Rostock, ACM Press, 1998

7. Ursula Martin, Martin Dunstan, Tom Kelsey and Steve Linton Formal methods for extensions to computer algebra systems In *Proc FM'99: World Congress on Formal Methods in the design of computing systems* Lecture Notes in Computer Science 1709, pages 1758-1777, Springer Verlag, 1999
8. B. Dutertre Elements of Mathematical Analysis in PVS, In *Theorem Proving in Higher Order Logics: 9th International Conference, TPHOLs '96*,Lecture Notes in Computer Science 1125, pages 141-156, Springer-Verlag 1996
9. A Edalat and A Lieutier Domain theory and differential calculus Proc IEEE LICS 17, IEEE Press 2002
10. Robust Flight Control Design Challenge Problem Formulation and Manual: the High Incidence Research Model (HIRM) Garteur - Group for aeronautical research and technology in Europe Technical report, GARTEUR/TP-088-4, 1997
11. M. J. C. Gordon. Mechanizing programming logics in higher order logic. In G. Birtwistle and P. A. Subrahmanyam, editors, *Current Trends in Hardware Verification and Automated Theorem Proving*, pages 387–439. Springer-Verlag, 1989.
12. Hanne Gottliebsen, Tom Kelsey and Ursula Martin Hidden verification for computer algebra systems To appear, Journal of Symbolic Computation 2004
13. Hanne Gottliebsen Transcendental Functions and Continuity Checking in PVS In *Theorem Proving in Higher Order Logics: 13th International Conference, TPHOLs 2000* Lecture Notes in Computer Science 1869, pages 198-215, Springer-Verlag 2000
14. C. Gurr and K. Tourlas. Towards the principled design of software engineering diagrams. In *Proc. 22nd International Conference on Software Engineering*, pages 509–520, ACM Press, 2000.
15. J Harrison Theorem proving in the real numbers Cambridge University Press, 1995
16. M. Hasegawa Models of Sharing Graphs Springer 1997
17. C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 583, 1969.
18. Mats Jirstrand Nonlinear control system design by quantifier elimination, J. Symbolic Comput, 24 (1997), pp 137–152
19. D. Kalra and A.H. Barr Guaranteed Ray Intersections with Implicit Surfaces Computer Graphics (SIGGRAPH '89 Proceedings), Vol. 23(3), 1989, pages 297-306.
20. Bruce Krogh Approximating Hybrid System Dynamics for Analysis and Control HSCC 1999, LNCS 1569, Springer, 1999
21. The MathWorks. Simulink. `http://www.mathworks.com/products/simulink/`
22. The MathWorks. Matlab. `http://www.mathworks.com/products/matlab/`
23. B. Mahony. The DOVE approach to the design of complex dynamic processes. In *Proc. of the First International Workshop on Formalising Continuous Mathematics*, NASA conference publication NASA/CP-2002-211736, pages 167–187, 2002.
24. T. Nipkow. Hoare Logics in Isabelle/HOL. In *Proof and System-Reliability*, pages 341–367, Kluwer, 2002.
25. K. Ogata. *Modern Control Engineering*. Prentice-Hall, third edition, 1997.
26. R. W. Pratt, editor. *Flight Control Systems: Practical Issues in Design and Implementation*, volume 57 of *IEE Control Engineering Series*. The Institution of Electrical Engineers, 2000.
27. S. Owre, J. Rushby, and N. Shankar PVS: a prototype verification system In *11th Conf. on Automated Deduction*, volume 607 of Lecture Notes in Computer Science, pages 748–752. Springer Verlag, 1992.
28. D Richardson Some Unsolvable Problems Involving Elementary Functions of a Real Variable J. Symbolic Logic 33, 514-520, 1968.

29. A. Tiwari and G. Khanna. Series of abstractions for hybrid automata. In *Proc. 5th International Workshop on Hybrid Systems: Computation and Control (HSCC 2002)*, volume 2289 of Lecture Notes in Computer Science, Springer, 2002.