Robust Control Tools: Graphical User-Interfaces and LMI Algorithms[*]

Stephen Boyd[†]

Information Systems Laboratory
Electrical Engineering Department
Stanford University
Stanford CA 94305
Internet: `boyd@isl.stanford.edu`

April 23, 1994

Abstract

Robust control theory considers a fundamental and practically important issue in control engineering: *plant uncertainty.* It turns out that many of the simplest questions are very difficult to solve, but researchers have made considerable progress over the last twenty years. Nevertheless the theory has so far had less impact on practice than one might imagine. (Simulation and modeling tools, for example, have probably had a wider impact on practice.) Recent techniques of robust control theory, based on convex optimization over linear matrix

[*]Systems, Control and Information, volume 38, number 3, pages 111-117, March 1994. Special issue on Numerical Approaches in Control Theory, edited by S. Eiho, H. Kimura, A. Ohara, and T. Mori.

[†]Research supported in part by AFOSR (under F49620-92-J-0013-P00001) and NSF (under ECS-9222391)

inequalities (LMIs), coupled with new user-interfaces, might change this.

This note is not meant to report any serious scholarly work; it just a collection of musings on the current state and future of tools for robust control. Because of space limitations we give only a very few references; a complete set of references can be found in the monograph [2].

The outline is as follows. In §1 we give a high-level discussion of robust control theory. In §2 we develop an "artist's sketch" of a user-interface for a robust control tool we might hope to develop, perhaps someday soon. In §2 we do not consider the algorithms that do the underlying computations, which is the main intellectual challenge, and hence the entire focus of research, in robust control theory. From the point of view of robust control *theory*, the discussion of §2 is vacuous.

In §3 we turn to the important question of the underlying algorithms for such a tool. Here we simply make some general comments about some promising methods, especially, analysis and design algorithms based on linear matrix inequalities. These methods can be thought of as extensions of the work of Yakubovich and others in the 1960s, or as extensions of more recently developed methods such as Doyle's μ-synthesis or Skelton's covariance control.

1 Robust Control Theory

One of the main goals of feedback control is to maintain overall system performance despite changes in the plant. We now call this property *robustness*—but the idea has been around since the origins of control systems.

Control system robustness received less attention in the 1960s during the development of state-space optimal control and estimation theory (e.g., LQR and LQG) in which the idea of plant change, variation, or uncertainty played at best a secondary role. Some initial results showed that the state-feedback implementation of LQR was very tolerant of changes in the plant [15]. This led to the hope that controllers designed to be optimal for a fixed, known plant might *automatically* turn out to be robust, i.e., tolerant to changes in the plant. In a short note, Doyle pointed out that this is not the case [5]. In general, robustness does not come for free from a controller designed via optimal control and estimation theory.

Partly in reaction to the secondary role played by robustness in optimal control and estimation theory, the field called *robust control theory* was developed over the last fifteen years or so. In robust control theory, plant variation plays a central role. Some of the key questions in robust control theory are:

- *Characterizing plant variation.* What is a good way to describe plant variations or uncertainty? Some descriptions attempt to faithfully describe the variations that might be ecountered (e.g., probability distributions on physical parameters). Other descriptions are more convenient for the associated theory (e.g., bounds on singular values of transfer matrix errors).

- *Robustness analysis.* Given a controller and plant, and some description of the plant uncertainty, how can we determine such things as "typical" or "worst-case" performance? How can we predict the performance degradation caused by variation in the plant? How can we verify that some performance specifications are met for some set of plants?

- *Robust controller synthesis.* Given a plant and some description of the plant uncertainty, how can we design a controller that optimizes "typical" or "worst-case" performance? How can we design a controller that meets some performance specifications for some set of plants (e.g., all or typical)?

It's important to remember several things: First, these questions were asked, and partial answers obtained, before the term "robust control" was coined. Second, the qualifier "robust" shouldn't be necessary since a *well-designed* controller must be able to tolerate the plant changes or variations that can be expected. To put it another way, a controller that cannot tolerate variations in the plant that will be encountered in operation is simply a *poorly-designed* controller, not just a non-robust controller. (But the extra qualifier "robust" has helped re-focus attention on this important aspect of control engineering.) Finally, we mention that in the former Soviet Union a similar field developed, called "guaranteed control" in English translations.

Over the last fifteen years thousands of papers and many books have been written on the topic of robust control. Researchers have made very substantial progress, and have obtained many partial answers to the questions posed above. But with a few notable exceptions, it seems that the problems posed above are very difficult to solve. Some very simple problems of robustness analysis have been shown to be NP-hard, hence as difficult as some other famous problems for which no efficient solutions are known to exist, or are likely to be found. One example is determining stability of a linear system in which several parameters vary over given ranges [3, 4].

One of the most famous contributions of robust control theory is the development of ("central") $\mathbf{H}_\infty$ controller synthesis (see, e.g., [7]). This synthesis procedure is no more complicated than LQG synthesis, but yields controllers that *can* be more robust to plant variations than LQG controllers.

The type of plant variations such $\mathbf{H}_\infty$ controllers can tolerate has a very specialized form, and not one that can be said to accurately represent variations that might arise in actual plants. This is analogous to LQG: not many real disturbances are accurately characterized as Gaussian processes, and only in rare circumstances are quadratic cost functions truly appropriate. In both cases we have an analytic solution to a problem that is not really the problem we'd like to solve, but is similar enough to make the method useful in practice (with a lot of trial-and-error tuning of weights). So although $\mathbf{H}_\infty$ synthesis can be a useful tool for synthesizing robust controllers, it does not *directly* solve the robust controller synthesis problems that really arise.

Let's consider a simple sketch of a real robust synthesis problem, without any details.

- *The plant and its variation.* The plant is a mechanical system, perhaps a platform that we wish to isolate from base motion and vibration. Certain physical parameters (e.g., load mass, stiffness, internal damping) can vary in given ranges (e.g., $\pm 20\%$). The actuators have substantial nonlinearities, which are fairly well modeled by lookup tables that were determined experimentally. Measurements of various transfer functions for this system are not repeatable at high frequencies; but from these experiments we have some ideas of the transfer function variations that we may encounter.

- *Design goals.* Our goal is to design a controller that achieves some performance specifications despite these variations. The specifications might be that with certain disturbances present (base motion, sensor noises, etc.) the actuator signals and other critical signals do not exceed given limits, and the velocity at the top of the platform is small.

Note how down-to-earth this problem statement is. It can be readily explained to and appreciated by any engineer, and especially, one who is not an expert in robust control theory!

In contrast, consider the problem statement for an $\mathbf{H}_\infty$-synthesis problem. The statement involves singular values, $\mathbf{H}_\infty$ norms, and so on. It cannot be understood even by an experienced control engineer (unless they are up to date on the special language and concepts of robust control theory). Comparing this problem statement with the one sketched above reveals a gap between what we would really like to solve and what we can solve using $\mathbf{H}_\infty$ sunthesis. (To be fair, we should point out that an engineer using $\mathbf{H}_\infty$-synthesis could probably synthesize a pretty good controller, with much trial-and-error, for the real problem described above.) Teaching new control engineers about singular values and $\mathbf{H}_\infty$ norms (which I do) is *not* the whole solution to closing this gap. The point is that despite the thousands of papers and much real progress, robust control theory has not yet provided a *straightforward* or *direct* solution to simple and realistic problems of robust control.

A notable development that addresses this issue is μ-analysis and synthesis, pioneered by Doyle, Safonov, and others [16, 6]. They recognized that describing plant variations in terms of singular values is rarely realistic, and just a case of forcing a practical problem to fit into the hypothesis of a theorem (the Small Gain theorem in this case). Instead they describe uncertainties in much more realistic ways: they consider systems in which various internal transfer functions can vary by frequency-dependent amounts. The plant might consist of various subsystems, each of which has its own description of uncertainty. Extensions of the method include parameter variation as well. These methods can attack the problem described above provided we choose to ignore the nonlinearity and measure performance in certain ways. One disadvantage is that the designer must learn the special vocabulary and concepts associated with the method.

Most likely an engineer facing the problem sketched above would not use the methods of robust control theory to synthesize a controller, or analyze its robustness. To analyze robustness the engineer would probably run many simulations of the closed-loop system under various scenarios, i.e., with many different disturbance patterns, various values of the physical parameters, and even different high frequency dynamics. For each run, he or she checks that the actuator limits are respected, and the platform velocity stays within the desired limits. Worst-case and typical values are noted. Of course, this is nothing more than *responsible* simulation. (*Irresponsible* simulation is when the engineer runs one or just a handful of simulations, without varying the physical parameters, etc.) In summary, an engineer facing the problem sketched above would probably use simulation and Monte Carlo methods for robustness analysis, and not the methods of robust control theory (despite the thousands of papers that have appeared . . .).

Why is this? There are a few related reasons: First, simulation tools can be used for a very wide variety of systems. In contrast, the results of robust control theory (so far) apply to fairly restricted sets of systems and answer fairly restricted sets of questions. Moreover, simulation tools have been made convenient to use; the *user* of a simulation tool does not have to be an expert on numerical integration to use the tool. In contrast, the user of a current generation robust control tool has to know quite alot about robust control theory.

2 User-Interface for a Robust Control Tool

It is useful to imagine how a robust control tool should look to the user, even if we cannot yet build such a tool. I imagine it as an extension of current system simulation tools like SystemBuild and Simulink. A graphical interface allows the user to describe the system via block diagram, by manipulating boxes (icons) and the connections between them. The icons contain subsystems such as linear dynamics, nonlinearities (saturator, deadzone, hysteresis, lookup table), and user-supplied subroutine call. These icons are found in current system simulation tools, but in an extension that handles robustness analysis, we might find some additional icons:

- *Unknown or uncertain parameter.* The user selects the range, or perhaps a nominal value and percent variation, for a parameter. This is used to represent unknown parameters, e.g., a physical parameter that varies $\pm 20\%$. The use might optionally specify a distribution for the parameter value. Another option might allow the user to "link" or "bind" several uncertain parameters together, which constrains them to have the same value. This is useful when a single parameter enters the model in many places (e.g., Young's modulus, aerodynamic pressure).

- *Unknown transfer function.* This is used to model error in transfer functions. The user might specify, in some graphical way, the uncertainty at each frequency.

- *Unknown or uncertain (sector) nonlinearity.* This is used to represent a nonlinearity whose graph is known to lie between specified lower and upper bounds. The user specifies the lower and upper bounds.

- *One of many (enumerated possibilities).* The icon represents one out of a given list of possibilities, i.e., the possible values are explicitly enumerated. As an example, suppose we have 100 transfer function *measurements* of some subsystem, e.g., an actuator; in the block diagram we represent the actuator as one of the 100 measured transfer functions. As another example we may list 10 typical values for some parameter. The same mechanism could be used to describe a list of typical diturbance or command signals. The user might optionally assign probabilites to each possibility.

The user describes the system variations or uncertainties by incorporating these icons into the block diagram. It is not hard to imagine a block diagram representation, including plant uncertainties, for the problem sketched above. Such an interface would allow any engineer to build up a model of a system that includes uncertainties, even if he or she does not know about robust control theory (i.e., its special language, recent results, techniques).

In a similar way, the user might even describe *performance specifications* via block diagrams. This could be accomplished with some new icons:

- *Performance meter.* A performance meter is used to measure the size of a signal. The user might select RMS, peak, or other norms; time or frequency-domain weightings, and so on.

  The user might specify a maximum (or goal) level, and some level of "hardness" varying between a constraint that must be satisfied and an objective that is to be minimized (see e.g., [8]).

  Performance meters could also support some sort of *alarm* mechanism that would stop a simulation or design, or raise some kind of alarm, if the signal exceeds some limits. It might also have general data-logging capabilities.

- *Noise, disturbance, or command source.* This icon is used to generate a signal that is unpredictable or uncertain. The user selects the model of uncertainty, e.g., stochastic, unknown-but-bounded, or one-of-many. In addition the user sets associated parameters, e.g., amplitude, bandwidth, slew rate. Similar icons are already used in simulation tools.

Once again it is not hard to imagine a block diagram that corresponds to the specifications loosely described in the example sketched above.

This robust control analysis tool would support several modes beyond simulation, e.g.:

- Monte Carlo simulation,
- approximate worst-case simulation, and
- synthesis of guaranteed bounds.

In *Monte Carlo simulation*, multiple simulations are performed with random selections from the uncertain parameters, transfer functions, nonlinearities, and signal sources. Important statistics could be compiled and reported. Simulation plots could show the nominal response plot along with error bars that summarize the response statistics point by point. Small error bars show that the effect of the uncertainties is small, i.e., the system is robust.

In *approximate worst-case simulation*, an attempt is made to find the worst response among the possible values of parameters, transfer functions, nonlinearities, and so on. Simulation plots might show nominal responses along with error bars showing extreme values found. This analysis mode is based on some type of *local* optimization, so the results come with no *guarantee* that the worst-case values and objectives found are in fact the worst-case. (Thus, the qualifier, "approximate".) In many cases, of course, the worst-case values found will be correct, i.e., the (global) worst-case values. In the language of robust control theory, approximate worst-case simulation yields *lower bounds* on performance degradation due to uncertainty.

In the mode *synthesis of guranteed bounds,* the tool searches for *upper bounds* on the worst-case performance degradation. In this case the tool can produce a "performance guarantee certificate" that *proves* some performance specification is met for all possible values of the uncertainties. (The exact form of the certificate depends on the types of plant variations and performance specifications involved. It might be a Popov-type Lyapunov function, a set of multipliers, etc.)

Such provable bounds would be displayed graphically. The important part is that bounds are reported back to the user in a simple and natural format, e.g., as the report that the tool has determined that the worst-case RMS platform velocity is no more than 0.3m/sec. In particular, understanding or interpreting the results should not require an extensive background in robust control theory. The user must understand the meaning of the results reported back, but *not* necessarily how the results were obtained.

Consider as an analogy the user-interface of a typical simulation tool. I can *use* a simulation tool even though I am not an expert in the "details" of simulation (e.g., proper handling of stiff systems, discrete events); the internal details are appropriately hidden (from me). A dialog box might have buttons to select *Fast, rough simulation* or *Slow, accurate simulation.* I understand what this choice means, but not exactly what happens internally (i.e., the integration algorithm and associated parameter values that correspond to each choice). A well-designed user-interface allows the advanced user to see and manipulate the inner workings of the tool, but doesn't force the ordinary user to deal with any more than is necessary.

I believe that user-interfaces in this style and spirit could be developed for robust control analysis and synthesis. The ordinary user (i.e., one lacking a background in robust control theory) could describe the system, including uncertainties, and get back useful performance bounds, but not details such as how the bounds were obtained. (The advanced user, however, could access detailed information and set parameters such as the basis to use in a "multipliers search".)

Let us return to the three modes: Monte Carlo simulation, approximate worst-case simulation, and synthesis of guaranteed bounds. It's important to understand that the three modes complement each other. Roughly speaking, Monte Carlo simulation gives us an idea of "typical" performance degradation due to plant uncertainty, whereas approximate worst-case simulation and synthesis of guaranteed bounds give us an interval in which the true worst-case performance degradation lies. If we are lucky, the interval is small, i.e., approximate worst-case simulation shows performance degradation that is not too far from our bound on worst-case degradation. But because most of the problems of robust control theory are inherently difficult (or at least, as hard as other problems that are generally considered difficult) we cannot expect the two to always be close, with reasonable computation time. On at least some problems, we expect that the bounds will not be close, or the time required to compute them will be large.

Let us finish our "artist's sketch" of the user-interface for a robust control tool by considering how controller *design* or *synthesis* might be handled. We could introduce a few new icons that represent parameters, transfer functions, or nonlinearities that are to be designed, e.g.:

- *Design parameter.* The user can change the value at any time. More importantly, the *tool* can change the value (in AutoDesign mode; see below) to optimize performance. The user could optionally set lower and upper limits for the parameter, or add a term involving the parameter to some optimization objective.

From this atomic element, we can build up more complex subsystems that are to be designed, for example, PID or fixed order controllers. As a convenience, separate icons for these might be available. The user then specifies *controller structure* graphically, by building up the appropriate block diagram. The user can specify, e.g., a decentralized linear controller, with fixed orders for each diagonal term. As another example, the user might specify a "fuzzy-logic" or "neural network" controller by forming the appropriate block diagram out of nonlinearities and weights (which are the design parameters).

The tool might support an *AutoDesign mode.* In this mode the tool will change the design parameters to effect robust synthesis. The cost function and constraints might be set graphically with the performance meters and noise sources. The synthesis could be set to optimize typical performance (via Monte Carlo simulation), approximate worst-case, or guaranteed bounds.

There are many "details" of such a user-interface for robust control analysis and design that would have to be carefully thought out. But the basic design goal is simple: it should be usable by an engineer not familiar with the language or results of robust control theory. The user must understand what the performance bounds or design objectives mean, but does not need to understand exactly how the performance bounds or synthesized controller were obtained.

3 LMI-based analysis and design

In the previous section we discussed what the user-interface for a robust control tool should look like, without any consideration of the algorithms that would do the underlying computation. In this section, we turn to the question of what algorithms might be used in such a tool. There is not much to say about Monte Carlo analysis, which is straightforward. Similarly, approximate worst-case analysis can be based on standard methods of local optimization. So we concentrate on the synthesis of guaranteed bounds.

It is clear that no *analytical result* (like central $\mathbf{H}_\infty$ synthesis) or conventional *theorem* will be able to provide sufficiently good bounds for the problems posed via block diagrams in such a tool. The problem statement is quite detailed, whereas "analytical solutions" generally correspond to simple problem statements. The algorithms must involve some type of numerical search or optimization.

In any case, the distinction between an "analytical solution" and a "numerical search solution" does not matter too much, at least for our robust control tool. What does matter is the *computational complexity* of the algorithm. As an example, an analytical result that allows us to check, say, robust stability of a system by checking some very large but finite number of parameter values, although of great theoretical interest, may not be useful in such a tool. A "numerical search procedure" for the same problem, which requires us to solve some matrix inequalities by convex programming, might have a lower computational complexity than the analytical solution, and hence be more useful as an algorithm for use in our tool.

In my opinion, the most promising candidates are algorithms based on convex optimization over linear matrix inequalities (LMIs). The basic idea is to reformulate the robust control analysis or synthesis problem in terms of LMIs, which are then solved by new interior-point algorithms. The theory is covered in the monograph[2] and the references cited there; the new interior-point algorithms for these problems are described in the monograph [11] and its references, as well as the papers [1, 18, 9] and their references. Many researchers are now actively pursuing this approach: Safonov, Packard, Doyle, Feron, Balakrishnan, Gahinet, Skelton, Ohara, Geromel, Peres, Bernussou, Gu, and others; see [2] for a fairly complete list.

There are several ways to view LMI-based design. It can be considered an extension of work done mostly in the Soviet Union by Yakubovich and colleagues in the 1960s (see e.g., [19, 20, 21, 22, 13], on quadratic and Lur'e-Postnikov type Lyapunov functions and the so-called $\mathcal{S}$-procedure. These researchers formulated problems in terms of LMIs, and then tried to obtain analytical, often frequency-domain, solutions (via the Kalman-Yakubovich-Popov theorem, for example). The new twist is the ability to directly solve LMI problems very efficiently using interior-point algorithms. LMI-based analysis can also be considered as the next step in the methods based on the "structured singular value", or earlier work by Narendra, Taylor, and others on "multipliers" for stability analysis (see, e.g., [10]).

Perhaps the earliest article that contains all the ideas of LMI-based analysis is the 1982 article [14]. In this article Pyatnitski and Skorodinsky describe a method for checking stability of a system with several sector nonlinearities, by formulating the search for a Lur'e-Postnikov Lyapunov function as a convex optimization problem which they solve with the ellipsoid algorithm. It is not hard to imagine a user-interface like the one described in §2 for their algorithm!

The overall tool might work as follows. From the user's block diagram the tool compiles a complicated-looking (but readily solved) LMI problem. Each subblock is handled appropriately—"multipliers" are associated with each uncertain parameter, Lur'e-Postnikov Lyapunov function terms are associated with each sector nonlinearity, and so on. (The very same translation from block diagram to LMI problem might also be interpreted as an LMI-based approximation of a "real stuctured-singular value problem".) Then, a specialized interior-point algorithm solves the resulting LMI problem. The results are translated back into terms the user can understand and interpret, and finally, displayed graphically.

Some specialized robust synthesis problems can be directly cast as LMI-problems and then solved. But most synthesis problems, for example those arising when the user specifies the controller structure as described at the end of §2, are not likely to be reduced to LMI problems. Many researchers have independently found a heuristic method that appears to work well in practice: alternating between *analysis,* with the design parameters fixed, and *synthesis,* with the analysis tool fixed (Lyapunov function, scalings, multipiers, etc.). Each of these steps is a problem that can be cast in terms of LMIs and hence efficiently solved. Doyle dubbed this procedure "μ-synthesis" or D–K iteration (D is the scaling transfer matrix associated with the analysis step; K is the controller). I think more in terms of Lyapunov functions, so I might call it V–K iteration, i.e., alternating between finding a good Lyapunov function (V) with the controller (K) fixed, and finding a good controller with the Lyapunov function fixed. The overall algorithm is not guaranteed to converge to the globally best controller, but is reported by Safonov, Skelton, and others to work well in practice. The V–K iteration is heuristic in the sense of not always finding the globally optimal controller, but it is *not* heuristic in its worst-case analysis: V–K iteration terminates with a controller K (which may not be the best) and a certificate that *guarantees* a certain performance level.

4 Conclusion

The discussion of §1 is not to be interpreted as criticism of robust control theory. Robust control theory has many goals, and providing algorithms for useful software tools is just one of them. It has been very successful in identifying and elucidating the key issues.

It has been observed many times that there is a gap between control theory and control practice. The control engineer who faces the problem sketched at the end of §1, and consults the (huge and growing) robust control literature, could easily end up frustrated or disappointed. He or she will be *extremely frustrated* after reading, for example, [2], which requires not only a strong background in robust control theory but a good knowledge of convex optimization as well. This will cause the gap to widen a little bit more.

My dream is that by coupling a well-designed user-interface to the latest esoteric algorithms and methods of robust control theory, we can make robust control tools nearly as convenient to use as current simulation tools. After the practicing engineer uses such a tool to solve a few practical problems (without learning the special language of robust control theory or interior-point optimization for matrix inequalities) the gap will close a little bit. And who knows, he or she may eventually become curious about the inner workings of the tool, and even read[2].

References

[1] S. Boyd and L. El Ghaoui. Method of centers for minimizing generalized

eigenvalues. *Linear Algebra and Applications, special issue on Numerical Linear Algebra Methods in Control, Signals and Systems*, 188:63–111, July 1993.

[2] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. Linear matrix inequalities in systems and control theory, 1993. Monograph in preparation.

[3] G. E. Coxson and C. L. DeMarco. Testing robust stability of general matrix polytopes is an NP-hard computation. In *Proc. Annual Allerton Conf. on Communication, Control and Computing*, Allerton House, Monticello, Illinois, 1991.

[4] G. E. Coxson and C. L. DeMarco. Computing the real structured singular value is NP-hard. Technical Report ECE-92-4, Dept. of Elec. and Comp. Eng., Univ. of Wisconsin-Madison, June 1992.

[5] J. Doyle. Guaranteed margins for LQG regulators. *IEEE Trans. Aut. Control*, AC-23(4):756–757, August 1978.

[6] J. Doyle. Analysis of feedback systems with structured uncertainties. *IEE Proc.*, 129-D(6):242–250, November 1982.

[7] J. Doyle, K. Glover, P. P. Khargonekar, and B. A. Francis. State-space solutions to standard $\mathbf{H}_2$ and $\mathbf{H}_\infty$ control problems. *IEEE Trans. Aut. Control*, 34(8):831–847, August 1989.

[8] M. Fan, L. Wang, J. Koninckx, and A. Tits. Software package for optimization-based design with user-supplied simulators. *IEEE Control Syst. Mag.*, 9(1):66–71, 1989.

[9] P. Gahinet and A. Nemirovsky. *LMI Lab: A Package for Manipulating and Solving LMIs.* INRIA, 1993.

[10] K. S. Narendra and J. H. Taylor. *Frequency domain criteria for absolute stability.* Electrical science. Academic Press, New York, 1973.

[11] Yu. Nesterov and A. Nemirovsky. *Interior point polynomial methods in convex programming: Theory and applications.* SIAM, 1993.

[12] A. Ohara, K. Souda, and N. Suda. Quadratic stabilization and robust disturbance attenuation using parametrization of stabilizing state feedback gains — convex optimization approach. Technical Report 92-02, Department of Systems Engineering, Osaka University, 1992.

[13] E. S. Pyatnitskii. New research on the absolute stability of automatic control (review). *Automation and Remote Control*, 29(6):855–881, June 1968.

[14] E. S. Pyatnitskii and V. I. Skorodinskii. Numerical methods of Lyapunov function construction and their application to the absolute stability problem. *Syst. Control Letters*, 2(2):130–135, August 1982.

[15] M. G. Safonov. *Stability and Robustness of Multivariable Feedback Systems.* MIT Press, Cambridge, 1980.

[16] M. G. Safonov. Stability margins of diagonally perturbed multivariable feedback systems. *IEE Proc.*, 129-D:251–256, 1982.

[17] M. G. Safonov and R. Y. Chiang. Real/complex K_m-synthesis without curve fitting. In C.T. Leondes, editor, *Control and Dynamic Systems*, volume 56, pages 303–324. Academic Press, New York, 1993.

[18] L. Vandenberghe and S. Boyd. Primal-dual potential reduction method for problems involving matrix inequalities. To be published in *Math. Programming*, 1993.

[19] V. A. Yakubovich. Absolute stability of nonlinear control systems in critical cases. Pt. I,II,III. *Automation and Remote Control*, pages 273–282, 655–668, 545–556, December 1963-1964.

[20] V. A. Yakubovich. Solution of certain matrix inequalities encountered in nonlinear control theory. *Soviet Math. Dokl.*, 5:652–656, 1964.

[21] V. A. Yakubovich. The method of matrix inequalities in the stability theory of nonlinear control systems, *I*, *II*, *III*. *Automation and Remote Control*, 25-26(4):905–917, 577–592, 753–763, April 1967.

[22] V. A. Yakubovich. The $\mathcal{S}$-procedure in non-linear control theory. *Vestnik Leningrad Univ. Math.*, 4:73–93, 1977. In Russian, 1971.