Proceedings of the 37th IEEE
Conference on Decision & Control
Tampa, Florida USA • December 1998

Large-scale rigorous actuator and sensor selection

Bram de Jager Marc van de Wal Ramidin Kamidi

Faculty of Mechanical Engineering
Eindhoven University of Technology
P. O. Box 513, 5600 MB Eindhoven, The Netherlands
Email: A.G.de.Jager@wfw.wtb.tue.nl

Abstract

Control system performance depends on the actuators and sensors used in the closed loop. Selection of these devices based on an exhaustive candidate-by-candidate test is a combinatorial problem. The selection problem can be simplified by not using a candidate-by-candidate approach, but then it is likely to be less rigorous and favorable combinations of actuators and sensors can be missed. Here another approach is taken. The selection is still based on a per candidate test, but using an efficient feasibility test combined with a search strategy that, under certain conditions, is polynomial in some measures of the problem size, large-scale problems can be tackled in acceptable time. Although with this approach the problem is still combinatorial, in practice the complexity behaves polynomially, the combinatorics only coming into play with a problem size that is larger than most people want to consider. An application with 28 input/output devices to choose from, making $\approx 266 \cdot 10^6$ unique combinations possible, shows the approach to be feasible. Only 270 or 658 candidate combinations were tested for feasibility, to completely determine all combinations of sensors and actuators that were guaranteed to reach a specified level of robust performance. This is almost 10^6 times less than an exhaustive search would require.

Keywords: mechanical control system, active suspension, robust control, μ-synthesis, combinatorial optimization, maximal independent set.

1. Introduction

Input/output (IO) selection is the phase in the control system design cycle where the following question is answered: Which actuators have to be used to influence the behavior of, and which sensors have to gather knowledge of the plant to be controlled? Answering this question is often not recognized as an activity that lends itself well to a systematic study, and it is therefore hard to answer systematically with the limited theory and tools available. Those tools are in general ad-hoc and non-rigorous. They sometimes lack a strong theoretical foundation, and to make them readily applicable, the selection criteria used are only loosely tied to the goals set out for the control system. In practice, the design activities related to IO selection, and also similar activities like control structure selection, are based on intuition and trial and error.

The intuitive design is founded on, what can be called, an "internal" model the control system designer has obtained during his work with the system and his related experiences gathered in the past. As all human models, this one is a biased and distorted view of reality. People working in the same field do not share the same model, their knowledge is based on different experiences in the past, and they do not always agree with each other. This is not a happy state of affairs. Therefore, a more systematic and rigorous method, and tools to implement it, are necessary.

The main effect of such a systematic IO selection procedure is probably not the resulting "optimal" combination of input/output devices, but more likely the increased knowledge of the system, acquired during the procedure. This acts like a two-sided sword.

One side is that the designer is faced with the need to rethink the goals of the control system and the lines along which these goals could be achieved. The goals need to be precisely formulated in the case of a robust performance selection requirement, *e.g.*, for an $\mathcal{H}_\infty$ criterion in the form of weighting functions and required performance level. The lines to achieve the goals are related to formulating potential control structures and actuators and sensors. In these formulations no freedom is left for vagueness.

The other side is that the results obtained may be surprising or counter intuitive. This again forces one to reassess one's internal model of the behavior of the system, and to update that model.

From those renewed insights naturally a more mature view of the problem follows, maybe leading to the conclusion that an intuitive or a standard solution was good enough. This does not mean the whole exercise was in vain: the activity at least delivered the satisfaction knowing the job was done well initially.

To improve the current state of affairs, the main goal of this paper is to provide a rigorous, theoretically well founded, control goal oriented, selection criterion, and an, in some way, efficient selection algo-

0-7803-4394-8/98 $10.00 © 1998 IEEE

rithm. See [1] for an overview of existing methods for IO selection.

Section 2 supplies theoretical background of the selection algorithm and criterion, while Section 3 displays the tool that implements the selection procedure. An application of the proposed method to a large-scale tractor-semitrailer active suspension design in Section 4 illustrates the use of the tool that was developed. Section 5 concludes the paper.

2. Theory

The development of any systematic and rigorous IO selection procedure faces two major problems. First, the selection is a combinatorial feasibility or optimization problem. When there are n_u actuators and n_y sensors to choose from, the possible number of ways they can form different combinations is $(2^{n_u} - 1)(2^{n_y} - 1) + 1$. This number accounts for the fact that when the open loop is tested there is no need to test combinations without sensors or actuators, because in those cases the loop is still open. For any reasonably sized problem, this number is huge, and an exhaustive search is out of the question. More ingenious ways to solve the problem are called for.

Second, to make the selection rigorous, a quantitative measure, indicating irrefutably that a certain combination of actuators and sensors is guaranteed to meet the specifications, is needed. For control system design a rigorous specification is a robust performance requirement. Almost all effects that diminish performance, *i.e.*, structural limitations of the plant, disturbances, noise, plant-model mismatch, can be brought into the problem formulation. This means that a combination that meets the requirement can be expected to work in practice with high probability. There are several design methods, *e.g.*, $\mathcal{H}_\infty$ controller design or μ-synthesis, that address this specification. A selection criterion that is based on a full controller design is not attractive time wise. So, a more compact and efficient criterion has to be set up.

The following sections discuss the methods devised to, more or less, beat the curse of dimensionality, and to assess robust performance efficiently.

2.1. Preliminaries

To formulate the selection procedure use is made of the standard plant in Fig. 1. The variables w and z represent performance and uncertainty related signals.

The goal is to select all combinations of actuators and sensors, called IO sets, for which a controller exists that gives the closed loop system the desired level of robust performance. This leads to the criterion

$$\|H(G_i, C_i)\|_\mu < \gamma,$$

where H is the closed loop plant, G_i the generalized plant corresponding to the i-th IO set, C_i the

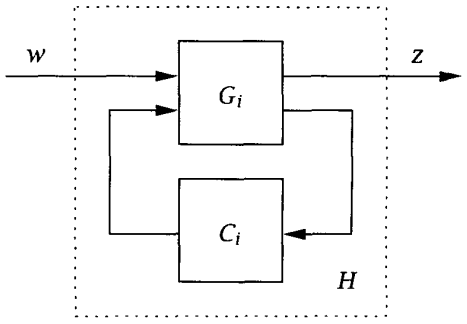

Figure 1: Standard plant

μ-optimal controller for the i-th IO set, $\|\cdot\|_\mu$ the peak value of μ across frequency, and γ the desired performance level. Robustness is guaranteed by employing μ, the structured singular value. This criterion depends on the controller, which is not desired nor always needed. So, in the following the criterion is, at least partially, adapted to only depend on G_i.

2.2. Search method

All possible IO sets can be mapped one-to-one on the power set of $n_u + n_y$ pairs of $\{0, 1\}$, where the 1 indicates that a device is used. The subset inclusion relation between the possible IO sets establishes a partial order and gives the power set the structure of a partial ordered set (poset). This poset can be graphically represented in a Hasse diagram, see Fig. 2. At the top of the diagram is the full IO set, using all potential IO devices. At the bottom is the empty IO set, that is just the open loop.

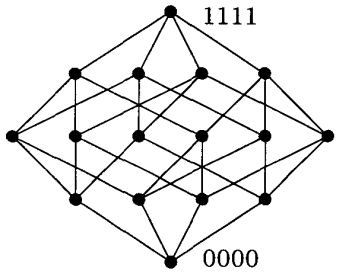

Figure 2: Hasse diagram for $n = 4$

The goal is to split up the Hasse diagram or poset in two parts, one part containing all IO sets that are feasible, the other with those that are not, with only performing a, preferably small, number of feasibility tests. If an IO set i, with its associated generalized plant G_i, optimal controller C_i, and fitness function $f(G_i, C_i)$, meets a certain criterion $f(G_i, C_i) < \gamma$, it is feasible. Better performance is translated into lower levels of γ.

When the fitness function f has the property that if a certain IO set is not feasible, all its subsets are not feasible also, we call the function monotonous. Formally,

Definition 1
A fitness function $f(G_i, C_i)$ is called monotonous *if for all IO sets k, and for all IO sets j that are subsets of IO set k, holds*

$$f(G_k, C_k) \leq f(G_j, C_j).$$

The monotonous function property proposed above establishes the structure of an independence system on the poset. This means that the break-up in feasible/infeasible can be characterized by all so-called maximal independent or minimal dependent sets. It is therefore sufficient to establish one of these sets. In terms of IO selection, the dependent sets are feasible, so they meet the selection criterion, and the independent ones are not. A minimal dependent set is a combination of actuators and sensors that, when any one of the elements of the set is excluded, does not meet the criterion. A maximal independent set is such that including any input/output device, that is not already in the set, makes it feasible.

For a general independence system it is known that it is not possible to determine all maximal independent or minimal dependent sets in time polynomial in $n = n_u + n_y$. Stronger, it is not even possible in time polynomial in n and $K + M$, where K is the number of maximal independent sets and M the number of minimal dependent sets [2]. This means that it is unlikely that large-scale problems can be solved efficiently: the problem is called intractable.

There are certain classes of independence systems that are tractable, because they have additional properties. For instance, matroids have one additional property that make them easily solvable by greedy algorithms [3]. Unfortunately, the IO selection based on a robust performance requirement cannot be formulated as a problem using matroids. Perhaps with other fitness functions it is possible, while still keeping the fitness function physically relevant, but that is unknown up-till now.

However, in this case there is another additional property, namely a kind of time separation, where the feasibility test takes much more time than all other operations needed to establish all maximal independent or minimal dependent sets for n not too large. In an implementation of the tool to solve the IO selection problem the other operations are mainly bitstring comparisons. A single feasibility test takes approximately $10^7 \cdots 10^{10}$ more time than a single bitstring comparison for moderately sized systems. Solving an IO selection problem is therefore dominated by the feasibility tests if n is not too large.

Fortunately, it is possible to solve the problem with a number of feasibility tests that is polynomial in n, K, and M, due to the following proposition [4].

Proposition 1
All minimal dependent sets or all maximal independent sets of a power set with partial order and independence system structure can be found with $O(nK + M)$ or $O(nM + K)$ visits to an oracle that decides whether a set is independent or not.

Remark 1
This result may look better than it really is, because both K and M may depend combinatorially on n. If only a partial solution is needed, *e.g.*, a single minimal dependent set, the problem is much simpler and no longer intractable, requiring only $n + 1$ tests, but such a solution may not be the one that is asked for: it just meets the performance requirement, but may still employ an unnecessarily large number of actuators and sensors. Looking for the minimal dependent sets that are the smallest ones makes the problem intractable.

Remark 2
The proof of the proposition is based on the analysis of an algorithm for a recursive search that traverses the Hasse diagram. If the search is started from the top (full IO set) at most $(n+1)M+K$ visits to the oracle are needed. If the search is started from the bottom (empty IO set) the number of visits is bounded by $(n + 1)K + M$. Depending on the values of M and K, one of the two search directions is preferred.

2.3. Fitness function

To use the previous results and to get a more or less efficient method to select IO sets that are feasible, *i.e.*, that meet the robust performance requirement, use is made of the following.

Proposition 2
Let the design problem be formulated in such a way that for all IO sets i, including the empty one, the closed loops $H(G_i, 0)$ are identical. Then, $\|H(G_i, C_i)\|_\mu$ is a monotonous fitness function.

Proof: With the assumption that all $H(G_i, 0)$ are equal, the only difference in the closed loop for all possible IO sets is the controller. It is enough to note that, for all IO sets j that are subsets of IO set k, the controllers C_j are contained in the class of controllers covered by C_k. For a suitable ordering of inputs and outputs C_k can be chosen as

$$C_k = \begin{bmatrix} C_j & 0 \\ 0 & 0 \end{bmatrix},$$

and the closed loop is equal to that for C_j, for all j. So, controller C_k has more freedom to achieve a lower peak value of μ, but is at least able to keep μ the same as is possible with controllers C_j. ■

Remark 3
In practice μ nor a μ-optimal controller can be computed efficiently and exactly. Almost exclusively a method based on the μ upper bound and so-called

D-scales is used, intertwined with the computation of an $\mathcal{H}_\infty$-optimal controller in an iterative manner. To establish feasibility based on this approximate method is *not* necessarily monotonous. In practice this appeared to be not that much a problem, because the μ upper bound is generally tight and, even if it is not, the selection is conservative because it will accept IO sets for which the μ upper bound is smaller than the selection level γ, so μ itself will be smaller also.

Remark 4
Not all fitness functions employed in IO selection have the monotonous property defined above. For instance, a fitness function based on the condition number, that is often used in approximate methods for IO selection, is not necessarily monotonously non-decreasing if actuators or sensors are removed from a given IO set.

Remark 5
The assumption made in the proposition excludes the use of actuators and sensors that require, *e.g.*, a structural or parametric change in the model of the system. Those devices cannot be included in or excluded from an IO set by simply changing the input and output matrices of a state space representation of the generalized plant, possibly generating other differences in the closed loop than those caused by the use of another controller only. The assumption is only sufficient. It may be relaxed in some cases, but the assumption in its present form has the advantage of not depending on the controller, that will not be available at this stage in the design process.

From the previous discussion the main result now follows

Theorem 1
Let for all IO sets i the closed loops $H(G_i, 0)$ be identical. Then, the sets characterizing the full solution of the IO selection problem using as fitness function $\|H(G_i, C_i)\|_\mu$ can be found with at most $O(nK + M)$ or $O(nM + K)$ feasibility tests.

3. Tool

A tool has been developed to select feasible IO sets. It consists of two major components. The first part is a search strategy and computer routine to compute the complete independence system structure. All minimal dependent and maximal independent sets are searched for. The second part is a routine to compute the fitness function based on μ and $\mathcal{H}_\infty$ theory and to check feasibility.

3.1. Search strategy

The search algorithm mentioned in Remark 2 and detailed in [4] is implemented in the program `visor` (vast independence system optimization routine). It is called an optimization routine, not a feasibility one, because it computes all maximal independent sets, so also the largest ones, the maximum, and all minimal dependent sets, so also the smallest ones, the minimum. This program enables the IO selection process to be carried out for large-scale problems.

Investigations for some artificial worst-case problems have shown it to be efficient for at least problems with up to 24 binary items to choose from. For randomly chosen independence system structures, thought to be more representative of real world problems, a size of 34 items is also solvable in reasonable time. For special, simple cases the maximum number of 64 items for the current implementation, being the word size on 64-bit computers, is also possible. In the foregoing, a time to solve the problem is reasonable if it is dominated by the feasibility test, so still shows a "polynomial" growth rate.

3.2. Feasibility test

To get a criterion that does not depend on a controller and to further reduce the time needed to determine the sets characterizing the solution, it is advantageous to make the feasibility test as far as possible controller-independent and efficient. This has been carried out for a test based on sufficient and necessary conditions for the existence of a controller achieving a specified $\mathcal{H}_\infty$ norm, see [5,6]. Those conditions can be based on Riccati equations, or on feasibility of a related linear matrix inequality problem. The first option is preferred based on running time, because it is faster, the second is preferred based on general applicability, because it removes some technical assumptions.

Here, the problem is not a straightforward $\mathcal{H}_\infty$ control one, but it can be cast into this form by employing D-scales that are based on data from approximating μ by its upper bound and that are normally used in generating μ-optimal controllers.

The feasibility test for an IO set i now proceeds as follows.

1. Set up the generalized plant G_i corresponding to IO set i.
2. Expand the plant with the most recently computed D-scale transfer functions or with those successfully used for the previous IO set.
3. Check if the $\mathcal{H}_\infty$ existence conditions for a level γ are met, if yes, return TRUE.
4. Compute an $\mathcal{H}_\infty$-optimal controller.
5. Form closed loop and compute μ for a grid of frequencies.
6. Check if peak value of μ is less than γ, if yes, return TRUE.
7. Check if iteration with D-scales has converged, if yes, return FALSE.
8. Compute new D-scale transfer functions.
9. Proceed with 2.

Remark 6
In practice, quite often the $\mathcal{H}_\infty$ existence conditions check is the only thing done, no controller needs to be computed. Only IO sets that fail these conditions need to go through the whole rigmarole of controller computation to rigorously verify if they are feasible.

Remark 7
The test procedure outlined above is reminiscent of the iterative method used to design μ-optimal controllers. The main differences are that it starts with D-scales that are already available and is terminated prematurely if the test level is met. It is known that μ-optimal controller design is a non-convex optimization problem, so reaching a global optimum cannot be guaranteed. This is another factor why in practice a monotonous fitness function is not really realized.

4. Example

The methods outlined above, and their corresponding implementation, have withstood the test of a practical, large-scale problem with $n_u = 7$ actuators and $n_y = 21$ sensors to choose from for an active suspension control problem of a tractor-semitrailer model with ten degrees-of-freedom. The vehicle is a multi-body system, with as main bodies three axles, semitrailer, tractor chassis, engine, and cabin. See Fig. 3 for an overview of the system and the potential actuators and sensors. Force generators act on the system and position and acceleration are sensed. The part of the variables w and z related to the performance specifications are indicated also. More information on the model, the model parameters, and the control problem formulation, including the motivation for the choice of weighting functions, can be found in [7].

The uncertainty considered is the load of the semitrailer. The semitrailer mass and inertia vary between 0.1 and 1 times the values for the fully loaded semitrailer, while the stiffness of the rear suspension changes proportionally with its load due to the use of self-leveling air springs. Pulling the uncertainties out of the plant requires four uncertainty-related signals in w and in z. The load variations of the semitrailer are modeled as a complex repeated uncertainty. This implies that both the left and right D-scales are four by four full transfer function matrices.

The number of states for the generalized plant, tractor-semitrailer model plus weighting functions, was 26. The order of the model for $\mathcal{H}_\infty$ controller design, *i.e.*, the generalized plant with D-scales, varied between less than 50 and more than 200, depending on the maximal order of the entries of the D-scale transfer functions, which has been varied between three and eight. The closed loop used for μ computation, *i.e.*, the generalized plant with the $\mathcal{H}_\infty$-optimal controller, contained between 50 and 200 states, depending on the order of the controller.

The large number of states posed some problems in the D-scales and $\mathcal{H}_\infty$-controller computation and required some trickery to avoid numerical errors:

- sprinkling around calls to a function that balances the system representation, to avoid ill conditioned matrices and hardly controllable or observable poles
- fiddling with numerical tolerances.

It also became clear that the iteration with D-scales did not always converge to a global minimum, see Remark 7. If the iteration got stuck close to the test level γ it was restarted with different initial D-scales. Remark that the iterative test is normally started with the D-scales that were computed for the last tested IO set that was feasible. Those D-scales may be optimal for that IO set, but not for the current one by any margin.

Results for this problem were obtained for several values of the test level γ. Although the search strategy appeared to be rather efficient, the feasibility test for an IO set that is not feasible requires several iterations with controller and μ computations. This test is thus rather involved, but manageable when K and M are not too large. Those values depend, among other factors, on γ. A value of γ has been used that still yields feasible IO sets with only one actuator and one sensor and that improves the peak μ value by at least a factor 1.7 compared with the passive suspension.

A search starting with the full IO set gives $M = 58$, $K = 9$. The number of IO sets actually tested was 658. This represents a fraction $2.5 \cdot 10^{-6}$ of all IO sets. The selection procedure was completed in exactly 15 hours computing time. Table 1 gives the results of the IO selection.

Table 1: Feasible IO sets with $n_u = 1$ and $n_y = 1$

Actuator	Sensor
Rear suspension trailer u_3	Suspension deflec. trailer y_3
Rear suspension trailer u_3	Acceleration trailer axle y_{10}
Rear suspension trailer u_3	Acceleration rear trailer y_{13}

As can be seen from Table 1 there were three minimal IO sets with one actuator and one sensor. Furthermore, there were five IO sets with one actuator and two sensors and 50 IO sets with one actuator and three sensors among the minimal dependent sets.

The table also shows that actuation and sensing at the rear of the semitrailer is advantageous. This is explained by considering the uncertainty in the semitrailer load, that is most evident and easiest to tackle at or near the semitrailer itself.

From the values of M and K it may appear that a search starting from the empty IO set would be more efficient, see Remark 2. The number of feasibility tests for this case is much reduced to 270. Nevertheless, the time needed for the computation was twice

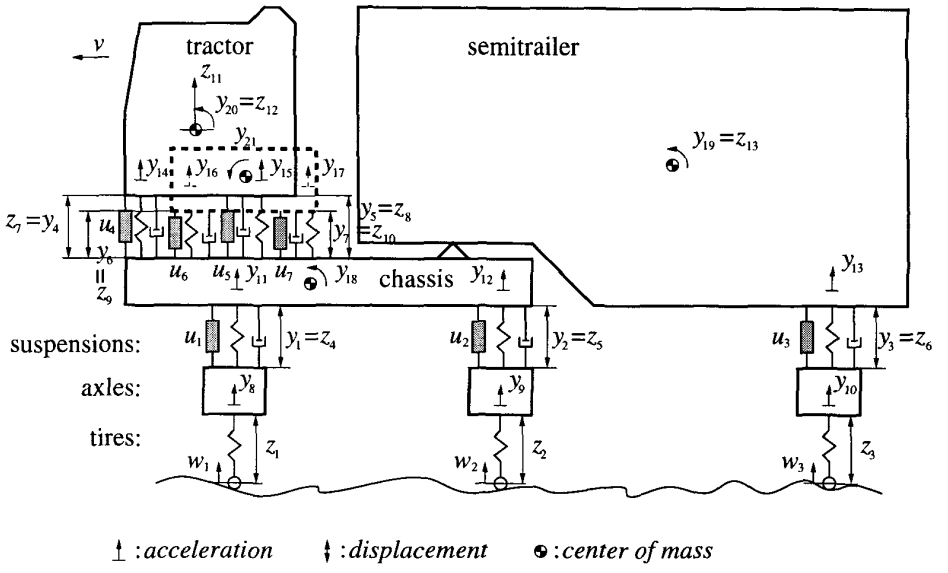

Figure 3: Tractor-semitrailer with actuators u_i, $i = 1,\ldots,7$ and sensors y_i, $i = 1,\ldots,21$

as large, namely $\approx$ 28 hours, so in average 10 tests per hour are handled. The reason is that the fraction of tested IO sets that are termed infeasible is much larger now and those IO sets need more time in the feasibility test.

It also appeared that for the search starting at the empty IO set one obtains $M = 64$ and $K = 13$, which differ from the previous values. A few cases where the test level γ can hardly be met, leading to erroneous results by convergence to a local optimum in the iterative feasibility test, see Remark 7, are the cause. A few of those cases are remedied by restarting the iteration with other D-scales, but not all. The three smallest feasible IO sets are still generated, however.

5. Conclusions

A selection strategy and a feasibility test were designed for a class of IO selection problems. A tool, enabling a rigorous assessment of all possible combinations of actuators and sensors, without performing a feasibility test for all of them, but only for a very small fraction, was developed. The approach is therefore called efficient. It may still require an amount of work that is combinatorial in terms of the number of input and output devices, but it is polynomial in terms of this number and of the cardinality of the sets that completely characterize the solution.

Application to a large-scale IO selection problem, for a medium sized plant with 28 input/output devices to choose from, shows the method and the corresponding tool to be able to tackle this problem, with enough potential to make application to even larger problems a realistic option.

Acknowledgment

We thank J. K. Lenstra for bringing [2] to our attention.

References

[1] M. van de Wal and B. de Jager, "Control structure design: A survey," in *Proc. of the 1995 American Control Conf.*, vol. 1, (Seattle, WA), pp. 225–229, IEEE, Piscataway, NJ, June 1995.

[2] E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Generating all maximal independent sets: NP-hardness and polynomial-time algorithms," *SIAM J. Comput*, vol. 9, pp. 558–565, Aug. 1980.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. The MIT Electrical and Computer Science Series, Cambridge, MA: The MIT Press, 1994. Eleventh printing.

[4] B. de Jager and O. Toker, "Complexity of input output selection," in *Book of Abstracts of the Internat. Symp. on the Mathematical Theory of Networks and Systems (MTNS 98)*, (Padova, Italy), p. FE8, 1998.

[5] M. van de Wal and B. de Jager, "Selection of sensors and actuators for an active suspension control problem," in *Proc. of the 5th IEEE Conf. on Control Applications*, (Dearborn, MI), pp. 55–60, IEEE, Piscataway, NJ, Sept. 1996.

[6] M. van de Wal and B. de Jager, "Actuator and sensor selection for robust performance," in *Proc. of the 36th IEEE Conf. on Decision and Control*, vol. 5, (San Diego, CA), pp. 4377–4378, IEEE, Piscataway, NJ, Dec. 1997.

[7] B. de Jager, "Multiobjective suspension control problem," in *Proc. of the 34rd IEEE Conf. on Decision and Control*, vol. 4, (New Orleans, LA), pp. 3652–3657, IEEE, Piscataway, NJ, Dec. 1995.