1992 BODE PRIZE LECTURE

Controller Design: Moving from Theory to Practice

Brian D.O. Anderson

Modern controller design packages often fall short of offering what is truly practical: low order controllers, discrete-time controllers operating in a sampled-data loop, and finite word length realizations of controllers with the FWL property minimally impacting closed-loop performance. Here, I shall try to describe how to achieve these objectives.

Theory Shortfall

Most good models of physical plants are continuous-time models and most physical plants are, in fact, nonlinear. The continuous-time models we use are very often linear because we are not skilled enough to handle nonlinear models. Here I will discuss the linear case. Let us suppose, then, that we have a linear model of a physical plant. There are three aspects of the controller design I want to point to:

- A great many physical plants are modeled with high-order equations. If you use a typical commercial software design package, MATRIXx for example, you will typically produce, using robust control or H_2 design methods, a controller which is also high-order.
- Physical plants are normally modeled using continuous-time models, with the coefficients in the models often corresponding to physical parameters. The result of using a commercial software design package is normally, again, a continuous-time controller.

Video Tape Available

A video tape of the Bode Lecture delivered by Dr. Anderson at the 1992 Conference on Decision and Control is available from the IEEE. Control Systems Society members should order using catalog number HV0239-3, "1992 Bode Lecture," CSS member rate $49.95 (plus $4.00 shipping and handling). This is a substantial discount from the list price ($129) and from the IEEE member rate ($89.95). For more information, call toll free (in U.S.A.) IEEE Customer Service at 1-800-678-IEEE, fax: 908-981-1667, phone: 908-981-0060.

- Even if a software design package results in a discrete-time controller for whatever reason, the description of that controller will be via a transfer function, or by a state variable realization which generally reflects no preoccupation with numerical issues.

A few years ago, I received from Dagfinn Gangsaas of the Boeing Corporation an example which was a useful testbed for a lot of work. The example was of an airframe model with two inputs, two outputs, 55 states, with unstable poles, with random disturbances, and with nonminimum phase zeros.[1] You would appreciate that if you do an LQG design or an $H\infty$ design, you are likely to come up with a continuous-time controller of an order of about 55. This cannot be implemented, because the controller would be too complex, and in any case it would have to be done in discrete time.

Presented at the IEEE Conference on Decision and Control, Tucson, AZ, December 1992. *Brian D.O. Anderson is with the Department of Systems Engineering and Cooperative Research Centre for Robust and Adaptive Systems, Research School of Physical Sciences and Engineering, Australian National University, Canberra, ACT 0200, Australia.* *This work was partly supported by the Australian Commonwealth Government under the Cooperative Research Centres Program.*

[1] *Equations for the model can be found in [1].*

 0272-1708/93/$03.00©1993IEEE

What do practicing engineers need? They need controllers which

- are low complexity,
- operate in discrete-time, i.e., are computer-implementable, and
- when implemented are free of numerical problems.

The shortfall, of course, is that many controller design algorithms do not produce these properties. So I am going to, for much of this discussion, dwell on the three issues of complexity, discrete-time controllers, and the realization of digital controllers. And I should say that there is now commercial software that more or less handles the controller complexity problem.[2] There is not, as you will appreciate further on in the discussion, good software for handling the discrete-time controller problem let alone the problem of realizing digital controllers. And at the end I do have some concluding remarks that aren't just a summary.

There are several approaches to achieving low-order controller design. Consider Fig. 1. On the left hand side is the idea that you start with a complex model of the plant, you simplify it with model reduction and then you do a controller design on the low-order plant. Alternatively, see the diagonal path, if you are lucky you might have an algorithm that takes you directly from the high-order plant down to a low-order controller. The third possibility uses a high-order model to produce a high-order controller which is then simplified. You can make some quick comments about these methods, and like all quick comments

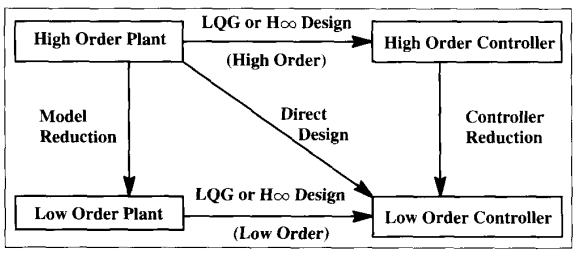


Fig. 1. Basic principles of low order controller design.

they're an approximation of the full truth. So I apologize if these quick comments don't accommodate your particular viewpoint.

As far as model reduction is concerned, where you start with a high order model as the plant, then simplify it and design a controller, the practical designers will tell you often that this doesn't work. And it really doesn't work, I believe, because the *first* step in getting to a low order controller is an approximation step. There are a number of other steps which massage that approximation into a totally unfriendly form, and the approximation criterion that has typically been used is often too crude because it doesn't respect enough closed-loop properties.

As far as direct design is concerned (and this is a topic I worked on a few years ago with my colleague John Moore [2]), there is little or no commercial software available for such designs. This means that design this way is not really especially practical. Nor is the situation likely to change in the short term because the algorithms which might be the basis for software are hard to understand intuitively, and hard to use in a way yielding convergent results.

By contrast, there is now software for controller reduction where one does a high-order controller design and then simplifies it. In that situation we are approximating last, so we can understand what is going on in the approximation process. Further, there are logically motivated approximation criteria available.

I should say that the direct design in my view has a lot of promise and a bit more work in the area of what Dennis Bernstein and Dave Hyland have been doing [3] may well change the picture as far as that's concerned. The QFT approach may also be relevant [4].

Controller Complexity

I want to focus now on controller approximation, where one is reducing the order of a controller. The key contention is that this must take into account the closed-loop, because what *one is trying to approximate is closed-loop behavior* rather than the open-loop controller sitting in isolation. This is a tremendously important observation. It's crystal clear once you've seen it but I think it was first brought to my attention well on in my professional career by Dale Enns, who was a student of Gene Franklin's at Stanford, at that time working on controller reduction. So, I want to suggest to you that if we are going to approximate a controller we first need to have some way of measuring the quality of approximation that reflects closed-loop behavior. Then you can go on from that and figure out how to actually replace a high-order controller C by a low-order controller $\hat{C}$. There are a number of ways you can measure quality of approximation and I name three. They all use the plant P.

- Transfer function matching;
- Stability robustness;
- Signal spectra matching.

Transfer Function Matching

Our goal here is that the closed-loop transfer function with a high-order controller is to be as near as possible to that with a low-order controller, so you use an index like the top one in Fig. 2. If you are concerned about matching in certain frequency bands rather than others, you could use the second index of the Fig. 2. And if the error is small enough, you can see that you can represent that index as a frequency weighted error between the low-order controller $\hat{C}$ and the high-order controller C.

Transfer Function Matching

$$\|PC(I + PC)^{-1} - P\hat{C}(I + P\hat{C})^{-1}\|_\infty$$

or with frequency weighting

$$\|PC(I + PC)^{-1}W - P\hat{C}(I + P\hat{C})^{-1}W\|_\infty$$

or with small error

$$\left\| \underbrace{(I + PC)^{-1}}_{KNOWN}\underbrace{P(\hat{C}-C)}_{ERROR}\underbrace{(I + PC)^{-1}W}_{KNOWN} \right\|_\infty$$

Fig. 2. Index for transfer function matching.

[2] *The package MATRIXx included a model reduction module in 1992, and this should soon be available in X-Math.*

Stability Robustness

Assume

- same count of unstable open loop poles in C and $\hat{C}$
- C stabilizes P
- $\|(C - \hat{C})P(I + CP)^{-1}\|_\infty < 1$

Then $\hat{C}$ stabilizes P.

Seek to minimize

$$\left\| \underbrace{(C - \hat{C})P(I + CP)^{-1}}_{ERROR \quad KNOWN} \right\|_\infty$$

Fig. 3. Motivation of index based on stability robustness.

Stability Robustness

Dale Enns' idea was to consider stability robustness to get an index, and he used a fairly well known stability theorem which says: Suppose you design $\hat{C}$ so that it has the same number of unstable poles as C and suppose C stabilizes the plant, which of course it should; then if the inequality of Fig. 3 is satisfied $\hat{C}$ will also stabilize P. Further, if the quantity on the left of the inequality of Fig. 3 is very small, $\hat{C}$ is going to do a very nice job of stabilizing P and thus be an even better approximation of C. So that motivates looking at this frequency weighted index which is a bit different to the previous transfer function matching one, but it is still frequency weighted.

Signal Spectrum Matching

I became alerted to yet another index through Bob Skelton of Purdue. Suppose the controller is processing signals that are very strong around 1 Hz and very weak away from 1 Hz. Then it makes sense to insert a weighting that says the reduced order controller should do a very good job near 1 Hz and it doesn't matter what sort of a job it does away from 1 Hz. Put another way, you use a weighting that is based on the spectrum of the controller input. Thus if the spectrum Φ of the controller input factors as $\Phi = WW^*$, with W stable and minimum phase, one adopts $\|(C - \hat{C})W\|_\infty$ as the index.

There are variations on the above indices, but the global point is that in all these indices the plant is somehow reflected, and in the three examples I have given you, you have in fact $C - \hat{C}$ (the controller error) modified by some frequency weight. And that frequency weight involves the plant.

Having posed the index, how do you then reduce it? One way you can do this, and it has found its way into the commercial software, is to use a modification of balanced realization truncation. The modification, again, originally came from Dale Enns. Balanced truncation without weighting does a good, but approximate, job of finding a low order $\hat{C}$ given a high-order C which should minimize $\|C - \hat{C}\|_\infty$. The frequency weighted version is aimed at the index $\|W_1(C - \hat{C})W_2\|_\infty$. A proof of the effectiveness of this algorithm (using, as in the unweighted case, an error bound in terms of Hankel singular values) has yet to be published (unpublished results are known to the author).

In the last few years people have also been trying to do a frequency weighted Hankel norm reduction. I consider I came to grief on that problem, at least in practical terms, but Keith Glover, David Limebeer, and Y.S. Hang [5] have obtained some results which might enable you to do this reduction more efficaciously than with a generalization of balanced realization truncation.

Open-Loop Unstable Controllers and Fractional Representations

Since balanced realization truncation demands stability, you might ask what happens if the controller is unstable. All you can really do is reduce the stable part of the controller and copy the unstable part into the reduced order controller. Another thing though that you might think of doing, if you have an unstable controller, is to represent the controller transfer function as a fraction ND^{-1} where N and D are both stable rational transfer functions (or possibly stable rational transfer function matrices) and proper. And then you approximate both the numerator N and the denominator D and construct $\hat{C}$ using the approximants $\hat{N}$ and $\hat{D}$ of N and D. But then there is an infinite number of fractions so the question arises which fraction should be chosen. Keith Glover and Duncan McFarlane have used normalized representations of the plant (i.e., $NN* + DD* = I$) and that is related to stability robustness [6].

Several years ago a student of mine, Yi Liu, and I used a different idea [7]. Suppose the controller has been obtained by a linear quadratic design process (see Fig. 4). Then it is fairly easy to write down a particular fractional representation ND^{-1} of the controller transfer function; again, see the figure. If you design the LQG correctly, both N and D are, of course, stable. And if you take the view that you are going to approximate the controller based on thinking about the spectrum of its input, this gives you a very easy approach. Observe that N is the transfer function from the point labeled innovations through to the output of the block $-F$, and D to within an inessential constant is the transfer function from the innovations round to the output of C. So if you didn't have that feedback interconnection with the minus sign, N and D would be the two transfer functions from the innovations modulo minus signs and constants. Since the innovations are white for a normal LQG design, and if you take the view that a good way to approximate controllers is to take notice of the spectrum of the controller input, it turns out in this case due to the whiteness that you don't need any frequency weighting. So you can approximate both N and D without weighting, and then the reduced order controller is obtained by implementing $\hat{N}$, and $\hat{D}$, plus the interconnection.

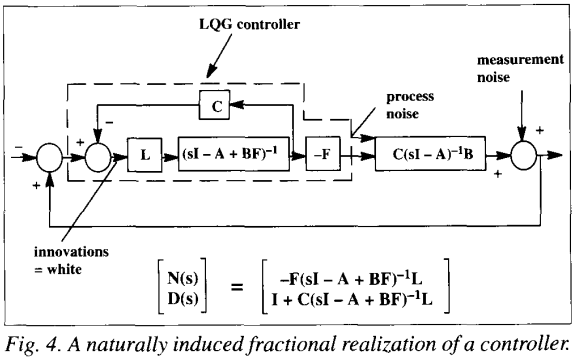


Fig. 4. A naturally induced fractional realization of a controller.

If you use stability robustness rather than the spectrum of the controller input, there is a rather complicated formula that shows that there is a natural weighting. One has

$$\begin{bmatrix} I + F(sI - A + LC)^{-1}B & -F(sI - A + LC)^{-1}L \\ -C(sI - A + LC)^{-1}B & I - C(sI - A + LC)^{-1}L \end{bmatrix} \begin{bmatrix} N(s) \\ D(s) \end{bmatrix} = \begin{bmatrix} I \\ 0 \end{bmatrix}$$

which suggests that if the 2×2 block is described by $W(s)$, one should seek $\hat{N}$, $\hat{D}$ to minimize

$$\left\| W(s) \left\{ \begin{bmatrix} N(s) \\ D(s) \end{bmatrix} - \begin{bmatrix} \hat{N}(s) \\ \hat{D}(s) \end{bmatrix} \right\} \right\|_{\infty}.$$

In Enns' thesis there was a multidisk system and I have chosen that to illustrate with Fig. 5 what can happen. We are looking here at a step response obtained when an eighth order controller has been reduced to a second order controller. And you can see, there are three horrible curves, and those horrible curves were obtained by reduction procedures which did not take into account the closed loop which just focused on the controller alone. There are also three curves almost together; one is the full order system and the other two are what you get when you use Enns' method, and when you use that method combined with the idea of the fractional representations and the innovations. So you can see in this case, provided you take into account the presence of the plant in a reduction process, you have a very simple acceptable controller. A full discussion, including references to the less attractive methods, can be found in [8].

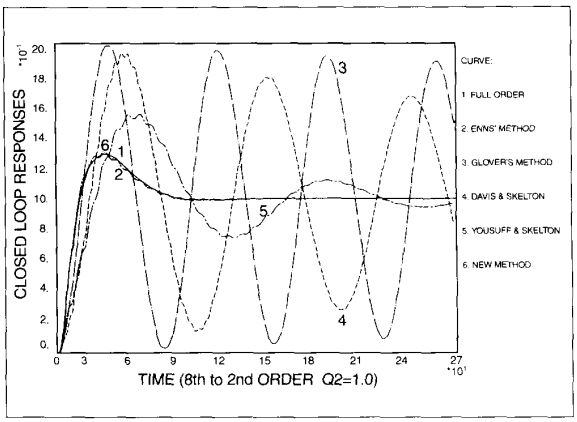


Fig. 5. Step response comparison.

Discrete Time Controllers

So let me switch to the next topic, which is Discrete Time Controllers. And it has something in common with the previous one. The fundamental question that I want to consider that maybe you considered in a first year graduate course or even a senior year digital control course is: *Given a continuous time controller how should we determine an equivalent discrete time controller with sample, hold, and anti-aliasing?*

Controller discretization is a form of approximation. You have got a close-to-ideal controller that's continuous time and you want to replace it by something which is close. Based on what I said about controller reduction, it should be no surprise to you if I claim here that we should take into account the closed loop properties in performing an approximation process, in this case a discretization process, *so controller discretization should in some way reflect the plant.* And yet if you look at a number of excellent textbooks by people who rightly have been honored in our profession and whom I honor personally, textbooks [9]–[11] by Karl Åström, Bjorn Wittenmark, and Gene Franklin *et al.* and Jürgen Ackermann, stars like these, you will find collections of formulas like those of Fig. 6. These are formulas for replacing continuous time controllers by discrete time controllers and the plant never appears. So my precept or dictum has been violated. This may be acceptable if you are discretizing with a short sampling interval, but it may not be acceptable if you have long sampling times.

With $C(s)$ continuous time and $C_d(z)$ discrete time,

$C_d(z) = C\left(\frac{z-1}{T}\right)$	Euler or forward difference
$C_d(z) = C\left(\frac{z-1}{zT}\right)$	Balanced difference
$C_d(z) = C\left(\frac{2}{T}\frac{z-1}{z+1}\right)$	Tustin or bilinear
$C_d(z) = C\left(\frac{\omega_1}{\tan\left(\frac{\omega_1 T}{2}\right)}\frac{z-1}{z+1}\right)$	Tustin with prewarping
$C_d(z) = \frac{(z-1)}{Tz}\frac{1}{2\pi j}\int_{\gamma-j\infty}^{\gamma+j\infty}\frac{e^{sT}}{z-e^{sT}}\frac{G(s)}{s}ds$	Step-invariance
$C_d(z) = \frac{(z-1)^2}{Tz}\frac{1}{2\pi j}\int_{\gamma-j\infty}^{\gamma+j\infty}\frac{e^{sT}}{z-e^{sT}}\frac{G(s)}{s^2}ds$	Ramp-invariance

Poles and zeros of $C_d(z)$ are images under $z = e^{sT}$ of those of $C(s)$, with $C_d(1) = C(0)$.

Zero-order hold equivalance.
First-order hold equivalance.
Triangular-hold equivalance.

Fig. 6. Textbook schemes for replacing a continuous-time controller by a discrete-time controller.

You may believe that an easy way out is to discretize the plant at the beginning and obtain a discrete time controller. I'll remind you of the arguments against that.

• You may lose sight of intersample behavior, depending what method you use. (Only in the last two years have people ex-

plained for example how to do $H\infty$ design with a discrete compensator attached to a continuous plant.)

- Very importantly, you may lose physical insight; the continuous time plant F matrix becomes exp FT on discretization, and if F has a physical parameter in only one or two entries, in exp FT the parameters just go everywhere.
- Design for parameter variations then becomes more difficult.
- Practitioners report that typically the choice of the sampling time has to be very conservative, that is, short.

How are we going to measure the quality of discretization? Here are two possibilities, there are certainly others.

- One can try try to match input-output performances of the continuous-time and sampled-data systems. The latter does not possess a transfer function description, which makes the problem hard.
- You could set up a stability robustness index, by adapting the procedure that Enns used in the controller reduction.

To look at one of these, I have picked the transfer function index and setting up this index is a good part of the problem. Much of this work I carried out with a visiting colleague, Jürg Keller, from the Swiss Federal Institute of Technology; for a summary see [12]. We have a plant $P(s)$, a continuous controller $C(s)$ and we'll assume that the anti-aliasing filter $F_a(s)$ has been specified. We will assume that the sampling rate has been specified, and we will assume that there is a zeroth order hold. (Later you can play with other sorts of holds for other sampling intervals). We will also reasonably assume that the continuous-time closed loop is stable and has a transfer function $T(s) = PC\ (I + PC)^{-1}$. To set up the index, consider Fig. 7, where you will see both the closed loop with a continuous controller, and the closed loop with a discrete controller. Ideally, you would like e to be zero no matter what r is, but that's not possible. So we decide that we would like e to be as small as we can obtain through choice of $C_d(z)$. And we may decide to put in some weighting $W(s)$ to allow better matching of input performance at some frequencies and worse matching at others. With a stabilizing discrete compensator, then from r to e there is an operator that maps L_2 signals to L_2 signals; it's a bounded operator, and of course it depends on $C_d(z)$. It is evidently not describable by a transfer function, but it is periodically time variant. If you formally write out what that operator is, one obtains

$$J_T = TW - PHC_d[I + SF_aPHC_d]^{-1}SF_aW$$
$$= G_{11} - G_{12}C_d(I + G_{22}C_d)^{-1}G_{21}$$

with obvious definitions of the G_{ij}. Of course, S denotes the sampler. The norm of J_T measure the quality of discretization of $C(s)$ by $C_d(z)$; we want to minimize the norm of J_T to get the best possible discretization. That is a reasonable control engineering objective.

If you just saw J_T, and someone said the task is to choose C_d to minimize J_T, and you didn't know what the G_{ij} were, you would probably say that you had an H_∞ problem. But this isn't directly a standard H_∞ problem because C_d is a discrete time transfer function but the G_{ij} are not. They would have to be all of the same family for this to be a discrete time H_∞ problem ,and they are not all of the same family. But the structure of the J_T formula is very reminiscent of H_∞.

Having obtained J_T, we are now faced with two key questions.

- How should one evaluate the norm?
- How can one choose C_d in order to minimize the norm and be stabilizing?

There are a number of people who have worked on similar problems and it turns out that this problem is identical in mathematical structure to a sampled data H_∞ problem depicted in Fig. 8. The plant, the hold, and the sampling interval are specified, and you have to find $C_d(z)$ to minimize the gain from w to z. People including Boyd Pearson, Bamieh and other students, Shinji Hara from Tokyo, Bruce Francis, and so on, have all looked at this problem (see e.g., [13]–[15]). And they have looked at it in the main by different techniques than those that Keller and I have used, and I am going to tell you about Keller's and my technique. Let me return then to controller discretization, noting that the same idea behind the solution will work for a sampled data H∞ problem. There are two key steps in the solution, both aimed at dealing with the key questions above.

- First perform an approximation that is as good as you like, which replaces this mixed continuous discrete problem by a multirate problem.

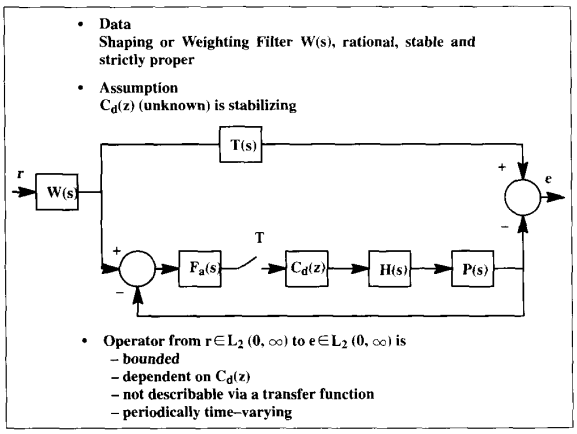


Fig. 7. Measuring the difference between using continuous and discrete controllers.

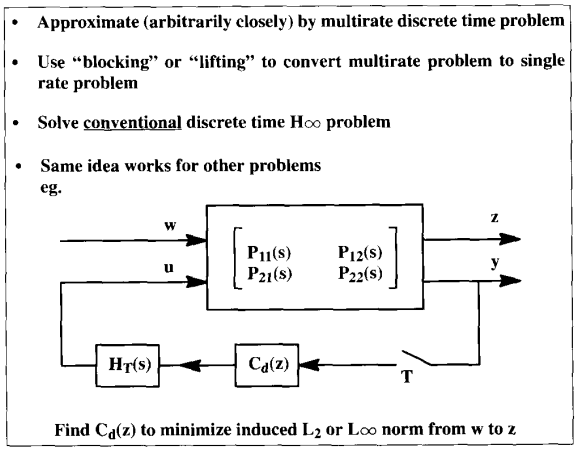


Fig. 8. Sampled-data H∞ problem.

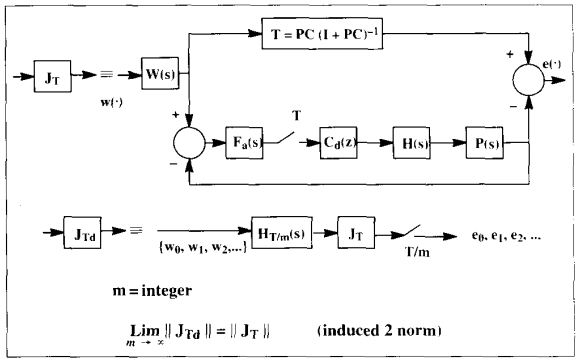


Fig. 9. Replacement of hybrid system by multirate discrete-time system with effectively same gain.

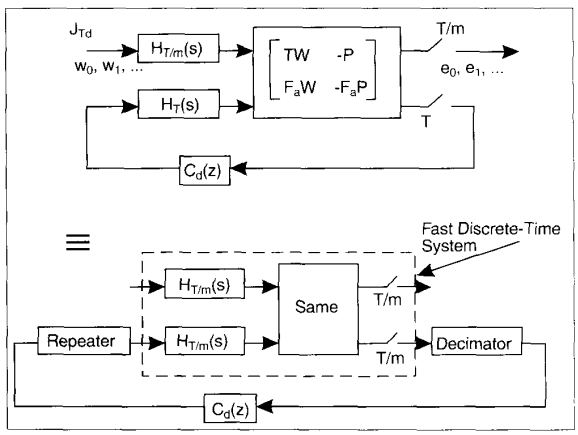


Fig. 10. Redrawing of multirate system.

• Second, convert the multirate problem to a single rate problem, and then use a commercial package to solve a conventional discrete time $H\infty$ problem.

How does this work? The first one of these steps involves converting a mixed continuous discrete problem to a multirate problem. Fig. 9 shows the main idea. A hold and sampler are introduced at the input and output of J_T, with sampling rate an integral multiple m of that applying to $C_d(z)$. This adjustment J_{Td} to J_T is a multirate system. When you make m big enough, not surprisingly J_T and J_{Td} have the same gain, or virtually the same gain. So if you're interested in evaluating the norm of J_T, you might as well evaluate the norm of J_{Td}. And if you're interested in minimizing the norm of J_T you might as well minimize the norm of J_{Td}. So that replaces the mixed continuous problem by a multirate problem.

Fig. 10 depicts a block diagram reorganization of what is going on. In the upper part the additional hold introduced is $H_{T/m}$, and the additional sampler introduced is labeled T/m. The hold associated with the discrete controller is labeled $H_T(s)$ and the sampler associated with the discrete controller is labeled T. The lower part is a redrawing of the upper part, with adjustment of the slow rate hold and sampler in an obvious manner. The decimator lets through every mth sample presented to it. The repeater causes the lower hold of length T/m to be driven for m successive intervals of length T/m by the same signal from C_d. The dashed lines enclose a fast discrete time system with the underlying sampling interval T/m, $C_d(z)$ is a slow discrete time system with the underlying sampling interval T, and the decimator and the repeater just connect up the fast and slow systems, ensuring that the right signals are presented at the right time.

The second step, having set up a multirate system, is to redescribe this as a single rate system. I first learned about this idea in the signal processing community in 1980 and I think the earliest reference that I was shown at that time was about 1972. But it's been around in the control community for a very long time also. Boyd Pearson introduced me to a 1960 reference [16]. It's known as "blocking," or it was known as blocking in 1980, but the more mathematically oriented or literate or inclined control theorists like to call it "lifting." A digital electronics person calls it "serial to parallel conversion" because that's really all that is involved. It is a trick which allows you to fool around with the rate of a discrete time system. We are going to use the trick to change the multirate system into a single rate system. The broad assertion is that if you tell me a system with a certain sampling interval, I can tell you a rewriting of the system that has a sampling interval that is an integral multiple of the one that you give me. This means we are slowing down a fast system and replacing it by an equivalent slow system; how does that work? Look at Fig. 11. The upper system has signals arriving say at 1/s, while the lower system has one 3 vector arriving every 3 s. Likewise, the upper system has one output coming out every second, and the lower system has one 3 vector coming out every 3 s. The total information entering and leaving the two systems is identical; it is just presented differently. If A,B,C,D matrices of a state-variable description of the fast system are known, the corresponding matrices for the slow system are easily found; see Fig. 11. The case of arbitrary m, rather than $m = 3$, is a trivial modification. So there is the possibility of instant conversion of a fast sampled system to a slow sampled system, where there is an integral relation between the sample rates. From an abstract point of view the inputs and the outputs of the two systems are

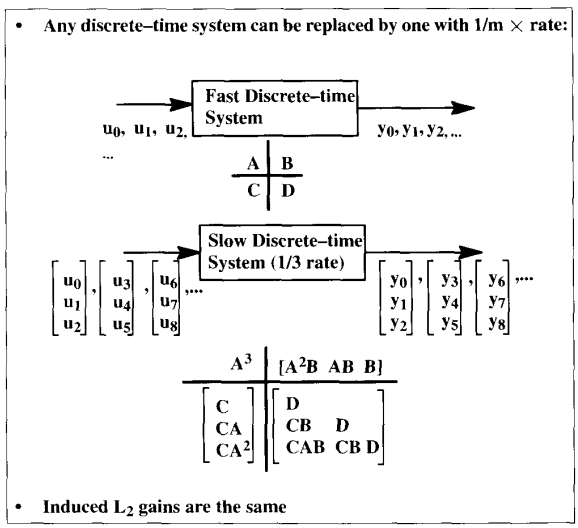


Fig. 11. Blocking or lifting on serial to parallel conversion.

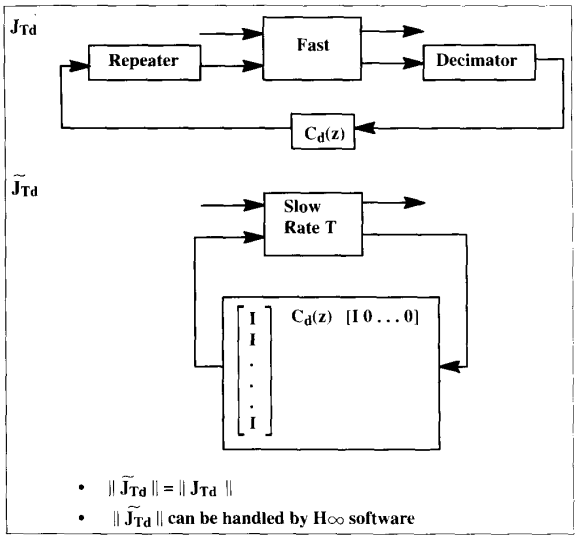


Fig. 12. Multirate to single rate conversion.

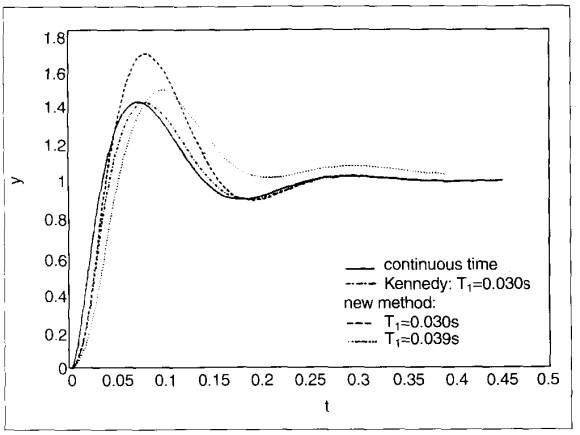


Fig. 13. Step responses with various controller discretizations.

exactly the same, they are just coded or written down a little bit differently. So the gains of the two systems must be the same.

Now we can carry out this trick on the multirate system, see Fig. 12, by just doing the trick on the fast part and leaving the slow part alone. The repeater and decimator blocks in the multirate system correspond to the $[I\ I \cdots I]'$ and $[I\ 0 \cdots 0]$ blocks in the single rate system, and the gains are the same. In the single-rate system, everything is known except C_d. Our first problem, how to evaluate the gain of J_T, has now become easy, because $\hat{J}_{Td}$ is just an interconnection of discrete time entities, all with transfer functions and all with the same rate. And in fact, determination of the C_d to minimize that gain is a standard $H\infty$ problem! That is how you can define a C_d which will be a good discretization of a $C(s)$, and it's also how you can solve a sampled data $H\infty$ problem, and to me it has got a lot of intuitive content.

Let's look at an example. There is a book by Katz [17] which compares a number of standard controller discretization methods for a loop with plant and (continuous-time) controller given by

$$P(s) = \frac{863.3}{s^2} \quad C(s) = \frac{2940(s + 29.4)}{(s + 294)^2}.$$

The sampling interval was 0.03. Only one of the methods gives a stable closed-loop, and the overshoot is enormous. Fig. 13 shows the result of using the scheme outlined, with $T = 0.03$ and a larger $T = 0.039$. (Compare these values with the rise time.) The label Kennedy refers to the person who introduced me to the problem of controller discretization. He was interviewing to be a doctoral student and for his Master's thesis he had worked out a general discretization method (only applicable to scalar systems) and designed a controller for a radio telescope antenna, after learning that all the standard textbook methods didn't work [18].

Although in this particular example seven out of the eight conventional methods give instability, and the eighth conventional method gives an unacceptable design, the problem is that one is seeking too long a sampling interval for the controller. The conventional methods will be satisfactory if you choose a short enough sampling interval, even though they don't respect the precept enunciated earlier that discretization should really involve consideration of the plant in the closed loop.

Incidentally, there are other methods for controller discretization that respect closed loop behavior. You can look at the spectrum of the signal going into the controller as an example, and there are some other methods peculiar to single input single output systems. So there has been scattered attention to this material but it hasn't become very well known.

Realizing Digital Controllers

Now I want to go on to the problem of realizing digital controllers. I should say that I was introduced really to this problem by my good friend Michel Gevers, and one of his students, Li Gang, who is now at Nanyang University of Technology in Singapore. One of my colleagues at ANU, Darrell Williamson, also works on these sorts of problems.

What is the technological problem? As soon as you have some digital hardware you have to reckon with three phenomena that control theory people often pretend shouldn't be a bother. Those are:

- quantization noise (after every arithmetic operation),
- finite word length implementation of filter coefficients, (so there is going to be a little error in general, unless you are implementing a 1 or a 0 or a 1/2 or something like that), and
- overflow (but only if you are not careful).

This means that if you tell me a transfer function, and I tell you a realization for it and plug it in and implement it in finite wordlength arithmetic, the performance you actually get will depend on the realization. Finite word length effects will clearly differ, depending on the particular realization. And *there may well be an optimum state variable realization.* Needless to say, the digital signal processing people have been on to this issue for a very long time so let me tell you about some early results due to Huang, Mullis, and Roberts [19], [20].

For the moment, we suppose that we are not implementing a controller, but just implementing a stable digital filter that is not sitting in a loop. We will suppose it has a realization given by a

triple $\{A,B,C\}$ and transfer function $H(z) = C(zI - A)^{-1}B$, and we will focus first on sensitivity. We are interested in what happens to H when A is fiddled with a little bit due to finite word length effects and we are interested in H at many frequencies. Hence, one sets up an index like

$$\left|\left|\frac{\partial H}{\partial A}\right|\right|_p^2 = \left[\frac{1}{2\pi}\int_0^{2\pi}\left[\sum_{i,j}\left|\frac{\partial H}{\partial a_{ij}}\right|^2\right]^{p/2} d\omega\right]^{2/p}.$$

We can choose $p = 1, 2, \infty$ and so on. We can do the same thing for B and C. The digital filter people strung the three together to come up with an index

$$M = \left|\left|\frac{\partial H}{\partial A}\right|\right|_1^2 + \left|\left|\frac{\partial H}{\partial B}\right|\right|_2^2 + \left|\left|\frac{\partial H}{\partial C}\right|\right|_2^2.$$

The reason why p is not the same for the three summands is that they could not handle the 2,2,2 case but could handle the 1,2,2 case.

Some analytical steps, involving initially an inequality that later turns out to be an equality, lead to

$$\mathrm{M} = (\text{trace } W_O)(\text{trace } W_C) + \text{trace } W_O + \text{trace } W_C$$

where W_O, W_C are the observability and controllability grammians associated with A,B,C. Further, M is minimized by choosing A,B,C so that

$$\text{trace } W_O = \text{trace } W_C.$$

If you look at the round off noise gain problem, which is another phenomenon but a bit related, it's virtually the same result, so that is very pretty. And of course everything is analytically computable.

You might think that it would be useful to the digital filtering people (and it is) if you introduced some frequency weights. After all, you might be more concerned to have a good realization of your digital filter from 0 to 1 kHz than from 1 to 5 kHz. And so you introduce that sort of consideration into the problem formulation via frequency dependent weights which in some way influence the calculation of the sensitivity coefficients. This yields an index like

$$M = \left|\left|W_A(e^{j\omega})\frac{\partial H}{\partial A}\right|\right|_1^2 + \left|\left|W_B(e^{j2})\frac{\partial H}{\partial B}\right|\right|_2^2 + \left|\left|W_C(e^{j\omega})\frac{\partial H}{\partial C}\right|\right|_2^2$$
$$= \text{trace } W_1 \text{ trace } W_2 + \text{trace } W_3 + \text{trace } W4$$

where the W_i are like frequency weighted grammians. If you transform A to $T^{-1}AT$, B to $T^{-1}B$ and C to CT, one has a new value for M, dependent only on

$$P = TT'$$

It is

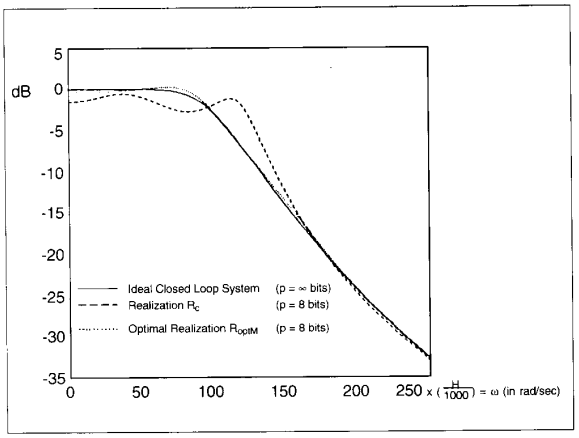


Fig. 14. Frequency response comparison.

$$M(P) = \text{trace } (PW_1) \text{ trace } (P^{-1}W_2) + \text{trace } (PW_3) + \text{trace } (P^{-1}W_4)$$

It is a nontrivial result that this frequency-weighted index does have a minimum with respect to P and it's a global minimum [21]. That is really quite a surprise. In general, it can't be analytically found but because it's a global minimum you can iterate towards it. And as a matter of fact, if you use a 2,2,2 index, which was beyond the capabilities of the digital filter people, only very recently in the last two years have people worked out how to solve that. The work was by my colleague John Moore, Uwe Helmke, from Regensburg, and a student, Jane Perkins [22].

With this background, let's go to the problem of digital controller sensitivity minimization.

First, consider just a discrete-time loop, with controller $C_d(z)$, plant $G(z)$ and closed-loop transfer function $R = GC_d(I + GC_d)^{-1}$. (Later, we can consider a full sampled-data loop.) Then if α is an entry of the A,B or C matrix of the controller $C_d(z) = C(zI - A)^{-1}B$, there holds

$$\frac{\partial R}{\partial \alpha} = (I + GC_d)^{-1}G\frac{\partial C_d}{\partial \alpha}(I + GC_d)^{-1}$$
$$= W_1\frac{\partial C_d}{\partial \alpha}W_2$$

where W_1, W_2 are certain computable weights. Any index involving the closed-loop transfer function can be turned into an index involving C_d, with different weights, which we know how to study.

What sort of results can you get when you do this sort of thing? Observe Fig. 14 which compares the ideal closed loop system with two different realizations with 8 bits.

Evidently there is an error of something like 5 dB in the realized closed loop transfer function when you optimize and when you don't optimize the state variable realization of the controller.

Suppose now you get a bit more ambitious and decide that you are going to use the sensitivity minimization problem point of view looking at the plant output at all time and not just the sample instants. Thus we want to work with a full continuous

time model of the plant rather than the discrete model which suppresses consideration of the intersample behavior. In formal terms we could write down a closed loop operator as

$$R = (I + P(s)H(s)C_d(z)SF_a(s))^{-1} P(s)H(s)C_d(z)SF_a(s) .$$

(Here, $H(s)$, S and $F_a(s)$ refer to hold, sampler and anti-aliasing filter, with the forward part of the loop PHC_dSF_a.) Formally again, one can write quantities like

$$\frac{\partial R}{\partial a_{ij}} = (I + PHC_dSF_a)^{-1} PHC(zI - A)^{-1} e_i e_j'(zI - A)^{-1} \times BSF_a(I + PHC_dSF_a)^{-1} .$$

This is all looking a bit unpleasant. The key issue is really how can we define an index so that we can make contact with all those previous ideas and at the same time get something that we can minimize.

It is not hard to verify that $\frac{\partial R}{\partial a_{ij}}$ is a causal, periodically time-varying impulse response. It makes sense, then, to define the A-dependent summand for the performance index by

$$||S_A||_2^2 = \int_0^T \left\{ \int_{-\infty}^{t} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\frac{\partial R}{\partial a_{ij}}(t,s) \right)^2 ds \right\} dt .$$

This can be evaluated using the fast sampling and lifting trick, just like an index of a time-invariant system. The overall index is more or less like one that has already been tackled, and it has the property, albeit non-obvious, that there is a global minimum with respect to choice of coordinate basis and therefore there is, in principle anyway, a good algorithm for iterating towards the minimum [23].

Duel Observation

Let me attempt to unify all the above remarks.

Because the controller sits in a loop and because the controller is there to achieve certain closed loop properties, whenever you fiddle with the controller, you should have in mind the goal of preserving the closed loop properties. And fiddling with the controller can mean

- reducing its order,
- discretizing it, and
- finding a state variable realization for its discrete time transfer function.

And in all of these things, to do them properly, or to do them in the difficult situations, you are are going to have to take the plant into account. There will be some easy situations where you don't need to be quite so sophisticated, but in principle you should take the plant into account. And you may need to do each of the above to get a practical controller. And the broad reason is the same for all three potential controller adjustments.

Now there is a dual observation to which I haven't referred to up to this point at all, and now I am moving over to a whole new body of work by people like Lennart Ljung, Robert Kosut, Bob Bitmead, Michel Gevers again, and Delft people. The plant sits in a loop and we are just as concerned about closed loop properties when the plant is sitting in a loop as when the controller is sitting in a loop. Consider the task of identifying a plant, which is an approximation task where you are trying to approximate reality with a mathematical object. Then *you need to take into account the fact that the controller is there to understand what exactly it is you are approximating when you are identifying.* You are not really approximating the open loop performance of the plant. You are approximating some aspect of the plant that most closely mirrors the closed loop behavior and that includes the controller. So if you were to identify the plant in the closed loop and then change the controller you shouldn't expect that your initially well executed identification would necessarily be a great predictor of performance with a changed controller because the loop is different.

Similarly, *if you have a very high-order plant and you think you would like to reduce it, then it only really makes sense to talk about an approximation criterion when you understand the loop in which it is, or the controller to which it is connected.* And that is the fundamental reason why solving the problem of low-order controller design via plant reduction, followed by controller design, is no good. The plant reduction step, the first step of that procedure, takes place before you know what the controller is. (There is a sort of fudge due to Michael Safonov, not discussed here).

As a third illustration of the dual observation, suppose you want to simulate a plant. Simulating a plant involves really running some form of discretization of the plant, and we just said that discretizing the controller can be dangerous if you don't take the plant into account. So *discretizing the plant when you are just simulating may be dangerous unless you somehow take the controller into account.*

I must not leave the impression that all the interesting problems are solved. There are many many future challenges, and with some easy things to do, some moderately difficult things to do, and some hard things to do. For example:

1) Controller order reduction for a sampled-data loop. (This is now virtually worked out. This is an easy problem.)

2) Controller reduction/discretization or realization with variable plants or uncertain plants. This is harder but probably doable.

3) Doing all the above for nonlinear plants, which is, I think, very difficult.

In conclusion, I must say I have had a huge amount of fun and joy and real pleasure from the interactions I have had with so many colleagues in my own workplace, and colleagues in other universities, and my students. I would encourage you to think about your professional life the same way.

References

[1] Y. Liu, B.D.O. Anderson, and Ly Uy-Loi, "Coprime factorization controller reductions with bezout identity induced frequency weighting," *Automatica,* vol. 26, no. 2, pp. 233–249, 1990.

[2] B.D.O. Anderson and J.B. Moore, *Linear Optimal Control.* Englewood Cliffs, NJ: Prentice Hall, 1971.

[3] D.S. Bernstein and D.C. Hyland, "The optimal projection equations for fixed-order dynamic compensation," *IEEE Trans. Auto. Control,* vol. AC-29, pp. 1034–1037, 1984.

[4] C.H. Houpis, "Introduction to quantitative feedback theory (QFT) techniques," in *Proc. Amer. Control Conf.,* San Francisco, CA, 1993.

[5] K. Glover, D.J.N. Limebeer, and Y.S. Hang, "A structured approximation problem with applications to frequency weighted model reduction," *IEEE Trans. Auto. Control,* vol. 37, pp. 447–465, 1992.

[6] D. McFarlane and K. Glover, "Robust controller design using normalized coprime factor plant descriptions," in *Lecture Notes in Control and Information Sciences,* vol. 138. New York: Springer Verlag, 1989.

[7] B.D.O. Anderson and Y. Liu, "Controller reduction: Concepts and approaches," *IEEE Trans. Auto. Control,* vol. 34, no.8, pp. 802–812, Aug. 1989.

[8] Y. Liu and B.D.O. Anderson, "Controller reduction via stable factorization and balancing," *Int. J. Control,* vol. 44, no. 2, pp. 507–531, 1986.

[9] K. Åström and B. Wittenmark, *Computer-Controlled Systems,* 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1990.

[10] G.F. Franklin, J.D. Powell, and M.L. Workman, *Digital Control of Dynamic Systems,* 2nd ed. Reading, MA: Addison Wesley, 1990.

[11] J. Ackermann, *Sampled-Data Control Systems.* Berlin: Springer Verlag, 1985.

[12] B.D.O. Anderson and J. Keller, "Discretization techniques in control systems," *Control & Dynamic Systems Series.* Academic, to be published.

[13] B. Bamieh and J.B. Pearson, "A general framework for linear periodic systems with applications to $H\infty$ sampled-data control," *IEEE Trans. Auto. Control,* vol. 37, pp. 418–435, 1992.

[14] P.T. Kabamba and S. Hara, "Worst case analysis and design of sampled data control systems," in *Proc 29th IEEE Conf on Decision and Control,* Hawaii, 1990, pp. 202–203.

[15] B. Bamieh, J.B. Pearson, B.A. Francis, and A. Tannenbaum, "A lifting technique for linear periodic systems with applications to sampled-data control," *Systems and Control Letts.,* vol. 17, pp. 79–88, 1991.

[16] B. Friedland, "Sampled-data control systems containing periodically varying members," in *Proc 1st IFAC World Congress,* Moscow, U.S.S.R., 1960, pp. 361–368.

[17] P. Katz, *Digital Control Using Microprocessors.* Englewood Cliffs, NJ: Prentice Hall, 1981.

[18] R.A. Kennedy and R.J. Evans, "Digital redesign of a continuous controller based on closed-loop performance," in *Proc 29th IEEE Conf. Dec. Control,* Hawaii, 1990, pp. 1898–1901.

[19] S.Y. Huang, "Minimum uncorrolated unit noise in state space digital filtering," *IEEE Trans. Acoust. Speech Sig. Proc.,* vol. ASSP-25, pp. 273–281, 1977.

[20] C.T. Mullis and R.A. Roberts, "Synthesis of minimum round-off noise fixed point digital filters," *IEEE Trans. Circuits Syst.* vol. CAS-23, pp. 551–562, 1976.

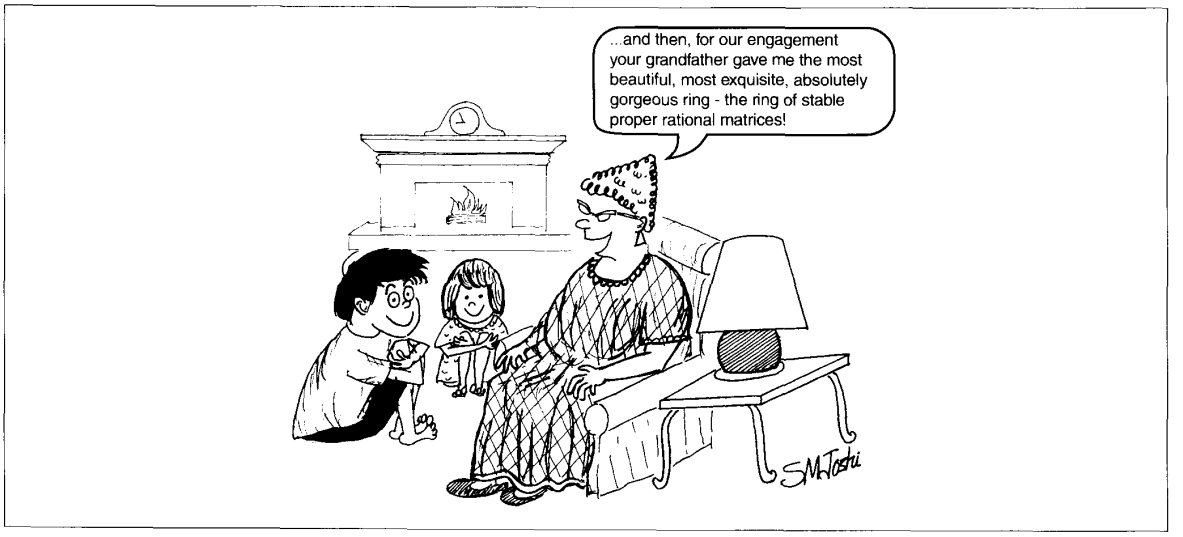
[21] L. Gang, B.D.O. Anderson, M. Gevers, and J. Perkins, "Optimal FWL design of state-space digital systems with weighted sensitivity minimization and sparseness consideration," *IEEE Trans. Circuits Syst.*, Pt. I., vol. 39, pp. 365–377, 1991.

[22] J.E. Perkins, U. Helmke, and J.B. Moore, "Balanced realization via gradient flow techniques," *Syst. Control Letts.,* vol. 14, pp. 369–380, 1990.

[23] A.G. Madievski, B.D.O. Anderson, and M.R. Gevers, "Optimum FWL design of sampled-data controllers," unpublished.

Brian D.O. Anderson was born in Sydney, Australia, and received his undergraduate education at the University of Sydney and Stanford University. Following completion of his education, he worked in industry in Silicon Valley and served as an Assistant Professor in the Department of Electrical Engineering at Stanford. He was Foundation Professor of Electrical Engineering at the University of Newcastle, Australia, from 1967 until 1981 and is now Professor of Systems Engineering at the Australian National University. His interests are in control and signal processing. He is a Fellow of the Royal Society, the Australian Academy of Science, the Australian Academy of Technological Sciences and Engineering, and the Institute of Electrical and Electronic Engineers, and an Honorary Fellow of the Institution of Engineers, Australia. He holds a doctorate (honoris causa) from the Universite Catholique de Louvain, Belgium. He is serving a term as President of the International Federation of Automatic Control from 1990 to 1993.

Out of Control



Family folklore at a typical Controls household.