Dear John

It seems LQG (repackaged in MPC) is today used almost everywhere, but there is little mention of the robustness problems with small (and even zero) gain margins that you pointed out in your famous counterexample in 1978. Did the problem "disappear" because it is rare to find cases where LQG gives poor robustness, provided that the engineer chooses "reasonable" weights? In order to look into this, I looked more carefully at your paper.

Actually, I already use your counterexample quite frequently, but in another context. I use it when reviewers complain that I don't have proper proof of stability for control structures where I make use of selectors. They typically make claims that MPC could be used instead and that it has proven stability. But most MPC proofs for such systems assume perfect measurements of all states (so, essentially they use LQR). So I say that since this assumption is not satisfied, these stability proofs are not relevant. They then argue that one may add an estimator and I reply that this my give arbitrary poor robustness (and I quote your paper). This is often enough to stop the complaints from the reviewers, especially since I can show that my PID controllers have some build-in robustness.

However, as I said, now I'm wondering about applications where MPC / LQG give non-robust designs. In your 1978-paper, the process is simply

y=m/(s-1)^2 (u + s w) (where m=1 nominally)

which in itself does not seem so difficult. Probably, the most "unusual" thing about your example is that the process noise w (disturbance) enters simultaneously into both states, so your gain matrix for w is Bw=[1;1] rather than the two "normal" cases of Bw1=[0; 1] (one disturbance at the plant input) or Bw2=[1 0; 0 1] (two independent disturbances). That is, in matlab, your problem formulation gives Qw=[1 1; 1 1] rather than Qw1= [0 0; 0 1] or Qw2=I. In the following, I will use the "normal1" case (Qw1) because I found that this gives the best LQG-results.

Here are stability margins for your process for the following designs

1. LQG Doyle (your choice for Qw). What is strange here that it is not possible to recover the LQR margins
2. LQR (with expected good margins)
3. LQG normal1 (with Qw1 = q[0 0; 0 1])
4. My SIMC PID-controller. I simply took my recommended tunings for a double integrating process.

Here are the numerical results using Matlab (PM is in degrees, wc (at PM) is in rad/s]: (see matlab-code at the end)

	GML	GM	PM	wc	DM=PM/wc	
LQG Doyle (q=s=1)	0.92	1.06	9.0	1.19	0.132	% very bad gain margins
LQG Doyle (q=1,s=1e12)	0.904	1+2e-7	18.4	3.97	0.252	% much worse GM
LQR (q=1)	0.471	inf	61.8	3.98	0.270	% Good margins
LQGnormal1 (q=1,s=1)	0.908	1.08	9.2	1.27	0.127	% very bad gain margins
LQGnormal1 (q=1,s=1e6)	0.517	6.05	51.1	3.58	0.249	% partly recovered LQR
LQGnormal1 (q=1,s=1e12)	0.474	167	61.4	3.97	0.270	% fully Recovered LQR!
SIMC-PID (tauc=0.1)	0.333	inf	46.1	10.4	0.0774	% Surprisingly good

There are many interesting things here.

1.The first is that "LQG Doyle" is always bad, no matter what values you choose for the weights (q and s= σ for the process noise). This is not seen from the above, but it follows from your analytical results: $GM = \frac{1}{df} + 1$, GM$L = \frac{4-d-f}{2df} + 1$ where $f = 2 + \sqrt{4+q}$ and $d = 2 + \sqrt{4+\sigma}$. In your 1978-paper, you stress that both GM and GML approach 1 (arbitrary poor margins) if you make q and/or σ large. But what I find even more "shocking" is that the best achievable margins (achieved when q and σ approach 0) are $GM = \frac{1}{16} + 1 = 1.0625$ and GM$L = \frac{-1}{8} + 1$ = 0.875, which both are totally unacceptable margins in practice. So no "loop transfer recovery" is possible in this case.

2.The LQR design (where both states are measured) gives as expected good margins. By increasing the weight q one gets even better GM and PM, but the delay margin (DM) becomes unacceptable because of very high gain crossover wc. Here are more detailed results:

	GML	GM	PM	wc	DM=PM/wc	
LQR (q=1)	0.471	inf	61.8	3.98	0.271	% Good margins
LQR (q=0.1)	0.497	inf	60.2	3.76	0.28	% Good margins
LQR (q=10)	0.348	inf	69.6	5.56	0.22	% Good margins
LQR (q=1000)	0.059	inf	86.6	33.65	0.045	% Good gain margins, low DM
LQR (q=1e6)	0.002	inf	69.9	1002	0.0016	% Too fast, very low DM

3. My “normal1” LQG controller (with the disturbance w entering only state x2, which is at the process input) gives poor robustness with $q = \sigma = 1$. However, as shown above, by increasing σ we are able to “recover” the LQR margins.

4. I’m also a bit surprised that my PID controller works so well. It has good robustness, in spite of having added I-action.

I hope you find this interesting!

I now have some questions for you, related to what you write in your 1978-paper

Q1. In the paper you write: *“It may, however, be possible to improve the robustness of a given design by relaxing the optimality of the filter with respect to error properties.”*

I guess you are here referring to the loop transfer recovery (LTR) approach. Yes, I find that it is possible to recover the LQ robustness by first changing Qw from [1 1; 1 1] to Qw1=[0 0; 0 1] in the filter design and at the same time choosing σ (the magnitude of w) extremely large (1.e12). However, I don’t see that people are referring to LTR in the design of MPC. Or maybe they do? Do you know how MPC people handle systems with RHP zeros (which I know from the old days may have robustness problems with LQG) ?

Q2. In the paper you write: *“It is also important to recognize that vanishing margins are not only associated with open-loop unstable systems. It is easy to construct minimum phase, open-loop stable counterexamples for which the margins are arbitrarily small”.*

Really? Even stable cases with no RHP-zeros! Given that MPC is now used so widely, I think it would be very interesting if you could share some of your counterexamples with the control community.

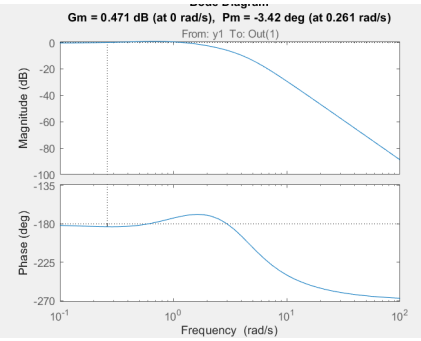
Best wishes,

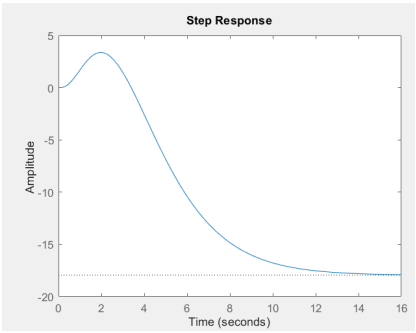
Sigurd

APPENDIX. Matlab files and results

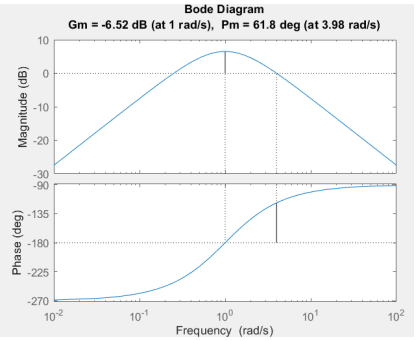
```
%Process
A=[1 1; 0 1]; B=[0;1]; C=[1 0]; D=0;
SYS=ss(A,B,C,D),  G=tf(SYS)
s=tf('s'); Gd=G*s;

%1.LQG from Doyle
q=1; Q=[1 1; 1 1]; QXU = blkdiag(q*Q,1);
s=1.e12; QW=[1 1;1 1]; QWV = blkdiag(s*QW,1);
KLQG = lqg(SYS,QXU,QWV); C=tf(KLQG)
L=-G*C; S=1/(L+1); T=1-S, step(T), figure(2), margin(L), allmargin(L), figure(1), step(T)  %
```

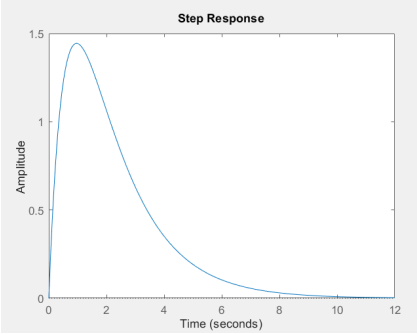
```
  GainMargin: [1.0557 0.9208 1.9809]
GMFrequency: [0 0.6039 2.9785]
PhaseMargin: [-3.4228 9.0336]
PMFrequency: [0.2611 1.1949]
DelayMargin: [23.8358 0.1320]
DMFrequency: [0.2611 1.1949]
```



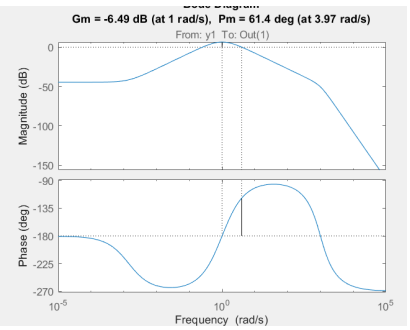


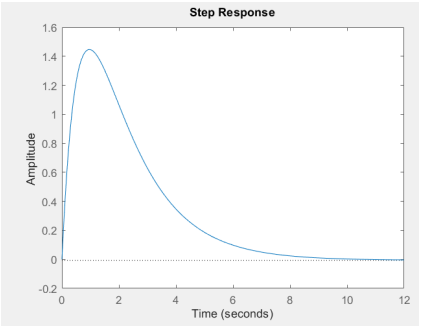

```
% 2.Try LQR (assume we can measure both states)
A=[1 1; 0 1]; B=[0;1];
q=1; Q=q*[1 1; 1 1]; R=1; N=[0; 0];
[K,SRIC,CLP] = lqr(A,B,Q,R,N)
Cx=eye(2); D=0; SYSx=ss(A,B,Cx,D); Lss=K*SYSx, L=tf(Lss), S=1/(1+L)
T=1-S, step(T), figure(2), margin(L), allmargin(L), figure(1), step(T)  %
```






```
% 3.Try LQG with another structure for QW and increase sigma to recover LQR. Yes!
A=[1 1; 0 1]; B=[0;1]; C=[1 0]; D=0;
SYS=ss(A,B,C,D),  G=tf(SYS)
q=1; Q=[1 1; 1 1]; QXU = blkdiag(q*Q,1);
s=1.e12; QW=[0 0;0 1]; QWV = blkdiag(s*QW,1);
KLQG = lqg(SYS,QXU,QWV); C=tf(KLQG)
L=-G*C; S=1/(L+1); T=1-S, step(T), figure(2), margin(L), allmargin(L), figure(1), step(T)  %
```





%4. What about PID? Tune with tauc=0.1 and approx. process as G(s)=1/s^2

```
%SIMC-tuning:
tauc=0.1; Kc=1/(4*tauc^2); taui=4*tauc ; taud=4*tauc ; tauf=tauc/10;
s=tf('s'); cpid = Kc*(1+1/(taui*s))*(taud*s+1)/(tauf*s+1);
L=G*cpid, S=1/(L+1); T=1-S, step(T), figure(2), margin(L), allmargin(L)  % GM=0.333, GM=inf
cpid=1*cpid; S=1/(G*cpid+1); T=1-S, figure(1), step(T)  %    nice response
```