Bode analysis of hydraulic positioner system
by Frank Owen, Professor, Mechanical Engineering, Cal Poly-SLO, 7 March 2014

I worked out the open-loop transfer function, closed-loop transfer function, and did the coefficient matching for the hydraulic positioning system in the Controls Lab at Cal Poly in a separate document. All the calcs were then put into a spreadsheet that is in the instructor’s lab manual for the ME 422 lab. The Matlab analysis for this is given here, using Bode diagrams to try out different things for controllers.

These first commands put in the transfer function.

$$ g = \frac{1.807}{(1/14400s^3+145/14400s^2+s)} $$

$$ g = \frac{1.807}{6.944e-05 s^3 + 0.01007 s^2 + s} $$

Continuous-time transfer function.

$$ h = 1 $$

$$ h = 1 $$

$$ g_0 = g \cdot h $$

$$ g_0 = \frac{1.807}{6.944e-05 s^3 + 0.01007 s^2 + s} $$

Continuous-time transfer function.

Now we draw the Bode plot for this system:

$$ \text{bode}(g_0) $$
But this is for the system without a gain, i.e. $K_p = 1$. My data showed me that the system went unstable at $K_p = 80$. Let’s see if the Bode diagrams confirm this.

```matlab
>> kp = 80
kp =
     80

>> golkp = kp*go1

144.6
------------------
6.944e-05 s^3 + 0.01007 s^2 + s

Continuous-time transfer function.

>> hold on

>> margin(golkp)
```
This indeed confirms that $K_p = 80$ brings the system to marginal stability. We had you in lab reduce the gain to half this value. Let’s see what that does.

\[
\text{>> } \text{golkphalf=golkp}/2
\]

\[
golkphalf = 144.6
\]

\[
\frac{0.0001389}{s^3} + \frac{0.02014}{s^2} + 2 \frac{1}{s}
\]

Continuous-time transfer function.

\[
\text{>> margin(golkphalf)}
\]
Note that reducing $K_p$ by half does not put the resulting log mag curve halfway between the curve with $K_p = 50$ and the curve with $K_p = 1$. $20 \log (0.5)$ is -6.02 dB, so this is how much the red curve above has been lower by halving $K_p$. This gives $\Phi M = 39.3$ degrees @ $\omega \Phi M = 74.5$ rad/sec. What would this give us for a step response? Obviously it’s a type 1 system, so with a step input, there would be no steady state error. Our $\Phi M$ of 39.3 degrees give us $\zeta = 0.393$, using the abbreviated formula. This should then give a %OS:

```
>> z = 0.393

z =

    0.3930

>> protos = exp(-z*pi/sqrt(1-z^2))*100

protos =

    26.1145
```

Let’s see if this is so. In the lab, the step size is 0.2 inches. There, we don’t use a higher step size because we’ll saturate the output from the controller, which is limited to ±10 volts. But in our linear model here, there is no saturation. So we’ll leave the step size at 1 inch so we can see the %OS clearly.
Also, we can calculate the peak time:

\[
\begin{align*}
\text{>> } wfm &= 74.5 \\
\text{wfm} &= 74.5000 \\
\text{>> } wn &= \frac{wfm}{\sqrt{2+\sqrt{1+4z^4}}} \\
\text{wn} &= 86.7380 \\
\text{>> } wd &= wn\sqrt{1-z^2} \\
\text{wd} &= 79.7589 \\
\text{>> } Tp &= \frac{\pi}{wd} \\
\text{Tp} &= 0.0394
\end{align*}
\]

To verify this, we need the closed-loop transfer function, because the Bode analysis uses the open-loop transfer function to conclude things about the closed-loop response. One caution here is that the relationships between $\Phi_M$ and $%OS$ and $\omega\Phi_M$ and $\omega n$ are for a pure second-order system, and we know that this is a third-order system.

\[
\begin{align*}
\text{>> } gcl &= \frac{gokphalf}{1+gokphalf} \\
gcl &= \frac{0.02008s^3 + 2.911s^2 + 289.1s}{1.929e-08s^6 + 5.594e-06s^5 + 0.0009611s^4 + 0.1006s^3 + 6.911s^2 + 289.1s}
\end{align*}
\]

Continuous-time transfer function.
The resulting closed-loop transfer function seems incorrect, but this (hopefully) is just an artifact of Matlab not simplifying for you. Let’s try this out:

```
>> figure(2)
>> step(gcl)
>> grid on
>> figure(2)
>> step(gcl)
>> grid on
```

A close-up of the peak:
The overshoot looks like it is more like 34% than the calculate 26%, but the $\Phi M = \Phi M(\zeta)$ relationship is only an approximation, and then it's an approximation for pure second-order systems, which this is not. Likewise, the peak time is about 0.041 seconds instead of the predicted 0.0394 seconds, but this is pretty close.

Let's go back to the original, marginally stable system and see if we can stabilize it with PD control—i.e. perform a “phase lift” on it. $\omega \Phi M$ was 120 rad/sec. Let’s put a lead in one decade before (shooting for a new FM of 90 degrees, but knowing we will get less). What this will do hopefully will stabilize the system without de-tuning like we did by cutting $K_p = 80$ in half. So the lead will go in at 12 rad/sec.
This did indeed lift the phase curve up. But it also raised the log mag curve, so that \( \omega \Phi M \) has moved out to where \( \phi \) is not far (18.9 degrees) above -180 degrees. It looks like we might be able to fix this somewhat by moving the break frequency of the lead a little to the right, maybe half a decade. Since the horizontal scale is logarithmic, let's move the break frequency to about 60. (To see this, you need to draw out the logarithmic scale and see that 50-60 is sort of halfway between 12 and 120.)

```matlab
>> golkplead2 = golkp(1/60*s+1)
golkplead2 =

2.409 s + 144.6
-----------------------------------
6.944e-05 s^3 + 0.01007 s^2 + s

Continuous-time transfer function.
>> margin(golkplead2)
```
This system has a much better phase margin. It is slower than the system with the first lead. Let’s look at the step response with this system and then compare it with the detuned system. The phase margin is a little less than the detuned system (33.4 degrees vs. 39.3 degrees), so we can expect a little more overshoot. But the system with this new lead will be much faster than the detuned system.

```
>> gcllead = golkplead2/(1+golkplead2)

gcllead =

0.0001673 s^4 + 0.0343 s^3 + 3.865 s^2 + 144.6 s

---------------------------------------------------------------------
4.823e-09 s^6 + 1.399e-06 s^5 + 0.0004076 s^4 + 0.05444 s^3 + 4.865 s^2

+ 144.6 s

Continuous-time transfer function.
```

```
>> figure(2)
>> hold on
>> step(gcllead)
```
Actually, the system turns out to have less %OS than the detuned system, AND it is much faster. I’m guessing that this system behaves more like a pure second order, that’s why the %OS is smaller.

The controller now is $G_c = 80*(1/60*s+1) = K_p + K_d*s = K_p*(K_d/K_p*s + 1)$. So $K_p = 80$ and $K_d/K_p = 1/60$. So

$$\gg k_d = \frac{80}{60}$$

$$k_d = 1.3333$$

From here, you could de-tune this system to get more FM and reduce the %OS, again at the expense of slowing it down. But we’ve made it so fast, we have room to spare.

Also it might be that in certain applications you need $\text{ess} = 0$ for a ramp input. For that you would need to add I-control, which would de-stabilize the system. But if you add an integrator, you will also add another lead, since this is PI control. And with the previously added lead, you have a PID controller.