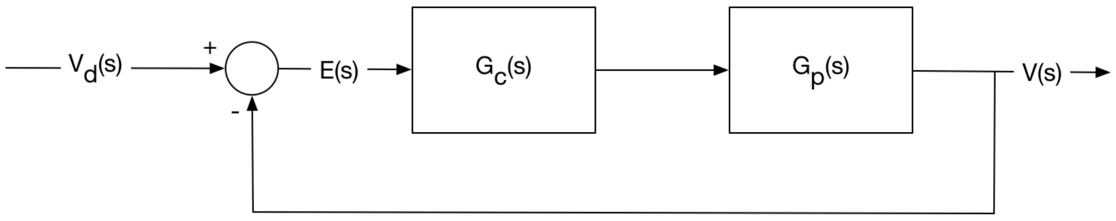
Deriving Transfer Functions for Arbitrary Block Diagrams

Disclaimer: this notebook is based on my experience using Mathematica to solve for the transfer functions of block diagrams. I don't claim that the information in this notebook represents any deep theory or formal algorithm of how to setup these problems. In other words, these strategies have worked for me (Paul) across a range of problems, but don't take them as more than that.

In this notebook we'll show you how to derive the transfer function for block diagrams of non-standard configurations. This will be useful to you when determining the poles of your system.

As background, let's look at our old friend the simple feedback control system. It doesn't really matter all that much what the blocks in this diagram represent (the algebra is still the same), but for concreteness let's assume that it represents a cruise control system where a controller block (transfer function $G_c(s)$) is controlling a car (transfer function $G_p(s)$) and our goal is to achieve a desired velocity (Laplace transform
$V_d(s)$).



We know from earlier in the semester that the transfer function of the whole system can be written in the following manner.

$$\frac{V(s)}{V_d(s)} = \frac{G_c(s)\,G_p(s)}{1+G_c(s)\,G_p(s)}$$

This is a useful formula, but we need not take it on faith (well you already saw Kat Kim's derivation if you watched that video). Next, I'll show you how to have Mathematica derive this for you. We'll setup the problem, explain what we are doing, and then solve it.

```
In[1]:= << Notation`
        Symbolize[ ⌢ ]
        Symbolize[ _ ]
```

```
In[4]:= eq1 = V[s] == E[s] Gc[s] Gp[s];
        eq2 = E[s] == Vd[s] - V[s];
```

The equations above represent relationships that must hold for the various blocks in our system. Here are some guidelines for how we wrote these down.

(1) Any block with a circle needs to balance inputs versus outputs (careful with the signs!)
(2) We need to express how the overall output of the system depends on other Laplace transforms in the system. To write down this equation we take the longest path uninterrupted by a circle node, chaining together transfer functions as we go.

Once we have these relationships, we can use Mathematica to solve for the Laplace transforms in our system (we won't need to solve for *V_d*[*s*] since it is the input). Once we have the Laplace transforms, we can divide by the Laplace transform of the input to get our transfer function.

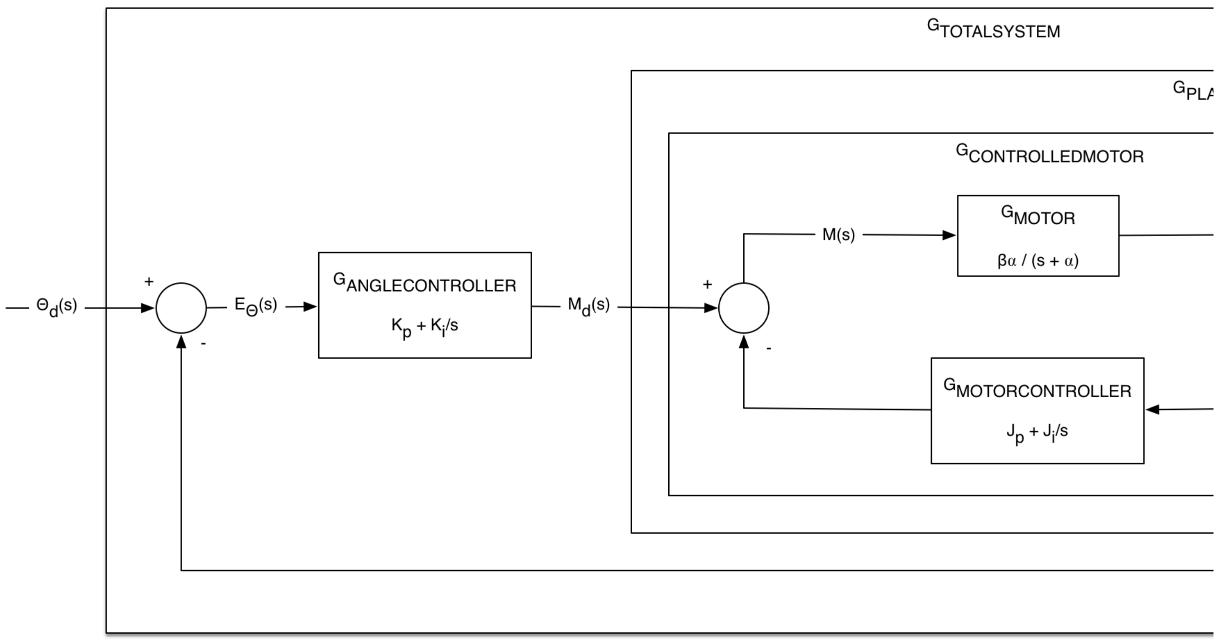
```
In[6]:= sol = Solve[{eq1, eq2}, {E[s], V[s]}][[1]]
        {Gvdtov[s] → V[s]/Vd[s] /. sol} (* this is a rule to replace Gvdtov,
        you can just extract the value by using the righthand side of the rule *)
```

Out[6]= $\left\{ \mathbb{e}[s] \to \frac{V_d[s]}{1 + G_c[s]\, G_p[s]},\ V[s] \to \frac{G_c[s]\, G_p[s]\, V_d[s]}{1 + G_c[s]\, G_p[s]} \right\}$

Out[7]= $\left\{ G_{vdtov}[s] \to \frac{G_c[s]\, G_p[s]}{1 + G_c[s]\, G_p[s]} \right\}$

Sweet! From here you could substitute the specific transfer function of your blocks, use Factor to get the transfer function as a ratio of two polynomials, and solve for the roots of the denominators to get your poles.

Let's try another one. You saw in the notebook "Rocky Don't Runaway" that you can remove the runaway behavior by adding a block in the feedback loop. Below is the diagram of the full system, but let's start out by focusing on the part labeled
"$G_{\text{CONTROLLEDMOTOR}}$".



To solve for $G_{\text{CONTROLLEDMOTOR}}$ we have to write down the equations that represent the relationships in the block diagram. We'll avoid substituting the specific transfer functions of the blocks, instead opting to work with them symbolically.

```
In[12]:= eq1 = M[s] == Md[s] - GMOTORCONTROLLER[s] V[s];
         eq2 = V[s] == M[s] GMOTOR[s];

         sol = Solve[{eq1, eq2}, {M[s], V[s]}][[1]];
         {GCONTROLLEDMOTOR[s] → Simplify[V[s] / Md[s] /. sol]}
         (* this is a rule to replace GCONTROLLEDMOTOR,
         you can just extract the value by using the righthand side of the rule *)
```

Out[15]= $\left\{G_{\text{CONTROLLEDMOTOR}}[s] \to \frac{G_{\text{MOTOR}}[s]}{1 + G_{\text{MOTOR}}[s]\, G_{\text{MOTORCONTROLLER}}[s]}\right\}$

You'll notice that this is equivalent to what was in the notebook if you divide the numerator and the denominator of that expression (the one from the other notebook) by G_{MOTOR}. For reference, the expression from that notebook was: $G_{\text{CONTROLLEDMOTOR}} = 1 / (1 / G_{\text{MOTOR}} + G_{\text{MOTORCONTROLLER}})$.

The next question to ask is how you would do this for really big systems. To show you how to do this, let's solve for the transfer function of the entire system from the Rocky Don't Runaway notebook (the diagram avove). We'll need to keep in mind the following guidelines.

From before:
(1) Any block with a circle needs to balance inputs versus outputs (careful with the signs!). (also, when expressing the inputs to the block, use the longest path uninterrupted by a circle node, chaining

together transfer functino as we go. This avoids referencing intermediate Laplace transforms unecessar ily)
(2) We need to express how the overall output of the system depends on other Laplace transforms in the system. To write down this equation we take the longest path uninterrupted by a circle node, chaining together transfer functions as we go.

Additional guidelines:
(3) Any intermediate signal that feeds back, needs to have an additional equation to define it in terms of the longest path from a circle node (you’ll see this below when we define V[s])
(4) You always need as many equations as unknown Laplace transforms, and you need to solve for them all simultaneously with one call to the Solve function.

Here’s a worked example of how you would do this for the full system. I’ve annotated important lines in the code below with the corresponding guideline that it arises from.

```
In[18]:= eq1 = θ[s] == M[s] G_MOTOR[s] G_VELOCITYTOANGLE[s]; (* guideline 2 *)
         eq2 = V[s] == M[s] G_MOTOR[s]; (* guideline 3 *)
         eq3 = M[s] == E_θ[s] G_ANGLECONTROLLER[s] - V[s] G_MOTORCONTROLLER[s];
         (* guideline 1. Notice that we use the longest path uninterrupted
          by a circle node to avoid refetencing intermediate transfer
          functions. E_θ[s]G_ANGLECONTROLLER[s] is really the same as M_d[s] *)
         eq4 = E_θ[s] == θ_d[s] - θ[s]; (* guideline 1 *)
         sol = Solve[{eq1, eq2, eq3, eq4}, {θ[s], V[s], M[s], E_θ[s]}][[1]] ;
         (* guideline 4 *)
         {G_TOTALSYSTEM[s] → θ[s]/θ_d[s] /. sol} (* this is a rule to replace G_TOTALSYSTEM,
         you can just extract the value by using the righthand side of the rule *)
```

Out[19]=
$$\left\{G_{\text{TOTALSYSTEM}}[s] \to \frac{G_{\text{ANGLECONTROLLER}}[s]\, G_{\text{MOTOR}}[s]\, G_{\text{VELOCITYTOANGLE}}[s]}{1 + G_{\text{MOTOR}}[s]\, G_{\text{MOTORCONTROLLER}}[s] + G_{\text{ANGLECONTROLLER}}[s]\, G_{\text{MOTOR}}[s]\, G_{\text{VELOCITYTOANGLE}}[s]}\right\}$$

Okay, so that doesn’t look completely unreasonable, but how can we build some confidence that we are right? One method is to just compare this with the solution derived by applying various block diagram rules in the “Rocky Don’t Runaway notebook.”

```
In[16]:= Factor[θ[s]/θ_d[s] /. sol /. {G_ANGLECONTROLLER[s] → K_p + K_i/s, G_VELOCITYTOANGLE[s] → s/(L s^2 - g),
            G_MOTORCONTROLLER[s] → J_p + J_i/s, G_MOTOR[s] → β α / (s + α)}]
```

Out[•]=
$$\frac{s\,(K_i + K_p\, s)\,\alpha\,\beta}{-g\, s^2 + L\, s^4 - g\, s\, \alpha + L\, s^3\, \alpha - g\, J_i\, \alpha\, \beta - g\, J_p\, s\, \alpha\, \beta + K_i\, s\, \alpha\, \beta + K_p\, s^2\, \alpha\, \beta + J_i\, L\, s^2\, \alpha\, \beta + J_p\, L\, s^3\, \alpha\, \beta}$$

From the other notebook we see that the transfer function is the same as the one above (the equation below is cut-and-pasted from the output of the other notebook).

$$(s\,\alpha\,\beta\,(K_i + s\,K_p)) \Big/ \left(-g\,s^2 + L\,s^4 - g\,s\,\alpha + L\,s^3\,\alpha - g\,\alpha\,\beta\,J_i + L\,s^2\,\alpha\,\beta\,J_i - g\,s\,\alpha\,\beta\,J_p + L\,s^3\,\alpha\,\beta\,J_p + s\,\alpha\,\beta\,K_i + s^2\,\alpha\,\beta\,K_p\right)$$

One cool thing about this approach is that you get some of the intermediate transfer functions for free. For instance, we are also interested in the transfer function relating the V[s] to $\theta_d[s]$, which we canb get easily from the variable “sol”.

In[21]:= $\left\{G_{\theta dtov}[s] \to \frac{V[s]}{\theta_d[s]} \text{ /. sol}\right\}$

Out[21]= $$\left\{G_{\theta dtov}[s] \to \frac{G_{\text{ANGLECONTROLLER}}[s]\,G_{\text{MOTOR}}[s]}{1 + G_{\text{MOTOR}}[s]\,G_{\text{MOTORCONTROLLER}}[s] + G_{\text{ANGLECONTROLLER}}[s]\,G_{\text{MOTOR}}[s]\,G_{\text{VELOCITYTOANGLE}}[s]}\right\}$$