Okay, so as long as I have a SS system that is observable and controllable, I can just do this pole placement thing and BAM! perfect control?

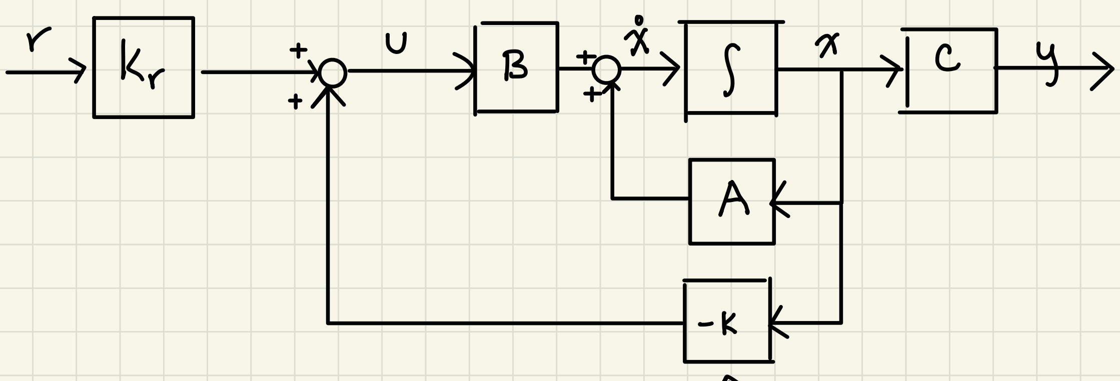
Well... yes... but also no...

The question is - where do you want to put the poles? We really only know how to select pole locations for systems of 2^{nd} order or that can be approximately 2^{nd} order. But we have seen that not all systems work like this.

Wouldn't it be EVEN BETTER if we could just decide how important our performance criteria are and get the controller to pick the Ks for us to get the poles in a good location?

Linear Quadratic Regular "LQR" is one optimal control method that is widely used

So let's go back to our block diagram we used in pole placement. We'll use the same one for LQR, but the method to find the K values will change.



↑ LQR focuses on "optimal K" values

First-what does optimal mean?

Let's imagine that COVID is over and spring break is back! We're going to plan a trip! Where should we go?

alternatives ↓	"performance" ↓	"cost" ↓
<u>Destination</u>	<u>Awesomeness</u> (1-10)	<u>100s of \$ for trip</u>
GA mountains	2	\$2
FL beach	5	\$5
Bahamas	8	\$15
European Getaway	10	\$30

Where will you go? How will you decide?

$$J = Q \cdot \frac{1}{\text{Awesomeness}} + R \cdot \$ \quad \leftarrow \text{minimize}$$

What if $Q \gg R$? $R \gg Q$? $R = Q$?

Now what does this have to do with controls? Well, let's think about what this looks like for a control system.

The x state vector tells performance

The u vector tells actuation effort (cost)

The objective function is the weighted sum of performance and cost

$$J = \int_0^\infty (\underbrace{x^T Q x}_{\text{performance}} + \underbrace{u^T R u}_{\text{cost}})\,dt + \cancel{\int_0^\infty 2x^T N u\,dt}$$

0 for us

deals w/ crossterms b/w u and x. often $\emptyset$

So let's talk about why our objective function looks this way and how it works.

First, some observations:

1) This is quadratic (x and u are squared)
2) For the math to work, Q and R are square matricies with dimensions equal to the length of x and u respectively.
3) Q & R must be positive definate (+ eigenvales)
   → convex → can use gradient methods

$\underset{1\times n}{x^T}\,\underset{n\times n}{Q}\,\underset{n\times 1}{x}$ becomes a positive scalar (or zero)

same for $\underset{1\times m}{u^T}\,\underset{m\times m}{R}\,\underset{m\times 1}{u}$

So the cost function will always give a positive number, and the goal is to minimize this

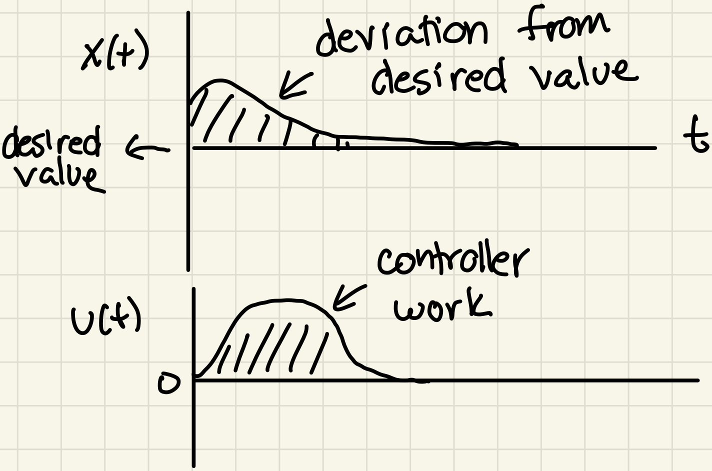
We have an optimization problem!

$$\boxed{\begin{array}{ll} \text{minimize } J & u \in \mathbb{R}^m \\ \text{such that} & \dot{x} = Ax + Bu \end{array}}$$

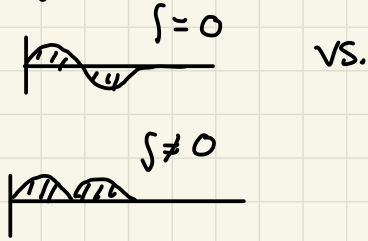
Basically, I want to find the control setup that gets the best performance for the least cost, where "best" is determined by the values in Q & R

$\int x^T Q x$ measures deviation from desired value

$\int u^T R u$ measures how much control effort is needed to get there



* as a side note the "squared" part ensures all deviation counts as ⊕ to the $\int$



when $Q >> R$ → react quickly, use lots of control effort

$R >> Q$ → control response is slow, control effort is low

control is "cheap"

ex: aircraft reacting to wind

control is expensive

→ ex: satellite thrusters need to minimize their use / fuel consumption

Okay, so we more or less get how this works! But where are my Ks?!?

$$J = \int_0^{\infty} \left(\underbrace{x^T Q x}_{\text{performance}} + \underbrace{u^T R u}_{\text{cost}} \right) dt$$

Well, let's talk about u. For our feedback controller,

$u(t) = -K x(t)$ ⇐ control law for full state feedback controller

So we need to solve our optimization problem by adjusting K

↳ it's the only thing we can affect as designers!

Solving the optimization Problem

Find $u = Kx$

to minimize

$$J = \int_0^\infty (x^T Q x + u^T R u)\,dt$$

such that

$$\dot{x} = Ax + Bu$$

We aren't going to go through the derivation to the solution to this optimization problem.

You can take my word that it is

$$u = -Kx$$

where $K = R^{-1} B^T S$

S is the solution to the Algebraic Ricatti Equation: (it is $n \times n$ and symmetric)

$$S = \begin{bmatrix} S_1 & S_c \\ S_c & S_2 \end{bmatrix}$$

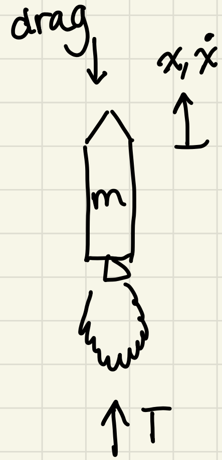
$$A^T S + SA - SBR^{-1} B^T S + Q = 0$$

So A & B are known from plant, Q & R are design choices, then we solve for S, and calculate K.

Luckily, this solver is built into Matlab for us.

```
K = lqr(A, B, Q, R)
```

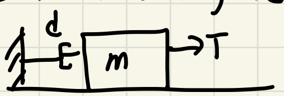
Let's look at a simple example



We have a rocket taking off. We will define position and velocity in the direction of the forcing function, thrust, and will model drag as a damping term porportional to velocity with coefficient d.

So $m\ddot{x} = T - d\dot{x}$

or, the more boring version:



$x_1 = x$

$x_2 = \dot{x}_1 = \dot{x}$

$u(t) = T$

in state space:

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -d/m \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u$$

so let's put some simple values in. $m=1$, $d=.2$

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -.2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

and let's decide on Q & R

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad R = [.01]$$

we prioritize performance

over effort

Lets play with this in Matlab, and look at some scenarios for Q and R

① - control is cheap $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $R = [.01]$

② - control is expensive $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $R = [1000]$

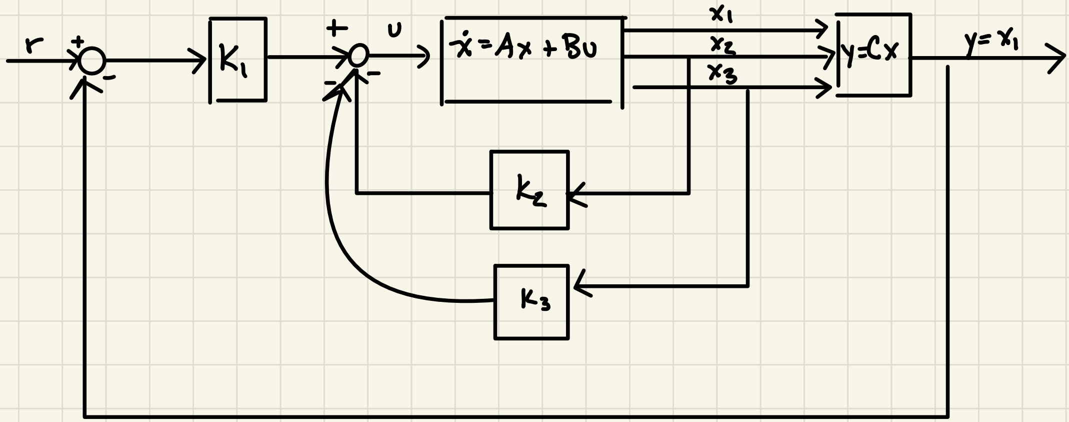
③ - only a non-zero velocity error is expensive

$$Q = \begin{bmatrix} .001 & 0 \\ 0 & 10 \end{bmatrix} \quad R = [1]$$

	K_1	K_2	CG_1	CE_2	$T_s\,x_1$	$T_s\,x_2$
①	10	10.75	627	53	~4s	~8s
②	.0316	.1229	0.032	0.0007	~27s	~46s
③	.0316	2.978	0.4314	0.0425	~378s	~579s

lqrex1.m

LQR ex 2:



$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -2 & -3 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$$D = [0]$$

we are interested in x_1, but directly control x_3

$$u = k_1(r - x_1) - k_2 x_2 - k_2 x_3$$

$$= k_1 r - (k_1 x_1 + k_2 x_2 + k_3 x_3)$$

$$K = [k_1 \quad k_2 \quad k_3)$$

$$Q = \begin{bmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{bmatrix} \qquad R = [r_c] \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

We care about x_1 the most, so

$$Q = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Now let's play with R. We will input a step and plot the x and the u.

lqrex2.m

$R = .01, 1, 100$

Notice how u changes based on the willingness to expend control effort.