State space methods

- these methods became very popular in the 1960's. They allow for simplifying the controller synthesis problem (as long as one is willing to accept relatively high order controllers)

  They also lead to the concepts of controllability (e.g. can we steer any initial condition to a desired state) and observability (can I reconstruct the inital condition from my observations?)

  Finally, state space methods lead to computationally tractable algorithms to solve many practical problems, including different versions of optimal control

- Main drawback: If improperly used, they can lead to "fragile" systems. Some of the robustness metrics (gain margin, phase margin) are harder to visualize in state space form

- Current state of the art: blend of classical & state space. We use some concepts & ideas motivated from classical control (stability margins), but solve the resulting problems using efficient state-space based methods

- Realization theory: Suppose that we are given a TF

$$G(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdot \; b_1 s + b_0}{s^n + a_{n-1} s^{n-1} \qquad a_1 s + a_0} \qquad n \geq m$$

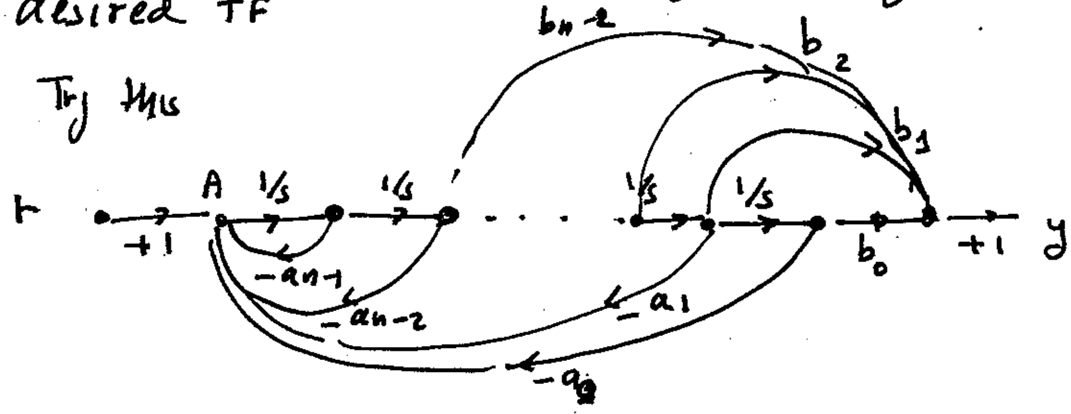
Q: How do we build a "physical" circuit that has this TF? Can we always do it? How many components do we need?

A: To answer these questions, we need to go back to the beginning of the semester & Mason's formula:

$$T_{yc} = \frac{\sum M_i \Delta_i}{\Delta}$$

The idea is to build first a signal flow graph with the desired TF

Try this



We have n loops with gains: $-\frac{a_{n-1}}{s}, -\frac{a_{n-2}}{s^2} \dots -\frac{a_1}{s^{n-1}}, -\frac{a_0}{s^n}$
all of which are touching $\Rightarrow \Delta = 1 + \frac{a_{n-1}}{s} + \frac{a_{n-2}}{s^2} + \cdot \frac{a_1}{s^{n-1}} + \frac{a_0}{s^n}$
We also have m different paths from r to y
with gains $\frac{b_0}{s^n}, \frac{b_1}{s^{n-1}} \dots \frac{b_m}{s^{n-m}}$

Note that in all cases $\Delta_i = 1$ since when we drop the node labeled A, all loops are opened

$$\Rightarrow T_{yr} = \frac{\sum M_i \Delta_i}{\Delta} = \frac{\sum \frac{b_i}{s^{n-i}}}{1 + \frac{a_{n-1}}{s} + \cdots \frac{a_0}{s^n}} = \frac{s^n\left[\frac{b_0}{s^n} + \frac{b_1}{s^{n-1}} + \cdot \frac{b_m}{s^{n-m}}\right]}{\left(s^n + a_{n-1}s^{n-1} + \cdot \, a_0\right)}$$

$$= \boxed{\frac{b_m s^m + b_{m-1} s^{m-1} \cdots \quad b_0}{s^n + a_{n-1} s^{n-1} + \cdots \quad a_0}} = G(s)$$

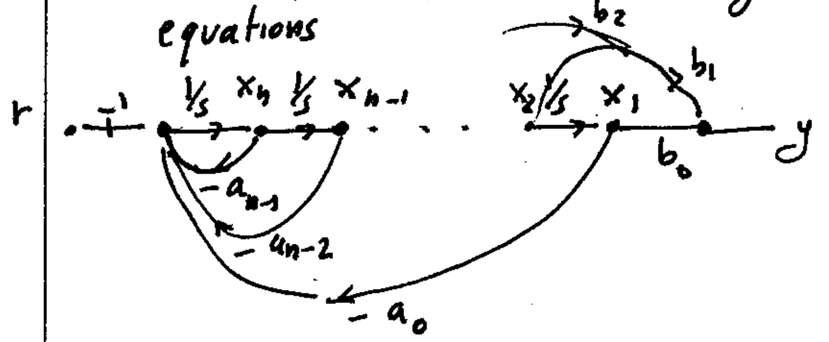
To answer the questions above:

(a) can we always build this? Yes, as long as I can build integrators & summing junctions (both doable with op-amps)

(b) How many integrators do we need?: n
(it can be shown that unless there is a pole/zero

cancellation, this is indeed the minimum number of integrators needed.

Let's label the intermediate signals and write down the equations



$y = b_0 x_1 + b_1 x_2 + \cdots b_m x_{m+1}$

$x_1 = \frac{1}{s} x_2 \quad // \quad x_2 = s x_1$

$x_2 = 1/s\, x_3 \quad // \quad x_3 = s x_2$

$\xrightarrow[\text{domain}]{\text{time}}$

$\dot{x}_1 = x_2$

$\dot{x}_2 = x_3$

$\vdots$

$s x_n = r - a_{n-1} x_n \cdots - a_0 x_1$

$\dot{x}_n = -a_{n-1} x_n \cdot -a_0 x_1 + r$

Writing the time domain equations in matrix form yields:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdot & \cdot & 0 \\ 0 & 0 & 1 & & & 0 \\ & & & \ddots & & 0 \\ 0 & 0 & 0 & & & 1 \\ -a_0 & -a_1 & & & & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \cdot \\ 1 \end{bmatrix} r$$

$$y = \begin{bmatrix} b_0 & b_1 & \cdots & b_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

or, in compact matrix notation

$$\dot{x} = Ax + Br$$
$$y = Cx + \overset{}{\cancel{D}}_{0} r$$

(Here A is an $n \times n$ matrix
B is $n \times 1$
C is $1 \times n$)

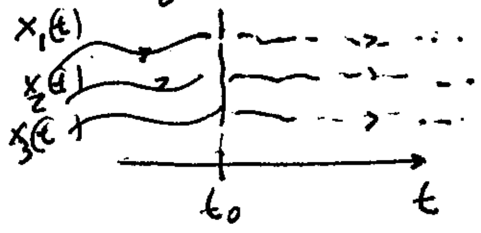
This is an example of a state space model

the vector x is called the "state" of the system

The name comes from the fact that it contains all the information that I need to predict the future:

if we know $x(t_0)$ and $r(t) \rightarrow x(t)$ is completely determined

In other words, $x(t_0)$ contains all the past information of the system



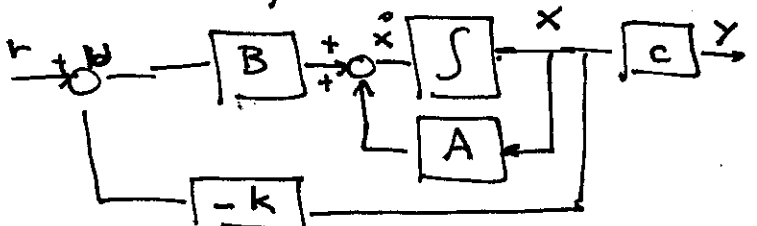
The specific realization above with $A = \begin{bmatrix} 0 & 1 & \cdots \\ 0 & 0 & 1 \\ -a_0 & & -a_{n-1} \end{bmatrix}$ $B = \begin{bmatrix} 0 \\ \vdots \\ 1 \end{bmatrix}$

$C = [b_0 \cdots b_{n-1}]$

is called "controllable canonical form" for reasons that will see next.

- Q: How does "state-space" help in designing controllers?
- A: If you have access to the states, then, under certain conditions, you can place your closed loop poles at any arbitrary location

Assume first that your system is in controllable canonical form



$A = \begin{bmatrix} 0 & 1 & & \\ 0 & 0 & & 1 \\ -a_0 & \cdots & & -a_{n-1} \end{bmatrix}$ $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$C = [b_0 \cdot b_{n-1}]$

and consider a control law $u = -kx$

with $k = [k_1 \cdots k_n]$ $\Rightarrow$ $u = -k_1 x_1 - k_2 x_2 \cdots - k_n x_n$

$$\left.\begin{aligned} \dot{x} &= Ax + B[r - kx] = \overbrace{(A - Bk)}^{A_{cl}}x + Br \\ y &= Cx \qquad\qquad\quad = Cx \end{aligned}\right\} \text{closed loop equations}$$

Using the explicit expressions for A, B, k yields

$$A_{cl} = \begin{bmatrix} 0 & 1 & \cdot & \cdot & 0 \\ 0 & 0 & 1 & & \cdot \\ & & & & \\ -a_0 & -a_1 & & & -a_{n-1} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \\ 1 \end{bmatrix} [k_1 \cdot k_n] = \begin{bmatrix} 0 & 1 & - & - & 0 \\ 0 & 0 & 1 & \cdots & - \\ & & & & \\ & & & & 1 \\ -(a_0+k_1) & \cdots & & & -(a_{n-1}+k_n) \end{bmatrix}$$

So, the net effect of closing the loop is $a_i \rightarrow (a_i - k_{i+1})$

Since the closed loop matrix A_{cl} has the same form as before, it follows that the new transfer function is:

$$T_{cl} = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdot \quad b_1 s + b_0}{s^n + (a_{n-1}+k_n)s^{n-1} + \cdot \cdot \quad (a_1+k_2)s + (a_0+k_1)}$$

So, the effect of k is to change the coefficients of the characteristic polynomial. Since k_i are arbitrary (and independent), we can always choose k_i to get any desired closed loop poles

Suppose that we want poles at $p_1, p_2 \cdot \cdot \quad p_n$

$\Rightarrow$ char polynomial should look like

$$\varphi(s) = (s-p_1)(s-p_2) \cdot \cdot (s-p_n) = s^n + \alpha_{n-1}s^{n-1} + \cdot \quad \alpha_1 s + \alpha_0$$

so we need $\quad a_{n-1} + k_n = \alpha_{n-1} \quad \Rightarrow \quad k_n = \alpha_{n-1} - a_{n-1}$

$$k_{n-1} = \alpha_{n-2} - a_{n-2}$$

$$\vdots$$

$$k_1 = \alpha_0 - a_0$$

This is a special case of <u>Ackermann's formula</u>

Matlab's commands

$k = \text{place}(A, B, P)$ ← vector with the desired closed loop poles

or $k = \text{acker}(A, B, P)$ ← numerically unstable for high order systems

Let $M_c = [B \; AB \cdot \cdot \; A^{n-1}B]$

Ackermann's formula transforms from any realization to the canonical one, computes the gains and transforms back

$$K = [0 \; . \; . \; 0 \; 1] M_c^{-1} [A^n + \alpha_{n-1}A^{n-1} + \quad \cdot \; \alpha_1 A + \alpha_0 I]$$

where α_i are the coeff of the desired char. equation

Note that for Ackermann's formula to work, we need M_c to be invertible, that is

$$\text{rank}\begin{bmatrix} B & AB & \cdot & A^{n-1}B \end{bmatrix} = n$$

this is precisely the condition for the pair (A, b) to be controllable (that is, any arbitrary initial condition can be steered to the origin)