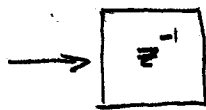
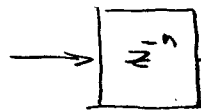
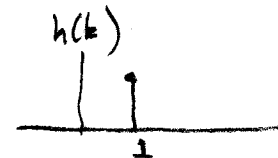


3.4 Simulation Diagrams

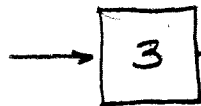
elements



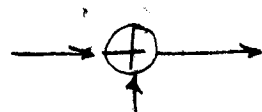
one step delay



n step delay



gain

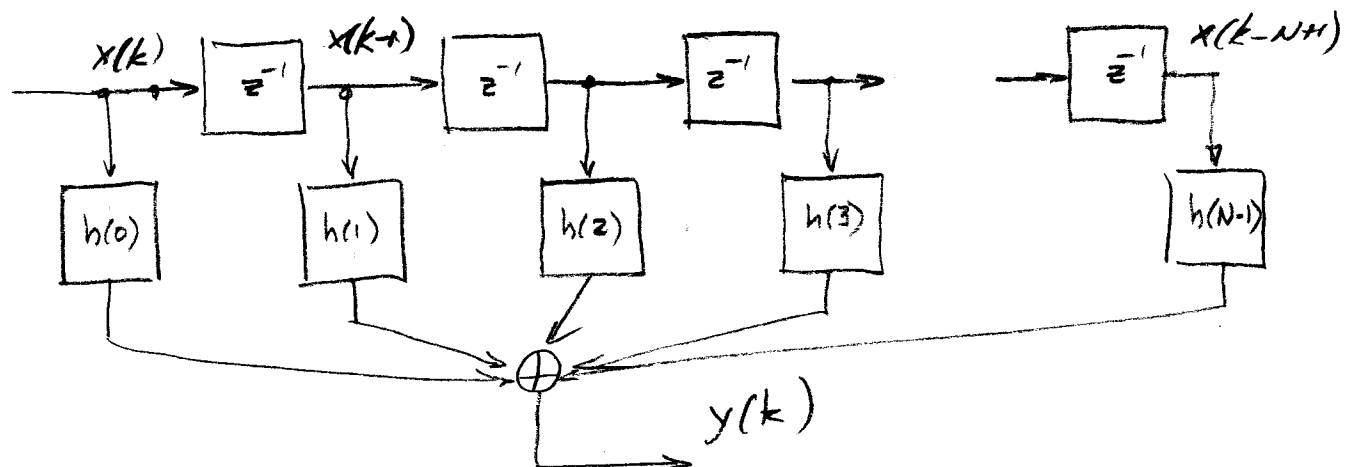


summing point



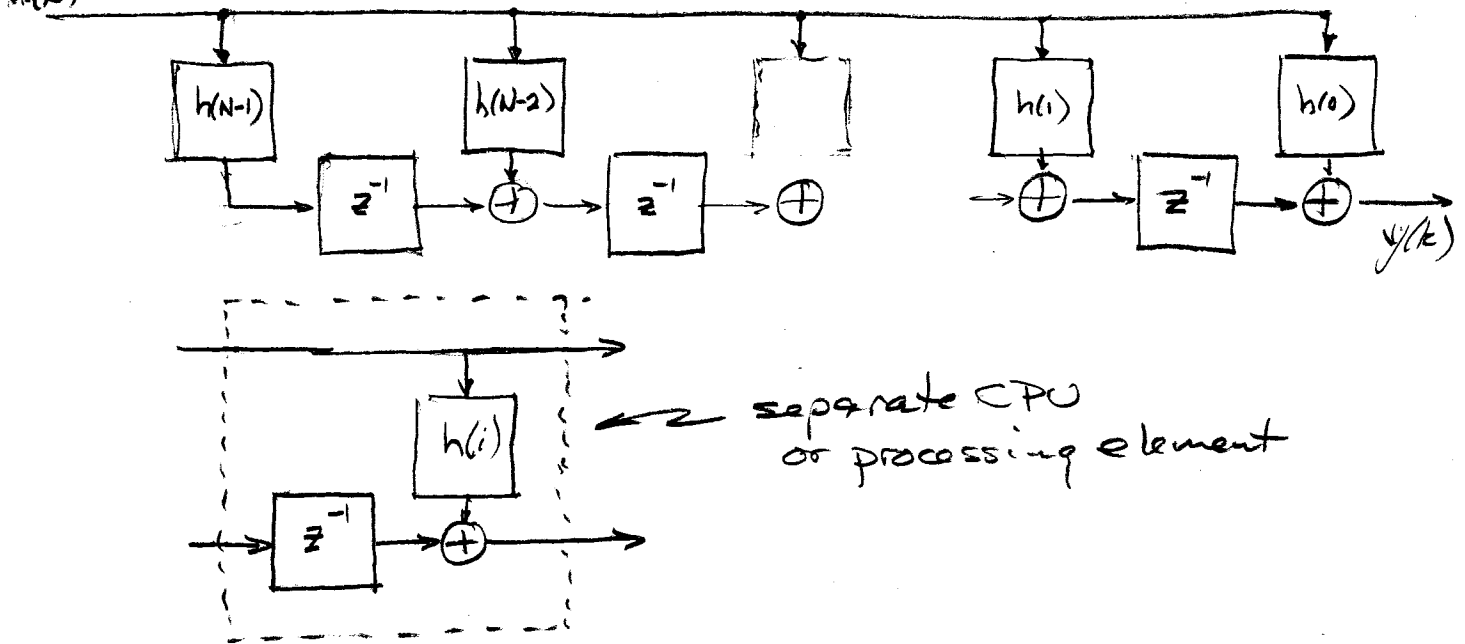
simulation diagrams describe system and suggest implementation

FIR Systems - TDL form



good for adaptive filters, but lots of adds in one place

FIR Systems - Pipeline Form



IIR (Recursive) Systems

- ΔE 's use past values of output in calculating present values, hence have IIR.
- use second order as example. want a structure which implements:

$$y(k) + a_1 y(k-1) + a_0 y(k-2) = b_2 x(k) + b_1 x(k-1) + b_0 x(k-2)$$

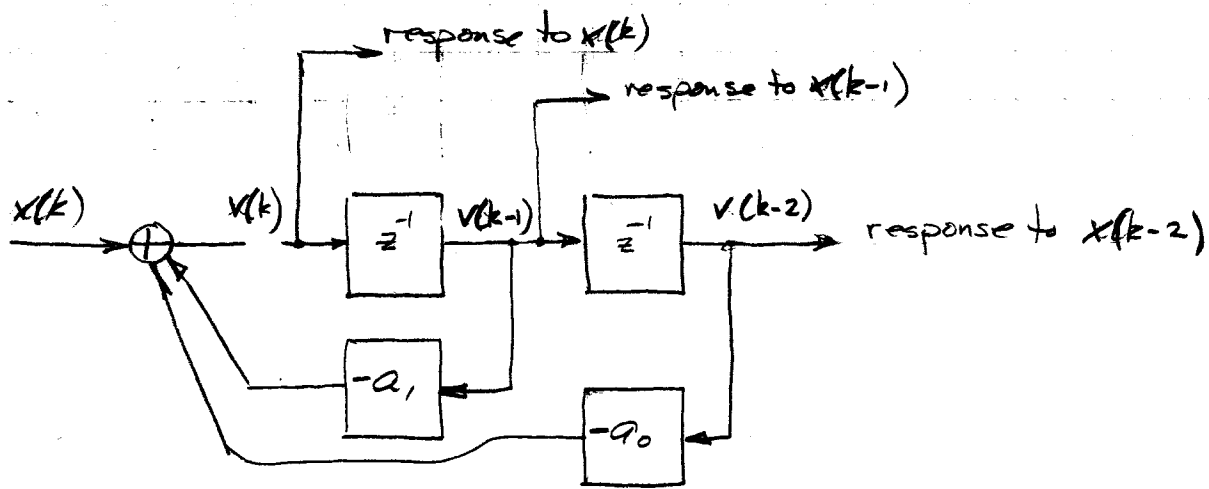
- sneak up on problem. first consider an associated system

$$v(k) + a_1 v(k-1) + a_0 v(k-2) = x(k)$$

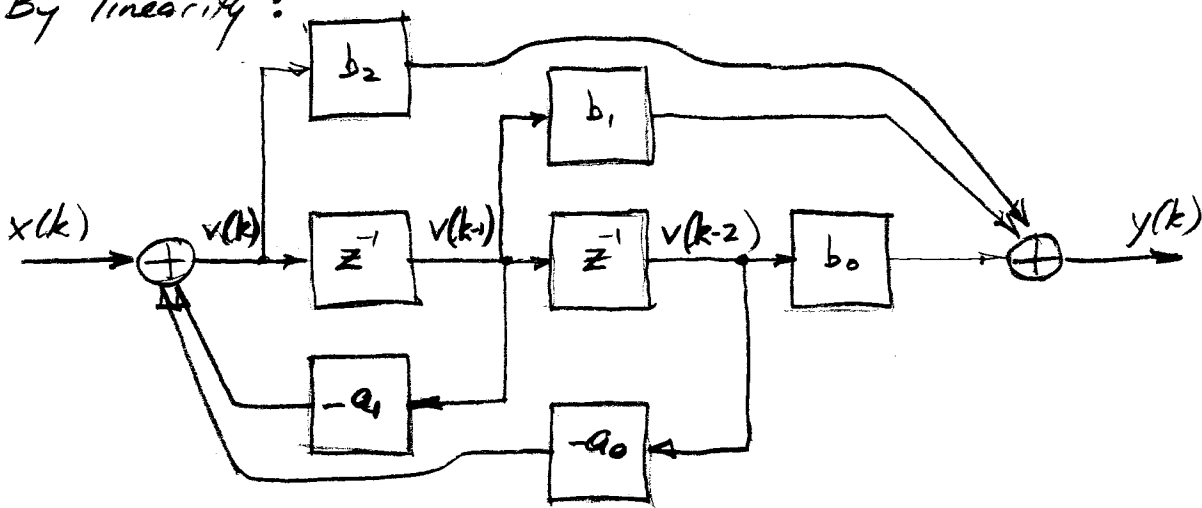
OR

$$v(k) = -a_1 v(k-1) - a_0 v(k-2) + x(k)$$

Trick: assume you have $v(k)$; make $v(k-1)$ etc;
then make $v(k)$.

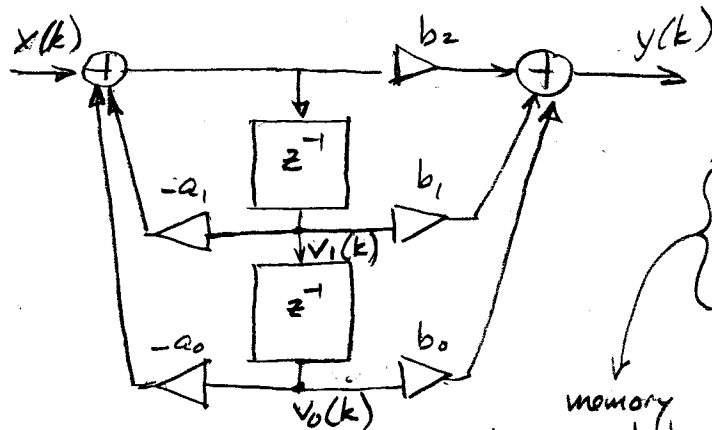


- But we want response to $b_2 x(k) + b_1 x(k-1) + b_0 x(k-2)$.
By linearity:



Note delays are clocked, simultaneous update

- Implement as sample & holds, or as elements in an array if done digitally.
- IIR digital filters used to mimic analog filters
- also drawn $x(k)$



Note definition

$$\begin{cases} v_0(k) = v(k-2) \\ v_1(k) = v(k-1) \end{cases}$$

$v_2(k) = v(k)$

memory
or state
variable

- digital filter repeat

get x ;

$v_2 := u - a_1 * v_1 - a_0 * v_0$;

$y := b_2 * v_2 + b_1 * v_1 + b_0 * v_0$;

$v_0 := v_1$; {update for next time}

$v_1 := v_2$

until doomsday;

- initial conditions for y can be translated to initial values stored in delay devices. But digital filters normally start at rest (all IC's = 0) so not a problem.

Simulation Diagrams - Continuous Time

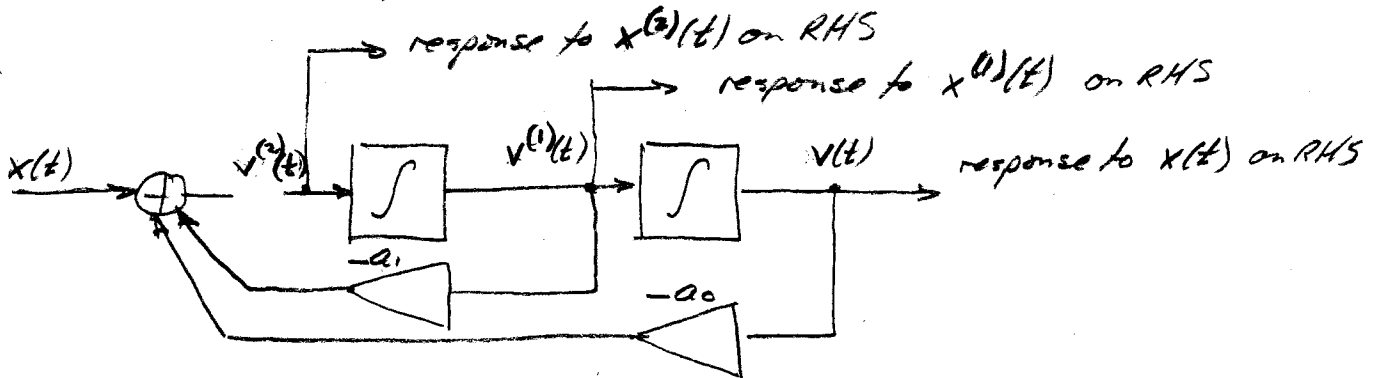
- Want a structure which implements:

$$y^{(n)} + a_{n-1} y^{(n-1)} + \dots + a_0 y = b_n x^{(n)} + b_{n-1} x^{(n-1)} + \dots + b_0 x$$

- Again, sneak up on it with an associated system:

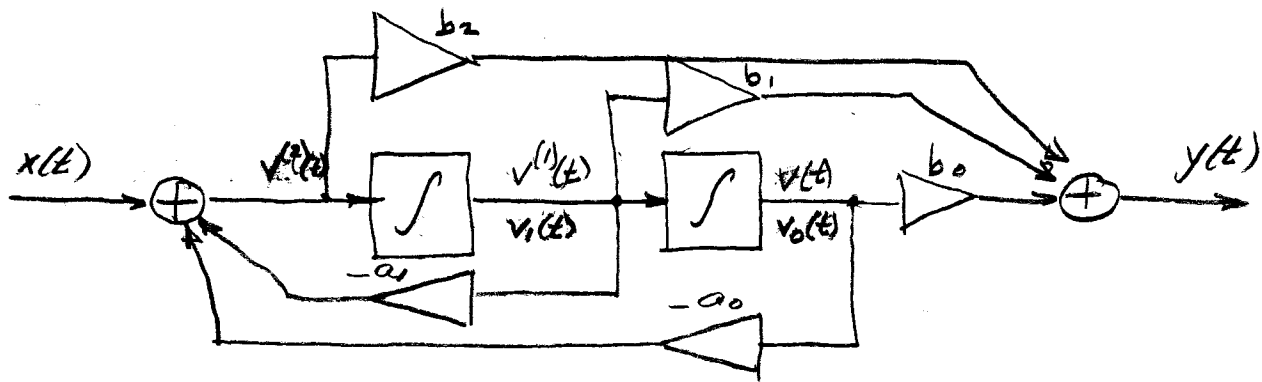
$$v^{(n)} + a_{n-1} v^{(n-1)} + \dots + a_0 v = x$$

$$v^{(n)} = -a_{n-1} v^{(n-1)} - \dots - a_0 v + x$$

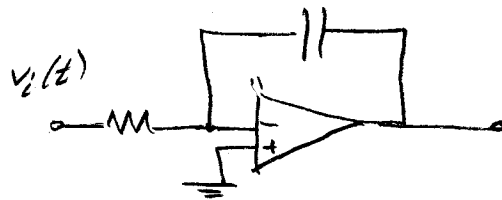


- $y(t)$ is the response to $b_2 x^{(2)} + b_1 x^{(1)} + b_0 x$, so

$$y(t) = b_2 v^{(2)}(t) + b_1 v^{(1)}(t) + b_0 v(t)$$



- Implement with op amps for integrators



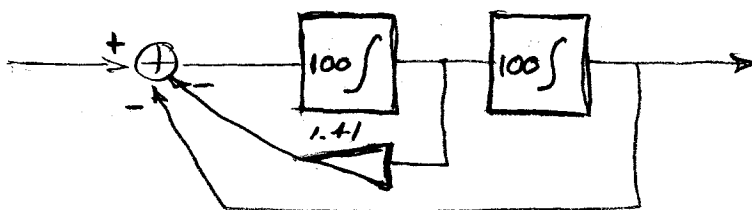
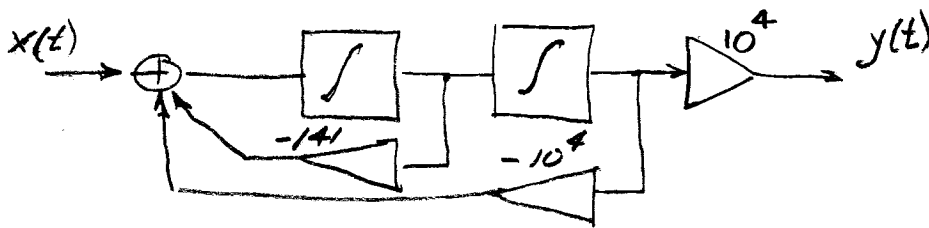
$$v_0(t) = v_0(0^-) - \frac{1}{RC} \int_0^t v_1(\alpha) d\alpha$$

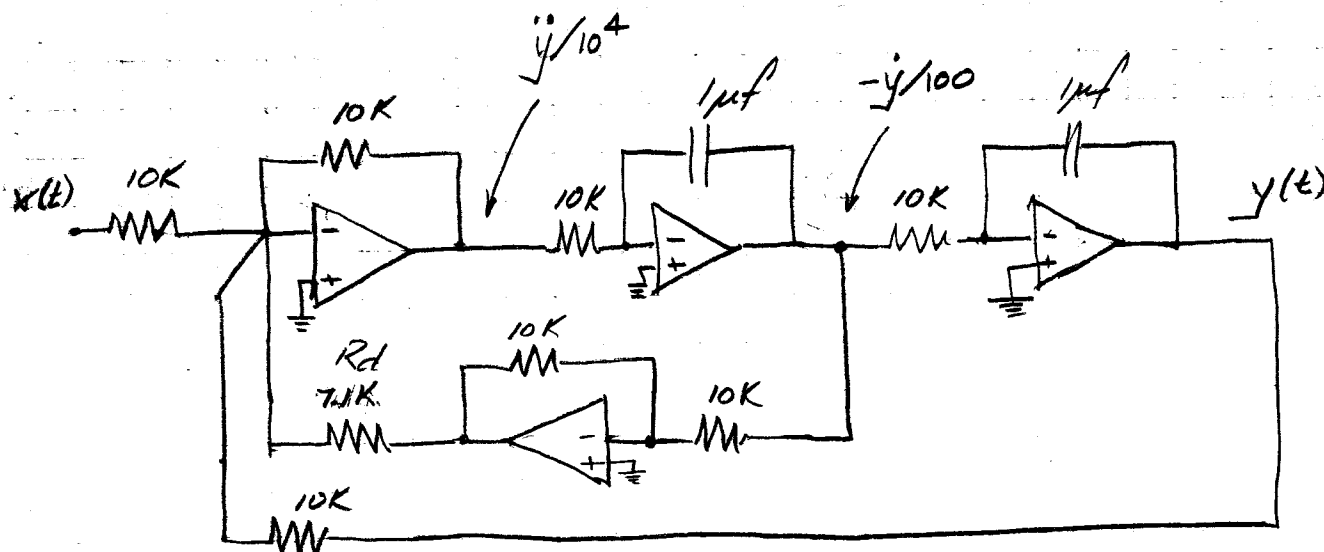
result is a "state variable" active filter.

example implement the lowpass filter

$$\ddot{y} + 141\dot{y} + 10^4 y = 10^4 x$$

$$(\omega_c = 100 \text{ rad/s})$$





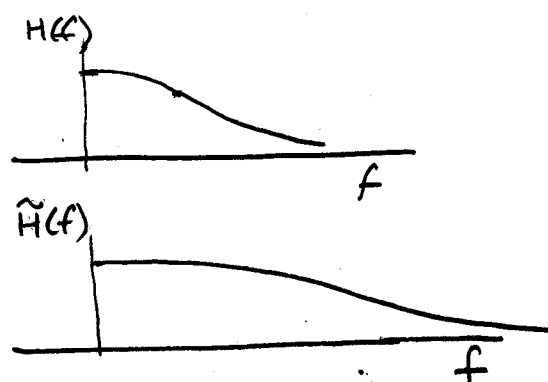
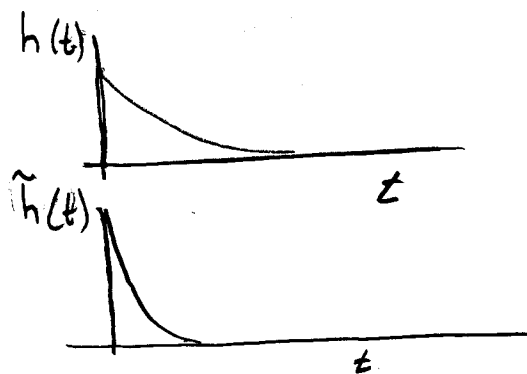
An active filter in state variable form.

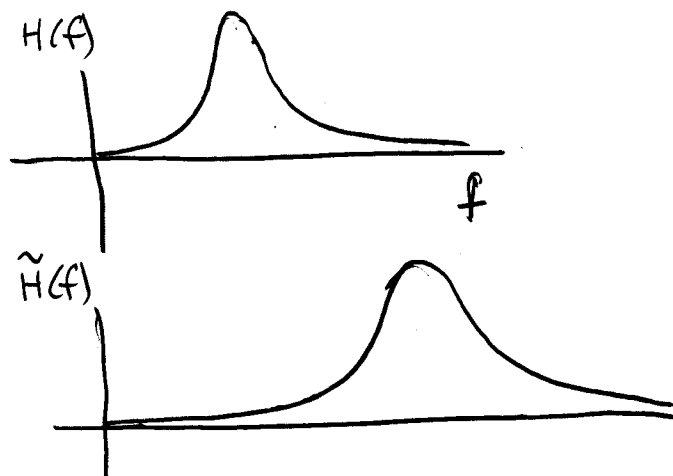
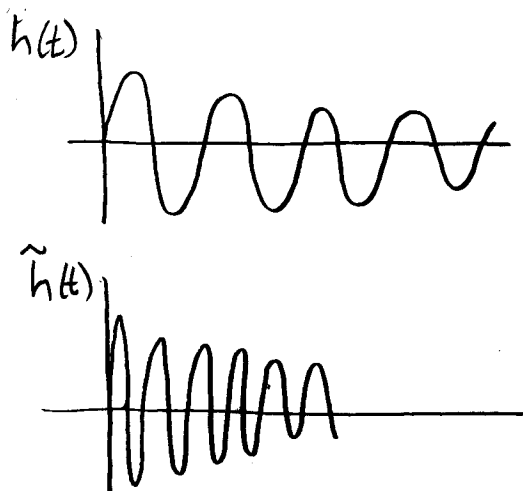
Note — reduce damping by increasing R_d

— for a resonant bandpass, use large R_d , use $\dot{y}$ as output.

• Time scaling in simulation diagrams

- Suppose you have a good design for a filter or control system, and you want to adapt it to a similar problem on a different time scale. Just stretch or compress the response. [Forward ref: this scales the frequency response in inverse proportion].

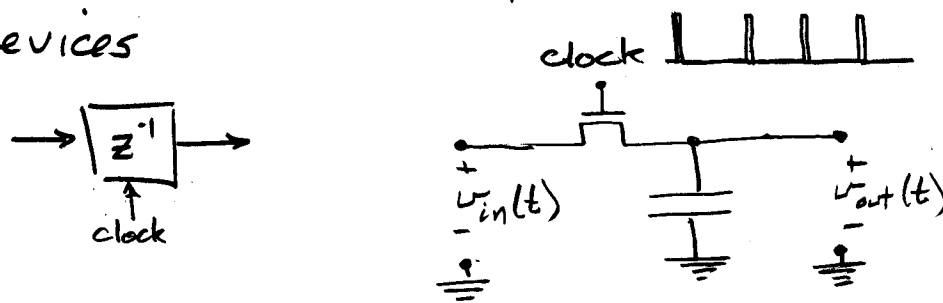




- For discrete-time systems, it's easy to change the time scale:

$$y(k) + a_{n-1}y(k-1) + \dots \text{ etc}$$

Just change the sample spacing — change the clock rate to the sample-and-hold delay devices



This permits the design of analog discrete-time filters, such as CCD filters or switched cap filters, in which the bandwidth can be changed by changing the clock frequency.

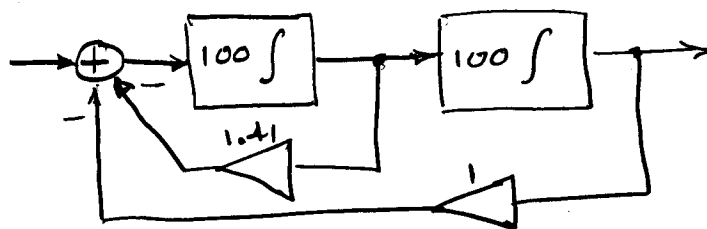
In DSP, just change sampling freq and loop timing

- In continuous time systems, it's a little more complex.
Suppose scaled time, original time are related by
 $\tilde{t} = r t$ (speed up if $r > 1$)

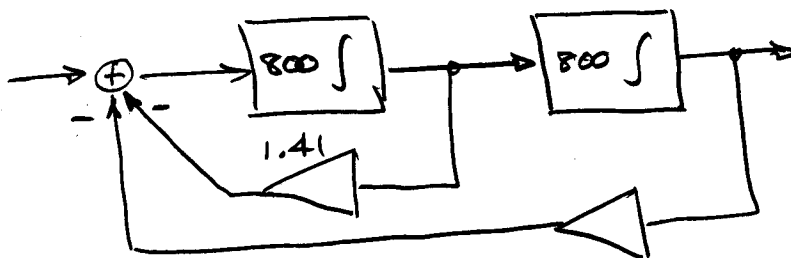
Then $\frac{d}{d\tilde{t}} = \frac{1}{r} \frac{d}{dt}$, $\int \bullet d\tilde{t} = r \int \bullet dt$

so replace $\rightarrow \boxed{\int} \rightarrow$ with $\rightarrow \boxed{r \int} \rightarrow$

Example: Recall the lowpass

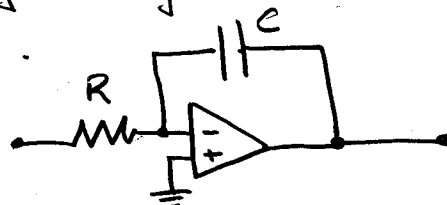


3dB point
100 rad/s



3dB point
800 rad/s

just change the gains on the integrators



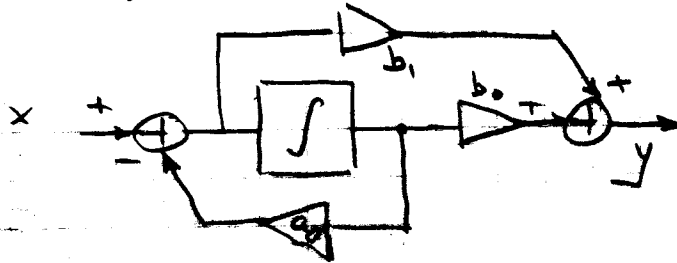
$$\text{gain} = -\frac{1}{RC}$$

- An alternative way to obtain a discrete time equivalent of a continuous time system:

- Recall that the impulse invariance method works for any LTI system

$$h_d(k) = T h_c(kT)$$

- For finite order systems, here's another way:



The essential dynamic element is the integrator

Replace the integrator with a discrete-time equivalent, such as

