BINGHAMTON
UNIVERSITY

State University of New York

EECE 301
Signals & Systems

Prof. Mark Fowler

Note Set 39

Using a D-T System to Simulate a C-T System

Simulation of CT Systems via DT Systems

Sometimes we are designing and building a DT system "on its own terms"… meaning we are required to design it to meet certain specs expressed directly in terms of the "DT world"

However, when we are trying to design or analyze a CT system we generally want to use computers to help assess the design. Thus, it would be helpful if we could use DT system ideas to ***simulate*** a given CT LTI system.

Although it is possible to do this there are some pitfalls…
So in this lecture we'll take a look at some of these ideas.

We'll only touch on this topic… Further study in DSP is needed to learn the details!

For guitarists… all those guitar amp simulators (like "Line 6") use these ideas to make boxes that simulate classic tube guitar amps (but they also have to simulate **Non**-Linearities!

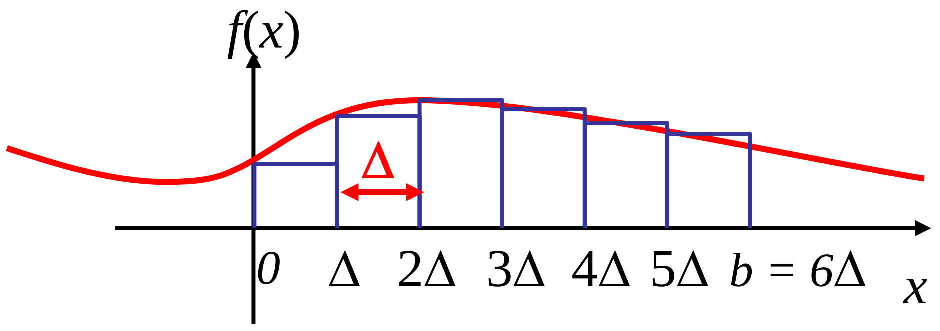
Method #1: Approximating C-T Conv w/ D-T Conv

We've seen that a CT system can be described via convolution.

We can use D-T convolution to numerically approximate a C-T convolution:

- **C-T Convolution involves an Integral**
- **D-T Convolution involves a Summation**

Recall Calc I… **Integration was *defined* as the limit of a series of Summations**



$$\boxed{\int_0^b f(x)dx \approx \sum_{i=0}^{5} f(i\Delta)\Delta}$$

So… to **approximately** compute the integral we only need **Samples** of the function!

Obviously… to be reasonably accurate we need Δ quite small

So apply this idea to C-T convolution:

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

$$\approx \sum_{i=-\infty}^{\infty} x(iT)h(t-iT)T$$

Now… we'd be happy to get closely spaced samples of the output… so replace *t* by *nT*

$$\boxed{y(nT) \approx \sum_{i=-\infty}^{\infty} x(iT)h(nT-iT)T = \sum_{i=-\infty}^{\infty} x[i]h[n-i]T = x[n]*\big(Th[n]\big)}$$

D-T Convolution!

This result says that if

- *h*(*t*) is the C-T system's impulse response
- *T* is the spacing between time samples in the D-T simulation

Then the equivalent D-T system has an impulse response given by:

$$\boxed{h[n] = Th(nT)}$$

"Impulse Invariance Method"

Choice of T (Fs = 1/T)

- We are sampling *h*(*t*)… need to ensure insignificant aliasing.
- ***Usually OK if***…
  - signal is sampled to minimize acceptable real-world rates ***and***
  - the upper edge of passband of the CT system lies sufficiently below Fs/2
  - So… **highpass filters are not suitable for this method!**

So… how do we use this DT *h*[*n*]???

We can <u>truncate</u> this to a finite length…

- Make sure you keep enough to capture its significant part
- May apply a window to taper off at the end (but usually not front!)

A Little Trick…

- If $h(t)$ is discontinuous… replace $h[0]$ w/ half its value!

$$\boxed{h[0] = Th(0)/2}$$

Check its frequency response to see if compares well to CT system's

H=freqz(h,1,w)

Implement the simulating DT system as an <u>FIR filter</u>

y = filter(h,1,x)

Drawbacks to this method

1. Can't handle highpass-type passbands
2. Can result in very long FIR filters
   - Computationally Complex!

Method #2: Approximating C-T Freq Resp w/ D-T Freq Resp

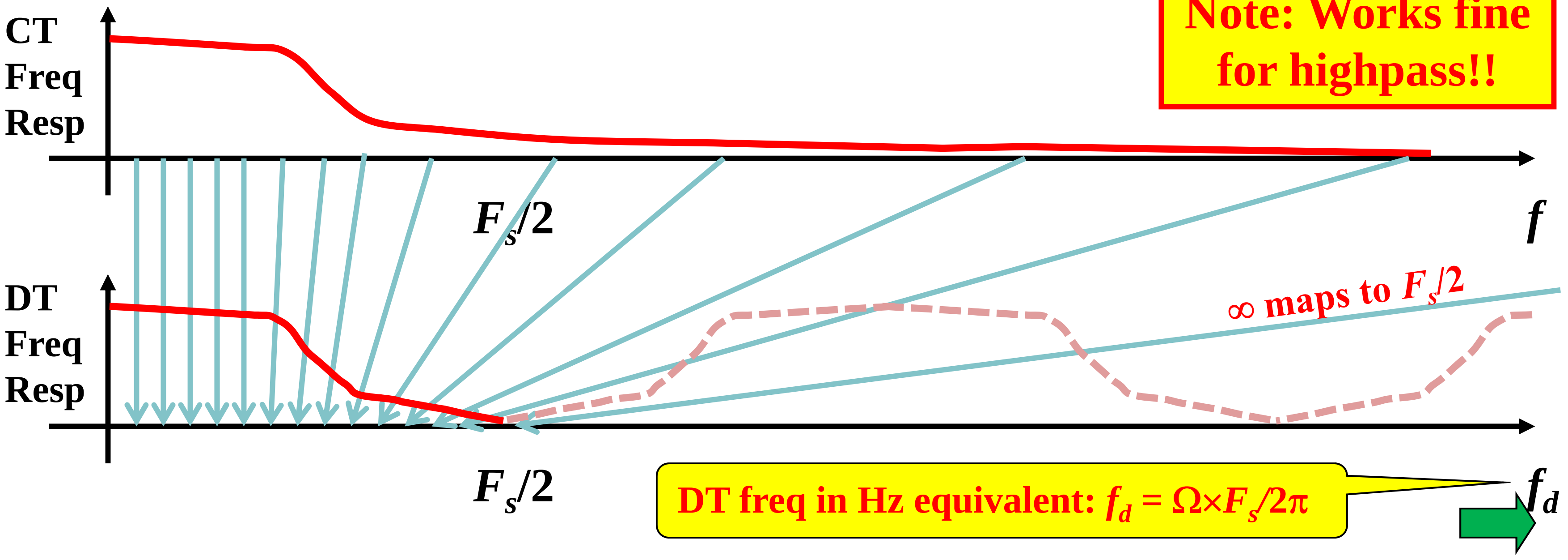
Instead of trying to match the impulse response we'll now try to match the Frequency Response….

Sampling the impulse response causes aliasing in the frequency response…
That is why this method fails for highpass type passbands!

But… even for non-highpass systems the frequency response "spreads" out over the whole –∞ to ∞ range…
And the parts outside the $–F_s/2$ to $F_s/2$ range get aliased back into that range.

So… we are going to pull a slick math trick to "squish" the whole –∞ to ∞ range down into the $–F_s/2$ to $F_s/2$ range… Something ***like*** this:

Note: Works fine for highpass!!



Without deriving it… a mapping called the **"Bilinear Transform"** does this!

The bilinear mapping says… replace s in the CT $H(s)$ with: $\boxed{s = \frac{2}{T}\frac{z-1}{z+1}}$

Not only does this "squish" the frequency the way we want it… but it also transforms a **rational $H(s)$** $\rightarrow$ **rational $H(z)$**…

What we have from LTI system described by a Differential Equation

What we have from LTI system described by a Difference Equation

An IIR DT System!!!

$$\boxed{H(z) = \frac{B(z)}{A(z)}}$$

Check its frequency response to see if compares well to CT system's

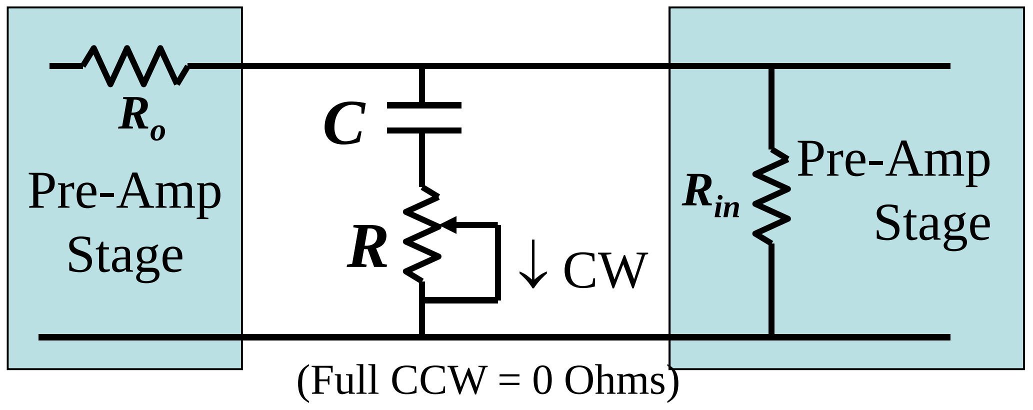
H=freqz(b,a,w)

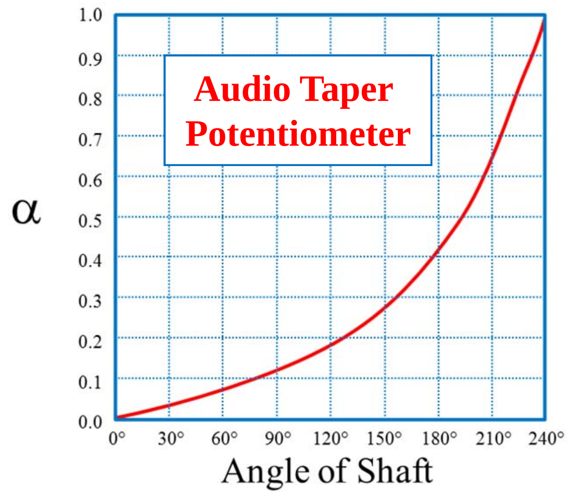
Implement the simulating DT system as an FIR filter

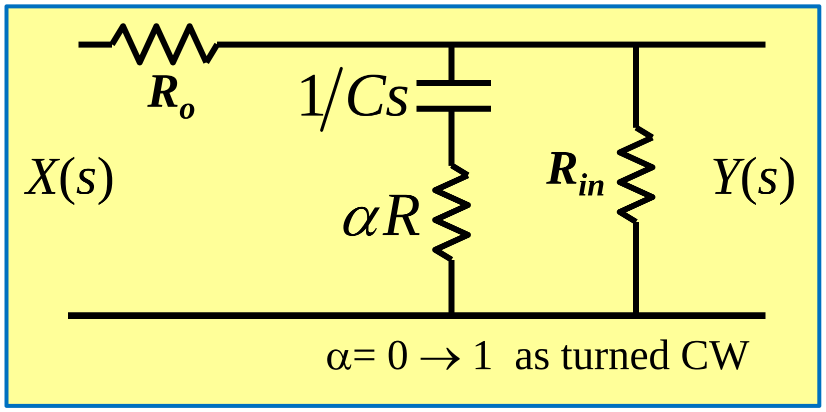
y = filter(b,a,x)

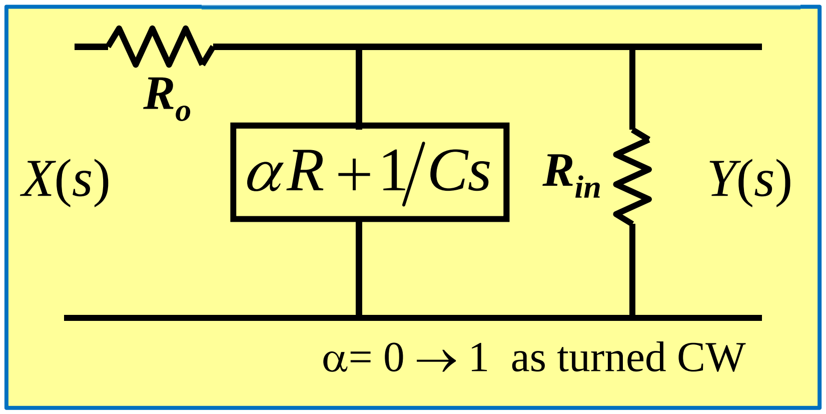
Example: Guitar Amp Simple Single Tone Control

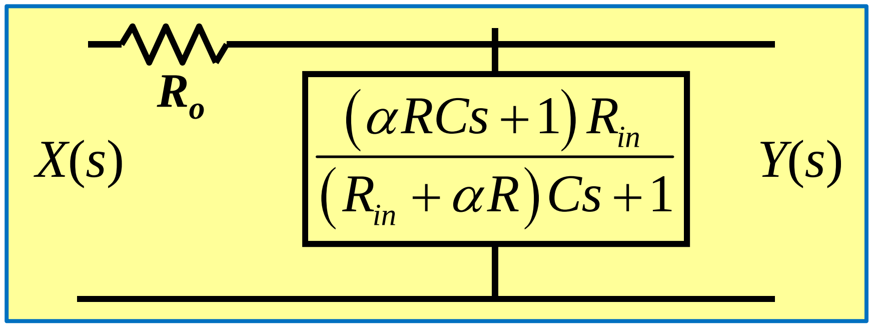
Circuit structure from book *Electronics for Guitarists* by D. J. Dailey, Springer 2011.

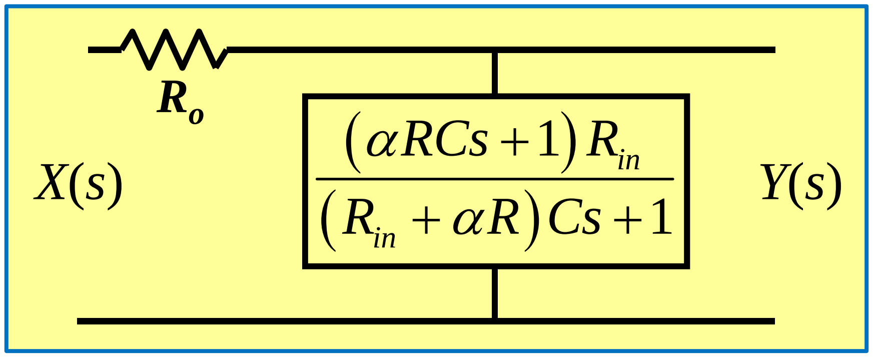












Now do voltage divider:

$$Y(s)=\frac{\dfrac{(\alpha RCs+1)R_{in}}{\boxed{(R_{in}+\alpha R)Cs+1}}}{\dfrac{(\alpha RCs+1)R_{in}}{\boxed{(R_{in}+\alpha R)Cs+1}}+R_o}X(s)$$

$$H(s)=\frac{(\alpha RCs+1)R_{in}}{(\alpha RCs+1)R_{in}+R_o\left[(R_{in}+\alpha R)Cs+1\right]}=\frac{(\alpha RCs+1)}{(\alpha RCs+1)+R_o/R_{in}\left[(R_{in}+\alpha R)Cs+1\right]}$$

$$H(s)=\frac{(\alpha RCs+1)}{(\alpha RCs+1)+(R_o+\alpha RR_o/R_{in})Cs+R_o/R_{in}}$$
$$=\frac{(\alpha RCs+1)}{(R_o+\alpha R(R_o/R_{in}+1))Cs+R_o/R_{in}+1}$$

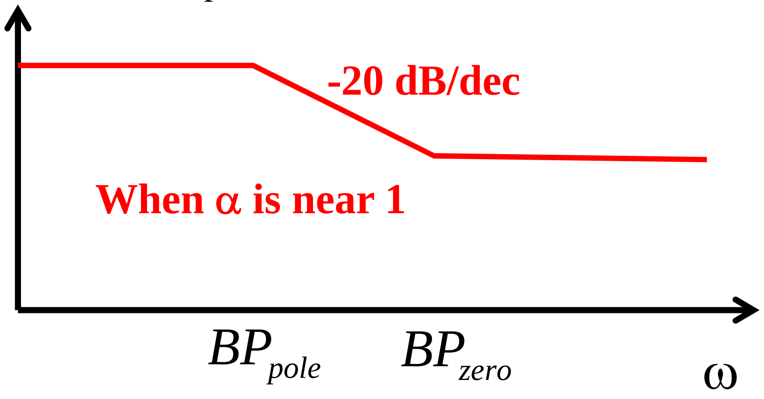
$$H(s)=\frac{(\alpha RCs+1)}{(R_o+\alpha R(R_o/R_{in}+1))Cs+R_o/R_{in}+1}=\frac{1}{R_o/R_{in}+1}\frac{(\alpha RCs+1)}{(R_o+\alpha R)Cs+1}$$

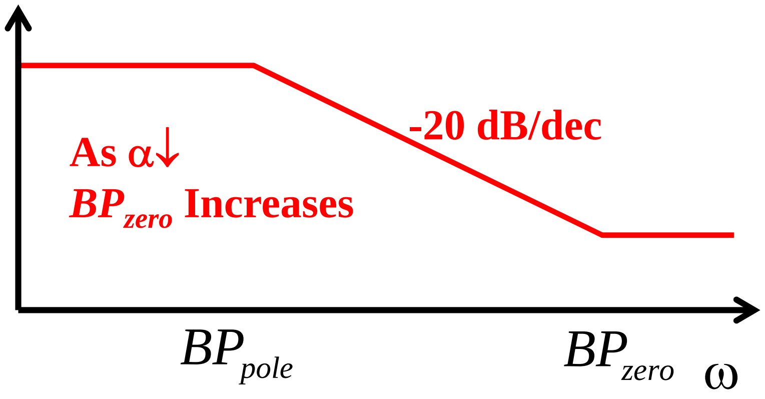
$$\boxed{H(s)=\left(\frac{R_{in}}{R_o+R_{in}}\right)\left[\frac{(\alpha RCs+1)}{(R_o+\alpha R)Cs+1}\right]}$$

One Zero & One Pole

$$BP_{zero}=1/\alpha RC$$
$$BP_{pole}=1/(R_o+\alpha R)C$$

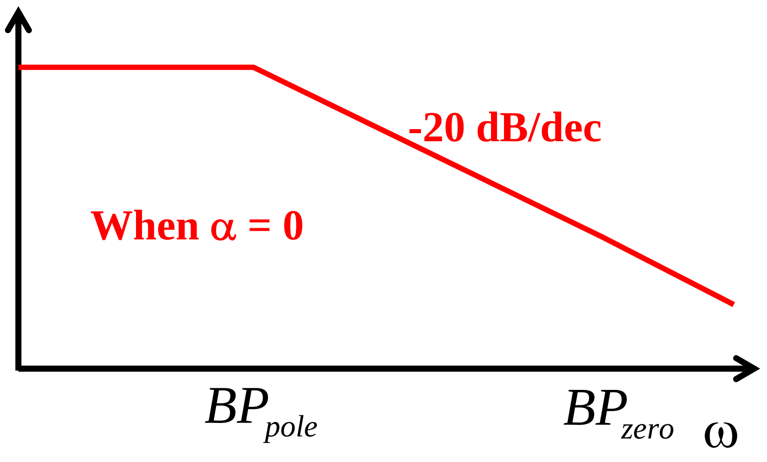
Note that $BP_{pole} < BP_{zero}$ so you have this shape:

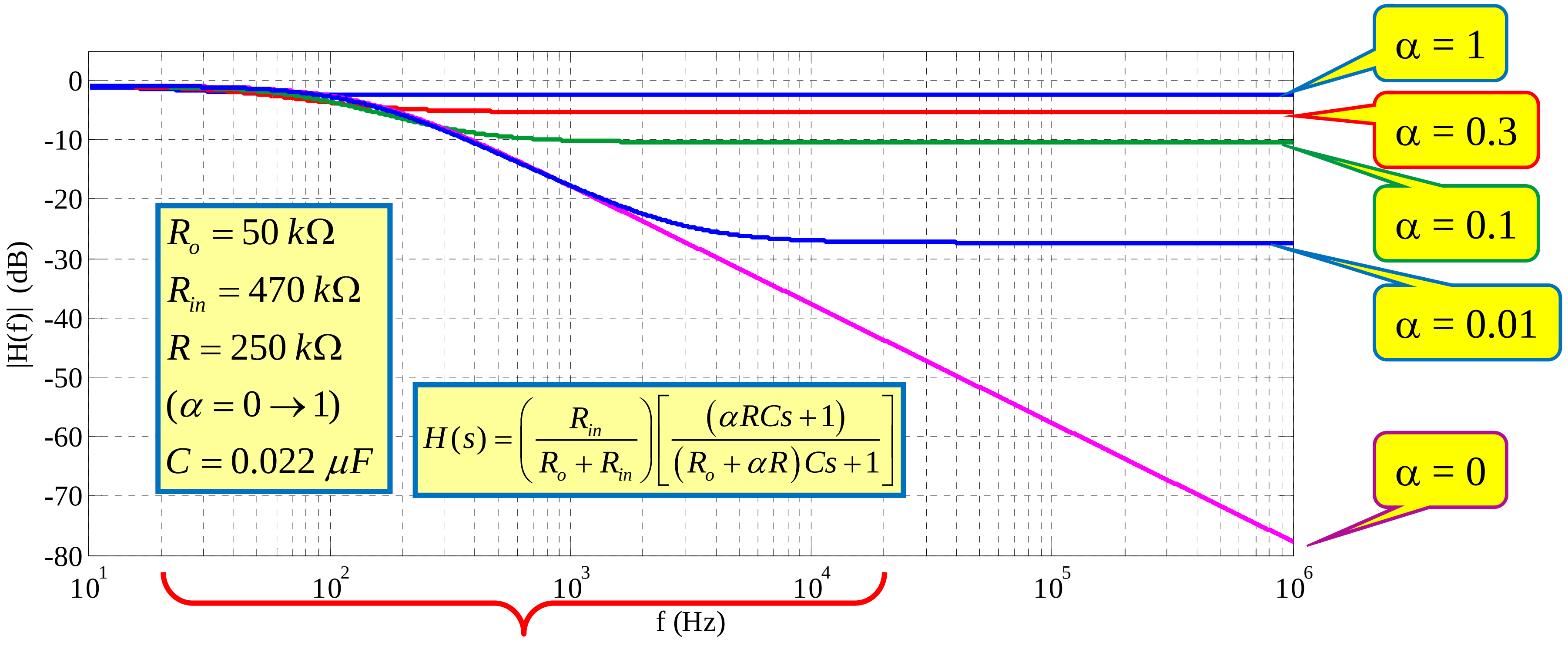




When α = 0 there is no zero:

$$\boxed{H(s)=\left(\frac{R_{in}}{R_o+R_{in}}\right)\left[\frac{1}{R_oCs+1}\right]}$$





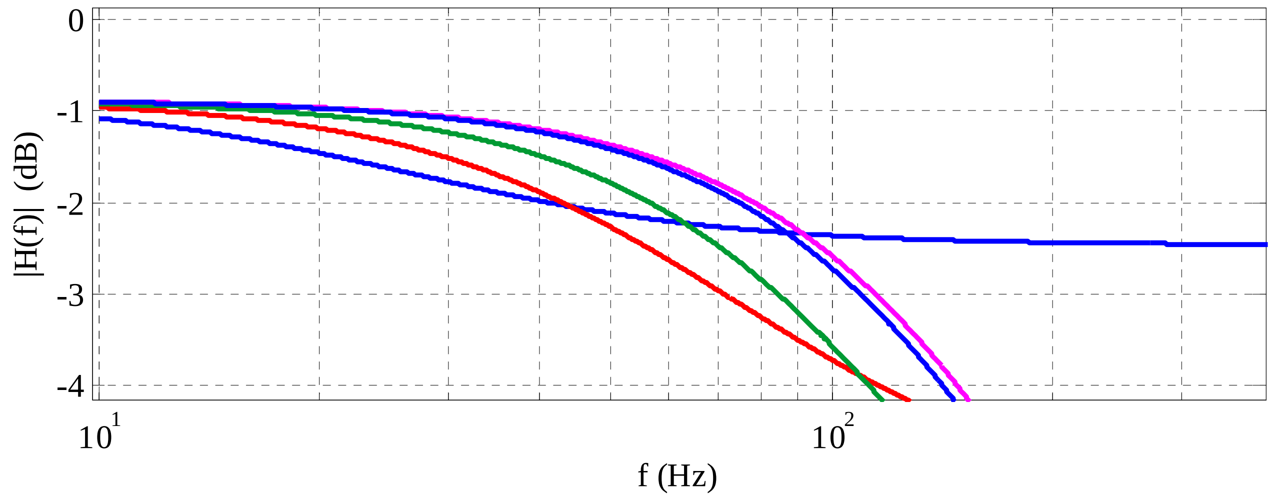
$R_o = 50\ k\Omega$
$R_{in} = 470\ k\Omega$
$R = 250\ k\Omega$
$(\alpha = 0 \to 1)$
$C = 0.022\ \mu F$

$$H(s) = \left(\frac{R_{in}}{R_o + R_{in}}\right)\left[\frac{(\alpha RCs + 1)}{(R_o + \alpha R)Cs + 1}\right]$$

Audio Range

$BP_{pole} = [24\quad 58\quad 96\quad 138\quad 145]$ Hz

$BP_{zero} = [289\quad 965\quad 2894\quad 28{,}937\quad \infty]$



See code on next slide…

Checking the Simulating DT System in Matlab

Because the Freq Resp stays flat (for $\alpha > 0$) we can't use our first method (based on sampling the impulse response).

So… we will use the bilinear transform method…

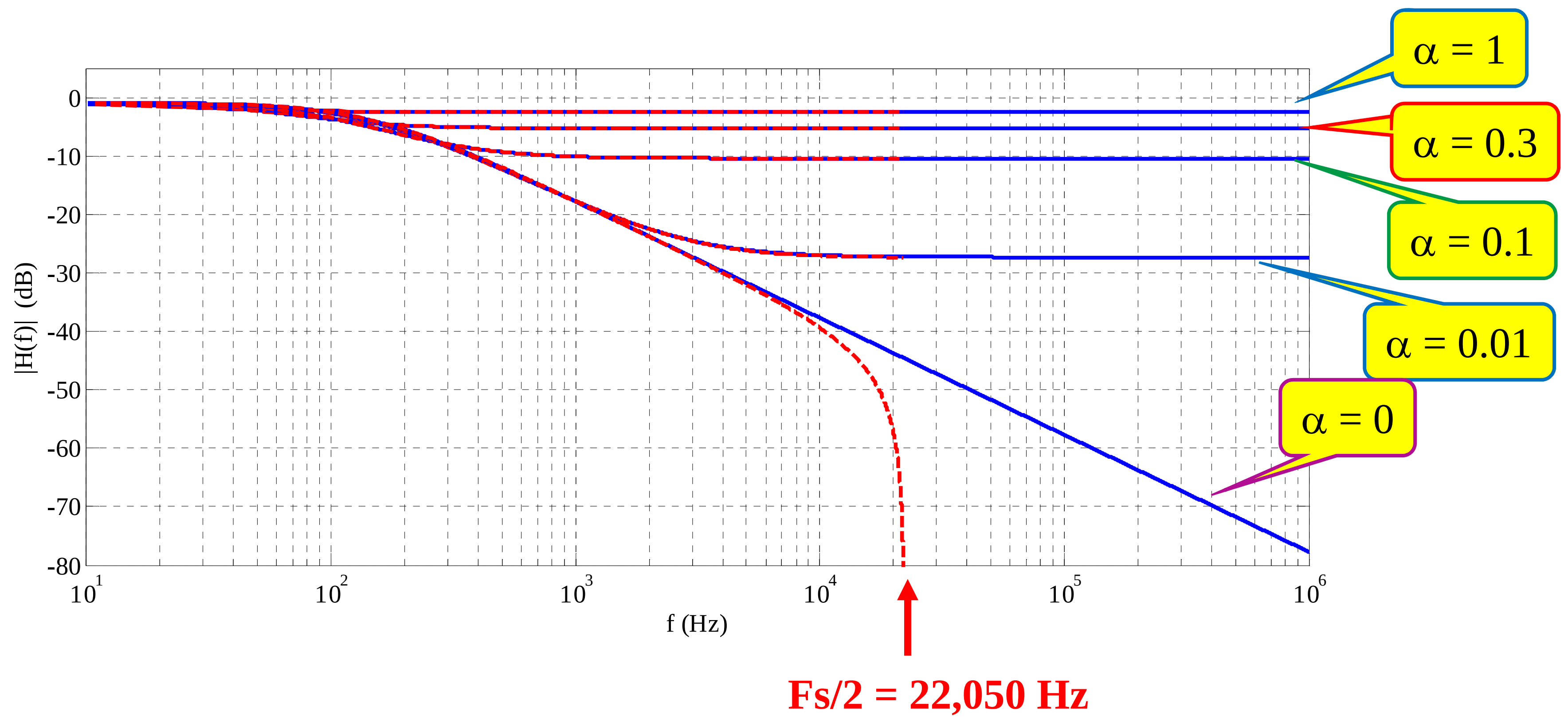
$$H(s) = \left(\frac{R_{in}}{R_o + R_{in}}\right)\left[\frac{(\alpha RCs + 1)}{(R_o + \alpha R)Cs + 1}\right]$$

$$R_o = 50\ k\Omega \qquad R = 250\ k\Omega \quad (\alpha = 0 \rightarrow 1)$$
$$R_{in} = 470\ k\Omega \qquad C = 0.022\ \mu F$$

Code to Compare Circuit's CT Freq Resp to DT Freq Resp

```
%%%%  First compute the true frequency response…
Ro=50000;Rin=470000;R=250000;C=0.022e-6;
G=Rin/(Ro+Rin);
alpha=0.3;   % Set as desired 0 to 1
f=logspace(1,6,10000);  % Specify range of freqs in Hz to plot Freq Resp
w=2*pi*f;    % Convert to rad/sec
H=G*(alpha*R*C*j*w + 1)./ ( (Ro+alpha*R)*C*j*w + 1);
semilogx(f,20*log10(abs(H)))   % plot computed H in dB on log freq axis
hold on
%%%%%   Now design the equivalent DT filter…
Fs = 44100;    % Set desired sampling rate
[NUMd,DEN_d]=bilinear(G*[alpha*R*C 1],[(Ro+alpha*R)*C  1],Fs);
wd=linspace(0,pi,2048);  % Set DT freq in rad/sample
H_bl=freqz(NUMd,DEN_d,wd);  % Compute DT Freq Resp
fd=Fs*wd/(2*pi);   % Convert DT rad/sample to equivalent values in Hz
semilogx(fd,20*log10(abs(H_bl)),'r--')    % plot
```

Comparing Bilinear Designed DT Frequency Response



Running the Simulation of the Circuit in Matlab

```
[x,Fs]=wavread('guitar1.wav');   %  read in wave file of guitar recording
%%%  Note… Fs = 44100 for this file
sound(x,Fs);
Ro=50000;Rin=470000;R=250000;C=0.022e-6;
G=Rin/(Ro+Rin);
alpha = input('Enter alpha value: ');   % Set as desired 0 to 1
[NUMd,DEN_d]=bilinear(G*[alpha*R*C 1],[(Ro+alpha*R)*C  1],Fs);
y=filter(NUMd,DEN_d,x);  %% this D-T filter approximates the C-T circuit
vol_boost=sqrt(sum(x.^2)/sum(y.^2));
%%% vol_boost tries to re-set the volume of the filtered signal
%%%  to be more like the original so that we are comparing only tone
%%%  changes
sound(vol_boost*y,Fs)
```

This code (single_tone_control.m) and the wav file (guitar1.wav) are available on the course web site…

Run the m-file to hear the effect of this filter on the guitar signal for different values of α. You should note that the guitar now sounds "muffled" for small α … that is because the filter has removed some of the high frequency content.