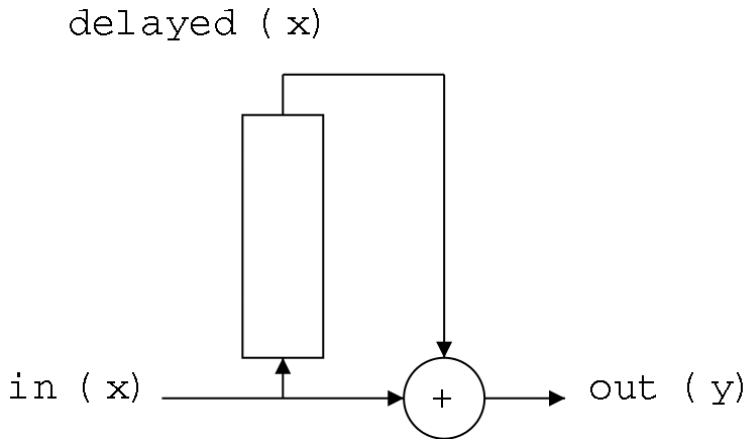
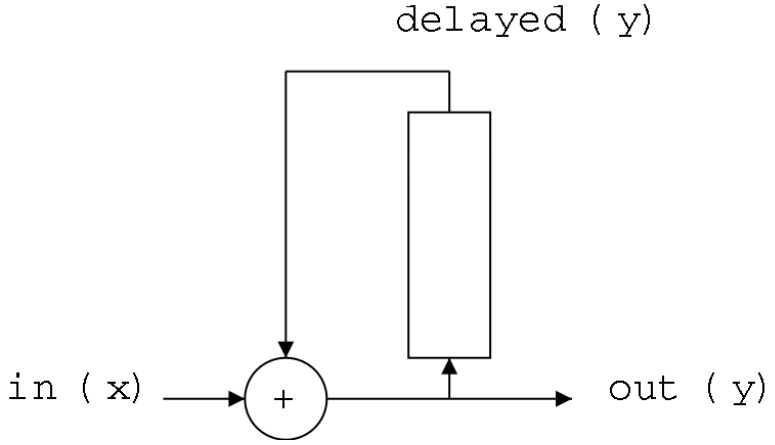
FIR vs. IIR

- one wall
- one-shot delay
- feedforward



Out = in + delayedIn

- two walls
- recirculating delay
- feedback



Out = in + delayedOut

...as math

x_n = this input sample
$x_{(n-N)}$ = Nth previous input

y_n = this output sample
$y_{(n-N)}$ = Nth previous output

$$y_n = a_0 x_n + a_N x_{(n-N)}$$

FIR

$$y_n = a_0 x_n - b_N y_{(n-N)}$$

IIR

• $a_0 = .5$ • $a_1 = .5$	coeffs	• $a_0 = .5$ • $b_1 = -.5$
• $x_0 = 0$	state	• $y_0 = 0$
• $y_n = a_0x_n + a_1x_{(n-1)}$	system	• $y_n = a_0x_n - b_1y_{(n-1)}$

use a pencil

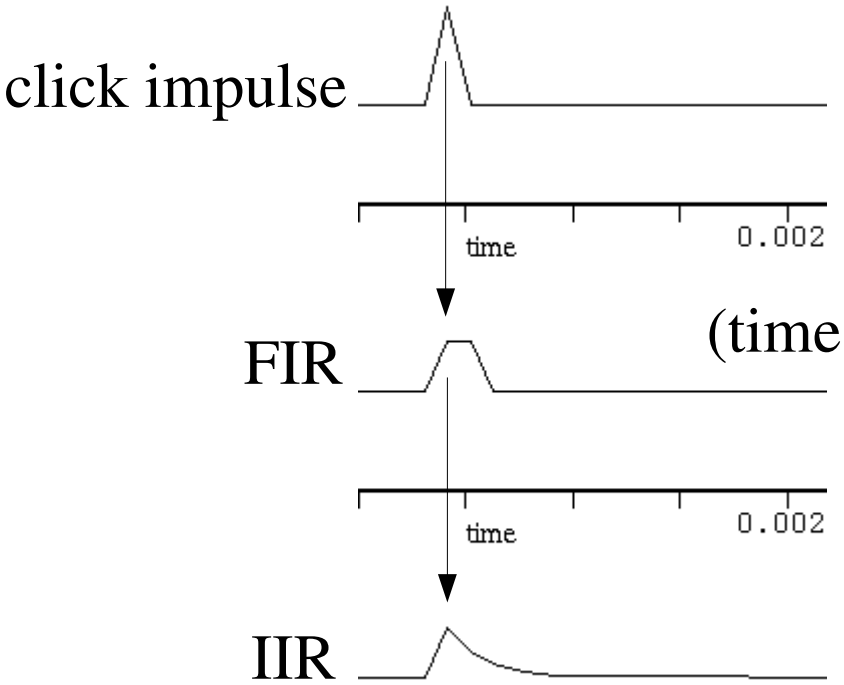
$x_{0,1,2,3...} = \underline{0,1,0,0,0,0,0,...}$

- $a_0 = .5$
- $a_1 = .5$
- $x_0 = 0$
- $y_n = a_0x_n + a_1x_{(n-1)}$

- $a_0 = .5$
- $b_1 = -.5$
- $y_0 = 0$
- $y_n = a_0x_n - b_1y_{(n-1)}$

write out $y_{1,2,3...} = \underline{...}$

impulse responses

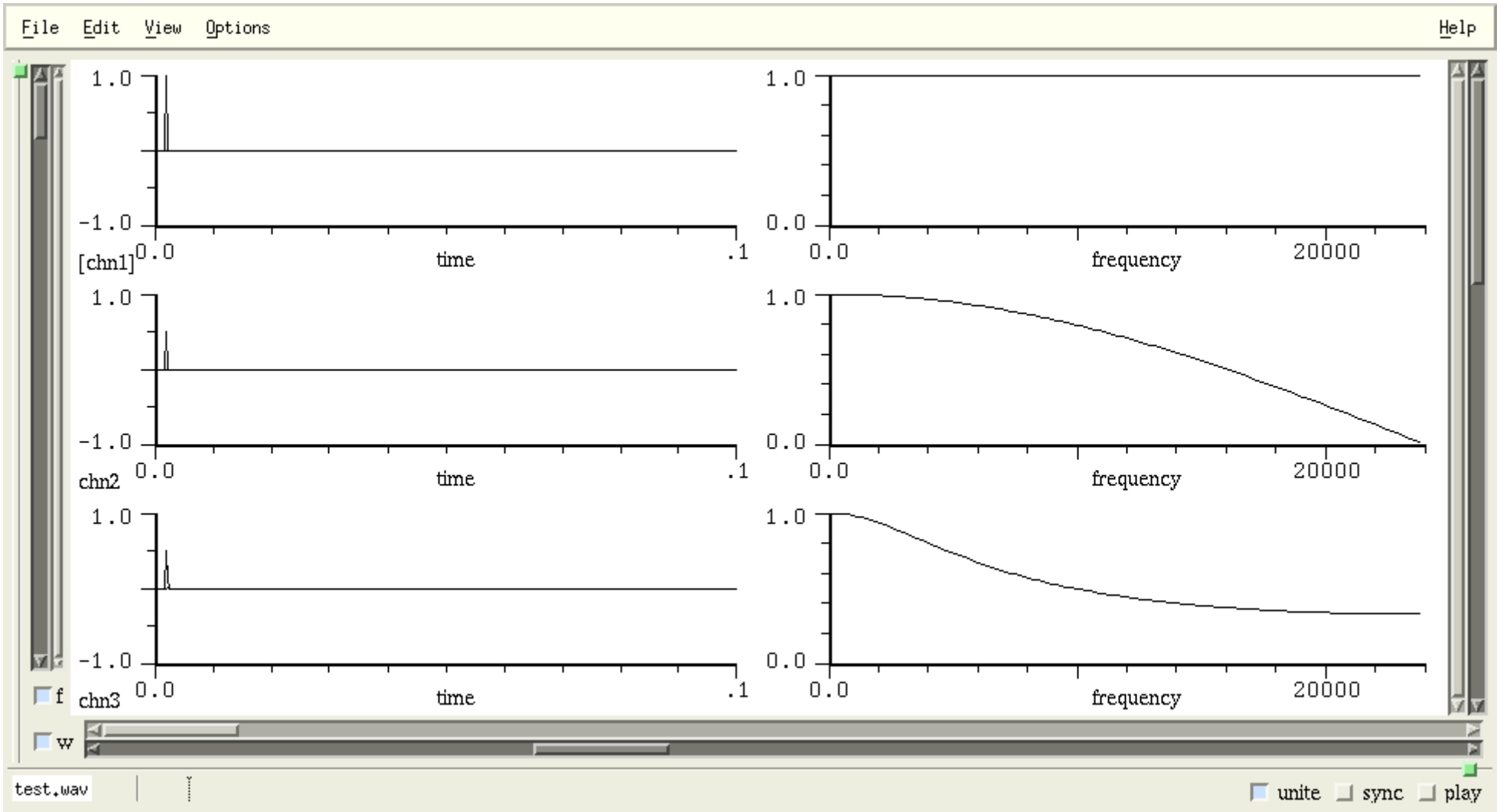


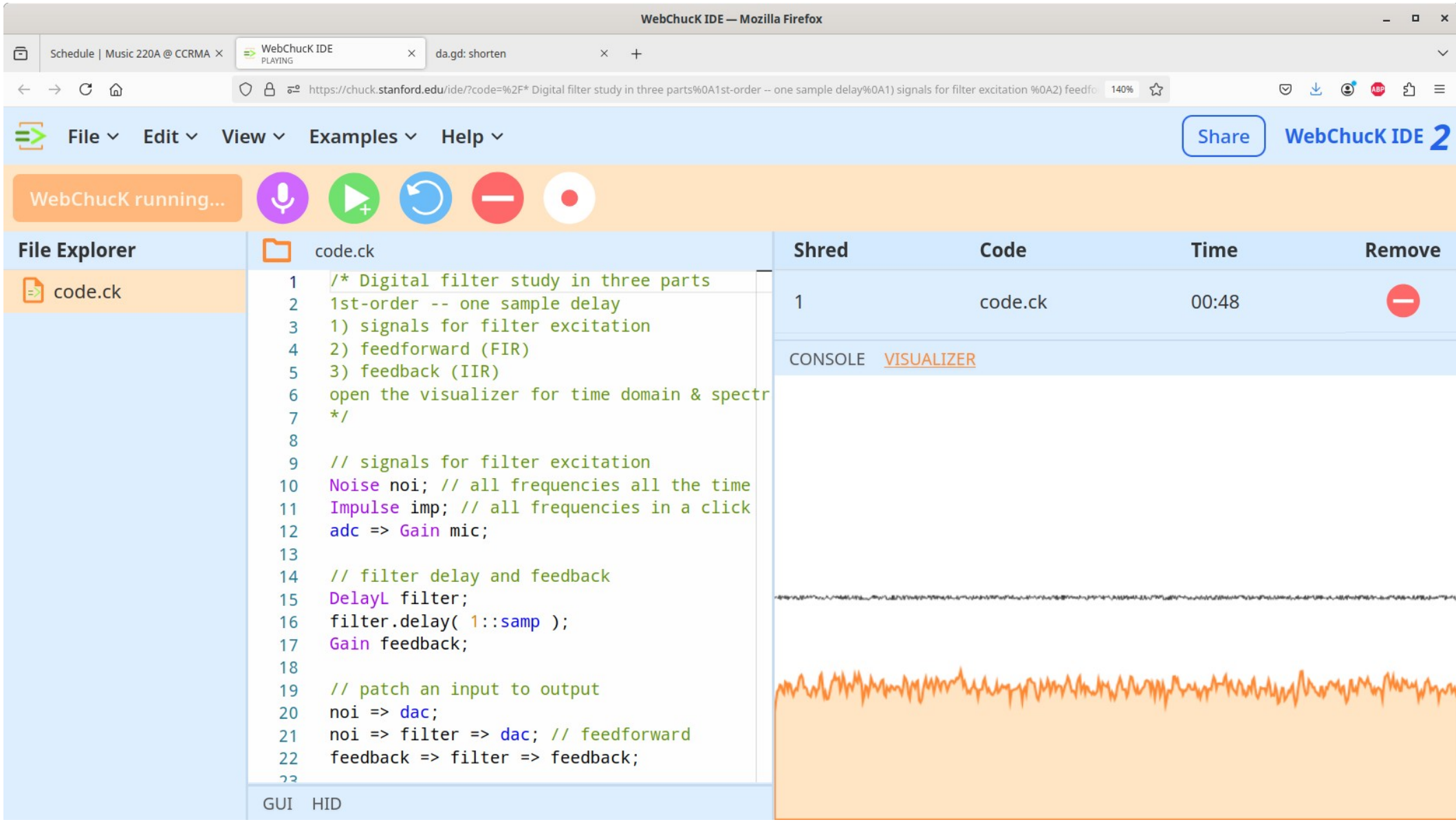
the all-in-one filter difference equation (as C code)

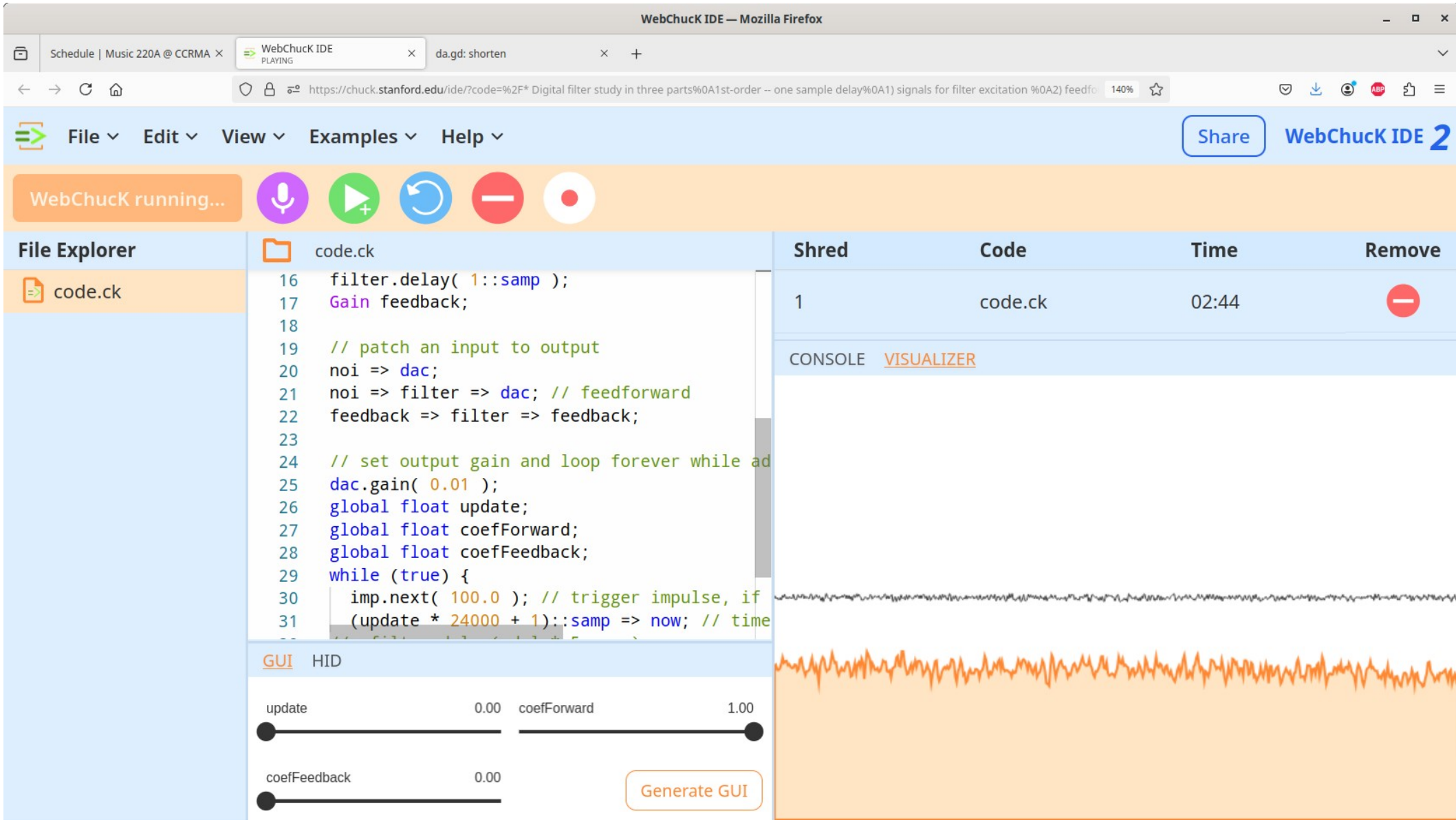
```
/* This code implements the standard difference equation, (Cook: pg. 26)
      y[0] = g * (x[0] + a[1] * x[1]... +a[N] * x[N])
                 - b[1] * y[1] - ... -b[M] * y[M]
For simplicity, N = M = "ORDER" */
```

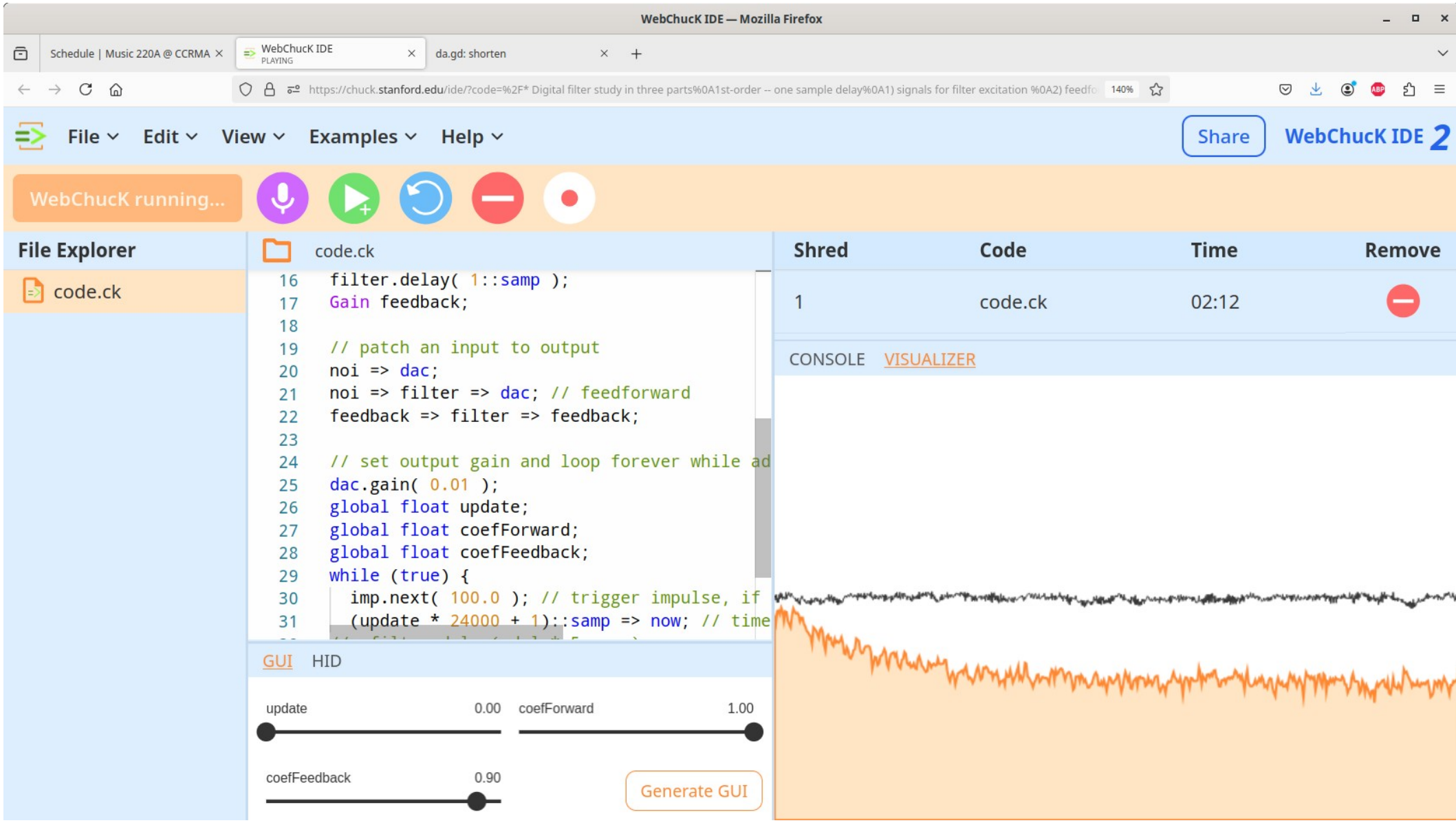
Out = In + delayedIn - delayedOut

[0] = this sample
[1] = last sample
.
.
.
[N] = Nth delayed sample

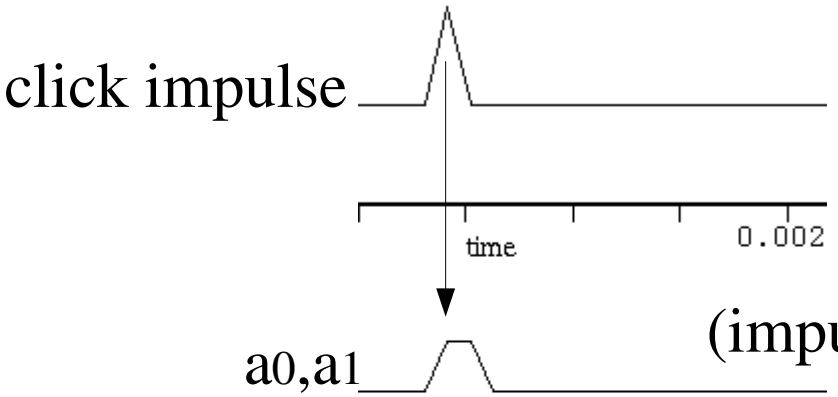

File
Edit
View
Options
Help
1.0
-1.0
0.0
time
.1
[chn1]
chn2
chn3
1.0
0.0
frequency
20000
f
w
test.wav
unite
sync
play


WebChucK IDE — Mozilla Firefox
Schedule | Music 220A @ CCRMA
WebChucK IDE
PLAYING
da.gd: shorten
https://chuck.stanford.edu/ide/?code=%2F* Digital filter study in three parts%0A1st-order -- one sample delay%0A1) signals for filter excitation %0A2) feedfo
140%
File
Edit
View
Examples
Help
Share
WebChucK IDE 2
WebChucK running...
File Explorer
code.ck
code.ck
1 /* Digital filter study in three parts
2 1st-order -- one sample delay
3 1) signals for filter excitation
4 2) feedforward (FIR)
5 3) feedback (IIR)
6 open the visualizer for time domain & spectr
7 */
8
9 // signals for filter excitation
10 Noise noi; // all frequencies all the time
11 Impulse imp; // all frequencies in a click
12 adc => Gain mic;
13
14 // filter delay and feedback
15 DelayL filter;
16 filter.delay(1::samp);
17 Gain feedback;
18
19 // patch an input to output
20 noi => dac;
21 noi => filter => dac; // feedforward
22 feedback => filter => feedback;
GUI
HID
Shred
Code
Time
Remove
1
code.ck
00:48
CONSOLE
VISUALIZER


WebChucK IDE — Mozilla Firefox
Schedule | Music 220A @ CCRMA
WebChucK IDE
PLAYING
da.gd: shorten
https://chuck.stanford.edu/ide/?code=%2F* Digital filter study in three parts%0A1st-order -- one sample delay%0A1) signals for filter excitation %0A2) feedfo
140%
File
Edit
View
Examples
Help
Share
WebChucK IDE 2
WebChucK running...
File Explorer
code.ck
code.ck
16 filter.delay(1::samp);
17 Gain feedback;
18
19 // patch an input to output
20 noi => dac;
21 noi => filter => dac; // feedforward
22 feedback => filter => feedback;
23
24 // set output gain and loop forever while ad
25 dac.gain(0.01);
26 global float update;
27 global float coefForward;
28 global float coefFeedback;
29 while (true) {
30 imp.next(100.0); // trigger impulse, if
31 (update * 24000 + 1)::samp => now; // time
GUI
HID
update
0.00
coefForward
1.00
coefFeedback
0.00
Generate GUI
Shred
Code
Time
Remove
1
code.ck
02:44
CONSOLE
VISUALIZER


WebChucK IDE — Mozilla Firefox
Schedule | Music 220A @ CCRMA
WebChucK IDE
PLAYING
da.gd: shorten
https://chuck.stanford.edu/ide/?code=%2F* Digital filter study in three parts%0A1st-order -- one sample delay%0A1) signals for filter excitation %0A2) feedfo
140%
File
Edit
View
Examples
Help
Share
WebChucK IDE 2
WebChucK running...
File Explorer
code.ck
code.ck
16 filter.delay(1::samp);
17 Gain feedback;
18
19 // patch an input to output
20 noi => dac;
21 noi => filter => dac; // feedforward
22 feedback => filter => feedback;
23
24 // set output gain and loop forever while ad
25 dac.gain(0.01);
26 global float update;
27 global float coefForward;
28 global float coefFeedback;
29 while (true) {
30 imp.next(100.0); // trigger impulse, if
31 (update * 24000 + 1)::samp => now; // time
GUI
HID
update
0.00
coefForward
1.00
coefFeedback
0.90
Generate GUI
Shred
Code
Time
Remove
1
code.ck
02:12
CONSOLE
VISUALIZER

convolution (FIR)

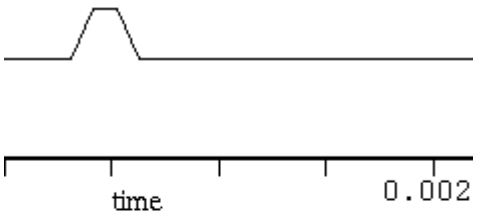


(impulse response = coefficients!)

convert any IR to FIR

Low order FIR

a0,a1



High order FIR

a0,a1,a2,a3,a4,...

(coefficients made from IR of IIR)

summary

- filters, linear, don't add frequencies (as opposed to modulation or distortion, non-linear)
- filters change duration and spectral weighting
- size from big to little: echo, pitch, eq, image pos.
- Sharpness ('Q') from high to low: string, tube
- it's hard to find simple filters in nature, most are combinations, complexes