Digital Signal Processing Lab

Demo 6 - Exercise 4 (Canonical form)

Saad Zubairi
shz2020

September 24, 2025

Solution

In this solution, we show how we can implement a fourth-order difference equation with just 4 variables to store past values (i.e 4 delay units). We use the canonical form for this purpose. For this we made the following changes to the original `wave_filter_python.py` file:

- We begin by initializing the 4 state variables, as opposed to 8 in the original example:

```
    w1 = 0.0
    w2 = 0.0
    w3 = 0.0
    w4 = 0.0

```

Snippet 1: Canonical States initialization

- we then calculate the output value using canonical direct form as per Orfanidis Eq. 7.2.5:

```
    w0 = x0 - a1*w1 - a2*w2 - a3*w3 - a4*w4
    y0 = b0*w0 + 0*w1 + b2*w2 + 0*w3 + b4*w4

```

Snippet 2: Canonical (Direct Form II)

- And of course, we also change the state update/delays by reverse order shift:

```
    w4 = w3
    w3 = w2
    w2 = w1
    w1 = w0

```

Snippet 3: delays or state update (reverse-order shift)

Rest of the code of course, remains largely the same. The resulting audio output from this implementation is the same as the one produced by the demo program, thereby certifying that a fourth-order difference equation can be implemented using just 4 variables.

Full code

```
import pyaudio
import wave
import struct

def clip16( x ):
    # Clipping for 16 bits
    if x > 32767:
        x = 32767
    elif x < -32768:
        x = -32768
    else:
        x = x
    return (x)

wavefile = 'author.wav'  # same as the demo

print('Play the wave file %s.' % wavefile)
wf = wave.open(wavefile, 'rb')

num_channels  = wf.getnchannels()
RATE          = wf.getframerate()
signal_length = wf.getnframes()
width         = wf.getsampwidth()

print('The file has %d channel(s).'            % num_channels)
print('The frame rate is %d frames/second.'    % RATE)
print('The file has %d frames.'                % signal_length)
print('There are %d bytes per sample.'         % width)

# Difference equation coefficients
b0 =  0.008442692929081
b2 = -0.016885385858161
b4 =  0.008442692929081

# a0 =  1.000000000000000
a1 = -3.580673542760982
a2 =  4.942669993770672
a3 = -3.114402101627517
a4 =  0.757546944478829

# canonical states initialization
w1 = 0.0
w2 = 0.0
w3 = 0.0
w4 = 0.0

p = pyaudio.PyAudio()

# Open audio stream
stream = p.open(
    format    = pyaudio.paInt16,
    channels = num_channels,
    rate      = RATE,
    input     = False,
    output    = True
)
```

```

# Get first frame from wave file
input_bytes = wf.readframes(1)  # 1 frame (mono)

while len(input_bytes) > 0:
    # Convert binary data to number
    input_tuple = struct.unpack('h', input_bytes)
    x0 = float(int(input_tuple[0]))

    # Canonical (Direct Form II)
    w0 = x0 - a1*w1 - a2*w2 - a3*w3 - a4*w4
    y0 = b0*w0 + 0*w1 + b2*w2 + 0*w3 + b4*w4

    # delays or state update (reverse-order shift)
    w4 = w3
    w3 = w2
    w2 = w1
    w1 = w0

    # Compute output value
    output_value = int(clip16(y0))    # Integer in allowed range

    # Convert output value to binary data
    output_bytes = struct.pack('h', output_value)

    # Write binary data to audio stream
    stream.write(output_bytes)

    # Get next frame from wave file
    input_bytes = wf.readframes(1)

print('* Finished')

stream.stop_stream()
stream.close()
p.terminate()
```
Snippet 4: example code