**Demo 7 #4**

4. Modify the demo program filter\_play\_mic.py to process the input signal x(t). The output signal should be

y(t) = x(t) cos(2pif0t)

there f0 = 400 Hz. The output signal y(t) should both be played to the speaker and save to a WAV file. This is amplitude modulation. What is the effect of this on the voice signal? Submit your WAV file of yourself talking, as well as your code.

**Python code :** 07.py

import math

import struct

import pyaudio

import wave

WIDTH = 2

CHANNELS = 1

RATE = 16000

DURATION = 10

f0 = 400

N = DURATION \* RATE

def clip16( x ):

if x > 32767:

x = 32767

elif x < -32768:

x = -32768

else:

x = x

return (x)

p = pyaudio.PyAudio()

stream = p.open(

format = p.get\_format\_from\_width(WIDTH),

channels = CHANNELS,

rate = RATE,

input = True,

output = True)

print('\* Start')

newwf = wave.open('JingjieSheng\_7.wav', 'w')

newwf.setnchannels(CHANNELS)

newwf.setsampwidth(WIDTH)

newwf.setframerate(RATE)

for n in range(0, N):

input\_string = stream.read(1, exception\_on\_overflow = False)

input\_tuple = struct.unpack('h', input\_string)

input\_value = input\_tuple[0]

x0 = input\_value

y0 = math.cos(2\*math.pi\*f0\*n/RATE)\*x0

output\_value = int(clip16(y0))

output\_string = struct.pack('h', output\_value)

stream.write(output\_string)

newwf.writeframesraw(output\_string)

print('\* Finished')

stream.stop\_stream()

stream.close()

p.terminate()

**WAV file:** JingjieSheng\_7.wav

**Comment :**

I only need to change the difference equation because it is amplitude modulation.

However, I first modified the code as follow: ‘y0 = math.cos(2\*math.pi\*f0\*n)\*x0’. The program run with a normal voice and I thought it may be correct.

After the fourth lecture, I noticed that when the variation is t that means time. N = DURATION \* RATE is the total frames, so if I need to get the time I should use n divide RATE. The voice after filter became more lower, since the divisor effected on the frequency. That means I get lower frequency.