# DIGITAL SIGNAL PROCESSING IIR FILTER

## 1 TITLE

Solve a problem which requires real-time filtering of a physical quantity.

## 2 INTRODUCTION

The focus of this project is to filter a physical quantity using Infinite Impulse Response (IIR) filter. This filtering will be done in real-time using Arduino for data acquisition. The physical quantity that is filtered is Temperature. The temperature is measured using LM35 temperature sensor.

PROBLEM STATEMENT: To measure the temperature of a flowing hot liquid, we can’t put the LM35 temperature sensor directly inside the hot liquid as the material makeup of the temperature sensor will be destroyed. A conventional solution is to put the temperature sensor in a casing of material that is resistant to the hot liquid. The downside of this solution is that the casing will act as an insulator; hence there will be a time delay in measuring the actual temperature.

PROPOSED SOLUTION: The Infinite Impulse Response (IIR) filter is a recursive filter, meaning that the output is calculated using the current and prior inputs and outputs. There is output feedback in the filter structure because the filter uses prior output values. An IIR filter is used to simulate the time delay when using an LM35 temperature sensor in a casing to measure the temperature of a hot liquid. Basically, the IIR filter will be configured with a time constant that corresponds to the temperature measurement time delay.

## 3 MATHEMATICS INVOLVED

The Fourier transform of a signal is defined as



where ω is digital frequency in radians and is a complex exponential sequence.

The operation in time domain which relates the input signal x(n), impulse response h(n), and the output signal y(n), is called the convolution, and is defined as

y(n) = x(n)\*h(n)

In frequency domain, the convolution operation becomes multiplication

Y() = X()H()

Where Y(), X() and H() are the Fourier transforms of y(n), x(n) and h(n) respectively. The quantity H() is called the frequency response of the digital filter, and it is a complex function of the frequency ω with a period 2π.

H() = HR() + HIm() = |

The amplitude | is called the magnitude response and the phase φ(ω) is called the phase response of the digital filter.

The magnitude response can be expressed in logarithmic form in decibels as

where is the Gain function.

Since Y() = X()H(),

H() =

Using Z-transform, we have that

H(z) =

The roots of the numerator for which H(z)=0 are the locations of the zeros in the complex z plane and the roots of the denominator for which H(z) become infinite are the locations of the poles. A zero is usually denoted by a circle o and the pole by a cross x.

## 4 IMPLEMENTATION

## 5 DRAWBACKS

IIR filtering is more susceptible to the noise that limit cycles and calculations generate than the other type of Digital Signal Processing filtering, Finite Impulse Response (FIR) filtering.

## 6 APPENDICES

import numpy as np

import pylab as pl

import iir\_filter

from scipy import signal

from pyfirmata2 import Arduino, util

import time

board = Arduino('COM3')

print("Communication Successfully started")

it = util.Iterator(board)

it.start()

board.analog[0].enable\_reporting()

time.sleep(3)

sample = (board.analog[0].read())

data = np.loadtxt('ecg\_50hz\_1.dat')

data = data - 2048

data = data \* 2E-3 \* 500 / 1000

fs = 1000.0

#

pl.title('ECG')

#

y = data[:,1]

t = data[:,0]

pl.subplot(310)

pl.plot(t,y)

pl.xlabel('time/sec')

pl.ylabel('ECG/mV')

#

f0 = 48.0

f1 = 52.0

sos1 = signal.butter(4, [f0/fs\*2,f1/fs\*2], 'bandstop', output='sos')

f2 = 100

sos2 = signal.butter(4, f2/fs\*2, output='sos')

iir1 = iir\_filter.IIR\_filter(sos1)

iir2 = iir\_filter.IIR\_filter(sos2)

y2 = np.zeros(len(y))

# for i in range(len(y)):

# y2[i] = iir1.filter(iir2.filter(y[i]))

y2= iir1.filter(iir2.filter(sample))

# yf = np.fft.fft(y2) / len(y2)

yf = np.fft.fft(y2)

yf[0] = 0

pl.subplot(312);

pl.plot(np.linspace(0,fs,len(yf)),20\*np.log10(abs(yf)))

pl.xlabel('time/sec')

pl.ylabel('Spectrum/dB')

pl.xlim(0,fs/2)

#

#

pl.subplot(312)

pl.plot(t,y2)

pl.xlabel('time/sec')

pl.ylabel('ECG/mV')

freqResponse = np.fft.fft(y2)

freqResponse = abs(freqResponse[0:int(len(freqResponse)/2)])

xfF = np.linspace(0,fs/2,len(freqResponse))

pl.subplot(313)

pl.figure("Frequency Response")

pl.plot(xfF,np.real(freqResponse))

pl.xlabel("Frequency [Hz]")

pl.ylabel("Amplitude")

pl.title("Bandstop")

pl.show()

## 7 REFERENCES

Digital Filters by Gordana Jovanovic Dolecek; *Encyclopedia of Multimedia Technology and Networking by Margherita Pagani Bocconi University, Italy.*

[Introduction to Digital Signal Processing](https://www.sciencedirect.com/science/article/pii/B9780750683975000076) by Ian Grout in *[Digital Systems Design with FPGAs and CPLDs](https://www.sciencedirect.com/book/9780750683975/digital-systems-design-with-fpgas-and-cplds), 2008.*