

C++ Code for generating IIR transfer function coefficients

Created by Charlie Laub, May 2024

```
#include <stdlib.h>
#include <math.h>
#include <string>
#include <iostream>
#include <iomanip>
using namespace std;

int main () {

// ===== C++ CODE TO CALCULATE NOTCH FILTER IIR BIQUAD COEFFICIENTS =====
// Author: Charlie Laub, May 2024
// The general biquadratic 2nd order transfer function in z has the form:
//  $H(z) = (b_0 + b_1z^{-1} + b_2z^{-2}) / (a_0 + a_1z^{-1} + a_2z^{-2})$ 

// NOTE: the DC gain is automatically corrected as part of the calculations

//these parameters describe the notch filter:
const float polarity = 1;      //set polarity to either 1 or -1
const float Fp = 750.0;        //pole frequency in Hertz
const float Qp = 2.0;          //pole Q
const float Fz = 1500.0;       //zero frequency in Hertz
const float SR = 48000.0;      //sample rate in Hertz

//declare program variables:
double Aa0, Aa1, Aa2, Ab0, Ab1, Ab2; //analog TF coefficients
double Da0, Da1, Da2, Db0, Db1, Db2; //IIR digital TF coefficients
const double K = 2.0*SR;
const double K2 = K*K;
double gain, Wp, Wz, Wp2, Wz2;

//Calculate IIR transfer function coefficients via analog form
//calculate analog domain radian frequencies
Wp = 6.2831853072*Fp;
Wz = 6.2831853072*Fz;
//apply pre-warping for IIR filter calculations
Wp = K*tan( Wp/K );
Wz = K*tan( Wz/K );
Wp2 = Wp*Wp;
Wz2 = Wz*Wz;
//calculate gain correction for lowpass notch using pre-warped frequencies
gain = 1.0;
if (Wp < Wz) {
    gain = Wp2 / Wz2;
}
//initialize these analog coefficient values to zero
Aa1 = Aa2 = Ab0 = Ab1 = Ab2 = 0.0;
//and initialize the value of analog coefficient a0 to 1.0
Aa0 = 1.0;

//2nd order notch specified by gain, polarity, Fp, Qp, Fz
Ab2 = polarity * gain;
Ab0 = polarity * gain * Wz2;
Aa2 = 1.0;
Aa1 = Wp/Qp;
Aa0 = Wp2;
```

```

//convert the analog TF coefficients to z^-1 domain TF coefficients
Db0 = Ab2*K2 + Ab1*K + Ab0;
Db1 = 2.0*Ab0 - 2.0*Ab2*K2;
Db2 = Ab2*K2 - Ab1*K + Ab0;
Da0 = Aa2*K2 + Aa1*K + Aa0;
Da1 = 2.0*Aa0 - 2.0*Aa2*K2;
Da2 = Aa2*K2 - Aa1*K + Aa0;

//convert to normalized form by dividing thru by Da0:
Db0 /= Da0;
Db1 /= Da0;
Db2 /= Da0;
Da1 /= Da0;
Da2 /= Da0;

//print results to screen
cout << "==== NOTCH FILTER IIR BIQUAD COEFFICIENTS FROM USER PARAMETERS =====" << endl;
cout << "                Author: Charlie Laub, May 2024" << endl << endl;
cout << " The general biquadratic 2nd order transfer function in z has the form:" << endl;
cout << "   H(z) = (b0 + b1*z^-1 + b2*z^-2) / (a0 + a1*z^-1 + a2*z^-2)" << endl;
cout << endl;
cout << "NOTE: the DC gain will automatically be corrected for LP notch filters" << endl;
cout << endl;

cout << "Given the input parameters:" << endl;
cout << "   polarity = " << polarity << endl;
cout << "   pole frequency = " << Fp << " Hz" << endl;
cout << "   pole Q = " << Qp << endl;
cout << "   zero frequency = " << Fz << " Hz" << endl;

cout << "The response will be a ";
if (Fp < Fz) cout << "lowpass"; else if (Fp == Fz) cout << "symmetric"; else cout <<
"highpass";
cout << " notch." << endl << endl;

std::cout << std::fixed;
std::cout << std::setprecision(0);
cout << "The IIR Transfer Function Coefficients for a sample rate of " << SR << " Hz are:"
<< endl;
std::cout << std::setprecision(15);
cout << "   a0 = " << 1.0 << endl;
cout << "   a1 = " << Da1 << endl;
cout << "   a2 = " << Da2 << endl;
cout << "   b0 = " << Db0 << endl;
cout << "   b1 = " << Db1 << endl;
cout << "   b2 = " << Db2 << endl;
cout << endl << endl;

} //end program int main()

```

Example calculation: The following output should be obtained using the input parameters provided in the code. The z^{-1} domain coefficients are provided in normalized form.

```
===== NOTCH FILTER IIR BIQUAD COEFFICIENTS FROM USER PARAMETERS =====
                        Author: Charlie Laub, May 2024

The general biquadratic 2nd order transfer function in z has the form:
 $H(z) = (b_0 + b_1z^{-1} + b_2z^{-2}) / (a_0 + a_1z^{-1} + a_2z^{-2})$ 

NOTE: the DC gain has been automatically corrected for LP notch filters

Given the input parameters:
    polarity = 1
    pole frequency = 750 Hz
    pole Q = 2
    zero frequency = 1500 Hz
The response will be a lowpass notch.

The IIR Transfer Function Coefficients for a sample rate of 48000 Hz are:
    a0 = 1.000000000000000
    a1 = -1.942763424541570
    a2 = 0.952163625981326
    b0 = 0.244609383768365
    b1 = -0.479818566096973
    b2 = 0.244609383768365
```

NOTES:

The code can be compiled under Linux using the GCC compiler. Save the code to a file, e.g. IIR_calc.cpp.

To generate an executable file, run the command:

```
g++ IIR_calc.cpp
```

To run the executable file, run the command:

```
./a.out
```

If you wish to calculate the coefficients for a different set of parameters, edit the cpp file and then repeat the steps above.

In the IIR biquadratic transfer function, the assignment of coefficients a_n and b_n to the numerator or denominator is arbitrary and sometimes the a_n will be used to represent the numerator coefficients (that create the zero) and the b_n to represent the denominator coefficients (that create the pole). If the response does not appear correct, compare the form of the transfer function that is being used in the application against the form shown above, and apply the numerator and denominator coefficients accordingly.

Checking the IIR filter response can be done using an online IIR response calculator. One example can be found at this URL:

<https://www.earlevel.com/main/2021/09/02/biquad-calculator-v3/>