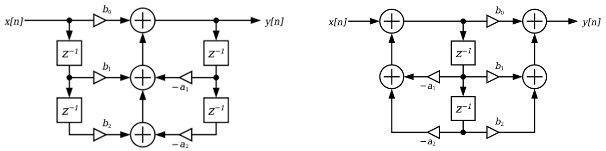
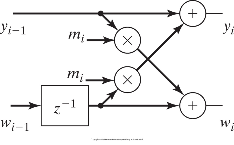
**EE 419 - Project 6 - Project Report**

Efficient IIR Filter Implementations

|  |  |
| --- | --- |
| **Names: Chris Adams & Aiku Shintani** | **Lab Date: 2/12/19** |
| **Bench #: 9** | **Section: 2** |

**Learning Objectives:**

* Observe the effects of coefficient precision on the frequency response, pole/zero locations and stability of IIR filters.
* Develop and verify filter implementations that reduce the effects of coefficient rounding on filter performance.
* Observe “limit cycles” and find the relationship of their severity to filter coefficient values and quantization precision

**1) Modeling the Effects of Coefficient Quantization on an IIR Double Notch Filter**

**Ideal (Full Precision) Filter Design and Analysis:**

**Filter Design Specifications:**

Notch filter (two notches)

Notch Center Frequencies: *f1* = 1000 + 5% Hz, *f2* = 2000 + 5% Hz,

Sampling Rate S = 48 kHz

Pole radius = 0.95

Peak Passband Gain |H(f)|max = 1.0 (+/- 1%)

a) What are the pole and zero locations? (Using polar coordinates: Radius  Angle )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Poles: | ° | ° | ° | ° |
| Zeros: | ° | ° | ° | ° |

b) Report your difference equation filter coefficients (floating point values):

Overall Gain Constant *K* = 0.9037008

Difference Equation Coefficients:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **k=0** | **k=1** | **k=2** | **k=3** | **k=4** |
| Ak: | **1** | **-3.71900430655947** | **5.26216053096260** | **-3.3564013866699** | **0.8145062500** |
| Bk: | **0.9037008** | **-3.53775491267499** | **5.26916197402696** | **-3.53775491267499** | **0.9037008** |

**Filter Analysis Results:**

Notch Center Frequencies: (Specification: *f1* = 1000 + 50 Hz, *f2* = 2000 + 100 Hz )

*f1* = **1000.1 Hz**  % Error *f1* =  **0.01%**

*f2* = **1999.9 Hz**  % Error *f2* =  **0.005%**







**Scaled Integer Filter Implementation Modeling – 4th-order Direct Form II Transposed Filter:**

**4th Order Direct Form IIR Filter - Scale Factor = 512**

Overall Gain Constant *K* = 0.904296875

Difference Equation Coefficients:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **k=0** | **k=1** | **k=2** | **k=3** | **k=4** |
| **Ak** (integer): | **512** | **-1904** | **2694** | **-1718** | **417** |
| **Bk** (integer): | **463** | **-1811** | **2698** | **-1811** | **463** |
| **Ak** (effective): | **1** | **-3.7187500** | **5.26171875** | **-3.35546875** | **0.814453125** |
| **Bk** (effective): | **0.904296875** | **-3.537109375** | **5.26953125** | **-3.537109375** | **0.904296875** |
| **% Error Ak** | **0%** | **-0.006838028%** | **0.0083954292158%** | **-0.0277868039%** | **0.0065223563355%** |
| **% Error Bk** | **0.06595933079%** | **-0.0182470999%** | **0.0070082486525%** | **-0.0182470999%** | **0.06595933079%** |

# Bits Required for integer coefficients = 13

Notch Center Frequencies: (Specification: *f1* = 1000 + 50 Hz, *f2* = 2000 + 100 Hz )

*f1* = **NA Hz**  % Error *f1* =  **100%**

*f2* = **NA Hz**  % Error *f2* =  **100%**

Pole and Zero Locations: (Using polar coordinates: Radius  Angle )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Poles (effective): | ° | ° | ° | ° |
| Zeros (effective): | ° | ° | ° | ° |





****

**4th Order Direct Form IIR Filter – Sufficient Precision - Scale Factor =2^13 = 8192**

Overall Gain Constant *K* = 0.9036865234375

Difference Equation Coefficients:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **k=0** | **k=1** | **k=2** | **k=3** | **k=4** |
| **Ak** (integer): | **8192** | **-30466** | **43108** | **-27496** | **6672** |
| **Bk** (integer): | **7403** | **-28981** | **43165** | **-28981** | **7403** |
| **Ak** (effective): | **1** | **-3.71899414063** | **5.26220703125** | **-3.3564453125** | **0.814453125** |
| **Bk** (effective): | **0.9036865234375** | **-3.53771972656** | **5.2691650390625** | **-3.53771972656** | **0.9036865234375** |
| **% Error Ak** | **0** | **0.000273351%** | **0.00088367291584%** | **0.001308718%** | **0.0065223563355%** |
| **% Error Bk** | **0.0015797886314%** | **0.000994588754%** | **0.0005816931725%** | **0.000994588754%** | **0.0015797886314%** |

# Bits Required for integer coefficients = 18

Notch Center Frequencies: (Specification: *f1* = 1000 + 50 Hz, *f2* = 2000 + 100 Hz )

*f1* = **1033Hz**  % Error *f1* =  **3.3%**

*f2* = **1994Hz**  % Error *f2* =  **0.3%**

Pole and Zero Locations: (Using polar coordinates: Radius  Angle )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Poles (effective): | ° | ° | ° | ° |
| Zeros (effective): | ° | ° | ° | ° |







**Scaled Integer Filter Implementation – Cascaded 2nd-order Filters:**

**Cascaded 2nd Order IIR Filters – Scale Factor = 512**

Difference Equation Coefficients and Pole/Zero Locations:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **First 2nd Order Section** | | | **Second 2nd Order Section** | | |
| **k=0** | **k=1** | **k=2** | **k=0** | **k=1** | **k=2** |
| **Ak** (exact): | **1** | **-1.883745237** | **0.90250** | **1** | **-1.83525907** | **0.90250** |
| **Bk** (exact): | **0.9037008** | **-1.791939029** | **0.9037008** | **0.9037008** | **-1.745815884** | **0.9037008** |
| **Ak** (integer): | **512** | **-964** | **462** | **512** | **-940** | **462** |
| **Bk** (integer): | **463** | **-917** | **463** | **463** | **-894** | **463** |
| **Ak** (effective): | **1** | **-1.8828125** | **0.90234375** | **1** | **-1.8359375** | **0.90234375** |
| **Bk** (effective): | **0.904296875** | **-1.791015625** | **0.904296875** | **0.904296875** | **-1.74609375** | **0.904296875** |
| Poles (effective): | ° | ° |  | ° | ° |  |
| Zeros (effective): | ° | ° |  | ° | ° |  |

Notch Center Frequencies: (Specification: *f1* = 1000 + 50 Hz, *f2* = 2000 + 100 Hz )

*f1* = **1057Hz**  % Error *f1* =  **5.7%** 2nd Order Section: (First or Second)

*f2* = **2018Hz**  % Error *f2* =  **0.9%** 2nd Order Section: (First or Second)

**First 2nd-Order Section Pole/Zero Diagram**



**Second 2nd-Order Section Pole/Zero Diagram**



**First 2nd-Order Section Frequency Response**



**Second 2nd-Order Section Frequency Response**



**Two Cascaded 2nd-Order Sections Overall Frequency Response**



Compare the performance of the Cascaded 2nd-Order Section to that of the 4th-0rder Direct Form implementation with the same scale factor.

The cascaded 2nd order performs as well as the sufficient precision 4th order implementation without sacrificing the number of bits required to quantize. This allows for quicker computation and less memory. In contrast, the 4th order of same scale factor was ineffective and did not even produce notches near the respective frequencies (large quantization error).

**Scaled Integer Filter Implementation – Lattice-Ladder Filter Structure**

**4th Order Lattice-Ladder Filter – Scale Factor = 512**

Reflection and Ladder Coefficients:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **k=0** | **k=1** | **k=2** | **k=3** | **k=4** |
| **mk** (exact): | **-0.98740016714** | **0.9897933530542** | **-0.9722786694** | **0.8145062500** |  |
| **Ck** (exact): | **0.00013140236803** | **0.000511412468** | **-0.0040208042** | **-0.176887745634** | **0.903700800** |
| **mk** (integer): | **-506** | **507** | **-498** | **417** |  |
| **Ck** (integer): | **0** | **0** | **-2** | **-91** | **463** |
| **mk** (effective): | **-0.9882812500** | **0.99023437500** | **-0.9726562500** | **0.81445312500** |  |
| **Ck** (effective): | **0** | **0** | **-0.0039062500** | **-0.17773437500** | **0.90429687500** |

Notch Center Frequencies: (Specification: *f1* = 1000 + 50 Hz, *f2* = 2000 + 100 Hz )

Exact Filter Coefficients:

*f1* = **1001.5Hz**  % Error *f1* =  **0.15%**

*f2* = **2001.6Hz**  % Error *f2* =  **0.08%**

Quantized (Reduced Precision) Filter Coefficients:

*f1* = **1003.2Hz**  % Error *f1* =  **0.32%**

*f2* = **2003.2Hz**  % Error *f2* =  **0.16%**

**Exact Coefficient Lattice-Ladder Filter Frequency Response**



**Quantized Coefficient Lattice-Ladder Filter Frequency Response**



**Compare the frequency response results** of these two filters (compared the full precision and quantized filters to each other), and compare these results to those from the 4th-0rder Direct Form, and cascaded 2nd-order Direct Form implementations that used the same scale factors. Did the lattice-ladder implementation achieve similar or better/worse results? [Explain the filter performance metrics by which you reached your conclusions.]

**Of the quantized filter implementations, the lattice ladder produces the closest result to the full precision filter. The error in the notch frequencies are 0.32% and 0.16% respectively with a scale of 512. In comparison, the 2nd order cascaded had errors of 5.7% and 0.9% respectively. Lastly the 4th order direct form produced only a single notch and was nothing like the original filter. Therefore it is clear that the lattice ladder implementation is the most accurate for the same scale factor.**

**Matlab code** used for computing the lattice-ladder filter coefficients, and for plotting the frequency response plots:

%get exact Ak and Bk coefficients for desired notch

[Ak, Bk, Hf, Fd] = show\_filter\_responses\_pz([0.95\*exp(j\*pi\*15/180), 0.95\*exp(-j\*pi\*15/180), 0.95\*exp(j\*pi\*7.5/180), 0.95\*exp(-j\*pi\*7.5/180)],[1\*exp(j\*pi\*15/180), 1\*exp(-j\*pi\*15/180), 1\*exp(j\*pi\*7.5/180), 1\*exp(-j\*pi\*7.5/180)], 0.9037008, 48e3,100e3, 40, 1);

%get lattice ladder coefficients, then compute the exact lattice ladder

[M, C] = tf2latc(Bk, Ak);

[Y] = latcfilt(M, C, Hf);

plot\_freq\_responses(Fd, Y, 48e3, 9)

ScaleFactor = 512;

%compute scaled coefficients and round to integer

scaled\_M0 = round(M(1)\*ScaleFactor);

scaled\_M1 = round(M(2)\*ScaleFactor);

scaled\_M2 = round(M(3)\*ScaleFactor);

scaled\_M3 = round(M(4)\*ScaleFactor);

%get the quantized version of the coefficients

unscaled\_M0 = scaled\_M0/ScaleFactor;

unscaled\_M1 = scaled\_M1/ScaleFactor;

unscaled\_M2 = scaled\_M2/ScaleFactor;

unscaled\_M3 = scaled\_M3/ScaleFactor;

%compute scaled coefficients and round to integer

scaled\_C0 = round(C(1)\*ScaleFactor);

scaled\_C1 = round(C(2)\*ScaleFactor);

scaled\_C2 = round(C(3)\*ScaleFactor);

scaled\_C3 = round(C(4)\*ScaleFactor);

scaled\_C4 = round(C(5)\*ScaleFactor);

%get the quantized version of the coefficients

unscaled\_C0 = scaled\_C0/ScaleFactor;

unscaled\_C1 = scaled\_C1/ScaleFactor;

unscaled\_C2 = scaled\_C2/ScaleFactor;

unscaled\_C3 = scaled\_C3/ScaleFactor;

unscaled\_C4 = scaled\_C4/ScaleFactor;

%create arrays for coefficients

scaled\_M = [scaled\_M0, scaled\_M1, scaled\_M2, scaled\_M3];

scaled\_C = [scaled\_C0, scaled\_C1, scaled\_C2, scaled\_C3, scaled\_C4];

unscaled\_M = [unscaled\_M0, unscaled\_M1, unscaled\_M2, unscaled\_M3];

unscaled\_C = [unscaled\_C0, unscaled\_C1, unscaled\_C2, unscaled\_C3, unscaled\_C4];

%compute rounded version of lattice ladder

[Y\_q] = latcfilt(unscaled\_M, unscaled\_C, Y);

plot\_freq\_responses(Fd, Y\_q, 48e3, 11)

**3) Limit Cycles from Quantization of Multiplier Outputs**

Compose a Matlab function that will compute and plot the unit sample responses for two filters with the following difference equations:

y[n] = x[n] – A1y[n-1] (unquantized)

.

yq[n] = x[n] – Q{ A1y[n-1] } (quantized multiplier output)

where Q{ w } is a quantized, finite bit precision version of w,

Q{ w } = round[w \*(2# bits)] /(2# bits)

The function should accept the following input parameters:

1. Feedback loop filter coefficient: A1
2. The number of bits for the multiplier output quantization: # bits
3. The length of the unit sample response sequence to compute & plot.

**Test Results:**

**Test Case: A1 = 0.75; # bits = 7**



**Test Case: A1 = 0.75; # bits = 5**



**Test Case: A1 = 0.75; # bits = 3**



**Test Case: A1 = 0.85; # bits = 5**



**Test Case: A1 = 0.60; # bits = 5**



**Describe your observations** about the effects of changing the number of quantization bits and the A1 filter coefficient on the limit cycle oscillations.

**By increasing the number of quantization bits, you effectively decrease the quantization error but increase the computation time and cycles. Therefore, there is a tradeoff. Furthermore, increasing the A1 coefficient will increase the length of the unit sample response, but the error between points in the response is les than that of lower coefficients.**

**Matlab m-file function code:**

function [y\_n, y\_q, n] = limit\_cycles\_unit\_sample\_response(A\_coeff, bits, size)

%this function takes in a feedback loop coefficient, # of bits to quantize,

%and length of response to compute both the full precision output and

%and quantized output based of the difference equation:

%y[n] = x[n] - A1y[n-1]

%the function will plot the two on a single plot for comparison

%initialize sequences

x\_n = zeros(1,size);

y\_n = zeros(1,size);

y\_q = zeros(1,size);

x\_n(1) = 1;                                 %unit sample

n = 0:size-1;                             %sample integer

A\_1 = A\_coeff;                              %full precision coefficient

A\_q = round(A\_coeff\*2^bits);                %quantized coefficient

%loop transfer function

for i=1:length(x\_n)

    if i == 1                               %first point

        %assume zero initial conditions

        y\_n(i) = x\_n(i);

        y\_q(i) = x\_n(i);

    else

        y\_n(i) = x\_n(i) - A\_1\*y\_n(i-1);     %calc full precision output

%calc quantized output

        y\_q(i) = x\_n(i) - round(A\_coeff\*2^bits\*y\_n(i-1))/(2^bits);

    end

end

%plot both output sequences on same plot

figure

stem(n, y\_n)

hold on

stem(n, y\_q)

legend\_2 = ['Quantized to ', num2str(bits), ' bits'];

legend('Full Precision', legend\_2)

grid

title1 = strcat('Limit Cycles in IIR Filter (A1=',num2str(A\_coeff), ')');

xlabel('Sample Number')

ylabel('Amplitude')

title(title1)

end

**Conclusions:**

*Summarize one or two learnings about the effects of numeric quantization in digital filter implementations that this project helped you understand better. Also describe any particular challenges that you had to overcome, and at least one suggestion for improvement of this lab in the future.*

**Name: Aiku Shintani**

**Conclusions:** Through completion of this project, an understanding of the effects of numeric quantization was established. The effectiveness of numeric quantization was accessed for several test cases including: adjustment of the scale factor, adjustment of the filter architecture to a cascaded one, and adjustment of the filter architecture to a lattice-ladder one. Increasing the scale factor results in an increase in the precision of the quantization and results in the best results. Splitting a 4th order architecture into two individual cascaded 2nd order filters, while using the same scale factor for both, proved the effectiveness of cascading filters. The adjustment of the architecture to a lattice-ladder one proved to be the most effective in obtaining the desired frequency response. The most challenging part of the lab was ensuring that the peak gain would be 1.00000, exactly. If one uses an inadequate K value, the rest of the computations (especially quantization’s), for the remainder of the lab, will be affected negatively.

**Name: Chris Adams**

**Conclusions:** By completing this project, a better understanding of how quantization effects digital filter was achieved. While quantization can cause substantial error, there are methods such as the 2nd order cascaded direct form and the lattice ladder implementations that minimize error. Quantization allows for lower complexity hardware and faster computation which is why it is important to understand the tradeoffs associated with it. One of the harder parts of this experiment was to visually tell the difference between some of the quantization implementations. Some of the methods produce closely accurate results such that it is hard to see graphically. However, by calculating the percent error, it is far easier to compare. I thought this lab was well structured, but if anything could change, I would want to learn about the benefits and drawbacks of different quantization methods on the hardware side.