LabLektion 5 - Implementation av digitala filter II IIR

Simon Berthilsson

January 27, 2015

In this Lab exercise we are to look into some aspects of the IIR filter. IIR filters have several advantages over FIR filters in that they are more general than FIR filters (if you think about it, the FIR filter is just a special case of the IIR filter with a denominator of degree zero). To us, that means that any given filter may be implemented in an IIR filter with at most the same number of taps as an FIR implementation would need. Usually however, fewer taps are needed. The cost at which this comes is the risk of instability and sometimes a non linear phase behaviour which may be undesirable. From an implementation point of view, the instability issue is by far the greatest concern.

1 preparations

Before we get started, download the file "Lab-L5_framework.c" from Studentportalen and set the environment up, as in the previous labs by including the following libraries:

- ABDAC – Audio Bitstream DAC (driver)
- ADC – Analog to Digital Converter (driver)
- DSPLib – Digital signal processing library (service)
- PDCA – Peripheral DMA Controller (driver)
- PM – Power Manager (driver)
- TC – Time/Counter (driver)
- USART – Universal Synchronous/Asynchronous Receiver/Transmitter (driver)
- System Clock Control (service)

1.1 Implementation of a Direct Form II IIR filter

Your assignments will be in **bold font**, things will run more smoothly if you wait until you know what the assignment is before you try to solve it. Implementing IIR filters on direct form is rarely a good idea. One key problem is the sensitivity of the poles of the filter. Another, perhaps more easily grasped issue will be apparent when implementing the filter of Table 1.

Finding the scale factor of an IIR filter requires slightly more work than in the FIR case. The impulse response of a FIR filter is simply the filter coefficients while the IIR filter consists of a numerator, as the FIR does too, but has also a denominator.

Recall the filter structure of the direct form II filter (Figure 1). We identify two nodes (the summation points S_w and S_y at the top of the figure) where the risk of overflow is the greatest. We

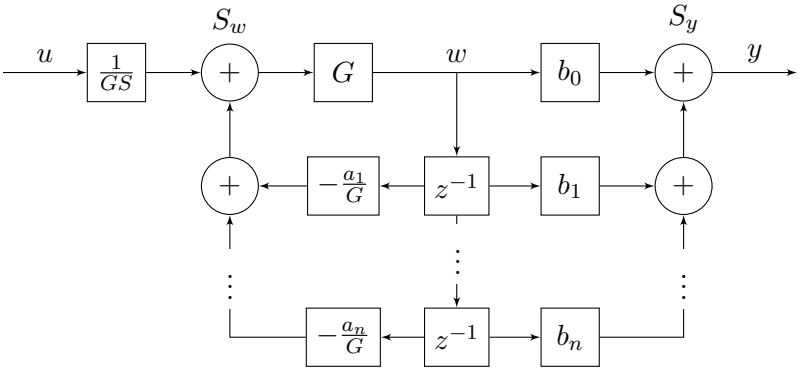

Figure 1: The Direct form II filter structure.

will be safe if we scale the input signal to the filter so that overflow does not occur in any of these points.

The first point is only affected by the denominator of the filter and the input signal. The second point is affected by both the numerator and the denominator. Now, all we need to do is to find the greatest value of any one of the sums S_w and S_y and make sure that this is never greater than one. To this end, we will make use of the filters' impulse response. Making sure that the maximum of the l_1 norm of the impulse response at the point S_w and the l_1 norm of the impulse response at the point S_y is less than or equal to one protects us from overflows.

The function in MATLAB for finding the impulse response of a discrete time IIR filter is impz(Num,Den). Thus, we may use sum(abs(impz(Num,Den))) and sum(abs(impz(1,Den))) to find the scale factor.

Find the scale factor that ensures that the filter at hand will not overflow.

Having found the scale factor, we are moments away from an implementation but we notice that the denominator will cause an immediate overflow if we try to implement the filter in the $Q_{0.15}$ format. This is a common situation in IIR filters and we need a method to handle it. What we do is that we choose a factor by which we scale all elements of the denominator of the filter, making them fit in our chosen numerical interval. If we make sure that the intermediate variable, w, remains unchanged we can be certain that the output will be the same as before. Choosing the factor G, we have at point w:

$$w'(t) = u(t) - \frac{a_1}{G}w(t-1) - \frac{a_2}{G}w(t-2) - ... - \frac{a_n}{G}w(t-n). \tag{1}$$

If we multiply $w'(t)$ by G and divide our input, $u(t)$ by G, we will recreate the original w but with rescaled denominator coefficients that fit our chosen numerical range

$$w(t) = G\left(\frac{u(t)}{G} - \frac{a_1}{G}w(t-1) - \frac{a_2}{G}w(t-2) - ... - \frac{a_n}{G}w(t-n)\right). \tag{2}$$

The schematic representation of the filtering process with scaling and denominator magnitude adjustments is shown in Figure 1.

Keep in mind that the greatest fraction that we can multiply by in the $Q0.15$ context is one, we can however still use "regular" integer multiplications by using the operator "*" instead of dsp16_op_mul, make therefore sure to choose an integer for G. Moreover, if G is a power of two, we can use right and left bit shift operations (>> and <<). The framework code comes with a bit shift implemented (change the variable n in the definition section of the file to change the number of steps to shift) but it should not be very hard to change if you need a scaling that is not a power of two.

It is good practice to do multiplications with the reciprocal instead of divisions, as multiplications are much more efficiently implemented in the processor.

Note that this methodology will introduce some noise in the computations as we first scale down the input signal u causing round off errors, then scale it back up again amplifying those errors.

Compute the overflow-safe $Q_{0.15}$ representation of the filter in Table 1 using the l_1 Norm.

The high gain of the denominator forces us to remove a great deal of the energy of the input signal in order to avoid overflow in the summing stages. This makes the implementation extremely noisy, we reduce the input so much that it can barely be represented with the precision we have and then proceed to amplify what is left.

Table 1: The IIR filter taps

	Tap Value	
Delay [samples]	Numerator	Denominator
0	0.005599682867751	1.000000000000000
1	-0.008456725415086	-3.464005228254822
2	0.011996777730602	4.661145168012926
3	-0.008456725415086	-2.874353823291369
4	0.005599682867751	0.684263180613489

1.2 Implementation of a cascaded form IIR filter

Hopefully (and likely), a cascaded form of the filter will be more successful. The cascaded structure consists of several sections of second order direct form II implementations. By breaking up the numerator and denominator in second order sections, the gain difference may be reduced and the robustness increased.

Partition the filter into second order sections using the MATLAB command [SOS,S] = tf2sos(Num,Den).

Now, every section of the filter will require the same treatment as the direct form II implementation we went through above. We want to reduce the input of every section so that no saturation will occur in that section. As the output of one section is the input of the next, the scaling of the input may be applied to the numerator coefficients of the previous section instead of in a separate step (see Figure 3). This way, all but one scaling multiplications can be done beforehand and all the scaling we need to do is to divide (multiply with the inverse) the input signal with the scale factor of the first section.

Note also that the norm of the impulse response at S_y is the absolute maximal value that each filter section will output. If the norm at S_w is greater than that at S_y, that will be used for the scaling of that section and the maximal output value from that section will not be one, but $\frac{S_y}{S_w}$. When the scaling of the following section is computed, it is assumed that the input to that section will be, at most, one. This possible discrepancy may cause us to scale the signal unnecessarily, increasing the noise of the filtered signal. Make therefore sure that you reduce the scale factor of each section by the maximum output from the previous;

$$S_{n+1} = max(||S_{w_{n+1}}||, ||S_{y_{n+1}}||)||S_{y_n}^{scaled}||, \tag{3}$$

where $||S_{w_k}||$ is the norm of the impulse response of section k at point S_w and the superscript 'scaled' indicates that the norm is computed for the section with proper scaling implemented.

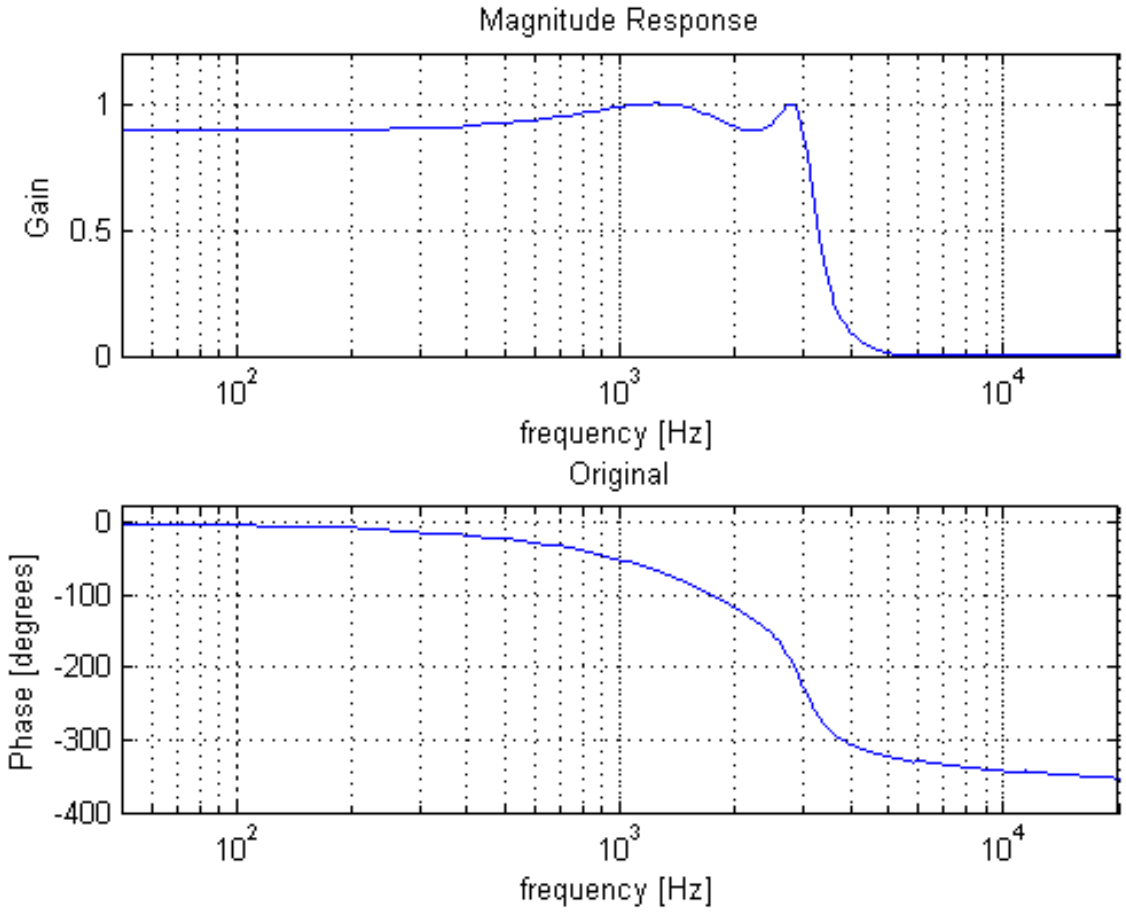

Figure 2: Desired characteristics of the IIR filter

The algorithm for finding the final filter coefficients and scale factors would thus be:

1. Define a filter (the filter is given in this exercise).
2. Convert the filter to second order sections (using tf2sos in matlab).
3. Make sure that the denominator of the first section fits in our chosen numerical representation. If it does not, divide by an appropriate integer, G_1.
4. Make sure that the numerator coefficients fit our current representation by following the same procedure as above. Call this scale factor N_1.
5. Make sure that the section does not overflow internally by dividing the input to the section by max of the norm of the filter at points S_w and S_y. We call this scale factor S_1.
6. Repeat for the second section.
7. Compute the scaled version of the norm of the previous section at S_y i.e. $C_2 = \frac{||S_{y1}||}{S_1}$.
8. Multiply S_2 by C_2 to compensate for the fact that we implicitly assume that the input to a section is at most one. In cascade this is not necessarily true as the input to one section is the output from the previous section. Therefore we need to include the magnitude of the output of that section in our calculations.
9. divide the numerator coefficients of section 1 by S_2C_2 in order to save some multiplications in the computations. (This way, we do not have to divide the input to each section by a factor, this is already done in the output of the previous stage).
10. Repeat for all following sections.
11. Find the output gain adjustment factor by multiplying all N_n by the scale factor from the function tf2SOS and all scale factors S_nC_n.
12. Convert to Q0.15.
13. Implement the filter, make sure that the input is divided by S_1G_1, that the intermediate result of every section, $w_n^{'}$ is rescaled by G_n in order to restore the intended w_n.

The function tf2sos will affect the gain of the filter, in order to recreate the desired output of the filter, we need to address this. The output S of tf2sos holds the inverse of the gain change, if we multiply the output of our filter with this value, we will reconstruct our original filter.

When you find the scaling needed to avoid internal overflows in the filtering process, these will likely affect the total gain of the filter as well. Remember these and multiply S with them in order to cancel their effect at the output. Additionally, you may have to scale down the numerator coefficients if they are greater than one. If so, remember that scale factor and multiply the total output gain factor with that number in order to get a true gain behaviour of the implemented filter.

Fill out Tables 2 and 3 and implement the filter given in Table 1 with the correct scaling so that the filter will not overflow.

The desired filter characteristics are shown in Figure 2.

Verify, against figure 2 that the implemented filter follows the specification.

Table 2: Filter parameters

	Tap Value	
Variable	Section 1	Section 2
G		
N		
S_d		
S_y		
S		
C	-x-	
$1/(S_2C_2)$	-x-	
C	-x-	

Table 3: Filter parameters ($Q_{0.15}$)

Section 1				
$1/G_1S_1$	G_1	b_0/G_2S_2	b_1/G_2S_2	b_2/G_2S_2
x	x	x	$-a_1/G_1$	$-a_2/G_1$
x	x	x		

Section 2				
G_2	b_0	b_1	b_2	G_{out}
x	x	x	$-a_1/G_2$	$-a_2/G_2$
x	x	x		

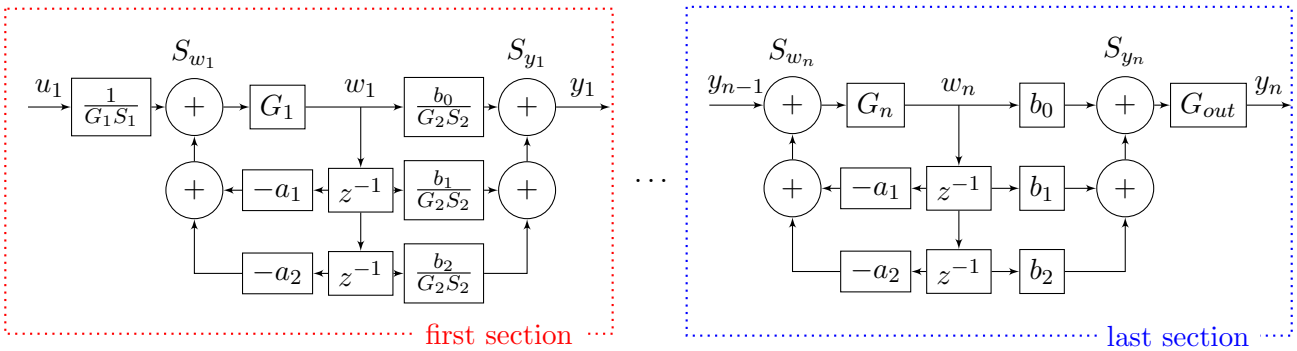

Figure 3: Cascaded filter structure