# A fully differential switched capacitor wavelet filter

**Report by: Michiel Grashuis**

**Supervisor: Dr. Wouter Serdijn**

**The faculty of electronics, mathematics and computer science**

**In fulfillment of the requirements for the degree of**

**Master of Science**
**In electrical engineering**

**June 2009**

Thesis committee:

Prof. Dr. John R. Long
Dr. ir. Wouter A. Serdijn
Dr. ir. M. Keijzer

# Table of Contents

## 1 Abstract

The purpose of this project is to develop a low power, signal specific analog to digital conversion suitable to detect the QRS complex for use in pacemakers. In order to obtain this we have resorted to wavelet analysis in front of the analog to digital conversion. The wavelet analysis is done by means of a switched capacitor filter. The design process involved three design steps. The first is deriving the filter characteristics in a state-space representation by means of a SVD approximation. Next we optimize the state-space representation for dynamic range. The last step involves matrix manipulation to convert the state-space representation into a tridiagonal Schwarz form which enables easy implementation and offers low sensitivity with respect to component variations.

# 2 Introduction

## 2.1 ECG

In this thesis we focus on application in a pace-maker. Pace-makers monitor the rhythm of the heart beat and, if needed, the pace maker will generate a pulse to stimulate the heart. A pace maker thus needs to decide if a stimulation pulse is needed. In order to do this the pace maker will have to be able to monitor the heartbeat closely. A recording of a human heart beat is known as Electrocardiograph (ECG) and is shown in figure 2.1.

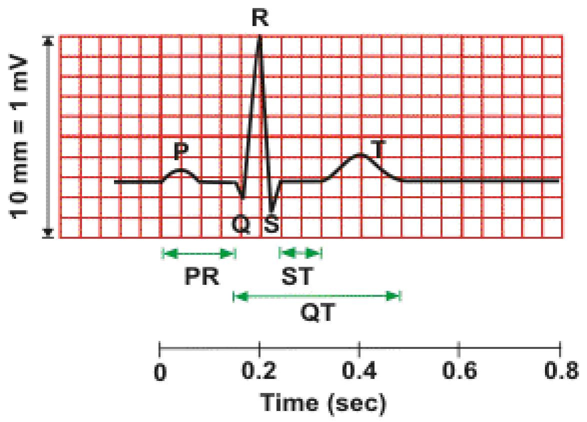


Figure 2.1 A human ECG

From figure 2.1 one can see that the subsequent waves of the heart signal are labeled P, Q, R, S and T. The waves denote the depolarization and repolarization of the muscles in the heart. In this thesis we focus on detecting the R wave or QRS complex. This is the biggest peak and the main feature of the ECG. The QRS complex denotes the activity of the heart pumping the blood around in your body. This QRS complex is always the dominant feature of the ECG and if it is missing your heart needs some pacing.

Electrodes placed on the heart detect the heart signal and an analog to digital converter (ADC) in the pace maker converts this analog signal into a digital signal. The ADC's in some pacemakers use their full resolution to do this. High resolution ADC's consume a lot of power. If the resolution of the ADC can be reduced then we can save some power. In this case we aim to lower the power consumption by making the ADC signal specific. We do this by studying the effects of implementing a filtering action in or around the ADC. By creating a filtering action before, inside or after the ADC, we can lower the ADC's resolution and hence lower the power consumption.

## 2.2 ADC

The options to choose from when combining a wavelet filter with an ADC are the following three.

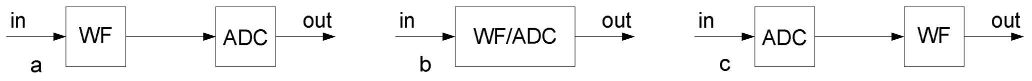


Figure 2.2. a: wavelet filter before ADC. b: wavelet filter inside ADC. c: wavelet filter after ADC

By putting the wavelet filter after the ADC we cannot gain any reduction in the ADC's resolution. The ADC still has to convert the analog signal into a digital signal using its full resolution. This option is therefore not a good solution and will not be considered further. The other two options will be discussed further in a subsequent chapter.

## 2.3 The need for wavelet transforms

Fourier analysis is a tool that allows the designer to look at the frequency contents of a signal. A Fourier analysis performed on a time domain signal thus yields the frequency spectrum of that signal. This is a very useful and frequently used tool in circuit design. It has however the drawback that during the transformation to the frequency domain all time domain information is lost. This means that from the frequency domain spectrum one cannot see at what time a particular frequency component occurred in the time domain signal. Look for example at figure 2.3.

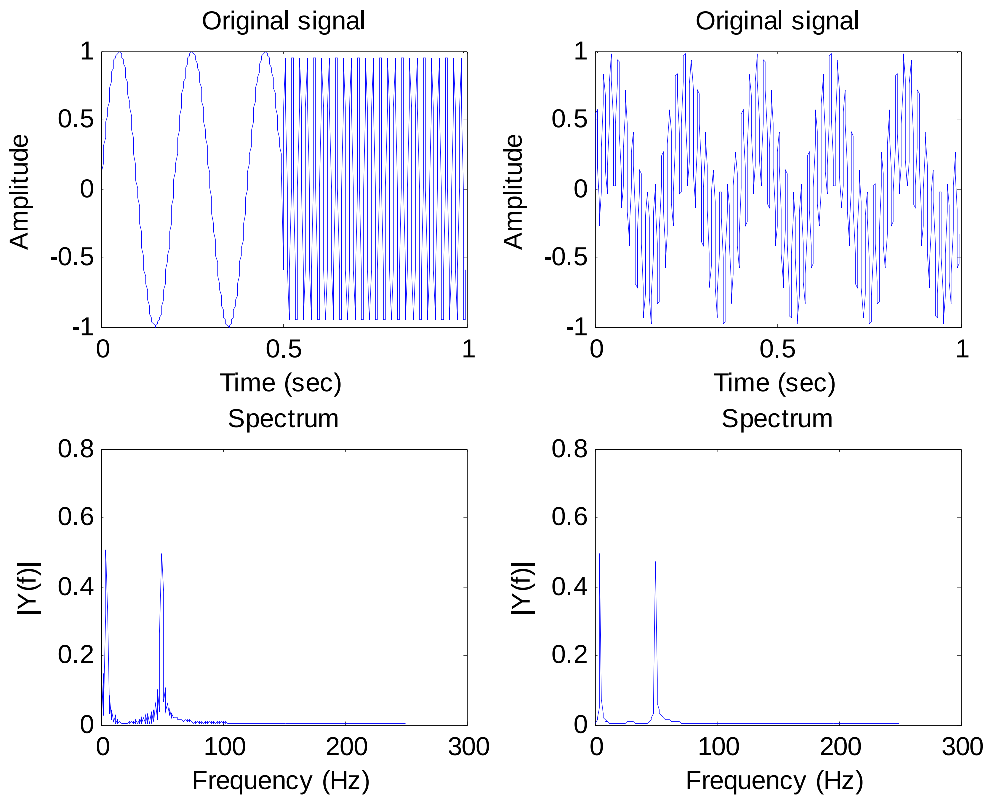


Figure 2.3. Two time domain signals and their corresponding Fourier spectra.

In the first column of figure 2.3 one can see a time domain signal on the top and its corresponding Fourier spectrum on the bottom. In the second column of figure 2.3 one can see another time domain signal on the top and its corresponding Fourier spectrum on the bottom. The two time domain signals look very different but their Fourier spectra are almost the same. This is because Fourier analysis only correlates the input signal with sine waves. This means that Fourier analysis only calculates how much of a certain frequency component is present in the time domain signal. For the 1st column in figure 2.3 the two time domain signals span half the plot (Low frequency for $0<t<0.5$. and a higher frequency for $0.5<t<1$) whereas in the 2nd column of figure 2.3 both signals span the entire plot.

Wavelet analysis however does preserve the time domain information [1]. Wavelet analysis is different from Fourier analysis in that it correlates the input signal with a wavelet instead of sine waves. This allows the wavelet analysis to preserve the time domain information of the input signal. The term wavelet actually means “small wave”. It has a compact support.

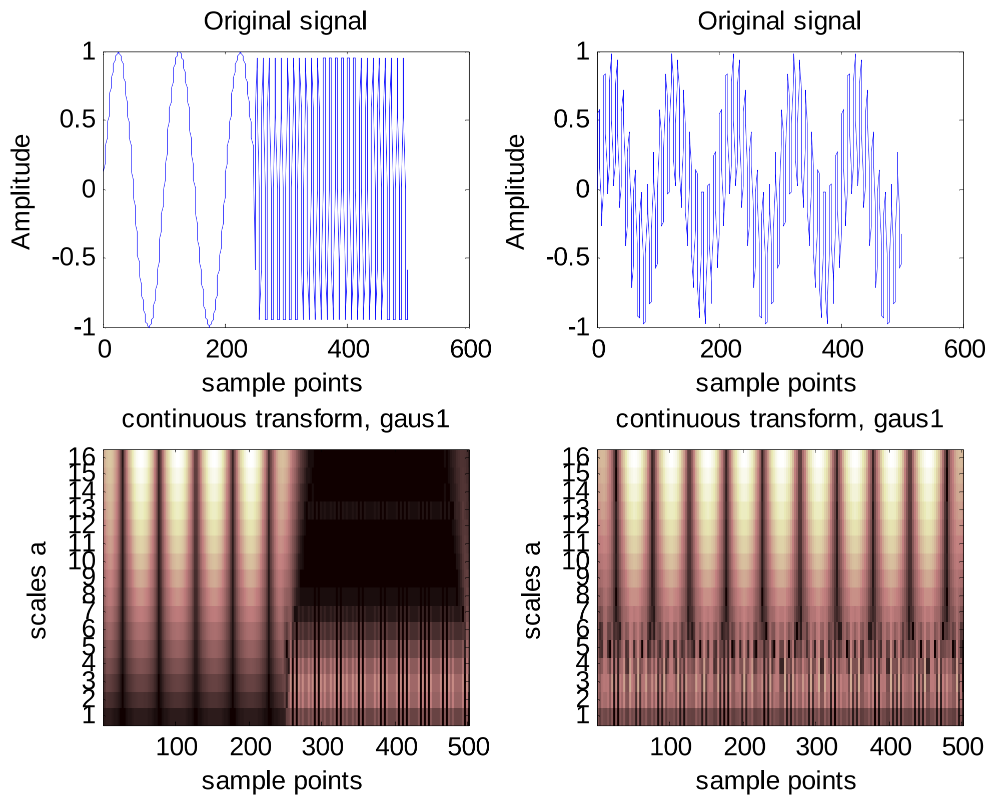


Figure 2.4. Two time domain signal and their corresponding wavelet analysis.

Look for example at figure 2.4. The time domain signals on the top row are the same as in figure 2.3 but now on the bottom row we have the corresponding wavelet analysis. In this case the wavelet analysis is performed for different scales (1 to16 on the y axis) using a gaus1 (1st derivative of a Gaussian) as the mother wavelet and plotted versus time (x-axis). The scale of the wavelet analysis can be seen as a zooming parameter in the frequency domain. A low scale means that the wavelet analysis zooms in on the higher frequency components in the signal and ignores the low frequency components, and vice versa for a high scale. In figure 2.4 this is nicely illustrated. The fluctuations in the input signal are expressed as alternating black and white colors. For a low scale the high frequency component of the input signal is clearly detected. For an intermediate scale the high and low frequency components are both detected. This is because the low and high frequency are not very far apart. For a high scale only the low frequency component is detected. This shows that wavelet analysis is suitable to break a signal up into frequency components and preserve the time domain information. However wavelet analysis can only do this using several scales. This means that if we want to make a wavelet ADC we need to use a filter bank where each filter corresponds to a wavelet scale.

Because we focus on a pacemaker application in this thesis we have also performed a wavelet analysis on an ECG signal. On the top row of figure 2.5 you can see an ideal ECG without noise and interference. On the middle row of figure 2.5 you can see the same ECG signal but now it is corrupted by 50 Hz interference. While on the top row of figure 2.5 all the waves are clearly visible on the middle row only the QRS complex can be identified. All the other waves are drowning in the 50 Hz interference. On the third row you can see the result of the wavelet analysis performed on the ECG signal plus interference using the gaus1 as the mother wavelet. This wavelet analysis is also performed with 16 scales. For the lower scales the higher frequency components in the signal become visible. In this case it is only the 50 Hz interferer that is visible. For the higher scales the lower frequency components become visible. In this case the ECG signal is visible again.

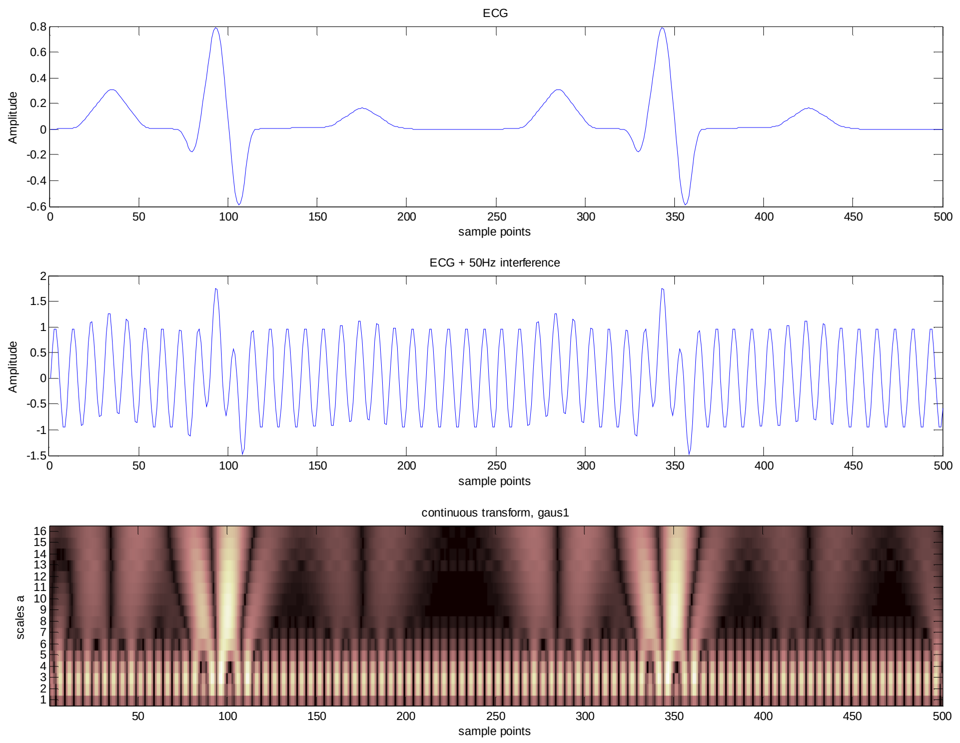


Figure 2.5 A wavelet analysis on an ECG signal.

## 2.4 Choosing the mother wavelet

The wavelet transform is written as:

$$W_f(\tau,a) = \frac{1}{\sqrt{a}} \cdot \int_{-\infty}^{\infty} f(t) \cdot \varphi^*\left(\frac{t-\tau}{a}\right) \cdot dt \qquad 2.1$$

In equation 2.1 f(t) is the input signal or the signal to be analyzed, a is the scale parameter and τ is the translation parameter in time. The factor $1/\sqrt{a}$ is used for energy normalization. $\varphi$ is the mother wavelet or wavelet base and * denotes taking the complex conjugate transpose. The wavelet base is a small oscillatory wave or wavelet. The wavelet base has to satisfy the following conditions.

$$\int_{-\infty}^{\infty} \varphi(t) \cdot dt = 0 \qquad 2.2$$

$$\int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|} \cdot d\omega = C_\Psi < \infty \qquad 2.3$$

Equation 2.2 denotes an oscillatory wavelet base with zero mean. Or a wavelet base with finite duration. Equation 2.3 is the admissibility condition. The wavelet base needs to satisfy this condition in order to be able to reconstruct the inverse transform. From equation 2.1 it is observed that performing a wavelet transform is actually performing a convolution of the input signal with scaled versions of the wavelet base. In this case scaled can also be read as dilated because, as the scaling parameter a increases, the impulse response of the wavelet filter becomes longer i.e. it has a longer duration. Therefore each scale corresponds to a certain band in the frequency domain. A wavelet filter is thus a filter that computes the wavelet transform for one scale.

There is a wide variety of wavelet bases. Choosing the "best" wavelet base can be very challenging. But in general one can say that the more the wavelet base looks like the signal to be analyzed the better. For this reason we opted for a first derivative of a Gaussian (gaus1) as the mother wavelet. A picture of a gaus1 is shown in figure 2.6.

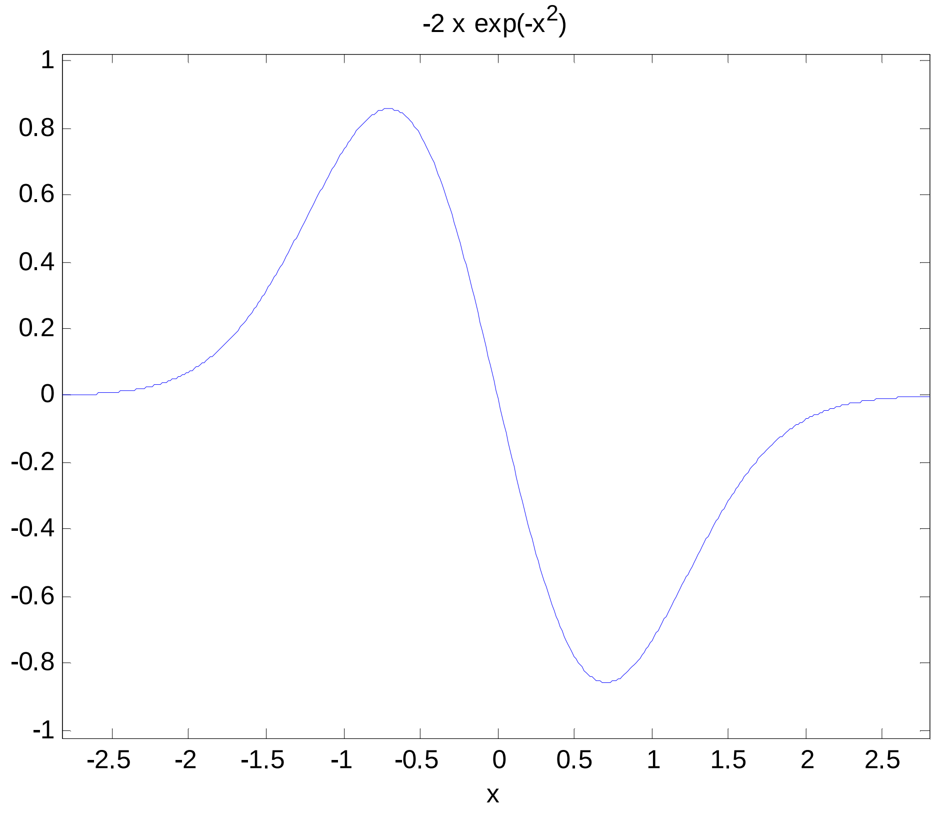


Figure 2.6. The first derivative of a Gaussian.

# 3 System design

As mentioned before putting the wavelet filter after the ADC is not a good solution. The remaining two options are discussed in the following subsections.

## 3.1 Wavelet in ADC

Here we consider the wavelet filter inside the ADC. At this stage a choice has to be made concerning the ADC topology. Our choice for a suitable candidate ADC was the sigma delta ADC shown in figure 3.1. We have chosen for this topology because its noise shaping character seemed well capable of adjustment to wavelet filtering. With this type of ADC low pass and band pass ADCs have already been made.

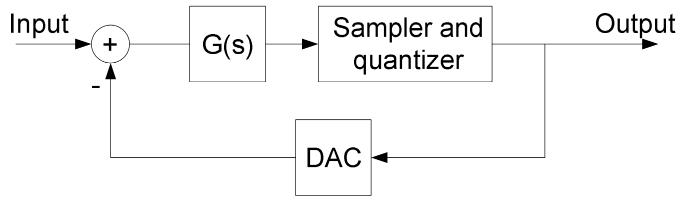


Figure 3.1. A sigma delta converter.

The block diagram in figure 3.1 is a low pass sigma delta converter if G(s) is 1/s and has been simulated in Matlab Simulink. For the input a sinusoid of 1001Hz and amplitude of 0.9V was used. The input and output signals are shown in figure 3.2.

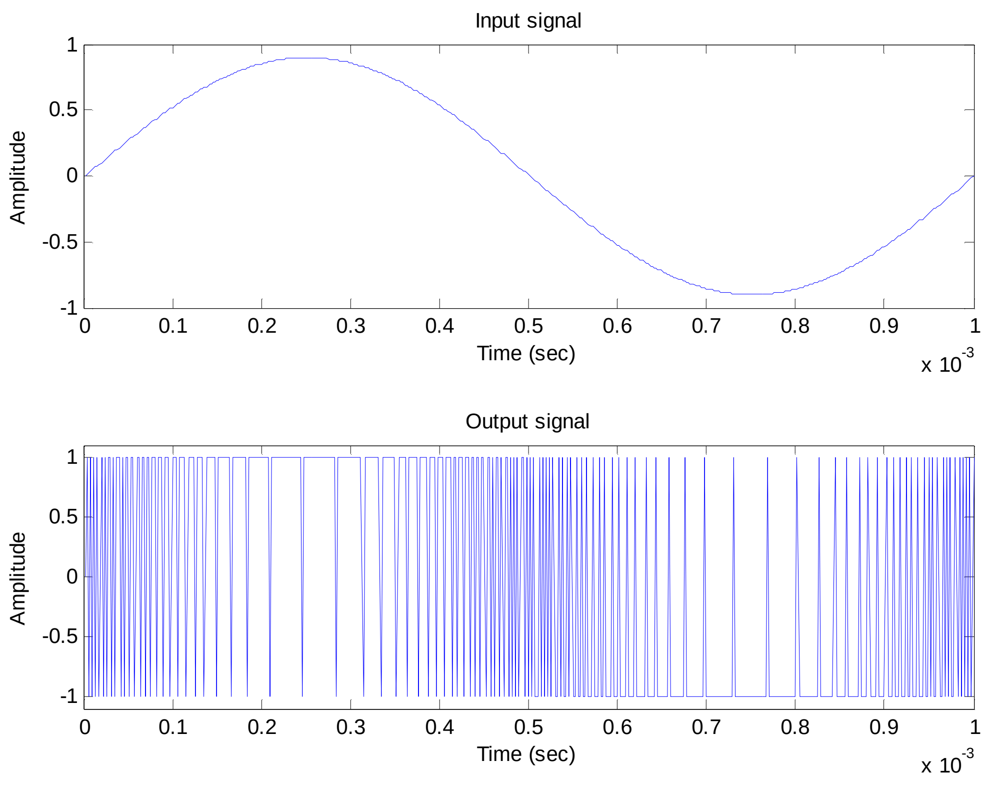


Figure 3.2. The input and output signals of a sigma-delta modulator.

From figure 3.2 one can see that the output signal of the sigma delta converter is a train of pulses. In fact, the sigma delta modulator is a pulse density modulator. As the amplitude of the input signal increases the number of positive pulses increase and if the amplitude of the input signal decreases then the number of negative pulses increase. If the input signal is zero then the output signal contains an equal number of positive and negative pulses. The input signal is thus contained in the density of the negative and positive pulses. By performing a Fourier analysis on the output signal we can see the frequency content of the signal. This allows us to demonstrate the noise shaping character of the sigma delta converter. The spectrum plot is shown in figure 3.3. The input tone is clearly visible and that the quantization noise underwent a high pass filtering action. The signal transfer is different from the quantization noise transfer because the quantization noise is generated at the quantizer and the signal is applied to the input. The signal transfer from input to output is called STF and is calculated as follows:

$$\mathrm{STF} = \frac{G(s)}{1 + G(s)} \qquad 3.1$$

The noise transfer from the quantizer to the output is calculated as follows.

$$\mathrm{NTF} = \frac{1}{1 + G(s)} \qquad 3.2$$

In this case G(s) is 1/s or a continuous time integrator. Substituting 1/s in both equations yields a low pass filter for signals applied to the input and a high pass filter for the noise generated by the quantizer.

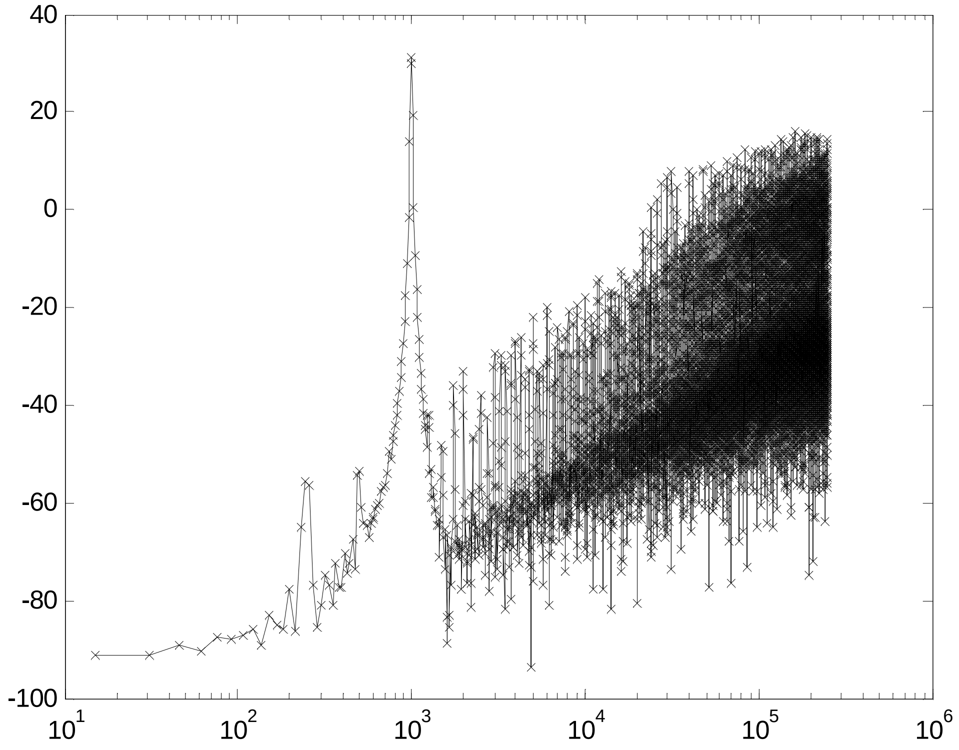


Figure 3.3. A plot of the output of a sigma delta converter.

A band pass sigma delta has a structure shown in figure 3.4.

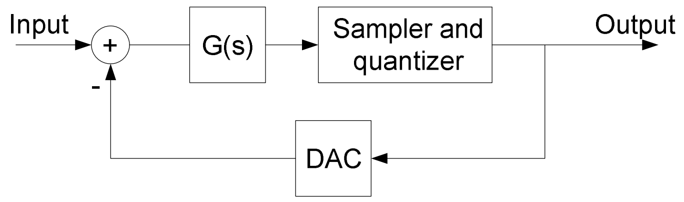


Figure 3.4. A band pass sigma delta converter.

In figure 3.4 G(s) can have the following form.

$$G(s) = \frac{s}{s^2+100s+(4\pi 10^4)^2} \qquad 3.3$$

In this case G(s) is a band pass filter. The bode plot of G(s) looks as follows.

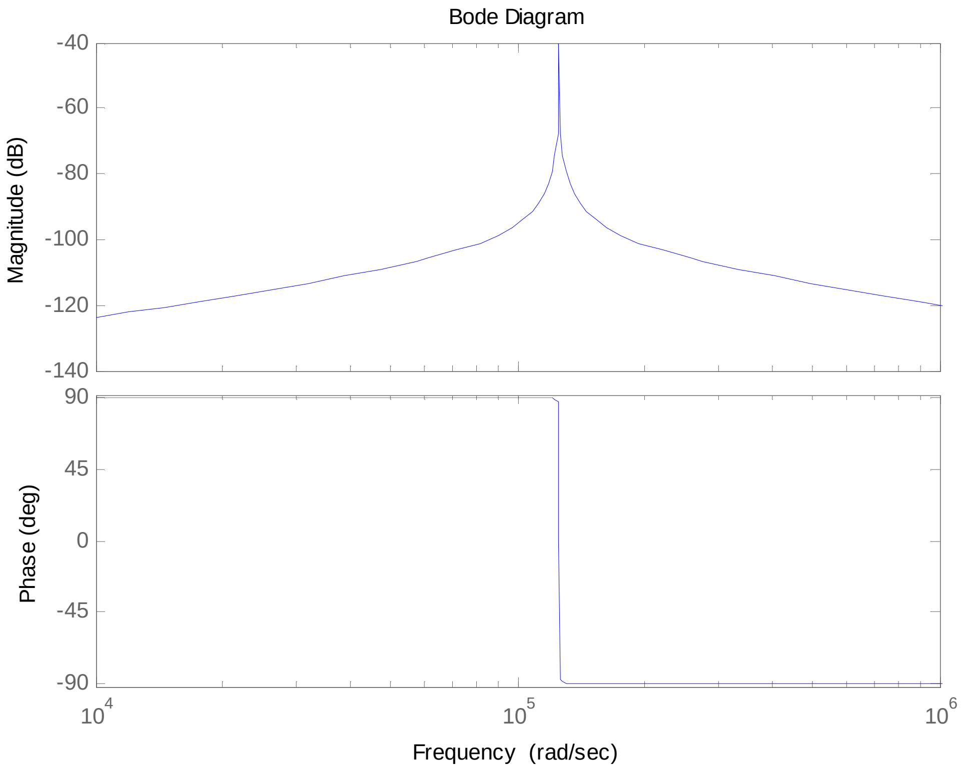


Figure 3.5 The bode plot of 3.3.

One can see that this is a band pass filter with a very narrow bandwidth. The centre frequency lies at 20k Hz. With this band pass filter inside the loop of the sigma delta converter we obtain a band pass sigma delta converter. The spectrum plot of the band pass sigma delta converter is shown in figure 3.6

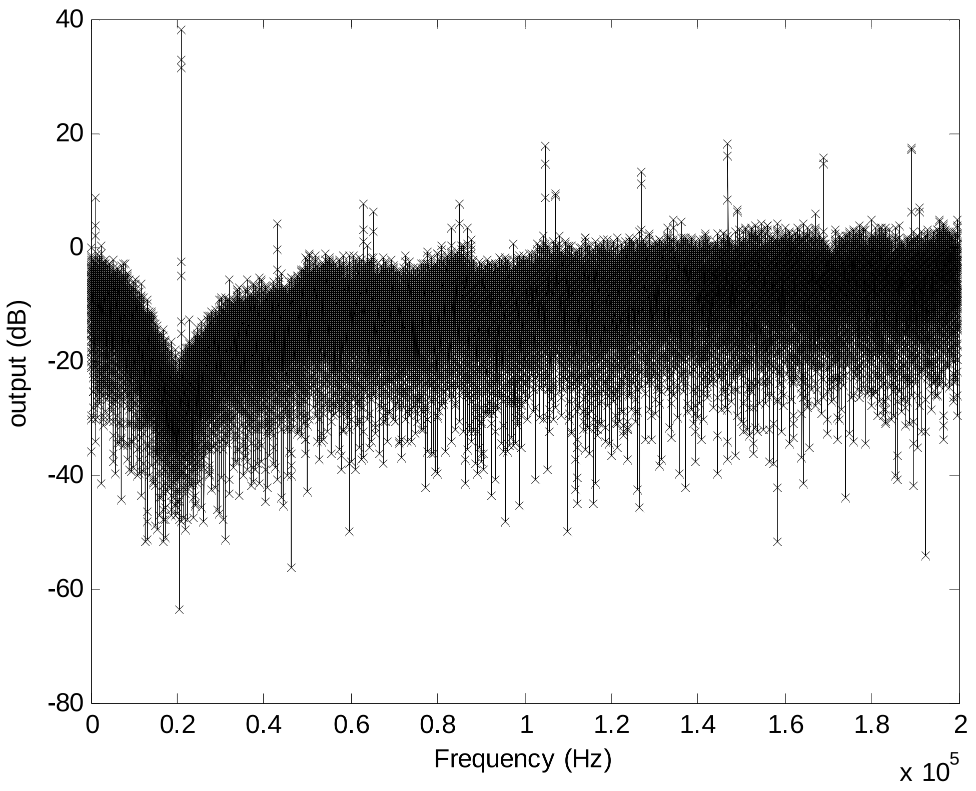


Figure 3.6 A plot of the output of a band pass sigma delta converter

Now the noise transfer is a band reject filter and the signal transfer is a band pass filter. So the noise is effectively “shaped” outside the band of interest.

Since a wavelet filter is a kind of band pass filter, the step from band pass sigma delta to wavelet sigma delta didn’t seem too big. We have tried to make a wavelet sigma delta converter by implementing a G(z) that behaves like a wavelet filter.

$$G(z) = \frac{0.1455 \cdot z^4 - 0.5752 \cdot z^3 + 0.8526 \cdot z^2 - 0.5616 \cdot z + 0.1387}{z^5 - 4.985 \cdot z^4 + 9.942 \cdot z^3 - 9.913 \cdot z^2 + 4.942 \cdot z - 0.9855} \qquad 3.4$$

This discrete time transfer function models a fifth order wavelet filter. By inserting this filtering action in side the loop of a sigma delta converter we hoped to obtain a sigma delta converter that shows wavelet behavior. We did not succeed in this. One reason for this is that the system became unstable. The instability was caused by the quantizer primarily. The quantizer breaks the loop and because of this the loop gain becomes dependent on the input signal. This causes the poles of the system to move outside the unit circle and thus the system is unstable. Another reason why we did not succeed is that for a sigma delta converter that was stable it did not show the desired wavelet behavior. This again was due to the quantizer. As the pole positions became dependent on the input signal, the wavelet behavior was corrupted. Since we haven’t found a solution to this problem we decided to abandon this approach and focus on the last remaining option, a wavelet filter before the ADC.

## 3.2 Wavelet before ADC

Since two out of three options have been rejected only one remains. If the analog signal undergoes some analog preprocessing then the resolution of the A to D converter can be lower and hence consume less power. In this thesis an analog preprocessing by means of a wavelet filter is proposed.

# 4 Approximation of the mother wavelet

To make an electronic filter that implements a wavelet transfer we need to make a filter that has as its impulse response the mother wavelet. The mother wavelet, until now, has been given as a time domain function i.e. infinite order. Electronic filters have only a limited number of poles i.e. limited order. This calls for approximation.

## 4.1 Pade

The Pade based approximation [1], [2] offers an approximation in the Laplace domain. It takes the time domain function as input and gives a transfer function as output.

The time domain function can be any of the known wavelet bases (Morlet, Daubechies and Gaus1). The approximation starts by taking the Laplace transform of the time domain function. This gives a closed form expression in the s-domain. On this Laplace transform we will perform a Taylor series expansion (A(s)) up to a certain order.

$$A(s)=\sum_{i=0}^{9} a_i \cdot s^i \qquad 4.1$$

This Taylor expansion can then be put into a transfer function ($P_L(s)/Q_M(s)$) by using Pade. Here $P_L(s)$ denotes the numerator of the transfer function and it is a polynomial of order L. $Q_M(s)$ is the denominator of the transfer function and it is a polynomial of order M.

$$A_N(s) - \frac{P_L(s)}{Q_M(s)} = 0 \qquad N = L + M + 1 \qquad 4.2$$

From equation 4.2 one can see that the Taylor series expansion has to be done up to order N. In order to solve equation 4.2 we multiply by $Q_M(s)$, rearrange and set the normalization condition $Q_M(0) = 1$. This results in the following set of equations.

$$\begin{aligned}
a_0 &= p_0 \\
a_1 + a_0 \cdot q_1 &= p_1 \\
a_2 + a_1 \cdot q_1 + a_0 \cdot q_2 &= p_2 \\
\vdots \qquad & \quad \vdots \\
a_L + a_{L-1} \cdot q_1 + \cdots + a_0 \cdot q_L &= p_L \\
a_{L+1} + a_L \cdot q_1 + \cdots + a_{L-M+1} \cdot q_M &= 0 \\
\vdots \qquad & \quad \vdots \\
a_{L+M} + a_{L+M-1} \cdot q_1 + \cdots + a_L \cdot q_M &= 0
\end{aligned} \qquad 4.3$$

In this set of equations all a's are known. The $P_L(s)$ and $Q_M(s)$ vectors are to be determined. We can find a solution for vector $Q_M(s)$ by solving the sub set of equations that equal zero.

$$\begin{aligned} a_{L+1} + a_L \cdot q_1 + \cdots + a_{L-M+1} \cdot q_M &= 0 \\ \vdots \qquad\qquad & \quad \vdots \\ a_{L+M} + a_{L+M-1} \cdot q_1 + \cdots + a_L \cdot q_M &= 0 \end{aligned} \tag{4.4}$$

By solving for the null space for equation 4.4 we obtain a solution for $Q_M(s)$. Once we have a solution for $Q_M(s)$ we can find a solution for $P_L(s)$ by substituting $Q_M(s)$ in equation 4.3. Now we have arrived at a continuous time transfer function.

$$\frac{P_0 \cdot s^M + P_1 \cdot s^{M-1} + \cdots + P_M}{Q_0 \cdot s^L + Q_1 \cdot s^{L-1} + \cdots + Q_L} = 0 \tag{4.5}$$

This approximation can be summarized in a block diagram.

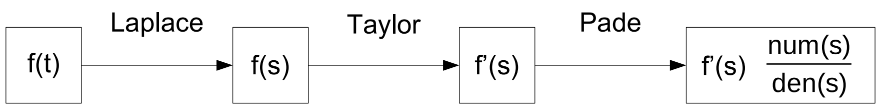


Figure 4.1 Block diagram of the Pade approximation.

Having done the Pade approximation we have arrived at a continuous time transfer function in the s-domain.

A good property of this approximation method is that it is straight forward and easy to implement in Matlab. However a drawback is that Pade does not always yield a stable system especially for higher order approximations.

Later on in this thesis we will talk about state-space descriptions rather than transfer functions. A transfer function can easily be converted into a state-space description by a standard method. Take for example the following transfer function.

$$H(s) = \frac{n_1 \cdot s^3 + n_2 \cdot s^2 + n_3 \cdot s + n_4}{s^4 + d_1 \cdot s^3 + d_2 \cdot s^2 + d_3 \cdot s + d_4} \tag{4.6}$$

This transfer function can directly be inserted in a state-space description.

$$\frac{dx(t)}{dt} = \begin{bmatrix} -d_1 & -d_2 & -d_3 & -d_4 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot x(t) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot u(t) \tag{4.7}$$

$$y(t) = \begin{bmatrix} n_1 & n_2 & n_3 & n_4 \end{bmatrix} \cdot x(t) \tag{4.8}$$

This state-space description is called controllable canonical form because it is guaranteed to be controllable.

An example approximation using Pade is shown in figure 4.2. For generating this plot we used a time shifted (1.7 sec) gaus1. The order of approximation is 5 and the Taylor expansion is done in s = 0.

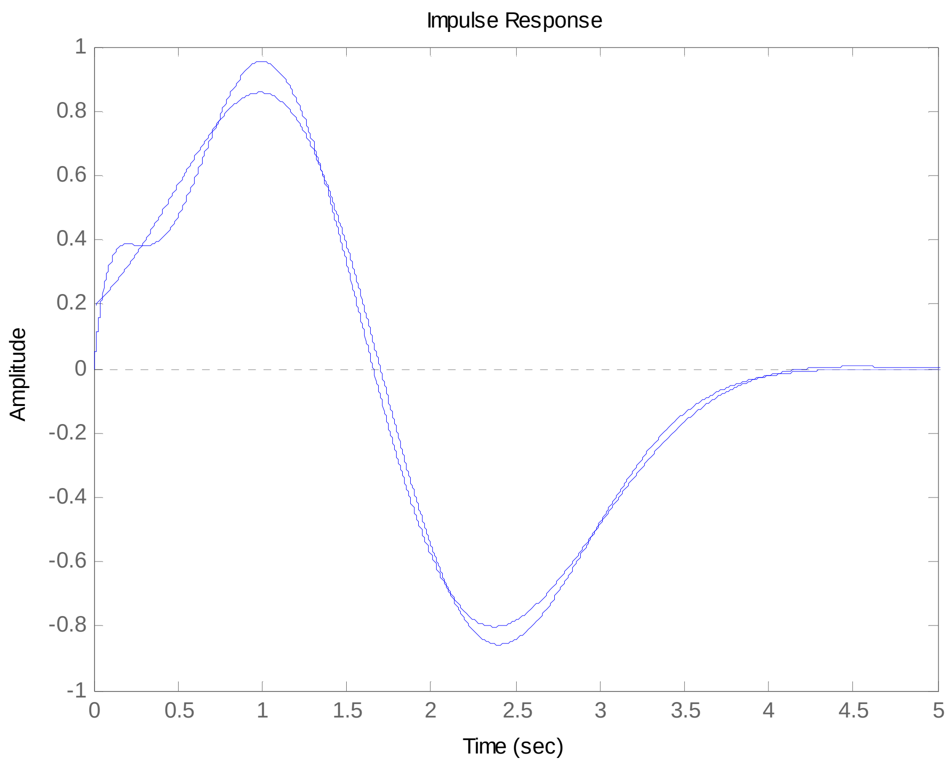


Figure 4.2. The original gaus1 overlapped with the Pade based approximation.

As one can see Pade does not perform very well. The approximated gaus1 has a wiggle at the start of the signal. This comes from the fact that the Taylor expansion has been done in s = 0. This is necessary because an expansion done in any other point than s = 0 results in ringing. Also at t = 1s the top of the approximation overshoots and at t = 2.4s the top of the approximation is too shallow. These are all unwanted effects as they affect the outcome of the wavelet transformation. The mean square error is calculated and is found to be 0.00110.

## 4.2 SVD

This approximation method relies on the Singular Value Decomposition or SVD [3]. The approximation is done in the discrete time domain rather than the continuous time domain. It takes the sampled version of the time domain signal as input and produces a state-space description as output.

Linear systems have an input output relation which is defined by the transfer matrix T.

$$y = T \cdot u \tag{4.9}$$

Here the input and an output respectively look like the following.

$$u = \begin{bmatrix} \dots & u_{-1} & \boxed{u_0} & u_1 & \dots \end{bmatrix}^T \tag{4.10}$$

$$y = \begin{bmatrix} \dots & y_{-1} & \boxed{y_0} & y_1 & \dots \end{bmatrix}^T \tag{4.11}$$

The boxed entry denotes the instance at t = 0. The matrix T is the matrix that represents our system. Because it is a linear system the impulse response of the system can be calculated as follows.

$$T \cdot \begin{bmatrix} \dots & 0 & \boxed{1} & 0 & \dots \end{bmatrix}^T = h = \begin{bmatrix} \dots & 0 & 0 & \boxed{h_0} & h_1 & h_2 & \dots \end{bmatrix}^T \tag{4.12}$$

Because T is an LTI system it can be shown that equation 4 holds for an impulse as input at any time t. This means that the system matrix T must have a lower triangular structure.

$$T = \begin{bmatrix} \ddots & 0 & 0 & 0 & 0 & 0 \\ \ddots & h_0 & 0 & 0 & 0 & 0 \\ \ddots & h_1 & h_0 & 0 & 0 & 0 \\ \ddots & h_2 & h_1 & h_0 & 0 & 0 \\ \ddots & h_3 & h_2 & h_1 & h_0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \tag{4.13}$$

Matrix T is constant along its diagonals because of time invariance and lower diagonal due to causality. Of course T is now a matrix of infinite dimension which is not really workable. One way of reducing the size of T is to only take the inputs in the past and the outputs in the future. This will result in the following structure:

$$\begin{bmatrix} \vdots \\ x \\ x \\ \overline{y_0} \\ y_1 \\ y_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} x & 0 & 0 & 0 & 0 & 0 & 0 \\ x & x & 0 & 0 & 0 & 0 & 0 \\ x & x & x & 0 & 0 & 0 & 0 \\ \cdots & h_2 & \overline{h_1}| & \boxed{x} & 0 & 0 & 0 \\ & h_3 & h_2 & x & x & 0 & 0 \\ \iddots & & h_3 & x & x & x & 0 \\ & & \vdots & x & x & x & x \end{bmatrix} \cdot \begin{bmatrix} \vdots \\ u_{-3} \\ u_{-2} \\ \underline{u_{-1}} \\ 0 \\ 0 \\ \vdots \end{bmatrix} \tag{4.14}$$

The idea here is that only inputs in the past generate outputs in the future. By doing this we obtain a T matrix which is at least semi infinite. The system in equation 6 can also be written as:

$$y_+ = H \cdot u_- \quad \Leftrightarrow \quad \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 & \cdots \\ h_2 & h_3 & & \\ h_3 & & \ddots & \\ \vdots & & & \end{bmatrix} \cdot \begin{bmatrix} u_{-1} \\ u_{-2} \\ u_{-3} \\ \vdots \end{bmatrix} \qquad 4.15$$

Matrix H is called the hankel matrix. Its columns consist of shifted versions of the impulse response. From this hankel matrix H we will calculate the A B C and D matrixes of the state-space realization we want to obtain.

The impulse response of a state space system looks as follows:

$$h = \begin{bmatrix} \cdots & 0 & 0 & 0 & \boxed{D} & CB & CAB & CA^2B & \cdots \end{bmatrix}^T \qquad 4.16$$

One can see now that matrix H has great similarities with h. The columns of H are shifted versions of the impulse response. So H has a structure which looks like:

$$H = \begin{bmatrix} CB & CAB & CA^2B & \cdots \\ CAB & CA^2B & & \\ CA^2B & & \ddots & \\ \vdots & & & \end{bmatrix} \qquad 4.17$$

In our case we know the H matrix. We construct it from sampled and shifted versions of the required impulse response. Having done that we need to extract the A, B, C and D matrixes from H. We do this by decomposing matrix H into two matrixes, a matrix O called the observability matrix and a matrix K called controllability matrix.

$$H = O \cdot K \qquad 4.18$$

Where:

$$O = \begin{bmatrix} C & CA & CA^2 & \cdots \end{bmatrix}^T \qquad 4.19$$

$$K = \begin{bmatrix} B & AB & A^2B & \cdots \end{bmatrix} \qquad 4.20$$

A reliable way to compute the O and K matrix is by performing a Singular Value Decomposition (SVD) on the hankel matrix. An SVD on the hankel matrix looks like this:

$$H = U \cdot \Sigma \cdot V^H \qquad 4.21$$

In the SVD the matrix $\sum$ is a diagonal matrix with the singular values arranged in decreasing order. Of course if H is infinite then $\sum$ is also of infinite order. In this case $\sum$ is rounded off to a rank 5 matrix. This will result in a 5th order system which can be implemented in a switched capacitor topology. The O and K matrix can be calculated from the SVD as follows:

$$O = \hat{U} \cdot \hat{\Sigma}^{1/2} \qquad 4.22$$

$$K = \hat{\Sigma}^{1/2} \cdot \hat{V}^{H} \qquad 4.23$$

Where: $\hat{U}$, $\hat{\Sigma}$ and $\hat{V}$ are the rank 5 approximation of the original U, $\sum$ and V matrixes respectively. Now having obtained the O and K matrixes we need to extract the A matrix of the state space description from this. A simple way of doing this is by dividing the O or K matrix in two and divide them to extract the A matrix. In this example we choose the O matrix.

$$O = \begin{bmatrix} \overbrace{C \quad CA \quad CA^2 \quad \cdots \quad CA^{n-1}}^{O_x} \quad CA^n \end{bmatrix}^T \qquad 4.24$$

$$O = \begin{bmatrix} C \quad \overbrace{CA \quad CA^2 \quad \cdots \quad CA^{n-1} \quad CA^n}^{O_y} \end{bmatrix}^T \qquad 4.25$$

One can now see that:

$$O_x \cdot A = O_y \qquad 4.26$$

From 4.26 the A matrix is easily found by dividing $O_y$ by the pseudo inverse of $O_x$.

$$A = O_x^{+} \cdot O_y \qquad 4.27$$

Now only the B and C matrixes of a state space description have to be found. They are easily found by taking the first row and first column of O and K respectively.

$$B = K(:,1) \qquad 4.28$$

$$C = O(1,:) \qquad 4.29$$

Matrix D is set to zero in this case. This is done because we want to obtain a causal system.

In this thesis we will make use of this approximation method. The reason we choose this method is because in comparison to other approximations it gives the best approximation and it yields state-space descriptions with the least component spread. An example approximation is given below. In this case the approximation order is 5 and the time shift is 1.7 seconds, the same as for the Pade approximation.

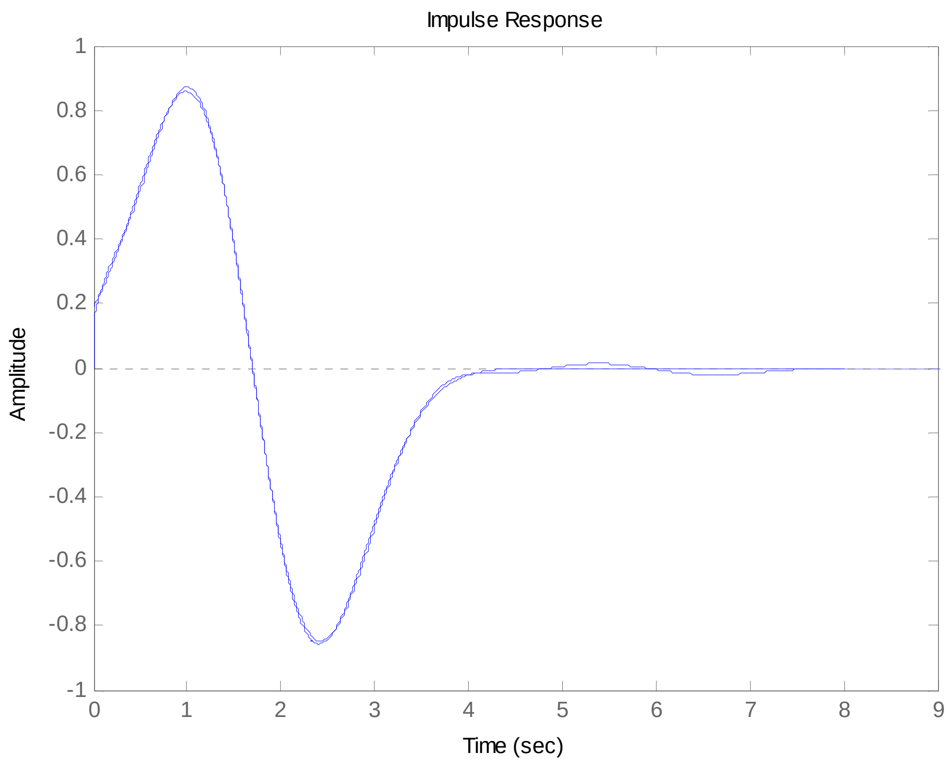


Figure 4.3. The original Gaus1 overlapped with the approximation.

For the approximation in figure 4.3 one can see that the approximated version overlaps the original gaus1 pretty well. The wiggle is no longer there and the performance at the tops is also better. The mean square error is calculated and found to be 0.00011. This is a factor of ten better than the Pade approximation.

## 4.3 L2-approximation

The L2 approximation [4] is a method that works directly in the time and frequency domain. A drawback of this method is that it largely depends on the starting point that is used. The L2 approximation has local optima and a bad choice of the starting point can cause the iterative process of approximation to end up in a local optimum instead of the global optimum. For this reason we have chosen not to use this method.

# 5 State-space descriptions

LTI systems are usually described by their transfer functions or impulse responses. This transfer function completely defines the relationship between the input and the output but it doesn't contain any information about the internal states of a system. State-space descriptions [5] however do describe the behavior of the internal states of a system.

## 5.1 State-space

A state-space description is a mathematical model that describes the behavior of a system by its input, output and state variables. These variables are related to each other by 1st order differential equations. State-space descriptions can be either in the continuous time domain or discrete time domain. State-space descriptions provide a convenient and compact way of modeling a system.

### 5.1.1 Continuous time state-space descriptions

The state-space description of a system looks as follows:

$$\frac{dx(t)}{dt} = A \cdot x(t) + B \cdot u(t) \tag{5.1}$$

$$y(t) = C \cdot x(t) + D \cdot u(t) \tag{5.2}$$

In these equations x(t) represents the state of the system u(t) is the input to the system and y(t) is the output of the system. A is the n by n system matrix, B is the n by m input matrix and C is the p by n output matrix. D is the direct feed through matrix of size p by m. Here n is the number of state variables, p is the number of inputs and m the number of outputs. The state-space descriptions can also be given in the Laplace domain.

$$s \cdot x(s) = A \cdot x(s) + B \cdot u(s) \tag{5.3}$$

$$y(s) = C \cdot x(s) + D \cdot u(s) \tag{5.4}$$

Also a block diagram of a state-space description is depicted in figure 5.1. A state of a system, which is a collection of system variables values, completely defines the entire system at any given time. This means that no information of the past of the system helps in predicting the future of the system, if the states of the present are known.

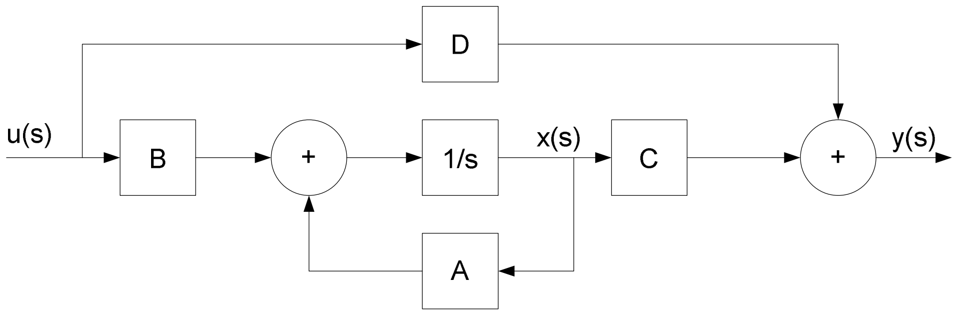


Figure 5.1: Block diagram of a continuous time state-space description

From a state-space description the transfer function of the system can be derived.

$$H(s) = C \cdot \left(s \cdot I - A\right)^{-1} \cdot B + D \qquad 5.5$$

But of course in a transfer function all the information about the internal states is lost.
A big advantage of the state space descriptions is that we have control over the internal states. This means that we can actually optimize the system with respect to dynamic range. In order to optimize for dynamic range we need to calculate the controllability (K) and the observability (O) matrixes. The official definition of these matrixes is.

$$K=\begin{bmatrix} B & A \cdot B & A^2 \cdot B & \cdots & A^{n\text{-}1} \cdot B \end{bmatrix} \qquad 5.6$$

$$O=\begin{bmatrix} C & C \cdot A & C \cdot A^2 & \cdots & C \cdot A^{n\text{-}1} \end{bmatrix}^T \qquad 5.7$$

Controllability of a system means that the states of a system can be steered from any initial condition to any final value within a given time. Observability of a system means how well the internal states can be inferred from the output.

$$G = \int_0^\infty e^{At} \cdot B \cdot B^T \cdot e^{A^T t} \cdot dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} F\left(j\omega\right) \cdot F^*\left(j\omega\right) \cdot d\omega \qquad 5.8$$

$$W = \int_0^\infty e^{A^T t} \cdot C^T \cdot C \cdot e^{At} \cdot dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} G^*(j\omega) \cdot G(j\omega) \cdot d\omega \qquad 5.9$$

G is the controllability gramian and W is the observability gramian. For a stable system the matrixes G and W satisfy the following two lyapunov equations.

$$A \cdot G + G \cdot A^T + B \cdot B^T = 0 \qquad 5.10$$

$$A^T \cdot W + W \cdot A + C^T \cdot C = 0 \qquad 5.11$$

Gramian matrixes are positive semi definite, and every positive semi definite matrix is the gramian matrix for some set of vectors. The gramian matrix of any orthonormal basis is the identity matrix.

### 5.1.2 Discrete time state-space descriptions

The discrete time domain state-space description looks as follows.

$$x(n+1) = A \cdot x(n) + B \cdot u(n) \qquad 5.12$$
$$y(n) = C \cdot x(n) + D \cdot u(n) \qquad 5.13$$

Here x(n), u(n)and y(n) represent the state, input and output of the system respectively. And also A is the n by n system matrix, B is the n by m input matrix and C is the p by n output matrix. D is the direct feed through matrix of size p by m. Here n is the number of state variables, p is the number of inputs and m the number of outputs. After applying the z-transform to equations 5.12 and 5.13 we obtain the following equations.

$$z \cdot x(z) = A \cdot x(z) + B \cdot u(z) \qquad 5.14$$
$$y(z) = C \cdot x(z) + D \cdot u(z) \qquad 5.15$$

A block diagram of a discrete time system is shown in figure 5.2. It is exactly the same as the continuous time system

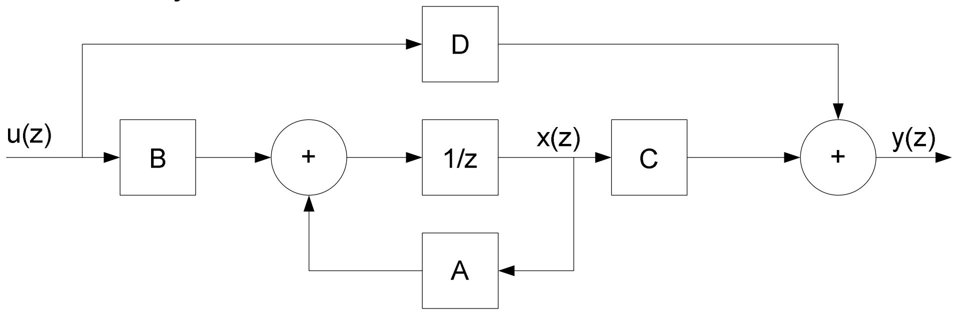


Figure 5.2. Block diagram of a discrete time state-space system.

From this discrete time state-space description we can also derive the transfer function.

$$H(z) = C \cdot (z \cdot I - A)^{-1} \cdot B + D \qquad 5.16$$

The lyapunov equations for a discrete time state-space system look a little different from for a continuous time system.

$$A \cdot G \cdot A^T - G + B \cdot B^T = 0 \qquad 5.17$$
$$A^T \cdot W \cdot A - W + C^T \cdot C = 0 \qquad 5.18$$

The controllability matrix (K) and observability matrix (O) of discrete time systems are calculated in a similar manner as for the continuous time systems.

$$K = \begin{bmatrix} B & A \cdot B & A^2 \cdot B & \cdots & A^{n-1} \cdot B \end{bmatrix} \qquad 5.19$$
$$O = \begin{bmatrix} C & C \cdot A & C \cdot A^2 & \cdots & C \cdot A^{n-1} \end{bmatrix}^T \qquad 5.20$$

The gramians are now easier to calculate then for the continuous time case. We don’t have to calculate the exponent of a matrix. We just have to calculate a matrix multiplication.

$$G=\left[K\cdot K^{T}\right] \tag{5.21}$$

$$W=\left[O^{T}\cdot O\right] \tag{5.22}$$

Gramian matrixes are positive semi definite, and every positive semi definite matrix is the gramian matrix for some set of vectors. The gramian matrix of any orthonormal basis is the identity matrix.

## 5.2 State-space realization

Standard discrete time state-space descriptions are based on discrete time delays as active elements. However our system will be build in a switched capacitor technology. This means that the active elements are discrete time integrators or accumulators. To accommodate for this type of active element we have to make a small modification to the first state-space equation.

$$z\cdot x(z)=A\cdot x(z)+B\cdot u(z) \tag{5.23}$$

Now if we take the left hand side of the equation and add and subtract one from z we obtain the following equation.

$$(z+1-1)\cdot x(z)=A\cdot x(z)+B\cdot u(z) \tag{5.24}$$

After rearranging this equation we arrive at a new first state-space equation.

$$(z-1)\cdot x(z)=(A-1)\cdot x(z)+B\cdot u(z) \tag{5.25}$$

From this we can see that the A matrix has to be modified. Simply subtracting the identity matrix from A is sufficient.

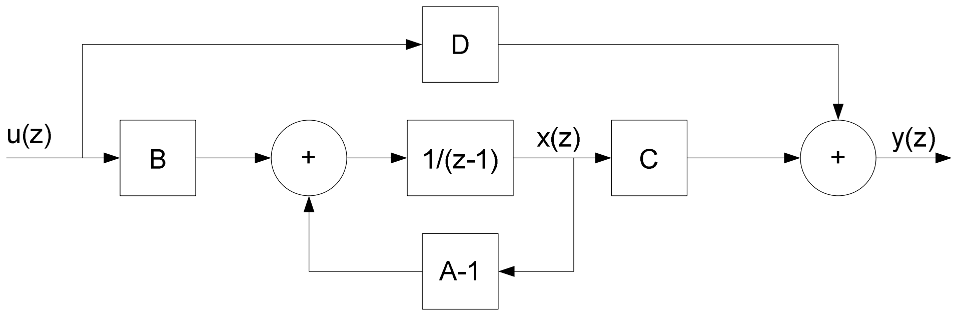


Figure 5.3. Block diagram of a discrete time state-space system with discrete time integrators.

### 5.2.1 Dynamic range optimization

The dynamic range of a system is defined as the ratio of the maximum signal swing and the minimum signal swing that a system can handle. From the thesis of Guillen [6] and Rocha [10] it is shown that the dynamic range of a system can be expressed as.

$$\mathrm{DR} = \frac{\left(\frac{\mathrm{M}}{\delta(\mathrm{p})}\right)^2 \cdot \mathrm{TR(GQ)}}{\max_\mathrm{i} \cdot \mathrm{G_{ii}}} \cdot \frac{1}{\gamma \sum_\mathrm{i} \frac{\alpha_\mathrm{i}}{\mathrm{C_i}} \cdot \mathrm{W_{ii}}} \qquad 5.26$$

Where:

- M is the maximum output amplitude of the integrators.
- δ(p) is a nonlinear monotonically increasing function of the fraction of time p that integrator i is allowed to clip.
- TR is the trace of a matrix, the sum of the elements on its main diagonal.
- Q is the state weighing matrix: $Q = C^TC$.
- $\gamma = \xi 2kT$ is the integrator noise figure with T is temperature in Kelvin, k is the Boltzmann's constant and $\xi \geq 1$ is a constant.
- $\alpha_i = \sum_j |a_{ij}|$ is the absolute sum of the elements of the i^th row of A.
- $C_i$ is the i^th integrator capacitance.
- $G_{ii}$ are the main diagonal elements of G.
- $W_{ii}$ are the main diagonal elements of W.

In this thesis we will also make use of this equation to calculate the dynamic range expressed in dB's.

In order to maximize the dynamic range of the system the maximum distortion-less signal that the system can handle needs to be maximized and the noise (minimum signal) needs to be minimized. These requirements mean that the controllability gramian needs to be the identity matrix. For the observability gramian a similar thing happens. The observability gramian needs to be diagonal as well. The entries on the main diagonal do not need to be the same. Having obtained the correct G and W gramians the state space description can now be optimized for dynamic range. This is done by performing similarity transforms on the state-space matrixes. The result of these transforms is this:

$$\mathrm{A_T} = \mathrm{T^{-1}} \cdot \mathrm{A} \cdot \mathrm{T} \qquad 5.27$$

$$\mathrm{B_T} = \mathrm{T^{-1}} \cdot \mathrm{B} \qquad 5.28$$

$$\mathrm{C_T} = \mathrm{C} \cdot \mathrm{T} \qquad 5.29$$

$$\mathrm{D_T} = \mathrm{D} \qquad 5.30$$

In order to find a correct transformation matrix T we start by looking at the lyapunov equation. For the matrixes A and B, or input pair (A, B), the controllability matrix is G (lyapunov equation). If the system is transformed by means of a similarity transform then input pair (A, B) becomes ($TAT^{-1}$, TB) and G becomes $TGT^T$. This can be verified by substitution into the lyapunov equation. The new G matrix is now required to be the identity matrix. Which means that $TGT^T = I$. After rearranging this equation we arrive at $T^{-1}T^{-T} = G$. This means that $T^{-1}$ has to be a square root of G.

After the first transform we have that G is the identity matrix I. The newly obtained state space descriptions are optimized for maximum output swing and thus yield a good dynamic range for the system. The second step in the optimization procedure is to diagonalize the observability gramian. This step is similar to the first optimization step and therefore not further discussed here.

Now that we have optimized the state-space equation would like to know how much we have gained by this optimization process. We can check this by comparing the Dynamic range calculation before and after the optimization process.

$DR_{before} = 140.9dB$
$DR_{after} = 162.7dB$

One can clearly see that we have gained little over 20 dB here. The calculations have been done on a 5th order SVD approximation of a gaus1. The state-space equations are not shown here for brevity. The only drawback of this state space description is that the matrixes are fully dense. To make sparser matrixes the system can be decomposed into a Schwarz form [7].

### 5.2.2 Schwarz

The Schwarz form of the A, B, C and matrixes look like this.

$$A_T = \begin{bmatrix} -a_{11} & -a_{21} & 0 & 0 & 0 \\ a_{21} & 0 & -a_{32} & 0 & 0 \\ 0 & a_{32} & 0 & -a_{43} & 0 \\ 0 & 0 & a_{43} & 0 & -a_{54} \\ 0 & 0 & 0 & a_{54} & 0 \end{bmatrix} \quad 5.31$$

$$B_T = \begin{bmatrix} b_1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad 5.32$$

$$C_T = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \end{bmatrix} \quad 5.33$$

$$D_T = \begin{bmatrix} 0 \end{bmatrix} \quad 5.34$$

To obtain a state space description that has a Schwarz form we start by looking at the properties of the A and B matrices.

If we substitute G = I into the Lyapunov equation we arrive at the following equation.

$$A + A^T = -B \cdot B^T \qquad 5.35$$

In our case we are working with a single input system. This means that the B matrix has only one nonzero entry, namely the first entry. This results in the right hand side of equation 5.35 to be almost the zero matrix. Only the top left entry of this matrix is nonzero. This means that matrix A must have the property of being nearly skew symmetric. A skew symmetric matrix has the property that $A^T$ = -A. This means that the rows of A are equal to there corresponding negative columns ($a_{ij} = -a_{ji}$). Here is an example of a skew symmetric matrix.

$$\begin{bmatrix} 0 & 2 & -1 \\ -2 & 0 & -4 \\ 1 & 4 & 0 \end{bmatrix} \qquad 5.36$$

This matrix already has great similarities with the matrix of 5.31 except for two things. In 5.31 $a_{11}$ is nonzero and the upper and lower triangle above and below the sub diagonal are zero respectively. Although in our case $a_{11}$ is nonzero the property $A^T$ = -A still holds and we will use it to turn our state space system into the Schwarz form.

Assuming that the A matrix has the property of skew symmetry it is sufficient that we transform matrix A into an upper-Hessenberg form matrix in order to get to the tri-diagonal Schwarz form of 5.31. A matrix which is upper Hessenberg has the following shape.

$$\begin{bmatrix} 1 & 4 & 2 & 3 \\ 3 & 4 & 1 & 7 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 \end{bmatrix} \qquad 5.37$$

A matrix that is upper-Hessenberg and has the property of being skew symmetric implies a tri-diagonal matrix structure which is exactly what we are looking for because it has a lot of zero entries. In order to find a suitable transformation matrix T we look at the controllability matrix of the system we want. With A upper-Hessenberg and B containing only one nonzero entry the controllability matrix K must be upper-triangular. In other words TK must become upper–triangular. This requires an orthogonal matrix T that converts the original K into an upper-triangular matrix (TK). To write it down differently, K = ($T^{-1}$)(TK). This corresponds to a QR decomposition of the original controllability matrix K.

These matrixes are a lot sparser but of course we gave in a little on the optimal dynamic range. However we still end up with a close-to optimum dynamic range given a system order and decomposition. If we take the same fifth order SVD approximation as in the previous section and decompose it into a Schwarz form we can calculate the dynamic range and compare it to the optimal state-space description.

$DR_{schwarz} = 161.9dB$

This is just a little bit less (0.8dB) than the optimum calculated in the previous section.

# 6 Circuit

The circuit is implemented in a switched capacitor topology [1],[6],[8]. This topology closely resembles the RC integrator. The structure of the RC integrator is shown in figure 6.1.

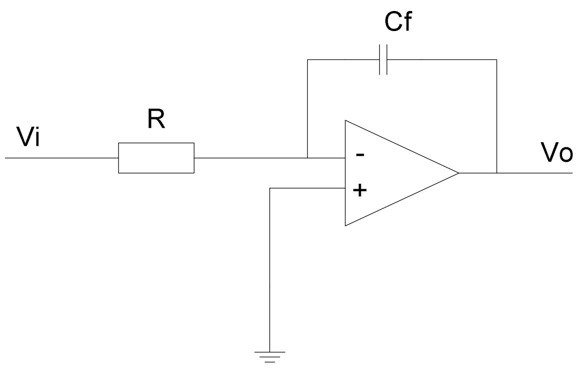


Figure 6.1. The RC integrator.

The output of this circuit is given in equation 6.1.

$$V_0(t) = -\frac{1}{R \cdot Cf}\int_0^t V_i(\tau)\cdot d\tau + V_0(0) \qquad 6.1$$

In a switched capacitor integrator the resistor is replaced by a capacitor and two switches as can be seen in figure 6.2.

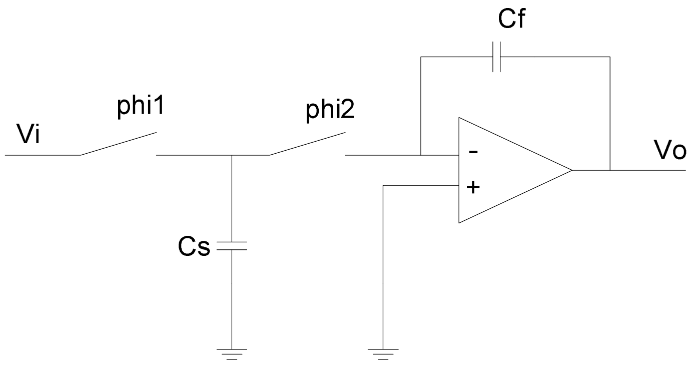


Figure 6.2. A switched capacitor integrator.

The output of this circuit is given in equation 6.2.

$$V_0(k) = -\frac{Cs}{Cf}\cdot\sum_0^t V_i(k) + V_0(0) \qquad 6.2$$

The operation of the switched capacitor integrator is as follows. Capacitor $C_S$ is charged during clock phase phi1. During clock phase phi2 the charge stored on $C_S$ is transferred to $C_F$. The average current can now be calculated as.

$$I_{avg} = \frac{C_S(Vi - V_-)}{T_S} \qquad 6.3$$

Here $I_{avg}$ is the average current, Vi is the input voltage, $V_-$ is the voltage at the virtual ground node and $T_S$ is the sampling time. This equation compares to ohms law by saying that R equals $T_S/C_S$. Thus comparing the continuous time integrator with the discrete time integrator we can see that there is a relation between the resistor and the switched capacitor.

$$R = \frac{1}{Fs \cdot Cs} \tag{6.4}$$

We have to keep in mind here that the sampling frequency needs to be at least twice as high as the input signal or else aliasing will occur. Also, because of the sampling in the time domain of the input signal, there will be some sinc(t) distortion. This is because sampling a signal in the time domain is the same as multiplying the signal with an impulse train. In order to reconstruct the sampled signal we need to use a rectangular baseband filter. This means that the signal is multiplied with the rectangular baseband filter in the frequency domain, which in turn means that the sampled signal is convolved with a sinc(t) function.

One very important aspect here that has not been mentioned jet is that the two clock phases need to be non-overlapping. If this is not the case than charge will inadvertently be lost.

## 6.1 (non)inverting SC integrator

The coefficients in the matrixes of the state-space description will be implemented by capacitor ratios. This ratio will be the ratio between the feedback capacitor and the switched capacitor. We have a fifth order system thus we have five integrators. How all the matrix coefficients are fitted onto the integrators can be seen in the following matrixes.

$$A = \begin{bmatrix} -C_{SA1}/C_{FA} & -C_{SA3}/C_{FA} & 0 & 0 & 0 \\ C_{SB1}/C_{FB} & 0 & -C_{SB2}/C_{FB} & 0 & 0 \\ 0 & C_{SC1}/C_{FC} & 0 & -C_{SC2}/C_{FC} & 0 \\ 0 & 0 & C_{CD1}/C_{FD} & 0 & -C_{SD2}/C_{FD} \\ 0 & 0 & 0 & C_{SE1}/C_{FE} & 0 \end{bmatrix} \tag{6.5}$$

$$B^T = \begin{bmatrix} C_{SA2}/C_{FA} & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.6}$$

$$C = \begin{bmatrix} C_1/C_{FF} & C_2/C_{FF} & C_3/C_{FF} & C_4/C_{FF} & C_5/C_{FF} \end{bmatrix} \qquad 6.7$$

In the above equations one can see where all the capacitor ratios go and how the integrators are connected together. In total there are five integrators labeled A, B, C, D and E. The feedback capacitor corresponding to an integrator can be found in the denominator of the matrix entries. Along each row of the matrix the denominators are constant and the numerators differ. This means that this particular integrator gets its inputs from neighboring integrators. Let's look at the second row for example. It contains two nonzero entries, $C_{SB1}/C_{FB}$ and $-C_{SB2}/C_{FB}$. Here $C_{FB}$ denotes the feedback capacitor of integrator B and $C_{SB(1/2)}$ denotes the switched capacitors of integrator B. These ratios are on positions $a_{21}$ and $a_{23}$ in the matrix respectively. This means that integrator B receives its input signals from the outputs of integrators A and C. For the rest of matrix A a similar thing happens. Matrix B has only one nonzero entry which is actually the input of the filter. Matrix C implements a weighted summation of the integrator outputs.

There are positive and negative entries in matrix A. In order to implement this in a switched capacitor topology we need inverting and non-inverting integrators. The non-inverting SC integrator is shown in figure 6.3. It implements a first order transfer function given by:

$$H(z) = \left(\frac{C_S}{C_F}\right) \cdot \frac{1}{z-1} \qquad 6.8$$

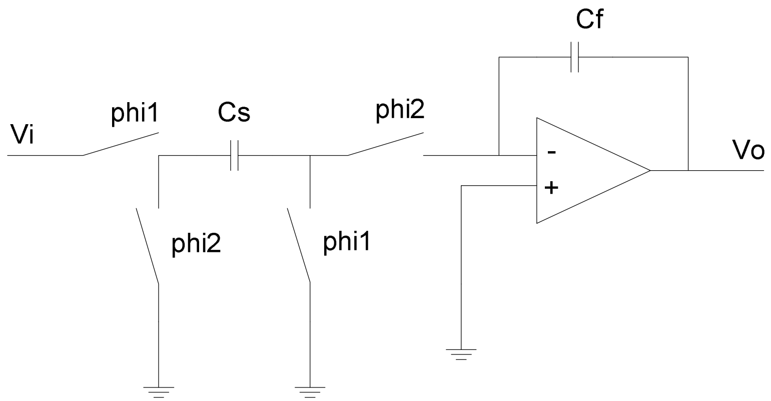


Figure 6.3. Non-inverting SC integrator.

Figure 6.4 shows an inverting SC integrator. Both inverting and non-inverting SC integrators now have four switches instead of two. This is because the topology with four switches suffers les from parasitics.

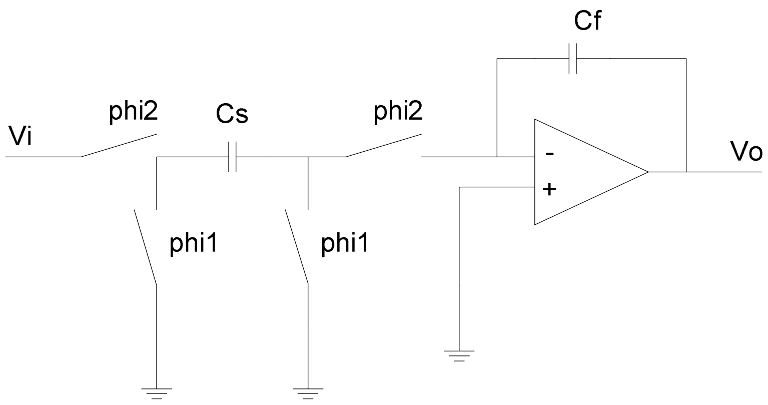


Figure 6.4. Inverting SC integrator.

The transfer function of the inverting SC integrator is given by:

$$H(z) = -\left(\frac{C_S}{C_F}\right)\cdot\frac{1}{z-1} \qquad 6.9$$

Now that we know the main building blocks of the circuit we can do one more optimization step. For a given system order we can scale the feedback capacitors of the integrators so that the relative noise contributions are the same [1]. This is done with the following equation.

$$C_i = \frac{\sqrt{\alpha_i \cdot W_{ii} \cdot K_{ii}}}{\sum_j \sqrt{\alpha_j \cdot W_{jj} \cdot K_{jj}}}\cdot C_{tot} \qquad 6.10$$

Here

$$\alpha_i = \sum_j \left|A_{ij}\right| \qquad 6.11$$

$C_{tot}$ : Is the total feedback capacitance available for the integrators.
$C_i$ : Is the capacitance of the feedback capacitor of $i^{th}$ integrator.

After this last optimization step we can see that the new dynamic range is very close to the optimum.

$DR_{schwarz+capscaling} = 162.1dB$

## 6.2 Circuit realization

The circuit is implemented in a switched capacitor topology in a fully differential fashion. The switches will be implemented by MOS transistors. Also the ideal opamp will be implemented by a real opamp.

### 6.2.1 The switches

The circuit is implemented in a fully differential fashion. This means that the switches will also be implemented in a fully differential fashion. An example of a fully differential switched capacitor integrator is shown in figure 6.5. For implementing the switches we choose to use CMOS switches or transmission gates for the switches connected to the input. CMOS switches are usually employed to allow for a large signal swing. This is also the case for our input switch. All the other switches are NMOS switches. We have used the parasitic insensitive configuration for the switches. This means that the parasitic capacitors have no influence. The switches that have their source and drain both connected to the input of the opamp are dummy switches. They are there to cancel the charge injection from the preceding switches. They are half the size of the preceding switches.

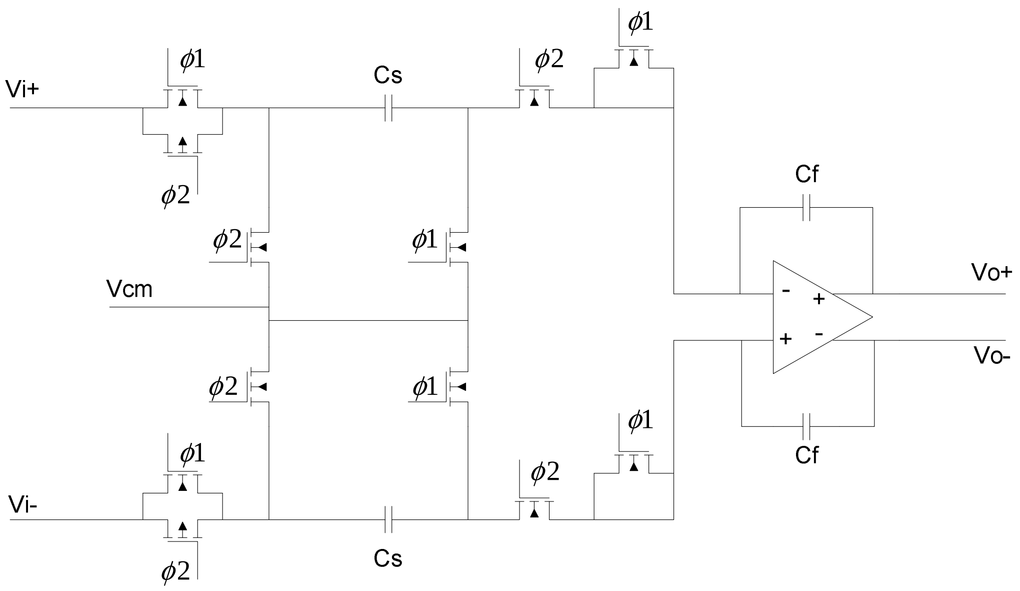


Figure 6.5 A fully differential switched capacitor block.

### 6.2.2 The opamp

The opamp used for implementing the integrator is a fully differential opamp [8] operating in weak inversion. The specifications for the opamp are derived trough simulation. The setup of this simulation is as follows. The entire filter is constructed from the real capacitors and switches but the opamp is an ideal one. By ideal we mean a voltage controlled current source. At the output of this ideal opamp we connect a resistor. The value of the resistor is set to R. Now we can determine the DC voltage gain by playing with the gm of the voltage controlled current source because the DC voltage gain is gm*R. We started off with a high setting of gm and then run a transient simulation to produce the impulse response of the filter. This impulse response is compared to the impulse response of the ideal fifth order filter. If the Impulse response does not differ from the ideal then I lowered the gm of the voltage controlled current source and did the simulation again. This process is repeated until the impulse response starts to differ from the ideal. With this procedure the required DC voltage gain is found to be gm*R is 40dB. So the allowable steady state error is 0.01. The bandwidth and settling time requirement is obtained in a similar manner. A capacitor is connected in parallel to the resistor. The time constant of the resistor and capacitor sets the bandwidth of the opamp and assuming a first order behavior it also sets the gain bandwidth (GBW) of the opamp. The GBW was calculated to be 1 kHz. This figure is calculated using the slides of Klaas Bult [11].

This is an elegant way of deriving the specifications of the opamp, but there is one thing worth mentioning here. We do not know how accurate we need to approximate the ideal wavelet. This means that the specifications obtained by simulation are indicative. The exact specifications are difficult to derive since there is no way of knowing how good the impulse response needs to be. Sind's there is no way of knowing what the acceptable error is; we work with the specifications obtained here. That is, 40dB DC gain and GBW is 1 kHz.

The opamp used in the filter is a fully differential folded cascode opamp, shown in figure 6.6.

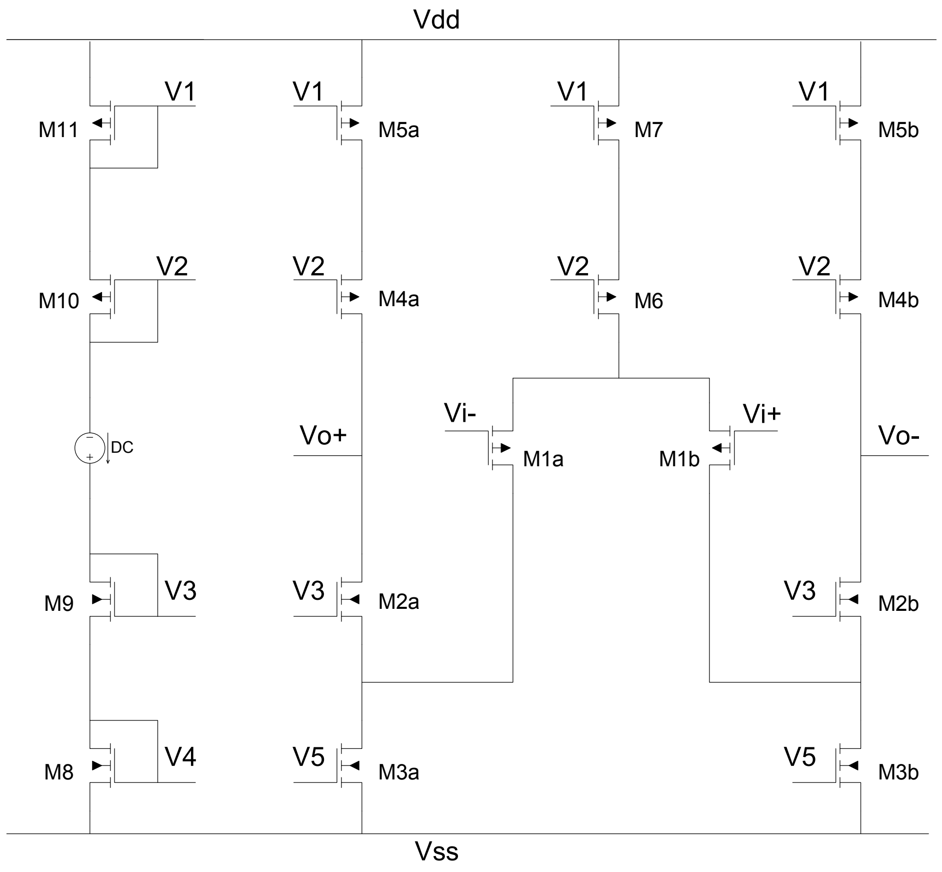


Figure 6.6. The folded cascode opamp.

The left most branch in figure 6.6 is the biasing circuit the remaining part of the figure is the opamp itself.

The folded cascode architecture is chosen because of the DC Voltage gain and output swing. A simple differential pair doesn't provide enough gain. The DC gain of a differential pair (M1a, M1b) is gm*ro. This can be increased by adding a cascode stage. This cascode stage increases the DC gain by increasing the output resistance. Look for example at the differential half circuit of the opamp shown in figure 6.7.

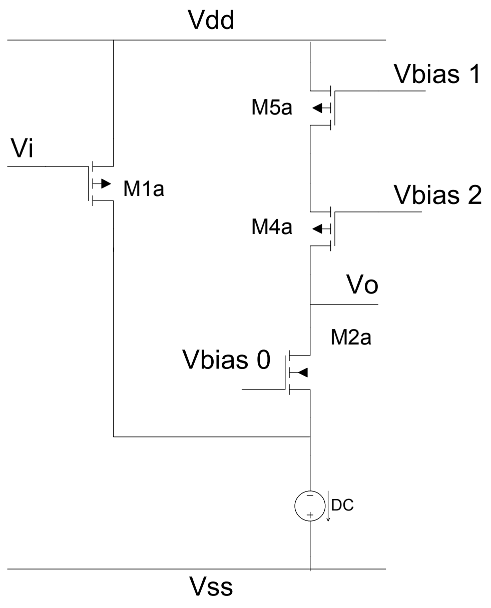


Figure 6.7. The differential half circuit.

The gm of this circuit is almost the same as the gm of transistor M1a so gm = gm1. The output resistance can be calculated as follows. Transistor M1a is a common source stage and has an output impedance of ro1. Transistor M2a is a common gate stage. The output impedance depends on its source and load. The output impedance is given by.

$$\begin{aligned} &ro = (ro4 + ro5)//[ro2 + ro1(1 + gm2)\cdot ro2] \\ &\text{now if we take } (ro4 + ro5) \rightarrow \infty \\ &ro = ro2 + ro1 + gm2\cdot ro1\cdot ro2 \\ &ro \approx gm2\cdot ro1\cdot ro2 \end{aligned} \tag{6.12}$$

Now the voltage gain can be calculated as we know the gm and the ro of the opamp.

$$Av = gm\cdot ro = gm1\cdot gm2\cdot ro1\cdot ro2 \tag{6.13}$$

This is more gain than a normal common source stage

The opamp is fully differential because the entire filter is built up differentially. An alternative for a fully differential opamp is to use a 2 times single ended opamp, but this would consume more power. The drawback of using a fully differential opamp is that it requires a common mode feedback circuit [9] to set the output common mode of the opamp properly.

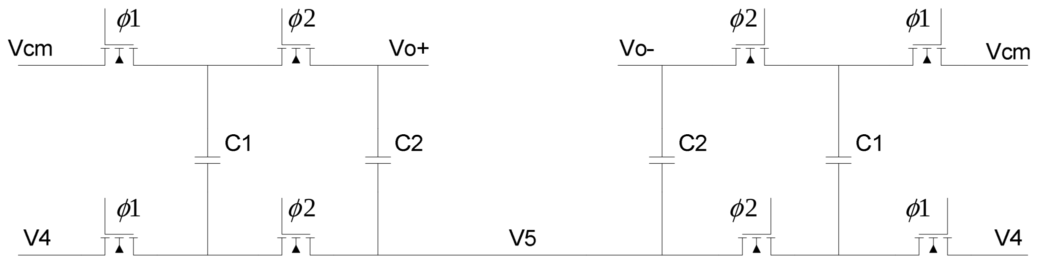


Figure 6.8 The common mode feedback circuit.

The common mode feedback circuit used is shown in figure 6.8. It is also a switched capacitor circuit because it consumes the least power with respect to a continuous time version.

In general a common mode feedback circuit consists of a common mode sense circuit to sense the output common mode level. This common mode level is then compared to a reference level. Then the result of this comparison plus a certain bias level is then fed back to a common mode control node in the circuit to set the output common mode voltage.

The way the common mode feedback circuit works is as follows. Consider the circuit in fig 6.9 a.

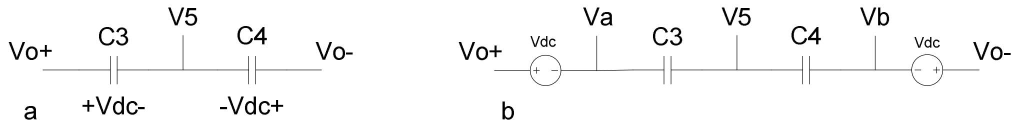


Figure 6.9.

The voltages Vo+ and Vo- are level shifted by Vdc and averaged by capacitors C3 and C4. This results in the bias voltage at node V5. The Vdc in figure 6.9 a (consisting of V4 + Vcm) can also be represented by a series connected voltage source as shown in figure 6.9 b.

By equating the current through the capacitors we can derive the following formula.

$$V5 = \frac{C3 \cdot Va + C4 \cdot Vb}{C3 + C4} \qquad 6.14$$

In equation 6.14 Va = (Vo+) - Vdc and Vb = (Vo-) - Vdc. Substituting this into equation 6.14 we get.

$$V5 = \frac{C3 \cdot \left((Vo+) - Vdc\right) + C4 \cdot \left((Vo-) - Vdc\right)}{C3 + C4} \qquad 6.15$$

After rearranging this equation we arrive at.

$$V5 = \frac{C3 \cdot (Vo+) + C4 \cdot (Vo-)}{C3 + C4} - Vdc \qquad 6.16$$

Now if Capacitor C3 is the same as C4 we get.

$$V5 = \frac{(Vo+) + (Vo-)}{2} - Vdc = Vcm - Vdc \qquad 6.17$$

From equation 6.17 we can conclude that the series connected capacitors can be used as a common mode feed back circuit. Another way of looking at the common mode feedback circuit is shown in figure 6.10.

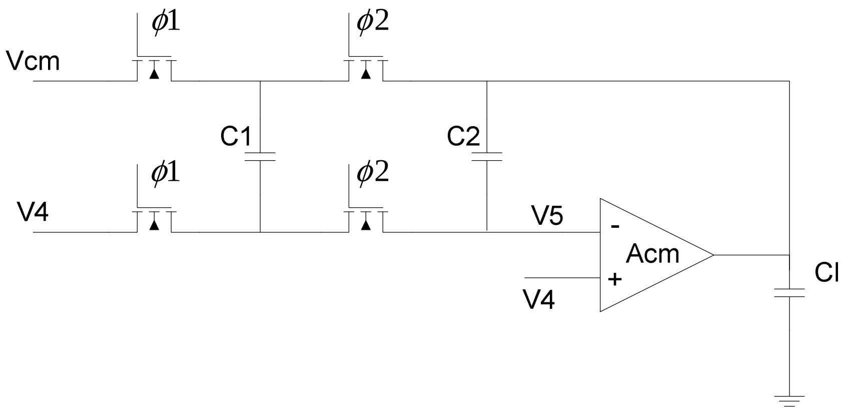


Figure 6.10. A switched capacitor common mode half circuit.

The opamp in this figure is the actual folded cascode opamp that is used for the integrator but now we are interested in its common mode gain. Capacitor C1 is charged to Vcm – V4 during clock phase 1. During clock phase 2 capacitors C1 and C2 are connected in parallel. The charge is redistributed and the opamp will steer its output to such a level that the voltage of node V5 is the same as the voltage of node V4. This occurs if the output of the opamp is Vcm.

# 7 Simulations

Here in this chapter the simulations on the circuit are explained. The circuit is implemented in the TSMC 0.13 process. The simulations have been performed on system and circuit level. For the system level simulations Matlab has been used and for circuit level simulations Cadence Spectre RF proved a handy tool.

## 7.1 System simulation

The system level simulation has been carried out with Matlab Simulink. After having done the approximation and optimization we have arrived at the state-space description we want to implement. This state-space description is built using the standard components in Simulink and an impulse is applied to the input to check its functionality.

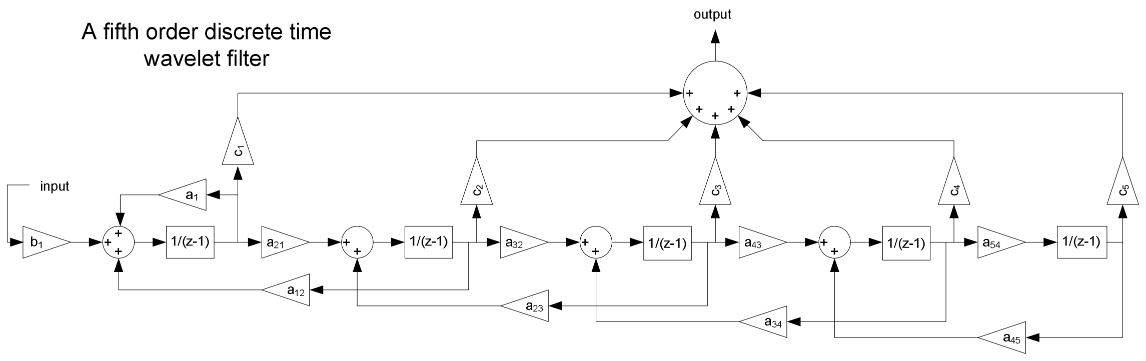


Figure 7.1. A fifth order state-space description.

In figure 7.1 one can see the block schematic as it was simulated in Simulink. In this figure all the triangles labeled a, b and c are the entries from the A, B and C matrices respectively. The functionality of this block schematic is verified by applying an impulse at the input. At the output the impulse response is observed, which is the approximated Gaus1. In figure 7.2 the impulse response of the block schematic is shown in overlap with the ideal Gaus1.

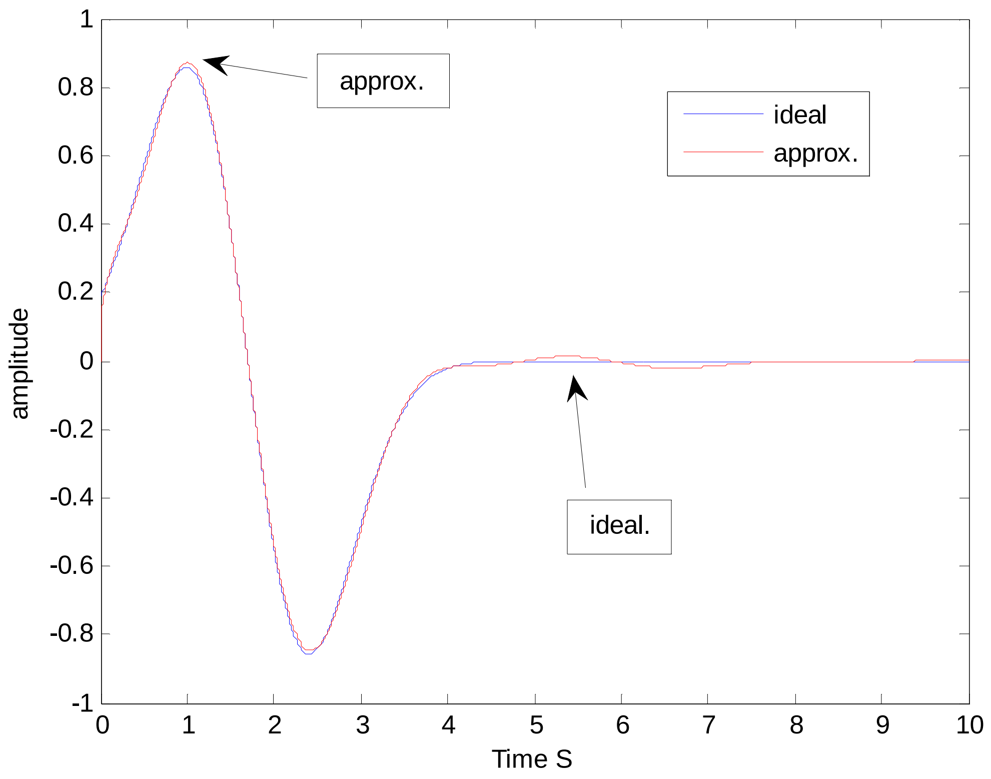


Figure 7.2. The impulse response

The response in figure 7.2 looks rather good. This is because the components in figure 7.1 are all ideal. All these blocks will be implemented by real components. These real components will have imperfections which will degrade the performance of the circuit. The degradation of performance is shown in the next section.

## 7.2 Circuit simulation

The circuit simulations have been carried out in Cadence. First the wavelet filter is built up with ideal components (ideal switches, ideal opamp and ideal caps). This ideal filter is then simulated using a transient simulation. After a successful simulation with ideal components, one by one the ideal components are replaced by real components which will degrade the performance of the circuit. After replacing an ideal component for a real component the functionality of the circuit is checked again by means of a transient simulation. In this way the entire circuit is slowly turned into a real circuit and in each intermediate step the functionality is verified. The impulse response of the ideal circuit is shown in figure 7.3.

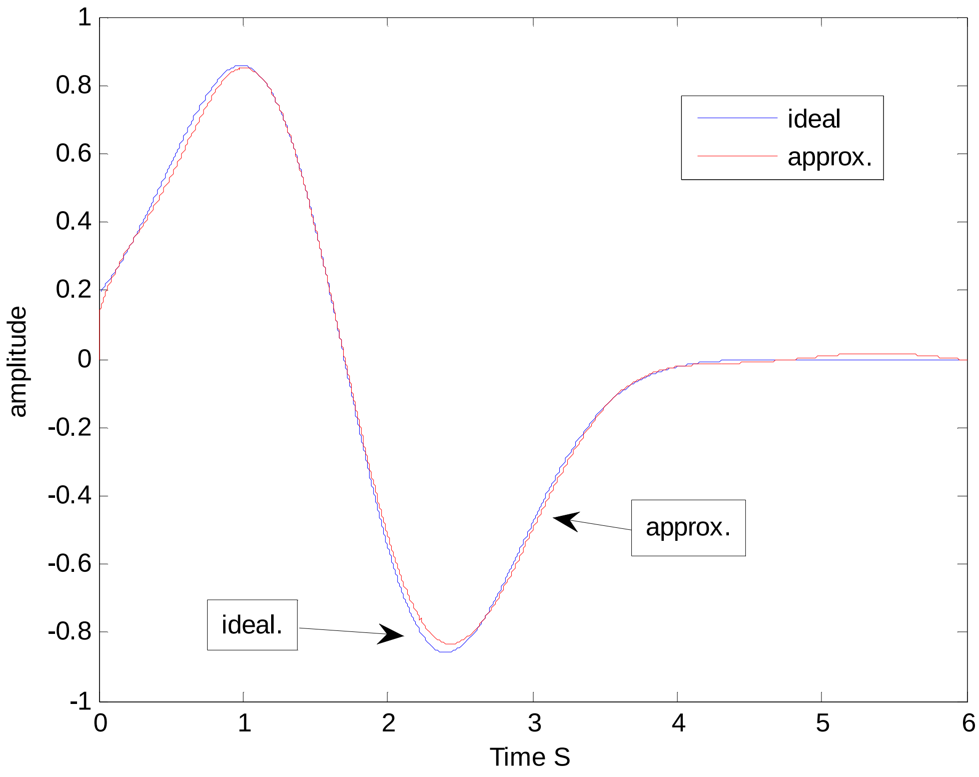
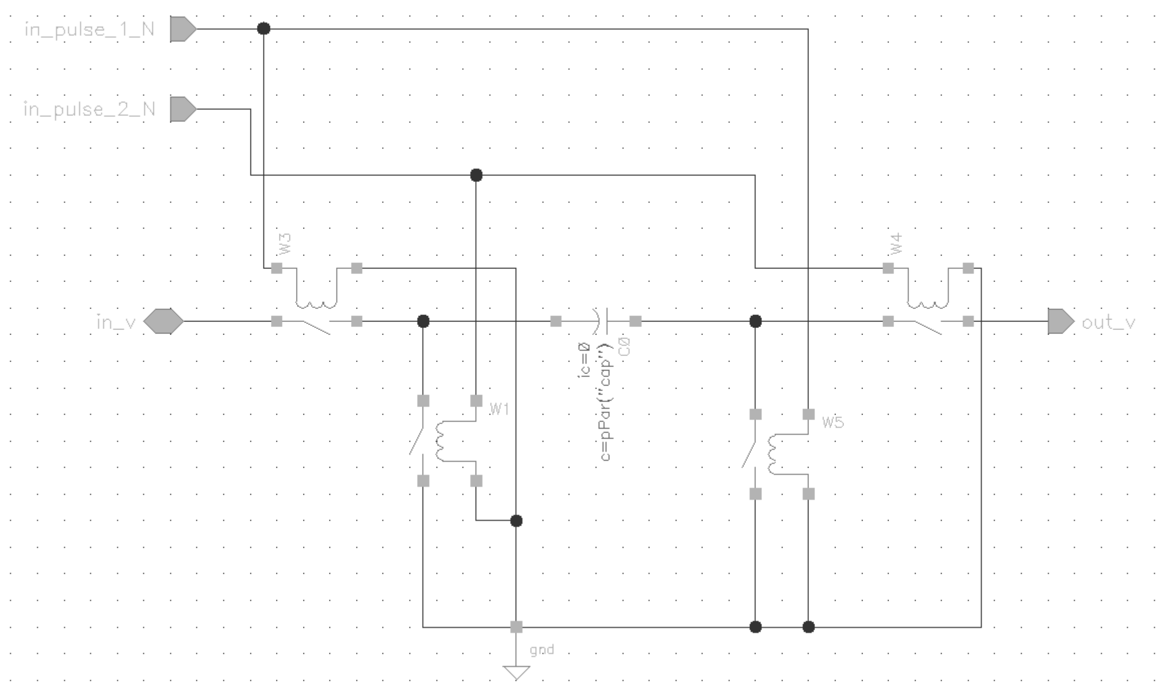
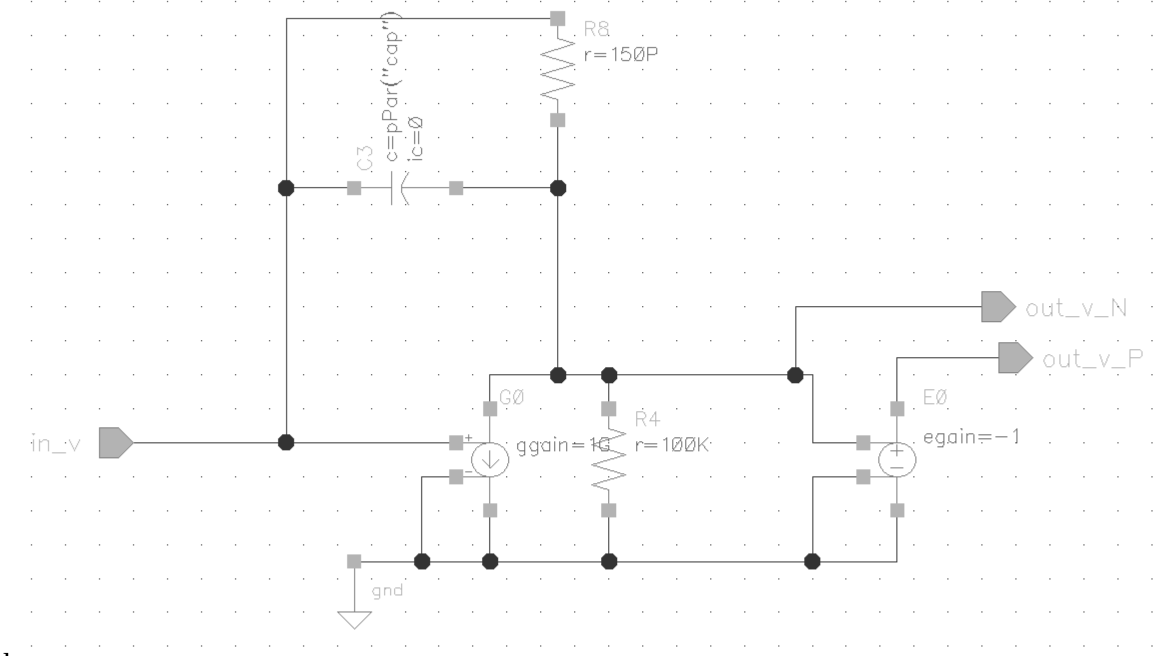


Figure 7.3. The impulse response of the ideal circuit.

To give the reader an impression of how the ideal circuit is build up we give a schematic of one integrator section of the circuit. This schematic is shown in figure 7.4. Here the ideal opamp is just a voltage controlled current source. The switches are, as you can see, ideal switches.



a



b

Figure 7.4. a: A switched capacitor section of the ideal circuit. b: An opamp section of the ideal circuit.

The first step taken to turn this ideal circuit into a real circuit is to change the ideal switches for real MOS switches. We started using minimum size N-MOS switches. This is because they cause the least charge injection and the least clock feed through. From the simulation this proved insufficient. The signal swing at the input of the switched capacitor was such that a transmission gate or CMOS switch is needed. The charge injected by the N-MOS switch connected to the opamp can be annihilated by a dummy switch in a subsequent clock phase. The switch configuration can be seen in figure 6.5.

The opamp is a fully differential folded cascode opamp as it is shown in figure 6.6. On this circuit we have performed an ac simulation. This is because in this case we are interested in the frequency performance of the circuit. The result of the ac simulation is shown in figure 7.5. The markers in the plot indicate the DC gain and the unity gain frequency.

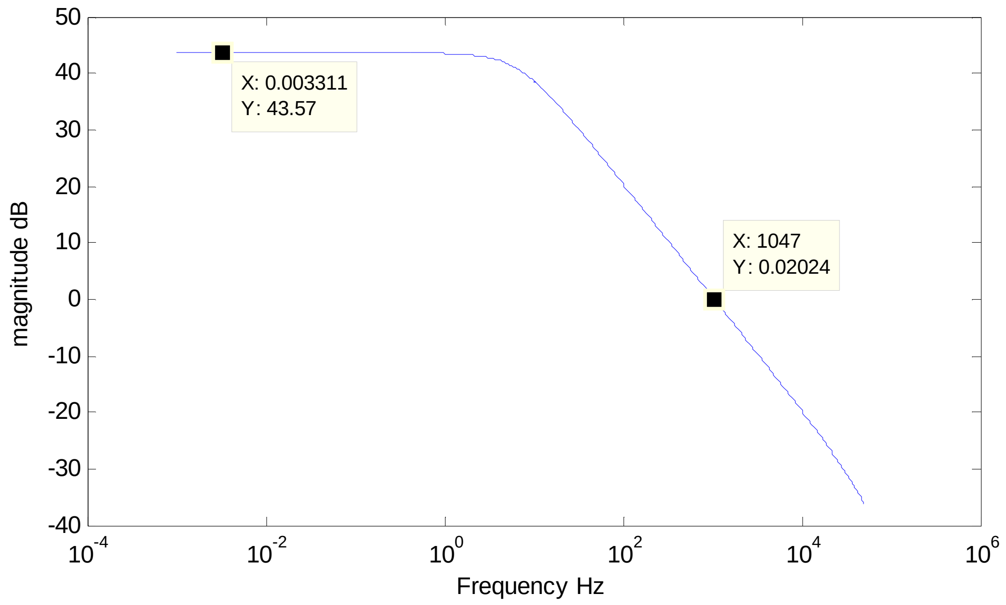


Figure 7.5. The ac performance of the opamp.

The opamp discussed above needs a common mode feedback circuit as described in the previous chapter. The functionality of this circuit is verified by means of a transient analysis. Again we started by building with ideal components as shown in figure 7.6. We only build a differential half circuit to check the functionality. This is sufficient since the full differential circuit can be divided into two half circuits which are equal. Simulating only a half circuit is sufficient to verify its functionality.

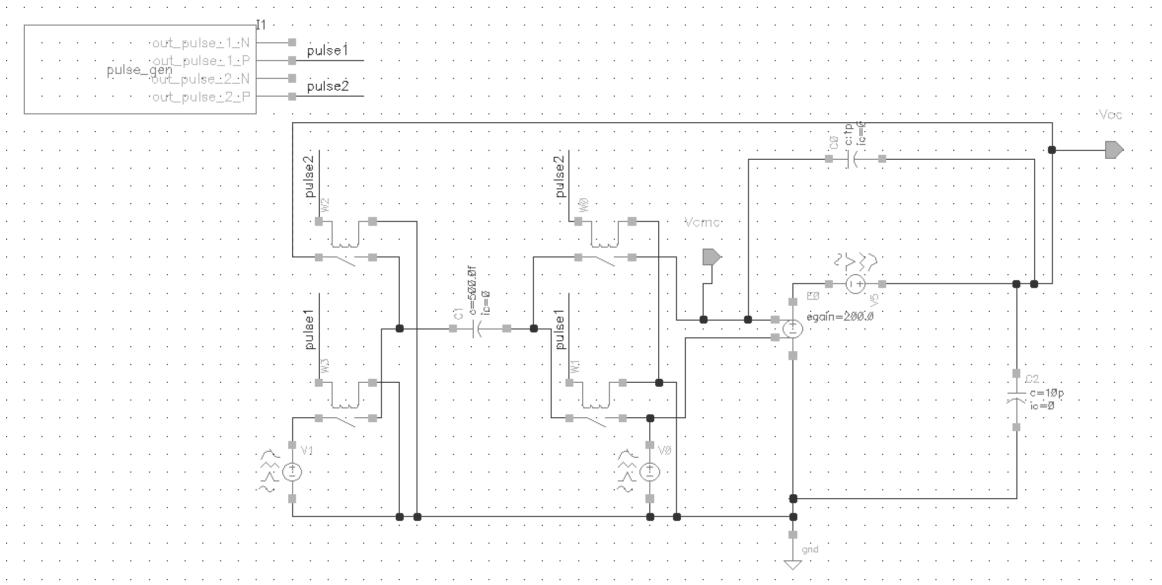


Figure 7.6. A differential half circuit of the common mode feedback circuit.

The common mode value for our circuit is set to be 600mV. This exactly half of the power supply of the circuit. We have chosen this value because it yields the maximum voltage swing at the output of the integrator. This value of half the power supply is a consequence of the dynamic range optimization we have done. There we said that the dynamic range is optimum if all the integrators have the same signal swing at the output. The transient response of the common mode half circuit is shown in figure 7.7.

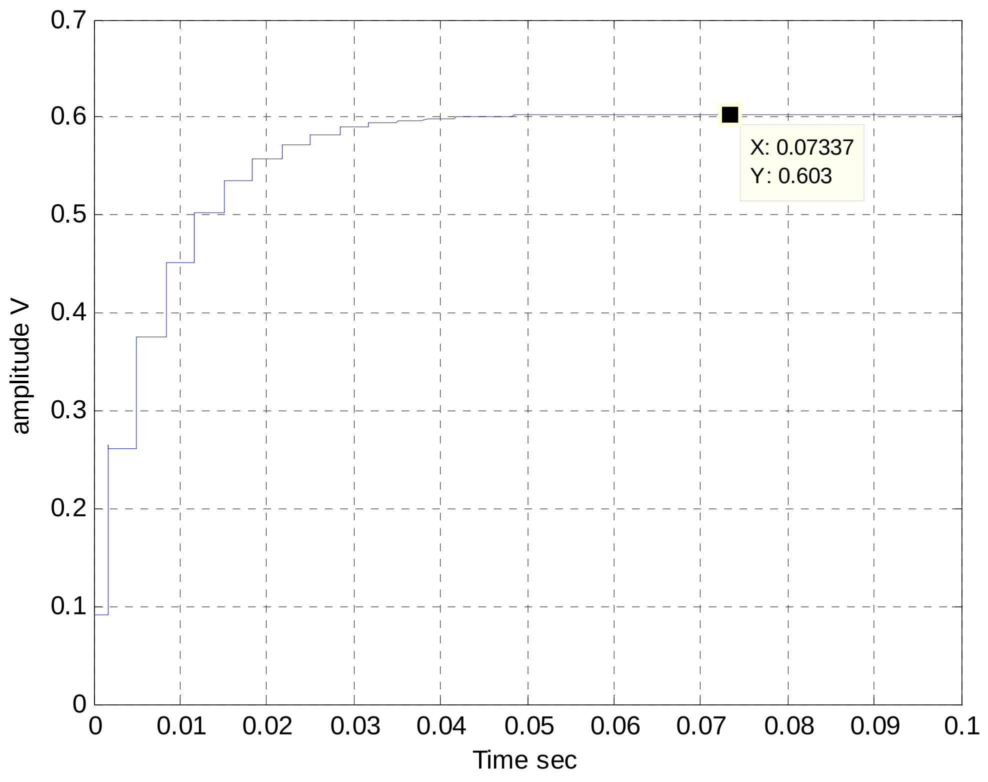


Figure 7.7. Transient response of the common mode half circuit.

For this circuit only the ideal switches have to be implemented by real switches. The ideal opamp shown in figure 7.6 will in reality be the folded cascode opamp. The only thing is that for the common mode feedback circuit we will use the common mode gain of this opamp instead of the differential mode gain.

Now that all ideal components have been replaced by real components we have arrived at an implementable circuit. The whole circuit is simulated by means of a transient analysis. The result is shown in figures 7.8 and 7.9.

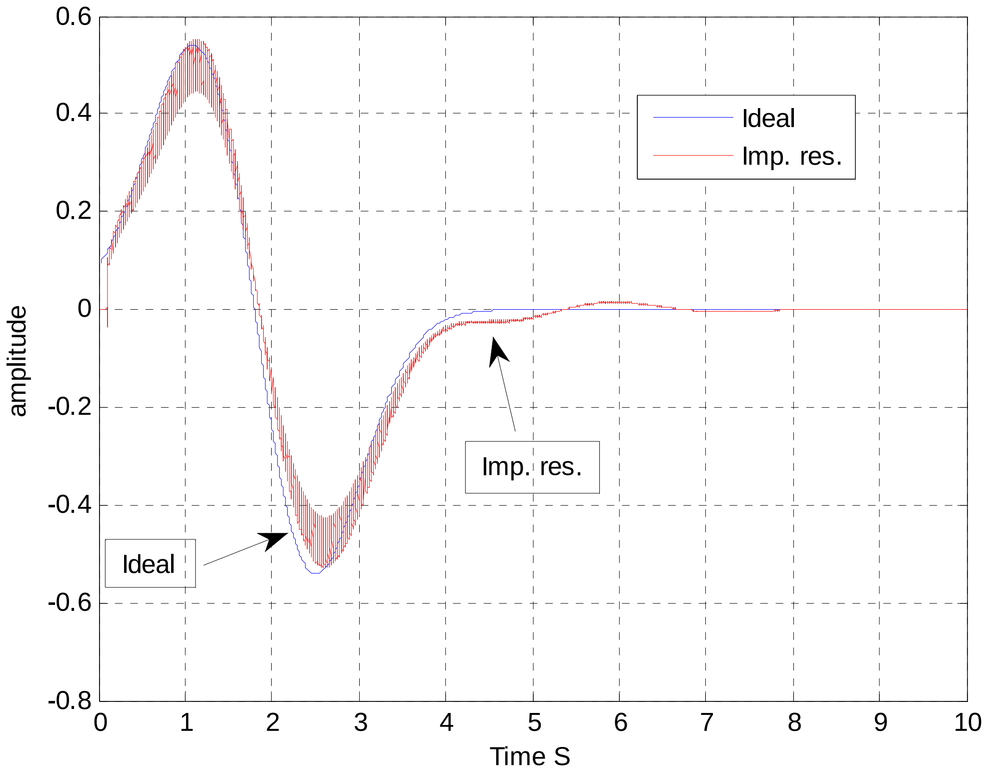


Figure 7.8. The impulse response of the filter + the ideal gaus1.

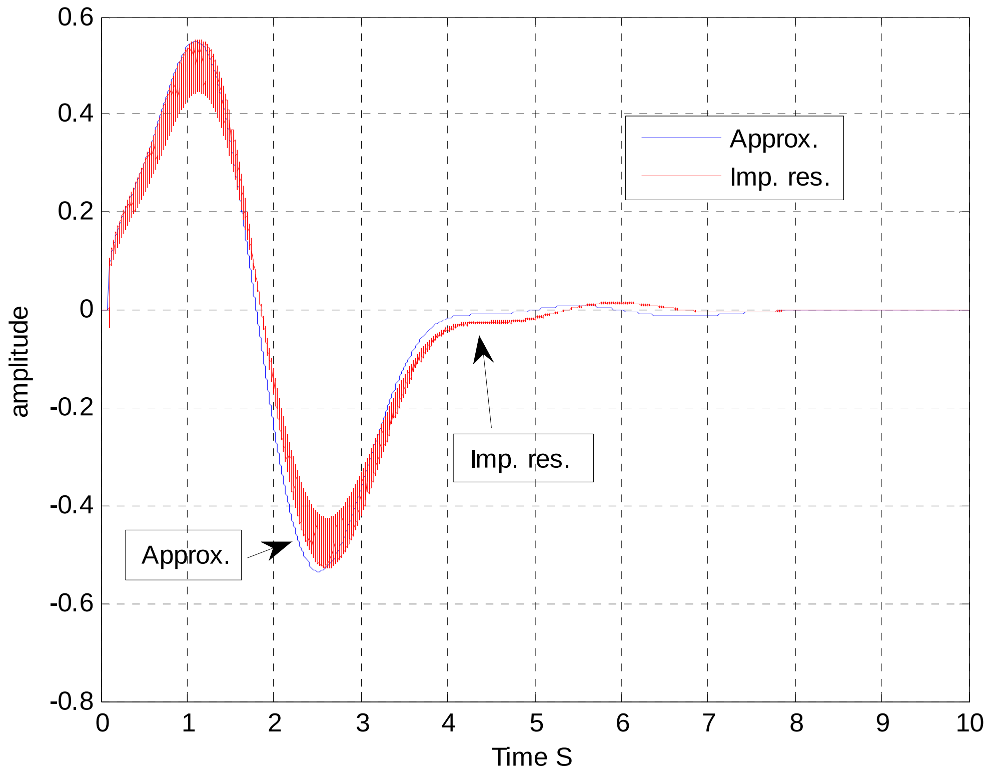


Figure 7.9. The impulse response of the filter + the approximation.

The whole circuit has also been simulated by means of a pac simulation in order to obtain the bode plot of the circuit. The result is shown in figures 7.10 and 7.11.

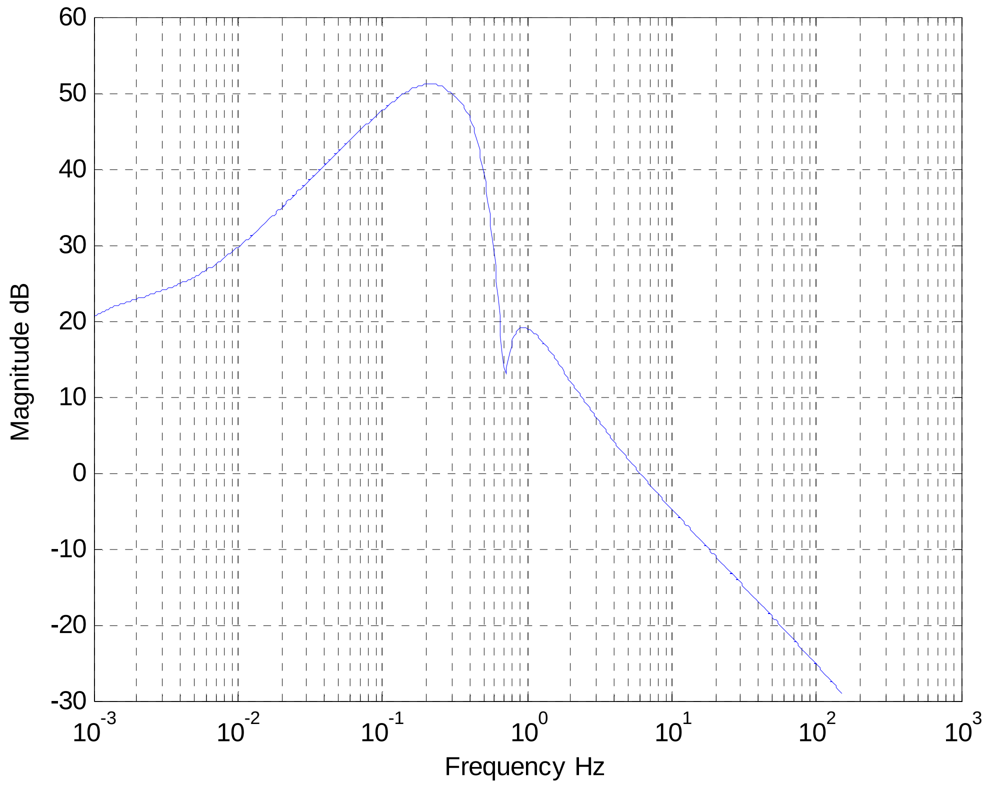


Figure 7.10. The magnitude response of the circuit.

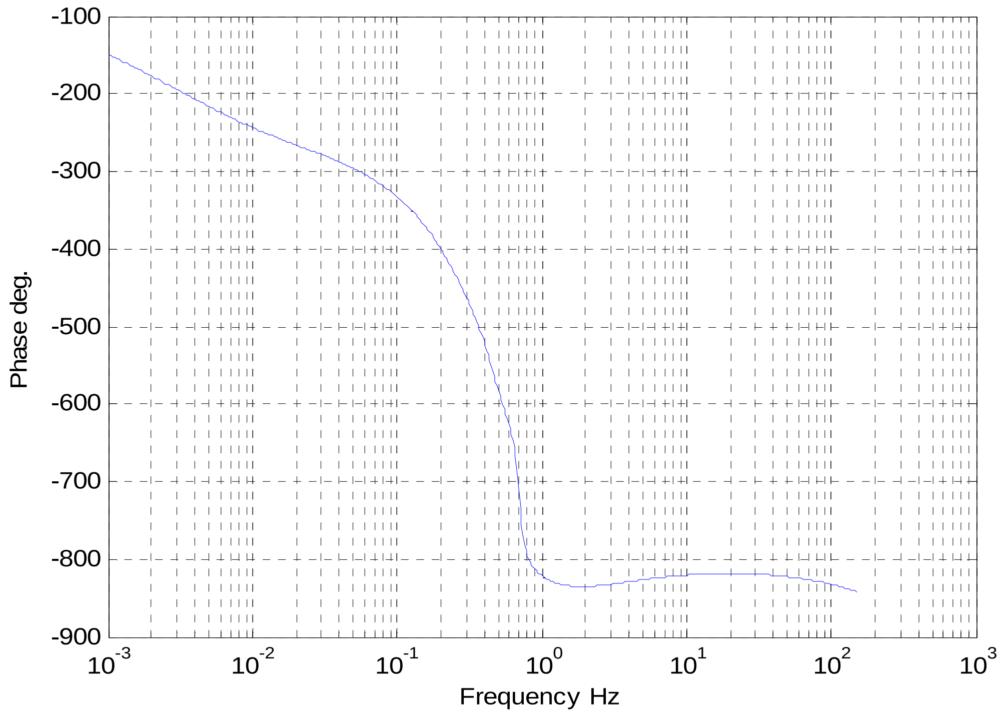


Figure 7.11. The phase response of the circuit.

# 8 Conclusions

The Pade method offers a straight forward way of approximating the wavelet. It has however a few shortcomings. The first is that it does not always yield a stable solution, especially for higher orders of approximation. The second is that it shows some ringing at the beginning of the approximation which is undesirable.

The SVD approximation performs much better in this sense. It always yields a stable solution, even for high orders of approximation. It doesn't show the ringing as in the Pade approximation. And in terms of mean square error it outperforms the Pade method. The last benefit of SVD over Pade, which is worth mentioning here, is that the SVD method in the end yields matrix entries which are closer to each other.

The optimization method which is based on the work of Groenewold [12] and described by Haddad [1] and described in more detail by Rocha [10] provides a considerable improvement in the circuit's dynamic range. However, a drawback of this procedure is that it typically yields state-space descriptions which are fully dense. This drawback is overcome by the Schwarz decomposition. We gave in al little on the dynamic range but we obtained a sparse state-space description which is easy to implement.

The work of this thesis has led to a Matlab program in which all the steps, approximation, optimization and decomposition, have been incorporated. Together with the requirements (wavelet, order of approximation and sampling frequency) of the user this program will derive the optimal, sparse state-space description ready for implementation.

In this thesis the wavelet filter was implemented in a fully differential switched capacitor technology. The reason for choosing this technology is that it provides the maximum dynamic range, namely the voltage swing at the output of the integrator spans the entire supply voltage and therefore it has the highest dynamic range. The differential structure made it easy to implement the negative entries of the state-space description.

The contributions I have made are the application of the SVD approximation and the application of the Schwarz form. Furthermore it is the first time that a wavelet filter is constructed in a fully differential switched capacitor circuit with common mode feedback. Also the program which has been written provides an easy design tool for wavelet filter design. The work done for this thesis has led to two publications. One ISCAS paper and an abstract and poster. Both can be found in the appendix.

## 9 Appendix

## A The Matlab program

```
%----------------------------------------------------------------------
-
%This program requires the user to define a symbolic expression.
%This program will first perform a Pade approximation on that
expression.
%The order of this approximation is defined by the user. The Pade
%approximation always yields a transfer function in the S domain.
Later,
%if needed, it can be converted into the Domain by the impinvar method
or
%the bilinear method.
%Then the program will convert this Pade approximation into a state-
%space description of the form chosen by the user.
%The output of this program is the variable SSN, which is a state-
%space description.
%----------------------------------------------------------------------
-

disp('Enter the symbolic expression you want to approximate.');
disp('Default value is -2*(t-1.7)*exp(-(t-1.7)^2)');
user_entry1 = input('Give the symbolic expression:');
if size(user_entry1) == 0
    syms t;
    user_entry1 = sym('-2*(t-1.7)*exp(-(t-1.7)^2)');    %default value
end
disp(' ');

user_entry2 = input('Give the order of the numerator (default=2):');
if size(user_entry2) == 0
    user_entry2 = 2;    %default value
end

user_entry3 = input('Give the order of the denominator (default=4):');
if size(user_entry3) == 0
    user_entry3 = 4;    %default value
end
disp(' ');
sys_a = myapprox(user_entry1, user_entry2, user_entry3)    %the PADE
approx.

%Make a choice between the S or the Z domain (S is default)
disp('The program now requires you to choose between the S or');
disp('the Z domain.');
disp('The options to choose from are "S" for the S domain and "Z"
for');
disp('the Z domain (default=S)');

user_entry4 = input('Give the domain:','s');
if isempty(user_entry4)
    user_entry4 = 'S';    %default value
```

```
end
disp(' ');

if user_entry4 == 'S'   %S domain
    disp('The approximation in the S domain is:');
    sys_a    %display the transfer function
    disp(' ');
    disp('The program now requires you to choose the specific');
    disp('state space description.');
    disp('The options to choose from are "O" for the orthonormal ');
    disp('form and "S" for the schwarz form (default=O)');

    user_entry5 = input('Give the specific State Space 
desription:','s');
    if isempty(user_entry5)
        user_entry5 = 'O';    %default value
    end
    disp(' ');

    if user_entry5 == 'O'   %orthonormal
        SS = myorth_ladder(sys_a);
    end
    if user_entry5 == 'S'   %schwarz
        temp = myss(sys_a);   %create canonical SS
        opt = myoptimal(temp);  %optimize for dynamic range
        SS = schwarzform(opt);    %schwarzform
    end
end

if user_entry4 == 'Z'  %Z domain
    disp('The program will have to transfer a function from the S 
domain');
    disp('to the Z domain. This can be done by the Impulse Invariant 
method');
    disp('or the Bilinear method.');
    disp('Choose "I" for the Impulse Invariant method or "B" for the 
Bilinear');
    disp('method. (default=I)');
    user_entry9 = input('Give the transfer method:','s');
    if isempty(user_entry9)
        user_entry9 = 'I';  %default value
    end
    disp(' ');

    disp('Also, the program needs a sampling frequency to make a 
conversion');
    disp('from S domain to Z domain.');
    user_entry10 = input('Give the sampling frequency 
(default=unspecified):');
    if size(user_entry10) == 0
        user_entry10 = -1;  %default value
    end
    disp(' ');

    if user_entry9 == 'I'   %impinvar method
        [num_s den_s] = tfdata(sys_a,'v');
        [num_z, den_z] = impinvar(num_s, den_s, user_entry10);
        disp('The approximation in the Z domain is:');
        sys = tf(user_entry10*num_z, den_z, (1/user_entry10))
    end
    if user_entry9 == 'B'   %bilinear method
        [num_s den_s] = tfdata(sys_a,'v');
        [num_z,den_z] = bilinear(num_s, den_s, user_entry10);
        disp('The approximation in the Z domain is:');
        sys = tf(num_z, den_z, (1/user_entry10))
    end

    disp(' ');
    disp('The program now requires you to chïose the type of active
element');
    disp('for your implemuntation.');
    disp('The options to choose from are "D" for 1/z or "I" for 1¯)z-
1)');
    disp('(default=I)');

    user_entry7 = input('Enter the type of active element:','s');
    if isempty(user_entry7)
        user_entry7 = 'I';  %default value
    end
    disp(' ');

    if user_entry7 == 'D'   %the ictive element is a delay
        disp('The program now requires you to choose the specific');
        disp('state space description.');
        disp('The options to choose from are "O" for the orthonormal
');
        disp('form and "S" for the schwarz form (default=O)');

        user_entry6 = input('Give the specific State Space
desription:','s');
        if isempty(user_entry6)
            user_entry6 = 'O';    %default value
        end
        disp(' ');

        if user_entry6 == 'O'   %orthonormal
            SS = myorth_ladder_z(sys);
        end
        if user_entry6 == 'S'   %schwarz
            temp = myss_z(sys);   %create canonical SS
            opt = myoptimal_z(temp);  %optimize for dynamic range
            SS = schwarzform_z(opt);   %schwarz form
        end
    end
    if user_entry7 == 'I'   %the active element is an integrator
        disp('The program now requires you to choose the specific');
        disp('state space description.');
        disp('The options to choose from are "O" for the orthonormal
');
        disp('form and "S" for the schwarz form (default=O)');
        user_entry8 = input('Give the specific State Space
desription:','s');
        if isempty(user_entry8)
            user_entry8 = 'O';    %default value
        end
        disp(' ');

        if user_entry8 == 'O'   %orthonormal
            orth1 = myorth_ladder_z(sys);
            [a b c d ts] = ssdata(orth1);
            a = a-eye(length(a));   %create the integrator
            SS = ss(a,b,c,d,ts);
        end
        if user_entry8 == 'S'   %schwarz
            temp1 = myss_z(sys);   %create canonical SS
            [a b c d ts] = ssdata(temp1);
            a = a-eye(length(a));   %create the integrator
            temp = ss(a,b,c,d,ts);
            opt = myoptimal_z(temp);  %optimize for dynamic range
            SS = schwarzform_z(opt);   %schwarz form
        end
    end
end

% The program will now determine if the State Space description (SS)
% has complex coefficients. These coefficients will be turned into
% real coefficients.

[a b c d ts] = ssdata(SS);
N = length(b);
for t = 1:N
    if imag(b(t)) ~= 0
        b(t) = b(t)*i;
    end
    if imag(c(t)) ~= 0
        c(t) = c(t)/i;
    end
end

% noise scaling

[Ks,Ws] = grams(SS);
sumAr = sum(abs(a'));
sumAc = sum(abs(a));
alfaWsr = sumAr.*(diag(Ws).*diag(Ks))';
alfaWsc = sumAc.*(diag(Ws).*diag(Ks))';
Copt = sqrt(alfaWsr)./sum(sqrt(abs(alfaWsc)));

SSN = ss(a,b,c,d,ts)

return
```

---

```
%-----------------------------------------------------------------------
----
```

```
%this function generates the Pade approximation of a symbolic
expression.
%t is a function in the form of a symbolic expression.
%num is the numerator of the Pade approximation.
%den is the denominator of the Pade approximation.
%this function returnes a transferfunction (tf) with name S.
%-----------------------------------------------------------------------
----

function S = myapprox(t,num,den)

format long;
syms s;
f = laplace(t);
L = num;      %Numerator order
M = den;      %Denominator order
N = L+M+1;       %Taylor approx. order
A = taylor(f,N,0);     % A(x)
a = sym2poly(A)';   %Convert to a vector
a(1:1:N) = a(N:-1:1);   %Reverse the order
C = zeros(N,M+1);   %Create a matrix of dimension N*M+1
temp = zeros(1,N);    %Create a vector of length N
temp(1:1:N) = a(1:1:N); %Copy a in temp

for n = (1:1:M+1)   %The main matrix C
    C(n:1:end,n) = temp(1:1:N+1-n);
end

Ct = C(L+2:1:end,1:1:end);  %Sub matrix
Ct = rref(Ct);   %Reduced row echelon form
n = norm(Ct);    %The norm of Ct
Q = null(Ct);

if Q(1,1)<0
    Q = null(Ct)*(-n);  %Find the Q coefficients
else
    Q = null(Ct)*n;    %Find the Q coefficients
end

Q;
P = C*Q;    %Find the P coefficients
P = P(1:1:L+1); %Only the first L+1 coeff.
P(1:1:end) = P(end:-1:1);   %Reverse the order
Q(1:1:end) = Q(end:-1:1);   %Reverse the order
S = tf(P',Q');

format short
```

---

```
%-----------------------------------------------------------------------
----
%H = myorth_ladder(TF) returns the state-space description (H)
representing
%an orthonormal filter. TF is the transfer function containing the
```

```
%coeffecients in the numerator and denominator, starting with the
highest
%order coefficient.
%-----------------------------------------------------------------------
----

function H = myorth_ladder(sys_tf)

[num den] = tfdata(sys_tf,'v');
D=num(1);
%fill up with zeros in num
m = length(num);
n = length(den);
if (m~=n)
    num1 = num;
    for i=1:1:(n-m)
        num(i) = 0;
    end
    for i=(n-m)+1:1:n
        num(i) = num1(i-(n-m));
    end
end

%Check whether partial fraction expansion is needed
if num(1)~=0
    num=num-num(1)*den;
end;

N=length(den);
%separate odd and even denominator coefficients
if mod(N,2)==0  %returns X - n.*Y where n = floor(X./Y)
    for i=1:ceil(N/2)
        first(i)=den(2*i-1);
        second(i)=den(2*i);
    end;
else
    first(1)=den(1);
    for i=2:ceil(N/2)
        first(i)=den(2*i-1);
        second(i-1)=den(2*i-2);
    end;
end;



%loop for getting reactances
for i=1:(N-1)
    r(N-i)=first(1)/second(1);
    first=first(2:end)-r(N-
i)*[second(2:end),zeros(1,(length(first(2:end))-
length(second(2:end))))];
    temp=first;first=second;second=temp;
end;

%convert reactances in orthonormal coefficients
alpha(N-1)=1/r(N-1);
alpha(1:end-1)=1./sqrt((r(1:end-1).*r(2:end)));

%calculate F
beta1=sqrt(r(1)/pi);
F(1,N-1)=beta1*den(N);
F(2,N-2)=F(1,N-1)/alpha(1);
for i=3:N-1
    F(i,:)=([F(i-1,2:end),0]+alpha(i-2)*F(i-2,:))/alpha(i-1);
end;

%create state space system
A=diag(alpha(1:end-1),1)+diag(-alpha(1:end-1),-1);
A(N-1,N-1)=-alpha(N-1);
B(N-1)=sqrt(alpha(N-1)/pi);
B=B.';
C(N-1)=num(2)/F(N-1,1);
C(N-2)=num(3)/F(N-2,2);
for i=3:N-1
    C(N-i)=(num(i+1)-C*F(:,i))/F(N-i,i);
end;

H=ss(A,B,C,D);
```

---

```
%----------------------------------------------------------------------
----
%H = myorth_ladder(TF) returns the state-space description representing
%an orthonormal filter. TF is the transfer function containing the
%coeffecients in the numerator and denominator, starting with the
highest
%order coefficient.
%----------------------------------------------------------------------
----

function H = myorth_ladder_z(sys_tf)

[num den Ts] = tfdata(sys_tf,'v');
D=num(1);
%fill up with zeros in num
m = length(num);
n = length(den);
if (m~=n)
    num1 = num;
    for i=1:1:(n-m)
        num(i) = 0;
    end
    for i=(n-m)+1:1:n
        num(i) = num1(i-(n-m));
    end
end

%Check whether partial fraction expansion is needed
if num(1)~=0
    num=num-num(1)*den;
end;
```

```
N=length(den);
%separate odd and even denominator coefficients
if mod(N,2)==0  %returns X - n.*Y where n = floor(X./Y)
    for i=1:ceil(N/2)
        first(i)=den(2*i-1);
        second(i)=den(2*i);
    end;
else
    first(1)=den(1);
    for i=2:ceil(N/2)
        first(i)=den(2*i-1);
        second(i-1)=den(2*i-2);
    end;
end;


%loop for getting reactances
for i=1:(N-1)
    r(N-i)=first(1)/second(1);
    first=first(2:end)-r(N-
i)*[second(2:end),zeros(1,(length(first(2:end))-
length(second(2:end))))];
    temp=first;first=second;second=temp;
end;

%convert reactances in orthonormal coefficients
alpha(N-1)=1/r(N-1);
alpha(1:end-1)=1./sqrt((r(1:end-1).*r(2:end)));

%calculate F
beta1=sqrt(r(1)/pi);
F(1,N-1)=beta1*den(N);
F(2,N-2)=F(1,N-1)/alpha(1);
for i=3:N-1
    F(i,:)=([F(i-1,2:end),0]+alpha(i-2)*F(i-2,:))/alpha(i-1);
end;

%create state space system
A=diag(alpha(1:end-1),1)+diag(-alpha(1:end-1),-1);
A(N-1,N-1)=-alpha(N-1);
B(N-1)=sqrt(alpha(N-1)/pi);
B=B.';
C(N-1)=num(2)/F(N-1,1);
C(N-2)=num(3)/F(N-2,2);
for i=3:N-1
    C(N-i)=(num(i+1)-C*F(:,i))/F(N-i,i);
end;

H=ss(A,B,C,D,Ts);
```

---

```
%-----------------------------------------------------------------------
----
```

```
%This function creates a canonical SS description of a
transferfunction.
%This function returnes a state space description (ss)with the name
sys_ss.
%-----------------------------------------------------------------------
----

function sys_ss = myss(sys_tf)

sys_ss = canon(sys_tf,'companion');
```

---

```
%-----------------------------------------------------------------------
----
%This function creates a canonical SS description of a
transferfunction.
%This function returnes a state space description (ss).
%-----------------------------------------------------------------------
----

function sys_ss = myss_z(sys_tf)

sys_ss = canon(sys_tf,'companion');
```

---

```
%-----------------------------------------------------------------------
----
% This function optimizes the state space discription.
% The input and output of this function are both state space
discriptions.
%-----------------------------------------------------------------------
----

function F = myoptimal(sys)

% first calculate the controllability and observability grammians

[A1,B1,C1,D1] = ssdata(sys);    %convert ss into single matrixes
[K,W] = grams(sys); %calculate the controllability and observability
grammians

% first optimization step finds a similarity transform, such that the
% controllability grammian of the new system becomes a diagonal matrix
with
% equal diagonal entries.

[P,Ds] = eig(K); %calculate the eigenvectors and eigen values
D = sqrt(Ds);   %square root
T = P*D;
Tt = transpose(T);
Tinv = inv(T);
Tinvt = transpose(Tinv);

K1 = Tinv*K*Tinvt;
```

```
W1 = Tt*W*T;

Anew = Tinv*A1*T;
Bnew = Tinv*B1;
Cnew = C1*T;
Dnew = D1;
sysnew = ss(Anew,Bnew,Cnew,Dnew);

% Second optimization step.

[WMat,WAutov] = eig(W1);    %calculate the eigenvectors and eigen
values
T2 = WMat;
T2t = transpose(T2);
T2inv = inv(T2);
T2invt = transpose(T2inv);

Wfin = T2t*W1*T2;
Kfin = K1;

Afin = T2inv*Anew*T2;
Bfin = T2inv*Bnew;
Cfin = Cnew*T2;
Dfin = D1;

sysfin = ss(Afin,Bfin,Cfin,Dfin);

F = sysfin;
```

---

```
%----------------------------------------------------------------------
----
%This function optimizes the state space discription.
%The input and output of this function are both state space
discriptions.
%----------------------------------------------------------------------
----

function F = myoptimal_z(sys)

% first calculate the controllability and observability grammians

[A1,B1,C1,D1,Ts] = ssdata(sys);    %convert ss into single matrixes
[K,W] = grams(sys); %calculate the controllability and observability
grammians

% first optimization step finds a similarity transform, such that the
% controllability grammian of the new system becomes a diagonal matrix
with
% equal diagonal entries.

[P,Ds] = eig(K); %calculate the eigenvectors and eigen values
D = sqrt(Ds);   %square root
T = P*D;
```

```
Tt = transpose(T);
Tinv = inv(T);
Tinvt = transpose(Tinv);

K1 = Tinv*K*Tinvt;
W1 = Tt*W*T;

Anew = Tinv*A1*T;
Bnew = Tinv*B1;
Cnew = C1*T;
Dnew = D1;
sysnew = ss(Anew,Bnew,Cnew,Dnew);

% Second optimization step.

[WMat,WAutov] = eig(W1);    %calculate the eigenvectors and eigen
values
T2 = WMat;
T2t = transpose(T2);
T2inv = inv(T2);
T2invt = transpose(T2inv);

Wfin = T2t*W1*T2;
Kfin = K1;

Afin = T2inv*Anew*T2;
Bfin = T2inv*Bnew;
Cfin = Cnew*T2;
Dfin = D1;

sysfin = ss(Afin,Bfin,Cfin,Dfin,Ts);

F = sysfin;
```

---

```
%----------------------------------------------------------------------
----
% H = SCHWARZFORM(opt)
%
% Transforms a continuous-time stable state-space system (A,B,C) into
% input normal Schwarz form.
%
% By Ralf Peeters,
% Dept. Mathematics, University Maastricht.
% This version: 19 October 2006.
%----------------------------------------------------------------------
----

function H=schwarzform(opt)

[a b c d] = ssdata(opt);

% P is the controllability Grammian.
p=lyap(a,b*b');
```

```
% T is a square root of P.
[u,s,v]=svd(p);
t=u*(s^.5)*v';
% This applies a state space transformation to achieve input normality.
aa=t\a*t;
bb=t\b;
cc=c*t;
% M is the controllability matrix of the input normal system.
m=ctrb(aa,bb);
% This performs QR-decomposition on M. Then Q can be used to make the
% controllability matrix upper triangular while maintaining input
% normality.
[q,r]=qr(m);
% The signs of the diagonal entries of R are used to make R positive
upper
% triangular.
sig=sign(diag(r));
q=q*diag(sig);
% The orthogonal matrix Q is used to achieve an input normal Schwarz
form.
as=q'*aa*q;
bs=q'*bb;
cs=cc*q;
% Make all the theoretically zero entries in A and B into hard zeros.
[n,m]=size(a);
for i=2:n,
    as(i,i)=0;
    bs(i)=0;
end;
for i=3:n,
    for j=1:i-2,
        as(i,j)=0;
        as(j,i)=0;
    end;
end;
ds = d;
H=ss(as,bs,cs,ds);
return
```

---

```
%-----------------------------------------------------------------------
----
% H = SCHWARZFORM(opt)
%
% Transforms a continuous-time stable state-space system (A,B,C) into
% input normal Schwarz form.
%
% By Ralf Peeters,
% Dept. Mathematics, University Maastricht.
% This version: 19 October 2006.
%-----------------------------------------------------------------------
----

function H=schwarzform_z(opt)

[a b c d Ts] = ssdata(opt);
```

```
% P is the controllability Grammian.
p=lyap(a,b*b');
% T is a square root of P.
[u,s,v]=svd(p);
t=u*(s^.5)*v';
% This applies a state space transformation to achieve input normality.
aa=t\a*t;
bb=t\b;
cc=c*t;
% M is the controllability matrix of the input normal system.
m=ctrb(aa,bb);
% This performs QR-decomposition on M. Then Q can be used to make the
% controllability matrix upper triangular while maintaining input
% normality.
[q,r]=qr(m);
% The signs of the diagonal entries of R are used to make R positive
upper
% triangular.
sig=sign(diag(r));
q=q*diag(sig);
% The orthogonal matrix Q is used to achieve an input normal Schwarz
form.
as=q'*aa*q;
bs=q'*bb;
cs=cc*q;
% Make all the theoretically zero entries in A and B into hard zeros.
[n,m]=size(a);
for i=2:n,
    as(i,i)=0;
    bs(i)=0;
end;
for i=3:n,
    for j=1:i-2,
        as(i,j)=0;
        as(j,i)=0;
    end;
end;
ds = d;
H=ss(as,bs,cs,ds,Ts);
return
```

---

```
% This function calculates the K matrix (Controllability Grammian) and
also
% W matrix (Observability Grammian) for the state space input to the
% function..

function [K,W] = grams(sys)

K = gram(sys,'c');
W = gram(sys,'o');
```

---

## B The SVD approximation

```
%-----------------------------------------------------------------------
----
% This code implements the system identification of a signal. A signal
in
% time domain is used as input to the calculation. From this signal a
% discrete time State Space description will be derived that has this
% signal as its impulse response.
%-----------------------------------------------------------------------
----

ts = 1/300; % Sample time
x = [0:ts:8];   % Sample time vector
F = (-2*(x-1.7).*exp(-(x-1.7).^2)); % Gaus 1
figure(1);
plot(F);

r = zeros(1,length(x)); % Column vector of zero's
r(1,1) = F(1,1);    % First entry of r is first entry of F
T = toeplitz(F,r);  % Toeplitz matrix of F (lower triangular)

H = hankel(T(:,1)); % Hankel matrix of T
[U S V] = svd(H);    % Calculate the SVD
cond(S);
figure(2);
plot(diag(S),'x')
% resize U, S and V according to the approximation.
U = U(:, 1:1:5);
S = S(1:1:5, 1:1:5);
V = V(:, 1:1:5);
H_appr = U*S*V'; % The new approximated hankel matrix.
cond(S);

C = S^(1/2)*V';   % Controllability matrix
O = U*S^(1/2);    % Observability matrix

Ot = O(1:1:end-1, 1:1:end); % The upper part of the observability
Ob = O(2:1:end, 1:1:end); % The lower part of the observability
a = Ot\Ob;  % Calculate the A matrix
b = C(:,1); % Calculate the B matrix
c = O(1,:); % Calculate the C matrix

SS = ss(a,b,c,0,ts);    % Create the State-space system
figure(3)
impulse(SS)

Wc = gram(SS,'c');     % controllability grammian
Wo = gram(SS,'o');     % observability grammian
test = Wc*Wo;
```

## C The paper

# D The abstract and poster

## A STRUCTURED WAVELET FILTER DESIGN METHODOLOGY

**Michiel A. Grashuis, Wouter A. Serdijn**

*Biomedical Electronics Group
Delft University of Technology,
Mekelweg 4, 2628 CD Delft
The Netherlands
e-mail: M.A.Grashuis@student.tudelft.nl; W.A.Serdijn@tudelft.nl
Web page: http://elca.et.tudelft.nl/~wout

### ABSTRACT

A structured design methodology for on-chip, analog wavelet filters for biomedical signal analysis is proposed. In many wearable and implantable devices, the analog-to-digital converter (ADC) takes up a large chunk of the device's power consumption. For this reason, we propose to reduce the overall power consumption by performing analog pre-processing employing a wavelet filter. Despite consuming power itself, the wavelet filter allows for a reduction of the ADC resolution and hence may entail a favorable reduction in power consumption of the entire system.

A wavelet filter is a device which has a (time-reversed) mother wavelet as its impulse response and implements one scale of the corresponding wavelet transform. We present a design flow that can generate a wavelet filter topology from an arbitrary given wavelet base [1]. The design flow is implemented in Matlab code. It requires a function description as input and produces an optimized state-space description and wavelet filter topology.

The program first requires the user to define the wavelet base in the form of a function description (in the time domain). This can be any known wavelet, e.g., a Gauss1 (1st derivative of a Gaussian), Daubechies or Morlet wavelet, or a (multi-) wavelet tailored to the detection of a particular signal morphology [2]. The user of the program then has the option to choose between a continuous-time (CT) and a discrete-time (DT) system. For the DT wavelet filter, the user subsequently can define the sampling frequency (Fs) and the nature of the basic building block employed, e.g. a delay (1/z) element or an integrator (1/(z-1)) element. Based on the proper choice of this element, the system can be consequently implemented by means of a switched current (SI) or switched capacitor (SC) circuit technique. For the CT case, the only active element the designer can select is the integrator (1/s), which can be implemented, e.g., using gm-C, opamp-RC or dynamic-translinear (DTL) circuit techniques. For both the CT and the DT case the user can choose the order of the filter. The output of the program in both cases is a state-space description of which the A matrix has an orthonormal ladder form [3], corresponding with a wavelet filter topology with close-to-optimal dynamic range, minimal power consumption and minimal sensitivity to component variations.

As an example, we used the program to derive a 5th order discrete-time Gauss1 wavelet filter for cardiac signal analysis in pacemakers. The resulting wavelet filter has discrete-time integrators as active elements, runs at a sampling frequency of 300Hz and is implemented using switched capacitor circuit techniques. Its power consumption allows implementation in the sense amplifier of cardiac pacemakers to reliable detect the heart rate, even in presence of strong (50 or 60 Hz) interference.

The design methodology can also be adopted for the design of electronic filters to implement any arbitrary transfer function or impulse response.

## REFERENCES

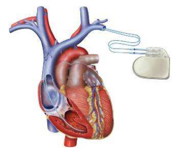
[1] S.A.P. Haddad, N. Verwaal, R. Houben and W.A. Serdijn, “Optimized dynamic translinear implementation of the Gaussian wavelet transform”, proc. IEEE International Symposium on Circuits and Systems, Vancouver, Canada, May 23-26, 2004.

[2] R.L.M. Peeters, J.M.H. Karel, R.L. Westra, S.A.P. Haddad and W.A. Serdijn, “Multiwavelet Design for Cardiac for cardiac signal processing”, 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC 2006), New York, August 30 - September 3, 2006.

[3] D.A. Johns, W.M. Snelgrove and A.S. Sedra, “Orthonormal ladder filters”, IEEE Transactions on Circuits and Systems, Vol. 36, No. 3, pp. 337-343, March 1989.

# A Structured Wavelet Filter Design

**Michiel A. Grashuis, Wouter A. Serdijn**



In Wearable Implantable Medical Devices (WIMD) power consumption is a big issue. In order to reduce power consumption we propose analog preprocessing by means of an analog wavelet filter bank before the signal enters the Analog to Digital Converter (ADC). This will allow for a reduction in the resolution of the ADC and hence a reduction in power consumption.

Old

$$\begin{tikzcd}\text{Input Signal} \arrow[r] & \boxed{\text{ADC}} \arrow[r] & \boxed{\text{DSP}}\end{tikzcd}$$

- High ADC resolution
- Large power consumption

New

$$\begin{tikzcd}\text{Input Signal} \arrow[r] & \boxed{\text{Wavelet Filter}} \arrow[r] & \boxed{\text{ADC}} \arrow[r] & \boxed{\text{DSP}}\end{tikzcd}$$

- Analog preprocessing
- Reduced ADC resolution
- Reduced Power consumption

Why wavelet filter?

- Performs the wavelet transform.
- Provides local analysis of nonstationary signals.
- Looks at the signal at various windows and analyses it with various resolutions.

How to design? A programme is developed which can generate the wavelet filter topology.

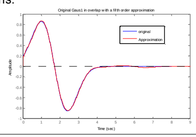
The programme offers the designer a flexible design approach. Starting from the input signal, which a time domain function description, the designer has the following choices.

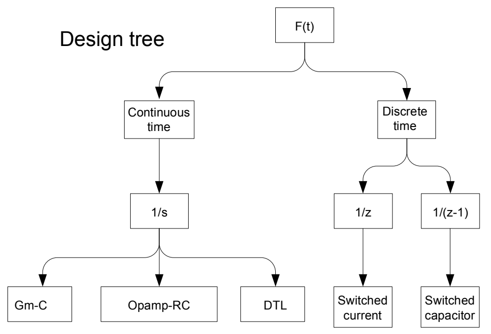
- The order of approximation. (accuracy versus power consumption).
- A choice between continuous time (C.T.) or discrete time (D.T.) solution.
- The nature of the elementary building blocks.
- The implementation of the elementary building blocks.

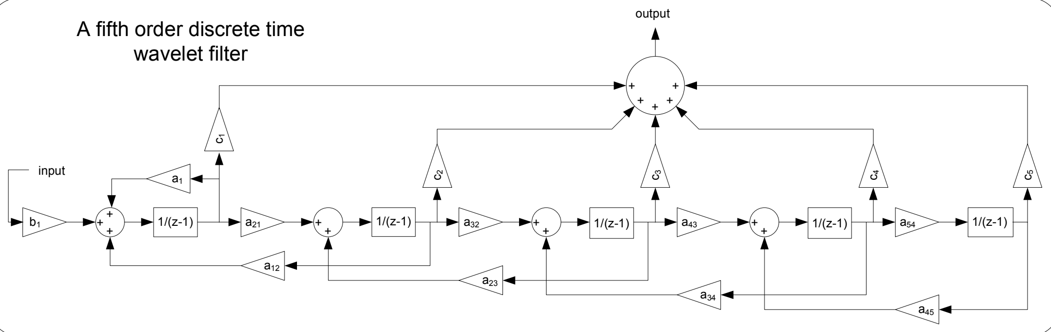
The output → Orthonormal ladder topology → Wavelet filter with

- Close-to-optimal dynamic range.
- Minimal power consumption.
- Minimal sensitivity to component variations.

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 \\ a_{21} & 0 & a_{23} & 0 & 0 \\ 0 & a_{32} & 0 & a_{34} & 0 \\ 0 & 0 & a_{43} & 0 & a_{45} \\ 0 & 0 & 0 & a_{54} & 0 \end{bmatrix}$$







Using this structured design methodology a fifth order ultra low power switched capacitor wavelet filter is designed. The methodology employed can be adopted to for the design of electronic filters to implement any arbitrary transfer function or impulse response.

For more information: contact Wouter A. Serdijn, w.a.serdijn@tudelft.nl

## References

[1] S.A.P. Haddad, “Ultra Low-power biomedical signal Processing”, PhD Thesis, Delft University of technology, December 2006.

[2] G.A. Baker jr., “Essentials of Pade approximants”, Academic Press New York, 1975.

[3] A.J. van der Veen, E.F. Deprettere and A.L. Swindlehurst, “Subspace based Signal Analysis using Singular Value Decomposition”, Proceedings of the IEEE, vol.81, pp. 1277-1308, September 1993.

[4] J.M.H. Karel, R.L.M. Peeters, R.L. Westra, S.A.P. Haddad and W.A. Serdijn, “An $L_2$-based approach for wavelet approximation”, 44th IEEE Conference on Decision and Control, and European Control Conference, pp.7882-7887, Seville, Spain, December 2005.

[5] V. Verdult, J. Scherpen and T. van den Boom, “Introduction to Modeling and System Analysis”, Reader SC2010ET, Delft University of technology, January 2004.

[6] A. Guillen, “Realization of a low power analog SC-MOS filter that implements the Morlet wavelet transform to analyze ECG signals in pace makers”, MsC Thesis, Delft University of technology.

[7] C. Sawigun, M. Grashuis, R. Peeters and W.A. Serdijn, “Micropower Sampled-data Wavelet Filter Design Using Switched-Gain Cell Technique”, IEEE ISCAS, May 2009.

[8] P.R. Gray, P.J. Hurst, S.H.Lewis and R.G. Meyer, “Analysis and design of analog integrated circuits”, John Wiley & Sons, 2005.

[9] O. Choksi and L.R. Carley, “Analysis of Switched –Capacitor Common-Mode Feedback Circuit”, IEEE transactions on circuit and systems, Vol 50, No 12, December 2003.

[10] D. Rocha, “Optimal design of analogue Low-power system”, PhD Thesis, Delft University of technology, April 2003.

[11] Klaas Bult, “Introduction to analog CMOS design”, Master course ET 4295, Delft University of technology.

[12] G. Groenewold, “Optimal dynamic range integrators”, IEEE Transactions on Circuits and Systems-I: Fundamental theory and applications, 39(8):614-27, August 1992.