# Time domain simulation of Sigma Delta A/D converters using Ptolemy

Frank Schilder
TU Ilmenau

September 16, 1998

## Contents

## 1 Installing and using the package

### Installing

Note: Throughout this documentation I will use 'A/D ...' as a synonym for '$\Sigma\Delta$-A/D ...' because no other A/D modulators or converters are considered here.

For installing the A/D converter simulation package do the following steps:

1. Ptolemy must be installed on your System except you are interested in the source code only. If Ptolemy is not installed you can get it and its documentation from *http://ptolemy.eecs.berkeley.edu/*.

2. If not already done, download the file `adc.tgz` via *ftp://ftp.mathematik.tu-ilmenau.de/pub/numerik/adc.tgz*.

3. Move the file `adc.tgz` to the location where you want to install this package.

4. Untar the file with the command `gtar -xzf adc.tgz`. (Note: `gtar` may be renamed as `tar`.) If gnu's tar is not installed on your system try the following:
```
mv adc.tgz adc.tar.gz
gunzip adc.tar.gz
tar -xf adc.tar
```
(If this still does not work consider consulting your system administrator.) This step will create two directories `adc` and `ivps` as well as the files `INSTALL` and `adc.ps`. `adc` contains the simulation package described in this and the next section, `ivps` some `ivps` scripts refered to in the last section of this documentation, `INSTALL` is a copy of these installation instructions and `adc.ps` this documentation itself.

5. Change to the directory `adc`: `cd adc`.

6. Run the script `setup`. This will invoke the Ptolemy utility program `masters` to set the paths to correct values. (Note: If you move the package, you must invoke `setup` again.) If your Ptolemy is in standard installation, no further adjustments are necessary. You may check this with `masters`: Type `masters facets/ad1` and enter the command `"?"`, if the path starting with `$PTOLEMY` is labeled as invalid then you need to set this too. Ask your system administrator which paths are changed to what values. Look at the command in `setup` how to set these paths for all facets at once.

7. Now you are ready to use the package, the central facet is in `adc/adc.pal`, open it with the open-facet-command (F) within Ptolemy.

### Using the simulation package within Ptolemy

In the rest of this documentation I assume you are familar with using Ptolemy. If not please read 'The Micro Almagest' [5] for an introduction. This package runs completely in SDF domain, so you may read the chapters related to this domain only.

In the description below there are also given the model equations. This simulation deals with mixed analog-digital circuits. We use the clock rate of the digital part also for the numerical simulation of the analogous part (because of efficiency and simpleness). The interval between two time tacts is denoted as $\Delta t$. All the terms in the given model equations below are related to grid points of the time grid $t_k = t_0 + k \cdot \Delta t, \ \ k = 0, 1, \ldots$. The lower left index indicates the grid point on which the considered value is defined (e.g. $U_k^{in}$ means the value of the input signal $U^{in}(t)$ at the time $t_k$, i.e. $U_k^{in} = U^{in}(t_k)$).

Now start the Ptolemy vem console `ptiny` (or `pigi` whatever you prefer) and open the facet `adc/adc.pal` with the open facet (F) command. You will see a window like fig. 1. It is seperated in four parts: 'basic components', 'A/D modulators and converters', 'signal sinks and sources' and 'complete simulations'. The first part contains the stars (blue outline) that are the heart of this package:

**F 2 M, M 2 F:** These stars are datatype conversion stars (*float to matrix*, *matrix to float*) that convert a single floating point number into a matrix resp. extract a single floation point number out of a matrix. These stars are needed for connecting to and from 'A/D Integrator'-stars. The source of these stars is located in `adc/stars/SDFf2m.pl` resp. `adc/stars/SDFm2f.pl`.

**A/D Integrator:** This is the central star of this package – the integrator. It inputs a matrix generated either by a previous 'A/D Integrator' or a 'F 2 M' conversion star (see also the examples described below) and an intervallwise constant input on the feedback gain. 'Intervallwise constant' means that this input *must not change* within two time tacts. It may only be set to a new value at every time tact. The output is a matrix with a complicated datastructure (see the next section on this) especially designed for cascading of 'A/D Integrator's. If you are intrested in the ('real world') output signal only you may extract this with a 'M 2 F' conversion star from the output. Its function is to integrate the difference between the input signal $U^{in}$ and the feedback signal $U^{fb}$ over a time interval $\Delta t$. This star has several states that can be set. Summary:

| **states** | **description** |
|---|---|
| U0 | initial output voltage at time $t_0$ |
| T | the (inverse) integration constant of the integration circuit |
| dt | the (inverse) clock rate |
| particles consumed | 1 |
| particles fired | 1 |
| **source** | `adc/stars/SDFadint.pl` |
| **model eqn.** | $U_k^{out} = \frac{1}{T} \int_{t_0}^{t_k} (U^{in}(\tau) - U^{fb}(\tau)) d\tau + U_0$ |

$U^{out}$ ist the output signal, $T$ the integration constant, $U^{in}$ the input signal, $U^{fb}$ the feedback, $U_0$ the initial value for $U^{out}$ at $t_0$ and $t_{k+1} - t_k$ is a time-interval between two time-tacts.

**b quant:** The binary quantizer. It inputs a matrix generated by an 'A/D Integrator' and outputs a floating point value. It converts the matrix input implicitely to a floating point number. Its function is to compare the input signal $U^{in}$ with the threshold value $s$ and output 1 resp. 0 if the signal is greater than or equal to $s$ resp. less than $s$. Summary:

| **states** | **description** |
|---|---|
| s | the threshold value $s$ |
| particles consumed | 1 |
| particles fired | 1 |
| **source** | `adc/stars/SDFbinquant.pl` |
| **model eqn.** | $X_k = \theta(U_k^{in} - s)$ |

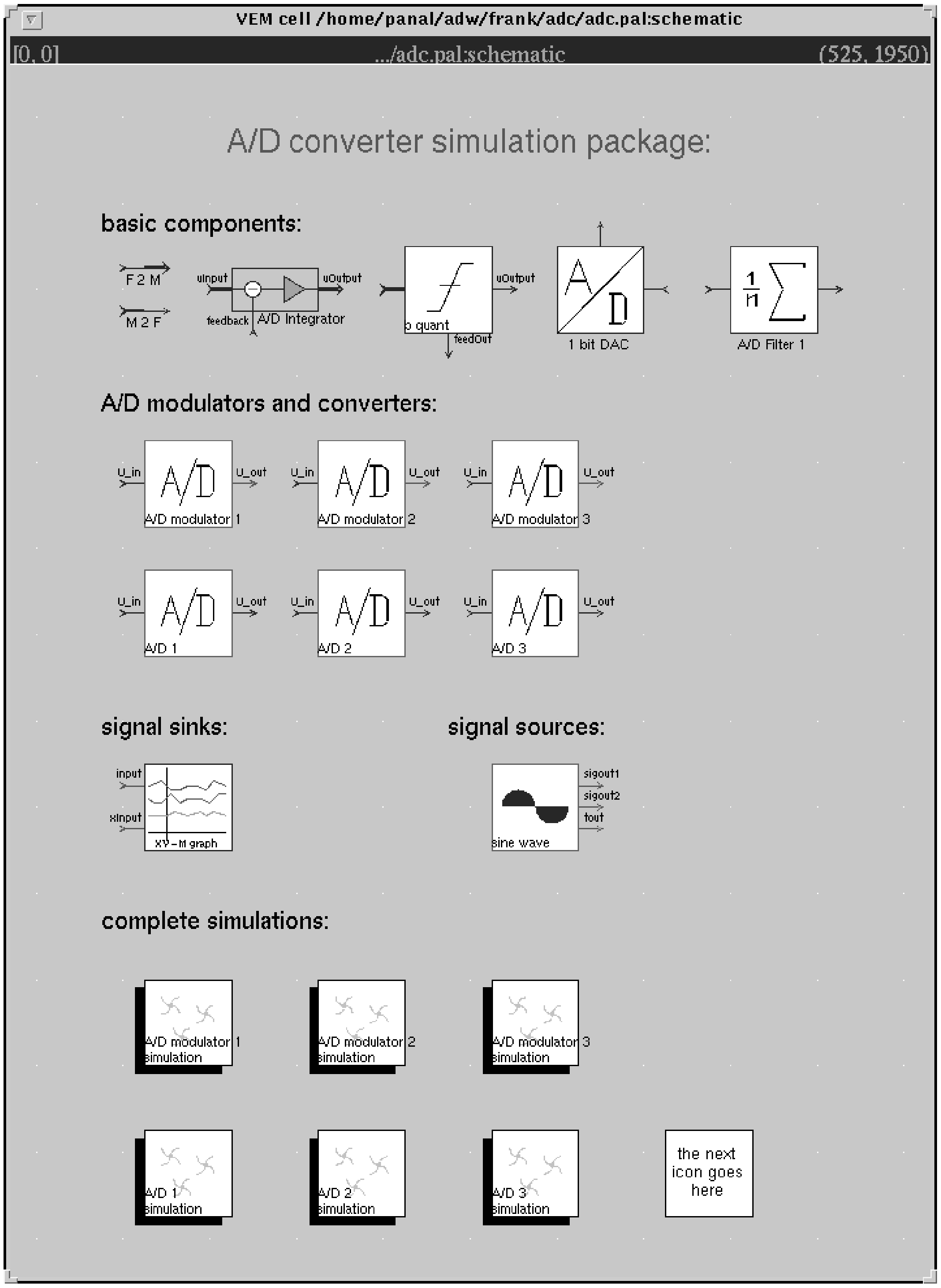


Figure 1: The facet `adc/adc.pal` containing all components of this simulation package.

$\theta$ is the Heaviside function $\theta(x) = \begin{cases} 1 \text{ for } x \geq 0 \\ 0 \text{ for } x < 0 \end{cases}$. $X_k$ is for $k = 0, 1, \ldots$ the bit-sequence generated by an A/D-modulator (see below for the definition of the terms 'A/D modulator' and 'A/D converter' used in this documentation).

**1 bit DAC:** A one bit digital analog converter. It inputs a floating point number $X_k$ usually generated by a binary quantizer and outputs a floating point number. Its primary function is to convert a bit sequence into a rectangular signal scaled with $\alpha U^{ref}$, where $U^{ref}$ is the reference voltage, and is usually used in the feedback loop to our 'A/D Integrator'. If e.g. the input sequence is a bit sequence (for instance from our binary quantizer) then the output is a rectangular signal with the values $-\alpha U_{ref}$ and $\alpha U_{ref}$ only. Summary:

| **states** | **description** |
|---|---|
| U_ref | the reference voltage $U^{ref}$ |
| alpha | the scaling factor $\alpha$ |
| particles consumed | 1 |
| particles fired | 1 |
| **source** | `adc/stars/SDFdau.pl` |
| **model eqn.** | $U_k^{out} = \alpha U^{ref} X_k$ |

$X_k$ is a bit sequence but in general it may contain arbitrary values between (including) 0 and 1. So you are able to simulate A/D modulators/converters containing multivalue quantizers. Remember: The only condition for the feedback input of our 'A/D Integrator' is that its value be constant between two time tacts $t_{k-1}$ and $t_k$.

**A/D Filter 1:** A simple filter for bit sequences. It inputs a sequence of floating point numbers $X_k$ usually generated by a binary quantizer and outputs a floating point value $Y_k$. Its function is to calculate the sliding average over $2^b - 1$ values to restore a signal from a bit sequence. $b$ is the bit width of the filter (it generates values from 0 to $2^b - 1$). This star is called 'A/D Filter 1' because it uses the simplest method for performing the restauration of a signal. In this implementation the star 'A/D Filter 1' implicitely rescales its output between $-U_{ref}$ and $+U_{ref}$. If you rely on the integer values you may remove the scaling part from the source (or better outcomment it). Summary:

| **states** | **description** |
|---|---|
| bits | the bit width $b$ |
| U_ref | the reference voltage $U^{ref}$ |
| particles consumed | 1 |
| particles fired | 1 |
| **source** | `adc/stars/SDFfilter.pl` |
| **model eqn.** | $Y_k = U^{ref}(\frac{2}{2^n-1}(\sum_{i=k-2^n+2}^{k} X_i) - 1)$ |

$Y_k$ is the generated (restored) signal value at $t_k$. For the initial calculation it is assumed that all bits $X_i$ are zero (i.e. where $i$ would have negative values in the sum above).

You get a short description for every star with the Ptolemy profile command (,). This description includes the default values for the states and the names of the input and output ports.

The second part 'A/D modulators and converters' contains galaxies (green outline) assambled with the stars described above that simulate several A/D modulators resp. converters. In this documentation the term **A/D modulator** is used for an electronic component that **inputs an analog signal** $U$ and **outputs a bit sequence** $X_k$. In distinction to that **A/D converter** is used if the **output signal is a reconstruction of the input signal**. In our case that means an A/D modulator with an additional filtering stage gives an A/D converter. The software-design of these components reflects that fact (do a look inside (i)). There are two different types of galaxies there:

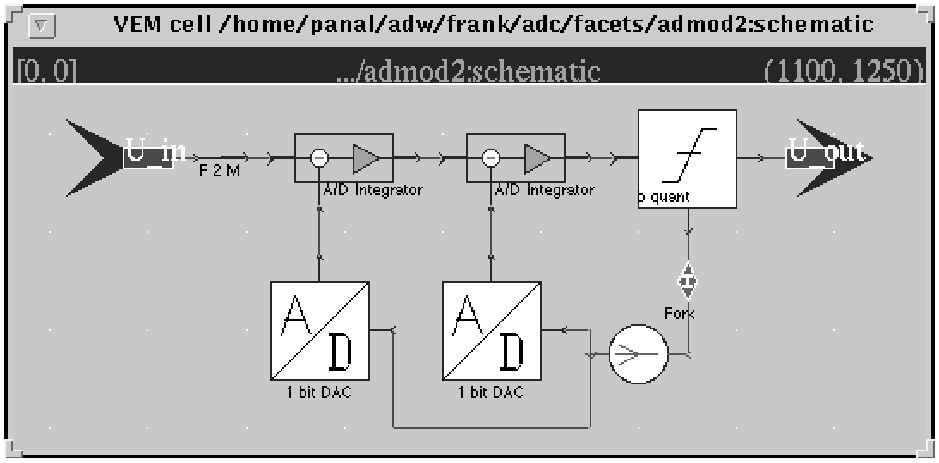


Figure 2: The facet of the A/D modulator of second order, containing two cascaded integrators, a binary quantizer and two 1 bit DACs (and some Ptolemy system stars as well).

**A/D modulators:** (first row) These are A/D modulators of order one, two and three (from left to right). If you look inside (i) you will see a window like fig. 2. This star gets a continuous signal as input and outputs a bit sequence that (hopefully) correlates in some way with the input signal. The mathematical model of these stars is subject of the last section. Summary:

| **states** | **description** |
|---|---|
| T1, T2, T3 | integration constants of 1st, 2nd and 3rd integrator* |
| alpha, beta, gamma | scaling factors of the feedback DACs* |
| X0 | initial output state of the binary quantizer |
| U_ref | the reference voltage $U^{ref}$ |
| s | the threshold value of the bin. quantizer |
| dt | the (inverse) clock rate |
| U1_0, U2_0, U3_0 | the initial output states of the integrators* |
| particles consumed | 1 |
| particles fired | 1 |

* The actual number of these parameters equals the number of integrators of the modulator considered. In the first order A/D modulator the numbering is omitted.

**A/D converters:** (second row) These are A/D converters of order one, two and three (from left to right). If you look inside (i) you will see a window like fig. 3. This star gets a continuous signal as input and outputs the restored signal. Remember, the output is already rescaled between $-U_{ref}$ and $+U_{ref}$. This is automatically changed if you change the implementation of the 'A/D Filter 1' (see above). The mathematical description of A/D modulators applies to this stars as well (see last section). The difference is the additional stage of filtering. Summary:

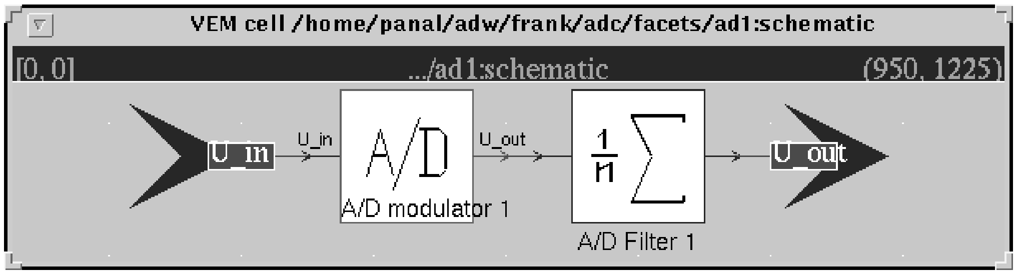


Figure 3: The facet of the A/D converter of first order, containing an A/D modulator of first order and a b-bit filter.

| **states** | **description** |
|---|---|
| T1, T2, T3 | integration constants of 1st, 2nd and 3rd integrator* |
| alpha, beta, gamma | scaling factors of the feedback DACs* |
| bits | the bit width $b$ of the 'A/D Filter 1' |
| X0 | initial output state of the bin. quantizer |
| U_ref | the reference voltage $U^{ref}$ |
| s | the threshold value of the bin. quantizer |
| dt | the (inverse) clock rate |
| U1_0, U2_0, U3_0 | the initial output states of the integrators* |
| particles consumed | 1 |
| particles fired | 1 |

* The actual number of these parameters equals the number of integrators of the converter considered. In the first order A/D converter the numbering is omitted.

The third part 'signal sinks and sources' contains two elements only, the plotting device 'XY-Mgraph' and the 'sine wave' signal generator. The 'XY-Mgraph' is derived from 'XMgraph' of the Ptolemy package. See 'The Micro Almagest' [5] for a description. The 'XY-Mgraph' is not yet tuned for plotting rectangular funktions as are generated by our components. Therefore the graphics appears somewhat right-shifted (by '$\Delta t$'). You are welcome to change this (you will need two seperate input ports, one for 'ordinary' input and the other for 'rectangular' input).

The signal generator 'sine wave' has two settable states, 'dt' and 'amplitude'. 'dt' is the (inverse) clock rate at which particles are fired (see also 'dt' in the components above) and 'amplitude' sets the amplitude of the generated sine wave.

The last part 'complete simulations' contains six example simulations, one for each modulator resp. converter. To start a particular simulation open the facet (look inside (i)) and invoke the run command (R). You are asked then 'when to stop'. Accept this time the default of 400 iterations and press the 'go' button. You will get a graphical output like one of fig. 5.

All of these simulations have in common that all components agree about 'dt'. *This is a basic condition for correct working of the simulations.* For becoming familar with the package change some parameters (with the edit parameter command (e)) and maneuver through the components (look inside (i)). Try to create a simulation for a fourth order A/D modulator (correct linking of the parameters costs some nerves). Look at the examples how to correctly forward the parameters to the stars. You may also consult 'The Micro Almagest' [5], part 'Using Galaxies'.

Now create your own A/D modulators resp. converters and have Ptolemy simulate them!

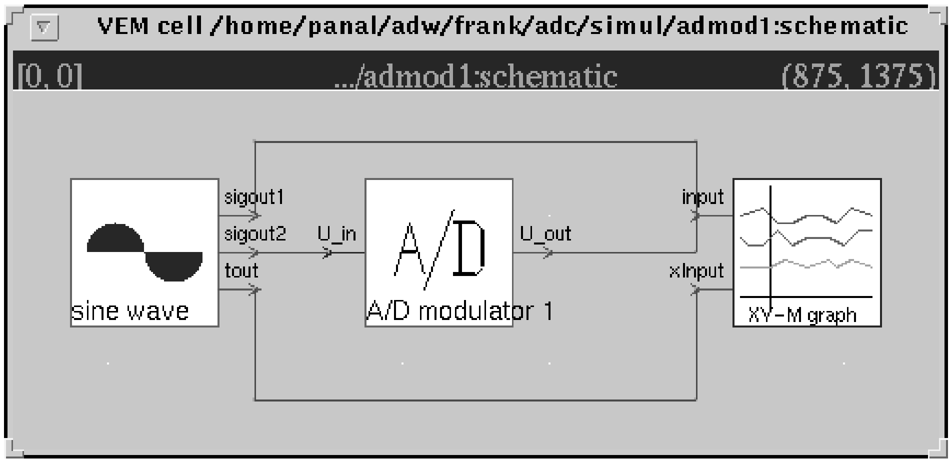


Figure 4: The facet of the 'A/D modulator 1 simulation' containing the signal source 'sine wave', the 'A/D modulator 1' and the signal sink 'XY-Mgraph'.

## 2 Mathematical description of the implementation

If you are going to change or extend the algorithms described in the following you should be familar with the Ptolemy programming interface. See therefore 'The Almagest Vol. 2' [6].

All basic elements except the 'A/D Integrator' have fairly simple model equations and so their implementation is straightforeward. The algorithm of the 'A/D Integrator' requires a closer look. Because of the intended use of our package for long term simulations we need the following conditions (goals) satisfied:

1. It should work with high precision, i.e. we wish results with high order error terms (like usual high order integration methods).
2. It must be cascadable, i.e. the output must carry enough information so that the next stage is able to satisfy condition 1 again.

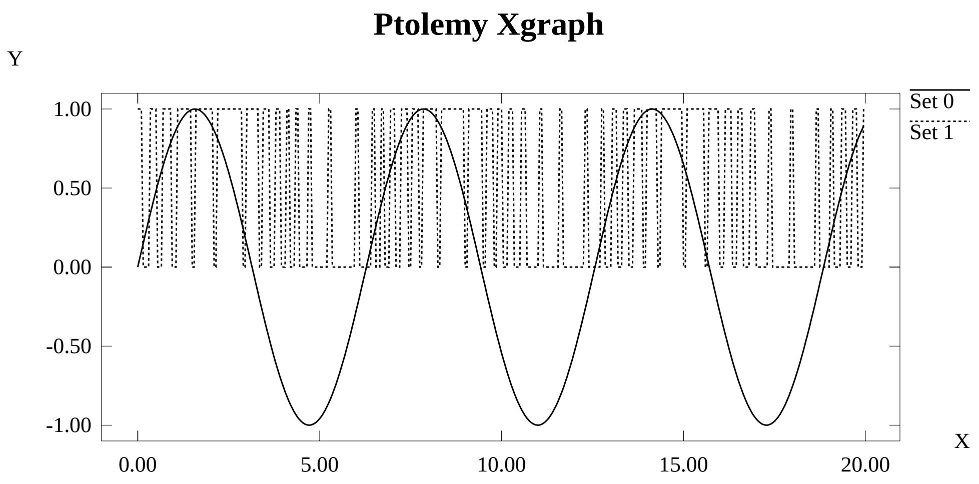
If we look closer at our model equation of an integrator at stage $n$, omitting the initial values:

$$\begin{aligned} U_k^{out} &= \frac{1}{T_n}\int_{t_0}^{t_k}\left(\frac{1}{T_{n-1}}\int_{t_0}^{\tau_n}\left(\cdots\frac{1}{T_1}\int_{t_0}^{\tau_2}(U^{in}(\tau_1)-U^{fb_1}(\tau_1))d\tau_1-\cdots\right.\right. \\ &\left.\left. -U^{fb_{n-1}}(\tau_{n-1})\right)d\tau_{n-1}-U^{fb_n}(\tau_n)\right)d\tau_n \qquad (1) \end{aligned}$$

we see where the problem arises. We cannot simply use a high order numerical method because the $U^{fb_i}(t)$ are discontinuous functions of time. However, any numerical integration method of order $p$ requires that the integrated function be $p$ times continuous differentiable otherwise the method is likely to fail to give useful results.

The idea now is to split up the integral in (1) into two main parts, one containing $U^{in}$ and the other containing the feedbacks $U^{fb_i}$:,

$$U_k^{out} = \frac{1}{T_n}\int_{t_0}^{t_k}\frac{1}{T_{n-1}}\int_{t_0}^{\tau_n}\cdots\frac{1}{T_1}\int_{t_0}^{\tau_2}U^{in}(\tau_1)d\tau_1\cdots d\tau_{n-1}d\tau_n$$



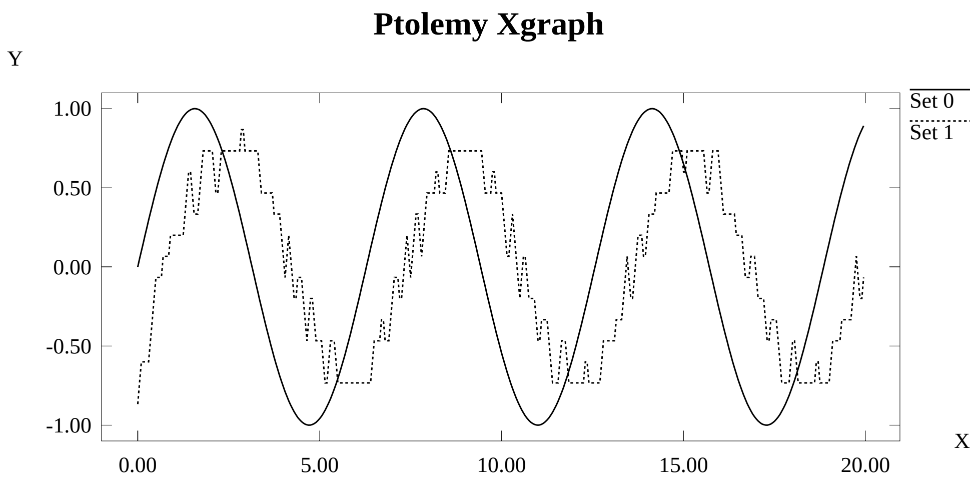


Figure 5: The output of 'A/D modulator 2 simulation' (top) and 'A/D 2 simulation' (bottom) with $U^{in}(t) = \sin(t)$, $U^r = \alpha = \beta = T_1 = T_2 = 1$, $\Delta t = 0.05$ and 4 bit filtering.

$$
\begin{aligned}
&- \frac{1}{T_n}\int_{t_0}^{t_k}\frac{1}{T_{n-1}}\int_{t_0}^{\tau_n}\cdots\frac{1}{T_1}\int_{t_0}^{\tau_2}U^{fb_1}(\tau_1)d\tau_1\cdots d\tau_{n-1}d\tau_n \\
&- \ldots \\
&- \frac{1}{T_n}\int_{t_0}^{t_k}\frac{1}{T_{n-1}}\int_{t_0}^{\tau_2}U^{fb_{n-1}}(\tau_1)d\tau_1 d\tau_2 \\
&- \frac{1}{T_n}\int_{t_0}^{t_k}U^{fb_n}(\tau)d\tau \qquad (2)
\end{aligned}
$$

which are then treated seperately. In the first integral the function $U^{in}(t)$ fullfils our global assumtion of being sufficiently smooth so we can use a standard integration method. For the other integrals we try to derive a recursion formula that gives exact results. It is clear that by combining these results we obtain a method of overall error order of the used numerical integration methods.

To calculate the multiple integral over $U^{in}$ we rewrite it as a differential equation of order $n$ (integrating a differential equation is much more efficient than calculating a multiple integral):

$$
(U_k^{out}(t))^{(n)} = \frac{1}{T}U^{in}(t). \qquad (3)
$$

For applying numerical methods we would rewrite (3) as a system of order 1. Because of efficiency we then choose implicit Adams-Moulton multistep methods for integration and a simple calculation shows that the cascaded application of this methods is here equivalent to simultaneous solving the system (because the right hand side of (3) depends on time only). This is not shown here explicitely but fairly simple. Because of that the integrator simply applies the Adams-Moulton method to the $n-1$ times integral value produced by the previous integrator. For completeness, the Adams-Moulton methods used are given here, the implementation is straightforward:

| order | formula |
|---|---|
| 1 | $U_k^{out} = U_{k-1}^{out} + \frac{\Delta t}{T}U_k^{in}$ |
| 2 | $U_k^{out} = U_{k-1}^{out} + \frac{\Delta t}{2T}(U_k^{in} + U_{k-1}^{in})$ |
| 3 | $U_k^{out} = U_{k-1}^{out} + \frac{\Delta t}{12T}(5U_k^{in} + 8U_{k-1}^{in} - U_{k-2}^{in})$ |
| 4 | $U_k^{out} = U_{k-1}^{out} + \frac{\Delta t}{24T}(9U_k^{in} + 19U_{k-1}^{in} - 5U_{k-2}^{in} + U_{k-3}^{in})$ |
| 5 | $U_k^{out} = U_{k-1}^{out} + \frac{\Delta t}{720T}(251U_k^{in} + 646U_{k-1}^{in} - 264U_{k-2}^{in} + 106U_{k-3}^{in} - 19U_{k-4}^{in})$ |
| 6 | $U_k^{out} = U_{k-1}^{out} + \frac{\Delta t}{1440T}(475U_k^{in} + 1427U_{k-1}^{in} - 798U_{k-2}^{in} + 482U_{k-3}^{in} - 173U_{k-4}^{in} + 27U_{k-5}^{in})$ |

Now we derive an exact recursion formula for the multiple integrals over the feedbacks $U^{fb_i}$. For simplicity we denote the intervallwise constant function $U^{fb_1}$ as $x$ and go to calculate

$$
I_k^n = \frac{1}{T_n}\int_{t_0}^{t_k}\frac{1}{T_{n-1}}\int_{t_0}^{\tau_n}\cdots\frac{1}{T_1}\int_{t_0}^{\tau_2}x(\tau_1)d\tau_1\cdots d\tau_{n-1}d\tau_n \qquad (4)
$$

where $I_k^n$ denotes the nth integral at $t_k$. What we try now is to get a formula that depends on the previous integral values and the last recent function value only. Therefore we divide the integral (4) into two parts:

$$
I_k^n = \frac{1}{T_n}\int_{t_0}^{t_{k-1}}\frac{1}{T_{n-1}}\int_{t_0}^{\tau_n}\cdots\frac{1}{T_1}\int_{t_0}^{\tau_2}x(\tau_1)d\tau_1\cdots d\tau_{n-1}d\tau_n
$$

$$+\frac{1}{T_n}\int_{t_{k-1}}^{t_k}\frac{1}{T_{n-1}}\int_{t_0}^{\tau_n}\cdots\frac{1}{T_1}\int_{t_0}^{\tau_2}x(\tau_1)d\tau_1\cdots d\tau_{n-1}d\tau_n \tag{5}$$

what we can rewrite as:

$$I_k^n = I_{k-1}^n+\frac{1}{T_n}\int_{t_{k-1}}^{t_k}\frac{1}{T_{n-1}}\int_{t_0}^{\tau_n}\cdots\frac{1}{T_1}\int_{t_0}^{\tau_2}x(\tau_1)d\tau_1\cdots d\tau_{n-1}d\tau_n. \tag{6}$$

The remaining integral can also being rewritten:

$$\begin{aligned}
&\frac{1}{T_n}\int_{t_{k-1}}^{t_k}\frac{1}{T_{n-1}}\int_{t_0}^{\tau_n}\cdots\frac{1}{T_1}\int_{t_0}^{\tau_2}x(\tau_1)d\tau_1\cdots d\tau_{n-1}d\tau_n\\
&= \frac{1}{T_n}\int_{t_{k-1}}^{t_k}\left(\frac{1}{T_{n-1}}\int_{t_0}^{t_{k-1}}\cdots\frac{1}{T_1}\int_{t_0}^{\tau_2}x(\tau_1)d\tau_1\cdots d\tau_{n-1}+\frac{1}{T_{n-1}}\int_{t_{k-1}}^{\tau_n}\cdots\frac{1}{T_1}\int_{t_0}^{\tau_2}x(\tau_1)d\tau_1\cdots d\tau_{n-1}\right)d\tau_n\\
&= \frac{1}{T_n}\int_{t_{k-1}}^{t_k}I_{k-1}^{n-1}d\tau_n+\frac{1}{T_n}\int_{t_{k-1}}^{t_k}\frac{1}{T_{n-1}}\int_{t_{k-1}}^{\tau_n}\cdots\frac{1}{T_1}\int_{t_0}^{\tau_2}x(\tau_1)d\tau_1\cdots d\tau_{n-1}d\tau_n\\
&= \frac{1}{T_n}I_{k-1}^{n-1}\Delta t+\frac{1}{T_n}\int_{t_{k-1}}^{t_k}\frac{1}{T_{n-1}}\int_{t_{k-1}}^{\tau_n}\cdots\frac{1}{T_1}\int_{t_0}^{\tau_2}x(\tau_1)d\tau_1\cdots d\tau_{n-1}d\tau_n. \qquad (7)
\end{aligned}$$

If we insert (7) into (6) we get:

$$I_k^n = I_{k-1}^n+\frac{1}{T_n}I_{k-1}^{n-1}\Delta t+\frac{1}{T_n}\int_{t_{k-1}}^{t_k}\frac{1}{T_{n-1}}\int_{t_{k-1}}^{\tau_n}\cdots\frac{1}{T_1}\int_{t_0}^{\tau_2}x(\tau_1)d\tau_1\cdots d\tau_{n-1}d\tau_n. \tag{8}$$

This process can be continued until we arrive at the following equation:

$$\begin{aligned}
I_k^n &= I_{k-1}^n+\frac{1}{T_n}I_{k-1}^{n-1}\Delta t+\frac{1}{T_nT_{n-1}}I_{k-1}^{n-2}\frac{\Delta t^2}{2}+\cdots\\
&\quad +\frac{1}{T_n}\int_{t_{k-1}}^{t_k}\frac{1}{T_{n-1}}\int_{t_{k-1}}^{\tau_n}\cdots\frac{1}{T_1}\int_{t_{k-1}}^{\tau_2}x(\tau_1)d\tau_1\cdots d\tau_{n-1}d\tau_n.\\
&= \sum_{i=0}^{n-1}\frac{1}{\prod_{j=1}^{i}T_{n-j+1}}I_{k-1}^{n-i}\frac{\Delta t^i}{i!}+\frac{1}{T_n}\int_{t_{k-1}}^{t_k}\frac{1}{T_{n-1}}\int_{t_{k-1}}^{\tau_n}\cdots\frac{1}{T_1}\int_{t_{k-1}}^{\tau_2}x(\tau_1)d\tau_1\cdots d\tau_{n-1}d\tau_n. \qquad (9)
\end{aligned}$$

Because $x(t)$ is constant in the interval $[t_{k-1},t_k)$ we can solve the last remaining integral and end up with:

$$I_k^n = \sum_{g=0}^{n-1}\frac{1}{\prod_{j=1}^{g}T_{n-j+1}}I_{k-1}^{n-g}\frac{\Delta t^g}{g!}+\frac{1}{T_nT_{n-1}\cdots T_1}U_{k-1}^{fb_1}\frac{\Delta t^n}{n!} \tag{10}$$

where $x$ has already been replaced with $U^{fb_1}$.

This derivation process can be repeated for the multiple integrals over the other feedbacks $U^{fb_i}$ ($i$ indicates the stage of the integrator that received the feedback $U^{fb_i}$) we finally obtain:

$$ {}^iI_k^{n-i+1} = \sum_{g=i-1}^{n-1} \frac{1}{\prod_{j=1}^{g-i+1} T_{n-j+1}} {}^iI_{k-1}^{n-g} \frac{\Delta t^{g-i+1}}{(g-i+1)!} + \frac{1}{T_n T_{n-1}\cdots T_i} U_{k-1}^{fb_i} \frac{\Delta t^{n-i+1}}{(n-i+1)!} \tag{11} $$

With (11) at hand we can go to formulate the (somewhat messy) algorithm to use. As it suggests we need for calculating the multiple integral intermediate results from earlier stages of 'A/D Integrator's. The easiest way to give the required data trough is using the following matrices:

$$ \text{Input} = \begin{pmatrix} {}^{n-1}U_k^{out} & \hat{U}_k^{n-1} & & & \\ \hat{U}_k^{fb_{n-1}} & {}^{n-1}\hat{I}_k^1 & & & \\ \hat{U}_k^{fb_{n-2}} & {}^{n-2}\hat{I}_k^1 & {}^{n-2}\hat{I}_k^2 & & \\ \vdots & \vdots & \vdots & \ddots & \\ \hat{U}_k^{fb_1} & {}^1\hat{I}_k^1 & {}^1\hat{I}_k^2 & \cdots & {}^1\hat{I}_k^{n-1} \end{pmatrix} \tag{12} $$

$$ \text{Output} = \begin{pmatrix} {}^nU_k^{out} & \bar{U}_k^n & & & & \\ \bar{U}_k^{fb_n} & {}^nI_k^1 & & & & \\ \bar{U}_k^{fb_{n-1}} & {}^{n-1}\bar{I}_k^1 & {}^{n-1}I_k^2 & & & \\ \bar{U}_k^{fb_{n-2}} & {}^{n-2}\bar{I}_k^1 & {}^{n-2}\bar{I}_k^2 & {}^{n-2}I_k^3 & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \\ \bar{U}_k^{fb_1} & {}^1\bar{I}_k^1 & {}^1\bar{I}_k^2 & {}^1\bar{I}_k^3 & \cdots & {}^1I_k^n \end{pmatrix} \tag{13} $$

where the individual terms have the following meaning (fields not filled are ignored):

| | | |
|---|---|---|
| ${}^{n-1}U_k^{out}$, ${}^nU_k^{out}$ | : | the output signal of the n-1st resp. nth integrator |
| $\hat{U}_k^{n-1}$, $\bar{U}_k^n$ | : | the n-1st resp. nth integral over $U^{in}$ |
| $\hat{U}_k^{fb_i}$ | : | $\hat{U}_k^{fb_i} = \frac{1}{T_{n-1}\cdots T_i} U_k^{fb_i} \frac{\Delta t^{n-i}}{(n-i)!}$ |
| $\bar{U}_k^{fb_i}$ | : | $\bar{U}_k^{fb_i} = \frac{1}{T_n} \hat{U}_k^{fb_i} \frac{\Delta t}{n-i+1}$ |
| ${}^i\hat{I}_k^j$ | : | ${}^i\hat{I}_k^j = \frac{1}{\prod_{l=j+i}^{n-1} T_l} {}^iI_k^j \frac{\Delta t^{n-i-j}}{(n-i-j)!}$ |
| ${}^i\bar{I}_k^j$ | : | ${}^i\bar{I}_k^j = \frac{1}{T_n} {}^i\hat{I}_k^j \frac{\Delta t}{n-i-j+1}$ |
| ${}^iI_k^{n-i+1}$ | : | see below step 4 of algorithm |

With this notation we can verbally formulate our algorithm:

1. read the input (signalmatrix and feedback), assamble an intermediate matrix from the feedback and the inputmatrix
2. transform the intermediate matrix: convert the 'hated' values into the 'bared' values using the formulaes given just before
3. calculate the nth integral over the input signal $U^{in}$ using implicit Adams-Moulton methods
4. calculate the multiple integrals over the feedbacks for each stage using the now very simple looking formula:

$$ {}^iI_k^{n-i+1} = \bar{U}_{k-1}^{fb_i} + \sum_{j=1}^{n-i} {}^i\bar{I}_{k-1}^j + {}^iI_{k-1}^{n-i+1} \tag{14} $$

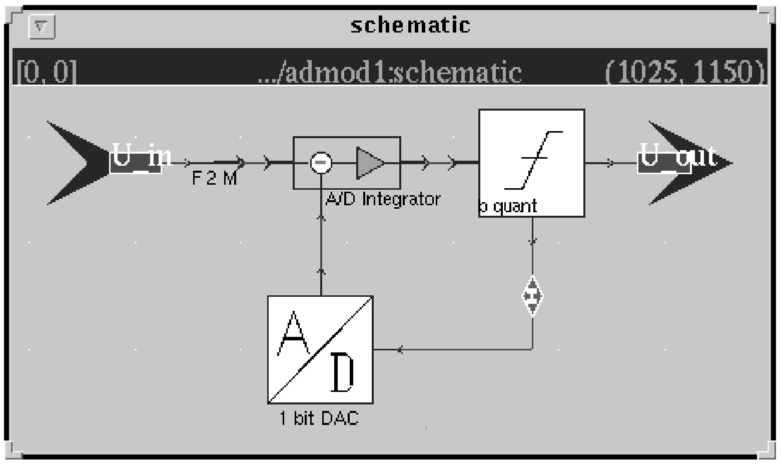


Figure 6: The facet of the A/D modulator of first order, containing one integrator, a binary quantizer and one 1 bit DACs (and some Ptolemy system stars as well).

5. calculate the output signal:

$$ {}^{n}U_k^{out} = \bar{U}_k^n - \sum_{i=1}^{n} {}^{i}\bar{I}_k^{n-i+1}, \tag{15} $$

   now the intermediate matrix is in state (13)

6. save one copy of the outputmatrix for the next iteration

7. output the outputmatrix

Some hints for further improvements:

- allowing the integration step size be smaller than the clock rate of the circuit, if $h$ is the integration step size then should be $\Delta t = m \cdot h$ with an integer $m$, this would allow the simulation with highly (fast) varying signals as input

- monitoring of the local integration error, this could give hints about the quality of the generated data as well as how (where) to improve the algorithms, see a book about numerics on ordinary differential equations how to do this

- with the monitoring of the local error one can go to add some adaptiveness, you can choose the order of the method depending on this error and a user-defined maximum tolerance

## 3 New ideas for analytical treatment

In this last section I would like to describe the basic concepts of an idea for merely analytical studies of A/D modulators and converters. The goal is to derive an ordinary differential equation that is related to our model and reflects the qualitative resp. approximates the quantitative behaviour of the A/D modulators and converters considered here.

First of all let us derive the model equation for a first order A/D modulator (A/DM1, see fig. 6). For brevity we call the real input signal $U^0$, the output signal $U^1$ and the reference voltage $U^r$.

We than can write:

$$U^1(t) = \frac{1}{T}\int\limits_{t_0}^{t} \left(U^0(\tau) - \alpha U^r\left(2\theta_k(U^1(\tau)-s)-1\right)\right)d\tau + U^1(t_0) \tag{16}$$

where $\theta_k$ denotes a special Heaviside function defined as:

$$\theta_k(x(t)) = \begin{cases} 1 & \text{for} \quad x(t_k) \geq 0, \quad t \in [t_k, t_{k+1}) \\ 0 & \text{for} \quad x(t_k) < 0, \quad t \in [t_k, t_{k+1}) \end{cases} \qquad t_k = t_0 + k\cdot\Delta t, \;\; k = 0, 1, \ldots$$

i.e. a $\theta$ function that tests its argument at the grid points $t_k$ only and keeps its value for the remaining interval. Again, we see where the difficulties come from, the nonlinearity is contained in the $\theta_k$ function which makes the use of classical methods nearly impossible. Also it is clear that this equation describes completely the A/DM1, the generated bit sequence is simply $\theta_k(U^1(\tau)-s)$ but introducing this into (16) would raise the complexity without adding any information.

If we denote the output of the second integrator in our cascaded A/DM2 (see fig. 2 page 5) with $U^2$ we get the following system of equations:

$$U^1(t) = \frac{1}{T_1}\int\limits_{t_0}^{t} \left(U^0(\tau) - \alpha U^r\left(2\theta_k(U^2(\tau)-s)-1\right)\right)d\tau + U^1(t_0) \tag{17}$$

$$U^2(t) = \frac{1}{T_2}\int\limits_{t_0}^{t} \left(U^1(\tau) - \beta U^r\left(2\theta_k(U^2(\tau)-s)-1\right)\right)d\tau + U^2(t_0) \tag{18}$$

It is easily seen that the system of equations for a cascaded A/DM of order $n$ are:

$$U^i(t) = \frac{1}{T_i}\int\limits_{t_0}^{t} \left(U^{i-1}(\tau) - \alpha_i U^r\left(2\theta_k(U^n(\tau)-s)-1\right)\right)d\tau + U^i(t_0), \;\; i = 1, 2, \ldots, n \tag{19}$$

where $U^i$ denotes the output signal of the ith integrator (the integrator at the ith stage).

Now we rewrite (19) as follows (omitting the $i = 1, 2, \ldots, n$):

$$\begin{aligned} U^i(t) &= \frac{1}{T_i}\int\limits_{t_0}^{t} U^{i-1}(\tau)d\tau - \frac{1}{T_i}\int\limits_{t_0}^{t} 2\alpha_i U^r\theta_k(U^n(\tau)-s)d\tau + \frac{1}{T_i}\int\limits_{t_0}^{t}\alpha_i U^r d\tau + U^i(t_0) && (20) \\ &= \frac{1}{T_i}\int\limits_{t_0}^{t} U^{i-1}(\tau)d\tau - \frac{2\alpha_i U^r}{T_i}\int\limits_{t_0}^{t}\theta_k(U^n(\tau)-s)d\tau + \frac{1}{T_i}\int\limits_{t_0}^{t}\alpha_i U^r d\tau + U^i(t_0). && (21) \end{aligned}$$

In the next step we are going to show, that:

$$\int\limits_{t_0}^{t}\theta_k(U^n(\tau)-s)d\tau \approx \int\limits_{t_0}^{t}\varphi(\tau)d\tau \tag{22}$$

is plausible with a so called 'density function' $\varphi$. We assume without loss of generality that $t$ be in the interval $t \in [t_0, t_0 + (m+1)\cdot\Delta t)$ (this $m$ can always be found). Then we can write:

$$\begin{aligned} \int\limits_{t_0}^{t}\theta_k(U^n(\tau)-s)d\tau &= \sum_{k=0}^{m-1}\theta_k(U^n(\tau)-s)\cdot\Delta t + \theta_m(U^n(\tau)-s)\cdot(t-t_m) && (23) \\ &= \sum_{k=0}^{m-1}\frac{\theta_k(U^n(\tau)-s)}{m}\cdot(t_m-t_0) + \theta_m(U^n(\tau)-s)\cdot(t-t_m). && (24) \end{aligned}$$

Now we divide the intervall $[t_0, t_m)$ into $l$ disjunced intervals $\mathcal{I}_j = [t_{m_{j-1}}, t_{m_j})$, $j = 1, 2, \ldots, l$, $m_0 = 0$ with $\cup_{j=1}^{l} \mathcal{I}_j = [t_0, t_m)$ and the sublengths $\Delta_j t = t_{m_j} - t_{m_{j-1}}$. If we now set

$$\begin{aligned}
\bar{\varphi}_j &= \sum_{k=m_{j-1}}^{m_j} \frac{\theta_k(U^n(\tau) - s)}{m_j} && (25) \\
\text{average density} &= \frac{\text{bits set in subinterval}}{\text{number of all bits}}
\end{aligned}$$

we arrive at:

$$\int_{t_0}^{t} \theta_k(U^n(\tau) - s) d\tau = \sum_{j=1}^{l} \bar{\varphi}_j \cdot \Delta_j t + \theta_m(U^n(\tau) - s) \cdot (t - t_m). \quad (26)$$

We do now let $\Delta t \longrightarrow 0$ (this is the point where the model changes, we can interpret it as an A/DM with infinite clock rate, $\theta_k$ becomes $\theta$ the 'real' Heaviside function and $\bar{\varphi}_i$ becomes a measure) and get:

$$\int_{t_0}^{t} \theta(U^n(\tau) - s) d\tau = \sum_{j=1}^{l} \bar{\varphi}_j \cdot \Delta_j t \quad (27)$$

(the second term on the right hand side vanishes but the sum expression is still valid). If we let now on the right hand side $\Delta_j t \longrightarrow 0$ we finally find:

$$\int_{t_0}^{t} \theta(U^n(\tau) - s) d\tau = \int_{t_0}^{t} \varphi(\tau) d\tau, \quad (28)$$

the avarage density becomes the 'real' density and we have the equation that at least suggests that (22) holds. *Remark: That is not clean mathematics but it points out the idea and I'm not considering cleaning up this mess anyhow.* We simply hope that the approximation (22) is not too bad and replace this into (19) to obtain:

$$U^i(t) = \frac{1}{T_i} \int_{t_0}^{t} \left(U^{i-1}(\tau) - \alpha_i U^r \left(2\varphi(\tau) - 1\right)\right) d\tau + U^i(t_0), \quad i = 1, 2, \ldots, n \quad (29)$$

Now we are going to do the interesting step: We derive a differential equation for $\varphi$ and look what this gives us. At first we try to figure out how our first order A/DM works and then what an infinitely fast A/DM does. This will turn out as a key point in our thoughts.

Let us try to find out the behaviour of $U^1(t)$ in (16). For simplicity we set $s = 0$ and $U^1(t_0) = 0$ and let the upper bound of the integral be an arbitrary grid point $t = t_k$. As you will see this does not influence our considerations. Because $U^1$ appears in the argument of the $\theta_k$ function we should make the following case distinction:

$$\begin{aligned}
U_k^1 &= \frac{1}{T} \int_{t_0}^{t_k} \left(U^0(\tau) - \alpha U^r \left(2\theta_k(U^1(\tau)) - 1\right)\right) d\tau \\
&= U_{k-1}^1 + \frac{1}{T} \int_{t_{k-1}}^{t_k} \left(U^0(\tau) - \alpha U^r \left(2\theta_k(U_{k-1}^1) - 1\right)\right) d\tau \\
&= U_{k-1}^1 + \frac{1}{T} \int_{t_{k-1}}^{t_k} U^0(\tau) d\tau + \begin{cases} -\frac{\alpha U^r \Delta t}{T} & \text{for} \quad U_{k-1}^1 \geq 0 \\ \frac{\alpha U^r \Delta t}{T} & \text{for} \quad U_{k-1}^1 < 0 \end{cases} \quad (30)
\end{aligned}$$

If our modulator shall work at all, a first condition is suggested by the following requirement: If $U^1_{k-1}$ is negative, than $U^1_k$ *must be greater than* $U^1_{k-1}$ *and vice versa.* This leads to the following (case $U^1_{k-1} < 0$):

$$\begin{aligned} 0 < U^1_k - U^1_{k-1} &= \frac{1}{T}\int\limits_{t_{k-1}}^{t_k} U^0(\tau)d\tau + \frac{\alpha U^r \Delta t}{T} \\ -\int\limits_{t_{k-1}}^{t_k} U^0(\tau)d\tau &< \alpha U^r \Delta t \end{aligned} \tag{31}$$

If we state that $|U^0(t)| \leq U^{max}$ for all $t$ than we get the stronger inequality ($U^{min} = -U^{max}$):

$$U^{max} < \alpha U^r. \tag{32}$$

The case $U^1_{k-1} \geq 0$ gives the same result. With (32) we can calculate the maximum jump height $h$, i.e. the maximum difference between two iterations of $U^1$. We chose the case $U^1_{k-1} < 0$, the other one will again give the same result because of symmetry:

$$\begin{aligned} |U^1_k - U^1_{k-1}| &= \max_{t\in[t_{k-1},t_k)} \left| \frac{1}{T}\int\limits_{t_{k-1}}^{t_k} U^0(\tau)d\tau + \frac{\alpha U^r \Delta t}{T} \right| \\ &\leq \frac{U^{max}\Delta t}{T} + \frac{\alpha U^r \Delta t}{T} \\ &\leq h = \frac{2\alpha U^r \Delta t}{T}. \end{aligned} \tag{33}$$

With (32) and (33) we can describe the qualitative behaviour of $U^1$ as follows (see also fig. 7): The modulator starts in an 'initial phase' where $U^1_0 \neq 0$ (if we have $U^1_0 = 0$ then this phase is finished immediately). Because of (32) $U^1$ is driven to zero and eventually crosses the zero axis. At this moment the 'working phase' starts. In this phase $U^1$ jumps around zero but remains within $U^1_k \in (-h, +h)$. This is because if $U^1 > 0$ then it will decrase and if $U^1 < 0$ it will increase (again (32) applies). The only chance to grow absolutely is the jump from one side of zero to the other, but that height is bounded by $h$. That means that the maximum value of $|U^1|$ is bounded by the maximum jump height $h$.

For our infinitely fast A/DM ($\Delta t \longrightarrow 0$) that means that $U^1$ will increase resp. decrase until it touches the zero-axis. From then on it will remain constant because it is bounded by $|U^1| \leq \lim_{\Delta t \longrightarrow 0} \frac{2\alpha U^r \Delta t}{T} = 0$.

If we write down the equation for the nth integrator of system (19):

$$U^n(t) = \frac{1}{T_n}\int\limits_{t_0}^{t} \left(U^{n-1}(\tau) - \alpha_n U^r \left(2\theta_k(U^n(\tau) - s) - 1\right)\right) d\tau + U^n(t_0) \tag{34}$$

we see, that our result holds also for the last integrator in a cascaded system if only $|U^{n-1}(t)| < \alpha_n U^r$ is satisfied (the equations are the same). *This is in general a first working condition for such a cascaded system.*

With these results at hand we can derive a differential equation for $\varphi$. Therefore we rewrite (29) as a system of differential equations:

$$\dot{U^i}(t) = \frac{1}{T_i}\left(U^{i-1}(t) - \alpha_i U^r \left(2\varphi(t) - 1\right)\right), \ i = 1, 2, \ldots, n. \tag{35}$$

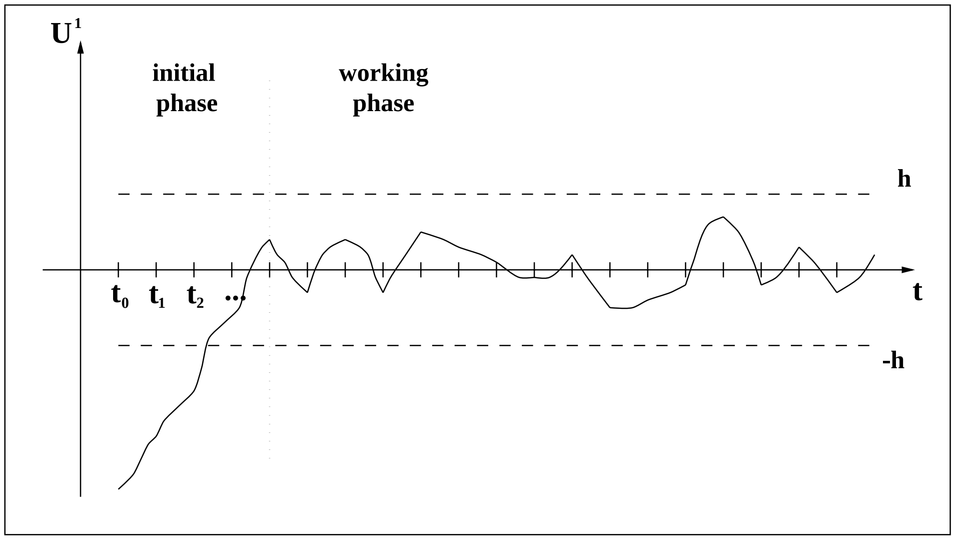


Figure 7: A sketch of the qualitative behaviour of $U^1$ of the A/DM1. The initial phase is finished after $U^1$ crosses the zero axis the first time. From then on the function remains bounded by $h = \frac{2\alpha U^r \Delta t}{T}$. Note that this function may not be (classically) differentiable at $t_k,\ k = k_0, k_1, \ldots$ where $k_0$ marks the end of the initial phase.

We replace $U^n$ with the limit function $U^n(t) \equiv$ constant what gives $\dot{U}^n(t) \equiv 0$. This is another crucial point in this derivation process because the first derivative of $U^1$ becomes during the limit process described just before not even bounded to zero where it is classically differentiable. On the other hand the limit function itself is differentiable and gives $\dot{U}^1(t) \equiv 0$. I just didn't think about how to resolve this, so you may earn the fame. Lastly we write down our system (for the working phase, in the initial phase $\varphi$ is either 1 or 0):

$$\dot{U}^i(t) = \frac{1}{T_i}\left(U^{i-1}(t) - \alpha_i U^r\left(2\varphi(t) - 1\right)\right),\ \ i = 1, 2, \ldots, n-1 \tag{36}$$

$$0 = \frac{1}{T_n}\left(U^{n-1}(t) - \alpha_n U^r\left(2\varphi(t) - 1\right)\right). \tag{37}$$

You get the equations for $\varphi$ by inserting one equation of (36) after another into (37). For the A/D modulators of order 1,2 resp. 3 you will find the following equations:

A/DM1:

$$\varphi = \frac{1}{2\alpha}\left(\frac{U^0}{U^r} + \alpha\right) \tag{38}$$

A/DM2:

$$\dot{\varphi} + \frac{\alpha}{\beta T_1}\varphi = \frac{1}{2\beta T_1}\left(\frac{U^0}{U^r} + \alpha\right) \tag{39}$$

A/DM3:

$$\ddot{\varphi} + \frac{\beta}{\gamma T_2}\dot{\varphi} + \frac{\alpha}{\gamma T_1 T_2}\varphi = \frac{1}{2\gamma T_1 T_2}\left(\frac{U^0}{U^r} + \alpha\right) \tag{40}$$

with $\beta = \alpha_2$ and $\gamma = \alpha_3$.

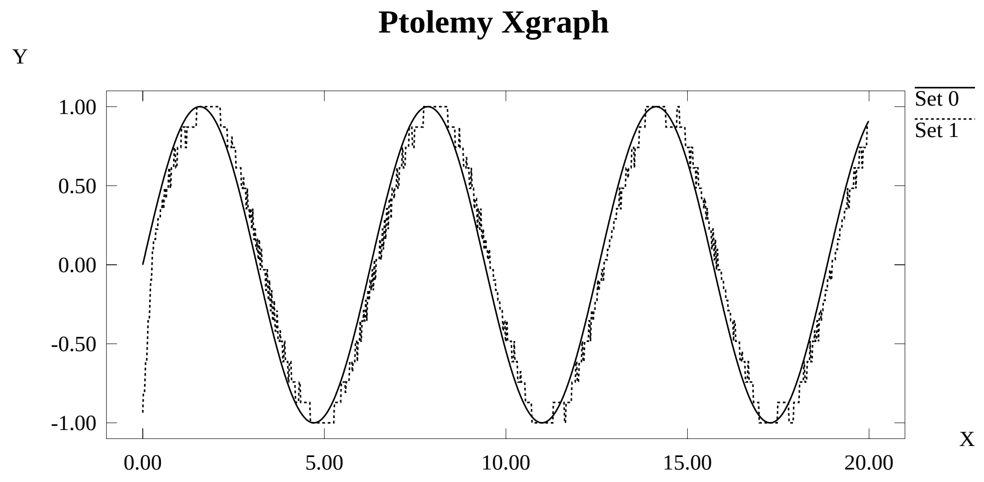


Figure 8: The output of the simulation of the A/D converter of first order with $\alpha = T = U^r = 1$, $\Delta t = 0.01$, 5 bit filtering and $U^{in}(t) = \sin(t)$. You can easily verify that (38) results in $2\varphi - 1 = U^{in}$. The graph shows the well matching of $\sin(t)$ and the simulation output.

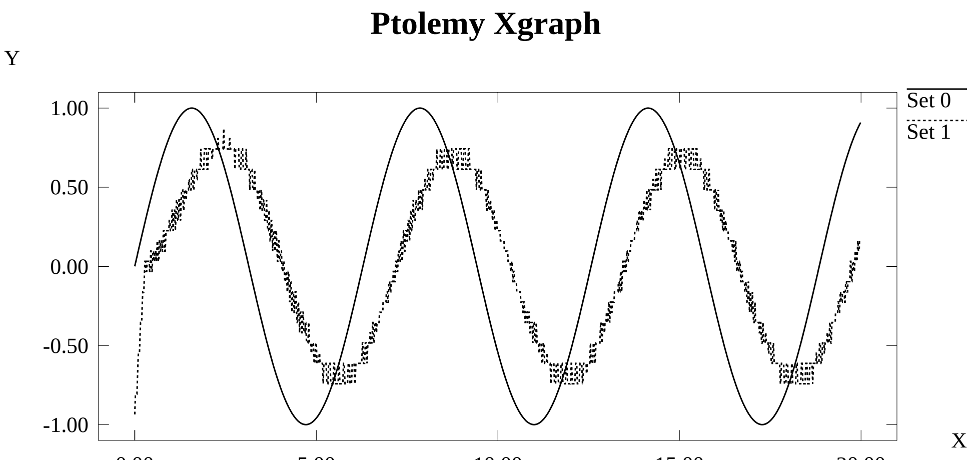
The first test to look if that makes any sense is to compare numerical results of (38) - (40) with results of our simulation (which was fortunately part of this paper). Note that the filtering luckily gives a value $Y_k = \bar{\varphi}_k$, an average value of $\varphi$. So we can directly compare the output of our A/D converters with the result of the numerical integration of (38) - (40). Unfortunately, we use here a left average so the plot of $\bar{\varphi}$ appears right shifted by $2^{b-1}\Delta t$ compared to $\varphi$. To circumvent this one should make sure that $2^b \cdot \Delta t$, where $b$ is the bit dephth, is small enough. On the other hand you may also adjust the output as a central avarage what gives usually better plots.

For the numerical integration I choose the package `ivps` which you can download from *http://www.mathematik.tu-ilmenau.de/~fschild/projects.html*. There is only a german description [7] available yet. The `ivps` scripts and some `gnuplot` scripts for (39) and (40) ((38) is no diff. eqn. at all) are contained in the directory `ivps` of this package, they are mostly self-describing. You can start the numerical integration by typing `ivps filename`.

The figures 8 to 10 show a graphical comparison of the results with $\Delta t = 0.01$ and 5 bit filtering. It is surprising how well the results fit for this mathematically 'large' $\Delta t$. That suggests that in fact the models are closely related to each other and further investigations do make sense.

Some suggested fields of further interest:

- For the mathematicians:
  - First of all, clean up the mathematical derivation process using suitable Sobolev spaces.
  - Try to apply disturbation theory, interpreting $\Delta t > 0$ (the real system) as a disturbation of system (36)-(37).
  - Find out how the equations work on Fourier series as signal input functions.
  - Do some stability investigations.
- For the technicians: Simply assume equations (36)-(37) reflect the qualitative and approximate the quantitative behaviour of the A/D modulators sufficiently (Note: In practical applications $\Delta t$ is very small.). Then you can:
  - Solve the equations (36)-(37). This is simple because they are linear. The process of varying the constants may be voluminous. Try Fourier series as input for simplification.



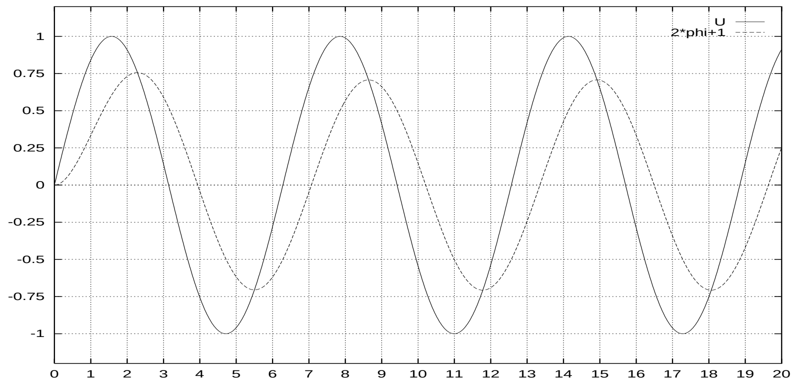
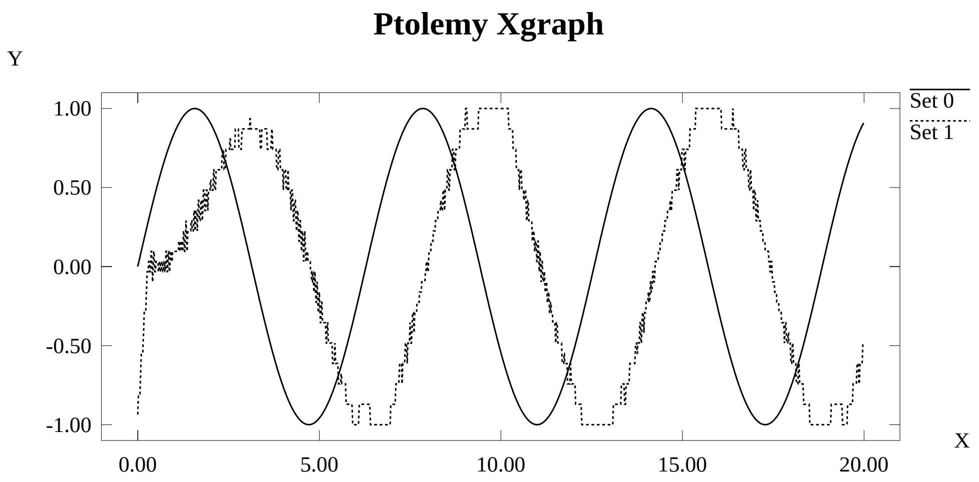


Figure 9: The output of the simulation of the A/D converter of second order with $\alpha = \beta = T_1 = T_2 = U^r = 1$, $\Delta t = 0.01$, 5 bit filtering and $U^{in}(t) = \sin(t)$ (top) and the numerical integration of (39) (see `ivps/ad2.ivp`). The graphic shows the matching of the two solutions.



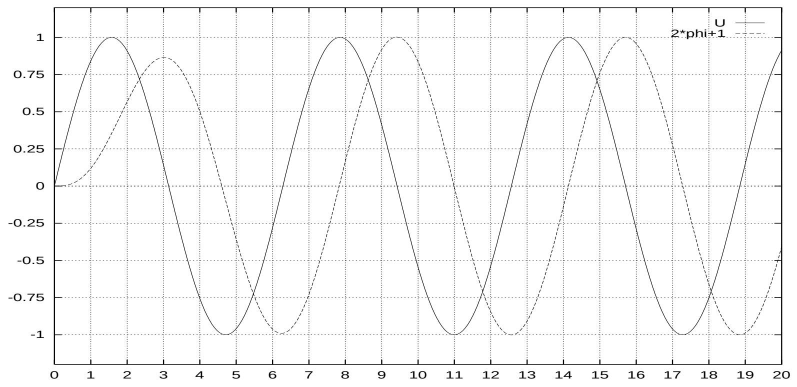


Figure 10: The output of the simulation of the A/D converter of third order with $\alpha = \beta = \gamma = T_1 = T_2 = T_3 = U^r = 1$, $\Delta t = 0.01$, 5 bit filtering and $U^{in}(t) = \sin(t)$ (top) and the numerical integration of (40) (see `ivps/ad3.ivp`). The graphic shows again the matching of the two solutions.

– Look for solutions where (32) is fullfilled and where the homogenuous solution vanishes with maximum speed.

– Figure out the resonance frequencies and try to avoid them.

– Make sure you can reconstruct the signal form $\varphi$, i.e. you should only need to rescale and shift $\varphi$.

– Now you should have a stable system that is not self exciting. In addition you will still have plenty of free parameters for further adjustments (there are arbitrary many good A/D modulators of a given order).

## References

[1] A. Hoffmann, B. Marx: Error estimations of a sigma–delta converter. IEEE Transaction on circuits and Systems-II: Analog and Digital Signal Processing (submitted 12/1997)

[2] A. Hoffmann, B. Marx: Mathematical model of a sigma–delta modulator. Journal of Difference Equations 17 (1998)

[3] B. Metzger, Th. Neubert: Mathematical modelling of a second order sigma–delta modulator by difference equations, Workshop des Informatik–Graduiertenkollegs im Rahmen der Jahrestagung der Gesellschaft f"ur Informatik an der RWTH Aachen vom 22. - 23. Sep. 1997, 169 - 181 (englisch)

[4] R. E. Mickens: Difference Equations, Theory and Applications, Van Nostred Reinhold, New York (1990)

[5] Micro Almagest, Ptolemy 0.6 Instructional DSP Manual, University of California at Berkeley (JAN/20/97), (http://ptolemy.eecs.berkeley.edu/)

[6] The Almagest, Vol. 2 – Ptolemy 0.7 Programmer's Manual, University of California at Berkeley (AUG/27/97), (http://ptolemy.eecs.berkeley.edu/)

[7] F. Schilder: Ivps Dokumentation, TU-Ilmenau (1997), (http://www.mathematik.tu-ilmenau.de/ fschild)