HAL
open science

IIR digital filtering of non-uniformly sampled signals via state representation

Laurent Fesquet, Brigitte Bidégaray-Fesquet

To cite this version:

Laurent Fesquet, Brigitte Bidégaray-Fesquet. IIR digital filtering of non-uniformly sampled signals via state representation. Signal Processing, 2010, 90 (10), pp.2811-2821. 10.1016/j.sigpro.2010.03.030 . hal-00493354

HAL Id: hal-00493354

https://hal.science/hal-00493354v1

Submitted on 9 Jun 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

IIR Digital Filtering of Non-uniformly Sampled Signals *via* State Representation

L. Fesquet*, B. Bidégaray-Fesquet†

Authors's version of Signal Processing **90** (2010) 2811–2821
doi:10.1016/j.sigpro.2010.03.030

Abstract

This article describes a new kind of processing chain based on a non-uniform sampling scheme provided by a level-crossing ADC. The chain implements IIR filters which process directly the non-uniform samples without resampling in a regular scheme. The non-uniformity in the sample times leads to choose a state representation for the filters. The stability is studied and the performances of various numerical schemes used to implement the filters in this representation are compared.

Keywords: Non-uniform sampling, IIR filter, numerical schemes for ODEs, stability.

1 Introduction

In many applications such as electronic embedded systems, there is a need for low consuming devices. One way to achieve this goal is to use asynchronous circuits, in which no clock rules the functioning. In these devices, components do not wait for clock signals, instead they each treat information as fast as they can and use specific protocols to communicate with other components. The consequence (as the components work asynchronously) is that the slowest component does not determine anymore the performances of the entire system. Moreover, the asynchronous circuits are event-driven which means they consume energy only if they have data to process.

The classical sampling scheme which consists in taking signal samples at regular clock times has no interest and meaning any more with these asynchronous systems. In order to be compliant with these asynchronous circuits, the signals can be sampled in a dual way. In classical synchronous systems, sampling

*L. Fesquet: TIMA (CNRS, Grenoble INP, Université Joseph Fourier), 46 avenue Felix-Viallet, 38031 Grenoble Cedex, France. `Laurent.Fesquet@imag.fr`

†B. Bidgaray-Fesquet: LJK (CNRS, Université Joseph Fourier, Grenoble INP, Université Pierre Mendès-France), B.P. 53, 38041 Grenoble Cedex 9, France. `Brigitte.Bidegaray@imag.fr`

times are known precisely and the amplitudes are quantized (with an amplitude quantization error). In asynchronous systems thresholds are predefined within the amplitude dynamical range and known precisely. The time instants are quantized with a local clock (with a time quantization error). This procedure is called the level crossing sampling scheme.

Samples are therefore only taken when the signal has some meaningful variation. This induces a lot of precision in active parts of the signal and no activity when the signal is constant (asynchronous circuits are event-driven), implying low consumption for the targeted applications and also other advantages like low electromagnetic pollution.

There are a lot of applications where it can be useful to have such a sampling. This is the case when the signal activity is significant only on short times compared to the total duration of the signal, such as speech, electro-cardiogram signals, seismic signals, etc.

The design flow has to be completely redefined for these systems including signal processing tools which are our concern in this paper. Indeed, the general goal is not only to treat level crossing sampled signals but to treat them using asynchronous systems. We address here only IIR filtering for non-uniformly sampled signals. We also do not address the implementation with asynchronous chips, which has been done in [2]. Other approaches in processing non-uniform sampled signals may be found e.g. in [10] or [12].

In Section 2 we describe our non-uniform data, the IIR filters in the state representation and the standard numerical schemes in the literature (Euler, bilinear, integral). In Section 3 we point out stability as a criterion to choose a proper scheme and give a design flow which takes it into account, both for unconditionally and conditionally stable schemes. In Section 4 we give other possible schemes and compare them for three types of low-pass filters (Butterworth, elliptic and Chebyshev) in terms of stability, efficiency and algorithmic complexity.

2 State of art

2.1 Definitions and notations

The output of a level-crossing ADC consists in a sequence of couples (a_n, dt_n) (see Fig. 1). The amplitude a_n of the signal is captured each time the signal crosses some predefined quantization levels. The time delay elapsed since the last sample dt_n is computed by a local timer with precision T_C. From an initial time t_0, we may reconstruct the time t_n of the n th input sample using the recurrence relation $t_n = t_{n-1} + dt_n$. This value is used to describe our approach but not in the algorithms since it is not available in practice. We will also define the half-time between samples n and $n+1$ by $t_{n+1/2} = (t_n + t_{n+1})/2$.

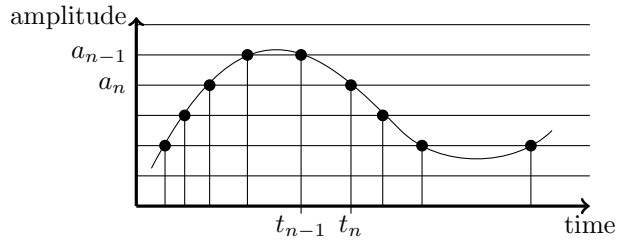


Figure 1: Non-uniform sampling scheme

Such a non-uniform sampling was first introduced in [11]. An asynchronous implementation, called A-ADC (for asynchronous ADC), has been defined in [3] and analyzed in terms of signal-to-noise ratio. In digital signal processing we are interested in filtering an input signal $i(t)$ represented by the samples (i_n, dt_n) to obtain an output signal $o(t)$ represented by the samples (o_n, dt_n). Keeping the same time delays is not convenient for FIR filters, where the output delays depend on the input and the impulse response samples (see [1], [9]). We are interested in IIR filters for which the output is given at some later time than the input due to computational delay, but it can be considered as constant and does not affect time delays. A closer study of this point would necessitate to consider an effective asynchronous implementation of the schemes, which is not our goal here but is discussed in [2].

2.2 State representation for an IIR filter

The aim is not to design asynchronous filters. Instead we use standard filters. Usually a uniformly sampled input signal $I(s)$ is written in the Laplace domain and filtering consists only in multiplying by the filter transfer function

$$H(s) = \frac{\sum_{j=0}^{N} \alpha_j s^j}{\sum_{j=0}^{N} \beta_j s^j}$$

and obtaining the output filtered signal $O(s) = H(s)I(s)$. This is based on efficient Laplace transforms which are not available for non-uniformly sampled signals. Therefore, we use the state representation of the filter where all the signals (input i, output o) are written in the time domain. This necessitates the use of a vector-valued state variable x:

$$\frac{dx(t)}{dt} = Ax(t) + Bi(t), \quad (1)$$

$$o(t) = Cx(t) + Di(t), \quad (2)$$

where

$$A = \begin{pmatrix} 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ -\beta_0 & -\beta_1 & \cdots & -\beta_{N-2} & -\beta_{N-1} \end{pmatrix}$$

is the $N \times N$ state matrix, $B = (0 \cdots 0\ 1)^t$ is the command vector, $C = (\alpha_0 - \alpha_N\beta_0\ \cdots\ \alpha_{N-1} - \alpha_N\beta_{N-1})$ is the observation vector and $D = \alpha_N$ the direct link coefficient.

The integral form for Eq. (1) is given by

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bi(\tau)d\tau. \tag{3}$$

The characteristic polynomial of the state matrix A reads

$$\det(\lambda \mathrm{Id} - A) = \lambda^N + \beta_{N-1}\lambda^{N-1} + \ldots + \beta_1\lambda + \beta_0,$$

(where Id is the $N \times N$ identity matrix) and the poles of the transfer function are exactly the zeros (or eigenvalues) of the state matrix. We can define a linear change of basis P for the state vector in $\mathbb{R}^n$ and replace it by $y(t) = P^{-1}x(t)$ such that $\tilde{A} = P^{-1}AP$ is the Jordan form of matrix A (if all the roots of A are distinct, $\tilde{A}$ is diagonal). If we further define $\tilde{B} = P^{-1}B$, $\tilde{C} = CP$ and $\tilde{D} = D$, the new state variable $y(t)$ is solution to

$$\begin{aligned} \frac{dy(t)}{dt} &= \tilde{A}y(t) + \tilde{B}i(t), \\ o(t) &= \tilde{C}y(t) + \tilde{D}i(t). \end{aligned}$$

In the sequel, we will keep the original system, but this form justifies the stability proof in Section 3.1.5.

2.3 Euler approximation of an IIR filter

The Euler method consists in writing equation Eq. (1) at time t_{n-1} and use a forward approximation for the time derivative, namely

$$\frac{x_n - x_{n-1}}{dt_n} = Ax_{n-1} + Bi_{n-1}, \tag{4}$$

which also reads as

$$x_n = (\mathrm{Id} + dt_nA)x_{n-1} + Bdt_ni_{n-1}.$$

Then the output is simply computed by

$$o_n = Cx_n + Di_n. \tag{5}$$

2.4 Bilinear approximation of an IIR filter

Poulton and Oksman [7, 8] have chosen a bilinear method to approximate the time derivative in Eq. (1). This method consists in writing a centered approximation of the equation at time $t_{n-1/2}$, that is

$$\frac{x_n - x_{n-1}}{dt_n} = A\frac{x_n + x_{n-1}}{2} + B\frac{i_n + i_{n-1}}{2}. \tag{6}$$

The output is once more computed using Eq. (5).

We may give an explicit form for Eq. (6), namely

$$x_n = \Psi_n x_{n-1} + \Lambda_n \frac{1}{2}(i_n + i_{n-1}),$$

where

$$\Psi_n = \left(\mathrm{Id} - \frac{dt_n}{2}A\right)^{-1}\left(\mathrm{Id} + \frac{dt_n}{2}A\right) \text{ and } \Lambda_n = \left(\mathrm{Id} - \frac{dt_n}{2}A\right)^{-1} dt_n B.$$

This algorithm displays several advantages: it is second order, and is much more effective than the Euler method. Nevertheless, it is relatively expensive in computational time since we need to invert a matrix for each new output sample. This is the reason why we will suggest other approximation methods which do not share this drawback. Other criteria for the choice of a "good method" will be given below in our specifications.

2.5 Discretization in the integral form

In [5], Fontaine and Ragot choose to discretize the integral form (3) of the state equation directly. Their only approximation consists in replacing the continuous signal $i(t)$ by a sample-hold or piecewise linear interpolation. For example, for sample-hold interpolation, they compute

$$x_n = e^{Adt_n}x_{n-1} - A^{-1}(\mathrm{Id} - e^{Adt_n})Bi_{n-1}.$$

The stability proof below (see Section 3.1.2) for the continuous variables leads to the stability of such an approximation, and the results presented in [5] are rather good in term of filtering. They suggest to split operators into second- (or first-) order filters in order to have a simpler evaluation of the quantity $\exp(Adt_n)$. We will use this idea to compare the complexity of the different methods in Section 4.3.

3 Stability specifications and design flow

3.1 Stability

3.1.1 Definition

Definition 1 *A filtering process is said to be* stable *if when perturbed by an input signal with finite time duration, the output signal eventually returns to an equilibrium state.*

A well known necessary and sufficient condition for a filter to be stable is that the poles of its transfer function (eigenvalues of A) have a negative real part.

3.1.2 Stability in the state representation

Suppose the input signal is constant from time t_* on. Then, for $t \geq t_*$, the solution to the state equation reads as

$$\begin{aligned} x(t) &= e^{A(t-t_*)}x(t_*) + \int_{t_*}^{t} e^{A(t-\tau)}Bi(\tau)d\tau \qquad (7) \\ &= e^{A(t-t_*)}x(t_*) + A^{-1}(e^{A(t-t_*)} - \mathrm{Id})Bi(t_*), \\ o(t) &= Cx(t) + Di(t_*), \qquad (8) \end{aligned}$$

and since the eigenvalues of A are supposed to have a negative real part

$$\lim_{t\to+\infty} o(t) = (D - CA^{-1}B)i(t_*) = (\alpha_N + \frac{\alpha_0 - \alpha_N\beta_0}{\beta_0})i(t_*) = \frac{\alpha_0}{\beta_0}i(t_*).$$

In particular, the limit does not depend on the state of the system when perturbed but only on the constant value of the input. Thus we obtain the classical result that if the eigenvalues of A have a negative real part then the filtering process is stable in the sense of Definition 1.

3.1.3 Sampled state equation

We have already seen two types of approximations for an asynchronous signal: the Euler and the bilinear methods. The problem we deal with is the approximation of an ordinary differential equation sampled on a non uniform time discretization. This problem is classical in numerical analysis. Both the Euler and the bilinear methods can be cast in a more general framework of one-step schemes which reads as

$$x_n = \Phi_n x_{n-1} + \Gamma_n \hat{i}_n. \qquad (9)$$

(This framework could be easily enlarged to also encompass multi-step schemes, e.g. Taylor approximations). The input signal is approximated by $\hat{i}_n$. To illustrate notation Eq. 9, the Euler method corresponds to

$$\Phi_n = \mathrm{Id} + dt_n A, \qquad \Gamma_n = dt_n B, \text{ and } \hat{i}_n = i_{n-1}$$

and the bilinear method to

$$\begin{aligned} \Phi_n &= \left(\mathrm{Id} - \frac{dt_n}{2}A\right)^{-1}\left(\mathrm{Id} + \frac{dt_n}{2}A\right), \\ \Gamma_n &= \left(\mathrm{Id} - \frac{dt_n}{2}A\right)^{-1} dt_n B, \text{ and } \hat{i}_n = \frac{1}{2}(i_n + i_{n-1}). \end{aligned}$$

In Section 3.1.2 we have seen that in the stability is ensured if the eigenvalues of A have a negative real part, i.e. if the eigenvalues of $\exp(A(t-t_*))$ lie inside

the unit disk. We prove below that, if the eigenvalues of Φ_n lie uniformly in the unit disk then the stability of the approximation is ensured. The stability condition only involves Φ_n (and not $\Gamma_n \hat{\imath}_n$).

3.1.4 "Good method" with respect to stability

For any method that we will define, Φ_n preserves in practice the eigendirections of matrix A and an eigenvalue λ of A corresponds to an eigenvalue μ_n of Φ_n *via* the transform $\mu_n = \mathcal{T}_n(\lambda)$.

Examples For the Euler method, $\mathcal{T}_n(\lambda) = 1 + dt_n\lambda$. The eigenvalue λ lies in the left half of the complex plane ($\mathrm{Re}(\lambda) < 0$) if and only if μ_n lies in the region ($\mathrm{Re}(\mu_n) < 1$). This includes the unit disk and therefore the inverse image of the unit disk is a subset of the left half-plane (LHP).

For the bilinear method,

$$\mathcal{T}_n(\lambda) = \frac{1 + dt_n\lambda/2}{1 - dt_n\lambda/2}.$$

This is the well known homographic function, which maps the left half-plane onto the unit disk (see Fig. 2).

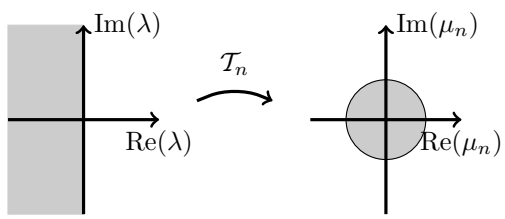


Figure 2: Action of the $\mathcal{T}_n$ transform for the bilinear transformation. Left eigenvalues λ of matrix A, right eigenvalues μ_n of matrix Φ_n shown on the complex plane

Discussion The choice of a good method may also be done following two types of questioning.

1. We may want a method which will give good results for any stable filter.
2. We may select filters which eigenvalues in some restricted region of the complex plane such that the eigenvalues for the approximate method lie inside the unit disk.

The bilinear method is good in both respects. The Euler method should be rejected if the first point of view is adopted. Otherwise, the filter should be chosen such that $|1 + dt_n\lambda| < 1$. This is a disk of radius $1/dt_n$ which is included in the left half-plane (see Fig. 3). If the filter is given, and has no eigenvalues on the imaginary axis, this implies an upper bound on the sampling time.

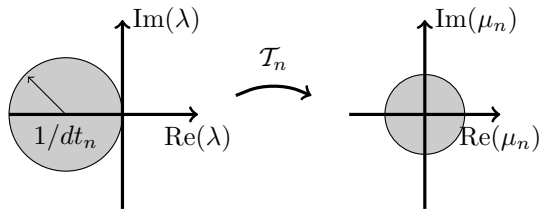


Figure 3: Action of the $\mathcal{T}_n$ transform for the Euler scheme. Left eigenvalues λ of matrix A, right eigenvalues μ_n of matrix Φ_n shown on the complex plane

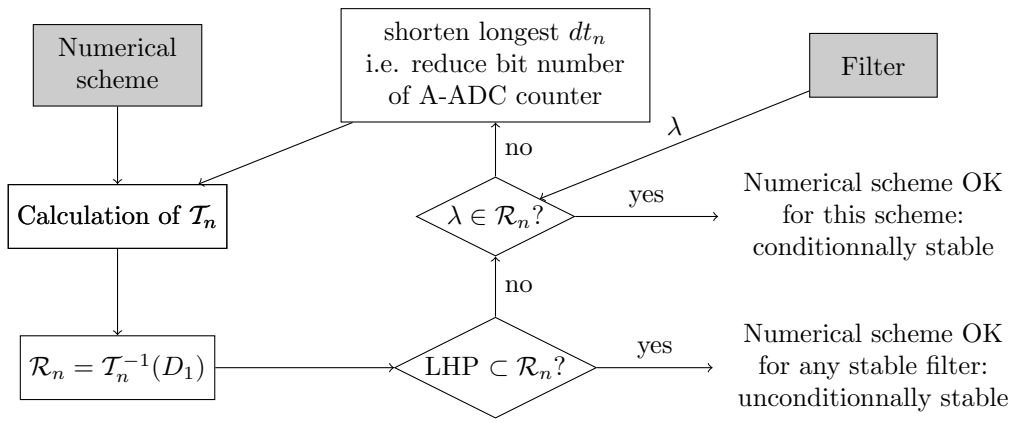


Figure 4: Design flow to implement stable numerical filters

3.1.5 Proof of the stability of the approximation

In both cases, Φ_n is an approximate value of $\exp(dt_n A)$. The stability of the approximation will be proved following the proof for the continuous equation. A simple recurrence from Eq. 9 implies that

$$x_n = \left(\prod_{j=1}^{n} \Phi_j\right) x_0 + \sum_{j=1}^{n} \left(\prod_{k=j+1}^{n} \Phi_k\right) \Gamma_j \hat{i}_j.$$

Let us first prove stability in the simple case when the time step is constant. In this case, $\Phi_j = \Phi$ and $\Gamma_j = \Gamma$ for all j. Besides we assume that $\hat{i}_j$ is constant and equal to i_* from index n_0 onwards. Therefore

$$x_n = \Phi^n x_0 + \sum_{j=1}^{n} \Phi^{n-j} \Gamma \hat{i}_j$$

$$= \quad \Phi^n x_0 + \Phi^{n-n_0+1}\left(\sum_{j=1}^{n_0-1} \Phi^{n_0-j-1}\Gamma\hat{i}_j\right) + \sum_{j=0}^{n-n_0} \Phi^j \Gamma i_*.$$

Since the eigenvalues of Φ are supposed to lie inside the unit circle the first two terms vanish as $n \to \infty$. Thus

$$\lim_{n\to\infty} x_n = \lim_{n\to\infty} \sum_{j=0}^{n-n_0} \Phi^j \Gamma i_* = (\mathrm{Id} - \Phi)^{-1}\Gamma i_*.$$

In the case of non-constant time steps, the fact that the eigenvalues of all matrices lie in the unit circle does not prove that products like $\prod_{j=1}^{n} \Phi_j$ vanish (counter examples are easy to find). However this is true for triangular matrices and therefore for the Jordan form given in Section 2.2. The proof can be made with this formulation and the result is still valid in the initial variables.

Besides our definition of stability and the principle of the A-ADC imply that if the input is constant, no new output is computed and the output is therefore constant. This way of thinking is at least valid for absolute stable methods. We have seen that we may have to define an upper bound T_U for the sampling time to ensure stability. Then we have to add new samples with the same amplitude and delay time T_U. In this case, the stability proof is the same as in the synchronous case.

3.2 Design flow

The preceding discussion leads to define a design flow to implement a stable numerical filter. This design flow is shown in Fig. 4. There are two inputs: a numerical scheme and a filter. For a given value of dt_n, a transform $\mathcal{T}_n$ is calculated from the numerical scheme. Then we define the stability domain as the region $\mathcal{R}_n$ of the complex plane which is the inverse image of the unit disk D_1:

$$\mathcal{R}_n = \mathcal{T}_n^{-1}(D_1).$$

If the left half plane is included in $\mathcal{R}_n$ then the numerical scheme yields a stable method whatever the filter is. We will call such a scheme an unconditionally stable scheme. Otherwise, if the filter poles λ (eigenvalues of matrix A) belong to $\mathcal{R}_n$ then the numerical scheme yields a stable method for *this* specific filter. If this condition is not fulfilled then a possible solution is to reduce the maximal value of dt_n. A generic situation is indeed that of the Runge–Kutta schemes (see below Section 4.1.1).

The region $\mathcal{R}_n$ is plotted in Figs 5 and 6 for the RK4 and RK23 schemes respectively. RK23 is an unconditionally stable scheme with a "funny" (i.e. non convex) stability region. RK4 is a conditionally stable scheme. The regions $\mathcal{R}_n$ do not overlap for different values of dt_n. However the particular form of these regions allow to include any given set of points from the left half plane by taking a small enough value of dt_n.

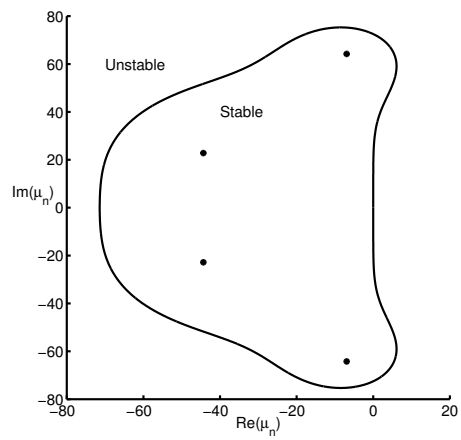


Figure 5: Stability domain $\mathcal{R}_n$ for the RK4 scheme and $dt_n = 0.0388$ s. Stars are $\mathcal{T}_n^{-1}(\{0\})$ and lie in the stable domain

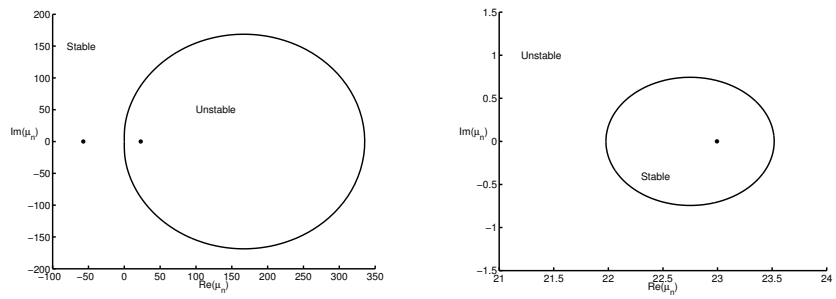
4 Numerical schemes

4.1 Explicit and implicit schemes

The numerical schemes are used to discretize equation (1). The left-hand side is always discretized as $(x_n - x_{n-1})/dt_n$. If the right-hand side is given in terms of the state and the entries at times before t_n, i.e. t_{n-1}, t_{n-2}, ... then the scheme is said to be explicit. We have already presented the Euler method which is an explicit scheme. If time t_n is used the scheme is said to be semi-implicit. The bilinear method is an example of such a method. If later times, like t_{n+1}, are used, the scheme is said to be implicit. Such schemes are not used for filtering methods since they use future entries (which would be possible by insertion of delays) but also future states of the system. In addition it would be prohibited by the large amount of calculations needed.

4.1.1 Explicit schemes are not unconditionally stable

By no means can an explicit method be unconditionally stable. Indeed, for an explicit method, $\mathcal{T}_n$ is a polynomial and no polynomial can map the left half-plane in any bounded domain of the complex plane. This is a major drawback, but we want to consider explicit methods for their costless implementation. Therefore, for explicit methods we will be interested in finding the filtering methods which will lead to the less restrictive condition on time steps.



(a) Full view (b) Zoom

Figure 6: Stability domain $\mathcal{R}_n$ for the RK23 scheme and $dt_n = 0.0388$ s. Stars are $\mathcal{T}_n^{-1}(\{0\})$ and lie in the stable domain. The zoom (b) explains why one star seems to be in the unstable domain in the full view (a): the stable domain is the union of the outer domain of the large boundary (which includes the left half plane) and the inner domain of the small one

4.1.2 The Runge–Kutta 4 scheme

Many explicit schemes exist in the numerical analysis literature [6], among them explicit Runge–Kutta schemes. We have used the 4th order Runge–Kutta scheme (RK4) which for our particular state equation and a linear approximation of $i(dt_{n-\frac{1}{2}})$ reads as $x_n = \Phi_n x_{n-1} + \Gamma_n \hat{i}_n$, where

$$\begin{aligned}
\Phi_n &= \mathrm{Id} + dt_n A + \frac{dt_n^2}{2}A^2 + \frac{dt_n^3}{6}A^3 + \frac{dt_n^4}{24}A^4 \\
\Gamma_n \hat{i}_n &= dt_n[\frac{\mathrm{Id}}{2} + \frac{dt_n}{3}A + \frac{dt_n^2}{8}A^2 + \frac{dt_n^3}{24}A^3]Bi_{n-1} \\
&\quad + dt_n[\frac{\mathrm{Id}}{2} + \frac{dt_n}{6}A + \frac{dt_n^2}{24}A^2]Bi_n.
\end{aligned}$$

We may notice that the iteration matrix that operates on y_{n-1} is the 4th order Taylor expansion of $\exp(dt_n A)$ which is the exact value. For this scheme, the inverse image of the unit circle for

$$\mathcal{T}_n(\lambda) = 1 + dt_n\lambda + \frac{dt_n^2}{2}\lambda^2 + \frac{dt_n^3}{6}\lambda^3 + \frac{dt_n^4}{24}\lambda^4$$

is shown in Fig 5.

4.1.3 Semi-implicit schemes

The bilinear method is a semi-implicit scheme. We have also tested a third order two-stage Runge–Kutta semi-implicit method (RK23). For our specific state equation, it reads as $x_n = \Phi_n x_{n-1} + \Gamma_n \hat{i}_n$, where

$$\Phi_n = [\mathrm{Id} - dt_n A]^{-1}[\mathrm{Id} - 2\frac{dt_n}{3}A]^{-1}[\mathrm{Id} - 2\frac{dt_n}{3}A - \frac{dt_n^2}{2}A^2],$$

$$\Gamma_n \hat{i}_n = [\mathrm{Id} - dt_n A]^{-1}[\mathrm{Id} - 2\frac{dt_n}{3}A]^{-1} \times$$
$$\times dt_n([\frac{\mathrm{Id}}{2} - \frac{dt_n}{2}A]Bi_{n-1} + [\frac{\mathrm{Id}}{2} - 2\frac{dt_n}{3}A]Bi_n).$$

For this scheme, the inverse image of the unit circle for

$$\mathcal{T}_n(\lambda) = \frac{1 - 2\frac{dt_n}{3}\lambda - \frac{dt_n^2}{2}\lambda^2}{(1 - dt_n\lambda)(1 - 2\frac{dt_n}{3}\lambda)}$$

is shown in Fig 6 and is twofold as already noticed.

A very well known semi-implicit scheme is also the retrograde Euler scheme, which is in some sense "too stable" as will be seen in the simulations below. It reads as

$$\frac{x_n - x_{n-1}}{dt_n} = Ax_n + Bi_n, \tag{10}$$

which should be compared with Eqs (4) or (6). For this scheme

$$\Phi_n = [\mathrm{Id} - dt_n A]^{-1},\ \Gamma_n = [\mathrm{Id} - dt_n A]^{-1} dt_n B,\ \text{and}\ \hat{i}_n = i_n$$

and therefore

$$\mathcal{T}_n(\lambda) = \frac{1}{1 - dt_n\lambda}.$$

This scheme is unconditionally stable, and the stability domain is the exterior of the circle of center $(1/dt_n, 0)$ and of radius $1/dt_n$, which contains the left half-plane.

4.2 Comparison of the schemes for three low-pass filters

We address the filtering of the superposition of DC, 1 and 4 Hz signals:

$$i(t) = 0.45\sin(2\pi t) + 0.45\sin(4 \times 2\pi t) + 0.9$$

at a 2 Hz cut-off frequency. We first apply an A-ADC converter with dynamic range [0V,1.8V] on this signal to obtain 641 samples displayed in Fig. 7.

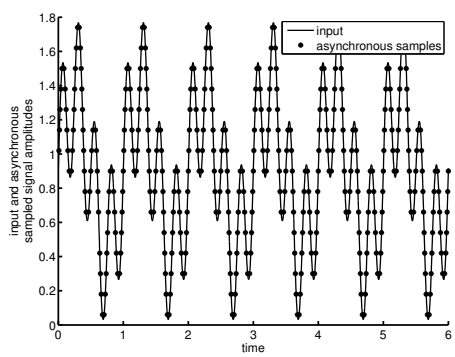


Figure 7: Asynchronous signal used for scheme comparisons

The filter order is 10 for the three filters. Each time five schemes are used, and compared on Figs 8 to 13, namely Euler, retrograde Euler, bilinear, RK23 and RK4 schemes. As we have seen, three of them are unconditionally stable.

To be able to discriminate between filter characteristics and sampling or numerical effects, we give in Table 1 the amplification coefficients for the frequencies of our test input signal and the cut-off frequency. The values will have to be compared with those in Tables 2–4.

	0Hz	1Hz	2Hz	4Hz
Butterworth	1.0000	1.0000	0.7071	0.0010
elliptic	0.9441	0.9996	0.9441	0.0086
Chebychev	0.7943	0.9340	0.7943	0.0000

Table 1: Filter amplifications for the input frequencies 0Hz, 1Hz and 4Hz, and the cut-off frequency 2Hz.

Horizontal lines in Figs. 8, 10 and 12 display the theoretical lower and upper values for the output due to the characteristics of the filter only.

4.2.1 Butterworth filter

We first try a Butterworth filter. The filtering results are displayed in Fig. 8 and show three very comparable and good results corresponding to the bilinear, RK4 and RK23 schemes. The retrograde Euler scheme is of course stable but damps the solution too much. Some amplification is observed for the Euler scheme.

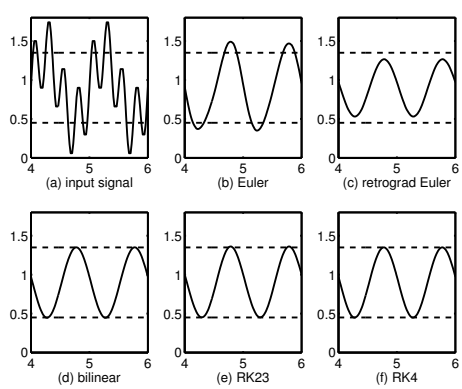


Figure 8: Filtering with five schemes for the Butterworth filter.

This is accounted for in Fig. 9 on which eigenvalues $\mu_n = \mathcal{T}_n(\lambda)$ are plotted for the five schemes. The Euler scheme has two eigenvalues outside the unit

disk. They lie not too far from the unit circle which explains that the solution is still reasonable but a longer simulation would make this solution blow up.

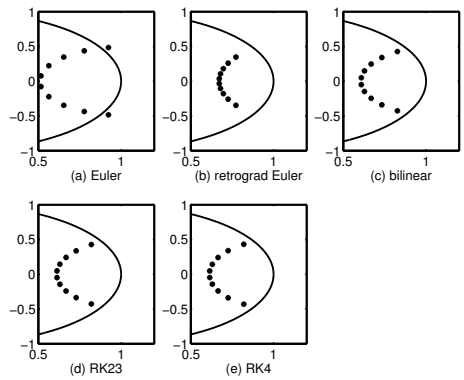


Figure 9: Eigenvalues of the five schemes for the Butterworth filter and the maximal time step $dt_{max} = 0.0388$ s.

In Table 2 are displayed the mean value and the half amplitude which are supposed to reflect the DC and the 1 Hz components of the filtered signal. In order not to be perturbed by the transient values and more generally by the initial and final times used to compute the mean, we consider it is the half value between the minimum and the maximum of the filtered signal. The results are given for all schemes except the Euler schemes which is clearly unstable. The comparison is made with the theoretical value which is the value displayed in Table 1 multiplied by the input amplitude of the DC and the 1 Hz components, namely 0.9 and 0.45, respectively.

	theoretical	retro Euler	bilinear	RK23	RK4
mean	0.9000	0.8986	0.9044	0.9116	0.9054
half amplitude	0.4500	0.3686	0.4578	0.4604	0.4568

Table 2: Mean and half amplitude of the Butterworth filtered signals for all the schemes compared to the theoretical values at DC and 1 Hz frequency.

There is a good agreement with the expected theoretical values, especially for the mean value. The 1 Hz component is as already noticed too much damped by the retrograde Euler scheme.

4.2.2 Elliptic filter

The same computations are performed for an elliptic filter displaying comparable results for the bilinear, RK4 and RK23 schemes (see Fig. 10). The retrograde

Euler scheme yields better results than for the Butterworth filter. The Euler scheme is very bad in this situation. Most samples for this scheme are not displayed in Fig. 10 because they are out of the dynamic range.

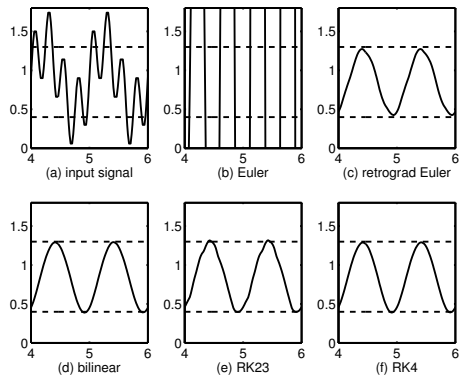


Figure 10: Filtering with five schemes for the elliptic filter.

This is coherent with the eigenvalue computations plotted in Fig. 11: only two eigenvalues lie on the unit disk for the Euler scheme and the others are far from the unit circle.

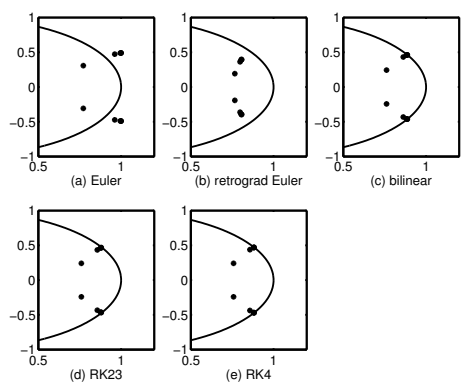


Figure 11: Eigenvalues of the five schemes for the elliptic filter and the maximal time step $dt_{max} = 0.0388$ s.

Table 3 is the equivalent of Table 2 for the elliptic filter. It would not be fair to compare the filtered amplitudes with the ideal values (0.9 for the DC component and 0.45 for the 1 Hz component) since the performance of the elliptic filter is far from that at least for the DC component.

	theoretical	retro Euler	bilinear	RK23	RK4
mean	0.8497	0.8487	0.8412	0.8540	0.8404
half amplitude	0.4498	0.4249	0.4585	0.4676	0.4547

Table 3: Mean and half amplitude of the elliptic filtered signals for all the schemes compared to the theoretical values at DC and 1 Hz frequency.

4.2.3 Chebyshev filter

The same computations are performed for a Chebyshev filter and shown in Fig. 12. Once again, the bilinear, RK4 and RK23 schemes yield good results, the retrograde Euler scheme damps the solution two much and Euler is out of range as the eigenvalue analysis displayed in Fig. 13 explains (only four eigenvalues on the unit disk).

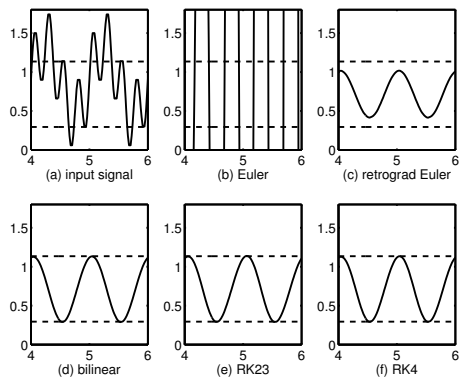


Figure 12: Filtering with five schemes for the Chebyshev filter.

As can be seen in Table 4 the performance of the Chebyshev filter is poor especially for the DC component. However, the applied schemes do not alter the performance more.

	theoretical	retro Euler	bilinear	RK23	RK4
mean	0.7149	0.7401	0.7124	0.7154	0.7166
half amplitude	0.4203	0.3349	0.4546	0.4559	0.4557

Table 4: Mean and half amplitude of the Chebyshev filtered signals for all the schemes compared to the theoretical values at DC and 1 Hz frequency.

The numerical results with the different schemes are comparable to that obtained with a uniform sampling. The main difference is that the value of longest dt_n has to be checked in order to ensure the filter stability in the case of conditionally stable schemes.

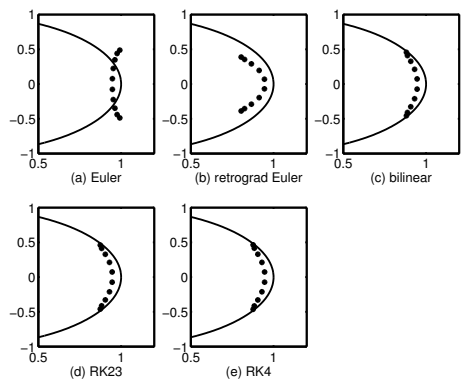


Figure 13: Eigenvalues of the five schemes for the Chebyshev filter and the maximal time step $dt_{max} = 0.0388$ s.

After the filtering process we obtain a signal which is not a level crossing sampled signal. In particular the low-pass filtered signals are clearly over-sampled, with respect to the Nyquist criterion (which is always the case for level crossing samples in active parts of the signal) but also compared to a level crossing sampled signal. A new sampling block should therefore be added behind the filtering block.

4.3 Complexity analysis

We drop now the study of both Euler schemes which have been proven to be less efficient, and compare the other schemes with respect to their complexity, which will be our last argument to choose a good scheme.

4.3.1 Reduction to one- or two-order filters

In order to implement an N th order filter, it is usual to decompose it in multiple first and second order filters which are easily implemented with classical electrical structures like Rauch or Sallen-key structures. In our case, such a reduction is also attractive because it is much easier to invert scalars or 2×2 matrices than an $N \times N$ matrix. Moreover, the decomposition provides the ability to make explicit the calculus of the matrix inversion which can be a real improvement in terms of computation speed and complexity.

4.3.2 Algorithmic complexity

In order to evaluate the implementation cost, the number of operations and the memory needs have been estimated for an N-order filter decomposed into one- and two-order filters for four schemes (bilinear, Runge–Kutta 4, Runge–Kutta 23 and the integral form). To obtain these values, storage has been

favored *vs.* computation. Other choices can be made but this is connected to implementation issues which are out of the scope of this article. For the semi-implicit schemes, the values in Tables 6 and 5 refer to the worst case, that is when there is at most one ($N[2]$, i.e. N modulo 2) one-order filter and $\lfloor N/2 \rfloor$ two-order filters). It is always less complex to split a two-order filter into two filters when possible: the coefficient in front of $\lfloor N/2 \rfloor$ is always larger than twice the one in front of $N[2]$. For RK4, since no matrix inversion is needed, the decomposition into low-order filters is not necessary.

Scheme	bilinear	RK23	RK4	integral
Memory needs	$3\lfloor \frac{N}{2} \rfloor + 2N[2]$	$3\lfloor \frac{N}{2} \rfloor + 2N[2]$	$N+1$	$3\lfloor \frac{N}{2} \rfloor + 2N[2]$

Table 5: Memory needs for several schemes.

Scheme	bilinear	RK23	RK4	integral
$+$	$10\lfloor \frac{N}{2} \rfloor + 5N[2]$	$26\lfloor \frac{N}{2} \rfloor + 8N[2]$	$12N+1$	$13\lfloor \frac{N}{2} \rfloor + 3N[2]$
$\times$	$18\lfloor \frac{N}{2} \rfloor + 7N[2]$	$44\lfloor \frac{N}{2} \rfloor + 9N[2]$	$10N-1$	$20\lfloor \frac{N}{2} \rfloor + 6N[2]$
shifts	$2\lfloor \frac{N}{2} \rfloor + 2N[2]$	$\lfloor \frac{N}{2} \rfloor + N[2]$	$3N+2$	-
exp	-	-	-	N

Table 6: Comparison of the operation number for several schemes.

The tables 5 and 6 give us the complexity overview related to each schemes. It appears that the RK4 scheme and the bilinear scheme have about comparable complexities. The integral scheme requires an exponentiation.

5 Conclusion

In the case of non-uniform sampling, the only still valid representation for IIR filters is the state representation, which is an Ordinary Differential Equation representation. The discretization is usually performed for uniform samples but may be as well performed for non-uniform samples. We have compared different numerical schemes in terms of stability, complexity and quality of the filtering result when applied to classical low-pass filters as Butterworth, elliptic or Chebyshev filters.

Euler schemes are both to be rejected, the explicit one for being unstable and the implicit one for being in a sense too stable, i.e. too dissipative. The three other studied schemes (bilinear, RK23 and RK4) give qualitatively good results. If applied to N-order filters, only RK4 is effective (no matrix inversion), but if 1- and 2-order decompositions are used, the complexity study does not allow to rank one of them clearly first. For RK23 and RK4, some oversampling is needed for inactive inputs to ensure stability, while this is unnecessary for the bilinear scheme.

This work is part of a wider study of signal processing in a complete asynchronous framework (asynchronous representation and processing of the data) [4]. The goal is definitively to reduce the number samples, the computational load and so the energy consumption. We strongly believe that this is an attractive approach for autonomous embedded systems.

References

[1] F. Aeschlimann, E. Allier, L. Fesquet, and M. Renaudin, *Asynchronous FIR filters: towards a New Digital Processing Chain*, 10th International Symposium on Asynchronous Circuits and Systems, Async'04, Hersonisos, Crete, April 19–23 2004, IEEE 2004, pp. 198–206.

[2] F. Aeschlimann, *Traitement du signal échantillonné non uniformément : algorithme et architecture*, PhD thesis, INPG, Grenoble, France, February 2006.

[3] E. Allier, G. Sicard, L. Fesquet, and M. Renaudin, *A New Class of Asynchronous A/D Converters Based on Time Quantization*, 9th International Symposium on Asynchronous Circuits and Systems, Async'03, Vancouver, Canada, May 12–15 2003, IEEE 2003, pp. 196–205.

[4] B. Bidegaray-Fesquet and L. Fesquet, *A fully non-uniform approach to FIR filtering*, 8th International Conference on Sampling Theory and Applications, SampTa'09, Marseille, France. May 18–22, 2009.

[5] L. Fontaine and J. Ragot, *Filtering of irregularly sampled signals*, Traitement du signal, **12**(2), 89–101, 2001.

[6] C.W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, 1971.

[7] D. Poulton and J. Oksman, *Digital filters for non uniformly sampled signals*, NORSIG 2000, Nordic Signal Processing Symposium Vildmarkshotellet Kolmarden, Sweden, June 2000, pp. 421–424.

[8] D. Poulton and J. Oksman, *Digital filters for non-uniformly sampled signals*, Traitement du signal, **12**(2), 81–88, 2001.

[9] S.M. Qaisar, L. Fesquet, and M. Renaudin, *Adaptive Rate Filtering for a Signal Driven Sampling Scheme*, 32nd International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2007, vol. 3, Honolulu, Hawaï, USA, April 15–20 2007, IEEE 2007, pp. 1465–1468.

[10] S.M. Qaisar, L. Fesquet, and M. Renaudin. *Adaptive rate sampling and filtering based on level crossing sampling*, EURASIP Journal on Advances in Signal Processing, **2009**, Article ID 971656, 13 pages, 2009.

[11] N. Sayiner, H.V. Sorensen, and T.R. Viswanathan, *A Level-Crossing Sampling Scheme for A/D Conversion*, IEEE Transactions on Circuits and Systems II, **43**, 335–339, 1996.

[12] C. Vezyrtzis and Y. Tsividis. *Processing of signals using level-crossing sampling*, IEEE International Symposium on Circuits and Systems, ISCAS 2009, Taipei, Taiwan, May 24–27 2009, IEEE, 2009, pp. 2293–2296.