# ECG baseline wander removal and noise suppression analysis in an embedded platform


**Proyecto Fin de Máster**
**en Ingeniería de Computadores**

Máster en Investigación en Informática, Facultad de Informática, Universidad Complutense de Madrid

Autor: Víctor Barbero Romero

Profesor director: David Atienza Alonso

Profesora colaboradora: Nadia Khaled, UC3M.

**Curso 2008-2009**

**Abstract**
ECG signal processing in an embedded platform is a challenge which has to deal with several issues. One of the commonest problems in ECG signal processing is baseline wander removal and noise suppression, which determine posterior signal process. In this report, two filtering techniques are presented and implemented to work on a Shimmer platform. Baseline wander removal based on cubic splines and morphological filtering are evaluated to check whether are suitable for realtime execution. The use of cubic splines is made to estimate the baseline wander in an ECG signal and then substract it from the input dataset to remove the baseline wander. Morphological operators are useful for signal processing and noise suppression. These techniques have been implemented and tested by a wavelet-based delineation algorithm and results are provided for comparison purposes. The project goal is to develop an implementation for baseline wander removal and noise suppression to be executed on an embedded platform, meeting its specific hardware constraints, and leaving room for posterior signal processing. This would allow to design a Wireless Body Sensor Network to support non-ambulatory healthcare.
**Keywords:** ECG, signal processing, noise suppression, baseline correction, embedded platform, cubic spline, morphological filtering.

**Resumen**
El procesamiento de señal de electrocardiograma (ECG) en un sistema empotrado es un reto que tiene que afrontar distintas facetas. Uno de los problemas más comunes en el procesamiento de ECG es la eliminación de *baseline wander* y la supresión de ruido, que condiciona el posterior procesamiento de la señal. En esta memoria se presentan dos técnicas de filtrado que se han implementado en un nodo Shimmer. La eliminación de *baseline wander* basada en *cubic splines* y un filtrado morfológico son evaluados para comprobar si son asumibles para su ejecución en tiempo real. El uso de *cubic splines* se realiza para estimar el *baseline wander* en una señal de ECG y eliminarlo. Los operadores morfológicos se utilizan para eliminación de *baseline wander* y supresión de ruido. Estas técnicas han sido implementadas y evaluadas por un algoritmo de delineación basado en *wavelets*, presentando sus resultados para su comparación. El objetivo del proyecto es desarrollar una implementación de filtrado para ser ejecutada en un sistema empotrado, asumiendo sus limitaciones hardware y permitiendo un posterior procesamiento de la señal. Esto permitiría entonces diseñar una *Wireless Body Sensor Network* para potenciar el cuidado de la salud fuera de un contexto ambulatorio.
**Palabras clave:** ECG, procesamiento de señal, supresión de ruido, corrección de *baseline*, sistema empotrado, *cubic spline*, filtrado morfológico.

# Contents

# Chapter 1

# Introduction

## 1.1 ECG signal processing

The function of the human body is based on signals of electrical, chemical or acoustic origin. Such signals provide information which may not be immediately perceived but which is hidden in the structure of the signal. This hidden information has to be decoded in some way before the signals can be given useful interpretations. The decoding of body signals has been found helpful in explaining and identifying several pathological conditions. This decoding process is sometimes easy to perform since only involves a limited manual effort such as visual inspection of the signal printed on a paper or in a computer screen. However, there are signals whose complexity is often considerable and, therefore, biomedical signal processing has become an indispensable tool for extracting clinically significant information hidden in the signal.

The process of biomedical signals is an interdisciplinary topic. It is needed some knowledge about the physiology of the human body to avoid the risk of designing an analysis method which may distort or even remove significant medical information. Of course, it is also valuable to have a good knowledge of other topics such as linear algebra, calculus, statistics and circuit design. Some decades ago, when computers first arrived in the area of medicine, automation was the main goal, but this has been modified over the years, since a physician must be ultimately responsible for the diagnostic decisions taken. Nowadays, the goal is develop computers systems which offer advanced aid to the physician in making decision.

Historically, biomedical signals have been assessed visually and manual ruler-based procedures were developed to make sure that those measurements

could be obtained in a standardized manner. However, there is relatively poor concordance between manually obtained measurements, since they depend on the personal criteria of a physician. The introduction of computer-based methods for the purpose of quantifying different signal characteristics is the result of a desire to improve measurement accuracy as well as reproducibility.

Another challenge for biomedical signal processing is to extract features to help to characterize and understand the information contained in a signal. These feature extraction methods can be designed to mimic manual measurement and support the diagnosis made by a physician whereas they are often designed to extract information which is not easy to find through simple visual assessment. For instance, small variations in heart rate cannot be perceived by the human eye but they have been found to contain very valuable clinical information.

In many situations, the recorded signal is corrupted by different types of noise and interference, originated by another physiological process of the body. When an electrode is poorly attached to the body surface or when an external source such as the sinusoidal 50Hz powerline interferes with the signal, the recorded signal is distorted in a way that it could be difficult to perform any automatic diagnosis. Therefore, noise reduction represents a crucial objective of biomedical signal processing. In some cases, the desired signal is so drastically masked by noise that its presence can only be revealed once appropriate signal processing and noise suppression has been performed.

There are three major clinical contexts in which algorithms for biomedical signal processing are designed: diagnosis, therapy and monitoring. In the *diagnosis* context, medical conditions are identified by the examination of signal information. This signal is acquired by non-invasive procedures which makes the examination less taxing on the patient. A diagnostic decision rarely requires immediate availability of the results from signal analysis, so this process can be done offline on a PC.

With regard to biomedical signal processing, *therapy* may imply the use of an algorithm to modify the behavior of certain physiological processes, as a pacemaker does with respect to cardiac activity. This algorithm is designed for its implementation in an implantable device like a heart defibrillator and, therefore, it must meet the demands of online and realtime analysis. Such demands determine serious constraints in terms of algorithmic complexity, maximal acceptable time delay and power consumption -the battery of a pacemaker is expected to last up to ten years. Biomedical signal processing algorithms are also an important part of realtime systems for *monitoring* of patients, spe-

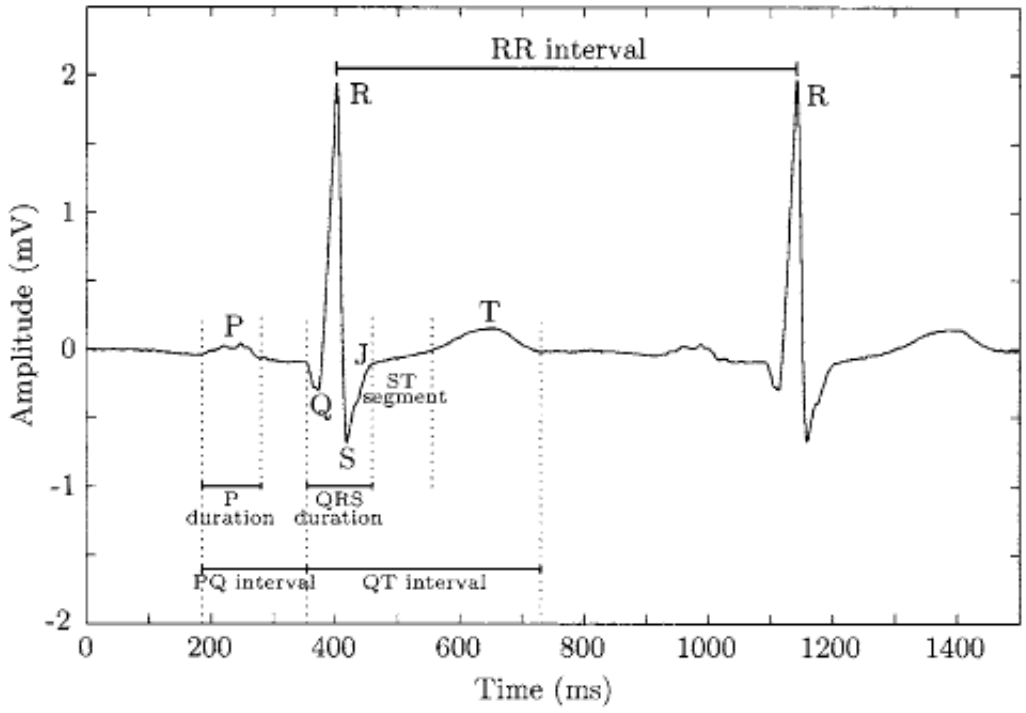
cially for those who suffer from a life-threatening condition. Systems needed in this clinical context are designed to detect changes in cardiac or neurological function. Thanks to this, irreversible damage can sometimes be prevented.

The **electrocardiogram** -ECG- describes the electrical activity of the heart. It is obtained by placing electrodes on the chest, arms and legs. With every heartbeat, an impulse travels through the heart, which determines its rhythm and rate and causes the heart muscle to contract and pump blood. The voltage variations measured by the electrodes are cause by the action potentials of the excitable cardiac cells, as they make the cells contract. The ECG is characterized by a series of waves whose morphology and timing provide information used for diagnosing diseases reflected by disturbances of the electrical activity of the heart. The time pattern that characterizes the occurrence of successive heartbeats is also very important.

The first ECG recording device was developed by the Dutch physiologist Willem Einthoven, using a string galvanometer which was sensitive enough to record electrical potentials on the body surface. He also defined sites for electrode placement on the arms and legs which remain in use today. Since then, ECG recording has developed incredibly and become an indispensable tool in many different contexts. The ECG record is used today in a wide variety of clinical applications. Its importance has been strengthened thanks to the discoveries of subtle variability patterns which are present in rhythm or wave morphology.

The electrodes used for ECG recording are positioned so that the spatiotemporal variations of the cardiac electrical field are sufficiently well-reflected. The difference in voltage between a pair of electrodes is referred to as a *lead*. The ECG is typically recorded with a multiple-lead configuration. The electrode wires are connected to a differential amplifier specially designed for bioelectrical signals. The ECG ranges from a few microvolts to about 1V in magnitude. Whereas the characteristic waves of an ECG have a maximal magnitude of only few millivolts, a wandering baseline in the ECG due to variations in electrode-skin impedance may reach 1V. The amplifier bandwith is commonly between 0.05 and 100-500Hz.

The characteristic waves of an ECG are shown in the following image. Atrial depolarization is reflected by the $P$ wave, and ventricular depolarization is reflected by the $QRS$ complex, whereas the $T$ wave reflects ventricular repolarization. The amplitude of a wave is measured with reference to the ECG baseline level, commonly defined by the isoelectric line which immediately precedes the $QRS$ complex.




Wave definitions of a heart beat and important wave durations and intervals.

One of the main reasons for computer-based ECG analysis is the capability to improve poor signal quality thanks to the use of signal processing algorithms. There are several most common types of noise and artifacts in the ECG. The **baseline wander** is an extraneous, low-frequency activity in the ECG which may interfere with the signal analysis, making the clinical interpretation inaccurate. When baseline wander takes place, ECG measurements related to the isoelectric line cannot be computed since it is not well-defined. Baseline wander is often exercise-induced and may have its origin in a variety of sources, including perspiration, respiration, body movements and poor electrode contact. The spectral content of the baseline wander is usually in the range between 0,05-1Hz [11] but, during strenuous exercise, it may contain higher frequencies.

The **electromyographic noise** is caused by the electrical activity of skeletal muscles during periods of contraction, commonly found in ECGs recorded during ambulatory monitoring exercise. Different muscles are active in producing the noise which corrupts the ECG signal. This kind of noise can either be intermittent in nature, due to a sudden body movement or have more stationary noise properties. The frequency components of this noise considerably overlap those of the QRS complex.

## 1.2 Wireless Body Sensor Networks

A Wireless Sensor Network -WSN- is a network composed of very small devices called nodes and, usually, a base station which stands for communication and nodes control. The nodes in a WSN are spread to measure a set of parameters and because of this, a wireless communication among nodes and between a node and a base station is needed.

There is a wide variety of issues, regarding science, government, health that calls for high fidelity and realtime observations of physical world. A network of smart wireless sensors could help to reveal what was previously unobserved in the location in which a phenomenon is taking place. Therefore, there is a challenge to design physically-coupled, robust, scalable and distributed systems based on embedded networked sensors. These nodes can help to monitor physical world and, by the use of its ad-hoc network, to coordinate and perform high-level identification. The information gathered by those nodes can be processed to perform a realtime embedded analysis thanks to which several actions could be taken after deciding whatever it may be necessary.

Those nodes in a WSN are supposed to work independently and are designed specifically to support an analysis process. The behaviour of a WSN is directly dependent on the process it is designed for. For instance, in the biologic research field, non-invasive habitat monitoring and a study of wildlife populations could be possible thanks to a WSN. Environmental monitoring, agriculture, architecture, transportation, military services, traffic report, stock management, healthcare... are fields in which the use of a WSN could help to improve the productivity.

Nodes in a WSN are basically composed of sensors, to recollect information, a microcontroller, to process that information and a radio transceiver to communicate the information to a base station or between nodes in the network. There are different types of node with regard to the main goal to achieve: low-power consumption or high-performance are topics taken into consideration by manufacturers.

The software in a node can be either a stand-alone application or an operating system. A stand-alone application provides the programmer an accurate control of the hardware but, in some cases, it could lead to hardware-dependent applications difficult to port. On the other hand, the use of an operating system provides advanced features such as task scheduling and inter-process communication. Typical tasks to be performed by the applications in the node are the control of sensor sampling and transmission frequencies, data aggregation

and signal processing to filtered the sensed signal and compress it so that the amount of data to be sent is reduced.

In the healthcare field, a Wireless Body Sensor Network enables continuous biomedical monitoring and care to support prevention and early diagnosis of diseases and, on the other hand, it allows to enhance patient autonomy thanks to not having to carry out an ambulatory test. Regarding the ECG recording, since they are done by placing electrodes on the body surface, it could be possible to perform a realtime ECG analysis by using several nodes placed in the skin.

These nodes have to use its sensors to gather ECG information, filter and process it and, just in case, communicate any circumstance detected. ECG monitoring has been performed by using a bulky patient unit or wirelessly transmitting the ECG to a monitoring system without processing. The use of a WBSN could achieve a real patient autonomy while the monitoring process is going on. By exploiting the limited processing power and memory resources of the nodes, it could be possible to perform an online detection of the fiducial waves of the ECG signal and, by interpreting that information, an online diagnosis of arrhythmias.

## 1.3 The Shimmer platform

Shimmer [23] is a small wireless sensor platform designed to support wearable applications. Its size and technology stands for low power consumption so that it can be used as a test node for WBSN healthcare application.

It is based on the Texas Instrument MSP430F1611 processor, which works at a maximum frequency of 8MHz and has 10KB of RAM and 4KB of Flash memory. It is equipped with several peripherals such as digital I/O, analog to digital converters, 802.15.5 radio, Class 2 Bluetooth radio, a MicroSD slot... and it is a proven solution in medical sensing applications. There is a ECG board daughter card to capture ECG data.

The MSP430F1611 is a 16-bit ultra low-power microcontroller based on a RISC architecture. The CPU is integrated with 16 registers that provides reduced instruction execution time, since the register-to-register operation execution time is one cycle of the CPU clock long. The instruction set consists of 51 instructions with seven address modes. Each instruction can operate on word and byte data. 24 of these instructions are emulated: they do not have op-code themselves and are replace automatically by the assembler with an

equivalent core instruction.

The microcontroller does not have a floating point unit and all the floating point operations required are transformed into several integer compatible operations. It does not support hardware division but it has a hardware multiplier. All the division and multiplication operations by a multiple of 2 are converted into a shifting operation. The compiler provides translation for division operations into equivalent integer ones whereas the hardware multiplier is used to execute multiplications which cannot be translated.

## 1.4 State of the art

The removal of the baseline wander in an ECG signal has been one of the first challenges in biomedical signal processing. The two major techniques employed for the removal of baseline wander are linear filtering and polynomial fitting.

The design of a linear, time-invariant, highpass filter involves the consideration of choosing the filter cut-off frequency and phase response characteristic. Obviously, the cut-off frequency should be chosen so that the clinical information in the ECG remains undistorted, so it is essencial to find the lowest frequency component of the ECG spectrum. Since the heart beat is not regular it is needed to choose a lower cut-off frequency, approximately $F_c = 0.5$Hz. Linear phase filtering is needed to prevent phase distortion from altering characteristic waves in the cardiac cycle. Finite impulse response filters can have an exact linear phase response, whereas infinite impulse response -IIR- filters introduce signal distortion due to nonlinear phase response. To avoid this non-linear phase response in an IIR filter, the use of forward-backward filtering stands as a remedy since the overall result is filtering with a zero-phase transfer function.

Unfortunately, filtering based on that cut-off frequency cannot sufficiently remove baseline wander that may occur, for instance, during a stress test, so the use of a linear time-invariant filtering would limit the use of an implementation to an ambulatory resting context. For this purpose, calculate the heart rate as inversely proportional to the RR interval length is a simple but useful way. Then, it could be possible to relate a time-varying cut-off frequency $f_c(n)$ to the heart rate so that a low-pass filter could be integrated with the filter structure. Linear filtering based on filters with variable cut-off frequency was initially suggested for off-line processing of ECG signals [24] and then extended for use in on-line processing [25]. Other approaches to linear, time-variant filtering have also been described based on adaptive, LMS techniques [26].

An alternative to baseline wander removal with linear filtering is to fit a polynomial to representative samples of the ECG, with one knot being defined for each beat. The polynomial estimating the baseline is fitted by requiring it to pass through each of the knot smoothly. This technique requires that the QRS complexes first be detected and it needs the PQ intervals to be accurately detected. This baseline wander removal technique is implemented and evaluated in this project.

Muscle noise -electromyographic noise- is a major problem in many ECG applications. Muscle noise is not removed by narrowband filtering but represents a much more difficult problem since the spectral content of muscle activity considerably overlaps that of the PQRST complex. Successful noise reduction by ensemble averaging is, however restricted to one particular QRS morphology at a time and requires several beats to work properly. One approach to muscle noise filtering is to use a filter with a variable frequency response, such as a Gaussian impulse response. The resulting performance on ECG signal of this techniques can be found in [27]. An application of this variable frequency response filtering to the baseline wander removal challenge can be found on [28]. However, time-varying properties may introduce artificial waves: a filter that provides considerable smoothing of the low-frequency ECG segments outside the QRS complex is likely to result in undesirable effects during the transitional periods. This distortion renders the filtered signal unsuitable for diagnostic interpretation of the ECG. There is a host of additional techniques to muscle noise reduction, but no single method has gained wide acceptance for use in clinical routing. As a result, the muscle noise problem remains largely unsolved.

In the field of WBSN, the ECG signal processing for embedded platforms represents a major challenge. Filtering techniques proposed require, overall, a good computation performance due to the use of floating point operations. To support realtime filtering, implementations based on these techniques have to meet the hardware constraints of that kind of platforms. Except for high performance nodes, no floating point hardware units are available. Furthermore, the amount of memory available is also a bound.

The ECG signal processing in an embedded platform concerns several issues related, such as signal filtering, signal compression, ECG delineation... These challenges are being on development under several research projects in our University as well as in the community interested in this field. CodeBlue [29] is an approach to healthcare based on WBSN. It is designed so that nodes collect heart rate and oxygen saturation to send data to a PDA in realtime. Devices raise an alert in case vital signs fall outside normal parameters.

The project here presented stands for the search of an efficient and high performance baseline wander removal and noise suppression implementation to allow posterior ECG delineation and interpretation in an embedded platform.

## 1.5 Project goal

The goal of this project is to implement a baseline wander removal and noise suppression filter to work on the Shimmer wearable sensor platform. For this purpose, two baseline wander removal techniques are implemented and then tested. These techniques have been designed to work offline so a realtime implementation is a challenge regarding the limited resources offered by the Shimmer platform. Electromyographic noise suppression is analyzed by an implementation based on morphological filtering. Baseline wander removal is done by the use of cubic splines and of morphological operators.

As input dataset, we will use the Physionet QT database [2] and to analyze the filtering result, a wavelet-based delineation algorithm [10] will help us to check whether the delineation after the filtering process is good enough to perform realtime ECG analysis so that, for instance, an arrhythmia diagnosis could be carried out later on.

Having a suitable baseline wander removal and noise suppression implementation for the Shimmer platform may help to design a Wireless Body Sensor Network in which each node would perceive ECG data and, after a filtering process, perform any other operation such as delineation, arrhythmia detection, signal compression...

# Chapter 2

# Use of cubic splines

## 2.1 Description

The following process of baseline wander removal is based on the paper by C. R. Meyer and H. N. Keiser [4]. There have been several approaches, as explained above, to the baseline wander removal problem. The technique here presented uses a polynomial to try to adapt to the baseline wander. In each beat, a representative sample is defined and called "knot". These knots in the input signal are chosen from the silent isoelectric line which, in most heart rhythms, is represented by the PQ interval.

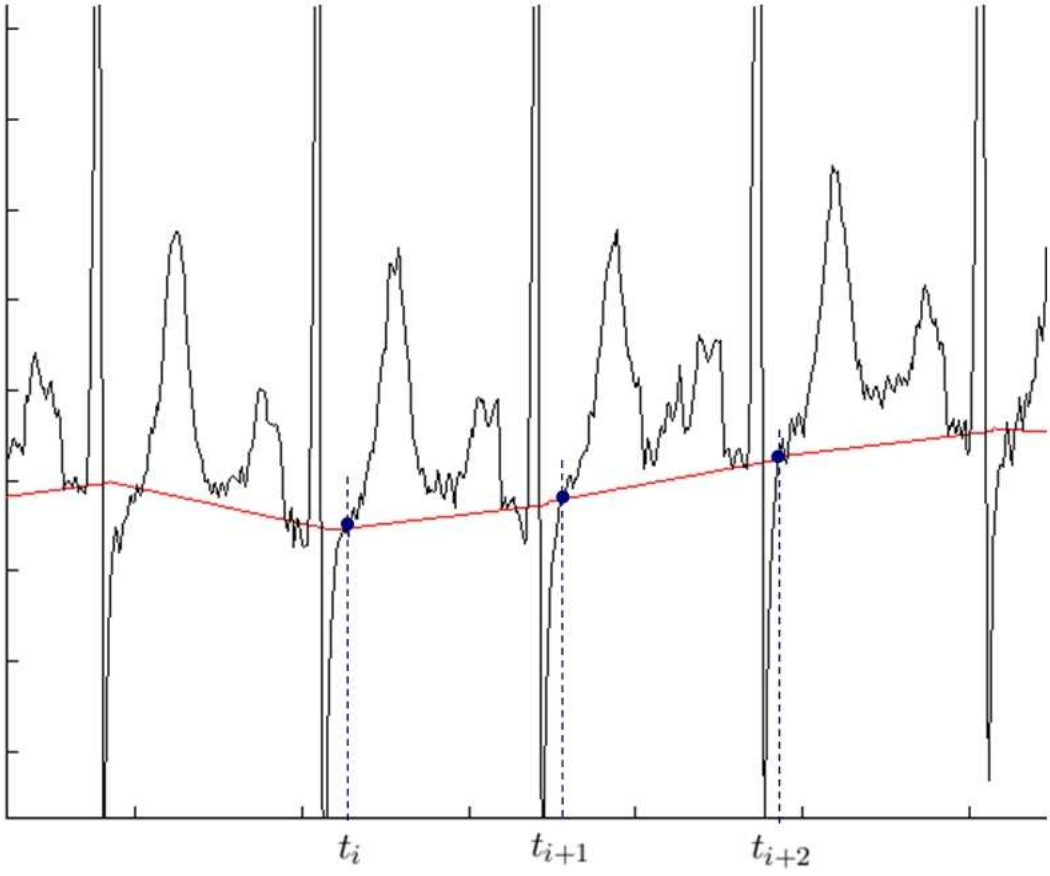
This technique comes from the work of some investigators who tried to adapt a straight-line to the segments connecting the pre-P-wave period and the post-T-wave period of each beat as successive baseline estimates. While this solution preserves low-frequency heart activity and leads to a small computational cost, such a first-order estimator can only accurately track baselines of very low frequencies [12]. Furthermore, the resulting baseline estimate does not adapt properly to the variations and, what is worse, its derivatives at the knots are discontinuous.

Increasing the order of the polynomial and selecting one knot per beat through which the baseline estimation must pass is the method used to remove higher-frequency baseline noise and preserve low-frequency heart information, which is useful for other processes to apply after the baseline wander removal. By using higher-order polynomials, the likelihood of producing an accurate baseline estimate increases, although it is obviously linked to an increased computational complexity.

Instead of letting the order increase as the number of knots does, third-

order polynomial fitting to successive triplets of knots represents a popular approach [4 and 13] and leads to good results in terms of baseline removal. This technique requires the QRS complexes to be detected and the corresponding PQ intervals to be accurately determined. It is chosen one averaged point in each PQ segment of the ECG as sample of the baseline. This segment is used because of the ease and accuracy in locating it.

At each PQ segment there is a knot through which the baseline noise estimator must pass. By fitting a third-order polynomial through these knots in the ECG signal we get the estimation for the baseline. These knots could be also defined by the end of the P wave. The polynomial is fitted in such a way that, one subtracted to the original signal, these knots have a value of 0.




ECG signal with three knots and the cubic spline baseline wander estimation $y(t)$

This technique intuitively approaches the benefits of using Lagrange's method to define a polynomial passing through all of the PQ-segment knots of the total ECG record without the penalties in complexity associated to high-order

polynomials. If we used Lagrange's method over a 20 sec record of ECG at 60bpm, the result would be nearly a 20th-order polynomial to be evaluated at each sample point in the record.

The use of this technique in an embedded platform has to consider the fact that we need to defined accurately the PQ interval in each beat. Fortunately, the paper suggests a PQ-segment locator, although we have had to redefine its working process. Furthermore, computing the polynomial in the interval of the input signal which is between three following beats leads to a memory consumption which has to be taken into consideration, as shown below. Finally, the function used to fit the polynomial requires some operations that are not easy to perform in a embedded platform as the MSP430.

The first step is to locate the knots of the successive beats in the input signal. These knots are denoted for the signal $x(t)$ as

$$x(t_i), i = 0, 1, 2, \ldots,$$

The baseline estimate $y(t)$ is computed for the interval $[t_i, t_{i+1}]$ by incorporating the three knots $x(t_i), x(t_{i+1}), x(t_{i+2})$ into the Taylor series expanded around $t_i$.

$$y_\infty(t) = \sum_{l=0}^{\infty} \frac{(t-t_i)^l}{l!} y_\infty^{(l)}(t_i)$$

For a third-order polynomial description, this series is truncated to

$$y(t) = y(t_i) + (t-t_i)y'(t_i) + \frac{(t-t_i)^2}{2}y''(t_i) + \frac{(t-t_i)^3}{6}y'''(t_i)$$

And the series expansion for the first derivative $y'(t)$ is

$$y'(t) = y'(t_i) + (t-t_i)y''(t_i) + \frac{(t-t_i)^2}{2}y'''(t_i)$$

At $t = 0$ we assume, to get this technique working, that

$$y(0) = x(0)$$

We must approximate the first derivative $y'(t_i)$ at $t_i$ by the slope between $x(t_{i+1})$ and $x(t_i)$

$$y'(t_i) = \frac{x(t_{i+1}) - x(t_i)}{t_{i+1} - t_i}$$

As shown in [14], classical splines of order three and higher, in which only the highest derivative is discontinuous, suffer stability problems during computation so we define both $y(t)$ and $y'(t)$ at each knot to arrive at a stable solution.

At the next beat, and to keep the cubic spline adapted to pass through all the knots considered, we must approximate, once more,

$$y'(t_{i+1}) = \frac{x(t_{i+2}) - x(t_i)}{t_{i+2} - t_i}$$

To find the remaining two variables $y''(t_i)$ and $y'''(t_i)$ in $y(t)$ the Taylor series for $y(t)$ and $y'(t)$ is studied for $t = t_{i+1}$

$$y(t_{i+1}) = y(t_i) + y'(t_i)(t_{i+1} - t_i) + y''(t_i)\frac{(t_{i+1} - t_i)^2}{2} + y'''(t_i)\frac{(t_{i+1} - t_i)^3}{6}$$

and

$$y'(t_{i+1}) = y'(t_i) + y''(t_i)(t_{i+1} - t_i) + y'''(t_i)\frac{(t_{i+1} - t_i)^2}{2}$$

To get the cubic spline to pass through this knot

$$y(t_{i+1}) = x(t_{i+1})$$

Inserting these values of $y(t_{i+1})$ and $y'(t_{i+1})$ into the previous equations we get

$$y''(t_i) = \frac{6(y(t_{i+1}) - y(t_i))}{(t_{i+1} - t_i)^2} - \frac{2(2y'(t_i) + \frac{y(t_{i+2}) - y(t_i)}{(t_{i+2} - t_i)})}{(t_{i+1} - t_i)}$$

$$y'''(t_i) = -\frac{12(y(t_{i+1}) - y(t_i))}{(t_{i+1} - t_i)^3} + \frac{6(y'(t_i) + \frac{y(t_{i+2}) - y(t_i)}{(t_{i+1} - t_i)})}{(t_{i+1} - t_i)^2}$$

where, as we know,

$$y(t_{i+2}) = x(t_{i+2})$$

We have then the baseline estimate $y(t)$ completely specified to be computed in the interval $[t_i, t_{i+1}]$. To get the signal without baseline wander we have to subtract from the ECG signal samples in that interval the baseline estimate $y(t)$. Then, this procedure has to be repeated for the next interval $[t_{i+1}, t_{i+2}]$ using the knots $x_{i+1}, x_{i+2}$ and so on.

The performance of the cubic spline technique is critically dependent on the accuracy of the knot determination. The PQ interval is relatively easy to

delimit in ECGs recorded during resting conditions but it may be difficult to find in recordings with muscle noise or when certain types of arrhythmias are present, such as ventricular tachycardia, which distorts severely the ECG signal and makes the location process almost impossible. When these circumstances take place, the PQ interval is not well-defined and, therefore, this technique is inapplicable.

On the other hand, this approach to baseline wander removal results in a time-variable cut-off frequency linear filtering since the baseline estimate tracks rapid baseline wander when a fast heart rate is encountered. More knots become available at faster heart rates, so the segment between beats is shorter and the cubic spline can adapt itself better to the knots located. According to this, polynomial fitting performs poorly when the available knots are too far apart since the interval between $t_i$ and $t_{i+1}$ is too long to achieve a proper estimation.

Taking this into consideration, we must determine the PQ-interval knot as accurately as possible. In [4], it is detailed how to locate the knot for each beat. The location in time of the PQ-interval knot is placed 66msec before the Q-wave's maximum downslope. This might seem a simple approach but having chosen this time distance between the maximum slope and the knot as a constant makes the knot detection to trigger in the same relative position in each beat. Therefore, we get a sample point in each beat next to the beginning of the QRS complex which can be considered as belonging to the silent isoelectric line and, due to the use of the cubic spline and by adjusting it to the knots located, we get always a spline which pass through the same relative points in each beat, which is the purpose of this technique.

The first step in locating the PQ-interval knot is to detect the Q-wave's maximum downslope. The downslope of the ECG signal at any sample with time index $t$ is computed using an average negative slope estimate where

$$downslope(t) = x(t-3) + x(t-1) - x(t+1) - x(t+3)$$

Since we are using a 250Hz sampling frequency, the time interval between two adjacent samples is 2msec. It is defined to detect the maximum downslope in the working sample when the computed downslope value exceeds 60% of the previous maximum.

Once the PQ-knot is located following the previous procedure, the ordinal value for the knot is calculated as the average ordinal value of the four data points which are nearest to the sample in which the knot has been detected. Using these four points to estimate the ordinal value of the knot eliminates the

effects of the 60Hz noise, according to the data sampling frequency of 250Hz: an average over four points acquired at 250Hz spans 16msec or nearly one cycle of 60Hz noise. As shown in [4], from digital filtering theory, we know that averages consisting of symmetrically space points spreading exactly over one cycle of a sinusoidal signal are not biased by that signal component.

At this point, we are able to implement the technique proposed for baseline wander removal: we must get the locator working to calculate, for each data sample, the downslope. If the computed downslope exceeds 60% of the previous maximum, we know that the knot is 17 data samples (66msec at 250Hz) before the point which triggers the downslope. Then, by averaging the four points next to and including this sample, we get the ordinal value of the knot, which is going to be considered as $x(t_i)$. As shown before, we need three consecutive beats which its correspondent knots to start calculating the cubic spline, so we need to store in memory the data samples from the knot at $t_i$ to the last sample of the third beat and, when it is completed, process the first beat.

This takes a lot of memory to operate so, in the next section, it is explained the implementation proposed and some optimizations to the knot locator.

## 2.2 Implementation

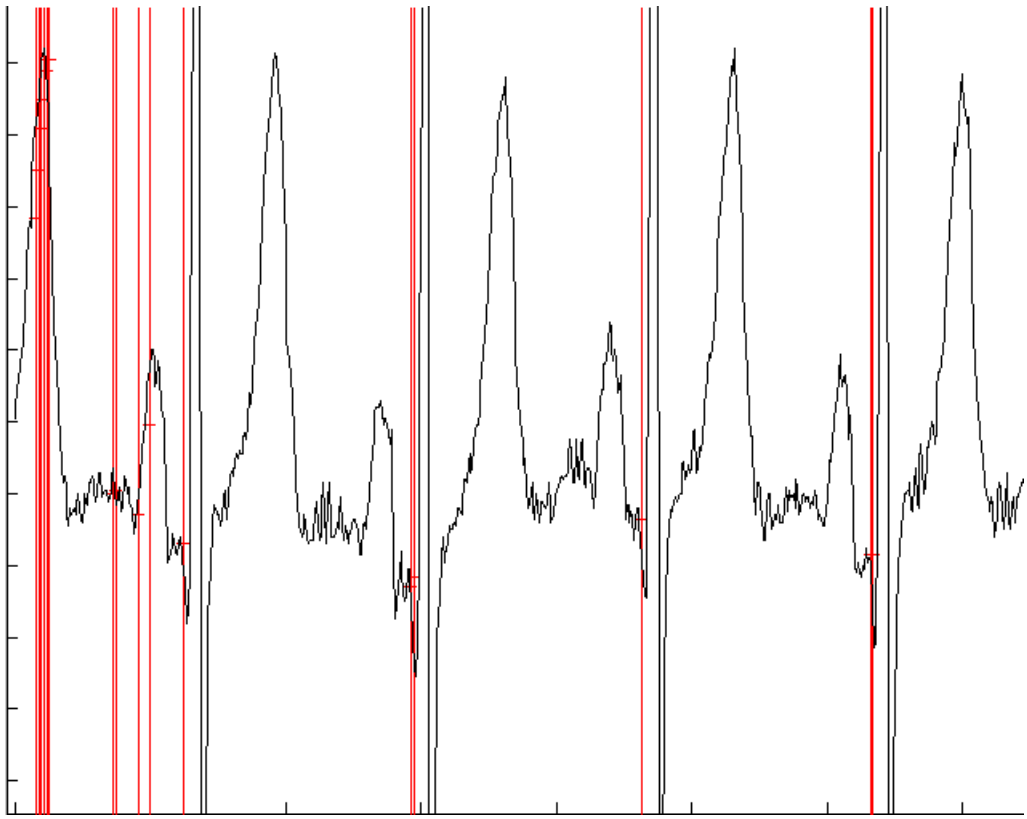
The implementation of the cubic spline technique has to take into consideration the fact that the embedded platform used in this project is limited in terms of memory and computation hardware. As explained before, this technique leads to a memory consumption that has to be considered and it uses some mathematical operations that are not directly supported by the platform but translated into compatible ones by the compiler, resulting in more complex and slower operations.

At first, we have to begin by implementing the knot locator. This is done by a static circular buffer in which input samples are stored and by the of use several indexes to rapid access to those positions of this buffer in which we can find the samples needed for the downslope calculation. The first samples of the input ECG signal are only stored and produce no result -or zero output signal- since are needed to fill the buffer to get enough data to start operating.

Once the buffer is initialized, it keeps storing each input sample. For each new sample, the downslope value is calculated and then we check whether it exceeds the 60% of the value of the last maximum downslope as explained before. This makes the locator enter in the "search period": according to [4],

during this period, the first downslope value which is less than it predecessor defines its predecessor's value as the new maximum and, therefore, we assume its predecessor is the point of the maximum downslope in the QRS segment.

So, to get the locator working, we must keep storing the value of the downslope of the previous sample and of the sample in which we are working. If we are searching and the downslope value is less than the last obtained, we finish the search period, since we have located a knot. Then, we can store the 60% of its value to trigger the next search period.



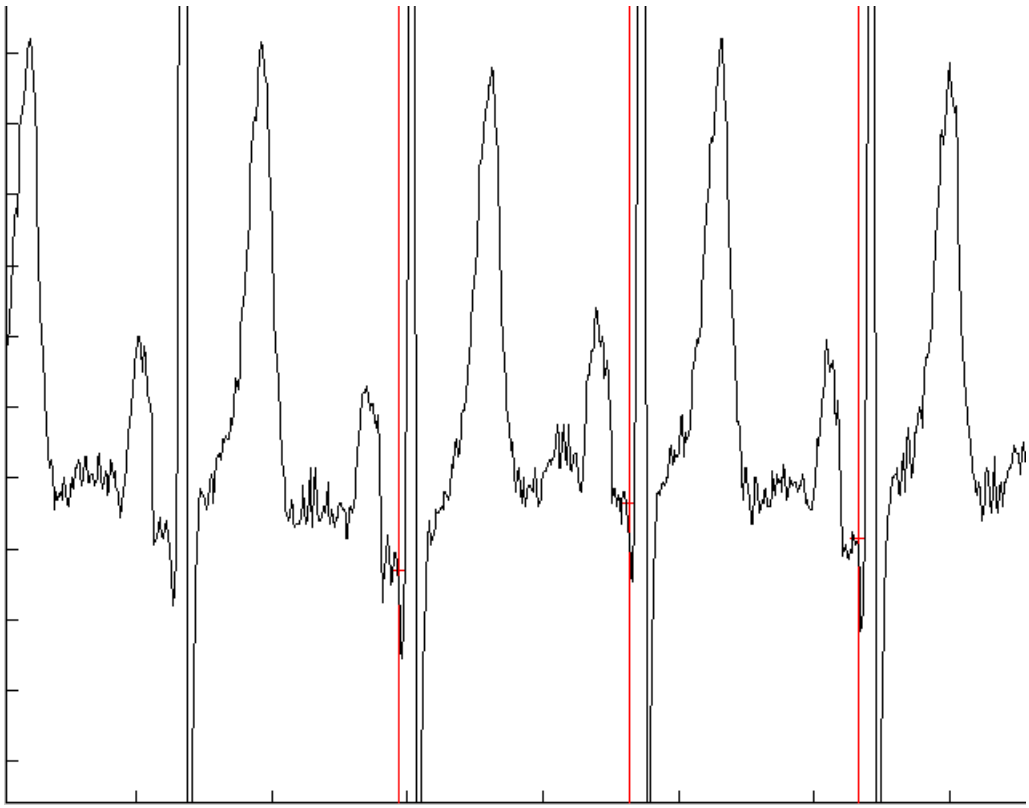
Initial overlocation failures until the maximum downslope is achieved

This process, detailed in [4], did not work properly for our purposes. As shown in the previous image, since there is not an initial maximum downslope value achieved, the downslope locator is triggered as the first QRS takes place. Vertical red lines are displaced to the left to indicate where the knot is encountered but, obviously, there is no knot in the downslope. These initial failures would lead to an extra computation cost because of detecting beats where there are not. To overcome this problem, we have defined an initial threshold period of samples in which the downslope calculus is done but no beat is processed.

This allows the locator to get an appropriate value of the maximum downslope without causing trouble in the following steps of the technique.

Another problem encountered in the implementation of the locator was that, when the QRS downslope takes some time to happen, several maximum downslope values were detected in the same downslope and, once more, this results in a overlocation failure. To solve this, we define a 60 samples period at least between each beat to be detected: the maximum downslope search is not triggered until there have been 60 samples since the last knot. This makes the locator to work only for a 250bpm heart rate as a maximum, which we consider is high enough to ensure the stability of the implementation. Although there is no agreement in the maximum heart rate that a human can achieve and in the light that it depends on terms of age [15], the following equation of Inbar [16] shows that we can assume this 250bpm maximum heart rate.

$$Max.HeartRate = 205.8 - 0.685age$$



First knot locations after the initial threshold period

Once the knot is located, we get the average of the four points nearest to and including it and, as we know, we must start to process the following data

samples as belonging to a new beat. To support the fact that we need three consecutive beats and to store all the information associated with each one, we use a struct. This struct stores information about the beat concerning how many data samples long is, the ordinal value of its knot, a pointer to a buffer in which all the data samples of the beat are stored and how many bits are needed as a maximum to represent these values.

When a new knot is located, it also means that the previous beat has just finished. The length of the previous beat is then stored in its associated struct and, by using a variable to keep track of the maximum sample value in that beat, we can get how many bits are needed to represent all the values in that beat. The purpose of this will be explained further below. We can then restart the length counter and reinitialize the variable for the search of the maximum value in the new beat.

When a new beat has been detected, it is time to proceed with the cubic spline procedure in case it is at least the third beat that has taken place. Otherwise, we have to wait until we have the three first beats detected, with its associated structs completely filled and all the samples values stored in memory. While we are online processing, we can operate with the beat before the previous when a new one is detected. By using the amount of bits needed to represent all the values in the beat we can get how many bits shift to the left each sample in the beat to get the best accuracy in all the operations.

For a beat $i$, and following the equations explained in the previous section, we need to get the ordinal value of its knot and of the knots of the two following beats. These values are arithmetically left-shifted to get the most of this technique. Furthermore, we need the information about how long the beats $i$ and $i+1$ are. With these data, we can obtain $y'(t_i)$ and then $y''(t_i)$ and $y'''(t_i)$. These variables use the length of the beat to the power of 3 and to the power of 2. Once we get the value of these three variables, which will remain constant during the process of all the beat, we use a new struct to facilitate the computation of the result.

If we were able to use a realtime operating system in the node, with a dispatcher and threads, it would be easy to have two threads in execution to support this technique: one thread would be responsible for the knot location and another thread would be in charge of computing the result of this filter when it had available input data. However, all the tests in the node are made in a monolithic way so this implementation has to consider itself the scheduling and production of the result. Thus, a new struct is used to store the values of $y'(t_i)$, $y''(t_i)$ and $y'''(t_i)$, the ordinal value of the knot of the beat, the length

of the beat, a pointer to the buffer with all its samples and how many bits the variables in the calculation have been shifted.

This new struct uses a queue scheme: we will have a pointer to a struct with the information about the beat in which the computation is taking place and this struct will also have a pointer to the following struct of a beat ready to be processed, if it is the case. When the new struct is ready, it is stored in the process queue. The struct used while the beat was being received via the input signal and in which had to wait until being processed is no longer needed, so we can dispose it for lower memory consumption.

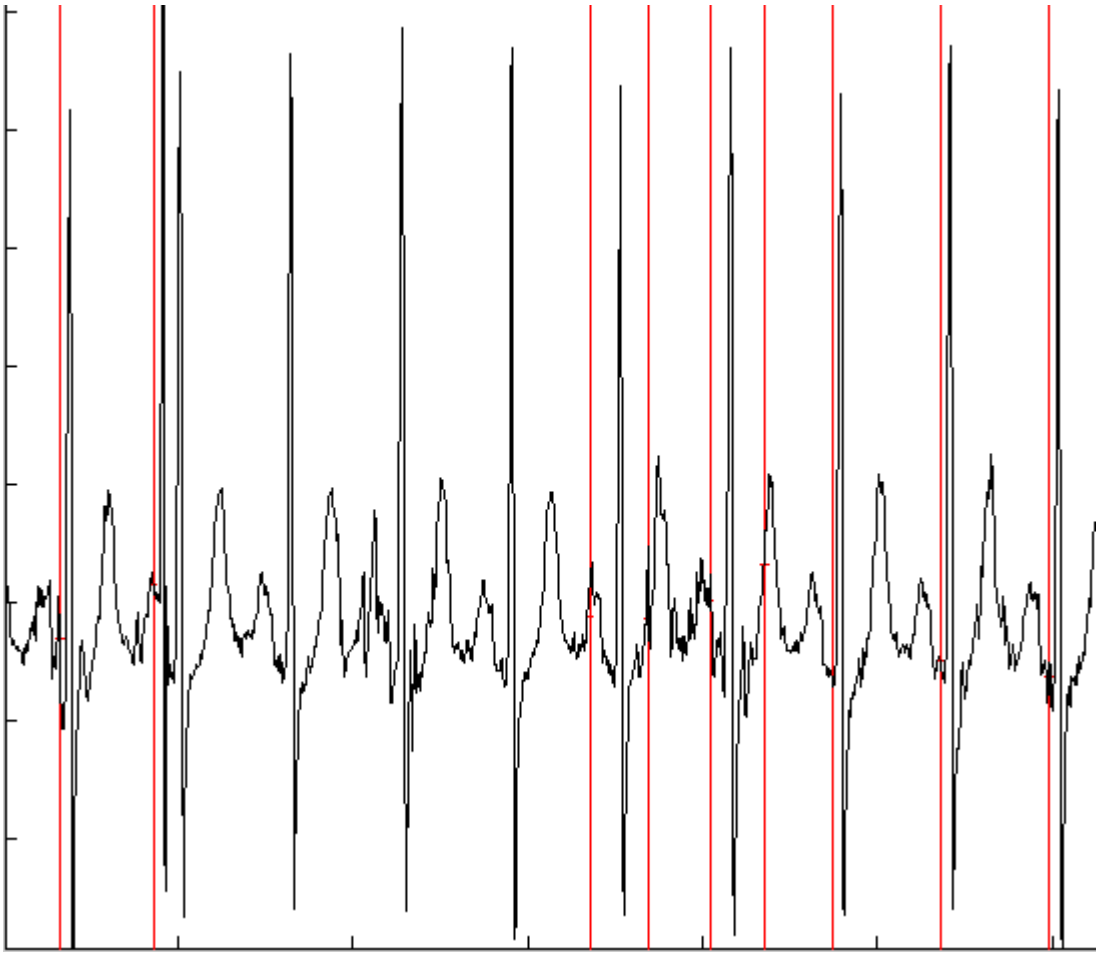
Finally, once the new input sample is processed, we pay attention to the queue to check whether there is a beat to calculate or not. If so, we simulate realtime operation by providing as output only one new value of the beat which is first in the queue. With the values stored in its struct inserted into the equation for $y(t)$, we get the baseline estimation for the sample in which we are working. Then, subtracting the estimation from the input sample, we get a new sample of the output signal without baseline wander. After moving indexes for the circular buffer, and increasing the counter for the length of the current, the implementation is ready to receive a new input sample.

While we were implementing and testing this technique in a PC platform, we used float datatypes for the computation of $y'(t_i)$, $y''(t_i)$ and $y'''(t_i)$ but the MSP430 platform does not support floating point computation. Therefore, we had to adapt the calculation to integer variables. At first, simple integer implementation is not accurate enough for the calculations needed by this technique so decided to arithmetically left-shift as much as possible all the data needed in the computation of those variables to get the best accuracy. When no shifting is made and integer variables are used, the result is a straight horizontal line which passes through the knot of each beat.

In terms of calculation, the MSP430 platform meets the requirements of this technique except for the computation of $y'''(t_i)$. As defined in its equation, this variable uses the power of 3 of the length of the beat. The MSP430 platform uses 16-bits width registers, so only 40 to the power of 3 can be operated. Beats whose length is longer than 41 samples, which is obviously the commonest case -40 samples between each beat means a heart rate of 375bpm-, lead to an overflow in the register. Therefore, the calculation of $y'''(t_i)$ has to be performed using 32-bits width operations, supported by the MSP430 but more expensive in terms of computation.

Another factor we have considered is that we do not know how long a beat is going to be. We can approximate to the length of each beat once we have

detected the first beats in the signal but this length cannot be considered as constant. Therefore, we have to decide the length of the buffer in which all the data samples of a beat is going to be stored. In our tests, we consider a maximum length for a beat of 500 samples. This results in a 30bpm minimum heart rate for this technique to work but, on the other hand, and to improve the stability of the implementation, we release the maximum downslope restriction once the length of the current beat is approaching to that limit. This helps to avoid a buffer overflow and, according to our tests, improves the endurance of the implementation to artefacts.

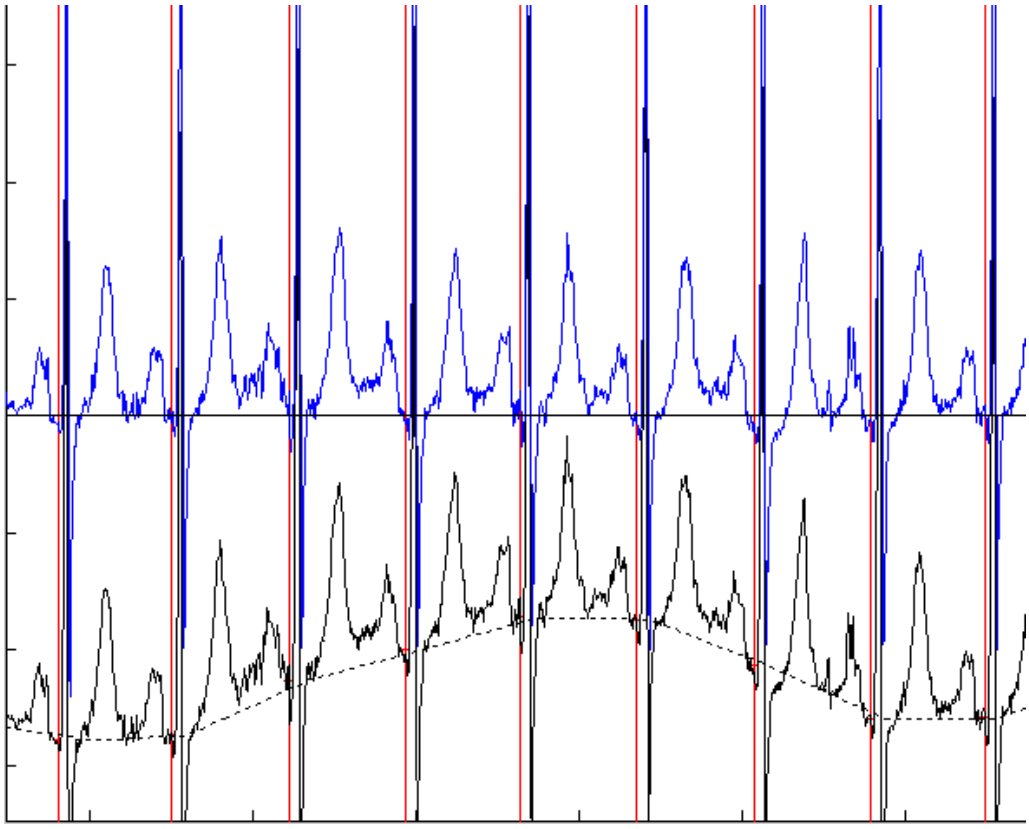


The knot locator is sensitive to artefacts by checking the limit of the buffer

The image above shows that this technique supports the presence of artifacts in the input signal. The second vertical red line corresponds to the knot location of a downslope that has triggered the locator. However, that

downslope does not belong to a QRS complex but it is an artifact or something truly irregular in the beat. The high downslope of this peak triggers the knot locator and, thus, defines a new last maximum. The following QRS complexes and its downslope does not exceed the 60% of the last maximum, so almost three following beats are ignored. Once the buffer for this pseudobeat is going to be filled, the last maximum value turns to 0 and then there happen some overestimation failures. There are four knots located wrongly, until the last maximum value is properly set again and the technique keeps working as expected, at the right end of the image.

To conclude with the implementation of this technique the following image shows how it works for a given input with baseline wander. The vertical red lines mark the position of the knot in each beat detected. The black ECG signal is the input signal whereas the dotted spline is the estimation of the baseline wander. Finally, the blue ECG signal is the result of the baseline wander removal from the original input signal. The horizontal solid line shows the zero value.



Input ECG signal, baseline estimation and output signal

# Chapter 3

# Morphological filtering

## 3.1 Description

Morphological operators have been widely used in the signal and image processing fields because of their robust and adaptive performance in extracting the shape information in addition to their simple and quick set computation. The technique here presented is based on the work of Y. Sun et al. published in [3]. This work comes from a previous development by Chu and Delp, who used the combined opening and closing operators for baseline correction of ECG signals and good filtering performance was obtained. However, their morphological filtering algorithm distorts the characteristic points in the ECG signal, which is not suitable for our purposes.

Mathematical morphology, based on sets operations, provides an approach to the development of non-linear signal processing methods, in which the shape information of a signal is incorporated. In these operations, the result of a data set transformed by another set depends on the shapes of the two sets involved. A structuring element has to be designed depending on the shape characteristics of the signal that is to be extracted.

There are two basic morphological operators: erosion ($\ominus$) and dilation ($\oplus$). Using erosion and dilation we can define derived operators: opening ($\circ$) and closing ($\bullet$). We consider $f(n), \{n = 0, 1, \ldots, N-1\}$ as a discrete signal consisting of $N$ points and $B(m), \{m = 0, 1, \ldots, M-1\}$ a symmetric structuring element of $M$ points.

Erosion ($\ominus$) is a shrinking operator in which the values of $f \ominus B$ are always

less than those of $f$.

$$(f \ominus B)(n) = min_{m=0,\ldots,M-1}\left\{ f\left(n - \frac{M-1}{2} + m\right) - B(m) \right\}$$

Dilation ($\oplus$) is an expansion operator in which the values of $f \oplus B$ are always greater than those of $f$.

$$(f \oplus B)(n) = max_{m=0,\ldots,M-1}\left\{ f\left(n - \frac{M-1}{2} + m\right) + B(m) \right\}$$

$$\text{for } n = \left\{ \frac{M-1}{2}, \ldots, N - \frac{M+1}{2} \right\}$$

The opening of a data sequence can be interpreted as sliding a structuring element along the data sequence from beneath and the result is the highest points reached by any part of the structuring element. Opening is used to suppress peaks and is defined as: $f \circ B = f \ominus B \oplus B$. The closing of a data sequence can be interpreted as sliding a flipped-over version of the structuring element along the data sequence from above, and the result is the set of lowest points reached by any part of the structuring element. Closing is often used to suppress pits and is defined as: $f \bullet B = f \oplus B \ominus B$.

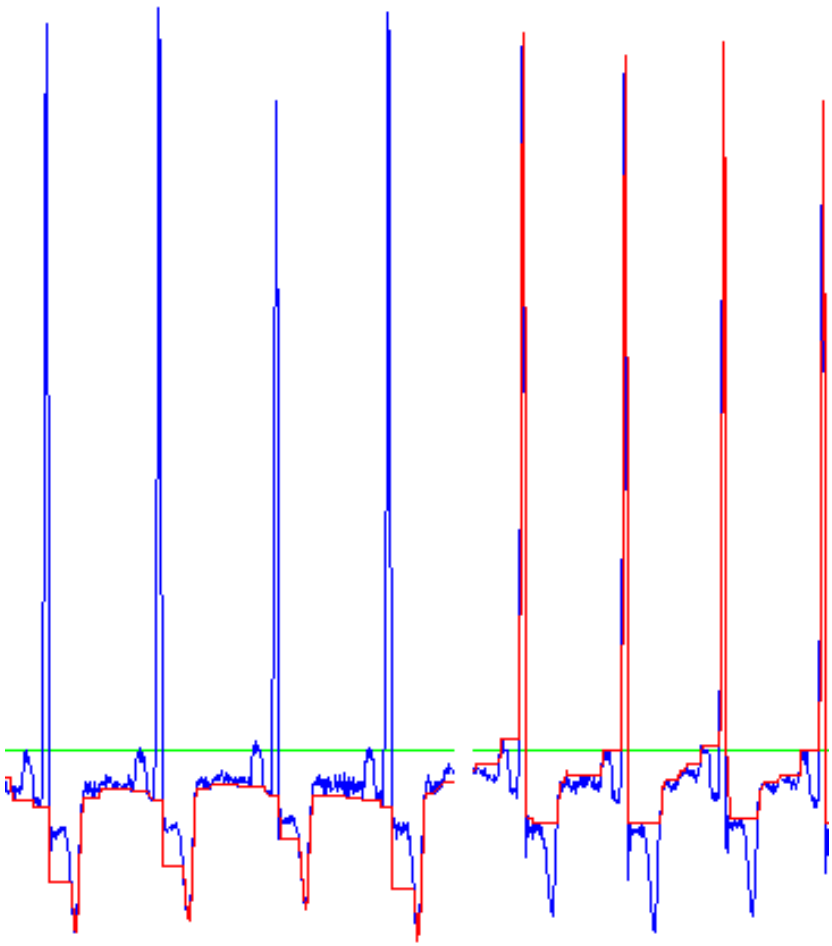
This technique uses a sequence of opening and closing operations to perform a baseline wander removal and electromyographic noise suppression. Based on the different characteristics of the baseline drift and the noise contamination in the ECG signals, different structuring elements and different morphological operators are used.

The baseline wander removal is performed by removing the drift in background from the original ECG signal, following the method presented in [17]. To get the baseline estimation we use $f_b = f_o \circ B_o \bullet B_c$ and then this baseline drift is subtracted from the original input signal to get the filtered output signal. The signal is first opened by a structuring element $B_o$ for removing peaks in the signal. Then, the resultant waveforms with pits are removed by a closing operation using the other structuring element $B_c$. $B_o$ and $B_c$ are defined as two horizontal line segments of zero amplitude but with different lengths. The result of this compound operation is then an estimate of the baseline drift $f_b$ and, therefore, the correction of the baseline is then done by subtracting $f_b$ from the original signal $f_o$.

Different lengths in $B_c$ and $B_o$ are used because the construction of the structuring element for baseline correction depends on the duration of the

characteristic wave and the sample frequency of the ECG signal $F_s$Hz. If the width of a characteristic wave is $T_w(s)$, the number of samples of that wave is $T_w F_s$ so the structuring element $B_o$ should have a length larger than $T_w F_s$. The subsequent closing operation, which uses $B_c$, takes place to remove the pit left by the opening operation, so the length of the structuring element $B_c$ must be longer than the length of $B_o$. In an ECG signal, the more characteristic waves are the P wave, the T wave and the QRS complex, which are generally less than 0.2 sec.

Hence, $L_o$, the length of $B_o$ is selected as $0.2F_s$ and $L_c$, the length of $B_c$ is typically selected to be longer than $B_o$, at about $1.5L_o$. Since we are using $F_s = 250$Hz as sampling frequency, we get $L_o = 0.2F_s = 0.2 \times 250 = 50$ and $L_c = 1.5L_o = 1.5 \times 50 = 75$.



Closing operation (left) and opening operation (right) with a zero structuring element ($B_o$) on a ECG signal

After baseline correction, noise suppression is performed by an opening and closing concurrent operation, and then the results are averaged. The opening and closing operations for noise suppression proposed use a structuring element pair, $B_{pair}$ defined as $B_{pair} = \{B_1, B_2\}$ with $B_1$ and $B_2$ different in shape but equals in length. The process of signal conditioning for noise suppression is described by

$$f = \frac{1}{2}(f_{bc} \oplus B_1 \ominus B_2 + f_{bc} \ominus B_1 \oplus B_2)$$

where $f$ is the resultant signal after noise suppression and $f_{bc}$ the signal after baseline correction. The $B_{pair}$ is selected by considering the purpose of analysis and the morphological properties of the ECG signal. $B_1$ is selected to be a triangular shape, used to retain the peaks and valleys of the characteristic waves, such as the QRS complex. To minimize the distortion to the ECG signal, the length of $B_1$ is chosen to be the same as that of $B_2$. The length of both structuring elements is related to the bandwidth of the ECG signal and the sampling rate. Since the sampling frequency is fixed, a shorter structuring element can be used to reduce the distortion of the waveform, so $B_1 = (0, 1, 5, 1, 0)$. $B_2$ is chosen to be a line segment of zero value and the same length as $B_1$, so $B_2 = (0, 0, 0, 0, 0)$.

Using the proposed structuring element pair, noise can be suppressed while reducing the smoothing of the significant peaks and valleys in the ECG signal, which are essential to subsequent reliable detection of the characteristic waves of the input signal.

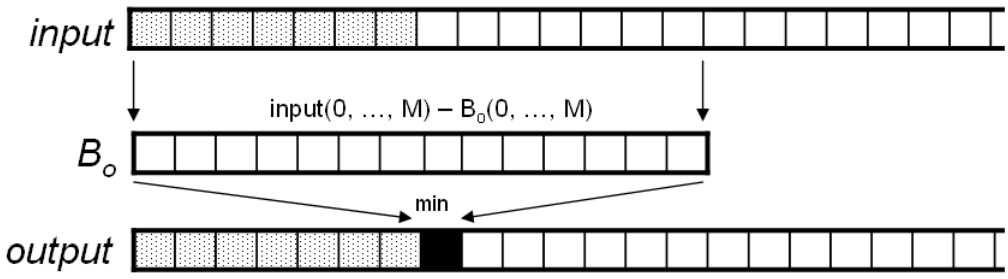
## 3.2 Implementation

Without a doubt, this morphological filtering technique has been designed to work offline. According to the definition of the morphology operators, we need all the input data to do the first erosion operation and this is impossible to perform in a realtime embedded platform. What is worse is that the different morphology operations are compound: the second dilation operation needs the result of the previous erosion so we need to keep track of a big amount of data in memory to get this technique working.

However, our implementation uses a fixed low amount of memory and supports the baseline wander removal and the noise suppression operations provided by this technique. We use several circular buffers to operate and to store the partial results needed in each morphology operation and the final implementation has been optimized to reduce the execution time due to an

improvement in the use of buffer indexes and peaks/pits comparison analysis, as will be exposed later.

All the buffers used in this implementation are circular buffers made of 16-bits integers, the basic datatype in which the MSP430 platform operates. The calculation of $f_b = f_o \circ B_o \bullet B_c$, the detected baseline drift, is made through a sequence of erosion and dilation operations. The first operation, $f_1 = f_o \ominus B_o$ is an erosion operation that uses the input signal $f_o$ and the zero value structuring element $B_o$. According to the definition of this technique, $B_o$ is 50 elements long. If we pay attention to what the erosion operation means, each sample $f(n)$ in the input dataset in which $n = \{\frac{M-1}{2}, \ldots, N - \frac{M+1}{2}\}$ is going to be used in an operation where to this sample will be subtracted each element of $B_o$. The minimum value obtained in all the subtractions will be stored in $(f \ominus B_o)(n)$. For $n = \{0, \ldots, \frac{M-1}{2} - 1\}$ there is no definition for $(f \ominus B_o)(n)$ so we use $(f \ominus B_o)(n) = f(n)$. And, since it is a realtime implementation, we do not know how long the input data will be, so there is no definition for $N$.




Representation of the erosion operation

Thanks to the fact that the structuring element $B_o$ is an horizontal line segment of zero value, the first erosion operation can be implemented as a simple minimum search in 50 elements, the length of $B_o$. Hence, the first circular buffer we use to support the erosion operation is 50 elements long. In this buffer, each new input sample is hosted in a position. When the buffer is completely filled for the first time, each new input sample will overwrite the sample stored 50 samples before, since it is a circular buffer. To implement the erosion operation we use an index in the array which points to the minimum value available in the buffer. Obviously, there is another index which points to the latest position used in the buffer so that its circular scheme can work.

The first 50 input data samples are only used to fill this circular buffer. When a new sample is received in this initialization period, its value is compared to the previous minimum value received and stored in the buffer. In case

of being lower or equal to the previous minimum, the correspondent index is updated and it is ready to receive a new sample. The idea of updating the index when the new value is equal to the previous minimum is made to keep the minimum index alive as much as possible. Since it is a circular buffer, it is better to have all the indexes used as updated as possible and pointing to the latest minimum, although equal to the previous one, because of the fact that it reduces the chances of those indexes to be disabled by an overwritting operation in the position they are pointing.

While the circular buffer is filling, the result of the erosion operation is the same value received as input. Once the buffer is filled the operation is different. Before storing the new input sample, we check whether it is going to overwrite the previous minimum found in the buffer. If it happens but the new value is equal or even lower than the previous minimum, there is no action to take: the minimum index will stay pointing to the current position and the operation will work properly. If the new value is higher than the previous minimum and it is going to overwrite it, we need to find the new minimum in the buffer. This is made by storing the new input sample in the buffer and then linear searching completely for a new minimum.

If the new sample is not going to overwrite the previous minimum -*minimum index* $\neq$ *current position*-, we have to check whether its value is lower or equal to the previous minimum. If it is, the minimum index is updated to point at the current position. Otherwise, no change is made to any index. There are several approaches to the use of a circular buffer but we use this implementation because it provides a low number of indexes changes and comparisons.

However, this implementation has a flaw in terms of efficiency and computation cost. As explained, once we check a new input sample overwrites the previous minimum value, a linear search is made in the circular buffer. This buffer is 50 elements long so such a linear search is likely to be avoidable. The amount of linear searches made in the circular buffer depends squarely on the shape of the input signal. It could be possible to find an input dataset in which the variation of its values would not trigger enough linear searches to consider any solution to avoid them. However, we are working with ECG signals and, as explained above, a structuring element of only 50 elements in length. When a steep slope is found in the ECG signal and, after that, a period of no variation, it could be easy to suffer from a minimum overwrite that would trigger a linear search.

One solution to avoid the linear search would be the use of a second index to the minimum value but this introduces more comparisons and *if* sentences that

makes computation slower. So we had to analyze how many linear searches were made and whether using a second index was worthy. For test purposes, we have been using several ECG signals of 225000 samples -recordings of 15 minutes. Without the use of a second pointer, for a reference input signal, 25983 linear searches were made in this first erosion operation, which corresponds to the 11,55% of the input samples, so we decided to use a second index to point at the first lower or equal minimum value in the buffer.

This solution changes the behaviour of the implementation of the operation and increases the amount of comparisons. The second pointer can be disabled by storing the value $-1$. Now, in the initialization period, if the new input sample is not lower or equal to the previous minimum found, we have to check whether it is lower or equal to the value pointed by the second index. As before, updating this pointer in case of the new sample to be equal to the value pointed makes this index to remain alive as long as possible so that the final number of linear searches is reduced.

When the circular buffer is filled, the operation with two indexes is more complex: if the new sample is going to overwrite the minimum in the buffer, we check whether it is lower or equal to that value. If it is, there is no change to make and the buffer is ready. Otherwise, we pay attention to the second index. If this second index is $-1$, we have no other solution to do than a linear search to update the minimum and the second minimum indexes. If the second index is pointing to another value in the buffer and the new input sample is not lower or equal to that value, we make the minimum index point at that value in the buffer and disable the second index.

If the new sample does not overwrite the position of the minimum value but it is lower or equal to this minimum previously found, the minimum index points at the current position and the second minimum index points at the previous minimum position. If the new sample is going to overwrite the value in the position pointed by the second minimum pointer, we disable this pointer in case of the new value to be greater than the previous second minimum. Otherwise, the second minimum pointer is still valid. Finally, if this second minimum index is not disabled and the new input sample is lower or equal to the value pointed by it, we change the index to point at the current position, to make the second pointer keep valid.

Once implemented and tested that both indexes works properly, for the same test input signal as before, only 12536 linear searches were performed, which corresponds to the 5,57% of the input samples and it means a reduction of 13447 searches thanks to the use of two indexes instead of one. We have

detected that linear searches were triggered by the change in the shape of the input signal after a steep downslope, specifically that one in the QRS complex.

After the operations associated to a new sample in this first circular buffer are performed, the result of the erosion operation is always the value in the position pointed by the minimum index. While the buffer is receiving and processing new input samples, it always keep track of the minimum value in the 50 last samples, which corresponds to the $(f \ominus B_o)(n)$ erosion operation. By copying the value pointed by the minimum index to wherever it may be useful, the erosion operation is valid and efficient in terms of memory consumption -only a buffer of 50 elements is needed- and computation -only 5,57% of the values of the test input signal triggered a linear search.

To resume the operation within the circular buffer $B$, the minimum index *min*, the second minimum index *secmin* and the new position to be written *cur*, here is a brief description in pseudocode:

```
if cur == min then
        if newSample > B[min] then
                if secmin == -1 then
                        perform a linear search
                else if newSample > B[secmin] then
                        min = secmin
                        secmin = -1
                        end if
                end if
        end if
else if newSample <= B[min] then
        secmin = min
        min = cur
else if cur = secmin and newSample > B[secmin] then
        secmin = -1
else if secmin != -1 and newSample <= B[secmin] then
        secmin = cur
end if
```

The next operation to be performed for the technique to work is the dilation operation which corresponds to the first opening with the input signal and the structuring element. This dilation operation takes as input the result of the previous erosion operation and uses the same structuring element, $B_o$. As

it is a zero value horizontal line, this operation is similar to what has been explained before. The dilation operation for a new input sample means to add samples in the input to all the elements in the structuring array and return the greatest value obtained. As before, we need another circular buffer of 50 elements, corresponding to the length of the structuring element $B_o$.

Since there is no definition for the result of the first values of the input signal when $n < \frac{M-1}{2}$, this buffer is initialized with the input data until its half is reached. Once it is half full, it will have to wait for the first values of the result of the first erosion operation: at first, we must fill the first circular buffer and, then, we can fill this second circular buffer with the result of the first one. The dilation operation is based on sums and a search for a maximum element, so here it is needed to keep two indexes to the maximum value and the second maximum value in the buffer. The behaviour of these indexes are equal to those in the first buffer except for the maximum value they point at. While its initial filling is made, we pay attention to the new values to keep track of where the maximum values are stored.

In this buffer, the need of a second index for pointing at the maximum value was bigger than in the previous buffer. In the first implementation we did, without the use of a second pointer, for the same test input of 225000 samples, 77507 linear searches were made, corresponding to a 34,45% of the input data. Such an amount of searches justifies the additional comparison process by the use of a second pointer. As a result of this improvement, only 10255 linear searches were made, corresponding to a 4,56% of the input data. It is truly significant that for an amount of searches greater than those which were triggered in the previous buffer, the use of a second pointer reduces even more the final amount. This is explained according to the shape of the data provided as input in this dilation operation. At first, more linear searches are made but, with a second index, we get an greater reduction because of the lower number of invalidations of this second index.

This buffer cannot return any result until it is completely full. The previous circular buffer had a specific initialization process which granted that after 50 input samples it could be possible to get results from it. However, to get this dilation buffer working we have to wait the first buffer to fill and then this second buffer. 75 input samples are needed to initialize the first buffer: 25 of those 75 are going to initialize also this dilation buffer, another 25 elements for filling the first erosion buffer and then another 25 elements to fill this dilation buffer. These initialization processes make the technique unable to work in real time for the initial values, since the circular buffers are filling. As will be explained later, once all the buffers are filled the implementation provides

always a output sample for each input one.

The first opening operation, compound by an erosion and a dilation, is already implemented following the previous instructions. At this point, when the dilation buffer is filled, the value in the position pointed by the maximum index is the result of the dilation operation and, therefore, of the opening operation $f_o \circ B_o$. The next operation to be performed is a closing operation, compound by a dilation and an erosion, that uses as input the result of the previous opening operation and the structuring element $B_c$. After this closing operation we will get the baseline drift $f_b$ to be subtracted from the input signal.

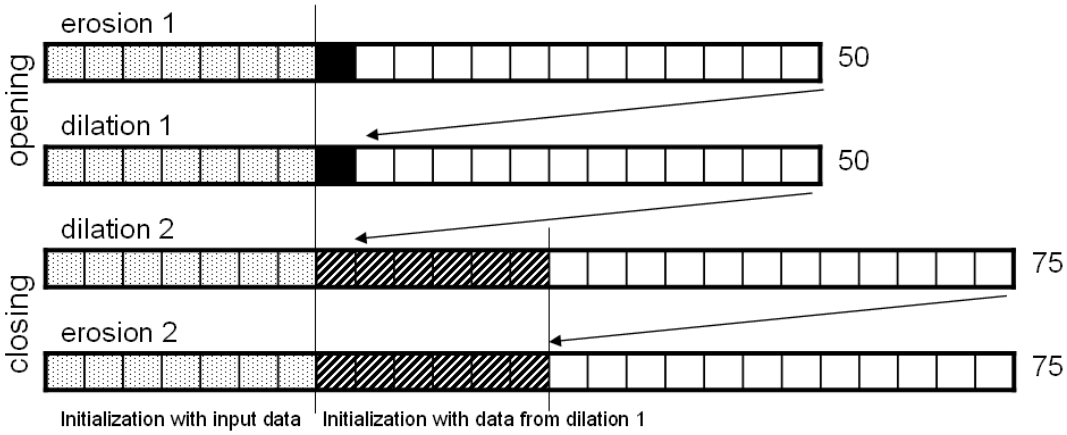
This closing operation uses $B_c$, which is a zero value horizontal line with 75 elements length and is compound by a dilation operation and an erosion operation which receives the result of the previous dilation. As before, we use a circular buffer to store the result of the previous dilation operation to perform this new first dilation. It is going to be 75 elements long as the structuring element. To initialize this buffer, the first 25 samples are taken from the input signal, since there is no definition for them in the morphological operators. For this buffer to keep filling, it has to wait until the two previous buffers are filled. Our implementation uses a control variable to check whether the previous dilation buffer is filled and, then, start copying its result data to this new dilation buffer. While it receives the first raw input data it keeps track of the values to keep the index to the maximum value stored updated.

When a new data sample is received, the procedure is equal to that in the previous buffers: check whether the new sample is going to overwrite a value pointed by any index, check whether the new value is greater than those maxima pointed and store the new sample. According to our tests, the dilation operation triggers more linear searches than the erosion operation. In this buffer, using the input signal of 225000 samples, 67492 linear searches were made in the implementation with an index, which corresponds to a 30% of the input data. However, by using a second index, the amount of linear searches is reduced incredibly to only 3419, a 1,52% of the input data. This reduction is explained by several factors. Firstly, the signal this operation receives as input has been previously processed by an opening operation so its shape is not as rich as it was at raw input. Secondly, the structuring element, $B_c$ is 50% longer than the element, $B_o$ used in the previous opening operation, which makes less possible to suffer from a index overwriting.

This dilation operation can start returning output samples when this dilation circular buffer is completely filled. Then, the value in the position pointed

by the maximum index is the result of the dilation operation. The baseline drift computation finishes with the last erosion operation, which also completes the closing operation with the $B_c$ structuring element. This last operation is implemented by a new circular buffer whose length is 75 elements, equal to the length of the structuring element used.

The initialization of this buffer is more complex than the process followed with the previous buffers. Once more, the first 25 elements of the input dataset are stored directly at the beginning of this buffer. The previous dilation buffer had to wait until the last buffer of the opening operation was completely filled. This last erosion buffer starts receiving values from that last buffer of the opening operation, since there are values before the half of the previous buffer that are going to be copied directly. For a better understanding, pay attention to the following image:




Scheme of the use of circular buffers for the implementation of the opening and closing operations and their initialization process

This buffer allows to perform a erosion operation. Since the structuring element $B_c$ is a zero value segment, there is no need to perform any calculation and the behaviour of this buffer is similar to the previous one. The erosion operation needs the minimum element in the array to be returned as a result. In the implementation without double index, for the test input signal of 225000 samples, 34416 linear searches were performed, corresponding to a 15,3% of the input dataset. By using a second minimum index, the number of linear searches is reduced to 3062, corresponding to a 1,36% of the input signal length. The reasons for this huge reduction are the same as before: the structuring element is longer, so the circular buffer is longer. This means the

chance of a index disabling are lower and, thus, the use of a second index is completely justified.

When this last circular buffer is completely filled, we are able to get values for the baseline drift estimate $f_b$. After processing a new input sample, performing the opening and closing operations, the value in the position pointed by the minimum index of this last erosion buffer is the new sample for the baseline drift. This value has to be subtracted to the input signal to get the filtered input signal without baseline wander. However, we have to remind that there is a gap between the input signal and the result of the filter because of the initialization process of the different buffers. While each buffer is being filled, no outcome is returned so the gap is increasing until all the buffers are full. To solve this, it is needed another buffer in which the input signal is stored. This new buffer helps to overcome the samples gap between the input and the output. It is also a circular buffer with two indexes: one to control which position is going to be written next with a new input sample and another index to control which position has the value next to be used for the final output.

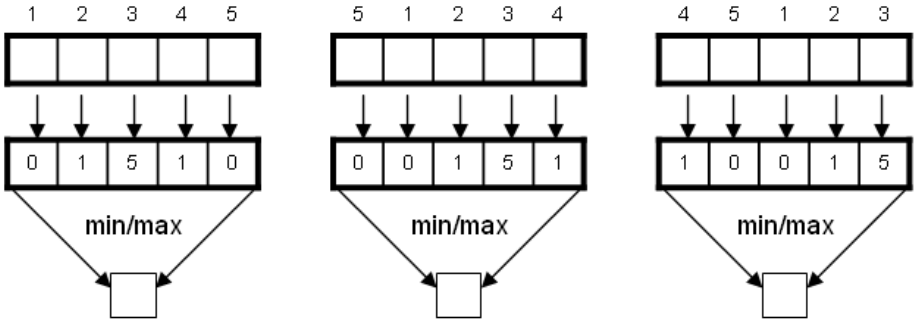
For each output value returned from the circular buffer for the last erosion operation, is has to be subtracted to the correspondent input data sample which is store in this recently described buffer. When the subtraction is performed, we have the baseline correction of the input signal $f_{bc} = f - f_b$.

The next step is to suppress noise in the signal. This is done by a concurrent closing and opening operation based on the use of two different structuring elements. These elements are five positions long so, as we have been doing, we will use a buffer of equal length. The first concurrent operation to be performed is an erosion and a dilation using $B_1 = (0, 1, 5, 1, 0)$. The approach to this calculation is different to what has been carried out before.

At first, we use a circular buffer of five elements to store the output of the baseline correction $f_{bc}$. This buffer is only five elements long because it is the length of the structuring element $B_1$. To carry out the first erosion and dilation, we use a static array of elements to store the offset of the $B_1$ for its use with the circular buffer. Since we can find the first element in order in the five elements buffer stored in any of its positions, we have to control the structuring element to match with the right element in the buffer. This is done by unfolding all the combinations of $B_1$, according to its starting point, in a bigger array of 20 elements: `(0, 1, 5, 1, 0, 0, 0, 1, 5, 1, ...)`.

Then, to calculate the first erosion and dilation operation we have only to add -dilation- or subtract -erosion- each value in the structuring element

to/from each value in the circular buffer. In case of the erosion operation, we will use a variable to store the minimum value obtained as a subtraction result and, likewise, we will use another variable to store the maximum value obtained as an add result. By accessing to these values once the buffer has been covered, we get the result of this new erosion and dilation operation.




Matching of the circular buffer and the correspondent instance of the $B_1$ structuring element in its array

We need another variable to control in which position of the structuring element array we have to start to calculate. When a new value is stored in the circular buffer, we have to get which is the maximum and the minimum value by the calculation with the structuring element. This is done by going through the circular buffer and add or subtract each of its values with the correspondent value of the $B_1$ array. We use a chain of comparisons to get the final maximum and minimum result and, to optimize the amount of changes in each control variable, an analysis was made to know in which relative positions were the maximum and minimum found.

The easiest implementation of this erosion and dilation operations would be to start by the first element in $B_1$ and calculate the result with the correspondent first element in the circular buffer. Then, process the second and so on. Since we are interested in an efficient implementation, we performed an analysis to check whether the maximum and the minimum in each operation can be found more probably in a fixed position, so that we could reduce the amount of comparison and changes of variables. Just by going through the array in a linear sequence, we cannot be sure of whether we are following the best path in the comparison chain.

The first implementation used a linear sequence to calculate with the structuring element by `buffer[0] + B1[j], buffer[1] + B1[j+1], ...`. At first, one solution to reduce the amount of changes of the value of the variables

would be to analyze in which positions the maximum and the minimum are found more often. It, obviously, should depend on the value of $j$, the variable which controls in which instance of the structuring element $B_1$ we are working on in the buffer `B1`. However, performing such an analysis for each instance and implementing this variable access method would lead to an extra cost of addressing, since the access to the array `B1` would depend, for optimization purposes, on another static array in which the best access chain for each instance would be stored: `buffer[0] + B1[opt[j]]`. This optimization would be nonsense since the extra addressing operation would lead to an extra computation and memory access cost.

The solution to avoid this extra cost while trying to perform such an optimization is not to consider in which instance of the $B_1$ we are working in the array `B1`. At first, we could consider that the maximum and minimum values are stored in whatever position so that it might be difficult to find a pattern for their locations. Thanks to the use of control variables, we discovered that, for the input test of 225000 samples, 295232 changes in the variable for the dilation operation were made and 271059 changes took place in the variable for the erosion dilation. This amount of changes corresponds to 1,31 changes per sample in the dilation and 1,16 changes per sample in the erosion dilation.

To try to reduce this figures, we performed an analysis using all the signals in the QT Database. The structuring element has five elements so we had only to check in which of these positions the maximum and the minimum were found while doing the calculation. Counting the changes for all the signals, for the sequence of accesses we tested, the sequence `(3, 4, 0, 1, 2)` resulted in 57596223 changes for the variable of the dilation operation and the sequence `(2, 3, 4, 0, 1)` gave as a result 57262531 changes for the variable of the erosion operation. We did not test all the combinations available since the first element of the sequence can be used to prune the search space for the best combination.

The best access sequence for the dilation operation is `(0, 2, 4, 1, 3)`, which reduces the amount of changes to 53035250 and, for the erosion operation, the best sequence is `(0, 2, 3, 1, 4)`, which reduces the amount of changes for all the input dataset to 53075361. Applying these sequences to the implementation and using the same input test, the result was that only 261362 changes were made -a 11,47% reduction and only 1,16 changes per sample- in the dilation operation. In the erosion operation, only 254809 changes took place -a 6% reduction for 1,13 changes per sample.

This optimization takes no extra computation cost since the implementa-

tion has to go through the hole array for the structuring element. Although the number of comparisons cannot be reduced, because we have to compare all the values, the number of changes in the variables of the morphological operations is reduced, leading to a better performance.

The code for the implementation of these concurrent erosion and dilation operations is needed in three different parts of the general code. The structuring element is only 5 elements long, so it fills completely in the first five input samples. While the first buffer for the erosion operation at the beginning of the implementation is filling, those values has to be stored in the first buffer for noise suppression since its length is of only five elements. Therefore, the noise suppression code starts returning results before the baseline correction is done.

While the buffer for the second erosion in the baseline correction is filling, those samples have to be also processed by the noise suppression part, since they produce no output for the baseline correction but are needed in this filtering process. Finally, once all the buffers are full, the code for this first concurrent operation remains needed to produce a right result.

These first erosion and dilation returns their result by the value of the variable whose changes we have tried to reduce. When the calculation is done by going through the array and getting the maximum and the minimum values, the associated variables store the result of those operations.

The second concurrent operation corresponding to the final erosion and dilation in the concurrent opening and closing is based on the use of the structuring element $B_2$. This stands for the completion of the noise suppression: we have already explained how to perform $f_{bc} \oplus B_1$ and $f_{bc} \ominus B_1$. The next step is to use the structuring element $B_2$ to perform a new pair of operations. $B_2$ is a zero value horizontal line with a length of five elements. The steps to follow for the implementation of this new concurrent operation are similar to those taken for the baseline removal part.

| Variable | Changes w/o opt. | Changes w/ opt. | Reduction |
|---|---|---|---|
| dilation (max.) | 295232 (1,31c/s) | 261362 (1,16c/s) | 11,47% |
| erosion (min.) | 271059 (1,2c/s) | 254809 (1,13c/s) | 6% |

Table 3.1: Changes made in the variable for each operation and changes per sample without and with optimization.

We use a circular buffer of five elements to support these operations and, since the structuring element has a constant zero value, we do not have to go through the buffer as we had to do with the use of $B_1$. We can use the same

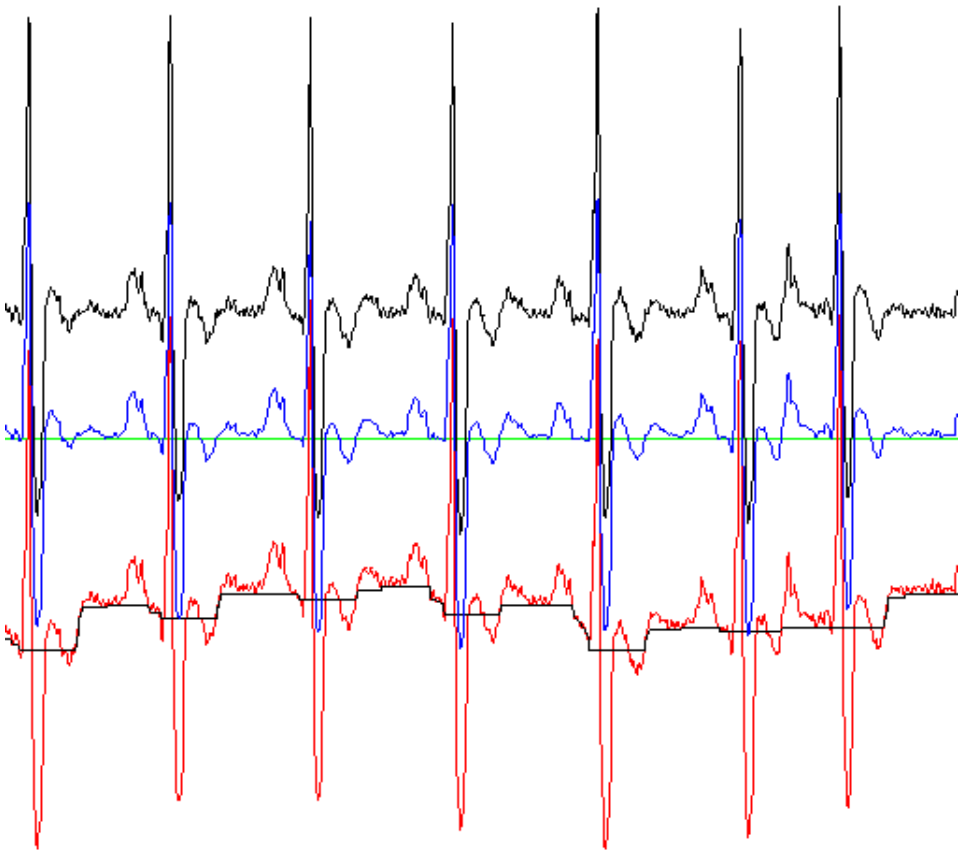
approach as before: control with an index were the maximum and the minimum value is stored. These new circular buffers have to be properly initialized by waiting until the previous buffers are full and start to give output. The code for this new circular buffers is next to the code of the previous ones, since these buffers depend on the output of those and, therefore, each sample processed by the previous compound operation entails a new operation in these new buffers.

One buffer is devoted to the erosion operation and the other to the dilation operation. Since these buffers are only five elements long one approach to its implementation would be to go through them to get the minimum or maximum value as output. However, this is more expensive in terms of computation than the use of an index to the appropriate value and check whether the new value to be stored changes the maximum or minimum value index. It would be also easy to think that there is no need to use a second index since the search in an array of five elements is rapid. The first implementation gave as a result that, for the dilation operation, 71090 linear searches were made, corresponding to the 31,6% of the input test data. In the erosion operation, 97650 linear searches were performed, which equals to a 43,4% of the input data.

This huge number of linear searches triggered by the overwriting of the value pointed by the index is explained by the length of the structuring element. As happened before, the shape of the input signal to these operations determines how many searches we can avoid by the use of a second pointer. In these operations, by using a second maximum and minimum index, the number of linear searches were reduced to 31844 -14,15% of the input data- in the dilation operation and to 45061 -20,03%- in the erosion operation. This last figure still shows a great number of linear searches. We tried to reduce this number by inserting the use of a third index but the extra computation cost was not justified by the slight reduction achieved.

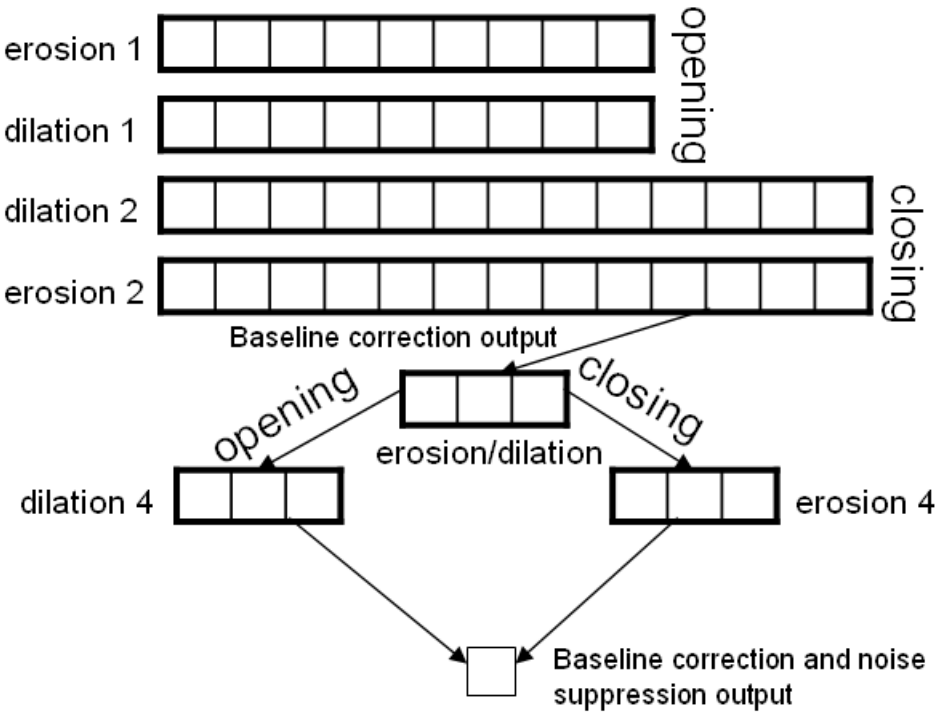
| Circular buffer | Num. lin. searches | Num. srch. w/ 2nd index | Reduction |
|---|---|---|---|
| erosion 1 | 25983 (11,55%) | 12536 (5,57%) | 51,75% |
| dilation 1 | 77507 (34,45%) | 10255 (4,56%) | 86,77% |
| dilation 2 | 67492 (30%) | 3419 (1,52%) | 94,93% |
| erosion 2 | 34416 (15,3%) | 3062 (1,36%) | 91,1% |
| dilation 4 | 71090 (31,6%) | 31844 (14,15%) | 55,21% |
| erosion 4 | 97650 (43,4%) | 45061 (20,03%) | 53,85% |

Table 3.2: Reduction of linear searches in the circular buffers due to the use of a second index for a test input of 225000 samples.



Baseline correction and noise suppression output (centered, in blue), baseline correction output (in black, at the top) and input signal and baseline estimation (below, in red and black).

Once the concurrent operations are performed, by accessing to the value pointed by the maximum and the minimum indexes in the last circular buffers we get the result of $f_{bc} \oplus B_1 \ominus B_2$ and $f_{bc} \ominus B_1 \oplus B_2$. To get the output of the filter after the noise suppression part, we have only to add those values and average them: $f = \frac{1}{2}(f_{bc} \oplus B_1 \ominus B_2 + f_{bc} \ominus B_1 \oplus B_2)$. The result of the each part of this filter is shown in the previous image.




Scheme of the circular buffers and the operations performed

In contrast to the cubic spline technique, this morphological filtering provides more stability. There is no estimator to use and, in fact, all the technique works the same for any point in any characteristic wave of the ECG signal. We have to keep in mind that the use of morphological operators are based on structuring elements chosen according to the shape of the signal we are trying to process. Thanks to the fact that the structuring elements $B_o$, $B_c$ and $B_2$ are a horizontal line of zero value, the implementation can be done efficiently with no extra cost for calculation with each element in those structuring arrays. Then, the main goal is to reduce the amount of linear searches. This technique is not artifact dependant since no artifact in the input signal can trigger any special condition in the filtering process.

# Chapter 4

# Results

This chapter presents the results for the validation process of these techniques. During the implementation process, all the validation tests were performed in a x86 platform without any constraint in terms of memory consumption and computation cost. The implementation was validated using the Physionet QT database (QTDB) [2] and the MIT-BIH Noise stress test database.

Firstly, both implementations were tested by visualizing its results in a Matlab scenario to test at first glance whether the implementation was working properly. For testing the morphological filtering technique, a previous offline implementation was done and then, while each morphological operation was implemented for realtime, its result were validating by the use of the offline implementation. This allowed us to work in parallel with this filtering. While the realtime implementation was being tested, others could work in a delineation algorithm.

As shown in the images of the previous chapter, several lines were added to the cubic spline implementation to check whether the knot locator was working as expected. The first steps taken in the implementation of this technique were, obviously, to test and to improve how the knot locator was fixing to the beats in the input signal. We found we needed an improvement to the locator due to the presence of some overlocation failures, as explained before. Once the locator was working properly, it was needed to implement the cubic spline technique.

The first implementation used floating point operations but, since the target embedded platform does not support this kind of operations, it was needed an adaptation to the use of integers. We found then needed to arithmetically left shift the cubic spline variables to keep it working since the integer implementation was not accurate enough. Then, the optimization process had the

goal to improve the efficiency of the implementation but we could only try to optimize the calculation of the cubic spline variables, using 32-bits operations instead of 16-bits in the embedded platform.

Since the cubic spline implementation uses dynamic memory to store the information and samples of three consecutive beats, we had to test whether there was any memory leak. The implementation uses several `malloc()` and `free()` instructions to work. In an embedded platform, the amount of memory is considerably small compared to that in a x86 platform and memory leaks should not take place. To check whether there was any memory leak or memory access failure, we used Valgrind [6]. It is an instrumentation framework for building dynamic analysis tools. It is a pack of six tools, from which we used memcheck. For a test signal of 225000 samples, the report of valgrind is

```
==3180== ERROR SUMMARY: 0 errors from 0 contexts
            (suppressed: 13 from 1)
==3180== malloc/free: in use at exit: 0 bytes in 0 blocks.
==3180== malloc/free: 4,494 allocs, 4,494 frees,
            1,587,466 bytes allocated.
==3180== For counts of detected errors, rerun with: -v
==3180== All heap blocks were freed -- no leaks are possible.
```

This report shows that no memory leaks took place and that 4494 memory blocks were needed for the removal of the baseline wander in that test.

The morphological filtering was at first implemented without the use of any optimization. Circular buffers had only one index to the correspondent maximum or minimum value. The implementation process took us more time than expected so we had to delay the optimizations until the hole first prototype was validated. Then, we performed some profiling tests to know how many linear searches were performed in each circular buffer and how many changes were made in some variables. This allowed us to carry out the changes explained in the previous chapter to get a more efficient implementation of that technique.

For a test signal of 225000 samples, the first implementation took 5,341sec to filter the hole input signal. Since the input signal is a recording of 15 minutes, this means that, on a x86 platform, all the samples were filtered in a 0,593% of the total time they represent. However, once the optimizations were made, the filter only took 4,801sec to filter the input signal, which means an improvement of 10,11% of the execution time and only 0,533% of the time represented by the input signal. The cubic spline implementation is more efficient in terms of execution time on a x86 platform. For the same input

signal, only 3,373sec were needed to filter the hole ECG, which corresponds with a 0,374% of the time the signal represents and an improvement of 29,74% of the execution time of the morphological filtering.

Despite these figures, we have to keep in mind that the morphological filtering carries out a noise suppression and that the cubic spline implementation is too artifact dependent and uses dynamic memory allocation. All of these figures have been obtained by a simulation on a x86 platform.

To validate the partial results obtained by the morphological implementation and the knot locator and the result of the cubic spline technique we have used several ECG signals from the Physionet QT database [2]. This database is designed to provide a set of signals to test the performance of several algorithms devoted to measure the width of characteristic waves in the ECG in order to detect abnormal electrical conduction in the heart, due to myocardial damage, and to stratify patients at risk of cardiac arrhythmias. The most important interval for such measurements is the QT segment. Several algorithms have been published to detect and measure that interval but all the studies lack of standardized databases containing a sufficiently large number of carefully annotated heartbeats with manually-made measurements of waveform boundaries. This reflects the tremendous effort required for a clinician to manually annotate a statistically significant set of QRST complexes. The QT database stands for addressing this problem by constructing an annotated reference database which includes a wide variety of ECG morphologies and a significant number of patient records.

The records were chosen primarily from among existing ECG databases, including the MIT-BIH Arrhythmia database [5], also used in this project. The QT database provides the same signals but with added reference annotations marking the location of waveform boundaries. The database contains a total of 105 fifteen-minute excerpts of two channel ECGs. Within each record, between 30 and 100 representative beats were manually annotated by cardiologists, who identified the beginning, peak and end of the P-wave, the beginning and end of the QRS-complex, and the peak and end of the T-wave. All records were sampled at 250Hz.

The records are provided in the MIT-BIH database format. Each record includes a signal file .dat, a header file .hea, describing the format of the signal file and several annotation files: .atr file contains the original annotations from the source database, .ari file contains QRS annotations obtained automatically by ARISTOTLE [7], a two lead ECG analysis program, and .man file contain the manual annotations.

To validate the results of the filtering process, we use the sel302 ECG from the QT database as a reference because it is very well defined without noise. There are several measures to test how a signal is conditioned by any operation on it. One way to test the techniques here presented could be to use a clean signal, add baseline wander and noise to it, filter it, and then calculate the *average root-mean square error* -RMS- which is a common measure to quantify the error between the original signal $x(n)$ and the filtered $\tilde{x}(n)$ for $n = \{0, 1, \ldots, N\}$.

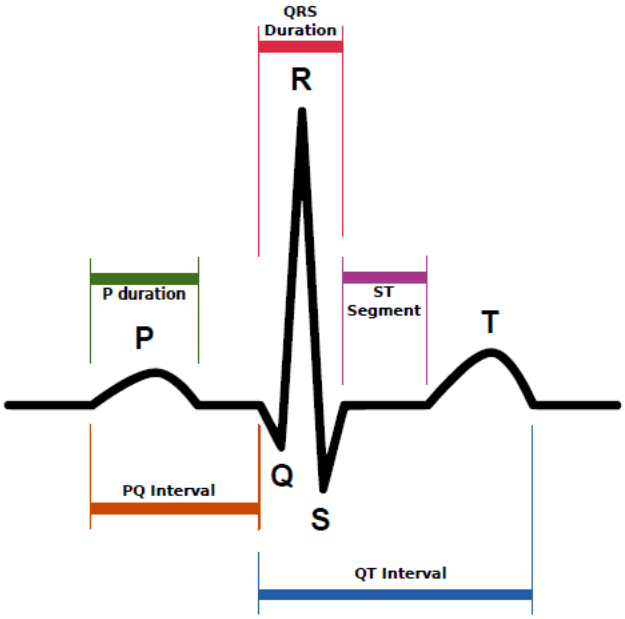
$$\rho_{RMS} = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}(x(n) - \tilde{x}(n))^2}$$

However, this is not a valid measurement for the baseline wander removal, since this process distorts the signal as expected. As can be seen in the previous images for the baseline wander removal, this process means a distortion of the input signal to get an isoelectric line of zero value. All the filtering processes distorts the signal and, in some cases, flattens their peaks. The baseline wander removal process means that all the characteristic waves remains in the same time index but it could be possible for a signal to be reduced the amplitude of its peaks. This makes the difference between the original signal $x(n)$ and the filtered one $\tilde{x}(n)$ increase and, therefore, the $RMS$ value is not acceptable. On the other hand, the baseline wander estimation is not a straight line between beats. For the cubic spline technique, it is obvious that some variation could be inserted in the filtered signal because of the ordinal value of the knot associated to a beat. For the morphological filtering, as can be seen, even when the beat seems to be centred without baseline, the estimation makes the signal move after the filtering process and, therefore, more error is introduced in the calculation of $RMS$.

To measure the results of these filtering techniques, we are going to use a wavelet based ECG delineation algorithm by Nicholas Boichat [10]. This delineation algorithm is based on the wavelet transform which was first presented in [18] and developed in [19]. The delineation process takes advantage of the fact that the ECG is roughly a periodic signal, and each beat is composed of a QRS complex, preceded by a P wave, and followed by a T wave. Each of these waves has a different frequency content -the QRS complex is made by relatively high frequencies, while the P and T waves are composed of low frequencies.

The ECG signal can be well decomposed using a dyadic discrete wavelet transform which provide us with outputs called scales. Each scale matches

different frequency bands of the original ECG signal, which allows us to perform a multi-scale analysis to detect the ECG waves. Once the outputs of the discrete wavelet transform are computed, it is needed to detect the main peak of the QRS complex, then perform the delineation of the QRS complex, consisting of detecting its secondary peaks, and find the onset and end of the complex.




ECG beat with its characteristic waves

By paying attention to the manual annotations provided in the QT Database, we can validate the result of the delineation algorithm. This algorithm provides as a result the detection of the characteristic waves in the ECG signal so that we can compare this automatic annotation with the manual annotation provided for each signal.

A manual annotation is considered as related to an automatic one if their time interval is smaller than 320ms. This pair of annotations is marked as a true positive $T_P$. Each manual annotation that has no corresponding automatic annotation is marked as a false negative $F_N$ and each automatic annotation without a manual annotation is marked as a false positive $F_P$.

Using these definitions, the sensitivity $S_e$ of the delineation and the positive predictivity $P^+$ is defined as follows:

$$S_e = \frac{T_P}{T_P + F_N}$$

$$P^+ = \frac{T_P}{T_P + F_P}$$

However, since the QT database is manually annotated, it is possible to find an automatic annotation when there is no manual one. This absence of any annotation can mean either the wave is not present -the automatic detection counts as a false positive- or that the cardiologist could not annotate the point with confidence. Therefore, an automatic detection does not necessarily mean a false positive. Because of this, the tables will show the value of $P^+_{min}$, which has to be considered as a lower bound of the real value of $P^+$.

The great implementation of the delineation algorithm by Nicholas Boichat takes advantage of the fact that the QT database provides recordings from two leads for each ECG, and the manual annotations have been performed by cardiologists having a look at both leads. Therefore, for a best comparison between the automatic and manual delineation, the algorithm was improved to run on both leads and then, for each manual annotation, the lead introducing the least error is considered. This is part of the ongoing work to get a multi-lead delineation algorithm. These filtering techniques are designed to serve as an input filter for that delineation implementation. That is the reason to use the wavelet delineation algorithm as a test for the validation of the implementations here presented.

To validate the results provided by these filtering techniques, we will get the delineation validation of the sel302 signal from the QT Database. Then, we will add noise to this signal, filter it and then provide the filtered signal as input for the delineation algorithm. By paying attention to the results of the delineation validation, we will be able to check how the filtering process affects the delineation algorithm.

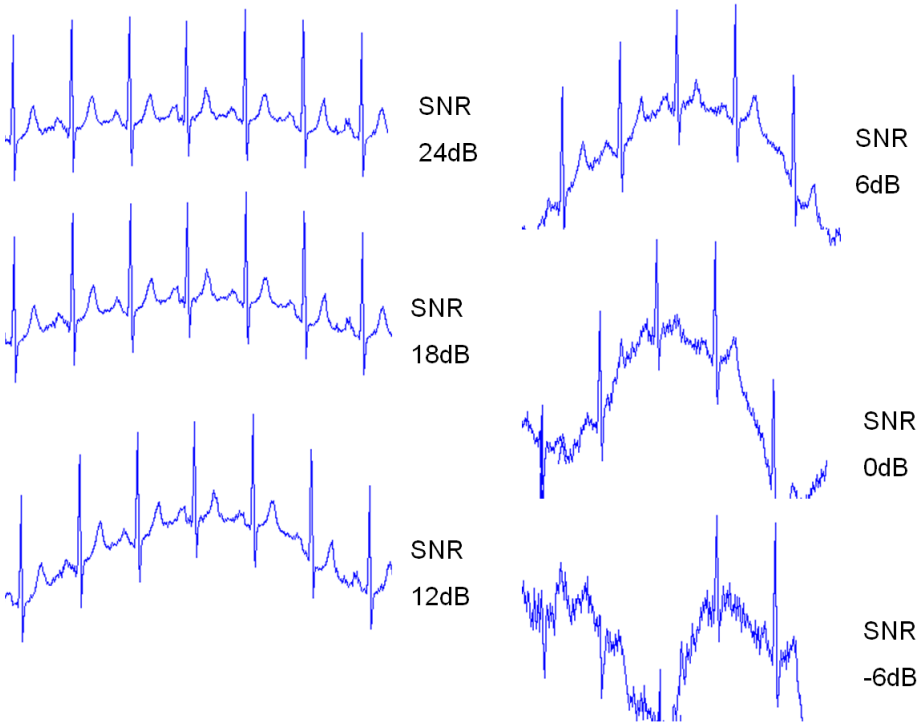
The MIT-BIH Noise stress test database [5] includes 12 half-hour ECG recordings and 3 half-hour recordings of noise typical in ambulatory ECG recordings. The noise recording were made using physically active volunteers and standard ECG recorders, leads and electrodes. These electrodes were placed on the limbs in positions in which the subjects' ECGs were not visible. The three noise records were then assembled from the recordings by selecting intervals that cointained predominantly baseline wander -records `bw`-, muscle artefact -`ma`- and electrode motion artefact -`em`-.

To prepare the test workbench, we use the PhysioToolkit Software [9], a large library of software for physiologic signal processing and analysis. Using `nst`, we introduce noise and baseline wander in the input test signal. In the MIT-BIH Noise stress database several noise recordings are provided. The first

analysis to perform is baseline wander removal, so we used the baseline wander recording and `nst` to modify the test signal. `nst` works as follows, by receiving a input signal, a reference to a noise recording, the name for the output signal and then the SNR value we desire.

```
nst -i <signal> bw -o <outputSignal> -s <SNR>
```

For our tests, we have used several SNR values to test the performance of these techniques. As will be shown, values 24, 18, 12, 6, 0, -6 for SNR show different results of the filtering techniques regarding its response to the SNR value. Thanks to this, we get six input ECG signals with different SNR value of baseline wander. `nst` generates an output in which there is a five-minute noise-free learning period, followed by two-minute periods of noisy and noise-free signals alternately until the end of the clean record. The gains to be applied during the noisy periods are determined by measuring the signal and noise amplitudes. The manual annotations for the signal `sel302` are available for the samples between 151105 and 156072, which are in the second two minutes noise period introduced by `nst`, so that we can use this signal to perform a test in which the signal in that period will be affected by the baseline wander injection and then filtered.




Result of baseline wander addition to signal `sel302` using `nst` for each SNR value.

Once we have the input signals for all values of SNR we can proceed with the validation test. At first, we need to get the delineation validation for the input signal without any noise. Then, we will get the validation of the delineation process for each filtered test signal. These validation results are provided by measuring the sensitivity $S_e$, the lower bound of positive predictivity $P^+_{min}$ and the mean $m$ of the error between the manual annotation and the automatic annotation obtained by the delineation process in msec. For each point in the delineation process it is also provided the standard deviation $s$ of the errors.

The CSE working party [11] provides several two-standard deviation tolerances for the annotation of the characteristic waves in the ECG, which correspond to measurement differences between cardiologists. The process to evaluate whether the filtering process is valid is based on paying attention to the result of the standard deviation obtained by the delineation algorithm and checking whether it is in the boundaries defined by the CSE working party. In the following tables, it is shown at first the delineation validation for the input `sel302` clean signal. Then, for each SNR, the same validation values are provided to compare. The last row of each table shows the tolerance values for the standard deviation $s$.

| | Param | $P_{on}$ | $P_{peak}$ | $P_{end}$ | $QRS_{on}$ | $QRS_{end}$ | $T_{peak}$ | $T_{end}$ |
|---|---|---|---|---|---|---|---|---|
| `sel302` | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 96.67 | 96.67 |
| clean | $P^+_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| signal | $m \pm s$ (ms) | $-0.7 \pm 8.2$ | $8.0 \pm 8.8$ | $-6.9 \pm 8.9$ | $0.3 \pm 5.5$ | $17.7 \pm 34.6$ | $14.6 \pm 5.1$ | $4.4 \pm 8.6$ |
| `sel302` | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 96.67 | 96.67 |
| SNR | $P^+_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 24dB | $m \pm s$ (ms) | $-1.2 \pm 6.1$ | $11.1 \pm 6.6$ | $8.7 \pm 4.7$ | $3.7 \pm 5.3$ | $22.0 \pm 35.3$ | $18.6 \pm 5.1$ | $11.3 \pm 7.4$ |
| `sel302` | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 96.67 | 96.67 |
| SNR | $P^+_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 18dB | $m \pm s$ (ms) | $-1.9 \pm 5.8$ | $11.5 \pm 6.6$ | $10.0 \pm 4.8$ | $2.9 \pm 5.5$ | $22.0 \pm 35.3$ | $18.5 \pm 5.1$ | $11.2 \pm 7.4$ |
| `sel302` | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 90.00 | 90.00 |
| SNR | $P^+_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 12dB | $m \pm s$ (ms) | $-2.9 \pm 7.4$ | $12.8 \pm 6.4$ | $11.1 \pm 5.3$ | $2.7 \pm 5.3$ | $31.2 \pm 54.1$ | $18.2 \pm 5.3$ | $9.5 \pm 7.8$ |
| `sel302` | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 80.00 | 80.00 |
| SNR | $P^+_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 6dB | $m \pm s$ (ms) | $-4.9 \pm 10.0$ | $13.1 \pm 6.4$ | $10.1 \pm 6.9$ | $2.0 \pm 6.1$ | $46.5 \pm 72.8$ | $18.2 \pm 4.8$ | $8.7 \pm 8.1$ |
| `sel302` | Se (%) | 96.67 | 96.67 | 96.67 | 100.00 | 100.00 | 63.33 | 63.33 |
| SNR | $P^+_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 0dB | $m \pm s$ (ms) | $-7.2 \pm 24.7$ | $9.2 \pm 24.2$ | $3.3 \pm 27.9$ | $-0.8 \pm 12.2$ | $69.7 \pm 84.3$ | $18.1 \pm 5.4$ | $5.7 \pm 8.9$ |
| `sel302` | Se (%) | 93.33 | 93.33 | 93.33 | 100.00 | 100.00 | 40.00 | 40.00 |
| SNR | $P^+_{min}$ (%) | 70.00 | 70.00 | 70.00 | 66.67 | 66.67 | 92.31 | 92.31 |
| -6dB | $m \pm s$ (ms) | $-8.4 \pm 41.4$ | $-4.1 \pm 41.1$ | $-13.9 \pm 45.6$ | $-10.7 \pm 34.3$ | $88.1 \pm 84.0$ | $-10.0 \pm 42.4$ | $-22.0 \pm 49.7$ |
| Tolerances for $s$ | | 10.2 | - | 12.7 | 6.5 | 11.6 | - | 30.6 |

Table 4.1: Validation of the delineation of the signal `sel302` with different baseline wander SNR. Using **morphological filtering** -baseline wander & noise suppression.

The previous table shows the results for morphological filtering with base-

line wander removal and noise suppression. Except for the $QRS_{end}$ standard deviation values, all the results are good for SNR from 24dB to 6dB. For $P_{on}$, this filtering increases the mean for any SNR value but it decreases the deviation value with regard to that in the clean signal for the first SNR values. $P_{end}$ values are also acceptable in terms of standard deviation, although a mean increase takes place too. The $QRS_{on}$ values for standard deviation are almost the same as those obtained for the clean signal whereas the $QRS_{end}$ values are worst. This can be explained by paying attention to the value obtained for the clean signal, since it triples the tolerance bound. Finally, although the mean error is higher than that for the clean signal results, all the deviation results are acceptable for the $T_{end}$ delineation.

The sensitivity and predictivity values remains almost the same and under an acceptable limit until SNR = 0dB, which is a SNR value in which the input signal is considerably distorted as shown in a previous image. The conclusion for the validation results of the morphological filtering is that although it increases the mean error between the manual and the automatic annotations performed by the delineation algorithm, all the standard deviation values are acceptable until the 0 dB level for SNR is reached. Sensitivity and predictivity is not affected except for those values corresponding to the $T$ wave.

| | Param | $P_{on}$ | $P_{peak}$ | $P_{end}$ | $QRS_{on}$ | $QRS_{end}$ | $T_{peak}$ | $T_{end}$ |
|---|---|---|---|---|---|---|---|---|
| `sel302` | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 96.67 | 96.67 |
| clean | $P^{+}_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| signal | $m \pm s$ (ms) | $-0.7 \pm 8.2$ | $8.0 \pm 8.8$ | $-6.9 \pm 8.9$ | $0.3 \pm 5.5$ | $17.7 \pm 34.6$ | $14.6 \pm 5.1$ | $4.4 \pm 8.6$ |
| `sel302` | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 96.67 | 96.67 |
| SNR | $P^{+}_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 24dB | $m \pm s$ (ms) | $0.1 \pm 6.2$ | $11.1 \pm 6.6$ | $8.7 \pm 5.6$ | $3.7 \pm 5.3$ | $21.2 \pm 35.4$ | $18.6 \pm 5.1$ | $10.6 \pm 7.1$ |
| `sel302` | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 96.67 | 96.67 |
| SNR | $P^{+}_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 18dB | $m \pm s$ (ms) | $-0.3 \pm 6.3$ | $12.3 \pm 7.4$ | $10.1 \pm 5.9$ | $3.3 \pm 5.3$ | $20.7 \pm 35.4$ | $18.3 \pm 5.2$ | $10.2 \pm 7.7$ |
| `sel302` | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 96.67 | 96.67 |
| SNR | $P^{+}_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 12dB | $m \pm s$ (ms) | $-1.2 \pm 6.3$ | $12.8 \pm 7.5$ | $10.9 \pm 5.9$ | $3.1 \pm 5.3$ | $19.6 \pm 35.7$ | $19.0 \pm 5.0$ | $9.5 \pm 7.5$ |
| `sel302` | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 90.00 | 90.00 |
| SNR | $P^{+}_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 6dB | $m \pm s$ (ms) | $-1.9 \pm 6.9$ | $13.3 \pm 8.3$ | $10.5 \pm 7.9$ | $0.1 \pm 9.4$ | $28.8 \pm 53.6$ | $18.8 \pm 5.0$ | $7.4 \pm 8.6$ |
| `sel302` | Se (%) | 96.67 | 96.67 | 96.67 | 100.00 | 100.00 | 70.00 | 70.00 |
| SNR | $P^{+}_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 0dB | $m \pm s$ (ms) | $-3.7 \pm 25.2$ | $3.9 \pm 28.3$ | $0.6 \pm 23.6$ | $-10.3 \pm 11.9$ | $59.2 \pm 80.5$ | $19.0 \pm 5.9$ | $3.4 \pm 8.8$ |
| `sel302` | Se (%) | 86.67 | 86.67 | 86.67 | 100.00 | 100.00 | 40.00 | 40.00 |
| SNR | $P^{+}_{min}$ (%) | 83.87 | 83.87 | 83.87 | 73.17 | 71.43 | 100.00 | 100.00 |
| -6dB | $m \pm s$ (ms) | $-67.5 \pm 52.1$ | $-62.5 \pm 57.1$ | $-76.3 \pm 60.0$ | $-12.0 \pm 27.5$ | $78.5 \pm 88.8$ | $22.3 \pm 26.3$ | $-3.3 \pm 26.2$ |
| Tolerances for $s$ | | 10.2 | - | 12.7 | 6.5 | 11.6 | - | 30.6 |

Table 4.2: Validation of the delineation of the signal `sel302` with different baseline wander SNR. Using **morphological filtering** -only baseline wander removal.

The results for only the baseline wander removal performed by the morphological filtering are almost the same as those obtained by the hole filtering. In this case, it is remarkable that the mean error does not increase as much as does with the noise suppression part. The standard deviation of the error remains lower than the value obtained by the hole filtering except for the $QRS_{on}$ values at 6dB SNR. For the $P_{on}$ point, this filtering increases the accuracy of the delineation for SNR = 24dB and 18dB. Regarding the sensitivity and predictivity values, this filtering obtains the same results as those for the clean signal for SNR values from 24dB to 12dB and for the $T$ wave, those values are better than those obtained with the noise suppression part. However, the results for the SNR values of 0dB and -6dB are better with noise suppression since the signal is considerably distorted.

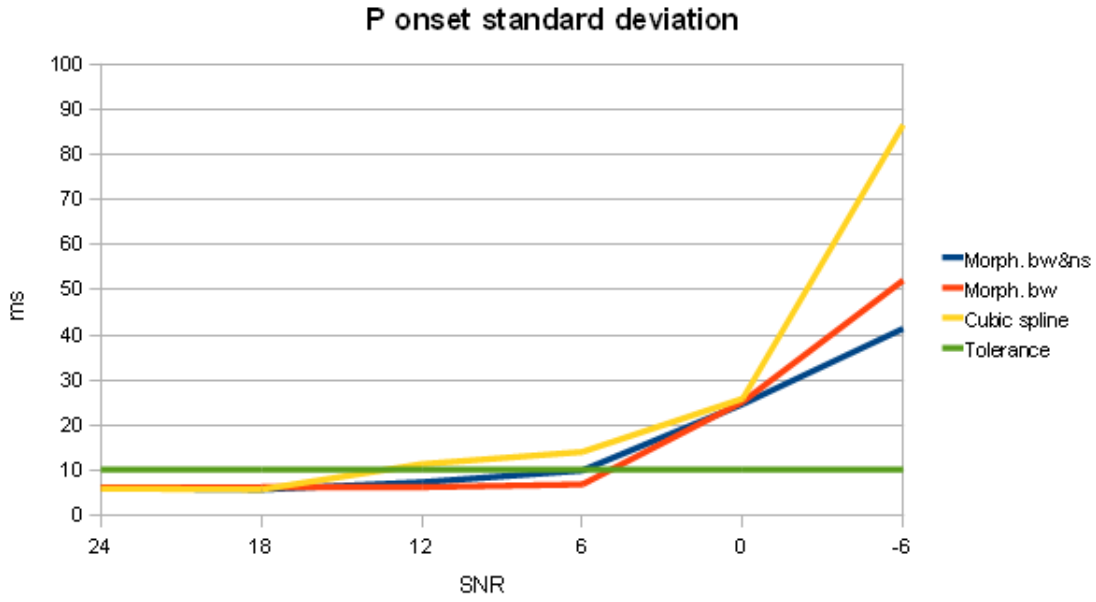
| | Param | $P_{on}$ | $P_{peak}$ | $P_{end}$ | $QRS_{on}$ | $QRS_{end}$ | $T_{peak}$ | $T_{end}$ |
|---|---|---|---|---|---|---|---|---|
| sel302 | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 96.67 | 96.67 |
| clean | $P^{+}_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| signal | $m \pm s$ (ms) | $-0.7 \pm 8.2$ | $8.0 \pm 8.8$ | $-6.9 \pm 8.9$ | $0.3 \pm 5.5$ | $17.7 \pm 34.6$ | $14.6 \pm 5.1$ | $4.4 \pm 8.6$ |
| sel302 | Se (%) | 96.67 | 96.67 | 96.67 | 96.67 | 96.67 | 90.00 | 90.00 |
| SNR | $P^{+}_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 24dB | $m \pm s$ (ms) | $0.8 \pm 6.0$ | $11.9 \pm 7.2$ | $13.9 \pm 5.7$ | $2.6 \pm 5.2$ | $19.6 \pm 36.4$ | $18.7 \pm 5.2$ | $13.9 \pm 7.3$ |
| sel302 | Se (%) | 93.33 | 93.33 | 93.33 | 96.67 | 96.67 | 90.00 | 90.00 |
| SNR | $P^{+}_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 18dB | $m \pm s$ (ms) | $0.7 \pm 5.8$ | $11.9 \pm 7.4$ | $13.6 \pm 6.4$ | $2.8 \pm 5.2$ | $19.6 \pm 36.4$ | $18.5 \pm 5.3$ | $13.3 \pm 7.7$ |
| sel302 | Se (%) | 96.67 | 96.67 | 96.67 | 96.67 | 96.67 | 90.00 | 90.00 |
| SNR | $P^{+}_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 12dB | $m \pm s$ (ms) | $-1.1 \pm 11.4$ | $12.0 \pm 11.0$ | $10.3 \pm 14.1$ | $2.9 \pm 5.6$ | $19.7 \pm 36.4$ | $19.0 \pm 5.4$ | $11.4 \pm 8.6$ |
| sel302 | Se (%) | 93.33 | 93.33 | 93.33 | 96.67 | 96.67 | 70.00 | 70.00 |
| SNR | $P^{+}_{min}$ (%) | 82.35 | 82.35 | 82.35 | 72.50 | 72.50 | 95.45 | 95.45 |
| 6dB | $m \pm s$ (ms) | $1.7 \pm 14.1$ | $10.7 \pm 14.2$ | $6.4 \pm 18.5$ | $0.4 \pm 8.4$ | $29.9 \pm 53.7$ | $11.6 \pm 24.1$ | $3.4 \pm 29.2$ |
| sel302 | Se (%) | 90.00 | 90.00 | 90.00 | 96.67 | 96.67 | 63.33 | 63.33 |
| SNR | $P^{+}_{min}$ (%) | 84.38 | 84.38 | 84.38 | 76.32 | 76.32 | 95.00 | 95.00 |
| 0dB | $m \pm s$ (ms) | $4.9 \pm 25.9$ | $9.3 \pm 27.5$ | $5.2 \pm 22.9$ | $-7.6 \pm 18.5$ | $51.7 \pm 74.8$ | $11.4 \pm 47.8$ | $-0.8 \pm 62.4$ |
| sel302 | Se (%) | 70.00 | 70.00 | 70.00 | 83.33 | 90.00 | 43.33 | 46.67 |
| SNR | $P^{+}_{min}$ (%) | 84.00 | 84.00 | 84.00 | 75.76 | 81.82 | 92.86 | 93.33 |
| -6dB | $m \pm s$ (ms) | $-51.6 \pm 86.5$ | $-55.8 \pm 76.8$ | $-63.4 \pm 72.5$ | $-16.3 \pm 46.3$ | $59.3 \pm 79.7$ | $26.8 \pm 48.7$ | $17.1 \pm 63.1$ |
| Tolerances for $s$ | | 10.2 | - | 12.7 | 6.5 | 11.6 | - | 30.6 |

Table 4.3: Validation of the delineation of the signal sel302 with different baseline wander SNR. Using **cubic spline technique**.

The results for the cubic spline technique are slightly worse than those obtained with both tests with the morphological filtering. For the $P_{on}$ parameter, only SNR values of 24dB and 18dB result in a deviation within the tolerance limits. For $P_{on}$ and $P_{end}$ only the results obtained with SNR = 24dB and 18dB are acceptable, since for SNR = 12dB, the standard deviation exceeds the tolerance limit. For the $QRS_{on}$ values, the result of filtering the input signal with 6dB SNR has its standard deviation outside the tolerance bounds. As before, the $QRS_{end}$ values triple the tolerance values and, for the $T_{end}$ standard de-

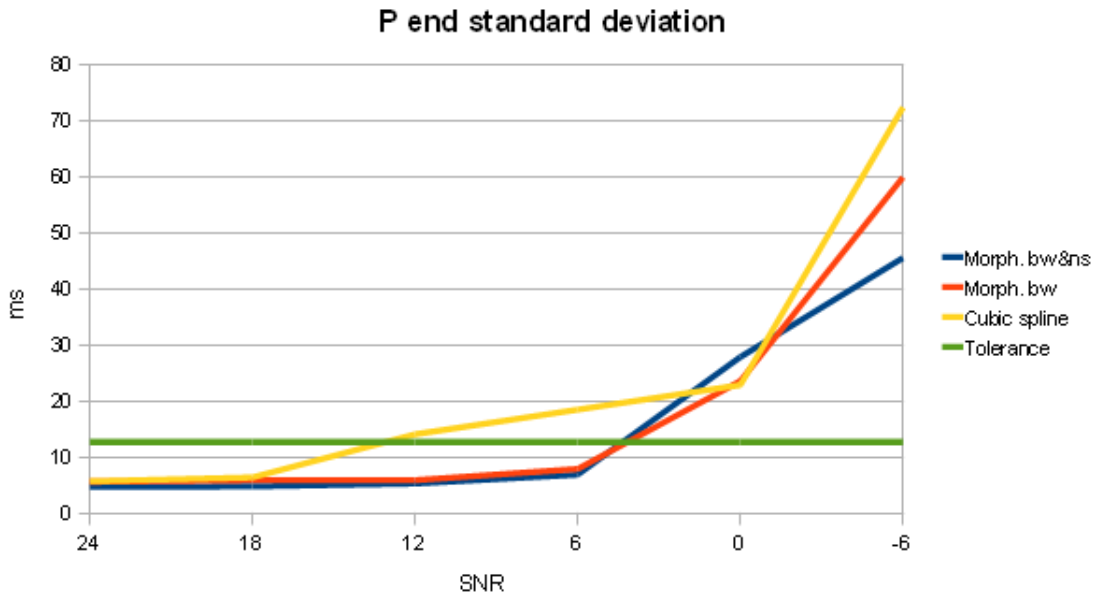
viation, the SNR = 6dB is next to the tolerance limit. Furthermore, there is a decrease in the predictivity and sensitivity values related to the SNR value. This is explained by the presence of several false positives and false negatives in the automatic detection procedure.

The next step is to compare the results obtained for each filtering technique. As follows, we will compare the standard deviation for each characteristic point in the ECG according to each implementation, considering morphological filtering with baseline wander removal and noise suppression and only the baseline wander removal filter.




$P_{onset}$ standard deviation for each SNR value and technique used.

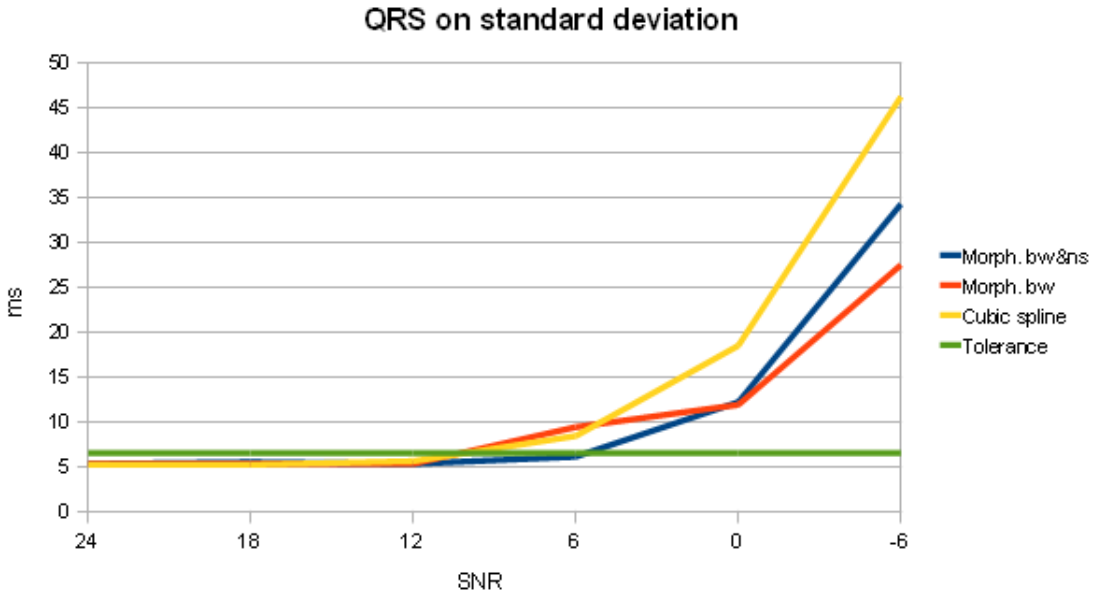
The $P_{onset}$ point offers good standard deviation values for SNR = 24dB and 18dB with each implementation. However, the cubic spline technique behaves poorly from a SNR value of 12dB, exceeding the tolerance limit. The morphological filtering with noise suppression results in worse standard deviation than only removing baseline wander. Until the SNR value of 6dB, both implementations of the morphological filtering result in an appropriate value for the standard deviation.




$P_{end}$ standard deviation for each SNR value and technique used.

The standard deviation for the annotation of the $P_{end}$ point is almost the same for both morphological filtering implementations. However, the cubic spline technique offers worse results since for SNR = 12dB, it exceeds the tolerance limits. It is remarkable that for SNR values from 24dB to 12dB, the morphological filtering results are almost the same for this point and only increases slightly when injecting a SNR of 6dB.

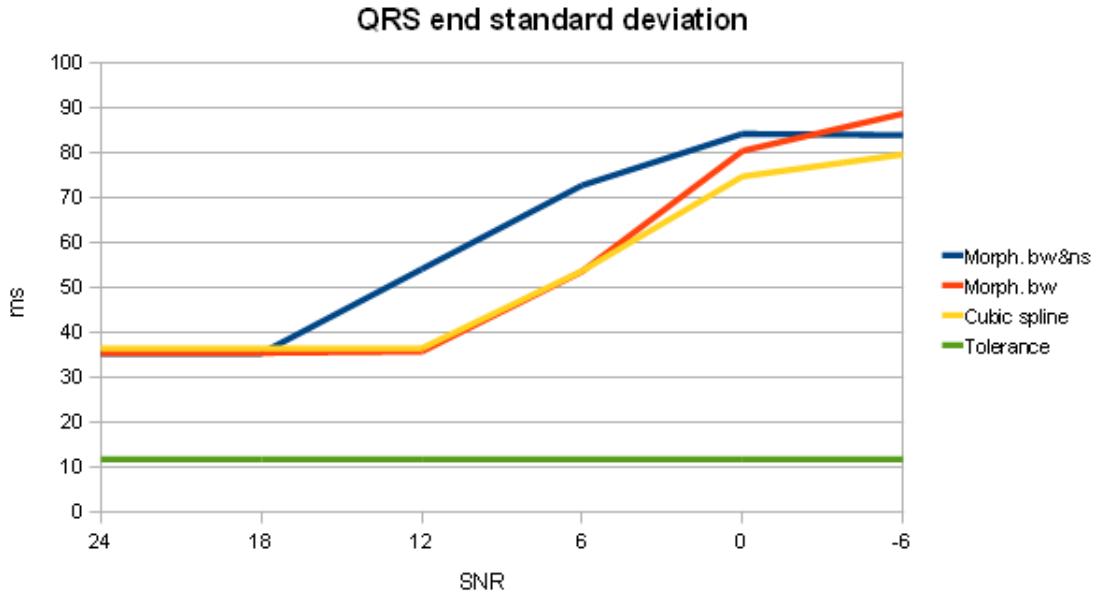
The $QRS_{on}$ standard deviation result is almost the same for the three implementations for a SNR value from 24dB to 12dB. For SNR = 6dB, the morphological filtering with baseline wander only behaves slightly worse than the cubic spline implementation whereas the complete morphological filtering implementation results in acceptable deviation values.



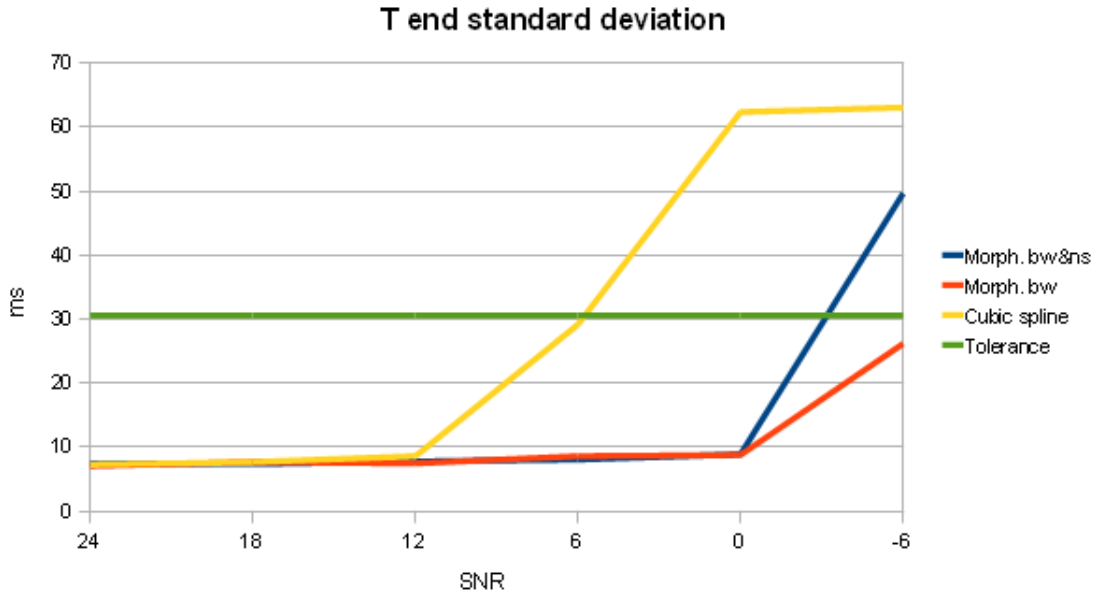

$QRS_{onset}$ standard deviation for each SNR value and technique used.

The $QRS_{end}$ point is not well delineated for any SNR value. This high value for standard deviation means those points for all the input signal are not well automatically delineated and, what is worse, the error between the manual and the automatic annotation is not constant. However, it is useful for us to state that the morphological filtering with baseline wander removal and noise suppression results are not so good as they are with other characteristic waves. Furthermore, the cubic spline and morphological filtering baseline wander removal behaves almost the same until SNR = 0dB.

Finally, the standard deviation for the annotation of the $T_{end}$ point is fine using both implementations of the morphological filtering until a SNR value of 0dB. However, the cubic spline technique results for SNR = 6dB are just in the tolerance limit. Once more, both implementations of the morphological filtering behaves the same until SNR = 0 dB, in which the input signal is considerably distorted in comparison to the original one.



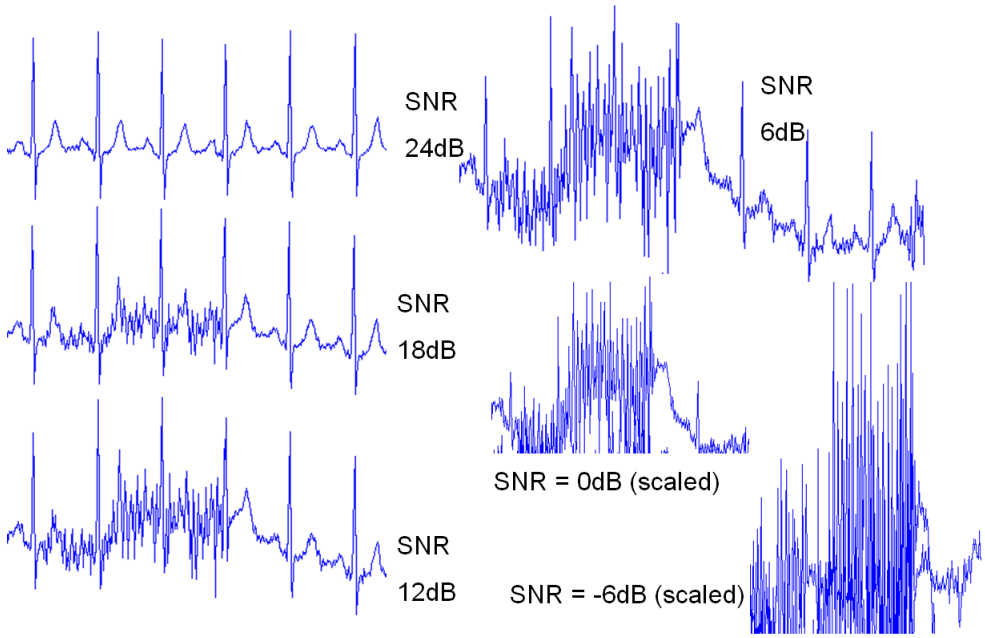

$QRS_{end}$ standard deviation for each SNR value and technique used.




$T_{end}$ standard deviation for each SNR value and technique used.

To test the noise suppression part of the morphological filtering we are going to inject using `nst` muscle artefact noise in the signal `sel302`. This corresponds to electromyographic noise, presented at the beginning of this report. The muscle artifact noise recordings are also available in the MIT-BIT noise stress database [5]. We are going to test the filtering for the same

SNR values as before and follow the same steps: compare the validation of the delineation values after filtering each input signal with those obtained with the clean signal.




Result of muscle artifact addition to the signal `sel302` using `nst` for each SNR value.

The results of the delineation for each SNR value are shown in the following table. For the delineation of the $P_{on}$ point, only 24dB for SNR results in an acceptable standard deviation since the value for 18dB exceeds the tolerance bound. The delineation of the $P_{peak}$ behaves almost the same for SNR from 24dB to 12dB, but increasing a 50% the mean of the errors. The $P_{end}$ point is well delineated after filtering noise until a value of 18dB for SNR. The delineation of $QRS_{on}$ is not accurate as for the baseline wander correction since it doubles the standard deviation in the first 24dB result. For the end of the $QRS$ complex, its delineation is difficult after filtering the noise since it always exceeds the tolerance level and the mean of the error is increased considerably. Finally, several delineation problems happen with the $T$ wave since for SNR = 12dB no delineation is possible (there are no true positives $T_P$). Furthermore, the delineation of the $T$ wave was good for the baseline wander removal but, after noise suppression, the mean of the error is considerably high and the standard deviation exceeds its bound for SNR = 6dB.

The poor performance for this noise suppression filtering in removing electromyographic noise for low SNR values could be explained by the fact that,

although morphological filtering was designed and tested by a Gaussian noise injection, which is useful for electromyographic noise simulation [21], the abnormal shape of the input signal when noise is injected leads to a poor delineation performance. For those values of SNR in which the noise does not distorts the input signal severely, the performance is acceptable.

| | Param | $P_{on}$ | $P_{peak}$ | $P_{end}$ | $QRS_{on}$ | $QRS_{end}$ | $T_{peak}$ | $T_{end}$ |
|---|---|---|---|---|---|---|---|---|
| sel302 | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 96.67 | 96.67 |
| clean | $P^{+}_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| signal | $m \pm s$ (ms) | $-0.7 \pm 8.2$ | $8.0 \pm 8.8$ | $-6.9 \pm 8.9$ | $0.3 \pm 5.5$ | $17.7 \pm 34.6$ | $14.6 \pm 5.1$ | $4.4 \pm 8.6$ |
| sel302 | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 86.67 | 86.67 |
| SNR | $P^{+}_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 24dB | $m \pm s$ (ms) | $-2.1 \pm 8.1$ | $12.7 \pm 8.9$ | $9.3 \pm 6.9$ | $1.3 \pm 10.6$ | $33.6 \pm 50.6$ | $18.6 \pm 5.5$ | $12.2 \pm 7.6$ |
| sel302 | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 80.00 | 80.00 |
| SNR | $P^{+}_{min}$ (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 18dB | $m \pm s$ (ms) | $-2.3 \pm 11.4$ | $12.8 \pm 9.7$ | $9.5 \pm 8.1$ | $-1.9 \pm 11.6$ | $48.9 \pm 70.5$ | $23.5 \pm 23.4$ | $13.5 \pm 17.7$ |
| sel302 | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 0.00 | 0.00 |
| SNR | $P^{+}_{min}$ (%) | 55.56 | 55.56 | 55.56 | 38.46 | 38.96 | NaN | NaN |
| 12dB | $m \pm s$ (ms) | $1.2 \pm 14.1$ | $12.4 \pm 12.0$ | $5.6 \pm 15.0$ | $-5.3 \pm 10.6$ | $59.3 \pm 81.4$ | NaN $\pm$ 0.0 | NaN $\pm$ 0.0 |
| sel302 | Se (%) | 96.67 | 96.67 | 96.67 | 100.00 | 100.00 | 6.67 | 6.67 |
| SNR | $P^{+}_{min}$ (%) | 50.88 | 50.88 | 50.88 | 41.67 | 41.10 | 100.00 | 100.00 |
| 6dB | $m \pm s$ (ms) | $4.0 \pm 54.5$ | $20.0 \pm 55.2$ | $11.2 \pm 52.7$ | $-10.8 \pm 24.9$ | $56.7 \pm 67.6$ | $-14.0 \pm 38.0$ | $-30.0 \pm 42.0$ |
| sel302 | Se (%) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 6.67 | 6.67 |
| SNR | $P^{+}_{min}$ (%) | 56.60 | 56.60 | 56.60 | 40.00 | 39.47 | 100.00 | 100.00 |
| 0dB | $m \pm s$ (ms) | $-1.3 \pm 75.5$ | $2.9 \pm 76.1$ | $-4.9 \pm 74.2$ | $-5.3 \pm 42.9$ | $38.1 \pm 65.4$ | $-50.0 \pm 2.0$ | $-60.0 \pm 8.0$ |
| sel302 | Se (%) | 96.67 | 96.67 | 96.67 | 96.67 | 100.00 | 6.67 | 6.67 |
| SNR | $P^{+}_{min}$ (%) | 43.94 | 43.94 | 43.94 | 29.90 | 30.30 | 100.00 | 100.00 |
| -6dB | $m \pm s$ (ms) | $-15.9 \pm 94.6$ | $0.0 \pm 97.8$ | $9.2 \pm 99.4$ | $-15.6 \pm 47.5$ | $29.1 \pm 70.8$ | $-18.0 \pm 42.0$ | $-34.0 \pm 30.0$ |
| Tolerances for $s$ | | 10.2 | - | 12.7 | 6.5 | 11.6 | - | 30.6 |

Table 4.4: Validation of the delineation of the signal `sel302` with different muscle artifact SNR using **morphological filtering**.

To test the filters in the Shimmer platform, we have used the open-source GCC toolchain for MSP430 [20], based on GCC 3.2.3, which is the family processor of the MSP430F1611 in the Shimmer node. The compilation for the node needs several changes in the source code. At first, all the input and output to and from the node have to be performed through the serial USB port. In the PC in which the node is attached, a program to send and receive data through the serial port is needed. This program will open the input file, read its contents and then send the input samples to the node. For delineation purposes, only those samples in the input signal which belong to an annotated interval are sent as input. The communication process between this program and the node is based on an acknowledgement process in which special characters are used to flag different periods of the transmission. In the node, the program in execution has to support the communication by using the same protocol.

Since the Shimmer node only has 10kB of RAM and 48kB of Flash memory, it is not enough to host all the input data. Therefore, the source code of the implementations of the filters had to be adapted to work with a small set of input samples. To support this, it was needed to use an input and an output buffer. The first step is to read through the UART the first samples to fill the input buffer. This buffer is 512 elements long so it is possible with the first load to fill the first circular buffers for the morphological filtering. In the cubic spline technique, the first input dataset also provides enough samples to initialize properly the knot locator.

Then, once all the samples in the input buffer have been processed, it is needed to ask the program in the PC for more input samples through the serial communication. When the received input dataset is not enough to full the input buffer the implementation ends its execution because of reaching the end of the input data. Our implementation uses the clock of the node to get the elapsed execution time without the transmission time, since it is possible to stop and resume timing.

Since there is no floating point hardware in the MSP430, all arithmetic operations performed in the node have to use integers only. The mspgcc toolchain translates floating point operations in slow integer-emulation compatible operations. The morphological filtering implementation only uses comparisons and addition operations to calculate except for the last average operation. Since it is a division by two, the compiler translates it as a right shift operation.

The operations needed in the cubic spline implementation are more complex. The mspgcc toolchain uses the hardware multiplier to execute multiplications by factors other than power of 2 -which can be replaced by shifting operations- and, since there is no hardware divisor, the compiler provides translation for division operations by software emulation. The cubic spline implementation uses exponentiation and division operations, which will result in slow compatible operations. Furthermore, exponentiation operations have to be performed using 32-bits instead of the 16-bits native operations to avoid overflow. Taking these facts into account, the cubic spline implementation is supposed to be slowest in execution because of the use of operations which are not directly supported by the MSP430 and because of the use of dynamic memory and the allocation cost associated.

In the following table, execution times in the Shimmer platform are shown for different input signals. For each signal, only those samples in the annotated interval are sent to the node. For the cubic spline technique, two different implementations are tested. To avoid the extra memory cost associated to the

use of a process queue to try to emulate realtime processing, the implementation without that queue provides output once a beat is ready for processing. This makes unnecessary to use a second queue and it is good to test whether there is an extra memory cost associated.

| Input signal | Samples<br>Real time | Morph.filt.<br>bw&ns | Morph.filt.<br>bw only | Cubic spline<br>w/o queue | Cubic spline |
|---|---|---|---|---|---|
| `sel100` | 8424<br>33,7sec | 1302ms<br>3,86% | 813ms<br>2,41% | 2576ms<br>7,64% | 2574ms<br>7,64% |
| `sel302` | 7467<br>29,8sec | 1215ms<br>4,07% | 760ms<br>2,54% | 2323ms<br>7,78% | 2317ms<br>7,76% |
| `sel123` | 11676<br>46,7sec | 1854ms<br>3,97% | 1137ms<br>2,43% | 3675ms<br>7,87% | 3656ms<br>7,83% |
| `sel16272` | 10372<br>41,49sec | 1730ms<br>4,17% | 1112ms<br>2,68% | 3263ms<br>7,86% | 3268ms<br>7,88% |
| `sel103` | 8911<br>35,64sec | 1430ms<br>4,01% | 892ms<br>2,5% | 2768ms<br>7,77% | 2764ms<br>7,75% |
| Average | | 1506,2ms<br>4,02% | 942,8ms<br>2,51% | 2921ms<br>7,78% | 2915,8ms<br>7,77% |

Table 4.5: Execution times for each input signal and implementation in the Shimmer platform.

These results show that the morphological filtering implementation is twice faster than the cubic spline technique for baseline wander removal and noise suppression. For baseline wander removal only, the morphological filtering implementation here presented is three times faster than the cubic spline technique. Percentages show the relationship between real time represented by the input samples and the execution time obtained. The worst relative time, 7,88%, is admissible for realtime operation and the morphological filtering results are good enough to support several posterior operations in the node with the filtered ECG signal.

The difference between both implementations of the morphological filtering is remarkable. Without noise suppression, the implementation is truly fast and, according to the results provided by the delineation phase, it offers a great performance. Since the result of the noise suppression part is not as good as expected, it would be better to use only the baseline wander removal part and then another filter which may offer better performance for noise suppression.

In a PC platform, the cubic spline technique is almost a 30% faster than the morphological filtering but in the Shimmer platform it is actually slower.

This could be explained because of the use of 32-bit operations and some multiplications and divisions needed, which lead to a poorer performance in an embedded platform with limited hardware resources.

| Implementation | Executable size | Static RAM usage |
|---|---|---|
| Morph. filt. bw only | 4780 bytes | 3932 bytes |
| Morph. filt. bw&ns | 6840 bytes | 4030 bytes |
| Cubic spline | 5946 bytes | 3205 bytes |
| Cubic spline w/o q. | 5116 bytes | 3199 bytes |

Table 4.6: Comparison of the executable size and static RAM usage.

The executable size results show that there is no great difference between the implementations. The static RAM usage is almost equal but it is necessary to keep in mind that both cubic spline implementations use dynamic memory.

# Chapter 5

# Conclusions and future work

As a conclusion, baseline wander removal is a filtering need which can be implemented efficiently in an embedded platform. In this project, we have developed several implementations for baseline wander removal and the results of posterior delineation show that it helps to process the ECG signal in noisy contexts in an embedded platform.

The morphological filtering technique offers a great performance even when using the noise suppression part. For baseline wander removal only, the execution time is remarkable. The results of the validation of the delineation are acceptable for SNR values greater than 0 dB. The RAM memory usage is fixed and leaves room for extra code to posterior signal processing. However, the noise suppression performance is not so good as expected since electromyographic noise is difficult to filter. For high SNR values, noise suppression offers acceptable results. The optimizations carried out in this development lead to a fast implementation which allows realtime operation while preserving power consumption.

Regarding the cubic spline implementation, its results, although acceptable, are worse than the morphological filtering implementation. The results of the validation of the delineation after filtering using this technique are poorer than those obtained with the other technique, and execution time is greater because of the complexity of the calculations in an embedded platform. Furthermore, the use of dynamic memory is associated to an extra operation cost and it would lead to a memory usage constraint for another posterior processing implementation which may be executed in the embedded platform at the same time.

The execution in the Shimmer platform show that both techniques are acceptable for realtime operation. However, the faster the execution is, the

lower power consumption, so fast execution is desirable. For this purpose, the morphological filtering implementation without noise suppression offers a great result whereas cubic spline implementations are not equal in efficiency.

The results obtained in a PC platform have been equal to those obtained by the execution in the Shimmer platform so there is not any kind of difference in the output regarding the architecture used for execution. The code has been designed to be portable and the mspgcc compiler offers good equivalent translation for those operations that are not directly supported by the MSP430 microcontroller.

For future work, it would be needed to use another source of ECG signal instead of the Physionet QT database. We have validated the implementations by using a program to send the node the input samples whereas this project stands for realtime ECG processing in a wearable platform. It has been proved that implementations here presented are acceptable to run in an embedded platform, so the next step may have been to perform a real ECG ambulatory processing by the use of a node.

Regarding the implementation of a Wireless Body Sensor Network, these implementations would be only the first step in a more complex system but, at the same time, a filtering stage is needed at first for posterior processing. Because of the optimizations made, the morphological filtering implementation offers a good performance to support posterior operations.

According to the results of noise suppression, it is needed a best performance. As exposed in the first chapter, noise suppression is a major challenge in ECG processing and there is not a suitable method yet. The implementation of morphological filtering offers good performance when SNR value is high but it increases the execution time. Another approach to noise suppression may result in a better performance so only the baseline wander removal made by this technique would be enough.

# Chapter 6

# International publications derived from this work

The morphological filtering implementation here presented has been used for multi-lead wavelet delineation on an embedded platform. As done in this report, input signal was filtered before performing a wavelet delineation. Multi-lead wavelet delineation stands for increasing delineation accuracy by the use of two leads at the same time. Baseline wander removal filtering is needed to process input signal before delineation and the morphological filtering has been chosen because of its performance. Results of this development have been published in *Computers in Cardiology*:

*Multi-Lead Wavelet-Based ECG Delineation on a Wearable Embedded Sensor Platform*, Francisco Rincón, Nicolas Boichat, Víctor Barbero, Nadia Khaled, David Atienza, Proc. of Computers in Cardiology (CinC'09), IEEE Press, pp. 1 - 4, Park City, Utah, USA, September 13-16, 2009.

# Chapter 7

# References


1. L. Sörnmo and P. Laguna. *Bioelectrical Signal Processing in Cardiac and Neurological Applications*. Elsevier Academic Press, 2005.

2. P. Laguna, R.G. Mark, A. Goldberg, and G.B. Moody. *A database for evaluation of algorithms for measurement of QT and other waveform intervals in the ECG*. Computers in Cardiology 1997, pages 673-676, Sep 1997. `http://www.physionet.org/physiobank/database/qtdb/doc/`.

3. Y. Sun, K.L. Chan, and S.M. Krishnan. *ECG signal conditioning by morphological filtering*. Computers in biology and medicine, 32(6):465-479, 2002.

4. C. R. Meyer and H. N. Keiser. *Electrocardiogram baseline estimation and removal using cubic splines and space-state computation techniques*. Computers and biological research 10, 1977, pages 459-470.

5. PhysioNet. *The MIT-BIH Noise Stress Test Database*. `http://www.physionet.org/physiobank/database/nstdb/`.

6. N. Nethercote and J. Seward. *Valgrind: A Framework for heavyweight dynamic binary instrumentation*. Proceedings of ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation (PLDI 2007), San Diego, California, USA, June 2007.

7. G. B. Moody and R. G. Mark, *Development and evaluation of a 2-lead ECG analysis program* in Computers in Cardiology. IEEE Computer Society Press, 1982, pp. 39-44.

8. G. B. Moody, W. E. Muldrow, R. G. Mark. *A noise stress test for arrhythmia detectors*. Computers in Cardiology 1984; 11:381-384.

9. PhysioToolkit, open source software for biomedical science and engineering. `http://www.physionet.org/physiotools/`.

10. N. Boichat, N. Khaled, F. Rincon, and D. Atienza. *Wavelet-Based ECG Delineation on a Wearable Embedded Sensor Platform.*

11. C.S.E.W. Party. *Recommendations for measurement standards in quantitative electrocardiography.* Eur. Heart J, 6:815-825, 1985.

12. P. W. Macfarlane, J. Peden, G. Lennox, M. P. Watts and T. D. V. Lawrie, *The Glasgow system* in Trends in Computer-Processed Electrocardiograms (J. H. van Bemmel and J. L. Willems, eds.), pp. 143-150, Amsterdam, North-Holland, 1977.

13. F. Badilini, A. J. Moss and E. L. Titlebaum, *Cubic spline baseline estimation in ambulatory ECG recordings for the measurement of ST segment displacements* in Proc. Conf. IEEE Eng. Med. Biol. Soc. (EMBS), pp.584-585, IEEE, 1991.

14. T. N. E. Greville, *Theory and applications of spline functions.* Academic Press. New York, 1969.

15. R. Robergs and R. Landwehr. *The Surprising History of the HRmax=220-age Equation.* Journal of Exercise Physiology 5 (2): 1-10. ISSN 1097-9751, 2002.

16. O. Inbar, A. Oten, M. Scheinowitz, A. Rotstein, R. Dlin, and R. Casaburi. *Normal cardiopulmonary responses during incremental exercise in 20-70 year old men.* Med Sci Sport Exerc 1994;26(5):538-546.

17. C.-H. Henry Chu, E.J. Delp, *Impulsive noise suppression and background normalization of electromagnetism signals using morphological operators*, IEEE Trans. Biomed. Eng.36 (2) (1989) 262-272.

18. Cuiwei Li, Chongxun Zheng, and Changfeng Tai. *Detection of ECG characteristic points using wavelet transforms.* Biomedical Engineering, IEEE Transactions on, 42(1):21-28, Jan. 1995.

19. J.P. Martinez, R. Almeida, S. Olmos, A.P. Rocha, and P. Laguna. *A wavelet-based ECG delineator: evaluation on standard databases.* Biomedical Engineering, IEEE Transactions on, 51(4):570-581, April 2004.

20. *The GCC toolchain for the Texas Instruments MSP430 MCUs.* `http://mspgcc.sourceforge.net/`.

21. R. O. Morales, M. A. Pérez Sánchez, J. V. L. Ginoria, R. Grau and R. R. Ramírez. *Evaluation of QRS morphological classifiers in the presence of noise*. Computers and Biomedical Research, 30, 200-210, 1997.

22. D. Atienza. *Wireless Sensor Networks*. Universidad Complutense de Madrid, 2009.

23. *Shimmer Sensor Platform*. `http://www.shimmer-research.com/`.

24. J. A. van Alsté, W. van Eck and O. E. Herrman, *ECG baseline wander reduction using linear phase filters*. Comput. Biomed. Res., vol. 19, 417-427, 1986.

25. L. Sörnmo, *Time-variable digital filtering of ECG baseline wander*. Med. Biol. Eng & Comput., vol. 31, 503-508, 1993.

26. R. Jané, P. Laguna, N. V. Thakor and P. Caminal. *Adaptive baseline wander removal in the ECG: comparative analysis with cubic spline technique*, in Proc. Computers in Cardiology, 143-146, IEEE Computer Society Press, 1992.

27. J. L. Talmon, J. A. Kors and J. H. van Bemmel. *Adaptive Gaussian filtering in routing ECG/VCG analysis*. IEEE Trans. Acoust. Speech Sig. Proc., vol. 34, 527-534, 1986.

28. V. de Pinto. *Filters for the reduction of baseline wander and muscle artifact in the ECG*. J. Electrocardiol., vol. 25 (suppl.), 40-48, 1991.

29. *CodeBlue: Wireless sensors for medical care*. Harvard Sensor Networks Lab. University of Hardvard. `http://fiji.eecs.harvard.edu/CodeBlue`.