NTNU

Kunnskap for en bedre verden

Department of ICT and Natural Sciences

AIS2201 - Digital Signal Processing

Portfolio Project Description Part 1: Signal Modeling & System Testing

Kai Erik Hoff

Date

Overview

This document contains 3 separate sections:

1. About the assessment method; Here is a detailed description of how portfolio assessment is utilized in the course, what course works make up the portfolio, and how the portfolio is to be handed in.
2. Description of the portfolio project in it's entirety, describing the problem to be addressed and specifications of the system to be developed.
3. A detailed description of part 1 of the project (there are 3 parts in total). Make sure to read this section thoroughly before you start working with the project.

Table of Contents

1 About portfolio assessment in AIS2201

There will be no final exam (written or oral) in this course. Instead, the basis for the final grade (A-F)[1] will be a **portfolio**[2] which is handed in at the end of the semester. A collection of completed course work will be submitted to **Inspera** as the final portfolio, consisting of both weekly assignments and a project.

- **Weekly assignments:** 20% *of final grade*
  PDF documents for up to 10 weekly assignments with end-of-semester review included. Instructions for exporting the assignment as PDF are found on the front page of each assignment.
- **Project:** 80% *of final grade*
  One project report covering the entirety of the project *and* functioning code solution for each project stage. Weighting of contributions to the final grade:
  - Report: 60% *of final grade*
  - Code solution: 20% *of final grade*

The project is handed out at the beginning of the semester, and will run in parallel with weekly assignments throughout the semester. Divided into 3 distinct parts, each part of the project focuses on a particular stage in the development process of a Digital Signal Processing (DSP) system which aims to tackle a particular problem.

- **Part 1 (begins week 34) - Signal modeling & system evaluation:**
  Focuses on using simplified mathematical and/or statistical signal models to create simulated input data to the DSP system. Pre-existing knowledge of the ideal system output will provide a basis for evaluating the system's performance. Use of Python/MATLAB is recommended at this stage.
- **Part 2 (begins week 40) - Algorithm development:**
  Focuses on adapting signal processing principles learned throughout the course to the particular problem encountered in the project. The simulation developed in part 1 should serve as a tool for testing and evaluating potential solutions. Use of Python/MATLAB is recommended at this stage.
- **Part 3 (begins week 46) - Implementation:**
  Focuses on implementing the DSP system developed during stage 2 on a STM32 microcontroller to run in real-time.

Before parts 2 & 3 can begin, you are required to take part in a supervision meeting with the course lecturer as part of the mandatory activities in this course. The focus of this meeting is to provide constructive feedback on your work so far.

For each part, two levels of complexity will be specified:

- **Level 1:** The minimum complexity required for a viable solution. Everyone is expected to be able to implement this.
- **Level 2:** Expanded functionality. Here will be given some challenges (still within the curriculum) that not everyone will be able to solve.

To achieve the highest grade (**A**), level 2 must be completed in addition to level 1 (when applicable). Completing only level 1 will, as a rule, indicate "average" performance, approximately a **C** on the grading scale. However, remember that implemented functionality is not the only criterion for grading.

[1] Link to NTNU grading scale
[2] Link to portfolio assessment memo

2 Project Description - ECG Heart Rate Monitor

NB! What follows is a description of the particular challenge which the DSP system developed throughout your project should aim to address. Further on in the document is a detailed description of what you are supposed to do for stage 1 of the project. Do not begin programming/developing before reading the description thoroughly!

2.1 Background

Medical measurements such as electrocardiograms (ECG) are well known for producing raw data which is often plagued with noise, as illustrated in figure 1. Due to sensors being placed on the skin and the relative weakness of the signals measured, interference can be picked up at a number of stages before the signal reaches an analog-to-digital converter. As a result, further digital signal processing is commonly applied once the signal has been sampled in order to minimize the impact noise has on the measurement's quality.

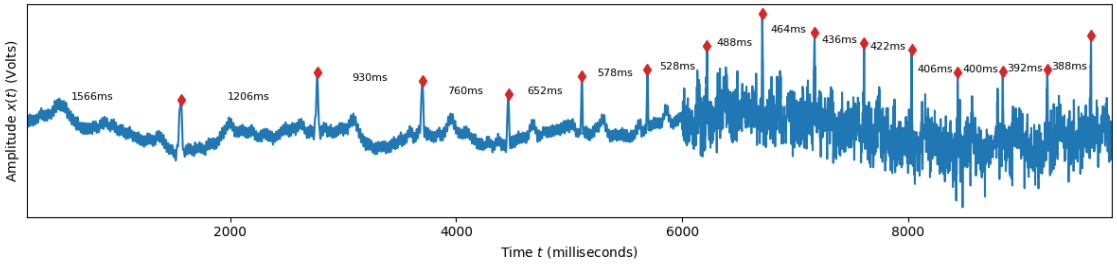


Figure 1: Noisy ECG signal

The challenge to be addressed in this project is to develop a digital signal processing system which aims to accurately estimate the heart rate of a person based on a potentially noisy single-lead ECG signal, such as the one produced by the AD8232[3] sensor. Once the system has been tested using simulated data, it will be implemented on an STM32 microprocessor to process ECG measurements in real-time.

2.2 System specification

Exactly how the system estimates the heart rate is up to each student, but the system must adhere to a specific format for input and output:

- **Input:** Raw ECG data sampled at a uniform sample rate f_s. You are free to modify the sample rate as you please, but a common sample rate used for ECG data is $f_s = 500$ samples per second. During development of the system this will be an array of pre-recorded data, but once implemented, the system will sample directly from an analog ECG sensor.
- **Output:** A series of estimated time durations between each heartbeat, as illustrated in Figure 1. The pulse durations should be given in milliseconds.

To illustrate usage of the specified system input/output, prototype code for a simple heartbeat detection system which registers threshold crossings is shown below as an example of how such a system could be modeled in Python. It is worth noting that the code in question is highly dependent on a "clean" ECG signal, and would not be successful in processing the signal shown in figure 1.

[3]Sparkfun Heart Rate Monitor based on the AD8232 Integrated Circuit

Code example 1: a simple function capable of calculating heartbeat durations

```
import numpy as np
def get_crossing_intervals(x, fs, threshold=0.0):
    """
    Function to detect positive threshold crossings in a discrete-time signal.
    Parameters:
     x - discrete signal samples
     fs - sampling frequency
     threshold - threshold value, defaults to 0.0.
    Returns:
     intervals - a list of time durations between each threshold crossing
                 registered
    """
    N = len(x)
    intervals = []
    above_threshold = True
    last_crossing = 0.0
    for n in range(N):
        if x[n] > threshold and not above_threshold:
            above_threshold = True
            current_crossing = n/fs
            intervals.append(current_crossing - last_crossing)
            last_crossing = current_crossing
        elif x[n] <= threshold and above_threshold:
            above_threshold = False
    return intervals
```

2.3 Expectations

While developing a DSP system that is *capable* of detecting heartbeats can be as simple as detecting threshold crossings, developing a system which can do so *reliably* - despite signal interference - is another matter entirely. Contrary to the weekly assignments, there is no single "correct" solution, and the main evaluation criteria for the project will be the degree to which it demonstrates understanding of signal processing principles.

Students are encouraged to be creative and adapt any principles introduced throughout the course to the problem at hand if they believe it has potential. For those brave souls who wish to venture beyond the course curriculum, there is extensive research to be found on the subject, but make sure to manage your time well.

It is important to note that the final evaluation of the project will not necessarily be dependent on the final system's objective performance and/or complexity. Rather, students will be evaluated by their ability to **apply** knowledge acquired from the course to an unfamiliar situation, **analyze** both simulated and real-world system behavior and **investigate** the cause of any discrepancies between expected and observed behavior.

If, for instance, the system you have developed does not perform as expected during testing, the process of determining the cause is often a more valuable learning experience than discarding your work and attempting something new. Documenting such investigative processes and including them in the final project report will count positively towards your grade.

2.4 Use of Artificial Intelligence (AI)

Use of artificial intelligence tools as an aid in developing your system is encouraged. Tools such as GitHub CoPilot, Jetbrains AI Assistant, ChatGPT etc. can be of great help **when used**

correctly! As a rule, you should NOT use AI-tools to generate code which you do not understand yourself, although expectations vary depending on what aspect of the project the code is related to:

- You **must** be able to account for the purpose and function of **every single** line of code in your project which is used either to generate simulated ECG signals or process/analyze ECG signals for the purpose of calculating heart rate.
- You are not expected to be an expert in use of data visualization tools such as `matplotlib`, `plotly` etc. Requirements for how well code related to data visualization is understood is significantly more lenient. Feel free to use AI to assist in annotating plots, creating fancy 3D visualizations or similar, but make sure the information being displayed in the figures is accurate.

Use of AI tools **must** be documented in the project report, with descriptions of which tools were used, as well as examples of use. If prompts are used to instruct the language model to respond in a particular manner, these prompts should be included.

2.5 The final report

A latex template for the final report is linked below [4]. The main emphasis of your report should be the following:

- A description of theoretical principles chosen for the DSP system and why you chose them (e.g. what do you expect a lowpass filter to accomplish and why).
- A description of the proposed DSP system including a system block diagram. The purpose of each system block should be explained in the text.
- Presentation and discussion of simulated test results, tying the observed behavior of your improvements to DSP principles covered in the course.
- An overview of your final system implementation, explaining the flow of sampled information from ADC to fundamental frequency estimate. System diagrams are encouraged.

Provide Python-code from parts 1 & 2, as well as the code of your "main.c" file as separate attachments when handing in the report.

[4] Overleaf project report template

3 Part 1: Signal modeling & System Testing

Before development of the actual DSP system can begin, it is important to have a detailed understanding of the challenges faced by the DSP system, as well as a method for evaluating how well the DSP system can deal with said challenges. The first part of the project will therefore involve developing a mathematical model for generating sampled noisy ECG signals, and preparing a routine for testing the signal processing algorithms using simulated input signals based on said mathematical model. The aim is to write a program in a high-level scripting language such as Python or MATLAB which does the following:

1. Use mathematical interference- and signal models to create a simulated noisy digital ECG signal with properties determined by simulation parameters.
2. Apply the simulated signal to a prototype DSP system and record the system's output.
3. Compare the DSP system output with the ideal output, quantifying the magnitude of errors, and providing appropriate representation of simulation results.

To ensure a solid foundation for later parts of the project, detailed step-by-step instructions describing the testing routine is given for both levels 1 and 2.

3.1 Level 1: adding noise to dataset

A data file `ECG_data.mat` containing a single-lead ECG signal of length 10s can be downloaded from Blackboard. This data may serve as a basis for generating simulated noisy ECG data, although other datasets are available online for those who wish to experiment.

The additive interference model

Essential to this part of the project is the topic of how signal interference is modeled, as it will form the basis of our simulations. The main type of interference with which we are concerned in this project is additive, meaning the measured ECG signal $x(t)$ is the sum of a "pure" ECG signal $s(t)$ and some interference signal $q(t)$.

$$x(t) = s(t) + q(t)$$

Once sampled, this can be modeled in the discrete domain as

$$x[n] = s[n] + q[n]$$

where $x[n]$, $s[n]$ and $q[n]$ are ideally sampled discrete representations of $x(t)$, $s(t)$ and $q(t)$ using a constant sampling rate f_s. For our heart rate detection system there are three main categories of interference to be concerned with: white noise, baseline drift and narrowband interference.

White noise

White noise will typically be present to some degree. For an ECG signal, muscle activity not related to the heartbeats will manifest as noise in the signal with an approximately white spectrum. White noise is modeled as a random process with zero mean, variance σ^2 equal to the noise power P_{white}, and a gaussian probability distribution.

$$q_{\text{white}}[n] \sim \mathcal{N}(\mu, \sigma^2), \quad \mu = 0, \quad \sigma^2 = P_{\text{white}} \tag{1}$$

Recommended approach:
Write a function which can generate N samples of white gaussian noise with power P_{white}.

Baseline drift

The ECG sensor electronics involves some integrating circuitry. As a consequence, the baseline is sensitive to drift as weak noise components accumulate over time. The resulting interference can be modeled as brownian noise. The simplest approach to modeling this in the discrete domain is:

$$q_{\text{drift}}[n] = \sum_{n=0}^{N-1} \frac{1}{f_s} \cdot r[n] \tag{2}$$

where $r[n]$ is some relatively low power white gaussian noise.

Recommended approach:
Write a function which can generate N samples of baseline drift given a sample rate f_s and the power P_r of the noise $r[n]$ which is accumulated towards drift.

Narrowband interference

The long sensor leads between the electrodes and the ECG sensor is vulnerable to induced voltages from nearby electromagnetic sources. The most notable source of narrowband interference is the the 50 Hz AC mains voltage, and is referred to as power line interference. Power line interference can be modeled as a pure 50 Hz sinusoid.

$$q_{\text{mains}}[n] = A \cdot \cos(2\pi \cdot 50 \cdot t + \phi) \tag{3}$$

Recommended approach:
Write a function which can generate N samples of modeled narrowband interference with power P_{mains}. The phase component ϕ should be randomized.

Normalizing signal and noise power

Signal-to-Noise Ratio (SNR) is a commonly used metric for signal quality. Usually measured in decibels (dB), it is a measure of the ratio between signal power and noise power:

$$\text{SNR}_{\text{dB}} = 10 \cdot \log_{10}\left(\frac{P_{\text{signal}}}{P_{\text{noise}}}\right) \tag{4}$$

In theory, extracting desired information from a noisy signal with a given SNR is equally challenging regardless of whether the absolute noise power is 0.0001W or 1KW, as the signal power would scale with noise power. SNR is therefore a suitable input parameter for the ECG signal simulation and is to be used to calibrate the relative power of simulated noise to the desired signal's power.

The power of a signal $s(t)$ is defined in equation 5, and can be estimated based on a discrete-time signal $s[n] = s(T_s \cdot n)$ using equation 6, where the value $\bar{s}$ is the average value of the signal $s[n]$ over the same N samples as described in equation 7.

$$P_{\text{signal}} = E\left((s(t) - E(s(t)))^2\right) \tag{5}$$

$$\hat{P}_{\text{signal}} = \frac{1}{N}\sum_{n=0}^{N-1}\left(s[n] - \bar{s}\right)^2 \tag{6}$$

$$\bar{s} = \frac{1}{N}\sum_{n=0}^{N-1} s[n] \tag{7}$$

Recommended approach:
Write a function which receives N samples of a source signal $s[n]$ as input, alongside parameters for interference type and target SNR. The output should be N samples of a noisy signal matching input specifications.

P.S.: Calculating noise power for baseline drift is not straightforward, as it's impact increases over time. For an infinitely long signal it's power approaches infinity, but for a discrete-time signal of length N generated using equation 2, the expected average power P_{drift} becomes:

$$P_{\text{drift}} = \frac{N \cdot P_r}{2 \cdot f_s^2}$$

While it is debatable whether the term SNR can be applied in this case, it may still serve as a useful scaling parameter, as the resulting interference scales proportionally to the ECG signal.

3.2 Level 2: generate ECG data from signal model

The data file `ecg_data_single.mat` contains one sample period from an ECG measurement, spanning from one pulse peak to the next as shown in figure 2. This waveform will serve as a basis for generating further simulated ECG data which will follow this specific shape. As a simplifying measure, we will be making one significant (and likely naive) assumption regarding the nature of ECG signals.

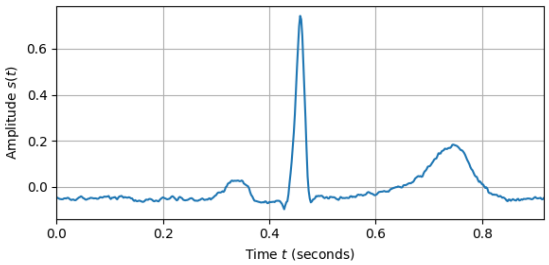


Figure 2: Single ECG Pulse

Assumption:

Adjusting the heart rate for an ECG signal equates to stretching or compressing the pulse shape along the time axis.

Stretching and compressing a waveform can be done quite easily using Fourier analysis, the procedure for which is described throughout this section.

Describing the ECG pulse as a function

The first step towards synthesizing ECG data is to identify a function $g(t)$ which describes the waveform in the data file `ecg_data_single.mat`. This can be done by extracting the Fourier Series

Coefficients of the signal's single period as a set of complex values $\boldsymbol{a}_k$. Once this is done, the function $g(t)$ can be described by equation 8 where T is the pulse duration in seconds (equal to the length fo the data file).

$$g(t) = \boldsymbol{a}_0 + 2 \cdot \sum_{k=1}^{N-1} |\boldsymbol{a}_k| \cdot \cos\left(2\pi \cdot \frac{k}{T} \cdot t + \angle(\boldsymbol{a}_k)\right) \tag{8}$$

Recommended approach:
Write a program / function which can find the function $g(t)$ based on the data file, and generate a highly similar pulse shape which extends to an arbitrary time interval $t \in [t_1, t_2)$. The total number of fourier series components to use is up to you. If you have not already completed assignment 2, it is recommended you do this first.

Adjusting the frequency

Once the function $g(t)$ is found, the heart rate may be adjusted through *time scaling* of the function $g(t)$. For a constant target heart rate R in beats per minute (bpm), this results in a scaled function $g_{\text{scaled}}(t)$ as given by equation 9.

$$g_{\text{scaled}}(t) = g\left(\frac{R \cdot T}{60} \cdot t\right) \tag{9}$$

Recommended approach:
Expand your program to generate a ECG signal of adjustable length with constant heart rate based on the function $g(t)$, where the heart rate is set by a parameter R.

Time-varying heart rate

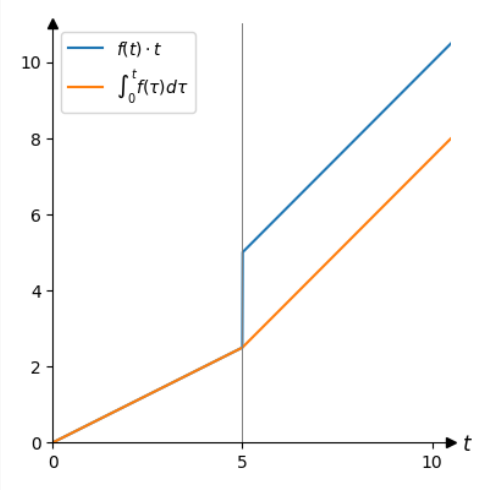
In real life the heart rate of a person will not remain constant over long periods of time. Allowing the heart rate to fluctuate over the duration of the simulated signal is desirable, but equation 9 only works if the heart rate R is a constant. If the heart rate is also a function of time $R(t)$, equation 9 will at worst lead to discontinuities in the generated signal, and is guaranteed to generate unwanted frequency components. The reasoning for this is presented with a simplified exampel below:

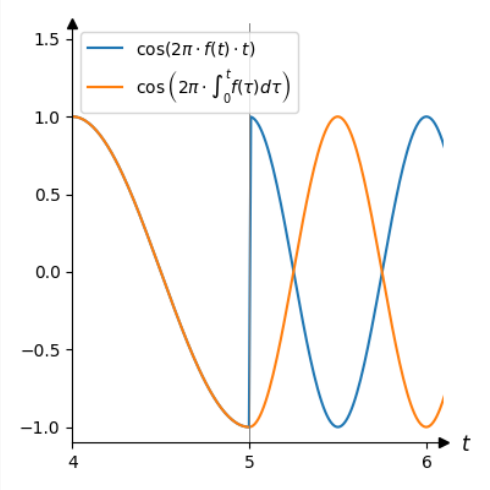
Example 3.2.1: Time-varying frequency sinusoid generation

Consider the sinusoidal signal $x(t) = \cos(2\pi \cdot f \cdot t)$. We wish to generate a signal where the frequency f is not constant, but is dependent on the time t as described below.

$$f(t) = \begin{cases} 0.5 \text{ Hz}, & t < 5 \text{ s} \\ 1 \text{ Hz}, & t \geq 5 \text{ s} \end{cases}$$

When the time t reaches 5 s, the argument to the $\cos(\cdot)$ function (i.e. the product $f(t) \cdot t$) undergoes a discontinuity, leading to a discontinuous waveform. Instead of multiplying the instantaneous frequency $f(t)$ by the time t, we now need to calculate the bounded integral of $f(\tau)$ for $\tau \in [0, t]$ to ensure continuous function curves as shown below:





As a result, the updated expression for the function $x(t)$ which allows for a time-varying sinusoid frequency $f(t)$ is as follows:

$$x(t) = \cos\left(2\pi \cdot \int_0^t f(\tau)d\tau\right)$$

Recommended approach:
Expand your program program to generate a simulated ECG signal when the heart rate is a function $R(t)$ rather than a constant value. Use graphical analysis to verify that the generated signal follows the expected progression over time. You are free to choose any mathematical function to use as $R(t)$, but keep in mind that aliasing will be an issue if you let it go excessively high.

3.3 Evaluating system output

Once a method for generating simulated input data has been established, the DSP system can be tested and it's output can be interpreted. Since the output is specified to be a sequence of heartbeat durations, there are two interesting metrics we can use:

- **True detection ratio:**
  One type of error the system can make is to fail to report a heartbeat where one is present. A measure of how prevalent such errors are would be to calculate how many of the heartbeats present in the source ECG signal are accounted for in the system's output. For instance, if the input signal has does contain 20 heart beats and the output correctly identifies 16 of them, the ratio will be 80%.
- **False detection ratio:**
  Another type of error the system can make is to report the presence of a heartbeat where

there is none. A measure for this type of error could be how many of the heartbeats reported in the system's output correspond to actual heartbeats (e.g. if the output indicates a total of 20 detected pulses, 4 of which occur at different times than those actually present, the false detection ratio will be 20%).

In both cases, we can allow an error margin of ±10ms when determining if a registered heartbeat is valid. Assuming a DSP system is available which can correctly identify heartbeat pulses in an uncorrupted signal, the process of conducting the system evaluation can be automated and boils down to five steps:

1. Load an ECG dataset or generate a customized ECG signal (level 1 or 2).
2. Process ECG signal with prototype DSP system, and record output.
3. Add desired noise to ECG signal.
4. Process noisy ECG signal again with prototype DSP system, and record output.
5. Compare the timestamps for the two outputs and calculate detection ratios.

Visualization

One simulation with a specific set of simulation parameters as described above will produce a result indicating DSP system performance under those specific conditions. And while such a result is useful, it may not convey the entire picture. A perhaps more useful basis for evaluating the DSP system is to conduct successive simulations with progressive levels of noise, and generate plots showing e.g. detection ratio vs. NSR (noise-to-signal ratio).

For those who have completed level 2 in the simulation setup, it may also be informative to include heart rate as an added dimension in the evaluation. Is it safe to assume that the DSP system's robustness is independent of heart rate?

Appendix A contains a number of figures showing different ways the simulation results could be visualized. Some are fairly simple, and some fairly advanced. It is left to the student to decide freely how advanced this part of the project should be.

3.4 Documenting your work

It is recommended that you begin working on the project report at once, rather than leaving all the work for the end of semester. Familiarize yourself with the project report template, and start writing as you work. Throughout part 1 of the project, the general layout of chapters 2 & 4 should begin to take shape.

- **Chapter 2: Background**

  Provide a brief explanation of the ECG signal properties and principles relevant for your project. For example, if a digital filter is introduced later in the project to suppress white noise, the reader needs to be familiar with the frequency-domain properties *of* white noise.

- **Chapter 4: Simulation and Results**

  Explain your choice of simulated test signals, and how the DSP system performance is to be quantified. Where system evaluation is dependent on analyzing plots, provide an explanation of how the plots are to be interpreted.

Appendix

A Visualization Examples

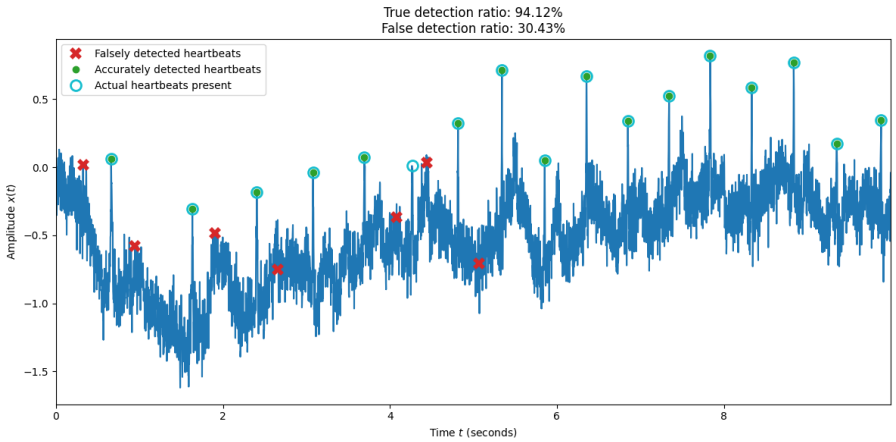


Figure 3: Time series plot of noisy ECG signal comparing detected heartbeats to actual heartbeats.

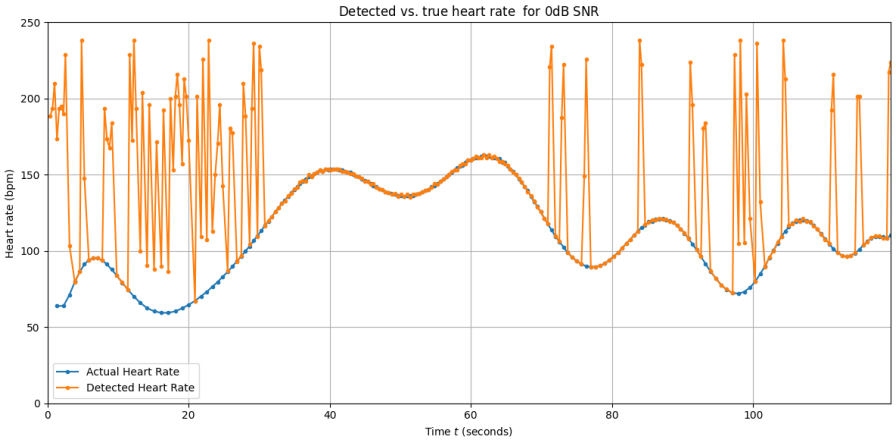


Figure 4: Time series plot of calculated heart rate based on both detected and actual heartbeats.

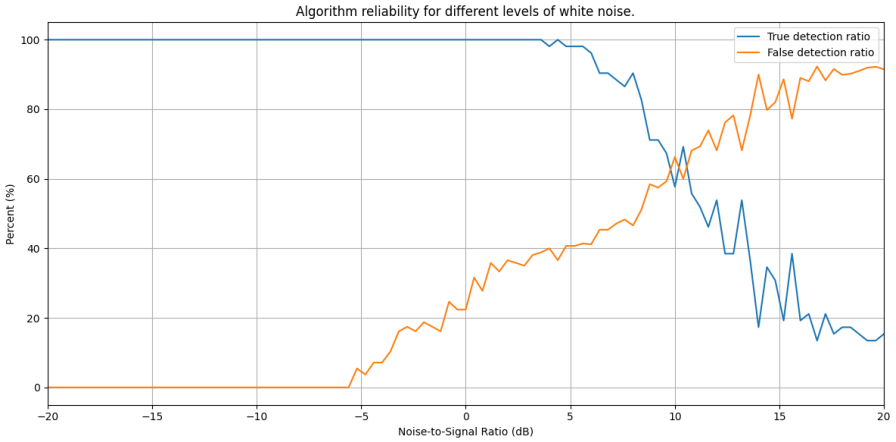


Figure 5: Visualization of results from multiple simulations showing how algorithm reliability degrades with increasing noise levels.

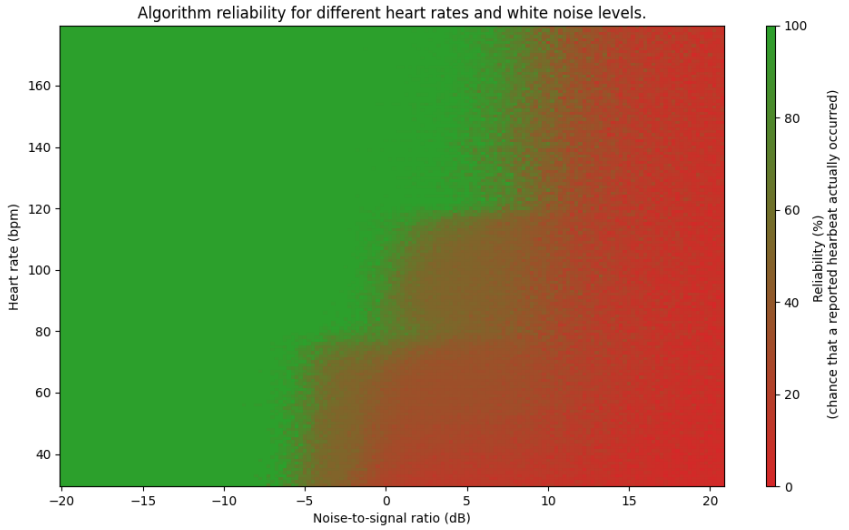


Figure 6: Visulaization of results from a large number of simulations showing how algorithm reliability is impacted both by noise levels and source signal heart rate.