COMP4953: Thesis C

A Computer Model of Electrocardiogram Signals

By Andrew Au (z5020593)

Supervisor: Socrates Dokos

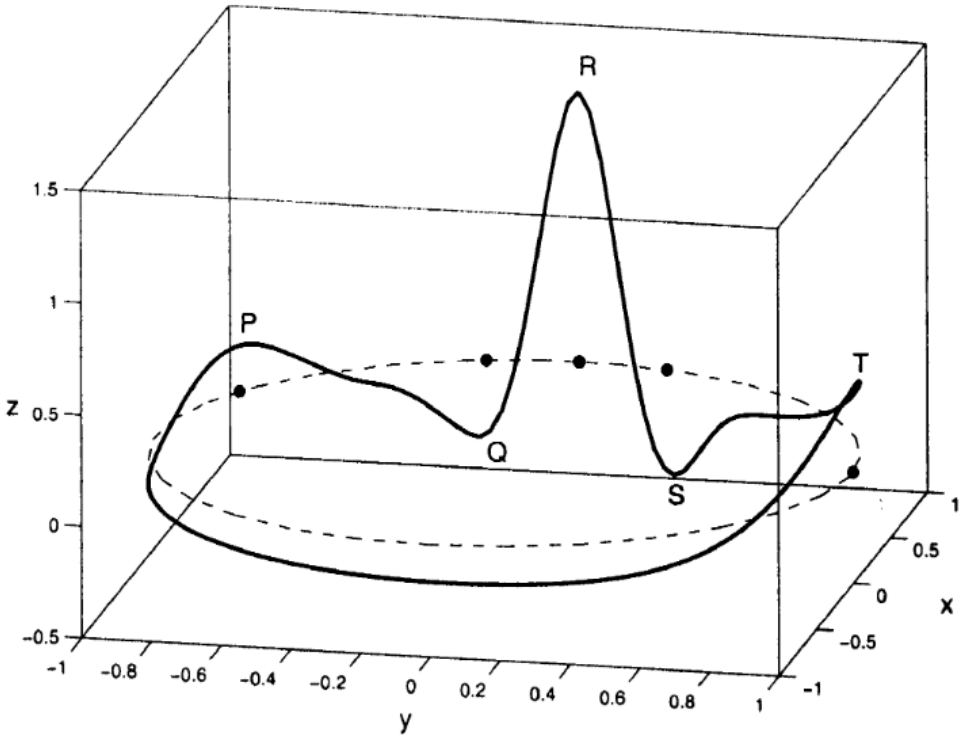
Assessor: Bruno Gaeta

Project Description

Electrocardiogram signals **(ECGs)** are generated by electric activity in the heart. These signals are captured by ECG machines and used to diagnose heart conditions such as arrhythmia.

ECGs can be described with a mathematical model and generated with a computer model.

One method to dynamically generate ECGs was introduced by McSharry et.al which utilizes a system of differential equations in a 3D space.



Trajectory generated by McSharry model in 3-D space (*x, y, z*).

Project Description

Computer models are useful for testing signal processing algorithms against generated regular and irregular ECG signals without having to record data from patients.

Other uses of computer models include noise filtering and parameter estimation of ECGs. Implementations of these features exist with Extended Kalman Filters, which can be used to filter noisy signals and estimate the state of the computer model.

An open source implementation exists for generating ECGs, but extensions with noise filtering and parameter estimation are mostly only available in proprietary software such as MATLAB.

Project Description

The aim of this project is to create an open-source implementation of a computer model of ECG signals that can generate regular and irregular heart signals as well as perform parameter estimation on imported ECG samples.

Implementation: Software

- Python3
- PyQt5
- SciPy (NumPy, matplotlib, SymPy)
- Numexpr (evaluate strings to numbers)

Implementation: ECG Model

The method to generate ECG signals follows the McSharry model which utilizes three differential equations with 16 parameters
a1, .., b1, .., Θ1, .., Θ5, ω.

These equations were written with NumPy and solved with the ODE solver scipy.solve_ivp().

$$\dot{x} = \alpha x - \omega y$$

$$\dot{y} = \alpha y + \omega x$$

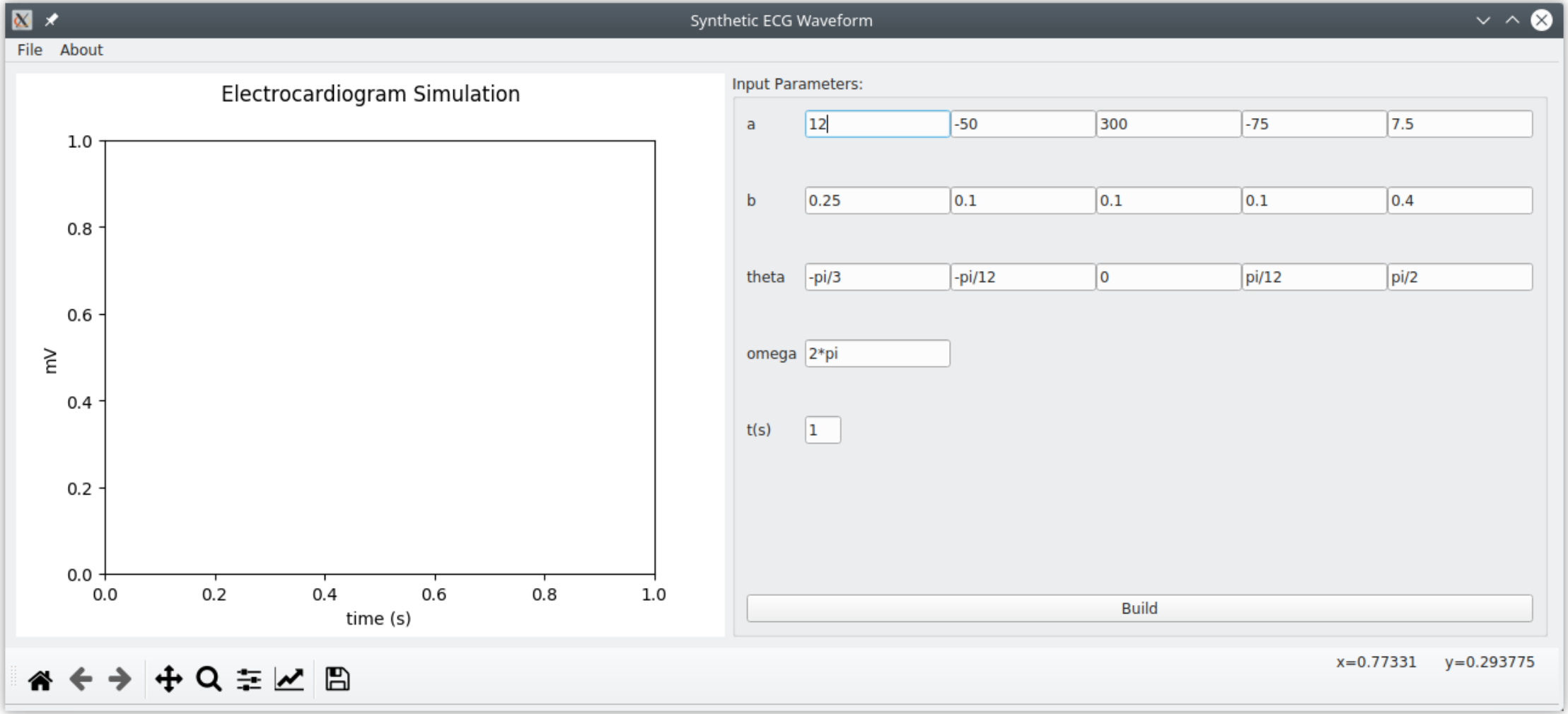
$$\dot{z} = -\sum_{i \in \{P,Q,R,S,T\}} a_i \Delta\theta_i \exp\left(-\frac{\Delta\theta_i^2}{2b_i^2}\right) - (z - z_0)$$

Implementation: Graphical User Interface

PyQt5 was used to implement a graphical user interface (GUI) for user interaction. A number of features were implemented with PyQt:

- Import/export parameters used to generate ECG
- Import/export csv samples
- Plotting/Removing graphs
- Forms
  - Form for inputting 16 variables to generate ECG
  - Form for importing timeframe of sample
  - Form for specifying initial covariance in Extended Kalman Filter

Implementation: Graphical User Interface



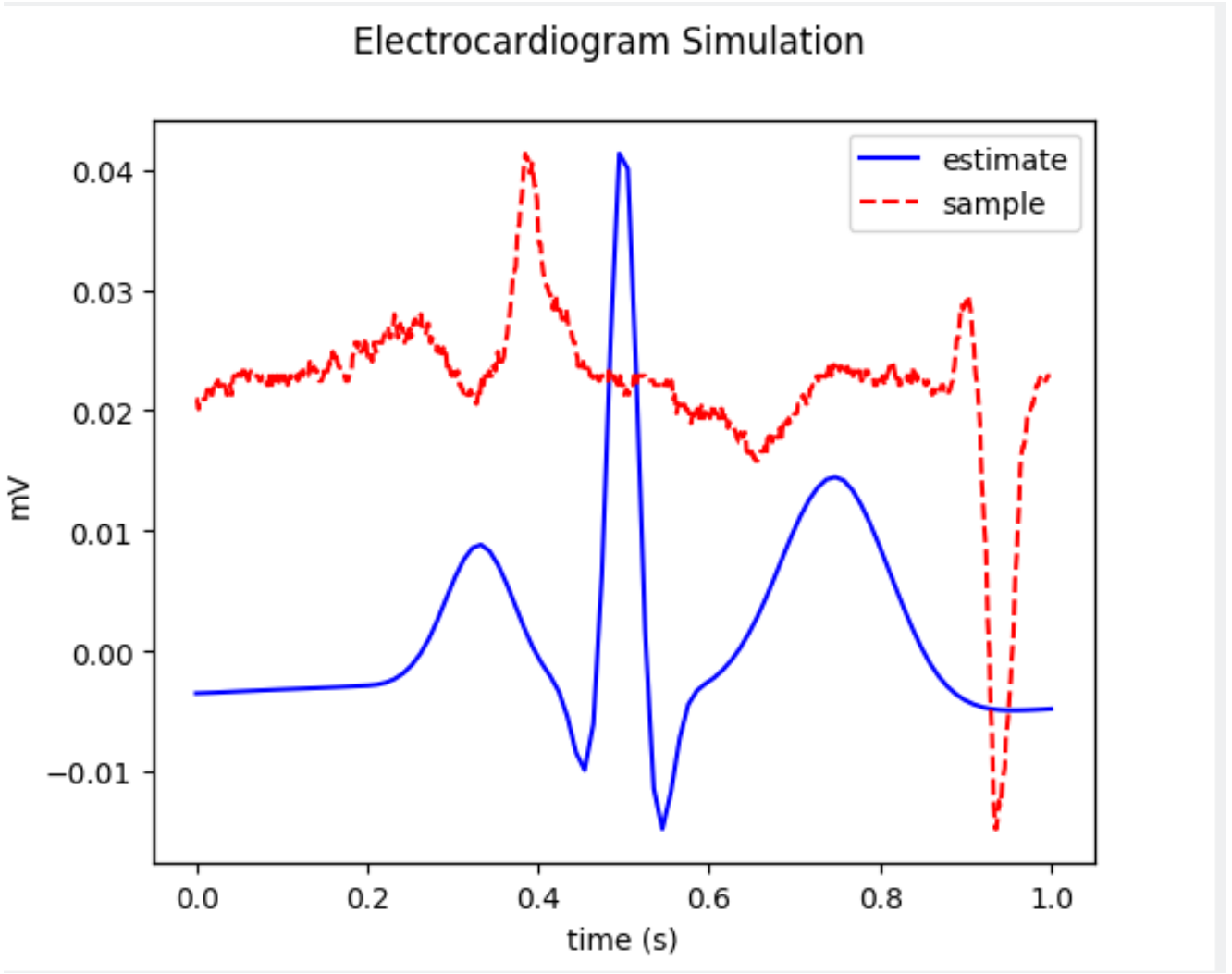
Implementation: Graphing

Matplotlib is a plotting library from SciPy that is used to plot ECGs. It has bindings for many GUI toolkits including PyQt5.

Using matplotlib, it was possible to plot and attach legends to multiple plots in a single canvas.

The previous implementation of plotting used duplicated axes objects on a single canvas. This was over complicated and was rewritten so that interactions with plots were handled by their labels.

Implementation: Graphing



Implementation: Parameter Estimation

Parameter estimation can be performed with a Kalman Filter, which performs a prediction with a model and then updates it with a series of measurements.

For non-linear equations such as the McSharry model, an Extended Kalman Filter (EKF) must be implemented.

The prediction step of the EKF can be implement with scipy.solve_ivp(), but it is rather difficult to modify the base code to perform Kalman Filtering.

Implementation: Parameter Estimation

The predict step of EKF was instead implemented with a discrete version of the McSharry model where *h* is the change in time (effectively Euler's method).

$$(1+\alpha h)x(k)-\omega h y(k)+w_1(k)$$

$$(1+\alpha h)y(k)+\omega h x(k)+w_2(k)$$

$$-\sum_{i\in\{P,Q,R,S,T\}}\frac{A_i\omega}{b_i^2}h\Delta\theta_i\exp\left(-\frac{\Delta\theta_i^2}{2b_i^2}\right)-((h-1)z(k)-hz_0)+w_3(k)$$

Implementation: Parameter Estimation

The predict step of the EKF also requires the Jacobian matrix of the discrete model. Rather than computing the Jacobian manually, the Jacobian matrix was computed with SymPy by symbolically defining the discrete model and returning a function which returns the Jacobian matrix.

The EKF uses the parameters in the form and fixed values x=-1, y=0, z=0 as the initial state. The initial uncertainty of the initial state can be customized when running parameter estimation via a short form.

After the EKF finishes the main application form is updated with the result.

Results and Discussion

Parameter estimation was tested against generated ECG samples.

The Kalman Filter is able to produce an estimate of the parameters of the ECGs, but is highly reliant on the provided initial state. Longer samples would provide a better parameter estimate, but would not be as effective as providing a proper initial estimate.

Another point to note is that a certain ECG may have more than one solution for the dynamic parameters.

It should also be noted that Euler's method is reliant on sampling rate to predict the next step of the filter. Small sampling rates would produce a smooth plot, whereas a large sampling time will produce a sharper plot.

Results and Discussion

Future work for parameter estimation with EKF could be improved by estimating the initial state of the imported sample before running the Kalman Filter.

This would lead to the Kalman Filter converging quickly and producing a better parameter estimate.

End of Presentation