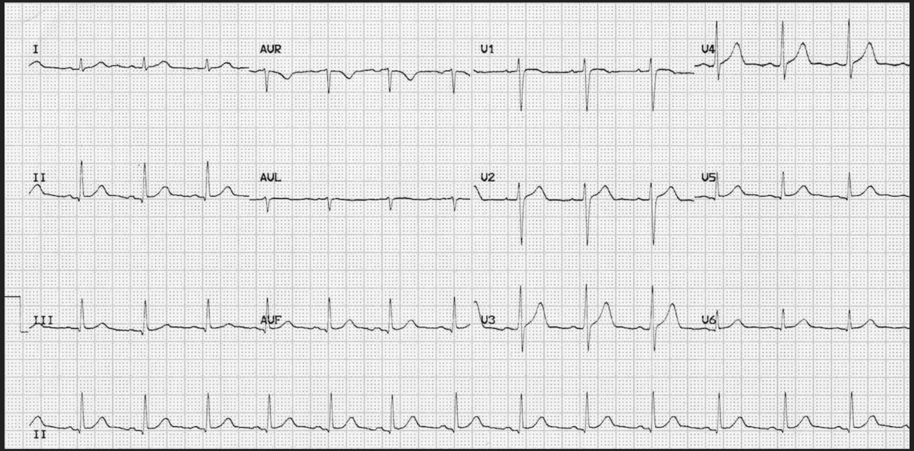
ECG Simulator Based on a Neural Network Trained With Real Patient Data

By
Raul Verduzco | Salvatore Gutierrez | Abisai Diaz

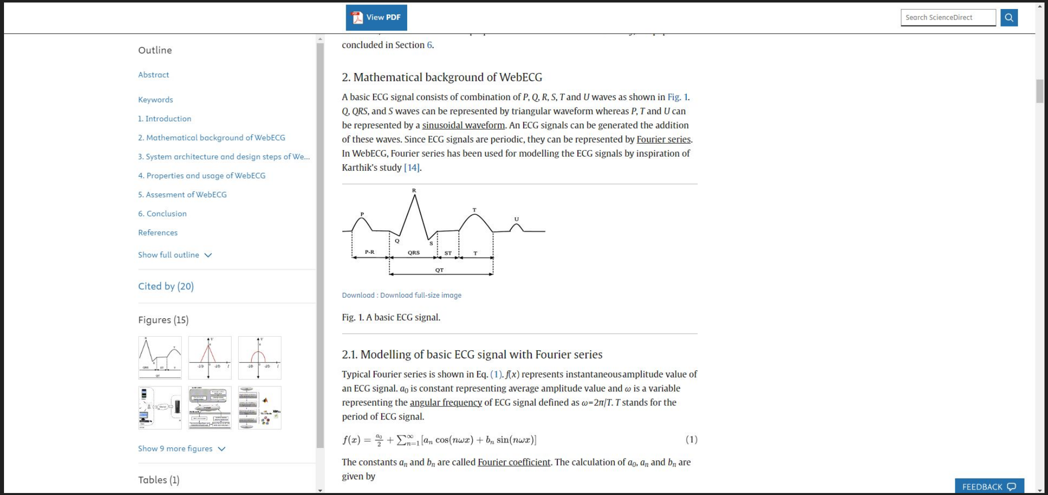
Data

- Created Dataset
  - Ex: normal sinus rhythm, bradycardia, tachycardia, Vfib …
  - Images with all six leads
  - P waves, QRS wave, and T waves



Data

- Ecg data from anonymous patients found online
- Google Scholar
  - Using google scholar for research and find ideas on how to solve our problem



concluded in Section 6.

2. Mathematical background of WebECG

A basic ECG signal consists of combination of *P*, *Q*, *R*, *S*, *T* and *U* waves as shown in Fig. 1. *Q*, *QRS*, and *S* waves can be represented by triangular waveform whereas *P*, *T* and *U* can be represented by a sinusoidal waveform. An ECG signals can be generated the addition of these waves. Since ECG signals are periodic, they can be represented by Fourier series. In WebECG, Fourier series has been used for modelling the ECG signals by inspiration of Karthik's study [14].

Download : Download full-size image

Fig. 1. A basic ECG signal.

2.1. Modelling of basic ECG signal with Fourier series

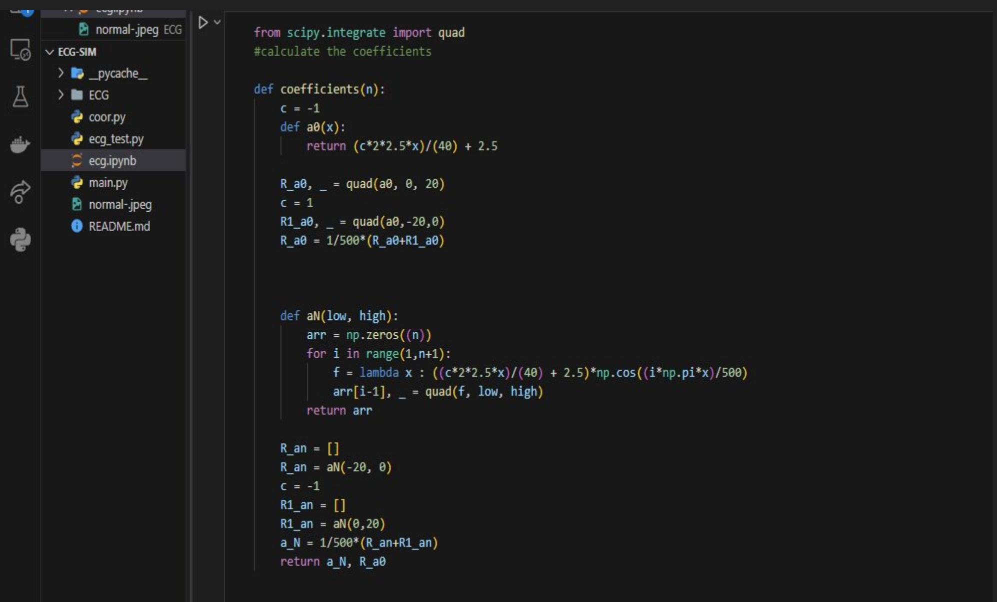
Typical Fourier series is shown in Eq. (1). $f(x)$ represents instantaneousamplitude value of an ECG signal. a_0 is constant representing average amplitude value and ω is a variable representing the angular frequency of ECG signal defined as $\omega=2\pi/T$. T stands for the period of ECG signal.

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}[a_n \cos(n\omega x) + b_n \sin(n\omega x)] \tag{1}$$

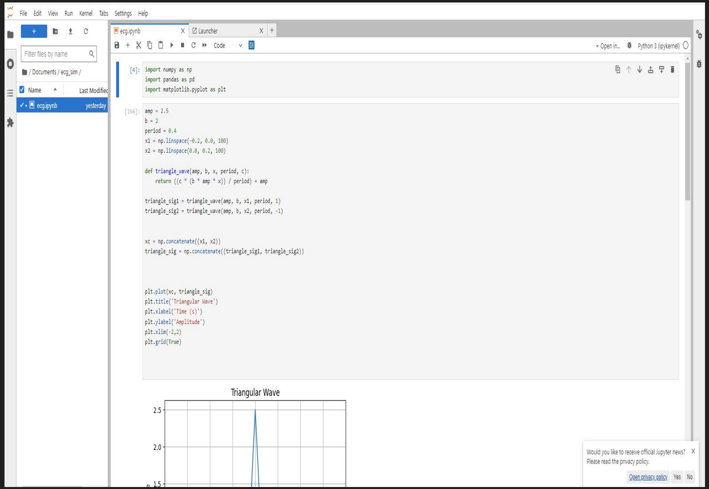
The constants a_n and b_n are called Fourier coefficient. The calculation of a_0, a_n and b_n are given by

Tools

- Python with Jupyter Notebook or Visual Studio Code
- Github
- Numpy library for math calculations
    - np.cos (cosine wave), np.quad (integrate), etc.



```
from scipy.integrate import quad
#calculate the coefficients

def coefficients(n):
    c = -1
    def a0(x):
        return (c*2*2.5*x)/(40) + 2.5

    R_a0, _ = quad(a0, 0, 20)
    c = 1
    R1_a0, _ = quad(a0,-20,0)
    R_a0 = 1/500*(R_a0+R1_a0)



    def aN(low, high):
        arr = np.zeros((n))
        for i in range(1,n+1):
            f = lambda x : ((c*2*2.5*x)/(40) + 2.5)*np.cos((i*np.pi*x)/500)
            arr[i-1], _ = quad(f, low, high)
        return arr

    R_an = []
    R_an = aN(-20, 0)
    c = -1
    R1_an = []
    R1_an = aN(0,20)
    a_N = 1/500*(R_an+R1_an)
    return a_N, R_a0
```



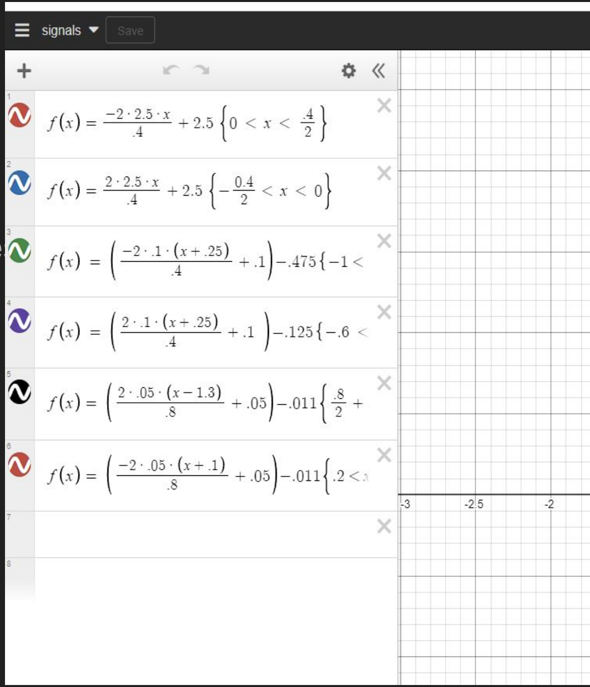
```
[4]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt

[164]: amp = 2.5
       b = 2
       period = 0.4
       x1 = np.linspace(-0.2, 0.0, 100)
       x2 = np.linspace(0.0, 0.2, 100)

       def triangle_wave(amp, b, x, period, c):
           return ((c * (b * amp * x)) / period) + amp

       triangle_sig1 = triangle_wave(amp, b, x1, period, 1)
       triangle_sig2 = triangle_wave(amp, b, x2, period, -1)



       xc = np.concatenate((x1, x2))
       triangle_sig = np.concatenate((triangle_sig1, triangle_sig2))



       plt.plot(xc, triangle_sig)
       plt.title('Triangular Wave')
       plt.xlabel('Time (s)')
       plt.ylabel('Amplitude')
       plt.xlim(-2,2)
       plt.grid(True)
```

Would you like to receive official Jupyter news? Please read the privacy policy.

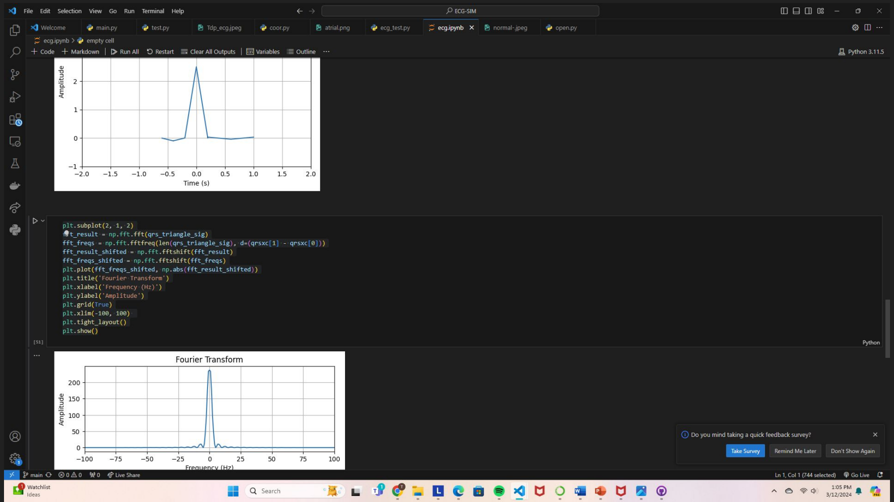
Open privacy policy Yes No

Tools

- Desmos graph/Matplotlib
  - Using desmos graph (or matplot) to test our functions that repre



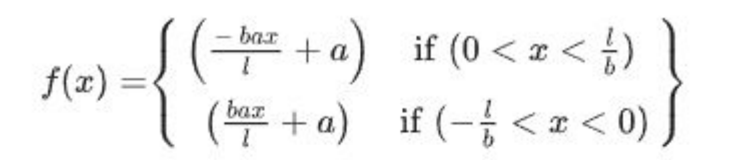
1. $f(x)=\frac{-2\cdot 2.5\cdot x}{.4}+2.5\left\{0<x<\frac{.4}{2}\right\}$
2. $f(x)=\frac{2\cdot 2.5\cdot x}{.4}+2.5\left\{-\frac{0.4}{2}<x<0\right\}$
3. $f(x)=\left(\frac{-2\cdot .1\cdot (x+.25)}{.4}+.1\right)-.475\{-1<$
4. $f(x)=\left(\frac{2\cdot .1\cdot (x+.25)}{.4}+.1\right)-.125\{-.6<$
5. $f(x)=\left(\frac{2\cdot .05\cdot (x-1.3)}{.8}+.05\right)-.011\{\frac{.8}{2}+$
6. $f(x)=\left(\frac{-2\cdot .05\cdot (x+.1)}{.8}+.05\right)-.011\{.2<$



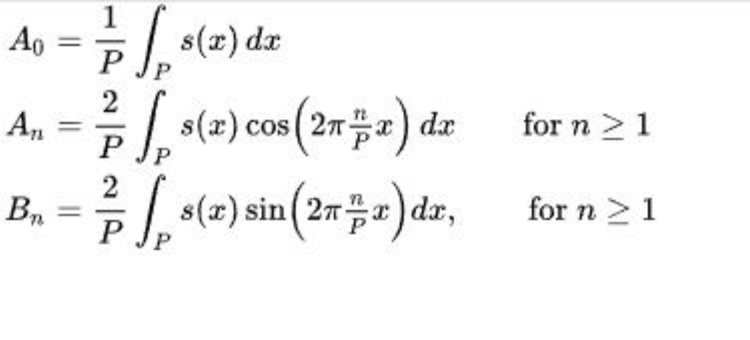
```python
plt.subplot(2, 1, 2)
fft_result = np.fft.fft(qrs_triangle_sig)
fft_freqs = np.fft.fftfreq(len(qrs_triangle_sig), d=(qrsxc[1] - qrsxc[0]))
fft_result_shifted = np.fft.fftshift(fft_result)
fft_freqs_shifted = np.fft.fftshift(fft_freqs)
plt.plot(fft_freqs_shifted, np.abs(fft_result_shifted))
plt.title('Fourier Transform')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude')
plt.grid(True)
plt.xlim(-100, 100)
plt.tight_layout()
plt.show()
```

Other Tools

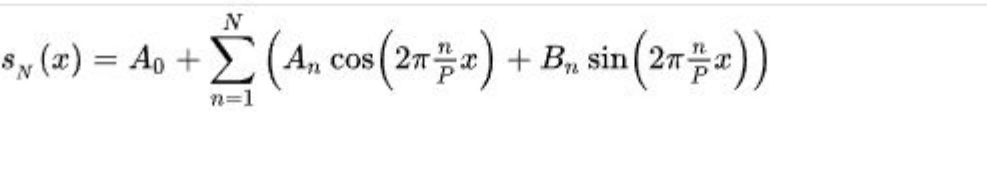
- Math
  - Basic Understanding of calculus concepts like integrating
  - Other



$$f(x) = \left\{ \begin{array}{ll} \left(\frac{-bax}{l} + a \right) & \text{if } (0 < x < \frac{l}{b}) \\ \left(\frac{bax}{l} + a \right) & \text{if } (-\frac{l}{b} < x < 0) \end{array} \right\}$$



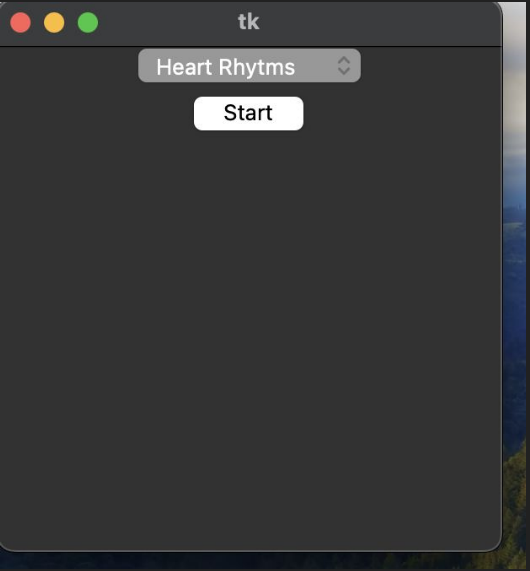
$$\begin{aligned} A_0 &= \frac{1}{P} \int_P s(x)\, dx \\ A_n &= \frac{2}{P} \int_P s(x) \cos\left(2\pi \tfrac{n}{P} x\right) dx \qquad \text{for } n \geq 1 \\ B_n &= \frac{2}{P} \int_P s(x) \sin\left(2\pi \tfrac{n}{P} x\right) dx, \qquad \text{for } n \geq 1 \end{aligned}$$

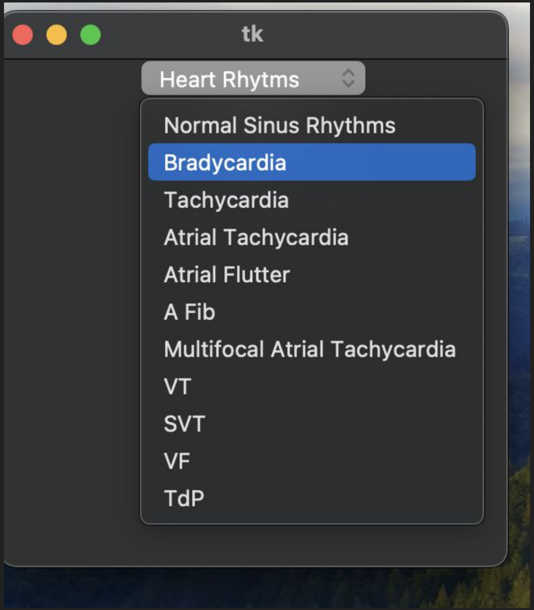


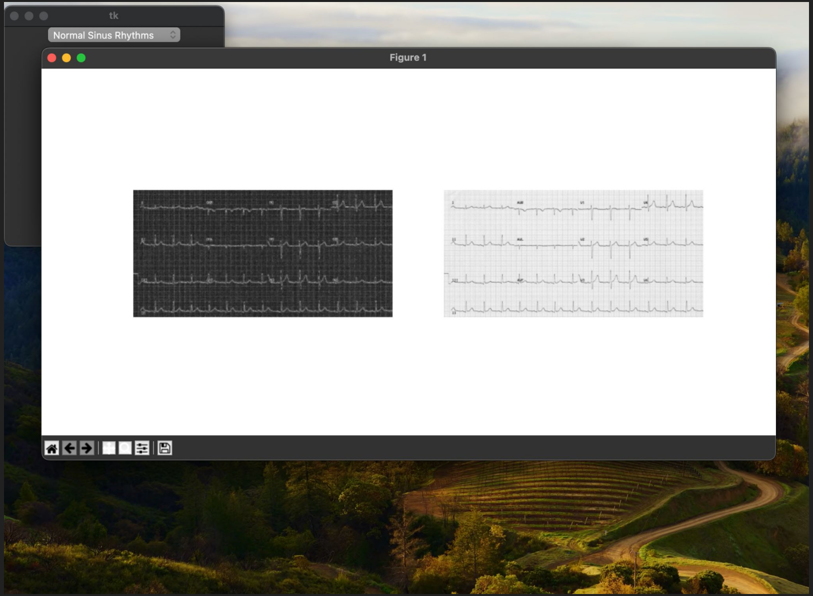
$$s_N(x) = A_0 + \sum_{n=1}^{N} \left(A_n \cos\left(2\pi \tfrac{n}{P} x\right) + B_n \sin\left(2\pi \tfrac{n}{P} x\right) \right)$$

GUI

- Python with tkinter for the gui development
- ECG window using tkinter

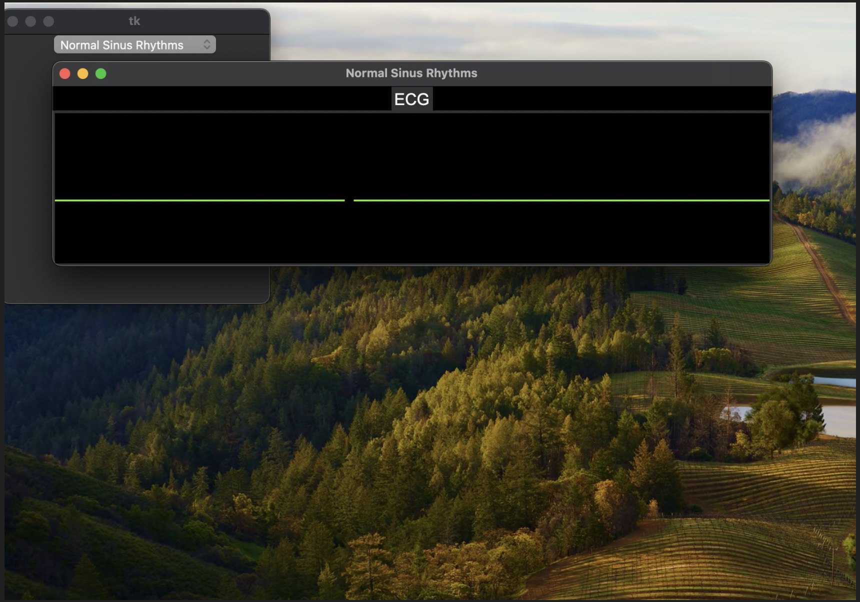


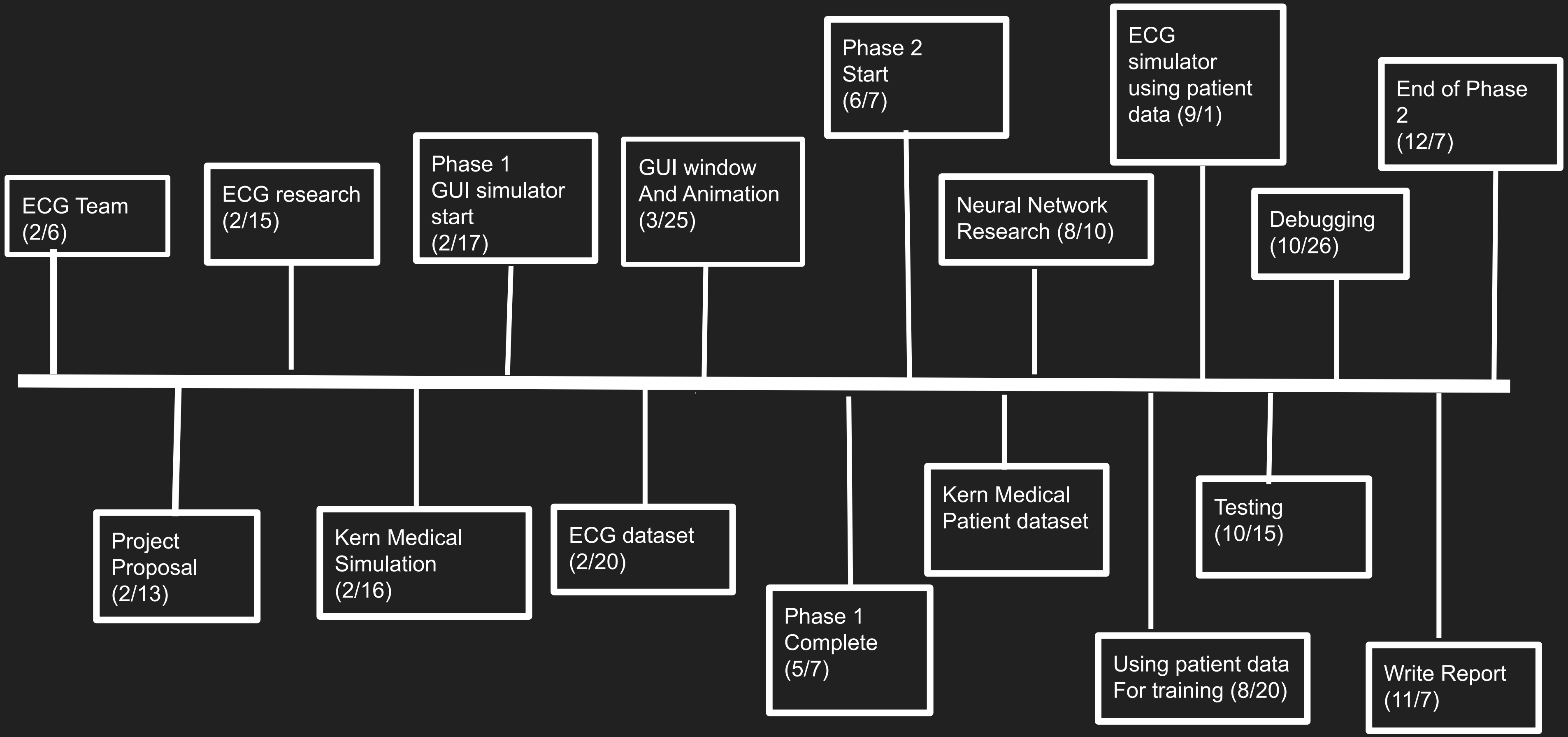




Phase 1

- Create functions that represent the waves
- Create multiple periods
- Animation for ECG
- ECG pattern
- Heart rate and other





ECG Team
(2/6)

Project
Proposal
(2/13)

ECG research
(2/15)

Kern Medical
Simulation
(2/16)

Phase 1
GUI simulator
start
(2/17)

ECG dataset
(2/20)

GUI window
And Animation
(3/25)

Phase 1
Complete
(5/7)

Phase 2
Start
(6/7)

Kern Medical
Patient dataset

Neural Network
Research (8/10)

Using patient data
For training (8/20)

ECG
simulator
using patient
data (9/1)

Testing
(10/15)

Debugging
(10/26)

Write Report
(11/7)

End of Phase
2
(12/7)