04-Phase_space_reconstruction_with_ECG_and_Poincare_plots

January 4, 2021

1 Phase space reconstruction from electrocardiogram recordings and Poincaré plots

1.0.1 Ana Daniela del Río Pulido and Erin C. McKiernan

1.0.2 Facultad de Ciencias, UNAM

In this practical, students already know the basics of electrocardiogram analysis and have passed throught he phase space recontruction notebook. They will learn how to make and interpret a Poincaré plot. Raw recordings for this practical can be collected by students, or they can work with existing recordings from our repository.

1.1 Human electrocardiogram

The human electrocardiogram (ecg) is a physiological signal that is measured by placing electrodes that record the heart's voltage changes. Each part of the obtained signal corresponds to a particular action of the heart [Boron and Boulpaep, 2012] [BackyardBrains, 2017] [Kantz and Schreiber, 2004, pg. 344], this can be seen in the following figure. One period of the electrical signal of the heart is called the PQRST wave complex.

2 Setting up the notebook

We begin by setting up the Jupyter notebook and importing the Python modules needed for plotting figures, create animations, etc. We include commands to view plots in the Jupyter notebook, and to create figures with good resolution and large labels. These commands can be customized to produce figures with other specifications.

[1]:
```python
# Imports python libraries
import numpy as np
import random as rd
import wave
import sys
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
sys.path.insert(1, r'./../functions') # add to pythonpath

#For plotting in 3D
from matplotlib import rc
from mpl_toolkits.mplot3d import Axes3D
```

```
# commands to create high-resolution figures with large labels
%config InlineBackend.figure_formats = {'png', 'retina'}
plt.rcParams['axes.labelsize'] = 16 # fontsize for figure labels
plt.rcParams['axes.titlesize'] = 18 # fontsize for figure titles
plt.rcParams['font.size'] = 14 # fontsize for figure numbers
plt.rcParams['lines.linewidth'] = 1.4 # line width for plotting
```

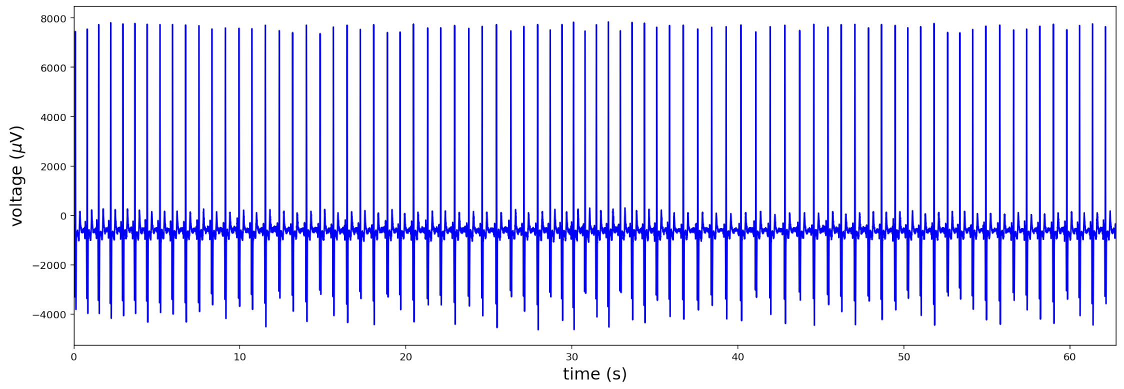
2.1 Extracting and graphing the data

ECG recordings were obtained using the Backyard Brains Heart and Brain Spiker Box. The recordings are saved as audio files in .wav format. The first thing we have to do is open the .wav files and extract the data. We can extract the number of recording channels, sampling rate, etc.

[2]:
```
#Function that extracts the number of recording channels, sampling rate, time␣
 ↪and signal
#variable is the path and filename of the .wav file
def ecg(variable):
    record = wave.open(variable, 'r') # load the data

    # Get the number of channels, sample rate, etc.
    numChannels = record.getnchannels() #number of channels
    numFrames = record.getnframes() #number of frames
    sampleRate = record.getframerate() #sampling rate
    sampleWidth = record.getsampwidth()

    # Get wave data
    dstr = record.readframes(numFrames * numChannels)
    waveData = np.frombuffer(dstr, np.int16)

    # Get time window
    timeECG = np.linspace(0, len(waveData)/sampleRate, num=len(waveData))

    return timeECG, waveData
```

Now, let's plot the raw ECG signal.

[3]:
```
#Obtaining data
timeECG, waveData = ecg("ECG_samples/S1_rest.wav")

# Plotting EMG signal
plt.figure(figsize=(18,6))
plt.xlabel(r'time (s)')
plt.ylabel(r'voltage ($\mu$V)')
plt.plot(timeECG,waveData, 'b')
plt.xlim(0,max(timeECG))
plt.show()
```



3 What do time series and electrocardiogram have in common?

Well, the ecg is precisely a time series of the heart's electric signals! It is very important to study the heart because the leading cause of death globall are cardiovascular diseases. 3 out of 10 deaths are because of heart disease [WHO, 2019]. If simpler techniques for studying the heart are developed, more people could be saved from heart failure. With this in mind, we will start uncovering the intricacies of the heart's dynamics.

4 Heart phase space reconstruction in 2D

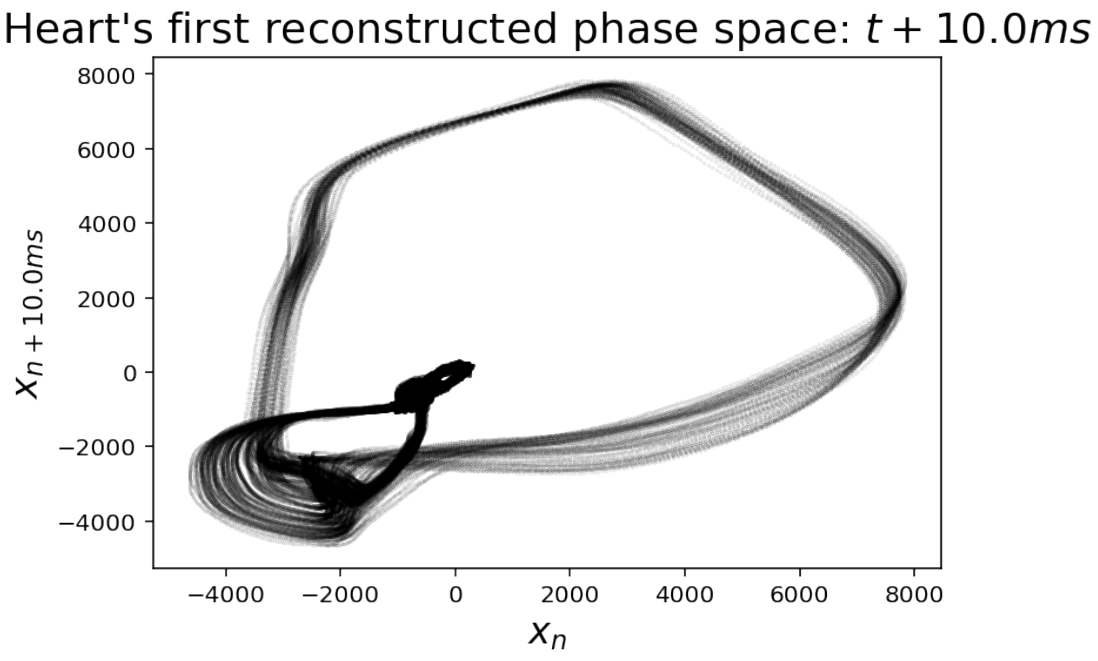
As the time delays are an arbitrary election, we can choose whichever we want! The only objective of this time lag is to unfold the dynamics of the attractor in phase space.

Let us generate a function that will reconstruct the phase space for a certain time delay. The time series we will be working with is the heart's change in voltage. In the following plots, the axes are x_n, which simply represents voltage.

[4]:
```
#Generating a function that will reconstruct the phase space for a certain time␣
↪delay
# data_series is the voltage of our signal
# period is the time delay
# identifier is a string that will help us identify that particular graph
def graph_one(data_series, period = 210, identifier = "xx"):
    time = period*0.1 #time is in miliseconds
    n = np.size(data_series) #size of the voltage vector

    plt.figure(2)
    plt. plot(data_series[0: n-period], data_series[period: n],
              marker = "o", markersize = 0.05, linewidth = 0.005, color =␣
↪"black")
    plt.title(identifier+r": $t+"+str(time)+"ms$")
    plt.xlabel(r"$x_n$")
    y = r'$x_{n+'+str(time)+"ms}$"
    plt.ylabel(y)
    plt.show()
    return None
```

Trying out this function.

[5]:
```
graph_one(waveData, 100, "Heart's first reconstructed phase space")
```



4.0.1 Exercise: reconstruct phase space with at least 5 different time lags. What can you say about graph's changes?

[ ]:
```
```

5 Reproducing Petr Saparin's work

Now that we saw that the function works, let us reproduce Petr Saparin's work. These time delays are: 1.3 ms, 12.5 ms, 25 ms and 125 ms.

As there is no theoretically accurate time lag, people have used different time lags to study the heart attractor. In some studies, they use this time lag to be the 25% of the ventricular fibrillation cycle length, i.e. 25 ms [Umapathy et al., 2010, pg. 109] [Gray et al., 1998, pg. 78]. In the following figure, we can see the reconstructed heart's phase space. Notice how every cycle lies in these well defined shapes. We can talk specifically about attractor recontruction in Saparin's work because of the clear dynamics in phase space. The figures below were made with the following time lags:

- Top left: = 1.25ms data is close to the diagonal because consecutive values are very similar.
- Top right: = 12.5ms large loop corresponds to QRS complex.
- Bottom left: = 25ms slower features like P and T waves.
- Bottom right: = 125ms complicated attractors.

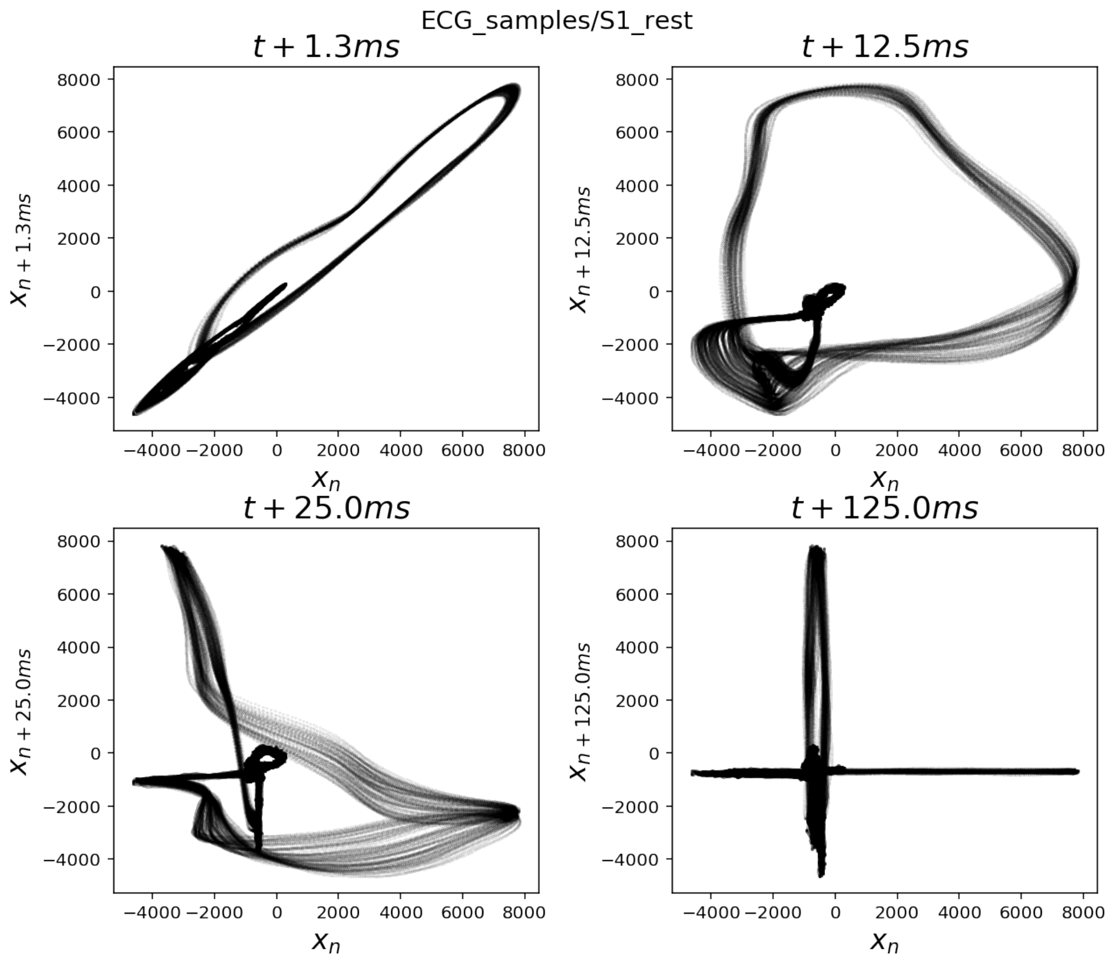
The previous figure is taken from [Kantz and Schreiber, 2004, pg. 40]. Originally from Petr Saparin (1995).

```
[6]: def graphic_fun(data_series, time_delays = [13, 125, 250, 1250], identifier =␣
     ↪"xx"):
         time = time_delays[0]*0.1
         n = np.size(data_series)
         fig, axs = plt.subplots(2,2,figsize=(9,9))

         st = fig.suptitle(identifier, fontsize="x-large")
         axs[0, 0].plot(data_series[0: n-time_delays[0]], data_series[time_delays[0]:
     ↪ n],
                   marker = "o", markersize = 0.05, linewidth = 0.005, color =␣
     ↪"black")
         axs[0, 0].set_title(r"$t+"+str(time)+"ms$")
         axs[0, 0].set_xlabel(r"$x_n$")
         y = r'$x_{n+'+str(time)+"ms}$"
         axs[0, 0].set_ylabel(y)

         time = time_delays[1]*0.1
         axs[0, 1].plot(data_series[0: n-time_delays[1]], data_series[time_delays[1]:
     ↪ n],
                   marker = "o", markersize = 0.05, linewidth = 0.005, color =␣
     ↪"black")
         axs[0, 1].set_title(r"$t+"+str(time)+"ms$")
         axs[0, 1].set_xlabel(r"$x_n$")
         y = r'$x_{n+'+str(time)+"ms}$"
         axs[0, 1].set_ylabel(y)

         time = time_delays[2]*0.1
         axs[1, 0].plot(data_series[0: n-time_delays[2]], data_series[time_delays[2]:
     ↪ n],
                   marker = "o", markersize = 0.05, linewidth = 0.005, color =␣
     ↪"black")
         axs[1, 0].set_title(r"$t+"+str(time)+"ms$")
         axs[1, 0].set_xlabel(r"$x_n$")
         y = r'$x_{n+'+str(time)+"ms}$"
         axs[1, 0].set_ylabel(y)
```

```
    time = time_delays[3]*0.1
    axs[1, 1].plot(data_series[0: n-time_delays[3]], data_series[time_delays[3]:
↪ n],
                marker = "o", markersize = 0.05, linewidth = 0.005, color =␣
↪"black")
    axs[1, 1].set_title(r"$t+"+str(time)+"ms$")
    axs[1, 1].set_xlabel(r"$x_n$")
    y = r'$x_{n+'+str(time)+"ms}$"
    axs[1, 1].set_ylabel(y)

    #Shifts down the subplots
    fig.tight_layout()
    st.set_y(0.85)
    fig.subplots_adjust(top=0.8)
    return None
```

Trying out the previous function.

```
[7]: name = "ECG_samples/S1_rest"
     time_delays = [13, 125, 250, 1250]

     timeECG, waveData = ecg(name+".wav")
     graphic_fun(waveData, time_delays, name)
```



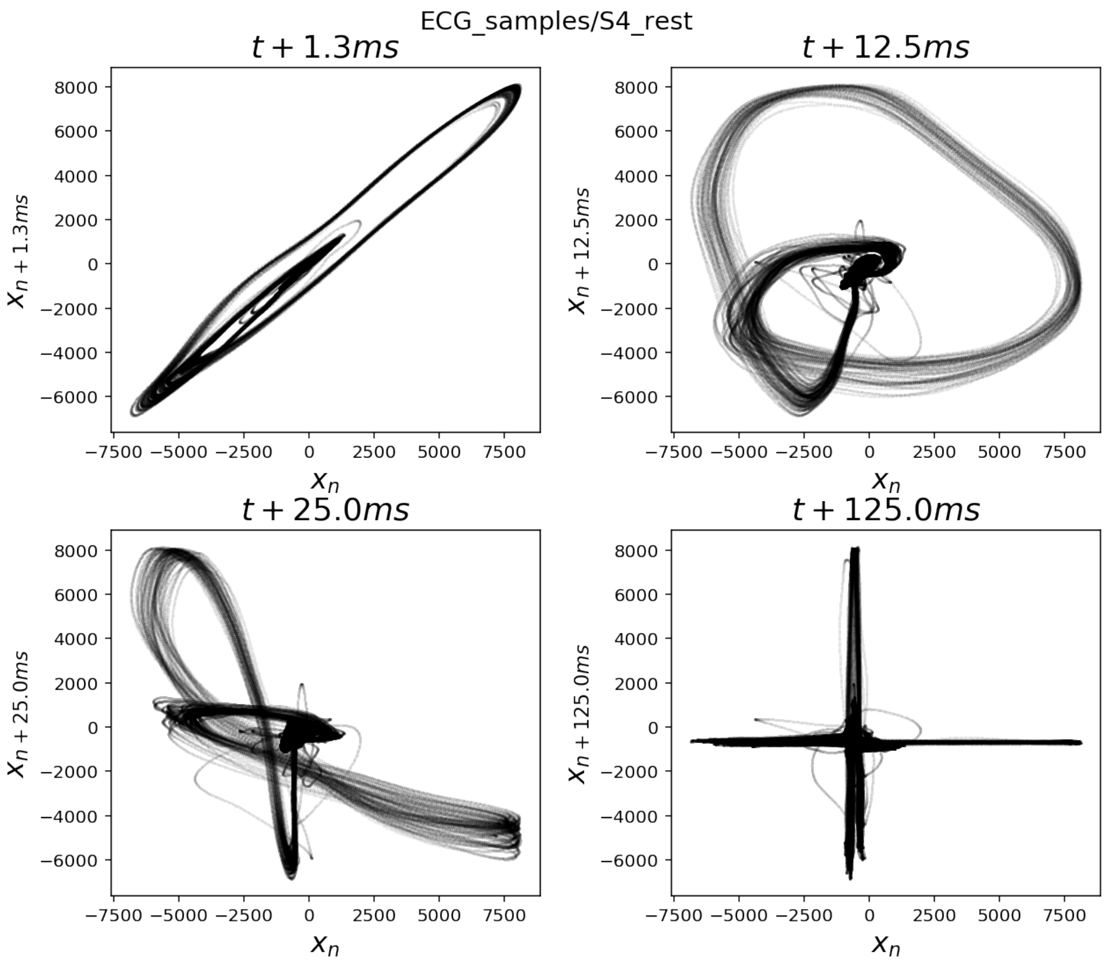
5.0.1 Exercise: With the previous exercise, can you say there is an idea time lag? Explain.

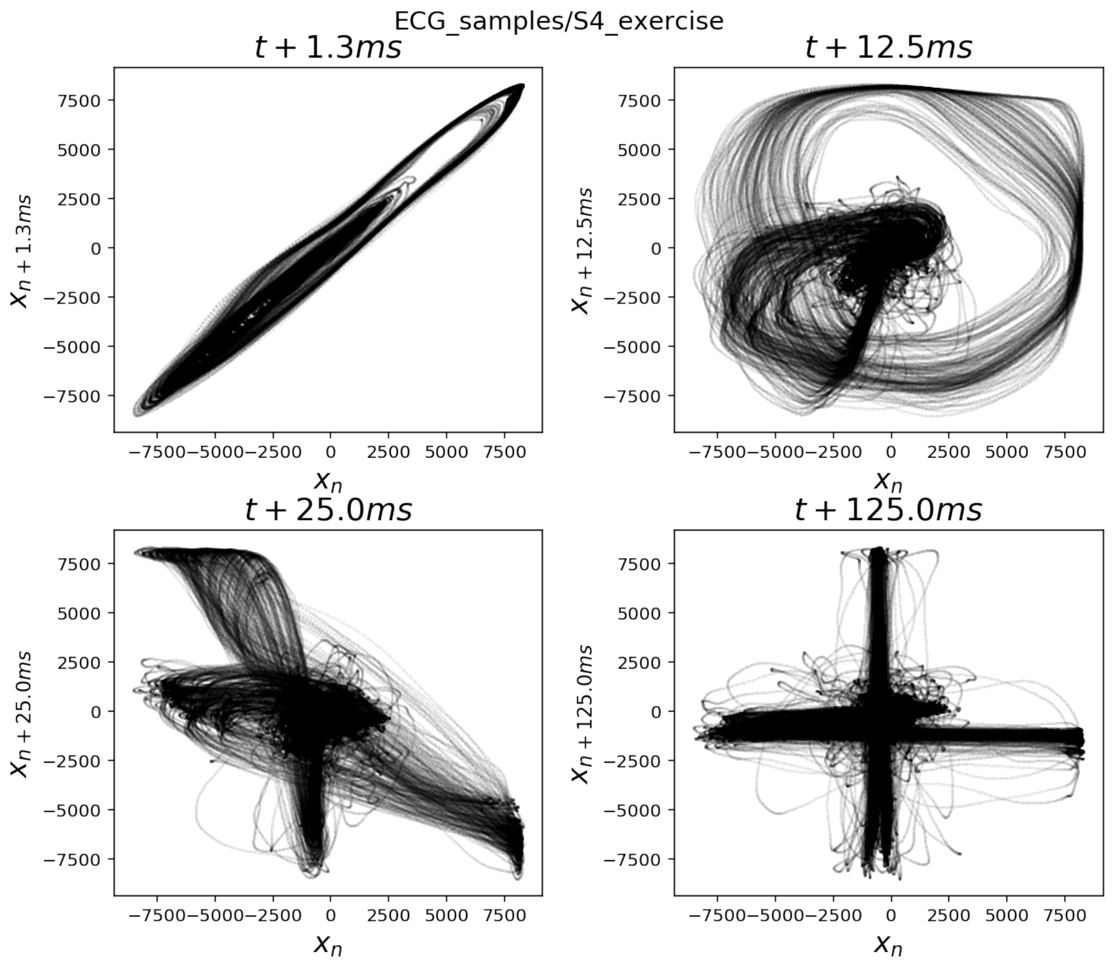
```
[ ]:
```

6 Rest vs. Exercise

Are you curious to see what happens to the phase space of a person after doing exercise? Try running the following code.

```
[8]: identifiers = ["ECG_samples/S4_rest", "ECG_samples/S4_exercise"]
     time_delays = [13, 125, 250, 1250]

     for i in range(0, np.size(identifiers)):
         timeECG, waveData = ecg(identifiers[i]+".wav")
         graphic_fun(waveData, time_delays, identifiers[i])
```


ECG_samples/S4_rest
$t+1.3ms$
$t+12.5ms$
$t+25.0ms$
$t+125.0ms$
$x_{n+1.3ms}$
$x_{n+12.5ms}$
$x_{n+25.0ms}$
$x_{n+125.0ms}$
x_n
8000
6000
4000
2000
0
−2000
−4000
−6000
−7500
−5000
−2500
0
2500
5000
7500

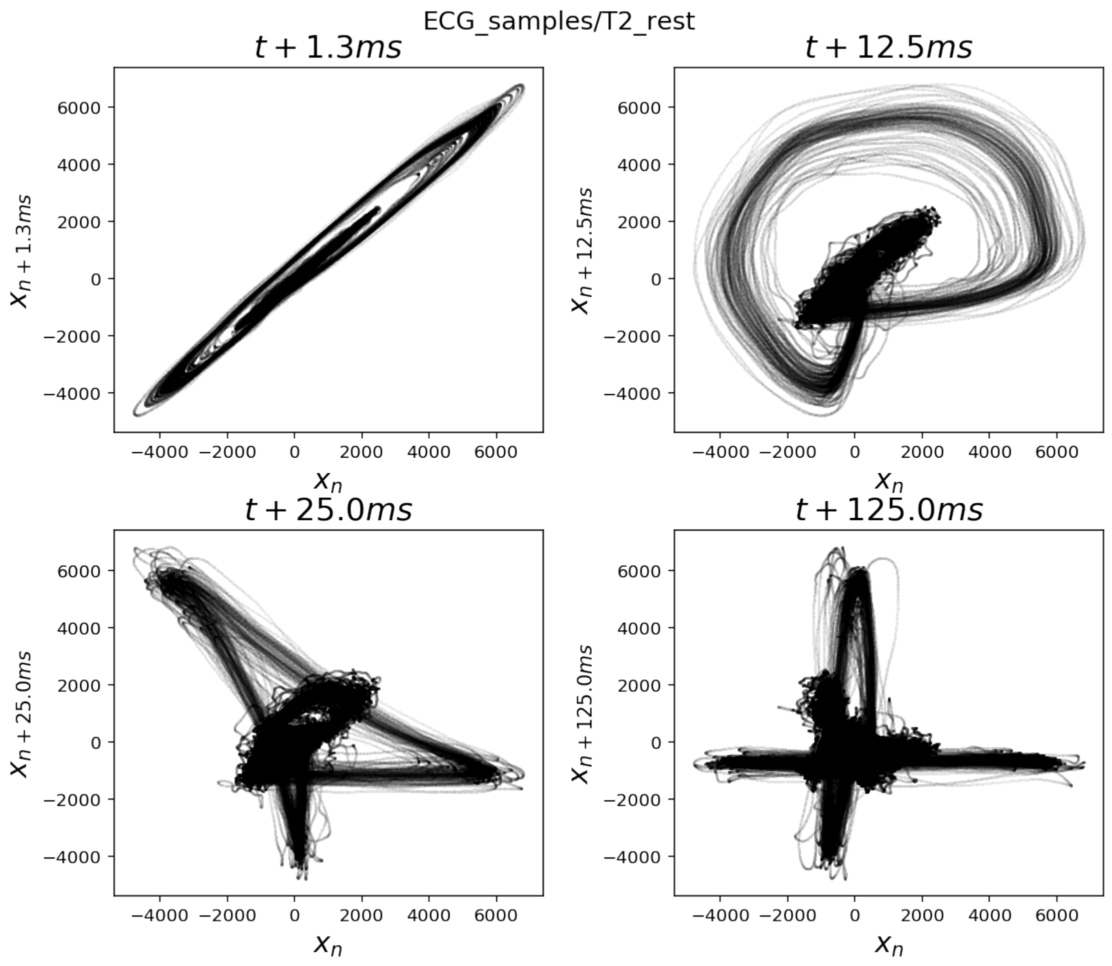


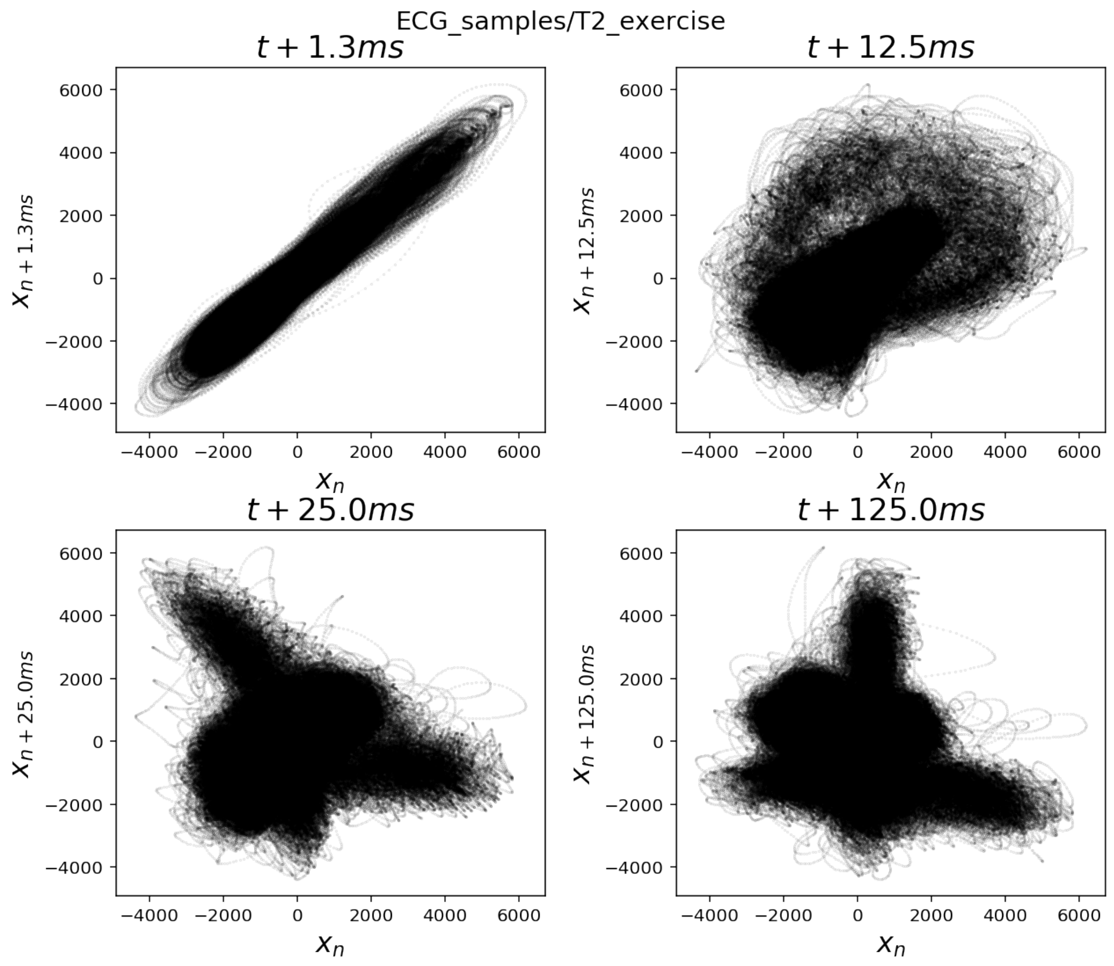
The trayectories in phase space seem to have spread out after the individual does exercise. In other words, the trajectories seem to have a wider range of possibilities than in the resting state.

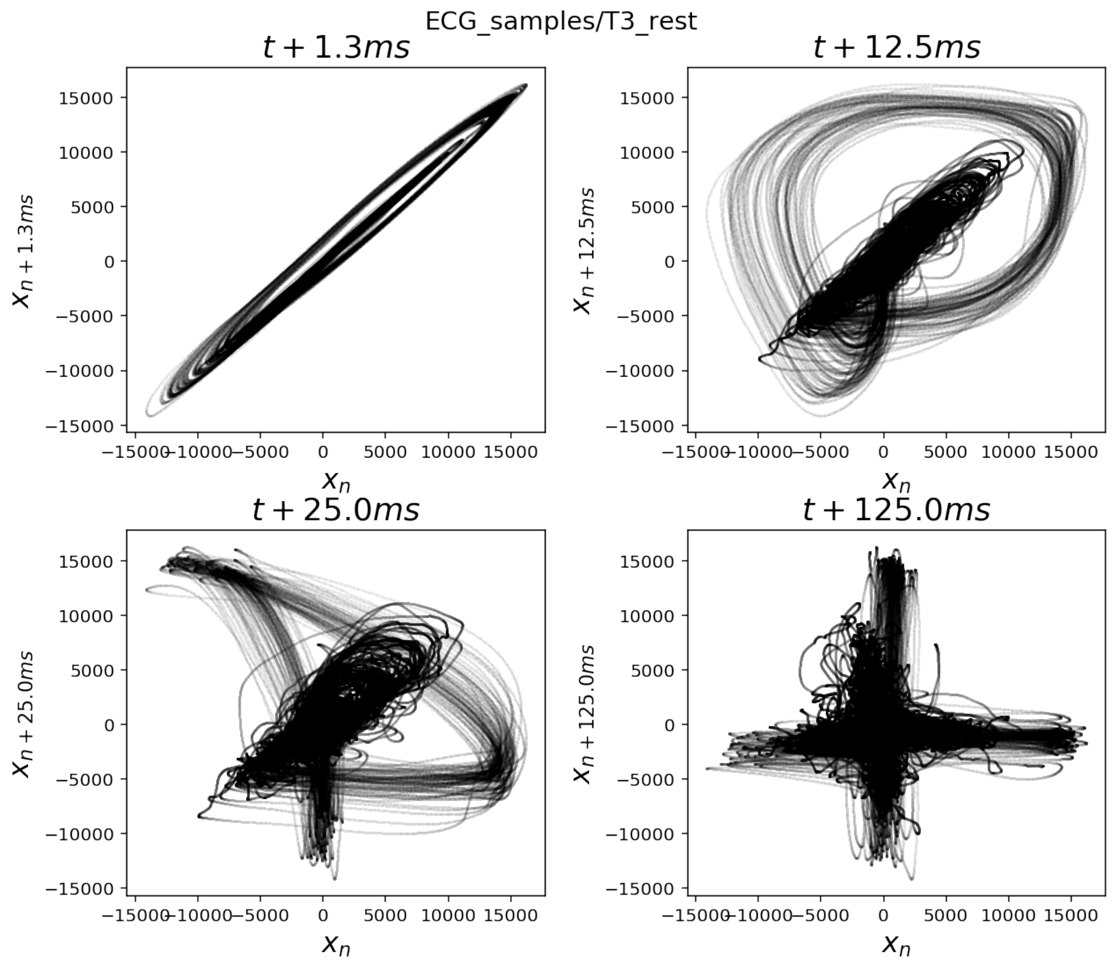
7 Early indicator of heart disease?

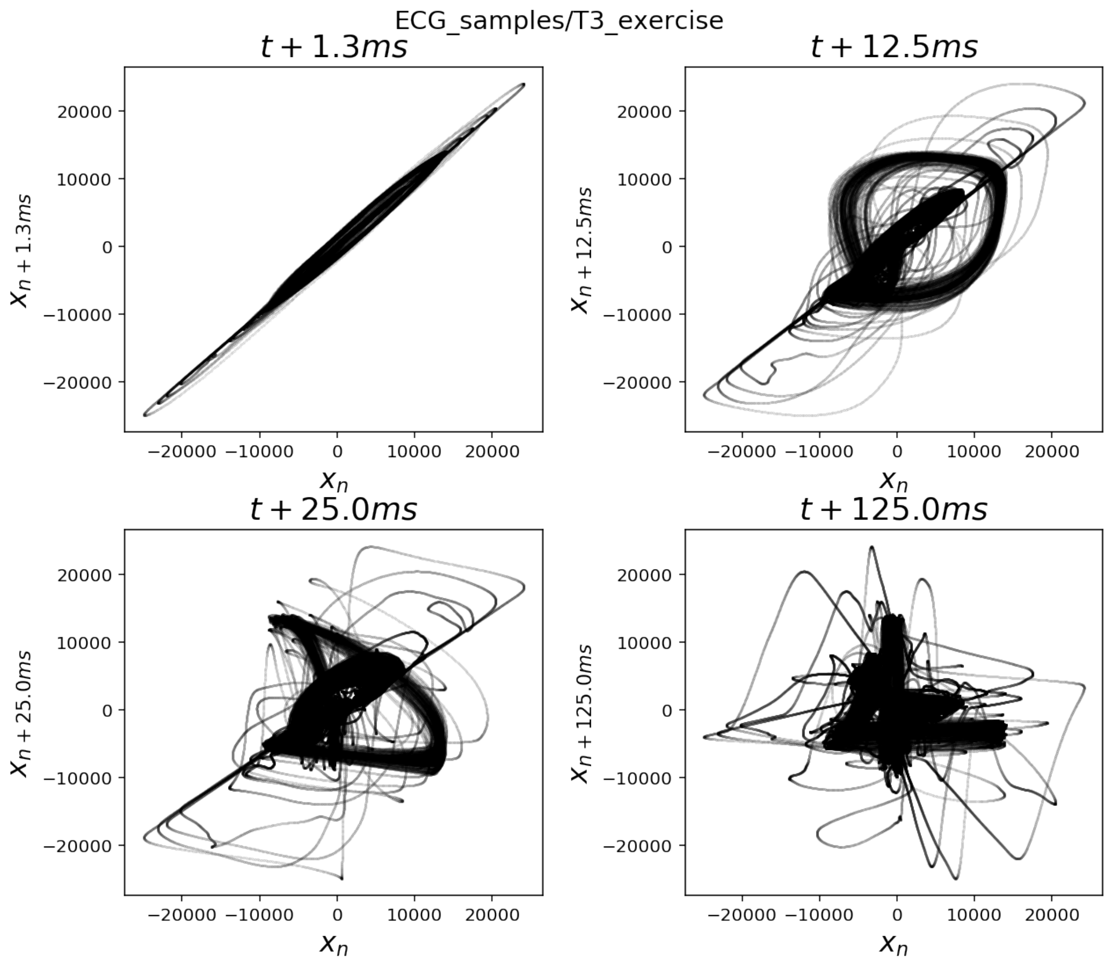
Run the following code. Notice if there is any difference with the reconstructed phase spaces you have previously observed.

[9]:
```
identifiers = ["ECG_samples/T2_rest", "ECG_samples/T2_exercise", "ECG_samples/
 ↪T3_rest", "ECG_samples/T3_exercise"]
time_delays = [13, 125, 250, 1250]

for i in range(0, np.size(identifiers)):
    timeECG, waveData = ecg(identifiers[i]+".wav")
    graphic_fun(waveData, time_delays, identifiers[i])
```


ECG_samples/T2_rest
$t + 1.3ms$
$t + 12.5ms$
$t + 25.0ms$
$t + 125.0ms$
$x_{n+1.3ms}$
$x_{n+12.5ms}$
$x_{n+25.0ms}$
$x_{n+125.0ms}$
x_n
−4000
−2000
0
2000
4000
6000


ECG_samples/T2_exercise
$t + 1.3ms$
$t + 12.5ms$
$t + 25.0ms$
$t + 125.0ms$
$x_{n+1.3ms}$
$x_{n+12.5ms}$
$x_{n+25.0ms}$
$x_{n+125.0ms}$
x_n
−4000
−2000
0
2000
4000
6000


ECG_samples/T3_rest
$t + 1.3ms$
$t + 12.5ms$
$t + 25.0ms$
$t + 125.0ms$
$x_{n+1.3ms}$
$x_{n+12.5ms}$
$x_{n+25.0ms}$
$x_{n+125.0ms}$
x_n
15000
10000
5000
0
−5000
−10000
−15000



T2 is an individual who had the fastest heart rate of a group of around 6 people (*circa* 140 bpm after performing exercise). Notice how after doing exercise, the reconstructed phase space looks like a mass of uncorrelated dots. In this case, we cannot talk about an attractor because there is no defined shape in phase space which the heart follows each time. This could be thought of as an early indicator of heart disease.

T3 is an individual who had a prescription for consuming vasodilators, but did not take them. Notice how the dynamics of this heart changes after doing exercise. This is the only case where the scale for representing the heart's dynamics doubles after performing exercise. This could be a case of how one type of cardiac arrhythmia looks in state space. Even though we know that this person has a heart condition, we cannot really say if the dynamics we are observing in the phase space is due to his condition or other variables. What we can say is that the dynamics in the recontructed phase space looks different than that of other recordings.

8 Heart's phase space reconstruction in 3D

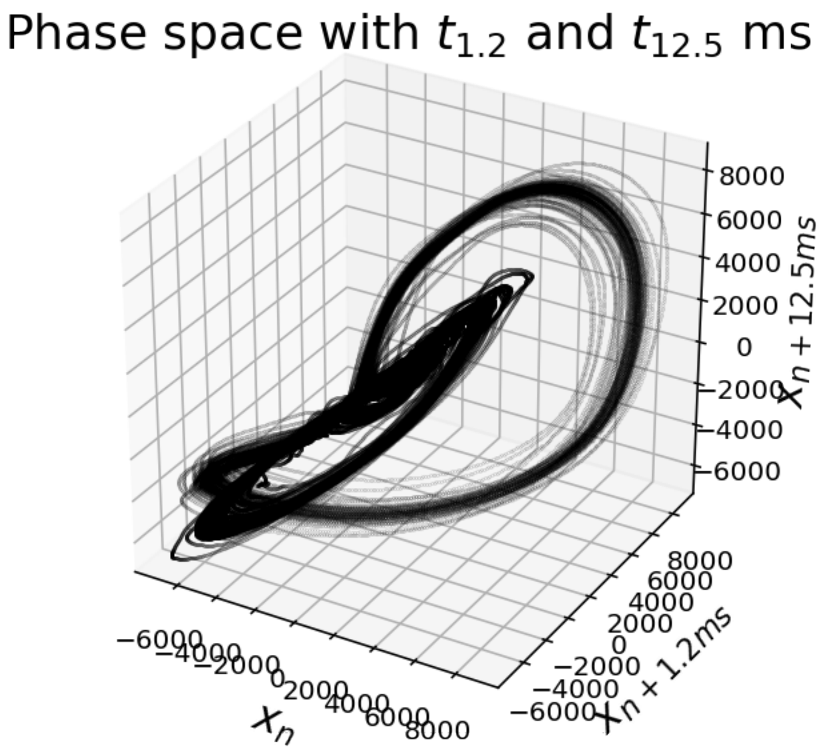
We limit the reconstructed phase space to a 2 dimensional space because it is what we can visualize in a screen. Nevertheless, we know that it is N-dimensional. With current computer software, we can recreate some 3D graphs of this pahse space. The time delays used here are all the possible combinations of the time delays used for the 2D graphs created above. This individual had a healthy and active lifestyle. The recording is at a resting state.

If you are not familiar with plotting in 3 dimensions, you might want to take a look at the following link: https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html

```
[10]: def graph_3d(data_series, period1, period2): #Making graphs in 3D for phase␣
      ↪space
          # The vectors should be of the same size, so the last elements will be␣
      ↪deleted in case there are extra for a certain vector
          time1 = round(period1*0.1, 2)
          time2 = round(period2*0.1, 2)

          n = np.size(data_series)
          x = data_series[0: n-period2]
          y = data_series[period1: n-(period2-period1)]
          z = data_series[period2: n]

          fig = plt.figure(figsize=(5,5))
          ax = fig.gca(projection='3d')
          ax.set_xlabel(r"$x_n$")
          ax.set_ylabel(r'$x_{n+'+str(time1)+"ms}$")
          ax.set_zlabel(r'$x_{n+'+str(time2)+"ms}$")
          ax.plot(x, y, z, marker = "o", markersize = 0.05, linewidth = 0.005, color␣
      ↪= "black")
          lbl = r'Phase space with $t_{' + str(time1) +"}$ and $t_{"+ str(time2) +␣
      ↪"}$ ms"
          plt.title(lbl)
          plt.show()

          return None
```

Trying the previous function.

```
[11]: timeECG, waveData = ecg("ECG_samples/T1_rest.wav")
      graph_3d(waveData, 12, 125)
```

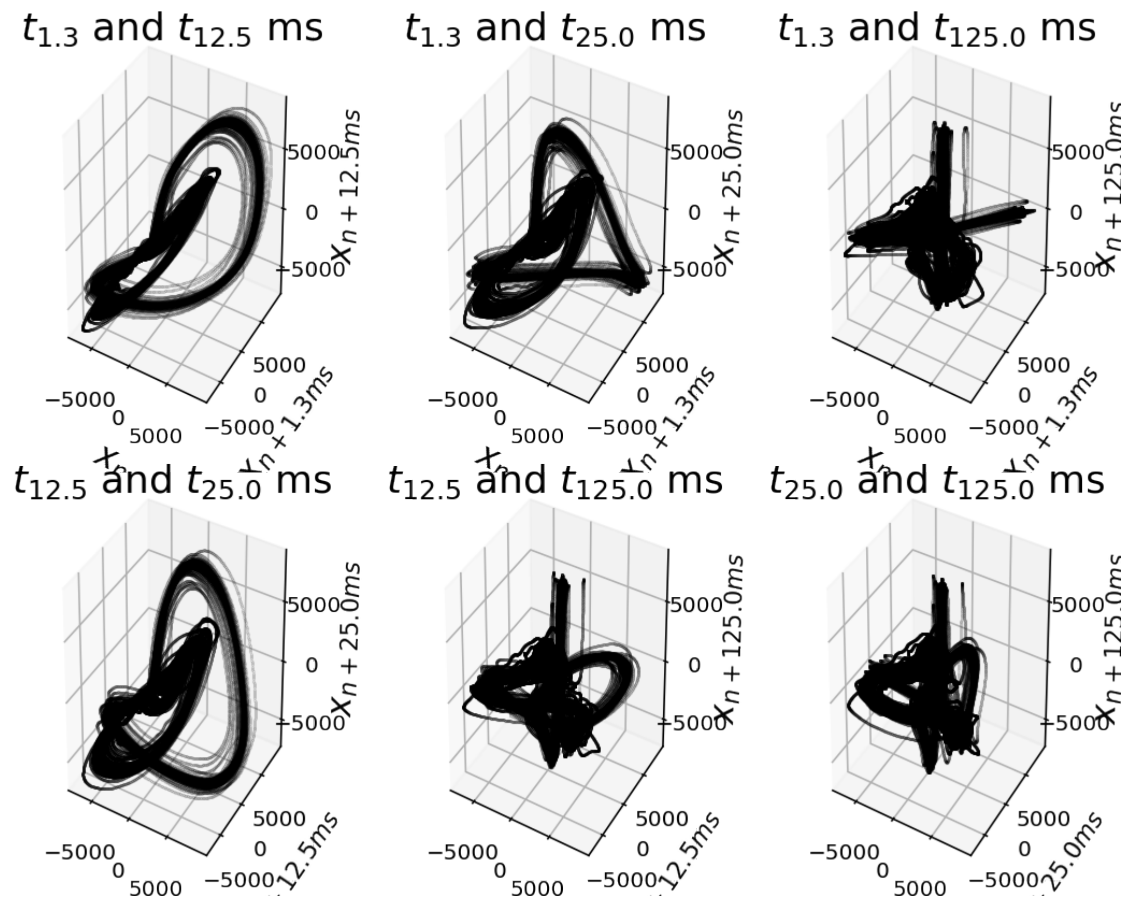


Now, we want a single function that plots various combinations of graphs.

```
[12]: def graphi_many_3d(time_delays = [13, 125, 250, 1250], identifier = "xx"):

          timeEMC, data_series = ecg(identifier)
          n = np.size(data_series)

          fig = plt.figure(figsize=(7.5, 6))
          counter = 1

          for i in time_delays:
              for j in time_delays:
                  if i != j and i < j: #Each pair of possible combiantions with no␣
       ↪repeats
                          time1 = round(i*0.1, 2)
                          time2 = round(j*0.1, 2)

                          n = np.size(data_series)
                          x = data_series[0: n-j]
                          y = data_series[i: n-(j-i)] #Notice that j>i always, so no need␣
       ↪for abs
                          z = data_series[j: n]
```

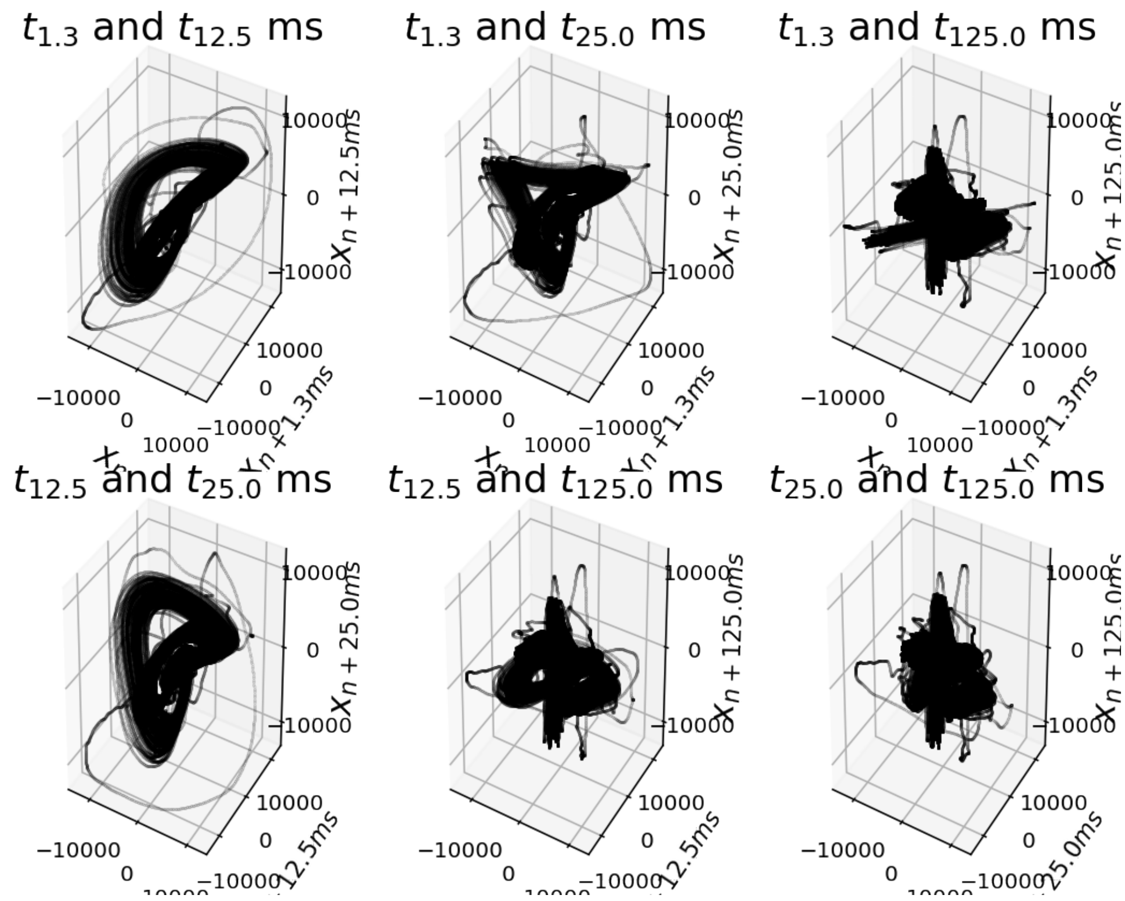
```
                a = 230+counter

                ax = fig.add_subplot(a, projection='3d')
                ax.set_xlabel(r"$x_n$")
                ax.set_ylabel(r'$x_{n+'+str(time1)+"ms}$")
                ax.set_zlabel(r'$x_{n+'+str(time2)+"ms}$")
                ax.plot(x, y, z, marker = "o", markersize = 0.05, linewidth = 0.
↪005, color = "black")
                lbl = r'$t_{' + str(time1) +"}$ and $t_{"+ str(time2) + "}$ ms"
                ax.set_title(lbl)
                counter += 1
    fig.tight_layout()
    plt.show()
    return None
```

[13]:
```
graphi_many_3d([13, 125, 250, 1250], "ECG_samples/T1_rest.wav")
```



Now, what do you think will happen to the heart's phase space after this person does exercise?

```
[14]: graphi_many_3d([13, 125, 250, 1250], "ECG_samples/T1_exercise.wav")
```



9 Poincaré plots

Poincaré plots have been used in electrocardiogram analysis in the literature since XXX.

A Poincaré plot is basically a scatter plot of a time series. In one axis you plot the current entry and in the other axis you plot the immediate next value. Does this remind you of a reconstructed phase space?

This plot gives you a comparison of how alike are two consecutive values of a time series. We imagine that the Poincaré plot must depend first in the sampling rate. If the sampling rate is high enough compared to the time at which your phenomena is occuring, then two consecutive points must be near each other. Thus, the points must lie near the identity:

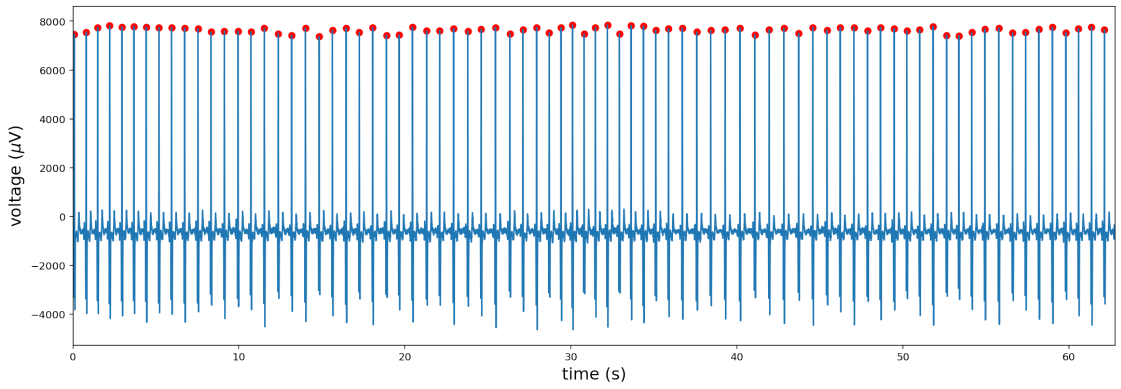
$$f(x) = x$$

If you sit down a think a while about this technique, it is basically the reconstruction of phase space. Nevertheless, there are whole books about this topic that never mention phase space!

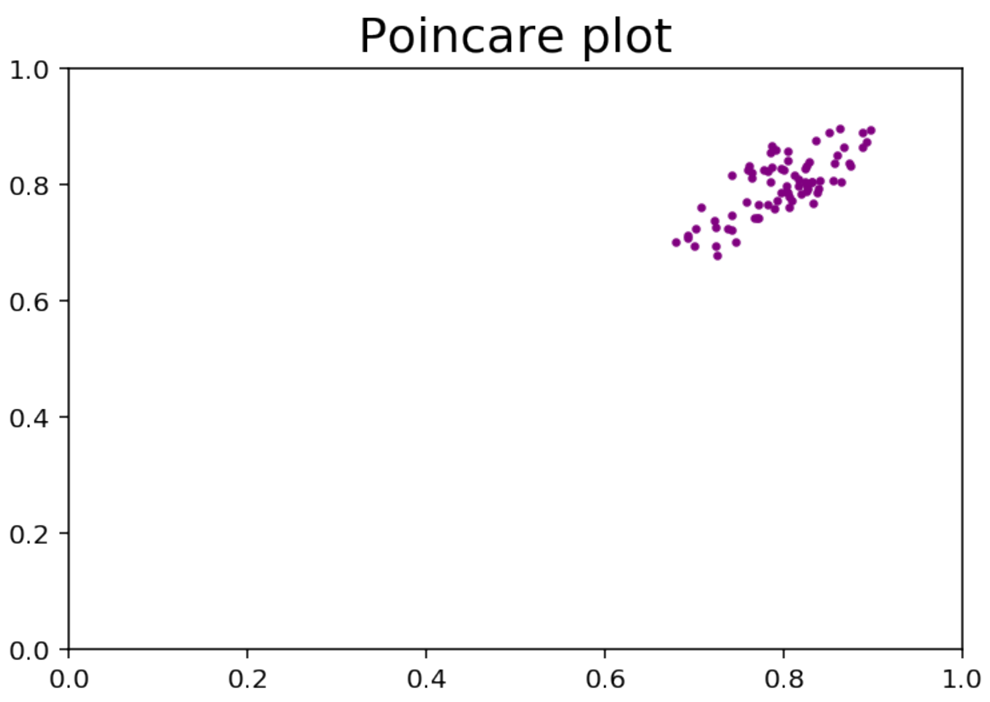
[15]:
```
# Functions for detecting R peaks

def detects_local_maxima(timeECG, waveData, threshold_ratio=0.7):
    # If not all the R peaks are detected, lower the threshold_ratio
    # If components that are not R peaks (like T waves) are detected, higher␣
 ↪the threshold_ratio

    if len(timeECG) != len(waveData): #Raises an error if the two arrays have␣
 ↪different lengths
        raise Exception("The two arrays have different lengths.")

    interval = max(waveData) - min(waveData)
    threshold = threshold_ratio*interval + min(waveData)
    maxima = []
    maxima_indices = []
    mxs_indices = []
    banner = False

    for i in range(0, len(waveData)):

        if waveData[i] >= threshold:#If a threshold value is surpassed,
            # the indices and values are saved
            banner = True
            maxima_indices.append(i)
            maxima.append(waveData[i])

        elif banner == True and waveData[i] < threshold: #If the threshold␣
 ↪value is crossed
            # the index of the maximum value in the original array is saved
            index_local_max = maxima.index(max(maxima))
            mxs_indices.append(maxima_indices[index_local_max])
            maxima = []
            maxima_indices = []
            banner = False

    return mxs_indices

# If the input of this function is time, the intervals will be given in those␣
 ↪same units
# Obtaining the indexes at which the R peaks occur.
def R_intervals(time_indices):
    length = len(time_indices)
    intervals = np.zeros(length-1)

    for i in range(0, length-1):
        intervals[i] = time_indices[i+1]-time_indices[i]
    return intervals
```

[16]:
```
timeECG, waveData = ecg("ECG_samples/S1_rest.wav")
mxs_indices = detects_local_maxima(timeECG, waveData)
xx = R_intervals(timeECG[mxs_indices])
timeRpeaks = timeECG[mxs_indices]
```

[17]:
```
plt.figure(figsize=(18,6))
plt.xlabel(r'time (s)')
plt.ylabel(r'voltage ($\mu$V)')
plt.xlim(min(timeECG),max(timeECG))
plt.plot(timeECG, waveData)
plt.scatter(timeECG[mxs_indices], waveData[mxs_indices], color='r')
plt.show()
```



[18]:
```
plt.scatter(xx[1:-1], xx[:-2], s = 5, c ="purple")
plt.xlim(0,1)
plt.ylim(0,1)
plt.title("Poincare plot")
plt.show()
```



Surprisingly, there are a lot of articles published that mention Poincaré plots, but not as many that mention phase space. People have tried to obtain heart rate variability measurements from these plots. For example, the distribution of these points in a certain axis, or fitting an elipse that surrounds the data considering the standard deviation in the $f(x) = x$ axis and in its ortogonal axis. Actually, there are some articles that mention the benefits of using statistics for measuring some parameters in the Poincaré plot, but there are others that mention its pitfalls.

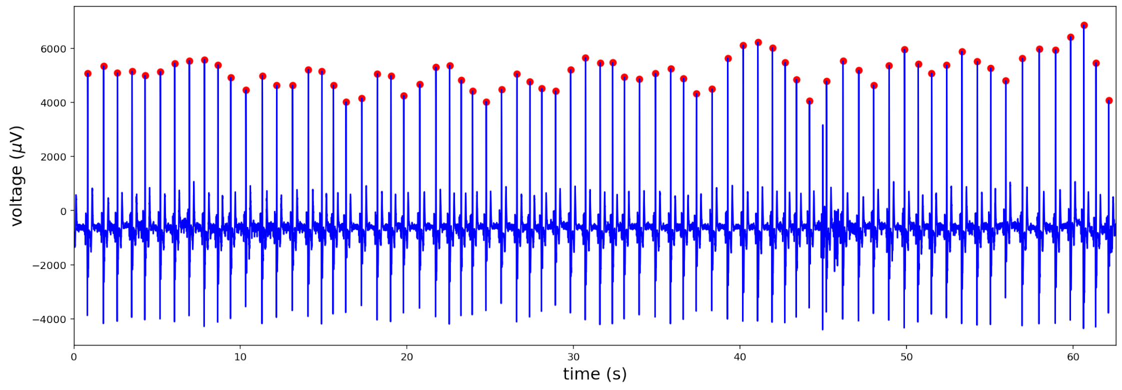
Let's use the same technique for analizying another recording.

[19]:
```
#Obtaining data
timeECG, waveData = ecg("ECG_samples/S2_rest.wav")
```

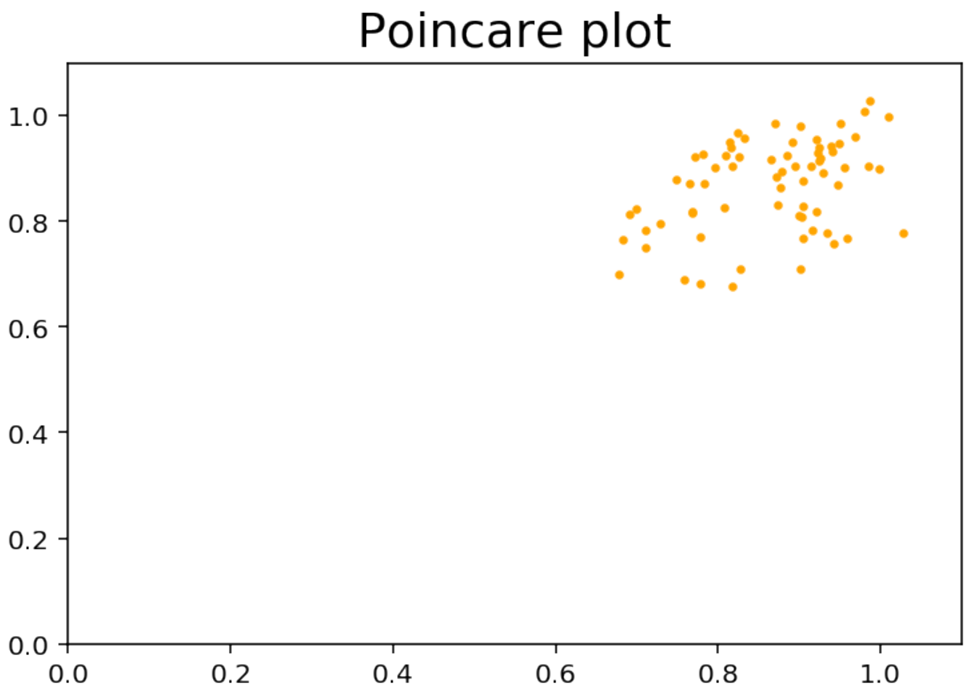
[20]:
```
mxs_indices = detects_local_maxima(timeECG, waveData)
yy = R_intervals(timeECG[mxs_indices])
timeRpeaks = timeECG[mxs_indices]
```

We must make sure that the function `detects_local_maxima` is actually detecting the R peaks.

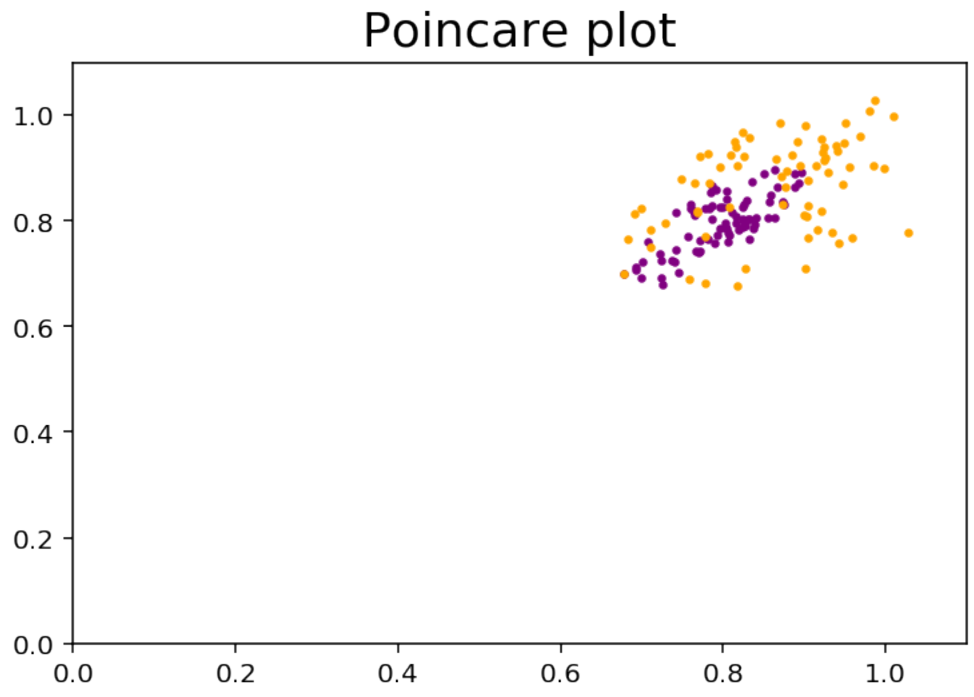
[21]:
```
# Plotting EMG signal
plt.figure(figsize=(18,6))
plt.xlabel(r'time (s)')
plt.ylabel(r'voltage ($\mu$V)')
plt.plot(timeECG, waveData, 'b')
plt.scatter(timeECG[mxs_indices], waveData[mxs_indices], color='r')
plt.xlim(0,max(timeECG))
plt.show()
```



[22]:
```
plt.scatter(yy[1:-1], yy[:-2], s = 5, c ="orange")
plt.xlim(0,1.1)
plt.ylim(0,1.1)
plt.title("Poincare plot")
plt.show()
```



[23]:
```
plt.scatter(xx[1:-1], xx[:-2], s = 5, c ="purple")
plt.scatter(yy[1:-1], yy[:-2], s = 5, c ="orange")
plt.xlim(0,1.1)
plt.ylim(0,1.1)
plt.title("Poincare plot")
plt.show()
```



Notice how the distribution and shape of the dot cloud is different comparing both recordings. In general, a greater variability or distance between the dots is related with a better health. Of course, not to fall into an extreme because if not we could be dealing with an arrythmia.

What do you imagine will happen to a Poincaré plot after a subject does exercise? ### Exercise: Find out what happens to a Poincaré plot after doing exercise? Consider that the R peak selection must be done correctly for obtaining a correct Poincaré plot.

[ ]:

```

```

If you are interested in knowing more about Poincaré plots, here is a list of articles and a book (denoted by a *) about Poincaré plots.

- Brennan, M., Palaniswami, M., & Kamen, P. (2001). Do existing measures of Poincare plot geometry reflect nonlinear features of heart rate variability?. IEEE transactions on biomedical engineering, 48(11), 1342-1347.
- Brennan, M., Palaniswami, M., & Kamen, P. (2002). Poincare plot interpretation using a physiological model of HRV based on a network of oscillators. American Journal of Physiology-Heart and Circulatory Physiology, 283(5), H1873-H1886.
- Carrasco, MJ Gaitán, R. González, O. Yánez, S. (2001). Correlation among Poincare plot

indexes and time and frequency domain measures of heart rate variability. Journal of medical engineering & technology, 25(6), 240-248.

- de Figueiredo, J. C. B., & Furuie, S. S. (2001, May). Using statistical distances to detect changes in the normal behavior of ECG-Holter signals. In Medical Imaging 2001: Ultrasonic Imaging and Signal Processing (Vol. 4325, pp. 557-565). International Society for Optics and Photonics.
- Guzik, P., Piskorski, J., Krauze, T., Schneider, R., Wesseling, K. H., Wykretowicz, A., & Wysocki, H. (2007). Correlations between Poincaré plot and conventional heart rate variability parameters assessed during paced breathing. The Journal of Physiological Sciences, 0702020009-0702020009.
- Hnatkova, K., Copie, X., Staunton, A., & Malik, M. (1995). Numeric processing of Lorenz plots of RR intervals from long-term ECGs: comparison with time-domain measures of heart rate variability for risk stratification after myocardial infarction. Journal of electrocardiology, 28, 74-80.
- Hoshi, R. A., Pastre, C. M., Vanderlei, L. C. M., & Godoy, M. F. (2013). Poincaré plot indexes of heart rate variability: relationships with other nonlinear variables. Autonomic Neuroscience, 177(2), 271-274.
- Hundewale, N. (2012). The application of methods of nonlinear dynamics for ECG in Normal Sinus Rhythm. International Journal of Computer Science Issues (IJCSI), 9(1), 458.
- Kamen, P. W., & Tonkin, A. M. (1995). Application of the Poincaré plot to heart rate variability: a new measure of functional status in heart failure. Australian and New Zealand journal of medicine, 25(1), 18-26.
- Kamen, P. W., Krum, H., & Tonkin, A. M. (1996). Poincare plot of heart rate variability allows quantitative display of parasympathetic nervous activity in humans. Clinical science, 91(2), 201-208.
- *Khandoker, A. H., Karmakar, C., Brennan, M., Palaniswami, M., & Voss, A. (2013). Poincaré plot methods for heart rate variability analysis. Boston, MA, USA: Springer US.
- Mourot, L., Bouhaddi, M., Perrey, S., Cappelle, S., Henriet, M. T., Wolf, J. P., ... & Regnard, J. (2004). Decrease in heart rate variability with overtraining: assessment by the Poincare plot analysis. Clinical physiology and functional imaging, 24(1), 10-18.
- Parvaneh, S., Hashemi Golpayegani, M. R., Firoozabadi, M., & Haghjoo, M. (2012). Predicting the spontaneous termination of atrial fibrillation based on Poincare section in the electrocardiogram phase space. Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine, 226(1), 3-20.
- Tulppo, M. P., Makikallio, T. H., Seppanen, T., Airaksinen, J. K., & Huikuri, H. V. (1998). Heart rate dynamics during accentuated sympathovagal interaction. American Journal of Physiology-Heart and Circulatory Physiology, 274(3), H810-H816.

10 Have fun analyzing the recordings!

11 References

[BackyardBrains, 2017] BackyardBrains (2009-2017). Experiment: Heart Action Potentials. https://backyardbrains.com/experiments/heartrate. Accessed December 20, 2019.

[Boron and Boulpaep, 2012] Boron, W. F. and Boulpaep, E. L. (2012). Medical physiology, 2e updated edition e-book: with student consult online access. Elsevier health sciences.

[Gray et al., 1998] Gray, R. A., Pertsov, A. M., and Jalife, J. (1998). Spatial and temporal organization during cardiac fibrillation. Nature, 392(6671):75.

*[Kantz and Schreiber, 2004] Kantz, H. and Schreiber, T. (2004). Nonlinear time series analysis, volume 7. Cambridge university press.

*[Srinivasan et al., 2003] Srinivasan, N., Wong, M., and Krishnan, S. (2003). A new phase space analysis algorithm for cardiac arrhythmia detection. In Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No. 03CH37439), volume 1, pages 82–85. IEEE.

[Umapathy et al., 2010] Umapathy, K., Nair, K., Masse, S., Krishnan, S., Rogers, J., Nash, M. P., and Nanthakumar, K. (2010). Phase mapping of cardiac fibrillation. Circulation: Arrhythmia and Electrophysiology, 3(1):105–114.

[WHO, 2019] WHO (2019). Cardiovascular Diseases. https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds). Accessed December 21, 2019.

* Denotes the most useful resources for the creation of this notebook.