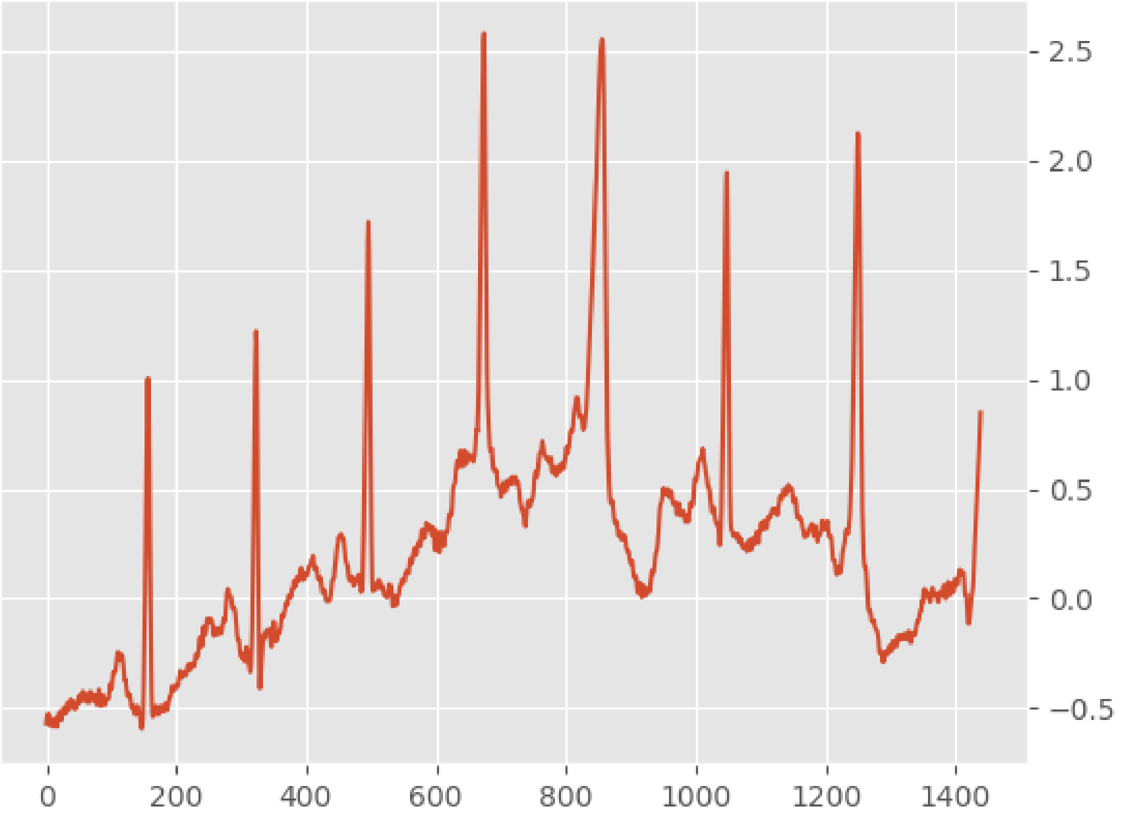
NUS-ISS
Intelligent Sensing and Sense Making

Module 2 - Foundations of sensor signal processing

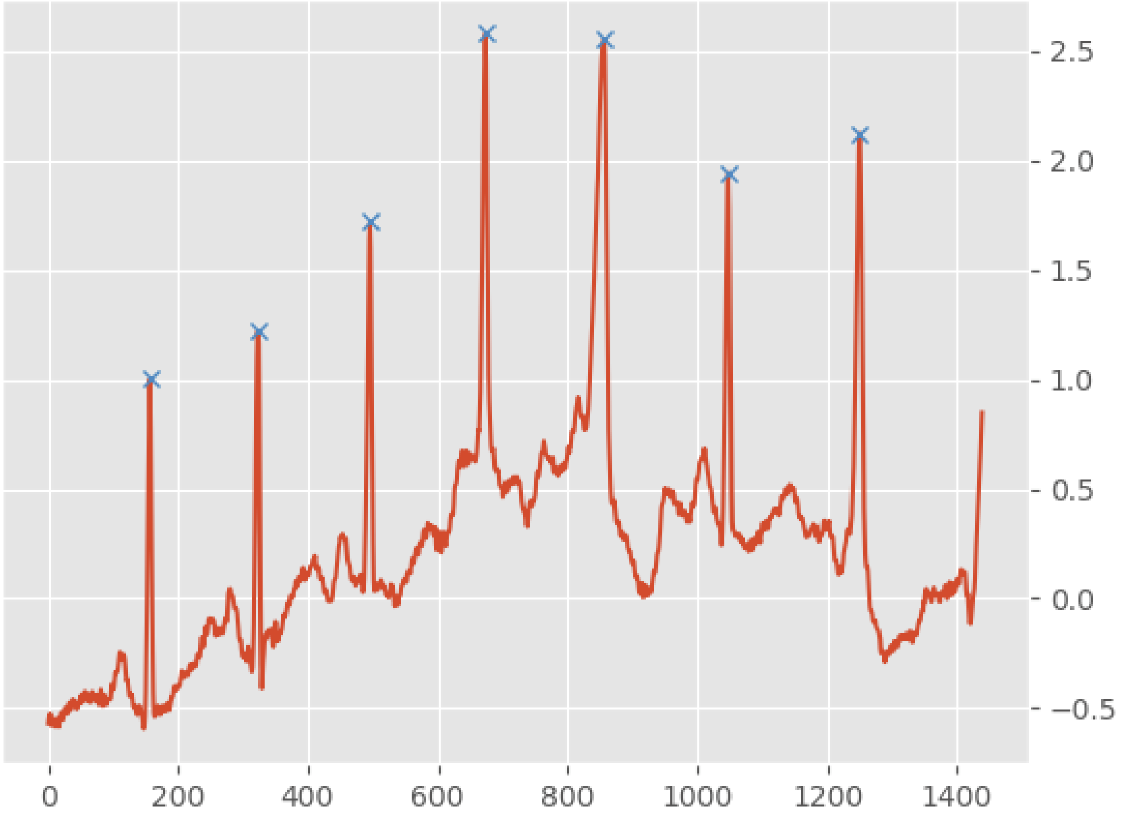
by Nicholas Ho

Problem

original



peaks detected



Peaks

How about health or financial world?

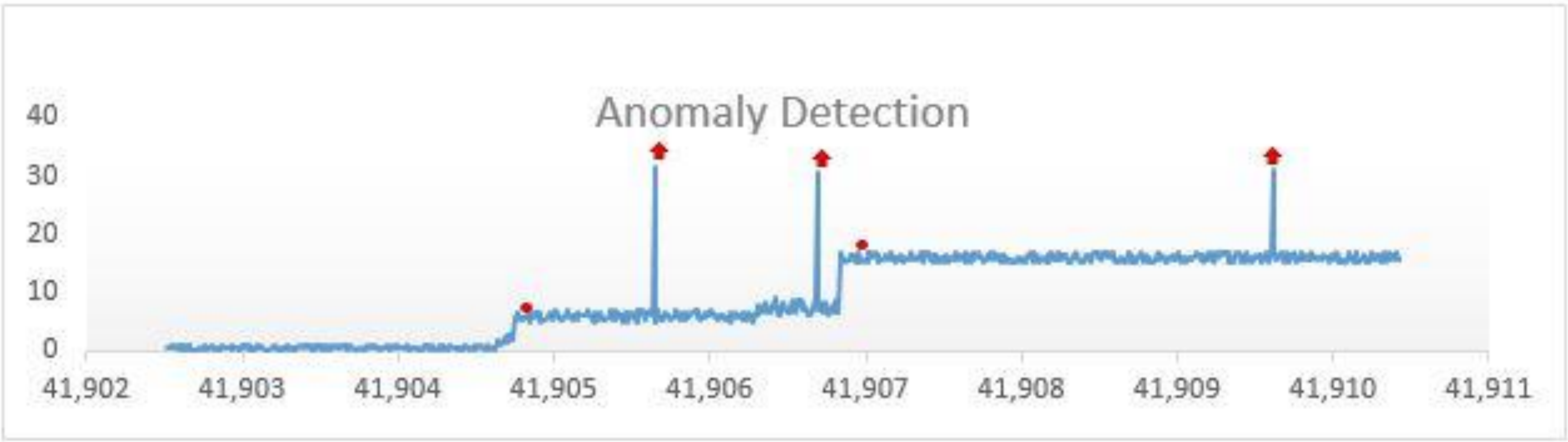
Source: https://pixabay.com/en/alps-mountain-peaks-nature-snow-2194319/

- Peaks: useful topological features of a time-series
- Indicate significant events.
- In network distribution data, indicate sudden high demands
- In server utilization workload, indicate sharp increase in workload

Anomaly detection

For log analytics

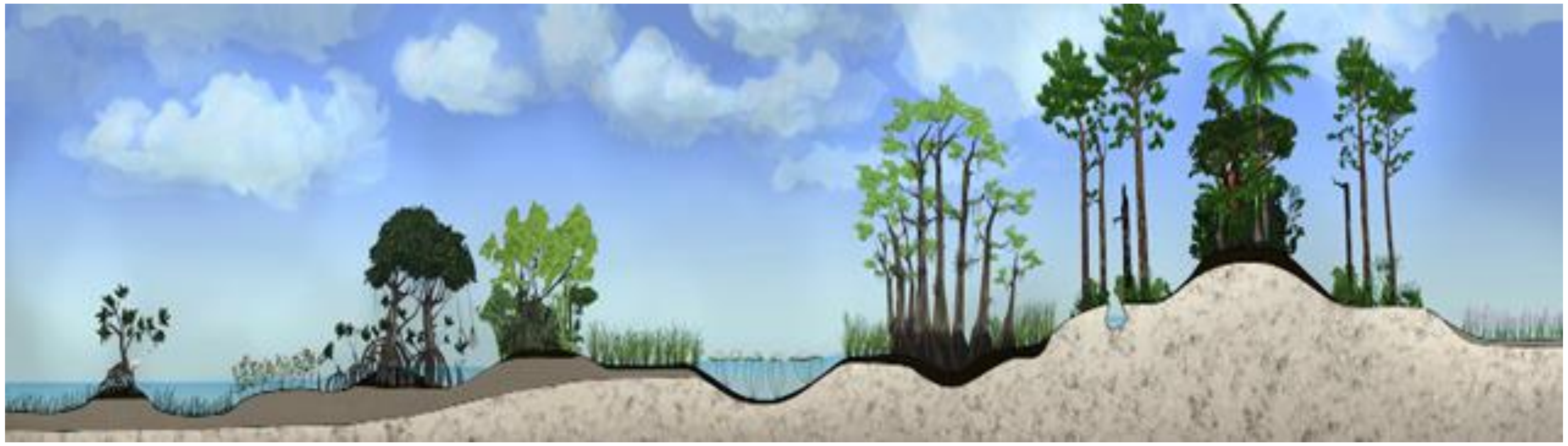
- Time series has 2 distinct level changes and 3 spikes
- Red dots show the time when level change is detected
- Red arrows show detected spikes



Source: https://blogs.technet.microsoft.com/machinelearning/2014/11/05/anomaly-detection-using-machine-learning-to-detect-abnormalities-in-time-series-data/

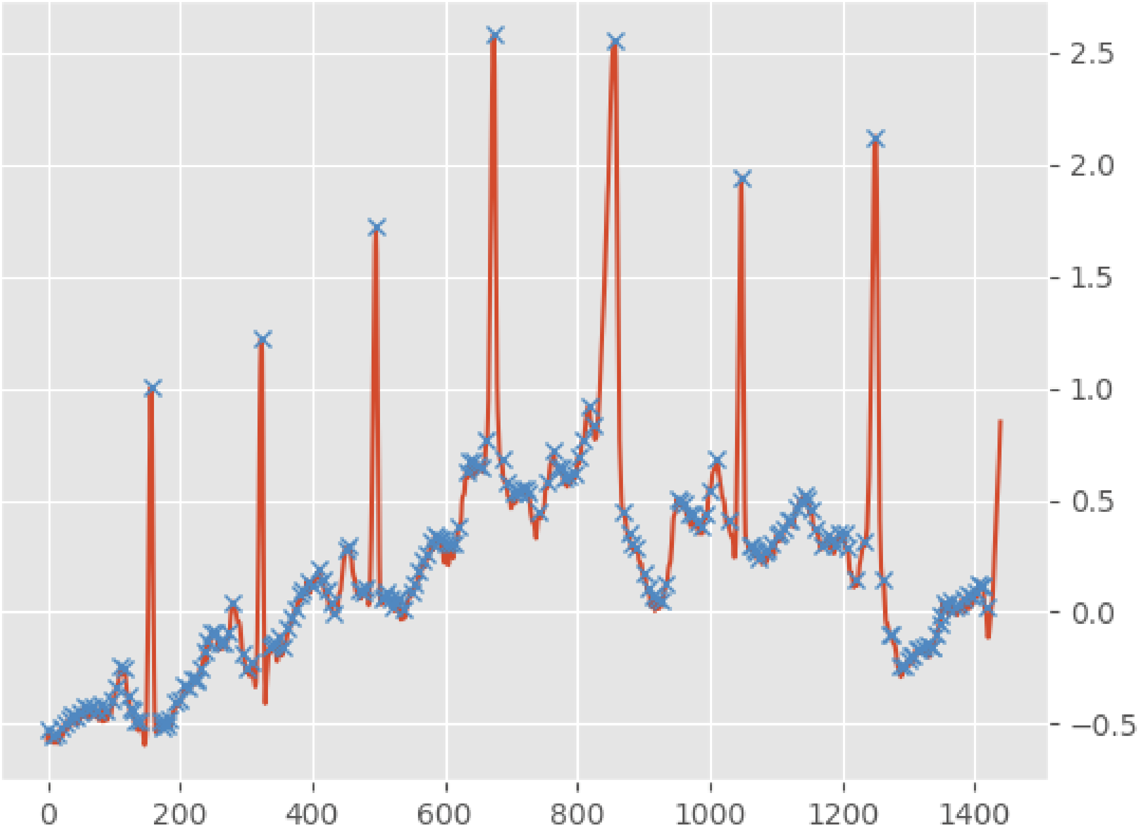
Peaks / Troughs

- Troughs: considered inverted peaks, equally important in many applications
- After peaks detected, analysis of peaks involving
  - identifying periodicity of peaks
  - forecasting the time of occurrence and value of next peak
  - identifying dependencies among of peaks of two or more time-series



Source: https://www.nps.gov/ever/learn/education/learning/mntsandvalleys.htm

Peaks / Troughs



- Peaks easily identified visually, but **not so straightforward doing through algorithm**

- Noise in data creates **tremendous amount of false positive**

- Some peaks are not result of noise, but still not relevant/desirable in analysis

Group exercise

State a use case of peak detection in any industry or research field. Avoid re-use any examples given in the lecture notes. Describe clearly about the signal, the motivation behind the use of peak detection and the purpose it serves.

Write your use case at https://docs.google.com/spreadsheets/d/1L5gww0H0xfvYX_ycELEnkciKu3x4h3rotHBfERn0_-Q/edit?usp=sharing

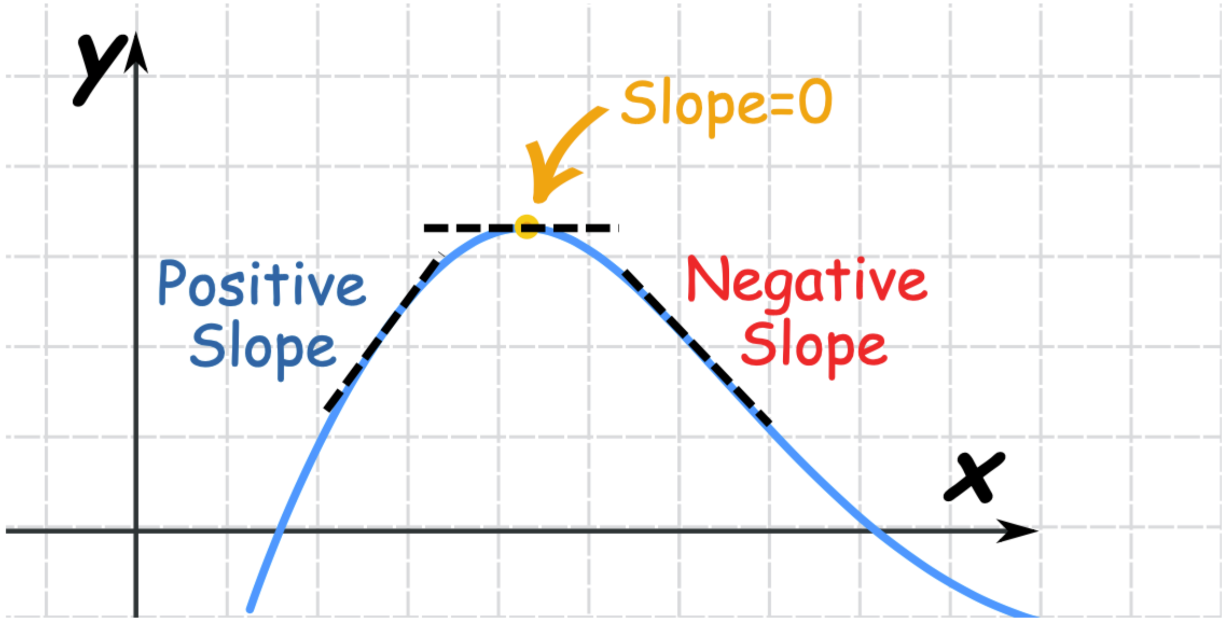
Maxima / Minima

- For a time-continuous signal, peaks / troughs can be determined by searching maxima / minima in the signal

- Assume $y = f(x) \qquad x, y \in R$

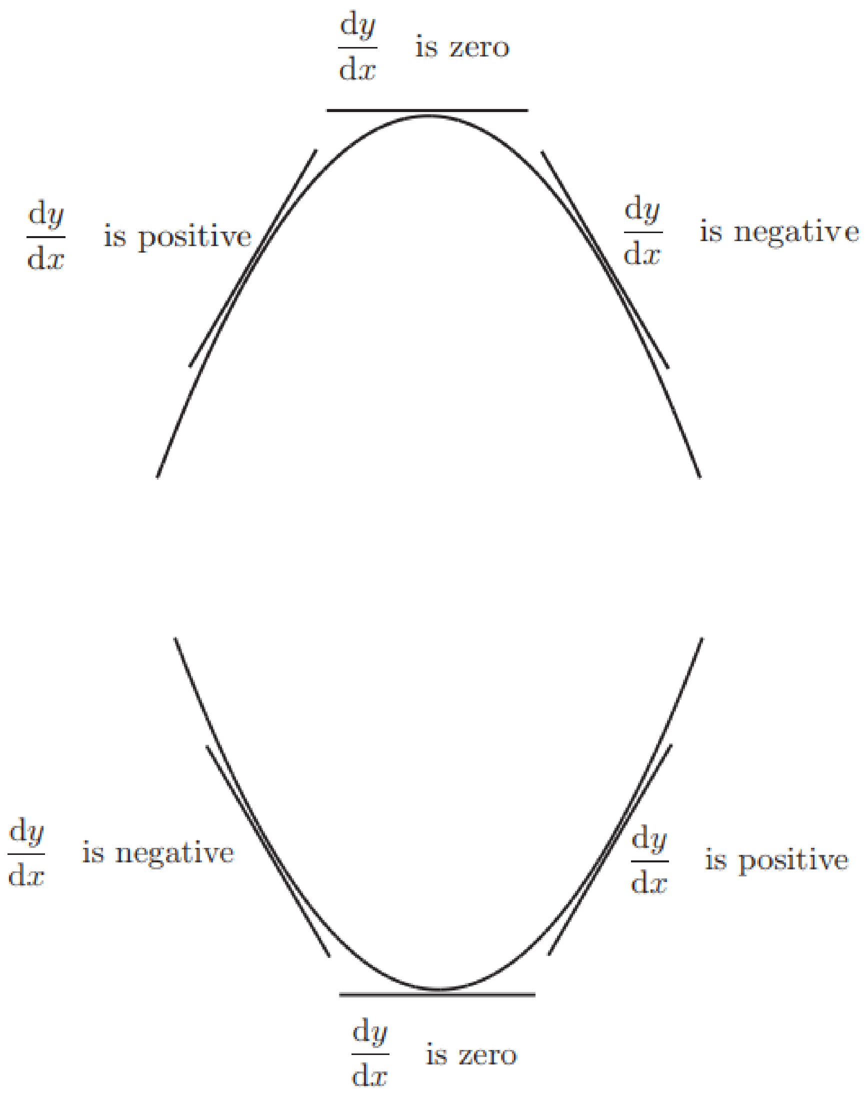
- The maxima and minima are points where

$$\frac{\mathrm{dy}}{\mathrm{d}x} = f'(x) = 0$$



Source: https://www.mathsisfun.com/calculus/maxima-minima.html

Maxima / Minima



$\frac{dy}{dx}$ is zero
$\frac{dy}{dx}$ is positive
$\frac{dy}{dx}$ is negative
$\frac{dy}{dx}$ is negative
$\frac{dy}{dx}$ is positive
$\frac{dy}{dx}$ is zero

Source: https://www.toppr.com/guides/maths/application-of-derivatives/maxima-and-minima/

- At a maxima, f'(x) changes sign from + to -
- At a minima, f'(x) changes sign from - to +
- Question: Is this entire idea applicable to discrete signal?
- Question: Should we search the signal and find the point where f'(x) equals to 0?

This method NOT applicable to discrete signals!

Because not continuous!

Finite difference

- Forward difference

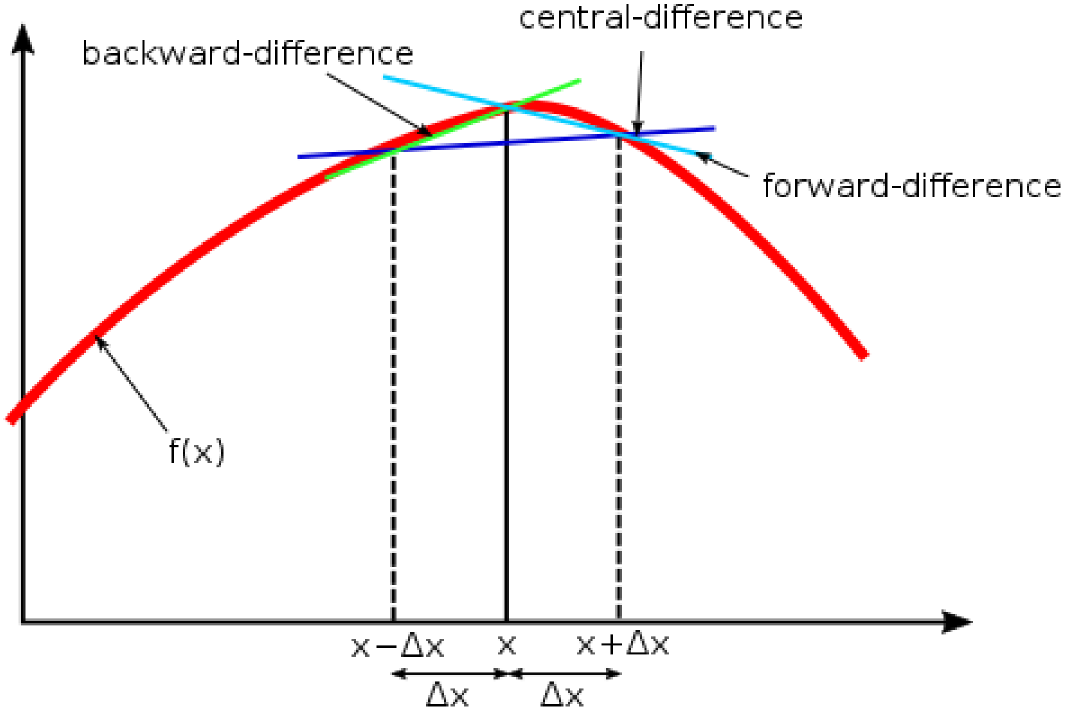
$$f'(x) = \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

- Backward difference

$$f'(x) = \frac{f(x) - f(x - \Delta x)}{\Delta x}$$

- Central difference

$$f'(x) = \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x}$$



Source: https://www.toppr.com/guides/maths/application-of-derivatives/maxima-and-minima/

Delta x is the sampling period

For simplicity purposes, it is assumed to be 1 for our WS

Finite difference in numpy

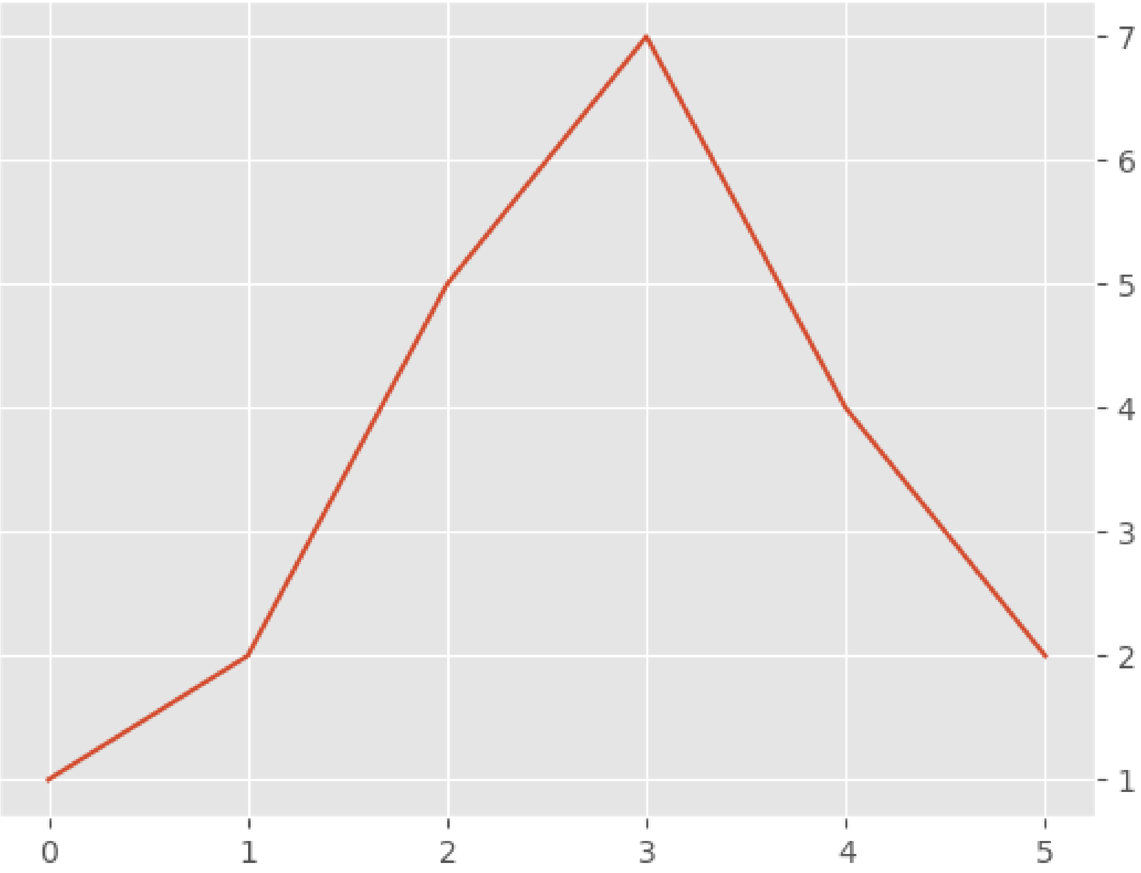
`numpy` library can help us calculate finite difference

- Import the necessary, create the array

```
> import numpy as np
> sg  = np.array([1,2,5,7,4,2])
```

- Calculate the finite difference (forward difference by default in numpy)

```
> dsg = np.diff(sg)
> dsg
: array([ 1,  3,  2, -3, -2])
```



Plotting ....

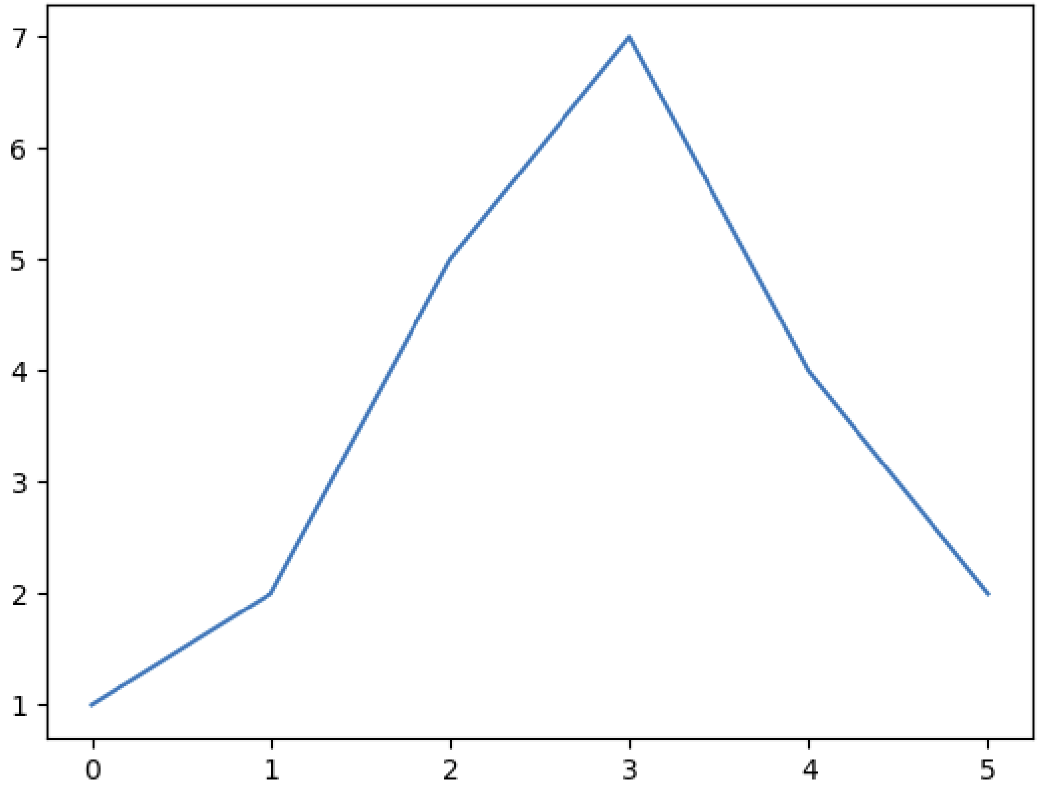
- Use matplotlib to do plotting

```
> import matplotlib.pyplot as plt
```

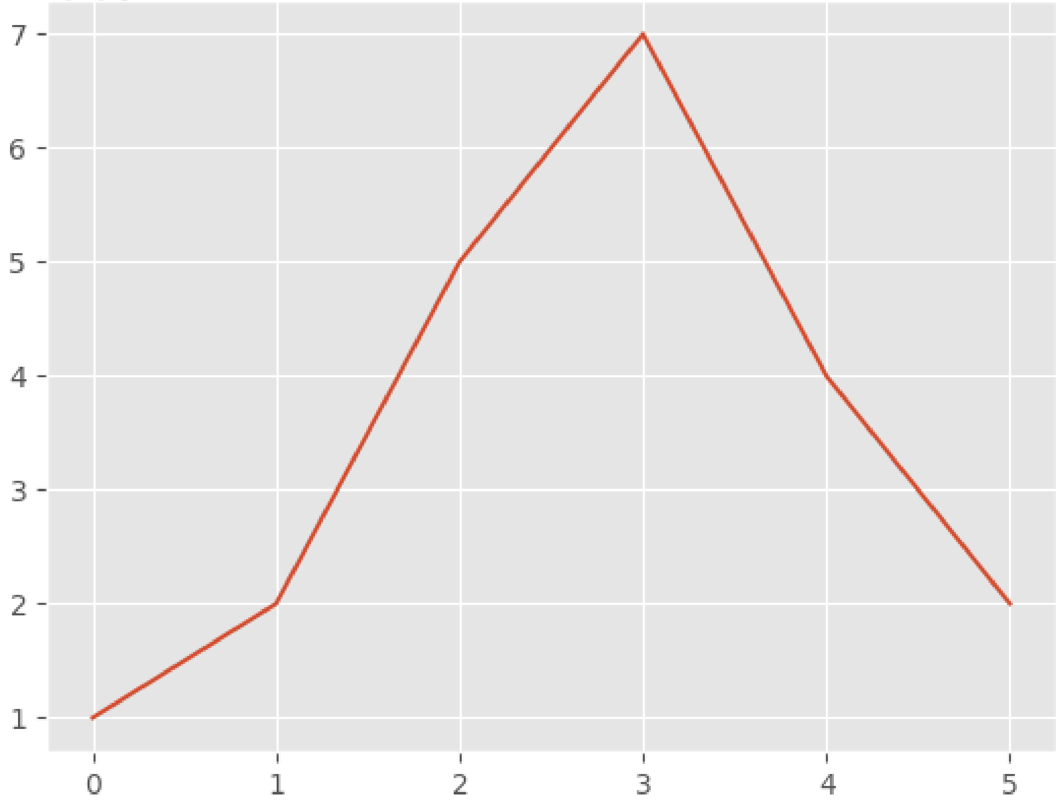
- Change plotting style to 'ggplot', if want to change back to default, set 'classic'

```
> plt.style.use('ggplot')
```

'classic'



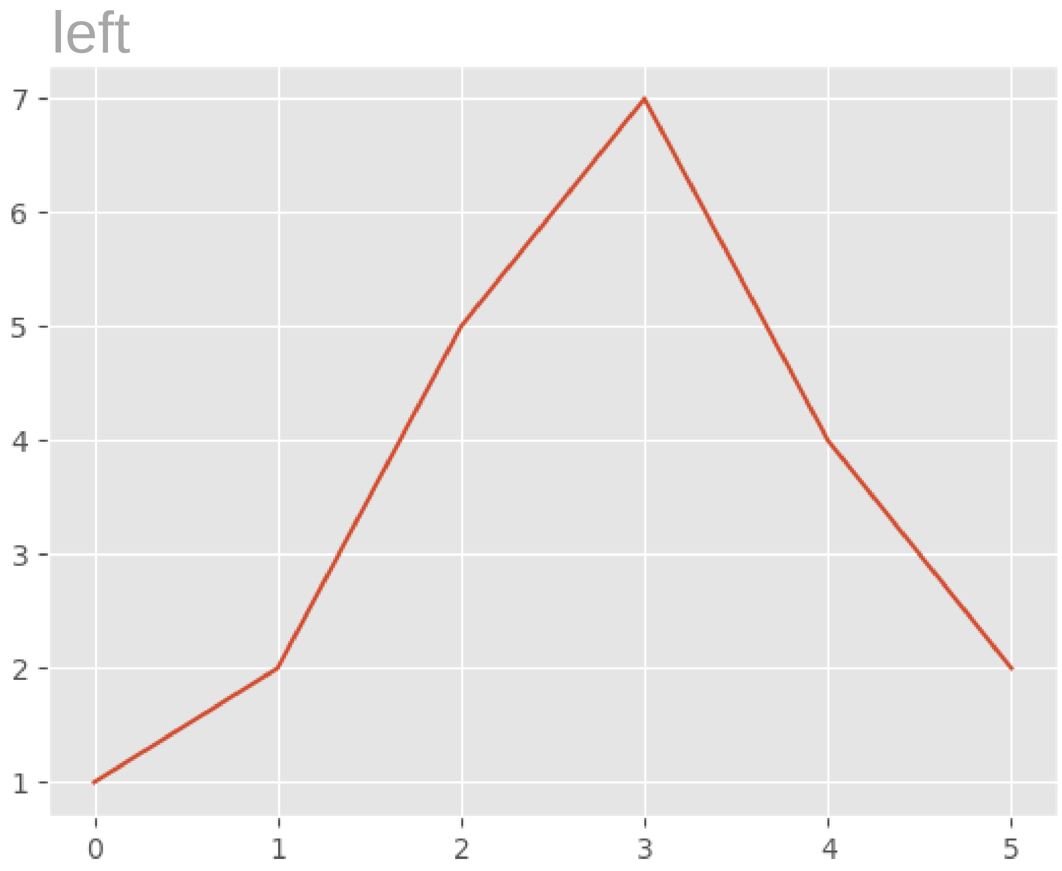
'ggplot'

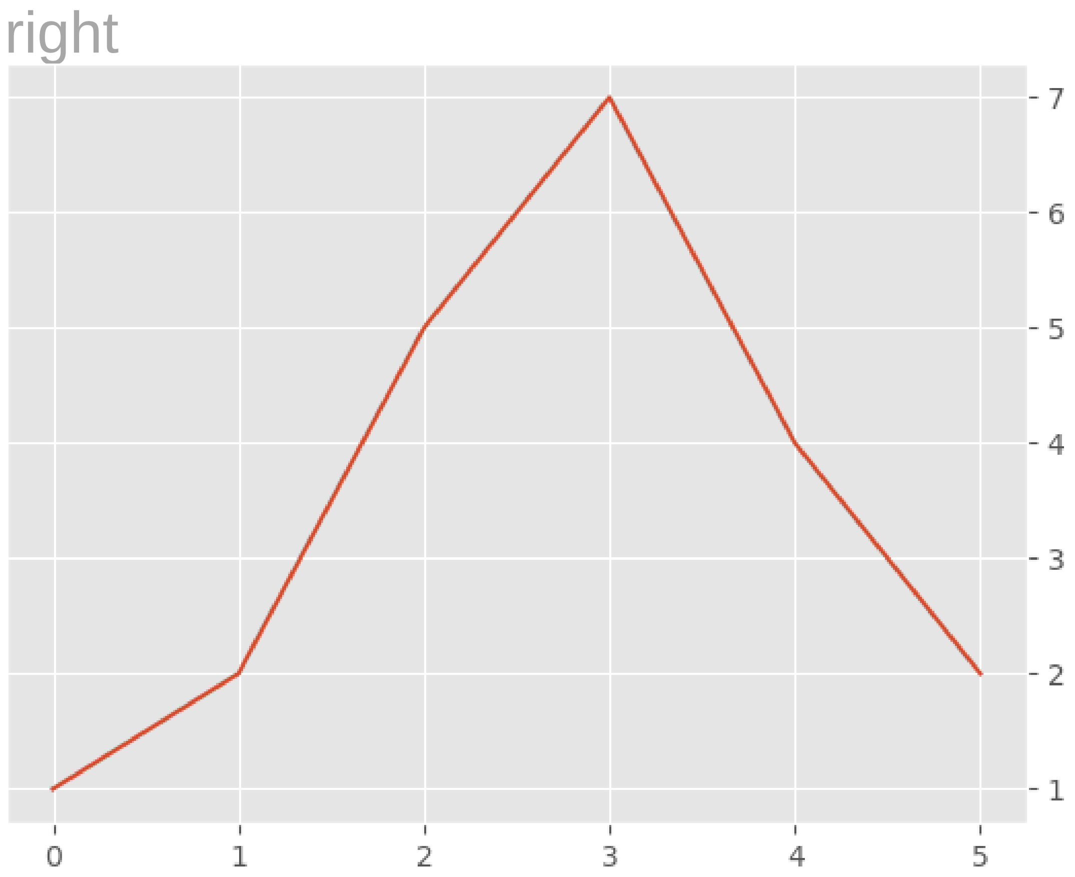


Plotting ....

- Show the y-axis label values at the right side

```
> plt.rcParams['ytick.right']      = True
> plt.rcParams['ytick.labelright']= True
> plt.rcParams['ytick.left']       = False
> plt.rcParams['ytick.labelleft'] = False
```





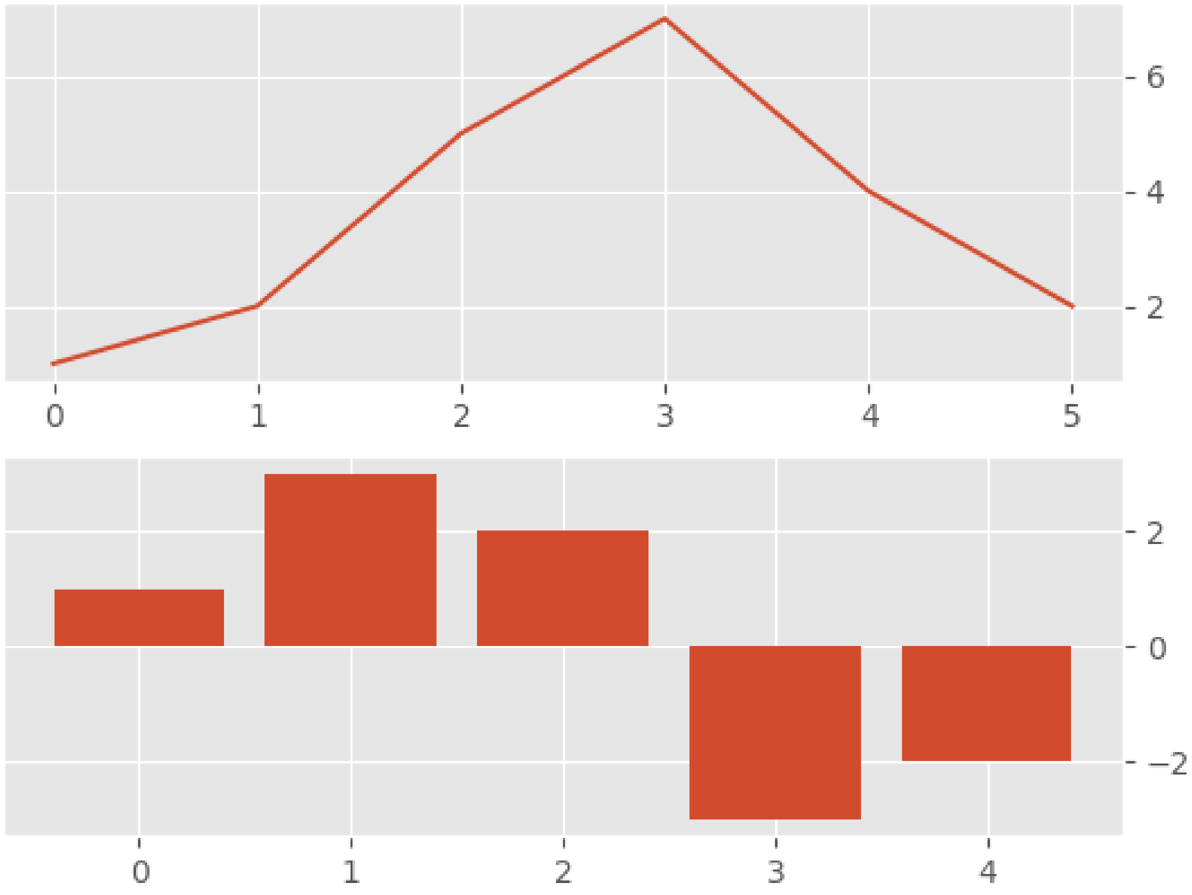
Finite difference in numpy

Downside of finite difference methods?

All the peaks will be located; many of which are not useful; need an option to filter out

- Plot graph and take a look

```
> plt.figure()
> plt.subplot(211)
> plt.plot(sg)
> plt.subplot(212)
> plt.bar(np.arange(dsg.shape[0]),
           dsg,)
```



Looking at this chart, what is the right strategy to search for peaks in discrete signal?

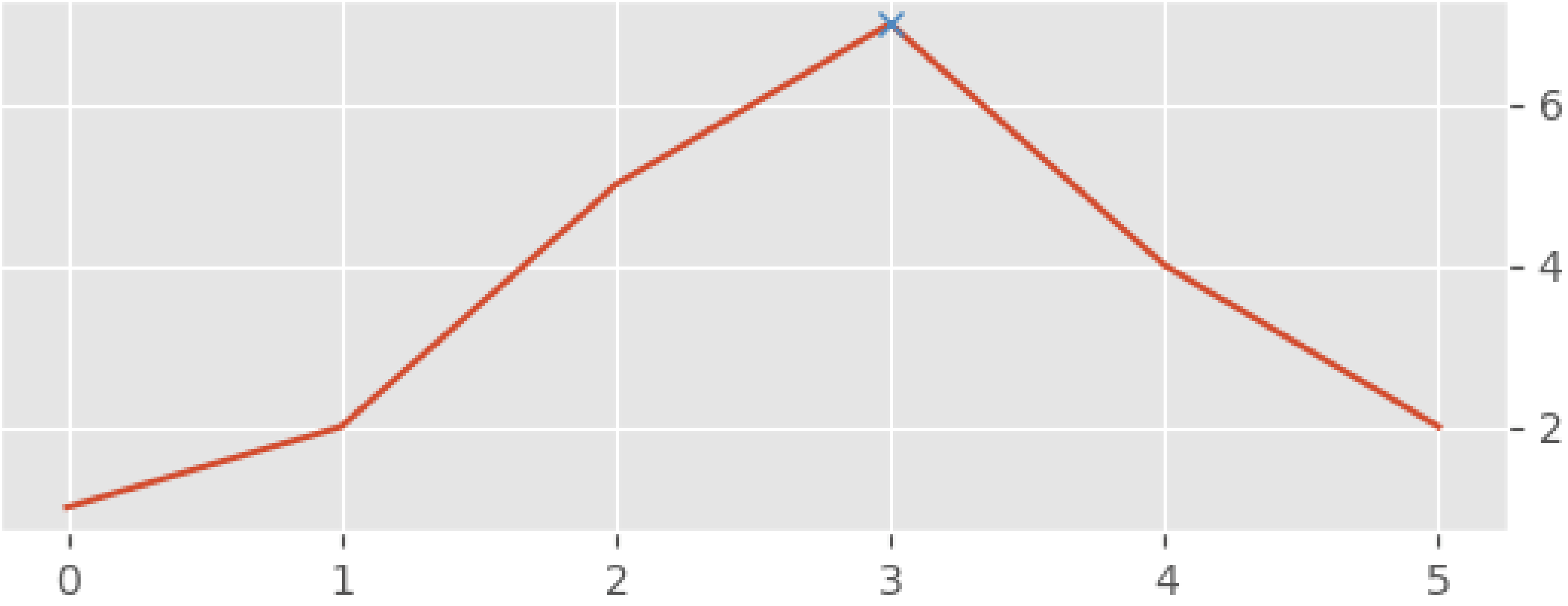
Finding peak

- using findPeaks from scipy

```
> from scipy.signal import find_peaks as findPeaks
> (Pks,_) = findPeaks(sg)
> Pks
: array([3])
```

- Plot the finding

```
> plt.figure()
> plt.plot(sg)
> plt.plot(Pks,sg[Pks],'x')
```



Back to the problem

A useful tool to enhance the finite difference method:

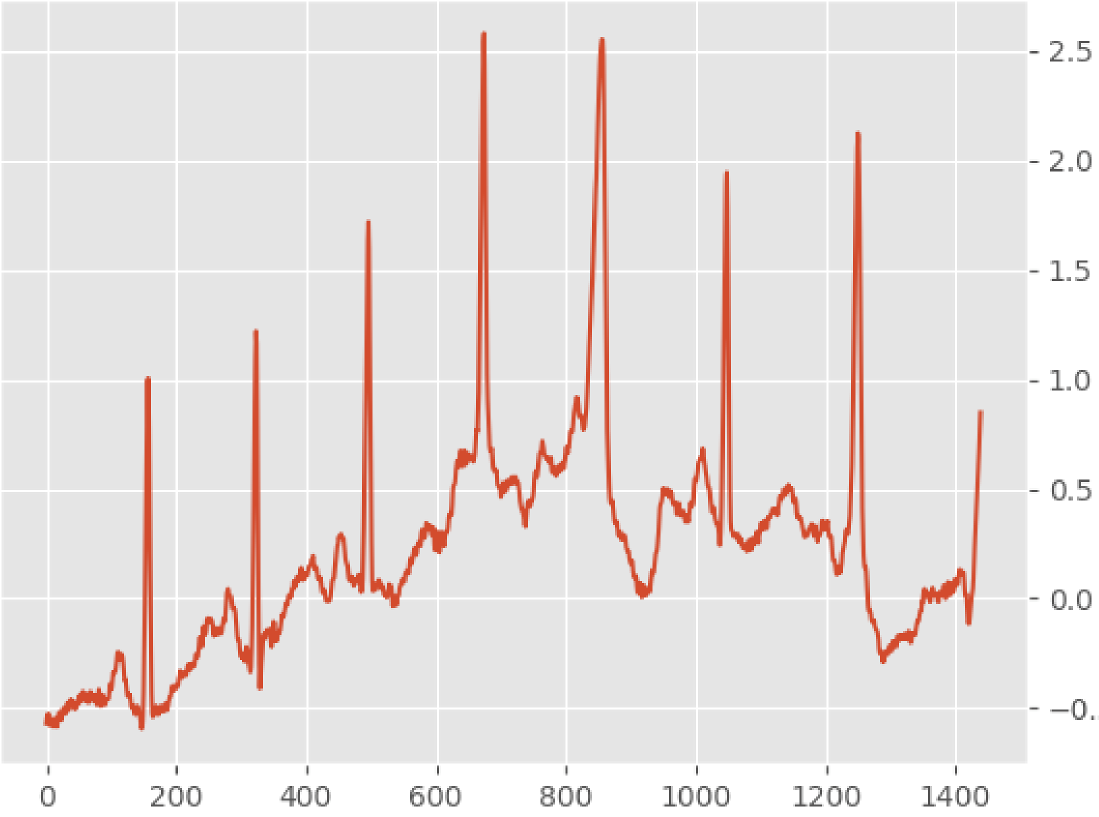
`findPeaks` API

Steps to do:

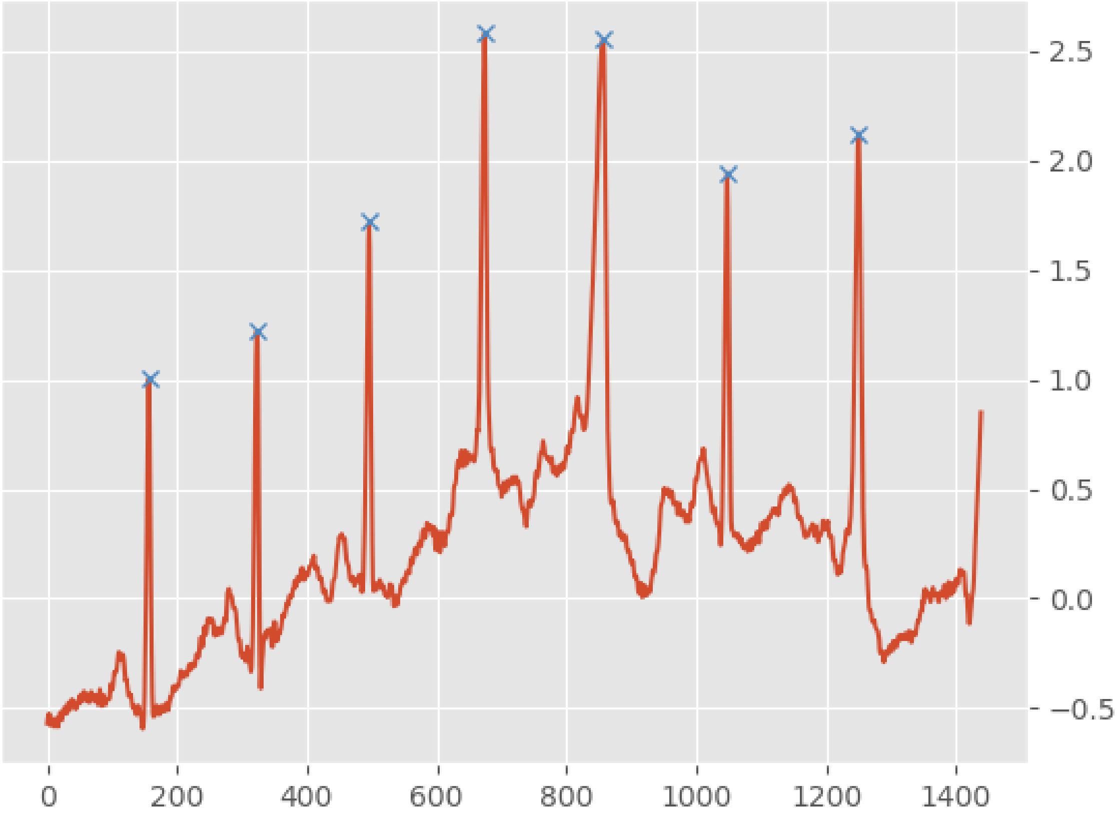
1. Load data
2. run `findPeaks`
3. Fine tune / optimizing
4. Repeat step 2 and 3 until we get satisfactory detection
5. Display output

Filtering process here

original



peaks detected



Solving the problem

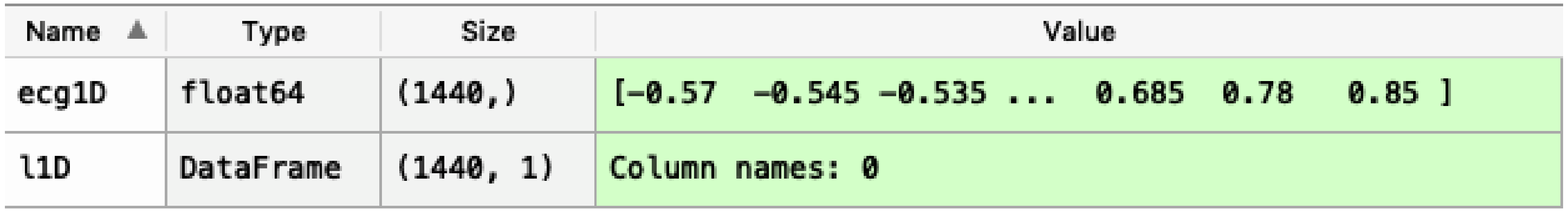
1. Load data

- Many times 1D signal comes in the form of comma-separated values (csv)
  - fields/columns separated by comma
  - record/rows terminated by newline

```
-0.57
-0.545
-0.535
-0.525
-0.545
-0.58
-0.575
-0.565
-0.555
-0.55
-0.57
-0.585
-0.57
-0.56
-0.555
-0.545
-0.565
-0.585
-0.55
-0.535
-0.52
-0.535
-0.55
-0.555
.
.
.
```

- Use pandas to read in csv

```
> import pandas as pd
> l1D  = pd.read_csv('ecg1D.csv',
                     header=None)
> ecg1D= l1D[0].values
```



Name ▲	Type	Size	Value
ecg1D	float64	(1440,)	[-0.57 -0.545 -0.535 ... 0.685 0.78 0.85]
l1D	DataFrame	(1440, 1)	Column names: 0

About pandas ...

reading csv...

- Read in a csv with header

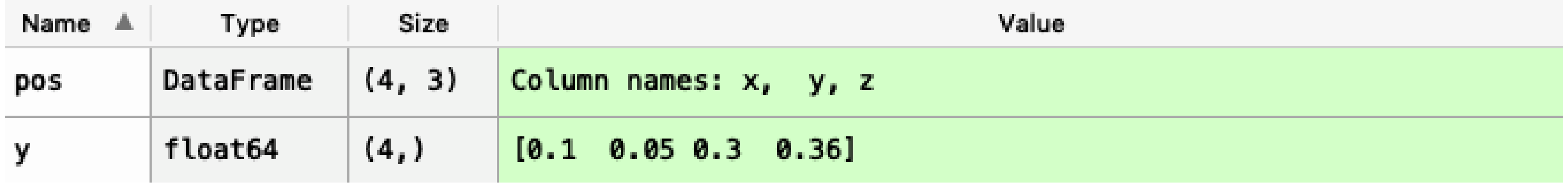
```
> pos   = pd.read_csv('pos.csv')
> list(pos)
: ['x', ' y', 'z']
```

- Sometimes it is good to list out the header, because some headers may have an empty space before letters

- To convert a column into a numpy array

```
> y    = pos[' y'].values
```

```
x, y,z
1.3,0.1,2.2
1.1,0.05,2.6
-0.7,0.3,2.5
-0.5,0.36,2.5
```



Name ▲	Type	Size	Value
pos	DataFrame	(4, 3)	Column names: x, y, z
y	float64	(4,)	[0.1 0.05 0.3 0.36]

Solving the problem

2. run `findPeaks`

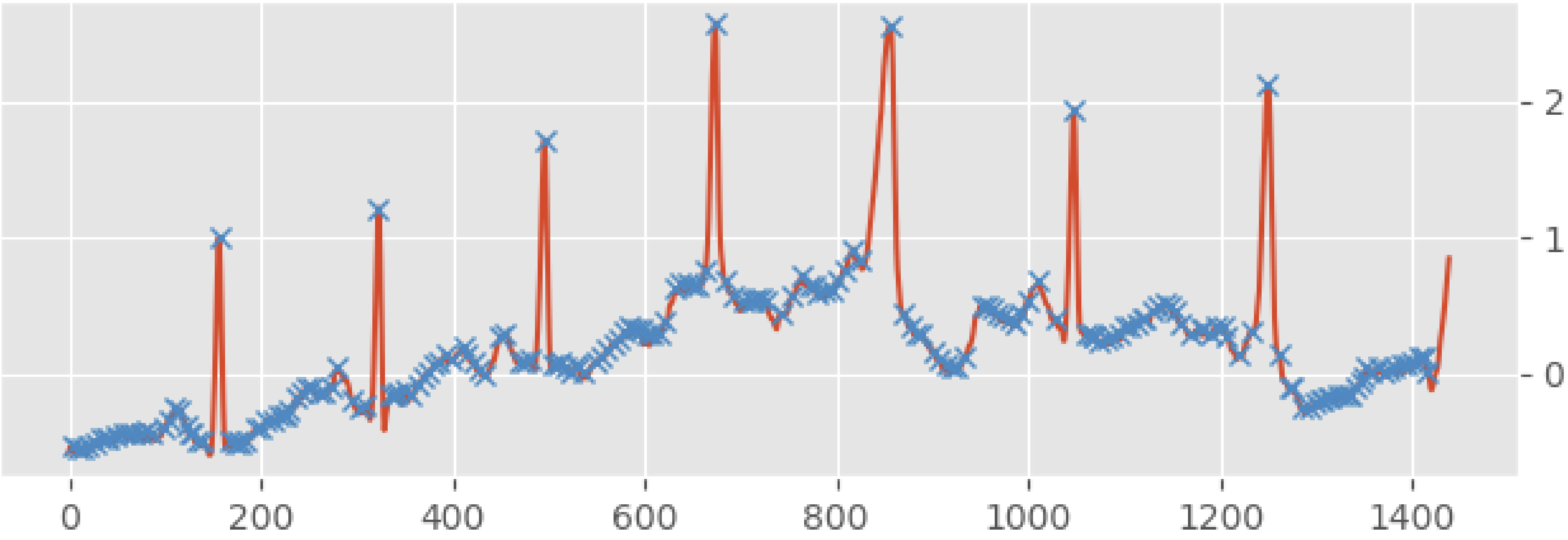
- run findPeaks without any adjustment or arguments

```
> from scipy.signal import find_peaks as findPeaks
> (allPks,_) = findPeaks(ecg1D)
```

Need to filter out unwanted peaks!

- Plot the output

```
> plt.figure()
> plt.plot(ecg1D)
> plt.plot(allPks,ecg1D[allPks],'x')
```



Solving the problem

2, 3. fine tune and re-run

`findPeaks`

Height may not be a good strategy due to variations in signals (e.g. amplitude, baseline)!

Removing the baseline may help increase the effectiveness (TBC)

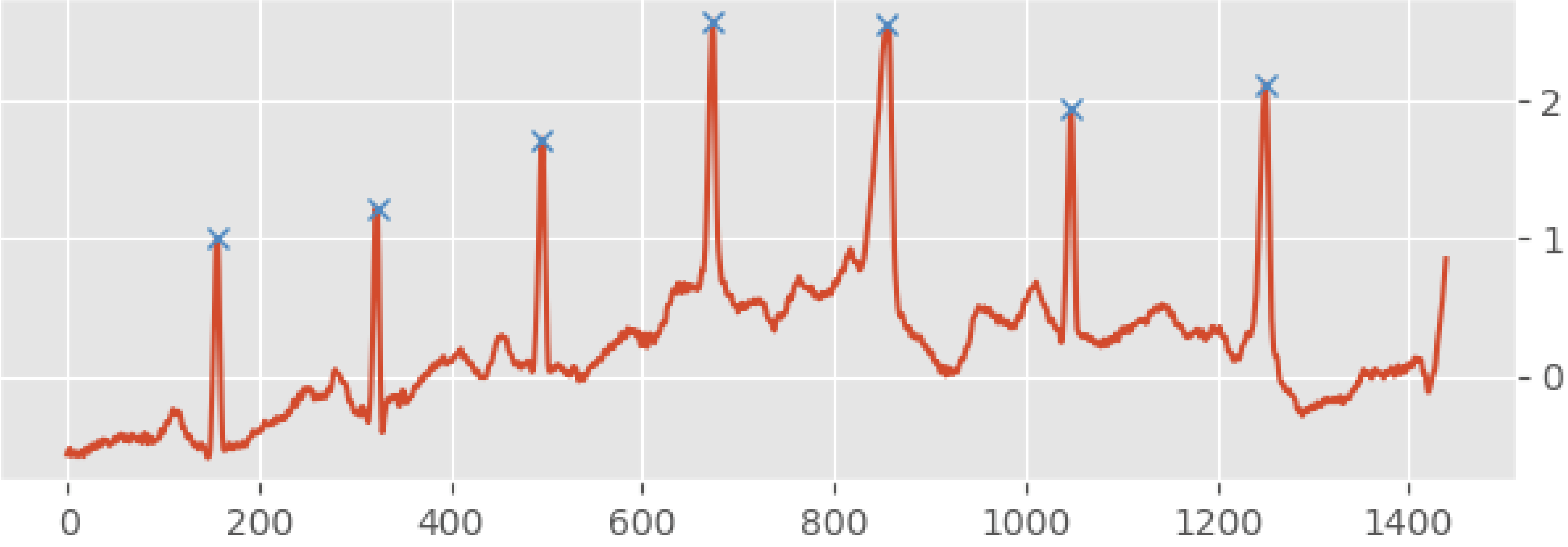
- Add additional arguments to improve outcome

```
> (somePks,_) = findPeaks(ecg1D,height=1)
```

- Get only peaks with height of at least 1

```
> plt.figure()
> plt.plot(ecg1D)
> plt.plot(somePks,ecg1D[somePks],'x')
```

It works, but is this a good strategy?



Solving the problem

2, 3. fine tune and re-run

findPeaks

Like “height”, distance may not be a good strategy too; depends on use cases

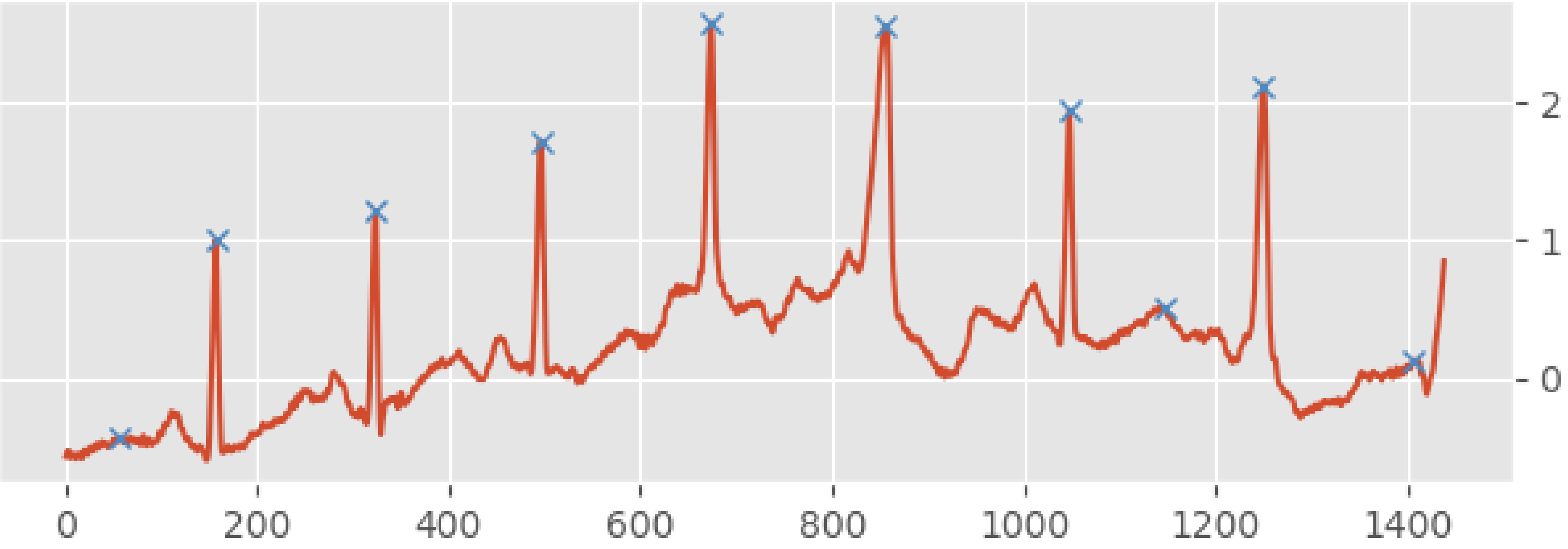
- Try another strategy, use distance

```
> (distPks,_) = findPeaks(ecg1D,distance=100)
```

- Get only peaks with at least 100 points apart

```
> plt.figure()
> plt.plot(ecg1D)
> plt.plot(somePks,ecg1D[somePks],'x')
```

Some points at T wave are picked up, some are not, why?



Solving the problem

2, 3. fine tune and re-run

`findPeaks`

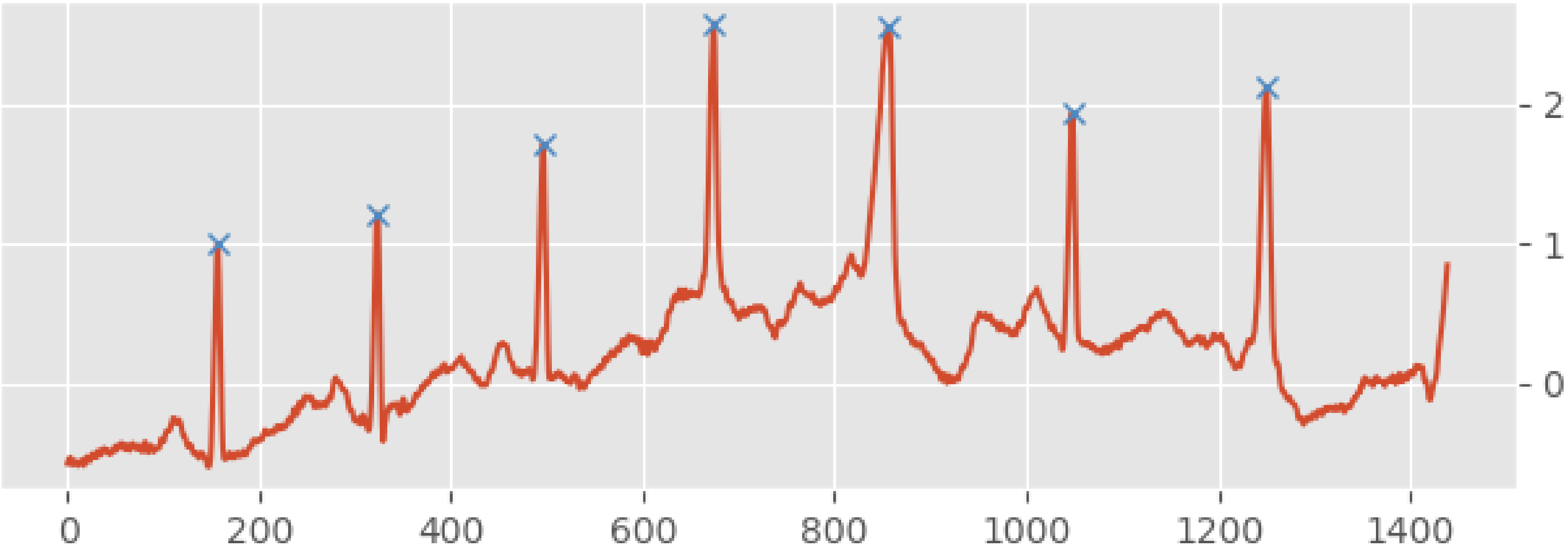
- Use prominence

```
> (prmPks,_) = findPeaks(ecg1D,prominence=0.5)
```

- Get only peaks with prominence of at least 0.5

```
> plt.figure()
> plt.plot(ecg1D)
> plt.plot(prmPks,ecg1D[prmPks],'x')
```

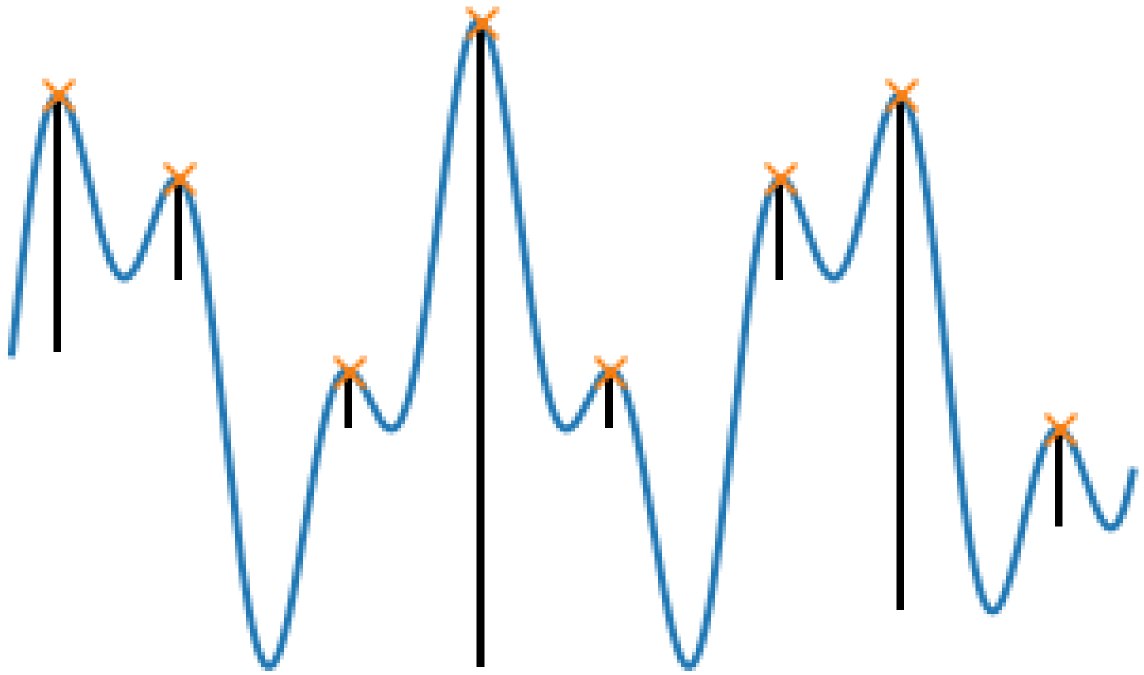
It works, but what is prominence?



Prominence

2, 3. fine tune and re-run

`findPeaks`



- The prominence of a peak measures **how much a peak stands out from the surrounding baseline of the signal**. The strategy:

1. Extend a horizontal line from the current peak to the left and right until the line either reaches the window border or intersects the signal again at the slope of a higher peak. An intersection with a peak of the same height is ignored.
2. On each side find the minimal signal value within the interval defined above. These points are the peak’s bases.
3. The higher one of the two bases marks the peak’s lowest contour line. The prominence can then be calculated as the vertical difference between the peaks height itself and its lowest contour line.

Prominence

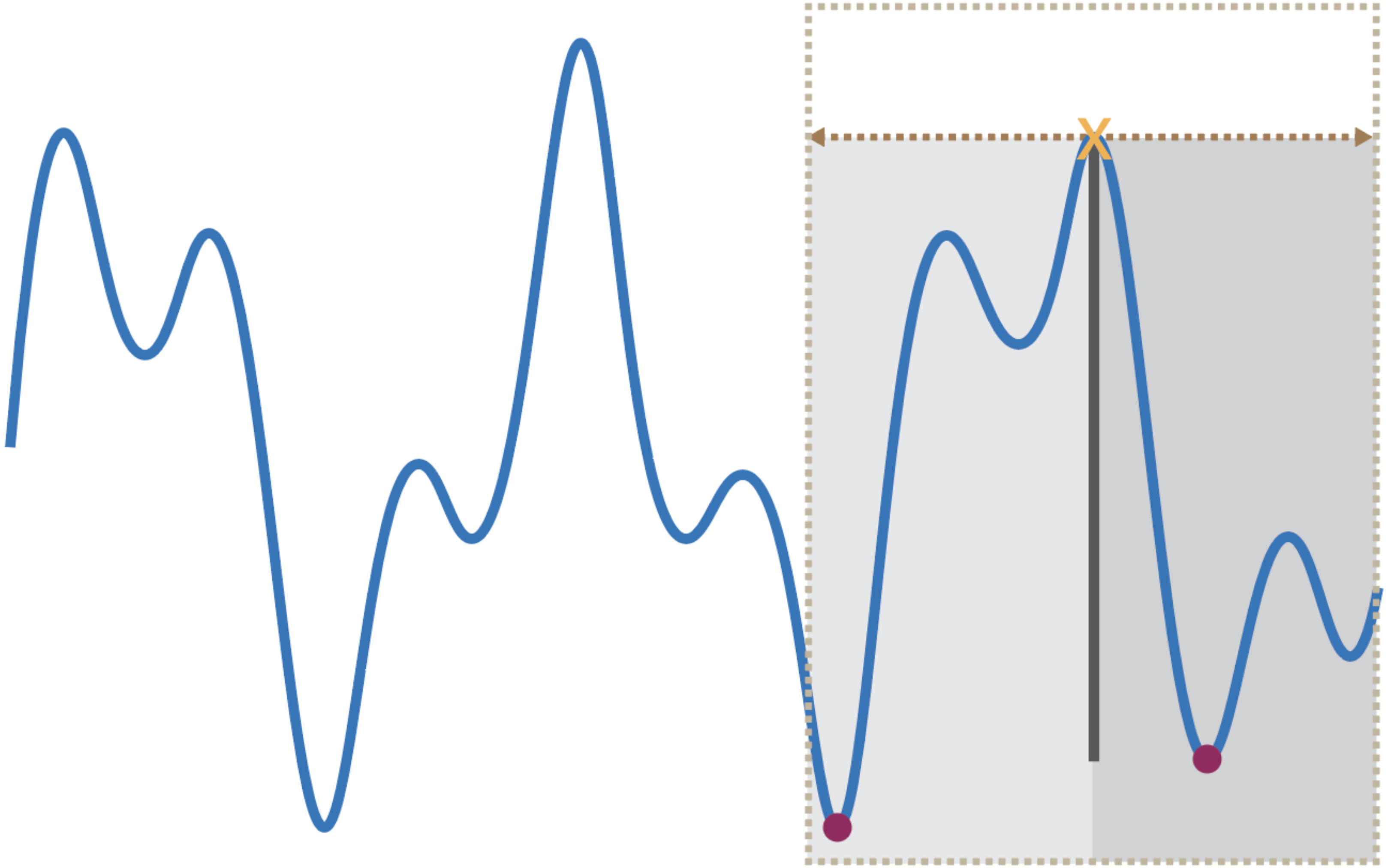
2, 3. fine tune and re-run

`findPeaks`

1. Extend a horizontal line from the current peak to the left and right until the line either reaches the window border or intersects the signal again at the slope of a higher peak. An intersection with a peak of the same height is ignored.

2. On each side find the minimal signal value within the interval defined above. These points are the peak's bases.

3. The higher one of the two bases marks the peak's lowest contour line. The prominence can then be calculated as the vertical difference between the peaks height itself and its lowest contour line.

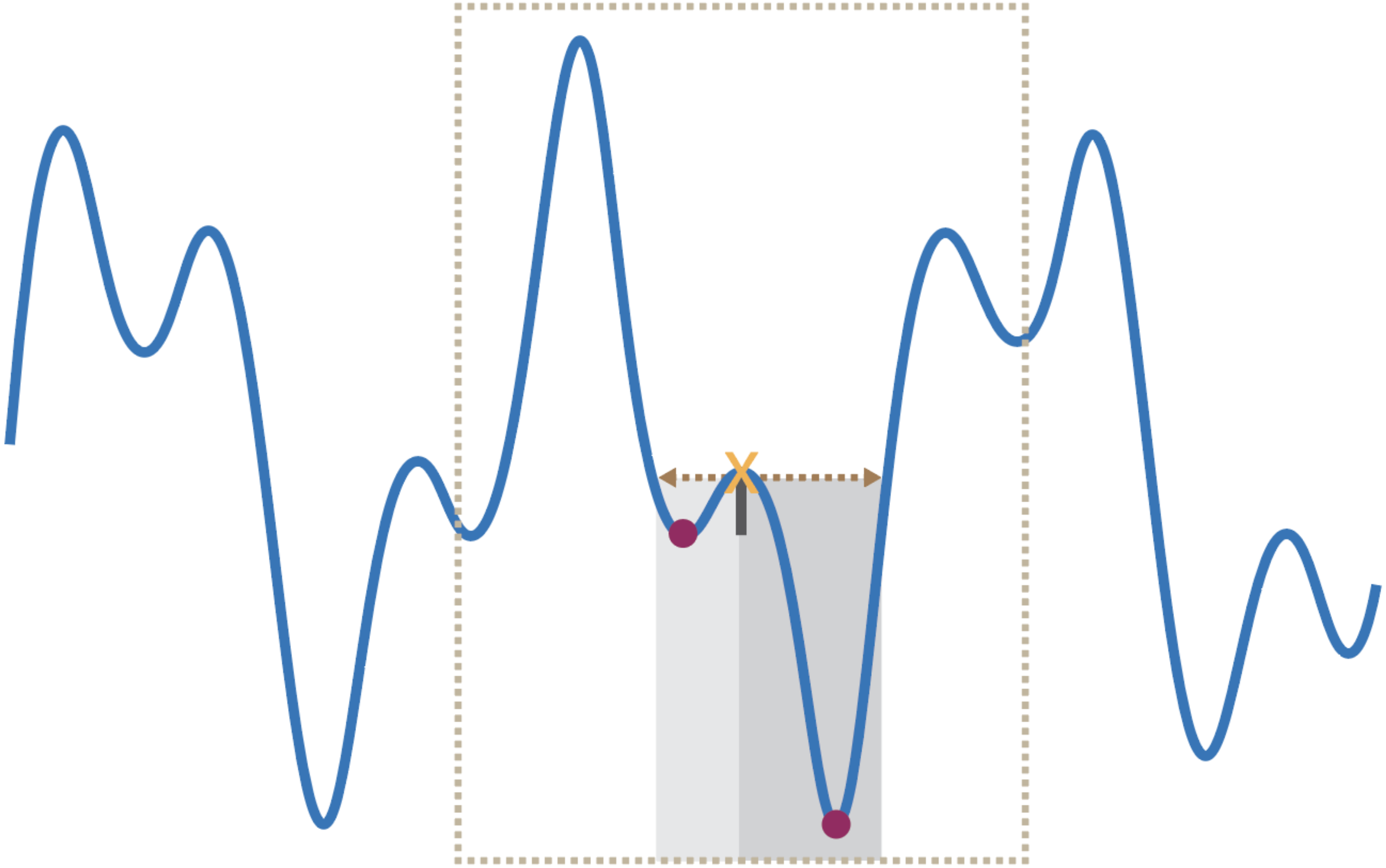


Prominence

2, 3. fine tune and re-run

`findPeaks`

1. Extend a horizontal line from the current peak to the left and right until the line either reaches the window border or intersects the signal again at the slope of a higher peak. An intersection with a peak of the same height is ignored.
2. On each side find the minimal signal value within the interval defined above. These points are the peak's bases.
3. The higher one of the two bases marks the peak's lowest contour line. The prominence can then be calculated as the vertical difference between the peaks height itself and its lowest contour line.



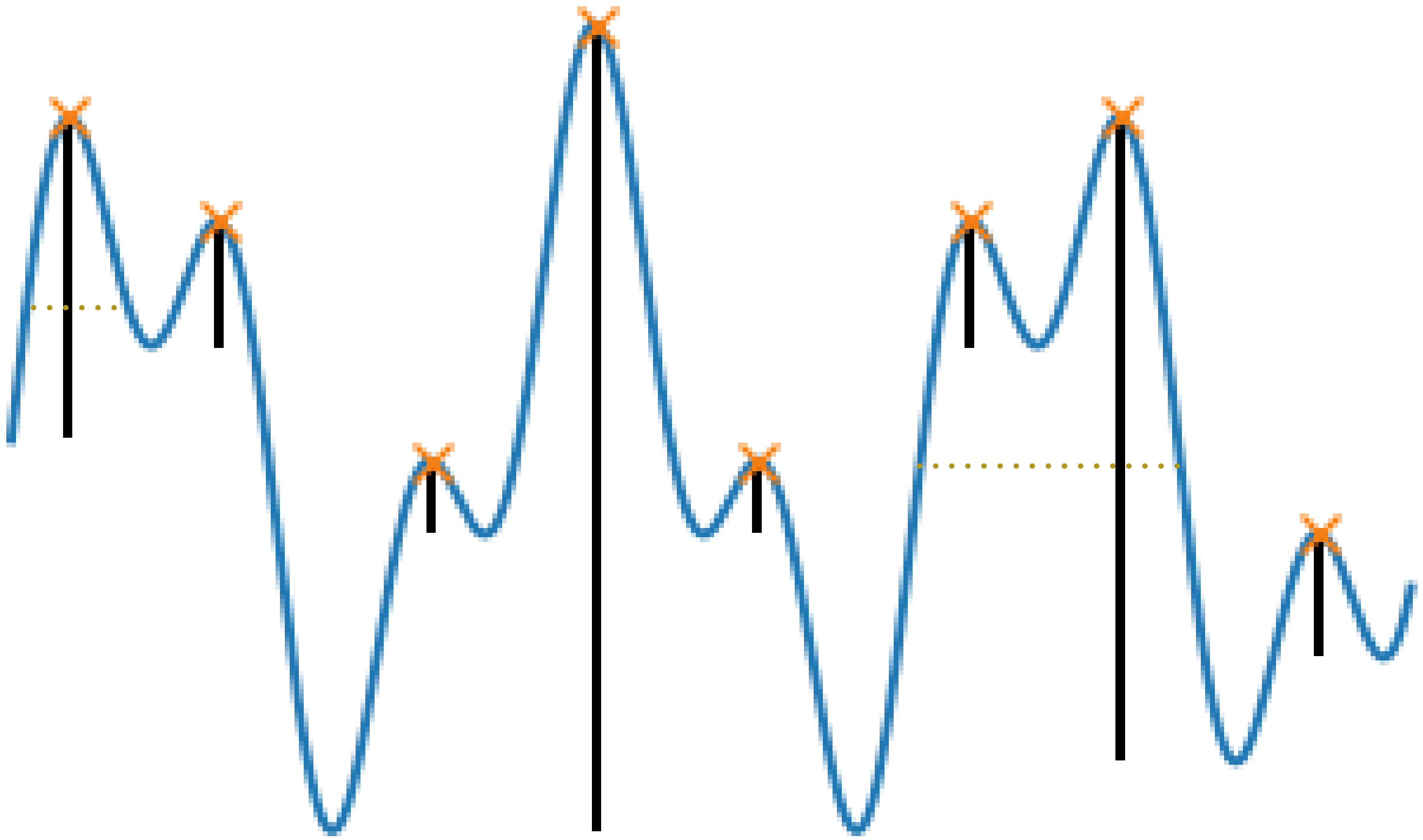
Width

2, 3. fine tune and re-run

`findPeaks`

- By default, the width of a peak is defined from the position half of of the prominence
- We can fine-tune the peak searching using **multiple parameters**

```
findPeaks(x,prominence=0.1,distance=10)
findPeaks(x,distance=5,width=2)
....
```



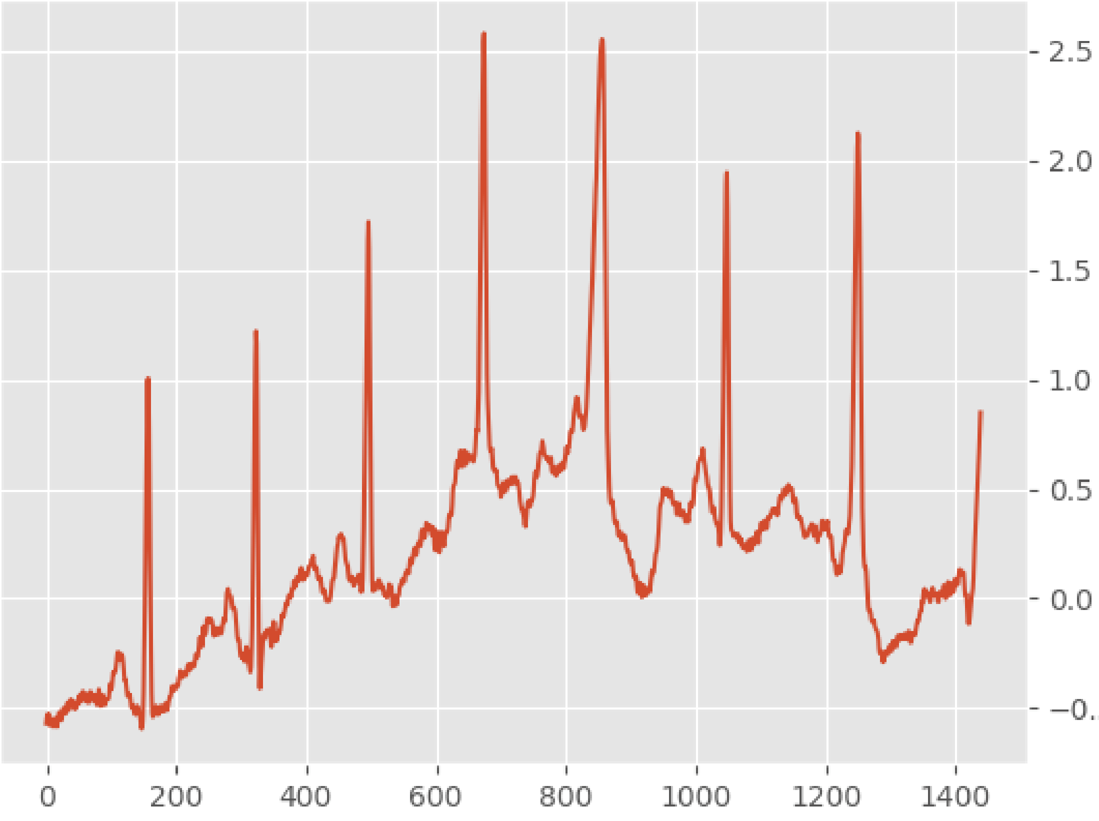
Baseline correction

Problem

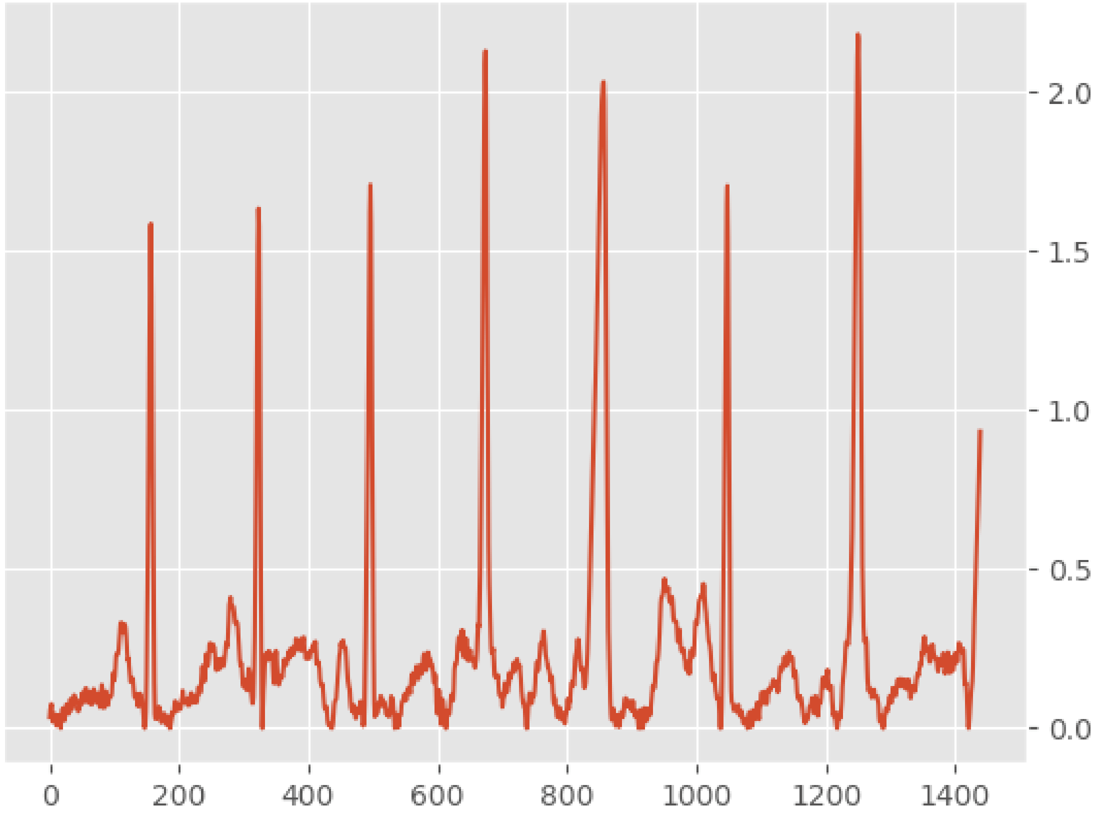
Sometimes baselines of signals can be unstable; can lead to wrong deduction by algorithms

Correcting baselines can solve this problem and enable easier peak and trough detection, hence promoting easier pattern comparison

original

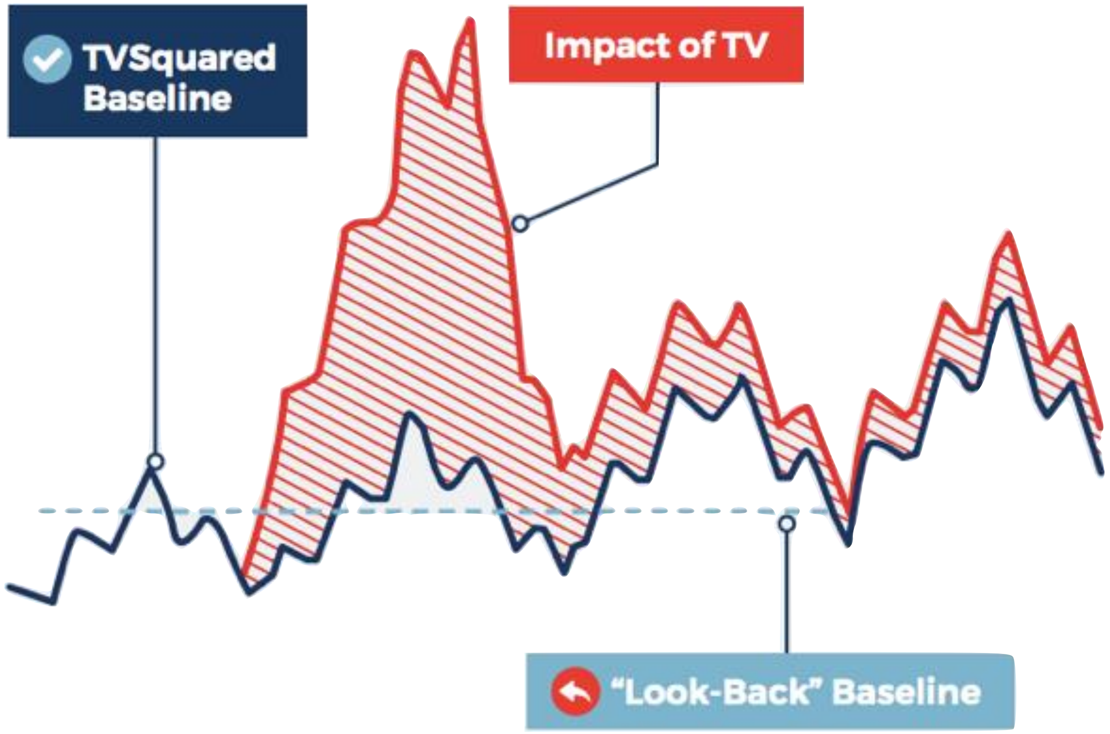


baseline corrected



Baseline correction

The cheaper your sensor, the more you will encounter this problem!



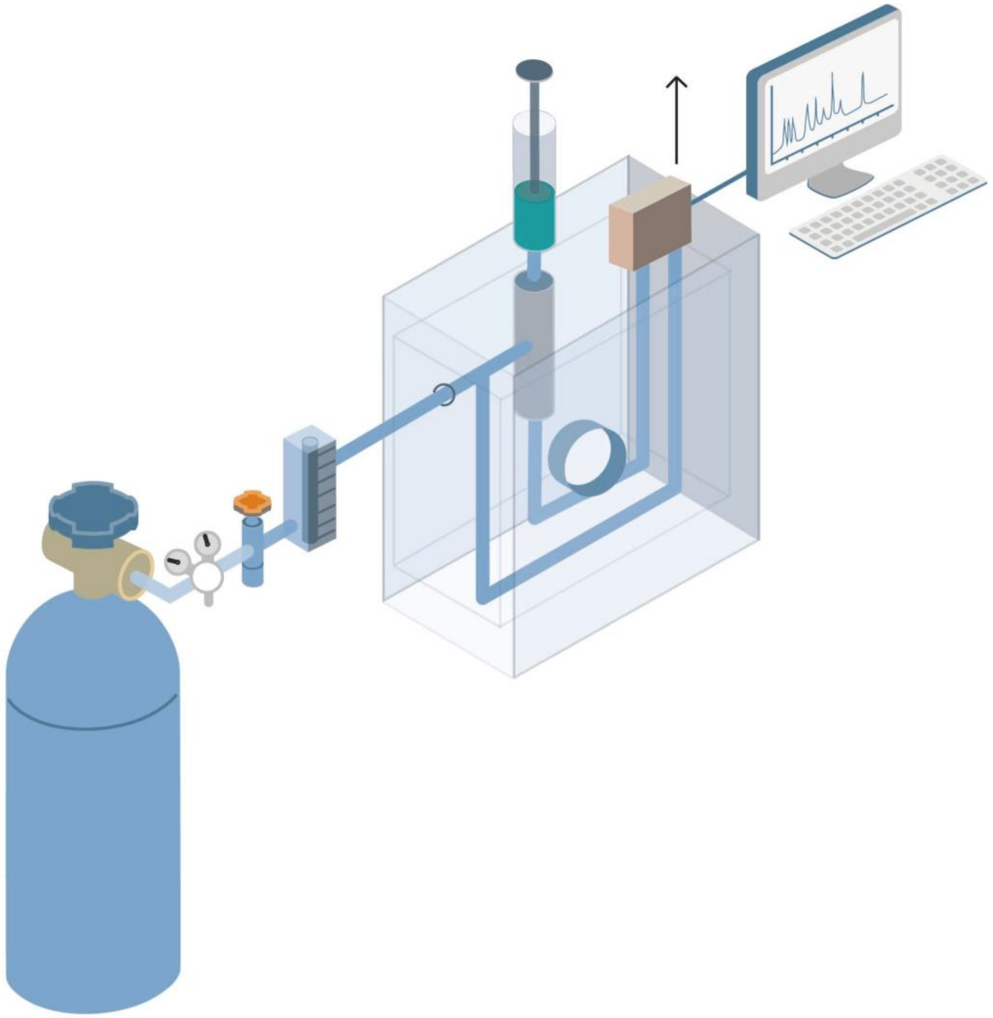
Source: https://tvsquared.com/baseline-works-better/

- Unstable baselines occur in many types of instrumental measurements
- Drift in sensor values are common phenomenon, due to changes in environment factors such as temperature and humidity
- They can cause problem, for example, disturb peak detection, pattern comparison
- Correction is routinely needed

GC chromatogram

Example

- GC chromatogram: used to analyze content of a chemical product
- Examples: analyzing the contents of a lavender oil; measuring toxic substances in soil, air or water
- Also used extensively in forensic science



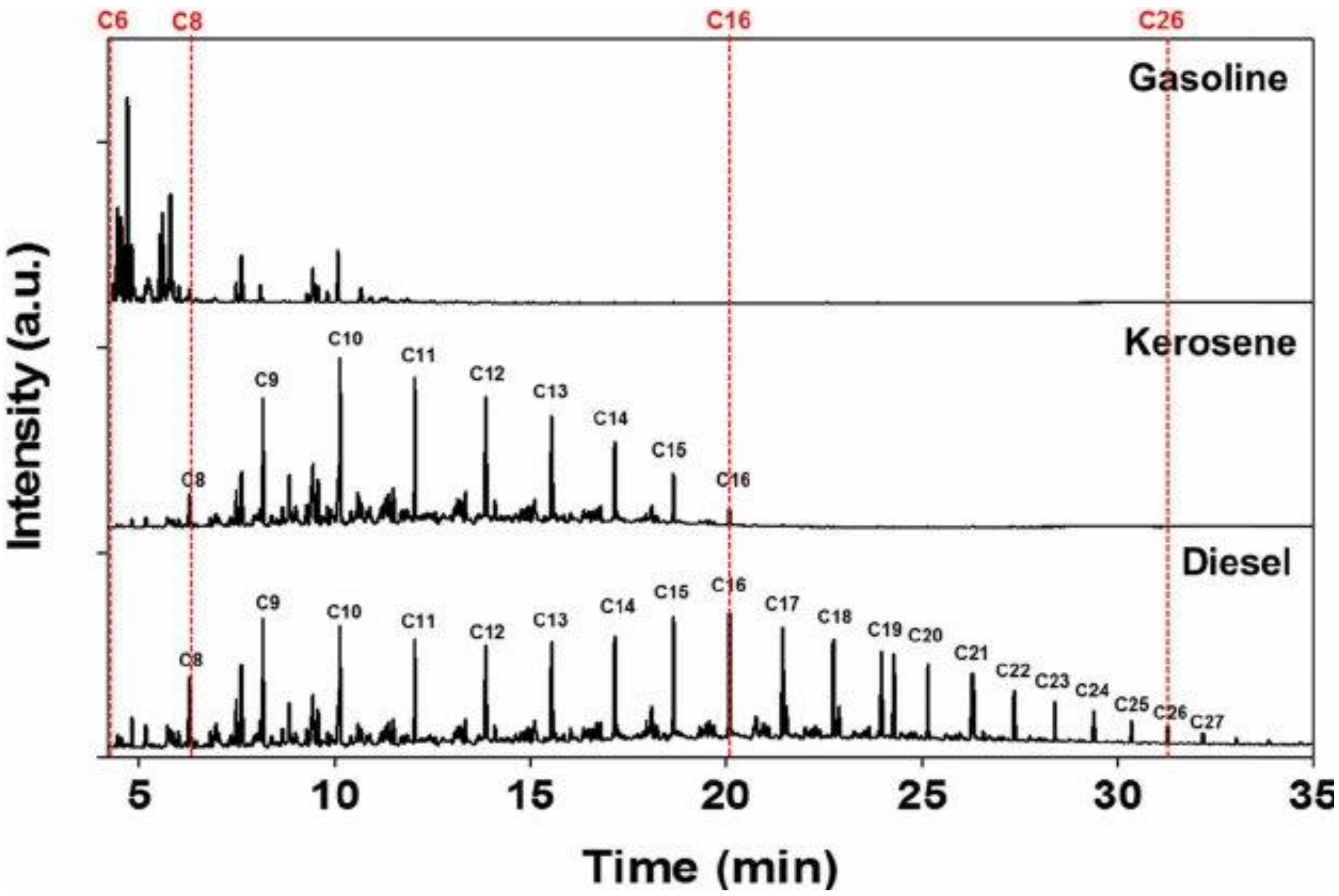
Source: http://hiq.linde-gas.com/en/analytical_methods/gas_chromatography/index.html

GC chromatogram

Example

- GC chromatograms of refined petroleum products

- The other name of Gasoline is … petrol



Critical to find out what are the components existing in each product

Different components CXX have different intensity levels!

Source: https://www.researchgate.net/publication/275046766_Identification_of_refined_petroleum_products_in_contaminated_soils_using_an_identification_index_for_GC_chromatograms/figures?lo=1

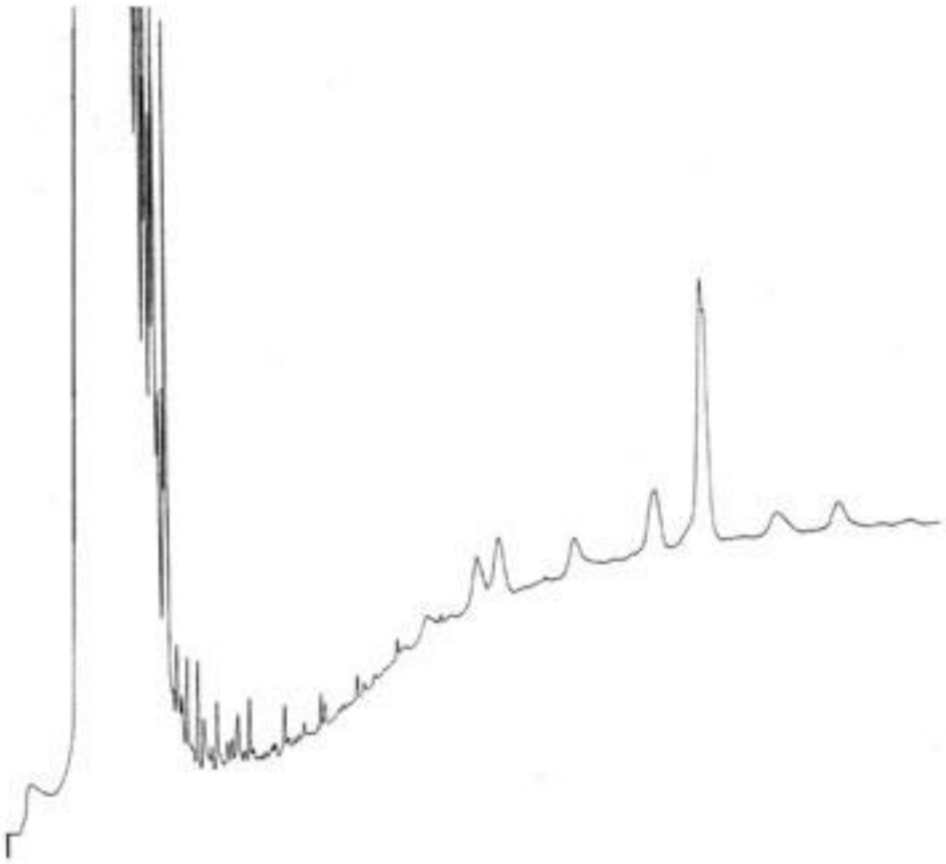
GC chromatogram

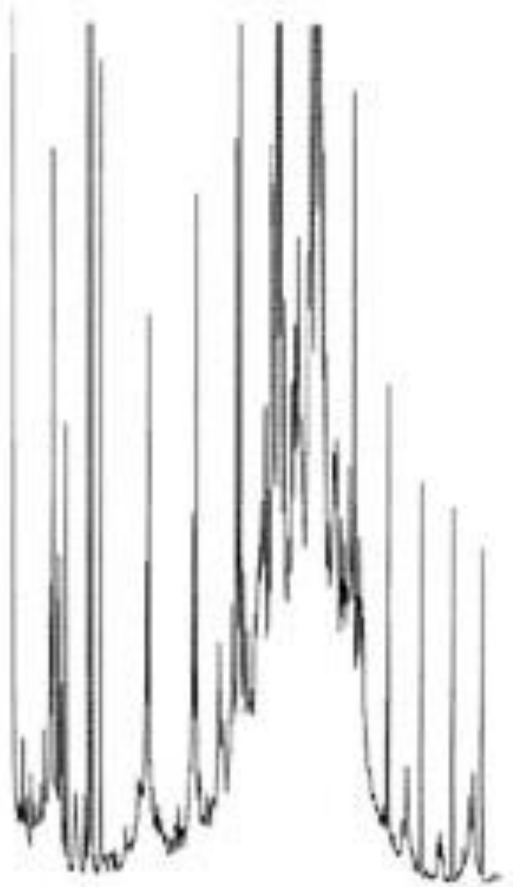
Example

- Sometimes the signals acquired are not ideal

Baseline wandering examples

- For example ….





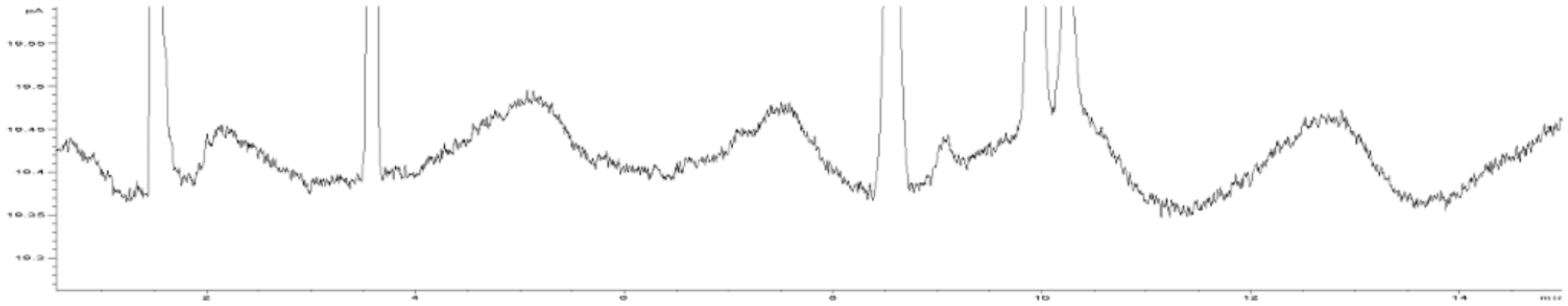
Source: https://chem.libretexts.org/Bookshelves/Analytical_Chemistry/Supplemental_Modules_(Analytical_Chemistry)/Chromedia/01Gas_Chromotography_(GC)/Gas_Chromotography%3A_Basic_Theory/27Some_%27bad%27_GC_chromatograms

GC chromatogram

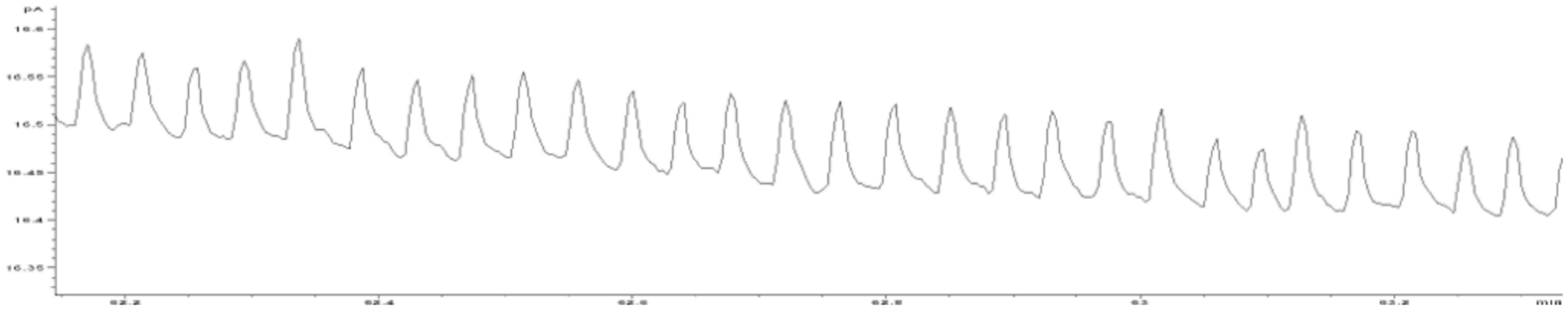
Example

- Sometimes the signals acquired are not ideal

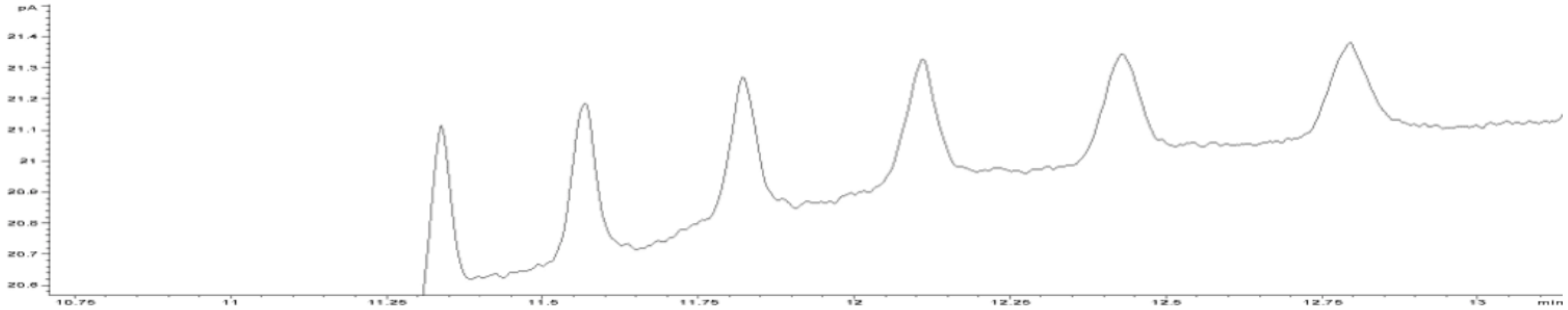
Slow, Low Frequency Oscillations



Fast, Higher Frequency Oscillations



Temperature Dependent Oscillations

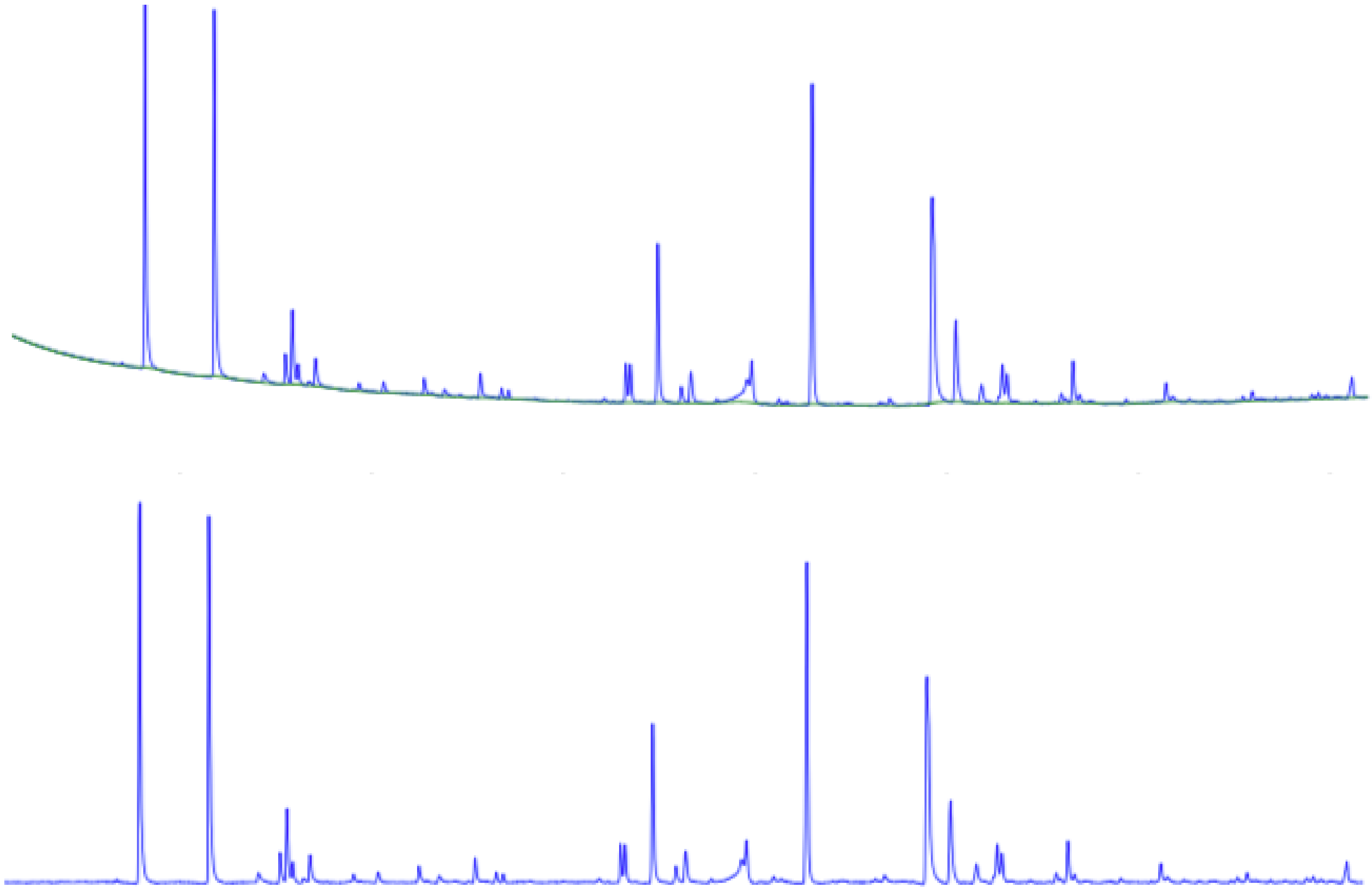


Source: https://www.phenomenex.com/Info/Page/baselineosc

Baseline correction

Example

- Need a solution to correct baseline

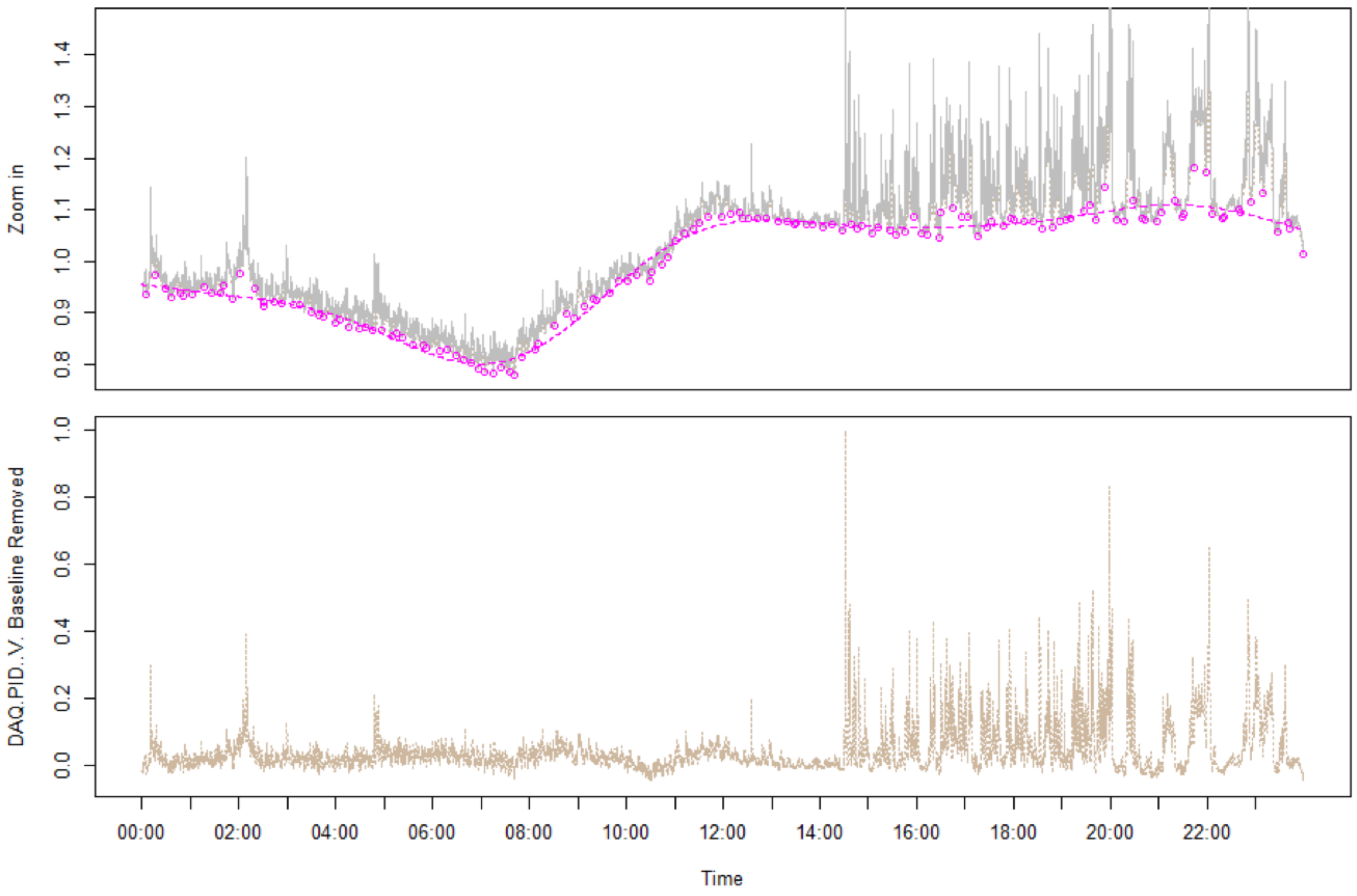


Source: "Baseline correction with asymmetric least square", by Eilers and Boelens

Air quality

Monitoring of particle concentration

- Low cost air sensors have issues of baseline drift, not so useful for particle concentration, but spikes could characterize emissions events



Source: https://cfpub.epa.gov

Baseline correction

Basic idea

Derivation steps (NOT important!)

- Assume **y** is the original signal, **z** as the other signal which has this two properties: smooth, faithful to **y**

$$\mathbf{y} = [y_1, y_2, \ldots, y_i, \ldots y_L]$$
$$\mathbf{z} = [z_1, z_2, \ldots, z_i, \ldots z_L]$$

- The baseline can be estimated by minimizing the penalized least squared function

First term here measures the fit to the data; second term is a penalty on non-smooth behavior of z; parameter lambda tunes the balance between the two terms

$$S = \sum_i w_i (y_i - z_i)^2 + \lambda \sum_i (\Delta^2 z_i)^2$$

- where

$$\Delta^2 z_i = (z_i - z_{i-1}) - (z_{i-1} - z_{i-2}) = z_i - 2z_{i-1} + z_{i-2}$$

Source: https://www.mathsisfun.com/calculus/maxima-minima.html

Baseline correction

Basic idea

yc is the parameter of interest but easy to calculate z or yc?

Usually y is known and z is not. Need to find z here first in order to find yc!

What's the solution?

- The minimization problems leads to the below system of equations

$$(W + \lambda D'D)z = Wy$$

- The baseline corrected signal is then

$$y_c = y - z$$

$$W = \begin{bmatrix} w_1 & 0 & 0 & \cdots & 0 \\ 0 & w_2 & 0 & \cdots & 0 \\ 0 & 0 & w_i & \cdots & 0 \\ \vdots & \vdots & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & w_L \end{bmatrix}$$

Weights

$$\begin{array}{cc} \text{L-2 columns} & \\ D = \begin{bmatrix} 1 & & & \\ -2 & 1 & & \\ 1 & -2 & \ddots & \\ & 1 & \ddots & 1 \\ & & \ddots & -2 \\ & & & 1 \end{bmatrix} & \text{L rows} \end{array}$$

Diagonal sparse matrix

Source: https://www.mathsisfun.com/calculus/maxima-minima.html

Baseline correction

The code

These codes will help you correct the baseline using the previous formulas

- Define function `alsbase`
- Suggested hyperparameter values:

$$0.001 \leq p \leq 0.1$$

$$10^2 \leq \lambda \leq 10^9$$

- Number of iterations: 5 to 10

```
> from scipy import sparse
> from scipy.sparse.linalg import spsolve

> def alsbase(y, lam, p, niter=10):
      L = len(y)
      D = sparse.diags([1,-2,1],[0,-1,-2], shape=(L,L-2))
      w = np.ones(L)

      for i in range(niter):
          W = sparse.spdiags(w, 0, L, L)
          Z = W + lam * D.dot(D.transpose())
          z = spsolve(Z, w*y)
          w = p * (y > z) + (1-p) * (y < z)
      return z
```

Note that:

y = data signal

niter = No. of iterations

Baseline correction

Correct the ecg1D

Rmb yc = y – z →

```
> ecgbase      = alsbase(ecg1D, 10^5,0.000005,niter=50)
  ecgcorr      = ecg1D-ecgbase
```

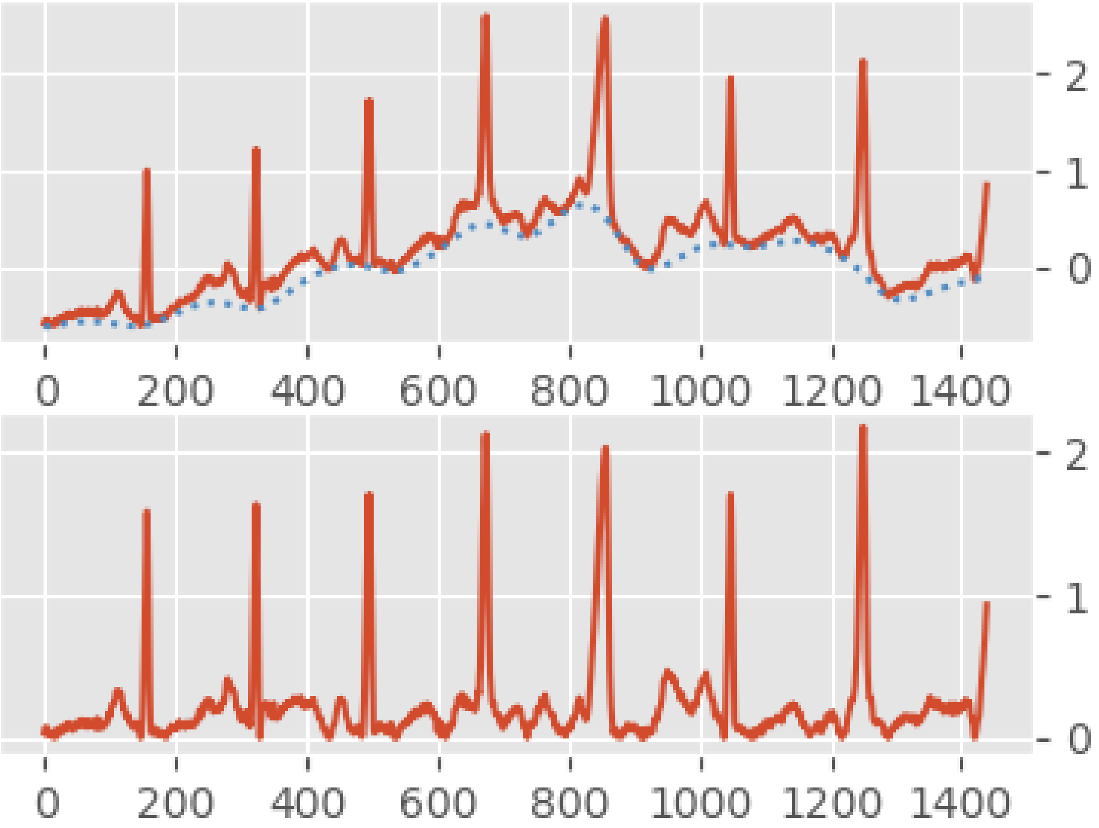
Qn: Can this method be used for real time processing?

- Put ecg1D into alsbase

- Plot the output

```
> plt.figure()
> plt.subplot(211)
> plt.plot(ecg1D)

> plt.plot(ecgbase,
           color="C1",
           linestyle='dotted')
> plt.subplot(212)
> plt.plot(ecgcorr)
```



Workshop

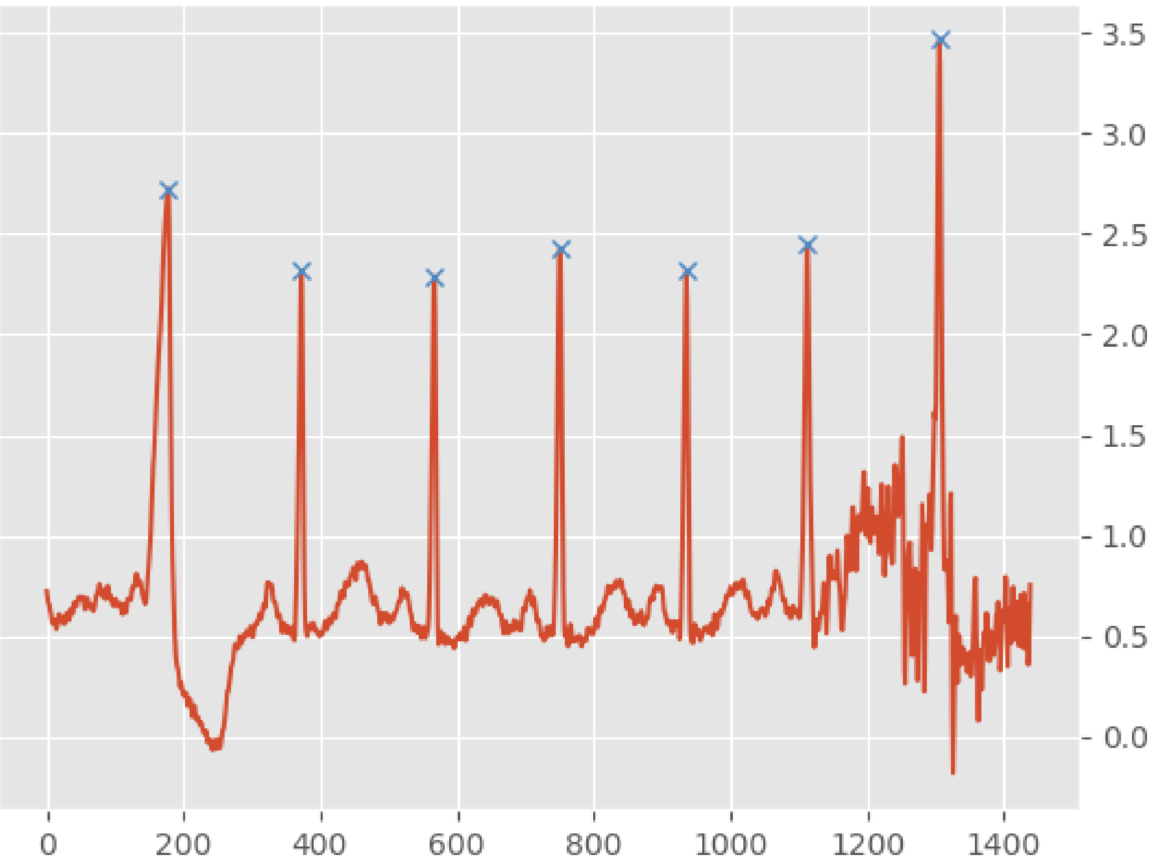
Detect all the peaks in all the ECGs and perform baseline correction

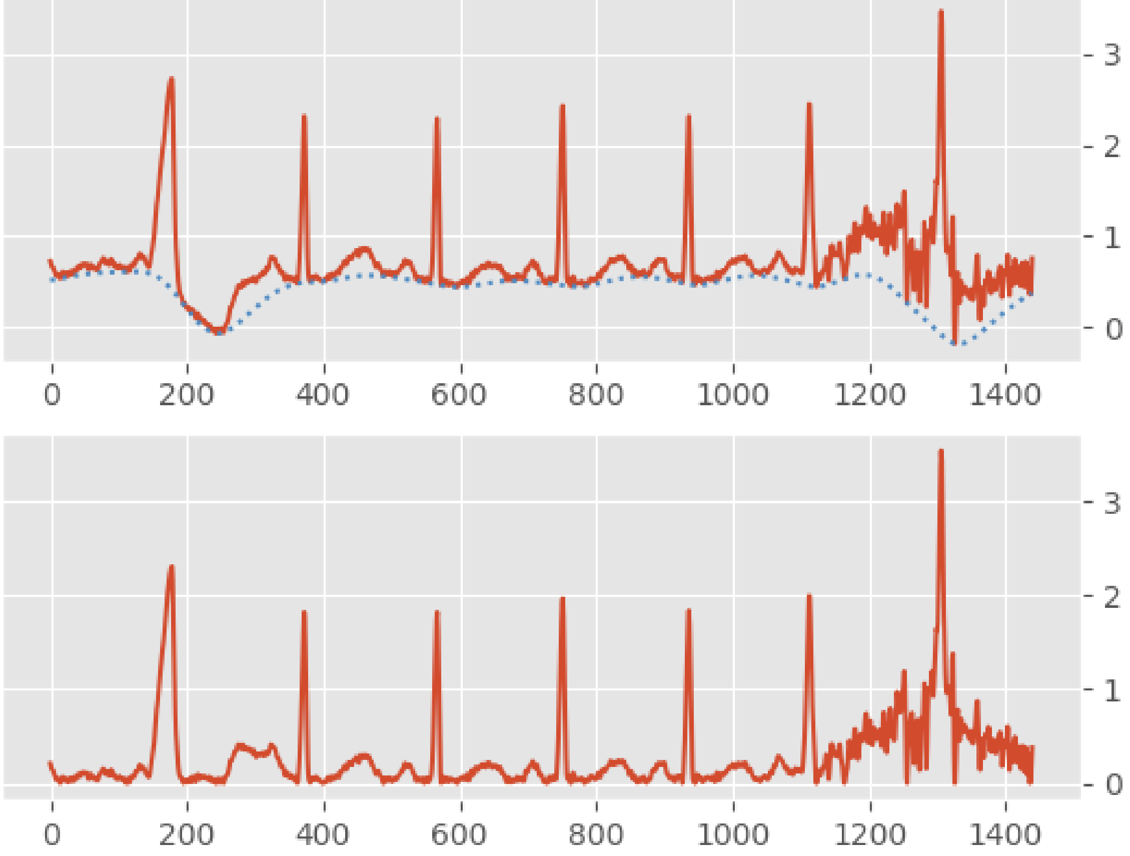
- Use pandas to read in csv

```
> import pandas as pd
> l2D  = pd.read_csv('ecg2D.csv',
                     header=None)
```

Note that for this workshop, use ecg2D data; ecg1D is for you to test out

ecg2D data is a representative of 3 various signals (choose 1 column for your workshop; if you cannot decide, just choose the 2nd column)


3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000
1200
1400


3
2
1
0
0
200
400
600
800
1000
1200
1400
3
2
1
0
0
200
400
600
800
1000
1200
1400