Gartley Pattern recoGnizer

**Note:** *This program was made on the request of my elder brother, who works as a freelancer FOREX, Crypto trader/market analyst. This is a project made by the two of us, with him providing the trading knowledge, verification of the results’ correctness, and me developing the program itself.*

**Mathematical background**:

1. Fibonacci

Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

Fibonacci ratios **forward**: 0, 1, 0.5, 0.66, 0.6, 0.625, 0.615... **converges to 0.618.**

„Forward” means dividing each number by the number in front of it to get the ratios.

Fibonacci ratios **backward**: inf., 1, 2, 1.5, 1.66, 1.6, 1.625, 1.615... **converges to 1.618.**

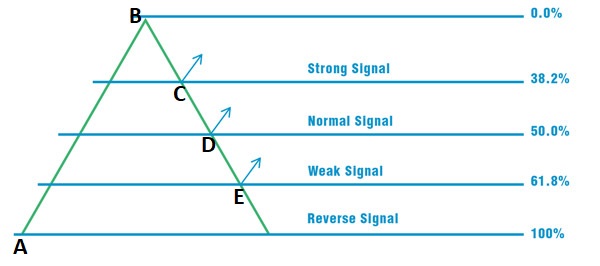
„Backwards” means dividing each number by the number behind it.

There are other ratios, for example dividing each number by the number located 2 steps away (1/2 or 3/8) converges to a different number. There’s a whole family of Fibonacci ratios.

1/0.618 = 1.618 and 1/1.618 = 0.618 🡪 „**Golden Ratio**”

2. Retracements

A retracement is basically a price movement that goes against the price movement before it, „retracing” that earlier movement.



C = 38.2 Retracement of AB

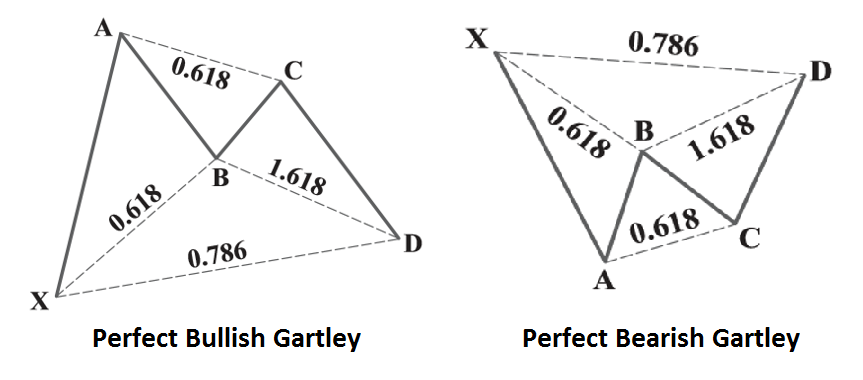
D = 50.0 Retracement of AB

E = 61.8 Retracement of AB

For example, if we were to take the price difference between A & B, and calculate the fibonacci ratios, we would get the length of the BE retracement, which is AB \* 0.618 (61.8 % of AB).

So putting these retracements together we get patterns.

**Gartley pattern**:



AB is a retracement of XA, BC is a retracement of AB, and CD is a retracement of BC.

Dashlines indicate just about how much of a retracement that retracement should be.

Example: XA \* 0.618 = AB, AB \* 0.618 = BC, BC \* 1.618 = CD or XA \* 0.786 = CD.

These numbers are for demonstration only, they may vary, although not by a great margin. XA is the longest, AB should be close to CD in length and the full (ABCD) retrace should be between 0.618 and 0.786 for this pattern to work.

**Basic implementation**:

After importing our current .csv dataset to work on, we need to identify our price points (X,A etc.). In order to do that, we must find local (relative) extremas/peaks. This procedure is implemented in harmonic\_patterns.py’s **„find\_peak”** **function**, with the help of an imported function (from scipy) called argrelextrema. The most important thing in using argrelextrema is to correctly set the number for it’s „order” parameter, which ultimately determines how many peak points we will get (how frequently).

We will examine five extremas and their distance at one time. If their distances satisfy the conditions declared in harmonic\_pattern.py’s **„is\_gartley” function**, and the movements from one point to another draw an either „up-down-up-down” or „down-up-down-up” pattern, we’ve found ourselves a potential (bullish or bearish) Gartley pattern. It is potential because after finding one, it’s up for the trader using the alogrithm to correctly evaluate the given situation.

Upon finding a plausiable pattern, the program draws it on top of the price values and the previously found peaks on a single plot. It also prints the type and the date of the pattern to the console.

**Future plans**:

* Using an API instead of csv-s to get data, automatizing the program to run once or twice a day on different hour (1, 4 etc.) data
* Implementing other patterns besides Gartley, like the Butterfly, or Bat pattern
* Making predictions based on results
* ...

**Feature update 1:**

Integrates API connection to Binance and Bitmex.

The program is now capable of getting price data from Binance, Bitmex or from an existing .csv file into a pandas dataframe, and running the gartley finder algorithm on these dataframes.