Uniwersytet Jagielloński w Krakowie
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Karolina Góra
Nr albumu: 1143418

Porównanie różnych algorytmów obliczania wykładnika Hursta

Praca licencjacka
na kierunku Informatyka

Praca wykonana pod kierunkiem
dr hab. Paweł Góra prof. UJ
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Kraków 2020

Oświadczenie autora pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Kraków, dnia Podpis autora pracy

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Kraków, dnia Podpis kierującego pracą

Abstract

Implementation of algorithms *Rescaled Range* (R/S), *Detrended Fluctuation Analysis* (DFA) and *Detrended Moving Average* (DMA). Observed how, for different data series, each method creates a straight, which directional factor is a value of Hurst exponent and what possible discrepancies there can appear.

All methods are programmed in Python3, with use of ready numerical library (numpy) for calculations and graphic library (matplotlib) for generating final charts.

Abstrakt

Do analizy szeregów czasowych wykorzystuje się szeroki wachlarz metod i technik, których celem jest zbadanie charakteru tych danych i ukrytych pod nimi procesów oraz możliwości określenia ich zachowań w przyszłości na podstawie dostępnych danych. W niniejszej pracy przedstawiono metody wykorzystywane do analizy i wykrywania procesów charakteryzujących się tzw. długą pamięcią: metoda analizy długozasięgowej (ang. Rescaled Range), metoda analizy odtrędowionych fluktuacji (Detrended Fluctuation Analysis), metoda odtrędowionej średniej kroczącej (Detrended Moving Average). Przeprowadzono analizę wyników uzyskanych przy pomocy poszczególnych metod dla różnych ciągów danych oraz wskazano zaobserwowane rozbieżności.

Wszystkie metody zostały zaprogramowane w języku Python3, z wykorzystaniem gotowych bibliotek numerycznych (numpy) do obliczeń oraz biblioteki graficznej (matplotlib) do generowania wykresów wynikowych.

Spis treści

1. Wstęp

1.1. O wykładniku Hursta

1.1.1. Geneza

Przewidywanie przyszłości zaprzątało umysły i rozbudzało wyobraźnię ludzi od najdawniejszych czasów. Obok poszukiwania kamienia filozoficznego oraz eliksiru młodości było przedmiotem niezliczonych badań i dociekań najbardziej przenikliwych umysłów na przestrzeni dziejów, by wymienić tylko takich gigantów jak Fermat, Pascal, Jakub Bernoulli, Gauss, Laplace czy Boltzmann. Wprawdzie cel nie został do dzisiaj osiągnięty i być może nigdy do tego nie dojdzie, tym niemniej w ostatnich stuleciach dokonano liczących się odkryć oraz odniesiono sporo sukcesów, mających swój bardzo praktyczny wymiar.

Lepsze poznanie i zrozumienie zjawisk występujących w przyrodzie wymaga przede wszystkim zgromadzenia odpowiednich danych. Im większa liczba (częstotliwość) i lepsza jakość (regularność) zebranych danych, tym większe szanse na odkrycie praw rządzących badanymi procesami. Istnieje wiele sposobów gromadzenia danych. W niniejszej pracy skupiono się na analizie zbiorów danych charakteryzujących się równymi odstępami czasu (interwałami) pomiędzy poszczególnymi obserwacjami (pomiarami), czyli analizie szeregów czasowych.

W analizie szeregów czasowych dotyczących tak zjawisk fizycznych, jak i procesów finansowych i ekonomicznych, kluczową kwestią jest rozstrzygnięcie, czy dane je opisujące pozostają ze sobą w jakiejkolwiek zależności, czy może są czysto losowe? Na szczęście istnieje wiele technik umożliwiających udzielenie odpowiedzi na to pytanie.

Jedną z metod służących do określania, czy w danym ciągu czasowym istnieje jakakolwiek zależność, jest tzw. analiza długozasięgowa oparta o wyznaczanie wykładnika Hursta. Badania, które doprowadziły do stworzenia powyższej metody pierwotnie były prowadzone w dziedzinie hydrologii, na początku XX wieku. Na podstawie obszernych danych o wylewach Nilu, próbowano określić optymalne wymiary tamy, która pozwalałaby na regulowanie poziomu wody w warunkach niestabilnych opadów i susz w dorzeczu Nilu. Przedsięwzięcie to pokazało, że zmiany poziomu wody nie są zjawiskiem całkowicie losowym oraz odkryto istnienie długoterminowej pamięci zdarzeń, dla nieskończenie długich szeregów czasowych. Warto nadmienić, że do wykonujący obliczenia Harold Edwin Hurst miał do dyspozycji dane z 847 lat. Wykładnik swoją nazwę otrzymał na cześć wyżej wspomnianego hydrologa brytyjskiego, który był głównym członkiem zespołu prowadzącego wspomniane badania. Można spotkać się z określeniami, że wykładnik ten, jest *indeksem zależności* lub *indeksem długo-zasięgowej (long-range) zależności*. W geometrii

fraktalnej wykładnik Hursta jest oznaczany przez symbol **H**, co jest uhonorowaniem Harolda Hursta oraz Ludwiga Otto Höldera przez Benoit'a Mandelbrota, twórcy tej gałęzi matematyki.

W geometrii fraktalnej wykładnik H mierzy *moc łagodnej* lub *dzikiej* losowości serii danych. Znając jego wartość można oszacować prawdopodobieństwo utrzymania się trendu badanego szeregu, określenia czy posiada tendencję do silnej regresji do średniej, czy też nie wykazuje żadnej zależności.

1.1.2. Interpretacja wartości wykładnika

Podczas badań nad zmianami poziomu Nilu, dostrzeżono potęgową zależność pomiędzy odchyleniem standardowym σ , a długością analizowanego szeregu czasowego *n*:

$$\sigma = n^H$$

gdzie *H* odpowiada wykładnikowi Hursta.

Logarytmując obie strony powyższego równania $\ln(\sigma) = \ln(n^H)$, można przejść do postaci będącej równaniem prostej: $\ln(\sigma) = H \cdot \ln(n) + const$ której współczynnik kierunkowy, wyznaczony przy pomocy regresji liniowej, jest wykładnikiem Hursta – H. Wykres prostej przedstawia się w układzie kartezjańskim z osiami w skali logarytmicznej.

W praktyce zależność pomiędzy wyznacznikiem, a współczynnikiem kierunkowym prostej, można zdefiniować na dwa sposoby:

1. Przedstawiając widmo mocy badanego ciągu na wykresie podwójnie logarytmicznym i dopasowanie do niego prostej. Wtedy $H = \frac{1-\alpha}{2}$, gdzie α to współczynnik kierunkowy.
2. Przedstawiając na wykresie podwójnie logarytmicznym dane będące serią ułamkową i tworzące zależność $F(L) \sim L^{\alpha}$, gdzie $H = \alpha$.

Drugi z przedstawionych sposobów opiera się na tej samej zależności potęgowej, którą odkrył w czasie badań Hurst, dlatego w niniejszej pracy wykorzystano go do przedstawiania wszystkich uzyskanych wyników.

Wartość wykładnika H znajduje się w zakresie $[0,1]$. Wynika to z własności prawdopodobieństwa, które maksymalnie może wynosić 1. Uzyskanie wartości brzegowych 0 i 1 oraz wartości 0.5 jest rzadko spotykane, przez co często uważane są za czysto teoretyczne.

Wartość otrzymanego wykładnika Hursta, informuje o charakterze badanego szeregu czasowego:

- ➔ $H = 0.5$ – szereg nieskorelowany, czyli dane tworzą biały szum. Oznacza to, że szereg zachowuje się jak błądzenie losowe i nie posiada ukierunkowanego trendu.
- ➔ $0 < H < 0.5$ – korelacja negatywna – szereg czasowy, w którym przez długi czas dane będą często i szybko zmieniać swój kierunek – po wartości wysokiej prawdopodobnie

nastąpi wartość niska, a po niej znów wysoka z tą samą tendencją do przełączania się między wartościami wysokimi i niskimi.

- ➔ $0.5 < H < 1$ – długoterminowa dodatnia autokorelacja. Ciąg przedstawia długo-zasięgową zależność (Long Range Dependence), to znaczy, że gdy ciąg ma tendencję wzrostową (lub malejącą), to prawdopodobnie ten trend się utrzyma. Badane zdarzenie potrzebuje silnego bodźca, aby spowodować zmianę trendu.

1.1.3. Metody wyznaczania wykładnika Hursta

Istnieje wiele metod pozwalających na wyprowadzenie wartości wykładnika *H*. Dobiera się je w zależności od badanych szeregów. Do serii czasowych jednowymiarowych, do których ograniczona została niniejsza praca, można zastosować algorytmy takie jak:

- ➔ R/S – metoda analizy długozasięgowej (ang. Rescaled Range);
- ➔ DFA – metoda analizy odtrędowionych fluktuacji (Detrended Fluctuation Analysis);
- ➔ DMA – metoda odtrędowionej średniej kroczącej (Detrended Moving Average).

Pierwsza z wymienionych metod – R/S, jest pierwotną metodą estymacji wykładnika Hursta. H.E. Hurst zdefiniował ją jako funkcję przedziałów w szeregu czasowym:

$$\frac{R(n)}{S(n)} = C \cdot n^H \quad przy \quad n \to \infty$$

gdzie:

- ➔ R(n) – jest to *zasięg* pierwszych *n* złożonych *odchyleń* od średniej;
- ➔ S(n) – jest ich odchyleniem standardowym;
- ➔ n – wielkość przedziału czasowego, segmentu, na którym przeprowadzane są obserwacje;
- ➔ C – pewna stała.

Każdy z wyżej wspomnianych algorytmów opiera się na wielokrotnym przebadaniu serii danych, dla przedziałów o różnej wielkości *n.* Celem użytych w nich metod jest otrzymanie, opisanej w podrozdziale wcześniej, zależności potęgowej między długością serii danych, a parametrem charakterystycznym dla danego algorytmu. Na przykład w metodzie R/S jest to relacja zasięgu odchyleń od średniej, do odchylenia standardowego na przedziale.

Metoda DFA dochodzi do pożądanej zależności skupiając się na fluktuacji wokół lokalnego trendu danych, zamiast na zasięgu sygnałów. Z tego powodu może być stosowana nie tylko do wyznaczania wykładnika Hursta, ale także posiada szersze zastosowanie, przy badaniu sygnałów, których podstawowe statystyki (np. średnia, wariancja) lub dynamika zmieniają się w czasie. Jest ona rozszerzeniem zwykłej analizy fluktuacji (FA) i została wprowadzona przez fizyka Chung-Kang Peng w 1994 roku. Do wyznaczenia wykładnika H należy wyprowadzić średnią wartość

fluktuacji, dla różnych wielkości przedziałów czasowych. Na ich podstawie odczytuje się wskaźnik korelacji danych.

Ostatnia z metod, metoda DMA, jest algorytmem o najmniej skomplikowanej strukturze. Opiera się ona bowiem na doborze przedziału – okna badawczego – i przemieszczeniu się nim, jak suwakiem, po badanych danych. W ten sposób wyznacza się średnią kroczącą, dla okna o wybranej wielkości n. Następnie modyfikuje się wartości szeregu czasowego o wyznaczone średnie. Na podstawie powstałego, zmodyfikowanego ciągu odczytuje się szukaną zależność, pomiędzy długością okna czasowego, a średnią znajdujących się w nim wartości.

Oprócz wspomnianych już metod, do wyznaczania wykładnika H służą także algorytmy falkowe, którym poświęcona została równolegle wykonana praca Pana Szymona Peszka pod tytułem *„Porównanie różnych algorytmów wyznaczania wykładnika Hursta”*. Algorytmów falkowych jest bardzo wiele, żeby wymienić tylko kilka z nich, takich jak falka Haar’a, Daubechie rzędu 4, Coiflet’a rzędu 1 czy Symlet rzędu 10. Bazują one na dyskretnej transformacji falkowej oraz określonych bazach falkowych. Celem tych algorytmów jest wyznaczenie tak zwanego widma falkowego i dopasowania do niego prostej, która spełnia takie samo zadanie, jak w opisanych na początku metodach.

1.2. Wyjaśnienie pojęć

1.2.1. Korelacja

Korelacja przedstawia statystyczny związek pomiędzy dwiema zmiennymi losowymi X i Y. Znając wartość jednej z nich, można określić (przynajmniej w niektórych przypadkach) dokładną lub przybliżoną wartość tej drugiej.

W statystyce analiza korelacji polega na zbadaniu, czy dwie zmienne są ze sobą istotnie powiązane, doszukując się współzależności pomiędzy ich dowolnymi cechami, atrybutami lub własnościami. To dlatego wspomniane wcześniej trzy metody estymacji wykładnika Hursta, mimo że każda opiera się na badaniu innego parametru szeregu, pozwalają na wyciągnięcie podobnego wniosku.

W analizie korelacji nieistotne jest wyjaśnienie dlaczego wykryta zależność ma miejsce, ale udzielenie odpowiedzi na trzy pytania:

1. Czy w badanym zjawisku występuje związek?
2. Jaki jest współczynnik korelacji? (W odniesieniu do niniejszej pracy, odpowiada on wykładnikowi Hursta)
3. Jak silny jest związek pomiędzy badanymi zmiennymi?

Wynik analizy korelacji graficznie prezentuje się na tak zwanym wykresie rozrzutu. W przypadku wykładnika Hursta jest to, wcześniej już wspomniany, wykres podwójnie logarytmiczny, obrazujący zależność między długością przedziału czasowego, a badaną własnością szeregu.

1.2.2. Odchylenie standardowe

Odchylenie standardowe (σ) jest pojęciem z dziedziny statystyki, zaliczanym do miar rozproszenia, przeznaczonych do badania stopnia zróżnicowania wartości zmiennej. Jej zadaniem jest określenie, o ile średnio wartości w badanym szeregu czasowym odchylają się od średniej arytmetycznej badanej zmiennej.

Niskie wartości σ świadczą o małym rozproszeniu danych, czyli ich niewielkim zróżnicowaniu. Wysokie wartości odchylenia przeciwnie, oznaczają silne rozproszenie.

1.2.3. Random Walk

Błądzenie losowe (ang. Random Walk) jest pojęciem z zakresu matematyki i fizyki, określającym ruch przypadkowy. Oznacza to, że w kolejnych momentach czasu, cząstka (obserwowany obiekt) przemieszcza się z aktualnego położenia do innego, w losowy sposób.

W niniejszej pracy, błądzenie losowe będzie wykorzystane w celu modyfikacji rozproszonych danych, przed rozpoczęciem ich przetwarzania. Algorytm transformujący pierwotny szereg czasowy przy pomocy metody Random Walk, opiera się na zamianie każdej wartości na sumę wartości ja poprzedzających, zmodyfikowanych o średnią arytmetyczną całej serii czasowej. Operację tę przedstawia równanie $X_n=\sum_{k=1}^{N}\left(x_k-\langle x_n\rangle\right)$, gdzie:

- ➔ X_n - transformowany element serii czasowej
- ➔ N - długość szeregu czasowego
- ➔ x_k - k-ta wartość szeregu czasowego
- ➔ $\langle x_n\rangle$ - średnia arytmetyczna z całej serii czasowej

1.2.4. Fluktuacja

Fluktuacja, inaczej wahania przypadkowe, są to niedające się przewidzieć odchylenia od wartości średniej zmiennej losowej, podlegającej losowym zmianom w czasie i nie wykazujące żadnej tendencji. Fluktuacje są ściśle związane z błędami statystycznymi oraz prognostycznymi. Pojawiają się jako składowa szeregu czasowego (trend + okresowość + wahania przypadkowe). Dla wielkości proporcjonalnych do liczby N serii danych, rozproszenie danej wielkości A(t) związane z fluktuacjami $\sigma^2(A)=\langle A^2\rangle-\langle A\rangle^2$ jest proporcjonalne do N. Z tego wynika względna fluktuacja: $\frac{\sigma(A)}{A}=\frac{\sqrt{N}}{N}=\frac{1}{\sqrt{N}}$. Taką zależność zaobserwować będzie można w rozdziale 2, podczas opracowywania metody DFA, która oblicza fluktuację lokalnego trendu w celu wyprowadzenia wykładnika Hursta.

1.3. Technologie i narzędzia

1.3.1. Język programowania

Metody numeryczne wymagają zadbania o szybkość wykonania, przejrzystość zapisu i ograniczenie ich złożoności do minimum. Dla analizowanych w tej pracy algorytmów, istotne jest efektywne czytanie plików, zawierających tysiące rekordów do przetworzenia oraz możliwość zobrazowania osiągniętych rezultatów. Powyższe wymagania spełania język Python.

Python3 - posiada technologię, pozwalającą na swobodne i łatwe odczytywanie plików tekstowych. Pozwala na modelowanie dużych zbiorów danych i oferuje wbudowane funkcje do ich analizy. Dobrze zaimplementowane biblioteki numeryczna (numpy) i graficzna (pyplot), dają możliwość szybkiej i wydajnej implementacji algorytmów oraz wizualizacji ich wyników.

1.3.2. Biblioteki pomocnicze

Korzystając z Python3, do implementacji opisywanych w pracy metod użyte zostały dwie z wbudowanych bibliotek:

- **numpy** – rozbudowana biblioteka funkcji służących do obliczeń w metodach numerycznych i wykonywania ich z najmniejszą złożonością obliczeniową. Funkcje wykorzystane do implementacji omawianych metod, to:
  - **average()** – funkcja zwraca średnią z wartości przekazanych w tablicy;
  - **polyfit()** – funkcja służy do interpolacji krzywej zadanego stopnia n, na podstawie stabelaryzowanych danych (tablica indeksów oraz tablica wartości im odpowiadająca) i zwraca współczynniki dopasowanej krzywej. Jeżeli podany stopień krzywej wynosi 1, zwrócone zostaną współczynniki prostej.
  - **log()** – funkcja zwraca logarytm naturalny z podanego argumentu. Może także wykonać mapowanie, przekształcając tablicę z wartościami, na tablicę z logarytmami tych wartości;
  - **sqrt()** – funkcja zwraca pierwiastek z podanego argumentu.
- **pyplot** – biblioteka funkcji do tworzenia wykresów danych. Wykorzystane zostały funkcje:
  - **scatter()** – definiuje wykres jako punktowy (styl wykresu);
  - **title()** – nadaje tytuł wykresu;
  - **ylabel()** – nadaje nazwę osi Y;
  - **xlabel()** – nadaje nazwę osi X;
  - **text()** – pozwala na dodanie tekstu w polu wykresu. Funkcja ta została wykorzystana do wyświetlania współczynnika kierunkowego α prostej;
  - **plot()** – generuje wykres o atrybutach nadanych mu funkcjami wyżej;

- **show()** – wyświetla utworzony wykres.

1.3.3. Środowisko pracy

Jako środowisko pracy wykorzystane zostały JetBrains PyCharm oraz Visual Studio Code.

Program PyCharm jest przystosowany specjalnie do pracy z językiem Python oraz jego bibliotekami. Pozwala on na estetyczne pisanie kodu, jego automatyczne formatowanie i wykrywanie błędów. Posiada wygodny tryb pracy, umożliwiający systematyczny podgląd i zapis generowanych przez program wykresów.

1.4. Dane

1.4.1. Typ danych

Tak jak zostało wcześniej wyjaśnione, analizowane algorytmy wykorzystuje się przy szeregach czasowych jednowymiarowych. Szeregi takie muszą spełniać jeden kluczowy warunek:

- seria danych musi być szeregiem, gdzie pomiary zostały wykonane w równych odstępach czasowych.

Dodatkowo, jeśli jest to możliwe, powinien być to szereg ułamkowy, dla zwiększenia dokładności wykonywanych obliczeń.

1.4.2. Wybrane zbiory danych

Wszystkie metody estymacji współczynnika Hursta, przedstawione w drugiej części pracy, zostały wykonane dla różnych serii danych. W części trzeciej przedstawiono i porównano uzyskane wyniki. Wybrane dane wejściowe zapisane zostały w plikach:

- **nile.txt** – dane na temat poziomu wody rzeki Nil;
- **births.txt** – liczba urodzeń (dziennie) dziewczynek w roku 1959 w Californii;
- **temp.txt** – minimalna dzienna temperatura w Melbourne (lata 1981-1990);
- **zurich.txt** – liczba zaobserwowanych plam słonecznych w danym miesiącu w Zurichu (lata 1749- 1983);
- **assigment4.txt**
- **nino.txt** – temperatury powierzchni morza (264 rekordy);
- **warner.txt** – dane oferowane przez bibliotekę python – *pywelvets* dotyczącą zagadnień falek.

2. Implementacje

2.1. Funkcje pomocnicze

2.1.1. prepareData() - przygotowanie tablic z danymi to przetwarzania

W pliku o nazwie *prepareData.py* znajduje się funkcja, przygotowująca do przetwarzania dane, z podanego źródła. Każdy zapisany rekord składać musi się z dwóch wartości. Jedna jest wartością porządkującą, a druga zapisaną obserwacją, która zostanie poddana analizie. Każdy rekord zapisany jest w nowej linii, a jego dane oddzielone spacją.

```
5  def prepareData(file, max_n, min_n):
6      X = []
7      indexes = []
8      with open(file) as data:
9          for line in data:
10             i, value = line.split()
11             indexes.append(int(i))
12             X.append(float(value))
13
14     L = []
15     if max_n:
16         w = max_n
17     else:
18         w = len(X)
19
20     while w / 2 > min_n:
21         if w % 2 == 0:
22             L.append(int(w / 2))
23             w = w / 2
24         else:
25             w -= 1
26
27     return indexes, X, L
```

funkcja przygotowująca dane tekstowe do przetwarzania przez algorytmy

Funkcja przyjmuje trzy argumenty:

- ➔ file – ścieżka do pliku tekstowego, w którym zapisane są dane;
- ➔ max_n – maksymalna długość przedziałów (n), na jakie dzielone są dane;
- ➔ min_n – minimalna długość przedziałów (n), na jakie dzielone są dane;

i zwraca trzy tablice:

- ➔ indexes – tablica indeksów pobranych z przekazanego pliku;
- ➔ X – tablica zgromadzonych danych, które poddane zostaną obserwacji. Kolejność danych w tablicy X, jest w ścisłym związku z kolejnością danych tablicy *indexes*. Argumentowi *indexes[i]* odpowiada wartość *X[i]* i zależność ta nie może zostać zaburzona;

➔ L – tablica wybranych długości *n* dla badanych przedziałów, ustalonych w taki sposób, że ich ilość w przetwarzanej serii danych jest potęgą 2.

Korzystając z możliwości importowania funkcji między plikami (*from prepareData import **), udostępniono wyżej zaimplementowaną funkcję każdemu z implementowanych algorytmów.

2.2. Algorytm R/S (Rescaled Range)

Algorytm *R/S*, jest podstawową metodą wyznaczania wykładnika Hursta. Wymaga ona dużej liczby zebranych danych – Hurst przeprowadzał analizę na podstawie obserwacji z kilkuset lat i mógł wykonywać obliczenia dla dużych *n*. Przy mniej licznych seriach, algorytm może nie działać zgodnie z oczekiwaniami.

Metoda R/S opiera się na dzieleniu szeregu czasowego na równe przedziały (ang. ranges) o długości *n* i analizie danych w ich obrębie. Jak wspomniane zostało w rozdziale 1.1.3, algorytm ten szuka zależności pomiędzy wielkościami badanych przedziałów czasowych, a odchyleniem standardowym wartości się w nim znajdujących. Przypominając, powiązanie z wykładnikiem H przedstawia równanie:

$$\frac{R(n)}{S(n)} = C\,n^{H} \qquad n \to \infty \qquad C = const.$$

gdzie:

➔ **n** – wielkość przedziałów (ranges) czasowych, na których przeprowadzane są obserwacje (ilość *danych* w nich zawartych);

➔ **R(n)** – jest to *zasięg* pierwszych *n* złożonych *odchyleń* od średniej;

➔ **S(n)** – jest ich odchyleniem standardowym.

1. Algorytm wymaga odpowiedniego przygotowania danych. Korzystając z opisanej w rozdziale 2.1.1 funkcji *prepareData*, otrzymano tablicę danych *X* oraz tablicę *L* z wielkościami przedziałów, dla których została przeprowadzona analiza. W metodzie R/S indeksowanie danych jest zbędnie, ale nie oznacza to, że można zaburzyć kolejność badanych wartości X.

```
9    #0
10   indexes, X, L = prepareData('nile.txt', None, 2)        # pobranie danych
11   N = len(X)                                               # N = ilość badanych danych
12   AVG = []                                                 # tablica, w której zbieramy końcowe wyniki dla każdego n
```

Deklaracja danych wejściowych

Dla zilustrowania działania algorytmu, wykorzystano szereg czasowy z obserwacjami dotyczących poziomów wody w Nilu. Dane wejściowe przedstawiały się w następująco:

➔ $X = [1157.0, 1088.0, 1169.0, 1169.0, 984.0, \ldots, 1205.0, 1054.0, 1151.0, 1108.0]$ - tablica o długości $N = 662$ pomiarów, oraz

➔ tablica $L = [331, 165, 82, 41, 20, 10, 5]$ z wybranymi długościami [*n*] segmentów.

Istotną własnością przedziałów, na które dzieli się serię, jest ich rozłączność – kolejne przedziały nie mogą nachodzić na siebie nawzajem.

Przygotowane w powyższy sposób dane, wykorzystano następnie do wykonania kolejnych kroków obliczeniowych.

```
for n in L:                                 # (1)
    R_S = []                                # wartości R/S dla serii o długości n
    R = []                                  # zbiór największych różnic odchyleń w przedziałach długości n
    S = []                                  # zbiór wartości odchyleń standardowych dla przedziałów o długości n
    i = 0
    while i <= N - n:                       # (2)
        segment = X[i:i+n]                  # wybranie kolejnego segmentu o długości n
        m = np.average(segment)             # wyliczenie średniej dla wybranego segmentu o długości n
        Y = []                              # Seria odchyleń dla danego segmentu
        Z = []                              # tablica sum odchyleń dla wszystkich serii o długości n
        for s in range(i, i+n):             # (3)
            Y.append(X[s] - m)
            Z.append(np.sum(Y))             # zapisanie pełnego odchylenia średniej dla przedziału

        R.append(max(Z) - min(Z))           # (6) Największa rónica odchyleń dla zbadanego podziału
        S.append(satndardDeviation(n, Y))   # (4) Odchylenie standardowe dla wyznaczonego przedziału
                                            # (5)
        i += n                              # wybranie początku następnego przedziału o długości n

    for r, s in zip(R, S):                  # (7)
        if s != 0:
            R_S.append(r/s)                 # (8) wyznaczenie R/S dla każdego przedziału o długości n

    AVG.append(np.average(R_S))             # (9) zapisanie średniej ze wszystkich zebranych wartości R_S[n]
```

Implementacja algorytmu R/S

2. Dla każdego przedziału o długości *n* w serii danych X, obliczono średnią arytmetyczną: $m = \frac{1}{n}\sum_{i=1}^{n} X_i$. Przykładowo, dla segmentów o długości $n = 82$ średnia z pierwszych n danych wynosiła $m = 1153.62$ *.

3. W kolejnym kroku wykonano przeskalowanie, gdzie dane w każdym przedziale o długości *n* skorygowano o wyliczoną dla niego średnią: $Y_t = X_t - m$ (3). Otrzymano tym sposobem nowy szereg czasowy. Dla pierwszych 82 wartości serii danych X, powstał nowy przedział:
$Y = [32.05, -36.95, 44.05, \ldots, -43.95, 71.05, 71.05]$ *.

4. Na podstawie zbudowanej w kroku 3 tablicy *Y*, stworzono serię sum odchyleń Z, zgodnie z równaniem: $Z_t = \sum_{i=1}^{t} Y_i$. Oznacza ono, że każdemu elementowi z_t, przypisuje się sumę wartości tablicy Y od y_0 do y_t. Oba szeregi mają długość *n* oraz $y_0 = z_0$. Odnosząc się do przykładu z danymi dla Nilu, przy $n = 82$ szereg sum odchyleń dla pierwszych *n* elementów badanego szeregu czasowego składa się z wartości:
$Z = [32.05, -4.90, 39.15, \ldots, -142.10, -71.05, 4.55\text{e-}13]$ *.

5. W kolejnym etapie wyznaczone zostało odchylenie standardowe S: $S(n)=\sqrt{\frac{1}{n}\sum_{i=1}^{n} Y_i^2}$. W celu zachowania przejrzystości kodu, zaimplementowano dodatkową funkcję pomocniczą: *standardDeviation (10)*. Funkcja ta zwraca wartość odchylenia standardowego dla badanego przedziału. Jako argumenty przyjmuje:
   - ➔ **n** – rozmiar przedziału.
   - ➔ **Y** – tablica o rozmiarze *n*, zawierająca skorygowane w kroku 3 wartości serii danych X.

```
52  def satndardDeviation(n, Y):
53      cumulative_sum = 0
54      for i in range(0, n):
55          cumulative_sum += Y[i] * Y[i]                # (10)
56      return np.sqrt(cumulative_sum / n)
```

Implementacja funkcji standardDeviation

Po wywołaniu funkcji (4), zwracana przez nią wartość zapisywana jest w tabeli S, jako odchylenie standardowe badanego segmentu danych. Dla analizowanego przykładu $S_{82}[0]=90.64$ *

6. Mając wyliczone wartości S dla wszystkich przedziałów o długości *n*, do obliczenia lewej strony równania R/S brakuje już tylko wartości R, będącej największą różnicą sum odchyleń w badanym segmencie: $R=max(Z)-min(Z)$. Do jej wyliczenia służy tablica Z, zbudowana w kroku 4. Uzyskaną wartość R zapisano do tablicy R, ze zbiorem różnic odchyleń w analizowanych segmentach. Dla danych o Nilu $R_{82}[0]=1077.77$

7. Procedura przedstawiona w punktach od 2 do 6 została powtórzona dla wszystkich kolejnych, rozłącznych przedziałów o długości *n*, czyli dla łącznej liczby segmentów wynoszącej $\lfloor\frac{N}{n}\rfloor$. Poniższa tabela zawiera zestawienie wyników będących rezultatem wykonania powyższych kroków dla wszystkich przedziałów szeregu czasowego X, przy $n=82$:

n = 82				
i	Badany zakres	Średnia przedziału [m] *	Największa różnica sum odchyleń w przedziale - $R_n[i]^*$	Odchylenie standardowe - $S_n[i]^*$
0	0 - 81	1153.62	1077.77	90.64
1	82 - 163	1073.99	2033.66	79.13
2	164 - 245	1129.02	1541.54	89.34
3	246 - 327	1141.97	1502.19	75.34
4	328 - 409	1125.08	1288.69	77.20
5	410 - 491	1190.05	1389.27	74.56
6	492 - 573	1226.57	1481.23	64.48
7	574 - 655	1143.27	64.48	69.20

Tabela 1: Wyniki obliczeń dla danych o Nilu, przy badaniu przedziałów o długości 82 rekordów

8. Następnie dla każdej pary R_i i S_i, obliczono wartość R/S (8) i zapisano w tablicy R_S_n. Wartości R/S dla poszczególnych segmentów analizowanego przykładu zestawiono w tabeli 2:

n = 82				
i	Badany zakres	Największa różnica sum odchyleń w przedziale - $R_n[i]$ *	Odchylenie standardowe - $S_n[i]^*$	$\frac{R_n[i]}{S_n[i]}$ *
0	0 - 81	1077.77	90.64	11.89
1	82 - 163	2033.66	79.12	25.70
2	164 - 245	1541.54	89.34	17.25
3	246 - 327	1502.19	75.34	19.94
4	328 - 409	1288.69	77.20	16.69
5	410 - 491	1389.27	74.55	18.63
6	492 - 573	1481.23	64.47	22.97
7	574 - 655	64.48	69.20	20.89

Tabela 2: Wynik obliczeń wartości RS dla segmentów długości 82

9. W ostatnim etapie dotyczącym obliczeń, wyliczono średnią wartość R/S z przedziałów o tej samej długości *n*. Otrzymano tym samym końcową wartość $(\frac{R}{S})_n$, którą zapisano w tablicy wynikowej AVG (9).

10. Kroki od 2 do 9 powtórzono dla każdej kolejnej wartości *n*, z tablicy *L*. Ostatecznie otrzymano pełną tablicę wynikową AVG, o równej długości co tablica L i wiążącą je zależnością $AVG_j = C \cdot L_j^H \quad C = const$. W tabeli 3 przedstawiono końcowe średnie wartości R/S dla poszczególnych wielkości *n*:

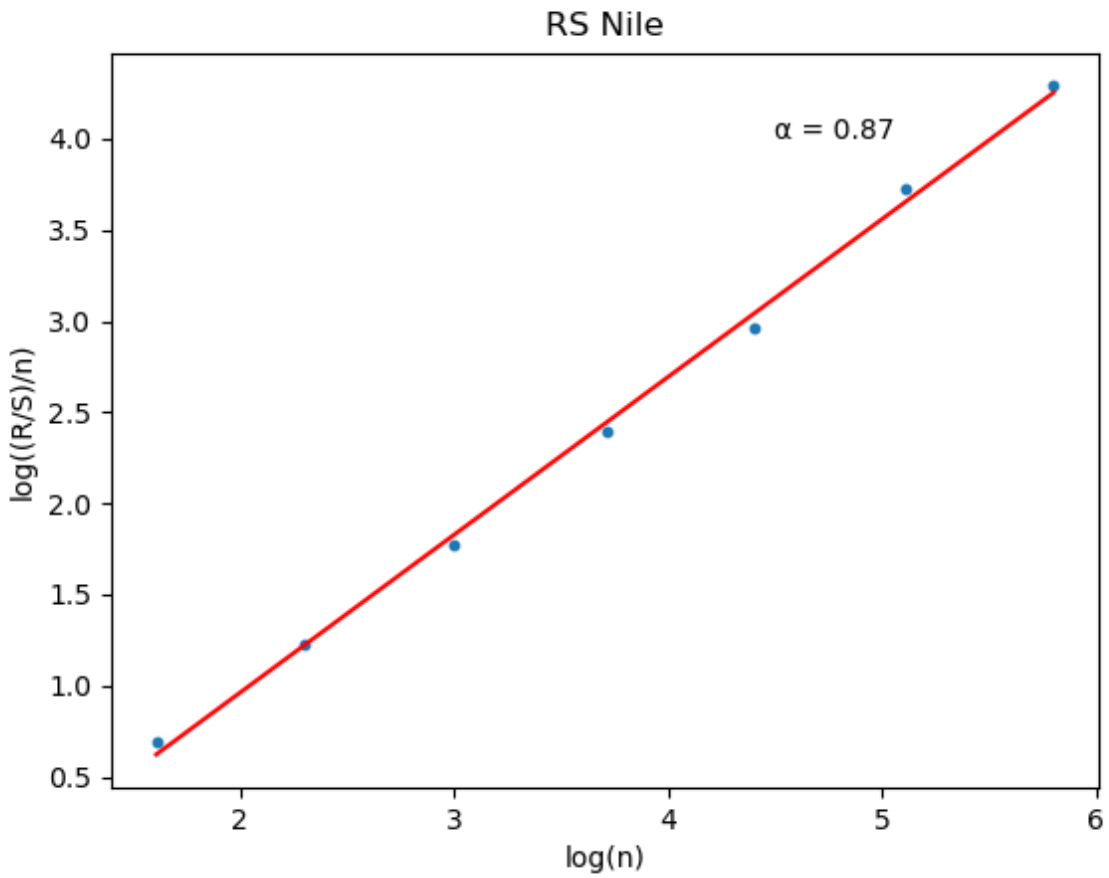
X		
j	L_j = n	$\frac{R/S}{n}$ = AVG_j *
0	331	72.89
1	165	41.37
2	82	19.25
3	41	11.02
4	20	5.92
5	10	3.41
6	5	1.99

Tabela 3: Zestawienie danych końcowych agorytmu, na podstawie których odczytujemy wniosek

W celu przedstawienia wyników, wygenerowano wykres podwójnie logarytmiczny (*Rys. 1*), gdzie na osi X znalazły się logarytmy długości segmentów – *n*, a na osi Y logarytmy otrzymanych średnich. Do układu dopasowano prostą metodą regresji liniowej. Poniżej przedstawiono fragment kodu odpowiadający za jego powstanie, przy użyciu biblioteki pyplot.

```
38    plt.scatter(np.log(L), np.log(AVG), s=10)
39    plt.title('RS Nile')
40    plt.ylabel('log((R/S)/n)')
41    plt.xlabel('log(n)')
42    result = np.polyfit(np.log(L), np.log(AVG), 1)
43
44    plt.text(4.5, 4, '\u03B1 = {}'.format(round(result[0], 2)))
45    x1 = np.log(L[0])
46    x2 = np.log(L[-1])
47    plt.plot([np.log(L[0]), np.log(L[-1])], [result[0] * x1 + result[1], result[0] * x2 + result[1]], 'red')
48    plt.show()
```

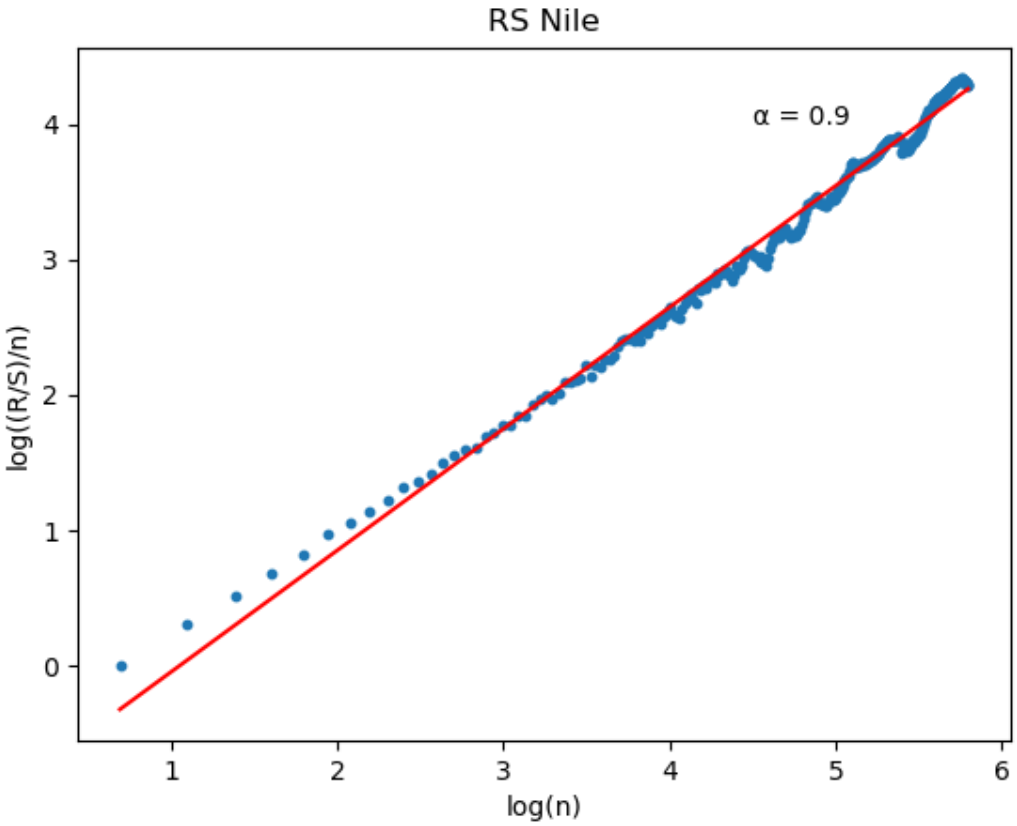
Implementacja generowania wizualizacji wyniku algorytmy RS



Rys. 1 Zobrazowanie danych wynikowych nilu, dla wyżej określonych długości serii danych

Współczynnik kierunkowy α prostej, jest poszukiwanym wykładnikiem H. Dla analizowanych danych dot. Nilu $\alpha = H \approx 0.87$ [*]. Wartość ta należy do zakresu $[\frac{1}{2}, 1]$, co pozwala na wyciągnięcie wniosku, że analizowany szereg czasowy jest silnie skorelowany. Oznacza to, że trend danych występujący dla obserwowanego zjawiska, z dużym prawdopodobieństwem zostanie zachowany.

Aby ocenić wiarygodności metody R/S, wykonano obliczenia dla 329 różnych przedziałów wielkości n, mieszczących się w szeregu Nile, którego długość wynosi $N = 662$. Otrzymane wyniki są zgodne z poprzednimi i układają się, z lekkimi odchyleniami, w linii prostej. Wartość estymowanego wykładnika zbliżyła się do 0.9, wskazując na jeszcze silniejszą korelację wartości szeregu. Poniżej zamieszczono wykres (*Rys. 2*), gdzie do obliczeń wybrano przedziały o kolejnych długościach $n \in [2, 331]$ i $n \in \mathbb{Z}$.



Rys. 2 Zobrazowanie danych wynikowych nilu dla 329 badanych serii

2.3. Algorytm DFA (Detrended Fluctuation Analysis)

Algorytm DFA, zgodnie ze swoją nazwą, jest metodą analizy fluktuacji wokół lokalnego trendu danych. Był szeroko wykorzystywany od połowy lat 90-tych XX wieku, do analizy szeregów czasowych pod kątem ich persystentności, czyli posiadania cechy określanej mianem *długiej pamięci*. Jego kluczowymi operacjami są:

- ➔ modyfikacja danych wejściowych do nowego szeregu, przy użyciu błądzenia losowego (random walk), opisanego w rozdziale 1.2.3;
- ➔ wyznaczenie lokalnych trendów w badanych przedziałach o wielkości *n* i na ich podstawie obliczenie wartości fluktuacji, których średnia jest podstawą to estymacji wykładnika H.

Ponieważ w niniejszej pracy, do każdej z metod wykorzystywany jest ten sam sposób odczytu wykładnika H, opierający się na zależności $F(L) \sim L^{\alpha}$ gdzie $\alpha = H$, dla omawianego algorytmu DFA, przedstawić można ją równaniem:

$$\widetilde{F}(n) = C \cdot n^{H} \qquad C = const$$

gdzie:

- ➔ ***n*** – długość badanych przedziałów, należących do analizowanego szeregu czasowego;
- ➔ $\widetilde{F}(n)$ – średnia wartość fluktuacji, wyprowadzonych dla wszystkich rozłącznych przedziałów o wielkości *n*, należących do badanego szeregu czasowego.

1. Dane wejściowe do algorytmu przygotowano podobnie jak w algorytmie R/S, korzystając z funkcji pomocniczej *prepareData (2.1.1)*. W rezultacie otrzymano trzy tablice:
   - ➔ **indexes –** w przeciwieństwie do metody R/S, metoda DFA nie pozwala na zignorowanie wartości porządkujących. Są ono bowiem istotne w procesie dopasowywania trendu na badanych przedziałach.
   - ➔ **D** – wartości pomiarów, pierwotnie zapisane w tablicy X (nazwa zmieniona ze względu na późniejszą kolizję oznaczeń);
   - ➔ **L** – tablica z wyselekcjonowanymi długościami przedziałów.

```
 8   def DFA():
 9       # 0
10       indexes, X, L = prepareData('nile.txt', None, 2)
11       N = len(X)
12
13       # 1 średnia wszystkich danych
14       avg = np.average(X)
```

Implementacja przygotowania danych do przetwarzania

W celu zobrazowania sposobu działania algorytmu DFA, wykorzystano te same dane co w rozdziale poświęconym metodzie R/S.

Pierwszą operacją, jaką należało wykonać, było wcześniej wspomniane przekształcenie pierwotnego szeregu danych, przy pomocy błądzenia losowego. W tym celu obliczono średnią z wartości analizowanego szeregu czasowego (tablica D) (1). Dla przyjętego przykładu średnia wynosiła $avg = 1148.20$ [*].

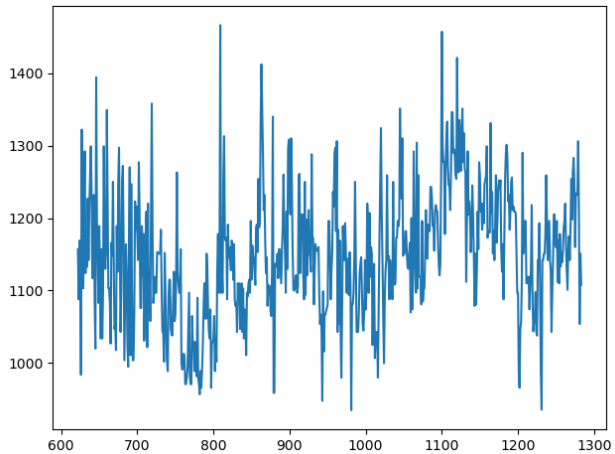
2. Korzystając z wyliczonej w kroku 1 średniej arytmetycznej całego szeregu, dokonano jego transformacji. W tym celu zastosowano prosty wzór (2), według którego każdą wartość szeregu czasowego, zamienienia się na sumę poprzedzających ją pomiarów (z wyznaczanym elementem włącznie), pomniejszonych o wyliczoną wcześniej średnią szeregu.

$$D_n = \sum_{k=1}^{n} \left(d_k - \langle d_n \rangle \right) \quad \leftarrow \quad \langle d_n \rangle = avg \quad (2)$$

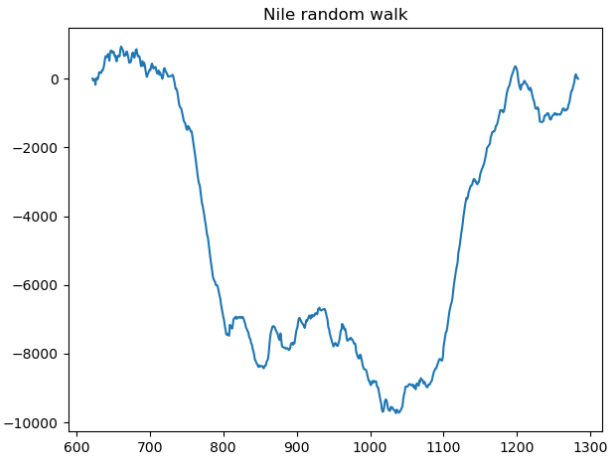
```
16      # 2 zmiana danych na random walk
17      randomWalk = []
18      cumulative_sum = 0.0
19      for i in range(0, N):
20          cumulative_sum += array[i] - avg
21          randomWalk.append(cumulative_sum)
```

Implementacja zamiany danych przy pomocy metody Random Walk

Poniżej przedstawiono wykresy z danymi oryginalnymi (*D*, *Rys. 3*) oraz przekształconymi za pomocą błądzenia losowego (*Rys. 4*), które wykorzystano do dalszych obliczeń.



Rys. 3 Dane oryginalne - D



Rys. 4 Dane zmodyfikowane - randomWalk

Przyglądając się zobrazowanym danym (*Rys 4*), można dostrzec jak przedstawiać będzie się trend, w dowolnie dobranych przedziałach czasowych.

3. Dysponując tak przygotowanymi danymi, przystąpiono do wykonania obliczeń zgodnie z kolejnymi krokami algorytmu DFA, zaprezentowanymi poniżej.

```
23      # petla do wybierania długości segmentów
24      F_avg = []
25      for segment_size in L:
26          # 3
27          temp_array = randomWalk.copy()
28          X = indexes.copy()
29          i = 0
30          k = indexes[0]
31          F = []
32          while i <= N - segment_size:
33              # znalezienie prostej w segmencie: line[0]=a; line[1]=b;
34              line = np.polyfit(X[0:segment_size], temp_array[0:segment_size], 1)
35
36              del temp_array[0:segment_size]
37              del X[0:segment_size]
38
39              # 4 wyliczenie F
40              F.append(calculateF(line, segment_size, i, k, randomWalk))
41              k = k + segment_size
42              i = i + segment_size
43
44          # 5 obliczenie sredniej fluktuacji dla danej dlugosci segmentu
45          F_avg.append(np.average(F))
46          print(segment_size, np.average(F))
```

Implementacja obliczeń algorytmu DFA

Ponieważ działanie algorytmu opera się na badaniu przedziałów, szereg czasowy podzielono na podzbiory o długości n, gdzie $n \in L$. Podobnie jak wcześniej, działanie algorytmu zilustrowano dla segmentów o wymiarze $n = 82$.

Na tym etapie algorytmu, dokonano przemianowania zbiorów danych, dla lepszego zobrazowania ich znaczenia w układzie kartezjańskim. Tablica z analizowanymi wartościami jest od teraz tablicą Y, a tablica z wartościami porządkującymi jest tablicą X.

4. W każdym przedziale o długości n określono lokalny trend. W tym celu do każdego z nich dopasowano prostą (4), przy pomocy regresji liniowej.

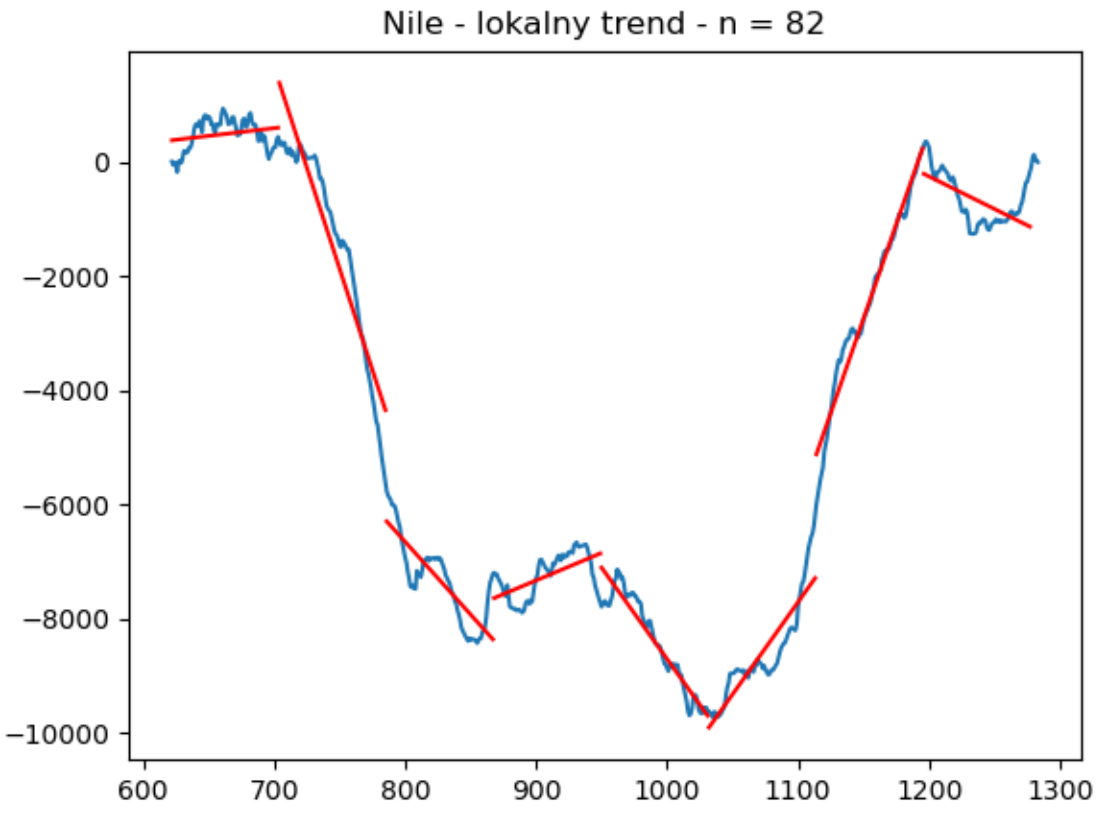
$$Y_i = a \cdot i + b \quad \leftarrow \quad i \in X \quad (4)$$

Po uzyskaniu współrzędnych prostej reprezentującej trend, wykorzystano je w następnym kroku (5), do obliczenia jego fluktuacji.

Powtarzając krok 4 otrzymano zestawienie współrzędnych prostych (*Tabela 4*), obrazujących trend lokalny wszystkich segmentów o długości n. Układ zilustrowano na wykresie (*Rys. 5*).

n = 82					
id	Badany przedział	Zakres indeksów badanego przedziału [X]	Wartości skrajne przedziału [Y] *	a *	b *
0	0 - 81	622 - 703	8.80, 444.40	2.72	-1311.41
1	82 - 163	704 - 785	372.20, -5641.20)	-70.82	51250.62
2	164 - 245	786 - 867	-5767.40, -7213.79	-25.50	13750.07
3	246 - 327	868 - 949	-7196.00, -7724.79	9.64	-1.60
4	328 - 409	950 - 1031	-7791.59, -9620.00	-32.05	23341.44
5	410 - 491	1032 - 1113	-9626.19, -6188.60	3.23	-4.32
6	492 - 573	1114 - 1195	-5990.79, -237.81	6.61	-7.88
7	574 - 655	1196 - 1277	298.61, -166.78	-1.14	1.34

Tabela 4: Wartości prostych określających trend lokalny w przedziałach o wielkości 82 danych



Rys. 5 Zobrazowanie lokalnych trendów w badanych przedziałach

5. Wartości fluktuacji trendu w poszczególnych przedziałach obliczono według poniższego wzoru:

$$F(n)=\sqrt{\frac{1}{n}\sum_{i=i_0;\,j=k}^{i_0+n-1;\,j=k+n-1}\left(RW_j-i\cdot a-b\right)^2} \quad (8)$$

$$i_0=X_0; \quad k=id\cdot n; \quad RW-RandomWalk$$

W analizowanym przykładnie Nilu, dla pierwszego segmentu składającego się z 82 rekordów, powyższe równanie (8), przyjmuje następującą postać liczbową:

$$F(82)=\sqrt{\frac{1}{82}\sum_{i=622;\,j=0}^{i=703;\,j=81}\left(RW_j-i\cdot 2.72+1211.41\right)^2}$$

Obliczenia wartości fluktuacji *F* wykonano wykorzystując zaimplementowaną funkcję pomocniczą *calculateF (8)*, która zwraca wartość fluktuacji dla podanego przedziału. Funkcja przyjmuje 5 argumentów:

- ➔ **line** – tablica ze współrzędnymi *a* i *b*, które otrzymano w kroku 4, przy użyciu funkcji *polyfit*;
- ➔ **n** – wielkość badanego przedziału;
- ➔ **i** – określa początkową wartość *x*, w równaniu prostej (4), dla badanego przedziału danych;
- ➔ **k** – określa indeks początkowy, od którego należy zacząć pobieranie danych z tablicy RW;
- ➔ **RW** – zmodyfikowany, przez błądzenie losowe, szereg wartości początkowych .

```
77  # 8
78  def calculateF(line, n, i, k, RW):
79      segment_sum = 0
80      for n in range(i, i + n):
81          segment_sum += (RW[n] - (line[0] * k) - line[1]) * (RW[n] - (line[0] * k) - line[1])
82          k = k + 1
83
84      F = np.sqrt((1 / n) * segment_sum)
85      return F
```

Implementacja funkcji pomocniczej wyliczającej wartość fluktuacji

Wartości fluktuacji w poszczególnych segmentach o wielkości *n* zapisano w tablicy *F* (5).

n = 82					
id	a *	b *	i_0	k	Wartość fluktuacji – F_n[id] *
0	2.72	-1311.41	622	0	270.09
1	-70.82	51250.62	704	82	587.09
2	-25.50	13750.07	786	164	403.90
3	9.64	-1.60	868	246	313.87
4	-32.05	23341.44	950	328	249.05
5	3.23	-4.32	1032	410	380.49
6	6.61	-7.88	1114	492	290.24
7	-1.14	1.34	1196	574	364.27

Tabela 5: Wartości fluktuacji trendu na przedziałach o wielkości n = 82

6. Po wykonaniu obliczeń dla wszystkich segmentów analizowanej serii podziałów o wielkości *n* i uzyskaniu pełnej tablicy F, wyliczono na jej podstawie średnią fluktuację.

```
57        # 6 obliczenie sredniej fluktuacji dla danej dlugosci segmentu
58        F_avg.append(np.average(F))
```

Wyliczenie średniej fluktuacji dla podziałów o wielkości n

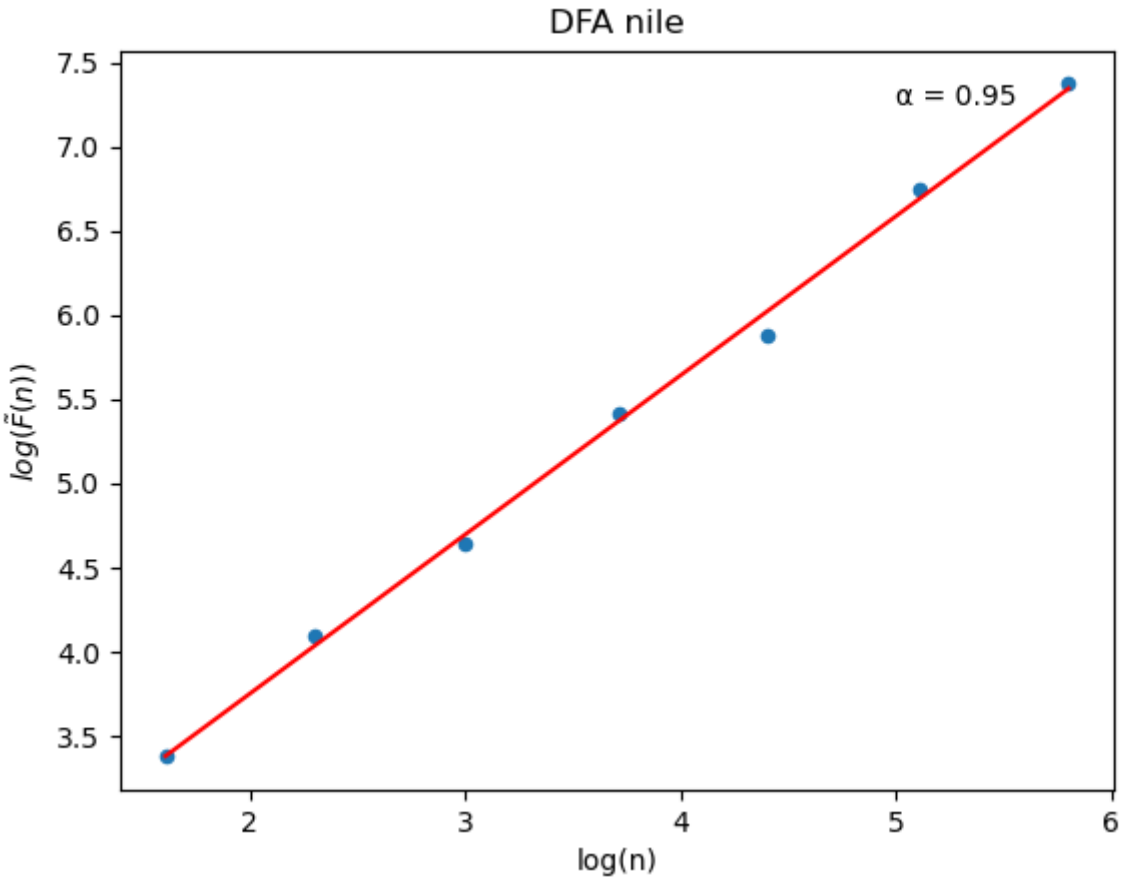
Otrzymane wartości dodano do tablicy *F_avg* (6), w której zebrano wszystkie średnie wartości fluktuacji przedziałów o wielkości *n*.

7. Kroki od 3 do 6 powtórzono dla każdej wielkości $n \in L$.

8. Po wykonaniu algorytmu dla każdej wartości z tablicy *L*, wyniki zebrano w t*abeli 5 i* zobrazowano na wykresie podwójnie logarytmicznym (*Rys. 6*).

D		
j	$n = L_j$	$\widetilde{F}(n) = F\_avg_j^*$
0	331	1592.46
1	165	854.18
2	82	357.37
3	41	226.09
4	20	104.35
5	10	59.97
6	5	29.31

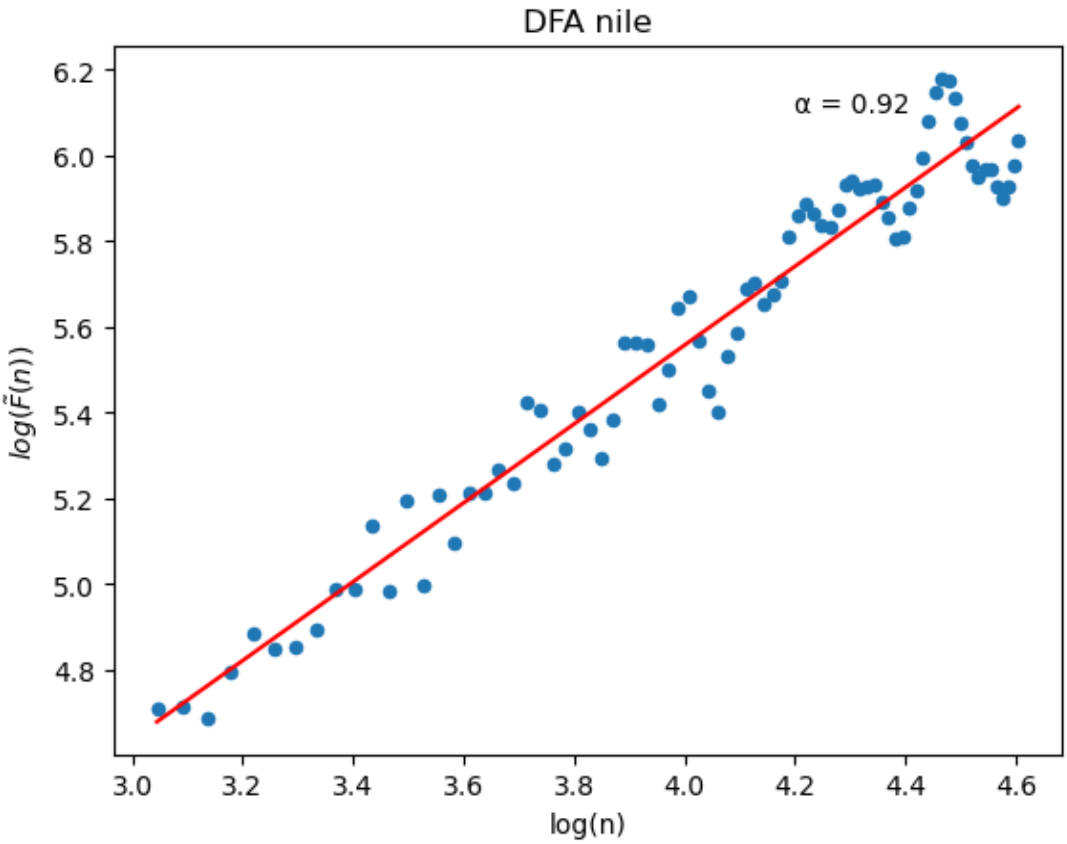
Tabela 6: Zestawienie zależności średniej fluktuacji przedziałów od ich długości n



Rys. 6 Wynik algorytmy DFA dla danych Nilu, przy określonych wyżej przedziałach czasowych

Jak zostało wcześniej wyjaśnione, współczynnik kierunkowy zilustrowanej prostej $\alpha = H \approx 0.95$. Jest to wynik zgodny z rezultatem jaki uzyskano w metodzie R/S. Jego wartość mieszcząca się w zakresie $[\frac{1}{2}, 1]$ świadczy o zachodzącej silnej korelacji w szeregu czasowym, opisującym badania wylewów rzeki Nil.

Przeprowadzono podobne doświadczenie jak w metodzie R/S i wykonano dodatkową analizę szeregu algorytmem DFA, dla 79 kolejnych wielkości przedziałów czasowych $n \in [21, 100]$ i $n \in \mathbb{Z}$ (*Rys. 7*). Zaobserwowano bardziej chaotyczny rozkład danych, niż w przypadku R/S, jednak konsekwentnie oscylujący przy prostej wyznaczonej regresja liniową. Wyciągnięto z tej obserwacji wniosek, że z wysokim prawdopodobieństwem, trend analizowanego szeregu zostanie w przyszłości zachowany, a do zmiany będzie wymagał silnego bodźca zewnętrznego.



Rys 7. Rezultat algorytmu DFA dla danych nilu, przy segmentach wielkości od 21 do 100

2.4. Algorytm DMA (Detrended Moving Average)

Algorytm DMA jest najprostszym do implementacji. Mimo że dąży do znalezienia takiej samej zależności potęgowej, stosuje zupełnie inną zasadę wyboru przedziałów. Badany szereg czasowy nie jest dzielony na $\lfloor \frac{N}{n} \rfloor$ rozłącznych przedziałów o długości *n*, tylko na $N-n$ przedziałów o długości *n*. Przy pomocy średniej arytmetycznej z każdego takiego przedziału, wyznaczana jest nowa wartość jego ostatniego elementu. Takiej operacji nadana jest nazwa średniej kroczącej. Odpowiada ona modyfikacji szeregu przez błądzenie losowe w algorytmie DFA.

Niektóre metody, tak jak ta, wymagają przemyślanego dopasowania okresów obserwacji, które mogą zależeć od rodzaju czy ilości zgromadzonych danych. Zmieniając długości i ilość badanych przedziałów czasowych, można doprowadzić układ danych końcowych do najbardziej jednoznacznego rezultatu.

Do przeprowadzenia analizy szeregów tą metodą wykonano kroki:

1. Przygotowano dane z pliku nile.txt w podobny sposób jak w poprzednich metodach. Otrzymano zebrane obserwacje w tablicy **X**, oraz tablicę **L** ze specjalnie wybranymi wielkościami okienka badawczego, którym algorytm przemieszcza się po szeregu czasowym. Tym razem wartości te są następującymi po sobie kolejnymi wielkościami, czyli np. zbiór liczb całkowitych $n \in L = \{300, 301, ..., 400\}$.

```
8   def DMA():
9       # 0
10      indexes, X, L = prepareData('nile.txt', 20, 4)
11      N = len(X)
12
13      modifiedStandardDeviation = []
14      for n in L:
15          # 1 srednia ruchoma dla przedziałów dlugosci n
16          i = n-1
17          movingAverage = X.copy()
18          while i < N:
19              cumulative_sum = 0.0
20              for k in range(0, n):
21                  cumulative_sum += X[i-k]
22              movingAverage[i] = cumulative_sum/n
23              i += 1
24
25          #2
26          sum = 0.0
27          for i in range(n, N):
28              sum += (X[i-1] - movingAverage[i-1]) * (X[i-1] - movingAverage[i-1])
29          modifiedStandardDeviation.append(np.sqrt(sum / (N - n)))
```

Implementacja algorytmu DMA

2. Wybrano długość okna z tablicy L i obliczono średnią dla wszystkich segmentów o tej długości. Zaczęto od przedziału $[0,n-1]$ i każdy kolejny wybierano przesuwając się o jeden element. W taki sposób zbadano cały szereg i zebrano $N-n$ wartości średnich przedziałów należących do szeregu X.

   Pętlę taką wykonano stosując wzór, w którym elementom tablicy X, zaczynając od elementu $i=n-1$, przypisuje się $\widetilde{x}_i$ jako średnią z sumy wyrazu i-tego i n-1 poprzedzających go elementów.

$$\widetilde{x}(i)=\frac{1}{n}\sum_{k=0}^{n-1} X(i-k) \quad (1)$$

```
16          # 1 srednia ruchoma dla przedziałów dlugosci n
17          i = n-1
18          movingAverage = X.copy()
19          while i < N:
20              cumulative_sum = 0.0
21              for k in range(0, n):
22                  cumulative_sum += X[i-k]
23              movingAverage[i] = cumulative_sum/n
24              i += 1
```

Implementacja obliczania średniej ruchomej dla okna badawczego o wielkości n

Na przykład przy n=300, pierwszą wartość średniej ruchomej wyliczono dla X(299), sumując wartości od X(0) do X(299) i dzieląc tę sumę przez 300. Kolejno obliczono średnią ruchomą dla X(300) sumując wartości od X(1) do X(300) i dzieląc przez 300. Tak przesuwając się, dokonano wyliczenia ostatniej wartości średniej dla X(661), gdzie sumę elementów od X(360) do X(661) podzielono przez 300.

3. W kolejnym kroku wyznaczono *zmodyfikowane odchylenie standardowe*. Jest to połączenie modyfikacji danych wejściowych o wyliczoną w kroku 2 średnią oraz na ich podstawie obliczania odchylenia dla aktualnie rozpatrywanego n. Działanie to przedstawia wzór:

$$\sigma_{DMA}(n)=\sqrt{\frac{1}{N-n}\sum_{i=n}^{N}(x_i-\widetilde{x}_i)^2}$$

W wyrażeniu podpierwiastkowym, znajduje się suma, która odpowiada za modyfikację szeregu danych X, o wyliczoną średnią.

```
26          #2
27          sum = 0.0
28          for i in range(n, N):
29              sum += (X[i-1] - movingAverage[i-1]) * (X[i-1] - movingAverage[i-1])
30          modifiedStandardDeviation.append(np.sqrt(sum / (N - n)))
```

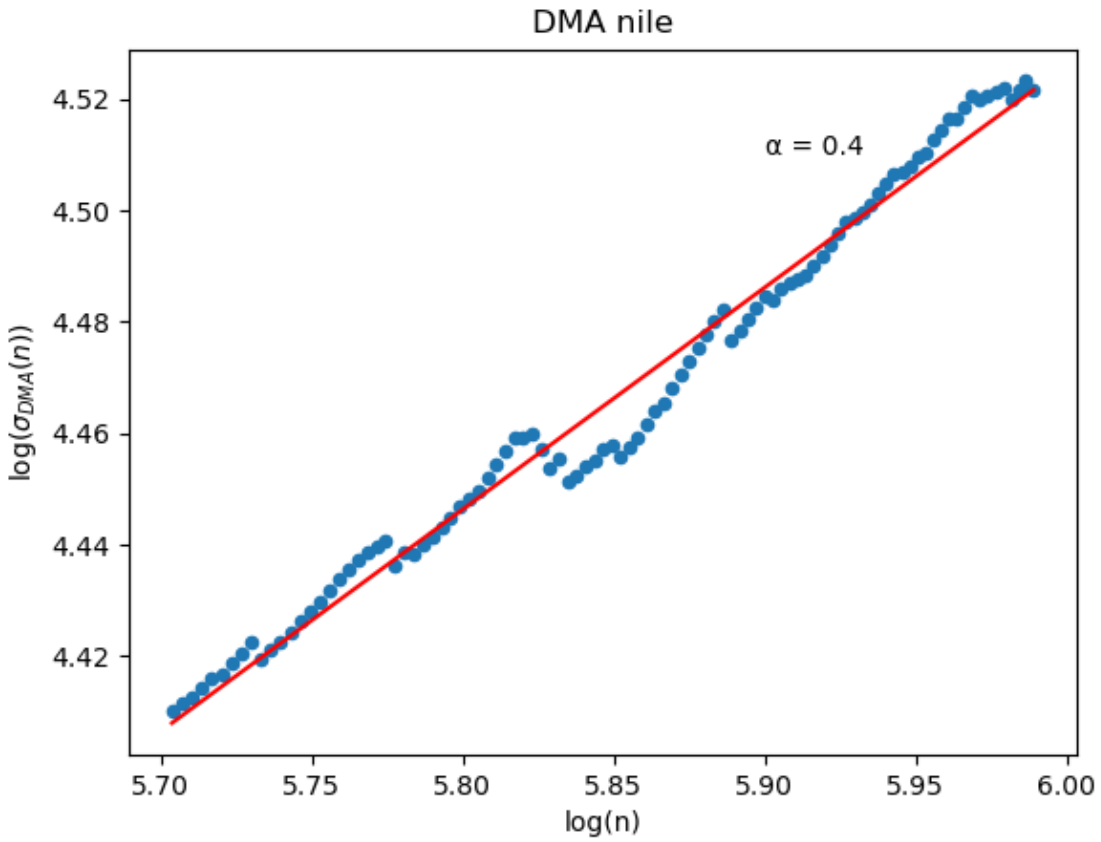
Implementacja wyprowadzania zmodyfikowanego odchylenia standardowego dla okna o wielkości n

4. Kroki 2 i 3 powtórzono dla kolejnych wielkości n z tablicy L. Po wykonaniu algorytm dla danych Nilu, przy *n* z zakresu [300, 400), otrzymano przedstawione niżej wyniki (*Tabela 7*).

$n \in L$	$\sigma(n)^*$
300	82.29
301	82.41
302	82.47
303 - 397	...
398	92.13
399	91.10

Tabela 7: Zestawienie zmodyfikowanego odchylenia standardowego dla okien badawczych o długości n

5. Na koniec przedstawiono dane na wykresie podwójnie logarytmicznym (*Rys. 8*).



Rys. 8 Wynik algorytmu DMA dla danych nilu przy n z zakresu 300-399

Wartości współrzędnej α istotnie się różni od wyników uzyskanych metodami R/S i DFA. Algorytm DMA w implementacji trywialny, w interpretacji już takim nie jest. Mimo, że jest to metoda oficjalnie uznawana za jedną z metod wyznaczania wykładnika Hursta, jest także tą najczęściej odrzucaną, ponieważ jej rezultat zazwyczaj jest bardzo odmienny od osiągniętych wyników przy pomocy innych algorytmów. Jak widać na omawianym przykładzie, z algorytmu DMA otrzymano wartości po przeciwnej stronie 0.5 niż z RS i DFA, gdzie wykładnik dążył do 1.

Kolejną jego wadą jest bardzo duża wrażliwość na dobór wielkości okna badawczego. Wybierając okna badawcze o różnych rozpiętościach i w różnej liczbie, można zbliżyć się do $\alpha=0$, zejść poniżej 0, a nawet osiągnąć wyniki $\alpha>4$ – gwałtownie wychodzący poza oczekiwany przez nas zakres.

W teorii bada się przedziały niewielkie, ale w dużych ilościach. Niestety w wielu przypadkach założenie to zawodzi. W związku z tym wątpliwa jest wiarygodność wyniku. Otrzymanie wykładnika H bliskiego 0, podczas gdy pozostałe metody zwracają korelację dodatnią, sugeruje odrzucenie algorytmu DMA jako metody rozstrzygania mocy korelacji w szeregu.

3. Analiza wyników dla różnych szeregów czasowych

W tym rozdziale omówiono otrzymane wyniki analiz dla wybranych szeregów czasowych, wymienionych w podrozdziale 1.4.2., przeprowadzonych z wykorzystaniem metod, którym poświęcona jest niniejsza praca. Zostały one zestawione z rezultatami algorytmów falkowych, o których szczegółowo napisał Pan Szymon Peszek w swojej pracy.

Dla metod R/S, DFA oraz wszystkich metod falkowych, obliczenia zostały wykonane dla rozłącznych przedziałów o wielkości *n*, implikując liczbę takich przedziałów, mieszczących się w badanym szeregu czasowym, równą $\lfloor \frac{N}{n} \rfloor$. W przypadku algorytmu DMA wartości te określono indywidualnie, dla każdego rozpatrywanego szeregu.

W poniższym zestawieniu (*Tabela 8*), zebrano wyniki dotyczące analizy korelacji danych dot. Nilu. Szereg czasowy omówiony podczas opracowywania działania poszczególnych metod.

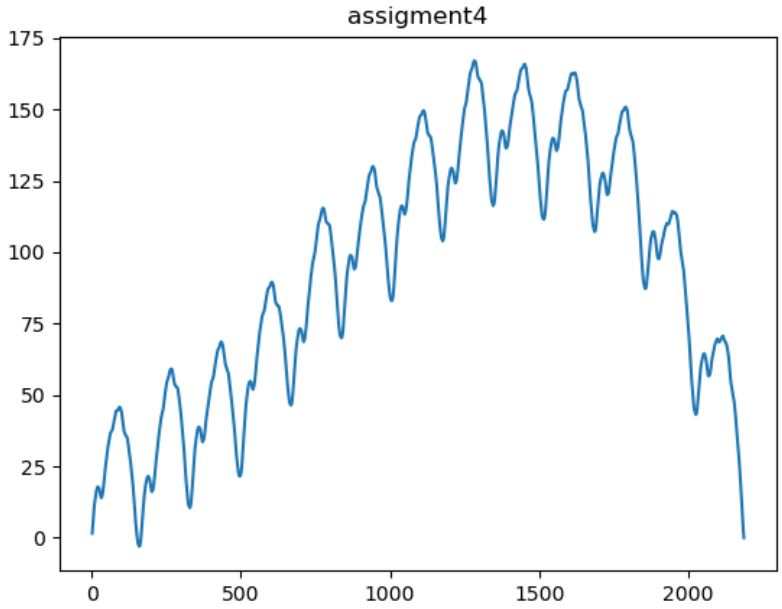
Metoda	Wykładnik H
R/S *	0.87
DFA *	0.95
DMA *	0.4
Falka Haar'a *	0.99
Falka Daubechie rzędu 4 *	0.99
Falka Coiflet'a rzędu 1 *	0.98
Falka Symlet rzędu 10 *	0.99

Tabela 8: nile.txt – zestawienie wyników dla różnych algorytmów

3.1 Szereg czasowy dla metody DFA (2087 pomiarów)

Szereg czasowy zapisany w pliku assigment4.txt został przygotowany specjalnie do zademonstrowania działania metody DFA. Dane wejściowe po przekształceniu błądzeniem losowym przedstawiono na wykresie (*Rys. 9*). Można zaobserwować, że zobrazowany proces charakteryzuje się zarówno trendem jak i cyklicznością.

Wykładnik Hursta wyliczono z zastosowaniem wszystkich trzech omawianych w pracy metod. Dla algorytmu DMA wielkość okien badawczych *n*, zawierała się w zakresie $[5, 20]$. W tabeli (*Tabela 9*) zaprezentowano uzyskane wyniki.

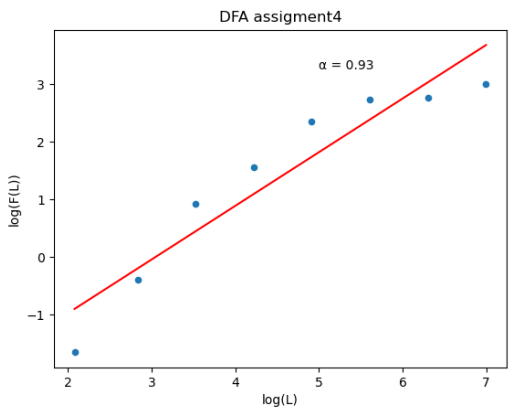


Rys. 9 Zmodyfikowany szereg czasowy - assigment4

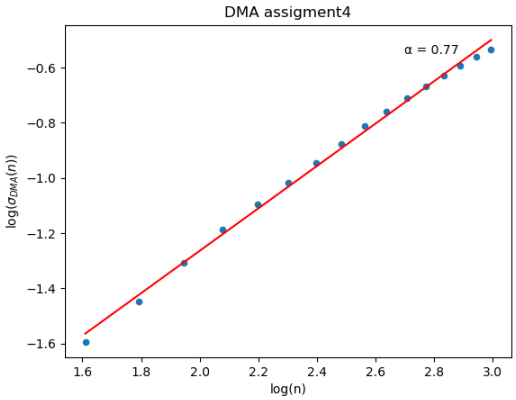
assigment4			
Metoda	DFA*	R/S*	DMA*
wykładnik H	0.93	0.96	0.77

Tabela 9: assigment4.txt – zestawienie wyników dla różnych algorytmów

Jak zaobserwowano wcześniej, analizując danych o Nilu, metody DFA i R/S zwracają w rezultacie bardzo zbliżony współczynnik Hursta. Natomiast algorytm DMA niezależnie od wybranych okien badawczych, generuje wynik różniący się od pozostałych metod. Jednak biorąc pod uwagę, że wszystkie otrzymane wykładniki znajdują się w tym samym zakresie (powyżej 0.5), można stwierdzić, że występujący trend w badanym szeregu czasowym zostanie zachowany. W przypadku metod DFA i R/S prawdopodobieństwo to jest silniejsze.



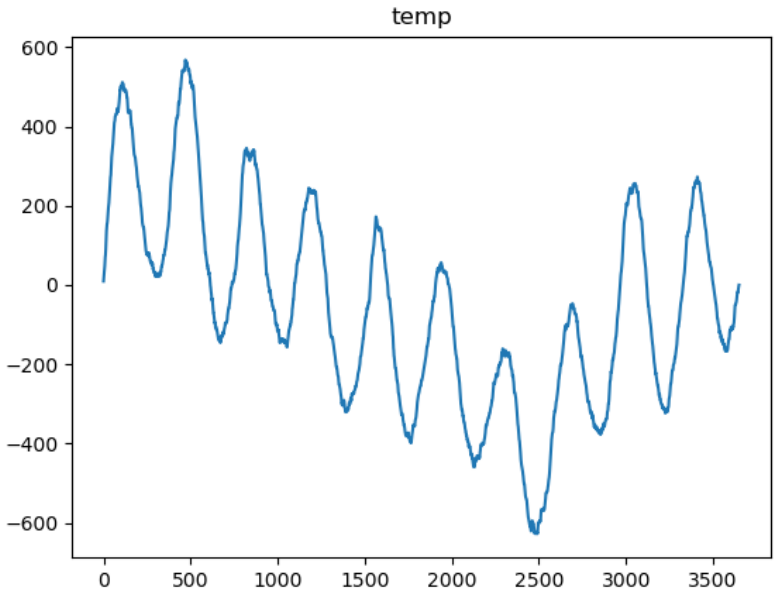
Rys. 10 assigment4.txt - algorytm DFA



Rys. 11 assigment4.txt - algorytm DMA

3.2 Analiza temperatury w Melbourne (3650 pomiarów)

Zawarty w pliku temp.txt szereg czasowy, zawiera pomiary minimalnej dziennej temperatury w Melbourne, Australia w latach 1981-1990. Dane po przekształceniu w szereg o charakterze błądzenia losowego, przedstawiono na wykresie (*Rys. 12*).



Rys. 12 Zmodyfikowane dane z pliki temp.txt

Wykładnik Hursta wyliczono wszystkimi analizowanymi metodami i porównano z dwoma algorytmami falkowymi. Dla metody DMA dobrano dwa przedziały badawcze, w celu zilustrowania wrażliwości tego algorytmu na zmianę tego parametru. Zestawienie wyników przedstawiono w tabeli poniżej:

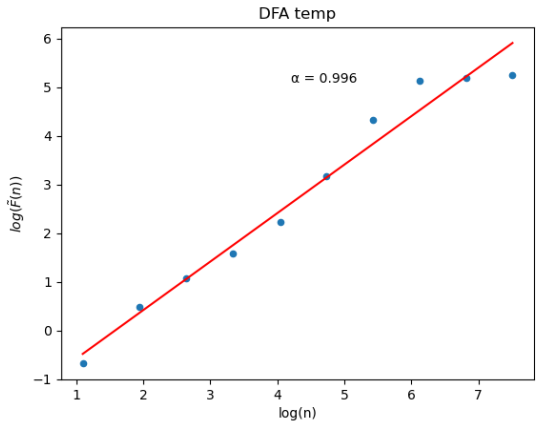
temp						
Metoda	R/S *	DFA *	DMA *		Falka	
			$n \in [5,20]$	$n \in [81,100]$	Haar'a *	Symlet rzędu 10 *
Wykładnik H	0.85	0.996	0.11	0.36	0.98	0.9993

Tabela 10: temp.txt – zestawienie wyników dla różnych algorytmów

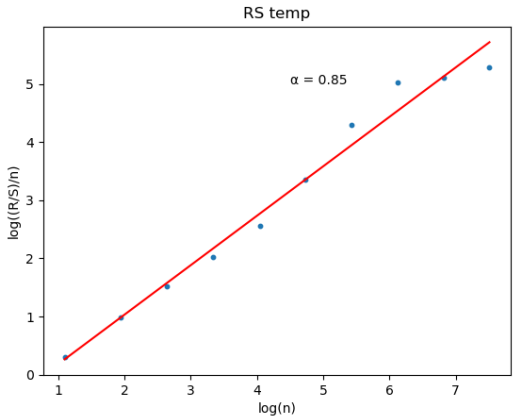
Wyniki uzyskane dla tego szeregu danych, pozwoliły na wyciągnięcie jednego wniosku na temat obserwowanego zjawiska oraz dwóch wniosków na temat metod.

W odniesieniu do użytych metod, DFA i falka Symlet rzędu 10 wyznaczyły współczynnik H bliski 1. Jest to bardzo silna korelacja, której wystąpienie jest czysto teoretyczne.

Z kolei algorytm DMA daje wyniki H znacznie poniżej 0.5 i jako jedyny wskazuje na korelacje negatywną. Nie ma możliwości dobrania zakresu okien badawczych, dla których wykładnik H byłby zbieżny z wartościami zwróconymi przez pozostałe metody. W związku z tym przy analizie tego szeregu czasowego, odrzucono metodę DMA.



Rys. 13 temp.txt - algorytm DFA



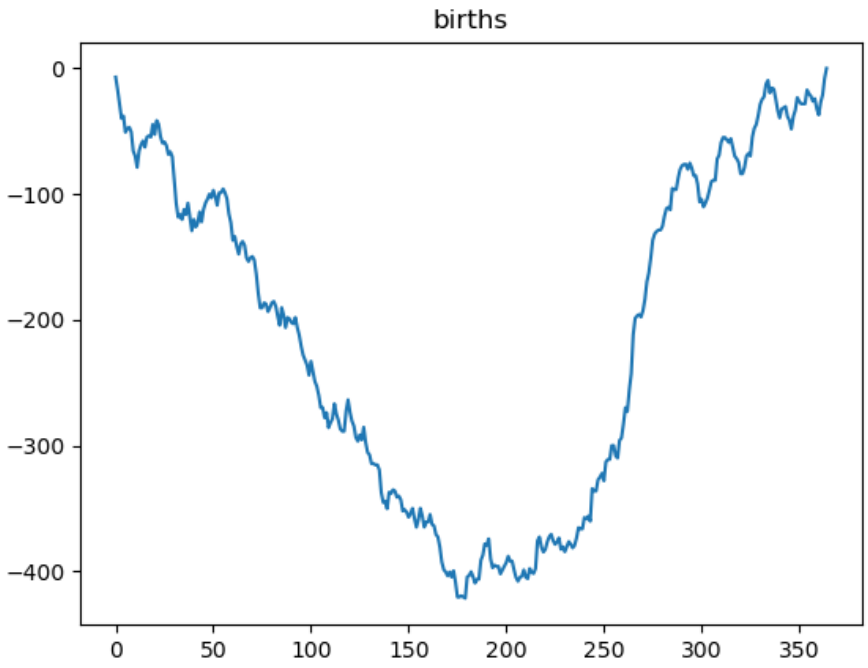
Rys. 14 temp.txt - algorytm R/S

Biorąc pod uwagę poczynione obserwacje na temat metod oraz wynik algorytmu R/S, wyciągnięto wniosek, że dane na temat temperatury w Melbourne są w silnej korelacji.

3.3 Analiza dziennej liczby urodzeń (356 pomiarów)

Plik births.txt zawiera obserwacje dziennej liczby urodzonych dziewczynek w Californii w 1959 roku. Dane po przekształceniu do postaci błądzenia losowego przedstawiono na wykresie (*Rys. 15*).

Analizę przeprowadzono metodami DFA, RS i algorytmami falkowymi, z powołaniem się w wynikach na falkę Symlet rzędu 10. Metodę DMA pominięto, ze względu na całkowitą rozbieżność z pozostałymi wynikami (otrzymano wartości zbliżone do 0), niezależnie od wybranych przedziałów. Wyniki zapisano w tabeli (*Tabela 11*).



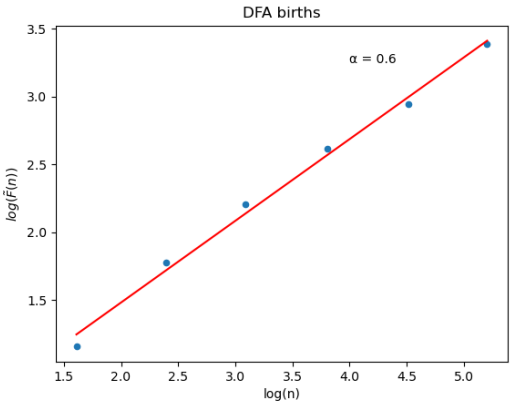
Rys. 15 Zmodyfikowane dane z pliku births.txt

births				
Metody	R/S *	DFA *	Falka Symlet *	DMA*
wykładnik H	0.62	0.60	0.95	0.11 – 0.39

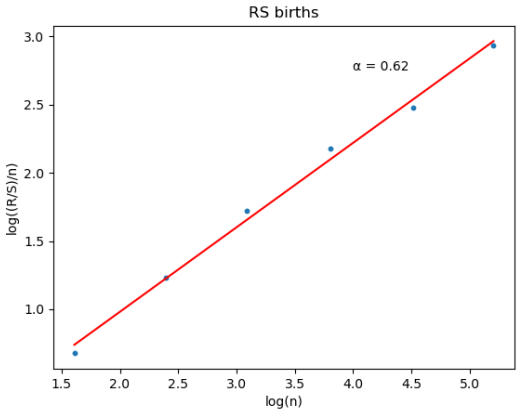
Tabela 11: births.txt – zestawienie wyników dla różnych algorytmów

Na podstawie przedstawionych na wykresie (*Rys. 15*) danych i zebranych w tabeli wyników metod, dokonano kilku spostrzeżeń. Po pierwsze, algorytmy falkowe zwróciły w rezultacie wykładnik H średnio rzędu 0.95., znacząco odbiegając od wyników uzyskanych metodami R/S i DFA. Wątpliwym wydaje się, żeby liczba urodzonych dziewczynek w czasie wyróżniała się tak niezmiennym trendem ($H \approx 0.95$).

Konkludując metody R/S i DFA są najbardziej wiarygodne w analizowanym wypadku. Dają one bowiem zgodny wynik oznaczający szereg słabo skorelowany, czyli o niskim prawdopodobieństwie, że jego trend się utrzyma. Wykładnik H dąży do $\frac{1}{2}$ co zbliża szereg do białym szumem pozbawionego trendu, co w badanym przypadku jak najbardziej może, a nawet powinno mieć miejsce.



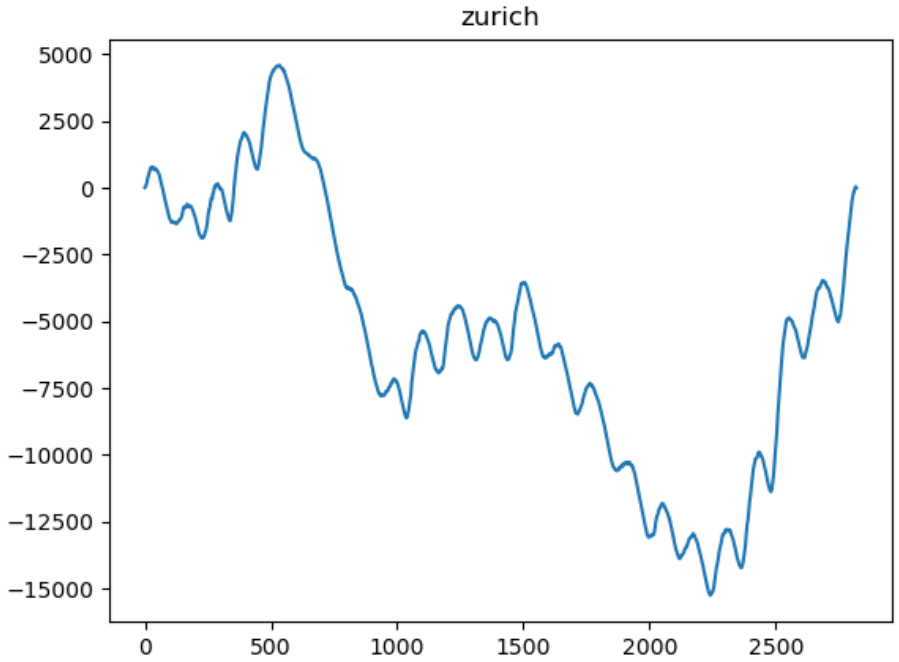
Rys. 16 births.txt - algorytm DFA



Rys. 17 births.txt - algorytm R/S

3.4 Analiza plam słonecznych (2820 pomiarów)

W pliku zurich,txt zapisano liczbę zaobserwowanych plam słonecznych w ciągu miesiąca, przez ponad 230 lat (1749-1983) w Zurychu. Zbiór został skompletowany w 1985 roku przez Andrew oraz Herzberg'a.



Rys. 18 Zmodyfikowane dane z pliku zurich.txt

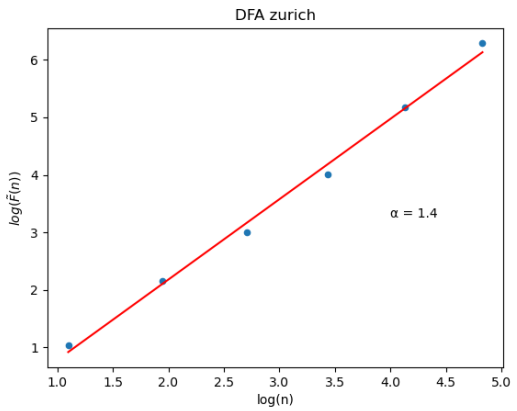
Analizę przedmiotowego szeregu czasowego przeprowadzono za pomocą algorytmów R/S, DFA, DMA i porównano z jednym z algorytmów falkowych – falką Haar'a. Metoda DMA została

przeprowadzona na przedziałach czasowych $n \in [11, 50]$ dla których to uzyskano najbardziej spójne z pozostałymi metodami. Rezultaty zebrano w tabeli poniżej (*Tabela 12*).

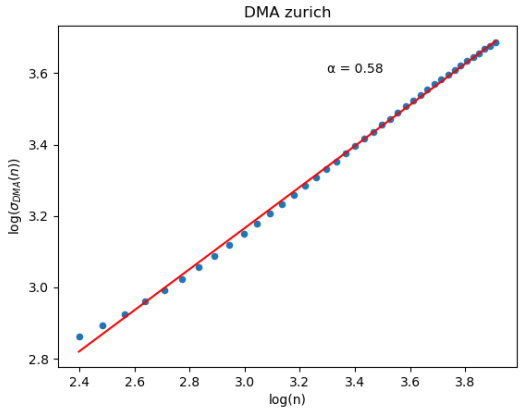
zurich				
Metoda	R/S *	DFA *	DMA *	Falka Haar'a *
Wykładnik H	0.89	1.4	0.58	0.91

Tabela 12: zurich.txt – zestawienie wyników dla różnych algorytmów

Uzyskane wyniki są bardzo silnie rozproszone – od współczynnika $H \approx 0.58$ (DMA), wskazującego na bardzo słabą persystentność badanego szeregu, aż do $H \approx 1.4$ (DFA) wykraczającego poza zakres wahań wskaźnika. Powyższe rezultaty prowadzą do wniosku, że analizowany szereg prawdopodobnie jest skorelowany, jednakże siła trendu jest trudna do oszacowania przy pomocy użytych metod.



Rys. 19 zurich.txt - algorytm DFA



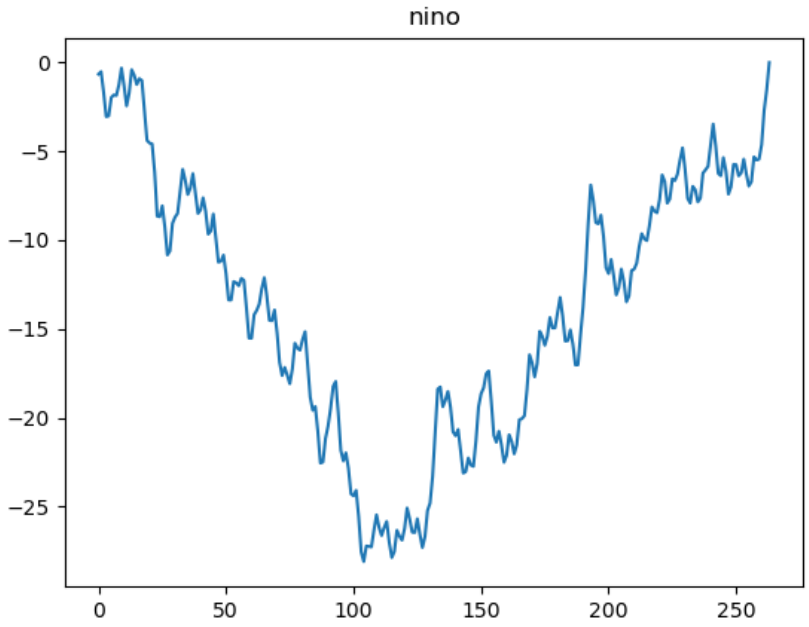
Rys. 20 zurich.txt - algorytm DMA

3.5 Analiza temperatury na powierzchni morza (263 pomiary)

Plik nino.txt zawiera pomiary temperatury powierzchni morza.

Jest to szereg o dosyć krótkim okresie obserwacji, dlatego dobrane okna badawcze dla metody DMA mają bardzo niskie wartości. Wybrano zakresy $n \in [5, 7]$ oraz $n \in [5, 8]$. Zakresy te różnią się jednym pomiarem i zostały przedstawione, w celu pokazania, jak może zmienić się wartość wykładnika H, w zależności od ilości badanych długości n.

Przy metodzie DFA także odrzucono jeden (*Rys. 23*) i dwa pomiary oraz przedstawiono różnicę w otrzymanych wynikach (*Tabela 13*).



Rys. 21 Zmodyfikowane dane z pliku nino.txt

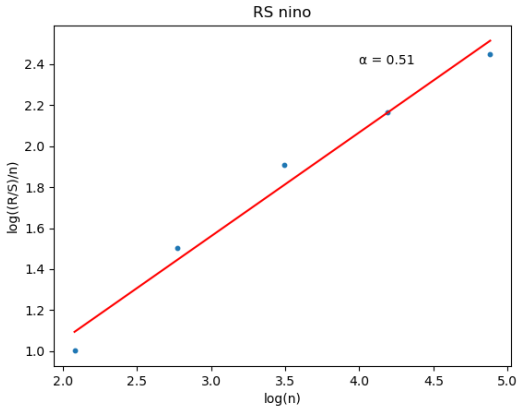
nino					
Metoda	R/S *	DFA *		DMA *	
		Odrzucono 2	Odrzucono 1	$n \in [5,7]$	$n \in [5,8]$
Wykładnik H	0.54	0.59	0.51	0.53	0.45

Tabela 13: nino.txt – zestawienie wyników dla różnych algorytmów

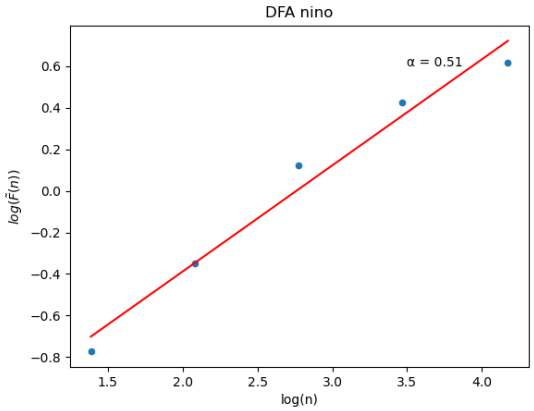
Wykładnik H dla metody R/S jest bliski wartości brzegowej 0.5, która jak wyjaśniono na wstępie, jest uznawana za teoretyczną. Siła korelacji oscylująca wokół 0.5, tak jak w badanym przypadku, może oznaczać szereg nieskorelowany.

W odniesieniu do metod DFA i DMA zaobserwowano, że różnica w liczbie wykonanych pomiarów, nawet jeśli jest to tylko jeden rząd wielkości, może dać widocznie odmienny wynik. Zauważono, że podczas zmniejszania liczby badanych przedziałów, siła korelacji szeregu rośnie. Dla obu metod, odrzucając zaledwie jeden pomiar, wartość wykładnika H wzrosła o 0.08.

Na podstawie opisanych obserwacji wysunięto wniosek, że dane szeregu są ze sobą bardzo słabo powiązane, a współczynnik korelacji waha się w przedziale od 0.4 do 0.6.



Rys. 22 nino.txt - algorytm RS



Rys. 23 nino.txt - algorytm DFA - odrzucony 1 pomiar

4. Podsumowanie

Wykładnik Hursta pozwala na określenie typu korelacji i jej mocy w danym szeregu czasowym. Znajomość tej własności daje możliwość przewidzenia, jak zachowają się wartości szeregu w przyszłości. Jest to wiedza wysoce pożądana nie tylko w kręgach finansowych, ale także bardzo przydatna przy przewidywaniu zmian klimatycznych, gdzie może pomóc przygotować się mieszkańcom danego regionu na, na przykład, powodzie.

Algorytmy, którym została poświęcona ta praca oraz dyskretna transformacja falkowa opisana przez Pana Szymona Peszka, są tylko jednymi z wielu metod obliczania wykładnika Hursta. Przeprowadzając nimi analizy różnych szeregów czasowych, dokonano wielu obserwacji.

Po pierwsze, analizując wykładnik Hursta, napotyka się duży problem, a mianowicie charakteryzuje się on dużą zmiennością, jeśli chodzi o metody, a badanie nimi serie. W związku z tym należy być szczególnie ostrożnym przy wyciąganiu jakichkolwiek wniosków.

Idący za tym wniosek drugi, o zależności między metodami. Jeśli algorytmy dają podobne, prawie równe wyniki dla danego szeregu czasowego, to nie można założyć, że będą ze sobą zgodne w każdym innym przypadku (przykłady 3.2 i 3.3).

Kolejna obserwacja dotyczy ilości badanych przedziałów w szeregu. Branie pod uwagę mniejszej ilości analizowanych podziałów, nawet o jedna wielkość *n*, może dać widocznie odmienny wynik. Często wykonuje się taki zabieg w algorytmach falkowych, ale także opiera się na nim metoda DMA. Zaobserwowano w przykładzie 3.5, że im więcej różnych przedziałów zostanie przebadanych, tym moc korelacji będzie mniejsza.

Wyprowadzając wykładnik Hursta, należy przygotować odpowiednie dane, spełniające warunek o równych odstępach czasu wykonywania pomiarów. Do każdego szeregu czasowego trzeba dobrać odpowiednią dla niego grupę algorytmów. Interpretując uzyskane nimi wyniki,

zwraca się uwagę na wartość współczynnika korelacji oraz na różnice między wynikami dla poszczególnych metod.

Niestety wykładnik Hursta nigdy nie daje stuprocentowej dokładności w prognozowaniu dalszych pomiarów, ani nie tłumaczy skąd biorą się obserwowane zależności. Pozwala on jedynie na określenie prawdopodobieństwa, z jakim trend danego zjawiska się utrzyma. Być może w przyszłości ktoś odkryje algorytm umożliwiający pawdziwe *przewidywanie przyszłości*.

Przypisy

* – wszystkie podane wartości są w przybliżeniu do dwóch miejsc po przecinku.

Literatura

1. *Porównanie różnych algorytmów wyznaczania wykładnika Hursta, 2020, Szymon Peszek*
2. http://www.ptzp.org.pl/files/konferencje/kzz/artyk_pdf_2016/T1/t1_0455.pdf
3. http://www.actaphys.uj.edu.pl/fulltext?series=Reg&vol=36&page=2403
4. https://en.wikipedia.org/wiki/Hurst_exponent
5. http://th-www.if.uj.edu.pl/zfs/gora/timeseries18/lecture08.pdf
6. http://prac.im.pwr.edu.pl/~hugo/stronaHSC/Podstrony/ksiazki/lma/lma.pdf
7. https://arxiv.org/pdf/physics/0608313.pdf
8. https://quantdare.com/demystifying-the-hurst-exponent/
9. https://pogotowiestatystyczne.pl/slowniczek/korelacja/
10. http://obliczeniastatystyczne.pl/odchylenie-standardowe/
11. https://pl.wikipedia.org/wiki/Fluktuacja
12. https://pl.wikipedia.org/wiki/B%C5%82%C4%85dzenie_losowe
13. http://www.foton.if.uj.edu.pl/documents/12579485/e3ed4beb-cb0f-450a-a1db-77cc97849204