НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

«На правах рукопису»
УДК 004.942

«До захисту допущено»
Завідувач кафедри
_____ О.Л. Тимощук
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 124 Системний аналіз

на тему: «Система прийняття рішень на основі вейвлетної ідентифікації хвиль Елліота»

Виконав:
Студент ІІ курсу, групи КА-62м
Слюсар Андрій Вячеславович _____

Керівник:
д.т.н., проф.
Данилов В.Я. _____

Рецензент:
д.т.н., проф.
Качинський А.Б. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ
2018

РЕФЕРАТ

Магістерська дисертація: 72 с., 49 рис., 25 табл., 26 джерел та 2 додатки.

Метою цієї роботи є побудова системи прийняття рішень для часових рядів валютних котирувань. У роботі досліджуються результати вейвлет-перетворень, показники Херста та фрактальної розмірності. На основі виявлених закономірностей побудовані індикатори та система прийняття рішень.

Об’єкт дослідження — часові ряди валютних котирувань на біржах.

Предметом дослідження є:

- неперервне вейвлет-перетворення хвиль Елліотта;
- система прийняття рішень на основі індикаторів технічного аналізу;
- показник Херста та фрактальної розмірності.

Результати роботи:

- продемонстровано підходи до аналізу часових рядів за допомогою неперервних вейвлет-перетворень;
- використано індикатори Херста та фрактальної розмірності для виділення ознак часового ряду;
- проаналізовано результати перетворень часових рядів на основі вже відомих вейвлетів;
- розроблено алгоритм рекомендації валютних операцій за допомогою індикаторів на основі вейвлет-перетворень;
- реалізовано систему прийняття рішень, що дозволяє отримати аналіз на основі реалізованих індикаторів та рекомендації щодо валютних операцій для заданого часового ряду.

Результати цієї роботи рекомендовано використовувати для аналізу часових рядів, історичних даних та для прийняття рішень відносно валютних операцій на валютній біржі.

ВЕЙВЛЕТ-АНАЛІЗ, ХВИЛІ ЕЛЛІОТТА, ВЕЙВЛЕТ-ПЕРЕТВОРЕННЯ, ПРИЙНЯТТЯ РІШЕНЬ, ФРАКТАЛЬНА РОЗМІРНІСТЬ.

ABSTRACT

Master's tesis: 72 p., 49 fig., 8 tabl., 26 sources and 2 appendices.

The purpose of this work is to build a decision-making system for time series of currency quotes. The paper investigated the results of wavelet transformations, Hurst and fractal dimension indicators. Based on the revealed patterns, indicators and a decision-making system are built.

Object of research — the time series of currency quotations on exchanges.

Subject of the study:

- continuous wavelet transform of Elliott waves;
- decision making system based on technical analysis indicators;
- Hurst index and fractal dimension.

The results of the work:

- demonstrated approaches to time series analysis using continuous wavelet transforms;
- used Hurst indicators and fractal dimension to distinguish time series patterns;
- analyzed the results of transformations of time series based on already known wavelets;
- developed an algorithm for the recommendation of currency transactions using indicators based on wavelet transforms;
- implemented a decision-making system that allows you to get an analysis based on realized indicators and recommendations for currency operations for a given time series.

The results of the work are recommended for analyzing time series, historical data and for making decisions regarding currency transactions on the currency exchange.

WAVELET ANALYSIS, ELLIOTT WAVES, WAVELET TRANSFORM, DECISION MAKING, FRACTAL DIMENSION.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

DSS - Decision Support System
СППР - Система підтримки прийняття рішень
ЕМА - експоненціальне ковзне середнє
SIG - сигнальна лінія індикатора
SMA - просте ковзне середнє
НЕЛ - найбільша експонента Ляпунова
ЕКС - експоненційне ковзне середнє
ДВП - Дискретне вейвлет-перетворення
НВП - Неперервне вейвлет-перетворення
ВЧ - Високі частоти
НЧ - Низькі частоти
FFT - Fast Fourier transform
DFT - Discrete Fourier transform
DOG - Derivative of Gaussian

ВСТУП

В наш час суттєві економічні зміни відбуваються чи не раз на декілька років і під них буває непросто підлаштуватися. В таких умовах дуже важливо навчитися їх передбачати. Аналіз часових рядів валютного котирування зараз набирає актуальності, оскільки відкриває можливості для заробітку на основі купівлі-продажу валюти та наукової діяльності в економічній сфері.

Наразі для аналізу валютного котирування використовується близько двохсот індикаторів і їх кількість постійно збільшується. Така тенденція свідчить про те, що в цій області постійно ведуться дослідження і вона ще повністю не вивчена. Тому кожний новий підхід до аналізу валютного котирування покращує розуміння прихованих процесів, які впливають на курс валютних пар та економіку в цілому.

Однією з найбільш ґрунтовних праць в області аналізу економічних тенденцій є так званий хвильовий аналіз Еліота. Хоча сам аналіз був придуманий ще в 30-х роках XX століття, він досі залишається популярним напрямком досліджень. Математичною основою теорії Еліота, за визнанням самого автора, стали так звані числа Фібоначчі — послідовність чисел, відкрита Фібоначчі.

Для моделей, виділених Еліотом, характерна повторюваність за формою, але не обов’язково за часом або амплітудою. Всього він виявив 13 подібних моделей (хвиль), що постійно виникають у даних про ринкові ціни. Еліот назвав, визначив і проілюстрував ці моделі. Описав, як пов’язані між собою, вони формують більш великі за розміром аналоги, які, в свою чергу, формують ті ж самі моделі ще більшого розміру і т. д. Еліот назвав це явище хвильовим принципом.

Суть хвильового аналізу полягає в тому, що рух ціни є закономірним і постійно повторюється одна і та ж модель цінового «шляху». Головна складність застосування хвильового аналізу полягає у правильному знаходженні хвилі, адже потрібно правильно відділити шуми та інші явища циклічного характеру від власне хвилі Еліота.

Оскільки ми маємо справу з хвильовим аналізом, тоді логічним кроком було б застосувати математичні інструменті, які працюють з хвилями. Аналіз хвиль виконувався за допомогою Фур'є і вейвлет перетворень.

Вейвлет-перетворення – це перетворення, що схоже на перетворення Фур'є. Основна відмінність лежить в наступному: перетворення Фур'є розкладає сигнал на складові у вигляді синусів і косинусів, тобто функцій, локалізованих в Фур'є-просторі, а вейвлет-перетворення використовує функції, локалізовані як в реальному, так і в в Фур'є-просторі.

Усі вейвлет-перетворення розглядають функцію (взяту як функцією від часу) у термінах коливань, локалізованих за часом (простором) і частотою. Для даного дослідження використовувались вейвлети Хаара, Добеші, Морлет та інші.

Для того щоб дана теорія була цікава для бізнесу та наукових досліджень процес виявлення хвиль Елліотта за допомогою вейвлет перетворень потрібно автоматизувати. На цій основі була побудована

Система підтримки прийняття рішень — комп'ютеризована система, яка шляхом збору та аналізу великої кількості інформації може впливати на процес прийняття управлінських рішень в бізнесі та підприємництві.

Сучасні системи підтримки прийняття рішень виникли у результаті злиття управлінських інформаційних систем і систем управління базами даних, як системи, що максимально пристосовані до розв'язування задач щоденної управлінської діяльності, і є інструментом, щоб надати допомогу тим, хто вирішує (робить вибір). За допомогою СППР може проводитись вибір рішень у певних неструктурованих і слабко структурованих задачах, у тому числі й тих, що мають багато критеріїв.

Таким чином, метою роботи є створення та аналіз індикаторів на основі вейвлет-перетворення та розробка системи прийняття рішень.

Для досягнення такої мети були вирішені наступні задачі:

а) створенно індикатори на основі спектрумів вейвлет-перетворень;
б) проведено аналіз створених індикаторів на виявлення хвиль Елліотта;
в) проведено прівняльних аналіз існуючих систем прийняття рішень для валютних корирувань на біржі;

г) розроблено систему прийняття рішень на основі отриманих результатів.

Об’єктом дослідження є фінансові часові ряди валютних котирувань.

Предметом дослідження є вейвлет-перетворення, модель хвиль Елліотта, система прийняття рішень.

В якості методів дослідження використовуються вейвлет-перетворення.

Наукова новизна роботи: запропоновано індикатори для валютних котирувань на основі вейвлет-перетворення та створено систему прийняття рішень на основі вейвлетної ідентифікації хиль Елліотта.

Практичними результатами роботи є розробка індикаторів на основі вейвлет перетворення та система прийняття рішень.

РОЗДІЛ 1 ТЕОРЕТИЧНА ЧАСТИНА

1.1 Аналіз актуальності задачі

Згідно теорії Елліота, ринок рухається циклічними хвилями з видом самоподібності всередині хвиль. Великі хвилі складаються з менших хвиль, які безпосередньо містять менші хвилі, і так далі. Таким чином данні хвилі описують деякі циклічні процеси, які впливають на ринок. Таким чином теорія Елліота дає можливість аналізувати ринок в контексті описаних ним хвиль – а саме циклічною закономірною поведінкою. Ці дослідження можуть перетинатися с масовою психологією людей, які впливають на ринок, на фізіологію людей, що проявляється як наслідок впливу циклів Землі, Сонця, Місяця і т.д.

Вейвлет-аналіз дає можливість детально дослідити властивості хвиль Елліота, а саме порівняти вейвлет-перетворення різних хвиль між собою та виявити закономірності. Проте аналіз спектрума вейвлет-перетворення не є тривіальною задачею, тому творення індикаторів полегшує розуміння результатів.

Наразі в технічному аналізі хвилі Елліота шукають вручну, а це означає, що одні люди можуть виокремити хвилю, а інші – ні. Цей факт завдає складнощів в аналізі ринку та інколи дискредитує хвилі Елліота.

Систем прийняття рішень для валютних котирувань багато, проте всі системи працюють з хвилями Елліотта в контексті чисел Фібоначчі. Створенная система - це новий підхід до прогнозування стратегії поведінки на біржі. Також система забеспечує аналітина новими індикаторами.

1.2 Дослідження існуючих підходів

На даний момент існує лише один підхід до виявлення та аналізу хвиль Елліотта - це аналіз рівнів на основі чисел Фібоначчі.

У своїй теорії Ральф Нельсон Елліотт виділив три основних аспекти в змінах цін: модель, співвідношення і час. Модель відноситься до хвильовим моделям і утворенням, в той час як співвідношення (взаємовідношення між числами, суто в рядах Фібоначчі) використовується для вимірювання хвиль. Для використання теорії в щоденній торгівлі трейдер визначає основну хвилю або надцикл, відкриває довгу позицію і потім продає, або відкриває коротку позицію, оскільки дія моделі припиняється або намічається розворот.

Числа Фібоначчі дозволяють математично обгрунтувати хвильову теорію Елліотта. Хоча співвідношення Фібоначчі застосовуються в різних технічних індикаторах, найбільш значна їх роль залишається в вимірі корекції хвиль. [3]

Послідовність Фібоначчі починається з простої одиниці і триває шляхом додавання попереднього числа ось до чого:

0 + 1 = 1, 1 + 1 = 2, 2 + 1 = 3, 3 + 2 = 5, 5 + 3 = 8, 8 + 5 = 13, 13 + 8 = 21, 21 + 13 = 34, 34+ 21 = 55, 55 + 34 = 89, ...

У цій серії багато характерних особливостей:

+ Коефіцієнт співвідношення сумарного числа до будь-якого числа в серії одно 0.618 або 61.8

+ Коефіцієнт співвідношення будь-якого сумарного числа до числа через одне становить 0.382 або 38.2Корекція - це зміна в цінах, яке простежує етапи попередніх змін. Зазвичай ринок відстежується в одному з трьох рівнів Фібоначчі - 38.2 %, 50 %, and 61.8 %. Корекція цін по Фібоначчі визначається основними мінімальними і максимальними коливаннями, які встановлюють рівень підтримки, як тільки ринок відступає від максимуму.

Корекція протікає також до максимального від мінімального коливання, використовуючи ті ж співвідношення, які розглядаються як рівні опору, як тільки ринок відскакує від мінімуму(рис. 1.1).

Пролонгація цін по Фібоначчі використовується трейдерами для визначення зон на графіку, при яких варто увійти на ринок під час наступного стійкого руху цін, будь-то висхідний або спадний тренд. Відсоткове продовження рівнів вибудовується на графіку в горизонтальні лінії над або під попереднім зміною тренду. Найбільш поширені рівні продовження - 61.8 %, 100.0 %, 138.2 % і 161,8 %.

Рисунок 1.1 – Приклад виявлення хвилі Елліотта на основі рівнів Фібоначчі

1.3 Формалізація постановки задачі

Метою даної роботи є дослідження існуючих методів знаходження хвиль Елліотта і розробка власного алгоритму для виявлення хвиль Елліотта за допомогою попередньої обробки рядів та вейвлет-перетворень.

У даному дослідженні виділяємо такі підзадачі:

- Розгляд існуючих рішень, що дозволяють виявити хвилю Елліотта
- Розробка алгоритму для виявлення хвиль Елліотта
- Дослідження фрактальної структури хвиль Елліотта
- Порівняльна характеристика фур’є- та вейвлет-перетворень для виявлення хвиль Елліотта

- Дослідження взаємозв’язку між історичними подіями та зародженням хвиль Елліотта

1.4 Хвильова теорія Елліотта

Хвильова теорія Еліота - один з найбільш відомих технічних методів аналізу та прогнозування фінансових ринків [1]. Викликаючи чималий інтерес з боку трейдерів, незважаючи через труднощі її інтерпретації, вона залишається одним з найбільш популярних методів аналізу в технічному аналізі.

1.4.1 Походження хвиль Елліотта

Теорія хвильового руху ринків була запропонована в 30-х роках XX століття Ральфом Нельсоном Еліотом, згідно з якою всі рух цін на ринку підпорядковується психології людей і є циклічним процесом зміни імпульсних хвиль на корекційні і навпаки. Згідно з Елліоттом, хвилі Елліотта є квантифікацією реакції натовпу на події, що не завжди можуть повторюватися в часі або амплітуді, але завжди повторюються структурно.

1.4.2 Типові хвилі Елліотта

Імпульсні хвилі являють собою послідовність п’яти коливань ціни, корекційні хвилі - послідовність трьох або п’яти коливань ціни. Імпульсні хвилі за формою, структурою, а також застосовним до них правилами бувають наступних типів:

Імпульси(рис. 1.2):

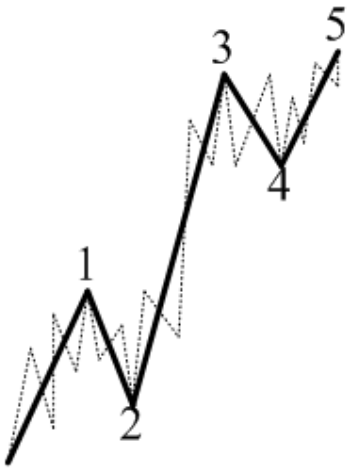


Рисунок 1.2 – Імпульс

- закінчення другої хвилі ніколи не заходить за початок першої хвилі;
- третя хвиля завжди простягається далі вершини першої хвилі;
- закінчення четвертої хвилі ніколи не заходить за вершину першої хвилі;
- третя хвиля ніколи не буває найкоротшою з усіх діючих хвиль;
- третя хвиля завжди є імпульсом;
- перша хвиля може бути або імпульсом, або клином;
- п’ята хвиля може бути або імпульсом, або діагоналлю;
- друга хвиля може прийняти форму будь корекційної хвилі за винятком трикутника;
- четверта хвиля може прийняти форму будь корекційної хвилі.

Клини(рис. 1.3):

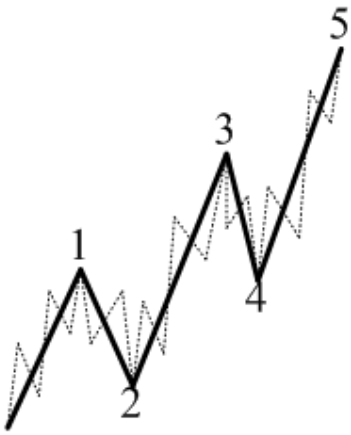


Рисунок 1.3 – Клин

- закінчення другої хвилі ніколи не заходить за початок першої хвилі;
- третя хвиля завжди простягається далі вершини першої хвилі;
- закінчення четвертої хвилі завжди заходить за вершину першої хвилі , але ніколи не заходить за початок третьої хвилі;
- третя хвиля ніколи не буває найкоротшою з усіх діючих хвиль;
- третя хвиля завжди є імпульсом;
- перша хвиля може бути або імпульсом , або клином;
- п’ята хвиля може бути або імпульсом чи діагоналлю;
- друга хвиля може прийняти форму будь корекційної хвилі за винятком трикутника;
- четверта хвиля може прийняти форму будь корекційної хвилі.

Діагоналі(рис. 1.4):

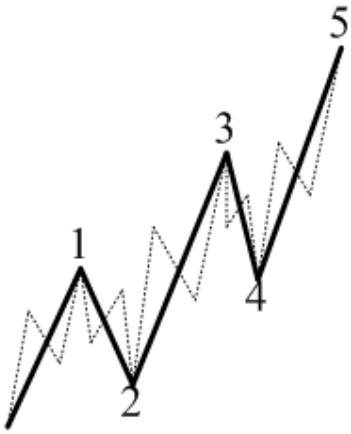


Рисунок 1.4 – Діагональ

- закінчення другої хвилі ніколи не заходить за початок першої хвилі;

- третя хвиля завжди простягається далі вершини першої хвилі;
- закінчення четвертої хвилі, як правило, заходить за вершину першої хвилі, але ніколи не заходить за початок третьої хвилі;
- третя хвиля ніколи не буває найкоротшою з усіх діючих хвиль;
- перша, друга і третя хвилі можуть прийняти форму будь корекційної хвилі за винятком трикутника;
- четверта і п’ята хвиля можуть прийняти форму будь корекційний хвилі. Коррекційні хвилі розподіляють на: Зигзаги(рис. 1.5):

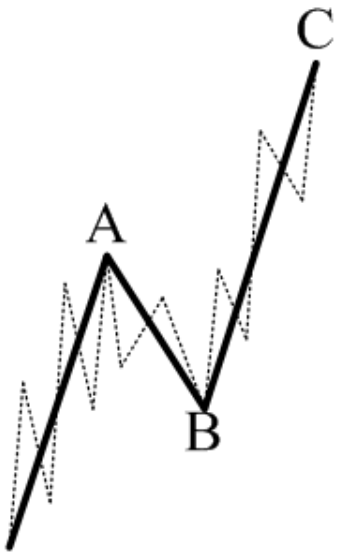


Рисунок 1.5 – Зигзаг

- хвиля А може прийняти форму імпульсу або клина;
- хвиля С може прийняти форму імпульсу або діагоналі;
- хвиля В може прийняти форму будь корекційної хвилі.

Хвиля С простягається далі вершини хвилі А; Закінчення хвилі В не заходить за початок хвилі А; Площини(рис. 1.6):

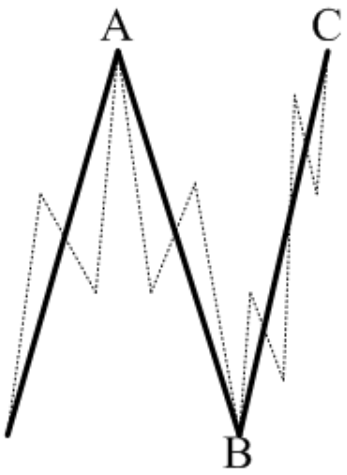


Рисунок 1.6 – Площина

- хвиля A може прийняти форму будь корекційної хвилі за винятком трикутника;
- хвиля B може прийняти форму будь корекційної хвилі;
- хвиля C може прийняти форму імпульсу або діагоналі.

Подвійні зигзаги(рис. 1.7):

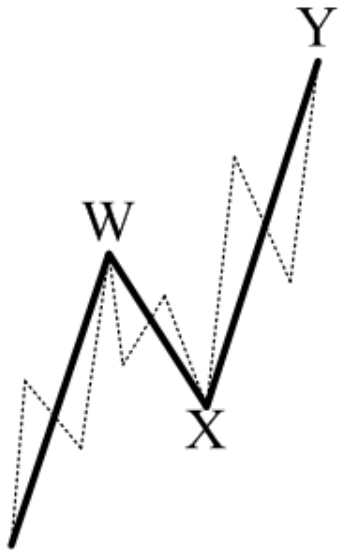


Рисунок 1.7 – Подвійний зигзаг

- хвиля W і хвиля Y приймають форму зигзага;
- хвиля X може прийняти форму будь корекційної хвилі;
- хвиля Y простягається далі вершини хвилі W;
- закінчення хвилі X не заходить за початок хвилі W.

Потрійний зигзаг(рис. 1.8):

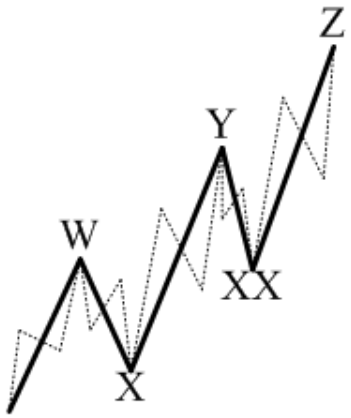


Рисунок 1.8 – Потрійний зигзаг

- хвиля W, хвиля Y і хвиля Z приймають форму зигзага;
- хвиля X може прийняти формул будь корекційної хвилі за винятком трикутника;
- хвиля XX може прийняти форму будь корекційної хвилі;
- хвиля Y простягається далі вершини хвилі W;
- хвиля Z простягається далі вершини хвилі Y;
- закінчення хвилі X не заходить за початок хвилі W;
- закінчення хвилі XX не заходить за початок хвилі Y.

Подвійні трійки(рис. 1.9):

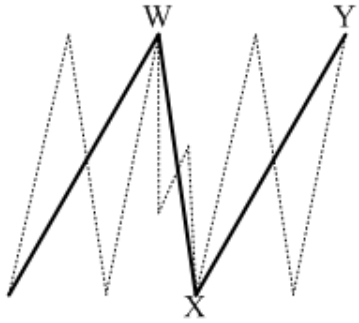


Рисунок 1.9 – Подвійна трійка

- хвиля W приймає форму будь корекційної хвилі за винятком трикутника;
- хвиля X і хвиля Y приймають форму будь корекційної хвилі.

Потрійні трійки(рис. 1.10):

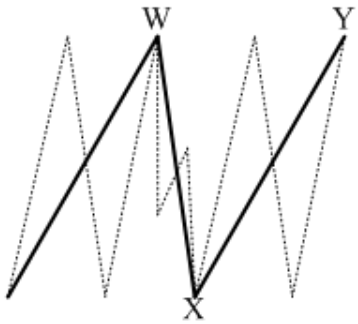


Рисунок 1.10 – Потрійна трійка

- хвиля W, хвиля X і хвиля Y можуть прийняти форму будь корекційної хвилі за винятком трикутника;
- хвиля XX і хвиля Z можуть прийняти форму будь корекційної хвилі.

Трикутники що збігаються(рис. 1.11):

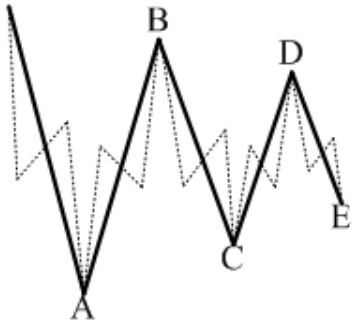


Рисунок 1.11 – Трикутники, що збігаються

- хвиля C ніколи не виходить за цінові межі хвилі B;
- хвиля D ніколи не виходить за цінові межі хвилі C;
- хвиля E ніколи не виходить за цінові межі хвилі D;
- хвиля A, хвиля B і хвиля C можуть прийняти форму будь корекційної хвилі за винятком трикутника;
- хвиля D і хвиля E можуть прийняти форму будь корекційної хвилі.

Трикутник що розбігається(рис. 1.12):

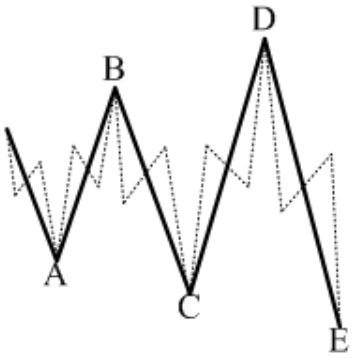


Рисунок 1.12 – Трикутники, що розбігаються

- хвиля C завжди перевищує хвилю B по довжині;
- хвиля D завжди перевищує хвилю C по довжині;
- хвиля A, хвиля B і хвиля C можуть прийняти форму будь корекційної хвилі за винятком трикутника;
- хвиля D і хвиля E можуть прийняти форму будь корекційної хвилі.

Представлені вище моделі хвиль і правила відповідають лише класичному уявленню про хвильовий аналіз. [1]

Також, хвилі Елліотта пов’язані між собою рівнями Фібоначчі. В таб. 1.1 можна побачити оцінку співвідношень хвиль Елліотта між собою.

Таблиця 1.1 – Співвідношення хвиль Елліотта один до одного

Хвиля	Класичне співвідношення хвиль
1	-
2	0.382, 0.5 або 0.618 довжини хвилі 1
3	1.618, 0.618 або 2.618 довжини хвилі 1
4	0.382 або 0.5 довжини хвилі 3
5	0.382, 0.5 або 0.618 довжини хвилі 3
A	1, 0.618 або 0.5 довжини хвилі 5
B	0.382 або 0.5 довжини хвилі A
C	1.618, 0.618 або 0.5 довжини хвилі A

Існує також його сучасне уявлення, сформоване при вивченні ринку Forex. Найдена, наприклад нова модель похилого (зрушується) трикутника, виявлені імпульси з трикутником в другій хвилі та ін. Як видно з рисунків 2.23 - 2.33, кожна імпульсна або корекційна хвиля складається з таких же

імпульсних і корекційних хвиль (виділені штриховий лінією), але вже меншою мірою. Це так звана фрактальність (вкладеність) хвиль Еліота : хвилі великих ступенів складаються з хвиль менших ступенів , які в свою чергу складаються з хвиль ще менших ступенів і так далі(рис. 1.13):.

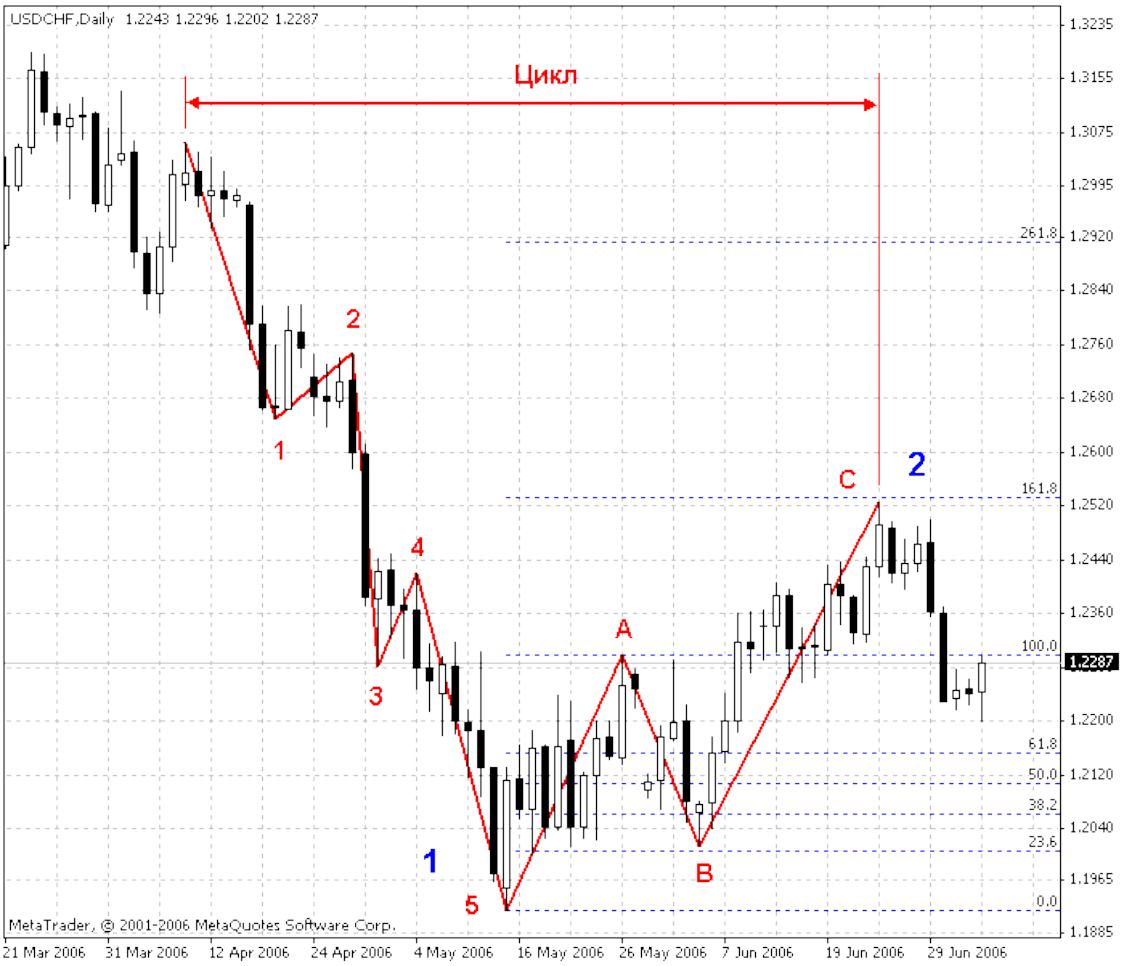


Рисунок 1.13 – Хвиля Еліота на реальному сигналі

Математичною основою теорії Еліота, за визнанням самого автора, стали так звані числа Фібоначчі. Хвильова теорія Еліота свідчить, що ринок може перебувати в двох широких фазах - бичачий ринок та ведмежий ринок. Основний принцип в теорії Еліота - кожна імпульсна хвиля складається з п’яти хвиль меншого розміру (хвилі 1, 2, 3, 4, 5), а кожна хвиля коректування (проти тренду) - з трьох хвиль (хвилі A, B, C). Найдовший цикл, згідно теорії Еліота називається Великим суперциклом, який складається з 8 хвиль (хвилі 1, 2, 3, 4, 5, A, B, C), ці хвилі в свою чергу теж складаються з 8 хвиль меншого циклу і т.д.

Для математичного викладу своєї теорії Еліот використовував принцип чисел Фібоначчі. Числа Фібоначчі відіграють важливу роль у будові повного ринкового циклу, що описується в самій теорії Еліота. Кількість хвиль, що утворюють тенденцію, збігається з числами Фібоначчі. Якщо уважно подивитися на Малюнок нижче, то можна помітити, що повний цикл складається з двох великих хвиль, восьми середніх хвиль, 34 маленьких хвиль. Аналогічно якщо розглядати бичачий ринок, то можна побачити, що бичача більша хвиля складається з однієї великої хвилі, п'яти середніх хвиль, та 21 маленьких хвиль. Якщо продовжити цей список, то наступним числом на висхідному тренді буде 89 і т.д. Відповідно, ведмежа більша хвиля складається з однієї великої хвилі, трьох середніх хвиль та 13 маленьких хвиль. Якщо ж перейти на ще нижчий рівень, то налічується вже 55 дуже маленьких хвиль і т.д.

Згідно теорії, кожна хвиля має набір характеристик. Ці характеристики засновані на масивах ринкової поведінки. У теорії Хвиль Еліота особлива увага приділяється індивідуальним прикметам кожної з хвиль. Крім того, існують певні правила пропорцій побудови хвиль Еліота. Ці правила допомагають правильно визначити моменти початку побудови хвиль і їх тривалість. Довжини хвиль вимірюються від high до low відповідної хвилі. Нижче наведені класичні співвідношення хвиль між собою які підтверджуються з погрішністю 10%. Таку похибку можна пояснити короткостроковим впливом деяких технічних або фундаментальних факторів. У цілому дані співвідношення досить умовні. Важливим є і те, що ставлення розмірів всіх хвиль один до одного може приймати значення 0.382, 0.50, 0.618, 1.618. Розглянемо характеристики кожної хвилі: Хвиля 1 Відбувається, коли «психіка ринку» майже повністю ведмежа (бича). Новини усе ще негативні (позитивні). Як правило, дуже сильна, якщо являють собою різку зміну в поточній ситуації (зміна ведмежого тренда на бичачий, прорив потужного рівня опору (підтримки) і т.д.). У спокійній ситуації зазвичай демонструє незначний рух цін на тлі загальної нерішучості. Хвиля 2 Відбувається, коли ринок різко робить відкат від підвищення (зниження) цін. Її корекція може становити майже 100% Хвилі 1, але не нижче її початку. Зазвичай становить близько 60% від Хвилі 1, розвивається на тлі домінуючого переважання інвесторів, які надають перевагу фіксуванню прибутків. Хвиля 3 Є тим, заради чого еліоти-

ки живуть. Спостерігається різке зростання оптимізму серед інвесторів. Це найбільш потужна і довга хвиля зростання (зниження) цін (ніколи не може бути найкоротшою), на якій відбувається прискорення цін і збільшення обсягів. Типова Хвиля 3 перевищує Хвилю 1 принаймні в 1,618 рази та може становити навіть більше цього значення. Хвиля 4 Часто буває проблематичною для ідентифікації. Зазвичай вона відкочується не більше, ніж на 38% Хвилі 3. Її глибина і тривалість, як правило, невелика. Оптимістичні настрої все ще переважають на ринку. Хвиля 4 не повинна перекривати Хвилю 2 до тих пір, поки п'ятихвильовий цикл є частиною кінцевого трикутника. Хвиля 5 Часто ідентифікується по імпульсній дивергенції (momentum divergences). Зростання цін на середніх обсягах торгів. Проходить на тлі масового ажіотажу у публіки. До кінця хвилі, найчастіше, відбувається різке зростання обсягів торгів. Хвиля А Більшість продовжує вважати, що зростання (зниження) скоро продовжиться з новою силою. Але вже почали з'являтися гравці, переконані в зворотному. Характеристики цієї хвилі часто дуже схожі на Хвилю 1. Хвиля В Часто дуже схожа на Хвилю 4 і дуже важка для ідентифікації. Показує незначний рух вгору (вниз) на залишках оптимізму. Хвиля С Сильна хвиля на тлі загальної переконаності про початок нової понижувальної (зростаючої) тенденції. Між тим, деякі інвестори починають обережну покупку (продаж). Вона складається з п'яти менших хвиль та становить в сумі до 1.618 рази хвилі 3.

На жаль, хвилі Еліота дуже добре помітні на «історії» ринку і погано передбача.ться для майбутнього. У зв'язку з цим практичне використання хвильової теорії Еліота найчастіше проблематично для початківців, тому що вимагає спеціальних знань та практичних навичок. [26]

1.5 Фрактальна природа хвиль Елліотта

Одним з важливих висновків Елліотта було твердження про природу, досліджуваним ним хвиль: всі хвилі всередині себе складаються з інших, більш дрібних хвиль (тобто мають таку саму внутрішню структуру, як і вся

модель) і, в свою чергу, приймають участь у формуванні точно такої ж моделі більшого часового і цінового масштабів. Таким чином, представлена на рисунку 1.14 модель приймає наступний вигляд:

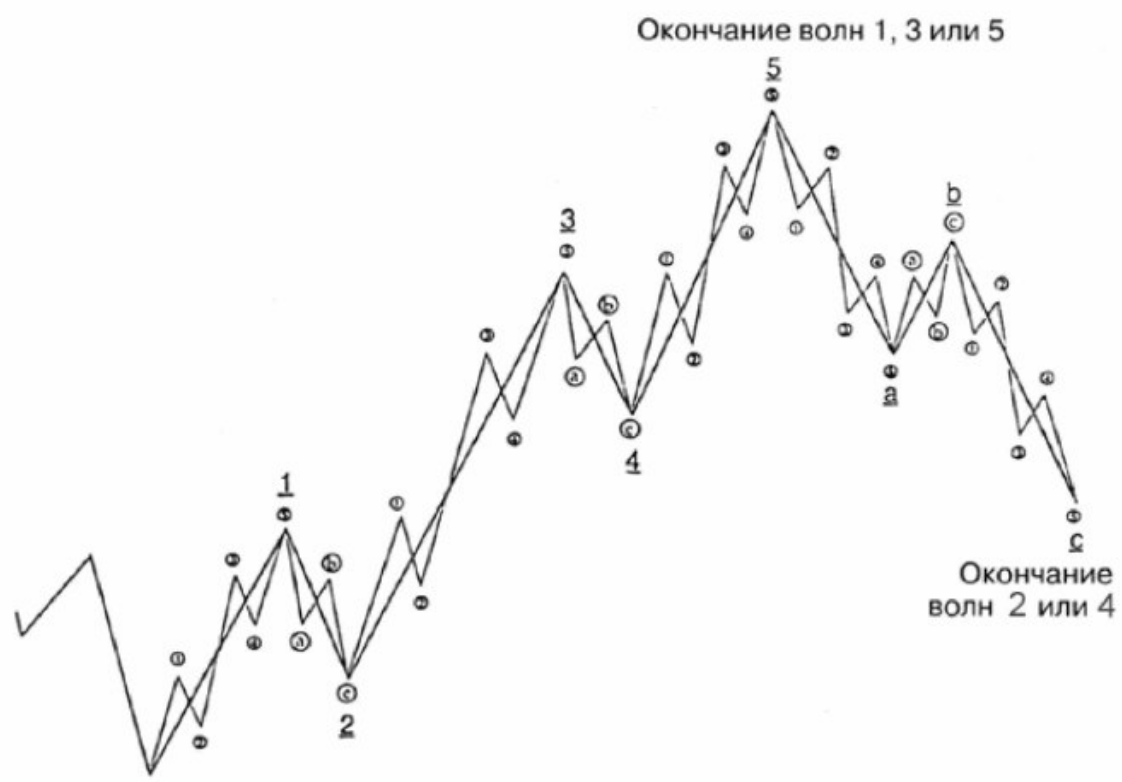


Рисунок 1.14 – Фрактальна структура хвиль Елліота

Слід зазначити, що в своєму чистому вигляді модель, представлена на рисунку 1.14, зустрічається здебільшого при малих часових інтервалах чи низькій кількості значень цін. Більш розповсюдженими є розширення окремих хвиль Елліота, описані вище, наприклад, як на рисунку 1.15. Найбільш загальним випадком є модель, представлена на рисунку 1.15. Аналізуючи її, ми можемо виділити великі хвилі (1)-(5) та (a),(b),(c), які в свою чергу складаються з більш дрібних хвиль. Таким чином, ми отримуємо схему, представлену на рисунку 1.14.

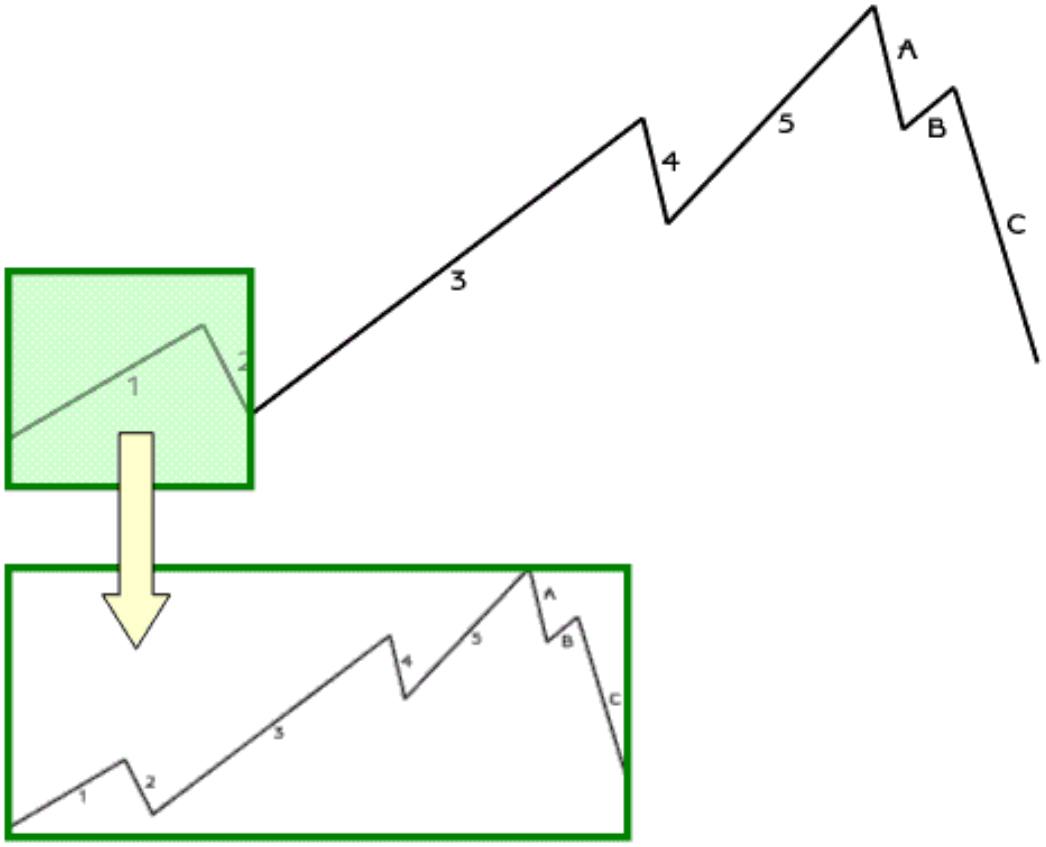


Рисунок 1.15 – Окремий випадок розтягнення хвилі Елліота

У моделі хвиль Елліота закладена самоподібність, тобто має фрактальну структуру. Для її дослідження ми можемо використати фрактальну розмірність *D*, що означає статичну величину, яка дозволяє зрозуміти наскільки повно фрактал заповнює простір, якщо збільшувати його до дрібних часток. Принцип фрактальної розмірності демонструє грубість фракталу в порівнянні з чистотою та топологічною розмірністю, якою володіють класичні геометричні фігури. Наприклад, для прямої лінії розмірність дорівнює 1.

Фрактальна розмірність має безліч специфічних визначень. Найважливішими з них є: розмірність Реній, розмірність Хаусдорфа та компактна розмірність. Через свою простоту на практиці застосовуються розмірність Мінковського і кореляційна розмірність. Хоч для деяких фракталів усі ці розмірності збігаються. В цілому, вони не є еквівалентними.

В цій роботі використано розмірність Мінковського саме через зручність її обчислення за допомогою електронно - обічслювальних засобів. [22]

1.6 Вейвлет аналіз хвиль Елліотта

Вейвлет-аналіз є варіацією спектрального аналізу, в якому роль простих коливань відіграють особливі функції так звані вейвлети. Базова вейвлет-функція- це деяке ”коротке” коливання, але не тільки. Поняття частоти класичного спектрального аналізу тут замінено масштабом, і, щоб заміщення ”короткими хвилями” всю тимчасову вісь, введено часові зрушення функцій. Таким чином, базис вейвлетів - це функції типу $\psi\left(\frac{t-b}{a}\right)$, де b - зсув, a – масштаб. Окрім того, щоб бути як вейвлет, функція $\psi(t)$ повинна мати у своїй структурі нульову площу, а краще, рівними нулю перший, другий та інші моменти. Фур’є - перетворення таких функцій дорівнює нулю при $\omega = 0$ і має вигляд фільтра у формі смужки. При різних значеннях $'a'$ це буде набір смугових фільтрів. Угрупуваннння вейвлетів у тимчасовій або частотній області використовуються для представлення сигналів і функцій у вигляді суперпозиції вейвлетів на різних масштабних рівнях розкладання сигналів. [23]

Аналітика вейвлетного перетворення сигналів визначається математичнім підґрунтям розкладання сигналів, яке є аналогічнім перетворенню Фур’є. Відмінною особливістю вейвлет-перетворень є нова основа розкладання сигналів - вейвлетної функції. Властивості якої принципово важливі як для самої можливості розкладання сигналів за одиничними вейвлетними функціями, так і для цілеспрямованих дій над вейвлетними спектрами сигналів, в тому числі з подальшою реконструкцією сигналів з обробленими вейвлетними спектрами.

Вейвлети бувають: ортогональними, полуортогональнимі, біортогональних. Ці функції можуть бути симетричними, асиметричними і несиметричними, з компактною областю визначення і не мають такої, а також мати різну ступінь гладкості. Деякі функції мають аналітичний вираз, інші - швидкий алгоритм обчислення вейвлет-перетворення. На практиці бажано було б мати ортогональні симетричні і асиметричні вейвлети, але таких ідеальних вейвлетів не існує. Найчастіше застосовуються біортогональні вейвлети. [24]

1.6.1 Визначення вейвлета

До вейвлетів відносять локалізовані функції, які конструюються з одного материнського вейвлета $\psi(t)$ (або з якоїсь іншої незалежної змінної) шляхом операцій зсуву по аргументу (b) і масштабної зміни (a):

$$\psi_{ab}(t) = (1/(\sqrt{|a|}))\psi((t-b)/a), (a,b) \in R, \psi(t) \in L^2(R), \qquad (1.1)$$

де множитель $(1/(\sqrt{|a|}))$ забезпечує незалежність норми функцій від масштабного числа ’a’.

Вейвлетний масштабно-часовий спектр $C(a,b)$ на відміну від Фур’є - спектра є функцією двох аргументів: масштабу вейвлета ’a’ (в одиницях, зворотних частоті), і тимчасового зсуву вейвлета за сигналом ’b’ (в одиницях часу), при цьому параметри ’a’ і ’b’ можуть приймати будь-які значення в межах областей їх визначення.

Для кількісних методів аналізу (декомпозиція сигналів з можливістю подальшої лінійної реконструкції сигналів з оброблених вейвлет-спектрів) в якості вейвлетніх підґрунть можна використовувати будь-які локалізовані функції $\psi(t)$, якщо для них існують функції-двійники $\psi^{\#}(t)$, такі, що сімейства $\{\psi_{ab}(t)\}$ і $\{\psi_{ab}(t)\}$ можуть утворювати парні базиси функціонального простору $L^2(R)$. Вейвлети, певні таким чином, дозволяють уявити будь-яку довільну функцію в просторі $L^2(R)$ у вигляді ряду:

$$s(t) = \sum_{a,b} C(a,b)\psi^{\#}_{ab}(t), (a,b) \in I, \qquad (1.2)$$

де коефіцієнти $C(a,b)$ – проекції сигналу на вейвлетного базис простору, які визначаються скалярним твором

$$C(a,b) = \langle s(t), \psi_{ab}(t) \rangle = \int_{-\infty}^{+\infty} s(t)\psi_{ab}(t)\,dt \tag{1.3}$$

Якщо вейвлет $\psi(t)$ має властивість ортогональності, то $\psi^{\#}(t)\psi(t)$ і вейвлетного базис ортогонален. Вейвлет може бути ортогональним, проте якщо він має двійника, і пара $(\psi(t), \psi^{\#}(t))$ дає можливість сформувати угруповання $\{\psi_{mk}(t)\}$ і $\{\psi^{\#}_{zp}(t)\}$, задовольняють умові ортогональності на цілих числах I:

$$\langle \psi_{mk}(t), \psi^{\#}_{zp}(t) \rangle = \delta_{mz}\delta_{kp}, m,k,z,p \in I, \tag{1.4}$$

то можливо розпад сигналів на вейвлетні ряди зі зворотним формулами реконструкції. [24]

1.6.2 Неперервне вейвлет перетворення

Вейвлет-перетворення для неперервного сигналу відносно вейвлет функції визначається наступним чином [9]:

$$T(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\psi^{*}\left(\frac{t-b}{a}\right) dt \tag{1.5}$$

де ψ^{*} означає комплексне спряження для ψ, параметр $b \in R$ відповідає часовому зсуву, і називається параметром положення, параметр $a > 0$ задає мас-

штабування і називається параметром розтягнення.

$$w(a) \equiv \frac{1}{\sqrt{a}}. \tag{1.6}$$

Ми можемо визначити нормовану функцію наступним чином:

$$\psi_{a,b} = \frac{1}{\sqrt{a}} \psi \left(\frac{t-b}{a} \right) \tag{1.7}$$

що позначає часовий зсув на b і масштабування по часу на a. Тоді формула вейлет-перетворення зміниться на

$$T(a,b) = \int_{-\infty}^{\infty} x(t)\, \psi_{a,b}^{*} dt \tag{1.8}$$

Вхідний сигнал может бути відтворенний по формулі оберненного перетворення

$$x(t) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} T(a,b)\, \psi_{a,b}(t)\, da\, db \tag{1.9}$$

1.6.3 Перелік основних вейвлетів

Основні функції, які створюють вейвлети, так звані материнські вейвлети, наведені в табл. 1.2.

Таблиця 1.2 – Материнскі функції вейвлетів

Вейвлети	**Аналітичний запис** $\psi(t)$	**Спектральна щільність** $\psi(\omega)$
Дійсні безперервні базиси		
Гаусові: – Першого порядка або WAVE-вейвлет – Другого порядка або МНАТ-вейвлет «мексиканське капелюх» – n -порядка	$-t\exp(-t^2/2)$ $(1-t^2)\exp(-t^2/2)$ $(-1)^n\frac{d^n}{dt^n}[\exp(-t^2/2)]$	$(i\omega)\sqrt{2\pi}\exp(-\omega^2/2)$ $(i\omega)^2\sqrt{2\pi}\exp(-\omega^2/2)$ $(-1)^n(i\omega)^n\sqrt{2\pi}\exp(-\omega^2/2)$
DOG-difference of gaussians	$e^{-t^2 2}-0{,}5e^{-t^2/8}$	$\sqrt{2\pi}(e^{-\omega^2/2}-e^{-2\omega^2})$
Дійсні дискретні базиси		
HAAR-вейвлет	$\geq\begin{cases}1, & 0\le t\le 1/2\\ -1, & 1/2\le t\le 1\\ 0, & t<0,\ t>0\end{cases}$	$ie^{i\omega/2}\frac{\sin^2\omega/4}{\omega/4}$
Комплексні		
Морле (Morlet)	$e^{i\omega_0 t}e^{-t^2/2}$	$\sigma(\omega)\sqrt{2\pi}e^{-(\omega-\omega_0)^2/2}$
Пауля (Paul) – чим більше n, тим більше нульових моментів має вейвлет	$\Gamma(n+1)\frac{i^n}{(1-n)^{n+1}}$	$\sigma(\omega)\sqrt{2\pi}(\omega)^n e^{-\omega}$

Найбільш поширені базиси створюються на основі похідних функції Гауса ($g_0(t)=e^{-t^2/2}$). Це обумовлено тим, що функція Гауса має найкращі показники локалізації як у часовій, так і в частотній областях.Серед комплексних вейвлетів найбільше часто використовується базис, заснований на добре локалізованому у часовій та в частотної областях вейвлеті Морле. Характерний параметр ω_0 дозволяє змінювати вибірковість базису. Дійсна та уявна частини $\psi(t)$ – це амплітудно-модульовані коливання.

Вибір конкретного материнського вейвлета цілком залежить від характеру поставленого завдання та від конкретного аналізованого сигналу. [5]

1.6.4 Відображення вейвлет-перетворення

Результатом вейвлет-перетворення одновимірного числового ряду (сигналу) є двовимірний масив значень коефіцієнтів C (a, b). Розподіл цих показників у просторі (a, b) є: часовий масштаб, тимчасова локалізація, дають інформацію про зміну в часі відносного вкладу в сигнали вейвлетного компоненту різного масштабу і називаються спектром коефіцієнтів вейвлет- перетворення, масштабно-тимчасовим (частотно-тимчасовим) спектром або просто вейвлет-спектром (wavelet spectrum).

Спектр C (a, b) одновимірного сигналу являє собою поверхню в тривимірному просторі. Способи візуалізації спектру можуть бути різноманітними. Найбільш поширений спосіб - проекція на площину ab з ізолініями (ізорівнями), що дозволяє відстежити зміни коефіцієнтів на різних часових масштабах, а також виявити картину локальних екстремумів цих поверхонь (”па- горбів” і ”западин”), так званий ”скелет” (skeleton) структури аналізованого процесу. При широкому діапазоні масштабів застосовуються логарифмічні координати (log a, b). Приклад вейвлетного спектра найпростішого сигналу при його розкладанні вейвлетом Mhat наведено на рис. 1.16 .

На вертикальних перетинах (перетинах зсуву b) вейвлет-спектр передає компонентний склад сигналу (з даного комплекту вейвлетов) в кожний поточний момент. За змістом перетворення, як скалярного сигналу з вейвлетом, ясно, якщо значення коефіцієнтів кожної поточної тимчасо- вої точки на маштабних перетинах більше, чим сильніша кореляція між вейвлетом дцього маштабу і поведінка сигналу в околицях цієї точки. Відповідно, перетину за параметром ’a’ демонструють зміни в сигналі компоненти цього масштабу ’a’ з часом.

Вейвлетні складові сигналу в перетинах його спектра не мають зовсім нічого спільного з синусоїдами, і представлені, як правило, сигналами з досить складною і не завжди зрозумілою формою, що може ускладнювати їх наочне уявлення і розуміння. [24]

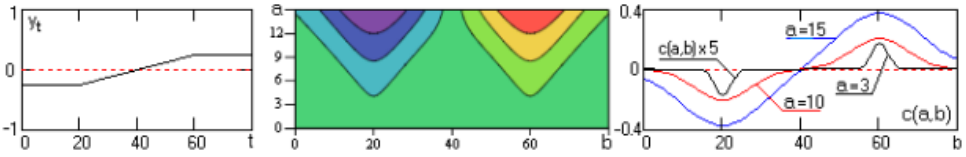


Рисунок 1.16 – Приклад вейвлет перетворення

1.6.5 Процедура вейвлет-перетворення

Процедура перетворення бере початок із масштабу $a = 1$ і триває при зростаючих значеннях a, тобто, аналіз починається з високих частот і проводиться в сторону низьких. Перше значення ’a’ відповідає найбільш стислому вейвлету. При збільшенні значення ’a’ вейвлет розширюється. Він вміщається в початок сигналу ($t = 0$), перемножується із сигналом, інтегрується на інтервалі свого завдання і до нормального рівня $\frac{1}{\sqrt{a}}$. При завданні парних або непарних функцій вейвлетів результат обчислення $C(a,b)$ вміщується в точку $(a = 1, b = 0)$ масштабно-часового спектру перетворення. Зрушення b може розглядатися як час з моменту $t = 0$, при цьому координа- тна вісь b, по суті, повторює тимчасову вісь сигналу. Для повного включення в обробку всіх точок вхідного сигналу потрібно завдання початкових (і кінце- вих) умов перетворення (певних значень вхідного сигналу при $t < 0$ і $t > tmax$ на напівширину вікна вейвлета). При односторонньому завданні вейвлетів результат відноситься, як правило, до тимчасового положення середньої точки вікна вейвлету.

Надалі вейвлет масштабу $a = 1$ зсувається вправо на значення b і процес повторюється. Отримуємо значення, відповідне $t = b$ в рядку $a = 1$ на частотно-часовому плані. Процедура повторюється до тих пір, поки вейвлет не досягне кінцевого сигналу. Таким чином отримуємо рядок точок на масштабно-часовому плані для масштабу $a = 1$.

Для обчислення наступної маштабної функції рядки значення α збільшуються на деяке значення. При НПВ в аналітичній формі $\Delta b \to 0$ і $\Delta a \to 0$. При виконанні перетворення в комп’ютері обчислюється апроксимація зі

збільшенням обох параметрів з певним кроком. Тим самим ми здійснюємо дискретизацию масштабно-часової площини.

Початкове значення маштабного коефіцієнта може бути і менбезше 1. Взагалі, для деталізації найвищих частот сигналу мінімальний розмір вікна вейвлета не повинен перевищувати періоду самої високочастотної гармоніки. Якщо в сигналі присутні спектральні компоненти, відповідні поточному значенню α, то інтеграл вейвлета з сигналом в інтервалі, де ця спектральна компонент присутній, дає відносно велике значення. В іншому ви- падку - мало або дорівнює нулю, тому що середнє значення вейвлетного функції дорівнює нулю. Зі збільшенням масштабу (ширини вікна) вейвлета перетворення виділяє все більш низькі частоти.

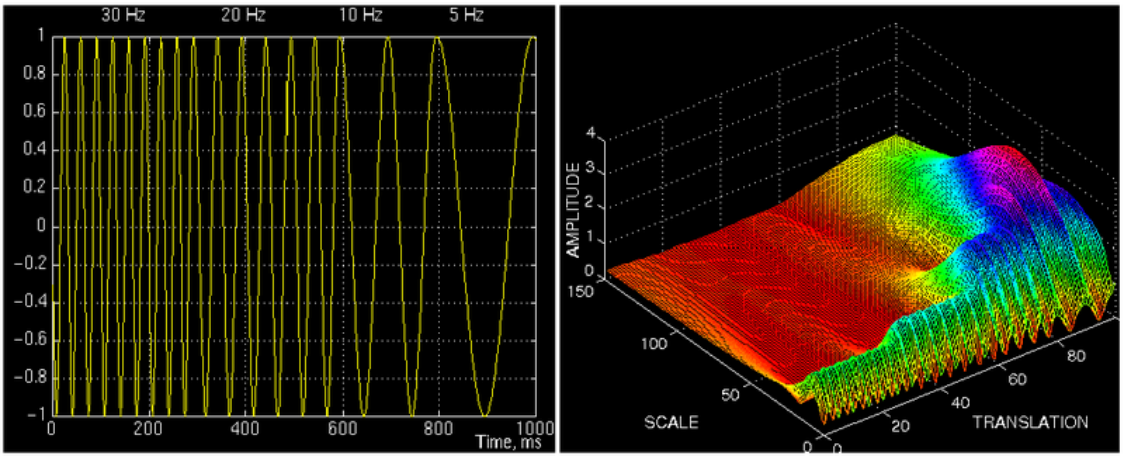


Рисунок 1.17 – Неперервне вейвлет перетворення

На (рис. 1.17) наведено приклад модельного сигналу і спектра його безперервного вейвлет-перетворення. У загальному випадку, значення параметрів ’a’ і ’b’ в (рис. 1.17) є безперервними, і безліч ґрунтовних функцій є понад нормою. В силу цього безперервне перетворення сигналів містить дуже великий обсяг інформації. Сигнал, визначеним на R, відповідає вейвлетному спектру на $R \times R$. З позиції збереження обсягу інформації при перетвореннях сигналів це означає, що вейвлетний спектр НПВ має величезну надмірність. [25]

1.7 Архітектура системи підтримки та прийняття рішень

Архітектура СППР залежить того, де система буду використовватися. На основі цього використовуються різні компоненти СППР, типи обробки даних, інтерфейси користувача і тд.

1.7.1 Компоненти СППР

СППР складається з наступних компонентів:

а) мовленнєва система – складається з усіх повідомлень, які може сприймати СППР;

б) система подання проміжних та кінцевих результатів – містить у собі всі повідомлення, які може генерувати СППР;

в) база знань та даних – містить всі знання, моделі, дані, правила, алгоритми, які необхідні для генерації висновків та вибору розв'язків;

г) система обробки даних та генерації варіантів розв'язків – виконує основні функції по генерації розв'язків – це активна частина СППР.

Всі системи зображені на рисунку 1.18 :

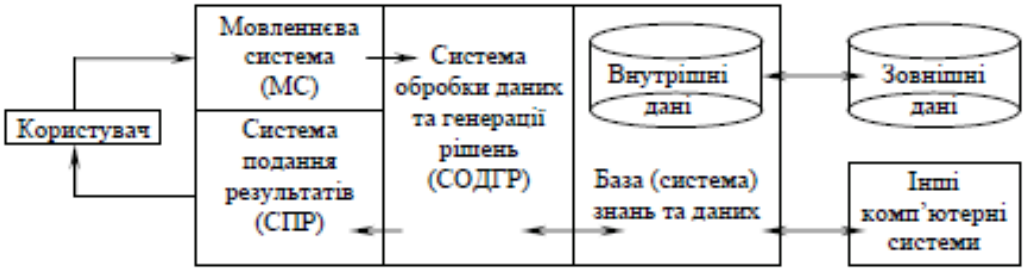


Рисунок 1.18 – Компоненти СППР

Послідовність дій при роботі з СППР:

а) користувач робить запит до СППР за допомогою елементів мовленнєвої системи. Це може бути: запит на введення додаткових даних чи знань; запит на уточнення попереднього запиту або відповіді; запит на розв’язання конкретної задачі;
б) запит передається до СОДГР, яка виконує його обробку. В процесі обробки СОДГР звертається до БЗД та генерує результат;
в) потім надається повідомлення користувачу про виконану чи невиконану роботу. Це здійснюється за допомогою системи подання результатів;
г) система обробки даних та генерації варіантів розв’язків – виконує основні функції по генерації розв’язків – це активна частина СППР.

1.7.2 Функції СППР

Базові функції СППР зображені на рисунку 1.19 :

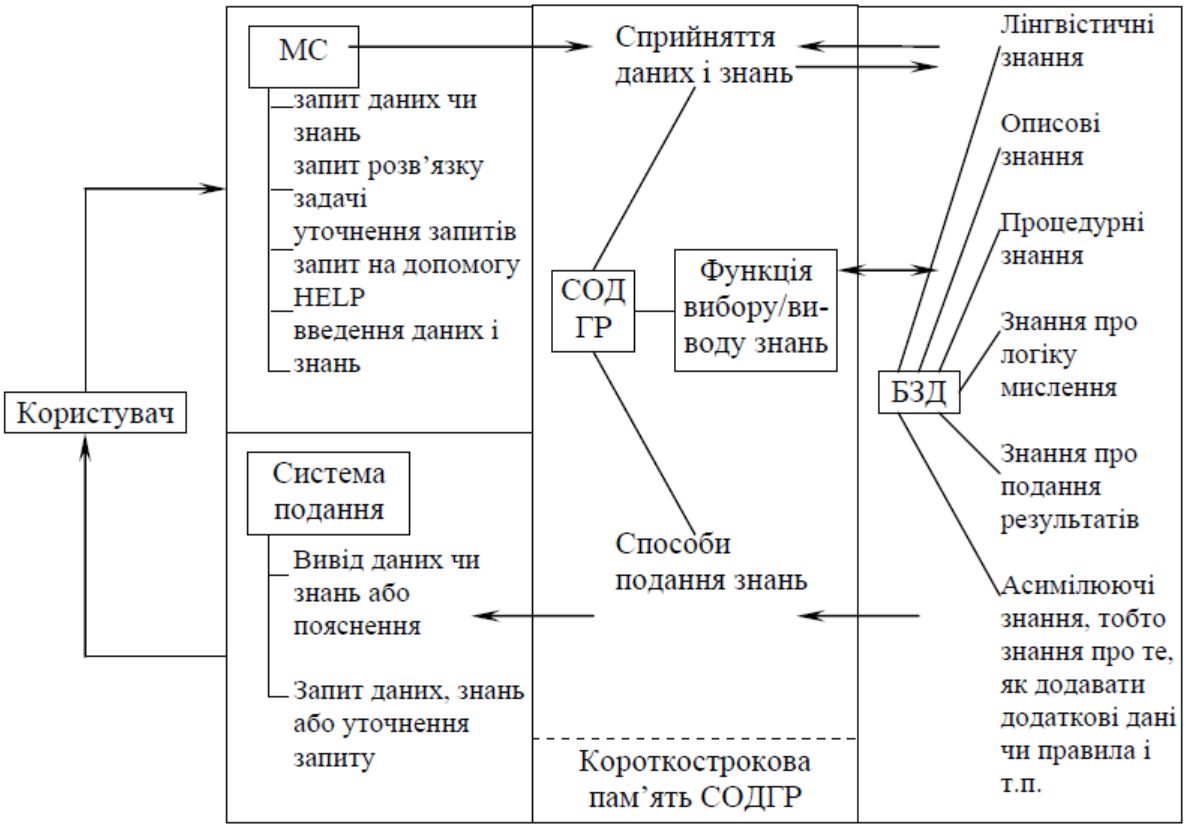


Рисунок 1.19 – Базові функції СППР

При надходженні запиту від користувача на розв'язання задачі в дію вступає функція СОДГР: "Функція вибору/виводу знань". В результаті СОДГР селектує або виводить (обчислює) знання, які становлять розв'язок.

1.7.3 Уточнення компонентів СППР

Мовленнєва система – це не програмний продукт, хоча може бути його частиною. Це система мовленнєвих конструкцій, що включає усі запроси, які може зробити користувач. Для організації запиту користувач обирає одну з припустимих конструкцій мовленнєвої системи. Може бути реалізована у вигляді:

а) командних рядків;
б) меню;
в) миші;

г) природньої мови;
д) заповнення певної форми.

Система подання – також не є програмним продуктом. Це також система мовленнєвих та графічних конструкцій, що включає усі типи повідомлень та відгуків, які може генерувати СППР. Вибір елементу, який буде використаний в якості відгуку, здійснює СОДГР. Відгук може бути у вигляді тексту, таблиці, графіка і т.д.

База знань та даних – містить усілякі типи знань та даних:

а) числові дані;
б) алгоритми або процедурні знання;
в) лінгвістичні змінні;
г) правила виводу (наприклад, композиційне правило виводу);
д) знання про подання результатів (в якому вигляді: графіки, таблиці, дерево розв’язків і т.п.).
е) моделі, умови, обмеження;
ж) асимілюючі знання, тобто знання про те, як розширювати БЗД.

Система обробки даних та генерації варіантів розв’язків – основне програмне забезпечення, яке реагує на запроси користувача та генерує належну відповідь. Вона приймає запит користувача, здобуває необхідні дані та знання з БЗД, виконує необхідні обчислювальні дії та передає результат у систему подання результатів. При цьому СОДГР може модифікувати або розширювати БЗД.

1.7.4 Класифікація СППР за типом обробки даних та знань

Визначними факторами для класифікації є такі:

а) обмеження на зміст та метод зберігання інформації в БЗД;
б) обмеження на можливості СОДГР по обробці даних та генерації. результатів.

1.7.4.1 Текстово-орієнтовані СППР

В цих СППР:

а) БЗД складається з текстових файлів, які являють собою інформації для ОПР (це так звана електронна документація);
б) СОДГР виконує різноманітні маніпуляції над текстовою документацією, містить програмне забезпечення, що полегшуєкористувачу складання викликів; результатів;
в) система подання містить усі можливі формати подання текстової,табличної та графічної інформації. Також повідомлення, які полегшують користувачу спілкування з СППР.

Ще одна властивість текстової СППР: можливість гіпертекстової підтримки. Гіпертекст встановлює зв'язок поміж знаннями, що містяться урізних файлах тексту. При цьому кожний фрагмент тексту пов'язується з іншими фрагментами, які концептуально з ним пов'язані.

1.7.4.2 СППР, орієнтовані на використання бази даних

Це ще один приватний випадок СППР. Найбільш розповсюджені – реляційні БД, тобто у цьому випадку йде обробка суворо структурованих знань у вигляді числових та описових даних. У такій системі СОДГР містить три типи програмного забезпечення:

а) ПЗ для СУБД;
б) інтерактивне ПЗ для обробки запитів користувача;
в) спеціальне ПЗ, створене для задоволення потреб користувача (воно зазвичай містить деякі логічні правила аналізу даних і формування відповіді на запит, а також необхідні обчислення: – статистика, прогноз, порівняння).

1.7.4.3 СППР, орієнтовані на використання електронних таблиць

При використанні технології на основі електронних таблиць для керування знаннями користувач СППР не тільки може створити, переглянути та модифікувати процедурні знання в БЗ, але може надати запит СОДГР виконати команди, які там містяться.

БЗ містить файли з таблицями, які наповнені описовими та процедурними знаннями. СОДГР може виконувати алгоритмічні процедури. Функції СОДГР:

а) визначення вмісту комірок електронних таблиць;
б) створення макровизначень;
в) об’єднання кількох таблиць в одну з метою розв’язання більш складних задач;
г) надання допомоги користувачу в організації запитів;
д) форматування електронних таблиць та передача результатів до системи подання.

1.7.4.4 СППР на основі рішаючих процедур (алгоритмів)

Областю застосування такої СППР є: оптимальне інвестування, максимізація прибутку конкретного підприємства, розташування центру та розподілення пунктів для торгової мережі. Така СППР розв’язує задачі: фінансування, екстраполювання, прогнозування, статистичного аналізу, планування виробництва, оптимізації.

Існує два підходи до проектування СППР:

а) фіксований – алгоритм є складовою частиною СОДГР, а це означає, що практично досить важко додати новий або модифікувати існуючий алгоритм;
б) гнучкий – всі процедури зберігаються в БЗД.

1.7.4.5 СППР на основі правил

В СППР такого типу в БЗД зберігаються набори правил та описів поточного стану досліджуваного об’єкта, а СОДГР обирає необхідний набір правил з БЗД з метою реалізації логічного виводу типу: If <описувана ситуація> then <дія, яку необхідно виконати> because.

1.7.4.6 Гібридна архітектура СППР

До неї можуть входити всі відомі елементи. Джерела інформації – зовнішні та внутрішні.

1.7.5 Моделювання процесу прийняття рішень

СППР повинна допомагати людині у прийнятті рішення, а тому характер її функціонування має бути узгоджений із процесом прийняття рішень людиною. Рішення – це обгрунтований вибір однієї з можливих альтернатив дій. Існують наступні типи рішень:

а) рішення особистого характеру;
б) ділові рішення (які стосуються діяльності організації).

1.7.6 Загальна характеристика процесу прийняття рішення

Процес прийняття рішення складається з таких етапів:

а) постановка задачі відносно того, якого типу рішення ми повинні прийняти. Постановка задачі ускладнюється, якщо задача розв’язується групою ОПР;
б) визначення множини можливих альтернатив;
в) визначення критеріїв вибору рішень. Критерії можуть мати чіткий чисельний або евристичний характер.

1.7.7 Фунціональна схема СППР

Фунціональна схема зображена на рисунку 1.20 :

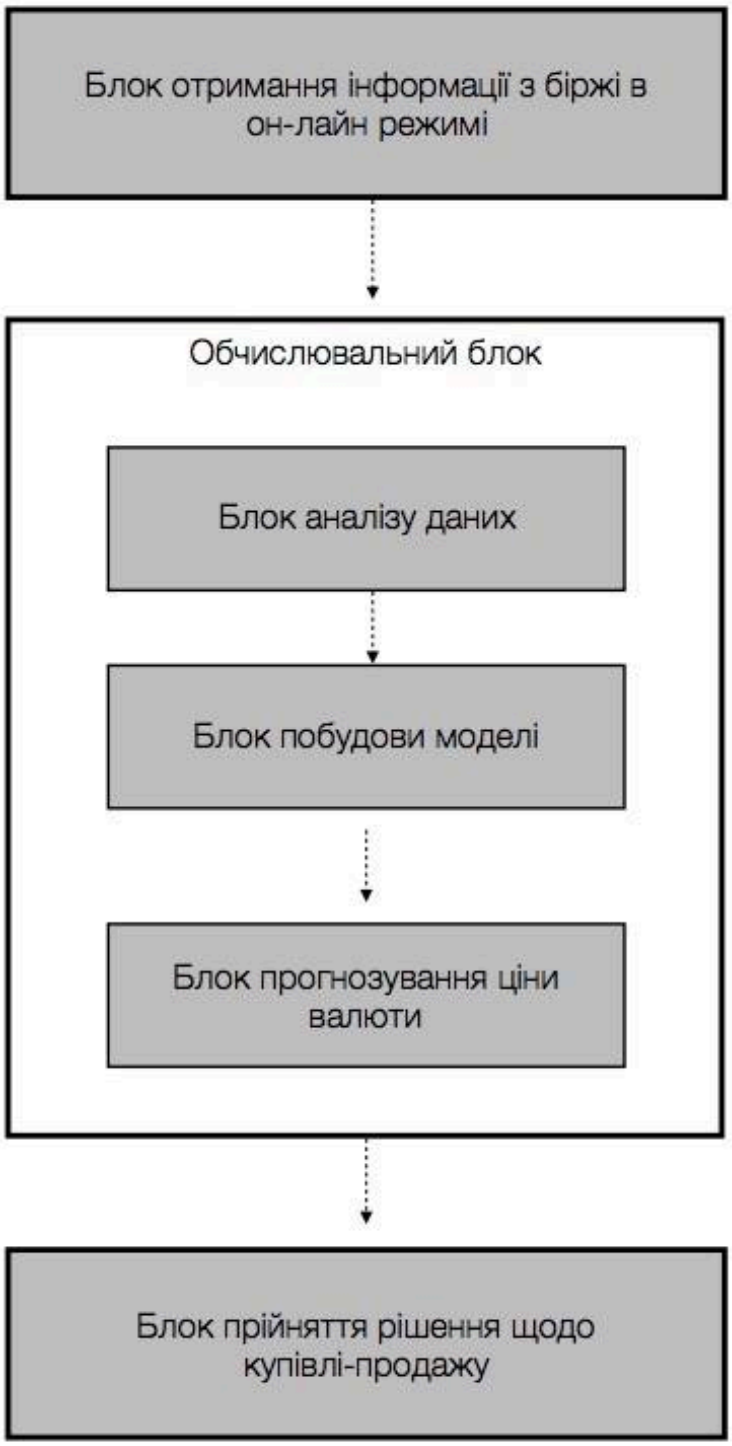


Рисунок 1.20 – Структура СППР при проведенні торгівельних операцій

Висновки до розділу

У цьому розділі проведено аналіз існуючих підходів до аналізу фінансового ринку, оцінені можливі недоліки цих підходів.

Описано походження хвиль Елліотта, їх основні властивості, фрактальну природу, класифікацію та наведено приклади.

Описано вейвлет-аналіз у контексті хвиль Елліотта і наведено базисні функції вейвлет-перетворення.

Надано теоретичні відомості про СППР та їх класифікацію. Розроблено фунціональну та архітектурну структуру СППР для роботи на ринку ціних паперів.

Можливими напрямками розвитку представленої системи є:

- інтеграція даної системи з іншими СППР;
- розробка спеціалізованого інтерфейсу для роботи з виділення хвиль Елліотта на основі вейвлетних індикаторів;
- використання нейронних мереж для автоматичного виявлення хвиль Елліотта.

РОЗДІЛ 2 ПОБУДОВА МОДЕЛІ ПРИЙНЯТТЯ РІШЕНЬ

2.1 Дослідження отриманих результатів

За допомогою розробленної системи прийняття рішень для часових рядів валютних котирувань проведено аналіз ефективності ідентифікації хвиль Елліотта за допомогою різних вейвлет-перетворень. У дослідженні приймали участь такі вейвлет перетворення: DOG, Ріккера, Морле та Пауля.

Це дослідження спрямовано на виявлення вейвлет-перетворення, яке дозволить передбачувати наступні значення валютних котирувань. Щоб передбачити наступне значення за допомогою вейвлет-перетворення, воно повинно точно описувати поведінку тренду часового ряду на граничних значеннях. Перевірка ефективності вейвлет-перетворення для даної задачі виконано на основі наступних кроків:

а) розбиття часового ряду на тестову вибірку і навчальну(основну) для вейвлет-перетворення;
б) вейвлет-перетворення виконувалось на основній вибірці;
в) вейвлет-перетворення виконувалось на всій вибірці;
г) з результатів вейвлет-перетворення згенерується індикатор
д) напрямок тренду повинен співпадати в індикаторі та тестовій вибірці та напрям тренду на обох перетворення повинен співпадати.

Для достлідження вибраний часовий ряд валютних котирувань евро/американський долар. Вибірка даних була розбита так:

а) вибірка з 11-04-2017 до 29-09-2017 - навчальна(основна);
б) вибірка з 29-09-2017 до 04-11-2017 - тестова.

2.1.1 Розробка індикатора вейвлет-перетворення

Результатом неперервного вейвлет-перетворення є частотно-часовий спектр. Як можна побачити данні представлені у трьохвимірному просторі 2.1 .

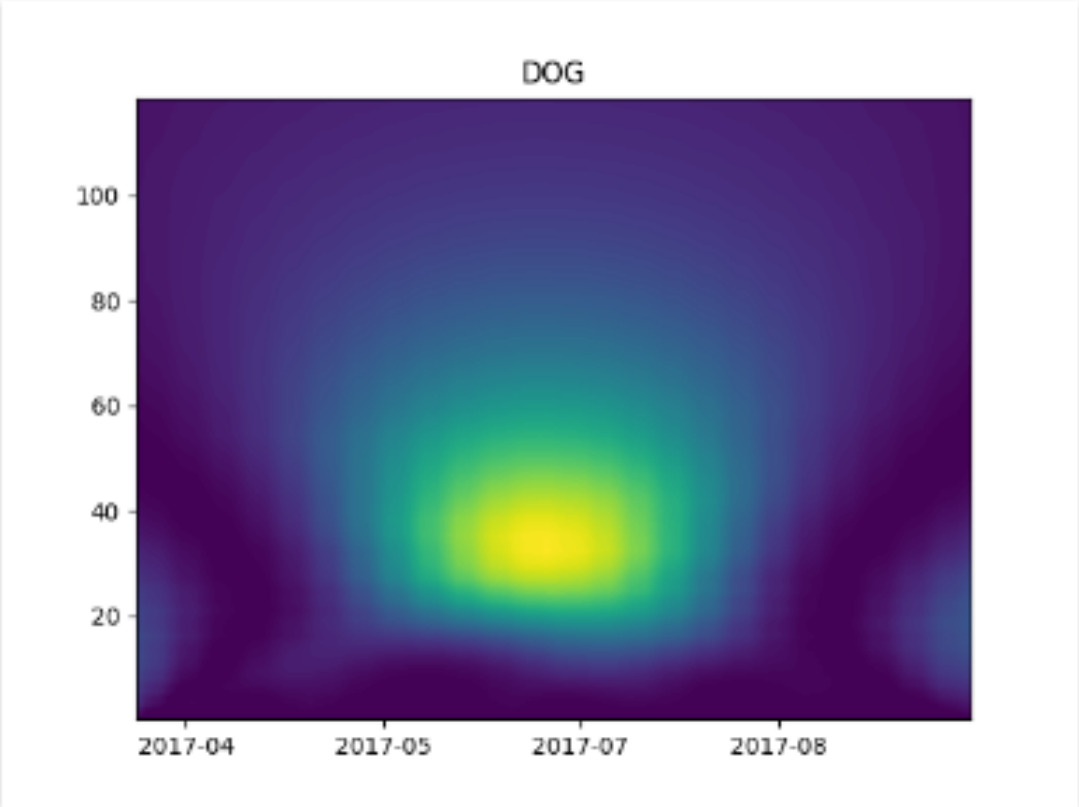


Рисунок 2.1 – Спектр вейвлет-перетворення на основі вейвлета DOG

Такі дані важко аналізувати, а тим паче використовувати для наступних досліджень або алгоритмів.

На основі цього було прийнято рішення переформатувати дані та представити їх більш читабельному вигляді. Таким чином всі частости було сплюснуто та нормалізовано.

В результаті вейвлет перетворення має наступний вигляд 2.2 .

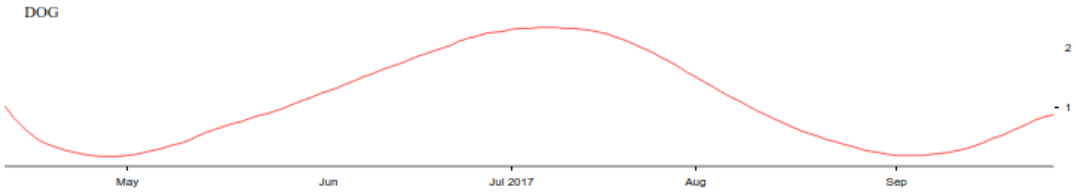


Рисунок 2.2 – Індикатор вейвлет-перетворення на основі вейвлета DOG

2.1.2 Аналіз DOG(Derivative of Gaussian) вейвлета

Спочатку було вирішено досліджувати вейлет DOG.

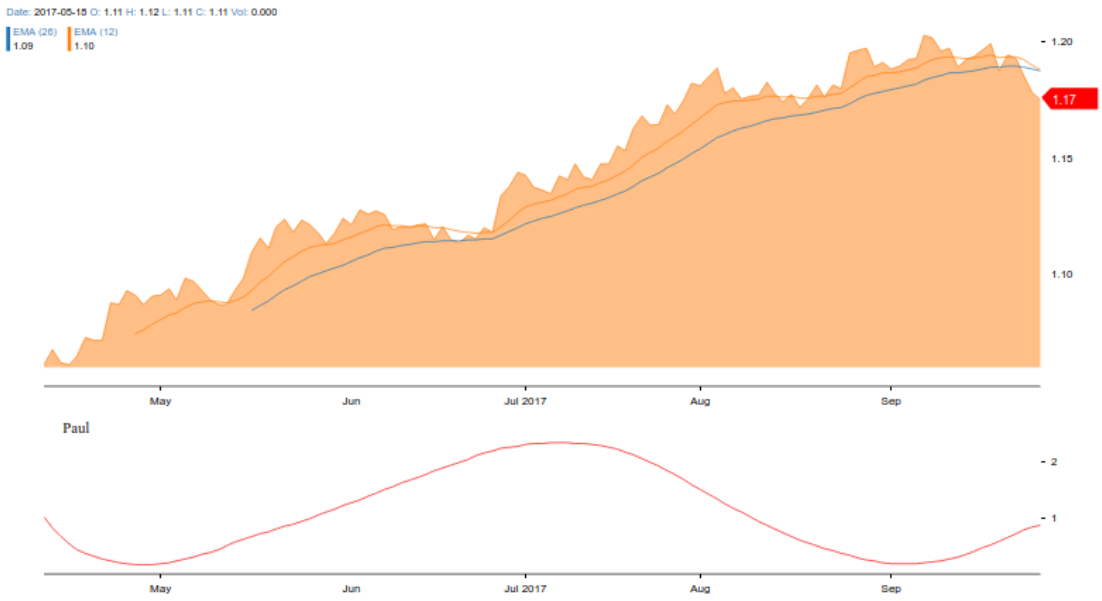


Рисунок 2.3 – Индикатор вейвлета DOG на навчальній вибірці

На рисунку 2.3 видно, що вейвлет погано описує поведінку часового ряду валютних котирувань. Проте вейвлет добре виявляет імпульсну хвилю про що свідчить великий пік на графіку індикатора.

Наступним кроком було виконання вейвлет-перетворення на всьому графіку за тестовою вибіркою. Результати представлені на рисунку 2.4 .

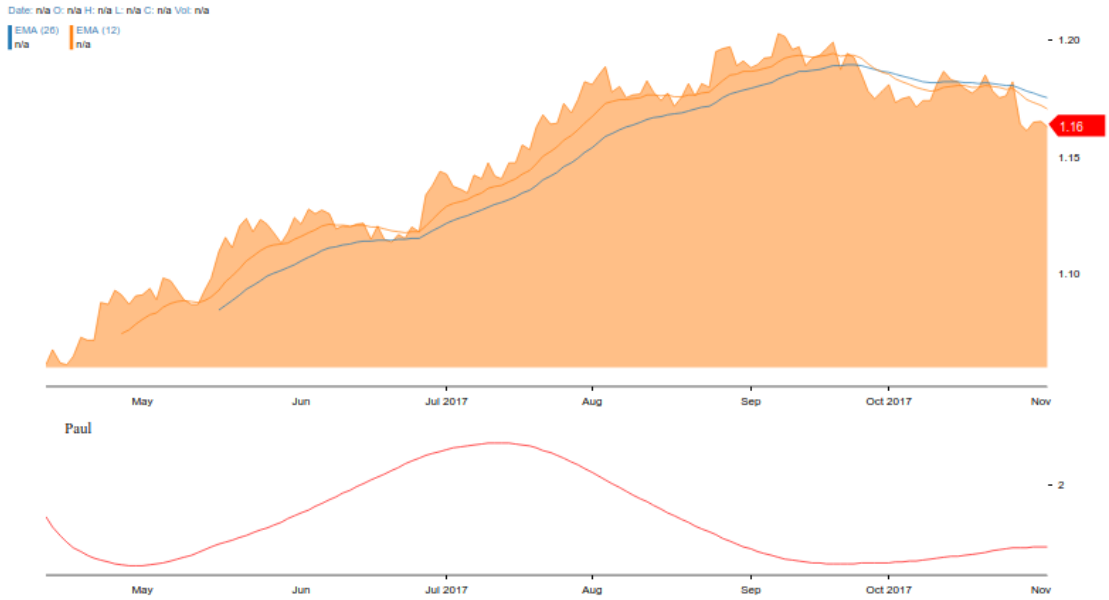


Рисунок 2.4 – Индикатор вейвлета DOG на навчальній вибірці та тестовій вибірці

По графіку індикатора видно, що за допомогою цього вейвлета неможливо спрогнозувати наступні значення часового ряду, оскільки вейвлет не чутливий до характеру коливань валютних котирувань.

2.1.3 Аналіз вейвлета Морле

Вейвлет-перетворення часового ряду валютних котирувань на основі вейвлета Морле зображено на рисунку 2.5 .

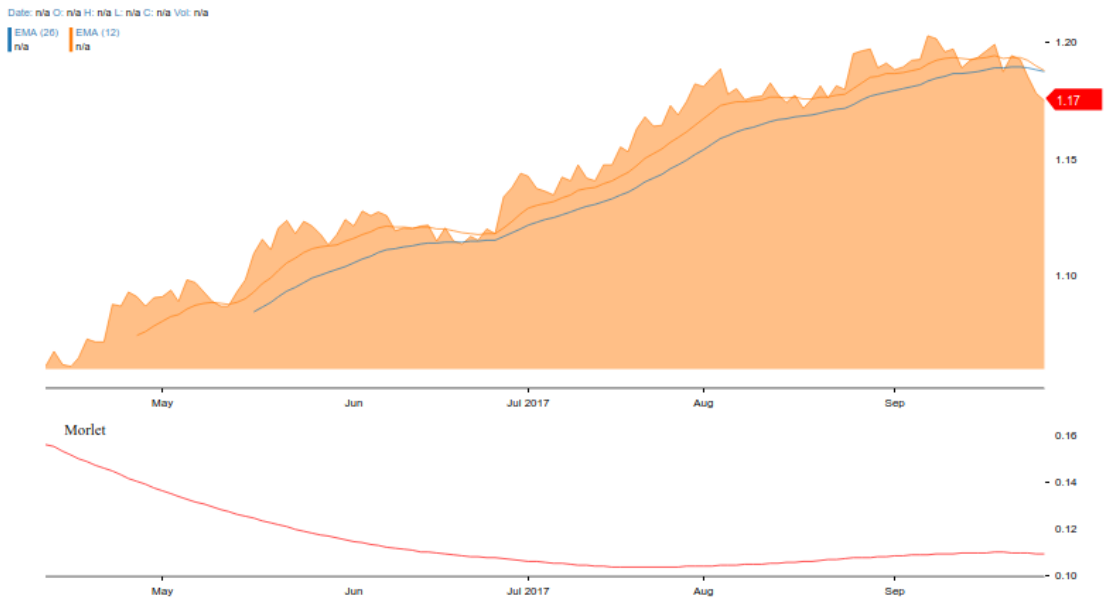


Рисунок 2.5 – Индикатор вейвлета Морле на навчальній вибірці та тестовій вибірці

На даному рисунку видно, що значення вейвле-перетворення на основі вейвлета Морле дуже погано корелює зі значен часового ряду валютних корирувань. Така повідінка прослідковується і на спектрумі вейвле-перетворення на рисунку 2.6 .

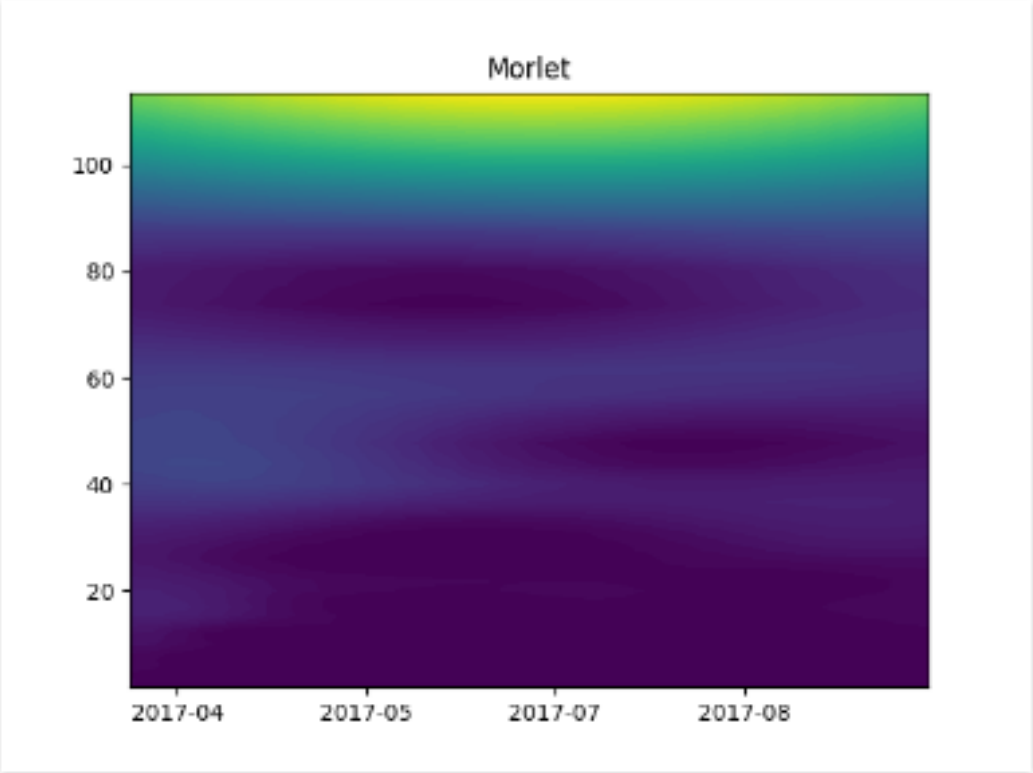


Рисунок 2.6 – Спектрум вейвлет перетворення на основі вейвлета Морле

На основі отриманих результатів можна зробити висновок, що передбачення на основі тредна цього вейвле-перетворення неможливе.

2.1.4 Аналіз вейвлета Рікера

Вейвлет рікера краще за інших виявляет імпульси в циклах хвиль Еллі-отта, тому даний вейвлет також дослідженний на молживість передбачення наступних значень валютних котирувань на основі нього.

Результати вейвлет-перетворення зображені на рисунку 2.7 .

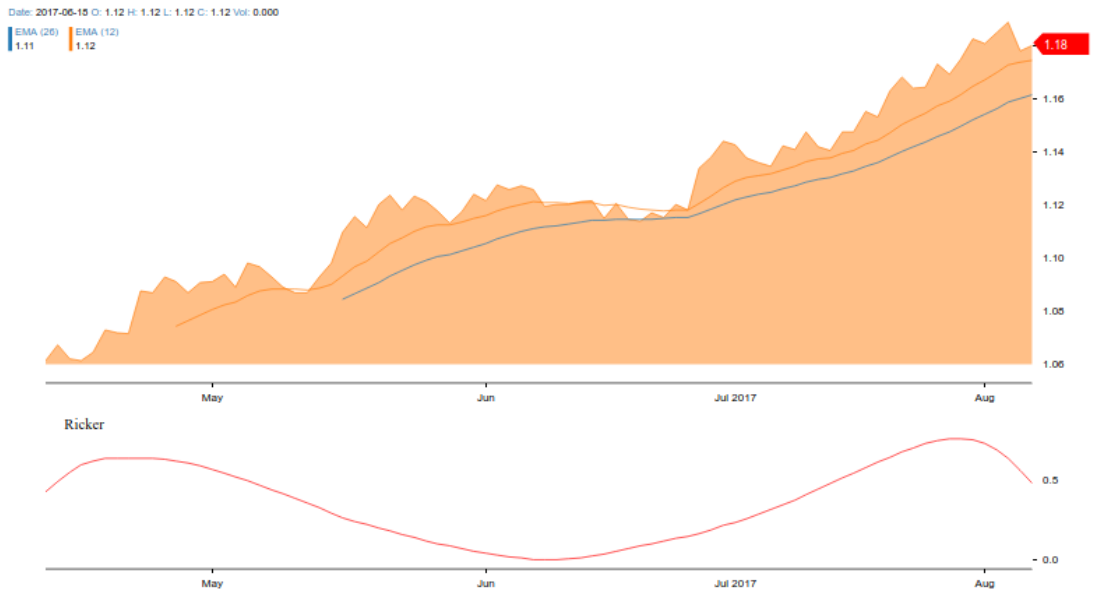


Рисунок 2.7 – Индикатор вейвлета Рікера на навчальній вибірці

На даному рисунку можна побачити, що вейвлет-перетворення на граничному інтервалі не має кореляції з графіком валютних котирувань. На наступному рисунку 2.8 можна побачити, що ситуація надалі не покращується.

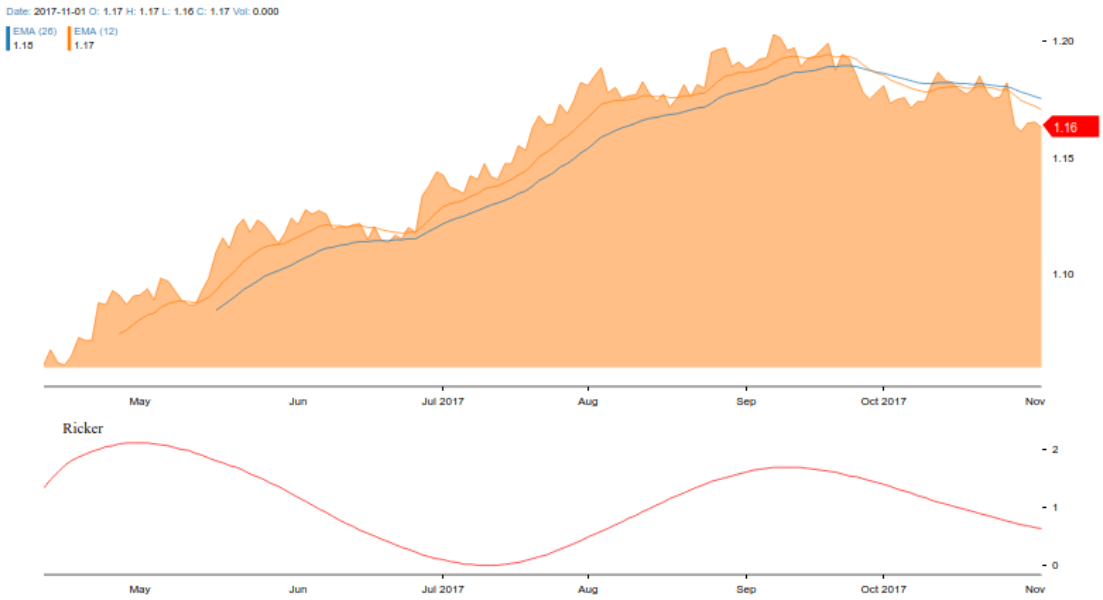


Рисунок 2.8 – Индикатор вейвлета Рікера на навчальній вибірці та тестовій вибірці

2.1.5 Аналіз вейвлета Пауля

На рисунку 2.9 зображено індикатор вейвлета Пауля. На граничному інтервалі можна побачити, що тренд йде на спад в обох графіках. В цього можна зробити висновок, що вейвлет Пауля описую поведінку часового ряду.

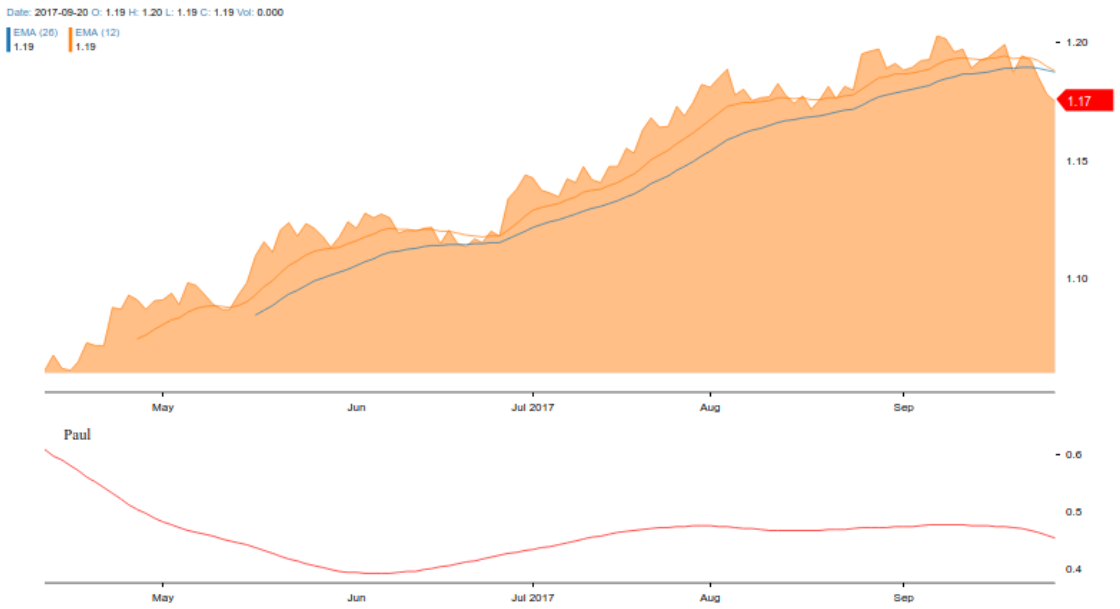


Рисунок 2.9 – Индикатор вейвлета Пауля на навчальній вибірці

На наступному рисунку 2.10 вейвлет Пауля повністю повторює поведінку валютних котирувань на тестовій вибірці. Проте швидкість зміни індикатора вейвлета Пауля меньша ніж основного графіка. На основі цього можна зробити висновок, що напрямок тренду даного індикатора можна використовувати для передбачення наступних значень валютних котирувань.

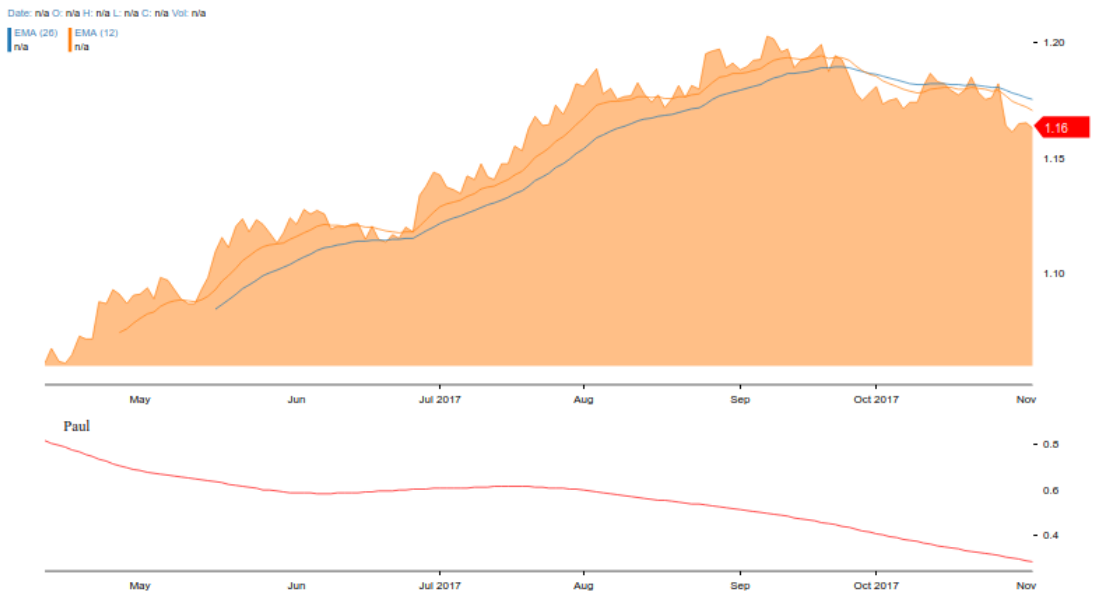


Рисунок 2.10 – Индикатор вейвлета Пауля на навчальній вибірці та тестовій вибірці

2.1.6 Аналіз індикатора фрактальної розмірності

Для аналізу обох індикаторів використовувалась вибірка для навчання. Індикатор фрактальної розмірності зображений на рисунку 2.11 .

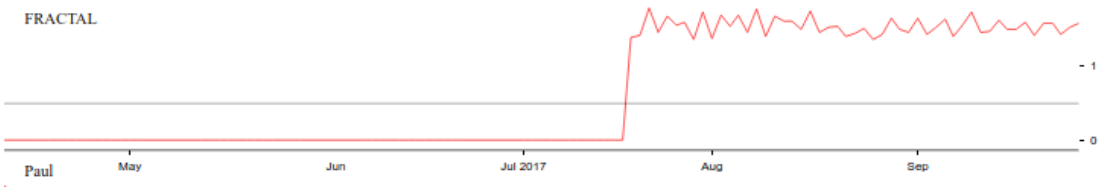


Рисунок 2.11 – Індикатор фрактальної розмірності на навчальній вибірці

Як можна побачити з рисунку, інформації недостатньо щоб прийняти якесь рішення чи побачити закономірність. Розмір вікна 70 точок надто великий для такої вибірки, проте алгорим не буду працювати на меньшій кількості точок.

Наступний крок в аналізі даного індикатора було збільшення розміру вибірки. Результати зображені на рисунку 2.12. На даному рисунку видно, на

місці ліній трендів виникають великі флуктуації. Ці результати свідчать про ідентифікацію хвиль Елліотта, оскільки вони мають фрактальну структуру.

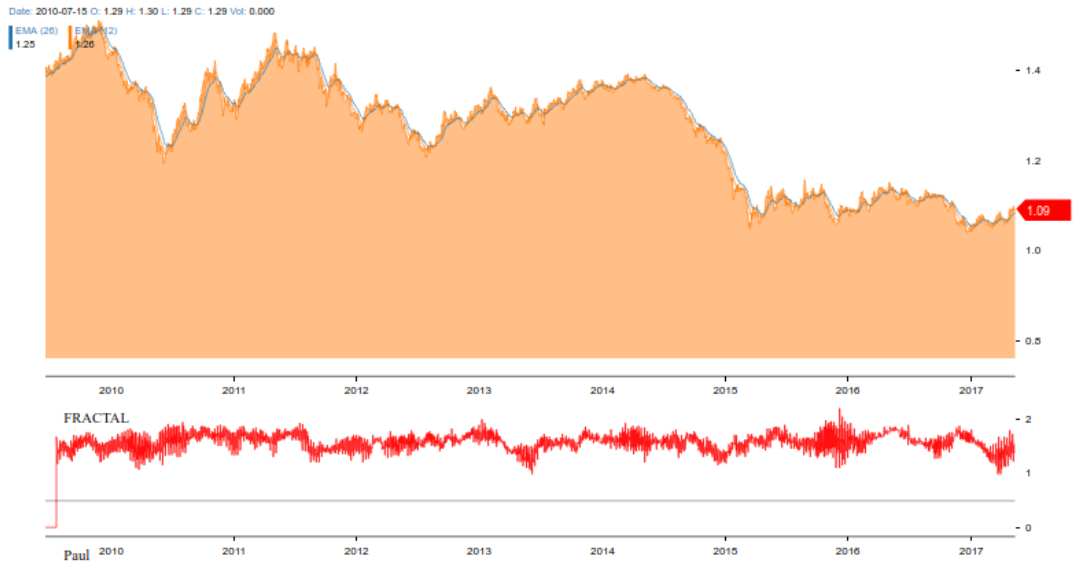


Рисунок 2.12 – Індикатор фрактальної розмірності на великій вибірці

2.1.7 Аналіз індикатора Херста

Наступним кроком в досліжені був аналіз індикатора Херста. Результати роботи індикатора можна побачити на рисунку 2.13. На рисунку можна побачити, що значення індикатора Херста слабо корелюють зі значеннями валютніх котирувань. Таким чином даний індикатор не підходить для пронозуванню для наступних значень. Проте даний індикатор добре підходить для аналізу історичних даних, осколькі виділяю інтервали трендових ліній.

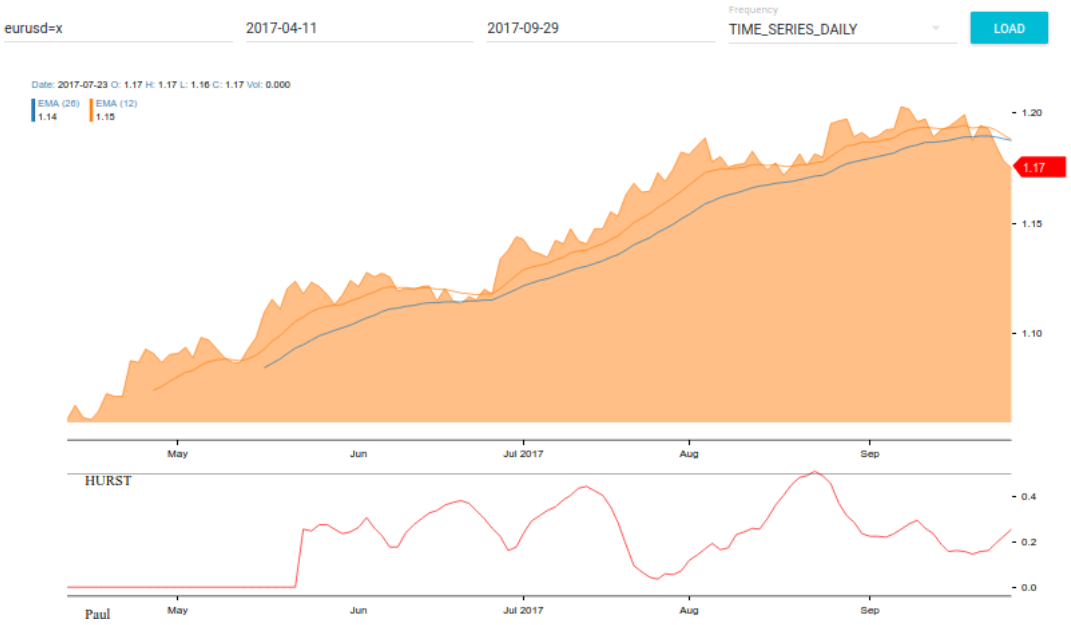


Рисунок 2.13 – Індикатор Херста на навчальній вибірці

2.1.8 Рекомендації на основі індикаторів

На основі виявлених вище закономірностей можна побудувати систему для рекомендації майбутніх операцій на біржі. Рекомендація валютних операцій на біржі буду більш точною, якщо подувати модель та передбачувати наступні значення на основі спрощенної поведінки часового ряду - на основі значень індикатора. Таким чином ми позбавляємося великої кількості шумів та працьємо з агрегованими даними на основі вейвлет-перетворень.

Аналіз наявних індикаторів виявив, що апроксимація поведінки значень часового ряду валютних котирувань найвища у індикатора на основі вейвлета Пауля.

Алгорит рекомендації валютних операцій складається з таких кроків:

- отримання значення індикатора на основі вейвлета Пауля;
- виявлення локального максимума на граничному інтервалі часового ряду індикатора;
- побудова моделі поліноміальної регресії для граничних значень індикатора;

- прогнозуєм наступного значення індикатора;
- порівняння локального максимума з прогнозованим значенням для виявлення напрямку тренда;
- приймаємо рішення на основі напрямку тренда.

Таким чином для прогозування наступних значень індикатора використана поліноміальна регресія. Модель на основі даної регресії більш точна, оскільки вона краще апроксимує нелінійну поведінку значень часового ряду.

Висновки до розділу

У цьому розділі використовувалая створена СППР для часових рядів валютних котирувань євро/американський долар за період 11-04-2017 - 04-11-2017. В цій СППР реалізовано алгоритм вейвлет-перетворення на основі чотирьох вейвлетів, індикаторів Херста та фрактальної розмірності. Для ефективного аналізу неперевних вейвлет перетворень розроблено алгоритм стискання спектруму до двухвимірного простору.

Кожний вейвлет, що розглянуто в цьому розідлі, було переведено до вигляду індикатора та проаналізовано на точність апроксимації часового ряду. В результаті виявлено найбільш точний індикатор, повідінка якого на граничних інтервалах співпадає зі поведінкою часового ряду валютних котирувань.

На основі отриманих результатів представлено алгоритм для рекомендації валютних операцій на біржі на основі індикатора вейвлета Пауля та побудови поліноміальної регресії.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

Створений програмний продукт автоматизує індентифікацію хвиль Елліотта та предбачая наступні значення валютних котирувань на основі вейвлет-претворень. Реалізовано алгоритм перетворення спектруму вейвлет-перетворення до двухвимірного графіка. Для даної СППР реадізовано веб-інтерфейс з можливістю зручного аналізу часового ряду валютних котирувать та рекомендації наступних валютних операцій.

3.1 Обґрунтування вибору платформи та мови реалізації програмного продукту

Реалізація даного програмного продукту складается з двох частин: back-end та front-end.

Серверна частина(back-end) написана за допомогою мови програмування Python. У цій мові реалізовані бібліотеки для вейвлет-перетворень, а також широкі можливості для графічного представлення часових рядів. Також у мові Python є бібліотека Flask для створення веб-сервера.

Окрім Python, для реалізації програмного продукту також розглядався набір інструментів Matlab, що дозволяє реалізовувати більшість математичних моделей.

Таблиця 3.1 – Порівняльна характеристика мов для реалізації серверної частини

Критерій	Python	Matlab
Наявність бібліотек для роботи з вейвлетами	+	+
Можливість завантаження даних з інтернету	+	-
Бібліотека для побудови графіків	+	+
Об’єм написаної документації для користувачів	Великий	Середній

Клієнтська частина(front-end) релазована за допомогою мови програмування JavaScript. У сучасній веб-розробці суттєве значення має вибір фреймворку. Тому в данній роботі використано звязку фреймворків ReactJS та Redux.

Даний підхід до розробки було порівняно з розробкою клієнткої частина на мові JavaScript без фреймворків та бібліотекою JQuery.

Таблиця 3.2 – Порівняльна характеристика підходів до реалізації клієнтської частини

Критерій	React + Redux	JS + JQuery
Кількість сучасних додаткових бібліотек	Велика	Низька
Структурованість коду	Велика	Низька
Складність масштабування проекту	Низька	Велика
Складність підтримки проекту	Низька	Велика
Складність інтеграії проекту з існюучими рішеннями	Низька	Велика

3.2 Аналіз архітектури продукту

Структура взаємозв’язків частин програмного продукту зображена на рисунку 3.1

Виділяються такі основні програмні частини проекту:

- веб-сервер;
- набір інструментів для застосування показників;
- набір інструментів для застосування вейвлет-перетворення;
- кліентська частина.

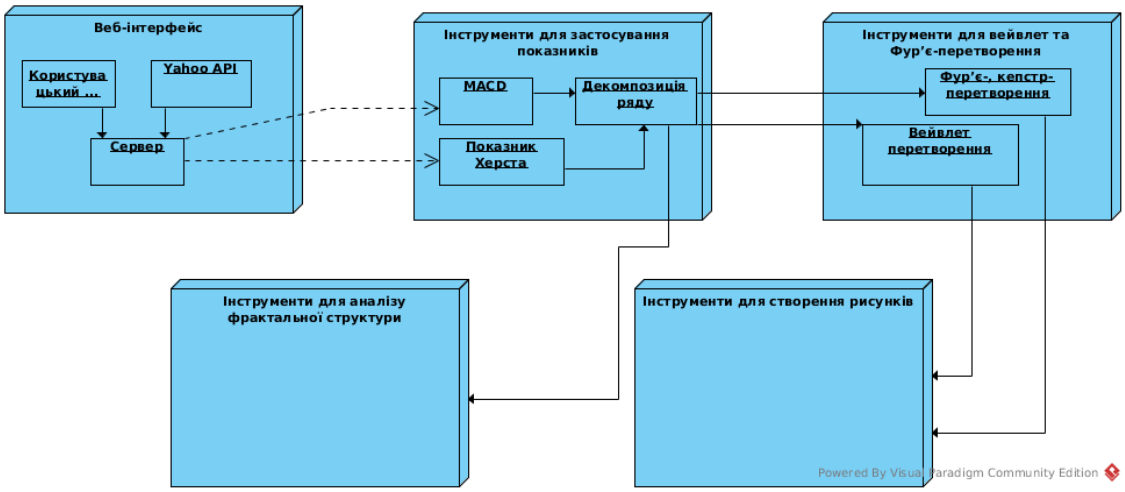


Рисунок 3.1 – Структура взаємозв’язків модулів проекту

Зазначимо, що розробка програмного продукту виконувалася таким чином, щоб виділяти часто використовувані типи функцій у окремі модулі. Таким чином був створений шар допоміжних модулів, що реалізують: роботу з веб-інтерфейсом, додаткову роботу з csv-файлами, зв’язок з API фінансових даних Alphavantage.

3.2.1 Опис програмних модулів серверної частини

Серверна частина представляє собою систему, яка завантажує дані з відкритого API, опрацьовую запити користувача викорнує розрахунки індикаторів на основі вейвлет-перетворень та інших індикаторів індикаторів.

3.2.1.1 Інструменти для застосування показників

У цьому модулі реалізовано набір інструментів для застосування показників, а саме показник Херста, показник Ляпунова, MACD та допоміжні функції. Модуль складається з наступних функцій:

- hurst(ts) - повертає показник Херста для даного ряду, розрахований для певного вікна елементів;
- moving_average(ts, moving_average_width) - розрахунок простого ковзного середнього для вибраного часового ряду ts з шириною вікна moving_average_width;
- exp_moving_average(ts, moving_average_width) - розрахунок експоненційного ковзного середнього для вибраного часового ряду ts з шириною вікна moving_average_width;
- macd(ts, width1=12, width2=26) - розрахунок MACD для часового ряду ts, width1 - ширина короткого експоненційного середнього(за замовчуванням 12), width2 - ширина довгого експоненційного середнього(за замовчуванням 26).

3.2.1.2 Інструменти для вейвлет-перетворення

Модуль відповідає за розрахунок дискретних та неперервних вейвлет-перетворень. Функції блоку наступні:

- compute_dwt(x, wavelet_name) - приймає на вхід масив чисел та назву вейвлета, виконує дикретне вейвлет перетворення, повертає масив чисел;
- compare_dwt(date, x, file_name) - приймає на вхід часовий ряд у вигляді мисивів дат, значень та назву файла куди зберігати рисунок, виконує вейвлет-перетворення з вейвлетами Добеши, Хаара та Койнфлета, зберігає у файл у вигляді порівняння вхідного ряда та нових перетворень;
- compare_fft(date, x, file_name) - приймає на вхід часовий ряд у вигляді мисивів дат, значень та назву файла куди зберігати рисунок, виконує Фур'є-перетворення та зберігає у файл у вигляді порівняння вхідного ряда та нового перетворення;
- wavelet_reserch(date, x, type, wavelet) - приймає на вхід часовий ряд у вигляді мисивів дат, значень; назву метода по якому виконується роз-

биття та назва вейвлета за допомогою якого виконується дискретне вейвлет перетворення;

- wavelet_reserch(date, x, type, wavelet) - приймає на вхід часовий ряд у вигляді мисивів дат, значень; назву метода по якому виконується розбиття та назва вейвлета за допомогою якого виконується дискретне вейвлет перетворення. Повертається перетворений масив значень та дат;
- calculate_cwt(time_scale, date, x, folder_name, wavelet_name) - приймає на вхід часовий ряд у вигляді мисивів дат, значень; масштаб часу для виведення на графіку, назва папки для виведення графіків, назва вейвлета. Виконується неперервне вейвлет перетворення. Зберігаються графіки до папки;
- cwt(data, wavelet=None, widths=None, dt=1, frequency=False) - функція-обгортка, що виконує неперервне вейвлет-перетворення. На вхід приймається часовий ряд data, клас вейвлету wavelet, масштаби widths, крок часу dt, перемикач для розрахунку по часу або по частоті frequency;
- cwt_time(data, wavelet, widths, dt) - функція, що виконує неперервне вейвлет-перетворення по часу. На вхід приймається часовий ряд data, клас вейвлету wavelet, масштаби widths, крок часу dt;
- cwt_freq(data, wavelet, widths, dt) - функція, що виконує неперервне вейвлет-перетворення по частота. На вхід приймається часовий ряд data, клас вейвлету wavelet, масштаби widths, крок часу dt;
- WaveletTransform - клас, що відповідає за вейвлет-перетворення;
- Morlet - клас, що реалізує вейвлет Морле;
- Paul - клас, що реалізує вейвлет Paul;
- DOG - клас, що реалізує вейвлет DOG;
- Ricker - клас, що реалізує вейвлет Рікера.

3.2.1.3 Отримання даних

Для програмного продукту було реалізовано два способи отримання даних:

- використання ALPHA VANTAGE API;
- використання історичних даних у попередньо завантаженому csv-форматі.

3.2.2 Опис програмних модулів кліентської частини

Кліентська частина представляє собою окремий сервер та веб-сайт, який використовує кліент для вземодіє з СППР.

Веб-сайт написаний на мові прорамування JavaScript з використаннями фреймворків React та Redux. Стилізація даного веб-сайту виконана за допомогою бібліотеки MaterialUI.

Сервер кліентської частини написаний також на мові програмування JavaScript та запущений на програмній платформі NodeJS.

3.2.2.1 Опис архітектури проекту побудованому на фрейворках React та Redux

React — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків.

Зв'язок React та Redux дозволяє реалізувати ідею розділення компонентів для презентації та зберігання. Опис різниці між компонентами знаходиться в таблиці 3.3.

Таблиця 3.3 – Порівняльна характеристика презентаційних та контейнерних компонентів

	Презентаційні компоненти	Контейнерні компоненти
Мета	як все виглядає (розмітка, стилі)	як все працює (отримання даних, оновлення стану всього додатка)
Використовує патер Redux	ні	так
Зчитування даних	зчитування даних з параметрів компонента	зчитування даних з стану всього додатка
Запис даних	визивається функція з параметрів	визиваються дії з Redux
Спосіб створення	написані вручну	зазвичай згенеровані з допомогою React Redux

Короткий опис принципів роботи Redux:

а) компоненти беруть коллбекі як властивості і викликають їх, коли відбувається UI-подія;
б) ці коллбекі створюють і відправляють дії в залежності від події;
в) редюсери обробляють дії, обчислюючи новий стан;

г) новий стан всього програми поміщається в одне сховище;
д) компоненти отримують новий стан як властивість і переотрісовивают себе при необхідності.

Схема роботи даного патерна зображена на рисунку 3.2 .

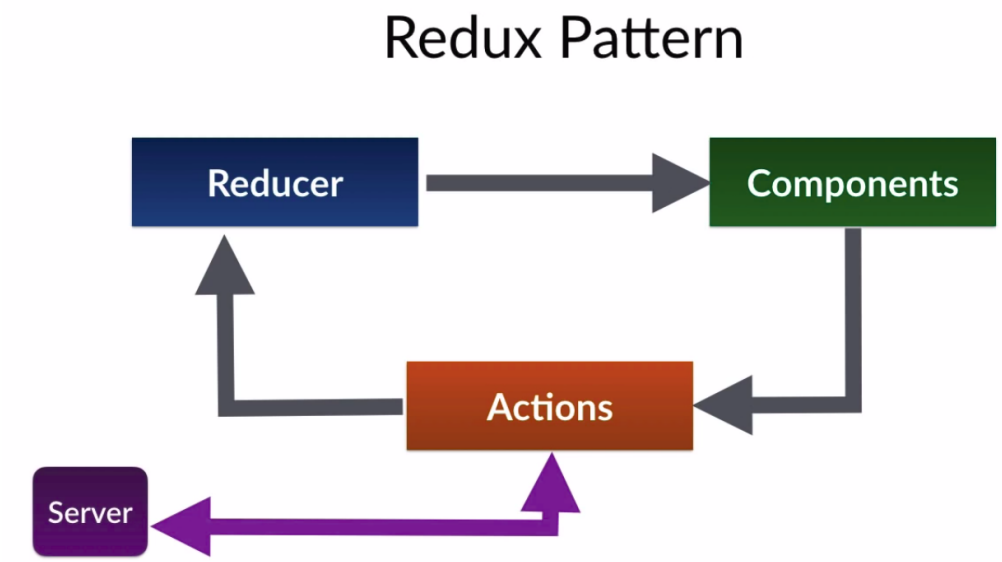


Рисунок 3.2 – Схема роботи патерну Redux

Таким чином комбінування даних підходів дозволяє створити гнучку архітектуру, яку буде зручно редагувати, підтримувати та масштабувати.

3.2.2.2 Опис бібліотеки MaterialUI

Вперше концепція Material Design була представлена компанією Google 25 червня 2014 року. Мінімалістичний і акуратний, він швидко став сучасною класикою. Мобільні додатки Google, такі як Gmail, YouTube, Google Drive і багато інших можуть допомогти вам зрозуміти, наскільки такі додатки гарні на практиці. Якщо ви плануєте слідувати цьому широко поширеній тренду і програму, що реалізує принципи Material Design це якраз те, що вам потрібно, ви можете покластися на Material-UI. Ця бібліотека представляє з себе колекцію компонентів, таких як кнопки, що випада-

ють меню, перемикачі, тулбари, і т.п. Бібліотека чітко слід керівництву по Material Design від Google і прекрасно документована. Відмінна можливість об’єднати в одному проекті модний і сучасний користувальницький інтерфейс з можливостями React.

3.3 Інструкція користувача

В данів роботі було створено СППР для прийняття рішень на часових рядах валютних котирувань на основі вейвлетної ідентифікації хвиль Елліотта.

Створенно 3 основні частини:

- інтерфейс для ознайомлення з хвилями Елліотта та вейвлет перетвореннями;
- інтерфейс для вводу параметрів для роботи СППР;
- інструменти для візуального аналізу індикаторів;
- набір спектрумів вейвлет-перетворень часового ряду;
- ремендації щодо операцій на валютній біржі.

Інтервейс для ознайомлення має в собі навчальний контент, який заменьшує рівень входу для розуміння загальної тематики(рис. 3.3).

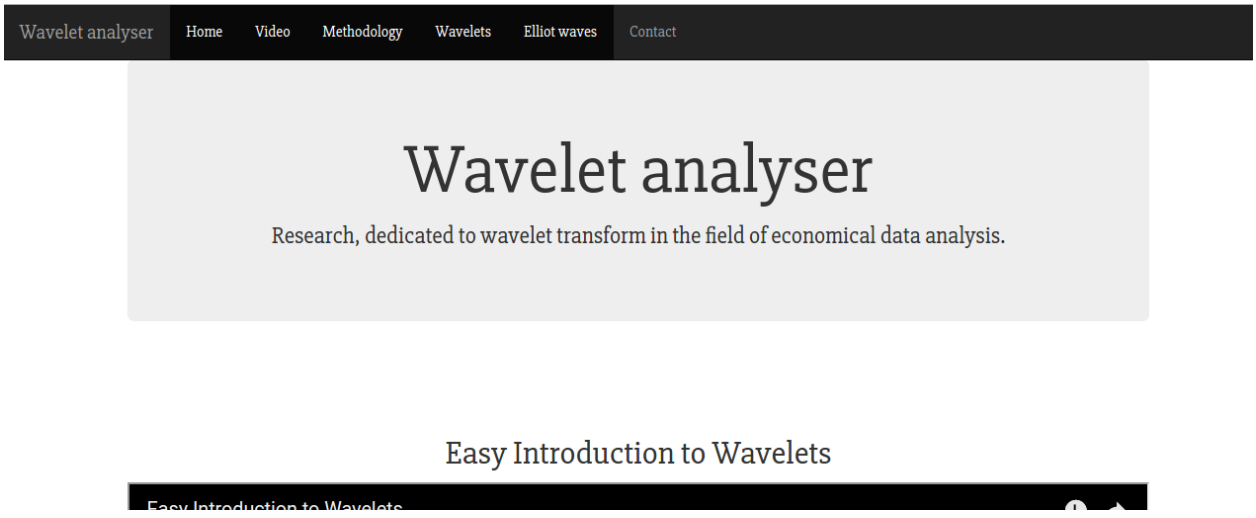


Рисунок 3.3 – Меню інтерфейса ознайомлення користувача

3.4 Інтервейс вводу параметрів

Інтервейс для вводу параметрів(рис. 3.4) має форму, яка включає в себе поля:

- валютна пара;
- інтервал на якому працює алгоритм;
- частота даних.

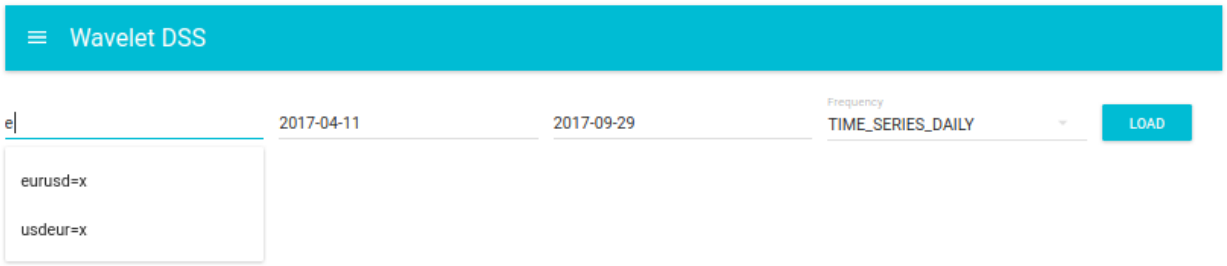


Рисунок 3.4 – Форма для вводу параметрів для СППР

3.4.1 Опис результатів роботи програми

3.4.1.1 Графік часового ряду валютних котирувань

Результати обробки часового ряду зображені на рисунку 3.5 . На даному графіку часовий ряд зображено заштрихованою областю. Дана реалізація дуже зручна тим, що можна злегкістю змінювати масштаб та значення точок графіку відображаеться в верхньому лівому кутку.

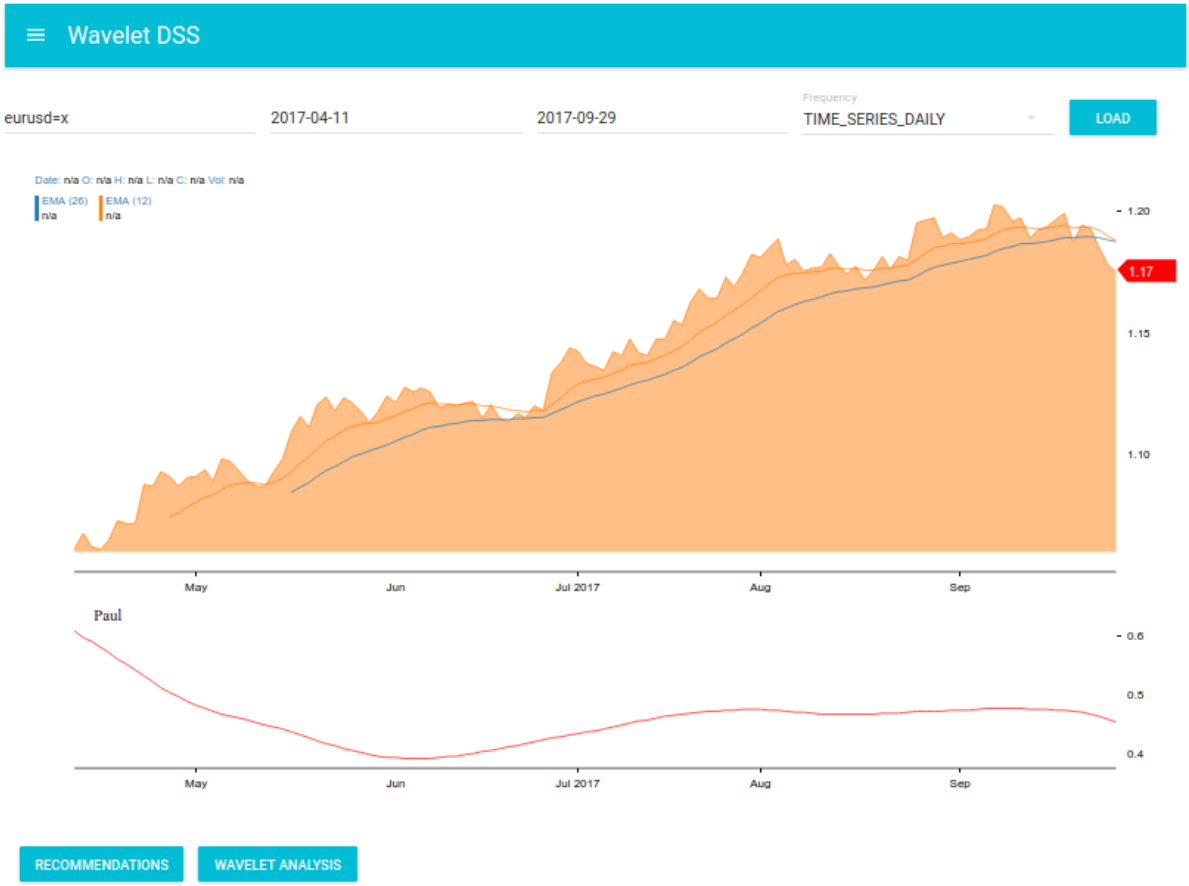


Рисунок 3.5 – Результати опрацювання часового ряду валютних котирувань

Також можна побачити, що на графку також намальовані 2 криві синього та помаранчевого кольору - це 2 індикатори EMA(12) та EMA(24).

Під основним графіком нижче розташований графік індикатора на основі вейвле-перетворення з використанням вейвлета Пауля.

3.4.1.2 Набір індикаторів для візуального аналізу валютних котирувань

Після натискання кнопки ”WAVELET ANALYSIS” до графіка з індикатором вейвлета Пауля додаються ще інші індикатори(рис. 3.6), а саме:

- індикатор вейвлета Морле;
- індикатор вейвлета Рікера;

- індикатор вейвлета DOG;
- індикатор фрактальної розмірності.

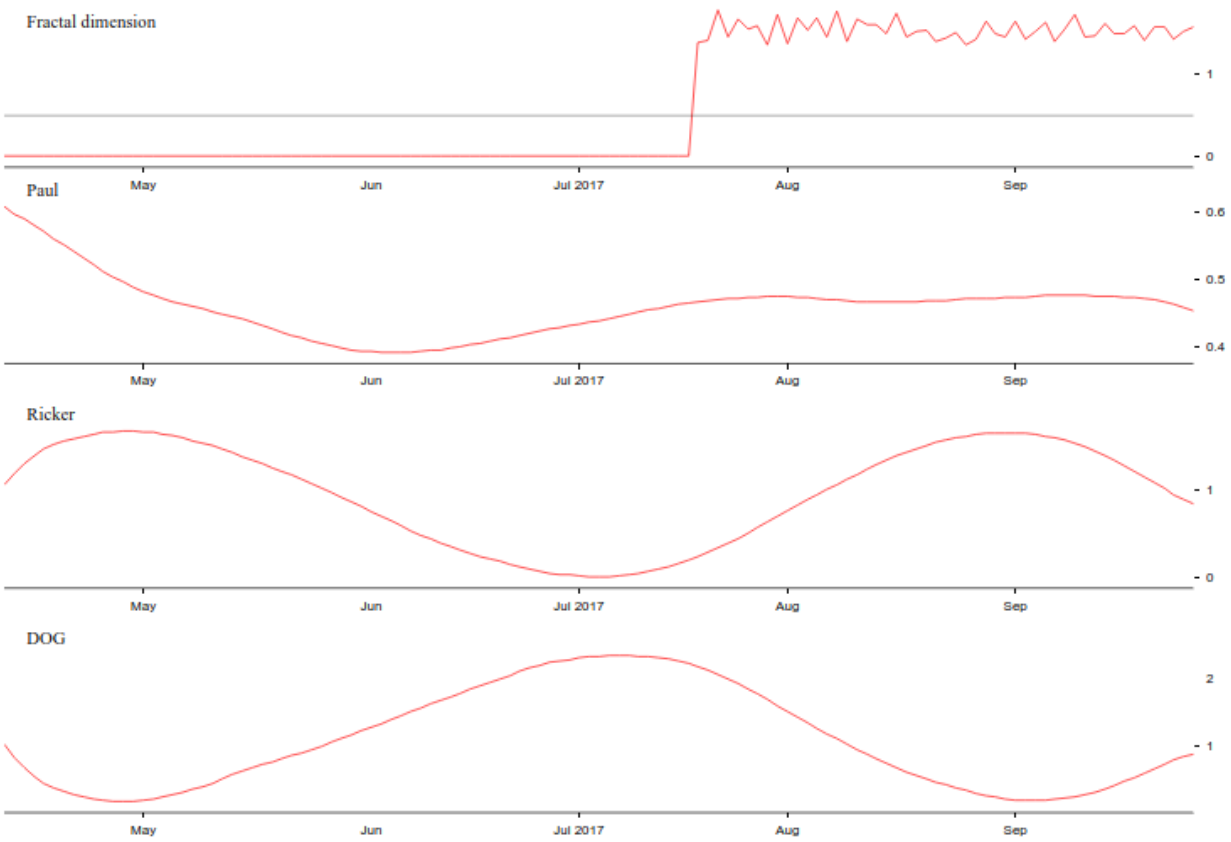


Рисунок 3.6 – Набір індикаторів для аналізу валютних котирувань

3.4.1.3 Набір спектрумів вейвлет-перетворень

Для більш детального аналізу додано набір спектрумів, які зображені на рисунку 3.7 . Цей набір спектрумів дозволяє отримати більше інформації про вейвлет перетворення, а саме провести аналіз амплітудних значень вейвле-перетворень.

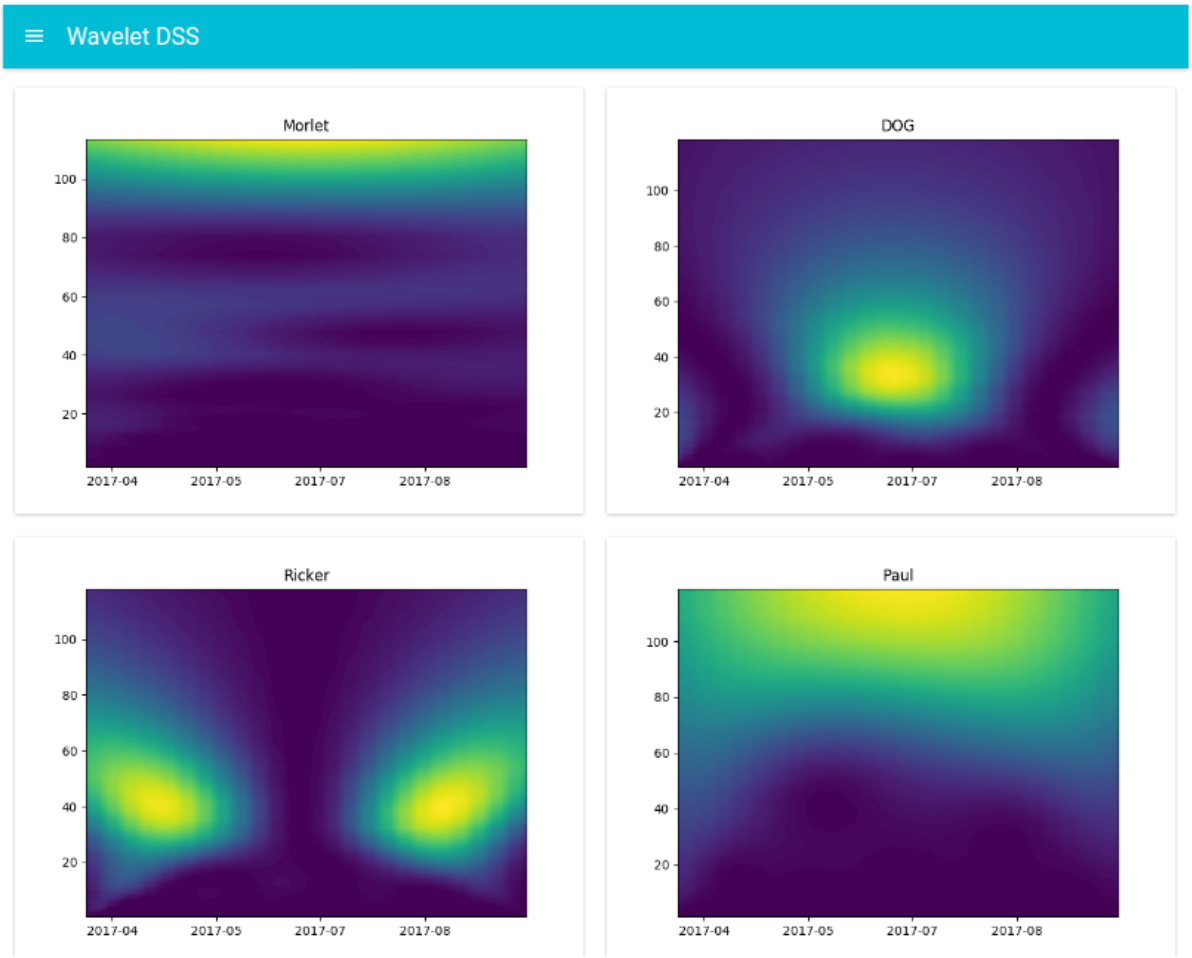


Рисунок 3.7 – Набір спектрумів вейвлет-перетворень валютних котирувань

3.4.1.4 Рекомендації щодо операцій на біржі

Функціонал рекомендацій щодо валютних операцій реалізовано в даній СППР. Теоретична частина роботи цього модуля описана у теоретичному розділі.

Для коректного прогозування наступних значень індикатора використовувалась поліноміальна регресія. Підпод відповідного степеня полінома подбирався експерементальним шляхом і дорівнює 2. Модель з даним степенем полінома була найбільш адекватна та апроксимувала історичні дані.

Логіка прогнозування та навчання реалізована за допомогою бібліотеки sklearn, де реалізовані найбільш популярні алгоритми для машинного навчання.

Приклад роботи модуля рекомендацій зображено на рисунку 3.8 .

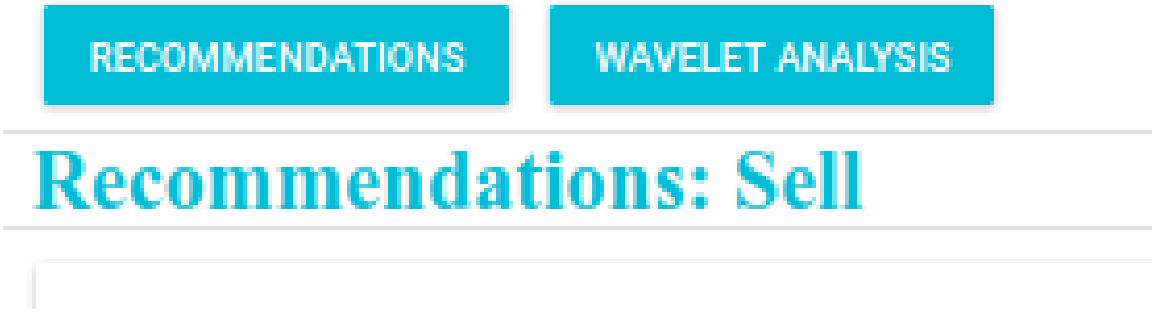


Рисунок 3.8 – Рекомендації щодо валютних операцій на біржі

Висновки до розділу

В цьому розділі описано структурну схему розробленої СППР. Програмний продукт складається з двох серверних частин:

- кліентський сервер, де зосередженна робота системи з експертом;
- сервер, де зосереджена робота з API, зберігання даних, розрахунок вейвлет-перетворень та індикаторів.

Обґрунтовано вибір мови програмування, інструментарію для отримання даних та архітектури цієї системи.

Надана коротка інструкція з експлуатації програмного продукту.

Отже, система дає можливість:

- отримувати фінансові дані;
- обробляти дані;
- виконувати вейвлет-перетворення та розраховувати індикатори на їх основі;
- рекомендувати валютні операції на основі індикаторів та поліноміальної регресії;

- зручно виводити дані в інтерфейс користувача;
- мати систему в сучасному дизайні;
- масшабувати систему у разі необхідності.

На основі аналізу отриманих результатів можна зробити висновок, що СППР загалом виконує поставлені цілі, а частоту виникнення багатьох проблемних випадків можна відносно легко зменьшити.

РОЗДІЛ 4 ОПИС СТАРТАП-ПРОЕКТУ

Опис стартап-проекту представлено в таблицях 4.1-4.22.

Таблиця 4.1 - Опис ідеї стартап-проекту

Назва проекту	СМПР для ринку цінніх паперів
Автори проекту	Слюсар Андрій Вячеславович
Коротка анотація	Система прийняття рішень для ринку цінних паперів буде використовуватися трейдерами для аналізу валютних котирувань та оптимізації виконання операцій на біржі. Система буде використовувати хвилі Елліотта та вейвлет перетворення для аналізу валютних котирувань.
Термін реалізації проекту	24 (тривалість проекту у місяцях)
Необхідні ресурси	1 програміст, 1 аналітик, ноутбук, 2000 доларів. Перелік усіх необхідних ресурсів (фінансових, матеріальних, інтелектуальних та ін.)
Опис проблеми, яку вирішує проект	Проект дозволяє трейдерам приймати рішення щодо операцій на біржі на основі набору показників для аналізу часових рядів валютних котирувань. СМПР використовує показники, які розраховуються на основі хвиль Елліотта, - це дозволяє оцінювати куди буде рухатися тренд.
Головні цілі та завдання проекту	Реалізувати систему прийняття рішень, яка буде використовувати показники на основі хвиль Елліотта та вейвлет перетворень.
Очікувані результати	Ця система допоможе трейдерам приймати рішення на біржі щодо валютних операцій на основі валютних котирувань. Використання показника на основі хвиль Елліотта приведе по покращення його якості після широкої експуатації.

Таблиця 4.2 - Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Проект дозволяє трейдерам приймати рішення щодо операцій на біржі на основі набору показників для аналізу часових рядів валютних котирувань. СППР використовує показники, які розраховуються на основі хвиль Елліотта, - це дозволяє оцінювати куди буде рухатися тренд.	використовуватися трейдерами для аналізу валютних котирувань	Автоматична система прийняття рішень на біржі
	оптимізація виконання операцій на біржі.	Автоматичне виявлення хвиль Елліотта
	дослідження валютних котирувань на основі хвиль Елліотта	Гнучка конфігурація алгоритму

Таблиця 4.3 - Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
		Технологія 1 (технологія виготовлення товару, надання послуги)	Чи вони наявні, або ж необхідно їх розробити/доробити?	Чи вони доступні авторам проекту?
1	Вейвлет перетворення	Бібліотека PyWaveletes та додані вейвлети	Наявна	Доступна, допрацьована
2	Сервер	Flask	Наявна	Доступна
3	Інтерфейс	ReactJS, JS, HTML, CSS	Наявна	Доступна
4	Алгоритми опрацювання данних	Python, NumPy, SciPy	Наявна	Доступна

Таблиця 4.4 - Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	0
2	Загальний обсяг продаж, грн/ум.од	0
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Наявність валютних котирувань
5	Специфічні вимоги до стандартизації та сертифікації	Визнання асоціацєю трейдерів
6	Середня норма рентабельності в галузі (або по ринку), %	200

Таблиця 4.5 - Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Точне прознозуввання значень та тренду валютних котирувань	Трейдери, компанії по торговлі валютою	Мінлива економіка може вплинути на якість прогнозу	Висока якість прогнозу Технічна підтримка Зручні інтерфейси

Таблиця 4.6 - Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Важко увійти у ринок	Споживачі можуть не погодитися користуватися продуктом	Збільшення реклами, випуски матеріалу для роз’яснення продукту
2	Не визнання алгоритму Елліотта	Існують вчені, які намагаються спростувати теорію Елліотта	Демонстрація роботи алгоритму

Таблиця 4.7 - Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Зацікавленість споживачів	Трейдери зацікавленні в розширенні ряду інструментів для аналізу валютних котирувань	Впровадження продукту

Таблиця 4.8 - Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції - монополія/олігополія/ монополістична/чиста	Мономолія	Покращення якості продукту Постійне оновлення, додаткові можливості для клієнта Технічна підтримка цілодобово
2. За рівнем конкурентної боротьби - локальний/національний/…	міжнародний	

Продовження таблиці 4.8

3. За галузевою ознакою - міжгалузева/ внутрішньогалузева	Внутрішньогалузева	
4. Конкуренція за видами товарів: - товарно-родова - товарно-видова - між бажаннями	Товарно-видова	
5. За характером конкурентних переваг - цінова / нецінова	Цінова	
6. За інтенсивністю - марочна/не марочна	Не марочна	

Таблиця 4.9 - Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Системи для оцінки часових рядів валютних котирувань на основі інших показників	Системи прогнозуван-ня на основі штучного інтелекту	Системи агрегації часових рядів валютних котирувань	Потреба у високій якості прогнозу	Аутсорсин-гові фірми, що займають-ся прогноза-ми
Висновки:	Можливість застосува-ння показників, які вже більш перевірені часом	Можливість входження на ринок є, але ризики занадто великі, оскільки алгоритми ще не перевірені часом	Постачальн-ик може диктувати свої умови і збільшувати ціну на вибірки	Потріб-на демонст-рація коректної роботи продук-ту	Не рентабельні-сть використання ресурсів, викоирис-тання алгоритму дешевше і надійніше

Таблиця 4.10 -Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Висока якість прогнозу	Якість прогнозу дуже велика після тестування на реальних данних
2	Постійна технічна підтримка	Швидке позбавлення проблем з боку розробника та вчасні відповіді на запитання по користуванню продуктом
3	Легке використання	Зручний інтерфейс користувача

Таблиця 4.11 - Порівняльний аналіз сильних та слабких сторін «назва проекту»

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з … (назва підприємства)						
			–3	–2	–1	0	+1	+2	+3
1	Висока якість прогнозу	4				+			
2	Постійна технічна підтримка	1					+		
3	Легке використання	2		+					

Таблиця 4.12 - SWOT- аналіз стартап-проекту

Сильні сторони: Висока точність обчислень Швидкість роботи Легкість використання Зручний інтерфейс користувача	Слабкі сторони: Невелика перевірка часом(надійність) Відсутність фінансових ресурсів Обмеженість у часі

Продовження таблиці 4.12

Можливості: Трейдери зацікавленні в розширенні спектру показників для оценки часових рядів Використання нового показника роширить покаршить результативніть торгів на біржі	Загрози: Споживачі можуть не погодитись купувати продукт Новизна показника принесе недовіру до торговлі великими сумами

Таблиця 4.13 - Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Інтеграція показника в велику платформу	Висока	1 місяць

Таблиця 4.14 - Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивні-сть конкурен-ції в сегменті	Простота входу у сегмент
1	Трейдери	Середня. Якщо показати, що якість прогнозува-ння велика	Високий	Прямих конкурен-тів на ринку немає.	Зацікав-ленність користува-чів
2	Компанії по трейдингу	Середня. Якщо показати, що використан-ня показиникак збільшить дохід компанії.	Середній	Прямих конкурен-тів на ринку немає.	Якщо в компації буде бюджет для експеремен-тування з новим показником.

Продовження таблиці 4.14

3	Комплексні платформи для прогнозування	Висока	Середній	Прямих конкурентів на ринку немає.	Поглинання показника платформою
Які цільові групи обрано: Трейдерів					

Таблиця 4.15 - Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	Регулюють-ся процентом від доходу	-	Більше 1 млрд грн/рік	Пудписка 10-20$ на місяць, чт покупка на рік за 120-260

Таблиця 4.16- Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспромож ні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Просувати свій продукт для трейдерів	Переконати трейдерів в адеватності та результатив-ності показникак	Висока якість передбачення	Стратегія спеціалізації (для трейдерів, які хочуть ефективніше оцінювати ринок)

Таблиця 4.17 - Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопроходцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Ні, трейдери вже використовують інші показники для аналізу ринку	Компанія буде забирати існуючих	Програма буде копіювати інтервейси роботи та доробляти їх, щоб інтерфейс був знайомим	Стратегія виклику лідера (атакування лідера найкращим співвідноше-нням «ціна-якість».

Таблиця 4.18 - Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Висока якість прогнозу Технічна підтримка Зручний інтерфейс	Стратегія спеціалі-зації	Висока якість прогнозуван-ня	Прогноз, валютні ринки, хвилі Елліотта

Таблиця 4.19 - Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Висока якість прогнозу	Автоматично прогнозує тренд руху ціни на валюту	Теорія Елліотта писують рух ціни валюти у вигляді хвилі, яка залежить від психічної особливості людей при інвестуванні.
2	Легке використання продукту	Зрозумілий інтерфейс	Інтерфейс став ще більш зручним
3	Легке обслуговува-ння продукту	Технічна підтримка	Якщо виникають проблемні ситуації, компанія надає технічну підтримку

Таблиця 4.20- Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>І. Товар за задумом</td><td colspan="3">Програмний продукт, який буде давати прогноз ціни та тренду валюти.</td></tr>
<tr><td rowspan="5">ІІ. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Невелика ціна
2. Технічне обслуговування</td><td>?</td><td>?</td></tr>
<tr><td colspan="3">Якість: висока якіть проботи навіть з гривнею у нестабільний час</td></tr>
<tr><td colspan="3">Програма, яку можно завантажити з сайту чи інтегрувати в систему</td></tr>
<tr><td colspan="3">Марка: ТрейдЕлліотт</td></tr>
<tr><td rowspan="2">ІІІ. Товар із підкріплення м</td><td colspan="3">Після продажу: в перспективі додавання функціоналу для прийняття автоматично рішень щодо операцій на ринку.</td></tr>
<tr><td colspan="3">До продажу: прогнозування тренду валюти</td></tr>
</table>

Таблиця 4.21 - Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Купівля ліцензії трендерами на місяць чи інтреграція в систему	Постачальник товару – офіційний сайт, на якому можна оплатити ліцензію і підписку, а також отримати сам продукт.	Канал нульового рівня	Офіційний сайт

Таблиця 4.22 - Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Необхідно дати зрозуміти потенційному клієнту якість прогнозування	Інтернет, e-mail	Контент-маркетинг, публікації у наукових та фінансових журналах, email-маркетинг безпосередньо трейдерам	Привернути увагу, дати можливість спробувати даний продукт.	Контекстна реклама, велика кількість публікацій у спеціалізованих виданнях.

Висновки до розділу

У цьому розділі розглянута модель стартап-проекту системи прийняття рішень на основі вейвлетної ідентифікації хвиль Елліотта. В результаті роботи

представлениа модель побудови та аналіз стартап-проекту. На основі побудованої моделі продукт можна надавати інвестору для розгляду.

ВИСНОВКИ ПО РОБОТІ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

У роботі розроблено підхід до ефективного аналізу часових рядів валютних котирувань за допомогою розроблених індикаторів на основі вейвлет-перетворення.

На основі цього підходу розроблено Систему підтримки та прийняття рішень для вейвлетної ідентифікації хвиль Елліотта. Вона має сучасний інтерфейс та архітектуру програмного забезпечення.

Для подальшого вдосконалення результатів можна запропонувати наступні напрямки розвитку:

- навчання нейронних мереж для автоматичного виявлення хвиль Елліотта на основі розробленого алгоритму для обробки часових рядів;
- розробити індикатор на основі фрактальної розмірності та методів машинного навчання;
- побудова більш складної моделі на основі розроблених індикаторів та нейронних мереж.

ПЕРЕЛІК ПОСИЛАНЬ

1. A. J. Frost Elliott Wave Principle: Key to Market Behavior. / A. J. Frost, R. R. Prechter — Delhi: Elliott Wave International, 2005. — 190 p.
2. C. Tan Financial Time Series Forecasting Using Improved Wavelet Neural Network: дис. ... канд. комп. наук: 6.050101. / Chang Tan — Arhus, Denmark, 2009. — 113 p.
3. Джозеф Т. Упрощенный Анализ Волны Эллиота. / Джозеф Т. — Санкт—Петербург: Литера, 2012. — 80 с.
4. S. B. Achelis Technical Analysis from A to Z. / S. B. Achelis — Probus: Probus Pub, 1995. — 80 p.
5. Воробьев В.И. Теория и практика вейвлет-преобразования. / Воробьев В.И., Грибунин В.Г. — СПб.: ВУС, 1999. — 364 с.
6. Мэрфи Джон Дж. Технический анализ фьючерсных рынков: теория и практика. / Мэрфи Джон Дж. — Москва: Диаграмма, 2000. — 281 с.
7. Добеши И. Десять лекций по вейвлетам. / Добеши И. — Ижевск: РХД, 2001. — 464 с.
8. Малла С. Вэйвлеты в обработке сигналов. / Малла С. — М.: Мир, 2005. — 672 с.
9. Смоленцев Н.К. Введение в теорию вейвлетов. / Смоленцев Н.К. — Ижевск: РХД, 2010. — 292 с.
10. Юэн Ч. Микро-процессорные системы и их применение при обработке сигналов. / Юэн Ч., Бичем К., Робинсон Дж. — М: Радио и связь, 1986. — 296 с.
11. Харкевич А.А. Спектры и анализ. / Харкевич А.А. — М.: Физматгиз, 1963. — 432 с.
12. Mallat S. A theory for multiresolutional signal decomposition: the wavelet representation. / Mallat S. — Paris: N7, 1989. — 693 p.
13. D.G. Childers. The Cepstrum: A Guide to Processing. / D.G. Childers, D.P. Skinner, R.C. Kemerait. — Boston: The MIT Press, 1977. — 1443 p.
14. Отнес Р. Прикладной анализ временных рядов. / Отнес Р., Эноксон Л. — М.: Мир, 1982. — 428 с.

15. Эрлих А. Технический анализ товарных и финансовых рынков. / Эрлих А. — М.: ИНФРА, 1996. — 566 с.
16. Калуш Ю. А. Показатель Хёрста и его скрытые свойства. / Калуш Ю. А., Логинов В. М. // Сиб. журн. индустр. матем.. — 2002. — №5:4. — С. 29—37.
17. Короновский А. А. Непрерывный вейвлетный анализ и его приложения. / Короновский А. А., Храмов А.Е. — Москва: Физматлит, 2003. — 158 с.
18. H. J. Nussbaumer Fast Fourier Transform and Convolution Algorithms. / H. J. Nussbaumer — London: Springer, 1982. — 240 p.
19. P. Bloomfield Fourier Analysis of Time Series: An Introduction. / P. Bloomfield — Boston: The MIT Press, 2014. — 288 p.
20. B. Mandelbrot Fractals in Petroleum Geology and Earth Processes. / B. Mandelbrot — New York: P.R. La Pointe, 1969. — 969 p.
21. А.Н. Ширяев Вероятность. / А.Н. Ширяев — Москва: Физматлит, 2012. — 520 с.
22. Фрактальна розмірність [Електронний ресурс] / Вікіпедія – вільна енциклопедія. – Режим доступу https://uk.wikipedia.org/wiki/
23. Витоки вейвлет-перетворення. Основи вейвлет-перетворення. Базисні функції вейвлет-перетворення.Властивості вейвлет-перетворення [Електронний ресурс] / Цифрова обробка сигналів кафедра АЕІ. – Режим доступу http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=21619
24. Властивості вейвлет-перетворення [Електронний ресурс] / Ігор Широков. – Режим доступу http://ukrdoc.com.ua/text/35306/index-3.html
25. Основи вейвлет-перетворення сигналів [Електронний ресурс] / Ігор Широков. – Режим доступу http://ukrdoc.com.ua/text/35306/index-2.html
26. Фінансовий практикум [Електронний ресурс] / Новосибірський державний технічний університет. – Режим доступу http://ukrdoc.com.ua/text/49476/index-1.html

ДОДАТОК А ІЛЮСТРАТИВНІ МАТЕРІАЛИ ДОПОВІДІ

Система прийняття рішень на основі вейвлетної ідентифікації хвиль Елліотта

Слюсар Андрій Вячеславович
КА-62м

1

Вступ

- Метою цієї роботи є побудова системи прийняття рішень для часових рядів валютних котирувань. Для побудови системи проаналізовані результати вейвлет перетворень, показників Херста та фрактальної розмірності. На основі виявлених закономірностей побудовані нові індикатори та система прийняття рішень.
- Результати цієї роботи рекомендовано використовувати для аналізу часових рядів, історичних даних та для прийняття рішень відносно валютних операцій на валютній біржі.

2

Структура дослідження

Об’єктом дослідження є часові ряди валютних котирувань на біржах.

Предметом дослідження є:

- неперервне вейвлет-перетворення хвиль Елліотта;
- система прийняття рішень на основі індикаторів технічного аналізу;
- показники Херста та фрактальної розмірності.

Метою цієї роботи є ідентифікація хвиль Елліотта за допомогою-вейвлет перетворень та створення на цій основі системи прийняття рішень

3

Постановка задачі

- Провести дослідження вейвлет-перетворень часових рядів валютних котирувань;
- Розробити індентифікатори для часових рядів на основі вейвлет перетворень;
- Створити систему прийняття рішень щодо валютних операцій на біржі.

4

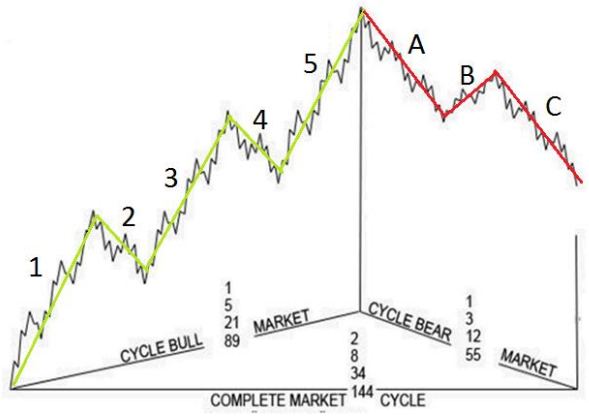
Хвилі Елліотта

Відомості про хвилі Елліотта:

- Фондовий ринок підпорядковується повторюваному ритму 5-3 хвиль зростання (імпульсні), три хвилі падіння (корекційні);
- Залежність між ланками хвиль пов'язана з рівнями Фібоначчі;
- Хвилі Елліотта мають фрактальну структуру;
- Основні моделі 5-3 лишаються постійними, хоча проміжок їх тривалості може змінюватися.

5

Приклад повного ринкового циклу хвилі Елліотта



Зеленим кольором (1-5) виділена імпульсна ланка хвилі, червоним (A-C) - корекційна

6

Вейвлет-перетворення

Вейвлет (англ. Wavelet - невелика хвиля) - математична функція, що дозволяє аналізувати різні частотні компоненти даних. Графік функції виглядає як хвилеподібні коливання з амплітудою, що зменшується до нуля далеко від початку координат.

Усі вейвлет-перетворення розглядають функцію у термінах коливань, локалізованих за часом і частотою.

Існують такі підходи до визначення вейвлета: через масштабний фільтр, масштабуючу функцію, вейвлет-функцію;

Вейвлет-перетворення зазвичай поділяють на дискретне вейвлет-перетворення (ДВП) і неперервне вейвлет-перетворення (НВП).

7

Вейвлет-перетворення

Інтегральне вейвлет перетворення - це вейвлет перетворення, що визначается як:

$$[W_\psi f](a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} \overline{\psi\left(\frac{x-b}{a}\right)} f(x)dx$$

,де ψ - це материнський вейвлет.

8

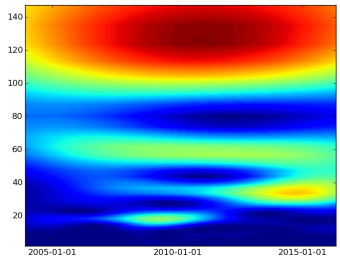
Вейвлети

Гаусові: – Першого порядка або WAVE-вейвлет – Другого порядка або MHAT-вейвлет «мексиканське капелюх» – n -порядка	$-t\exp(-t^2/2)$ $(1-t^2)\exp(-t^2/2)$ $(-1)^n\frac{d^n}{dt^n}[\exp(-t^2/2)]$
DOG-difference of gaussians	$e^{-t^2 2}-0{,}5e^{-t^2/8}$
LP-Littlewood&Paley	$(\pi t)^{-1}(\sin 2\pi t-\sin \pi t)$
Морле (Morlet)	$e^{i\omega_0 t}e^{-t^2/2}$
Пауля (Paul) – чим більше n, тим більше нульових моментів має вейвлет	$\Gamma(n+1)\frac{i^n}{(1-n)^{n+1}}$

9

Силовий спектр

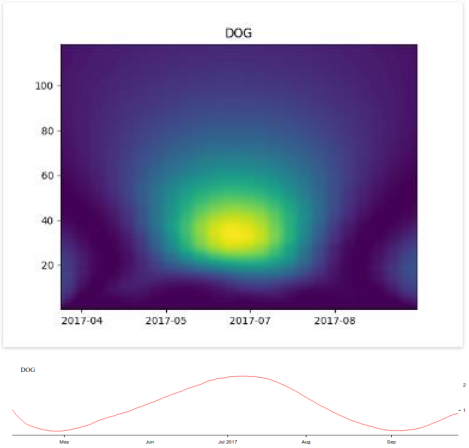
Силовий спектр – це різновид графіку, який показує амплітуду та частоту виявлених функцій в залежності від часу (корисний при використанні неперервного вейвлет-перетворення).



Приклад спектру: по осі X йде часова шкала, по осі Y – шкала періодів, колір визначає величину амплітуди

10

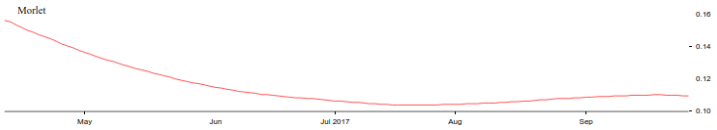
Індикатор на основі вейвлет-перетворення



Приклад перетворення спектруму вейвле-перетворення на двохвимірний індикатор

11

Порівняння створених індикаторів



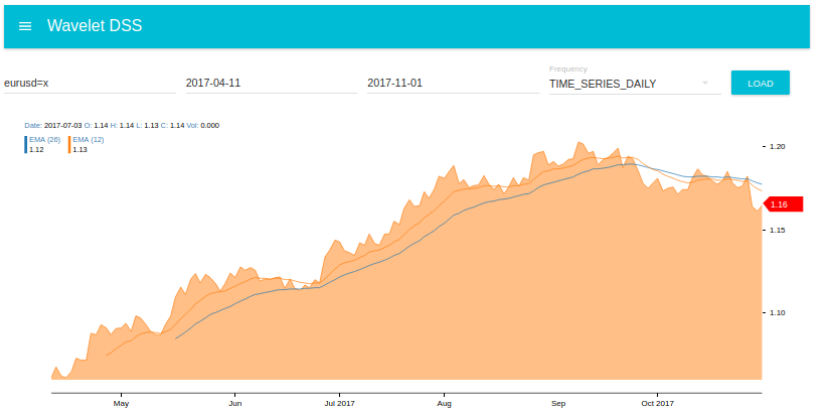
Індикатор на основі вейвлета Морле

Індикатор на основі вейлета Пауля

З наведених вище рисунків видно, що останній індикатор є більш чутливим до змін часового ряду. На основі дослідження індикаторів вейлет-перетворень за допомогою парних порівнянь виявлено, що вейвлет Пауля краще за інших описує поведінку часового ряду валютних котирувань.

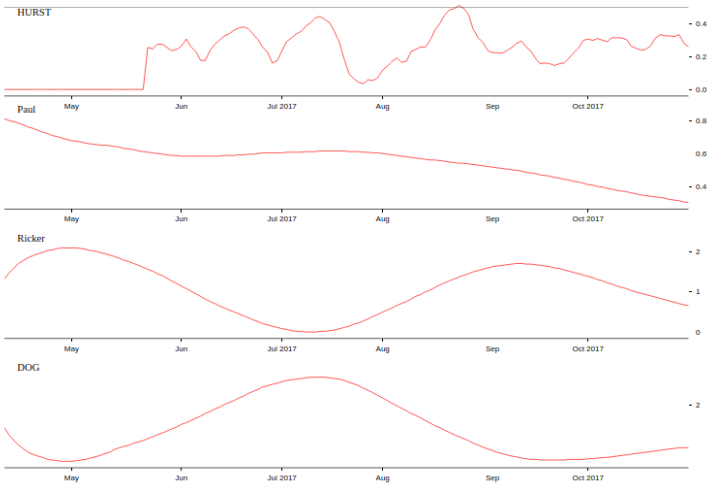
12

Інтерфейс СПР



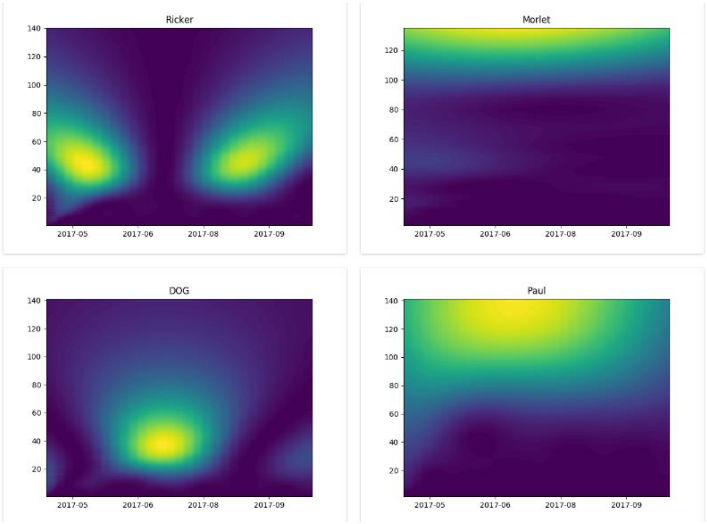
13

Індикатори в СПР



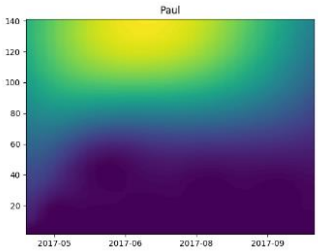
14

Вейвлет-перетворення в СПР



15

Рекомендації в СПР



RECOMMENDATIONS WAVELET ANALYSIS

Recommendations: Sell

16

Технології для створення СПР

- Система побудована на архітектурі «клієнт-сервер»;
- Цей підхід дозволяє розділяти додаток на дві функціональні частини – це додає гнучкості системи для розробки та підтримки;
- Частина «клієнта» розроблена за допомогою останніх технологій в frontend розробці. Були використані технології React, Redux, JavaScript, MaterialUI;
- Сервер було розроблено за допомогою мови програмування Python, сервера Flask. Всі підрахунки проходять на стороні сервера(індикатори, вейвлет-перетворення, тощо).

17

Дякую за увагу!

18

ДОДАТОК Б ЛІСТИНГ КОДУ

csvretriever.py

```
import pandas as pd
import time
import datetime
import os


def get_historical_gdp():
    df =
    pd.read_csv(r'../../data/GDP.csv')
    date = df['DATE']
    new_date = []
    for element in date:

    new_date.append(datetime.datetime.strptime(element,
    '%Y-%m-%d'))
    date = new_date
    value = df['VALUE']
    return date, value


def
    get_historical_quotes(start_date=datetime.datetime(1990,
    1, 2),

    end_date=datetime.datetime(2016, 1,
    1),
    csv_path='../../data/usdgbp1990.csv'):
    df = pd.read_csv(csv_path)
    date = df['DATE'].values
    value = df['VALUE'].values

    return trim(date, value,
    start_date, end_date)

    # date, x =
    get_historical_quotes(start_date=datetime.da
    1, 2),
    end_date=datetime.datetime(2009, 1,
    2))


def trim(date, value, start_date,
    end_date):
    new_date = []
    for element in date:

    new_date.append(datetime.datetime.strptime(el
    '%Y-%m-%d'))
    date = new_date
    # print(len(value))
    start_cut = date.index(start_date)
    end_cut = date.index(end_date)
    date = date[start_cut:end_cut]
    value = value[start_cut:end_cut]

    return date, value


def showPlot(date, data, file_name):
    fig, ax = plt.subplots()
    ax.plot(date, data)
    fig.savefig(file_name)
    plt.close(fig)


# def generate_elliot_waves(folder):

common_folder = 'static/results/'
folder_name = 'elliot/'
```

elliot.py

```
import numpy as np
import matplotlib.pyplot as plt
import os
import pywt
import datetime

elliot_waves = []
wavelets = pywt.families()
```

```
def trend(data):
    return 0 * np.arange(len(data))


def print_wave(data, file_name):
    date = np.arange(len(data))
    return showPlot(date, data,
    file_name)


def print_fft(data, file_name):
    A = np.fft.fft(data)
    frrAbs = np.abs(A)
    return print_wave(frrAbs, file_name)


def get_filter(arr):
    return list([x for x in arr])


def print_wavelet(data, wavelet_name,
    filter, file_name):
    result = pywt.dwt(data,
    wavelet_name)
    if filter == 'high':
        return print_wave(result[0],
    file_name)
    elif filter == 'low':
        return print_wave(result[1],
    file_name)


def build_elliot_waves(path):
    x = [1, 3, 2, 4]
    z = [1, 3, 2, 4, 3, 5]
    # print('lol1',
    os.path.abspath(path))
    if not
    os.path.exists(os.path.abspath(path)):

    os.makedirs(os.path.abspath(path))
    minus_x = np.subtract(np.max(x), x)
    minus_z = np.subtract(np.max(z), z)

    # print_wave(x, trend, path +
    'x.png')
    elliot_waves.append('x')
    # print_wave(z, trend, path +
    'z.png')
    elliot_waves.append('z')

    # print_wave(minus_x, trend, path +
    'minus_x.png')
    elliot_waves.append('minus_x')
    # print_wave(minus_z, trend, path +
    'minus_z.png')
    elliot_waves.append('minus_z')

    # print_fft(z, trend, 'fft.png')
    # print_wavelet(z, trend, 'coif1',
    'high', 'db.png')


#
build_elliot_waves('static/results/elliot/')


def generate_elliot_waves(scale=4):
    import random
    w1 = np.zeros(6)
    w2 = np.zeros(4)

    w1[1] = int(random.randint(3,
    scale))
    # print(w1[1])
    w1[2] = int(random.randint(1, w1[1]
    - 1))
    valid = True
    while valid:
        temp = int(random.randint(1,
    3)) * int(random.randint(1, scale))
        # print(w1[1], w1[2], temp)
        if w1[1] >= w1[2] + temp:
            continue
        w1[3] =
    int(random.randint(w1[1], w1[2] +
    temp))
        if (w1[3] - w1[2]) > w1[1]:
            valid = False
    w1[4] = int(random.randint(w1[1],
    w1[3]))
    w1[5] = w1[4] + w1[1]

    # print(w1)
```

```
        w2[0] = w1[5]
        valid = True
        while valid:
            try:
                w2[3] =
        int(random.randint(w1[4] + 1, w1[5]
        - 2))
                w2[2] =
        int(random.randint(w2[3] + 1, w1[5]
        - 1))
                w2[1] =
        int(random.randint(w2[3] + 1, w2[2]
        - 1))
                valid = False
            except Exception:
                _ = 1
        # print(w1, w2)
        result = list(w1) + list(w2)[1:]
        # print(result)
        proliferated_result = []  #
        multiply number of points by
        step_number
        step_number = 20
        for i in range(1, len(result)):
            for j in range(0, step_number):

        proliferated_result.append(result[i
        - 1] + (float(j) * (result[i] -
        result[i - 1])) / step_number)
        result = proliferated_result
        # print(result)
        result_m =
        np.subtract(np.max(result), result)
        return result, result_m


def
    generate_elliot_waves_wrapper(scale=4):
    print("generate_elliot_waves_wrapper")
    value, _ =
    generate_elliot_waves(scale)
    base = datetime.datetime(2001, 1, 1)
    date = [base +
    datetime.timedelta(days=x) for x in
    range(0, len(value))]
    # print("work", len(date),
    len(value))
    date = list(date)
    print(date)
    value = np.array(value)
    return date, value

    # z = [ 0.,  3.,  2.,  6.,  5.,  8.]
    # mx = [10, 11, 9]
    # arr = generate_elliot_waves(50)
    # for i in range(1,10):
    #     arr =
    generate_elliot_waves(50)
    #     print_wave(arr[0],
    'ElliotTest/Rising/zx'+str(i)+'.png')
    #
    print_fft(arr[0],'ElliotTest/Rising/z_fft_'+s
    #
    print_wavelet(arr[0],'coif4','low','ElliotTes
    #     print_wave(arr[1],
    'ElliotTest/Falling/zx'+str(i)+'.png')
    #
    print_fft(arr[1],'ElliotTest/Falling/z_fft_'+
    #
    print_wavelet(arr[1],'coif4','low','ElliotTes
    # print_wave(z, 'zx.png')


    # def generate_elliot_wave
```

fractal.py

```
import nolds
import numpy as np

rwalk =
    np.cumsum(np.random.random(1000))

print("fractal␣
    {}".format(nolds.dfa(rwalk)))
print("Lup␣
    {}".format(nolds.lyap_e(rwalk)))
print("Lup␣
    {}".format(nolds.lyap_r(rwalk)))
print("Hurst␣
    {}".format(nolds.hurst_rs(rwalk)))
```

prediction.py

```
import numpy as np
from sklearn.linear_model import
    LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import
    PolynomialFeatures


def predictNextValue(indicator,
    degree=2):
    indexes = np.arange(len(indicator))
    model = Pipeline([('poly',
    PolynomialFeatures(degree)),
                      ('linear',
    LinearRegression(fit_intercept=False))])

    model = model.fit(indexes[:,
    np.newaxis], indicator)
    return model.predict(len(indicator))
```

preprocessing.py

```
import os
import urllib.error
import urllib.parse
import urllib.request
from datetime import datetime

import numpy as np


def generatePath(stock, frequency):
    return
    os.path.dirname(os.path.abspath(__file__))
    + "/cache/" + stock + frequency +
    ".csv"


def checkExistingCache(path):
    return os.path.exists(path)


def loadStock(stock, frequency,
    withCache=True):
    path = generatePath(stock,
    frequency)
    if checkExistingCache(path) and
    withCache:
        return loadFormCache(path)
    else:
        return fetchStockData(stock,
    frequency, withCache)


def saveStockDataCache(path, stockFile):
    file = open(path, "w")
    file.writelines(stockFile)
    file.close()


def fetchStockData(stock, frequency,
    withCache):
    stockFile = []
    try:
        print('Currently␣Pulling',
    stock)
        urlToVisit =
'https://www.alphavantage.co/query?function=
    + frequency + '&' \
                    + 'symbol=' +
    stock + '&' \
                    +
    'outputsize=full' + '&' \
                    +
    'apikey=EAMPC2LUJ6VV0KEN' + '&' \
                    + 'datatype=csv'
        print('URL', urlToVisit)
        try:
            sourceCode =
    urllib.request.urlopen(urlToVisit).read().de
            splitSource =
    sourceCode.split('\n')

            # remove
    headerhttps://www.alphavantage.co/query?func
            del splitSource[0]
            del splitSource[-1]

            for eachLine in splitSource:
```

```
                splitLine =
    eachLine.split(',')
52              if len(splitLine) == 6:
                    if
    float(splitLine[1]) != 0:

    stockFile.append(eachLine)
        except Exception as e:
            print(str(e), 'failed␣to␣
    organize␣pulled␣data.')
57      except Exception as e:
        print(str(e), 'failed␣to␣pull␣
    pricing␣data')

    if len(stock) > 0 and withCache:
                                        97
    saveStockDataCache(generatePath(stock,
    type), stockFile)
62
    return stockFile


def loadFormCache(path):                102
67      file = open(path, "r")
    lines = file.readlines()
    file.close()

    return lines
72


# start - direction = -1
# end   - direction = 1
def findValidDateIndex(startDate,
    dateRange, direction):
77      result = -1
    while result < 0:
        try:
            result =
    dateRange.index(startDate)
        except ValueError:
            startDate =
82      startDate.replace(day=(startDate.day
    + direction))                       112
            result = -1

    return result

87
```

```
def trim(dates, prices, start_date,
    end_date):
    start_cut =
    findValidDateIndex(start_date,
    dates, 1)
    end_cut =
    findValidDateIndex(end_date, dates,
    -1)

92  updated_values = []
    dates = dates[start_cut:end_cut]
    for value in prices:

    updated_values.append(value[start_cut:end_cut
97  return (dates, *updated_values)


def prepareData(stock, frequency,
    startDate, endDate):
    stockFile = loadStock(stock,
    frequency)
102 stockFile.reverse()  # depends on
    API
    types = {
        'names': ('timestamp', 'open',
    'high', 'low', 'close', 'volume'),
        'formats': ('S10', 'f', 'f',
    'f', 'f', 'i')
    }
107 timestamp, open, high, low, close,
    volume = \
        np.loadtxt(stockFile,
    delimiter=',', unpack=True,
    dtype=types)
    timestamp =
    [datetime.strptime(str(x.decode('ascii')),
    '%Y-%m-%d') for x in timestamp]

    trimmedData = trim(timestamp,
    [close, high, low, open, volume],
    startDate, endDate)
112
    columns = ['date', 'open', 'high',
    'low', 'close', 'volume']

    return dict(zip(columns,
    trimmedData))
```

research.py

```
from datetime import datetime

import pandas as pd
import json

from flask import jsonify

from
    core.parts.preprocessing.csv_retriever
    import get_historical_quotes
from
    core.parts.preprocessing.preprocessing
    import prepareData
from core.parts.processing.test import
    get_wavelet
from core.parts.processing.wavelet
    import compute_dwt
from server_side.utils import
    DateTimeEncoder
from wavelet_research.waveletMaker
    import *


class Image:
    def __init__(self, name, img):
        self.name = name
        self.img = img


def simple_wavelet_research():
    data = prepareData('eurusd=x',
    "TIME_SERIES_DAILY", datetime(2006,
    8, 2), datetime(2007, 8, 2))

    # fig, ax = plt.subplots()
    # ax.plot(data["date"],
    data["open"])
    # ax.set(title="raw data")
    # ax.grid()
    # figfile = BytesIO()
    # fig.savefig(figfile, format='png')
    # figfile.seek(0)
    transforms =
    countWaveletTransform(data["date"],
    data["open"])
    dataFrame = pd.DataFrame(data)

    hurstIndex =
    prepareHurstIndex(data["date"],
    data["open"])  # hurst index of
    close
    dataFrame["hurst"] =
    pd.Series(hurstIndex["value"],
    index=dataFrame.index)

    transforms =
    countWaveletTransform(data["date"],
    data["open"])
    flattenTransforms =
    flatWaveletTransform(transforms)
    #
    dataFrame.append(pd.DataFrame(flattenTransfo

    tsDf = pd.concat([dataFrame,
    pd.DataFrame(flattenTransforms)],
    axis=1)

    waveletDetails =
    collectPlots(transforms)
    result = {"timeSeries":
    json.loads(tsDf.to_json(orient="records",
    date_format="iso")),
              "waveletDetails":
    waveletDetails}

    jsonify(json.dumps(result))



def macd_research():
    date, x, _, _, _, _ =
    prepareData('eurusd=x',
    "TIME_SERIES_DAILY", datetime(2006,
    8, 2), datetime(2007, 8, 2))
    prefix = "no_elliott"
    print(type(date))
    x = exp_moving_average(x, 5)
    tx = macd(x, 12, 26)
    ax = list()
```

```
    at = list()
    splt_x, splt_date =
split_timeline(tx, date)
    ax += list(splt_x)
    at += list(splt_date)
    wx = list()
    wt = list()
    wt += list(splt_date)

    wavelet_sum = []
    for t in splt_x:
        temp = get_wavelet(t, 'haar')
        wavelet_sum.append(temp)
    wx += list(wavelet_sum)

    fx = list()
    ft = list()
    ft += list(splt_date)

    fft_sum = []
    for t in splt_x:
        temp = np.fft.fft(t)
        fft_sum.append(temp)
    fx += list(fft_sum)
    showPlot(date, x, prefix +
'_w_x.png')
    # macd
    print('macd')
    showPlotMixSeparate(ax, at, prefix
+ '_w_macd.png')
    # wavelet
    print('wt')
    showPlotMixSeparate(wx, wt, prefix
+ '_w_wt.png')

    print('fft')
    showPlotMixSeparate(fx, ft, prefix
+ 'perfect_w_fft.png')

    mainLoop("x", "2y", date, x)


def hurst_research():
    date, x =
get_historical_quotes(start_date=datetime(2003,
9, 2),
                      end_date=datetime(2004, 6, 2))

    x = exp_moving_average(x, 5)
    tx = hurst(x)
    ax = list()
    at = list()

    splt_x, splt_date =
split_timeline(tx, date,
division_line=0.5)

    ax += list(splt_x)
    at += list(splt_date)

    wx = list()
    wt = list()
    wt += list(splt_date)

    wavelet_sum = []
    for t in splt_x:
        temp = get_wavelet(t, 'dmey')
        wavelet_sum.append(temp)
    wx += list(wavelet_sum)

    fx = list()
    ft = list()
    ft += list(splt_date)

    fft_sum = []
    for t in splt_x:
        temp = np.fft.fft(t)
        fft_sum.append(temp)
    fx += list(fft_sum)

    showPlot(date, x, 'w_x.png')

    # macd
    print('hurst', at)
    showPlotMixSeparate(ax, at,
'w_hurst.png')

    # wavelet
    print('wt')
    showPlotMixSeparate(wx, wt,
'w_hurst_wt.png')
```

```
    print('fft')
    showPlotMixSeparate(fx, ft,
    'w_hurst_fft.png')


def wavelet_reserch(date, x,
    type='macd', wavelet='db1'):
    tx = []
    x = exp_moving_average(x, 5)

    if type == 'hurst':
        tx = hurst(x)
    elif type == 'macd':
        tx = macd(x, 12, 26)

    ax = list()
    at = list()

    if type == 'hurst':
        division_line = 0
    else:
        division_line = .5
    splt_x, splt_date =
    split_timeline(tx, date,
    division_line=division_line)

    ax += list(splt_x)
    at += list(splt_date)

    wx = list()
    wt = list()
    wt += list(splt_date)

    wavelet_sum = []
    for t in splt_x:
        temp = compute_dwt(t, wavelet)
        wavelet_sum.append(temp)
    wx += list(wavelet_sum)

    # fx = list()
    # ft = list()
    # ft += list(splt_date)
    #
    # fft_sum = []
    # for t in splt_x:
    #     temp = np.fft.fft(t)
    #     fft_sum.append(temp)
    # fx += list(fft_sum)

    showPlot(date, x, 'w_x.png')

    # showPlotMixSeparate(ax, at,
    'w_hurst.png')
    # showPlotMixSeparate(wx, wt,
    'w_hurst_wt.png')
    # showPlotMixSeparate(fx, ft,
    'w_hurst_fft.png')
    return ax, at, wx, wt


# hurst_research()
# macd_research()

simple_wavelet_research()
```

routes.py

```
import json
from datetime import datetime

from flask import Flask, jsonify
from flask import render_template
from flask import request
from flask_cors import CORS,
    cross_origin

from core.parts.analysis.analyser
    import analyse

from
    core.parts.preprocessing.csv_retriever
    import get_historical_quotes
from core.parts.processing.elliot
    import elliot_waves
from core.parts.processing.wavelet
    import calculate_cwt
from core.parts.wavelets.wavelets
    import __all__
from server_side.utils import
    format_date, DateTimeEncoder
from wavelet_research.waveletMaker
    import *
```

```
app = Flask(__name__)
CORS(app)
app.config['CORS_HEADERS'] =
    'Content-Type'

elliot_folder_name = 'elliot/'


class Image:
    def __init__(self, name, img):
        self.name = name
        self.img = img


@app.route('/')
def hello_page(name=None):
    wavelet_list_retrieved = ["All"] +
    __all__
    #
    print(os.path.dirname(os.path.realpath(__file__)))
    # print(wavelet_list_retrieved)
    return render_template('index.html')
    # return
    render_template('plot_page.html',
    name=name,
    wavelet_list=wavelet_list_retrieved)


# @app.route('/analyser')
# def analyser_page(name=None):
#     wavelet_list_retrieved = ["All"]
    + __all__
#     return
    render_template('plot_page.html',
    name=name,
    wavelet_list=wavelet_list_retrieved)


@app.route('/elliott')
def elliot_page(name=None):
    return
    render_template('elliot.html',

    elliot_list=[Image(name,
    common_folder + elliot_folder_name
    + name + '.jpg') for name in
    elliot_waves])


@app.route('/wavelets',
    methods=['POST'])
def show_wavelets():
    # print(request.form)
    wavelet_name =
    request.form["wavelet_name"]
    stock = request.form["ticker"] +
    "=x"
    wrange = request.form["period"]  #
    moving_avg_width =
    request.form["ma_param"]
    moving_avg_width =
    int(moving_avg_width)
    folder_name = stock + '_' + wrange
    wavelet_image_name = []
    print("1␣-␣" + wavelet_name)
    # date, x, _, _, _, _ =
    prepareData(stock, wrange)
    # 2003 9 2 - 2004 6 2

    date, x =
    get_historical_quotes(start_date=datetime(19
    2, 2),

    end_date=datetime(2003, 2, 2))
    for i in range(moving_avg_width -
    1, len(x)):
        for j in range(i -
    moving_avg_width + 1, i):
            x[i] += x[j]
        x[i] /= moving_avg_width

    plot_name = common_folder +
    folder_name + '/' + input_plot_name
    + '.png'

    wavelet_image_name.append(Image('input_plot'
    plot_name))
    showPlot(date, x, plot_name)
```

```
78      macd_plot = common_folder +
        folder_name + '/' + macd_name +
        '.png'
        ma1 = 10
        ma2 = 50

        wavelet_image_name.append(Image('macd_plot:␣
        %d—%d' % (ma1, ma2), macd_plot))
        calculateMACD(date, x, ma1, ma2,
        macd_plot)

83
        if wavelet_name != 'All':
            print("1␣—␣" + wavelet_name)
            folder_name = stock + '_' +
        wrange
            time_scale = int(wrange[:—1])
88
        calculate_cwt(math.ceil(time_scale
        / 4.), date, x, folder_name,
        wavelet_name)
            wavelet_image_name.append(
                Image(wavelet_name,
        common_folder + folder_name + '/' +
        wavelet_name + '.png'))
        else:
            mainLoop(stock, wrange, date, x)
93          for name in __all__:

        wavelet_image_name.append(Image(name,
        common_folder + folder_name + '/' +
        name + '.png'))

        print(123321)
        # hurst_plot = common_folder +
        folder_name + '/' + hurst_plot_name
        + '.png'
98      #
        wavelet_image_name.append(Image('hurst_plot',
        hurst_plot))
        # calculateHurst(date, x,
        hurst_plot)

        # lyapunov_plot = common_folder +
        folder_name + '/' +
        lyapunov_plot_name + '.png'
                                                #
                                                wavelet_image_name.append(Image('lyapunov_plo
                                                lyapunov_plot))
                                                # calculateLyapunov(date, x,
                                                lyapunov_plot)
103

                                                wavelet_list_retrieved = ["All"] +
                                                __all__
                                                return
                                                render_template('plot_page.html',
                                                wavelet_list=wavelet_list_retrieved,
108
                                                wavelet_image_names=wavelet_image_name)


                                            @app.route('/sendRequest',
                                                methods=['POST'])
                                            def requestResponse():
113                                             print(request.form)
                                                response = json.dumps(request.form)
                                                print(response)
                                                return response


118
                                            @app.route('/analyse', methods=['POST'])
                                            @cross_origin()
                                            def analysis():
                                                parsed_json =
                                                json.loads(request.data.decode("utf—8"))
123
                                                startDate =
                                                format_date(parsed_json["startDate"])
                                                endDate =
                                                format_date(parsed_json["endDate"])

                                                result =
                                                analyse(parsed_json["currency"],
                                                parsed_json["frequency"],
                                                startDate, endDate)
128
                                                return jsonify(result)


                                            if __name__ == '__main__':
133                                             app.run(debug=True)
```

transform.py

```
from __future__ import division

import numpy as np
import scipy
import scipy.signal
import scipy.optimize
import scipy.special

from .wavelets import Morlet

__all__ = ['cwt', 'WaveletAnalysis',
    'WaveletTransform']


def cwt(data, wavelet=None,
    widths=None, dt=1, frequency=False,
    axis=-1):
    """Continuous wavelet transform
    using the fourier transform
    convolution as used in Terrence and
    Compo.

    (as opposed to the direct
    convolution method used by
    scipy.signal.cwt)

    *This method is over 10x faster
    than the scipy default.*

    Performs a continuous wavelet
    transform on `data`,
    using the `wavelet` function. A CWT
    performs a convolution
    with `data` using the `wavelet`
    function, which is characterized
    by a width parameter and length
    parameter.

    Parameters
    ----------
    data : (N,) ndarray
        data on which to perform the
    transform.

    wavelet : function
        Wavelet function in either time
    or frequency space, which
        should take 2 arguments. If the
    wavelet is frequency based,
        frequency must be set to True.

        The first parameter is time or
    frequency.

        The second is a width
    parameter, defining the size of the
    wavelet
        (e.g. standard deviation of a
    gaussian).

        The wavelet function, Y, should
    be such that
        Int[-inf][inf](|Y|^2) = 1

        It is then multiplied here by a
    normalisation factor,
        which gives it unit energy.

        In the time domain, the
    normalisation factor is

            (s / dt)

        In the frequency domain, the
    normalisation factor is

            (2 * pi * dt / s) ^ (1/2),

    widths : (M,) sequence
        Widths to use for transform.

    dt: float
        sample spacing. defaults to 1
    (data sample units).

    frequency: boolean. Whether the
    wavelet function is one of
                time or frequency.
    Default, False, is for a time
                representation of the
    wavelet function.
```

```
    axis: int, the axis in the data over which to perform the 1D
          transform (default 0)

    Returns
    -------
    cwt: (M, N) ndarray
        Will have shape of (len(data), len(widths)).

    """
    if widths is None:
        raise UserWarning('Have to specify some widths (scales)')

    if not wavelet:
        raise UserWarning('Have to specify a wavelet function')

    if frequency:
        return cwt_freq(data, wavelet, widths, dt, axis)
    elif not frequency:
        return cwt_time(data, wavelet, widths, dt, axis)


def cwt_time(data, wavelet, widths, dt, axis):
    # wavelets can be complex so output is complex
    output = np.zeros((len(widths),) + data.shape, dtype=np.complex)

    # compute in time
    slices = [None for _ in data.shape]
    slices[axis] = slice(None)
    for ind, width in enumerate(widths):
        # number of points needed to capture wavelet
        M = 10 * width / dt
        # times to use, centred at zero
        t = np.arange((-M + 1) / 2., (M + 1) / 2.) * dt
        # sample wavelet and normalise
        norm = (dt / width) ** .5
        wavelet_data = norm * wavelet(t, width)
        output[ind, :] = scipy.signal.fftconvolve(data,

                                                  wavelet_data[slices],

                                                  mode='same')
    return output


def cwt_freq(data, wavelet, widths, dt, axis):
    # compute in frequency
    # next highest power of two for padding
    N = data.shape[axis]
    pN = int(2 ** np.ceil(np.log2(N)))
    # N.B. padding in fft adds zeros to the *end* of the array,
    # not equally either end.
    fft_data = scipy.fft(data, n=pN, axis=axis)
    # frequencies
    w_k = np.fft.fftfreq(pN, d=dt) * 2 * np.pi

    # sample wavelet and normalise
    norm = (2 * np.pi * widths / dt) ** .5
    wavelet_data = norm[:, None] * wavelet(w_k, widths[:, None])

    # Convert negative axis. Add one to account for
    # inclusion of widths axis above.
    axis = (axis % data.ndim) + 1

    # perform the convolution in frequency space
    slices = [slice(None)] + [None for _ in data.shape]
    slices[axis] = slice(None)

    out = scipy.ifft(fft_data[None] * wavelet_data.conj()[slices],
                     n=pN, axis=axis)
```

```
        # remove zero padding
        slices = [slice(None) for _ in
        out.shape]
        slices[axis] = slice(None, N)

        if data.ndim == 1:
            return out[slices].squeeze()
        else:
            return out[slices]


    class WaveletTransform(object):
        """
        Sx.y are references to section x.y
        in Torrence and Compo,
        A Practical Guide to Wavelet
        Analysis (BAMS, 1998)


        ### Wavelet function requirements
        (S3.b) ###

        To be admissible as a wavelet, a
        function must:

        - have zero mean
        - be localised in both time and
        frequency space

        These functions are a function of a
        dimensionless time
        parameter.

        ### Function selection
        considerations (S3.e) ###

        #### Complex / Real

        A *complex* wavelet function will
        return information about both
        amplitude and phase and is better
        adapted for capturing
        *osillatory behaviour*.

        A *real* wavelet function returns
        only a single component and
        can be used to isolate *peaks or
        discontinuities*.

        ### Width

        Define the width of a wavelet as
        the e-folding time of the
        wavelet amplitude.

        The resolution of the wavelet
        function is determined by the
        balance between the width in real
        and fourier space.

        A narrow function in time will have
        good time resolution but
        poor frequency resolution and vice
        versa.

        ### Shape

        The wavelet function should
        represent the type of features
        present in the time series.

        For time series with sharp jumps or
        steps, choose a boxcar-like
        function such as Harr; while for
        smoothly varying time series,
        choose something like a damped
        cosine.

        The choice of wavelet function is
        not critical if one is only
        qualitatively interested in the
        wavelet power spectrum.

        ### Equivalent Fourier period
        (S3.h) ###

        The peak wavelet response does not
        necessarily occur at 1 / s.

        If we wish to compare wavelet
        spectra at different scales with
        each other and with fourier modes,
        we need a common set of
```

```
        units.

        The equivalent fourier period is
defined as where the wavelet
        power spectrum reaches its maximum
and can be found analytically.
        """
        def __init__(self, data=None,
time=None, dt=1,
                     dj=0.125,
wavelet=Morlet(), unbias=False,
                     mask_coi=False,
frequency=False, axis=-1):
            """Arguments:
                data -- 1 dimensional input
signal
                time -- corresponding times
for the input signal
                    not essential, but
the coi will be calculated
                    for time starting at
zero.
                dt -- sample spacing
                dj -- scale resolution
                wavelet -- wavelet class to
use, must have an attribute
                    'time', giving a
wavelet function that takes (t, s)
                    as arguments and,
if frequency is True, an
                    attribute
'frequency', giving a wavelet
function
                    that takes (w, s)
as arguments.
                unbias -- boolean, whether
to unbias the power spectrum, as
                    in Liu et al. 2007
(default False)
                frequency -- boolean,
compute the cwt in frequency space?
                     (default False)
                mask_coi -- disregard
wavelet power outside the cone of
                    influence when
computing global wavelet spectrum
                    (default False)
                axis -- axis of the input
data to transform over (default -1)
            """
            self.data = data
            if time is None:
                time =
np.indices((data.shape[axis],)).squeeze()
* dt
            self.time = time
            self.anomaly_data = self.data -
self.data.mean(axis=axis,

                           keepdims=True)
            self.N = data.shape[axis]
            self.data_variance =
self.data.var(axis=axis,
keepdims=True)
            self.dt = dt
            self.dj = dj
            self.wavelet = wavelet
            # which continuous wavelet
transform to use
            self.cwt = cwt
            self.frequency = frequency
            self.unbias = unbias
            self.mask_coi = mask_coi
            self.axis = axis

        @property
        def fourier_period(self):
            """Return a function that
calculates the equivalent fourier
            period as a function of scale.
            """
            return getattr(self.wavelet,
'fourier_period')

        @property
        def fourier_periods(self):
            """Return the equivalent
fourier periods for the scales
used."""
            return
self.fourier_period(self.scales)

        @property
        def s0(self):
```

```
        if not hasattr(self, '_s0'):
            return self.find_s0()
        else:
            return self._s0

    @s0.setter
    def s0(self, value):
        setattr(self, '_s0', value)

    def find_s0(self):
        """Find the smallest resolvable
scale by finding where the
        equivalent fourier period is
equal to 2 * dt. For a Morlet
        wavelet, this is roughly 1.
        """
        dt = self.dt

        def f(s):
            return
self.fourier_period(s) - 2 * dt
        return scipy.optimize.fsolve(f,
1)[0]

    @property
    def scales(self):
        if not hasattr(self, '_scales'):
            return
self.compute_optimal_scales()
        else:
            return self._scales

    @scales.setter
    def scales(self, value):
        setattr(self, '_scales', value)

    def compute_optimal_scales(self):
        """Form a set of scales to use
in the wavelet transform.

        For non-orthogonal wavelet
analysis, one can use an
        arbitrary set of scales.

        It is convenient to write the
scales as fractional powers of
        two:

            s_j = s_0 * 2 ** (j * dj),
j = 0, 1, ..., J

            J = (1 / dj) * log2(N * dt
/ s_0)

        s0 - smallest resolvable scale
        J - largest scale

        choose s0 so that the
equivalent Fourier period is 2 * dt.

        The choice of dj depends on the
width in spectral space of
        the wavelet function. For the
morlet, dj=0.5 is the largest
        that still adequately samples
scale. Smaller dj gives finer
        scale resolution.
        """
        dt = self.dt
        # resolution
        dj = self.dj
        # smallest resolvable scale,
chosen so that the equivalent
        # fourier period is
approximately 2dt
        s0 = self.s0

        # Largest scale
        J = int((1 / dj) *
np.log2(self.N * dt / s0))

        sj = s0 * 2 ** (dj *
np.arange(0, J + 1))
        return sj

    # TODO: use np.frompyfunc on this
    # TODO: can we just replace it with
fftfreqs?
    def w_k(self, k=None):
        """Angular frequency as a
function of fourier index.

        If no k, returns an array of
all the angular frequencies
```

```
        calculated using the length of the data.

        See eq5 of TC.
        """
        dt = self.dt
        N = self.N
        a = 2 * np.pi / (N * dt)
        if k is None:
            k = np.arange(N)
            w_k = np.arange(N) * a
            w_k[np.where(k > N // 2)] *= -1
        elif type(k) is np.ndarray:
            w_k = a * k
            w_k[np.where(k > N // 2)] *= -1
        else:
            w_k = a * k
            if k <= N // 2:
                pass
            elif k > N // 2:
                w_k *= -1
        return w_k

    @property
    def wavelet_transform(self):
        """Calculate the wavelet transform."""
        widths = self.scales

        if self.frequency:
            wavelet = self.wavelet.frequency
        else:
            wavelet = self.wavelet.time

        return self.cwt(self.anomaly_data,
                        wavelet=wavelet,
                        widths=widths,
                        dt=self.dt,
                        frequency=self.frequency,
                        axis=self.axis)

    @property
    def wavelet_power(self):
        """Calculate the wavelet power spectrum, optionally using
        the bias correction factor introduced by Liu et al. 2007,
        which is to divide by the scale.
        """
        if self.unbias:
            return (np.abs(self.wavelet_transform).T ** 2 / self.scales).T
        elif not self.unbias:
            return np.abs(self.wavelet_transform) ** 2

    def reconstruction(self, scales=None):
        """Reconstruct the original signal from the wavelet
        transform. See S3.i.

        For non-orthogonal wavelet functions, it is possible to
        reconstruct the original time series using an arbitrary
        wavelet function. The simplest is to use a delta function.

        The reconstructed time series is found as the sum of the
        real part of the wavelet transform over all scales,

        x_n = (dj * dt^(1/2)) / (C_d * Y_0(0)) \
                    * Sum_(j=0)^J { Re(W_n(s_j)) / s_j^(1/2) }

        where the factor C_d comes from the recontruction of a delta
        function from its wavelet transform using the wavelet
        function Y_0. This C_d is a constant for each wavelet
        function.
        """
        dj = self.dj
```

```
        dt = self.dt
        C_d = self.C_d
        Y_00 = self.wavelet.time(0)
        if scales is not None:
            old_scales = self.scales
            self.scales = scales

        s = self.scales
        W_n = self.wavelet_transform

        if scales is not None:
            self.scales = old_scales

        # use the transpose to allow broadcasting
        real_sum = np.sum(W_n.real.T / s ** .5, axis=-1).T
        x_n = real_sum * (dj * dt ** .5 / (C_d * Y_00))

        # add the mean back on (x_n is anomaly time series)
        x_n += self.data.mean(axis=self.axis, keepdims=True)

        return x_n

    @property
    def global_wavelet_spectrum(self):
        if not self.mask_coi:
            mean_power = np.mean(self.wavelet_power, axis=1)
        elif self.mask_coi:
            mean_power = self.coi_mean(self.wavelet_power, axis=1)
        var = self.data_variance
        return mean_power / var

    def coi_mean(self, arr, axis=1):
        """Calculate a mean, but only over times within the cone of influence.

        Implement so can replace np.mean(wavelet_power, axis=1)
        """
        # TODO: consider applying upstream, inside wavelet_power
        coi = self.wavelet.coi
        s = self.scales
        t = self.time
        T, S = np.meshgrid(t, s)
        inside_coi = (coi(S) < T) & (T < (T.max() - coi(S)))
        mask_power = np.ma.masked_where(~inside_coi, self.wavelet_power)
        mask_mean = np.mean(mask_power, axis=axis)
        return mask_mean

    @property
    def C_d(self):
        """Constant used in reconstruction of data from delta wavelet function. See self.reconstruction and S3.i.

        To derive C_d for a new wavelet function, first assume a
        time series with a delta function at time n=0, given by x_n = d_n0. This time series has a Fourier transform x_k = 1 /
        N, constant over k.

        Substituting x_k into eq4 at n=0 (the peak of the delta
        function), the wavelet transform becomes

            W_d(s) = (1 / N) Sum[k=0][N-1] { Y'*(s, w_k) }

        The reconstruction then gives

            C_d = (dj * dt^(1/2)) / Y_0(0) \
                        * Sum_(j=0)^J { Re(W_d(s_j)) / s_j^(1/2) }
```

```
        C_d is scale independent and a constant for each wavelet
        function.
        """
        if hasattr(self.wavelet, 'C_d'):
            return self.wavelet.C_d
        else:
            return self.compute_Cdelta()

    def compute_Cdelta(self):
        """Compute the parameter C_delta (see self.C_d), used in
        reconstruction. See section 3.i of TC98.

        FIXME: this doesn't work. TC98 gives 0.776 for the morlet
        wavelet with dj=0.125.
        """
        dj = self.dj
        dt = self.dt
        s = self.scales
        W_d = self.wavelet_transform_delta

        # value of the wavelet function at t=0
        Y_00 = self.wavelet.time(0)

        real_sum = np.sum(W_d.real / s ** .5)
        C_d = real_sum * (dj * dt ** .5 / Y_00)
        return C_d

    @property
    def wavelet_transform_delta(self):
        """Calculate the delta wavelet transform.

        Returns an array of the transform computed over the scales.
        """
        Y_0 = self.wavelet.frequency  # wavelet as f(w_k, s)
        WK, S = np.meshgrid(self.w_k(), self.scales)

        # compute Y_ over all s, w_k and sum over k
        norm = (2 * np.pi * S / self.dt) ** .5  # normalisation factor with dt=1
        W_d = (1 / self.N) * np.sum(norm * Y_0(WK, S), axis=1)

        # N.B This W_d is 1D (defined only at n=0)
        return W_d

    @property
    def wavelet_variance(self):
        """Equivalent of Parseval's theorem for wavelets, S3.i.

        The wavelet transform conserves total energy, i.e. variance.

        Returns the variance of the input data.
        """
        # TODO: mask coi for calculation of wavelet_variance
        # is this possible? how does it change the factors?
        dj = self.dj
        dt = self.dt
        C_d = self.C_d
        N = self.N
        s = np.expand_dims(self.scales, 1)

        A = dj * dt / (C_d * N)

        var = A * np.sum(np.abs(self.wavelet_transform) ** 2 / s)

        return var

    @property
    def coi(self):
```

```
        """The Cone of Influence is the region near the edges of the
        input signal in which edge effects may be important.

        Return a tuple (T, S) that describes the edge of the cone
        of influence as a single line in (time, scale).
        """
        Tmin = self.time.min()
        Tmax = self.time.max()
        Tmid = Tmin + (Tmax - Tmin) / 2
        s = np.logspace(np.log10(self.scales.min()),
                        np.log10(self.scales.max()),
                        100)
        c1 = Tmin + self.wavelet.coi(s)
        c2 = Tmax - self.wavelet.coi(s)

        C = np.hstack((c1[np.where(c1 < Tmid)], c2[np.where(c2 > Tmid)]))
        S = np.hstack((s[np.where(c1 < Tmid)], s[np.where(c2 > Tmid)]))

        # sort w.r.t time
        iC = C.argsort()
        sC = C[iC]
        sS = S[iC]

        return sC, sS

    def plot_power(self, ax=None, coi=True):
        """Create a basic wavelet power plot with time on the
        x-axis, scale on the y-axis, and a cone of influence
        overlaid.

        Requires matplotlib.
        """
        import matplotlib.pyplot as plt

        if not ax:
            fig, ax = plt.subplots()

        Time, Scale = np.meshgrid(self.time, self.scales)
        ax.contourf(Time, Scale, self.wavelet_power, 100)

        ax.set_yscale('log')
        ax.grid(True)

        if coi:
            coi_time, coi_scale = self.coi
            ax.fill_between(x=coi_time,
                            y1=coi_scale,
                            y2=self.scales.max(),
                            color='gray',
                            alpha=0.3)

        ax.set_xlim(self.time.min(), self.time.max())

        return ax

WaveletAnalysis = WaveletTransform

# TODO: derive C_d for given wavelet
```

wavelets.py

```
from __future__ import division

import numpy as np
import scipy
import scipy.signal
import scipy.optimize
import scipy.special
from scipy.misc import factorial

# __all__ = ['Morlet', 'Paul', 'DOG',
    'Ricker']
```

```
__all__ = ['DOG', 'Morlet', 'Paul',
    'Ricker', 'Ricker']

class Morlet(object):
    def __init__(self, w0=6):
        """w0 is the nondimensional
    frequency constant. If this is
        set too low then the wavelet
    does not sample very well: a
        value over 5 should be ok,
    Terrence and Compo set it to 6.
        """
        self.w0 = w0
        if w0 == 6:
            # value of C_d from TC98
            self.C_d = 0.776

    def __call__(self, *args, **kwargs):
        return self.time(*args,
    **kwargs)

    def time(self, t, s=1.0,
    complete=True):
        """
        Complex Morlet wavelet, centred
    at zero.

        Parameters
        ----------
        t : float
            Time. If s is not
    specified, this can be used as the
            non-dimensional time t/s.
        s : float
            Scaling factor. Default is
    1.
        complete : bool
            Whether to use the complete
    or the standard version.

        Returns
        -------
        complex: value of the morlet
    wavelet at the given time

        See Also
        --------
        scipy.signal.gausspulse

        Notes
        -----
        The standard version::

            pi**-0.25 * exp(1j*w*x) *
    exp(-0.5*(x**2))

        This commonly used wavelet is
    often referred to simply as the
        Morlet wavelet.  Note that this
    simplified version can cause
        admissibility problems at low
    values of w.

        The complete version::

            pi**-0.25 * (exp(1j*w*x) -
    exp(-0.5*(w**2))) * exp(-0.5*(x**2))

        The complete version of the
    Morlet wavelet, with a correction
        term to improve admissibility.
    For w greater than 5, the
        correction term is negligible.

        Note that the energy of the
    return wavelet is not normalised
        according to s.

        The fundamental frequency of
    this wavelet in Hz is given
        by ``f = 2*s*w*r / M`` where r
    is the sampling rate.

        """
        w = self.w0

        x = t / s

        output = np.exp(1j * w * x)

        if complete:
            output -= np.exp(-0.5 * (w
    ** 2))
```

```
        output *= np.exp(-0.5 * (x **
2)) * np.pi ** (-0.25)

        return output

    # Fourier wavelengths
    def fourier_period(self, s):
        """Equivalent fourier period of
morlet"""
        return 4 * np.pi * s / (self.w0
+ (2 + self.w0 ** 2) ** .5)

    # Frequency representation
    def frequency(self, w, s=1.0):
        """Frequency representation of
morlet.

        s - scale
        w - angular frequency
        """
        x = w * s
        # heaviside mock
        Hw = np.array(w)
        Hw[w <= 0] = 0
        Hw[w > 0] = 1
        return np.pi ** -.25 * Hw *
np.exp((-(x - self.w0) ** 2) / 2)

    def coi(self, s):
        """The e folding time for the
autocorrelation of wavelet
        power at each scale, i.e. the
timescale over which an edge
        effect decays by a factor of
1/e^2.

        This can be worked out
analytically by solving

            |Y_0(T)|^2 / |Y_0(0)|^2 = 1
/ e^2
        """
        return 2 ** .5 * s


class Paul(object):
    def __init__(self, m=4):
        """Initialise a Paul wavelet
function of order m.
        """
        self.m = m

    def __call__(self, *args, **kwargs):
        return self.time(*args,
**kwargs)

    def time(self, t, s=1.0):
        """
        Complex Paul wavelet, centred
at zero.

        Parameters
        ----------
        t : float
            Time. If s is not
specified, i.e. set to 1, this can
be
            used as the non-dimensional
time t/s.
        s : float
            Scaling factor. Default is
1.

        Returns
        -------
        complex: value of the paul
wavelet at the given time

        The Paul wavelet is defined (in
time) as::

            (2 ** m * i ** m * m!) /
(pi * (2 * m)!) \
                    * (1 - i * t / s)
** -(m + 1)

        """
        m = self.m
        x = t / s

        const = (2 ** m * 1j ** m *
factorial(m)) \
            / (np.pi * factorial(2 *
m)) ** .5
```

```
        functional_form = (1 — 1j * x)
** —(m + 1)

        output = const * functional_form

        return output

    # Fourier wavelengths
    def fourier_period(self, s):
        """Equivalent fourier period of
Paul"""
        return 4 * np.pi * s / (2 *
self.m + 1)

    # Frequency representation
    def frequency(self, w, s=1.0):
        """Frequency representation of
Paul.

        Parameters
        ——————————
        w : float
            Angular frequency. If s is
not specified, i.e. set to 1,
            this can be used as the
non—dimensional angular
            frequency w * s.
        s : float
            Scaling factor. Default is
1.

        Returns
        ———————
        complex: value of the paul
wavelet at the given time

        """
        m = self.m
        x = w * s
        # heaviside mock
        Hw = 0.5 * (np.sign(x) + 1)

        # prefactor
        const = 2 ** m / (m *
factorial(2 * m — 1)) ** .5

        functional_form = Hw * (x) ** m
* np.exp(—x)

        output = const * functional_form

        return output

    def coi(self, s):
        """The e folding time for the
autocorrelation of wavelet
        power at each scale, i.e. the
timescale over which an edge
        effect decays by a factor of
1/e^2.

        This can be worked out
analytically by solving

            |Y_0(T)|^2 / |Y_0(0)|^2 = 1
/ e^2
        """
        return s / 2 ** .5


class DOG(object):
    def __init__(self, m=1):
        """Initialise a Derivative of
Gaussian wavelet of order m."""
        if m == 2:
            # value of C_d from TC98
            self.C_d = 3.541
        elif m == 6:
            self.C_d = 1.966
        else:
            pass
        self.m = m

    def __call__(self, *args, **kwargs):
        return self.time(*args,
**kwargs)

    def time(self, t, s=1.0):
        """
        Return a DOG wavelet,

        When m = 2, this is also known
as the "Mexican hat", "Marr"
```

```
    or "Ricker" wavelet.

    It models the function::

        ``A d^m/dx^m exp(-x^2 / 2)``,

    where ``A = (-1)^(m+1) / (gamma(m + 1/2))^.5``
    and   ``x = t / s``.

    Note that the energy of the return wavelet is not normalised
    according to s.

    Parameters
    ----------
    t : float
        Time. If s is not specified, this can be used as the
        non-dimensional time t/s.
    s : scalar
        Width parameter of the wavelet.

    Returns
    -------
    float : value of the ricker wavelet at the given time

    Notes
    -----
    The derivative of the gaussian has a polynomial representation:

    from http://en.wikipedia.org/wiki/Gaussian_function.

    "Mathematically, the derivatives of the Gaussian function can be
    represented using Hermite functions. The n-th derivative of the
    Gaussian is the Gaussian function itself multiplied by the n-th
    Hermite polynomial, up to scale."


    http://en.wikipedia.org/wiki/Hermite_polynom

    Here, we want the 'probabilists' Hermite polynomial (He_n),
    which is computed by scipy.special.hermitenorm

    """
    x = t / s
    m = self.m

    # compute the hermite polynomial (used to evaluate the
    # derivative of a gaussian)
    He_n = scipy.special.hermitenorm(m)
    gamma = scipy.special.gamma

    const = (-1) ** (m + 1) / gamma(m + 0.5) ** .5
    function = He_n(x) * np.exp(-x ** 2 / 2)

    return const * function

def fourier_period(self, s):
    """Equivalent fourier period of derivative of gaussian"""
    return 2 * np.pi * s / (self.m + 0.5) ** .5

def frequency(self, w, s=1.0):
    """Frequency representation of derivative of gaussian.

    Parameters
    ----------
    w : float
```

```
        Angular frequency. If s is not specified, i.e. set to 1,
        this can be used as the non-dimensional angular
        frequency w * s.
    s : float
        Scaling factor. Default is 1.

    Returns
    -------
    complex: value of the derivative of gaussian wavelet at the
             given time
    """
    m = self.m
    x = s * w
    gamma = scipy.special.gamma
    const = -1j ** m / gamma(m + 0.5) ** .5
    function = x ** m * np.exp(-x ** 2 / 2)
    return const * function

def coi(self, s):
    """The e folding time for the autocorrelation of wavelet
    power at each scale, i.e. the timescale over which an edge
    effect decays by a factor of 1/e^2.

    This can be worked out analytically by solving

        |Y_0(T)|^2 / |Y_0(0)|^2 = 1 / e^2
    """
    return 2 ** .5 * s


class Ricker(DOG):
    def __init__(self):
        """The Ricker, aka Marr / Mexican Hat, wavelet is a
        derivative of gaussian order 2.
        """
        DOG.__init__(self, m=2)
        # value of C_d from TC98
        self.C_d = 3.541


# aliases for DOG2
Marr = Ricker
Mexican_hat = Ricker

all_wavelets = [Morlet, DOG, Paul,
    Ricker, Ricker]
```

actionfilter.js

```
import {fetchStockData} from "../api/analisys";
export function changeStartDate(startDate) {
    return {
        type: 'CHANGE_START_DATE',
        startDate
    }
}

export function changeEndDate(endDate) {
    return {
        type: 'CHANGE_END_DATE',
        endDate
    }
}

export function changeCurrencyPair(pair) {
    return {
        type: 'CHANGE_CURRENCY_PAIR',
        pair
    }
}

export function changeFrequency(frequency) {
    return {
        type: 'CHANGE_FREQUENCY',
        frequency
```

```
    }
}

export function resetStockData() {
    return {
        type: 'RESET_STOCK_DATA',
    }
}

// todo develop it
export function loadStockDataFailure() {
    return {}
}

export function
    loadStockDataSuccess(stockData) {
    return {
        type: 'LOAD_STOCK_DATA_SUCCESS',
        stockData
    }
}
export function loadList() {
    return new Promise((resolve,
    reject) => {
        resolve(JSON.stringify(
            {}
        ))
    })
}
export function loadStockData(currency,
    frequency, startDate, endDate) {
    return (dispatch) => {
        resetStockData();
        return fetchStockData(currency,
    frequency, startDate, endDate)
            .then(stockData => {

    dispatch(loadStockDataSuccess(stockData))
            }).catch(error => {

    loadStockDataFailure(error)
            });
    };
}
```

date.jsx

```
import React from "react";
import autobind from
    "autobind-decorator";
import DatePicker from
    "material-ui/DatePicker";
import styles from "./css/styles.css";
import {changeStartDate, changeEndDate}
    from "../../actions/filter";

const propTypes = {
    dispatch:
    React.PropTypes.func.isRequired,
    filter: React.PropTypes.object
};


export class RangeDatePicker extends
    React.Component {

    @autobind
    handleStartDateChangeEvent(event,
    date) {
        const {dispatch} = this.props;
        dispatch(changeStartDate(date));
    };

    @autobind
    handleEndDateChangeEvent(event,
    date) {
        const {dispatch} = this.props;
        dispatch(changeEndDate(date));
    };


    render() {
        const classes =
    [styles.container,
    styles.input_field,
    styles.date_field].join("␣");

        return (
            <span>
                <DatePicker
                    className={classes}

    onChange={this.handleStartDateChangeEvent}
```

```
                    hintText="Start␣date"
                    mode="landscape"
                    defaultDate={this.props.filter.startDate}
                    autoOk={true}/>
                <DatePicker
                    className={classes}
                    onChange={this.handleEndDateChangeEvent}
                    hintText="End␣date"
                    mode="landscape"
                    defaultDate={this.props.filter.endDate}
                    autoOk={true}/>
            </span>
        );
    }
}

RangeDatePicker.propTypes = propTypes;
export default RangeDatePicker;
```

loadButton.jsx

```
import React from "react";
import autobind from
    "autobind-decorator";
import styles from "./css/styles.css";
import {loadStockData} from
    "../../actions/filter";
import RaisedButton from
    "material-ui/RaisedButton";

const propTypes = {
    dispatch:
    React.PropTypes.func.isRequired,
    filter: React.PropTypes.object
};


export class LoadButton extends
    React.Component {

    @autobind
    handleLoadButtonClicked() {
        const {dispatch} = this.props;
        const {currencyPair, frequency,
    startDate, endDate} =
    this.props.filter;

    dispatch(loadStockData(currencyPair,
    frequency, startDate, endDate));
    };

    render() {
        const classes =
    [styles.container,
    styles.load_button].join("␣");

        return (
            <span className={classes}>
                <RaisedButton
    label="Load" primary={true}
    onClick={this.handleLoadButtonClicked}/>
            </span>
        );
    }
}

LoadButton.propTypes = propTypes;
export default LoadButton;
```

frequency.jsx

```
import React from "react";
import autobind from
    "autobind-decorator";
import SelectField from
    "material-ui/SelectField";
import MenuItem from
    'material-ui/MenuItem';

import styles from "./css/styles.css";
import {changeFrequency} from
    "../../actions/filter";

const propTypes = {
    dispatch:
    React.PropTypes.func.isRequired,
    filter: React.PropTypes.object
```

```
};

export class FrequencyPicker extends
    React.Component {

    @autobind
    handleFrequencyChangeEvent(event,
    val) {
        const {dispatch} = this.props;
        let frequency =
    this.props.filter.availableFrequency[val];

    dispatch(changeFrequency(frequency));
    };

    render() {
        return (
            <span
    className={styles.container}>
                <SelectField
    floatingLabelText="Frequency"

    value={this.props.filter.frequency}

    onChange={this.handleFrequencyChangeEvent}
                >

    {this.props.filter.availableFrequency.map((f
    =>
                        <MenuItem
    key={freq} value={freq}
    primaryText={freq}/>
                    )}
                </SelectField>
            </span>
        );
    }
}

FrequencyPicker.propTypes = propTypes;
export default FrequencyPicker;
```

modelfilter.js

```
export class Filter {
    constructor(initialState) {

        // this.startDate = new
    Date("2016-11-11");
        this.startDate = new
    Date("2017-04-11");
        // this.endDate = new
    Date("2017-10-01");
        this.endDate = new
    Date("2017-09-29");

        this.currencyPair = 'eurusd=x';
        this.commonCurrencies =
    ['eurusd=x', 'usdeur=x'];
        this.availableFrequency =
    ['TIME_SERIES_DAILY'];
        Object.assign(this,
    initialState);
    }
}
```

reducerfilter.js

```
import {Filter} from "../model/filter";

export function filter(state = new
    Filter(), action) {
    switch (action.type) {
        case 'CHANGE_START_DATE':
            return Object.assign({},
    state, {
                startDate:
    action.startDate,
            });
        case 'CHANGE_END_DATE':
            return Object.assign({},
    state, {
                endDate: action.endDate,
            });
        case 'CHANGE_CURRENCY_PAIR':
```

```
            return Object.assign({},
        state, {
                currencyPair:
        action.pair,
            });
        case 'CHANGE_FREQUENCY':
            return Object.assign({},
        state, {
                    frequency:
        action.frequency,
            });
        default:
            return state;
    }
}
```