РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: алгоритми аналізу даних ф‘ючерсного ринку.

Мета кваліфікаційної роботи: розробка програмного забезпечення засобів згладжування, обробки та аналізу даних на ф‘ючерсному ринку.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано платформу для розробки, виконано проєктування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні програмного додатка, що надає можливість реалізації алгоритмів аналізу та прогнозування на ф‘ючерсних ринках, яке реалізоване та може бути використане в роботі з курсами котирувань валют, цінами на стратегічні товари, тощо.

Актуальність даної роботи визначається необхідністю розробки програмного застосунку, яке застосовується для обробки даних котирувань курсів валют або цін на основні стратегічні товари, згладжування шумів даних, визначення тренду, побудови індикаторів та осциляторів для детального аналізу ринку.

Список ключових слів: Ф’ЮЧЕРСНИЙ РИНОК, ТЕХНІЧНИЙ АНАЛІЗ, В-СПЛАЙНИ, ЗГЛАДЖУВАННЯ ДАНИХ, ТРЕНДОВИЙ СПЛАЙН-АНАЛІЗ, ІНДИКАТОР, ОСЦИЛЯТОР.

ABSTRACT

Explanatory note: ___ pp., ___ fig., ___ table, __ appendix, ___ sources.

Object of development: algorithms for analysis of futures market data.

Purpose of the qualification work: development of software for smoothing, processing and analysis of data in the futures market.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development are defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

In the second section the analysis of existing solutions is performed, the platform for development is chosen, the program is designed and developed, the algorithm and structure of program operation are described, input and output data are defined, characteristics of technical means parameters are given, call and program loading are described.

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

Of practical importance is the creation of a software application that provides the ability to implement algorithms for analysis and forecasting in futures markets, which is implemented and can be used to work with exchange rates, prices for strategic goods, etc.

The relevance of this work is determined by the need to develop a software application that is used to process data on quotations of exchange rates or prices for key strategic goods, data noise smoothing, trend determination, construction of indicators and oscillators for detailed market analysis.

Keyword list: FUTURE MARKET, TECHNICAL ANALYSIS, B-SPLINES, DATA SMOOTHING, TREND SPLINE ANALYSIS, INDICATOR, OSCILLATOR.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ADX - Average Directional Index, індикатор середнього спрямованого руху;

ATR - Average True Range, середній правдивий діапазон;

DM - Directional Movement, додатній і від'ємний спрямовані рухи;

CCI - Commodity Channel Index, індекс товарного каналу;

MI - Momentum indicator, індикатор моменту;

ROC - Rate of Change, швидкість зміни;

RSI - Relative Strength Index, індикатор індекс відносної сили.

ЗМІСТ

ВСТУП

Сучасний рівень інформаційних технологій та задачі, пов'язані з інвестиційними проектами, накладають певні вимоги на розробку автоматизованих робочих місць оперативного аналізу ф'ючерсних ринків.

Ф'ючерсні ринки по визначенню складні, а характер їх функціювання не належить до розряду загальновідомих істин. Коли учасник ринку продає або купує ф'ючерсний контракт, він накладає на себе ризик цінової зміни. Тому рішення про інвестування коштів повинно засновуватися на детальному аналізі очикуваної прибутковості та можливого ризику.

Окрім того інтернет-трейдінг вже не є прерогативою окремих трейдерів. Для трейдера таке середовище як Internet стає глобальним розподіленим середовищем фінансових транзакцій та інвестицій, доступ до якого здійснюється з будь-якої точки земної кулі при наявності мінімальних апаратних та програмних ресурсів. Електронна революція вже змінює і саму природу ринка. Ця обставина накладає додаткові вимоги щодо засобів аналізу.

Область застосування інвестиційних проектів дуже велика: будь то прийняття рішень на основі поточної діяльності фірми, аналізу кредитоспроможності клієнтів банку, оцінка ризику страхових операцій, прогнозування успіху біржової діяльності або багато інших задач. Корпоративні чи інституціональні інвестори можуть отримувати значниц прибуток при хеджуванні за допомогою валютних ф'ючерсів. Торгівельні операції з валютними ф'ючерсами можуть допомогти транснаціональним корпораціям компенсувати валютний ризик, пов'язаний з позицією інвестування шляхом купівлі або продажу з розрахунком в іноземній валюті або шляхом експортно-імпортних операцій.

Сучасні програмні засоби зводять до мінімуму втрати трейдера, та допомагають йому поповнити свій капітал. Вони повинні своєчасно повідомляти про наявність збиткових позицій, в результаті чого звільнені активи можна розміщувати більш вигідно; ретельно стежити за станом ціни на

ринку, з метою своєчасного повідомлення трейдера про зміну стана ціни і винесення пропозиції щодо подальших кроків для отримання максимальног прибутку; ретельно стежити за фінансовим портфелем учасника ринка з метою своєчасного корегування інвестиційної стратегії трейдера.

Отже, актуальною є задача створення інформаційної системи обробки даних на ф'ючерсних ринках.

Метою кваліфікаційної роботи є розробка програмного забезпечення засобів згладжування, обробки та аналізу даних на ф'ючерсному ринку.

У даній роботі представлено програмне забезпечення реалізації алгоритмів аналізу та прогнозування на ф'ючерсних ринках, яке реалізоване та може бути використане в роботі з курсами котирувань валют, цінами на стратегічні товари, тощо.

Розроблена програмна система застосована для обробки даних котирувань курсів валют або цін на основні стратегічні товари, згладжування шумів даних, визначення тренду, побудови індикаторів та осциляторів для детального аналізу ринку.

Програмний продукт реалізовано на мові C++ в середовищі візуального програмування Visual Studio 2017.

Програмне забезпечення може бути використане у довільній області, де потрібнен аналіз даних котирувань валют або цін на основні стратегічні товари.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

1.1.1. Технічний аналіз ф‘ючерсного ринку

Ф’ючерс є одним з характерних представників цілого класу інструментів фондового ринку, відомого як «похідні цінні папери». Відповідно до п. 1.26 ст. 1 Закону України «Про оподаткування прибутку підприємств» ф’ючерсна операція - це господарська операція суб’єкта підприємницької діяльності, що передбачає придбання (продаж) ф’ючерсної угоди, тобто зобов’язання придбати (продати) продукцію сезонного виробництва, цінні папери або валютні цінності у домовлений час у майбутньому, за ціною, встановленою у момент домовленості про угоду. При цьому покупець (продавець) ф’ючерсної угоди може відмовитись від її виконання тільки за згодою іншої сторони.

Розрахунки за ф’ючерсною операцією здійснюються в момент фактичної реалізації продукції сезонного виробництва, цінних паперів і валютних операцій. Премія за укладення ф’ючерсної угоди сплачується під час такого укладення.

Ринок Forex (Форекс) − від англійського скорочення Foreign Exchange (переводиться як міжнародний обмін або міжнародна біржа) являє собою сукупність різних торговельних, інвестиційних і спекулятивних операцій з валютою, що реалізуються через систему інститутів.

Технічний аналіз − метод прогнозування цін винятково за рахунок даних історії ринку цін, об'єму й відкритого інтересу.

Стосовно до ринку Forex звичайно використають тільки саму цінову історію, оскільки об'єм на Forex доступний тільки тиковий, а відкритий інтерес не розраховується. Під тиковим об'ємом на Forex розуміється кількість тиків (змін котирування) в одиницю часу, тобто кількість котирувань маркет-мейкерами своїх пропозицій по покупці й продажу валюти в інформаційну

систему. Тиковий об'єм - досить умовний показник, він має мало загального з об'ємом і лише побічно показує активність ринку.

Більшість методів технічного аналізу використають саме ціну, оскільки дані про ціну звичайно є загальнодоступними й у більшості випадків надходять в on-line режимі.

Технічний аналіз є свого роду філософією, що базується на трьох постулатах.

1. Ринок враховує все. Це означає, що будь-який зовнішній фактор, що може впливати на ціну (економічний, політичний, психологічний або який-небудь інший, а також взаємозв'язок всіх факторів) вже врахований у ціні заздалегідь й є присутнім у ціновому графіку, тому у вивченні впливу фундаментальних факторів немає ніякої необхідності, а самої цінової історії досить для прогнозування цін. Цей постулат фомулює головну відмінність технічного аналізу від іншого типу аналізу - фундаментального. Фундаментальний аналіз говорить, що для того, щоб прогнозувати зміну ціни необхідно спрогнозувати фактори, що впливають на попит або пропозицію.

2. Рух цін підлеглий тенденціям. Одним з головних понять у технічному аналізі є поняття тренда (тенденції) або спрямованого руху цін. Основна задача технічного аналітика - розпізнавати нові тенденції на графіку цін на ранніх стадіях розвитку, використати їх у торгівлі й вчасно виходити з ринку, коли тенденції закінчуються. Основний наслідок з положення полягає в тому, що діюча тенденція, цілком ймовірно, буде розвиватися далі, а не звертатися у власну протилежність.

Тенденції діляться на 3 види:

– Бичача тенденція (bullish) – характеризується тим, що нижні ціни коливань цін підвищуються;

– Ведмежа тенденція (bearish) – максимальні ціни коливань ринку знижуються;

– Бічна тенденція (sideways, flat-market, trading range - торговельний діапазон) - ціна практично не рухається.

Вважається, що всі три типи тенденцій не зустрічаються на ринку в чистому виді як рух "по прямій". Тому бичачою тенденцією вважається та, при якій підвищення цін перевершують зниження, а ведмежа навпаки.

3. Історія повторюється. Вважається, що людська психологія у своїй основі незмінна або майже незмінна, у всякому разі, ринок в схожих випадках поводиться по схожих сценаріях, тому якщо які-небудь моделі працювали (давали можливість діставати прибуток) раніше, тобто всі підстави думати, що й в майбутньому вони будуть працювати.

1.1.2. Класифікація методів технічного аналізу ф‘ючерсного ринку

Методи технічного аналізу діляться на ряд категорій:

1. Графічні методи. Під графічними методами аналізу Forex розуміються ті, у яких для прогнозування використовується винятково графічне зображення ринку (графіки ціни, об'єму) і його графічні моделі. Це найпростіші методи аналізу, оскільки вони практично не вимагають програмного забезпечення. Вони й з'явилися раніше всіх. Самі графічні методи поділяють залежно від побудови графіка (баровий, японські свічі й т.д.).

2. Методи, які використовують прості або складні цифрові фільтри й математичну апроксимацію. Ці методи з'явилися досить давно, але активно розвивалися тільки в останні десятиліття, що було пов'язане з розвитком комп'ютерних технологій. Вони підрозділяються на методи проходження за трендом і випереджальні методи.

3. Теорії Циклів. У технічному аналізі використаються ряд теорій циклічних коливань цін. Деякі з них побудовані на чіткому математичному апараті, деякі на теорії економічних циклів, а деякі на власній філософії (наприклад, теорія хвиль Элліотта). Існують навіть теорії циклічності ринку залежно від магнітного поля землі й сонячної активності.

Існує також ряд теорій, які важко віднести до однієї з груп методів технічного аналізу, такі як фрактали, профіль ринку й т.д.

У цілому, технічний аналіз на сьогоднішній день є найбільш популярним методом, що по оцінці активно використають близько 50% трейдерів (інші 50% використають інші методи: фундаментальний аналіз, торгівлю на новинах і навіть інтуїції).

Така популярність технічного аналізу викликана поруч факторів:

– доступ до якісної цінової історії й поточній ціновій динаміці звичайно є відкритим або недорогим, що не можна сказати про фундаментальну інформацію;

– основні технічні індикатори вже в готовому виді присутні в більшості торговельних терміналів й, крім того, існує маса програмного забезпечення що дозволяє створювати власні методи технічного аналізу;

– технічний аналіз не потребує великого аналітичного відділу (і несення всіх відповідних витрат) - це аналіз скоріше для трейдеров, що працюють окремо.

1.1.3. Графічні методи

Курси валют змінюються нелінійно, ріст переміняється спадом і навпаки. Це відбувається тому, що попит та пропозиція на валюту міняються хаотично під впливом величезної кількості різних факторів: ринок складається з агентів бажаючих купити валюту й бажаючих продати валюту не тільки для спекулятивних цілей, але й з метою фактичної поставки в наявному або безготівковому виді.

Технічний аналіз ринку досить умовно ділиться на кілька великих категорій, одна із яких - графічний аналіз.

Графіки для технічного аналізу Forex будуються у двох координатах ціна й/або тиковий об'єм (звичайно відкладається по вертикальній осі) і час (звичайно відкладається по горизонтальній осі).

Торговельним періодом або масштабом будемо називати базовий (одиничний) інтервал часу, використаний для побудови графіка. Іноді в західній літературі він називається timeframe .

Як тимчасовий масштаб цінових графіків традиційно використовуються наступні торговельні періоди: місяць, тиждень, день, 4 години, 1 година, 30 хвилин, 15 хвилин, 10 хвилин, 5 хвилин, 1 хвилина, тик.

Тиком будемо називати одиничну зміну ціни маркет-мейкером ринку, що відобразилась на формі нових цін покупки й продажу (Ask та Bid).

Для побудови графіків (крім тикових) використаються наступні типи даних:

– ціна відкриття періоду (open) − ціна на ринку Forex, яка склалася на початок торговельного періоду (оскільки ціни завжди дві (Bid та Ask), то в більшості випадків використовується Ask+Bid/2);

– ціна закриття періоду (close) − ціна на ринку Forex, яка склалася на кінець торговельного періоду (знову ж таки, в більшості випадків використовують Ask+Bid/2);

– максимальна ціна періоду (high) − найвища ціна на ринку Forex за період (звичайно використовується Ask);

– мінімальна ціна періоду (low) − найнижча ціна на ринку Forex за період (звичайно використовується Bid);

– тиковий об'єм (tick volume) − кількість тиків (змін ціни маркет-мейкеров) за період.

1. Тиковий графік (Tick chart) (рис. 1.1). Тиковий графік являє собою графік самого дрібного масштабу - одиничних котирувань ціни. Оскільки на Forex відсутня інформація про кількість угод й їхньому об'ємі (на відміну від бірж), то тиковий графік являє собою котирування Bid й Ask маркет-мейкеров ринку.

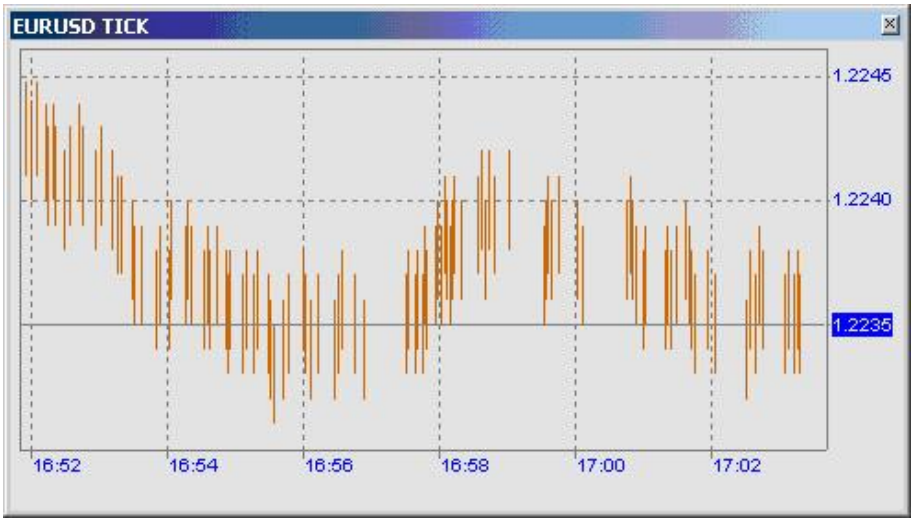

Рис 1.1. Типовий графік

Верхня частина кожної риси являє собою котирування Ask, а нижня − Bid. Тиковий графік використається трейдерами Forex в основному не для аналізу ринку, а для визначення конкретної крапки входу в ринок, коли рішення про вхід у ринок (з усіма відповідними параметрами) вже прийнято.

2. Лінійний графік (Line Chart) (рис. 1.2). Для побудови лінійного графіка ціни в більшості випадків використовується ціна закриття (Close), іноді ціну відкриття (Open), найвищу ціну за період (High), мінімальну ціну за період (Low). Використовують синтетичні варіанти цін, наприклад Median Price ([High + Low]/2) або Typical Price ([High + Low + Close]/3).

Рис. 1.2. Лінійний графік

Вважається що цей тип графіків має кілька значних недоліків. Наприклад, на лінійному графіку ціни відсутня можливість навіть приблизно оцінити те, що відбувалося всередині торговельного періоду. Ще одним недоліком є неможливість побачити цінові розриви між торговельними періодами (gap - гепи), хоча на Forex вони зустрічаються досить рідко. Перевага ж лінійного графіка полягає в тому, що він не видає на екран інформацію, яку багато хто з трейдерів може вважати надлишковою.

3. Графік барів (Bar chart) (рис. 1.3).

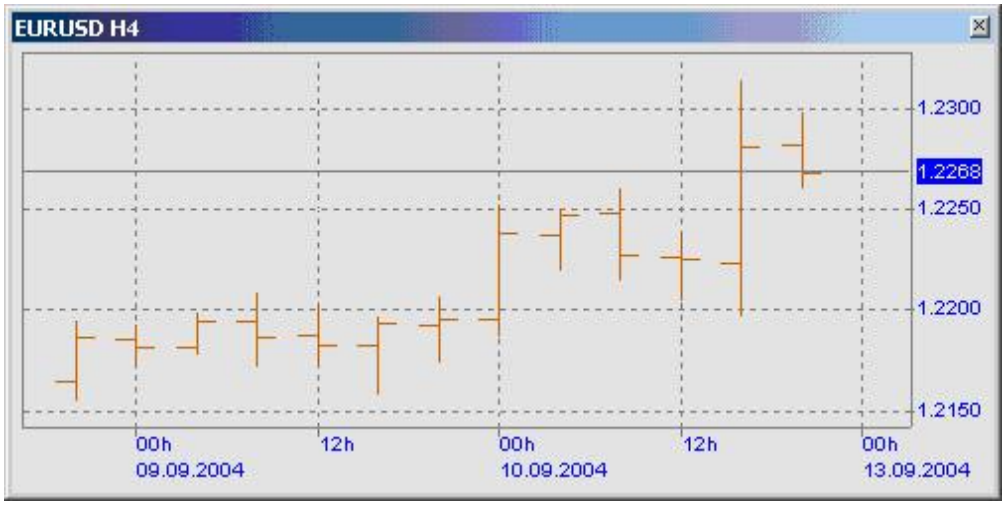

Рис. 1.3. Графік барів

Кожен базовий торговельний період на графіку розуміють наступним чином (рис. 1.4):

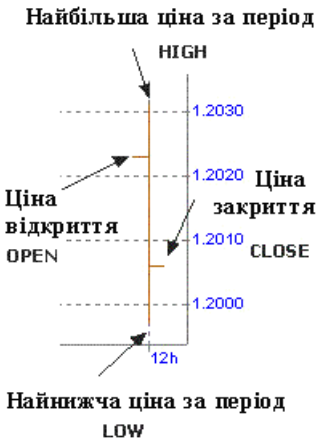

Рис. 1.4. Базовий торговельний період

Якщо ліва риса вище правої це означає, що ціни на початку базового періоду були вище, ніж наприкінці, іншими словами ціна на ринку падала. Якщо нижче, ціна росла.

Незважаючи на те, що графік барів дозволяє бачити діапазон зміни ціни всередині періоду, він не дозволяє бачити, який характер мали рухи ціни всередині періоду. Для цього треба розглядати більш дрібні таймфрейми.

4. Японські свічі (Japanese candlesticks) (рис. 1.5):

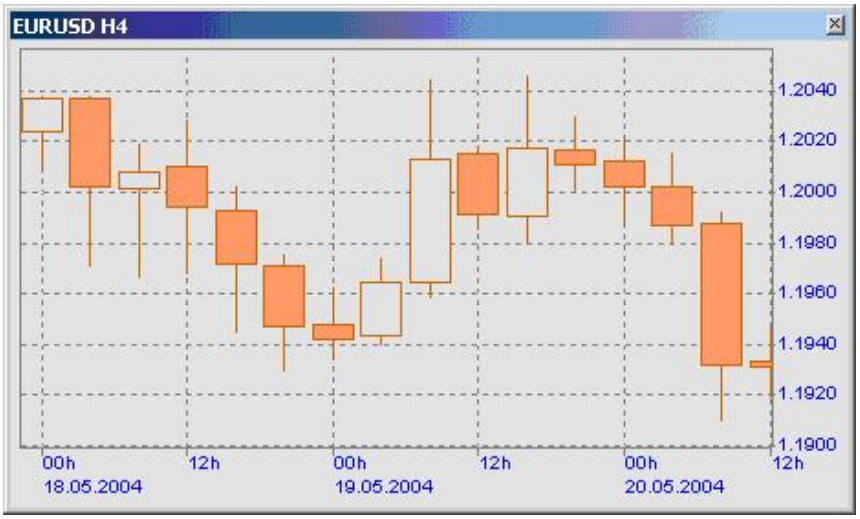

Рис. 1.5. Японські свічі

Цей тип графіку досить схожий на баровий графік, але є більш зручним для візуального сприйняття. Як і баровий графік він теж будується за цінами open, high, low, close (рис. 1.6).

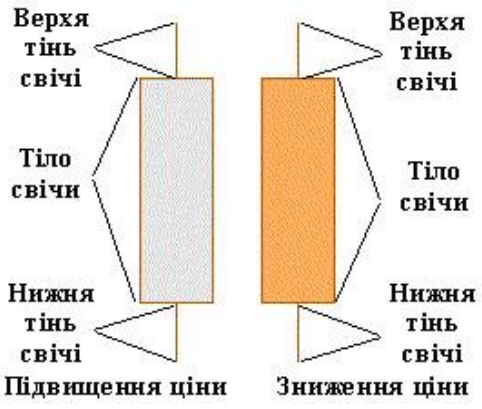

Рис. 1.6. Японська свіча

Відстань між ціною відкриття й ціною закриття утворить тіло свічі , що зафарбовується залежно від співвідношення цін відкриття й закриття. Традиційно діє наступне правило: якщо ціна відкриття вище ціни закриття (тобто ціна впала за торговельний період), то тіло свічі зафарбовується в темні кольори, якщо ж ціна відкриття нижче ціни закриття (ціна за торговельний період зросла), то тіло свічі зафарбовується у світлі кольори або не зафарбовується.

Відстань між тілом свічі й максимальною ціною дня рисується у вигляді вертикальної лінії, що називають верхньою тінню свічі (uwakage) . Відстань між тілом свічі й мінімальною ціною періоду також рисується у вигляді лінії й називається нижньою тінню свічі (shitakage).

Так само як і на баровом графіці, графік японських свіч не дозволяє бачити характер руху всередині торговельного періоду, а показує лише діапазон коливань ціни.

1.1.4. Лінії тренда (Trend line)

Якщо попит на валюту перевищує пропозицію , тобто об'єм валюти, що ринкові агенти бажають купити по даній конкретній ціні більше чим об'єм валюти, що інші агенти бажають продати за цією ціною, то курс валюти на Forex зростає. У зворотному випадку, якщо об'єм валюти, що учасники ринку бажають продати по даній конкретній ціні більше чим об'єм валюти який інші учасники ринку бажають купити, то курс падає.

Якщо попит тривалий час перевищує пропозицію й викликає ріст курсу (тенденцію росту або висхідний тренд), то рано або пізно він насичується й з деякого моменту пропозиція починає перевищувати попит, приводячи до продажу валюти, що виражається в деякому зниженні її курсу - корекції тренда (trend correction).

Корекція тренда: рух, спрямований проти напрямку попереднього тренда не переважаюче по величині попередній тренд. Якщо розглядати тренд як рух

викликане зміною факторів, що лежать в основі попиту та пропозиції, то корекція - це явище, що повертає ціни в «правильне русло», що не дає ринковому руху сильно «захопитися» і відхилитися від фундаментальних факторів.

Іноді це явище можна також охарактеризувати як «узяття прибутку» (profit taking), що позначає на висхідному тренді, що частина учасників ринку, що відкривали довгі позиції (позиції на покупку), задоволені ростом курсу валюти й починають масово закривати позиції - продавати валюту з метою скоріше зафіксувати свій прибуток. Однак якщо основні причини, що викликають підвищений попит на валюту, не змінилися, то покупка валюти відновляється знову, і курс підвищується, перевищуючи попереднє максимальне значення, тоді рух курсу здобуває напрямок або тренд.

Як вже було вказано, розрізняють основні види тренду:

1. Висхідний тренд (ascending trend, бичача тенденція) − щораз у хвилі досягається більше високе в порівнянні з попереднє значення курсу - ціновий рух при якому кожен наступний локальний максимум і локальний мінімум вище попереднього. Нижні крапки хвиль (локальні мінімуми) з'єднуються прямою лінією лінією тренда (рис. 1.7):

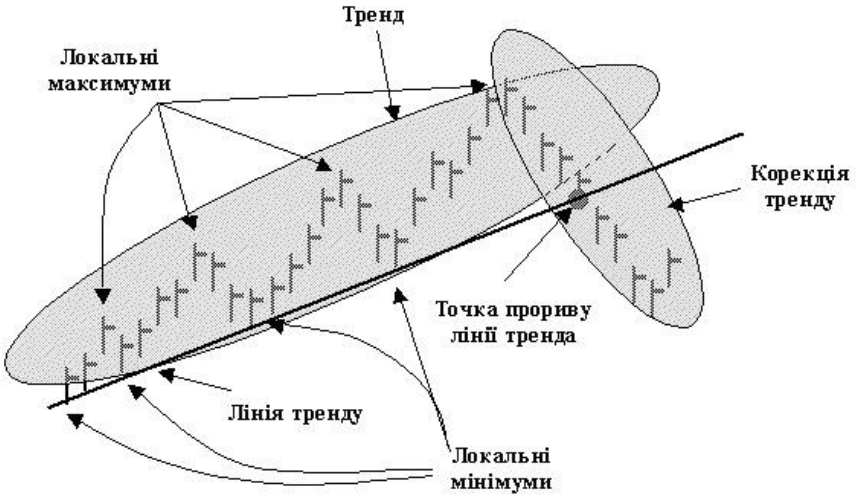

Рис. 1.7. Висхідний тренд

2. Спадний тренд (descending trend, ведмежа тенденція) - щораз досягається більше низьке значення курсу - ціновий рух при якому кожен наступний локальний максимум і локальний мінімум нижче попереднього. Ліния тренда будується шляхом з'єднання верхніх крапок хвиль руху курсу (локальних максимумів) (рис. 1.8):

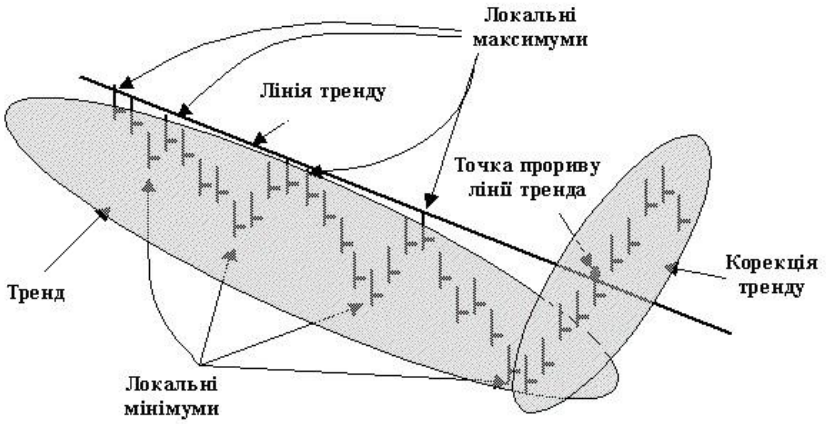

Рис. 1.8. Спадний тренд

Одним із критеріїв сили тренда є його реакція на рівні опору й підтримки. Пробиття рівня підтримки/опору означає, що домінуючий тренд зберігає свою силу. А чим більша кількість разів тренд натрапляє на свій опір або підтримку, будучи не в змозі їх перебороти, тим більше сильний сигнал про слабості тренда ми одержуємо, і тем більше ймовірність розвороту в майбутньому.

Існує ряд загальних правил визначення сили тренда:

– чим довше за часом зберігається тренд, тим він сильніше, однак у нього є межа;

– чим тренд крутіше й швидше, тим він сильніше;

– тривалий пологий тренд має більші шанси на своє продовження;

– дуже крутий тренд може також круто перевернутися;

– будь-який тренд згодом слабшає, однак імовірність продовження тренда в будь-якій його крапці вище ймовірності його розвороту.

Зміна тренда (reversal) виражається в зміні напряму руху курсу після крапки прориву (penetration point). Однак її варто відрізняти від простого нестандартного відхилення курсу, не ведучого до зміни тренда.

Крапка прориву для підтвердженої зміни висхідного тренда утворить сигнал до продажу (sell signal), підтверджена зміна спадного тренда утворить сигнал до покупки (buy signal).

Найкраще підтвердження зміни тренда можна одержати тоді, коли колишня лінія опору стає лінією підтримки тренда й навпаки (рис. 1.9).

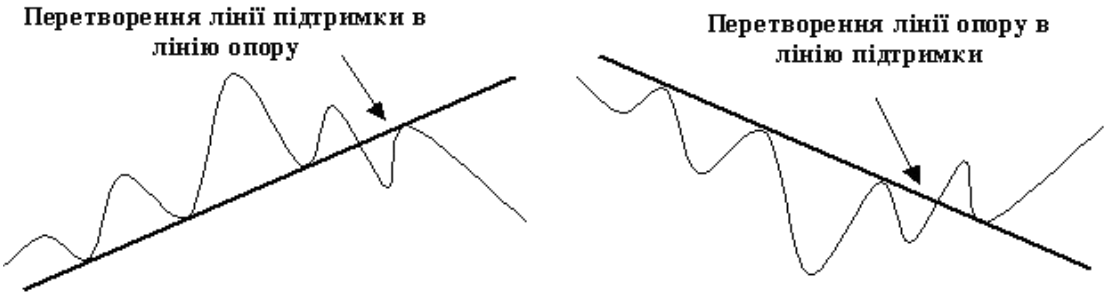

Рис. 1.9. Лінії опору та підтримки

Приклад переходу лінії підтримки в лінію опору (рис. 1.10):

Рис 1.10. Перехід лінії підтримки в лінію опору

1.1.5. Канал (Channel) і торговельний діапазон (Trading range)

Досить часто коливання курсу валюти проходять у певному коридорі - каналі. У цьому випадку нижню границю утворить рівень підтримки (support line), а верхню - рівень опору (resistance line). Відстань між цими рівнями звичайно називають торговельний діапазон каналу або ширина каналу (trading range).

Канали досить умовно можна розділити на висхідні (uptrend channel) (рис. 1.11, 1.12), спадні (downtrend channel) (рис. 1.13, 1.14) і бічні (range) (рис. 1.15, 1.16).

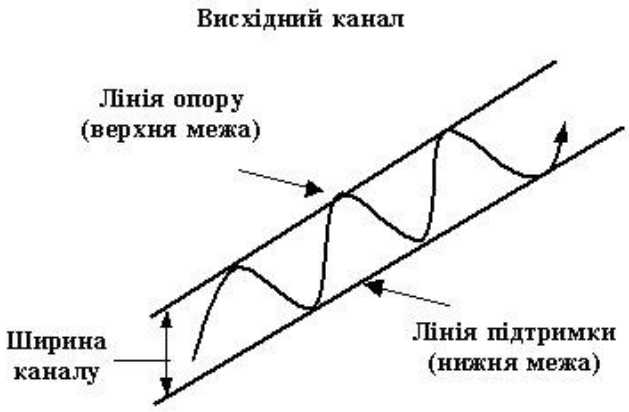

Рис 1.11. Висхідний канал

Рис. 1.12. Приклад висхідного каналу

Основна стратегія трейдерів, що проводять спекулятивні операції в рамках висхідного каналу полягає в тому, щоб купувати на локальних мінімумах і продавати на локальних максимумах.

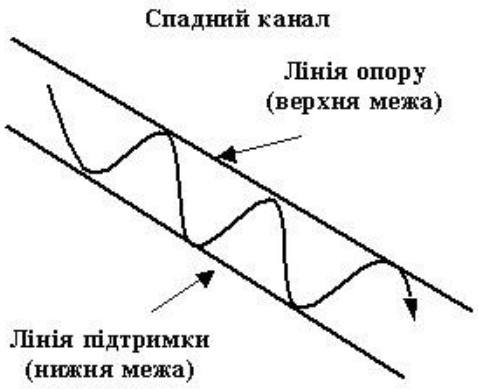

Рис. 1.13. Спадний канал

Рис. 1.14. Приклад спадного каналу

Основна стратегія трейдерів, що проводять спекулятивні операції в рамках спадного каналу полягає в тому, щоб продавати валюту локальних максимумах і відкуповувати її на на локальних мінімумах.

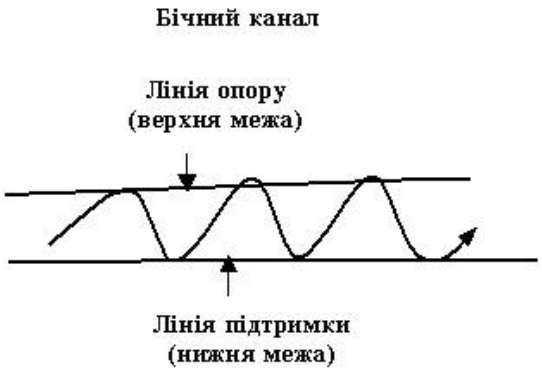

Рис 1.15. Бічний канал

Рис. 1.16. Приклад бічного каналу

Бічний, або горизонтальний канал характеризується відсутністю яскраво вираженого висхідного або спадного тренда курсу валюти. При бічному каналі ціни коливаються в діапазоні, в околиці свого середнього значення.

Стратегії вироблені для висхідного й спадного каналу застосовні також і для бічного каналу, що іноді просто називають торговельним діапазоном.

1.1.6. Лінії підтримки й опору

Лінія, яку ціна не може пробити нагору, називається лінією опору (Resistance), а лінія, що ціна не може пробити вниз, називається лінією підтримки (Support).

Лінія опору (resistance line) валютного курсу - це ні що інше як психологічний рівень, що значна кількість трейдеров розглядає як крапку,

вигідну з погляду продажу валюти. Така єдність думок у великої групи людей приводить до того, що при наближенні цін до лінії або рівня опору значна кількість учасників ринку починає продавати валюту ордерами по ринку, або за допомогою розміщених заздалегідь відкладених ордерів, що знаходять на виконанні в дилерів.

Масове розміщення ордерів й як результат масові продажі як би відштовхують курс від невидимої лини, до якої він підходить знизу (рис. 1.7).

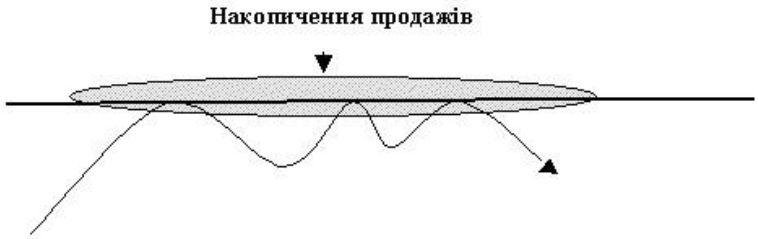

Рис. 1.17. Лінія опору – накопичення приказів на продаж

Лінія підтримки, навпаки, розглядається трейдерами як рівень, на якому валюту вигідно купувати по яким або причинах, тому тут відбуваються покупки по поточних ринкових цінах і спрацьовують ордера на покупку валюти, установлені заздалегідь валютними трейдерами й находящиеся на виконанні в дилерів (рис. 1.18).

Рис. 1.19. Лінія підтримки – накопичення приказів на закупівлю

Як правило, скупчення наказів розміщаються не на одному рівні, а в деякій зоні цін, утворюючи не чіткий рівень підтримки або опору, а зону підтримки або опору.

Розміщення однорідних наказів у тих самих зонах відбувається тому, що різні трейдери, використовуючи подібні правила технічного аналізу, розраховують приблизно однаковий рівень курсу для продажів і покупок, результатом чого є зворотний зв'язок між методами технічного аналізу й характером ринкових рухів.

Коли ціни виходять із бічного діапазону або з каналу, пробиваючи одну з ліній (підтримку або опір), як правило, відбувається прискорення руху. Тому при пробитті лінії опору виникає сигнал на покупку, а при пробитті лінії підтримки сигнал на продаж. Ці сигнали ставляться як до відкриття, так і до закриття позицій.

1.2. Призначення розробки та область застосування

Область застосування інвестиційних проектів дуже велика: будь то прийняття рішень на основі поточної діяльності фірми, аналізу кредитоспроможності клієнтів банку, оцінка ризику страхових операцій, прогнозування успіху біржової діяльності або багато інших задач. Корпоративні чи інституціональні інвестори можуть отримувати значниц прибуток при хеджуванні за допомогою валютних ф’ючерсів. Торгівельні операції з валютними ф’ючерсами можуть допомогти транснаціональним корпораціям компенсувати валютний ризик, пов’язаний з позицією інвестування шляхом купівлі або продажу з розрахунком в іноземній валюті або шляхом експортно-імпортних операцій.

Сучасні програмні засоби зводять до мінімуму втрати трейдера, та допомагають йому поповнити свій капітал. Вони повинні своєчасно повідомляти про наявність збиткових позицій, в результаті чого звільнені активи можна розміщувати більш вигідно; ретельно стежити за станом ціни на ринку, з метою своєчасного повідомлення трейдера про зміну стана ціни і винесення пропозиції щодо подальших кроків для отримання максимальног

прибутку; ретельно стежити за фінансовим портфелем учасника ринка з метою своєчасного корегування інвестиційної стратегії трейдера.

Програмне забезпечення може бути використане у довільній області, де потрібен аналіз даних котирувань валют або цін на основні стратегічні товари.

Розроблена програмна система застосована для обробки даних котирувань курсів валют або цін на основні стратегічні товари, згладжування шумів даних, визначення тренду, побудови індикаторів та осциляторів для детального аналізу ринку.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка програмного забезпечення реалізації алгоритмів аналізу та прогнозування на ф‘ючерсних ринках» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

Темою кваліфікаційної роботи бакалавра є «Розробка програмного забезпечення реалізації алгоритмів аналізу та прогнозування на ф‘ючерсних ринках»

Виходячи з актуальності задачі алгоритмізації обробки даних на ф’ючерсних ринках, необхідно розробити програмне забезпечення на базі сучасних обчислювальних методів та засобів.

При створенні даного програмного забезпечення необхідно реалізувати наступне:

1. Методи згладжування даних: за допомогою медіанного згладжування, локальних поліноміальних сплайнів на основі B-сплайнів.
2. Методи трендового сплайн-аналізу.
3. Використати методи технічного аналізу ф’ючерсних ринків.

4. Розробити програмне середовище для реалізації аналізу даних за допомогою вищезгаданих методів.

5. Провести тестування даної системи.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для реалізації поставленого завдання необхідно розробити програмне забезпечення аналізу ф’ючерсного ринку

Структура програми має складатися з наступних частин:

1. Блок введення та згладжування даних.
2. Блок аналізу.

Блок введення та згладжування даних повинен дозволяти вводити дані з файлу для програми, де відбувається згладжування даних, будуються графічні моделі даних. Передбачити згладжування даних декількома способами. Результати згладжувань різними способами необхідно відобразити на графіках.

Блок аналізу повинен реалізувати обрані методи технічного аналізу: побудову індикаторів та осциляторів. В даному блоці повинен бути реалізований і трендовий сплайн-аналіз.

1.5.2. Вимоги до інформаційної безпеки

Для забезпечення цілісності роботи системи необхідно забезпечити перевірку виконання ситуацій, що можуть призвести до виникнення помилок. А саме:

- перевірка на існування відкритого екземпляру програмного додатку;
- перевірка введення некоректних даних;
- перевірка при видаленні даних;
- перевірка на існування даних при введенні;

– запобігання помилок для компонентів;

– використання обробників подій.

1.5.3. Вимоги до складу та параметрів технічних засобів

Рекомендовані вимоги до апаратного забезпечення:

– процесор Intel Pentium CPU N3540, 2.16GHz або краще;

– місце на жорсткому диску 2 Гб на початковому етапі без урахування операційної системи;

– обсяг оперативного запам’ятовуючого пристрою 1 Гб і більше.

1.5.4. Вимоги до інформаційної та програмної сумісності

Дане програмне забезпечення повинно бути розроблене засобами середовища C++ в середовищі візуального програмування Visual Studio 2017 і функціонувати в операційному середовищі родини Windows.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Метою кваліфікаційної роботи є розробка програмного забезпечення засобів згладжування, обробки та аналізу даних на ф‘ючерсному ринку.

У роботі представлено програмне забезпечення реалізації алгоритмів аналізу та прогнозування на ф‘ючерсних ринках, яке реалізоване та може бути використане в роботі з курсами котирувань валют, цінами на стратегічні товари, тощо.

Програма розбита на два основних блока: блок введення та згладжування даних та блок аналізу.

Блок введення та згладжування даних дозволяє вводити дані з файлу для програми, відбувається згладжування даних, будуються графічні моделі даних. Дані можуть бути згладжені декількома способами. Результати згладжувань різними способами можна переглянути на графіках.

Блок аналізу реалізує обрані методи технічного аналізу : побудову індикаторів та осциляторів. Сюди ж відноситься й реалізація трендового сплайн-аналізу.

Розроблена програмна система застосована для обробки даних котирувань курсів валют або цін на основні стратегічні товари, згладжування шумів даних, визначення тренду, побудови індикаторів та осциляторів для детального аналізу ринку.

2.2. Опис застосованих математичних методів

2.2.1. Технічні індикатори

2.2.1.1. Просте ковзне середнє

Ковзне середнє ставиться до класу індикаторів, що випливають за трендом, воно допомагає визначити початок нової тенденції і її завершення, по його куті нахилу можна визначити силу (швидкість руху), воно ж як основу (або фактору, що згладжує) застосовується у великій кількості інших технічних індикаторів. Іноді ковзне середнє називають лінією тренда.

Будують ковзне середнє, користуючись формулою

$$Simple\ Moving\ Average = \frac{\sum_{i=1}^{n} P_i}{n}, \qquad (2.1)$$

де P_i - ціни на ринку (звичайно беруться ціни Close, але іноді використають Open, High, Low, Median Price, Typical Price);

n - основний параметр - довжина згладжування (кількість цін вхідних у розрахунок ковзні). Іноді цей параметр називають порядком ковзного середнього.

Просте ковзне середнє є звичайним арифметичним середнім від цін за певний період. Ковзне середнє являє собою показник ціни рівноваги (рівновага попиту та пропозиції на ринку) за певний період – чим коротше ковзне середнє, тим за менший період береться рівновага. Згладжуючи ціни, воно завжди йде за головною тенденцією ринку, фільтруючи дрібні коливання. Чим менше параметр ковзного середнього (говорять, що ковзне середнє коротше), тим воно швидше визначає нову тенденцію, але й одночасно робить більше помилкових коливань, і навпаки чим більше параметр (говорять довге ковзне середнє), тим повільніше визначається новий тренд, але надходить менше помилкових коливань.

Ковзні середні не спрогнозують зміни в тренді, а лише просигналять про тренд, що вже з'явився. Оскільки ковзні є індикаторами, що йдуть за трендом, то їх краще використовувати в періоди тренда, а коли на ринку тренд не присутній, вони стають абсолютно неефективними.

Виходячи з того, які ковзні спрямовані нагору, а які вниз визначають який тренд висхідний а який спадний (короткостроковий, середньостроковий, довгостроковий).

Моменти найбільшої розбіжності двох середніх з різними параметрами розуміють як сигнал до можливої зміни тренда.

2.2.1.2. Зважене ковзне середнє

Зважене ковзне (Weighted Moving Average) є звичайною модифікацією звичайного ковзного середнього с вагами підібраними таким чином, що останні ціни мають більшу вагу. Будується наступним чином:

$$Weighted\ Moving\ Average = \frac{\sum_{i=1}^{n} P_i \times W_i}{\sum_{i=1}^{n} W_i}, \qquad (2.2)$$

де P_i - значення ціни і-періодів назад, (і сьогодні =1) ;

W_i - значення ваг для ціни і-періодів назад.

Ваги можуть вибиратися різним шляхом, наприклад, у випадку лінійно-зваженого ковзного середнього:

$W_i = |i - n - 1|$ - ваги ковзного середнього підбираються так, що останні ціни мають максимальну вагу, а далекі - мінімальний.

Ваги можуть також мінятися по логарифмічній або параболічній функції. Ціни також можуть бути різними: Close, Open, High, Low, Median Price, Typical Price.

Як аналітичний інструмент зважене ковзне середнє знімає частину недоліків звичайного ковзного, але не усуває їх повністю. Існує множина модифікацій зваженого ковзного, що використають різні варіанти розрахунку ваг.

2.2.1.3. Експонентне ковзне середнє

Експонентне ковзне середнє (Exponential Moving Average) зменшує лаг, надаючи більшу вагу останнім цінам у порівнянні з більш далекими цінами. Це дозволяє більш швидко реагувати на поточні зміни ціни в порівнянні зі звичайним ковзним середнім. Вага останньої ціни залежить від періоду ковзної середньої. Наскільки коротший період Exponential Moving Average, настільки більша вага буде надаватися останній ціні. Наприклад, 10-перідне Exponential Moving Average дає вагу останній ціні в 18,18%, у той час як 20-періодне - 9,25%.

Експонентна ковзна середня може бути визначена двома шляхами - як процентне ковзне середнє або як періодне ковзне середнє. Відповідно в процентному ковзному, єдиним параметром є вага (відсоток), а в періодном - період ковзного середнього.

Основна формула виглядає в таким чином:

$$EMA_i = EMA_{i-1} + (K \times [P_i - EMA_{i-1}]), \qquad (2.3)$$

де

i - сучасний момент часу;

$(i-1)$ - попередній момент часу;

$K = \frac{2}{n+1}$, n - період середньої в барах.

Необхідно відзначити, що теоретично в розрахунку цієї ковзної використаються всі ціни, за весь період її побудови й, незважаючи на те, що вплив старих цін зникає згодом, воно не зникає до кінця. Ефект старих цін зникає швидше для більше коротких EMA, у порівнянні з більш довгими.

2.2.1.4. Смуги Болліджера (Bollinger Bands)

Смуги Болліджера (Bollinger Bands) − можна визначити як індикатор, схожий з конвертами ковзне середніх, але заснований на поточної волатильності ринку. На відміну від конвертів ковзних середніх, смуги Болліджера змінюються не тільки залежно від напрямку руху ціни, але й від характеру цього руху (швидкості руху ціни).

Смуги Болліджера дають статистичну оцінку того, як далеко може піти короткостроковий рух перш, ніж він повернеться в русло основної тенденції.

Мірою волатильності в смугах Болліджера є середнє-квадратичне відхилення (σ).

Індикатор складається із трьох смуг:

– центральна смуга - являє собою ковзне середнє, котре може бути звичайним, зваженим, експонентним або іншого типу $Central\ Line = Moving\ Average$;

– верхня смуга являє собою $Upper\ Band = Moving\ Average + (K\sigma)$;

– нижня смуга обчислюється:

$$Lower\ Band = Moving\ Average - (K\sigma), \qquad (2.4)$$

де $\sigma = \sqrt{\dfrac{\sum_{j=1}^{N}(P_j - \bar{P})^2}{N}}$,

де P - ціна активу;

N - кількість періодів для розрахунку.

В індикаторі Болліндже́ра відстань смуг від центральної лінії залежить від поводження цін. Коридори смуги індикатора Болліндже́ра називаються коридорами стандартного відхилення, відповідно ширина смуги пропорційна середньоквадратичному відхиленню значення ціни від заданого порядку ковзного середнього для досліджуваного періоду часу.

Відстань між смугами обумовлюється стандартним відхиленням цін, смуги розширюються, коли волатильність цін збільшується, і звужуються, коли їх волатильність зменшується. Якщо смуга звужується, то ми спостерігаємо бічний рух ринку. Якщо смуга розширюється, то ми маємо спрямований рух ринку нагору або вниз - тобто тренд. Згідно положення середньої лінії щодо ціни можна робити висновок про напрямок ринку.

Джон Болліндже́р виділив наступні характеристики свого індикатора:

– рух графіка ціни із зовнішньої сторони смуг, може сигналізувати про продовження тренда;

– якщо за підйомами й падіннями за межами смуги випливають підйоми й падіння усередині смуги, можливий розворот тренда;

– рух ціни, що розпочався від однієї з границь смуги, прагне досягти протилежної смуги;

– різкі коливання цін звичайно відбуваються після звуження смуги, що відповідає зниженню волатильності. Досвід показує, що сильні рухи цін відбувалися після звуження смуг приблизно до того самого рівня.

2.2.1.5. Конверти ковзних середніх

Конверти ковзних середніх (Moving Average Envelopes), як і смуги Болліндже́ра використовується для визначення границь поточного руху ціни.

Індикатор складається із трьох смуг:

Верхня смуга:

$$Moving\ Average\ Upper\ Band = Moving\ Average + (\frac{K \times Moving\ Average}{100}), \quad (2.5)$$

де К - відсоток від ціни, на який ковзне зміщається нагору .

Нижня смуга:

$$Moving\ Average\ Lower\ Band = Moving\ Average - (\frac{N \times Moving\ Average}{100}), \quad (2.6)$$

де N - відсоток, на який ковзне середнє зміщається вниз.

Принцип використанням смуг наступний: ціна завжди після деяких коливань повертається до свого основного тренда (до своєї центральної ковзної середнього). Це відбувається у зв'язку з тим, що чим більше ціна відрізняється від своєї основної тенденції, тим більше трейдерів фіксують прибуток, повертаючи ціну в "нормальне русло".

Чим більше волатильність аналізованого ринку, тим більше границі смуг необхідно вибирати.

Після ретельного підбора коефіцієнта зсуву використання верхньої лінії як лінія опору, а нижньої як лінія підтримки. Відповідна цьому тактика: продаж у верхньої лінії й покупка в нижньої.

2.2.1.6. Індикатор середнього спрямованого руху

Індикатор середнього спрямованого руху - Average Directional Index (ADX) дозволяє аналізувати тенденції ринку й приймати торговельні рішення, у тому числі й на ринку Forex.

Розраховується додатній і від’ємний спрямовані рухи (Directional Movement або DM) - $+DM_j$ й $-DM_j$

Якщо $High_j$ (максимум поточного бару) > $High_{j-1}$ (максимум попереднього бару), то $+DM_j = High_j - High_{j-1}$, інакше $+DM_j = 0$;

Якщо Low_j (мінімум поточного бару) < Low_{j-1} (мінімум попереднього бару), то $-DM_j = Low_{j-1} - Low_j$, інакше $-DM_j = 0$;

Якщо $+DM_j > -DM_j$, то $-DM_j = 0$;

Якщо $+DM_j < -DM_j$, то $+DM_j = 0$;

Якщо $+DM_j = -DM_j$, то $\begin{cases} +DM_j = 0; \\ -DM_j = 0. \end{cases}$

Визначається діапазон - TR_j (True Range) :

$$TR = \max\{|High - Low|, |High - Close_{j-1}|, |Low - Close_{j-1}|\}, \quad (2.7)$$

де $Close_{j-1}$ - ціна закриття попереднього періоду.

Визначається індикатор додатного напрямку й індикатор від'ємного напрямку - $+DI_j$ та $-DI_j$ (Directional Index).

$$\begin{aligned} +DI_j &= Exponential\ Moving\ Average_j(+SDI, N) \\ -DI_j &= Exponential\ Moving\ Average_j(-SDI, N) \end{aligned} \quad (2.8)$$

якщо $TR_j \neq 0$, то $\begin{cases} +SDI_j = \dfrac{+DM_j}{TR_j} \\ -SDI_j = \dfrac{-DM_j}{TR_j} \end{cases}$;

якщо $TR_j = 0$, то $\begin{cases} +SDI_j = 0; \\ -SDI_j = 0, \end{cases}$

Визначається середній спрямований рух - ADX_j.

$$ADX_j = Exponential\ Moving\ Average_j(DX, N), \quad (2.9)$$

де DX_j обчислюється по формулі $DX_j = \frac{{}^{+}DI - {}^{-}DI}{{}^{+}DI + {}^{-}DI} \times 100$.

Фактично ADX відноситься до класу осциляторов, що коливається в діапазоні від 0 до 100. Незважаючи на те, що рух індикатора перебуває в діапазоні від 0 і до 100, рух вище оцінки в 60 відбувається досить рідко. Значення нижче 20 сигналізує про слабкий тренд, значення вище 40 сигналізує про сильний тренд. Дані вище 40 можуть показувати, як сильний спадний, так і сильний висхідний тренд. ADX також може використатися для визначення потенційних змін на ринку. Коли показання індикатора переходять границю 20 знизу нагору - це може говорити про зміну тренда й подальшому його розвитку. Коли індикатор показує значення менше 40, падаючи з більше високого рівня , це може означати те, що тренд втратив силу.

2.2.2. Осцилятори

2.2.2.1. Середній правдивий діапазон

Середній правдивий діапазон - Average True Range (ATR) є індикатором волатильності.

$$ATR = Moving\ Average(TR, n), \quad (2.10)$$

де $TR = \max\{|High - Low|, |High - Close_{j-1}|, |Low - Close_{j-1}|\}$

Середній правдивий діапазон (ATR) виводиться з TR шляхом усереднення по якому-небудь із методів – звичайному середньому, експонентному або іншому.

Екстремальні значення індикатора часто вказують на розворотні крапки й або початок нового руху. Як й інші індикатори що показують волатильність, як,

наприклад смуги Болліндже́ра (Bollinger Bands), Average True Range не може пророчити напрямок або тривалість руху, він вказує тільки на рівень активності.

Низький рівень індикатора означає тиху торгівлю в невеликому діапазоні, а високі значення позначають інтенсивну торгівлю в широкому діапазоні. Довгий період низького ATR позначає консолідацію, що, швидше за все, приведе до швидкого продовження руху або розвороту. Високі значення ATR звичайно є наслідком швидких рухів і рідко залишаються на тривалий період. Оскільки ATR показує абсолютне значення волатильність, то валютні пари на Forex с низькими цінами будуть за інших рівних умов мати більш низький ATR і навпаки.

Основну концепцію індикатора можна виразити в такий спосіб: чим більше значення індикатора, тим більше ймовірність розвороту тренда; чим менше значення індикатора, тим слабкіше спрямованість тренда.

Як недоліки звичайно вказується один - при великому періоді ATR може запізнюватися, указуючи не поточну а минулу волатильність.

2.2.2.2. Індекс товарного каналу

Індекс товарного каналу (Commodity Channel Index - CCI) створений, як індикатор для визначення разворотных крапок на товарних ринках, однак згодом він став користуватися популярністю на ринку акцій і на ринку Forex. Припущення, на якому базується індикатор полягає в тому, що всі активи рухаються під впливом певних циклів, а максимуми й мінімуми з'являються з певним інтервалом.

CCI ставиться до типу осциляторів, що вимірюють швидкість руху ціни. Спочатку розраховується характерна ціна (Typical Price):

$$Typical\ \ Price(TP) = \frac{high + low + close}{3} \quad . \qquad (2.11)$$

Розраховується просте ковзне середнє від характерної ціни:

$$Simple\ Moving\ \ Average\ (TP) = \frac{\sum_{i=1}^{n} TP_i}{n}. \qquad (2.12)$$

Розраховується відхилення:

$$Mean\ Deviation = \frac{\sum_{i=1}^{n} |\ \overline{TP} - TP_i\ |}{N}, \qquad (2.13)$$

де $\overline{TP} = Simple\ Moving\ \ Average\ (TP)$.

Формула самого Індексу товарного каналу буде виглядати в такий спосіб:

$$Commodity\ Chanel\ \ Index = \frac{TP_i - \overline{TP}}{0,015 \times Mean\ Deviation}. \qquad (2.14)$$

Сам CCI коливається вище й нижче нульової оцінки. Відсоток значень Commodity Channel Index, які перебувають між +100 й -100 буде залежати від кількості періодів, використаних для його побудови. Більше короткий (з меншою кількістю періодів) CCI буде більше волатильним, і менше його значень буде попадати в діапазон між +100 й -100. Відповідно, чим більше періодів буде використано для розрахунку CCI, тим більше відсотків значень індикатора буде перебувати між +100 й -100.

CCI є досить різнобічним індикатором, здатним створювати великий спектр сигналів на покупку й продаж. Трейдери й інвестори використають CCI для визначення розворотних цінових крапок, экстремумів на графіку ціни й

сили тренда. Як і більшість індикаторів CCI повинен використатися з підтвердженнями від інших технік.

Рекомендації автора по роботі Індексом товарного каналу стосувалися рухів, які виходять вище +100 і нижче -100 і дають сигнали на покупку й продаж. Оскільки від 70 до 80% часу значення CCI перебувають між рівнями +100 й -100, сигнали на покупку й продаж будуть з'являтися 20 - 30% відсотків часу.

Коли CCI перетинає +100 знизу нагору, уважається, що валютна пара входить у сильний висхідний тренд, і подається сигнал на покупку. Позиція закривається по зворотному сигналі, коли CCI падає нижче +100. Коли CCI перетинає -100 зверху вниз, вважається, що валютна пара йде в сильний спадний тренд, і подається сигнал на продаж. Позиція закривається, коли CCI обернено перетинає свій рівень -100.

Вважається, що валютна пара перепродана, якщо CCI падає нижче -100 і перекуплений, якщо він виростає вище +100. Після того як CCI увійшов у зону нижче -100, сигнал на покупку виникає при перетинанні рівня -100 у зворотному напрямку (знизу нагору). Коли CCI перебуває в зоні вище +100 сигнал на продаж виникає, якщо CCI перетинає рівень +100 у зворотному напрямку (зверху вниз).

Як і на більшості осциляторов, на CCI використається аналіз дивергенцій, що підсилює торговельний сигнал. Позитивна дивергенція нижче -100 (тобто два послідовних мінімуми на CCI, коли другий мінімум вище першого на індикаторі, але нижче першого на графіку цін) збільшують силу сигналу при перетинанні ціною рівня -100 знизу нагору. Негативна дивергенція вище рівня +100 (тобто два послідовних максимуми на індикаторі вище +100, коли другий максимум нижче першого на індикаторі, але вище на ціновому графіку) збільшують силу сигналу подаваного при перетинанні CCI рівня +100 зверху вниз.

Пробиття ліній тренда, що утворилися на самому індикаторі теж можуть розцінюватися як сигнали входу або виходу з позиції. Трендові лінії можуть

бути намальовані шляхом з'єднання послідовних максимумів або мінімумів. На рівні нижче -100 пробиття такої трендової лінії нагору вважається сигналом до росту ринку й навпаки на рівнях вище +100 пробиття трендової лінії вниз може бути розцінене як сигнал на продаж.

2.2.2.3. Швидкість зміни

Швидкість зміни (Rate of Change - ROC) − один з найпростіших і дуже ефективних осциляторов, що показує процентну зміну ціни від одного періоду до іншого.

Rate of Change розраховується, як порівняння поточної ціни із ціною минулого періоду, що відстоїть від поточні на N періодів. Періодами як завжди можуть бути інтервали від хвилини до місяця.

$$ROC = 100 \times \frac{P_0}{P_{-n}}, \tag{2.15}$$

де P_0 - ціна закриття поточного періоду;

де P_{-n} - ціна закриття сьогодні N періодів назад;

Як правило Rate of Change, трохи випереджає основну цінову тенденцію й досягає максимуму або мінімуму раніше, ніж це робить ціна.

Перетинання індикатором рівня 100 і подальший його ріст означає прискорення темпів росту ціни й збільшення ймовірності її продовження. Розворот індикатора на високих рівнях зверху вниз означає, що тенденція триває, але вже не набирає силу, а починає поступово сповільнюватися.

Падіння індикатора над лінією 100 означає, що поточна тенденція росту цін сповільнює швидкість.

Перетинання зверху вниз одиниці й продовження падіння означає прискорення спадного руху цін.

Розворот на рівнях нижче одиниці означає вповільнення падіння цін і високу ймовірність розвороту тенденції нагору.

Ріст індикатора нижче лінії 100 означає загасання спадної тенденції.

Вважається, що Rate of Change вимірює рівень оптимізму або песимізму ринку відносно даного активу, якщо індикатор росте, залишаючись вище 100, це означає, що на ринку нова хвиля оптимізму, якщо з'являється новий мінімум у зоні нижче одиниці, це означає, що нова хвиля песимізму на ринку збільшує ймовірність подальшого падіння ціни.

Якщо ціни продовжують рости, роблячи на графіку новий максимум вище попереднього, а ROC росте але його новий максимум нижче попереднього, це означає що з'явилася розбіжність (дивергенція) у показаннях індикатора й ціни, і потрібно готується до можливого падіння цін.

У зворотному випадку, якщо на графіку ціни роблять новий мінімум нижче попереднього, а на графіку індикатора з'являється новий мінімум, але вище попереднього, то потрібно готуватися до розвороту ринку нагору.

Таким чином, нахил індикатора означає прискорення або вповільнення тенденції, а його положення щодо рівня одиниці - яка ця тенденція - падіння або ріст.

2.2.2.4. Індекс відносної сили

Індикатор індекс відносної сили Relative Strength Index (RSI) став одним з найбільш популярних осциляторів, що оцінюють силу руху. Relative Strength Index порівнює величину підйомів ціни активу за останнім часом з величиною її падінь і надає цю інформацію у вигляді числа в діапазоні від 0 до 100. Єдиний параметр індексу відносної сили - часовий період, використаний у розрахунку.

$$RSI = 100 - \frac{100}{1 + RS}; \qquad RSI = \frac{CU}{CD}, \tag{2.16}$$

де

CU(n) – середнє значення позитивних змін ціни закриття;

CD(n) – середнє значення негативних змін ціни закриття;

$$CU = \begin{cases} \dfrac{\sum_{j=1}^{n}(C_j - C_{j-1})}{N}, C_j > C_{j-1} \\ \dfrac{\sum_{j=1}^{n}(C_{j-1} - C_j)}{N}, C_j < C_{j-1} \end{cases}. \quad (2.17)$$

Необхідно відзначити, що для періоду RSI рівного 14, середнє значення додатних змін ціни закриття (і середнє значення від'ємних змін ціни закриття) ділиться на 14, навіть якщо самих цих позитивних (або негативних) відхилень усього 5.

Середнє значення додатних змін ціни закриття й середнє значення від'ємних змін ціни закриття не є дійсними ковзними середніми. Замість ділення на кількість позитивних і негативних змін обидва показники в знаменнику мають загальну кількість періодів для розрахунку RSI.

Коли середнє значення додатних змін ціни закриття більше чим середнє значення від'ємних змін ціни закриття RSI росте, оскільки значення RS більше одиниці, відповідно, коли середнє значення додатних змін ціни закриття менше, ніж середнє значення від'ємних змін ціни закриття RSI падає, оскільки RS менше одиниці.

Як й у більшості осциляторів, чим більше даних використовується для розрахунку RSI (більше період RSI), тим більше точними будуть результати. У більшості випадків для аналізу ринку по RSI використовується типовий метод зон перекупленості й перепроданості осциляторів. Автор (Уайлдер) рекомендує використовувати для визначення зон перекупленості й перепроданості відповідно зони вище 70 і нижче 30. Зона вище 70 у такий спосіб по авторі

показує, що ринок перекуплений (тобто подальший рух нагору незабаром вичерпає себе), а зона нижче 30 вважається зоною перепроданості (тобто подальший рух униз незабаром вичерпає себе). Однак існує багато інших варіацій щодо співвідношення цих рівнів: 75/25 або 80/20. На ринку Forex частіше використається останнє співвідношення рівнів. У якості центральної, звичайно розглядають рівень 50. Однак і тут існують варіації. Так, іноді на висхідному тренді центральну лінію й рівні перекупленості перепроданості зміщають нагору, а на спадному тренді - униз.

Relative Strength Index в основному використається на ринках, що перебувають у бічному тренді, оскільки на ринках, які перебувають у спрямованому тренді, він може використатися лише для визначення крапок входу й виходу усередині тренда, тобто для прогнозування локальних максимумів і мінімумів. Використання Relative Strength Index як основний індикатор на трендовом ринку може привести до великої кількості помилкових сигналів.

2.2.2.5. Індикатор моменту

Індикатор моменту (Momentum indicator) показує швидкість зміни цін, або іншим словами, темп зміни цін, залишаючись при цьому одним з найпростіших й ефективних інструментів технічного аналізу.

$$Momentum\ Simple = C - C_{-n}, \tag{2.18}$$

де C- ціна закриття поточного періоду;

C_{-n} - ціна закриття N періодів назад.

$$\begin{bmatrix} \begin{cases} M1 = Momentum\ Simple \\ M2 = 0 \end{cases}, Momentum\ Simple > 0 \\ \begin{cases} M1 = 0 \\ M2 = -Momentum\ Simple \end{cases}, Momentum\ Simple < 0 \end{bmatrix}, \tag{2.19}$$

$$SM1 = \sum_{i=1}^{n} M1_i$$
$$SM2 = \sum_{i=1}^{n} M2_i \quad , \qquad (2.20)$$

де і - періоди розрахунку.

$$Chandle\ Momentum\ Oscillator = \frac{\sum_{i=1}^{n} M1_i - \sum_{i=1}^{n} M2_i}{\sum_{i=1}^{n} M1_i + \sum_{i=1}^{n} M2_i} \times 100 \quad . \qquad (2.21)$$

Momentum являє собою найпростіший трендовий індикатор. Перевищення поточних цін закриття над минулими, показує висхідний тренд, а якщо поточні ціни нижче минулих за обраний часовий інтервал, це означає спадний тренд.

Іноді використається згладжений моментум, щоб зменшити його волатильність. Для цих цілей застосовуються ковзні середні.

Індикатор у більшості випадків є випереджальним стосовно основного цінового руху. Перед вершиною тренда індикатор розвертається (ціни при цьому звичайно продовжують рости) і починає падати, тільки після цього звичайно тренд розвертається вниз. Перед мінімумом ринку навпаки, індикатор сповільнює своє падіння (ціни при цьому звичайно продовжують падати, але більше повільними темпами), розвертається й починає рости, після чого тренд розвертається нагору.

Індикатор Momentum може використатися в якості осцилятора (хоча таким не є в класичному розумінні). Сигнал на продаж з'являється, коли Momentum або його ковзне середнє піднімається на істотну величину, розвертається й починає падати. Сигнал на покупку з'являється, коли Momentum або його ковзне середнє падає на значну величину, розвертається й

починає підніматися нагору. Ця "значна величина" буде розрізнятися на кожній валютній парі на Forex й її потрібно тестувати окремо.

Індикатор Momentum може використовуватися на Forex у якості трендового індикатора, у цьому випадку як сигнал використаються перетинання його нульової лінії. Коли лінія перетинається зверху вниз, подається сигнал на продаж, коли знизу нагору, подається сигнал на покупку.

Недоліки в індикатора моментум, такі ж як й у багатьох інших індикаторів: він швидко змінюється при вході в розрахунок нових цін, що сильно відрізняються від основного русла. Таке поводження дає нестабільність результатів.

2.3. Опис використаної архітектури та шаблонів проектування

Під час розробки програмного забезпечення використаний об'єктно-орієнтований шаблон проектування, який відображає відношення між класами та об'єктами, без вказівки на те, як буде зрештою реалізоване це відношення. Докладніше опис структури та компонентів системи розкривається в п. 2.5.2.

2.4. Опис використаних технологій та мов програмування

Найбільш поширеною мовою програмування протягом декількох останніх десятиліть, безперечно, є мова C ++, на підставі якої «виросло» багато сучасних мов програмування і програмних середовищ. Цьому сприяли такі її властивості, як лаконічність, потужність, гнучкість, мобільність, можливість доступу до всіх функціональних засобів системи. Програмувати на C ++ можна як для Windows, так і для Unix.

C++ - універсальна мова програмування високого рівня з підтримкою декількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Розроблена Б'ярном Страуструпом в AT&T Bell Laboratories у 1979 році та названа «Сі з класами». Страуструп перейменував мову у C++ у

1983 р. Базується на мові Сі. Визначена стандартом ISO/IEC 14882:2011 [19].

Мова програмування С++ є дуже зручною у розробці прикладних програм; драйверів пристроїв; розробці ОС; відеоігор. Програми, складені мовою С++, є мобільними, тобто можуть бути виконані на комп'ютерах різних виробників і в різних операційних системах, завдяки чому С++ є особливо популярною.

Реалізацією мови С++ займаються одночасно декілька проєктів як безкоштовних, так і комерційних, а саме: GNU, Microsoft і Embarcadero.

С++ - високорівнева мова програмування загального призначення, поєднує властивості як високорівневих, так і низькорівневих мов програмування. Від свого попередника, мови програмування С, С++ відрізняється тим, що найбільшу увагу при розробці цієї мови було приділено підтримці об'єктно-орієнтованого та узагальненого програмування.

У 1990-х роках С++ стала однією з найуживаніших мов програмування загального призначення [5].

Стандартна бібліотека С++ включає стандартну бібліотеку Сі з невеликими змінами, які роблять її відповіднішою для мови С++. Інша велика частина бібліотеки С++ заснована на Стандартній Бібліотеці Шаблонів (STL).

Основним способом організації інформації в Сі++ є класи. На відміну від типу структура (struct) мови Сі, що складається тільки з полів, клас Сі++ складається з полів і функцій-членів або методів. Поля бувають публічними (public), захищеними (protected) і приватними (private). У Сі++ тип структура аналогічний типу клас, відмінність в тому, що за умовчанням поля і функції-члени у структури публічні, а у класу — приватні.

Функції-члени, як і поля, можуть бути публічними, захищеними і приватними. Публічні функції може викликати будь-хто, а захищені і власні - тільки функції-члени і друзі [14].

При створенні С++ прагнули зберегти сумісність з мовою С. Більшість програм на С справно працюватимуть і з компілятором С++.

Нововведеннями С++ порівняно з С є:

– підтримка об'єктно-орієнтованого програмування через класи;
– підтримка узагальненого програмування через шаблони;
– доповнення до стандартної бібліотеки;
– додаткові типи даних;
– обробка винятків;
– простори імен;
– вбудовані функції;
– перевантаження операторів;
– перевантаження імен функцій;
– посилання і оператори управління вільно розподіленою пам'яттю [1].

2.5. Опис структури програми та алгоритмів її функціонування

2.5.1. Проектування алгоритмів аналізу даних

2.5.1.1. Згладжування даних за допомогою локальних поліноміальних сплайнів на основі B-сплайнів

Нехай заданий набір із $n+1$ точки $M_0(x_0, y_0), M_1(x_1, y_1), ..., M_n(x_n, y_n)$ на площині, отриманих як набір вхідних даних. Один з найпростіших методів згладжування даних заснований на заміні трьох послідовних точок центром ваги трикутника з вершинами в цих точках.

Цей алгоритм зображений на рис. (2.1), де вхідні дані – це точки $M_0, M_1, ..., M_n$, а згладжені точки - $M_1^*, ..., M_{n-1}^*$.

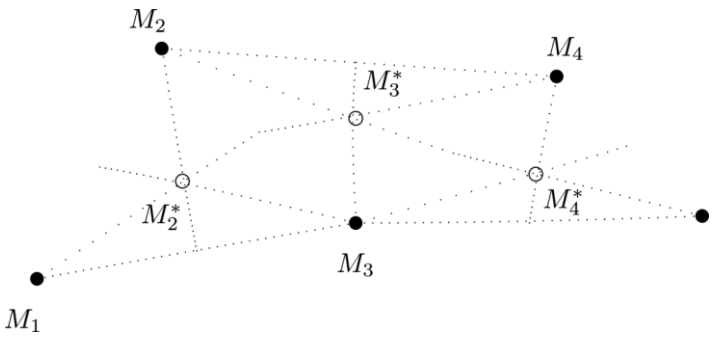

Рис. 2.1. Алгоритм згладжування даних

Тобто точки $M_1^*,...,M_{n-1}^*$ - це точки перетинання медіан трикутників з вершинами в точках $M_0,M_1,M_2;M_1,M_2,M_3$;M_{n-2},M_{n-1},M_n. Ясно, що $M_i^*=\frac{1}{3}(M_{i-1}+M_i+M_{i+1})$ $(i=1,2,....,n-1)$, тобто:

$$\begin{cases} x_i^*=\frac{1}{3}(x_{i-1}+x_i+x_{i+1}) & (i=1,2,....,n-1) \\ y_i^*=\frac{1}{3}(y_{i-1}+y_i+y_{i+1}) & (i=1,2,....,n-1) \end{cases}. \qquad (2.22)$$

Отримані рівності можна переписати в еквівалентному виді

$$\begin{cases} x_i^*=x_i+\frac{1}{3}\Delta^2 x_i & (i=1,2,....,n-1), \\ y_i^*=y_i+\frac{1}{3}\Delta^2 y_i & (i=1,2,....,n-1). \end{cases} \qquad (2.23)$$

У процесі згладжування кількість даних (точок) зменшилась на дві. Як правило, процес згладжування повторюють кілька разів. У результаті ми губимо частину інформації, що приводить до втрати точності при опису контуру в кінців. Щоб цього не відбулося, необхідно поповнити розумним способом вихідні дані й поповнені дані згладити.

Приведемо два способи довизначення даних у тому випадку, коли дані описують незамкнуту криву. Перший спосіб полягає в наступному – точки M_{-1} й M_{n+1} визначаємо з умови

$$M_{-1}M_0=M_0M_1, M_nM_{n-1}=M_{n+1}M_n, \qquad (2.24)$$

тобто:

$$
\begin{aligned}
x_{-1} &= 2x_0 - x_1, \quad x_{n+1} = 2x_n - x_{n-1}, \\
y_{-1} &= 2y_0 - y_1, \quad y_{n+1} = 2y_n - y_{n-1}.
\end{aligned}
\tag{2.25}
$$

Другий спосіб викладений на рис. 2.2.

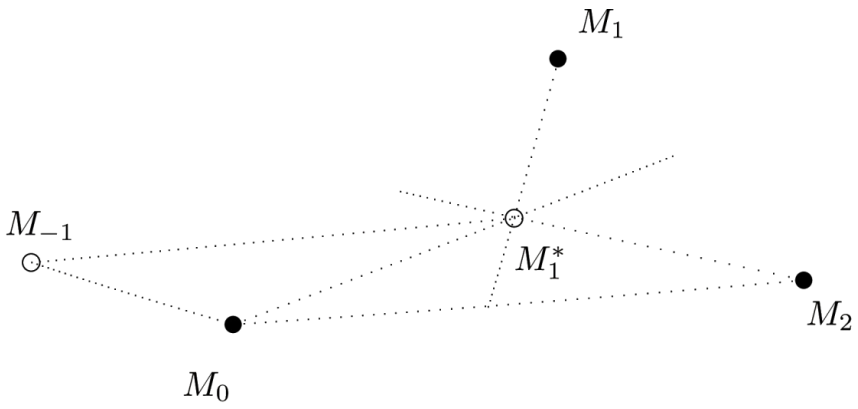

Рис. 2.2. Алгоритм згладжування даних

У цьому випадку точки M_{-1} й M_{n+1} ми вибираємо з умов $M_2M_{-1} = M_2M_1^* + M_2M_0$, $M_{n-2}M_{n-1} = M_{n-2}M_{n-1}^* + M_{n-2}M_n$ або, що те ж,

$$
\begin{aligned}
x_2 - x_{-1} &= x_2 - \frac{1}{3}(x_0 + x_1 + x_2) + x_2 - x_0, \\
x_{n+1} - x_{n-2} &= x_{n-2} - \frac{1}{3}(x_{n-2} + x_{n-1} + x_n) + x_n - x_{n-2}. \\
y_2 - y_{-1} &= y_2 - \frac{1}{3}(y_0 + y_1 + y_2) + y_2 - y_0, \\
y_{n+1} - y_{n-2} &= y_{n-2} - \frac{1}{3}(y_{n-2} + y_{n-1} + y_n) + y_n - y_{n-2}.
\end{aligned}
\tag{2.26}
$$

Відшукуючи з рівностей $x_{-1}, x_{n+1}, y_{-1}, y_{n+1}$, одержуємо формули поповнення

$$\begin{cases} x_{-1} = \frac{1}{3}(4x_0 + x_1 - 2x_2) \\ x_{n+1} = \frac{1}{3}(4x_n + x_{n-1} - 2x_{n-2}) \\ y_{-1} = \frac{1}{3}(4y_0 + y_1 - 2y_2) \\ y_{n+1} = \frac{1}{3}(4y_n + y_{n-1} - 2y_{n-2}) \end{cases} . \qquad (2.27)$$

Слід зазначити, що при малих помилках виміру, алгоритм, наведений вище, може (навіть при однократному застосуванні) «перегладити» дані, спотворивши достовірну інформацію. У таких випадках звичайно застосовують (краще кілька разів) більше м'які алгоритми згладжування, які відрізняються від (2.26) меншим коефіцієнтом при других різницях.

2.5.1.2. Згладжування даних за допомогою локальних поліноміальних сплайнів на основі B-сплайнів

Представимо обчислювальну схему побудови локальних поліноміальних сплайнів на основі B-сплайнів, наведену в роботі [3].

Функція $s(t)$, задана та неперервна на відрізку $[a;b]$, називається поліноміальним сплайном (або просто сплайном порядку) порядку r $(r \geq 1)$ з вузлами t_i, $i = \overline{0,N}$, $a = t_0 < t_1 < ... < t_N = b$, якщо на кожному з проміжків $[a;t_1],[t_i;t_{i+1}]$, $i = \overline{2,N-2}$, $[t_{N-1};b]$, $s(t)$ є алгебраїчний многочлен ступеня, що не перевищує r, а в кожній із точок $t_i, i = \overline{0,N}$ деяка похідна $s^{(v)}(t), (1 \leq v \leq r)$ може мати розрив.

Головними характеристиками сплайну є найбільший порядок r багаточленів із яких він складається, кількість і розташування вузлів та гладкість склеювання в кожному вузлі. Говорять, що сплайн $s(t)$ порядку r має дефект k_i у вузлі t_i $(1 \leq k_i \leq r)$ якщо в точці t_i функції $s(t), s'(t), ..., s^{(r-k_i)}(t)$

неперервні, а похідна $s^{(r-k_i+1)}(t)$ в t_i має розрив. Число $k=\max\limits_{i=0,N} k_i$ називають дефектом сплайну $s(t)$.

Якщо задано фіксовану систему точок $\Delta[a;b]: a=t_0<t_1<t_2<...<t_N=b$, або, іншими словами, задано розбиття відрізку $[a;b]$, то множину $S_r^k(\Delta[a;b])$ називають множиною сплайнів порядку r дефекту k за розбиттям $\Delta[a;b]$.

Множину $S_r^1(\Delta[a;b])$ (або просто $S_r(\Delta[a;b])$) називають множиною сплайнів мінімального дефекту.

Якщо довільним чином доповнити розбиття $\Delta[a;b]$ точками $t_{-r}<t_{-r+1}<...t_{-1}<a;\quad b<t_{N-1}<...t_{N+r}$, у результаті чого одержують систему точок:

$$t_{-r}<t_{-r+1}<...t_{-1}<t_0<t_1<t_2<...<t_N<t_{N-1}<t_{N+r}, \qquad (2.28)$$

тоді для кожного $i=\overline{-r,N-1}$ існує сплайн $B_{r,i}(t)$ порядку r дефекту 1 за розбиттям $t_i<t_{i+1}<...<t_{i+r+1}$, який визначається рівностями:

$$\int_{t_i}^{t_{i+r+1}} B_{r,i}(t)dt=1, B_{r,i}(t)=0,\quad t<t_i; t>t_{i+r+1}, \qquad (2.29)$$

або

$$B_{r,i}(t)=\frac{t-t_i}{t_{i+r}-t_i}B_{r-1,i}(t)+\frac{t_{i+r+1}-t}{t_{i+r+1}-t_{i+1}}B_{r-1,i+1}(t),\ \sum_{i=-r}^{N-1}B_{r,i}(t)=1, \qquad (2.30)$$

де $B_{0,i}(t)=\begin{cases}0, & t\notin[t_i;t_{i+1}]\\ 1, & t\in[t_i;t_{i+1}]\end{cases}$.

Сплайн $B_{r,i}(t) \in C^{r-1}(-\infty;\infty)$, визначений на всій дійсній вісі називається *B*-сплайном порядку r на сітці $t_i < t_{i+1} < ... < t_{i+r+1}$. При цьому відрізок $[t_i;t_{i+r+1}]$, на якому $B_{r,i}(t) > 0$, називають носієм *B*-сплайну.

Має місце ствердження:

система із N+r *B*-сплайнів $B_{r,i}(t), i = \overline{-r, N-1}$ порядку r за розбиттям $t_{-r} < t_{-r+1} < ... t_{-1} < t_0 < t_1 < t_2 < ... < t_N < t_{N-1} < t_{N+r}$ з носієм $[t_i;t_{i+r+1}]$ є базисом в $S_r(\Delta[a;b])$.

Як наслідок, можна зазначити, що будь-який сплайн $s(t) \in S_r(\Delta[a;b])$ єдиним чином можна представити у вигляді;

$$s(t) = \sum_{i=-r}^{N-i} c_i B_{r,i}(t), t \in [a;b], \tag{2.31}$$

де $c_i, i = \overline{-r, N-1}$ - деякі дійсні числа, такі, що $s(t) = \sum_{i=-r}^{N-i} c_i B_{r,i}(t) = 0$, тоді і тільки тоді, коли $c_{-r} = c_{-r+1} = ... = c_{N-1} = 0$.

Нехай при деякому $h > 0$ задано рівномірне розбиття Δ_h дійсної вісі $R_1 = (-\infty;\infty)$ точками ih, $i \in Z$. Множину сплайнів порядку r мінімального дефекту, визначену на розбитті Δ_h позначають $S_r(\Delta_h)$. Тоді, якщо:

$$B_{0,h}(t) = \begin{cases} 1\left(|t| < \frac{h}{2}\right), \\ 0\left(|t| \geq \frac{h}{2}\right), \end{cases} \tag{2.32}$$

то *B*-сплайн $B_{r,h}(t) \in S_r(\Delta_h)$ порядку $r\,(r \geq 1)$ визначається рекурентно із співвідношення

$$B_{r,h}(t)=\frac{1}{h}\int_{t-\frac{h}{2}}^{t+\frac{h}{2}}B_{r-1,h}(\tau)d\tau\,. \qquad (2.33)$$

На розбиття Δ_h B-сплайн порядку r має в якості носія проміжок:

$$d_r=[-\frac{r+1}{2}h,\frac{r+1}{2}h]\text{, отже } \int_{-\infty}^{\infty}B_{r,h}(t)dt=\int_{d_r}B_{r,h}(t)dt=\int_{-\frac{(r+1)h}{2}}^{\frac{(r+1)h}{2}}B_{r,h}(t)dt=h \qquad (2.34)$$

Із визначення B-сплайну та виразів (2.33), (2.34) не важко отримати аналітичні представлення $B_{r,h}(t)$ при різних значеннях r.

Так само, як і у випадку нерівномірного розбиття, щодо B-сплайнів $B_{r,h}(t)$ можна стверджувати: якщо $S_r(\Delta_h)$- множина всіх сплайнів мінімального дефекту за розбиттям Δ_h і $B_{r,h}(t)\in S_r(\Delta_h),\ \ r\geq 1$, то лінійна комбінація $S_r(t)$ сплайнів $B_{r,h}(t)$ також буде належати множині $S_r(\Delta_h)$, тому має місце вираз, аналогічний (2.2) $S_r(t)=\sum_{i\in Z}c_iB_{r,h}(t)\in S_r(\Delta_h)$.

Отже, лінійна комбінація $S_r(t), r\geq 1$ - є сплайн мінімального дефекту. Найчастіше подібні сплайни називають локальними поліноміальними сплайнами на основі B-сплайнів r-го порядку.

2.5.1.3. Трендовий сплайн-аналіз

Відмітною рисою трендового сплайн-аналізу є те, що моделі тренду описуються шматочно-поліноміальними функціями $f(t)\in C^q_{[a;b]}$ на розбитті $\Delta_t: a=T_0<T_1<\ldots<T_k=b$ вигляду:

$$S(t;\vec{\Theta}) = \sum_{i=1}^{k} S_i(t;\vec{\theta}_i) I_i(t) + \varepsilon, \qquad (2.35)$$

де ε - вектор-стовпець випадкових похибок: $E\{\varepsilon\} = 0, D\{\varepsilon\} = \sigma^2 = const$;

$\vec{\theta}_i = \{a_v^{(i)}, v = \overline{0,k}, i = \overline{1,m}\}$ - вектор параметрів;

$$I_i(t) = \begin{cases} 1, t \in [T_{i-1}, T_i] \\ 0, t \notin [T_{i-1}, T_i] \end{cases}$$

Функції $S(t;\vec{\Theta})$ мають властивості:

- $S(t,\vec{\Theta}) \in C^{k-d}_{[a;b]}$

- $\min\limits_{\vec{\theta}_v} E\{S_з^2\}, \quad S_з^2 = \sum\limits_{r=1}^{n} w_r (x_r - S(t_r;\vec{\theta}_v))^2; \quad w_r - ваги\ x_r, r = \overline{1,n}$

Наведена модель тренда (2.35) обгрунтовується:

1. Більш адекватним описанням тренду, отже, рішенням задачі знаходження:

$$\rho_0 = \min_v \min_{\vec{\theta}_v} \sum_{r=1}^{n} w_r (x_r - S(t_r;\vec{\theta}_v))^2; \qquad (2.36)$$

2. Відображенням наявності структурних змін вимірювань у процесі, що спостерігається, тощо.

Найпростішими моделями тренду на $t \in [a;b]$ є параметричні згладжуючи сплайни з одним і двома вузлами. Останнє дозволяє побудувати ефективні обчислювальні процедури виділення тренду. Так, для моделей сплайн-функцій з одним вузлом належать:

– лінія-лінія $\tilde{x}(t) = \begin{cases} a_1 + b_1 t, \quad t \in [a, t_1], \\ \tilde{x}(t_1) + b_2 (t - t_1), \quad t \in [t_1, b]; \end{cases}$

– лінія-парабола $\tilde{x}(t) = \begin{cases} a_1 + b_1 t, \quad t \in [a, t_1], \\ \tilde{x}(t_1) + b_2 (t - t_1) + c_2 (t - t_1)^2, \quad t \in [t_1, b]; \end{cases}$

– парабола-лінія $\tilde{x}(t)=\begin{cases} a_1+b_1t+c_1t^2, & t\in[a,t_1], \\ \tilde{x}(t_1)+b_2(t-t_1), & t\in[t_1,b]; \end{cases}$

– парабола-парабола $\tilde{x}(t)=\begin{cases} a_1+b_1t+c_1t^2, & t\in[a,t_1], \\ \tilde{x}(t_1)+b_2(t-t_1)+c_2(t-t_1)^2, & t\in[t_1,b]; \end{cases}$

де $\tilde{x}(t)=\hat{a}_1+\hat{b}_1t_1+\hat{c}_1t_1^2;$

Аналогічно вводяться сплайн-функції з двома вузлами склеювання :

– лінія-лінія-лінія $\tilde{x}(t)=\begin{cases} a_1+b_1t_1, & t\in[a,t_1], \\ \tilde{x}(t_1)+b_2(t-t_1), & t\in[t_1,t_2], \\ \tilde{x}(t_2)+b_3(t-t_2), & t\in[t_2,b]; \end{cases}$

– де $\tilde{x}(t_1)=\hat{a}_1+\hat{b}_1t_1;\quad \tilde{x}(t_2)=\tilde{x}(t_1)+\hat{b}_2(t_2-t_1)$

– лінія-лінія-парабола $\tilde{x}(t)=\begin{cases} a_1+b_1t_1, & t\in[a,t_1], \\ \tilde{x}(t_1)+b_2(t-t_1), & t\in[t_1,t_2], \\ \tilde{x}(t_2)+b_3(t-t_2)+c_3(t-t_2)^2, & t\in[t_2,b]; \end{cases}$

– лінія-парабола-лінія $\tilde{x}(t)=\begin{cases} a_1+b_1t_1, & t\in[a,t_1], \\ \tilde{x}(t_1)+b_2(t-t_1)+c_2(t-t_1)^2, & t\in[t_1,t_2], \\ \tilde{x}(t_2)+b_3(t-t_2), & t\in[t_2,b]; \end{cases}$

де $\tilde{x}(t_2)=\tilde{x}(t_1)+\hat{b}_2(t_2-t_1)+\hat{c}_2(t_2-t_1)^2$, та інші.

Для тренду на одній точці , вираз для $S_{зj}^2$, відповідно, набуває вигляду

$$S_{з2}^2=\frac{1}{n-2}\Big(\sum_{i=1}^{s}(x(ih)-a_1-b_1(ih))^2+\sum_{i=s+1}^{n}(x(ih)-\tilde{x}(is)-b_2(ih-is)-c_2(ih-sh)^2)^2\Big);$$

$$S_{з3}^2=\frac{1}{n-5}\Big(\sum_{i=1}^{s}(x(ih)-a_1-b_1(ih)-c_1(ih)^2)^2+\sum_{i=s+1}^{n}(x(ih)-\tilde{x}(sh)-b_2(ih-sh))^2\Big);$$

$$S^2_{з4} = \frac{1}{n-6}\Big(\sum_{i=1}^{s}(x(ih) - a_1 - b_1(ih) - c_1(ih)^2)^2 + \sum_{i=s+1}^{n}(x(ih) - \tilde{x}(sh) - b_2(ih - sh) - c_2(ih - sh)^2)^2\Big).$$

Оцінки вектора параметрів $\vec{\theta}_v$ знаходять із умови:

$$\min_v \min_{\vec{\theta}_v} S^2_{зj} \quad або \quad \frac{\partial S^2_{зj}}{\partial \theta_l} = 0,\ j = \overline{2;4},\ l = 1,2,..., \tag{2.37}$$

Для тренду, що описується сплайн-функціями з двома вузлами, задача відтворення тренду розв'язується аналогічно.

Вузли сплайн-функції можуть бути задані значеннями s,k або знайдені шляхом реалізації ітераційної процедури на підставі перебирання точок склеювання або інших обчислювальних схем.

2.5.2. Структура програмного забезпечення

2.5.2.1. Загальна структура проекту

Схема роботи програмного середовища представлена на рис. 2.3.

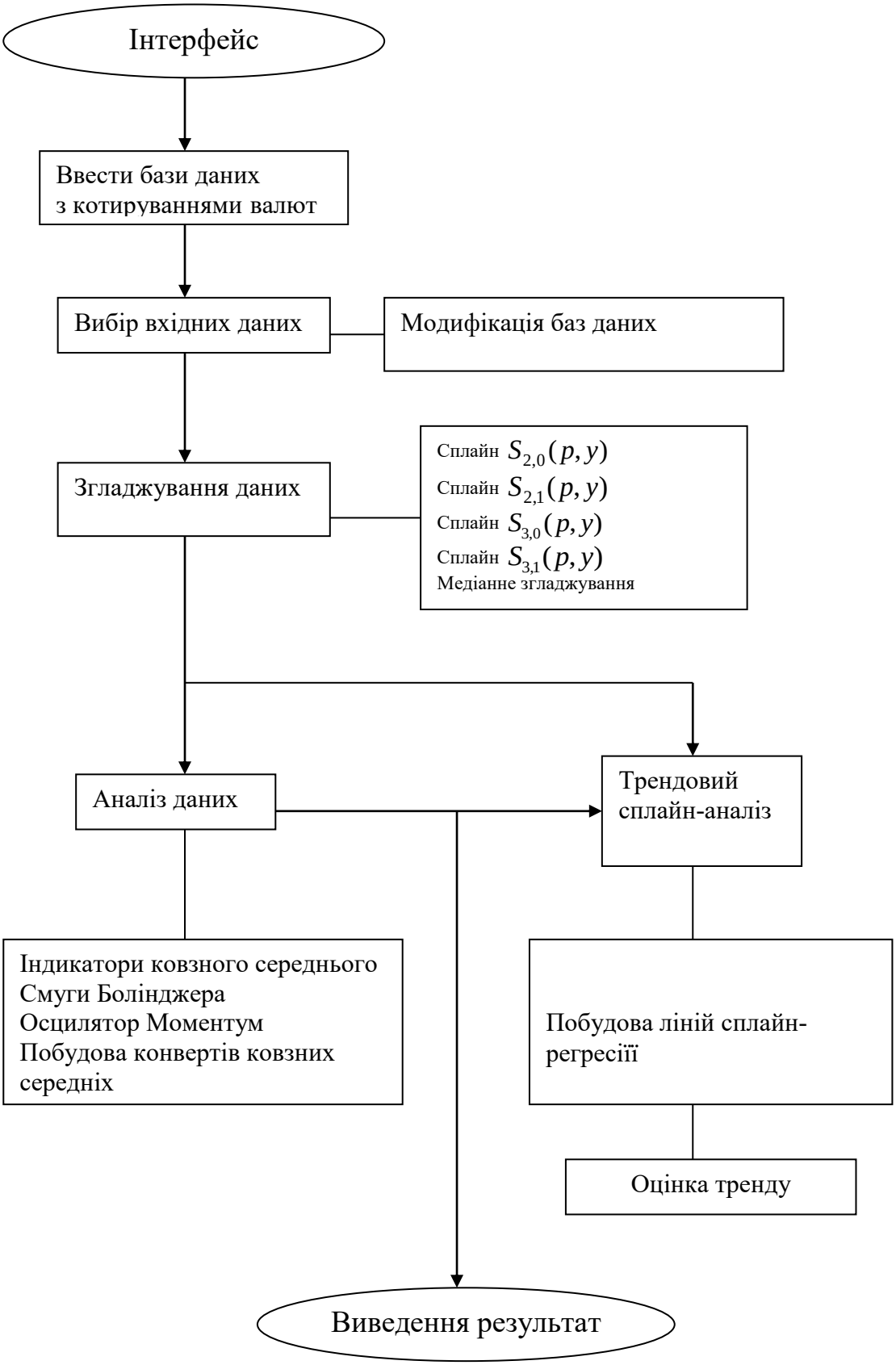

Рис. 2.3. Схема роботи програмного середовища

Програма розбита на два основних блока: блок введення та згладжування даних та блок аналізу.

Блок введення та згладжування даних дозволяє вводити дані з файлу для програми, відбувається згладжування даних, будуються графічні моделі даних. Дані можуть бути згладжені декількома способами. Результати згладжувань різними способами можна переглянути на графіках.

Блок аналізу реалізує обрані методи технічного аналізу: побудову індикаторів та осциляторів. Сюди ж відноситься й реалізація трендового сплайн-аналізу.

Система включає шість основних фізичних модулей:

– Unit1.cpp – головний модуль програми;

– Spline.h – модуль з розробками алгоритмів згладжування;

– Statan.h – модуль з основними алгоритмами статистичного аналізу даних;

– Regress.h – модуль з алгоритмами побудови ліній регресії;

– Indicator.h – модуль з розробками алгоритмів роботи індикаторів та осциляторів;

– Trend.h – алгоритми визначення тренду.

2.5.2.2. Опис функцій-членів програми

Spline.h – модуль з розробками алгоритмів згладжування містить наступні складові:

– Index * Do_SplineB2 (Index * ,float ,double *) – по заданих параметрах будує структуру – згладжені за допомогою B-сплайнів другого порядку дані. Повертає посилання на початок структури;

– Index * Do_SplineB21 (Index * ,float ,double *) – по заданих параметрах будує структуру – згладжені уточнені дані. Повертає посилання на початок структури;

– Index * Do_SplineB3 (Index * ,float ,double *) – по заданих параметрах будує структуру. Повертає посилання на початок структури;

– Index * Do_SplineB31 (Index * ,float ,double *) – по заданих параметрах будує структуру – згладжені уточненого дані. Повертає посилання на початок структури.

Statan.h – модуль з основними алгоритмами статистичного аналізу даних:

– double GetVarX(Index * T,int,int,int) – повертає величину $\bar{x}$ для ряду $T = \{(x, y)_i : i = \overline{1, n}\}$;

– double GetVarY(Index * T,int, int) – повертає величину $\bar{y}$ для ряду $T = \{(x, y)_i : i = \overline{1, n}\}$;

– double GetVarXY(Index * T,int,int) – повертає величину $\overline{xy}$ для ряду $T = \{(x, y)_i : i = \overline{1, n}\}$;

– double GetS2x(Index * T,int,int) – повертає величину S_x^2 для ряду $T = \{(x, y)_i : i = \overline{1, n}\}$;

– double GetS2y(Index * T,int,int) – повертає величину S_y^2 для ряду $T = \{(x, y)_i : i = \overline{1, n}\}$;

– double GetS2Vx(Index * T,int,int) – повертає величину $S_{\bar{x}}^2$ для ряду $T = \{(x, y)_i : i = \overline{1, n}\}$;

– double GetS2Vy(Index * T,int,int) – повертає величину $S_{\bar{y}}^2$ для ряду $T = \{(x, y)_i : i = \overline{1, n}\}$

– double GetKorel(Index * T, int,int) – повертає коефіцієнт кореляції;

– double KorelOtn (Index * T) – повертає кореляційне відношення;

– double GetLineb (Index * T, int,int) – повертає параметр b лінії лінійної регресії;

– double GetLinea (Index * T,int,int) – повертає параметр a лінії лінійної регресії;

– double cov (Index * T,int,int) – повертає коваріацію;

– void GetNonLineParameters(Index * T,double * a,double * b,double * c,int,int) – формує параметри для лінії нелінійної регресії;

Regress.h – модуль з алгоритмами побудови ліній регресії, містить наступні функції:

– double GetS2LineLine(Index * T,int k) – повертає середньоквадратичне відхилення побудованої лінії регрессії з вузлом в точці k типу пряма-пряма від початкових даних;

– void DoLineLine(Index * T,int * x,double * Min) – будує лінію сплайн-регрессії з вузлом в точці k типу пряма-пряма;

– void DrawLineLine(Index * T,TLineSeries * A,int k,TColor Col) – зображує на графіку побудовану лінію сплайн-регрессії з вузлом в точці k типу пряма-пряма.

Indicator.h – модуль з розробками алгоритмів роботи індикаторів та осциляторів:

– void SimpleMovingAverage (Index * T,int k, TLineSeries * L, TColor C,int bol,int s = 0) – реалізація індикатора звичайного ковзного середнього. При певному наборі вхфдних параметрів працює і як індикатор – смуги Болліджера;

– void WeightedMovingAverage (Index * T,int k, TLineSeries * L, TColor C,int bol,int s=0) – реалізація індикатора зваженого ковзного середнього. При певному наборі вхідних параметрів працює і як індикатор – смуги Болліджера;

– void ExponentialMovingAverage (Index * T,int k, TLineSeries * L, TColor C,int bol,int s=0) – реалізація індикатора експонентного ковзного середнього. При певному наборі вхфдних параметрів працює і як індикатор – смуги Болліджера;

– MovingAverageEnvelopes(Index * T,int k,int g,TLineSeries * LU,TLineSeries * LD,TColor C,int method) – реалізація конвертів ковзних середніх;

– void Momentum (Index * T,int k,TLineSeries * L,TColor C) – реалізація осцилятора Моментум.

– void Stohastic (Index * T,int k,int k2, TLineSeries * L1,TLineSeries * L2, TColor C) – реалізація стохастичного осцилятора.

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Вхідними даними можуть бути дані, які зчитуються з файлу . txt в такому форматі :

Таблиця 2.1

Приклад вхідних даних програми

Дата	Вага	Ціна
06.01.2021	10	10449.74
08.01.2021	10	10662.60
09.01.2021	10	10609.09
10.01.2021	10	10609.09

Вихідними даними роботи програми є :

– медіанне згладжування та згладжування даних за допомогою локальних поліноміальних B-сплайнів другого та третього порядків та їх уточнених модифікацій;

– побудова ліній лінійної та нелінійної регресії;

– трендовий сплайн-аналіз з одним вузлом та з двома вузлами.

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Для розробки даного ПЗ було використано ПК з наступними характеристиками:

- тип процесора: процесор з частотою 2.2 ГГц;
- ОЗУ об'ємом 4 Гб;
- 300 Мб доступного простору на жорсткому диску;
- жорсткий диск з частотою обертання 5400 об / хв.;
- дозвіл екрану 1024x768;
- клавіатура;
- маніпулятор «миша».

2.7.2. Використані програмні засоби

Дане програмне забезпечення розроблене засобами середовища C++ в середовищі візуального програмування Visual Studio 2017

2.7.3. Виклик та завантаження програми

Програма не потребує додаткового налаштування. Для запуску програмного додатку потрібно відкрити подвійним кліком файл - «Analysis.exe».

2.7.4. Опис інтерфейсу користувача

Після запуску програми, користувач повинен відкрити файл з вхідними даними. Це можна зробити двома шляхами: скористуватися кнопкою «Файл – Відкрити» на панелі меню, або обрати закладку «Завантаження й різка даних» (рис. 2.4):

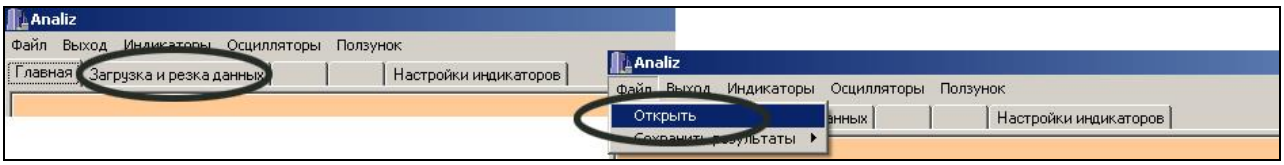

Рис. 2.4. Завантаження даних

Якщо користувач обирає пункт меню «Файл – Відкрити», він побачить діалогове вікно, де система запропонує йому обрати необхідний файл.

Якщо ж, файлів багато і користувачеві необхідно зробити вибір, користуючись лише графічною інформацією, він може перейти на закладку «Завантаження й різка даних» (рис. 2.5):

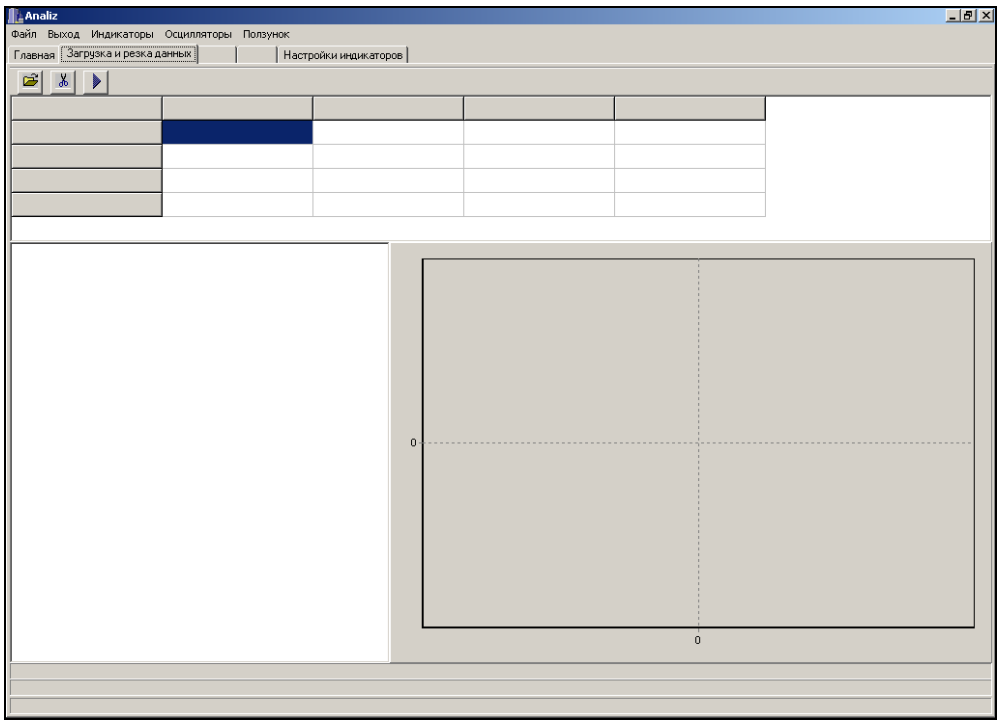

Рис. 2.5. Закладка «Завантаження і редагування даних»

На цій закладці розміщені кнопки:

1. «Завантаження даних».
2. «Вирізати період».
3. «Передати обраний файл в головну».

Натиснувши на кнопку «Завантаження даних», користувач має змогу в діалоговому вікні обрати декілька файлів та відкрити їх. Після завершення завантаження всіх обраних файлів, користувач зможе переглянути графічну інформацію кожного з обраних файлів просто і зручно, користуючись клавішами ⇑,⇓. У відповідному вікні можна побачити текстову інформацію відповідного файлу (рис. 2.6).

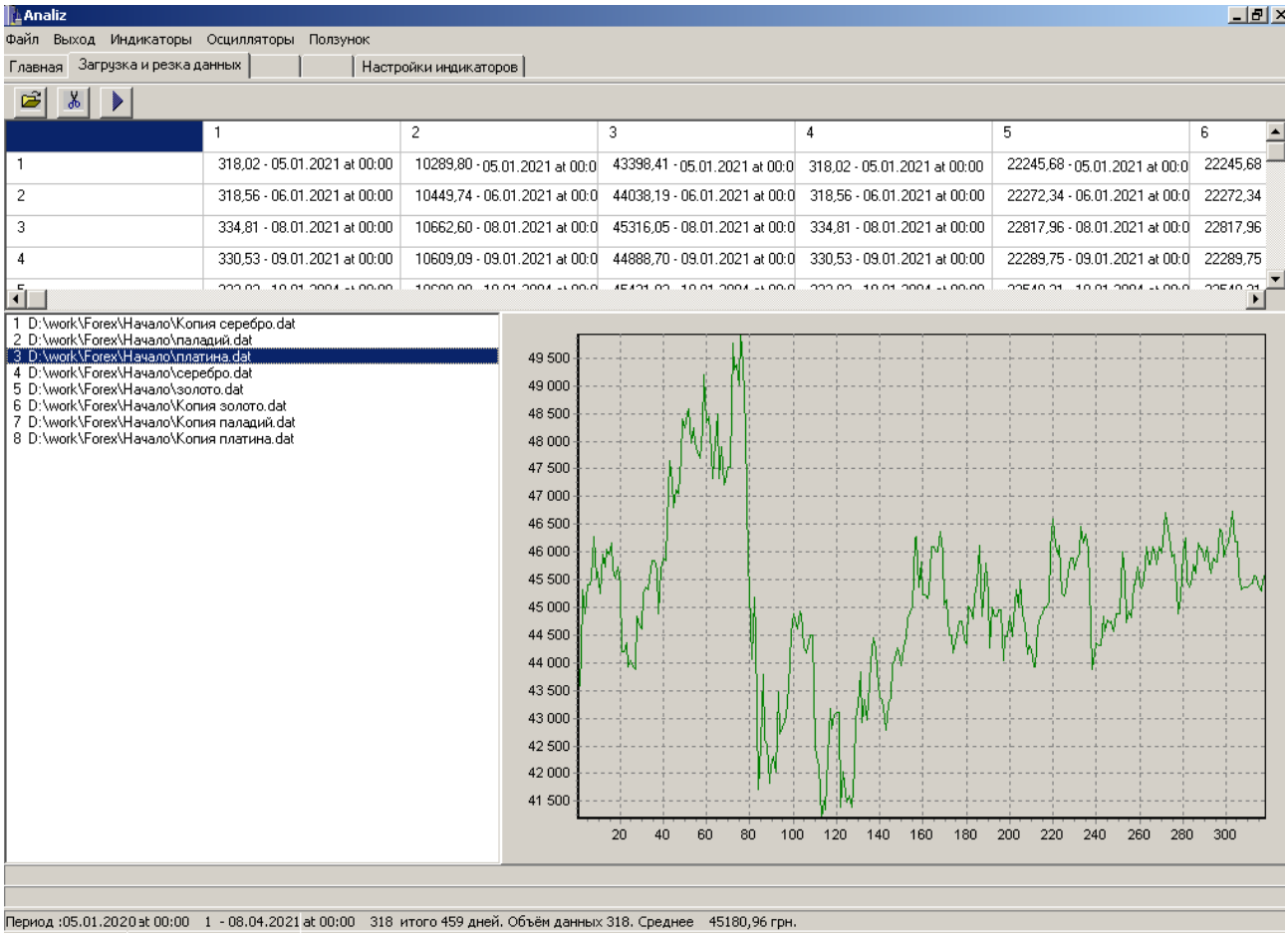

Рис. 2.6. Завантажені дані в графічному виді

Натиснувши на кнопку «Вирізати період», користувач має змогу редагувати файл даних - відокремити конкретний період часу, для майбутньої його обробки (рис. 2.7):

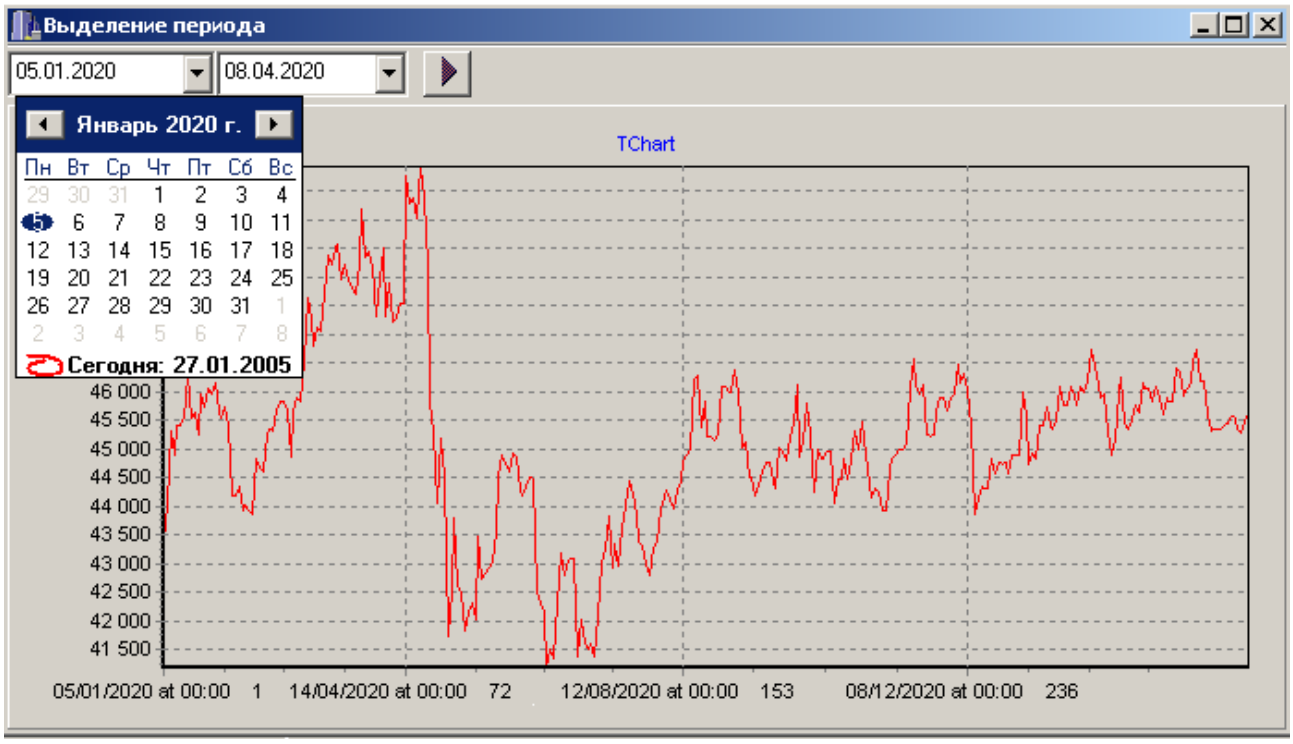

Рис. 2.7. Відокремлення періоду

За допомогою кнопки «Передати обраний файл в головну» можна обрані і відредаговані дані передати в головну закладку для аналізу даних.

Коли дані підготовлені до обробки, користувач переходить на закладку «Головна».

Цикл алгоритмів, що реалізують процеси згладжування даних та трендового сплайн-аналізу користувач має змогу застосувати наступним чином: знаходячись на закладці «Головна», користувач за допомогою контекстного меню може викликати наступні процедури:

– згладжування будь-яким з вищезазначених методів, а саме: медіанне згладжування, згладжування даних за допомогою локальних поліноміальних B-сплайнів другого та третього порядків та їх уточнених модифікацій;

– побудова ліній лінійної та нелінійної регресії;

– активування/деактивування лінії руху ціни на графіку;

– відображення панелі інструментів;

– трендовий сплайн-аналіз з одним вузлом;

– трендовий сплайн-аналіз з двома вузлами.

Панель інструментів має майже таку ж структуру та наповнення, як і контекстне меню, крім того, кожна з кнопок панелі супроводжується help-повідомленням, яке можне прочитати, підвівши мишу до кнопки (рис. 2.8).

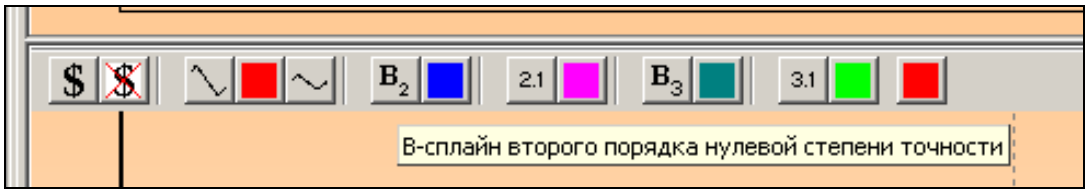

Рис. 2.8. Панель інструментів

Процедури трендового сплайн-аналізу мають свої опції (рис. 2.9):

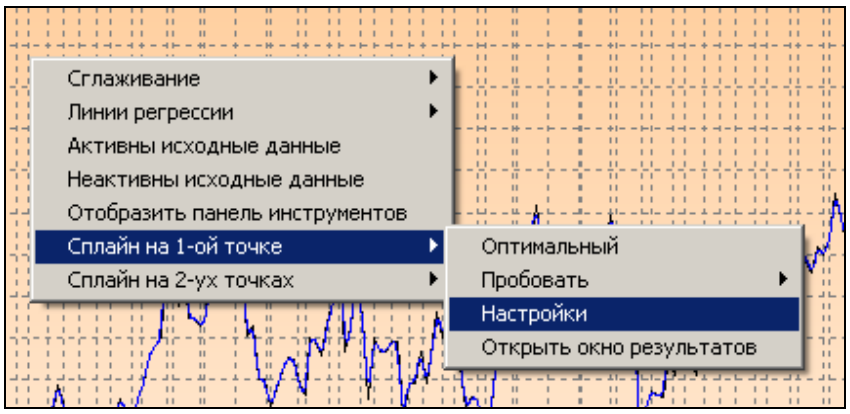

Рис. 2.9. Процедури трендового сплайн-аналізу

Оскільки процедура знаходження лінії тренду – це перебирання всіх можливих варіантів перестановок прямих та парабол з однією точкою склеювання, або з двома, то цей процес може, у разі використання даних великого обсягу, займати занадто багато часу. З огляду на це, з метою скорочення часу, користувач має змогу відключити деякі варіанти перевірки (рис. 2.10):

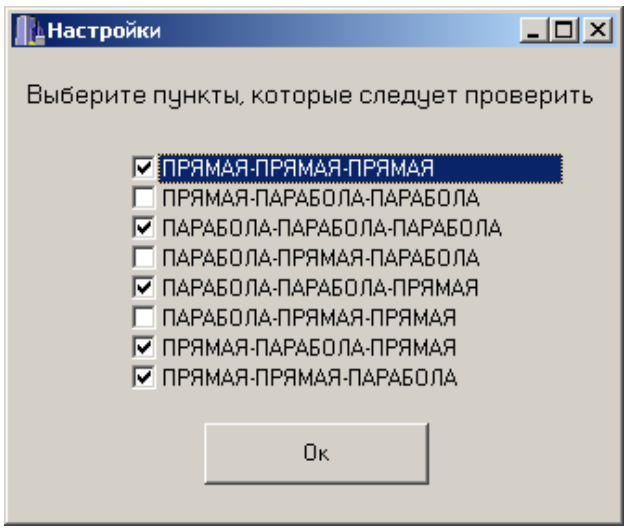

Рис. 2.10. Опції трендового сплайн-аналізу

Після того, як процедура трендового сплайн-аналізу отримає результати, вони будуть збережені у вікні, яке можна відкрити наступним чином (рис. 2.11):

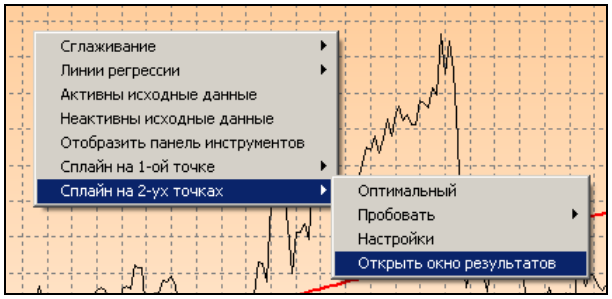

Рис. 2.11. Процедури трендового сплайн-аналізу

Вікно результатів виглядає наступним чином (рис. 2.12):

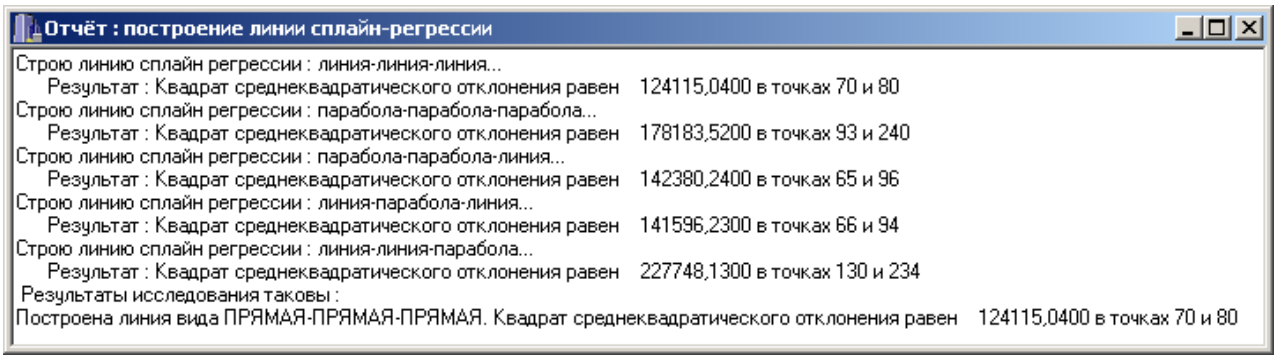

Рис. 2.12. Вікно результатів

Щоб система знайшла та побудувала оптимальний варіант лінії тренду, треба обрати пункт «Оптимальний», для того ж, щоб проглянути лінії якогось з можливих варіантів, треба скористатися пунктом «Пробувати».

Застосування функцій технічного аналізу, здійснюється наступним чином: в головному меню користувач обирає пункти «Індикатори» та «Осцилятори». Вони мають такі структури:

1. Звичайне ковзне середнє:
   - побудова короткого;
   - знищення короткого;
   - побудова довгого;
   - знищення довгого;
   - побудова конверту;
   - знищення конверту.
2. Зважене ковзне середнє:
   - побудова короткого;
   - знищення короткого;
   - побудова довгого;
   - знищення довгого;
   - побудова конверту;
   - знищення конверту.
3. Експонентне ковзне середнє:
   - побудова короткого;
   - знищення короткого;
   - побудова довгого;
   - знищення довгого;
   - побудова конверту;,
   - знищення конверту.
4. Лінії Боліндже́ра:
   - побудова на основі звичайного ковзного середнього;
   - побудова на основі зваженого ковзного середнього;

– знищення лінії Боліндженра.

Пункт «Осцилятори» має таку структуру:

1. Momentum.
2. RSI.
3. Знищення лінії осцилятора.

Закладка «Опції індикаторів» містить всі необхідні опції індикаторів та осциляторів.

Функція «Повзунок» (розміщується на головному меню) використовується, в ситуації, коли одночасно користуються даними з вікна осциляторів та головного вікна, причому в якомусь з вікон використані можливості зуму, тобто порушена візуальна відповідність даних в одному вікні – іншому. В таких випадках користуються функцією, яка встановлює цю відповідність (рис. 2.13):

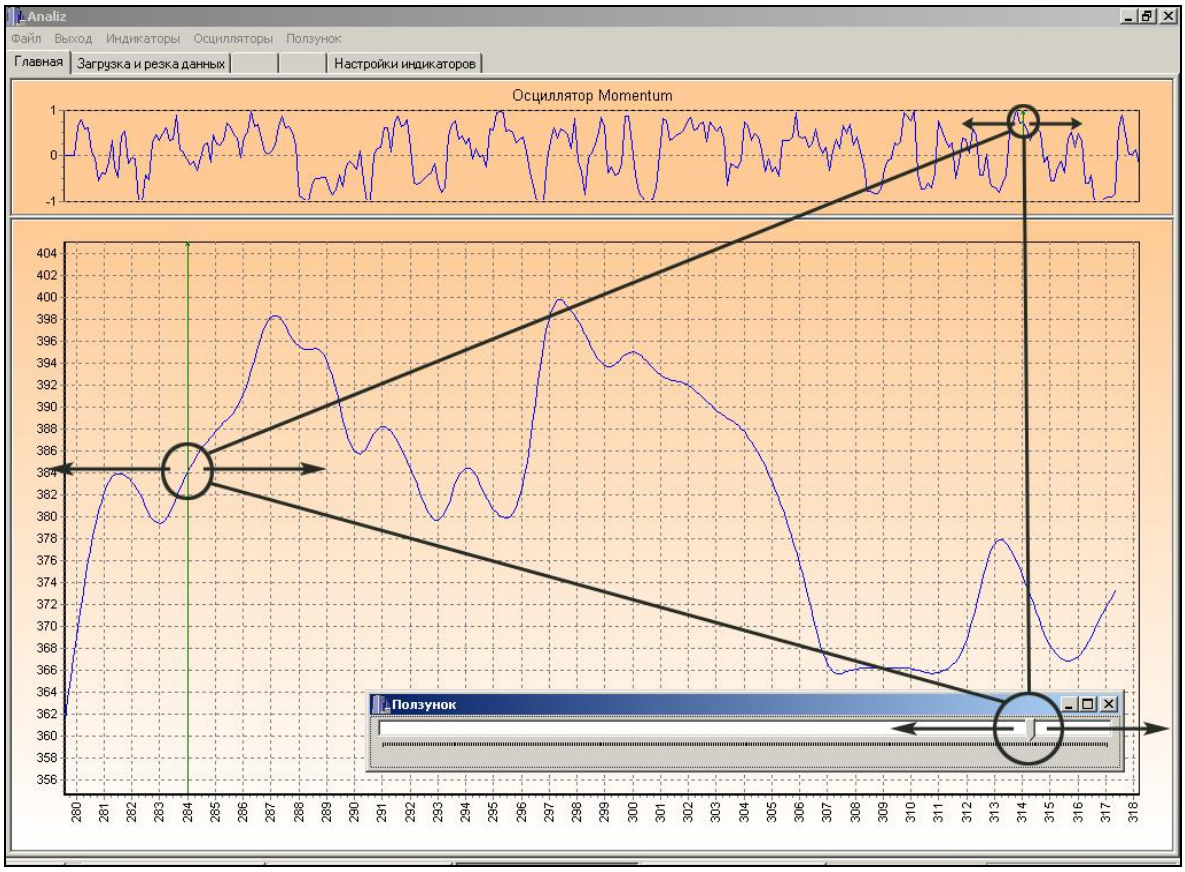

Рис. 2.13. Функція «Повзунок»

У будь-який момент можна вийти з програми за допомогою пункта меню «Вихід» або натискаючи на хрестик у правому верхньому куті додатку.

2.7.5. Тестування програмного засобу

Для тестування роботи, було зібрано цикл вхідних даних, а саме:

- котирування австралійського долара (01.01.2007 – 08.04.2021);
- котирування євро (01.01.2007 – 11.04.2021);
- котирування української гривні (01.01.2000 – 08.04.2021);
- котирування японської йени (01.01.2005 – 08.04.2021);
- котирування американського долара (01.01.2000 – 08.04.2021);
- котирування датської крони (01.01.2005 – 08.04.2021);
- котирування білоруського рубля (01.01.2005 – 08.04.2021);
- котирування бельгійського франку (01.01.2005 – 22.02.2021);
- котирування англійського фунту стерлінгів (01.01.2007 – 08.04.2021);
- котирування австрійського шилінгу (01.01.2007 – 22.02.2021);
- котирування цін на паладій (05.01.2020 – 08.04.2021);
- котирування цін на платину (05.01.2020 – 08.04.2021);
- котирування цін на золото (05.01.2020 – 08.04.2021);
- котирування цін на срібло (05.01.2020 – 08.04.2021).

Тестування роботи програми проводено на даних – котируваннях цін срібла за період 05.01.2020 – 08.04.2021:

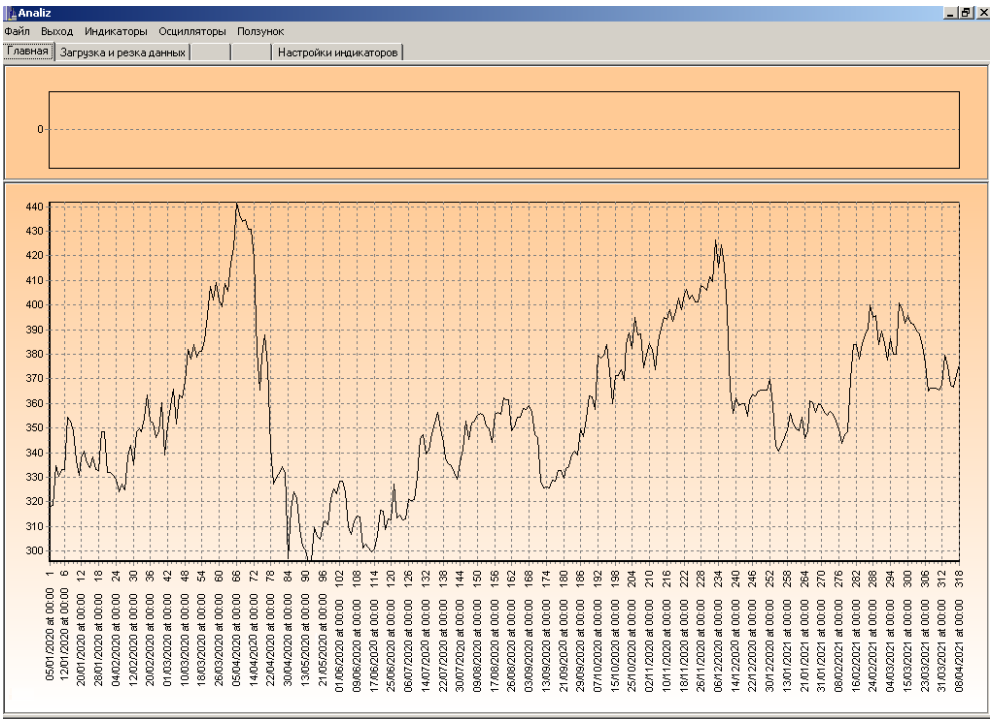

Рис. 2.14. Завантаження даних

Проведемо згладжування даних:

– медіанне згладжування (рис. 2.15):

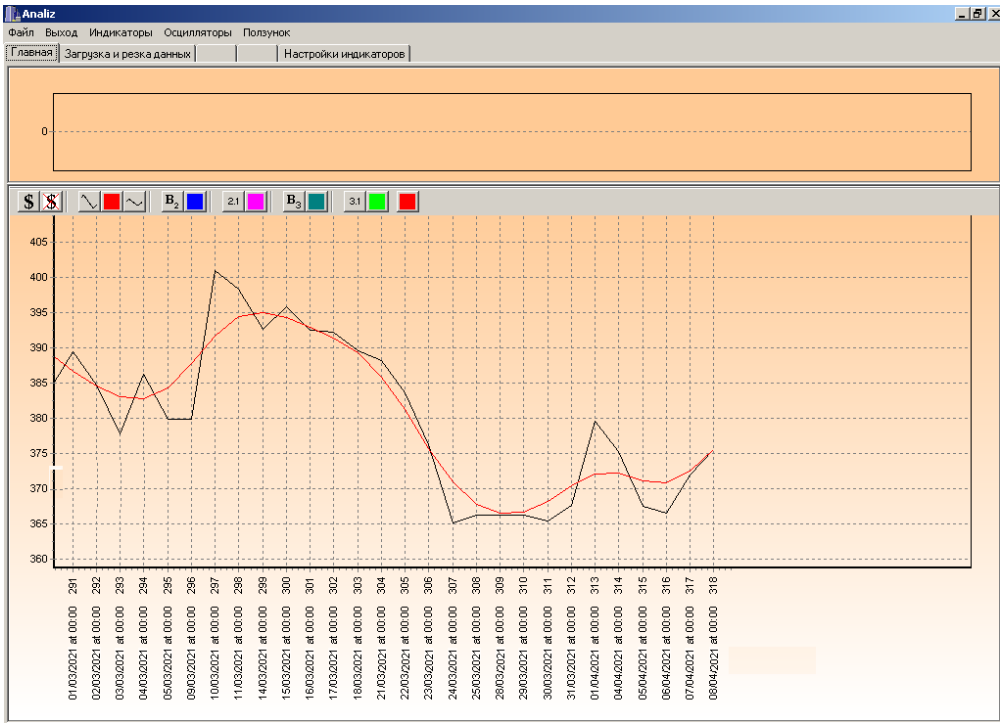

Рис. 2.15. Завантажені дані згладжені медіанним згладжуванням

– за допомогою B-сплайнів другого порядку (рис. 2.16):

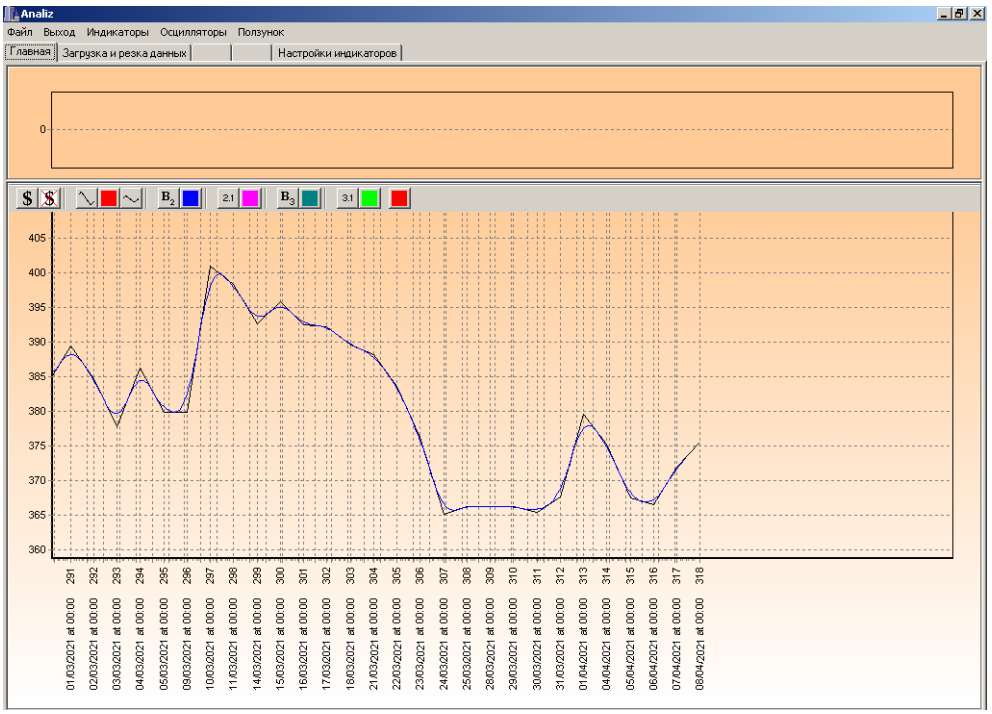

Рис. 2.16. Завантажені дані згладжені за допомогою B-сплайнів другого порядку

– за допомогою B-сплайнів другого порядку уточнених (рис. 2.17):

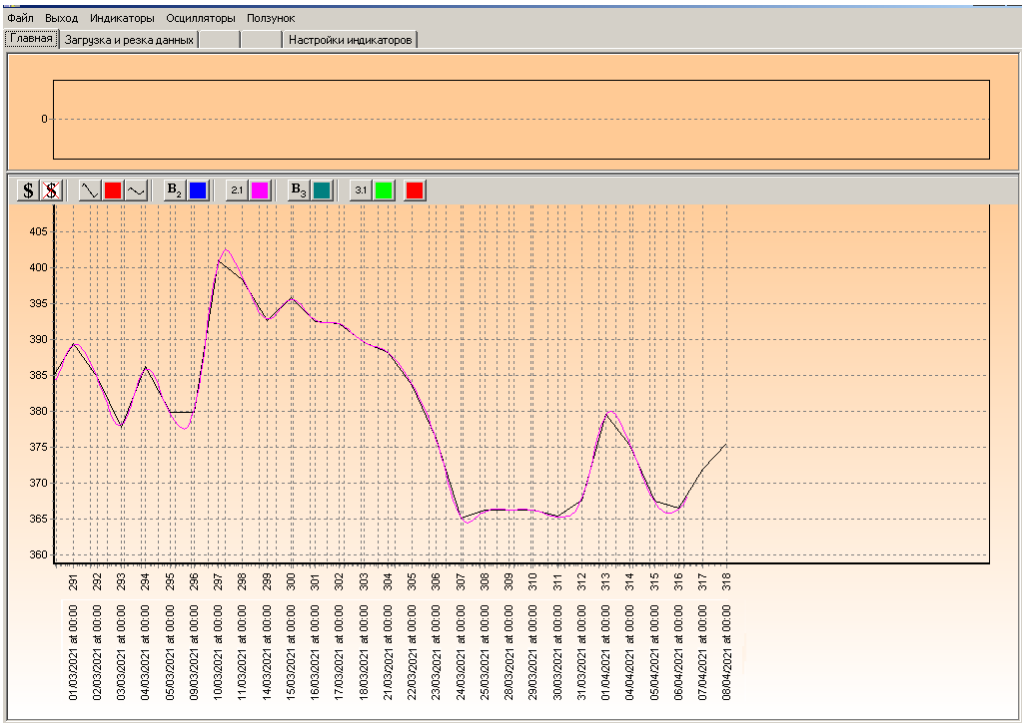

Рис. 2.17. Завантажені дані згладжені за допомогою B-сплайнів другого порядку уточнених

– за допомогою B-сплайнів третього порядку (рис. 2.18):

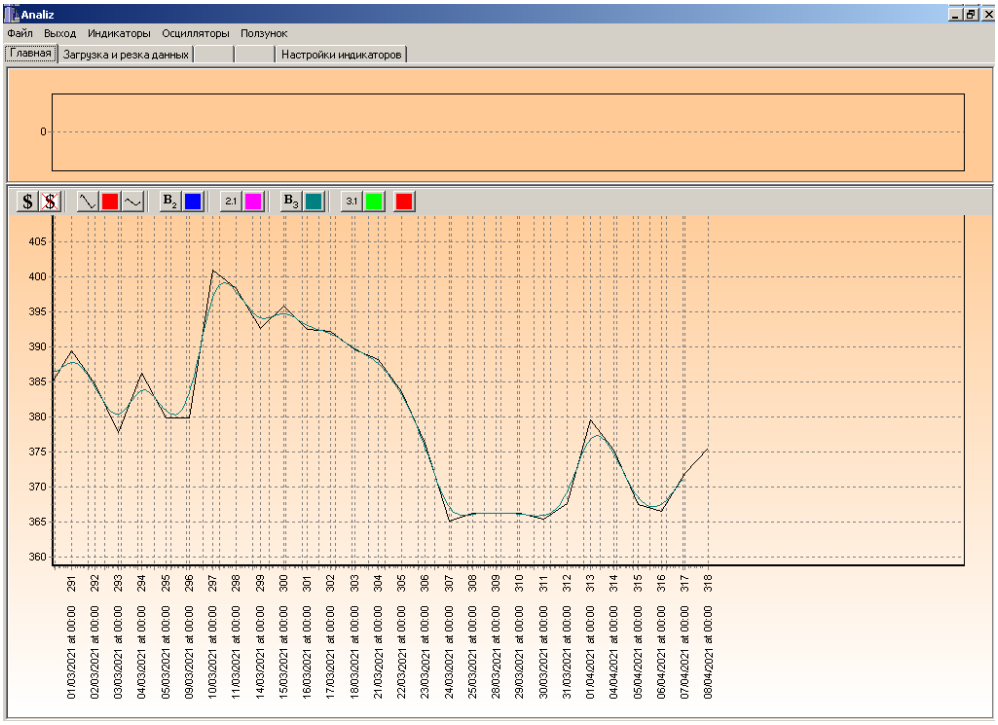

Рис. 2.18. Завантажені дані згладжені за допомогою B-сплайнів третього порядку

– за допомогою B-сплайнів третього порядку уточнених (рис. 2.19):

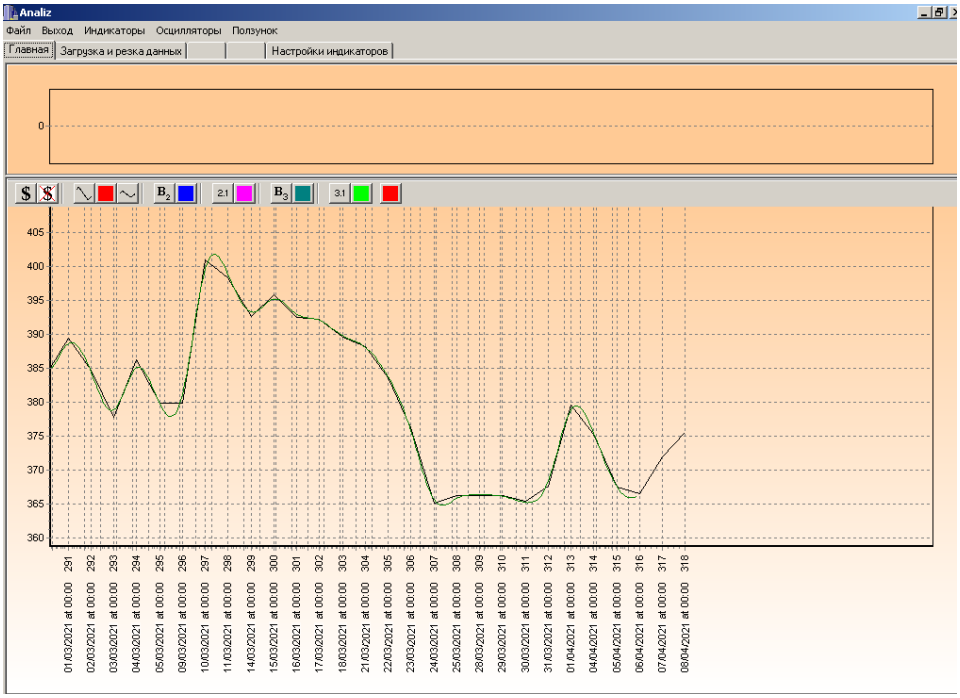

Рис. 2.19. Завантажені дані згладжені за допомогою B-сплайнів третього порядку уточнених

Як видно з графіків, кожен з алгоритмів згладжування має свої особливості щодо отриманих результатів. В залежності від даних та їх зашумленості, користувач сам повинен зробити висновок й обрати той тип згладжування, який максимально відповідає вимогам поставленої задачі. В даному випадку найбільш адекватно згладжені дані за допомогою B-сплайну другого порядку (рис 2.16). Отже, отримано результат (рис. 2.20):

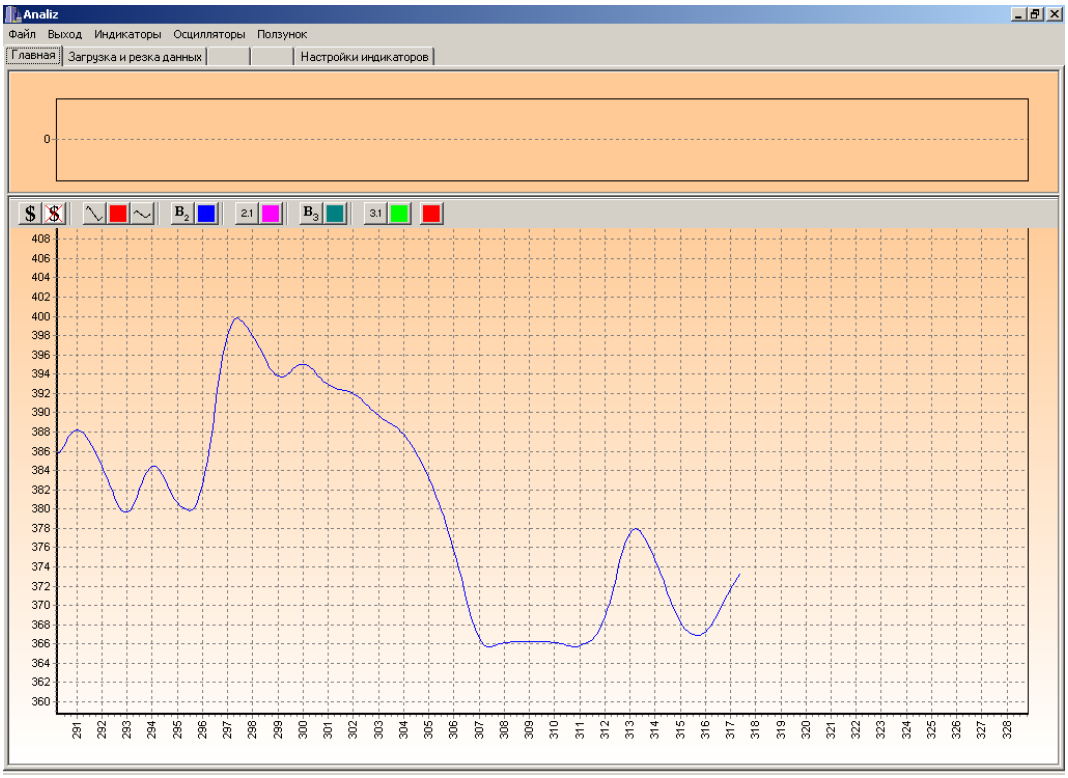

Рис. 2.20. Згладжені дані

Тепер використаємо індикатори для аналізу ринка. Почнемо зі звичайного ковзного середнього з періодом 5 (рис. 2.21):

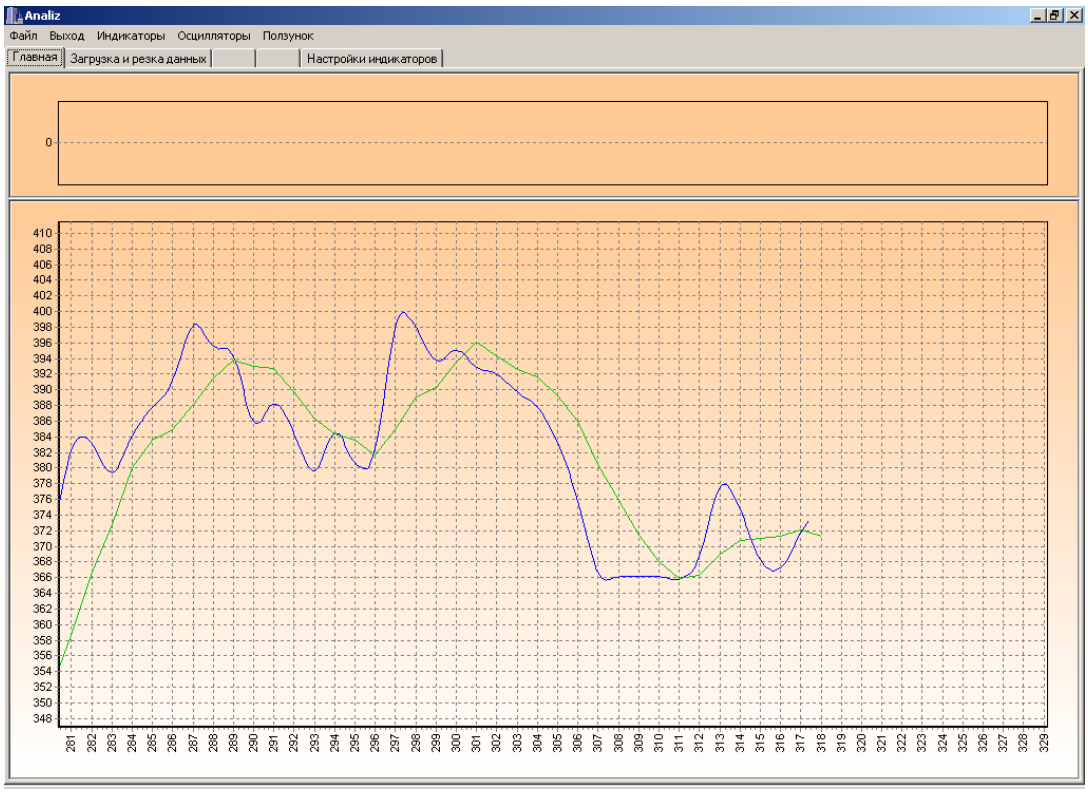

Рис. 2.21. Звичайне ковзне середнє. Період дорівнює 5

Як бачимо, цей індикатор з запізненням, але вказує трейдерові деякі моменти, а саме:

– 25.02.2021 – індикатор змінює напрямок руху (має максимум) в умовах спаданні ціни – сигнал продавати;

– 09.03.2021 – індикатор змінює напрямок руху (має мінімум) в умовах зростання ціни – сигнал купувати;

– 16.03.2021 – індикатор змінює напрямок руху (має максимум) в умовах спаданні ціни – сигнал продавати;

– 30.03.2021 – індикатор змінює напрямок руху (має мінімум) в умовах зростання ціни – сигнал купувати.

Побудуємо 3% - конверт звичайного ковзного середнього. Основна тактика: продавати, коли ціна біля верхньої смуги і навпаки (рис. 2.22):

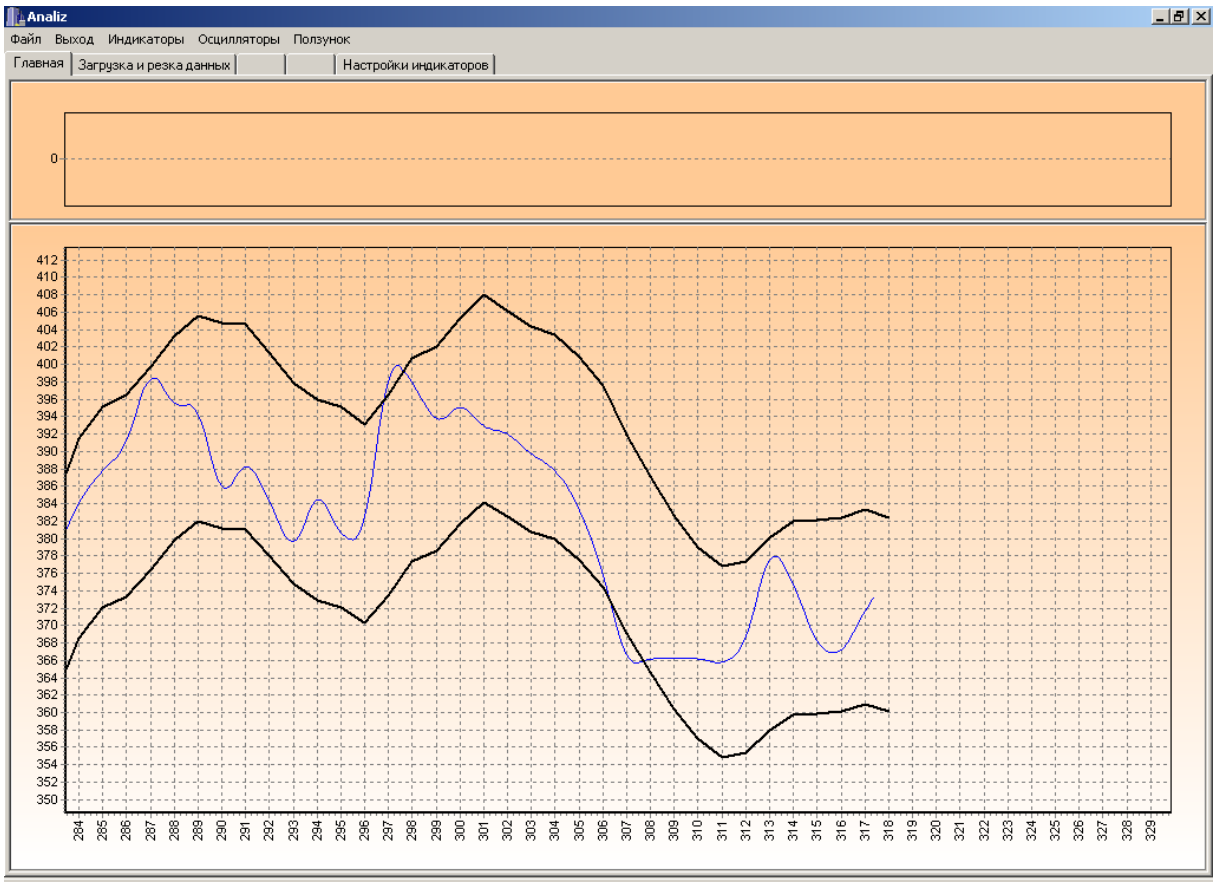

Рис. 2.22. Конверт звичайного ковзного середнього

Отже, з отриманого випливає :

– з 23.02.2021 – продавати;
– з 03.03.2021 – покупати;
– 10.03.2021 – 11.03.2021 – продавати;
– з 25.03.2021 – покупати;
– 01.04.2021 – продавати.

Побудуємо смуги Боліндже́ра на основі звичайного ковзного середнього (рис. 2.23):

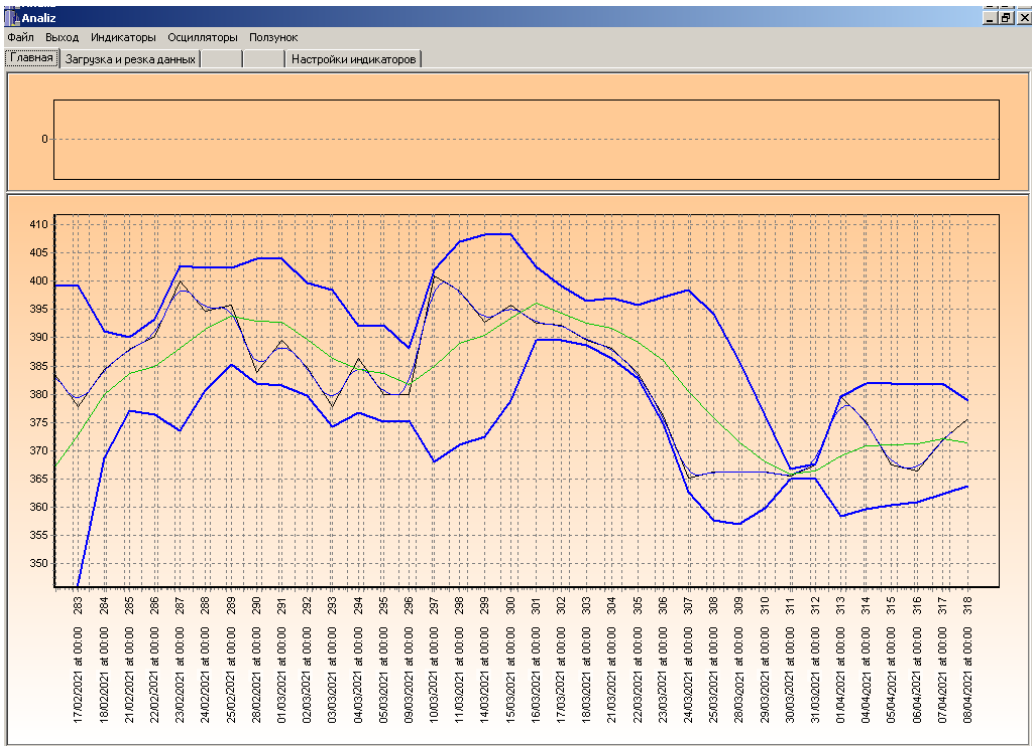

Рис. 2.23. Смуги Боліндже́ра

Тут також можна користуватися торговою стратегією: при наближенні ціни до верхньої смуги – продавати, при наближенні до нижньої – купувати. Відповідно до цього можна виділити періоди закупівель і продаж.

Скористаємось осцилятором моментів (рис. 2.24):

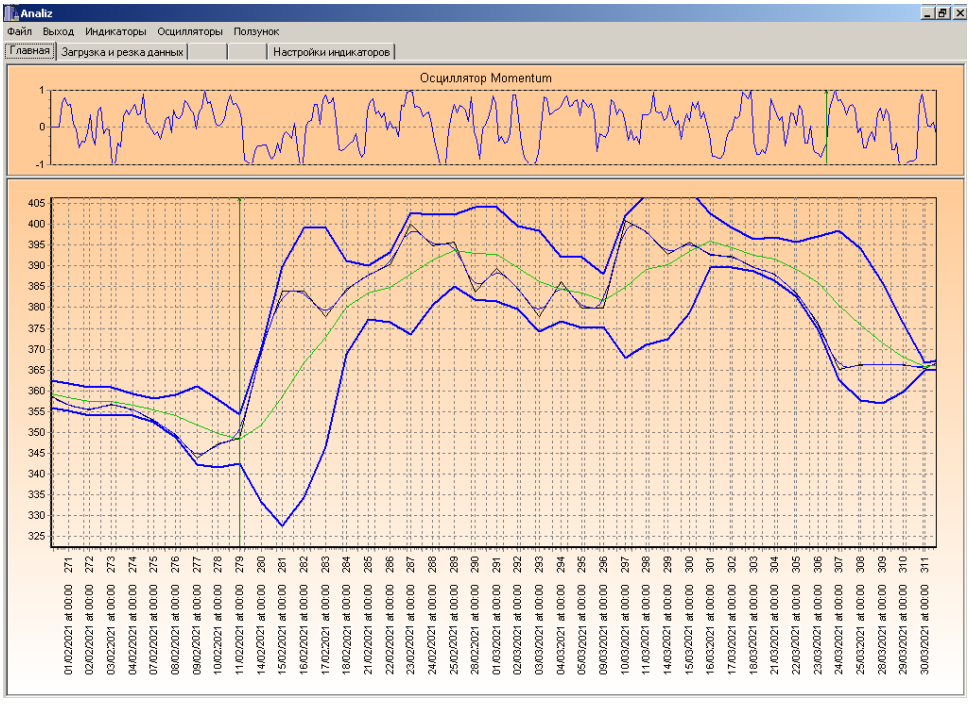

Рис. 2.24. Осцилятор Momentum

Отримані результати показують, що, наприклад, 11.02.2021 необхідно було купувати, адже Моментум перетнув свою нульову лінію на підйомі; а 25.02.2021 вже настав час продаж – осцилятор набув свого максимуму, розвернувся і почав падати.

Проведемо тепер трендовий сплайн-аналіз:

– з одним вузлом (рис. 2.25):

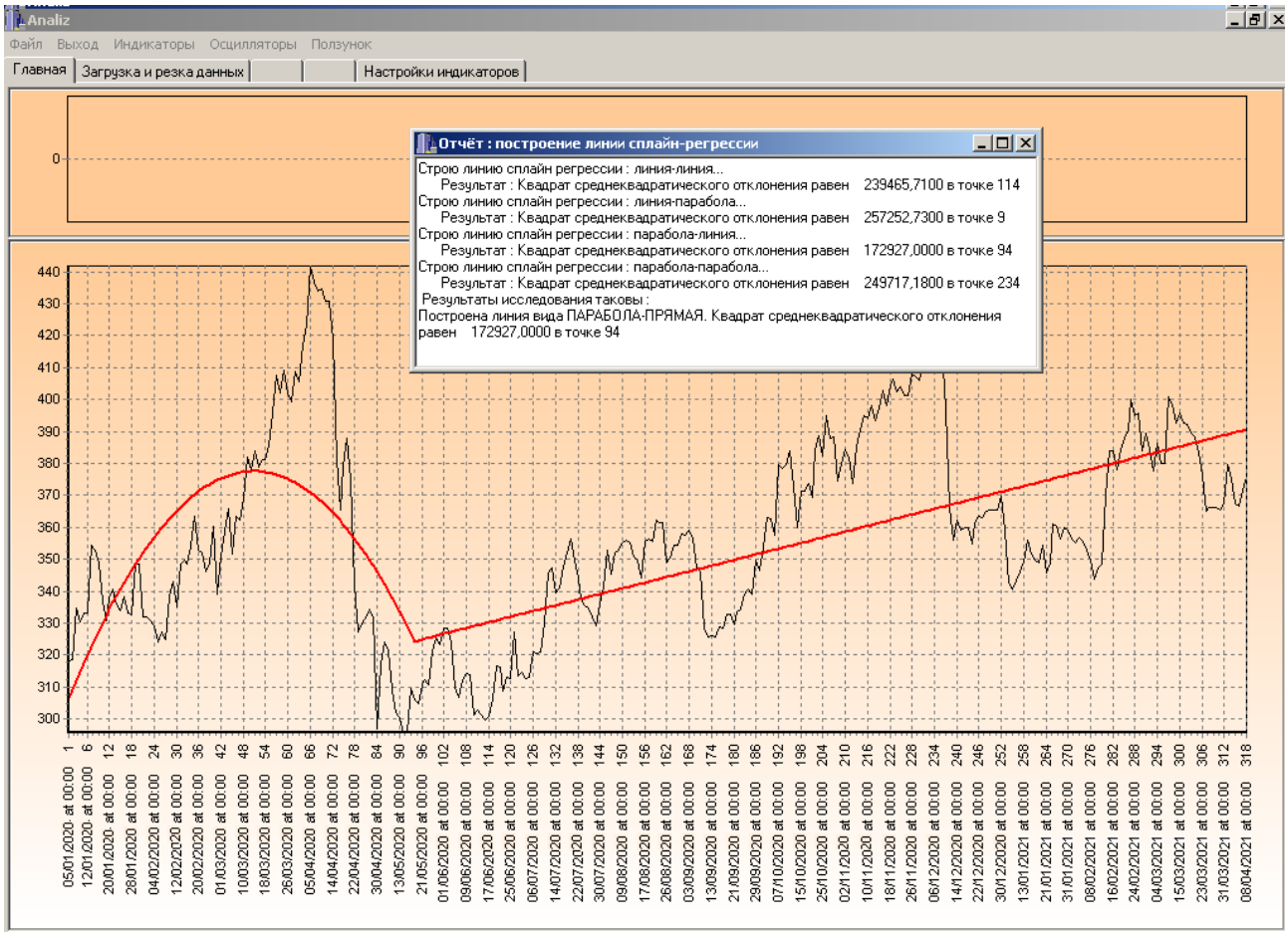

Рис. 2.25. Трендовий сплайн-аналіз з одним вузлом

Система визначила: 05.01.2020 – 19.05.2020 – параболічний тренд: зростання в період з 05.01.2020 до 15.03.2020; падіння в період з 16.03.2020 до 19.05.2020. Далі відбувається лінійне зростання ціни до кінця розглядуваного періоду часу. Як бачимо, сума квадратів відхилень від реальної ціни, тобто адекватність побудованої лінії тренду дорівнює 172927,00 од.

– з двома вузлами (рис. 2.26):

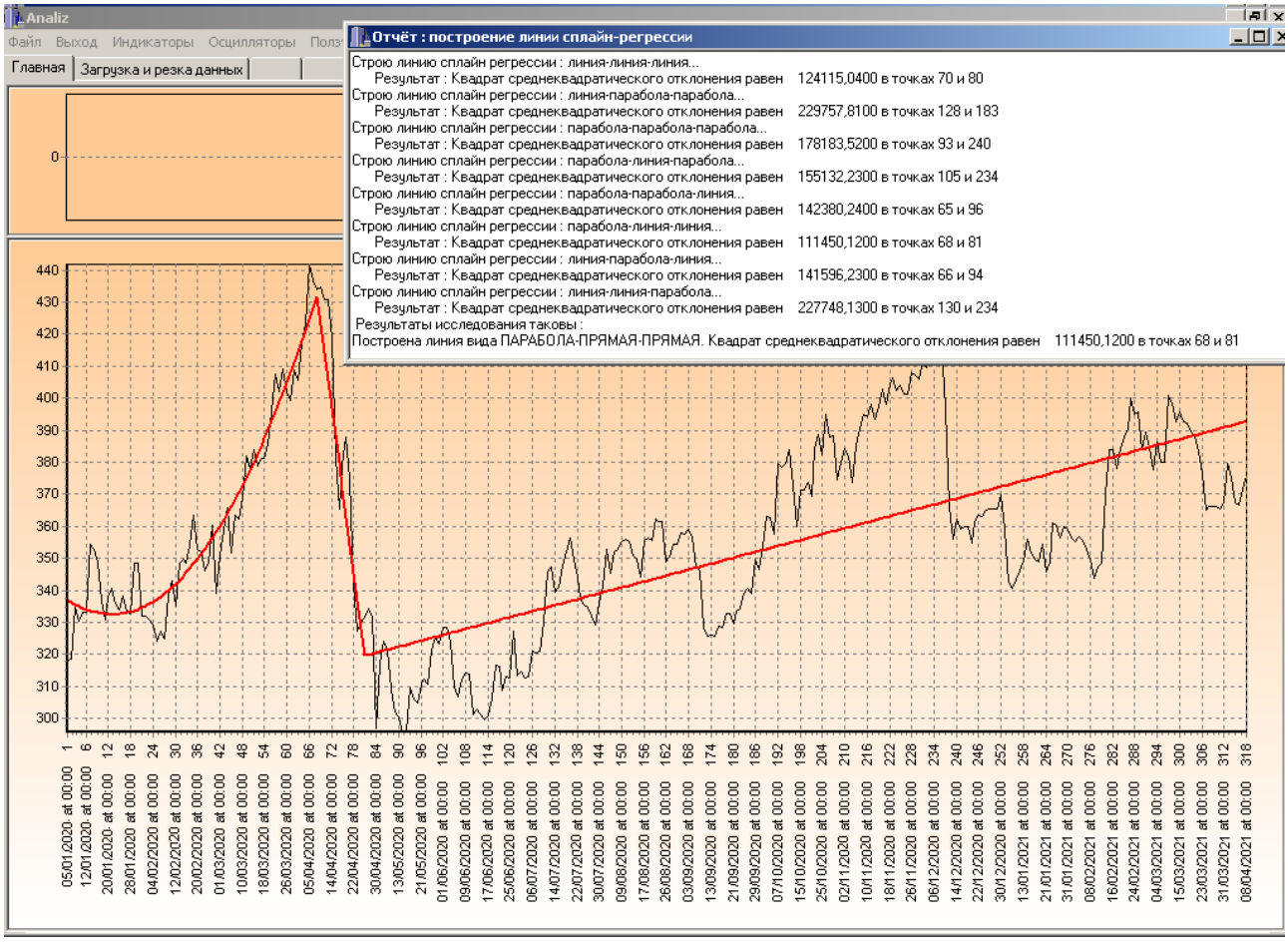

Рис. 226. Трендовий сплайн-аналіз з двома вузлами

Система визначила: 05.01.2020 – 07.04.2020 – параболічний тренд: падіння в період з 05.01.2020 до 21.01.2020; зростання в період з 20.01.2020 до 07.04.2020. Далі відбувається лінійне падіння ціни до 27.04.2020. В період з 28.04.2020 до 08.04.2021 – лінійний зростаючий тренд. Сума квадратів відхилень від реальної ціни, тобто адекватність побудованої лінії тренду дорівнює 111450,12 од.

Отже побудована модель на 2-ох вузлах склеювання більш адекватно зображує лінію тренда. Про це свідчить як візуальний аналіз, так і сума квадратів відхилень.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів – 1500;
- коефіцієнт складності програми – 2;
- коефіцієнт корекції програми в ході її розробки – 0,08;
- годинна заробітна плата програміста, грн / год – 100.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де to - витрати праці на підготовку й опис поставленої задачі (приймається 50),

ти - витрати праці на дослідження алгоритму рішення задачі,

та - витрати праці на розробку блок-схеми алгоритму,

тп - витрати праці на програмування по готовій блок-схемі,

тотл - витрати праці на налагодження програми на ЕОМ,

тд - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \times C \times (1+ p)\text{ , людино-годин,} \qquad (3.2)$$

де q - передбачуване число операторів,

С - коефіцієнт складності програми,

р - коефіцієнт кореляції програми в ході її розробки.

$$Q = 1500 \cdot 2 \cdot (1 + 0{,}08) = 3240 \text{ людино-годин.}$$

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{QB}{(75...85)K}\text{ , людино-годин,} \qquad (3.3)$$

де В - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; В=1.2 ... 1.5,

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{3240 \cdot 1{,}3}{80 \cdot 1{,}2} = 44\text{ , людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_a = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \qquad (3.4)$$

$$t_a = \frac{3240}{22 \cdot 1{,}2} = 123 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \tag{3.5}$$

$$t_n = \frac{3240}{23 \cdot 1{,}2} = 117 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4...5)K} \quad \text{людино-годин.} \tag{3.6}$$

$$t_{отл} = \frac{3240}{5 \cdot 1{,}2} = 540 \text{ людино-годин.}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \tag{3.7}$$

де tдр - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)K}, \text{ людино-годин.} \tag{3.8}$$

$$t_{др} = \frac{3240}{18 \cdot 1{,}2} = 150 \text{ людино-годин,}$$

tдо - трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.} \tag{3.9}$$

$$t_{до} = 150 + 73 = 223 \text{ людино-годин.}$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 44 + 123 + 117 + 540 + 223 = 1097$$ людино-годин.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.10)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн,} \quad (3.11)$$

де t - загальна трудомісткість, людино-годин,

Спр - середня годинна заробітна плата програміста, грн/година.

$$Ç_{ÇÏ} = 1097 \cdot 100 = 109700 \text{ ãðí.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times C_{М}, \text{ грн,} \quad (3.12)$$

де tотл - трудомісткість налагодження програми на ЕОМ, год,

Смч - вартість машино-години ЕОМ, грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$З_{МВ} = 540 \times 5 = 2700 \text{ грн.}$$

$$Êîï = 2700 + 109700 = 112400 \text{ ãðí.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.} \qquad (3.13)$$

де Bk - число виконавців,

Fp - місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = \frac{1097}{1 \cdot 176} = 6.23 \text{ міс.}$$

Висновки: визначено трудомісткість розробленого додатку (1097 люд-год), проведений підрахунок вартості роботи по створенню програми (112400 грн.) та розраховано час на його створення (6,2 міс).

ВИСНОВКИ

Метою кваліфікаційної роботи є розробка програмного забезпечення засобів згладжування, обробки та аналізу даних на ф‘ючерсному ринку.

У даній роботі представлено програмне забезпечення реалізації алгоритмів аналізу та прогнозування на ф‘ючерсних ринках, яке може бути використане в роботі з курсами котирувань валют, цінами на стратегічні товари, тощо.

Під час виконання роботи проведено детальний огляд методик технічного аналізу ф’ючерсного ринку Forex та програмних систем розв’язання задач алгоритмізації обробки даних на ф’ючерсних ринках, розроблено обчислювальні схеми і процедури згладжування даних, трендового сплайн-аналізу. Було реалізовано індикатори та осцилятори для обробки та аналізу даних ф’ючерсних ринків. На основі обраних методів спроектовано програмне забезпечення й розроблено програмну систему згладжування та аналізу даних ф’ючерсних ринків. Створене програмне забезпечення дозволяє зберігати й обробляти інформацію, у ручній формі та наглядному вигляді, проводити аналіз та дослідження ринкових закономірностей.

Програмний продукт реалізовано на мові C++ в середовищі візуального програмування Visual Studio 2017.

Розроблена програмна система застосована для обробки даних котирувань курсів валют або цін на основні стратегічні товари, згладжування шумів даних, визначення тренду, побудови індикаторів та осциляторів для детального аналізу ринку.

Результати тестування вказують на можливість застосування цього програмного забезпечення в реальному процесі дослідження ринку, а також для розв’язання інших реальних задач аналізу та прогнозування на ф‘ючерсних ринках.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бабак В.П., Білецький А.Я., Приставка О.П., Приставка П.О. Статистична обробка даних /Монографія. – Київ: „МІВВЦ”,2010.-388с.

2. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).

3. Бурлак Г.Н., Кузнецова О.И., Сергеева Н.В. Техника валютных операций. Практикум. – Финстатинформ, 1999 – 181с.

4. Бусигін Б.С., Коротенко Г.М., Коротенко Л.М. Прикладна інформатика. Підручник для студентів комп'ютерних спеціальностей. – Дніпропетровськ: Видавництво НГУ, 2004. – 559 с. URL: http://www.programmer.dp.ua/book-ua-k01.php. дата звернення: 15.03.2019.

5. Бусыгин Б.С., Дивизинюк М.М., Коротенко Г.М., Коротенко Л.М. Введение в современную информатику. Учебник. – Севастополь: Издательство СНУЯЭиП, 2005. – 644 с. / URL: http://www.programmer.dp.ua/book-ru-k02.php. дата звернення: 15.01.2018.

6. Бьёрн Страуструп. Язык программирования C++ = The C++ Programming Language / Пер. с англ. - 3-е изд. - СПб.; М.: Невский диалект - Бином, 1999. - 991 с. - ISBN 5-7940-0031-7 (Невский диалект), ISBN 5-7989-0127-0 (Бином), ISBN 0-201-88954-4 (англ.).

7. Вступ до програмування мовою С++. Організація обчислень : навч. посіб. / Ю. А. Бєлов, Т. О. Карнаух, Ю. В. Коваль, А. Б. Ставровський. – К. : Видавничо-поліграфічний центр "Київський університет", 2012. – 175 с. с.: іл. ISBN (укр.).

8. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

9. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.

10. Козырев А.А. Информационные технологии в экономике и управлении: учебник. -4-е изд. -СПб: Высш. Проф. Образование. 2005. -444 с.

11. Кудренко Д.В., Приставка О.П., Хохольков О.М.,Смірнов С.О. Аналіз інформаційних технологій задачі оперативного аналізу ф'ючерсних ринків : Огляд. – Дніпропетровськ: Наука і освіта, 2003. – 80 с.

12. Лигун А.А., Шумейко А.А. Ассимптотические методы восстановления кривых.-К.: ИМ НАУ, 1997.-358с.

13. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності "Комп'ютерні системи " / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

14. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 121 «Програмна інженерія» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

15. Огляд і основи мови програмування C++// URL: http://www.znannya.org/?view=Cpp_basics. Дата звернення: 12.05.2021.

16. Приставка А. Ф. Информационная технология восстановления неоднородных двумерных нормальных распределений. Актуальні проблеми автоматизації та інформаційних технологій / А. Ф. Приставка, О. Н. Мацуга – Д.: РВВ ДНУ. – 2004.– С. 20-31.

17. Приставка П. О. Поліноміальні сплайни при обробці даних: Монографія / П. О. Пристака – Д.: Вид-во Дніпропетр. ун-ту. – 2004.

18. Приставка О. П. Статистичний аналіз а АСОД. Часові ряди: редагування, прогнозування / О. П. Приставка, П. О. Приставка, С. О. Смирнов. – Д.: РВВ ДНУ. – 2001. – 92 с.

19. Приставка А. Ф. Сплайн-распределения в статистическом анализе / А. Ф. Приставка –Д.: Изд-во Днепропетр.ун-та. – 1995. – 152с.

20. Черкасов Ю.М. Информационные технологии управления: учеб. пособие/ под ред. Ю.М.Черкасова. -М : Инфра-М, 2001. -216 с.

ДОДАТОК А

КОД ПРОГРАМИ

```
#include "stdafx.h"
#include "ExRgnWin32.h"
#define MAX_LOADSTRING 100
#define DELGDIOBJECT(handle) if(handle != NULL){DeleteObject(handle); handle = NULL;}
// Global Variables:
HINSTANCE hInst;                // current instance
HWND hWnd;
TCHAR szTitle[MAX_LOADSTRING];                  // The title bar text
TCHAR szWindowClass[MAX_LOADSTRING];  // the main window class name
// Forward declarations of functions included in this code module:
void ReturnView();
void DeleteObjects();
int APIENTRY _tWinMain(HINSTANCE hInstance,
                HINSTANCE hPrevInstance,
                LPTSTR    lpCmdLine,
                int       nCmdShow)
{
        // TODO: Place code here.
        MSG msg;
        HACCEL hAccelTable;
        // Initialize global strings
        LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
        LoadString(hInstance, IDC_EXRGNWIN32, szWindowClass, MAX_LOADSTRING);
        MyRegisterClass(hInstance);
        // Perform application initialization:
        if (!InitInstance (hInstance, nCmdShow))
        {
                return FALSE;
        }
        hAccelTable = LoadAccelerators(hInstance, (LPCTSTR)IDC_EXRGNWIN32);
        // Main message loop:
```

```
        while (GetMessage(&msg, NULL, 0, 0))
        {
                if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
                {
                        TranslateMessage(&msg);
                        DispatchMessage(&msg);
                }
        }
        return (int) msg.wParam;
}
MyRegisterClass(HINSTANCE hInstance)
{       WNDCLASSEX wcex;
        wcex.cbSize = sizeof(WNDCLASSEX);
        wcex.style                      = CS_HREDRAW | CS_VREDRAW;
        wcex.lpfnWndProc    = (WNDPROC)WndProc;
        wcex.cbClsExtra             = 0;
        wcex.cbWndExtra             = 0;
        wcex.hInstance              = hInstance;
        wcex.hIcon                      = LoadIcon(hInstance, (LPCTSTR)IDI_EXRGNWIN32);
        wcex.hCursor            = LoadCursor(NULL, IDC_ARROW);
        wcex.hbrBackground = (HBRUSH) GetStockObject(BLACK_BRUSH);
        wcex.lpszMenuName = (LPCTSTR)IDC_EXRGNWIN32;
        wcex.lpszClassName = szWindowClass;
        wcex.hIconSm                = LoadIcon(wcex.hInstance, (LPCTSTR)IDI_SMALL);
        return RegisterClassEx(&wcex);
}
//
//
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
  hInst = hInstance; // Store instance handle in our global variable
  hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
     CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

  if (!hWnd)
```

```
   {
      return FALSE;
   }
   ShowWindow(hWnd, nCmdShow);
   UpdateWindow(hWnd);
   return TRUE;
}
//  FUNCTION: WndProc(HWND, unsigned, WORD, LONG)
// PURPOSE:  Processes messages for the main window.
//  WM_COMMAND - process the application menu
//  WM_PAINT         - Paint the main window
//  WM_DESTROY    - post a quit message and return
//RESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM
lParam)
{       int wmId, wmEvent;
        PAINTSTRUCT ps;
        HDC hdc;
        switch (message)
        {
        case WM_COMMAND:
                wmId    = LOWORD(wParam);
                wmEvent = HIWORD(wParam);
                // Parse the menu selections:
                switch (wmId)
                {
                case IDM_ABOUT:
                        DialogBox(hInst, (LPCTSTR)IDD_ABOUTBOX, hWnd,
(DLGPROC)About);
                        break;
        // команды теста регионов
                case IDM_TEST1:
                        {
                                // изменим вид окна
                                ViewChange();
                                // освободим память от предыдущих объектов
```

```
                    DeleteObjects();
                    hBmp = LoadBitmap(hInst,
MAKEINTRESOURCE(IDB_BITMAP1));
                    HRGN rgn = GetRgn(IDR_RGN1);
                    SetWindowRgn(hWnd, rgn, TRUE);
                }
                break;
            case IDM_TEST2:
                {
                    // изменим вид окна
                    ViewChange();
                    // освободим память от предыдущих объектов
                    DeleteObjects();
                    hBmp = LoadBitmap(hInst,
MAKEINTRESOURCE(IDB_BITMAP2));
                    HRGN rgn = GetRgn(IDR_RGN2);
                    SetWindowRgn(hWnd, rgn, TRUE);
                }
                break;
            case IDM_TEST3:
                {   // изменим вид окна
                    ViewChange();
                    // освободим память от предыдущих объектов
                    DeleteObjects();
                    hBmp = LoadBitmap(hInst,
MAKEINTRESOURCE(IDB_BITMAP3));
                    HRGN rgn = GetRgn(IDR_RGN3);
                    SetWindowRgn(hWnd, rgn, TRUE);
                }
                break;
            case IDM_TEST4:
                {   DeleteObjects();
                    hRgn = GetRgn(IDR_RGN1);
                    RECT rc;
                    //GetRgnBox(hRgn, &rc);
```

```
                GetClientRect(hWnd, &rc);
                InvalidateRect(hWnd, &rc, TRUE);
            }
            break;
        case IDM_TEST5:
            {// освободим память от предыдущих объектов
                DeleteObjects();
                hRgn = GetRgn(IDR_RGN2);
                RECT rc;
                //GetRgnBox(hRgn, &rc);
                GetClientRect(hWnd, &rc);
                InvalidateRect(hWnd, &rc, TRUE);
            }
            break;
        case IDM_TEST6:
            {
                // освободим память от предыдущих объектов
                DeleteObjects();
                hRgn = GetRgn(IDR_RGN3);
                RECT rc;
                //GetRgnBox(hRgn, &rc);
                GetClientRect(hWnd, &rc);
                InvalidateRect(hWnd, &rc, TRUE);
            }
            break;
        case IDM_EXIT:
            DestroyWindow(hWnd);
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
        }
        break;
case WM_KEYDOWN:
        {           // возврат окна в первоначальный вид
            ReturnView();
```

```
                }
                break;
                case WM_PAINT:
                hdc = BeginPaint(hWnd, &ps);
                                        if(hBmp != NULL)
                        {
                                BITMAP bitmap;
                                GetObject(hBmp, sizeof(BITMAP), &bitmap);
                                DrawState(hdc, NULL, NULL, (LPARAM)hBmp, 0,
0,0,bitmap.bmWidth,bitmap.bmHeight, DST_BITMAP);
                        }
                        if(hRgn != NULL)
                        {
                        HBRUSH brush = CreateSolidBrush(RGB(255,255,255));
                                FrameRgn(hdc, hRgn, brush, 1,1);
                        }
                }
                EndPaint(hWnd, &ps);
                break;
        case WM_DESTROY:
                PostQuitMessage(0);
                break;
        default:
                return DefWindowProc(hWnd, message, wParam, lParam);
        }
        return 0;
}
LRESULT CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM
lParam)
{       switch (message)
        {
        case WM_INITDIALOG:
                return TRUE;

        case WM_COMMAND:
```

```
                if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)
                {EndDialog(hDlg, LOWORD(wParam));
                        return TRUE;
                }
                break;
        }
        return FALSE;
}
HRGN GetRgn(UINT id_res)
{       HRSRC hrRgn = FindResource(hInst, MAKEINTRESOURCE(id_res),"RGN");
        // загрузим ресурс
        LPRGNDATA pRgnData = (LPRGNDATA)LoadResource(hInst, hrRgn);
        HRGN rgn = ExtCreateRegion(NULL, SizeofResource(hInst,hrRgn), pRgnData);
        FreeResource(pRgnData);
        return rgn;
}
// изменить вид окна
void ViewChange()
{       // запомним меню
        hMenu = GetMenu(hWnd);
        // отключим меню от окна
        SetMenu(hWnd, NULL);
                DWORD dwStyle = ::GetWindowLong(hWnd, GWL_STYLE);
        DWORD dwNewStyle = (dwStyle & ~WS_OVERLAPPEDWINDOW);
        SetWindowLong(hWnd, GWL_STYLE, dwNewStyle);
}
void ReturnView()
{
        DWORD dwStyle = ::GetWindowLong(hWnd, GWL_STYLE);
        DWORD dwNewStyle = (dwStyle | WS_OVERLAPPEDWINDOW);
        SetWindowLong(hWnd, GWL_STYLE, dwNewStyle);
        SetMenu(hWnd, hMenu);
        SetWindowRgn(hWnd, NULL, TRUE);
}
void DeleteObjects()
```

```
{       DELGDIOBJECT(hBmp);
        DELGDIOBJECT(hRgn);
}
Trans.cpp : implementation file
#include "stdafx.h"
#include "Trans3.h"
#include "TransPulse.h"
#include ".\transpulse.h"
#include <math.h>
IMPLEMENT_DYNAMIC(CTransPulse, CDialog)
CTransPulse::CTransPulse(CWnd* pParent /*=NULL*/)
        : CDialog(CTransPulse::IDD, pParent)
{
}
CTransPulse::~CTransPulse()
{
        m_EditS_V[0].Detach();
        m_EditS_V[1].Detach();
        m_EditS_V[2].Detach();
        m_EditS_V[3].Detach();
        m_EditS_V[4].Detach();
        m_StaticOut_SD[0].Detach();
        m_StaticOut_SD[1].Detach();
        m_StaticOut_SD[2].Detach();
        m_StaticOut_SD[3].Detach();
        m_StaticOut_SD[4].Detach();
        m_StaticOut_SD[5].Detach();
        m_StaticOut_SD[6].Detach();
        m_StaticOut_SD[7].Detach();
        m_StaticOut_SD[8].Detach();
        m_StaticOut_SD[9].Detach();
        m_StaticOut_SD[10].Detach();
        m_StaticOut_SC[0].Detach();
        m_StaticOut_SC[1].Detach();
        m_StaticOut_SC[2].Detach();
```

```
	m_StaticOut_SC[3].Detach();
	m_StaticOut_SC[4].Detach();
	m_StaticOut_SC[5].Detach();
	m_StaticOut_SC[6].Detach();
}
void CTransPulse::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	DDX_Control(pDX, IDC_EDIT_PV1, m_EditPV1);
	DDX_Control(pDX, IDC_STATIC_PC0, m_StaticPC0);
	DDX_Control(pDX, IDC_STATIC_PD0, m_StaticPD0);
	DDX_Control(pDX, IDC_STATIC_P, m_P1);
	DDX_Control(pDX, IDC_STATIC_attention, m_StaticAttention);
}
BEGIN_MESSAGE_MAP(CTransPulse, CDialog)
	ON_EN_CHANGE(IDC_EDIT_SA6, OnEnChangeEditSa6)
ON_EN_CHANGE(IDC_EDIT_SA7, OnEnChangeEditSa7)
ON_EN_CHANGE(IDC_EDIT_SA8, OnEnChangeEditSa8)
ON_EN_CHANGE(IDC_EDIT_SA9, OnEnChangeEditSa9)
ON_EN_CHANGE(IDC_EDIT_SV4, OnEnChangeEditSv4)
ON_EN_CHANGE(IDC_EDIT_SV5, OnEnChangeEditSv5)
ON_EN_CHANGE(IDC_EDIT_SV6, OnEnChangeEditSv6)
ON_EN_CHANGE(IDC_EDIT_OUTWARDDIAM, OnEnChangeEditOutwarddiam)
ON_EN_CHANGE(IDC_EDIT_FREQ, OnEnChangeEditFreq)
ON_EN_UPDATE(IDC_EDIT_PV1, OnEnUpdateEditPv1)
ON_EN_UPDATE(IDC_EDIT_TESLA, OnEnUpdateEditTesla)
ON_EN_UPDATE(IDC_EDIT_VKE, OnEnUpdateEditVke)
ON_WM_CTLCOLOR()
END_MESSAGE_MAP()
// CTransPulse message handlers
BOOL CTransulse::OnInitDialog()
{	CDialog::OnInitDialog();
	// TODO:  Add extra initialization here
		m_EditS_V[0].Attach(GetDlgItem(IDC_EDIT_SV0)->m_hWnd);
	m_EditS_V[1].Attach(GetDlgItem(IDC_EDIT_SV1)->m_hWnd);
```

```
	m_EditS_V[2].Attach(GetDlgItem(IDC_EDIT_SV2)->m_hWnd);
	m_EditS_V[3].Attach(GetDlgItem(IDC_EDIT_SV3)->m_hWnd);
	m_EditS_V[4].Attach(GetDlgItem(IDC_EDIT_SV4)->m_hWnd);

	m_EditS_A[0].Attach(GetDlgItem(IDC_EDIT_SA0)->m_hWnd);
	m_EditS_A[1].Attach(GetDlgItem(IDC_EDIT_SA1)->m_hWnd);
	m_EditS_A[2].Attach(GetDlgItem(IDC_EDIT_SA2)->m_hWnd);
	m_EditS_A[3].Attach(GetDlgItem(IDC_EDIT_SA3)->m_hWnd);
	m_EditS_A[4].Attach(GetDlgItem(IDC_EDIT_SA4)->m_hWnd);
	m_EditS_A[5].Attach(GetDlgItem(IDC_EDIT_SA5)->m_hWnd);

	m_StaticOut_SD[0].Attach(GetDlgItem(IDC_STATIC_SD0)->m_hWnd);
	m_StaticOut_SD[1].Attach(GetDlgItem(IDC_STATIC_SD1)->m_hWnd);
	m_StaticOut_SD[2].Attach(GetDlgItem(IDC_STATIC_SD2)->m_hWnd);
	m_StaticOut_SD[3].Attach(GetDlgItem(IDC_STATIC_SD3)->m_hWnd);
	m_StaticOut_SD[4].Attach(GetDlgItem(IDC_STATIC_SD4)->m_hWnd);
	m_StaticOut_SD[5].Attach(GetDlgItem(IDC_STATIC_SD5)->m_hWnd);
	m_StaticOut_SD[6].Attach(GetDlgItem(IDC_STATIC_SD6)->m_hWnd);

	m_StaticOut_SC[0].Attach(GetDlgItem(IDC_STATIC_SC0)->m_hWnd);
	m_StaticOut_SC[1].Attach(GetDlgItem(IDC_STATIC_SC1)->m_hWnd);
	m_StaticOut_SC[2].Attach(GetDlgItem(IDC_STATIC_SC2)->m_hWnd);
	m_StaticOut_SC[3].Attach(GetDlgItem(IDC_STATIC_SC3)->m_hWnd);
	m_StaticOut_SC[4].Attach(GetDlgItem(IDC_STATIC_SC4)->m_hWnd);
	m_StaticOut_SC[5].Attach(GetDlgItem(IDC_STATIC_SC5)->m_hWnd);
	m_StaticOut_SC[6].Attach(GetDlgItem(IDC_STATIC_SC6)->m_hWnd);
	m_EditVKE.SetWindowText("1.6");
	CFont font;
	font.CreatePointFont(20, "Arial Black");
	m_StaticSign.SetFont(&font);
	CString str_Help(".");
	m_StaticHelp.SetWindowText(str_Help);
	return TRUE;  // return TRUE unless you set the focus to a control
	// EXCEPTION: OCX Property Pages should return FALSE
}
```

```
void CTransPulse::OnEnChangeEditPv1()
{       // TODO:  Add your control notification handler code here
        OnCalculation();
}
void CTransPulse::OnEnChangeEditSa0()
{       // TODO:  Add your control notification handler code here
        OnCalculation();
}
void CTransPulse::OnEnChangeEditSa1()
{       // TODO:  Add your control notification handler code here
        OnCalculation();
}
void CTransPulse::OnEnChangeEditSa2()
{       // TODO:  Add your control notification handler code here
        OnCalculation();
}

void CTransPulse::OnEnChangeEditSa3()
{       // TODO:  Add your control notification handler code here
        OnCalculation();
}
void CTransPulse::OnEnChangeEditSa4()
{       // TODO:  Add your control notification handler code here
        OnCalculation();
}
void CTransPulse::OnEnChangeEditSa5()
{       // TODO:  Add your control notification handler code here
        OnCalculation();
}
void CTransPulse::OnEnChangeEditSa6()
{       // TODO:  Add your control notification handler code here
        OnCalculation();
}
void CTransPulse::OnEnChangeEditSa7()
{       // TODO:  Add your control notification handler code here
```

```
	OnCalculation();
}void CTransPulse::OnEnChangeEditSa8()
{	// TODO:  Add your control notification handler code here
	OnCalculation();
}
void CTransPulse::OnEnChangeEditSa9()
{	// TODO:  Add your control notification handler code here
	OnCalculation();
}
void CTransPulse::OnEnChangeEditSa10()
{	// TODO:  Add your control notification handler code here
	OnCalculation();
}
void CTransPulse::OnEnChangeEditSa11()
{	// TODO:  Add your control notification handler code here
	OnCalculation();
}
void CTransPulse::OnEnChangeEditSv0()
{	// TODO:  Add your control notification handler code here
	OnCalculation();
}
void CTransPulse::OnEnChangeEditSv1()
{	// TODO:  Add your control notification handler code here
	OnCalculation();
}
void CTransPulse::OnEnChangeEditSv2()
{	// TODO:  Add your control notification handler code here
	OnCalculation();
}
void CTransPulse::OnEnChangeEditSv3()
{	// TODO:  Add your control notification handler code here
	OnCalculation();
}
void CTransPulse::OnEnChangeEditSv4()
{	// TODO:  Add your control notification handler code here
```

```
	OnCalculation();
}
void CTransPulse::OnEnChangeEditSv5()
{	// TODO:  Add your control notification handler code here
	OnCalculation();
}
void CTransPulse::OnEnChangeEditSv6()
{	// TODO:  Add your control notification handler code here
	OnCalculation();
}
void CTransPulse::OnEnChangeEditSv7()
{	// TODO:  Add your control notification handler code here
	OnCalculation();
}
void CTransPulse::OnEnChangeEditSv8()
{	// TODO:  Add your control notification handler code here
	OnCalculation();
}
void CTransPulse::OnEnChangeEditSv9()
{	// TODO:  Add your control notification handler code here
	OnCalculation();
}
void CTransPulse::OnEnChangeEditSv10()
{	// TODO:  Add your control notification handler code here
	OnCalculation();
}
void CTransPulse::OnEnChangeEditSv11()
{	// TODO:  Add your control notification handler code here
	OnCalculation();
}
void CTransPulse::OnCalculation(void)
{	float P1 = 0.0, P2[12] = {0.0}, Pn = 0.0;
	CString str_PV;
	m_EditPV1.GetWindowText(str_PV);
	int pri_V = atoi(str_PV);
```

```
int sec_V[12];
float sec_A[12];
for(int p = 0; p < 12; p++){
	CString str_SV, str_SA;
	if(m_EditS_V[p].IsWindowVisible() == TRUE){
		m_EditS_V[p].GetWindowText(str_SV);
		sec_V[p] = atoi(str_SV);
		m_EditS_A[p].GetWindowText(str_SA);
		sec_A[p] = (float)atof(str_SA);

		P2[p] = sec_V[p]*sec_A[p];

		Pn += (float)P2[p];
		P1 += (float)(P2[p]*1.3);
	}
}
CString str_P1;
str_P1.Format("%.1f", P1);
if(P1 != 0){
	//str_P1 += " Вт";
	m_P1.SetWindowText(str_P1);
}	else
	m_P1.SetWindowText("#####");
// w1=0,25*10^4*U1/f*Bmax*Sc.
CWnd* pWndFreg = GetDlgItem(IDC_EDIT_FREQ);
CString str_Freg;
pWndFreg->GetWindowText(str_Freg);
int Freg = atoi(str_Freg);
CWnd* pWndTesla = GetDlgItem(IDC_EDIT_TESLA);
CString str_Tesla;
pWndTesla->GetWindowText(str_Tesla);
float Tesla = (float)atof(str_Tesla);
CWnd* pWndH = GetDlgItem(IDC_EDIT_HEIGHT);
CString str_Height;
pWndH->GetWindowText(str_Height);
```

```
float Height = (float)atof(str_Height)/10;
CWnd* pWndInWard = GetDlgItem(IDC_EDIT_INWARDDIAM);
CString str_InWard;
pWndInWard->GetWindowText(str_InWard);
float diamInWard = (float)atof(str_InWard)/10;
CWnd* pWndOutWard = GetDlgItem(IDC_EDIT_OUTWARDDIAM);
CString str_OutWard;
pWndOutWard->GetWindowText(str_OutWard);
float diamOutWard = (float)atof(str_OutWard)/10;
float Sc =  (diamOutWard-diamInWard)*Height/2;
float w1 = 0;
CString str_VKE;
m_EditVKE.GetWindowText(str_VKE);
int U1 = (int)((pri_V/2)-atof(str_VKE));
w1 =  (float)0.25*10000*U1/(Freg*Tesla*Sc);
CString str_w1; float temp_w1 = 0.0;
if(w1 > 0 && Freg != 0 && Tesla != 0 && Sc != 0){
        temp_w1 = w1 + 1;
        str_w1.Format("%.0f", temp_w1);
        m_StaticPC0.SetWindowText(str_w1);
}
else
        m_StaticPC0.SetWindowText("#####");
//w2=w1*U2/U1
int w2 = 0;
for(int sV = 0; sV < 12; sV++){
        if(m_EditS_V[sV].IsWindowVisible() == TRUE){
                if(U1 != 0)
                        w2 = (int)temp_w1*sec_V[sV]/U1;
                CString str_w2; str_w2.Format("%d", w2);
                if(sec_V[sV] != 0 && sec_A[sV] != 0)
                        m_StaticOut_SC[sV].SetWindowText(str_w2);
                else
                        m_StaticOut_SC[sV].SetWindowText("#####");
        }
```

```
}
//d1=0,6*(кв. корень из I1max)
float I1max = (float)(Pn/(0.77*U1));
float d1 = (float)(0.6*sqrt(I1max));
//вывод на табло
CString str_d1;
str_d1.Format("%.2f", d1);
if(U1 != 0 && Pn != 0 && d1 >= 0)
{
	float section = (float)AreaCircle(d1);
	CString str_section;
	str_section.Format("%.2f", section);
	str_d1 += "/";
	str_d1 += str_section;
	m_StaticPD0.SetWindowText(str_d1);
}
else
	m_StaticPD0.SetWindowText("#####");
//d2=0.6*(кв. корень из I2max),
for(int sA = 0; sA < 12; sA++){
	if(m_EditS_V[sA].IsWindowVisible() == TRUE){
		float d2 = (float)(0.6*sqrt(sec_A[sA]));
		//вывод на табло
		CString str_d2;
		str_d2.Format("%.2f", d2);
		if(sec_V[sA] != 0 && sec_A[sA] != 0){
			float section = (float)AreaCircle(d2);
			CString str_section;
			str_section.Format("%.2f", section);
			str_d2 += "/";
			str_d2 += str_section;
			m_StaticOut_SD[sA].SetWindowText(str_d2);
		}
		else
			m_StaticOut_SD[sA].SetWindowText("#####");
```

```
        }
}       CString str_PGab;
GetDlgItem(IDC_STATIC_PGAB)->GetWindowText(str_PGab);
float fGabPower = (float)atof(str_PGab);
CString strUsePower;
GetDlgItem(IDC_STATIC_P)->GetWindowText(strUsePower);
float fUsePower = (float)atof(strUsePower);
if(fGabPower >= fUsePower)
        m_StaticAttention.ShowWindow(SW_HIDE);
else
        m_StaticAttention.ShowWindow(SW_SHOW);
        CEdit* pwndPower = (CEdit*)GetDlgItem(IDC_STATIC_P);
CString s;
pwndPower->GetWindowText(s);
if(atof(s) > 600){
        CString str_Edit;
        CWnd* pWnd = GetFocus();
        if(pWnd != NULL)
                pWnd->GetWindowText(str_Edit);
        if(str_Edit.GetLength() > 0){
                AfxMessageBox("Расчет ведется до мощности 500 Вт");
                str_Edit.Delete(str_Edit.GetLength()-1);
                pWnd->SetWindowText(str_Edit);
        }
}
CString str_Edit;
CEdit* pWndFocus = (CEdit*)GetFocus();
if(pWndFocus != NULL){
        pWndFocus->GetWindowText(str_Edit);
        pWndFocus->SetSel(LOWORD(str_Edit.GetLength()), str_Edit.GetLength());
}
        CEdit* pWnd = NULL;
pWnd = (CEdit*)GetFocus();
if(pWnd != NULL){
        CString str_Edit;
```

```
		pWnd->GetWindowText(str_Edit);
		pWnd->SetSel(LOWORD(str_Edit.GetLength()), str_Edit.GetLength());
	}
	*/
}void CTransPulse::OnCalcGabPower(void)
{
	CWnd* pWndInWard = GetDlgItem(IDC_EDIT_INWARDDIAM);
	CString str_InWard;
	pWndInWard->GetWindowText(str_InWard);
	float diamInWard = (float)atof(str_InWard)/10;
	float So = (float)AreaCircle(diamInWard);
	CWnd* pWndH = GetDlgItem(IDC_EDIT_HEIGHT);
	CString str_Height;
	pWndH->GetWindowText(str_Height);
	float Height = (float)atof(str_Height)/10;
	CWnd* pWndOutWard = GetDlgItem(IDC_EDIT_OUTWARDDIAM);
	CString str_OutWard;
	pWndOutWard->GetWindowText(str_OutWard);
	float diamOutWard = (float)atof(str_OutWard)/10;
	float Sc = (float)(diamOutWard-diamInWard)*Height/2;
	CWnd* pWndFreg = GetDlgItem(IDC_EDIT_FREQ);
	CString str_Freg;
	pWndFreg->GetWindowText(str_Freg);
	int Freg = atoi(str_Freg);

	CWnd* pWnd = GetDlgItem(IDC_EDIT_TESLA);
	CString str_Tesla;
	pWndTesla->GetWindowText(str_Tesla);
	float Tesla = (float)atof(str_Tesla);
	float Pgab = Sc*So*Freg*Tesla/150;
	CWnd* pWndStaticGab = GetDlgItem(IDC_STATIC_PGAB);
	CString str_PGab;
	str_PGab.Format("%.1f", Pgab);
	pWndStaticGab->SetWindowText(str_PGab);
		OnCalculation();
```

```
}
void CTransPulse::OnEnChangeEditHeight()
{
	OnCalcGabPower();
}
void CTransPulse::OnEnChangeEditInwarddiam()
{
	OnCalcGabPower();
}
void CTransPulse::OnEnChangeEditOutwarddiam()
{
	OnCalcGabPower();
}
void CTransPulse::OnEnChangeEditFreq()
{
	OnCalcGabPower();
}
void CTransPulse::OnEnChangeEditTesla()
{
	OnCalcGabPower();
}
void CTransPulse::OnEnChangeEditVke()
{	OnCalcGabPower();
}
void CTransPulse::IsCorrectnumber(CEdit* pWnd)
{	CString str_Edit;
	pWnd->GetWindowText(str_Edit);
	CPoint pt = ((CEdit*)pWnd)->GetCaretPos();
	int curPos = ((CEdit*)pWnd)->CharFromPos(pt);
	int countPoint = 0;
	for(int i = 0; i < str_Edit.GetLength(); i++){
		if(	str_Edit[i] != '1' && str_Edit[i] != '2' &&
			str_Edit[i] != '3' && str_Edit[i] != '4' &&
			str_Edit[i] != '5' && str_Edit[i] != '6' &&
			str_Edit[i] != '7' && str_Edit[i] != '8' &&
```

```
                    str_Edit[i] != '9' && str_Edit[i] != '0' &&
                    str_Edit[i] != '.'){
                        str_Edit.Delete(i);
                        pWnd->SetWindowText(str_Edit);
                    }
                    if(str_Edit[i] == '.')
                        countPoint++;
        }           for(int j = 0; j < str_Edit.GetLength(); j++){
            if(countPoint > 1)
                    if(str_Edit[j] == '.' && j != curPos){
                        str_Edit.Delete(j);
                        pWnd->SetWindowText(str_Edit);
                        countPoint--;
                    }
        }
        if(str_Edit[0] == '.' &&  str_Edit[1] != '0'){
            CString str = "0";
            str += str_Edit;
            pWnd->SetWindowText(str);
        }
        else{
            if(str_Edit.GetLength() > 1)
            if(str_Edit[1] != '.' && str_Edit[0] == '0'){
                str_Edit.Delete(0);
                pWnd->SetWindowText(str_Edit);
            }
        }
}
void CTransPulse::OnEnUpdateEditPv1()
{       IsCorrectnumber(&m_EditPV1);
}
void CTransPulse::OnEnUpdateEditTesla()
{       IsCorrectnumber(&m_EditTesla);
}
void CTransPulse::OnEnUpdateEditVke()
```

```
{       IsCorrectnumber(&m_EditVKE);
}
void CTransPulse::OnEnUpdateEditSa0()
{       IsCorrectnumber(&m_EditS_A[0]);
}
void CTransPulse::OnEnUpdateEditSa1()
{       IsCorrectnumber(&m_EditS_A[1]);
}
void CTransPulse::OnEnUpdateEditSa2()
{       IsCorrectnumber(&m_EditS_A[2]);
}
void CTransPulse::OnEnUpdateEditSa3()
{       IsCorrectnumber(&m_EditS_A[3]);
}
void CTransPulse::OnEnUpdateEditSa4()
{       IsCorrectnumber(&m_EditS_A[4]);
}
void CTransPulse::OnEnUpdateEditSa5()
{       IsCorrectnumber(&m_EditS_A[5]);
}
void CTransPulse::OnEnUpdateEditSa6()
{       IsCorrectnumber(&m_EditS_A[6]);
}
void CTransPulse::OnEnUpdateEditSa7()
{       IsCorrectnumber(&m_EditS_A[7]);
}
void CTransPulse::OnEnUpdateEditSa8()
{       IsCorrectnumber(&m_EditS_A[8]);
}
void CTransPulse::OnEnUpdateEditSa9()
{       IsCorrectnumber(&m_EditS_A[9]);
}
void CTransPulse::OnEnUpdateEditSa10()
{       IsCorrectnumber(&m_EditS_A[10]);
}
```

```
void CTransPulse::OnEnUpdateEditSa11()
{	IsCorrectnumber(&m_EditS_A[11]);
}
HBRUSH CTransPulse::OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{	HBRUSH hbr = CDialog::OnCtlColor(pDC, pWnd, nCtlColor);
	// TODO:  Change any attributes of the DC here
	pDC->SetBkMode(TRANSPARENT);
	pDC->SetTextColor(RGB(100, 0, 0));
	// TODO:  Return a different brush if the default is not desired
	return hbr;
}
BOOL CTransPulse::PreTranslateMessage(MSG* pMsg)
{	// TODO: Add your specialized code here and/or call the base class
	for(int i = 0; i < 12; i++){
		if(pMsg->hwnd == m_EditS_V[i].m_hWnd)
			if(pMsg->message == WM_CHAR){
				if(pMsg->wParam == ',')
					pMsg->wParam = '.';
			}
	}
	for(int i = 0; i < 12; i++){
		if(pMsg->hwnd == m_EditS_A[i].m_hWnd)
			if(pMsg->message == WM_CHAR){
				if(pMsg->wParam == ',')
					pMsg->wParam = '.';
			}
	}	return CDialog::PreTranslateMessage(pMsg);
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи