НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
Кафедра системного програмування і спеціалізованих комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ В.П. Тарасенко
(підпис)
"___"_____2019р.

Дипломний проект

освітньо-кваліфікаційного рівня "Бакалавр"

з напряму підготовки 6.050102 "Комп'ютерна інженерія"

на тему: «АВТОМАТИЗОВАНА ІНТЕЛЕКТУАЛЬНА СИСТЕМА МОНІТОРИНГУ РИНКУ»

Виконав: студент IV курсу, групи КВ-51
Чугаєвський В`ячеслав Андрійович _____
(підпис)

Керівник асистент каф. СПіСКС Радченко К.О. _____
(підпис)

Консультант кандидат технічних наук, доцент, Клятченко Я.М. _____
(підпис)

Рецензент доц., к.т.н. Марковський О.П. _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет прикладної математики

Кафедра системного програмування та спеціалізованих комп’ютерних систем

Освітньо-кваліфікаційний рівень “Бакалавр”

Напрям підготовки 6.050102 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ В.П. Тарасенко
“___” _____ 2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Чугаєвському В’ячеславу Андрійовичу

1. Тема проекту «АВТОМАТИЗОВАНА ІНТЕЛЕКТУАЛЬНА СИСТЕМА МОНІТОРИНГУ РИНКУ»,
керівник проекту Радченко Костянтин Олександрович , асистент каф. СПіСКС,
затверджені наказом по університету від 22.05.2019 року № 1330-С

2. Строк подання студентом проекту: “___” _____ 2019 р.

3. Вихідні дані для дипломного проектування: див. Технічне завдання.

4. Перелік задач, які потрібно вирішити:
   - розробити загальну структуру інтелектуальної системи моніторингу ринку;
   - реалізувати безперервну обробку даних;
   - розробити систему технічного аналізу в реальному часі;
   - розробити алгоритм поведінки системи;
   - виконати тестування інтелектуальної системи ринку.

5. Перелік обов’язкового ілюстративного матеріалу:
   - структура взаємозв’язків інтелектуальної системи(схема);
   - алгоритм технічного аналізу(схема);
   - алгоритм прийняття рішення(схема);
   - алгоритм управління ризиком(схема);

6. Консультанти:

Питання	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., доц., к.т.н.		

7. Дата видачі завдання: “___” _____ 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	05.05.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
6.	Підготовка графічної частини дипломного проекту	20.05.2019
7.	Оформлення документації дипломного проекту	25.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Студент _____ Чугаєвський В.А.

Керівник проекту _____ Радченко К.О.

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (___ с., ___ рис., ___ табл., ___ додатки).

Об'єкт розробки - Створення автоматизованої інтелектуальної системи для моніторингу ринку, та прийняття рішень на основі технічного аналізу даних.

Мета проекта: Автоматизувати процес обробки даних, технічний аналіз і автономне прийняття рішень, відповідно поточній ситуації ринку.

Область застосування: Застосовується в економічній сфері для технічного аналізу, і автономної торгівлі на різних фінансових ринках.

Дана система дозволять, в реальному часі обробляти потік інформації і проводити технічний аналіз отриманих даних ринках, що дозволяє зробити висновки щодо поточної ситуації ринку і автоматизувати процес прийняття рішень.

Ключові слова: технічний аналіз, інтелектуальна система, моніторинг ринка, автоматизація.

ABSTRACT

Qualification work includes explanation note (___ p., ___ pic., ___ tab., ___ additions).

Object of development - Creation of an automated intellectual system for monitoring the market, and making decisions based on technical data analysis.

The purpose of the project: Automate data processing, technical analysis and autonomous decision-making, corresponding to the current market situation.

Scope: Applicable in the economic field for technical analysis, and autonomous trade in various economic markets.

This system will allow, in real time, to process the flow of information and carry out technical analysis of the data obtained in the markets, which allows to draw conclusions on the current market situation and automate the decision-making process.

Keywords: technical analysis, intellectual system, monitoring of the market, automation.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045490.002 ТЗ	Автоматизована	4		
			інтелектуальна система			
			моніторингу ринку			
			Технічне завдання			
	А4	ІАЛЦ.045490.003 ТП	Автоматизована	2		
			інтелектуальна система			
			моніторингу ринку			
			Автоматизована			
			Відомість проекту			
	А4	ІАЛЦ.045490.004 ПЗ	Автоматизована	52		
			інтелектуальна система			
			моніторингу ринку			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Структура взаємозв’язків	1		
			інтелектуальної системи			
			Схема структурна			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.001 ОА		
Розроб.		Чугаєвський В.А.			Автоматизована інтелектуальна система моніторингу ринку **Опис альбому**	Літ.	Аркуш 1 / Аркушів 2
Перевір.		Радченко К.О.					
Н. контр.		Клятченко Я.М.				КПІ імені Ігоря Сікорського, ФПМ КВ-51	
Затвер.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.006 Д2	Алгоритм технічного	1		
			аналізу			
			Схема алгоритму			
	А4	ІАЛЦ.045490.007 Д3	Алгоритм прийняття	1		
			рішення			
			Схема алгоритму			
	А4	ІАЛЦ.045490.008 Д4	Алгоритм управління	1		
			ризиком			
			Схема алгоритму			
		Диск CD-ROM	Матеріали дипломного			
			проекту.			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.001 ОА	Арк. 2

ЗМІСТ

стор.

					ІАЛЦ.045490.002 ТЗ		
Зм.	Арк.	№ докум.	Підп.	Дата			
Розроб.		Чугаєвський В.А.			Автоматизована інтелектуальна система моніторингу ринку **Технічне завдання**	Літ.	Аркуш 1 / Аркушів 4
Перевір.		Радченко К.О.					
Н. контр.		Клятченко Я.М.				КПІ імені Ігоря Сікорського, ФПМ КВ-51	
Затв.		Тарасенко В.П.					

1. НАЙМЕНУВАННЯ І ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування роботи – «Автоматизована інтелектуальна система моніторингу ринку».

Область застосування: для проведення технічного аналізу і моніторингу різних фінансових ринків.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є автоматизувати процес обробки даних, технічний аналіз і автономне прийняття рішень, відповідно поточній ситуації ринку.

4. ДЖЕРЕЛА РОБОТИ

Конспект курсу «теорія ймовірності та математична статистика», наукова література з теорії побудови моделі ринку на базі фундаментального та технічного аналізу, технічна література з теорії проектування інтелектуальних систем, статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до алгоритму технічного аналізу

- На вхід приймає данні ринку.
- Виконує алгоритм математичного розподілу даних.
- Прораховує ймовірність відхилу границі від поточної.
- Виходом є дані технічного аналізу, за весь період роботи програми.

5.2 Вимоги до алгоритму пошуку рішення

- На вхід приймає дані технічного аналізу.
- Аналізує інструкції зазначені в технічному аналізі.
- Виконує пошук серед внутрішніх інструкцій.
- В разі знайденої дії виконує вимоги зазначені в інструкції.
- Якщо дія не була знайдена, переходить до дії відмови, і переходить до кроку технічного аналізу.

5.3 Вимоги до програмного забезпечення:

- Операційна система Windows.
- Metatrader 4.
- Браузер.

5.4 Вимоги до апаратного забезпечення:

- Монітор.
- Комп’ютер.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	05.05.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
6.	Підготовка графічної частини дипломного проекту	20.05.2019
7.	Оформлення документації дипломного проекту	25.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045490.004 ПЗ	Автоматизована	52		
			інтелектуальна система			
			моніторингу ринку			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Структура взаємозв’язків	1		
			інтелектуальної системи			
			Схема структурна			
	А4	ІАЛЦ.045490.006 Д2	Алгоритм технічного	1		
			аналізу			
			Схема алгоритму			
	А4	ІАЛЦ.045490.007 Д3	Алгоритм прийняття	1		
			рішення			
			Схема алгоритму			
	А4	ІАЛЦ.045490.008 Д4	Алгоритм управління	1		
			ризиком			
			Схема алгоритму			

					ІАЛЦ.045490.003 ТП
Змін.	Арк.	№ докум.	Підпис	Дата	
Розроб.		Чугаєвський В.А.			Автоматизована інтелектуальна система моніторингу ринку **Відомість дипломного проекту**
Перевір.		Радченко К.О.			
Н. контр.		Клятченко Я.М.			
Затвер.		Тарасенко В.П.			

Літ.	Аркуш	Аркушів
	1	2

КПІ імені Ігоря Сікорського, ФПМ КВ-51

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
		Диск CD-ROM	Матеріали дипломного			
			проекту.			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП	Арк. 2

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

БД – База даних

ПЗ – програмне забезпечення

Metatrader 4 – торгова платформа, реалізує доступ до ринку

MQL4 – алгоритмічна мова програмування MetaQuotes Language 4

Індикатор – інструмент для аналізу ринкової ситуації, технічного аналізу

Order – позиція, на покупку або продаж на фінансовому ринку

Таймфрейм - інтервал часу, який використовується для угруповання котирувань при побудові елементів цінового графіка

Математична модель - система математичних співвідношень, які описують досліджуваний процес або явище

МТС – механічна торгова система

АТС – автоматична торгова система

ChartObject – об’єкт діаграми побудований в результаті математичної обробки інструментами аналізу

MQH – заголовний тип файлу, описуючий структуру об’єкту, та залежні файли.

FIX – протокол передачі даних, що є міжнародним стандартом для обміну даними ринковими даними в режимі реального часу

Tick – часовий імпульс зміни даних отриманих за протоколом FIX.

ВСТУП

Сьогодні в епоху автоматизації та роботизації важко уявити процеси, роботу, які в теорії не можливо реалізувати завдяки автоматизованим системам. Розглядаючи економічну сферу, ми можемо уявити роботу з великим обсягом інформації, яка потребує подальшого аналізу, висновку та прийнята рішення за цими даними.

В економічній сфері часто застосовуються технічні навички та математичні які добре працюють в симбіозі, оскільки майже всі явища економічного характеру можна описати з математичної точки зору. Завдяки математичним моделям, технічним аналізам, оцінці ризику, прорахунку математично статичного розподілу та в поєднані з прорахунком ймовірності можна отримати корисну інформацію для людини, яка представляє цінність. Робота по аналізу такого обсягу інформації вручну є достатньо енергозатратною та не ефективною. Рішенням проблеми – є повна автоматизація процесу збору даних, аналізу та прийнята подальших дій на основі обробленої інформації. Сфера реалізації подібної системи достатньо широка, починаючи від управління власним капіталом закінчуючи торговими операціями на фінансових ринках.

Метою цієї роботи є гармонічне поєднання різнотипних сфер діяльності, з практичної точки зору, а також демонстрація переваг технічно-математичного підходу побудови алгоритмів та аналізу систематичних даних. Теж метою проекту є те, щоб продемонструвати переваги автоматизації процесів обробки даних на прикладі втілення інтелектуальної автоматизованої системи моніторингу ринку в фінансовій сфері. Подібна практика впровадження може позитивно вплинути на розвиток різнотипних сфер діяльності та інтересу до технічної сфери.

1. АНАЛІЗ ІСНУЮЧИХ АЛГОРИТМІВ ТА СИСТЕМ ОБРОБКИ ДАНИХ

1.1. Аналіз сервісів аналітики ринків

На сьогоднішній дені існує багато різних сервісів аналізу ринкової інформації. Одним з широко розповсюджених є веб-сервіси «календарі-новин», які збирають загально відомі дані, про майбутні події, сортують їх за датою, важністю, та роблять прогноз реакції ринку на ці події.

Переваги:

- Доступність
- Висока інформаційна цінність
- Не потребує налаштування, примітивний інтерфейс

Недоліки:

- Зазвичай прогнози, це лише припущення, не факт що ця подія покаже себе саме так
- Може використовуватися для дезінформації, або в інших маніпулятивних цілях
- Потребує виконувати рішення, і аналіз даних вручну, що потребує певних навичок

Не менш розповсюджені є веб-сервіси «ринкові сигнали», вони працюють за певними алгоритмами аналізу чи шаблонами, при спрацьовуванні яких користувачу відправляється сигнал, з рекомендаціями щодо прийняття рішень в конкретний момент.

Переваги:

- Простий в експлуатації, можливо запустити і на платформі MT4
- Не потребує спеціальних навичок, для аналізу інформації.

Недоліки:

- Ці сервіси платні і працюють за системою підписки.
- Великий ризик помилкового прогнозу
- Order – відкривається вручну

Expert advision – система автоматичної торгівлі, побудована за принципом запрограмованих алгоритмів, правил поведінки в залежності від ситуації на ринку [1].

Переваги:

- Велика різноманітність
- Автоматичне прийняття рішень
- Можливість аналізу ефективності роботи системи
- При прийняті рішень виключений людський фактор
- Пряма взаємодія з платформою MT4

Недоліки:

- Дані системи є платними
- Складність експлуатації та обновлення системи.

Саме цей тип сервісу підходить для реалізації теми дипломного проекту, оскільки, цей тип дозволяє нам на пряму взаємодіяти з платформою MT4, та реалізувати, систему обробки даних, аналізу, та прийнята рішення, що по факту і є автоматична торгівля. В системі моніторингу ринку автоматична торгівля є, по факту, результатом правильної роботи системи, тобто кінцевою і успішною роботою усіх модулів системи.

Найбільшим недоліком цих систем є те, що алгоритми закладені в їх основу працюють лише в при назначених для них умовах, тобто вони не є універсальними. Ідея проекту полягає в вирішені цього недоліка, реалізацією універсальних умов за допомогою технічної структури взаємодії, розбиття алгоритму автоматичної торгівлі на окремі структури: обробку даних,

технічний аналіз, прийняття рішення. Побудувати логічно-інтелектуальну систему взаємодії цих компонентів та алгоритми аналізу, спроектувати систему, яка базується на технічному аналізі, а саме побудованій на теорії ймовірності події.

1.2. Алгоритми аналізу ринку

Аналітика ринку – це збір інформації про об’єкт ринкової діяльності або конкретного клієнта, яка має цінність і може бути виражена як статистична інформація.

При аналітиці будь якого фінансового ринку існує два основні типу аналізу інформації: фундаментальний та технічний. Фундаментальний аналіз, який ґрунтується на зборі залежної абстрактної інформації, яка може бути фактором впливу на економічну ситуацію країни, підприємства тощо. Наступним є технічний аналіз, який оброблює числову статистичну інформацію, завдяки побудови математичної моделі, або алгоритмів статистичного розподілу.

Більш детально розглянемо два підходи, їх переваги та недоліки для реалізації автоматизованої системи. Ресурсом фундаментального аналізу, зазвичай, є вибірка новин, висновки, прогнози щодо економічної ситуації, промови та офіційні виступи світових економічних лідерів. Вся інформації є абстрактною, двоякою і потребує обробки спеціалістами або повного розуміння ситуації, яка оброблюється. Це зручно для людини, але не підходить для побудови конкретної системи обробки даних, оскілки, ці данні містять багато абстрактної інформації, яку система не здатна розуміти. Також важливим аспектом є те, що кожен ресурс є унікальним і не шаблонізованим, що ускладнює алгоритм аналізу цим методом.

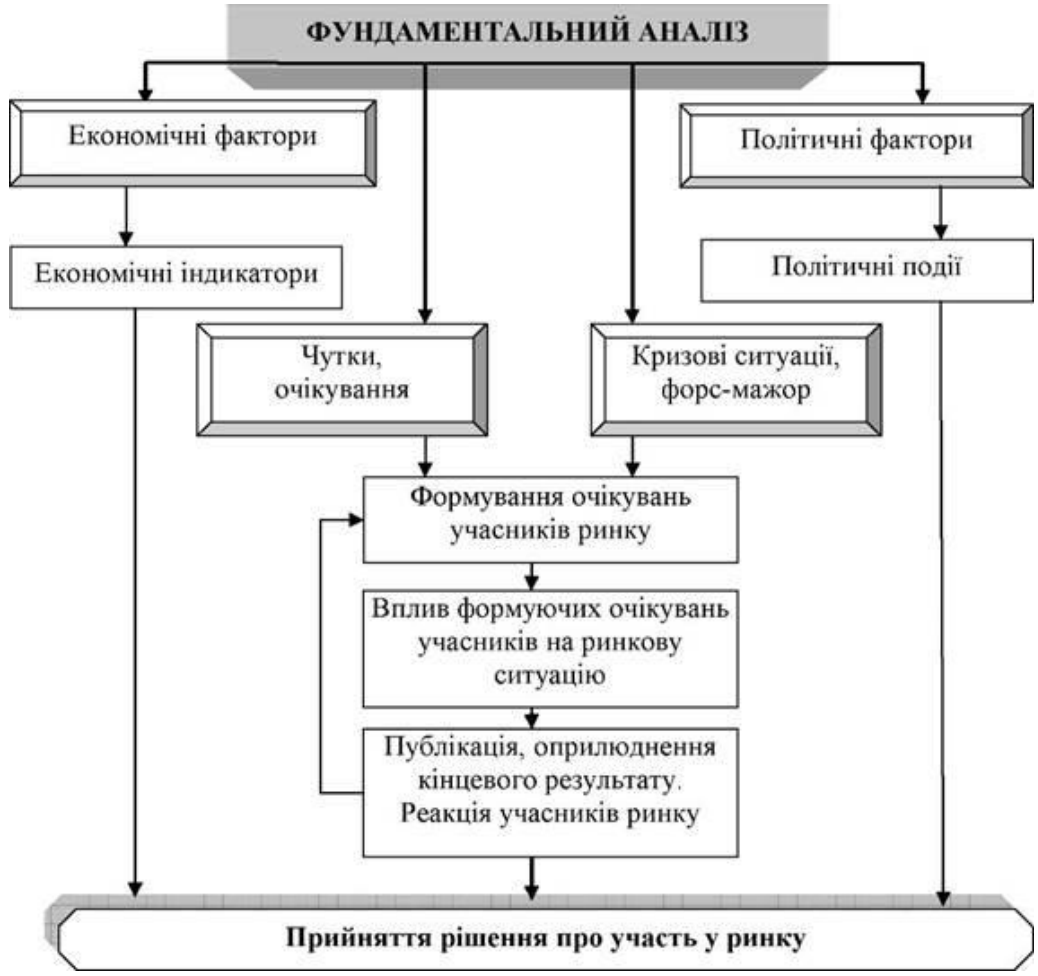


Рис. 1. Фундаментальний аналіз.

Переваги:

- Є ваговим фактором для аналізу глобальною ситуації
- Даною інформацією складно спекулювати, тобто ризик дезінформації мінімальний
- Абсолютна доступність інформації, і має велику різноманітність щодо ресурсів подання інформації

Недоліки:

- Неможливість алгоритмізувати аналіз
- Потребує велику кількість часу, і знань для успішного аналізу
- Присутній людський фактор
- Інформація довго оновлюється

Тепер розглянемо принцип технічного аналізу, корисна інформація для такого аналізу міститься в інформації часових рядів ціни, об'ємів торгівлі, та іншої статистичної інформації. Технічний аналіз базується на принципі обробки статистики минулого, щоб прорахувати майбутні зміни. Тобто, це повністю алгоритмізований аналіз, який базується на алгоритмах математичної статистики та теорії ймовірності, аналізуючи модель розподілу на основі статистичних даних та прорахування ризику відхилення від усередненого за розподілом значення. Завдяки технічному аналізу ми можемо реалізувати автоматизовані алгоритму аналізу ринку, цей тип усуває людський фактор, оскілки, він працює повністю на математичних розрахунках. Вся інформація, яка оброблюється цим типом аналізу, є корисною інформацією і стандартизованою, і, зазвичай, відображаються в чисельному вигляді. Тобто повне протиріччя фундаментальному аналізу. В даному випадку, цей тип аналізу складний для людини і потребує великої кількості часу, математичних розрахунків, але простий у випадку коли цей процес алгоритмізований і виконується системою, тобто ідеально підходить для нашого випадку.

Приклад технічного аналізу можна побачити на рисунку 2, в даному випадку такий тип технічного аналізу є ручним. На рисунку наведенні прикладі інструментів технічного аналізу, тобто, ресурсом є не тільки данні змінення ціни, але й вже розраховані данні інструментами аналізу (індикаторами), прикладом є лінії середнього значення, які перетинають бари ціни по-центру. Ці лінії відображають середню ціну за певну кількість барів. В подальшому, вся ця сукупність інформації, оброблюється алгоритмами прийняття рішення, та оцінки ризику [6].

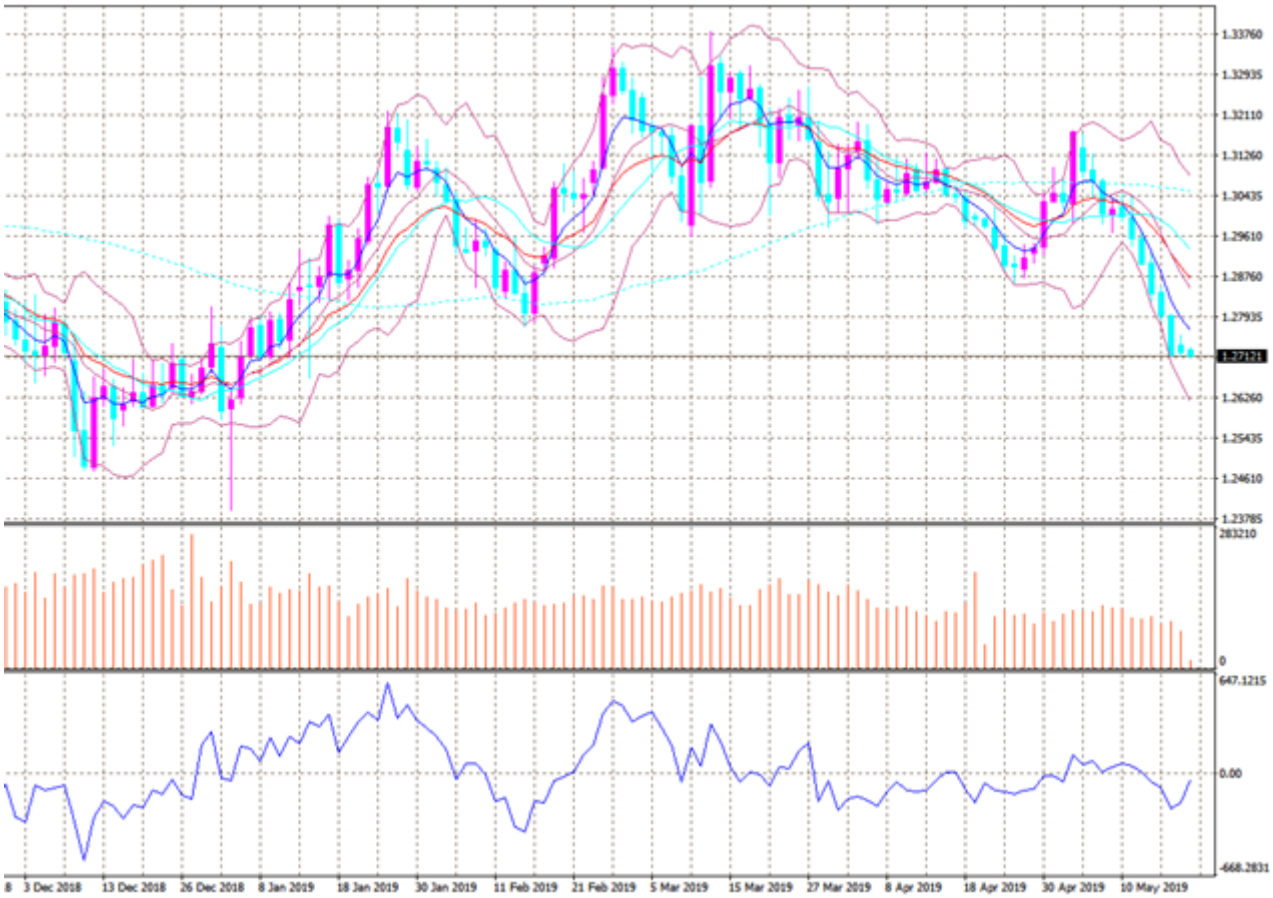


Рис. 2 Приклад технічного аналізу.

Переваги:

- Усунений людський фактор
- Працює тільки з корисною інформаціє, піддається математичному статистичному аналізу
- Легко алгоритмізується

Недоліки:

- Потребує неперервності аналізу, й великої вибірки інформації
- Точність прогнозу залежить від розподілу даних в певний період часу

У підсумку, можна з впевненістю сказати, що саме технічний аналіз підходить для реалізації теми дипломного проекту, проте, треба усунити

недоліки даного підходу або зменшити їх вплив на роботу алгоритму, якнайменше.

1.3. Обґрунтування теми дипломного проекту

На сьогоднішній час не існує системи, яка могла би могла успішно, та ефективно працювати, протягом великого часу, оскільки світ довкола змінюється, за ними йдуть зміни у всіх сферах діяльності і фінансова сфера не виключення. В середньому, алгоритми автоматичної торгівлі чи моніторингу ринку, ефективно працюють протягом 1-2 років, після чого вони потребують повного перепрограмування під сучасну ситуацію. В реалізації даного проекту планується розробити алгоритм, який буде менш піддатий до впливу зміни ситуації, тобто, як буде здатний з часом адаптуватися до нових умов ринкового середовища. Такий результат вдається отримати за рахунок усунення недоліків при обробці. Універсальність даного алгоритму полягає в тому, що він працює й розглядає усі данні чисто з математичної точки зору, не роблячи прогнозу, а керується внутрішньою границею ризику, тобто, ми прямо зможемо задавати межі ризику дозволені для прийняття рішення, також завдяки постійній зміні даних, буде змінюватися і математичні розрахунки, а саме ті, які є рішучими, це ймовірність ризику та ймовірність події, що в реальному часі дозволяє швидко приймати рішення по усуненню помилкового припущення (прийнятого рішення) таким чином мінімізується втрати по прийнятому рішенню. При розробці даної системи наглядно видно, що для успішної реалізації потрібно усунути недоліки на усіх етапах функціонування системи.

Недоліки обробки даних – оскільки, за основу ми беремо технічний аналіз, то перед нами постають два недоліка цього метода, які потрібно вирішити, щоб реалізувати ефективну роботу системи. Технічний аналіз потребує безперервного постачання нових даних і видалення застарілих, тобто, цю проблему ми можемо усунути чисто з програмної точки зору, реалізувавши

прями потік даних та безперервну обробку їх інструментами аналізу. Для проблеми розподілу даних реалізуємо інструменти обробки даних, побудовані на алгоритмах нормального розподілу, проектувавшись це на потік даних, ми можемо чітко виражати границі розподілу даних і робити більш чіткий прогноз по поточній ситуації, що прямо буде відображатися на алгоритмах прорахунку ризику. Таким методом можна знехтувати можливістю хаотичного розподілу даних, оскільки алгоритм буде мати чітку інформацію про таке явище і робити відповідні висновки, що прямо впливає на прийняття рішення.

Вирішивши проблему ефективного використання технічного аналізу, постає більш вагома проблема для ефективного функціонування системи, а саме її довговічність та мінімізації ризику помилкового прийняття рішення.

Оскільки, обробку даних ми залишаємо на наші інструменти технічного аналізу, то на відмінність від інших систем ми не потребуємо реалізовувати ці системи в самій програмі. Завдяки цьому, ми і вирішуємо проблему мінімізації ризику, оскільки, дані технічного аналізу, ми збираємо безперервно з усіх інструментів, а саме це дозволяє системи в будь який момент коректувати процес по прийняттю рішення, швидко реагувати на вже прийняті рішення та усунити їх як змога раніше.

Завдяки реалізації можливості зміни ключових границь, змінних, інструментів аналізу, тощо, ми можемо маніпулювати алгоритмом аналізу, що дозволяє підстроюватися під зміни властивостей ринку. В результаті ми отримуємо гнучкість системи моніторингу та довговічність в плані морального старіння алгоритму. Даний проект дозволяє в повній мірі реалізувати усі здобуті знання на протязі навчання в університеті, в цьому проекті гармонічно поєднані знання математичних, технічних та інших прикладних наук. В результаті це дозволяє знайти та створити інноваційні реалізації суттєвих проблем.

2. ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ. РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ.

2.1. Створення інструментів технічного аналізу

Основною рушійною силою нашої системи є якісний технічний аналіз, який представляє усю корисну інформацію логічним алгоритмам. Для технічного аналізу, зазвичай, використовують індикатори, які виконують розрахунки поточних даних ринку.

Першим етапом розробки системи є створення саме інструментів технічного аналізу, оскільки, правильна обробка даних є фундаментальною основою даної системи. На отриманій інформації технічного аналізу базується послідовність та правила поведінки наступних алгоритмів.

Першим ми розробимо індикатор «Force Index», який буде прораховувати відносний об’єм та індекс рушійної сили ринку, тобто, він порівнює швидкість росту і швидкість падіння ціни у вибраному проміжку часу. За допомогою цього індикатора можна робить прогноз на рахунок рушійної сили(тренду) та робити похибку для алгоритмів базуючи на законі нормального розподілу [6].

Більшість індикаторів лише оброблюють чи перетворюють вхідні дані у більш інформативний режим, тобто, вони побудовані на простих математичних функціях та структурі індикатора. Індикатори розрізняють за структурою, деякі з них реалізують математичні алгоритми, які не вбудовані в MQL4 Розберемо приклад структури на основі індикатора «Force Index»:

Ініціалізація класу індикатора «Force Index»

```
class CiForce : public CIndicator{
protected:
  int               m_ma_period; - таймфрейм на якому працює індикатор.
  ENUM_MA_METHOD    m_ma_method;
  ENUM_APPLIED_PRICE m_applied;
public:
                    CiForce(void);
```

```
                ~CiForce(void);
```
Метод доступу до захищених даних
```
  int           MaPeriod(void)       const { return(m_ma_period); }
  ENUM_MA_METHOD  MaMethod(void)       const { return(m_ma_method); }
  ENUM_APPLIED_PRICE Applied(void)       const { return(m_applied);   }
```
Метод ініціалізації
```
  bool          Create(const string symbol,const ENUM_TIMEFRAMES period,
                       const int ma_period,const ENUM_MA_METHOD ma_method,const
ENUM_APPLIED_PRICE applied);
```
Методи доступу до даних індикатора
```
  virtual double  GetData(const int buffer_num,const int index) const;
  double          Main(const int index) const;
```
Метод ідентифікації
```
  virtual int     Type(void) const { return(IND_FORCE); }
protected:
```
Інтерфейс налаштування
```
  virtual bool        Initialize(const string symbol,const ENUM_TIMEFRAMES period,const int
num_params,const MqlParam &params[]);
  bool          Initialize(const string symbol,const ENUM_TIMEFRAMES period,
                       const int ma_period,const ENUM_MA_METHOD ma_method,const
ENUM_APPLIED_PRICE applied);
;
```
Конструктор
```
CiForce::CiForce(void) : m_ma_period(-1),
                 m_ma_method(-1),
                 m_applied(-1)
  {  }
```
Деструктор
```
CiForce::~CiForce(void)
  {  }
```

Метод створення

```
bool CiForce::Create(const string symbol,const ENUM_TIMEFRAMES period,
                     const int ma_period,const ENUM_MA_METHOD ma_method,const
ENUM_APPLIED_PRICE applied)
  {
   SetSymbolPeriod(symbol,period);
   return(Initialize(symbol,period,ma_period,ma_method,applied));
  }
```

Ініціалізація з універсальними параметрами

```
bool CiForce::Initialize(const string symbol,const ENUM_TIMEFRAMES period,const int
num_params,const MqlParam &params[])
  {
return(Initialize(symbol,period,(int)params[0].integer_value,(ENUM_MA_METHOD)params[1].i
nteger_value,
        (ENUM_APPLIED_PRICE)params[2].integer_value));  }
```

Ініціалізація з унікальними параметрами

```
bool CiForce::Initialize(const string symbol,const ENUM_TIMEFRAMES period,
                         const int ma_period,const ENUM_MA_METHOD ma_method,const
ENUM_APPLIED_PRICE applied)
  {
```

Зберігаємо дані

```
   m_name="Force";
   m_ma_period=ma_period;
   m_ma_method=ma_method;
   m_applied  =applied;
   return(true);
  }
```

Доступ до буферу зовнішнього

```
double CiForce::GetData(const int buffer_num,const int index) const
  {
   return(CiForce(m_symbol,m_period,m_ma_period,m_ma_method,m_applied,index));
  }
```

Доступ до буферу внутрішнього

```
double CiForce::Main(const int index) const
 {
  return(GetData(0,index));  }
```

Moving averages – індикатор який відображає середнє значення ціни за даний період за певним розміром вибірки даних. Зазвичай, сам індикатор дублюється декілька разів з використанням різних функцій розрахунку середнього значення до різного розміру вибірки. В цілому, за показниками усіх індикаторів такого типу, можна робити припущення про зміну довгострокової тенденції росту чи падіння ціни. З математичної точки зору можна відображати як зміну розподілу значень від одного діапазону в інший, тобто до якої границі рухається вибірка середніх значень, до нижньої, так і до верхньої.

Цей індикатор розширює функціонал MQL4 та, окрім, структури створює нові алгоритми розрахунку. Оскільки, для аналізу потрібні різні види розрахунку середнього значення, зручніше це буде втілити в індикатор, щоб отримувати дані вже в результативному форматі [1].

Розробка індикатора, і функцій розрахунку:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_color1 Red
```

Параметри індикатора, налаштування.

```
input int          InpMAPeriod=13;   - діапазон вибірки барів ціни.
input int          InpMAShift=0; - відхилення індикатора
input ENUM_MA_METHOD InpMAMethod=MODE_SMA;
double ExtLineBuffer[]; - буфер
```

Функція створення індикатора, за заданими параметрами, та ринкової інформації. Ця функція в залежності від налаштування, та інших заданих змінних, будує об’єкт лінію на екрані ринкової інформації, яка відповідає обрахованим значенням заданим методом.

```
int OnInit(void)
  {
   string short_name;
   int    draw_begin=InpMAPeriod-1;
   switch(InpMAMethod)
     {
      case MODE_SMA  : short_name="SMA(";              break;
      case MODE_EMA  : short_name="EMA(";  draw_begin=0; break;
      case MODE_SMMA : short_name="SMMA(";             break;
      case MODE_LWMA : short_name="LWMA(";             break;
      default :        return(INIT_FAILED);
     }
   IndicatorShortName(short_name+string(InpMAPeriod)+")");
   IndicatorDigits(Digits);
   if(InpMAPeriod<2)
     return(INIT_FAILED);
   SetIndexStyle(0,DRAW_LINE);
   SetIndexShift(0,InpMAShift);
   SetIndexDrawBegin(0,draw_begin);
   SetIndexBuffer(0,ExtLineBuffer);
   return(INIT_SUCCEEDED);
  }
```

Функція форматування даних, для подальшого розрахунку. Розбиває часовий ряд, на окрему інформацію, робить перевірку на валідність даних, викликає функцію розрахунку певним методом, повертає результат.

```
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   if(rates_total<InpMAPeriod-1 || InpMAPeriod<2)
     return(0);
   ArraySetAsSeries(ExtLineBuffer,false);
   ArraySetAsSeries(close,false);
   if(prev_calculated==0)
     ArrayInitialize(ExtLineBuffer,0);
   switch(InpMAMethod)
```

```
   {
    case MODE_EMA:  CalculateEMA(rates_total,prev_calculated,close);        break;
    case MODE_LWMA: CalculateLWMA(rates_total,prev_calculated,close);       break;
    case MODE_SMMA: CalculateSmoothedMA(rates_total,prev_calculated,close); break;
    case MODE_SMA:  CalculateSimpleMA(rates_total,prev_calculated,close);   break;
   }
  return(rates_total);
 }
```

Ковзне середнє значення – є різновидом математичної згортки, використовується в технічному аналізі, для аналізу довгострокового руху ціни. В даному випадку вхідним значенням є часові ряди ціни, що дає видиму аргументацію технічного аналізу з математичної точки зору [8]. Обчислюється за формулою (1.1).

$$SMA_t = \frac{1}{n}\sum_{i=0}^{n-1} p_{t-i} \qquad (1.1)$$

, де p_t – значення з чисельного ряду ціни, SMA_t – значення простого ковзного середнього, n – кількість значень.

```
void CalculateSimpleMA(int rates_total,int prev_calculated,const double &price[])
 {
  int i,limit;
  if(prev_calculated==0)

    {
     limit=InpMAPeriod;
                    double firstValue=0;
     for(i=0; i<limit; i++)
       firstValue+=price[i];
     firstValue/=InpMAPeriod;
     ExtLineBuffer[limit-1]=firstValue;
    }
  else
     limit=prev_calculated-1;
  for(i=limit; i<rates_total && !IsStopped(); i++)
     ExtLineBuffer[i]=ExtLineBuffer[i-1]+(price[i]-price[i-InpMAPeriod])/InpMAPeriod;

 }
```

Експоненціальне зважене середнє значення - різновид зваженої ковзної середньої, ваги якої зменшуються експоненціальною і ніколи не дорівнюють нулю. Обчислюється за формулою (1.2).

$$EMA_t = \alpha \cdot p_t + (1 - \alpha) \cdot EMA_{t-1} \quad (1.2)$$

, де EMA_t- значення експоненціально зваженого середнього, EMA_{t-1}- попередне знячення функції EMA_t, p_t – значення з чисельного ряду ціни, α - коефіцієнт що характеризує швидкість зменшення ваг, приймає значення від 0 до 1, чим менше його значення тим більше вплив попередніх значень на поточну величину середнього.

```
void CalculateEMA(int rates_total,int prev_calculated,const double &price[])
{
 int    i,limit;
 double SmoothFactor=2.0/(1.0+InpMAPeriod);
 if(prev_calculated==0)
   {
    limit=InpMAPeriod;
    ExtLineBuffer[0]=price[0];
    for(i=1; i<limit; i++)
      ExtLineBuffer[i]=price[i]*SmoothFactor+ExtLineBuffer[i-1]*(1.0-SmoothFactor);
   }
 else
    limit=prev_calculated-1;
 for(i=limit; i<rates_total && !IsStopped(); i++)
    ExtLineBuffer[i]=price[i]*SmoothFactor+ExtLineBuffer[i-1]*(1.0-SmoothFactor);
}
```

Лінійно зважене середнє значення - ковзне середнє, при обчисленні якого вага кожного члена вихідної функції, починаючи з меншого, дорівнює відповідному члену арифметичній прогресії [2]. При обчисленні WMA для тимчасового ряду, ми вважаємо останні значення вихідної функції більш значущі ніж попередні, причому функція значущості лінійно спадна. Обчислюється за формулою (1.3).

$$WMA_t = \frac{2}{n \cdot (n+1)} \sum_{i=0}^{n-1} (n-i) \cdot p_{t-i} \quad (1.3)$$

, де WMA_t- значення лінійно зваженого середнього, n – кількість значень, p_t – значення з чисельного ряду ціни.

```
void CalculateLWMA(int rates_total,int prev_calculated,const double &price[])
 {
  int       i,limit;
  static int weightsum;
  double    sum;
  if(prev_calculated==0)
    {
     weightsum=0;
     limit=InpMAPeriod;
     double firstValue=0;
     for(i=0;i<limit;i++)
       {
        int k=i+1;
        weightsum+=k;
        firstValue+=k*price[i];
       }
     firstValue/=(double)weightsum;
     ExtLineBuffer[limit-1]=firstValue;
    }
  else
     limit=prev_calculated-1;
  for(i=limit; i<rates_total && !IsStopped(); i++)
    {
     sum=0;
     for(int j=0;j<InpMAPeriod;j++)
        sum+=(InpMAPeriod-j)*price[i-j];
     ExtLineBuffer[i]=sum/weightsum;
    }
 }
```

Згладжена зважена середня – середня в якому останнім значенням присвоюється більшу вагу, а більш раннім - меншу. Обчислюється за формулою (1.4).

$$\mathrm{MMA}_t = \frac{p_t + (n-1)\cdot \mathrm{MMA}_{t-1}}{n} \quad (1.4)$$

, де MMA_t- значення згладженої середньої, p_t – значення з чисельного ряду ціни, n – кількість значень.

```
void CalculateSmoothedMA(int rates_total,int prev_calculated,const double &price[])
 {
  int i,limit;
  if(prev_calculated==0)
    {
     limit=InpMAPeriod;
     double firstValue=0;
     for(i=0; i<limit; i++)
       firstValue+=price[i];
     firstValue/=InpMAPeriod;
     ExtLineBuffer[limit-1]=firstValue;
    }
  else
     limit=prev_calculated-1;
  for(i=limit; i<rates_total && !IsStopped(); i++)
     ExtLineBuffer[i]=(ExtLineBuffer[i-1]*(InpMAPeriod-1)+price[i])/InpMAPeriod;
 }
```

CiBands – індикатор, який при обробці даних, вираховує середнє значення за період, та будує дві лінії верхньої і нижньої границі, що відображає межі інтервалу нормального розподілу (-2 σ, 2 σ), тобто випадкова величина потрапляє в ці межі з ймовірність 95.4% коли випадкова, величина виходить за ці межі, можна проаналізувати ситуацію на ринку, яка не підкорюється нормальному розподілу, а тобто є спровокованою умовами або подією які впливають на ринок, що дозволяє більш доцільно досліджувати поточну ситуацію в реальному часі. Сам індикатор робить розрахунки на вбудованій функції відхилення значення, якому передається значення інтервалу, а за основу бере середнє значення поточного бару ціни.

Побудова індикатора кардинально відрізняється від попередніх, тим що потребує реалізації, до 4х буферів, тобто до самого середнього

значення(середньої лінії), по якій вираховується границі відхилення, нижня та верхня межа границі, та значення ціни бару. Реалізується наступним чином:

Ініціалізація классу CiBands

```
class CiBands : public CIndicator {
protected:
  int           m_ma_period; - значення періоду
  int           m_ma_shift; - значення зміщення індикатора
  double          m_deviation; - значення відхилення
  int           m_applied;
public:
                CiBands(void);
               ~CiBands(void);
```

Доступ до значення ціни

```
double CiBands::GetData(const int buffer_num,const int index) const  {
return(iBands(m_symbol,m_period,m_ma_period,(int)m_deviation,m_ma_shift,m_applied,buffer
_num,index));
 }
```

Доступ основи(середньої лінії).

```
double CiBands::Base(const int index) const
 {
  return(GetData(MODE_MAIN,index));
 }
```

Доступ до верьхнього буферу

```
double CiBands::Upper(const int index) const
 {
  return(GetData(MODE_UPPER,index));
 }
```

Доступ до нижнього буферу

```
double CiBands::Lower(const int index) const
 {
  return(GetData(MODE_LOWER,index));
 }
```

Розроблені індикатори проектуються на поверхні екрану з ринковими даними, що дозволяє, також, візуально сприймати інформацію, в результаті індикатори не є лише структурою автоматизованої системи, він теж дозволяє вручну робити технічний аналіз, при необхідності коректувати дії системи

вручну. Тобто, дані інструменти є корисними як при ручній торгівлі, так і при автоматичній, оскільки, дані індикаторів можна отримувати напряму з буферу.

Маючи основу, яка постачає дані ринку та робить безперервну функціональну обробку даних, що є основою технічного аналізу, можна переходити до наступного пункту розробки, а саме створення структури системи та реалізації алгоритмів прогнозу, прийняття рішення та оцінки ризику.

2.2. Створення структури системи.

Загальна структура вбирає в себе наступні функції:

- OnInit – виконує функцію ініціалізації на систему Metatrader 4, враховуючи таймфрейм та валютну пару на яку проектують систему.
- OnDeinit – видаляє систему, з платформи.
- OnTick – Оброблює потік даних, містить функції кінцевого прийняття рішення.
- Stop – функція ліквідування негативних рішень, та закриття позиції.
- Traill – функція відкладеного закриття позиції, вираховує час, та умови при яких треба закрити позицію, може змінювати час в залежності від ситуації, приймає рішення по середньо строковим позиціям.
- MoveToBreakEven – функція оцінки ризику, прораховує ризик, підлаштовує безпечний об’єм для позиції.
- Scalper – функція короткочасного прогнозу, та прийняття рішення.
- LotsOptimized – оптимізує об’єм для відкриття позиції.
- getOpenOrders – функція відкриття позиції
- OrderModifyCheck – функція оновлення, та модифікації позиції
- Exitsells – функція закриття позиції на продаж

- Exitbuys – функція закриття позиції на покупку

Інтерфейс початкових змінних створюється окремо, та містить наступні зміні:

- USEMOVETOBREAKEVEN – типу Boolean, відповідає за флаг агресивної чи безпечної торгівлі, напряму залежить на дію алгоритму ризику.
- WHENTOMOVETOBE – змінна примусового закриття прибуткової позиції через певну кількість числових рядів
- PIPSTOMOVESL – аналогічно WHENTOMOVETOBE, але працює в випадку с збиткової позиції
- Lots – змінна, об’єму позиції
- MaximumRisk – змінна допустимого ризику
- TrailingStop – змінна задає значення за замовчуванням для функції Traill.
- Stop_Loss – змінна, яка визначає максимальний допустимий збиток по позиції, якщо він перевищує позиція автоматично закривається.
- TakeProfit – аналогічно Stop_Loss, автоматично закриває позицію після отримання певного прибутку

Інші змінні є службовими, для отримання інформації з індикаторів та платформи Metatrader4. Усі функції в MQL4 розбиваються на функції які повертають значення в зазначену зміну або функцію яка оголошується зміною, тобто розрахункова функція, в процесі якої оголошена зміна приймає значення роботи функції. Аналогічно присвоюванню змінній результату роботи функції, але в першому випадку, така функція не потребує оновлення даних, і завжди перераховує результат коли, відбувається робота з такою зміною. Оголошення інтерфейсу налаштування системи, має наступну структуру, яка включає усі вище перечисленні змінні:

```
#property strict

extern bool        USEMOVETOBREAKEVEN=true
extern double      WHENTOMOVETOBE=10
extern double      PIPSTOMOVESL=5;
extern double      Lots=0.01;
input  double      MaximumRisk   =0.02;
input  double      DecreaseFactor=3;
extern double      TrailingStop=40;
extern double      Stop_Loss=20;
input  double      TakeProfit=50;
int          err;
bool             FractalsUp=false;
bool             FractalsDown=false;
double            FractalsUpPrice=0;
double            FractalsDownPrice=0;
int              FractalsLimit=200;
int total=0;
double
Lot,Dmax,Dmin,
Lts,
Min_Lot,
Step,
Free,
One_Lot,
Price,
pips,
MA_1,MA_2,MA_3,MACD_SIGNAL;
int Type,freeze_level,Spread;
double priceopen,stoploss,takeprofit;
int orderticket;
int
Period_MA_2,  Period_MA_3,
Period_MA_02, Period_MA_03,
K2,K3,T;
```

Даний фрагмент коду відповідає лише за файл налаштування вхідних даних. Завдяки ньому ми налаштовуємо систему, і можемо її корегувати в будь який момент часу, що дозволяє досягти гнучкості системи, та адаптуватися під нові економічні тенденції.

2.3. Створення алгоритму прийняття рішення.

Під алгоритмом прийняття рішення – мається на увазі, алгоритми здатний приймати рішення про відкриття позиції на покупку або продаж, опираючись на технічний аналіз, та за можливістю зробити це без перевищення допустимого ризику. Тобто це кінцевий логічний алгоритм, котрий робить заключення, отримавши усі дані, з усіх інструментів, алгоритмів системи. Процесом роботи алгоритму є так звана стратегія ринкової торгівлі, це правила поведінки, які потрібно робити в залежності від тих, чи інших обставин. Алгоритми закладені в основі є достатньо примітивними, і простими в реалізації, але не дуже ефективними, тому ми використовую допоміжні алгоритму аналізу ризику, та закриття позицій, щоб цей недуг виправити, в результаті отримуємо більш ефективну систему, ймовірність успішних позицій в якій вище, ніж в виконані цієї ж стратегії, без цих допоміжних алгоритмів. В алгоритмі об’єднуються три стратегії торгівлі, а саме:

- Для коротко строкових позицій – відкривається позиція за аналізом швидкиї таймфреймів, 1MIN - 5MIN
- Середньо строкових – відкриває позиції за аналізом середніх таймфреймів 15MIN - 4H
- Довгострокові(інвестиційні) – це аналіз, довгострокових таймфреймів, 1D - 1Y

Особливість різних таймфреймів відрізняється часом, за який відмалюється один чисельний ряд, тобто для таймфрейму 1MIN – це одна хвилина, а для 1D – це один день. Тобто це розподіл інформації ринку, за об’ємом інформації в чисельному ряді. Для різних таймфреймів використовуються різні підходи технічного аналізу. Та в залежності від таймфрейму також залежить час існування позиції, тобто для кожного

таймфрейму вона не повинна перевищувати ніж 5-10 чисельних рядів. Для 1MIN – це 5-10 хвилин, для 1D – це 5-10 днів.

Для більшої ефективності, ми працюємо одразу по за трьома типами таймфреймів, з різними стратегіями. Швидкі, ми використовуємо як основні позиції, середні для продовження позиції, або для покриття збитків, а довгострокові, як інвестиційні, якщо маємо чітку інформацію і можливість для відкриття даної позиції з мінімальним ризиком.

Тепер детально розберемо логіку роботи кожного з алгоритмів, для кожної з стратегій. Розглянемо алгоритму в послідовності збільшення таймфреймів. Алгоритми опрацьовування швидких позицій називають Scalper, вони є найбільш ефективними оскільки, здатні відкривати за робочу сесію багато позицій, та в сумі видають результат кращий, оскільки працюють навіть коли середньо строкові, та довгострокові позиції відкривати не вигідно. В нашому проекті цей алгоритм реалізує функція Scalper, тепер більш детально розберемо алгоритм роботи даної функції.

Вимоги виконання роботи алгоритму:

- даний алгоритм виконується при малих об’ємах, значення яких ми отримуємо з індикатора Force Index
- при виконанні закону нормального розподілу, які ми отримуємо з індикатора CiBands

Алгоритм працює лише коли усі вимоги присутні, та алгоритм оцінки ризику, дозволяє відкрити позицію. Правило за яким алгоритм відкриває позицію дуже просте, якщо виконується закон нормального розподілу, та об’єми ринкової торгівлі мають не великі обсяги, тобто не можуть швидко вплинути на ринкову ціну, то відбувається розрахунок різниці верхньої та нижньої границі відхилення, якщо зміни ціни від однієї границі до іншої вистачає щоб отримати прибуток, то при максимальному приближені ціни до однієї з границі, відкривається позиція в сторону від руху ціни. В результаті в

умовах виконання подальшого закону нормального розподілу, позиція закривається в прибуток біля межі протилежної границі.

Реалізація алгоритму:

```
int scalper()
  {
   for(int i=3; i<20;i++)
     {

      double  EMAH8=iMA(NULL,0,8,T,MODE_LWMA,PRICE_TYPICAL,0);
      double  EMAH20=iMA(NULL,0,20,T,MODE_LWMA,PRICE_TYPICAL,0);
      double  EMA21 =iMA(NULL,0,21,0,MODE_LWMA,PRICE_TYPICAL,i);
      double  EMA13 =iMA(NULL,0,13,0,MODE_LWMA,PRICE_TYPICAL,i);
      double  EMA8=iMA(NULL,0,8,0,MODE_LWMA,PRICE_TYPICAL,i);

      //if(EMAH8>EMAH20)
      if(EMA8>EMA13 && EMA13>EMA21)
        if(((Low[i]<EMA8 || Low[i]==EMA8) && Low[i]>EMA21) &&
Ask>iHighest(NULL,0,MODE_HIGH,5,0))
          {
           return(1);
           //TREND=1;
           Comment("The trend is up");
          }

      if(EMA8<EMA13 && EMA13<EMA21)
        if(((High[i]>EMA8 || High[i]==EMA8) && High[i]<EMA21) &&
Ask<iLowest(NULL,0,MODE_LOW,5,0))
          {
           return(2);
           //TREND=2;
           Comment("The trend is down");
          }
     }
    return(0);
   }
```

Після прийняття рішення, алгоритм виконує функцію OnTick, яка відкриває позицію, та відповідає за подальшу роботу за цією позицією. Роздивляючись взаємодію системи, функція Scalper, лише передає інформацію, про можливість відкриття позиції, та не може власноруч, відкривати позиції, цю роботу виконують інші функції.

Розглянемо алгоритм середньо строкової тривалості, дані позиції називають середньо денною торгівлею, це зв’язано з тим що більшість таких

позицій відкривається по середині робочої сесії, коли ліквідність ринку, та об’єми є найвищими. Ефективність даних алгоритмів дуже залежить від сили руху ціни, та її стабільності.

Вимоги виконання роботи алгоритму:

- Висока ліквідність, та об’єми ринку
- Чітке та стійке заломлення коефіцієнту сили в одну із напрямків, значення якого ми отримуємо з індикатора Force index
- Дані отримані з Moving Average по довгостроковим таймфреймах не суперечать, з індикатором Force Index
- Дані отримані з Moving Average по середнім таймфреймах задовольняє оцінку ризику

При виконані усіх вище перерахованих вимог, можна робити заключення, що ринкова ціна, з великою рушійною силою, та стабільною закономірністю, рухається в одному з напрямків. Отримуючи значення з індикатора Force Index, в залежності позитивного або від’ємного значення коефіцієнту сили, можна робити висновок, про напрямок руху, для підтвердження стабільності з цього же індикатора, робиться аналіз даних, за певний період часу, для підтвердження цієї тенденції якийсь проміжок часу. Даними ковзкого середнього значення отриманого за довгостроковими таймфреймом, можна робити прогноз на довгостроковий потенціал даної тенденції, якщо він не суперечить поточній ситуації, ризик даної позиції зменшується. Отримані дані за середнім ковзким по середньо строковим таймфреймах ми аналізуємо, для оцінки поточної ціни порівнюючи її з найближчою середньою, щоб оцінювати потенціал прибутку, та поточну цінову ситуацію, після цього ці, дані також оброблюються алгоритмом оцінки ризику, якщо співвідношення ризику до потенційного прибутку мінімум співвідноситься як 1 до 5, то ця інформація відправляється на функцію OnTick, в результаті чого передається інформація про заключення ризику, за цими результатами відкриваються позиція, з змінними STOP_LOSS та TAKE_PROFIT, в відповідності до співвідношення ризику до прибутку, та

позиція автоматично закривається при досягненні ціни зазначених у одній із цих змінних.

Також слід зазначити що, якщо функція MoveToBreakEven активна, то вона автоматично буде змінювати значення STOP_LOSS, після отримання певного прибутку за позицією, щоб мінімізувати ризик закриття позиції в збиток. Алгоритм середньо строкових позицій прораховує функція Traill, ця функція також виконує аналіз про необхідність модифікації позиції, що дозволяє швидкі позиції переводити у середньо строкові.

Реалізація функції:

```
void Trail1()
  {
  total=OrdersTotal();
  for(int cnt=0;cnt<total;cnt++)
    {
     if(!OrderSelect(cnt,SELECT_BY_POS,MODE_TRADES))
       continue;
     if(OrderType()<=OP_SELL &&   // check for opened position
       OrderSymbol()==Symbol())  // check for symbol
      {
       //--- long position is opened
       if(OrderType()==OP_BUY)
         {
          if(TrailingStop>0)
            {
             if(Bid-OrderOpenPrice()>pips*TrailingStop)
               {
                if(OrderStopLoss()<Bid-pips*TrailingStop)
                  {
                   RefreshRates();
                   stoploss=Bid-(pips*TrailingStop);
                   takeprofit=OrderTakeProfit()+pips*TrailingStop;

double
StopLevel=MarketInfo(Symbol(),MODE_STOPLEVEL)+MarketInfo(Symbol(),MODE_SPREA
D);
                   if(stoploss<StopLevel*pips) stoploss=StopLevel*pips;
                   string symbol=OrderSymbol();
                   double point=SymbolInfoDouble(symbol,SYMBOL_POINT);
                   if(MathAbs(OrderStopLoss()-stoploss)>point)

     if((pips*TrailingStop)>(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_FREEZE_L
     EVEL)*pips)
```

```
                    //--- modify order and exit
                    if(CheckStopLoss_Takeprofit(OP_BUY,stoploss,takeprofit))

if(OrderModifyCheck(OrderTicket(),OrderOpenPrice(),stoploss,takeprofit))

if(!OrderModify(OrderTicket(),OrderOpenPrice(),stoploss,takeprofit,0,Green))
                        Print("OrderModify error ",GetLastError());
                return;
                }
              }
            }
          }
        else // go to short position
          {
           if(TrailingStop>0)
             {
              if((OrderOpenPrice()-Ask)>(pips*TrailingStop))
                {
                 if((OrderStopLoss()>(Ask+pips*TrailingStop)) || (OrderStopLoss()==0))
                   {
                    RefreshRates();
                    stoploss=Ask+(pips*TrailingStop);
                    takeprofit=OrderTakeProfit()-pips*TrailingStop;
                    double
StopLevel=MarketInfo(Symbol(),MODE_STOPLEVEL)+MarketInfo(Symbol(),MODE_S
PREAD);
                    if(stoploss<StopLevel*pips) stoploss=StopLevel*pips;
                    if(takeprofit<StopLevel*pips) takeprofit=StopLevel*pips;
                    string symbol=OrderSymbol();
                    double point=SymbolInfoDouble(symbol,SYMBOL_POINT);
                    if(MathAbs(OrderStopLoss()-stoploss)>point)

if((pips*TrailingStop)>(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_FREEZE_L
EVEL)*pips)
                    //--- modify order and exit
                    if(CheckStopLoss_Takeprofit(OP_SELL,stoploss,takeprofit))

if(OrderModifyCheck(OrderTicket(),OrderOpenPrice(),stoploss,takeprofit))


if(!OrderModify(OrderTicket(),OrderOpenPrice(),stoploss,takeprofit,0,Red))
                        Print("OrderModify error ",GetLastError());
                return;
                }
              }
            }
          }
```

Розглянемо алгоритм довгострокової інвестиційного прийняття рішення.

Даний алгоритм не є ефективним для автоматизованої системи, та має деякі

обмеженні можливості в плані технічного аналізу даного явища. Довгострокові тенденції задають умови ринкової економіки, політичні явища, тощо, тобто довгострокові позиції краще піддаються фундаментальному аналізу. Не дивлячи на це, технічний аналіз також може прослідковувати такі тенденції, їх можна помітити аналізуючи значення отримані з індикатора Moving average, обробляючи дані з довгострокових функцій одночасно різних типів обрахунків середнього ковзного значення.

Вимоги виконання алгоритму:

- Одночасний чітко виражений напрямок тенденції відслідковуваний з за значеннями усіх середніх ковзних довгострокових таймфреймів отриманих з індикаторів типу Moving Average.
- Співвідношення оцінки ризику 1 до 100 прибутку
- Довгостроковий стабільний коефіцієнт сили в одному напрямку, проаналізований на довгостроковому таймфреймі

Якщо усі вимоги підтверджуються, то можна зробити висновок, що даний приклад є довгостроковою тенденцією, та може розглядатися як можливість для довгострокової позиції. Алгоритми оцінки ризику, дозволяються виконати такі позиції лише при наявності великого запасу маржі, та лише мінімально допустим об’ємом для позиції. Даний алгоритм обраховується безпосередньо в функції OnTick, яка є керуючою системою кінцевого прийняття рішення, та обробки інформації оцінки ризику. Робота даної функції є безпосередньо обробка отриманих прогнозів з інших функцій прийняття рішень, після чого, вона викликає функцію оцінки ризику, якщо вона у відповідь отримує значення ризику близьке до співвідношення ризик до прибутку, отриманого за алгоритмів прийняття рішень, то викликає процес відкриття позиції за переданими параметрами. В кінцевому результаті, усі відкриті позиції безперервно оброблюються алгоритмом оцінки ризику, та в разі чого модифікує або закриває вже відкрить позиції.

Реалізація функції OnTick:

```
void OnTick(void)
  {
   if(USEMOVETOBREAKEVEN) MOVETOBREAKEVEN();
   Trail1();
   int    ticket;
   static datetime dtBarCurrent=WRONG_VALUE;
   datetime dtBarPrevious=dtBarCurrent;
   dtBarCurrent=(datetime) SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
   bool NewBarFlag=(dtBarCurrent!=dtBarPrevious);
   if(Bars<100)
     {
      Print("bars less than 100");
      return;
     }
   if(TakeProfit<10)
     {
      Print("TakeProfit less than 10");
      return;
     }
   double middleBB=iBands(Symbol(),0,20, 2,0,0,MODE_MAIN,1);//middle
   double lowerBB=iBands(Symbol(),0,20, 2,0,0,MODE_LOWER,1);//lower
   double upperBB=iBands(Symbol(),0,20, 2,0,0,MODE_UPPER,1);//upper
   double  MacdMAIN=iMACD(NULL,0,12,26,9,PRICE_CLOSE,MODE_MAIN,1);
   double  MacdSIGNAL=iMACD(NULL,0,12,26,9,PRICE_CLOSE,MODE_SIGNAL,1);
   double  EMAH8=iMA(NULL,0,8,T,MODE_LWMA,PRICE_TYPICAL,0); // MA_1
   double  EMAH20=iMA(NULL,0,20,T,MODE_LWMA,PRICE_TYPICAL,0); // MA_2
   double  EMA21 =iMA(NULL,0,21,0,MODE_LWMA,PRICE_TYPICAL,1); // MA_1
   double  EMA13 =iMA(NULL,0,13,0,MODE_LWMA,PRICE_TYPICAL,1); // MA_2
   double  EMA8=iMA(NULL,0,8,0,MODE_LWMA,PRICE_TYPICAL,1); // MA_1
   double  Stoc1  =iStochastic(NULL,0,5,3,3,MODE_SMA,0,MODE_MAIN,1); //Stochastic
   double  Stoc2  =iStochastic(NULL,0,5,3,3,MODE_SMA,0,MODE_MAIN,2); //Stochastic
   double  SAR=iSAR(NULL,0,0.02,0.2,0);

   if(getOpenOrders()==0)

     {
      if(AccountFreeMargin()<(1000*Lots))
        {
         Print("We have no money. Free Margin = ",AccountFreeMargin());
         return;
        }
      if(NewBarFlag)
        {
         //--- check for long position (BUY) possibility
         //if(Volume[0]>1) return;{
         if(Stoc2<20 && Stoc1>=20)
           // if(SAR<Open[1])
               //if(scalper()==1)
```

```
            {
            ticket=OrderSend(Symbol(),OP_BUY,LotsOptimized(),ND(Ask),3,NDTP(Bid-
Stop_Loss*pips),NDTP(Bid+TakeProfit*pips),"renko",MagicNumber,0,PaleGreen);
            if(ticket>0)
              {
               if(OrderSelect(ticket,SELECT_BY_TICKET,MODE_TRADES))
                  Print("BUY order opened : ",OrderOpenPrice());
              }
            else
              Print("Error opening BUY order : ",GetLastError());
            return;
            }
        //--- check for long position (SELL) possibility
        if(Stoc2>80 && Stoc1<=80)
          // if(SAR>Open[1])
            //if(scalper()==2)
            {

ticket=OrderSend(Symbol(),OP_SELL,LotsOptimized(),ND(Bid),3,NDTP(Ask+Stop_Loss*pips),
NDTP(Ask-TakeProfit*pips),"Short 1",MagicNumber,0,Red);
            if(ticket>0)
              {
               if(OrderSelect(ticket,SELECT_BY_TICKET,MODE_TRADES))
                  Print("SELL order opened : ",OrderOpenPrice());
              }
            else
              Print("Error opening SELL order : ",GetLastError());
            }
           return;
        }
     }
  }
void stop()
  {
   int cnt;
   double middleBB=iBands(Symbol(),0,20, 2,0,0,MODE_MAIN,1);//middle
   double lowerBB=iBands(Symbol(),0,20, 2,0,0,MODE_LOWER,1);//lower
   double upperBB=iBands(Symbol(),0,20, 2,0,0,MODE_UPPER,1);//upper
   double  MacdMAIN=iMACD(NULL,0,12,26,9,PRICE_CLOSE,MODE_MAIN,1);
   double  MacdSIGNAL=iMACD(NULL,0,12,26,9,PRICE_CLOSE,MODE_SIGNAL,1);
   for(cnt=0;cnt<total;cnt++)
     {
      if(!OrderSelect(cnt,SELECT_BY_POS,MODE_TRADES))
         continue;
      if(OrderType()<=OP_SELL &&   // check for opened position
         OrderSymbol()==Symbol())  // check for symbol
        {
         //--- long position is opened
         if(OrderType()==OP_BUY)
           {
```

```
        //--- should it be closed?

        if(((Bid<OrderOpenPrice()-iATR(NULL,0,14,0))
          || ((MacdMAIN>0 && MacdMAIN<MacdSIGNAL) || (MacdMAIN<0 &&
(MathAbs(MacdMAIN)>MathAbs(MacdSIGNAL))))
          || Close[1]==upperBB))
          exitbuys();
              }
      else // go to short position
        {
        if(((Ask>OrderOpenPrice()+iATR(NULL,0,14,0))
          || ((MacdMAIN>0 && MacdMAIN>MacdSIGNAL) || (MacdMAIN<0 &&
(MathAbs(MacdMAIN)<MathAbs(MacdSIGNAL)))))
          || Close[1]==lowerBB)
          exitsells();

        }
      }
    }
  }
```

Данна функцію має внутрішні методи, для відкриття та закриття позицій, проте випадки вимушеного закриття позиції, в разі зміни оцінки ризику, прилягає на роботу алгоритмів закриття позицій. Також, є зовнішня функція модифікації позицій, яка безпосередньо та виведення інформації по відкритим позиціям, які безпосередньо працюють з алгоритмом оцінки ризику.

2.4. Створення алгоритму розрахунку ризику.

Задачі які виконує даний алгоритм, це розрахунок безпечного об’єму для певної позиції, в залежності від її таймфрейму, та співвідношення ризику до прибутку, ці дані алгоритм безпосередньо отримує з алгоритмів прийняття рішення. Та задача постійного аналізу поточних позицій, даний алгоритм аналізує їх послідовно безперервним потоком, від найстарішої позиції до найновішої, в ході аналізу перераховує співвідношення ризику до прибутку, та на основі цього модифікує позицію змінюючи значення змінних Take_profit та Stop_loss, що дозволяє мінімізувати ризик, за збільшити прибуток, якщо початковий прогноз був не достатньо точним.

Прорахунок безпечного об’єму розраховується з розрахунком правил математично доцільного забезпечення підтримки рівня вільної маржі, що є не більшим 10% від загального об’єму ресурсів. Прогнозує найближчі можливі позиції, та при необхідності ставить їх в чергу, або примусово закриває вже прибуткові позиції, щоб мати необхідний запас маржі, для витримки просадки за найгіршим прогнозом по всім позиціям. Реалізація функції розрахунку безпечного об’єму :

```
double LotsOptimized()
  {
   double lot=Lots;
   int    orders=OrdersHistoryTotal();
   int    losses=0;
```

Вибір оптимального об’єму з оцінкою співвідношення ризик до прибутку

```
   if(MaximumRisk>0)
     {
      lot=NormalizeDouble(AccountFreeMargin()*MaximumRisk/1000.0,1);
     }
   if(DecreaseFactor>0)
     {
      for(int i=orders-1;i>=0;i--)
        {
         if(OrderSelect(i,SELECT_BY_POS,MODE_HISTORY)==false)
           {
            Print("Error in history!");
            break;
           }
         if(OrderSymbol()!=Symbol() /*|| OrderType()>OP_SELL*/)
           continue;
         if(OrderProfit()>0) break;
         if(OrderProfit()<0) losses++;
        }
      if(losses>1)
        lot=NormalizeDouble(lot-lot*losses/DecreaseFactor,1);
     }
```

Прорахунок мінімального об’єму

```
   double minlot=SymbolInfoDouble(Symbol(),SYMBOL_VOLUME_MIN);
   if(lot<minlot)
     { lot=minlot; }
  Print("Volume is less than the minimal allowed ,we use",minlot);}
lot=minlot;
```

Прорахунок максимально допустимого обєму

```
  double maxlot=SymbolInfoDouble(Symbol(),SYMBOL_VOLUME_MAX);
  if(lot>maxlot)
    { lot=maxlot;  }
 Print("Volume is greater than the maximal allowed,we use",maxlot);}
lot=maxlot;
  double volume_step=SymbolInfoDouble(Symbol(),SYMBOL_VOLUME_STEP);
  int ratio=(int)MathRound(lot/volume_step);
  if(MathAbs(ratio*volume_step-lot)>0.0000001)
    {  lot=ratio*volume_step;}
  return(lot);
/* else  Print("StopOut level  Not enough money for ",OP_SELL," ",lot," ",Symbol());
  return(0);*/
  }
```

Даний алгоритм має внутрішню функцію для отримання інформації по усім відкритим позиціями, що повертає позицію за її номером.

```
int getOpenOrders()
 {   int Orders=0;
  for(int i=0; i<OrdersTotal(); i++)
    {
     if(OrderSelect(i,SELECT_BY_POS,MODE_TRADES)==false)
       {        continue;       }
   if(OrderSymbol()!=Symbol() || OrderMagicNumber()!=MagicNumber)
       {        continue;       }
    Orders++;
    }
  return(Orders);
 }
```

Функція модифікації позиції перша частина робить перевірку, чи змінилися умови за позицією, та перевіряє поточну ситуацію прибутковості позиції та її тривалість. Саме ця функція працює при необхідності звільнення маржі для наступної позиції, виконуючи пошук, позиції яка вже має високий прибуток, або морально застарілу позицію. Реалізація функції OrderModifyCheck:

```
bool OrderModifyCheck(int ticket,double price,double sl,double tp)
 {
  if(OrderSelect(ticket,SELECT_BY_TICKET))
    {
     string symbol=OrderSymbol();
     double point=SymbolInfoDouble(symbol,SYMBOL_POINT);
     bool PriceOpenChanged=true;
```

```
   int type=OrderType();
   if(!(type==OP_BUY || type==OP_SELL))
     {
      PriceOpenChanged=(MathAbs(OrderOpenPrice()-price)>point);
     }
   bool StopLossChanged=(MathAbs(OrderStopLoss()-sl)>point);
   bool TakeProfitChanged=(MathAbs(OrderTakeProfit()-tp)>point);
   if(PriceOpenChanged || StopLossChanged || TakeProfitChanged)
     return(true);  // order can be modified
   else
     PrintFormat("Order #%d already has levels of Open=%.5f SL=%.5f TP=%.5f",
           ticket,OrderOpenPrice(),OrderStopLoss(),OrderTakeProfit());
  }
 return(false);       // no point in modifying
}
```

Функція CheckStopLoss_Takeprofit перевірки змінних - працює майже аналогічно але перевіряє значення змінних Take_profit і Stop_loss, на актуальність поточній ситуації, в викликає функцію обрахування нового аналізу співвідношення ризику до прибутку, та в разі необхідності, оновлює дані змінних. Реалізація функції :

```
bool CheckStopLoss_Takeprofit(ENUM_ORDER_TYPE type,double SL,double TP)
 {
```

Повертає значення Stop_Loss

```
  int stops_level=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
  if(stops_level!=0)
    {
     PrintFormat("SYMBOL_TRADE_STOPS_LEVEL=%d: StopLoss and TakeProfit must"+
           " not be nearer than %d points from the closing price",stops_level,stops_level);
    }
  bool SL_check=false,TP_check=false;
  switch(type)
    {
```

Перевіряє коефіцієнт ризику для позиції на покупку

```
     case  ORDER_TYPE_BUY:
       {
       Перевіряє stop loss
       SL_check=(Bid-SL>stops_level*_Point);
       if(!SL_check)
         PrintFormat("For order %s StopLoss=%.5f must be less than %.5f"+
               " (Bid=%.5f - SYMBOL_TRADE_STOPS_LEVEL=%d points)",
               EnumToString(type),SL,Bid-stops_level*_Point,Bid,stops_level);
```

Перевіряє значення TakeProfit

```
    TP_check=(TP-Bid>stops_level*_Point);
    if(!TP_check)
      PrintFormat("For order %s TakeProfit=%.5f must be greater than %.5f"+
            " (Bid=%.5f + SYMBOL_TRADE_STOPS_LEVEL=%d points)",
            EnumToString(type),TP,Bid+stops_level*_Point,Bid,stops_level);
    return(SL_check&&TP_check);
   }
```

Перевіряє коефіцієнт ризику для позиції на покупку

```
  case  ORDER_TYPE_SELL:
   {
    SL_check=(SL-Ask>stops_level*_Point);
    if(!SL_check)
      PrintFormat("For order %s StopLoss=%.5f must be greater than %.5f "+
            " (Ask=%.5f + SYMBOL_TRADE_STOPS_LEVEL=%d points)",
            EnumToString(type),SL,Ask+stops_level*_Point,Ask,stops_level);
    TP_check=(Ask-TP>stops_level*_Point);
    if(!TP_check)
      PrintFormat("For order %s TakeProfit=%.5f must be less than %.5f "+
            " (Ask=%.5f - SYMBOL_TRADE_STOPS_LEVEL=%d points)",
            EnumToString(type),TP,Ask-stops_level*_Point,Ask,stops_level);
        return(TP_check&&SL_check);
   }
  break;
 }
return false;
}
```

2.5. Створення алгоритму закриття позиції.

Задачею даного алгоритму є знаходження і примусове закриття позицій котрі є збитковими, та зайшли в критичну помітки співвідношення ризику до прибутку, провокується алгоритмом оцінкою ризику, також дана функція може бути оброблена при необхідності звільніти маржу для відкриття нової позиції. Примусово закриває позицію за заданим номером. Реалізація функції:

```
    void exitbuys()
     {
      double result;
      for(int i=OrdersTotal()-1; i>=0; i--)
        {
         if(OrderSelect(i,SELECT_BY_POS,MODE_TRADES))
           {
            if(OrderType()==OP_BUY && OrderSymbol()==Symbol() &&
OrderMagicNumber()==MagicNumber)
```

```
                {
                 result=OrderClose(OrderTicket(),OrderLots(),OrderClosePrice(),3,clrNONE);
                 if(result!=true)//if it did not close
                   {
                    err=GetLastError(); Print("LastError = ",err);//get the reason why it didn't close
                   }
                }
              }
             }
        void exitsells()
          {
           double result;
           for(int i=OrdersTotal()-1; i>=0; i--)
             {
              if(OrderSelect(i,SELECT_BY_POS,MODE_TRADES))
                {
                 if(OrderType()==OP_SELL && OrderSymbol()==Symbol() &&
OrderMagicNumber()==MagicNumber)
                 {
                  result=OrderClose(OrderTicket(),OrderLots(),OrderClosePrice(),3,clrNONE);
                  if(result!=true)//if it did not close
                    {
                     err=GetLastError(); Print("LastError = ",err);//get the reason why it didn't close
                    }
                 }
                }
          }
```

3. ТЕСТУВАННЯ, ОЦІНКА НАДІЙНОСТІ.

3.1. Тестування функцій прийняття рішень.

На основі платформи Metatrader 4, проведемо тестування, по історії чисельних значення, для переконання в працездатності системи. Тестування працює наступним чином, використовуючи історію ринкової інформації, процес тестування запускає нашу систему оброблювати ці дані, для нашої системи це є реальні дані, і вони опрацьовуються як і в реальних умовах.

Задачею тестування функцій прийняття рішення, є перевірка правильного виконання алгоритмів коротких, середніх, так довгострокових таймфреймів. Спочатку знайдемо на графіку, початок відкриття та закриття позицій, та впевнимося в тому, що робота даного алгоритму є коректною.

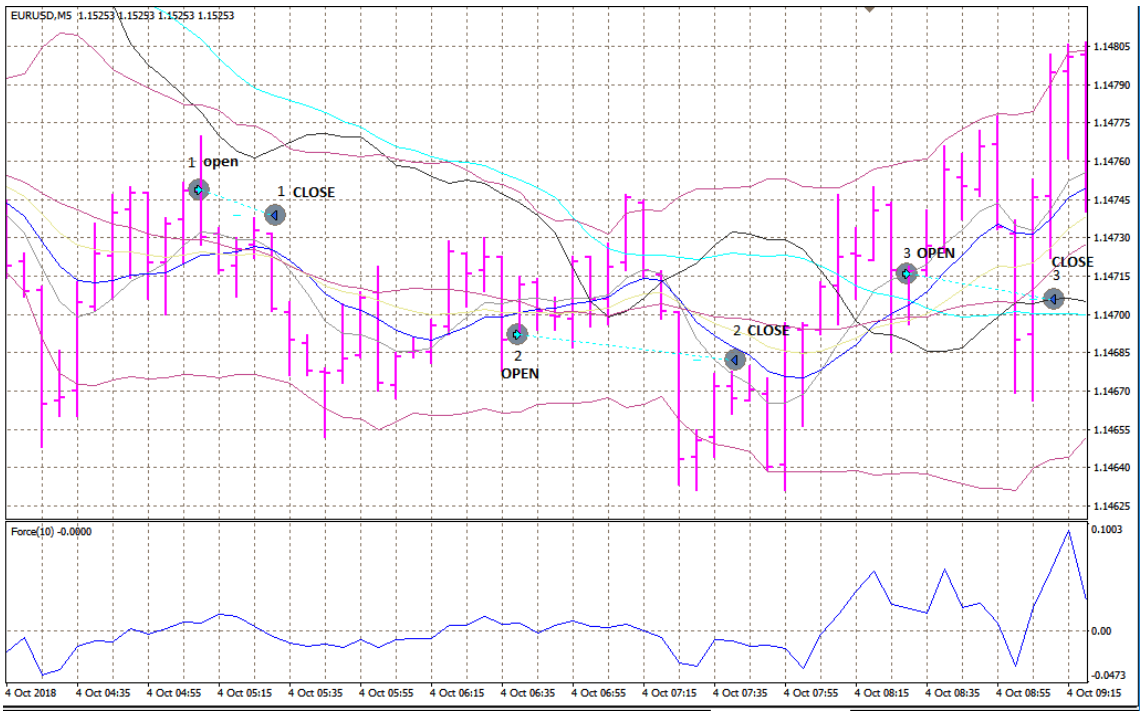


Рис 3. Тестування алгоритму швидких таймфреймів.

З графіку видно, що позиції відкриваються та закриваються згідно алгоритму прийняття рішення для швидких таймфреймів. В даному випадку,

перша позиція закривається за змінення оцінки ризику, друга та третя позиція відповідно до наближення до границі відхилення індикатора CiBands.

Аналогічно протестуємо середньо строкові, та довгострокові таймфрейми. Тестування алгоритму середньо строкових таймфреймів, тестування буде проводитись на таймфреймі 15MIN.

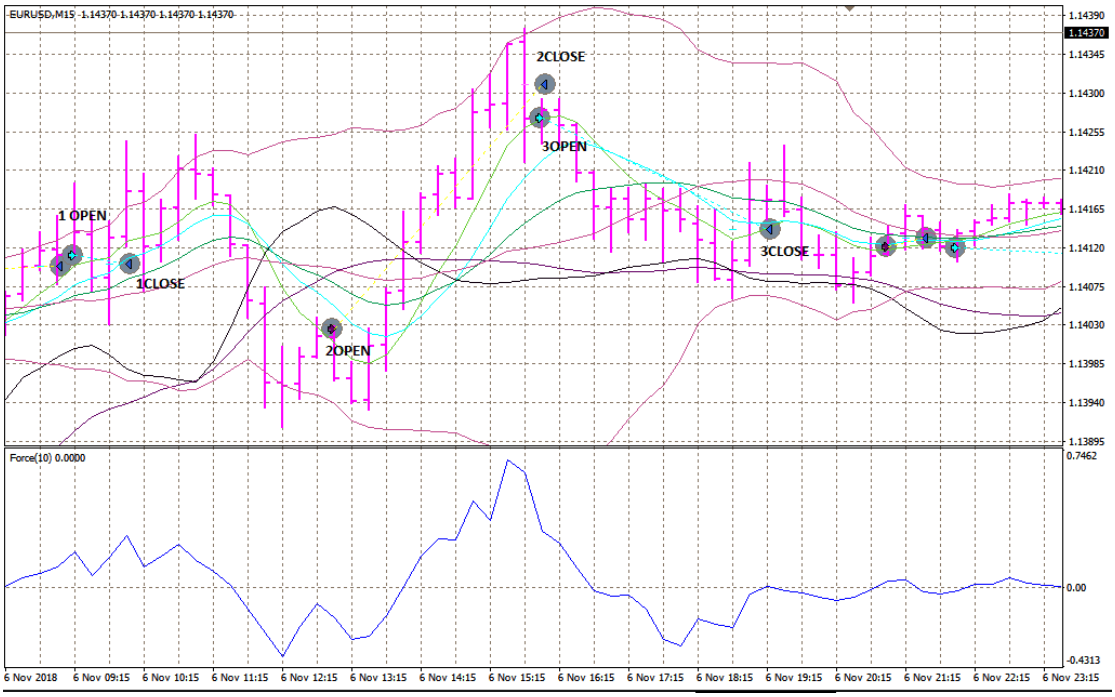


Рис. 4. Тестування алгоритму середніх таймфреймів.

З графіку добре видно що, при зміні параметру індикатора Force Index, відкривається позиція, та закривається при аналогічній дії. Добре проглядається робота алгоритму на другій та третій позиції, коли після різкого змінення значення коефіцієнту сили, та наближені до верхньої границі індикатора CiBands, закривається друга позиція, та відкривається третя.

Довгострокові таймфрейми будемо досліджувати на таймфреймі 1D (денних часових множин), оскільки вимоги для відкриття позиції для даного таймфрейму зустрічаються не часто. Головне дослідити тенденцію, правильність роботи алгоритму по опрацьовуванні довгострокових позицій та прослідкувати виконання усіх вимог відкриття даної позиції.

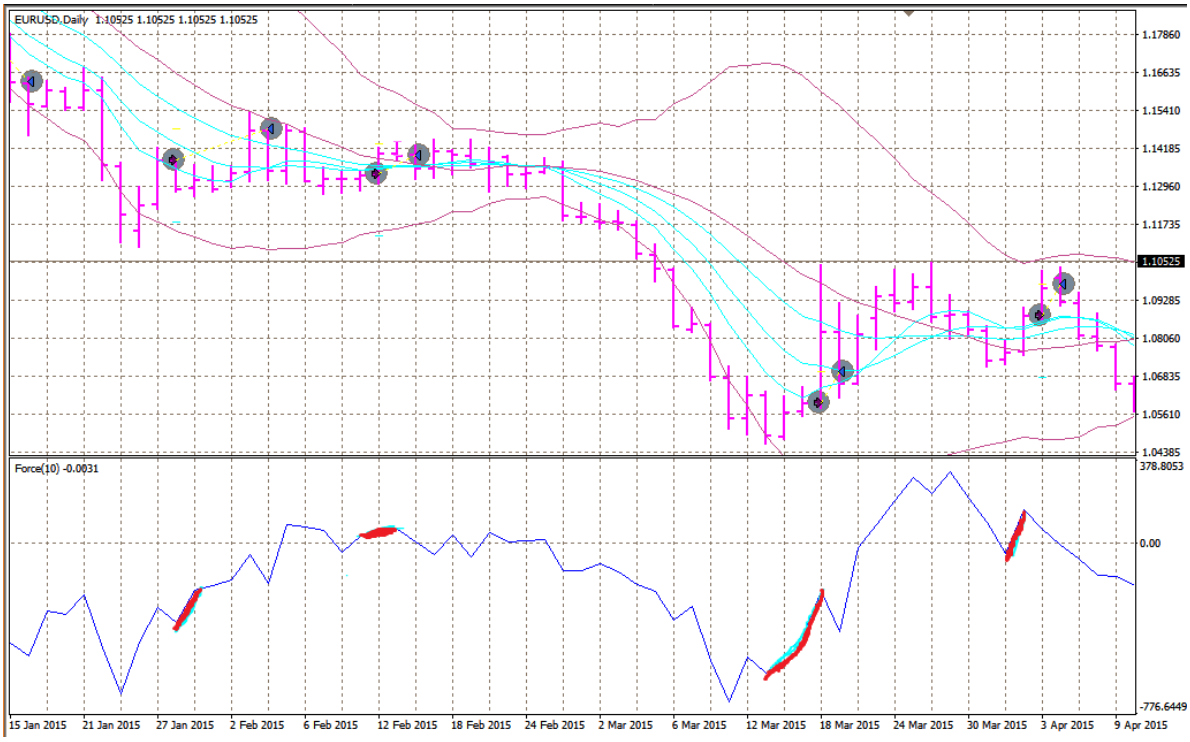


Рис. 5. Тестування алгоритму довгострокових таймфреймів.

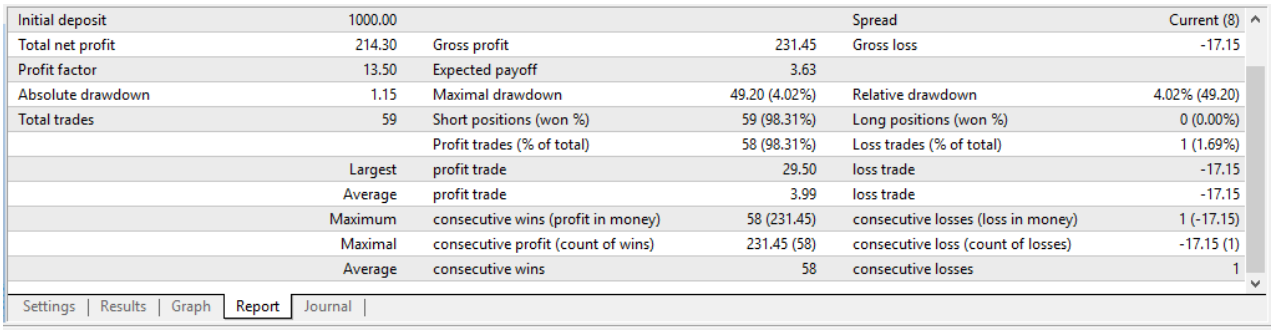
На графіку, видно позиції відкриття та закриття позиції, на індикаторі Force index, виділені червоним місця, які були сигналом для відкриття позицій, легко помітити що індикатор Moving Average, не суперечить індикатору Force index, а в момент коли лінії індикатора Movin Average перетинаються, позиція закривається, оскільки оцінка ризику різко збільшується, та не виконуються вимоги відкриття позиції, це є сигналом для примусового закриття позиції.

Тобто усі механізми даного алгоритму працюють правильно, але нажаль даний метод не є ефективним, це легко помітити за тим що дані вимоги відбуваються не часто, і в середньому кожна позиція була відкрита на протязі цілого місяці, що є доволі ризикованим, і не ефективним підходом.

3.2. Тестування функцій управління ризиками

Тестування ризику будемо проводити за аналізами прибутку, просадки, та збитку, тобто детального торгівельного звіту.

Порівняємо три алгоритми прийняття рішення, за попередньою оцінкою найбільш надійною повинен бути алгоритм швидких позиції, оскільки він працює заздалегідь в безпечних умовах, та ще коректується в процесі роботи, алгоритм середніх позицій, повинен не дуже гірше себе показувати, оскільки система ризику прямо впливає на його достроковість, та постійно оновлює дані змінних для закриття позиції, тобто він оброблюється та постійно адаптується від початку створення позиції, до її повно закриття. Найгірше себе має показати алгоритм довгострокових позицій, оскільки, він працює в умовах підвищеного ризику, та умови зміни довгострокової тенденції дуже складно точно відслідковувати без фундаментального аналізу.



Initial deposit	1000.00			Spread	Current (8)
Total net profit	214.30	Gross profit	231.45	Gross loss	-17.15
Profit factor	13.50	Expected payoff	3.63		
Absolute drawdown	1.15	Maximal drawdown	49.20 (4.02%)	Relative drawdown	4.02% (49.20)
Total trades	59	Short positions (won %)	59 (98.31%)	Long positions (won %)	0 (0.00%)
		Profit trades (% of total)	58 (98.31%)	Loss trades (% of total)	1 (1.69%)
	Largest	profit trade	29.50	loss trade	-17.15
	Average	profit trade	3.99	loss trade	-17.15
	Maximum	consecutive wins (profit in money)	58 (231.45)	consecutive losses (loss in money)	1 (-17.15)
	Maximal	consecutive profit (count of wins)	231.45 (58)	consecutive loss (count of losses)	-17.15 (1)
	Average	consecutive wins	58	consecutive losses	1

Settings | Results | Graph | Report | Journal |

Рис. 6. Звіт роботи алгоритму коротких позицій

- Сумарний прибуток = 231.45 пунктів
- Кількість прибуткових позицій = 98.31%
- Максимальна просадка = 4.02%
- Коефіцієнт ефективності =13.5
- Середня прибутковість за позицію = 3.99 пунктів.
- Кількість відкритих позицій 59
- Серед них збиткових позицій – 1

Initial deposit	1000.00			Spread	Current (9)
Total net profit	134.58	Gross profit	169.25	Gross loss	-34.67
Profit factor	4.88	Expected payoff	3.36		
Absolute drawdown	37.05	Maximal drawdown	73.71 (6.78%)	Relative drawdown	6.78% (73.71)
Total trades	40	Short positions (won %)	21 (100.00%)	Long positions (won %)	19 (89.47%)
		Profit trades (% of total)	38 (95.00%)	Loss trades (% of total)	2 (5.00%)
	Largest	profit trade	28.40	loss trade	-30.43
	Average	profit trade	4.45	loss trade	-17.34
	Maximum	consecutive wins (profit in money)	19 (85.85)	consecutive losses (loss in money)	1 (-30.43)
	Maximal	consecutive profit (count of wins)	85.85 (19)	consecutive loss (count of losses)	-30.43 (1)
	Average	consecutive wins	19	consecutive losses	1

Рис. 7. Звіт роботи алгоритму середніх позицій.

- Сумарний прибуток = 134.58 пунктів
- Кількість прибуткових позицій = 95%
- Максимальна просадка = 6.78%
- Коефіцієнт ефективності =4.88
- Середня прибутковість за позицію = 4.45 пунктів.
- Кількість відкритих позицій 40
- Серед них збиткових позицій – 2

Initial deposit	1000.00			Spread	Current (9)
Total net profit	581.50	Gross profit	2700.75	Gross loss	-2119.25
Profit factor	1.27	Expected payoff	10.03		
Absolute drawdown	204.83	Maximal drawdown	778.96 (49.46%)	Relative drawdown	49.46% (778.96)
Total trades	58	Short positions (won %)	0 (0.00%)	Long positions (won %)	58 (77.59%)
		Profit trades (% of total)	45 (77.59%)	Loss trades (% of total)	13 (22.41%)
	Largest	profit trade	97.79	loss trade	-409.66
	Average	profit trade	60.02	loss trade	-163.02
	Maximum	consecutive wins (profit in money)	7 (633.73)	consecutive losses (loss in money)	2 (-430.62)
	Maximal	consecutive profit (count of wins)	633.73 (7)	consecutive loss (count of losses)	-430.62 (2)
	Average	consecutive wins	4	consecutive losses	1

Рис.8. Звіт роботи алгоритму довгострокових позицій.

- Сумарний прибуток = 581.50 пунктів
- Кількість прибуткових позицій = 77.59%
- Максимальна просадка = 49.46%
- Коефіцієнт ефективності =1.27
- Середня прибутковість за позицію = 60.02 пунктів.
- Кількість відкритих позицій 58
- Серед них збиткових позицій – 13

Теоретичні припущення підкріплюються тестами, за даними звіту можна заключити, що найбільш надійним є алгоритм короткострокових позицій, та в порівняні з потенціальною прибутком, і найефективнішим. Менш надійним, але достатньо безпечним є алгоритм середніх позицій, але перевагою середніх позицій є більша середня прибутковість за позицію. Найгірше себе показав алгоритм довгострокових позицій, що було прогнозовано на етапі розробки, даний алгоритм, є дуже ефективним по одиночній позиції, оскільки він має найвищий прибуток за позицію, проте він потребує значно високий об'єм ресурсів, що може призупиняти роботу систему із-за нестачі вільної маржі, та й сам алгоритм є дуже ризиковим, затрачаючи на просадку більш ніж пів усього об'єму ресурсу, тобто якщо, просадка була би на 20-30 процентів вища, позиція була би закрита автоматично платформою Metatrader 4 із-за недостачі маржі для підтримки маржі, що спровокувало б колосальні збитки.

3.3. Тестування ефективності системи

Ефективність системи краще всього аналізувати за кривою функції прибутку, чим плавніше графік, тим він є більш безпечним, там ефективним. Крива по факту показує усі відкриті позиції, та стан об'єму ресурсу на протязі усього часу торгової сесії. Знову для порівняння розглянемо три різні алгоритми системи, для різного типу позицій.

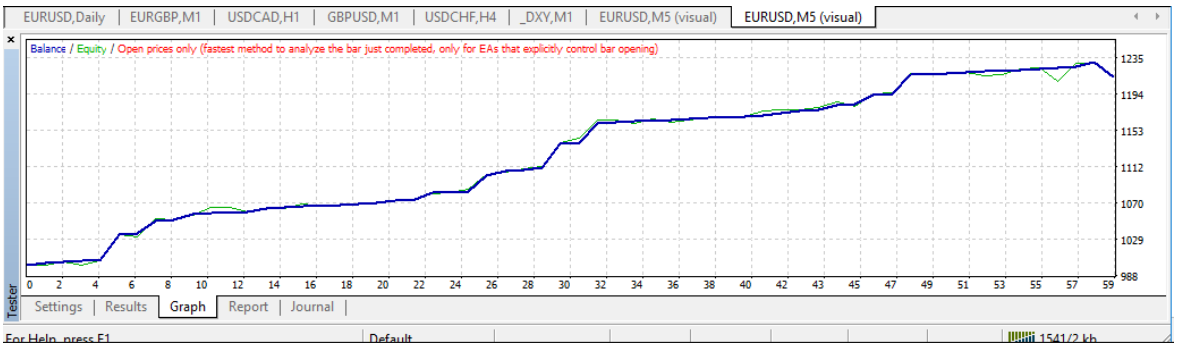


Рис. 9. Крива ефективності алгоритму коротких позицій.

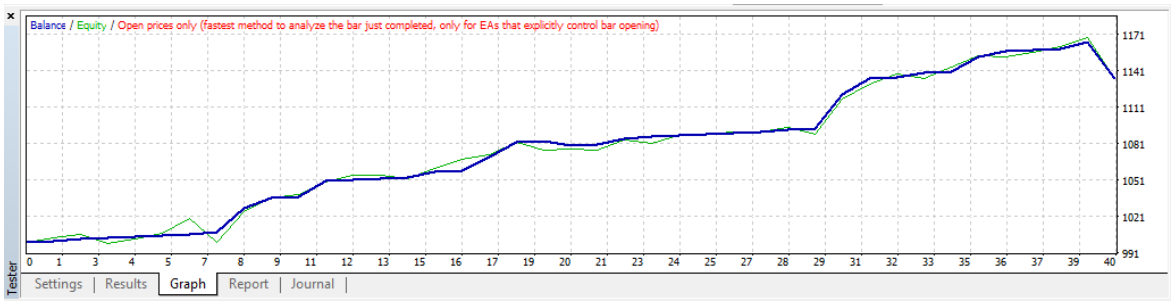


Рис. 10. Крива ефективності алгоритму середніх позицій.

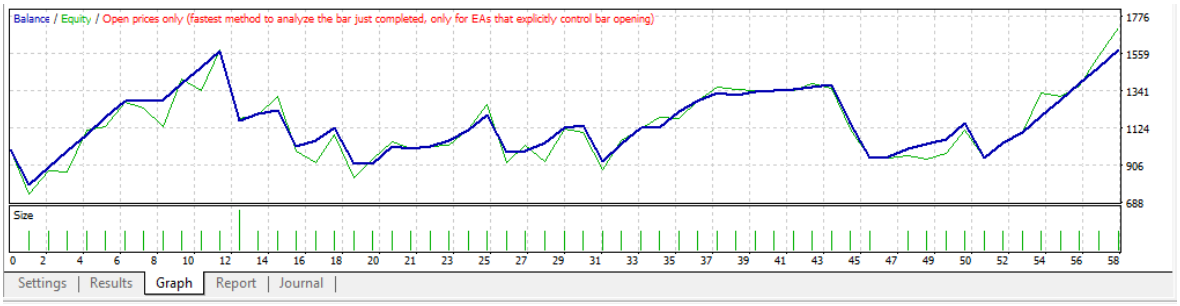


Рис. 11. Крива ефективності алгоритму довгострокових позицій.

Знову на результатах тесту підтверджуються теоретичні припущення, найбільш плавною є функція алгоритму короткострокових позицій, що відповідає дійсності, та показує добре співвідношення прибутку до ризику, наступним йде алгоритм середніх позицій, та найгірше показує себе алгоритм довгострокових позицій, крива виглядає дуже хаотичною та випадковою, що дає цілком оправдане припущення про високий ризик даного алгоритму, проте він має найбільший прибуток, але це не можна розглядати як ефективний алгоритм.

Отже підсумуємо, в цілому оглядаючись на результати прибутку, об’єднуючі дані за всіма трьома алгоритмами маємо загальний результат прибутковості 90.3%, та загальну максимальну просадку 17%, тобто співвідношення прибутку до ризику є 1 до 5, що дає можливість заключити що дана система є ефективною автоматизованою системою моніторингу ринку.

4. РЕЗУЛЬТАТ РОБОТИ СИСТЕМИ, ПОРІВНЯННЯ З РЕАЛЬНИМИ ДАНИМИ.

В даному розділи аналізуються та порівнюються робота системи в порівняні з ручною торгівлею людини, в реальних умовах ринку. Аналіз буде проводитися на базі офіціального звіту, за два тижні торгівлі.

Account: 60181718 **Name: Chuhaievskyi Viacheslav** **Currency: USD** **Leverage: 1:500** **2018 December 6, 20:22**

Open Trades:

Ticket	Open Time	Type	Size	Item	Price	S / L	T / P		Price	Commission	Taxes	Swap	Profit
No transactions													
										0.00	0.00	0.00	0.00
												Floating P/L:	**0.00**

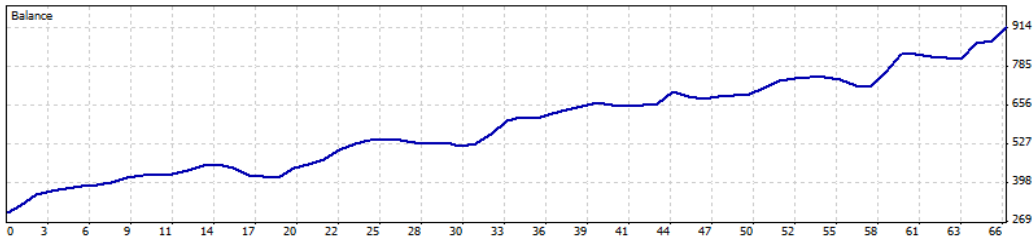
Working Orders:

Ticket	Open Time	Type	Size	Item	Price	S / L	T / P	Market Price
No transactions								

Summary:

Deposit/Withdrawal:	**300.00**	**Credit Facility:**	**0.00**		
Closed Trade P/L:	**620.04**	**Floating P/L:**	**0.00**	**Margin:**	**0.00**
Balance:	**920.04**	**Equity:**	**920.04**	**Free Margin:**	**920.04**

Details:



Gross Profit:	**782.61**	**Gross Loss:**	**162.57**	**Total Net Profit:**	**620.04**
Profit Factor:	**4.81**	**Expected Payoff:**	**9.39**		
Absolute Drawdown:	**0.00**	**Maximal Drawdown:**	**43.22 (9.41%)**	**Relative Drawdown:**	**9.41% (43.22)**
Total Trades:	**66**	**Short Positions (won %):**	**37 (67.57%)**	**Long Positions (won %):**	**29 (72.41%)**
		Profit Trades (% of total):	**46 (69.70%)**	**Loss trades (% of total):**	**20 (30.30%)**
Largest		**profit trade:**	**60.60**	**loss trade:**	**-23.20**
Average		**profit trade:**	**17.01**	**loss trade:**	**-8.13**
Maximum		**consecutive wins ($):**	**9 (126.01)**	**consecutive losses ($):**	**4 (-43.22)**
Maximal		**consecutive profit (count):**	**126.72 (8)**	**consecutive loss (count):**	**-43.22 (4)**
Average		**consecutive wins:**	**5**	**consecutive losses:**	**2**

Рис. 12. Звіт торгової сесії людини.

- Сумарний прибуток = 620.04 пунктів
- Кількість прибуткових позицій = 69.7%
- Максимальна просадка = 9.41%
- Коефіцієнт ефективності =4.81

- Середня прибутковість за позицію = 17.01 пунктів.
- Кількість відкритих позицій 66
- Серед них збиткових позицій – 20

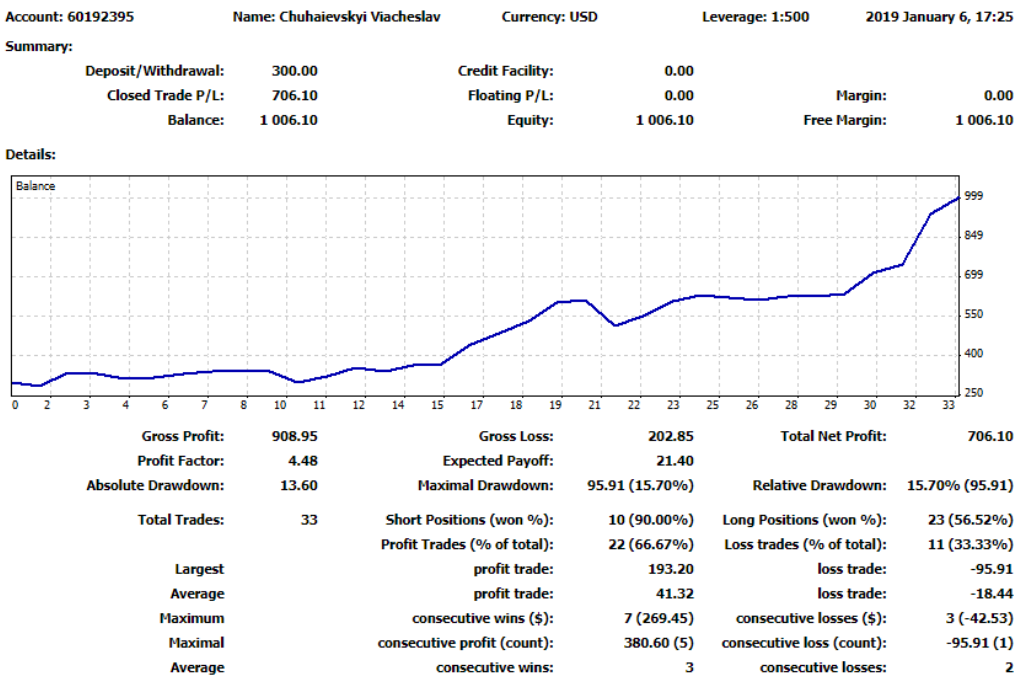


Рис. 13. Звіт торгової сесії автоматизованої інтелектуальної системи моніторингу ринку.

- Сумарний прибуток = 706.10 пунктів
- Кількість прибуткових позицій = 66.67%
- Максимальна просадка = 15.7%
- Коефіцієнт ефективності =4.48
- Середня прибутковість за позицію = 41.32 пунктів.
- Кількість відкритих позицій 33
- Серед них збиткових позицій – 11

Порівнюючи ці результати одразу видно слабкі та сильні сторони ручної та автоматичної торгівлі. Ручна торгівля більше стабільна що на коротких, що на довгострокових позиціях, при ручній торгівлі прибутковість позицій складає

близько 70%, та з мінімальним ризиком, максимальна просадка лише 9.41%, проте при ручній торгівлі, в процес включається людський фактор, що добре видно на результатах прибутковості короткочасних позицій лише в 70%, а також враховуючи час затрачений на одну торгову сесію в день це мінімум від 6 до 8 годин в день, безпосередньо тяжкої праці, це є великий недолік в порівняні з автоматизованою торгівлею.

Аналізуючи дані торгівлі автоматизованої системи, слід замітити, що короткочасні позиції маю 90% прибутковість, що ми і бачили на тестах. Проте зразу видно слабку сторону алгоритму, це довгострокові позиції де прибутковість встановила лише 56.52%, що прямо відзначилась на коефіцієнті максимальної просадки, яка встановила аж 15.7%, проте це теж дуже гарний результат. Коефіцієнт ефективності в обох випадка майже рівний, проте зазначимо те що системі знадобилося менше часу, та відкритих позицій щоб досягти такого результату.

Підсумуємо, в кожного з методів є свої недоліки, проте що на тестах, шо на реальних результатах, можна з впевненістю заявити, що дана автоматизована інтелектуальна система моніторингу ринку, нічим не уступає в даному випадку людині. Та система має значну перевагу, в тому що вона працює в автономному режимі, та може не здавати в ефективності на протязі необмеженого часу, коли людина вже після декількох годин, гірше починає справлятися з аналітичними функціями. За цими результатами можна зробити висновок, що даний проект повністю відповідає теоретичним, тестовим та реальним значенням, і задача по розробці системи виконана успішно.

ВИСНОВКИ

Отже, за час роботи над дипломним проектом була створена автоматизована інтелектуальна система моніторингу ринку. При створені даної системи були використані сучасні алгоритми технічного аналізу, обробки даних, та алгоритмів прийняття рішення спроектованих на платформі Metatrader 4 завдяки алгоритмічній мові MQL4.

Розроблена є ефективнішою за своїх аналогів, та в порівняні з людиною. Теоретичні дані, тестування та реальні дані збігаються в розрахунках, що інформує про велику точність та правильність роботи системи. Система ризику реалізує функцію адаптації системі під нові зміні умови ринку. Даний проект може зацікавити як студентів так і спеціалістів технічної області, дана система може стати фундаментальною основою для інших систем завдяки своєї гнучкості, та розбитої на алгоритми структури.

Ця система є втіленням усіх знань отриманих в процесі навчання в університеті, та добре відображає різноманітність та якість підготовки технічних спеціалістів. В цілому автоматизована інтелектуальна система виконана повністю з відповідністю до задачі дипломного проекту.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Программуваня на алгорітмічній мові MQL4 [Електронний ресурс], 2008. – Дата доступу: лютий 2008
2. Маталицький М.О. Теорій ймовірності та математична статистика/ Маталицький Михайло Олексійович, 2015. – 584 с.
3. Матвієнко М.П. Математична логіка та теорія алгоритмів/ Матвієнко Микола Павлович, 2015. – 212 с.
4. Атчісон Лі Масштабування додатків. Вирощування складних систем/ Атчісон Лі, 2017. – 256 с.
5. Мерфи Д. Дж. Міжринковий аналіз: принципи взаємодії фінансових ринків/ Мерфи Джон Джосеф, 2012. – 305 с.
6. Швагер Д. Технічний аналіз/ Швагер Джек, 2017. – 1022 с.
7. Живетін В.Б. Управління ризиками банківських систем/ Живетін Володимир Борисович, 2009. – 480 с.
8. Мартишін С.А. Основи теорії надійності інформаційних систем/ Мартишін Сергій Анатолійович, 2013. – 256 с.
9. Труханов В.М. Новий підхід до забезпечення надійності складних систем/ Труханов Володимир Михайлович, 2010. – 242 с.
10. Рижіков Ю.И. Імітаційне моделювання. Авторська імітація систем з чергами/ Рижіков Юрій Іванович , 2016. – 112 с.

ДОДАТКИ

Додаток 1 Структура взаємозв’язків інтелектуальної системи. Схема структурна.

Додаток 2. Алгоритм технічного аналізу. Схема алгоритму.

Додаток 3. Алгоритм прийняття рішення. Схема алгоритму.

Додаток 4. Алгоритм управління ризиком. Схема алгоритму.