Forecasting Stock Prices and Accounting for Stock Market on Multicore Computers

Lesia Mochurad1[0000-0002-4957-1512], Nataliya Boyko1[0000-0002-6962-9363],

Natalia Stanasiuk [0000-0002-6885-9431]

Lviv Polytechnic National University, Lviv79013, Ukraine
lesia.i.mochurad@lpnu.ua, nataliya.i.boyko@lpnu.ua,
nataliya.s.stanasiuk@lpnu.ua

Abstract. The paper proposes an approach to solving multidimensional systems of nonlinear equations based on the use of the OpenMP parallelization mechanism and the multicore architecture of modern computers. The software product, which performs the main function - the parallelization of the numerical solution of multidimensional SNE by the Newton method, is developed. The paper introduces the algorithm of parallelization of the Black-Scholes algorithm based on OpenMP technology for prediction of the option on the European market is proposed. The analysis of the speed and efficiency of calculations with different number of processor cores is carried out. As a result, appropriate estimates of the acceleration and efficiency coefficients were obtained. The proposed method is easily scaled to a different number of processor cores. A number of numerical experiments were conducted. The obtained results also indicate the possibility of further optimization of the computational process by developing the multi-core architecture of modern computers.

Keywords: Forecasted Option, Black-Scholes Method, Paralleling, Multi-threading.

1 Introduction

In a modern economic society it is impossible to do without forecasting. One of the approaches is to use the Black-Scholes method for prediction the behavior of the stock on the stock market for a certain time. **The purpose of this work** is to create a program that will allow you to quickly and effectively use the Black-Scholes method algorithm in practice. The paper introduces the algorithm of parallelization of the Black-Scholes algorithm based on OpenMP technology for prediction of the option on the European market is proposed. For implementation were used the Microsoft Visual Studio 2017 programming environment, the C ++ programming technology, and OpenMP standard (Open Multi-Processing).

Black-Scholes method requires fast execution of the program. One of the best ways to speed up the work of a software product is to use the directives of parallelization of modern OpenMP technology. It allows to compare the results obtained and to analyze the speed and efficiency of the program with different streams and the corresponding multi-core architecture of modern computers. The object of the study is a parallel algorithm for forecasting options in the European market. The object of the research is the use of OpenMP technology in the process of parallelizing the Black-Scholes method.

2 Review of the Literature

Known [1-4], that the market for derivatives, derivatives of securities can perform its functions and stabilize the economy only under condition of reliable forecasting of prices for options.The problems of mathematical modeling of processes in economics and finances paid attention many Ukrainian scientists, among them Alekseev A., Besedin V., Veliky A., Vitlinsky V., Heyets V., Yeleyko Ya., Kostina N., Lukinov I., Sergienko I . Modern science has developed a fairly reliable apparatus for determining the fair price of an option. In the study of this problem, the contribution of domestic (Leonenko M., Mishura Yu., Parkhomenko V., Yadrenko M.) and foreign (Merton R., Bryan JO, Shiryaev A.) scientists is significant. World-famous is the Black-Scholes model.

The Black-Scholes method is one of the most effective stock market actions in predicted behavior [5]. Thanks to it, buyers / sellers of shares can get an estimate of the option price for the future and know whether they need to buy / sell their underlying assets.

When using the Black-Scholes method, you should be prepared for the next:

1. The method requires a large amount of input data.
2. The method requires a powerful computer to quickly process and process these data.
3. The method has minimal error and risk.

Option requirements, the value of which can be determined by the Black-Scholes method, limit its use for real options. So, using the method involves the assumption that there is a fixed date of option execution or decision making. Therefore, as a rule, this method is only suitable for options related to the scale of the project, in particular the expansion options, and can not be used for options related to the duration of the project and the reduction or refusal of the project. In fact, the limitation on the application of the Black-Scholes method naturally arises because the method itself was created to determine the value of financial options, for which a fixed date is mandatory. For real options related to the term of execution, reduction or refusal of the project, it is worth using the binomial method [6].

3 Statement of the problem

To successfully solve current tasks of current management of socio-economic processes, it is necessary to carry out a thorough quantitative analysis of these processes in order to forecast them. In our society a lot of things need to be predicted. Every day we see or hear the weather forecast, the forecast of currency growth, the forecast of changes in oil and gas prices, and others. For great precision, any method of forecasting requires the processing of a large amount of data. The study of the theoretical price for European options is also very important. You can do this by using the Black-Scholes Options pricing method [7].

This model is widely used in practice, among other things, can be used to evaluate all derivatives, including converting securities, and even to assess the equity of financially dependent firms.

According to the Black-Scholes model, the key element in determining the value of an option is the expected volatility of the underlying asset. Depending on the fluctuation of the asset, the price increases or decreases, which directly affects the cost of the option. Thus, if the value of the option is known, then it is possible to determine the level of volatility expected by the market [1-3].

The task is to develop a parallel algorithm for forecasting options in the European market based on the Black-Scholes method and the modern OpenMP parallel programming technology [8, 9]. And also create a software product that could be used effectively in practice. To perform the above task you need:

- Analyze the subject area, available input and output data;
- Get acquainted with possible ways to solve the problem and choose the best one;
- Thoroughly examine the algorithm of the method and evaluate the implementation methods;
- Create ways to store data;
- Organize the Black-Scholes method;
- Enable the possibility of parallel execution of the program

3.1 Black-Scholes model for option evaluation

An option is a contract concluded between two investors, one of which writes out (sells) an option, and the other one buys it and acquires the right (but not obligation) in the term specified in the option term or to purchase a fixed asset at a fixed price (in its composition may include, for example, futures contracts) of the person that the option, or sell him the asset.

In this regard, the following options are distinguished [10]:

- Call option – the right of the buyer of the option (and not his obligation) to obtain from the seller the option of a certain property value (equity, loan, etc.) at a fixed price, or to make a calculation, in the agreed time.
- Put option – the right to sell property value at a fixed price, or make a settlement at a certain future time point.

- Currently, the terms "call" and "put" are used as standard notation for options, regardless of their basis. Stock exchange sometimes introduce option-to-double options - they contain the seller's right to realize a double amount of value (commodity, security, security), which is the basis of the option, - Put-to-more Option, or buyer's right to purchase a double amount these values - Call-of-more Option.

The owner (buyer) of the option pays the seller a kind of commission - the premium per unit of the underlying asset, which is called the price, or the full value of the option (like the price of the insurance policy).

The size of the bonus affects a whole set of factors:

- the ratio between the current (market and contract prices);
- term of the option;
- stability of the course of the underlying asset.

The buyer's risk in the options deal is within the premium paid to them; the seller's risk is not limited and the latter's income is based on premiums. The buyer's payment of the premium is obligatory and is made on the stock exchange through the payment.

The price of the underlying asset (its unit), agreed in the option contract, is called the exercise price (exercise) of the option, or the contract price.

Here it should be emphasized that there are three types of different options within the options for buying (calling) and selling (put) shares, each of which has its own peculiarities.

These include:

- Internal options - have a strike price below the current market price for the put option. This means that the buyer of such an option can immediately exercise his right and earn a net income. However, given that each participant in options trading is interested in profitable transactions, then, as a rule, the premium on internal options always covers the specified price difference. With respect to such options, there is a notion of intrinsic price. It is always equal to the difference between the market price and the execution price.
- Market options - have an exercise price equal to or very close to the price of the underlying stock at the time of the option sale.
- External Options - characterized by the fact that their exercise prices are well above the base stock price under call options and significantly lower for put options.

Thus, from these types of options we can see the relationship between the contract price and the risk [11]. Therefore, the premium increases from the first kind to the last.

Our program will use the classic options, they are based on the fact that the options that were the subject of the agreement from the beginning, firmly fixed the future price of the basis (or the value of the calculated indicator). In conventional classic options, the principle of constant price applies throughout the life of the contract.

In 1973, F. Black and M. Scholes published an article, The Pricing of Options and Corporate Liabilities, in the Journal of Political Economy. The model proposed in this article has radically changed the approach to the analysis of options and other securities. To determine the value of the option, the authors proposed a formula whose starting elements are known except one, and even this single element can be estimated in a reasonable approximation [5, 9].

Black and Scholes made a number of baseline assumptions that many researchers are working to validate. Among these tenets are:

1. You can estimate the fluctuations (standard deviation) of the stock return.
2. There is a constant interest rate on risk-free investments over time.
3. There is no cost to conclude the agreement; when concluding agreements without coverage for a term (transactions with a short position), the seller receives money immediately.
4. Taxes do not matter.
5. No dividends.
6. The share price is a random value; the price for period t has a log-normal distribution.
7. Trading is carried out continuously.

Obviously, this assumption is an idealization of real market power. Therefore, they should not be considered as joining people designed to use these models.

The model's conclusion is based on the concept of risk-free hedging. By buying stocks and at the same time selling call options on these stocks, an investor can construct a risk-free position where earnings on the shares will accurately offset losses on the options, and vice versa [17].

The risk-free hedged position should yield a profit at a rate equal to the risk-free interest rate, otherwise there would be an opportunity to remove the arbitrage income and investors, trying to take advantage of this opportunity, would bring the option price to the equilibrium level determined by the model.

The Black-Scholes model is based on the assumption that future stock returns have a log-normal distribution with constant root mean square deviation - and that's all that is said about common stock returns. The expected return on them in this model affects the price of the option only indirectly, because of the stock price.

3.2 Black-Scholes algorithm

The option cost is calculated using the Black-Scholes formula developed to evaluate financial options [12]:

$$F = SN(d_1) - S_0 \times N(d_2), \tag{1}$$

Where

$$d_1 = \frac{\ln \frac{S}{S_0} + (\sigma + 0.5\sigma^2) \times T}{\sigma\sqrt{T}}, \tag{2}$$

$$d_2 = \frac{\ln \frac{S}{S_0} + (\sigma - 0.5\sigma^2) \times T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T} \ , \qquad (3)$$

Here, $N(x)$ the probability that the deviation will be less under the conditions of the standard normal distribution (thus, and limit the range of values for the function of the standard normal distribution);

S - the current market value of the shares;

S_0 - the price of execution;

T -the expiration time of the option in years;

σ - standard annual deviation of the stock price.

From the analysis of this formula, it follows that the option price is then higher when:

- the current market price of the stock is high (*S*);
- more time before the expiration of the option (*T*);
- greater risk.

Therefore, to increase the investment attractiveness of the project, it is more appropriate for companies to focus on increasing revenues rather than reducing costs.

The main difficulties that may arise when applying this model are related to obtaining reliable source data required for the calculation (time to implementation of the projected capabilities, values of variance and others).

3.3 The parallelization algorithm of the Black-Scholes method

At each iteration of the Black-Scholes algorithm, the following operations are performed:

1. A pricing vector is created based on market price, risk, volatility, and time step.

1.1. We create an array of normally distributed random variables.

1.2. We calculate the estimated price at a given step.

1.3. Write down the value in the array of possible stock values.

1.4. We find the average of the array of all possible values at this step. This is an average value and will be our predicted price at some point.

In order to parallel this algorithm, we propose to use OpenMP concurrent programming technology, which can be considered as a high-level add-on over Pthreads (or similar thread libraries). OpenMP implements multi-threaded computations, in which the master thread creates a set of slave threads and the task is shared between them. It is assumed that threads run in parallel on a machine with multiple processors (the number of processors need not be greater than or equal to the number of threads) [13-17].

The tasks that flow in parallel, as well as the data needed to perform these tasks, are described using the specific preprocessor directives of the respective language -

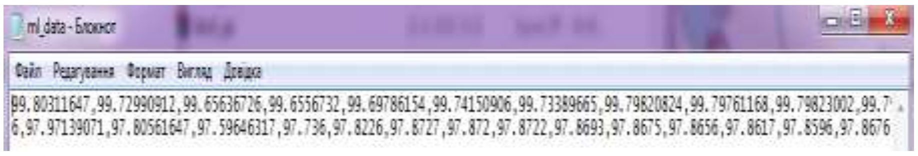
pragm. The number of streams created can be controlled by the program itself, by calling library procedures and externally, using environment variables [12,14].

Advantages of this technology:

- with the idea of "incremental parallelization", OpenMP is ideal for developers seeking to quickly parallel their computing programs with large, parallel cycles. The developer does not create a new concurrent program, but simply sequentially adds OpenMP directives to the program text;
- OpenMP is a rather flexible mechanism, which gives the developer great control over the behavior of the parallel program;
- it is envisaged that the OpenMP program on a single-processor platform can be used as a serial program, ie there is no need to support serial and parallel versions. OpenMP directives are simply ignored by a serial compiler, and stubs can be invoked to call OpenMP procedures;
- support for the so-called "orphan" (detached) directives, that is, directives for synchronization and division of labor may not be directly within the lexical context of a parallel domain.

4 Numerical experiments

The first step is to create the file ml_data.csv, which records the input data required for our method. Namely: 180 numbers, which in percentage show the behavior of the stock in the European market for 180 minutes. The data is written in one line, without spaces, but through commas.



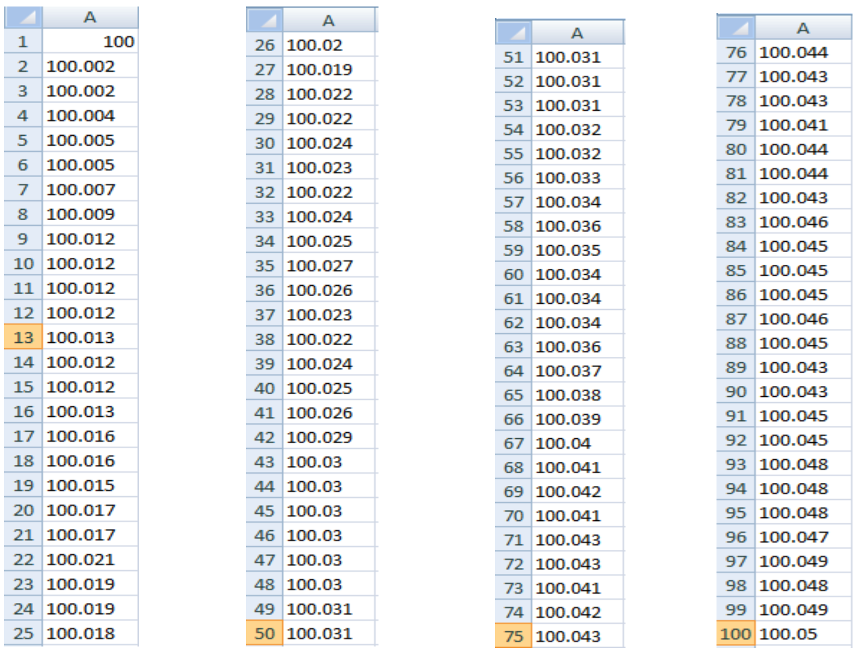
```
99.80311647,99.72990912,99.65636726,99.6556732,99.69786154,99.74150906,99.73389665,99.79820824,99.79761168,99.79823002,99.7
6,97.97139071,97.80561647,97.59646317,97.736,97.8226,97.8727,97.872,97.8722,97.8693,97.8675,97.8656,97.8617,97.8596,97.8676
```

Fig. **1.** The contents of the ml_data.csv file.

The program has the following basic functions:

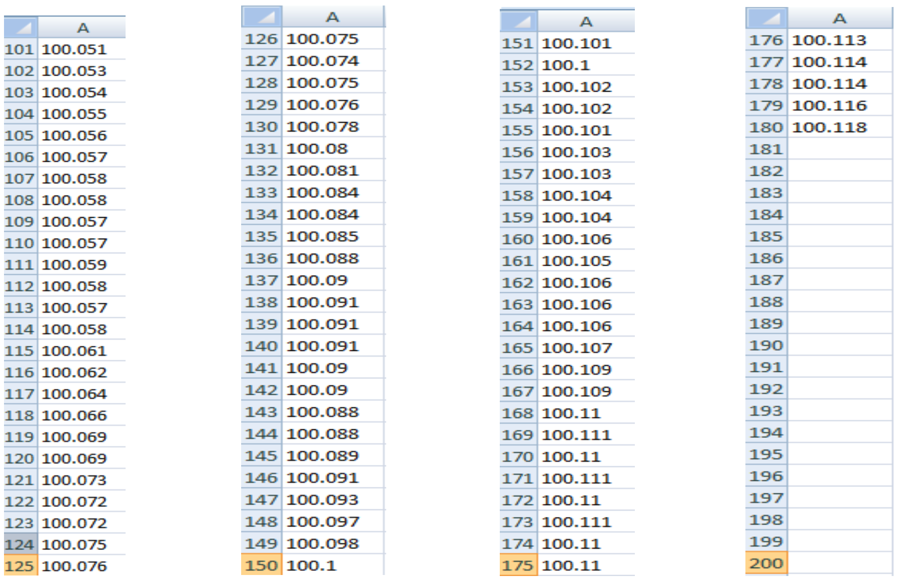
- calcVolatility(calculates volatility);
- find2DMean(finds the average value in the stock data array);
- randGen(generates an array of random numbers based on the normal distribution);
- runBlackScholesModel(designs the Black-Scholes model).

The result of the program execution is written to the opt.scv file, which will have 180 data describing the behavior of the price at each time step as a percentage.



	A		A		A		A
1	100	26	100.02	51	100.031	76	100.044
2	100.002	27	100.019	52	100.031	77	100.043
3	100.002	28	100.022	53	100.031	78	100.043
4	100.004	29	100.022	54	100.032	79	100.041
5	100.005	30	100.024	55	100.032	80	100.044
6	100.005	31	100.023	56	100.033	81	100.044
7	100.007	32	100.022	57	100.034	82	100.043
8	100.009	33	100.024	58	100.036	83	100.046
9	100.012	34	100.025	59	100.035	84	100.045
10	100.012	35	100.027	60	100.034	85	100.045
11	100.012	36	100.026	61	100.034	86	100.045
12	100.012	37	100.023	62	100.034	87	100.046
13	100.013	38	100.022	63	100.036	88	100.045
14	100.012	39	100.024	64	100.037	89	100.043
15	100.012	40	100.025	65	100.038	90	100.043
16	100.013	41	100.026	66	100.039	91	100.045
17	100.016	42	100.029	67	100.04	92	100.045
18	100.016	43	100.03	68	100.041	93	100.048
19	100.015	44	100.03	69	100.042	94	100.048
20	100.017	45	100.03	70	100.041	95	100.048
21	100.017	46	100.03	71	100.043	96	100.047
22	100.021	47	100.03	72	100.043	97	100.049
23	100.019	48	100.03	73	100.041	98	100.048
24	100.019	49	100.031	74	100.042	99	100.049
25	100.018	50	100.031	75	100.043	100	100.05

Fig. 1. The output from the file opt.csv (1).



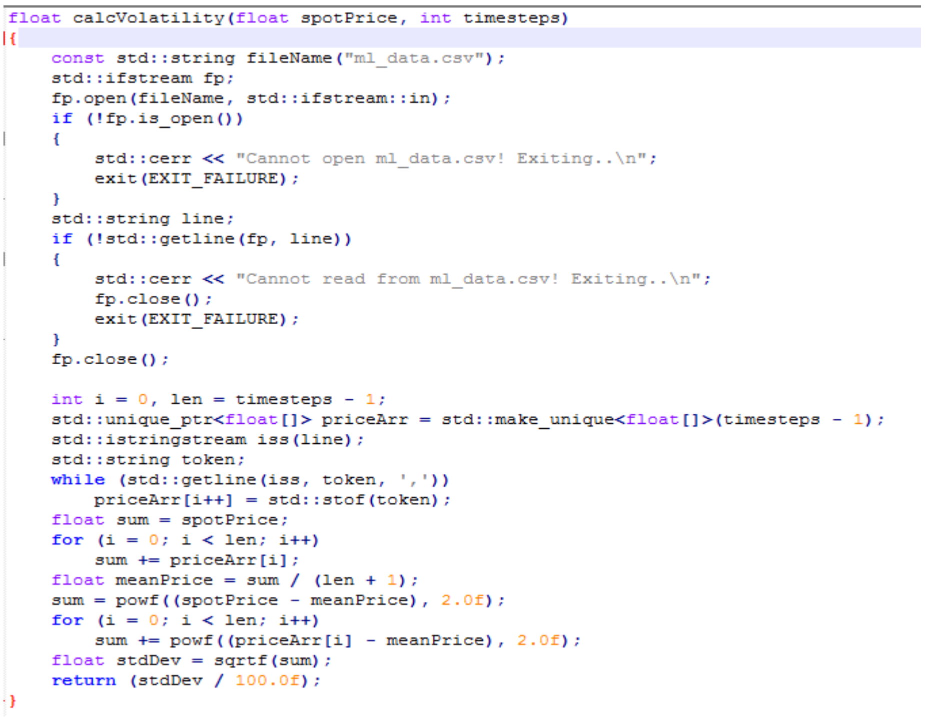
	A		A		A		A
101	100.051	126	100.075	151	100.101	176	100.113
102	100.053	127	100.074	152	100.1	177	100.114
103	100.054	128	100.075	153	100.102	178	100.114
104	100.055	129	100.076	154	100.102	179	100.116
105	100.056	130	100.078	155	100.101	180	100.118
106	100.057	131	100.08	156	100.103	181	
107	100.058	132	100.081	157	100.103	182	
108	100.058	133	100.084	158	100.104	183	
109	100.057	134	100.084	159	100.104	184	
110	100.057	135	100.085	160	100.106	185	
111	100.059	136	100.088	161	100.105	186	
112	100.058	137	100.09	162	100.106	187	
113	100.057	138	100.091	163	100.106	188	
114	100.058	139	100.091	164	100.106	189	
115	100.061	140	100.091	165	100.107	190	
116	100.062	141	100.09	166	100.109	191	
117	100.064	142	100.09	167	100.109	192	
118	100.066	143	100.088	168	100.11	193	
119	100.069	144	100.088	169	100.111	194	
120	100.069	145	100.089	170	100.11	195	
121	100.073	146	100.091	171	100.111	196	
122	100.072	147	100.093	172	100.11	197	
123	100.072	148	100.097	173	100.111	198	
124	100.075	149	100.098	174	100.11	199	
125	100.076	150	100.1	175	100.11	200	

Fig. 2. The output from the file opt.csv (2).

calcVolatility(float spotPrice, int timesteps): a function that calculates volatility based on stock input.

Stages of execution:

- Reading the file.
- The data in the file is recorded in a single tape. We read the tape, close the file.
- We turn a string into a stream.
- Extract values read from a file and write them to an array.
- We find the average value of the calculated prices in one minute.
- Calculation of market volatility as standard deviation.
- The function returns volatility as a percentage.



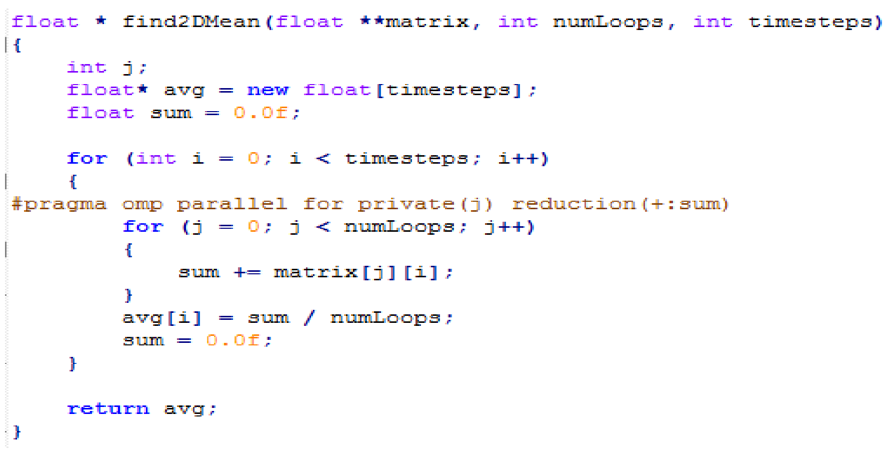
```
float calcVolatility(float spotPrice, int timesteps)
{
    const std::string fileName("ml_data.csv");
    std::ifstream fp;
    fp.open(fileName, std::ifstream::in);
    if (!fp.is_open())
    {
        std::cerr << "Cannot open ml_data.csv! Exiting..\n";
        exit(EXIT_FAILURE);
    }
    std::string line;
    if (!std::getline(fp, line))
    {
        std::cerr << "Cannot read from ml_data.csv! Exiting..\n";
        fp.close();
        exit(EXIT_FAILURE);
    }
    fp.close();

    int i = 0, len = timesteps - 1;
    std::unique_ptr<float[]> priceArr = std::make_unique<float[]>(timesteps - 1);
    std::istringstream iss(line);
    std::string token;
    while (std::getline(iss, token, ','))
        priceArr[i++] = std::stof(token);
    float sum = spotPrice;
    for (i = 0; i < len; i++)
        sum += priceArr[i];
    float meanPrice = sum / (len + 1);
    sum = powf((spotPrice - meanPrice), 2.0f);
    for (i = 0; i < len; i++)
        sum += powf((priceArr[i] - meanPrice), 2.0f);
    float stdDev = sqrtf(sum);
    return (stdDev / 100.0f);
}
```

find2DMean(float **matrix, int numLoops, int timesteps):a function that finds the mean of the 2D array through the first index inLoops.

Stages of execution:

- A private copy of the 'sum' variable is created for each thread.
- At the end of the run, all private copies of the variable that are written to the global variable are available
- We calculate the average value by columns
- Return the mean

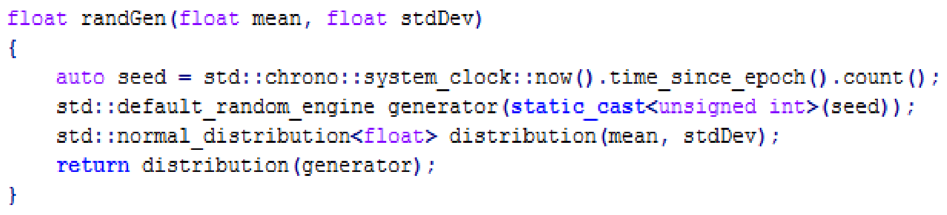


```
float * find2DMean(float **matrix, int numLoops, int timesteps)
{
    int j;
    float* avg = new float[timesteps];
    float sum = 0.0f;

    for (int i = 0; i < timesteps; i++)
    {
#pragma omp parallel for private(j) reduction(+:sum)
        for (j = 0; j < numLoops; j++)
        {
            sum += matrix[j][i];
        }
        avg[i] = sum / numLoops;
        sum = 0.0f;
    }

    return avg;
}
```

randGen(float mean, float stdDev):randGen function Creates a random number based on the standard distribution of the mean of 0.0 and standard deviation of 1.0.



```
float randGen(float mean, float stdDev)
{
    auto seed = std::chrono::system_clock::now().time_since_epoch().count();
    std::default_random_engine generator(static_cast<unsigned int>(seed));
    std::normal_distribution<float> distribution(mean, stdDev);
    return distribution(generator);
}
```

runBlackScholesModel (float spotPrice, int timesteps, float riskRate, float volatility):

a function that designs the Black-Scholes model.

Stages of execution:

1. Create an array of normally distributed random variables, an array of stock prices at each time interval.
2. We put a stock price at the beginning of data processing equal to the market price.
3. Fill the array with random normal values.
4. We force the Black-Scholes method to calculate the stock price for the next period of time.

$$C = S \cdot \exp\left(r - \frac{v^2}{2}\right) \cdot \sigma + v \cdot rand \cdot \sqrt{\sigma}\,, \qquad (5)$$

where:
C - predicted value;
S - market price;
r - risk (0.001);
v - volatile;

$\sigma - 1/t$;

rand – normally distributed random variable.

```
float * runBlackScholesModel(float spotPrice, int timesteps, float riskRate, float volatility)
{
    float  mean = 0.0f, stdDev = 1.0f;
    float  deltaT = 1.0f / timesteps;
    std::unique_ptr<float[]> normRand = std::make_unique<float[]>(timesteps - 1);
    float* stockPrice = new float[timesteps];
    stockPrice[0] = spotPrice;
    for (int i = 0; i < timesteps - 1; i++)
        normRand[i] = randGen(mean, stdDev);
    for (int i = 0; i < timesteps - 1; i++)
        stockPrice[i + 1] = stockPrice[i] * exp(((riskRate - (powf(volatility, 2.0f) / 2.0f))
    * deltaT) + (volatility * normRand[i] * sqrtf(deltaT)));
    return stockPrice;
}
```

5 Analysis of results

In Fig. 4. Black-Scholes option forecasting schedule for the next 180 minutes is presented.

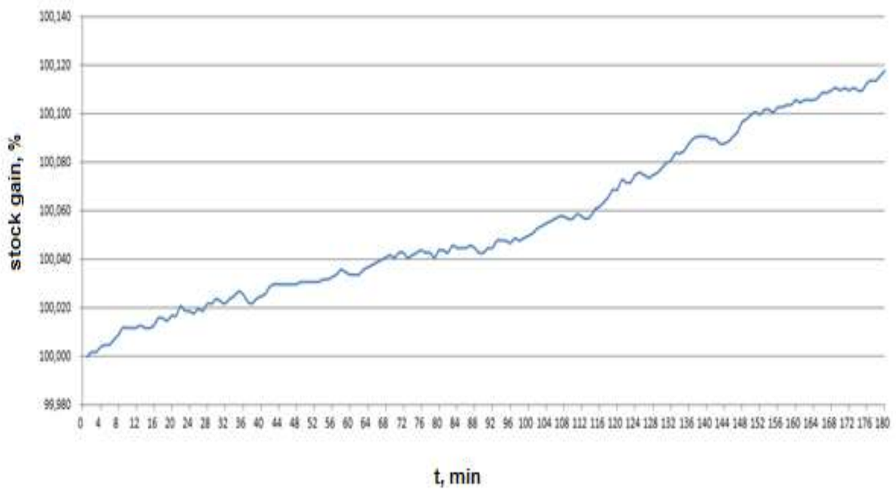


Fig. 3. Increase in shares for the next 180 minutes.

In the experimental study, Black-Shouse stock prediction was performed in parallel with one, two, four and eight threads on eight nuclear processors using OpenMP technology. In the Table 1 shows the results of numerical experiments.

Table 1. Parallel algorithm execution time.

Flows	1	2	4	8
Temporary review, s	3367	1673	883	526

Analyzing the results, we can conclude that with the increase in the number of flows, the time during which the algorithm of the program finds forecasting the option price in the European market is almost halved.

The acceleration coefficient according to the formula is calculated in the paper. In the Table 2 shows the value of the coefficient when the number of threads varies. The graph of acceleration change depending on the number of threads on the octa-core processor is presented in Fig. 5.

Table 2. The value of the acceleration coefficient.

Flows	Acceleration
1	1
2	2,01
4	3,813
8	6,401

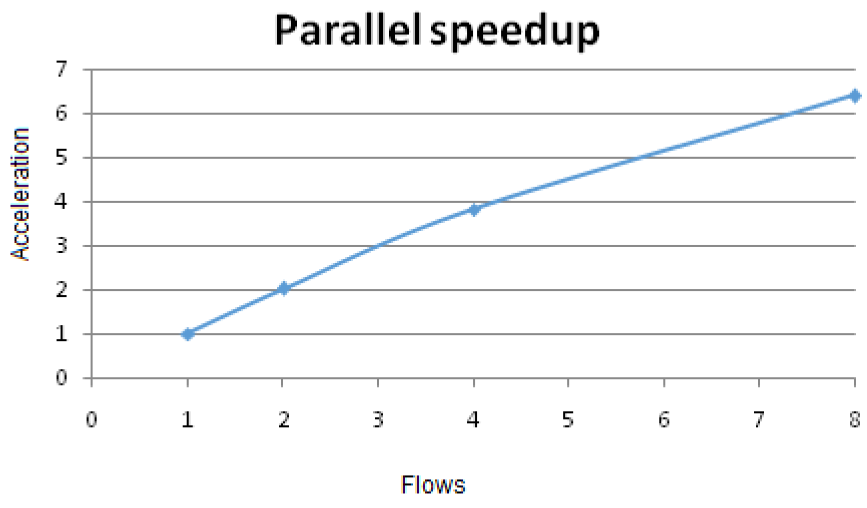


Fig. 4. Acceleration of algorithm execution depending on the number of threads

The results shown in Fig. 5, show that the developed parallel algorithm provides normal scalability, that is, the task execution time decreases proportionally as the number of threads increases. The highest acceleration is achieved by using 8 threads on an eight-core processor.

The coefficient of efficiency is calculated by the formula $E_p = S_p / p$. In the Table 3 shows the value of the coefficient when the number of threads varies. A graph of performance changes depending on the number of threads on an octa-core processor is presented in Fig. 6.

Table 3. The value of the efficiency ratio

Threads	Efficiency
1	1
2	1,005
4	0,95
8	0,8001

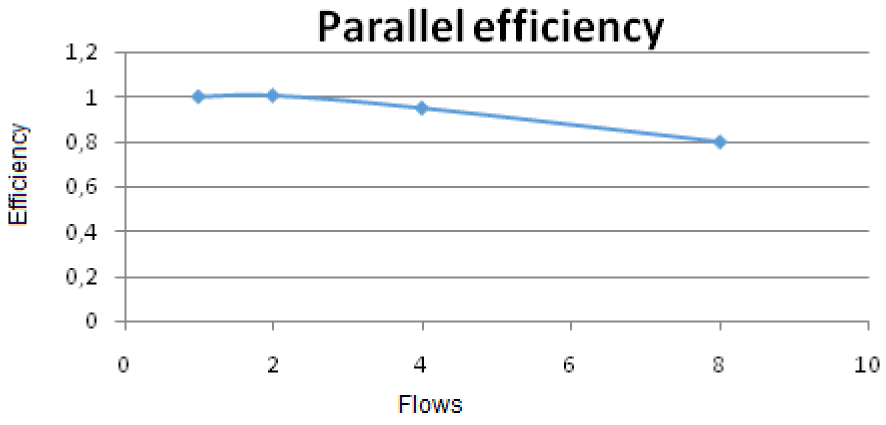


Fig. 5. Performance of the algorithm, depending on the number of threads

6 Conclusion

The subject area is analyzed in detail. The most universal method of finding effective investment solutions - the Black-Scholes method, has been studied and evaluated. Has been developed a program that enables the calculation of options forecasts in the European market over a given time. To speed up the forecast, the timing of the algorithm's performance has been significantly improved through the use of OpenMP concurrent programming technology and its many-stream properties. In doing so, we were able to compare the speed of program execution with varying the number of threads on the octa-core computer architecture.

References

1. Krste, Asanovic and all.: The Landscape of Parallel Computing Reseach: A View from Berkeley. University of California, Berkeley. Technical Report № UCB/EECS-2006-183, 56 p. (2006).

2. Yakovlev, M.F., Gerasymova, T.O., Nesterenko, A.N.: Characteristic feature of the solving both of non-linear systems and systems of ordinary differential equations on parallel computer. In Proceedings of international symposium "Optimization problems of computations" (OPC - XXXV). Kyiv: V.M. Glushkov Institute of cybernetics of NAS of Ukraine, 2009. Kyiv: Vol. 2. P. 435-439. (2009).
3. Yakovlev, MV.F., Nesterenko, A.N., Brusnikin, V.N. Problems of the efficient solving of non-linear systems on multi-processor MIMD-architecture computers. Mathematical machines and systems. (4). P. 12-17. (2014).
4. Shakhovska, N., Boyko, N., Pukach, P.: The information model of cloud data warehouses. In the International conference on computer science and information technologies "Advances in Intelligent Systems and Computing" (AISC), Vol. 871, pp. 182–191, CSIT 2018, Lviv, Ukraine (2018)
5. Khymych, A.N., Molchanov, Y.N., Popov, A.V. other: Parallel algorithms for solving problems of computational mathematics. Kiev: Scientific Opinion, 248 pp. (2008).
6. Autar, Kaw: NONLINEAR EQUATIONS - Newton-Raphson Method-More Examples, Civil Engineering. August 7, 4 pp. (2009)
7. Boyko, N., Shakhovska, K.: Information system of catering selection by using clustering analysis. In 2018 IEEE Ukraine Student, Young Professional and Women in Engineering Congress (UKRSYW), pp.7-13, October 2 – 6, 2018, Kyiv, Ukraine (2018)
8. Autar, Kaw: NONLINEAR EQUATIONS - Newton-Raphson Method-More Examples, Electrical Engineering. August 7, 4 pp. (2009)
9. Autar, Kaw. NONLINEAR EQUATIONS - Newton-Raphson Method-More Examples, Mechanical Engineering. August 7, 3 pp. (2009).
10. Kakhaner, D., Mouler, K., Nэsh, S.: Numerical methods and software. «Mir» publishing house, 575 pp., (1998).
11. Mochurad, L.I. Method of reduction model for calculation of electrostatic fields of electronic fields of electronic optics systems. Science journal Radioelektronika, informatics, management, 1(48), pp. 29-40 (2019). (In Ukrainian)
12. Voss, M.: OpenMP Share Memory Parallel programming. Toronto, Kanada (2003).
13. Chapman, B., Jost, G., Ruud van der Pas: Using OpenMP: portable shared memory parallel programming (Scientific and Engineering Computation). Cambridge, Massachusetts: The MIT Press (2008).
14. Chandra, R., Menon, R., Dagum, L., Kohr, D., Maydan, D., McDonald, J.: Parallel Programming in OpenMP. Morgan Kaufinann Publishers (2000).
15. Ananth, Grama, Anshul, Gupta, George, Karypis, Vipin, Kumar: «Introduction to Parallel Computing» Addison Wesley, ISBN- 0-201-64865-2, 856 p. (2003).
16. Boyko, N.: Advanced technologies of big data research in distributed information systems. Radio Electronics, Computer Science, Control, vol. 4, pp. 66-77, Zaporizhzhya: Zaporizhzhya National Technical University (2017)
17. Mochurad, L., Nataliya, Boyko.: Solving Systems of Nonlinear Equations on Multi-core Processors. In the International Conference on Computer Science and Information Technologies Advances in Intelligent Systems and Computing IV, pp. 90-106, CSIT 2019, September 17–20, 2019, Lviv, Ukraine. DOI: 10.1007/978-3-030-33695-0 (2019)