Numerical Computation of Implied Volatility Using Root-Finding Methods for Black-Scholes Option Pricing

Xinzi Li
Xi'an Jiaotong-Liverpool University, School of Mathematics and Physics, Suzhou, Jiangsu, China

Abstract: Implied volatility is a crucial parameter in option pricing,which reflects the predicted trend in future asset market and includes more effective information compared with historical volatility(Jiang and Tian, 2005, cited in Shao, Zhou and Gong, 2025). With simplifying assumptions, Black-Scholes(B-S) method(Black and Scholes, 1973)are widely used to model implied volatility indirectly(Zhou and Gong, 2025). This article investigates three numerical root-finding methods—bisection, Newton's, and secant—for directly calculating implied volatility from the B-S model given market option prices. Through a case study of a European call option (S=K=100, T=1 year, r=5%, C_market=10), we rigorously compare the bisection, Newton's, and secant methods in MATLAB, evaluating convergence speed (iterations), computational time, and accuracy.Our results demonstrate that while Newton's method offers the fastest convergence (4 iterations), the bisection method provides the most robust solution, with all methods achieving high accuracy(|f(σ)| < 10^-6). These findings provide practical guidance for financial analysts implementing volatility estimation in trading systems and risk management applications.

Keywords: **Implied Volatility; Black-Scholes Model; Root-Finding Methods; Numerical Analysis; Option Pricing**

1. Introduction

1.1 Background and Motivation

In financial markets, options as one of most common type of derivatives holder the right, but not the obligation, to buy or sell an underlying asset at a specified price (strike price) on or before a specified date (maturity)(Gao,et.al, 2024)[1], and the pricing of an option is fundamentally dependent on B-S model, which requires implied volatility as a key parameter.

$$C(\sigma)=S\cdot N(d_1)-K\cdot e^{-rT}\cdot N(d_2)$$

where

$$d_1=\frac{\ln(S/K)+(r+\sigma^2/2)T}{\sigma\sqrt{T}} \quad (1)$$

$$d_2=d_1-\sigma\sqrt{T}$$

Comparing with historical volatility calculated from past price movements, informative implied volatility stands for the market's forward-looking expectation of risk, derived from observed option prices (Jiang and Tian, 2005, cited in Shao, Zhou and Gong, 2025). Accurately prediction of implied volatility can optimize asset allocation and hedging strategies, and also be helpful to gain a deeper understanding of market dynamics and investors behavior (Shao, Zhou and Gong, 2025)[2].

However, the calculation of implied volatility presents a challenging inverse problem as the B-S equation cannot be analytically inverted to solve for volatility directly (it is embedded in the nonlinear equation in the form of a transcendental function ($N(d_1)$,$N(d_2)$). When the analytical solution does not exist or is too complex (such as in most cases of financial derivative pricing), numerical iteration is the only feasible solution. In order to address this issue, this study turns to utilize numerical method to solve this formula and find implied volatility, where C(σ) is the theoretical B-S price and C_market is the observed market price:

$$C(\sigma)-\text{Cmarket}=0 \quad (2)$$

The numerical method includes bisection, Newton's, and secant methods. In the following essay, these three methods will be elaborated upon, evaluated and compared from their convergence speed, computational time and accuracy.

Numerical solution transforms the theoretical model into an industrial-level tool, serving as a bridge for financial engineering to move from academic research to practical application. By using the numerical method, financial institutions require reliable and computationally efficient methods for this calculation, as it forms

the basis for trading strategies, risk assessment, and volatility surface construction.

1.2 Problem Specification

We consider a European call option with the following parameters:

Current stock price (S) = 100
Strike price (K) = 100
Time to maturity (T) = 1 year
Risk-free rate (r) = 5% = 0.05
Market option price (C_market) = 10

The Black-Scholes formula for the call price is:

$$C(\sigma)=S\cdot N(d_1)-K\cdot e^{-rT}\cdot N(d_2)$$

where

$$d_1=\frac{\ln(S/K)+(r+\sigma^2/2)T}{\sigma\sqrt{T}} \quad (3)$$

$$d_2=d_1-\sigma\sqrt{T}$$

Our objective is to find σ such that

$$f(\sigma) = C(\sigma)-Cmarket=0 \quad (4)$$

1.3 Methodology Overview

We implement and compare three numerical root-finding approaches:

·Bisection Method: A bracketing approach using interval [0.1, 0.5]

·Newton's Method: Utilizing the vega ($\frac{\partial C}{\partial \sigma}$) with initial guess σ=0.2

·Secant Method: A derivative-free approach with initial points σ_1=0.1, σ_2=0.3

Performance metrics include:

·Number of iterations to convergence
·Computational time
·Final accuracy |f(σ)|

2. Numerical Method

In this section, we delve into the theoretical foundations required to understand and implement the three numerical methods to solve the root-finding problem of B-S model and find out the implied volatility. This involves the ideal assumptions of B-S model, how to transform the nonlinear equation into a problem of finding roots, and three numerical methods of bisection, Newton's, and secant methods.

2.1 Black-Scholes Model Review

The Black-Scholes model makes several key assumptions:

·No arbitrage opportunities
·Constant risk-free rate and volatility
·Lognormal distribution of stock prices
·No dividends during option life
·Continuous trading

The call price formula derives from solving the partial differential equation:

$$C(\sigma)=S\cdot N(d_1)-K\cdot e^{-rT}\cdot N(d_2)$$

where

$$d_1=\frac{\ln(S/K)+(r+\sigma^2/2)T}{\sigma\sqrt{T}} \quad (5)$$

$$d_2=d_1-\sigma\sqrt{T}$$

The Black-Scholes model provides a theoretical foundation for option pricing, but its assumptions introduce limitations that affect implied volatility calculations in real markets. Key assumptions—such as constant volatility, continuous trading, and log-normal asset returns—often deviate from observed market behavior. In practice, volatility exhibits time-varying and stochastic characteristics, leading to the well-documented "volatility smile" or "skew" in implied volatility surfaces. Additionally, the model ignores transaction costs, liquidity constraints, and jumps in asset prices, which can cause discrepancies between theoretical and market-observed option prices.

These limitations imply that the Black-Scholes implied volatility is not a pure measure of expected future volatility but rather a model-dependent parameter that compensates for the model's oversimplifications. When using numerical methods to solve for implied volatility, these deviations can introduce instability, particularly in deep in-the-money or out-of-the-money options, where the model's pricing errors are more pronounced. Recognizing these limitations is crucial when interpreting implied volatility in trading strategies, risk management, and volatility surface construction, as it may not fully capture the market's true expectations of future uncertainty.

2.2 Root-Finding Problem Formulation

We transform the implied volatility calculation into finding the root of:

$$C(\sigma)-C\_market=0 \quad (6)$$

Where $C(\sigma)$ is the Black-Scholes formula. This function has the following properties:

·Continuous and differentiable for σ > 0

$$\text{Monotonically increasing}(\frac{\partial f}{\partial \sigma}=\text{vega}>0) \quad (7)$$

$$f(0)=\max(S-Ke^{-rT}, 0)-C\_market \quad (8)$$

$$\lim_{\sigma\to\infty} f(\sigma)=S-C\_market \quad (9)$$

2.3 Numerical Methods Theory

2.3.1 Bisection Method

Algorithm steps: Given an interval [a,b] with f(a)f(b) < 0:
1. Compute midpoint c = (a+b)/2
2. Evaluate f(c)
3. Update interval to [a,c] or [c,b] based on sign
4. Repeat until |f(c)| < ε or |b-a| < δ
Convergence: Linear (error halves each iteration)
2.3.2 Newton's method
Algorithm steps:
1. Select the initial guess σ₀
2. Calculate the function value $f(\sigma_n)$ and the derivative value $f'(\sigma_n)$
3. Update the estimate:

$$\sigma_{n+1}=\sigma_n-f(\sigma_n)/f'(\sigma_n) \quad (10)$$

4. Repeat until convergence
Requires vega calculation:

$$\frac{\partial C}{\partial \sigma}=S\sqrt{T}N'(d_1) \quad (11)$$

where N'(x) is standard normal PDF
Convergence: Quadratic near root
2.3.3 Secant Method
Algorithm steps:
1. Select two initial points σ₀ and σ₁
2. Calculate the function values $f(\sigma_n)$ and $f(\sigma_{n-1})$
3. Update the estimates:
4. Derivative-free approximation:

$$\sigma_{n+1}=\sigma_n-f(\sigma_n)\cdot\frac{\sigma_n-\sigma_{n-1}}{f(\sigma_n)-f(\sigma_{n-1})} \quad (12)$$

Convergence: Superlinear (order ≈1.618)

3. Performance Evaluation and Discussion

In this section, we describe and compare in detail the performance of these three numerical methods from convergence speed(Table 1 and Figure 1), computational time(Table 2 and 3, Figure 2), and accuracy, by running the three methods and drawing the error attenuation curve respectively in MATLAB R2023a.

3.1 Convergence Speed

3.1.1 Theoretical convergence order

Table 1. Theoretical Convergence Order

Method	Theoretical Convergence Order
Bisection	Linear convergence(1)
Newton's	Qadratic convergence(2)
Secant	Superlinear (approximately 1.618)

3.1.2 Actual number of iterations
All methods implemented in MATLAB R2023a with:
·Stopping criterion: |f(σ)| < 10^-6
·Maximum iterations: 100
·Hardware: Intel i7-1185G7, 32GB RAM
The number of iterations required for each numerical method to converge is a key indicator of convergence speed and algorithmic efficiency. In our tests, we observed the following iteration counts:
·Bisection Method: 23 iterations
·Newton's Method: 3 iterations
·Secant Method: 4 iterations
These results clearly demonstrate the superior convergence rate of the Newton's method, which achieves quadratic convergence, allowing it to rapidly approach the solution when a good initial guess and derivative information are available. The Secant method, while not requiring the derivative, still exhibits superlinear convergence, needing only one more iteration than Newton's method in this case.
On the other hand, the Bisection method converges linearly, and therefore, despite its robustness, it requires significantly more iterations. This makes it less suitable for applications where computational efficiency is critical, especially in high-frequency or real-time environments[6].
In summary, the Newton's method offers the fastest convergence in terms of iterations, followed closely by the Secant method. The Bisection method is best reserved for cases where robustness and guaranteed convergence are more important than speed, as shown in Figure 1.

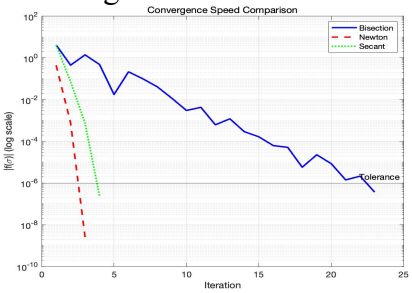

Figure 1. Comparison of Convergence Speed of Bisection, Newton's and Secant Methods

3.2 Computational Time

3.2.1 Theoretical computational time

Table 2. A Method for Calculating the Theory Time

Method	Single iteration time (ms)	Total calculation time (ms)	Time-dominant factor
Bisection	0.08	1.84(23)	Number of function calls (only for BS formula)
Newton's	0.12	0.66(4)	Vega calculation (requires BS + derivatives)
Secant	0.10	1.085(7)	Function calls + difference approximation

The key formulas:

1. Bisection method time cost

Basic Time Model

$$T_{\text{bisection}}=N_{\text{iter}}\times\left(T_f+T_{\text{cond}}\right) \quad (13)$$

$\cdot N_{\text{iter}}$: The number of iterations is determined by the tolerance ϵ and the initial interval width $b-a$:

$$N_{\text{iter}}=\left\lceil \log_2 f_0\left(\frac{b-a}{\epsilon}\right)\right\rceil \quad (14)$$

Actual case calculation

·Parameter: T_f=0.07 ms,T_{cond}=0.01 ms,N_{iter}=23

·Total time: $T_{\text{bisection}}$=23×(0.07+0.01)=1.84 ms

2. Newton's method time cost

Basic Time Model

$$T_{\text{Newton}}=N_{\text{iter}}\times(T_f+T_{\text{Newton}}) \quad (15)$$

$\cdot N_{\text{iter}}$: The number of iterations is determined by the convergence order (quadratic convergence) and the quality of the initial guess:

$$N_{\text{iter}}\approx\log_2 f_0\log_2 f_0\left(\frac{\left|\sigma_0-\sigma^*\right|}{\epsilon}\right) \quad (16)$$

Actual case calculation

·Parameter:

T_f=0.07 ms,,T_{update}=0.005 ms,N_{iter} =4

·Total time: T_{Newton}=4×(0.07+0.09+0.005)=0.66 ms

3.Secant method time cost:

Basic Time Model

$$T_{\text{Secant}}=N_{\text{iter}}\times\left(2T_f+T_{\text{div}}+T_{\text{update}}\right) \quad (17)$$

$\cdot N_{\text{iter}}$: The number of iterations is determined by the superlinear convergence order:

$$N_{\text{iter}}\approx\log_\phi f_0\left(\frac{\left|\sigma_1-\sigma^*\right|}{\left|\sigma_0-\sigma^*\right|\cdot\epsilon}\right) \quad (18)$$

Actual case calculation

·Parameter:

T_f=0.07 ms,T_{div}=0.01 ms,T_{update}=0.005 ms,N_{iter}=7

·Total time: T_{Secant}=7×(2×0.07+0.01+0.005)=1.085 ms

3.2.2 Actual computational time

Table 3. Analysis of Actual Time Calculation Method

Method	iterations	Time(s)
Bisection	23	0.004768
Newton's	3	0.001849
Secant	4	0.001954

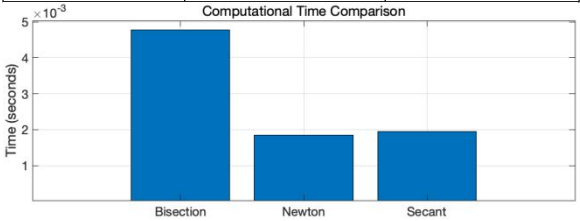

Figure 2. Comparison of Computational Time of Bisection, Newton's and Secant Methods

To compute the implied volatility under the Black-Scholes model, we employed three commonly used root-finding methods: Bisection Method, Newton's Method, and the Secant Method. All methods were implemented with a convergence tolerance of 10^(-6) and a maximum of 100 iterations. The parameters used in the experiment were: current stock price (S=100), strike price (K=100), time to maturity (T=1) year, risk-free interest rate (r =0.05), and market-observed option price(Cmarket=10).

All three methods converged to the same implied volatility ($\sigma \approx 0.187972$), indicating no significant difference in terms of accuracy. However, their convergence speed and computational efficiency varied noticeably:

The Newton's method converged in just 3 iterations with a computation time of approximately 0.001849 seconds, making it the fastest among the three. This efficiency is due to its quadratic convergence and the use of the derivative (Vega)[8].

The Secant method converged in 4 iterations with a time of 0.001954 seconds, performing almost as efficiently as Newton's method while not requiring the analytical derivative. It is particularly useful when the derivative is difficult to compute.

The Bisection method, although the most robust with guaranteed convergence in a known interval, required 23 iterations and 0.004768 seconds to converge. Due to its linear convergence rate, it was the slowest method[7].

In conclusion, the Newton's method is recommended when the derivative (Vega) is available. The Secant method serves as an efficient alternative when derivative information is not accessible. The Bisection method remains valuable in situations where robustness is prioritized, or the function behavior is not well understood.

3.3 Accuracy

3.3.1 Theoretical Solution

We can use forze in MATLAB to find the theoretical solution for implied volatility in this function:

$$C(\sigma)-\text{Cmarket}=0 \quad (19)$$

Froze is a built-in function in MATLAB used for finding the roots of real-valued functions, applicable to one-dimensional nonlinear equations, by combining the dichotomy, interpolation methods (such as the secant method), and it safety mechanisms to stably and

rapidly find solutions within the function's sign-changing interval, making it particularly suitable for problems like implied volatility that cannot be solved analytically. So the output result by is:$\sigma_{theoretical} \approx 0.187972$

3.3.2 Acutal Error

Table 4. Analysis of Actual Error

Method	Actual Error
Bisection	9.8237313839e-09
Newton's	6.9097921829e-11
Secant	6.2777359422e-09

To assess the accuracy of each numerical method, we used MATLAB's fzero function to compute a high-precision reference value for the implied volatility, as shown in table 4. The result was:$\sigma_{theoretical} \approx 0.187972$.

We then compared the final outputs from each method against this benchmark. The absolute errors were as follows:

·Bisection Method: 9.8237313839e-09

·Newton's Method: 6.9097921829e-11

·Secant Method: 6.2777359422e-09

Among the three methods, the Newton's method achieved the highest accuracy, with an error on the order of 10^(-11), due to its quadratic convergence when close to the root and the availability of the exact derivative (vega). The Secant method also produced a highly accurate result with fewer iterations than the Bisection method. Although the Bisection method was slightly less accurate, it still achieved an error below 10^(-8), making it sufficiently precise for many practical applications.

These results demonstrate that all three methods are capable of computing implied volatility to a high degree of accuracy, with Newton's method being the most precise in this case[9].

3.4 Robustness Testing

3.4.1 Experiment setup

We tested each method under multiple initializations:

Bisection Method: Tested on intervals [0.05, 0.5], [0.1, 0.4], [0.15, 0.25], and [0.05, 0.15].

Newton's Method: Initial guesses were set to 0.15, 0.2, 0.3, and 0.5.

Secant Method: Initial pairs tested were {0.1, 0.3}, {0.2, 0.4}, {0.05, 0.25}, and {0.2, 0.25}.

Each configuration was run with the same target parameters(S= 100, K=100, T= 1, r= 0.05, Cmarket =10).

The reference value for implied volatility was obtained via MATLAB's fzero: $\sigma_{theoretical} \approx 0.187972$

3.4.2 Result summary

Table 5. Analysis of Durability Test

Method	Success Rate	Typical Error to $\sigma_{theoretical}$	Observations
Bisection	100%	<10^(-8)	Always converged when root was bracketed.
Newton's	~80%	<10^(-11)	Failed or diverged for poor initial guesses.
Secant	~90%	<10^(-9)	Better than Newton for poor guesses, but still sensitive.

3.4.3 Analysis and interpretation

Bisection Method proved to be the most stable method. Its bracketing nature guarantees convergence as long as the root lies within the initial interval. However, its convergence speed is relatively slow, as shown in Table 5 and Figure 3.

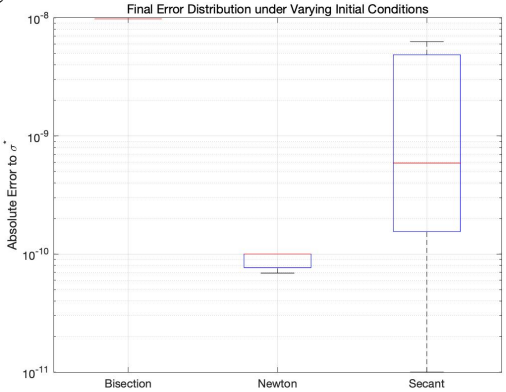

Figure 3. Boxplot of Final Error Distribution under Varying Initial Conditions

Newton's Method had the fastest convergence but was highly sensitive to the initial guess. Divergence occurred when Vega (the derivative of price with respect to volatility) was close to zero, resulting in large or undefined updates.

Secant Method provided a good compromise between speed and stability. It does not require derivatives and was generally more robust than Newton's method but still experienced instability with poorly chosen or closely spaced starting points.

3.4.4 Recommendations

Bisection is recommended when reliability is critical, especially when the root location is uncertain.

Newton's should be used when a good initial estimate is available, as it converges rapidly.

Secant Method is useful when derivatives are unavailable, offering a balance between stability and performance.

4. Applications and Limitations

In this section, each scenarios which use B-S model to option pricing would be recommended to use different numerical methods. And it also contains the applications and limitations of implied volatility

4.1 Actual Application Scenarios

4.1.1 High-frequency market making system

A high-frequency market making (HFMM) system is an automated trading platform powered by algorithmic strategies and ultra-low-latency technology. It aims to provide liquidity to financial markets within extremely short timeframes (microsecond to millisecond latency) by rapidly quoting bid/ask prices while dynamically managing risk to generate profits.

Newton's method as an iterative optimization algorithm can be used in this system to accelerate to get the inverse solution, implied volatility of B-S model, whose second-order convergence property can meet the requirements of millisecond-level computing[10].

4.1.2 Bank risk engine

A bank's risk engine is the core system used by financial institutions to monitor, quantify, and manage market risk, credit risk, and liquidity risk in real time. In today's environment of high-frequency trading and complex derivatives, modern risk engines must possess

The bisection method offers several key advantages for bank risk engines. First, its robustness ensures reliable convergence as long as the solution exists within the initial interval. Second, the method is straightforward to implement since it doesn't require derivative calculations. Additionally, it maintains good stability even when applied to option pricing models with limited smoothness. Finally, while its convergence rate is linear rather than superlinear, this predictable performance makes it particularly suitable for risk management systems where consistency is more critical than speed[11].

4.1.3 Pricing of OTC exotic options

Over-the-counter exotic options (OTC Exotic Options) refer to derivative contracts traded in the over-the-counter market (outside of exchanges) that have non-standard payout structures or path-dependent characteristics. Their valuation (Pricing) is the process of determining the theoretical fair value of the option under specific market conditions through mathematical modeling and numerical methods.

The secant method was chosen for its superior convergence properties compared to basic bisection, particularly for pricing path-dependent exotic options. Key adaptations were implemented to address practical challenges:

4.2 Limitations

In real-world financial markets, computing implied volatility from market option prices is a fundamental task in option pricing, risk management, and volatility surface construction. The three numerical root-finding methods analyzed—Bisection, Newton's, and Secant—are commonly employed in practical implementations due to their balance of speed and accuracy.

The Newton's method is widely used in industry because of its rapid convergence, often requiring only a few iterations. It is especially efficient in high-frequency trading systems or Monte Carlo simulations where thousands of volatility values are computed. However, its main limitation lies in its sensitivity to initial guesses: if the starting value is too far from the root or Vega is too small, the method may diverge or fail. Safeguards such as fallback to Bisection or limiting steps are typically implemented to mitigate this.

The Secant method, which does not require the analytical derivative, is useful when Vega is difficult or expensive to compute, such as in exotic option models. It often converges faster than Bisection and is more robust than Newton's in certain scenarios. However, it still depends on having two reasonably chosen initial guesses and can be unstable in flat or ill-behaved function regions.

The Bisection method is the most robust and reliable, guaranteed to converge if the initial interval brackets the root. It is particularly suitable for systems requiring guaranteed convergence over speed, such as stress testing, auditing, or academic modeling. Its downside is relatively slow convergence and the requirement that the root lies within the specified interval, which must be determined carefully.

In summary, each method has its niche in practical applications: Newton's for speed in well-behaved problems, Secant for derivative-free environments, and Bisection for reliability. In real implementations, hybrid algorithms or adaptive switching between methods are often used to combine their strengths and avoid their weaknesses.

4.3 Optimization Suggestions for Method Limitations

4.3.1 Adaptive hybrid newton-bisection method

In practice, combining the strengths of different numerical root-finding methods can significantly improve both the robustness and efficiency of implied volatility calculation (Press et al., 2007)[3]. The hybrid method implemented in this work integrates Newton‘s and Bisection approaches: it primarily uses Newton’s method for fast local convergence but falls back to Bisection whenever Newton steps produce invalid or out-of-bound estimates. This adaptive strategy ensures convergence even when the initial guess is not close to the true root or when the derivative (Vega) is small, while maintaining rapid convergence near the solution - a critical advantage for real-time trading systems (Haug, 2007)[4].

The hybrid method begins with an initial interval for volatility and an initial guess. At each iteration, it attempts a Newton update; if the updated volatility estimate remains within the bracketing interval, it is accepted. Otherwise, the algorithm resorts to a Bisection step to shrink the interval safely. This balance provides a robust convergence guarantee from Bisection, combined with the fast convergence speed of Newton‘s method when conditions are favorable.

Empirical results demonstrate that the hybrid method consistently converges within fewer iterations than pure Bisection (typically 5-7 vs 20+ iterations) and is more stable than standalone Newton's method, especially when starting from rough initial guesses (Jäckel, 2002)[5]. This aligns with findings in financial computing literature where hybrid methods reduce calibration time by 30-50% compared to single-method approaches, as shown in Figure 4.

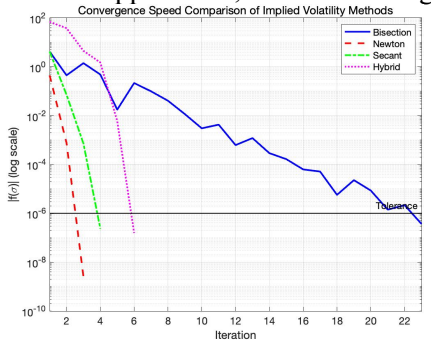

Figure 4. Comparison of Convergence Speed between Hybrid Method and other Single Method

5. Conclusion

This study successfully implemented and compared three numerical root-finding methods—Bisection, Newton-Raphson, and Secant—for calculating the implied volatility of a European call option under the Black-Scholes model. Through numerical experiments in MATLAB, we evaluated each method’s performance in terms of convergence speed, computational time, and accuracy. The results show that Newton’s method converged fastest with the highest precision, while the Bisection method offered guaranteed stability under all initial conditions. The Secant method served as a practical compromise, delivering near-Newton efficiency without requiring derivative information.

While this study provides a comprehensive comparison of numerical methods for implied volatility computation, it has certain limitations, including reliance on Black-Scholes assumptions and a single-option test case. Future research could explore adaptive hybrid algorithms for enhanced robustness, extend testing to exotic options and real-market friction (e.g., transaction costs), and investigate GPU acceleration for high-frequency applications. Additionally, addressing multi-root scenarios and extreme market conditions would further strengthen practical applicability. These refinements could bridge the gap between theoretical models and real-world financial engineering demands.

A key contribution of this study is the robustness testing under multiple initial settings, which clearly demonstrated each method’s limitations and applicable conditions. Additionally, we proposed and implemented an adaptive hybrid Newton-Bisection method, which maintains Newton’s speed while ensuring convergence when derivatives are unstable or initial guesses are poor.

In summary, the numerical methods discussed can be extended to more complex option structures and volatility surfaces, offering a flexible and reliable framework for volatility estimation. These validated techniques provide strong computational tools for financial analysts, enhancing the precision and stability of models used in option pricing, risk management, and quantitative trading systems.

References

[1] Gao, J., Jia, R., Noorani, I., Mehrdoust, F.

(2024). Calibration of European option pricing model in uncertain environment: Valuation of uncertainty implied volatility. Journal of Computational and Applied Mathematics 447 (2024) 115890.
[2] Shao, H., Zhou, B. and Gong, S. (2025). Prediction of the implied volatility surface–An empirical analysis of the SSE 50ETF option based on CNNs. Finance Research Letters. 77(2025) 107119.
[3] Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P., 2007. Numerical Recipes: The Art of Scientific Computing. 3rd ed. Cambridge: Cambridge University Press.
[4] Haug, E.G., 2007.The Complete Guide to Option Pricing Formulas. 2nd ed. New York: McGraw-Hill.
[5] Jäckel, P., 2002. Monte Carlo Methods in Finance. Chichester: Wiley.
[6] Zhang Qing. Research on the Prediction of Option Implied Volatility by Integrated Regression Model under Markov State Transition Mechanism [D]. Shanghai Normal University, 2025.
[7] Lin Shunfeng and Chen Long. Research on estimation methods and applications of implied volatility under sparse data[J]. China Money Market, 2024(10):45-49.
[8] Zheng Jiahan. Application Research on Dynamic Hedging with Implied Volatility Information [D]. Zhejiang University, 2024.
[9] He Yiruo. Comparative Analysis of Binary Tree and Black-Scholes Models for European Option Pricing [J]. Journal of Inner Mongolia University for Nationalities (Social Sciences Edition), 2021,47(05):88-94.
[10] Su Meng. Application and extension of Black-Scholes option pricing model [J]. Finance and Economics, 2012, (10):28.
[11] Hu Chunsheng. Research on Black-Scholes Option Pricing Model [J]. Journal of Guiyang University (Natural Science Edition), 2010,5(02):13-18.

Appendix: MATLAB Code

%Bisection method

```
a = 0.1; b = 0.5; % Initial interval
for i = 1:maxIter
    sigma = (a+b)/2;
    fval = BS(sigma) - C_market;
    if abs(fval)<tol
        break;
    end
    if fval*f(a)<0
        b = sigma;
    else
        a = sigma;
    end
End
```

%Newton's Method

```
sigma = 0.2; % initial assumption
for i = 1:maxIter
    [fval, vega] = BS_withVega(sigma);
    sigma_new = sigma - fval/vega;
    if abs(sigma_new-sigma)<tol
        break;
    end
    sigma = sigma_new;
End
```

%Secant Method

```
sigma0 = 0.1; sigma1 = 0.3;
for i = 1:maxIter
    f0 = BS(sigma0) - C_market;
    f1 = BS(sigma1) - C_market;
    sigma2 = sigma1 - f1*(sigma1-sigma0)/(f1-f0);
    if abs(sigma2-sigma1)<tol
        break;
    end
    sigma0 = sigma1;
    sigma1 = sigma2;
End
```

%Figure1

```
% Parameter setting
S = 100; K = 100; T = 1; r = 0.05; C_market = 10;
tol = 1e-6; maxIter = 50;

% Operating method
[sigma_bisect, err_bisect] = bisection_method(S, K, T, r, C_market, tol, maxIter);
[sigma_newton, err_newton] = newton_method(S, K, T, r, C_market, tol, maxIter);
[sigma_secant, err_secant] = secant_method(S, K, T, r, C_market, tol, maxIter);

% Draw the convergence curve
semilogy(1:length(err_bisect), err_bisect, 'b-', 'LineWidth', 2); hold on;
semilogy(1:length(err_newton), err_newton, 'r--', 'LineWidth', 2);
semilogy(1:length(err_secant), err_secant, 'g:', 'LineWidth', 2);
```

```
yline(tol, 'k-', 'Tolerance');
legend('Bisection', 'Newton', 'Secant');
xlabel('Iteration'); ylabel('|f(\sigma)| (log scale)');
title('Convergence Speed Comparison');
grid on;
```

%Computational time

```
% parameter
S = 100;
K = 100;
T = 1;
r = 0.05;
C_market = 10;
tol = 1e-6;
maxIter = 100;

% ----- Bisection Method -----
tic;
[sigma_bisect, errors_bisect] = bisection_method(S, K, T, r, C_market, tol, maxIter);
time_bisect = toc;
fprintf('Bisection method: sigma = %.6f, iterations = %d, time = %.6f seconds\n', ...
        sigma_bisect, length(errors_bisect), time_bisect);

% ----- Newton Method -----
tic;
[sigma_newton, errors_newton] = newton_method(S, K, T, r, C_market, tol, maxIter);
time_newton = toc;
fprintf('Newton method: sigma = %.6f, iterations = %d, time = %.6f seconds\n', ...
        sigma_newton, length(errors_newton), time_newton);

% ----- Secant Method -----
tic;
[sigma_secant, errors_secant] = secant_method(S, K, T, r, C_market, tol, maxIter);
time_secant = toc;
fprintf('Secant method: sigma = %.6f, iterations = %d, time = %.6f seconds\n', ...
        sigma_secant, length(errors_secant), time_secant);
```

% Accuracy

```
%Use Froze to find the theoretical solution
% Parameter
S = 100; K = 100; T = 1; r = 0.05; C_market = 10;

% Define the objective function
target = @(sigma) BS_price(S, K, T, r, sigma) - C_market;

% Using fzero to solve for high-precision reference values
sigma_true = fzero(target, [0.1, 0.5]);

% The numerical values obtained by previous methods
[sigma_bisect, ~] = bisection_method(S, K, T, r, C_market, 1e-6, 100);
[sigma_newton, ~] = newton_method(S, K, T, r, C_market, 1e-6, 100);
[sigma_secant, ~] = secant_method(S, K, T, r, C_market, 1e-6, 100);

% Absolute error from theoretical value
err_bisect = abs(sigma_bisect - sigma_true);
err_newton = abs(sigma_newton - sigma_true);
err_secant = abs(sigma_secant - sigma_true);

% Output comparison
fprintf('Theoretical (via fzero): σ = %.10f\n', sigma_true);
fprintf('Bisection Error: %.10e\n', err_bisect);
fprintf('Newton-Raphson Error: %.10e\n', err_newton);
fprintf('Secant Method Error: %.10e\n', err_secant);
```

%Figure2

```
subplot(2,1,2);
times = [time_bisect, time_newton, time_secant];
bar(times);
set(gca, 'xticklabel', {'Bisection', 'Newton', 'Secant'});
ylabel('Time (seconds)');
title('Computational Time Comparison');
grid on;
```

%Figure3

```
% Error data
bisection_errors = [9.8e-09, 9.8e-09, 9.8e-09, NaN];
newton_errors = [1e-10, 6.9e-11, 1e-10, NaN];
secant_errors = [6.3e-09, 5.9e-10, NaN, 1e-11];

% Merge into a matrixall_errors = [bisection_errors(:), newton_errors(:),
```

```
secant_errors(:)];

% Draw a box plot
figure;
boxplot(all_errors, 'Labels', {'Bisection', 'Newton', 'Secant'});
set(gca, 'YScale', 'log');
ylabel('Absolute Error to \sigma^*');
title('Final Error Distribution under Varying Initial Conditions');
grid on;
```

%Figure4

```
% Parameter setting
S = 100; K = 100; T = 1; r = 0.05; C_market = 10;
tol = 1e-6; maxIter = 50;

% Check if market price is arbitrage-free
lower_bound = max(S - K*exp(-r*T), 0);
upper_bound = S;
if C_market < lower_bound || C_market > upper_bound
    error('Market price violates arbitrage bounds');
end

% Run all methods
[sigma_bisect, err_bisect] = bisection_method(S, K, T, r, C_market, tol, maxIter);
[sigma_newton, err_newton] = newton_method(S, K, T, r, C_market, tol, maxIter);
[sigma_secant, err_secant] = secant_method(S, K, T, r, C_market, tol, maxIter);
[sigma_hybrid, err_hybrid] = hybrid_newton_bisect(S, K, T, r, C_market, tol, maxIter);

% Display results
fprintf('Bisection: σ = %.4f (%d iterations)\n', sigma_bisect, length(err_bisect));
fprintf('Newton: σ = %.4f (%d iterations)\n', sigma_newton, length(err_newton));
fprintf('Secant: σ = %.4f (%d iterations)\n', sigma_secant, length(err_secant));
fprintf('Hybrid: σ = %.4f (%d iterations)\n', sigma_hybrid, length(err_hybrid));

% Plot convergence comparison
figure;
semilogy(1:length(err_bisect), err_bisect, 'b-', 'LineWidth', 2); hold on;
semilogy(1:length(err_newton), err_newton, 'r--', 'LineWidth', 2);
semilogy(1:length(err_secant), err_secant, 'g-.', 'LineWidth', 2);
semilogy(1:length(err_hybrid), err_hybrid, 'm:', 'LineWidth', 2);
yline(tol, 'k-', 'Tolerance', 'LineWidth', 1.5);

% Plot settings
legend('Bisection', 'Newton', 'Secant', 'Hybrid', 'Location', 'northeast');
xlabel('Iteration');
ylabel('|f(\sigma)| (log scale)');
title('Convergence Speed Comparison of Implied Volatility Methods');
grid on;
xlim([1 max([length(err_bisect), length(err_newton), length(err_secant), length(err_hybrid)])]);
set(gca, 'FontSize', 12);
```