ACCELERATED PORTFOLIO OPTIMIZATION AND OPTION PRICING WITH REINFORCEMENT LEARNING

Hadi Keramati, Samaneh Jazayeri

Magnative AI {keramati, samanehj}@magnative.ai

ABSTRACT

We present a reinforcement learning (RL)-driven framework for optimizing block-preconditioner sizes in iterative solvers used in portfolio optimization and option pricing. The covariance matrix in portfolio optimization or the discretization of differential operators in option pricing models lead to large linear systems of the form $\mathbf{A}\mathbf{x} = \mathbf{b}$. Direct inversion of high-dimensional portfolio or fine-grid option pricing incurs a significant computational cost. Therefore, iterative methods are usually used for portfolios in real-world situations. Ill-conditioned systems, however, suffer from slow convergence. Traditional preconditioning techniques often require problem-specific parameter tuning. To overcome this limitation, we rely on RL to dynamically adjust the block-preconditioner sizes and accelerate iterative solver convergence. Evaluations on a suite of real-world portfolio optimization matrices demonstrate that our RL framework can be used to adjust preconditioning and significantly accelerate convergence and reduce computational cost. The proposed accelerated solver supports faster decision-making in dynamic portfolio allocation and real-time option pricing.

1 Introduction

Iterative methods are popular methods for portfolio optimization and option pricing Gulliksson & Mazur (2020). Option contracts are widely traded and are an inherently asymmetric opportunity that requires price valuation. The first approach for using the discretized Brownian motion for option pricing was suggested by Bachelier (1900). Black & Scholes (1973) suggested a partial differential equation to assess a fair price for European option contracts to consider the time and risk associated with the underlying asset. The Black-Scholes formula and some of the later suggested models have closed form solutions, but numerical methods provide solutions across the discretized time domain and are in most cases faster than a closed form formula in multi-dimension space. Many of these models, such as jump-diffusion models, consider discontinuous moves in the market in addition to the continuous Brownian motion which can be solved by iterative numerical solvers BALDINA; Merton (1976). Large portfolio optimization (e.g. Markowitz mean variance optimization) is also a linear system with a form of Ax = b where A is the covariance matrix of asset returns, x are the portfolio weights for each asset, and \mathbf{b} is a constraint vector ensuring a fully invested portfolio Bajeux-Besnainou et al. (2012); Gilli et al. (2019). Directly inverting the matrix A to solve this portfolio optimization problem is extremely expensive, which is why iterative methods are used to compute the vector \mathbf{x} .

Solving these linear systems of equations that govern portfolio optimization and option pricing with minimal computational overhead is crucial for successful deployment because it involves real-time decision making Gaikwad & Toke (2009). For example, portfolio managers must update the weights of each asset or asset class to reduce risk and increase return. However, when the portfolio includes a large number of assets, the covariance matrices become large. Option strategies also require real-time updates in option price valuation, but the corresponding matrices after discretization, mainly using finite difference with fine meshes, become large and time-consuming to solve Ikonen & Toivanen (2008); Korn & Korn (2001). Solving ill-conditioned problems, particularly in multiasset systems where each asset imposes a specific structure, results in slow convergence. Using preconditioners to accelerate the solvers, particularly for ill-conditioned matrices, is necessary, and block precon-

ditioners take advantage of the system's structure such as asset classes and boundary conditions to accelerate the convergence of the iterative solvers.

Iterative methods for solving Partial differential equations (PDEs) struggle with nonsymmetric or poorly conditioned problems Greenbaum et al. (1996). Flexible generalized minimal residual method (FGMRES) is a method that works based on the Krylov subspace and minimizes the residual in this subspace, which is particularly effective for large-scale, sparse problems arising from PDE discretization Saad & Schultz (1986); Saad (1993). FGMRES adds flexibility to adapt more effectively to problems where the preconditioner might change or require adjustment during iterations. Preconditioning is a crucial step in iterative methods, as it transforms the original problem into a form that allows for faster convergence. Common preconditioning strategies include techniques like Incomplete LU (ILU) factorization, multigrid methods, and domain decomposition Benzi (2002). The block-partitioned preconditioner, which divides the matrix into blocks and applies preconditioning within each block, has shown significant promise in improving the efficiency of both GM-RES and FGMRES Saad (2003). In recent years, machine learning approaches have emerged as a means to further optimize iterative solvers. Chen (2024) used a graph neural network to predict preconditioners for FGMRES. Reinforcement learning (RL), in particular, has shown potential in dynamically tuning solver parameters for improved performance. For example, Han et al. (2018) demonstrated the use of RL to optimize parameters in iterative solvers, leading to faster convergence. Proximal Policy Optimization (PPO), a popular RL algorithm, has been successfully applied to adapt preconditioning thresholds in real-time, balancing computational cost with convergence efficiency Schulman et al. (2017).

This paper explores the integration of RL and the block-partitioned preconditioner for portfolio optimization and option pricing problems. Using the dynamic adjustment capabilities of PPO, we train an agent to adjust the size of the preconditioner block during the iterative process to enhance the speed of the solver to achieve an accelerated real-time solver that is effective for portfolio weight adjustment and option pricing over time.

2 Preliminaries

2.1 Portfolio Optimization

In mean-variance portfolio optimization, the portfolio weight vector $\mathbf{x} \in \mathbb{R}^n$ minimizes portfolio variance while meeting a specified target return and ensuring full investment. This constrained-based optimization is stated as follows.

$$\begin{aligned} & \underset{\mathbf{x}}{\min} & \frac{1}{2} \mathbf{x}^T \Sigma \mathbf{x}, \\ & \text{s.t.} & \boldsymbol{\mu}^T \mathbf{x} = R_{\text{target}}, \\ & \mathbf{e}^T \mathbf{x} = 1 \end{aligned} \tag{1}$$

where $\Sigma \in \mathbb{R}^{n \times n}$ is the covariance matrix of asset returns, $\mu \in \mathbb{R}^n$ is the vector of expected returns, $\mathbf{e} \in \mathbb{R}^n$ is the vector of ones, and R_{target} is the desired portfolio return. To incorporate the constraints into the optimization framework, we form the Lagrangian.

$$\mathcal{L}(\mathbf{x}, \lambda_1, \lambda_2) = \frac{1}{2} \mathbf{x}^T \Sigma \mathbf{x} - \lambda_1 (\mathbf{e}^T \mathbf{x} - 1) - \lambda_2 (\boldsymbol{\mu}^T \mathbf{x} - R_{\text{target}}),$$
 (2)

where λ_1 and λ_2 are Lagrange multipliers. Taking the derivatives with respect to \mathbf{x} , λ_1 , and λ_2 and setting them equal to zero yields the first-order optimality conditions:

$$\nabla_{\mathbf{x}} \mathcal{L} : \quad \Sigma \mathbf{x} - \lambda_1 \mathbf{e} - \lambda_2 \boldsymbol{\mu} = 0,$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_1} : \quad \mathbf{e}^T \mathbf{x} - 1 = 0,$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_2} : \quad \boldsymbol{\mu}^T \mathbf{x} - R_{\text{target}} = 0.$$
(3)

These equations constitute a system of linear equations with unknown vector.

$$\mathbf{y} = \begin{pmatrix} \mathbf{x} \\ \lambda_1 \\ \lambda_2 \end{pmatrix}$$

Thus, the Karush-Kuhn-Tucker (KKT) conditions can be expressed in the compact form

$$\mathbf{A}\mathbf{v} = \mathbf{b} \tag{4}$$

where the coefficient matrix A and the vector on the right side b are defined by the matrices below.

$$\mathbf{A} = \begin{pmatrix} \Sigma & \mathbf{e} & \boldsymbol{\mu} \\ \mathbf{e}^T & 0 & 0 \\ \boldsymbol{\mu}^T & 0 & 0 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 0 \\ 1 \\ R_{\text{target}} \end{pmatrix}$$

This formulation is a linear system that can be solved using iterative solvers to efficiently find optimal portfolio weights \mathbf{x} in large-scale problems.

2.2 OPTION PRICING USING FINITE DIFFERENCE METHOD

In this section, we briefly review the mathematical foundations of option pricing Soleymani & Akgül (2019); Kovalov et al. (2007). We derive a finite difference discretization that transforms the Black–Scholes PDE into a system of linear equations. We then detail the implicit finite difference scheme that produces a linear system at each time step. For a European option, the value of an option at its expiration is as follows.

$$V(S,T) = \max(S - K, 0),\tag{5}$$

where S denotes the asset price, K is the strike price, and T is the time to expiry. Under the Black–Scholes framework, the option price V(S,t) satisfies the equation 6 Chen (2017).

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS \frac{\partial V}{\partial S} - rV = 0, \tag{6}$$

where σ is the volatility and r is the risk-free interest rate.

To numerically solve equation 6, we discretize both the spatial and temporal domains. We divide the interval $[0,S_{\max}]$ that S_{\max} is the upper bound of the asset price into M subintervals of uniform length, $\Delta S = \frac{S_{\max}}{M}$. Therefore, each point in the grid is given by $S_i = i \, \Delta S, \quad i = 0,1,\ldots,M$. The time interval [0,T] is divided into N equal steps of size of $\Delta t = \frac{T}{N}$ with intervals $t_n = n \, \Delta t, \quad n = 0,1,\ldots,N$. For any interior cell we use the following backward approximations Zhao et al. (2007).

$$\frac{\partial V}{\partial t} \approx \frac{V_i^n - V_i^{n-1}}{\Delta t},\tag{7}$$

where $V_i^n \approx V(S_i, t_n)$ and $V_i^{n-1} \approx V(S_i, t_{n-1})$.

The first spatial derivative is approximated by the following difference.

$$\frac{\partial V}{\partial S} \approx \frac{V_{i+1}^{n-1} - V_{i-1}^{n-1}}{2\Lambda S}.$$
 (8)

Similarly, the second spatial derivative is approximated by

$$\frac{\partial^2 V}{\partial S^2} \approx \frac{V_{i+1}^{n-1} - 2V_i^{n-1} + V_{i-1}^{n-1}}{\Delta S^2}.$$
 (9)

We can substitute the above finite difference approximations into the Black-Scholes PDE evaluated at t_{n-1} .

$$\frac{V_i^n - V_i^{n-1}}{\Delta t} + \frac{1}{2}\sigma^2 S_i^2 \frac{V_{i+1}^{n-1} - 2V_i^{n-1} + V_{i-1}^{n-1}}{\Delta S^2} + rS_i \frac{V_{i+1}^{n-1} - V_{i-1}^{n-1}}{2\Delta S} - rV_i^{n-1} = 0$$
 (10)

Rearranging terms for the unknowns results in the following.

$$-\alpha_i V_{i-1}^{n-1} + B_i V_i^{n-1} - \gamma_i V_{i+1}^{n-1} = V_i^n$$
(11)

with the coefficients defined by Equations 12, 13, and 14.

$$\alpha_i = \frac{\Delta t}{2} \left(\frac{\sigma^2 S_i^2}{\Delta S^2} - \frac{r S_i}{\Delta S} \right) \tag{12}$$

$$\gamma_i = \frac{\Delta t}{2} \left(\frac{\sigma^2 S_i^2}{\Delta S^2} + \frac{r S_i}{\Delta S} \right) \tag{13}$$

$$B_i = 1 + \Delta t \left(\frac{\sigma^2 S_i^2}{\Delta S^2} + r \right) \tag{14}$$

Rewriting in matrix terms yields the linear system.

$$\mathbf{A}\,\mathbf{V}^{n-1} = \mathbf{V}^n \tag{15}$$

where

$$\mathbf{V}^{n-1} = \begin{pmatrix} V_1^{n-1} \\ V_2^{n-1} \\ \vdots \\ V_{M-1}^{n-1} \end{pmatrix}$$

and the coefficient matrix **A** is tridiagonal with entries $A_{i,i-1} = -\alpha_i$, $A_{i,i} = B_i$, $A_{i,i+1} = -\gamma_i$, (i = 1, ..., M-1). The boundary conditions at S_0 and S_M , similar to any linear system, are the vector on the right side.

2.3 KRYLOV SUBSPACE SOLVERS

In this section, we show how linear systems are solved using iterative solvers. We rely on the GMRES algorithm that begins by selecting an initial vector v_1 and then uses the Arnoldi process to generate an orthonormal basis $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k\}$ and an upper Hessenberg matrix \mathbf{H}_k . This process reduces the problem to a least-squares problem that minimizes the residual norm $\|\mathbf{A}\mathbf{x}_k - \mathbf{b}\|$. The approximate solution is given by $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k \mathbf{y}_k$, where \mathbf{V}_k is the matrix of basis vectors, and \mathbf{y}_k solves the least squares problem involving \mathbf{H}_k . The Arnoldi process initializes with $\mathbf{v}_1 = \frac{\mathbf{r}_0}{\|\mathbf{r}_0\|}$, and for each $j = 1, 2, \dots, k$, $\mathbf{w} = \mathbf{A}\mathbf{v}_j$ is computed. The coefficients $h_{ij} = \mathbf{v}_i^T \mathbf{w}$ are calculated for $i = 1, 2, \dots, j$, and the vector \mathbf{w} is updated as $\mathbf{w} = \mathbf{w} - \sum_{i=1}^{j} h_{ij} \mathbf{v}_i$. Finally, the process sets $h_{j+1,j} = \|\mathbf{w}\|$ and normalizes the vector $\mathbf{v}_{j+1} = \frac{\mathbf{w}}{h_{j+1,j}}$ to update the solution approximation of \mathbf{x} [Trefethen & III (2022)].

3 METHODOLOGY

3.1 PROBLEM FORMULATION

The objective of the proposed algorithm is to solve linear systems of equations arising from portfolio optimization or option pricing which are in the form of $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the known matrix which is usually a sparse matrix and $\mathbf{b} \in \mathbb{R}^n$ is a vector. The goal is to find a solution $\mathbf{x} \in \mathbb{R}^n$ that minimizes the residual $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$. To improve the convergence rate of the iterative solver, the linear system is preconditioned using the block preconditioner. The matrix \mathbf{A} is split into smaller submatrices of size $k \times k$, resulting in an approximate block diagonal matrix \mathbf{M} :

$$\mathbf{M} = \operatorname{diag}(\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_k) \tag{16}$$

where $\mathbf{A}_i \in \mathbb{R}^{s \times s}$ are the blocks of \mathbf{A} .

3.2 BLOCK PRECONDITIONER

Since we are changing the size of blocks of preconditioner in each iteration, we use Flexible GMRES (FGMRES). The Block preconditioner for FGMRES method partitions the matrix ${\bf A}$ into smaller blocks and applies the preconditioner. This approach is particularly effective for handling large and structured linear systems by improving the condition number of the system matrix, which accelerates the convergence of the iterative solver. In detail, the matrix ${\bf A}$ is divided into $k \times k$ blocks. For each block, the preconditioning matrix ${\bf M}$ is constructed by performing QR decomposition, which decomposes the block into an orthogonal matrix ${\bf Q}$ and an upper triangular matrix ${\bf R}$. Specifically, for a block ${\bf A}_i$ of ${\bf A}$, the QR decomposition is given by ${\bf A}_i = {\bf Q}_i {\bf R}_i$. The preconditioner ${\bf M}$ for the entire matrix ${\bf A}$ is then assembled using these decomposed blocks. During the FGMRES iterations, this preconditioner is applied to each block adaptively based on the current residuals. The application of the preconditioner improves the convergence rate by reducing the effective condition number of the system, thus making the iterative process more efficient.

$$\mathbf{M}_{\mathrm{block}} \approx \mathbf{Q}\mathbf{R}$$
 (17)

where \mathbf{Q} and \mathbf{R} are obtained from the QR decomposition of the corresponding block $\mathbf{A}_{\mathrm{block}}$.

3.3 REINFORCEMENT LEARNING FOR PRECONDITIONING

A reinforcement learning (RL) agent is used to adjust the preconditioning process. The RL agent is defined by the current residual vector \mathbf{r} as the state, the adjustment of the block size for updating the preconditioner as the action, and the negative residual norm as the reward to encourage a reduction in the residual. The Proximal Policy Optimization (PPO) algorithm is used to train the RL agent to optimize the preconditioning process. As mentioned in the previous section, the GMRES algorithm iteratively refines the solution \mathbf{x} as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{V}_k \mathbf{y}_k \tag{18}$$

where V_k are the orthonormal basis vectors generated from the Arnoldi process, and y_k is obtained by solving the least-squares problem:

$$\mathbf{H}_k \mathbf{y}_k = \beta \mathbf{e}_1 \tag{19}$$

with \mathbf{H}_k being the upper Hessenberg matrix from the Arnoldi process, β the initial residual norm, and \mathbf{e}_1 the first standard basis vector.

The adaptive preconditioning guided by the RL agent is applied within each GMRES iteration to accelerate convergence. The algorithm terminates when the target residual tolerance is achieved or a maximum number of iterations is reached. The PPO-based block-preconditioner for the FGMRES method improves the traditional approach by integrating a Proximal Policy Optimization (PPO) agent that adjusts the block size k for the preconditioner and enables it to exploit the underlying structure of the matrix $\bf A$ to reduce its condition number. At each iteration, the agent observes the residuals of the linear system and, using its current policy, determines an optimal block size to apply the block-preconditioner. This integer value, which serves as the action taken by the RL agent, dictates how the preconditioner is applied, allowing the method to adapt to the varying structure of the matrix $\bf A$ throughout the iterations. Over multiple episodes, the PPO agent learns a policy that effectively configures the preconditioning process to optimize the overall convergence rate. The pseudocode for the framework is presented in Algorithm 1.

During initialization, the sparse matrix $\bf A$ and the right-hand side vector $\bf b$ are set up, representing the system $\bf Ax = \bf b$. Parameters such as maximum block size, tolerance, iteration limits, and restart frequency are configured. A PPO agent is initialized with the appropriate state and action dimensions, and the number of training episodes is defined. In the training phase, the agent is trained over long episodes to reach the convergence threshold. In each episode, the environment is reset to an initial state and the agent interacts with the environment by selecting actions to adjust the size of the preconditioning block. The PPO agent is trained to use these experiences to refine its value and policy functions. The same process applies to the inference phase, where the preconditioners are

Algorithm 1 Block-partitioned portfolio optimization and option pricing with PPO

```
1: Input: Matrix A, vector b, and convergence threshold
2: Initialize PPO agent
3: for each episode do
       Reset environment to \mathbf{x}_k
4:
5:
       while residual < threshold do
6:
           PPO agent choose block size
           Initialize preconditioners with new block size and QR decomposition
7:
           Initialize V and H for Arnoldi iteration
8:
9:
           for j in range restart do
10:
               Apply preconditioner to V[:, j] using PPO adjusted block size
               Update H and V via Arnoldi iteration
11:
12:
               Solve least squares for y
               Update x and compute residual
13:
           end for
14:
15:
           Store transition and update state
16:
       end while
17: end for
```

initialized by using QR decomposition for each block of the matrix $\bf A$. The initial guess ${\bf x}_0$ is set to zero, and the initial residual ${\bf r}_0$ is calculated. During Arnoldi iteration, the preconditioner is applied to the basis vectors $\bf V$. The Arnoldi process updates the matrices $\bf H$ and $\bf V$, and a least squares problem is solved to update $\bf x$. Residuals are computed and compared with the convergence criteria, allowing for early termination if the criteria are satisfied. Finally, we return the residuals for each iteration of the PPO-based FGMRES, which can be used to evaluate convergence performance.

4 Discussion

The results of the portfolio optimization matrices Davis & Hu (2011) are shown in Figures 1, 2, and 3. These real-world portfolio optimization problems are identified by their titles in the matrix collection data, facilitating reproducibility Davis & Hu (2011). Figure 1 presents the results for a portfolio optimization problem with a matrix of size 4008 that contains 8,188 non-zero elements. In Figure 2, the displayed curve corresponds to a covariance matrix of size 16,955 with 37,849 non-zero elements, which is denser than the matrix shown in Figure 1. Figure 3 compares the performance of the proposed RL framework with that of a constant block size method for a portfolio whose covariance matrix is of size 33,833 and comprises 73,249 non-zero elements. As mentioned earlier, these matrices are non-symmetric. It can be observed that the pre-trained PPO solver with an adaptive block size converges faster than the solver using a constant block size preconditioner across different matrix sizes. In the PPO-based solver, the block size is limited to the set of integer values considered for the constant block size method, ensuring a fair comparison.

Figure 4 presents the convergence behavior of option pricing systems using matrices of size 1000 with varying densities. These matrices are generated from synthetic data with volatilities ranging from 5% to 25% and a risk-free rate of 1%. Each matrix is constructed such that its density, defined as the ratio of nonzero elements to the total number of elements, remains constant. Figure 5 illustrates the convergence of the option pricing system for a coefficient matrix **A** of size 2000 at two different densities. In both cases, a clear speedup in obtaining the solution and achieving low residual values is observed across the different matrix sizes and densities.

Although the computational cost of training the RL agent is nontrivial, the results demonstrate that using an RL agent can significantly reduce the number of iterations required to solve an option pricing problem, in some cases, to as few as two iterations. This reduction in iterations is particularly promising for real-time option pricing applications.

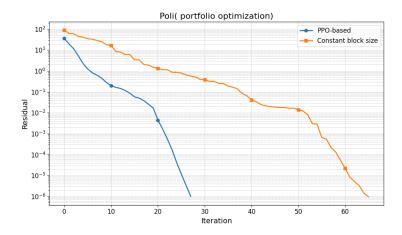


Figure 1: Convergence plots of PPO-based and constant block size for portfolio optimization of the size of 4008

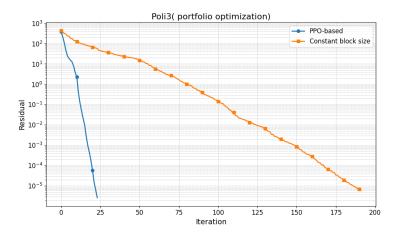


Figure 2: Convergence plots of PPO-based and constant block size for portfolio optimization of the size of 16,955

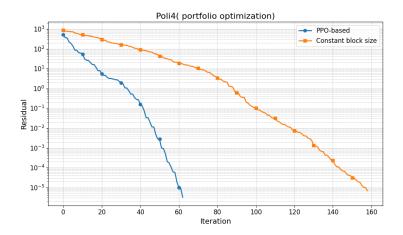


Figure 3: Convergence plots comparing PPO-optimized and constant block-size preconditioning for a portfolio optimization problem with a matrix of size 33,833.

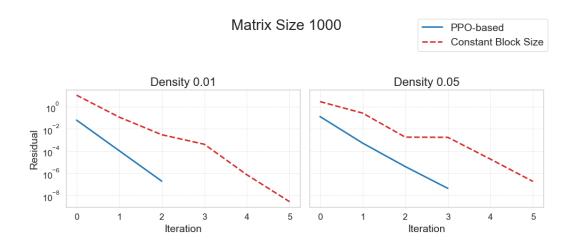


Figure 4: Convergence plots of option pricing for PPO-optimized and constant block preconditioners for a matrix size of 1000

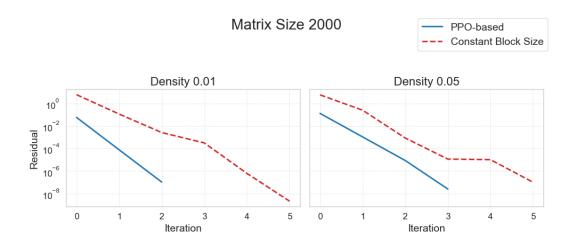


Figure 5: Convergence plots of option pricing for PPO-optimized and constant block preconditioners for a matrix size of 2000

5 CONCLUSION

This paper presents a reinforcement learning framework designed to accelerate the solution of portfolio optimization and option pricing problems formulated as linear systems of equations. The framework constructs a partitioned preconditioner for the FGMRES algorithm and employs an RL agent to learn the optimal block size, thereby enhancing the convergence speed of the solver. Experiments demonstrate accelerated convergence across various matrix sizes and densities for both portfolio optimization and option pricing. These results indicate that the proposed method is especially promising for non-symmetric and ill-conditioned asset models.

REFERENCES

Louis Bachelier. Théorie de la spéculation. In *Annales scientifiques de l'École normale supérieure*, volume 17, pp. 21–86, 1900.

- Isabelle Bajeux-Besnainou, Wachindra Bandara, and Efstathia Bura. A krylov subspace approach to large portfolio optimization. *Journal of Economic Dynamics and Control*, 36(11):1688–1699, 2012.
- ANDREA BALDINA. Iterative methods for option pricing in merton's jump diffusion model.
- Michele Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182:418–477, 2002.
- Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654, 1973.
- Jie Chen. Graph neural preconditioners for iterative solutions of sparse linear systems. *arXiv* preprint *arXiv*:2406.00809, 2024.
- Yuwei Chen. Numerical methods for pricing multi-asset options. *University of Toronto, Toronto*, pp. 1–75, 2017.
- Timothy A Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–25, 2011.
- Abhijeet Gaikwad and Ioane Muni Toke. Gpu based sparse grid technique for solving multidimensional options pricing pdes. In *Proceedings of the 2nd workshop on high performance computational finance*, pp. 1–9, 2009.
- Manfred Gilli, Dietmar Maringer, and Enrico Schumann. Numerical methods and optimization in finance. Academic Press, 2019.
- Anne Greenbaum, Vlastimil Pták, and Zdenv e k Strakoš. Any nonincreasing convergence curve is possible for gmres. SIAM Journal on Matrix Analysis and Applications, 17(3):465–469, 1996.
- Mårten Gulliksson and Stepan Mazur. An iterative approach to ill-conditioned optimal portfolio selection. *Computational Economics*, 56(4):773–794, 2020.
- Y. Han, X. Li, and W. E. A machine learning framework for solving high-dimensional mean field game and mean field control problems. *Proceedings of the National Academy of Sciences*, 115: 12757–12762, 2018.
- Samuli Ikonen and Jari Toivanen. Efficient numerical methods for pricing american options under stochastic volatility. *Numerical Methods for Partial Differential Equations: An International Journal*, 24(1):104–126, 2008.
- Ralf Korn and Elke Korn. Option pricing and portfolio optimization: modern methods of financial mathematics, volume 31. American Mathematical Soc., 2001.
- Pavlo Kovalov, Vadim Linetsky, and Michael Marcozzi. Pricing multi-asset american options: A finite element method-of-lines with smooth penalty. *Journal of Scientific Computing*, 33(3):209– 237, 2007.
- Robert C Merton. Option pricing when underlying stock returns are discontinuous. *Journal of financial economics*, 3(1-2):125–144, 1976.
- Youcef Saad. A flexible inner-outer preconditioned gmres algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993.
- Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.
- Yousef Saad. Iterative Methods for Sparse Linear Systems. SIAM, 2003.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Fazlollah Soleymani and Ali Akgül. Improved numerical solution of multi-asset option pricing problem: A localized rbf-fd approach. *Chaos, Solitons & Fractals*, 119:298–309, 2019.

L. N. Trefethen and David Bau III. Numerical Linear Algebra. SIAM, 2022.

Jichao Zhao, Matt Davison, and Robert M Corless. Compact finite difference method for american option pricing. *Journal of Computational and Applied Mathematics*, 206(1):306–321, 2007.