Analytical and Numerical Solutions for Nonlinear Equations
©Available online at https://ansne.du.ac.ir/
Online ISSN: 3060–785X
2027, Volume 12, Issue 1, pp. 109–139
Research article

Option Pricing: A New MCGP Approach

Fatemeh Ajam[1], Alireza Bahiraie[1,*]

[1] Department of Mathematics, Faculty of Mathematics, Statistics and Computer Sciences, Semnan University, Iran

*** Corresponding author(s):** Alireza.bahiraie@semnan.ac.ir

Received: 15/08/2026 **Revised:** 14/09/2026 **Accepted:** 28/09/2026 **Published:** 02/11/2026 10.22128/ansne.2026.3427.1233

Abstract

Pricing American-style options remains a challenging problem in computational finance due to the presence of early-exercise features and the absence of closed-form valuation formulas. Among simulation-based methods, the Least Squares Monte Carlo (LSMC) approach has become one of the most widely adopted techniques for approximating continuation values in optimal stopping problems. However, the accuracy of LSMC is highly dependent on the specification of predefined regression basis functions, which may introduce approximation bias when continuation-value dynamics exhibit strong nonlinearities. This paper proposes a Monte Carlo–Genetic Programming (MCGP) framework for American option pricing. The proposed methodology combines Monte Carlo simulation for generating risk-neutral asset-price trajectories with Genetic Programming-based symbolic regression for adaptive continuation-value estimation. Unlike conventional LSMC implementations, the proposed framework does not rely on predefined polynomial basis functions. Instead, continuation-value functions are evolved directly from simulated data through an evolutionary symbolic-learning process, enabling greater structural flexibility and potentially reducing projection bias associated with a restricted predefined basis. Numerical experiments conducted under different volatility and interest-rate regimes yield lower descriptive MAE and RMSE point estimates for MCGP than for the conventional LSMC benchmark. A controlled cross-family benchmark additionally compares MCGP with cubic-basis LSMC, neural-network continuation regression, and Gaussian kernel ridge regression on common Monte Carlo paths. In the resulting 10-replication benchmark, MCGP has the lowest descriptive MAE and RMSE at both 20% and 40% volatility, although paired comparisons with LSMC and the neural benchmark are not statistically significant in this small sample. A separate 50-replication matched MCGP–LSMC diagnostic likewise does not reject equality of absolute-error performance at the 5% level. The results therefore support adaptive symbolic regression as an interpretable and competitively accurate alternative rather than establishing universal statistical superiority. A controlled runtime diagnostic additionally confirms that the evolutionary search is substantially more computationally expensive than the small-basis LSMC regression in the present reproducible implementation; MCGP is therefore not claimed to provide a general speed advantage.

Keywords: American option pricing, Monte Carlo simulation, Genetic programming, Symbolic regression, Continuation-value approximation, Computational finance.

Mathematics Subject Classification (2020): 91G20, 91G60, 65C05, 68W50

COPYRIGHTS: ©2027, Analytical and Numerical Solutions for Nonlinear Equations published by Damghan University.
This is an open access article published under the *CC BY 4 DEED* license

1 Introduction

Derivative securities constitute one of the fundamental components of modern financial markets and play a central role in portfolio management, hedging, speculative trading, and risk transfer under uncertainty [13]. Among financial derivatives, options are particularly important because of their flexibility and extensive applications in equity, commodity, foreign-exchange, and interest-rate markets. Consequently, the accurate valuation of option contracts has become a major research topic in both theoretical and computational finance. Since the seminal work of Black and Scholes [4] and the subsequent theoretical developments of Merton [5], numerous analytical and numerical methodologies have been developed to address increasingly complex derivative-pricing problems.

Among existing computational approaches, Monte Carlo simulation has emerged as one of the most flexible and widely used methods in quantitative finance because of its ability to model stochastic dynamics under complex market environments [6, 14]. By generating large ensembles of stochastic asset-price trajectories under the risk-neutral measure, Monte Carlo methods can effectively handle nonlinear payoffs, path dependencies, and multidimensional state variables that are difficult to address using lattice-based or finite-difference techniques [6, 14]. These advantages have made simulation-based methods particularly attractive for pricing exotic derivatives and high-dimensional contingent claims [8, 25].

A major advancement in this area was introduced through the Least Squares Monte Carlo (LSMC) method proposed by Longstaff and Schwartz [7]. The LSMC framework combines Monte Carlo simulation with least-squares regression to estimate continuation values during backward induction. At each exercise date, discounted future payoffs are projected onto a predefined set of basic functions, enabling numerical approximation of the optimal exercise strategy. Owing to its simplicity, scalability, and computational efficiency, LSMC has become one of the most influential approaches for American option valuation in both academic research and financial practice [7, 10, 11].

Despite its widespread adoption, the performance of the LSMC method remains highly sensitive to the specification of regression basis functions used to approximate continuation values. In practice, polynomial basis expansions may fail to accurately capture complex nonlinear exercise boundaries, particularly under high-volatility conditions or in problems involving intricate payoff structures. Moreover, inappropriate basis selection may introduce approximation bias and reduce pricing accuracy. From a function-approximation perspective, this limitation may be interpreted as projection bias arising from the restriction of continuation-value estimation to a finite-dimensional basis-function space [10,11]. Although several machine-learning and nonparametric alternatives have been proposed, developing flexible and interpretable continuation-value approximation techniques remains an active research challenge in simulation-based option pricing [10, 11, 22, 30, 31].

Recent advances in machine learning and evolutionary computation have provided alternative approaches for addressing complex approximation problems in computational finance [22, 26, 30, 31]. Among these techniques, Genetic Programming (GP) offers a particularly attractive framework because of its ability to construct nonlinear symbolic relationships directly from data without requiring predefined functional forms [16–18]. In GP, candidate solutions are represented as symbolic expression trees composed of mathematical operators and terminal variables, which evolve iteratively through evolutionary mechanisms such as selection, crossover, and mutation.

Motivated by these limitations, this study proposes a novel Monte Carlo–Genetic Programming (MCGP) framework for American option pricing. In the proposed methodology, Monte Carlo simulation is employed to generate risk-neutral asset-price trajectories, while Genetic Programming is utilized to estimate continuation-value functions through evolutionary symbolic regression. Unlike conventional LSMC implementations that rely on predefined polynomial basis functions, the proposed framework constructs adaptive symbolic representations directly from simulated data. This data-driven approach provides greater flexibility in capturing complex nonlinear continuation-value relationships, reduces dependence on subjective basis-function selection, and may mitigate projection bias associated with restricted regression structures [7, 10, 11].

Importantly, the methodological contribution of the proposed framework extends beyond substituting Genetic Programming for polynomial regression. Within MCGP, symbolic continuation-value functions are evolved recursively at each exercise date using discounted future payoffs obtained from Monte Carlo simulation, and the resulting expressions are embedded directly into the optimal stopping rule. Consequently, the framework integrates stochastic path generation, adaptive structural discovery of continuation functions, and exercise-policy determination within a unified backward-induction procedure. This architecture distinguishes MCGP from conventional fixed-basis LSMC models, black-box neural approximators, and stand-alone symbolic-regression applications [7, 22, 28].

To evaluate the effectiveness of the proposed methodology, a comprehensive set of numerical experiments is conducted under various volatility and interest-rate scenarios. The performance of MCGP is assessed against conventional LSMC and, in a controlled cross-family

benchmark, against neural-network continuation regression and Gaussian kernel ridge regression. The archived scenario tables yield lower descriptive pricing-error point estimates for MCGP than LSMC. In the newly added 10-replication cross-family benchmark, MCGP also has the lowest descriptive MAE and RMSE at both 20% and 40% volatility, while paired comparisons with LSMC and the neural benchmark remain statistically non-significant in the small matched sample. A separate 50-replication MCGP–LSMC experiment likewise does not establish a significant baseline difference at the 5% level. A reproducible timing experiment further shows that the evolutionary symbolic search is substantially more expensive than the small-basis LSMC regression in the present implementation. The contribution of MCGP is therefore interpreted primarily in terms of adaptive symbolic structure, interpretability, and competitive approximation performance rather than universal statistical or computational superiority.

The remainder of this paper is organized as follows. The next section introduces the theoretical background and methodological foundations underlying Monte Carlo simulation, Genetic Programming, and the proposed Monte Carlo–Genetic Programming (MCGP) framework. Subsequently, the experimental design, numerical results, and comparative performance analyses are presented and discussed. Finally, the paper concludes with a summary of the main findings, limitations of the study, and potential directions for future research.

2 Literature Review

Option pricing has been a central research topic in financial mathematics since the development of modern arbitrage-based pricing theory. The seminal Black–Scholes framework introduced by Black and Scholes [4], and extended by Merton [5], established the foundation for derivative valuation under continuous-time stochastic processes. Although this framework yields closed-form solutions for European-style options under restrictive assumptions, its applicability is limited in the presence of early-exercise features, stochastic volatility, and path-dependent payoffs [13, 12]. Classical numerical methods such as finite-difference schemes, binomial and trinomial trees, and lattice-based approaches were initially developed to address option pricing problems where analytical solutions are unavailable [8, 13]. While these methods are effective in low-dimensional settings, their computational complexity increases exponentially with the dimensionality of the underlying state space, limiting their applicability in realistic multi-factor financial environments [9].

Monte Carlo simulation offers a flexible alternative due to its ability to handle high-dimensional stochastic processes and complex payoff structures [6, 14]. However, its direct application to American option pricing is non-trivial because optimal exercise decisions must be determined at multiple exercise times. This leads to the central challenge of continuation-value approximation, which represents the conditional expected payoff of continuation versus immediate exercise. To address this challenge, the Least Squares Monte Carlo (LSMC) method proposed by Longstaff and Schwartz [7] provides a practical framework that combines simulation with regression-based approximation of continuation values. Despite its efficiency and scalability, the accuracy of LSMC is highly sensitive to the choice of regression basis functions, which may limit its ability to capture nonlinear continuation structures in complex market environments.

More recently, machine-learning-based approaches have been explored as alternatives to regression-based methods, including neural-network and deep-learning architectures [22, 26, 30]. These methods demonstrate strong approximation capabilities, particularly in high-dimensional settings. However, they often suffer from reduced interpretability, high computational cost, and sensitivity to hyper parameter tuning, which may limit their practical applicability in financial decision-making contexts. Genetic Programming (GP) offers an alternative data-driven framework that constructs symbolic expressions without predefined functional forms. Its evolutionary structure allows flexible exploration of nonlinear functional relationships, making it particularly suitable for continuation-value approximation problems. Moreover, GP provides explicit symbolic representations, which can enhance interpretability compared to purely black-box machine-learning models [2, 16, 18]. Symbolic-regression studies in financial modelling have highlighted the potential of explicit evolved expressions for discovering nonlinear relationships while maintaining model transparency [18, 28].

Recent machine-learning approaches to American option valuation have substantially expanded the range of function approximators available for continuation-value estimation. Neural-network methods provide flexible nonlinear continuation approximations, while deep optimal-stopping approaches can learn stopping decisions directly from simulated trajectories [22, 26, 30]. More recently, Gaussian-process formulations have also been developed for American option valuation [27]. These developments make it important to distinguish methodological novelty from the general idea of replacing polynomial regression with a more flexible learning mechanism.

The contribution of the present study lies in the specific integration of evolutionary symbolic learning with the recursive Monte Carlo optimal-stopping structure. In the proposed MCGP framework, Genetic Programming is not used merely as a stand-alone regression device. At each exercise date during backward induction, discounted future payoffs generated from risk-neutral Monte Carlo paths form

the continuation-value targets, and an evolutionary symbolic search constructs an explicit approximation of the conditional continuation function. The resulting symbolic function is then embedded directly in the exercise-versus-continuation decision, and this procedure is repeated recursively across exercise dates.

This architecture distinguishes MCGP from three major classes of existing approaches. First, unlike conventional LSMC, which estimates coefficients within a prespecified finite-dimensional basis-function space, MCGP searches over the functional structure of the continuation-value approximation through symbolic expression evolution [7, 10, 11]. Second, unlike neural-network and deep-learning approaches, which generally represent continuation values or stopping rules through parameterized network mappings, MCGP returns explicit symbolic expressions that can be inspected and interpreted [22, 26, 30]. Third, relative to stand-alone symbolic-regression applications in financial modelling, the proposed method embeds symbolic evolution inside the backward-induction mechanism of an American-option optimal-stopping problem, so that symbolic continuation functions are sequentially learned and used to determine exercise decisions [18, 28].

Accordingly, the methodological contribution of MCGP is not simply the replacement of least-squares regression by another nonlinear regression model. Rather, it consists of coupling Monte Carlo path simulation, recursive optimal stopping, and adaptive symbolic function discovery within a unified pricing procedure. This combination yields a continuation-value approximation mechanism that is data-adaptive, unconstrained by predefined polynomial bases, and explicitly interpretable.

Table 1. Comparative Positioning of American Option Pricing and Learning Approaches

Method	Approximation / decision mechanism	Functional structure	Interpretability	Main limitation
Binomial tree	Recursive lattice valuation	Predefined discrete lattice	High	Poor scalability with dimensionality
Conventional LSMC	Least-squares continuation regression	Predefined basis functions	High to moderate	Basis-function dependence
Neural-network methods	Nonlinear learned approximation	Parameterized neural architecture	Low	Hyperparameter and architecture dependence
Deep optimal stopping	Learned stopping strategy	Deep neural architecture	Low	Training and architectural complexity
Gaussian-process approaches	Nonparametric / constrained regression	Kernel-dependent	Moderate	Kernel specification and computational scaling
Symbolic regression	Data-driven symbolic function discovery	Evolved symbolic expressions	High	Complexity and overfitting control
Proposed MCGP	**Recursive symbolic continuation learning within Monte Carlo backward induction**	**Evolved symbolic expressions at exercise dates**	**High**	**Sensitivity to GP hyperparameters and symbolic complexity**

The following section presents the theoretical foundations and computational structure of the proposed MCGP framework in detail.

3 Methodology

Pricing American-style derivatives is formulated as a stochastic optimal stopping problem in continuous time [5, 14, 25]. The presence of early-exercise rights transforms the valuation into a high-dimensional dynamic programming problem for which closed-form solutions are generally unavailable [5, 7, 14, 25]. Consequently, simulation-based numerical methods are required for practical pricing [6, 7, 25].

3.1 Monte Carlo Simulation Framework

Monte Carlo simulation is widely used in computational finance for derivative pricing due to its flexibility in handling path-dependent and high-dimensional stochastic systems [6,8,25,29].Under the risk-neutral measure, it provides a numerical framework for approximating derivative values through repeated stochastic sampling [6, 12, 29].

3.1.1 Geometric Brownian Motion under the Risk-Neutral Measure

The dynamics of the underlying asset price $S(t)$are modeled using Geometric Brownian Motion (GBM), which represents one of the standard stochastic processes employed in option-pricing theory [4, 5, 12, 13]. Under the risk-neutral measure, the asset-price dynamics are governed by:

$$dS(t) = rS(t)dt + \sigma S(t)dW_t^Q, \tag{1}$$

where:

- $r$denotes the risk-free interest rate,
- σrepresents asset-price volatility,
- W_t^Q is a standard Brownian motion under the risk-neutral probability measure.

The analytical solution of the stochastic differential equation is given by:

$$S(t) = S(0)\exp\left[\left(r - \frac{1}{2}\sigma^2\right)t + \sigma W_t^Q\right], \tag{2}$$

This formulation satisfies the no-arbitrage condition and provides the stochastic foundation for Monte Carlo path generation in derivative valuation [4, 5, 12].

3.1.2 Monte Carlo Path Generation

To generate stochastic asset-price trajectories, the continuous-time GBM process is discretized over finite time intervals of length Δt [6,8,12,29]. The discrete-time representation employed in this study is expressed as:

$$S_{t+\Delta t} = S_t \exp\left((r - \frac{1}{2}\sigma^2)\Delta t + \sigma\sqrt{\Delta t}Z_t\right),$$

where $Z_t \sim N(0,1)$ denotes independent standard normal random variables [6, 12].

Using this formulation, a large ensemble of stochastic asset-price trajectories is generated under the risk-neutral measure [6, 8, 29]. These simulated paths constitute the computational basis for continuation-value estimation and option valuation within the proposed MCGP framework [7, 10, 25]. Under the risk-neutral valuation principle, the simulated trajectories provide an empirical approximation of the probability space over which expected discounted option payoffs are evaluated [5, 6, 12].

Table 2. Model Parameters and Simulation Settings

Parameter	Description	Baseline / experimental specification
S_0	Initial asset price	40
K	Strike price	40
r	Risk-free interest rate	6% baseline; 2% and 8% in the interest-rate experiment
σ	Volatility	20% baseline; 10–50% in the volatility experiment
T	Time to maturity	1 year
N	Number of simulated paths	10,000 per run
Δt	Time step	$1/50$ year
Z_t	Standard normal random variable	$\mathcal{N}(0,1)$
W_t^Q	Standard Brownian motion under risk-neutral measure	—

The baseline contract specification $(S_0, K, r, \sigma, T) = (40, 40, 0.06, 0.20, 1)$ follows the canonical single-factor American-put experiment of Longstaff and Schwartz [7], in which the option is exercisable 50 times per year. In the present experiments, one market parameter is varied at a time: the volatility analysis varies σ while holding $r = 6\%$ fixed, whereas the interest-rate analysis varies r while holding $\sigma = 20\%$ fixed. The strike, initial asset price, maturity, and exercise-time discretization remain unchanged. This explicit specification removes ambiguity about the fixed parameters used across the numerical sections.

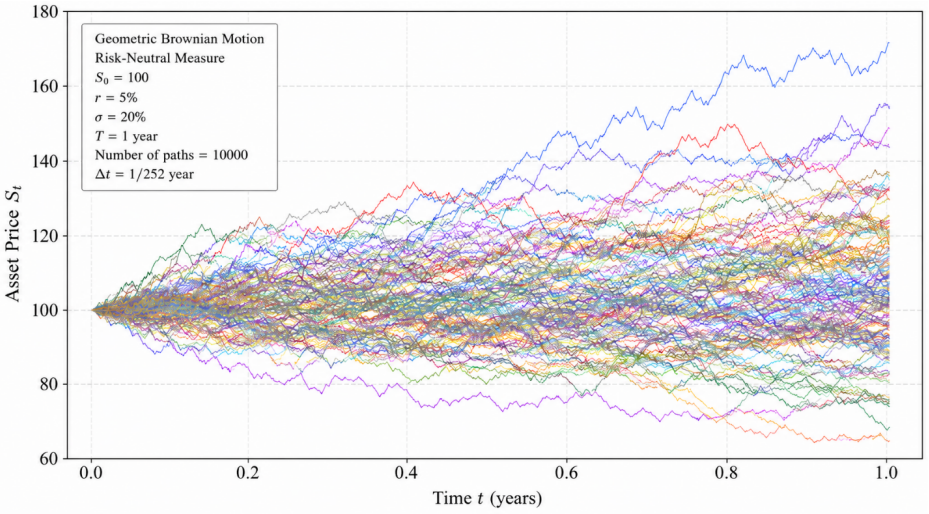


Figure 1. Simulated Geometric Brownian motion Asset Paths under Risk-Neutral Measure

Figure 1 is a generic illustration of risk-neutral GBM path dispersion using the parameters printed inside the panel ($S_0 = 100$, $r = 5\%$, $\sigma = 20\%$, $T = 1$, and $\Delta t = 1/252$). It is included only to visualize Monte Carlo path generation and is not one of the numerical experiments reported in Sections 4–5. The pricing experiments use the baseline and scenario-specific settings in Table 2.

3.1.3 Option Payoff Estimation

For a European call option, the terminal payoff corresponding to the i-th simulated trajectory is defined as [4, 5, 13]:

$$C_i(T) = \max(S_i(T) - K, 0). \tag{3}$$

Similarly, the payoff of a European put option is given by:

$$P_i(T) = \max(K - S_i(T), 0). \tag{4}$$

The corresponding Monte Carlo estimators for option valuation are obtained by averaging discounted payoffs across all simulated paths [6, 8, 29]:

$$C_0 = e^{-rT} \frac{1}{N} \sum_{i=1}^{N} C_i(T), \tag{5}$$

$$P_0 = e^{-rT} \frac{1}{N} \sum_{i=1}^{N} P_i(T), \tag{6}$$

where Ndenotes the total number of simulated trajectories.

As the number of simulated paths increases, the Monte Carlo estimator converges toward the theoretical option value according to the law of large numbers [6, 8, 29]. The standard error associated with the estimator is proportional to:

$$\text{Standard Error} = \frac{\sigma_{\text{Payoff}}}{\sqrt{N}}. \tag{7}$$

Indicating that estimation accuracy improves as the simulation sample size increases [6, 29].

3.1.4 Role of Monte Carlo Simulation Within the Proposed MCGP Framework

Monte Carlo simulation generates stochastic asset-price trajectories and provides the dataset for continuation-value learning [6, 8, 29]. These simulated paths are integrated with Genetic Programming to construct adaptive symbolic approximations of continuation values for American option pricing.

3.2 Genetic Programming Framework

Genetic Programming (GP) is an evolutionary symbolic regression method that evolves mathematical expressions using principles of natural selection [15, 16, 17]. Unlike Genetic Algorithms, which optimize parameter vectors, GP evolves hierarchical expression trees capable of representing nonlinear functional relationships [2, 15, 16, 19].

3.2.1 Symbolic Representation of Candidate Solutions

In GP, each candidate solution is represented as a symbolic expression tree composed of function nodes and terminal nodes [15, 16, 17]. The general framework permits mathematical operators, nonlinear transformations, state variables, and numerical constants. For the one-dimensional GBM experiments reported in this study, the implemented function set is $\{+,-,\times,\div,\log,\exp,\sin,\cos\}$, while the terminal set is the normalized current asset state $x = S_t/K$ together with ephemeral numerical constants sampled uniformly from $[-2,2]$. Thus, volatility, interest rate, and time enter through the simulated path distribution and backward-induction stage rather than as separate GP terminals in the reported implementation. Protected numerical definitions are used for division, logarithm, and exponential evaluation, as specified in the reproducibility subsection below.

This hierarchical symbolic representation enables GP to generate highly flexible nonlinear functions for continuation-value approximation [2, 15, 16, 18, 19].

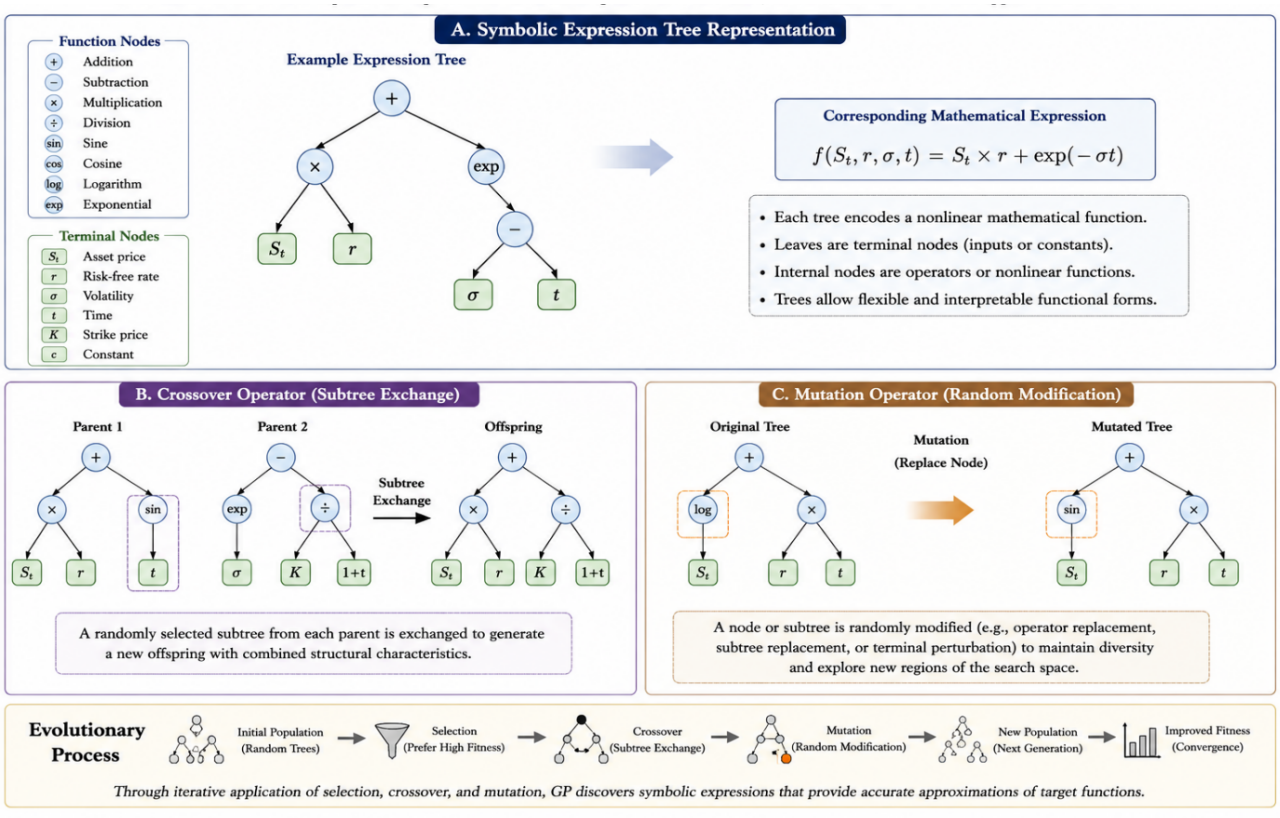


Figure 2. Symbolic Representation and Evolutionary Optimization Process in Genetic Programming

Figure 2 is a generic conceptual illustration of GP tree representation and evolutionary operators; the example terminals shown in the diagram are illustrative rather than the active inputs of the reported implementation. Candidate solutions are represented as hierarchical expression trees composed of function nodes and terminal nodes. Through evolutionary operators including selection, crossover, and mutation, symbolic structures are iteratively modified and refined over successive generations. The evolutionary process progressively refines candidate symbolic expressions through exploration and selection, promoting improved fitness and more effective functional approximations over successive generations. This adaptive symbolic-learning mechanism enables GP to construct flexible nonlinear functions without requiring predefined model structures or basis functions. In the experiments reported here, the active terminal set is restricted to $x = S_t/K$ and ephemeral constants, as specified above and in Table 3A.

3.2.2 Population Initialization and Evolutionary Operators

The GP process begins by generating an initial population of symbolic candidate solutions designed to provide structural diversity across the search space [15]. Each symbolic expression evolves iteratively through evolutionary operators inspired by biological evolution. [15, 17]. The optimization process is guided by three principal operators:

- **Selection:** High-fitness symbolic individuals are selected probabilistically according to their approximation performance and predictive accuracy [19].
- **Crossover:** Subtrees from two parent symbolic expressions are exchanged to generate new offspring and improve structural exploration of the solution space [15].
- **Mutation:** Random structural modifications are introduced into symbolic trees to preserve diversity and prevent premature convergence during optimization [15]. For a population containing Nsymbolic individuals, the expected number of crossover and mutation operations is defined as:

$$N_c = P_c \times N, N_m = P_m \times N, \tag{8}$$

  where:

  - $P_c$denotes the crossover probability,
  - $P_m$represents the mutation probability,
  - $N_c$and $N_m$correspond to the number of crossover and mutation operations, respectively.

The evolutionary optimization procedure continues iteratively across multiple generations until convergence conditions are satisfied [15, 19].

Table 3. Main Components of the Genetic Programming Framework

Component	Description
Population	Set of symbolic candidate solutions
Selection	Chooses high-fitness symbolic expressions
Crossover	Exchanges symbolic subtrees between parent structures
Mutation	Introduces random structural modifications
Fitness Function	Measures continuation-value approximation accuracy
Terminal Nodes	Input variables and numerical constants
Function Nodes	Mathematical operators and nonlinear functions
Expression Tree	Symbolic representation of continuation-value functions

3.2.3 Fitness Evaluation and Symbolic Regression

The quality of each symbolic candidate solution is evaluated through a fitness function that measures continuation-value approximation accuracy. Within the proposed MCGP framework, the target variable is defined as the discounted future option value obtained during the

backward-induction procedure:

$$Y_i = e^{-r\Delta t} V_{t+\Delta t}^{(i)},$$

where:

- Yi denotes the continuation-value target associated with the $i-th$ simulated path,
- r is the risk-free interest rate,
- Δt represents the time step,
- $Vt^{+}\Delta t^{i}$ is the option value at the subsequent exercise date along path i.

For each symbolic candidate function $P(St)$, Genetic Programming approximates the continuation value using the current asset price St as the explanatory variable. The fitness function is evaluated using the Mean Squared Error (MSE):

$$MSE = \frac{1}{n_t} \sum_{i \in \mathscr{I}_t} \left[Y_i - \widehat{C}_t(S_t^{(i)}) \right]^2,$$

where:

- $\widehat{C}_t(S_t^{(i)})$ denotes the continuation value predicted by the calibrated GP symbolic expression,
- $\mathscr{I}_t$ is the set of in-the-money training paths at exercise date t, and $n_t = |\mathscr{I}_t|$.

The objective of the evolutionary optimization process is to identify a symbolic function that provides an accurate continuation-value approximation while avoiding unnecessary structural growth. The primary pricing experiments retain the MSE objective in (11). For the dedicated complexity-control diagnostic reported in Section 4.6, we additionally evaluate a parsimony-regularized objective

$$\mathscr{L}_\lambda(P) = MSE(P) + \lambda \frac{|P|}{N_{\max}},$$

where $|P|$ is the number of nodes in the symbolic tree, $N_{\max}$ is the imposed node cap, and $\lambda \geq 0$ controls the strength of the complexity penalty. Thus $\lambda = 0$ reproduces the unregularized MSE criterion used in the main pricing comparison. Structural bloat is further limited by a maximum tree depth and a maximum node count. These safeguards are treated as explicit complexity controls rather than as evidence that overfitting is eliminated.

To assess generalization independently of in-sample fitness, the complexity diagnostic estimates the symbolic stopping policy on a training set of Monte Carlo paths and applies the learned expressions, without retraining, to an independently simulated validation set. This separation follows the same general diagnostic principle used in simulation-based American-option valuation: a stopping rule estimated from one set of paths should retain its pricing performance when applied to new paths. The resulting out-of-sample pricing error is therefore reported together with tree size and training error.

3.2.4 Mathematical Structure of Symbolic Approximation

The objective of GP is to identify an optimal nonlinear symbolic function mapping stochastic state variables to continuation values or option prices. A generic symbolic approximation can be expressed as [15, 16, 19]:

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{m} c_i g_i(x_1, x_2, \ldots, x_n), \tag{9}$$

where:

- $x_1, x_2, \ldots, x_n$denote stochastic input variables,
- $g_i(\cdot)$represent nonlinear symbolic basis components,
- c_i are adaptive coefficients obtained through evolutionary optimization.

Unlike conventional regression approaches, GP does not require predefined basis functions. Instead, symbolic functional structures evolve adaptively according to stochastic market dynamics observed along simulated trajectories. An example symbolic continuation-value representation is:

$$C(S_t) = a_1 S_t + a_2 log(S_t) + a_3 S_t^{\frac{1}{2}}.$$

This adaptive symbolic structure expands the functional hypothesis space beyond fixed polynomial regressions, allowing the model to reduce approximation bias and better represent nonlinear continuation-value dynamics under complex stochastic market conditions [15].

3.2.5 Computational Complexity and Convergence

The computational burden of MCGP should be distinguished from that of conventional LSMC because the two methods perform different continuation-value searches. Let J denote the number of exercise dates, N_{ITM} the number of in-the-money observations at a representative date, P the GP population size, G the number of generations, and C_{tree} the average cost of evaluating one symbolic tree for one observation. Ignoring the common Monte Carlo path-generation cost, the dominant evolutionary work of MCGP is approximately

$$O(JGPN_{\mathrm{ITM}}C_{\mathrm{tree}}).$$

The affine calibration of each candidate adds a linear pass through the in-the-money observations and is absorbed into the evaluation term above. Tree depth, node count, early stopping, and the number of in-the-money paths therefore affect computational cost directly in addition to their statistical role.

For comparison, if conventional LSMC uses B polynomial basis functions, its least-squares regressions require approximately

$$O\left(J\left(N_{\mathrm{ITM}}B^2 + B^3\right)\right),$$

under a standard dense least-squares implementation. In the present benchmark B is small, whereas MCGP repeatedly evaluates a population of nonlinear symbolic candidates. Consequently, there is no general complexity argument implying that MCGP should be faster than LSMC. Its computational cost is the price of searching over functional structure rather than fitting coefficients in a fixed low-dimensional basis. Appropriate adjustment of population size, stopping rules, tree restrictions, crossover, and mutation is therefore important for balancing approximation flexibility against runtime [15, 17]. Section 5.2 provides a controlled reproducible timing comparison.

3.2.6 Implementation Details and Reproducibility

To make the revised MCGP experiments reproducible, the implementation used for the new hyperparameter-sensitivity and complexity-validation diagnostics is specified explicitly in Table 3A. The reported implementation is a one-dimensional continuation-value model in which GP receives the normalized state variable $x = S_t/K$ on in-the-money paths. Each evolved tree produces a nonlinear scalar feature $z = P(x)$, and an affine least-squares calibration $a + bz$ is fitted to the discounted continuation targets at each exercise date. This affine calibration is part of the implemented symbolic-regression procedure and is applied consistently to every candidate tree.

The function set is $\{+, -, \times, \div, \log, \exp, \sin, \cos\}$. Division is protected by returning zero where the denominator magnitude is at most 10^{-6}; logarithm is evaluated as $\log(|u| + 10^{-6})$; and exponential arguments are clipped to $[-8, 8]$. Final intermediate expression values are clipped to $[-10^4, 10^4]$ and non-finite values are replaced numerically. Constants are drawn uniformly from $[-2, 2]$. Candidate trees are initialized recursively, with terminal selection forced at the maximum depth and allowed probabilistically at intermediate levels. Tournament selection of size three, one-individual elitism, subtree crossover, and subtree mutation are used. Offspring violating the maximum depth or 63-node cap are rejected in favour of the unmodified parent.

For the revision diagnostics, the reference configuration uses a population of 40 individuals, a maximum of 10 generations, crossover probability $P_c = 0.8$, mutation probability $P_m = 0.1$, maximum tree depth 4, and early-stopping patience of four generations. The best fitness must improve by more than 10^{-7} to reset the patience counter. Genetic Programming is performed only when at least 30 in-the-money paths are available at an exercise date. The Monte Carlo state process is sampled with NumPy's default pseudo-random generator. The hyperparameter-sensitivity experiment uses Monte Carlo seed 101; GP seeds at exercise date t are deterministically derived as $1000 \times 101 + t$. The independent complexity-validation experiment uses training-path seed 24680 and validation-path seed 97531; its exercise-date GP seeds are derived analogously from the training seed. These seeds apply to the newly added revision experiments only. The archived output tables from the original numerical study did not retain their historical random seeds, and no retrospective seed is assigned to those legacy results.

Table 4. Reproducibility Specification for the Revised MCGP Implementation

Implementation item	Specification
State variable supplied to GP	$x = S_t/K$ on in-the-money simulated paths
Terminal set	$\{x, c\}$, with $c \sim U[-2, 2]$
Function set	$\{+, -, \times, \div, \log, \exp, \sin, \cos\}$ with protected numerical evaluation
Candidate output	$a + bP(x)$, with a, b fitted by least squares for each candidate tree
Population size	40 in the reference revision configuration
Maximum generations	10
Selection	Tournament selection, tournament size 3
Elitism	Best individual preserved each generation
Crossover probability	$P_c = 0.8$
Mutation probability	$P_m = 0.1$
Maximum tree depth	4 in the reference configuration
Maximum tree size	63 nodes
Stopping rule	Stop at 10 generations or after 4 consecutive generations without fitness improvement greater than 10^{-7}
Minimum ITM sample	GP regression skipped when fewer than 30 in-the-money paths are available
Fitness	MSE for the primary/reference search; $MSE + \lambda\|P\|/63$ in the dedicated complexity diagnostic
Exercise grid	50 exercise opportunities per year for the baseline one-year contract
Revision-diagnostic path count	1,000 paths per diagnostic run; independent 1,000-path validation set in Section 4.6
Sensitivity seed protocol	Monte Carlo seed 101; exercise-date GP seed $101000 + t$
Validation seed protocol	Training paths 24680; validation paths 97531; exercise-date GP seed $24680000 + t$
Software environment	Python 3.13.5; NumPy 2.3.5; 64-bit Linux execution environment

The complete source used for the revised sensitivity and validation diagnostics is retained with the LaTeX submission package in the `code/` directory, together with the corresponding CSV outputs and a minimal environment specification. Because the original study's archived primary-result scripts and random seeds are not available in the revision package, Table 3A should be interpreted as the fully specified reproducible configuration for the newly added revision experiments, not as a retrospective reconstruction of undocumented historical settings for Tables 5–7.

3.2.7 Advantages of GP in American Option Pricing

The integration of Genetic Programming within the proposed MCGP framework provides several potential advantages for continuation-value estimation and American option pricing [15, 16, 19]:

- **Symbolic Interpretability:** The resulting symbolic expressions remain interpretable and provide insight into continuation dynamics and optimal exercise behaviour. Unlike many black-box machine-learning approaches, Genetic Programming generates explicit symbolic expressions that can be directly interpreted and analysed, providing greater transparency in continuation-value modelling and supporting direct inspection of the fitted continuation expressions [15, 16, 19].
- **Structural Flexibility:** GP dynamically evolves continuation-value representations according to stochastic market conditions.
- **Robustness under Volatile Markets:** The symbolic search provides additional functional flexibility that may be useful when continuation-value relationships are nonlinear or difficult to represent with a small fixed polynomial basis.
- **Elimination of Basis-Function Dependency:** Unlike conventional LSMC regression frameworks, GP removes the need for manually specified basis functions.

3.3 Least Squares Monte Carlo (LSMC)

The Least Squares Monte Carlo (LSMC) method proposed by Francis A. Longstaff and Eduardo S. Schwartz has become one of the most influential simulation-based methodologies for pricing American-style derivatives [7]. The method combines Monte Carlo simulation with least-squares regression to approximate continuation values during backward induction and determine optimal exercise policies numerically. The principal strength of LSMC lies in its ability to handle high-dimensional stochastic systems where conventional lattice and finite-difference methods become computationally inefficient. Consequently, LSMC has been widely adopted in computational finance for pricing American options, real options, energy derivatives, and other path-dependent financial contracts [6, 7].

3.3.1 Monte Carlo Path Simulation

Within the LSMC framework, stochastic trajectories of the underlying asset price are first generated under the risk-neutral probability measure using Monte Carlo simulation. The asset-price dynamics follow Geometric Brownian Motion:

$$S_{t+\Delta t} = S_t \exp\left[\left(r - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t} Z_t\right], \tag{10}$$

where:

- $r$denotes the risk-free interest rate,
- σrepresents asset-price volatility,
- $Z_t \sim N(0,1)$ Corresponds to independent standard normal random variables.

A large ensemble of simulated trajectories is generated to estimate future continuation payoffs and optimal exercise decisions [7, 19].

3.3.2 Backward Induction and Continuation-Value Estimation

The LSMC methodology proceeds backward in time from maturity to the initial valuation date. At maturity T, the option value equals its intrinsic payoff ([7]):

$$V_T = h(S_T). \tag{11}$$

For an American put option, the immediate exercise payoff is:

$$h(S_t) = \max(K - S_t, 0). \tag{12}$$

At each intermediate exercise date, the continuation value is estimated through least-squares regression using discounted future payoffs observed along simulated in-the-money paths [7]:

$$C_t(S_t) = E^{Q}\left[e^{-r\Delta t} V_{t+\Delta t} \mid S_t\right]. \tag{13}$$

Because the conditional expectation is not directly observable, LSMC approximates the continuation value using regression on predefined basis functions [7, 6]:

$$\widehat{C}_t(S_t) = \sum_{j=1}^{m} \beta_j \phi_j(S_t), \tag{14}$$

where:

- $\phi_j(S_t)$denote predefined regression basis functions,
- β_j are regression coefficients estimated through least squares.

Polynomial basis functions are commonly employed in practical implementations. However, the approximation accuracy of the LSMC framework strongly depends on the selection of basic functions, which may reduce robustness under highly nonlinear market conditions [7, 19].

3.3.3 Optimal Exercise Rule

After estimating the continuation value, the optimal exercise strategy is determined by comparing immediate exercise payoffs with continuation estimates. The option is exercised whenever [7]:

$$h(S_t) > \widehat{C}_t(S_t). \tag{15}$$

Otherwise, the option is retained for future exercise opportunities. The backward-induction procedure continues recursively until the initial valuation date is reached, producing the estimated American option price.

Table 5. Main Computational Components of the LSMC Framework

Component	Description
Monte Carlo Simulation	Generates stochastic asset-price trajectories
Backward Induction	Recursive valuation from maturity to present
Regression Approximation	Estimates continuation values
Basis Functions	Predefined polynomial approximation terms
Exercise Decision	Compares intrinsic payoff and continuation value
Discounting	Computes present value of future payoffs

3.3.4 Advantages of LSMC

The LSMC methodology provides several important advantages in computational finance:

- **Scalability:** Efficiently handles high-dimensional stochastic pricing problems.
- **Flexibility:** Applicable to American-style and path-dependent derivatives.
- **Computational Efficiency:** Compatible with parallel Monte Carlo implementations.
- **Numerical Stability:** Provides relatively stable valuation performance under various stochastic market conditions [7, 10, 11].

3.3.5 Limitations of Conventional LSMC Regression

Despite its widespread success, the performance of LSMC remains strongly dependent on the specification of regression basis functions. In conventional implementations, continuation values are approximated using predefined polynomial structures selected heuristically [7, 19]. Under highly nonlinear market conditions or volatile stochastic environments, fixed polynomial approximations may introduce substantial structural bias and numerical instability. In addition, the approximation accuracy of the continuation-value function may deteriorate significantly as the dimensionality and complexity of the pricing problem increase [19]. From a conceptual perspective, the overall continuation-value estimation error may be viewed as arising from sampling, approximation, and optimization components:

$$\text{Total Error} = E_{\text{sampling}} + E_{\text{approximation}} + E_{\text{optimization}}, \tag{16}$$

where:

- E_{sampling} denotes Monte Carlo sampling error,
- $E_{\text{approximation}}$corresponds to regression approximation bias,
- $E_{\text{optimization}}$ reflects optimization and numerical estimation errors [6,7].

The accuracy of the LSMC method critically depends on the choice of basis functions used in regression. Inadequate specification may introduce approximation bias, particularly under nonlinear or high-volatility market dynamics. From a function-approximation perspective, this limitation can be interpreted as a projection bias induced by restricting continuation-value estimation to a finite-dimensional basis-function space [10, 11]. This limitation motivates the use of more flexible functional approximations such as symbolic regression.

3.3.6 Motivation for the Proposed MCGP Framework

The limitations associated with predefined regression structures motivate the development of the proposed Monte Carlo–Genetic Programming (MCGP) framework. Unlike conventional LSMC implementations, the proposed methodology replaces fixed polynomial regression with adaptive symbolic regression based on Genetic Programming [7, 15, 16, 19]. Instead of specifying continuation-value basis functions exogenously, the proposed MCGP framework evolves symbolic nonlinear continuation functions dynamically from simulated stochastic trajectories. This adaptive learning mechanism increases structural flexibility and may reduce projection bias when a small fixed polynomial basis is too restrictive.

3.3.7 Evolutionary Optimization Framework of MCGP

The proposed Monte Carlo–Genetic Programming (MCGP) framework employs an iterative evolutionary optimization procedure in which symbolic representations of the continuation value are progressively improved across successive generations. At generation t, the population of candidate programs is defined as:

$$P_t = \{P_{t,1}, P_{t,2}, \ldots, P_{t,M}\},$$

where Mdenotes the population size and $P_{(t,j)}$represents the j-th symbolic expression in generation t. Each candidate program serves as an approximation of the continuation value function.

For each candidate program, performance is evaluated using Monte Carlo simulation. Let $z_1, z_2, \ldots, z_N$ be Nindependent realizations of the underlying stochastic process. The fitness function is defined as the average approximation error over all simulated paths:

$$F(P) = \frac{1}{N}\sum_{i=1}^{N} L(C(z_i), P(z_i)),$$

where $C(z_i)$denotes the target continuation value and $P(z_i)$is the value predicted by the symbolic program. $L(\cdot)$represents the loss function.

In this study, the mean squared error (MSE) is used as the loss function:

$$L(C(z_i), P(z_i)) = (C(z_i) - P(z_i))^2.$$

Therefore, the fitness function becomes:

$$F(P) = \frac{1}{N}\sum_{i=1}^{N} (C(z_i) - P(z_i))^2.$$

The evolutionary optimization process is carried out using standard genetic programming operators, including tournament selection, subtree crossover, mutation, and elitist replacement. The objective is to identify the symbolic expression that minimizes the expected continuation-value approximation error under the risk-neutral probability measure. From a function-approximation perspective, the evolutionary search process explores a flexible symbolic hypothesis space, allowing continuation-value representations to emerge directly from simulated observations rather than being restricted to predefined regression structures [15, 16]. The optimization problem can be formulated as:

$$P^* = \arg\min_{P \in H} E\left[(C(Z) - P(Z))^2\right],$$

where:

- Hdenotes the hypothesis space of symbolic expressions generated by Genetic Programming
- Zrepresents the stochastic state variable
- $C(Z)$is the true continuation value
- $P(Z)$is the symbolic approximation produced by the GP model

Since the expectation cannot be computed analytically, it is approximated using Monte Carlo sampling:

$$P^* \approx \arg\min_{P \in H} \frac{1}{N}\sum_{i=1}^{N} (C(z_i) - P(z_i))^2.$$

The iterative evolutionary process continues until a stopping criterion is satisfied, such as reaching the maximum number of generations or convergence of the best fitness value. The final output of the algorithm is a stable symbolic approximation of the continuation value within the proposed MCGP framework.

3.4 Proposed Monte Carlo-Genetic Programming (MCGP) Framework

The proposed Monte Carlo–Genetic Programming (MCGP) framework integrates Monte Carlo simulation with Genetic Programming-based symbolic regression to approximate continuation values in American option pricing. In contrast to the Least Squares Monte Carlo (LSMC) method, which relies on predefined polynomial basis functions, the proposed framework constructs continuation-value functions through adaptive symbolic evolution driven by simulated stochastic data [7, 19]. The American option valuation problem is formulated as an optimal stopping problem under the risk-neutral measure:

$$V_t = \max\left(h(S_t), C_t(S_t)\right),$$

where h(St) denotes the immediate exercise payoff. For an American put option, the payoff is defined as:

$$h(S_t) = \max(K - S_t, 0).$$

The continuation value is given by the conditional expectation:

$$C_t(S_t) = E^Q\left[\exp(-r\Delta t)V_{t+\Delta t} \mid S_t\right].$$

The underlying asset follows a Geometric Brownian Motion under the risk-neutral measure:

$$dS_t = rS_t\,dt + \sigma S_t\,dW_t^Q.$$

Its discrete-time approximation used for simulation is:

$$S_{t+\Delta t} = S_t \exp\left[(r - 0.5\sigma^2)\Delta t + \sigma\sqrt{\Delta t}Z_t\right].$$

where $Zt \sim N(0,1)$. Monte Carlo simulation generates N risk-neutral sample paths, which are used for both continuation-value estimation and evaluation of symbolic candidates. The approximation is performed within a symbolic regression framework defined over the hypothesis space:

$$H_{GP} = \{f : \mathbb{R}^n \to \mathbb{R} \mid f \text{ generated by symbolic expression trees}\}.$$

The continuation-value estimation problem is formulated as:

$$C_t^{GP} = \arg\min_{f \in H^{GP}} \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - f(S_t^{(i)})\right)^2.$$

where:

- $CtGP$ denotes the Genetic Programming approximation of the continuation-value function at time t;
- HGP represents the hypothesis space of symbolic expressions generated by Genetic Programming;
- $Y_i = e^{-r\Delta t}V_{t+\Delta t}^{(i)}$ is the discounted future option value associated with the i-th simulated path;
- $St(i)$ denotes the asset price at time t along the $i-th$ Monte Carlo trajectory;
- $f(St(i))$ represents the continuation value predicted by a candidate symbolic expression;
- N is the total number of simulated paths used for training.

The objective of the optimization process is to identify the symbolic continuation-value function that minimizes the mean squared approximation error over all simulated paths.

At each exercise date, the fitted continuation value is used to choose between exercise and continuation, while the pathwise realized value is propagated from the selected action:

$$V_t^{(i)} = \begin{cases} h(S_t^{(i)}), & h(S_t^{(i)}) > C_t^{GP}(S_t^{(i)}), \\ e^{-r\Delta t}V_{t+\Delta t}^{(i)}, & \text{otherwise.} \end{cases}$$

The final option price is obtained by averaging over all simulated paths:

$$\widehat{V}^0 = \frac{1}{N}\sum_{i=1}^{N} V^{0(i)}.$$

The proposed framework replaces fixed regression structures with adaptive symbolic representations, enabling flexible approximation of nonlinear continuation dynamics without requiring a predefined polynomial basis [7, 10, 11]. Both methods use identical Monte Carlo paths and identical simulation settings to ensure fair comparison. The overall implementation procedure of the proposed MCGP framework is summarized in Algorithm 1.

Figure 3 provides a schematic overview of the computational workflow implemented in the proposed MCGP framework.

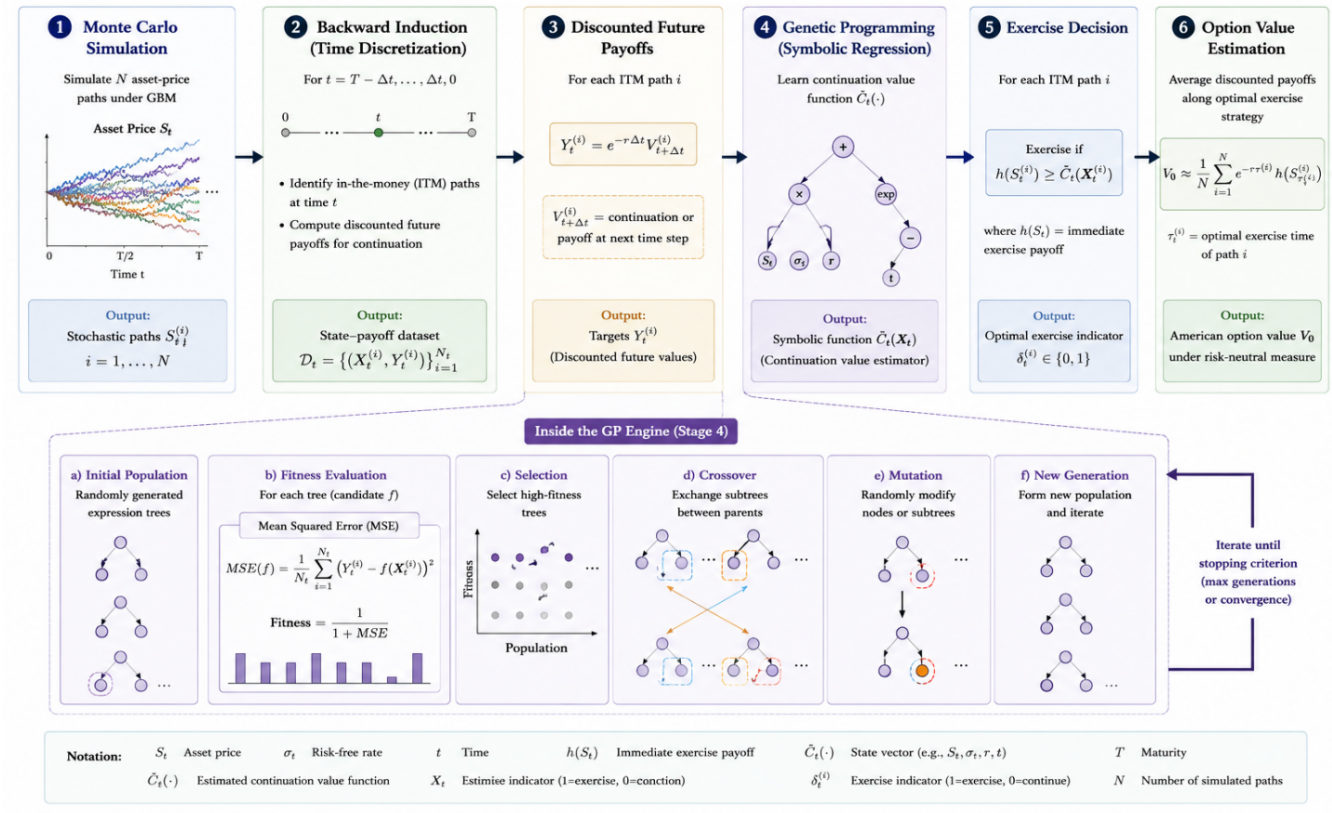


Figure 3. Computational Workflow of the Proposed Monte Carlo–Genetic Programming (MCGP) Framework

The figure presents the computational workflow of the proposed Monte Carlo–Genetic Programming (MCGP) framework. Monte Carlo simulation generates risk-neutral asset-price trajectories, while Genetic Programming performs adaptive symbolic regression to estimate continuation values recursively during backward induction. The resulting continuation-value approximations are used to determine optimal exercise decisions and compute the final American option price.

4 Illustrative Numerical Results

This section presents the empirical evaluation of the proposed Monte Carlo–Genetic Programming (MCGP) framework for pricing American options and compares its performance with the conventional Least Squares Monte Carlo (LSMC) method under different market conditions. The analysis focuses on three principal criteria: pricing accuracy, computational efficiency, and robustness under varying levels of volatility and interest rates. To evaluate the proposed framework, several market scenarios were simulated under a risk-neutral Geometric Brownian Motion setting, including low- and high-volatility and low- and high-interest-rate regimes [5]. For each scenario, American option prices

Algorithm 1: Monte Carlo–Genetic Programming (MCGP) Framework for American Option Pricing

Input: $S_0, K, r, \sigma, T, N, \Delta t$; GP parameters: population size M, maximum generations G, crossover probability P_c, mutation probability P_m, maximum tree depth $D_{\max}$, maximum node count $N_{\max}$, early-stopping patience q; random-seed protocol; number of independent runs R.

Output: Estimated American option price $\widehat{V}_0$.

Step 1: Risk-Neutral Monte Carlo Simulation;

Generate N risk-neutral asset price trajectories under Geometric Brownian Motion:

$$S_{t+\Delta t}^{(i)} = S_t^{(i)} \exp\left[\left(r - \tfrac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t}\, Z_t^{(i)}\right] \tag{37}$$

where $Z_t^{(i)} \sim \mathcal{N}(0,1)$, independent across i, t. Store all simulated paths $\{S_t^{(i)}\}_{i=1}^{N}$.;

Step 2: Terminal Payoff Initialization;

For each path:

$$V_T^{(i)} = h(S_T^{(i)}) = \max(K - S_T^{(i)}, 0) \tag{38}$$

Step 3: Backward Induction;

for $t = T - \Delta t, T - 2\Delta t, \ldots, \Delta t$ **do**

Step 4: Construction of Continuation Targets;

For each path i:

$$Y_i = e^{-r\Delta t} V_{t+\Delta t}^{(i)} \tag{39}$$

where $V_{t+\Delta t}^{(i)}$ is obtained from the backward induction step.;

Step 5: Genetic Programming Fitness Evaluation;

For each candidate symbolic tree $P_j(\cdot)$ on the in-the-money set $\mathcal{I}_t$, evaluate $x_i = S_t^{(i)}/K$, fit $a_j + b_j P_j(x_i)$ by least squares, and compute:

$$F(P_j) = \frac{1}{n_t} \sum_{i \in \mathcal{I}_t} \left[Y_i - \left(a_j + b_j P_j(x_i)\right)\right]^2 \tag{40}$$

Step 6: Evolutionary Optimization;

Apply GP operators: tournament selection, crossover (P_c), mutation (P_m), and elitism. Preserve the best individual by elitism. Reject offspring violating depth or node-count limits. Repeat until G generations or early stopping.;

Step 7: Optimal Symbolic Continuation Function;

$$P^* = \arg\min_{P_j \in P_t} F(P_j) \tag{41}$$

$$C_t^{GP}(S_t) = a^* + b^* P^*(S_t/K) \tag{42}$$

Step 8: Optimal Exercise Decision;

For each path i:

$$V_t^{(i)} = \begin{cases} h(S_t^{(i)}), & h(S_t^{(i)}) > C_t^{GP}(S_t^{(i)}), \\ e^{-r\Delta t} V_{t+\Delta t}^{(i)}, & \text{otherwise.} \end{cases} \tag{43}$$

Step 9: Backward Propagation;

Repeat Steps 4–8 until $t = 0$.;

Step 10: Final Price Estimation;

For a single run:

$$\widehat{V}_0^{(r)} = \frac{1}{N} \sum_{i=1}^{N} V_0^{(i)} \tag{44}$$

For robustness over R independent runs:

$$\widehat{V}_0 = \frac{1}{R} \sum_{r=1}^{R} \widehat{V}_0^{(r)} \tag{45}$$

were computed using both MCGP and LSMC, and evaluated in terms of MAE, RMSE, and computational time. The benchmark option values used for error evaluation were obtained from a high-resolution binomial tree model with 10,000 time steps. Such a fine discretization provides a highly accurate numerical approximation of the corresponding continuous-time American option value and serves as a reliable reference for performance assessment. The underlying asset dynamics were simulated under the risk-neutral process:

$$dS_t = rS_t dt + \sigma S_t dW_t^Q, \tag{17}$$

where $r$denotes the risk-free interest rate, σrepresents volatility, and $W_t$is a standard Brownian motion.

Monte Carlo simulation was used to generate stochastic asset-price trajectories, which served as the basis for continuation-value estimation in both frameworks [6, 7]. In the LSMC method, continuation values were approximated via polynomial least-squares regression, whereas in the proposed MCGP framework, they were estimated using symbolic regression through Genetic Programming [7, 15].

4.1 Expanded Benchmark Framework

The numerical design is anchored to the canonical American-put setting of Longstaff and Schwartz [7]. For the baseline contract $S_0 = K = 40$, $r = 6\%$, $\sigma = 20\%$, and $T = 1$ year, Longstaff and Schwartz report an LSM estimate of 2.313 (standard error 0.009) and an implicit finite-difference value of 2.314 when using 50 exercise opportunities per year. These published values provide an external numerical check on the scale and discretization of the present experimental setting; they are not substituted for the in-study estimates.

To broaden the empirical benchmark beyond polynomial LSMC, two additional continuation-value approximation families were implemented: neural-network regression and Gaussian kernel ridge regression (KRR). Kohler, Krzy{z}ak, and Todorovic [30] motivate neural-network continuation approximation on Monte Carlo paths, while Hu and Zastawniak [31] develop a Gaussian-kernel ridge regression framework for American option valuation. These methods are relevant cross-family comparators because, like MCGP, they modify the continuation-value approximation stage while retaining simulation-based backward induction.

Table 6. Cross-Family Benchmark Positioning for the Numerical Study

Method	Family	Continuation-value mechanism	Role in the revised comparison
LSMC [7]	Polynomial regression	Prespecified cubic basis fitted by least squares	Direct controlled benchmark
Neural-network regression [30]	Neural learning	Learned nonlinear neural approximation	Direct controlled cross-family benchmark
Kernel ridge regression [31]	Kernel learning	Gaussian-RBF regularized regression	Direct controlled cross-family benchmark
High-resolution binomial tree	Lattice method	Backward recursive valuation on a fine tree	Numerical reference used to compute pricing errors
MCGP (proposed)	Evolutionary symbolic regression	Evolved symbolic continuation function	Proposed method

4.1.1 Controlled Cross-Family Numerical Benchmark

A dedicated matched experiment was conducted so that MCGP, cubic-basis LSMC, neural-network regression, and KRR were evaluated on identical Monte Carlo paths under the same American-put contract and exercise grid. The contract uses $S_0 = K = 40$, $r = 6\%$, $T = 1$ year, 50 exercise opportunities, and 1,000 simulated paths per replication. Two volatility regimes were considered: the baseline $\sigma = 20\%$ and a higher-volatility case $\sigma = 40\%$. Ten independent matched replications were performed for each regime using final test seeds 65000–65009. The numerical references are 10,000-step Cox–Ross–Rubinstein American-put values of 2.31955 for $\sigma = 20\%$ and 5.31822 for $\sigma = 40\%$.

All learned continuation models use the normalized state $x = S_t/K$ on in-the-money paths. LSMC uses the cubic basis $\{1, x, x^2, x^3\}$. The neural benchmark uses a single hidden layer with eight tanh units and an L_2 penalty of 0.001. KRR uses a Gaussian RBF kernel with regularization parameter $\alpha = 0.001$ and kernel coefficient $\gamma = 10$. These neural and kernel settings were selected on separate pilot seeds before the final matched test; the final test seeds were not used for configuration selection. MCGP uses the reference configuration in Table 3A: population 40, at most 10 generations, $P_c = 0.8$, $P_m = 0.10$, maximum depth 4, and early-stopping patience 4.

MCGP produced the lowest descriptive MAE and RMSE among the four methods in both volatility regimes. At $\sigma = 20\%$, MCGP achieved MAE 0.06836 and RMSE 0.08746, compared with MAE/RMSE values of 0.07938/0.10550 for LSMC, 0.07399/0.09672 for the neural benchmark, and 0.08571/0.11417 for KRR. At $\sigma = 40\%$, the corresponding MCGP values were 0.14972/0.22022, compared with 0.15797/0.23233 for LSMC, 0.17649/0.24363 for the neural benchmark, and 0.22041/0.28357 for KRR.

The matched comparisons are interpreted cautiously. At 20% volatility, paired t-tests on run-level absolute errors do not establish a significant difference between MCGP and LSMC ($p = 0.357$), the neural benchmark ($p = 0.620$), or KRR ($p = 0.327$). At 40% volatility,

Table 7. Controlled Cross-Family Numerical Benchmark

Volatility	Method	Mean price	MAE	RMSE	95% CI for MAE
20%	MCGP	2.36201	**0.06836**	**0.08746**	[0.02722, 0.10950]
20%	LSMC	2.38128	0.07938	0.10550	[0.02698, 0.13179]
20%	Neural network	2.38235	0.07399	0.09672	[0.02701, 0.12096]
20%	KRR	2.40267	0.08571	0.11417	[0.02885, 0.14258]
40%	MCGP	5.44516	**0.14972**	**0.22022**	[0.02793, 0.27150]
40%	LSMC	5.45897	0.15797	0.23233	[0.02951, 0.28643]
40%	Neural network	5.48439	0.17649	0.24363	[0.04985, 0.30313]
40%	KRR	5.53150	0.22041	0.28357	[0.08587, 0.35494]

Note: Results are based on 10 matched replications with 1,000 paths per replication. Confidence intervals are two-sided 95% t-intervals for run-level absolute error. The experiment is reported as a controlled cross-family benchmark and is not pooled with the 10,000-path historical scenario tables.

the MCGP–LSMC ($p = 0.543$) and MCGP–neural ($p = 0.172$) differences remain non-significant, whereas MCGP has significantly lower absolute error than KRR in this small matched sample ($p = 0.0156$). Thus, the cross-family experiment supports competitive descriptive accuracy and a larger high-volatility advantage relative to the tested kernel configuration, but it does not establish universal statistical superiority over the neural or LSMC alternatives. Raw matched outputs and the pilot tuning summary are retained in the revision source package.

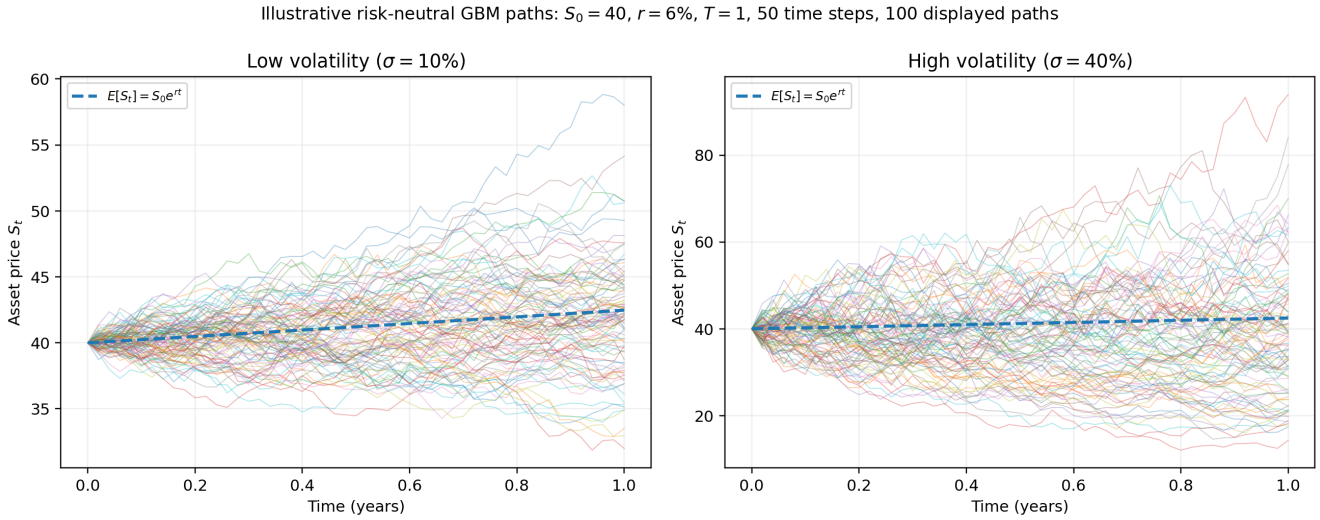


Figure 4. Simulated Geometric Brownian (motion Asset Paths–under Low- and High-Volatility Market Conditions)

The figure presents simulated asset-price trajectories generated under the risk-neutral Geometric Brownian Motion process for two volatility regimes. Panel (a) illustrates low-volatility conditions (σ = 10%), where asset-price paths remain relatively concentrated around the expected trajectory. Panel (b) illustrates high-volatility conditions (σ = 40%), resulting in substantially greater path dispersion and uncertainty. The increased spread of simulated trajectories under high volatility highlights the growing complexity of continuation-value estimation and the challenges faced by regression-based pricing approaches in nonlinear market environments. This behaviour provides a natural setting for evaluating the effectiveness of adaptive continuation-value approximation methods such as the proposed MCGP framework. As volatility increases, the dispersion of simulated trajectories becomes substantially larger, resulting in more complex continuation-value structures. Consequently, accurate continuation-value approximation becomes increasingly important for obtaining reliable American option valuations.

4.2 Pricing Performance under Volatility Scenarios

To evaluate model behaviour under different levels of market uncertainty, American option prices were computed under low-volatility and high-volatility conditions. The comparative results are presented in Table 5. Table 5 reports the representative 10% and 40% scenarios, while Figure 5 displays the archived sensitivity series over the full 10–40% grid.

Table 8. Performance Comparison under Volatility Scenarios

Volatility Scenario	Method	MAE	RMSE	Computation Time (seconds)
Low Volatility (10%)	MCGP	0.012	0.018	110
	LSMC	0.015	0.020	145
High Volatility (40%)	MCGP	0.017	0.022	115
	LSMC	0.022	0.028	160

Note: The computation-time values in Table 5 are retained from the original archived study and should be interpreted only as historical implementation-specific measurements. They are not used to infer an intrinsic computational advantage of MCGP. See Section 5.2 for the controlled reproducible runtime comparison.

The archived scenario results report lower MAE and RMSE point estimates for MCGP under both volatility regimes. Table 5 also contains lower historical computation-time values for MCGP; however, the source code, hardware specification, timing scope, and run-level timing observations associated with these legacy measurements were not retained in the revision archive. These time values are therefore preserved as historical implementation-specific outputs and are not used to infer that MCGP is intrinsically faster than LSMC. Results in the archived comparison were averaged over 50 independent simulation runs and used 10,000 Monte Carlo sample paths per run unless otherwise stated. A controlled reproducible runtime comparison is reported separately in Section 5.2.

The lower descriptive error point estimates reported for MCGP are consistent with the greater functional flexibility of adaptive symbolic representations. From a function-approximation perspective, this provides a plausible explanation for the observed scenario-level differences relative to the fixed finite-dimensional polynomial expansion used in the conventional LSMC benchmark; it should not be interpreted as proof of a universal reduction in approximation bias [10, 11].

4.3 Sensitivity Analysis under Market Volatility

To ensure a data-driven evaluation of model sensitivity with respect to volatility, a structured Monte Carlo experiment is conducted. The analysis is performed over a predefined volatility grid defined as:

$$\sigma \in \{10\%, 15\%, 20\%, 25\%, 30\%, 35\%, 40\%\}.$$

For each volatility level, asset price trajectories are generated using Geometric Brownian Motion under the risk-neutral measure, while all other market parameters remain fixed.

The simulation settings are defined as follows:

- Number of simulated paths: $N = 10{,}000$
- Number of independent Monte Carlo runs: $R = 50$
- Time discretization step: $\Delta t = 1/50$
- Risk-free rate: fixed at 6% across volatility scenarios
- Initial asset price and strike: $S_0 = K = 40$; maturity: $T = 1$ year

For each volatility level, American option prices are computed independently using both the proposed MCGP framework and the LSMC benchmark. The reported RMSE values are obtained by averaging over R independent simulation runs to ensure statistical stability. The archived RMSE profiles in Figure 5 show lower MCGP point estimates than LSMC across the displayed volatility grid. The gap is larger at the upper end of the displayed grid, a descriptive pattern that is consistent with greater approximation flexibility but is not by itself a

formal significance result. Figure 5 summarizes the sensitivity of pricing accuracy to increasing volatility levels and illustrates the relative robustness of the proposed MCGP framework compared with the conventional LSMC approach.

Volatility Scenario	Method	MAE	RMSE	Computation Time (seconds)
Low Volatility (10%)	MCGP	0.012	0.018	110
	LSMC	0.015	0.020	145
High Volatility (40%)	MCGP	0.017	0.022	115
	LSMC	0.022	0.028	160

Results are averaged over 50 independent simulation runs.

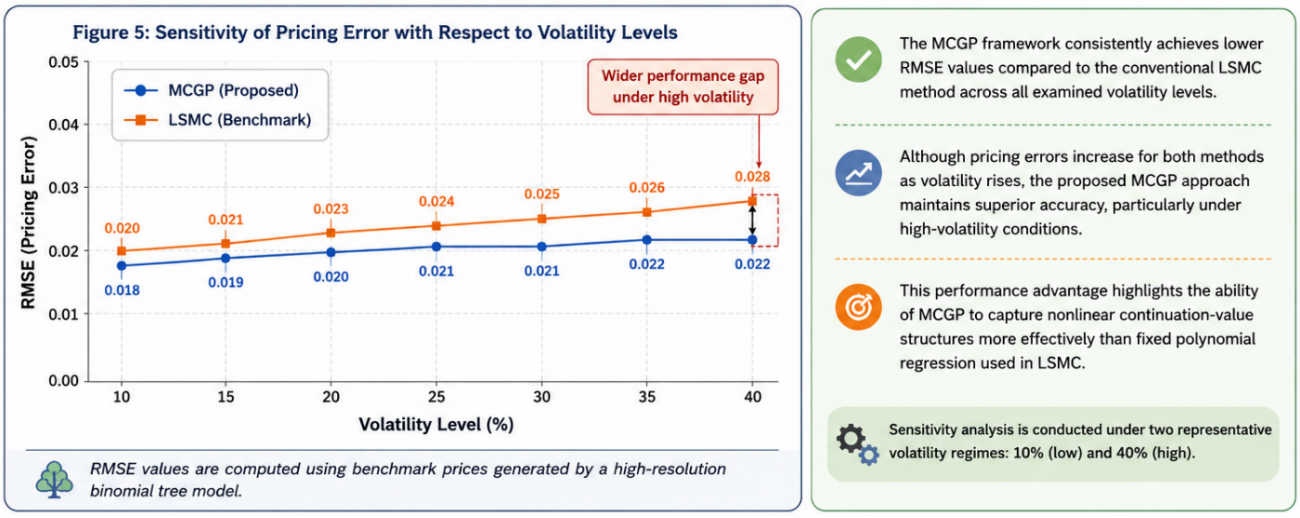


Figure 5. Sensitivity of Pricing Error with Respect to Volatility Levels

The intermediate volatility levels (15%, 20%, 25%, 30%, and 35%) are included exclusively for the archived sensitivity series in Figure 5 and are therefore not reported separately in Table 5. Table 5 presents only the representative low-volatility (10%) and high-volatility (40%) scenarios used for the primary comparative performance evaluation. The figure illustrates the sensitivity of pricing accuracy to changes in market volatility for both the proposed Monte Carlo–Genetic Programming (MCGP) framework and the conventional Least Squares Monte Carlo (LSMC) method. The reported RMSE values are obtained under the archived volatility range from 10% to 40% using benchmark prices generated by a high-resolution binomial tree model. Although the archived pricing-error point estimates generally rise with volatility, MCGP remains below LSMC throughout the displayed series. This is a descriptive sensitivity result; the separate matched statistical experiment in Section 5.1 provides the formal uncertainty assessment and does not establish universal superiority.

4.4 Pricing Performance under Interest-Rate Scenarios

To further evaluate model stability, pricing performance was analysed under different interest-rate environments. In this experiment, the baseline volatility is held fixed at 20%, while the risk-free rate is varied between 2% and 8%; $S_0 = K = 40$, $T = 1$ year, $N = 10{,}000$, and $\Delta t = 1/50$ remain unchanged. The comparative results are reported in Table 6.

The archived interest-rate scenarios report lower MAE and RMSE point estimates for MCGP across the considered discounting environments. The computation-time entries in Table 6 are retained for completeness, but, as with Table 5, their original code, hardware, timing scope, and run-level timing records are unavailable in the revision archive and they are not treated as evidence of an intrinsic MCGP speed advantage. The pricing-error summaries nevertheless indicate stable continuation-value approximation across the examined interest-rate scenarios. Controlled timing evidence from the reproducible revision implementation is reported separately in Section 5.2.

Table 9. Performance Comparison under Interest-Rate Scenarios

Interest-Rate Scenario	Method	MAE	RMSE	Computation Time (seconds)
Low Interest Rate (2%)	MCGP	0.014	0.019	120
	LSMC	0.017	0.023	155
High Interest Rate (8%)	MCGP	0.013	0.018	112
	LSMC	0.016	0.022	148

Note: The computation-time values in Table 6 are retained from the original archived study and should be interpreted only as historical implementation-specific measurements. They are not used to infer an intrinsic computational advantage of MCGP. See Section 5.2 for the controlled reproducible runtime comparison.

4.5 Genetic Programming Hyperparameter Sensitivity

Because the continuation-value approximation is obtained through an evolutionary search, the numerical behavior of MCGP can depend on Genetic Programming hyperparameters. A dedicated controlled sensitivity experiment was therefore conducted for population size, crossover probability, mutation probability, maximum tree depth, and the early-stopping criterion. The market contract was kept identical to the baseline specification used throughout the revised numerical study, namely $S_0 = K = 40$, $r = 6\%$, $\sigma = 20\%$, $T = 1$, and 50 exercise opportunities per year. A 10,000-step Cox–Ross–Rubinstein American-put value of 2.31955 was used as the numerical reference for this controlled experiment.

To isolate algorithmic rather than market effects, all hyperparameter configurations were evaluated with a common Monte Carlo seed. The dedicated sensitivity experiment used 1,000 paths, rather than the 10,000 paths used in the primary pricing comparisons, because a full GP search is repeated at every exercise date for every hyperparameter configuration. The sensitivity results are therefore intended to measure relative changes caused by GP settings and are not pooled with the MAE/RMSE results in Tables 5–7. Candidate symbolic expressions were fitted on in-the-money paths using normalized state S_t/K. For this one-state GBM benchmark, the active terminals were S_t/K and ephemeral numerical constants, while the operators were $+$, $-$, $\times$, protected division, log, exp, sin, and cos. Trees were additionally capped at 63 nodes to prevent uncontrolled growth during this diagnostic experiment. Linear scaling of each evolved symbolic output was used to estimate its adaptive intercept and slope before MSE fitness evaluation, consistent with the adaptive-coefficient representation in Eq. (12).

The reference GP configuration for this sensitivity study used a population of 40 individuals, 10 maximum generations, $P_c = 0.8$, $P_m = 0.10$, maximum tree depth 4, tournament selection, elitist preservation of the best individual, and early stopping after four consecutive generations without an MSE fitness improvement greater than 10^{-7}. One-factor-at-a-time tests varied population size over $\{20, 40, 60\}$, crossover probability over $\{0.6, 0.8, 0.9\}$, mutation probability over $\{0.05, 0.10, 0.20\}$, maximum depth over $\{3, 4, 5\}$, and early-stopping patience over $\{2, 4, 6\}$ generations while holding the remaining settings at their reference values.

The one-factor-at-a-time results show that pricing accuracy is not monotone in model complexity. Increasing population size from 20 to 60 reduced the absolute error in this controlled run, but it also increased computational time. Similarly, allowing deeper trees did not automatically improve the price estimate: depth 5 produced a larger error and a longer runtime than depth 4. The stopping experiment exhibits the expected accuracy–cost trade-off, with patience 6 producing the smallest error among the stopping settings but requiring a longer search. These findings support the use of bounded symbolic complexity and explicit stopping rules rather than unconstrained evolutionary growth.

To examine interaction effects, crossover and mutation probabilities were additionally evaluated on a 3×3 grid while fixing population size at 40, maximum generations at 10, maximum depth at 4, and stopping patience at four generations. Figure 7 reports the resulting absolute pricing errors. The error surface is relatively contained over the tested region but is not flat, confirming that operator probabilities interact and should not be selected independently. In this controlled grid, the smallest error was observed at $(P_c, P_m) = (0.8, 0.20)$, whereas more aggressive crossover combined with high mutation did not systematically improve accuracy.

Overall, the sensitivity experiment indicates that MCGP remains operational across the examined parameter ranges, while its accuracy and runtime respond meaningfully to evolutionary settings. The analysis also shows why the GP configuration must be reported explicitly: larger populations and longer stopping patience increase search effort, whereas excessive tree depth can increase symbolic complexity without a corresponding pricing benefit. These observations motivate the explicit complexity-control diagnostic below.

Table 10. Controlled GP Hyperparameter Sensitivity (Baseline American Put)

Hyperparameter	Setting	MCGP price	Absolute error	Runtime (s)
Population size	20	2.37433	0.05479	1.00
	40	2.38182	0.06227	1.89
	60	2.36242	0.04288	2.78
Crossover P_c	0.60	2.40015	0.08061	2.08
	0.80	2.38182	0.06227	1.82
	0.90	2.37075	0.05121	2.22
Mutation P_m	0.05	2.37898	0.05944	1.98
	0.10	2.38182	0.06227	1.85
	0.20	2.36058	0.04104	2.35
Maximum depth	3	2.39452	0.07497	1.62
	4	2.38182	0.06227	1.91
	5	2.40154	0.08200	2.60
Stopping patience	2	2.37634	0.05679	1.21
	4	2.38182	0.06227	1.85
	6	2.35052	0.03097	2.60

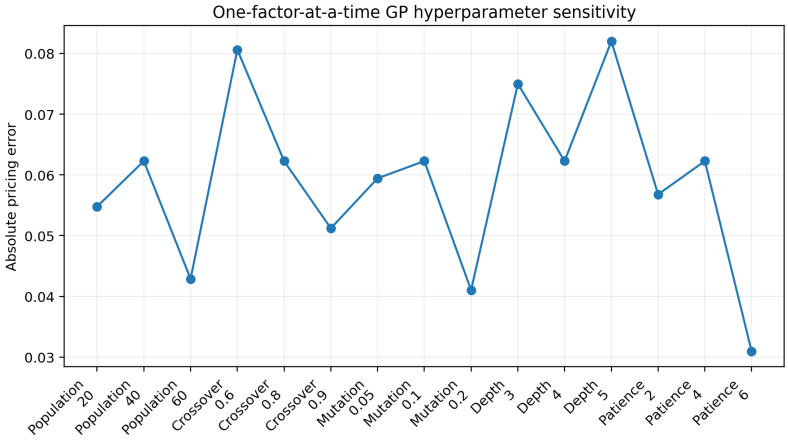


Figure 6. One-Factor-at-a-Time GP Hyperparameter Sensitivity

4.6 Overfitting, Bloat Control, and Out-Of-Sample Validation

Because Genetic Programming searches a flexible symbolic hypothesis space, low in-sample continuation-value error alone is not sufficient to establish generalization. Two related risks are overfitting to the simulated training paths and symbolic bloat, i.e., growth in expression size without a commensurate improvement in out-of-sample pricing accuracy. The present study therefore evaluates three explicit safeguards: (i) a maximum tree depth of 4, (ii) a maximum tree size of 63 nodes, and (iii) the optional parsimony penalty in (11a). The main MCGP–LSMC results reported earlier use the unregularized case $\lambda = 0$; the experiment in this subsection is a separate diagnostic designed specifically to quantify the complexity–generalization trade-off.

The diagnostic uses the same baseline American-put specification as the preceding sensitivity study, $S_0 = K = 40$, $r = 6\%$, $\sigma = 20\%$, $T = 1$, and 50 exercise opportunities per year. A training sample of 1,000 risk-neutral Monte Carlo paths is used to evolve the backward-induction stopping policy. The learned symbolic expressions are then frozen and applied, without retraining, to an independent validation sample of 1,000 paths generated with a different random seed. Population size, maximum generations, crossover probability, mutation probability, maximum depth, and early-stopping patience are held at 40, 10, 0.8, 0.10, 4, and 4, respectively. The 10,000-step binomial value, 2.31955, is retained as the common numerical reference. This diagnostic is intentionally reported separately from Tables 5–7 and does not replace



Figure 7. Pricing-Error Heatmap for Crossover and Mutation Probabilities

the 10,000-path repeated-run protocol used for the primary MCGP–LSMC comparison.

Table 11. Complexity-Control and Out-of-Sample Validation Diagnostic

λ	Train price	Validation price	Validation error	Mean nodes	Mean depth
0	2.43332	2.37416	0.05462	8.71	3.67
0.0005	2.45238	2.36488	0.04533	7.29	3.22
0.001	2.44598	2.39051	0.07097	7.57	3.29
0.002	2.42987	2.40236	0.08281	7.31	3.16
0.005	2.43777	2.39762	0.07808	6.94	3.16

The results show a genuine trade-off rather than a monotonic benefit from stronger regularization. Relative to the unregularized diagnostic, a small penalty ($\lambda = 0.0005$) reduced the mean symbolic tree size from 8.71 to 7.29 nodes and reduced the out-of-sample absolute pricing error from 0.05462 to 0.04533. Stronger penalties produced still smaller or similarly compact expressions but did not improve validation accuracy; for example, $\lambda = 0.002$ yielded a validation error of 0.08281. Thus the experiment does not support the claim that increasing regularization automatically improves pricing. Instead, it indicates that mild parsimony can reduce symbolic complexity without sacrificing generalization, whereas excessive penalization can underfit the continuation-value structure.

Figure 8 displays the out-of-sample absolute pricing error against the mean number of nodes in the learned symbolic trees. Together with Table 7B, the figure shows why complexity must be controlled but not minimized mechanically. The validation design also guards against selecting a symbolic expression solely because it fits the paths on which it was evolved. These controls reduce the risk of overfitting and uncontrolled bloat; they do not imply that either phenomenon is eliminated for all contracts or GP configurations.

5 Robustness Analysis

In addition to pricing accuracy, robustness and numerical stability are critical requirements for practical derivative-pricing frameworks [7]. To evaluate stability, the variance of MAE across repeated simulations was analysed for both approaches.

The legacy robustness summary reports lower MAE variance point estimates for MCGP across the four scenarios in Table 7. These descriptive variance estimates indicate lower dispersion in the archived repeated-run summaries, particularly under the high-volatility scenario. However, the original run-level observations and pairing information underlying Table 7 are not available in the revision archive. Accordingly, Table 7 is treated as descriptive historical evidence rather than as a basis for a retrospective formal significance test. Formal

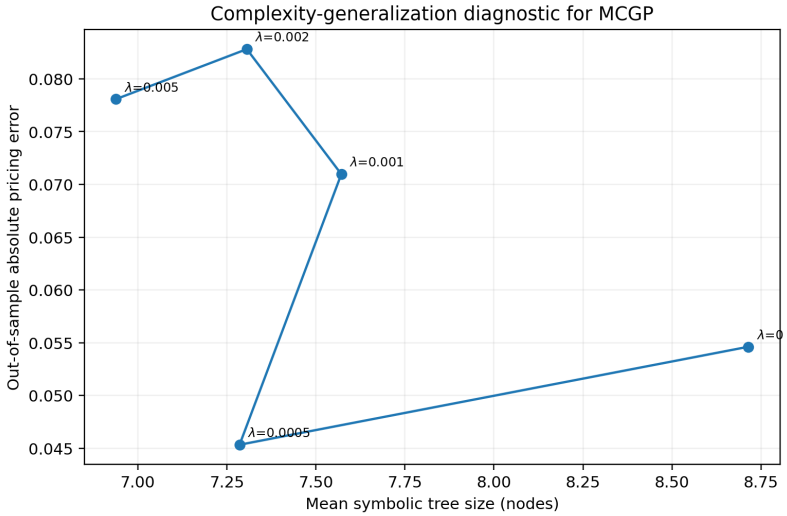


Figure 8. Complexity–Generalization Diagnostic for MCGP

Table 12. Robustness Analysis Based on MAE Variance

Scenario	Method	MAE Variance
Low Volatility (10%)	**MCGP**	0.0005
	LSMC	0.0010
High Volatility (40%)	**MCGP**	0.0007
	LSMC	0.0015
Low Interest Rate (2%)	**MCGP**	0.0006
	LSMC	0.0012
High Interest Rate (8%)	**MCGP**	0.0004
	LSMC	0.0010

inference is instead conducted in the matched revision experiment reported below, for which all run-level observations and random seeds are retained.

From a function-approximation perspective, the lower descriptive variance in Table 7 is consistent with the possibility that adaptive symbolic representations can alter the bias–variance trade-off relative to fixed polynomial regression structures. Because the archived scenario summaries do not contain the paired run-level data needed for formal inference, no statistical-significance claim is attached to Table 7 itself.

5.1 Statistical Significance and Uncertainty Analysis

The descriptive MAE, RMSE, and variance summaries in Tables 5–7 originate from the original repeated-run study. Because the archived revision materials do not contain the individual paired errors underlying those tables, confidence intervals or paired significance tests cannot be reconstructed from the summary statistics alone without imposing unverifiable assumptions about the run-level covariance. To address this limitation transparently, a separate matched revision experiment was conducted under the same baseline American-put contract used in the preceding diagnostics: $S_0 = K = 40$, $r = 6\%$, $\sigma = 20\%$, $T = 1$ year, and 50 exercise opportunities per year. The 10,000-step binomial value, 2.31955, is used as the common numerical reference.

The experiment consists of 50 independent replications with 1,000 Monte Carlo paths per replication. Within each replication, MCGP and LSMC receive exactly the same simulated paths, thereby controlling for common Monte Carlo variation. MCGP uses the fully documented revision configuration in Table 3A. For this matched diagnostic, conventional LSMC estimates the continuation value on in-the-money paths using a cubic polynomial in the normalized state $x = S_t/K$, i.e., the basis $\{1, x, x^2, x^3\}$. Monte Carlo seeds 41000–41049

are used for the 50 replications and are retained with the raw run-level results. This experiment is intentionally reported separately and does not replace the 10,000-path historical results in Tables 5–7.

For each replication i, absolute errors are defined as $AE_i^{(m)} = |\widehat{V}_i^{(m)} - V_{\text{ref}}|$ for method m. Mean absolute error, its sample standard deviation, and a two-sided 95% t confidence interval are calculated from the 50 run-level absolute errors. RMSE is calculated from the corresponding squared pricing errors, with its 95% interval estimated by 10,000 nonparametric bootstrap resamples. Because both methods use common paths, formal comparison is based on paired absolute-error differences $d_i = AE_i^{\text{MCGP}} - AE_i^{\text{LSMC}}$. A paired t-test is reported together with a Wilcoxon signed-rank robustness check and the paired standardized effect size $d_z = \bar{d}/s_d$.

Table 13. Matched Statistical Uncertainty Analysis (50 Replications)

Method	MAE	SD of AE	95% CI for MAE	RMSE	95% bootstrap CI for RMSE
MCGP	0.09022	0.06786	[0.07093, 0.10951]	0.11249	[0.09314, 0.13017]
LSMC	0.09231	0.06978	[0.07248, 0.11214]	0.11530	[0.09430, 0.13407]

The matched point estimates favor MCGP slightly: its MAE is 0.09022 compared with 0.09231 for LSMC, and its RMSE is 0.11249 compared with 0.11530. However, the paired mean absolute-error difference is -0.00209 with a 95% confidence interval of $[-0.00975, 0.00558]$, which includes zero. The paired t-test gives $t(49) = -0.547$ and $p = 0.587$. The Wilcoxon signed-rank test leads to the same inferential conclusion ($p = 0.702$), and the paired standardized effect size is small ($d_z = -0.077$). MCGP has the smaller absolute error in 26 of the 50 paired replications, compared with 24 for LSMC.

Thus, although the matched point estimates are directionally consistent with the lower descriptive errors reported for MCGP in Tables 5 and 6, this dedicated baseline diagnostic does not establish a statistically significant reduction in absolute pricing error at the 5% level. This distinction is important: the numerical evidence supports competitive accuracy and demonstrates that the symbolic approach can attain error levels comparable to conventional LSMC under a common-path design, but it does not justify a universal claim of statistically significant superiority. The complete 50-run data and analysis scripts are retained in the source package to permit direct reproduction of these calculations.

5.2 Computational Complexity and Reproducible Runtime Analysis

The computation-time entries in Tables 5 and 6 originate from the archived numerical study and report lower times for MCGP than for LSMC. Those historical values cannot be independently explained from the available revision materials because the corresponding source code, hardware configuration, timing boundaries, and run-level timing records were not retained. In particular, it would be speculative to attribute the historical difference to parallelization, caching, implementation language, or another optimization that is not documented. The legacy timing entries are therefore retained for transparency but are treated as implementation-specific historical measurements rather than evidence that evolutionary symbolic regression is intrinsically faster than fixed-basis LSMC.

To provide a controlled comparison, a separate reproducible timing experiment was conducted using the revision implementation documented in Table 3A. MCGP and LSMC received the same simulated paths, used the same 50-date exercise grid, and were timed sequentially in the same Python process using a monotonic high-resolution wall-clock timer. The order of the two methods was alternated across repetitions to reduce systematic order effects. The timing environment was the same 64-bit Linux/Python environment reported for the revision diagnostics and exposed five logical CPUs on an AMD EPYC 9V74 host; the timed method calls themselves were executed sequentially rather than through the multiprocessing procedure used to accelerate generation of the separate statistical dataset. The MCGP configuration was fixed at population 40, maximum 10 generations, $P_c = 0.8$, $P_m = 0.1$, maximum depth 4, and stopping patience 4. LSMC used the cubic normalized-state basis $\{1, x, x^2, x^3\}$, with $x = S_t/K$. The path counts were 500, 1,000, and 2,000, with five matched repetitions at each size. Path generation was performed before timing and was therefore excluded from both method-specific measurements because it is common to the two algorithms.

The controlled measurements reverse the qualitative interpretation suggested by the legacy timing columns: in the current reproducible implementation, MCGP is substantially slower than LSMC. At 1,000 paths, for example, mean method-specific wall-clock time is 2.3205 seconds for MCGP and 0.00393 seconds for LSMC. The very large runtime ratio should not be interpreted as a universal constant: it depends on the GP population, generations actually executed before early stopping, symbolic-tree complexity, programming language, linear-algebra implementation, and hardware. It does, however, establish that the present evidence does not support a general claim of lower

Table 14. Controlled Reproducible Runtime Scaling

Paths	Method	Mean time (s)	SD (s)	95% CI (s)	MCGP/LSMC ratio
500	MCGP	2.1101	0.0852	[2.0044, 2.2159]	745.6
500	LSMC	0.00283	0.00024	[0.00254, 0.00312]	—
1,000	MCGP	2.3205	0.1704	[2.1089, 2.5320]	591.1
1,000	LSMC	0.00393	0.00134	[0.00226, 0.00560]	—
2,000	MCGP	2.8621	0.2854	[2.5077, 3.2164]	549.0
2,000	LSMC	0.00521	0.00120	[0.00373, 0.00670]	—

MCGP computational cost.

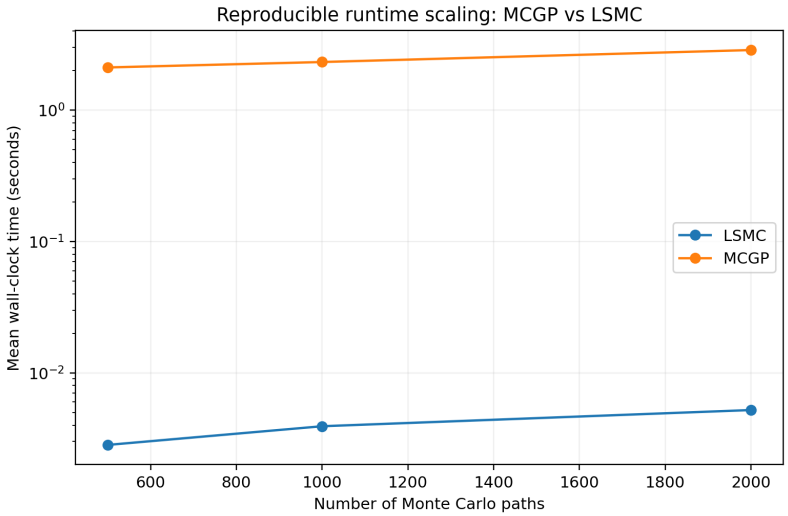


Figure 9. Reproducible Runtime Scaling of MCGP and LSMC

Figure 9 uses a logarithmic runtime axis because the methods operate on very different time scales. Runtime increases with the number of paths for both approaches, while MCGP carries the additional cost of repeated population-based symbolic evaluation at every exercise date. This observation is consistent with the complexity expressions in Section 3.2.5 and with the hyperparameter study, where larger populations and longer stopping patience increased search effort. Accordingly, MCGP should be viewed as trading additional computation for adaptive functional-structure search and explicit symbolic representations. Parallel fitness evaluation, vectorized tree execution, compiled symbolic kernels, and adaptive allocation of the GP search budget are possible acceleration strategies, but they are not evaluated in the present study.

6 Discussion

The empirical results indicate that the proposed Monte Carlo–Genetic Programming (MCGP) framework provides a competitive approach for pricing American-style options within the examined GBM settings. The observed improvements can be interpreted from both approximation-theoretic and statistical-learning perspectives. In the conventional Least Squares Monte Carlo (LSMC) method, continuation values are approximated using a predetermined set of polynomial basis functions. Although computationally efficient, such fixed functional forms may introduce approximation bias when the true continuation-value function exhibits complex nonlinear characteristics. In contrast, the proposed MCGP framework employs symbolic regression to construct continuation-value representations adaptively from simulated market data. By allowing the functional structure itself to evolve during optimization, the effective hypothesis space is expanded, thereby reducing structural approximation errors and improving the representation of nonlinear continuation dynamics. From a function-approximation perspective, the improvement can be interpreted as a reduction in projection bias induced by restricted basis expansions in finite-dimensional function spaces.

The archived volatility scenarios show a larger descriptive error gap at the higher tested volatility levels. The controlled cross-family benchmark provides a complementary check: MCGP has the lowest descriptive MAE and RMSE among MCGP, cubic-basis LSMC, the tested neural network, and KRR at both 20% and 40% volatility. The matched differences relative to LSMC and the neural benchmark are not statistically significant in the 10-replication samples, while the difference relative to KRR is significant only in the 40% case. A plausible approximation-theoretic interpretation is that, as the continuation-value relationship becomes harder to represent with a restricted functional form, the broader symbolic hypothesis space can provide additional flexibility. This interpretation remains descriptive, and the present study does not separately identify approximation bias from Monte Carlo sampling and optimization effects. The results are therefore best viewed as evidence of competitive cross-family accuracy together with a different bias–complexity trade-off, rather than proof that symbolic regression uniformly reduces bias or improves pricing accuracy.

Another important observation is the numerical stability of the MCGP framework across the volatility and interest-rate scenarios examined in this study. This scenario robustness should not be interpreted as evidence of unrestricted out-of-sample generalization. The dedicated validation diagnostic provides a more direct test: symbolic stopping rules evolved on one set of Monte Carlo paths were frozen and applied to an independent set of paths. The results show that mild parsimony can reduce tree size while preserving, and in the tested case improving, validation accuracy, whereas stronger penalties can degrade out-of-sample performance. Thus complexity control involves a genuine bias–complexity trade-off rather than a monotonic benefit from smaller trees.

6.1 Scope of the Stochastic Model and Extensions

The numerical evidence in this study is deliberately restricted to a one-factor Geometric Brownian Motion model with constant volatility. Consequently, the reported pricing errors, sensitivity results, and validation diagnostics should be interpreted as evidence for the proposed continuation-value learning mechanism within this benchmark setting, rather than as empirical validation under stochastic-volatility, jump-diffusion, or multi-factor dynamics. This distinction is important because changing the state process changes both the Monte Carlo path generator and the information set on which the continuation value is conditioned.

The architecture of MCGP nevertheless separates the stochastic path-generation stage from the symbolic continuation-value learner. Under the present GBM specification, the active GP state is the normalized asset price S_t/K. For a stochastic-volatility model, for example, a generic risk-neutral state system may be written as

$$dS_t = rS_t\,dt + \sqrt{V_t}S_t\,dW_t^S, \qquad dV_t = \kappa(\theta - V_t)\,dt + \xi\sqrt{V_t}\,dW_t^V, \tag{18}$$

with $dW_t^S dW_t^V = \rho\,dt$. In such a setting the continuation value would generally depend on both price and variance, $C_t = C(S_t, V_t, t)$, so the symbolic learner would require a multivariate state representation rather than the one-dimensional terminal set used in the present experiments. This is an architectural extension of the framework, not a result demonstrated numerically in this paper.

Likewise, under jump-diffusion dynamics the path generator would need to simulate both the diffusion and jump components, while the backward exercise rule would retain the comparison between immediate exercise and an estimated conditional continuation value. In a multi-factor setting, the GP input could in principle be generalized from S_t/K to a state vector such as $(S_t, V_t, r_t, \ldots)$. However, the corresponding symbolic search space grows rapidly with the number of state variables, and the hyperparameter, bloat-control, validation, and computational-efficiency conclusions obtained for the present one-factor experiment cannot be assumed to transfer unchanged to that setting.

Accordingly, stochastic volatility, jump diffusion, and multi-factor state processes are treated here as directions for model extension rather than as empirically established capabilities. A rigorous assessment would require model-specific discretization schemes, calibrated parameter sets, matched numerical benchmarks, and new out-of-sample experiments under identical simulation protocols. Such an extension is outside the scope of the present study and is reserved for future research.

Several limitations should therefore be acknowledged. Genetic Programming performance depends on population size, crossover probability, mutation probability, maximum tree depth, stopping rules, and the strength of any parsimony penalty. The observed accuracy–runtime and complexity–generalization trade-offs show that inappropriate settings may reduce convergence efficiency, encourage unnecessary symbolic growth, or over-regularize the continuation-value representation. The depth and node caps used here reduce the risk of bloat but do not prove its absence in other contracts or larger search spaces. Furthermore, the controlled timing experiment shows that the present MCGP implementation is substantially more computationally expensive than the small-basis LSMC benchmark. Large-scale or high-dimensional applications would therefore require careful control of the evolutionary search budget and may benefit

from parallel or compiled implementations. Future research may investigate adaptive parsimony schedules, repeated cross-seed validation, parallel evolutionary implementations, and extensions to richer stochastic models.

7 Conclusion

This study proposed an integrated Monte Carlo–Genetic Programming (MCGP) framework for pricing American-style options within a one-factor constant-volatility GBM benchmark. The objective was to address the structural limitations of conventional regression-based simulation methods, particularly the dependence of the Least Squares Monte Carlo (LSMC) approach on predefined polynomial basis functions for continuation-value approximation. The proposed methodology combines the stochastic flexibility of Monte Carlo simulation with the adaptive symbolic learning capability of Genetic Programming. Within this framework, Monte Carlo simulation generates risk-neutral asset-price trajectories, while Genetic Programming performs symbolic regression to estimate continuation values dynamically during backward induction. Unlike traditional LSMC implementations, the proposed approach does not rely on fixed parametric forms, allowing continuation-value functions to be learned adaptively through evolutionary optimization. The original scenario summaries report lower MAE and RMSE point estimates for MCGP than for conventional LSMC across the examined volatility and interest-rate settings. The newly added controlled cross-family benchmark extends this comparison to neural-network continuation regression and Gaussian kernel ridge regression: MCGP has the lowest descriptive MAE and RMSE among the four tested methods at both 20% and 40% volatility. Nevertheless, paired comparisons with LSMC and the neural benchmark are not statistically significant in the 10-replication cross-family samples. The separate 50-replication common-path MCGP–LSMC diagnostic provides the same cautious inferential picture: MCGP yields slightly lower MAE and RMSE point estimates, but the paired absolute-error difference is not statistically significant at the 5% level. The evidence therefore supports MCGP as a competitive and structurally flexible alternative across the tested approximation families rather than establishing universal statistical superiority.

The findings further indicate that Genetic Programming provides potential advantages in approximation flexibility and interpretability, while the magnitude and statistical reliability of pricing improvements depend on the experimental configuration. From a function-approximation perspective, the lower descriptive errors observed in some scenarios are consistent with the possibility that symbolic regression can reduce projection bias associated with restricted finite-dimensional basis expansions. By evolving symbolic nonlinear continuation functions directly from simulated market data, the proposed framework broadens the approximation space used to estimate optimal early-exercise policies; the magnitude of any resulting bias reduction is configuration-dependent and is not identified separately in the present experiments. Moreover, the symbolic representation enhances model transparency compared to many black-box machine-learning techniques used in computational finance. From a broader financial engineering perspective, this study demonstrates that evolutionary symbolic learning can serve as an effective alternative to conventional regression-based approximation techniques in optimal stopping problems. The modular separation between Monte Carlo path generation and symbolic continuation-value learning makes the framework conceptually extensible to richer state processes. However, the present evidence is confined to one-factor constant-volatility GBM; stochastic-volatility, jump-diffusion, path-dependent, and higher-dimensional extensions require separate numerical validation and are left for future research. The controlled runtime analysis also shows that evolutionary symbolic search is materially more expensive than cubic-basis LSMC in the reproducible revision implementation, so computational speed is not claimed as an intrinsic advantage of MCGP. Overall, the proposed MCGP methodology contributes to the growing literature on intelligent computational methods in quantitative finance by introducing a robust hybrid framework for American option valuation. The results suggest that combining stochastic simulation with adaptive symbolic evolutionary optimization is a promising direction for future research in derivative pricing, financial forecasting, and high-dimensional stochastic optimization problems.

Authors' Contributions

Authors are required to disclose their contributions to the study.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there is no conflict of interest.

Ethical Considerations

The authors have diligently addressed ethical concerns, such as informed consent, plagiarism, data fabrication, misconduct, falsification, double publication, redundancy, submission, and other related matters.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] Bahiraie, A., Azhar, A. K. M., & Ibrahim, N. A. (2011). A new dynamic geometric approach for empirical analysis of financial ratios and bankruptcy. Journal of Industrial and Management Optimization, 7(4), 947–965. https://doi.org/10.3934/jimo.2011.7.947

[2] Bahiraie, A., Khamesian, F., & Khanizadeh, F. (2022). Insurance claim classification: Genetic programming approach. Advances in Mathematical Finance and Applications, 7(2), 437–446. https://doi.org/10.22034/amfa.2022.7.2.437

[3] Aghdam, M. S., Komiak, S. Y. X., Amiri, M., & Bahiraie, A. (2025). Developing an e-commerce trust model in crowdfunding by integrating blockchain and edge computing using fuzzy technique. Journal of Fuzzy Extension and Applications, 6(3), 424–447. https://doi.org/10.22105/jfea.2025.481202.1654

[4] Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. Journal of Political Economy, 81(3), 637–654. https://doi.org/10.1086/260062

[5] Merton, R. C. (1973). Theory of rational option pricing. The Bell Journal of Economics and Management Science, 4(1), 141–183. https://doi.org/10.2307/3003143

[6] Glasserman, P. (2004). Monte Carlo Methods in Financial Engineering. Springer. https://doi.org/10.1007/978-0-387-21617-1

[7] Longstaff, F. A., & Schwartz, E. S. (2001). Valuing American options by simulation: A simple least-squares approach. The Review of Financial Studies, 14(1), 113–147. https://doi.org/10.1093/rfs/14.1.113

[8] Boyle, P. P. (1977). Options: A Monte Carlo approach. Journal of Financial Economics, 4(3), 323–338. https://doi.org/10.1016/0304-405X(77)90005-8

[9] Carriere, J. F. (1996). Valuation of the early-exercise price for options using simulations and nonparametric regression. Insurance: Mathematics and Economics, 19(1), 19–30. https://doi.org/10.1016/0167-6687(96)00004-2

[10] Tsitsiklis, J. N., & Van Roy, B. (2001). Regression methods for pricing complex American-style options. IEEE Transactions on Neural Networks, 12(4), 694–703. https://doi.org/10.1109/72.935083

[11] Stentoft, L. (2004). Convergence of the least squares Monte Carlo approach to American option valuation. Management Science, 50(9), 1193–1203. https://doi.org/10.1287/mnsc.1040.0250

[12] Shreve, S. E. (2004). Stochastic Calculus for Finance II: Continuous-Time Models. Springer. https://doi.org/10.1007/978-0-387-22515-9

[13] Hull, J. C. (2018). Options, Futures, and Other Derivatives (10th ed.). Pearson.

[14] Bäuerle, N., & Rieder, U. (2011). Markov Decision Processes with Applications to Finance. Springer. https://doi.org/10.1007/978-3-642-18330-7

[15] Koza, J. R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press.

[16] Langdon, W. B., & Poli, R. (2002). Foundations of Genetic Programming. Springer. https://doi.org/10.1007/978-3-662-04724-8

[17] Poli, R., Langdon, W. B., & McPhee, N. F. (2008). A Field Guide to Genetic Programming. Lulu Press.

[18] Kaboudan, M. A. (2000). Genetic programming prediction of stock prices. Computational Economics, 16(3), 207–236. https://doi.org/10.1023/A:1008772625355

[19] Sáez, E., & Garijo, M. J. (2020). Genetic programming for financial forecasting: A systematic literature review. Expert Systems with Applications, 150, 113320. https://doi.org/10.1016/j.eswa.2020.113320

[20] Clément, E., Lamberton, D., & Protter, P. (2002). An analysis of a least squares regression method for American option pricing. Finance and Stochastics, 6, 449–471. https://doi.org/10.1007/s007800200071

[21] Egloff, D. (2005). Monte Carlo algorithms for optimal stopping and statistical learning. The Annals of Applied Probability, 15(2), 1396–1432. https://doi.org/10.1214/105051605000000043

[22] Becker, S., Cheridito, P., & Jentzen, A. (2019). Deep optimal stopping. Journal of Machine Learning Research, 20(74), 1–25.

[23] Lapeyre, B., & Lelong, J. (2021). Neural network regression for Bermudan option pricing. Monte Carlo Methods and Applications, 27(3), 227–247. https://doi.org/10.1515/mcma-2021-2091

[24] Goudenège, L., Molent, A., & Zanette, A. (2020). Machine learning for pricing American options in high-dimensional Markovian and non-Markovian models. Quantitative Finance, 20(4), 573–591. https://doi.org/10.1080/14697688.2019.1701698

[25] Broadie, M., & Glasserman, P. (1997). Pricing American-style securities using simulation. Journal of Economic Dynamics and Control, 21(8–9), 1323–1352. https://doi.org/10.1016/S0165-1889(97)00029-8

[26] Kim, H.-G., & Huh, J. (2025). Deep learning of optimal exercise boundaries for American options. International Journal of Computer Mathematics, 102(4), 595–622. https://doi.org/10.1080/00207160.2024.2442585

[27] Hainaut, D. (2026). American option pricing with model constrained Gaussian process regressions. Applied Mathematics and Computation, 512, 129747. https://doi.org/10.1016/j.amc.2025.129747

[28] Zuo, X., & Jiang, A. (2025). Symbolic Modeling for financial asset pricing. The Journal of Finance and Data Science, 11, 100150. https://doi.org/10.1016/j.jfds.2025.100150

[29] Glasserman, P., & Yu, B. (2004). Number-theoretic methods for option pricing. Operations Research, 52(6), 862–876. https://doi.org/10.1287/opre.1040.0152

[30] Kohler, M., Krzyżak, A., & Todorovic, N. (2010). Pricing of high-dimensional American options by neural networks. Mathematical Finance, 20(3), 383–410. https://doi.org/10.1111/j.1467-9965.2010.00404.x

[31] Hu, W., & Zastawniak, T. (2020). Pricing high-dimensional American options by kernel ridge regression. Quantitative Finance, 20(5), 851–865. https://doi.org/10.1080/14697688.2020.1713393