Extensions to the Longstaff and Schwartz Method

Asger Damgaard-Sørensen

UNIVERSITY OF COPENHAGEN

The problem

- **American put option:** payoff $H_i = (K - S_i)^+$.
- **Stopping time τ:** choose exercise date to maximize expected discounted payoff

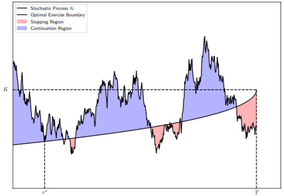
$$V_0 = \sup_{\tau \in \mathcal{T}} \mathbb{E}_Q\left[e^{-r\tau\Delta t} H_\tau\right].$$

- **Value function V_i:** optimal value at time t_i.
- **Snell envelope recursion:**

$$V_N = H_N, \quad V_i = \max\left\{H_i,\ e^{-r\Delta t}\,\mathbb{E}[V_{i+1} \mid \mathcal{F}_i]\right\}.$$

- **Interpretation:** Exercise if immediate payoff $\geq$ continuation value.
- **Goal:** Compute V_0 (the option price today) and characterize the exercise boundary.

Figure 1: Stylized Illustration of the Early Exercise Boundary



Note: Early exercise boundary for a process modeled as a geometric Brownian motion. By following the policy set by the early exercise boundary the holder of this specific option should exercise at τ^*

Simulation: GBM & Monte Carlo

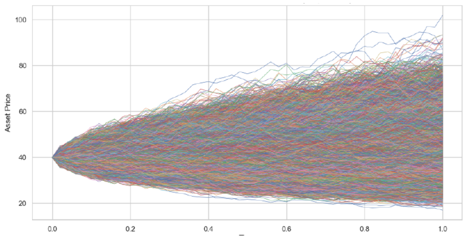
- **Geometric Brownian Motion (GBM):** model for the stock price under risk-neutral measure Q

$$dS_t = rS_t dt + \sigma S_t dW_t$$

- **Discretization:**

$$S_{t+\Delta t} = S_t \exp\Big((r-\tfrac{1}{2}\sigma^2)\Delta t+\sigma\sqrt{\Delta t}\,Z\Big), \quad Z \sim N(0$$

- **Monte Carlo (MC):** method to simulate many random paths following GBM where it draws Z.
- Each path $\Rightarrow$ one possible future stock price scenario.
- Expectations $\Rightarrow$ approximated by averaging discounted payoffs across all paths. Helping curse of dimensionality.



Longstaff–Schwartz Algorithm (LSM)

- **1. Simulate paths:** GBM stock price paths.
- **2. Terminal payoff/European option:** $V_n = H(S_T)$.
- **3. Backward induction:**
  - Identify in-the-money paths.
  - Discount cashflows.
  - Regress on basis functions $\phi_k(S)$.
- **4. Exercise decision:**
  - If payoff $\geq$ continuation $\Rightarrow$ exercise.
  - Else continue.
- **5. Monte Carlo estimate:** Average payoffs across all paths $\Rightarrow$ option price.

Algorithm 1 Longstaff-Schwartz

Input: Simulated paths $S_{t_i}^m$ for $i = 0, \dots, n$, $m = 1, \dots, M$
$V_n^m \leftarrow H(S_{t_n}^m)$ ▷ Terminal payoff
for $i = n-1$ **downto** 0 **do**
  $\mathcal{M}_i \leftarrow \{m : H(S_{t_i}^m) > 0\}$ ▷ Set of in-the-money paths at time t_i
  if $|\mathcal{M}_i| > 0$ **then**
    $X_m \leftarrow [\phi_0(S_{t_i}^m), \dots, \phi_{K-1}(S_{t_i}^m)]$ for $m \in \mathcal{M}_i$
    $Y_{i+1}^m \leftarrow e^{-r\Delta t} \cdot V_{i+1}^m$ for $m \in \mathcal{M}_i$
    Estimate $\beta_{i,k}$ by regressing Y_{i+1}^m on X_m
    for $m = 1$ **to** M **do**
      $\widehat{C}_i^m \leftarrow \sum_{k=0}^{K-1} \beta_{i,k} \cdot \phi_k(S_{t_i}^m)$
      if $H(S_{t_i}^m) \geq \widehat{C}_i^m$ **then**
        $V_i^m \leftarrow H(S_{t_i}^m)$
        $\tau^m \leftarrow t_i$
      else
        $V_i^m \leftarrow e^{-r\Delta t} \cdot V_{i+1}^m$
$\widehat{V}_0 \leftarrow \frac{1}{M} \sum_{m=1}^{M} V_{\tau^m}^m$
return $\widehat{V}_0$

Key Functions

- **Payoff:**
$$H(s) = (K - s)^+$$
- **Value function (Snell):**
$$V_i(s) = \max\{H(s), C_i(s)\}$$
- **Continuation value:**
$$C_i(s) = e^{-r\Delta t}\,\mathbb{E}^Q[V_{i+1}(S_{t_{i+1}}) \mid S_{t_i} = s]$$
- **Basis functions:**
$$\phi_k(S), \quad k = 0, \dots, K-1$$
- **Regression (approx.):**
$$\hat{C}_i^m = \sum_{k=0}^{K-1} \beta_{i,k}\,\phi_k(S_{t_i}^m)$$
- **Exercise rule:**
$$V_i^m = \begin{cases} H(S_{t_i}^m), & \text{if } H(S_{t_i}^m) \geq \hat{C}_i^m \\ e^{-r\Delta t} V_{i+1}^m, & \text{else} \end{cases}$$
- **MC estimate:**
$$\hat{V}_0 = \frac{1}{M}\sum_{m=1}^{M} V_{\tau^m}^m$$

Basis Functions in Practice

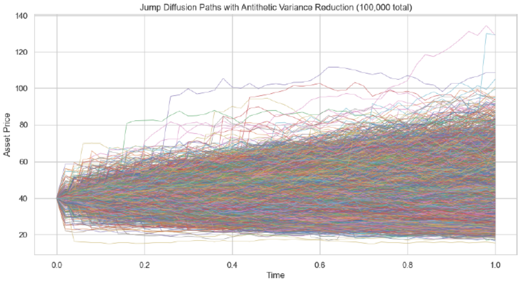
- At each time t_i, for in-the-money paths:
$$Y_{i+1}^m = e^{-r\Delta t} V_{i+1}^m$$
are regressed on basis functions $\phi_k(S_{t_i}^m)$.
- Using e.g. L_0, L_1, L_2 means:
$\hat{C}_i^m = \beta_{i,0} L_0(S_{t_i}^m) + \beta_{i,1} L_1(S_{t_i}^m) + \cdots + \beta_{i,K} L_K(S_{t_i}^m)$.
- All functions are combined; regression chooses the coefficients $\beta_{i,k}$.
- **Weights (coefficients):** Found by OLS regression at every t:
$$\boldsymbol{\beta}_i = (X^\top X)^{-1} X^\top Y$$
with $X =$ matrix of basis functions, $Y =$ discounted payoffs.

- **Interpretation:** Basis functions span a function space. Regression picks the best linear combination.
- **Where:** Approximation is done at each time step t_i, only using in-the-money paths.
- **Goal:** Get a smooth estimate of continuation value $C_i(s)$ to compare with immediate payoff.

QMC and Jump Diffusion

- **Quasi-Monte Carlo (QMC):**
  - Uses low-discrepancy Sobol sequences.
  - Sobol draws fill the space more evenly than random draws(MC).
  - Provides more uniform coverage $\Rightarrow$ lower variance where MC can create clusters leading to over/under sampling.
- **Jump Diffusion:**
  - Extends GBM with jumps to capture sudden price changes.
  - Typical form:

$$dS_t = rS_t dt + \sigma S_t dW_t + J_t S_t dN_t$$



Simulation with jump diffusions

with $N_t \sim \text{Poisson}(\lambda)$, $J_t = \text{jump size}$.

Basket Options and Curse of Dimensionality

- **Multi-asset simulation:** Basket options depend on several underlying assets.
- **Correlation:** Introduced via **Cholesky decomposition** of covariance matrix Σ:

$$Z = L \cdot \epsilon, \quad \epsilon \sim N(0, I), \quad \Sigma = LL^\top$$

- **Tensor-product basis functions:** For n assets and degree d:

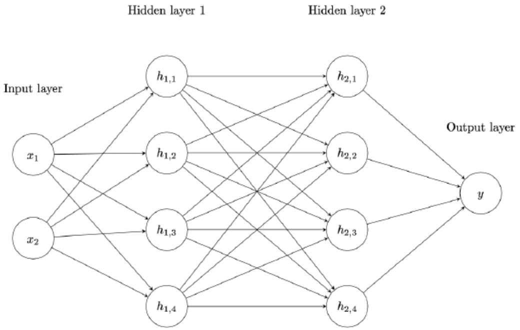
$$\varphi_\alpha(S) = \prod_{j=1}^{n} P_{\alpha_j}(\tilde{S}^{(j)}), \quad \alpha = (\alpha_1, \ldots, \alpha_n),\ \alpha_j \in [0, d].$$

$\Rightarrow (d+1)^n$ basis functions.

- Captures both single-asset effects and cross-asset interactions.
- But number of basis functions explodes with n (dimension) and d (polynomial degree).
- **Curse of dimensionality:**
  - Simulation of paths still feasible due to MC.
  - Approximation/regression step becomes intractable for more than 4 assets.

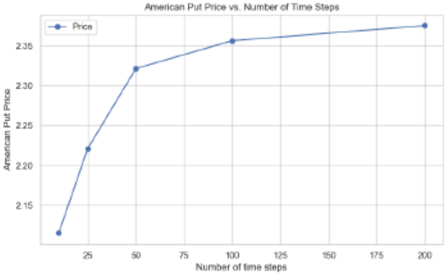
Neural Network

- **Goal:** replace polynomial regression with a NN at each t_i.
- **Input:** state S_{t_i} (vector for baskets).
- **Output:** $\hat{C}_i = f_{\theta_i}(S_{t_i})$ (estimate of "value if we wait").
- **Layers (in picture):**
  - **Input layer:** features S_{t_i} (one node per asset).
  - **Hidden layer 1/2:** weighted sums $\Rightarrow$ combine information and learn patterns..
  - **Output layer:** single number = continuation value $\hat{C}_i$.
- **Train:** ITM paths; targets $Y = e^{-r\Delta t} V_{i+1}$; minimize MSE with Adam; **learning rate** $\approx 10^{-3}$.
- **Decision:** exercise if $H(S_{t_i}) \geq \hat{C}_i$; else continue.
- **Setup:** 2 hidden layers (∼40 units)



Convergence of Longstaff–Schwartz Method

(a) Number of paths vs price



(b) Number of time steps for 1 periode vs price

- **Paths vs. Price:**
  - Price stabilizes around ∼30,000 paths.
  - 100,000 paths ⇒ stable and precise.
  - Fluctuations beyond this are marginal.
- **Time Steps vs. Price:**
  - Price rises with more steps.
  - Converges near 50 time steps.
  - Captures early exercise behavior well.

Polynomial Degree Convergence (Theory)

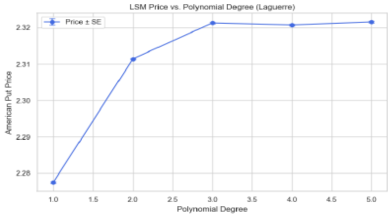
High-degree terms contribute very little.
Judd (Theorem 6.4.2): For a smooth function $f \in C^k[-1,1]$, it has a Chebyshev expansion

$$f(x) = \tfrac{1}{2}c_0 + \sum_{j=1}^{\infty} c_j T_j(x),$$

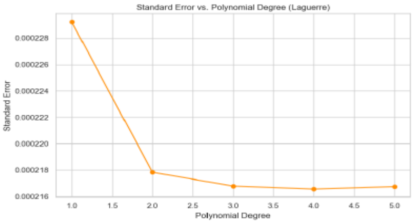
with coefficients satisfying

$$|c_j| \leq c \cdot j^{-k}.$$

- f is smooth: theory (GBM + Snell) ensures continuity, and the convergence results.
- Same idea applies to Laguerre polynomials in LSM as both Chebyshev and Laguerre are orthogonal polynomials.
- Thus 3 degrees are sufficient.



(a) LSM vs. Polynomial Degrees



(b) Std. error vs. Polynomial Degrees

Basis Function Type (1D)

- Compared Laguerre, Hermite, Chebyshev (degree 3).
- All give very similar prices ≈ 2.32 with tiny s.e.
- Computation times also almost identical.
- Reason: In 1D, continuation value is smooth and easy to approximate.
- Different polynomial families $\Rightarrow$ all span rich function spaces.
- Differences only matter more in higher dimensions.

Basis	Price	(s.e.)	Time
Laguerre	2.3213	(0.0002)	33.3s
Hermite	2.3177	(0.0002)	32.6s
Chebyshev	2.3209	(0.0002)	33.1s

Different Simulation Methods

- **QMC (Sobol):** More uniform sampling of ITM region ⇒ more stable regressions, slightly higher prices.
- **MC (GBM):** Random sampling, noisier regressions ⇒ downward bias in price (lower than QMC).
- **Jump Diffusion:** Downward jumps ⇒ earlier exercise, higher put value. Reduces computation time (paths end earlier).
- **Takeaway:** All methods stable (low s.e.), but QMC and JD give more realistic or efficient results.

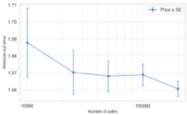
Method	Price	s.e.	Time
QMC (GBM)	2.5369	0.0002	43.9s
Jump Diffusion	2.4867	0.0003	20.9s
MC (GBM)	2.3213	0.0002	43.0s

Two Assets: Convergence and Basis Functions

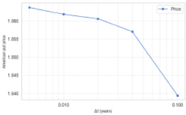
- All three polynomial bases (Laguerre, Chebyshev, Hermite) give the same price and std. error.
- Only difference: small variation in computation time.
- Confirms consistency across basis choices in 2D.

Poly	Price	s.e.	Time	Euro
Laguerre	1.6609	0.0064	3s	1.4103
Chebyshev	1.6609	0.0064	2s	1.4103
Hermite	1.6609	0.0064	4s	1.4103

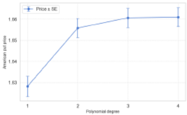
Convergence: Assets



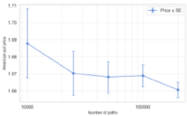
(a) Chebyshev: Monte Carlo



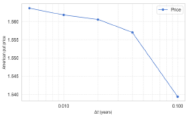
(b) Chebyshev: Time-step



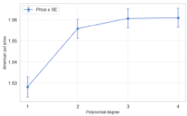
(c) Chebyshev: Basis-degree



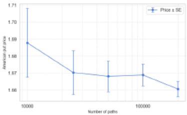
(d) Laguerre: Monte Carlo



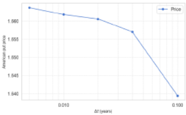
(e) Laguerre: Time-step



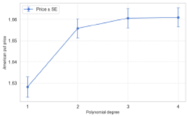
(f) Laguerre: Basis-degree



(g) Hermite: Monte Carlo



(h) Hermite: Time-step



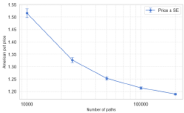
(i) Hermite: Basis-degree

Five Assets: Basis Functions and Convergence

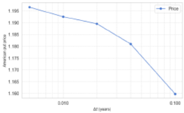
- Prices differ across polynomials (Laguerre ≈ 1.200, Chebyshev ≈ 1.211, Hermite ≈ 1.202).
- Standard errors small $\Rightarrow$ differences are structural, not noise.
- Computation $\sim$ 2.5 minutes, grows non-linearly with more assets.
- Issues:
  - No stable convergence across degrees (especially Chebyshev).
  - Some MC convergence, but large changes remain when adding paths.
  - Grid refinement (time-steps) has negligible effect.
- Neural Network (MLP) tested, but fails: price below European $\Rightarrow$ infeasible.

Poly	Price	s.e.	Time	Euro
Laguerre	1.2003	0.0046	146s	0.9205
Chebyshev	1.2106	0.0046	160s	0.9205
Hermite	1.2021	0.0046	140s	0.9205
MLP	0.9148	0.0029	147s	0.9205

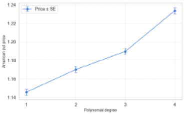
Convergence: Assets



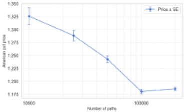
(a) Chebyshev: Monte Carlo



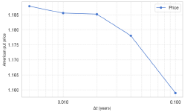
(b) Chebyshev: Time-step



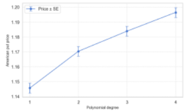
(c) Chebyshev: Basis-degree



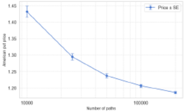
(d) Laguerre: Monte Carlo



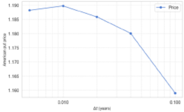
(e) Laguerre: Time-step



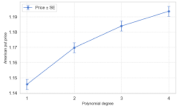
(f) Laguerre: Basis-degree



(g) Hermite: Monte Carlo



(h) Hermite: Time-step



(i) Hermite: Basis-degree

Alternative Pricing Methods

- **Numerical quadrature (Gauss–Hermite, Judd 1998):** Efficient and accurate in 1D, fast convergence for smooth payoffs. But costly with finer time grids and impractical in higher dimensions (curse of dimensionality).
- **Reinforcement methods (LSPI, FQI):** Potential alternatives, but computationally expensive and no clear accuracy gains. Not investigated further.
- **Sparse grid interpolation (Yang & Li 2025):** Maps asset space to bounded domain, uses high-order sparse grids. Allows pricing basket options up to 16 assets. Scales better than regression, but requires smoothness and more complex to implement.

Discussion of results & Concluding Remarks

- Convergence achieved very well with Longstaff and Schwartz framework.
- QMC (Sobol) reduces regression noise $\rightarrow$ higher price.
- Jump diffusion increases put price and reduces runtime due to volatility.
- In $>$5D the computational time becomes extreme $\rightarrow$ curse of dimensionality.
- Neural network test failed (price $<$ European). This can be because of insufficient network depth, suboptimal hyperparameter tuning, or overfitting to local regions of the state space without capturing the global exercise boundary.