# Hand-In Exercise #2

Continuous Time Finance 2 (FinKont2)

**Yasin Baysal (cmv882)**

Mathematics-Economics
University of Copenhagen
Block 3, 2025

# Hand-In Exercise #2

Yasin Baysal (cmv882)
University of Copenhagen

## Question 1

To replicate Table 1 of Poulsen (2025) with volatility estimates for the Longstaff & Schwartz example, we begin by introducing the dataset and the derived log-returns. In the example, we have 8 simulated stock price paths, each observed at 4 discrete time points, given in Table 1 below:

| Path | $t=0$ | $t=1$ | $t=2$ | $t=3$ |
|---|---|---|---|---|
| 1 | 1.00 | 1.09 | 1.08 | 1.34 |
| 2 | 1.00 | 1.16 | 1.26 | 1.54 |
| 3 | 1.00 | 1.22 | 1.07 | 1.03 |
| 4 | 1.00 | 0.93 | 0.97 | 0.92 |
| 5 | 1.00 | 1.11 | 1.56 | 1.52 |
| 6 | 1.00 | 0.76 | 0.77 | 0.90 |
| 7 | 1.00 | 0.92 | 0.84 | 1.01 |
| 8 | 1.00 | 0.88 | 1.22 | 1.34 |

**Table 1:** Stock price paths from the Longstaff & Schwartz' example.

From these prices, we derive one-year log-returns that capture the continuously compounded return over each period. Specifically, the log-return between two successive time points is defined as

$$\xi_{3(i-1)+j} = \ln\left(\frac{S[i,j+1]}{S[i,j]}\right) \text{ for } i = 1,\dots,8 \text{ and } j = 1,2,3,$$

where $S[i,j]$ denotes the stock price for the $i$th simulated path at time $j$. Since each of the 8 paths produces 3 log-returns (one for each consecutive pair of time points), the total number of log-returns is defined by $M := 8 \cdot 3 = 24$. This set of 24 one-year log-returns serves as the basis for our two volatility estimation methods: realized volatility estimator and maximum likelihood estimator.

### Realized Volatility

The first way to estimate $\sigma$ is to take the standard deviation of the log-returns, which we will denote by $\sigma_{RV}$, where RV naturally stands for realized volatility. This realized volatility estimator is computed as the sample standard deviation (using the unbiased estimator), which is given by

$$\hat{\sigma}_{RV} = \sqrt{\frac{1}{M-1}\sum_{i=1}^{M}(\xi_i - \bar{\xi})^2}, \text{ where } \bar{\xi} = \frac{1}{M}\sum_{i=1}^{M}\xi_i,$$

where we have re-indexed the log-returns as $\{\xi_i\}_{i=1}^{24}$ for notational convenience after collecting all 24 returns. By Ahn and Fessler (2003), the standard error (SE) is then approximately estimated as

$$\text{SE}(\hat{\sigma}_{RV}) \approx \frac{\hat{\sigma}_{RV}}{\sqrt{2(M-1)}}.$$

Looking at the code for Question 1 below, this method has been implemented in the `estimate_volatility_RV` function of the `VolatilityEstimator` class. However, the drawback of this approach is that it does not fully adhere to the parametric assumptions of the Black–Scholes model, since the realized-volatility estimator treats log-returns simply as an i.i.d. sample from which the standard deviation is calculated, without explicitly incorporating the Black–Scholes drift term $\left(r - \frac{1}{2}\sigma^2\right)$, which leads us to the next method for estimating $\sigma$.

### Maximum Likelihood Estimator

By Poulsen (2025), the Black–Scholes model with stock prices is modelled as Geometric Brownian Motions

$$S(u) = S(t)e^{\left(r-\frac{1}{2}\sigma^2\right)(u-t)+\sigma(W(u)-W(t))} \text{ for any } u, t \text{ for which } u \geq t,$$

where $W$ is a Brownian Motion under the risk-neutral probability measure $Q$, while the one-period log-returns are i.i.d. distributed with common normal distribution, i.e., they are modelled as

$$\xi \overset{Q}{\sim} \mathcal{N}\left(\mu(\sigma), \sigma^2\right), \text{ with } \mu(\sigma) = r - \tfrac{1}{2}\sigma^2,$$

where $r = 0.06$ is the given continuously compounded rate, such that it becomes a single-parameter normal statistical model. Since we observe that the unknown parameter $\sigma$ appears both in the mean $\mu(\sigma)$ and also in the variance $\sigma^2$, we adopt a maximum likelihood approach to account for these dependencies.

By Definition 1.15 in Lauritzen (2023), the likelihood function for the $M$ independent and identically normally distributed log-returns is given by

$$L_M(\sigma) = \prod_{i=1}^{M} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\tfrac{(\xi_i-\mu(\sigma))^2}{2\sigma^2}\right).$$

By then taking the logarithm of this likelihood function, it gives us the log-likelihood function by

$$\ell_M(\sigma) = \log L_M(\sigma) = -M\ln(\sigma) - \frac{M}{2}\ln(2\pi) - \frac{1}{2\sigma^2}\sum_{i=1}^{M}\left(\xi_i - \left(r - \frac{1}{2}\sigma^2\right)\right)^2.$$

Since $\sigma$ appears in both the mean and the variance, setting the likelihood (or score) equation $\frac{\partial \ell}{\partial \sigma} = 0$ does not yield a closed-form solution. Thus, the estimator $\hat{\sigma}_{ML}$ is obtained by numerical maximization or, equivalently, by minimizing the negative log-likelihood, which we do. Following Poulsen (2025), we do this by evaluating the second derivative of the log-likelihood function at its optimum, taking the negative reciprocal of that value, and then taking the square root. So, for the observed information at the MLE defined as

$$I\left(\hat{\sigma}_{ML}\right) = -\left.\frac{\partial^2 \ell}{\partial \sigma^2}\right|_{\sigma=\hat{\sigma}_{ML}},$$

the standard error for the MLE is then approximated by

$$\mathrm{SE}\left(\hat{\sigma}_{ML}\right) \approx \frac{1}{\sqrt{I\left(\hat{\sigma}_{ML}\right)}}.$$

In the code for Question 1 below, this method has been implemented in the `estimate_volatility_ML` function of the `VolatilityEstimator` class. Here, we first apply a bounded optimization procedure to the negative of the log-likelihood function over a range for $\sigma \in [0.1, 0.3]$, yielding the $\sigma$ that maximizes the log-likelihood that is taken to be the maximum-likelihood estimate $\hat{\sigma}_{ML}$. Once $\hat{\sigma}_{ML}$ is obtained, we approximate the second derivative of $\ell_M(\sigma)$ at $\sigma = \hat{\sigma}_{ML}$ using a finite-difference method by choosing a small increment $\varepsilon = 0.001$ and computing $\ell(\hat{\sigma}_{ML} + \varepsilon)$, $\ell(\hat{\sigma}_{ML})$, and $\ell(\hat{\sigma}_{ML} - \varepsilon)$. The second derivative is then approximated by

$$\left.\frac{\partial^2 \ell}{\partial \sigma^2}\right|_{\sigma=\hat{\sigma}_{ML}} \approx \frac{\ell(\hat{\sigma}_{ML} + \varepsilon) - 2\,\ell(\hat{\sigma}_{ML}) + \ell(\hat{\sigma}_{ML} - \varepsilon)}{\varepsilon^2}.$$

Since the observed information is $I(\hat{\sigma}_{ML}) = -\left.\frac{\partial^2 \ell}{\partial \sigma^2}\right|_{\sigma=\hat{\sigma}_{ML}}$ by Definition 1.20 in Lauritzen (2023), we obtain

$$I(\hat{\sigma}_{ML}) \approx -\frac{\ell(\hat{\sigma}_{ML} + \varepsilon) - 2\,\ell(\hat{\sigma}_{ML}) + \ell(\hat{\sigma}_{ML} - \varepsilon)}{\varepsilon^2}.$$

Finally, the standard error of the estimate is computed as the square root of the reciprocal of this information, as stated earlier. The two estimation methods applied to the Longstaff & Schwartz example yield the results shown in Table 2, fully replicating Table 1 in Poulsen (2025) as desired.

| **Method** | **Estimate** | **Standard error** |
|---|---|---|
| $\sigma_{ML}$ | 0.1484 | 0.021 |
| $\sigma_{RV}$ | 0.1519 | 0.022 |

**Table 2:** Volatility estimates for the Longstaff & Schwartz' example.

As also noted by Poulsen (2025), the ML estimator's standard error is slightly smaller than that of the realized volatility estimator. This can be explained by the Cramér–Rao lower bound from Remark 4.6 in Lauritzen (2023), which states that the unbiased estimator $\hat{\sigma}_{ML}$ for $\sigma_{ML}$ satisfies

$$\mathrm{Var}(\hat{\sigma}) \geq \frac{1}{E_\sigma[I(\sigma)]},$$

where $E_\sigma[I(\sigma)] =: i(\sigma)$ is the Fisher information cf. page 15 in Lauritzen (2023). Thus, since the ML estimator is asymptotically efficient, it attains this lower bound in the Cramér–Rao inequality and cannot be outperformed in terms of standard error by any other unbiased estimator (cf. page 101 in Lauritzen (2023)).

```
library(R6)

VolatilityEstimator <- R6Class("VolatilityEstimator",
  public = list(
    S = NULL,
    r = NULL,
```

```
    paths = NULL,
    steps = NULL,
    log_returns = NULL,
    M = NULL,

    initialize = function(S, r) {
      self$S <- as.matrix(S)
      self$r <- r
      dims <- dim(self$S)
      self$paths <- dims[1]
      self$steps <- dims[2]
      self$log_returns <- self$compute_log_returns()
      self$M <- length(self$log_returns)
    },

    compute_log_returns = function() {
      log_rets <- c()
      for (i in 1:self$paths) {
        for (j in 1:(self$steps - 1)) {
          log_rets <- c(log_rets, log(self$S[i, j+1] / self$S[i, j]))
        }
      }
      return(log_rets)
    },

    estimate_volatility_RV = function() {
      sigma_RV <- sd(self$log_returns)
      sigma_RV_se <- sigma_RV / sqrt(2 * (self$M - 1))
      return(list(sigma_RV = sigma_RV, sigma_RV_se = sigma_RV_se))
    },

    log_likelihood = function(sigma) {
      x <- self$log_returns
      mu <- self$r - 0.5 * sigma^2
      N <- length(x)
      ll <- -N * log(sigma) - 0.5 * N * log(2 * pi) -
            sum((x - mu)^2) / (2 * sigma^2)
      return(ll)
    },

    estimate_volatility_ML = function() {
      nll <- function(sigma) {
        -self$log_likelihood(sigma)
      }
      opt <- optimize(nll, interval = c(0.1, 0.3))
      sigma_ML <- opt$minimum
      eps <- 1e-3
      ll_plus <- self$log_likelihood(sigma_ML + eps)
      ll_minus <- self$log_likelihood(sigma_ML - eps)
      ll_current <- self$log_likelihood(sigma_ML)
      second_deriv <- -(ll_plus - 2 * ll_current + ll_minus) / (eps^2)
      sigma_ML_se <- sqrt(1 / second_deriv)
      return(list(sigma_ML = sigma_ML, sigma_ML_se = sigma_ML_se))
    }
  )
)

S <- matrix(c(
  1.00, 1.09, 1.08, 1.34,
  1.00, 1.16, 1.26, 1.54,
```

```
  1.00, 1.22, 1.07, 1.03,
  1.00, 0.93, 0.97, 0.92,
  1.00, 1.11, 1.56, 1.52,
  1.00, 0.76, 0.77, 0.90,
  1.00, 0.92, 0.84, 1.01,
  1.00, 0.88, 1.22, 1.34
), nrow = 8, byrow = TRUE)

r <- 0.06
estimator <- VolatilityEstimator$new(S, r)
rv_results <- estimator$estimate_volatility_RV()
ml_results <- estimator$estimate_volatility_ML()

(results <- data.frame(
  Method           = c("sigma_{ML}", "sigma_{RV}"),
  Estimate         = c(sprintf("%.4f", ml_results$sigma_ML),
                       sprintf("%.4f", rv_results$sigma_RV)),
  "Standard Error" = c(sprintf("%.4f", ml_results$sigma_ML_se),
                       sprintf("%.4f", rv_results$sigma_RV_se))
))

##       Method Estimate Standard.Error
## 1 sigma_{ML}   0.1484         0.0213
## 2 sigma_{RV}   0.1519         0.0224
```

## Question 2

To replicate Table 2 of Poulsen (2025) with standard Black-Scholes model put option prices connected to the Longstaff & Schwartz example, we begin with the Black–Scholes framework for a put option and then discretize it via a binomial tree to approximate American and Bermudan put option prices.

First, under the Black–Scholes model, the stock price $S(t)$ follows the stochastic process given by

$$dS(t) = rS(t)dt + \sigma S(t)dW(t),$$

where $r$ is the risk-free interest rate, $\sigma$ the volatility and $W(t)$ the standard Brownian motion under the risk-neutral measure $Q$. For a European put option, we know that the terminal payoff at time $T$ is simply given by $V_T = (K - S(T))^+ = \max(K - S(T), 0)$ with $K$ as the strike price.

To approximate American and Bermudan options (which can be exercised before $T$), we use a discrete-time binomial tree, which is a stepwise approximation to the continuous Black–Scholes model cf. section 7.1 in Lando and Poulsen (2023). In the binomial model, price movements follow a binomial distribution, which converges to the log-normal distribution of Black–Scholes as the number of steps grows. For European options without dividends, the binomial model price converges to the Black–Scholes formula as the time steps increase.

Using section 5.5 in Lando and Poulsen (2023), we partition the time interval $[0, T]$ be split into $n$ distinct

dates with length $\Delta t = T/n$, and we denote these these time points by $t_i = i\Delta$, i.e. $\{0, T/n, 2T/n, \ldots, T\}$, while $S(0)$ is the current price at time 0. The dynamics of the stock price path are then given by

$$S(t_{i+1}) = S(t_i) \cdot \begin{cases} u := \exp(\alpha\Delta t + \sigma\sqrt{\Delta t}) & \text{with probability } q \\ d := \exp(\alpha\Delta t - \sigma\sqrt{\Delta t}) & \text{with probability } 1-q, \end{cases}$$

where $\sigma$ is volatility, and $\alpha$ is a constant we set to zero due to its negligible effect on the price cf. Lando and Poulsen (2023). The price at $t_i$ is $S(t_i) = S(0)u^j d^{i-j}$, where $j$ is the number of upward movements. Since $q$ is the probability of an upward movement and $1-q$ of a downward one, they can be calculated by using

$$q = \frac{R-d}{u-d} = \frac{e^{r\Delta t} - d}{u-d},$$

where $r$ is the risk-free rate. For an option with payoff $h(\cdot)$, we use a recursive algorithm to price it by

$$\begin{cases} h(S(0)u^j d^{T-j}) & \text{for} \quad t = T \\ e^{-r\Delta t}\left(qV_{j+1}(t+1) + (1-q)V_j(t+1)\right) & \text{for} \quad t \leq T-1, \end{cases}$$

where $V_{j+1}(t+1)$ and $V_j(t+1)$ are the option values at the upward and downward nodes in the next time step. At each node, we compare the immediate exercise value to the expected continuation value. If $h(\cdot)$ is the highest value, exercising the option immediately is optimal. Here we specifically have $h(x) = (K-x)^+$ as the payoff of the put option, such that at maturity the put payoff at a node with $j$ upward moves is

$$h(S(0)u^j d^{T-j}) = (K - S(0)u^j d^{T-j})^+.$$

If the option is American, it can be exercised at every node, so we compare the continuation value with the immediate payoff $h$ as in the recursion above. For Bermudan options, exercise is only allowed at certain times (e.g. once per year), so we set the immediate payoff to zero at nodes that are not exercise dates.

We have implemented the above procedure in the function `binomial_bermudan`, which can be seen below:

```
binomial_bermudan <- function(S0, capT, strike, r, sigma, n, m,
                              divyield = 0.0, opttype = 2) {
  dt <- capT / n
  u  <- exp(sigma * sqrt(dt))
  d  <- exp(-sigma * sqrt(dt))
  R  <- exp(r * dt)
  q  <- (R - (d + divyield * dt)) / (u - d)

  if (opttype == 1) {
    payoff <- function(x) pmax(x - strike, 0)
  } else {
    payoff <- function(x) pmax(strike - x, 0)
  }

  exponents <- 0:n
  prices <- payoff(S0 * (u^exponents) * (d^(n - exponents)))
```

```
  for (i in n:2) {
    exponents_i <- 0:(i - 1)
    St <- S0 * (u^exponents_i) * (d^((i - 1) - exponents_i))

    exercise_value <- if ((i - 1) %% m == 0) payoff(St) else numeric(length(St))
    cont <- (q * prices[2:(i + 1)] + (1 - q) * prices[1:i]) / R
    prices <- pmax(cont, exercise_value)
  }

  return((q * prices[2] + (1 - q) * prices[1]) / R)
}
```

For our simulations, we use the parameter values from the Longstaff & Schwartz example that are:

| Parameter | $S(0)$ | $K$ | $T$ | $r$ | $n$ |
|---|---|---|---|---|---|
| Value | 1 | 1.1 | 3 | 0.06 | $3 \cdot 252 = 756$ |

**Table 3:** Parameter values used for the simulations in the Longstaff & Schwartz example.

Three values of $\sigma$ are used: two of them are the ones calculated in Question 1, namely $\sigma_{ML} = 0.1484$ and $\sigma_{RV} = 0.1519$, while the last one is $\sigma = 0.1500$, which is the one used in Longstaff and Schwartz (2001). The algorithm has been implemented by allowing early exercise only at certain nodes:

- American: exercise allowed at every step ($m = 1$).
- Bermudan: exercise only allowed at specific dates (e.g. once per year, $m = 252$ steps per year).
- European: exercise allowed only at maturity ($m = n$).

Thus, the backward induction procedure on the tree yields option values $V(0)$ for these three cases. The resulting Black-Scholes put option prices in Table 4 roughly replicates Table 2 in Poulsen (2025) as desired.

| $\sigma$ | **American** | **Bermudan** | **European** |
|---|---|---|---|
| 0.1484 | 0.1085 | 0.0918 | 0.0627 |
| 0.1500 | 0.1092 | 0.0926 | 0.0637 |
| 0.1519 | 0.1099 | 0.0937 | 0.0649 |

**Table 4:** Put option prices under different volatilities $\sigma$ and exercise styles (American, Bermudan, European).

However, it is seen that some of the prices from the paper differ slightly (by 0.0001) from those produced by my implementation. This discrepancy originates from a minor error in the original code, where the condition $i \bmod m = 0$ is used to decide when early exercise is allowed. For a European option (with $m = n$), this condition erroneously permits exercise at $T - \Delta t$ since $n \bmod n = 0$. Changing the condition to $(i - 1) \bmod m = 0$ ensures that exercise is allowed only at the correct nodes, so that for a European option exercise occurs exclusively at maturity $T$ (same for the two Bermudan prices that also differ by 0.0001).

```
S0     <- 1
strike <- 1.1
capT   <- 3
```

```
n_steps <- 3 * 252

estimator <- VolatilityEstimator$new(S, r)
rv_est  <- estimator$estimate_volatility_RV()
ml_est  <- estimator$estimate_volatility_ML()

sigma_RV <- rv_est$sigma_RV
sigma_ML <- ml_est$sigma_ML

sigmas <- c(sigma_ML, 0.15, sigma_RV)
prices <- matrix(0, nrow = 3, ncol = 4)
colnames(prices) <- c("Sigma", "American", "Bermudan", "European")
prices[,1] <- sigmas

for (i in seq_along(sigmas)) {
  vol_i <- sigmas[i]
  am_price <- binomial_bermudan(S0, capT, strike, r, vol_i, n_steps, 1)
  bm_price <- binomial_bermudan(S0, capT, strike, r, vol_i, n_steps, 252)
  eu_price <- binomial_bermudan(S0, capT, strike, r, vol_i, n_steps, n_steps)

  prices[i, 2] <- am_price
  prices[i, 3] <- bm_price
  prices[i, 4] <- eu_price
}

print(round(prices, 4))

##       Sigma American Bermudan European
## [1,] 0.1484   0.1085   0.0918   0.0627
## [2,] 0.1500   0.1092   0.0926   0.0637
## [3,] 0.1519   0.1099   0.0937   0.0649
```

## Question 3

To assess the effect of different polynomial degrees on the LSM American put price, we start by outlining the LSM algorithm by Longstaff and Schwartz (2001), using the framework in Baysal (2024). Assume a complete probability space $(\Omega, \mathcal{F}, \mathbb{P})$ over $[0, T]$ with $K$ discrete times $0 < t_1 \leq t_2 \leq \cdots \leq t_K = T$, where $\Omega$ contains all economic outcomes from $t = 0$ to $t = T$ with $\omega_i$ being a sample path. We consider $\boldsymbol{F} = \{\mathcal{F}_t \mid t \in [0, T]\}$, where $\mathcal{F}_t = \mathcal{F}$ by assumption. Assuming a no-arbitrage EMM $\mathbb{Q}$, we examine options in $\mathcal{L}^2(\Omega, \mathcal{F}, \mathbb{Q})$, where

$$\mathbb{E}(h^2) = \int_\Omega h(\omega, \cdot)^2 \, \mathrm{d}P(\omega) < \infty.$$

We focus on square-integrable payoff functions $h(\omega, t_k)$. The asset price process $(S(t_k))_{k=0}^K$ is modeled as a Geometric Brownian Motion, with the payoff process $(Z(t_k))_{k=0}^K$ defined by $Z(t_k) = h(\omega, t_k)$. Since American options are continuously exercisable, we approximate them with many discrete time steps. At time $T$, the option is exercised if in-the-money; at each $t_k$, the holder decides based solely on the immediate payoff. Dynamic programming is used to determine the optimal stopping time $\tau^*$ from $\mathcal{T}(t_k) = \{t_k, t_{k+1}, \ldots, t_K\}$.

We divide the interval $[0, T]$ into $M$ steps at $0 = t_0, t_1, \ldots, t_M = T$, i.e. subintervals $[t_k, t_{k+1}]$ for $k =$

$1, \ldots, M-1$ with length $\Delta t = T/M$. At $t_M$, set the payoff using the final price. Then, moving backward from $t_{M-1}$, compute the option's value at each step by discounting the expected future cash flows to $t_k$. Let $V(\omega, t_k)$ denote the option value at $t_k$ for a sample path $\omega$; if exercise is deferred, the continuation value is

$$H(\omega, t_k) \equiv \mathbb{E}^Q(e^{-r\Delta t}V(\omega, t_{k+1}) \mid S(\omega, t_k)),$$

where $e^{-r\Delta t} = e^{-r(t_{k+1}-t_k)}$ is the discount factor, and $Q$ the risk-neutral measure. Immediate exercise yields payoff $Z(\omega, t_k)$. Since $\mathcal{L}^2$ is a Hilbert space with an inner product norm, it has a countable orthonormal basis cf. Longstaff and Schwartz (2001), so the expectation can be expressed as a linear function of this basis by

$$H(\omega, t_k) = \sum_{m=0}^{\infty} \phi_m(S(\omega, t_k))\beta_m(t_k).$$

Since $H(\omega, t_k)$ is unknown, we approximate it. We estimate each path's continuation value via least squares regression — comparing immediate exercise to the conditional expectation — and exercise when the former is higher from $t_{M-1}, t_{M-2}, \ldots$ to $t_1$. This regression determines optimal coefficients for a linear combination of orthonormal basis functions in $\mathcal{L}^2$. Let $H_M(\omega, t_k)$ denote the estimate using $M < \infty$ time steps as

$$H_M(\omega, t_k) = \sum_{m=0}^{M-1} \phi_m(S(\omega, t_k))\beta_m(t_k),$$

where $\{\phi_m(\cdot)\}_{m=0}^{M-1}$ forms a basis that we choose later and $\{\beta_m(\cdot)\}_{m=0}^{M-1}$ coefficients. When estimating this function, we apply regression, and with $N$ simulations, we can construct an approximation of $H_M(\omega, t_k)$ by

$$\hat{H}_M^N(\omega, t_k) = \sum_{m=0}^{M-1} \phi_m(S(\omega, t_k))\hat{\beta}_m^N(t_k),$$

where $\{\hat{\beta}_m^N(\cdot)\}_{m=0}^{M-1}$ are the estimated regression coefficients. Since the option's value at $T$ matches the option's intrinsic value, we iteratively do approximations to estimate the value functions for continuation via the this dynamic programming algorithm cf. the Snell envelope from section 2.5.1 in Lamberton and Lapeyre (2008):

$$\begin{cases} \tilde{V}_M^N(\omega, t_K) = Z(t_K) \\ \tilde{V}_M^N(\omega, t_k) = \max\left(Z(\omega, t_k), \tilde{H}_M^N(\omega, t_k)\right), \quad 1 < k \leq K-1. \end{cases}$$

Hence, if immediate payoff is greater than estimated continuation value, the option is exercised immediately; otherwise, the holder waits until the optimal exercise time is reached.

We use polynomial basis functions for the regression. For a degree $m$ (e.g., $m = 3, 4, 5$), they are given by

$$(\phi_0(x), \phi_1(x), \phi_2(x), \ldots, \phi_m(x)) = (1,\, x,\, x^2,\, \ldots,\, x^m),$$

where $x = S(\omega, t_k)$ is the asset price at time $t_k$. The regression coefficients $\hat{\beta}_j$ are computed via least squares:

$$\hat{\beta} = (\mathbf{\Phi}^\top \mathbf{\Phi})^{-1} \mathbf{\Phi}^\top \boldsymbol{Y},$$

where $\boldsymbol{Y} = (y_1, \ldots, y_N)$ for sample paths $\omega = 1, \ldots, N$ is the discounted option payoff with $y_\omega = e^{-r\Delta t} Z(\omega, t_k) = e^{-r\Delta t} h(S(\omega, t_k))$, and $\mathbf{\Phi} = \phi_m(x_\omega)$ for an underlying asset $x_\omega = S(\omega, t_k)$ at time $t_k$. Here, $h(\cdot)$ is the option payoff function expressed in terms of the option's payoff structure, i.e. put payoffs in this case.

Finally, we can estimate the option value at time 0 using Monte Carlo by

$$\tilde{V}_M^N(0) = \frac{1}{N} \sum_{n=1}^{N} e^{-r\Delta t} \tilde{V}(\omega_n, t_1),$$

which is the average of the discounted continuation values at $t_1$ over all $N$ paths $\omega_n$ given $S(\omega, t_0) = S$. The LSM algorithm has been implemented in the `LSM_AmericanPut` function below:

```r
LSM_AmericanPut <- function(S, strike, r, degree = 5, method = "lm") {
  paths <- nrow(S)
  steps <- ncol(S)
  disc <- exp(r * (-(1:(steps - 1))))

  P <- matrix(0, nrow = paths, ncol = steps)

  for (j in 1:paths) {
    P[j, steps] <- max(strike - S[j, steps], 0)
  }

  for (h in (steps - 1):2) {
    Y <- rep(0, paths)
    for (j in 1:paths) {
      Y[j] <- sum(disc[1:(steps - h)] * P[j, (h + 1):steps])
    }

    dummy <- strike - S[, h]
    pick <- dummy > 0
    if (sum(pick) == 0) next

    if (method == "matrix") {
      X <- cbind(1, sapply(1:degree, function(k) S[, h]^k))
      X_pick <- X[pick, ]
      Y_pick <- Y[pick]
      b <- solve(t(X_pick) %*% X_pick, t(X_pick) %*% Y_pick)
      predicted <- X %*% b
    } else if (method == "lm") {
      data_pick <- data.frame(Y = Y[pick], S = S[pick, h])
      model <- lm(Y ~ poly(S, degree, raw = TRUE), data = data_pick)
      predicted <- predict(model, newdata = data.frame(S = S[, h]))
    } else {
      stop("Invalid method specified. Choose 'matrix' or 'lm'.")
    }

    for (j in 1:paths) {
      if (pick[j] && (predicted[j] < max(dummy[j], 0))) {
        P[j, h] <- max(dummy[j], 0)
```

```
        P[j, (h + 1):steps] <- 0
      }
    }
  }

  temp <- rep(0, paths)
  for (j in 1:paths) {
    temp[j] <- sum(disc * P[j, 2:steps])
  }

  LSprice <- mean(temp)
  return(LSprice)
}
```

To assess the effect of higher-degree polynomials on LSM American put option price estimates, we have extended the regression by including basis functions

$$\{1, S_t, S_t^2, \ldots, S_t^d\}.$$

We then rerun the LSM algorithm for $d = 3, 4, 5$ using both the direct OLS matrix solution and R's `lm` function. We also include $d = 2$ as validation to ensure that we get the same price as in Poulsen (2025):

```
S <- matrix(c(1, 1.09, 1.08, 1.34,
              1, 1.16, 1.26, 1.54,
              1, 1.22, 1.07, 1.03,
              1, 0.93, 0.97, 0.92,
              1, 1.11, 1.56, 1.52,
              1, 0.76, 0.77, 0.90,
              1, 0.92, 0.84, 1.01,
              1, 0.88, 1.22, 1.34),
            nrow = 8, ncol = 4, byrow = TRUE)

degrees <- c(2, 3, 4, 5)
price_results <- data.frame(Degree = degrees, lm = NA, matrix = NA)

for (i in 1:length(degrees)) {
  d <- degrees[i]
  price_results$lm[i] <- LSM_AmericanPut(S, strike = 1.1, r = 0.06, degree = d, method = "lm")
  price_results$matrix[i] <- LSM_AmericanPut(S, strike = 1.1, r = 0.06, degree = d, method = "matrix")
}
```

```
## Warning in predict.lm(model, newdata = data.frame(S = S[, h])): prediction from rank-deficient fit; attr(*, "non-estim") has
doubtful cases
```

```
## Warning in predict.lm(model, newdata = data.frame(S = S[, h])): prediction from rank-deficient fit; attr(*, "non-estim") has
doubtful cases
```

```
## Error in solve.default(t(X_pick) %*% X_pick, t(X_pick) %*% Y_pick): system is computationally singular: reciprocal condition
number = 4.84555e-18
```

```
print(price_results)
```

```
##   Degree        lm    matrix
## 1      2 0.1144343 0.1144343
## 2      3 0.1154327 0.1154327
## 3      4 0.1242868 0.1242868
## 4      5 0.1242868        NA
```

In table 5 below with the LSM American put option prices using the two methods, we observe that for $d = 2, 3, 4$, both methods produce the same prices without errors. However, for $d = 5$, the `lm` approach returns a price but with a warning about a "rank-deficient fit"; the matrix-inversion approach fails completely with an error message indicating that the system is "computationally singular," leading to an `NA` price.

| Polynomial degree | OLS-formula in matrix form | Build-in `lm` function in R |
|---|---|---|
| 2 | 0.1144 | 0.1144 |
| 3 | 0.1154 | 0.1154 |
| 4 | 0.1243 | 0.1243 |
| 5 | NA | 0.1243 |

**Table 5:** Put option prices under different polynomial degrees $\sigma$ and methods (OLS matrix form, build-in `lm` function).

To explain this difference, we illustrate how a high-degree polynomial basis can cause this numerical instability when the number of in-the-money paths is small. Specifically, we use $d = 5$ (which requires 6 polynomial terms: $\{1, S, S^2, \ldots, S^5\}$) and examine a single time step $h = 3$. First, we specify the polynomial degree and time step, compute the immediate payoff, and identify in-the-money paths:

```
degree <- 5
h <- 3
dummy <- strike - S[, h]
pick <- dummy > 0

cat("At time step h =", h)

## At time step h = 3

cat("Asset values:", S[, h])

## Asset values: 1.08 1.26 1.07 0.97 1.56 0.77 0.84 1.22

cat("Immediate payoff (strike - S):", dummy)

## Immediate payoff (strike - S): 0.02 -0.16 0.03 0.13 -0.46 0.33 0.26 -0.12

cat("Number of in-the-money paths:", sum(pick))

## Number of in-the-money paths: 5
```

Next, we construct the polynomial (Vandermonde) design matrix $X$ of degree 5 for all paths at time $h$. Here, each row corresponds to one path, and the columns are $\{1, S, S^2, S^3, S^4, S^5\}$. We use a polynomial basis of degree $d = 5$, such that we have $1, S_h, (S_h)^2, (S_h)^3, (S_h)^4, (S_h)^5$. We also subset $X$ to $X_{\text{pick}}$ that only contains rows with in-the-money paths, which are used in the regression:

```
X <- cbind(1, sapply(1:degree, function(k) S[, h]^k))
cat("Full design matrix X (for all paths):")

## Full design matrix X (for all paths):

print(X)
```

```
##      [,1] [,2]   [,3]     [,4]      [,5]      [,6]
## [1,]    1 1.08 1.1664 1.259712 1.3604890 1.4693281
## [2,]    1 1.26 1.5876 2.000376 2.5204738 3.1757969
## [3,]    1 1.07 1.1449 1.225043 1.3107960 1.4025517
## [4,]    1 0.97 0.9409 0.912673 0.8852928 0.8587340
## [5,]    1 1.56 2.4336 3.796416 5.9224090 9.2389580
## [6,]    1 0.77 0.5929 0.456533 0.3515304 0.2706784
## [7,]    1 0.84 0.7056 0.592704 0.4978714 0.4182119
## [8,]    1 1.22 1.4884 1.815848 2.2153346 2.7027082
```

```
X_pick <- X[pick, ]
cat("Design matrix X_pick (in-the-money paths):")
```

```
## Design matrix X_pick (in-the-money paths):
```

```
print(X_pick)
```

```
##      [,1] [,2]   [,3]     [,4]      [,5]      [,6]
## [1,]    1 1.08 1.1664 1.259712 1.3604890 1.4693281
## [2,]    1 1.07 1.1449 1.225043 1.3107960 1.4025517
## [3,]    1 0.97 0.9409 0.912673 0.8852928 0.8587340
## [4,]    1 0.77 0.5929 0.456533 0.3515304 0.2706784
## [5,]    1 0.84 0.7056 0.592704 0.4978714 0.4182119
```

Since American options can be exercised early only if they are in-the-money, the LSM algorithm regresses the continuation value on these in-the-money states. If $X_{\text{pick}}$ has too few rows (paths) relative to the number of polynomial terms, linear dependence can arise cf. section 9.7 in Seber and Lee (2003). Therefore, we inspect the rank of $X_{\text{pick}}$ and compute its condition number to measure how close it is to being singular:

```
rank_X_pick <- qr(X_pick)$rank
cat("Rank of X_pick:", rank_X_pick, "out of", ncol(X_pick), "columns\n")
```

```
## Rank of X_pick: 5 out of 6 columns
```

```
cond_number <- kappa(X_pick)
cat("Condition number of X_pick:", cond_number, "\n")
```

```
## Condition number of X_pick: 205725.8
```

Here, $\text{rank}(X_{\text{pick}})$ tells us how many independent columns are in the design matrix. If $\text{rank}(X_{\text{pick}}) < d+1$, then at least one column is a linear combination of the others. The condition number $\kappa(X_{\text{pick}})$ is defined as

$$\kappa(X_{\text{pick}}) \;=\; \frac{\sigma_{\max}(X_{\text{pick}})}{\sigma_{\min}(X_{\text{pick}})}$$

by Definition 9.2 in Seber and Lee (2003), where $\sigma_{\max}$ and $\sigma_{\min}$ are the largest and smallest singular values of $X_{\text{pick}}$. Since we have a large $\kappa = 205725.8$, it indicates that $X_{\text{pick}}$ is rank-deficient and ill-conditioned, so inverting $X_{\text{pick}}^{\top} X_{\text{pick}}$ can lead to large numerical errors. As we also see that $\text{rank}(X_{\text{pick}}) = 5 < 6 = d+1$, it means that $X_{\text{pick}}^{\top} X_{\text{pick}}$ is singular , such that the OLS formula

$$\hat{\beta} = \left(X_{\text{pick}}^{\top} X_{\text{pick}}\right)^{-1} X_{\text{pick}}^{\top} Y_{\text{pick}}$$

cannot be computed via direct matrix inversion, resulting in the error: `"system is computationally singular: reciprocal condition number = ..."` When using `lm()` for the regression, R handles rank deficiency by dropping redundant columns. So, if $\text{rank}(X_{\text{pick}}) = 5$ even though $d = 5$, the effective polynomial degree is really only 4, so the fitted model and LSM price remain unchanged from the degree-4 case. As we still get a warning about getting predictions from a rank-deficient fit, the price estimate is not fully reliable.

## Question 4

We now suppose that the true data generating process is given by

$$dS(t) = \alpha S(t)dt + \sigma S(t)dW(t) = 0.06S(t)dt + 0.2S(t)dW(t).$$

Then, we want to run 10,000 similar experiments, where we simulate a 8-by-4 matrix of stock price paths in each of the experiments, and for that simulated matrix, we estimate the American (i.e., Bermudan to be more precise) put option price – just like it is done in the Longstaff & Schwartz example.

Since we note that the data generating process is a Geometric Brownian Motion (GBM) with drift $\alpha = 0.06$ and volatility $\sigma = 0.2$, Proposition 5.2 in Björk (2019) gives us its closed-form solution by

$$\begin{aligned} S(t) &= S(0)e^{\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W(t)} \\ &= S(0)e^{\left(0.06 - \frac{1}{2}(0.2)^2\right)t + 0.2W(t)} \\ &= S(0)e^{(0.06-0.02)t + 0.2W(t)} \\ &= S(0)e^{0.04t + 0.2W(t)} \end{aligned}$$

where $W$ is a Brownian Motion under the risk-neutral measure $Q$. Following (3.2) in Glasserman (2003) and using that increments are independent and normally distributed, we simulate $(W(t_1), \ldots, W(t_k))$ at times $0 < t_1 < \cdots < t_k$ using independent $Z_1, \ldots, Z_k \sim \mathcal{N}(0, 1)$. With $W(0) = 0$, we update recursively as

$$W(t_{i+1}) = W(t_i) + \sqrt{t_{i+1} - t_i}Z_{i+1}, \quad i = 0, \ldots, k-1.$$

Thus, we simulate paths of the GBM $S(t)$ using the following recursion

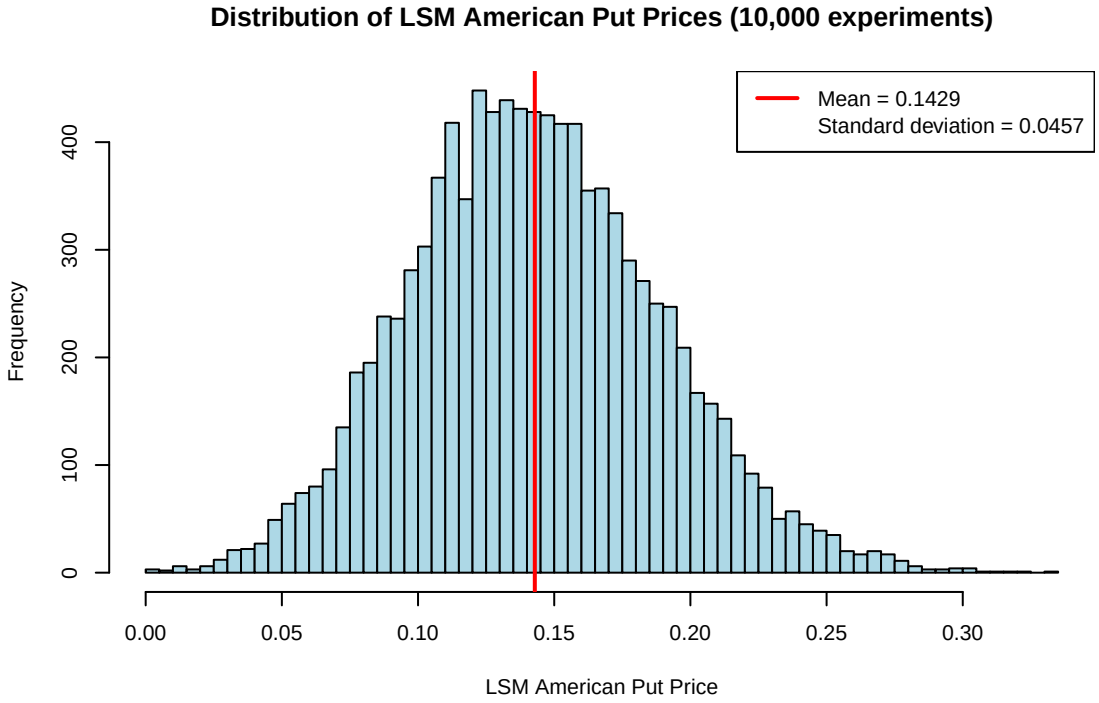
$$\begin{cases} S(t_0) = S(0) \\ S(t_{i+1}) = S(t_i)e^{\left(\mu - \frac{1}{2}\sigma^2\right)(t_{i+1}-t_i) + \sigma\sqrt{t_{i+1}-t_i}Z_{i+1}} = S(t_i)e^{0.04(t_{i+1}-t_i) + 0.2\sqrt{t_{i+1}-t_i}Z_{i+1}} \end{cases}$$

at times $0 = t_0 < \cdots < t_k = T$ for $i = 0, \ldots, N-1$ with random variables $Z_1, \ldots, Z_k$ i.i.d. $\mathcal{N}(0, 1)$.

Once these paths are generated, we estimate the American (Bermudan) put option price with $S(0) = 1$, $K = 1.1$, $r = 0.06$, $\sigma = 0.2$ and $T = 3$ by applying the LSM algorithm from the previous question (Question 3) with a polynomial degree of 2 in the regression. In this case, we take $N = 8$ paths and $k = 3$ time steps at

$t_1 = 1$, $t_2 = 2$, and $t_3 = 3$. Thus each experiment yields an $8 \times 4$ (with $t_0$) matrix of prices. We then repeat this process over 10,000 independent experiments, yielding 10,000 put prices. The code below shows the implementation with a histogram of the empirical price distribution, sample mean, and standard deviation:

```
set.seed(123)
n_experiments <- 10000
LSM_prices <- numeric(n_experiments)

S0 <- 1
strike <- 1.1
r <- 0.06
sigma_true <- 0.2
n_paths <- 8
n_steps <- 4
dt <- 1

for(exp in 1:n_experiments){
  S <- matrix(0, nrow = n_paths, ncol = n_steps)
  S[,1] <- S0
  for(j in 1:(n_steps - 1)){
    Z <- rnorm(n_paths)
    S[, j+1] <- S[, j] * exp((r - 0.5 * sigma_true^2) * dt + sigma_true * sqrt(dt) * Z)
  }
  LSM_prices[exp] <- LSM_AmericanPut(S, strike, r, degree = 2, method = "lm")
}

mean_price <- mean(LSM_prices)
sd_price <- sd(LSM_prices)

par(cex = 0.7)
hist(LSM_prices, breaks = 50, probability = FALSE,
     main = "Distribution of LSM American Put Prices (10,000 experiments)",
     xlab = "LSM American Put Price",
     col = "lightblue", border = "black")

abline(v = mean_price, col = "red", lwd = 2)

legend("topright", legend = c(paste("Mean =", round(mean_price, 4)),
                              paste("Standard deviation =", round(sd_price, 4))),
       col = c("red", "white"), lwd = 2)
```




The histogram shows a roughly bell-shaped distribution centered around 0.1429, which is the sample mean of the estimated prices, while the standard deviation is 0.0457. Even though each individual experiment only uses 8 paths, aggregating the results over 10,000 experiments produces an estimator whose distribution appears close to normal, consistent with central limit theorem. The moderate spread (standard deviation) reflects the variability from having relatively few paths (i.e., a small sample size) per experiment.

## Question 5

Now, we have to simulation experiments like in the previous question, but with the following twist: Keep the numerical values of the regression coefficients (i.e. the matrix $B$) fixed at the values from the Longstaff & Schwartz example. For each simulated path, $B$ determines an early exercise/stopping strategy. Then, we have to determine the expected discounted payoff from following that strategy.

The fixed regression coefficient matrix $B$ obtained in the Longstaff–Schwartz example is

$$B = \begin{pmatrix} 2.037512 & -1.069988 \\ -3.335443 & 2.983411 \\ 1.356457 & -1.813576 \end{pmatrix},$$

where the two columns correspond to the regression at the two exercise dates (first column for time $t_1$ and

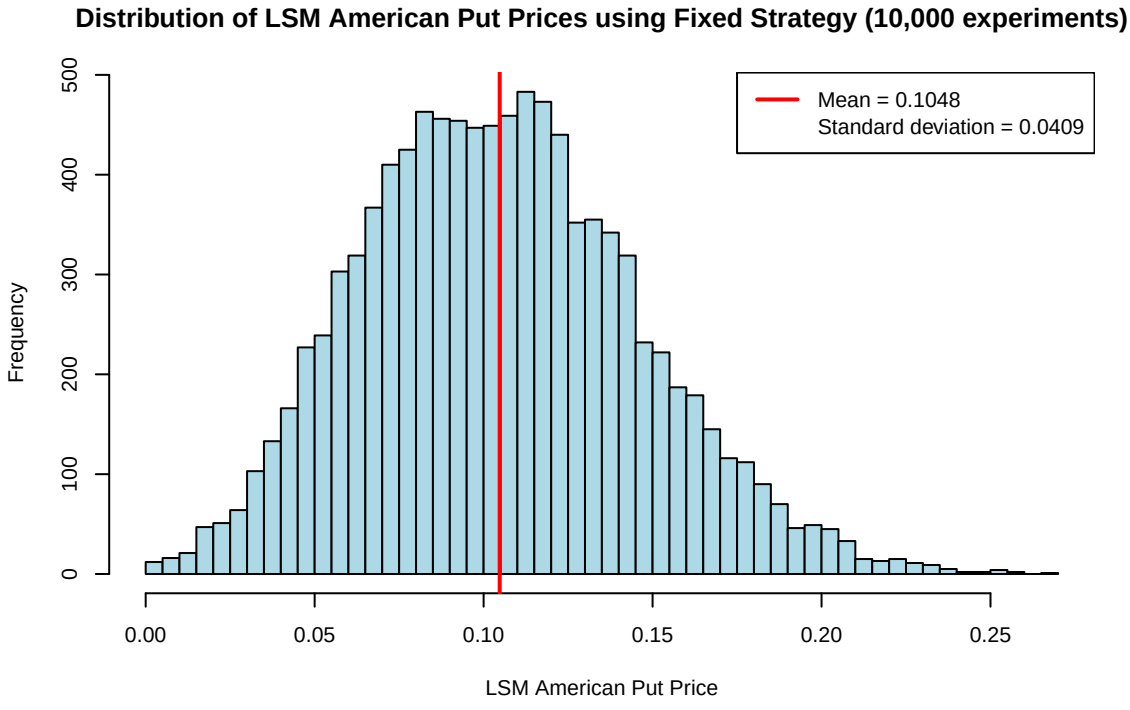
second column for time $t_2$; exercise is not allowed at time $t_0$). As in Question 4, we then start the simulation experiment by generating an $8 \times 4$ matrix of stock prices using the same GBM recursion.

Then, we apply the fixed strategy by using the fixed coefficients to compute the continuation value for each in-the-money path and decide whether to exercise immediately or to continue. For each path, we then determine the cash flow from the stopping strategy and discount it back to time 0 using the factor $e^{-r\Delta t} = e^{-0.06\Delta t}$ (applied over the periods until exercise). The option price is then the average of the discounted payoffs over the 8 paths. It has been implemented in `LSM_FixAmericanPut` below:

```
B_fixed <- matrix(c(2.037512, -3.335443, 1.356457, -1.069988,  2.983411, -1.813576), nrow = 3, ncol = 2)

LSM_FixedAmericanPut <- function(S, strike, r) {
  paths <- nrow(S)
  steps <- ncol(S)
  P <- matrix(0, nrow = paths, ncol = steps)
  P[, steps] <- pmax(strike - S[, steps], 0)

  for (h in (steps - 1):2) {
    immediate <- pmax(strike - S[, h], 0)
    X <- cbind(1, S[, h], S[, h]^2)

    if (h == steps - 1) {
      b_fixed <- B_fixed[, 2]
    } else if (h == 2) {
      b_fixed <- B_fixed[, 1]
    } else {
      stop("Unexpected time step.")
    }

    predicted <- as.vector(X %*% b_fixed)

    for (i in 1:paths) {
      if (immediate[i] > predicted[i]) {
        P[i, h] <- immediate[i]
        if (h < steps) {
          P[i, (h + 1):steps] <- 0
        }
      }
    }
  }

  payoff <- numeric(paths)
  for (i in 1:paths) {
    exer_time <- which(P[i, ] > 0)[1]
    if (is.na(exer_time)) exer_time <- steps
    payoff[i] <- P[i, exer_time] * exp(-r * (exer_time - 1))
  }

  price <- mean(payoff)
  return(price)
}
```

Repeating this over 10,000 experiments again provides the following distribution of option price estimates:

```r
set.seed(123)
n_experiments <- 10000
fixed_prices <- numeric(n_experiments)

for (exp in 1:n_experiments) {
  S <- matrix(0, nrow = n_paths, ncol = n_steps)
  S[, 1] <- S0
  for (j in 1:(n_steps - 1)) {
    Z <- rnorm(n_paths)
    S[, j + 1] <- S[, j] * exp((r - 0.5 * sigma_true^2) * dt + sigma_true * sqrt(dt) * Z)
  }
  fixed_prices[exp] <- LSM_FixedAmericanPut(S, strike, r)
}

mean_price_fixed_B <- mean(fixed_prices)
sd_price_fixed_B <- sd(fixed_prices)

par(cex = 0.7)
hist(fixed_prices, breaks = 50, probability = FALSE,
     main = "Distribution of LSM American Put Prices using Fixed Strategy (10,000 experiments)",
     xlab = "LSM American Put Price",
     col = "lightblue", border = "black")

abline(v = mean_price_fixed_B, col = "red", lwd = 2)

legend("topright", legend = c(paste("Mean =", round(mean_price_fixed_B, 4)),
                              paste("Standard deviation =", round(sd_price_fixed_B, 4))), col = c("red", "white"), lwd = 2)
```




We see that the expected discounted payoff from following that strategy is 0.1048 (with a standard deviation of 0.0409), which is lower than the 0.1429 that got by re-estimating the regression in Question 4. The fixed

regression coefficients come from a scenario with the volatily being $\sigma = 0.15$, whereas our simulation for the stock price paths uses $\sigma = 0.2$ instead in the dynamics. In effect, the "optimal" stopping rule is mis-specified. Because these fixed coefficients undervalue the higher continuation benefits at the higher volatility, the policy tends to exercise too early, which reduces the expected payoff relative to a more "optimal" strategy that would be derived if the regression were recalibrated to the actual simulated paths with $\sigma = 0.2$.

## References

Ahn, S. A. and Fessler, J. A. (2003). Standard errors of mean, variance, and standard deviation estima- tors. Working paper. Available at: https://web.eecs.umich.edu/~fessler/papers/files/tr/stderr.pdf.

Baysal, Y. (2024). American option pricing: A differential machine learning approach. Bachelor's thesis.

Björk, T. (2019). *Arbitrage Theory in Continuous Time*. Oxford Academic, 4th edition.

Glasserman, P. (2003). *Monte Carlo Methods in Financial Engineering*. Stochastic Modelling and Applied Probability. Springer New York, NY, 1st edition.

Lamberton, D. and Lapeyre, B. (2008). *Introduction to Stochastic Calculus Applied to Finance*. Chapman and Hall/CRC, 2nd edition.

Lando, D. and Poulsen, R. (2023). Finance 1 and beyond. Available at: https://www.dropbox.com/scl/fi/9txvcmdnsxls1spox5byl/MotherFile_RunMe.pdf?rlkey=peqyvbqb6a7kd04q6tr6jsm4h&e=1&dl=0.

Lauritzen, S. (2023). *Fundamentals of Mathematical Statistics*. CRC Press, 1st edition.

Longstaff, F. A. and Schwartz, E. S. (2001). Valuing American Options by Simulation: A Simple Least-Squares Approach. *The Review of Financial Studies*, 14(1):113–147.

Poulsen, R. (2025). Secrets of longstaff schwartz aged 23 3/4: Part i, the example. Available at: https://www.dropbox.com/scl/fi/2wd5ohoufw5od5knocff4/LSMat23andthreequarters.pdf?rlkey=3r42c8t795syb7ukwbx4n38mp&e=2&dl=0.

Seber, G. A. F. and Lee, A. J. (2003). *Linear Regression Analysis*. John Wiley Sons, Inc., 2nd edition.