Exercise – Understanding importance sampling using security price expectation

1. Aim of the exercise

The aim of this exercise is to understand weighted importance sampling in the context of reducing the variance of an integral estimate when pricing a security with uncertain future cash flows.

2. Theory

Suppose there are n securities with known prices. A new security, such as an option or asset, is introduced and defined by a random cash flow d to be received at the end of the period. Let the duration of one period be a single day, and denote R as the one-period risk-free return. What is the price of this new security at the end of the day?

One way to calculate the price P of a security with future payoff random variable $d(X)$, where $X \sim f_X(X)$, and risk-free return R, is by the discounted expectation (Luenberger, 2013):

$$P = \frac{\mathbb{E}[d(X)]}{R} = \frac{1}{R}\int_{-\infty}^{\infty} d(x) \cdot f_X(x)\, dx. \tag{1}$$

Suppose we know that the risk-free return is $R = 1.05$ and the random cash flow is characterized by the following function:

$$d : \mathbb{R} \to \mathbb{R}_{\geq 0}, \quad d(x) := \left| \sin\left(\frac{2\pi x}{21}\right) \cdot \cos\left(\frac{2\pi x}{e}\right) \right|. \tag{2}$$

The random variable X follows a Gamma distribution with shape parameter $\alpha = 1.2$ and rate parameter $\beta = 0.5$, denoted $X \sim \text{Gamma}(1.2, 0.5)$, and has probability density function $f_X(x)$. However, suppose we can only determine the distribution up to a normalizing constant. In other words, we observe the unnormalized density

$$X \sim \tilde{f}_X(x) := x^{0.2} \cdot e^{-\frac{1}{2}x} \propto f_X(x),$$

where $\tilde{f}_X(x)$ is proportional to the true density $f_X(x)$.

Since the distribution of $d(X)$ is only known up to a normalizing constant, we can use weighted importance sampling to compute the integral of interest in equation (1).

Let us choose the exponential distribution with rate $\frac{1}{2}$, denoted Expo $\left(\frac{1}{2}\right)$, as the proposal density:

$$p_X(x) = \frac{1}{2}e^{-\frac{1}{2}x}.$$

Hence, the importance weights are calculated as follows:

$$w(x) = \frac{\tilde{f}_X(x)}{p_X(x)} = \frac{x^{0.2} \cdot e^{-\frac{1}{2}x}}{\frac{1}{2}e^{-\frac{1}{2}x}} = 2x^{0.2}, \tag{3}$$

where x are realizations from a random sample $\{X_i\}_{i=1}^n$ drawn from the proposal distribution p_X.

The weighted importance sampling estimator is given by:

$$\hat{P}_{\text{IS},w} = \frac{1}{R}\sum_{i=1}^{n} w_i \cdot d(x_i), \tag{4}$$

where $w_i = \frac{w(x_i)}{\sum_{j=1}^{n} w(x_j)}$ are the normalized importance weights, and $x_i \sim p_X$ are realizations drawn from the proposal distribution. This estimator allows us to approximate the price of the new security even when the true density $f_X(x)$ is only known up to a constant.

The algorithm for weighted importance sampling was previously introduced in the theoretical discussion of this technique.

3. Clear the memory

Remove all existing variables from the workspace to ensure a clean computational environment.

```
%% 3. Clear the memory

% Clear the memory
clear;
```

4. Visualize the shape of the dividend function

In the MATLAB code, the dividend function is defined using an anonymous function syntax, where `@(x)` declares a function that takes a single input variable x. The function itself is composed of a sine term and a cosine term, each scaled by different frequency factors. The sine component oscillates with a period determined by $\frac{2\pi}{21}$, while the cosine component uses $\frac{2\pi}{e}$, where $e \approx 2.718$ is Euler's number. These two waveforms are multiplied element-wise using the `.*` operator, producing a composite oscillation. A sequence of x-values from -5 to 25 is generated in steps of 0.1, and the function is evaluated over this domain. The resulting values are plotted with labeled axes and a vertical range limited to $[-2, 2]$, effectively visualizing the fluctuating nature of the dividend function.

```
%% 4. Define dividend function and visualize its shape

% 4.1. Define dividend function as anonymous function
d_fun = @(x) sin((2*pi)/21.*x).*cos((2*pi)/exp(1).*x);

% 4.2. Create sequence for plotting
x = -5:0.1:25;

% 4.3. Evaluate dividend function over sequence
d = d_fun(x);

% 4.4. Create figure and plot dividend function over x
figure
plot(x,d)
title('Fig. 1. Dividend function')
xlabel('x')
ylabel('d')
ylim([-2 2])
```

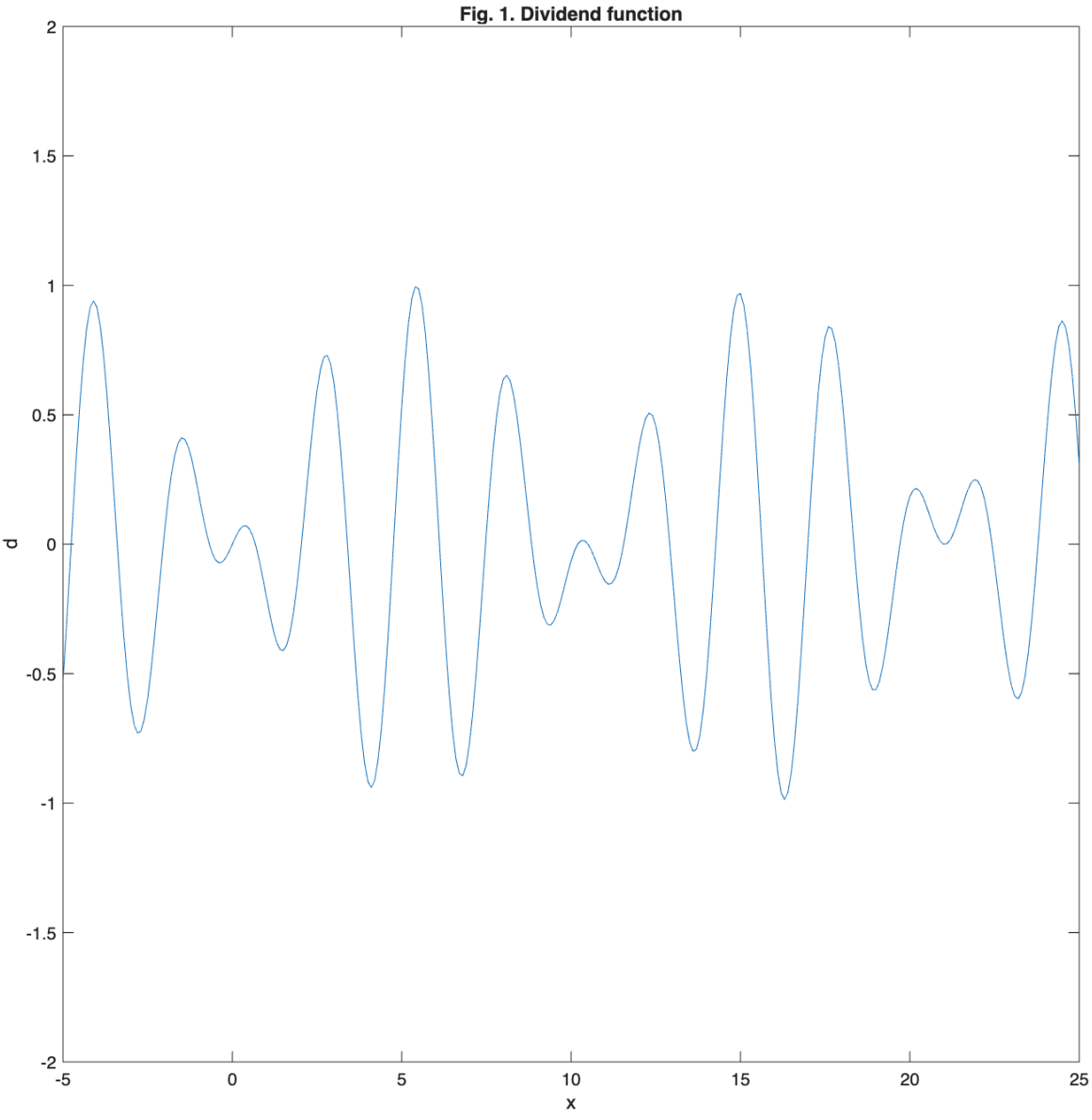


Fig. 1. Dividend function
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
d
-5
0
5
10
15
20
25
x

5. Create the trajectory of the dividend random variable

In the MATLAB code, a sample of 101 values is drawn from a Gamma distribution with shape parameter 1.2 and scale parameter 0.5. The Gamma distribution is commonly used to model non-negative skewed data, making it suitable for financial quantities like dividends. These sampled values are then passed into the previously defined dividend function `d_fun`, and the absolute value of the output is taken to ensure all dividend realizations are non-negative. The resulting values are plotted against a sequence from -5 to 5, with labeled axes and a title. The spikes in the graph reflect the variability introduced by the random sampling, capturing the irregular nature of dividend payments over time.

```
%% 5. Create and visualize the trajectory of the dividend random variable

% 5.1. Draw samples from Gamma distribution
X = random('Gamma',1.2,0.5,[1 101]);

% 5.2. Evaluate dividend realizations
d_X = abs(d_fun(X));

% 5.3. Create figure and plot dividend realizations
figure
plot(-5:0.1:5,d_X)
title('Fig. 2. Trajectory of the dividend random variable');
xlabel('Sample index');
ylabel('Dividend (currency units)');
```

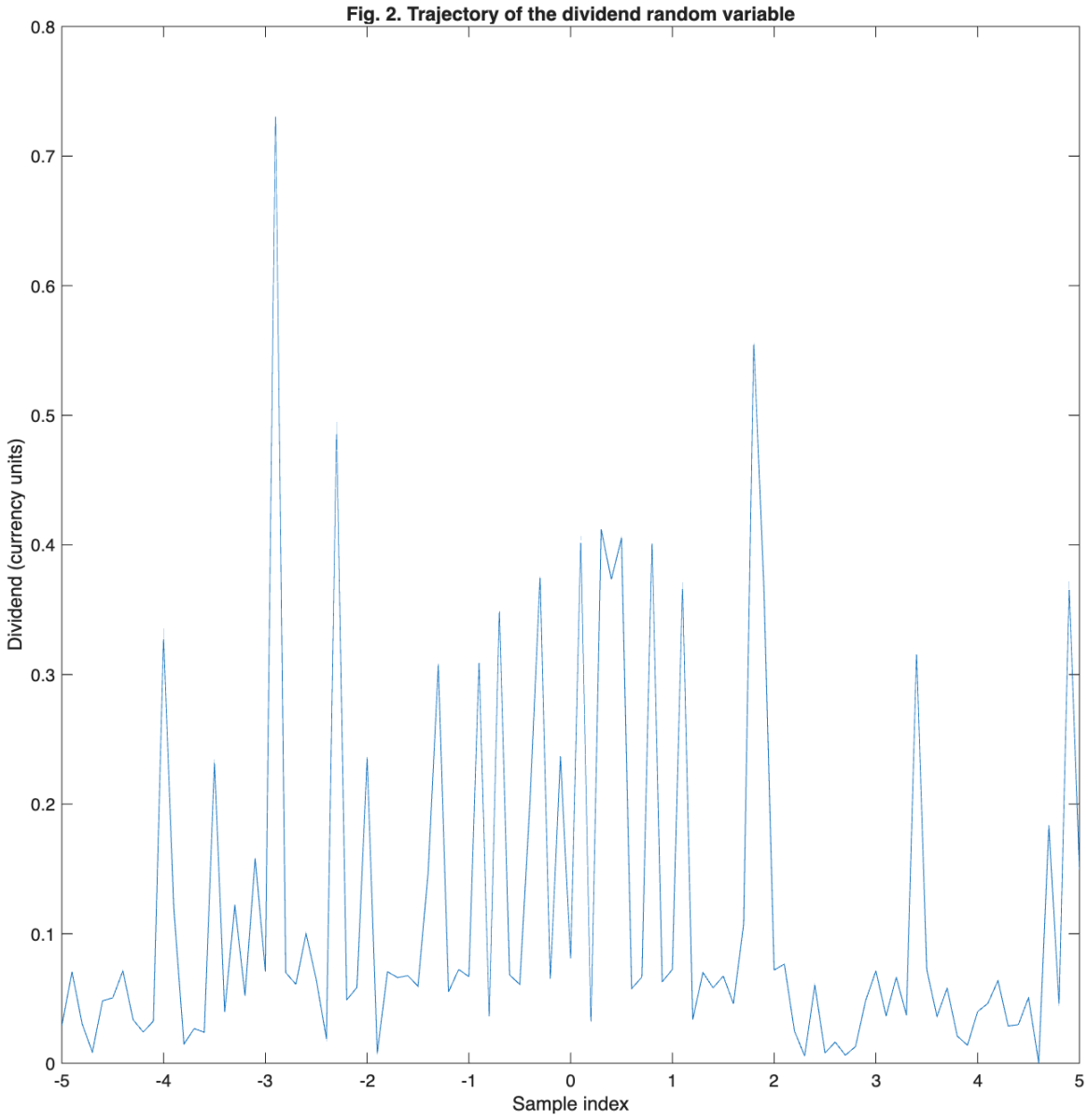


Fig. 2. Trajectory of the dividend random variable
Dividend (currency units)
Sample index
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
-5
-4
-3
-2
-1
0
1
2
3
4
5

6. Compute the weighted importance sampling estimate

The following code computes the weighted importance sampling estimate $\hat{\theta}_{IS,w}$ and its sample variance for a range of sample sizes. The proposal distribution is set as an exponential distribution with mean 0.5, corresponding to a rate parameter of 2. A weight function $w(x) = 2x^{0.2}$ is defined to adjust the contribution of each sample. The scaled dividend function $g(x) = \frac{1}{R}|d(x)|$, with $R = 1.05$, is also defined using the previously introduced dividend function $d(x)$. A vector of sample sizes ranging from 1000 to 100,000 in increments of 1000 is created. For each sample size, the code draws samples from the exponential distribution, evaluates the scaled dividend function and the weights, normalizes the weights to sum to one, and computes the weighted importance sampling estimate. The sample variance of each estimate is then calculated using the normalized weights and stored for analysis. This procedure allows for studying the behavior and stability of the importance sampling estimator across increasing sample sizes.

```
%% 6. Compute the weighted importance sampling estimate

% 6.1. Set proposal distribution: Exponential with rate 2 (mean = 0.5)
proposal_mean = 0.5;

% 6.2. Define weight function
w_fun = @(x) 2.*x.^0.2;

% 6.3. Define scaled dividend function g(x) = (1/R)*|d(x)|
g_fun = @(x) (1/1.05).*abs(d_fun(x));

% 6.4. Create vector of sample sizes
N_sizes = 1000:1000:100000;

% 6.5. Preallocate vector for importance sampling estimates
weighted_is_estimate = zeros(size(N_sizes));

% 6.6. Preallocate vector for sample variances of estimates
sample_variance_expo = zeros(size(N_sizes));

% 6.7. Loop over each sample size to compute IS estimate and variance
for i = 1:length(N_sizes)

    % 6.7.1. Set current sample size
    N = N_sizes(i);

    % 6.7.2. Draw N samples from Exponential(mean = 0.5)
    Xp = random('Exponential',proposal_mean,[N 1]);

    % 6.7.3. Evaluate scaled dividend function g(X_i)
    g_vals = g_fun(Xp);

    % 6.7.4. Compute importance weights w(X_i)
    w_vals = w_fun(Xp);

    % 6.7.5. Normalize weights to sum to 1
```

```
    w_norm = w_vals/sum(w_vals);

    % 6.7.6. Compute weighted importance sampling estimate
    weighted_is_estimate(i) = sum(w_norm.*g_vals);

    % 6.7.7. Compute sample variance of the IS estimate
    sample_variance_expo(i) = (N/(N-1))* ...
        sum((w_norm.^2).*(g_vals-weighted_is_estimate(i)).^2);
end
```

7. Visualize convergence behavior of importance sampling estimate

Figure 3 shows the weighted importance sampling estimates over an increasing sample size. At first glance, it seems that the estimates deviate a lot, but the variance is relatively small.

```
%% 7. Visualize convergence behavior of importance sampling estimate

% Create figure and plot IS estimates across sample sizes
figure
plot(N_sizes,weighted_is_estimate,'DisplayName','IS estimate')
title('Fig. 3. Convergence of the IS estimate')
xlabel('Sample size')
ylabel('Estimated price (currency units)')
legend('show')
```

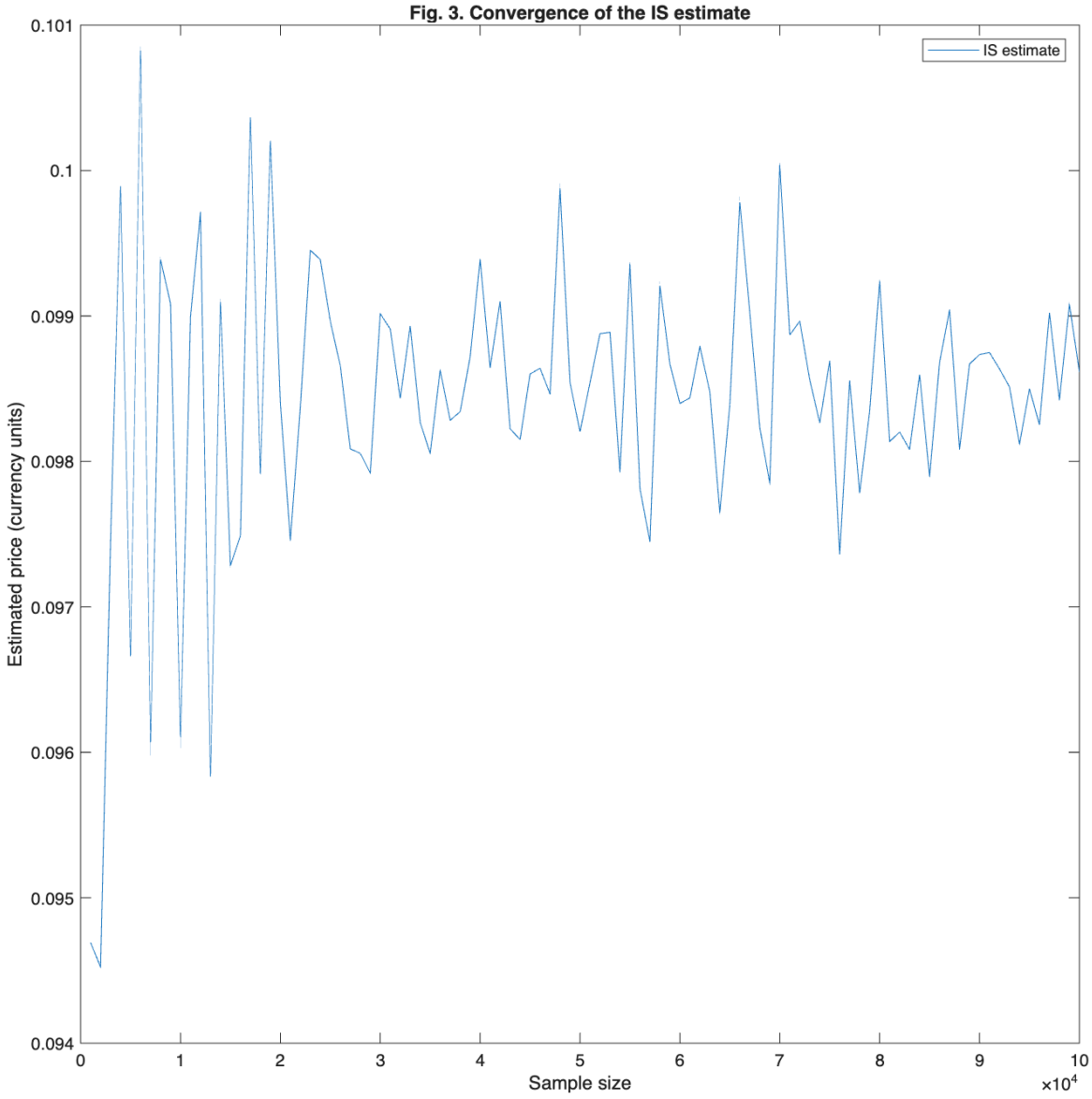


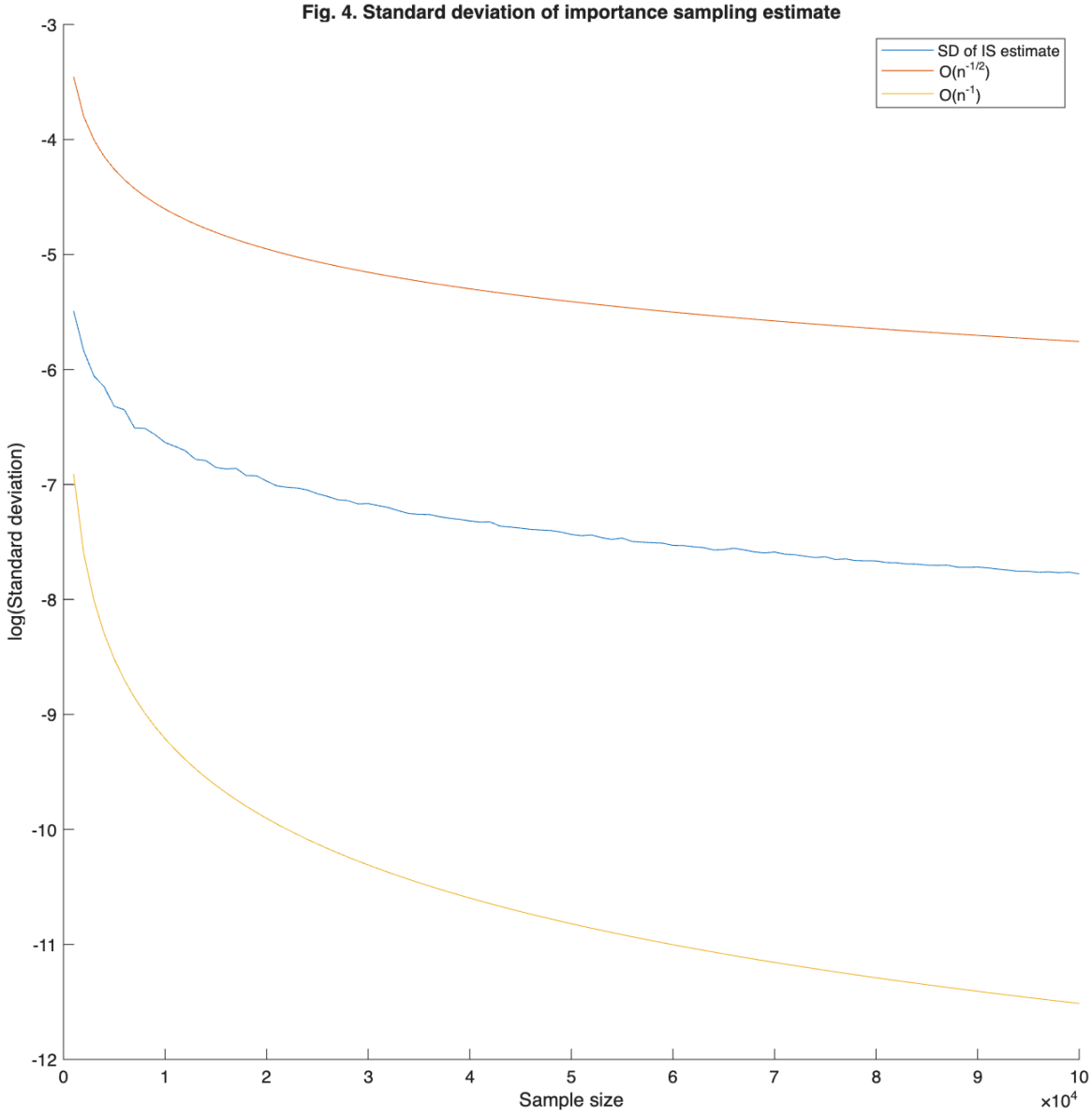
Fig. 3. Convergence of the IS estimate
IS estimate
Estimated price (currency units)
0.101
0.1
0.099
0.098
0.097
0.096
0.095
0.094
0
1
2
3
4
5
6
7
8
9
10
Sample size
×10^4

8. Visualize standard deviation behavior of importance sampling estimate

To assess the efficiency of the weighted importance sampling estimator $\hat{\theta}_{IS,w}$, the MATLAB code compares the empirical behavior of its sample standard deviation against two theoretical convergence rates. Specifically, it computes the logarithm of the sample standard deviation using the precomputed sample variances, and contrasts this with the logarithms of the rates $\mathcal{O}(n^{-1/2})$ and $\mathcal{O}(n^{-1})$, which represent the standard convergence rate of Monte Carlo estimators and a faster benchmark, respectively. These quantities are plotted on a logarithmic scale to highlight their relative decay as the sample size increases. As shown in Figure 4, the sample standard deviation of $\hat{\theta}_{IS,w}$ decreases more rapidly than the $\mathcal{O}(n^{-1/2})$ rate, indicating that the importance sampling method used here achieves superior variance reduction.

```
%% 8. Visualize std. dev. behavior of importance sampling estimate

% 8.1. Compute log of sample std. dev. from IS estimates
log_sd = log(sqrt(sample_variance_expo));

% 8.2. Compute log of theoretical O(n^{-1/2}) convergence rate
log_n_half = log(1./sqrt(N_sizes));

% 8.3. Compute log of theoretical O(n^{-1}) convergence rate
log_n_inv = log(1./N_sizes);

% 8.4. Create figure for log-transformed standard deviation comparison
figure
hold on
plot(N_sizes,log_sd,'DisplayName','SD of IS estimate');
plot(N_sizes,log_n_half,'DisplayName','O(n^{-1/2})');
plot(N_sizes,log_n_inv,'DisplayName','O(n^{-1})');
title('Fig. 4. Standard deviation of importance sampling estimate');
xlabel('Sample size');
ylabel('log(Standard deviation)');
legend('show');
hold off
```

Fig. 4. Standard deviation of importance sampling estimate



11. Final notes

 We have used the following reference. Luenberger, D. G., 2013. Investment science. Oxford: Oxford University Press.