Variance Reduction Methods in Monte Carlo Methods

Ahmer Nadeem Khan

12 April 2025

1 Abstract

We derive and prove certain interesting results that complement results shown in class about variate-reduction and look at an easy example initially. We eventually investigate numerically the effectiveness of a European Call Option as a Control variate for pricing Asian Call options using GBM simulation on Julia. The results, graphs and output of the experiments and code is available at the end.

2 Problems and Experiments

Problem 1. Consider the Bernoulli probability mass function

$$f(x) = p^x(1-p)^{1-x}; \quad x = 0, 1$$

Compute the titled mass function $f_t(x)$ for the Bernoulli probability mass function.

The titled mass is defined as

$$f_t(x) = \frac{e^{tx} f(x)}{M(t)}$$

where M is the moment generating function of the original mass function.

Note that we can write the Bernoulli mass function more simply using indicators:

$$f(x) = p\,\mathbb{1}_1(x) + (1-p)\,\mathbb{1}_0(x)$$

(this also extends the domain to $\mathbb{R}$ naturally, as is required formally). Then we first compute the moment generating function

$$M(t) = \mathbb{E}\, e^{tX} = p\, e^{t(1)} + (1-p)\, e^{t(0)} = pe^t + 1 - p$$

then we plug into the formula for the tilted mass function to get

$$\begin{aligned} f_t(x) &= \frac{e^{tx}}{pe^t + 1 - p}(p\,\mathbb{1}_1(x) + (1-p)\,\mathbb{1}_0(x)) \\ &= \frac{pe^{tx}}{pe^t + 1 - p}\,\mathbb{1}_1(x) + \frac{(1-p)e^{tx}}{pe^t + 1 - p}\,\mathbb{1}_0(x) \end{aligned}$$

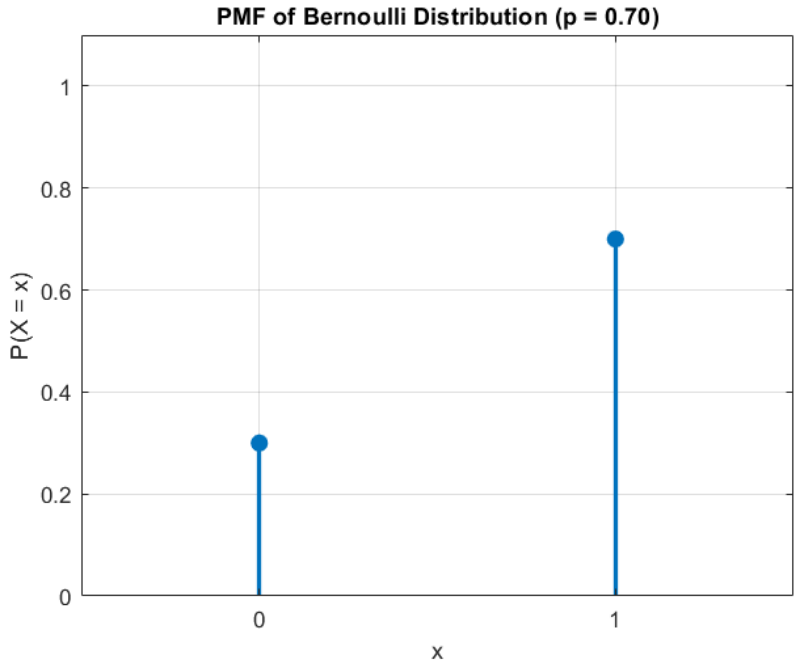
which due to the indicators, we can substitute the values 1 and 0 into the relevant terms since these are the only values of x that have potentially non-zero contributions:

$$f_t(x) = \frac{pe^t}{pe^t + 1 - p}\,\mathbb{1}_1(x) + \frac{(1-p)}{pe^t + 1 - p}\,\mathbb{1}_0(x)$$

Set q_1, q_0 as the values indicated at 1 and 0 respectively. We can see that this simply re-weights the probability masses at 1 and 0 to new probabilities, by noting that

$$q_1 + q_0 = \frac{pe^t}{pe^t + 1 - p} + \frac{(1-p)}{pe^t + 1 - p} = \frac{pe^t + (1-p)}{pe^t + 1 - p} = 1$$

and $0 \leq q_1, q_0 \leq 1$ is clear from each term. Then we again have a unit probability measure, and therefore a new Bernoulli mass function on the same support. We can see an example as such:



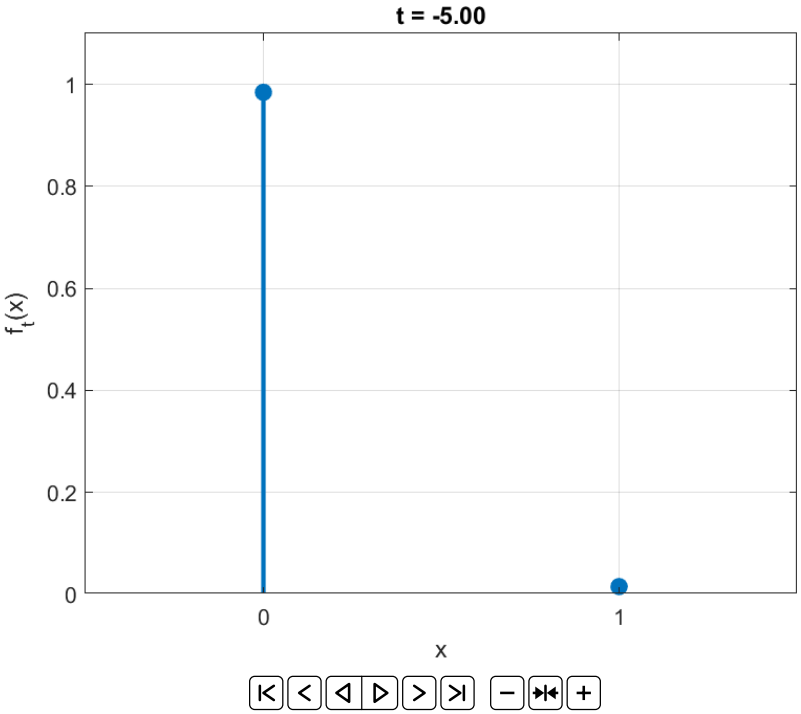


Figure 1: Tilted Bernoulli PMF changing with $t \in [-5, 5]$ (p = 0.7)

Note: Use Adobe Acrobat (or a PDF viewer with JavaScript functionality) to see the animation!

Problem 2. Prove the identity

$$\sum_{i,j} \min(t_i, t_j) = \sum_{i=1}^{n} (2n - (2i-1))\ t_i$$

for $1 \leq i, j \leq n$, and assuming the order on the t_i values i.e.

$$t_1 < t_2 < \ldots < t_n$$

We follow a constructive proof assuming that the matrix C such that $c_{ij} = \min(t_i, t_j)$ is the covariance matrix of the normal multivariate vector $\mathbf{W} = (W(t_1), W(t_2), \ldots, W(t_n))^\top$, and that the covariance matrix of AW is $ACA^\top$, where A is any $k \times n$ matrix (stated without proof). Then consider

$A = [1, 1, \ldots, 1] = \mathbf{1}^\top$

$$\begin{aligned} ACA^\top &= \mathbf{1}^\top C\mathbf{1} = \mathbf{1}^\top C(\mathbf{e}_1 + \mathbf{e}_2 + \ldots + \mathbf{e}_n) \\ &= \mathbf{1}^\top (C\mathbf{e}_1 + C\mathbf{e}_2 + \ldots + C\mathbf{e}_n) \\ &= \mathbf{1}^\top (\mathbf{c}_1 + \mathbf{c}_2 + \ldots + \mathbf{c}_n) \\ &= \mathbf{1}^\top \mathbf{c}_1 + \mathbf{1}^\top \mathbf{c}_2 + \ldots + \mathbf{1}^\top \mathbf{c}_n \end{aligned}$$

where $\mathbf{c}_j$ is the j-th column of C. Then we get that

$$\mathbf{1}^T \mathbf{c}_j = \sum_{i=1}^{n} c_{ij} = \sum_{i=1}^{n} \min(t_i, t_j)$$

Note that this is the sum of the entries of the j-th column. Then, in the full sum (i.e. when we sum over j) we get exactly the required sum i.e.

$$\sum_{i,j} \min(t_i, t_j) = \sum_{j=1}^{n} \sum_{i=1}^{n} \min(t_i, t_j) = \sum_{j=1}^{n} \sum_{i=1}^{n} c(t_i, t_j)$$

which is exactly what our construction gets us. Then, notice that for each j, we get $t_1, t_2, \ldots t_{j-1} < t_j$ and $t_j, t_{j+1}, t_{j+2}, \ldots, t_n \geq t_j$. To see this more clearly, we can write the full sum down more explicitly.

$$\begin{aligned} &= \mathbf{1}^\top \mathbf{c}_1 + \mathbf{1}^\top \mathbf{c}_2 + \ldots + \mathbf{1}^\top \mathbf{c}_n \\ &= \sum_{i=1}^{n} \min(t_i, t_1) + \sum_{i=1}^{n} \min(t_i, t_2) + \sum_{i=1}^{n} \min(t_i, t_3) + \ldots + \sum_{i=1}^{n} \min(t_i, t_{n-1}) + \sum_{i=1}^{n} \min(t_i, t_n) \end{aligned}$$

and then we can expand each of these terms explicitly. Take the first two terms as an example:

$$\begin{aligned} \sum_{i=1}^{n} \min(t_i, t_1) &= \min(t_1, t_1) + \min(t_2, t_1) + \min(t_3, t_1) + \ldots + \min(t_{n-1}, t_1) + \min(t_n, t_1) \\ &= t_1 + t_1 + \ldots + t_n \\ &= \sum_{i=1}^{n} t_1 = n\, t_1 \end{aligned}$$

and the second term becomes

$$\begin{aligned} \sum_{i=1}^{n} \min(t_i, t_2) &= \min(t_1, t_2) + \min(t_2, t_2) + \min(t_3, t_2) + \ldots + \min(t_{n-1}, t_2) + \min(t_n, t_2) \\ &= t_1 + (t_2 + \ldots + t_2) \\ &= t_1 + \sum_{i=2}^{n} t_2 = t_1 + (n-1)\, t_2 \end{aligned}$$

The min function is symmetric (which is also why the covariance matrix is symmetric, as it ought to be) and the pattern persists for each term. This follows from the previous order on t values. For the term involving t_j, the first $1 \leq k \leq j-1$ terms are such that $t_k < t_j$, but for $j \leq k \leq n$ we get that $t_k \geq t_j$, where for $j = 1$, we get no terms in the first part and for $j = n$, we get no terms in the second part. For the j-th term, we get

$$\sum_{i=1}^{n} \min(t_i, t_j) = (t_1 + t_2 + \ldots + t_{j-1}) + (n-(j-1))\, t_j = \left(\sum_{k=1}^{j-1} t_k\right) + (n-(j-1))\, t_j$$

The full sum is obtained by summing over all j values i.e. $1 \leq j \leq n$. We can then collect like terms for each t_i value in the full sum:

$$\sum_{j=1}^{n} \left(\sum_{k=1}^{j-1} t_k + (n-(j-1))\, t_j\right)$$

which becomes

$$= \sum_{j=1}^{n} \sum_{k=1}^{j-1} t_k + \sum_{j=1}^{n} (n-(j-1))\, t_j$$

$$= \sum_{j=1}^{n} (t_1 + t_2 + \ldots t_{j-1}) + \sum_{j=1}^{n} (n-(j-1))\, t_j$$

$$= (n-1)\, t_1 + (n-2)\, t_2 + \ldots (n-(n-1))\, t_{n-1} + (n-n)\, t_n + \sum_{j=1}^{n} (n-(j-1))\, t_j$$

since in the first part, each t_j appears $n-j$ times (because of the sum up to t_{j-1}). The collecting terms we get

$$= \sum_{j=1}^{n} (n-(j-1) + (n-j))\ t_j$$

$$= \sum_{j=1}^{n} (2n-(2j-1))\ t_j$$

Replacing the dummy variable j with i completes the proof. Note that this is the variance of the Normal random variable which is the sum of the components of $\mathbf{W}$. Note that since we eventually sum over all entries of the matrix C, there are multiple ways to prove this identity. The matrix C in general looks like:

$$C = \begin{bmatrix} t_1 & t_1 & t_1 & \cdots & t_1 & \cdots & \cdots & t_1 \\ t_1 & t_2 & t_2 & \cdots & t_2 & \cdots & \cdots & t_2 \\ t_1 & t_2 & t_3 & \cdots & t_3 & \cdots & \cdots & t_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & & \vdots & \\ t_1 & t_2 & t_3 & \cdots & t_{j-1} & t_j & \cdots & t_j \\ \vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ t_1 & t_2 & t_3 & \cdots & \cdots & t_{n-2} & t_{n-1} & t_n \end{bmatrix}$$

Problem 3. Consider $Y(\beta) = Y - \beta(C - \mu_C)$, the control variate estimator, and let β^* be the optimal variance minimizing parameter. Let $\hat{\beta}^*$ be an approximation of β^*, such that $\beta^* = \hat{\beta}^* + \epsilon$, where ϵ is a small parameter. Prove that

$$\text{Var}(Y(\hat{\beta}^*)) = \text{Var}(Y(\beta^*)) + \epsilon^2 \text{Var}(C).$$

We first get the two equations

$$Y(\beta^*) = Y - \beta^*(C - \mu_C) \tag{1}$$
$$Y(\hat{\beta}^*) = Y - \hat{\beta}^*(C - \mu_C) \tag{2}$$

Then we use (1) and make the substitution $\beta^* = \hat{\beta}^* + \epsilon$ to get

$$\begin{aligned} Y(\beta^*) &= Y - \beta^*(C - \mu_C) \\ &= Y - (\hat{\beta}^* + \epsilon)(C - \mu_C) \\ &= Y(\hat{\beta}^*) - \epsilon(C - \mu_C) \end{aligned}$$

which gives us the equation

$$Y(\hat{\beta}^*) = Y(\beta^*) + \epsilon\, \overline{C} \tag{3}$$

where $\overline{C} = C - \mu_C$. Since this is just the de-meaned variable C, it has the same variance as C (we only subtract the constant μ_C) i.e. $\text{Var}(\overline{C}) = \text{Var}(C)$. Then we apply the Var operator to equation (3) to get

$$\begin{aligned} \text{Var}(Y(\hat{\beta}^*)) &= \text{Var}(Y(\beta^*) + \epsilon\, \overline{C}) \\ &= \text{Var}(Y(\beta^*)) + \text{Var}(\epsilon\, \overline{C}) + 2\,\text{Cov}(Y(\beta^*), \epsilon\, \overline{C}) \\ &= \text{Var}(Y(\beta^*)) + \epsilon^2\, \text{Var}(C) + 2\,\text{Cov}(Y(\beta^*), \epsilon\, \overline{C}) \end{aligned}$$

where we have used expectation algebra for the variance operator, and the fact that $\text{Var}(\overline{C}) = \text{Var}(C)$. We then only have to show that

$$\text{Cov}(Y(\beta^*), \epsilon\, \overline{C}) = 0$$

We use the fact that Cov is a bilinear operator, and use the relevant properties (assuming all values are real constants or real valued r.v.'s).

$$\begin{aligned} \text{Cov}(Y(\beta^*), \epsilon\, \overline{C}) &= \text{Cov}(Y - \beta^* \overline{C}, \epsilon\, \overline{C}) \\ &= \text{Cov}(Y, \epsilon\, \overline{C}) - \text{Cov}(\beta^* \overline{C}, \epsilon\, \overline{C}) \\ &= \epsilon\, \text{Cov}(Y, \overline{C}) - \epsilon \beta^*\, \text{Cov}(\overline{C}, \overline{C}) \\ &= \epsilon\, (\text{Cov}(Y, \overline{C}) - \beta^*\, \text{Var}(\overline{C})) \\ &= \epsilon\, (\text{Cov}(Y, C) - \beta^*\, \text{Var}(C)) \end{aligned}$$

where we used the fact that $\text{Cov}(Y, \overline{C}) = \text{Cov}(Y, C - \mu_C) = \text{Cov}(Y, C) - \text{Cov}(Y, \mu_c) = \text{Cov}(Y, C)$ since μ_C is a constant and has 0 variance, and the corresponding fact for Var. Now, β^* is defined as

$$\beta^* = \frac{\text{Cov}(Y, C)}{\text{Var}(C)}$$

and was shown to be the minimizer in the notes. Substituting this yields

$$\text{Cov}(Y, C) - \beta^* \text{Var}(C) = \text{Cov}(Y, C) - \frac{\text{Cov}(Y, C)}{\text{Var}(C)} \text{Var}(C) = \text{Cov}(Y, C) - \text{Cov}(Y, C) = 0$$

which completes the proof. This is in fact one of the reasons for this choice for B^* (it has a similar use as the *beta* in market factor models).

Problem 4. Investigate numerically if a European call option can be used effectively as a control for pricing an Asian call option.

We implement this in Julia in the file *options.ji* using code provided in the lecture notes and modifying it a little. The original code used Geometric Asian Options as a Control for pricing Arithmetic Asian options, so we simply remove the calculation for the Asian Geometric estimates and add European Call estimation (which simply uses the final stock price), and use the Black-Scholes value as the mean for our control C. We then present a our results:

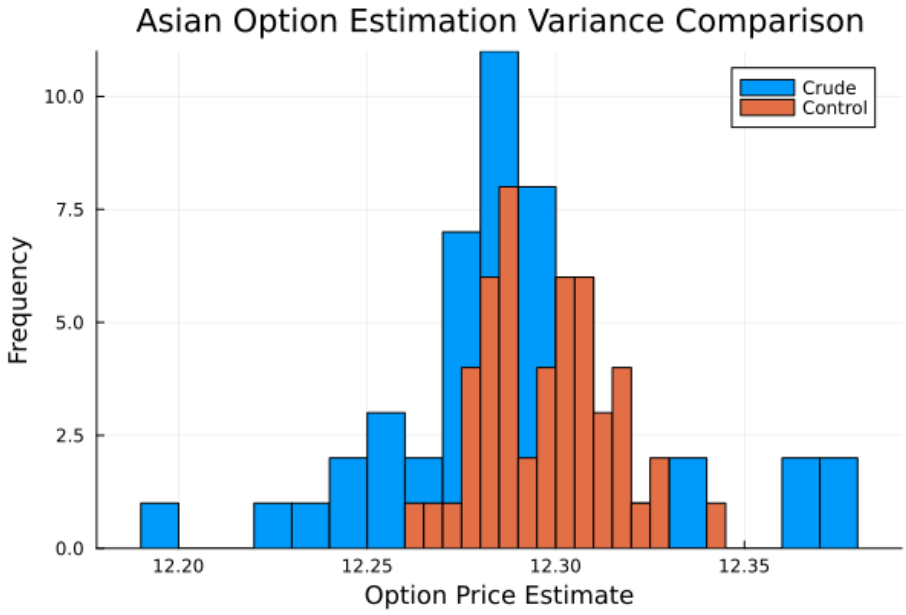


Figure 2: Smaller spread for Control vs Crude

The terminal outputs look like this for each experiment and display the variances, the variance reduction factors (VRF) and also the time for the computations:

Output Run 1:

```
Running simulations for variance comparison...

1.737150 seconds (162.36 k allocations: 7.856 MiB, 14.60% compilation time)
2.509036 seconds (35.21 M allocations: 621.028 MiB, 6.14% gc time,
                0.22% compilation time)

--- Variance Comparison Results ---
Variance of Crude Estimates:    0.0012371939281244544
Variance of Control Estimates:  0.000261594578094135
Variance Reduction Factor:      4.73x
```

Output Run 2:

```
Running simulations for variance comparison...

1.229044 seconds (69.08 k allocations: 3.478 MiB, 6.49% compilation time)
2.484037 seconds (35.15 M allocations: 617.907 MiB, 15.24% gc time,
                7.39% compilation time)

--- Variance Comparison Results ---
Variance of Crude Estimates:    0.002033757423848868
Variance of Control Estimates:  0.00027143779339439816
Variance Reduction Factor:      7.49x
```

We can conclude from these experiments that the control pricing takes about 2x as long as the crude method, so it is about **half as efficient** and achieves a small level of variance reduction. It is not nearly as big as get for some examples in the lecture notes, but still desirable. For example, the second run shows that we can sometimes get reductions factors up to the order of 10 ($\approx 7, 8$).

For more thorough testing, we can run a 2-D grid sweep on different values of the strike price K and the volatility of the underlying σ. I first display the table for the various values, and then visualize the results more concisely in a heatmap.

Table:

σ	K	Crude Variance	Control Variance	Variance Reduction
0.1	80	0.000363	9.2e-5	3.94x
0.1	90	0.00034	9.3e-5	3.64x
0.1	100	0.000136	6.7e-5	2.02x
0.1	110	1.3e-5	8.0e-6	1.58x
0.2	80	0.001403	0.000347	4.04x
0.2	90	0.001084	0.000317	3.42x
0.2	100	0.000798	0.000185	4.31x
0.2	110	0.00021	6.0e-5	3.49x
0.3	80	0.003324	0.000922	3.61x
0.3	90	0.001855	0.000679	2.73x
0.3	100	0.001048	0.000377	2.78x
0.3	110	0.000792	0.000295	2.69x
0.4	80	0.005453	0.001049	5.2x
0.4	90	0.003876	0.001246	3.11x
0.4	100	0.004026	0.000987	4.08x
0.4	110	0.001384	0.000973	1.42x

Heatmap:

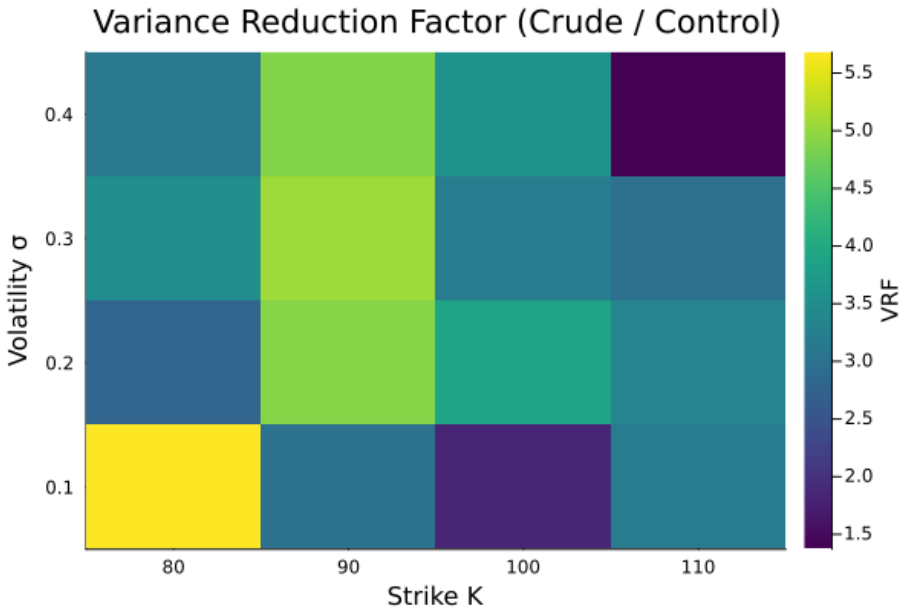


Figure 3: Heatmap for different parameters

We can conclude that the average VRF is ≈ 3, and generally lower strike prices mean more reduction (which we saw in the lecture notes for other ex-

amples). We do not get very massive variance reductions, so while there is some variance reduction, in practice the loss of efficiency is perhaps not worth it, depending on the application. **Therefore, European Calls do act as a possible control for pricing Arithmetic Asian Calls, but are not very effective in general**.

Code Appendix

Option Pricing and Control Variate Reduction

```
using Distributions
using Plots
using Statistics

stdn=Normal(0,1)

phi(x)=cdf(stdn,x);

function bsm(r,sigma,S,K,T)
    d1=(log(S/K)+(r+sigma^2/2)*T)/(sigma*sqrt(T))
    d2=d1-sigma*sqrt(T)
    S*phi(d1)-K*exp(-r*T)*phi(d2)
end

function AsianArithPriceCrude(r,sigma,szero,K,T,n,N)
    # n is the number of time steps, h=T/n
    h=T/n
    price=0
    for i in 1:N
        sum=0
        s=szero
        for j in 1:n
            s=s*exp((r-sigma^2/2)*h+sigma*sqrt(h)*randn())
            sum=sum+s
        end
        price=price+max(sum/n-K,0)
    end
    exp(-r*T)*price/N
end

function EuroCallPriceCrude(r,sigma,szero,K,T,N)
    sum=0
    for i in 1:N
        sfinal=szero*exp((r-sigma^2/2)*T+sigma*sqrt(T)*randn())
        sum=sum+max(sfinal-K,0)
```

```
    end
    return price=exp(-r*T)*sum/N
    println("Crude MC call option price: $price")
end



function AsianArithPriceControl(r, sigma, szero, K, T, n, N)
    # n is the number of time steps, h = T/n
    h = T / n
    bsp = bsm(r, sigma, szero, K, T)

    prsum = 0
    csum = 0
    ysum = 0
    price = 0
    y = Array{Float64}(undef, N)
    c = Array{Float64}(undef, N)

    for i in 1:N
        sum = 0
        s = szero
        for j in 1:n
            s = s * exp((r - sigma^2 / 2) * h + sigma * sqrt(h) * randn())
            sum += s
        end

        y[i] = exp(-r * T) * max(sum / n - K, 0)
        c[i] = exp(-r * T) * max(s - K, 0)
    end

    ybar = sum(y) / N
    cbar = sum(c) / N

    for i in 1:N
        prsum += (y[i] - ybar) * (c[i] - cbar)
        csum += (c[i] - cbar)^2
        ysum += (y[i] - ybar)^2
    end

    beta = prsum / csum
    rho = prsum / (sqrt(csum) * sqrt(ysum))

    for i in 1:N
        price += y[i] - beta * (c[i] - bsp)
    end
```

```
    return price / N
end


println("\nRunning simulations for variance comparison...\n")

@time CrudeEstimates = [AsianArithPriceCrude(0.035, 0.2, 100, 90, 1, 20, 100000)
                        for _ in 1:50];
@time ControlEstimates = [AsianArithPriceControl(0.035, 0.2, 100, 90, 1, 20, 100000)
                        for _ in 1:50];

var_crude = var(CrudeEstimates)
var_control = var(ControlEstimates)
vrf = var_crude / var_control

println("\n--- Variance Comparison Results ---")
println("Variance of Crude Estimates:    $var_crude")
println("Variance of Control Estimates:  $var_control")
println("Variance Reduction Factor:      $(round(vrf, digits=2))x")
```